INSTRUMENTACION ELECTRONICA



MICROINSTRUCTOR

MI-650C

PROMAX

OIF 7020

INDICE

INTRODUCCION..... 1
CARACTERISTICAS TECNICAS DEL MICROINSTRUCTOR..... 2

1. INTRODUCCION A LOS MICROCOMPUTADORES..... 5
1.1 DESCRIPCION DEL MICROCOMPUTADOR..... 5
1.2 SISTEMAS DE NUMERACION..... 8
1.3 REPRESENTACIONES NUMERICAS..... 11
1.3.1 Números sin signo..... 11
1.3.2 Números con signo..... 12
1.3.3 Números Reales..... 14
1.3.4 Números decimales..... 15
1.3.5 Representaciones alfanuméricas..... 16
PRACTICA 1.1 PUESTA EN MARCHA Y UTILIZACION..... 18

2. DESCRIPCION DE LOS ELEMENTOS DE UN MICROCOMPUTADOR..... 21
2.1 MEMORIA DEL MICROCOMPUTADOR..... 21
2.1.1 Tipos de memoria..... 21
2.1.2 Función de la memoria en el microcomputador..... 26
2.1.3 Direccionamiento de las memorias..... 27
2.1.4 Ejemplo de Decodificación..... 30
2.2 EL MICROPROCESADOR..... 32
2.2.1 Estructura del microprocesador..... 32
2.2.2 Proceso Básico de Ejecución de un Programa..... 34
2.2.3 La Unidad de Control..... 35
2.2.4 La Unidad Aritmético-Lógica (ALU)..... 36
2.2.5 El Acumulador..... 38
2.2.6 Registros Auxiliares..... 39
2.2.7 Registro de Estado ó Status..... 39
2.2.8 Estados o ciclos de la CPU..... 40
2.2.9 Subrutinas..... 42
2.2.10 Interrupciones..... 45
2.3 ELEMENTOS DE ENTRADA/SALIDA..... 48
2.3.1 Transferencia en paralelo..... 48
2.3.2 Transferencia en serie..... 50
PRACTICA 2.1 ACTUACION DE LAS INTERRUPCIONES..... 55

3. EL MICROINSTRUCTOR MI-650C..... 57
3.1 LA MEMORIA DEL MICROINSTRUCTOR..... 57
3.2 EL MICROPROCESADOR 65C02..... 62
3.2.1 Introducción..... 62
3.2.2 Estructura hardware interna..... 63
3.2.3 Modos de direccionamiento..... 71

3.2.4 Registro Status del 65C02..... 77
3.2.5 Juego de instrucciones..... 78
3.3 ELEMENTOS ASOCIADOS A LA CPU EN EL MICROINSTRUCTOR..... 86
3.4 ELEMENTOS DE ENTRADA-SALIDA DEL MICROINSTRUCTOR..... 91
3.4.1 Entradas/salidas de usuario: la VIA 65C02..... 91
3.4.2 Entradas/salidas locales: teclado y display..... 95
3.4.3 Cassette..... 97
3.5 ALIMENTACION..... 103
PRACTICA 3.1 DECODIFICADOR DEL MICROINSTRUCTOR.... 104
PRACTICA 3.2 DECODIFICACION DE LOS DIGITOS DEL DISPLAY..... 106

4. UTILIZACION DEL MICROINSTRUCTOR..... 109
4.1 INICIALIZACION DEL MICROINSTRUCTOR..... 110
4.2 INSPECCION Y ALTERACION DE MEMORIA..... 112
4.2.1 Visualización del contenido de la memoria..... 112
4.2.2 Introducción de datos en la memoria..... 113
4.2.3 Corrección de errores..... 114
4.3 PROTECCION DE LA RAM CONTRA ESCRITURA..... 117
4.4 EJECUCION DE UN PROGRAMA DE USUARIO..... 118
4.4.1 Modo continuo (RUN)..... 118
4.4.2 Modo paso a paso..... 119
4.5 VISUALIZACION DE LOS REGISTROS DE LA CPU..... 121
4.6 VECTORES..... 127
4.6.1 Inicialización de vectores..... 127
4.6.2 Programa vectorizado..... 128
4.7 PUNTO DE RUPTURA (BREAKPOINT)..... 131
4.8 CALCULO DEL OFFSET..... 136
4.9 UTILIZACION DEL CASSETTE..... 139
4.9.1 Control del motor..... 139
4.9.2 Almacenamiento de datos en cassette.. 140
4.9.3 Verificación de los datos grabados... 141
4.9.4 Lectura de los datos del cassette.... 142
PRACTICA 4.1 UTILIZACION DE LAS LINEAS DE ENTRADA/SALIDA..... 144
PRACTICA 4.2 SIMULACION DE UNA PUERTA AND..... 146

5. TECNICAS DE PROGRAMACION..... 149
5.1 LENGUAJES DE PROGRAMACION..... 149

5.2 LENGUAJE ENSAMBLADOR..... 153
5.3 METODOLOGIA EN EL DISEÑO DEL SOFTWARE..... 157
5.4 DISEÑO DE UN PROGRAMA..... 160
5.5 TABLAS DE CONVERSION. ALGORITMOS MATEMATICOS. 167

6. DESCRIPCION DEL SOFTWARE DEL MICROINSTRUCTOR..... 169
6.1 EL PROGRAMA MONITOR..... 169
6.1.1 Inicialización y bucle de reposo..... 170
6.1.2 Atención a las interrupciones..... 172
6.1.3 Control de entradas/salidas..... 176
6.2 RUTINAS DE ENTRADA/SALIDA..... 177
PRACTICA 6.1 MENSAJE MOVIL EN DISPLAY..... 185

7. PRACTICAS..... 189
7.1 INTRODUCCION..... 189
7.2 MOVIMIENTO DE DATOS ENTRE ZONAS FIJAS..... 192
7.3 MOVIMIENTO DE DATOS ENTRE ZONAS VARIABLES.... 193
7.4 CONTADOR BINARIO UP-DOWN DE 8 BITS..... 196
7.5 ANALIZADOR LOGICO (0F419H-0F445H)..... 198
7.6 SUMA EN DOBLE PRECISION..... 201
7.7 RESTA EN DOBLE PRECISION (F446H-F454H)..... 203
7.8 MULTIPLICACION DE DOS NUMEROS SIN SIGNO..... 204
7.9 DIVISION DE DOS NUMEROS SIN SIGNO..... 206
7.10 RAIZ CUADRADA (F455H-F470H)..... 207
7.11 CALCULADORA..... 209
7.12 CONVERSION HEXADECIMAL-DECIMAL (F471H-F522H) 212
7.13 CONVERSION DECIMAL-HEXADECIMAL (F523H-F5AFH) 216
7.14 CRONOMETRO (F5B0H-F637H)..... 219
7.15 RELOJ HORARIO CONTINUO EN TIEMPO REAL (F638H- F6E5H)..... 224
7.16 ORGANO MUSICAL DE UNA OCTAVA..... 228
7.17 RECOLOCADOR (F334H - F3C9H)..... 232
7.18 JUEGO ADIVINE EL NUMERO (F6E6H-F76FH)..... 237

APENDICE A JUEGO DE INSTRUCCIONES DEL 65C02..... A-1

APENDICE B MODULOS DE APLICACION..... B-1

APENDICE C DIRECCIONES DE LOS PROGRAMAS..... C-1

APENDICE D LISTADO PROGRAMA MONITOR..... D-1

APENDICE E MANTENIMIENTO Y VERIFICACION..... E-1

APENDICE F ESQUEMAS..... F-1

INTRODUCCION

I.E.PROMAX,SA ha diseñado el Microinstructor MI-650C, que es un microcomputador basado en las versiones CMOS de la CPU de 8 bits 6502 (65SC02 - 65C02), como un equipo ideal para la introducción en el mundo de los microprocesadores. Pero el Microinstructor MI-650C no sólo es una excelente herramienta para el aprendizaje, sino que dadas sus características, puede emplearse, dotado del software adecuado, para el desarrollo de pequeños equipos de control autónomo. Con este fin I.E.PROMAX,SA ha implementado una familia de Módulos de Aplicación para el Microinstructor que permiten adquirir un conocimiento claro de las aplicaciones del microprocesador como elemento de control.

El Microinstructor MI-650C es un equipo imprescindible en la enseñanza de electrónica y control industrial ya que, al ser un equipo transparente para el usuario, éste podrá seguir y evaluar las operaciones que realiza la CPU con sus periféricos, de forma sencilla y eficaz.

Para facilitar el manejo y la comprensión del funcionamiento del Microinstructor MI-650C, I.E.PROMAX,SA ha editado esta documentación procurando que el texto se complemente con unas prácticas que resalten los aspectos más importantes de cada capítulo. De este modo, el seguimiento del manual puede convertirse en un completo cursillo sobre el microprocesador y sus aplicaciones, para el que no es necesario tener unos conocimientos amplios de Electrónica. El profesor podrá adecuar la lectura y las prácticas de los capítulos a las características de cada curso. Siguiendo las directrices del presente manual, se habrán adquirido, al termino de su lectura, los conocimientos básicos sobre los siguientes aspectos:

- Hardware y software que utilizan los microcomputadores.
- Código máquina y lenguaje ensamblador para el desarrollo y comprensión de programas.
- Diseño y mantenimiento de sistemas que utilizan el microprocesador como elemento de control.

CARACTERISTICAS TECNICAS DEL MICROINSTRUCTOR

Microprocesador:

65C02.

Clock:

1 MHz a cristal de cuarzo.

Memoria:

Dispone de una memoria RAM de 8 Kbytes y una memoria EPROM de 8 Kbytes. La memoria EPROM contiene el programa monitor, con una serie de programas de utilidades, los programas de ejemplo que aparecen el el último capítulo del manual del Microinstructor y los programas de ejemplo de los Módulos de Aplicación. Existe además un zócalo libre que puede ser ocupado por una memoria RAM o por una EPROM y tanto de 2 Kbytes como de 8 Kbytes, mediante la selección adecuada de unos puentes en los conectores disponibles sobre la placa de circuito impreso.

Depuración de programas:

El programa monitor permite la posibilidad de depurar programas mediante la utilización de puntos de ruptura, y la ejecución de programas paso a paso.

Display:

Visualización mediante 6 dígitos de 7 segmentos, 4 para la presentación de las direcciones y 2 para los datos.

Teclado:

22 teclas, 14 de las cuales tienen doble función.

Entradas/salidas:

El Microinstructor tiene dos conectores:

- **Conector de Expansión:** Permite el acceso a los buses del microprocesador a través de una serie de **Buffers.**

- **Conector de Aplicación:** Presenta 16 líneas programables cada una de ellas como entrada o como salida y 4 líneas de control.

- **Cassette:** Conector con control remoto de puesta en marcha para cassette standard de audio.

Alimentación:

110-125-220-240 V AC +/-10% 50-60 Hz.

Consumo:

13.5 W.

Dimensiones:

Ancho 220 x Alto 100 x Prof. 325 mm.

Peso:

3 Kgr.

Accesorios:

- Un conector macho tipo DIN 45329 de siete contactos.

- Un fusible de 0.25 A.

- Un cable de red.

- Un manual de introducción al Microinstructor MI-650C.

CAPITULO 1

INTRODUCCION A LOS MICROCOMPUTADORES

1.1 DESCRIPCION DEL MICROCOMPUTADOR

Un microcomputador es un equipo electrónico basado en un microprocesador como elemento principal de control, que es capaz de efectuar una secuencia de operaciones previamente definida, a la que se denomina programa. Aunque estas operaciones elementales efectuadas son muy simples, la gran velocidad de ejecución permite, por combinación de múltiples operaciones, la ejecución de funciones de un elevado grado de complejidad en un tiempo reducido.

El hecho de que la secuencia de operaciones se pueda alterar en función de las instrucciones contenidas en un programa, confiere una gran flexibilidad a estos elementos ya que se puede alterar radicalmente la función realizada por el microcomputador.

Por tanto, los microcomputadores se componen de un soporte físico que recibe el nombre de **circuito (hardware)** y una secuencia ordenada de instrucciones a ejecutar, que constituyen el **programa (software).**

El hardware condiciona y limita el campo de aplicación del equipo, siendo por tanto distinta la configuración de un equipo destinado al control industrial de procesos de la de uno destinado a gestión.

El software, por su parte, personaliza el equipo. Por ejemplo, si la configuración es la correspondiente a un controlador, mediante el software adecuado se puede destinar al control de robots, control de temperatura, etc.

Debido a que cada microprocesador tiene su propio repertorio de operaciones (instrucciones) que puede realizar, los programas escritos en un lenguaje directamente inteligible por el microprocesador han de ser destinados tanto a una determinada aplicación como a un equipo específico.

El hardware de un microcomputador puede dividirse en tres partes:

a. Un bloque de control encargado del tratamiento de la información y del control en general, cuyo elemento principal es el microprocesador o Unidad Central de Proceso (CPU).

b. Una memoria en la que se almacena en forma de código interpretable por el bloque de control, la secuencia de operaciones que ésta debe realizar. También se emplea por el citado bloque de control como lugar de almacenamiento temporal de datos.

c. Unos elementos de entrada/salida que permiten la intercomunicación entre el microcomputador y el usuario.

Estos tres bloques están intercomunicados entre sí, de tal manera que se produce una transferencia de información entre ellos controlada por la CPU según las instrucciones del programa. Así, en cada transferencia hace falta indicar, de forma directa o indirecta:

- El dato en sí.

- Dónde va a parar o de dónde se toma el dato.

- Para qué es el dato.

Para ello la CPU emplea una serie de líneas internas al equipo, a través de las cuales, y de forma codificada, suministra esta información. En la mayoría de los casos, esta transferencia de información se efectúa en paralelo, es decir, transfiriendo simultáneamente información por varias líneas que deben interpretarse como un conjunto y en instantes determinados. Se definen, generalmente, tres conjuntos de líneas o señales que cumplen, cada uno de ellos, una función específica. A cada uno de estos grupos se les denomina **Buses.**

- Bus de datos: transfiere físicamente los datos entre todos los elementos del microcomputador. El número de líneas que lo componen da una idea de la potencia o capacidad de cálculo de un determinado microprocesador y, de hecho se suele utilizar este número para clasificar a las distintas CPU's entre 4, 8, 16 ó 32 bits, para los microprocesadores actuales.

- Bus de direcciones: el conjunto de líneas que lo forman indican la dirección física de donde o a donde se

transfiere un dato. En función del número de lineas que lo formen, podrá accederse a más o menos posiciones de memoria en cada sistema.

- Bus de control: agrupa a todas las señales que sincronizan la transferencia de datos, dan información sobre el estado del sistema o permiten situar al microcomputador en un estado conocido.

El diagrama de bloques de un microcomputador como el Microinstructor MI-650C es el de la figura 1.1.

Los diferentes datos que constituyen la secuencia de operaciones a realizar, se encuentran en unidades de almacenamiento (memorias) a las cuales puede acceder la CPU. Cada una de estas unidades de almacenamiento puede ser identificada por un código único, denominado **Dirección de memoria.**

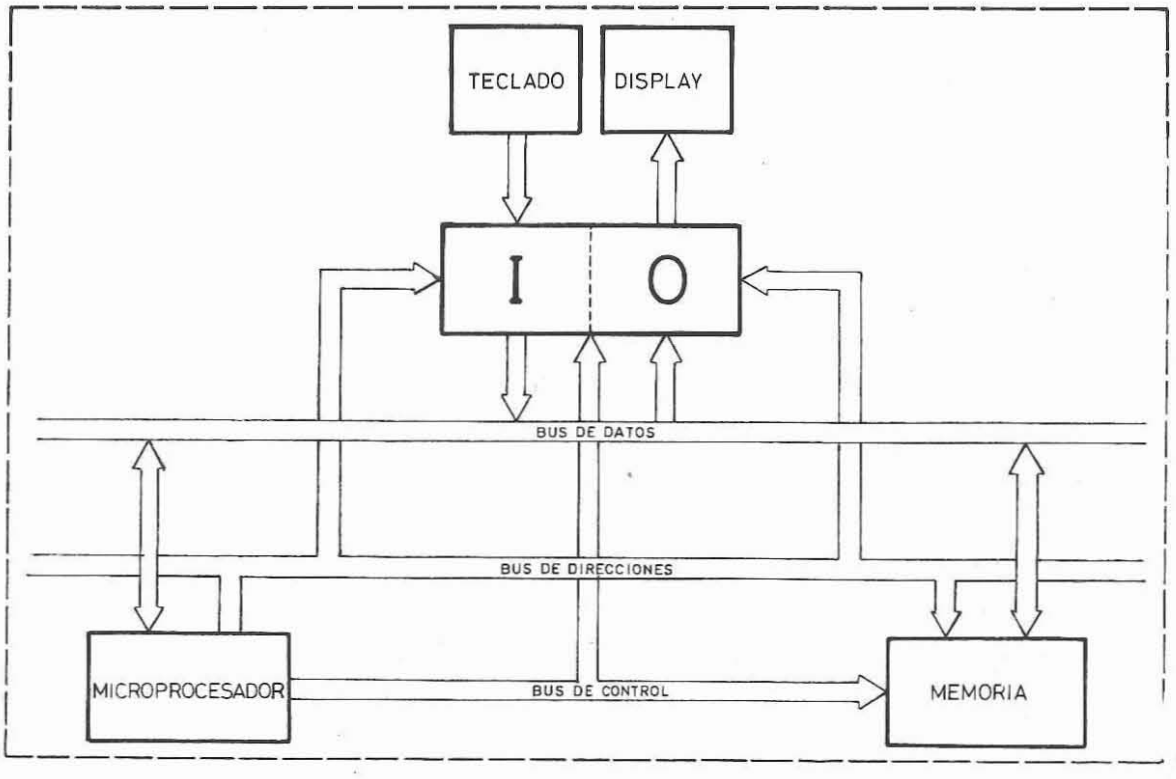


Figura 1.1. Diagrama de bloques de un microcomputador.

1.2 SISTEMAS DE NUMERACION

Tanto si se utilizan para el proceso de datos como para el control industrial, los sistemas digitales y entre ellos los microprocesadores, han de realizar operaciones con números discretos. Los números pueden representarse en diversos sistemas de numeración, que se diferencian por su base. La base de un sistema de numeración es el número de símbolos distintos utilizados para la representación de las cantidades en el mismo. El sistema de numeración normalmente utilizado es el de base diez, en el cual existen diez símbolos distintos, del 0 al 9.

Como ya es sabido, los sistemas digitales se realizan con componentes que sólo distinguen entre dos estados físicos.

Por esta razón el sistema de numeración más util para trabajar con elementos digitales es el de base dos, o **Sistema Binario,** en el cual existen solamente dos símbolos, que pueden ser el 0 y el 1, para representar a los dos estados posibles de cada elemento.

En el campo de la electrónica digital, cada cifra de un número binario recibe el nombre de **bit.** A un grupo de 8 bits se le denomina **byte.** Según esta terminología, el tamaño del Bus de Datos de un microprocesador de 8 bits es de 1 byte y el tamaño del Bus de Direcciones es de 2 bytes ó 1 **word** (16 bits).

El elevado número de bits que tienen los buses de un microprocesador hace que el manejo de la información expresada en sistema binario sea engorroso y sujeto a posibles errores. Por esta razón la notación normalmente empleada en microprocesadores es el **Sistema Hexadecimal,** o sea, en base 16. Este sistema está compuesto por los dígitos 0 a 9 y por las letras A,B,C,D,E y F.

HEXADECIMAL	BINARIO	DECIMAL
0	0 0 0 0	0
1	0 0 0 1	1
2	0 0 1 0	2
3	0 0 1 1	3
4	0 1 0 0	4
5	0 1 0 1	5
6	0 1 1 0	6
7	0 1 1 1	7
8	1 0 0 0	8
9	1 0 0 1	9
A	1 0 1 0	10
B	1 0 1 1	11
C	1 1 0 0	12
D	1 1 0 1	13
E	1 1 1 0	14
F	1 1 1 1	15

Figura 1.2. Conversión hexadecimal-binario-decimal

La razón de que normalmente se emplee este sistema es la facilidad de conversión que existe entre esta base y la binaria. En la figura 1.2 puede verse la tabla de correspondencia entre los sistemas Hexadecimal-Binario-Decimal.

Para convertir un número binario a decimal se multiplica el primer dígito (empezando por la derecha) por 2^0 , el segundo por 2^1 , el tercero por 2^2 y así sucesivamente; la suma de los resultados obtenidos dará el número decimal correspondiente.

Ejemplo:

1 1 1 1 1 1 0 1 = 253

1 1 1 1 1 1 0 1

$1x2^7 + 1x2^6 + 1x2^5 + 1x2^4 + 1x2^3 + 1x2^2 + 0x2^1 + 1x2^0 = 253$

Para convertir un número hexadecimal a decimal el proceso es similar.

Ejemplo:

FD = 253

F = 15

D = 13

F D

$15x16^1 + 13x16^0 = 253$

Como puede verse, la única diferencia consiste en que, en este caso, en lugar de multiplicar por potencias de 2 lo hacemos por potencias de 16.

De estos dos ejemplos anteriores puede deducirse que la conversión de binario a hexadecimal es inmediata, sin más que separar el número binario en grupos de 4 bits y asignar a cada uno de ellos el dígito hexadecimal correspondiente extraído de la tabla de la figura 1.2. La conversión inversa (hexadecimal-binario) se efectúa de manera análoga, empleando por cada dígito hexadecimal un grupo de 4 bits.

Este método permite representar la información en el Bus de Datos de un microprocesador de 8 bits mediante sólo dos digitos hexadecimales y la información en el Bus de Direcciones mediante cuatro dígitos.

Otro sistema a veces empleado es el **Octal** (base 8). Aunque este sistema fue muy utilizado en los primeros equipos, está actualmente en desuso, dado que un dígito octal representa sólo 3 bits. De este modo, para representar un número de 8 bits son necesarios 3 dígitos en el sistema octal, mientras que en hexadecimal sólo son necesarios 2.

La base de numeración empleada en cada caso se suele indicar mediante una letra al final de la cifra. En este manual se emplean los siguientes símbolos:

Número Decimal	(ninguno)
Número Binario	B
Número Hexadecimal	H
Número Octal	O

Por tanto, cuando a lo largo del manual aparezca por ejemplo el número 16H, debemos reconocerlo como un número hexadecimal y diferenciarlo del 16 decimal.

1.3 REPRESENTACIONES NUMERICAS

Con frecuencia, al desarrollar un programa que implemente una aplicación basada en un microprocesador, será necesario realizar operaciones matemáticas. En estas operaciones se tratarán variables cuyo tipo dependerá de la aplicación y de la precisión necesaria en cada caso. La precisión básica de un determinado microprocesador queda fijada por el número de bits del dato que pueda tratar directamente la CPU. Asi, para el 6502, la unidad de cálculo es el byte (dato de 8 bits) que permite distinguir entre 256 valores numéricos ditintos. Cuando por el tipo de aplicación sea necesario distinguir entre valores positivos y negativos, se suele utilizar el bit de más peso para indicar el signo.

En los cálculos que requieran mayor precisión, se suelen encadenar las operaciones más simples en pequeños programas que efectuen las operaciones con datos de 2 bytes (doble precisión) o n-bytes, hasta conseguir la precisión deseada. En estos casos, el significado de un determinado grado de precisión depende de la CPU utilizada, dado que operaciones realizadas con datos de 16 bits son de simple precisión para una CPU de 16 bits pero de doble precisión para una de 8 bits. En este apartado vamos a hacer un repaso a las formas principales de representar valores numéricos y a como éstos son tratados por el microprocesador.

1.3.1 Números sin signo

Internamente todos los datos deben ser almacenados como números binarios. Mediante una palabra de 8 bits se puede representar cualquier número entero positivo comprendido entre 0 y 255 (2^8 = 256 posibles combinaciones). En caso de que el número a representar sea mayor, es preciso recurrir a una técnica de múltiple precisión. Esta técnica consiste en agrupar dos o más bytes. Así, por ejemplo, empleando 4 bytes (32 bits) se pueden representar números entre 0 y 4.294.967.295.

Generalmente suele ser suficiente operar en doble precisión, que permite la representación de números en un margen de 0 a 65.535.

1.3.2 Números con signo

Si es necesaria la representación de números tanto positivos como negativos, el margen del valor absoluto del número se reduce, ya que es necesario guardar también la información sobre el signo del número.

Exiten tres modalidades de representación de números con signo:

- Magnitud más signo: El bit de mayor peso indica el signo del número, de forma que si es 0 es un número positivo y si es 1 es un número negativo. De este modo en una palabra de 8 bits el margen de representación queda limitado entre -127 (FFH) y +127 (7FH), quedando el valor 0 represetado dos veces (80H y 00H).

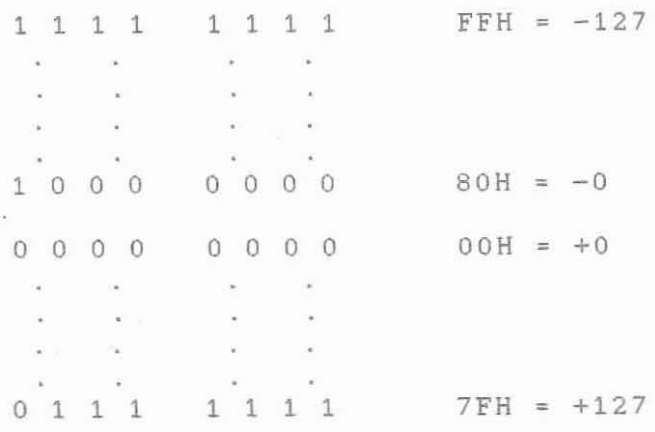


Figura 1.3. Magnitud más signo

- Complemento a dos: En este modo, una palabra de 8 bits podrá representar números comprendidos entre -128 y +127. Las primeras 128 combinaciones de bits, del 0 al 127 (0-7FH), se definen como números positivos. Las restantes 128 combinaciones (80H-FFH) representan los primeros 128 números negativos en complemento a 2. De manera que FFH es el -1, FEH es el -2, y así sucesivamente hasta 80H que es -128. Nótese que el bit más significativo indica el signo, de forma que si es 0 el número es positivo y si es 1 es negativo.

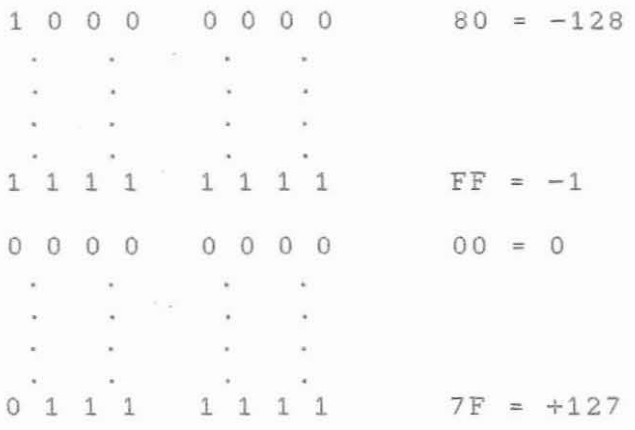


Figura 1.4. Complemento a dos.

El procedimiento para calcular la representación en complemento a dos de un número es simple. Para los números positivos el complemento a dos y el binario son iguales. Para los números negativos se siguen los siguientes pasos:

a. Escribir la representación binaria del valor absoluto del número (p.e. para -5 será 0000 0101).

b. Pasar los unos a ceros y los ceros a unos, es decir, efectuar el complemento a 1 del número (en el ejemplo anterior 0000 0101 pasa a 1111 1010).

c. Sumar uno a esta cantidad para obtener el complemento a dos (1111 1010 + 1 = 1111 1011 = FBH).

El procedimiento para obtener el valor absoluto de un número negativo representado en complemento a dos es el mismo: complementar el número y sumar uno.

En el ejemplo anterior: 11111011 pasa a 0000 0100 al ser complementado, y al sumarle uno queda 0000 0101. Este resultado indica que el número representado es el -5.

Obsérvese que el número 1111 1011 también puede interpretarse como 251 en decimal si lo hubieramos considerado como binario y no como complemento a dos. Por tanto, al tratar un número, es imprescindible conocer el sistema de representación empleado.

La representación de números en complemento a dos es ampliamente utilizada ya que, al efectuar operaciones

aritméticas con este tipo de números dan siempre resultados en complemento a dos.

1.3.3 Números Reales

Para representar números reales existen dos técnicas.

- Coma fija: En esta técnica la coma tiene asignada una posición fija. De este modo los algoritmos de realización de las operaciones son sencillos, pero presenta el inconveniente de que la magnitud de los números a representar queda muy reducida.

- Coma flotante: Esta forma de representación no presenta la limitación del sistema anterior pero tiene el inconveniente de que los algoritmos de realización de operaciones aritméticas son más complejos. El formato estandar en coma flotante es:

Signo Exponente Mantisa

Que corresponde a la representación normalizada de un valor numérico cualquiera R, como:

$$R = (-1)^{S} \times 2^{E} \times 0.nnnn...$$

La indicación 'ññ' debe interpretarse como "elevado a". S es el bit de signo, E el valor decimal equivalente del código indicado en el campo del exponente, y nnnn... es el código binario de la mantisa normalizada. El número de bits del campo destinado al exponente y a la mantisa puede ser variable para cada tipo de ordenador, pero los valores más frecuentes son según la siguiente distribución:

- Signo: 1 bit. Si es 0 el número es positivo. Si es 1 el número es negativo.

- Exponente: 7 bits. Representan el exponente de 2. Requiere una representación especial con signo, dado que la mayoría de operaciones en coma flotante se basan en una primera comparación del exponente. Para que en esta comparación pueda deducirse directamente si uno de los exponentes es mayor, menor o igual que el otro, se

utiliza el bit de más peso del exponente como bit de signo, y se le hace corresponder un '1' si es positivo. Los 6 bits restantes indican el valor del exponente. Cuando éste es negativo, se toma el complemento a 2 del exponente positivo, codificado según lo anterior.

Por ejemplo, si el exponente es +6 (0000110), su representación en el campo del exponente, como valor positivo sería (1000110). El mismo exponente con signo negativo (-6) se obtendría como el complemento a 2 del código anterior (0111010). Como se observa, el bit de mayor peso del exponente indica directamente el signo del mismo.

Esta codificación permite realizar cálculos entre +/-2**(+127) y +/-2**(-128) ó, de forma equivalente, entre +/-10**(+38) y +/-10**(-39).

⊕ Mantisa: 24, 40 ó 56 bits.

1.3.4 Números decimales

Ciertos equipos basados en microprocesadores requieren que la entrada y salida de información se efectúe en decimal, como por ejemplo una calculadora, ya que ésta es la base de numeración que utiliza el usuario.

Un método posible de tratamiento de estos números consiste en transformar el número decimal de entrada a binario en complemento a dos e inversamente para la salida, efectuando todas las operaciones intermedias en complemento a dos. Este método, aunque resulte de una gran facilidad y rapidez para la ejecución de operaciones, presenta el inconveniente de que los programas de conversión de decimal a binario y de binario a decimal son relativamente complejos y requiere un cierto tiempo para su ejecución.

Otro método empleado para el tratamiento de números decimales es tomar cada uno de los digitos decimales del número y transformarlos independientemente en números binarios de 4 bits y luego agruparlos de dos en dos en palabras de 8 bits. El número 26, por ejemplo, se representa como 0010 0110. Este sistema de representación recibe el nombre de **Decimal Codificado en Binario (BCD)** y presenta la ventaja de que la conversión decimal a BCD y BCD a decimal es inmediata. No obstante, presenta el inconveniente de que el margen queda limitado entre 0 y 99 para una palabra, lo cual implica que para representar un mismo número son necesarios un mayor número de bits que en complemento a dos. Otro inconveniente que presentan estos números es que los programas aritméticos son más complejos y lentos que en binario o

complemento a dos. Por otro lado, en este sistema de representación no existe un método análogo al complemento a dos para representar números negativos, por lo que el signo debe guardarse en un bit adicional en caso de trabajar tanto con números positivos como con números negativos.

1.3.5 **Representaciones alfanuméricas**

Muchos sistemas de microprocesadores deben operar no solamente con números, sino también con letras. Por ejemplo, un terminal de ordenador debe traducir carácteres alfanuméricos del teclado y guardarlos en forma de ceros y unos, por lo que se requiere algún tipo de codificación.

El código más empleado es el **ASCII** ("American Standard Code for Information Interchange"), indicado en la tabla de la figura 1.5.

Según esta tabla a cada carácter se le asigna un código binario de 7 bits.

Existen ciertas combinaciones (de 00H a 1FH) que no corresponden a ningún carácter alfanumérico y que sirven para transmitir órdenes de control. Algunos de estos cógigos tienen una función de control estandarizada, como por ejemplo:

CR	0DH	Retorno al inicio de la línea.
LF	0AH	Salta a la línea siguiente.
BS	08H	Retrocede un carácter.

				b7	0	0	0	0	1	1	1	1
				b6	0	0	1	1	0	0	1	1
				b5	0	1	0	1	0	1	0	1
b4	b3	b2	b1		0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	FS	,	<	L	\	l	|
1	1	0	1	13	CR	GS	-	=	M	]	m	}
1	1	1	0	14	SO	RS	.	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	_	o	DEL

NUL	Null
SOH	Start of Heading
STX	Start of Text
ETX	End of Text
EOT	End of Transmission
ENQ	Enquiry
ACK	Acknowledge
BEL	Bell
BS	Backspace
HT	Horizontal Tabulation
LF	Line Feed
VT	Vertical Tabulation
FF	Form Feed
CR	Carriage Return
SO	Shift Out
SI	Shift In
DLE	Data Link Espace
DC	Device Control
NAK	Negative Acknowledge
SYN	Synchronous Idle
ETB	End Transmission Block
CAN	Cancel
EM	End of Medium
SUB	Substitute
ESC	Escape
FS	File Separator
GS	Group Separator
RS	Record Separator
US	Unit Separator
SP	Space (Blank)
DEL	Delete

Figura 1.5. Tabla de conversión ASCII

PRACTICA 1.1 PUESTA EN MARCHA Y UTILIZACION

Para efectuar la puesta en marcha del equipo es necesario seguir los siguientes pasos:

a. Asegúrese de que el fusible situado en la parte posterior de la caja sea el correcto para la alimentación a emplear (250 mA para 125 y 220 vóltios), como se indica en la figura 1.6.

b. Conectar el cable de alimentación a la red seleccionada y colocar el interruptor a ON, indicado por el LED de puesta en marcha. El Microinstructor en estos momentos efectuará un test de los LEDs y los microinterruptores y aparecerá por unos instantes el mensaje "650 C" en el display. Despues de este test el display permanecerá con los segmentos centrales de todos los dígitos encendidos y quedará a la espera de una orden para ejecutar una función.

> **NOTA:** En caso de no obtener el saludo indicado, pasar igualmente al siguiente apartado.

c. Pulsar la tecla **[RESET]**

Despues de hacer el test quedan encendidos los segmentos centrales del display.

d. Pulsar la tecla **[STP/BRK]**

El display permanece en el mismo estado, indicando que el Microinstructor está en situación de reposo y, como en el apartado b, esperando una orden para ejecutar una función.

e. Pulsar **[GO] [F] [3] [C] [A]**

El display presentará F3CA GO

f. Pulsar **[FIN]**

En este momento se inicia el programa "secuenciador luminoso" grabado en la memoria ROM, encendiéndose secuencialmente los LEDs. Una vez finalizada la secuencia, los LEDs están apagados unos segundos, volviéndo a repetir el proceso.

g. Pulsar **[STOP/BRK]**

para detener el programa y volver al monitor.

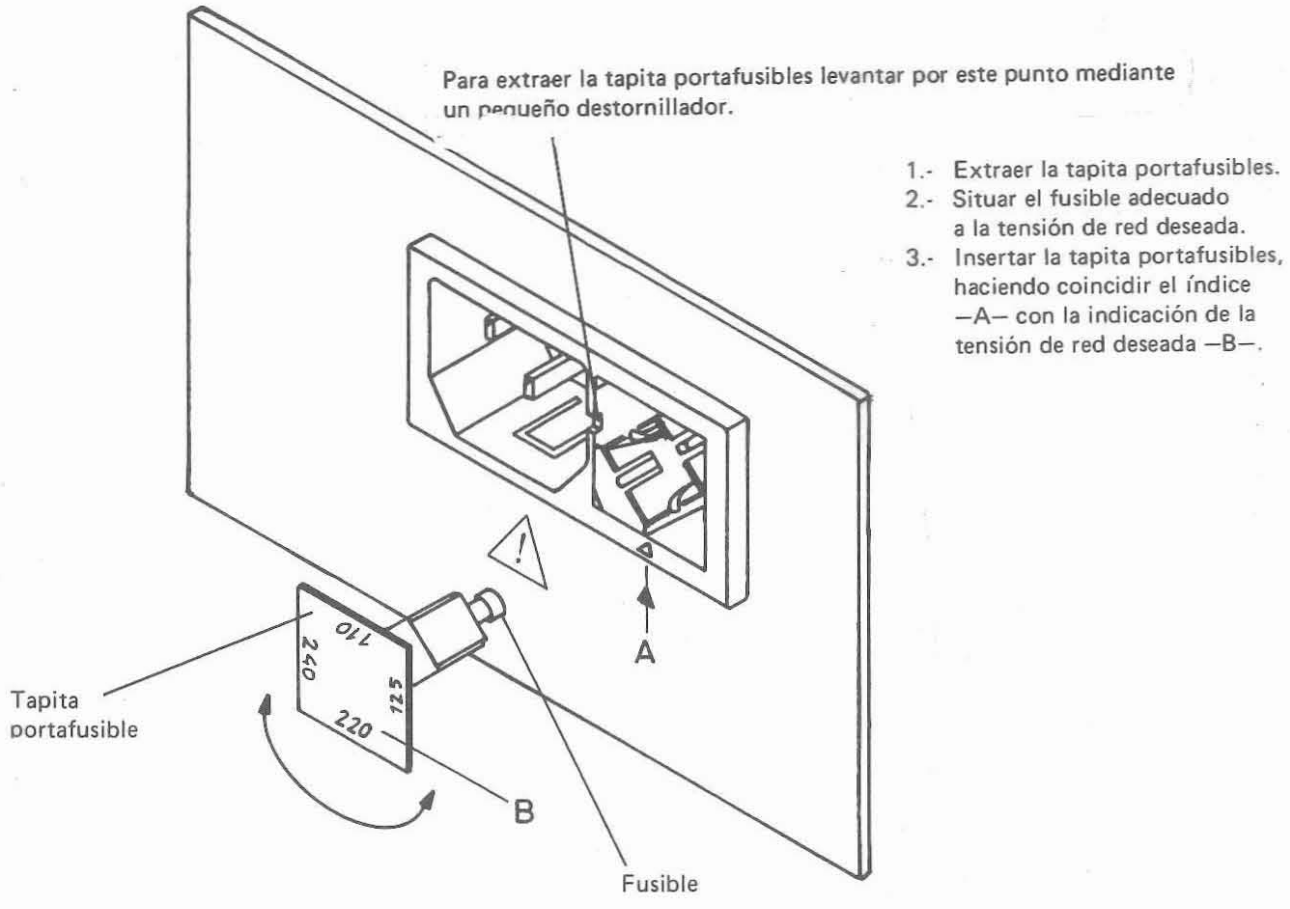


Figura 1.6. Cambio de tensión de red

h. Pulsar **[GO] [F] [F] [7] [1]**

El display indicará FF71 GO.

i. Pulsar **[FIN]**

En este momento se ejecutará el programa en ROM que inicializa el sistema.

Con esta práctica hemos ordenado al Microinstructor que ejecute dos programas que están contenidos en ROM, pulsando la secuencia: **[GO] dirección [FIN].** Para salir de ellos debe pulsarse la tecla **[STOP/BRK].**

Estos dos programas son del tipo de bucle, es decir, que siempre están repitiendo la misma secuencia de instrucciones.

Comprobamos que sin variar el hardware podemos realizar múltiples funciones, sólamente cambiando el software.

CAPITULO 2

DESCRIPCION DE LOS ELEMENTOS DE UN MICROCOMPUTADOR

Tal como se describe en el capítulo anterior, las funciones básicas que pueden asignarse a un microcomputador son la **memorización** de datos y programas, el **procesado** de estos datos de acuerdo a una descripción ordenada de códigos (programa) y la **transferencia** de información con sistemas externos al microcomputador. Cada una de estas funciones básicas puede realizarse físicamente de formas diversas según la filosofía de diseño, los requisitos de la aplicación que se pretenda desarrollar y la tecnología disponible en el momento. Estas funciones básicas se corresponden aproximadamente con los elementos que se describen en este capítulo:

- memoria
- microprocesador
- dispositivos de entrada/salida

2.1 MEMORIA DEL MICROCOMPUTADOR

Se denomina genéricamente como "memoria" a todos aquellos dispositivos que puedan mantener, de forma temporal o indefinida, información útil para la ejecución de un determinado proceso. Para que un elemento de memoria sea eficaz, ha de mantener integra la información en él almacenada y ha de permitir su recuperación (lectura).

2.1.1 Tipos de memoria

La clasificación de los tipos de memoria puede realizarse desde distintos puntos de vista. En función de si el microprocesador puede o no acceder directamente a la información almacenada:

- Memoria de masa: está constituida por sistemas de almacenamiento de gran capacidad a los que no se puede acceder directamente. Es necesario realizar una operación

previa de transferencia a una memoria de acceso directo para poder utilizar los datos en ella contenidos. Un ejemplo típico de este tipo de memoria son los sistemas de almacenamiento en disco o en cassette.

- Memoria de acceso directo: formada por todos aquellos elementos de memoria accesibles por el microprocesador durante la ejecución de un programa, con instrucciones simples. Algunos de estos elementos pueden estar incluidos dentro del propio microprocesador, como los **registros** y los **indicadores (flags),** otros suelen ser externos, como los dispositivos de memoria que contienen el programa y los datos de la aplicación.

Los registros tienen una capacidad de almacenamiento que suele ser igual a la longitud de los datos que puede tratar directamente la CPU (1 byte para los microprocesadores de 8 bits). Deben considerarse como unidades de memoria especiales, dado que al estar situados dentro del microprocesador, las transferencias de datos con ellos son mucho más rápidas que si se utilizan dispositivos exteriores. Los flags pueden considerarse como parte de un registro especial en el que se memoriza el estado del microprocesador. En el apartado correspondiente a los microprocesadores se expondrá de forma detallada las características de estos elementos.

Otra clasificación frecuente de las memorias es por la forma como se accede a los datos en ellas almacenados. Así podemos distinguir entre:

- Memorias de acceso secuencial: en ellas se accede a una determinada información siguiendo una secuencia preestablecida. Es decir, el orden en que han sido introducidos los datos determina también el orden en que pueden extraerse del dispositivo de memoria. Un ejemplo claro de acceso secuencial puede ser la lectura de datos grabados en cassette; el dato n no es accesible hasta que se hayan leido los n-1 datos anteriores.

- Memorias de acceso aleatorio: son las que tienen un tiempo de acceso igual para cualquiera de los datos en ellas contenidos, con total independencia de su posición física y del orden en que hayan sido introducidos. El caso más claro está en la memoria que contiene el programa que está ejecutando el microprocesador. En ella se accede al código de cada instrucción en función de la combinación binaria presente en el bus de direcciones. Cada dirección determina una única posición de memoria a la que se accede independientemente de las posiciones de memoria contiguas u otros factores de orden.

En algunos casos, un dispositivo de memoria puede ser utilizado con acceso aleatorio y secuencial (lectores de disco magnético). También el microprocesador puede utilizar parte de la memoria de acceso aleatorio como memoria de acceso secuencial, como veremos más adelante al analizar el funcionamiento de la **pila.** Es frecuente utilizar, bajo el control de programas de aplicación, parte de la memoria de acceso aleatorio organizada como memoria de acceso secuencial, por necesidades de la aplicación.

Los dispositivos de memoria de acceso directo externos al microprocesador son normalmente de acceso aleatorio y pueden ser, básicamente, de dos tipos: memorias RAM ("Random Access Memory") y memorias ROM ("Read Only Memory").

a. **Memorias alterables (RAM):**

Son memorias de acceso aleatorio que se caracterizan por la posibilidad de ser escritas y leídas. En la mayor parte de memorias de este tipo al desaparecer la alimentación se pierde la información en ella almacenada. Las memorias RAM están divididas en dos grandes familias, según la técnica empleada para memorizar la información:

RAM dinámica: Se mantiene la información de cada célula de memoria cargando o descargando un pequeño condensador. Debido a la natural pérdida de carga, el estado de cada una de las células debe regenerarse periódicamente (refresco de memoria) por lo que debe utilizarse una circuitería especial para realizar esta función. Su principal ventaja reside en la gran simplicidad de la célula básica de memoria, lo que permite fabricar memorias dinámicas de grán capacidad, en comparación con otros tipos de memorias.

RAM estática: Su célula básica de memorización es un circuito biestable que hace bascular su salida a "0" ó "1" en función del dato de entrada. En este caso la información no se altera con el tiempo y no se precisa de ningún circuito exterior para mantener su contenido. Al ser el circuito básico más complejo, las memórias que se pueden fabricar actualmente en esta tecnología son de menor capacidad que las RAM dinámicas.

La memoria RAM se utiliza para guardar las variables necesarias para la ejecución de un programa o para cargar programas desde un medio externo (memoria de masa: cassette, disco,...) y su posterior ejecución. También es de gran utilidad durante el periodo de desarrollo de un programa,

por la facilidad de modificación de su contenido.

En ciertas aplicaciones se utilizan memorias RAM con la circuitería adecuada para cumplir funciones específicas. Dos ejemplos típicos son:

RAM no volátil: Se basan en el empleo de RAMs construidas con tecnología CMOS de bajo consumo, diseñadas para mantener la información aun con tensiones de alimentación muy bajas (2-3 voltios), y provistas de un circuito de alimentación auxiliar con baterías de NiCd. Un circuito interno de control conmuta la alimentación de las células de memoria a la bateria, cuando desaparece la tensión exterior del sistema.

Dado el bajo consumo de estas memorias, se puede conseguir la retención de datos durante grandes periodos de tiempo.

RAM con protección de escritura: Se basan también en el empleo de memoria RAM convencional, pero en este caso la circuitería se orienta a impedir que los datos almacenados en ella puedan ser alterados por la CPU, de tal manera que, una vez protegidas, ofrezcan un funcionamiento similar a una ROM. Aunque son memorias volátiles, son de gran utilidad en equipos de desarrollo de software y, de hecho, el Microinstructor incorpora una zona de memoria con estas características.

b. **Memorias no alterables (ROM):** Como su nombre indica, este tipo de memorias, al ser accedida por la CPU, sólo permite la lectura de los datos en ella contenidos. Además son memorias no volátiles, es decir, los datos permanecen memorizados en ausencia de alimentación. Según la forma en que los datos son grabados y memorizados se pueden subdividir en:

ROM ("Read Only Memory"): En este tipo de memorias los datos se graban en la memoria durante el proceso de fabricación del circuito integrado, por lo que no podrán ser nunca modificados por el usuario. Debido al elevado coste de fabricación de la máscara de estas unidades, sólo la fabricación de grandes series justifica la utilización de este tipo de memorias.

PROM ("Programmable Read Only Memory"): En estas memorias los elementos memorizadores son una serie de fusibles incluidos dentro del circuito integrado. Cuando el fabricante suministra la memoria todos los fusibles están intactos. Mediante un equipo especial, se destruyen los fusibles necesarios para que los datos asociados con cada dirección sean los adecuados, en función de la información a almacenar.

Una vez programada, sólo puede ser alterada fundiendo algún fusible hasta entonces intacto, lo que generalmente no suele ser suficiente para cambiar datos a nuevos valores, de tal manera que puede considerarse como no reprogramable.

Estas memorias se fabrican en tecnología bipolar cuyas características principales son su consumo relativamente alto y la rapidez de acceso a los datos memorizados, lo que determina su campo de aplicación.

EPROM ("Erasable-Programmable Read Only Memory"): Este tipo de memorias es el más útil durante el desarrollo de nuevos diseños o en la fabricación de equipos en series pequeñas, debido al hecho de que se pueden reprogramar para volver a ser utilizadas en otra operación. Es el tipo de memorias empleadas en la familia Microinstructor y Módulos. El almacenamiento de información se efectúa cargando una serie de condensadores incluidos en el circuito integrado y que prácticamente tienen pérdidas nulas, ya que son capaces de mantener la información durante periodos de tiempo superiores a los 10 años. Al no alterarse ningún elemento constitutivo del circuito integrado, pueden ser "borradas" descargando el condensador de almacenamiento. El borrado se efectúa mediante luz ultravioleta que incide sobre el "chip" a través de una ventana de cuarzo.

La programación de su contenido se efectúa mediante impulsos de tensión mayor a la de funcionamiento (21 ó 25 voltios, en la mayoría de modelos) que se generan con equipos especiales. I.E. Prómax,SA ha diseñado para este propósito el Programador de Memorias PR-545, con capacidad para grabar memorias EPROM de hasta 256K.

EEPROM o EAROM ("Electrically Erasable-Programable Read Only Memory" o "Electrically Alterable Read Only Memory"): Estas memorias son parecidas a las EPROMs pero se

diferencian en que el borrado se efectúa introduciendo un pulso de una tensión dada en una de sus patillas.

La programación y el borrado de estas memorias también puede efectuarse con el equipo Programador de Memorias PR-545.

2.1.2 Función de la memoria en el microcomputador

En un microcomputador las funciones de los distintos tipos de memoria son las siguientes:

a. **ROM:** En esta memoria se encuentran los programas para el control del funcionamiento del microcomputador. Aunque el programa principal esté en memoria RAM, siempre se requiere al menos un pequeño programa que tome control de la CPU tras el encendido del equipo, aunque su única función sea cargar el programa principal en RAM y transferirle a éste el control.

En el caso de equipos de entrenamiento, como el Microinstructor, este programa se llama **Monitor** y se encarga del control de las funciones básicas del equipo, como la entrada y ejecución de órdenes o la carga de un programa de aplicación desde un medio externo.

La memoria ROM también puede contener tablas de datos constantes que se requieran para el funcionamiento del sistema. Estos datos pueden ser tablas de conversión o condiciones de funcionamiento que el microcomputador deba tomar inicialmente o por defecto.

b. **RAM:** Esta memoria tiene varias funciones dentro de un microcomputador. Para cada una de ellas se establece, sobre la memoria RAM, una zona propia.

- **Pila (Stack):** Es una zona de memoria RAM empleada de forma automática por la CPU, como memoria secuencial. En el Microinstructor esta zona es un grupo de 256 bytes entre 100H y 1FFH. Más adelante se analizará con más detalle.

- **Memoria temporal (Scratch area):** Es la zona de memoria RAM que contiene las variables que emplean los programas para su ejecución. Estas variables pueden ser utilizadas como contadores, almacenamiento

temporal, etc. En esta zona, el contenido de una determinada posición de memoria puede tener significados distintos a lo largo de la ejecución de un programa.

- **Memoria de datos:** Es la zona de memoria RAM donde se encuentran aquellas variables que son manipuladas por el programa. Un tipo particular de datos que se almacena en esta zona de memoria son las tablas. Si por ejemplo pensamos en un secuenciador programable (ver capítulo 5), los datos que definen la secuencia pueden estar en RAM, permitiendo de esta manera que la citada secuencia pueda ser alterada por el usuario. El programa podrá utilizar estos datos y modificarlos, pero su significado se mantendrá durante todo el programa.

- **Memoria de programa:** En esta zona de memoria se almacenan programas que han de ser ejecutados por la CPU pero que se desea poder alterar o sustituir por otros. Además de este almacenamiento temporal de programas durante el proceso de diseño y depuración de los mismos, en esta zona pueden almacenarse también programas de control que han sido cargados desde la memoria ROM o desde otro medio de almacenamiento de información.

NOTA: La letra K es una denominación estandar empleada para indicar la cantidad de memoria. Según esta denominación, 1 K = 1024. Así, un bloque de 4 Kbytes incluye realmente 4096 (4x1024) bytes de memoria y 32.768 (4096x8) bits. 16 K son 16.384 posiciones de memoria direccionables, sin indicación del tamaño del dato que en ellas se puede almacenar.

2.1.3 Direccionamiento de las memorias

La selección de los distintos dispositivos de memoria y de entrada/salida se realiza mediante la activación de un conjunto de señales eléctricas que llamaremos líneas de selección. En función del estado lógico de este conjunto de líneas, la memoria queda conectada o aislada de la CPU. En los casos más simples se dispone de una única línea de selección que llamaremos genéricamente /CS (chip select). La barra inclinada inicial

significa que la señal es activa en su nivel bajo. Por tanto, el dispositivo quedaría seleccionado si /CS=0 y desconectado del bus de datos del sistema cuando /CS=1.

Si la memoria permite tanto el almacenaciento como la lectura de datos, deberá tener una entrada adicional para seleccionar uno de los dos estados (R/W).

CS	R/W	FUNCION
0	0	ESCRITURA
0	1	LECTURA
1	X	ALTA Z

0=L=BAJO
1=H=ALTO
X=ESTADO INDIFERENTE

Figura 2.1. Control de una RAM

Todos estos elementos de memoria y de entrada/salida se conectan al bus de datos (ver figura 2). Para evitar que haya colisión en el movimiento de información en el bus de datos, mientras un elemento no es seleccionado (/CS=1) las salidas del bus de datos permanecen en alta impedancia ("Three state") estado equivalente a la desconexión del sistema.

La CPU indica el elemento de memoria direccionado en cada caso mediante un conjunto de señales digitales (16 bits para un microprocesador típico de 8 bits), a través del bus de direcciones. Se requiere, por tanto, una serie de decodificadores que, a partir de estos 16 bits y otras líneas de control, permitan activar los circuitos de memoria correspondientes y acceder a cada elemento de memoria.

Esta decodificación se efectúa en dos fases: por una parte, dentro de cada circuito integrado de memoria ya existe un decodificador; por ejemplo, en una memoria de 8Kx8, las 8192 líneas de selección se generan internamente a partir de 13 líneas del bus que se conectan al circuito integrado de memoria.

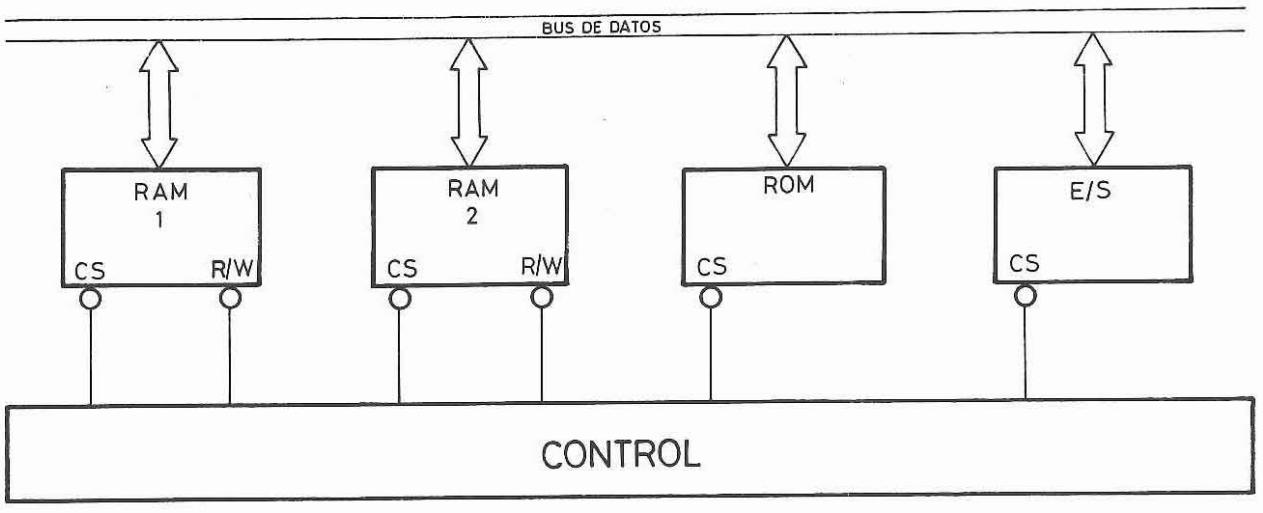


Figura 2.2. Dispositivos 'Three state'

La segunda fase de decodificación se efectúa mediante decodificadores integrados externos a los circuitos de memoria y actuando sobre las restantes líneas del bus de direcciones. En el ejemplo anterior, suponiendo un bus de direcciones de 16 bits, quedarían 3 líneas más del bus de direcciones para determinar la posición absoluta de cada elemento de la memoria, dentro de las 65.536 posiciones teóricamente disponibles.

El posicionamiento de cada dispositivo de memoria y E/S dentro de la zona de memoria direccionable por el microprocesador, es una de las partes más importantes del diseño de sistemas digitales basados en microprocesador. Básicamente, se trata de generar un **mapa de memoria** en el que se sitúe, sobre la extensión total de memoria direccionable por la CPU, la posición de cada dispositivo que se pretenda conectar a la misma. Sobre esta representación gráfica de la memoria, se indican las zonas ocupadas y las disponibles, el tipo de memoria o dispositivo de E/S y la dirección inicial y final del sector ocupado por cada dispositivo.

El circuito decodificador debe diseñarse para que cada dispositivo se active únicamente para las direcciones a él asignadas en el mapa de memoria.

En el capitulo 3 se describe el circuito de decodificación utilizado en el Microinstructor.

2.1.4 Ejemplo de Decodificación

Supongamos que se desea diseñar un pequeño equipo con 2 K de memoria RAM, 2 K de memoria ROM y 2 K para los dispositivos de entrada/salida (que trataremos como una zona de memoria igual a las anteriores), de manera que sólo utilicemos 6 K de los 64 K que puede direccionar el microprocesador. Veamos dos formas de resolver el problema, de entre las múltiples posibilidades disponibles:

a. Dividir el área de direccionamiento en 4 bloques de 16 K cada uno. Esto se logra introduciendo las 2 líneas más altas del bus de direcciones (A15 y A14) en un decodificador de 2 a 4 (por ejemplo 74LS139).

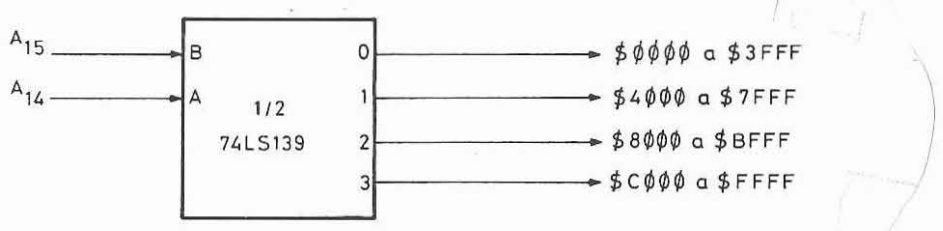


Figura 2.3. Decodificador de 2 a 4

Tres de las salidas del decodificador se conectan a las entradas de selección (/CS) de los tres integrados. De este modo desperdiciamos una gran zona del mapa de memoria puesto que sólo nos quedarán 16 K libres para posibles ampliaciones. por otro lado, cada dispositivo de 2K de memoria útil está ocupando 16K del mapa de memoria. Este tipo de decodificación se denomina "incompleta" y suele utilizarse en sistemas muy simples o con poca necesidad de memoria y periféricos.

b. Dividir estos cuatro bloques de nuevo en partes más pequeñas. Para ello se conecta la primera línea generada por el decodificador anterior a la entrada de selección del circuito integrado 74LS138 y las líneas A13, A12 y A11 a sus entradas.

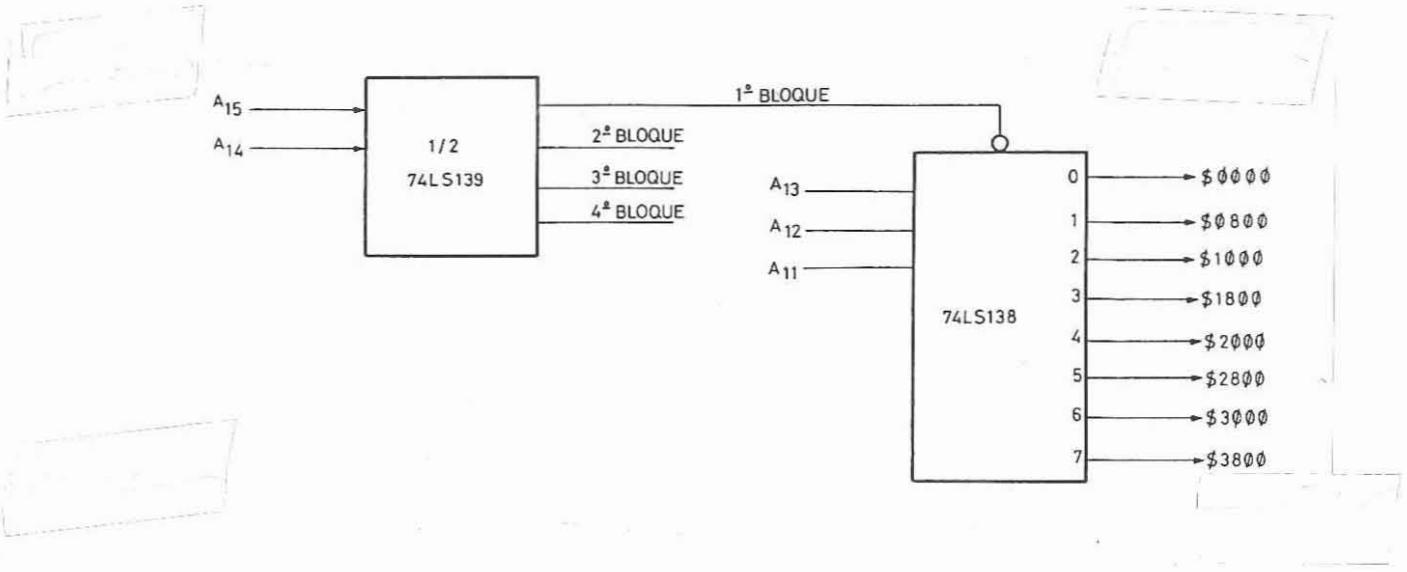


Figure 2.4. Decodificador de 8 salidas

Con este método logramos particiones de 2 K cada una y, por tanto, todavía quedan 58 K direccionables. Este caso modela la decodificación completa, en que el espacio ocupado por cada dispositivo sobre el mapa de memoria se ajusta a su capacidad real. El circuito decodificador es más complejo y su utilización se centra en aquellos sistemas que requieren un gran número de periféricos y de circuitos de memoria.

2.2 EL MICROPROCESADOR

Un microprocesador es una Unidad Central de Proceso (CPU) integrada en un solo chip. Técnicamente, es un sistema digital secuencial reprogramable, en el que los estados que se van recorriendo dependen, además de su propia estructura circuital, de un programa exterior. Este hecho le proporciona una gran flexibilidad de trabajo en aplicaciones de control y cálculo.

2.2.1 Estructura del microprocesador

El microprocesador es la parte principal del microcomputador. Su función es la de recoger datos de elementos de entrada o de memoria, procesarlos adecuadamente en función de las instrucciones del programa y remitirlos de nuevo a memoria o a unidades de salida. En la figura 2.5 se representa un diagrama de bloques que puede ser válido, en líneas generales, para cualquier CPU.

El tipo de operaciones elementales que pueden realizar estos sistemas son las siguientes:

- Cargar un registro con un dato.
- Leer una determinada posición de memoria y llevar el dato leído a algún registro de la CPU.
- Escribir en una determinada posición de memoria el dato que estaba contenido en un registro de la CPU.
- Efectuar operaciones aritméticas y lógicas elementales entre dos datos.
- Transferir el contenido de un registro a otro.
- Incrementar o decrementar el contenido de un registro.

Cualquier operación más compleja podría descomponerse en una determinada combinación de las aquí expuestas. Además pueden ejecutar instrucciones de salto y salto condicional, de forma que puedan alterar la ejecución lineal del programa, en función de que se cumplan o no determinadas condiciones.

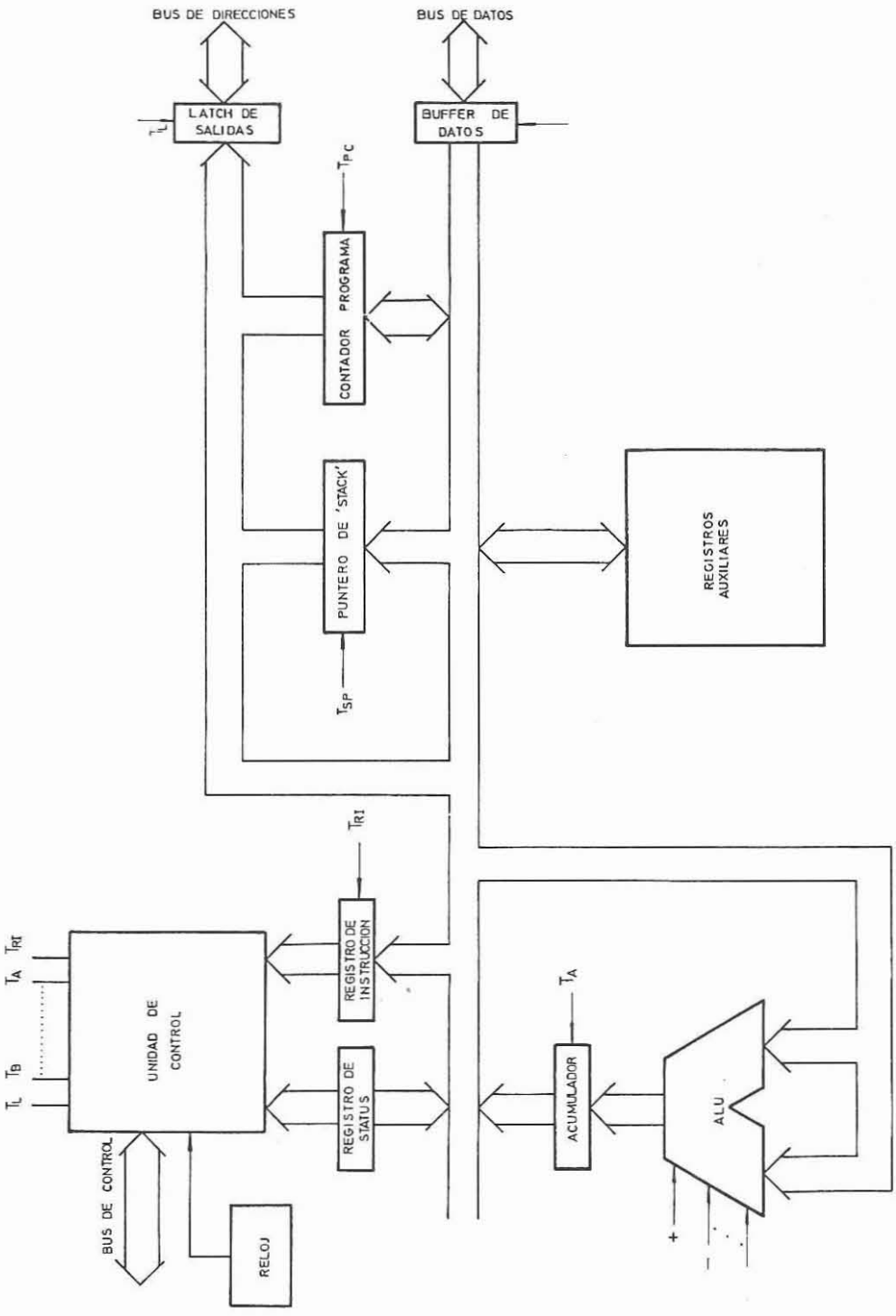


Figura 2.5. Diagrama de bloques de una CPU.

2.2.2 Proceso Básico de Ejecución de un Programa

Para ejecutar un programa hemos de partir de un conjunto de instrucciones codificadas en forma de una secuencia de valores binarios. Además es preciso que la CPU conozca la dirección del mapa de memoria donde se inicia dicho programa.

El programa monitor, debe ejecutarse de forma automática cada vez que se ponga en marcha el equipo, por tanto, debe residir en una memoria ROM u otra memoria no volátil. En este caso, el procedimiento para determinar la dirección inicial del programa, es el de forzar un **RESET** con el transitorio de conexión de la fuente de alimentación. La condición de reset (reinicio) establece una dirección de memoria como inicio de ejecución, de forma automática. En el microprocesador 6502, está dirección se toma del vector almacenado en las posiciones FFFCH (byte bajo) y FFFDH (byte alto) del mapa de memoria. El contenido de estas dos posiciones de memoria apunta al inicio del programa de arranque del sistema o, en nuestro caso, del programa monitor.

A partir del momento en que se ha determinado la dirección de inicio del programa, comienza un proceso iterativo, para la ejecución del mismo. En él intervienen los distintos elementos que configuran la CPU, y que se expondrán en los apartados siguientes. Este proceso se puede resumir, a grandes rasgos, en:

- Búsqueda de instrucción: se pasa la nueva dirección al bus de direcciones y se inicia de lectura del código de instrucción. Este código se entra por el bus de datos y se sitúa en un registro interno especial.

- Decodificación: el código de instrucción adquirido se interpreta y se generan, si son necesarios, los ciclos de lectura de memoria para completar su significado. Por ejemplo, si se trata de sumar un valor a un registro, debe leerse, previamente a su ejecución, el valor a sumar.

- Ejecución: Se generan internamente una secuencia de microinstrucciones para cada uno de los elementos que configuran la CPU que estén relacionados con la instrucción en curso (ALU, registros, indicadores, etc...)

- Cálculo de la dirección siguiente: en la mayoría de los casos se tratará únicamente de incrementar el contenido del registro interno que contenga la última dirección leida. Pero si la instrucción ejecutada implica un salto, condicional o incondicional, se deberá calcular la dirección sobre la que debe realizarse el siguiente ciclo de búsqueda de instrucción.

Algunos de estos procesos pueden realizarse simultaneamente, cuando los elementos necesarios para calcular y generar las direcciones de búsqueda de la siguiente instrucción son independientes de los elementos de ejecución de instrucciones. Este tipo de arquitecturas de CPU se denomina "pipeline" y se corresponde con la de la familia 6502. Este tipo de estructuras permiten aumentar la velocidad de ejecución de programas.

2.2.3 La Unidad de Control

La Unidad de Control es el circuito que gobierna y coordina todas las operaciones que tienen lugar dentro de la CPU. Esto lo realiza generando una secuencia de microórdenes. Cada salida de la Unidad de Control genera una microorden específica destinada a actuar sobre un elemento de la CPU.

Estas microórdenes son del tipo: un registro acepta un dato o lo entrega al bus, un registro se incrementa o decrementa, un registro se desplaza, la ALU efectúa una operación determinada, etc... Las microórdenes son generadas a partir de dos informaciones:

- La Instrucción, que procede del exterior de la CPU.

- El Status, que procede del interior de la CPU.

Una instrucción es un código binario que indica a la CPU la función que debe realizar. La unidad interna decodifica esta instrucción de forma que, para cada instrución distinta se ejecutará una secuencia de microórdenes distinta.

Por otro lado, la acción del controlador está condicionada por la historia anterior de la CPU presente en su entrada de status. Por esta entrada llegan al controlador informaciones tales como que en un registro hay valores positivos o negativos, que se ha producido desbordamiento (overflow) en una operación de la ALU, etc. En función de esta información la Unidad de Control puede generar microórdenes diferentes para una misma instrucción.

En la Unidad de Control todos los cambios de estado están sincronizados por una señal de reloj que generalmente procede del exterior de la CPU. La frecuencia de esta señal de reloj determina la velocidad a que se efectúan todas las operaciones.

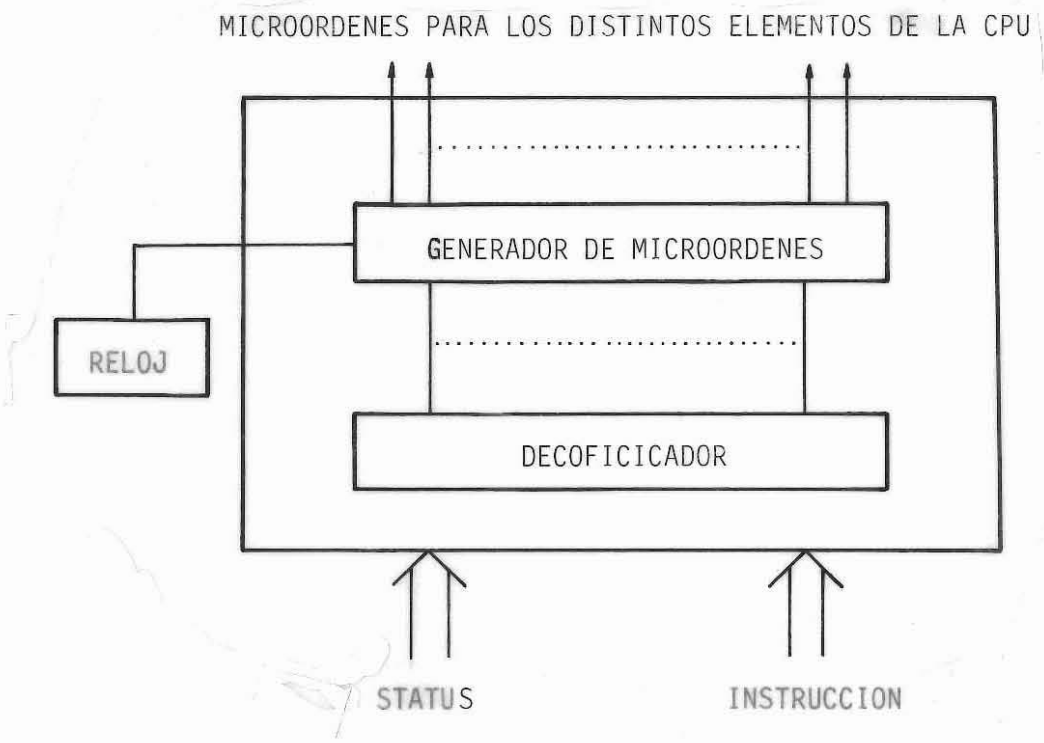


Figura 2.6. Unidad de control

2.2.4 La Unidad Aritmético-Lógica (ALU)

La ALU es un circuito combinacional responsable de las operaciones aritméticas y lógicas que es capaz de realizar la CPU. Dispone de entradas para dos operandos. El resultado se deposita en un registro llamado Acumulador. Las entradas de control permiten seleccionar el tipo de operación que se va a realizar. Además pueden existir una entrada/salida de acarreo (carry) y una salida de desbordamiento (overflow).

Las funciones lógicas que normalmente realiza la ALU son AND, OR, OR-exclusivo y NOT (inversión) o combinaciones de ellas.

Además puede realizar las funciones aritméticas suma y resta con y sin signo y con o sin acarreo. Cualquier otra operación aritmética que una CPU de estas características sea capaz de llevar a cabo está basada en una adecuada secuenciación de estas operaciones elementales.

En la figura 2.7. se muestra un ejemplo del modo de funcionamiento de la ALU para un caso determinado de operandos de entrada.

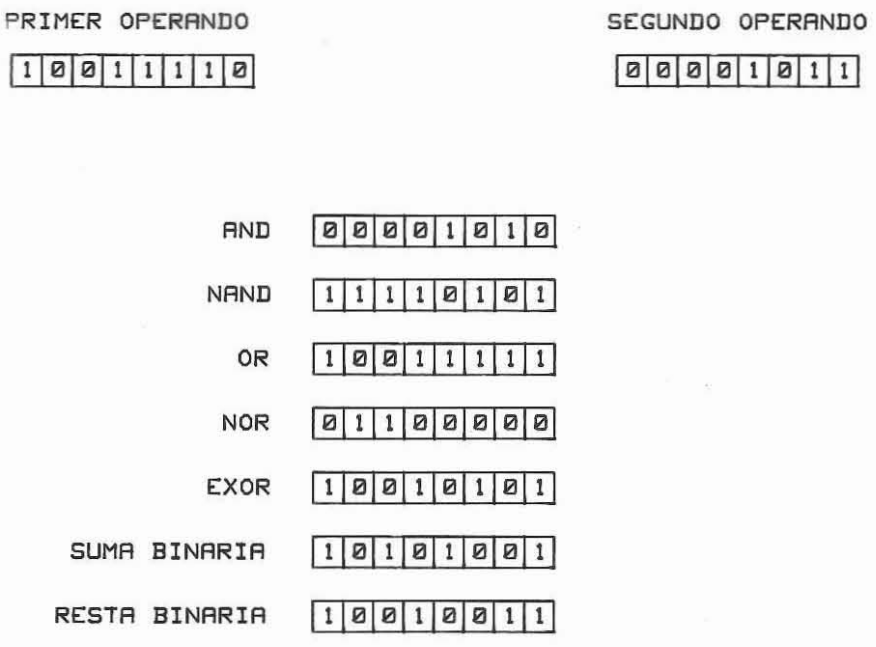


Figura 2.7. Ejemplo de operaciones de la ALU

En la figura 2.8. vemos que se utilizan tres registros en torno a la ALU, dos registros para almacenar los operandos y un Acumulador. No obstante, en muchos casos se opta por suprimir uno de los registros de operando sustituyéndolo por el propio Acumulador. De modo que antes de la operación, el Acumulador contiene un operando y después de la operación, almacena el resultado.

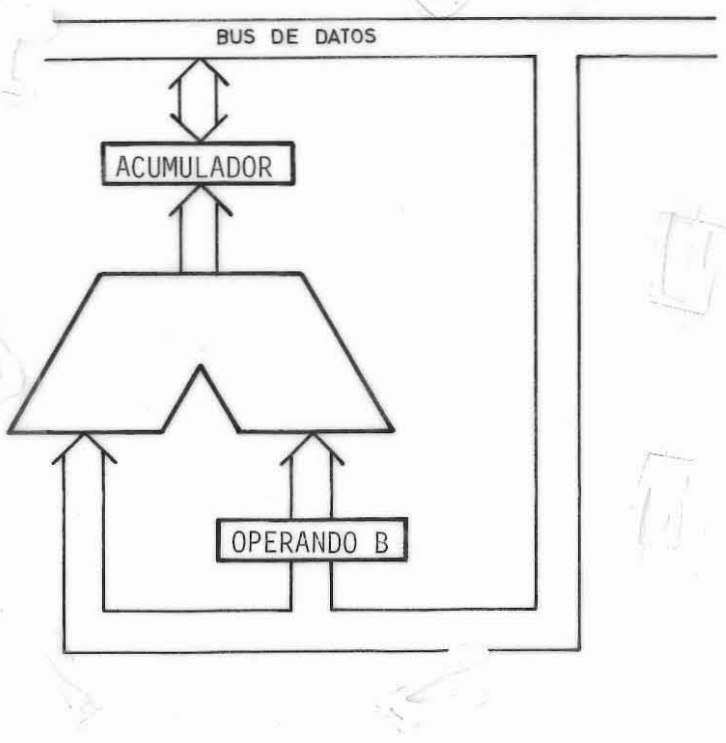


Figura 2.8. ALU

2.2.5 El Acumulador

Es un registro especial sobre el que se realizan la mayor parte de operaciones en las que debe intervenir la ALU. En este registro se suelen poder efectuar operaciones especiales, no disponibles para otros registros de proposito general. Entre otras, suelen encontrarse las operaciones de rotación y desplazamiento de bits. La tendencia actual disminuye las posibles diferencias entre el Acumulador y los otros registros, por lo que se suelen tratar como un banco de registros homogéneos, en los que cada uno puede ser dato de entrada a la ALU y destino de un posible cálculo.

2.2.6 **Registros Auxiliares**

Además del Acumulador la CPU suele contar con una serie de registros cuyo número y función varían mucho en función del microprocesador. En general se puede decir que estos registros tienen la función de almacenar datos temporales sin necesidad de acceder a la unidad de memoria y aumentando, por tanto, la velocidad del proceso. Otro propósito de estos registros es el de facilitar la generación de direcciones de memoria ampliando la capacidad de direccionamiento de la CPU. Para ello se utilizan los registros llamados de índice que permiten sumar o restar un valor, que puede ser variable, a una dirección base de memoria. Esta posibilidad aumenta la potencia de tratamiento de bloques de datos y se ampliará su estudio al analizar las distintas formas de direccionamiento para los microprocesadores de la serie 6502. Otro registro especial es el llamado **Puntero de Pila (Stack Pointer)** que se estudiará con más detalle en el apartado dedicado a las subrutinas.

2.2.7 **Registro de Estado ó Status**

Este registro guarda una síntesis de los estados anteriores por los que ha pasado la CPU. En el registro de Estado cada bit está destinado a memorizar si ha tenido lugar un determinado suceso dentro de la CPU. Cada uno de estos bits recibe el nombre de **Indicador (flag).** Los sucesos que se suelen memorizar son los siguientes:

- si el resultado de la operación ha sido cero (Flag de cero).
- si el resultado de la operación ha sido negativo (Flag de signo).
- si en la última operación se ha producido acarreo (Flag de acarreo).
- etc.

Como ya se ha explicado anteriormente, la Unidad de Control genera, para algunas instrucciones, secuencias de microcódigos distintos en función del estado de determinados bits de este registro. a su vez, la ejecución de cada instrucción puede modificar el contenido de alguno de estos bits de estado.

El código de operación es totalmente necesario en todas las instrucciones. En los microprocesadores de 8 bits más utilizados, el código de operación es el primer byte de la instrucción. Este byte es el que ocupa el Registro de Instrucción.

En general las instrucciones se realizan en secuencia lineal y están almacenadas en posiciones contiguas de memoria. Por ello se puede prescindir, en algunos casos, de la información que indica la dirección de memoria donde se encuentra la instrucción siguiente.

El **Contador de Programa (Program Counter).** es un registro especial que contiene la dirección de memoria de la instrucción en curso. En los casos de ejecución de un programa sin saltos ni bifurcaciones, es suficiente incrementar su contenido en una unidad despues de realizada cada instrucción para obtener la dirección de la siguiente instrucción a ejecutar. Pero el microcomputador ha de poder tomar decisiones, es decir, modificar la secuencia de operaciones en función del resultado obtenido al realizar una instrucción. Para ello debe existir, dentro del conjunto de instrucciones, una que indique la dirección de la siguiente instrucción a realizar. A esta instrucción se la denomina instrucción de salto o de ruptura de secuencia y puede ser condicional o incondicional. Si es condicional la instrucción realiza una operación y según el resultado obtenido rompe o no la secuencia. La ruptura de la secuencia se produce cambiando el contenido del Contador de Programa por la dirección o por el salto indicado en la instrucción.

Para reducir más el número de bits del campo de información de la instrucción el resultado de una operación se puede almacenar en la misma dirección de uno de los operandos, de manera que se haga innecesario especificar en la instrucción la dirección del resultado.

La mayoría de los microprocesadores no poseen un único registro acumulador sino que tienen varios independientes entre sí o interconectados formando una pequeña unidad de memoria. Esto permite al microprocesador almacenar resultados parciales sin necesidad de utilizar memoria externa.

De lo dicho anteriormente se deduce que la información, mínima, que debe proporcionar una instrucción se reduce a:

- un **código de operación (OPCODE),** que indique la operación a efectuar.

- un **operando** que indica sobre qué dato o datos se ha de efectuar la operación.

Durante la ejecución de un programa, la CPU está permanentemente llevando a cabo ciclos de búsqueda ("fetch") y ejecución de instrucciones, excepto si por alguna razón se ha detenido la operación mediante alguna de las líneas de control que se citan más adelante.

Este hecho es válido en todo momento, aunque no se observe ninguna actividad apreciable en los elementos de entrada/salida. Por ejemplo, en el Microinstructor MI-650C y tras la pulsación de la tecla RESET, el equipo aparentemente deja de funcionar. No obstante, internamente está constantemente analizando el teclado y presentando datos en el display.

2.2.9 **Subrutinas**

En un programa puede ocurrir que una serie de instrucciones deban ser ejecutadas varias veces a lo largo del mismo. Estos conjuntos de instrucciones pueden agruparse en bloques independientes del programa principal denominados **subrutinas.**

Por lo general, existirá una instrucción (o varias) que alteren el contenido del Contador de Programa de forma que la ejecución continúe en el inicio de una subrutina. Una vez se ha llegado al final de la secuencia de instrucciones que forman la subrutina el Contador de Programa recupera el valor que tenía y puede continuar la ejecución del programa principal a partir de la instrución siguiente a la de salto a subrutina. Este retorno al programa principal también se indica mediante una instrucción específica.

A modo de ejemplo, supongamos un programa que contiene dos veces una secuencia de instrucciones que efectúa una multiplicación. Estas instrucciones pueden agruparse como subrutina y a lo largo del programa, cuando se han de efectuar las multiplicaciones, hacer un salto a esta subrutina (ver figura 2.9).

Cuando se llega a la última posición de la subrutina el Contador de Programa se sitúa automáticamente en la instrucción del programa principal que sigue a la de salto a la subrutina MULT y sigue la ejecución del programa a partir de este punto.

Una de las ventajas que tiene la utilización de subrutinas a lo largo de un programa es que éstos, al ser más cortos, ocupan menos memoria. Del mismo modo, la tarea del programador también se reducirá al tener que escribir menos instrucciones. Otra de las ventajas de utilizar subrutinas es que su uso ayuda a la estructuración de los programas y por tanto a la mejor comprensión de los mismos. Como contrapartida, la utilización de

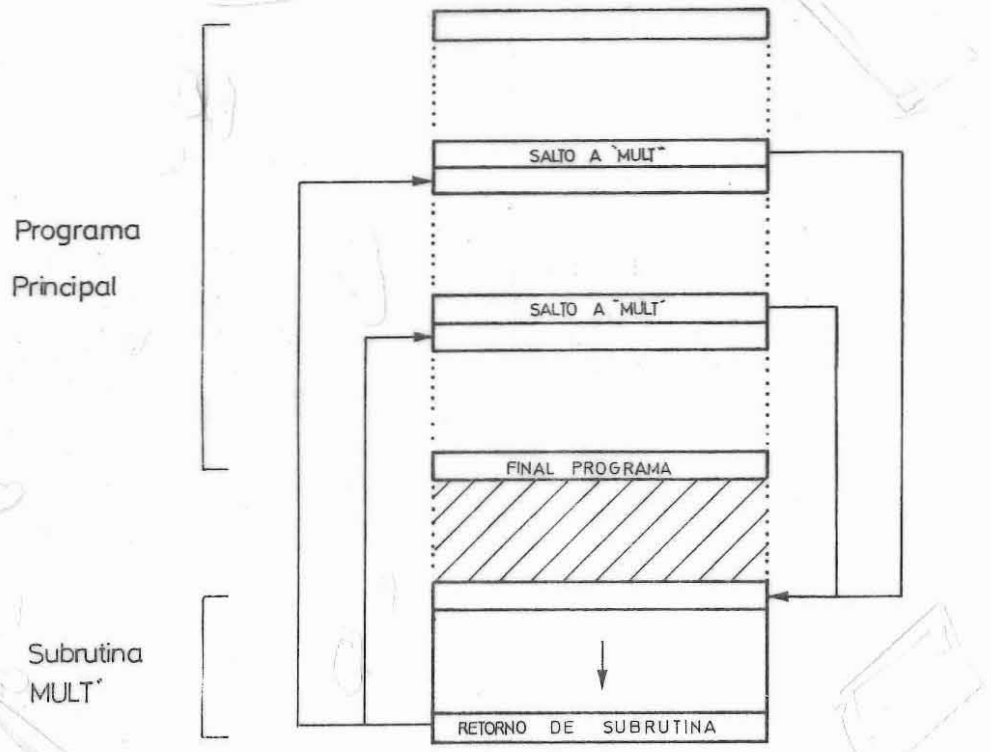


Figura 2.9. Saltos a subrutinas

subrutinas disminuye la velocidad de ejecución del programa, respecto del mismo programa con ejecución lineal (sin saltos).

Es posible que desde una subrutina se produzcan saltos a otras subrutinas y se cree por tanto un anidamiento. Esto requiere, como en el caso anterior, la memorización del valor del Contador de Programa en el momento de cada salto. La memorización de estas direcciones se efectúa normalmente en una zona de la memoria RAM llamada **Pila (Stack)**

La CPU dispone a su vez de un registro auxiliar denominado **Puntero de Pila (Stack Pointer)** que indica la dirección de la pila del último valor almacenado. La Pila esta configurada como una zona de memoria de acceso secuencial del tipo LIFO (Last In - First Out) en la que el último dato introducido es también el primero en salir al ser "leida" la pila por el programa. Se puede tener una idea bastante gráfica de su funcionamiento si imaginamos el stack como el resultado de ir apilando un conjunto de papeles sobre una mesa. El último papel apilado es el primero que tenemos disponible para su lectura. De forma equivalente, cada vez que accedemos a una subrutina, se introduce en la parte superior del stack la dirección de retorno para su recuperación al final de la misma.

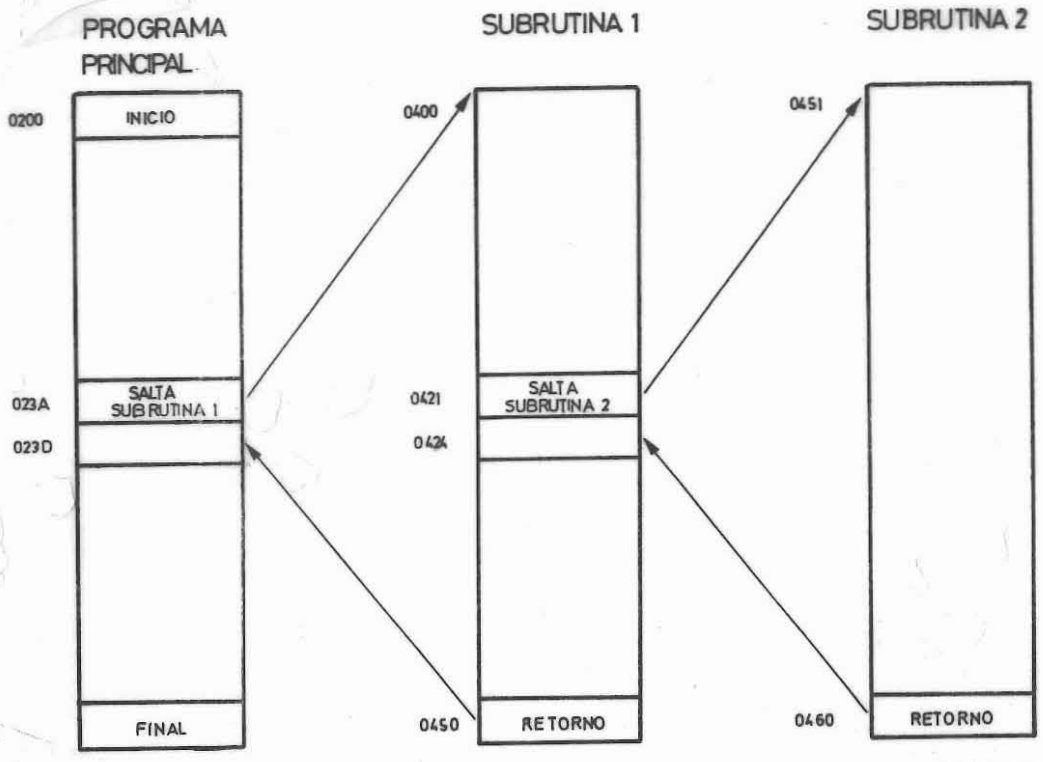


Figura 2.10. Subrutinas anidadas

Pero al ejecutarse un salto a una subrutina además del Contador de Programa, puede que sea necesario también recordar los datos contenidos en otros registros de la CPU con el fin de que a la vuelta al programa principal se reanude la ejecución del mismo en las mismas condiciones existentes en el momento de producirse el salto a la subrutina.

En la figura 2.10 puede verse una estructura de programa anidado y en la figura 2.11 el estado del "Stack" y del "Stack Pointer" en diversos puntos del programa indicados mediante el Contador de Programa.

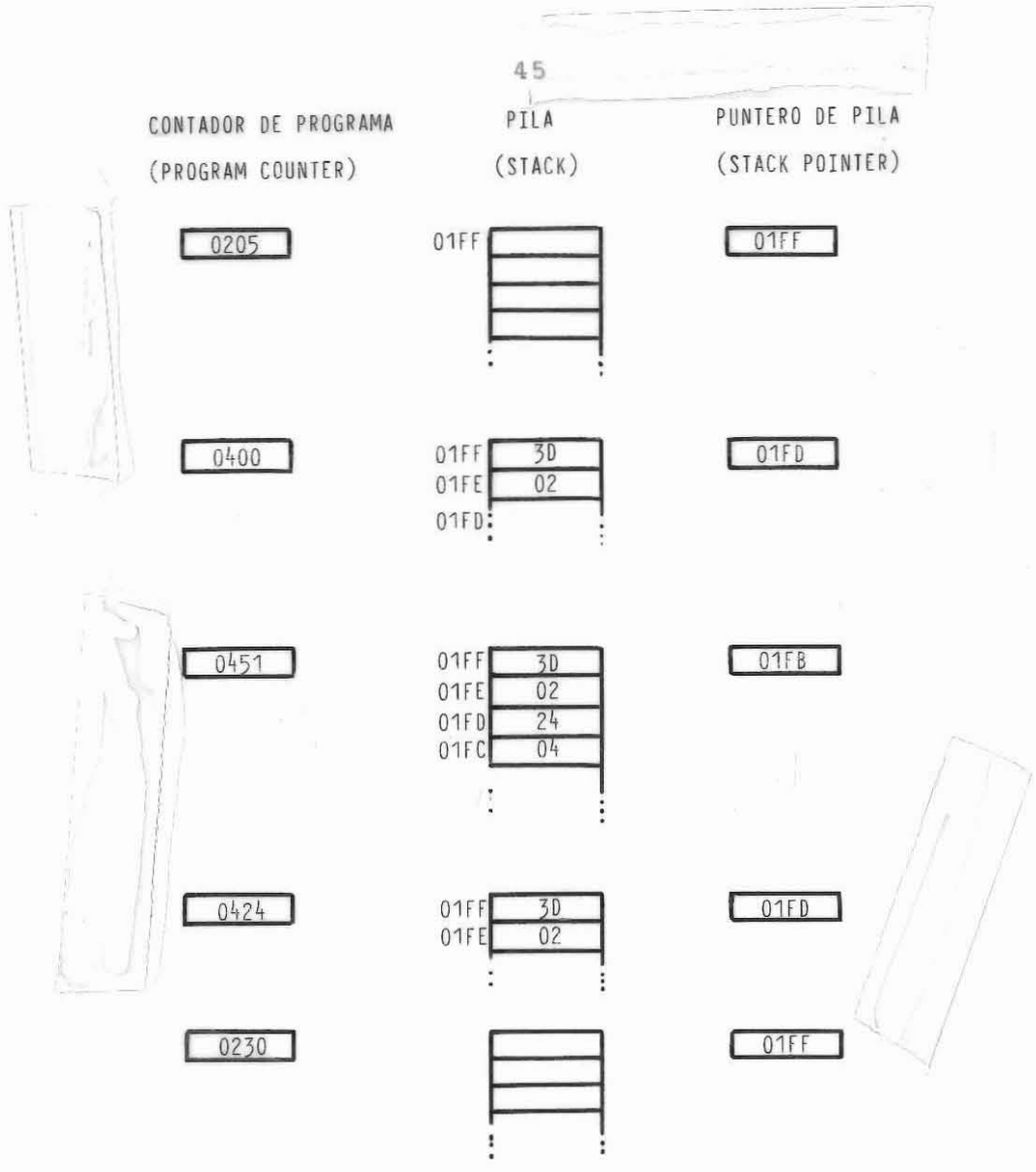


Figura 2.11. Operación del Stack

2.2.10 **Interrupciones**

Las interrupciones son un mecanismo muy importante para la sincronización de los programas con acontecimientos externos.

El funcionamiento de las interrupciones es muy parecido al de las subrutinas. La diferencia más importante estriba en la manera en que se produce la interrupción. Mientras las subrutinas se ejecutan cada vez que se encuentra una instrucción que así lo indica, las interrupciones se ponen en marcha al activarse una línea de entrada de la CPU, es decir, mediante un mecanismo "hardware", pudiendo producirse en cualquier momento de la ejecución del programa.

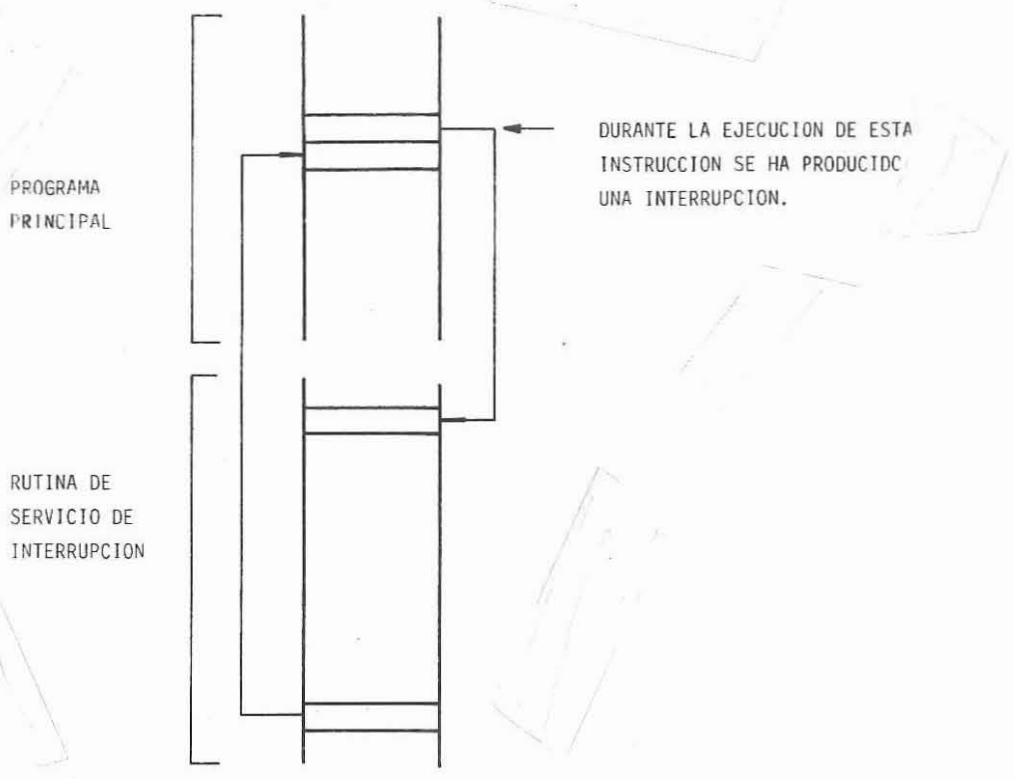


Figure 2.12. Rutinas de interrupción

El segmento de programa que se ejecuta como consecuencia de una interrupción se denomina rutina de interrupción. Como en las subrutinas, la **rutina de interrupción** debe finalizar con una instrucción de **retorno de interrupción.**

También al igual que las subrutinas, antes de saltar a la rutina de interrupción, la CPU debe guardar los datos necesarios para que luego pueda reanudar la tarea donde la dejó. Este proceso es análogo al que se realiza para las subrutinas.

Cada tipo de microprocesador admite un determinado conjunto de interrupciones. Estas interrupciones se conectan a distintos elementos del sistema, a los que el microprocesador debe dar servicio cuando éstas se produzcan.

Para evitar la posibilidad de que uno de los periféricos bloquee el sistema por una petición frecuente de interrupción, se establecen una serie de prioridades entre los distintos tipos de interrupción. De esta forma, si dos interrupciones se producen simultaneamente, se atiende en primer lugar a la de mayor prioridad.

Otro mecanismo frecuentemente disponible es el de enmascaramiento de interrupciones. Se trata de evitar que determinado tipo de interrupciones se produzca durante la

ejecución de un segmento de programa. Para ello se dispone de instrucciones específicas de bloqueo o permisión de interrupciones, o bien, se tiene la posibilidad de formar "máscaras" que indiquen, sobre un posible registro de interrupciones, aquellas que pueden admitirse en cada momento.

2.3 ELENEMTOS DE ENTRADA/SALIDA

Los elementos de entrada/salida son el medio que emplea el microprocesador para comunicarse con el exterior. Se utilizan para obtener los datos y suministrar los resultados de un determinado proceso.

Estos elementos de transferencia de información con el exterior, se denominan **periféricos.** Ejemplos típicos de periféricos son una pantalla, un teclado, una impresora, etc.

La salida de un dato se reducirá, desde el punto de vista del microprocesador, a depositar dicho dato en el elemento de entrada/salida. A partir de aquí el elemento exterior se encargará de recoger el dato y tratarlo del modo que sea preciso.

De forma similar se produce la entrada de datos. En estas operaciones el microprocesador no tiene más que recoger los datos que el periferico ha cargado en el elemento de entrada/salida.

Previamente a estas operaciones simples de lectura/escritura, si el periférico es programable, el microprocesador deberá configurar al periférico de forma que interprete correctamente la información transferida. Si el elemento de entrada/salida no es programable, o la estructura con que debe transferirse la información es fija, el microprocesador deberá adaptarse, mediante un programa adecuado, al formato de datos requerido.

La comunicación entre los elementos de entrada/salida y el exterior puede ser, básicamente, de dos tipos: paralelo o serie.

2.3.1 Transferencia en paralelo

Una **transferencia en paralelo** se efectúa cuando se emplea más de una línea de entrada/salida y se suministran, o se reciben, bits de información al mismo tiempo por cada línea. Si el bus de datos tiene 8 líneas, el número máximo de bits que se pueden presentar al mismo tiempo a la salida o que se pueden leer a la vez es de 8.

Un elemento de entrada/salida se emplea básicamente para dos fines: el control individual de elementos externos y la comunicación entre ordenadores o entre ordenadores y periféricos.

En el primero normalmente cada línea tiene su significación propio, sirve para el control de un determinado elemento (relés,

conmutadores, etc).

En el segundo la comunicación se suele efectuar byte a byte, agrupando 8 líneas de salida para transmitir los diferentes bits simultáneamente. Un ejemplo típico de transmisión de este último tipo puede ser efectuada desde un ordenador a una impresora. La simple transmisión de los datos no es suficiente ya que hay que compensar las distintas velocidades de transmisión y recepción que tiene cada unidad interconectada, indicar cuando hay dato válido, etc. Esto requiere el empleo de unas líneas adicionales que adapten la transmisión de datos entre los elementos que efectúan la comunicación. A esta operación se la denomina **Handshake.**

A grandes rasgos el proceso de Handshake consiste en:

a. El receptor indica que está libre para aceptar datos.

b. El transmisor coloca el dato en las líneas correspondientes e indica que hay un dato válido.

c. El receptor toma el dato enviado e indica que lo ha recogido.

d. El transmisor acepta esta información, espera la señal de que el receptor está libre para aceptar datos y se repiten los apartados anteriores.

De esta manera, podemos compensar asincronismos entre las unidades. Si por ejemplo, un elemento receptor lento (una impresora) está ocupado (escribiendo una línea), mantiene una línea adicional utilizada para el handshake en "ocupado" y el elemento transmisor (ordenador) ha de esperar a que esta línea pase a indicar "libre" para enviarle un nuevo dato.

El grado de complejidad de un proceso de transferencia varía bastante, en función del grado de seguridad que se requiera en la transferencia de información y de la velocidad a que ésta deba efectuarse. Tampoco el número de señales auxiliares o de control que se requieren es fijo.

Al nivel básico, todas las transferencias de información de la CPU con los circuitos de control de periféricos se realizan a través del Bus de Datos y, por tanto, en paralelo.

La principal ventaja de la transferencia de datos en paralelo está en la mayor velocidad de intercambio de datos que permite. También evita la necesidad de convertir la información a otro formato (p.e. serie) ya que originariamente se genera en paralelo. Una limitación de este tipo de transmisión en paralelo de datos está en la máxima distancia a la que puede realizarse,

que se limita en función de la velocidad de transferencia, y no suele ser superior a 1 metro.

2.3.2 **Transferencia en serie**

La transferencia de datos en un formato serie proviene de la necesidad de reducción al máximo del número de líneas utilizadas para transmitir una información. Este hecho se suele dar cuando las características del diseño así lo imponen o cuando la transferencia debe realizarse a distancias considerables (p.e. conexión entre pantallas y ordenador central) Con este tipo de transferencia las distancias máximas alcanzables son de 1 a 2 órdenes de magnitud superiores al formato paralelo, y resulta facil configurar las señales de partida para utilizar otros medios de propagación, como la línea telefónica o la radio, para conseguir enlaces de datos a grandes distancias.

La transferencia de datos en serie implica una transmisión bit a bit de los valores que codifican la información. Por tanto requerirá, al menos, una línea para cada uno de los sentidos de transmisión y una de masa para determinar la tensión de referencia. Generalmente, y en función del formato utilizado, se emplean algunas líneas auxiliares para controlar en que momento cada uno de los extremos de la línea de datos está preparado para transmitir o recibir, o para sincronizar, con una señal de reloj, cada uno de los bits transmitidos.

De forma equivalente, se pueden transmitir con los bits que forman el dato de información, otros bits adicionales para asegurar la sincronización y la integridad del proceso de transferencia.

Existe un amplio abanico de normativas y estandars que regulan el formato y los procedimientos de utilización de las transferencias de datos en serie. El seguimiento de estas normas, aunque no es preceptivo, facilita enormemente el entendimiento entre distintos sistemas y ordenadores, por lo que siempre son recomendables. En el caso de la transmisión serie, se suele seguir la norma RS-232C que especifica los tipos de señales, el conector y la distribución de conexiones en el mismo, y un conjunto de características eléctricas y mecánicas a seguir para conectar dos sistemas mediante un enlace de datos en serie.

La velocidad de transmisión se mide en **Baudios,** término que equivale al número de bits transmitidos por segundo, sin tener en cuenta que sean de dato, de start o de stop. Estas velocidades están normalizadas, siendo típicos los valores de 110, 300, 1200, 2400, 4800 y 9600 baudios, entre otros.

Existen dos procesos básicos de transmisión serie, denominados **asíncrono** y **síncrono** Aunque, evidentemente, es preciso en todos los casos que el transmisor y el receptor se sincronicen por algún procedimiento para que la transferencia de información sea efectiva, ambos procesos se diferencian en la forma como se realiza dicha sincronización. En la transmisión asíncrona, cada dato contiene información adicional para su sincronización e incluso para una eventual detección de error de transmisión. Por tanto se sincroniza dato a dato.

En modo síncrono la información se transmite en paquetes de datos. Estos paquetes transportan la información para detectar todos los datos contenidos y detectar posibles errores cometidos. La sincronización se realiza por bloques de información y no es preciso añadir bits auxiliares a cada dato transmitido.

Por lo general, cuando se habla de transmisión serie, se entiende que la información transferida está formada por un conjunto de carácteres codificados en ASCII, por lo que la unidad de información transmitida es el carácter.

Debido a la complicación de las estructuras hardware necesarias para efectuar este tipo de transmisiones, se han integrado en una sóla pastilla LSI los componentes necesarios para llevarlas a cabo. Las denominaciones de los fabricantes dadas a estos tipos de circuitos integrados son:

- ACIA (Asynchronous Communications Interface Adapter)
- UART (Universal Asynchronous Receiver Transmitter)
- USART (Universal Synchronous-Asynchronous Receiver Transmitter)
- SIO (Serial Input-Output)

2.3.2.1 Transmisión en serie asíncrona

Este tipo de transmisión es asíncrona dado que los carácteres se transmiten aisladamente, pudiendo variar el tiempo que separa un carácter del siguiente. No obstante, una vez se ha detectado la llegada de un nuevo carácter, el receptor debe empezar a muestrear el estado de cada bit del carácter comportándose así de forma síncrona.

En la figura 2.13 puede verse el formato de una transmisión asíncrona.

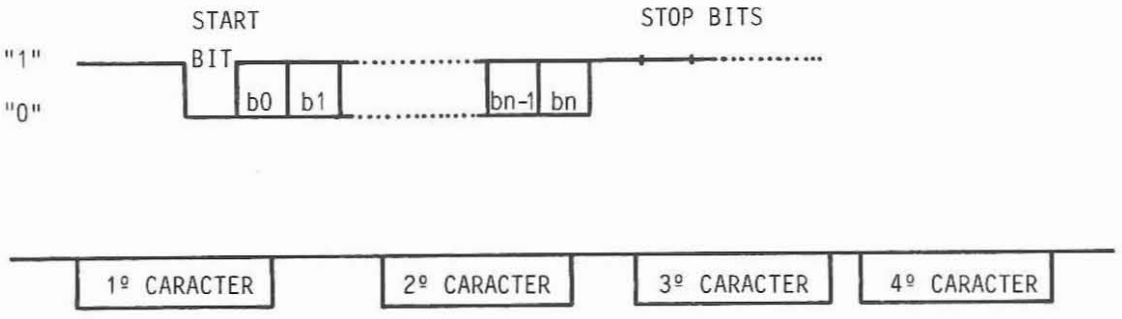


Figura 2.13. Formato de una transmisión asíncrona

El formato de un carácter transmitido mediante una comunicación de este tipo contiene un cierto número de parámetros variables, que suelen ser función del tipo de datos a transmitir y de la velocidad a que se efectúe la transmisión, y de otros parámetros fijos que permiten el proceso de sincronización. El formato básico es el siguiente:

- Cada carácter va precedido por un Bit de Arranque o **Start Bit** que se detecta porque la línea de transmisión, que en estado de reposo está a nivel alto, pasa a nivel bajo. Este Bit de Arranque es de una duración fija e igual al tiempo de un bit.

- Al final del carácter, antes de que se pueda transmitir el siguiente, la línea debe permanecer en estado alto durante el tiempo equivalente a uno, uno y medio o dos bits como mínimo. Estos bits reciben el nombre de **Stop Bits** o Bits de Parada.

- La información de cada carácter consta de 5, 6, 7 u 8 bits. El primer bit transmitido corresponde al bit de menor peso del carácter y el último al de mayor peso.

- Opcionalmente se incluye al final del dato y antes del bit de parada, un **Bit de Paridad** que indica si el número de '1' en el carácter es par o impar. Esta información se puede utilizar en recepción para detectar posibles errores de transmisión.

Antes de efectuar la transmisión el receptor debe conocer cuales son las características con que se efectúa la transmisión. Se debe especificar:

- Cual es la velocidad de transmisión.

- Cual es el número de bits de cada carácter.

- Si se utilizará un bit de paridad para detección de errores de transmisión y el tipo de paridad (par o impar).

- El número de bits de parada o Stop bits.

2.3.2.2 Transmisión en serie síncrona

En la transmisión síncrona los datos o carácteres no se envían aisladamente sino formando bloques de gran longitud (paquetes de datos). Al no existir separación entre los carácteres de este bloque la sincronización a nivel de carácter debe efectuarse contando el número de bits que le corresponden a cada uno. Para ello debe ser capaz de detectar un primer carácter con el que sincronizar los siguientes. Esta es una de las razones por las que al comienzo de cada bloque se transmiten varios carácteres de sincronismo.

El formato de una transmisión síncrona es el de la figura 2.14.



Figura 2.14. Formato de una transmisión síncrona

Este formato se denomina **Bisync** por utilizar al inicio de cada bloque de información dos caracteres de sincronismo y es uno de los métodos más utilizados. Normalmente se utiliza como carácter de sincronismo el SYN del código ASCII (16H en Hexadecimal) aunque puede programarse, tanto en el receptor como en el emisor, cualquier combinación de dos carácteres ASCII.

En la transmisión síncrona no es posible trabajar con un oscilador de reloj en el receptor independiente del reloj del transmisor. La transmisión debe efectuarse al ritmo de un solo reloj de forma que al receptor le llegue información exacta de la frecuencia utilizada en el transmisor.

Para lograr esta identidad de frecuencias pueden utilizarse dos sistemas. Uno consiste en utilizar una conexión adicional que permita la transmisión directa de la señal de reloj (figura 2.15). El otro procedimiento consiste en la utilización de algún sistema de modulación que haga llegar al receptor la información sobre la frecuencia de reloj original. Los dos carácteres de sincronismo que se transmiten previamente a cada bloque tienen como misión, además de efectuar el sincronismo de palabra como se ha indicado anteriormente, indicar en qué momento se inicia la recepción del paquete.

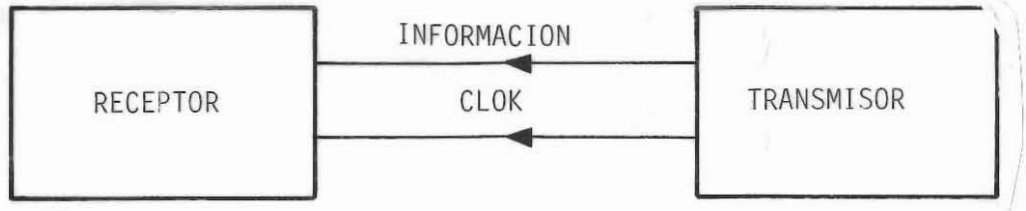


Figura 2.15. Transmisión de la señal de reloj

Puede decirse que en transmisión síncrona el sincronismo se realiza a dos niveles:

- a nivel de bit, mediante la transmisión de la señal de reloj junto a la de información.

- a nivel de bloque de información, mediante los carácteres de sincronismo que indican al receptor en que momento debe empezar a interpretar la cadena de bits recibidos como inicio de datos.

En la mayoría de sistemas de transmisión síncrona, se prevee una zona al final de cada bloque que permite incluir unos códigos de detección de errores en el bloque de información. La generación de estos códigos se basa en la evaluación de unos polinomios. La complejidad de esta opción impone el uso de circuitos de control específicamente diseñados para este tipo de transmisión.

En líneas generales, este tipo de transmisión se utiliza, preferentemente, en el enlace entre ordenadores que comparten una serie de canales de transmisión, formando una red, o en enlaces a grandes distancias por líneas de transmisión sometidas a tarificación. En estos casos, el empleo del bloque de datos como unidad de información resulta ventajoso frente a la transmisión asíncrona (carácter a carácter), tanto a nivel económico como de eficacia.

PRACTICA 2.1 ACTUACION DE LAS INTERRUPCIONES

En esta práctica vamos a comprobar el funcionamiento de las interrupciones. Como en la práctica 1.1 vamos a utilizar un programa que ya está grabado en la memoria del Microinstructor de forma que no sea necesario entrar uno nuevo.

a. Pulsar: **[GO] [F] [3] [C] [A] [FIN]**

   En este instante se iniciará la ejecución del programa secuenciador luminoso.

b. Mediante un puente eléctrico unir la patilla 9 de J9 (/IRQ) con la línea de masa (GND).

c. En este momento el microprocesador interrumpe el programa que estaba ejecutando y salta a la rutina de servicio de interrupción. En el campo de direcciones del display aparece el estado del Contador de Programa, indicando la dirección en la que se encontraba el microprocesador al producirse la interrupción. En el campo de datos aparece el mensaje "ie".

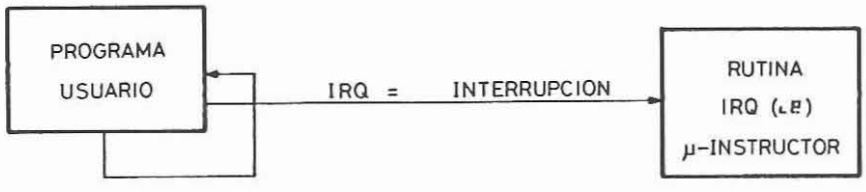


Figura 2.16. Actuación de una interrupción

CAPITULO 3

EL MICROINSTRUCTOR MI-650C

El Microinstructor MI-650C es un microcomputador basado en una versión CMOS del microprocesador de 8 bits 6502. Aunque está destinado fundamentalmente a fines educativos y de evaluación de la citada CPU, dadas sus características puede emplearse tambien como equipo de control para determinadas aplicaciones. Con este fin se ha diseñado el equipo de modo que todas las señales sean accesibles, bien a través del Conector de Expansión, bien a través del Conector de Aplicación.

Como todo microcomputador en el Microinstructor pueden distinguirse tres partes principales. Como ya se vió en el capítulo anterior estos tres elementos son: la memoria, la CPU y los elementos de entrada/salida.

Sobre la placa de circuito impreso del equipo se pueden distinguir dos serigrafías. La blanca incluye información acerca de la localización de los diversos elementos del circuito; la amarilla divide el equipo en una serie de bloques funcionales, dentro de los cuales se encuentran los componentes a él asignados.

En la figura 3.1 puede verse el diagrama de bloques del Microinstructor.

3.1 **LA MEMORIA DEL MICROINSTRUCTOR**

El Microinstructor MI-650C se suministra con 8 Kbytes de memoria RAM y 8 Kbytes de memoria ROM.

En la memoria EPROM 2764 (IC17) que se suministra con el equipo se encuentra el programa monitor, que se estudiará detalladamente en el capítulo 6. También en esta memoria se encuentran grabadas algunas de las prácticas que se exponen en el capítulo 7 y los programas propuestos como ejemplo de los Módulos de Aplicación del Microinstructor (MM-6XX). En el apéndice B se describe brevemente cada uno de estos módulos y en el apéndice C se detallan las direcciones de memoria en las que se encuentran grabados todos estos programas.

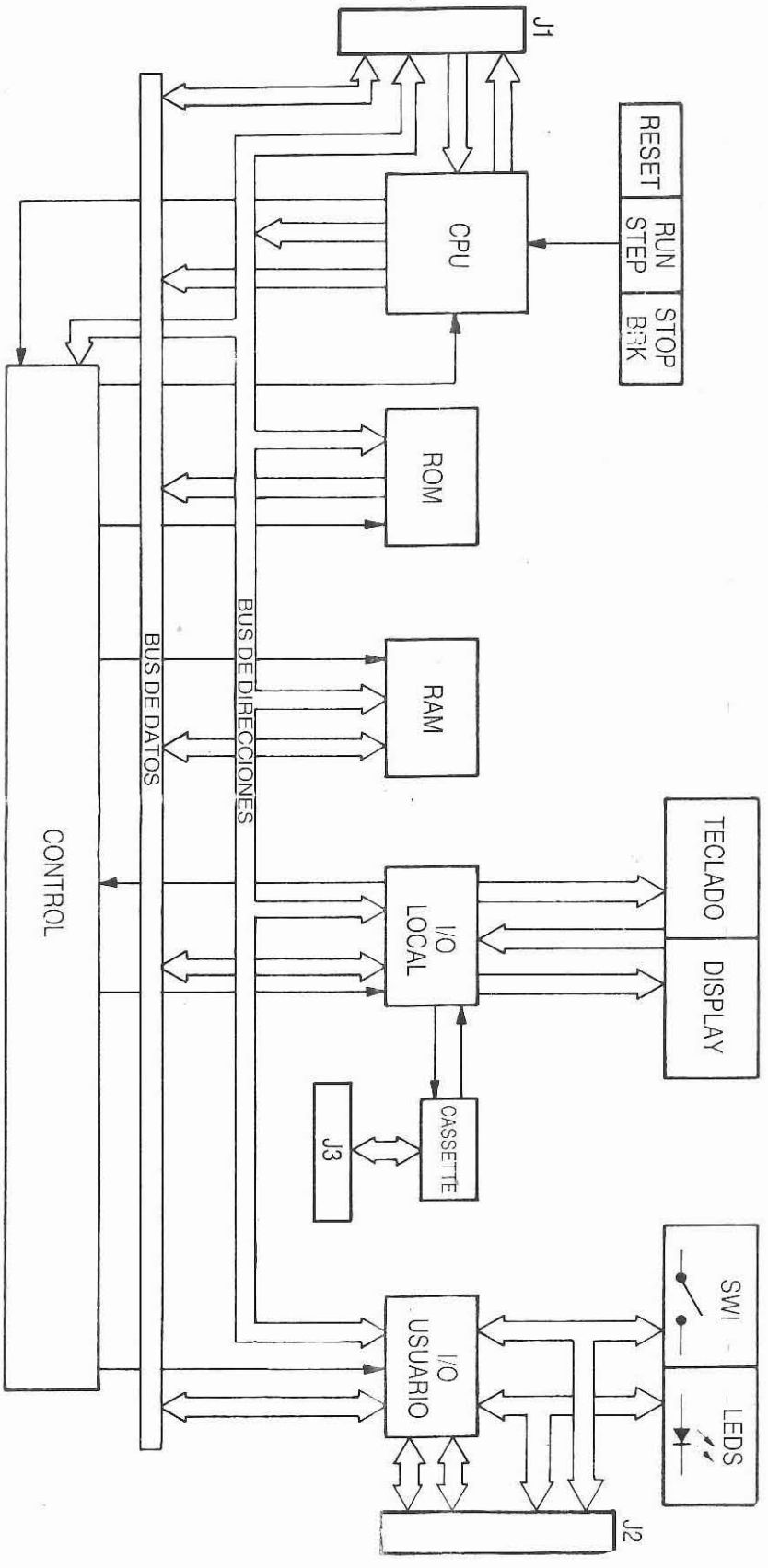


Figura 3.1. Diagrama de bloques del Microinstructor.

La capacidad de memoria puede ser ampliada directamente a través del zócalo de memoria Auxiliar. Esta memoria puede seleccionarse RAM ó EPROM y de una capacidad de 2 ó de 8 Kbytes. Las conexiones programables J9 y J6 permiten selecionar los parámetros anteriores.

Existe otro selector (J5) que que inhibe el direccionamiento del zócalo Auxiliar, para poder seleccionar otro zócalo exterior de memoria en la misma posición del mapa de memoria del Microinstructor. Este selector debe estar normalmente en la posición INT. Cuando se conecta la impresora (MM-613) al Microinstructor, este selector ha de pasarse a la posición EXT. Esto es debido a que en el mapa de memoria de la impresora y del Microinstructor coinciden las direcciones de la memoria Auxiliar en el Microinstructor con la EPROM que contiene el programa que gobierna la impresora.

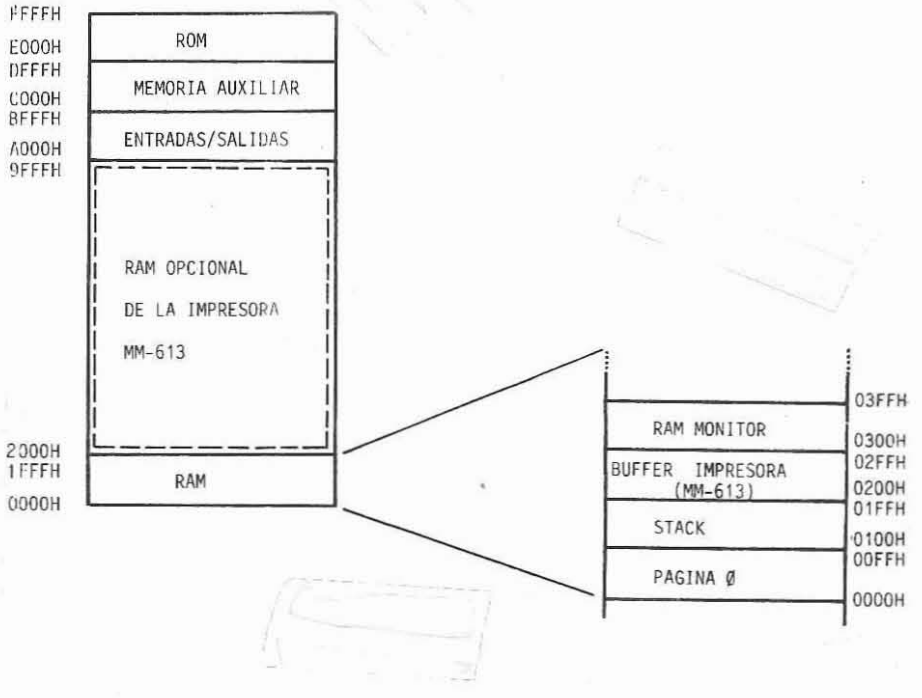


Figura 3.2. Mapa de memoria del Microinstructor

Los 64 Kbytes de memoria direccionables mediante las 16 líneas del bus de direcciones se dividen en 8 zonas de 8 Kbytes cada una. Esto se consigue mediante la decodificación de las 3 líneas de mayor peso del bus de direcciones (A13, A14 y A15), mediante el circuito integrado 74HC138 (IC1). De las 8 salidas de este primer decodificador, cuatro son aprovechadas directamente por el equipo.

Con esta primera división el mapa de memoria queda del modo que se muestra en la figura 3.2. Las distintas áreas en que queda dividida la memoria serán:

- Area 0000H a 1FFFH ---> Asignada a RAM
- Area A000H a BFFFH ---> Asignada a I/O
- Area C000H a DFFFH ---> Asignada a memoria auxiliar
- Area EFFFH a FFFFH ---> Asignada a ROM

La zona de 8 Kbytes destinada a dispositivos de entrada/salida (I/O) se divide en 8 áreas de 1 Kbyte cada una. Esto se logra mediante un segundo decodificador de 8 salidas 74HC138 (IC7) que utiliza como entradas las siguientes 3 líneas del bus de direcciones (A12,A11 y A10).

Las dos primeras señales (/SEL0 y /SEL1) se emplean para la selección de los dos circuitos LSI de entrada/salida existentes en el Microinstructor. La señal /SEL1 selecciona la VIA ("Versatile Interface Adapter") 65C22 (IC19) que utiliza el Microinstructor para controlar el teclado y el display. La señal /SEL0 por su parte selecciona una segunda VIA 65C22 (IC22) que se encarga de controlar las líneas de entrada y de salida accesibles a través del Conector de Aplicación del Microinstructor y que actúan sobre los LEDs DL4 a DL11 y sobre los microinterruptores SW23. En la figura 3.3 se puede ver el esquema electrónico del circuito de decodificación del Microinstructor.

Las líneas correspondientes a las dos zonas siguientes /PERI0 y /PERI1 se envían al Conector de Expansión del Microinstructor de manera que se puedan seleccionar dispositivos externos como parte del mapa de memoria del Microinstructor.

Esta zona de memoria quedará del siguiente modo:

- Area A000H a A3FFH ---> SEL1
- Area A400H a A7FFH ---> SEL0
- Area A800H a ABFFH ---> PERI0
- Area AC00H a AFFFH ---> PERI1
- Area B000H a BFFFH ---> No se utiliza

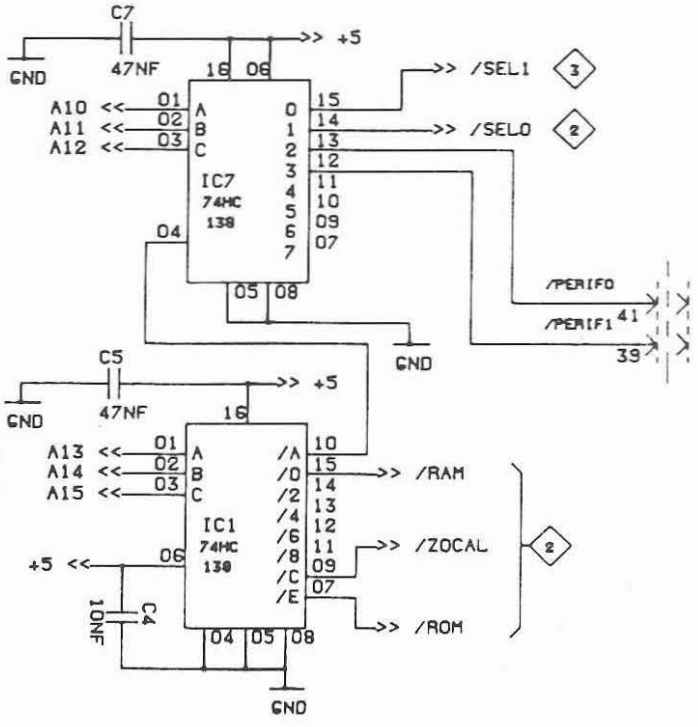


Figura 3.3. Esquema del circuito decodificador

Aunque las áreas de memoria ocupadas para la selección de estos elementos de entrada/salida son mucho mayores de las que realmente necesitan, esta es la solución que parece más adecuada debido a la simplicidad de los circuitos de decodificación.

3.2 EL MICROPROCESADOR 65C02

3.2.1 Introducción

El Microinstructor MI-650C está basado en una versión CMOS del microprocesador 6502, que incorpora una serie de mejoras de hardware y sorftware sobre su predecesor, sin perder la compatibilidad. Existen dos versiones CMOS para esta familia de microprocesadores, el 65SC02 de GTE, y el 65C02 de ROCKWELL. Este último está dotado de dos nuevas instrucciones de tratamiento de bits, en relación al anterior. Por lo demás, ambas versiones son prácticamente idénticas. Por ello, el MI-650C se suministra con cualquiera de los dos, refiriéndose a ellos en el texto como 65C02 únicamente.

El 6502 es un microprocesador de 8 bits (ALU y bus de datos) dotado de un bus de direcciones de 16 bits, que le permite acceder a 64k bytes de memoria. El acceso a memoria incorpora el concepto de paginación que consiste en suponer que el mapa de memoria está fraccionado en páginas. Cada una de ellas se direcciona a través del byte alto del bus de direcciones, y se obtiene, por el byte bajo, la dirección completa de la posición de memoria seleccionada. Debido a este concepto de paginación, las instrucciones de salto condicional, pensadas para trabajar dentro de la misma página de memoria, tienen un tiempo de ejecución menor que si trabajaran con direcciones absolutas. No obstante, el tiempo de ejecución se incrementa en un ciclo de reloj cuando el salto relativo se produce fuera de la página activa.

Dos de estas páginas del mapa de memoria tienen un significado especial para la CPU. La "página 0" que corresponde a las direcciones absolutas entre 0000H y 00FFH (byte alto de direcciones = 00) permite accesos para lectura/escritura de datos, sensiblemente más rápidos que en instrucciones equivalentes con direccionamiento absoluto. La "página 1", situada entre las posiciones 0100H y 01FFH, es la zona reservada para implementar el stack, que el microprocesador trata de forma automática como memoria de acceso secuencial.

Otra característica de este microprocesador es el reducido juego de instrucciones que presenta, en relación a otros microprocesadores de 8 bits, potenciado por el gran número de direccionamientos posibles en cada una de ellas. De este hecho resulta un tiempo de aprendizaje reducido además de la facilidad de manipulación de datos en memoria.

La versión CMOS de este microprocesador, utilizada en el Microinstructor MI-650C presenta algunas mejoras respecto de la

versión NMOS. Cabe destacar la inclusión de 12 nuevas instrucciones, con un total de 59 nuevos códigos de operación; la disminución del consumo, debido a la tecnología empleada; y la eliminación de algunos errores que se habían detectado en las versiones NMOS.

3.2.2 **Estructura hardware interna**

En la figura 3.4 se presenta el diagrama de bloques de esta CPU.

Las funciones y características de cada uno de estos bloques y señales son las siguientes:

a. **Registro de instrucción:** Como ya se ha explicado anteriormente este registro es el encargado de memorizar durante la ejecución, la instrucción que se ha recogido de memoria mediante un ciclo de búsqueda (fetch). Las instrucciones de este microprocesador tienen entre 1 y 3 bytes.

b. **Decodificador de instrucciones:** Durante el estado de ejecución, este bloque se encarga de decodificar el código de operación de la instrucción actualmente almacenada en el registro de Instrucción, y generar la secuencia de microórdenes asociada a la función a realizar, sincronizada con la señal de reloj. En el apéndice A se detallan cuales son los pasos que realiza cada instrucción con lo cual se podría deducir la secuencia de microórdenes que lleva asociada cada instrucción.

c. **Unidad Aritmético-Lógica:** En este bloque tienen lugar las operaciones aritméticas y lógicas que han de realizarse en el microprocesador.

   Según se especifique en el programa, estas operaciones pueden efectuarse, bien sobre un registro aislado (por ejemplo, incrementando el registro X), o bien entre dos operandos, uno de los cuales siempre es un registro de la CPU.

   Para las operaciones aritméticas existen dos modos de funcionamiento. Uno de ellos considera los dos operandos como números binarios El otro, modo decimal, considera cada operando como dos números BCD, dando también un resultado en BCD. Más adelante se verá cómo se indica el modo de trabajo de la ALU. Después de un reset o de una interrupción el modo de funcionamiento de la ALU se inicializa a modo binario (sólo en el 65C02).

d. **Registros:** El número y el propósito de los registros internos de la CPU varía mucho de un microprocesador a otro. Los

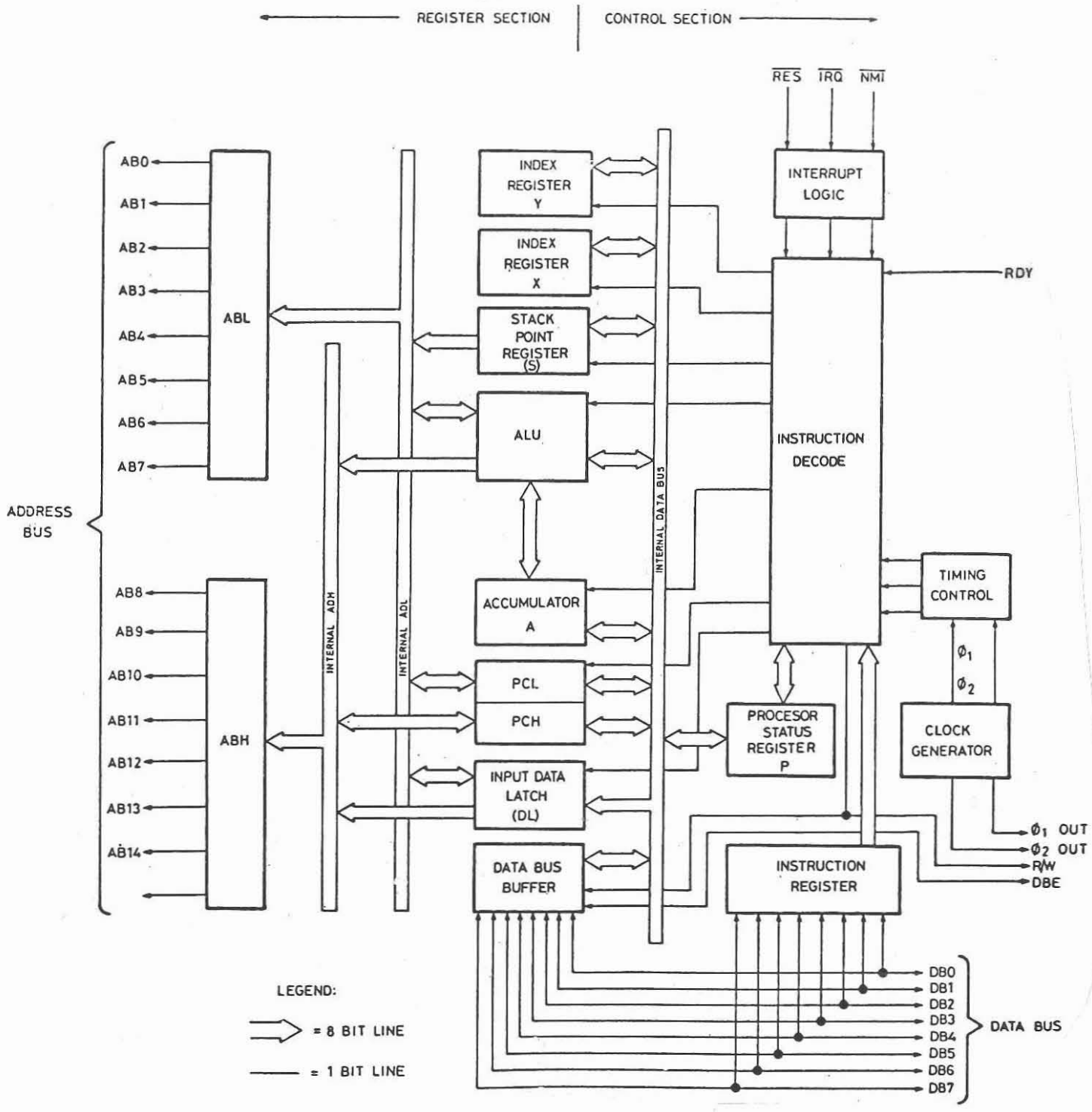


Figura 3.4. Estructura del 65C02

registros del 65C02 pueden verse en la figura 3.4 y sus funciones son las siguientes:

- **Contador de programa (PC):** Este es el único registro de 16 bits que incorpora el microprocesador 65C02. En él se mantiene la dirección del siguiente byte de programa que debe ser leido para su ejecución. Cuando se inicia el ciclo de búsqueda de la instrucción el contenido del PC se deposita en el bus de direcciones. Inmediatamente se incrementa el contenido de este registro. Este proceso se repite hasta que se recoge algún tipo de instrucción que modifica el contenido del PC, lo cual permite continuar el programa a partir de otra dirección.

- **Acumulador:** En este microprocesador la mayor parte de transferencias entre memoria y CPU se efectúan a través de este registro.

- **Indices (X,Y):** Pese a que estos dos registros están especialmente indicados para actuar de índices en referencias a memoria, tambien se pueden utilizar como contadores, registros para el almacenamiento temporal de datos, etc.

- **Stack Pointer o Puntero de Pila (S):** Como se ha expuesto anteriormente, el stack se crea en la página 1 del mapa de memoria. En ella, el byte alto del bus de direcciones es constante e igual a 01H, por tanto, es suficiente un registro de 8 bits para formar el Puntero de Pila. Este registro contiene el byte de menor peso de Puntero. En el programa monitor del Microinstructor el registro S se inicializa con el valor FFH, con el fin de aprovechar al máximo la página 1, empleando de este modo como área para el Stack las direcciones comprendidas entre 100H y 1FFH.

- **Registro de Estado o Status (P):** En este microprocesador este registro es también de 8 bits. Cada uno de ellos tiene un significado propio y refleja el estado de la CPU, como indica su nombre. Cada uno de estos bits, llamados **Flags o Indicadores** tiene la misión de señalar la ocurrencia de un hecho concreto.

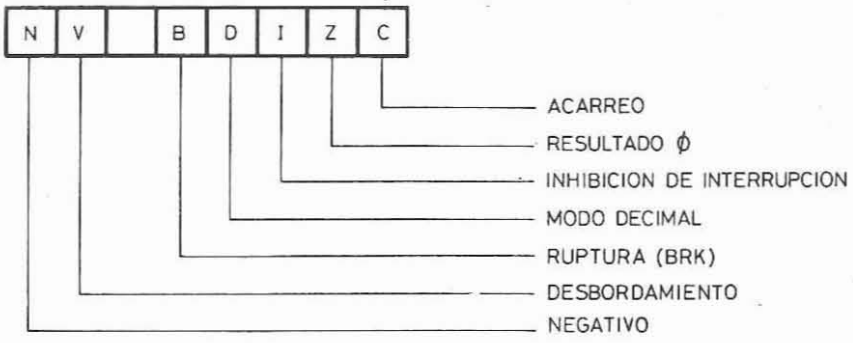


Figura 3.5. Registro de Estado (Status)

En el apartado de descripción del juego de instrucciones puede estudiarse con más detalle la misión de cada uno de estos flags.

e. **Bus de direcciones:** Está compuesto por los terminales A0 a A15. Son salidas con niveles lógicos TTL, capaces cada una de ellas de excitar una carga de una puerta TTL estandar. En estos terminales siempre está presente una dirección, correspondiente al contenido de los "latches" ABL y ABH.

f. **Bus de datos:** Está formado por los terminales DB0 a DB7. Son bidireccionales y el estado normal es el de alta impedancia (Tri-state) cuando no se realiza ningún ciclo de lectura o escritura.

g. **Señal de lectura/escritura (R/W):** Indica a la memoria y a los elementos de entrada/salida si se trata de una operación de lectura (R/W=1) o de escritura (R/W=0).

h. **Reloj (clock):** La señal de reloj en el Microinstructor se genera mediante un cristal de cuarzo de 1 MHz. Esta señal se aplica en la patilla ∅0. En el interior de la CPU existen unos circuitos que, a partir de ∅0, generan las dos fases ∅1 y ∅2 que aparecen en los terminales correspondientes para servir de reloj a otros elementos periféricos del sistema. Durante el tiempo en que ∅1=1, se producen los cambios en el bus de direcciones mientras que cuando ∅2=1 es cuando se producen las transferencias de datos. En la figura 3.6 se puede ver la temporización de estas señales.

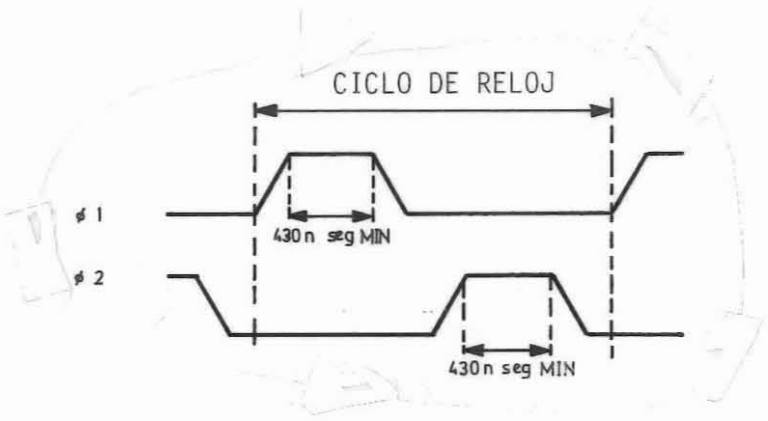


Figura 3.6. Ciclo de reloj.

i. **Señal de 'Ready' (RDY):** Esta señal está pensada para retardar la ejecución de un ciclo de búsqueda de instrucción, durante un ciclo de reloj. Con ello se hace posible acceder a memorias (o, en general, a periféricos) lentos sin tener que disminuir la frecuencia de reloj del microprocesador y, por tanto, su velocidad de trabajo. Cuando se aplica un flanco negativo a RDY durante Ø1=1, se reconoce el estado de esta señal durante Ø2=1, y se congela la ejecución del ciclo interno de la CPU, hasta el siguiente ciclo de reloj. Por tanto se duplica el tiempo de duración del estado.

j. **Señal de Sincronismo (SYNC):** Se trata de una señal que genera la CPU con el fin de señalar aquellos ciclos de reloj durante los cuales se produce la lectura del código de operación de una instrucción. Se utiliza en combinación con la entrada RDY para la ejecución "instrucción por instrucción". Para ello se provoca la detención del procesador mediante RDY cuando se detecta el flanco de subida de SYNC, indicando que va a comenzar una nueva instrucción. Este paro se produce dejando las salidas de direcciones señalando la posición del programa que contiene el código de operación de la siguiente instrucción a ejecutar.

k. **Línea de 'Set Overflow' (S.O.):** Esta entrada sirve para poner a "1" el bit de "overflow" del registro de "Status". Puede usarse para dar servicio a un periférico que active esta línea en un programa que efectúe continuas comprobaciones del Flag V del registro de Status.

l. **Reset (RES):** Esta señal se utiliza para inicializar el microprocesador, tras la conexión de la alimentación al equipo. Durante el tiempo de establecimiento de la tensión de alimentación, la señal de reset debe permanecer a '0' y, al pasar a '1', la CPU carga el Contador de Programa (PC) con el vector almacenado en las direcciones FFFCH-FFFDH e inicia la ejecución normal de un programa. En el Microinstructor en

este vector se encuentra la dirección de inicio del Monitor. Una de las primeras funciones que realiza el Programa Monitor es la de sacar un mensaje inicial por el display (650 C) y realizar un test de los LED's y de los interruptores conectados a uno de los periféricos del Microinstructor. Durante este test se encienden los LED's uno a uno, secuencialmente, y se lee el estado de los 8 microinterruptores. Al término de la subrutina de test, el estado de los 8 microinterruptores se transfiere a la fila de LED's. Posteriormente se pasa a un bucle en el que se va refrescando el display y se analiza el teclado para detectar la petición de funciones.

m. **Interrupción enmascarable (/IRQ):** Un nivel bajo de esta línea da lugar a una secuencia de interrupción, en la que se efectúan las siguientes operaciones:

- Se finaliza la instrucción que se estaba efectuando.
- La CPU analiza el contenido del bit I del registro P. Si el valor de este bit es 1, se ignora la interrupción y continúa el trabajo normal. Si, por el contrario, su valor es 0, la interrupción es aceptada y continúa el proceso.
- Impide otra interrupción y guarda el valor del contador de programa (PC) y el registro P en el Stack.
- Recoge el vector en FFFEH-FFFFH y comienza la ejecución en la dirección de memoria indicada por éste.
- La ejecución finaliza cuando se encuentra una instrucción RTI, tras la cual recoge los datos antes guardados y continúa la operación en que se detuvo.

n. **Interrupción no enmascarable (/NMI):** Esta interrupción obliga al procesador a alterar el contenido del PC, independientemente del flag I del registro de Estado. Origina el salto de la ejecución del programa a la dirección contenida en las posiciones FFFAH y FFFBH, previa salvaguarda del contador de programa y del registro de Estado en el Stack.

La activación de la línea /NMI se logra manteniéndola en el estado bajo por lo menos durante dos ciclos de reloj, pudiendo permanecer en este estado indefinidamente sin que se produzcan nuevas interrupciones, ya que éstas se originan únicamente por el flanco de bajada (paso de "1" a "0") de /NMI.

El modo de tratamiento de las interrupciones (tanto NMI como IRQ) se efectúa de manera análoga a las subrutinas. La

diferencia está en que, cuando se atiende una interrupción, se guarda en el Stack, además del PC, el registro Status (P). Otra diferencia es que el salto a la rutina de interrupción se efectúa a partir de un vector fijo en memoria. De igual manera que existe una instrucción para indicar que se ha llegado al final de una subrutina y que debe devolverse el control al programa principal, existe una instrucción específica (RTI) para finalizar la rutina de atención de la interrupción. La única diferencia estriba en que RTI recupera, además del PC, el registro Status.

En cualquier hoja de datos o manual técnico de este microprocesador pueden estudiarse con detalle los diagramas de temporización de las señales de este microprocesador. Cada lectura o escritura en memoria dura un ciclo de reloj, por lo que la duración de la ejecución de una determinada instrucción, depende del número de accesos a memoria (lectura o escritura) que requiera.

Durante la ejecución de un programa se alternan ciclos de lectura, ciclos de escritura y ciclos en los que no se realiza ninguna operación sobre memoria, durante los cuales la CPU trabaja internamente. En la figura 3.7 podemos ver el diagrama de las señales que aparecerían al ejecutar una instrucción de carga del Acumulador, en los sucesivos accesos a memoria, hasta completar su ejecución.

El número de ciclos de reloj que tarda en ejecutarse cada instrucción viene especificado por el fabricante, y permite calcular el tiempo total de ejecución de un programa. Las instrucciones en este microprocesador pueden tener entre 2 y 7 ciclos de reloj. Por tanto, el tiempo de ejecución de una instrucción estará entre 2 y 7 microsegundos, para una frecuencia de reloj de 1MHz.

En el primer ciclo de todas las intrucciones se realiza un ciclo de "fetch" o búsqueda del código de operación. Durante estos ciclos se activa la señal SYNC. Los restantes ciclos de cada instrucción pueden ser de lectura, de escritura o de operación interna. Durante los ciclos ocupados por este último tipo de instrucciones los buses externos al microprocesador se encuentran libres.

Como ya se ha repetido en varias ocasiones, una vez la CPU ha realizado el ciclo de búsqueda del código de operación de la siguiente instrucción a ejecutar, se inicia su decodificación y la ejecución de las diversas microoperaciones correspondientes. Dado que estas microoperaciones deben estar sincronizadas, cada una de estas acciones elementales se realizará en un ciclo de reloj. No obstante, en el 65C02, se utilizan elementos distintos para controlar la parte de generación de direcciones y la de

DIRECCION MEMORIA	MNEMONICO	DATO
.	.	.
.	.	.
.	.	.
.	.	.
.	.	.
40F	DEX	CA
410	LDA 0450H	AD
411		50
412		04
413	TAY	A8
.	.	.
.	.	.
.	.	.
450		FF
.	.	.
.	.	.
.	.	.

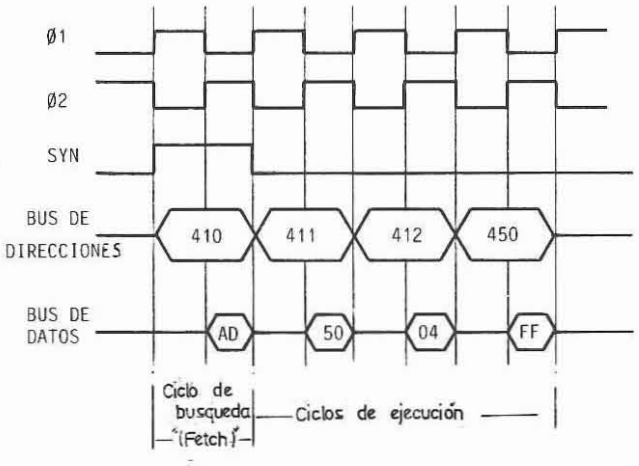


Figura 3.7. Diagrama de tiempos durante la ejecucion de una instruccion

cálculo (pipelining) por lo que se pueden realizar varias microórdenes simultaneamente en un sólo ciclo de reloj, con lo cual se acorta el tiempo total de ejecución de la instrucción.

Debido a esta ejecución en paralelo, el 65C02, pese a funcionar a una frecuencia de reloj menor a la de otros microprocesadores, puede llegar a tener un tiempo de ejecución de las instrucciones parecido e incluso inferior al de éstos.

3.2.3 Modos de direccionamiento

Una instrucción es un código binario con un significado preestablecido. Sin embargo, con el fin de facilitar al programador la memorización y el tratamiento de estos códigos, es conveniente adjudicar a cada instrucción una abreviatura que dé una idea de la acción que realiza. Esta abreviatura se denomina **Nemónico.**

Siempre debe tenerse presente que esta abreviatura sólo es una forma más clara de representar las instrucciones de cara al programador, ya que la CPU sólo entiende códigos binarios.

La instrucción que indica, por ejemplo, que se efectúe la carga del Acumulador con un valor determinado tiene en el 65C02 el código máquina:

0 1 1 0 1 0 0 1 = A9H

y se le asigna el nemónico:

LDA (Load Accumulator)

INSTRUCCION	MODO DE DIRECCIONAMIENTO	CODIGO OPERACION	LONGITUD INSTRUCCION
LDA	INMEDIATO	A9	2 bytes
LDA	ABSOLUTO	AD	3 bytes
LDA	ABSOLUTO INDEXADO X	BD	3 bytes
LDA	ABSOLUTO INDEXADO Y	B9	3 bytes
LDA	PREINDEXADO	A1	2 bytes
LDA	POSTINDEXADO	B1	2 bytes
LDA	ABSOLUTO PAGINA 0	A5	2 bytes
LDA	INDEXADO PAGINA 0	B5	2 bytes
LDA	INDIRECTO	B2	2 bytes

Figura 3.8. Modos de direccionamiento de LDA

Pero cada instrucción debe especificar también, el dato o datos con los que opera o la situación de los mismos.

La forma de indicar cuál es o dónde se encuentra un dato da lugar a los diversos **modos de direccionamiento.**

De esta forma, la mayoría de nemónicos dan lugar a varios códigos de operación según el modo de direccionamiento utilizado. Por ejemplo, la instrucción LDA (Carga del Acumulador) del 65C02 puede dar lugar a 9 códigos de dirección distintos (ver figura 3.8)

Todas estas operaciones tienen en común que realizan una operación de carga, pero difieren en la forma de encontrar el dato que se va a utilizar en la carga.

Esta gran variedad de modos de direccionamiento distintos facilita la labor del programador que deberá recordar un grupo de nemónicos más reducido que en otros microprocesadores. El 65C02 tiene un total de 68 nemónicos, que expandidos en los 15 modos de direccionamiento posibles dan un total de 210 códigos de operación.

Los modos de direccionamiento admitidos por el microprocesador 65C02 son los siguientes:

a. **Direccionamiento implícito**

Las instrucciones que utilizan este tipo de direccionamiento contienen toda la información necesaria para identificar la posición del dato con el que operan o las direcciones de transferéncia necesarias, en el propio código de operación. Por tanto, sólo tienen 1 byte de longitud.

Este modo de direccionamiento es propio de instrucciones que afectan a un solo registro interno o que efectúan transferencias de datos entre los registros o entre éstos y el "Stack". De las 68 instrucciones, 29 utilizan este modo de direccionamiento.

Un ejemplo de este tipo de direccionamiento puede ser:

```
TAX              Transfiere el Acumulador al registro X
```

b. **Direccionamiento de Acumulador [Ac]**

En este modo de direccionamiento los datos se encuentran en el Acumulador antes y después de la ejecución de la instrucción. Estas instrucciones también están formadas por un solo byte.

En el 65C02 utilizan este tipo de direccionamiento las instrucciones que implican un desplazamiento de bits de los datos del Acumulador y las que lo incrementan o decrementan.

Un ejemplo de este tipo de direccionamiento puede ser:

```
DEC A          Decrementa A y deja el resultado en A
```

c. **Direccionamiento inmediato [#nn]**

Las instrucciones que utilizan este modo de direccionamiento tienen una longitud de 2 bytes. El primero, como en todas las instrucciones, indica el código de operación y el segundo es el operando. Precediendo al operando se utiliza el símbolo #.

Un ejemplo de este tipo de direccionamiento puede ser:

```
LDA #2FH       Cargar el Acumulador con 00101111B
```

d. **Direccionamiento absoluto [ABS]**

Las instrucciones que utilizan este tipo de direccionamiento están compuestas de tres bytes, uno de código de operación y dos de operando. Los dos bytes que siguen al código de operación especifican la dirección absoluta de memoria en la que se encuentra el dato a procesar. Esta dirección está almacenada en memoria en orden inverso, de manera que, primero se encuentra el byte bajo y despues el byte alto.

Un ejemplo de este tipo de direccionamiento es la instrucción:

```
LDX 1234H      Carga el registro X con el dato que se
               encuentra en la dirección 1234H.
```

e. **Direccionamiento de página cero [ZP]**

Es un caso particular de direccionamiento absoluto que utiliza 2 bytes, uno para el código de operación y otro para indicar el operando. El operando indica la dirección de la página 0 a la que deseamos acceder. Al ser ejecutada por el microprocesador, el byte bajo de la dirección del dato se toma del segundo byte de la instrucción, y el byte alto del bus de direcciones se coloca automáticamente a 0. Por tanto este método de direccionamiento proporciona un mayor aprovechamiento de la memoria y una velocidad de ejecución

mayor que el direccionamiento absoluto.

Todas las instrucciones válidas en direccionamiento absoluto lo son también en direccionamiento de página cero, con la excepción de JMP y JSR. Es un buen método de programación reservar la página cero (del byte 0 al byte FFH de la memoria) para la declaración de variables.

Un ejemplo de este tipo de direccionamiento es la instrucción:

STA 20H	Almacena el contenido del Acumulador en la dirección 0020H.

f. **Direccionamiento absoluto indexado [ABS,X ó Y]**

Es una variante del direccionamiento absoluto comentado anteriormente. El indexado puede efectuarse indistintamente mediante el registro X o el Y. El valor efectivo de la dirección de memoria con la que se efectúe la transferencia se obtiene sumando la dirección absoluta, indicada en los dos bytes siguientes al código de operación, con el valor que en ese momento tenga el registro índice usado. Dado que tanto el registro X como el Y son de 1 byte, mediante este direccionamiento se puede acceder a cualquier dirección de un bloque de memoria de 256 bytes. De este modo se facilitan ciertos tipos de tratamientos de tablas.

Un ejemplo de este tipo de direccionamiento es la instrucción:

LDA 1000H,X	Cuando X contiene 25H, transfiere al Acumulador el dato contenido en la dirección 1025H.

g. **Direccionamiento de página cero indexado [ZP,X ó Y]**

Las instrucciones que utilizan este tipo de direccionamiento incluyen un solo byte de operando que constituye la dirección base de página cero a la que se sumará el registro de índice. Según el valor de este índice se puede acceder a cualquier dirección de página cero.

Un ejemplo de este tipo de direccionamiento es la instrucción:

LDA 10H,X	Cuando X contiene 25H, transfiere al Acumulador el dato contenido en la dirección de página cero 35H.

h. **Direccionamiento indirecto post-indexado [(IND),Y]**

Las instrucciones que utilizan este tipo de direccionamiento constan de dos bytes. El primero, como siempre, indica el código de operación y el segundo representa una dirección de página 0. El contenido de esta dirección de memoria y el de la siguiente forman a su vez una nueva dirección de memoria que al ser sumada con el contenido del registro índice Y, proporciona la dirección efectiva del dato.

Este modo de direccionamiento, con la ayuda de las instrucciones INY o DEY facilita la exploración de tablas situadas en cualquier zona del mapa de memoria.

Un ejemplo de este tipo de direccionamiento es la instrucción:

LDA (12H),Y	Cuando las direcciones de página 0 12H y 13H contengan respectivamente 00H y 10H y el valor del registro Y sea 5, trás la ejecución de esta instrucción se colocará en el Acumulador el contenido de la dirección 1005H.

i. **Direccionamiento indirecto preindexado [(IND,X)]**

Las instrucciones que utilizan este tipo de direccionamiento contienen como operando una dirección de página cero, que sumada con el contenido del registro X apunta a otra dirección de página cero. Si la suma de ambos términos excede de FFH, se ignora el acarreo, no pasando a la página 01 sino produciéndose un truncamiento. En esta dirección y en la siguiente se encuentra, en orden inverso (primero byte bajo y luego byte alto) la dirección efectiva donde está el dato.

Este modo de direccionamiento permite escribir programas que, con ayuda de las instrucciones INX o DEX, obtengan datos situados en posiciones no contiguas en cualquier punto del mapa de memoria. Por contra exige que en la página cero se disponga de una tabla contigua con las direcciones de base de cada uno de estos datos.

Un ejemplo de este tipo de direccionamiento es la instrucción:

LDA (20H,X)	Si el registro X contiene 08H, en las posiciones 0028H y 0029H se ecuentra respectivamente 34H y 12H y en la direccción 1234H se encuentra el dato 55H; al ejecutar esta instrucción el acumulador

se cargará con el dato 55H.

j. **Direccionamiento relativo [REL]**

Este modo de direccionamiento es utilizado exclusivamente por las instrucciones de salto condicional (branch). La dirección de memoria a la que se transfiere el control del programa se obtiene sumando el valor del contador de programa al segundo byte de la instrucción, denominado "offset". Este offset es un número en complemento a 2, con lo que se pueden efectuar saltos de hasta 127 posiciones hacia adelante o 128 hacia atrás.

Un ejemplo de este tipo de direccionamiento es la instrucción:

BNE 20H	Si la instrucción está contenida en las posiciones 1000H y 1001H del programa, el valor del P.C. actualizado será 1002H. Si el flag Z del registro Status es distinto de 0 el valor del "offset" se sumará al del P.C. y la ejecución del programa continuará a partir de la dirección 1022H de memoria. En caso contrario la siguiente instrucción a ejecutar será la que ocupe la dirección 1002H.

k. **Direccionamiento indirecto absoluto [(ABS)]**

Estas instrucciones son de tres bytes que contienen el código de la instrucción y la dirección absoluta en la que se encuentra la parte alta de la dirección efectiva del dato. La parte baja de la dirección efectiva del dato se encuentra en la siguiente dirección.

Un ejemplo de este tipo de direccionamiento es la instrucción:

LDA (2000H)	Si la dirección 2000H contiene el dato 10H y la siguiente FFH, al ejecutar esta instrucción el Acumulador se cargará con el dato que contenga la dirección de memoria FF10H.

l. **Direccionamiento indirecto preindexado absoluto [(ABS,X)]**

Este modo de direccionamiento sólo afecta a la instrucción de salto incondicional JMP. El contenido de los bytes segundo y

tercero de la instrucción se suma al registro X. El resultado, de 16 bits, es la dirección de memoria donde se encuentra el byte bajo de la dirección efectiva del dato. En la siguiente dirección se encuentra el byte alto de la dirección efectiva del dato.

Un ejemplo de este tipo de direccionamiento es la instrucción:

JMP (1234H,X) Si X contiene 02H y en las direcciones 1236H y 1237H se encuentran los datos 00H y 30H respectivamente, despues de ejecutar esta instrucción el P.C. pasará a ejecutar la que este en la dirección 3000H.

3.2.4 Registro Status del 65C02

Antes de entrar en el análisis de las instrucciones del 65C02, conviene estudiar con detalle el registro de Status, ya que gran número de estas instrucciones modifican o consultan el estado de los bits de este registro (flags).

Estos flags son los siguientes:

a. **Bit de Acarreo o 'Carry' (C):** La misión de este Flag es la de avisar de que, como consecuencia de una operación aritmética, el resultado desborda la capacidad de un registro de 8 bits. Este bit puede ser modificado directamente mediante las instrucciones SEC, que lo pone a "1" y CLC que lo pone a "0".

b. **Bit de Cero (Z):** Cuando Z=1 indica que el resultado de una operación es cero. No se puede modificar directamente el estado de este bit mediante una instrucción, sin embargo hay algunas que lo consultan (p.e. BEQ, saltar si Z=1).

c. **Bit de Interrupción (I):** Las interrupciones o peticiones de servicio de la CPU que llegan a través de la línea /IRQ serán atendidas solamente si I=0. Este Flag puede ser modificado por dos instrucciones: SEI para ponerlo a '1' y CLI para ponerlo a '0'.

d. **Bit de Modo Decimal (D):** Cuando este bit está a "1" la ALU realiza las operaciones aritméticas como si los operandos estuvieran codificados en BCD. La instrucción SED pone a "1" este Flag y la instrucción CLD lo pone a "0".

e. **Bit de Break (B):** La instrucción BRK (Break) da lugar a una interrupción por programa, saltando a la posición indicada por el vector que ocupa las posiciones FFFEH y FFFFH, de la misma manera que si se hubiese recibido una interrupción por la línea /IRQ. La única forma de distinguir el origen de la interrupción es consultando este Flag, que será '1' si la interrupción procede de una instrucción BRK y si procede de una interrupción externa.

f. **Bit de 'overflow' o desbordamiento (V):** Si, como consecuencia de una operación aritmética, el número resultante excede al máximo que puede representarse con 8 bits en complemento a 2 y, por tanto, se ha alterado el bit de signo, es necesario advertir que se ha producido desbordamiento. En este caso se pone a "1" el bit V indicando que el resultado es erróneo. Si se está trabajando con números sin signo, el estado de este Flag no es significativo. Este bit puede ponerse a "0" directamente mediante la instrucción CLV.

   La CPU tiene un terminal denominado S.O. ("Set Overflow") que permite la activación directa de este Flag para dar aviso de un cierto hecho exterior.

g. **Bit de signo (N):** Contiene la cópia del bit 7 resultante de todas las operaciones aritmeticas y lógicas. Trabajando con números en complemento a dos, el bit 7 representa el signo, siendo 0 si es signo '+' y 1 si es signo '-'.

3.2.5 Juego de instrucciones

En este capítulo se expondrán los distintos grupos en que se pueden dividir las instrucciones del microprocesador 65C02, sin embargo para una explicación detallada de la función de cada una de las instrucciones consultar el Apéndice A.

Instrucciones de Movimiento de Datos: Aunque casi todas las instrucciones implican alguna forma de movimiento de datos, nos referimos aquí a aquellas cuya función básica es la transferencia de datos entre registros y/o memoria.

a. transferencias entre registros:

- TAX: Acumulador al reg. X
- TXA: reg. X al Ac.
- TAY: Ac al reg. Y
- TYA: reg. Y al Ac.
- TSX: Puntero de Stack (SP) al reg. X

- TXS: reg. X al SP

b. transferencias entre registros y Stack:
   - PHA: Ac. al Stack
   - PLA: Stack al Ac.
   - PHP: reg. de Estado (P) al Stack
   - PLP: Stack al reg. P
   - PHX: reg. X al Stack
   - PLX: Stack al reg. X
   - PHY: reg. Y al Stack
   - PLY: Stack al reg. Y

c. transferencias entre registros y memoria:

   - LDA: Memoria al Ac.
   - STA: Ac. a Mem.
   - LDX: Mem. al reg. X
   - STX: reg. X a Mem.
   - LDY: Mem. al reg. Y
   - STY: reg. Y a Mem.
   - STZ: '0' a Mem.

Los dos primeros grupos de instrucciones son de direccionamiento implícito y, por tanto, de un solo byte. El grupo de transferencia con memoria admite distintos tipos de direccionamiento, tal como puede apreciarse en el Apéndice A.

Instrucciones Aritmético-lógicas: Son las que efectúan este tipo de operaciones con el contenido de los registros o de memoria.

a. operaciones aritméticas: se reducen a la suma y resta de operandos, y al incremento o decremento de registros o memoria.

   - ADC: Ac.+Mem.+carry
   - INC: Mem.+1
   - INX: reg.X+1
   - INY: reg.Y+1
   - SBC: Ac.-Mem.-(/carry)
   - DEC: Mem.-1
   - DEX: reg.X-1
   - DEY: reg.Y-1

b. operaciones lógicas: se reducen a tres funciones básicas que se efectúan entre un operando y el acumulador. Estas funciones operan bit a bit.

- AND: Ac. [AND] Mem.
- ORA: Ac. [OR] Mem.
- EOR: Ac. [OR-exclusivo] Mem.

Otras funciones lógicas se pueden obtener por combinación de las anteriores. En particular, es interesante considerar la inversión, bit a bit, del contenido del acumulador. Esta función puede obtenerse de forma sencilla con la instrucción:

```
EOR #0FFH
```

Las operaciones de "suma" y "resta" están preparadas para facilitar su encadenamiento en los cálculos con datos mayores de 1 byte (doble precisión o mayor). En estos casos, el bit de carry del registro de estado almacena el acarreo entre cada operación parcial. Así para sumar dos datos de N bytes, sólo es preciso iniciar el bit de carry a '0' y proceder de forma cíclica con cada uno de los bytes que componen el dato. Un ejemplo de este procedimiento puede ser:

```
CLC              ;BORRADO INICIAL DEL CARRY
LDA     BYTEA_1  ;CARGA DEL PRIMER BYTE DEL DATO A
ADC     BYTEB_1  ;SUMA PRIMER BYTE A+B
STA     SUMA_1
LDA     BYTEA_2  ;IDEM SEGUNDO BYTE
ADC     BYTEB_2
STA     SUMA_2
...
...
LDA     BYTEA_N  ;IDEM ULTIMO BYTE
ADC     BYTEB_N
STA     SUMA_N   ;FINAL DE A+B
```

En el caso de la resta múltiple, conviene recordar que el valor inicial del bit de carry ha de ser '1' (SEC).

Otro aspecto importante, es el comportamiento de los flags en estos dos tipos de operaciones. Dado que el microprocesador desconoce si los datos con los que opera se consideran con signo o no, duplica la información generada después de cada operación para cubrir las dos posibilidades. Así, el bit de carry indica cuando se produce acarreo suponiendo los datos sin signo, y el bit de overflow realiza la misma función pero considerando que los datos con que se opera están codificados en complemento a 2. El programador debe utilizar en cada caso la información adecuada al tipo de datos utilizados.

Las instrucciones de incremento y decremento, no modifican el contenido de los dos flags que indican acarreo (carry y overflow).

Instrucciones de Comparación: efectúan una comparación entre uno de los registros y la memoria. El resultado se presenta como una modificación de los flags N, Z y C que permite deducir si entre los valores numéricos comparados existe una relación de "MAYOR", "MENOR" o "IGUAL".

Para efectuar esta comparación, la CPU realiza internamente una resta entre el registro y la memoria, sin que el resultado modifique ninguno de los operandos. Debe tenerse en cuenta que los flags se modifican de forma equivalente a si se efectuara la diferencia con la instrucción 'SBC', a excepción del flag V (overflow) que no se altera, y sin tener en cuenta el flag de carry. Por tanto, no es posible conocer si se produce desbordamiento cuando se comparan valores con signo.

Hay tres instrucciones básicas de comparación, cada una de ellas con distintas formas de direccionamiento:

- CMP: compara el Ac. con la Mem.
- CPX: compara el reg.X con la Mem.
- CPY: compara el reg.Y con la Mem.

Analizando el valor de cada uno de los flags afectados por esta instrucción, podemos determinar la relación entre los operandos, registros (R) y memoria (M), según la tabla siguiente:

```
C = 0  -->  R <  M     (menor)
C = 1  -->  R >= M     (mayor o igual)
Z = 0  -->  R <> M     (distinto)
Z = 1  -->  R =  M     (igual)
```

El flag de signo (N) toma el valor del bit 7 del resultado de la operación diferencia que se realiza para la comparación. Las instrucciones de comparación se utilizan, en combinación con las de salto condicional, para crear puntos de toma de decisiones en un programa. Con ello se consigue bifurcar la ejecución de un programa en función de una condición variable.

Instrucciones de Desplazamiento y Rotación: Son un conjunto de 8 instrucciones que afectan al acumulador o a la memoria

indicada según el modo de direccionamiento, desplazando su contenido a derecha o izquierda. En estos desplazamientos interviene el flag de carry, que actúa como un noveno bit del elemento que se desplaza. En las instrucciones de rotación se cierra el camino de desplazamiento formando un bucle. Por tanto, la información contenida antes de ejecutar la instrucción de rotación, no se pierde.

En las instrucciones de desplazamiento, se pierde un bit de información que se sustituye por un '0' (ver apéndice A).

El nemónico de estas instrucciones varia cuando se refieren al acumulador, según lo siguiente:

- ASL: desplazar Mem. a la izquierda
- LSR: desplazar Mem. a la derecha
- ROL: rotar Mem. a la izquierda
- ROR: rotar Mem. a la derecha
- ASL A: desplazar Ac. a la izquierda
- LSR A: desplazar Ac. a la derecha
- ROL A: rotar Ac. a la izquierda
- ROR A: rotar Ac. a la derecha

Una aplicación importante de este tipo de instrucciones es la conversión serie-paralelo y paralelo-serie de datos. En el Microinstructor se utiliza este procedimiento para grabar y leer datos de cassette.

Otra aplicación interesante es la multiplicación y división por potencias de 2. En este caso particular, es suficiente desplazar el dato N veces a izquierda o derecha, siendo N el exponente de 2. Así, puede realizarse un programa que multiplique el contenido de una posición de memoria (p.e.) por 8 (N=3), utilizando el método siguiente:

```
LDA     DATO    ;CARGA DEL DATO DE PARTIDA
STA     PROD1   ;NO SE DESTRUYE EL DATO DE PARTIDA
ASL     PROD1   ;...x2
ROL     PROD2   ;DESPLAZAMIENTO AL SEGUNDO BYTE
ASL     PROD1   ;...x4
ROL     PROD2
ASL     PROD1   ;...x8
ROL     PROD2
```

DATO es la posición de memoria que contiene el valor que queremos multiplicar por 8. El resultado de este producto será una cifra de 16 bits, por lo que se requieren 2 bytes para guardarlo:

[PROD2][PROD1] = [DATO] x 8

Un programa equivalente puede realizarse para dividir, cambiando el sentido de desplazamiento.

Instrucciones de Salto: Son todas aquellas que modifican la ejecución lineal de un programa, alterando el contenido del Contador de Programa (PC). Tal como ya se ha expuesto, al término de la ejecución de una instrucción, el registro PC se incrementa para direccionar la instrucción siguiente en memoria. Pero, cuando se ejecuta una instrucción de salto, el nuevo valor del PC se obtiene en función de la instrucción ejecutada. Esta alteración del PC puede ser incondicional o condicionada al estado de alguno de los flags del registro de Status.

a. salto incondicional: son aquellas instrucciones cuyo código incluye, en sus distintas formas de direccionamiento, el nuevo valor del Contador de Programa:

   - BRA: salto relativo incondicional
   - JMP: salto absoluto incondicional
   - JSR: salto a subrutina
   - BRK: salto a subrutina de Break
   - RTS: retorno de subrutina
   - RTI: retorno de rutina de interrupciÃn

   La instrucción 'BRK' toma la dirección de salto de un vector almacenado en una posición fija de memoria (FFFEH-FFFFH) e, igual que la instrucción de salto a subrutina 'JSR', almacena en el stack la dirección de retorno al punto de partida, para que sea utilizada por 'RTS' y 'RTI'.

b. salto condicional: la modificación del PC depende del valor de alguno de los bits del registro de Status. Estas instrucciones siempre producen saltos relativos a la dirección de inicio de la siguiente instrucción del programa. Estos saltos relativos tienen el valor contenido en el segundo byte de la instrucción, expresado en complemento a 2:

   - BCC: salto si C = 0
   - BCS: salto si C = 1
   - BEQ: salto si Z = 0
   - BNE: salto si Z = 1
   - BPL: salto si N = 0
   - BMI: salto si N = 1
   - BVC: salto si V = 0
   - BVS: salto si V = 1

- BBRn: salto si el bit n de Mem. es '0'
- BBSn: salto si el bit n de Mem. es '1'

Dado que estas son las únicas instrucciones que permiten evaluar estados en un programa, cualquier situación que requiera una toma de decisión debe reducirse a la modificación de uno de los cuatro flags anteriores o de uno de los bits de una posición de memoria en página 0.

Instrucciones de Modificación de Flags: Permiten fijar el estado de alguno de los flags del registro de Status. Aunque la manipulación directa de todos los flags no es posible, se pueden llevar a un estado conocido por medio de operaciones indirectas.

- CLC: C = 0
- SEC: C = 1
- CLD: D = 0 (pasa a modo binario)
- SED: D = 1 (pasa a modo decimal)
- CLI: I = 0 (permite interrupcion /IRQ)
- SEI: I = 1 (inhibe interrupción /IRQ)
- CLV: V = 0

Cualquier bit del registro de Status puede modificarse indirectamente a través del Stack, por ejemplo hacer V=1 y Z=1:

```
PHP                         ;FLAGS AL STACK
PLA                         ;FLAGS AL ACUMULADOR
ORA     #01000010B          ;V=1 Y Z=1
PHA                         ;AC. AL STACK
PLP             ;AC. AL REG. DE STATUS
```

Este método permite inicializar simultaneamente el valor de varios flags, aunque en la mayoría de aplicaciones, la utilización de las instrucciones específicas es suficiente y más rápida.

Instrucciones de Manipulación y Test de Bits: Son instrucciones que operan directamente sobre los bits de la posición de memoria seleccionada. Permiten comprobar el estado de cualquiera de los bits y/o forzarlo a '0' o a '1'.

- BIT: test de los bits de Mem. con el Ac.
- RMBn: pone a '0' el byte 'n' de la Mem.
- SMBn: pone a '1' el byte 'n' de la Mem.
- TRB: test de Mem. con Ac. y puesta a '0'

- TSB: test de Mem. con Ac. y puesta a '1'

La operación de test significa que se realiza internamente un AND del acumulador con la posición de memoria seleccionada. Si su resultado es '0', se activa el flag Z para su posterior comprobación. En las instrucciones TRB y TSB, sólo se modifican aquellos bits de memoria que corresponden con '1' en el acumulador. Así, si queremos poner a '0' todos los bits pares de una posición de memoria genérica MEM1, podemos seguir el procedimiento siguiente:

```
LDA     #10101010B      ;MASCARA DE SELECCION
TRB     MEM1
```

Analizando posteriormente el flag Z, podemos deducir si alguno de estos bits estaba previamente a '1' o no.

Instrucción Nula: Esta instrucción no modifica el estado de la CPU ni realiza operación alguna. Pero es de gran utilidad cuando se pretende ajustar la duración de un determinado fragmento de programa a un valor conocido. También puede utilizarse durante el desarrollo de un programa en RAM, para reservar espacio entre instrucciones, de forma que pueda posteriormente insertarse alguna instrucción sin tener que introducir de nuevo una parte del programa.

- NOP: no operación

3.3 ELEMENTOS ASOCIADOS A LA CPU EN EL MICROINSTRUCTOR

En este capítulo se analízan las funciones de los principales elementos asociados a la CPU en el Microinstructor.

a. **Amplificadores de Corriente (Buffers):**

Tanto las líneas del Bus de Datos como las del Bus de Direcciones están disponibles en el Conector de Expansión del Microinstructor para permitir el control de elementos externos que utilicen dichas líneas.

Como puede verse en el esquema del circuito, para proporcionar la corriente necesaria para controlar correctamente a todos los circuitos conectados a los buses de datos y direcciones, se intercalan unos amplificadores de corriente que, a la vez, proporcionan una cierta protección al Microprocesador. Estos buffers están formados por los circuitos IC16, IC3 e IC8.

b. **Control de lectura/escritura:** Dadas las características técnicas de esta CPU, la línea de escritura R/W procedente de la CPU no puede ser utilizada directamente para controlar el proceso de lectura/escritura de las memorias RAM. Esto se debe a que la señal R/W permanece a '0' (escritura) durante un tiempo mayor al que el valor que se pretende escribir permanece en el Bus de Datos. En lugar de esta señal se emplea /RWR, generada a partir de Ø2 y de R/W mediante el circuito que se muestra en la figura 3.9 de tal manera que /RWR sólo es válida (nivel '0') durante el periodo de tiempo en que Ø2=1 y R/W=0. Esta señal se emplea para la RAM opcional (conector J9) y se debe usar para el control de la memoria externa si se amplía el equipo.

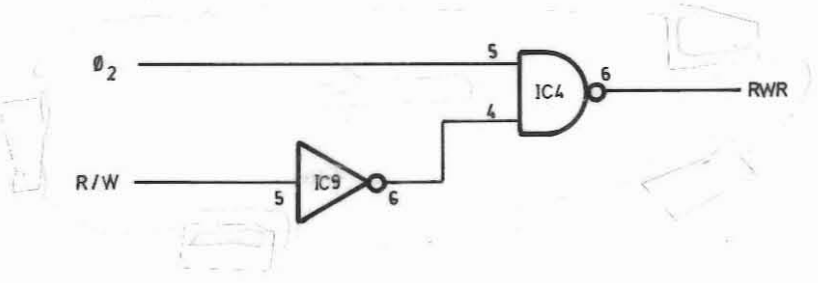


Figura 3.9. Generación de la señal de control de lectura/escritura

La selección de lectura/escritura en la RAM básica (IC15) se realiza mediante la señal /RAM/WRP, generada a partir de la señal /RWR y que constituye un circuito de protección contra escritura controlado dinamicamente por el usuario (ver figura 3.10).

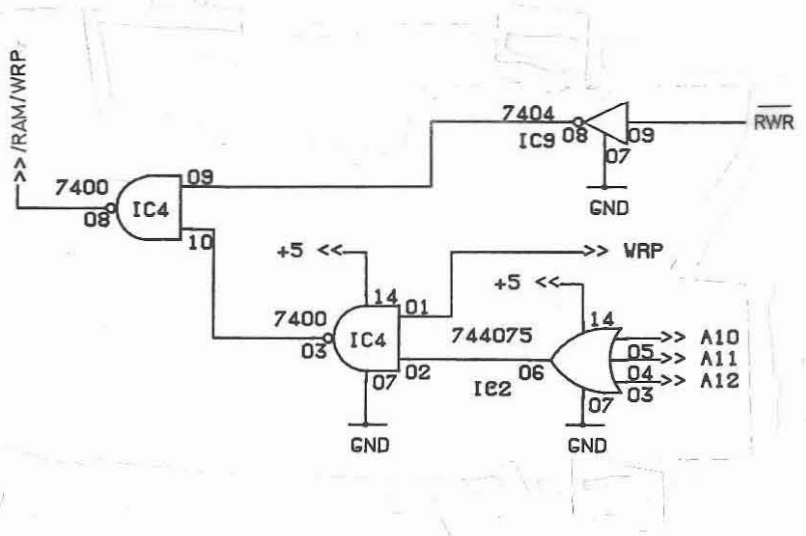


Figura 3.10. Generación de la señal de protección de escritura

En condiciones normales de trabajo la señal WRP, que es la línea PB4 del 65C22 (IC19), está a nivel bajo. Esta señal aplicada a la entrada de una puerta NAND (IC4) provoca un nivel alto en su salida. De este modo la señal /RAM/WRP coincidirá en todo momento con /RWR, ya que la puerta NAND de IC4 (salida pin 8) se comportará como un inversor.

Cuando la protección de escritura esté activada, la línea PB4 de IC19 (WRP) pasa a '1'. Esto se consigue mediante la programación adecuada de las líneas de este circuito integrado. Cuando se está direccionando la RAM en la zona 0000H a 03FFH, las líneas A10, A11 y A12 están a nivel bajo, con lo cual la salida (pin 6) de la puerta OR de IC2 también permanecerá en este nivel. De este modo la salida de la primera puerta NAND estará, igual que en el caso anterior, a nivel alto y por tanto la señal /RWR seguirá siendo igual a /RAM/WRP. Por el contrario, cuando se accede a una dirección que está en la zona 0400H-1FFFH, la salida de la puerta OR estará a nivel alto, lo que da lugar a que la salida de la primera puerta NAND pase a '0' y por tanto, la señal /RAM/WRP permanezca en nivel alto sea cual sea el nivel de la señal /RWR, con lo cual se inpide la escritura en la memoria RAM.

De esta manera, controlando por programa el nivel lógico de la línea WRP, se seleccionará o no esta protección.

c. **Circito de Reset:** Debido a que inicialmente C16 está descargado, cuando se aplica tensión al equipo la tensión en el pin 1 de IC12 es 0v, manteniendose a nivel lógico '0' la entrada de /RES del microprocesador. Debido a la carga de C16 a través de R12, el nivel de la tensión en la entrada del inversor llega a un nivel suficiente para cambiar de estado y a su vez bascula el estado del siguiente y la línea /RES pasa a '1'.

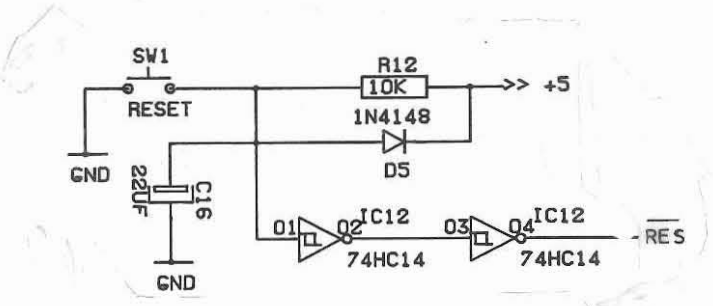


Figura 3.11. Circuito de Reset

De este modo se inicializa el funcionamiento del equipo con un salto a la dirección prevista en el vector de reset, en el instante de la puesta en marcha. D5 constituye un camino rápido de descarga para C16 al desaparecer la tensión de alimentación. El pulsador "RESET" también produce una rápida descarga de C16 que permite efectuar un reinicio manual, cuando sea necesario.

d. **Interrupción No Enmascarable (/NMI):** Los elementos asociados a esta línea cumplen una doble misión. Por una parte, se emplea para la ejecución paso a paso del programa y, por otra, para devolver el control al programa Monitor al pulsar la tecla STOP/BRK. El flanco generado durante la pulsación de dicha tecla, actúa como señal de reloj de un biestable tipo D (IC10), después de pasar por una red antirebote (R9-C15) y un conformador de señal (IC12). La salida /Q de este biestable pasa a nivel '0' y, a través de la puerta NAND de IC11 y la etapa inversora formada por T10, da lugar a que el nivel en la entrada /NMI de la CPU pase a '0', generando una secuencia de interrupción.

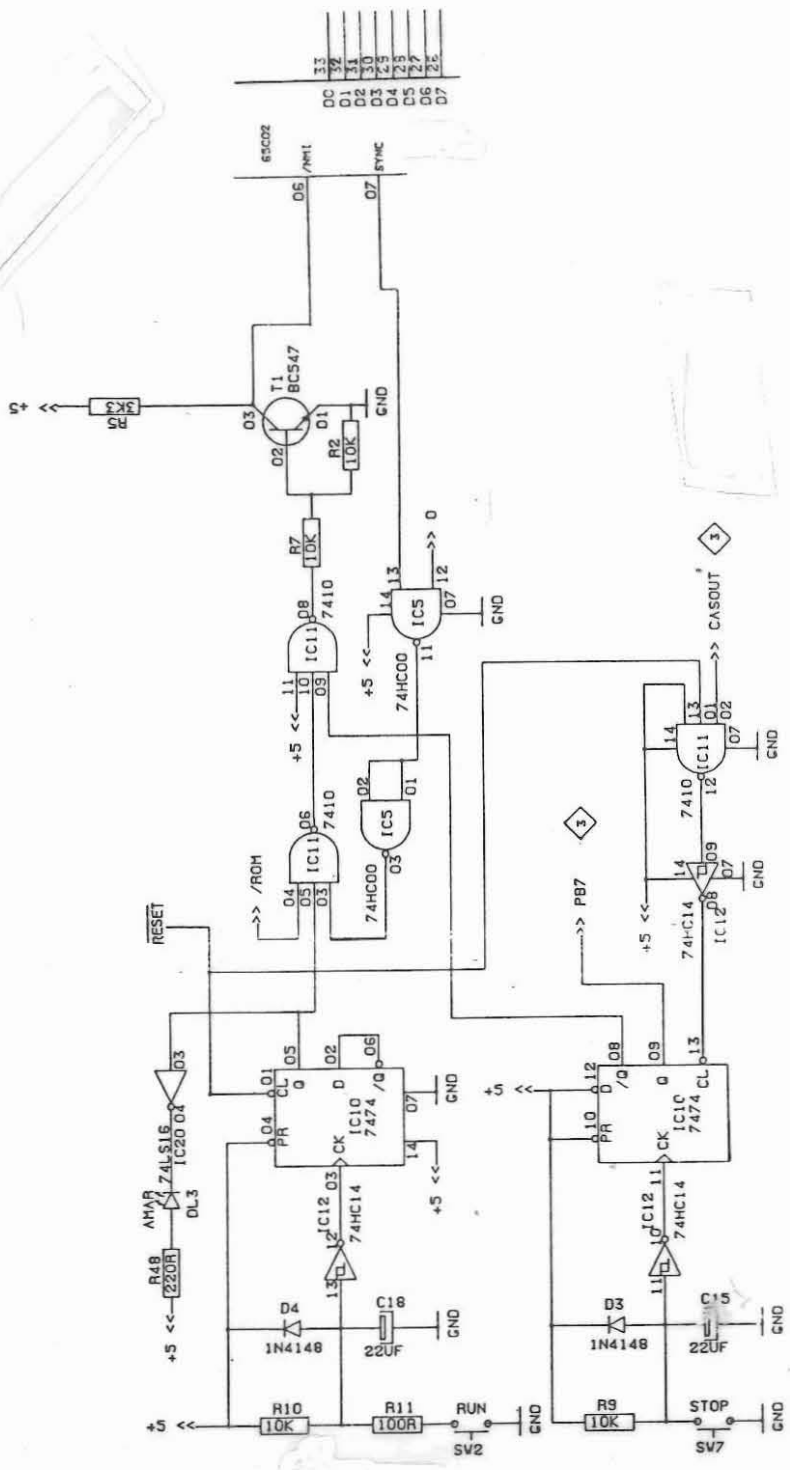


Figura 3.12. Circuito de control de /NMI

La salida Q del mismo biestable, que está conectada a PB7 del IC19 (65C22), pasa a '1', lo cual sirve como información de que la interrupción se ha producido por la pulsación de la tecla STOP/BRK. Dentro de la secuencia de NMI en el programa monitor, se analiza el nivel de esta línea al producirse una

interrupción y, si su estado es '1', pasa momentaneamente a '0' la línea CASOUT, que actúa sobre la entrada CL del biestable y restablece el estado de reposo (Q=0). Seguidamente el programa efectúa la secuencia de acciones necesarias para atender a la interrupción. La conexión del pin 13 de IC11 a la señal de /RES se emplea para inicializar adecuadamente este biestable, que de otra manera podría quedar inoperante, ya que la detección de /NMI se efectúa por el flanco de bajada de la señal y no por un nivel bajo, como sucede con /IRQ.

La ejecución paso a paso del programa se efectúa provocando una interrupción cada vez que la CPU ejecuta un ciclo de lectura de código de operación, indicado por el paso a '1' de la señal SYNC. Esta señal se debe sincronizar previamente con la señal 2 mediante una puerta NAND de IC5. Por otra parte, se debe anular este modo de funcionamiento cuando se esté ejecutando el programa entre las direcciones E000H y FFFFH.

Las entradas 3 y 4 de una puerta NAND de IC11, mezclan las señales anteriormente descritas con la que permite el funcionamiento paso a paso (pin 5), haciendo que su salida pase a cero únicamente cuando, estando permitido, se esté recogiendo el código de operación de una instrucción fuera de la memoria ROM que contiene el programa Monitor.

Esta información se mezcla con la procedente de STOP/BRK en las entradas 9 y 10 de una puerta NAND de IC11 y mediante un inversor se envía a la entrada /NMI de la CPU. La línea PB7 del 65C22 (IC19) no cambia de nivel, con lo cual el programa reconoce esta interrupción como ejecución paso a paso o solicitud externa mediante una instrucción de BREAK.

La pulsación de la tecla [RUN/STEP] pasa por un circuito idéntico al que existe entorno al pulsador STOP/BRK para evitar los rebotes y conformar la señal, la cual hace bascular al biestable. Al estar conectado este biestable como divisor por dos, su salida cambiará de estado a cada pulsación de la tecla. La salida Q de este biestable, además de controlar la permisión de la ejecución paso a paso mediante la entrada 5 de IC11, es enviada a la entrada 3 de un inversor de IC20 para controlar el encendido del LED "STEP", indicando de esta manera el estado de funcionamiento (paso a paso o normal). La señal de /RES, a través del pin 1 de este biestable efectúa, como en el caso anterior, una puesta a cero del biestable, pasando así a modo RUN.

3.4 ELEMENTOS DE ENTRADA-SALIDA DEL MICROINSTRUCTOR

3.4.1 Entradas/salidas de usuario: la VIA 65C02

La VIA 65C02 (IC22) constituye el elemento de entrada/salida de usuario. Está totalmente libre para prácticas y es accesible a través del Conector de Aplicación.

Este circuito consta de los siguientes elementos integrados:

- Dos ports de entrada/salida configurables.
- Dos temporizadores.
- Un registro de desplazamiento.

En la figura 3.13 puede verse el diagrama de bloques de este circuito.

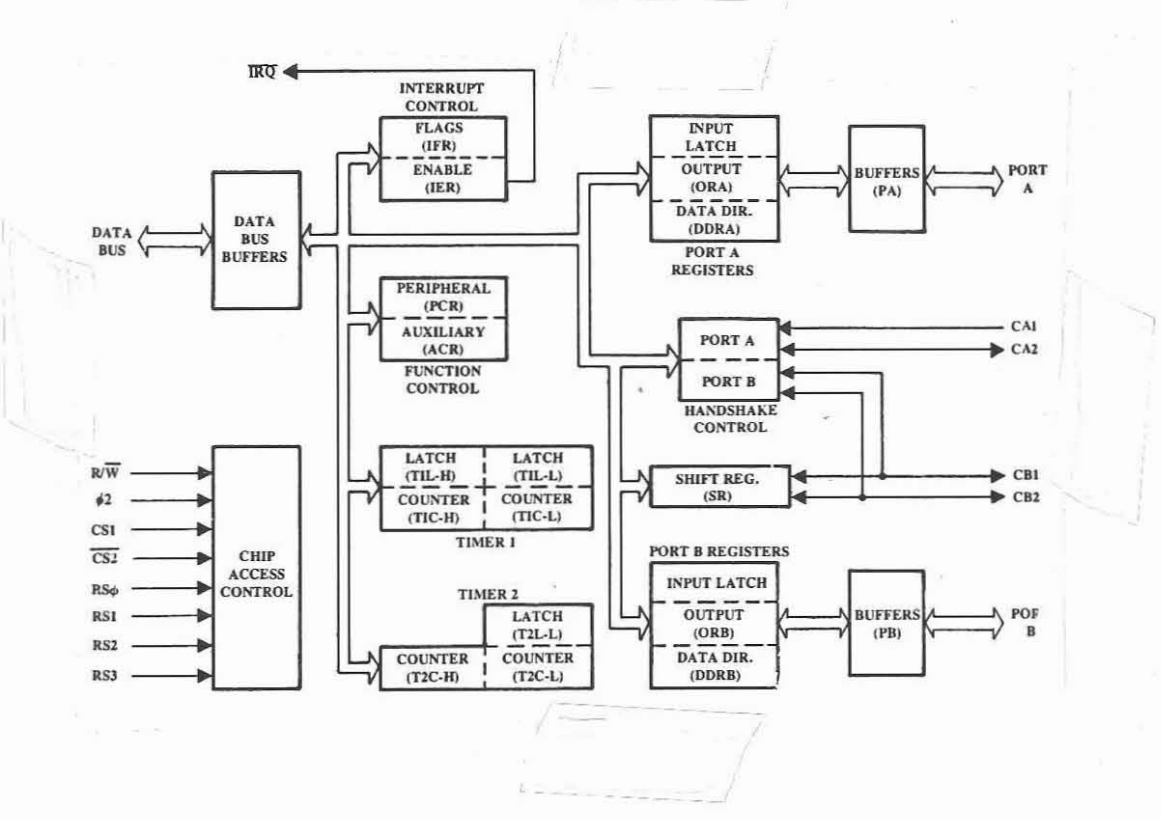


Figura 3.13. Diagrama de bloques de la VIA 65C22

La conexión de la VIA al sistema se realiza a través de los siguientes terminales:

- Bus de datos.
- R/W.
- Reloj Ø2.
- Terminales de selección (CS1 y /CS2).
- /IRQ.
- /RESET
- Entradas de selección de registros, que permiten seleccionar uno de los 16 registros internos que posee, de la forma indicada en la figura 3.14. Estos terminales se suelen conectar a las líneas de menos peso del bus de direcciones. De este modo, las direcciones de los registros internos de la VIA quedan comprendidas entre XXX0H y XXXFH. Los bits de mayor peso de esta dirección vienen definidos por el circuito decodificador formado por IC1-IC7. La VIA de entradas/salidas de usuario utilizada en el Microinstructor se selecciona con la señal /SEL0 (A400H-A7FFH).

Los Ports de la VIA

La VIA dispone de dos ports denominados port A y port B. Cada port consta de 8 terminales, configurables cada uno de ellos como entrada o como salida.

En el Microinstructor las líneas del port A están conectadas a 8 microinterruptores para simular entradas y las del port B a una fila de 8 diodos LED que pueden utilizarse para la simulación de entradas o la monitorización del estado de las líneas de este port. Todas estas líneas son accesibles a través del **Conector de Aplicación** del Microinstructor.

Asociados a cada port existen seis registros denominados ORA, ORB ("Output Register"); IRA, IRB ("Input Register") y DDRA, DDRB ("Data direction Register").

El contenido del registro DDRA (o DDRB) determina, bit a bit, el sentido de cada terminal. Un '1' en el bit 3 (p.e.) significa que la línea correspondiente del port A (port B) está activada como salida. Estos registros deben ser escritos como parte del proceso de configuración del periférico, antes de efectuar cualquier operación de lectura o escritura en los ports. Al efectuar un RESET todos los terminales de los ports quedan

REGISTRO ACCEDIDO	RS3	RS2	RS1	RSØ
PORT B	0	0	0	0
PORT A	0	0	0	1
CONFIGURACION PORT B	0	0	1	0
CONFIGURACION PORT A	0	0	1	1
TEMPORIZADOR 1	0	1	0	0
	0	1	0	1
	0	1	1	0
	0	1	1	1
TEMPORIZADOR 2	1	0	0	0
	1	0	0	1
REGISTRO DE DESPLAZAMIENTO	1	0	1	0
REGISTRO DE CONTROL AUXILIAR	1	0	1	1
REGISTRO DE CONTROL DE PERIFERICO	1	1	0	0
"FLAGS" DE INTERRUPCION	1	1	0	1
MASCARA DE INTERRUPCION	1	1	1	0
PORT A	1	1	1	1

Figura 3.14. Registros de la VIA 65C22

configurados como entradas.

Tanto al registro ORA (u ORB) como al IRA (o IRB), se le asigna una misma dirección, de modo que las operaciones de lectura leen el contenido de IRA y las de escritura escriben sobre ORA.

Al escribir un dato sobre alguno de los registros de salida, solamente son efectivos aquellos bits del port que se han programado como salidas, quedando el resto bloqueado en el correspondiente registro de salida. El dato que se escribe en ORA (u ORB)

Cuando se efectúa una operación de lectura en un port, se obtiene el dato contenido en IRA (o IRB). Este registro puede funcionar de dos modos:

- modo transparente: su contenido refleja en cada momento el estado lógico de las líneas del port.

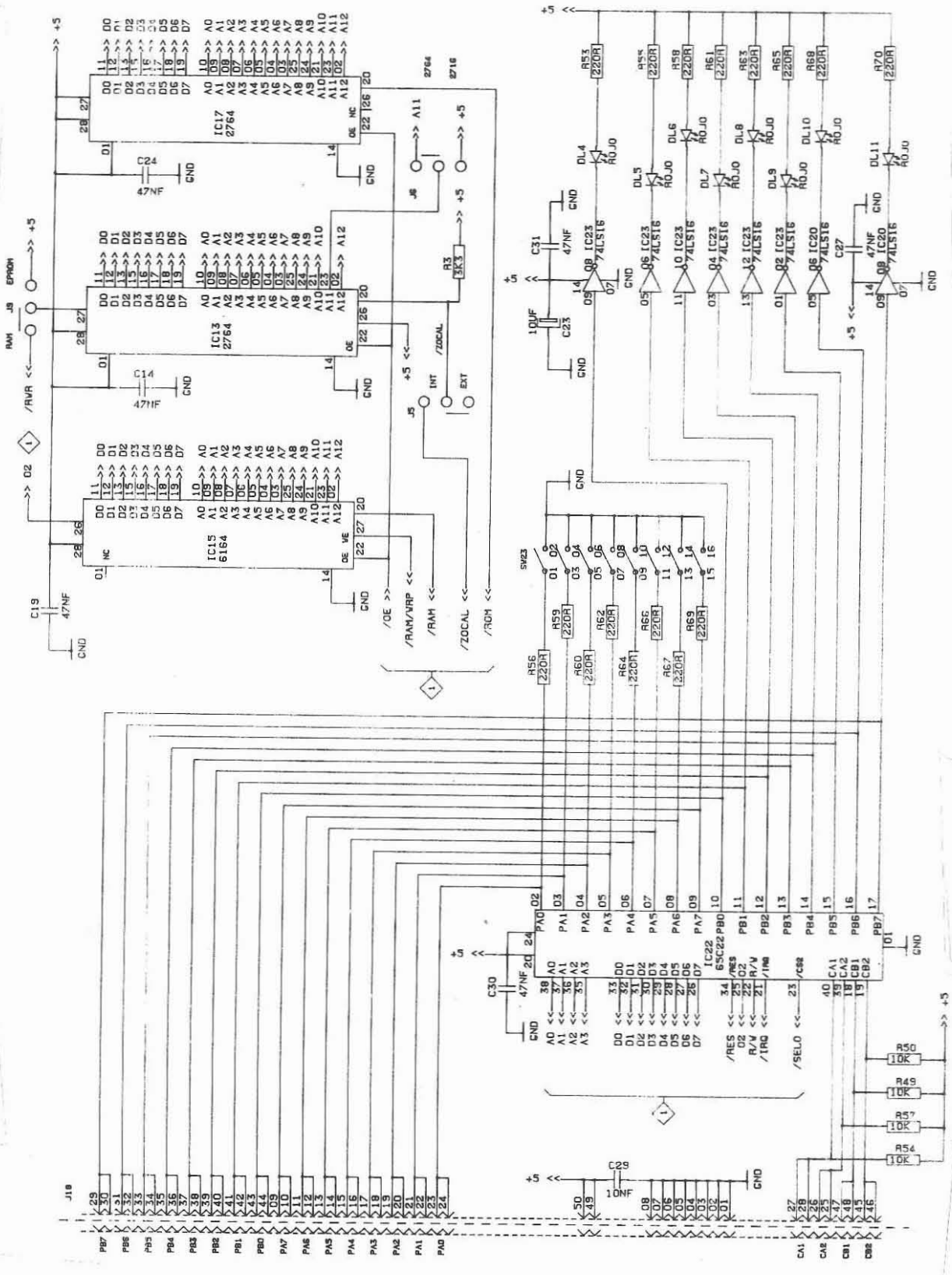


Figura 3.15. La VIA 65C22 de E/S de usuario.

- modo "latch": su contenido memoriza el estado de las líneas del port en el instante en que una señal auxiliar de control es activada.

En cualquier momento puede realizarse una lectura de los ports incluso si se han definido como salidas. En este caso, la lectura refleja el estado real de las líneas de salida del port.

Estos dos modos de funcionamiento de IRA e IRB quedan determinados por el contenido de los bits 0 y 1 del registro ACR ("Auxiliary Control Register"). En modo latch la "orden de captura" se recibe del exterior a través de los terminales auxiliares CA1 o CB1.

Terminales auxiliares

Estos terminales pueden tener distintas funciones:

CA1 y CB1:
- Entrada de interrupción.
- Entrada de orden de memorización, en modo latch.
- Entrada de control, para transferencia de datos.

CA2 y CB2:
- Entrada de interrupción.
- Salida de control, para transferencia de datos.
- Salida de impulsos.
- Salida programable.

La selección de cada uno de estos modos de trabajo puede hacerse programando el registro PCR ("Periferal Control Register") adecuadamente.

Para más información sobre el empleo de estas señales y de los temporizadores, registro de desplazamiento y señales de interrupción de que dispone este circuito integrado, se deben consultar las hojas técnicas proporcionadas por el fabricante.

3.4.2 Entradas/salidas locales: teclado y display

El control del teclado y del display se efectúa mediante una VIA 65C22 (IC19) y los circuitos a ella asociados.

Las líneas PB0 a PB3 de esta VIA se envían al decodificador 74HC42 (IC18), de tal manera que el código hexadecimal en ellas

existente da lugar a la activación de una de las 10 salidas posibles.

El **teclado** está constituído en forma de matriz de 3x8, parcialmente aprovechada puesto que sólo se utilizan 19 de los 24 nudos posibles. Las columnas están controladas por las tres salidas de menor peso de IC18, mientras que las líneas de matriz están gobernadas por las líneas PA0-PA7 de IC19.

Cuando no haya ninguna tecla pulsada todas las entradas PA estarán a '1', independientemente del nivel lógico de las columnas de la matriz.

Supongamos que se pulsa la tecla [C]. Cuando se efectúa la exploración del teclado, la CPU selecciona secuencialmente las columnas de la matriz del teclado poniendo a '0' consecutivamente las salidas de las patillas 1, 2 y 3 del decodificador decimal IC18. Mientras no se seleccione la columna correspondiente al pin 2 de este integrado detectaremos en el port B todos los bits a '1'. Al seleccionar en el decodificador la combinación de entrada 0010B, se selecciona la salida correspondiente al pin 2, pasando a '0'. Si en este momento leemos el dato presente a la entrada del port B, detectaremos 11101111B. El programa de control interpreta la posición del cero como "tecla [C] pulsada" y procede a la ejecución de la función que tiene asignada. A cada tecla corresponde una única configuración de las líneas PB0-PB3 y PA0-PA7.

El encendido del **Display** se hace de un modo muy similar a como se efectúa el análisis de teclas. Supongamos que se desea encender uno del los displays con una cierta configuración de segmentos. En primer lugar se debe colocar en las líneas PB0 a PB3 la dirección correspondiente al display que se pretende iluminar. Si, por ejemplo, se pone un 3 (0011B) en dichas líneas, se activará la patilla 4 de IC18 pasando a '0', lo cual pondrá en conducción al transistor T4 saturándolo y proporcionando la tensión necesaria para polarizar al ánodo correspondiente a los siete segmentos más el punto decimal, de uno de los dígitos. A continuación, en las líneas PA0-PA7 de IC19, definidas ahora como salidas, se coloca la configuración de segmentos que se desea encender. Estos niveles lógicos se transmiten por IC20 y IC21 y, a través de unas resistencias de 220 ohmios, para cerrar el circuito de polarización de los diodos LED de cada segmento que se quiere encender. De esta manera, sólo pasará corriente y se encenderá aquel segmento cuyo cátodo esté a '0', con la correspondiente línea del port A en estado alto.

Este método permite encender solamente un dígito al mismo tiempo, por lo que el proceso deberá repetirse para los 6 dígitos que componen el display de manera secuencial, seleccionándolos uno a uno y enviando el código de segmentos que corresponde al

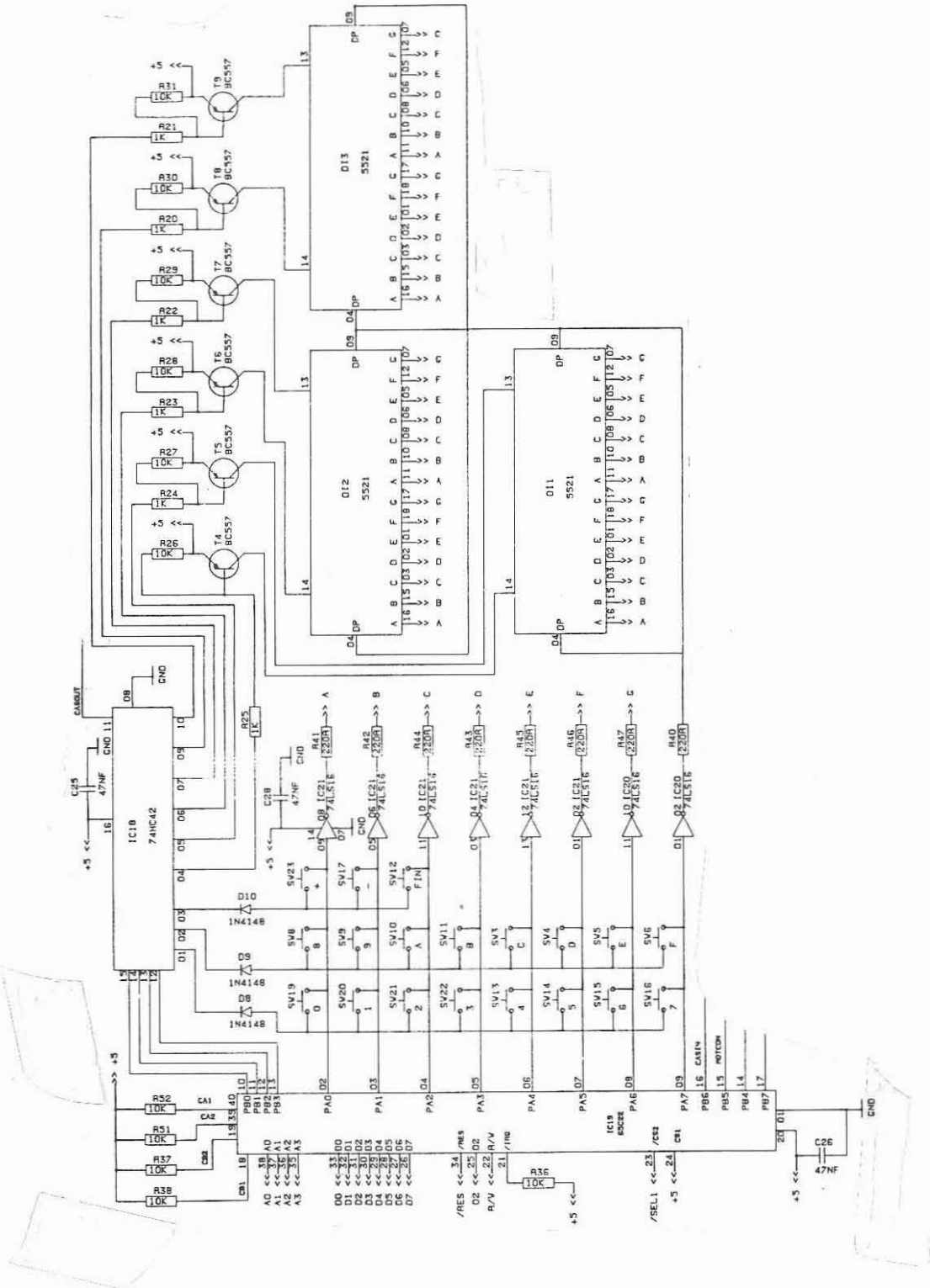


Figura 3.16. Teclado y display

seleccionado en cada momento. El software de control debe efectuar un refresco completo al menos 50 veces por segundo para obtener una imagen estable y sin parpadeos.

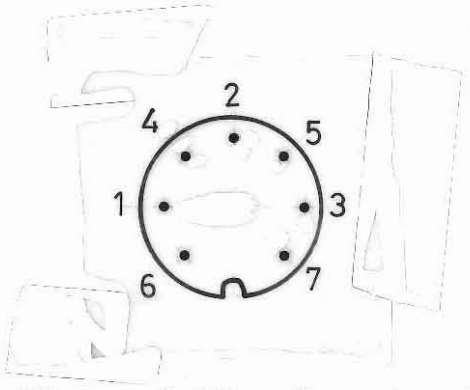
3.4.3 **Cassette**

Mediante este interface se puede conectar al Microinstructor un cassette de audio que permita la grabación para ser utilizado como medio para almacenar programas, datos, etc.

En la circuitería de este interface se genera la señal de audio que pasa a cinta mediante un programa, lo cual permite, por una parte, una gran flexibilidad en cuanto a velocidad de transferencia, formas de onda, etc y por otra, simplifica enormemente los circuitos requeridos para éste.

El nivel de salida de grabación procedente de CASOUT (patilla 11 de IC18) se reduce mediante R33, R34 y R35, proporcionando a su salida dos niveles de tensión, OUTH, adecuado para la entrada AUX del cassette, y OUTL para la entrada del micrófono (MIC). C22 contribuye a acondicionar la señal de entrada al cassette.

La entrada, que debe estar conectada a la salida de altavoz auxiliar del cassette, está constituida por un desacoplo de corriente continua (C21), una etapa atenuadora (R16 y R14), un limitador de tensión de entrada (Z1). El inversor de IC12 que le sigue, actúa como comparador, transformando la señal de entrada a los niveles lógicos adecuados (TTL). A la salida de este comparador (patilla 6 de IC12) se conecta, a través de un inversor, el LED (DL2) que da la indicación de lectura de información del cassette.



1.... M2
2.... GND
3.... OUTL
4.... M3
5.... OUTH
6.... M1
7.... IN

Figura 3.17. Conector del cassette

El control del motor se efectúa a través de MOTCON (PB5 de IC19). Esta configuración permite el control remoto de

prácticamente cualquier modelo de cassette existente. Incluye 3 salidas M1, M2 y M3. Dos de estas salidas se conectarán al control remoto del cassette. Para determinar la conexión a emplear, usar el siguiente procedimiento:

a. Insertar el jack de control remoto en el cassette.

b. Pulsar la tecla PLAY en el cassette. El motor deberá estar parado.

c. Medir la tensión en el terminal central de este jack empleando como referencia la masa del cable del altavoz. Anotar esta medida que denominaremos A.

d. Medir la tensión en la toma exterior del jack de control remoto empleando la misma referencia. Esta medida se denominará B.

e. Cortocircuitar los terminales central y externo del jack de control remoto. El cassette se deberá poner en marcha.

		LECTURA A		
		NEGATIVA	CERO	POSITIVA
LECTURA B	NEGATIVA		C-M3 E-G	
	CERO	C-G E-M3		C-M1 E-M2
	POSITIVA		C-M2 E-M1	

C: Conexión central del jack de control remoto.
E: Conexión exterior del jack de control remoto.
M1, M2, M3: Conexiones del Microinstructor.
G: Toma de masa del conector de cassette.

Figura 3.18. Tipos de conexiones

f. Empléese la tabla de la figura 3.18 para determinar el tipo de conexión a efectuar en el cassette.

NOTA: Para el correcto funcionamiento del cassette deberán estar conectados además del jack remoto, el de entrada o el de salida.

El formato de la información grabada en cinta es la siguiente:

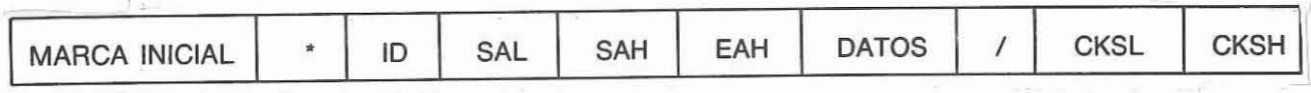


Figura 3.19. Formato de la información grabada en cinta

Marca inicial: Consiste en una secuencia de carácteres SYN (16H) que permite la sincronización en la lectura de datos.

*: Carácter ASCII (2AH) que sirve para marcar el inicio de datos.

ID: Número de identificación del programa.

SAL, SAH: Dirección inicial del bloque de datos introducida en "di".

EAL, EAH: Dirección final menos 1 del bloque de datos. Introducida en "df" sumándole 1.

DATOS: Bytes de datos.

/: Carácter ASCII (2FH) que sirve para marcar el final de datos.

CKSL, CKSH: Bytes de comprobación de datos en la cinta. Se obtienen sumando sin acarreo los distintos bytes almacenados en cinta para posterior comprobación o verificación (check-sum).

Los bytes de información se envian transmitiendo en primer lugar los bits menos significativos. El formato para cada bit es el que se muestra en la figura 3.20.

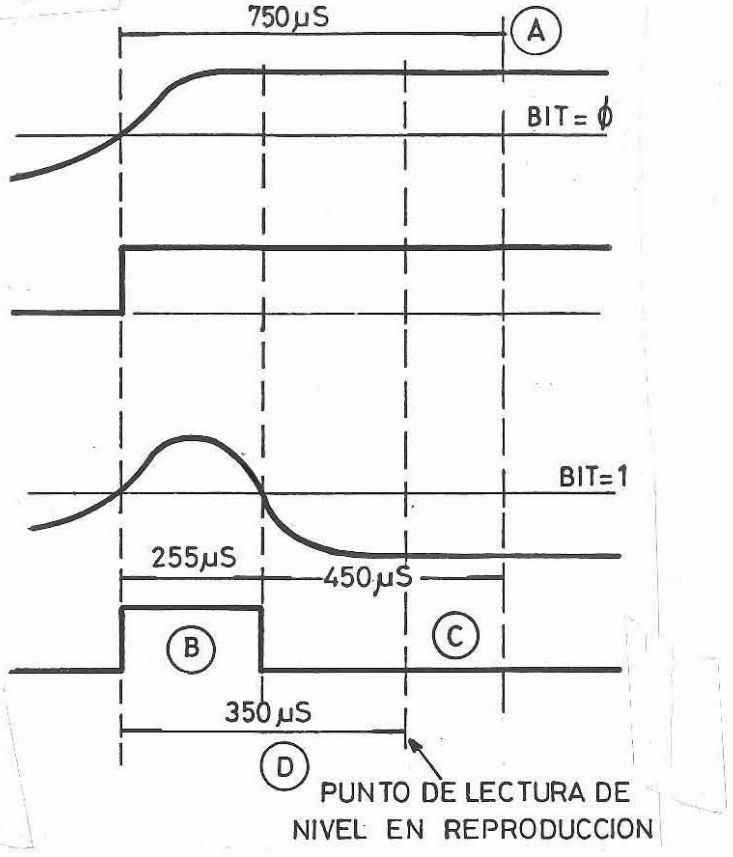


Figura 3.20. Formato de bit grabado en cinta

El tramo de marca inicial se utiliza para ajustar los controles de volumen y tono. Durante la lectura de esta zona la letra "S" que aparece en el display indica que se está en el periodo de sincronización. El programa está tratando de detectar un primer carácter de sincronismo (SYN). El indicador "S" se apaga al detectar este primer carácter y debe permanecer apagado hasta detectar el identificador (ID). Una imagen del valor binario del identificador aparece en el display hasta el final de la carga de datos. Los controles de volumen y tono estarán bién ajustados cuando se consiga apagar el indicador "S" durante el tramo de sincronismo. En caso de ser necesario se puede aumentar la longitud de esta marca alterando el contenido de la dirección 0370H (TAPDEL).

Algunas notas prácticas sobre la utilización del cassette como unidad de almacenamiento conectada al Microinstructor son las siguientes:

a. Los mejores rendimientos se obtienen para señales de entrada de un nivel de 2 voltios pico a pico, o bien a

volumen máximo (saturación).

b. Los controles de tono se colocan en una zona intermedia. Un ajuste de tonos ligeramente agudos suele dar mejores resultados en algunos cassettes.

c. Emplee una cinta de alta calidad.

d. Entre el cassette y el Microinstructor utilice un cable apantallado.

e. Mantenga límpios los cabezales del cassette.

f. Evite la grabación muy próxima al principio de la cinta.

g. Asegúrese de que la cinta está correctamente colocada en el cassette.

h. Asegúrese de que la conexión de los jacks es correcta.

i. Rebobine las cintas antes de sacarlas del cassette.

j. Evite la exposición de las cintas al polvo, humedad, calor, campos magnéticos, etc.

k. Antes de grabar en una cinta nueva rebobínela completamente en ambos sentidos para evitar posibles tensiones en la cinta.

l. Emplee siempre que sea posible un adaptador de red para cassettes. Si se emplean pilas, sustitúyalas al menor síntoma de mal funcionamiento, aunque aparentemente tenga carga suficiente.

3.5 ALIMENTACION

El transformador utilizado por el Microinstructor permite su funcionamiento a 110, 125, 220 o a 240 vóltios y 50 HHz y con fusibles de 250 mA para 220 y 240 vóltios, y de 500 mA para 110 y 125 vóltios.

Los diodos D1 y D2 constituyen un rectificador de doble onda cuya salida es filtrada por C1, C2 y C3 y estabilizada a 5 vóltios por IC6 para la alimentación general de todo el equipo. También está conectada al Conector de Expansión (J1) para posibles ampliaciones del usuario. La corriente máxima disponible es de 250 mA aproximadamente, un mayor consumo hace que el Microinstructor funcione defectuosamente.

PRACTICA 3.1 DECODIFICADOR DEL MICROINSTRUCTOR

Para la realización de esta práctica es necesario un analizador lógico AL-1610 de I.E. Prómax,SA u otro equivalente.

a. Apagar el Microinstructor.

b. Conectar las salidas de los decodificadores a los 7 primeros canales del Analizador Lógico según la siguiente relación:

Salidas decodificadores	Canales AL-1610
ROM	Canal 0
RAM	" 1
AUX	" 2
SEL1	" 3
SEL0	" 4
PERI0	" 5
PERI1	" 6

Programar la condición de disparo del Analizador Lógico cuando se cumpla que el canal 0 se hace "1".

Programar la frecuencia de muestreo a 100 nsg.

c. Encender el Microinstructor y disparar el Analizador Lógico de manera que quede a la espera de que se cumpla la condición de disparo para empezar la adquisición de datos.

d. Una vez hecha la adquisición el display mostrará un diagrama como el de la figura 3.21.

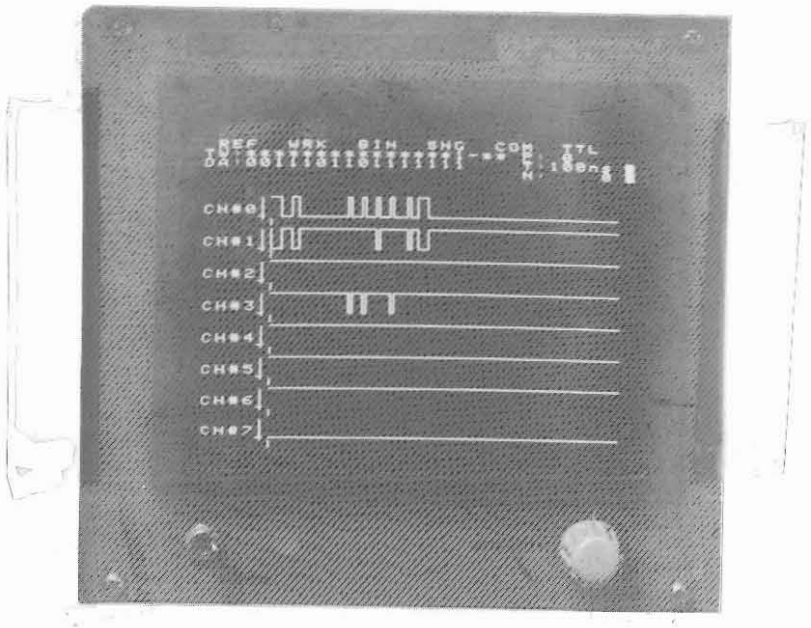


Figura 3.21. Señales de selección

Como se deduce de la figura anterior el programa que se está ejecutando (el bucle de reposo), está constantemente actuando con la memoria ROM, la memoria RAM y la VIA que controla el display, seleccionada mediante la señal /SEL1.

PRACTICA 3.2 DECODIFICACION DE LOS DIGITOS DEL DISPLAY

Para efectuar la salida de información por el display el microprocesador selecciona todos los dígitos secuencialmente mediante el decodificador decimal IC18 y coloca el código de segmentos correspondiente al dígito que se pretende encender en el instante en el que este dígito está seleccionado en el port A de la VIA IC19. En esta práctica vamos a comprobar el funcionamiento de este decodificador.

Como en la práctica anterior, para su realización se utiliza el analizador lógico AL-1610 de I.E. Prómax,SA.

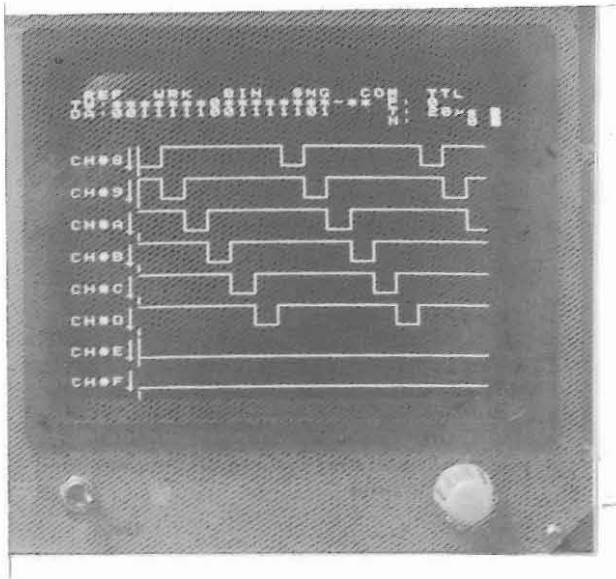


Figura 3.22. Señales de selección de dígito

a. Apagar el Microinstructor.

b. Conectar las salidas de los decodificadores a los canales del Analizador Lógico según la siguiente relación:

Salidas decodificador	Canales AL-1610
Pin 10 (IC18)	Canal 8
Pin 9 (IC18)	" 9
Pin 7 (IC18)	" A
Pin 6 (IC18)	" B
Pin 5 (IC18)	" C
Pin 4 (IC18)	" D

Programar la condición de disparo del Analizador Lógico cuando se cumpla que el canal 8 se hace "0".

Programar la frecuencia de muestreo a 10 usg.

c. Encender el Microinstructor y disparar el Analizador Lógico de manera que quede a la espera de que se cumpla la condición de disparo para empezar la adquisición de datos.

d. Una vez hecha la adquisición el display mostrará un diagrama como el de la figura 3.22.

CAPITULO 4

UTILIZACION DEL MICROINSTRUCTOR

Una vez estudiado cada uno de los componentes del Microinstructor MI-650C, vamos a analizar en este capítulo el modo de utilizarlo.

El sistema de comunicación entre el usuario y el Microinstructor es interactivo, esto es, el Microinstructor indica, en cada paso del proceso de introducción de una orden, qué tipo de dato está esperando, e interpreta automáticamente cada tecla pulsada como una función o como un valor numérico. Según la función introducida, solicita al usuario los datos auxiliares que requiere para su ejecución. De este modo queda eliminada la ambigüedad que existe en el teclado hexadecimal entre dígitos y funciones. En efecto, suponiendo el equipo en reposo, la primera acción realizada es indicar la operación que deseamos efectuar. En este caso, la tecla pulsada se interpreta como función. A partir de este momento y hasta el fin de la ejecución de la instrucción, sólo se requieren datos, por lo que la tecla pulsada se interpreta como el valor hexadecimal asociado con ella.

Mientras el Microinstructor permanece en este estado de espera y presentación de datos en el display se encienden los puntos para indicar el lugar, tamaño, etc, del dato solicitado.

El estado de reposo del equipo se establece al pulsar la tecla de "RESET" o "BREAK" y se reconoce por el encendido del segmento central de cada dígito del display.

4.1 INICIALIZACION DEL MICROINSTRUCTOR

a. Pulsar la tecla **[RESET]**.

Al pulsar esta tecla se produce el mismo efecto que al poner en marcha el equipo, se inicializan los registros y se efectúa un test de los diodos LED y los microinterruptores. Al final se presenta la indicación de reposo con todos los segmentos centrales del display encendidos.

b. Pulsar la tecla **[MEM/0]**.

El display se altera y muestra una serie de datos que más adelante interpretaremos. Lo más importante a observar en estos momentos es que ahora sí existen puntos encendidos en el display, por lo que no está ya en situación de reposo. Si se pulsa ahora una tecla hexadecimal cualquiera, aparecerá en el display la cifra de la tecla pulsada. Como se ha dicho anteriormente el Microinstructor sabe en todo momento si la tecla pulsada corresponde a una función o a un dígito hexadecimal.

c. Pulsar la tecla **[RESET]**.

Al soltarla el equipo se vuelve a inicializar y el display queda con los segmentos centrales encendidos.

d. Pulsar la tecla **[STOP/BRK]**.

Se enciende el segmento central en todos los dígitos. El equipo está ahora también en situación de reposo. Esta función está prevista para efectuar una operación análoga a "RESET", pero de tal manera que solamente interrumpe la tarea en curso, devolviendo el control al monitor pero sin efectuar la inicialización de los registros de la CPU ni se reescribe la zona de datos del programa monitor (scratch).

Básicamente tiene dos aplicaciones:

- Abandonar el programa de usuario cuando está ejecutando un bucle sin fin. En este caso, tras la reentrada al monitor e inspeccionando los registros imagen que retienen en memoria el estado de los registros de la CPU durante la ejecución del programa de usuario, podemos saber el estado de la CPU en el momento de la interrupción.

- Suspender una función de teclado en un punto intermedio (p.e. durante un diálogo para situar un vector en una dirección determinada) si hemos iniciado la función inadecuadamente, si algún parámetro ya introducido es erróneo, etc.

POR TANTO, ES IMPORTANTE RECORDAR QUE PARA VOLVER A LA SITUACION INICIAL DEL EQUIPO SE DEBE PULSAR LA TECLA **[STOP/BREAK]**. UNICAMENTE SE HA DE PULSAR **[RESET]** CUANDO SE DESEE REINICIALIZAR TOTALMENTE EL MICROINSTRUCTOR O CUANDO LA FUNCION "BREAK" SEA INOPERANTE.

4.2 INSPECCION Y ALTERACION DE MEMORIA

4.2.1 Visualización del contenido de la memoria

a. Pulsar **[RESET]**.

En el display aparece la indicación inicial y el Microinstructor queda a la espera de que se le introduzca una función.

b. Pulsar **[MEM/0]**.

Inmediatamente, en el display aparecerá:

Campo de Direcciones: se encenderán los puntos de los cuatro dígitos que componen este campo y aparecerá un número hexadecimal de cuatro dígitos.

Campo de Datos: el texto "ad", indicando que el número en el campo de dirección es una direccion de memoria.

c. Pulsar **[0] [4] [0] [0]**.

Podemos observar que tras cada pulsación, los dígitos se desplazan a la izquierda y el número correspondiente a la última tecla pulsada aparece en el dígito de la derecha. Cuando se hayan pulsado más de cuatro teclas, los primeros números introducidos desaparecerán del display. Este sistema permite corregir errores, ya que sólo son válidos los números presentados en el display. Por lo tanto si la dirección introducida es errónea, bastará teclear de nuevo la correcta.

d. Pulsar **[FIN]**.

Esta función indica al Microinstructor que se ha terminado la introducción de la dirección de la memoria a analizar. Una vez pulsada esta tecla se producen las siguientes acciones:

- Desaparece el texto "ad" quedando sustituido por el dato actual en memoria.

- Los puntos decimales se encienden en el campo de datos, indicando con ello que, si se pulsa alguna tecla numérica, se interpretará como un dígito hexadecimal que inicia la introducción de un nuevo dato en la posición de

memoria seleccionada. Por este procedimiento, un dato contenido en memoria RAM puede ser modificado.

- Para finalizar la función de introducción de datos, pulsar la tecla **FIN** y se volverá al estado inicial. Esta situación es equivalente a la que se alcanza trás pulsar la tecla **[STOP/BRK]**, quedando el Microinstructor a la espera de recibir una nueva orden.

4.2.2 Introducción de datos en la memoria

a. Pulsar **[STOP/BRK] [MEM/0] [4] [0] [0] [FIN].**

El Microinstructor nos indicará que el dato que contiene la dirección en este momento puede ser alterado (puntos en los dígitos de datos encendidos).

b. Pulsar **[E] [A].**

Siguen encendidos los puntos en los dígitos de datos, indicando que si este dato no es el correcto puede ser cambiado.

c. Pulsar **[+].**

En este momento hemos retenido en la memoria 0400H el dato. En el display aparecen ahora la siguiente dirección (401H) y el dato que contiene.

d. Pulsar **[4] [C] [+].**

Igual que en los apartados b y c, el contenido del campo de datos se ha almacenado en la dirección de memoria indicada (401H), mostrando el campo de direcciones del display la siguiente dirección (402H).

e. Pulsar **[0] [0] [+].**

El dato 00 es retenido en la dirección 0402H.

f. Pulsar **[0] [4] [+].**

El dato 04 es retenido en la dirección 403H.

En este momento la memoria contiene:

DIRECCIONES	DATOS	INSTRUCCION
400	EA	NOP
401	4C	JMP 400H
402	00	
403	04	

g. Pulsar **[-]**.

Esta tecla, al contrario que la tecla **[+]**, decrementa en una unidad la dirección de la memoria que en estos momentos aparece en el display, de tal forma que podemos comprobar los datos introducidos. En caso de tener algún dato equivocado seguir el procedimiento de la sección siguiente.

h. Pulsar **[FIN]** para terminar la entrada o visualización de datos y volver al punto de entrada de nuevas órdenes.

4.2.3 Corrección de errores

a. Pulsar **[MEM/0] [4] [0] [0]**.

Supongamos que esta dirección 400H se ha tecleado por error y en su lugar debería haberse introducido la 410H.

b. Pulsar **[4] [1] [0]**.

Aquí podemos comprobar lo que se ha explicado en el apartado 4.2.1.

c. Pulsar **[FIN] [6] [8]**.

Hemos entrado el dato 68 en la dirección de memoria 410H. Si en realidad el dato que queríamos introducir es el 96, pulsamos seguidamente: **[9] [6] [+]**.

Supongamos que también nos hemos equivocado, y que deberíamos haber introducido 88. Ahora para corregirlo pulsamos: **[-] [8] [8] [+]**

En la dirección 410 quedará introducido el dato correcto 88. Para comprobarlo bastará pulsar: **[-]** y el display indicará "0410 88".

> **NOTA:** Recordar que para introducir un dato en memoria tiene que ir seguido de la tecla **[+]** o **[-]**. Al pulsar una de estas dos teclas es cuando se lleva a cabo la escritura del nuevo valor en memoria RAM y se verifica que el contenido actual de la memoria coincide con el dato introducido. Si se sale de esta función sin pulsar **[+]** o **[-]**, no se altera la posición de memoria seleccionada.

d. Pulsar ahora: **[FIN] [MEM/0] [F] [7] [D] [0] [FIN].**

El display indicará F7D0 18.

Pulsar: **[3] [6] [+].**

En este caso la dirección no se incrementa como en en apartado c, apareciendo de nuevo en el display 18. Esto se debe a que el equipo incluye una verificación de que el dato ha sido correctamente introducido y como en este caso estamos intentando alterar una memoria ROM la operación queda anulada. Salir de esta función pulsando la tecla **[FIN].**

En resumen, en este apartado se ha analizado el funcionamiento de varias teclas del Microinstructor:

La tecla **[MEM/0]** permite llamar una posición de memoria; al pulsar esta tecla la dirección que aparecerá en el display es la última que ha sido utilizada (esto es válido siempre que estemos funcionando en inspección de memoria).

La tecla **[FIN]** indica la finalización de la entrada de un dato o dirección, relacionados con la ejecución de alguna función del Microinstructor. En el caso de modificación del contenido de la memoria, visualizar en el display el dato que contiene la dirección de memoria seleccionada.

Si estamos dentro de una función seleccionada, su pulsación devuelve el control al programa principal de control, quedando a la espera de que una nueva orden sea introducida desde el teclado.

La tecla **[+]** memoriza el dato que contiene el display e incrementa la dirección de memoria en una unidad.

La tecla **[-]** memoriza el dato que contiene el display y decrementa la dirección de memoria en una unidad.

La tecla **[STOP/BRK]** interrumpe la función que se está realizando devolviendo el Microinstructor a la situación de reposo.

4.3 PROTECCION DE LA RAM CONTRA ESCRITURA

Esta función sólo es válida para el bloque de memoria comprendido entre 400H y 1FFFH.

a. Pulsar **[RESET]**

   Esto inicializa a "libre" la memoria en caso de haber sido protegida inadvertidamente por manipulaciones anteriores.

b. Observar el bloque de memoria entre 0400H y 1FFFH. Introduciendo datos como en la práctica 4.2.2 se puede comprobar que se comporta como RAM, ya que se pueden introducir y modificar datos. Devolver el equipo a la situación de reposo pulsando la tecla de **[RESET]**.

c. Pulsar **[WRP/8]**.

   Sobre el display aparecerá momentáneamente el texto "ON", indicando que la memoria ha sido protegida.

d. Repetir el apartado b, viendo que ahora el comportamiento de la memoria es similar a una ROM (no se pueden modificar los datos). Volver a la situación de reposo pulsando la tecla **[FIN]**.

e. Pulsar **[WRP/8]**.

   Aparece momentáneamente el texto "OF" y la memoria queda libre de nuevo. Esta operación puede realizarse tantas veces como se quiera. El estado protegida/no protegida de la memoria sólo se puede alterar con la pulsación de las teclas **[WRP/8]** o la tecla **[RESET]**. En este último caso, la memoria pasa siempre a situación de "libre" (no protegida).

   La protección de un bloque de memoria es muy conveniente para evitar que durante la ejecución de un programa éste pueda alterarse a si mismo por una instrucción errónea que hayamos introducido.

4.4 EJECUCION DE UN PROGRAMA DE USUARIO

Durante todo el tiempo en que hemos estado haciendo los ejercicios anteriores, el Microinstructor ha estado ejecutando exclusivamente un programa denominado **Monitor.** La ejecución de un programa de usuario consiste en forzar a la CPU a que abandone la ejecución del programa monitor y pase a interpretar y ejecutar las instrucciones de otro programa.

4.4.1 Modo continuo (RUN)

En este apartado utilizaremos el programa introducido en el apartado 4.2.2. Este programa consta de dos instrucciones. La primera, que es de un solo byte, es un NOP y su función es no efectuar ninguna operación. Normalmente se utiliza para introducir un pequeño retardo o bien para suprimir instrucciones sin interrumpir la ejecución del programa. La siguiente instrucción es de 3 bytes. El primero de ellos corresponde al código de operación que indica que se trata de una instrucción de salto (JMP) absoluto a la dirección indicada por los dos bytes siguientes.

a. Introducir este programa según lo explicado en el apartado 4.2.2.

b. Pulsar **[STOP/BRK] [GO/1] [4] [0] [0].**

c. Pulsar **[FIN].**

   El programa está ahora ejecutándose. Obsérvese que los dígitos del display están ahora apagados. Ello es debido a que el programa es un bucle cerrado, es decir, siempre está corriendo de la dirección 0400 a la 0401, y nunca regresa al programa monitor del Microinstructor.

d. Pulsar **[STOP/BRK].**

   Observamos ahora que aparecen los segmentos en el display, lo cual nos indica que hemos parado el programa de usuario y hemos devuelto el control al programa monitor.

4.4.2 Modo paso a paso

Ya hemos comentado anteriormente que el Microinstructor incluye el hardware y el software necesarios para la ejecución de programas paso a paso.

Esta operación se activa y desactiva mediante sucesivas pulsaciones de la tecla **[RUN/STEP]** y su estado queda indicado por un LED amarillo situado junto a esta tecla. Cuando está encendido, quiere decir que el programa se ejecuta paso a paso (STEP) y, cuando está apagado, de forma continua (RUN).

a. Con el mismo programa anterior pulsar: **[RUN/STEP].**

   Inmediatamente se enciende el LED de la función "STEP" indicando que el Microinstructor está preparado para ejecutar el programa paso a paso.

b. Pulsar: **[GO/1] [4] [0] [0].**

   Notar que, aunque la dirección es 0400H no es necesario teclear los ceros situados a la izquierda, ya que al introducir la primera cifra los dígitos restantes se ponen automáticamente a cero.

c. Pulsar: **[FIN].**

   Inmediatamente el display indicará 0401 en el campo de direcciones y 4C en el de datos, que representa el código de operación de la siguiente instrucción. Esto nos indica que el Microinstructor, al pulsar **[FIN],** ha ejecutado la instrucción NOP y, al finalizar ésta, devuelve el control al programa monitor para mostrar la dirección de la instrucción siguiente, con su código de operación.

d. Pulsar: **[STEP/C].**

   De nuevo el display indicará 0400 en el campo de direcciones y EA en el de datos. El programa ha efectuado la instrucción de salto JMP, pasando luego a monitor mostrando el código de operación de la siguiente instrucción en el display.

e. Si seguimos pulsando **[STEP/C]** se irán repitiendo los apartados c y d.

f. Para parar la ejecución paso a paso y pasar a ejecutar otra función como puede ser la de visualizar alguna posición de memoria, bastará con pulsar la tecla **[STOP/BRK]**. Si en lugar de esta pulsamos **[RESET]** la función queda anulada, apagándose el LED.

En resumen la función "STEP" nos permite ejecutar un programa paso a paso, es decir, instrucción a instrucción, lo cual facilita el seguimiento de su ejecución para la detección de los posibles errores y el punto en el que éstos se producen, cosa muy difícil de conseguir en el modo RUN. De este modo, mediante la ejecución paso a paso, podemos visualizar el estado de la memoria y de los registros internos de la CPU en todo momento como veremos en la siguiente práctica.

4.5 VISUALIZACION DE LOS REGISTROS DE LA CPU

Vamos a realizar un pequeño programa de entrada de datos en los registros A, X e Y, cuyo diagrama de flujo es el de la figura:



Figura 4.1. Escritura de registros

El programa será el siguiente:

```
DIRECCIONES      DATOS     INSTRUCCION      COMENTARIOS
-----------      -----     -----------      -----------

     400          A9        LDA #12H        ;dato en registro A
     401          12
     402          A2        LDX #34H        ;dato en registro X
     403          34
     404          A0        LDY #56H        ;dato en registro Y
     405          56
     406          4C        JMP 400H        ;salta y repite
     407          00
     408          04
================================================================
```

La tecla del Microinstructor que permite la visualización de los registros es la **[REG/2].**

a. Pulsar **[RESET].**

b. Pulsar **[REG/2].**

El display nos muestra "F. 7. D. 0." en el campo de

direcciones y "P C" en el de datos, indicando cual es el contenido del Contador de Programa.

c. Pulsar **[+]**.

En el campo de direcciones del display aparece "F. F." y en el campo de datos el indicador de que corresponde al contenido del Stack Pointer o Puntero de Stack "rS".

d. Pulsar **[+]**.

En el campo de direcciones del display aparece "0. 0." y en el campo de datos el indicador de que corresponde al contenido del Acumulador "AC".

e. Pulsar **[+]**.

En el campo de direcciones del display aparece "0. 0." y en el campo de datos el indicador de que corresponde al contenido del registro de indice X "rX".

f. Pulsar **[+]**.

En el campo de direcciones del display aparece "0. 0." y en el campo de datos el indicador de que corresponde al contenido del registro de indice Y "rY".

g. Pulsar **[+]**.

En el campo de direcciones del display aparece "0. 0." y en el campo de datos el indicador de que corresponde al contenido del registro de flags o Status "rP".

h. Pulsar **[+]**.

Se repite de nuevo el apartado b.

i. Esta visualización del estado de los registros internos de la CPU tambien puede realizarse en sentido contrario mediante la tecla **[-]**.

Obsérvese que al pulsar **[RESET]** se inicializa el Stack Pointer a FF y los demás registros a 00. El PC apunta a la

dirección F7D0H del monitor.

Entrar el programa que se ha indicado anteriormente del modo que ya se ha descrito en apartados anteriores.

a. Pulsar **[MEM/0] [4] [0] [0] [FIN]**

b. Pulsar **[A] [9] [+].**

El dato 0A9H correspondiente al código de operación de la instrucción LDA # (LDA con direccionamiento inmediato), se ha memorizado en la dirección 400H.

c. Pulsar **[1] [2] [+].**

El dato 12H es introducido en la dirección 401H.

d. Continuar introduciendo todos los datos en las direcciones correspondientes.

e. Una vez introducidos todos los datos pulsar **[-].** Como ya sabemos esto nos permite retroceder y verificar si los datos introducidos son correctos.

f. Cuando se hayan corregido todos los errores se puede proteger la memoria contra escritura pulsando las teclas **[FIN] [WRP/8].**

g. Pulsar **[RUN/STEP].**

El LED "step" se encenderá.

h. Pulsar **[GO/1] [4] [0] [0] [FIN].**

El campo de direcciones del display indica "0402" y el de datos "34". La dirección 0402H corresponde a la siguiente instrucción ha ejecutar. La instrucción 0400H ya ha sido ejecutada y por tanto el dato 12H ha sido introducido en el Acumulador.

i. Pulsar **[REG/2].**

El display nos muestra "0402" "Pc", indicando que el registro contador de programa PC contiene la dirección de la siguiente instrucción a ejecutar 0402.

j. Pulsar **[+]**.

El display muestra "1.2." "Ac", indicando que la instrucción LDA #12H ha sido correctamente ejecutada y, por lo tanto, el acumulador contiene el dato 12, en lugar del 00 que tenía después de inicializar el sistema.

k. Pulsar **[+]**.

El registro X contiene 00.

l. Pulsar **[FIN] [STEP/C]**.

Se ha ejecutado la siguiente instrucción, LDX #34H. El display muestra "0404" "A0".

m. Pulsar **[REG/2]**.

El display nos muestra "0404" "Pc".

n. Pulsar **[+] [+] [+]**.

El display indica "3.4." "rX" que corresponde al dato introducido en el registro X por medio de la instrucción LDX #34H. Después de inicializar el sistema este registro contenía el dato 00.

o. Pulsar **[+]**.

El registro Y contiene 00.

p. Pulsar **[FIN] [STEP/C]**.

El display nos muestra "0406" "4C" y se ha ejecutado la instrucción LDY #56H.

q. Pulsar **[REG/2]**.

El display nos muestra "0406" "Pc".

r. Pulsar **[+] [+] [+] [+]**.

El display nos indica "5.6." "rY", que corresponde al dato introducido en el registro Y.

s. Pulsar **[+]**.

El display indica "2.0." "rP", que corresponde al valor contenido en el registro de flags y que antes era 00. Al ejecutar algunas intrucciones el valor de los diferentes bits de este registro se ven afectados.

t. Pulsar **[FIN] [STEP/C]**.

El display muestra "0400" "A9" y se ha ejecutado la instrucción de salto JMP para repetir de nuevo el programa.

> **NOTA:** Como se habrá podido comprobar, cuando se visualizan los registros aparecen también los puntos, indicando siempre que se puede variar el valor. Por ejemplo, si el registro de índice Y contiene "5.6." como en el apartado anterior, podemos cambiar este valor a "2.0." con sólo pulsar **[2] [0] [+]**

Con esta práctica se ha comprobado que se puede trazar un programa paso a paso y seguir en todo momento el proceso de su ejecución mediante la visualización de los registros imagen que son una copia en memoria del estado de los registros de la CPU durante la ejecución del programa.

Al manipular los registros imagen debemos tener en cuenta los siguientes hechos:

- El valor del registro imagen PC pasa al contador de programa antes de ejecutarse la siguiente instrucción del programa. Por tanto, puede ser alterado para cambiar la dirección de la siguiente instrucción a ejecutar.

- Los valores de estos registros sólo se corresponden con los que existirán en la CPU en caso de que el retorno del control al monitor se efectúe por uno de estos procedimientos:

  - instrucción BRK.

- ⊕ pulsación de tecla **[STOP/BRK]**.
- ⊕ reconocimiento de interrupción (IRQ o NMI).
- ⊕ trazado del programa (realmente interrupción NMI) y nunca por una instrucción RTS.

- Tras pulsar **[RESET]** los registros se inicializan a valores fijos.

4.6 VECTORES

Un vector es una dirección de memoria contenida en dos direcciones consecutivas, de tal modo que la primera dirección contiene la parte baja de este vector y la segunda dirección la parte alta.

No todas las parejas de bytes en memoria se pueden considerar vectores. Para que lo sean en el programa se les debe tratar como a tales.

Con el fin de poder manejar este tipo de información, el Microinstructor dispone de una función específica.

4.6.1 Inicialización de vectores

Veamos cuales son las operaciones necesarias para la introducción de un vector.

a. Pulsar **[STOP/BRK].**

   En este momento el equipo estará en reposo, presentando los segmentos en el display.

b. Pulsar **[STV/3].**

   En los dígitos del campo de dirección del display presentará sólo puntos, y en los del campo de datos la indicación "Ud". Con este mensaje el Microinstructor está solicitando la entrada del vector.

c. Pulsar **[1] [2] [3] [4] [FIN].**

   El display pasa a una situación igual a la anterior, pero mostrando ahora el texto "UA" en el campo de datos, indicando que espera la dirección de memoria en la que se ha de colocar el vector.

d. Pulsar **[5] [0] [0] [FIN].**

   El display pasa al estado de reposo.

e. Pulsar **[MEM/0]**

Nótese que aparece en el display 0500 en dirección.

f. Pulsar **[FIN]**

El display nos muestra que la dirección 0500H contiene el dato 34.

g. Pulsar **[+]**

Vemos que en la dirección 0501H aparece 12H.

La función **[STV/3]** automáticamente ha introducido correctamente el vector en las direcciones 0500 y 0501. En caso de no existir ésta, habríamos tenido que efectuar lo siguiente:

[MEM/0] [5] [0] [0] [FIN]

[3] [4] [+] [1] [2] [+] [FIN]

Por tanto habríamos tenido que introducir primero el byte bajo y después el byte alto. Frecuentemente, se producen errores cuando se quiere introducir un vector manualmente, confundiendo la posición correspondiente al byte alto y la correspondiente al byte bajo. Utilizando la función **[STV/3]** no es necesario invertir el orden de los bytes, ya que el Microinstructor lo realiza automáticamente.

4.6.2 Programa vectorizado

Vamos a realizar un pequeño programa en el que se utiliza un vector. El programa consiste en transferir un vector a la posición de memoria JUMP para que efectúe un salto vectorizado al programa monitor del Microinstructor que muestra el mensaje "650C" en el display. Su listado es el siguiente.

DIRECCIONES	DATOS	ETIQUETA	INSTRUCCION	COMENTARIOS
400	A2		LDX #00H	;inicio índice
401	00			
402	BD	CONT	LDA TABLA,X	
403	10			
404	04			
405	9D		STA JUMP,X	;transfiere...

```
406         12                                  ;... vector
407         04
408         E8              INX
409         C9              CMP #02H
40A         02
40B         F0              BEQ CONT
40C         F5
40D         6C              JMP (JUMP)
40E         12
40F         04
410         71      TABLA
411         FF
412         xx      JUMP
413         xx
=============================================================
```

Colocar el vector FF71H en la memoria TABLA utilizando la función "STV". La memoria 040EH es asignada como TABLA, la 040FH como TABLA+1. La dirección 0411H corresponde a JUMP+1 y la 0410H a JUMP. Su contenido inicial es indiferente para el resultado del programa, lo que se indica con la notación "xx".

El diagrama de flujo de este programa sería el siguiente:

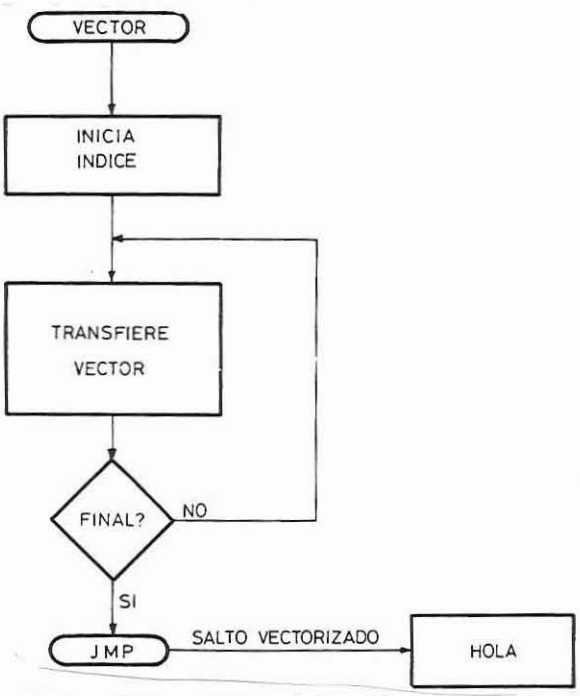


Figura 4.2. Programa vectorizado

Para ejecutar este programa:

a. introducir el listado.

b. Pulsar **[GO/1] [4] [0] [0] [FIN]**

Podemos ver que al ejecutar el programa, éste pasa a mostrar el mensaje "650 C" en el display.

4.7 PUNTO DE RUPTURA (BREAKPOINT)

A veces es interesante ejecutar un programa en modo RUN y detener la ejecución en un determinado punto. Esto puede realizarse colocando una instrucción BRK en el programa que obligará al procesador a efectuar un salto desde el programa usuario al monitor del Microinstructor ya que, como se vió en el capítulo anterior, esta instrucción fuerza una interrupción por software, y después de recoger el vector de interrupción se pasa a ejecutar la subrutina de interrupción.

El Microinstructor permite manejar un punto de ruptura (breakpoint) para la depuración de programas. Para ello incluye 3 funciones que permiten la colocación, reposición e inspección del mismo.

Para ilustrar el funcionamiento y la aplicación de los puntos de ruptura vamos a utilizar un programa que consiste en colocar un dato fijo en un bloque de memoria y, al finalizar, retorna al monitor del Microinstructor. En la figura 4.3 podemos ver el diagrama de flujo correspondiente.

El dato que introduciremos será el 00 y se grabará en las direcciones comprendidas entre la 501H y la 510H. Su listado es el siguiente:

```
DIRECCIONES   DATOS  ETIQUETA  INSTRUCCION      COMENTARIOS
-----------   -----  --------  -----------      -----------

    400         A9     INICIO     LDA #00H    ;dato a colocar
    401         00
    402         A2                LDX #10H    ;tamaño bloque
    403         10
    404         9D     BUCLE      STA 500H,X  ;a memoria
    405         00
    406         05
    407         CA                DEX
    408         D0                BNE BUCLE   ;repite hasta fin
    409         FA
    40A         60                RTS
==========================================================
```

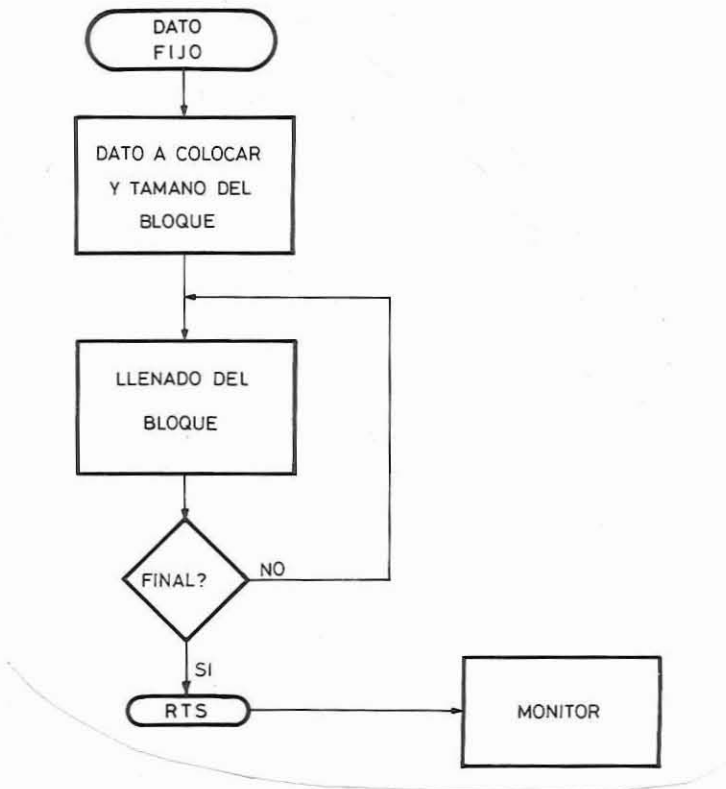


Figura 4.3. Introducción de un dato fijo en un bloque de memoria

a. Pulsar **[STOP/BRK]** y introducir el listado anterior.

b. Observar que el diodo LED "step" está apagado y por tanto está en modo RUN. En caso contrario, pulsar la tecla **[RUN/STEP]**

c. Pulsar **[GO/1] [4] [0] [0] [FIN].**

   El programa es ejecutado, finalizando en el monitor del Microinstructor.

d. Pulsar **[MEM/0] [5] [0] [0] [FIN] [+]**, etc, hasta la memoria 510H. Vemos que no contienen 00 como se pretendia. Ello es debido a que en el programa se ha introducido un

error, para poder ilustrar la función "Breakpoint" del Microinstructor.

e. Pulsar **[SETB/B]**.

El dislplay preguntará por una dirección mediante el texto "br".

f. Pulsar **[4] [0] [7]**.

Sólo es válida una dirección que contenga el código de operación de una instrucción del programa. En este caso la dirección 407 contiene el código de operación de la instrucción DEX.

g. Pulsar **[FIN]**.

En este momento el dato existente en la dirección de la memoria indicada se guarda para su posterior reposición y en su lugar se coloca una instrucción BRK (código de operación 00).

h. Podemos comprobar si en el Microinstructor hay en estos momentos algún breakpoint. Para ello pulsar la tecla **[BR?/A];** en el display aparece, en el campo de dirección "0407" y, en el de dato "b?", lo que nos indica que el breakpoint ha sido introducido correctamente. Pulsar **[FIN]**.

i. Pulsar **[GO/0] [4] [0] [0] [FIN]**.

Ahora el programa ha sido ejecutado, pero al llegar a la dirección 0407H ha encontrado un breakpoint que detiene la ejecución, mientras que el display indica la dirección siguiente a ejecutar 0408H.

j. Pulsar **[REG/2] [+] [+]**.

Podemos ver que el Acumulador no contiene 00, por lo que deducimos que en el programa existe una instrucción que nos altera este dato. Observando el listado vemos que la dirección 0404H que corresponde a la instrucción LDA 500H,X debe ser STA 500H,X.

k. Pulsar **[FIN] [MEM/0] [4] [0] [4] [FIN]**

[9] [D] [+] [FIN], el programa ahora está corregido.

l. Repetir los apartados i y j. Se puede comprobar que ahora el Acumulador contiene 00 y, por lo tanto, tenemos el programa corregido.

m. Pulsar **[CLRB/9].**

En el display aparece momentáneamente el mensaje "OF", lo cual nos indica que el breakpoint ha sido eliminado y en su lugar se ha restablecido el código de operación correcto.

n. Repetir los apartados c y d, comprobando ahora que entre las direcciones 501H y 510H de memoria el contenido es 00.

En resumen la función **[SETB/B]** coloca automáticamente el código de operación 00 y guarda el código de la instrucción del programa.

Cuando se utiliza esta función, en el display pueden aparecer los siguientes errores:

ERROR 05: Se produce tras la pulsación de la tecla **[SETB/B]** e indica que ya hay un breakpoint en el programa (solamente puede mantenerse uno en la memoria). Deberá eliminarse el anterior antes de colocar el nuevo.

ERROR 03: Se produce al introducir la dirección del punto de ruptura, tras la pulsación de la tecla **[FIN].** Indica que la memoria no ha admitido el cambio, debido a que esta zona de memoria corresponde a ROM o que la memoria está protegida.

Téngase en cuenta que, si se efectúa un RESET cuando hay fijado un breakpoint, la información que se guardaba sobre cuál era el código de operación de la instrucción que ocupaba el lugar que ahora ocupa el "breakpoint", se pierde, con lo cual deberá reponerse manualmente, mediante la función MEM.

Cuando se inspecciona la dirección del breakpoint con la tecla **[BR?/A]** puede aparecer el siguiente error:

ERROR 04: Si no existe BRK inicializado.

Cuando se utiliza la función **[CLRB/9]** pueden aparecer los siguientes errores:

ERROR 03: Al pulsar **[CLRB/9]** existe breakpoint, pero la memoria está protegida y no es posible su eliminación.

ERROR 04: No existe breakpoint.

El seguimiento del programa mediante breakpoints es útil para detectar errores en programas largos, colocando el breakpoint en diferentes secciones hasta encontrar el error.

4.8 CALCULO DEL OFFSET

Como se ha visto en el capítulo anterior, existe un tipo de instrucciones de salto o "branch" en las cuales el operando es un número en complemento a dos cuyo valor se suma al PC actual para determinar la dirección a la cual se efectúa la transferencia del programa. Estas instrucciones se denominan de "ramificación".

Durante el diseño del programa se conocen la dirección en la cual está la instrucción de salto y la dirección a la que se desea transferir el control, debiendo calcularse el valor del offset para definir el byte de operando. Este cálculo requiere efectuar operaciones con números binarios, normalmente difíciles y sujetas a errores, por no ser el sistema de numeración al que estamos acostumbrados.

El valor del offset en una instrucción de salto es un número de 8 bits en complemento a dos, por lo que su valor se limita al rango +127 a -128. Un número positivo indica un salto hacia adelante, esto es, aumentando el PC y un salto hacia atrás implica un valor de offset negativo.

En el Microinstructor, la función "CALC" determina automáticamente el valor de este offset partiendo de las direcciones origen y destino del salto.

Para comprobar el funcionamiento de esta función utilizaremos el programa de la sección anterior. En él hay una instrucción de ramificación en la dirección 0408H (BNE BUCLE). La dirección a la que debe saltar el programa es la 0404H.

a. Pulsar **[STOP/BRK]**

b. Pulsar **[CALC/D].** El Microinstructor solicita con el texto "ca" un número de 4 dígitos.

c. Introducir en primer lugar la parte baja de la dirección del código de la instrucción de salto **[0] [8]** e inmediatamente la parte baja de la dirección del código de operación de la instrucción a la cual se ha de transferir control **[0] [4]**

d. Pulsar **[FIN].**

En en campo de dirección del display siguen apareciendo los datos antes introducidos, pero en el campo de datos el texto

"ca" se sustituye por un número que es el valor del offset a colocar en la instrucción, en este caso **FA**.

Antes de dar por bueno este valor, debemos verificar que sea lógico. Para ello emplear las siguientes reglas:

- Si el salto era hacia adelante, el offset estará comprendido entre 0 y 7FH (0 y 127).

- Si el salto era hacia atrás, el offset estará comprendido entre FFH y 80H (-1 y -128).

En caso de no ser así, se está intentando dar un salto de mayor longitud que la permitida, por lo que, deberá alterarse el programa para llevar el salto dentro de los márgenes permitidos.

Veamos varios ejemplos de aplicación:

- Salto desde 40AH a 423H. El parámetro de la operación será 0A23, dando un resultado de 17H. Es un resultado correcto ya que es positivo y el salto es hacia adelante.

- Salto desde 40AH a 490H. El parámetro será 0A90 y el resultado 84H. Este resultado no es válido ya que el salto es hacia adelante y el offset negativo. Se deberá retocar el programa.

- Salto desde 490H a 503H. El parámetro será 9003. Nótese que se ha ignorado el hecho de que las partes altas de las direcciones no sean iguales. El resultado es 71H. Es válido por ser positivo y el salto ser hacia adelante.

- Salto desde 432H a 40AH. El parámetro será 320A y el resultado D6H. Es un resultado válido ya que el salto es hacia atrás y por tanto le corresponde un offset negativo.

> **NOTA:** Debe tenerse en cuenta que el cálculo del offset debe efectuarse a partir del valor que tiene el "Program Counter" en el momento del salto y por tanto después de ejecutar la instrucción de branch correspondiente.

En el siguiente fragmento de programa se puede observar como se efectúa el cálculo del offset en el Microinstructor.

En la posición indicada por los guiones '--' se debe introducir el offset que se debe sumar al contenido del PC en caso que la condición de salto sea verdadera. Debido a que una vez leída la instrucción el PC ya está apuntando al principio

de la siguiente instrucción, el offset que le corresponde es 02. Para comprobarlo utilizar la función "CALC" entrando el dato 0004.

```
DIRECCION  CONTENIDO  ETIQUETA  INSTRUCCION
---------  ---------  --------  -----------
  400         D0                 BNE  BUCLE
  401         --
  402         A9                 LDA  #3
  403         03
  404         EA       BUCLE     NOP
=========================================
```

Análogamente, si el salto es hacia atrás, el offset incluye los dos bytes de la instrucción de branch puesto que, cuando se efectúa el salto, esta instrucción ya ha sido ejecutada. En el ejemplo de la sección anterior que hemos desarrollado al principio de este apartado, hemos comprobado como al entrar 0804 en la función "CALC" el offset resultante era FA. Complementando a 2 nos queda un salto negativo de 6 posiciones.

4.9 UTILIZACION DEL CASSETTE

Cuando se trabaja con programas de una cierta complejidad, el hecho de tener que introducirlos en memoria a través del teclado cada vez que se desea utilizarlos da lugar a una gran pérdida de tiempo. Por ello los ordenadores incluyen generalmente algún medio de almacenamiento de información de tipo no volátil, de manera que, los datos almacenados en memoria RAM puedan almacenarse en este dispositivo y recuperarse cuando haga falta (memoria de masa).

En ordenadores de una cierta complejidad se suele emplear el almacenamiento en discos magnéticos ("floppy disk" y "hard disk"). No obstante su implantación es suficientemente costosa como para hacerlo prohibitivo en pequeños sistemas de evaluación y enseñanza. Para estos equipos es típica la utilización de un cassette de audio normal que, aunque no tiene unas prestaciones tan brillantes como los sistemas antes señalados, constituyen un método de almacenamiento idóneo por su simplicidad y bajo costo.

La explicación del hardware que incorpora el Microinstructor para poder controlar la grabación, reproducción y control remoto del motor del cassette se ha hecho en el capitulo 3.

A continuación vamos ha analizar detalladamente la utilización de las cuatro funciones que incorpora el cassette.

4.9.1 Control del motor

Cuando se conecta la unidad de cassette al Microinstructor, el accionamiento del motor se hace de forma remota, bajo control del Monitor. Esta función libera el control del motor, permitiendo en un momento dado, el rebobinado de la cinta, buscar la zona donde se encuentre un determinado programa, etc..., sin necesidad de desconectar ningún cable de control. La tecla que controla esta acción es **[MOT/7]**. Pulsándola una vez, el motor se pone en marcha, apareciendo el texto "On" en el display temporalmente. Una nueva pulsación de esta tecla para el motor, apareciendo el texto "OF" en el display.

Tras la pulsación de **[RESET]** el estado de este control se inicializa en apagado.

Una vez se ha efectuado la conexión puede apretarse la tecla "PLAY" del cassette y comprobarse que pulsando repetidas veces la tecla **[MOT/7]** del Microinstructor el mecanismo del cassette arranca y para sucesivamente.

4.9.2 Almacenamiento de datos en cassette

El control de cassette que incorpora el Microinstructor permite grabar en la cinta los datos contenidos en un cierto bloque de memoria, tanto si corresponden a un programa, como si son datos, etc.

A este bloque de memoria se le asigna un número de identificación por parte del operador, con el fin de poder recuperarlo dentro de una cinta que incluye diversos bloques de datos.

Las operaciones requeridas son las siguientes:

a. Pulsar la tecla **[SAVE/4].** El equipo solicitará el número de identificación del programa, que ha de ser de dos dígitos, mediante el identificador "id". Responder con un número hexadecimal cualquiera, excepto 00 y FFH, que tienen una aplicación especial, como luego veremos. Pulsar **[FIN].**

b. El equipo solicita la dirección inicial del bloque de datos a grabar mediante el texto "di". Introducir esta dirección de cuatro dígitos y terminar con **[FIN]**

c. Por último, el equipo solicita la dirección final del bloque mediante el texto "dF". Después de introducir la dirección final del bloque deseada y antes de pulsar **[FIN]** asegurarse de que:

   - El cassette esté conectado al Microinstructor y hay una cinta en él.
   - Los controles de volumen y tono están en una posición correcta.
   - En caso de que exista control remoto del motor, estén pulsadas las teclas de grabación.

d. Si hay control remoto, pulsar **[FIN].** Si no existe este control, poner el cassette en grabación, esperar 2 ó 3 segundos y pulsar **[FIN].** Al comenzar la grabación de datos, el display se apaga totalmente. Al finalizar, el cassette se parará y el display mostrará la línea típica de reposo. Si no hay control remoto del cassette, pararlo a mano cuando el display muestre esta situación de reposo.

> **NOTA:** Si se aprovecha una cinta ya usada, es conveniente borrarla totalmente o bien en la zona que se desee grabar.

4.9.3 **Verificación de los datos grabados**

Esta función permite comprobar los datos que han sido grabados en cinta. Para ello se efectúa una lectura de los datos grabados, comparándolos byte a byte con los existentes en memoria. Cualquier diferencia da lugar a un error.

Seguir el siguiente procedimiento:

a. Rebobinar la cinta hasta un punto anterior al de inicio de la grabación del programa. Parar el cassette y colocarlo en reproducción si hay control remoto.

b. Pulsar la tecla **[VER/6]**. El equipo solicitará el número de identificación de los datos a verificar. Introducir el mismo número de identificación con que se grabaron estos datos y después pulsar la tecla **[FIN]**.

c. Inmediatamente el display presentará una "S" en el dígito izquierdo, indicando que se están buscando datos. Si hay control remoto, el cassette se pondrá en marcha; en caso contrario apretar la tecla "play" del cassette.

   Cuando el programa encuentre un bloque de datos, desaparecerá la "S" del display y, si el identificador corresponde al número solicitado, comenzará la verificación de los datos grabados, comparándolos con los existentes en memoria pero sin alterarla.

   Si todo el proceso es correcto, al finalizar, el display muestra la línea de reposo. Si por el contrario hay algún problema, aparecerá alguno de los siguientes textos de error:

   - Err 01: Los datos de la cinta no pueden ser leídos correctamente.
   - Err 02: Existe error en los carácteres de control calculados por el programa (checksum).
   - Err 03: Los datos leídos no son iguales a los existentes en memoria.

   En cualquiera de los tres casos, debe repetirse el proceso completo de grabación y de verificación.

   Cuando se haya producido el error 03, al volver al programa Monitor, pulsando la tecla **[MEM/0]**, aparece en el display la dirección de memoria que contiene el dato que no coincide.

Verificar que el dato en la memoria no ha sido alterado entre la grabación y la verificación.

4.9.4 **Lectura de los datos del cassette** Mediante esta función se transfiere a memoria RAM un bloque de datos o programas previamente grabado en cinta.

a. Asegurarse de que la cinta está correctamente colocada y posicionada en un punto anterior al inicio de los datos a recuperar. Asegurarse también de que el cassette está conectado y los mandos de volumen y tono ajustados.

b. Pulsar la tecla **[LOAD/5].** El equipo solicita el número de identificación del programa. Aquí caben tres posibilidades:

   - Introducir el número correspondiente al bloque de datos a recuperar.

   - Introducir como identificador el 00. Esto da lugar a la lectura del primer bloque de datos encontrado, con independencia de su número propio de identificación grabado en la cinta.

   - Introducir como identificador el FFH. Esto permite leer el primer bloque de datos encontrado, pero colocándolo en otra zona de memoria diferente de la que tenía en el momento de la grabación.

c. En caso de emplear el identificador FFH, se deberá responder con las direcciones inicial y final del lugar de memoria donde ha de ir el programa grabado en cinta. Así, si al grabar la cinta se dieran las direcciones 500H y 590H como inicio y final, y deseamos recuperarlo a partir de la dirección 400H de memoria, deberá responderse con 400H y 490H a las direcciones inicial y final solicitadas.

d. Bien tras la introducción de la dirección final, si el número de identificación fue FFH, o bien tras la introducción del número de identificación, si ha sido otro, el cassette se pondrá automáticamente en marcha (pulsar "Play" si no hay control remoto), y aparecerá la letra "S" hasta que se encuentren los datos en cinta con el identificador indicado. En este momento se inicia la lectura de cinta hasta el final del bloque, tras lo cual se para el motor y el display queda en reposo.

Obsérvese que, aproximadamente al cabo de 6 segundos de desaparecer la "S", en su lugar aparecen una serie de

segmentos encendidos que muestran el número de identificación del bloque de datos que se está leyendo, empleándose un método de conversión directa de bits a segmentos.

Durante la lectura pueden detectarse errores tipo 01, 02 ó 03. Los dos primeros son debidos a causas ya citadas. El error 03 en la lectura de datos, se debe a que la memoria no acepta los datos a memorizar, ya sea porque se pretenden grabar en una zona que no corresponde a memoria RAM, ya porque la memoria RAM esté protegida.

En caso de que se produzca algún error, revisar las posiciones de los potenciómetros de volumen y tono, inspeccionar el estado de los cabezales, comprobar que la cinta esté bien insertada, etc.

PRACTICA 4.1 UTILIZACION DE LAS LINEAS DE ENTRADA/SALIDA

En esta práctica vamos a ver el modo de programar las líneas del port A y port B del 65C22 (IC22). Como sabemos estas líneas se pueden programar como entradas o como salidas dependiendo de los valores colocados en DDRA (A403H) y DDRB (A402H).

Empleo de las líneas de entrada.

a. Pular **[RESET].**

   El port A se programa como entradas y el port B como salidas. Durante la inicialización se lee el estado de los microinterruptores. Todos los que estén en la posición "ON" provocarán el encendido del LED correspondiente del port B.

b. Pulsar **[MEM] [A] [4] [0] [3] [FIN].**

   Vemos que su valor es 00, lo cual indica que todas las líneas de PA están definidas como entradas.

c. Pulsar **[FIN] [MEM] [A] [4] [0] [1] [FIN].**

   El valor que aparece en el apartado de DATO del display es el que se corresponde con el estado de las líneas en este momento. Si todos los conmutadores están en "OFF" su estado será FF.

d. Actuar sobre diversos conmutadores. Se verá que el dato en el display es la representación hexadecimal del dato binario seleccionado en los microconmutadores. La posición "ON" equivale a un 0 del bit correspondiente.

e. Pulsar: **[FIN] [MEM] [A] [4] [0] [3] [FIN] [0] [F] [+]**

   Esto define PA0 a PA3 como salidas y PA4 a PA7 como entradas.

f. Pulsar: **[FIN] [MEM] [A] [4] [0] [1] [FIN]**

   Vemos que el dígito de la derecha (línea 0 a 3) tiene el valor 0 y que no es alterado por los cambios en el conmutador. El dígito izquierdo, por el contrario, sigue presentando el valor correspondiente a las posiciones de los conmutadores 4 a 7.

g. Colocando otros valores en DDRA (A403H), veremos que solamente las líneas definidas como de entrada quedan posibilitadas para seguir el estado del conmutador correspondiente.

h. Pulsar [RESET]. Coloca automáticamente 00 en DDRA (A403H) y todas las líneas quedan direccionadas como entradas.

Empleo de las líneas de salida:

a. Poner cuatro conmutadores en "ON" y cuatro "OFF".

b. Pulsar [RESET]. Como se ha dicho, el programa de test que se ejecuta al pulsar dicha tecla, lee el estado del port A y lo escribe en el port B, de manera que todos los conmutadores que estén en estado "ON" encenderán el LED correspondiente del port B.

c. Pulsar **[STOP/BRK] [MEM] [A] [4] [0] [0] [FIN].**

   El dato de esta posición se corresponde directamente con el estado de los LEDs.

d. Pulsar **[5] [5] [+]**

   Inmediatamente los LEDs correspondientes a los bits '1' del dato se encienden. Por otra parte, en RB se puede leer este dato almacenado, con lo que, mediante un programa, la CPU puede saber en todo momento el estado de las líneas de salida de un port.

e. Pulsar **[FIN] [MEM] [A] [4] [0] [2] [FIN] [0] [F] [+]**

   Ahora las líneas PB4 a PB7 pasan a ser entradas y, por tanto, se encienden todos los LED's que corresponden a líneas de entrada ya que los inversores de IC20 e IC23 detectan un estado equivalente a un '1' en sus entradas. Los 4 LEDs inferiores pueden ser alterados como en el caso anterior.

   Nótese que el dígito alto del display permanentemente muestra el valor 'F', nivel '1' de las líneas definidas como entradas.

f. Fijar todas las líneas del port B como entradas (00 en A402H) y colocar el dato 55H en RB (A400H). El dato no es aceptado aparentemente. No obstante, sí ha sido retenido adecuadamente por el 65C22, como podemos ver fijando las líneas de PB de nuevo como salidas (FFH en DDRB). Esta capacidad de poder programar el registro de salida de un port aún cuando no esté definido como salida, puede aplicarse para evitar impulsos espúreos en las lineas de salida.

PRACTICA 4.2 SIMULACION DE UNA PUERTA AND

Como sabemos, la salida de una puerta AND es '1' cuando todas sus entradas son '1'. Cuando alguna entrada es '0' la salida también es '0'. En esta práctica vamos a utilizar las tiras de LEDs y de conmutadores del microinstructor para simular el funcionamiento de una puerta de este tipo.

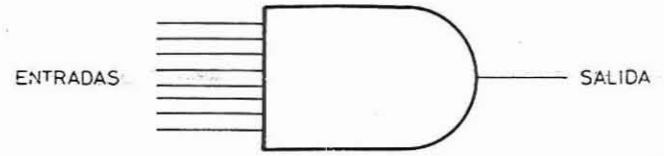


Figura 4.4. Puerta AND de 8 entradas

El diagrama de flujo de este programa es el siguiente:

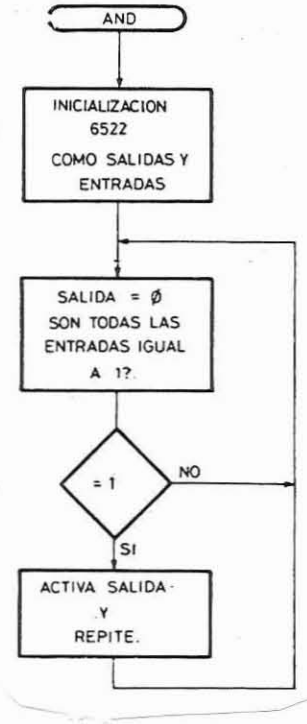


Figura 4.5. Diagrama de flujo puerta AND

Las entradas de la puerta AND las constituyen los microconmutadores asociados al port A y la salida el LED 0

conectado al port B.

El programa por tanto deberá leer las entradas y, solamente en caso de que su valor sea FFH, enviar un '1' lógico a la línea PB0.

PROGRAMA PRINCIPAL

DIRECCION	CONTENIDO	ETIQUETA	INSTRUCCION		COMENTARIO
400	A9	START	LDA	#0FFH	
401	FF				
402	8D		STA	DDRB	;PB a salidas
403	02				
404	A4				
405	A9		LDA	#0	
406	00				
407	8D		STA	DDRA	;PA a entradas
408	03				
409	A4				
40A	8D		STA	RB	;Todo a 0
40B	00				
40C	A4				
40D	A2	BUCLE	LDX	#0	;Salida=0
40E	00				
40F	AD		LDA	RA	
410	01				
411	A4				
412	C9		CMP	#FFH	;Todas entradas=1?
413	FF				
414	D0		BNE	NO	;No
415	02				
416	A2		LDX	#1	;Sí, salida=1
417	01				
418	8E	NO	STX	RB	;Activa salida
419	00				
41A	A4				
41B	4C		JMP	BUCLE	;Repite
41C	0D				
41D	04				

Los valores de los símbolos empleados en este listado son los siguientes:

```
RB    = A400H
RA    = A401H
DDRB  = A402H
DDRA  = A403H
START = 400H
BUCLE = 40DH
NO    = 418H
```

Las primeras instrucciones del programa definen el sentido de las líneas de PA como entradas y las de PB como salidas, fijando todas las líneas de PB a 0. Seguidamente se entra en un bucle en el que constantemente se obtiene el valor de las líneas de entrada y se compara con FFH, colocando en el registro X un 1 ó un 0 según si todas están a 1 o no. Por último, el valor de X se almacena en RB, para activar o no PB0, y se repite el bucle ininterrumpidamente.

Entrar el listado del programa anterior y verificarlo.

Para ejecutar el programa pulsar: [GO] [4] [0] [0] [FIN]. si todas las entradas están en OFF, el LED 0 se ilumina y, si colocamos uno o más de los conmutadores a ON, el LED 0 se apaga. Con ello podemos ver que cumple la función AND.

Se puede simular el comportamiento de una puerta NOR con sólo colocar 00 en la dirección 413H en lugar de FFH.

CAPITULO 5

TECNICAS DE PROGRAMACION

5.1 LENGUAJES DE PROGRAMACION

Los lenguajes de programación son los medios de que dispone el programador para traducir una idea a instrucciones que pueda entender y ejecutar el microcomputador.

Una vez se dispone del soporte hardware adecuado (circuito) a las necesidades que se desean satisfacer, aún queda la labor fundamental que consiste en programar el equipo para que realice las funciones que deseamos.

En el capítulo 3 se ha estudiado con detalle la forma de programación del microprocesador y se ha visto que, aunque los únicos códigos inteligibles por el microprocesador son las secuencias de ceros y unos, el programador utiliza para el desarrollo de sus programas una serie de nemónicos con los cuales se hace más intuitiva la función que se está realizando en cada momento. Una vez se ha diseñado todo el programa es cuando se traducirá cada una de las instrucciones a su equivalente binario, denominado **código máquina.** El programador conoce el número de instrucciones que ocupa su programa y por tanto, puede calcular el tiempo de ejecución.

El proceso de traducción de un programa escrito en código nemónico al código binario que el microprocesador es capaz de interpretar se hace mediante la ayuda de las tablas de equivalencias del microprocesador utilizado. Pero además habrá que traducir a binario todos los números utilizados, calcular los offsets de los saltos relativos, etc...

Cuando los programas son largos este sistema es lento y muy expuesto a errores. En estos casos se utiliza un programa que se encarga de efectuar estas operaciones, denominado **Programa Ensamblador.**

Para poder escribir un programa en lenguaje ensamblador, es necesario tener un buen conocimiento de la estructura hardware del equipo con el que se trabaja, por lo que es un lenguaje ideal para utilizar en equipos de estudio y evaluación. Además, cada microprocesador tiene su lenguaje ensamblador, con sus códigos

nemónicos, adecuado a su estructura hardware.

Como contrapartida, el desarrollo de programas en lenguaje ensamblador es laborioso y, sus resultados, difíciles de interpretar. Su gran dependencia con el hardware, hace que las aplicaciones no puedan trasladarse con facilidad entre distintas CPU's. Estos factores pueden hacer necesario que en algunas ocasiones se utilicen los llamados **lenguajes de alto nivel.**

Sus principales ventajas son la facilidad de traducir una idea estructurada a un programa correcto, la rapidez en el desarrollo de aplicaciones y la "transportabilidad" o capacidad de ejecutar un mismo código fuente (o con pocas modificaciones) en distintos microcomputadores. Este último aspecto se debe a que los lenguajes de alto nivel no están escritos para ningún microprocesador en concreto y a que la mayoría de ellos tienen los mismos tipos de instrucciones. En los lenguajes de alto nivel (BASIC, PASCAL, C, FORTRAN, etc...), las instrucciones del programa son textos alfanuméricos (por ejemplo LET C=12*A), cada uno de los cuales representa una o varias subrutinas en código máquina. De este modo, aunque el programador pierde el control directo de hardware del sistema y los tiempos de ejecución se incrementan al igual que la cantidad de código máquina generado, la labor de programar se hace mucho más rápida.

Dentro de los lenguajes de alto nivel se puede hacer una segunda clasificación que depende del modo en que la CPU interpreta y ejecuta las instrucciones del programa. Esto da lugar a dos tipos básicos de lenguajes: el **lenguaje interpretado** y el **lenguaje compilado.**

En un **lenguaje interpretado,** las instrucciones de alto nivel se guardan en memoria como cadenas de carácteres ASCII que corresponden, cada una de ellas, a una línea del programa fuente. Existe un programa en lenguaje máquina que, cada vez que se ejecuta el programa, lee estos carácteres, interpreta lo que quieren decir y los ejecuta de manera adecuada. Debido a ésto, los lenguajes interpretados son normalmente los más fáciles de implantar y manejar, ya que el ordenador trabaja directamente con los textos escritos. No obstante, presentan el inconveniente de que, debido a que la ejecución debe ir acompañada de una interpretación, la velocidad de trabajo es relativamente lenta.

En un **lenguaje compilado,** por el contrario, las cadenas de carácteres que constituyen el programa de alto nivel, son traducidas en una primera operación al código máquina de la CPU utilizada. Esto da lugar a una elevada velocidad de ejecución, pero debido a la primera fase de traducción (compilación) su manejo es un poco más complejo.

Seguidamente se expone un ejemplo de un programa implementado en lenguaje ensamblador y en Basic. Consiste en un contador hasta 10 que sigue el diagrama de flujo de la figura 5.1. Obsérvese que, aunque el programa en lenguaje ensamblador ocupa una instrucción más que en Basic, al traducir las instrucciones escritas en Basic a lenguaje ensamblador, su longitud será mucho mayor.

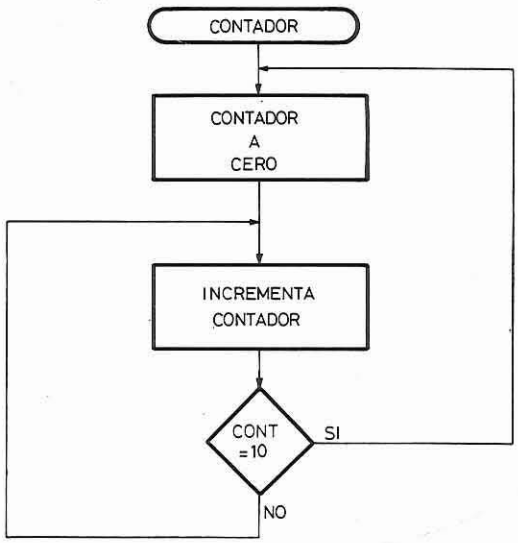


Figura 5.1. Diagrama de flujo de un contador decimal

PROGRAMA EN BASIC DE UN CONTADOR DECIMAL

```
[LINEA]        [INSTRUCCION]          [DESCRIPCION]

   10          LET  COUNT=0            Contador a 0
   20          LET  COUNT=COUNT+1      Incrementa contador
   30          IF COUNT=10 THEN 10     Ir a línea 10 si contador=10
   40          GO TO 20                Ir a línea 20
```

PROGRAMA EN ENSAMBLADOR DE UN CONTADOR DECIMAL

```
[ETIQUETA]      [INSTRUCCION]      [COMENTARIOS]

  START           LDX  #0          ;Pone a cero el registro X
  LOOP            INX              ;Incrementa el registro X
                  CPX  #10         ;Comprueba si X=10
                  BEQ  START       ;Si X=10 salta a START
                  JMP  LOOP        ;Si no salta LOOP
```

5.2 LENGUAJE ENSAMBLADOR

El lenguaje ensamblador no es propiamente un lenguaje de programación, sino una codificación directa de las instrucciones elementales de cada microprocesador. Por tanto, cada microprocesador tiene su propio lenguaje ensamblador definido por el fabricante.

El lenguaje ensamblador es un lenguaje compilado, esto es, se efectúa una traducción del texto a código máquina puro. Para su empleo correcto se requiere, por lo menos, un teclado alfanumérico y una pantalla de datos. Estos elementos no existen en el Microinstructor, no obstante, es una buena técnica de trabajo el escribir siempre los programas en Ensamblador y luego efectuar "a mano" la traducción del programa a código máquina; con ello se consigue que los programas estén bien documentados para futuras referencias a los mismos.

Este método es adecuado durante la fase de aprendizaje, ya que fuerza al alumno a trabajar directamente con el microprocesador, pero es totalmente inadecuado cuando de trata de producir programas de un tamaño o complejidad elevados.

No obstante, estos programas más complejos y largos pueden ser desarrollados mediante un **Sistema de Desarrollo** y una vez depurados, puede transmitirse el código máquina del programa ensamblado al Microinstructor a través del interface RS-232 disponible conectando el Microinstructor con el módulo MM-608 ó MM-613.

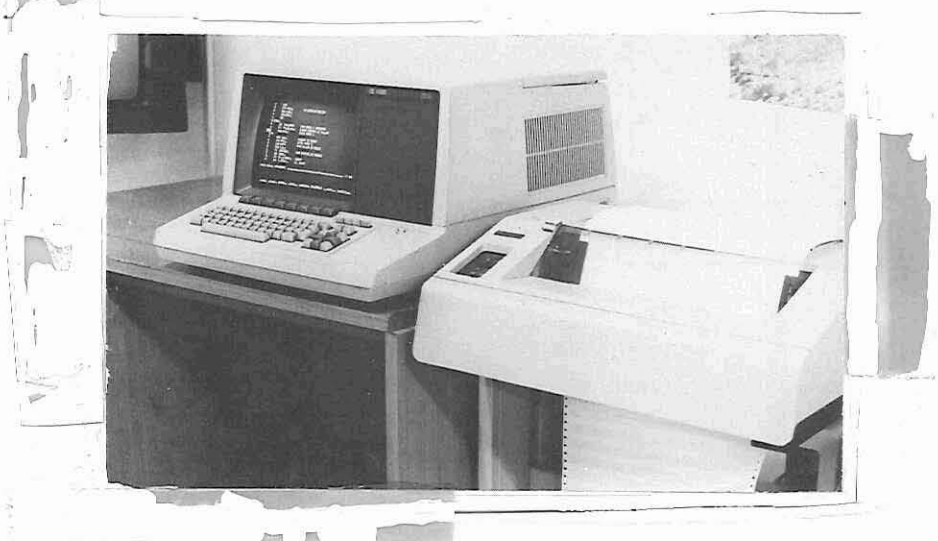


Figura 5.2. Sistema de Desarrollo

Un sistema de desarrollo es básicamente un microcomputador con pantalla y teclado alfanuméricos, con posibilidad de

almacenamiento en disco y dotado de una circuitería especial que le permite "emular" (sustituir) el funcionamiento del microprocesador. A este conjunto de elementos hardware se le dota de una serie de programas específicos para editar y ensamblar programas, programar memorias EPROM o ejecutar los programas desarrollados, de forma controlada. Existe una sofisticada pieza de auxilio denominada **Emulador,** que se conecta a la placa del circuito bajo diseño en lugar del microprocesador, probando el software en condiciones reales de utilización.

De las diversas herramientas de diseño de software de que se dispone, sin duda la más importante para nosotros es el programa ensamblador. El programa ensamblador, como hemos dicho anteriormente, realiza la conversión del código nemónico al código máquina.

Aunque este programa no es accesible para el usuario del Microinstructor, es interesante conocer sus posibilidades y modo de empleo para que, aunque el ensamblado se efectúe a mano, los programas se escriban en lenguaje ensamblador, lo cual crea un hábito de trabajo que facilitará el empleo posterior de un equipo de desarrollo.

Básicamente, un programa escrito en ensamblador consiste en una serie de líneas, cada una de ellas representando una instrucción de programa, y que tienen los siguientes campos:

```
[ETIQUETA]   [CODIGO] [OPERANDO]   [COMENTARIO]

    .          .          .            .
    .          .          .            .
  DELAY        LDA       #02H       ;Tiempo Retardo
    .          .          .            .
    .          .          .            .
```

- **Etiqueta:** mediante las etiquetas se puede asignar a una determinada dirección del programa, a un dato numérico, etc, un nombre más descriptivo y fácilmente utilizable, de modo que al referirnos a dichas posiciones o datos, no sea necesario el conocimiento de su valor.

- **Código:** el campo de código puede estar compuesto por instrucciones o por pseudoinstrucciones.

  Las **instrucciones** son las propias del microprocesador utilizado.

  Las **pseudoinstrucciones o directivas** son códigos nemónicos que

sirven para controlar la ejecución del propio programa ensamblador, pero que no generan código. Algunas de las directivas que están presentes en todos los ensambladores son las siguientes:

1. Directiva Origen: Se coloca al principio del programa y sirve para definir la posición de memoria en donde se debe depositar el primer byte del código máquina que se genera.

2. Directiva Igualdad: Sirve para establecer la identidad entre un símbolo y un valor numérico. Por ejemplo:

```
DOS EQU 2
  ó
DOS = 2 (según el programa ensamblador utilizado)
```

   establece que el símbolo DOS equivale al valor numérico 2 en todos los puntos en que se encuentre.

3. Directiva Reserva: Se utiliza para reservar un cierto número de bytes de memoria en donde posteriormente se van a escribir datos.

4. Directiva Valor: Reserva un cierto número de bytes, pero al mismo tiempo escribe en ellos los datos que se indican en el operando.

5. Directiva Texto: Reserva un cierto número de bytes, escribiendo en ellos los códigos ASCII de los carácteres que componen el texto que se indica en el operando.

- **Operando:** los operandos pueden indicar el modo de direccionamiento, datos numéricos o etiquetas. Puede ser una expresión compleja en la que se incluyan símbolos, pudiendo incluso estar unidos varios factores mediante los sígnos '+' y '-'.

- **Comentarios:** Este campo es opcional y es totalmente ignorado por el ensamblador, sólo sirve para explicar el funcionamiento del programa. Esta explicación puede ser útil, tanto para las personas que tengan que interpretar el programa como para el propio programador.

Estos campos constituyen la entrada del programa ensamblador que se denomina **Programa Fuente.**

Una vez ha sido **ensamblado** el programa, o sea, una vez se ha ejecutado el programa ensamblador sobre el programa fuente, se genera el **Programa Objeto,** que es el código máquina

correspondiente al programa ensamblado y que es directamente ejecutable por el microprocesador.

La mayor parte de ensambladores tienen la posibilidad de generar, además, un fichero listable que consta de los siguientes campos:

```
[DIRECCION]        [LINEA]        [NEMONICO]      [COMENTARIO]
    .     [CODIGO]     .  [ETIQUETA]   .  [OPERANDO]    .
    .        .         .       .       .       .        .
    .        .         .       .       .       .        .
  041DH     A902       12    DELAY     LDA     #02H   ;Retardo
  041FH      .         .       .       .       .        .
    .        .         .       .       .       .        .
```

Un programa ensamblador normalmente efectúa dos pasadas a través del programa fuente. En la primera, crea la denominada tabla de etiquetas en RAM, en la que hay una entrada por cada etiqueta existente en el programa y en la que se incluye el nombre y la dirección de la misma. En la segunda pasada se genera el código objeto, colocándolo en memoria y empleando en todas las referencias a etiquetas que aparezcan en el operando, los valores a ellas asociados en la tabla generada en la primera pasada.

Si el ensamblador realizara un sólo paso surgirían errores debidos a que el programa no puede efectuar saltos a etiquetas que todavía no han sido definidas.

5.3 METODOLOGIA EN EL DISEÑO DEL SOFTWARE

Cuando se pretende el diseño de un programa de cierta complejidad se debe seguir un orden en el proceso de diseño, que dependerá del tipo de programa, de la capacidad del programador, de su experiencia, etc. En general, se sugiere la siguiente secuencia:

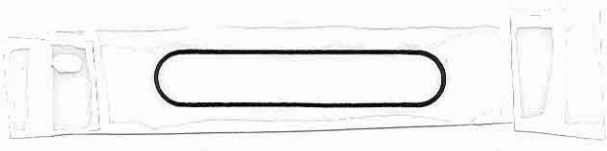
a. **Definición del problema.** Antes de iniciar el diseño de un programa debe entenderse perfectamente el problema a resolver, incluso llegando a crear una lista de especificaciones a cumplir por el mismo, elementos de entrada y de salida requeridos, hardware de que se dispone, etc.

b. **Diseño de la solución.** Este es quizás el paso más importante. Se deben estudiar detenidamente el programa y las especificaciones citadas anteriormente para diseñar la manera óptima de ejecutarlo, pero trabajando en términos funcionales, abstractos. En este paso se decide el empleo o no de interrupciones y la utilización de subrutinas, la manera óptima de resolver el problema, consideraciones sobre la velocidad de ejecución en algunas partes del mismo, la ubicación en memoria de tablas de datos si son necesarias, etc.

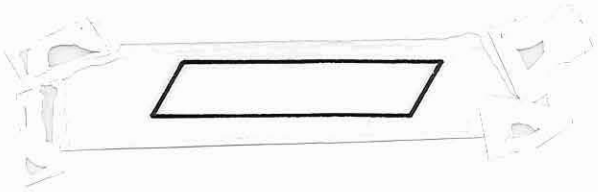
   Al final de esta fase el problema global debe quedar subdividido en una serie de subrutinas y tareas individuales que, aunque han de estar interconectadas entre sí al final del desarrollo completo, sean fácilmente escritas y probadas como bloques independientes.

c. **Realización de los diagramas de flujo de los programas.** Esta fase puede simplificar mucho la labor de las siguientes. Aquí, trabajando por bloques independientes, se efectúa una descripción lo más detallada posible de las diversas operaciones que debe realizar el programa, pero empleando un método simbólico estándar. Esta fase puede ser interesante repetirla varias veces obteniendo unos diagramas de flujo cada vez más detallados, ya que esto nos permite optimizarlo y conseguir un diagrama de flujo que será de gran utilidad en el momento de la prueba y depuración de los distintos bloques. Generalmente se utilizan los siguientes símbolos gráficos:

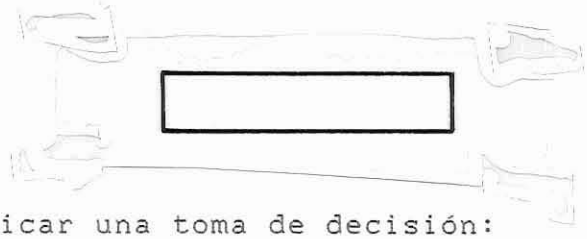
   Para señalar el inicio y el final del programa:



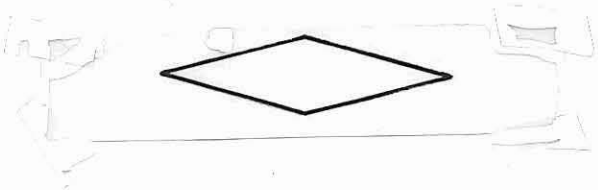
Para indicar una transferencia de datos a través de alguno de los elementos de entrada/salida:



Para indicar la realización de un proceso concreto:



Para indicar una toma de decisión:



d. **Escritura de los programas.** Este paso, si el anterior ha sido correcto y exaustivamente realizado, consiste simplemente en una traducción de los diagramas de flujo de los diferentes bloques, en una secuencia de instrucciones equivalentes. En caso de disponer de un ensamblador, en esta fase se define totalmente el código fuente.

e. **Compilación de los programas.** Si se dispone de un ensamblador este proceso es automático. En este caso el primer paso consiste en la escritura del código fuente mediante el editor y su traducción a código objeto por el ensamblador, con posible corrección tanto de errores mecanográficos como de escritura del propio programa (etiquetas no definidas, ramificaciones fuera de margen, etc), repitiendo si es preciso el paso anterior, al menos parcialmente. Al final de este proceso el programador puede disponer de un código del programa apto para obtener un listado mediante la impresora y del código objeto situado en memoria RAM.

Si para este paso no se dispone de un ensamblador, estas operaciones se deberán efectuar a mano, empleando en todo caso algún método auxiliar de cálculo (por ejemplo la función CALC del Microinstructor), introduciendo asímismo el programa objeto en memoria.

f. **Prueba y depuración de los bloques.** Trabajando a este nivel, y auxiliado por los listados y diagramas de flujo de los diferentes bloques del programa, se verificará su correcto funcionamiento, repitiendo si es preciso las fases de escritura y compilación, hasta que cada uno de los bloques esté funcionando independiente y correctamente. Se deberá prestar especial atención a los valores y datos que se han de pasar a la memoria, al inicio del bloque y que habrán sido inicializados por otros cuando el programa funcione en su totalidad.

g. **Prueba y depuración del programa.** En este punto se conectarán entre sí los diferentes bloques del programa y se comprobará el funcionamiento global del mismo, repitiendo si es preciso, los pasos anteriores. El programa se deberá probar bajo todas las condiciones posibles, de manera que sean ejecutadas incluso aquellas partes del mismo que sólo se ejecutan en contadas ocasiones. Por ejemplo, si se incluyen unas subrutinas de detección de error, deberán provocarse todos y cada uno de los errores posibles para ver su correcto funcionamiento. También deberán tratarse condiciones extremas de funcionamiento, o sea, probar el funcionamiento del programa, no sólo en condiciones normales, sino bajo cualquier condición extrema de datos de entrada.

   Sólo en este momento puede considerarse el programa correctamente finalizado.

5.4 DISEÑO DE UN PROGRAMA

En esta sección se describe el proceso de diseño de un secuenciador luminoso.

Siguiendo la metología de diseño descrita en el apartado anterior, el primer paso será el de definición del problema.

Vamos a considerar que se quiere encender un panel de 8 bombillas colocadas en línea (por ejemplo para decoración de un escaparate) y que su encendido tiene que producirse siguiendo una rotación de abajo hacia arriba ininterrumpidamente de forma secuencial, con tiempos de encendido diferentes según la secuencia.

La secuencia del controlador luminoso es la siguiente:

1. Inicio. Todas las bombillas apagadas.
2. Tiempo bombillas apagadas.
3. Encendido bombilla 1.
4. Tiempo bombilla encendida.
5. Apagado de la bombilla 1 y encendido de la bombilla 2.
6. Tiempo bombilla encendida.
7. Apagado de la bombilla 2 y encendido de la bombilla 3.
8. Tiempo bombilla encendida.
9. Apagado de la bombilla 3 y encendido de la bombilla 4.
10. Tiempo bombilla encendida.
11. Apagado de la bombilla 4 y encendido de la bombilla 5.
12. Tiempo bombilla encendida.
13. Apagado de la bombilla 5 y encendido de la bombilla 6.
14. Tiempo bombilla encendida.
15. Apagado de la bombilla 6 y encendido de la bombilla 7.
16. Tiempo bombilla encendida.
17. Apagado de la bombilla 7 y encendido de la bombilla 8.
18. Tiempo bombilla encendida.
19. Todas las bombillas encendidas.
20. Tiempo bombillas encendidas.
21. Repite el proceso. Salta al punto 1.

El siguiente punto a seguir será el de diseñar una posible solución. Para ello podríamos realizar el programa siguiendo esta misma secuencia, pero no es esta la mejor solución. Si estudiamos la secuencia con detenimiento, vemos que el programa se compone básicamente, de dos funciones repetitivas (apagado y encendido y tiempo de apagado o tiempo de encendido).

Para probar el programa podemos utilizar el bloque de 8 LEDs del Microinstructor, programando la secuencia de operaciones por

medio de los registros de la VIA 65C22. Para ello será necesario programar el port B como salidas poniendo el registro DDRB (0A402H) a 0FFH. En este momento dicho port presentará en sus líneas y por tanto en los LEDs a ellas asociados, el estado del registro RB (0A400H).

Para calcular los tiempos de espera en un estado determinado se utilizará uno de los dos contadores de que dispone el 65C22 de aplicaciones. Se puede, por ejemplo, utilizar el segundo contador y cargar en él un valor proporcional al tiempo de espera que se quiera lograr. Este valor se introduce en TIM2L (0A408H) y TIM2H (0A409H). A partir del instante de carga el contador empezará a descontar. Cuando llegue a valer 0 el bit 6 del registro IFR (0A40DH) pasará a '1'.

Una vez pensada una solución del problema, se puede pasar a dibujar el diagrama de flujo. El diagrama de flujo que sigue una posible solución es el siguiente:

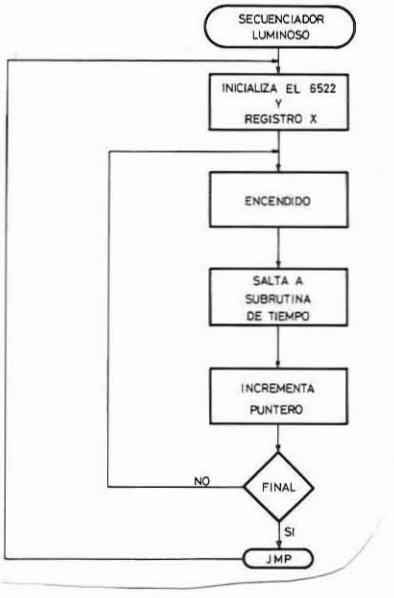


Figura 5.3. Diagrama de flujo programa principal

Este programa inicializa los registros del 65C22 y, a continuación, pasa a un bucle sin fin de presentación de datos.

Se utilizan dos tablas en las que se programan las secuencias de encendido de las bombillas y los tiempos entre cada una de las secuencias programadas. de cada bombilla (TIMTB). El registro X actúa como puntero a estas dos tablas.

En cada pasada por el bucle del programa se saca en primer lugar la configuración de bits de salida (luces encendidas) correspondientes por el port B de la VIA de aplicaciones del Microinstructor y, a continuación, se efectúa un retardo indicado por el valor asociado en la tabla "TIMTB". Por último, se incrementa el puntero de las tablas (registro X) y se comprueba si ya ha recorrido toda la tabla. Si no es así, se saca la siguiente configuración de salida y se repite la secuencia. Cuando ya ha recorrido toda la tabla el salto se efectúa a "INICIO", que reinicia el puntero y repite toda la tabla.

El tiempo de encendido y apagado de los LEDs se ha implementado en forma de subrutina (figura 5.4).

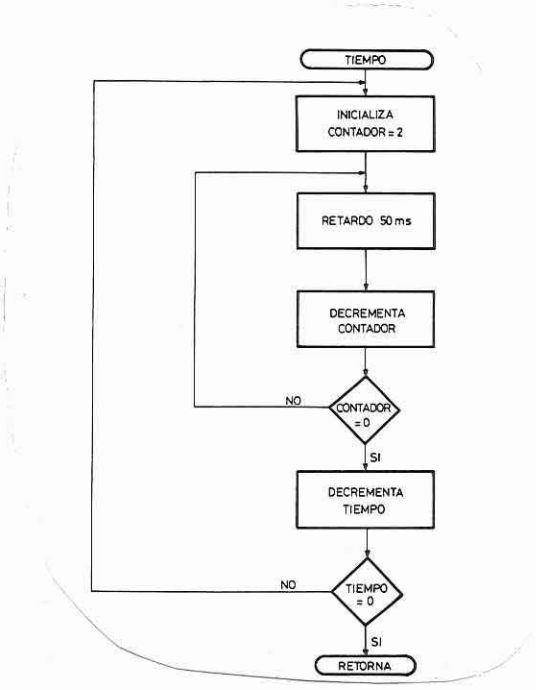


Figura 5.4. Diagrama de flujo de la subrutina

Al estar como un solo bloque en el programa principal, la razón por la que se ha separado este bloque de programa como subrutina no es para ahorrar memoria si no para que pueda ser utilizado en otro programa sin más que copiarlo.

Esta subrutina consta de tres bucles que, segun el valor del Acumulador en el momento de entrada, introducen un retardo comprendido entre 0.1 y 25.6 segundos.

Una vez concretado el funcionamiento y fijados los diagramas de flujo del programa, se puede proceder a escribirlo. Primero escribiremos el programa en lenguaje ensamblador, empleando los

nemónicos del 65C02. El resultado puede ser el siguiente:

```
INICIO    LDA   #0FFH
          STA   DDRB       ;Port B salidas
PUNT      LDX   #0         ;Inicializa puntero
BUCLE     LDA   LEDTB,X
          STA   RB         ;Activa salidas
          LDA   TIMTB,X
          JSR   RETAR      ;Y espera
          INX
          CPX   #0AH       ;Toda la tabla?
          BNE   BUCLE      ;No, repite
          BEQ   PUNT
```

Despues de programar el port B como salida se inicializa el puntero X que se utiliza para direccionar la tabla LEDTB y colocar el dato apuntado en RB. Cada nueva pasada por el bucle incrementará el valor de X, con lo que se recogerá el siguiente valor de la tabla LEDTB.

Para direccionar TIMTB se utiliza el mismo procedimiento, siendo el valor recogido de la tabla el que indica el retardo que introducirá esta subrutina. Una vez se ha regresado al programa principal, se incrementa el puntero y se repite de nuevo el bucle si aún no se han producido 10 incrementos y por tanto todavía no ha sido explorada toda la tabla; en caso contrario pasa a inicio PUNT, con lo cual se habrá finalizado la secuencia y se inicializa de nuevo el puntero.

La subrutina de control de tiempo es la siguiente:

```
RETAR     STA   0          ;Guarda el valor total
          STX   1          ;Guarda Regitro X
RET1      LDX   #2         ;2 veces (100 ms)
RET2      LDA   #50H
          STA   TIM2L      ;Inicia tiempo
          LDA   #C3H       ;A 50 ms
          STA   TIM2H      ;Inicia cuenta
RET3      LDA   #20H
          AND   IFR        ;Espera a fin de tiempo
          BEQ   RET3
          DEX
          BNE   RET2       ;No, repite
          DEC   0          ;Multiplicador
          BNE   RET1
          LDX   1          ;Recupera X
          RTS
```

Al iniciar la subrutina se guarda el dato de TIMTB,X en la posición 0 de la página 0 de memoria y el valor del registro X en

la posición 1. Seguidamente, se coloca en el contador interno del 65C22 el valor C350H (50000 us), iniciando la cuenta hasta que el valor del registro IFR del contador del 65C22 es igual a 20H (el tiempo empleado es de 50 ms). A continuación se decrementa el registro X y se inicia una segunda cuenta, lo que dará un tiempo total de 100 ms. Finalizada ésta, se decrementa el multiplicador y se inicia de nuevo el contador, repitiéndose esta secuencia varias veces hasta que el dato de la memoria sea 0, regresando al programa principal con el valor del puntero en el registro X.

Sólo queda ahora introducir el programa en el Microinstructor y depurarlo en caso de que no siga el funcionamiento previsto.

PROGRAMA PRINCIPAL

DIRECCION	CONTENIDO	ETIQUETA	INSTRUCCION		COMENTARIO
400	A9	INICIO	LDA	#0FFH	
401	FF				
402	8D		STA	DDRB	;PB a salidas
403	02				
404	A4				
405	A2	PUNT	LDX	#0	;Reentrada
406	00				
407	BD	BUCLE	LDA	LEDTB,X	
408	3B				
409	04				
40A	8D		STA	PB	;Activa salidas
40B	00				
40C	A4				
40D	BD		LDA	TIMTB,X	
40E	45				
40F	04				
410	20		JSR	RETAR	;Espera
411	1A				
412	04				
413	E8		INX		
414	E0		CPX	#0AH	;Toda la tabla?
415	0A				
416	D0		BNE	BUCLE	;NO, repite
417	EF				
418	F0		BEQ	PUNT	;SI, reinicia
419	EB				

RUTINA DE TIEMPO

```
41A     85      RETAR   STA  0          ;Guarda A
41B     00
41C     86              STX  1          ;Guarda X
41D     01
41E     A2      RET1    LDX  #2         ;100 ms
41F     02
420     A9      RET2    LDA  #50H
421     50
422     8D              STA  TIM2L      ;Inicia tiempo
423     08
424     A4
425     A9              LDA  #0C3H      ;50 ms
426     C3
427     8D              STA  TIM2H      ;Inicia cuenta
428     09
429     A4
42A     A9      RET3    LDA  #20H
42B     20
42C     2D              AND  IFR        ;Espera a fin tiempo
42D     0D
42E     A4
42F     F0              BEQ  RET3
430     F9
431     CA              DEX
432     D0              BNE  RET2       ;NO, repite
433     EC
434     C6              DEC  0          ;Multiplicador
435     00
436     D0              BNE  RET1
437     E6
438     A6              LDX  1          ;Recupera X
439     01
43A     60              RTS
```

En la tabla LEDTB se encuentran los datos correspondientes a los LEDs que se deberán encender en cada pasada.

```
43B         00
43C         01
43D         02
43E         04
43F         08
440         10
441         20
442         40
443         80
444         FF
```

En la tabla TIMTB se encuentran los datos correspondientes a los tiempos de espera a programar para cada uno de los estados distintos de los LEDs.

```
445         0A
446         08
447         07
448         06
449         05
44A         04
44B         03
44C         02
44D         01
44E         05
```

Si se desea cambiar la secuencia y los tiempos de encendido de los LEDs se pueden cambiar los valores de estas dos tablas.

En caso de ser necesaria una depuración del programa, puede hacerse mediante la función GO en modo STEP, como se ha explicado en el capítulo 4.

5.5 TABLAS DE CONVERSION. ALGORITMOS MATEMATICOS

Un problema que se presenta normalmente en programación es la conversión de una representación numérica o código a otro. Por ejemplo, consideremos el problema de representar un dígito hexadecimal en un indicador de 7 segmentos. De alguna forma, se debe determinar qué segmentos se deben encender para representar el carácter apropiado. Para ello se necesita una conversión de código binario a código de 7 segmentos. Esto se hace utilizando la técnica llamada **tabla de conversión.** En una zona de la memoria llamada tabla, se almacena la pauta de segmentos para el carácter '0' en la primera posición, la correspondiente al carácter '1' en la siguiente y así sucesivamente. Para traducir un código binario al correspondiente código de 7 segmentos, se mira simplemente el código de la tabla. Para hacer esto, el número binario que se va a convertir, se suma a la dirección de la primera entrada en la tabla. El resultado es la dirección de la entrada que contiene el código de 7 segmentos del dígito deseado. Luego se lee el contenido de la posición direccionada, completándose así la conversión. Es de gran utilidad para esta aplicación, el direccionamiento indexado.

Como hemos visto, las posibilidades de realizar funciones matemáticas en el 65C02 y en la mayoría de los microprocesadores de 8 bits, se limitan a las operaciones de suma y resta. Entonces, ¿Cómo puede una calculadora que utilice este microprocesador realizar las operaciones de división, multiplicación, funciones trigonométricas, etc...? La respuesta está en que todas estas funciones matemáticas se pueden realizar utilizando sólo instrucciones de suma y resta.

La multiplicación se puede realizar muy sencillamente mediante una serie de sumas y desplazamientos. Para ver el procedimiento, consideremos que estamos multiplicando a mano: el multiplicando se multiplica cada vez por un dígito del multiplicador y el resultado de cada dígito se desplaza un lugar a la izquierda y se suman todos los resultados. Se puede utilizar la misma técnica para los números binarios. La multiplicación de dos números binarios es trivial, ya que un número multiplicado por '1' es el número original y multiplicado por '0' es '0'. Por tanto, cualquier multiplicación se puede realizar utilizando sólo el desplazamiento y la suma.

La división se realiza con un procedimiento parecido, usando el desplazamiento y la resta. A las técnicas que realizan una operación dada se las llama **algoritmos.**

El cálculo de una función como el seno de un ángulo es más complicado, pero hay algoritmos que se aproximan con mucha

exactitud utilizando sólo combinaciones de funciones más elementales. El cálculo del seno se puede hacer evaluando un número de términos de su desarrollo en serie, según la aproximación que deseemos:

$$\text{SEN } X = X - \frac{X^3}{3 \times 2} + \frac{X^5}{5 \times 4 \times 3 \times 2} - \frac{X^7}{7 \times 6 \times 5 \times 4 \times 3 \times 2} + \ldots\ldots\ldots\ldots$$

Para cada función existe un algoritmo que permite su evaluación aproximada, utilizando tan sólo funciones elementales que puede realizar el microprocesador.

CAPITULO 6

DESCRIPCION DEL SOFTWARE DEL MICROINSTRUCTOR

6.1 EL PROGRAMA MONITOR

Es el programa que se encarga del control del microcomputador desde el momento de su conexión, y está siendo ejecutado permanentemente, hasta que se transfiere el control a un programa de usuario.

En el MI-650C este programa ocupa algo más de 2 Kbytes y se encuentra entre las direcciones F7D0H y FFFFH. A partir de la posición F100H se encuentra una pequeña zona de subrutinas, para adaptar lel programa monitor a la nueva versión del Microinstructor.

El programa monitor está compuesto por una serie de rutinas encargadas de:

- Analizar el teclado detectando las teclas pulsadas.
- Representar la información adecuada en el display de 7 segmentos.
- Analizar y ejecutar las diversas órdenes de control introducidas a través del teclado.
- Atender las interrupciones.

Como es común en este tipo de programas, existen una serie de subrutinas, especialmente en las secciones de control del teclado y del display, que se han diseñado de manera que puedan ser fácilmente empleadas en programas de aplicación.

Existen tres bloques básicos de programa. El primer bloque es el que controla la inicialización del Microinstructor y lo deja en un estado de espera de recepción de órdenes. El segundo bloque es el encargado del control de las interrupciones. El último bloque es el que se encarga del control de los elementos de entrada/salida, que incluye la identificación y ejecución de las funciones del teclado.

6.1.1 **Inicialización y bucle de reposo**

En el momento en que se enciende el Microinstructor o que se pulsa la tecla [RESET], el microprocesador recoge automáticamente el vector de reset que se encuentra en las direcciones FFFCH (parte alta) y FFFDH (parte baja) y pasa a ejecutar el bloque de programa apuntado por este vector. Este bloque recibe el nombre de **Entrada de Reset.**

La primera operación efectuada por este bloque consiste en inicializar el Puntero de la Pila ("Stack Pointer"). Seguidamente se borra el "flag" que indica el modo decimal. Aunque esta última operación es reiterativa cuando el microprocesador utilizado es el 65C02 o el 65SC02, tiene gran importancia cuando se utiliza el 6502 ya que, tras la secuencia interna de reset este bit no es inicializado de ninguna forma por esta CPU.

A continuación se inicializan las líneas de entrada/salida de la VIA 65C22 que controla el teclado y el display y, por último, se fijan los diversos valores en RAM que emplea el programa monitor para su funcionamiento.

Tras estas operaciones se efectúa un salto al bloque principal del programa o bucle de reposo (figura 6.1). Este bucle tiene dos puntos básicos de entrada: "MONITR" que es la entrada normal y "ERROR" que da lugar a la salida del texto "Err" y el número binario contenido en el acumulador en el momento de entrada. La entrada "MONITR" efectúa un test de los leds del "Port B" del 65C22 que controla las líneas de entrada/salida de usuario encendiendo consecutivamente todos los LEDs asociados a dicho port. Mientras se está efectuando este proceso en el display aparece el mensaje "650 C". Una vez terminado este test inicial se efectúa una lectura de los microinterruptores asociados al "Port A" del 65C22 de usuario y se encienden los LEDs del "Port B" de dicho circuito que se corresponden con microinterruptores que están en estado ON. Tanto si la entrada se produce a través de "ERROR" como a través de "MONITR", el programa pasa a un bucle de espera de instrucción, en el cual se efectúa un refresco del display y análisis del teclado, hasta detectar la pulsación de una tecla. Si esta tecla corresponde a una función primaria válida (códigos 0 a D), el programa salta a la rutina de ejecución de la misma; si es la tecla [FIN], salta a "MONITR" (efectúa el borrado de un posible texto en display) y si es otra (E, F, + ó -) efectúa un salto indirecto a través del vector situado en "ESPVEC". Este vector, al ser inicializado, apunta a "ERROR", pero si se altera permite asignar funciones específicas a estas teclas, de modo que se amplien las funciones básicas del monitor, empleando el contenido del Acumulador para diferenciar las teclas.

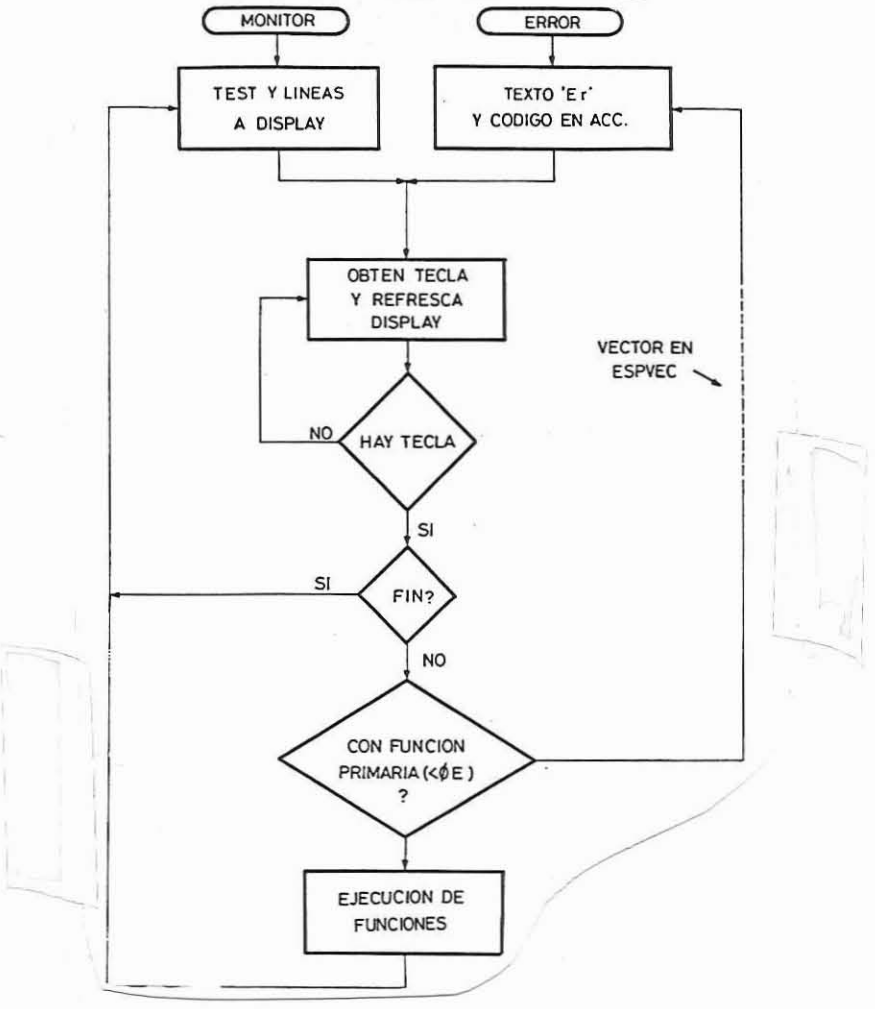


Figure 6.1. Diagrama de flujo del teclado en reposo

Las rutinas correspondientes a la ejecución de las diferentes funciones de control básicas ocupan la mayor parte del programa monitor. Si han sido correctamente completadas, la salida de estas rutinas se efectúa mediante la secuencia:

```
CLC
RTS
```

En caso de detectarse algún tipo de error durante su ejecución, la secuencia pasa a ser:

```
SEC
RTS
```

encontrandose en el Acumulador el código del error que se ha producido.

6.1.2 **Atención a las interrupciones**

Como ya se ha visto en el capítulo 3 el microprocesador 65C02 dispone de dos entradas de interrupción (IRQ y NMI) y una interrupción por software del tipo IRQ, mediante la instrucción BRK. Cada una de ellas recibe un tratamiento distinto, pero las dos son atendidas mediante vectores, de manera que éstos puedan, en cada caso particular, reconfigurarse para cubrir las necesidades del software en el desarrollo de programas de aplicación.

IRQ y BRK: El vector primario de esta interrupción se encuentra en ROM en FFFEH y FFFFH. Este vector apunta a IRQJMP. En esta dirección se encuentra una instrucción de salto indirecto a la dirección de memoria apuntada por IRQVEC y IRQVEC+1. IRQVEC está situado en RAM, y por tanto es alterable.

Tras pulsar RESET, el contenido de IRQVEC se inicializa (en la zona de inicialización del "scratch") y toma el valor IRQBRK. Cuando de produzca una interrupción, bien por una solicitud externa de interrupción IRQ, o bien por la ejecución de una instrucción BRK, se transferirá el control a la rutina IRQBRK. Dado que normalmente la finalidad de estas dos acciones es diferente, la primera operación que es necesario efectuar en la rutina IRQBRK, consiste en diferenciar la causa de la interrupción, para lo que se emplea el flag B del registro de estado, que, como sabemos, se pone a 1 en caso de proceder de una instrucción BRK. Esta distinción da lugar a una bifurcación, implantada también en forma vectorizada a través de IRQUS (activada por IRQ) y BRKUS (activada por BRK).

Estos dos vectores, ubicados también en RAM, apuntan respectivamente a las subrutinas IRQSIS y BRKSIS. En éstas se guardan los registros internos de la CPU en los registros imagen, se presenta en display la dirección de programa en curso cuando se provocó la interrupción y el texto "ie" o "be", según la fuente de interrupción sea IRQ o BRK, y se devuelve el control al monitor, con lo que finaliza la atención.

En resumen, en el proceso de atención a este tipo de interrupción intervienen 3 vectores: IRQVEC, IRQUS y BRKUS, situados todos en RAM, por lo que son alterables. Están

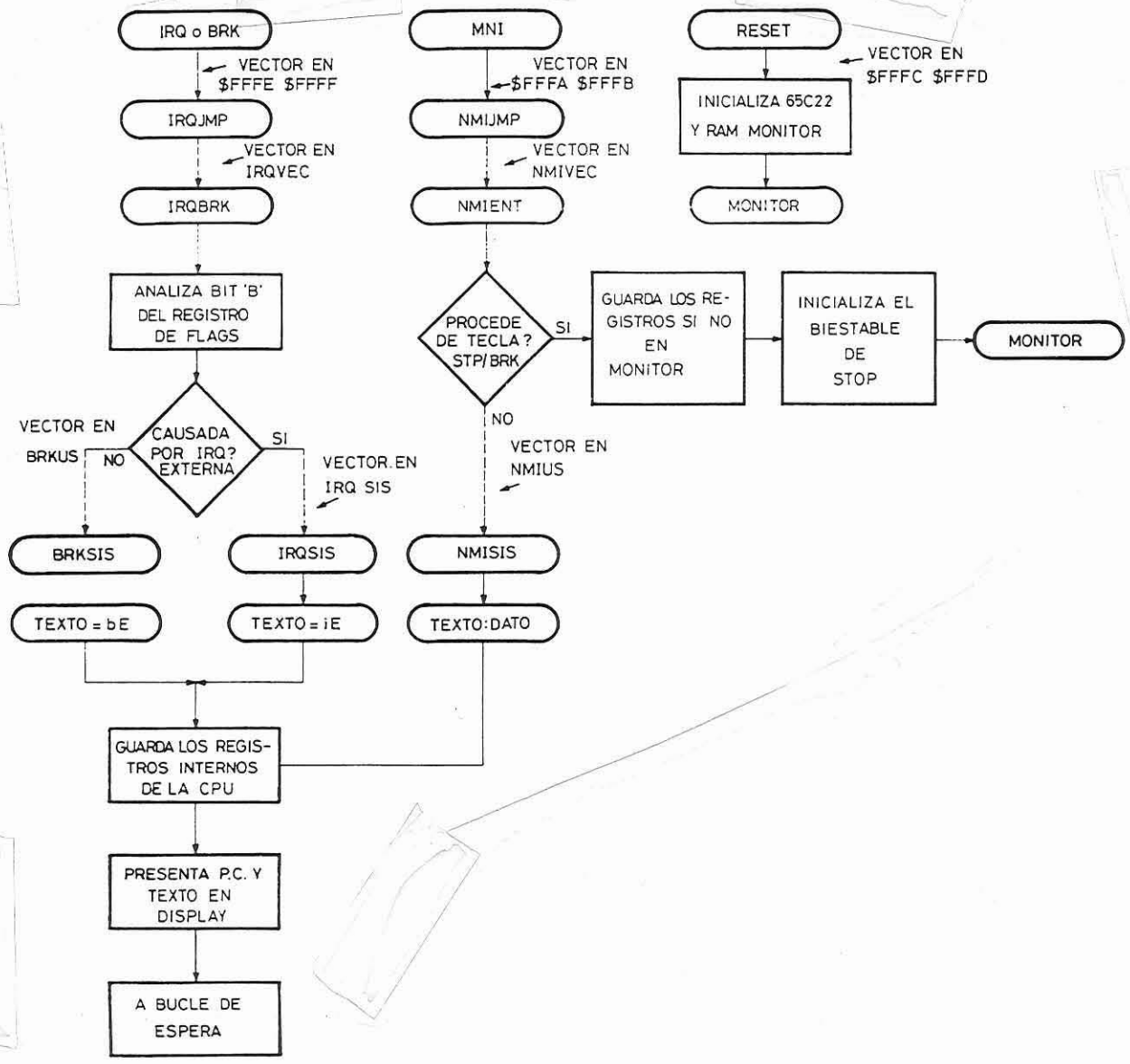


Figure 6.2. Interrupciones del Microinstructor

encadenados a través del programa para provocar al final del proceso una detención de la ejecución, indicando la causa de la misma. Según la función que se desee ejecutar estos vectores pueden ser alterados:

IRQVEC: En caso de alterar este vector el salto se hará siempre que se produzca una interrupción tanto por hardware (IRQ), como por software (BRK), con lo cual se pierde la capacidad de distinguir entre IRQ y BRK,

a menos que se haga en el programa de usuario.

IRQUS: En este caso se saltará a la dirección apuntada siempre que se produzca una interrupción externa IRQ. Mantiene operativa la instrucción BRK para usarla en la depuración de programas.

BRKUS: La variación de este vector se utiliza para efectuar acciones optativas durante la depuración de un programa, empleando la instrución BRK, o bien para asignar una misión específica a esta instrucción.

NMI: El vector primario de este tipo de interrupción está situado en FFFAH-FFFBH y apunta a NMIJMP. Igual que en el caso de la interrupción enmascarable en esta posición se encuentra una instrucción de salto indirecto a la dirección indicada por NMIVEC y NMIVEC+1. Como en el otro tipo de interrupciones, este vector está situado en RAM.

Tras pulsar RESET, el contenido de NMIVEC se establece según un valor almacenado en ROM (en la zona de "scratch"), apuntando a NMIENT (FEBEH). En esta rutina se efectúa en primer lugar una discriminación de la causa de la interrupción que puede estar provocada por las siguientes causas:

a. La pulación de la tecla STOP/BRK.

b. El circuito empleado para la ejecución paso a paso del programa.

c. Una actuación externa en la línea NMI.

Para conocer la causa de la interrupción se emplea la línea PB7 del 65C22 que controla el teclado y el display (IC15). Al pulsar la tecla de [STOP/BRK], esta línea se pone a '1' al bascular el biestable IC10, permaneciendo a '0' en los otros dos casos.

Si la interrupción se ha producido por la pulsación de la tecla [STOP/BRK] se transfiere el control a la rutina NMISTP, la cual efectúa las acciones siguientes:

- Pone a 0 el biestable STOP, liberando así la línea NMI para otro uso.

- Analizar el valor del contenido del PC en el momento de la entrada. Si este valor está comprendido entre F000H y FFFFH (ejecutando el programa monitor) transfiere el control a MONITOR sin alterar los registros internos. Esto

se emplea para detener la introducción de datos durante el diálogo, en una función de control.

Si, por el contrario, el valor está fuera de estos márgenes se presupone la ejecución de un programa de aplicación, con lo cual se guardan los registros internos en los registros imagen y se devuelve, como en el caso anterior, el control al monitor en MONITR.

Si la interrupción se produce por alguna de las otras dos causas, se efectúa un salto a la posición apuntada por el vector situado en NMIUS en RAM. Tras RESET, NMIUS apunta a la rutina NMISIS que, como en el caso de IRQ guarda los registros de la CPU y devuelve el control al monitor, mostrando la dirección de interrupción. En el Microinstructor la solicitud de interrupción por parte de un elemento externo no se efectúa a través de esta línea, lo cual elimina la ambigOedad que pudiera darse para diferenciar cuál de estas dos causas es la que ha producido la interrupción.

Dado de que la interrupción NMI no puede enmascararse, suele utilizarse en muchos casos, para detectar condiciones que requieran un servicio inmediato por parte de la CPU. Es frecuente el uso de esta interrupción para detectar errores fatales de funcionamiento o fallos de tensión en la entrada de tensión del sistema.

En resumen, en este caso existen solamente dos vectores de atención a la interrupción, ambos en RAM, que pueden ser alterados según la función que se desee ejecutar:

NMIVEC: En este caso, al producirse una interrupción por la línea NMI por cualquiera de las tres causas señaladas anteriormente, la ejecución del programa sigue en la posión apuntada por este vector. Si se desea el empleo de la tecla [STOP/BRK] para simular una interrupción, se deberá incorporar en el programa de usuario las instrucciones que efectúan la puesta a cero del biestable de STOP.

NMIUS: En este caso la tecla [STOP/BRK] permanece activa. Mediante este vector se pueden incluir rutinas auxiliares de depuración del programa mediante la ejecución paso a paso del mismo; por ejemplo, mostrar la dirección de ejecución y el contenido del Acumulador.

6.1.3 Control de entradas/salidas

El display utilizado por el Microinstructor no incluye en su estructura ningún elemento de retención de datos, efectuándose para la presentación de éstos el encendido momentáneo y secuencial de cada uno de los dígitos que lo componen. Para que los datos representados aparezcan estables, se requiere su presentación contínua en el display a un ritmo superior a 50 veces por segundo, o sea, cada 20 milisegundos. La subrutina que incorpora el Microinstructor efectúa la presentación de los datos sobre cada uno de los dígitos del display aproximadamente cada 1.8 milisegundos, tardando unos 10 milisegundos en presentar los datos en los 6 dígitos del display.

Por otra parte, el hardware del teclado sólo permite saber si hay una tecla pulsada en un momento dado. Normalmente al ser pulsada una tecla se producen una serie de rebotes del contacto, cuya duración depende de la configuración física de la tecla y que deben ser ignorados o eliminados para evitar errores en la toma de datos. El método más utilizado para evitar este problema consiste en que una vez detectada la pulsación de una tecla se espere un tiempo superior a la duración de este tren de rebotes, se lea de nuevo el teclado y se verifique que sigue pulsada esa tecla. Este es el método seguido en el Microinstructor, que utiliza como tiempo de retardo la duración de la subrutina de presentación de datos en el display.

Otro elemento a tener en cuenta durante el análisis de un teclado es la posibilidad de pulsación de dos o más teclas simultáneamente, lo cual se elimina con el procedimiento de esperar, una vez detectada una pulsación, a que el teclado pase a reposo (todas las teclas soltadas), antes de procesar otra pulsación.

En el siguiente apartado se analizan con detalle las subrutinas que componen este bloque.

Una parte importante de las subrutinas de E/S, la constituyen los programas de ejecución de las funciones del teclado. Cuando se ha detectado la pulsación de una tecla, y ésta corresponde a una de las funciones básicas, se salta a "EJEC" donde se inicia una comparación en cadena del valor numérico de la tecla pulsada, hasta detectar la función básica a que corresponde. En este punto, se inicia la ejecución de la subrutina correspondiente. Debe tenerse en cuenta que la ejecución de cualquier función del teclado (incluyendo la transferencia del control a un programa de usuario) se considera como una subrutina del bucle principal de análisis del teclado y refresco del display. Por ello, es posible devolver el control al programa Monitor finalizando un programa con la instrucción RTS (retorno de subrutina).

6.2 RUTINAS DE ENTRADA/SALIDA

Como se ha dicho anteriormente, estas rutinas son las encargadas de controlar los procesos de entrada/salida del Microinstructor, que son utilizadas por el programa monitor y que tambien pueden ser utilizadas por programas de usuario.

REFDIS (FE4DH): Rutina básica de salida de datos a display. Presenta momentáneamente (10 milisegundos) sobre el display el contenido del buffer DISBUF que se encuentra en la memoria RAM (0353H). Este buffer está compuesto por 6 bytes que controlan cada uno de ellos el encendido de un dígito, correspondiendo DISBUF al dígito izquierdo y DISBUF+5 al derecho.

Cada bit de cada una de estas posiciones controla el encendido de uno de los segmentos del dígito que le corresponde, de manera que un '1' provoca el encendido de dicho segmento. La correspondencia entre el bit de una posición de DISBUF y el segmento del dígito asociado es la siguiente:

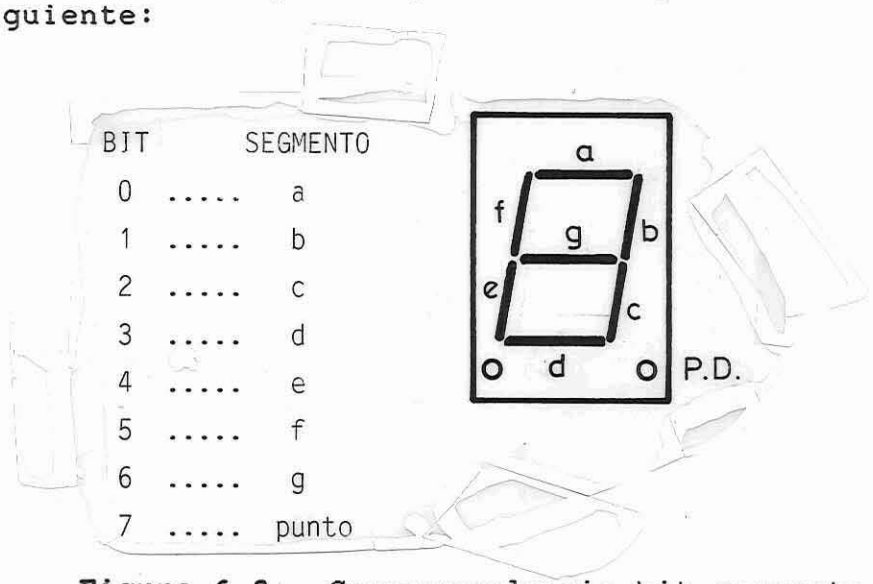


Figura 6.3. Correspondencia bit-segmento

Para ilustrar mejor el funcionamiento de esta subrutina puede estudiarse el siguiente programa en el que se efectúa una escritura sobre los seis dígitos de DISBUF en el código de segmentos equivalentes al valor que se desea representar y luego se pasa a un bucle en el que se alternan un estado de refresco del display y otro de no refresco. Durante el estado de refresco del display se ejecuta la subrutina REFDIS 96 veces.

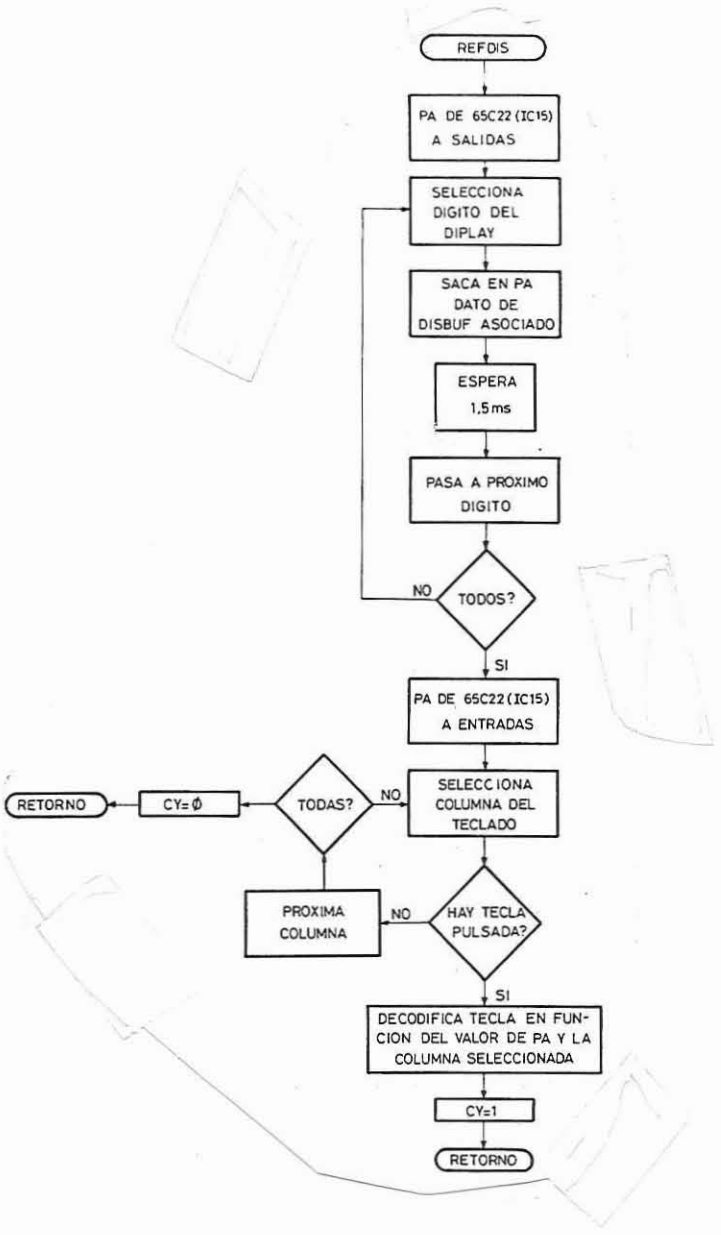


Figura 6.4. Diagrama de flujo de REFDIS

```
                    1 "650X"
          < 0001 >  2 CONTADOR        EQU     1
          < 0353 >  3 DISBUF          EQU     0353H
          < FE4D >  4 REFDIS          EQU     0FE4DH
                    5
                    6                 ORG     400H
                    7
0400                8 INICIO
0400 A906           9                 LDA     #06H          ;1
```

```
0402 8D0353        10              STA     DISBUF
0405 A95B          11              LDA     #5BH        ;2
0407 8D0354        12              STA     DISBUF+1
040A A94F          13              LDA     #4FH        ;3
040C 8D0355        14              STA     DISBUF+2
040F A966          15              LDA     #66H        ;4
0411 8D0356        16              STA     DISBUF+3
0414 A96D          17              LDA     #6DH        ;5
0416 8D0357        18              STA     DISBUF+4
0419 A97D          19              LDA     #7DH        ;6
041B 8D0358        20              STA     DISBUF+5
041E               21 ON
041E A960          22              LDA     #60H
0420 8501          23              STA     CONTADOR
0422               24 ON1
0422 20FE4D        25              JSR     REFDIS
0425 C601          26              DEC     CONTADOR
0427 A501          27              LDA     CONTADOR
0429 C900          28              CMP     #0
042B D0F5          29              BNE     ON1
042D               30 OFF
042D A2FF          31              LDX     #0FFH
042F               32 DELAY
042F A0FF          33              LDY     #0FFH
0431               34 DELAY1
0431 88            35              DEY
0432 EA            36              NOP
0433 EA            37              NOP
0434 EA            38              NOP
0435 D0FA          39              BNE     DELAY1
0437 CA            40              DEX
0438 D0F5          41              BNE     DELAY
043A 4C041E        42              JMP     ON
```

SCAN (FE7DH): Esta es la subrutina básica de análisis de teclado. Efectúa una inspección del teclado, saliendo con el flag de carry del registro S igual a '0' si no hay tecla pulsada y con dicho flag igual a '1' si hay alguna tecla pulsada. En este último caso, en el Acumulador se incluye el número de identificación de la tecla pulsada, con el bit 7 = '1'.

Esta subrutina no efectúa una protección anti-rebote del teclado y se utiliza únicamente para analizar su estado en tiempo real (para saber en todo momento si hay o no tecla pulsada).

GETKEY (FE09H): Esta subrutina se encarga de controlar adecuadamente las dos subrutinas anteriores para efectuar las siguientes operaciones:

- Refresco del display mediante REFDIS.
- Análisis del teclado mediante SCAN.
- Protección anti-rebote de la tecla pulsada.
- Evita la pulsación simultánea de más de una tecla.

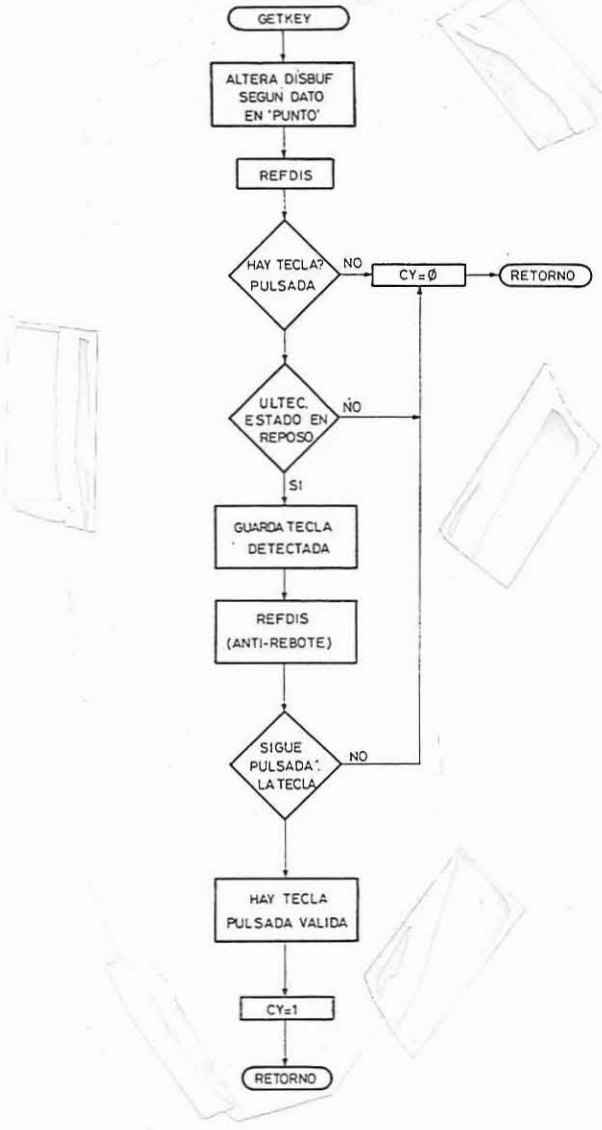


Figura 6.5. Diagrama de flujo de GETKEY

Además de estas funciones, esta subrutina también altera el contenido de DISBUF de acuerdo con el contenido de la variable PUNTO (0361H), poniendo a 1 el bit del punto decimal del byte correspondiente a DISBUF según la tabla:

Bit de PUNTO	Bit 7 del buffer DISBUF alterado
0	DISBUF + 5 (Dígito derecho)
1	DISBUF + 4
2	DISBUF + 3
3	DISBUF + 2
4	DISBUF + 1
5	DISBUF (Dígito izquierdo)
6 y 7	Ignorados

En caso de que la tecla pulsada sea válida, la rutina finaliza con el flag de carry del registro S igual a '1' y el código de la tecla pulsada en el Acumulador. Los códigos para cada tecla son los siguientes:

TECLA PULSADA	VALOR EN ACUMULADOR
[0/MEN]	0
[1/GO]	1
[2/REG]	2
[3/STV]	3
[4/SAVE]	4
[5/LOAD]	5
[6/VER]	6
[7/MOT]	7
[8/WRP]	8
[9/CLRB]	9
[A/BR?]	AH
[B/SET B]	BH
[C/STEP]	CH
[D/CALC]	DH
[E]	EH
[F]	FH
[+]	10H
[-]	11H
[FIN]	12H

Si no hay tecla válida pulsada la salida se produce con el flag de carry='0' y un valor indeterminado en el Acumulador.

El siguiente ejemplo se ha desarrollado para comprender mejor el funcionamiento de esta subrutina. Inicialmente se dejan todas las posiciones del DISBUF a '0', excepto los dos puntos digitales de las posiciones correspondientes al dato, que se ponen a '1'. Despues se entra en un estado de espera, hasta que se pulsa alguna tecla. Si la tecla pulsada es [0], sale el mensaje "SI 0" en el display; y si la tecla pulsada es distinta de [0], sale el mensaje "no 0".

Programa ejemplo de GETKEY:

```
                        1 "650X"
          < 0353 >      2 DISBUF          EQU     0353H
          < 0361 >      3 PUNTO           EQU     0361H
          < FE09 >      4 GETKEY          EQU     0FE09H
                        5
                        6                 ORG     400H
                        7
0400 A900               8                 LDA     #0
0402 8D0353             9                 STA     DISBUF
0405 8D0354            10                 STA     DISBUF+1
0408 8D0355            11                 STA     DISBUF+2
040B 8D0356            12                 STA     DISBUF+3
040E 8D0357            13                 STA     DISBUF+4
0411 8D0358            14                 STA     DISBUF+5
0414 A903              15                 LDA     #3
0416 8D0361            16                 STA     PUNTO
0419 18                17                 CLC
041A                   18 REDITE
041A 20FE09            19                 JSR     GETKEY
041D 90FB              20                 BCC     REDITE
041F C900              21                 CMP     #0
0421 F012              22                 BEQ     SI0
0423 A954              23                 LDA     #54H          ;n
0425 8D0355            24                 STA     DISBUF+2
0428 A95C              25                 LDA     #5CH          ;o
042A 8D0356            26                 STA     DISBUF+3
042D A93F              27                 LDA     #3FH          ;0
042F 8D0358            28                 STA     DISBUF+5
0432 4C041A            29                 JMP     REDITE
0435                   30 SI0
0435 A96D              31                 LDA     #6DH          ;S
0437 8D0355            32                 STA     DISBUF+2
043A A930              33                 LDA     #30H          ;I
043C 8D0356            34                 STA     DISBUF+3
043F A93F              35                 LDA     #3FH          ;0
```

```
0441 8D0358          36                 STA     DISBUF+5
0444 4C041A          37                 JMP     REDITE
```

OUTADD (FD22H): Coloca un valor en las 4 primeras posiciones del buffer DISBUF, que corresponden a los dígitos del campo de direcciones. El valor que carga en este buffer corresponde al código de segmentos equivalente a cada uno de los cuatro dígitos de la palabra de 16 bits formada por el registro rY (byte alto) y por el Acumulador (byte bajo). Esta subrutina no coloca puntos, no altera DISBUF+4 ni DISBUF+5 y no presenta datos en el display.

Como ejemplo del funcionamiento de esta subrutina se expone el siguiente programa:

```
                      1 "650X"
          < FD22 >    2 OUTADD           EQU     0FD22H
          < FE4D >    3 REFDIS           EQU     0FE4DH
                      4
                      5                  ORG     400H
                      6
0400 A934             7                  LDA     #34H
0402 A012             8                  LDY     #12H
0404 20FD22           9                  JSR     OUTADD
0407                 10 RD
0407 20FE4D          11                  JSR     REFDIS
040A 4C0407          12                  JMP     RD
```

OUTADH (FD2DH): Coloca en DISBUF y DISBUF+1 el código correspondiente al valor del Acumulador de forma similar a la anterior.

OUTDAT (FD32H): Esta subrutina es análoga a la anterior pero coloca el código en DISBUF+4 y DISBUF+5 (campo de datos).

OUTACC (FD34H): Saca el Acumulador como dos carácteres colocándolos en DISBUF a partir de la posición indicada por rX (el máximo valor de rX será 4).

OUTACH (FD3DH): Saca los bits 0 a 3 del Acumulador como un carácter, colocándolo en DISBUF+X.

CLRDIS (FD48H): Borra el buffer del display (DISBUF a DISBUF+5=0).

ON (FD67H): Presenta el texto "ON" durante aproximadamente 1 segundo en el display.

OFF (FD6FH): Presenta el texto "OFF" durante aproximadamente 1 segundo en el display.

GETADE (FD89H): Obtiene la dirección (4 dígitos) del teclado rotando a la izquierda hasta la pulsación de [FIN]. Los datos introducidos se presentan en el campo de dirección del display y se encienden los puntos correspondientes. Tras la primera pulsación, pone a '0' los tres dígitos izquierdos. Provoca error si se pulsa [+], [-] ó [FIN] como primera tecla. El dato introducido se coloca en las posiciones "PARH", "PARL" de memoria RAM.

GETADD (FD90H): Análoga a la anterior, pero no hay error si se pulsa [FIN] como primera tecla.

GETPAR (FD97H): Análoga a "GETADE", pero solamente obtiene dos dígitos que se colocan a la salida en "PARL".

GETP3 (FDB7H): Hace "PUNTO"=30H si "MODEC"=00H (dato) o "PUNTO"=3CH si "MODEC"=01H (dirección). Luego sigue en GETP31.

GETP31 (FDC0H): Guarda el valor del Acumulador a la entrada en "PUNTO" y pone a 1 la posición "PRIMER" para la detección de la primera pulsación en introducción de datos. Sigue en GETP32.

GETP32 (FDC8H): Obtiene un string de teclas hexadecimales. Si la tecla pulsada es [+] o [-] detiene la toma y sale con el flag de Carry del registro Status=1. Saca un dato o una dirección, según sea el valor de "MODEC".

En el APENDICE D, puede consultarse el listado completo del Programa Monitor del MI-650C.

PRACTICA 6.1 MENSAJE MOVIL EN DISPLAY

Este programa presenta un mensaje en el display del Microinstructor, en forma móvil de izquierda a derecha, permitiendo de este modo la presentación de textos con más de 6 carácteres.

Una vez iniciada la ejecución del programa, la presentación se efectúa ininterrumpidamente. Para separar cada presentación del mensaje, no se inicia una nueva presentación hasta que la anterior ha desaparecido totalmente.

El mensaje se grabará en una tabla de datos. Al principio de esta tabla se debe indicar el número de bytes que contiene.

El programa accede al display mediante las rutinas del Microinstructor CLRDIS, REFDIS, DISBUF y PUNTO. A continuación se explica el método empleado para la presentación del texto:

Al principio de cada presentación se inicializa a 00H un puntero de memoria. A partir de este puntero se extraen 6 datos del texto y se colocan en el buffer de display y, tras una espera (refrescado del display para presentación del buffer), se incrementa el puntero y se repite el proceso. El hecho de que cada vez se inicie la presentación de un carácter, produce la impresión de que el texto se desplaza hacia la izquierda. Cuando durante la toma de datos se alcance el fin de texto, se completan los carácteres sobrantes con espacios, para crear la separación entre textos y, cuando el puntero de memoria alcance el final, se reinicia a 00H, repitiendo el proceso.

Los datos que componen el mensaje a presentar representan cada uno un carácter en código de segmentos. Cada bit del dato controla un segmento, según la tabla de conversión presentada en la figura 6.3. Un '1' en un cierto bit da lugar al encendido del segmento correspondiente.

En esta práctica se propone la realización del ensamblado del programa, de modo que compruebe si ha adquirido los conocimientos necesarios para poder desarrollar sus propios programas en lenguaje máquina.

PROGRAMA PRINCIPAL

DIRECCION	CONTENIDO	ETIQUETA	INSTRUCCION		COMENTARIO
400		START	JSR	CLRDIS	;Limpia display
401					
402					
403			STX	PUNTO	;Y puntos
404					
405					
406			STX	0	;Inicia puntero
407					
408		PRESEN	LDY	#0	;Puntero a display
409					
40A			LDX	0	;Inicia puntero
40B					
40C		PRES1	LDA	TEXTO,X	;Toma carácter
40D					
40E					
40F		PRES2	STA	DISBUF,Y	;Pon en display
410					
411					
412			INY		
413			CPY	#6	;Completo?
414					
415			BCS	FINPR	;Sí, final.
416					
417			INX		;No, próximo.
418			CPX	LONG	;Fin texto
419					
41A					
41B			BCC	PRES1	;No
41C					
41D			LDA	#0	;Sí, borra carácter
41E					
41F			BEQ	PRES2	;Siempre.
420					
421		FINPR	INC	0	;Avanza inicio
422					
423			LDA	0	
424					
425			CMP	LONG	;Completo?
426					
427					
428			BCC	DELAY	;No,continúa
429					
42A			LDA	#0	;Sí, reinicia
42B					
42C			STA	0	;Puntero

```
42D
42E                 DELAY   LDA  #32H        ;Aprox. 500 ms
42F
430                         STA  1           ;Contador
431
432                 DELB1   JSR  REFDIS      ;Presenta (10 ms)
433
434
435                         DEC  1
436
437                         BNE  DELB1
438
439                         BEQ  PRESEN      ;Repite
```

TABLA DE DATOS
==========

```
43A                 LONG                     ;N' carácteres texto
43B                 TEXTO                    ;Texto
43C
 .
 .
 .
```

Las direcciones de las subrutinas de monitor empleadas son las siguientes:

```
CLRDIS = FD48H
REFDIS = FE4DH
DISBUF = 0353H
PUNTO  = 0361H
```

Las posiciones 0 y 1 de página 0 corresponden al puntero de texto y al contador interno respectivamente.

SOLUCION: La solución para un texto determinado es la siguiente:

```
20H, 48H, FDH, 8EH, 61H, 03H, 86H, 00H, A0H, 00H, A6H, 00H,  BDH,
3CH,  04H, 99H, 53H, 03H, C8H, C0H, 06H, B0H, 0AH, E8H, ECH, 3BH,
04H, 90H, EFH, A9H, 00H, F0H, EEH, E6H, 00H, A5H, 00H, CDH,  3BH,
04H,  90H, 04H, A9H, 00H, 85H, 00H, A9H, 32H, 85H, 01H, 20H, 4DH,
FEH, C6H, 01H, D0H, F9H, F0H, CDH, 0FH, 00H, 00H, 00H, 00H,  00H,
76H, 3FH, 38H, 77H, 00H, 77H, 5EH, 06H, 3FH, 6DH
```

CAPITULO 7

PRACTICAS

7.1 INTRODUCCION

En este capítulo se incluyen una serie de prácticas que pueden servir como complemento a los conocimientos teóricos adquiridos en los capítulos anteriores.

Algunas de ellas han sido incluídas en la memoria ROM del Microinstructor para permitir su ejecución sin que sea preciso introducir el programa desde el teclado. Las restantes deberán ser introducidas a mano.

Los programas grabados en ROM pueden transferirse a memoria RAM para ser modificados o variar alguno de sus parámetros. Para ello se proporciona un programa que efectúa esta transferencia automáticamente. Aunque este programa se muestra en la práctica 7.3 (Movimiento de datos entre zonas variables), el proceso para utilizarlo es el siguiente:

a. Colocar la parte baja de la dirección inicial del programa que se quiere transferir a memoria RAM, en la dirección de memoria 0000H.

b. Colocar la parte alta de la dirección inicial del programa que se quiere transferir a memoria RAM, en la dirección de memoria 0001H.

c. Colocar la parte baja de la dirección final del programa que se quiere transferir a memoria RAM, en la dirección de memoria 0002H.

d. Colocar la parte alta de la dirección final del programa que se quiere transferir a memoria RAM, en la dirección de memoria 0003H.

e. Colocar la parte baja de la dirección de destino donde se quiere cargar el programa, en la dirección de memoria 0004H.

f. Colocar la parte alta de la dirección de destino donde se quiere cargar el programa, en la dirección de memoria 0005H.

g. Ejecutar el programa en la dirección F770H:

[GO] [F] [7] [7] [0] [FIN]

h. Cuando termine la ejecución se indicará mediante los segmentos en el display.

i. modificar todas las direcciones absolutas que aparezcan en el programa, adaptándolas al nuevo origen en RAM.

Como ejemplo de este proceso, veamos a continuación los listados de una parte de un mismo programa, en sus dos versiones: versión inicial, tal como está contenido en RAM, y versión alterada, después de seguir el proceso anterior de transferencia:

```
      .
      .
      .
F3D1          BUCLE
F3D1 BDF405           LDA     >LEDTB,X        ;SACA ESTADO LED'S.
F3D4 8DA400           STA     RB
F3D7 BDF40F           LDA     >TIMTB,X
F3DA 20F3E4           JSR     RETAR           ;TIEMPO DE ESPERA.
F3DD E8               INX
      .
      .
      .
```

y para la versión en RAM, con origen en 400H:

```
      .
      .
      .
0407          BUCLE
0407 BD043B           LDA     >LEDTB,X        ;SACA ESTADO LED'S.
040A 8DA400           STA     RB
040D BD0445           LDA     >TIMTB,X
0410 20041A           JSR     RETAR           ;TIEMPO DE ESPERA.
0413 E8               INX
      .
      .
      .
```

Los listados que se facilitan en cada práctica están producidos por un ensamblador que maneja los nemónicos estándars para la CPU 6502. Debe prestarse especial atención a las direcciones absolutas que aparecen a lo largo del programa, puesto que aparecen invertidas respecto al orden de introducción en memoria. Por ejemplo:

```
04A5   20 04 83          JSR   PRDAT
```

se introducirá del siguiente modo:

```
04A5   20
04A6   83
04A7   04
```

con lo que se saltará a ejecutar la subrutina que se encuentra en la dirección 0483H.

Otro ejemplo podría ser:

```
04C3   AD 03 5A          LDA   PARL
```

Igual que antes esta instrucción se introduciría:

```
04C3   AD
04C4   5A
04C5   03
```

Para evitar errores es aconsejable la utilización de la tecla [STV/3] para la introducción de vectores.

Antes de proceder a la ejecución de cualquier programa diseñado para ser ejecutado en la memoria RAM, es aconsejable la protección de la RAM contra escritura.

Es recomendable la utilización del cassette para grabar los programas de una cierta longitud, tanto si funcionan correctamente como si están en fase de depuración de errores.

Una vez familiarizado con las técnicas de programación, puede ser un buen ejercico rediseñar los programas incluídos en cada práctica para que cumplan otra función.

7.2 MOVIMIENTO DE DATOS ENTRE ZONAS FIJAS

La finalidad del programa incluído en esta práctica es el movimiento de un bloque de 256 bytes como máximo desde una zona de memoria a otra.

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      1 "650X"
                      2 ;************************************************
                      3 ;          MOVIMIENTO DE DATOS ENTRE ZONAS FIJAS
                      4 ;************************************************
                      5
                      6
                      7       ORG     0400H
                      8
0400 A210             9 START LDX     #10H    ;DIMENSION DE LAS ZONAS
0402                 10 BUCLE
0402 BD05FF          11       LDA     600H-1,X       ;DATO DE ORIGEN
0405 9D04FF          12       STA     500H-1,X       ;A DESTINO
0408 CA              13       DEX             ;TODOS?
0409 D0F7            14       BNE     BUCLE   ;NO,REPITE
040B 60              15       RTS
```

Cada vez que el programa hace un paso por el bucle, el Acumulador varía de acuerdo con los datos existentes en memoria a partir de 600H. Nótese también que el operando de las instrucciones en las direcciones 402H y 405H se ha alterado de manera que se compensa el valor no cero de X en la última pasada. Con estos valores, se mueve el bloque de datos de 600H a 60FH a las direcciones 500H a 50FH. El valor real de los operandos será 5FFH y 4FFH. Se ha puesto 600H-1 y 500H-1 para significar la posibilidad de cálculo que tiene el ensamblador y para mayor claridad de identificación.

Lógicamente este porograma se puede alterar para modificar las direcciones de programa origen y destino, así como el número de bytes transferidos.

Ejecutar el programa e inspeccionar las direcciones de memoria correspondiente a la zona alterada.

7.3 MOVIMIENTO DE DATOS ENTRE ZONAS VARIABLES (F770H - F7CFH)

Este programa es similar al de la práctica anterior, pero permite más flexibilidad en lo que respecta a movimiento de datos, al emplear el valor de dos vectores en la página 0 ("OLDSTR" y "OLDEND") como base para el bloque de datos a transferir y una variable, tambien en página 0, para almacenar la dirección destino de los datos a transferir.

```
LOCATION OBJECT CODE LINE    SOURCE LINE

                      1 "650X"
                      2 ;MOVIMIENTO DE DATOS
                      3 ;------------------
                      4 ;
                      5       ORG     00H
0000                  6 OLDSTR        RMB     2
0002                  7 OLDEND        RMB     2
0004                  8 NEWSTR        RMB     2
0006                  9 NEWEND        RMB     2
0008                 10 COUNT RMB     2
                     11 ;
                     12 ;PROG
                     13       ORG     0F770H
F770                 14 MOVIT
F770 D8              15       CLD
F771 A0FF            16       LDY     #0FFH
F773 38              17       SEC
F774 A502            18       LDA     OLDEND  ;CALCULA EL NUM DE BYTES A MOVER
F776 E500            19       SBC     OLDSTR
F778 8508            20       STA     COUNT
F77A A503            21       LDA     OLDEND+1
F77C E501            22       SBC     OLDSTR+1
F77E 8509            23       STA     COUNT+1
F780 18              24       CLC
F781 A508            25       LDA     COUNT
F783 6504            26       ADC     NEWSTR
F785 8506            27       STA     NEWEND
F787 A509            28       LDA     COUNT+1
F789 6505            29       ADC     NEWSTR+1
F78B 8507            30       STA     NEWEND+1
F78D E608            31       INC     COUNT
F78F E609            32       INC     COUNT+1
F791 38              33       SEC
F792 A504            34       LDA     NEWSTR
F794 E500            35       SBC     OLDSTR
F796 A505            36       LDA     NEWSTR+1
F798 E501            37       SBC     OLDSTR+1
F79A                 38 LOOP
F79A A200            39       LDX     #0
F79C 9002            40       BCC     MOVE
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

   F79E A202         41       LDX     #2
   F7A0              42 MOVE
   F7A0 A100         43       LDA     [OLDSTR,X]
   F7A2 8104         44       STA     [NEWSTR,X]
   F7A4 9014         45       BCC     DOWN
                     46       ;AJUSTA PUNTEROS SUPERIORES O DISMINUYE
   F7A6 C602         47       DEC     OLDEND
   F7A8 98           48       TYA     ;COMPARA CON $FF (RY) SIN ALTERAR CY
   F7A9 4502         49       EOR     OLDEND
   F7AB D002         50       BNE     NOFIN1  ;NO FIN TODAVIA
   F7AD C603         51       DEC     OLDEND+1
   F7AF              52 NOFIN1
   F7AF C606         53       DEC     NEWEND
   F7B1 98           54       TYA
   F7B2 4506         55       EOR     NEWEND
   F7B4 D002         56       BNE     NOFIN2
   F7B6 C607         57       DEC     NEWEND+1
   F7B8              58 NOFIN2
   F7B8 900C         59       BCC     TSTTOT  ;SIEMPRE
                     60       ;AJUSTA PUNTEROS INFERIORES (AUMENTA)
   F7BA              61 DOWN
   F7BA E600         62       INC     OLDSTR
   F7BC D002         63       BNE     NOFIN3
   F7BE E601         64       INC     OLDSTR+1
   F7C0              65 NOFIN3
   F7C0 E604         66       INC     NEWSTR
   F7C2 D002         67       BNE     TSTTOT
   F7C4 E605         68       INC     NEWSTR+1
                     69       ;AJUSTA EL TOTAL DE BYTES A MOVER
   F7C6 C608         70 TSTTOT        DEC     COUNT
   F7C8 D0D0         71       BNE     LOOP
   F7CA C609         72       DEC     COUNT+1
   F7CC D0CC         73       BNE     LOOP
                     74 ;FIN DEL PROGRAMA
   F7CE 18           75       CLC
   F7CF 60           76       RTS
                     77       END
```

En primer lugar se calcula el total de datos a transferir y se almacena en COUNT. Luego se busca la dirección final donde se guardará el último dato transferido y se inicia la transferencia de datos hasta llegar a esta posición. Para ello se utiliza un direccionamiento preindexado.

El programa empieza con la etiqueta "MOVIT" y se encuentra en memoria a partir de la dirección F770H. Por lo tanto, no es preciso introducirlo a mano y se puede utilizar como programa de utilidad para transferir los otros programas grabados en la memoria EPROM del Microinstructor a la memoria RAM, como ya se ha explicado anteriormente.

Aunque el funcionamiento del programa es básicamente igual que el de la práctica anterior, su concepto es radicalmente diferente. En la práctica anterior, tanto las direcciones de base como la longitud del bloque están fijadas en el programa y, por tanto, son inalterables.

En cambio, en este caso todos estos datos están en memoria, por lo que puede usarse como subrutina general de transferencia de datos, con los valores antes citados inicializados en cada caso particular por el programa que efectúa la llamada o por el usuario.

Obsérvese que las direcciones 0000 a 0009 no generan código y por tanto no son consideradas a la hora de introducir el programa en memoria.

Antes de ejecutar el programa se deben inicializar los vectores OLDSTR, OLDEND y NEWSTR con los valores deseados. Para ello puede utilizarse la función **STV** del Microinstructor. En caso de efectuarse mediante la función **MEM,** recuérdese que en el vector se colocan las partes alta y baja de la dirección invertidas (primero parte baja y luego parte alta).

7.4 CONTADOR BINARIO UP-DOWN DE 8 BITS

El objeto de esta práctica es simular por software un contador binario Up-Down, cuya salida está constituida por los LEDs conectados a PB. El sentido de cuenta se controla mediante la línea PA0 (conmutador 0), de tal manera que si está en ON cuente de modo ascendente y si está en OFF la cuenta sea descendente. La velocidad de cuenta está fijada a una frecuencia de 2 Hz.

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      1 "650X"
                      2 ;**********************************
                      3 ;CONTADOR BINARIO UP-DOWN DE 8 BITS.
                      4 ;**********************************
                      5 ;
                      6
           < A400 >   7 RB    EQU     0A400H          ;REGISTROS 6522
           < A401 >   8 RA    EQU     0A401H
           < A402 >   9 DDRB  EQU     0A402H
           < A403 >  10 DDRA  EQU     0A403H
           < A408 >  11 TIM2L EQU     0A408H          ;TIMER 2 (LOW)
           < A409 >  12 TIM2H EQU     0A409H          ;TIMER 2 (HIGH E INICIO)
           < A40D >  13 IFR   EQU     0A40DH          ;REGISTRO DE FLAGS
                     14
                     15        ORG     0400H
                     16
0400                 17 START
0400 A9FF            18        LDA     #0FFH
0402 8DA402          19        STA     DDRB            ;PB A SALIDAS
0405 A900            20        LDA     #0H
0407 8DA403          21        STA     DDRA            ;PA A ENTRADAS
040A 8DA400          22        STA     RB              ;PC AL CONTADOR
040D                 23 BUCLE
040D ADA401          24        LDA     RA              ;SWITCH DE SENTIDO
0410 2901            25        AND     #01H
0412 F006            26        BEQ     UP
0414 CEA400          27        DEC     RB              ;DOWN (DECREMENTA)
0417 4C041D          28        JMP     S1
041A                 29 UP
041A EEA400          30        INC     RB              ;UP (INCREMENTA)
041D                 31 S1
041D A20A            32        LDX     #0AH            ;500 ms DE RETARDO.
041F                 33 DEL1
041F A950            34        LDA     #050H
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

  0421 8DA408       35       STA    TIM2L          ;INICIA TIMER
  0424 A9C3         36       LDA    #0C3H          ;A 50 ms.
  0426 8DA409       37       STA    TIM2H          ;E INICIA CUENTA
  0429              38 DEL2
  0429 A920         39       LDA    #20H
  042B 2DA40D       40       AND    IFR            ;ESPERA FIN DE TIEMPO
  042E F0F9         41       BEQ    DEL2
  0430 CA           42       DEX                   ;10 VECES?
  0431 D0EC         43       BNE    DEL1           ;NO, REPITE
  0433 4C040D       44       JMP    BUCLE          ;PROCESA CONTEO DE NUEVO
```

La velocidad del contador se controla mediante una rutina de retardo de 500 ms. En ésta se emplea uno de los timers que incluye el circuito 65C22. Estos timers son contadores de 16 bits. El clock es de 1 MHz. Por tanto el máximo tiempo que se podrá optener es de aproximadamente 65 ms.

Dado que este tiempo es insuficiente, es necesario el empleo de dos bucles internos, uno controlado por el timer que da un tiempo de 50 ms y otro externo que repite 10 veces.

El control del temporizador requiere primeramente cargar en el contador el valor deseado en función del tiempo (en este caso 50 ms). Esto borra el flag en el bit 5 del registro IFR e inicia la cuenta atrás. Cuando el valor del contador llega a '0' se pone a '1' el citado flag. El bucle en las direcciones 429H y 42EH efectúa esta espera.

La ejecución del programa se efectúa a partir de la dirección START y la detención mediante la tecla [STOP/BRK].

7.5 ANALIZADOR LOGICO (0F419H-0F445H)

Un analizador lógico es un equipo de laboratorio especialmente destinado a trabajar con circuitos lógicos. Incluye un cierto número de entradas en paralelo que se conectan a diversos puntos del circuito. En el momento en que se le suministra una señal de arranque, comienza a tomar los datos presentes en esas líneas a una velocidad controlada por una señal de reloj y los mantiene en memoria. Una vez rellenada esta memoria se detiene la toma y se puede inspeccionar el contenido de esta memoria. Mediante este instrumento se pueden almacenar gran cantidad de datos a alta velocidad, efectuando después un detenido examen de los mismos, lo cual generalmente no es posible realizarlo con un osciloscopio convencional.

Las características que se pueden obtener en el analizador lógico incluido en esta práctica no pueden compararse con las obtenidas por los equipos convencionales, básicamente en lo que a velocidad de adquisición de datos se refiere. No obstante, es una aplicación interesante de un proceso de control de datos que, incluso con sus limitaciones, puede utilizarse eficazmente.

Aunque aumentando la complejidad del programa se podrían obtener mejores prestaciones, este programa las reduce al mínimo imprescindible, con el fin de no complicar excesivamente la práctica. Las características fundamentales son:

- Líneas de entrada: 8, todas con niveles lógicos TTL.
- Tamaño de la memoria: 16 palabras.
- Frecuencia máxima de trabajo: 10 KHz.
- Señal de disparo: válida con nivel lógico '1'.
- Clock: externo. La toma de datos se efectúa en el flanco de subida del clock.

Las líneas de entrada están constituidas por PA. La línea PB7 es la entrada de clock y PB6 es la de disparo.

Con el fin de observar la actividad del programa en trabajos muy lentos, los LEDs 0 a 3 indican (en binario) el número de orden del dato tomado.

Es interesante observar en este programa el empleo de las líneas PB7 y PB6.

El doble bucle TD1 y TD2 sincroniza la toma de datos en el flanco de subida del clock. El primer bucle espera la desaparición de la señal de clock (PA7='0') y el segundo espera la aparición de la misma (PB7='1').

Para utilizar este programa seguir el siguiente procedimiento:

- Conectar PA0 a PA7 a los puntos de medida.

- Conectar PB7 a la señal de clock que sincroniza los estados lógicos en el circuito de prueba.

- Conectar PA6 a una señal de disparo. La toma de datos se inicia en el momento en que esta señal pasa a '1', o bien en el momento del lanzamiento del programa si ya está a '1'. Si no hay señal de disparo utilizable se puede dejar esta línea sin conectar.

- Ejecutar el programa a partir de la dirección START.

- Cuando retorna el control al monitor, los datos tomados se pueden inspeccionar en BUFFER (direcciones 500H a 50FH) mediante la función MEM. El usuario deberá efectuar la transformación de hexadecimal a binario para ver el nivel de cada línea.

El programa admite gran cantidad de modificaciones y mejoras (incluir un reloj interno, cambiar los flancos del clock para toma de datos, ampliar la memoria de adquisición, etc), que se dejan como ejercicio para el alumno.

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      1       "650X"
                      2       ;*********************************************
                      3       ; a n a l i t z a d o r    l o g i c  (n_mma1)
                      4       ;*********************************************
           < A400 >   5 RB    EQU     0A400H
           < A401 >   6 RA    EQU     0A401H
           < A402 >   7 DDRB  EQU     0A402H
           < A403 >   8 DDRA  EQU     0A403H
                      9
                     10 ;BUFFER DE DATOS
           < 0500 >  11 BUFFER        EQU     0500H
                     12
                     13 ;PROG
                     14        ORG     0F419H
                     15
   F419              16 START
   F419 A900         17        LDA     #0
   F41B 8DA403       18        STA     DDRA    ;PA A ENTRADAS
   F41E 8DA400       19        STA     RB      ;PB A CERO
   F421 A90F         20        LDA     #0FH
   F423 8DA402       21        STA     DDRB    ;PB0-PB3: SALIDAS
                     22
                     23 ;ESPERA A SENAL DE START
   F426              24 STR1
   F426 2CA400       25        BIT     RB
   F429 50FB         26        BVC     STR1
                     27
                     28 ;INICIO DE TOMA
   F42B A200         29        LDX     #0      ;PUNTERO DE BUFFER
                     30
                     31 ;ESPERA FLANCO DE SUBIDA DE CLOCK (PB7)
   F42D              32 TD1
   F42D 2CA400       33        BIT     RB
   F430 30FB         34        BMI     TD1     ;ESPERA CLK=0
   F432              35 TD2
   F432 2CA400       36        BIT     RB
   F435 10FB         37        BPL     TD2     ;ESPERA CLK=1
                     38
                     39 ;TRANSFIERE DATO A BUFFER
   F437 ADA401       40        LDA     RA      ;TOMA ENTRADAS
   F43A 9D0500       41        STA     BUFFER,X
   F43D 8EA400       42        STX     RB      ;INDICA BYTE TOMADO
   F440 E8           43        INX
   F441 E010         44        CPX     #16     ;TODOS?
   F443 D0E8         45        BNE     TD1     ;NO, REPITE
   F445 60           46        RTS             ;FIN DE TOMA,RETORNO
```

7.6 SUMA EN DOBLE PRECISION

Esta es la primera práctica de una serie destinada a la familiarilización con el cálculo matemático mediante ordenador.

Se trata de sumar dos números en complemento a dos de 16 bits, dando un resultado también de 16 bits en complemento a dos.

Los sumandos se encuentran, en formato byte bajo-byte alto, en las cuatro primeras posiciones de página 0. y el resultado en las dos siguientes.

```
LOCATION OBJECT CODE LINE    SOURCE LINE

                      1 "650X"
                      2 ;**********************
                      3 ;SUMA EN DOBLE PRECISION
                      4 ;**********************
                      5 ;
                      6
            < 0000 >  7 OPR1  EQU    0            ;SUMANDO 1
            < 0002 >  8 OPR2  EQU    2            ;SUMANDO 2
            < 0004 >  9 RESULT       EQU    4     ;RESULTADO
                     10
                     11       ORG    0400H
                     12
0400                 13 START
0400 D8              14       CLD                 ;MODO BINARIO
0401 18              15       CLC                 ;INICIA CARRY
0402 A500            16       LDA    OPR1
0404 6502            17       ADC    OPR2         ;SUMA PARTES BAJAS
0406 8504            18       STA    RESULT
0408 A501            19       LDA    OPR1+1
040A 6503            20       ADC    OPR2+1       ;Y ALTAS
040C 8505            21       STA    RESULT+1
040E 60              22       RTS
```

El método empleado se basa en una iteración de la instrucción ADC (suma de dos números de 8 bits). Esta instrucción efectúa la suma de:

(Contenido de memoria)+(Contenido Acumulador)+(bit de carry)

Inicialmente se pone a '0' el bit de acarreo (carry). A continuación se suman las partes bajas de los sumandos y se almacenan en la parte baja del resultado. Si el resultado de esta operación excede los valores que pueden almacenarse en un byte, el bit de carry automáticamente se pone a '1'; en caso contrario, sigue en '0'. A continuación se realiza la suma de las partes altas de los sumandos, posiblemente alterada por el carry, y se guarda el valor obtenido en la parte alta del resultado. De nuevo inspeccionando el bit de carry podremos deducir si el resultado excede el rango de los números (16 bits).

Téngase en cuenta que los números están representados en complemento a dos.

Este programa puede ser aplicado, siguiendo la misma técnica, para sumar números de cualquier longitud y funciona exactamente igual si los sumandos son números binarios sin signo.

Cambiando la instrucción CLD por una SED (F8H), este programa se aplica para suma de números decimales en BCD. En este caso, los números serán sin signo (admitiendo el margen 0 a 9999) y se colocarán en memoria en las mismas direcciones y de igual manera que hemos indicado para los números binarios.

7.7 RESTA EN DOBLE PRECISION (F446H-F454H)

El programa para esta práctica es similar al anterior:

```
LOCATION OBJECT CODE LINE    SOURCE LINE

                      1 "650X"
                      2 ;************************
                      3 ;RESTA EN DOBLE PRECISION
                      4 ;************************
0000                  5 OPR1  RMB     2       ;MINUENDO
0002                  6 OPR2  RMB     2       ;SUSTRAENDO
0004                  7 RESULT        RMB     2       ;RESULTADO
                      8
                      9 ;
                     10       ORG     0F446H
                     11
F446                 12 START
F446 D8              13       CLD             ;MODO BINARIO
F447 38              14       SEC             ;INICIA CARRY
F448 A500            15       LDA     OPR1
F44A E502            16       SBC     OPR2
F44C 8504            17       STA     RESULT
F44E A501            18       LDA     OPR1+1
F450 E503            19       SBC     OPR2+1  ;ALTAS
F452 8505            20       STA     RESULT+1
F454 60              21       RTS
```

Obsérvese que en este caso en la inicialización se efectúa una puesta a '1' del bit de acarreo. Esto es debido a que la operación que efectúa la CPU con la instrucción SBC es la siguiente:

(Contenido Acumulador)-(Contenido de memoria)
-(bit de carry complementado)

El valor que representa al carry complementado recibe el nombre de Borrow.

Como en el caso anterior, esta rutina es válida para números binarios en complemento a dos y, cambiando CLD por SED, para decimales en BCD.

7.8 MULTIPLICACION DE DOS NUMEROS SIN SIGNO

En este caso se trata de multiplicar dos números binarios de 8 bits. El resultado tendrá 16 bits.

El método empleado para la multiplicación es análogo al empleado cuando ésta se hace manualmente.

```
          1 0 1 1 0     = 22   (16H)
        x   1 1 0 1     = 13   (DH)
        -----------
          1 0 1 1 0
        0 0 0 0 0
      1 0 1 1 0
    1 0 1 1 0
    ---------------
1 0 0 0 1 1 1 1 0     = 286  (11EH)
```

Como se puede observar, toda la operación está compuesta de sumas y desplazamientos.

```
LOCATION OBJECT CODE LINE   SOURCE LINE

                      1 "650X"
                      2 ;*********************************
                      3 ;MULTIPLICACION A 8 BITS SIN SIGNO
                      4 ;*********************************
                      5 ;
                      6 ;
               < 0000 >  7 MULTP EQU    0       ;FACTORES (8 BITS)
               < 0001 >  8 MULTR EQU    1
               < 0002 >  9 PROD  EQU    2       ;PRODUCTO (16 BITS)
                     10 ;
                     11       ORG    0400H
0400                 12 START
0400 D8              13       CLD             ;MODO BINARIO
0401 A900            14       LDA    #0       ;PAC DE MSBY DEL PRODUCTO
0403 A208            15       LDX    #8       ;NUMERO DE PASADAS
0405                 16 NXBT
0405 4601            17       LSR    MULTR    ;=0, NO SUMA
0407 9003            18       BCC    NOADD
0409 18              19       CLC
040A 6500            20       ADC    MULTP    ;=1, SUMA
040C                 21 NOADD
040C 4A              22       LSR    A        ;DESPLAZA EL RESULTADO
040D 6602            23       ROR    PROD     ;(16 BITS)
040F CA              24       DEX             ;8 VECES?
0410 D0F3            25       BNE    NXBT     ;NO, REPITE
0412 8503            26       STA    PROD+1   ;SI,SALVA ALTO DEL PRODUCTO
0414 60              27       RTS
```

El registro X se emplea como contador del número de iteraciones. Una por cada bit del multiplicador.

El funcionamiento del programa, una vez definidos MULTR como 00001101 y MULTP como 00010110, es el siguiente:

Primera pasada: la instrucción LSR MULTR introduce en el flag de carry el bit 1 del multiplicador. Dado que su valor es '1' se efectúa la suma. El valor que queda en MULTR es 00000110. El valor del acumulador es '0'. Al ser sumado con MULTP (00010110), da el mismo resultado (00010110). La instrucción LSR A deja en el Acumulador el valor 00001011 y en carry un 0, que pasa a la parte baja del resultado (PROD), que queda como 0XXXXXXX (X=indiferente).

Segunda pasada: LSR MULTR deja en MULTR el valor 00000011 y en carry un '0', luego no se efectúa la suma. El desplazamiento a derechas del resultado deja en el Acumulador el valor 00000101 y en PROD 10XXXXXX.

Tercera pasada: LSR MULTR deja en MULTR el valor 00000001, pone el carry a '1' y efectúa la suma. Su resultado es 10110+101=00011011, valor que pasa al acumulador. El desplazamiento del Acumulador y de PROD deja en el Acumulador el valor 00001101 y en PROD 110XXXXX.

Cuarta pasada: El nuevo valor de MULTR es 00000000 y el de carry es '1'. La suma deja el valor 00010110+00001101=00100011 en el Acumulador. Los desplazamientos colocan en el Acumulador el valor 00010001 y en PROD 110XXXXX.

En esta pasada y en las siguientes no se efectúa suma, ya que los bits correspondientes de MULTR son '0', por lo que se reducen a 4 desplazamientos del resultado, que queda:

```
Quinta pasada: Acumulador=00001000  PROD=11110XXX
Sexta pasada:      "     =00000100   "  =011110XX
Séptima pasada:    "     =00000010   "  =0011110X
Octava pasada:     "     =00000001   "  =00011110
```

El resultado queda igual que el que obtuvimos al efectuar las operaciones manualmente.

La multiplicación en doble precisión (operandos de 16 bits) se efectúa con el mismo algoritmo. La diferencia estriba en que, al tener que manejar hasta 32 bits de resultado, el programa se hace más largo. Su realización se deja como ejercicio para el alumno.

7.9 DIVISION DE DOS NUMEROS SIN SIGNO

En esta práctica se va a realizar un programa que divida un número binario de 8 bits en DIVID por otro de 8 bits en DIVIS. El cociente de 8 bits se obtiene en DIVID y el resto en RESTO.

El algoritmo empleado es similar al usado para la multiplicación (mediante desplazamientos y restas en este caso) y el programa es el siguiente:

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      1 "650X"
                      2 ;**************************
                      3 ;DIVISION A 8 BITS SIN SIGNO
                      4 ;**************************
                      5 ;
                      6 ;
           < 0000 >   7 DIVID EQU    0             ;DIVIDENDO (8 BITS)
           < 0001 >   8 DIVIS EQU    1             ;DIVISOR (8 BITS)
           < 0002 >   9 RESTO EQU    2             ;RESTO  (8 BITS)
                     10       ;EL COCIENTE SUBSTITUYE A DIVID
                     11 ;
                     12       ORG    0400H
                     13 ;
0400                 14 START
0400 D8              15       CLD            ;MODO BINARIO
0401 A900            16       LDA    #0      ;PAC DE DIVID PARCIAL
0403 A208            17       LDX    #8      ;NUMERO DE PASADAS
0405                 18 NXBT
0405 0600            19       ASL    DIVID   ;DESPLAZA DIVID/COCIENTE
0407 2A              20       ROL    A       ;EN DIVID PARCIAL
0408 C501            21       CMP    DIVIS   ;COMPARA CON DIVISOR
040A 9004            22       BCC    NOSB    ;DIVIS>DIVID.PARCIAL?
040C E501            23       SBC    DIVIS   ;NO, RESTA DIVISOR
040E E600            24       INC    DIVID   ;Y PON BIT EN COCIENTE
0410                 25 NOSB
0410 CA              26       DEX
0411 D0F2            27       BNE    NXBT
0413 8502            28       STA    RESTO   ;SALVA RESTO
0415 60              29       RTS
```

En este listado se indica la función de cada una de las instrucciones que lo componen. El análisis de funcionamiento se deja como ejercicio al alumno (efectuar un trazado del programa), así como su ampliación a números de doble precisión.

7.10 **RAIZ CUADRADA (F455H-F470H)**

En este caso vamos a calcular la raíz cuadrada de un número binario de 8 bits en OPR. El resultado se obtiene en SQR y el resto de la raíz en RESTO.

El método usado se basa en que la raíz cuadrada de un número entero es igual a las veces que pueden restarse de él los sucesivos números impares.

Veamos un ejemplo del cálculo de la raíz cuadrada de 25:

```
 25
 -1     Resultado parcial = 1
---
 24
 -3     Resultado parcial = 2
---
 21
 -5     Resultado parcial = 3
---
 16
 -7     Resultado parcial = 4
---
  9
 -9     Resultado parcial = 5
---
  0     Resto 0
```

Debido al método de iteración empleado, esta subrutina no es especialmente rápida, pero presenta la ventaja de su simplicidad.

El programa que implementa este algoritmo es el siguiente:

```
LOCATION OBJECT CODE LINE    SOURCE LINE

                      1 "650X"
                      2 ;*************
                      3 ;RAIZ CUADRADA
                      4 ;*************
                      5
0000                  6 OPR   RMB    1      ;OPERANDO (8 bits)
0001                  7 SQR   RMB    1      ;RAIZ (8 bits)
0002                  8 RESTO RMB    1      ;RESTO (8 bits)
0003                  9 TEMP  RMB    1      ;TEMPORAL DE CALCULO
                     10
                     11 .
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      12      ORG     0F455H
                      13
F455                  14 START
F455 D8               15      CLD
F456 A000             16      LDY     #0      ;RAIZ=0
F458 A901             17      LDA     #1      ;PRIMER IMPAR
F45A 8503             18      STA     TEMP    ;SALVA
F45C A500             19      LDA     OPR
F45E                  20 REP
F45E C503             21      CMP     TEMP    ;SE PUEDE RESTAR?
F460 900A             22      BCC     FIN     ;NO, FINAL
F462 E503             23      SBC     TEMP    ;SI, RESTA
F464 C8               24      INY             ;Y AUMENTA RESULTADO
F465 E603             25      INC     TEMP    ;PASA A PROXIMO IMPAR
F467 E603             26      INC     TEMP
F469 4CF45E           27      JMP     REP     ;REPITE
F46C                  28 FIN
F46C 8401             29      STY     SQR     ;SALVA RAIZ
F46E 8502             30      STA     RESTO   ;Y RESTO
F470 60               31      RTS
```

Como en otros casos, la traducción a la raíz cuadrada de un número de 16 bits es inmediata, con la consiguiente complejidad que presenta el mayor tamaño de los operandos.

7.11 **CALCULADORA**

En esta práctica se va a realizar una calculadora para sumar números positivos de hasta 4 dígitos (0 a 9999), empleando para ello el teclado y el display del Microinstructor.

Antes de realizar esta práctica es interesante leer con detenimiento el capítulo 6, en el que se describe el software del Microinstructor, especialmente en lo que se refiere al programa monitor, ya que se emplean algunas rutinas del mismo.

El listado de este programa es el siguiente:

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      1 "650X"
                      2 ;*****************************
                      3 ;         CALCULADORA
                      4 ;*****************************
                      5 ;
                      6 ;
                      7       ORG     0
                      8 ;VARIABLES
               < 0000 >   9 OPRL  EQU     0H      ;PRIMER SUMANDO (BAJO)
               < 0001 >  10 OPRH  EQU     1H      ;ALTO
                     11 ;
                     12 ;DIRECCIONES DEL MONITOR
                     13 ;
               < FD48 >  14 CLRDIS        EQU     0FD48H  ;BORRA EL DISPLAY
               < FD89 >  15 GETADE        EQU     0FD89H  ;OBTEN 4 DIGITOS HEX
               < FD22 >  16 OUTADD        EQU     0FD22H  ;SACA 4 DIGITOS HEX
               < FE09 >  17 GETKEY        EQU     0FE09H  ;ATIENDE TECLADO Y DISP.
               < 0353 >  18 DISBUF        EQU     353H    ;BUFFER DE DISPLAY
               < 0361 >  19 PUNTO EQU     361H    ;CONTROL DE PUNTOS
               < 0359 >  20 PARL  EQU     359H    ;PARAMETRO TOMADO EN GETADD
               < 035A >  21 PARH  EQU     35AH
                     22 ;
                     23       ORG     0400H
                     24 ;
                     25 ;OBTEN PRIMER SUMANDO
                     26 ;
0400                 27 START
0400 20FD48          28       JSR     CLRDIS  ;BORRA DISPLAY
0403 A906            29       LDA     #6
0405 8D0353          30       STA     DISBUF+5        ;NUMERO '1'
0408 20FD89          31       JSR     GETADE  ;OBTEN DATOS
040B AD0359          32       LDA     PARL
040E 8500            33       STA     OPRL    ;SALVA PRIMER SUMANDO
```

```
LOCATION OBJECT CODE LINE    SOURCE LINE

   0410 AD035A       34       LDA     PARH    ;
   0413 8501         35       STA     OPRH
                     36 ;
                     37 ;OBTEN SEGUNDO SUMANDO
                     38 ;
   0415 20FD48       39       JSR     CLRDIS
   0418 A95B         40       LDA     #5BH
   041A 8D0358       41       STA     DISBUF+5        ;NUMERO '2'
   041D 20FD89       42       JSR     GETADE
                     43 ;
                     44 ;EVALUA Y PRESENTA EL RESULTADO
                     45 ;
   0420 F8           46       SED             ;MODO DECIMAL
   0421 18           47       CLC
   0422 AD0359       48       LDA     PARL    ;BAJOS
   0425 6500         49       ADC     OPRL
   0427 48           50       PHA             ;SALVA EN STACK
   0428 AD035A       51       LDA     PARH
   042B 6501         52       ADC     OPRH
   042D A8           53       TAY
   042E 68           54       PLA             ;RESULTADO EN Y-A
   042F D8           55       CLD
   0430 A200         56       LDX     #0      ;ANALIZA OVERFLOW
   0432 9002         57       BCC     SIGUE   ;NO HAY
   0434 A220         58       LDX     #100000B        ;PUNTO EN DIG. IZQ.
                     59 ;
   0436              60 SIGUE
   0436 8E0361       61       STX     PUNTO
   0439 20FD22       62       JSR     OUTADD  ;SACA RESULTADO
   043C A950         63       LDA     #50H
   043E 8D0358       64       STA     DISBUF+5        ;LETRA 'r'
                     65 ;
   0441              66 BUC2
   0441 20FE09       67       JSR     GETKEY  ;ESPERA A TECLA
   0444 90FB         68       BCC     BUC2    ;NO HAY
   0446 4C0400       69       JMP     START   ;SI, REINICIA
```

Para utilizar este programa seguir el siguiente procedimiento:

- [GO] [4] [0] [0] [FIN]

- Introducir el primer número y pulsar [FIN]. Si hay errores proceder de manera normal para su corrección. (Entrada indicada por 1).

- Introducir el segundo número y pulsar [FIN]. (Entrada indicada por 2).

- El valor que aparece en el display es el resultado de la suma. Obsérvese que si si el resultado es mayor que 9999 el número 1 del quinto dígito no se presenta y en su lugar se enciende el punto del dígito izquierdo. (salida indicada por r).

- Pulsar cualquier tecla para reiniciar el cálculo.

- El programa se abandona mediante la tecla [STOP/BRK].

> **NOTA:** Cuando se introduzcan números debemos asegurarnos de que son decimales (dígitos 0 a 9 solamente), ya que en caso contrario se producirán errores de cálculo.

> **NOTA:** Este programa puede alterarse fácilmente, cambiando la instrucción SED por CLD, para efectuar sumas de números de 16 bits en complemento a 2.

7.12 CONVERSION HEXADECIMAL-DECIMAL (F471H-F522H)

En esta práctica vamos a tratar un método de conversión de base de numeración. La entrada será un número binario de 8 bits o binario en complemento a 2, opcionalmente. La salida será el número decimal (con signo) equivalente.

Dada la utilidad que tiene esta conversión, el programa se va a estructurar en tres bloques fácilmente separables:

- Adquisición del número binario.
- Conversión a decimal.
- Presentación de resultados.

Alterando adecuadamente los bloques de entrada y salida podemos hacer el programa compatible con cualquier aplicación.

El listado del programa es el siguiente:

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      1 "650X"
                      2 ;
                      3 ;CONVERSION HEXADECIMAL-DECIMAL DE 8 BITS
                      4 ;
                      5 ;DATOS DEL MONITOR
< 0359 >              6 PARL      EQU   00359H
< 035A >              7 PARH      EQU   0035AH
< FD34 >              8 OUTACC    EQU   0FD34H
< FD3D >              9 OUTACH    EQU   0FD3DH
< FD48 >             10 CLRDIS    EQU   0FD48H
< FD89 >             11 GETADE    EQU   0FD89H
< FE09 >             12 GETKEY    EQU   0FE09H
< 0353 >             13 DISBUF    EQU   00353H
< 0361 >             14 PUNTO     EQU   00361H
< FF9A >             15 SEGMTB    EQU   0FF9AH
                     16
                     17 ;VARIABLES
< 0000 >             18 NUML      EQU   00
< 0003 >             19 RESULT    EQU   03H
                     20
                     21 ;CONSTANTES
< 0010 >             22 MAS       EQU   10H
< 0012 >             23 FIN       EQU   12H
                     24 ;
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     25 ;
                     26 ;
                     27 ;
                     28          ORG   0F471H
                     29 ;
                     30 ;***BLOQUE DE ADQUISICION***
F471                 31 START
F471 A900            32          LDA   #0
F473 8D035A          33          STA   PARH
F476 20FD48          34          JSR   CLRDIS
F479 20FD89          35          JSR   GETADE  ;OBTEN NUMERO HEX DE 2 BYTES
F47C 20FD48          36          JSR   CLRDIS
F47F AD0359          37          LDA   PARL
F482 8500            38          STA   NUML
F484 A200            39          LDX   #0
F486 8E0361          40          STX   PUNTO   ;BORRA PUNTOS DEL DISPLAY
F489 AD035A          41          LDA   PARH
F48C D071            42          BNE   ERROR   ;MAS DE 1 BYTE--> ERROR
F48E 8603            43          STX   RESULT
F490 8604            44          STX   RESULT+1
F492 4CF4BC          45          JMP   POSI    ;SALTO ALTERABLE
                     46 ;
                     47 ;ELIMINANDO ESTE SALTO CONSIDERAREMOS EL
                         NUMERO DE ENTRADA COMO
                     48 ;BINARIO COMPLEMENTO A DOS
                     49 ;
F495 A500            50          LDA   NUML
F497 2980            51          AND   #80H            ;OBTEN SIGNO
F499 F021            52          BEQ   POSI    ;POSITIVO-->CONVERSION NORMAL
F49B A500            53          LDA   NUML    ;NEGATIVO-->COMPLEMENTA A 2
F49D 297F            54          AND   #7FH
F49F D008            55          BNE   NEGA    ;CASO DEL 80H
F4A1 A940            56          LDA   #40H
F4A3 8D0353          57          STA   DISBUF
F4A6 4CF4C1          58          JMP   BPODEC
                     59
                     60 ;***BLOQUE DE SALIDA DE SIGNO***
F4A9                 61 NEGA
F4A9 A940            62          LDA   #40H    ;SIGNO NEGATIVO A DISBUF+0
F4AB 8D0353          63          STA   DISBUF
F4AE A9FF            64          LDA   #0FFH   ;INVIERTE NUMERO BINARIO
F4B0 4500            65          EOR   NUML
F4B2 38              66          SEC           ;SUMA 1
F4B3 6900            67          ADC   #0
F4B5 297F            68          AND   #7FH
F4B7 8500            69          STA   NUML
F4B9 4CF4C1          70          JMP   BPODEC
F4BC                 71 POSI
F4BC A900            72          LDA   #0      ;TODOS LOS SEGMENTOS APAGADOS
F4BE 8D0353          73          STA   DISBUF
                     74
```

```
LOCATION OBJECT CODE LINE    SOURCE LINE

                     75 ;***BLOQUE DE CONVERSION***
F4C1                 76 BPODEC
F4C1 D8              77          CLD
F4C2 38              78          SEC
F4C3 A500            79          LDA   NUML
F4C5 E901            80          SBC   #1      ;RESTA 1 AL NUMERO HEX
F4C7 8500            81          STA   NUML    ;HASTA QUE SEA NEGATIVO
F4C9 9015            82          BCC   OUTCHR
F4CB                 83 NOZERO
F4CB 18              84          CLC
F4CC A901            85          LDA   #1      ;SUMA 1 EN MODO DEC. A RESULT
F4CE F8              86          SED           ;CADA VEZ QUE DECREMENTES EL
F4CF 6503            87          ADC   RESULT  ;NUMERO HEX
F4D1 8503            88          STA   RESULT
F4D3 90EC            89          BCC   BPODEC  ;SI CARRY, SUMA 1 A RESUL+1
F4D5 18              90          CLC
F4D6 A901            91          LDA   #1
F4D8 F8              92          SED
F4D9 6504            93          ADC   RESULT+1
F4DB 8504            94          STA   RESULT+1
F4DD 4CF4C1          95          JMP   BPODEC
                     96
                     97 ;***BLOQUE DE SALIDA DEL VALOR CONVERTIDO***
F4E0                 98 OUTCHR
F4E0 A201            99          LDX   #1      ;SACA RESULT+1 A DISBUF+1
F4E2 A504           100          LDA   RESULT+1
F4E4 20FD3D         101          JSR   OUTACH
F4E7 A202           102          LDX   #02H    ;SACA RESULT A DISBUF+2 I A
F4E9 A503           103          LDA   RESULT  ;DISBUF+3
F4EB 20FD34         104          JSR   OUTACC
                    105
                    106 ;***BLOQUE DE PRESENTACION***
F4EE                107 PRESEN
F4EE 20FE09         108          JSR   GETKEY
F4F1 90FB           109          BCC   PRESEN  ;TECLA PULSADA-->SIGUE
F4F3 C910           110          CMP   #MAS    ;+?
F4F5 D003           111          BNE   PRES1
F4F7 4CF471         112          JMP   START   ;SI-->NUEVO PROCESO
F4FA                113 PRES1
F4FA C912           114          CMP   #FIN    ;FIN?
F4FC D0F0           115          BNE   PRESEN
F4FE 00             116          BRK           ;SI-->BRK
                    117
                    118 ;***BLOQUE ENTRADA INCORRECTA***
F4FF                119 ERROR
F4FF A979           120          LDA   #79H
F501 8D0357         121          STA   DISBUF+4
F504 A950           122          LDA   #50H
F506 8D0358         123          STA   DISBUF+5
F509 4CF4EE         124          JMP   PRESEN
```

Obsérvese que existe un bloque, no citado anteriormente, para presentación de datos, pero que no afecta básicamente el proceso de conversión.

Eliminando el salto JMP POSI, se considerará el número de entrada como codificado en complemento a 2 y, por tanto, con signo. Para efectuar esta modificación, puede reemplazarse la instrucción de salto por instrucciones NOP (0EAH).

Introducir un número hexadecimal de hasta 4 dígitos terminando la introducción mediante la pulsación de la tecla [FIN].

Inmediatamente aparece el número decimal equivalente. Pulsando [+], se repite la conversión y, pulsando [FIN], se devuelve el control al programa monitor.

> **NOTA:** Un error en la introducción del número da lugar también a un retorno al monitor, con presentación de un texto de error.

7.13 CONVERSION DECIMAL-HEXADECIMAL (F523H-F5AFH)

Esta práctica es la complementaria de la anterior. La entrada estará constituída por un número decimal sin signo, entre 0 y 65535, y la salida será un número binario, también sin signo, de 16 bits.

Como en el caso anterior, el programa se ha dividido en bloques. No obstante, dado el diferente tratamiento que supone la conversión en este caso, se consideran solamente dos bloques:

- Adquisición/Presentación

- Conversión

La diferencia fundamental entre ambos métodos consiste en que, para la conversión de binario a decimal, se tomaba el número y a continuación se convertía, mientras que, en la conversión decimal a binario, la conversión se efectúa a medida que se reciben datos del teclado. Para independizar totalmente la adquisición de la conversión, será necesario crear un string con los datos de entrada, hasta la recepción de la tecla [FIN] y, posteriormente, recoger los diferentes carácteres del string y efectuar la conversión, lo que complica el programa.

Por razones similares no se incluye la posibilidad de entrada de números negativos, aunque, no obstante, puede ser un ejercicio interesante la modificación del programa para que sean admitidos.

El listado del programa es el siguiente:

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      1 "650X"
                      2 ;*******************************
                      3 ;conversio decimal  -- hexadecimal
                      4 ;*******************************
                      5 ;CONVERSION DECIMAL-HEX
                      6 ;VARIABLES
< 0000 >              7 NUML  EQU     00      ;NUMERO A CONVERTIR
< 0001 >              8 NUMH  EQU     01
< 0002 >              9 BUFPUN        EQU     02
                     10
                     11 ;DATOS DEL MONITOR
< F7D3 >             12 ERROR EQU     0F7D3H  ;REENTRADA DE ERROR
< FD48 >             13 CLRDIS        EQU     0FD48H
< FE09 >             14 GETKEY        EQU     0FE09H
< FD22 >             15 OUTADD        EQU     0FD22H
< 0353 >             16 DISBUF        EQU     0353H
< 0361 >             17 PUNTO EQU     0361H
< FF9A >             18 SEGMTB        EQU     0FF9AH  ;CONVER. NUM. SEGMENTOS
                     19
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     20 ;CONSTANTES
           < 0010 >  21 PLUS  EQU     10H     ;CODIGO DE TECLA '+'
           < 0012 >  22 FIN   EQU     12H     ;CODIGO DE TECLA 'FIN'
                     23
                     24 ;
                     25         ORG     0F523H
                     26
                     27
                     28
                     29 ;*** BLOQUE ADQUISICION-PRESENTACION ***
F523                 30 START
F523 D8              31         CLD
F524 20FD48          32         JSR     CLRDIS  ;BORRA DISPLAY
F527 8E0361          33         STX     PUNTO
F52A 8602            34         STX     BUFPUN  ;LUGAR DE ENTRADA
F52C 8600            35         STX     NUML    ;BORRA NUMERO
F52E 8601            36         STX     NUMH
F530                 37 STR1
F530 20FE09          38         JSR     GETKEY  ;OBTEN DIGITO
F533 90FB            39         BCC     STR1    ;NO HAY
F535 C912            40         CMP     #FIN    ;FIN DE DATOS?
F537 F024            41         BEQ     PRESEN  ;SI, A PRESENTACION
F539 A602            42         LDX     BUFPUN
F53B E005            43         CPX     #5      ;DEMASIADOS DIGITOS?
F53D 9005            44         BCC     STR2    ;NO
F53F A9FF            45         LDA     #0FFH   ;SI, ERROR
F541 4CF7D3          46         JMP     ERROR
F544                 47 STR2
F544 C90A            48         CMP     #9+1
F546 B0E8            49         BCS     STR1    ;NO VALIDA, IGNORA
F548 48              50         PHA             ;SALVA DIGITO
F549 AA              51         TAX
F54A BDFF9A          52         LDA     SEGMTB,X        ;PASO A SEGMENTOS
F54D A602            53         LDX     BUFPUN
F54F 9D0353          54         STA     DISBUF,X        ;SACA A DISPLAY
F552 E602            55         INC     BUFPUN
F554 68              56         PLA
F555 0930            57         ORA     #30H    ;PASA A ASCII (NORMALIZACION)
F557 20F575          58         JSR     DECBIN  ;CONVIERTE
F55A 4CF530          59         JMP     STR1    ;Y REPITE
                     60
                     61 ;FIN DEL DATO PRESENTA CONVERSION Y ESPERA TECLA
F55D                 62 PRESEN
F55D 20FD48          63         JSR     CLRDIS
F560 A500            64         LDA     NUML
F562 A401            65         LDY     NUMH    ;OBTEN BINARIO
F564 20FD22          66         JSR     OUTADD  ;SACA A DISPLAY
F567                 67 PRES1
F567 20FE09          68         JSR     GETKEY  ;OBTEN TECLA
F56A 90FB            69         BCC     PRES1
F56C C910            70         CMP     #PLUS
F56E F0B3            71         BEQ     START   ;REPITE
F570 C912            72         CMP     #FIN
F572 D0F3            73         BNE     PRES1   ;NO VALIDA, REPITE
F574 00              74         BRK             ;FINAL
                     75
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      76 ;*** BLOQUE DE CONVERSION ***
                      77 ;ACEPTA UNA CARACTER ASCII EN ACC CORRESPONDIENTE
                      78 ;A UN NUMERO, Y LO CONVIERTE A BINARIO SIN SIGNO
                      79 ;EMPACANDOLO EN NUMH, NUML.
                      80 ;PROVOCA ERROR SI SE SOBREPASA EL MAXIMO
                      81 ;VALOR ADMITIDO (65535)
F575                  82 DECBIN
F575 290F             83       AND     #0FH     ;ANUMERICO
F577 48               84       PHA
F578 0600             85       ASL     NUML
F57A 2601             86       ROL     NUMH     ;ACTUAL *2
F57C B02D             87       BCS     OVFLW    ;ERROR DE OVERFLOW
F57E A501             88       LDA     NUMH
F580 48               89       PHA              ;SALVA *2
F581 A500             90       LDA     NUML
F583 48               91       PHA
F584 0600             92       ASL     NUML
F586 2601             93       ROL     NUMH     ;ACTUAL *4
F588 B021             94       BCS     OVFLW
F58A 0600             95       ASL     NUML
F58C 2601             96       ROL     NUMH     ;ACTUAL *8
F58E B01B             97       BCS     OVFLW
F590 68               98       PLA              ;OBTEN 2 Y SUMA A 8
F591 18               99       CLC              ;ACTUAL POR 10
F592 6500            100       ADC     NUML
F594 8500            101       STA     NUML
F596 68              102       PLA
F597 6501            103       ADC     NUMH
F599 8501            104       STA     NUMH
F59B B00E            105       BCS     OVFLW
F59D 68              106       PLA              ;DIGITO A INCLUIR
F59E 6500            107       ADC     NUML
F5A0 8500            108       STA     NUML     ;COLOCA VALOR EN NUML,NUMH
F5A2 A900            109       LDA     #0
F5A4 6501            110       ADC     NUMH
F5A6 8501            111       STA     NUMH
F5A8 B001            112       BCS     OVFLW
F5AA 60              113       RTS              ;FIN DE CONVERSION
                     114 ;ERROR DE OVERFLOW
                     115 ;ABORTA A MONITOR
F5AB                 116 OVFLW
F5AB A90F            117       LDA     #0FH     ;CODIGO DE ERROR
F5AD 4CF7D3          118       JMP     ERROR
```

En esta práctica se han previsto dos condiciones de error:

- Err FF: Cuando el número tiene más de 5 carácteres.
- Err 0F: Cuando se ha detectado overflow en el número de entrada.

7.14 CRONOMETRO (F5B0H-F637H)

El objeto de esta práctica es la realización de un cronómetro digital basado en el control de interrupciones por parte del temporizador incluído en el circuito 65C22 (IC22).

Incluye dos controles: inicio-paro y puesta a cero, efectuados, respectivamente, por las teclas [+] y [FIN] del teclado.

La presentación de datos se efectúa en el display del Microinstructor MI-650C, siendo los dígitos de DATO los correspondientes a décimas y centésimas de segundo y las de DIRECCION, a minutos y segundos, por lo que se puede obtener un intervalo de presentación de 1 hora como máximo.

Dado que la resolución del cronómetro es de 0.01 segundos, se requiere una base de tiempos de 10 milisegundos. Como, por otra parte, el display debe estar refrescándose continuamente y el tiempo empleado en esta tarea es del mismo orden de longitud que la base de tiempos, el método empleado para efectuar retardos que se ha incluído en otras prácticas de este capítulo no es válido, ya que puede dar lugar a pérdida de cuentas. Por ello, el único método válido es un control por interrupciones.

El programa principal se encarga únicamente de la presentación de datos y análisis del teclado. Cada 10 ms se produce una interrupción y, sea cual sea el estado del programa, se efectúa una actualización de los datos correspondientes al tiempo transcurrido. La detención de la cuenta se efectúa, simplemente, deteniendo el tren de interrupciones. El listado del programa es el siguiente:

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      1 "650X"
                      2 ;*****************
                      3 ;CRONOMETRO DIGITAL
                      4 ;*****************
                      5
                      6 ;VARIABLES
< 0000 >              7 MINUTO        EQU     00      ;MINUTO EN CURSO
< 0001 >              8 SEGUND        EQU     01      ;SEGUNDO
< 0002 >              9 SEGFRC        EQU     02      ;FRACCION DE SEGUNDO
                     10
                     11 ;REGISTROS DE 6522
< A404 >             12 T1CL  EQU     0A404H  ;TIMER 1 LO(BORRADO INTERR)
< A405 >             13 T1CH  EQU     0A405H  ;TIMER 1 HI
< A406 >             14 T1LL  EQU     0A406H  ;TIMER 1 LO
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

          < A40B >  15 ACR   EQU     0A40BH  ;REGISTRO DE CONTROL
          < A40E >  16 IER   EQU     0A40EH  ;PERMISION DE INTERRUPCION
                    17
                    18 ;DATOS DEL MONITOR
          < FD22 >  19 OUTADD        EQU     0FD22H  ;SACA 4 CARACTERES
          < FD32 >  20 OUTDAT        EQU     0FD32H  ;SACA 2 DIGITOS EN DATO
          < FE09 >  21 GETKEY        EQU     0FE09H  ;DISPLAY/TECLADO
          < 0361 >  22 PUNTO EQU     0361H   ;PUNTOS ENCENDIDOS
          < 0378 >  23 IRQUS EQU     0378H   ;VECTOR IRQ DE USUARIO
                    24
                    25 ;CONSTANTES
          < 0010 >  26 PLUS  EQU     10H     ;VALOR DE TECLA '+'
          < 0012 >  27 FIN   EQU     12H     ;VALOR DE TECLA 'FIN'
                    28
                    29 ;PROG
                    30       ORG     0F5B0H
                    31
                    32 ;*** PROGRAMA PRINCIPAL ***
                    33 ;INICIALIZACION
F5B0                34 START
F5B0 78             35       SEI             ;IMPIDE INTERRUPCION
F5B1 A908           36       LDA     #INTERR.AN.00FFH
F5B3 8D0378         37       STA     IRQUS   ;DE ATENCION A IRQ
F5B6 A9F6           38       LDA     #INTERR/256
F5B8 8D0379         39       STA     IRQUS+1
F5BB                40 CLR
F5BB A900           41       LDA     #0      ;INICIA TIEMPO
F5BD 8500           42       STA     MINUTO
F5BF 8501           43       STA     SEGUND
F5C1 8502           44       STA     SEGFRC
F5C3 A910           45       LDA     #10000B
F5C5 8D0361         46       STA     PUNTO   ;SEPARA MINUTO-SEGUNDO
                    47
                    48 ;PRESENTACION DE DATOS
F5C8                49 REFRS
F5C8 A400           50       LDY     MINUTO
F5CA A501           51       LDA     SEGUND
F5CC 20FD22         52       JSR     OUTADD  ;SACA MIN-SEG
F5CF A502           53       LDA     SEGFRC
F5D1 20FD32         54       JSR     OUTDAT  ;Y FRACCION
F5D4 20FE09         55       JSR     GETKEY  ;REFRESCA DISPLAY/OBTEN TECLA
F5D7 90EF           56       BCC     REFRS   ;NO HAY PULSADA
                    57
                    58 ;ATENCION A TECLA PULSADA
F5D9 C912           59       CMP     #FIN            ;FINAL?
F5DB F0DE           60       BEQ     CLR             ;SI, REINICIA TIEMPO
F5DD C910           61       CMP     #PLUS
F5DF D0E7           62       BNE     REFRS           ;NO VALIDA
                    63
                    64 ;PROCESA START-STOP
F5E1 ADA40E         65       LDA     IER
F5E4 C9C0           66       CMP     #0C0H           ;EN CONTEO?
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

F5E6 F018          67       BEQ     STOP            ;SI, DETEN
                   68
                   69 ;PARADO INICIA CONTEO
F5E8 A9C0          70       LDA     #0C0H
F5EA 8DA40E        71       STA     IER             ;PERMITE IRQ EN TIM1
F5ED A940          72       LDA     #40H
F5EF 8DA40B        73       STA     ACR             ;MODO 'OSCILADOR'
F5F2 A910          74       LDA     #10000.AN.00FFH
F5F4 8DA406        75       STA     T1LL
F5F7 A927          76       LDA     #10000/256
F5F9 8DA405        77       STA     T1CH
F5FC 58            78       CLI                     ;PERMITE INTERRUPCION
F5FD 4CF5C8        79       JMP     REFRS           ;Y VUELTA A PRESENTACION
                   80
                   81 ;FUNCIONANDO DETEN CONTEO
F600               82 STOP
F600 A940          83       LDA     #40H
F602 8DA40E        84       STA     IER             ;IMPIDE IRQ POR TM1
F605 4CF5C8        85       JMP     REFRS           ;Y A COMUN
                   86
                   87 ;*** ATENCION A INTERUPCION ***
                   88 ;RUTINA ACTIVADA PERIODICAMENTE (10MS)
                   89 ;POR TIM1, ACTUALIZA LOS VALORES DEL
                   90 ;TIEMPO EN MEMORIA
F608               91 INTERR
F608 48            92       PHA                     ;SALVA ACUMULADOR
F609 ADA404        93       LDA     T1CL            ;BORRA INTERRUPCION
F60C F8            94       SED                     ;MODO DECIMAL
F60D 18            95       CLC
F60E A502          96       LDA     SEGFRC
F610 6901          97       ADC     #1              ;AUMENTA FRACCION
F612 8502          98       STA     SEGFRC
F614 C900          99       CMP     #00
F616 D01E         100       BNE     IRQSAL          ;NO FINAL
                  101
                  102 ;FIN DE FRACCION AUMENTA SEGUNDOS
F618 18           103       CLC
F619 A501         104       LDA     SEGUND
F61B 6901         105       ADC     #1
F61D 8501         106       STA     SEGUND
F61F C960         107       CMP     #60H            ;MAXIMO VALOR?
F621 9013         108       BCC     IRQSAL          ;NO
F623 A900         109       LDA     #0              ;SI REINICIA CONTADOR
F625 8501         110       STA     SEGUND
                  111
                  112 ;FIN DE SEGUNDO, AUMENTA MINUTO
F627 18           113       CLC
F628 A500         114       LDA     MINUTO
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

F62A 6901          115        ADC    #1
F62C 8500          116        STA    MINUTO
F62E C960          117        CMP    #60H
F630 9004          118        BCC    IRQSAL
F632 A900          119        LDA    #0
F634 8500          120        STA    MINUTO          ;REINICIA
                   121
                   122 ;FIN DE INTERRUPCION
F636               123 IRQSAL
F636 68            124        PLA                    ;RECUPERA ACC
F637 40            125        RTI
                   126
```

Se distinguen dos bloques fundamentales: el principal y el de atención a interrupción. El bloque principal se divide en:

- Inicialización: en este bloque se fija el vector de interrupción del monitor para que apunte a la rutina de interrupción de este programa, de forma que, la transferencia de control se efectúe adecuadamente. Se incluye también un grupo de instrucciones para la puestaa '0' de los datos del cronómetro.

- Presentación de datos: en este bloque se recogen de la memoria los datos que tengan en este momento, se convierten a código de segmentos y se almacenan en el lugar adecuado de DISBUF. A continuación se efectúa un ciclo de refresco del display y una lectura del teclado, repitiéndose ininterrumpidamente hasta la detección de tecla pulsada.

- Atención a tecla pulsada: se detecta si la tecla pulsada es [FIN] o [+]. En el primer caso, se efectúa una puesta a cero de los datos y se repite la presentación. En caso de ser [+], se cambia el estado (parado o en marcha) del cronómetro. Esto se efectúa, como antes indicamos, deteniendo el tren de interrupciones, lo cual implica alterar el registro IER.

  Para saber el estado del cronómetro se emplea el dato que hay en el registro IER (C0H si está contando y 80H si está parado), saltando a uno u otro bloque de programa según la operación a efectuar.

  El bloque de puesta en marcha requiere la inicialización del registro IER, del valor de la unidad de cuenta (10 ms) y del modo de funcionamiento (en ACR); tras lo cual permite la aceptación de interrupciones y se pasa a la presentación.

  El bloque de parada implica solamente la alteración del registro IER, de tal manera que, aunque el temporizador sigue activo, no se envía información de IRQ.

El bloque de atención a la interrupción se activa una vez cada 10 ms y su única acción consiste en actualizar adecuadamente los datos del cronómetro. Con el fin de evitar la conversión binario-decimal, estos datos se mantienen en BCD, lo que requiere un modo de trabajo decimal.

La rutina, al ser activada, empieza por guardar el Acumulador en el Stack. A continuación se fija el modo decimal y se actualizan adecuadamente los datos de tiempo. Obsérvese que tanto para segundos como para minutos se ha fijado un límite máximo de 60 (variación de 0 a 59), mientras que para la fracción no hay límite (variación ed 0 a 99).

Por último se recupera el Acumulador del Stack y se retorna al programa principal.

El inicio del programa se encuentra en la etiqueta START. La detención es aconsejable efectuarla mediante la tecla RESET, de manera que se devuelva a su valor original el vector IRQUS, alterado por este programa.

7.15 RELOJ HORARIO CONTINUO EN TIEMPO REAL (F638H - F6E5H)

En esta práctica se estudia el modo de implantar un reloj en tiempo real en el Micoinstructor MI-650C.

Para ello se requiere que el programa cumpla los siguientes requisitos:

- que sea controlado por interrupciones.
- que incluya unas rutinas de inicialización y otra de finalización pero que no afecten al funcionamiento del reloj.
- que emplee zonas de memoria no usadas normalmente.
- que mantenga la mayoría de las prestaciones del Microinstructor.

Para conseguir esta última característica se va a reestructurar la rutina de interrupción de tal manera que sólo sea ejecutada cuando provenga del TIMER 1 del 65C22. En otro caso, se efectuará una transferencia vectorizada del mismo tipo de la que está provista el Microinstructor, con la única diferencia de que el nuevo vector IRQUS estará en una dirección de memoria diferente.

Por último y para estudiar la manera de efectuar la ampliación de funciones en el Microinstructor, vamos a implantar la rutina de inspección como una función en la tecla E.

El listado es el siguiente:

```
OCATION OBJECT CODE LINE     SOURCE LINE

                      1 "650X"
                      2 ;********************
                      3 ;RELOJ EN TIEMPO REAL
                      4 ;********************
                      5 ;
                      6 ;VARIABLES
< 0600 >              7 HORA  EQU     00600H  ;HORA EN CURSO
< 0601 >              8 MINUTO        EQU     00601H  ;MINUTO
< 0602 >              9 SEGUND        EQU     00602H  ;SEGUNDO
< 0603 >             10 COUNT EQU     00603H  ;CONTADOR PARCIAL DE SEGUNDO
< 0604 >             11 IRQAUX        EQU     00604H  ;VECTOR AUXILIAR DE IRQ
                     12
                     13 ;REGISTROS DEL 6522
< A404 >             14 T1CL  EQU     0A404H  ;TIMER 1 LO (BORRADO INTERNO)
< A405 >             15 T1CH  EQU     0A405H  ;TIMER 1 HI
< A406 >             16 T1LL  EQU     0A406H  ;TIMER 1 LO
< A40B >             17 ACR   EQU     0A40BH  ;REGISTRO DE CONTROL
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                  < A40D >  18 IFR   EQU     0A40DH  ;FLAGS DE ORIGEN DE IRQ
                  < A40E >  19 IER   EQU     0A40EH  ;PERMISION DE INTERRUPCION
                            20
                            21 ;DATOS DEL MONITOR
                  < FD22 >  22 OUTADD        EQU     0FD22H  ;SACA 4 CARACTERES
                  < FD32 >  23 OUTDAT        EQU     0FD32H  ;SACA 2 DIGITOS EN DATO
                  < FE09 >  24 GETKEY        EQU     0FE09H  ;ATIENDE DISPLAY/TECLADO
                  < 0361 >  25 PUNTO EQU     00361H  ;PUNTOS ENCENDIDOS
                  < 0378 >  26 IRQUS EQU     00378H  ;VECTOR IRQ DE USUSARIO
                  < 0374 >  27 ESPVEC        EQU     0374H   ;VECTOR DE FUN. ESP.
                  < FEEC >  28 IRQSIS        EQU     0FEECH  ;ATENCION IRQ MONITOR
                            29 ;PROG
                            30        ORG     0F638H
                            31
                            32 ;*** PROGRAMA PRINCIPAL ***
                            33 ;INICIALIZACION
F638                        34 START
F638 78                     35        SEI                     ;IMPIDE INTERRUPCION
F639 A997                   36        LDA     #INTERR.AN.00FFH        ;VECTOR A RUTINA
F63B 8D0378                 37        STA     IRQUS           ;DE ATENCION A IRQ
F63E A9F6                   38        LDA     #INTERR/256
F640 8D0379                 39        STA     IRQUS+1
F643 A9EC                   40        LDA     #IRQSIS.AN.00FFH        ;VEC. IRQ AUX.
F645 8D0604                 41        STA     IRQAUX          ;NUEVA IMPLANTACION
F648 A9FE                   42        LDA     #IRQSIS/256     ;APUNTA AL MISMO LUGAR
F64A 8D0605                 43        STA     IRQAUX+1        ;QUE EL ANTIGUO IRQUS
F64D A972                   44        LDA     #PRESEN.AN.00FFH        ;ALTERA VECTOR
F64F 8D0374                 45        STA     ESPVEC          ;DE AMPLIACION.
F652 A9F6                   46        LDA     #PRESEN/256
F654 8D0375                 47        STA     ESPVEC+1
F657 A9C0                   48        LDA     #0C0H           ;INICIA 6522
F659 8DA40E                 49        STA     IER
F65C A940                   50        LDA     #40H
F65E 8DA40B                 51        STA     ACR
F661 A950                   52        LDA     #50000.AN.00FFH
F663 8DA406                 53        STA     T1LL
F666 A9C3                   54        LDA     #50000/256
F668 8DA405                 55        STA     T1CH
F66B A914                   56        LDA     #20
F66D 8D0603                 57        STA     COUNT           ;CONTADOR PARCIAL
F670 58                     58        CLI                     ;PERMITE INTERRUPCION
F671 60                     59        RTS                     ;RETORNO
                            60
                            61 ;*** PRESENTACION DE DATOS ***
F672                        62 PRESEN
F672 C90E                   63        CMP     #0EH    ;TECLA 'E'?
F674 F002                   64        BEQ     RFROK   ;SI, VALIDO
F676 38                     65        SEC             ;FLAG DE ERROR
F677 60                     66        RTS             ;FINAL
                            67
                            68 ;ENTRADA VALIDA DE PRESENTACION
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

F678                 69 RFROK
F678 A910            70       LDA     #10000B ;PUNTO DE SEPARACION
F67A 8D0361          71       STA     PUNTO
F67D AC0600          72       LDY     HORA
F680 AD0601          73       LDA     MINUTO
F683 20FD22          74       JSR     OUTADD
F686 AD0602          75       LDA     SEGUND
F689 20FD32          76       JSR     OUTDAT
F68C 20FE09          77       JSR     GETKEY
F68F 90E7            78       BCC     RFROK   ;NO PULSADA
F691 C912            79       CMP     #12H    ;CODIGO DE 'FIN'
F693 D0E3            80       BNE     RFROK   ;NO
F695 18              81       CLC              ;FLAG DE CORRECTO
F696 60              82       RTS              ;SI, FIN DE FUNCION
                     83
                     84 ;*** RUTINA DE INTERRUPCION ***
F697                 85 INTERR
F697 2CA40D          86       BIT     IFR     ;PROCEDE DE TIMI?
F69A 7003            87       BVS     IRQCLK  ;SI,ATIENDE RELOJ
                     88
                     89 ;OTRA CAUSA A EJECUCION DE USUARIO
F69C 6C0604          90       JMP     [IRQAUX]
                     91
                     92 ;INTERRUPCION DE RELOJ, PROCESA DATOS EN RAM
F69F                 93 IRQCLK
F69F 48              94       PHA
F6A0 ADA404          95       LDA     T1CL    ;BORRA INTERR
F6A3 CE0603          96       DEC     COUNT   ;20 PASADAS (1.SEG)?
F6A6 D03C            97       BNE     IRQOUT  ;NO,FINAL
F6A8 A914            98       LDA     #20     ;SI, REINICIA
F6AA 8D0603          99       STA     COUNT
F6AD F8             100       SED
F6AE 18             101       CLC
F6AF AD0602         102       LDA     SEGUND
F6B2 6901           103       ADC     #1
F6B4 8D0602         104       STA     SEGUND  ;AUMENTA SEGUNDOS
F6B7 C960           105       CMP     #60H    ;FINAL?
F6B9 9029           106       BCC     IRQOUT  ;NO, FINAL
F6BB A900           107       LDA     #0
F6BD 8D0602         108       STA     SEGUND  ;SI,REINICIA
F6C0 18             109       CLC
F6C1 AD0601         110       LDA     MINUTO
F6C4 6901           111       ADC     #1
F6C6 8D0601         112       STA     MINUTO
F6C9 C960           113       CMP     #60H
F6CB 9017           114       BCC     IRQOUT
```

```
LOCATION OBJECT CODE LINE    SOURCE LINE

    F6CD A900        115      LDA    #0
    F6CF 8D0601      116      STA    MINUTO
    F6D2 18          117      CLC
    F6D3 AD0600      118      LDA    HORA
    F6D6 6901        119      ADC    #1
    F6D8 8D0600      120      STA    HORA
    F6DB C924        121      CMP    #24H    ;FIN DEL DIA
    F6DD 9005        122      BCC    IRQOUT
    F6DF A900        123      LDA    #0
    F6E1 8D0600      124      STA    HORA
    F6E4             125 IRQOUT
    F6E4 68          126      PLA            ;FIN,RECUPERA ACC
    F6E5 40          127      RTI
                     128
                     129
```

Vemos que el programa es una versión similar a la del empleado en la práctica anterior, repitiendo incluso bloques de instrucciones.

Hemos observado que al pulsar en reposo una de las teclas [E], [F], [+] ó [-], se producen unos errores con números 0EH, 0FH, 10H y 11H, respectivamente. Internamente, el Microinstructor, al no encontrar una función para asignar a estas teclas, efectúa un salto vectorizado por ESPVEC y con el código citado en el Acumulador. Este vector originalmente apunta a la rutina ERROR. La ampliación de funciones se efectúa alterando este vector para que apunte a una rutina de usuario, en la que se vea si la tecla pulsada es la deseada, comparando el valor del Acumulador con el deseado (en nuestro caso E); si así es, se transfiere el control a la rutina adecuada y en caso contrario se remite a ERROR.

Para la puesta en marcha del programa efectuar un [GO] a START. Los datos de hora se pueden alterar colocando los adecuados en HORA, MINUTO y SEGUNDO. Esta inicialización puede hacerse antes de lanzar el programa.

La inspección del programa se efectúa mediante la tecla [E] y la salida mediante la tecla [FIN].

No pulsar las teclas [RESET], [STOP/BRK], [RUN/STEP] pues producen el paro del reloj.

7.16 ORGANO MUSICAL DE UNA OCTAVA

En esta práctica se va a efectuar la generación de sonidos de frecuencia variable mediante el temporizador del 65C22 (IC22).

El temporizador 1 del 65C22 puede seleccionarse para funcionar en modo "free-run" (oscilador), colocando a '1' el bit 6 del ACR. Colocando además a '1' el bit 7 de este registro, se controla la salida PB7 del 65C22, de tal manera que se invierte automáticamente su nivel de salida cada vez que la cuenta llega a '0'. El valor del temporizador se corresponde, por tanto, con el semiperiodo de la señal de salida que será una señal cuadrada con un ciclo de trabajo del 50%.

Esta cualidad vamos a emplearla para simular un órgano electrónico de una octava. La selección de cada nota se efectúa mediante el teclado (teclas 0 a 7). La tecla [0] corresponde al Do y la tecla [7] corresponde al Do de la siguiente escala. El procedimiento de lectura del teclado se ha alterado, de tal manera que el tono se produzca mientras esté pulsada la tecla (para cambiar de nota hace falta soltar la anterior). Al mismo tiempo que se genera el tono, aparece en el display el nombre de la nota musical pulsada.

La salida de señal se obtiene en el conector J2 del Microinstructor, entre las líneas 29-30 de J2 y la masa. Si no se dispone de un amplificador se puede emplear el siguiente circuito:

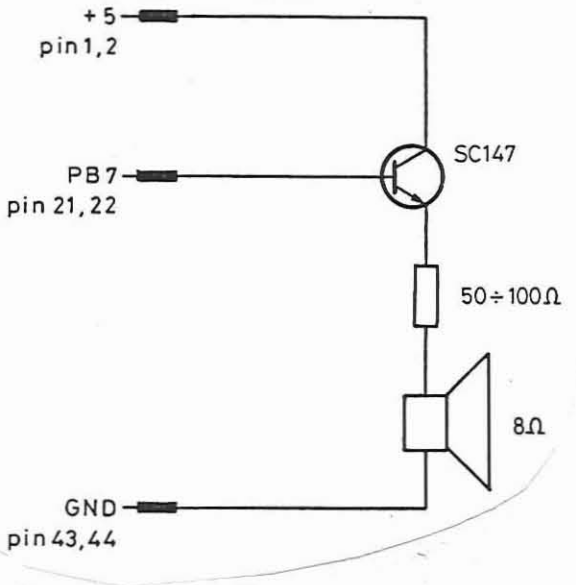


Figura 7.1. Circuito amplificador

Veamos, por último, como se crea la tabla de frecuencias para cada una de las notas. En cierta escala, las frecuencias de las diferentes notas (en Hz) son:

Do = 1046.50
Re = 1174.66
Mi = 1318.51
Fa = 1396.91
Sol= 1567.98
La = 1760.00
Si = 1975.53
Do = 2093.00

El valor que se ha de colocar en el temporizador es el correspondiente al semiperiodo de estas notas. Por ejemplo, para el primer Do el calculo será el siguiente:

Frecuencia = 1046.50

Periodo = 955.5 μs

Semiperiodo = 477.7

El programa propuesto es el siguiente:

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      1 "650X"
                      2 ;**********************
                      3 ; ORGANO DE UNA OCTAVA
                      4 ;**********************
                      5 ;
                      6 ;
                      7 ;!!REQUIERE AMPLIACION DE HARDWARE!!
                      8 ;(VER TEXTO EN EL MANUAL)
                      9 ;
                     10 ;REGISTROS DEL 6522
                     11 ;
< A400 >             12 DDRB  EQU     0A400H
< A402 >             13 RB    EQU     0A402H
< A40B >             14 ACR   EQU     0A40BH          ;REGISTRO DE CONTROL
< A406 >             15 T1L   EQU     0A406H
< A405 >             16 T1H   EQU     0A405H          ;TEMPORIZADOR
                     17 ;
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      18 ;DATOS DEL MONITOR
                      19 ;
          < FD48 >    20 CLRDIS       EQU     0FD48H  ;BORRADO DE DISPLAY
          < FE4D >    21 REFDIS       EQU     0FE4DH  ;REFRESCO DEL DISPLAY
          < 0353 >    22 DISBUF       EQU     353H    ;BUFFER DE DISPLAY
          < 0361 >    23 PUNTO        EQU     361H    ;PUNTOS DEL DISPLAY
                      24 ;
                      25       ORG     400H
                      26 ;
0400                  27 START
0400 A980             28       LDA     #80H
0402 8DA400           29       STA     DDRB            ;PB7 A SALIDA
                      30 ;
                      31 ;BUCLE DE SILENCIO
                      32 ;
0405                  33 SILENC
0405 20FD48           34       JSR     CLRDIS          ;BORRA DISPLAY
0408 8E0361           35       STX     PUNTO           ;APAGADOS DISPLAY
040B A900             36       LDA     #0
040D 8DA40B           37       STA     ACR             ;DETEN SONIDO
0410 8DA402           38       STA     RB              ;PB7 A CERO
                      39 ;
                      40 ;BUCLE DE SILENCIO ESPERA TECLA
                      41 ;
0413                  42 SILBUC
0413 20FE4D           43       JSR     REFDIS
0416 90FB             44       BCC     SILBUC          ;TECLA NO PULSADA
0418 290F             45       AND     #0FH
041A C908             46       CMP     #08H            ;VALIDA?
041C B0F5             47       BCS     SILBUC          NO, REPITE
                      48 ;
                      49 ;TECLA VALIDA, INICIA SONIDO
                      50 ;
041E 0A               51       ASL     A               ;*2
041F AA               52       TAX                     ;PUNTERO A TABLAS
0420 BD0445           53       LDA     >TEXNOT,X       ;TEXTO DE LA NOTA
0423 8D0353           54       STA     DISBUF          ;A BUFFER DE DISPLAY
0426 BD0446           55       LDA     >TEXNOT+1,X
0429 8D0354           56       STA     DISBUF+1
042C BD0455           57       LDA     >TABNOT,X       ;FRECUENCIA DE LA NOTA
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

   042F 8DA406          58        STA     T1L
   0432 BD0456          59        LDA     >TABNOT+1,X
   0435 8DA405          60        STA     T1H
   0438 A9C0            61        LDA     #0C0H
   043A 8DA40B          62        STA     ACR             ;INICIA SONORO
                        63 ;
                        64 ;BUCLE DE SONIDO, ESPERA A FIN DE TECLA
                        65 ;
   043D                 66 NOTBUC
   043D 20FE4D          67        JSR     REFDIS
   0440 B0FB            68        BCS     NOTBUC          ;PULSADA
   0442 4C0405          69        JMP     SILENC          ;NO, REPITE TODO
                        70
                        71        ;TABLAS DE DATOS
                        72        ;TEXTO DE LAS NOTAS
                        73
                        74 ;
                        75        ORG     445H
                        76 ;
   0445                 77 TEXNOT
   0445 5E5C            78        BYTE    05EH,05CH       ;DO
   0447 5079            79        BYTE    50H,79H         ;RE
   0449 3730            80        BYTE    37H,30H         ;MI
   044B 7177            81        BYTE    71H,77H         ;FA
   044D 6D5C            82        BYTE    6DH,5CH         ;SOL
   044F 3877            83        BYTE    38H,77H         ;LA
   0451 6D30            84        BYTE    6DH,30H         ;SI
   0453 5E5C            85        BYTE    5EH,5CH         ;DO
                        86
                        87 ;VALORES DE TIEMPO DE LAS NOTAS
                        88 ;
   0455                 89 TABNOT
   0455 01DE            90        WORD    478D
   0457 01AA            91        WORD    426D
   0459 017B            92        WORD    379D
   045B 0166            93        WORD    358D
   045D 013F            94        WORD    319D
   045F 011C            95        WORD    284D
   0461 00FD            96        WORD    253D
   0463 00EF            97        WORD    239D
```

7.17 RECOLOCADOR (F334H - F3C9H)

Cuando estamos diseñando un programa, no es extraño que sea necesario tener que abrir sitio para colocar instrucciones nuevas, o bien para modificar algunas de las existentes. Dado que en el Microinstructor no se dispone de un ensamblador, esta operación requiere normalmente la reescritura de todo el programa (o al menos parte de él), con el correspondiente cálculo de nuevos offsets, direcciones de tablas y saltos, etc, tarea generalmente tediosa.

Este programa efectúa estas operaciones automáticamente. Una vez suministrados los datos del movimiento, calcula y altera todos los datos que se ven variados por el desplazamiento.

Téngase en cuenta que durante el análisis del programa a alterar, el recolocador interpreta todos los bytes como si fueran un programa, por lo que se debe prestar especial atención si existen tablas intermedias, ya que pueden ser alteradas. En este caso será necesario efectuar el trabajo en dos o tres etapas, separando dichas tablas del resto del programa.

El recolocador no mueve los datos del programa, sino que solamente altera los operandos de las instrucciones que se han de modificar si se efectuara este movimiento. Por tanto es necesario emplear otro programa que haga este trabajo, para ello es idóneo el programa para el movimiento de datos entre zonas variables, que es directamente ejecutable en la dirección F770H.

Puede ser un ejercicio interesante unir ambos programas para que realicen de una sola vez la totalidad del trabajo.

Para su correcto funcionamiento el programa requiere los siguientes datos:

- Dónde comienza el programa, es decir, la dirección de la primera instrucción del programa (incluyendo la parte que no se mueve). Es necesario analizar la totalidad del programa, ya que pueden existir en cualquier punto del mismo referencias a instrucciones situadas dentro del bloque movido y que, por tanto, han de ser alteradas. Colocar esta dirección en STRPRG y STRPRG+1 en forma de parte baja-parte alta, como es normal.

- Dónde termina el programa. En este caso, en vez de un vector se emplea un byte especial de control. Colocar el byte FFH a continuación de la última instrucción del programa. Este dato puede ser eliminado cuando se termine el trabajo.

- Dónde comienza la recolocación, esto es, la primera dirección que deseamos mover en el programa (si estamos moviendo la

totalidad del programa, esta dirección coincidirá con STRPRG). Colocar el valor correspondiente en STRREC y STRREC+1.

- A qué distancia desde STRREC se ha de mover el programa. Este dato se considera un incremento y será un número binario en complemento a 2. Por ejemplo, si deseamos dejar sitio para 3 bytes en el programa el incremento será 0003; si deseamos eliminar 4 bytes del mismo, será FFFCH (-4); si deseamos mover todo el programa desde 400H a 600H, será 200H, etc. Colocar el valor del incremento en ADJUST y ADJUST+1.

- Página límite a partir de la cual se ignora la recolocación. Si el programa a alterar incluye referencias al circuito I/O, este límite deberá ser A0H (la zona de I/O empieza a partir de A000H). Por las características del hardware del Microinstructor, este valor nunca será superior a 100H y, por tanto, se considerará como dato fijo del programa. En caso de ampliar la capacidad de memoria o para aplicaciones especiales, cambiar en el programa la instrucción correspondiente, debidamente señalizada en los comentarios del listado.

Una vez introducidos estos datos, ejecutar el programa a partir de RELOC.

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      1 "650X"
                      2 ; **********************
                      3 ;  R E C O L O C A D O R
                      4 ; **********************
                      5 ;
                      6 ;
               < 0010 > 7 PAGLIM       EQU     10H     ;PAGINA LIMITE
                      8       ORG     0000H
                      9 ;
                     10 ;
               < 0000 > 11 ADJUST       EQU     0H      ;DISTANCIA A AJUSTAR
               < 0002 > 12 STRPRG       EQU     2H      ;INICIO DEL PROGRAMA
               < 0004 > 13 STRREC       EQU     4H      ;INICIO ZONA A RECOLOCAR
               < 0006 > 14 TEMP  EQU    6H      ;SCRATCH
               < 0007 > 15 LIMIT EQU    7H
                     16 ;
                     17 ;
                     18       ORG     0F334H
   F334              19 RELOC
   F334 D8           20       CLD
   F335 A000         21       LDY     #00
   F337 B102         22       LDA     [STRPRG],Y      ;OBTEN OPCODE
   F339 A8           23       TAY                     ;Y SALVA
   F33A A207         24       LDX     #07
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

F33C                25 LOOP
F33C 98             26       TYA                    ;RETOMA OPCODE
F33D 3DF3B3         27       AND     >TAB1-1,X      ;ELIMINA BITS NO SIGNIF.
F340 5DF3BA         28       EOR     >TAB2-1,X      ;Y MIRA LOS RESTANTES
F343 F003           29       BEQ     FOUND
F345 CA             30       DEX                    ;NO COINCIDE, REPITE
F346 D0F4           31       BNE     LOOP           ;HASTA FINAL DE TABLA
F348                32 FOUND
F348 BCF3C2         33       LDY     >TAB3,X        ;OBTEN LONG. O FLAG
F34B 300E           34       BMI     TRIPLE
F34D F023           35       BEQ     BRANCH
F34F                36 SKIP
F34F E602           37       INC     <STRPRG        ;PASA A PROXIMO OPCODE
F351 D002           38       BNE     SKIP1
F353 E603           39       INC     <STRPRG+1
F355                40 SKIP1
F355 88             41       DEY
F356 D0F7           42       BNE     SKIP
F358 F0DA           43       BEQ     RELOC          ;REPITE OPERACION
F35A                44 END
F35A 00             45       BRK                    ;FIN DEL PROGRAMA
                    46 ;
                    47 ;LONGITUD 3 O ILEGAL
F35B                48 TRIPLE
F35B C8             49       INY
F35C 30FC           50       BMI     END            ;ILEGAL (O FINAL),SALIDA
F35E C8             51       INY
F35F B102           52       LDA     [STRPRG],Y     ;LO DE OPERANDO
F361 AA             53       TAX
F362 C8             54       INY
F363 B102           55       LDA     [STRPRG],Y     ;HI DE OPERANDO
F365 20F39E         56       JSR     AJUSTA         ;AJUSTA SI NECESARIO
F368 9102           57       STA     [STRPRG],Y     ;Y REPON NUEVO O
F36A 88             58       DEY
F36B 8A             59       TXA
F36C 9102           60       STA     [STRPRG],Y
F36E A003           61       LDY     #3             ;LONGITUD
F370 D0DD           62       BNE     SKIP
                    63 ;
                    64 ;BRANCH
                    65 ;DETERMINA ADDRS ORIGEN Y DESTINO
F372                66 BRANCH
F372 C8             67       INY                    ;=1
F373 A602           68       LDX     STRPRG
F375 A503           69       LDA     STRPRG+1       ;ADDRS  ORIGEN
F377 20F39E         70       JSR     AJUSTA         ;AJUSTA VALOR
F37A 8606           71       STX     TEMP           ;SALVA LO AJUSTADO
F37C A2FF           72       LDX     #0FFH
F37E B102           73       LDA     [STRPRG],Y     ;OBTEN OPERANDO (OFFSET)
F380 18             74       CLC
F381 6902           75       ADC     #2             ;AJUSTA OFFSET
F383 3001           76       BMI     OVER           ;NO,ALTERA FLAG =0
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

   F385 E8           77       INX
   F386              78 OVER
   F386 8607         79       STX     LIMIT
   F388 18           80       CLC
   F389 6502         81       ADC     STRPRG          ;CALCULA DESTINO
   F38B AA           82       TAX
   F38C A507         83       LDA     LIMIT           ;00-$FFH
   F38E 6503         84       ADC     STRPRG+1        ;PARTE ALTA
   F390 20F39E       85       JSR     AJUSTA
   F393 CA           86       DEX
   F394 CA           87       DEX                     ;RECALCULA OFFSET
   F395 8A           88       TXA
   F396 38           89       SEC
   F397 E506         90       SBC     TEMP
   F399 9102         91       STA     [STRPRG],Y      ;Y COLOCA EN OPERANDO
   F39B C8           92       INY                     ;=2
   F39C D0B1         93       BNE     SKIP            ;A POR OTRA
                     94 ;
                     95 ;EXAMINA DIRECCION Y AJUSTA SI NECESARIO
                     96 ;
   F39E              97 AJUSTA
   F39E C510         98       CMP     PAGLIM          ;EN ZONA VALIDA?
   F3A0 B011         99       BCS     OUT             ;NO,FINAL
   F3A2 C505        100       CMP     STRREC+1        ;EN ZONA MOVIDA?
   F3A4 D002        101       BNE     TES2            ;(LIMITE INFERIOR)
   F3A6 E404        102       CPX     STRREC
   F3A8             103 TES2
   F3A8 9009        104       BCC     OUT             ;NO,FINAL
   F3AA 48          105       PHA
   F3AB 8A          106       TXA                     ;ALTERA ADDRS CON VALOR EN ADJU
   F3AC 18          107       CLC
   F3AD 6500        108       ADC     ADJUST          ;LO
   F3AF AA          109       TAX
   F3B0 68          110       PLA
   F3B1 6501        111       ADC     ADJUST+1        ;HI
   F3B3             112 OUT
   F3B3 60          113       RTS
                    114 ;
                    115 ;TABLAS DE IDENTIFICACION DE OPCODE
                    116 ;MASCARA
                    117 ;
   F3B4 0C1F0D871F  118 TAB1  BYTE    0CH,1FH,0DH,87H,1FH,0FFH,03H
                    119 ;
                    120 ;COMPARACION
                    121 ;
   F3BB 0C19080010  122 TAB2  BYTE    0CH,19H,08,00,10H,20H,03
                    123 ;
                    124 ;LONGITUD/FLAG
                    125 ;
   F3C2 02FFFF0101  126 TAB3  BYTE    02,0FFH,0FFH,01,01,00,0FFH,0FEH
```

El retorno se efectúa mediante una instrucción BRK. Una vez devuelto el control, es interesante ver que el puntero en STRPRG apunta al terminador (FFH) incluído al final del programa, prueba de que la operación ha sido correctamente realizada, ya que la detección de un operando erróneo da lugar también a una detención.

> **NOTA:** Es interesante, máxime en las primeras veces en que se emplee, el disponer de una copia del programa en cinta para evitar su destrucción si se ha cometido algún error en la introducción de los datos anteriores.

7.18 JUEGO ADIVINE EL NUMERO (F6E6H-F76FH)

Como última práctica de este manual, vamos a incluir un juego. El Microinstructor selecciona un número aleatorio entre 01 y 98 que el jugador debe tratar de adivinar.

Admite dos modalidades de juego:

- Con un jugador: debe tratar de adivinar el número en un máximo de 6 jugadas.

- Varios jugadores: cada jugador trata de evitar introducir el número correcto. El jugador que adivine el número queda fuera de juego. Esto se repite con los restantes jugadores hasta que quede uno sólo.

Modo de utilización:

- Ejecutar el programa a partir de START.

- El display muestra inicialmente los límites (00 y 99) entre los que puede estar el número.

- Introducir un número decimal de dos dígitos comprendido entre estos márgenes. En este momento el display cambia y muestra los números límites válidos. Por ejemplo, si el número secreto es 45 y se pulsa 33, el display mostrará 3399. En el dígito de la derecha aparece el número de jugadas efectuadas hasta el momento.

- Repetir la introducción hasta que el número sea correcto. Cuando esto suceda, el número aparace repetido en el display de límites (en el caso anterior 4545) y no se aceptan más entradas.

- Pulsar [+] para iniciar otra jugada y [STOP/BRK] para finalizar el juego.

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      1 "650X"
                      2 ;**********************
                      3 ;JUEGO ADIVINE EL NUMERO
                      4 ;**********************
                      5       ORG     0
                      6 ;VARIABLES
               < 0000 > 7 NUMINF        EQU     00H
               < 0001 > 8 NUMSUP        EQU     01H
               < 0002 > 9 JUGADA        EQU     02H
               < 0003 > 10 RNDM  EQU    03H
               < 0004 > 11 GUESS EQU    04H
                      12
                      13 ;DATOS DEL MONITOR
               < FD22 > 14 OUTADD       EQU     0FD22H
               < FD32 > 15 OUTDAT       EQU     0FD32H
               < FE09 > 16 GETKEY       EQU     0FE09H
               < 0361 > 17 PUNTO EQU    0361H
                      18
                      19 ;PROG
                      20       ORG     0F6E6H
                      21
  F6E6                22 START
  F6E6 A000           23       LDY     #00     ;INICIALIZA DATOS
  F6E8 8C0361         24       STY     PUNTO
  F6EB 8400           25       STY     NUMINF  ;FIJA MARGENES
  F6ED A999           26       LDA     #99H
  F6EF 8501           27       STA     NUMSUP
  F6F1 20FD22         28       JSR     OUTADD  ;SACA EXTREMOS
  F6F4 A900           29       LDA     #0
  F6F6 8502           30       STA     JUGADA
  F6F8 20FD32         31       JSR     OUTDAT  ;NUMERO DE INTENTOS
                      32
                      33 ;GENERACION ALEATORIA DEL NUMERO (SECRETO)
  F6FB A503           34 GRNDM LDA     RNDM    ;NUMERO SECRETO
  F6FD F8             35       SED
  F6FE 6901           36       ADC     #1      ;AUMENTA HASTA TECLA
  F700 A201           37       LDX     #1
  F702 C999           38       CMP     #99H    ;FINAL?
  F704 D001           39       BNE     GNRD1   ;NO
  F706 8A             40       TXA
  F707                41 GNRD1
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

  F707 8503          42       STA     RNDM
  F709 D8            43       CLD
  F70A 20FE09        44       JSR     GETKEY  ;MIRA TECLADO
  F70D 90EC          45       BCC     GRNDM   ;NO HAY, REPITE
  F70F C90A          46       CMP     #10     ;SI, VALIDA?
  F711 B0E8          47       BCS     GRNDM   ;NO
                     48
                     49 ;INICIO DEL JUEGO
  F713               50 JUEGO
  F713 0A            51       ASL     A       ;EMPACA DIGITOS
  F714 0A            52       ASL     A
  F715 0A            53       ASL     A
  F716 0A            54       ASL     A
  F717 8504          55       STA     GUESS
  F719               56 JUEGO1
  F719 20FE09        57       JSR     GETKEY  ;SEGUNDO DIGITO
  F71C 90FB          58       BCC     JUEGO1
  F71E C90A          59       CMP     #10
  F720 B0F7          60       BCS     JUEGO1  ;NO VALIDO
  F722 0504          61       ORA     GUESS   ;EMPACA AMBOS DIGITOS
                     62
                     63 ;MIRA SI ES JUGADA VALIDA
  F724 C500          64       CMP     NUMINF
  F726 9027          65       BCC     NEWJUE  ;NO, IGNORA
  F728 C501          66       CMP     NUMSUP
  F72A B023          67       BCS     NEWJUE
                     68
                     69 ;EN MARGENES PROCESA JUGADA
                     70 ;AUMENTA JUGADAS
  F72C 48            71       PHA
  F72D F8            72       SED
  F72E 18            73       CLC
  F72F A502          74       LDA     JUGADA
  F731 6901          75       ADC     #1
  F733 8502          76       STA     JUGADA
  F735 D8            77       CLD
  F736 68            78       PLA
                     79
                     80 ;COMPARA CON SECRETO
  F737 C503          81       CMP     RNDM
  F739 F020          82       BEQ     ACIER   ;FIN DEL JUEGO
  F73B 9004          83       BCC     BAJO
  F73D 8501          84       STA     NUMSUP  ;ALTO
  F73F B002          85       BCS     NEWLIM
  F741               86 BAJO
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

   F741 8500          87       STA     NUMINF
   F743               88 NEWLIM
   F743 A501          89       LDA     NUMSUP  ;SACA NUEVOS MARGENES
   F745 A400          90       LDY     NUMINF
   F747 20FD22        91       JSR     OUTADD
   F74A A502          92       LDA     JUGADA
   F74C 20FD32        93       JSR     OUTDAT  ;SACA VALORES A DISPLAY
                      94
                      95 ;INICIA NUEVA JUGADA
   F74F               96 NEWJUE
   F74F 20FE09        97       JSR     GETKEY
   F752 90FB          98       BCC     NEWJUE
   F754 C90A          99       CMP     #10     ;VALIDA?
   F756 B0F7         100       BCS     NEWJUE
   F758 4CF713       101       JMP     JUEGO   ;SI
                     102
                     103 ;SE ADIVINO EL NUMERO, FIN DE PARTIDA
   F75B              104 ACIER
   F75B A8           105       TAY
   F75C 20FD22       106       JSR     OUTADD  ;SACA EXTREMOS COINCIDENTES
   F75F A502         107       LDA     JUGADA
   F761 20FD32       108       JSR     OUTDAT
   F764              109 ACIER1
   F764 20FE09       110       JSR     GETKEY  ;ESPERA CONTINUACION/FIN
   F767 90FB         111       BCC     ACIER1
   F769 C910         112       CMP     #10H    ;TECLA '+'
   F76B D0F7         113       BNE     ACIER1
   F76D 4CF6E6       114       JMP     START   ;REINICIA
```

APENDICE A

INSTRUCCIONES DEL MICROPROCESADOR 65C02

El conjunto de instrucciones propias del 65C02 consta de 68 mnemónicos distintos que, junto con los diversos modos de direccionamiento posibles, producen un total de 210 códigos de operación distintos.

Esto representa un notable incremento respecto al número de instrucciones de la versión NMOS que dispone de 56 mnemónicos y 151 códigos de operación diferentes.

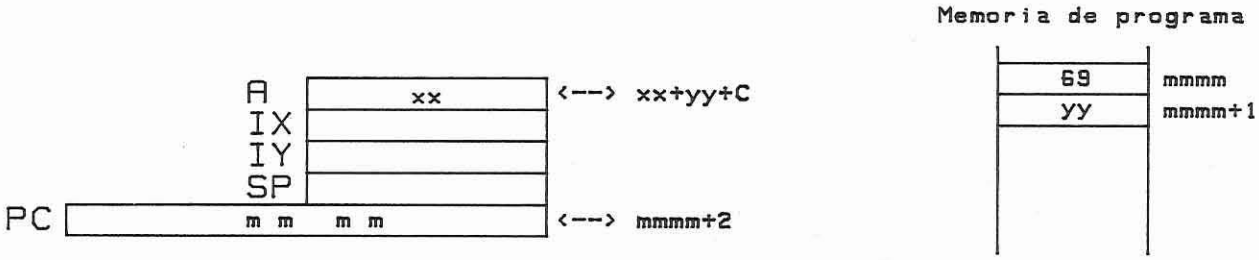
Existe una versión intermedia (65SC02) en la que están disponibles todos los códigos de operación de la versión NMOS más 27 códigos de operación adicionales formando un total de 64 instrucciones.

A continuación se detalla el juego de instrucciones del 65C02 con cada uno de los direccionamientos posibles. Los códigos de operación que sólo están disponibles en la versión 65C02 se indican mediante el símbolo '+'. Los que están permitidos tanto en el 65C02 como en el 65SC02 se indican mediante el símbolo '*'.

Si la ejecución de la instrucción puede alterar algún bit del Registro de Estado ('Status') la casilla correspondiente al flag que puede variar se indica mediante una 'x'. Cuando al ejecutar una instrucción alguno de estos flags toma un valor prefijado se indica mediante un '1' o un '0' en la casilla correspondiente a dicho flag.

ADC #dato — **SUMA CON ACARREO**
(Direccionamiento inmediato)





NUMERO DE CICLOS DE RELOJ:

2 (añadir 1 en modo decimal)

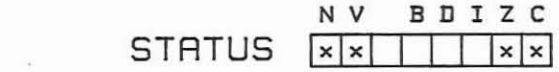
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

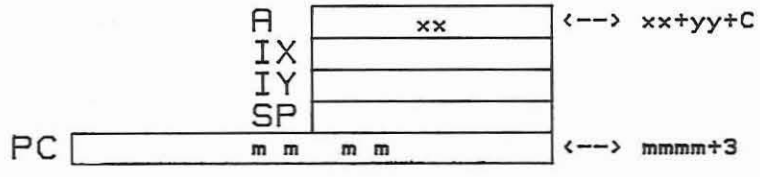
N: Se pone a 1 si el resultado es negativo.
V: Se pone a 1 si se produce "overflow".
Z: Se pone a 1 si el resultado es cero.
C: Se pone a 1 si hay acarreo.

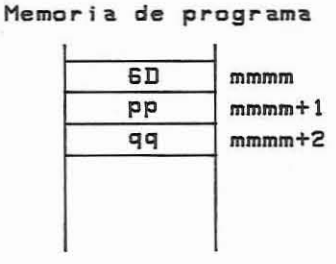
DESCRIPCION:

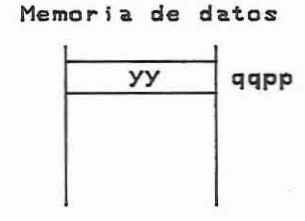
El contenido de la posición de memoria que sigue a la que contiene el código de operación y el bit de Carry del Registro de Status se suman con el Acumulador. El resultado queda en el acumulador.

ADC dir — **SUMA CON ACARREO**
(Direccionamiento absoluto)









NUMERO DE CICLOS DE RELOJ:

4 (añadir 1 en modo decimal)

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

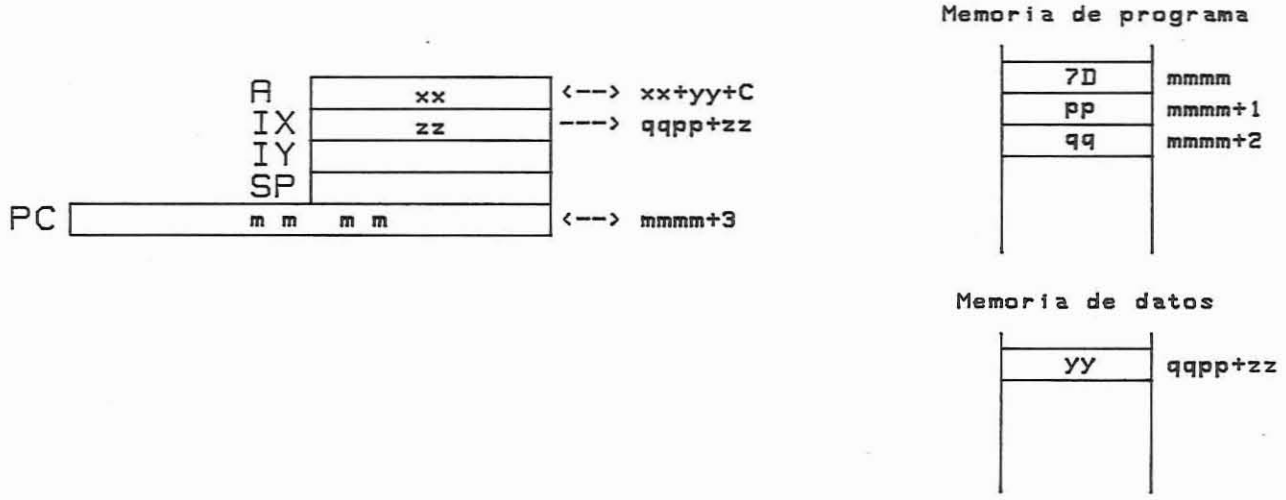
N: Se pone a 1 si el resultado es negativo.
V: Se pone a 1 si se produce "overflow".
Z: Se pone a 1 si el resultado es cero.
C: Se pone a 1 si hay acarreo.

DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria donde se encuentra el dato que se suma con el Acumulador y con el bit de Carry del Registro de Status. El resultado queda en el Acumulador.

ADC dir,X — **SUMA CON ACARREO**
(Direccionamiento absoluto indexado)





NUMERO DE CICLOS DE RELOJ:

4 (añadir 1 en modo decimal)
(añadir 1 si salta de página)

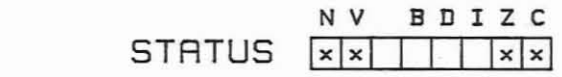
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

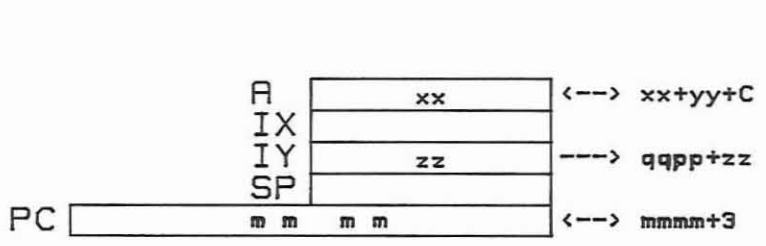
N: Se pone a 1 si el resultado es negativo.
V: Se pone a 1 si se produce "overflow".
Z: Se pone a 1 si el resultado es cero.
C: Se pone a 1 si hay acarreo.

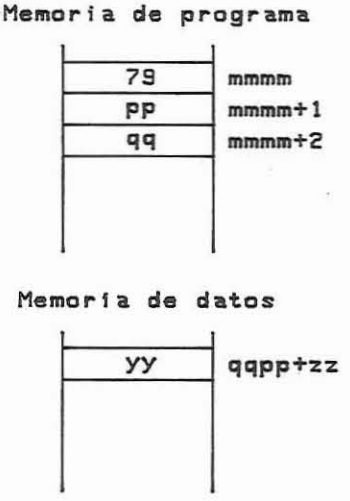
DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria que, sumada al contenido del registro X, indica la posición de memoria donde se encuentra el dato. Este dato se suma con el Acumulador y con el bit de Carry del Registro de Status. El resultado queda en el Acumulador.

ADC dir,Y — SUMA CON ACARREO
(Direccionamiento absoluto indexado)







NUMERO DE CICLOS DE RELOJ:

4 (añadir 1 en modo decimal)
(añadir 1 si salta de página)

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

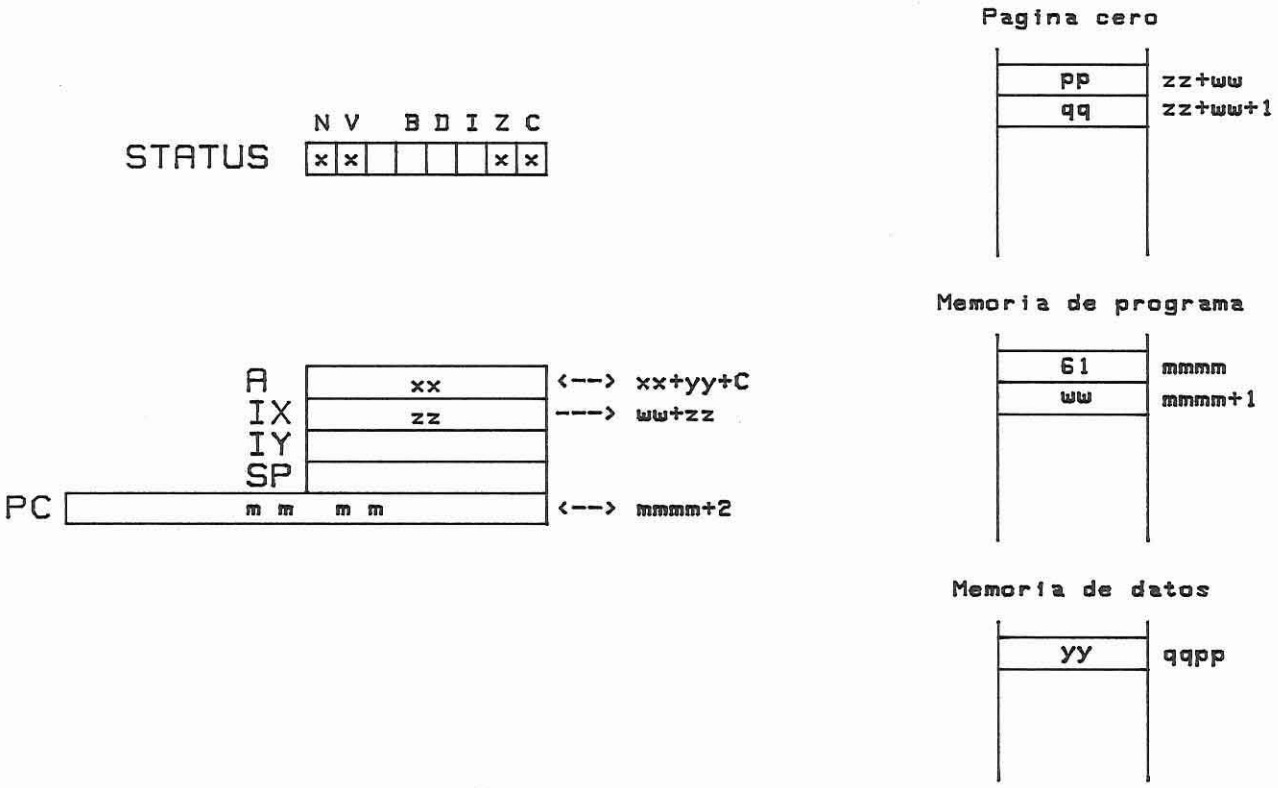
N: Se pone a 1 si el resultado es negativo.
V: Se pone a 1 si se produce "overflow".
Z: Se pone a 1 si el resultado es cero.
C: Se pone a 1 si hay acarreo.

DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria que, sumada al contenido del registro Y, indica la posición de memoria donde se encuentra el dato. Este dato se suma con el Acumulador y con el bit de Carry del Registro de Status. El resultado queda en el Acumulador.

ADC (dir,X)

SUMA CON ACARREO
(Direccionamiento preindexado)



NUMERO DE CICLOS DE RELOJ:

6 (añadir 1 en modo decimal)

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

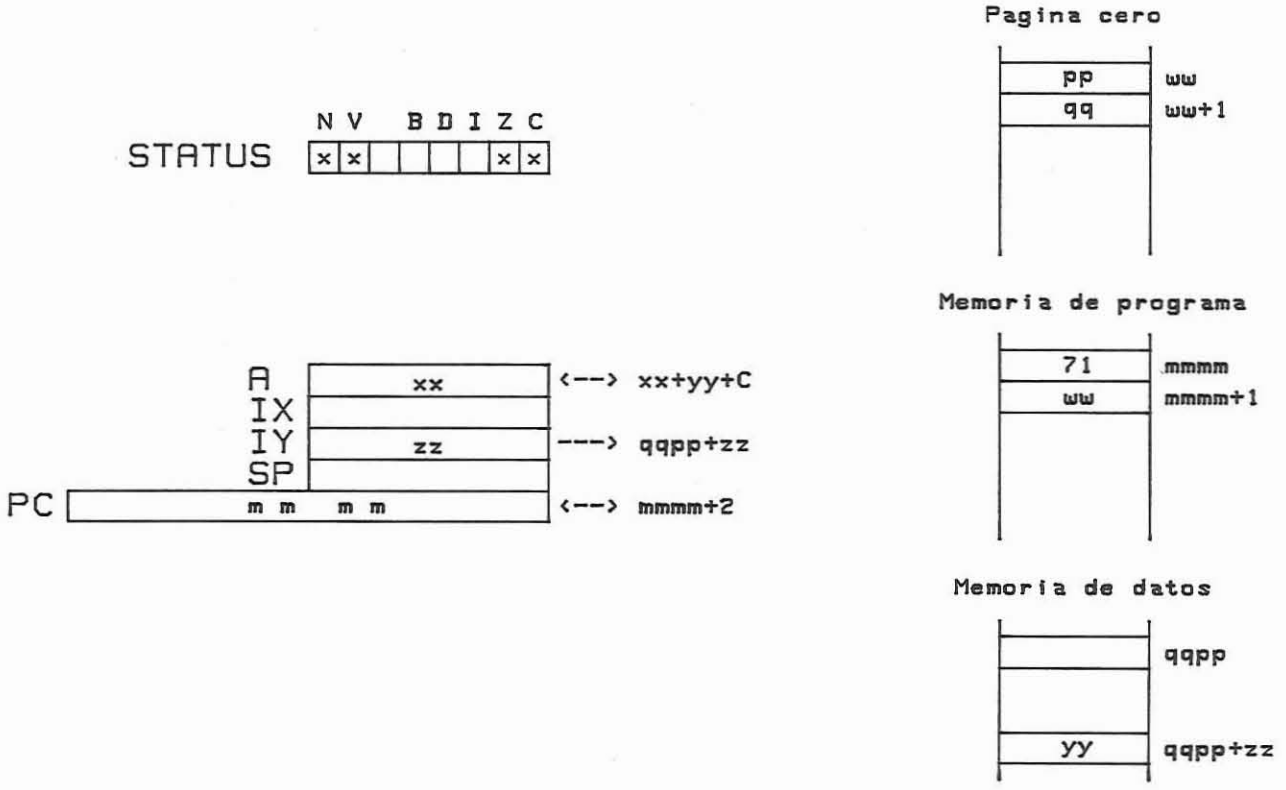
N: Se pone a 1 si el resultado es negativo.
V: Se pone a 1 si se produce "overflow".
Z: Se pone a 1 si el resultado es cero.
C: Se pone a 1 si hay acarreo.

DESCRIPCION:

El contenido de la posición de memoria que sigue a la que contiene el código de operación se suma con el contenido del registro X. El resultado (despreciando el acarreo si se ha producido) corresponde a la dirección de memoria en página cero que contiene la parte baja de la dirección efectiva del dato. La parte alta de la dirección efectiva del dato está en la siguiente posición de memoria en pagina cero. De la dirección efectiva del dato se toma el byte que se suma con el contenido del Acumulador y con el bit de Carry del Registro de Status. El resultado se carga de nuevo en el Acumulador.

ADC (dir),Y

SUMA CON ACARREO
(Direccionamiento postindexado)



NUMERO DE CICLOS DE RELOJ:

5 (añadir 1 en modo decimal)
(añadir 1 si salta de página)

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
V: Se pone a 1 si se produce "overflow".
Z: Se pone a 1 si el resultado es cero.
C: Se pone a 1 si hay acarreo.

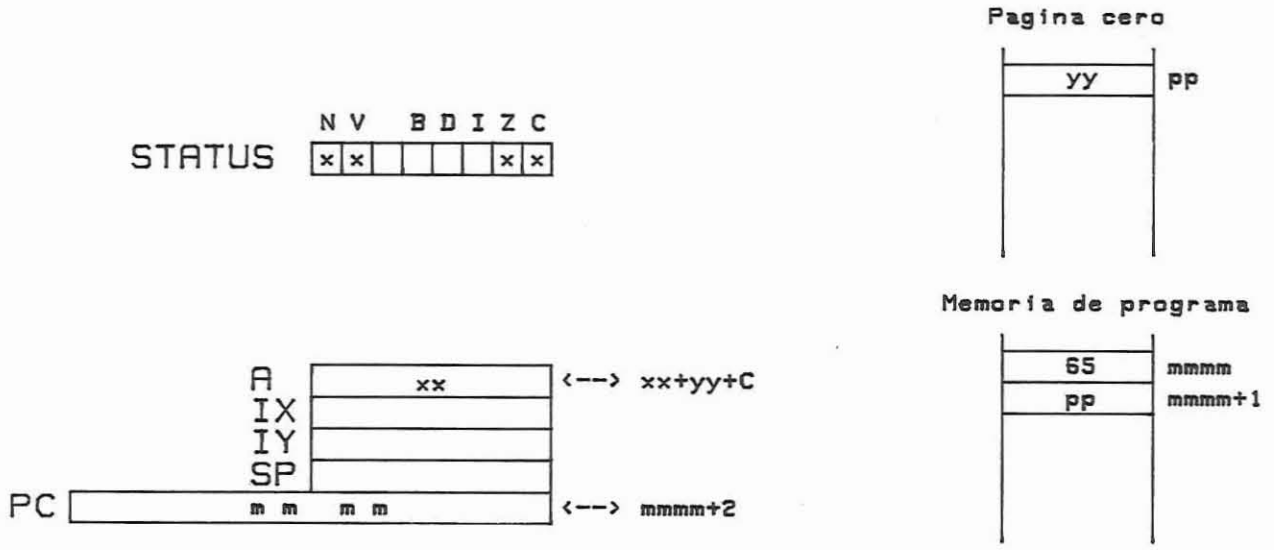
DESCRIPCION:

El contenido de la posición de memoria que sigue a la que contiene el código de operación contiene una dirección de memoria de página cero. En esta dirección y en la siguiente se encuentran, en orden inverso (primero el byte de menor peso y luego el byte mayor peso), los dos bytes que forman la dirección de memoria que sumada al contenido del registro Y, nos dará la dirección de memoria donde se encuentra el dato. Este dato sumado con el acumulador y con el bit de Carry del registro de Status da el resultado que se guarda de nuevo en el Acumulador.

ADC dir

SUMA CON ACARREO

(Direccionamiento página cero)



NUMERO DE CICLOS DE RELOJ:

3 (añadir 1 en modo decimal)

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

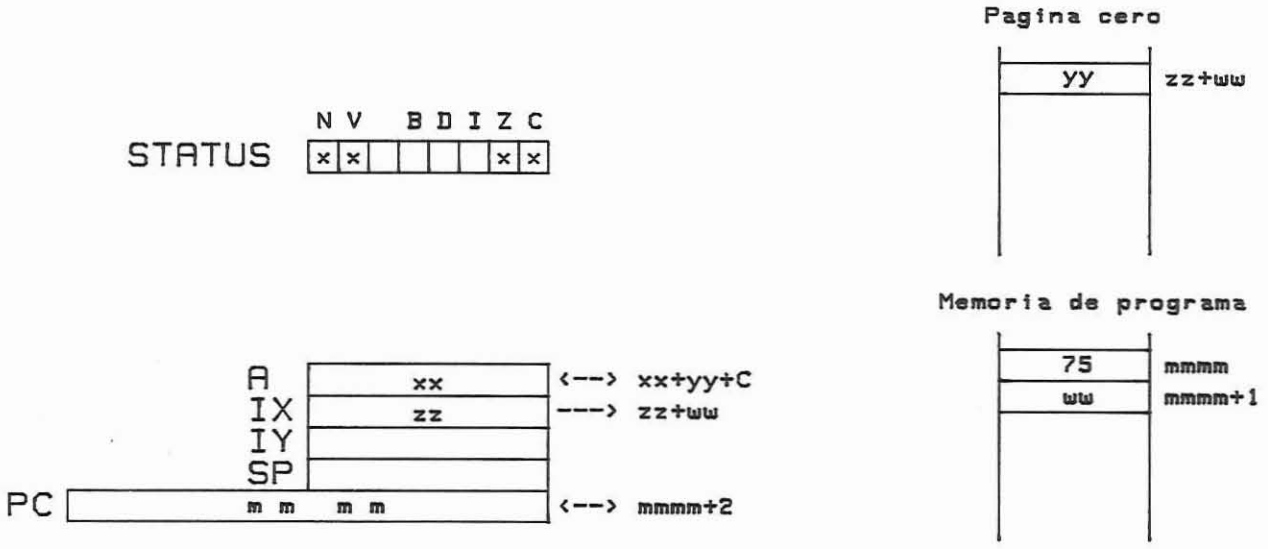
N: Se pone a 1 si el resultado es negativo.
V: Se pone a 1 si se produce "overflow".
Z: Se pone a 1 si el resultado es cero.
C: Se pone a 1 si hay acarreo.

DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la dirección de memoria de página cero donde se encuentra el dato que se suma con el Acumulador y con el bit de Carry del Registro de Status. El resultado queda en el Acumulador.

ADC dir,X — **SUMA CON ACARREO**

(Direccionamiento página cero indexado)



NUMERO DE CICLOS DE RELOJ:

4 (añadir 1 en modo decimal)

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

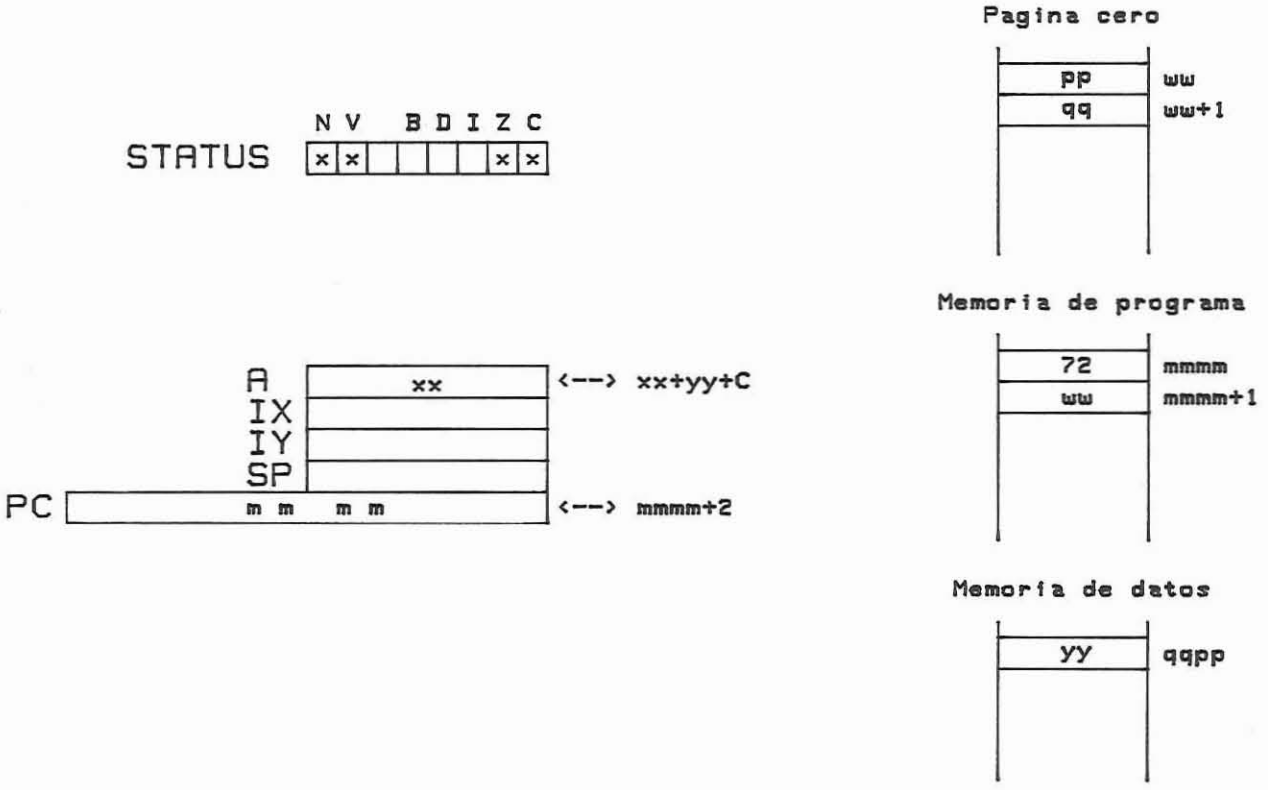
N: Se pone a 1 si el resultado es negativo.
V: Se pone a 1 si se produce "overflow".
Z: Se pone a 1 si el resultado es cero.
C: Se pone a 1 si hay acarreo.

DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la parte baja de la dirección de memoria que, sumada al contenido del registro X, da la dirección de memoria de página cero donde se encuentra el dato.Este dato se suma con el Acumulador y con el bit de Carry del Registro de Status. El resultado queda en el Acumulador.

ADC (dir) * SUMA CON ACARREO

(Direccionamiento indirecto mejorado)



NUMERO DE CICLOS DE RELOJ:

5 (añadir 1 en modo decimal)

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

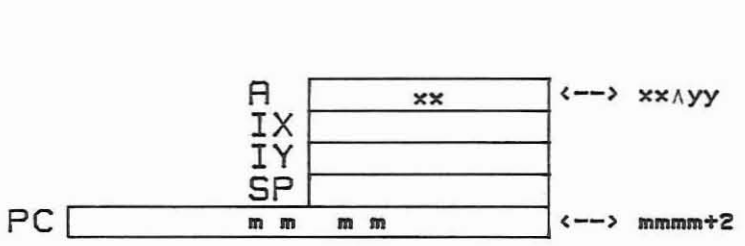
N: Se pone a 1 si el resultado es negativo.
V: Se pone a 1 si se produce "overflow".
Z: Se pone a 1 si el resultado es cero.
C: Se pone a 1 si hay acarreo.

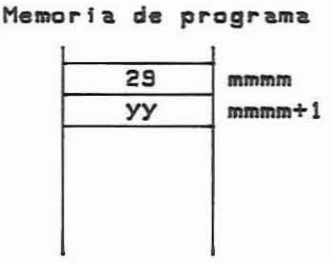
DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la parte baja de una dirección de memoria de página cero. En esta dirección y en la siguiente se encuentran, en orden inverso (primero byte de menor peso y despues byte de mayor peso), los dos bytes que forman la dirección de memoria donde se encuentra el dato que se suma con el Acumulador y con el bit de Carry del Registro de Status. El resultado queda en el Acumulador.

AND #dato — **FUNCION AND**
(Direccionamiento inmediato)







NUMERO DE CICLOS DE RELOJ:

2

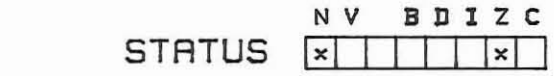
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

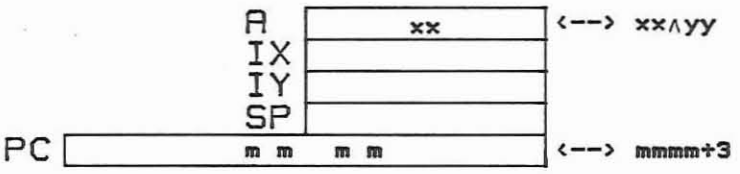
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

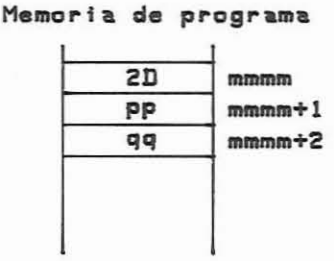
DESCRIPCION:

El contenido de la posición de memoria que sigue a la que contiene el código de operación efectúa bit a bit la operación AND con el Acumulador. El resultado queda en el Acumulador.

AND dir — **FUNCION AND**
(Direccionamiento absoluto)









NUMERO DE CICLOS DE RELOJ:

4

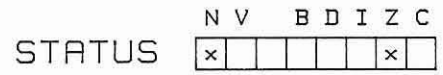
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

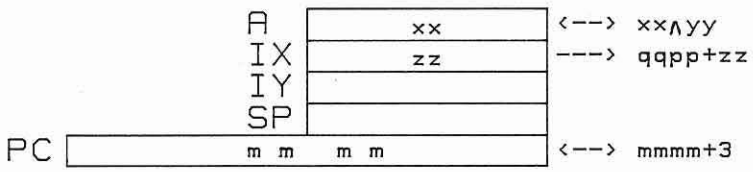
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

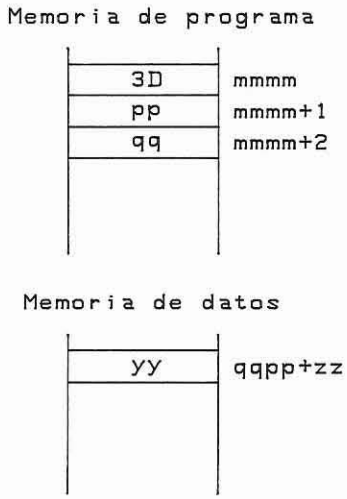
DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria donde se encuentra el dato que efectúa bit a bit la operación AND con el Acumulador. El resultado queda en el Acumulador.

AND dir,X **FUNCION AND**
(Direccionamiento absoluto indexado)







NUMERO DE CICLOS DE RELOJ:

4 (añadir 1 si salta de página)

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

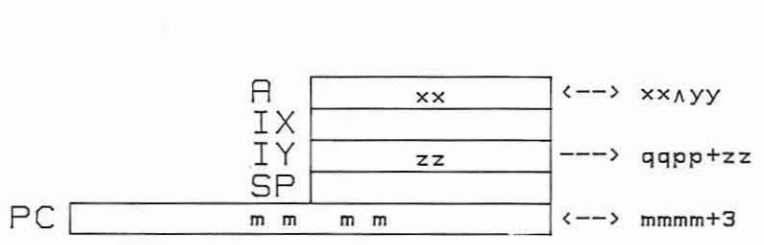
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

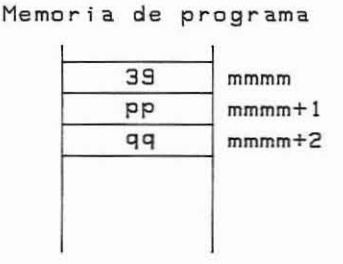
DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte de menor peso y luego el byte de mayor peso), la dirección de memoria que, sumada al contenido del registro X, indica la posición de memoria donde se encuentra el dato. Este dato efectúa bit a bit la operación AND con el Acumulador. El resultado queda en el Acumulador.

AND dir,Y **FUNCION AND**
(Direccionamiento absoluto indexado)









NUMERO DE CICLOS DE RELOJ:

4 (añadir 1 si salta de página)

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

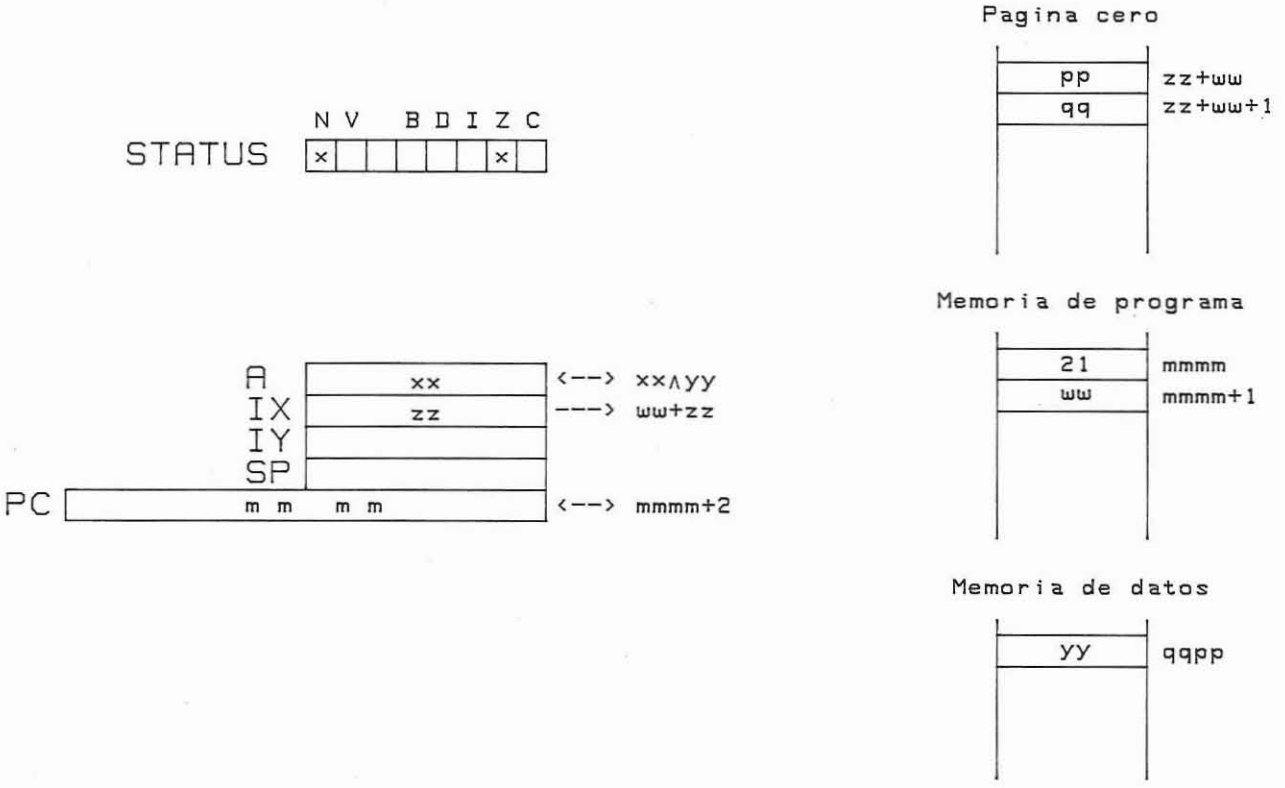
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria que, sumada al contenido del registro Y, indica la posición de memoria donde se encuentra el dato. Este dato efectúa bit a bit la operación AND con el Acumulador. El resultado queda en el Acumulador.

AND (dir,X) **FUNCION AND**

(Direccionamiento preindexado)



NUMERO DE CICLOS DE RELOJ:

6

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

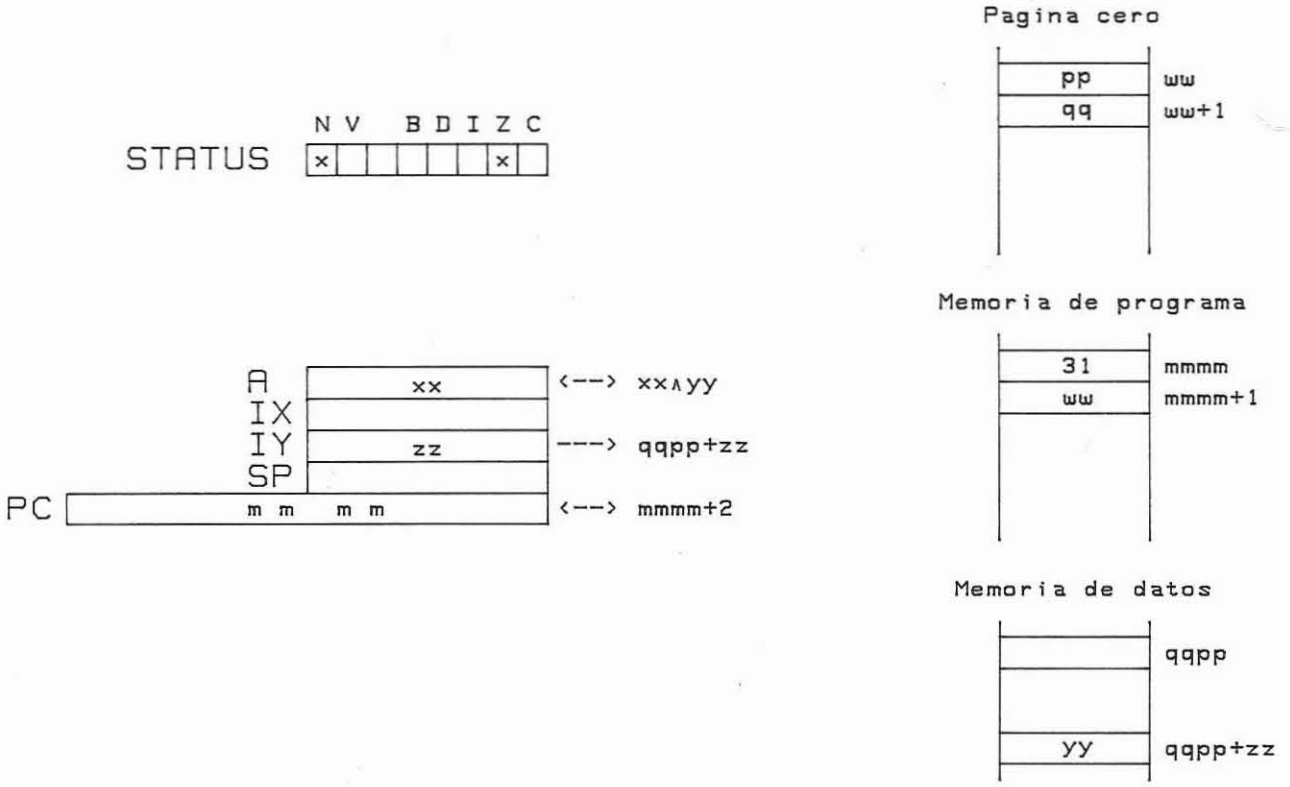
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

El contenido de la posición de memoria que sigue a la que contiene el código de operación se suma con el contenido del registro X. El resultado (despreciando el acarreo si se ha producido) corresponde a la dirección de memoria en página cero que contiene la parte baja de la dirección efectiva del dato. La parte alta de la dirección efectiva del dato está en la siguiente posición de memoria en pagina cero. De la dirección efectiva del dato se toma el byte que efectúa bit a bit la operación AND con el contenido del Acumulador. El resultado se carga de nuevo en el Acumulador.

AND (dir),Y **FUNCION AND**

(Direccionamiento postindexado)



NUMERO DE CICLOS DE RELOJ:

5 (añadir 1 si salta de página)

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

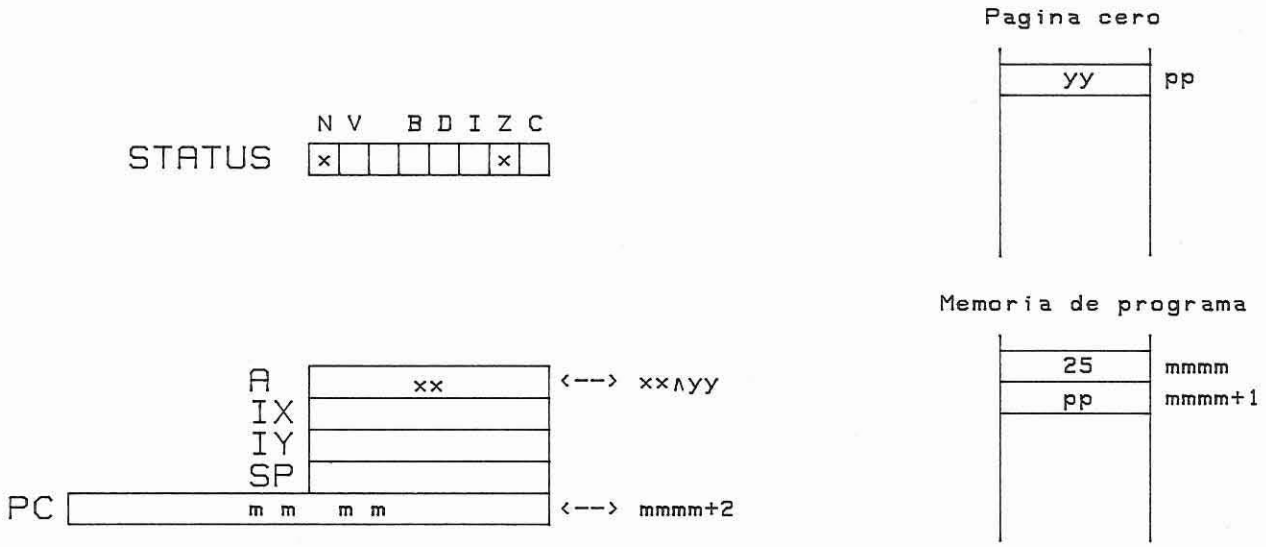
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

El contenido de la posición de memoria que sigue a la que contiene el código de operación contiene una dirección de memoria de página cero. En esta dirección y en la siguiente se encuentran, en orden inverso (primero el byte de menor peso y luego el byte mayor peso), los dos bytes que forman la dirección de memoria que sumada al contenido del registro Y, nos dará la dirección de memoria donde se encuentra el dato. Este dato efectúa la operación AND con el Acumulador. El resultado se guarda de nuevo en el Acumulador.

AND dir **FUNCION AND**

(Direccionamiento página cero)



NUMERO DE CICLOS DE RELOJ:

3

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

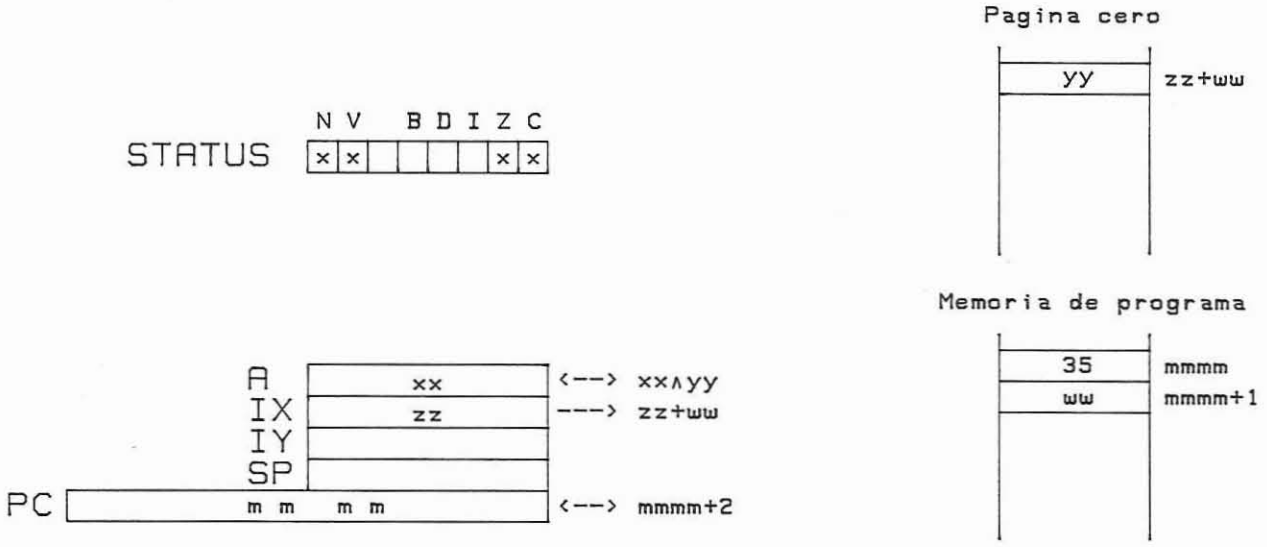
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la dirección de memoria de página cero donde se encuentra el dato que efectúa bit a bit la operación AND con el Acumulador. El resultado queda en el Acumulador.

AND dir,X **FUNCION AND**

(Direccionamiento página cero indexado)



NUMERO DE CICLOS DE RELOJ:

4

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

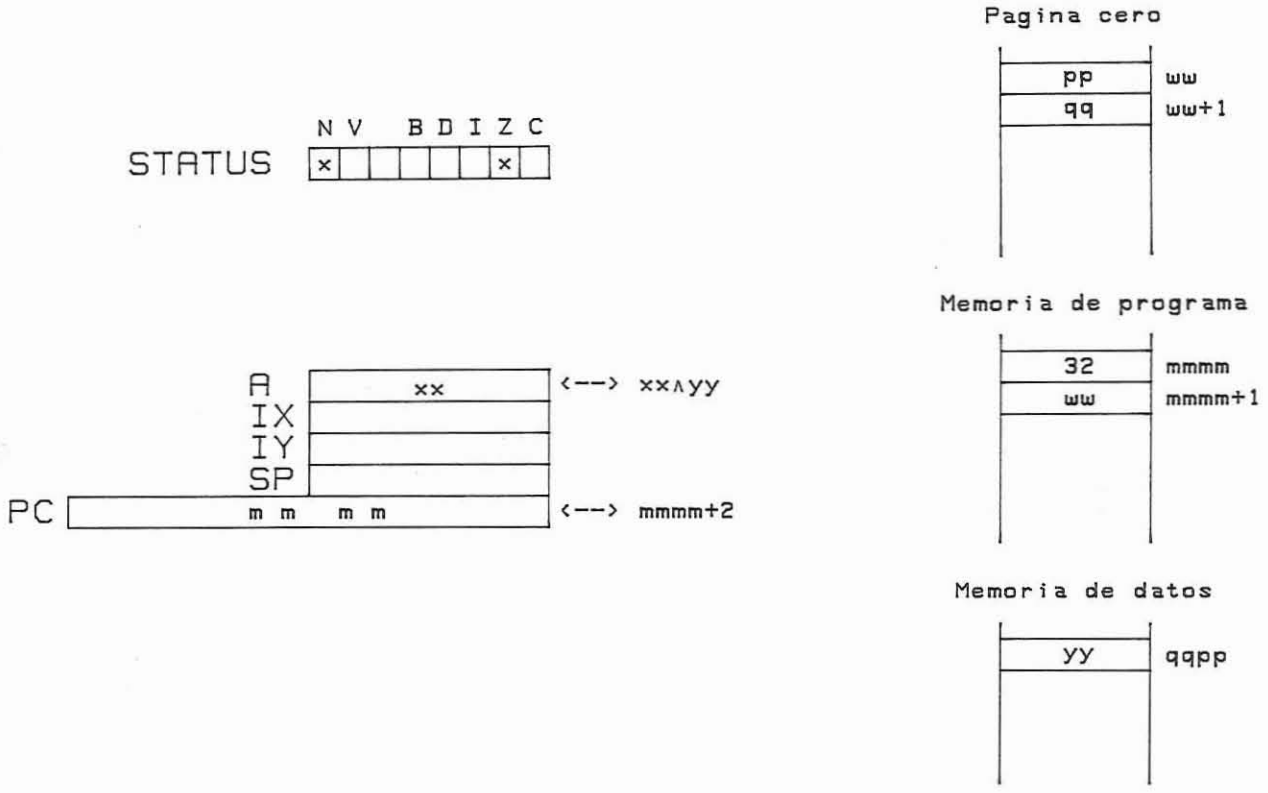
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la parte baja de la dirección de memoria que, sumada al contenido del registro X, da la dirección de memoria de página cero donde se encuentra el dato. Este dato efectúa bit a bit la operación AND con el Acumulador. El resultado queda en el Acumulador.

AND (dir) * **FUNCION AND**

(Direccionamiento indirecto mejorado)



NUMERO DE CICLOS DE RELOJ:

5

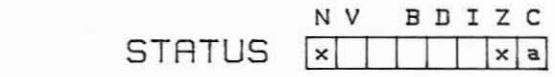
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

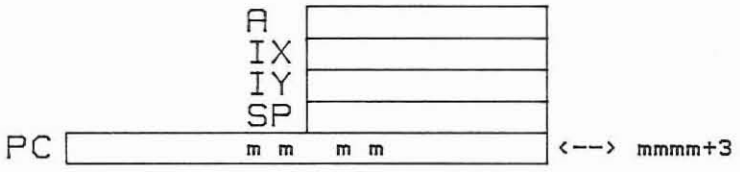
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

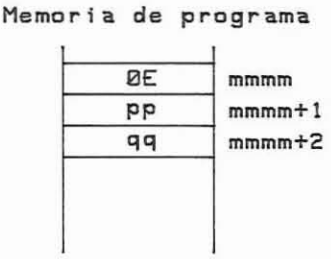
DESCRIPCION:

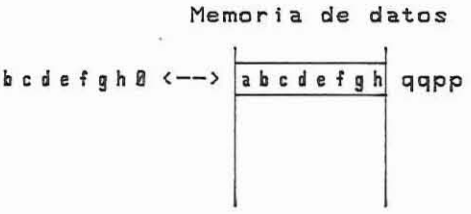
La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la parte baja de una dirección de memoria de página cero. En esta dirección y en la siguiente se encuentran, en orden inverso (primero byte de menor peso y despues byte de mayor peso), los dos bytes que forman la dirección de memoria donde se encuentra el dato que efectúa bit a bit la operación AND con el Acumulador. El resultado queda en el Acumulador.

ASL dir **DESPLAZAMIENTO DE UN BIT A LA IZQUIERDA**
(Direccionamiento absoluto)









NUMERO DE CICLOS DE RELOJ:

6

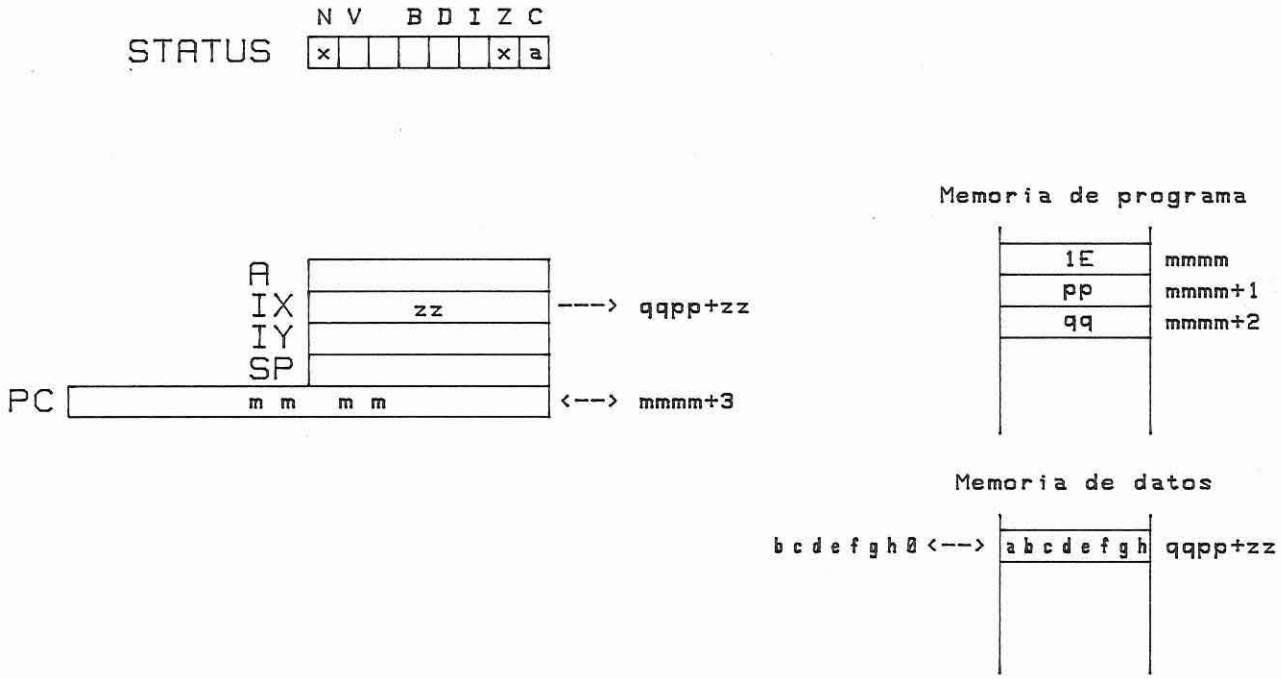
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.
C: Toma el valor del bit de mayor peso.

DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria donde se encuentra el dato cuyos bits son desplazados una posición hacia la izquierda. El bit de mayor peso pasa al Flag de Carry del Registro de Status y el bit de menor peso se pone a 0.

ASL dir,X **DESPLAZAMIENTO DE UN BIT A LA IZQUIERDA**
(Direccionamiento absoluto indexado)



NUMERO DE CICLOS DE RELOJ:

7

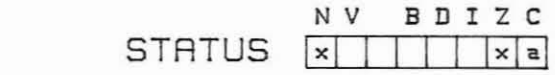
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

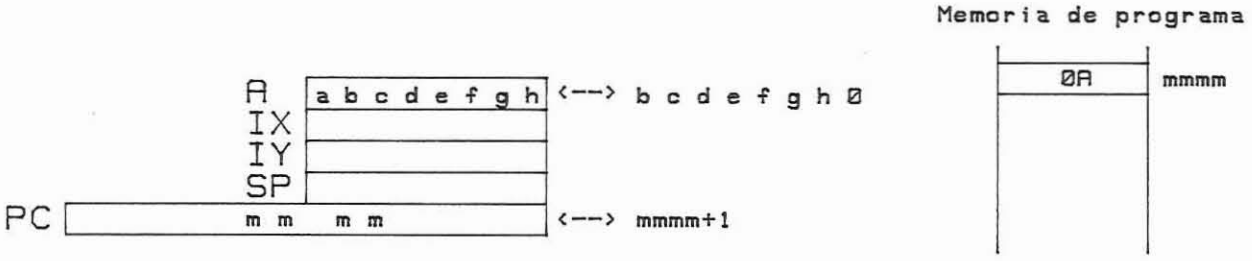
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.
C: Toma el valor del bit de mayor peso.

DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria que, sumada al contenido del registro X, indica la posición de memoria donde se encuentra el dato. Los bits de este byte se desplazan una posición hacia la izquierda. El bit de mayor peso pasa al Flag de Carry del Registro de Status y el bit de menor peso se pone a 0.

ASL A **DESPLAZAMIENTO DE UN BIT A LA IZQUIERDA**
(Direccionamiento de Acumulador)





NUMERO DE CICLOS DE RELOJ:

2

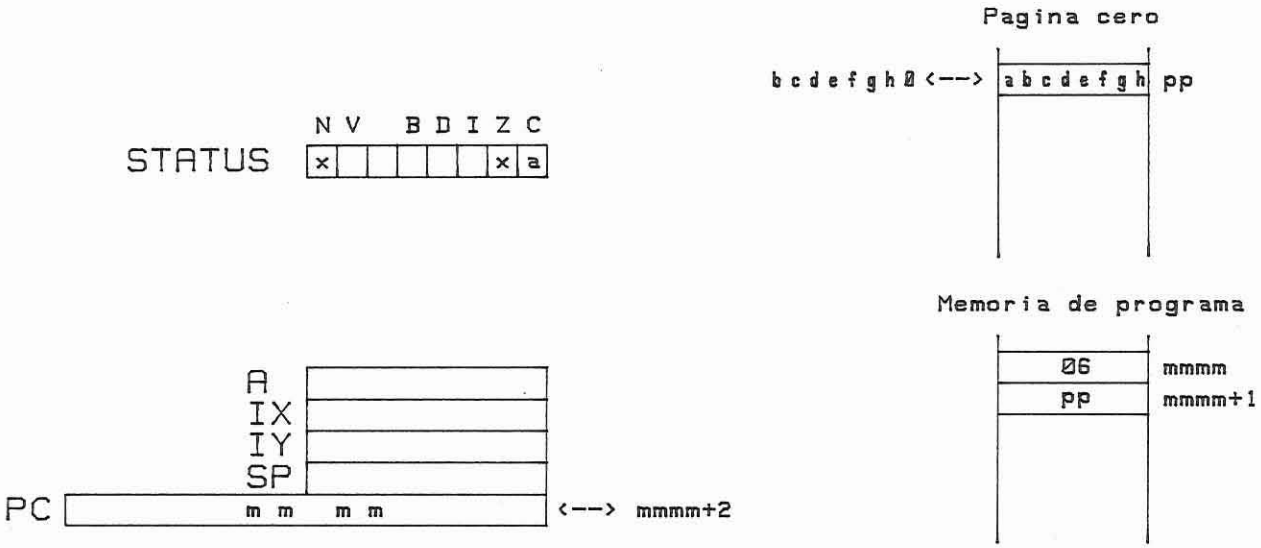
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.
C: Toma el valor del bit de mayor peso.

DESCRIPCION:

El contenido del Acumulador se desplaza un bit a la izquierda, de manera que el bit de más peso pasa al bit de Carry del Registro de Flags y el bit de menor peso se pone a cero.

ASL dir **DESPLAZAMIENTO DE UN BIT A LA IZQUIERDA**
(Direccionamiento página cero)



NUMERO DE CICLOS DE RELOJ:

5

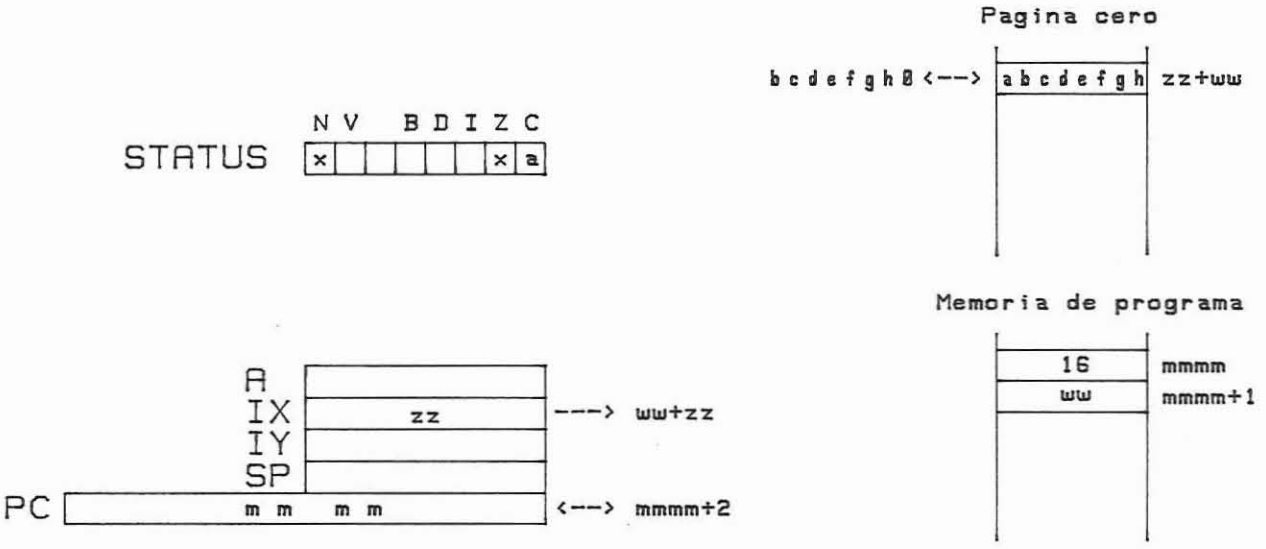
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.
C: Toma el valor del bit de mayor peso.

DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la dirección de memoria de página cero donde se encuentra el dato cuyos bits son desplazados una posición hacia la izquierda. El bit de mayor peso pasa al Flag de Carry del Registro de Status y el bit de menor peso se pone a 0.

ASL dir,X **DESPLAZAMIENTO DE UN BIT A LA IZQUIERDA**
(Direccionamiento página cero indexado)



NUMERO DE CICLOS DE RELOJ:

6

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

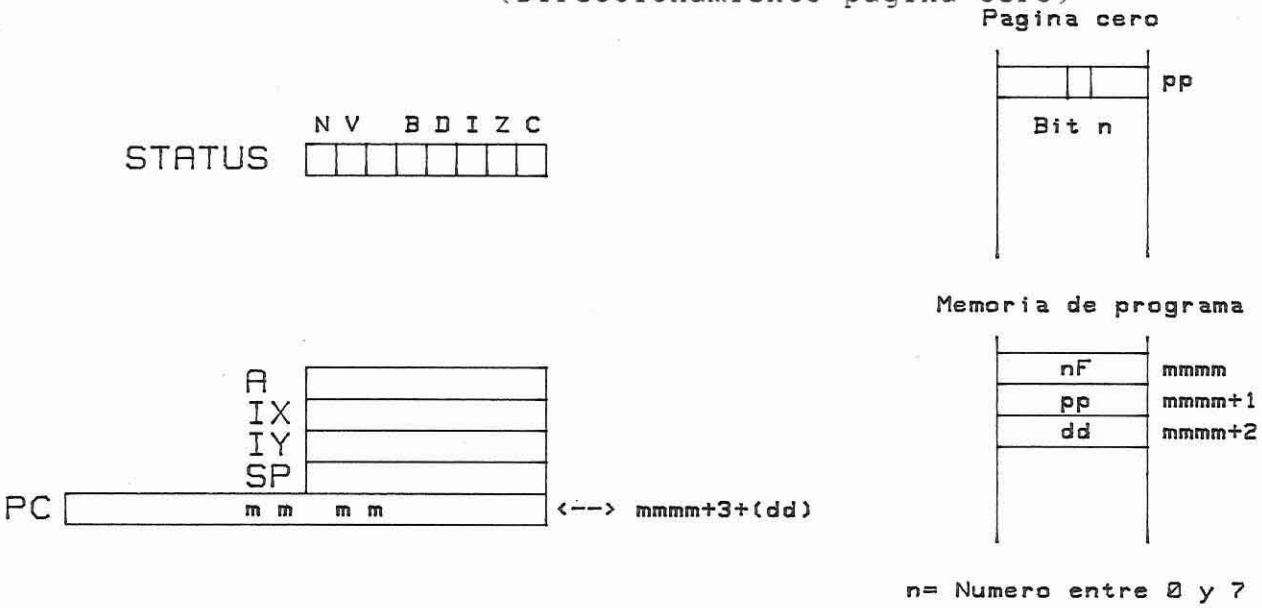
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.
C: Toma el valor del bit de mayor peso.

DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la parte baja de la dirección de memoria que, sumada al contenido del registro X, da la dirección de memoria de página cero donde se encuentra el dato. Los bits de este byte se desplazan una posición hacia la izquierda. El bit de mayor peso pasa al Flag de Carry del Registro de Status y el bit de menor peso se pone a 0.

BBR N dir, offset + **SALTO RELATIVO SI EL BIT N DE LA POSICION REFERENCIADA ES 0**

(Direccionamiento página cero)



NUMERO DE CICLOS DE RELOJ:

5 (añadir 1 si ocurre el salto detro de la misma página)
(añadir 2 si ocurre el salto a página diferente)

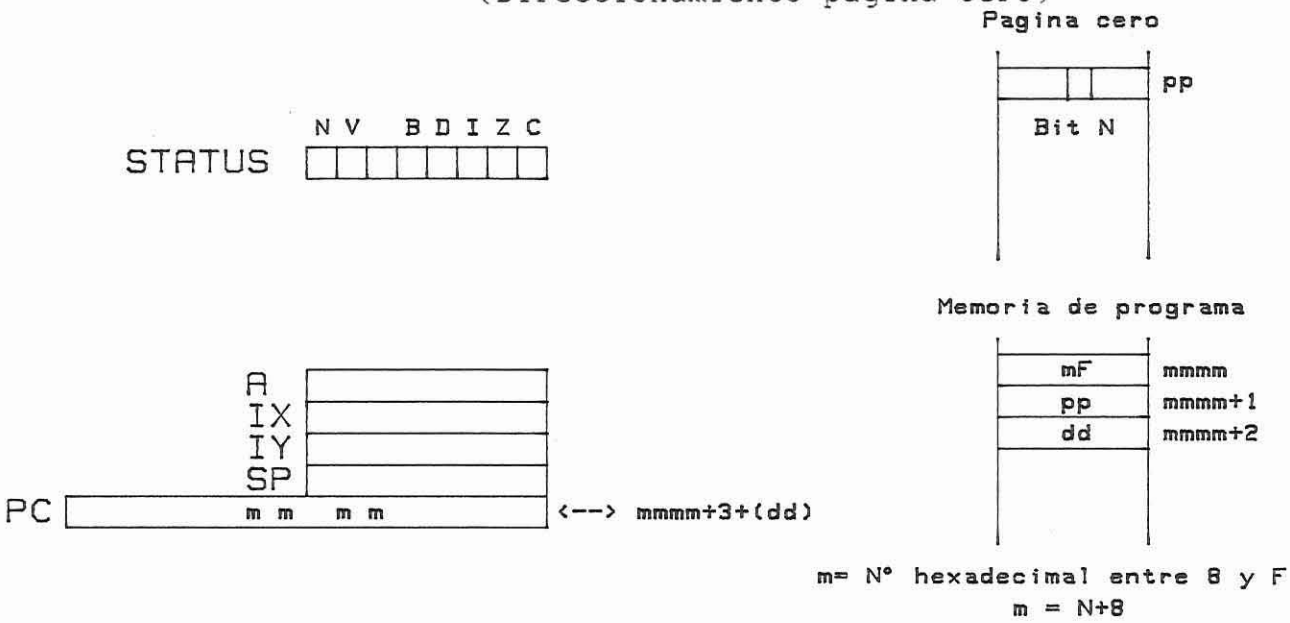
FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

DESCRIPCION:

Si el bit n de la dirección de memoria de página cero indicada por el segundo byte de la instrucción es 0, la posición de memoria que sigue a este segundo byte contiene el 'offset' en complemento a 2, que se suma al Contador de Programa (PC) una vez éste ya ha sido incrementado un número de veces igual al número de bytes de la instrucción. Este 'offset' estará entre +127 y -128. Si dicho bit es 1 la única operación realizada será el incremento de 3 del PC, con lo que se pasará a ejecutar la siguiente instrucción.

BBS N dir, offset + **SALTO RELATIVO SI EL BIT N DE LA POSICION REFERENCIADA ES 1**

(Direccionamiento página cero)



NUMERO DE CICLOS DE RELOJ:

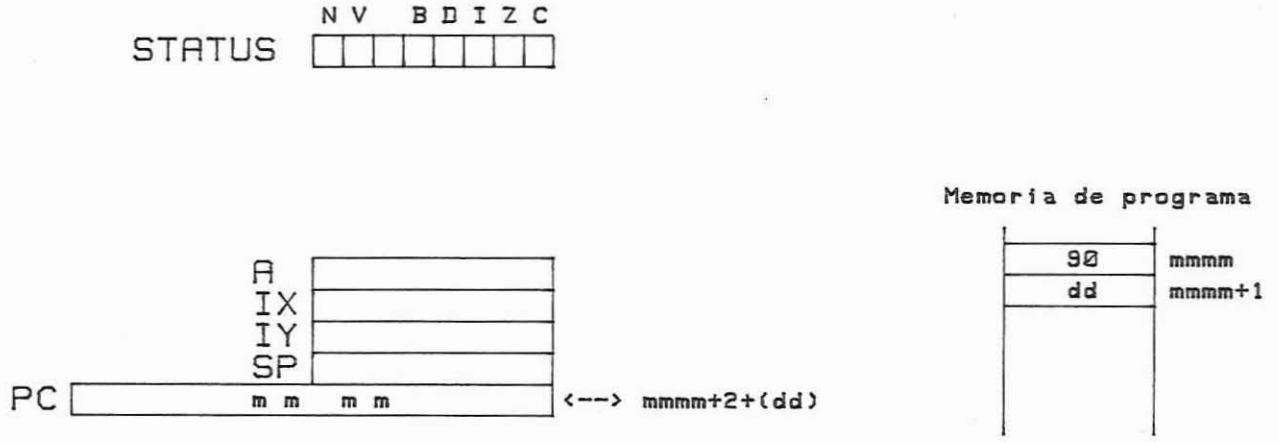
5 (añadir 1 si ocurre el salto detro de la misma página)
(añadir 2 si ocurre el salto a página diferente)

FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

DESCRIPCION:

Si el bit m de la dirección de memoria de página cero indicada por el segundo byte de la instrucción es 1, la posición de memoria que sigue a este segundo byte contiene el 'offset' en complemento a 2, que se suma al Contador de Programa (PC) una vez éste ya ha sido incrementado un número de veces igual al número de bytes de la instrucción. Este 'offset' estará entre +127 y -128. Si dicho bit es 0 la única operación realizada será el incremento de 3 del PC, con lo que se pasará a ejecutar la siguiente instrucción.

BCC offset — **SALTO SI EL BIT DE ACARREO ES 0**
(Direccionamiento relativo)



NUMERO DE CICLOS DE RELOJ:

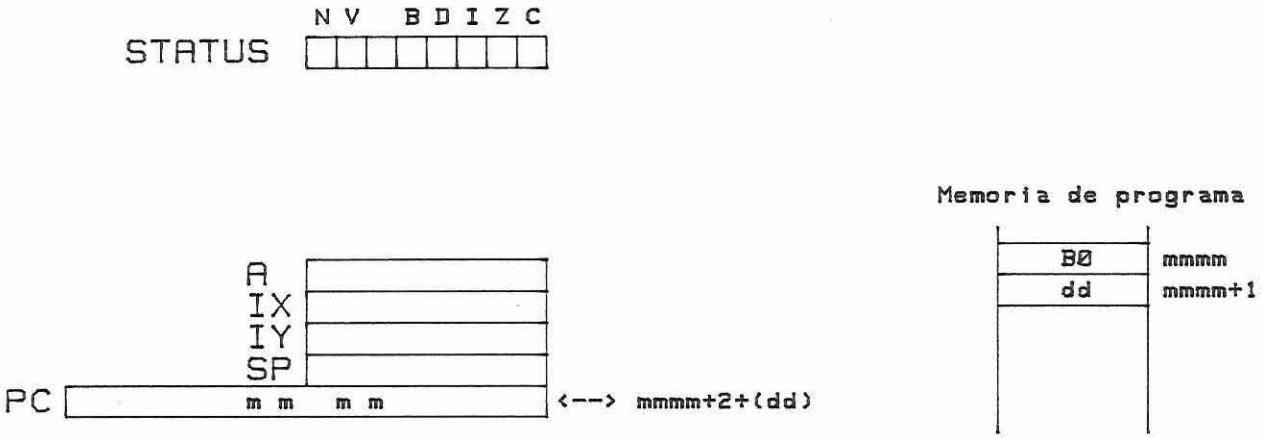
2 (añadir 1 si ocurre el salto detro de la misma página)
(añadir 2 si ocurre el salto a página diferente)

FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

DESCRIPCION:

Si el flag de Carry del Registro de Status es 0, la posición de memoria que sigue a la que contiene el código de operación, contiene el 'offset' en complemento a 2 que se suma al Contador de Programa (PC) una vez éste ya ha sido incrementado un número de veces igual al número de bytes de la instrucción. Este 'offset' estará entre +127 y -128. Si el bit de Carry es 1 la única operación realizada será el incremento de 2 del PC, con lo que se pasará a ejecutar la siguiente instrucción.

BCS offset — **SALTO SI EL BIT DE ACARREO ES 1**
(Direccionamiento relativo)



NUMERO DE CICLOS DE RELOJ:

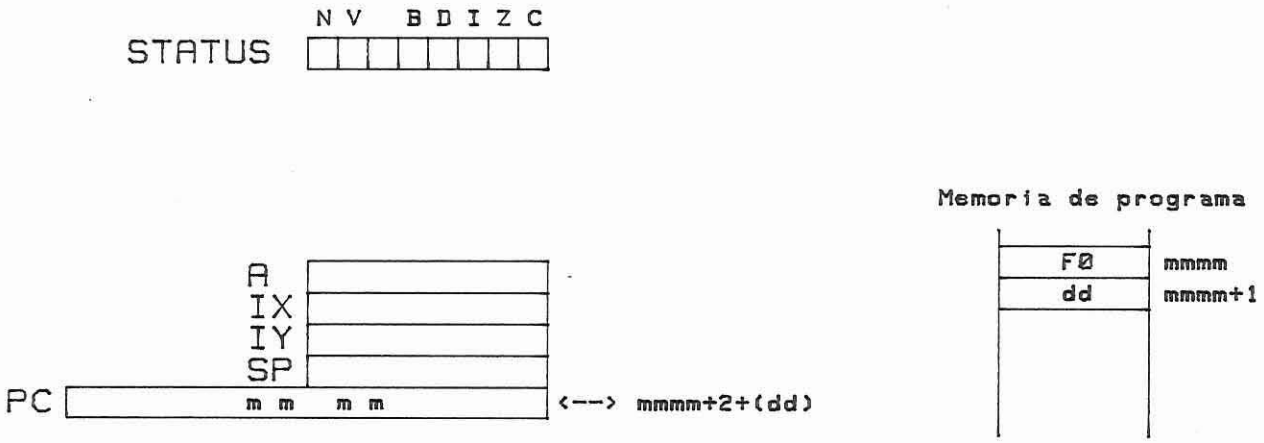
2 (añadir 1 si ocurre el salto detro de la misma página)
(añadir 2 si ocurre el salto a página diferente)

FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

DESCRIPCION:

Si el flag de Carry del Registro de Status es 1, la posición de memoria que sigue a la que contiene el código de operación, contiene el 'offset' en complemento a 2 que se suma al Contador de Programa (PC) una vez éste ya ha sido incrementado un número de veces igual al número de bytes de la instrucción. Este 'offset' estará entre +127 y -128. Si el bit de Carry es 0 la única operación realizada será el incremento de 2 del PC, con lo que se pasará a ejecutar la siguiente instrucción.

BEQ offset **SALTO SI EL RESULTADO FUE CERO (Z=1)**
(Direccionamiento relativo)



NUMERO DE CICLOS DE RELOJ:

2 (añadir 1 si ocurre el salto detro de la misma página)
(añadir 2 si ocurre el salto a página diferente)

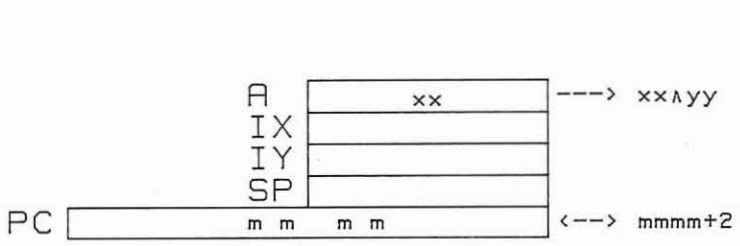
FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

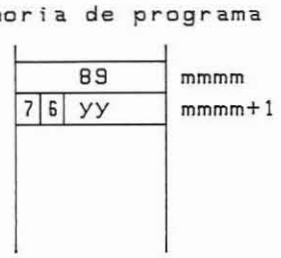
DESCRIPCION:

Si el flag de Cero (Z) del Registro de Status es 1, la posición de memoria que sigue a la que contiene el código de operación, contiene el 'offset' en complemento a 2 que se suma al Contador de Programa (PC) una vez éste ya ha sido incrementado un número de veces igual al número de bytes de la instrucción. Este 'offset' estará entre +127 y -128. Si el bit de Cero es 0 la única operación realizada será el incremento de 2 del PC, con lo que se pasará a ejecutar la siguiente instrucción.

BIT #dato * **COMPARACION AND Y COMPROBACION BITS 6 Y 7**
(Direccionamiento inmediato)







NUMERO DE CICLOS DE RELOJ:

2

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

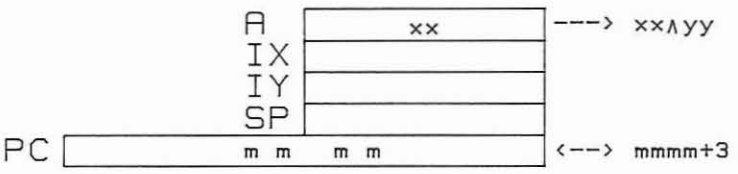
N: Toma el valor del bit 7 del dato analizado.
V: Toma el valor del bit 6 del dato analizado.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

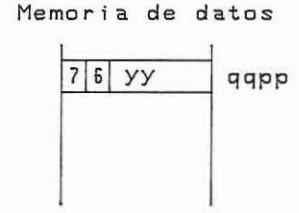
La posición de memoria que sigue a la que contiene el código de operación de la instrucción, contiene el dato que efectúa la operación AND lógica con el contenido del Acumulador. Si el resultado es cero el flag Z del Registro de Status se pone a 1 y si es uno el mismo flag se pone a 0. Esta instrucción permite analizar bit a bit el estado de cualquier posición de memoria, tan sólo poniendo previamente a 1 el bit del Acumulador correspondiente a la posición que queremos comprobar. Por otra parte, el flag N del Registro de Status toma el mismo valor que el bit 7 del dato analizado y el flag V toma el valor del bit 6.

BIT dir **COMPARACION AND Y COMPROBACION BITS 6 Y 7**
(Direccionamiento absoluto)









NUMERO DE CICLOS DE RELOJ:

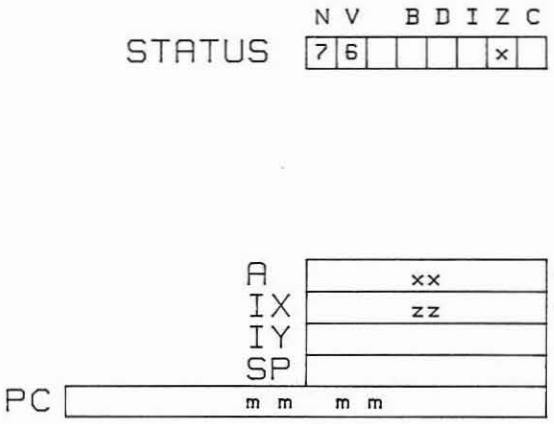
4

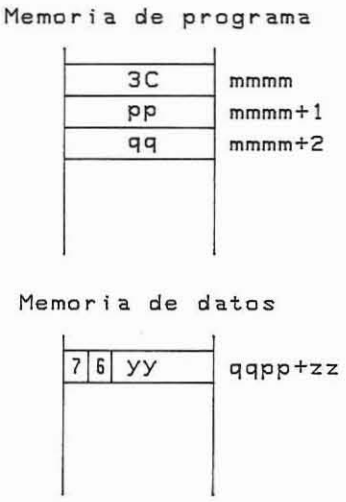
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Toma el valor del bit 7 del dato analizado.
V: Toma el valor del bit 6 del dato analizado.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION: Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria donde se encuentra el dato. Este byte efectúa la operación AND lógica con el contenido del Acumulador. Si el resultado es cero el flag Z del Registro de Status se pone a 1 y si es uno el mismo flag se pone a 0. Esta instrucción permite analizar bit a bit el estado de cualquier posición de memoria, tan sólo poniendo previamente a 1 el bit del Acumulador correspondiente a la posición que queremos comprobar. Por otra parte, el flag N del Registro de Status toma el mismo valor que el bit 7 del dato analizado y el flag V toma el valor del bit 6.

BIT dir,X * **COMPARACION AND Y COMPROBACION BITS 6 Y 7**
(Direccionamiento absoluto indexado)





NUMERO DE CICLOS DE RELOJ:

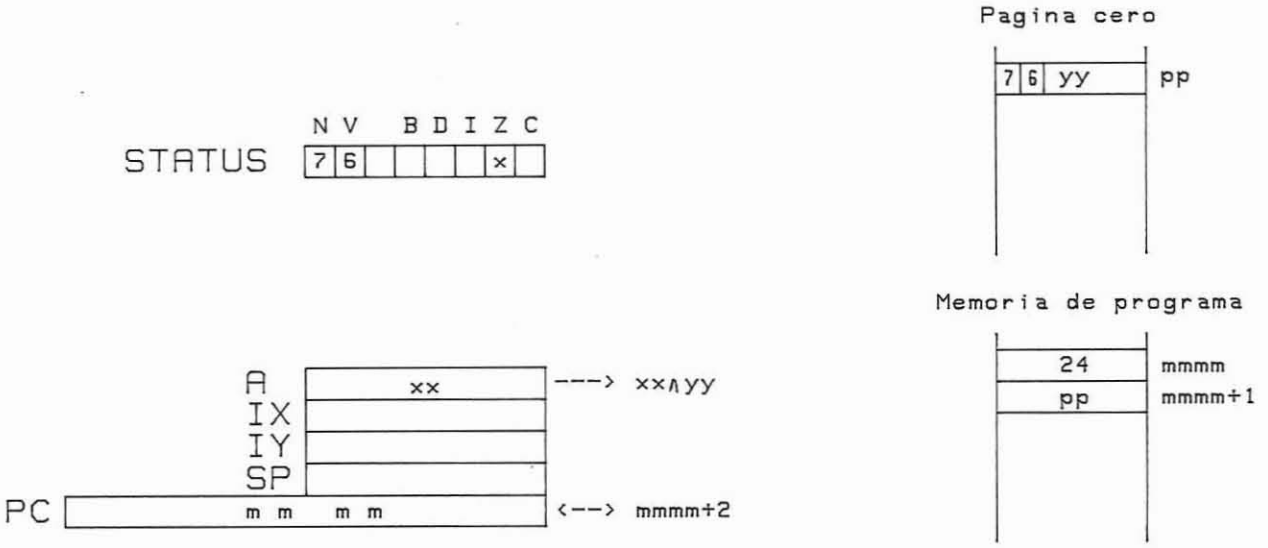
4 (añadir 1 si salta de página)

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Toma el valor del bit 7 del dato analizado.
V: Toma el valor del bit 6 del dato analizado.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION: Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria que, sumada al contenido del registro X, indica la posición de memoria donde se encuentra el dato. Este dato efectúa la operación AND lógica con el contenido del Acumulador. Si el resultado es cero el flag Z del Registro de Status se pone a 1 y si es uno el mismo flag se pone a 0. Esta instrucción permite analizar bit a bit el estado de cualquier posición de memoria, tan sólo poniendo previamente a 1 el bit del Acumulador correspondiente a la posición que queremos comprobar. Por otra parte, el flag N del Registro de Status toma el mismo valor que el bit 7 del dato analizado y el flag V toma el valor del bit 6.

BIT dir **COMPARACION AND Y COMPROBACION BITS 6 Y 7**
(Direccionamiento página cero)



NUMERO DE CICLOS DE RELOJ:

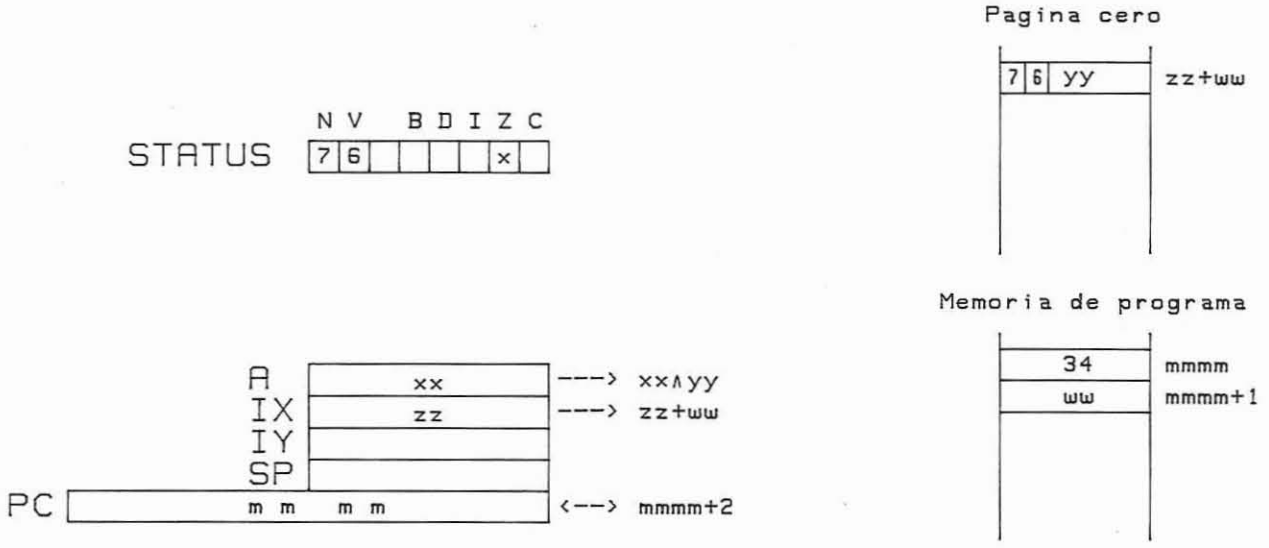
3

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Toma el valor del bit 7 del dato analizado.
V: Toma el valor del bit 6 del dato analizado.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION: La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la dirección de memoria de página cero donde se encuentra el dato. Este byte efectúa la operación AND lógica con el contenido del Acumulador. Si el resultado es cero el flag Z del Registro de Status se pone a 1 y si es uno el mismo flag se pone a 0. Esta instrucción permite analizar bit a bit el estado de cualquier posición de memoria de página cero tan sólo poniendo previamente a 1 el bit del Acumulador correspondiente a la posición que queremos comprobar. Por otra parte, el flag N del Registro de Status toma el mismo valor que el bit 7 del dato analizado y el flag V toma el valor del bit 6.

BIT dir,X * **COMPARACION AND Y COMPROBACION BITS 6 Y 7**
(Direccionamiento página cero indexado)



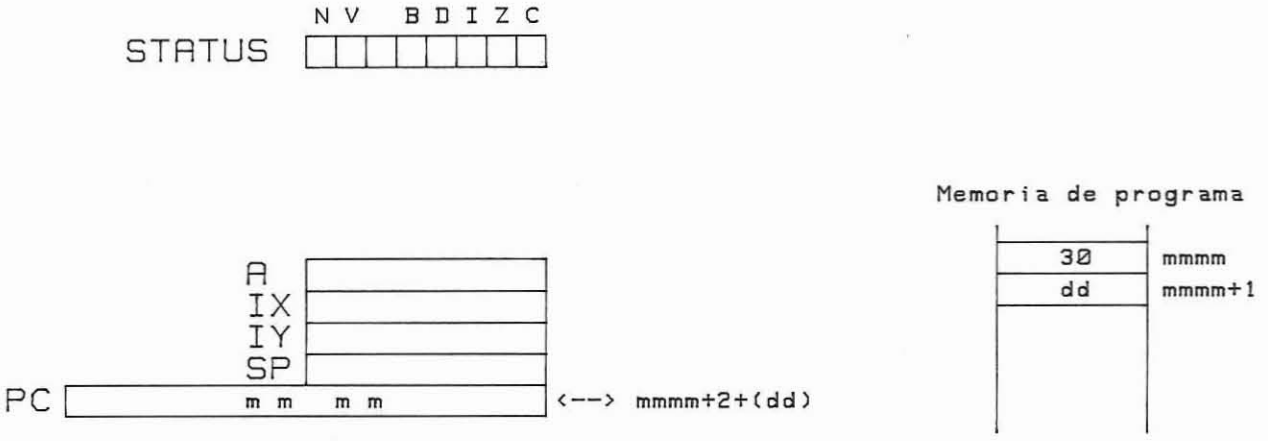
NUMERO DE CICLOS DE RELOJ:
4

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Toma el valor del bit 7 del dato analizado.
V: Toma el valor del bit 6 del dato analizado.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION: La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la parte baja de la dirección de memoria que, sumada al contenido del registro X, da la dirección de memoria de página cero donde se encuentra el dato. Este dato efectúa la operación AND lógica con el contenido del Acumulador. Si el resultado es cero el flag Z del Registro de Status se pone a 1 y si es uno el mismo flag se pone a 0. Esta instrucción permite analizar bit a bit el estado de cualquier posición de memoria de página cero, tan sólo poniendo previamente a 1 el bit del Acumulador correspondiente a la posición que queremos comprobar. Por otra parte, el flag N del Registro de Status toma el mismo valor que el bit 7 del dato analizado y el flag V toma el valor del bit 6.

BMI offset **SALTO SI EL RESULTADO FUE NEGATIVO (N=1)**
(Direccionamiento relativo)



NUMERO DE CICLOS DE RELOJ:

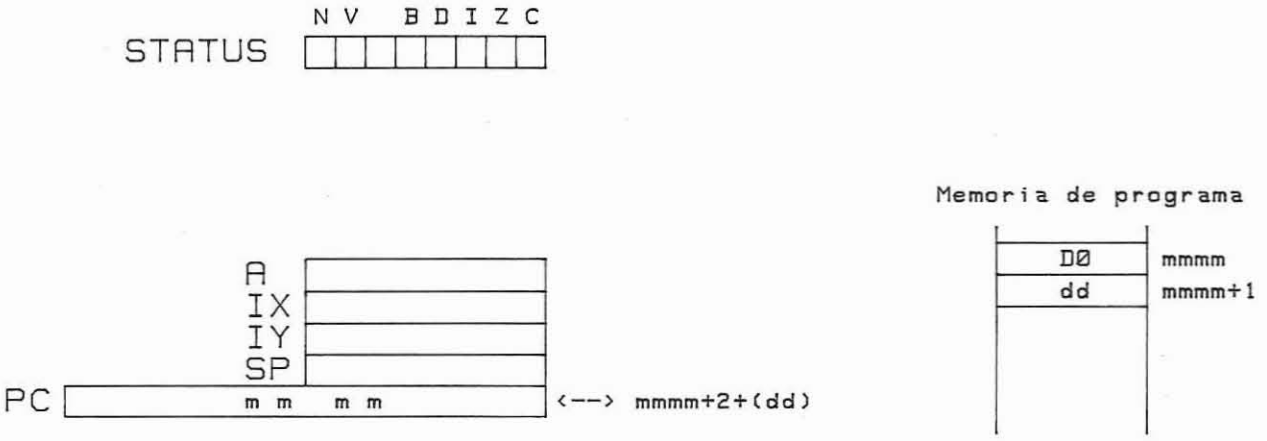
2 (añadir 1 si ocurre el salto detro de la misma página)
(añadir 2 si ocurre el salto a página diferente)

FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

DESCRIPCION:

Si el flag de Negativo del Registro de Status es 1, la posición de memoria que sigue a la que contiene el código de operación, contiene el 'offset' en complemento a 2 que se suma al Contador de Programa (PC) una vez éste ya ha sido incrementado un número de veces igual al número de bytes de la instrucción. Este 'offset' estará entre +127 y -128. Si el bit de Negativo es 0 la única operación realizada será el incremento de 2 del PC, con lo que se pasará a ejecutar la siguiente instrucción.

BNE offset **SALTO SI EL RESULTADO NO FUE CERO (Z=0)**
(Direccionamiento relativo)



NUMERO DE CICLOS DE RELOJ:

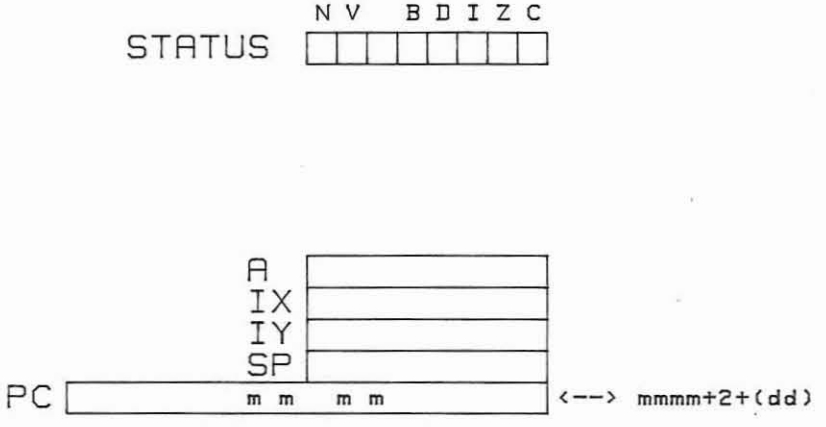
2 (añadir 1 si ocurre el salto detro de la misma página)
(añadir 2 si ocurre el salto a página diferente)

FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

DESCRIPCION:

Si el flag de Cero del Registro de Status es 0, la posición de memoria que sigue a la que contiene el código de operación, contiene el 'offset' en complemento a 2 que se suma al Contador de Programa (PC) una vez éste ya ha sido incrementado un número de veces igual al número de bytes de la instrucción. Este 'offset' estará entre +127 y -128. Si el bit de Cero es 1 la única operación realizada será el incremento de 2 del PC, con lo que se pasará a ejecutar la siguiente instrucción.

BPL offset **SALTO SI EL RESULTADO FUE POSITIVO (N=0)**
(Direccionamiento relativo)



NUMERO DE CICLOS DE RELOJ:

2 (añadir 1 si ocurre el salto detro de la misma página)
(añadir 2 si ocurre el salto a página diferente)

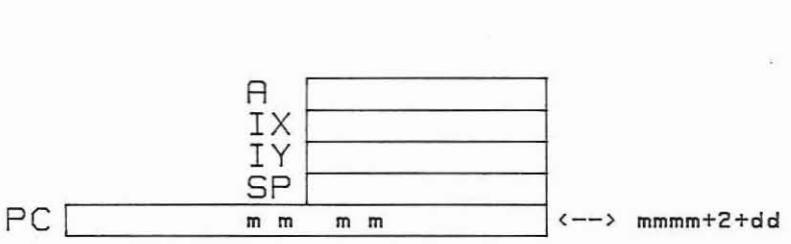
FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

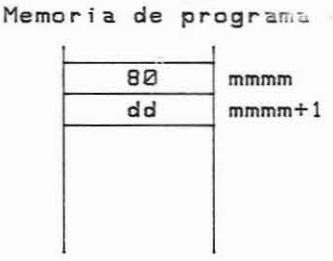
DESCRIPCION:

Si el flag de Negativo del Registro de Status es 0 (N=0), la posición de memoria que sigue a la que contiene el código de operación, contiene el 'offset' en complemento a 2 que se suma al Contador de Programa (PC) una vez éste ya ha sido incrementado un número de veces igual al número de bytes de la instrucción. Este 'offset' estará entre +127 y -128. Si el bit de Negativo es 1 la única operación realizada será el incremento de 2 del PC, con lo que se pasará a ejecutar la siguiente instrucción.

BRA * **SALTO INCONDICIONAL**
(Direccionamiento relativo)







NUMERO DE CICLOS DE RELOJ:

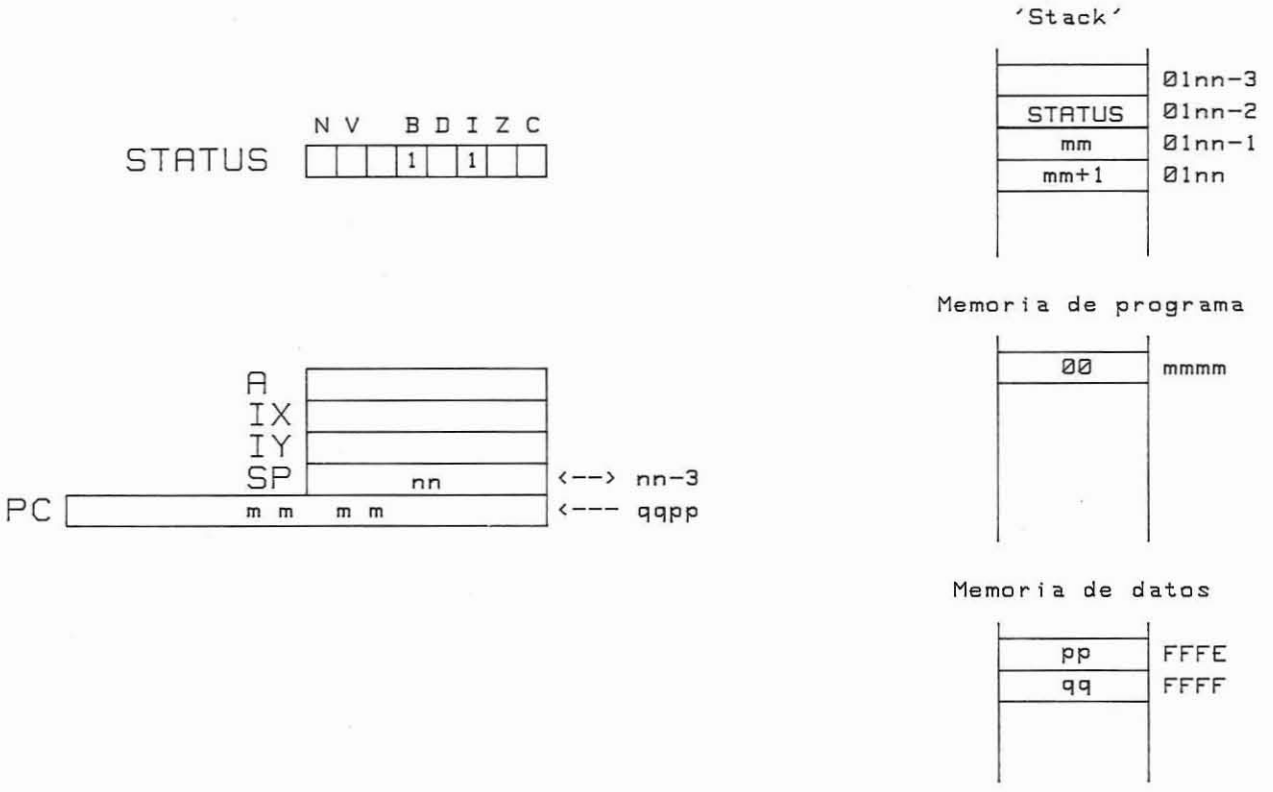
3 (añadir 1 si salta de página)

FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación, contiene el 'offset' en complemento a 2 que se suma al Contador de Programa (PC) una vez éste ya ha sido incrementado un número de veces igual al número de bytes de la instrucción. Este 'offset' estará entre +127 y -128. La siguiente instrucción que se ejecutará será la que se encuentra en esta posición de memoria.

BRK **INTERRUPCION POR PROGRAMA**
(Direccionamiento implicito)



NUMERO DE CICLOS DE RELOJ:

7

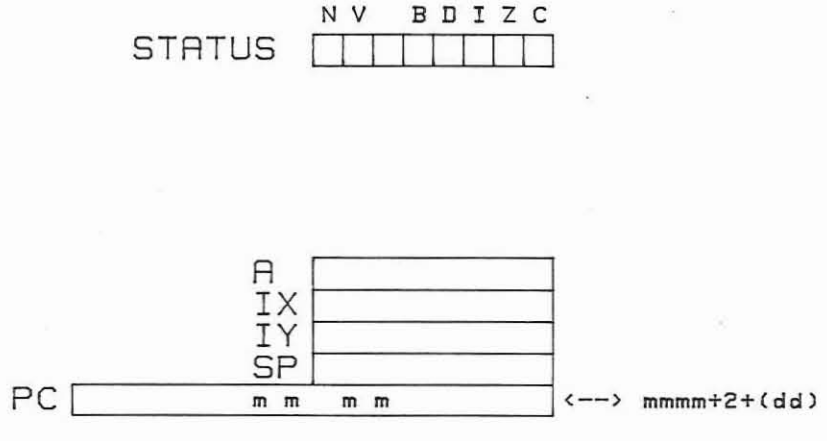
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

I: Se pone a 1.
B: Se pone a 1.

DESCRIPCION:

Al ejecutar esta instrucción se produce el mismo efecto que si se tratase de una interrupción del tipo íIRQ poniendose a 1 el flag I del Registro de Status. Por tanto se saltará a la posición indicada por el vector que se encuentra en las posiciones 0FFFEH (byte bajo) y 0FFFFH (byte alto) y se ejecutará la subrutina de interrupción. Dentro de esta subrutina se puede discernir si la fuente de la interrupción ha sido hardware o una instrucción de break analizando el estado del flag B del Registro de Status.

BVC offset **SALTO SI NO HUBO 'OVERFLOW' (V=0)**
(Direccionamiento relativo)



NUMERO DE CICLOS DE RELOJ:

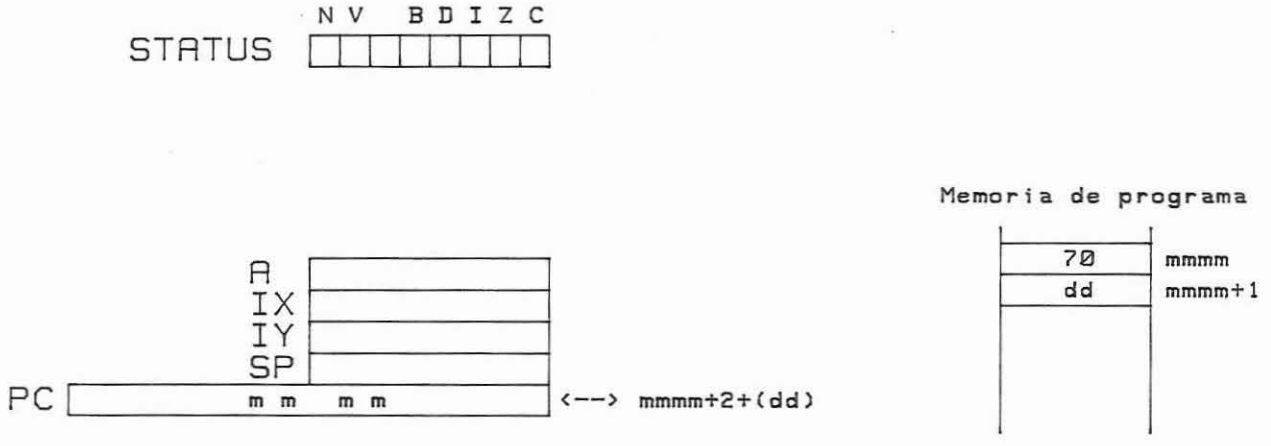
2 (añadir 1 si ocurre el salto detro de la misma página)
(añadir 2 si ocurre el salto a página diferente)

FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

DESCRIPCION:

Si el flag de Overflow (desbordamiento) del Registro de Status es 0, la posición de memoria que sigue a la que contiene el código de operación, contiene el 'offset' en complemento a 2 que se suma al Contador de Programa (PC) una vez éste ya ha sido incrementado un número de veces igual al número de bytes de la instrucción. Este 'offset' estará entre +127 y -128. Si el bit de Overflow es 1 la única operación realizada será el incremento de 2 del PC, con lo que se pasará a ejecutar la siguiente instrucción.

BVS offset — **SALTO SI 'OVERFLOW' (V=1)**
(Direccionamiento relativo)



NUMERO DE CICLOS DE RELOJ:

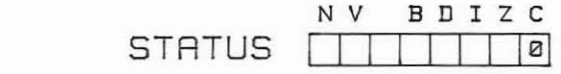
2 (añadir 1 si ocurre el salto detro de la misma página)
(añadir 2 si ocurre el salto a página diferente)

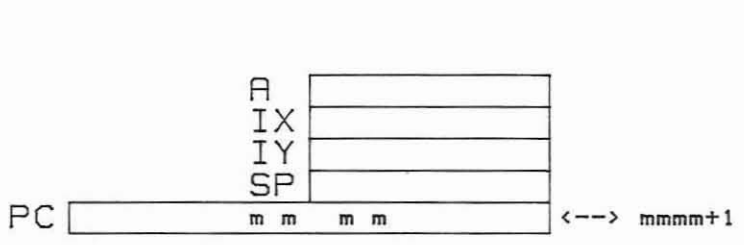
FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

DESCRIPCION:

Si el flag de Overflow (desbordamiento) del Registro de Status es 1, la posición de memoria que sigue a la que contiene el código de operación, contiene el 'offset' en complemento a 2 que se suma al Contador de Programa (PC) una vez éste ya ha sido incrementado un número de veces igual al número de bytes de la instrucción. Este 'offset' estará entre +127 y -128. Si el bit de Overflow es 0 la única operación realizada será el incremento de 2 del PC, con lo que se pasará a ejecutar la siguiente instrucción.

CLC **PONER A 0 EL BIT DE ACARREO**
(Direccionamiento implícito)







NUMERO DE CICLOS DE RELOJ:

2

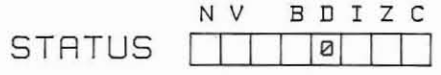
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

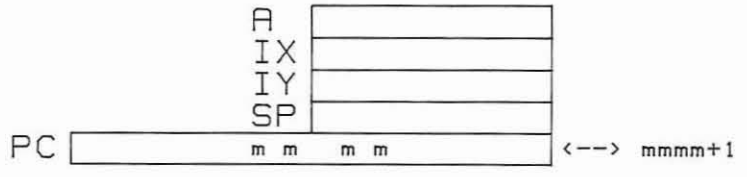
C: Se pone a 0.

DESCRIPCION:

Cuando se ejecuta esta instrucción, el flag de Carry del Registro de Status se pone a 0 fuera cual fuera el valor que tenía anteriormente.

CLD **PONER A 0 EL BIT DE MODO DECIMAL**
(Direccionamiento implícito)







NUMERO DE CICLOS DE RELOJ:

2

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

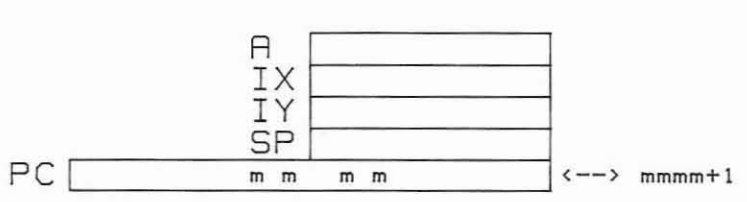
D: Se pone a 0.

DESCRIPCION:

Cuando se ejecuta esta instrucción, el flag D del Registro de Status se pone a 0 fuera cual fuera el valor que tenía anteriormente. Este flag indica, en las operaciones aritméticas que se realizan, que los operandos son números binarios de 8 bits.

CLI **PONER A 0 EL BIT DE PERMISION DE INTERRUPCIONES**
(Direccionamiento implícito)







NUMERO DE CICLOS DE RELOJ:

2

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

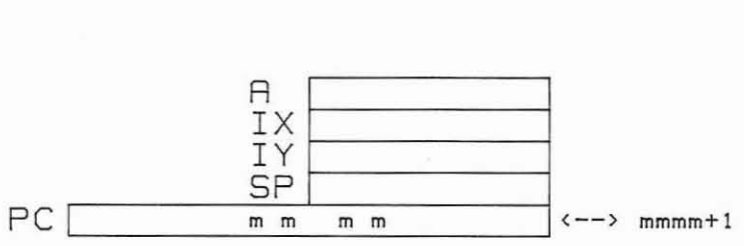
I: Se pone a 0.

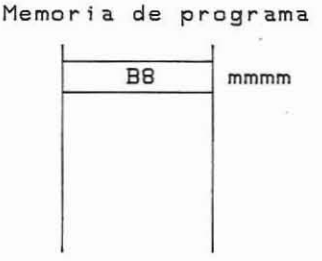
DESCRIPCION:

Cuando se ejecuta esta instrucción, el flag I del Registro de Status se pone a 0 fuera cual fuera el valor que tenía anteriormente. Este flag puesto a 0 indica que en el momento que se produzca una interrupción del tipo íIRQ se detendrá momentaneamente la tarea que se esté realizando y pasará a ejecutarse la subrutina de interrupción.

CLV **PONER A 0 EL BIT DE 'OVERFLOW'**
(Direccionamiento implícito)







NUMERO DE CICLOS DE RELOJ:

2

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

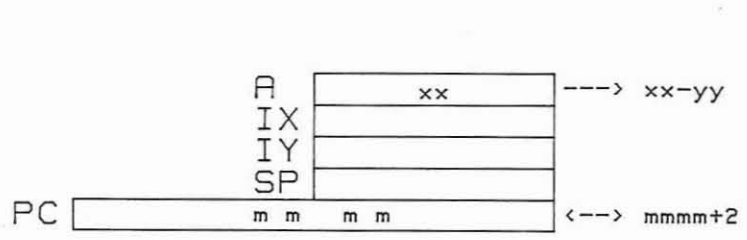
V: Se pone a 0.

DESCRIPCION:

Cuando se ejecuta esta instrucción, el flag V del Registro de Status se pone a 0 fuera cual fuera el valor que tenía anteriormente. Este flag indica, en las operaciones aritméticas ADC y SBC que se realizan, que ha habido 'overflow' o desbordamiento.

CMP #dato **COMPARAR CON ACUMULADOR**
(Direccionamiento inmediato)







NUMERO DE CICLOS DE RELOJ:

2

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

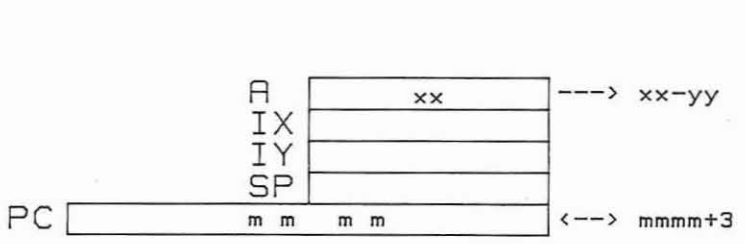
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.
C: Se pone a 1 si hay acarreo.

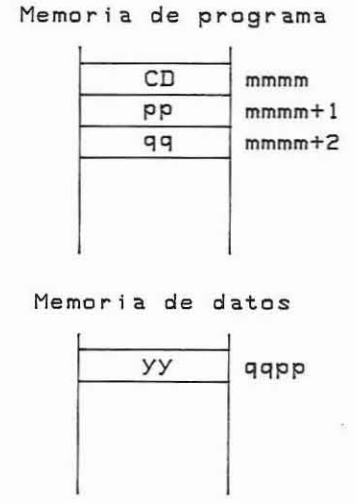
DESCRIPCION:

El contenido de la posición de memoria que sigue a la que contiene el código de operación se resta al Acumulador. El único registro afectado por esta instrucción es el registro de Status.

CMP dir — **COMPARAR CON ACUMULADOR**
(Direccionamiento absoluto)







NUMERO DE CICLOS DE RELOJ:

4

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

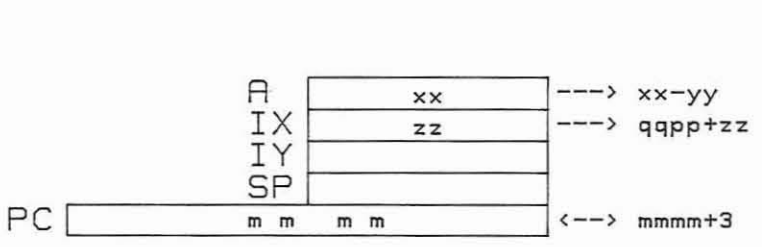
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.
C: Se pone a 1 si hay acarreo.

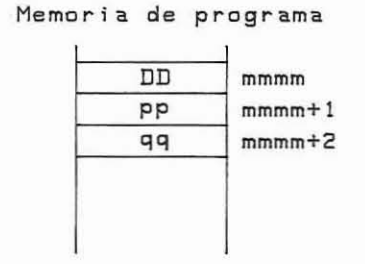
DESCRIPCION:

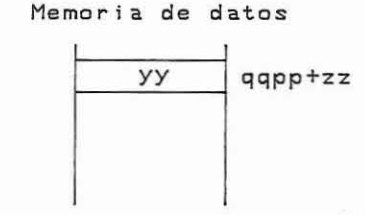
Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria donde se encuentra el dato que se resta al Acumulador. El único registro afectado por esta instrucción es el Registro de Status.

CMP dir,X — **COMPARAR CON ACUMULADOR**
(Direccionamiento absoluto indexado)









NUMERO DE CICLOS DE RELOJ:

4 (añadir 1 si salta de página)

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

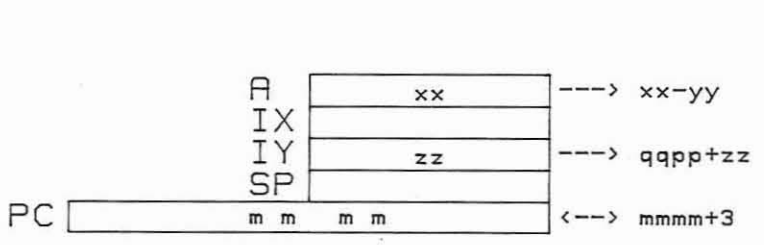
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.
C: Se pone a 1 si hay acarreo.

DESCRIPCION:

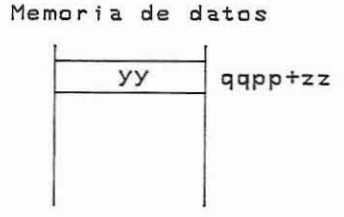
Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria que, sumada al contenido del registro X, indica la posición de memoria donde se encuentra el dato. Este dato se resta del Acumulador. El único registro afectado por esta instrucción es el registro de Status.

CMP dir,Y — **COMPARAR CON ACUMULADOR**
(Direccionamiento absoluto indexado)









NUMERO DE CICLOS DE RELOJ:

4 (añadir 1 si salta de página)

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

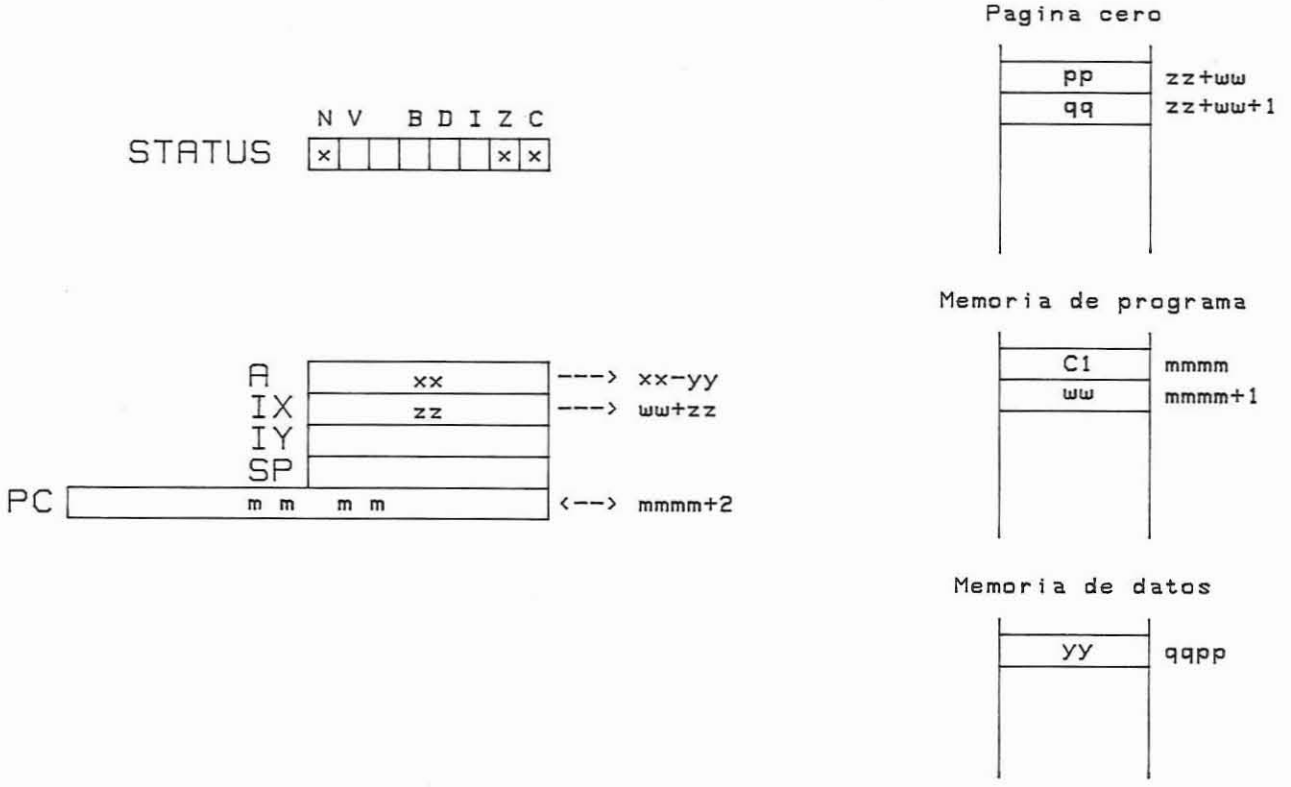
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.
C: Se pone a 1 si hay acarreo.

DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria que, sumada al contenido del registro Y, indica la posición de memoria donde se encuentra el dato. Este dato se resta del Acumulador. El único registro afectado por esta instrucción es el registro de Status.

CMP (dir,X)

COMPARAR CON ACUMULADOR
(Direccionamiento preindexado)



NUMERO DE CICLOS DE RELOJ:

6

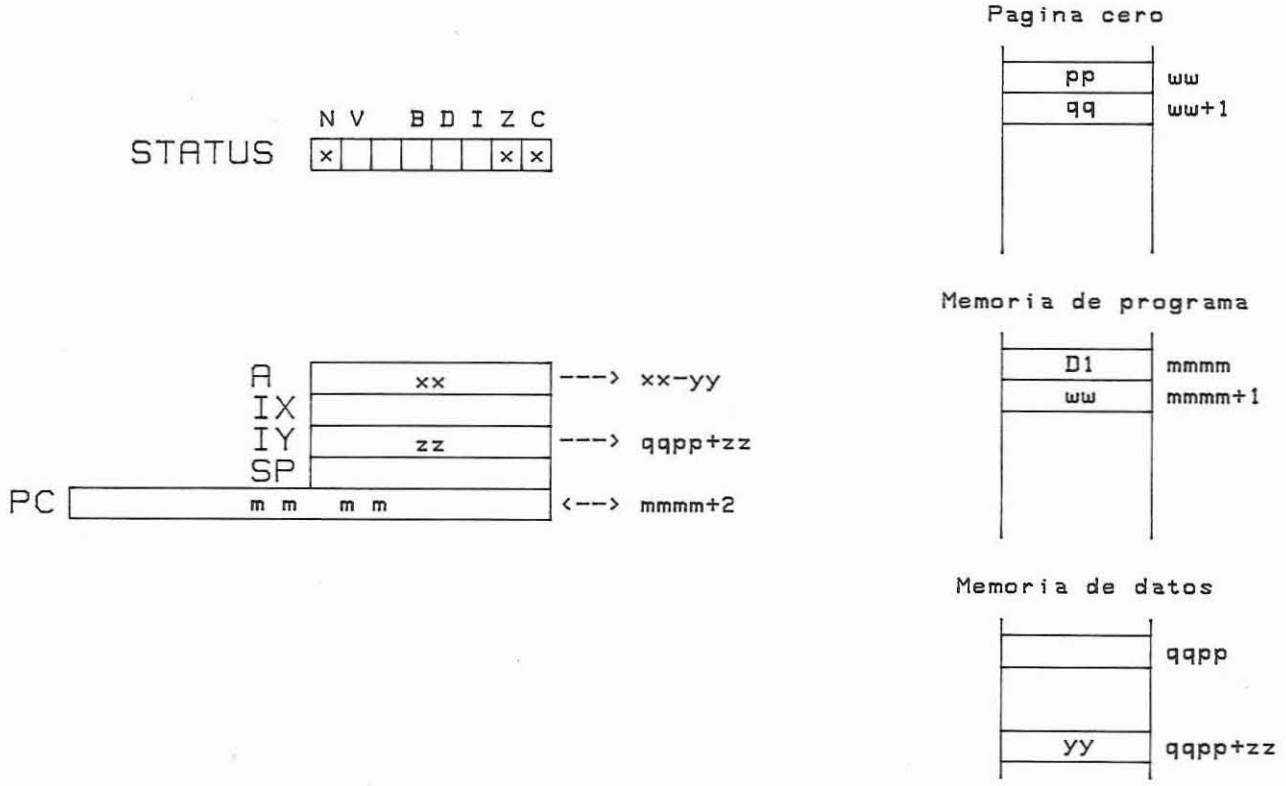
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.
C: Se pone a 1 si hay acarreo.

DESCRIPCION:

El contenido de la posición de memoria que sigue a la que contiene el código de operación se suma con el contenido del registro X. El resultado (despreciando el acarreo si se ha producido) corresponde a la dirección de memoria en página cero que contiene la parte baja de la dirección efectiva del dato. La parte alta de la dirección efectiva del dato está en la siguiente posición de memoria en pagina cero. De la dirección efectiva del dato se toma el byte que se resta del Acumulador. El único registro afectado por esta instrucción es el registro de Status.

CMP (dir),Y — **COMPARAR CON ACUMULADOR**
(Direccionamiento postindexado)



NUMERO DE CICLOS DE RELOJ:

5 (añadir 1 si salta de página)

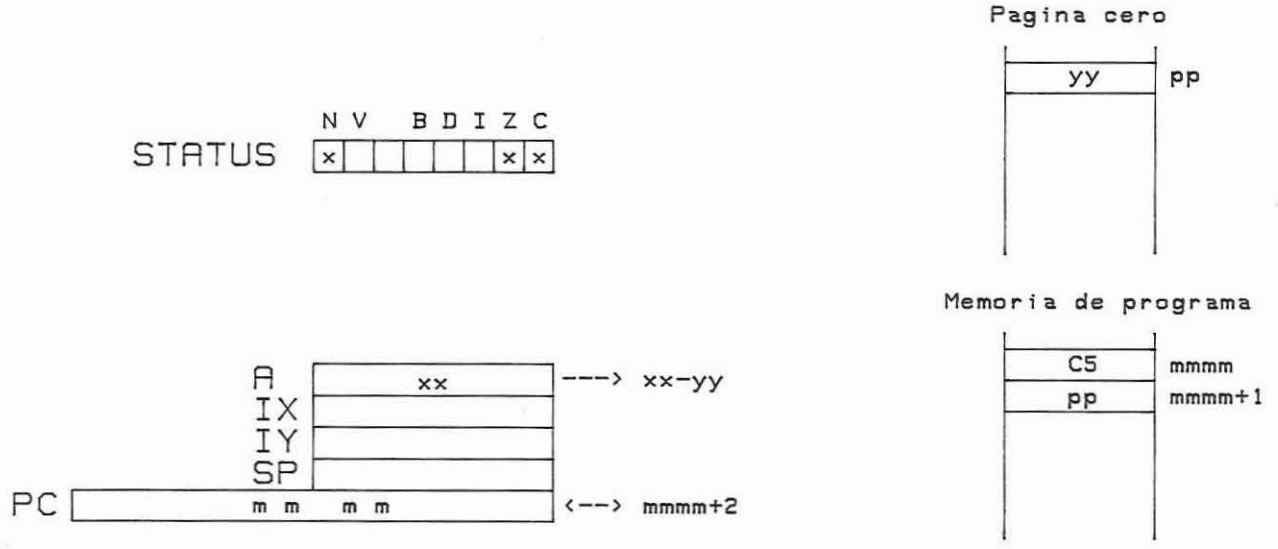
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.
C: Se pone a 1 si hay acarreo.

DESCRIPCION:

El contenido de la posición de memoria que sigue a la que contiene el código de operación contiene una dirección de memoria de página cero. En esta dirección y en la siguiente se encuentran, en orden inverso (primero el byte de menor peso y luego el byte mayor peso), los dos bytes que forman la dirección de memoria que sumada al contenido del registro Y, nos dará la dirección de memoria donde se encuentra el dato. Este dato se resta del Acumulador. El único registro afectado por esta instrucción es el registro de Status.

CMP dir **COMPARAR CON ACUMULADOR**
(Direccionamiento página cero)



NUMERO DE CICLOS DE RELOJ:

3

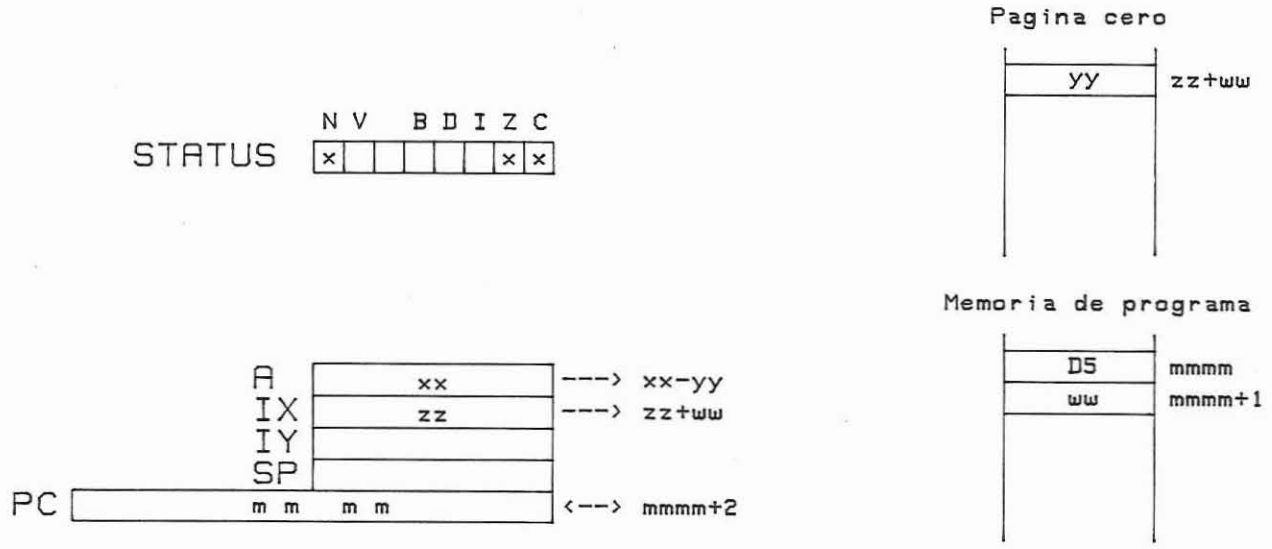
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.
C: Se pone a 1 si hay acarreo.

DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la dirección de memoria de página cero donde se encuentra el dato que se resta del Acumulador. El único registro afectado por esta instrucción es el registro de Status.

CMP dir,X — **COMPARAR CON ACUMULADOR**
(Direccionamiento página cero indexado)



NUMERO DE CICLOS DE RELOJ:

4

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

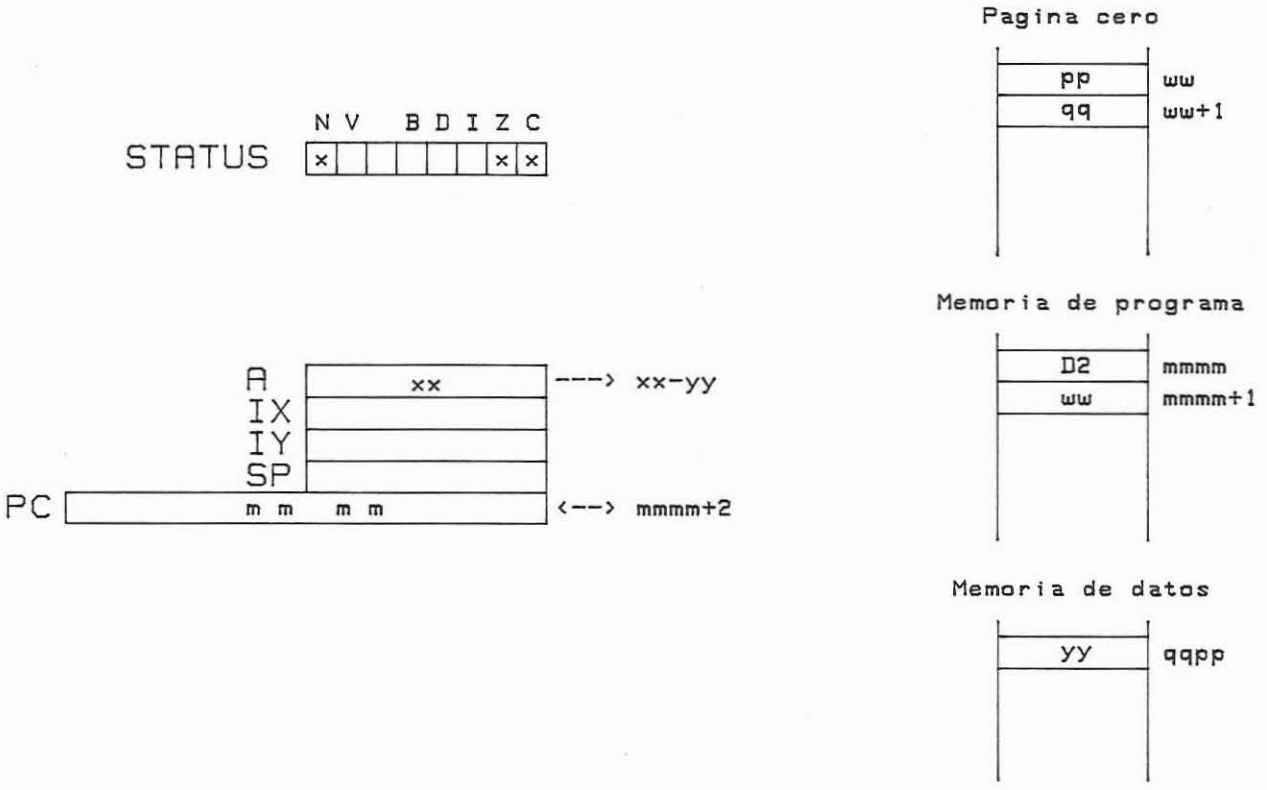
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.
C: Se pone a 1 si hay acarreo.

DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la parte baja de la dirección de memoria que, sumada al contenido del registro X, da la dirección de memoria de página cero donde se encuentra el dato.Este dato se resta del Acumulador. El único registro afectado por esta instrucción es el registro de Status.

CMP (dir) * **COMPARAR CON ACUMULADOR**

(Direccionamiento indirecto mejorado)



NUMERO DE CICLOS DE RELOJ:

5

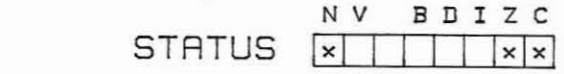
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

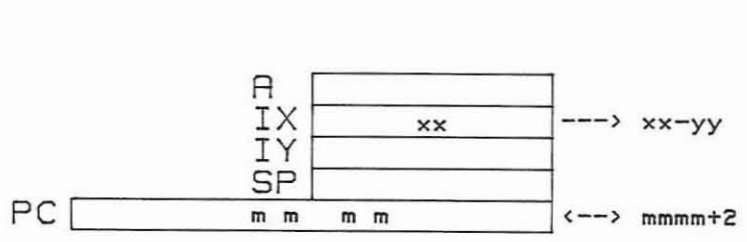
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.
C: Se pone a 1 si hay acarreo.

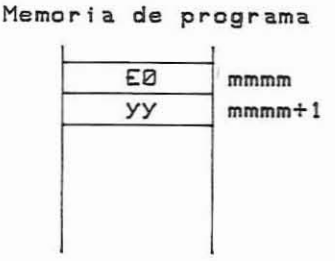
DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la parte baja de una dirección de memoria de página cero. En esta dirección y en la siguiente se encuentran, en orden inverso (primero byte de menor peso y despues byte de mayor peso), los dos bytes que forman la dirección de memoria donde se encuentra el dato que se resta del Acumulador. El único registro afectado por esta instrucción es el registro de Status.

CPX #dato — **COMPARAR CON REGISTRO X**
(Direccionamiento inmediato)







NUMERO DE CICLOS DE RELOJ:

2

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.
C: Se pone a 1 si hay acarreo.

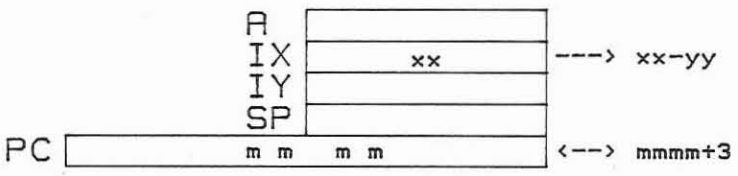
DESCRIPCION:

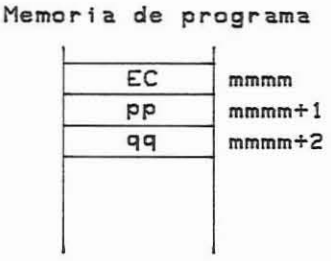
El contenido de la posición de memoria que sigue a la que contiene el código de operación se resta al contenido del registro X. El único registro afectado por esta instrucción es el registro de Status.

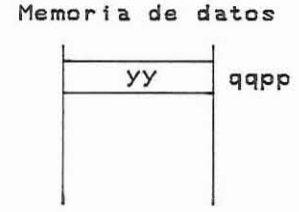
CPX dir

COMPARAR CON REGISTRO X
(Direccionamiento absoluto)









NUMERO DE CICLOS DE RELOJ:

4

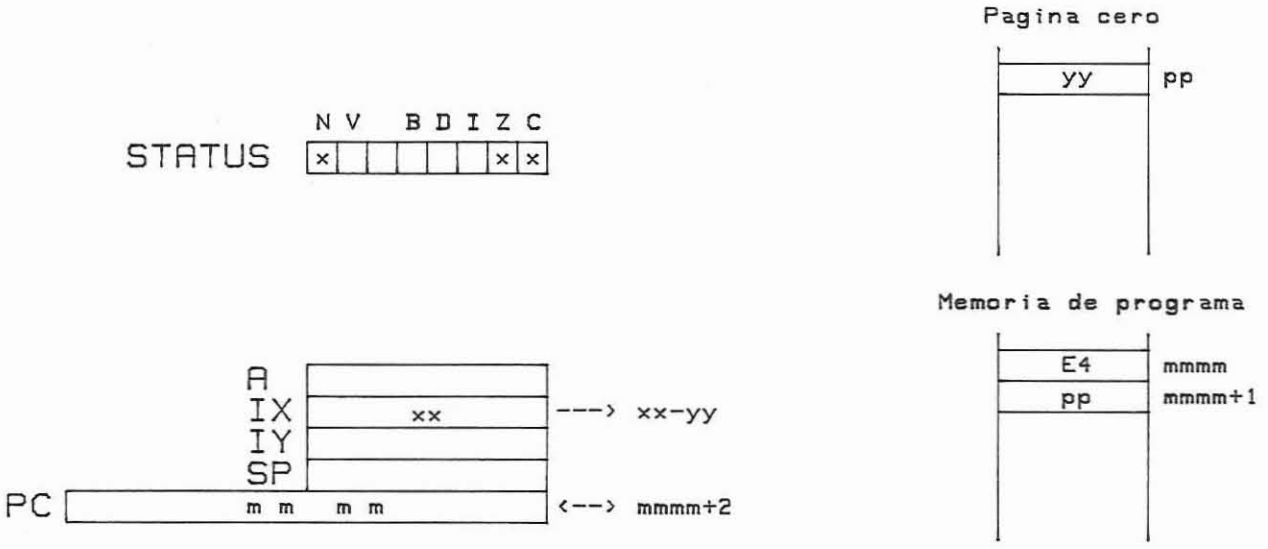
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.
C: Se pone a 1 si hay acarreo.

DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria donde se encuentra el dato que se resta al contenido del registro X. El único registro afectado por esta instrucción es el Registro de Status.

CPX dir **COMPARAR CON REGISTRO X**
(Direccionamiento página cero)



NUMERO DE CICLOS DE RELOJ:

3

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

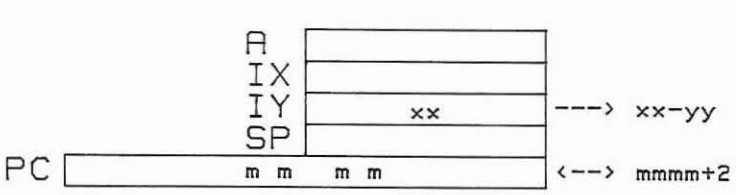
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.
C: Se pone a 1 si hay acarreo.

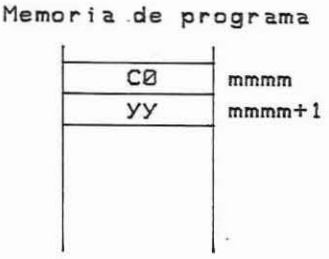
DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la dirección de memoria de página cero donde se encuentra el dato que se resta del registro X. El único registro afectado por esta instrucción es el registro de Status.

CPY #dato **COMPARAR CON REGISTRO Y**
(Direccionamiento inmediato)







NUMERO DE CICLOS DE RELOJ:

2

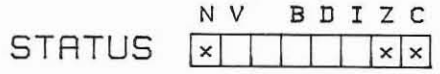
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

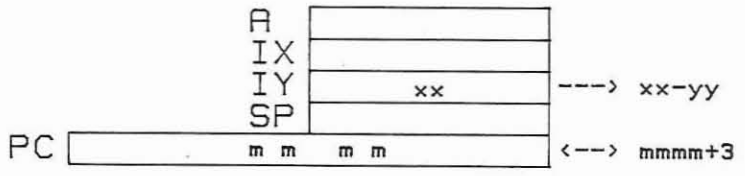
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.
C: Se pone a 1 si hay acarreo.

DESCRIPCION:

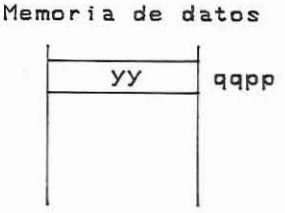
El contenido de la posición de memoria que sigue a la que contiene el código de operación se resta al contenido del registro Y. El único registro afectado por esta instrucción es el registro de Status.

CPY dir — **COMPARAR CON REGISTRO Y**
(Direccionamiento absoluto)









NUMERO DE CICLOS DE RELOJ:

4

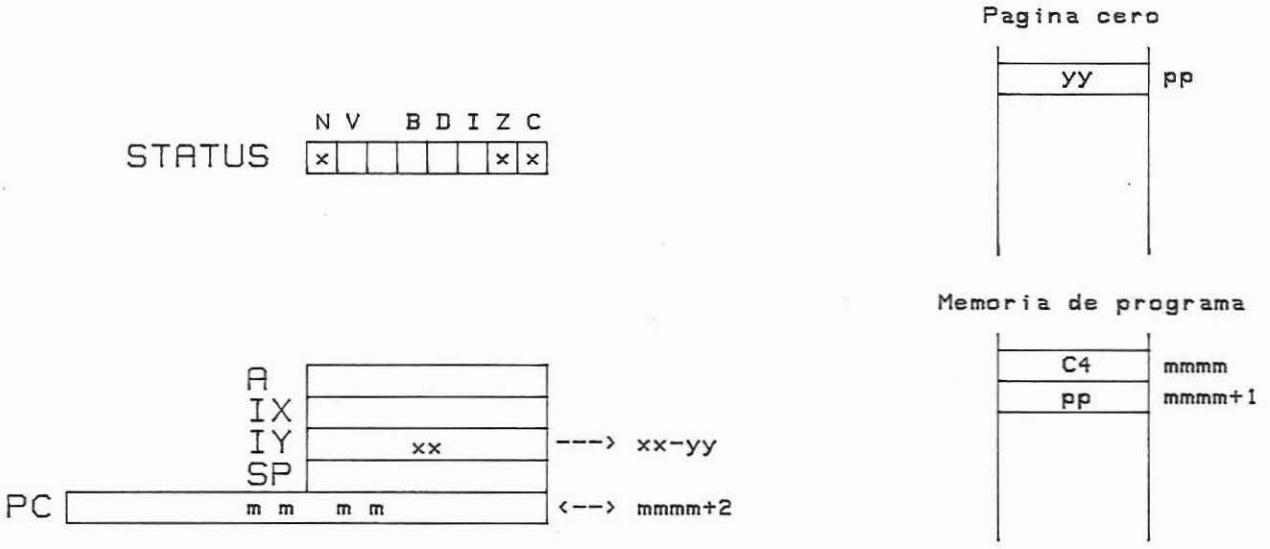
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.
C: Se pone a 1 si hay acarreo.

DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria donde se encuentra el dato que se resta al contenido del registro Y. El único registro afectado por esta instrucción es el Registro de Status.

CPY dir **COMPARAR CON REGISTRO Y**
(Direccionamiento página cero)



NUMERO DE CICLOS DE RELOJ:

3

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

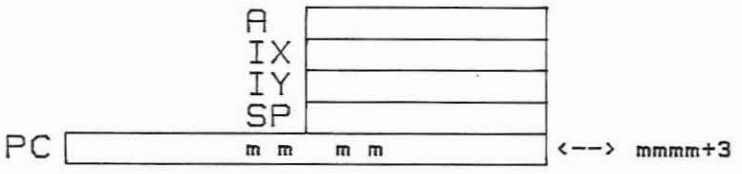
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.
C: Se pone a 1 si hay acarreo.

DESCRIPCION:

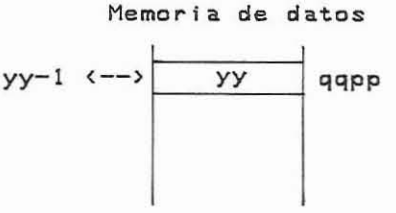
La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la dirección de memoria de página cero donde se encuentra el dato que se resta del registro Y. El único registro afectado por esta instrucción es el registro de Status.

DEC dir **DECREMENTAR EL CONTENIDO DE LA POSICION DE MEMORIA**
(Direccionamiento absoluto)









NUMERO DE CICLOS DE RELOJ:

6

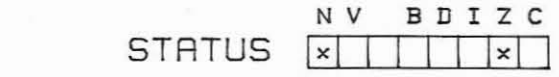
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

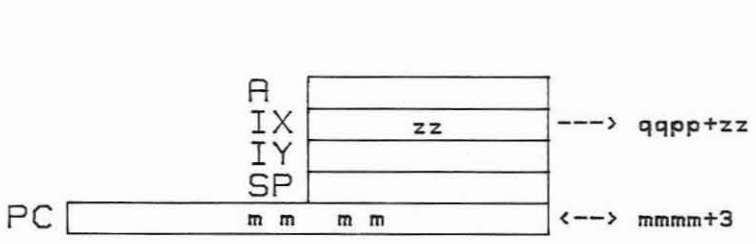
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

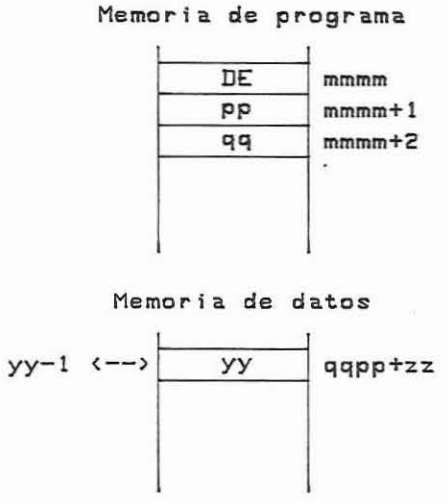
DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria donde se encuentra el dato que se decrementa. El resultado vuelve a quedar en el la misma dirección de memoria.

DEC dir,X **DECREMENTAR EL CONTENIDO DE LA POSICION DE MEMORIA**
(Direccionamiento absoluto indexado)







NUMERO DE CICLOS DE RELOJ:

7

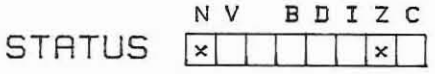
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

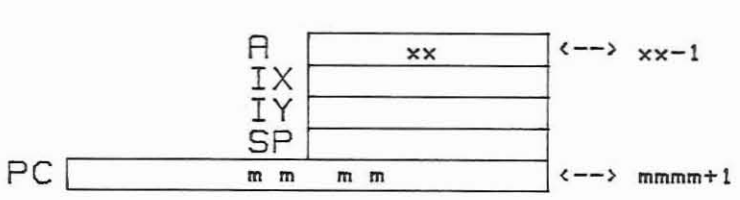
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria que, sumada al contenido del registro X, indica la posición de memoria donde se encuentra el dato que se decrementa. El resultado vuelve a quedar en el la misma dirección de memoria.

DEC A * **DECREMENTAR EL CONTENIDO DEL ACUMULADOR**
(Direccionamiento de Acumulador)







NUMERO DE CICLOS DE RELOJ:

2

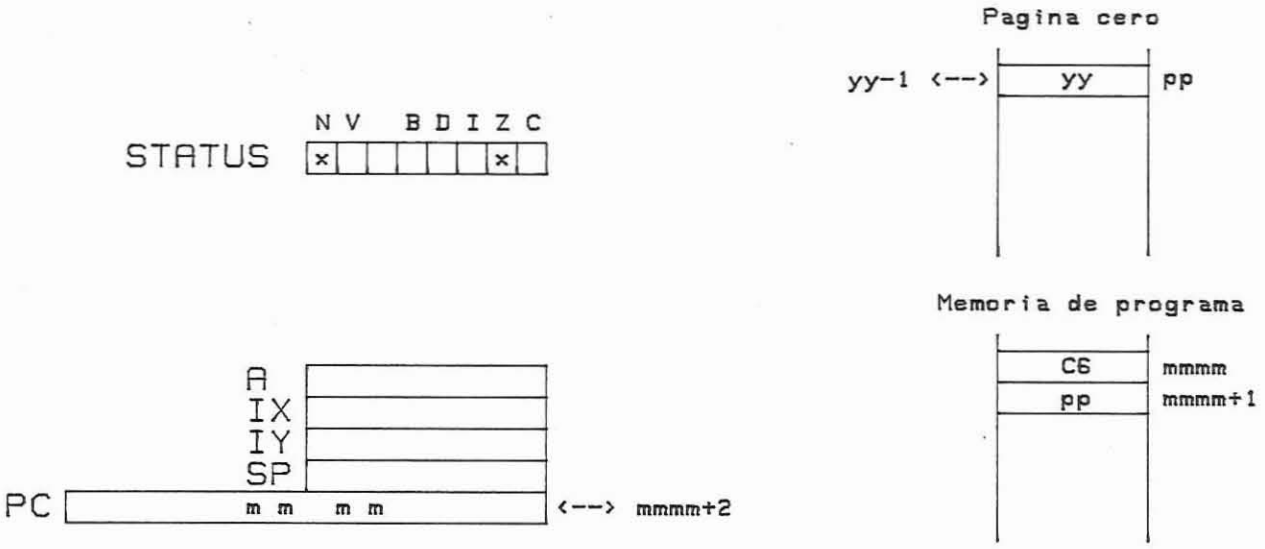
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

El contenido del Acumulador se decrementa. El resultado vuelve a quedar en el Acumulador.

DEC dir **DECREMENTAR EL CONTENIDO DE LA POSICION DE MEMORIA**
(Direccionamiento página cero)



NUMERO DE CICLOS DE RELOJ:

5

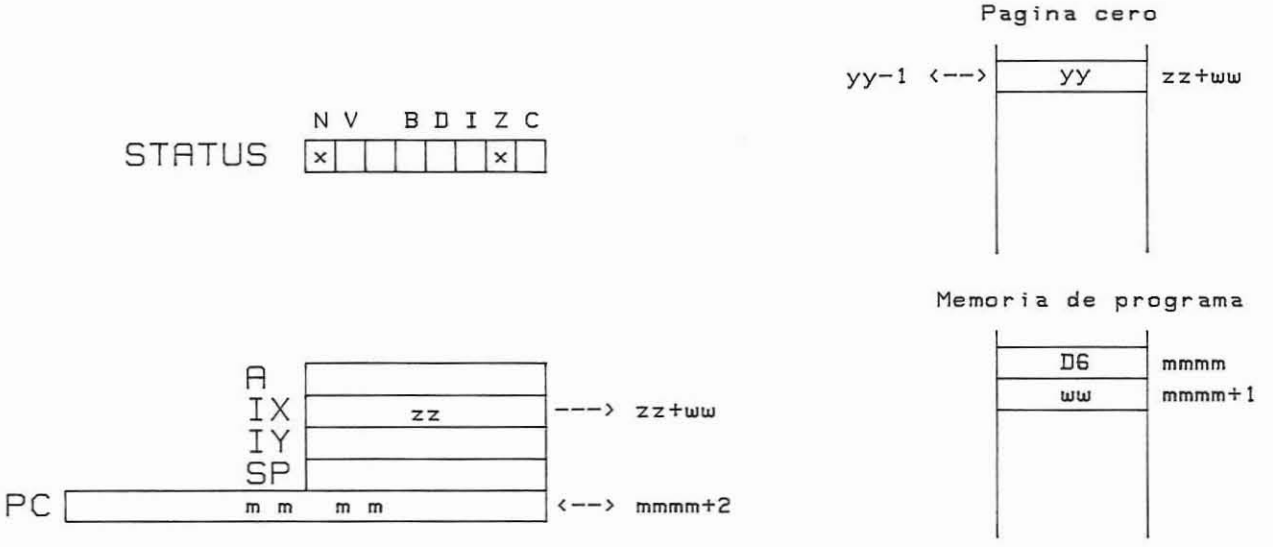
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la dirección de memoria de página cero donde se encuentra el dato que se decrementa. El resultado vuelve a quedar en el la misma dirección de memoria.

DEC dir,X **DECREMENTAR EL CONTENIDO DE LA POSICION DE MEMORIA**
(Direccionamiento página cero indexado)



NUMERO DE CICLOS DE RELOJ:

6

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

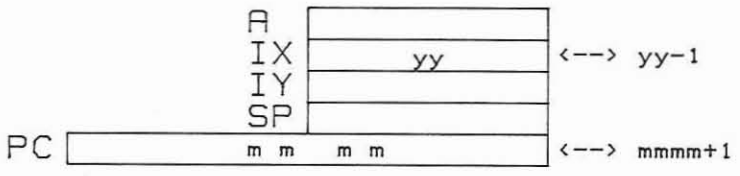
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la parte baja de la dirección de memoria que, sumada al contenido del registro X, da la dirección de memoria de página cero donde se encuentra el dato que se decrementa. El resultado vuelve a quedar en el la misma dirección de memoria.

DEX **DECREMENTAR EL CONTENIDO DEL REGISTRO X**
(Direccionamiento implícito)







NUMERO DE CICLOS DE RELOJ:

2

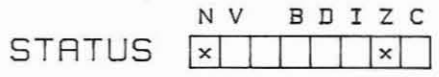
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

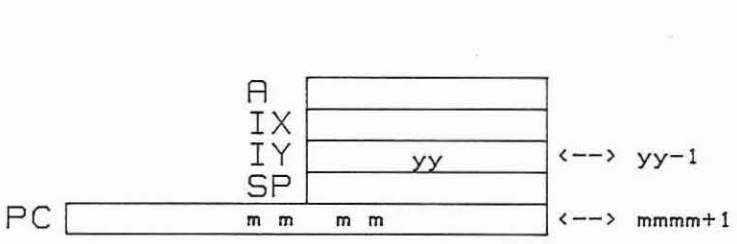
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

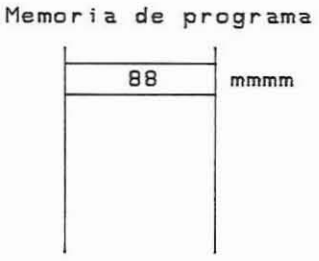
DESCRIPCION:

El contenido del registro X se decrementa. El resultado vuelve a quedar en el mismo registro.

DEY **DECREMENTAR EL CONTENIDO DEL REGISTRO Y**
(Direccionamiento implícito)







NUMERO DE CICLOS DE RELOJ:

2

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

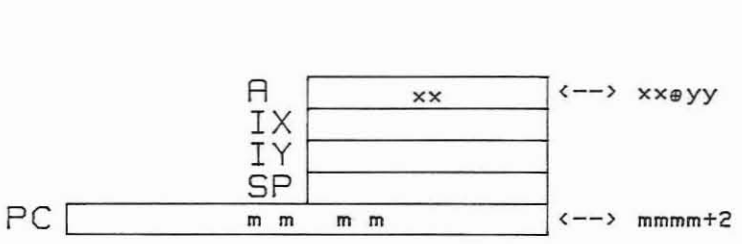
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

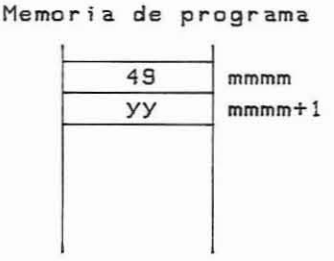
DESCRIPCION:

El contenido del registro Y se decrementa. El resultado vuelve a quedar en el mismo registro.

EOR #dato **FUNCION OR-EXCLUSIVA**
(Direccionamiento inmediato)







NUMERO DE CICLOS DE RELOJ:

2

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

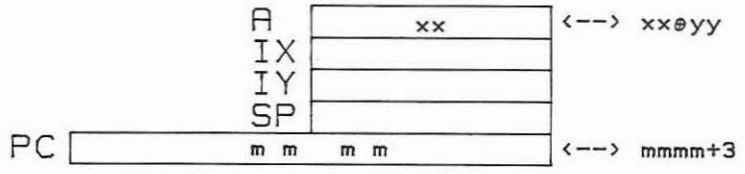
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

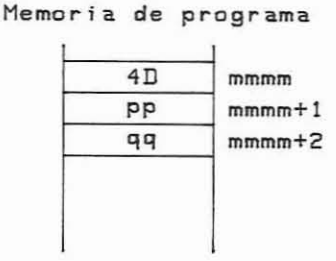
DESCRIPCION:

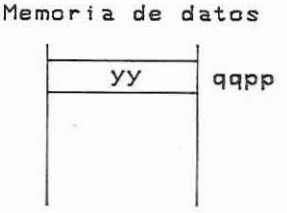
El contenido de la posición de memoria que sigue a la que contiene el código de operación efectúa bit a bit la operación OR-EXCLUSIVA con el Acumulador. El resultado queda en el Acumulador.

EOR dir **FUNCION OR_EXCLUSIVA**
(Direccionamiento absoluto)









NUMERO DE CICLOS DE RELOJ:

4

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

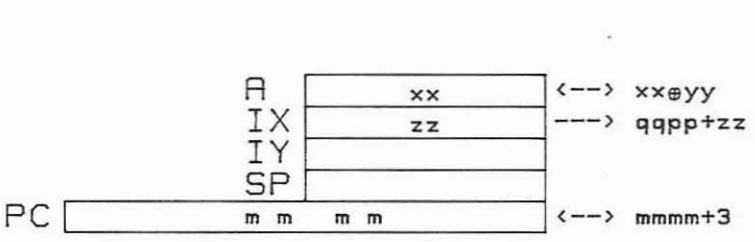
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria donde se encuentra el dato que efectúa bit a bit la operación OR-EXCLUSIVA con el Acumulador. El resultado queda en el Acumulador.

EOR dir,X — **FUNCION OR_EXCLUSIVA**
(Direccionamiento absoluto indexado)







NUMERO DE CICLOS DE RELOJ:

4 (añadir 1 si salta de página)

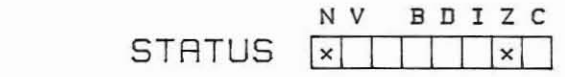
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

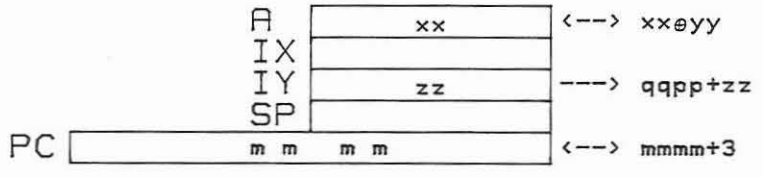
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

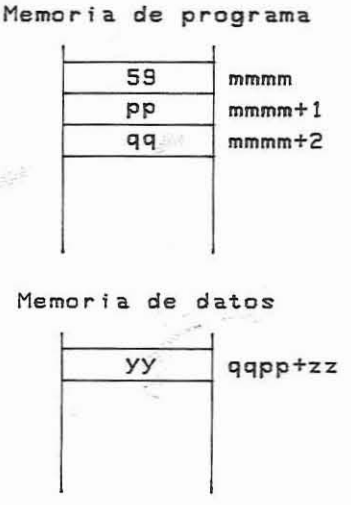
DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte de menor peso y luego el byte de mayor peso), la dirección de memoria que, sumada al contenido del registro X, indica la posición de memoria donde se encuentra el dato. Este dato efectúa bit a bit la operación OR-EXCLUSIVA con el Acumulador. El resultado queda en el Acumulador.

EOR dir,Y — **FUNCION OR-EXCLUSIVA**
(Direccionamiento absoluto indexado)







NUMERO DE CICLOS DE RELOJ:

4 (añadir 1 si salta de página)

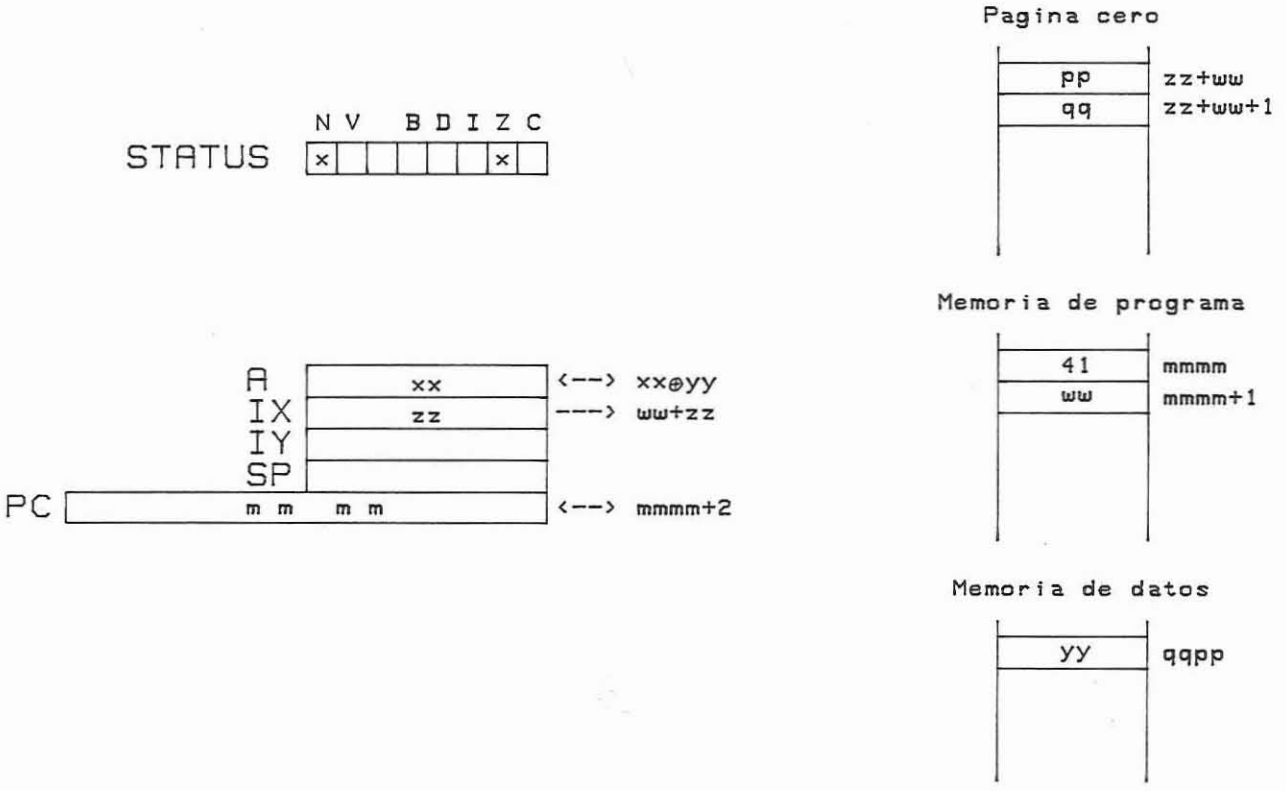
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria que, sumada al contenido del registro Y, indica la posición de memoria donde se encuentra el dato. Este dato efectúa bit a bit la operación OR-EXCLUSIVA con el Acumulador. El resultado queda en el Acumulador.

EOR (dir,X) **FUNCION OR-EXCLUSIVA**
(Direccionamiento preindexado)



NUMERO DE CICLOS DE RELOJ:

6

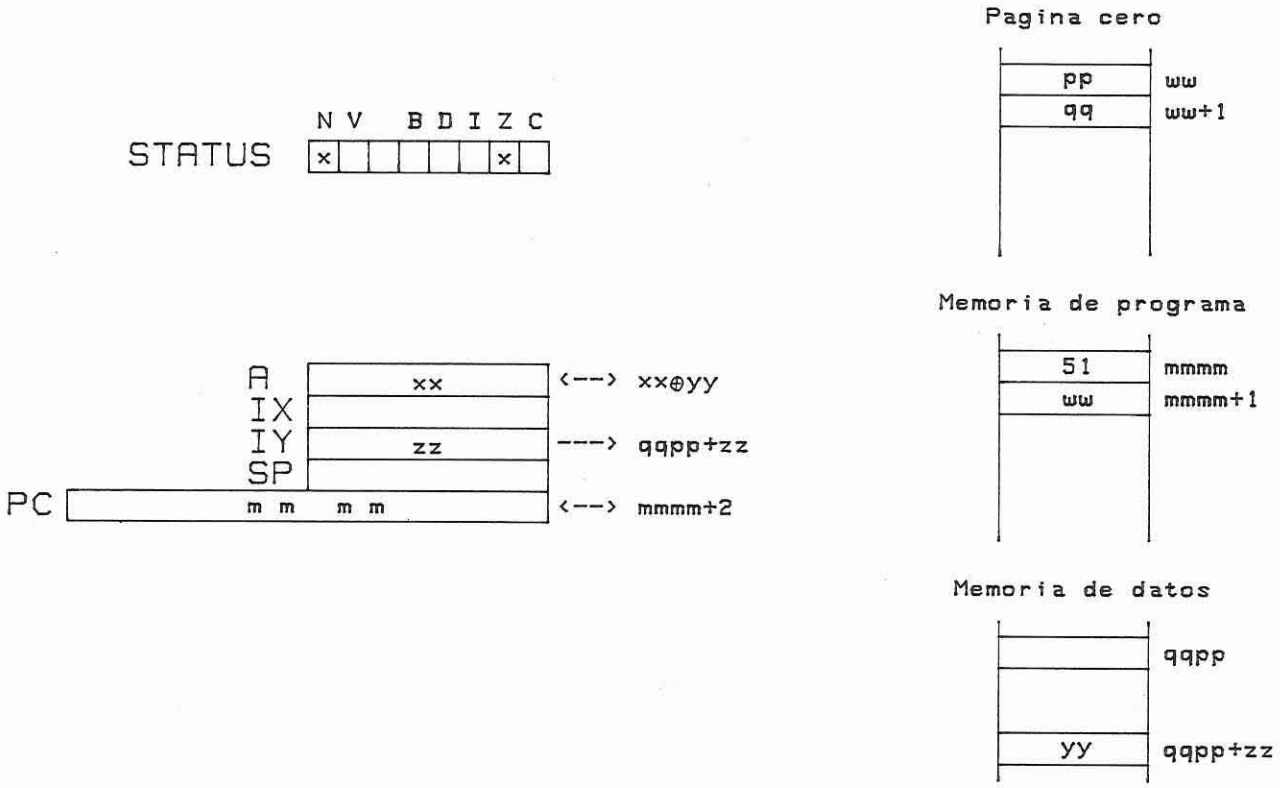
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

El contenido de la posición de memoria que sigue a la que contiene el código de operación se suma con el contenido del registro X. El resultado (despreciando el acarreo si se ha producido) corresponde a la dirección de memoria en página cero que contiene la parte baja de la dirección efectiva del dato. La parte alta de la dirección efectiva del dato está en la siguiente posición de memoria en pagina cero. De la dirección efectiva del dato se toma el byte que efectúa bit a bit la operación OR-EXCLUSIVA con el contenido del Acumulador. El resultado se carga de nuevo en el Acumulador.

EOR (dir),Y — **FUNCION OR-EXCLUSIVA**
(Direccionamiento postindexado)



NUMERO DE CICLOS DE RELOJ:

5 (añadir 1 si salta de página)

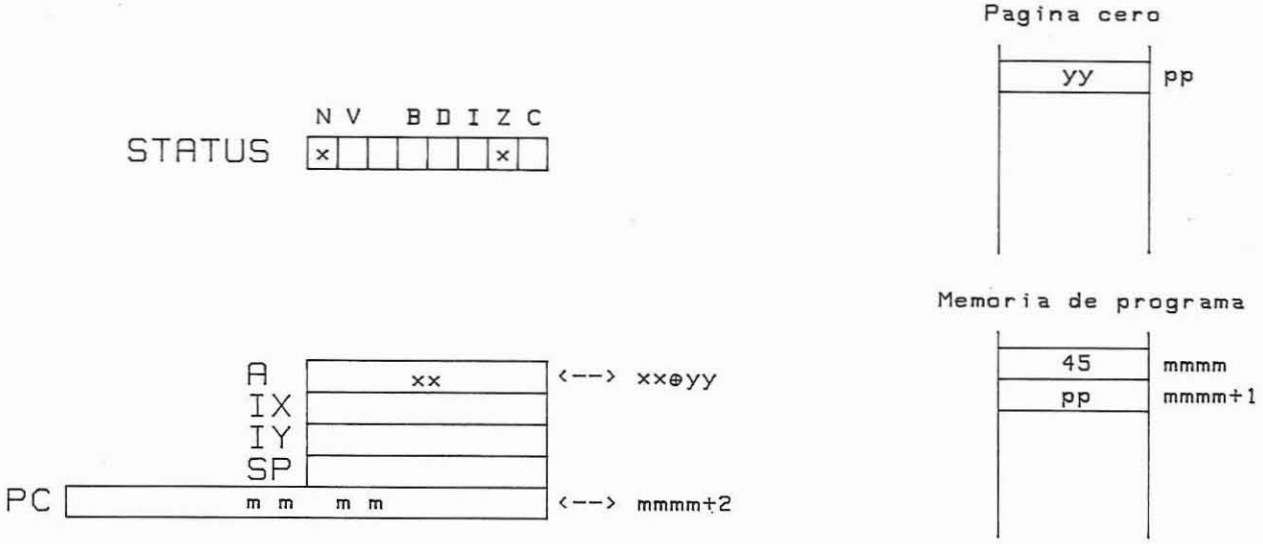
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

El contenido de la posición de memoria que sigue a la que contiene el código de operación contiene una dirección de memoria de página cero. En esta dirección y en la siguiente se encuentran, en orden inverso (primero el byte de menor peso y luego el byte mayor peso), los dos bytes que forman la dirección de memoria que sumada al contenido del registro Y, nos dará la dirección de memoria donde se encuentra el dato. Este dato efectúa la operación OR-EXCLUSIVA con el Acumulador. El resultado se guarda de nuevo en el Acumulador.

EOR dir **FUNCION OR-EXCLUSIVA**
(Direccionamiento página cero)



NUMERO DE CICLOS DE RELOJ:

3

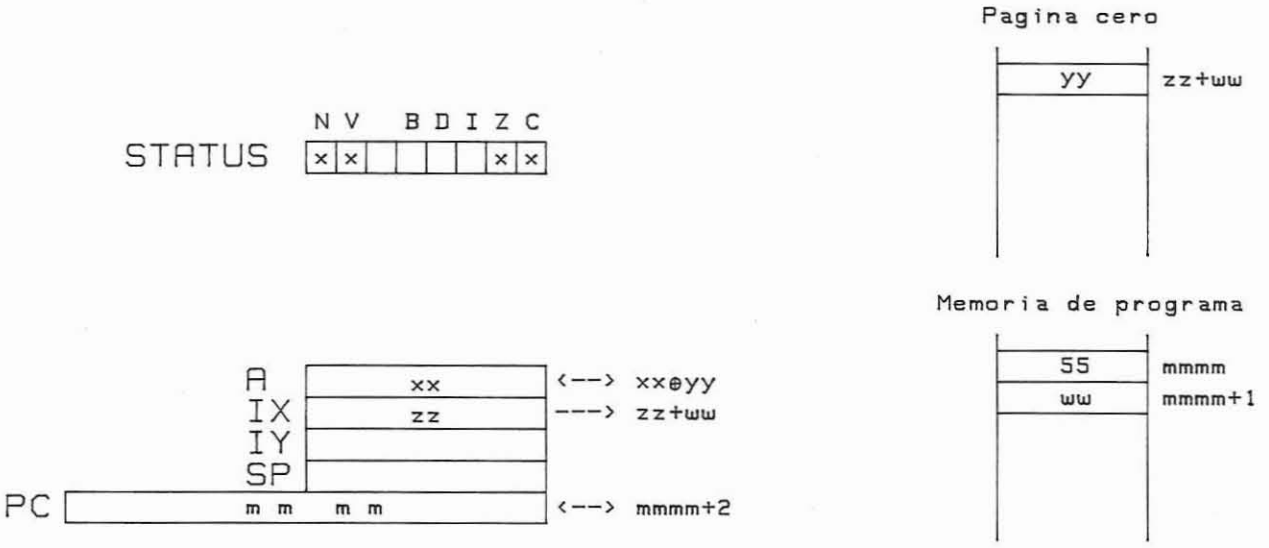
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la dirección de memoria de página cero donde se encuentra el dato que efectúa bit a bit la operación OR-EXCLUSIVA con el Acumulador. El resultado queda en el Acumulador.

EOR dir,X **FUNCION OR-EXCLUSIVA**
(Direccionamiento página cero indexado)



NUMERO DE CICLOS DE RELOJ:

4

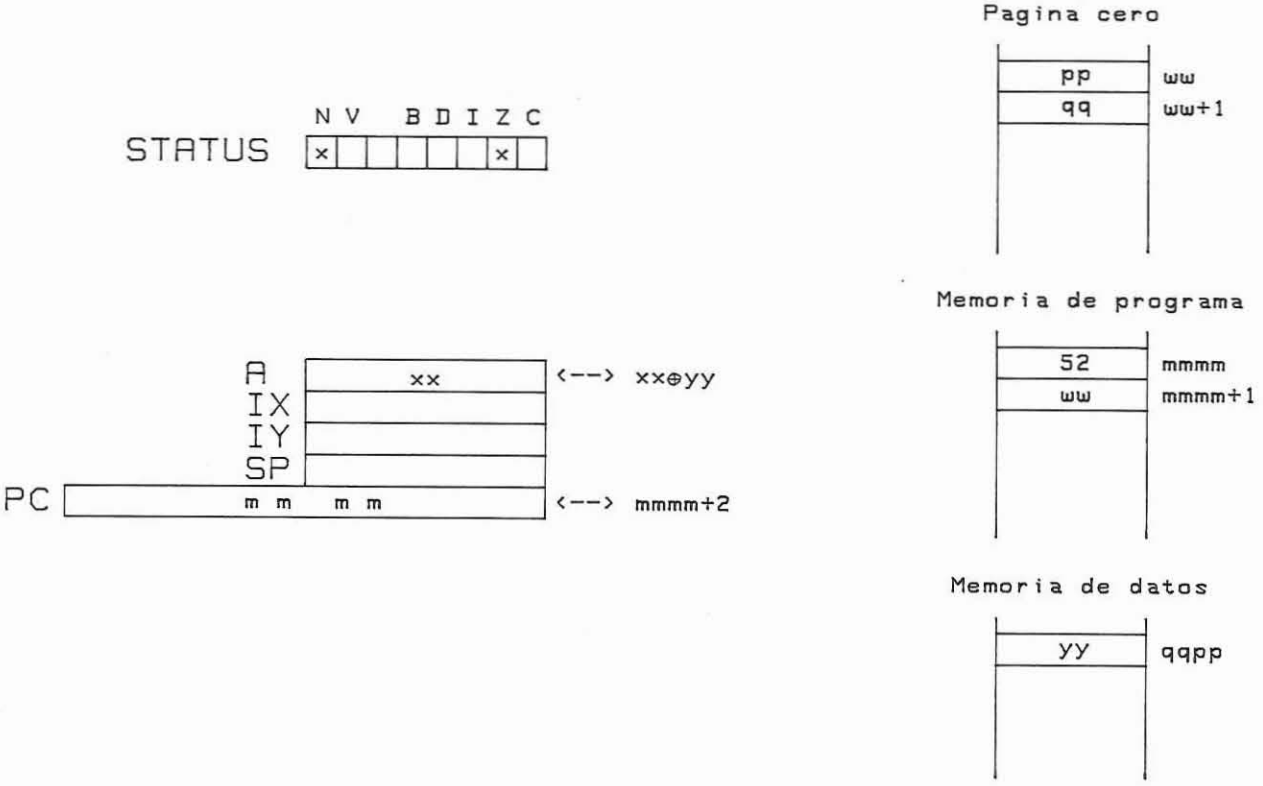
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la parte baja de la dirección de memoria que, sumada al contenido del registro X, da la dirección de memoria de página cero donde se encuentra el dato.Este dato efectúa bit a bit la operación OR-EXCLUSIVA con el Acumulador. El resultado queda en el Acumulador.

EOR (dir) * **FUNCION OR-EXCLUSIVA**
(Direccionamiento indirecto mejorado)



NUMERO DE CICLOS DE RELOJ:

5

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

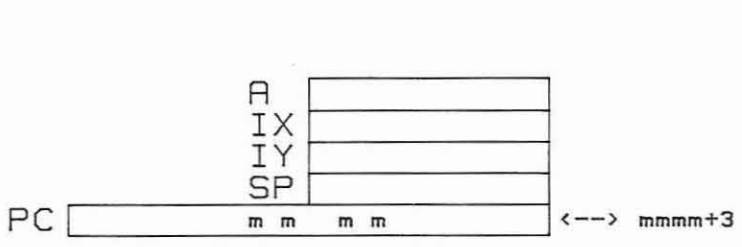
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

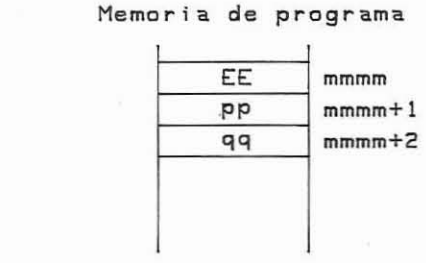
DESCRIPCION:

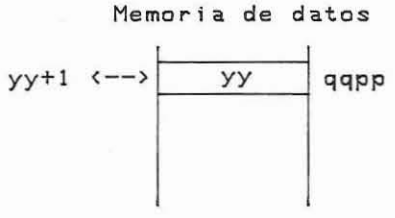
La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la parte baja de una dirección de memoria de página cero. En esta dirección y en la siguiente se encuentran, en orden inverso (primero byte de menor peso y despues byte de mayor peso), los dos bytes que forman la dirección de memoria donde se encuentra el dato que efectúa bit a bit la operación OR-EXCLUSIVA con el Acumulador. El resultado queda en el Acumulador.

INC dir **INCREMENTAR EL CONTENIDO DE LA POSICION DE MEMORIA**
(Direccionamiento absoluto)









NUMERO DE CICLOS DE RELOJ:

6

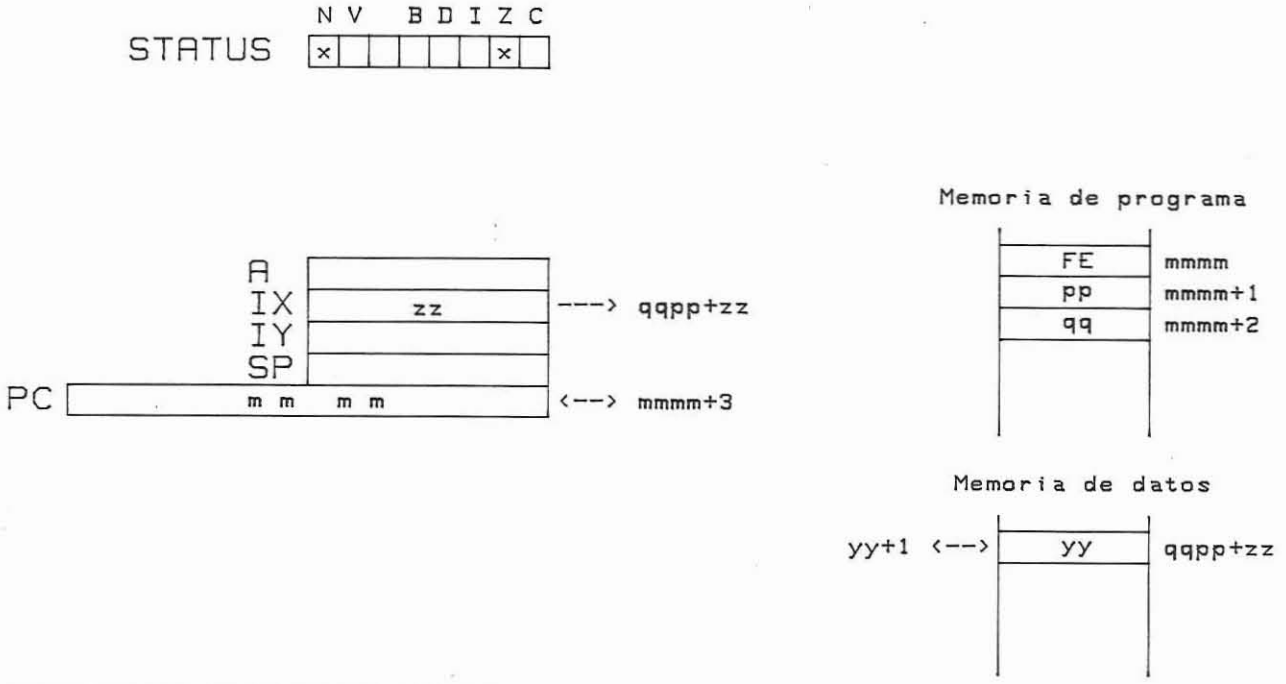
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria donde se encuentra el dato que se incrementa. El resultado vuelve a quedar en el la misma dirección de memoria.

INC dir,X **INCREMENTAR EL CONTENIDO DE LA POSICION DE MEMORIA**
(Direccionamiento absoluto indexado)



NUMERO DE CICLOS DE RELOJ:

7

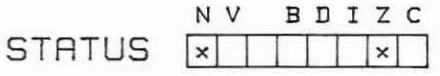
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

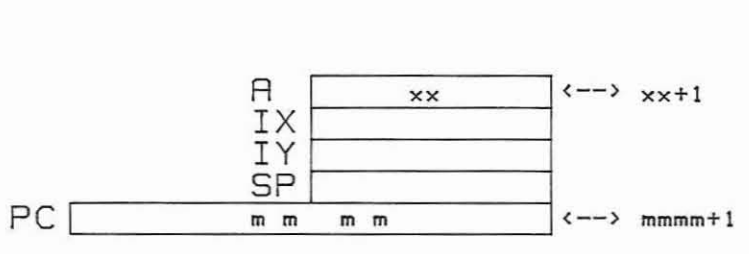
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria que, sumada al contenido del registro X, indica la posición de memoria donde se encuentra el dato que se incrementa. El resultado vuelve a quedar en el la misma dirección de memoria.

INC A * **INCREMENTAR EL CONTENIDO DEL ACUMULADOR**
(Direccionamiento de Acumulador)







NUMERO DE CICLOS DE RELOJ:

2

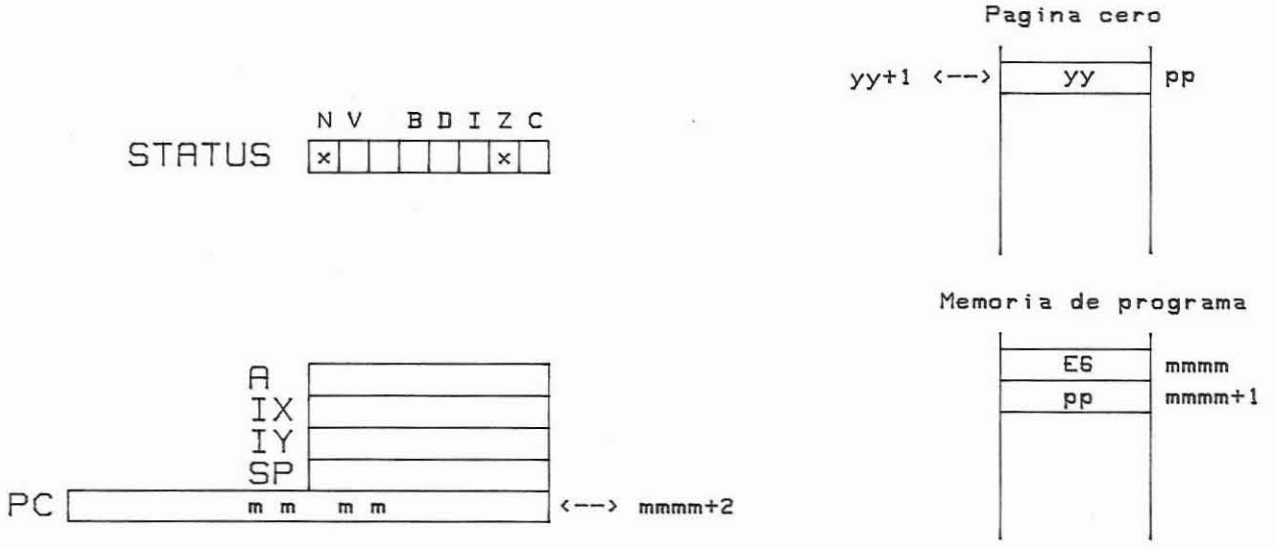
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

El contenido del Acumulador se incrementa. El resultado vuelve a quedar en el Acumulador.

INC dir **INCREMENTAR EL CONTENIDO DE LA POSICION DE MEMORIA**
(Direccionamiento página cero)



NUMERO DE CICLOS DE RELOJ:

6

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

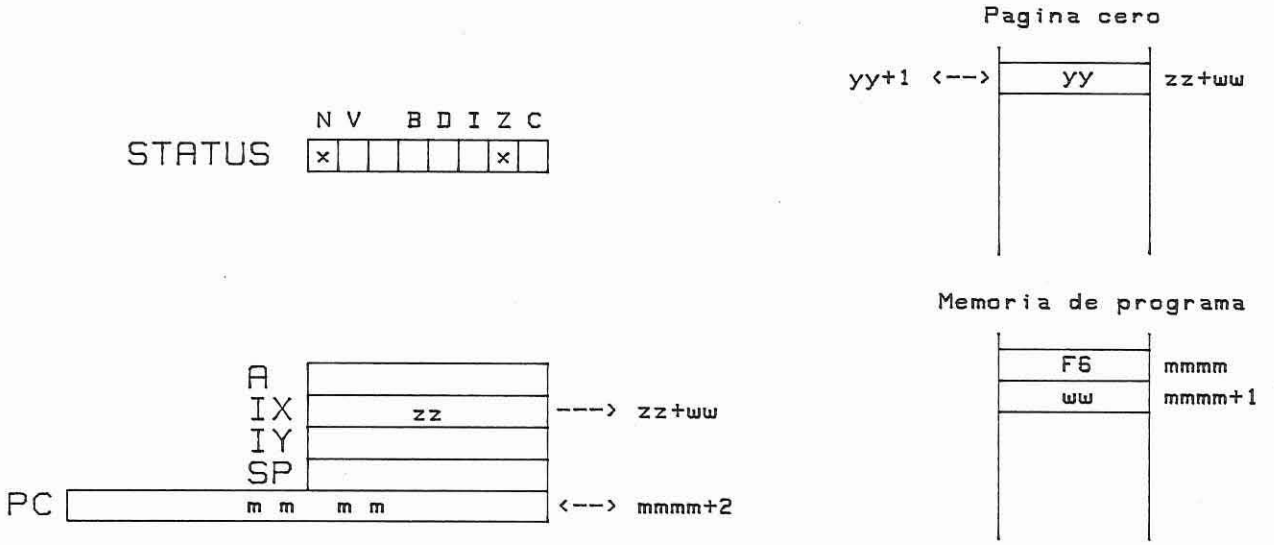
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la dirección de memoria de página cero donde se encuentra el dato que se incrementa. El resultado vuelve a quedar en el la misma dirección de memoria.

INC dir,X INCREMENTAR EL CONTENIDO DE LA POSICION DE MEMORIA

(Direccionamiento página cero indexado)



NUMERO DE CICLOS DE RELOJ:

6

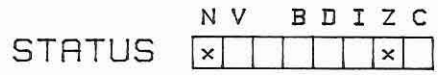
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

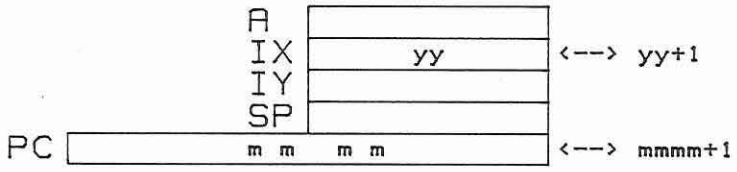
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la parte baja de la dirección de memoria que, sumada al contenido del registro X, da la dirección de memoria de página cero donde se encuentra el dato que se incrementa. El resultado vuelve a quedar en el la misma dirección de memoria.

INX **INCREMENTAR EL CONTENIDO DEL REGISTRO X**
(Direccionamiento implícito)







NUMERO DE CICLOS DE RELOJ:

2

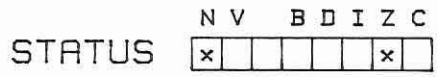
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

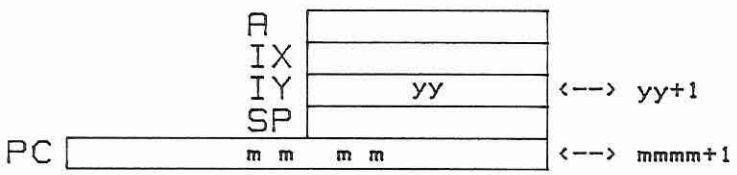
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

El contenido del registro X se incrementa. El resultado vuelve a quedar en el mismo registro.

INY **INCREMENTAR EL CONTENIDO DEL REGISTRO Y**
(Direccionamiento implícito)







NUMERO DE CICLOS DE RELOJ:

2

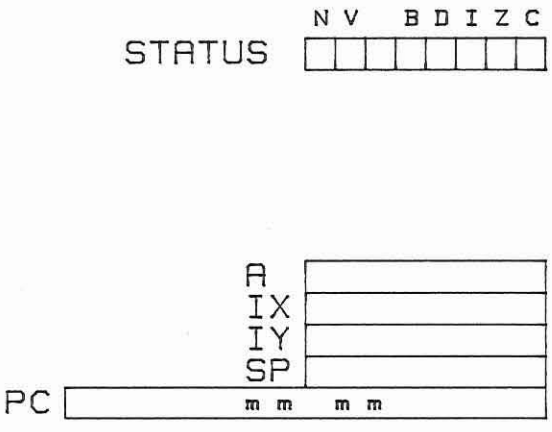
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

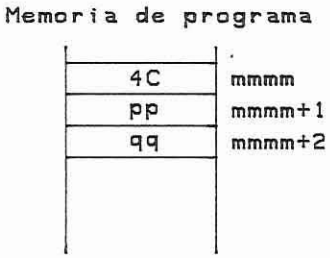
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

El contenido del registro Y se incrementa. El resultado vuelve a quedar en el mismo registro.

JMP dir — **SALTO INCONDICIONAL**
(Direccionamiento absoluto)





NUMERO DE CICLOS DE RELOJ:

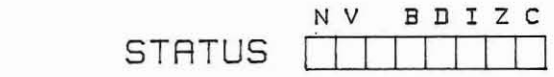
3

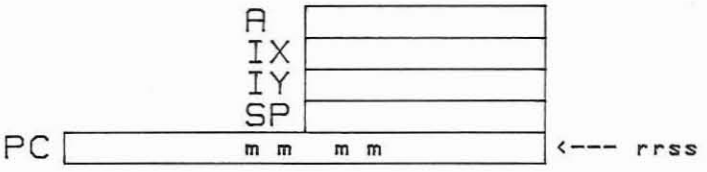
FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

DESCRIPCION:

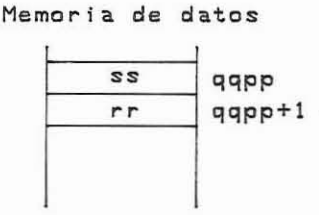
Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria donde se encuentra la siguiente instrucción a ejecutar. Por tanto esta dirección se cargará en el 'Program Counter'.

JMP (dir) **SALTO INCONDICIONAL**
(Direccionamiento indirecto)









NUMERO DE CICLOS DE RELOJ:

5

FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

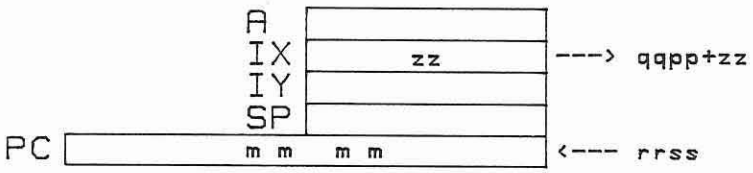
DESCRIPCION:

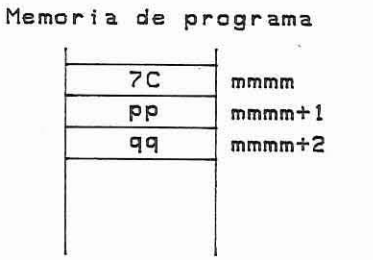
Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria donde se encuentra el byte bajo de la dirección de la siguiente instrucción a ejecutar. El byte alto de dicha dirección se encuentra en la siguiente posición. Por tanto esta dirección se cargará en el 'Program Counter'.

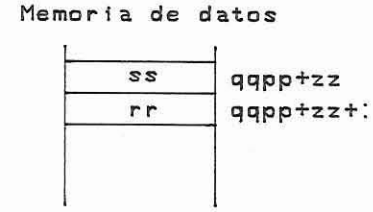
JMP (dir,x) *

SALTO INCONDICIONAL
(Direccionamiento indirecto indexado)









NUMERO DE CICLOS DE RELOJ:

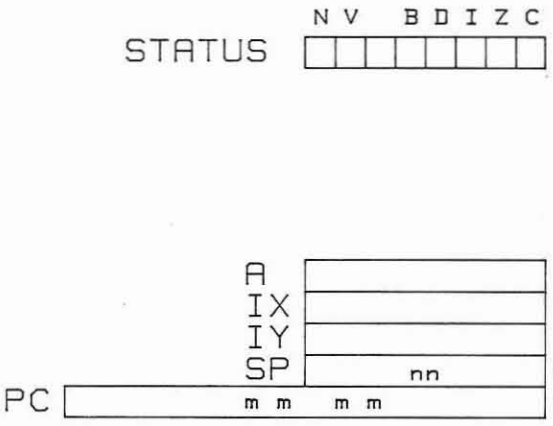
6

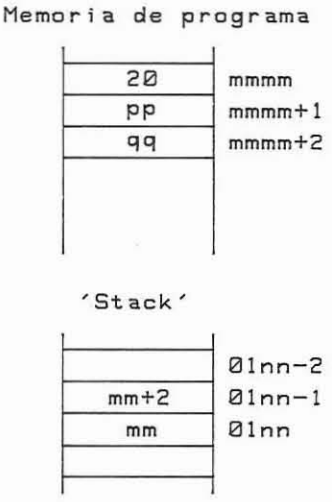
FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria donde se encuentra el byte bajo de la dirección que sumada al contenido del registro X dará la dirección de la siguiente instrucción a ejecutar. El byte alto de dicha dirección se encuentra en la siguiente posición. La dirección de la siguiente instrucción a ejecutar se cargará en el 'Program Counter'.

JSR dir **SALTO A SUBRUTINA**
(Direccionamiento absoluto)





NUMERO DE CICLOS DE RELOJ:

6

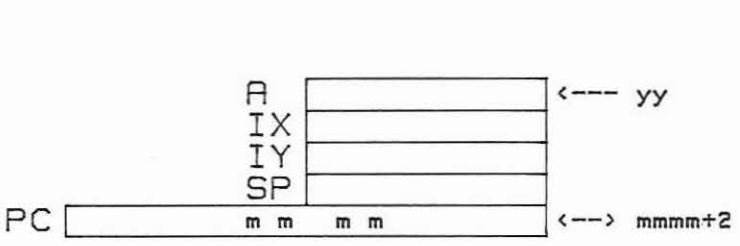
FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

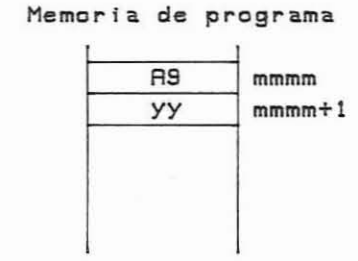
DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria donde se encuentra la siguiente instrucción a ejecutar. Por tanto esta dirección se cargará en el 'Program Counter'. La dirección de retorno de esta subrutina será la que corresponde a la instrucción que sigue a la de salto a subrutina. Esta dirección de retorno queda almacenada en el Stack y el Puntero de Pila o Stack Pointer queda apuntando a la siguiente posición libre de la pila.

LDA #dato — **CARGAR EL ACUMULADOR**
(Direccionamiento inmediato)







NUMERO DE CICLOS DE RELOJ:

2

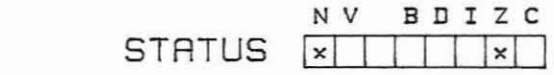
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

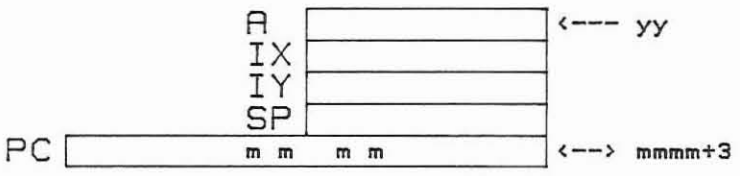
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

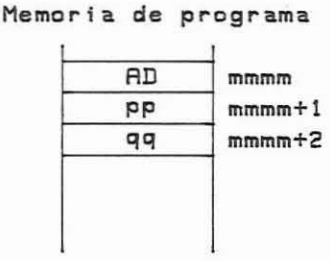
DESCRIPCION:

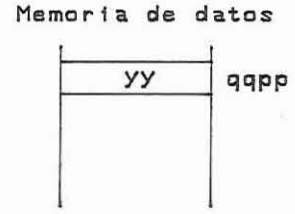
El contenido de la posición de memoria que sigue a la que contiene el código de operación se carga en el Acumulador.

LDA dir — **CARGAR EL ACUMULADOR**
(Direccionamiento absoluto)









NUMERO DE CICLOS DE RELOJ:

4

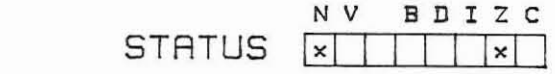
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

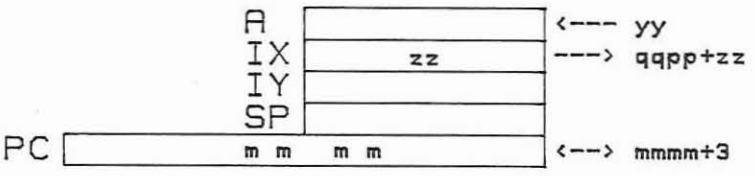
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

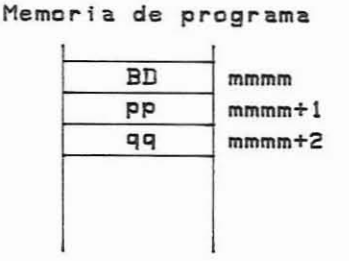
DESCRIPCION:

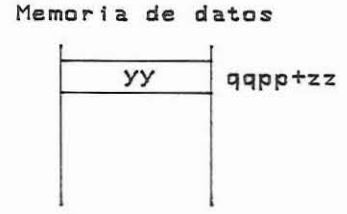
Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria donde se encuentra el dato que se carga en el Acumulador.

LDA dir,X — **CARGAR EL ACUMULADOR**
(Direccionamiento absoluto indexado)









NUMERO DE CICLOS DE RELOJ:

4 (añadir 1 si salta de página)

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

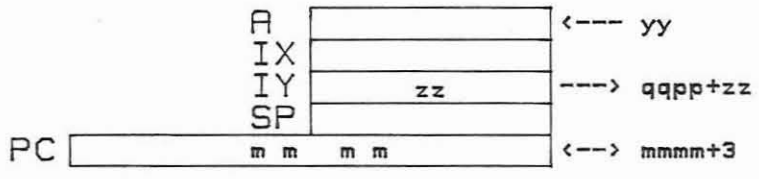
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

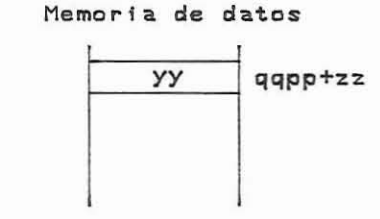
Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria que, sumada al contenido del registro X, indica la posición de memoria donde se encuentra el dato que se carga en el Acumulador.

LDA dir,Y — **CARGAR EL ACUMULADOR**
(Direccionamiento absoluto indexado)









NUMERO DE CICLOS DE RELOJ:

4 (añadir 1 si salta de página)

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

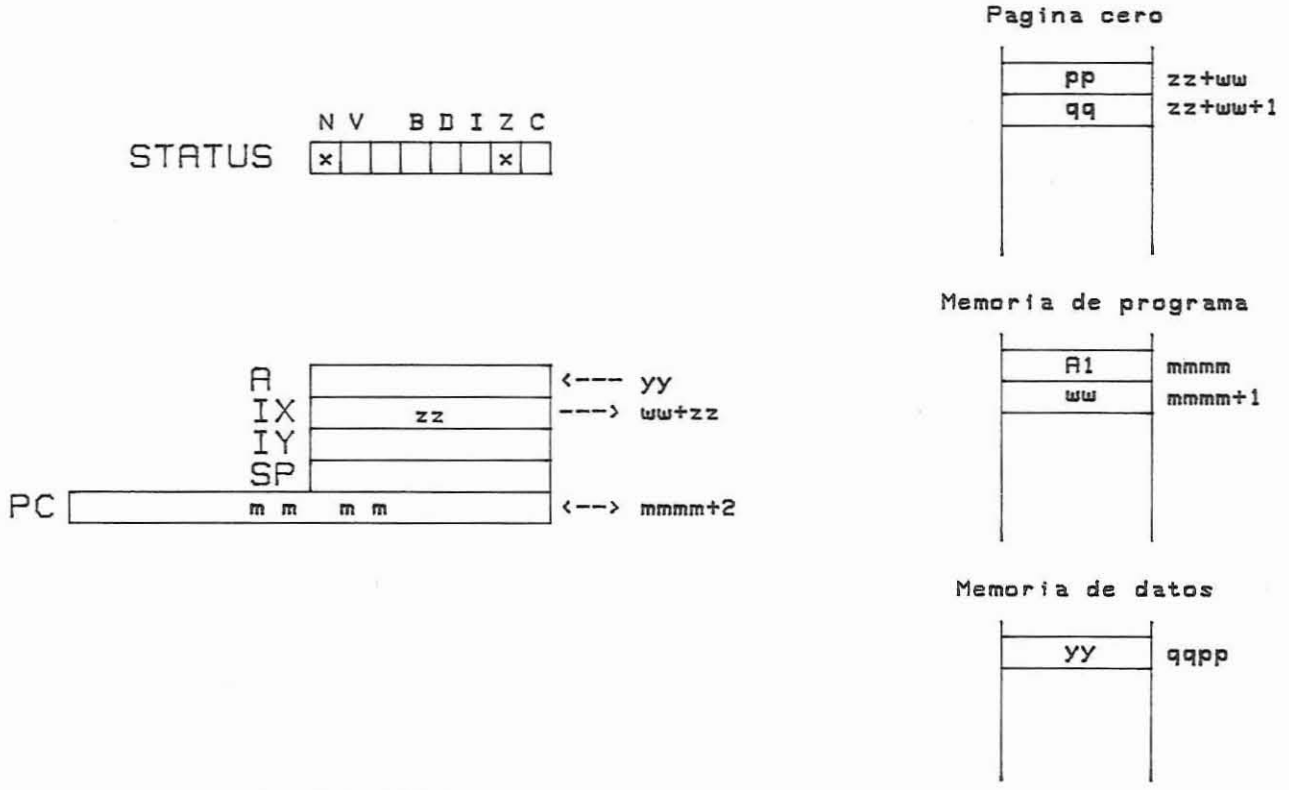
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria que, sumada al contenido del registro Y, indica la posición de memoria donde se encuentra el dato que se carga en el Acumulador.

LDA (dir,X)

CARGAR EL ACUMULADOR
(Direccionamiento preindexado)



NUMERO DE CICLOS DE RELOJ:

6

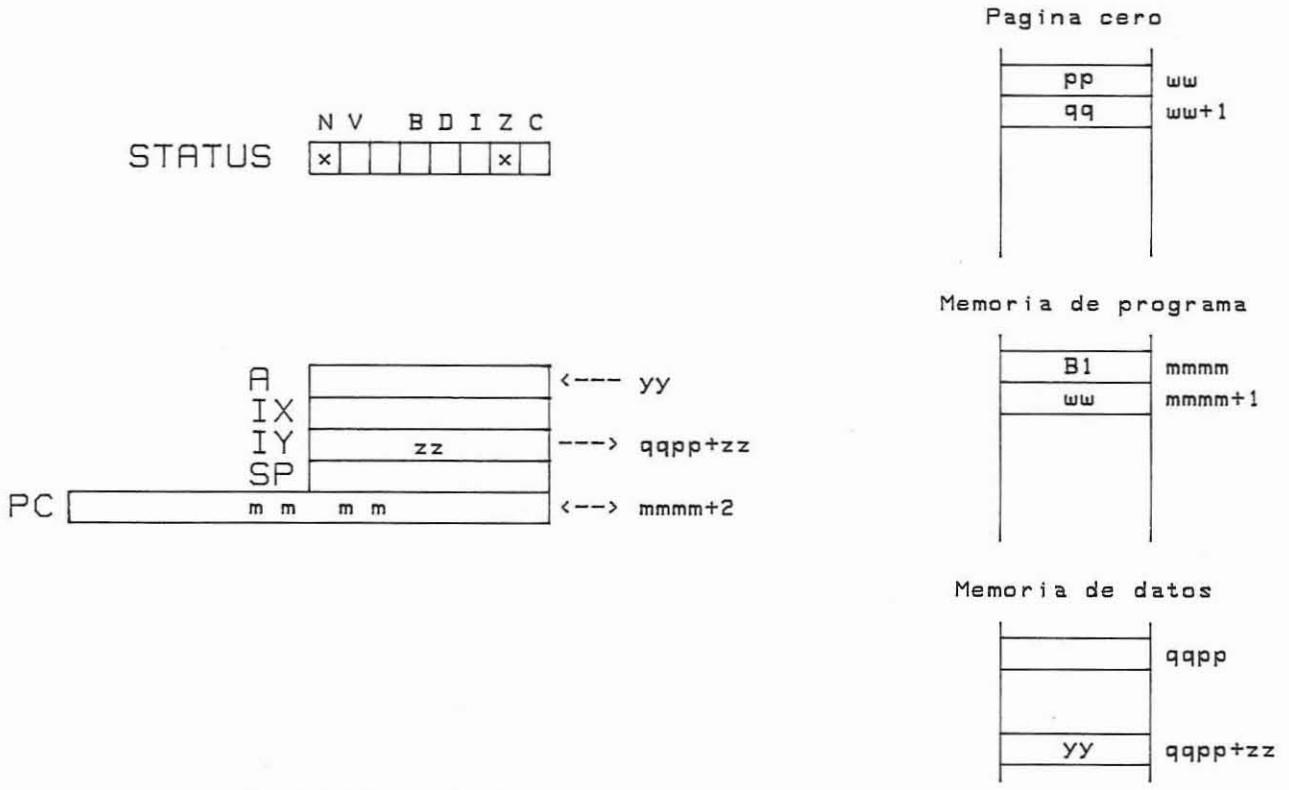
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

El contenido de la posición de memoria que sigue a la que contiene el código de operación se suma con el contenido del registro X. El resultado (despreciando el acarreo si se ha producido) corresponde a la dirección de memoria en página cero que contiene la parte baja de la dirección efectiva del dato. La parte alta de la dirección efectiva del dato está en la siguiente posición de memoria en pagina cero. De la dirección efectiva del dato se toma el byte que se carga en el Acumulador.

LDA (dir),Y **CARGAR EL ACUMULADOR**
(Direccionamiento postindexado)



NUMERO DE CICLOS DE RELOJ:

5 (añadir 1 si salta de página)

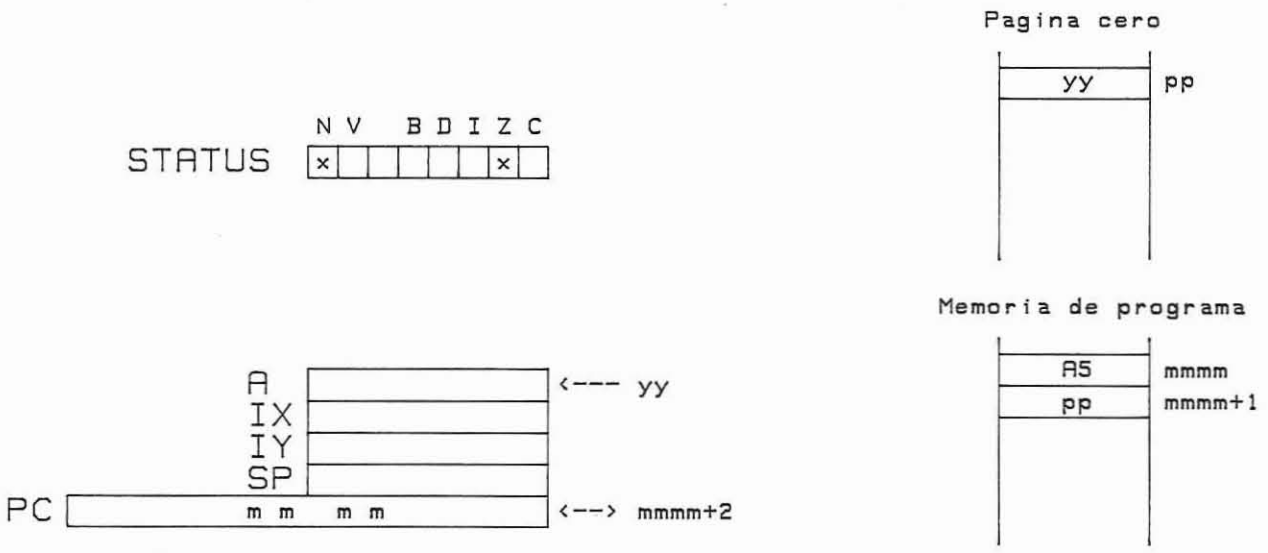
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

El contenido de la posición de memoria que sigue a la que contiene el código de operación contiene una dirección de memoria de página cero. En esta dirección y en la siguiente se encuentran, en orden inverso (primero el byte de menor peso y luego el byte mayor peso), los dos bytes que forman la dirección de memoria que sumada al contenido del registro Y, nos dará la dirección de memoria donde se encuentra el dato que se carga en el Acumulador.

LDA dir **CARGAR EN EL ACUMULADOR**
(Direccionamiento página cero)



NUMERO DE CICLOS DE RELOJ:

3

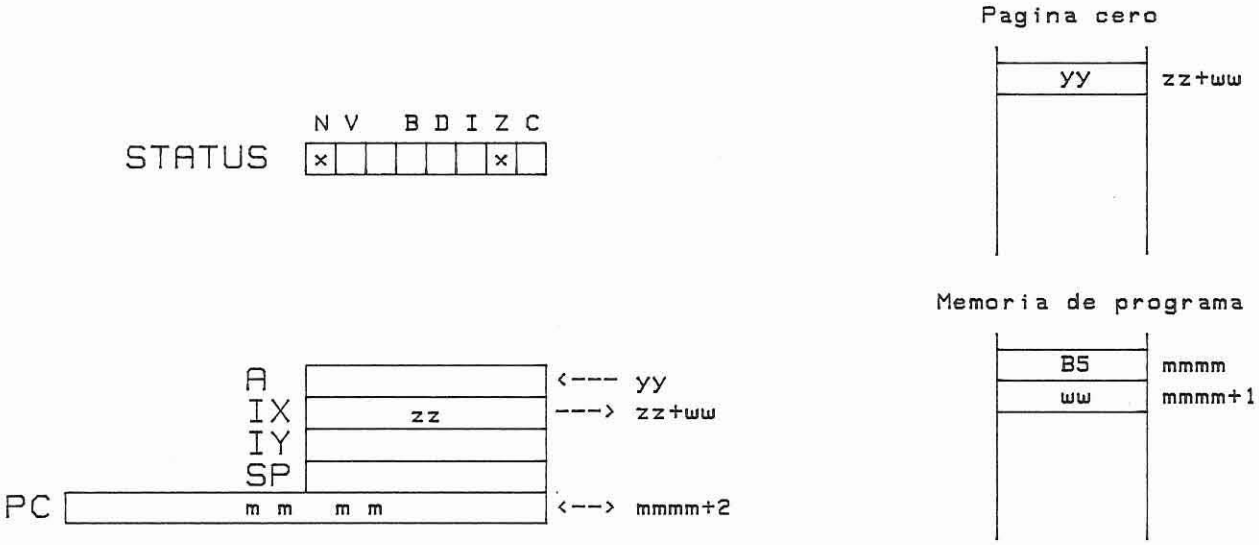
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la dirección de memoria de página cero donde se encuentra el dato que se carga en el Acumulador.

LDA dir,X — **CARGAR EN EL ACUMULADOR**
(Direccionamiento página cero indexado)



NUMERO DE CICLOS DE RELOJ:

4

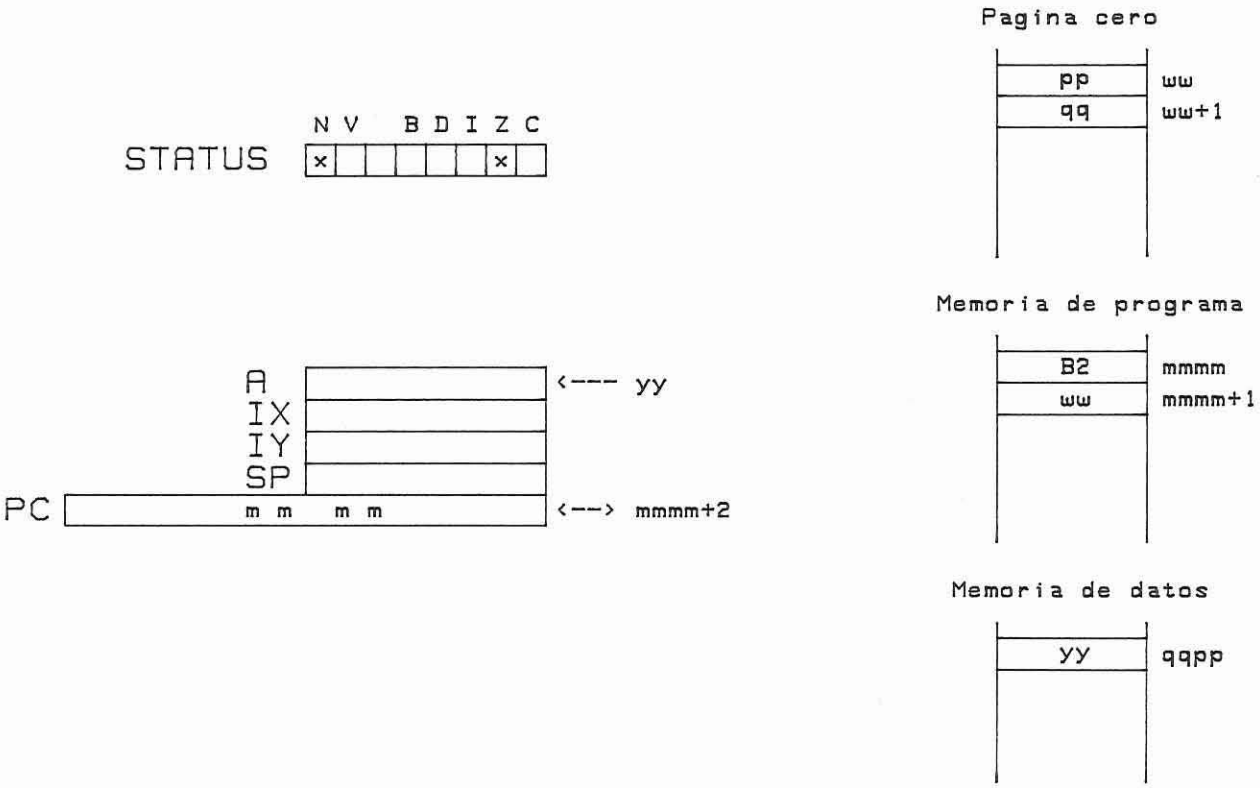
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la parte baja de la dirección de memoria que, sumada al contenido del registro X, da la dirección de memoria de página cero donde se encuentra el dato que se carga en el Acumulador.

LDA (dir) * **CARGAR EN EL ACUMULADOR**
(Direccionamiento indirecto mejorado)



NUMERO DE CICLOS DE RELOJ:

5

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

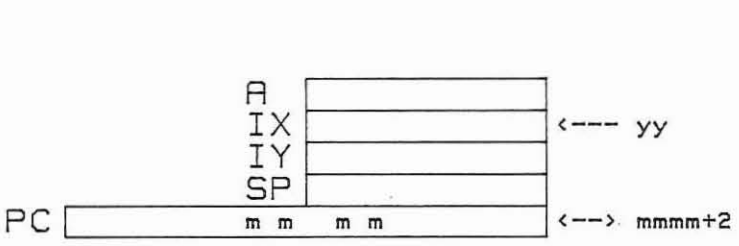
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

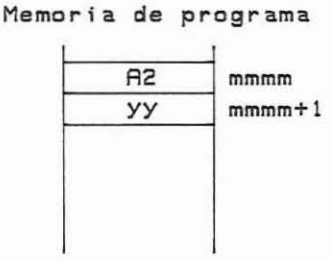
DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la parte baja de una dirección de memoria de página cero. En esta dirección y en la siguiente se encuentran, en orden inverso (primero byte de menor peso y despues byte de mayor peso), los dos bytes que forman la dirección de memoria donde se encuentra el dato que se carga en el Acumulador.

LDX #dato **CARGAR EL REGISTRO X**
(Direccionamiento inmediato)







NUMERO DE CICLOS DE RELOJ:

2

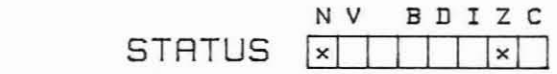
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

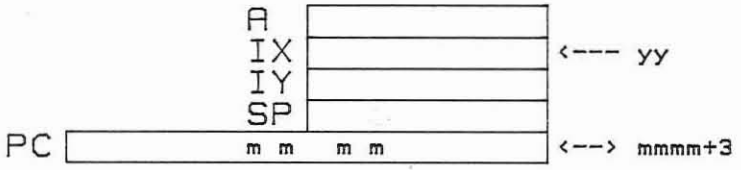
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

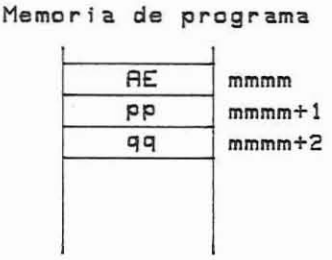
DESCRIPCION:

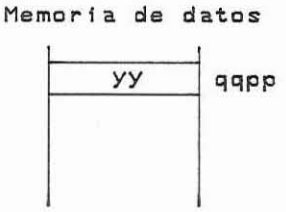
El contenido de la posición de memoria que sigue a la que contiene el código de operación se carga en el registro X.

LDX dir **CARGAR EL REGISTRO X**
(Direccionamiento absoluto)









NUMERO DE CICLOS DE RELOJ:

4

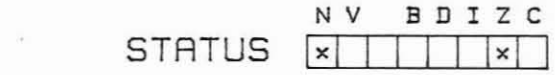
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

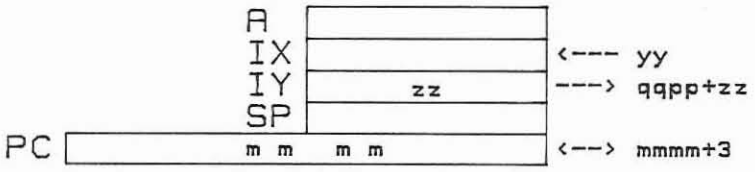
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

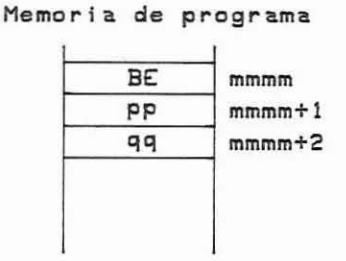
DESCRIPCION:

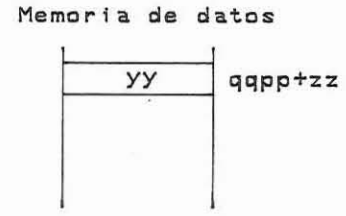
Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria donde se encuentra el dato que se carga en el Registro X.

LDX dir,Y — **CARGAR EL REGISTRO X**
(Direccionamiento absoluto indexado)









NUMERO DE CICLOS DE RELOJ:

4 (añadir 1 si salta de página)

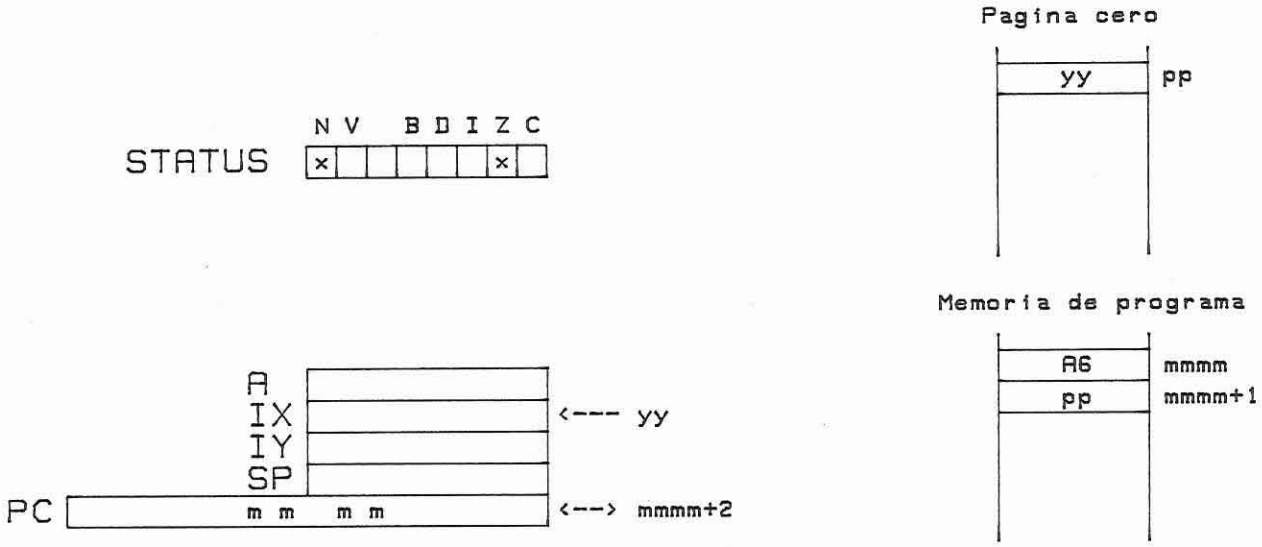
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria que, sumada al contenido del registro Y, indica la posición de memoria donde se encuentra el dato que se carga en el Registro X.

LDX dir **CARGAR EN EL REGISTRO X**
(Direccionamiento página cero)



NUMERO DE CICLOS DE RELOJ:

3

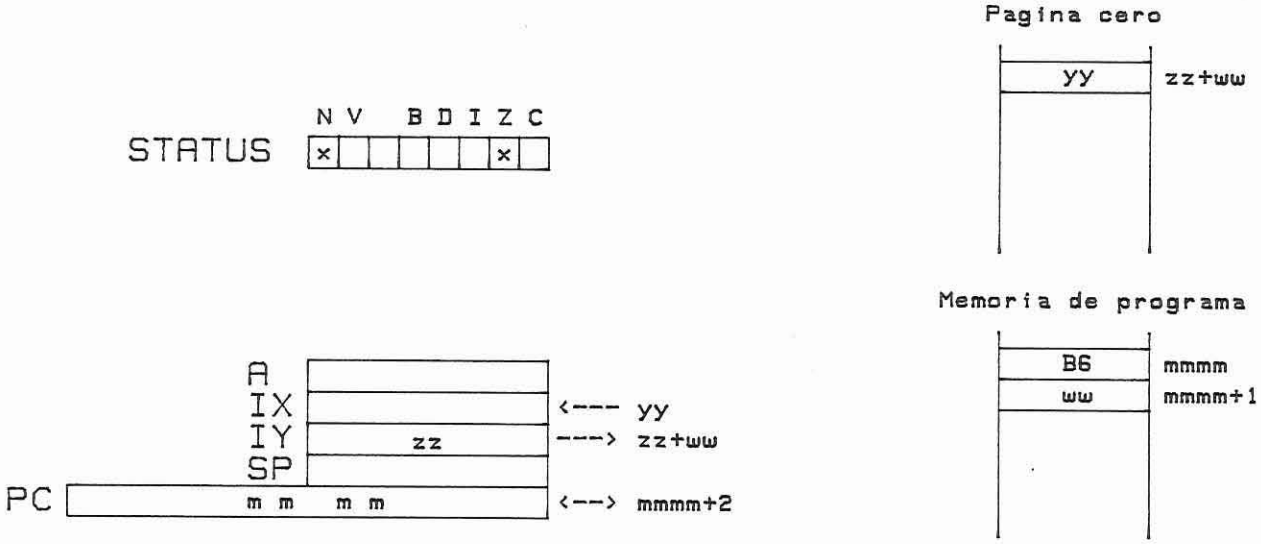
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la dirección de memoria de página cero donde se encuentra el dato que se carga en el Registro X.

LDX dir,Y — **CARGAR EN EL REGISTRO X**
(Direccionamiento página cero indexado)



NUMERO DE CICLOS DE RELOJ:

4

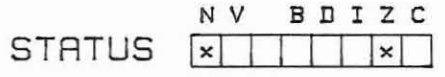
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

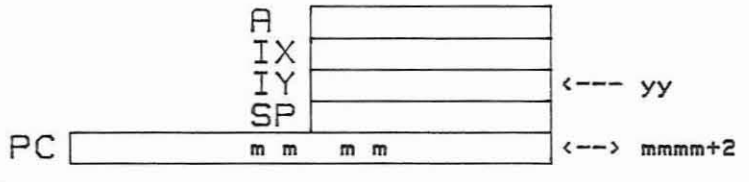
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

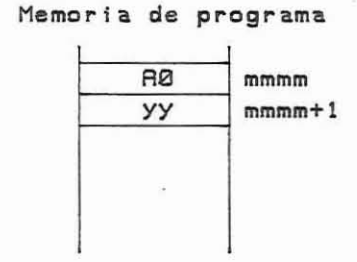
DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la parte baja de la dirección de memoria que, sumada al contenido del registro Y, da la dirección de memoria de página cero donde se encuentra el dato que se carga en el Registro X.

LDY #dato — **CARGAR EL REGISTRO Y**
(Direccionamiento inmediato)







NUMERO DE CICLOS DE RELOJ:

2

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

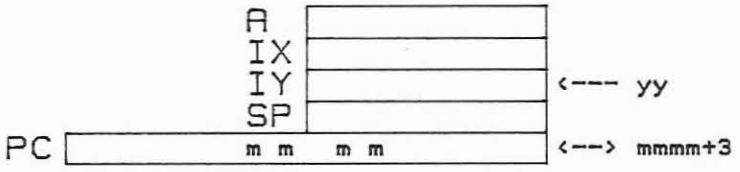
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

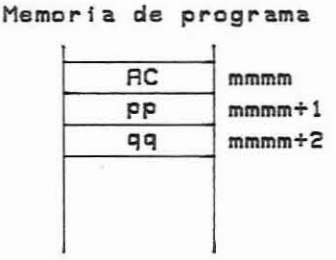
DESCRIPCION:

El contenido de la posición de memoria que sigue a la que contiene el código de operación se carga en el registro Y.

LDY dir — **CARGAR EL REGISTRO Y**
(Direccionamiento absoluto)









NUMERO DE CICLOS DE RELOJ:

4

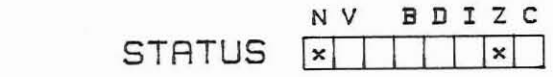
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

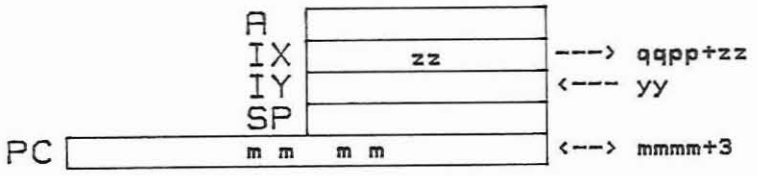
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

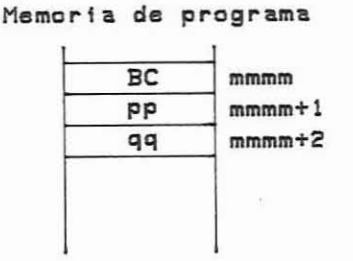
DESCRIPCION:

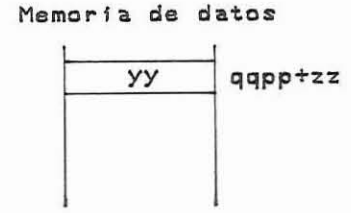
Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria donde se encuentra el dato que se carga en el Registro Y.

LDY dir,X — **CARGAR EL REGISTRO Y**
(Direccionamiento absoluto indexado)









NUMERO DE CICLOS DE RELOJ:

4 (añadir 1 si salta de página)

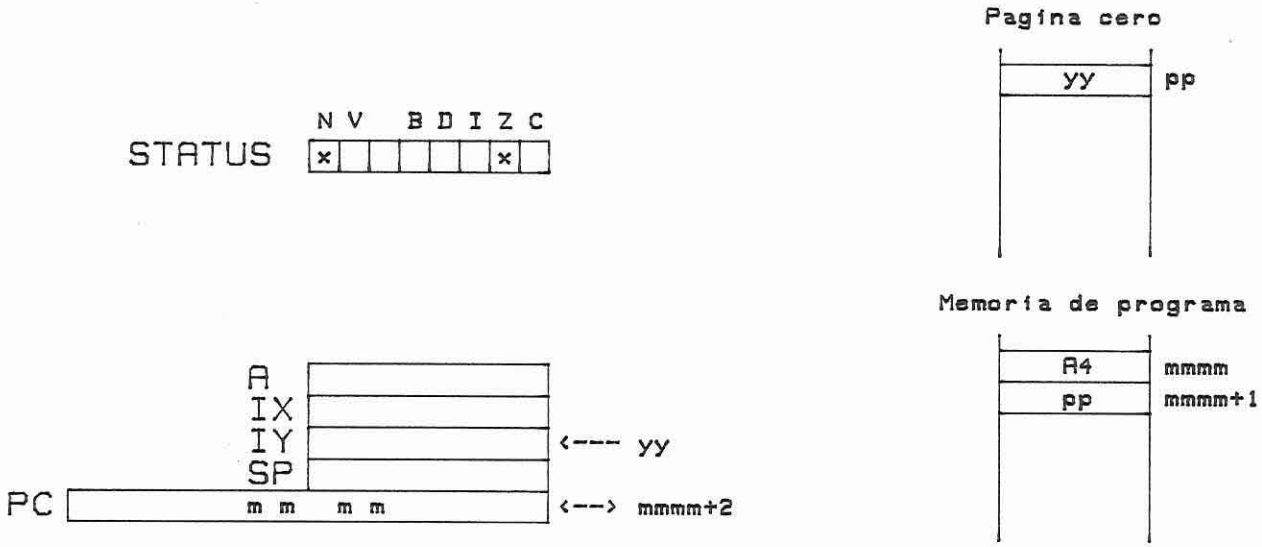
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria que, sumada al contenido del registro X, indica la posición de memoria donde se encuentra el dato que se carga en el Registro Y.

LDY dir **CARGAR EN EL REGISTRO Y**
(Direccionamiento página cero)



NUMERO DE CICLOS DE RELOJ:

3

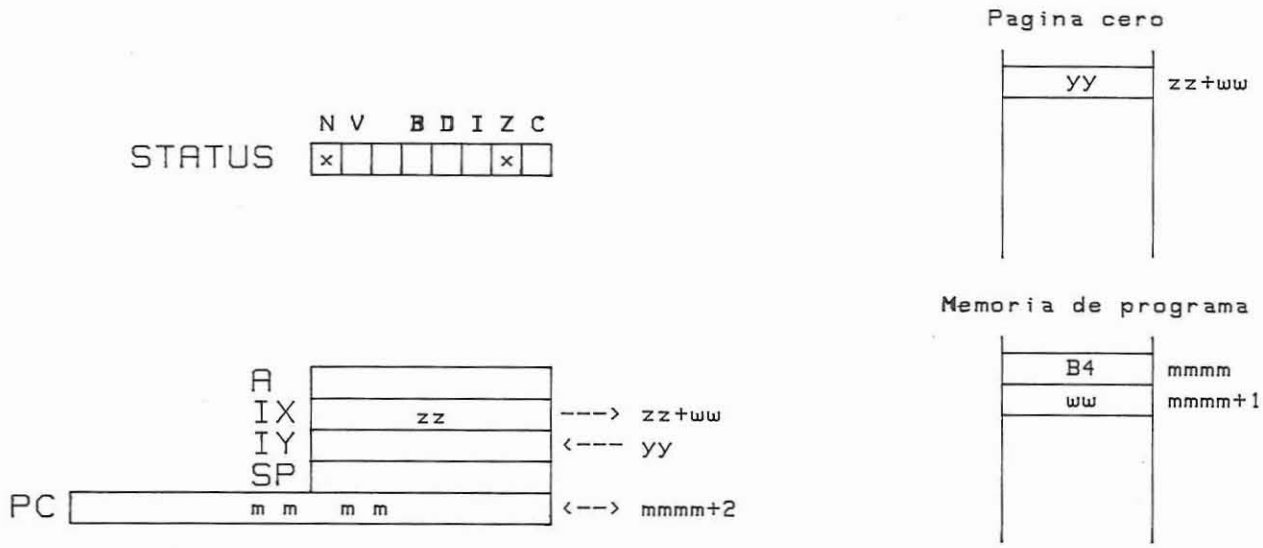
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la dirección de memoria de página cero donde se encuentra el dato que se carga en el Registro Y.

LDY dir,X **CARGAR EN EL REGISTRO Y**
(Direccionamiento página cero indexado)



NUMERO DE CICLOS DE RELOJ:

4

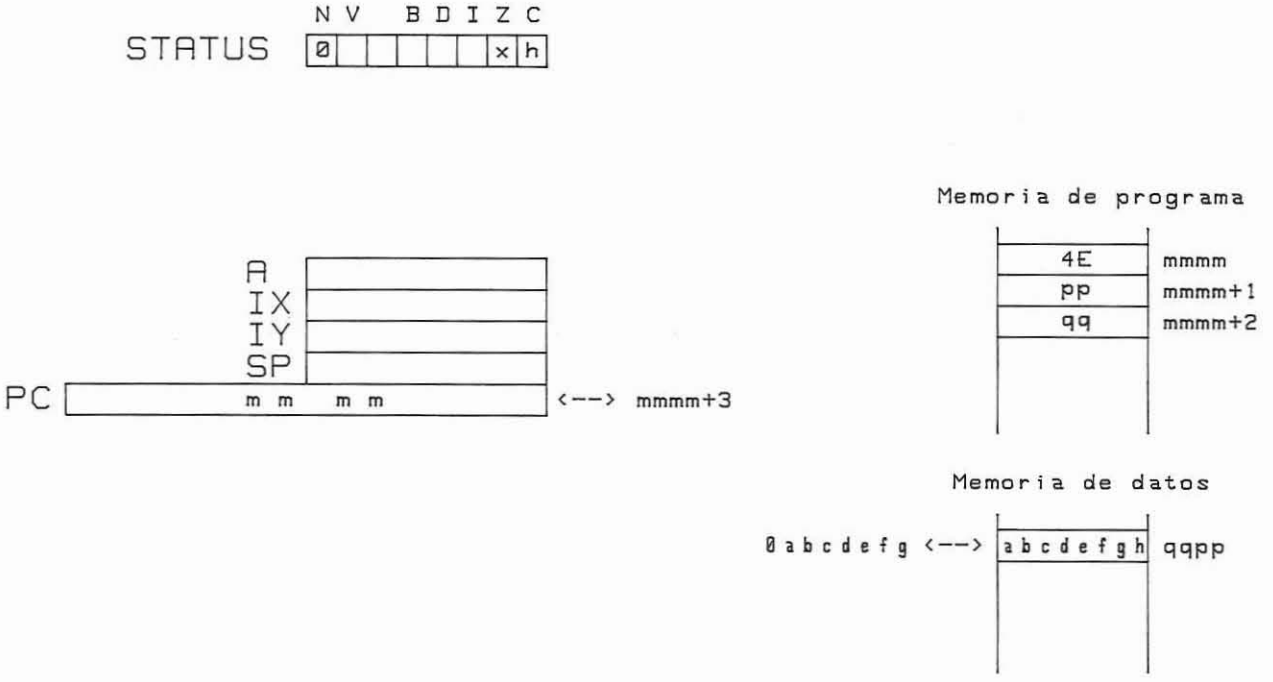
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la parte baja de la dirección de memoria que, sumada al contenido del registro X, da la dirección de memoria de página cero donde se encuentra el dato que se carga en el Registro Y.

LSR dir **DESPLAZAMIENTO DE UN BIT A LA DERECHA**
(Direccionamiento absoluto)



NUMERO DE CICLOS DE RELOJ:

6

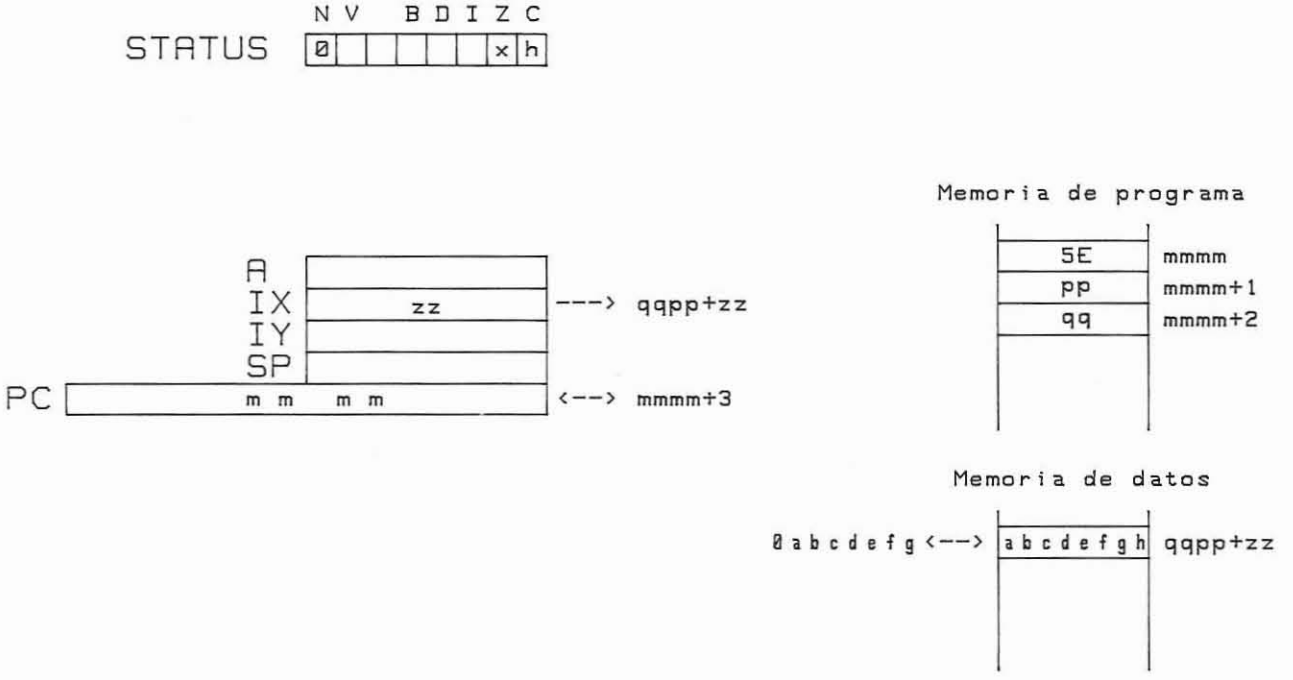
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: 0
Z: Se pone a 1 si el resultado es cero.
C: Toma el valor del bit de menor peso.

DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria donde se encuentra el dato cuyos bits son desplazados una posición hacia la derecha. El bit de menor peso pasa al Flag de Carry del Registro de Status y el bit de mayor peso se pone a 0.

LSR dir,X **DESPLAZAMIENTO DE UN BIT A LA DERECHA**
(Direccionamiento absoluto indexado)



NUMERO DE CICLOS DE RELOJ:

7

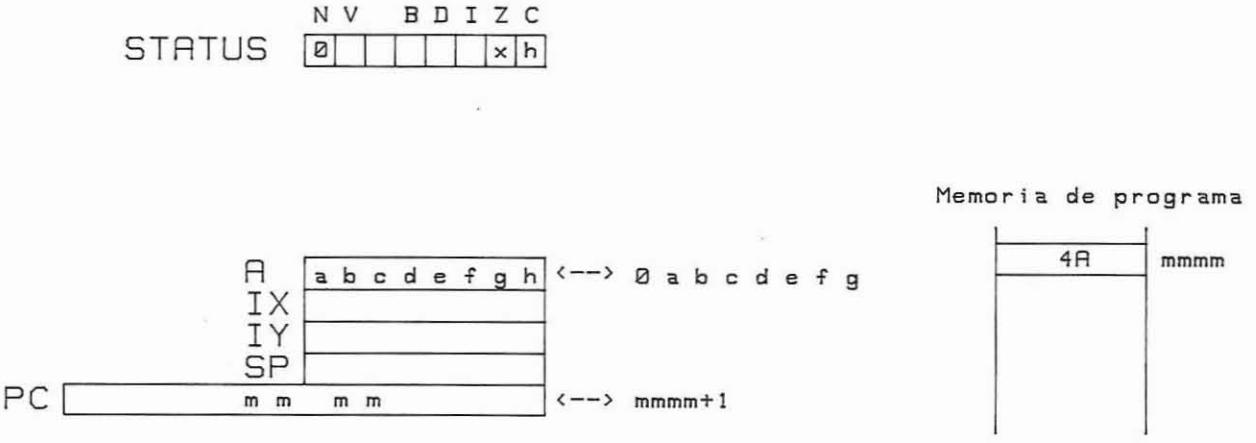
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: 0
Z: Se pone a 1 si el resultado es cero.
C: Toma el valor del bit de menor peso.

DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria que, sumada al contenido del registro X, indica la posición de memoria donde se encuentra el dato. Los bits de este byte se desplazan una posición hacia la derecha. El bit de menor peso pasa al Flag de Carry del Registro de Status y el bit de mayor peso se pone a 0.

LSR A **DESPLAZAMIENTO DE UN BIT A LA DERECHA**
(Direccionamiento de Acumulador)



NUMERO DE CICLOS DE RELOJ:

2

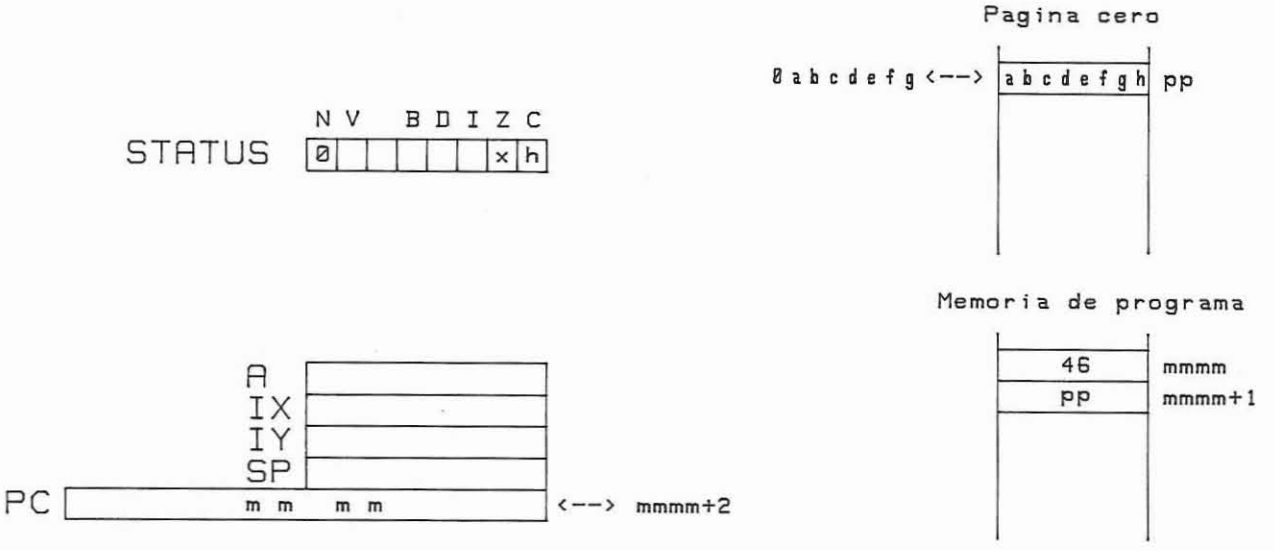
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.
C: Toma el valor del bit de menor peso.

DESCRIPCION:

El contenido del Acumulador se desplaza un bit a la derecha, de manera que el bit de menor peso pasa al bit de Carry del Registro de Flags y el bit de mayor peso se pone a cero.

LSR dir **DESPLAZAMIENTO DE UN BIT A LA DERECHA**
(Direccionamiento página cero)



NUMERO DE CICLOS DE RELOJ:

5

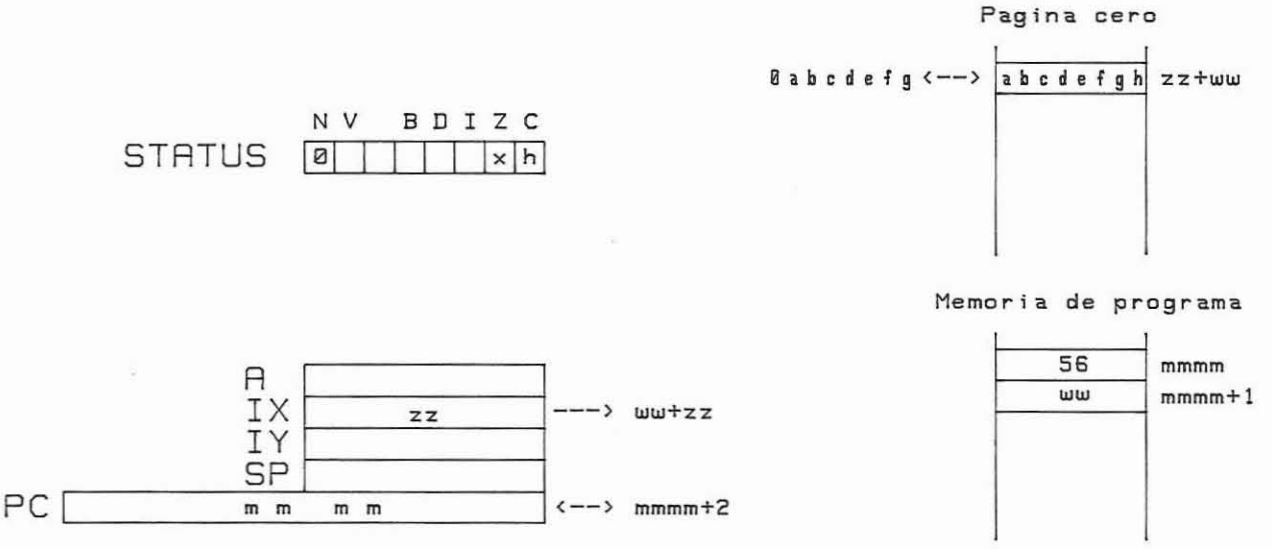
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.
C: Toma el valor del bit de menor peso.

DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la dirección de memoria de página cero donde se encuentra el dato cuyos bits son desplazados una posición hacia la derecha. El bit de menor peso pasa al Flag de Carry del Registro de Status y el bit de mayor peso se pone a 0.

LSR dir,X **DESPLAZAMIENTO DE UN BIT A LA DERECHA**
(Direccionamiento página cero indexado)



NUMERO DE CICLOS DE RELOJ:

6

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

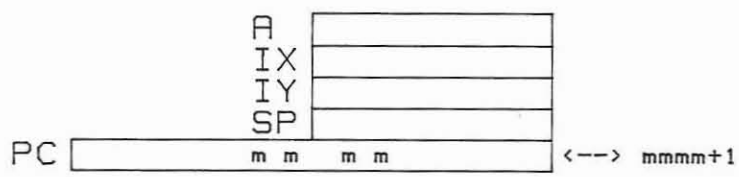
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.
C: Toma el valor del bit de menor peso.

DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la parte baja de la dirección de memoria que, sumada al contenido del registro X, da la dirección de memoria de página cero donde se encuentra el dato. Los bits de este byte se desplazan una posición hacia la derecha. El bit de menor peso pasa al Flag de Carry del Registro de Status y el bit de mayor peso se pone a 0.

NOP **NINGUNA OPERACION**
(Direccionamiento implícito)







NUMERO DE CICLOS DE RELOJ:

2

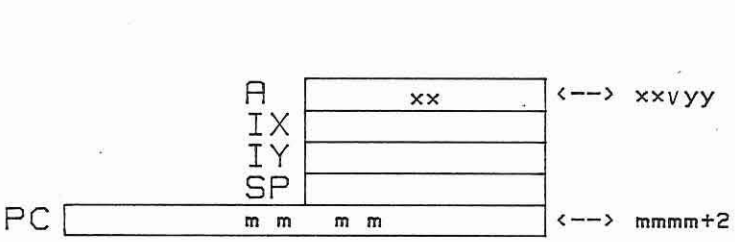
FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

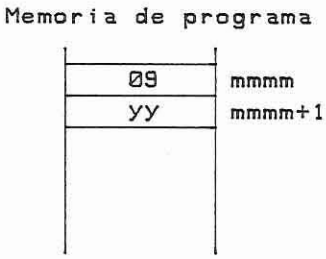
DESCRIPCION:

Cuando se ejecuta esta instrucción, la unica operación que se produce es el incremento del 'Program Counter'.

ORA #dato **FUNCION OR**
(Direccionamiento inmediato)







NUMERO DE CICLOS DE RELOJ:

2

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

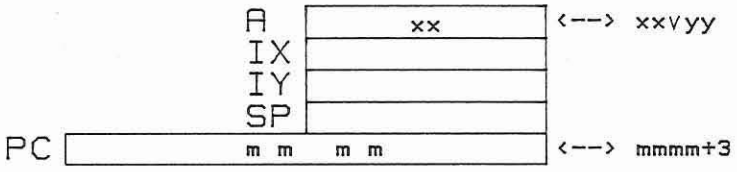
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

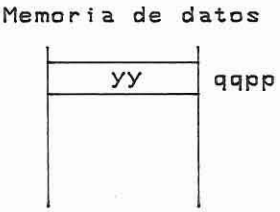
El contenido de la posición de memoria que sigue a la que contiene el código de operación efectúa bit a bit la operación OR con el Acumulador. El resultado queda en el Acumulador.

ORA dir **FUNCION OR**
(Direccionamiento absoluto)









NUMERO DE CICLOS DE RELOJ:

4

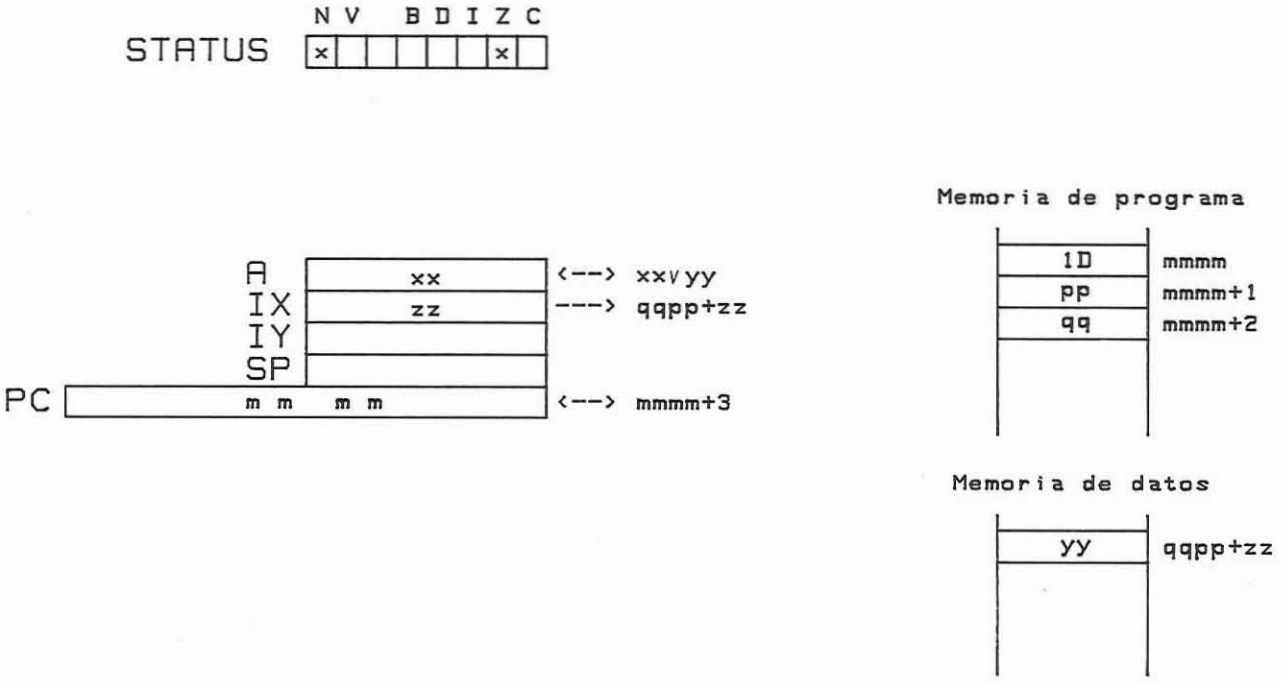
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria donde se encuentra el dato que efectúa bit a bit la operación OR con el Acumulador. El resultado queda en el Acumulador.

ORA dir,X — **FUNCION OR**
(Direccionamiento absoluto indexado)



NUMERO DE CICLOS DE RELOJ:

4 (añadir 1 si salta de página)

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

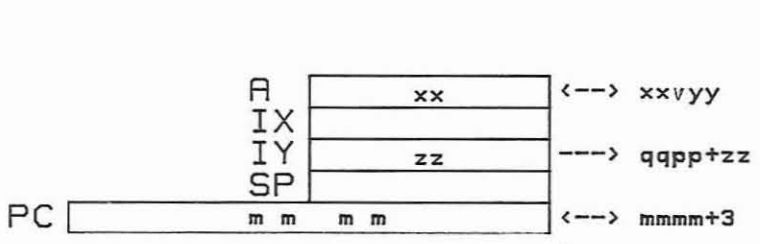
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte de menor peso y luego el byte de mayor peso), la dirección de memoria que, sumada al contenido del registro X, indica la posición de memoria donde se encuentra el dato. Este dato efectúa bit a bit la operación OR con el Acumulador. El resultado queda en el Acumulador.

ORA dir,Y — **FUNCION OR**
(Direccionamiento absoluto indexado)







NUMERO DE CICLOS DE RELOJ:

4 (añadir 1 si salta de página)

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

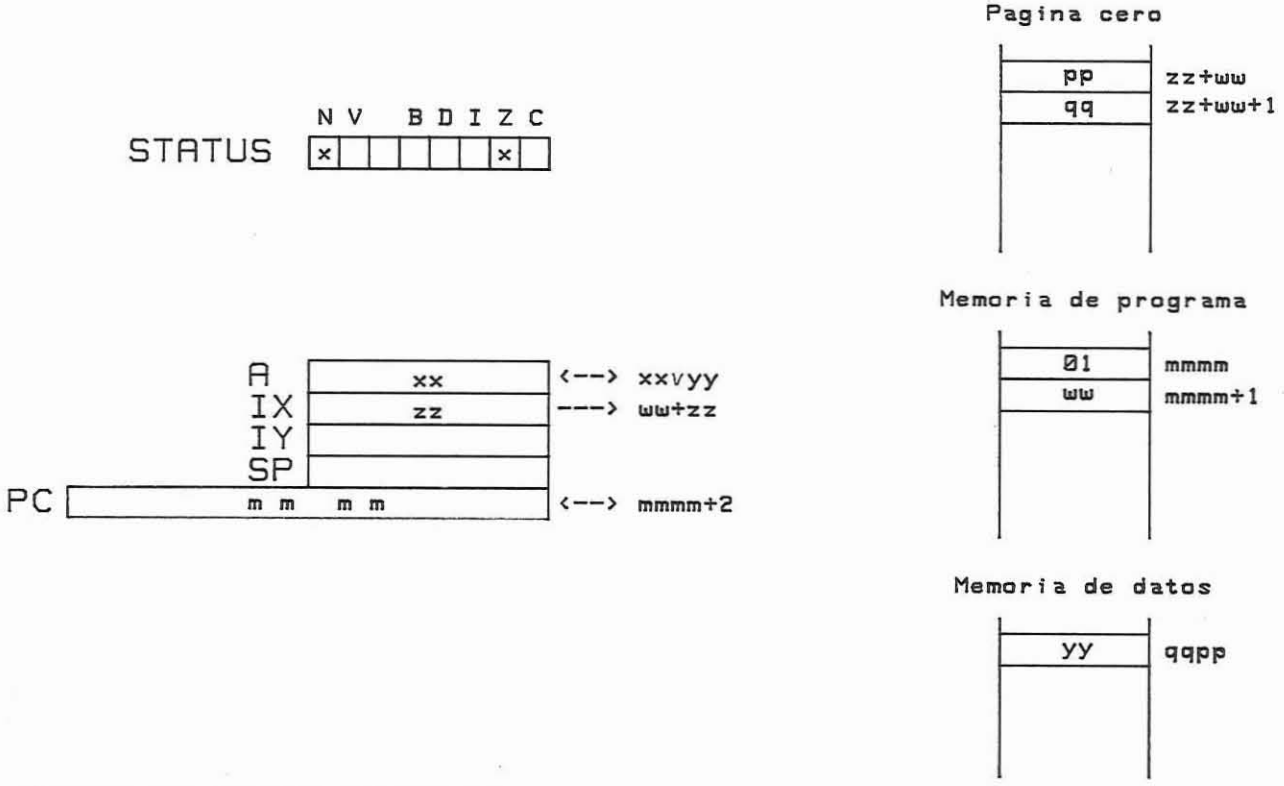
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria que, sumada al contenido del registro Y, indica la posición de memoria donde se encuentra el dato. Este dato efectúa bit a bit la operación OR con el Acumulador. El resultado queda en el Acumulador.

ORA (dir,X)

FUNCION OR
(Direccionamiento preindexado)



NUMERO DE CICLOS DE RELOJ:

6

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

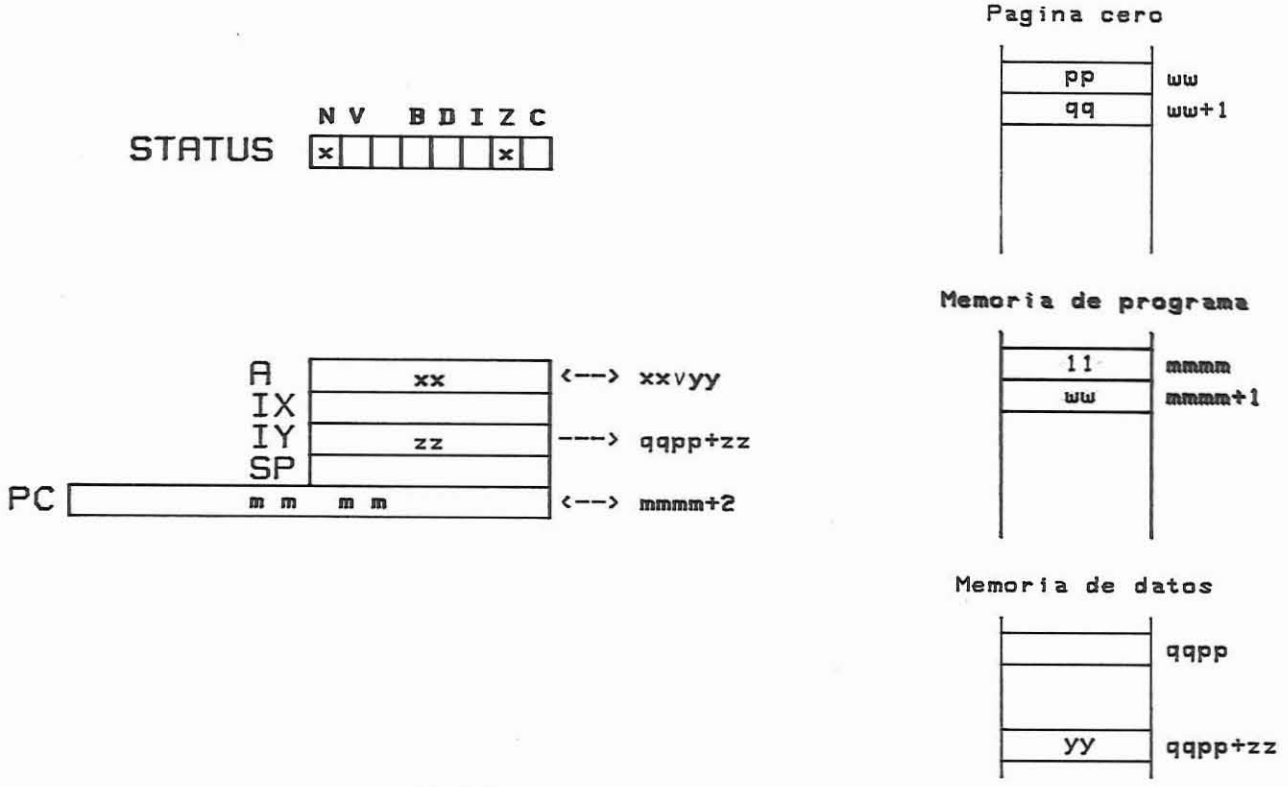
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

El contenido de la posición de memoria que sigue a la que contiene el código de operación se suma con el contenido del registro X. El resultado (despreciando el acarreo si se ha producido) corresponde a la dirección de memoria en página cero que contiene la parte baja de la dirección efectiva del dato. La parte alta de la dirección efectiva del dato está en la siguiente posición de memoria en pagina cero. De la dirección efectiva del dato se toma el byte que efectúa bit a bit la operación OR con el contenido del Acumulador. El resultado se carga de nuevo en el Acumulador.

ORA (dir),Y

FUNCION OR
(Direccionamiento postindexado)



NUMERO DE CICLOS DE RELOJ:

5 (añadir 1 si salta de página)

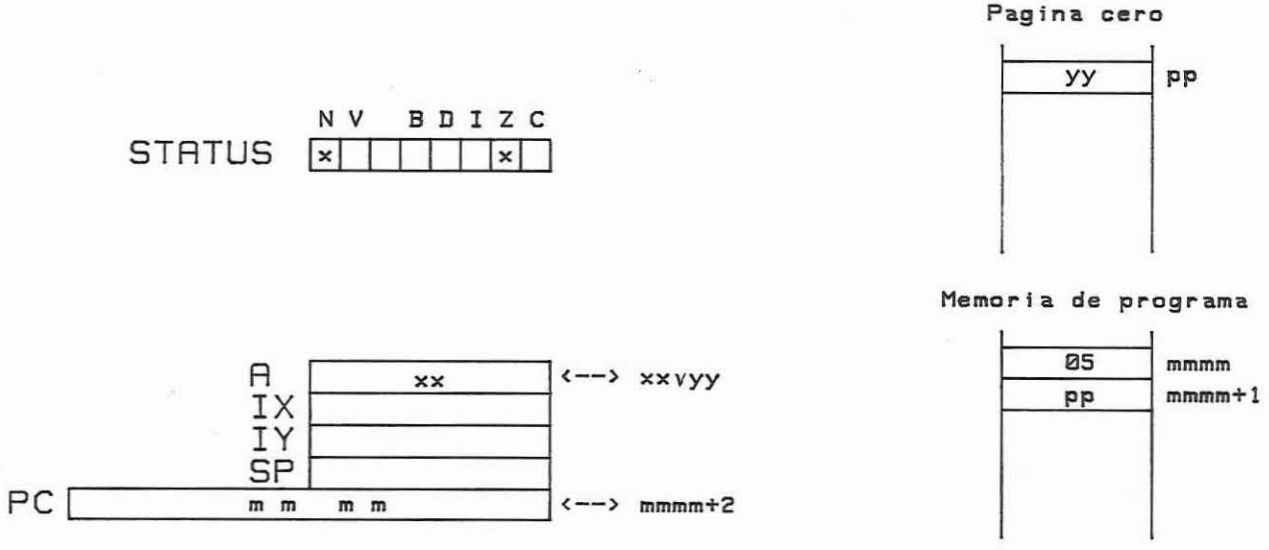
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

El contenido de la posición de memoria que sigue a la que contiene el código de operación contiene una dirección de memoria de página cero. En esta dirección y en la siguiente se encuentran, en orden inverso (primero el byte de menor peso y luego el byte mayor peso), los dos bytes que forman la dirección de memoria que sumada al contenido del registro Y, nos dará la dirección de memoria donde se encuentra el dato. Este dato efectúa la operación OR con el Acumulador. El resultado se guarda de nuevo en el Acumulador.

ORA dir — **FUNCION OR**
(Direccionamiento página cero)



NUMERO DE CICLOS DE RELOJ:

3

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

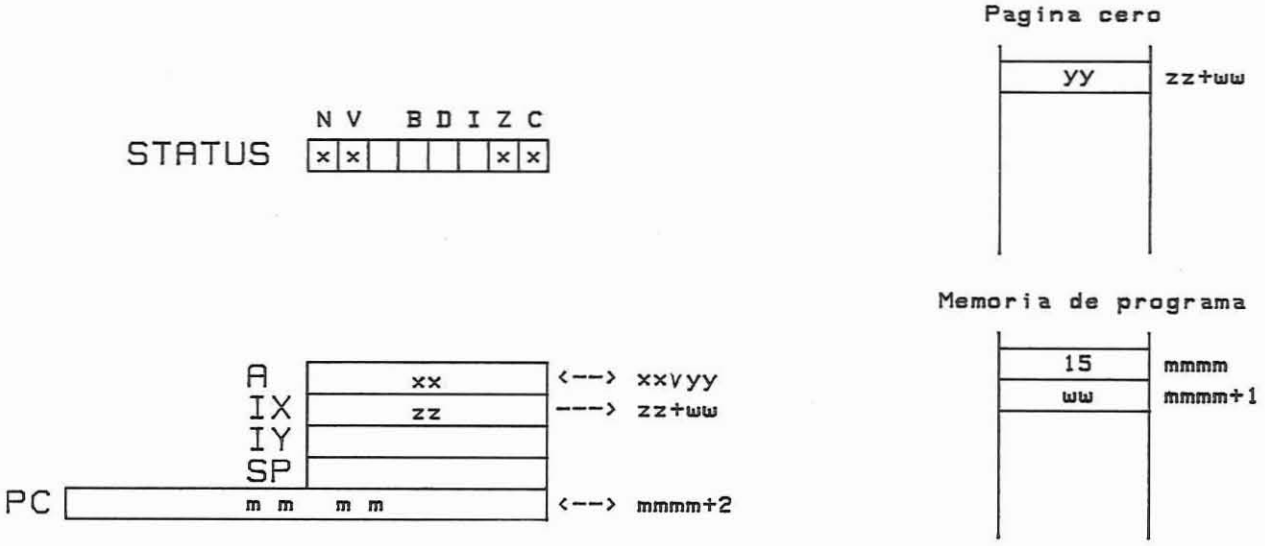
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la dirección de memoria de página cero donde se encuentra el dato que efectúa bit a bit la operación OR con el Acumulador. El resultado queda en el Acumulador.

ORA dir,X

FUNCION OR
(Direccionamiento página cero indexado)



NUMERO DE CICLOS DE RELOJ:

4

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

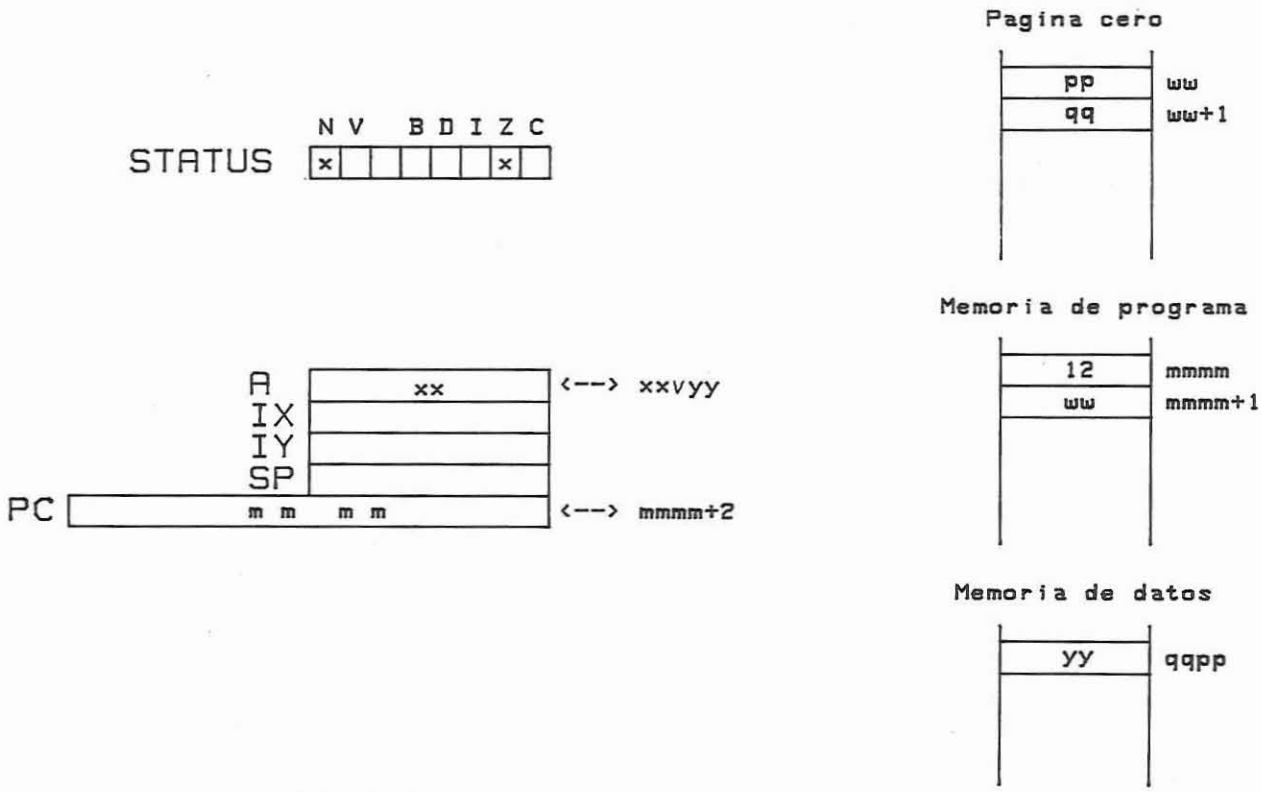
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la parte baja de la dirección de memoria que, sumada al contenido del registro X, da la dirección de memoria de página cero donde se encuentra el dato.Este dato efectúa bit a bit la operación con el Acumulador. El resultado queda en el Acumulador.

ORA (dir) *

FUNCION OR
(Direccionamiento indirecto mejorado)



NUMERO DE CICLOS DE RELOJ:

5

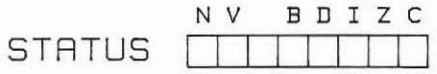
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

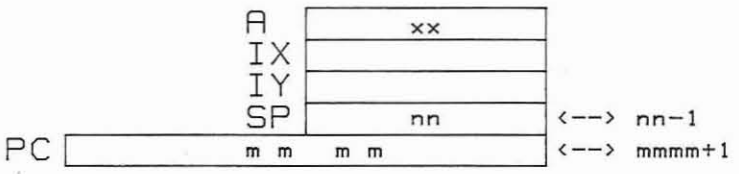
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

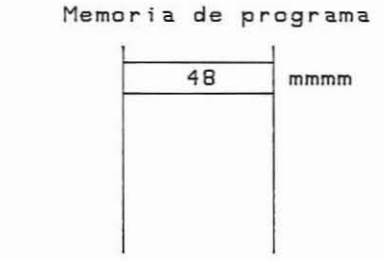
DESCRIPCION:

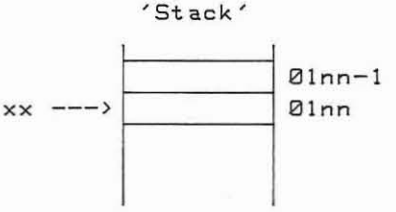
La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la parte baja de una dirección de memoria de página cero. En esta dirección y en la siguiente se encuentran, en orden inverso (primero byte de menor peso y despues byte de mayor peso), los dos bytes que forman la dirección de memoria donde se encuentra el dato que efectúa bit a bit la operación OR con el Acumulador. El resultado queda en el Acumulador.

PHA **TRANSFIERE EL CONTENIDO DEL ACUMULADOR AL STACK**
(Direccionamiento implícito)









NUMERO DE CICLOS DE RELOJ:

3

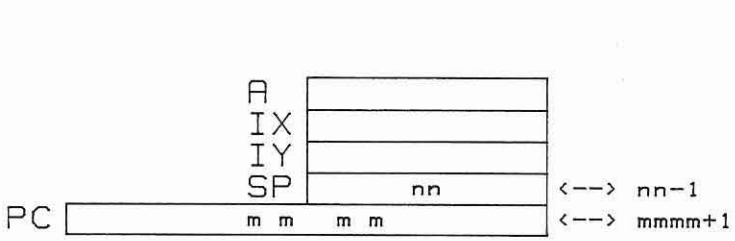
FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

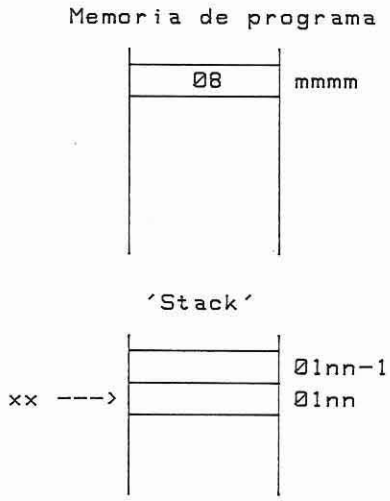
DESCRIPCION:

Esta instrucción transfiere el contenido del Acumulador al Stack y decrementa en una unidad el contenido del 'Stack Pointer'.

PHP **TRANSFIERE EL CONTENIDO DEL 'STATUS' AL 'STACK'**
(Direccionamiento implícito)







NUMERO DE CICLOS DE RELOJ:

3

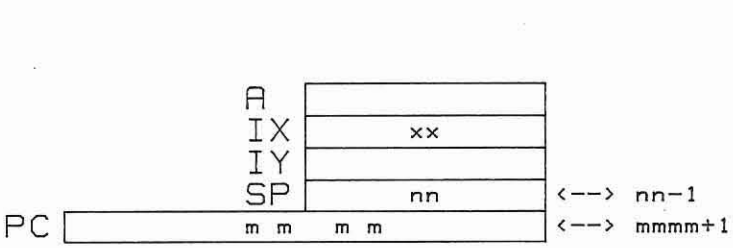
FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

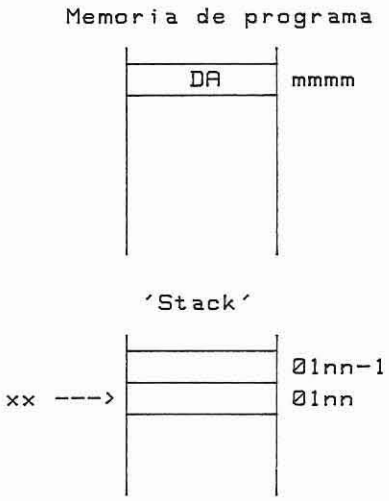
DESCRIPCION:

Esta instrucción transfiere el contenido del registro de Estado o 'Status' al Stack y decrementa en una unidad el contenido del 'Stack Pointer'.

PHX * **TRANSFIERE EL CONTENIDO DEL REGISTRO X AL STACK**
(Direccionamiento implícito)







NUMERO DE CICLOS DE RELOJ:

3

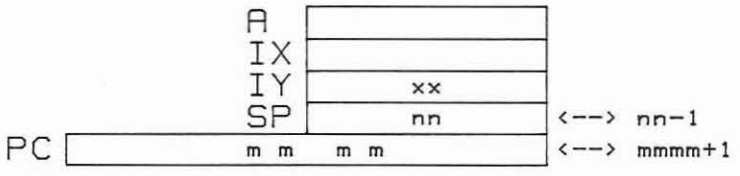
FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

DESCRIPCION:

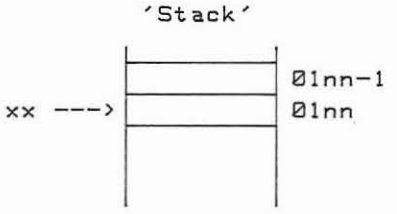
Esta instrucción transfiere el contenido del Registro X al Stack y decrementa en una unidad el contenido del 'Stack Pointer'.

PHY * **TRANSFIERE EL CONTENIDO DEL REGISTRO Y AL STACK**
(Direccionamiento implícito)









NUMERO DE CICLOS DE RELOJ:

3

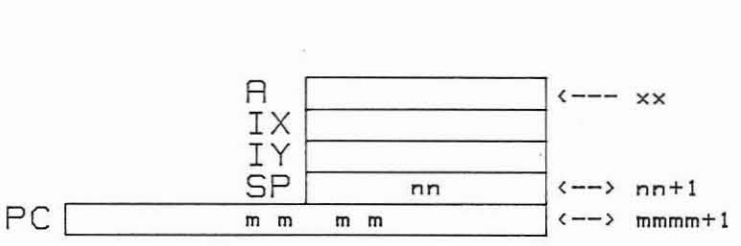
FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

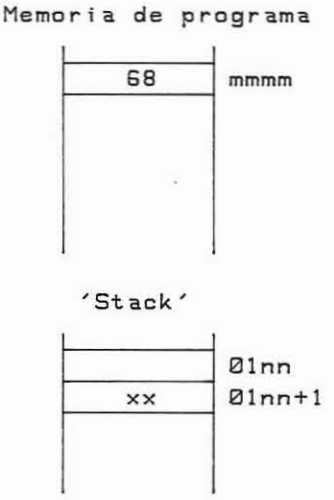
DESCRIPCION:

Esta instrucción transfiere el contenido del Registro Y al Stack y decrementa en una unidad el contenido del 'Stack Pointer'.

PLA **TRANSFIERE AL ACUMULADOR EL ULTIMO BYTE DEL STACK**
(Direccionamiento implícito)







NUMERO DE CICLOS DE RELOJ:

4

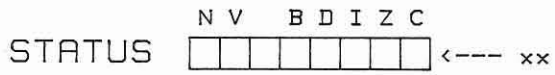
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

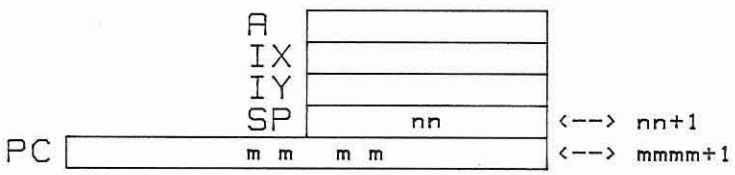
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

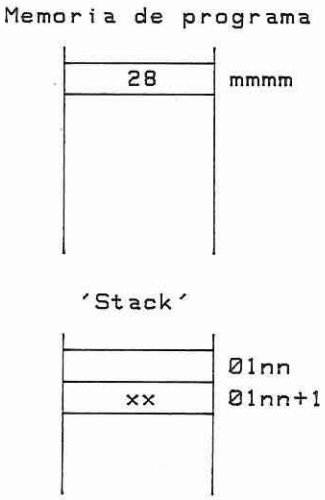
DESCRIPCION:

Esta instrucción transfiere el contenido del último byte almacenado en el Stack al Acumulador e incrementa en una unidad el contenido del 'Stack Pointer .

PLP **TRANSFIERE AL 'STATUS' EL ULTIMO BYTE DEL STACK**
(Direccionamiento implícito)







NUMERO DE CICLOS DE RELOJ:

4

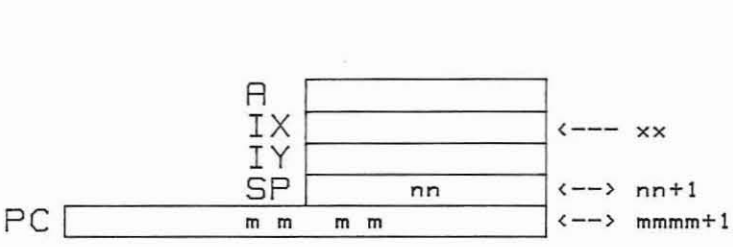
FLAGS DEL REGISTRO 'STATUS' AFECTADOS: todos

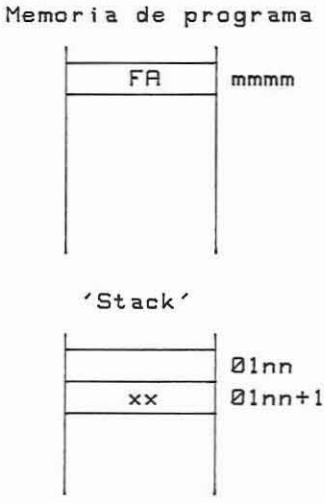
DESCRIPCION:

Esta instrucción transfiere el contenido del último byte almacenado en el Stack al Acumulador e incrementa en una unidad el contenido del 'Stack Pointer'.

PLX * **TRANSFIERE AL REGISTRO X EL ULTIMO BYTE DEL STACK**
(Direccionamiento implícito)







NUMERO DE CICLOS DE RELOJ:

4

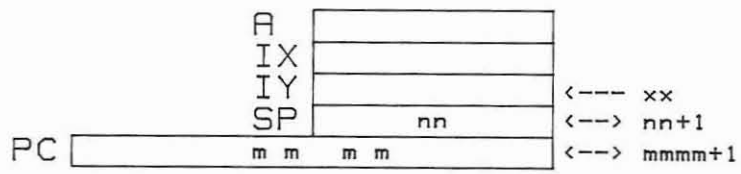
FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

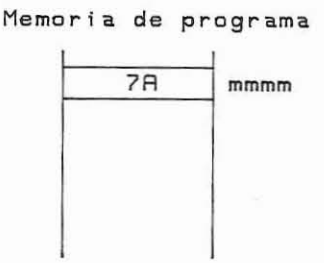
DESCRIPCION:

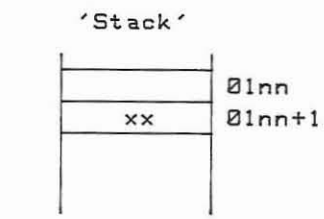
Esta instrucción transfiere el contenido del último byte almacenado en el Stack al Registro X e incrementa en una unidad el contenido del 'Stack Pointer'.

PLY * **TRANSFIERE AL REGISTRO Y EL ULTIMO BYTE DEL STACK**
(Direccionamiento implícito)









NUMERO DE CICLOS DE RELOJ:

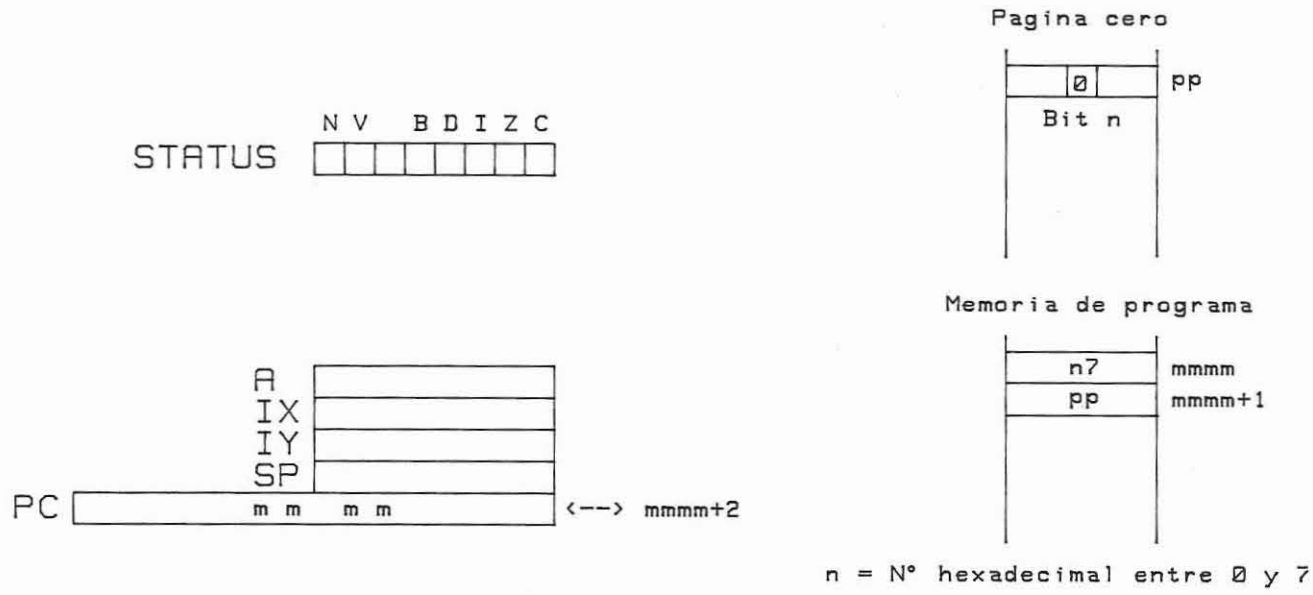
4

FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

DESCRIPCION:

Esta instrucción transfiere el contenido del último byte almacenado en el Stack al Registro Y e incrementa en una unidad el contenido del 'Stack Pointer'.

RMB N dir + **PONE A 0 EL BIT N DE LA POSICION DE MEMORIA INDICADA**
(Direccionamiento implícito)



NUMERO DE CICLOS DE RELOJ:

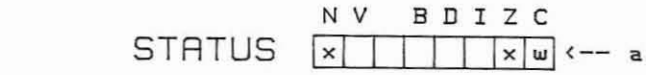
5

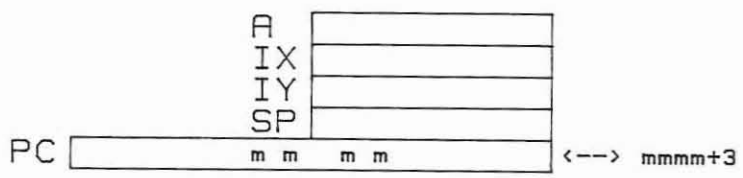
FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

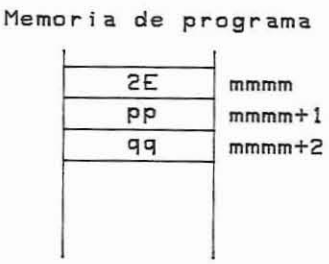
DESCRIPCION:

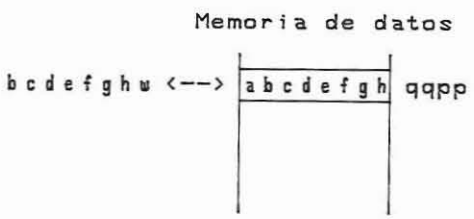
Las 4 líneas de mayor peso del byte que indica el código de operación de esta instrucción indican el bit que se pondrá a cero. La dirección de memoria donde se encuentra el dato cuyo bit N se pondrá a 0 biene indicada por el segundo byte de la instrucción.

ROL dir **ROTACION A LA IZQUIERDA**
(Direccionamiento absoluto)









NUMERO DE CICLOS DE RELOJ:

6

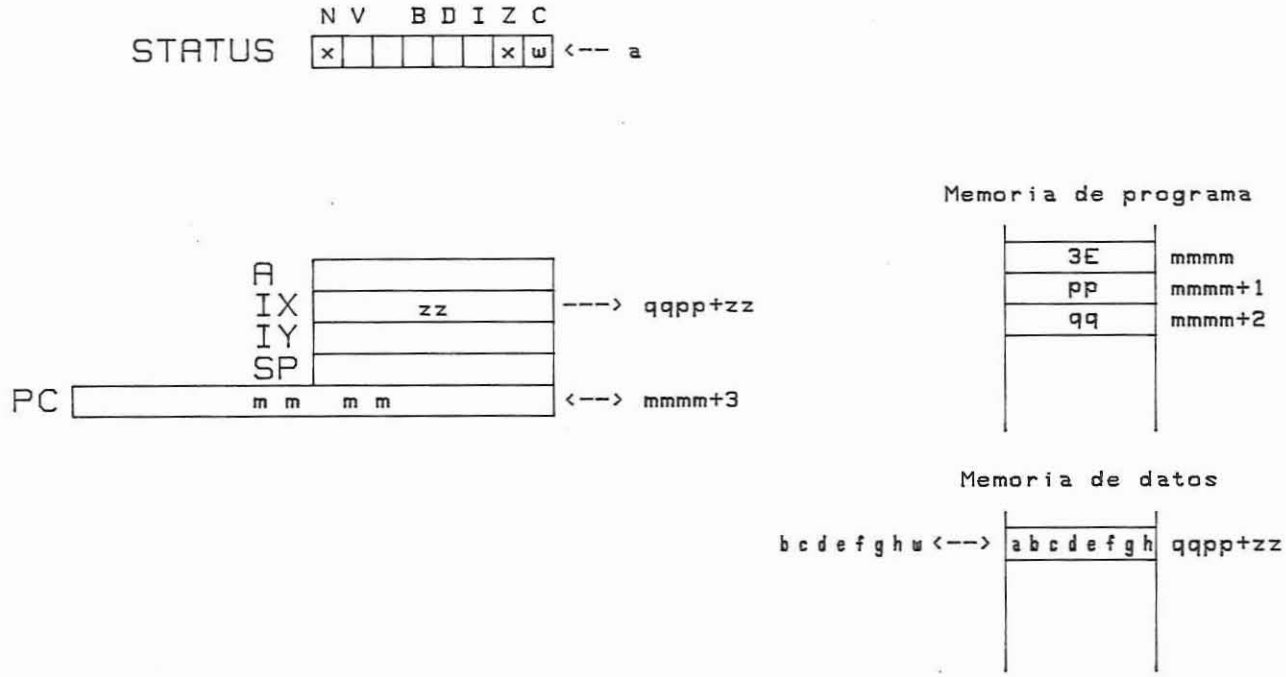
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.
C: Toma el valor del bit de mayor peso.

DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria donde se encuentra el dato cuyos bits son desplazados una posición hacia la izquierda. El contenido del Flag de Carry del Registro de Status pasa al bit de menor peso del dato y bit de mayor peso pasa al Flag de Carry del Registro de Status.

ROL dir,X — **ROTACION A LA IZQUIERDA**
(Direccionamiento absoluto indexado)



NUMERO DE CICLOS DE RELOJ:

7

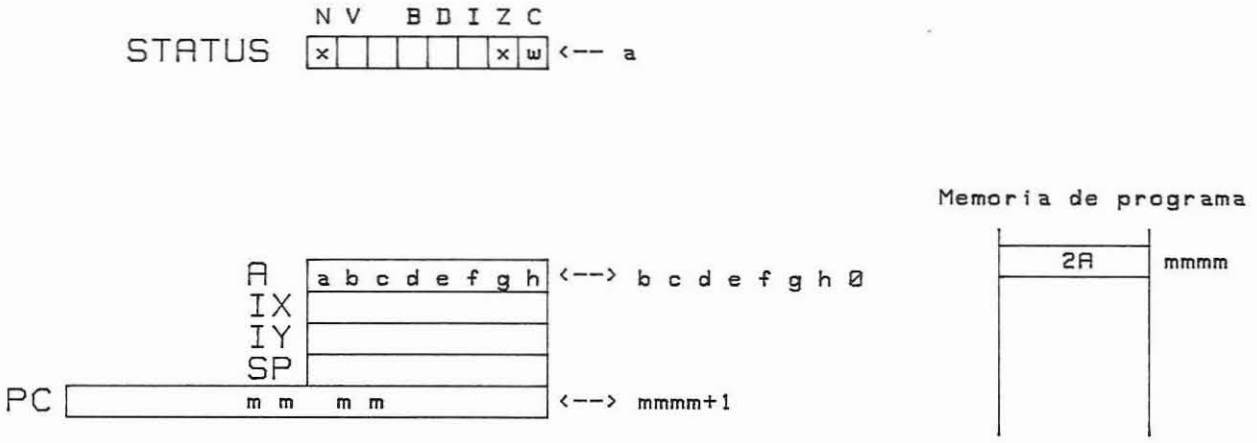
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.
C: Toma el valor del bit de mayor peso.

DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria que, sumada al contenido del registro X, indica la posición de memoria donde se encuentra el dato. Los bits de este byte se desplazan una posición hacia la izquierda. El contenido del Flag de Carry del Registro de Status pasa al bit de menor peso del dato y el bit de mayor peso pasa al Flag de Carry del Registro de Status.

ROL A — **ROTACION A LA IZQUIERDA**
(Direccionamiento de Acumulador)



NUMERO DE CICLOS DE RELOJ:

2

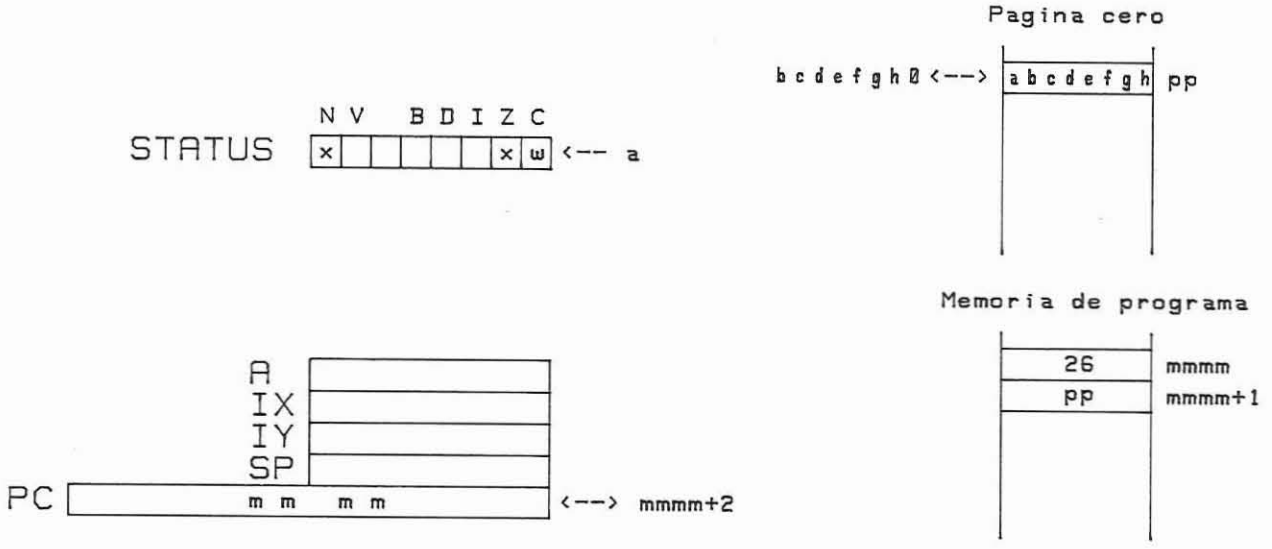
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.
C: Toma el valor del bit de mayor peso.

DESCRIPCION:

El contenido del Acumulador se desplaza un bit a la izquierda. El contenido del Flag de Carry del Registro de Status pasa al bit de menor peso del Acumulador y el bit de mayor peso pasa al Flag de Carry del Registro de Status.

ROL dir **ROTACION A LA IZQUIERDA**
(Direccionamiento página cero)



NUMERO DE CICLOS DE RELOJ:

5

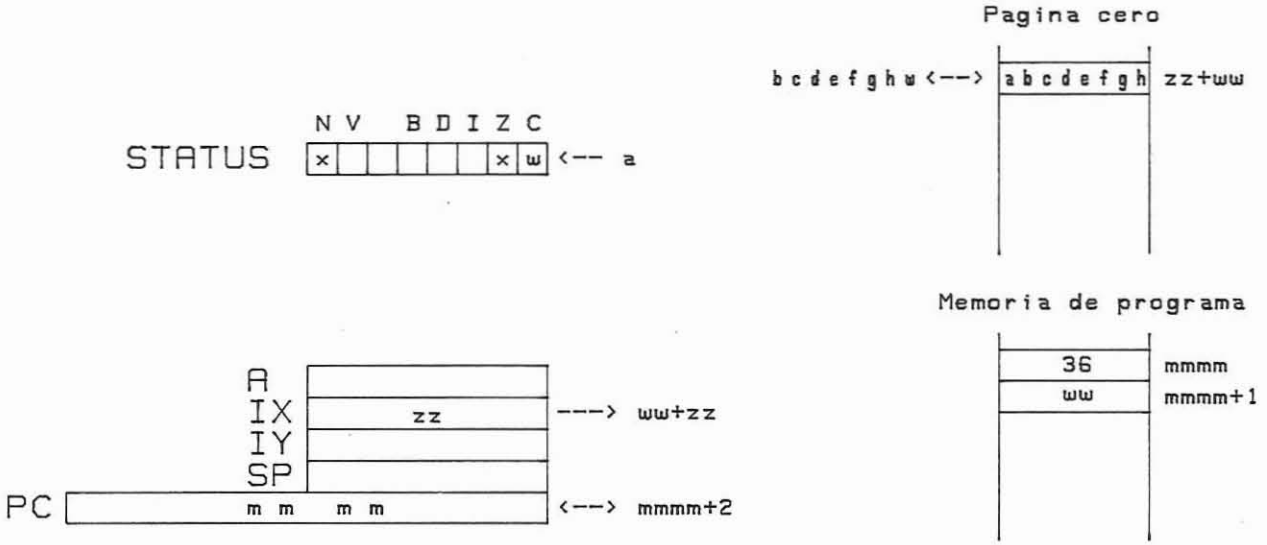
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.
C: Toma el valor del bit de mayor peso.

DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la dirección de memoria de página cero donde se encuentra el dato cuyos bits son desplazados una posición hacia la izquierda. El contenido del Flag de Carry del Registro de Status pasa al bit de menor peso del dato y el bit de mayor peso pasa al Flag de Carry del Registro de Status.

ROL dir,X — **ROTACION A LA IZQUIERDA**
(Direccionamiento página cero indexado)



NUMERO DE CICLOS DE RELOJ:

6

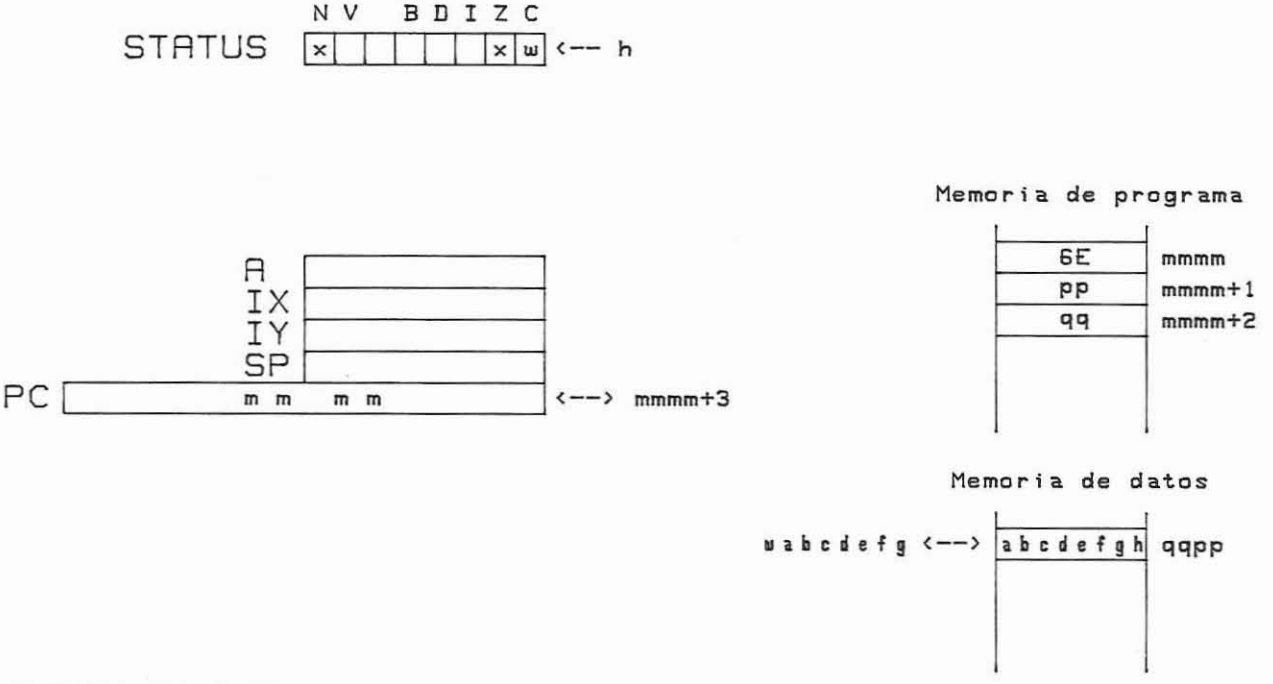
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.
C: Toma el valor del bit de mayor peso.

DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la parte baja de la dirección de memoria que, sumada al contenido del registro X, da la dirección de memoria de página cero donde se encuentra el dato. Los bits de este byte se desplazan una posición hacia la izquierda. El contenido del Flag de Carry del Registro de Status pasa al bit de menor peso del dato y el bit de mayor peso pasa al Flag de Carry del Registro de Status.

ROR dir **ROTACION A LA DERECHA**
(Direccionamiento absoluto)



NUMERO DE CICLOS DE RELOJ:

6

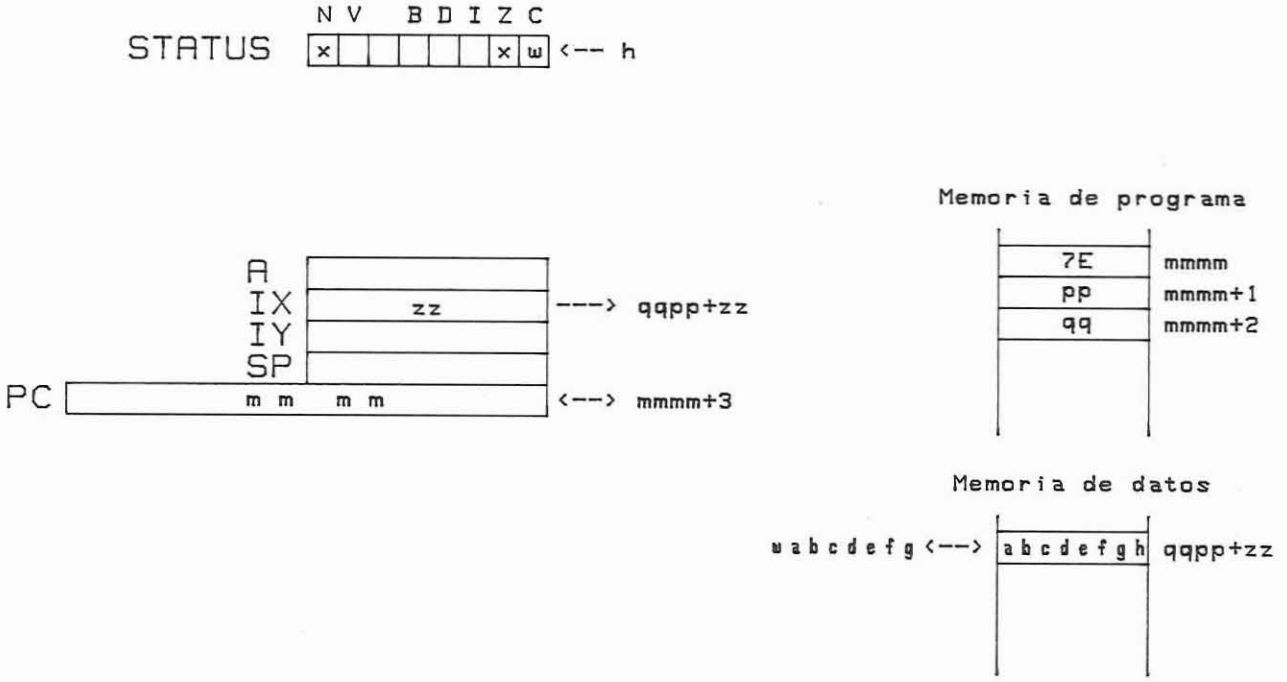
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.
C: Toma el valor del bit de menor peso.

DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria donde se encuentra el dato cuyos bits son desplazados una posición hacia la izquierda. El contenido del Flag de Carry del Registro de Status pasa al bit de mayor peso del dato y bit de menor peso pasa al Flag de Carry del Registro de Status.

ROR dir,X — **ROTACION A LA DERECHA**
(Direccionamiento absoluto indexado)



NUMERO DE CICLOS DE RELOJ:

7

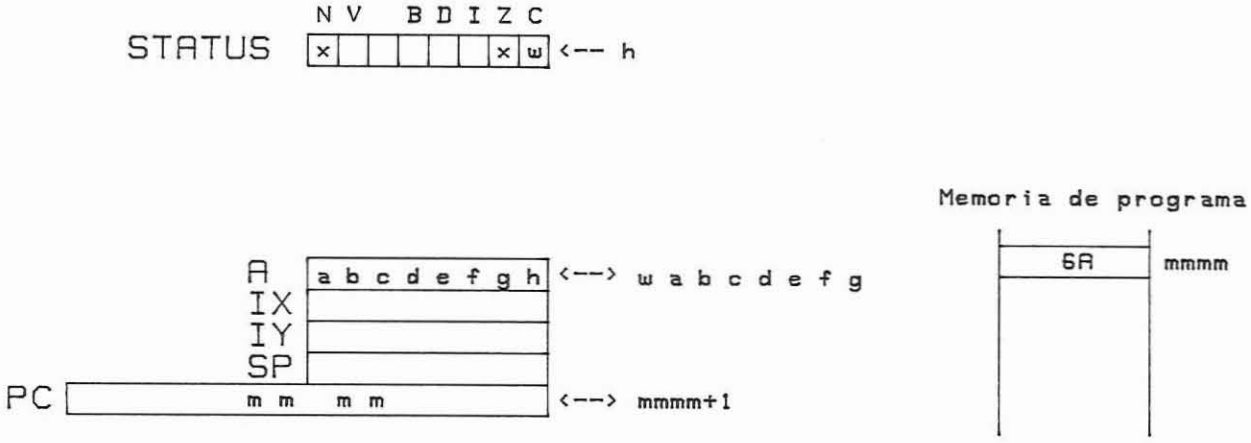
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.
C: Toma el valor del bit de menor peso.

DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria que, sumada al contenido del registro X, indica la posición de memoria donde se encuentra el dato. Los bits de este byte se desplazan una posición hacia la izquierda. El contenido del Flag de Carry del Registro de Status pasa al bit de mayor peso del dato y el bit de menor peso pasa al Flag de Carry del Registro de Status.

ROR A — **ROTACION A LA DERECHA**
(Direccionamiento de Acumulador)



NUMERO DE CICLOS DE RELOJ:

2

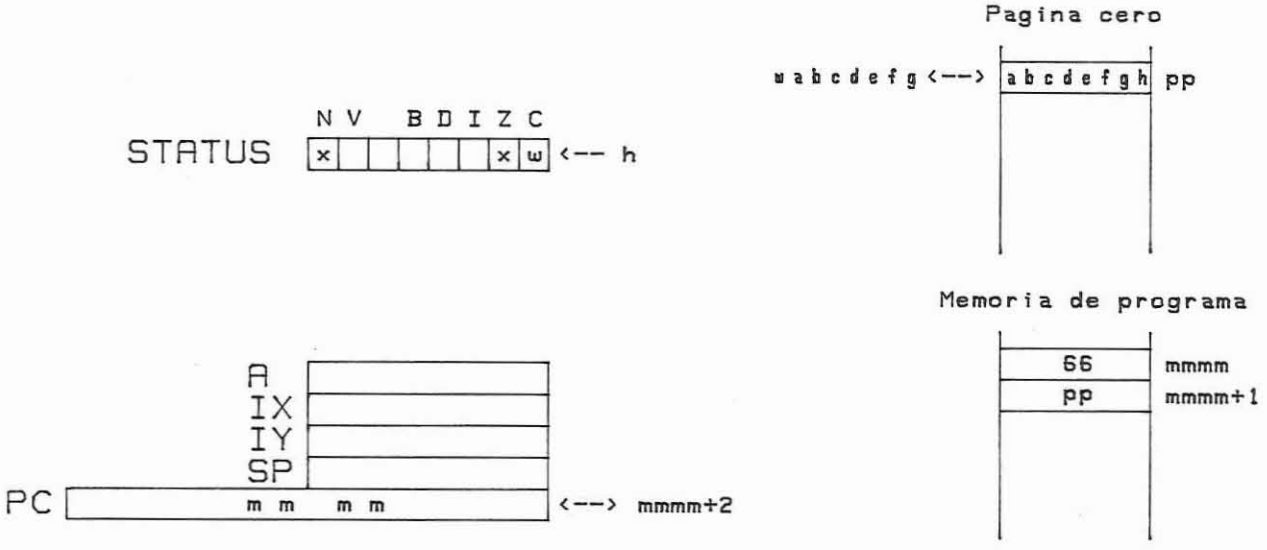
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.
C: Toma el valor del bit de menor peso.

DESCRIPCION:

El contenido del Acumulador se desplaza un bit a la izquierda. El contenido del Flag de Carry del Registro de Status pasa al bit de mayor peso del Acumulador y el bit de menor peso pasa al Flag de Carry del Registro de Status.

ROR dir **ROTACION A LA DERECHA**
(Direccionamiento página cero)



NUMERO DE CICLOS DE RELOJ:

5

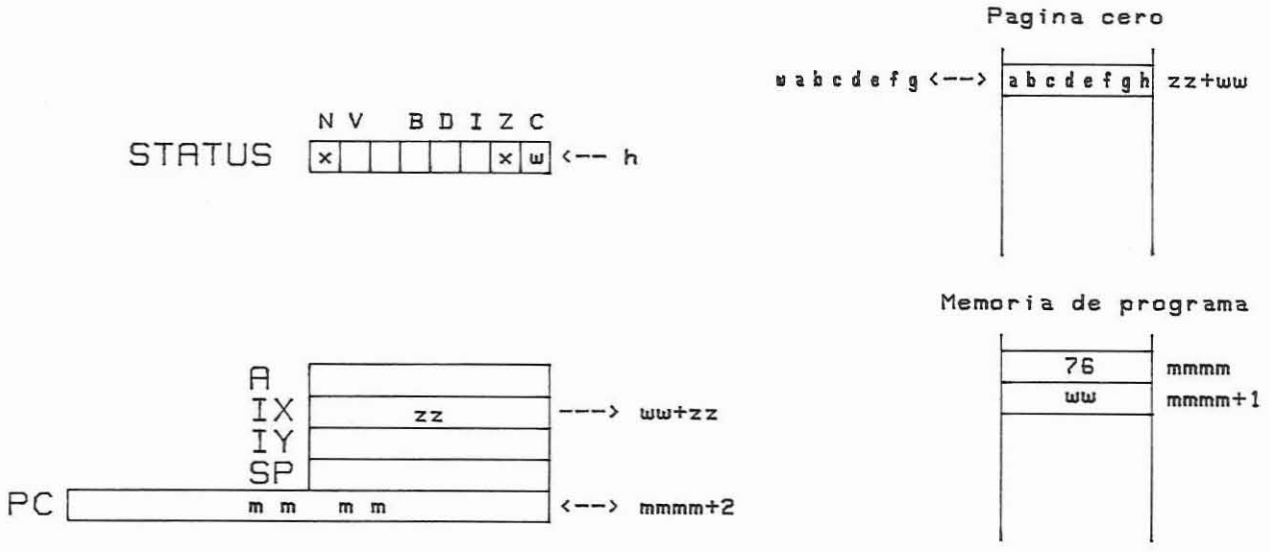
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.
C: Toma el valor del bit de menor peso.

DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la dirección de memoria de página cero donde se encuentra el dato cuyos bits son desplazados una posición hacia la izquierda. El contenido del Flag de Carry del Registro de Status pasa al bit de mayor peso del dato y el bit de menor peso pasa al Flag de Carry del Registro de Status.

ROR dir,X — **ROTACION A LA DERECHA**
(Direccionamiento página cero indexado)



NUMERO DE CICLOS DE RELOJ:

6

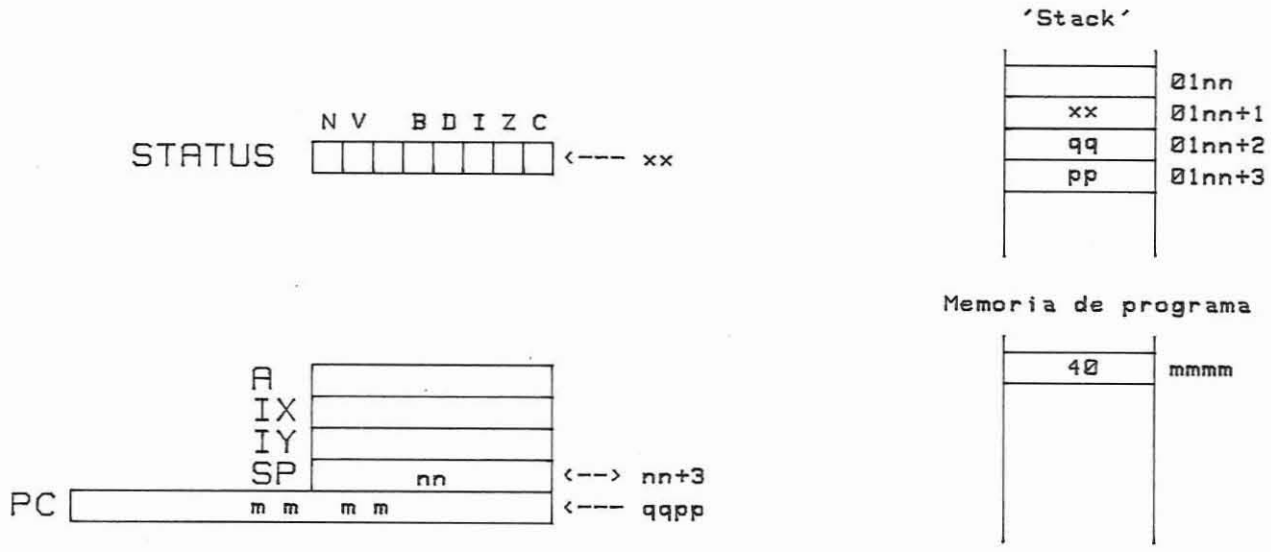
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.
C: Toma el valor del bit de menor peso.

DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la parte baja de la dirección de memoria que, sumada al contenido del registro X, da la dirección de memoria de página cero donde se encuentra el dato. Los bits de este byte se desplazan una posición hacia la izquierda. El contenido del Flag de Carry del Registro de Status pasa al bit de mayor peso del dato y el bit de menor peso pasa al Flag de Carry del Registro de Status.

RTI **RETORNO DE RUTINA DE INTERRUPCION**
(Direccionamiento implícito)



NUMERO DE CICLOS DE RELOJ:

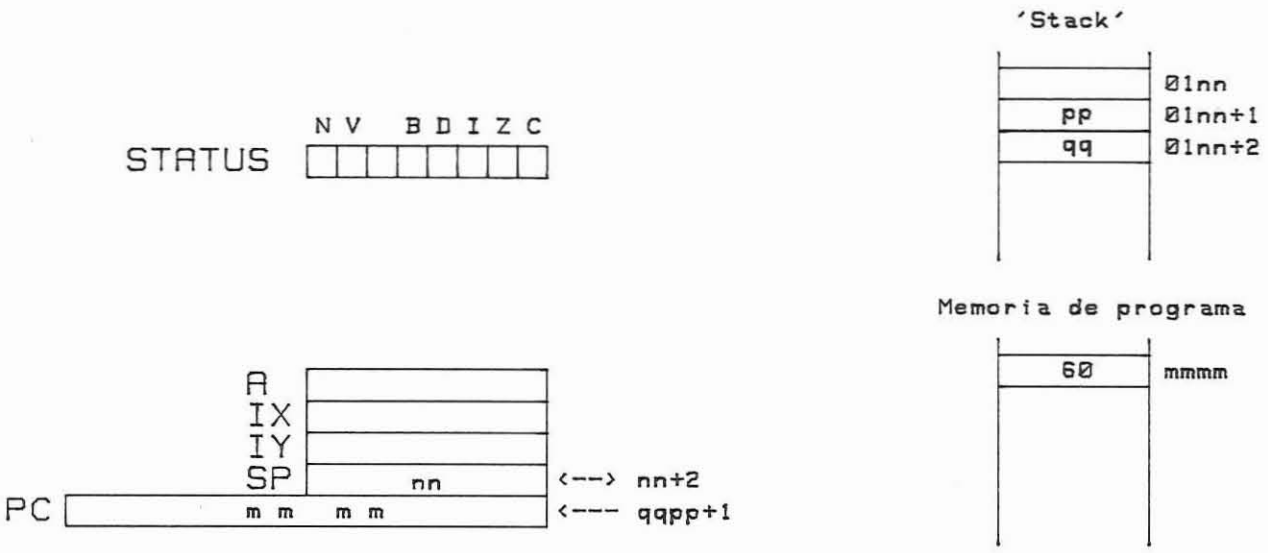
6

FLAGS DEL REGISTRO 'STATUS' AFECTADOS: todos

DESCRIPCION:

Cuando se ejecuta esta instrucción, el Program Counter recupera la dirección de la instrucción siguiente a ejecutar en el momento en que se produjo la interrupción, y el registro 'Status' recupera el estado que tenía en aquellos momentos. Estos tres bytes de información se extraen del 'Stack', donde se habían guardado en el momento de llegar la interrupción. Por tanto despues de esta instrucción el 'Stack Pointer' se incrementará tres unidades.

RTS **RETORNO DE SUBRUTINA**
(Direccionamiento implícito)



NUMERO DE CICLOS DE RELOJ:

6

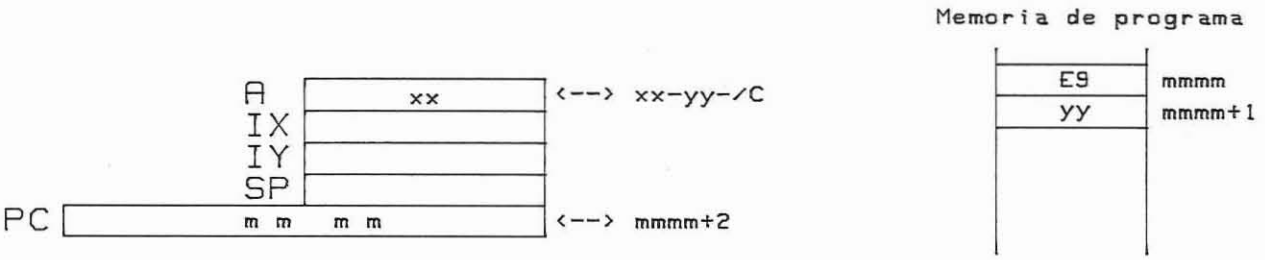
FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

DESCRIPCION:

Cuando se ejecuta esta instrucción, el Program Counter recupera la dirección de la instrucción siguiente a ejecutar en el momento en que se produjo la el salto a la subrutina. Esta dirección se extrae del 'Stack', donde se habían guardado en el momento de efectuarse el salto. Por tanto despues de esta instrucción el 'Stack Pointer' se incrementará dos unidades.

SBC #dato **RESTA**
(Direccionamiento inmediato)





NUMERO DE CICLOS DE RELOJ:

2 (añadir 1 en modo decimal)

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

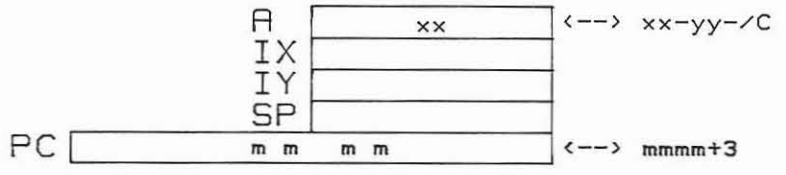
N: Se pone a 1 si el resultado es negativo.
V: Se pone a 1 si se produce "overflow".
Z: Se pone a 1 si el resultado es cero.
C: Se pone a 1 si hay acarreo.

DESCRIPCION:

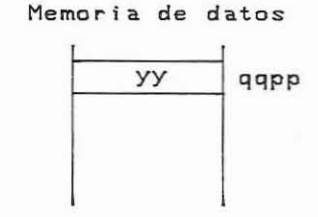
El contenido de la posición de memoria que sigue a la que contiene el código de operación y el complemento del bit de Carry del Registro de Status se restan del Acumulador. El resultado queda en el acumulador.

SBC dir **RESTA**
(Direccionamiento absoluto)









NUMERO DE CICLOS DE RELOJ:

4 (añadir 1 en modo decimal)

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

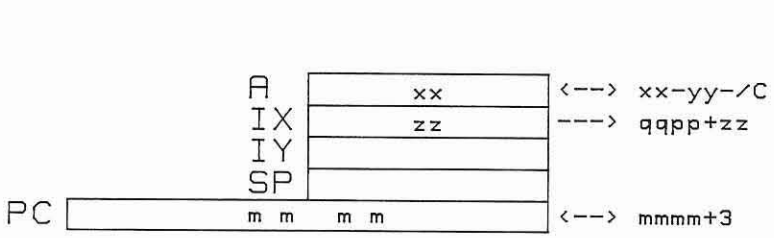
N: Se pone a 1 si el resultado es negativo.
V: Se pone a 1 si se produce "overflow".
Z: Se pone a 1 si el resultado es cero.
C: Se pone a 1 si hay acarreo.

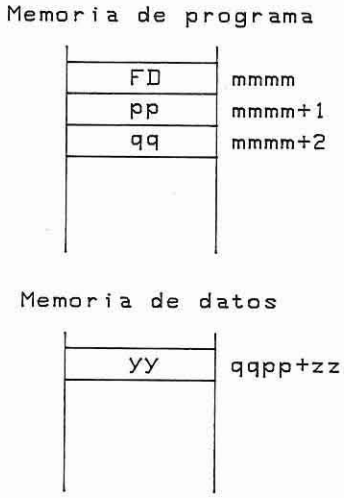
DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria donde se encuentra el dato que, junto con el bit de Carry del Registro de Status complementado, se restan del Acumulador. El resultado queda en el Acumulador.

SBC dir,X **RESTA**
(Direccionamiento absoluto indexado)







NUMERO DE CICLOS DE RELOJ:

4 (añadir 1 en modo decimal)
(añadir 1 si salta de página)

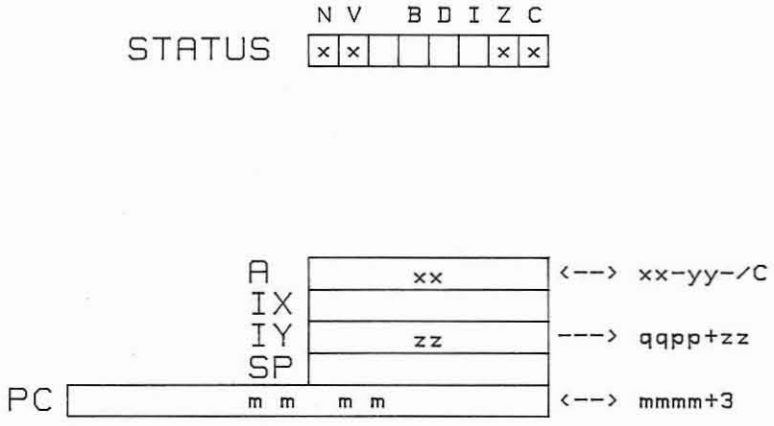
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
V: Se pone a 1 si se produce "overflow".
Z: Se pone a 1 si el resultado es cero.
C: Se pone a 1 si hay acarreo.

DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria que, sumada al contenido del registro X, indica la posición de memoria donde se encuentra el dato. Este dato y el bit de Carry del Registro de Status complementado se restan del Acumulador. El resultado queda en el Acumulador.

SBC dir,Y **RESTA**
(Direccionamiento absoluto indexado)





NUMERO DE CICLOS DE RELOJ:

4 (añadir 1 en modo decimal)
(añadir 1 si salta de página)

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

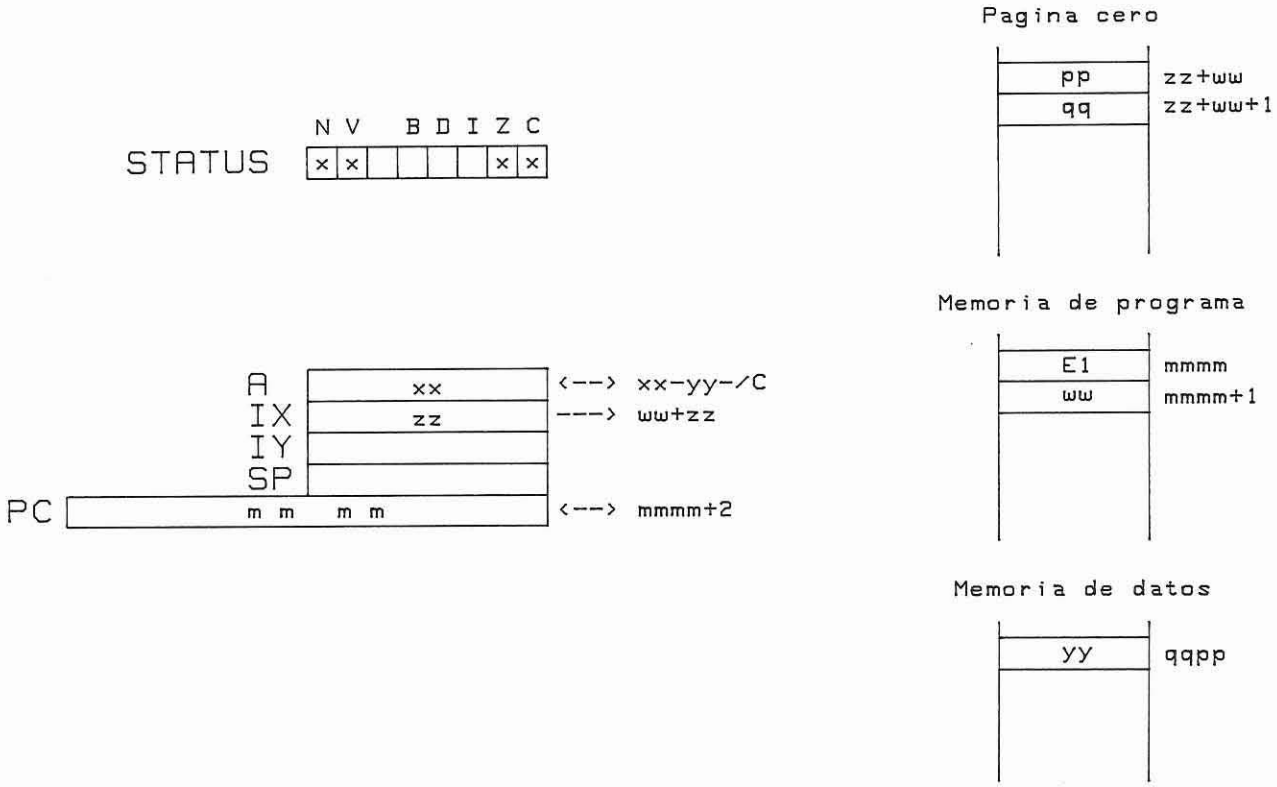
N: Se pone a 1 si el resultado es negativo.
V: Se pone a 1 si se produce "overflow".
Z: Se pone a 1 si el resultado es cero.
C: Se pone a 1 si hay acarreo.

DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria que, sumada al contenido del registro Y, indica la posición de memoria donde se encuentra el dato. Este dato y el bit de Carry del Registro de Status complementado se restan del Acumulador. El resultado queda en el Acumulador.

SBC (dir,X) **RESTA**

(Direccionamiento preindexado)



NUMERO DE CICLOS DE RELOJ:

6 (añadir 1 en modo decimal)

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

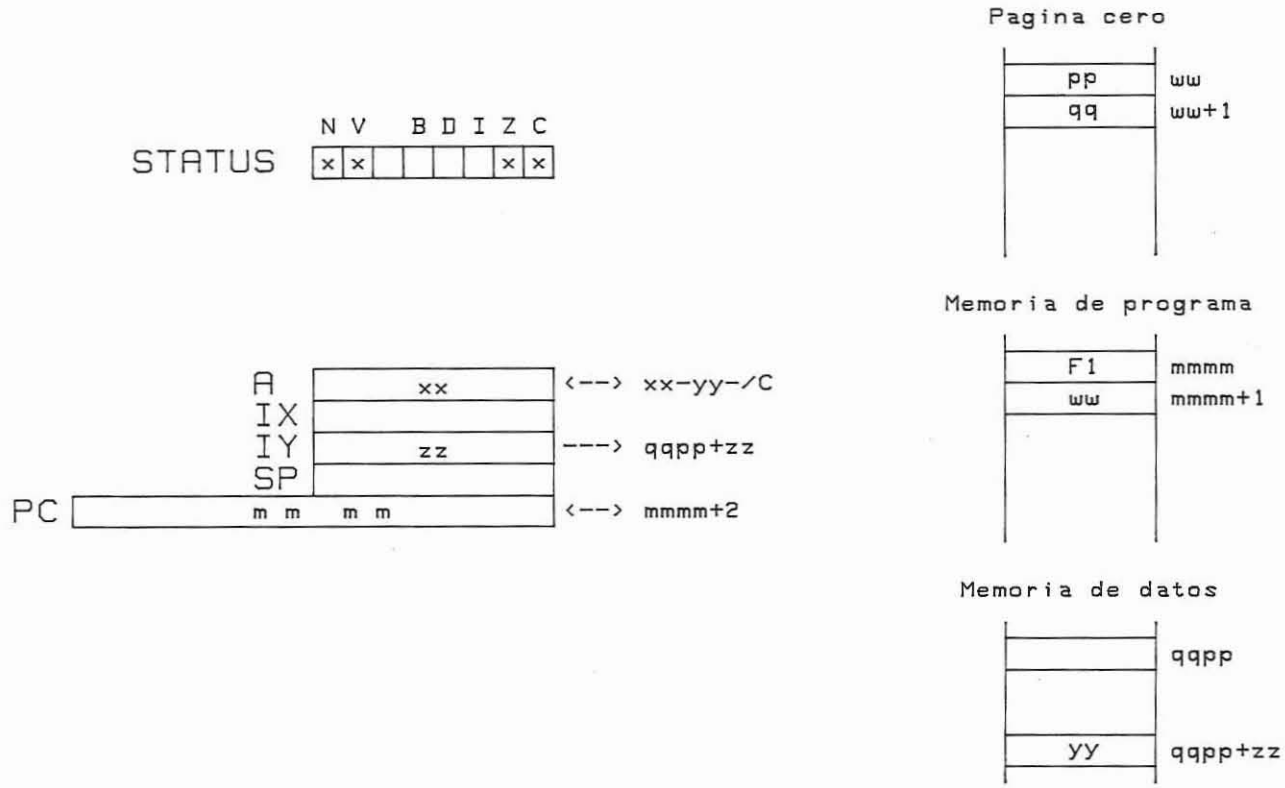
N: Se pone a 1 si el resultado es negativo.
V: Se pone a 1 si se produce "overflow".
Z: Se pone a 1 si el resultado es cero.
C: Se pone a 1 si hay acarreo.

DESCRIPCION:

El contenido de la posición de memoria que sigue a la que contiene el código de operación se suma con el contenido del registro X. El resultado (despreciando el acarreo si se ha producido) corresponde a la dirección de memoria en página cero que contiene la parte baja de la dirección efectiva del dato. La parte alta de la dirección efectiva del dato está en la siguiente posición de memoria en pagina cero. De la dirección efectiva del dato se toma el byte que junto con el bit de Carry del Registro de Status complementado se restan del Acumulador. El resultado queda en el Acumulador.

SBC (dir),Y **RESTA**

(Direccionamiento postindexado)



NUMERO DE CICLOS DE RELOJ:

5 (añadir 1 en modo decimal)
(añadir 1 si salta de página)

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

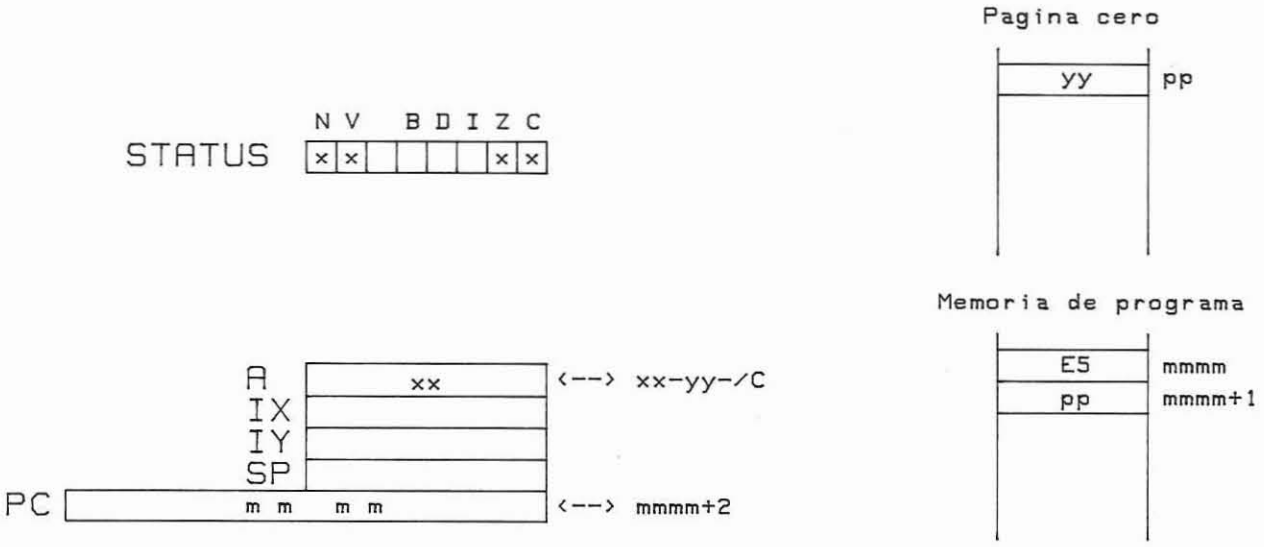
N: Se pone a 1 si el resultado es negativo.
V: Se pone a 1 si se produce "overflow".
Z: Se pone a 1 si el resultado es cero.
C: Se pone a 1 si hay acarreo.

DESCRIPCION:

El contenido de la posición de memoria que sigue a la que contiene el código de operación contiene una dirección de memoria de página cero. En esta dirección y en la siguiente se encuentran, en orden inverso (primero el byte de menor peso y luego el byte mayor peso), los dos bytes que forman la dirección de memoria que sumada al contenido del registro Y, nos dará la dirección de memoria donde se encuentra el dato. Este dato y el bit de Carry del Registro de Status complementado se restan del Acumulador. El resultado queda en el Acumulador.

SBC dir **RESTA**

(Direccionamiento página cero)



NUMERO DE CICLOS DE RELOJ:

3 (añadir 1 en modo decimal)

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

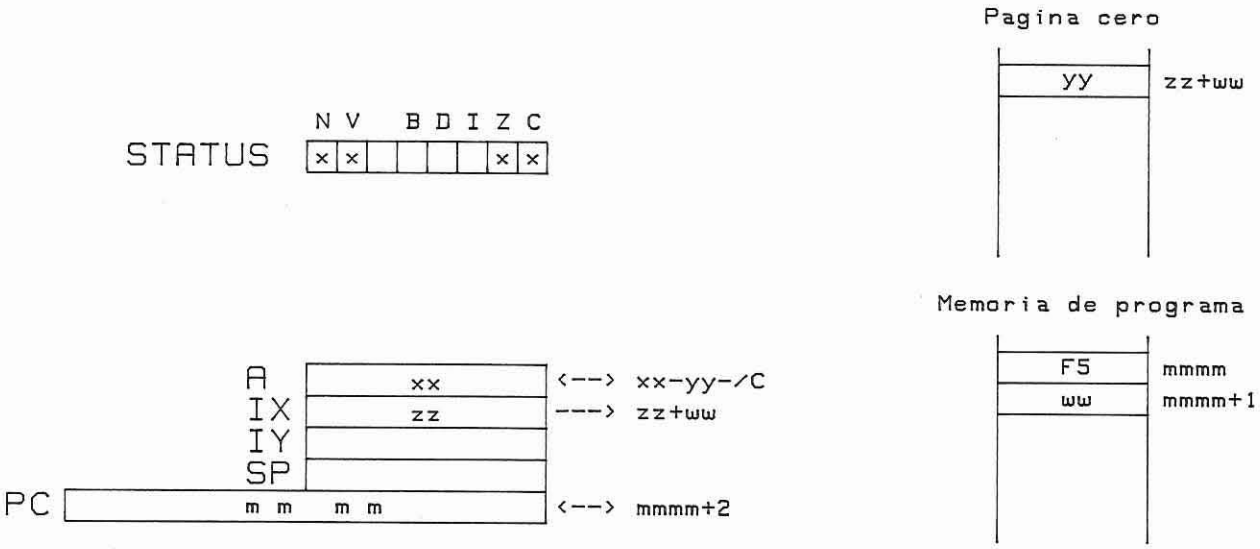
N: Se pone a 1 si el resultado es negativo.
V: Se pone a 1 si se produce "overflow".
Z: Se pone a 1 si el resultado es cero.
C: Se pone a 1 si hay acarreo.

DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la dirección de memoria de página cero donde se encuentra el dato que junto con el bit de Carry del Registro de Status complementado se resta del Acumulador. El resultado queda en el Acumulador.

SBC dir,X **RESTA**

(Direccionamiento página cero indexado)



NUMERO DE CICLOS DE RELOJ:

4 (añadir 1 en modo decimal)

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

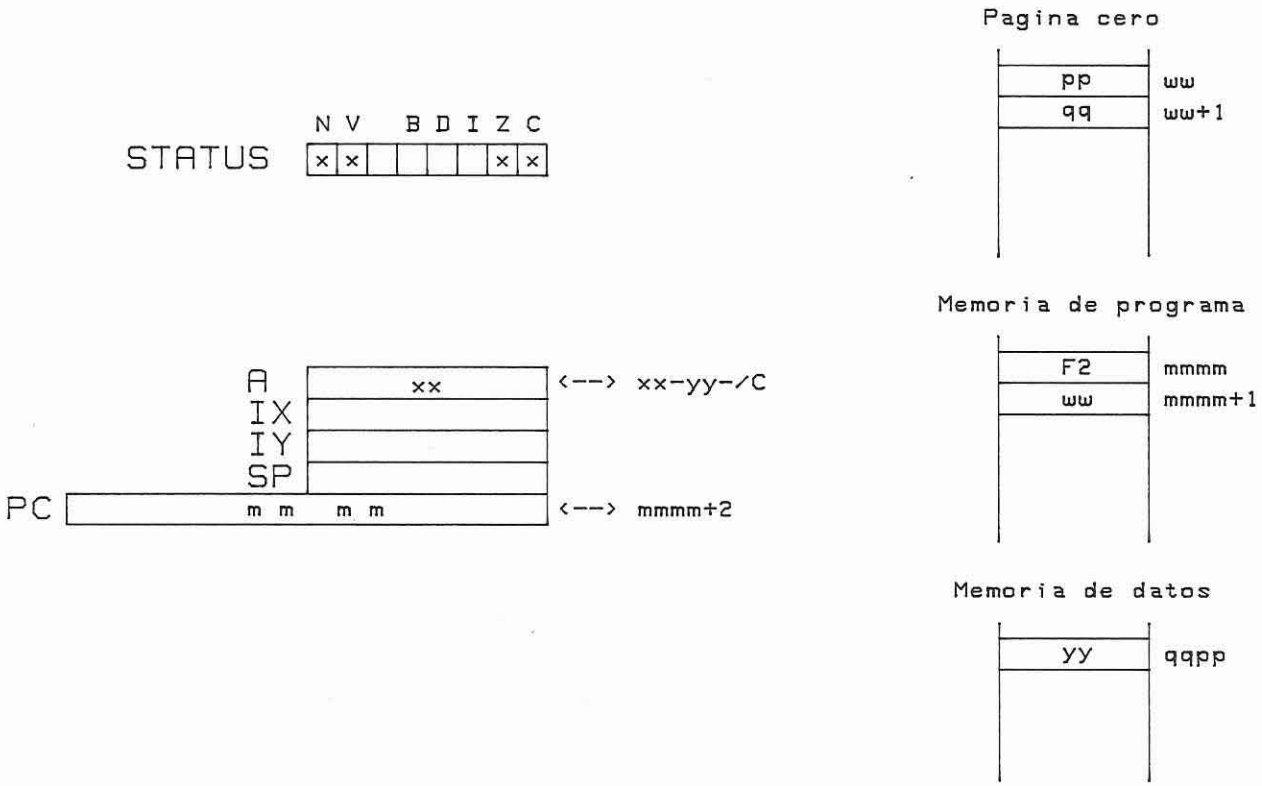
N: Se pone a 1 si el resultado es negativo.
V: Se pone a 1 si se produce "overflow".
Z: Se pone a 1 si el resultado es cero.
C: Se pone a 1 si hay acarreo.

DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la parte baja de la dirección de memoria que, sumada al contenido del registro X, da la dirección de memoria de página cero donde se encuentra el dato. Este dato y el bit de Carry del Registro de Status complementado se restan del Acumulador. El resultado queda en el Acumulador.

SBC (dir) * **RESTA**

(Direccionamiento indirecto mejorado)



NUMERO DE CICLOS DE RELOJ:

5 (añadir 1 en modo decimal)

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

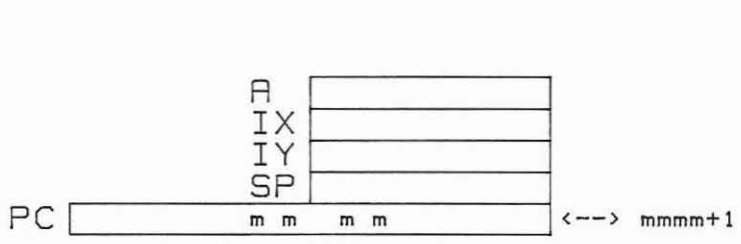
N: Se pone a 1 si el resultado es negativo.
V: Se pone a 1 si se produce "overflow".
Z: Se pone a 1 si el resultado es cero.
C: Se pone a 1 si hay acarreo.

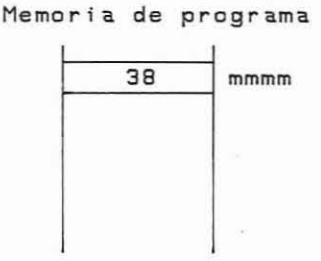
DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la parte baja de una dirección de memoria de página cero. En esta dirección y en la siguiente se encuentran, en orden inverso (primero byte de menor peso y despues byte de mayor peso), los dos bytes que forman la dirección de memoria donde se encuentra el dato que, junto con el bit de Carry del Registro de Status complementado, se resta del Acumulador. El resultado queda en el Acumulador.

SEC **PONER A 1 EL BIT DE ACARREO**
(Direccionamiento implícito)







NUMERO DE CICLOS DE RELOJ:

2

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

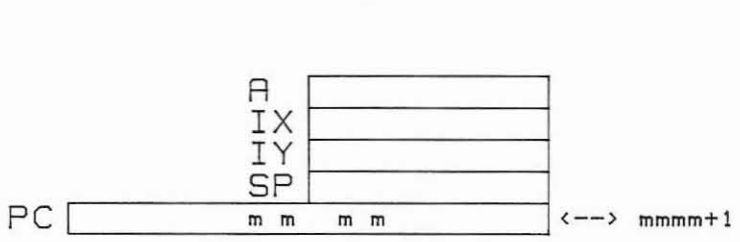
C: Se pone a 1.

DESCRIPCION:

Cuando se ejecuta esta instrucción, el flag de Carry del Registro de Status se pone a 1 fuera cual fuera el valor que tenía anteriormente.

SED **PONER A 1 EL BIT DE DECIMAL**
(Direccionamiento implícito)







NUMERO DE CICLOS DE RELOJ:

2

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

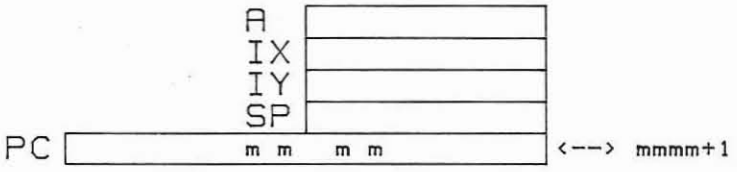
D: Se pone a 1.

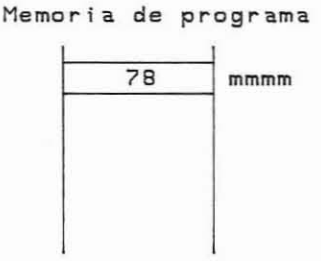
DESCRIPCION:

Cuando se ejecuta esta instrucción, el flag de Modo Decimal del Registro de Status se pone a 1 fuera cual fuera el valor que tenía anteriormente. El modo decimal queda activado.

SEI **PONER A 1 EL BIT DE INTERRUPCIONES**
(Direccionamiento implícito)







NUMERO DE CICLOS DE RELOJ:

2

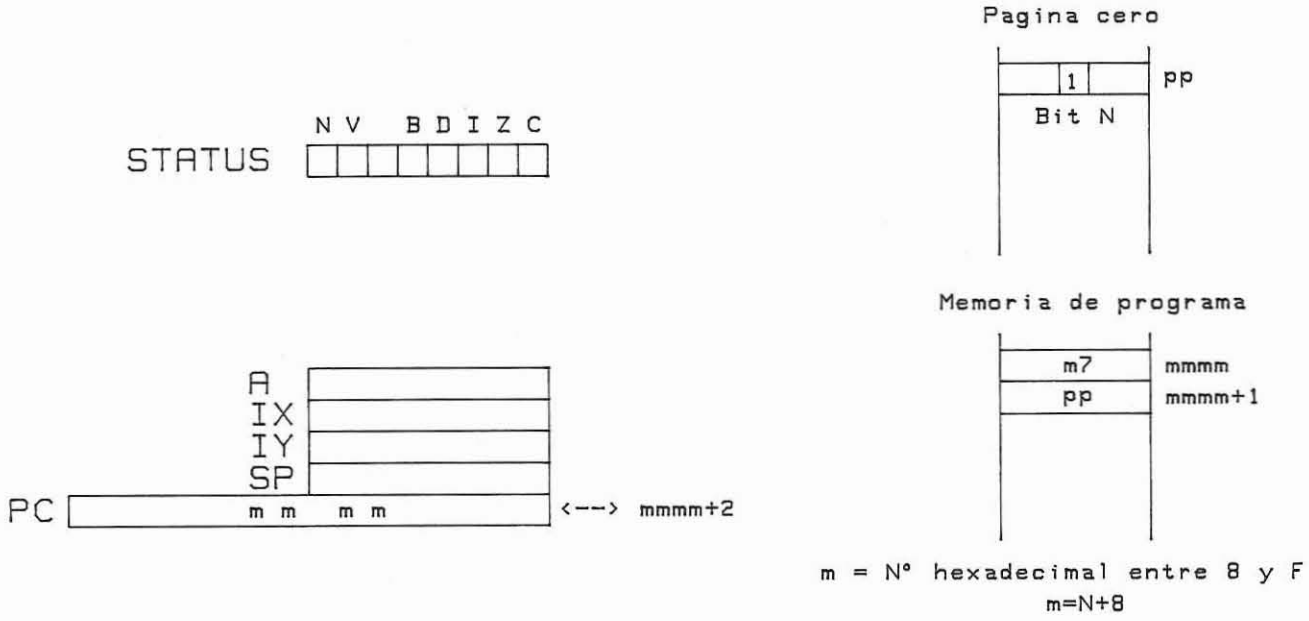
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

I: Se pone a 1.

DESCRIPCION:

Cuando se ejecuta está instrucción, el flag de Interrupción del Registro de Status se pone a 1 fuera cual fuera el valor que tenía anteriormente. Por tanto las interrupciones que lleguen a partir de este instante serán enmascaradas.

SMB N dir + **PONE A 1 EL BIT N DE LA POSICION DE MEMORIA INDICADA**
(Direccionamiento implícito)



NUMERO DE CICLOS DE RELOJ:

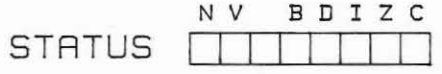
5

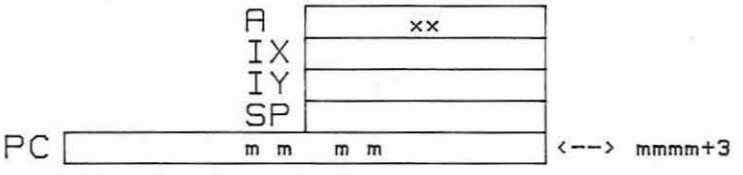
FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

DESCRIPCION:

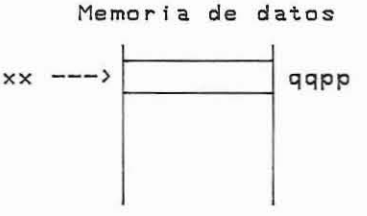
Las 4 líneas de mayor peso del byte que indica el código de operación de esta instrucción más 8 indica el bit que se pondrá a cero. La dirección de memoria donde se encuentra el dato cuyo bit N se pondrá a 0 biene indicada por el segundo byte de la instrucción.

STA dir **PASAR EL CONTENIDO DEL ACUMULADOR A MEMORIA**
(Direccionamiento absoluto)









NUMERO DE CICLOS DE RELOJ:

4

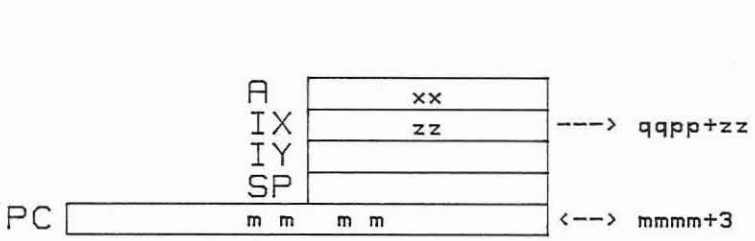
FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

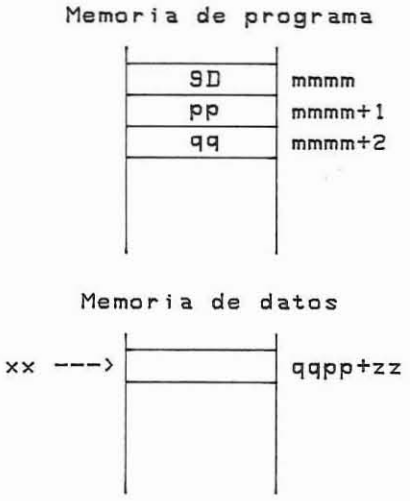
DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria donde se carga el dato del Acumulador.

STA dir,X **PASAR EL CONTENIDO DEL ACUMULADOR A MEMORIA**
(Direccionamiento absoluto indexado)







NUMERO DE CICLOS DE RELOJ:

5

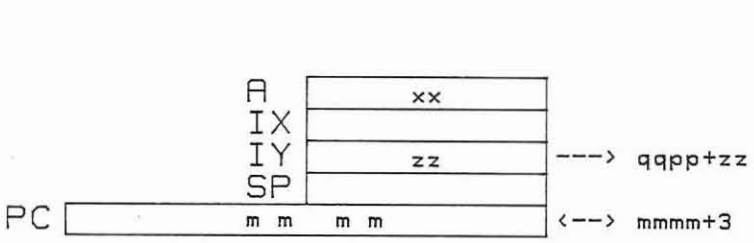
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:
ninguno

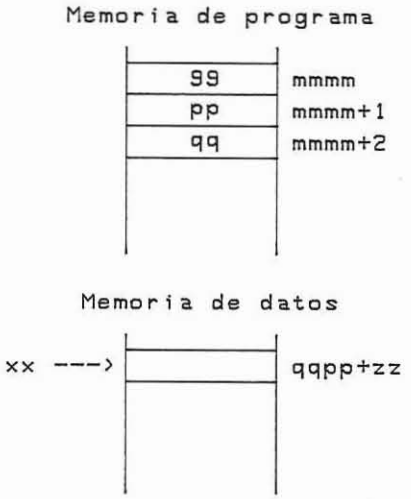
DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria que, sumada al contenido del registro X, indica la posición de memoria donde se carga el dato que se encuentra en el Acumulador.

STA dir,Y **PASAR EL CONTENIDO DEL ACUMULADOR A MEMORIA**
(Direccionamiento absoluto indexado)







NUMERO DE CICLOS DE RELOJ:

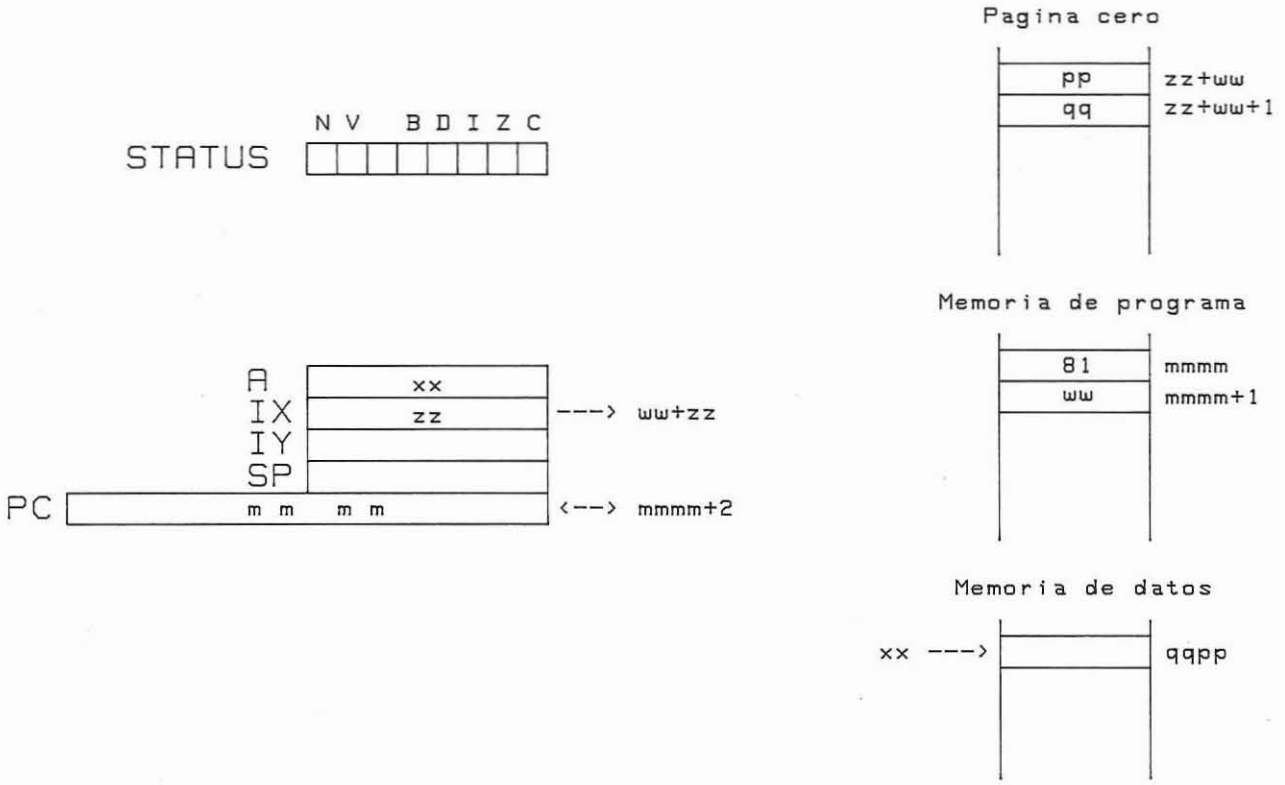
5

FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria que, sumada al contenido del registro Y, indica la posición de memoria donde se carga el dato que se encuentra en el Acumulador.

STA (dir,X) **PASAR EL CONTENIDO DEL ACUMULADOR A MEMORIA**
(Direccionamiento preindexado)



NUMERO DE CICLOS DE RELOJ:

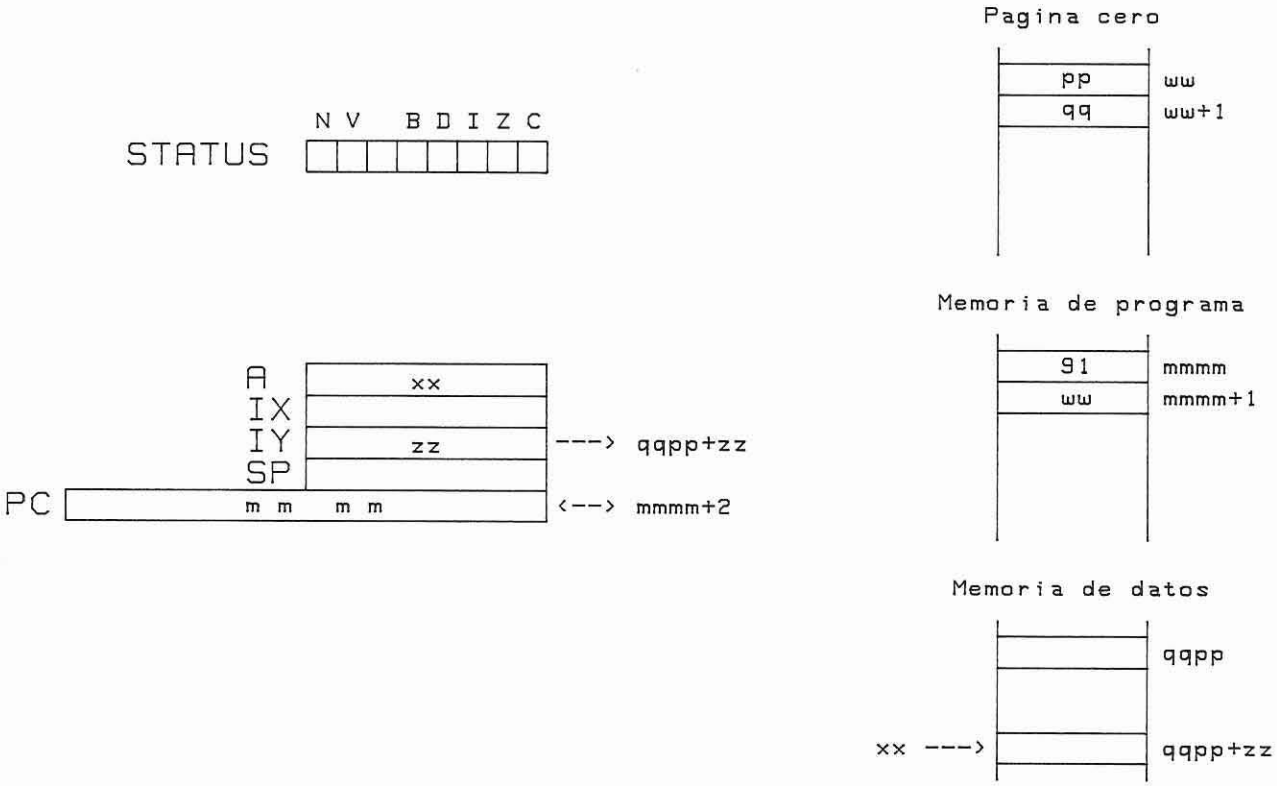
6

FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

DESCRIPCION:

El contenido de la posición de memoria que sigue a la que contiene el código de operación se suma con el contenido del registro X. El resultado (despreciando el acarreo si se ha producido) corresponde a la dirección de memoria en página cero que contiene la parte baja de la dirección efectiva del dato. La parte alta de la dirección efectiva del dato está en la siguiente posición de memoria en pagina cero. En la dirección efectiva del dato se carga el byte que contiene el Acumulador.

STA (dir),Y **PASAR EL CONTENIDO DEL ACUMULADOR A MEMORIA**
(Direccionamiento postindexado)



NUMERO DE CICLOS DE RELOJ:

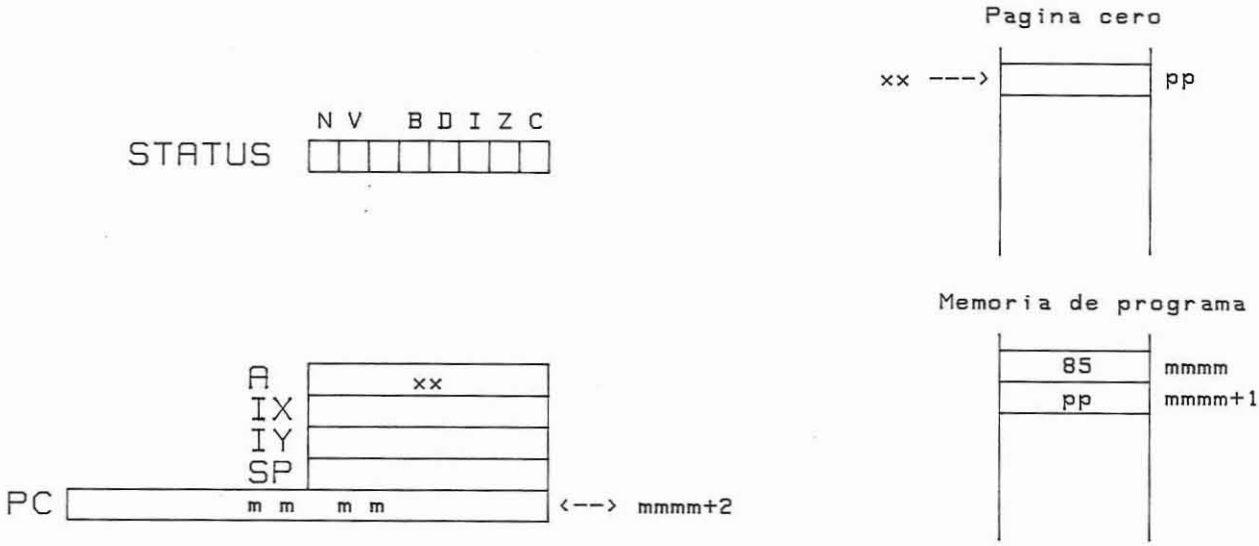
6

FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

DESCRIPCION:

El contenido de la posición de memoria que sigue a la que contiene el código de operación contiene una dirección de memoria de página cero. En esta dirección y en la siguiente se encuentran, en orden inverso (primero el byte de menor peso y luego el byte mayor peso), los dos bytes que forman la dirección de memoria que sumada al contenido del registro Y, nos dará la dirección de memoria donde se carga el dato que se encuentra en el Acumulador.

STA dir **PASAR EL CONTENIDO DEL ACUMULADOR A MEMORIA**
(Direccionamiento página cero)



NUMERO DE CICLOS DE RELOJ:

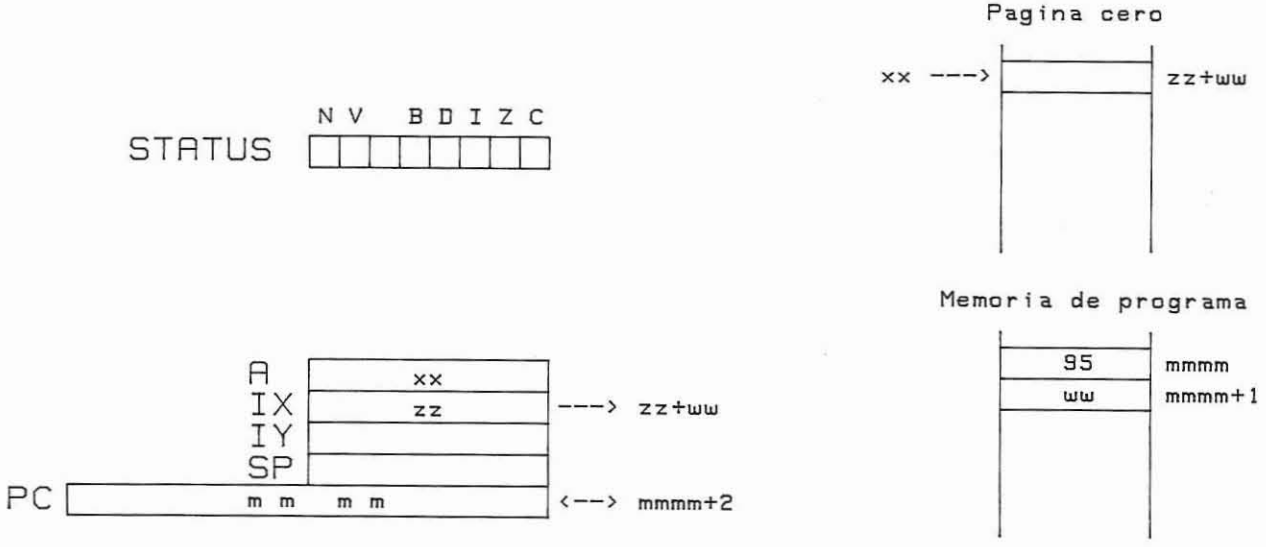
3

FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la dirección de memoria de página cero donde se carga el dato que se encuentra en el Acumulador.

STA dir,X **PASAR EL CONTENIDO DEL ACUMULADOR A MEMORIA**
(Direccionamiento página cero indexado)



NUMERO DE CICLOS DE RELOJ:

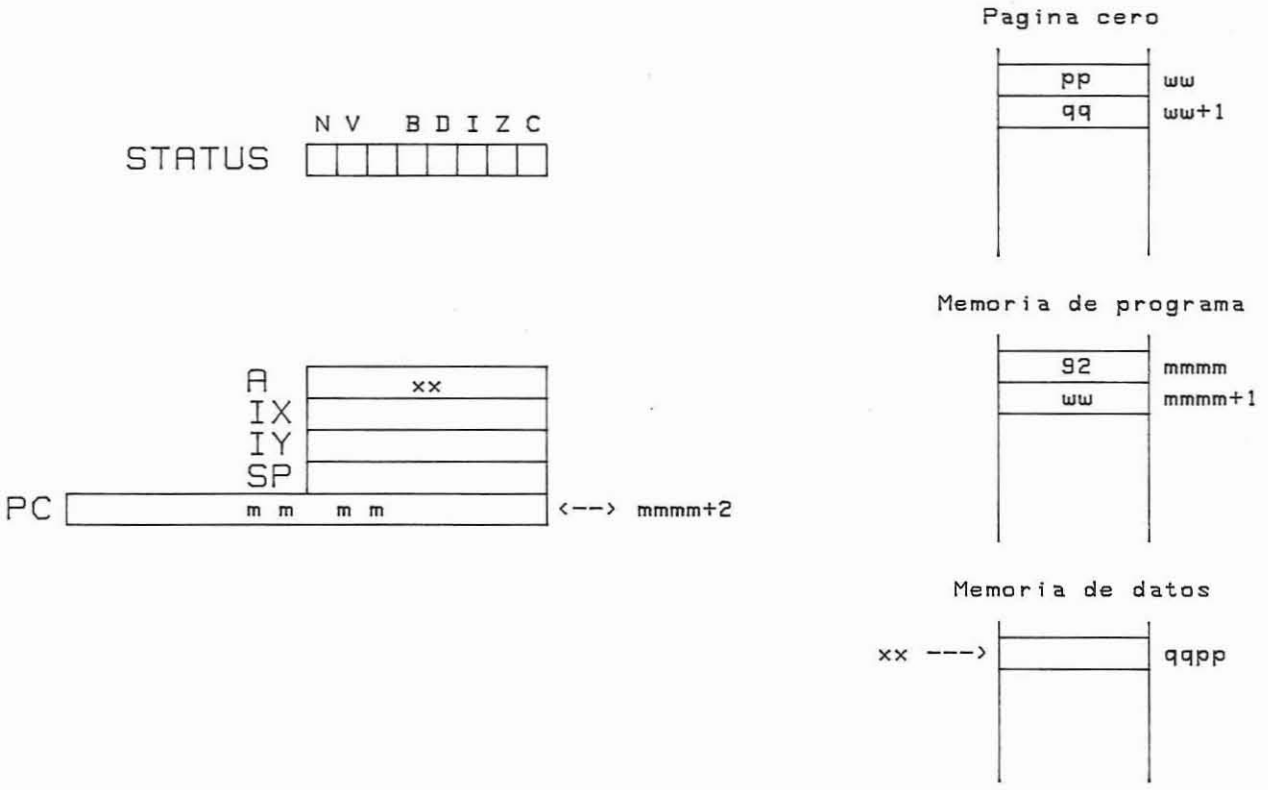
4

FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la parte baja de la dirección de memoria que, sumada al contenido del registro X, da la dirección de memoria de página cero donde se carga el dato que se encuentra en el Acumulador.

STA (dir) * **PASAR EL CONTENIDO DEL ACUMULADOR A MEMORIA**
(Direccionamiento indirecto mejorado)



NUMERO DE CICLOS DE RELOJ:

5

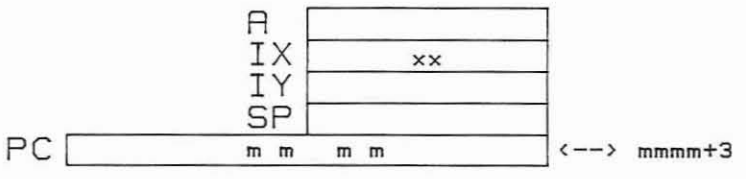
FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

DESCRIPCION:

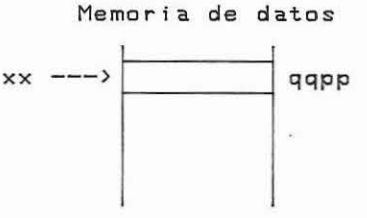
La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la parte baja de una dirección de memoria de página cero. En esta dirección y en la siguiente se encuentran, en orden inverso (primero byte de menor peso y despues byte de mayor peso), los dos bytes que forman la dirección de memoria donde se carga el dato que se encuentra en el Acumulador.

STX dir **PASAR EL CONTENIDO DEL REGISTRO X A MEMORIA**
(Direccionamiento absoluto)









NUMERO DE CICLOS DE RELOJ:

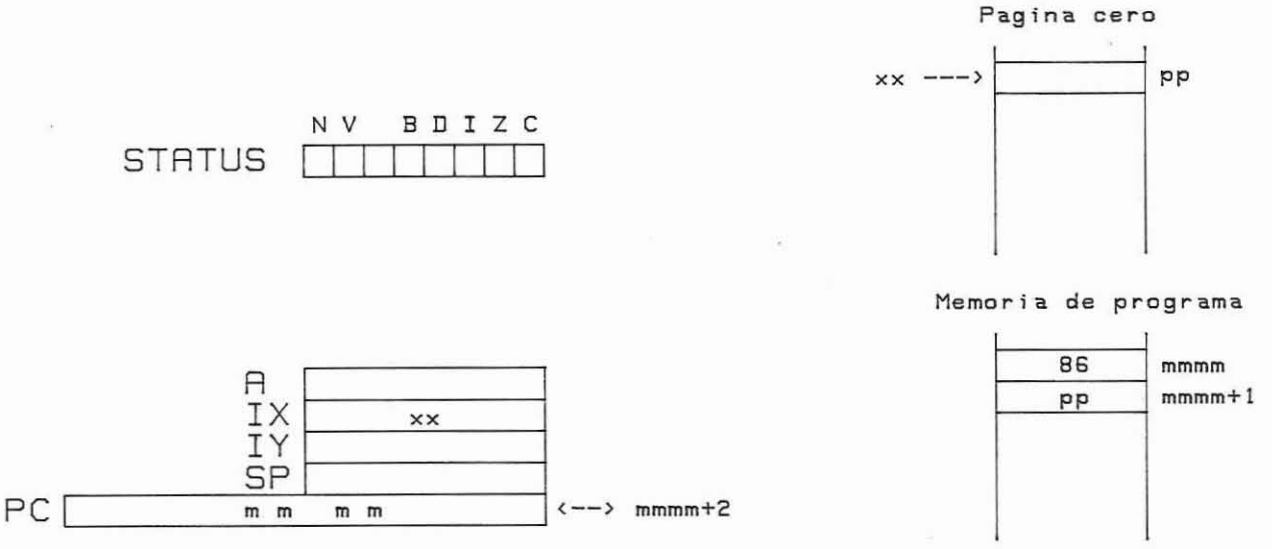
4

FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria donde se carga el dato que se encuentra en el Registro X.

STX dir **PASAR EL CONTENIDO DEL REGISTRO X A MEMORIA**
(Direccionamiento página cero)



NUMERO DE CICLOS DE RELOJ:

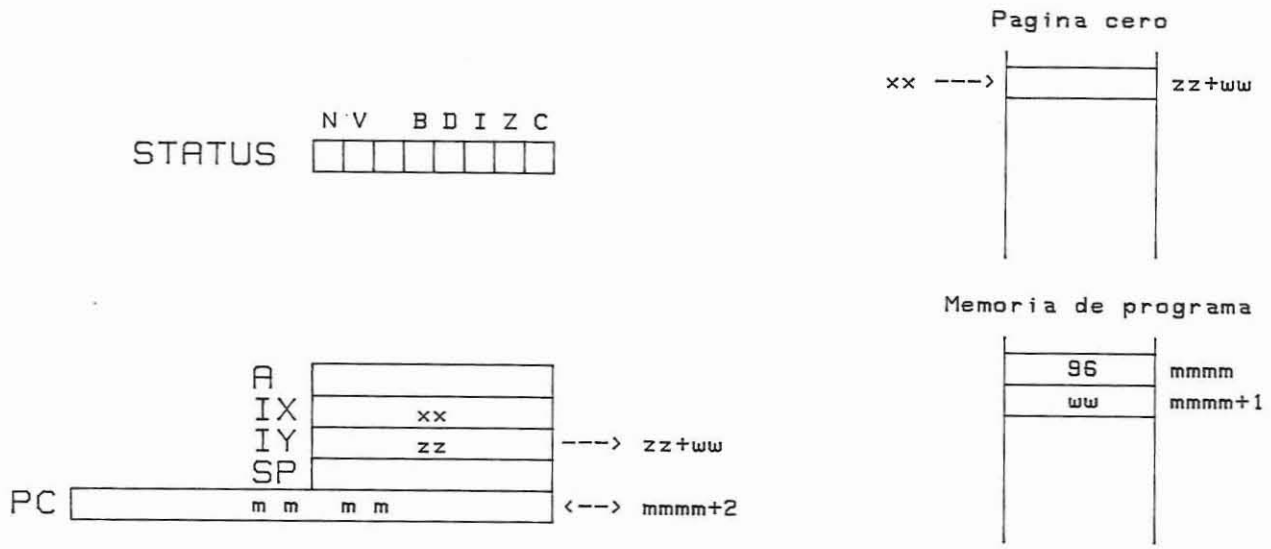
3

FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la dirección de memoria de página cero donde se carga el dato que se encuentra en el Registro X.

STX dir,Y **PASAR EL CONTENIDO DEL REGISTRO X A MEMORIA**
(Direccionamiento página cero indexado)



NUMERO DE CICLOS DE RELOJ:

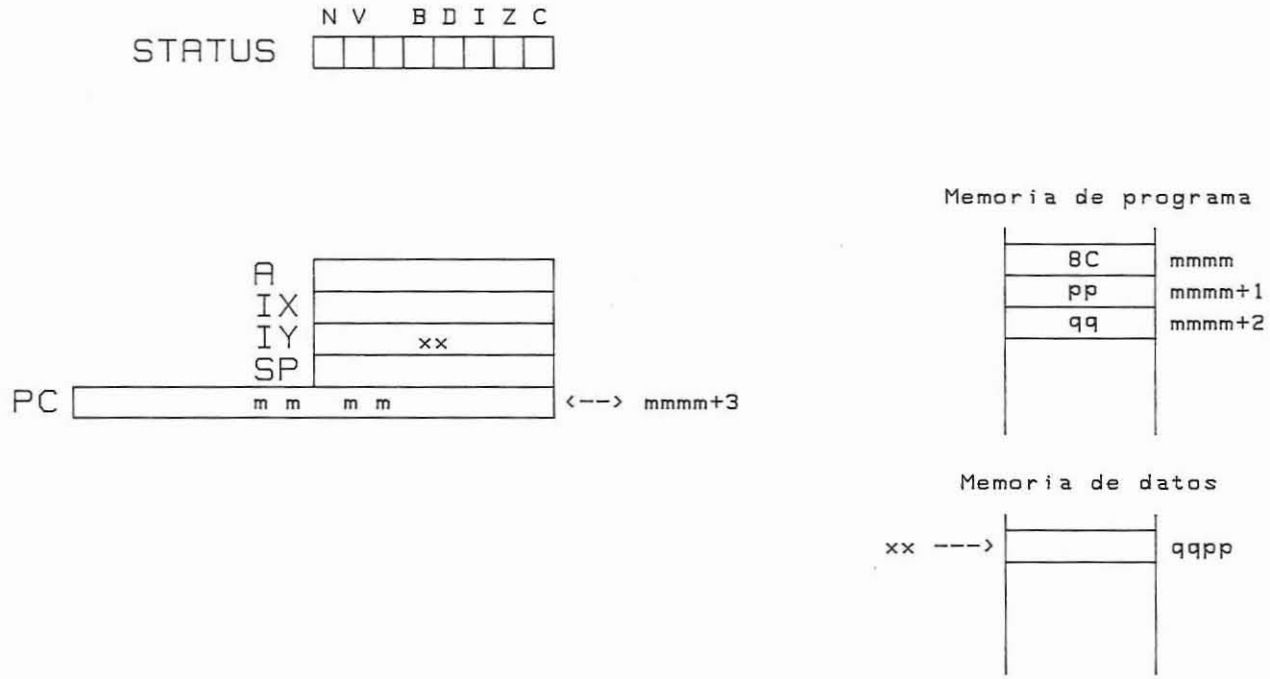
4

FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la parte baja de la dirección de memoria que, sumada al contenido del registro Y, da la dirección de memoria de página cero donde se carga el dato que se encuentra en el Registro X.

STY dir **PASAR EL CONTENIDO DEL REGISTRO Y A MEMORIA**
(Direccionamiento absoluto)



NUMERO DE CICLOS DE RELOJ:

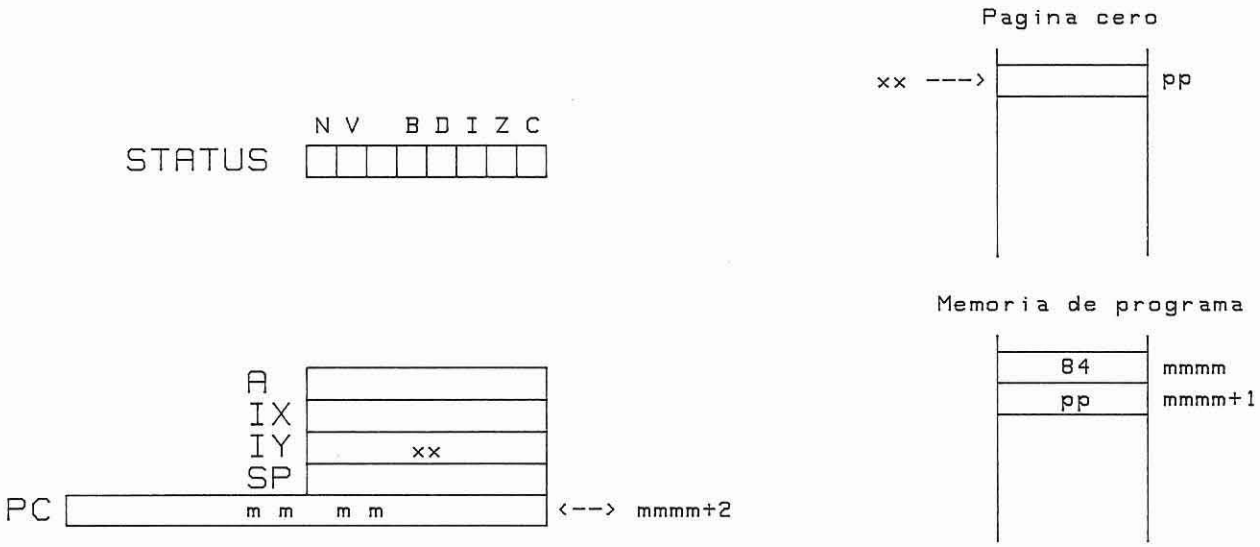
4

FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria donde se carga el dato que se encuentra en el Registro Y.

STY dir **PASAR EL CONTENIDO DEL REGISTRO Y A MEMORIA**
(Direccionamiento página cero)



NUMERO DE CICLOS DE RELOJ:

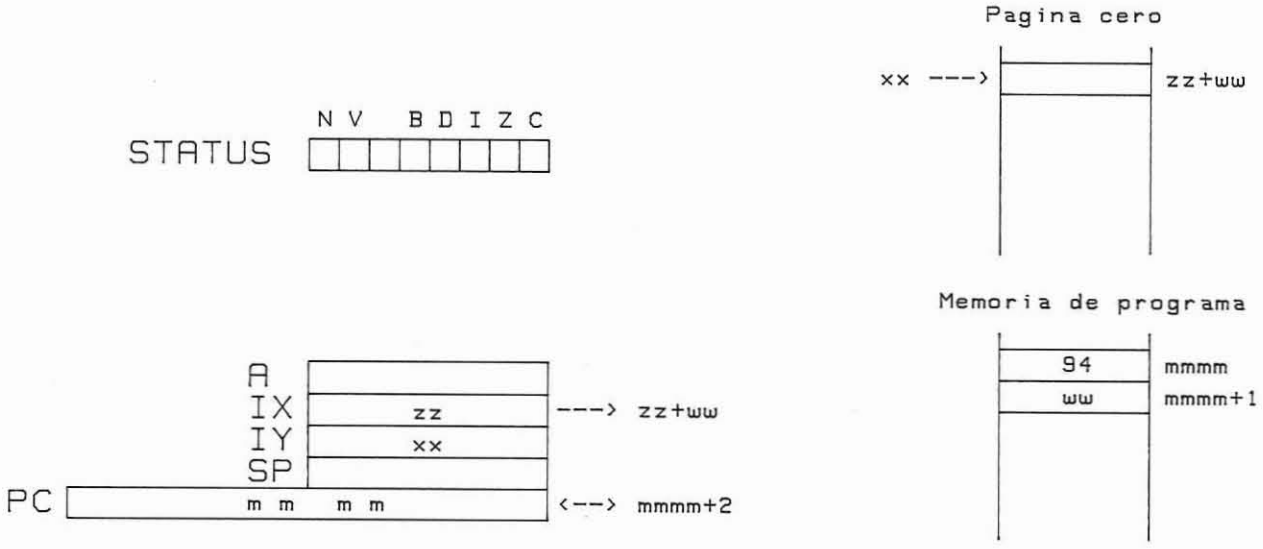
3

FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la dirección de memoria de página cero donde se carga el dato que se encuentra en el Registro Y.

STY dir,X **PASAR EL CONTENIDO DEL REGISTRO Y A MEMORIA**
(Direccionamiento página cero indexado)



NUMERO DE CICLOS DE RELOJ:

4

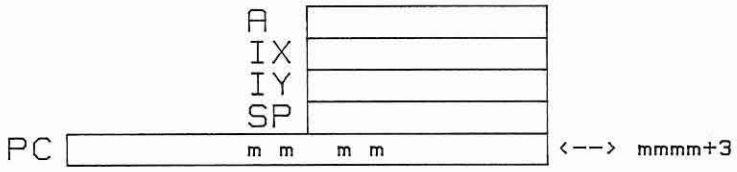
FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

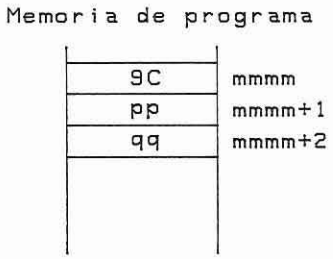
DESCRIPCION:

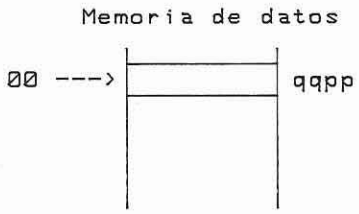
La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la parte baja de la dirección de memoria que, sumada al contenido del registro X, da la dirección de memoria de página cero donde se carga el dato que se encuentra en el Registro Y.

STZ dir * **PODE A 0 LA POSICION DIRECCIONADA**
(Direccionamiento absoluto)









qqpp

NUMERO DE CICLOS DE RELOJ:

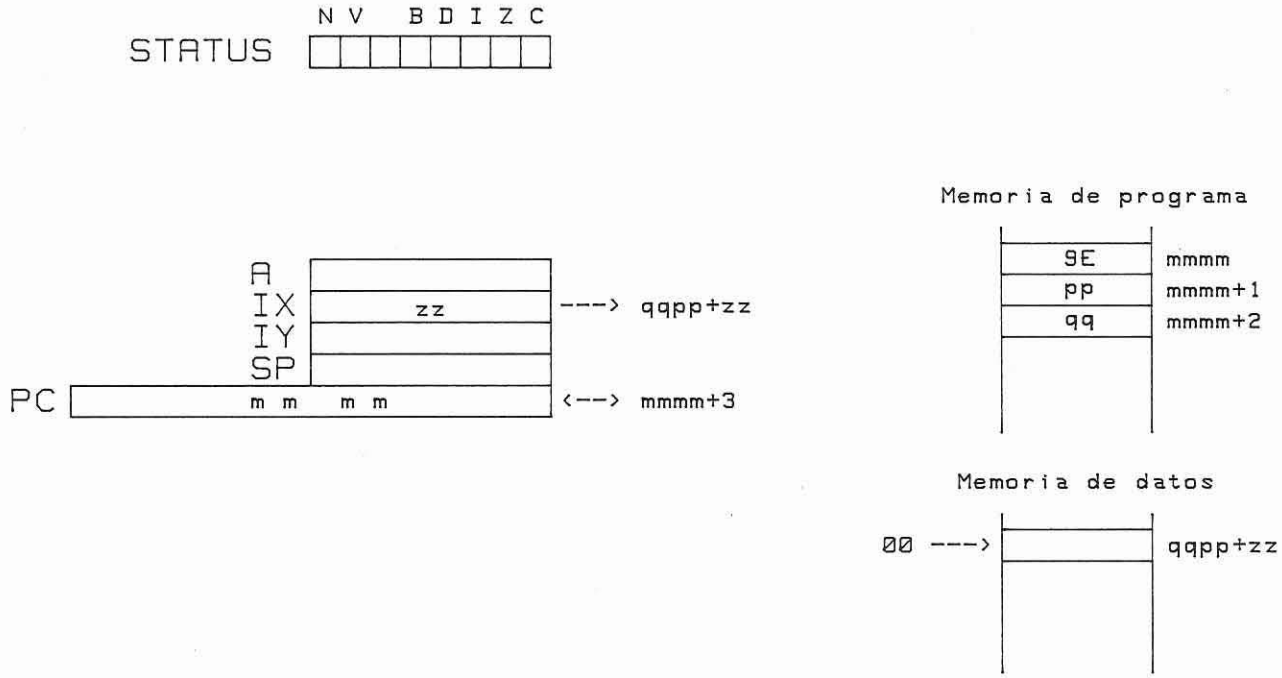
4

FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria donde se carga el byte 00.

STZ dir,X * **PONE A 0 LA POSICION DIRECCIONADA**
(Direccionamiento absoluto indexado)



NUMERO DE CICLOS DE RELOJ:

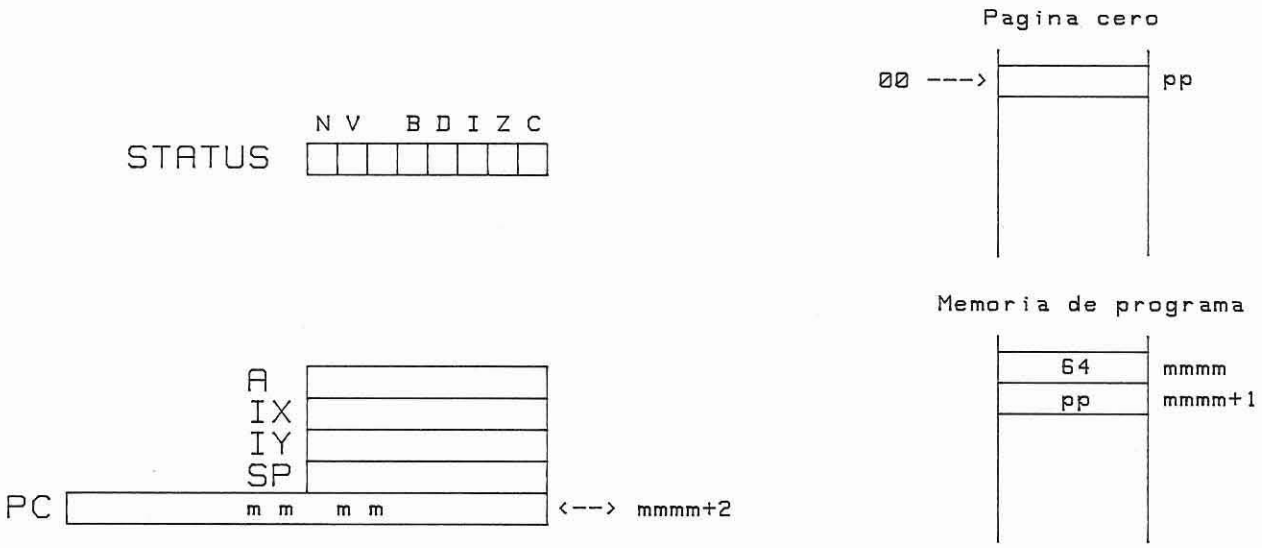
5

FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria que, sumada al contenido del registro X, indica la posición de memoria donde se carga el dato 00.

STZ dir * **PONE A 0 LA POSICION DIRECCIONADA**
(Direccionamiento página cero)



NUMERO DE CICLOS DE RELOJ:

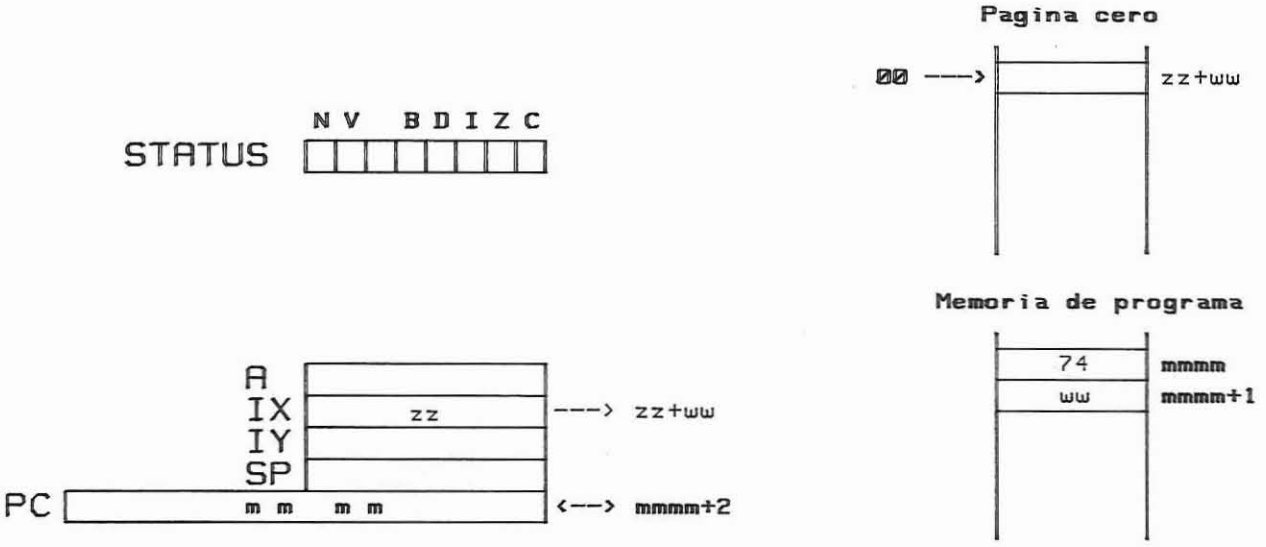
3

FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la dirección de memoria de página cero donde se carga el dato 00.

STZ dir,X * **PONE A 0 LA POSICION DIRECCIONADA**
(Direccionamiento página cero indexado)



NUMERO DE CICLOS DE RELOJ:

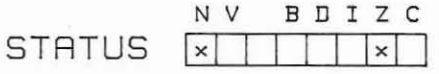
4

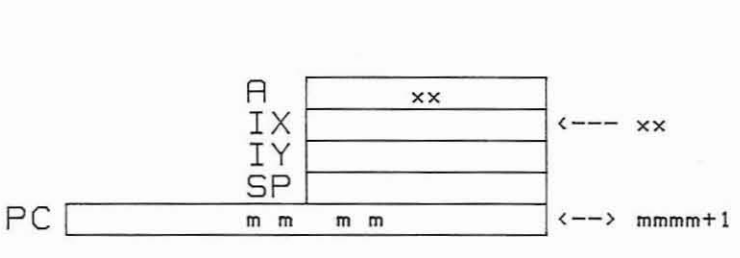
FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

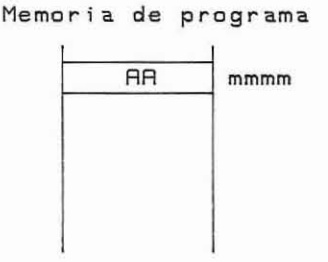
DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la parte baja de la dirección de memoria que, sumada al contenido del registro X, da la dirección de memoria de página cero donde se carga el dato 00.

TAX **TRANSFIERE EL CONTENIDO DEL ACUMULADOR AL REGISTRO X**
(Direccionamiento implícito)







NUMERO DE CICLOS DE RELOJ:

2

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

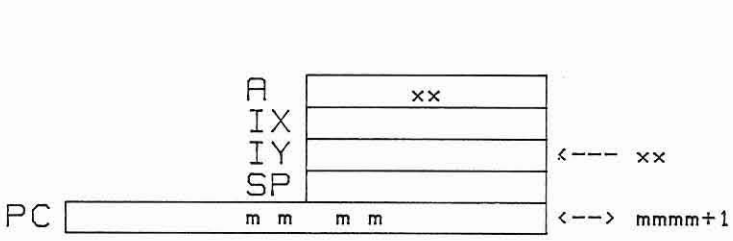
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

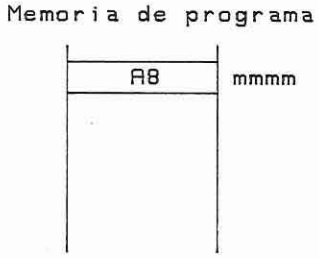
DESCRIPCION:

Cuando se ejecuta esta instrucción, el contenido del Acumulador pasa al Registro X.

TAY **TRANSFIERE EL CONTENIDO DEL ACUMULADOR AL REGISTRO Y**
(Direccionamiento implícito)







NUMERO DE CICLOS DE RELOJ:

2

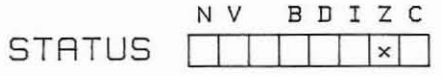
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

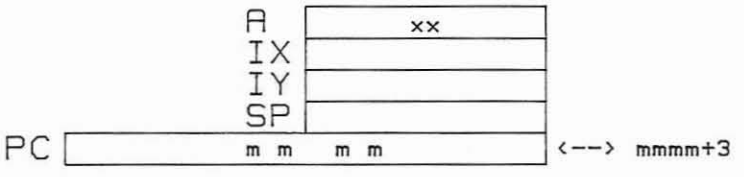
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

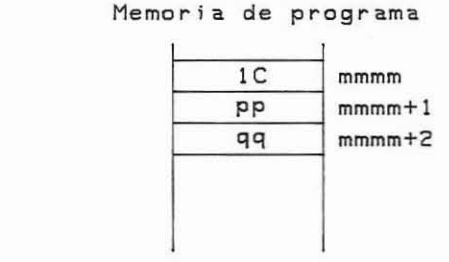
DESCRIPCION:

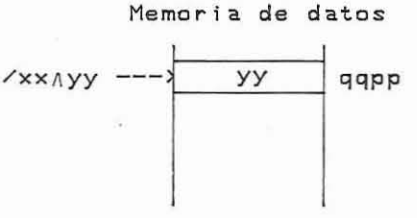
Cuando se ejecuta esta instrucción, el contenido del Acumulador pasa al Registro Y.

TRB dir * COMPRUEBA Y PONE A 0 LOS BITS QUE SE CORRESPONDEN CON UNOS
(Direccionamiento absoluto)









NUMERO DE CICLOS DE RELOJ:

6

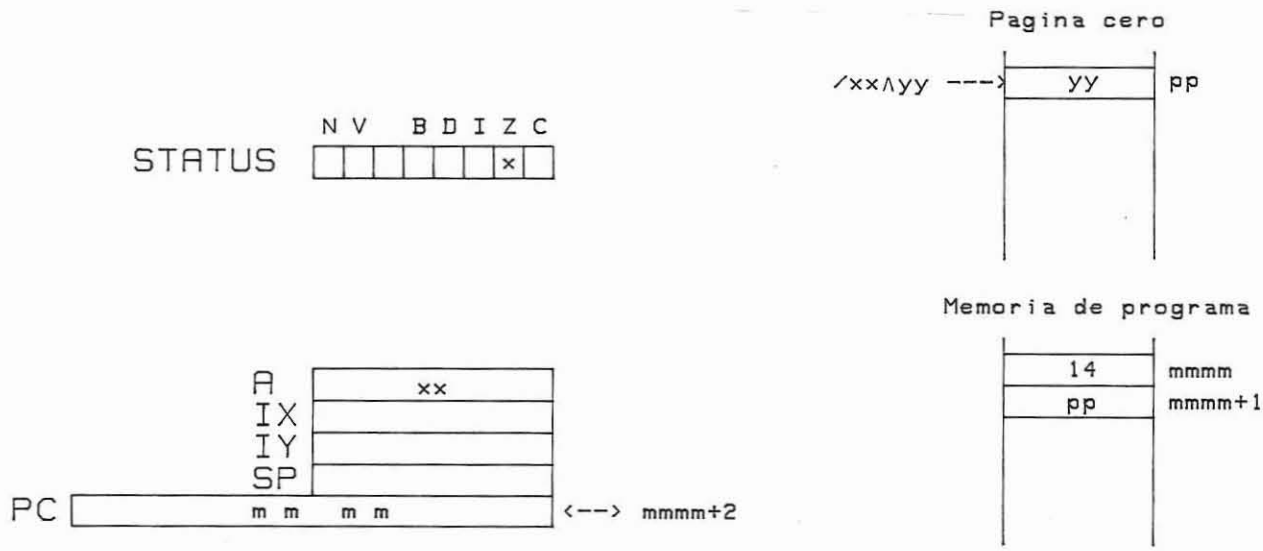
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria donde se encuentra el dato. Todas las posiciones de este dato que coincidan con unos del Acumulador quedarán a "0" y todas las que coincidan con ceros del Acumulador quedarán a igual. El resultado queda en la misma posición de memoria.

TRB dir * COMPRUEBA Y PONE A 0 LOS BITS QUE SE CORRESPONDEN CON UNOS (Direccionamiento página cero)



NUMERO DE CICLOS DE RELOJ:

5

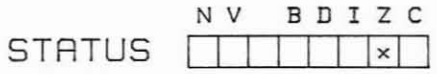
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

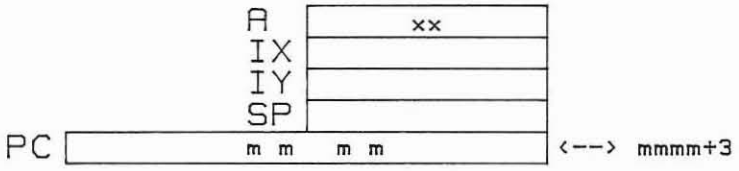
Z: Se pone a 1 si el resultado es cero.

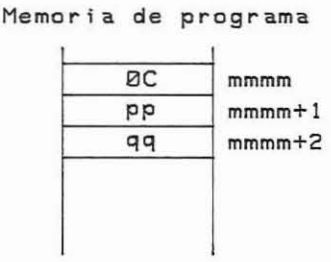
DESCRIPCION:

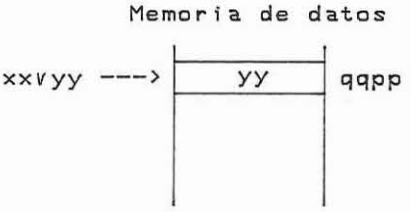
La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la dirección de memoria de página cero donde se encuentra el dato. Todas las posiciones de este dato que coincidan con unos del Acumulador quedarán a "0" y todas las que coincidan con ceros del Acumulador quedarán a igual. El resultado queda en la misma posición de memoria.

TSB dir * COMPRUEBA Y PONE A 1 LOS BITS QUE SE CORRESPONDEN CON UNOS
(Direccionamiento absoluto)









NUMERO DE CICLOS DE RELOJ:

6

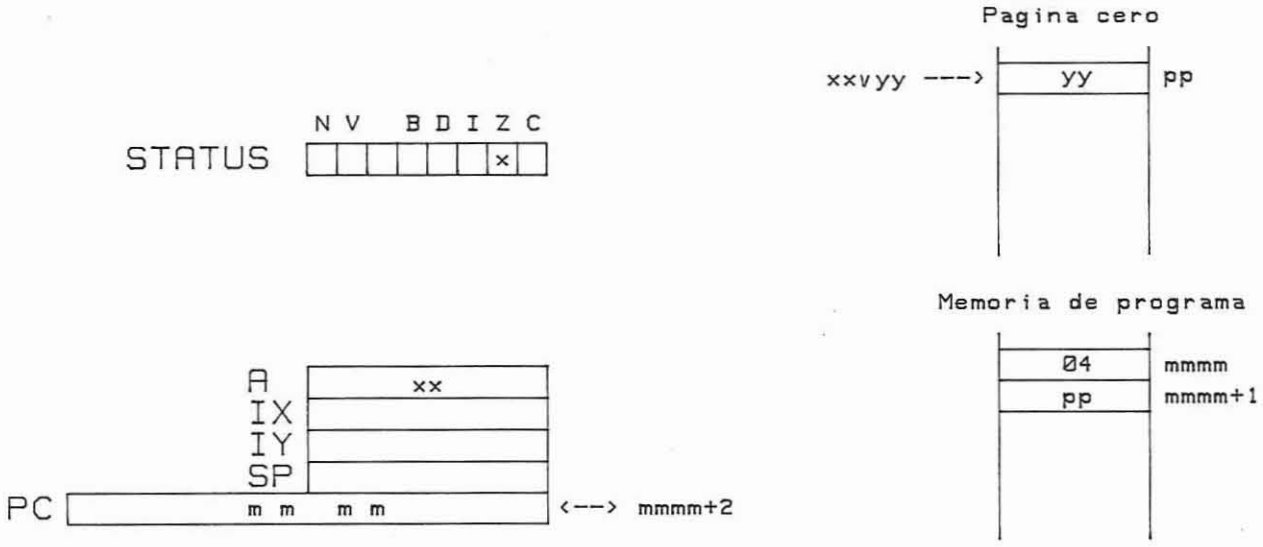
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

Las dos posiciones de memoria que siguen a la que contiene el código de operación de la instrucción contienen, en orden inverso (primero el byte bajo y luego el byte alto), la dirección de memoria donde se encuentra el dato. Todas las posiciones de este dato que coincidan con unos del Acumulador quedarán a "1" y todas las que coincidan con ceros del Acumulador quedarán a igual. El resultado queda en la misma posición de memoria.

TSB dir * COMPRUEBA Y PONE A 1 LOS BITS QUE SE CORRESPONDEN CON UNOS
(Direccionamiento página cero)



NUMERO DE CICLOS DE RELOJ:

5

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

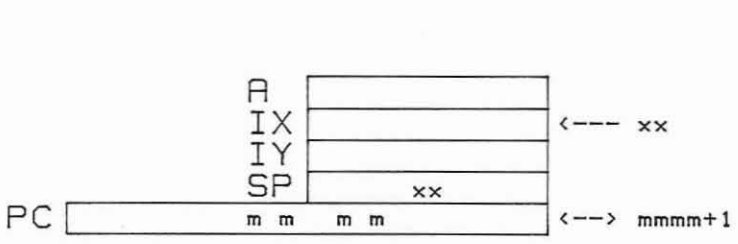
Z: Se pone a 1 si el resultado es cero.

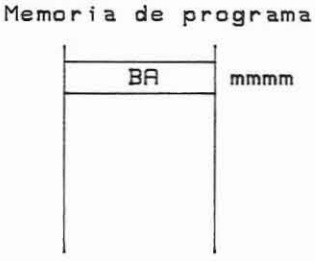
DESCRIPCION:

La posición de memoria que sigue a la que contiene el código de operación de la instrucción contiene la dirección de memoria de página cero donde se encuentra el dato. Todas las posiciones de este dato que coincidan con unos del Acumulador quedarán a "1" y todas las que coincidan con ceros del Acumulador quedarán a igual. El resultado queda en la misma posición de memoria.

TSX **TRANSFIERE EL CONTENIDO DEL 'STACK POINTER' AL REGISTRO X**
(Direccionamiento implícito)







NUMERO DE CICLOS DE RELOJ:

2

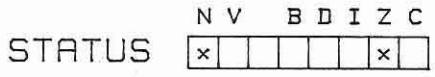
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

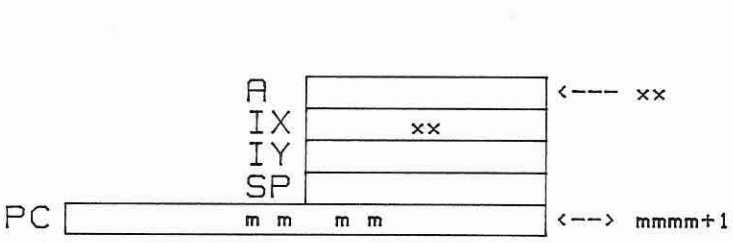
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

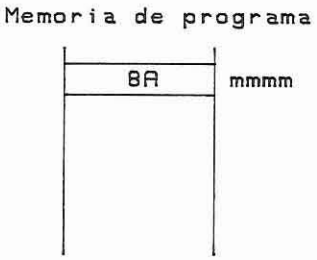
DESCRIPCION:

Cuando se ejecuta esta instrucción, el contenido del 'Stack Pointer' pasa al Registro X.

TXA **TRANSFIERE EL CONTENIDO REGISTRO X AL ACUMULADOR**
(Direccionamiento implícito)







NUMERO DE CICLOS DE RELOJ:

2

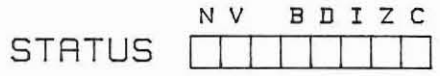
FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

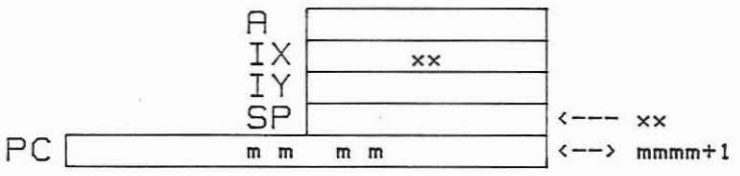
N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

Cuando se ejecuta esta instrucción, el contenido del Registro X pasa al Acumulador.

TXS **TRANSFIERE EL CONTENIDO REGISTRO X AL 'STACK POINTER'**
(Direccionamiento implícito)







NUMERO DE CICLOS DE RELOJ:

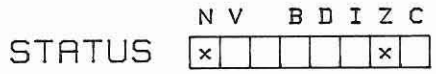
2

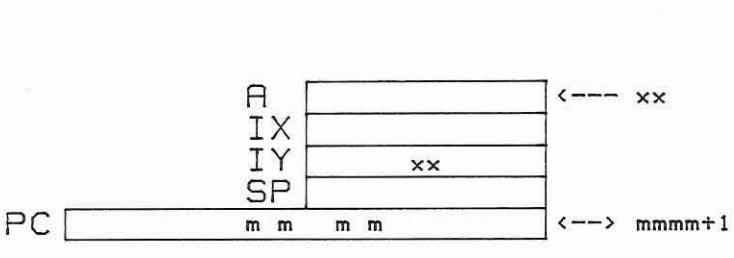
FLAGS DEL REGISTRO 'STATUS' AFECTADOS: ninguno

DESCRIPCION:

Cuando se ejecuta esta instrucción, el contenido del Registro X pasa al 'Stack Pointer'.

TYA **TRANSFIERE EL CONTENIDO REGISTRO Y AL ACUMULADOR**
(Direccionamiento implícito)







NUMERO DE CICLOS DE RELOJ:

2

FLAGS DEL REGISTRO 'STATUS' AFECTADOS:

N: Se pone a 1 si el resultado es negativo.
Z: Se pone a 1 si el resultado es cero.

DESCRIPCION:

Cuando se ejecuta esta instrucción, el contenido del Registro Y pasa al Acumulador.

APENDICE B

MODULOS DE APLICACION

Los módulos de la familia MM-6XX y MM-7XX están concebidos como una ampliación del Microinstructor MI-650 tanto para facilitar el aprendizaje y la aplicación de los microprocesadores como elementos de control, como para aumentar sus prestaciones.

Cada módulo consiste en un circuito de aplicación que funciona bajo el control del microprocesador. El usuario es quien debe interpretar las señales de control y las instrucciones para generar un programa adecuado a la función que desee realizar.

Para facilitar la labor de aprendizaje, cada módulo tiene una práctica de ejemplo ampliamente comentada en el manual que se proporciona junto con el módulo, y que se encuentra grabada en la memoria EPROM del Microinstructor en la dirección indicada en el Apéndice C.

Estos equipos han sido desarrollados para fines didácticos. Aunque no constituye su función principal, pueden ser utilizados en el desarrollo de pequeñas aplicaciones industriales. En cualquier caso, nunca se deberán rebasar las características especificadas, propias de este tipo de equipos.

Para facilitar la alimentación de los módulos de la familia MM-6XX y MM-7XX, I.E. PROMAX,SA ha diseñado la fuente de alimentación MM-600, capaz de proporcionar las tensiones y corrientes necesarias para todos éllos.

MM-601
SIMULADOR DE CRUCE CON SEMAFORO

El módulo MM-601 simula el cruce de dos calles con semáforos. Mediante un programa adecuado, podemos controlar el estado de los semáforos peatonales, el de los vehículos (con indicación de giro a la derecha), y el de precaución en el paso de peatones. Además disponemos de dos pulsadores para simular la presencia de vehículos en situación de espera, y otros dos para la petición de paso por parte de los peatones. El conjunto permite simular algoritmos de cruce de cualquier grado de complejidad y utilizar los pulsadores como entradas al Microinstructor que seleccionen varios modos de funcionamiento previamente programados: preferencia en la calle A, preferencia en la calle B, cruce en intermitencia ambar, etc

Este módulo está diseñado para ser controlado mediante las líneas de los Ports de 8 bits A y B del circuito integrado 65C22 que están disponibles en el Conector de Aplicación del Microinstructor (J2), actuando 4 de las líneas como entradas de los pulsadores y 12 como salidas a los LEDs.

El programa propuesto como aplicación de este módulo controla la secuencia normal de este tipo de cruces añadiendo además, un control del tiempo de duración de cada secuencia, que será menor si detecta espera de peatón o espera de coche mediante la pulsación de la tecla correspondiente.

MM-602
DOBLE CONVERSOR DIGITAL/ANALOGICO

El MM-602 consiste en dos conversores digital/analógicos controlados por los Ports de salida disponibles en el Conector de Aplicación (J2). Cada uno de estos conversores constituye un circuito funcionalmente independiente, de forma que es el programa de aplicación en el Microinstructor el que determina cual de ellos es activo y que tensión presentará su salida analógica correspondiente. El conjunto permite funciones de síntesis de señales de formas diversas, control por tensión, control en dos ejes tanto de trazadores analógicos como de osciloscopios, etc... El conversor utilizado,el DAC-08, es un conversor de 8 bits, rápido (85 nsg) y con un margen de tensión de salida de +10 a -10 voltios.

El programa que se presenta como ejemplo de aplicación genera a la salida de los conversores dos rampas entre -10 y +10 voltios defasadas 180 grados entre sí.

MM-603
CONVERSOR ANALOGICO/DIGITAL DE 8 CANALES

El MM-603 consiste en un conversor analógico/digital de 8 bits con 8 canales analógicos de entrada multiplexados. El sistema 'ADC' permite medir y monitorizar señales eléctricas o cualquier tipo de magnitudes físicas que, mediante los transductores apropiados, hayan sido convertidas previamente a niveles de tensión. Este módulo se controla mediante el bus de datos y el bus de direcciones del microprocesador y por tanto va conectado a través del Conector de Expansión del Microinstructor, de forma que queda configurado como una zona de memoria ROM dentro del mapa de memoria del MI-650C. El sistema de control selecciona cada uno de los 8 canales de entrada de forma sucesiva y direcciona la salida del conversor A/D a la posición de memoria correspondiente. Este proceso se realiza automáticamente y de forma independiente al funcionamiento del Microinstructor. El programa de control debe direccionar las posiciones de memoria que corresponden a cada uno de los canales de entrada y extraer el código digital correspondiente a las tensiones de entrada. La señal de reloj proporciona la base de tiempos que sincroniza el proceso de conversión y controla la selección de cada canal y la transferencia de datos.

El programa de ejemplo convierte al conjunto formado por el MI-650C y el MM-603 en un voltímetro digital de 8 canales de entrada.

MM-604
CONTROL DE MOTOR PASO A PASO

El módulo MM-604 es un circuito de control para un motor paso a paso de tipo bipolar. El sentido de giro y su velocidad son fácilmente programables a base de generar trenes de pulsos de frecuencia variable.

En el caso del módulo MM-604, las órdenes de control del motor se generan mediante un programa apropiado en el Microinstructor y pasan al circuito de control del módulo donde se decodifican y amplifican a la potencia necesaria para accionar el motor. El control de motor paso a paso está diseñado como un elemento de E/S, para ser controlado desde el Conector de Aplicación del Microinstructor. Existe la posibilidad de controlar simultaneamente dos motores con el mismo programa (movimiento en dos ejes). Se puede programar el motor de forma que al llegar un pulso de reloj, se efectue un paso completo o medio paso. Para el motor suministrado un paso completo

corresponde a 7.5 grados. Tambien dispone de una señal que se activa cada vez que el motor pasa por un punto de referencia. De esta forma permite una referencia de la posición del motor, contar el número de vueltas o medir la velocidad de rotación.

El programa propuesto como ejemplo de aplicación de este módulo controla el sentido de giro del motor paso a paso y lo mantiene a una velocidad constante. Esta velocidad se puede regular variando el tiempo de retardo que se introduce con la subrutina "DELAY".

MM-605
CONTROL DE MOTOR DC

El módulo MM-605 es un circuito de control de motores de continua que permite variar la velocidad de rotación mediante la regulación de la tensión aplicada en bornes de un pequeño motor de continua conectado a la placa. Esta tensión se selecciona digitalmente desde el Microinstructor de forma que se multiplican las posibilidades de control, dado que es posible la excitación del motor tanto de forma continua como pulsada. El sistema se complementa con un sensor óptico por reflexión que permite realizar una realimentación tacométrica para monitorizar la velocidad de rotación del motor.

El control del motor DC está diseñado como un elemento de E/S, para ser controlado a través de las 8 líneas del Port A del circuito 65C22 presentes en el Conector de Aplicación del Microinstructor. El control de velocidad se realiza por un conversor digital-analógico de 8 bits, siendo la velocidad máxima de rotación para el motor suministrado bajo el control de este circuito de 2.400 r.p.m.. La etapa amplificadora proporciona el nivel de tensión y corriente necesario para accionar el motor.

El programa presentado como ejemplo de aplicación de este módulo, calcula y visualiza la velocidad de giro del motor DC. Además permite incrementar o decrementar esta velocidad mediante la pulsación de las teclas [+] y [-] respectivamente.

MM-606
RELES DE ESTADO SOLIDO

El MM-606 es un placa de conmutación de cargas en alterna. Está formado por un conjunto de 8 relés de estado sólido controlables directamente con señales de nivel `TTL`. Este módulo está orientado a la conmutación de cargas básicamente resistivas (lámparas de incandescencia) de baja potencia. Su utilización con cargas de componente inductiva está condicionada a la conexión exterior de redes de protección de sobretensiones, de tipo `R-C` serie. La potencia de conmutación depende de las condiciones de disipación del circuito.

El MM-606 permite un gran número de aplicaciones de control de tipo práctico entre los que destacamos la generación de secuencias de luces programables, el control de válvulas y motores o la conmutación de elementos calefactores para el control de temperatura. Su utilización conjunta con el MM-603 (módulo conversor analógico-digital de 8 canales) permite cerrar un lazo de control con el mundo físico: controles de temperatura, nivel, intensidad de luz, humedad, ventilación, caudal,etc... Suponiendo que se ha conectado a la salida de cada relé una lámpara, el programa de aplicación que se plantea para este módulo, genera una secuencia luminosa igual sobre todas las líneas de salida del Port A que controlan a los relés y que están presentes en el Conector de Aplicación del Microinstructor. Esta secuencia luminosa enciende y apaga el conjunto de lámparas con una temporización creciente para cada nuevo período.

MM-607
PROGRAMADOR DE MEMORIAS EPROM 2716

Conectado al Microinstructor, el módulo MM-607, puede grabar, verificar y copiar en memoria RAM, cualquier memoria EPROM de los tipos 2716 y 2516 monotensión.

Estas memorias EPROM pueden utilizarse en el Microinstructor MI-650C en el zócalo AUX (seleccionando debidamente los selectores J5, J6 y J8). También puede utilizarse para grabar el programa de control de los módulos MM-610.

La conexión con el Microinstructor se realiza mediante dos Ports de 8 bits presentes en el Conector de Aplicación del Microinstructor.

El módulo dispone de un LED que indica el estado de grabación, y de un zócalo de grabación de fuerza de

extracción/inserción nula.

La tensión Vpp de grabación puede ser conmutada automáticamente mediante programa.

MM-608
INTERFACE RS-232C

El MM-608 es un módulo que permite recibir y transmitir información digital a través de un canal serie bidireccional del tipo RS-232C. Esta posibilidad amplía de forma considerable el campo de aplicación del Microinstructor, ya que, provisto de un interface serie, podemos conectarlo a la mayoría de impresoras, grabadores de memorias u otro tipo de dispositivos lógicos, ordenadores personales, sistemas de desarrollo, MODEMs y cualquier otro elemento informático que disponga de este tipo de interface.

El control del interface serie se realiza en su totalidad por programa, de forma que se puede adaptar fácilmente a cualquier posibilidad de velocidad de transmisión, número de bits de stop, control de paridad, etc... Se ha utilizado como base un circuito 6551 `ACIA` que realiza la conversión de los datos procedentes del microprocesador en paralelo y de forma asíncrona, a datos síncronos transmitidos en serie, y viceversa. La conexión del módulo al Microinstructor se realiza a través del Conector de Expansión.

El programa que se plantea como ejemplo de aplicación de este módulo, imprime, o visualiza en una pantalla, el contenido hexadecimal de todas las posiciones de memoria entre dos valores dados, dirección inicial y final. Para ello se requiere conectar la salida serie a una impresora o a una pantalla que dispongan de interface RS-232C

MM-609
AMPLIACION DE MEMORIA

Este módulo sólo es válido para funcionar bajo el control del Microinstructor MI-650 ó MI-650B.

MM-610
CONTROLADOR UNIVERSAL

El MM-610 está basado en un circuito que tiene todos los elementos básicos del Microinstructor, excepto el display y el teclado. Su función principal es la de controlar una determinada aplicación mediante un programa previamente desarrollado en el Microinstructor (u otro sistema de desarrollo cualquiera) y grabado en la memoria EPROM suministrada con el circuito.

Uno de los aspectos más interesantes de este módulo es la compatibilidad electrónica y funcional con el sistema Microinstructor (MI-650C y Módulos). El MM-610 genera las mismas señales de control y entrada/salida que el Microinstructor. La única diferencia está en que, a la hora de desarrollar los programas, debe tenerse en cuenta que los bloques de memoria del MI-650C son de 8 Kbytes mientras que los del módulo MM-610 son de 2 Kbytes. Por tanto, al desarrollar un programa en el MI-650C que luego se pretenda utilizar en el MM-610 se debe prestar atención a utilizar las zonas del mapa de memoria que coinciden en ambos.

La distribución de señales en los conectores de salida, es equivalente en ambos equipos. Esta característica permite desarrollar programas de control en el Microinstructor aprovechando todas las ayudas presentes en el programa monitor (visualización de memoria y registros, puntos de ruptura, ejecución paso a paso...). Cuando el programa se ejecuta correctamente en el Microinstructor, puede grabarse en una memoria y ejecutarse en el módulo MM-610, con la seguridad de un funcionamiento correcto. De esta forma se evita tener un Microinstructor ocupado en funciones de control permanentes y puede utilizarse para desarrollar nuevas aplicaciones.

MM-611
ENTRENADOR DE ROBOTICA

El Módulo MM-611 es un sistema que, conectado al Microinstructor (MI-650) lo convierte en un Entrenador de Robótica e implementa las funciones de control necesarias para la manipulación y posicionamiento del robot incluido en el equipo. Su campo de aplicación se extiende desde todos los niveles de enseñanza, hasta la investigación, pudiendo utilizarse en aspectos tan diversos como son el posicionamiento de pequeños objetos, manipulación en ambientes tóxicos, realización de ciclos de fatiga mecánica, estudio de algoritmos especiales de posicionamiento, desplazamiento por coordenadas, posicionamiento

automático en un punto de referencia, búsqueda y medida de objetos, etc... El robot es del tipo de Articulación Coordinada de 5 grados de libertad más el movimiento de la pinza que se controlan mediante 6 motores paso a paso capaces de actuar simultaneamente. El robot CS-113 posee un circuito inteligente basado en un microprocesador Z80 que realiza las funciones de interface con un microprocesador exterior, test funcional y control de los motores. La interconexión entre el robot y el Microinstructor se realiza mediante un interface paralelo "Centronics", que está implementado en el módulo MM-612 y se proporciona junto con el Robot y los demás accesorios. Además del robot y su interconexión con el Microinstructor, se proporciona con este módulo un programa de control (ROBOT2), que convierte al Microinstructor en un programador manual (teaching box) del robot, capaz de aprender una trayectoria y controlar su repetición.

MM-612
INTERFACE CENTRONICS

El módulo MM-612 es un interface paralelo del tipo "Centronics" que, además de adaptar las señales del Microinstructor a este tipo de bus para gobernar equipos como por ejemplo el robot CS-113, incluye un transductor acústico, dos relés 'reed' para control de elementos exteriores y una salida digital de nivel TTL. Estas características permiten el desarrollo de funciones auxiliares.

Este módulo utiliza las 16 líneas de entrada/salida del Conector de Aplicación del Microinstructor. El control del canal y de las líneas que regulan la transferencia de datos se realiza totalmente por software, por lo que se proporciona para su utilización una subrutina que activa e interpreta las líneas de control y sitúa los datos en en el bus en la secuencia adecuada.

MM-613
IMPRESORA

El módulo MM-613 es una impresora diseñada para ser conectada directamente al Conector de Expansión del Microinstructor y que se puede utilizar con sólo cambiar el conmutador J5 a la posición EXT. No obstante puede ser utilizada desde cualquier equipo que proporcione las señales de control adecuadas.

Además de esta función este módulo dispone de 32 Kbytes de memoria RAM como expansión de la memoria del Microinstructor (2000H - 9FFFH). También dispone de un canal RS232 para la comunicación serie que permite la conexión directa del Microinstructor con un terminal para la salida de datos a pantalla, la entrada de datos para su impresión, etc.

Para el control del mecanismo de impresión, del canal serie y de la interconexión con el Microinstructor este módulo también incluye una memoria EPROM de 8 Kbytes. Esta memoria se suministra con un programa que controla la impresión, entre dos posiciones de memoria previamente indicadas, de las direcciones y los datos que éstas contienen. La impresión de estos datos puede hacerse en ASCII o en hexadecimal.

Además también dispone de la posibilidad de imprimir los programas en nemónico mediante un desensamblado de los datos.

MM-614
EXPANSION PARA APLICACIONES DE HARDWARE

El módulo MM-614 no es un circuito de aplicación propiamente dicho sino un 'interface' que pone a disposición del usuario todas las señales del Conector de Expansión. Esta función de 'interface' la llevan a cabo los buffers unidireccionales 74HCT244 en el caso del bus de direcciones, bus de control y líneas de alimentación, y el buffer bidireccional 74HCT245 en el bus de datos. Dispone de unas regletas para inserción de cables que hacen muy rápida y cómoda la función de interconexión entre el módulo y las aplicaciones externas. Esto permite la fácil conexión de sus propios montajes prácticos al Microinstructor para ser controlados por el microprocesador 65C02. Además de todas las líneas de control, el bus de direcciones y el bus de datos, el módulo MM-614 dispone de una VIA (65C22) que permite la utilización de dos canales más de 8 bit cada uno, programables en cada una de sus líneas como entradas o como salidas.

MM-615
MODEM

El módulo MM-615 es un circuito modulador-demodulador para la transmisión de información digital a través de una línea telefónica, comunmente denominado MODEM. Su conexión con el Microinstructor se realiza a través del interface RS-232C. Aunque el MODEM que utilizamos dispone de las señales necesarias para efectuar tanto la llamada como la respuesta automáticas, el módulo MM-615 está implementado para que ambas deban efectuarse manualmente, de forma que incorpora dos pulsadores para simular la detección de llamada en el receptor y la orden de inicio de transmisión en el transmisor. Este funcionamiento es el más adecuado para los fines didácticos del equipo ya que el alumno se familiarizará mas rápidamente con el funcionamiento real del sistema. También por este motivo se incorporan una serie de "LEDs" para visualizar el estado de las líneas del canal principal en todo momento, y un zumbador para el reconocimiento acústico de la señal de entrada.

Como ejemplo de aplicación de este módulo se han propuesto dos programas, uno para la transmisión y otro para la recepción, que se ajustan al funcionamiento antes indicado.

MM-616
MODULO DE AUDIO

El módulo de audio es un generador de señal sinusoidal, triangular y cuadrada, que permite la programación de la amplitud y de la frecuencia de una de ellas. Además de estas 3 formas de onda puede seleccionar, tambien de forma programable, la entrada de una señal de audio Auxiliar.

Las 2 primeras etapas consisten en un conversor digital/analógico y un conversor tensión/frecuencia, de forma que, a través de las líneas del "port A" del Conector de Aplicación del Microinstructor se puede programar la frecuencia de la señal.

Para permitir el tratamiento espectral de la señal se incluye como siguiente etapa un filtro digital paso-alto y un filtro digital paso-bajo de frecuencias de corte programables entre cuatro valores cada uno de ellos. Esta frecuencia se puede seleccionar a través de los cuatro bits de menor peso del "Port A" del Microinstructor teniendo previamente seleccionadas las dos líneas de menor peso del "Port B" en el modo "Selección de la frecuencia de corte del filtro paso-alto" o "Selección de la

frecuencia de corte del filtro paso-bajo".

La cuarta etapa es la de amplificación/atenuación, programable a través de los cuatro dígitos de mayor peso del "port B " del Microinstructor.

La última es una etapa de protección que además proporciona una baja impedancia de salida.

MM-617
SIMULADOR DE ASCENSORES

El módulo MM-617 es un equipo didáctico destinado a simular el funcionamiento del sistema de control de un ascensor, tanto desde el punto de vista del control ejercido desde los mandos internos de la cabina y de los distintos pisos, como del mecanismo principal de control que detecta la proximidad a uno de los pisos destino y altera la velocidad del ascensor en la fase de frenado o de aceleración.

El control de este equipo se realiza a través de los dos Ports paralelo del Microinstructor, que pueden programarse, línea a línea de forma independiente, como entradas o como salidas. Por este procedimiento pueden implementarse los distintos elementos de control mediante simulación de "0" y "1" en cada una de las líneas de entradas o salidas y que corresponden funcionalmente a los elementos mecánicos como pulsadores o interruptores de proximidad, etc...

Paralelamente se presenta un programa de control que puede servir como introducción de otros ejemplos más complejos de control de este equipo.

MM-7XX

Los módulos de la familia MM-7XX están concebidos como una ampliación del Microinstructor para facilitar el aprendizaje de las técnicas de medida de magnitudes físicas en procesos de control industrial.

Cada módulo consiste en un circuito de aplicación formado por un transductor de medida de la magnitud física y por un acondicionador de señal que adecúa el nivel de la señal eléctrica entregada por el transductor.

Los módulos van interconectados entre sí a través de la linea de alimentación y poseen una salida de tensión proporcional a la magnitud física que están midiendo. Esta salida se puede conectar a alguno de los canales del módulo convertidor analógico-digital (MM-603). De esta forma, mediante el programa propuesto como ejemplo de aplicación de este módulo que mide el nivel de tensión de cada entrada, se puede obtener una medida directa o proporcional de la magnitud física medida entre los márgenes en los que el transductor se comporta de forma lineal.

a. MM-701 Sensor de fuerza

El MM-701 es un módulo de demostración de medida de fuerzas. Consiste en un transductor de fuerza (célula de carga) y un amplificador acondicionador de señal que adapta los niveles de tensión dados por la célula a los de entrada del convertidor analogico digital.

La celula de carga es del tipo extensométrico. La fuerza que se pretende medir produce una deformación en el transductor y esta deformación es transformada en señal eléctrica por medio de galgas extensiométricas.

b. MM-702 Sensor de temperatura

El MM-702 es un módulo medidor de temperatura. Ha sido diseñado con el objetivo de ayudar a la enseñanza de los sistemas de control de procesos térmicos. Existe una importante relación entre el estado de un proceso y su temperatura. Es por este motivo por lo que el control de este tipo de procesos es cada día más frecuente y por lo tanto más importante.

El módulo consta de una sonda sensora y de un acondicionador de señal. El transductor consiste en una resistencia de material semiconductor que varía su conductividad con la temperatura. El acondionador de señal se

encarga de convertir estas variaciones de resistencia en variaciones de tensión de forma que, para trabajar con la máxima resolución del convertidor analógico digital, un margen de temperatura de 0 a 100°C quede convertido en una variación de la tensión de salida de 0 a 10 voltios.

c. **MM-703 Sensor de inducción magnética**

El MM-703 es un módulo sensor de inducción magnética (B). Consta de una sonda de efecto Hall y de un acondicionador de señal. Es posible medir el valor de la inducción magnética en margen de 0 a 100 mT. El módulo posee ajustes de cero y de fondo de escala y está calibrado de forma que proporciona una salida de 10 Voltios para el valor máximo que cubre la sonda. De esta forma la medida se realiza siempre con la máxima resolución.

Es posible medir campos magnéticos alternos y detectar la polaridad aunque para ello es necesario ajustar el cero en el centro de la escala y detectar la polaridad por sofware según la tensión sea mayor o menor de 5 Voltios.

d. **MM-704 Sensor de iluminación**

El MM-704 es un módulo sensor de intensidad luminosa. Consta de un elemento fotoresistivo y de un acondicionador de señal que convierte las variaciones de resistencia en variaciones de tensión. El módulo posee ajustes de cero y de fondo de escala, de forma que es posible ajustar la sensibilidad en el margen deseado.

e. **MM-705 Sensor de presión**

El MM-705 es un módulo sensor de presión que puede ser usado para medidas de presión absoluta hasta 2 bares. Está formado por un transductor de presión y un acondicionador de señal. El transductor contiene en su interior una membrana sobre la que hay colocadas cuatro galgas semiconductoras en configuración de puente completo. Esta membrana se deforma debido a la diferencia de presiones entre el ambiente y una cámara de vacío que hay en su interior. El acondicionador de señal transforma los cambios de resistencia de las galgas en variaciones de tensión que pueden ser interpretadas por el Microinstructor a través del convertidor analógico-digital.



APENDICE C

DIRECCIONES DE LOS PROGRAMAS

Las direcciones de la memoria EPROM donde se encuentran los programas propuestos como ejemplos de los Módulos de Aplicación, las prácticas del capítulo 7 y el programa monitor del Microinstructor son las siguientes:

DIRECCION INICIAL	DIRECCION FINAL	PROGRAMA
E001H	E0B7H	MM-601
E0B8H	E0D0H	MM-602
E0D1H	E181H	MM-603
E182H	E1CBH	MM-604
E1CCH	E4ACH	MM-605
E4ADH	E4DBH	MM-606
E4DCH	E6CAH	MM-607
E6CBH	E7D3H	MM-608
E7D4H	E8F3H	MM-615 (RMODEM)
E8F4H	EA2AH	MM-615 (TMODEM)
EA2BH	ED1BH	MM-616
ED1CH	F025H	MM-617
F026H	F0FFH	Libre
F100H	F186H	Modificaciones Monitor
F187H	F333H	Libre

DIRECCION INICIAL	DIRECCION FINAL	PROGRAMA
F334H	F3C9H	Recolocador
F3CAH	F418H	Secuenciador
F419H	F445H	Analizador Lógico
F446H	F454H	Resta doble precisión
F455H	F470H	Raiz cuadrada
F471H	F522H	Conversor Hex.-Dec.
F523H	F5AFH	Conversor Dec.-Hex.
F5B0H	F637H	Cronómetro
F638H	F6E5H	Reloj
F6E6H	F76FH	Juego
F770H	F7CFH	Movimiento de bloques
F7D0H	FFFFH	MONITOR

APENDICE D

LISTADO

El listado del programa monitor del Microinstructor que seguidamente se presenta, consta inicialmente de una zona de correcciones (como ya se ha visto en el apéndice C). El programa principal, se encuentra a partir de la dirección F7D0H.

Debe recordarse que el ensamblado esta hecho con un sistema de desarrollo que invierte los bytes de las direcciones absolutas, de modo que el orden en que aparecen en el listado (primero byte alto y luego byte bajo) no se corresponde con el orden en que están grabados en la memoria (primero byte bajo y luego byte alto).

La presentación del programa es la siguiente:

[DIRECCION] [CODIGO] [LINEA] [NEMONICO] [OPERANDO] [COMENTARIO]

```
POSICION CODIGO      LINEA   CODIGO FUENTE

                       1 "650X"
                       2       ;DEFINICION DE DIRECCIONES
                       3
                       4       ;GENERALES
                       5       ;RAM: 0000H - 03FFH (0400H - 1FFFH)
                       6       ;ROM: 0F700H - 0FFFFH (0E000H - 0F7CFH)
                       7       ;I/O 0A000H - 0A00FH
                       8       ;VIA 65C22 Auxiliar: 0A400H - 0A7FFH
                       9       ;VIA 65C22 Local
                      10
< A001 >              11 RAIORA       EQU     0A001H
< A000 >              12 RAIORB       EQU     0A000H
< A003 >              13 RAIODRA      EQU     0A003H
< A002 >              14 RAIODRB      EQU     0A002H
< A004 >              15 RAIOTLC      EQU     0A004H
< A005 >              16 RAIOTHC      EQU     0A005H  ;TIMER'S
< A00E >              17 RAIOIER      EQU     0A00EH
< A00D >              18 RAIOIFR      EQU     0A00DH  ;INTERRUPT FLAG REGISTER
< A00B >              19 RAIOACR      EQU     0A00BH
< A00C >              20 RAIOPCR      EQU     0A00CH  ;PERIPHERAL CONTROL REGISTER
                      21
                      22       ;6522 Auxiliar
                      23
< A400 >              24 PB2          EQU     0A400H  ;PORT B
< A402 >              25 CPB2         EQU     0A402H  ;CONTROL DEL PORT B
< A401 >              26 PA2          EQU     0A401H  ;PORT A
< A403 >              27 CPA2         EQU     0A403H  ;CONTROL DEL PORT A
                      28
                      29       ;POSICIONES DE PG 0 UTILIZADAS EN RESET
                      30       ;PERO TRANSPARENTES AL USUARIO.
                      31
< 0002 >              32 BASE         EQU     02H     ;ADR. DEL VALOR DE RAMTEST.
< 0000 >              33 TRAML        EQU     00H     ;ADR. BAJA DEL PUNTERO EN RAM.
< 0001 >              34 TRAMH        EQU     01H     ;ADR. ALTA PUNTERO EN RAM.
                      35
                      36               .
                      37
                      38       ;RAM DE TRABAJO (DE 0352H A 03FFH)
                      39
                      40               ORG     0352H
                      41
0352                  42 REST         RMB     1       ;FLAG DE RESET
0353                  43 DISBUF       RMB     6       ;BUFFER DE DISPLAY
0359                  44 PARL         RMB     1
035A                  45 PARH         RMB     1
035B                  46 SCR1         RMB     1       ;VARIOS
035C                  47 SCR2         RMB     1
035D                  48 SCR3         RMB     1
035E                  49 ID           RMB     1       ;IDENTIFIC. DE CASSETTE
035F                  50 MODEC        RMB     1       ;MODO DE TRABAJO
0360                  51 PRIMER       RMB     1       ;FLAG
0361                  52 PUNTO        RMB     1       ;PUNTOS DE ZONA DE TRAB.
0362                  53 OLD          RMB     1       ;USO EN CASSETTE
0363                  54 LODIVER      RMB     1       ;CAS LOAD /VERIFICA
0364                  55 ULTEC        RMB     1       ;ULTIMA TECLA PULS.(FLAG)
0365                  56 BRKFLG       RMB     1       ;HAY/NO BREAK
0366                  57 BRKDAT       RMB     1       ;DATO EN MEM.SUST.POR BRK
```

```
POSICION CODIGO     LINEA   CODIGO FUENTE

    0367             58 BRKADL       RMB     1       ;DIRECCION DEL BREAK
    0368             59 BRKADH       RMB     1
    0369             60 PCLR         RMB     1       ;REGISTROS IMAGEN
    036A             61 PCHR         RMB     1
    036B             62 SR           RMB     1
    036C             63 AR           RMB     1
    036D             64 XR           RMB     1
    036E             65 YR           RMB     1
    036F             66 PR           RMB     1
    0370             67 TAPDEL       RMB     1       ;PARAMETROS DEL CASSETT
    0371             68 HSBDRY       RMB     1
    0372             69 TAPETI       RMB     1
    0373             70 TAPET2       RMB     1
    0374             71 ESPVEC       RMB     2       ;VECTOR DE MANDOS ESPEC.
    0376             72 BRKUS        RMB     2       ;VECTOR BRK DE USUARIO
    0378             73 IRQUS        RMB     2       ;VECTOR IRQ DE USUARIO
    037A             74 NMIUS        RMB     2       ;VECTOR NMI DE USUARIO
    037C             75 IRQVEC       RMB     2       ;VECTOR IRQ SECUNDARIO
    037E             76 NMIVEC       RMB     2       ;VECTOR NMI SECUNDARIO
    0380             77 CONTA        RMB     1       ;COTADOR DE CASSETTE
    0381             78 TEMPO        RMB     1       ;TEMPORAL DE CASSETTE
                     79
                     80         ;PAGINA 0
                     81
           < 00FD >. 82 CHAR         EQU     0FDH    ;TEMPORAL DE CARACTER
           < 00FE >  83 CURAD        EQU     0FEH    ;PUNTERO USO GENERAL
                     84
                     85 ;*******************************************************
                     86 ;
                     87 ;     MARCA PARA RECONOCER SI SE TRATA DEL MI-650C
                     88 ;     O DE VERSIONES ANTERIORES
                     89 ;
                     90                ORG     0E000H  ;POSICIO DE LA MARCA
    E000 AA          91 MARCA          BYTE    0AAH
                     92 ;
                     93 ;*******************************************************
                     94 ;*******************************************************
                     95 ;
                     96 ;     ZONA DE PATCH PARA EL MONITOR DEL MI-650C
                     97 ;     DEFINIDO PARA MANTENER CONSTANTES LAS DIRECCIONES
                     98 ;     DE LAS SUBRUTINAS DE MEMORIA UTILIZADAS EN
                     99 ;     LOS OTROS MODELOS DE MI-650.
                    100 ;
                    101               ORG     0F100H
                    102 ;
                    103 ;     ADAPTACION DEL SISTEMA DE LECTURA DE DATOS
                    104 ;     DEL CASSETTE DEBIDO AL CAMBIO 6532 POR 65C22.
                    105 ;     BASICAMENTE SUSTITUYE EL DIVISOR DE 8 BITS
                    106 ;     DEL CIRCUITO INICIAL POR UNO DE 16 BITS EN EL
                    107 ;     65C22.
                    108 ;
    F100            109 M_PATCH1
    F100 A903       110       LDA     #03H    ;DIV PER 8 PER TRIPLE SHIFT
    F102 8D0380     111       STA     CONTA   ;DEL CONTINGUT COMPTADOR 16 BITS
    F105 ADA005     112       LDA     RA1OTHC
    F108 2907       113       AND     #07H
    F10A 8D0381     114       STA     TEMPO
```

```
POSICION CODIGO      LINEA    CODIGO FUENTE

   F10D ADA004       115      LDA     RAIOTLC
   F110              116 LOOPA
   F110 18           117      CLC
   F111 4E0381       118      LSR     TEMPO   ;3 x SHIFT A TRAVES DEL 'CY'
   F114 6A           119      ROR     A
   F115 CE0380       120      DEC     CONTA
   F118 D0F6         121      BNE     LOOPA
   F11A 8CA004       122      STY     RAIOTLC ;CARREGAR BYTE LOW DEL TIMER 1
   F11D A007         123      LDY     #07H    ;BYTE ALT TIMER 1
   F11F 8CA005       124      STY     RAIOTHC ;DISPARAR EL TIMER 1
   F122 18           125      CLC
   F123 6D0371       126      ADC     HSBDRY
   F126 60           127      RTS
                     128 ;*****************************************************
                     129 ;
                     130 ;     INICIALIZA EL 65C22 DE I/O AUXILIAR
                     131 ;
   F127              132 M_PATCH2
                     133
   F127 E8           134      INX
   F128 8EA003       135      STX     RAIODRA
   F12B A90F         136      LDA     #00001111B
   F12D 8DA000       137      STA     RAIORB
   F130 A93F         138      LDA     #00111111B
   F132 8DA002       139      STA     RAIODRB
   F135 A900         140      LDA     #00H
   F137 8DA00B       141      STA     RAIOACR
   F13A 8DA00C       142      STA     RAIOPCR
   F13D A97F         143      LDA     #7FH
   F13F 8DA00E       144      STA     RAIOIER
   F142 8DA00D       145      STA     RAIOIFR
   F145 60           146      RTS
                     147 ;
                     148 ;*****************************************************
                     149 ;
                     150 ;     NUEVO MENSAJE DE SALIDA Y CONTROL LED's
                     151 ;
   F146              152 M_PATCH3
   F146 A9FF         153      LDA     #0FFH   ;PROG. PORT B DE SALIDA
   F148 8DA402       154      STA     >CPB2
   F14B A900         155      LDA     #00H
   F14D 8DA403       156      STA     >CPA2   ;PORT A DE ENTRADA
   F150 8D0353       157      STA     DISBUF
   F153 A97D         158      LDA     #7DH    ;6
   F155 8D0354       159      STA     DISBUF+1
   F158 A96D         160      LDA     #6DH    ;5
   F15A 8D0355       161      STA     DISBUF+2
   F15D A93F         162      LDA     #3FH    ;0
   F15F 8D0356       163      STA     DISBUF+3
   F162 A939         164      LDA     #39H    ;C
   F164 8D0357       165      STA     DISBUF+4
   F167 A900         166      LDA     #00H    ;
   F169 8D0358       167      STA     DISBUF+5
   F16C 38           168      SEC
   F16D A900         169      LDA     #0      ;ENCIENDE LED's
   F16F              170 LED
   F16F 2A           171      ROL     A
```

```
POSICION CODIGO      LINEA    CODIGO FUENTE

   F170 8DA400        172       STA     >PB2
   F173 48            173       PHA
   F174 08            174       PHP
   F175 A010          175       LDY     #10H    ;TEMPORIZACION
   F177 20FD76        176       JSR     ESPER+2
   F17A 28            177       PLP
   F17B 68            178       PLA
   F17C 90F1          179       BCC     LED
   F17E ADA401        180       LDA     >PA2    ;TEST DE MICROCONMUTADORES
   F181 49FF          181       EOR     #0FFH   ;INVERTIR (SW-ON == LED-ON)
   F183 8DA400        182       STA     >PB2
   F186 60            183       RTS
                      184 ;*****************************************************
                      185 ;*****************************************************
                      186 ;
                      187 ;     ZONA DE ROM RESERVADA PARA LOS PROGRAMAS DE
                      188 ;     PRACTICAS DEL CAPITULO 6 DEL LIBRO.
                      189 ;
                      190 ;     ORG     0F3CAH
                      191 ;
                      192 ;*****************************************************
                      193 ;*****************************************************
                      194 ;
                      195 ;     ZONA DE MEMORIA DEL PROGRAMA MONITOR
                      196 ;
                      197
                      198       ORG 0F7D0H
                      199
                      200             ;BUCLE DE REPOSO
                      201             ;SITUACION DE ESPERA DE INSTRUCCIONES
                      202             ;ENTRADA NORMAL
                      203             ;SACA LINEAS EN DISPLAY Y ESPERA
                      204
                      205 ;
   F7D0               206 MONITR
   F7D0 18            207       CLC
   F7D1 9001          208       BCC START
                      209 ;
                      210             ;ENTRADA DE ERROR
                      211             ;SACA TEXTO "Err" Y CODIGO EN ACC
   F7D3               212 ERROR
   F7D3 38            213       SEC
                      214                     ;ENTRADA COMUN
   F7D4 AE036B        215 START LDX     SR      ;NO ALTERES EN TRACE
   F7D7 9A            216       TXS           ;INICIALIZA STACK
   F7D8 D8            217       CLD           ;NO DECIMAL
   F7D9 B02D          218         BCS   STRT2
   F7DB               219 NORMAL
   F7DB AE0352        220       LDX     REST
   F7DE F01C          221       BEQ     NORM1   ;NO
   F7E0               222 SALUDO
   F7E0 20FD48        223       JSR     CLRDIS  ;BORRA DISPLAY
   F7E3 20F146        224       JSR     M_PATCH3        ;NOU MISSATGE
   F7E6 CE0352        225       DEC     REST
   F7E9 4CF7DB        226       JMP     NORMAL  ;TREURE GUIONS
                      227 ;
                      228       ORG     0F7FCH  ;NO VARIAR ADR.
```

```
POSICION CODIGO      LINEA    CODIGO FUENTE

                     229 ;
F7FC                 230 NORM1
F7FC A940            231       LDA     #40H    ;CODIGO DE -
F7FE A206            232       LDX     #6H
F800 9D0352          233 STRT1   STA   DISBUF-1,X
F803 CA              234         DEX
F804 D0FA            235         BNE   STRT1
F806 F015            236       BEQ     STRT3   ;A BUCLE
                     237
F808                 238 STRT2
F808 48              239       PHA             ;SALVA CODIGO
F809 20FD48          240         JSR   CLRDIS  ;BORRA DISPLAY
F80C A979            241         LDA   #79H    ;E
F80E 8D0353          242       STA     DISBUF
F811 A950            243       LDA     #50H    ;r
F813 8D0354          244       STA     DISBUF+1
F816 8D0355          245       STA     DISBUF+2
F819 68              246       PLA
F81A                 247 STRT21
F81A 20FD32          248       JSR     OUTDAT  ;CODIGO DE ERROR
                     249
                     250               ;BUCLE DE ESPERA EN REPOSO
                     251
F81D                 252 STRT3
F81D A900            253       LDA     #0
F81F 8D0361          254       STA     PUNTO   ;BORRA PUNTOS
F822 20FE09          255       JSR     GETKEY  ;OBTEN TECLA
F825 90F6            256       BCC     STRT3   ;NO HAY ESPERA
F827 C912            257       CMP     #12H    ;BORRA ERROR CON 'FIN'
F829 F0A5            258       BEQ     MONITR
F82B C90E            259       CMP     #0EH    ;VALIDA?
F82D 9006            260       BCC     STR3    ;SI
F82F 20F83B          261       JSR     ESPEC   ;NO, SALTO VECTORIZADO
F832 4CF7D4          262       JMP     START   ;Y RETORNO
                     263
                     264              .;EJECUCION DE INSTRUCCIONES EN AMPLIACION
                     265
F835                 266 STR3
F835 20F83E          267       JSR     EJEC
F838 4CF7D4          268       JMP     START   ;Y RETORNO
                     269
                     270               ;SALTO INDIRECTO DE EJECUCION DE AMPLIACIONES
                     271
F83B                 272 ESPEC
F83B 6C0374          273       JMP     (ESPVEC)
                     274               ;EJECUCION DE FUNCIONES
                     275
F83E                 276 EJEC
                     277
F83E 20FD48          278       JSR     CLRDIS  ;BORRA DISPLAY
F841 F003            279       BEQ     EJEC0
F843 4CF8C8          280       JMP     EJEC1
                     281
                     282               ;INSPECCIONA/ALTERA MEMORIA
                     283
F846                 284 EJEC0
F846 A906            285       LDA     #06H
```

```
POSICION CODIGO     LINEA    CODIGO FUENTE

    F848 20FD58      286      JSR    INDIC   ;SACA INDICADOR 'ad'
    F84B A5FE        287      LDA    CURAD   ;PASA ULTIMA DIRECCION
    F84D 8D0359      288      STA    PARL    ;USADA EN REFERENCIAS
    F850 A4FF        289      LDY    CURAD+1 ;A MEMORIA
    F852 8C035A      290      STY    PARH
    F855 20FD22      291      JSR    OUTADD  ;SACA A DISPLAY
    F858 20FD90      292      JSR    GETADD  ;OBTEN DIRECCION
    F85B AD0359      293      LDA    PARL
    F85E 85FE        294      STA    CURAD   ;TOMA DIRECCION
    F860 AD035A      295      LDA    PARH
    F863 85FF        296      STA    CURAD+1
    F865 A903        297      LDA    #000011B
    F867 8D0361      298      STA    PUNTO   ;FIJA PUNTOS
                     299
    F86A             300 M1
                     301
    F86A A000        302      LDY    #00H
    F86C B1FE        303      LDA    [CURAD],Y
    F86E 8C0360      304      STY    PRIMER
    F871 8D0359      305      STA    PARL    ;SALVA VALOR
    F874 20FD32      306 M1A  JSR    OUTDAT  ;SACA EL DATO
    F877 20FE09      307 M1B  JSR    GETKEY  ;TOMA TECLA
    F87A B007        308      BCS    M1C     ;HAY, PROCESA
    F87C AC0360      309      LDY    PRIMER  ;INTRODUCIDO DATO?
    F87F F0E9        310      BEQ    M1      ;NO SACA DATO PRES. EN CURAD
    F881 D0F4        311      BNE    M1B     ;SI SACA DATO INTROCUCIDO
    F883             312 M1C
    F883 C912        313      CMP    #12H    ;FINAL?
    F885 D002        314      BNE    M2
    F887 18          315      CLC             ;RETORNO
    F888 60          316      RTS
    F889             317 M2
    F889 B0EC        318      BCS    M1B     ;IGNORA OTRAS
    F88B C910        319      CMP    #10H    ;HEX?
    F88D 9033        320      BCC    MHEX    ;SI
    F88F 08          321      PHP             ;SERA '+' O '-' SALVA
    F890 AD0360      322      LDA    PRIMER
    F893 F00F        323      BEQ    M3      ;IGNORA SI NO HAY DATO NUEVO
    F895 AD0359      324      LDA    PARL
    F898 A000        325      LDY    #00H
    F89A 91FE        326      STA    [CURAD],Y       ;SALVA EN MEMORIA
    F89C D1FE        327      CMP    [CURAD],Y       ;HECHO?
    F89E F004        328      BEQ    M3      ;SI
    F8A0 68          329      PLA             ;NO AJUSTA STACK
    F8A1 4CF86A      330      JMP    M1      ;Y REPITE
    F8A4             331 M3
    F8A4 28          332      PLP             ;ALTERA DIRECCION
    F8A5 D009        333      BNE    M4
    F8A7 E6FE        334      INC    CURAD   ;AUMENTA
    F8A9 D00D        335      BNE    M6
    F8AB E6FF        336      INC    CURAD+1
    F8AD 4CF8B8      337      JMP    M6
    F8B0             338 M4
    F8B0 A5FE        339      LDA    CURAD   ;DISMINUYE
    F8B2 D002        340      BNE    M5
    F8B4 C6FF        341      DEC    CURAD+1
    F8B6             342 M5
```

```
POSICION CODIGO     LINEA   CODIGO FUENTE

   F8B6 C6FE        343      DEC    CURAD
   F8B8             344 M6
   F8B8 A5FE        345      LDA    CURAD
   F8BA A4FF        346      LDY    CURAD+1
   F8BC 20FD22      347      JSR    OUTADD  ;SACA DIRECCION
   F8BF 4CF86A      348      JMP    M1      ;Y REPITER
   F8C2             349 MHEX
   F8C2 20FDE6      350      JSR    PACK    ;EMPACA
   F8C5 4CF874      351      JMP    M1A     ;Y REPITE
   F8C8             352 EJEC1
   F8C8 C901        353      CMP    #01H
   F8CA D04E        354      BNE    EJEC2
                    355
                    356             ;SALIDA A PROGRAMA DE USUSARIO
                    357
   F8CC A907        358      LDA    #7H
   F8CE 20FD58      359      JSR    INDIC   ;INDICADOR 'Go'
   F8D1 AD0369      360      LDA    PCLR    ;PASA DEFAULT
   F8D4 8D0359      361      STA    PARL
   F8D7 AC036A      362      LDY    PCHR
   F8DA 8C035A      363      STY    PARH
   F8DD 20FD22      364      JSR    OUTADD
   F8E0 20FD90      365      JSR    GETADD
   F8E3 AD0360      366      LDA    PRIMER  ;HAY PARAMETRO?
   F8E6 F018        367      BEQ    GO      ;NO, USA ANTIGUO
   F8E8 A2FF        368      LDX    #0FFH
   F8EA 9A          369      TXS
   F8EB A9F7        370      LDA    #(MONITR-1)/256
   F8ED 48          371      PHA
   F8EE A9CF        372      LDA    #(MONITR-1).AN.00FFH
   F8F0 48          373      PHA
   F8F1 AD0359      374      LDA    PARL    ;TOMA DIRECC. DE ACCION
   F8F4 8D0369      375      STA    PCLR
   F8F7 AD035A      376      LDA    PARH
   F8FA 8D036A      377      STA    PCHR
   F8FD 4CF904      378      JMP    .G01
   F900 AE036B      379 GO   LDX    SR      ;PASA REGISTRO DE USUARIO
   F903 9A          380      TXS
   F904             381 G01
   F904 AD036A      382      LDA    PCHR
   F907 48          383      PHA
   F908 AD0369      384      LDA    PCLR
   F90B 48          385      PHA
   F90C AD036F      386      LDA    PR
   F90F 48          387      PHA
   F910 AD036C      388      LDA    AR
   F913 AE036D      389      LDX    XR
   F916 AC036E      390      LDY    YR
   F919 40          391      RTI             ;A EJECUCION
   F91A             392 EJEC2
   F91A C90C        393      CMP    #0CH
   F91C D003        394      BNE    EJEC3
                    395
                    396             ;CONTINUA EL TRAZADO DE PROGAMA
                    397             ;EQUIVALENTE A PULSAR 'C' Y 'FIN'
                    398
   F91E 4CF900      399      JMP    GO
```

```
POSICION CODIGO     LINEA    CODIGO FUENTE

   F921             400 EJEC3
   F921 C90B        401      CMP    #08H
   F923 D040        402      BNE    EJEC4
                    403
                    404                    ;FIJA BREAKPOINT
                    405
   F925 AD0365      406      LDA    BRKFLG  ;HAY YA?
   F928 F004        407      BEQ    BS1     ;NO
   F92A A905        408      LDA    #05H
   F92C             409 BSER
   F92C 38          410      SEC            ;ERROR
   F92D 60          411      RTS
   F92E             412 BS1
   F92E A908        413      LDA    #08H
   F930 20FD58      414      JSR    INDIC   ;INDICADOR 'br'
   F933 20FD89      415      JSR    GETADE  ;OBTEN DIRECCION
   F936 AD0359      416      LDA    PARL
   F939 85FE        417      STA    CURAD
   F93B AD035A      418      LDA    PARH
   F93E 85FF        419      STA    CURAD+1 ;PASA DIRECCION
   F940 A000        420      LDY    #00H
   F942 B1FE        421      LDA    [CURAD],Y       ;DATO EN MEMORIA
   F944 AA          422      TAX
   F945 98          423      TYA
   F946 91FE        424      STA    [CURAD],Y       ;PON BREAK
   F948 D1FE        425      CMP    [CURAD],Y       ;HECHO?
   F94A F007        426      BEQ    BS2             ;SI
   F94C 8A          427      TXA
   F94D 91FE        428      STA    [CURAD],Y
   F94F             429 BSER1
   F94F A903        430      LDA    #03H
   F951 D0D9        431      BNE    BSER    ;ERROR DE NO MEMORIZADO
   F953             432 BS2
   F953 8E0366      433      STX    BRKDAT  ;SALVA EN MEMORIA
   F956 A5FE        434      LDA    CURAD
   F958 8D0367      435      STA    BRKADL
   F95B A5FF        436      LDA    CURAD+1
   F95D 8D0368      437      STA    BRKADH
   F960 EE0365      438      INC    BRKFLG  ;PON FLAG
   F963             439 BSSAL
   F963 18          440      CLC
   F964 60          441      RTS            ;SALIDA OK
   F965             442 EJEC4
   F965 C909        443      CMP    #09H
   F967 D024        444      BNE    EJEC5
                    445
                    446                    ;BORRA BREAKPOINT
                    447
   F969 AD0365      448      LDA    BRKFLG  ;HAY?
   F96C D004        449      BNE    BC1     ;SI
   F96E             450 BCER
   F96E A904        451      LDA    #04H    ;NO,ERROR
   F970 D0BA        452      BNE    BSER
   F972 AD0367      453 BC1  LDA    BRKADL
   F975 85FE        454      STA    CURAD
   F977 AD0368      455      LDA    BRKADH
   F97A 85FF        456      STA    CURAD+1
```

```
POSICION CODIGO     LINEA    CODIGO FUENTE

  F97C A000          457       LDY    #00H
  F97E AD0366        458       LDA    BRKDAT
  F981 91FE          459       STA    [CURAD],Y
  F983 D1FE          460       CMP    [CURAD],Y
  F985 D0C8          461       BNE    BSER1
  F987 8C0365        462       STY    BRKFLG  ;QUITA EL FLAG
  F98A 4CFD6F        463       JMP    OFF     ;TEXTO Y FINAL
                     464
  F98D C90A          465 EJEC5 CMP    #0AH
  F98F D016          466       BNE    EJEC6
                     467
                     468              ;MUESTRA LA DIRECCION DEL BREAK ACTUAL
                     469
  F991 AD0365        470       LDA    BRKFLG  ;HAY
  F994 F0D8          471       BEQ    BCER    ;NO,ERROR
  F996 A909          472       LDA    #09H
  F998 20FD58        473       JSR    INDIC   ;INDICADOR "b?"
  F99B AD0367        474       LDA    BRKADL
  F99E AC0368        475       LDY    BRKADH
  F9A1 20FD22        476       JSR    OUTADD  ;SACA LA DIRECCION
  F9A4 4CF81D        477       JMP    STRT3   ;Y FINAL
  F9A7               478 EJEC6
  F9A7 C908          479       CMP    #08H
  F9A9 D012          480       BNE    EJEC7
                     481
                     482              ;PROTEGE CONTRA ESCRITURA RAM $400H-$7FF(AMPLIAC.)
                     483
  F9AB ADA000        484       LDA    RAIORB
  F9AE 4910          485       EOR    #00010000B
  F9B0 8DA000        486       STA    RAIORB;BASCULA LINEA
  F9B3 2910          487       AND    #00010000B      ;MIRA VALOR ACTUAL
                     488
  F9B5 F003          489       BEQ    PROOF
  F9B7 4CFD67        490       JMP    ON      ;INDICADOR DE 'on'
  F9BA               491 PROOF
  F9BA 4CFD6F        492       JMP    OFF     ;INDICADOR DE 'of'
  F9BD               493 EJEC7
  F9BD C902          494       CMP    #02H
  F9BF D072          495       BNE    EJEC8
                     496
                     497              ;LISTA/ALTERA LOS REGISTROS
                     498
  F9C1 A900          499       LDA    #00H
  F9C3               500 RB
  F9C3 8D035B        501       STA    SCR1
  F9C6 20FD55        502       JSR    CLRIND  ;SACA NOMBRE
  F9C9 AD035B        503       LDA    SCR1    ;TOMA PUNTERO
  F9CC AA            504       TAX
  F9CD E000          505       CPX    #00H    ;EN P.C.?
  F9CF D027          506       BNE    R1      ;NO
  F9D1 AD0369        507       LDA    PCLR    ;PASA DATOS
  F9D4 8D0359        508       STA    PARL
  F9D7 AC036A        509       LDY    PCHR
  F9DA 8C035A        510       STY    PARH
  F9DD 20FD22        511       JSR    OUTADD  ;SACA VALOR ACTUAL
  F9E0 A901          512       LDA    #01H
  F9E2 8D035F        513       STA    MODEC   ;4 BY
```

```
POSICION CODIGO     LINEA    CODIGO FUENTE

   F9E5 20FDB7      514      JSR    GETP3   ;OBTEN DATOS
   F9E8 48          515      PHA             ;SALVA TECLA
   F9E9 AD0359      516      LDA    PARL    ;CON NUEVO VALOR
   F9EC 8D0369      517      STA    PCLR
   F9EF AD035A      518      LDA    PARH
   F9F2 8D036A      519      STA    PCHR
   F9F5 4CFA13      520      JMP    R2      ;Y SIGUE
   F9F8             521 R1
   F9F8 BD036A      522      LDA    SR-1,X  ;RESTO DE REGISTROS
   F9FB 8D0359      523      STA    PARL
   F9FE 20FD2D      524      JSR    OUTADH
   FA01 A900        525      LDA    #00H
   FA03 8D035F      526      STA    MODEC   ;2BY
   FA06 20FDB7      527      JSR    GETP3   ;OBTEN DATOS
   FA09 48          528      PHA
   FA0A AE035B      529      LDX    SCR1
   FA0D AD0359      530      LDA    PARL
   FA10 9D036A      531      STA    SR-1,X  ;NUEVO VALOR
   FA13             532 R2
   FA13 68          533      PLA             ;TOMA ULTIMA TECLA
   FA14 C912        534      CMP    #12H
   FA16 D002        535      BNE    R3
   FA18 18          536      CLC
   FA19 60          537      RTS             ;SALIDA
   FA1A             538 R3
   FA1A AE035B      539      LDX    SCR1
   FA1D C910        540      CMP    #10H
   FA1F D009        541      BNE    R4
   FA21 E8          542      INX             ;AUMENTA
   FA22 E006        543      CPX    #06H
   FA24 9009        544      BCC    R5
   FA26 A200        545      LDX    #00H
   FA28 F005        546      BEQ    R5
   FA2A             547 R4
   FA2A CA          548      DEX             ;DISMINUYE
   FA2B 1002        549      BPL    R5
   FA2D A205        550      LDX    #05H
   FA2F             551 R5
   FA2F 8A          552      TXA
   FA30 4CF9C3      553      JMP    RB      ;REPITE
                    554
   FA33             555 EJEC8
   FA33 C903        556      CMP    #03H
   FA35 D041        557      BNE    EJEC9
                    558
                    559                     ;MEMORIZA VECTOR
                    560
   FA37 A90A        561      LDA    #0AH
   FA39 20FD58      562      JSR    INDIC   ;INDICADOR 'Vd'
   FA3C 20FD89      563      JSR    GETADE  ;OBTEN DATO
   FA3F AD0359      564      LDA    PARL
   FA42 8D035B      565      STA    SCR1    ;SALVA
   FA45 AD035A      566      LDA    PARH
   FA48 8D035C      567      STA    SCR2
   FA4B A90B        568      LDA    #0BH
   FA4D 20FD55      569      JSR    CLRIND  ;BORRA E INDICADOR 'VA'
   FA50 20FD89      570             JSR     GETADE
```

```
POSICION CODIGO     LINEA   CODIGO FUENTE

   FA53 AD0359      571       LDA     PARL
   FA56 85FE        572       STA     CURAD
   FA58 AD035A      573       LDA     PARH
   FA5B 85FF        574       STA     CURAD+1
   FA5D A000        575       LDY     #00H
   FA5F AD035B      576       LDA     SCR1    ;PASA VECTOR
   FA62 91FE        577       STA     [CURAD],Y
   FA64 D1FE        578       CMP     [CURAD],Y       ,MEMORIZADO?
   FA66 D00C        579       BNE     VECER   ;NO,ERROR
   FA68 C8          580       INY
   FA69 AD035C      581       LDA     SCR2
   FA6C 91FE        582       STA     [CURAD],Y
   FA6E D1FE        583       CMP     [CURAD],Y
   FA70 D002        584       BNE     VECER   ;ERROR
   FA72 18          585       CLC
   FA73 60          586       RTS     ;RETORNO
   FA74             587 VECER
   FA74 A903        588       LDA     #03H    ;ENTRADA DE NO MEMORIZA
   FA76 38          589       SEC             ;FLAG DE ERROR
   FA77 60          590       RTS
   FA78             591 EJEC9
   FA78 C90D        592       CMP     #0DH
   FA7A D01C        593       BNE     EJEC10
                    594
                    595               ;CALCULO DE OFFSET EN INSTRUCCION DE RAMIFICACION
                    596
   FA7C A914        597       LDA     #14H
   FA7E 20FD58      598       JSR     INDIC   ;TEXTO 'ca'
   FA81 20FD89      599       JSR     GETADE  ;OBTEN DIRECCION INICIAL Y FINAL
   FA84 A2FF        600       LDX     #0FFH   ;
                    601
   FA86 9A          602       TXS             ;INICIA STACK
   FA87 E8          603       INX
   FA88 8E0361      604       STX     PUNTO   ;BORRA PUNTOS
   FA8B EE035A      605       INC     PARH    ;RESULTADO=FINAL-INIC.-2
   FA8E 18          606       CLC
   FA8F AD0359      607       LDA     PARL
   FA92 ED035A      608       SBC     PARH
   FA95 4CF81A      609       JMP     STRT21  ;PRESENTA DATOS Y FINAL
   FA98             610 EJEC10
   FA98 C907        611       CMP     #07H
   FA9A D012        612       BNE     EJEC11
                    613
                    614               ;BASCULA MOTOR DEL CASSETTE
                    615
   FA9C ADA000      616       LDA     RAIORB
   FA9F 4920        617       EOR     #00100000B
   FAA1 8DA000      618       STA     RAIORB
   FAA4 2920        619       AND     #00100000B      ;TOMA VALOR ACTUAL
   FAA6 F003        620       BEQ     BASOF           ;OFF
   FAA8 4CFD67      621       JMP     ON
   FAAB             622 BASOF
   FAAB 4CFD6F      623       JMP     OFF
                    624
                    625
   FAAE             626 EJEC11
   FAAE C904        627       CMP     #04H
```

```
POSICION CODIGO     LINEA   CODIGO FUENTE

   FAB0 F003        628      BEQ    SCE
   FAB2 4CFB4F      629      JMP    EJEC12
                    630
                    631             ;SALVA DATOS EN CASSETTE
                    632             ;OBTEN PARAMETROS
   FAB5             633 SCE
   FAB5 A90C        634      LDA    #0CH
   FAB7 20FD58      635      JSR    INDIC   ;IDENTIFICACION 'id'
   FABA 20FD97      636      JSR    GETPAR  ;OBTEN NUMERO
   FABD AD0359      637      LDA    PARL
   FAC0 8D035E      638      STA    ID
   FAC3 A90D        639      LDA    #0DH
   FAC5 20FD55      640      JSR    CLRIND  ;DIRECCION INICIAL 'di'
   FAC8 20FD90      641      JSR    GETADD
   FACB AD0359      642      LDA    PARL
   FACE 85FE        643      STA    CURAD
   FAD0 AD035A      644      LDA    PARH
   FAD3 85FF        645      STA    CURAD+1
   FAD5 A90E        646      LDA    #0EH
   FAD7 20FD55      647      JSR    CLRIND  ;DIRECCION FINAL
   FADA 20FD90      648      JSR    GETADD
   FADD EE0359      649      INC    PARL
   FAE0 D003        650      BNE    CS0
   FAE2 EE035A      651      INC    PARH    ;AUMENTA
                    652
                    653             ;SALVA DATOS
                    654
   FAE5             655 CS0
   FAE5 20FC5A      656      JSR    STARTC
   FAE8 AD0370      657      LDA    TAPDEL
   FAEB 8D0360      658      STA    PRIMER  ;NUMERO DE SYNC/256
   FAEE             659 CS1
   FAEE A916        660      LDA    #16H    ;SYN
   FAF0 20FC98      661      JSR    OUTCSC  ;SACA SIN CHECKSUM
   FAF3 88          662      DEY
   FAF4 D0F8        663      BNE    CS1
   FAF6 CE0360      664      DEC    PRIMER
   FAF9 D0F3        665      BNE    CS1
   FAFB A92A        666      LDA    #2AH
   FAFD 20FC98      667      JSR    OUTCSC
   FB00 AD035E      668      LDA    ID
   FB03 20FC98      669      JSR    OUTCSC  ;IDENTIFICACION
   FB06 A5FE        670      LDA    CURAD   ;DIR INICIAL
   FB08 20FC95      671      JSR    OUTCCC
   FB0B A5FF        672      LDA    CURAD+1
   FB0D 20FC95      673      JSR    OUTCCC
   FB10 AD0359      674      LDA    PARL    ;DIR FINAL
   FB13 20FC95      675      JSR    OUTCCC
   FB16 AD035A      676      LDA    PARH
   FB19 20FC95      677      JSR    OUTCCC
   FB1C             678 CS2
   FB1C A5FE        679      LDA    CURAD   ;FINAL DE TRANSMISION
   FB1E CD0359      680      CMP    PARL
   FB21 D01C        681      BNE    CS3     ;NO
   FB23 A5FF        682      LDA    CURAD+1
   FB25 CD035A      683      CMP    PARH
   FB28 D015        684      BNE    CS3     ;NO
```

```
POSICION CODIGO      LINEA   CODIGO FUENTE

                     685
                     686                    ;FINAL DE TRANSMISION
FB2A A92F            687        LDA    #2FH    ;MARCA
FB2C 20FC98          688        JSR    OUTCSC
FB2F AD035B          689        LDA    SCR1    ;CHECKSUM
FB32 20FC98          690        JSR    OUTCSC
FB35 AD035C          691        LDA    SCR2
FB38 20FC98          692        JSR    OUTCSC
FB3B 18              693        CLC
FB3C 4CFC4D          694        JMP    LCOU    ;FINAL
                     695
                     696                ;TRANSMITE DATOS
                     697
FB3F                 698 CS3
FB3F A000            699        LDY    #00H
FB41 B1FE            700        LDA    [CURAD],Y
FB43 20FC95          701        JSR    OUTCCC
FB46 E6FE            702        INC    CURAD   ;AUMENTA VECTOR
FB48 D0D2            703        BNE    CS2
FB4A E6FF            704        INC    CURAD+1
FB4C 4CFB1C          705        JMP    CS2
                     706
FB4F                 707 EJEC12
                     708
FB4F C905            709        CMP    #05H
FB51 D004            710        BNE    EJEC13
                     711
                     712                ;CARGA DATOS DEL CASSETTE
                     713
FB53 A000            714        LDY    #00H    ;FLAG DE LOAD
FB55 F002            715        BEQ    LCE     ;Y COMUN
                     716
                     717                ;VERIFICA DATOS SALVADOS EN EL CASSETTE
                     718
FB57                 719 EJEC13
FB57 A080            720        LDY    #80H    ;FLAG DE VERIFICACION
                     721
                     722                ;PARTE COMUN DE CARGAR VERIFICAR DATOS
                     723
FB59                 724 LCE
FB59 8C0363          725        STY    LODIVER
FB5C A90C            726        LDA    #0CH
FB5E 20FD58          727        JSR    INDIC   ;TEXTO 'id'
FB61 20FD97          728        JSR    GETPAR  ;OBTEN IDENTIFICADOR
FB64 AD0359          729        LDA    PARL
FB67 8D035E          730        STA    ID
FB6A C9FF            731        CMP    #0FFH
FB6C D022            732        BNE    LCL     ;NO CARGA
FB6E A90D            733        LDA    #0DH
FB70 20FD55          734        JSR    CLRIND  ;TEXTO 'di'
FB73 20FD90          735        JSR    GETADD  ;NUEVA DIR INICIAL
                     736
FB76 AD0359          737        LDA    PARL
                     738
                     739
FB79 85FE            740        STA    CURAD
FB7B AD035A          741        LDA    PARH
```

```
POSICION CODIGO     LINEA   CODIGO FUENTE

    FB7E 85FF       742     STA   CURAD+1
    FB80 A90E       743     LDA   #0EH
    FB82 20FD55     744     JSR   CLRIND  ;TEXTO 'dF'
    FB85 20FD90     745     JSR   GETADD  ;NUEVA DIR. FINAL
    FB88 EE0359     746     INC   PARL    ;AUMENTA FINAL
    FB8B D003       747     BNE   LCL
    FB8D EE035A     748     INC   PARH
                    749
                    750           ;CARGA DATOS SINCRONIZA
                    751
    FB90            752 LCL
    FB90 20FC5A     753     JSR   STARTC
    FB93 0903       754     ORA   #03H    ;SELECC DIGITO IZQDO
    FB95 8DA000     755     STA   RAIORB
    FB98 A9FF       756     LDA   #0FFH
    FB9A 8DA003     757     STA   RAIODRA ;RA A SALIDAS
    FB9D            758 LC1
    FB9D 20FC6D     759     JSR   SYNC    ;SINCRONIZA
    FBA0            760 LC2
    FBA0 20FCC6     761     JSR   GETCSC
    FBA3 C92A       762     CMP   #2AH
    FBA5 F006       763     BEQ   LC21    ;SI
    FBA7 C916       764     CMP   #16H    ;SYN?
    FBA9 D0F2       765     BNE   LC1     ;NO
    FBAB F0F3       766     BEQ   LC2
                    767           ;ENCONTRADOS DATOS
    FBAD            768 LC21
    FBAD 0E0363     769     ASL   LODIVER ;TOMA BIT?
    FBB0 6A         770     ROR   A
    FBB1 8D035F     771     STA   MODEC
    FBB4 20FCC6     772     JSR   GETCSC
    FBB7 8DA001     773     STA   RAIORA  ;SACA 'id' AL DISPLAY
    FBBA CD035E     774     CMP   ID
    FBBD F022       775     BEQ   LC4     ;COINCIDE
    FBBF AD035E     776     LDA   ID
    FBC2 F01D       777     BEQ   LC4     ;PRIMERO ENCONTRADO
    FBC4 C9FF       778     CMP   #0FFH
    FBC6 F00A       779     BEQ   LC3     ;ESTE CON NUEVA DIRECCION
    FBC8 AD035F     780     LDA   MODEC
    FBCB 0940       781     ORA   #40H
    FBCD 8D035F     782     STA   MODEC   ;FLAG DE IGNORAR
    FBD0 D00F       783     BNE   LC4     ;Y SIGUE
                    784
                    785           ;DIRECCIONES SUMINISTRADAS POR EL USUARIO
                    786
    FBD2            787 LC3
    FBD2 20FCF0     788     JSR   GETCCC  ;PUNTEA A DATOS
    FBD5 20FCF0     789     JSR   GETCCC
    FBD8 20FCF0     790     JSR   GETCCC
    FBDB 20FCF0     791     JSR   GETCCC
    FBDE 4CF8F7     792     JMP   LC5
                    793
    FBE1            794 LC4
                    795           ;DIRECCIONES EN CINTA
    FBE1 20FCF0     796     JSR   GETCCC  ;DIR INICIAL
    FBE4 85FE       797     STA   CURAD
    FBE6 20FCF0     798     JSR   GETCCC
```

```
POSICION CODIGO     LINEA   CODIGO FUENTE

   FBE9 85FF        799      STA   CURAD+1
   FBEB 20FCF0      800      JSR   GETCCC  ;DIR FINAL
   FBEE 8D0359      801      STA   PARL
   FBF1 20FCF0      802      JSR   GETCCC
   FBF4 8D035A      803      STA   PARH
                    804
                    805            ;LECTURA DE DATOS
                    806
   FBF7             807 LC5
   FBF7 20FCC6      808      JSR   GETCSC
   FBFA A6FE        809      LDX   CURAD   ;FINAL?
   FBFC EC0359      810      CPX   PARL
   FBFF D007        811      BNE   LC6
   FC01 A6FF        812      LDX   CURAD+1
   FC03 EC035A      813      CPX   PARH
   FC06 F01F        814      BEQ   LCF     ;FINAL
   FC08             815 LC6
   FC08 20FCF3      816      JSR   AJCK    ;AJUSTA CHECKSUM
   FC0B A000        817      LDY   #00H
   FC0D 2C035F      818      BIT   MODEC
   FC10 700C        819      BVS   LC8     ;IGNORA
   FC12 3002        820      BMI   LC7     ;VERIFICAR
   FC14 91FE        821      STA   (CURAD),Y       ;SALVA DATO
   FC16             822 LC7
   FC16 D1FE        823      CMP   (CURAD),Y       ;VERIFICA
   FC18 F004        824      BEQ   LC8     ;OK
   FC1A A903        825      LDA   #3
   FC1C D023        826      BNE   LCER
   FC1E             827 LC8
   FC1E E6FE        828      INC   CURAD   ;AUMENTA VECTOR
   FC20 D0D5        829      BNE   LC5     ;Y REPITE
   FC22 E6FF        830      INC   CURAD+1
   FC24 4CFBF7      831      JMP   LC5
                    832
                    833            ;FIN DE DATOS
                    834
   FC27             835 LCF
   FC27 C92F        836      CMP   #2FH    ;MARCA DE FINAL?
   FC29 F004        837      BEQ   LC9     ;SI
   FC2B A900        838      LDA   #00H    ;NO,ERROR
   FC2D D012        839      BNE   LCER
   FC2F             840 LC9
   FC2F 20FCC6      841      JSR   GETCSC  ;MIRA CKSUM
   FC32 CD035B      842      CMP   SCR1
   FC35 D008        843      BNE   LC10    ;MAL
   FC37 20FCC6      844      JSR   GETCSC
   FC3A CD035C      845      CMP   SCR2
   FC3D F005        846      BEQ   LCOR
   FC3F             847 LC10
   FC3F A902        848      LDA   #02H    ;ERROR
   FC41             849 LCER
   FC41 38          850      SEC           ;FLAG DE MAL
   FC42 B001        851      BCS   LCSA
                    852
   FC44             853 LCOR
   FC44 18          854      CLC           ;FLAG DE OK
   FC45             855 LCSA
```

```
POSICION CODIGO     LINEA   CODIGO FUENTE

   FC45 2C035F      856     BIT     MODEC   ;PUENTEANDO?
   FC48 5003        857     BVC     LCOU    ;NO
   FC4A 4CFB90      858     JMP     LCL     ;SI ,REPITE
   FC4D             859 LCOU
   FC4D 48'         860     PHA
   FC4E ADA000      861     LDA     RAIORB
   FC51 290F        862     AND     #11011111B      ;PARA MOTOR
   FC53 090F        863     ORA     #0FH    ;SELECCION A REPOSO
   FC55 8DA000      864     STA     RAIORB
   FC58 68          865     PLA
   FC59 60          866     RTS     FINAL
                    867
                    868             ;SUBRUTINAS DE CASSETTE
                    869
   FC5A             870 STARTC
   FC5A A000        871     LDY     #00H    ;INICIALIZA
   FC5C 8C035B      872     STY     SCR1    ;CKSUM
   FC5F 8C035C      873     STY     SCR2    .
   FC62 A920        874     LDA     #20H
   FC64 0DA000      875     ORA     RAIORB  ;MOTOR ON
   FC67 29F0        876     AND     #0F0H
   FC69 8DA000      877     STA     RAIORB
   FC6C 60          878     RTS
                    879
                    880             ;SINCRONIZA LECTURA
                    881
   FC6D             882 SYNC
   FC6D A96D        883     LDA     #6DH    ;SACA 'S' A DISPLAY
   FC6F 8DA001      884     STA     RAIORA
   FC72 AD035F      885     LDA     MODEC
   FC75 0940        886     ORA     #40H
   FC77 8D035F      887     STA     MODEC
                    888
   FC7A             889 SY1
   FC7A 20FD02      890     JSR     SYNBIT
   FC7D 66FD        891     ROR     CHAR
   FC7F A5FD        892     LDA     CHAR.
   FC81 C916        893     CMP     #16H
   FC83 D0F5        894     BNE     SY1     ;ESPERA A 'SYN'
   FC85 A20A        895     LDX     #0AH
   FC87             896 SY2
   FC87 20FCC6      897     JSR     GETCSC  ;ESPERA 10 'SYN'
   FC8A C916        898     CMP     #16H
   FC8C D0EC        899     BNE     SY1
   FC8E CA          900     DEX
   FC8F D0F6        901     BNE     SY2
   FC91 8EA001      902     STX     RAIORA  ;BORRA 'S'
   FC94 60          903     RTS
                    904
                    905             ;SACA BYTE A CASSETTE
                    906
   FC95             907 OUTCCC
   FC95 20FCF3      908     JSR     AJCK    ;CON CHECKSUM
   FC98             909 OUTCSC
   FC98 A209        910     LDX     #09H    ;SIN CHECKSUM
   FC9A 8C035D      911     STY     SCR3
   FC9D 85FD        912     STA     CHAR
```

```
POSICION CODIGO     LINEA   CODIGO FUENTE

  FC9F ADA000       913       LDA     RAIORB  ;OBTEN NIVEL ANTERIOR
  FCA2              914 OUTCBI
  FCA2 46FD         915       LSR     CHAR
  FCA4 4909         916       EOR     #09H
  FCA6 8DA000       917       STA     RAIORB  ;INVIERTE NIVEL
  FCA9 AC0372       918       LDY     TAPET1
  FCAC              919 OUTC1
  FCAC 88           920       DEY
  FCAD D0FD         921       BNE     OUTC1   ;RETARDO
  FCAF 9012         922       BCC     OUTCN
  FCB1 4909         923       EOR     #09H    ;BASCULA SI BIT=1
  FCB3 8DA000       924       STA     RAIORB
                    925
                    926               ;SEGUNDO SEMICICLO
  FCB6              927 OUTC2
  FCB6 AC0373       928       LDY     TAPET2
  FCB9              929 OUTC3
  FCB9 88           930       DEY
  FCBA D0FD         931       BNE     OUTC3
  FCBC CA           932       DEX
  FCBD D0E3         933       BNE     OUTCBI  ;OTRO BIT
  FCBF AC035D       934       LDY     SCR3
  FCC2 60           935       RTS
  FCC3              936 OUTCN
  FCC3 EA           937       NOP
  FCC4 90F0         938       BCC     OUTC2
                    939
                    940               ;LECTURA DEL BYTE DE CINTA
                    941
  FCC6              942 GETCSC
  FCC6 8E035D       943       STX     SCR3
  FCC9 A208         944       LDX     #08H
  FCCB 20FD08       945       JSR     GETCTR
  FCCE B014         946       BCS     GETC3
  FCD0              947 GETC1
  FCD0 20FD08       948       JSR     GETCTR
  FCD3 9004         949       BCC     GETC2
  FCD5 20FD08       950       JSR     GETCTR
  FCD8 38           951       SEC
  FCD9              952 GETC2
  FCD9 66FD         953       ROR     CHAR
  FCDB CA           954       DEX
  FCDC D0F2         955       BNE     GETC1
  FCDE A5FD         956       LDA     CHAR    ;FIN DE TOMA
  FCE0 AE035D       957       LDX     SCR3
  FCE3 60           958       RTS
  FCE4              959 GETC3
  FCE4 2C035F       960       BIT     MODEC
  FCE7 70F7         961       BVS     GETC3-4
  FCE9 68           962       PLA
  FCEA 68           963       PLA
  FCEB A901         964       LDA     #01H    ;ERROR EN LOAD
  FCED 4CFC41       965       JMP     LCER
                    966
                    967               ;LECTURA DE BYTE DE CINTA CON CHECKSUM
                    968
  FCF0              969 GETCCC
```

```
POSICION CODIGO     LINEA    CODIGO FUENTE

   FCF0 20FCC6       970       JSR     GETCSC
                     971
                     972
                     973               ;AJUSTA CHECKSUM CON VALOR EN ACUMULADOR
                     974
   FCF3              975 AJCK
   FCF3 A8           976       TAY
   FCF4 18           977       CLC
   FCF5 6D035B       978       ADC     SCR1
   FCF8 8D035B       979       STA     SCR1
   FCFB 9003         980       BCC     AJCK1
   FCFD EE035C       981       INC     SCR2
   FD00              982 AJCK1
   FD00 98           983       TYA
   FD01 60           984       RTS
                     985               ;OBTEN BIT DURANTE SINCRONIZACION
                     986
   FD02              987 SYNBIT
   FD02 20FD08       988       JSR     GETCTR
   FD05 B001         989       BCS     GETCTR
   FD07 60           990       RTS
                     991
                     992               ;OBTEN TIEMPO DE TRANSICION
                     993
   FD08              994 GETCTR
   FD08 A0FF         995       LDY     #0FFH
   FD0A              996 GETCTR1
   FD0A ADA000       997       LDA     RAIORB
   FD0D 2940         998       AND     #40H
   FD0F CD0362       999       CMP     OLD
   FD12 F0F6        1000       BEQ     GETCTR1
   FD14 8D0362      1001       STA     OLD
   FD17 4CF100      1002       JMP     M_PATCH1        ;SALTA A MODIF. DEL MI650C
                    1003 ;
                    1004 ;     ESTA MODIFICACION INCLUYE EL RETURN.
                    1005 ;
                    1006 ;*****************************************************
                    1007               ;SUBRUTINAS DE SALIDA A DISPLAY
                    1008               ;SACA 4 CARACTERES EN ADDRESS
                    1009               ;ENTRADA ACC, PARTE BAJA EMPACADA
                    1010               ;R(Y) PARTE ALTA EMPACADA
                    1011
                    1012       ORG     0FD22H  ;AJUSTAR DIRECC. ANTIC MONITOR.
   FD22             1013 OUTADD
   FD22 48          1014       PHA
   FD23 98          1015       TYA
   FD24 A200        1016       LDX     #00H
   FD26 20FD34      1017       JSR     OUTACC  ;SACA PARTE ALTA
   FD29 68          1018       PLA
   FD2A 4CFD34      1019       JMP     OUTACC  ;SACA PARTE BAJA
                    1020
                    1021
                    1022               ;SACA DOS CARACTERES EN ADRESS (DIGITO 0 Y 1)
                    1023               ;ENTRADA ACC DATO EMPACADO
                    1024
   FD2D             1025 OUTADH
   FD2D A200        1026       LDX     #00H
```

```
POSICION CODIGO     LINEA   CODIGO FUENTE

   FD2F 4CFD34      1027      JMP    OUTACC  ;SACA
                    1028
                    1029             ;SACA DOS CARACTERES EN DATOS
                    1030             ;ENTRADA ACC DATO
   FD32             1031 OUTDAT
   FD32 A204        1032      LDX    #04H
                    1033
                    1034             ;SACA ACC A DISPLAY COMO DOS CARACTERES, A
                    1035             ;PARTIR DEL DIGITO INDICADO POR R(X).
                    1036             ;INCREMENTANDO ESTE
   FD34             1037 OUTACC
   FD34 48          1038      PHA    ;SALVA DATO
   FD35 4A          1039      LSR    A
   FD36 4A          1040      LSR    A
   FD37 4A          1041      LSR    A
   FD38 4A          1042      LSR    A
   FD39 20FD3D      1043      JSR    OUTACH  ;ALTO
   FD3C 68          1044      PLA
                    1045
                    1046             ;SACA BITS 0-3 DE ACC COMO UN CARACTER HEX
                    1047
   FD3D             1048 OUTACH
   FD3D 290F        1049      AND    #0FH
   FD3F A8          1050      TAY
   FD40 B9FF9A      1051      LDA    SEGMTB,Y        ;CODIGO DE SEGMENTOS
   FD43 9D0353      1052      STA    DISBUF,X        ;COLOCA EN BUFFER
   FD46 E8          1053      INX            ;AUMENTA LUGAR
   FD47 60          1054      RTS
                    1055
                    1056             ;BORRA DISPLAY
                    1057
   FD48             1058 CLRDIS
   FD48 48          1059      PHA    ;SALVA ACUMULADOR
   FD49 A900        1060      LDA    #00H
   FD4B A206        1061      LDX    #06H
   FD4D             1062 CLRD1
   FD4D 9D0352      1063      STA    DISBUF-1,X
   FD50 CA          1064      DEX
   FD51 D0FA        1065      BNE    CLRD1
   FD53 68          1066      PLA
   FD54 60          1067      RTS
                    1068
                    1069             ;SACA TEXTO CON BORRADO PREVIO
                    1070
   FD55             1071 CLRIND
   FD55 20FD48      1072      JSR    CLRDIS
                    1073
                    1074             ;SACA INDICADOR DE OPERACION A DISPLAY
                    1075             ;EL TEXTO (2 CARACTERES) EN LA ZONA DE DATOS)
                    1076             ;EN ACC ESTA EL CODIGO DEL TEXTO
                    1077
   FD58             1078 INDIC
   FD58 0A          1079      ASL    A       ;2
   FD59 A8          1080      TAY            ;PUNTERO
   FD5A B9FFAA      1081      LDA    MENTAB,Y
   FD5D 8D0357      1082      STA    DISBUF+4
   FD60 B9FFAB      1083      LDA    MENTAB+1,Y
```

```
POSICION CODIGO     LINEA   CODIGO FUENTE

    FD63 8D0358     1084        STA     DISBUF+5
    FD66 60         1085        RTS
                    1086
                    1087                ;SACA TEXTO 'ON' TEMPORALMENTE
                    1088
    FD67            1089 ON
    FD67 A912       1090        LDA     #12H
    FD69 20FD55     1091        JSR     CLRIND
    FD6C 4CFD74     1092        JMP     ESPER   ;ESPERA FINAL
                    1093
                    1094                ;SACA TEXTO 'OF' TEMPORALMENTE
                    1095
    FD6F            1096 OFF
    FD6F A913       1097        LDA     #13H
    FD71 20FD55     1098        JSR     CLRIND
                    1099
                    1100                ;HAZ RETARDO DE APROX. 8 seg.
                    1101
    FD74            1102 ESPER
    FD74 A064       1103        LDY     #64H
    FD76 8C035B     1104        STY     SCR1    ;CONTADOR
    FD79            1105 ESPER1
    FD79 20FE4D     1106        JSR     REFDIS  ;PRESENTA DATOS
    FD7C CE035B     1107        DEC     SCR1
    FD7F D0F8       1108        BNE     ESPER1
    FD81 18         1109        CLC
    FD82 60         1110        RTS
                    1111
                    1112                ;HAZ RETARDO DE APROXIMADAMENTE 1.5 mseg.
                    1113
    FD83            1114 DELAY
    FD83 A2FF       1115        LDX     #0FFH
    FD85 CA         1116        DEX
    FD86 D0FD       1117        BNE     DELAY+2
    FD88 60         1118        RTS
                    1119                ;*******************************
                    1120                ;SUBRUTINAS.DE ENTRADA DE DATOS
                    1121                ;*******************************
                    1122
                    1123
                    1124                ;DIRECCION HASTA FIN ,SI TECLA NO HEX O NO
                    1125                ;HAY DATOS,ERROR
                    1126
    FD89            1127 GETADE
    FD89 A901       1128        LDA     #1H
    FD8B 8D035F     1129        STA     MODEC
    FD8E D00C       1130        BNE     GETP1
                    1131
                    1132                ;DIRECCION HASTA FIN SI TECLA NO HEX, ERROR
                    1133
    FD90            1134 GETADD
    FD90 A901       1135        LDA     #1H
    FD92 8D035F     1136        STA     MODEC
    FD95 D013       1137        BNE     GETP2
                    1138
                    1139                ;PARAMETRO (2BY) HASTA FIN SI TECLA NO HEX
                    1140                ;O NO HAY DATOS ERROR
```

```
POSICION CODIGO      LINEA   CODIGO FUENTE

                     1141
                     1142
FD97                 1143 GETPAR
FD97 A900            1144        LDA     #0H
FD99 8D035F          1145        STA     MODEC
FD9C                 1146 GETP1
FD9C 20FDAA          1147        JSR     GETP2
FD9F AD0360          1148        LDA     PRIMER  ;HAY DATOS?
FDA2 F001            1149        BEQ     GAT
FDA4 60              1150        RTS             ;OK
FDA5                 1151 GAT
FDA5 A906            1152        LDA     #6H
FDA7 4CF7D3          1153        JMP     ERROR   ;ERROR
FDAA                 1154 GETP2
FDAA 20FDB7          1155        JSR     GETP3
FDAD C912            1156        CMP     #12H    ;FINAL?
FDAF D001            1157        BNE     GAT2
FDB1 60              1158        RTS             ;OK
FDB2                 1159 GAT2
FDB2 A907            1160        LDA     #7H
FDB4 4CF7D3          1161        JMP     ERROR   ;NO HEX
                     1162
                     1163                ;OBTEN STRING DE CARACTERES HASTA TECLA
                     1164                ;NO HEX. EMPACANDO EN PARH, PARL SEGUN
                     1165                ;SEA EL VALOR DE MODEC SACA 2 o 4 BYTES
                     1166                ;A DISPLAY
                     1167
FDB7                 1168 GETP3
FDB7 A930            1169        LDA     #110000B
FDB9 AC035F          1170        LDY     MODEC
FDBC F002            1171        BEQ     GETP31
FDBE A93C            1172        LDA     #111100B
FDC0                 1173 GETP31
FDC0 8D0361          1174        STA     PUNTO   ;PUNTOS INDICATIVOS
FDC3 A900            1175        LDA     #0H
FDC5 8D0360          1176        STA     PRIMER  ;FLAG
FDC8                 1177 GETP32
FDC8 20FE09          1178        JSR     GETKEY  ;OBTEN TECLA
FDCB 90FB            1179        BCC     GETP32  ;NO HAY
FDCD C910            1180        CMP     #10H    ;HEX?
FDCF 9001            1181        BCC     GAT3
FDD1 60              1182        RTS             ;NO HEX
FDD2                 1183 GAT3
FDD2 20FDE6          1184        JSR     PACK    ;EMPACA
FDD5 AE035F          1185        LDX     MODEC
FDD8 F006            1186        BEQ     GETP33
FDDA 20FD22          1187        JSR     OUTADD  ;SACA 4 BYTES
FDDD 4CFDC8          1188        JMP     GETP32  ;Y REPITE
FDE0                 1189 GETP33
FDE0 20FD2D          1190        JSR     OUTADH  ;SACA 2 BYTES
FDE3 4CFDC8          1191        JMP     GETP32  ;Y REPITE
                     1192
                     1193                ;EMPACA VALOR HEX EN ACC EN PARH, PARL
                     1194                ;SI PRIMERA VEZ, HAZ 0 PREVIAMENTE PARH,PARL
                     1195
FDE6                 1196 PACK
FDE6 AC0360          1197        LDY     PRIMER
```

```
POSICION CODIGO     LINEA    CODIGO FUENTE

   FDE9 D009        1198       BNE     PACK1
   FDEB 8C035A      1199       STY     PARH    ;BORRA
   FDEE 8C0359      1200       STY     PARL
   FDF1 EE0360      1201       INC     PRIMER  ;IMPIDE MAS VECES
   FDF4             1202 PACK1
   FDF4 A004        1203       LDY     #4H
   FDF6 0E0359      1204       ASL     PARL    ;ROTA A IZQDAS 4 BITS
   FDF9 2E035A      1205       ROL     PARH
   FDFC 88          1206       DEY
   FDFD D0F7        1207       BNE     PACK1+2
   FDFF 0D0359      1208       ORA     PARL    ;Y ANADE NUEVO VALOR
   FE02 8D0359      1209       STA     PARL
   FE05 AC035A      1210       LDY     PARH
   FE08 60          1211       RTS
                    1212       ;***************************
                    1213       ;MANEJO DE DISPLAY Y TECLADO
                    1214       ;***************************
                    1215
                    1216       ;SIMULTANEAMENTE PRESENTA LOS DATOS EN DISBUF
                    1217       ;Y HACE SCANNER DE TECLADO
                    1218       ;SI NO HAY TECLA PULSADA, CY=0
                    1219       ;SI HAY TECLA CY=1 Y CODIGO EN ACC
                    1220
   FE09             1221 GETKEY
   FE09 AD0361      1222       LDA     PUNTO   ;COLOCA PUNTOS
   FE0C 85FD        1223       STA     CHAR
   FE0E A205        1224       LDX     #5H
   FE10             1225 GEK0
   FE10 BD0353      1226       LDA     DISBUF,X
   FE13 46FD        1227       LSR     CHAR
   FE15 9004        1228       BCC     GEK1
   FE17 0980        1229       ORA     #80H    ;HAY
   FE19 D002        1230       BNE     GEK2
   FE1B             1231 GEK1
   FE1B 297F        1232       AND     #7FH    ;NO HAY
   FE1D             1233 GEK2
   FE1D 9D0353      1234       STA     DISBUF,X
   FE20 CA          1235       DEX
   FE21 10ED        1236       BPL     GEK0    ;HASTA FINAL
   FE23 20FE4D      1237       JSR     REFDIS  ;SACA DISPLAY Y MIRA TECLADO
   FE26 B007        1238       BCS     GEK3    ;PULSADA
   FE28             1239 GEKNT
   FE28 A900        1240       LDA     #0H
   FE2A 8D0364      1241       STA     ULTEC   ;BORRA FLAG
   FE2D             1242 GEKNT1
   FE2D 18          1243       CLC
   FE2E 60          1244       RTS
   FE2F             1245 GEK3
   FE2F CD0364      1246       CMP     ULTEC   ;SIGUE PULSADA?
   FE32 F0F9        1247       BEQ     GEKNT1  ;SI, IGNORA
   FE34 AE0364      1248       LDX     ULTEC   ;SE HABIA SOLTADO?
   FE37 D0F4        1249       BNE     GEKNT1  ;NO, IGNORA
   FE39 8D0364      1250       STA     ULTEC   ;SI SALVA NUEVA
   FE3C 20FE4D      1251       JSR     REFDIS  ;REPITE PROCESO
   FE3F 90E7        1252       BCC     GEKNT   ;NO SE MANTIENE
   FE41 CD0364      1253       CMP     ULTEC   ;LA MISMA?
   FE44 D0E7        1254       BNE     GEKNT1  ;NO, IGNORA
```

```
POSICION CODIGO     LINEA   CODIGO FUENTE

   FE46 38          1255       SEC          ;FLAG DE HAY TECLA
   FE47 AD0364      1256       LDA    ULTEC
   FE4A 297F        1257       AND    #7FH  ;QUITA FLAG
   FE4C 60          1258       RTS          ;Y SALIDA
                    1259
                    1260            ;REFRESCO DEL DISPLAY
                    1261
   FE4D             1262 REFDIS
   FE4D ADA000      1263       LDA    RAIORB
   FE50 29F0        1264       AND    #0F0H
   FE52 0908        1265       ORA    #8H
   FE54 A8          1266       TAY          ;ULTIMO DIGITO
   FE55 A205        1267       LDX    #5H   ;NUMERO DE DIGITOS
   FE57 A9FF        1268       LDA    #0FFH
   FE59 8DA003      1269       STA    RAIODRA ;A SALIDAS
   FE5C             1270 REDI1
   FE5C A900        1271       LDA    #0H
   FE5E 8DA001      1272       STA    RAIORA  ;BORRADO
   FE61 8CA000      1273       STY    RAIORB  ;SELECCIONA DIGITO
   FE64 BD0353      1274       LDA    DISBUF,X      ;TOMA CARACTER
   FE67 8DA001      1275       STA    RAIORA  ;SACA SEGEMENTOS
   FE6A 8A          1276       TXA
   FE6B 48          1277       PHA
   FE6C 20FD83      1278       JSR    DELAY
   FE6F 68          1279       PLA
   FE70 AA          1280       TAX
   FE71 88          1281       DEY
   FE72 CA          1282       DEX
   FE73 10E7        1283       BPL    REDI1   ;REPITE
   FE75 8CA000      1284       STY    RAIORB  ;DESELECCIONA
   FE78 A900        1285       LDA    #0H
   FE7A 8DA003      1286       STA    RAIODRA ;A ENTRADAS
                    1287
                    1288            ;SCANNER DE TECLADO
                    1289            ;SALIDA CY=1:TECLA PULSADA
                    1290
   FE7D             1291 SCAN
   FE7D A202        1292       LDX    #2H
   FE7F ADA000      1293       LDA    RAIORB
   FE82 29F0        1294       AND    #0F0H
   FE84 0902        1295       ORA    #2H
   FE86 A8          1296       TAY
   FE87             1297 SCAN1
   FE87 8CA000      1298       STY    RAIORB  ;SELECCIONA COLUMNA
   FE8A ADA001      1299       LDA    RAIORA
   FE8D 49FF        1300       EOR    #0FFH
   FE8F D006        1301       BNE    SCANH   ;HAY TECLA
   FE91 88          1302       DEY
   FE92 CA          1303       DEX
   FE93 10F2        1304       BPL    SCAN1
   FE95 18          1305       CLC
   FE96 60          1306       RTS    ;NO HAY TECLA
                    1307
                    1308            ;HAY TECLA, DECODIFICA
                    1309
   FE97             1310 SCANH
   FE97 48          1311       PHA    ;SALVA BIT
```

```
POSICION CODIGO     LINEA   CODIGO FUENTE

    FE98 8A         1312        TXA
    FE99 0A         1313        ASL     A
    FE9A 0A         1314        ASL     A
    FE9B 0A         1315        ASL     A
    FE9C AA         1316        TAX
    FE9D 68         1317        PLA
    FE9E            1318 SCAN2
    FE9E 4A         1319        LSR     A       ;OBTEN BIT
    FE9F B003       1320        BCS     SCANF   ;FINAL
    FEA1 E8         1321        INX
    FEA2 D0FA       1322        BNE     SCAN2   ;SIEMPRE
    FEA4            1323 SCANF
    FEA4 8A         1324        TXA             ;TOMA CODIGO
    FEA5 0980       1325        ORA     #80H    ;PON FLAG
    FEA7 38         1326        SEC
    FEA8 60         1327        RTS
                    1328
                    1329                ;ENTRADA DE IRQ/BRK
                    1330
    FEA9            1331 IRQBRK
    FEA9 8D036C     1332        STA     AR
    FEAC 68         1333        PLA
    FEAD 48         1334        PHA             ;OBTEN STATUS
    FEAE 2910       1335        AND     #10H    ;BRK
    FEB0 D006       1336        BNE     BRK
    FEB2 AD036C     1337        LDA     AR
    FEB5 6C0378     1338        JMP     (IRQUS) ;A IRQ DE USUSARIO
    FEB8            1339 BRK
    FEB8 AD036C     1340        LDA     AR
    FEBB 6C0376     1341        JMP     (BRKUS) ;A BRK DE USUSARIO
                    1342
                    1343                ;ENTRADA DE NMI
                    1344                ;SI PROCEDE DE STOP, VE A MONITR
                    1345                ;SI DE ELEMENTO EXTERNO A USUARIO
                    1346
    FEBE            1347 NMIENT
    FEBE 2CA000     1348        BIT     RAIORB  ;MIRA SI PROC. DE BRK LOCAL
    FEC1 3003       1349        BMI     NMISTP
    FEC3 6C037A     1350        JMP     (NMIUS) ;A NMI DE USUSARIO
                    1351
                    1352                        ;ES TECLA DE STOP
                    1353
    FEC6            1354 NMISTP
    FEC6 38         1355        SEC
    FEC7 2CFEE2     1356        BIT     >NMISDA
    FECA 20FF1C     1357        JSR     SAVREG  ; SALVA LOS REGISTROS
    FECD 20FD83     1358        JSR     DELAY   ;ANTI-REBOT. DE TEC. STOP
    FED0 ADA000     1359        LDA     RAIORB  ; PULSA LINEA DE CLEAR
    FED3 48         1360        PHA
    FED4 29F0       1361        AND     #0F0H
    FED6 0909       1362        ORA     #9H
    FED8 8DA000     1363        STA     RAIORB
    FEDB 68         1364        PLA
    FEDC 8DA000     1365        STA     RAIORB
    FEDF 4CF7D0     1366        JMP     MONITR  ;A MONITOR
    FEE2            1367 NMISDA
    FEE2 40         1368        BYTE    40H
```

```
POSICION CODIGO     LINEA   CODIGO FUENTE

                    1369
                    1370
                    1371            ;***************************************
                    1372            ;ENTRADA DE INTERRUPCIONES DEL SISTEMA
                    1373            ;***************************************
                    1374
                    1375            ;SACA PC E INDICADORES A DISPLAY
                    1376            ;BRK
                    1377
FEE3                1378 BRKSIS
FEE3 18             1379        CLC
FEE4 B8             1380        CLV
FEE5 20FF1C         1381        JSR     SAVREG
FEE8 A90F           1382        LDA     #0FH
FEEA D021           1383        BNE     INTIND
                    1384
                    1385
                    1386            ;IRQ
                    1387
                    1388
FEEC                1389 IRQSIS
FEEC 38             1390        SEC
FEED B8             1391        CLV
FEEE 20FF1C         1392        JSR     SAVREG
FEF1 A910           1393        LDA     #10H
FEF3 D018           1394        BNE INTIND
                    1395
                    1396            ;NMI
                    1397
FEF5                1398 NMISIS
FEF5 38             1399        SEC
FEF6 B8             1400        CLV
FEF7 20FF1C         1401        JSR     SAVREG
FEFA AD0369         1402        LDA     PCLR
FEFD 85FE           1403        STA     CURAD
FEFF AD036A         1404        LDA     PCHR
FF02 85FF           1405        STA   . CURAD+1
FF04 A000           1406        LDY     #0H
FF06 B1FE           1407        LDA     [CURAD],Y
FF08 20FD32         1408        JSR     OUTDAT  ;SACA DATO
FF0B D003           1409        BNE     GAT5
                    1410
                    1411            ;SACA PC Y TEXTO DE INTERRUPCION
                    1412
FF0D                1413 INTIND
FF0D 20FD65         1414        JSR     CLRIND  ; SACA TEXTO
FF10                1415 GAT5
FF10 AD0369         1416        LDA     PCLR
FF13 AC036A         1417        LDY     PCHR
FF16 20FD22         1418        JSR     OUTADD  ;SACA PC
FF19 4CF81D         1419        JMP     STRT3   ;A ESPERA
                    1420
                    1421            ;SALVA LOS REGISTROS SALVO SI STOP EN MONITOR
                    1422
FF1C                1423 SAVREG
FF1C 8D035B         1424        STA     SCR1
FF1F 8E035C         1425        STX     SCR2
```

```
POSICION CODIGO     LINEA    CODIGO FUENTE

    FF22 8C035D     1426       STY     SCR3
    FF25 BA         1427       TSX
    FF26 D8         1428       CLD
    FF27 08         1429       PHP
                    1430                ;SALVA BITS
    FF28 BD0104     1431       LDA     104H,X
    FF2B 69FF       1432       ADC     #0FFH
    FF2D A8         1433       TAY
    FF2E BD0105     1434       LDA     105H,X
    FF31 69FF       1435       ADC     #0FFH
                    1436
    FF33 28         1437       PLP
    FF34 5007       1438       BVC     SAVRE1  ;NO STOP, SALVA
    FF36 C9F0       1439       CMP     #0F0H
    FF38 9003       1440       BCC     SAVRE1  ; NO ,SALVA
    FF3A 4CFF70     1441       JMP     SAVRES  ;SI, ABORTA OPERACION
    FF3D            1442 SAVRE1
    FF3D 8D036A     1443       STA     PCHR
    FF40 8C0369     1444       STY     PCLR
    FF43 BD0103     1445       LDA     103H,X
    FF46 8D036F     1446       STA     PR
    FF49 BD0102     1447       LDA     102H,X
    FF4C 9D0105     1448       STA     105H,X
    FF4F BD0101     1449       LDA     101H,X
    FF52 9D0104     1450       STA     104H,X
    FF55 E8         1451       INX
    FF56 E8         1452       INX
    FF57 E8         1453       INX
    FF58 9A         1454       TXS
    FF59 E8         1455       INX
    FF5A E8         1456       INX
    FF5B 8E036B     1457       STX     SR
    FF5E AD035B     1458       LDA     SCR1
    FF61 8D036C     1459       STA     AR
    FF64 AD035C     1460       LDA     SCR2
    FF67 8D036D     1461       STA     XR      ;SALVA LOS REGISTROS DE TEMP.
    FF6A AD035D     1462       LDA     SCR3        .
    FF6D 8D036E     1463       STA     YR
    FF70            1464 SAVRES
    FF70 60         1465       RTS
                    1466
                    1467                ;ENTRADA DE RESET
                    1468
    FF71            1469 RESET
    FF71 A2FF       1470       LDX     #0FFH
    FF73 9A         1471       TXS
    FF74 D8         1472       CLD
                    1473
                    1474       ;INICIALIZA 65C22 I/O SISTEMA
                    1475
    FF75 20F127     1476       JSR     M_PATCH2
    FF78 4CFF83     1477       JMP     RES0    ;OBLIT PER SALVAR L'ORG
                    1478
                    1479
                    1480       ;INICIALIZA     SCRATCH
                    1481
                    1482       ORG     0FF83H
```

```
POSICION CODIGO     LINEA   CODIGO FUENTE

    FF83            1483 RES0
                    1484
    FF83 A21D       1485        LDX     #29
    FF85            1486 RES1
    FF85 BDFFD3     1487        LDA     >INTAB-1,X
    FF88 9D0364     1488        STA     BRKFLG-1,X
    FF8B 8E0352     1489        STX     REST
    FF8E CA         1490        DEX
    FF8F D0F4       1491        BNE     RES1
    FF91 4CF7D0     1492        JMP     MONITR  ;A BUCLE PRINCIPAL
                    1493
                    1494                ; SALTOS INDIRECTOS EN INTERRUPCIONES
                    1495
    FF94            1496 IRQJMP
    FF94 6C037C     1497        JMP     [IRQVEC]
    FF97            1498 NMIJMP
    FF97 6C037E     1499        JMP     [NMIVEC]
                    1500
                    1501                ;TABLAS
                    1502                ;CODIGO DE SEGMENTOS
                    1503
    FF9A            1504 SEGMTB
    FF9A 3F065B4F   1505        BYTE    3FH,06H,5BH,4FH ;0 1 2 3
    FF9E 666D7D07   1506        BYTE    66H,6DH,7DH,07H ;4 5 6 7
    FFA2 7F6F777C   1507        BYTE    7FH,6FH,77H,7CH ;8 9 A B
    FFA6 395E7971   1508        BYTE    39H,5EH,79H,71H ;C D E F
                    1509
                    1510
                    1511                ;TEXTOS SOBRE DISPLAY
                    1512
    FFAA            1513 MENTAB
    FFAA 7358       1514        BYTE    73H,58H ; 0 'Pc'
    FFAC 506D       1515        BYTE    50H,6DH ; 1 'rs'
    FFAE 7758       1516        BYTE    77H,58H ; 2 'Ac'
    FFB0 5064       1517        BYTE    50H,64H ; 3 'rx'
    FFB2 506E       1518        BYTE    50H,6EH ; 4 'rY'
    FFB4 5073       1519        BYTE    50H,73H ; 5 'rP'
    FFB6 5F5E       1520        BYTE    5FH,5EH ; 6 'ad'
    FFB8 3D5C       1521        BYTE    3DH,5CH ; 7 'Go'
    FFBA 7C50       1522        BYTE    7CH,50H ; 8 'br'
    FFBC 7C53       1523        BYTE    7CH,53H ; 9 'b?'
    FFBE 3E5E       1524        BYTE    3EH,5EH ; 10 'Vd'
    FFC0 3E77       1525        BYTE    3EH,77H ; 11 'VA'
    FFC2 105E       1526        BYTE    10H,5EH ; 12 'id'
    FFC4 5E10       1527        BYTE    5EH,10H ; 13 'di'
    FFC6 5E71       1528        BYTE    5EH,71H ; 14 'dF'
    FFC8 7C7B       1529        BYTE    7CH,7BH ; 15 'be'
    FFCA 107B       1530        BYTE    10H,7BH ; 16 'ie'
    FFCC 547B       1531        BYTE    54H,7BH ; 17 'ne'
    FFCE 5C54       1532        BYTE    5CH,54H ; 18 'on'
    FFD0 5C71       1533        BYTE    5CH,71H ; 19 'oF'
    FFD2 585F       1534        BYTE    58H,5FH ; 20 'ca'
                    1535
                    1536
                    1537                ;INICIALIZACION DE SCRATCH
                    1538
    FFD4 00000000   1539 INTAB BYTE     00H,00H,00H,00H
```

```
POSICION CODIGO      LINEA   CODIGO FUENTE

    FFD8 F7D0        1540      WORD    MONITR          ;P.C.
    FFDA FF00000000  1541      BYTE    0FFH,00H,00H,00H,00H
    FFDF 043D335A    1542      BYTE    04H,3DH,33H,5AH         ;DARTOS DE CASSETE
    FFE3 F7D3        1543      WORD    ERROR           ;VECTOR AMPLIACION DE FUNCIONES
    FFE5 FEE3        1544      WORD    BRKSIS          ;VECTORES DE USUARIO
    FFE7 FEEC        1545      WORD    IROSIS          ;
    FFE9 FEF5        1546      WORD    NMISIS          ;
    FFEB FEA9        1547      WORD    IROBRK          ;VECTORESA SECUNDARIOS
    FFED FEBE        1548      WORD    NMIENT
                     1549
                     1550              ;VECTORES DE LA CPU
                     1551
                     1552      ORG     0FFFAH
    FFFA FF97        1553      WORD    NMIJMP
    FFFC FF71        1554      WORD    RESET
    FFFE FF94        1555      WORD    IRQJMP
                     1556
                     1557
                     1558
                     1559
                     1560
                     1561
                     1562      END
```

^$

PROG.: Monitor TABLA DE REFERENCIAS D31

LINEA	SIMBOLO	TIPO	REFERENCIASS
975	AJCK	A	816, 908
982	AJCK1	A	980
63	AR	A	388,1332,1337,1340,1459
32	BASE	A	
622	BASOF	A	620
453	BC1	A	449
450	BCER	A	471
1339	BRK	A	1336
59	BRKADH	A	437, 455, 475
58	BRKADL	A	435, 453, 474
57	BRKDAT	A	433, 458
56	BRKFLG	A	406, 438, 448, 462, 470,1488
1378	BRKSIS	A	1544
72	BRKUS	A	1341
412	BS1	A	407
432	BS2	A	426
409	BSER	A	431, 452
429	BSER1	A	461
439	BSSAL	A	
82	CHAR	A	891, 892, 912, 915, 953, 956,1223,1227
1062	CLRD1	A	1065
1058	CLRDIS	A	223, 240, 278,1072
1071	CLRIND	A	502, 569, 640, 647, 734, 744,1091,1098,1414
77	CONTA	A	111, 120
27	CPA2	A	156
25	CPB2	A	154
655	CS0	A	650
659	CS1	A	663, 665
678	CS2	A	703, 705
698	CS3	A	681, 684
83	CURAD	A	287, 289, 294, 296, 303, 326, 327, 334, 336, 339, 341, 343, 345, 346, 417, 419, 421, 424, 425, 434, 436, 454, 456, 459, 460, 572, 574, 577, 578, 582, 583, 643, 645, 670, 672, 679, 682, 700, 704, 740, 742, 797, 799, 809, 812, 821, 823, 828, 830,1403,1405,1407
1114	DELAY	A	1117,1278,1358
43	DISBUF	A	157, 159, 161, 163, 165, 167, 233, 242, 244, 245,1052,1063,1082,1084,1226,1234,1274
276	EJEC	A	267
284	EJEC0	A	279
352	EJEC1	A	280
610	EJEC10	A	593
626	EJEC11	A	612
707	EJEC12	A	629
719	EJEC13	A	710
392	EJEC2	A	354
400	EJEC3	A	394
442	EJEC4	A	402
465	EJEC5	A	444
478	EJEC6	A	466
493	EJEC7	A	480
555	EJEC8	A	495
591	EJEC9	A	557
212	ERROR	A	1153,1161,1543
272	ESPEC	A	261
1102	ESPER	A	176,1092
1105	ESPER1	A	1108
71	ESPUEC	A	273

PROG.: Monitor TABLA DE REFERENCIAS D32

LINEA	SIMBOLO	TIPO	REFERENCIASS
1151	GAT	A	1149
1159	GAT2	A	1157
1183	GAT3	A	1181
1415	GAT5	A	1409
1225	GEK0	A	1236
1231	GEK1	A	1228
1233	GEK2	A	1230
1245	GEK3	A	1238
1239	GEKNT	A	1252
1242	GEKNT1	A	1247,1249,1254
1134	GETADD	A	292, 365, 641, 648, 735, 745
1127	GETADE	A	415, 563, 570, 599
947	GETC1	A	955
952	GETC2	A	949
959	GETC3	A	946, 961
969	GETCCC	A	788, 789, 790, 791, 796, 798, 800, 802
942	GETCSC	A	761, 772, 808, 841, 844, 897, 970
994	GETCTR	A	945, 948, 950, 988, 989
996	GETCTR1	A	1000
1221	GETKEY	A	255, 307,1178
1146	GETP1	A	1130
1154	GETP2	A	1137,1147
1168	GETP3	A	514, 527,1155
1173	GETP31	A	1171
1177	GETP32	A	1179,1188,1191
1189	GETP33	A	1186
1143	GETPAR	A	636, 728
379	GO	A	367, 399
381	GO1	A	378
68	HSBDRY	A	126
49	ID	A	638, 668, 730, 774, 776
1078	INDIC	A	286, 359, 414, 473, 562, 598, 635, 727
1539	INTAB	A	1487
1413	INTIND	A	1383,1394
1331	IRQBRK	A	1547
1496	IRQJMP	A	1555
1389	IRQSIS	A	1545
73	IRQUS	A	1338
75	IRQVEC	A	1497
758	LC1	A	765
847	LC10	A	843
760	LC2	A	766
768	LC21	A	763
787	LC3	A	779
794	LC4	A	775, 777, 783
807	LC5	A	792, 829, 831
815	LC6	A	811
822	LC7	A	820
827	LC8	A	819, 824
840	LC9	A	837
724	LCE	A	715
849	LCER	A	826, 839, 965
835	LCF	A	814
752	LCL	A	732, 747, 858
853	LCOR	A	846

PROG.: Monitor TABLA DE REFERENCIAS D33

LINEA	SIMBOLO	TIPO	REFERENCIASS
859	LCOU	A	694, 857
855	LCSA	A	851
170	LED	A	179
54	LODIVER	A	725, 769
116	LOOPA	A	121
300	M1	A	310, 330, 348
306	M1A	A	351
307	M1B	A	311, 318
312	M1C	A	308
317	M2	A	314
331	M3	A	323, 328
338	M4	A	333
342	M5	A	340
344	M6	A	335, 337
91	MARCA	A	
1513	MENTAB	A	1081,1083
349	MHEX	A	320
50	MODEC	A	513, 526, 771, 780, 782, 818, 856, 885, 887, 960,1129,1136,1145,1170,1185
206	MONITR	A	258, 370, 372,1366,1492,1540
109	M_PATCH1	A	1002
132	M_PATCH2	A	1476
152	M_PATCH3	A	224
1347	NMIENT	A	1548
1498	NMIJMP	A	1553
1367	NMISDA	A	1356
1398	NMISIS	A	1546
1354	NMISTP	A	1349
74	NMIUS	A	1350
76	NMIVEC	A	1499
230	NORM1	A	221
219	NORMAL	A	226
1096	OFF	A	463, 492, 623
53	OLD	A	999,1001
1089	ON	A	490, 621
1037	OUTACC	A	1017,1019,1027
1048	OUTACH	A	1043
1013	OUTADD	A	291, 347, 364, 476, 511,1187,1418
1025	OUTADH	A	524,1190
919	OUTC1	A	921
927	OUTC2	A	938
929	OUTC3	A	931
914	OUTCBI	A	933
907	OUTCCC	A	671, 673, 675, 677, 701
936	OUTCN	A	922
909	OUTCSC	A	661, 667, 669, 688, 690, 692
1031	OUTDAT	A	248, 306,1408
26	PA2	A	180
1196	PACK	A	350,1184
1202	PACK1	A	1198,1207
45	PARH	A	290, 295, 363, 376, 418, 510, 518, 566, 573, 605, 608, 644, 651, 676, 683, 741, 748, 803, 813 1205,1210
44	PARL	A	288, 293, 305, 324, 361, 374, 416, 508, 516, 523, 530, 564, 571, 607, 637, 642, 649, 674, 680 737, 746, 801, 810,1200,1204,1208,1209
24	PB2	A	172, 182
61	PCHR	A	362, 377, 382, 509, 519,1404,1417,1443

PROG.: Monitor TABLA DE REFERENCIAS D34

LINEA	SIMBOLO	TIPO	REFERENCIASS
60	PCLR	A	360, 375, 384, 507, 517,1402,1416,1444
66	PR	A	386,1446
51	PRIMER	A	304, 309, 322, 366, 658, 664,1148,1176,1197,1201
491	PROOF	A	489
52	PUNTO	A	254, 298, 604,1174,1222
521	R1	A	506
532	R2	A	520
538	R3	A	535
547	R4	A	541
551	R5	A	544, 546, 549
19	RAIDACR	A	141
13	RAIODRA	A	135, 757,1269,1286
14	RAIODRB	A	139
17	RAIOIER	A	144
18	RAIOIFR	A	145
20	RAIOPCR	A	142
11	RAIORA	A	773, 884, 902,1272,1275,1299
12	RAIORB	A	137, 484, 486, 616, 618, 755, 861, 864, 875, 877, 913, 917, 924, 997,1263,1273,1284,1293,1298, 1359,1363,1365
16	RAIOTHC	A	112, 124
15	RAIOTLC	A	115, 122
500	RB	A	553
1270	REDI1	A	1283
1262	RE[illegible]IS	A	1106,1237,1251
1483	RES0	A	1477
1486	RES1	A	1491
1469	RESET	A	1554
42	REST	A	220, 225,1489
222	SALUDO	A	
1442	SAVRE1	A	1438,1440
1423	SAVREG	A	1357,1381,1392,1401
1464	SAVRES	A	1441
1291	SCAN	A	
1297	SCAN1	A	1304
1318	SCAN2	A	1322
1323	SCANF	A	1320
1310	SCANH	A	1301
633	SCE	A	628
46	SCR1	A	501, 503, 529, 539, 565, 576, 689, 842, 872, 978, 979,1104,1107,1424,1458
47	SCR2	A	567, 581, 691, 845, 873, 981,1425,1460
48	SCR3	A	911, 934, 943, 957,1426,1462
1504	SEGMTB	A	1051
62	SR	A	215, 379, 522, 531,1457
215	START	A	208, 262, 268
870	STARTC	A	656, 753
266	STR3	A	260
233	STRT1	A	235
238	STRT2	A	218
247	STRT21	A	609
252	STRT3	A	236, 256, 477,1419
889	SY1	A	894, 899
896	SY2	A	901
987	SYNBIT	A	890
882	SYNC	A	759
67	TAPDEL	A	657

APENDICE E

MANTENIMIENTO Y VERIFICACION

Para la reposición de los fusibles ver la práctica del capítulo 1.

Proceder al seguimiento de los puntos indicados. Si la verificación falla en algún punto, el equipo presenta alguna anomalía y debe ser remitido a un servicio técnico autorizado para su reposición.

- Comprobar la alimentación: deberá ser de +5 vóltios +/-5 %.

- Comprobar la frecuencia del cristal que deberá ser de 1 MHz.

- Comprobar las líneas /RES, /IRQ, /NMI. Todas ellas deberán permanecer en estado alto ('1'). Mientras permanezca pulsada la tecla [RESET], la línea /RES deberá permanecer en estado bajo '0'.

- Pulsar la tecla [RESET]. Despues de efectuar un test de los LEDs y de mostrar, momentaneamente, en el display el modelo de Microinstructor, se encienden los segmentos centrales en todos los dígitos del Microinstructor.

- Pulsar la tecla [RUN/STEP]. El LED amarillo deberá encenderse y se apagará al volver a pulsarla. También deberá apagarse al pulsar [RESET].

- Pulsar [STOP/BRK]. Se encenderán los segmentos centrales de todos los dígitos lo mismo que al pulsar [RESET] pero sin hacer el test inicial.

- Pulsar [CAL/D]. En el display aparecerán puntos en los cuatro dígitos correspondientes a DIRECCION y "cA" en los dígitos que corresponden a DATO.

- Pulsar [STOP] para devolver el equipo a reposo.

- Pulsar [WRP/8]. En el display aparecerá durante un tiempo el texto "oF" en los dígitos de DATO. Al volver a pulsarla aparecerá, tambien por unos instantes, el texto "on".

- Pulsar [SET/B]. En el display aparecerán 4 puntos en DIRECCION y "br" en DATO. Pulsar [STOP] para volver al estado de reposo.

- Pulsar [STV/3]. En el display aparecerán 4 puntos en DIRECCION y "Ud" en DATO.

- Pulsar [SAVE]. En el display aparecerán dos puntos en los dígitos de dirección y "id" en los dígitos de DATO.

- Pulsar [LOAD]. Igual que la función anterior.

- Pulsar [VER]. Igual que la función anterior.

- Pulsar MOT. En el display aparecerá el texto "on" u "oF" durante un tiempo.

PROG.: Monitor TABLA DE REFERENCIAS D35

LINEA	SIMBOLO	TIPO	REFERENCIASS
70	TAPET2	A	928
69	TAPETI	A	918
78	TEMPO	A	114, 118
34	TRAMH	A	
33	TRAML	A	
55	ULTEC	A	1241,1246,1248,1250,1253,1256
587	VECER	A	579, 584
64	XR	A	389,1461
65	YR	A	390,1463

APENDICE F

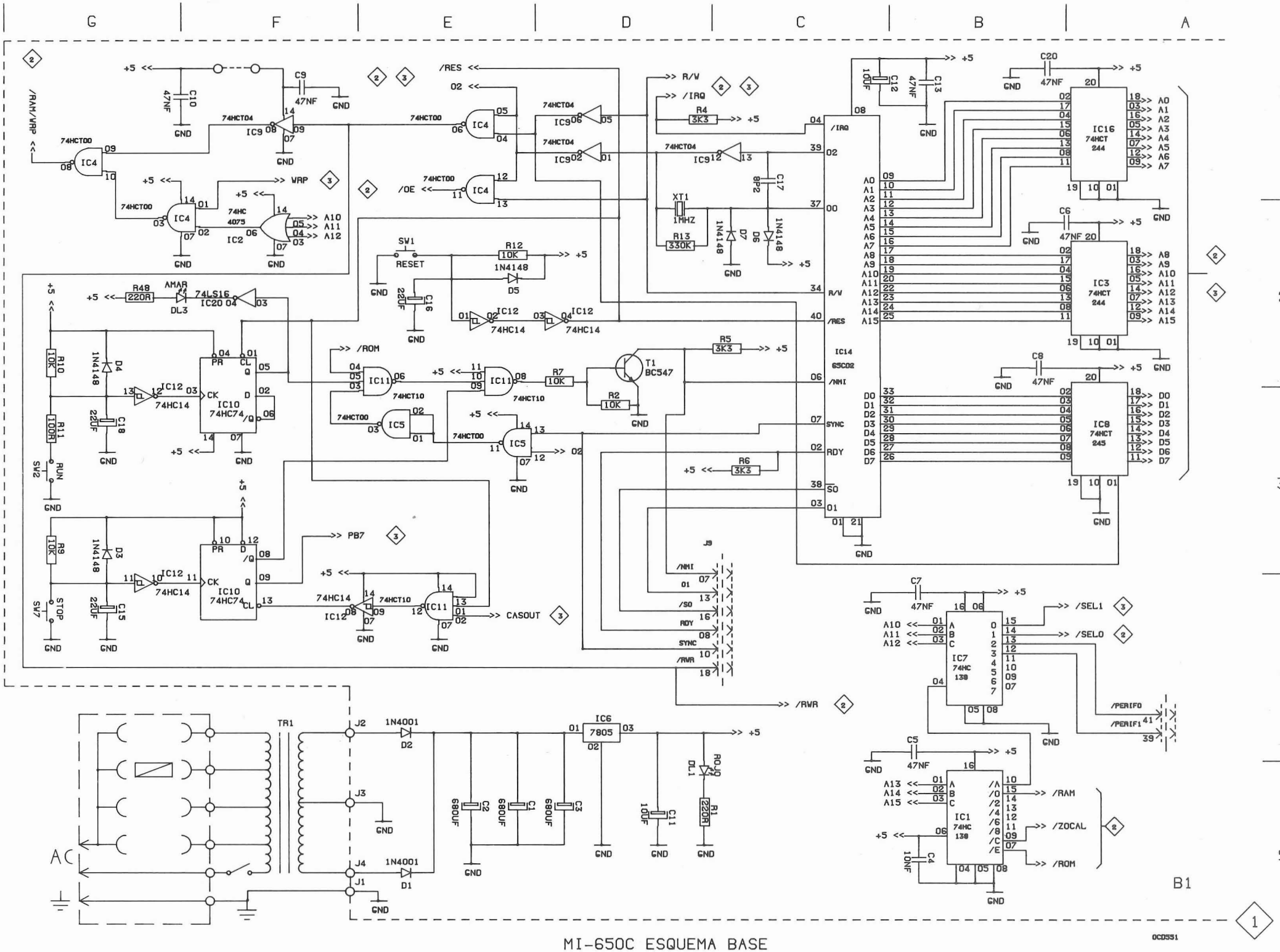
ESQUEMAS

A continuación se adjuntan los esquemas del Microinstructor MI-650C.

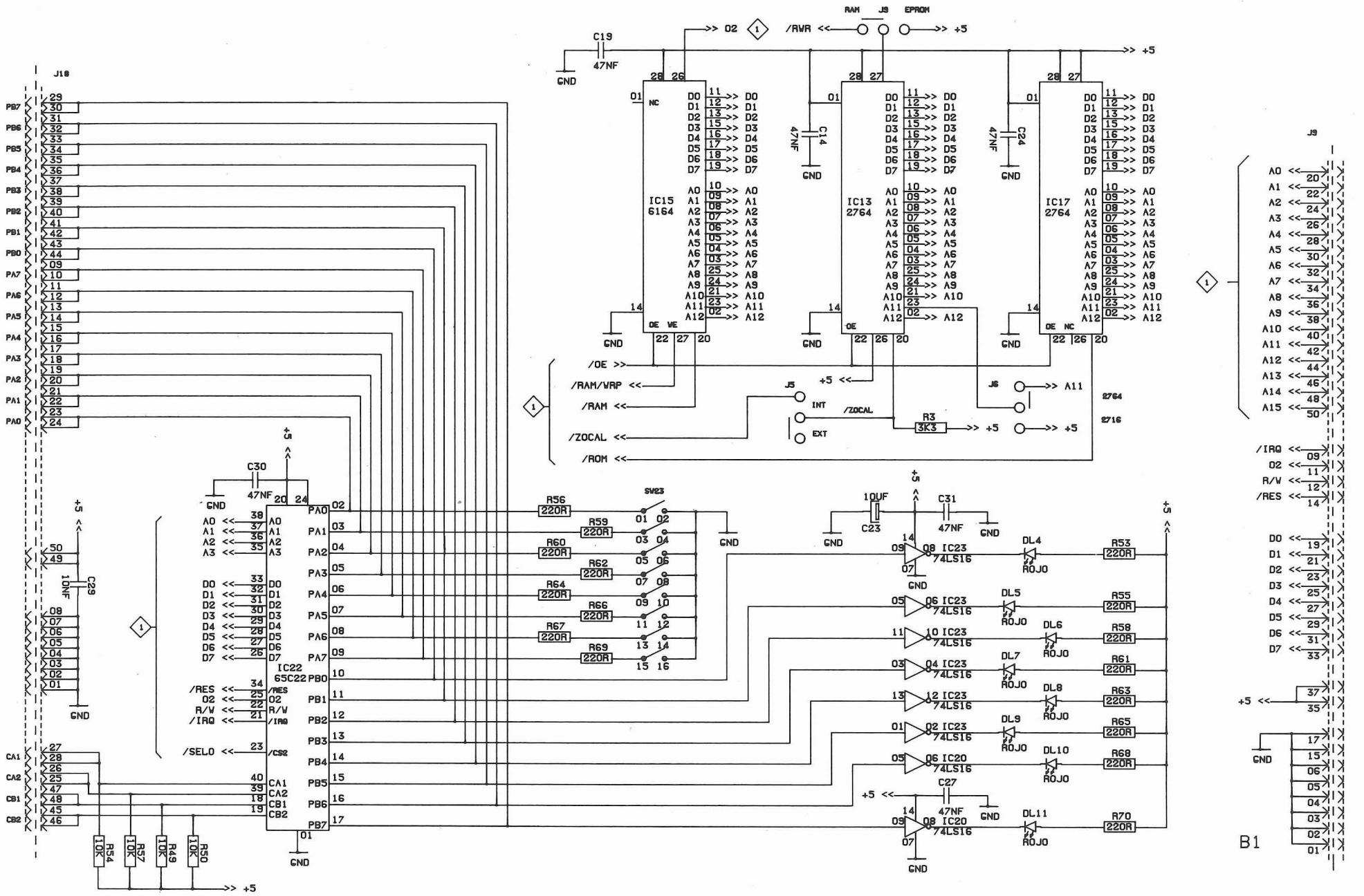
El primero muestra las conexiones entre la CPU y elementos a ella asociados, los buffers de los buses de datos y direcciones, los decodificadores y la fuentes de alimentación.

En el segundo esquema se muestran las conexiones de los tres zócalos de memoria y de la VIA 65C22 auxiliar.

El último esquema muestra las conexiones de la VIA 65C22 local que controla el teclado, el display y la grabación en cassette.

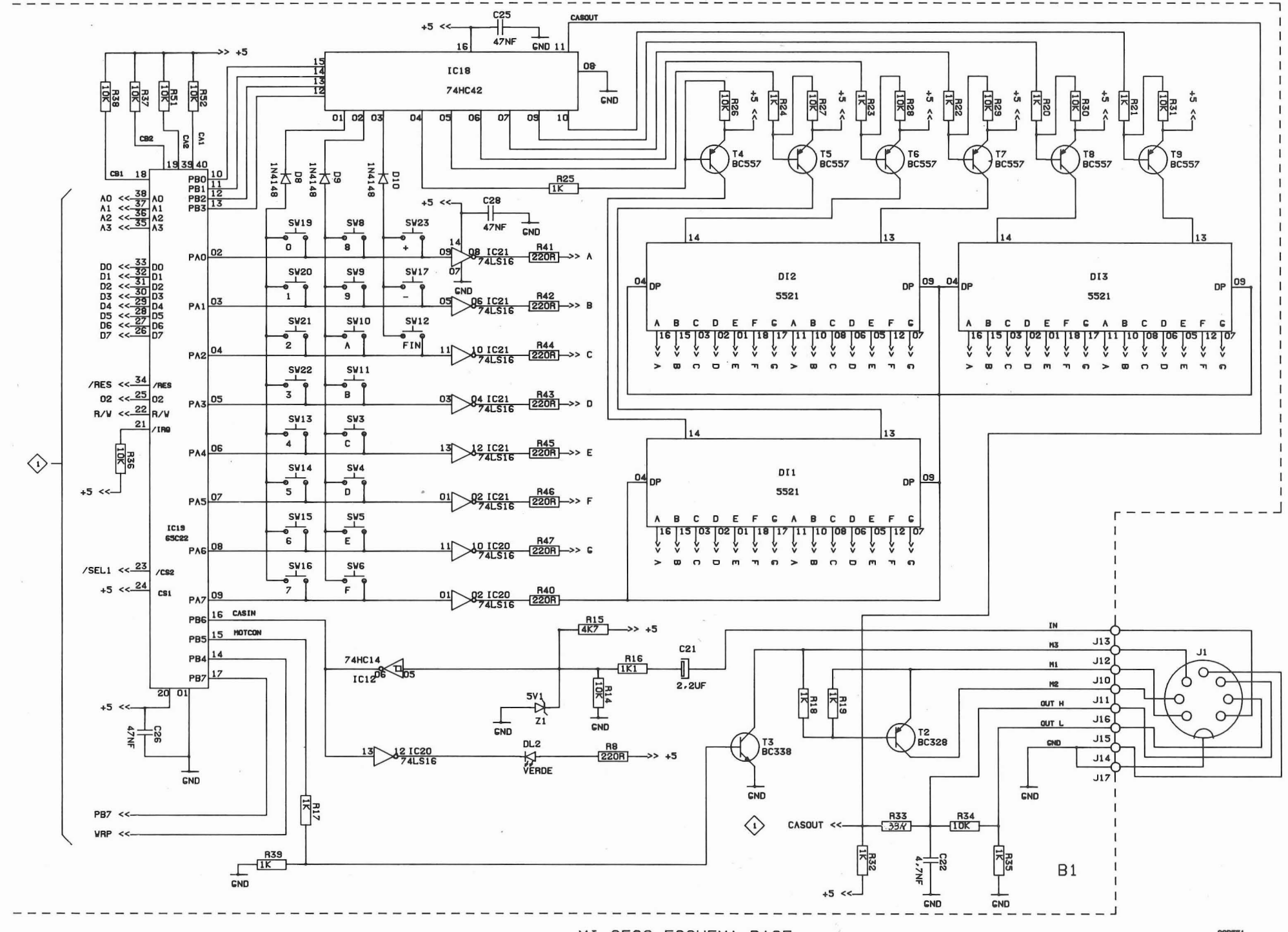


MI-650C ESQUEMA BASE



MI-650C ESQUEMA BASE

0C0551


G
F
E
D
C
B
A
1
2
3
4
5
IC18
74HC42
C25
47NF
C28
47NF
CASOUT
R25
1K
T4 BC557
T5 BC557
T6 BC557
T7 BC557
T8 BC557
T9 BC557
DI1 5521
DI2 5521
DI3 5521
IC19 65C22
IC21 74LS16
IC20 74LS16
SW19
SW8
SW23
SW20
SW9
SW17
SW21
SW10
SW12
SW22
SW11
SW13
SW3
SW14
SW4
SW15
SW5
SW16
SW6
1N4148
D8
D9
D10
R41 220R
R42 220R
R44 220R
R43 220R
R45 220R
R46 220R
R47 220R
R40 220R
R15 4K7
R16 1K1
C21 2,2UF
74HC14
IC12
5V1
Z1
R14 10K
R8 220R
DL2
VERDE
T3 BC338
T2 BC328
R18 1K
R19 1K
R17 1K
R39 1K
R33 33K
R34 10K
R32 1K
R35 1K
C22 4,7NF
C26 47NF
R35 10K
J1
IN
M3
M1
M2
OUT H
OUT L
GND
B1
MI-650C ESQUEMA BASE
OCD551