MOS TECHNOLOGY, INC.
NORRISTOWN, PA. 19401

MICROCOMPUTERS

MCS6500 MICROPROCESSOR SOFTWARE SUPPORT

MOS TECHNOLOGY'S support software is now available on United Computing Systems time-sharing service. The package available provides online support to assist the microcomputer applications design engineer or programmer in program development for the MCS650X microcomputer family.

TO USE MOS TECHNOLOGY SUPPORT SOFTWARE:

1. Contact your local USC sales representative and request MOS TECHNOLOGY'S MCS650X Software System under user catalog number M490. Also request the UCS System Guide and the UNIEDIT manuals.
2. Order your copy of the MCS6500 Microprocessor Hardware, Programming, Simulator, And Cross Assembler manuals from:
   MOS Technology Inc., 950 Rittenhouse Rd.,
   Norristown, Pa. 19401
3. Dial the appropriate telephone number supplied by your USC sales representative, sign on with your terminal, and begin entering your MCS650X microprocessor program.

THE SOFTWARE SUPPORT PACKAGE CONSISTS OF:

–MOS/*** - A text file containing the latest bulletins regarding MOS TECHNOLOGY Microprocessor Software.

–ASM/*** - An interactive program which builds the job control language required to submit your source code to ASM650X.

ASM650X MCS650X Cross Assembler: the Cross Assembler is a program which translates a mnemonic or symbolic form of a computer program to machine language.

–SIM/*** - An interactive program which builds the job control language required to submit your simulator command file to SIM650X.

SIM650X - MCS650X Simulator. The simulator uses the command file to simulate execution of the machine language instructions created by the cross assembler in the MCS650X microprocessor.

–DMP/*** - ROM dump program. This program creates an output file of machine language instructions in a format suitable for MOS microcomputer loader programs.

The sample program shown in this brochure uses the UCS time-sharing system to give the user an overview of the procedure to be followed for using MOS TECHNOLOGY'S support software.

In brief the procedure to be followed is:

1. Create a source file using the time-sharing editor and save the file.
2. Submit the source file to the Cross Assembler by answering the questions asked by -ASM/***.
3. When the Cross Assembler run is completed list the output file to obtain a listing of the assembled program.
4. Create a file of simulator commands using the time-sharing editor and save the file.
5. Submit the simulator command file and the machine language file to the simulator by answering the questions asked by -SIM/***.
6. When the simulator run is completed list the output file to obtain the results of the program simulation.
7. Obtain a ROM dump object tape by answering the questions asked by -DMP/***.

1. CREATE A SOURCE FILE.

```
|>p|>`p|pT63

UCS  11/19/75. 09.10.41.  I150
USER NUMBER:  M490010,EXAMPLE

 GENERAL:
 MOS TECHNOLOGY 650X MICROPROCESSOR SOFTWARE.
 FOR THE LATEST INFORMATION TYPE  -MOS/***

 MESSAGE(S) COMPLETE.

     0.013 /     0.038 /         9
READY - FOR!
-MOS/***

 11/19/75. 09.11.22.
PROGRAM   MOS

LAST UPDATED ON 11/19/75

.
BULLETINS REGARDING THE MOS TECHNOLOGY MICROPROCESSOR
SOFTWARE WILL APPEAR FROM TIME TO TIME IN THIS MANNER.

.
TO RUN THE 650X CROSS ASSEMBLER YOU MUST FIRST CREATE A
SOURCE FILE. THEN ENTER -ASM/*** TO SUBMIT YOUR SOURCE FILE
FOR BACKGROUND BATCH EXECUTION.

.
TO RUN THE 650X SIMULATOR YOU MUST FIRST CREATE A SIMULATOR
COMMAND FILE AND A CROSS ASSEMBLER INTERFACE FILE. THEN TYPE -
-SIM/*** TO SUBMIT YOUR  COMMAND FILE FOR SIMULATION.

.
THE 650X ROM DUMP PROGRAM WILL CREATE A REFORMATED FILE
SUITABLE FOR INPUT TO THE MOS MICCROCOMPUTER LOADER PROGRAMS.
YOU MUST HAVE CREATED AN INTERFACE FILE WITH THE CROSS
ASSEMBLER. TO RUN THE DUMP PROGRAM ENTER -DMP650X/***

.
THANK YOU.....MOS TECHNOLOGY
RUN COMPLETE.
NEW,SAMP4
READY - FOR!
AUT
00100 .PAGE 'MULTIPLE BYTE ADD'
00110 ;ADDITION OF TWO MULTIPLE PRECISION NUMBERS (BCD)
00150  *=0   ALLOCATE A DATA AREA IN FIRST PAGE OF MACHINE
00170 ADDR *=*+1
00190 NB=8
00200 PP *=*+NB
00210 Q *=*+NB
00220 RES *=*+NB
00270 MAIN LDX #$8F BEGIN MAIN ROUTINE TO TEST SUB. BCD.
00280  TXS   INITIALIZE STACK POINTER
00290  LDX #PP
00300  STX ADDR
00310  JSR BCD
00320  NOP
00330  JMP *-1   END OF MAIN PGM
00360  *=100      BEGIN SUBROUTINE
00370 BCD LDY #NB
00380  LDX ADDR  LOADS DATA ADDRESS
00390  CLC
00400  SED
00410 NEXT LDA NB-1,X
00420  ADC 2*NB-1,X
00430  STA 3*NB-1,X
00440  DEX
00450  DEY
00460  BNE NEXT   END OF LOOP
00470  CLD
00480  RTS
00490 ABCDEFGH NOP   THIS IS AN INTENTIONAL ERROR.
00500 .END
00510 *DEL*
SAVE
READY.
```

Enter proper response so that computer can determine your terminal's speed.

For 10 CPS enter ?63
For 15 CPS enter 863
For 30 CPS enter T63

Enter your user number and password to log on to UCS system.

Indicates FORTRAN system is ready. (FORTRAN is automatically assigned.)

Enter -MOS/*** to obtain latest bulletins.

Indicates the end of the bulletin.

Create a new file with file name "SAMP4".

Auto line number assignment.

Assembler directive to advance listing to top of page and title the page "MULTIPLE BYTE ADD".

Semicolon indicates the start of a comment field.

*= assembler directive sets the program counter.

Sets NB equal to 8.

Reserves 8 bytes of memory for the label "PP".

Start of program labeled "MAIN"

Note that there is only one space between a line number and a label. There are two or more spaces between a line number and an instruction. Comments may begin one space after the operand.

.END assembler directive defines the end of the source program.

Hitting the "ESC" key ends the auto line number assignment. The system replies "*DEL*".

SAVE is the command to save the new file just created.

2. SUBMIT TO CROSS ASSEMBLER.

```
-ASM/***

MOS TECHNOLOGY 650X CROSS ASSEMBLER SUBMITTOR

DO YOU WANT INSTRUCTIONS (YES OR NO) -- ? NO
ENTER USERNUM,PASSWORD, AND PID (IF NEEDED) --  ? M490010,EXAMPLE
DO YOU WANT TO CHANGE THE PRIORITY --  ? NO

ENTER SOURCE     FILE NAME --  ? SAMP4

SAVE  OUTPUT     FILE (YES OR NO) --  ? YES
ENTER OUTPUT     FILE NAME --  ? OUT4

SAVE  INTERFACE  FILE (YES OR NO) --  ? YES
ENTER INTERFACE  FILE NAME --  ? INT4

SAVE  ERROR      FILE (YES OR NO) --  ? YES
ENTER ERROR      FILE NAME --  ? ERR4

SAVE  DAYFILE    FILE (YES OR NO) --  ? YES
ENTER DAYFILE    FILE NAME --  ? DAY4

ENTER CONTROL    FILE NAME --  ? CON4

TO RUN ASSEMBLER TYPE --
OLD,CON4
RJE    (OR RBE)


 STOP.
OLD,CON4
READY - EXE!
RJE

 11/19/75. 09.15.45.
PROGRAM   CON4

 RJE COMPLETE,ID =  RJEDZQM
```

-ASM/*** invokes the cross assembler submittor software.

SOURCE file is the file containing the source code to be assembled.

OUTPUT file will contain the assembler listing.

INTERFACE file will contain the object code, line number and label information required by the simulator.

ERROR file will contain a listing of any errors that occur during the assembly.

DAY file is a history of steps taken by the UCS system in running your job.

CONTROL file is the file of JCL built by -ASM/*** to run your assembly.

Submits assembly job to the UCS system.

Indicates that the job has been submitted under the job name "RJEDZQM".

3. LIST OUTPUT FILE

```
OLD,OUT4
READY - EXE!
LIS

 11/19/75. 09.18.14.
PROGRAM   OUT4

+    MULTIPLE BYTE ADD                                              PAGE    1
0LINE  LOC     CODE      SOURCE
   110                   ;ADDITION OF TWO MULTIPLE PRECISION NUMBERS (BCD)
   150  0000              *=0   ALLOCATE A DATA AREA IN FIRST PAGE OF MACHINE
   170  0000             ADDR *=*+1
   190                   NB=8
   200  0001             PP *=*+NB
   210  0009             Q *=*+NB
   220  0011             RES *=*+NB
   270  0019  A2 8F      MAIN LDX #$8F BEGIN MAIN ROUTINE TO TEST SUB. BCD.
   280  001B  9A          TXS   INITIALIZE STACK POINTER
   290  001C  A2 01       LDX #PP
   300  001E  86 00       STX ADDR
   310  0020  20 64 00    JSR BCD
   320  0023  EA          NOP
   330  0024  4C 23 00    JMP *-1   END OF MAIN PGM
   360  0027              *=100      BEGIN SUBROUTINE
   370  0064  A0 08      BCD LDY #NB
   380  0066  A6 00       LDX ADDR  LOADS DATA ADDRESS
   390  0068  18          CLC
   400  0069  F8          SED
   410  006A  B5 07      NEXT LDA NB-1,X
   420  006C  75 0F       ADC 2*NB-1,X
   430  006E  95 17       STA 3*NB-1,X
   440  0070  CA          DEX
   450  0071  88          DEY
   460  0072  D0 F6       BNE NEXT   END OF LOOP
   470  0074  D8          CLD
   480  0075  60          RTS
   490  0076  EA EA EA  ABCDEFGH NOP   THIS IS AN INTENTIONAL ERROR.
 *****  ERROR ** LABEL GREATER THAN SIX CHARACTERS - NEAR COLUMN    1
   500                   .END


 END OF MOS/TECHNOLOGY 650X ASSEMBLY VERSION 4
 NUMBER OF ERRORS =   1,  NUMBER OF WARNINGS =   0
1         SYMBOL TABLE

   SYMBOL   VALUE  LINE DEFINED        CROSS-REFERENCES

   ADDR     0000          170   300  380
   BCD      0064          370   310
   MAIN     0019          270  ****
   NB       0008          190   200  210  220  370  410  420  430
   NEXT     006A          410   460
   PP       0001          200   290
   Q        0009          210  ****
   RES      0011          220  ****
RUN COMPLETE.
```

Terminal input to list the output file "OUT4".

Title created by .PAGE assembler directive.

Program counter. (Hexadecimal)

Hexadecimal instruction, data, or value.

Program counter set to hexadecimal 64 by assembler directive *=100.

Error line will also appear in the ERROR file.

The version number is changed as improvements are made to the Cross Assembler.

Note: For more detailed information refer to the MCS6500 Microprocessor Programming and Cross Assembler manuals.

4. CREATE SIMULATOR COMMANDS

```
NEW,ECSAMP1
READY - FOR!
AUTO
00100 SM 1 1 2 3 4 5 6 7 8
00110 SM 9 8 7 6 5 4 3 2 1
00120 DUMP 1 $18
00130 TRACE 0 $FFFF
00140 DO MAIN NEXT 3 .TIMES
00150 DUMP 1 $18
00160 EXIT
00170 *DEL*
SAVE
READY.
```

Create simulator command file called "ECSAMP1".

Starting at location 1 set consecutive memory locations to the specified values.

Dump the contents of memory from decimal 1 to hexadecimal 18.

Trace every instruction executed.

Begin simulated execution at label "MAIN" and continue until instruction at label "NEXT" has been executed 3 times.

EXIT terminates simulator run.

5. SUBMIT TO SIMULATOR

```
-SIM/***

MOS TECHNOLOGY 650X SIMULATOR SUBMITTOR

DO YOU WANT INSTRUCTIONS (YES OR NO) -- ? NO
ENTER USERNUM,PASSWORD, AND PID (IF NEEDED) --  ? M490010,EXAMPLE
DO YOU WANT TO CHANGE THE PRIORITY --  ? NO

ENTER COMMAND    FILE NAME --  ? ECSAMP1

ENTER INTERFACE  FILE NAME --  ? INT4

SAVE  OUTPUT     FILE (YES OR NO) --  ? YES
ENTER OUTPUT     FILE NAME --  ? EOUT4

SAVE  DAYFILE    FILE (YES OR NO) --  ? YES
ENTER DAYFILE    FILE NAME --  ? EDAY4

ENTER CONTROL    FILE NAME --  ? ECON4

TO RUN SIMULATOR TYPE --
OLD,ECON4
RJE    (OR RBE)


 STOP.
OLD,ECON4
READY - EXE!
RJE

 11/19/75. 09.23.50.
PROGRAM   ECON4

 RJE COMPLETE,ID =  RJEDZRY
```

-SIM/*** invokes the simulator submittor software.

COMMAND file is the file containing the simulator commands.

INTERFACE file is the interface file created by the cross assembler.

6. LIST SIMULATOR OUTPUT

```
OLD,EOUT4
READY - FOR!
LIST
```

Terminal commands required to list the Simulator output file.

```
 11/19/75. 09.26.05.
PROGRAM   EOUT4

1++++++ MOS TECHNOLOGY 650X MICROPROCESSOR SIMULATOR +++++


 00100 SM 1 1 2 3 4 5 6 7 8
 00110 SM 9 8 7 6 5 4 3 2 1
 00120 DUMP 1 $18
                 CONTENTS OF MEMORY LOCATION AT BASE ADDRESS PLUS........
   BASE ADDRESS   +0  +1  +2  +3  +4  +5  +6  +7  +8  +9  +A  +B  +C  +D  +E  +F
 DUMP ADDR=0000   00  01  02  03  04  05  06  07  08  08  07  06  05  04  03  02
 DUMP ADDR=0010   01  00  00  00  00  00  00  00  00  A2  8F  9A  A2  01  86  00
 00130 TRACE 0 $FFFF
 00140 DO MAIN NEXT 3 .TIMES
```

Output generated as a result of the DUMP command.

```
   IA  LABEL   OPCODE  A  S  X  Y  P  STATUS    PC   EA  EO   ICNT   TCNT   6501 TIME
 T0019 MAIN    LDX A2 00 00 8F 00 90 N  B      001B 001A 8F      1      2          0.
 T001B         TXS 9A 00 8F 8F 00 90 N  B      001C 001B 00      2      4          0.
 T001C         LDX A2 00 8F 01 00 10    B      001E 001D 01      3      6          0.
 T001E       . STX 86 00 8F 01 00 10    B      0020 0000 01      4      9          0.
 T0020         JSR 20 00 8D 01 00 10    B      0064 0064 00      5     15          0.
 T0064 BCD     LDY A0 00 8D 01 08 10    B      0066 0065 08      6     17          0.
 T0066         LDX A6 00 8D 01 08 10    B      0068 0000 01      7     20          0.
 T0068         CLC 18 00 8D 01 08 10    B      0069 0068 00      8     22          0.
 T0069         SED F8 00 8D 01 08 18    BD     006A 0069 00      9     24          0.
 T006A NEXT    LDA B5 08 8D 01 08 18    BD     006C 0008 08     10     28          0.
 T006C         ADC 75 09 8D 01 08 18    BD     006E 0010 01     11     32          0.
 T006E         STA 95 09 8D 01 08 18    BD     0070 0018 09     12     36          0.
 T0070         DEX CA 09 8D 00 08 1A    BD Z   0071 0070 00     13     38          0.
 T0071         DEY 88 09 8D 00 07 18    BD     0072 0071 00     14     40          0.
 T0072         BNE D0 09 8D 00 07 18    BD     006A 006A 00     15     43          0.
 T006A NEXT    LDA B5 07 8D 00 07 18    BD     006C 0007 07     16     47          0.
 T006C         ADC 75 09 8D 00 07 18    BD     006E 000F 02     17     51          0.
 T006E         STA 95 09 8D 00 07 18    BD     0070 0017 09     18     55          0.
 T0070         DEX CA 09 8D FF 07 98 N  BD     0071 0070 00     19     57          0.
 T0071         DEY 88 09 8D FF 06 18    BD     0072 0071 00     20     59          0.
 T0072         BNE D0 09 8D FF 06 18    BD     006A 006A 00     21     62          0.
```

Trace output generated during execution of the DO sequence.

```
 EMUL MONITOR DETECTED A WARNING-PAGE ZERO WRAP
```

A warning to the user that his program execution caused an index register to wrap around from hexadecimal FF to OO. This may not have been planned.

```
 T006A NEXT    LDA B5 06 8D FF 06 18    BD     006C 0006 06     22     66          0.
 +HILEV+ BREAKPOINT-NORMAL DO SEQUENCE END
```

Indicates normal DO sequence termination.

```
 00150 DUMP 1 $18
                 CONTENTS OF MEMORY LOCATION AT BASE ADDRESS PLUS........
   BASE ADDRESS   +0  +1  +2  +3  +4  +5  +6  +7  +8  +9  +A  +B  +C  +D  +E  +F
 DUMP ADDR=0000   01  01  02  03  04  05  06  07  08  08  07  06  05  04  03  02
 DUMP ADDR=0010   01  00  00  00  00  00  00  09  09  A2  8F  9A  A2  01  86  00
 00160 EXIT
 STOP.
RUN COMPLETE.
```

Note: For more detailed information refer to the MCS6500 Simulator manual.

7. PUNCH OBJECT TAPE

```
-DMP/***
```

—DMP/*** invokes the ROM dump program.

```
MOS TECHNOLOGY -- ROM DUMP

ENTER INTERFACE FILENAME ? INT4
ENTER OBJECT FILE NAME FOR OUTPUT --  ? OBJ4
OBJ4    CONTAINS OBJECT OUTPUT
```

INTERFACE file is the file created by the cross assembler.

OBJECT file is the file name the object code is to be saved in.

```
 STOP.

     0.135 /     0.809 /        18
OLD,OBJ4
READY - EXE!
PUNCH
```

Terminal commands required to list and punch the object tape.

Note: The paper tape punch should be turned on after the carriage return is entered.

```
 ;0E0019A28F9AA2018600206400EA4C230004F8
 ;100064A008A60018F8B507750F9517CA88D0F607D6
 ;050074D860EAEAEA046F
 ;0000030003


BYE
```

Sign-off the system by entering "BYE"

```
CT=00:20
M490010   LOG OFF.   09.30.38.
```

MCS6500 MICROPROCESSOR LANGUAGE

INSTRUCTION SET

ADC	Add Memory to Accumulator with Carry
AND	"AND" Memory with Accumulator
ASL	Shift Left One Bit (Memory or Accumulator)
BCC	Branch on Carry Clear
BCS	Branch on Carry Set
BEQ	Branch on Result Zero
BIT	Test Bits in Memory with Accumulator
BMI	Branch on Result Minus
BNE	Branch on Result not Zero
BPL	Branch on Result Plus
BRK	Force Break
BVC	Branch on Overflow Clear
BVS	Branch on Overflow Set
CLC	Clear Carry Flag
CLD	Clear Decimal Mode
CLI	Clear Interrupt Disable Bit
CLV	Clear Overflow Flag
CMP	Compare Memory and Accumulator
CPX	Compare Memory and Index X
CPY	Compare Memory and Index Y
DEC	Decrement Memory by One
DEX	Decrement Index X by One
DEY	Decrement Index Y by One
EØR	"Exclusive-or" Memory with Accumulator
INC	Increment Memory by One
INX	Increment X by One
INY	Increment Y by One
JMP	Jump to New Location
JSR	Jump to New Location Saving Return Address
LDA	Load Accumulator with Memory
LDX	Load Index X with Memory
LDY	Load Index Y with Memory
LSR	Shift One Bit Right (Memory or Accumulator)
NØP	No Operation
ØRA	"OR" Memory with Accumulator
PHA	Push Accumulator on Stack
PHP	Push Processor Status on Stack
PLA	Pull Accumulator from Stack
PLP	Pull Processor Status from Stack
RØL	Rotate One Bit Left (Memory or Accumulator)
RTI	Return From Interrupt
RTS	Return From Subroutine
SBC	Subtract Memory from Accumulator with Borrow
SEC	Set Carry Flag
SED	Set Decimal Mode
SEI	Set Interrupt Disable Status
STA	Store Accumulator in Memory
STX	Store Index X in Memory
STY	Store Index Y in Memory
TAX	Transfer Accumulator to Index X
TAY	Transfer Accumulator to Index Y
TSX	Transfer Stack Pointer to Index X
TXA	Transfer Index X to Accumulator
TXS	Transfer Index X to Stack Pointer
TYA	Transfer Index Y to Accumulator

EXECUTION TIMES (IN CLOCK CYCLES)

	Accumulator	Immediate	Zero Page	Zero Page, X	Zero Page, Y	Absolute	Absolute, X	Absolute, Y	Implied	Relative	(Indirect, X)	(Indirect), Y	Absolute Indirect
ADC	.	2	3	4	.	4	4*	4*	.	.	6	5*	.
AND	.	2	3	4	.	4	4*	4*	.	.	6	5*	.
ASL	2	.	5	6	.	6	7	.	.	.	.	.	.
BCC	.	.	.	.	.	.	.	.	.	2**	.	.	.
BCS	.	.	.	.	.	.	.	.	.	2**	.	.	.
BEQ	.	.	.	.	.	.	.	.	.	2**	.	.	.
BIT	.	.	3	.	.	4	.	.	.	.	.	.	.
BMI	.	.	.	.	.	.	.	.	.	2**	.	.	.
BNE	.	.	.	.	.	.	.	.	.	2**	.	.	.
BPL	.	.	.	.	.	.	.	.	.	2**	.	.	.
BRK	.	.	.	.	.	.	.	.	.	.	.	.	.
BVC	.	.	.	.	.	.	.	.	.	2**	.	.	.
BVS	.	.	.	.	.	.	.	.	.	2**	.	.	.
CLC	.	.	.	.	.	.	.	.	2	.	.	.	.
CLD	.	.	.	.	.	.	.	.	2	.	.	.	.
CLI	.	.	.	.	.	.	.	.	2	.	.	.	.
CLV	.	.	.	.	.	.	.	.	2	.	.	.	.
CMP	.	2	3	4	.	4	4*	4*	.	.	6	5*	.
CPX	.	2	3	.	.	4	.	.	.	.	.	.	.
CPY	.	2	3	.	.	4	.	.	.	.	.	.	.
DEC	.	.	5	6	.	6	7	.	.	.	.	.	.
DEX	.	.	.	.	.	.	.	.	2	.	.	.	.
DEY	.	.	.	.	.	.	.	.	2	.	.	.	.
EØR	.	2	3	4	.	4	4*	4*	.	.	6	5	.
INC	.	.	5	6	.	6	7	.	.	.	.	.	.
INX	.	.	.	.	.	.	.	.	2	.	.	.	.
INY	.	.	.	.	.	.	.	.	2	.	.	.	.
JMP	.	.	.	.	.	3	.	.	.	.	.	.	5
JSR	.	.	.	.	.	6	.	.	.	.	.	.	.
LDA	.	2	3	4	.	4	4*	4*	.	.	6	5*	.
LDX	.	2	3	.	4	4	.	4*	.	.	.	.	.
LDY	.	2	3	4	.	4	4*	.	.	.	.	.	.
LSR	2	.	5	6	.	6	7	.	.	.	.	.	.
NØP	.	.	.	.	.	.	.	.	2	.	.	.	.
ØRA	.	2	3	4	.	4	4*	4*	.	.	6	5*	.
PHA	.	.	.	.	.	.	.	.	3	.	.	.	.
PHP	.	.	.	.	.	.	.	.	3	.	.	.	.
PLA	.	.	.	.	.	.	.	.	4	.	.	.	.
PLP	.	.	.	.	.	.	.	.	4	.	.	.	.
RØL	2	.	5	6	.	6	7	.	.	.	.	.	.
RTI	.	.	.	.	.	.	.	.	6	.	.	.	.
RTS	.	.	.	.	.	.	.	.	6	.	.	.	.
SBC	.	2	3	4	.	4	4*	4*	.	.	6	5*	.
SEC	.	.	.	.	.	.	.	.	2	.	.	.	.
SED	.	.	.	.	.	.	.	.	2	.	.	.	.
SEI	.	.	.	.	.	.	.	.	2	.	.	.	.
STA	.	.	3	4	.	4	5	5	.	.	6	6	.
STX*	.	.	3	.	4	4	.	.	.	.	.	.	.
STY**	.	.	3	4	.	4	.	.	.	.	.	.	.
TAX	.	.	.	.	.	.	.	.	2	.	.	.	.
TAY	.	.	.	.	.	.	.	.	2	.	.	.	.
TSX	.	.	.	.	.	.	.	.	2	.	.	.	.
TXA	.	.	.	.	.	.	.	.	2	.	.	.	.
TXS	.	.	.	.	.	.	.	.	2	.	.	.	.
TYA	.	.	.	.	.	.	.	.	2	.	.	.	.

* Add one cycle if indexing across page boundary

** Add one cycle if branch is taken, Add one additional if branching operation crosses page boundary

ADDRESSING MODES

ACCUMULATOR ADDRESSING

This form of addressing is represented with a one byte instruction, implying an operation on the accumulator.

IMMEDIATE ADDRESSING

In immediate addressing, the operand is contained in the second byte of the instruction, with no further memory addressing required.

ABSOLUTE ADDRESSING

In absolute addressing, the second byte of the instruction specifies the eight low order bits of the effective address while the third byte specifies the eight high order bits. Thus, the absolute addressing mode allows access to the entire 65K bytes of addressable memory.

ZERO PAGE ADDRESSING

The zero page instructions allow for shorter code and execution times by only fetching the second byte of the instruction and assuming a zero high address byte. Careful use of the zero page can result in significant increase in code efficiency.

INDEXED ZERO PAGE ADDRESSING – (X, Y indexing)

This form of addressing is used in conjunction with the index register and is referred to as "Zero Page, X" or "Zero Page, Y". The effective address is calculated by adding the second byte to the contents of the index register. Since this is a form of "Zero Page" addressing, the content of the second byte references a location in page zero. Additionally due to the "Zero Page" addressing nature of this mode, no carry is added to the high order 8 bits of memory and crossing of page boundaries does not occur.

INDEXED ABSOLUTE ADDRESSING – (X, Y, indexing)

This form of addressing is used in conjunction with X and Y index register and is referred to as "Absolute, X", and "Absolute, Y". The effective address is formed by adding the contents of X or Y to the address contained in the second and third bytes of the instruction. This mode allows the index register to contain the index or count value and the instruction to contain the base address. This type of indexing allows any location referencing and the index to modify multiple fields resulting in reduced coding and execution time.

IMPLIED ADDRESSING

In the implied addressing mode the address containing the operand is implicitly stated in the operation code of the instruction.

RELATIVE ADDRESSING

Relative addressing is used only with branch instructions and establishes a destination for the conditional branch.

The second byte of the instruction becomes the operand which is an "Offset" added to the contents of the lower eight bits of the program counter when the counter is set at the next instruction. The range of the offset is -128 to +127 bytes from the next instruction.

INDEXED INDIRECT ADDRESSING

In indexed indirect addressing (referred to as (Indirect, X)), the second byte of the instruction is added to the contents of the X index register, discarding the carry. The result of this addition points to a memory location on page zero whose contents is the low order eight bits of the effective address. The next memory location in page zero contains the high order eight bits of the effective address. Both memory locations specifying the high and low order bytes of the effective address must be in page zero.

INDIRECT INDEXED ADDRESSING

In indirect indexed addressing (referred to as (Indirect), Y), the second byte of the instruction points to a memory location in page zero. The contents of this memory location is added to the contents of the Y index register, the result being the low order eight bits of the effective address. The carry from this addition is added to the contents of the next page zero memory location, the result being the high order eight bits of the effective address.

ABSOLUTE INDIRECT

The second byte of the instruction contains the low order eight bits of a memory location. The high order eight bits of that memory location is contained in the third byte of the instruction. The contents of the fully specified memory location is the low order byte of the effective address. The next memory location contains the high order byte of the effective address which is loaded into the sixteen bits of the program counter.

ASSEMBLER DIRECTIVES

.OPT – If used must be the first executable statement in the program.

.OPTIONS ARE: – (Options listed are the default value.)

COUNT (COU or CNT)	- List all instructions and their usage.
NOGENERATE (NOG)	- Do not generate more than one line of code for ASCII strings.
XREF (XRE)	- Produce a cross-reference list in the symbol table.
ERRORS (ERR)	- Create an error file.
MEMORY (MEM)	- Create an assembler object output file.
LIST (LIS)	- Produce a full assembly listing.

.BYTE – Produces a single BYTE in memory equal to each operand specified.

.WORD – Produces two BYTES in memory equal to each operand specified.

*= - Defines the beginning of a new program counter sequence.

.PAGE – Advances the listing to the top of a new page.

.END – Defines the end of a source program.

LABELS:

Labels begin in column 1 and are separated from the instruction by at least one space.

Labels can be up to 6 alphanumenic characters long and must begin with an alpha character.

A, X, Y, S, and P are reserved and cannot be used as labels.

LABEL = Expression can be used to equate labels to instructions.

LABEL * = * + N can be used to reserve areas in memory.

CHARACTERS USED AS SPECIAL PREFIXES:

.	Indicates an assembler directive.
#	Specifies the immediate mode of addressing.
$	Specifies a hexadecimal character.
@	Specifies an octal number.
%	Specifies a binary number.
'	Specifies an ASCII literal character.
()	Indicates indirect addressing.
;	In column 1 indicates a comment.

UCS SALES OFFICES

ATLANTA
Bldg. 1, Suite 106
5825 Glenridge Drive N.E.
Atlanta, Georgia 30328
Phone: (404) 256-3610

BOSTON
Fourth Floor
1050 Massachusetts Avenue
Cambridge, Massachusetts 02138
Phone: (617) 661-1720

CALGARY
Suite 1910
Bow Valley Square 2
P. O. Box 9235
Calgary, Alberta, Canada T2P2W5
Phone: (403) 265-4926

CHICAGO
Suite 1016
150 North Wacker Drive
Chicago, Illinois 60606
Phone: (312) 782-0865

CLEVELAND*
Two Commerce Park Square
23200 Chagrin Blvd.
Beachwood, Ohio 44122
Phone: (216) 464-9205

COLUMBUS*
P. O. Box 781
Delaware, Ohio 43015
Phone: (614) 548-6371

DALLAS
Suite 1112, Twin Towers South
8585 Stemmons Freeway
Dallas, Texas 75247
Phone: (214) 638-8260

DENVER
Suite 20C
2460 West 26th Avenue
Denver, Colorado 80211
Phone: (303) 458-8001

EAST ORANGE
33 Evergreen Place
East Orange, New Jersey 07018
Phone: (201) 677-2400

FT. WAYNE
Suite 101
3702 Rupp Drive
Fort Wayne, Indiana 46805
Phone: (219) 484-8522

FT. WORTH
Phone: (214) 263-0584
(Dallas office)

HOUSTON
4544 Post Oak Place
Suite 346
Houston, Texas 77027
Phone: (713) 622-5351

KANSAS CITY
500 W. 26th Street
Kansas City, Missouri 64108
Phone: (816) 221-9700

LOS ANGELES*
Suite 410
101 Continental Boulevard
El Segundo, California 90245
Phone: (213) 640-0891

MILWAUKEE
Suite 107
10701 West North Avenue
Wauwatosa, Wisconsin 53226
Phone: (414) 475-9392

NEW HAVEN
35 Worth Avenue
Hamden, Connecticut 06518
Phone: (203) 288-6287

NEW YORK
Suite 1847
Two Pennsylvania Plaza
New York, New York 10001
Phone: (212) 868-7785

OKLAHOMA CITY
Suite 252
Northwest Office Center
4334 N.W. Expressway
Oklahoma City, Oklahoma 73116
Phone: (405) 843-9784

ORLANDO
Suite 149
7200 Lake Elenor Drive
Orlando, Florida 32809
Phone: (305) 855-1810

PALO ALTO*
Suite 217
1032 Elwell Court
Palo Alto, California 94303
Phone: (415) 964-6990

PHILADELPHIA
Suite 210
500 Office Center
Ft. Washington, Pennsylvania 19034
Phone: (215) 542-8600

PHOENIX
Suite 104
5350 N. 16th St.
Phoenix, Arizona 85016
Phone: (602) 248-9176

SANTA ANA*
Suite 212
1651 East Fourth
Santa Ana, California 92701
Phone: (714) 835-3801

SAN FRANCISCO*
Suite 222
681 Market Street
San Francisco, California 94105
Phone: (415) 777-1885

SEATTLE
Suite B
Koll Commerce Center
699 Strander Blvd.
Tukwila, Washington 98188
Phone: (206) 243-8041

ST. LOUIS
Suite 100
7750 Clayton Road
Clayton, Missouri 63117
Phone: (314) 781-0123

TAMPA
Suite 518
1000 Ashley Drive
Tampa, Florida 33602
Phone: (813) 223-3921

TULSA
Suite 403
16 East 16 Street
Tulsa, Oklahoma 74119
Phone: (918) 582-7291

WASHINGTON, D.C.
Suite 319
7115 Leesburg Pike
Falls Church, Virginia 22043
Phone: (703) 532-1551

NATIONAL DATA CENTER
2525 Washington
Kansas City, Missouri 64108
Phone: (816) 221-9700

CORPORATE OFFICES
2525 Washington
Kansas City, Missouri 64108
Phone: (816) 221-9700

HEADQUARTERS –
MOS TECHNOLOGY, INC. 950 Rittenhouse Road
Norristown, Pa. 19401, (215) 666-7950, TWX: 510/660/4033

EASTERN REGION –
Mr. William Whitehead
MOS TECHNOLOGY, INC., Suite 312,
410 Jericho Turnpike, Jericho, N.Y. 11753
(516) 822-4240

WESTERN REGION –
MOS TECHNOLOGY, INC. 2172 Dupont Drive,
Patio Bldg., Suite 221. Newport Beach, CA. 92660
(714) 833-1600

Mr. Petr Sehnal, Regional Applications Mgr.
MOS TECHNOLOGY, INC., 22300 Foothill Blvd., Suite 311
Hayward, CA 94541
(415) 881-8080

MOS TECHNOLOGY, INC.