ZX65: Simulating a Micro

BY RICHARD M. KRUSE
6221 Woodlow Drive
Wichita, KS 67220

ABSTRACT

This article gives complete details, including source code, of an assembly language program which functionally simulates the operation of a 6502 microprocessor device using a Z80 based host computer system. Under control of ZX65, a wide variety of 6502 software may be directly executed in any of several modes. Useful for 6502 software development as well as for its tutorial value, ZX65 should prove to be a significant addition to the Z80 user's software library. Occupying only 4K Bytes of system memory, ZX65 is sufficiently flexible to be readily adapted to system environments other than the one described.

Having logged many hours of software development work on my Z80-based system, I recently decided that I was up to a particular challenge: using my system as a development tool for a totally different microprocessor; specifically, the 6502. This decision, I must confess, was not altogether based on a burning lust for knowledge, but rather on a more practical requirement: I had become involved in the development of a 6502-based device, and had no ready access to a 6502 development system. I decided that a simple cross-assembler wasn't good enough, and set out to write a 6502-to-Z80 object code translator/simulator/interpreter.

The program described here, which I have dubbed ZX65, is the result of my efforts to generate a software package to perform this task. It is useful not only for the intended purpose of software development, but also for the rewarding experience of exploring a whole new field of good software written for a processor other than one's own.

ZX65 operates as an interpretive simulator; that is, a simulated 6502 CPU is maintained in system memory, and each encountered 6502 instruction is decoded and processed interpreter-fashion to properly act upon the simulated CPU registers. The entire package resides in just over 4K Bytes of high system memory, leaving the remaining portion, including the all-important (for 6502) base page, free for program and data storage. Note that CP/M serves only as a loader and is not used thereafter. In fact, while the program loads normally into CP/M's Transient Program Area, it immediately relocates itself into high memory, overlaying both the CCP and the BDOS portions of CP/M. The user I/O linkages (CBIOS), however, remain intact and are used by the package to perform disk and console functions. The functions which must be provided by the system are a subset of those required by CP/M and should thus be readily available. These functions are all accessed via a jump table within the interpreter and thus can be readily changed to accommodate different system requirements (See Table 1). Parameters must be passed to the drivers in register "C" (register pair "BC" for item six) and data, if any, is returned in the accumulator.

ZX65 is referred to as a package since it includes three distinct function groups. The first and largest is the interpreter/simulator itself. The second group is an elementary monitor providing the functions of memory examine and modify. The third group consists of a self-contained, elementary disk operating system (DOS) (Note: ZX65 files are not CP/M compatible). The normal configuration assumes a two-drive system, with the system diskette on drive A, and all internal disk access routed to drive B (Modification for a single drive system could easily be done, requiring the user to switch diskettes after the package is loaded). The package is written in standard Zilog/Mostek source code and is quite heavily commented. Much use is made of the Z80's special instructions and index registers, and operation on an 8080 is definitely not possible without extensive modification. Although the commented listing should guide you through the program's operation without too much trouble, a brief description of the interpreter portion is in order. A study of the small but powerful instruction set of the 6502 will reveal certain patterns in the op codes, as also occurs with most other processors. The eight "BRANCH ON CONDITION" codes, for example, all have zero as a low-order nybble, while the high-order portion is always an odd value.

Another pattern, not so obvious, is that every instruction having an odd-valued op code uses multiple addressing modes. These and other such patterns are exploited by the program. Even so, much of the code is devoted solely to decoding the op code and transferring control to the proper instruction processor. The code beginning at "STEP" (line 241) performs the initial decoding, and fully decodes certain instructions such as "RTI", "JMP", and a few others. In most cases, decoding proceeds to a point where the instruction is known to be part of a group, such as the branch group described above. In these cases, look-up tables are used, but not always in the same way. The branch group and the flag set/reset group, for example, derive mask patterns from the tables which are used for testing, setting, or resetting the condition flags.

In the case of multiple addressing mode instructions, however, the tables serve an entirely different purpose. The table contents are, in these instances, decoded into addresses for indirect jumps; first to the addressing mode processors, and then to the actual instruction processors. Each processor performs its instruction as required. The "ADC" instruction, as an example, causes a byte of data to be fetched from memory or from the object code, depending on the addressing mode. After setting the Z80 carry flag to match the simulated 6502 carry flag, this data byte is added to the value currently in the simulated accumulator, and the result is stored there. The simulated carry flag is then updated to again match the Z80 carry, and the simulated sign, overflow, and zero flags are set or reset according to the results in the accumulator (In this case, the overflow flag is set or reset according to a logical exclusive "OR" of bits six and seven). Considerable care has been given to treatment of the flags, since Z80 flags and their 6502 namesakes do not always have the same meaning (Case in point: the 6502 carry flag assumes a logical

"NOT" BORROW meaning during subtract and compare operations, while the Z80 and most other common processors invert the carry flag internally during these operations so that it represents a "TRUE" BORROW; other flag differences, for the most part, are more subtle).

Finally, each individual processor is responsible for exiting with codes for the mnemonic, the addressing mode, and the number of bytes occupied by the instruction available for use by the "DISPLAY" routine (line 173) to index into yet another set of tables and retrieve the ASCII messages for mnemonic and addressing modes. Upon exiting the instruction processors, the value stored in "NEXT PROGRAM COUNTER" is updated, and control is passed back to the COMMAND section where the operating mode in effect determines whether to display the updated CPU contents, and whether or not to continue execution with the next instruction.

Several distinct operating modes are available, defined briefly as follows:

Single-Step Mode: One instruction is decoded and executed. The console then displays the address of the instruction just executed (Current Program Counter), the decoded instruction (Mnemonic and Operand), the state of all CPU registers, and the address of the next instruction to be executed (Next Program Counter). Control is then returned to the user.

Multi-Step Trace Mode: The user specifies up to 255 instructions to be executed with the results displayed as above following each step. This occurs at the rate of approximately two instructions per second. Control is returned to the user after the required number of steps have been executed and displayed.

Run-to-Breakpoint Mode: Instructions are executed without display up to and including the user-specified breakpoint. CPU registers are then displayed as above and control is returned to the user.

Free-Run Mode: Instructions are executed with no display and no breakpoints. The console keyboard, however, is monitored, and CPU contents are displayed as above and control is returned to the user at any point upon console input.

The user may examine and modify any or all CPU registers, including the program counter, at any time that he has control, without otherwise disturbing any operation in progress.

The 6502 "BRK" instruction (software interrupt) is handled through a vector location, just as in a normal system. Optionally, however, the BRK instruction may be used (and is initialized) to return control to the user in the same manner as the breakpoint mode described above. The BRK instruction and breakpoint mode operate independently, and both may be used in the same program. Additionally, up to five different "user" subroutines may be called for transfer of control to Z80 routines (such as console I/O). These are invoked by executing either a "JMP" or "JSR" instruction to addresses 0000, 0001, 0002, 0003, and 0004 (HEX). Control is passed via a jump table located within the intepreter, which the user may load with Z80 "JP" instructions to the desired subroutines. The subroutines must be terminated in the normal Z80 fashion by one of the "RET" instructions. Note that the first two entries in the table (JMP or JSR 0000, 0001 HEX) are initialized as console input and output via the 6502 accumulator. Three parameters may be passed to and from all user subroutines, as outlined in Table 2. Table 3 lists the locations of the BRK vector and the user subroutine jump table.

Although the interpreter is quite versatile and all 6502 instructions are handled properly, there are some limitations to its use. Perhaps the most important of these is that, even in the "free-run" mode, execution is quite slow compared to an actual 6502 running at normal clock speeds. No attempt has been made to correlate execution times with normal 6502 clock periods (this may, in fact, be impossible due to the way in which the system handles multiple addressing modes). Therefore, programs incorporating software timing loops will not execute properly. Also, since the package was intended for software development, hardware interrupts are not supported, and only a limited amount of hardware interfacing can be performed through the "user" subroutine calls.

Two program listings are included here. One, "ZXLD" is located at the end of this article. Its function is to boot the main program from the CP/M Transient Program Area to high memory. The other listing is the interpreter itself, assembled to reside from 6F00 to 7DFF (HEX) in a 32K system.

Note that this leaves 512 bytes, normally containing CBIOS, unaffected. The program is not relocatable, so that in order to run in a different memory configuration, it will be necessary to reassemble the source with a new value in the "SIZE EQU XX" statement. All other addresses are derived from this value. If the object code is to be manually loaded for the 32K system then "ZXLD" must be loaded first, starting at 0100 (HEX). Then start loading ZX65 at 0110 (HEX), continuing until the end. Finally, use the CP/M "SAVE" command (SAVE 15 ZX65.COM) to write the object code to disk. The package can now be invoked by answering "ZX65" to the CP/M prompt. If you are reassembling the package, then you should also reassemble the ZXLD routine (modified for your system) so that you have a .HEX file for each module. Then load the package using DDT as follows:

```
DDT ZXLD.HEX
IZX65.HEX
Rdisp
```

Where "disp" is the load offset value required to load the object code from the .HEX file at a different address than that specified. (DDT will calculate "disp" if you type "H1 10, ssss", where "ssss" is the actual starting address of your program. DDT will respond with the sum and difference of the two values, and the difference will be the required value "disp".) Finally, save the object code as directed above for manual loading.

To run the program, place the disk with ZX65.COM on drive A (this disk should also have a copy of CP/M so that the system can be restarted, if necessary, without changing disks). Place the ZX65 data disk on drive B. Start CP/M and invoke ZX65 by answering "ZX65" to the CP/M input prompt. The system will prompt you with a header and a ">" symbol. You are now in the command mode, and typing one of the command codes (see Table 4) will cause the appropriate action to occur. Note that if you are using a disk not previously used for ZX65 on the drive B, you will be unable to use the ZX65 DOS functions until you have initialized the disk. This is done simply by typing "I", at which point the system will ask you to verify this request with a "Y" or "N". The reason for this

(note this well) is that the "I" command clears and initializes the directory on disk B. This is the equivalent of the CP/M command "ERA *.*" and must be used with caution.

For those unwilling to type in a 4K Byte program by hand, I will make the package available on a standard 8 inch IBM format, single side, single density diskette (Sorry, I can't handle any other format!) at a nominal charge of $15 to cover time and material. Write to me and specify your system memory size. I will send a diskette containing both source and object files assembled to reside in high memory as described above, as well as a copy of ZXLD. A command summary, operating hints, and other general information will be included on the disk.

Finally, although this program has been rather exhaustively tested with a variety of 6502 software (both simple and sophisticated) I cannot guarantee it bug-free. I have included no copyright notices of any kind, and I hope users will feel free to copy, add, delete, and improve at will. (I would like to hear of any major modifications . . . I may want to include them myself!) As stated at the outset, the package was developed as a side benefit of another project, not as a money maker, and if anyone else finds it useful, so much the better.

Acknowledgments:

CP/M is a registered trademark of Digital Research, Pacific Grove, CA.

While not specifically mentioned in the text, I have relied heavily on the information contained in the book "6502 Assembly Language Programming" by Lance A. Leventhal (Osborne/McGraw-Hill Inc., 1979).

Function	Address
1. Test console input status	7A2EH
2. Get single character from console	7A2BH
3. Send single character to console	7A28H
4. Select disk drive (A or B)	7A34H
5. Home R/W head of selected drive	7A31H
6. Set disk transfer buffer address	7A3DH
7. Set sector number for disk access	7A37H
8. Set track number for disk access	7A3AH
9. Read one sector from selected drive	7A40H
10. Write one sector to selected drive	7A43H

Table 1: Functions which ZX65 requires of the host system, and their locations within the interpreter.

6502 Register	Passed as	Z80 Register
Accumulator		Accumulator
X Index		B Register
Y Index		C Register

Table 2: 6502/Z80 Parameter passing for user subroutines.

Description	Address of vector or jump instruction
BRK Vector	7A55H
User 0	7A46H
User 1	7A49H
User 2	7A4CH
User 3	7A4FH
User 4	7A52H

Table 3: Locations for 'BRK' vector and user jump table.

Command	Function
C	CPU register display and modify.
D	Directory display of ZX65 files on disk B. Format: FILNAM.TYP LOAD ADDRESS #RECORDS
E	Examine a block of memory. System will prompt for starting address and number of bytes.
G	Go to 6502 program per Current Program Counter and execute without display (free-run).
I	Initialize a fresh disk on drive B. (Must be formatted.) System will prompt for verification.
K	Kill a ZX65 file on drive B. Similar to CP/M ERA. System will prompt for file name.
L	Load a ZX65 file from disk B. System will prompt for file name.
M	Memory substitution. View sequential memory locations and update if desired. System will prompt for address. Mode will continue in effect until a '.' (period) is typed.
R	Load and run a ZX65 file from disk B, starting at the address specified in the directory. (Must be an executable 6502 program.) System will prompt for file name.
S	Save a ZX65 file on disk B. System will prompt for file name, starting address, and number of bytes.
T	Trace several 6502 instructions with display following each step. System will prompt for desired number of steps.
SPACE	Single-step command. System will execute one 6502 instruction, display the results, and stop.

Table 4: A command summary for ZX65 (in alphabetical order).

LISTING

```
0001 ; ZX65: SIMULATING A MICRO   1/80   R M KRUSE
0002 ; FOR Z80-BASED SYSTEMS USING THE CP/M DOS
0003 ;
0004 ; DESIGNED TO REPLACE ALL OF CP/M EXCEPT FOR USERS CBIOS
0005 ; THRU WHICH DISK AND I/O CALLS ARE MADE.
0006 ;
0007 ; THIS VERSION IS ASSEMBLED FOR A 32K SYSTEM. TO CHANGE
0008 ; TO ANOTHER SIZE, INSERT APPROPRIATE VALUE FOR 'SIZE'
0009 ; AND RE-ASSEMBLE. SOURCE CODE IS IN PROPER FORMAT FOR
0010 ; THE S. D. SALES Z80 ASSEMBLER V1.0.
0011 ;
0012 ; PROGRAM MAY BE LOADED BY THE FOLLOWING PROCEDURE:
0013 ;       DDT SIMLD.HEX
0014 ;       IZX65.HEX
0015 ;       R9210
0016 ;       GO
0017 ;       SAVE 15 ZX65.COM
0018 ;
0019 ; THE .RES. OPERATOR CAUSES THIS ASSEMBLER TO CLEAR ANY
0020 ; OVERFLOW IN THE OPERAND FIELD.
0021 ;
0022 SIZE    EQU   32              ; # K CP/M SYSTEM
0023 TOP     EQU   .RES.SIZE*1024  ; TOP OF RAM
0024 BASE    EQU   .RES.TOP-1100H  ; ACTUAL SYSTEM RUN ADDR
0025 HOST    EQU   .RES.TOP-200H   ; HOST SYSTEM CBIOS
0026 SYSTK   EQU   HOST            ; SYSTEM STACK
0027 CPWRM   EQU   .RES.HOST+3     ; CP/M WRM ST
0028 CONST   EQU   .RES.HOST+6     ; HOST SYSTEM INPUT TEST
0029 CONIN   EQU   .RES.HOST+9     ; HOST SYSTEM CONS IN
0030 CONOUT  EQU   .RES.HOST+0CH   ; HOST SYSTEM CONS OUT
0031 HOME    EQU   .RES.HOST+18H   ; DRIV HOME
0032 SETDRV  EQU   .RES.HOST+1BH   ; SELECT DRIVE
0033 SETTRK  EQU   .RES.HOST+1EH   ; SELECT TRACK
0034 SETSEC  EQU   .RES.HOST+21H   ; SELECT SECTOR
0035 SETBUF  EQU   .RES.HOST+24H   ; SET XFER ADDRESS
0036 READ    EQU   .RES.HOST+27H   ; READ A SECTOR
0037 WRITE   EQU   .RES.HOST+2AH   ; WRITE A SECTOR
0038 ;
0039         ORG   BASE
0040         LD    SP,SYSTK
0041         LD    C,1

>0020
>8000
>6F00
>7E00
>7E00
>7E03
>7E06
>7E09
>7E0C
>7E18
>7E1B
>7E1E
>7E21
>7E24
>7E27
>7E2A

>6F00 310007E
6F00
6F03 0E01
```

```
6F05  CD347A    0042          CALL    DRVS            ; ACTIVATE DRV B
6F08  CD317A    0043          CALL    HOMB            ; AND INIT IT
6F0B  FD21667A  0044          LD      IY,INTRO
6F0F  CDD979    0045          CALL    PRTXT           ; PRINT HEADER
6F12  CD4F79    0046          CALL    CRLF
                0047 ;
                0048 ; COMMAND LEVEL...RETURN HERE FROM ANY POINT IN PGM
                0049 ;
6F15  21156F    0050 CMND     LD      HL,CMND
6F18  E5        0051          PUSH    HL              ; RET ADDR ON STACK
6F19  CD4F79    0052          CALL    CRLF
6F1C  CD5979    0053          CALL    PROM
6F1F  CD167A    0054          CALL    GETCHR          ; GET COMMAND
6F22  FE45      0055          CP      'E'
6F24  CAE975    0056          JP      Z,EXAM          ; EXAM MEMORY
6F27  FE4D      0057          CP      'M'
6F29  CA4B76    0058          JP      Z,SUBM          ; SUB MEMORY
6F2C  DD215D7A  0059          LD      IX,VREG         ; FOR SIMX
6F30  FE42      0060          CP      'B'
6F32  2864      0061          JR      Z,BRKP-$        ; RUN TO BREAKPOINT
6F34  FE43      0062          CP      'C'
6F36  CACE6F    0063          JP      Z,RGSTR         ; DSPLY/MODIFY CPU REGS
6F39  FE47      0064          CP      'G'
6F3B  CAC16F    0065          JP      Z,RUN           ; RUN FROM NPC
6F3E  FE54      0066          CP      'T'
6F40  282D      0067          JR      Z,MULTX-$       ; TRACE/DISPLAY N STEPS
6F42  FE20      0068          CP      ' '
6F44  2836      0069          JR      Z,EXONE-$       ; EX ONE INSTR & STOP
6F46  DD216A7B  0070          LD      IX,BUFFR        ; FOR DOS
6F4A  FE44      0071          CP      'D'
6F4C  CAA876    0072          JP      Z,PDIR          ; DOS DIR
6F4F  FE49      0073          CP      'I'
6F51  CACB77    0074          JP      Z,INIT          ; DOS INITIALIZE
6F54  FE4B      0075          CP      'K'
6F56  CAB677    0076          JP      Z,KILL          ; DOS ERASE
6F59  FE4C      0077          CP      'L'
6F5B  CAF576    0078          JP      Z,LOAD          ; DOS LOADER
6F5E  FE52      0079          CP      'R'
6F60  CAEB76    0080          JP      Z,LDGO          ; LOAD AND RUN PGM
6F63  FE53      0081          CP      'S'
6F65  CA1377    0082          JP      Z,SAVE          ; DOS SAVER
6F68  0E3F      0083          LD      C,'?'
6F6A  CD287A    0084          CALL    PRNC            ; INVALID COMMAND
6F6D  18A6      0085          JR      CMND-$
                0086 ;
                0087 ; EXECUTE N STEPS
                0088 ;
6F6F  FD211F7B  0089 MULTX    LD      IY,MLTMSG
6F73  CDD979    0090          CALL    PRTXT
6F76  CD6379    0091          CALL    GETWRD          ; GET N
6F79  7B        0092          LD      A,E
6F7A  2002      0093          JR      NZ,DOMULT-$
6F7C  3E01      0094 EXONE    LD      A,1             ; EXECUTE 1 STEP
                0095 ;
                0096 ; EXECUTE # STEPS IN A
                0097 ;
6F7E  32577A    0098 DOMULT   LD      (NSTEPS),A      ; STORE N IN TEMP
6F81  CD9670    0099          CALL    STEP            ; EXEC & DSPLY INSTR
6F84  CD1170    0100          CALL    DSPLY
6F87  3A577A    0101          LD      A,(NSTEPS)
6F8A  3D        0102          DEC     A
6F8B  C8        0103          RET     Z
6F8C  F5        0104          PUSH    AF
6F8D  01597A    0105          LD      BC,DLYCON       ; DELAY FOR VIEWING
6F90  0B        0106 DOLP     DEC     BC
6F91  78        0107          LD      A,B
6F92  B1        0108          OR      C
6F93  20FB      0109          JR      NZ,DOLP-$
6F95  F1        0110          POP     AF
6F96  18E6      0111          JR      DOMULT-$
                0112 ;
                0113 ; RUN WITHOUT DISPLAY TO BREAKPOINT
                0114 ;
6F98  FD21317B  0115 BRKP     LD      IY,BKPMSG
6F9C  CDD979    0116          CALL    PRTXT
6F9F  CD6379    0117          CALL    GETWRD          ; GET IT
6FA2  ED535B7A  0118          LD      (BREAK),DE      ; STORE IT IN TEMP
6FA6  CD9670    0119 BKLP     CALL    STEP            ; EXEC ONE INSTR
6FA9  ED5B5B7A  0120          LD      DE,(BREAK)
6FAD  7C        0121          LD      A,H             ; CHECK IF AT BKPT
6FAE  BA        0122          CP      D
6FAF  2009      0123          JR      NZ,INTST-$
6FB1  7D        0124          LD      A,L
6FB2  BB        0125          CP      E
6FB3  2005      0126          JR      NZ,INTST-$
6FB5  CD9670    0127          CALL    STEP            ; DO ONE MORE
6FB8  1857      0128          JR      DSPLY-$
6FBA  CD2E7A    0129 INTST    CALL    KBST            ; TEST FOR KBD INP
6FBD  28E7      0130          JR      Z,BKLP-$        ; CONTINUE IF NONE
6FBF  1808      0131          JR      QDSP-$
                0132 ;
                0133 ; RUN ONLY...NO DISPLAY OR BREAKPOINT
                0134 ;
6FC1  CD9670    0135 RUN      CALL    STEP            ; DO A STEP
6FC4  CD2E7A    0136          CALL    KBST            ; CHECK FOR KB INPUT
6FC7  28F8      0137          JR      Z,RUN-$         ; IF NONE JUST REPEAT
                0138 ;
                0139 ; STOP, DISPLAY, & RET TO CMND
                0140 ;
6FC9  CD2B7A    0141 QDSP     CALL    KBIN            ; CLEAR UART
6FCC  1843      0142          JR      DSPLY-$         ; DISP CPU & RET
                0143 ;
                0144 ; DISPLAY & UPDATE CPU REGISTERS
                0145 ;
6FCE  DDE5      0146 RGSTR    PUSH    IX
6FD0  CD4F79    0147          CALL    CRLF
6FD3  FD21AA7A  0148          LD      IY,HEADR+42
6FD7  CD8C70    0149          CALL    PRN6            ; PRINT RG NAME
6FDA  ED5B647A  0150          LD      DE,(NPC)        ; NEXT PC TO DE
6FDE  CDB279    0151          CALL    PRTWRD          ; PRINT IT
6FE1  CD5979    0152          CALL    PROM            ; PROMPT FOR CHANGE
6FE4  CD6379    0153          CALL    GETWRD
6FE7  2804      0154          JR      Z,AREG-$        ; SPACE = NO CHANGE
6FE9  ED53647A  0155          LD      (NPC),DE        ; PUT IN NEW VALUE
6FED  FD218C7A  0156 AREG     LD      IY,HEADR+12
6FF1  0605      0157          LD      B,5
6FF3  CD4F79    0158 NREG     CALL    CRLF            ; DO REST OF REGS
6FF6  CD8C70    0159          CALL    PRN6            ; PRINT NAME
6FF9  DD5E00    0160          LD      E,(IX+0)
6FFC  CDB679    0161          CALL    PRTBYT          ; PRINT CONTENTS
6FFF  CD5979    0162          CALL    PROM            ; PROMPT FOR CHANGE
7002  CD6379    0163          CALL    GETWRD
7005  2803      0164          JR      Z,NXTREG-$      ; SPACE = NO CHANGE
7007  DD7300    0165          LD      (IX+0),E
700A  DD23      0166 NXTREG   INC     IX              ; POINT TO NEXT REG
700C  10E5      0167          DJNZ    NREG-$          ; DO AGAIN IF MORE
```

```
700E  DDE1        0168          POP     IX
7010  C9          0169          RET
                  0170 ;
                  0171 ; DISPLAY PC, INSTRUCTION, AND ALL REGISTERS
                  0172 ;
7011  DDE5        0173 DSPLY    PUSH    IX              ; SAVE POINTER
7013  C5          0174          PUSH    BC              ; SAVE MNEMONIC CODE
7014  CD4F79      0175          CALL    CRLF
7017  FD21807A    0176          LD      IY,HEADR
701B  CD8C70      0177          CALL    PRN6            ; 'NPC'
701E  ED5B627A    0178          LD      DE,(CPC)
7022  CDB279      0179          CALL    PRTWRD          ; VALUE
7025  CD8C70      0180          CALL    PRN6            ; 'INS'
7028  C1          0181          POP     BC
7029  FDE5        0182          PUSH    IY
702B  C5          0183          PUSH    BC
702C  FD21A87C    0184          LD      IY,MNMTBL       ; DET MNEM FROM BC
7030  3EFC        0185          LD      A,0FCH
7032  A1          0186          AND     C
7033  CB3F        0187          SRL     A
7035  5F          0188          LD      E,A
7036  1600        0189          LD      D,0
7038  FD19        0190          ADD     IY,DE
703A  FD5E00      0191          LD      E,(IY+0)        ; GET PACKED MNEM
703D  FD5601      0192          LD      D,(IY+1)
7040  CB23        0193          SLA     E               ; UNPACK IT
7042  CB12        0194          RL      D
7044  0E03        0195          LD      C,3             ; & PRINT THREE CHRS
7046  AF          0196 MNM2     XOR     A
7047  0605        0197          LD      B,5
7049  CB23        0198 MNM1     SLA     E
704B  CB12        0199          RL      D
704D  17          0200          RLA
704E  10F9        0201          DJNZ    MNM1-$
7050  CBF7        0202          SET     6,A
7052  C5          0203          PUSH    BC
7053  CD277A      0204          CALL    PRNA
7056  C1          0205          POP     BC
7057  0D          0206          DEC     C
7058  20EC        0207          JR      NZ,MNM2-$
705A  C1          0208          POP     BC              ; DECODE ADDR MODE
705B  78          0209          LD      A,B
705C  CB27        0210          SLA     A
705E  CB27        0211          SLA     A
7060  CB27        0212          SLA     A
7062  5F          0213          LD      E,A
7063  1600        0214          LD      D,0
7065  FD21187D    0215          LD      IY,AMDTBL
7069  FD19        0216          ADD     IY,DE
706B  CD9170      0217          CALL    PRN8            ; PRINT ADDR MODE
706E  FDE1        0218          POP     IY
7070  0605        0219          LD      B,5
7072  CD8C70      0220 DSP1     CALL    PRN6            ; PRINT ALL REGS
7075  DD5E00      0221          LD      E,(IX+0)
7078  CDB679      0222          CALL    PRTBYT
707B  DD23        0223          INC     IX
707D  10F3        0224          DJNZ    DSP1-$
707F  CD8C70      0225          CALL    PRN6            ; PRINT NEXT PC
7082  ED5B647A    0226          LD      DE,(NPC)
7086  CDB279      0227          CALL    PRTWRD
7089  DDE1        0228          POP     IX
708B  C9          0229          RET
                  0230 ;
708C  1E06        0231 PRN6     LD      E,6             ; PRINT 6 CHARS PER IY
708E  C3DB79      0232          JP      PRNLP
7091  1E08        0233 PRN8     LD      E,8             ; PRINT 8 CHRS PER IY
7093  C3DB79      0234          JP      PRNLP
                  0235 ;
                  0236 ; CODE FOR DECODING AND EXECUTING ONE INSTRUCTION
                  0237 ; ENTER WITH HL POINTING TO INSTR TO BE DECODED.  EXITS
                  0238 ; WITH HL POINTING TO NEXT INSTR, B HOLDING ADDRESSING
                  0239 ; MODE CODE, C HOLDING MNEMONIC CODE
                  0240 ;
7096  2A647A      0241 STEP     LD      HL,(NPC)        ; NEXT PC TO HL
7099  22627A      0242          LD      (CPC),HL        ; BECOMES CURRENT PC
709C  7E          0243          LD      A,(HL)          ; OP CODE
709D  B7          0244          OR      A               ; ZERO?
709E  CAF670      0245          JP      Z,PBRK          ; IF SO, 'BRK'
70A1  CB47        0246          BIT     0,A             ; TEST LSD FOR ODD/EVEN
70A3  C21873      0247          JP      NZ,MLTMOD       ; ODD INSTRS MULTI MODE
70A6  E60F        0248          AND     0FH
70A8  2018        0249          JR      NZ,FLGTST-$
70AA  7E          0250          LD      A,(HL)          ; RESTORE OP CODE
70AB  CB67        0251          BIT     4,A             ; MSD ODD/EVEN
70AD  C22A71      0252          JP      NZ,PBRNCH       ; IF ODD, BRANCH
70B0  FE20        0253          CP      20H
70B2  CA6271      0254          JP      Z,PJSR          ; PROCESS JSR
70B5  FE40        0255          CP      40H
70B7  CAD971      0256          JP      Z,PRTI          ; PROCESS RTI
70BA  FE60        0257          CP      60H
70BC  CAED71      0258          JP      Z,PRTS          ; PROCESS RTS
70BF  C31873      0259          JP      MLTMOD          ; ELSE MULTI MODE
70C2  FE08        0260 FLGTST   CP      8
70C4  200E        0261          JR      NZ,IMP2-$
70C6  7E          0262          LD      A,(HL)          ; RESTORE
70C7  CB67        0263          BIT     4,A             ; MSD ODD/EVEN
70C9  CA2172      0264          JP      Z,PIMPL         ; IF EVEN, MST BE IMPLD
70CC  FE98        0265          CP      98H             ; MSD = 9?
70CE  CADB72      0266          JP      Z,PTYA          ; YES, PROC TYA SPECIAL
70D1  C3E872      0267          JP      PFLGS           ; ELSE PROCESS FLAG OPS
70D4  FE0A        0268 IMP2     CP      0AH
70D6  2010        0269          JR      NZ,JMPTST-$     ; MUST BE JUMP
70D8  7E          0270          LD      A,(HL)          ; RESTORE
70D9  E6F0        0271          AND     0F0H            ; TESTS ON MSD
70DB  CB67        0272          BIT     4,A             ; ODD/EVEN
70DD  C20672      0273          JP      NZ,PTSS         ; PROSESS SP OPS SPECIAL
70E0  FE70        0274          CP      70H
70E2  DA1873      0275          JP      C,MLTMOD        ; PROCESS ACCUM MODES
70E5  C32172      0276          JP      PIMPL           ; ELSE MORE IMPLIEDS
70E8  7E          0277 JMPTST   LD      A,(HL)          ; RESTORE OP CODE
70E9  FE4C        0278          CP      4CH
70EB  CA8471      0279          JP      Z,PJABS         ; PROCESS ABS JMP
70EE  FE6C        0280          CP      6CH
70F0  CA8A71      0281          JP      Z,PJIND         ; PROCESS INDIRECT JMP
70F3  C31873      0282          JP      MLTMOD
                  0283 ;
                  0284 ; INSTURCTION PROCESSORS BEGIN HERE.  ENTER WITH
                  0285 ; HL = PC, IX POINTING TO SIM CPU
                  0286 ;
                  0287 ; IF BRK VECTOR (IPOINT) = 0, RETURN TO CMND
                  0288 ;
70F6  DDCB01E6    0289 PBRK     SET     4,(IX+1)        ; SET BRK STATUS BIT
70FA  2A557A      0290          LD      HL,(IPOINT)     ; SEE IF RET TO CMND
70FD  7C          0291          LD      A,H
```

```
70FE  B5      0292          OR      L
70FF  2007    0293          JR      NZ,BRK1-$
7101  C1      0294          POP     BC              ; ADJUST SP
7102  012904  0295          LD      BC,429H         ; RET TO CMND
7105  C31170  0296          JP      DSPLY
7108  23      0297 BRK1     INC     HL              ; NORMAL BREAK
7109  CD0475  0298          CALL    TSPY            ; SP TO IY
710C  FD7400  0299          LD      (IY+0),H        ; 'PUSH' PC
710F  FD2B    0300          DEC     IY
7111  FD7500  0301          LD      (IY+0),L
7114  FD2B    0302          DEC     IY
7116  DD7E01  0303          LD      A,(IX+1)        ; 'PUSH' P
7119  FD7700  0304          LD      (IY+0),A
711C  FD2B    0305          DEC     IY
711E  CD0D75  0306          CALL    TYSP            ; IY TO SP
7121  2A557A  0307          LD      HL,(IPOINT)     ; GET INTERRUPT VECTOR
7124  012904  0308          LD      BC,429H         ; CODE FOR # BYTES & MNEM
7127  C3D774  0309          JP      END1            ; AND FINISH
              0310 ;
              0311 ; PROCESS BRANCHES
              0312 ;
712A  0600    0313 PBRNCH   LD      B,0             ; FORM INDEX INTO MSKTBL
712C  4E      0314          LD      C,(HL)          ; INSTR TO C
712D  FD213C7C 0315         LD      IY,MSKTBL
7131  CB39    0316          SRL     C               ; GET MSD
7133  CB39    0317          SRL     C
7135  CB39    0318          SRL     C
7137  CB39    0319          SRL     C
7139  0D      0320          DEC     C
713A  FD09    0321          ADD     IY,BC
713C  23      0322          INC     HL              ; POINT TO OFFSET
713D  CB49    0323          BIT     1,C             ; MSD ODD/EVEN
713F  DD7E01  0324          LD      A,(IX+1)        ; P REG TO ACC
7142  FD4E00  0325          LD      C,(IY+0)        ; # CODE
7145  FD4601  0326          LD      B,(IY+1)        ; FLAG MASK
7148  2005    0327          JR      NZ,BRSET-$      ; ODD/EVEN DET SET/CLR
714A  A0      0328          AND     B               ; ISOLATE BIT FOR TEST
714B  2807    0329          JR      Z,TRUE-$
714D  180E    0330          JR      FALSE-$
714F  A0      0331 BRSET    AND     B
7150  2002    0332          JR      NZ,TRUE-$
7152  1809    0333          JR      FALSE-$
7154  1600    0334 TRUE     LD      D,0             ; CONDITION MET; BRANCH
7156  5E      0335          LD      E,(HL)          ; OFFSET TO E
7157  CB7B    0336          BIT     7,E             ; CHECK SIGN OF OFFSET
7159  2801    0337          JR      Z,TRU1-$
715B  15      0338          DEC     D               ; IF B7 IS SET, DO NEG
715C  19      0339 TRU1     ADD      HL,DE          ; ADD OFFSET FOR NEW PC
715D  0600    0340 FALSE    LD      B,0             ; COND NOT MET; GO ON
715F  C3D674  0341          JP      END
              0342 ;
              0343 ; PROCESS JUMPS AND JSR...JMP 0-4 AND JSR 0-4 GO INTO Z80
              0344 ; ROUTINES PER JUMP TABLE AT USR0.  CONTROL THEN RETURNS
              0345 ; TO SIMULATOR UPON Z80 'RET' (C9).  ALL OTHERS PROCESSED
              0346 ; NORMALLY.
              0347 ;
7162  23      0348 PJSR     INC     HL              ; PROCESS JSR
7163  CDA471  0349          CALL    SPCTST          ; TEST FOR SPECIALS
7166  2005    0350          JR      NZ,PJS2-$
7168  CDB271  0351          CALL    INUSR           ; GO TO USER ROUTINES
716B  1811    0352          JR      PJS3-$
716D  CD0475  0353 PJS2     CALL    TSPY            ; NORMAL JSR
7170  FD7400  0354          LD      (IY+0),H        ; PUSH CURRENT PC
7173  FD2B    0355          DEC     IY
7175  FD7500  0356          LD      (IY+0),L
7178  FD2B    0357          DEC     IY
717A  CD0D75  0358          CALL    TYSP
717D  EB      0359          EX      DE,HL           ; NEW PC
717E  017302  0360 PJS3     LD      BC,273H
7181  C3D774  0361          JP      END1
7184  23      0362 PJABS    INC     HL              ; ABS JUMP
7185  016F02  0363          LD      BC,26FH
7188  1808    0364          JR      DOJP-$
718A  23      0365 PJIND    INC     HL              ; INDIRECT JUMP
718B  5E      0366          LD      E,(HL)          ; JUMP POINTER
718C  23      0367          INC     HL
718D  56      0368          LD      D,(HL)
718E  EB      0369          EX      DE,HL           ; ADDR OF POINTER TO HL
718F  016F0C  0370          LD      BC,0C6FH
7192  CDA471  0371 DOJP     CALL    SPCTST          ; TEST FOR SPECIALS
7195  2009    0372          JR      NZ,DOJ1-$
7197  C5      0373          PUSH    BC
7198  CDB271  0374          CALL    INUSR           ; FOR SPECIALS
719B  CDF671  0375          CALL    PRET            ; DO SR RETURN
719E  C1      0376          POP     BC
719F  EB      0377          EX      DE,HL
71A0  EB      0378 DOJ1     EX      DE,HL           ; NEW PC
71A1  C3D774  0379          JP      END1
              0380 ;
              0381 ; TEST FOR USER SUBROUTINE CALLS/JUMPS.  IF TARGET
              0382 ; ADDRESS < 0005 THEN RET WITH Z FLG SET, ELSE CLR
              0383 ;
71A4  5E      0384 SPCTST   LD      E,(HL)          ; TEST FOR USER SPECIALS
71A5  23      0385          INC     HL              ; TARGET TO DE
71A6  56      0386          LD      D,(HL)          ; HI BYTE
71A7  23      0387          INC     HL              ; NXT INSTR
71A8  3E00    0388          LD      A,0             ; TARGET < 0005?
71AA  B2      0389          OR      D
71AB  C0      0390          RET     NZ
71AC  3E04    0391          LD      A,4
71AE  BB      0392          CP      E
71AF  D8      0393          RET     C
71B0  AF      0394          XOR     A               ; SET Z FLAG
71B1  C9      0395          RET
              0396 ;
              0397 ; INDEX INTO USER SUBROUTINE TABLE
              0398 ; PASS A, X, & Y TO/FROM USER ROUTINE AS A, B, & C.
              0399 ;
71B2  C5      0400 INUSR    PUSH    BC              ; SAVE CODES
71B3  1600    0401          LD      D,0
71B5  7B      0402          LD      A,E
71B6  83      0403          ADD     A,E
71B7  83      0404          ADD     A,E
71B8  5F      0405          LD      E,A
71B9  FD21467A 0406         LD      IY,USR0
71BD  01CE71  0407          LD      BC,USRET        ; RETURN ON STACK
71C0  C5      0408          PUSH    BC
71C1  FD19    0409          ADD     IY,DE
71C3  DD7E00  0410          LD      A,(IX+0)        ; PASS A,X, & Y
71C6  DD4602  0411          LD      B,(IX+2)
71C9  DD4E03  0412          LD      C,(IX+3)
71CC  FDE9    0413          JP      (IY)            ; AND GO
71CE  DD7700  0414 USRET    LD      (IX+0),A        ; RETURN HERE
71D1  DD7002  0415          LD      (IX+2),B        ; RETURN PARAMETERS
71D4  DD7103  0416          LD      (IX+3),C
71D7  C1      0417          POP     BC
```

```
71D8  C9        0418          RET
                0419 ;
                0420 ; PROCESS SR AND INTERRUPT RETURNS
                0421 ;
71D9  CD0475    0422 PRTI     CALL    TSPY            ; RTI
71DC  FD23      0423          INC     IY
71DE  FD7E00    0424          LD      A,(IY+0)        ; RESTORE FLAGS
71E1  DD7701    0425          LD      (IX+1),A
71E4  CDF971    0426          CALL    PRET1           ; RESTORE PC
71E7  01A504    0427          LD      BC,4A5H
71EA  C3D774    0428          JP      END1
71ED  CDF671    0429 PRTS     CALL    PRET            ; RTS...RESTORE PC
71F0  01A904    0430          LD      BC,4A9H
71F3  C3D774    0431          JP      END1
71F6  CD0475    0432 PRET     CALL    TSPY            ; SP TO IY
71F9  FD23      0433 PRET1    INC     IY              ; POINT TO LAST ENTRY
71FB  FD6E00    0434          LD      L,(IY+0)        ; RESTORE PCL
71FE  FD23      0435          INC     IY
7200  FD6600    0436          LD      H,(IY+0)        ; RESTORE PCH
7203  C30D75    0437          JP      TYSP            ; RESTORE NEW SP
                0438 ;
                0439 ; PROCESS SP EXCHANGES
                0440 ;
7206  FE90      0441 PTSS     CP      90H             ; WHICH ONE IS IT?
7208  200C      0442          JR      NZ,PTSX-$
720A  DD7E02    0443 PTXS     LD      A,(IX+2)        ; MOVE DATA FROM X...
720D  DD7704    0444          LD      (IX+4),A        ; TO SP
7210  01D904    0445          LD      BC,4D9H
7213  C3D674    0446          JP      END
7216  DD7E04    0447 PTSX     LD      A,(IX+4)        ; MOVE SP
7219  DD7702    0448          LD      (IX+2),A        ; TO X REG
721C  0ED1      0449          LD      C,0D1H
721E  C3E372    0450          JP      COMPL           ; AND MODIFY FLAGS
                0451 ;
                0452 ; PROCESS ALL REMAINING IMPLIED INSTRUCTIONS
                0453 ;
7221  4F        0454 PIMPL    LD      C,A             ; OP CODE
7222  CB39      0455          SRL     C               ; GET MSD
7224  CB39      0456          SRL     C
7226  CB39      0457          SRL     C
7228  CB39      0458          SRL     C
722A  E60F      0459          AND     0FH             ; CHECK LSD
722C  FE08      0460          CP      8
722E  FD215C7C  0461          LD      IY,IMPTBL       ; INDEX INTO TABLE
7232  CADA75    0462          JP      Z,JPIB
7235  FD21647C  0463          LD      IY,IMPTBL+8
7239  C3DA75    0464          JP      JPIB
                0465 ;
                0466 ; IMPLIED INSTR PROCESSORS
                0467 ;
723C  DD3402    0468 PINX     INC     (IX+2)          ; INX
723F  0E65      0469          LD      C,65H
7241  1813      0470          JR      IDEFLG-$
7243  DD3403    0471 PINY     INC     (IX+3)          ; INY
7246  0E69      0472          LD      C,69H
7248  180C      0473          JR      IDEFLG-$
724A  DD3502    0474 PDEX     DEC     (IX+2)          ; DEX
724D  0E55      0475          LD      C,55H
724F  1805      0476          JR      IDEFLG-$
7251  DD3503    0477 PDEY     DEC     (IX+3)          ; DEY
7254  0E59      0478          LD      C,59H
7256  DDCB018E  0479 IDEFLG   RES     1,(IX+1)        ; SET/RES Z FLG
725A  2004      0480          JR      NZ,SINTST-$     ; PER Z80 FLAGS
725C  DDCB01CE  0481          SET     1,(IX+1)
7260  DDCB01BE  0482 SINTST   RES     7,(IX+1)        ; ALSO SIGN FLG
7264  F21373    0483          JP      P,ENDIT
7267  DDCB01FE  0484          SET     7,(IX+1)
726B  C31373    0485          JP      ENDIT
726E  018504    0486 PNOP     LD      BC,485H         ; NOP
7271  C3D674    0487          JP      END
7274  CD0475    0488 PPHA     CALL    TSPY            ; PHA
7277  DD7E00    0489          LD      A,(IX+0)        ; ACCUM
727A  0E8D      0490          LD      C,8DH
727C  1808      0491          JR      DOPUSH-$        ; PUSH IT
727E  CD0475    0492 PPHP     CALL    TSPY            ; PHP
7281  DD7E01    0493          LD      A,(IX+1)        ; P REG
7284  0E91      0494          LD      C,91H
7286  C5      0495 DOPUSH   PUSH    BC
7287  FD7700    0496          LD      (IY+0),A        ; PLACE ON STACK
728A  FD2B      0497          DEC     IY              ; BUMP SP DOWN
728C  CD0D75    0498          CALL    TYSP            ; & REPLACE IT
728F  C1        0499          POP     BC
7290  C31373    0500          JP      ENDIT
7293  CDB472    0501 PPLA     CALL    DOPULL          ; PLA
7296  DD7700    0502          LD      (IX+0),A        ; DATA TO ACC
7299  CDED74    0503          CALL    SINS1           ; SET/RES S,Z FLGS
729C  CD0D75    0504          CALL    TYSP            ; RESTORE SP
729F  019504    0505          LD      BC,495H
72A2  C3D674    0506          JP      END
72A5  CDB472    0507 PPLP     CALL    DOPULL          ; PLP
72A8  DD7701    0508          LD      (IX+1),A        ; DATA TO P REG
72AB  CD0D75    0509          CALL    TYSP            ; RESTORE SP
72AE  019904    0510          LD      BC,499H
72B1  C3D674    0511          JP      END
72B4  CD0475    0512 DOPULL   CALL    TSPY            ; GET SP
72B7  FD23      0513          INC     IY              ; BUMP SP
72B9  FD7E00    0514          LD      A,(IY+0)        ; GET DATA
72BC  C9        0515          RET
                0516 ;
                0517 ; PROCESS ACC/INDEX TRANSFERS
                0518 ;
72BD  DD7E00    0519 PTAX     LD      A,(IX+0)        ; TAX
72C0  DD7702    0520          LD      (IX+2),A
72C3  0EC9      0521          LD      C,0C9H
72C5  181C      0522          JR      COMPL-$
72C7  DD7E00    0523 PTAY     LD      A,(IX+0)        ; TAY
72CA  DD7703    0524          LD      (IX+3),A
72CD  0ECD      0525          LD      C,0CDH
72CF  1812      0526          JR      COMPL-$
72D1  DD7E02    0527 PTXA     LD      A,(IX+2)        ; TXA
72D4  DD7700    0528          LD      (IX+0),A
72D7  0ED5      0529          LD      C,0D5H
72D9  1808      0530          JR      COMPL-$
72DB  DD7E03    0531 PTYA     LD      A,(IX+3)        ; TYA
72DE  DD7700    0532          LD      (IX+0),A
72E1  0EDD      0533          LD      C,0DDH
72E3  CDED74    0534 COMPL    CALL    SINS1           ; SET/RES S,Z FLGS
72E6  182B      0535          JR      ENDIT-$
                0536 ;
                0537 ; PROCESS FLAG SET/RES INSTRUCTIONS
                0538 ;
72E8  FD214C7C  0539 PFLGS    LD      IY,FLGTBL       ; TABLE START
72EC  0600      0540          LD      B,0
72EE  FEB8      0541          CP      0B8H            ; CLV SPECIAL
72F0  2002      0542          JR      NZ,PFL1-$
```

(Continued on pg. 30)