A KIM-1 DISASSEMBLER

Received: 77 Oct 28

I have just got my KIM1 up and running with a FORETHOUGHT motherboard kit, and an 8K ECONORAM kit from GODBOUT. I was impressed with the quality of both kits. The ECONORAM kit was a little too crowded. I had to unsolder some of the despiking capacitors that were too fat to fit between the 2102s. Next time, I will put the thin ones in these locations.

My first job was to get "A 6502 DISASSEMBLER FROM APPLE", (*DDJ* VOL 1 no. 8) to work. I had more than a little trouble, till I discovered that the KIM1 Outch and Prtbyt routines were clobbering the Y register that the disassembler was using as a counter. You might like to publish my version of this program which is enclosed. If anyone wants to avoid the problem of hand keying it in, I would be glad to furnish a copy on paper tape for $1.00 or on cassette for $3.00.

I am mainly interested in engineering applications. I would think that the micro computer is so cheap now that it would displace many applications now on time-sharing. Of course, the answer is in the cost of programming. That is where we computer buffs could perform a real service and have a good time doing it. Time sharing is good but the overhead is high. There is nothing like having your own computer right there when you want it.

I would like to use BASIC, but I fear it will be too slow for some of my problems. For example, a single problem in PIPE STRESS may need to invert a 6 X 7 matrix 20 times. If we could only debug with a small problem on BASIC, and then compile the source tape into machine language for running large problems. The construction of such a compiler is far beyond my capability. Does anyone market a BASIC with floating point, and arrays that will run on my KIM1?

To work on any big engineering program, you need at least two people. One must be knowledgeable with the problem. You need feedback in working up the documentation, and in other ways too. I am 70, retired and available. I have a degree and 40 years experience in mechanical engineering. I can run a typewriter. Let me know if I can help you with a problem. I submitted the following programs to the DECUS library: PIPE STRESS, CRITICAL PATH METHOD, REGRESSION ANALYSIS, BUILDING HEAT LOSS. Having only 4 K of core on my PDP8f that I recently sold, BASIC was out of the question. Having no feedback from anybody, my documentation was bad, in spite of very great effort.

Very truly yours,
Theodore E. Bridge

54 Williamsburg Dr.
Springfield, MA 01108

PS In writing the above mentioned programs for the DECUS library, I made good use of a FLOATING POINT INTERPRETER in the DECUS library. It uses much less memory than BASIC and runs very much faster. Would anyone like to help me write a similar interpreter for my KIM1? I worked up a trace routine for the interpreter. I grant that debugging with BASIC would be much easier. Also only a few people can communicate with you if you are using your own interpreter. However, a BASIC interpreter is not really useable on a very large engineering application.

```
A 6502 DISASSEMBLER FROM APPLE -- PUBLISHED BY
STEVE WOZNIAK & ALLEN BAUM IN VOLUME 1, NUMBER 8 OF
DR. DOBB'S JOURNAL. MODIFIED BY T.E.BRIDGE FOR KIM1.
THIS LISTING WAS PRODUCED BY ITSELF.
A
KIM
0044 00       PCL = addr. of next inst.
0045 02 3DC   PCH   to be disassembled
03DC 20 G     set BEGIN addr. and GO
0200-   A9 FF          LDA   #$FF   set line
0202-   85 46          STA   $46    count
0204-   20 12 02       JSR   $0212
0207-   20 EF 02       JSR   $02EF  update
020A-   85 44          STA   $44    program
020C-   84 45          STY   $45    counter
020E-   C6 46          DEC   $46
0210-   D0 F2          BNE   $0204
0212-   20 D3 02 in    JSR   $02D3
0215-   A1 44          LDA   ($44,X)
0217-   A8             TAY
0218-   4A             LSR
0219-   90 0B          BCC   $0226
021B-   4A             LSR
021C-   B0 17          BCS   $0235
021E-   C9 22          CMP   #$22
0220-   F0 13          BEQ   $0235
0222-   29 07          AND   #$07
0224-   09 80          ORA   #$80
0226-   4A      iven   LSR
0227-   AA             TAX
0228-   BD FE 02       LDA   $02FE,X
022B-   B0 04          BCS   $0231
022D-   4A             LSR
022E-   4A             LSR
022F-   4A             LSR
0230-   4A             LSR
0231-   29 0F rtmd     AND   #$0F
0233-   D0 04          BNE   $0239
0235-   A0 80 err      LDY   #$80
0237-   A9 00          LDA   #$00
0239-   AA      getf   TAX
023A-   BD 42 03       LDA   $0342,X
023D-   85 40          STA   $40
023F-   29 03          AND   #$03
0241-   85 41          STA   $41
0243-   98             TYA
0244-   29 8F          AND   #$8F
0246-   AA             TAX
0247-   98             TYA
0248-   A0 03          LDY   #$03
024A-   E0 8A          CPX   #$8A
024C-   F0 0B          BEQ   $0259
024E-   4A      mnd1   LSR
024F-   90 08          BCC   $0259
0251-   4A             LSR
0252-   4A      mnd2   LSR
0253-   09 20          ORA   #$20
0255-   88             DEY
0256-   D0 FA          BNE   $0252
0258-   C8             INY
0259-   88      mnd3   DEY
025A-   D0 F2          BNE   $024E
025C-   48             PHA
025D-   B1 44 prop     LDA   ($44),Y
025F-   20 EA 03       JSR   $03EA
0262-   A2 01          LDX   #$01
0264-   20 E6 02       JSR   $02E6
0267-   C4 41          CPY   $41
0269-   C8             INY
026A-   90 F1          BCC   $025D
026C-   A2 03          LDX   #$03
026E-   C0 04          CPY   #$04
0270-   90 F2          BCC   $0264
0272-   68             PLA

0273-   A8             TAY
0274-   B9 5C 03       LDA   $035C,Y
0277-   85 42          STA   $42
0279-   B9 9C 03       LDA   $039C,Y
027C-   85 43          STA   $43
027E-   A9 00 pmn1     LDA   #$00
0280-   A0 05          LDY   #$05
0282-   06 43 pmn2     ASL   $43
0284-   26 42          ROL   $42
0286-   2A             ROL
0287-   88             DEY
0288-   D0 F8          BNE   $0282
028A-   69 BF          ADC   #$BF
028C-   20 E2 03       JSR   $03E2
028F-   CA             DEX
0290-   D0 EC          BNE   $027E
0292-   20 E4 02       JSR   $02E4
0295-   A2 06          LDX   #$06
0297-   E0 03 pad1     CPX   #$03
0299-   D0 12          BNE   $02AD
029B-   A4 41          LDY   $41
029D-   F0 0E          BEQ   $02AD
029F-   A5 40 pad2     LDA   $40
02A1-   C9 E8          CMP   #$E8
02A3-   B1 44          LDA   ($44),Y
02A5-   B0 1C          BCS   $02C3
02A7-   20 EA 03       JSR   $03EA
02AA-   88             DEY
02AB-   D0 F2          BNE   $029F
02AD-   06 40 pad3     ASL   $40
02AF-   90 0E          BCC   $02BF
02B1-   BD 4F 03       LDA   $034F,X
02B4-   20 E2 03       JSR   $03E2
02B7-   BD 55 03       LDA   $0355,X
02BA-   F0 03          BEQ   $02BF
02BC-   20 E2 03       JSR   $03E2
02BF-   CA      pad4   DEX
02C0-   D0 D5          BNE   $0297
02C2-   60             RTS
02C3-   20 F2 02 ra    JSR   $02F2
02C6-   AA             TAX
02C7-   E8             INX
02C8-   D0 01          BNE   $02CB
02CA-   C8             INY
02CB-   98      pty    TYA
02CC-   20 EA 03 ptx   JSR   $03EA
02CF-   8A      pntx   TXA
02D0-   4C EA 03       JMP   $03EA
02D3-   A9 8D pcr      LDA   #$8D
02D5-   20 2F 1E       JSR   $1E2F
02D8-   A5 45          LDA   $45
02DA-   A6 44          LDX   $44
02DC-   20 CC 02       JSR   $02CC
02DF-   A9 AD          LDA   #$AD
02E1-   20 E2 03       JSR   $03E2
02E4-   A2 03 pb       LDX   #$03
02E6-   A9 A0 pb2      LDA   #$A0
02E8-   20 E2 03 pb3   JSR   $03E2
02EB-   CA             DEX
02EC-   D0 F8          BNE   $02E6
02EE-   60             RTS
02EF-   A5 41 pj       LDA   $41
02F1-   38      pj2    SEC
02F2-   A4 45 pj3      LDY   $45
02F4-   AA             TAX
02F5-   10 01          BPL   $02F8
02F7-   88             DEY
02F8-   65 44 pj4      ADC   $44
02FA-   90 01          BCC   $02FD
02FC-   C8             INY
02FD-   60      rt1    RTS
02FE-   40             RTI
02FF-   02             ???
```

more

KICKING THE 8080 HABIT

Maybe it's just my imagination, but it seems that a lot of people aren't utilizing the Z-80 to its fullest. Everyone is so used to writing code for the 8080 that they don't seem to bother upgrading their software when they upgrade their CPU. Or maybe, as in my case, since we've been eating and sleeping in Intel Octal for so long that we just can't seem to catch on to all the enhancements of the added Z-80 functions and instructions. I would like to see you guys (or, someone else) come out and explain all the nifty Z-80 tricks. I know I can't be the only one that is stuck in the rut of 8080 code. (Please!! Don't tell me I swapped my CPU board JUST for speed—the software potential is fantastic.)

Michael Settle   751 Washington #115
                           Arlington, TX 76011

How about it? Anyone care to send us something on "Taking full advantage of Z-80's inner mysteries"?—TRW

MICROCOMPUTER SOFTWARE INDEX

A great quantity of software has been published in microcomputer books and magazines since late 1975. These programs range from complete BASICs to assemblers, routines, games and direct application programs.

The Schreier Software Index, An Index to Published Microcomputer Software, indexes hundreds of published microcomputer programs. The S S I contains cross references and over 130 program divisions. Many of the cited programs specify chip description. In addition, the S S I features complete publication documentation and bibliographic data. Locating a random number routine for a 6800 or an 8080, for example, takes but seconds.

Direct orders may be placed with S S I, 4327 East Grove Street, Phoenix, Arizona 85040. Price postpaid in the United States, Mexico and Canada is $5.00. The S S I is also available from a number of microcomputer dealers across the U.S.

```
THESE CHANGES WERE ADDED TO SAVE
THE Y REGISTER WHEN CALLING
KIM ROUTINES  OUTCH & PRTBYT

KIM
03DC 20 G
03DC-    20 00 02    JSR    $0200  begin
03DF-    4C 4F 1C    JMP    $1C4F  start
03E2-    84 EE       STY    $EE    savey
03E4-    20 A0 1E    JSR    $1EA0  outch
03E7-    A4 EE       LDY    $EE    gety
03E9-    60          RTS
03EA-    84 EE       STY    $EE    savey
03EC-    20 3E 1E    JSR    $1E3E  prtbyt
03EF-    A4 EE       LDY    $EE    gety
03F1-    60          RTS

THESE ARE THE TABLES

KIM
03DC 20 44
0044 F2 DC.
0045 03 3DC
03DC 20 17F7
17F7 FF DC.    end of tables
17F8 FF 83.
17F9 1A 0300   start of tables
0300 45 C
;101 0004563D00B4009302245330C00B4009400245330008400940000058A
;19631B40EED0004000002244330D08C44CE11284433D08C449A10020782
;18033F4433D0004000182246330D0084009621373A90002182000000667
;18034B594D91902864AA59DAC49ACA0ABA4D900D8AAA4001C8A1C23000C
;18036005D8E1EA19D8A1D839D8E1DA10C0919AE69A8192324531E23085E
;18037824531 9A1001A5E5EA569242AAEAEA8AD09007C000159C6D000085E
;18039CA56929508413341 A569200A7D8605A48266294805444C854CA16
;1803A006544B94C0E4E88474C4286E7AF4CC4A72F2A48A00AAA2A20CE7
;1803C07474747246E8E2320200022001A1A260072708008C4CA264000997
;1803L64444A0C0200C0024C4F1C04EE20CAC1EA4EE6004EF203L1EA40A8F
;000006A000A
```

VDM-1 DRIVER MODS   by John Moorhead

```
0005 *  THIS IS THE VDM-1 DRIVER ORIGINATED
0010 *  BY LI CHEN WANG (DR. DOBB'S JOURNAL,
0015 *  VOLUME #1, ISSUE #6).
0020 *
0025 *  MODIFIED TO INCLUDE SPEED CONTROL AND
0030 *  GENERALLY EMULATE THE SOFTWARE DRIVER FOR
0035 *  PROCESSOR TECHNOLOGY'S VDM-1 VIDEO DISPLAY.
0040 *  THIS PROGRAM WILL ALSO ACT AS THE DRIVER
0045 *  FOR SOLID STATE MUSIC'S VB-1 DISPLAY.
0050 *
0055 *  WRITTEN BY JOHN MOORHEAD, 928 J STREET,
0060 *  DAVIS, CA.   95616   TEL NO 916-758-2495
0065 *
0070 *  MAY 5, 1977
0075 *
0080 *  BACKSPACE = CONTROL H OR B.S. KEY
0085 *  SCREEN CLEAR = "CANCEL" = CTRL-X
0090 *  AUTO CRLF ON INPUT OF A CARRIAGE RETURN
0095 *  CHANGE SPEED BY TYPING NUMBER DURING OUTPUT
0100 *  STOP DISPLAY BY TYPING SPACE BAR
0105 *  START DISPLAY BY TYPING ANY CHARACTER
0110 *
0115 *  NOTE - A SCREEN CLEAR MUST BE THE FIRST
0120 *  CHARACTER SENT IN ORDER TO INITIALIZE THE
0125 *  CURSOR AT THE BOTTOM LEFT CORNER OF THE
0130 *  SCREEN AND ENSURE PROPER SCROLLING.
0135 *
0140 *
0145 *  ENTER WITH CHARACTER IN THE ACCUMULATOR
0150 *
0155 *
0160 STRT   PUSH   H      SAVE SYSTEM REGISTERS
0165        PUSH   D
0170        PUSH   B
0175        PUSH   PSW    CHAR IS IN THE ACCUM
0180        LHLD   VDMP   GET SCREEN POSITION POINTER
0185        CPI    0DH    IS THE CHAR A CARRIAGE RET?
0190        JZ     CR     YES - SCROLL, OUTPUT A CR
0195        CPI    08H    IS IT A BACKSPACE?
0200        JZ     BS     YES - MOVE CURSOR BACK
0205        CPI    18H    IS IT A CTRL - X (SCREEN CLEAR)?
0210        JZ     CLEAR  YES - ERASE ENTIRE SCREEN
0215        CPI    7FH    IS IT A RUBOUT CHARACTER?
0220        JZ     SPEED  DON'T DISPLAY (FOR ALS-8 ONLY)
0225        CPI    20H    DON'T DISPLAY CONTROL CHARS
0230        JC     SPEED  EXIT TO CHANGE SPEED
0235        MOV    M,A    IT HAS TO BE DATA
0240        INX    H      UPDATE CHAR POSITION ON SCREEN
0245        MVI    M,0A0H PUT CURSOR ON SCREEN
0250        JMP    LINOV  TEST FOR LINE OVERFLOW
0255 BS     MVI    M,20H  REMOVE CURSOR
0260        DCX    H      BACK UP POINTER
0265        JMP    BS-5
0270 CR     MVI    M,20H  CHAR IS A CARRIAGE RETURN
0275        MOV    A,L    UPDATE NEXT CHAR POSITION
0280        ANI    0C0H
0285        ADI    40H    SETTING UP FOR NEW LINE
0290        MOV    L,A    ADDRESS OF NEW LINE
0295        MVI    A,0
0300        ADC    H      ADD WITH CARRY
0305        MOV    H,A
0310 LINOV  SHLD   VDMP   SAVE POINTER FOR NEXT CHAR
0315        MVI    A,7FH
0320        ANA    L
0325        JNZ    SPEED  EXIT
0330        MVI    M,20H
0335        LXI    H,0CFC0H  SET UP DATA LINE (LINE 16)
0340        SHLD   VDMP   SAVE POINTER
0345        LXI    H,0CC40H  15 LINES OF SCREEN DATA
0350        LXI    D,0CC00H  TOP OF SCREEN.  SET UP
0355        LXI    B,03C0H   TO SCROLL 15 LINES
0360 SCROL  MOV    A,M    START SCROLLING UP
```