4.3 Instant Input Assembler (IIA)

Programming microprocessors is a long and involved process- edit, assemble, load, and debug. During the debug process it is often necessary to patch the program rather than reassemble it. Patching is often avoided, however, because it is time consuming and error prone - nobody likes the idea of programming in hexadecimal! The Instant Input Assembler (trade mark, Technico Inc.) was designed to circumvent this problem. The IIA provides the capability to enter patches or short programs using the standard 9900 CPU mnemonics and operands. The best part is that it is available in EPROM so its always ready for use. The Instant Input Assembler is only 512 words long, but it is packed with features to simplify programming.

4.3.1 Operation

To start execution of the assembler, branch to location #F800 using the monitors GO command or the disk handler command IIA. The Instant Input Assembler will respond as follows:

```
0100:
```

The address printed is the address where your program will be entered in memory. The assembler always prompts by printing an address on a new line, and then it waits for you to type a source input statement. When the source statement is completed and you type a carriage return, the assembler will display the converted code that it has entered in memory; advance the program address accordingly; and again prompt by printing the address. A sample entry sequence is shown later in this section.

4.3.2 Entry Format

When the assembler is awaiting input, you may enter any one of the following types of commands:

(1) string constant - Type a "$" followed by characters that you want entered in memory. The assembler will convert each character from ASCII to hexadecimal and store the constants in memory. The string may be any length and is terminated by a carriage return. If the string is an odd number of bytes long, the assembler will add one more byte filled with a space to return to an even address.

(2) numeric constant - A numeric constant is indicated by typing a "+" or "-" followed by the desired constant. All constants are decimal unless otherwise indicated. To indicate a hexadecimal constant, preceed the number by a ">" (e.g.+>1234). To indicate a binary constant, preceed the

number by a "%" (e.g.+%10100011100)

(3) address redefinition - You can change the program counter location by typing a "/" followed by the new address (in hexadecimal). This feature is ideal for patching because it allows you to move about in memory without restarting the assembler.

(4) 9900 CPU assembly mnemonic - All of the 9900 CPU mnemonics are recognized by the instant input assembler.

The general format of an instruction entry is:

<instruction-mnemonic> space <operand-field> space

The complete set of allowable instruction mnemonics is described in the Instruction Set section of the system manual. There are several different operand fields that may be used:

a) register - for example: R0, R13, 12, 3

b) register indirect - for example: *R15, *6

c) register indirect with auto increment - for example: *R14+, *2+

d) indexed - for example: @12(R1), @123, @%101, @>12(3)

e) constant - for example: 12, %101, >123A

f) string - for example: 'A', 'BD'

g) relative displacement (for jumps) - The displacement is in words and is checked for allowable range. There are three possible formats: +N, -N, or N. The +/- format is a jump forward or backward N words from the next sequential word. The N format is a jump to address N, the assembler will calculate the offset in this case.

Refer to the Instruction Set section of the system maual for what operands are allowed with any given instruction. Sample instructions and mnemonics are:

```
?GF800


0100: C941  MOV R1,@%101(R5)
0102: 0005
0104: C820  MOV @2,@>123
0106: 0002
```

```
0108: 0123
010A: 1002   JMP +2
010C: 10F9   JMP >100
0110: CC91   MOV *R1,*R2+
0112:
```

The easiest way to learn to use the instant input assembler is to try a sample program. The following program will type the message "HELLO" on a new line and then return to the monitor. It uses the monitor XOP's for input/output. Try entering this program and then returning to the monitor (by typing break) and executing it (G 100).

```
?GF800

0100: 0201   LI R1,>120          ; R1 IS ADDR. COUNTER
0102: 0120
0104: 2C91   OUT *R1             ; PRINT ONE CHARACTER
0106: 0581   INC R1              ; ADVANCE TO NEXT ONE
0108: D011   MOVB *R1,R0         ; TEST FOR END OF MESS.
010A: 16FC   JNE >104            ; IF NOT END, CONT.
010C: 2C00   XOP 0,0             ; IF END, BACK TO MONITOR
0110:        /120                ; CHANGE PC
0120: 0D0A   +>0D0A              ; RETURN/FEED
0122: 4845   $HELLO
0124: 4C4C
0126: 4F20
0128: 0000   +0
012A:
?G100

HELLO
?
```

4.3.3 Line Editing

If you make a typing mistake, you can backspace and correct it by typing a backspace (CTRL-H). The assembler will line feed and backup (assumming your terminal can backspace) under the character to be changed. If your mistake is a big one, type ESC (escape) and the assembler will discard this line and allow you to start over again.

4.3.4 Error Messages

The instant input assembler is only 512 words long, so it cannot detect all user errors, however it does detect many of them. Whenever you type something that the assembler does not like or understand, it will type an error message on the

next line and reprompt with the old address. The possible error messages are:

*S - syntax error. The input contains a syntax error.

*D - displacement error. The target address of a jump is too far away and exceeds the allowable range.

*R - range error. The input is out of range. It should have been 0 to 15.

When the assembler detects an error and prints the message, it keeps the program counter set to the location of the error. That is, the program counter is not advanced until a statement is accepted and stored in memory.

4.3.5 Program Design

The heart of the assembler is the unique tree structured op-code table. Since some assemblers require more space for the op-code table than was available for the whole assembler that table had to be optimized. A review of the listing will illustrate how this table is constructed. To further reduce space, the monitors XOP's are used for input/output. A full listing of the IIA is provided on the following pages.

We have done our very best to check and verify this program. However, you may find some latent error or have a suggestion for further improving its usefulness. If so, drop us a line describing the change. We are always interested in hearing from the users.

```
                        TITL 'INSTANT INPUT ASSEMBLER (TM) VER 3 - 10/78'
0000                IDTIIA IDT
                    *
                    * THE INSTANT INPUT ASSEMBLER(TRADE MARK-
                    * TECHNICO, INC.) IS DESIGNED TO RUN WITH THE
                    * MIGHTY MONITOR(TRADE MARK-TECHNICO, INC.)
                    *
                    * ENTIRE PROGRAM COPYRIGHT 1978, TECHNICO, INC.
                    * NO PORTION MAY BE REPRODUCED WITHOUT EXPRESS
                    * WRITTEN CONSENT
                    *
                    * INTERFACE TO MONITOR ROUTINES
                    *
                          REF  IDTMM,TYPE,TYPEWD,DMEMN
0000                      DREG
0000 0460 020A            B    @IIABGN            ; GOTO START
                    *
                    * THE FOLLOWING TRANSFER VECTOR IS
                    * INCLUDED SO THAT SOFTWARE WRITTEN FOR
                    * EARLIER VERSIONS OF THE SYSTEM WILL WORK
                    *
0004 00B0           DISK   DATA >B0               ; DTH (WP,PC)
0006 F010                  DATA >F010
0008 0460 F008      TTIN   B    @>F008            ; TAPE IN
000C 0460 F00C      TTOUT  B    @>F00C            ; TAPE OUT
                    *
                    * RAM AREA
                    *
00B0                PC     EQU  >B0               ; PROGRAM COUNTER
00B2                PTR    EQU  >B2               ; TEXT POINTER
0100                DFPC   EQU  >100              ; DEFAULT P.C.
                    *
                    * MNEMONIC TABLE.  THIS TABLE
                    * IS CONSTRUCTED AS A BINARY TREE.
                    * EACH ENTRY HAS THE CHARACTER POSITION AND
                    * THE CHARACTER.  IF THE SIGN BIT IS SET
                    * THE CHARACTER IS A LEGAL END OF OP-CODE.
                    * THE ASCII CHARACTER IS IN THE RIGHTMOST
                    * FIVE BITS.
                    *
0000                P1     EQU  0                 ; CHAR ONE
0020                P2     EQU  32                ; CHAR TWO
0040                P3     EQU  64                ; CHAR THREE
0060                P4     EQU  96                ; CHAR FOUR
0080                P1E    EQU  >80+P1            ; CHAR ONE & END
00A0                P2E    EQU  >80+P2            ; CHAR TWO & END
00C0                P3E    EQU  >80+P3            ; CHAR THREE & END
00E0                P4E    EQU  >80+P4            ; CHAR FOUR & END
0010 81             OPS    BYTE P1E+'A'-'@'       ; A S,D
0011 A2                    BYTE P2E+'B'-'@'       ; AB S,D
0012 D3                    BYTE P3E+'S'-'@'       ; ABS S
0013 A9                    BYTE P2E+'I'-'@'       ; AI W,IOP
0014 2E                    BYTE P2+'N'-'@'        ;
0015 44                    BYTE P3+'D'-'@'        ;
0016 E9                    BYTE P4E+'I'-'@'       ; ANDI W,IOP
0017 82                    BYTE P1E+'B'-'@'       ; B S
0018 AC                    BYTE P2E+'L'-'@'       ; BL S
0019 57                    BYTE P3+'W'-'@'        ;
```

```
001A F0                 BYTE P4E+'P'-'@'        ; BLWP S
001B 83                 BYTE P1E+'C'-'@'        ; C S,D
001C A2                 BYTE P2E+'B'-'@'        ; CB S,D
001D A9                 BYTE P2E+'I'-'@'        ; CI W,IOP
001E 2C                 BYTE P2+'L'-'@'         ;
001F D2                 BYTE P3E+'R'-'@'        ; CLR S
0020 2F                 BYTE P2+'O'-'@'         ;
0021 C3                 BYTE P3E+'C'-'@'        ; COC S,W
0022 3A                 BYTE P2+'Z'-'@'         ;
0023 C3                 BYTE P3E+'C'-'@'        ; CZC S,W
0024 04                 BYTE P1+'D'-'@'         ;
0025 25                 BYTE P2+'E'-'@'         ;
0026 C3                 BYTE P3E+'C'-'@'        ; DEC S
0027 F4                 BYTE P4E+'T'-'@'        ; DECT S
0028 29                 BYTE P2+'I'-'@'         ;
0029 D6                 BYTE P3E+'V'-'@'        ; DIV S,W
002A 09                 BYTE P1+'I'-'@'         ;
002B 24                 BYTE P2+'D'-'@'         ;
002C 4C                 BYTE P3+'L'-'@'         ;
002D E5                 BYTE P4E+'E'-'@'        ; IDLE
002E AE                 BYTE P2E+'N'-'@'        ; IN S
002F C3                 BYTE P3E+'C'-'@'        ; INC S
0030 F4                 BYTE P4E+'T'-'@'        ; INCT S
0031 D6                 BYTE P3E+'V'-'@'        ; INV S
0032            FEED    EQU  $                  ; LINE FEED
0032 0A                 BYTE P1+'J'-'@'         ;
0033 25                 BYTE P2+'E'-'@'         ;
0034 D1                 BYTE P3E+'Q'-'@'        ; JEQ DIS
0035 27                 BYTE P2+'G'-'@'         ;
0036 D4                 BYTE P3E+'T'-'@'        ; JGT DIS
0037 A8                 BYTE P2E+'H'-'@'        ; JH DIS
0038 C5                 BYTE P3E+'E'-'@'        ; JHE DIS
0039 AC                 BYTE P2E+'L'-'@'        ; JL DIS
003A C5                 BYTE P3E+'E'-'@'        ; JLE DIS
003B D4                 BYTE P3E+'T'-'@'        ; JLT DIS
003C 2D                 BYTE P2+'M'-'@'         ;
003D D0                 BYTE P3E+'P'-'@'        ; JMP DIS
003E 2E                 BYTE P2+'N'-'@'         ;
003F C3                 BYTE P3E+'C'-'@'        ; JNC DIS
0040 C5                 BYTE P3E+'E'-'@'        ; JNE DIS
0041 CF                 BYTE P3E+'O'-'@'        ; JNO DIS
0042 2F                 BYTE P2+'O'-'@'         ;
0043 C3                 BYTE P3E+'C'-'@'        ; JOC DIS
0044 D0                 BYTE P3E+'P'-'@'        ; JOP DIS
0045 0C                 BYTE P1+'L'-'@'         ;
0046 24                 BYTE P2+'D'-'@'         ;
0047 43                 BYTE P3+'C'-'@'         ;
0048 F2                 BYTE P4E+'R'-'@'        ; LDCR S,C
0049 A9                 BYTE P2E+'I'-'@'        ; LI W,IOP
004A 4D                 BYTE P3+'M'-'@'         ;
004B E9                 BYTE P4E+'I'-'@'        ; LIMI IOP
004C 37                 BYTE P2+'W'-'@'         ;
004D 50                 BYTE P3+'P'-'@'         ;
004E E9                 BYTE P4E+'I'-'@'        ; LWPI IOP
004F 0D                 BYTE P1+'M'-'@'         ;
0050 2F                 BYTE P2+'O'-'@'         ;
0051 D6                 BYTE P3E+'V'-'@'        ; MOV S,D
0052 E2                 BYTE P4E+'B'-'@'        ; MOVB S,D
```

```
0053 30                 BYTE P2+'P'-'@'         ;
0054 D9                 BYTE P3E+'Y'-'@'        ; MPY S,W
0055 0E                 BYTE P1+'N'-'@'         ;
0056 25                 BYTE P2+'E'-'@'         ;
0057 C7                 BYTE P3E+'G'-'@'        ; NEG S
0058 2F                 BYTE P2+'O'-'@'         ;
0059 D0                 BYTE P3E+'P'-'@'        ; NOP
005A 0F                 BYTE P1+'O'-'@'         ;
005B 32                 BYTE P2+'R'-'@'         ;
005C C9                 BYTE P3E+'I'-'@'        ; ORI W,IOP
005D 35                 BYTE P2+'U'-'@'         ;
005E D4                 BYTE P3E+'T'-'@'        ; OUT S
005F 12                 BYTE P1+'R'-'@'         ;
0060 34                 BYTE P2+'T'-'@'         ;
0061 57                 BYTE P3+'W'-'@'         ;
0062 F0                 BYTE P4E+'P'-'@'        ; RTWP
0063 93                 BYTE P1E+'S'-'@'        ; S S,D
0064 A2                 BYTE P2E+'B'-'@'        ; SB S,D
0065 CF                 BYTE P3E+'O'-'@'        ; SBO BIT
0066 DA                 BYTE P3E+'Z'-'@'        ; SBZ BIT
0067 25                 BYTE P2+'E'-'@'         ;
0068 54                 BYTE P3+'T'-'@'         ;
0069 EF                 BYTE P4E+'O'-'@'        ; SETO S
006A 2C                 BYTE P2+'L'-'@'         ;
006B C1                 BYTE P3E+'A'-'@'        ; SLA W,N
006C 2F                 BYTE P2+'O'-'@'         ;
006D C3                 BYTE P3E+'C'-'@'        ; SOC S,D
006E E2                 BYTE P4E+'B'-'@'        ; SOCB S,D
006F 32                 BYTE P2+'R'-'@'         ;
0070 C1                 BYTE P3E+'A'-'@'        ; SRA W,N
0071 C3                 BYTE P3E+'C'-'@'        ; SRC W,N
0072 CC                 BYTE P3E+'L'-'@'        ; SRL W,N
0073 34                 BYTE P2+'T'-'@'         ;
0074 43                 BYTE P3+'C'-'@'         ;
0075 F2                 BYTE P4E+'R'-'@'        ; STCR S,C
0076 53                 BYTE P3+'S'-'@'         ;
0077 F4                 BYTE P4E+'T'-'@'        ; STST W
0078 57                 BYTE P3+'W'-'@'         ;
0079 F0                 BYTE P4E+'P'-'@'        ; STWP W
007A 37                 BYTE P2+'W'-'@'         ;
007B 50                 BYTE P3+'P'-'@'         ;
007C E2                 BYTE P4E+'B'-'@'        ; SWPB S
007D 3A                 BYTE P2+'Z'-'@'         ;
007E C3                 BYTE P3E+'C'-'@'        ; SZC S,D
007F E2                 BYTE P4E+'B'-'@'        ; SZCB S,D
0080 14                 BYTE P1+'T'-'@'         ;
0081 A2                 BYTE P2E+'B'-'@'        ; TB BIT
0082 98                 BYTE P1E+'X'-'@'        ; X S
0083 2F                 BYTE P2+'O'-'@'         ;
0084 D0                 BYTE P3E+'P'-'@'        ; XOP S,W
0085 D2                 BYTE P3E+'R'-'@'        ; XOR S,W
0086 00                 BYTE 0                  ; END OF TABLE
                *
                * BRANCH TABLE FOR OPERANDS
                * 0 - N/A
                * 1 - S OR D
                * 2 - W OR C
                * 3 - IOP
```

```
                    * 4 - N (SHIFT COUNT)
                    * 5 - DIS
                    * 6 - BIT
                    *
0088 0000 031A      OP      DATA 0,OPA,OPF,OPE
008C 03A6 0396
0090 038A 03AE              DATA OPD,OPG,OPH
0094 03F8
                    *
                    * BASIC OP-CODE TABLE
                    * EACH ENTRY HAS THE OP CODE,
                    * OPERAND ONE AND OPERAND TWO
                    * DESCRIPTION.
                    *
0009                FM1     EQU  >9                 ; FORMAT 1 - S,D
0005                FM2     EQU  >5                 ; FORMAT 2 - DIS
000A                FM3     EQU  >A                 ; FORMAT 3 - S,W
000A                FM4     EQU  >A                 ; FORMAT 4 - S,C
0014                FM5     EQU  >14                ; FORMAT 5 - W,N
0008                FM6     EQU  >8                 ; FORMAT 6 - S
0000                FM7     EQU  0                  ; FORMAT 7 - N/A
0013                FM8     EQU  >13                ; FORMAT 8 - W,IOP
000A                FM9     EQU  >A                 ; FORMAT 9 - S,W
0006                FMA     EQU  >6                 ; FORMAT A - BIT
0003                FMB     EQU  >3                 ; FORMAT B - IOP
0010                FMC     EQU  >10                ; FORMAT C - W
0096 A009           CODE    DATA >A000+FM1          ; A
0098 B009                   DATA >B000+FM1          ; AB
009A 0748                   DATA >0740+FM6          ; ABS
009C 0233                   DATA >0220+FM8          ; AI
009E 0253                   DATA >0240+FM8          ; ANDI
00A0 0448                   DATA >0440+FM6          ; B
00A2 0688                   DATA >0680+FM6          ; BL
00A4 0408                   DATA >0400+FM6          ; BLWP
00A6 8009                   DATA >8000+FM1          ; C
00A8 9009                   DATA >9000+FM1          ; CB
00AA 0293                   DATA >0280+FM8          ; CI
00AC 04C8                   DATA >04C0+FM6          ; CLR
00AE 200A                   DATA >2000+FM3          ; COC
00B0 240A                   DATA >2400+FM3          ; CZC
00B2 0608                   DATA >0600+FM6          ; DEC
00B4 0648                   DATA >0640+FM6          ; DECT
00B6 3C0A                   DATA >3C00+FM9          ; DIV
00B8 0340                   DATA >0340+FM7          ; IDLE
00BA 2C48                   DATA >2C40+FM6          ; IN
00BC 0588                   DATA >0580+FM6          ; INC
00BE 05C8                   DATA >05C0+FM6          ; INCT
00C0 0548                   DATA >0540+FM6          ; INV
00C2 1305                   DATA >1300+FM2          ; JEQ
00C4 1505                   DATA >1500+FM2          ; JGT
00C6 1B05                   DATA >1B00+FM2          ; JH
00C8 1405                   DATA >1400+FM2          ; JHE
00CA 1A05                   DATA >1A00+FM2          ; JL
00CC 1205                   DATA >1200+FM2          ; JLE
00CE 1105                   DATA >1100+FM2          ; JLT
00D0 1005                   DATA >1000+FM2          ; JMP
00D2 1705                   DATA >1700+FM2          ; JNC
00D4 1605                   DATA >1600+FM2          ; JNE
```

```
00D6 1905                DATA >1900+FM2          ; JNO
00D8 1805                DATA >1800+FM2          ; JOC
00DA 1C05                DATA >1C00+FM2          ; JOP
00DC 300A                DATA >3000+FM4          ; LDCR
00DE 0213                DATA >0200+FM8          ; LI
00E0 0303                DATA >0300+FMB          ; LIMI
00E2 02E3                DATA >02E0+FMB          ; LWPI
00E4 C009                DATA >C000+FM1          ; MOV
00E6 D009                DATA >D000+FM1          ; MOVB
00E8 380A                DATA >3800+FM9          ; MPY
00EA 0508                DATA >0500+FM6          ; NEG
00EC 1000                DATA >1000+FM7          ; NOP
00EE 0273                DATA >0260+FM8          ; ORI
00F0 2C88                DATA >2C80+FM6          ; OUT
00F2 0380                DATA >0380+FM7          ; RTWP
00F4 6009                DATA >6000+FM1          ; S
00F6 7009                DATA >7000+FM1          ; SB
00F8 1D06                DATA >1D00+FMA          ; SBO
00FA 1E06                DATA >1E00+FMA          ; SBZ
00FC 0708                DATA >0700+FM6          ; SETO
00FE 0A14                DATA >0A00+FM5          ; SLA
0100 E009                DATA >E000+FM1          ; SOC
0102 F009                DATA >F000+FM1          ; SOCB
0104 0814                DATA >0800+FM5          ; SRA
0106 0B14                DATA >0B00+FM5          ; SRC
0108 0914                DATA >0900+FM5          ; SRL
010A 340A                DATA >3400+FM4          ; STCR
010C 02D0                DATA >02C0+FMC          ; STST
010E 02B0                DATA >02A0+FMC          ; STWP
0110 06C8                DATA >06C0+FM6          ; SWPB
0112 4009                DATA >4000+FM1          ; SZC
0114 5009                DATA >5000+FM1          ; SZCB
0116 1F06                DATA >1F00+FMA          ; TB
0118 0488                DATA >0480+FM6          ; X
011A 2C0A                DATA >2C00+FM9          ; XOP
011C 280A                DATA >2800+FM3          ; XOR
                    *
                    * HEX, BINARY, OR DECIMAL INPUT
                    *
011E C04B           HEX  MOV  R11,R1             ; SAVE RETURN
0120 0208 0010           LI   R8,16              ; PRESET BASE
0124 1002                JMP  BIN10
0126 0208 0002      BIN  LI   R8,2               ; PRESET BASE
012A 069F           BIN10 BL  *R15
012C 1003                JMP  DEC5
012E C04B           DEC  MOV  R11,R1             ; SAVE RETURN
0130 0208 000A      DEC1 LI   R8,10              ; PRESET BASE
0134 04C7           DEC5 CLR  R7                 ; PRESET VALUE
0136 C184           DEC10 MOV R4,R6              ; PUT CHAR IN R6
0138 0226 FFD0           AI   R6,->30            ; REMOVE ASCII BIA
013C 110A                JLT  DEC30              ; NOT VALID
013E 0286 000A           CI   R6,10
0142 1105                JLT  DEC20              ; O.K.
0144 0226 FFF9           AI   R6,-7
0148 0286 000A           CI   R6,10
014C 1102                JLT  DEC30              ; NOT VALID
014E 8206           DEC20 C   R6,R8              ; IF NOT LT BASE - NOT GOOD
0150 1103                JLT  DEC40
```

```
0152 C2C1          DEC30  MOV  R1,R11          ; RESTORE EXIT
0154 C047                 MOV  R7,R1           ; R1=ANS.
0156 045B                 B    *R11            ; EXIT
0158 C006          DEC40  MOV  R6,R0
015A C187                 MOV  R7,R6
015C 3988                 MPY  R8,R6
015E A1C0                 A    R0,R7
0160 069F                 BL   *R15
0162 10E9                 JMP  DEC10
                   *
                   * GET REGISTER NAME
                   *
0164 C04B          GETR   MOV  R11,R1          ; SAVE RET
0166 069F                 BL   *R15
0168 C2C1                 MOV  R1,R11          ; TEMP. RESET OF R11
016A C30B          GETRA  MOV  R11,R12         ; SAVE RET
016C 0284 0052     GETR10 CI   R4,'R'          ; IF RX, SKIP THE R
0170 1601                 JNE  GETR20
0172 069F                 BL   *R15
0174 06A0 012E     GETR20 BL   @DEC            ; GET X
0178 0281 000F            CI   R1,15           ; TEST RANGE
017C 1B20                 JH   GETR30
017E 045C                 B    *R12            ; EXIT
                   *
                   * GET ADDRESS
                   *
0180 C04B          GETL   MOV  R11,R1          ; SAVE RET
0182 069F                 BL   *R15
0184 1001                 JMP  GETL10
0186 C04B          GETLA  MOV  R11,R1          ; SAVE RETURN
0188 0284 0025     GETL10 CI   R4,'%'          ; CHECK FOR BINARY
018C 13CC                 JEQ  BIN
018E 0284 0027            CI   R4,>27          ; CHECK FOR STRING (')
0192 1304                 JEQ  GETL20
0194 0284 003E            CI   R4,'>'          ; CHECK FOR HEX
0198 16CB                 JNE  DEC1            ; MUST BE DEFAULT
019A 10C2                 JMP  HEX+2           ; MUST BE HEX
019C 04C7          GETL20 CLR  R7              ; PRESET STRING
019E 069F          GETL30 BL   *R15            ; GET A CHAR
01A0 0284 0027            CI   R4,>27          ; IF ', DONE
01A4 1303                 JEQ  GETL40
01A6 0A87                 SLA  R7,8
01A8 E1C4                 SOC  R4,R7
01AA 10F9                 JMP  GETL30
01AC 069F          GETL40 BL   *R15            ; GET TERM.
01AE 10D1                 JMP  DEC30           ; EXIT
                   *
                   * TAB OVER SIX PLACES
                   *
01B0 0200 0006     TAB    LI   R0,6            ; R0=COUNTER
01B4 2CA0 0349     TAB10  OUT  @SPACE          ; Been changed to output >09 for ANN-ARBOR Termr
01B8 0600                 DEC  R0
01BA 16FC                 JNE  TAB10
01BC 045B                 B    *R11            ; EXIT
01BE 0204 522A     GETR30 LI   R4,'R*'         ; ISSUE RANGE ERROR
01C2 1071                 JMP  PT210           ; PLACED HERE FOR JMP NOT B
                   *
                   * GET ONE CHARACTER FROM USER BUFFER
```

```
                    * CHARACTER RETURNED RIGHT JUSTIFIED IN R4
                    *
01C4 C120 00B2  INPT    MOV  @PTR,R4          ; GET TEXT POINTER
01C8 05A0 00B2          INC  @PTR             ; SET FOR NEXT CHAR
01CC D114               MOVB *R4,R4           ; GET CHARACTER
01CE 0984               SRL  R4,8             ; RIGHT JUSTIFY
01D0 045B               B    *R11             ; EXIT
                    *
                    * ROUTINE:  BUFFIN
                    * BUFFER IN ONE LINE
                    * (ESC) WILL CANCEL THE LINE
                    * (CTRL-H) WILL BACKSPACE
                    * (RETURN) WILL END
                    *
01D2 0201 00B4  BUFFIN LI   R1,>B4            ; PRESET POINTER
01D6 C801 00B2          MOV  R1,@PTR
01DA 020C 0A0A  BUFF5   LI   R12,>0A0A        ; PRESET FOR B.S.
01DE 2C44       BUFF10  IN   R4               ; GET CHARACTER
01E0 DC44               MOVB R4,*R1+          ; SAVE IT
01E2 0984               SRL  R4,8
01E4 0284 001B          CI   R4,>1B           ; ESC?
01E8 1318               JEQ  PT120
01EA 0284 0008          CI   R4,>08           ; BACKSPACE?
01EE 1304               JEQ  BUFF20
01F0 0284 000D          CI   R4,>0D           ; CR?
01F4 16F2               JNE  BUFF5
01F6 045B               B    *R11             ; EXIT
01F8 0641       BUFF20  DECT R1               ; BACK UP ONE CHAR
01FA 2C8C               OUT  R12              ; LINE FEED, IF REQD.
01FC 04CC               CLR  R12              ; CANCEL REST
01FE 0281 00B4          CI   R1,>B4           ; NOT PAST FIRST ONE
0202 14ED               JHE  BUFF10
0204 2CA0 0349          OUT  @SPACE
0208 10E4               JMP  BUFFIN
                    *
                    * CONTROL LOOP - REQUEST ADDRESS,
                    * PRINT TRANSLATED OPCODES
                    *
020A 02E0 0080  IIABGN  LWPI >80              ; RESET WORKSPACE
020E 0201 0100          LI   R1,DFPC          ; SET DEFAULT PC
0212 020F 01C4          LI   R15,INPT         ; SET R15 FOR INPT CALL
0216 C801 00B0  PT110   MOV  R1,@PC           ; SAVE PC
021A C0A0 00B0  PT120   MOV  @PC,R2           ; R2=PC
021E 04C3               CLR  R3               ; R3=WORD COUNT
0220 C042       PT130   MOV  R2,R1            ; DISPLAY CURRENT ADDRESS
0222 06A0 0000          BL   @DMEMN           ; ADDRESS ON NEW LINE
0226 2CA0 0349          OUT  @SPACE           ; SPACE
022A C0C3       PT140   MOV  R3,R3            ; IF WORD COUNT NONZERO
022C 1307               JEQ  PT150            ;   DISPLAY INST. WORDS
022E C172               MOV  *R2+,R5          ; DISPLAY
0230 06A0 0000          BL   @TYPEWD
0234 C802 00B0          MOV  R2,@PC           ; UPDATE PC
0238 0643               DECT R3               ; REDUCE WORD COUNT
023A 10F2               JMP  PT130            ; CONT. TILL ALL DONE
023C 06A0 01B0  PT150   BL   @TAB             ; TAB OVER 6 PLACES
0240 06A0 01D2          BL   @BUFFIN          ; GET ONE LINE
                    *
                    * ACCEPT THE OP-CODE MNEMONIC
```

```
                        *
0244 020A 000F          LI   R10,OPS-1          ; R10=LOOKUP INDEX
0248 04C5               CLR  R5                 ; R5=CHAR. POS.
024A 04C6               CLR  R6                 ; R6=OPCODE COUNT
024C 069F       PT160   BL   *R15               ; GET ONE CHAR
024E C145               MOV  R5,R5              ; IF POS. ONE THEN
0250 160F               JNE  PT170              ;   CHECK FOR +/-/$
0252 0284 0024          CI   R4,'$'             ; CHECK FOR $(STRING)
0256 132C               JEQ  PT220
0258 0284 002B          CI   R4,'+'             ; CHECK FOR +(CONST.)
025C 1339               JEQ  PT250
025E 0284 002D          CI   R4,'-'             ; CHECK FOR -(CONST.)
0262 1339               JEQ  PT260
0264 0284 002F          CI   R4,'/'             ; CHECK FOR ADDR RESET
0268 1603               JNE  PT170
026A 06A0 011E          BL   @HEX               ; GET NEW ADDRESS
026E 10D3               JMP  PT110
0270 0284 0041  PT170   CI   R4,'A'             ; BE SURE WE HAVE A CHAR.
0274 1114               JLT  PT200
0276 0284 005A          CI   R4,'Z'
027A 1513               JGT  PAT90
027C 0AB4               SLA  R4,11              ; PUT CHAR IN LEFT 5 BITS
027E 058A       PT180   INC  R10                ; ADVANCE LOOKUP INDEX
0280 D01A       @       MOVB *R10,R0            ; GET CHAR. LEVEL
0282 130F               JEQ  PAT90              ; JUMP IF END OF TABLE
0284 1501               JGT  PT190              ; IF VALID END, UPDATE
0286 05C6               INCT R6                 ;    OPCODE COUNT
0288 0A10       PT190   SLA  R0,1               ; PUT POS. IN RIGHT BITS
028A 09E0               SRL  R0,14
028C 8005               C    R5,R0              ; COMPARE POS.
028E 11F7               JLT  PT180              ; LOWER POS.
0290 1508               JGT  PAT90              ; HIGHER - ERROR
0292 D01A               MOVB *R10,R0            ; SAME - CHECK CHAR.
0294 0A30               SLA  R0,3               ; CHAR IN LEFT 5 BITS
0296 9100               CB   R0,R4              ; COMPARE TO INPUT
0298 16F2               JNE  PT180              ; NO MATCH
029A 0585               INC  R5                 ; O.K. - UPDATE POS.
029C 10D7               JMP  PT160              ; GET REST OF OPCODE
029E D01A       PT200   MOVB *R10,R0            ; END - IS IT VALID?
02A0 1120               JLT  PT280              ; IF MINUS - O.K.
02A2 0204 532A  PAT90   LI   R4,'S*'            ; ERROR - SNATCH AWAY
02A6 2CA0 0032  PT210   OUT  @FEED              ; AVOID OVERPRINT
02AA 06A0 0000          BL   @TYPE              ; CONTROL AND START OVER
02AE 10B5               JMP  PT120              ; DON'T CHANGE PC
                        *
                        * HANDLE STRING ENTRIES.  COLLECT CHARACTERS
                        * UNTIL A CR.  THEN FORCE ADDRESS EVEN AND
                        * EXIT
                        *
02B0 C282       PT220   MOV  R2,R10
02B2 069F       PT225   BL   *R15               ; GET A CHAR.
02B4 0284 000D          CI   R4,>0D             ; IF CR - EXIT
02B7            CRET    EQU  $-1
02B8 1304               JEQ  PT230
02BA 0A84               SLA  R4,8               ; SAVE THE CHAR.
02BC DE84               MOVB R4,*R10+
02BE 0583               INC  R3
02C0 10F8               JMP  PT225
```

```
02C2 C003        PT230  MOV  R3,R0             ; IF ODD-INST. SPACE
02C4 0810               SRA  R0,1
02C6 1724               JNC  PT300
02C8 D6A0 0349          MOVB @SPACE,*R10       ; PAD WITH SPACE
02CC 0583               INC  R3
02CE 1020               JMP  PT300             ; GO PRINT RESULTS
                 *
                 * HANDLE CONSTANT ENTRIES.
                 * PT250 IS PLUS AND PT260 IS MINUS
                 *
02D0 06A0 0180   PT250  BL   @GETL             ; GETVALUE
02D4 1003               JMP  PT270             ; GO SAVE IT
02D6 06A0 0180   PT260  BL   @GETL             ; GET VALUE
02DA 0501               NEG  R1                ; -VALUE
02DC C481        PT270  MOV  R1,*R2            ; SAVE IT
02DE 05C3               INCT R3                ; SET R3
02E0 1017               JMP  PT300             ; GO PRINT
                 *
                 * THE OPCODE HAS BEEN LOCATED AND THE
                 * INDEX IS IN R6.  NOW COLLECT THE
                 * OPERANDS.
                 *
02E2 C2A6 0094   PT280  MOV  @CODE-2(R6),R10   ; R10=INST&PARSING INST.
02E6 C00A               MOV  R10,R0            ; PRESET THE INST.
02E8 0240 FFE0          ANDI R0,>FFE0
02EC C480               MOV  R0,*R2
02EE 05C3               INCT R3                ; COUNT=2
02F0 C04A               MOV  R10,R1            ; GET OP. ONE DESC.
02F2 0921               SRL  R1,2
02F4 0241 0006          ANDI R1,>6
02F8 C061 0088          MOV  @OP(R1),R1        ; R1=OPERAND INDEX
02FC 1301               JEQ  PT290             ; SKIP IF NO FIRST ONE
02FE 0691               BL   *R1               ; COLLECT FIRST ONE
0300 024A 0007   PT290  ANDI R10,7             ; COLLECT SECOND OP.
0304 0A1A               SLA  R10,1
0306 C1AA 0088          MOV  @OP(R10),R6
030A 1302               JEQ  PT300             ; JUMP IF NONE
030C 04CA               CLR  R10               ; SET FLAG
030E 0696               BL   *R6
                 *
                 * THE ENTIRE STATEMENT HAS BEEN
                 * ACCEPTED - PRINT TRANSLATION
                 * AND UPDATE P.C.
                 *
0310 2CA0 02B7   PT300  OUT  @CRET             ; RETURN
0314 06A0 01B0          BL   @TAB              ; TAB OVER SIX
0318 1088               JMP  PT140             ; GO DISPLAY OBJECT
                 *
                 * HANDLE S OR D
                 *      N
                 *      *N
                 *      *N+
                 *      @X(N)
                 *      @X
                 *
031A C38B        OPA    MOV  R11,R14           ; SAVE RETURN ADDRESS
031C 069F               BL   *R15              ; GET CHAR
031E 0284 002A          CI   R4,'*'            ; CHECK FOR *N OR *N+
```

```
0322 1324              JEQ  OPB              ; JUMP IF YES
0324 0284 0040         CI   R4,'@'           ; CHECK FOR @X OR @X(N)
0328 162D              JNE  OPC              ; JUMP IF NOT
032A 06A0 0180         BL   @GETL
032E C183              MOV  R3,R6            ; ADD TO MEMORY
0330 A182              A    R2,R6
0332 C581              MOV  R1,*R6           ; SAVE X
0334 05C3              INCT R3               ; UPDATE COUNT
0336 0201 0020         LI   R1,>20           ; ADDRESS MODE 2
033A 0284 000D         CI   R4,>0D           ; IF RETURN OR ',' DONE
033E 1311              JEQ  OPA10
0340 0284 002C         CI   R4,','
0344 130E              JEQ  OPA10
0346 0284 0020         CI   R4,' '           ; IF SPACE - DONE
0349             SPACE EQU  $-1
034A 130B              JEQ  OPA10
034C 0284 0028         CI   R4,'('           ; IF NOT ( - ERROR
0350 16A8              JNE  PAT90
0352 06A0 0164         BL   @GETR            ; GET REG. N
0356 0261 0020         ORI  R1,>20           ; SET MODE 2
035A 0284 0029         CI   R4,')'           ; IF NOT ) - ERROR
035E 16A1              JNE  PAT90
0360 069F              BL   *R15
0362 C00A        OPA10 MOV  R10,R0           ; REPOS. IT
0364 1601              JNE  OPA15
0366 0A61              SLA  R1,6
0368 E481        OPA15 SOC  R1,*R2           ; INSERT IT
036A 045E        OPA20 B    *R14             ; EXIT
036C 06A0 0164   OPB   BL   @GETR            ; GET N(FOR *N)
0370 0200 0010         LI   R0,>10           ; SET MODE = 1
0374 0284 002B         CI   R4,'+'           ; IF TERM. BY +
0378 1603              JNE  OPB10            ;   CHANGE MODE
037A 069F              BL   *R15
037C 0200 0030         LI   R0,>30           ; SET MODE = 3
0380 E040        OPB10 SOC  R0,R1            ; R1=REG&MODE
0382 10EF              JMP  OPA10
0384 06A0 016A   OPC   BL   @GETRA           ; GET N(FOR N)
0388 10EC              JMP  OPA10            ; MODE=0 - GO INSERT
                 *
                 * HANDLE SHIFT COUNT
                 *
038A C38B        OPD   MOV  R11,R14          ; SAVE RETURN
038C 06A0 0164         BL   @GETR            ; GET COUNT
0390 0A41              SLA  R1,4             ; REPOSITION
0392 E481              SOC  R1,*R2           ; INSERT
0394 10EA              JMP  OPA20            ; EXIT
                 *
                 * HANDLE IMMEDIATE OPERANDS
                 *
0396 C38B        OPE   MOV  R11,R14          ; SAVE RETURN
0398 06A0 0180         BL   @GETL            ; GET IOP
039C C183              MOV  R3,R6            ; ADD TO MEMORY
039E A182              A    R2,R6
03A0 C581              MOV  R1,*R6
03A2 05C3              INCT R3               ; ADJUST COUNT
03A4 10E2              JMP  OPA20            ; CONTINUE
                 *
                 * HANDLE W
```

```
                        *
03A6 C38B               OPF     MOV  R11,R14
03A8 06A0 0164                  BL   @GETR
03AC 10DA                       JMP  OPA10
                        *
                        * HANDLE DISPLACEMENTS
                        * + DIS
                        * - DIS
                        * ADDRESS  (CALCULATE DISPLACEMENT)
                        *
03AE C38B               OPG     MOV  R11,R14          ; SAVE RETURN
03B0 069F                       BL   *R15             ; GET FIRST CHAR
03B2 0284 002B                  CI   R4,'+'           ; CHECK FOR +DIS
03B6 1319                       JEQ  OPG30
03B8 0284 002D                  CI   R4,'-'           ; CHECK FOR -DIS
03BC 1319                       JEQ  OPG40
03BE 06A0 0186                  BL   @GETLA
03C2 C002                       MOV  R2,R0            ; MUST BE ADDRESS
03C4 05C0                       INCT R0               ; DIS*2=ADDRESS-(PC+2)
03C6 6040                       S    R0,R1
03C8 0811                       SRA  R1,1             ; DISP=BYTE STUFF/2
03CA 0281 007F  OPG10   CI      R1,>7F                ; CHECK RANGE
03CE 1509                       JGT  OPG20
03D0 0281 FF80                  CI   R1,>FF80
03D4 1106                       JLT  OPG20
03D6 0241 00FF  OPG15   ANDI    R1,>FF                ; RANGE O.K. SO
03DA E481                       SOC  R1,*R2           ; INSERT IT
03DC 0201 0002                  LI   R1,2             ; RESET R3
03E0 10C4                       JMP  OPA20            ; EXIT
03E2 0204 442A  OPG20   LI      R4,'D*'               ; RANGE ERROR
03E6 0460 02A6                  B    @PT210           ; GO ISSUE ERROR
03EA 06A0 0180  OPG30   BL      @GETL                 ; +DIS
03EE 10ED                       JMP  OPG10
03F0 06A0 0180  OPG40   BL      @GETL
03F4 0501                       NEG  R1               ; -DIS
03F6 10E9                       JMP  OPG10
                        *
                        * HANDLE BIT
                        *
03F8 C38B               OPH     MOV  R11,R14          ; SAVE RETURN
03FA 06A0 0180                  BL   @GETL
03FE 10EB                       JMP  OPG15            ; GO PROCESS IT
0400                            END  IIABGN
```

```
0126 BIN      012A BIN10    01DE BUFF10   01F8 BUFF20   01DA BUFF5
01D2 BUFFIN   0096 CODE     02B7 CRET     012E DEC      0130 DEC1
0136 DEC10    014E DEC20    0152 DEC30    0158 DEC40    0134 DEC5
0100 DFPC    *0004 DISK     0224 DMEMN    0032 FEED     0009 FM1
0005 FM2      000A FM3      000A FM4      0014 FM5      0008 FM6
0000 FM7      0013 FM8      000A FM9      0006 FMA      0003 FMB
0010 FMC      0180 GETL     0188 GETL10   019C GETL20   019E GETL30
01AC GETL40   0186 GETLA    0164 GETR    *016C GETR10   0174 GETR20
01BE GETR30   016A GETRA    011E HEX     *0000 IDTIIA  *0000 IDTMM
020A IIABGN   01C4 INPT     0088 OP       031A OPA      0362 OPA10
0368 OPA15    036A OPA20    036C OPB      0380 OPB10    0384 OPC
```

```
 038A OPD      0396 OPE      03A6 OPF      03AE OPG      03CA OPG10
 03D6 OPG15    03E2 OPG20    03EA OPG30    03F0 OPG40    03F8 OPH
  010 OPS      0000 P1       0080 P1E      0020 P2       00A0 P2E
 0040 P3       00C0 P3E      0060 P4       00E0 P4E      02A2 PAT90
 00B0 PC       0216 PT110    021A PT120    0220 PT130    022A PT140
 023C PT150    024C PT160    0270 PT170    027E PT180    0288 PT190
 029E PT200    02A6 PT210    02B0 PT220    02B2 PT225    02C2 PT230
 02D0 PT250    02D6 PT260    02DC PT270    02E2 PT280    0300 PT290
 0310 PT300    00B2 PTR      0000 R0       0001 R1       000A R10
 000B R11      000C R12     *000D R13      000E R14      000F R15
 0002 R2       0003 R3       0004 R4       0005 R5       0006 R6
 0007 R7       0008 R8      *0009 R9       0349 SPACE    01B0 TAB
 01B4 TAB10   *0008 TTIN     *000C TTOUT    02AC TYPE     0232 TYPEWD

EDIT/ASM/LOAD?
```