The Engineering Staff Of
TEXAS INSTRUMENTS INCORPORATED
Semiconductor Group

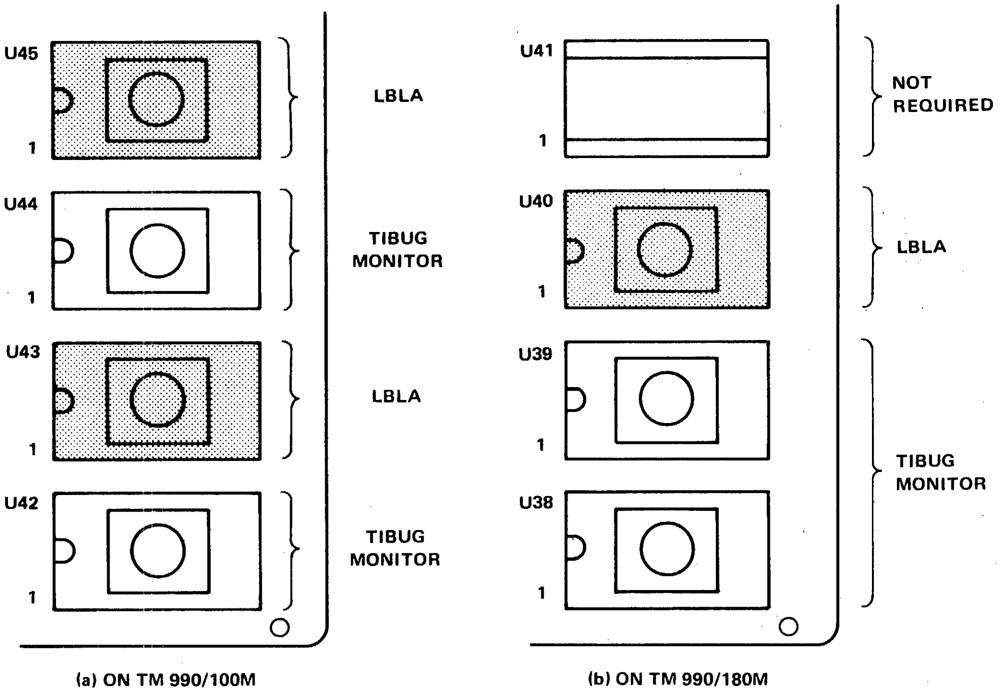
TM 990/402 LINE-BY-LINE ASSEMBLER USER'S GUIDE

EXIT LBLA
CRL|▲NL| ON VIDEO TERM

NOVEMBER 1977

TO GET INTO LBLA
FROM TIBUG, P=09E6

TEXAS INSTRUMENTS
INCORPORATED



(a) ON TM 990/100M

(b) ON TM 990/180M

FIGURE 1 – PLACEMENT OF TMS 2708 EPROM's

IMPORTANT NOTICE

Texas Instruments reserves the right to make changes at any time in order to improve design and to supply the best product possible.

TM 990/402
LINE-BY-LINE ASSEMBLER

1. GENERAL

The TM 990/402 Line-By-Line Assembler (LBLA) is a standalone program that assembles into object code the 69 instructions used by the TM 990/100M/180M microcomputers. Comments can be a part of the source statement; however, assembler directives are not recognized. Assembler TM 990/402-1 consists of two EPROM's and support the TM 990/100M microcomputer. TM 990/402-2 consists of one EPROM and supports the TM 990/180M microcomputer.

2. INSTALLATION

Remove the TMS 2708 chip(s) from the package and install as follows (see Figure 1):

(1) Turn off power to the TM 990/1XXM microcomputer.

(2) Place the chip(s) into the proper socket(s) as shown in Figure 1. The shaded components in Figure 1 denote the LBLA EPROM's correctly placed in their sockets. The corresponding socket number (UXX number) is marked on the EPROM.

NOTES

1. Place the TMS 2708(s) into the socket(s) with pin 1 in the lower left corner as denoted by a 1 on the board and on the EPROM. Be careful to prevent bending of the pins.

2. Do not remove EPROM's containing the monitor as shown in Figure 1. The monitor is used by the assembler.

(3) Verify proper positioning in the sockets. Apply power to the microcomputer board.

3. OPERATION

3.1 SETUP

NOTE

The examples in this guide use memory addresses obtainable in RAM on the TM 990/100M microcomputer. To exemplify the TM 990/180M addressing scheme, the reader should substitute a 3 for the F in the most significant digit (left most) of a four-digit memory address in the following examples (e.g., 3EE0$_{16}$ for FEE0$_{16}$).

- With the Line-By-Line Assembler EPROMs installed, call up the monitor by pressing the RESET switch in the upper left corner of the board and then pressing the A key at the terminal.

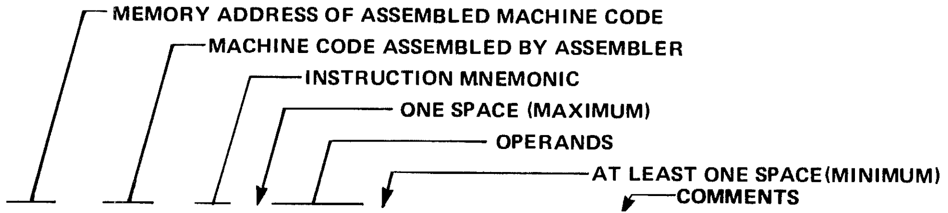
- Invoke the R keyboard command and set the Program Counter (PC) to 09E6$_{16}$. This is the memory address entry point for the Line-By-Line Assembler.

- Invoke the E (execute) command. The assembler will execute and print the memory address (M.A.) FE00$_{16}$ for the TM 990/100 or 3E00$_{16}$ for the TM 990/180M. The printhead will space to the assembly language opcode input column and wait for input from the keyboard.

```
?R
W=0BA4
P=000F   9E6  ◄────────── LBLA ENTRY ADDRESS
?E
FE00
```

3.2 INPUTS TO ASSEMBLER

The Line-By-Line Assembler accepts assembly language inputs from a terminal. As each instruction is input, the assembler interprets it, places the resulting machine code in an absolute address, and prints the machine code (in hexadecimal) next to its absolute address:


MEMORY ADDRESS OF ASSEMBLED MACHINE CODE
MACHINE CODE ASSEMBLED BY ASSEMBLER
INSTRUCTION MNEMONIC
ONE SPACE (MAXIMUM)
OPERANDS
AT LEAST ONE SPACE(MINIMUM)
COMMENTS

```
FE00 02E0 LWPI >FE80        SET UP WORKSPACE ADDRESS
FE02 FE80
FE04 0200 LI R0,10          SET UP COUNTER VALUE
FE06 000A
FE08 0201 LI R1,>FEA0       ADDRESS OF VALUES IN R1
FE0A FEA0
FE0C 0202 LI R2,>FEB0       ADDRESS OF STORAGE AREA IN R2
FE0E FEB0
FE10 CCB1 MOV *R1+,*R2+     MOVE VALUES TO STORAGE AREA
FE12 0600 DEC R0            DECREMENT COUNTER
FE14 1301 JEQ >FE18         EXIT IF COUNTER = ZERO
FE16 10FC JMP >FE10         LOOP BACK UNTIL 10 VALUES MOVED
FE18
```

Use only one space between the mnemonic and the operand. If you use the comment field, use at least one space between the operand and comment. If no comment is used, complete the instruction with a *space and carriage return.* If a comment is used, only a carriage return is required.

No loader tags are created; code is loaded in contiguous memory addresses by the assembler. The location can be changed as desired (explained in paragraph 3.2.2).

Labels cannot be used. Addressing is by byte displacement (jump instructions) or by absolute memory address.

NOTE

Be aware that the workspace for the TIBUG monitor begins in RAM at address $FFB0_{16}$ for the TM 990/100M and begins at address $3FB0_{16}$ for the TM 990/180M. Understand that assembled object code should not be entered at or above these addresses.

3.2.1 Program Preparation

Set up your program using flow charts with code written on a coding pad. Do not use assembler directives.

3.2.2 Changing Absolute Load Address

Code is located at the address written on the assembler output. When initialized, the assembler loads code contiguously starting at M.A. $FE00_{16}$ ($3E00_{16}$ for TM 990/180M). This address can be changed at any time during assembly by typing a slash (/) followed by the desired M.A.:

```
FE80 8081 C R1,R2                  COMPARE VALUES
FE82 1301 JEQ >FE86                IF EQUAL, SKIP ERROR ROUTINE
FE84 06A0 BL @>FF20                OTHERWISE DO ERROR ROUTINE
FE86 FF20
FE88      /FF20                    <———— CHANGE ADDRESS
FF20 2FA0 XOP @>FF26,14            SEND ERROR MESSAGE
FF22 FF26
FF24 045B B *R11                   RETURN TO CALLING PROGRAM
FF26 0A0D +>0A0D
FF28 4552 $ERROR FOUND
FF2A 524F
FF2C 5220
FF2E 464F
FF30 554E
FF32 4420
FF34 0000 +0000
FF36      /FE86                    <———— CHANGE ADDRESS
FE86
```

Note that this is similar to using an AORG (absolute origin) 990 assembler directive.

3.2.3 Entering Instructions

Any of the 69 instructions applicable to the TM 990/1XXM microcomputers can be interpreted by the Line-By-Line Assembler. The following apply:

(1) Place one space between instruction mnemonic and operand.

(2) Terminate entire instruction with a *space and a carriage return.* Lines with comments need only a carriage return. Character strings require two carriage returns.

(3) Do not use labels; addressing is through byte displacement (jump instructions) or absolute addresses:

```
FE8C 1607 JNE $+16
FE8E 10E8 JMP >FE60
FE90 C8A2 MOV @>FD20(R2),@>FE10(R2)
FE92 FD20
FE94 FE10
FE96
```

(4) Register numbers are in decimal and can be predefined (preceded by an R):

```
FE96 020C LI 12,>D00
FE98 0D00
FE9A 020D LI R13,>FFFF
FE9C FFFF
FE9E
```

(5) Jump instruction operand can be $+n, $-n, or >M where n is a decimal value of bytes (+256 ⩾ n ⩾ −254) and M is a memory address in hexadecimal. The dollar sign must be followed by a sign and number (JMP $ is not allowed).

```
FE20 1304 JEQ $+10          EXIT
FE22 1304 JEQ $+>A          EXIT
FE24 1304 JEQ $+%1010       EXIT
FE26 1304 JEQ >FE30         EXIT
FE28 10FF JMP $+0           LOOP AT THIS ADDRESS (>FE28)
FE2A 10FF JMP $-0           LOOP AT THIS ADDRESS
```

(6) Absolute numerical values can be in binary, decimal, or hexadecimal.

- Binary values are preceded by a percent sign (%). One to 16 ones and zeroes can follows; unspecified bits on the left will be zero filled:

```
FE58 0204 LI R4,%10101010   >AA IN R4
FE5A 00AA
FE5C 000A +%1010            DATA STATEMENT
FE5E FFF6 -%1010            DATA STATEMENT
FE60
```

- Decimal values have no prefix in an operand:

```
FE6C 0205 LI R5,100        LOAD COUNTER
FE6E 0064
FE70 0206 LI R6,32768      SET LIMIT
FE72 8000
FE74 8000 +32768
FE76 8000 -32768
FE78 7FFF +32767
FE7A 8001 -32767
FE7C FFFF -1
FE7E
```

- Hexadecimal values are preceded by the greater-than sign (>):

```
FE7E 02E0 LWPI >FF00       SET WP ADDRESS
FE80 FF00
FE82 FFFF +>FFFF           DATA STATEMENT
FE84 0001 ->FFFF           DATA STATEMENT
FE86
```

NOTE

In operands, absolute value must be unsigned values only. However, there is a method for using the assembler to compute and assemble a negative value; this method is especially useful with the immediate instructions (e.g., AI, CI, LI). Enter the instruction using the negative value. The assembled value will be all zeroes in the last assembled word. Use the slash command (paragraph 3.2.2) to assemble at the previous address, then enter the negative value as a data statement as shown in the following example:

```
FE1A  0201  LI  R1,->100        ← USE SIGNED OPERAND
FE1C  0000                      ← SIGNED NUMBER ASSEMBLIES AS 0000 (IN M.A.> FEIC)
FE1E        /FE1C               ← SET OBJECT LOAD ADDRESS TO PREVIOUS ADDRESS
FE1C  FF00  ->100               ← –>100 (>FF00) NOW IN M.A.>FEIC
FE1E
```

(7) Absolute addresses are used instead of labels:

```
FEA0 C820 MOV @>FE10,@>FED0        MOVE TO STORAGE
FEA2 FE10
FEA4 FED0
FEA6 16FC JNE >FEA0                LOOP BACK TO MOVE INSTRUCTION
FEA8
```

(8) Character strings are preceded by a dollar sign and are terminated with *two carriage returns.*

```
FF10 4142 $ABCD   1233
FF12 4344
FF14 2020
FF16 2031
FF18 3233
FF1A 3320                      ← UNUSED RIGHT BYTE FILLED WITH >20 (SPACE)
```

(9) Character strings of one or two characters can be designated by encoding the string in quotes. If not part of an operand, a plus or minus sign must precede the value. If the string is larger than two characters, the last two characters are interpreted.

```
FEAA 3132 +'12'        CHARACTERS  ONE AND TWO
FEAC 000C +12          VALUE OF POSITIVE TWELVE
FEAE FFF4 -12          VALUE OF NEGATIVE TWELVE
FEB0 0000 +            + FOLLOWED BY CTRL KEY AND NULL  KEY PRESSED
FEB2 0202 LI R2,'ABCD' ASSEMBLED LAST TWO CHARACTERS (C AND D)
FEB4 4344
FEB6 0202 LI R2,'E'    CHARACTER E IN RIGHT BYTE
FEB8 0045
FEBA 0202 LI R2,>E     VALUE >E IN RIGHT  BYTE
FEBC 000E
FEBE
```

(10) Signed numerical values of up to 16 bits can be designated by preceding the value with a plus or minus sign. If more than 16 bits are entered in binary or hexadecimal, the last 16 bits entered are used. If more than 16 bits are entered in decimal, the assembled value is the same as the remainder had the number been divided by 2^{15} ($65,536_{10}$).

```
FE18 00FF +%1111111100000000011111111
FE1A FF01 -%1111111100000000011111111
FE1C AAEE +>AAAAAAEE
FE1E 8000 +32768
FE20 8001 +32769
FE22 0000 +65536
FE24 FFFF +131071
FE26 0000 +131072
FE28 8000 -32768
FE2A 8001 -32767
FE2C 7FFF -32769
FE2E
```

3.3 ERRORS

When the assembler detects an error, it types an error symbol and readies the terminal for re-entering data at the same memory address. The following error symbols are used:

- D (Displacement error). The jump instruction destination is more than +256 or −254 bytes away.

```
FF38      JNC $+300*D
FF38      JNC >F000*D
FF38 170B JNC >FF50
FF3A
```

- R (Range error). The operand is out of range for its field:

```
FF30      LI R44,*R
FF30 0204 LI R4,200
FF32 00C8
```

- S (Syntax error). The instruction syntax was incorrect:

```
FF34      MOZ*S  } INCORRECT MNEMONICS
FF34      MOS*S  }
FF34 C802 MOV R2,@>FE90
FF36 FE90
```

4. EXITING TO THE MONITOR

Return control to monitor by pressing the escape (ESC) key,
SHIFT, CTRL & K TTY-/ KBD - CRL , NL

5. PSEUDO-INSTRUCTIONS

The TM 990/402 also interprets two pseudo-instructions. These pseudo-instructions are not additional instructions but actually are additional mnemonics that conveniently represent two members of the instruction set:

- The NOP mnemonic can be used in place of a JMP $+2 instruction which is essentially a no-op (no operation). This can be used to replace an existing instruction in memory, or it can be included in code to force additional execution time in a routine. Both NOP and JMP $+2 assemble to the machine code 1000_{16}.

- The RT mnemonic can be used in place of a B *R11 instruction which is a common return from a branch and (BL) subroutine. Both RT and B *R11 assemble to the machine code $045B_{16}$.

Note the following examples:

```
FE00 1000 JMP $+2        JUMP TO NEXT INSTRUCTION
FE02 1000 NOP            ALSO ASSEMBLES TO >1000
FE04 045B B *R11         RETURN COMMAND
FE06 045B RT             ALSO A RETURN COMMAND
```

TEXAS INSTRUMENTS
INCORPORATED
Semiconductor Group
Post Office Box 1444, Houston, Texas 77001

PRINTED IN U.S.A.

The Engineering Staff of
TEXAS INSTRUMENTS INCORPORATED
Semiconductor Group

TM 990/402-L LINE-BY-LINE ASSEMBLER LISTING

NOVEMBER 1977

TEXAS INSTRUMENTS
INCORPORATED

NOTES

TM 990/402 LINE-BY-LINE ASSEMBLER LISTING

1. GENERAL

This is an assembly language listing of the TM 990/402 Line-By-Line Assembler (LBLA) used with the TM 990/100M, TM 990/101M, and TM 990/180M microcomputers. This assembler listing is coded in the assembly language mnemonics used by Texas Instruments' 990 family. This language is further described in the following documents:

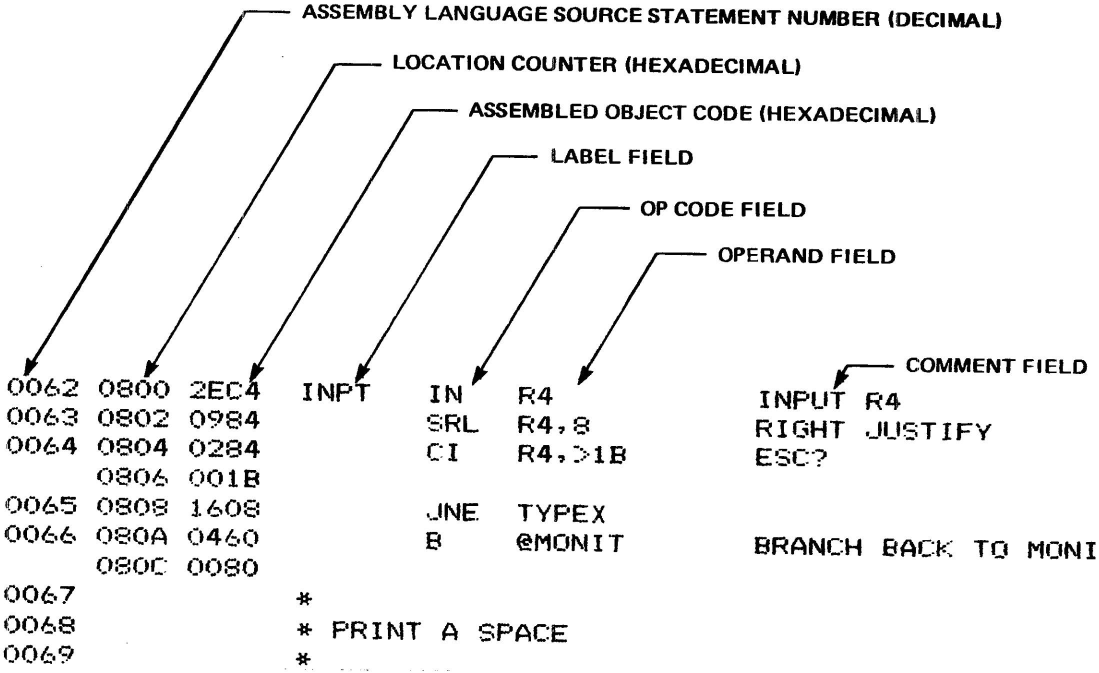
- *Model 990 Computer, TMS 9900 Microprocessor Assembly Language Programmer's Guide (P/N 943441-9701)*
- *TM 990/1XXM Microcomputer User's Guide (Section 4)*
- *TM 990/402 Line-By-Line Assembler User's Guide*

This listing was assembled on Texas Instruments 990 Software Development System Macro-assembler (SDSMAC).

Note that program data within the EPROM will include only hexadecimal object code at a corresponding location counter value as shown in Figure 1. This data begins at source statement number 0062 which shows the object code at absolute memory address (M.A.) 0800_{16} on the board. This statement is at the top of listing page 2.

2. LISTING FORMAT

Figure 1 identifies the different fields of the listing.


ASSEMBLY LANGUAGE SOURCE STATEMENT NUMBER (DECIMAL)
LOCATION COUNTER (HEXADECIMAL)
ASSEMBLED OBJECT CODE (HEXADECIMAL)
LABEL FIELD
OP CODE FIELD
OPERAND FIELD
COMMENT FIELD

```
0062 0800 2EC4  INPT   IN   R4           INPUT R4
0063 0802 0984         SRL  R4,8         RIGHT JUSTIFY
0064 0804 0284         CI   R4,>1B       ESC?
     0806 001B
0065 0808 1608         JNE  TYPEX
0066 080A 0460         B    @MONIT       BRANCH BACK TO MONITOR
     080C 0080
0067            *
0068            * PRINT A SPACE
0069            *
```

FIGURE 1. LISTING AND SOURCE STATEMENT FIELDS

2.1 **ASSEMBLY LANGUAGE SOURCE STATEMENT NUMBER.** This is the number, in decimal, of the statement in the Line-By-Line Assembler assembly language program. This shows the sequence in which the assembly language (source) statements were processed by the SDSMAC assembler.

2.2 **LOCATION COUNTER.** This is the hexadecimal number showing the location of assembled object code. This location is relative to the beginning of the program; thus it should begin with location 0000_{16}.One exception is where an absolute origin assembler directive (AORG) is used as in this program (source number 0057); the slash (/) directive in the Line-By-Line Assembler is equivalent to this directive.

Essentially, the location' counter number is the location in memory of the corresponding object code after a program has been loaded into memory with no load bias (bias of zero). In the Line-By-Line Assembler listing, this column shows the memory address in EPROM of the corresponding object code. For example, the object code at M.A. 0800_{16} is $2EC4_{16}$, at M.A. 0802_{16} it is 0984_{16}, etc.

2.3 **ASSEMBLED OBJECT CODE.** This column contains the resulting object code in hexadecimal after the source statement has been assembled.

2.4 **LABEL FIELD.** This six-character field contains an alphanumeric label that identifies the location of the source statement.

2.5 **OP CODE FIELD.** This four-character field contains assembly language operation code mnemonics. It is separated from the label field and operand field by at least one space.

2.6 **OPERAND FIELD.** This field contains the operands of the instruction. This field is separated from the op code and comment fields by at least one space.

2.7 **COMMENT FIELD.** The comments in this field are abbreviated auxiliary data to help further understand the instruction or the data flow.

3. INSTRUCTION SET AND INSTRUCTION FORMATS

The instruction set mnemonics, hexadecimal codes, formats, Status Register bits affected, and definitions are provided on pages 16 and 17 of this manual.

```
                  IDT  'LBLA'
           *
           * TITLE: ZERO LABEL ASSEMBLER
           *
           * REVISION: 9/19/77
           * COMPUTER: TM990/100M, TM990/180M MICROCOMPUTER
           * ABSTRACT: PROVIDES LIMITED ASSEMBLER CAPABILITY
           *           MOST FEATURES OF THE 990/4 ASSEMBLER
           *           ARE INCLUDED EXCEPT LABEL DEFINITION
           *           AND REFERENCE.
           *           THE LATEST UPDATE PUT ALLOWS COMMENTS
           *           TO BE MADE AFTER SOURCE CODE IS ENTERED.
           *           A SPACE CHARACTER IS STILL USED TO
           *           TERMINATE THE INSTRUCTION, HOWEVER A
           *           CARRIAGE RETURN MUST TERMINATE THE LINE.
           * CALLING SEQUENCE: BRANCH TO START ADDRESS
           *                   ZLABGN
           *
           * THE ENTRY ADDRESS IS AT ZLABGN=>09E6.
           *
           *
           * REGISTER EQUATES
           *
     0000  R0     EQU  0
     0001  R1     EQU  1
     0002  R2     EQU  2
     0003  R3     EQU  3
     0004  R4     EQU  4
     0005  R5     EQU  5
     0006  R6     EQU  6
     0007  R7     EQU  7
     0008  R8     EQU  8
     0009  R9     EQU  9
     000A  R10    EQU  10
     000B  R11    EQU  11
     000C  R12    EQU  12
     000D  R13    EQU  13
     000E  R14    EQU  14
     000F  R15    EQU  15
           *
     0080  MONIT  EQU  >0080        TOP OF TIBUG MONITOR, REV. A
           *
           * RAM AREA
           *
     FFFA  PC     EQU  >FFFA             PC ** >3FFA FOR TM990/180
     FFB0  WORKS  EQU  >FFB0             WP ** >3FB0 FOR TM990/180
     FE00  DFPC   EQU  >FE00             USER PC ** >3F00 FOR TM990/18
           *
           * MONITOR INTERFACE
           * ONLY XOP CALLS ARE VIA CALLS TO INPT,TYPE,
           * AND TYPEH
           *
                  DXOP OUT,12       OUTPUT CALL = 12
                  DXOP IN,11        INPUT CALL = 11
                  DXOP HEXC,10      HEX OUTPUT = 10
0800              AORG >0800        SET UP ORIGIN
           *
           * GET ONE CHARACTER FROM USER AND ECHO IT BACK
           * CHARACTER RETURNED RIGHT JUSTIFIED IN R4
           *
```

```
0062 0800 2EC4  INPT   IN    R4            INPUT R4
0063 0802 0984         SRL   R4,8          RIGHT JUSTIFY
0064 0804 0284         CI    R4,>1B        ESC?
     0806 001B
0065 0808 1608         JNE   TYPEX
0066 080A 0460         B     @MONIT        BRANCH BACK TO MONITOR
     080C 0080
0067                   *
0068                   * PRINT A SPACE
0069                   *
0070 080E 0204  TYPES  LI    R4,' '
     0810 0020
0071      0811  SPACE  EQU   $-1
0072                   *
0073                   * TYPE THE RIGHT BYTE OF R4.  AFTER THAT,
0074                   * TYPE THE LEFT BYTE IF IT IS NOT ZERO.
0075                   *
0076 0812 06C4  TYPE   SWPB R4             PUT IN RIGHT BYTE
0077 0814 2F04  TYPE1  OUT   R4            OUTPUT R4
0078 0816 0A84         SLA   R4,8          ANOTHER CHAR?
0079 0818 16FD         JNE   TYPE1         YES-TYPE IT
0080 081A 045B  TYPEX  B     *R11          RETURN
0081                   *
0082                   * TYPE THE FOUR DIGIT HEX NUMBER
0083                   * IN R5.
0084                   *
0085 081C 2E85  TYPEH  HEXC R5             HEX OUTPUT OF R5
0086 081E 045B         B     *R11          RETURN
0087                   *
0088                   * MNEMONIC TABLE.  THIS TABLE IS CONSTRUCTED
0089                   * AS A BINARY TREE. EACH ENTRY HAS THE
0090                   * CHARACTER POSITION AND THE CHARACTER.
0091                   * IF THE SIGN BIT IS SET THE CHARACTER IS A
0092                   * LEGAL END OF OP-CODE. THE ASCII CHARACTER
0093                   * IS IN THE RIGHTMOST FIVE BITS.
0094                   *
0095      0000  P1     EQU   0             CHAR ONE
0096      0020  P2     EQU   32            CHAR TWO
0097      0040  P3     EQU   64            CHAR THREE
0098      0060  P4     EQU   96            CHAR FOUR
0099      0080  P1E    EQU   >80+P1        CHAR ONE & END
0100      00A0  P2E    EQU   >80+P2        CHAR TWO & END
0101      00C0  P3E    EQU   >80+P3        CHAR THREE & END
0102      00E0  P4E    EQU   >80+P4        CHAR FOUR & END
0103 0820   81  OPS    BYTE P1E+'A'-'@'    A S,D
0104 0821   A2         BYTE P2E+'B'-'@'    AB S,D
0105 0822   D3         BYTE P3E+'S'-'@'    ABS S
0106 0823   A9         BYTE P2E+'I'-'@'    AI W,IOP
0107 0824   2E         BYTE P2+'N'-'@'
0108 0825   44         BYTE P3+'D'-'@'
0109 0826   E9         BYTE P4E+'I'-'@'    ANDI W,IOP
0110 0827   82         BYTE P1E+'B'-'@'    B S
0111 0828   AC         BYTE P2E+'L'-'@'    BL S
0112 0829   57         BYTE P3+'W'-'@'
0113 082A   F0         BYTE P4E+'P'-'@'    BLWP S
0114 082B   83         BYTE P1E+'C'-'@'    C S,D
0115 082C   A2         BYTE P2E+'B'-'@'    CB S,D
0116 082D   A9         BYTE P2E+'I'-'@'    CI W,IOP
0117 082E   2B         BYTE P2+'K'-'@'
0118 082F   4F         BYTE P3+'O'-'@'
```

```
 0119 0830   EE          BYTE P4E+'N'-'@'   CKON
 0120 0831   E6          BYTE P4E+'F'-'@'   CKOF
 0121 0832   2C          BYTE P2+'L'-'@'
 0122 0833   D2          BYTE P3E+'R'-'@'   CLR S
 0123 0834   2F          BYTE P2+'O'-'@'
 0124 0835   C3          BYTE P3E+'C'-'@'   COC S,W
 0125 0836   3A          BYTE P2+'Z'-'@'
 0126 0837   C3          BYTE P3E+'C'-'@'   CZC S,W
 0127 0838   04          BYTE P1+'D'-'@'
 0128 0839   25          BYTE P2+'E'-'@'
 0129 083A   C3          BYTE P3E+'C'-'@'   DEC S
 0130 083B   F4          BYTE P4E+'T'-'@'   DECT S
 0131 083C   29          BYTE P2+'I'-'@'
 0132 083D   D6          BYTE P3E+'V'-'@'   DIV S,W
 0133 083E   09          BYTE P1+'I'-'@'
 0134 083F   24          BYTE P2+'D'-'@'
 0135 0840   4C          BYTE P3+'L'-'@'
 0136 0841   E5          BYTE P4E+'E'-'@'   IDLE
 0137 0842   2E          BYTE P2+'N'-'@'
 0138 0843   C3          BYTE P3E+'C'-'@'   INC S
 0139 0844   F4          BYTE P4E+'T'-'@'   INCT S
 0140 0845   D6          BYTE P3E+'V'-'@'   INV S
 0141 0846   0A          BYTE P1+'J'-'@'
 0142 0847   25          BYTE P2+'E'-'@'
 0143 0848   D1          BYTE P3E+'Q'-'@'   JEQ DIS
 0144 0849   27          BYTE P2+'G'-'@'
 0145 084A   D4          BYTE P3E+'T'-'@'   JGT DIS
 0146 084B   A8          BYTE P2E+'H'-'@'   JH DIS
 0147 084C   C5          BYTE P3E+'E'-'@'   JHE DIS
 0148 084D   AC          BYTE P2E+'L'-'@'   JL DIS
 0149 084E   C5          BYTE P3E+'E'-'@'   JLE DIS
 0150 084F   D4          BYTE P3E+'T'-'@'   JLT DIS
 0151 0850   2D          BYTE P2+'M'-'@'
 0152 0851   D0          BYTE P3E+'P'-'@'   JMP DIS
 0153 0852   2E          BYTE P2+'N'-'@'
 0154 0853   C3          BYTE P3E+'C'-'@'   JNC DIS
 0155 0854   C5          BYTE P3E+'E'-'@'   JNE DIS
 0156 0855   CF          BYTE P3E+'O'-'@'   JNO DIS
 0157 0856   2F          BYTE P2+'O'-'@'
 0158 0857   C3          BYTE P3E+'C'-'@'   JOC DIS
 0159 0858   D0          BYTE P3E+'P'-'@'   JOP DIS
 0160 0859   0C          BYTE P1+'L'-'@'
 0161 085A   24          BYTE P2+'D'-'@'
 0162 085B   43          BYTE P3+'C'-'@'
 0163 085C   F2          BYTE P4E+'R'-'@'   LDCR S,C
 0164 085D   A9          BYTE P2E+'I'-'@'   LI W,IOP
 0165 085E   4D          BYTE P3+'M'-'@'
 0166 085F   E9          BYTE P4E+'I'-'@'   LIMI IOP
 0167 0860   32          BYTE P2+'R'-'@'
 0168 0861   45          BYTE P3+'E'-'@'
 0169 0862   F8          BYTE P4E+'X'-'@'   LREX
 0170 0863   37          BYTE P2+'W'-'@'
 0171 0864   50          BYTE P3+'P'-'@'
 0172 0865   E9          BYTE P4E+'I'-'@'   LWPI IOP
 0173 0866   0D          BYTE P1+'M'-'@'
 0174 0867   2F          BYTE P2+'O'-'@'
 0175 0868   D6          BYTE P3E+'V'-'@'   MOV S,D
 0176 0869   E2          BYTE P4E+'B'-'@'   MOVB S,D
 0177 086A   30          BYTE P2+'P'-'@'
 0178 086B   D9          BYTE P3E+'Y'-'@'   MPY S,W
```

```
0179 086C   0E          BYTE P1+'N'-'@'
0180 086D   25          BYTE P2+'E'-'@'
0181 086E   C7          BYTE P3E+'G'-'@'  NEG S
0182 086F   2F          BYTE P2+'O'-'@'
0183 0870   D0          BYTE P3E+'P'-'@'   NOP
0184 0871   0F          BYTE P1+'O'-'@'
0185 0872   32          BYTE P2+'R'-'@'
0186 0873   C9          BYTE P3E+'I'-'@'  ORI W,IOP
0187 0874   12          BYTE P1+'R'-'@'
0188 0875   33          BYTE P2+'S'-'@'
0189 0876   45          BYTE P3+'E'-'@'
0190 0877   F4          BYTE P4E+'T'-'@'  RSET
0191 0878   B4          BYTE P2E+'T'-'@'  RT
0192 0879   57          BYTE P3+'W'-'@'
0193 087A   F0          BYTE P4E+'P'-'@'  RTWP
0194 087B   93          BYTE P1E+'S'-'@'  S S,D
0195 087C   A2          BYTE P2E+'B'-'@'  SB S,D
0196 087D   CF          BYTE P3E+'O'-'@'  SBO BIT
0197 087E   DA          BYTE P3E+'Z'-'@'  SBZ BIT
0198 087F   25          BYTE P2+'E'-'@'
0199 0880   54          BYTE P3+'T'-'@'
0200 0881   EF          BYTE P4E+'O'-'@'  SETO S
0201 0882   2C          BYTE P2+'L'-'@'
0202 0883   C1          BYTE P3E+'A'-'@'  SLA W,N
0203 0884   2F          BYTE P2+'O'-'@'
0204 0885   C3          BYTE P3E+'C'-'@'  SOC S,D
0205 0886   E2          BYTE P4E+'B'-'@'  SOCB S,D
0206 0887   32          BYTE P2+'R'-'@'
0207 0888   C1          BYTE P3E+'A'-'@'  SRA W,N
0208 0889   C3          BYTE P3E+'C'-'@'  SRC W,N
0209 088A   CC          BYTE P3E+'L'-'@'  SRL W,N
0210 088B   34          BYTE P2+'T'-'@'
0211 088C   43          BYTE P3+'C'-'@'
0212 088D   F2          BYTE P4E+'R'-'@'  STCR S,C
0213 088E   53          BYTE P3+'S'-'@'
0214 088F   F4          BYTE P4E+'T'-'@'  STST W
0215 0890   57          BYTE P3+'W'-'@'
0216 0891   F0          BYTE P4E+'P'-'@'  STWP W
0217 0892   37          BYTE P2+'W'-'@'
0218 0893   50          BYTE P3+'P'-'@'
0219 0894   E2          BYTE P4E+'B'-'@'  SWPB S
0220 0895   3A          BYTE P2+'Z'-'@'
0221 0896   C3          BYTE P3E+'C'-'@'  SZC S,D
0222 0897   E2          BYTE P4E+'B'-'@'  SZCB S,D
0223 0898   14          BYTE P1+'T'-'@'
0224 0899   A2          BYTE P2E+'B'-'@'  TB BIT
0225 089A   98          BYTE P1E+'X'-'@'  X S
0226 089B   2F          BYTE P2+'O'-'@'
0227 089C   D0          BYTE P3E+'P'-'@'  XOP S,W
0228 089D   D2          BYTE P3E+'R'-'@'  XOR S,W
0229 089E   00          BYTE 0            END OF TABLE
0230                *
0231                * BRANCH TABLE FOR OPERANDS
0232                * 0 - N/A
0233                * 1 - S OR D
0234                * 2 - W OR C
0235                * 3 - IOP
0236                * 4 - N (SHIFT COUNT)
0237                * 5 - DIS
0238                * 6 - BIT
```

```
 0239                 *
 0240 08A0 0000       OP      DATA 0,OPA,OPF,OPE,OPD,OPG,OPH
      08A2 0B10
      08A4 0B9C
      08A6 0B8C
      08A8 0B80
      08AA 0BA4
      08AC 0BF8
 0241                 *
 0242                 * BASIC OP-CODE TABLE
 0243                 * EACH ENTRY HAS THE OP CODE, OPERAND
 0244                 * ONE AND OPERAND TWO DESCRIPTION.
 0245                 *
 0246      0009       FM1     EQU  >9               FORMAT 1 - S,D
 0247      0005       FM2     EQU  >5               FORMAT 2 - DIS
 0248      000A       FM3     EQU  >A               FORMAT 3 - S,W
 0249      000A       FM4     EQU  >A               FORMAT 4 - S,C
 0250      0014       FM5     EQU  >14              FORMAT 5 - W,N
 0251      0008       FM6     EQU  >8               FORMAT 6 - S
 0252      0000       FM7     EQU  0                FORMAT 7 - N/A
 0253      0013       FM8     EQU  >13              FORMAT 8 - W,IOP
 0254      000A       FM9     EQU  >A               FORMAT 9 - S,W
 0255      0006       FMA     EQU  >6               FORMAT A - BIT
 0256      0003       FMB     EQU  >3               FORMAT B - IOP
 0257      0010       FMC     EQU  >10              FORMAT C - W
 0258 08AE A009       CODE    DATA >A000+FM1        A
 0259 08B0 B009               DATA >B000+FM1        AB
 0260 08B2 0748               DATA >0740+FM6        ABS
 0261 08B4 0233               DATA >0220+FM8        AI
 0262 08B6 0253               DATA >0240+FM8        ANDI
 0263 08B8 0448               DATA >0440+FM6        B
 0264 08BA 0688               DATA >0680+FM6        BL
 0265 08BC 0408               DATA >0400+FM6        BLWP
 0266 08BE 8009               DATA >8000+FM1        C
 0267 08C0 9009               DATA >9000+FM1        CB
 0268 08C2 0293               DATA >0280+FM8        CI
 0269 08C4 03A0               DATA >03A0+FM7        CKON
 0270 08C6 03C0               DATA >03C0+FM7        CKOF
 0271 08C8 04C8               DATA >04C0+FM6        CLR
 0272 08CA 200A               DATA >2000+FM3        COC
 0273 08CC 240A               DATA >2400+FM3        CZC
 0274 08CE 0608               DATA >0600+FM6        DEC
 0275 08D0 0648               DATA >0640+FM6        DECT
 0276 08D2 3C0A               DATA >3C00+FM9        DIV
 0277 08D4 0340               DATA >0340+FM7        IDLE
 0278 08D6 0588               DATA >0580+FM6        INC
 0279 08D8 05C8               DATA >05C0+FM6        INCT
 0280 08DA 0548               DATA >0540+FM6        INV
 0281 08DC 1305               DATA >1300+FM2        JEQ
 0282 08DE 1505               DATA >1500+FM2        JGT
 0283 08E0 1B05               DATA >1B00+FM2        JH
 0284 08E2 1405               DATA >1400+FM2        JHE
 0285 08E4 1A05               DATA >1A00+FM2        JL
 0286 08E6 1205               DATA >1200+FM2        JLE
 0287 08E8 1105               DATA >1100+FM2        JLT
 0288 08EA 1005               DATA >1000+FM2        JMP
 0289 08EC 1705               DATA >1700+FM2        JNC
 0290 08EE 1605               DATA >1600+FM2        JNE
 0291 08F0 1905               DATA >1900+FM2        JNO
 0292 08F2 1805               DATA >1800+FM2        JOC
```

```
0293 08F4 1C05          DATA >1C00+FM2     JOP
0294 08F6 300A          DATA >3000+FM4     LDCR
0295 08F8 0213          DATA >0200+FM8     LI
0296 08FA 0303          DATA >0300+FMB     LIMI
0297 08FC 03E0          DATA >03E0+FM7     LREX
0298 08FE 02E3          DATA >02E0+FMB     LWPI
0299 0900 C009          DATA >C000+FM1     MOV
0300 0902 D009          DATA >D000+FM1     MOVB
0301 0904 380A          DATA >3800+FM9     MPY
0302 0906 0508          DATA >0500+FM6     NEG
0303 0908 1000          DATA >1000+FM7     NOP
0304 090A 0273          DATA >0260+FM8     ORI
0305 090C 0360          DATA >0360+FM7     RSET
0306 090E 045B          DATA >045B+FM7     RT
0307 0910 0380          DATA >0380+FM7     RTWP
0308 0912 6009          DATA >6000+FM1     S
0309 0914 7009          DATA >7000+FM1     SB
0310 0916 1D06          DATA >1D00+FMA     SBO
0311 0918 1E06          DATA >1E00+FMA     SBZ
0312 091A 0708          DATA >0700+FM6     SETO
0313 091C 0A14          DATA >0A00+FM5     SLA
0314 091E E009          DATA >E000+FM1     SOC
0315 0920 F009          DATA >F000+FM1     SOCB
0316 0922 0814          DATA >0800+FM5     SRA
0317 0924 0B14          DATA >0B00+FM5     SRC
0318 0926 0914          DATA >0900+FM5     SRL
0319 0928 340A          DATA >3400+FM4     STCR
0320 092A 02D0          DATA >02C0+FMC     STST
0321 092C 02B0          DATA >02A0+FMC     STWP
0322 092E 06C8          DATA >06C0+FM6     SWPB
0323 0930 4009          DATA >4000+FM1     SZC
0324 0932 5009          DATA >5000+FM1     SZCB
0325 0934 1F06          DATA >1F00+FMA     TB
0326 0936 0488          DATA >0480+FM6     X
0327 0938 2C0A          DATA >2C00+FM9     XOP
0328 093A 280A          DATA >2800+FM3     XOR
0329                *
0330                * HEX, BINARY, OR DECIMAL INPUT
0331                *
0332 093C C04B HEX   MOV  R11,R1        SAVE RETURN
0333 093E 0208      LI   R8,16         PRESET BASE
     0940 0010
0334 0942 1007      JMP  DEC5
0335 0944 0208 BIN  LI   R8,2          PRESET BASE
     0946 0002
0336 0948 069F      BL   *R15
0337 094A 1003      JMP  DEC5
0338 094C C04B DEC  MOV  R11,R1        SAVE RETURN
0339 094E 0208 DEC1 LI   R8,10         PRESET BASE
     0950 000A
0340 0952 04C7 DEC5 CLR  R7            PRESET VALUE
0341 0954 C184 DEC10 MOV R4,R6         PUT CHAR IN R6
0342 0956 0226      AI   R6,->30       REMOVE ASCII BIA
     0958 FFD0
0343 095A 110A      JLT  DEC30         NOT VALID
0344 095C 0286      CI   R6,10
     095E 000A
0345 0960 1105      JLT  DEC20         O.K.
0346 0962 0226      AI   R6,-7
     0964 FFF9
```

```
0347 0966 0286          CI   R6,10
     0968 000A
0348 096A 1102          JLT  DEC30          NOT VALID
0349 096C 8206   DEC20  C    R6,R8          IF NOT LT BASE - NOT GOOD
0350 096E 1103          JLT  DEC40
0351 0970 C2C1   DEC30  MOV  R1,R11         RESTORE EXIT
0352 0972 C047          MOV  R7,R1          R1=ANS.
0353 0974 045B          B    *R11           EXIT
0354 0976 C006   DEC40  MOV  R6,R0
0355 0978 C187          MOV  R7,R6
0356 097A 3988          MPY  R8,R6
0357 097C A1C0          A    R0,R7
0358 097E 069F          BL   *R15
0359 0980 10E9          JMP  DEC10
0360             *
0361             * GET REGISTER NAME
0362             *
0363 0982 C04B   GETR   MOV  R11,R1         SAVE RET
0364 0984 069F          BL   *R15
0365 0986 C2C1          MOV  R1,R11         TEMP. RESET OF R11
0366 0988 C34B   GETRA  MOV  R11,R13        SAVE RET
0367 098A 0284   GETR10 CI   R4,'R'         IF RX, SKIP THE R
     098C 0052
0368 098E 1601          JNE  GETR20
0369 0990 069F          BL   *R15
0370 0992 06A0   GETR20 BL   @DEC           GET X
     0994 094C
0371 0996 0281          CI   R1,15          TEST RANGE
     0998 000F
0372 099A 1B01          JH   GETR30
0373 099C 045D          B    *R13           EXIT
0374 099E 0204   GETR30 LI   R4,'R*'        ISSUE RANGE ERROR
     09A0 522A
0375 09A2 1075          JMP  PT210
0376             *
0377             * GET ADDRESS
0378             *
0379 09A4 C04B   GETL   MOV  R11,R1         SAVE RET
0380 09A6 069F          BL   *R15
0381 09A8 1001          JMP  GETL10
0382 09AA C04B   GETLA  MOV  R11,R1         SAVE RETURN
0383 09AC 0284   GETL10 CI   R4,'%'         CHECK FOR BINARY
     09AE 0025
0384 09B0 13C9          JEQ  BIN
0385 09B2 0284          CI   R4,>27         CHECK FOR STRING (')
     09B4 0027
0386 09B6 1305          JEQ  GETL20
0387 09B8 0284          CI   R4,'>'         CHECK FOR HEX
     09BA 003E
0388 09BC 16C8          JNE  DEC1           MUST BE DEFAULT
0389 09BE 069F          BL   *R15           MUST BE HEX
0390 09C0 10BE          JMP  HEX+2
0391 09C2 04C7   GETL20 CLR  R7             PRESET STRING
0392 09C4 069F   GETL30 BL   *R15           GET A CHAR
0393 09C6 0284          CI   R4,>27         IF ', DONE
     09C8 0027
0394 09CA 1303          JEQ  GETL40
0395 09CC 0A87          SLA  R7,8
0396 09CE E1C4          SOC  R4,R7
0397 09D0 10F9          JMP  GETL30
```

```
0398 09D2 069F  GETL40 BL    *R15          GET TERM.
0399 09D4 10CD         JMP   DEC30         EXIT
0400            *
0401            * TAB OVER FIVE PLACES
0402            *
0403 09D6 C20B  TAB    MOV   R11,R8        SAVE RETURN
0404 09D8 0200         LI    R0,5          R0=COUNTER
     09DA 0005
0405 09DC 06A0  TAB10  BL    @TYPES
     09DE 080E
0406 09E0 0600         DEC   R0
0407 09E2 16FC         JNE   TAB10
0408 09E4 0458         B     *R8           EXIT
0409            *
0410            * CONTROL LOOP - REQUEST ADDRESS,
0411            * PRINT TRANSLATED OPCODES
0412            *
0413 09E6 02E0  ZLABGN LWPI  WORKS         SET WORKSPACE
     09E8 FFB0
0414 09EA 0201         LI    R1,DFPC       SET DEFAULT PC
     09EC FE00
0415 09EE 020F         LI    R15,INPT      SET R15 FOR INPT CALL
     09F0 0800
0416 09F2 C801  PT110  MOV   R1,@PC        SAVE PC
     09F4 FFFA
0417 09F6 C0A0  PT120  MOV   @PC,R2        R2=PC
     09F8 FFFA
0418 09FA 04C3         CLR   R3            R3=WORD COUNT
0419 09FC C142  PT130  MOV   R2,R5         DISPLAY CURRENT ADDRESS
0420 09FE 0204         LI    R4,>0D0A      PRINT LINE FEED
     0A00 0D0A
0421 0A02 06A0         BL    @TYPE
     0A04 0812
0422 0A06 06A0         BL    @TYPEH        PRINT (R5) IN HEX
     0A08 081C
0423 0A0A 06A0         BL    @TYPES        SPACE OVER ONE
     0A0C 080E
0424 0A0E C0C3  PT140  MOV   R3,R3         IF.WORD COUNT NONZERO
0425 0A10 1307         JEQ   PT150         DISPLAY INST. WORDS
0426 0A12 C172         MOV   *R2+,R5       DISPLAY
0427 0A14 06A0         BL    @TYPEH
     0A16 081C
0428 0A18 C802         MOV   R2,@PC        UPDATE PC
     0A1A FFFA
0429 0A1C 0643         DECT  R3            REDUCE WORD COUNT
0430 0A1E 10EE         JMP   PT130         CONT. TILL ALL DONE
0431 0A20 06A0  PT150  BL    @TAB          TAB OVER 6 PLACES
     0A22 09D6
0432            *
0433            * ACCEPT THE OP-CODE MNEMONIC
0434            *
0435 0A24 020A         LI    R10,OPS-1     R10=LOOKUP INDEX
     0A26 081F
0436 0A28 04C5         CLR   R5            R5=CHAR. POS.
0437 0A2A 04C6         CLR   R6            R6=OPCODE COUNT
0438 0A2C 069F  PT160  BL    *R15          GET ONE CHAR
0439 0A2E 0284         CI    R4,' '        IF SPACE - END
     0A30 0020
0440 0A32 1329         JEQ   PT200
0441 0A34 C145         MOV   R5,R5         IF POS. ONE THEN
```

```
 0442 0A36 1610          JNE   PT170         CHECK FOR +/-/$
 0443 0A38 0284          CI    R4,'$'        CHECK FOR $(STRING)
      0A3A 0024
 0444 0A3C 132B          JEQ   PT220
 0445 0A3E 0284          CI    R4,'+'        CHECK FOR +(CONST.)
      0A40 002B
 0446 0A42 1339          JEQ   PT250
 0447 0A44 0284          CI    R4,'-'        CHECK FOR -(CONST.)
      0A46 002D
 0448 0A48 1339          JEQ   PT260
 0449 0A4A 0284          CI    R4,'/'        CHECK FOR ADDR RESET
      0A4C 002F
 0450 0A4E 1604          JNE   PT170
 0451 0A50 069F          BL    *R15          GET ANOTHER CHARACTER
 0452 0A52 06A0          BL    @HEX          GET NEW ADDRESS
      0A54 093C
 0453 0A56 10CD          JMP   PT110
 0454 0A58 0284   PT170  CI    R4,'A'        BE SURE WE HAVE A CHAR.
      0A5A 0041
 0455 0A5C 1116          JLT   PAT90
 0456 0A5E 0284          CI    R4,'Z'
      0A60 005A
 0457 0A62 1513          JGT   PAT90
 0458 0A64 0AB4          SLA   R4,11         PUT CHAR IN LEFT 5 BITS
 0459 0A66 058A   PT180  INC   R10           ADVANCE LOOKUP INDEX
 0460 0A68 D01A          MOVB  *R10,R0       GET CHAR. LEVEL
 0461 0A6A 130F          JEQ   PAT90         JUMP IF END OF TABLE
 0462 0A6C 1501          JGT   PT190         IF VALID END, UPDATE
 0463 0A6E 05C6          INCT  R6            OPCODE COUNT
 0464 0A70 0A10   PT190  SLA   R0,1          PUT POS. IN RIGHT BITS
 0465 0A72 09E0          SRL   R0,14
 0466 0A74 8005          C     R5,R0         COMPARE POS.
 0467 0A76 11F7          JLT   PT180         LOWER POS.
 0468 0A78 1508          JGT   PAT90         HIGHER - ERROR
 0469 0A7A D01A          MOVB  *R10,R0       SAME - CHECK CHAR.
 0470 0A7C 0A30          SLA   R0,3          CHAR IN LEFT 5 BITS
 0471 0A7E 9100          CB    R0,R4         COMPARE TO INPUT
 0472 0A80 16F2          JNE   PT180         NO MATCH
 0473 0A82 0585          INC   R5            O.K. - UPDATE POS.
 0474 0A84 10D3          JMP   PT160         GET REST OF OPCODE
 0475 0A86 D01A   PT200  MOVB  *R10,R0       END - IS IT VALID?
 0476 0A88 1120          JLT   PT280         IF MINUS - O.K.
 0477 0A8A 0204   PAT90  LI    R4,'S*'       ERROR - SNATCH AWAY
      0A8C 532A
 0478 0A8E 06A0   PT210  BL    @TYPE         CONTROL AND START OVER
      0A90 0812
 0479 0A92 10B1          JMP   PT120         DON'T CHANGE PC
 0480             *
 0481             * HANDLE STRING ENTRIES.  COLLECT CHARACTERS
 0482             * UNTIL A CR.  THEN FORCE ADDRESS EVEN AND
 0483             * EXIT
 0484             *
 0485 0A94 069F   PT220  BL    *R15          GET A CHAR.
 0486 0A96 0284          CI    R4,>0D        IF CR - EXIT
      0A98 000D
 0487 0A9A 1304          JEQ   PT230
 0488 0A9C 0A84          SLA   R4,8          SAVE THE CHAR.
 0489 0A9E DC84          MOVB  R4,*R2+
 0490 0AA0 0583          INC   R3
 0491 0AA2 10F8          JMP   PT220
 0492 0AA4 C003  PT230  MOV  R3,R0           IF ODD-INST. SPACE
 0493 0AA6 0810         SRA  R0,1
 0494 0AA8 1703         JNC  PT240
 0495 0AAA D4A0         MOVB @SPACE,*R2      PAD WITH SPACE
      0AAC 0811
 0496 0AAE 0583         INC  R3
 0497 0AB0 C0A0  PT240  MOV  @PC,R2          RESET PC
      0AB2 FFFA
 0498 0AB4 1024         JMP  PT300           GO PRINT RESULTS
 0499                *
 0500                * HANDLE CONSTANT ENTRIES.
 0501                * PT250 IS PLUS AND PT260 IS MINUS
 0502                *
 0503 0AB6 06A0  PT250  BL   @GETL           GETVALUE
      0AB8 09A4
 0504 0ABA 1003         JMP  PT270           GO SAVE IT
 0505 0ABC 06A0  PT260  BL   @GETL           GET VALUE
      0ABE 09A4
 0506 0AC0 0501         NEG  R1              -VALUE
 0507 0AC2 C481  PT270  MOV  R1,*R2          SAVE IT
 0508 0AC4 0203         LI   R3,2            SET R3
      0AC6 0002
 0509 0AC8 101A         JMP  PT300           GO PRINT
 0510                *
 0511                * THE OPCODE HAS BEEN LOCATED AND THE
 0512                * INDEX IS IN R6.  NOW COLLECT THE
 0513                * OPERANDS.
 0514                *
 0515 0ACA C2A6  PT280  MOV  @CODE-2(R6),R10   R10=INST&PARSING INST.
      0ACC 08AC
 0516 0ACE C00A         MOV  R10,R0          PRESET THE INST.
 0517 0AD0 0240         ANDI R0,>FFE0
      0AD2 FFE0
 0518 0AD4 C480         MOV  R0,*R2
 0519 0AD6 05C3         INCT R3              COUNT=2
 0520 0AD8 C04A         MOV  R10,R1          CHECK FOR 'RT'
 0521 0ADA 0281         CI   R1,>045B        AND HANDLE AS CONST.
      0ADC 045B
 0522 0ADE 13F1         JEQ  PT270
 0523 0AE0 C04A         MOV  R10,R1          GET OP. ONE DESC.
 0524 0AE2 0921         SRL  R1,2
 0525 0AE4 0241         ANDI R1,>6
      0AE6 0006
 0526 0AE8 C061         MOV  @OP(R1),R1      R1=OPERAND INDEX
      0AEA 08A0
 0527 0AEC 1301         JEQ  PT290           SKIP IF NO FIRST ONE
 0528 0AEE 0691         BL   *R1             COLLECT FIRST ONE
 0529 0AF0 0ADA  PT290  SLA  R10,13
 0530 0AF2 09CA         SRL  R10,12
 0531 0AF4 C1AA         MOV  @OP(R10),R6
      0AF6 08A0
 0532 0AF8 1302         JEQ  PT300           JUMP IF NONE
 0533 0AFA 04CA         CLR  R10             SET FLAG
 0534 0AFC 0696         BL   *R6
 0535                *
 0536                * THE ENTIRE STATEMENT HAS BEEN ACCEPTED
 0537                * PRINT ANY COMMENTS IF ENTERED, TERMINATE WITH
 0538                * A CARRIAGE RETURN, PRINT THE TRANSLATION AND
 0539                * UPDATE THE LOCATION COUNTER.
 0540                *
```

```
0541 0AFE 2EC4  PT300   IN    R4            GET A CHARACTER
0542 0B00 0984          SRL   R4,8          RIGHT JUSTIFY
0543 0B02 0284          CI    R4,>0D        CARRIAGE RETURN ?
     0B04 000D
0544 0B06 16FB          JNE   PT300         IF NO, GET ANOTHER CHAR
0545 0B08 06A0  PT310   BL    @TAB          TAB OVER SIX
     0B0A 09D6
0546 0B0C 0460          B     @PT140        GO DISPLAY OBJECT
     0B0E 0A0E
0547            *
0548            * HANDLE S OR D
0549            *       N
0550            *      *N
0551            *      *N+
0552            *      @X(N)
0553            *      @X
0554            *
0555 0B10 C38B  OPA     MOV   R11,R14       SAVE RETURN ADDRESS
0556 0B12 069F          BL    *R15          GET CHAR
0557 0B14 0284          CI    R4,'*'        CHECK FOR *N OR *N+
     0B16 002A
0558 0B18 1324          JEQ   OPB           JUMP IF YES
0559 0B1A 0284          CI    R4,'@'        CHECK FOR @X OR @X(N)
     0B1C 0040
0560 0B1E 162D          JNE   OPC           JUMP IF NOT
0561 0B20 06A0          BL    @GETL
     0B22 09A4
0562 0B24 C183          MOV   R3,R6         ADD TO MEMORY
0563 0B26 A182          A     R2,R6
0564 0B28 C581          MOV   R1,*R6        SAVE X
0565 0B2A 05C3          INCT  R3            UPDATE COUNT
0566 0B2C 0201          LI    R1,>20        ADDRESS MODE 2
     0B2E 0020
0567 0B30 0284          CI    R4,>0D        IF RETURN OR ',' DONE
     0B32 000D
0568 0B34 1311          JEQ   OPA10
0569 0B36 0284          CI    R4,','
     0B38 002C
0570 0B3A 130E          JEQ   OPA10
0571 0B3C 0284          CI    R4,' '        IF SPACE - DONE
     0B3E 0020
0572 0B40 130B          JEQ   OPA10
0573 0B42 0284          CI    R4,'('        IF NOT ( - ERROR
     0B44 0028
0574 0B46 16A1          JNE   PAT90
0575 0B48 06A0          BL    @GETR         GET REG. N
     0B4A 0982
0576 0B4C 0261          ORI   R1,>20        SET MODE 2
     0B4E 0020
0577 0B50 0284          CI    R4,')'        IF NOT ) - ERROR
     0B52 0029
0578 0B54 169A          JNE   PAT90
0579 0B56 069F          BL    *R15
0580 0B58 C00A  OPA10   MOV   R10,R0        REPOS. IT
0581 0B5A 1601          JNE   OPA15
0582 0B5C 0A61          SLA   R1,6
0583 0B5E E481  OPA15   SOC   R1,*R2        INSERT IT
0584 0B60 045E  OPA20   B     *R14          EXIT
0585 0B62 06A0  OPB     BL    @GETR         GET N(FOR *N)
     0B64 0982
```

```
 0586 0B66 0200          LI   R0,>10        SET MODE = 1
      0B68 0010
 0587 0B6A 0284          CI   R4,'+'        IF TERM. BY +
      0B6C 002B
 0588 0B6E 1603          JNE  OPB10         CHANGE MODE
 0589 0B70 069F          BL   *R15
 0590 0B72 0200          LI   R0,>30        SET MODE = 3
      0B74 0030
 0591 0B76 E040   OPB10  SOC  R0,R1         R1=REG&MODE
 0592 0B78 10EF          JMP  OPA10
 0593 0B7A 06A0   OPC    BL   @GETRA        GET N(FOR N)
      0B7C 0988
 0594 0B7E 10EC          JMP  OPA10         MODE=0 - GO INSERT
 0595             *
 0596             * HANDLE SHIFT COUNT
 0597             *
 0598 0B80 C38B   OPD    MOV  R11,R14       SAVE RETURN
 0599 0B82 06A0          BL   @GETR         GET COUNT
      0B84 0982
 0600 0B86 0A41          SLA  R1,4          REPOSITION
 0601 0B88 E481          SOC  R1,*R2        INSERT
 0602 0B8A 10EA          JMP  OPA20         EXIT
 0603             *
 0604             * HANDLE IMMEDIATE OPERANDS
 0605             *
 0606 0B8C C38B   OPE    MOV  R11,R14       SAVE RETURN
 0607 0B8E 06A0          BL   @GETL         GET IOP
      0B90 09A4
 0608 0B92 C183          MOV  R3,R6         ADD TO MEMORY
 0609 0B94 A182          A    R2,R6
 0610 0B96 C581          MOV  R1,*R6
 0611 0B98 05C3          INCT R3            ADJUST COUNT
 0612 0B9A 10E2          JMP  OPA20         CONTINUE
 0613             *
 0614             * HANDLE W
 0615             *
 0616 0B9C C38B   OPF    MOV  R11,R14
 0617 0B9E 06A0          BL   @GETR
      0BA0 0982
 0618 0BA2 10DA          JMP  OPA10
 0619             *
 0620             * HANDLE DISPLACEMENTS
 0621             * + DIS
 0622             * - DIS
 0623             * ADDRESS  (CALCULATE DISPLACEMENT)
 0624             *
 0625 0BA4 C38B   OPG    MOV  R11,R14       SAVE RETURN
 0626 0BA6 069F          BL   *R15          GET LEADER ($)
 0627 0BA8 0284          CI   R4,'$'
      0BAA 0024
 0628 0BAC 1607          JNE  OPG5
 0629 0BAE 069F          BL   *R15          GET FIRST CHAR
 0630 0BB0 0284          CI   R4,'+'        CHECK FOR +DIS
      0BB2 002B
 0631 0BB4 1319          JEQ  OPG30
 0632 0BB6 0284          CI   R4,'-'        CHECK FOR -DIS
      0BB8 002D
 0633 0BBA 131A          JEQ  OPG40
 0634 0BBC 06A0   OPG5   BL   @GETLA
      0BBE 09AA
```

```
0635 0BC0 C002          MOV  R2,R0          MUST BE ADDRESS
0636 0BC2 05C0          INCT R0             DIS*2=ADDRESS-(PC+2)
0637 0BC4 6040          S    R0,R1
0638 0BC6 0811   OPG10  SRA  R1,1           DISP=BYTE STUFF/2
0639 0BC8 0281          CI   R1,>7F         CHECK RANGE
     0BCA 007F
0640 0BCC 1509          JGT  OPG20
0641 0BCE 0281          CI   R1,>FF80
     0BD0 FF80
0642 0BD2 1106          JLT  OPG20
0643 0BD4 0241   OPG15  ANDI R1,>FF         RANGE O.K. SO
     0BD6 00FF
0644 0BD8 E481          SOC  R1,*R2         INSERT IT
0645 0BDA 0201          LI   R1,2           RESET R3
     0BDC 0002
0646 0BDE 10C0          JMP  OPA20          EXIT
0647 0BE0 0204   OPG20  LI   R4,'D*'        RANGE ERROR
     0BE2 442A
0648 0BE4 0460          B    @PT210         GO ISSUE ERROR
     0BE6 0A8E
0649 0BE8 06A0   OPG30  BL   @GETL          +DIS
     0BEA 09A4
0650 0BEC 0641   OPG35  DECT R1             ADJUST DIS FOR CUR. INST
0651 0BEE 10EB          JMP  OPG10
0652 0BF0 06A0   OPG40  BL   @GETL
     0BF2 09A4
0653 0BF4 0501          NEG  R1             -DIS
0654 0BF6 10FA          JMP  OPG35
0655             *
0656             * HANDLE BIT
0657             *
0658 0BF8 C38B   OPH    MOV  R11,R14        SAVE RETURN
0659 0BFA 06A0          BL   @GETL
     0BFC 09A4
0660 0BFE 10EA          JMP  OPG15          GO PROCESS IT
0661                    END
NO ERRORS
```

INSTRUCTION SET, ALPHABETICAL INDEX

ASSEMBLY LANGUAGE MNEMONIC	MACHINE LANGUAGE OP CODE	FORMAT*	STATUS REG. BITS AFFECTED	RESULT COMPARED TO ZERO	INSTRUCTION
A	A000	1	0-4	X	Add (word)
AB	B000	1	0-5	X	Add (byte)
ABS	0740	6	0-2	X	Absolute Value
AI	0220	8	0-4	X	Add Immediate
ANDI	0240	8	0-2	X	AND Immediate
B	0440	6	—		Branch
BL	0680	6	—		Branch and Link (R11)
BLWP	0400	6	—		Branch; New Workspace Pointer
C	8000	1	0-2		Compare (word)
CB	9000	1	0-2,5		Compare (byte)
CI	0280	8	0-2		Compare Immediate
CKOF	03C0	7	—		User Defined
CKON	03A0	7	—		User Defined
CLR	04C0	6	—		Clear Operand
COC	2000	3	2		Compare Ones Corresponding
CZC	2400	3	2		Compare Zeroes Corresponding
DEC	0600	6	0-4	X	Decrement (by one)
DECT	0640	6	0-4	X	Decrement (by two)
DIV	3C00	9	4		Divide
IDLE	0340	7	—		Computer Idle
INC	0580	6	0-4	X	Increment (by one)
INCT	05C0	6	0-4	X	Increment (by two)
INV	0540	6	0-2	X	Invert (One's Complement)
JEQ	1300	2	—		Jump Equal (ST2=1)
JGT	1500	2	—		Jump Greater Than (ST=1), Arithmetic
JH	1B00	2	—		Jump High (ST0=1 and ST2=0), Logical
JHE	1400	2	—		Jump High or Equal (ST0 or ST2=1), Logical
JL	1A00	2	—		Jump Low (ST0 and ST2=0), Logical
JLE	1200	2	—		Jump Low or Equal (ST0=0 or ST2=1), Logical
JLT	1100	2	—		Jump Less Than (ST1 and ST2=), Arithmetic
JMP	1000	2	—		Jump Unconditional
JNC	1700	2	—		Jump No Carry (ST3=0)
JNE	1600	2	—		Jump Not Equal (ST2=0)
JNO	1900	2	—		Jump No Overflow (ST4=0)
JOC	1800	2	—		Jump On Carry (ST3=1)
JOP	1C00	2	—		Jump Odd Parity (ST5=1)
LDCR	3000	4	0-2,5	X	Load CRU
LI	0200	8	—	X	Load Immediate
LIMI	0300	8	12-15		Load Interrupt Mask Immediate
LREX	03E0	7	12-15		Load and Execute
LWPI	02E0	8	—		Load Immediate to Workspace Pointer
MOV	C000	1	0-2	X	Move (word)
MOVB	D000	1	0-2,5	X	Move (byte)
MPY	3800	9	—		Multiply
NEG	0500	6	0-2	X	Negate (Two's Complement)
ORI	0260	8	0-2	X	OR Immediate
RSET	0360	7	12-15		Reset AU
RTWP	0380	7	0-15		Return from Context Switch
S	6000	1	0-4	X	Subtract (word)
SB	7000	1	0-5	X	Subtract (byte)
SBO	1D00	2	—		Set CRU Bit to One
SBZ	1E00	2	—		Set CRU Bit to Zero
SETO	0700	6	—		Set Ones
SLA	0A00	5	0-4	X	Shift Left Arithmetic
SOC	E000	1	0-2	X	Set Ones Corresponding (word)
SOCB	F000	1	0-2,5	X	Set Ones Corresponding (byte)
SRA	0800	5	0-3	X	Shift Right (sign extended)
SRC	0B00	5	0-3	X	Shift Right Circular
SRL	0900	5	0-3	X	Shift Right Logical
STCR	3400	4	0-2,5	X	Store From CRU
STST	02C0	8	—		Store Status Register
STWP	02A0	8	—		Store Workspace Pointer
SWPB	06C0		—	Swap Bytes	
SZC	4000	1	0-2	X	Set Zeroes Corresponding (word)
SZCB	5000	1	0-2,5	X	Set Zeroes Corresponding (byte)
TB	1F00	2	2		Test CRU Bit
X	0480	6	—		Execute
XOP	2C00	9	6		Extended Operation
XOR	2800	3	0-2	X	Exclusive OR

*Formats are defined on page 17.

INSTRUCTION FORMATS

FORMAT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	GENERAL USE
1	OP CODE			B	T_D		DR				T_S		SR				ARITHMETIC
2	OP CODE								SIGNED DISPLACEMENT								JUMP
3	OP CODE						WR				T_S		SR				LOGICAL
4	OP CODE						C				T_S		SR				CRU
5	OP CODE								C				R				SHIFT
6	OP CODE										T_S		SR				PROGRAM
7	OP CODE											NOT USED					CONTROL
8	OP CODE											N	R				IMMEDIATE
9	OP CODE						DR				T_S		SR				MPY, DIV, XOP

OP CODE	OPERATION CODE
B	BYTE INDICATOR (1=BYTE)
T_D	DESTINATION ADDRESS TYPE*
DR	DESTINATION REGISTER
T_S	SOURCE ADDRESS TYPE*
SR	SOURCE REGISTER
C	CRU TRANSFER COUNT OR SHIFT COUNT
R	REGISTER
N	NOT USED

*T_D OR T_S	ADDRESS MODE TYPE
00	DIRECT REGISTER
01	INDIRECT REGISTER
10	PROGRAM COUNTER RELATIVE, NOT INDEXED (SR OR DR = 0) PROGRAM COUNTER RELATIVE + INDEX REGISTER (SR OR DR>0)
11	INDIRECT REGISTER, AUTOINCREMENT REGISTER

TEXAS INSTRUMENTS
INCORPORATED
Semiconductor Group
Post Office Box 1443 Houston, Texas 77001

Printed in U.S.A.