



X2650

USER MANUAL FOR THE
2650 X8 SERIES CROSS-ASSEMBLER ON THE FDP8-E.

DECEMBER, 1976

SIERRA DIGITAL SYSTEMS
1440 WESTFIELD AVE.
RENO, NEVADA 89509
702-329-9548

ALTHOUGH THE INFORMATION IN THIS MANUAL HAS BEEN CHECKED FOR ACCURACY, NO RESPONSIBILITY IS ASSUMED FOR ERRORS. THIS DOCUMENTATION IS SUBJECT TO CHANGE WITHOUT NOTICE.

PDP AND OS/8 ARE REGISTERED TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.

TABLE OF CONTENTS: SECTION #

INTRODUCTION	1.0.0
OPERATION	2.0.0
LOADING THE CROSS-ASSEMBLER	2.1.0
CALLING SEQUENCE	2.2.0
INPUT/OUTPUT FILE EXTENSIONS	2.3.0
RUN-TIME OPTIONS	2.4.0
ASSEMBLER CHARACTER SET	3.0.0
STATEMENT FORMAT	4.0.0
CODING CONVENTIONS	4.1.0
LABELS	4.2.0
OPERATORS	4.3.0
OPERANDS	4.4.0
TERMS AND EXPRESSIONS	4.5.0
NUMERIC CONSTANTS	5.0.0
CONSTANTS WITH RADIX INDICATORS	5.1.0
CONSTANTS WITH ASCII INDICATORS	5.2.0
SYMBOLS	6.0.0
PERMANENT SYMBOLS	6.1.0
USER DEFINED SYMBOLS	6.2.0
LOCAL SYMBOLS	6.3.0
CURRENT LOCATION COUNTER	7.0.0
ARITHMETIC OPERATOR SET	8.0.0
UNARY OPERATORS	8.1.0
BYTE ACCESS OPERATORS (^L AND ^M)	8.1.2
THE COMPLEMENT OPERATOR (^C)	8.1.3
? OPERATOR	8.1.4
BINARY OPERATORS	8.2.0
PSEUDO-OPERATORS	9.0.0
ASSIGNMENT PSEUDO-OPS	9.1.0
.EQU	9.1.1
.SET	9.1.2
.DINST	9.1.3
.ORG	9.1.4
DEFAULT RADIX PSEUDO-OPS	9.2.0

TABLE OF CONTENTS: (CONT.)

SECTION #

DATA STORAGE PSEUDO-OPS.	9. 3. 0
. BYTE.	9. 3. 1
. DBYTE.	9. 3. 2
. ADDR.	9. 3. 3
. ZERO.	9. 3. 4
LISTING CONTROL DIRECTIVES.	9. 4. 0
. LIST.	9. 4. 1
. PAGE.	9. 4. 2
. TITLE.	9. 4. 3
CONDITIONAL ASSEMBLY PSEUDO-OPS.	9. 5. 0
. IFZERO.	9. 5. 1
. IFNZRO.	9. 5. 2
. IFDEF.	9. 5. 3
. IFNDEF.	9. 5. 4
. ENDC.	9. 5. 5
. END PSEUDO-OP.	9. 6. 0
ERROR MESSAGES.	10. 0. 0
MODIFICATION NOTES.	11. 0. 0
CROSS ASSEMBLER SPECIFICS.	12. 0. 0
CROSS-ASSEMBLER FILE NAMES.	12. 1. 0
LISTING FORMAT.	12. 2. 0
ADDITIONAL ERROR MESSAGES FOR THE 2650.	12. 3. 0
BINARY OUTPUT FILE.	12. 4. 0
MISCELLANEOUS NOTES.	12. 5. 0
2650 PAGE BOUNDRIES.	12. 5. 1
BXA/BSXA INDEX REGISTER.	12. 5. 2
REGISTER SPECIFICATION SYNTAX.	12. 5. 3
2650 ADDRESS SPACE.	12. 5. 4
ADDRESS DATA.	12. 5. 5
SAMPLE PROGRAM.	12. 6. 0
MICROPROCESSOR INSTRUCTION SET.	13. 0. 0
APPENDICES.	14. 0. 0
RUN-TIME OPTIONS.	APPENDIX A
INDICATOR SET.	APPENDIX B
PSUEDO-OPS.	APPENDIX C
ERROR MESSAGES.	APPENDIX D

1. 0. 0 INTRODUCTION.

THIS MANUAL DESCRIBES ONE OF THE X8 (CROSS EIGHT) SERIES OF MICRO-PROCESSOR CROSS-ASSEMBLERS SIERRA DIGITAL SYSTEMS HAS DEVELOPED FOR PDP8 USERS. THE X8 SERIES WILL HANDLE ALL OF THE POPULAR MICRO-PROCESSORS WITHIN A UNIVERSAL ASSEMBLER FORMAT. THIS COMMON BASE OF ASSEMBLER DIRECTIVES AND TECHNIQUES IS A SELECTED COMBINATION OF DESIRABLE FEATURES OBSERVED IN A SURVEY OF MANY EXISTING MINI-COMPUTER AND MICROPROCESSOR ASSEMBLERS. THE INSTRUCTION MNEMONICS AND ASSOCIATED SYNTAX OF EACH PARTICULAR MICROPROCESSOR ARE RETAINED UNCHANGED.

THIS MANUAL DESCRIBES THE USAGE OF ONE OF THE MICROPROCESSOR CROSS-ASSEMBLERS FROM THE SIERRA DIGITAL X8 SERIES. IN ORDER TO SIMPLIFY THE LEARNING PROCESS FOR INDIVIDUALS USING MORE THAN ONE CROSS-ASSEMBLER FROM THE SERIES, THIS MANUAL HAS BEEN DIVIDED INTO TWO MAJOR PARTS. SECTIONS 1 THROUGH 11 DOCUMENT THE UNIVERSAL ASSEMBLER FORMAT AS IT APPLIES TO ALL CROSS-ASSEMBLERS IN THE SERIES. THESE SECTIONS WILL BE IDENTICAL IN EVERY CROSS-ASSEMBLER MANUAL. SECTION 12 PRESENTS INFORMATION ON APPLICATION OF THE UNIVERSAL ASSEMBLER FORMAT TO THE SPECIFIC MICROPROCESSOR CROSS-ASSEMBLER. SECTION 13 PRESENTS A SUMMARY OF THE MNEMONIC INSTRUCTION CODES ASSIGNED BY THE MICROPROCESSOR VENDOR AND RECONIZED BY THE CROSS-ASSEMBLER. NO ATTEMPT HAS BEEN MADE TO DESCRIBE THE OPERATION OF THE MICROPROCESSOR ITSELF. SUCH INFORMATION MUST BE OBTAINED FROM THE MICROPROCESSOR VENDOR OR OTHER SOURCES. SECTION 14, THE APPENDICES, CONTAINS SUMMARY TABLES FOR QUICK REFERENCE ONCE THE USER GAINS EXPERTISE IN USING THE CROSS-ASSEMBLER.

WE AT SIERRA DIGITAL LOOK FORWARD TO DEVELOPING MORE ASSEMBLERS IN OUR X8 SERIES TO PROVIDE YOU, THE USER, WITH THE MEANS OF PIONEERING THE NEW WORLD OF MICROPROCESSORS.

2. 0. 0 OPERATION.

SIERRA DIGITAL'S CROSS-ASSEMBLER IS AN 8K, TWO PASS ASSEMBLER WHICH RUNS UNDER THE OS/8 OPERATING SYSTEM. THE CROSS-ASSEMBLER IS CODED IN PDP/8 ASSEMBLY LANGUAGE (PALS) TO GIVE FAST EXECUTION TIMES. (LESS THAN 30 SECONDS FOR A NORMAL 4K BYTE PROGRAM IS TYPICAL).

PASS 1 READS THE INPUT FILES AND SETS UP THE SYMBOL TABLES. PASS 2 THEN GENERATES THE OUTPUT FILE IN THE BINARY (OBJECT) FORMAT OF THE PARTICULAR MICROPROCESSOR. THE OUTPUT FILE CAN BE CHANGED TO BNPF FORMAT THROUGH USE OF THE /B RUN-TIME OPTION.

A THIRD ASSEMBLY PASS IS DONE WHEN A LISTING OUTPUT FILE IS SPECIFIED. WHEN NO BINARY FILE IS SPECIFIED, THE ASSEMBLER GOES DIRECTLY TO THE PASS 3 LISTING.

#2. 0. 0

THE CROSS-ASSEMBLER IS NOT RESTARTABLE. IF AN ATTEMPT IS MADE TO RESTART THE ASSEMBLER WITH A .ST COMMAND, THE KEYBOARD MONITOR RETURNS A "NO!!!" .

TYPING CTRL/C WILL HALT ASSEMBLY AND CAUSE AN IMMEDIATE EXIT TO THE KEYBOARD MONITOR.

TYPING CTRL/O AT THE KEYBOARD DURING ASSEMBLY WILL SUPPRESS THE LISTING OF ERROR MESSAGES TO THE CONSOLE DURING PASSES 1 AND 2. THE OUTPUT FILE WILL STILL SHOW THE ERROR MESSAGES IMMEDIATELY BEFORE THE LINE THAT IS IN ERROR.

2. 1. 0 LOADING AND SAVING THE CROSS-ASSEMBLER.

THE CROSS-ASSEMBLER IS PROVIDED IN BINARY FORMAT ON PAPER TAPE OR IN BOTH BINARY AND IMAGE FORMATS ON FILE-STRUCTURED MEDIA.

TO LOAD THE ASSEMBLER FROM PAPER TAPE AND SAVE IT, PLACE THE TAPE IN THE READER AND CALL THE ABSOLUTE LOADER:

```
. R ABSLDR
*PTR: $
```

```
. SAVE SYS: XNAME
```

FROM FILE STRUCTURED MEDIA, THE IMAGE FORMAT PROGRAM MAY BE COPIED DIRECTLY TO THE SYSTEM DEVICE OR THE BINARY FORMAT FILE MAY BE LOADED WITH THE ABSOLUTE LOADER. MODIFICATIONS TO THE IMAGE FILE, SUCH AS INVERTING THE SENSE OF A RUN-TIME OPTION, MAY BE IMPLEMENTED ACCORDING TO THE NOTES IN SECTION # 11. 0. 0 .

2. 2. 0 CALLING SEQUENCE.

ONCE LOADED AND SAVED, THE CROSS-ASSEMBLER IS CALLED FROM THE SYSTEM DEVICE BY TYPING:

```
. R XNAME
```

THE ASSEMBLER CALLS THE COMMAND DECODER WHICH RESPONDS WITH AN ASTERISK IN THE LEFT HAND MARGIN. THE USER MAY THEN TYPE IN THE INPUT AND OUTPUT FILE SPECIFICATIONS AND RUN-TIME OPTIONS:

```
*DEV: BIN, DEV: LIST<DEV: IN1, . . . DEV: IN9/OPT
```

THE FIRST OUTPUT FILE IS THE MICROPROCESSOR BINARY OBJECT FILE WRITTEN IN THE FORMAT SPECIFIED BY THE VENDOR OF THE PARTICULAR MICROPROCESSOR. (SEE SECTION 12. 0. 0 FOR THE FORMAT SPECIFICATIONS).

2.2.0

THE SECOND OUTPUT FILE IS THE OPTIONAL LISTING. WHEN ONLY THE FIRST OUTPUT FILE IS SPECIFIED, THE ASSEMBLER ASSUMES THAT IT WILL BE THE BINARY OUTPUT FILE AND THE LISTING IS OMITTED.

THE FOLLOWING EXAMPLE SPECIFIES FILE "IN1" TO BE READ FROM DECTAPE 0 AND THE BINARY (OBJECT) FILE TO BE OUTPUT TO THE PAPER TAPE PUNCH WITH NO LISTING:

```
. R XNAME
*PTP: <DTAO: IN1
```

THIS EXAMPLE SPECIFIES 2 FILES AS THE SOURCE INPUT (FROM THE DSK: DEVICE) WITH ONLY THE PASS 3 LISTING BEING OUTPUT TO THE LINE PRINTER:

```
. R XNAME
*, LPT: <IN1, IN2
```

UP TO NINE INPUT FILES CAN BE SPECIFIED AS ONE PROGRAM WHERE THE LAST FILE IS TERMINATED WITH AN .END STATEMENT.

2.3.0 INPUT/OUTPUT FILE EXTENSIONS.

IF THE EXTENSION TO AN INPUT FILE NAME IS OMITTED, THE ASSEMBLER ASSUMES THE .MS EXTENSION. IF THERE IS NO FILE WITH THAT NAME AND AN .MS EXTENSION, THE ASSEMBLER ASSUMES THE NULL EXTENSION. UNLESS EXTENSIONS ARE SPECIFIED, THE .MB AND .LS EXTENSIONS ARE ADDED TO THE OUTPUT BINARY AND LISTING FILES.

```
. MB - MICROPROCESSOR BINARY OUTPUT FILE EXTENSION.
. LS - OUTPUT LISTING FILE EXTENSION.
. MS - MICROPROCESSOR SOURCE FILE EXTENSION.
```

2.4.0 RUN-TIME OPTIONS.

TABLE #1 DESCRIBES THE OPTIONS WHICH MAY BE SPECIFIED AT RUN-TIME IN THE INPUT LINE TO THE COMMAND DECODER.

IF ONE OR MORE OF THESE OPTIONS IS CONTINUALLY CALLED, THE USER SHOULD CONSIDER MODIFYING THE ASSEMBLER TO INVERT THE SENSE OF THE OPTION. THE MODIFICATION NOTES IN SECTION #11.0.0 EXPLAIN HOW THIS MAY BE DONE. FOR EXAMPLE, A USER WHO PREFERS TO OUTPUT FILES IN BNPF FORMAT RATHER THAN BINARY CAN INVERT THE SENSE OF THE /B OPTION. THEN THE BINARY FILES ARE NORMALLY WRITTEN IN BNPF FORMAT. USE OF THE /B OPTION THEN CAUSES THE OUTPUT FILE TO BE WRITTEN IN THE STANDARD MICROPROCESSOR BINARY CODE. SPACE IS PROVIDED IN TABLE #1 TO CHECK OFF WHICH OPTIONS HAVE BEEN INVERTED FOR YOUR REFERENCE.

TABLE #1. RUN-TIME OPTIONS. (CONT.) #2. 4. 0

```

*****
OPTION                                MEANING                                INVERT?
*****

/H      INHIBIT HEADINGS AND PAGINATION.                                -----
        NORMALLY, THE ASSEMBLER AUTOMATICALLY PAGES THE
        OUTPUT, ADDING A HEADER TO THE TOP OF THE PAGE. USE
        OF THE /H OPTION WILL ELIMINATE THE HEADING AND THE
        PAGINATION.

/J      LIST UNASSEMBLED STATEMENTS AND CONDITIONAL                    -----
        ASSEMBLY PSEUDO-OPS.
        STATEMENTS WHICH DO NOT GET ASSEMBLED DUE TO
        CONDITIONAL ASSEMBLY PSEUDO-OPS ARE NORMALLY NOT
        LISTED. NEITHER ARE THE CONDITIONAL PSEUDO-OPS
        THEMSELVES. USE OF THE /J OPTION WILL ADD THESE
        STATEMENTS TO THE LISTING.

/K      EXPAND SYMBOL TABLE STORAGE INTO EXTRA CORE.                  -----
        NORMALLY MOST OF FIELD 1 IS USED FOR BOTH LOCAL AND
        NORMAL USER SYMBOL STORAGE. USE OF THE /K OPTIONS
        EXPANDS CORE USAGE TO 12K WHERE THE LOCAL SYMBOL
        TABLE RESIDES IN FIELD 2 AND THE REGULAR SYMBOL
        TABLE RESIDES IN FIELD 1.

/L      OUTPUT LEADER IN BINARY FILE FOR .ORG STATEMENTS              -----
        THIS OPTION MAY BE USED TO PHYSICALLY SEPARATE
        DISCONTINUOUS SECTIONS OF THE BINARY OUTPUT ON A
        PAPER TAPE.

/O      OUTPUT LISTING WITH BINARY CODE IN OCTAL FORMAT.              -----
        THE GENERATED BINARY CODE IS NORMALLY PRINTED IN
        HEXADECIMAL AT THE LEFT OF THE PROGRAM STATEMENTS
        IN THE LISTING FILE. THE /O OPTION WILL CAUSE THE
        BINARY CODE TO BE LISTED IN OCTAL INSTEAD OF
        HEXADECIMAL.

/N      LIST ONLY THE SYMBOL TABLE.                                    -----
        THE THIRD PASS LISTING NORMALLY CONSISTS OF THE
        STATEMENT LISTING PLUS THE USER SYMBOL TABLE
        LISTING. THE /N OPTION CAUSES ONLY THE SYMBOL TABLE
        TO BE LISTED.

/P      INCLUDE NORMALLY UNLISTED PSEUDO-OPS IN THE LISTING          -----
        SOME PSEUDO-OPS WILL NOT BE LISTED BY PASS 3 UNLESS
        THE /P OPTION IS USED.

/S      OMIT THE SYMBOL TABLE FROM LISTING.                            -----
        ONLY THE PROGRAM STATEMENTS ARE LISTED WITH THIS
        OPTION.

```

TABLE #1. RUN-TIME OPTIONS. (CONT.) #2. 4. 0

```

*****
OPTION                               MEANING                               INVERT?
*****

/T      REPLACE FORM/FEED WITH 3 CR/LF'S.           -----
        WHEN LISTING TO A DEVICE SUCH AS A TTY WHICH DOES
        NOT HAVE A FORM/FEED CONTROL, USE OF THE /T OPTION
        WILL REPLACE THE FORM/FEED WITH 3 BLANK LINES .

/W      INHIBIT WARNING MESSAGES.                 -----
        WHEN WARNING MESSAGES CAN BE SAFELY IGNORED, THIS
        OPTION WILL PREVENT THEM FROM BEING OUTPUT.

/O      USER FLAGS, USED WITH THE ? OPERATOR, SEE SECTION
TO /9   # 8. 1. 4 .

```

3. 0. 0 ASSEMBLER CHARACTER SET.

THE FOLLOWING CHARACTERS ARE LEGAL SOURCE CODE CHARACTERS:

- 1) ALPHABETICS A-Z, UPPER CASE ASCII
- 2) NUMERICS 0-9
- 3) THE SPECIAL CHARACTERS LISTED BELOW.

```

*      MULTIPLICATION
/      DIVISION
&      BOOLEAN AND
!      INCLUSIVE OR
+      ADDITION
-      SUBTRACTION
[ ]    PRECEDENCE INDICATORS
^      UNIVERSAL UNARY OPERATOR (UPARROW). USED WITH:
        ^C - COMPLEMENT (UPARROW C)
        ^B - BINARY RADIX INDICATOR (UPARROW B)
        ^D - DECIMAL RADIX INDICATOR (UPARROW D)
        ^H - HEXADECIMAL RADIX INDICATOR (UPARROW H)
        ^O - OCTAL RADIX INDICATOR (UPARROW O)
        ^L - LEAST SIGNIFICANT BYTE ACCESS OPERATOR
        ^M - MOST SIGNIFICANT BYTE ACCESS OPERATOR
;      COMMENT INDICATOR
" OR ' ASCII INDICATOR
?      USER FLAG OPERATOR
.      CURRENT LOCATION COUNTER (PERIOD)

```

3. 0. 0

THE CARRIAGE RETURN CHARACTER IS RECOGNIZED AS THE TERMINATOR FOR EACH SOURCE LINE. THE LINE-FEED, RUBOUT, FORM-FEED, AND NULL CHARACTERS ARE IGNORED BY THE ASSEMBLER. FORM-FEED CHARACTERS OCCURRING IN THE SOURCE HAVE NO AFFECT ON THE LISTING. ALL ASCII CHARACTERS MAY BE USED IN THE COMMENT FIELD OF A STATEMENT.

4. 0. 0 STATEMENT FORMAT.

STATEMENTS ARE WRITTEN IN THE GENERAL FORM:

LABEL OPERATOR OPERAND ; COMMENT

LABELS MUST START IN COLUMN 1. THEY MAY BE DIRECTLY FOLLOWED WITH AN OPTIONAL COMMA IF DESIRED. THE MODIFICATION NOTES EXPLAIN HOW TO REPLACE THE COMMA WITH ANOTHER DELIMITER SUCH AS A COLON.

OPERATORS MUST BE SEPARATED FROM THE LABEL WITH AT LEAST ONE SPACE OR TAB. WHEN NO LABEL IS PRESENT, THE OPERATOR MAY BEGIN IN ANY COLUMN BEYOND COLUMN 1.

THE OPERAND (IF ANY) MUST BE SEPARATED FROM THE OPERATOR WITH AT LEAST ONE SPACE OR TAB.

THE COMMENT (IF ANY) MUST BE SEPARATED FROM THE OPERAND (OR OPERATOR IF THERE IS NO OPERAND BY A SEMICOLON (;).

AN INPUT LINE MAY BE UP TO 127 CHARACTERS LONG (NOT INCLUDING THE CARRIAGE RETURN). WHEN THE INPUT LINES ARE OUTPUT TO THE LISTING FILE, ANY CHARACTERS AFTER THE 72D COLUMN ARE WRITTEN ON THE NEXT LINE(S) BEGINNING AT THE 25TH COLUMN OF THE FIRST SOURCE LINE (NORMAL COMMENT COLUMN). SEE THE MODIFICATION NOTES IN SECTION #11. 0. 0 TO ADJUST FOR NARROWER OR WIDER PAGE OUTPUT. THE CARRIAGE RETURN IS A TERMINATOR FOR BOTH THE STATEMENT AND THE LINE. ONLY ONE STATEMENT IS ALLOWED PER 127 CHARACTER LINE.

4. 1. 0 CODING CONVENTIONS:

ALTHOUGH THE ASSEMBLER WILL ACCEPT PROGRAMS WRITTEN IN FREE FORMAT, THE USE OF TABS MAKES FOR MORE READABLE CODE. TAB STOPS ARE SET EVERY 8 CHARACTERS IN THE LINE SO THAT THE USE OF THE TAB KEY SIMPLIFIES INPUT. GENERALLY:

LABELS	OCCUPY THE FIRST TAB FIELD, COLUMNS	1 THROUGH 8
OPERATORS	OCCUPY THE SECOND TAB FIELD, COLUMNS	9 THROUGH 16.
OPERANDS	OCCUPY THE THIRD TAB FIELD, COLUMNS	17 THROUGH 24.
COMMENTS	OCCUPY THE REMAINING FIELDS, COLUMNS	25 THROUGH 127.

4. 2. 0 LABELS.

A LABEL IS A SYMBOL WHICH PRECEDES THE OPERATOR AND MUST FOLLOW THE SYMBOL NAMING CONVENTIONS DESCRIBED IN SECTION # 6. 2. 0 . IN ALL BUT THE SYMBOL DEFINITION PSEUDO-OPS, (. EQU, . SET, . DINST) THE LABEL IS A LOCATION TAG AND IS EQUAL TO THE VALUE OF THE CURRENT LOCATION COUNTER.

EXAMPLE:

```

      2 1          . ORG      201
      0 6 LABEL1   . EQU      6      ; LABEL1=6
201  1 LABEL2   . BYTE     1      ; LABEL2=LOCATION TAG=201

```

NOTE THAT A JUMP TO LABEL1 WILL TRANSFER TO ADDRESS 6 WHILE A JUMP TO LABEL2 GOES TO ADDRESS 201.

A LABEL LACKING BOTH AN OPERATOR AND OPERAND IS SET EQUAL TO THE VALUE OF THE NEXT ADDRESS TO BE ASSEMBLED. IF USED AT THE BEGINNING OF THE PROGRAM, IT IS SET EQUAL TO THE VALUE OF THE FIRST ADDRESS. WHEN A SOLITARY LABEL IS FOLLOWED BY AN . ORG STATEMENT, IT RETAINS THE ORIGINAL VALUE ASSIGNED BEFORE THE ORIGIN CHANGE.

4. 3. 0 OPERATORS.

AN OPERATOR IS A MNEMONIC WHICH INDICATES THE ACTION TO BE PERFORMED AND IS EITHER A PSEUDO-OP OR ONE OF THE MICROPROCESSOR INSTRUCTIONS. PSEUDO-OPS ARE DESCRIBED IN SECTION #9. 0. 0. THE MICROPROCESSOR INSTRUCTION SET IS DESCRIBED IN SECTION #13. 0. 0 . THESE OPERATORS SHOULD NOT BE CONFUSED WITH ARITHMETIC OPERATORS USED IN OPERAND EXPRESSIONS.

4. 4. 0 OPERANDS.

AN OPERAND REPRESENTS THE PART OF THE INSTRUCTION WHICH IS TO BE ACTED ON. IT CAN BE A TERM OR AN EXPRESSION.

THE . BYTE, . DBYTE, AND . ADDR PSEUDO-OPS CAN HAVE MULTIPLE OPERANDS.

REFER TO THE EXPLANATION OF EACH OPERATOR FOR THE PROPER OPERAND FORMAT.

IT SHOULD BE NOTED THAT OPERAND EXPRESSIONS ARE EVALUATED TO A SINGLE NUMERICAL VALUE BY THE ASSEMBLER. BINARY CODE IS NOT GENERATED TO MAKE THE MICROPROCESSOR EVALUATE THE EXPRESSION.

4. 5. 0 TERMS AND EXPRESSIONS.

A TERM IS A SINGLE VALUE, A CONSTANT OR SYMBOL. THE CURRENT LOCATION COUNTER (REPRESENTED BY A PERIOD) IS CONSIDERED A TERM.

TERMS ARE COMBINED WITH OPERAND ARITHMETIC OPERATORS TO FORM EXPRESSIONS.

EXAMPLE: IN THE INSTRUCTION BELOW THE OPERAND IS AN EXPRESSION WHICH HAS TWO ARITHMETIC OPERATORS AND THREE TERMS.

```
SYMBOL .EQU 1+NEW * 15
```

16 BIT INTEGER ARITHMETIC IS USED TO EVALUATE EXPRESSIONS.

5. 0. 0 NUMERIC CONSTANTS.

A CONSTANT IS A NUMERIC VALUE REPRESENTED BY A STRING OF DIGITS. THE DEFAULT RADIX OR TEMPORARY RADIX INDICATORS IDENTIFY THE RADIX OF THE CONSTANT. A CONSTANT WITHOUT ANY TEMPORARY RADIX INDICATOR IS CONSIDERED TO BE IN THE DEFAULT RADIX, WHICH IS INITIALLY HEXADECIMAL.

EXAMPLE: THE HEXADECIMAL NUMBER 16 (22 IN BASE 10) IS STORED IN "VALUE" :

```
0 16 VALUE .EQU 16
```

THE MAXIMUM VALUE FOR A CONSTANT IS 65535 (BASE 10 UNSIGNED).

THE MINIMUM VALUE FOR A CONSTANT IS -32768 (BASE 10 SIGNED).

5. 1. 0 CONSTANTS WITH RADIX INDICATORS.

CONSTANTS IN A BASE DIFFERENT FROM THAT OF THE DEFAULT RADIX CAN BE SPECIFIED THROUGH USE OF THE TEMPORARY RADIX INDICATORS. THESE INDICATORS ARE VERY USEFUL FOR ENTERING INDIVIDUAL CONSTANTS. HOWEVER, IF A LARGE GROUP OF VALUES IN ANOTHER RADIX MUST BE ENTERED, IT IS MORE CONVENIENT TO CHANGE THE DEFAULT RADIX USING THE PSEUDO-OPS DESCRIBED IN SECTION # 9. 2. 0 .

THE TEMPORARY RADIX INDICATORS ARE:

```

^B    BINARY
^D    DECIMAL
^H    HEXADECIMAL
^O    OCTAL

```

THE ^ IS THE UPARROW CHARACTER (UNIVERSAL UNARY OPERATOR).

A HEXADECIMAL CONSTANT WHICH DOES NOT BEGIN WITH A NUMBER SHOULD BE WRITTEN WITH A LEADING ZERO TO DISTINGUISH IT FROM FROM A SYMBOL. A RADIX INDICATOR PRECEDING A SYMBOL IS IGNORED.

EXAMPLE: THE FIRST STATEMENT IS VALID, THE SECOND IS NOT.

```

VALUE .EQU ^HOA302          ;VALUE=A302, BASE 16
VALUE .EQU ^HA302          ;VALUE = SYMBOL A302

```

SINCE THE SYMBOL A302 MAY NOT EXIST, THE SECOND STATEMENT WILL PROBABLY CAUSE AN UNDEFINED SYMBOL ERROR. TEMPORARY RADIX INDICATORS AFFECT THE NEXT DIGIT STRING IN THE EXPRESSION UNLESS A SYMBOL NAME OR BINARY OPERATOR OCCURS FIRST. IN THAT CASE, THE TEMPORARY RADIX INDICATOR WOULD BE IGNORED. NO ERROR MESSAGE IS GIVEN.

5. 2. 0 CONSTANTS WITH ASCII INDICATORS.

THE " AND ' INDICATORS ARE USED TO FORM THE 7 BIT ASCII VALUE OF A CHARACTER. THERE ARE FOUR ACCEPTABLE WAYS TO WRITE THE INDICATORS:

```
"A" OR 'A' OR ^A OR ^A    ALL EQUAL 41 (BASE 16).
```

NOTE THAT THE CLOSING QUOTE IS OPTIONAL, BUT IF USED IT MUST MATCH THE OPENING QUOTE. ONLY ONE CHARACTER CAN FOLLOW THE INDICATOR.

THE " IS SPECIALLY HANDLED IN THE .BYTE PSEUDO-OP WHERE IT IS USED TO INPUT TEXT STRINGS. SEE SECTION # 9. 3. 1 .

6. 0. 0 SYMBOLS.

THE WORD "SYMBOL" IS USED HERE AS A GENERAL TERM FOR ANY MNEMONIC WHICH IS TO HAVE A VALUE. THIS IS IN CONTRAST TO AN OPERATOR, WHICH IS A MNEMONIC WHICH SPECIFIES A PROCESS.

A LABEL IS A SYMBOL THAT PRECEDES AN OPERATOR IN THE STATEMENT. IF THE LABEL IS USED TO STORE THE VALUE OF THE CURRENT LOCATION COUNTER , IT IS CALLED A LOCATION TAG.

6. 1. 0 PERMANENT SYMBOLS.

PERMANENT SYMBOLS ARE THE CROSS-ASSEMBLER PSEUDO-OPS AND MICROPROCESSOR OPERATORS. IF NECESSARY, THE .DINST STATEMENT CAN BE USED TO RENAME A MICROPROCESSOR OPERATOR. THE CROSS-ASSEMBLER PSEUDO-OPS CANNOT BE USED IN A .DINST INSTRUCTION. THE TABLES IN THE APPENDICES SUMMARIZE THE PERMANENT SYMBOL SET.

6. 2. 0 USER DEFINED SYMBOLS.

THESE SYMBOLS CAN BE LOCATION TAGS OR REPRESENT A VALUE.

A SYMBOL IS A STRING OF FROM ONE TO SIX ALPHANUMERIC CHARACTERS DELIMITED BY A NON-ALPHANUMERIC CHARACTER. USER-DEFINED SYMBOLS MUST CONFORM TO THE FOLLOWING RULES:

- 1) THE CHARACTERS MUST BE LEGAL ALPHA-NUMERICS.
(A-Z OR 0-9)
- 2) THE FIRST CHARACTER MUST BE ALPHABETIC (A-Z).
- 3) ONLY THE FIRST SIX CHARACTERS ARE USED, ANY OTHERS ARE IGNORED. SYMBOLS ARE STORED IN THE SYMBOL TABLE AND REFERENCED ONLY BY THE FIRST SIX CHARACTERS.
- 4) A USER-DEFINED SYMBOL CANNOT HAVE THE SAME NAME AS ANY OF THE PERMANENT SYMBOL NAMES. AS THE PERIOD IS CONSIDERED AS PART OF THE ASSEMBLER PSEUDO-OP NAME, A USER-DEFINED SYMBOL WHICH IS IDENTICAL EXCEPT FOR THE LEADING PERIOD IS LEGAL.

6.3.0 LOCAL SYMBOLS.

OFTEN, WHEN PROGRAMMING SHORT SECTIONS OF CODE WHICH INVOLVE NUMEROUS JUMP OR BRANCHING INSTRUCTIONS, THE USER FINDS IT DIFFICULT TO CREATE MEANINGFUL LABELS THAT WILL NOT CONFLICT WITH OTHER SYMBOLS IN THE PROGRAM. IN CASES LIKE THIS, LOCAL SYMBOLS CAN BE USED INSTEAD OF REGULAR SYMBOLS.

LOCAL SYMBOLS HAVE THE FORMAT "\$N" WHERE "N" IS A DECIMAL INTEGER FROM 0-255 INCLUSIVE.

LOCAL SYMBOLS MUST BE DEFINED AND REFERENCED WITHIN LOCAL SYMBOL BLOCKS. LOCAL SYMBOL BLOCKS ARE SECTIONS OF THE PROGRAM THAT START ON A STATEMENT HAVING A REGULAR SYMBOL USED AS A LOCATION TAG AND END ON THE STATEMENT JUST BEFORE THE OCCURANCE OF THE NEXT REGULAR SYMBOL LOCATION TAG. NOTE THAT LABELS FOR THE .EQU, .DINST AND .SET PSEUDO-OPS ARE NOT LOCATION TAGS AND DO NOT DELIMIT LOCAL SYMBOL BLOCKS.

THERE IS NO EFFECTIVE LIMIT TO THE SIZE OF A LOCAL SYMBOL BLOCK.

THE SAME LOCAL SYMBOL CAN BE DEFINED AND USED IN AN UNLIMITED NUMBER OF LOCAL SYMBOL BLOCKS.

EXAMPLE:

```

TAG1  . BYTE  "TEXT"  ; SYMBOL BLOCK BEGINS
$1    . EQU   VALUE   ; DEFINE LOCAL $1
$2    . EQU   -1      ; DEFINE LOCAL $2
VALU1 . EQU   $1-$2   ; CALCULATE NEW VALUE
TAG2  . BYTE  "TEXT"  ; NEW SYMBOL BLOCK
$1    . EQU   VALU1   ; DEFINE LOCAL $1
$2    . EQU   -2      ; DEFINE LOCAL $2
VALU2 . EQU   $1*$2   ; CALCULATE NEW VALUE.
TAG3  . BYTE  "TEXT"  ; ENDS SECOND BLOCK

```

7.0.0 CURRENT LOCATION COUNTER.

THE CURRENT LOCATION COUNTER IS INDICATED BY A PERIOD. IT REPRESENTS THE ADDRESS OF THE NEXT BYTE TO BE ASSEMBLED.

THE CURRENT LOCATION COUNTER CANNOT BE USED IN THE LABEL FIELD.

7. 0. 0

AT THE BEGINNING OF THE SOURCE INPUT THE CURRENT LOCATION COUNTER IS SET TO ZERO. IT CAN BE REASSIGNED THROUGH USE OF THE .ORG PSEUDO-OP.

EXAMPLE:

```
      0 60          .ORG      60      ; INITIAL ADDRESS
      0  0          .EQU      0      ; NO EFFECT ON .
    60 22          TAG      .BYTE    22      ; . = 60 (BASE 8)
      1 00          .ORG     100      ; REASSIGN COUNTER
    100 10         TAG1     .BYTE    10      ; . = 100
```

LOCATION TAGS ARE ALWAYS SET EQUAL TO THE VALUE OF THE CURRENT LOCATION COUNTER WHEN THEY ARE ASSEMBLED. IN THE EXAMPLE ABOVE, THE LOCATION TAG "TAG" = 60.

THE CURRENT LOCATION COUNTER IS AUTOMATICALLY UPDATED IN THE ASSEMBLER AS SOON AS THE CURRENT INSTRUCTION IS ASSEMBLED. NOTE THAT IN THE MULTI-OPERAND DATA STORAGE PSEUDO-OPS, (. BYTE, . DBYTE, AND . ADDR) THE LOCATION COUNTER IS CHANGING AS THE OPERANDS ARE ASSEMBLED.

EXAMPLE: THE LOCATION COUNTER IS USED AS AN OPERAND 3 TIMES IN AN . ADDR PSEUDO-OP.

```
      0 20          .ORG      20
    20 20  0        . ADDR    . . . .
    22 22  0
    24 24  0
    20 20  0
```

THE CURRENT LOCATION COUNTER USES THE FULL ADDRESS RANGE OF THE MICROPROCESSOR.

8. 0. 0 THE ARITHMETIC OPERATOR SET.

THERE ARE TWO TYPES OF ARITHMETIC OPERATORS: UNARY AND BINARY OPERATORS.

UNARY OPERATORS ACT ON ONLY ONE ITEM, THE TERM OR EXPRESSION FOLLOWING THEM.

BINARY OPERATORS ACT ON TWO ITEMS: THE TERM OR EXPRESSION PRECEDING THEM AND THE TERM OR EXPRESSION FOLLOWING THEM.

8. 1. 0 UNARY OPERATORS.

THE + (PLUS) AND - (MINUS) UNARY OPERATORS ASSIGN A POSITIVE OR NEGATIVE SIGN TO THE EXPRESSION FOLLOWING THEM. AN EXPRESSION IS ASSUMED TO BE POSITIVE IF NOT OTHERWISE SPECIFIED.

8. 1. 2 BYTE ACCESS OPERATORS.

THE ^L AND ^M (WHERE ^ IS THE UPARROW CHARACTER) ARE UNARY OPERATORS WHICH PROVIDE ACCESS TO THE LEAST AND MOST SIGNIFICANT 8 BIT BYTES OF THE VALUE OF AN EXPRESSION OR TERM.

EXAMPLE: TO SET "VALUE" EQUAL TO THE MOST SIGNIFICANT BYTE OF 3B61 (BASE 16), THE STATEMENT BELOW IS USED.

```
VALUE . SET ^M3B61 ; VALUE = 003B
```

THIS NEXT STATEMENT TAKES THE LEAST SIGNIFICANT BYTE.

```
VALUE . SET ^L3B61 ; VALUE = 0061
```

BYTE ACCESS OPERATORS MAY BE COMBINED WITH THE OTHER UNARY OPERATORS AND THE RADIX INDICATORS.

8. 1. 3 THE COMPLEMENT OPERATOR.

THE ^C (UPARROW C) IS A LOGICAL UNARY OPERATOR WHICH COMPLEMENTS THE EXPRESSION FOLLOWING IT.

EXAMPLE:

```
VALUE . EQU ^C7241 ; VALUE = 8DBE
```

THE COMPLEMENT OPERATOR CAN BE COMBINED WITH THE OTHER UNARY OPERATORS AND THE RADIX INDICATORS.

8. 1. 4. ? OPERATOR.

THIS IS THE USER FLAG OPERATOR, A UNARY OPERATOR USED IN CONJUNCTION WITH THE COMMAND DECODER USER FLAG OPTIONS (/0 TO /9). IT HAS THE FORM ?EXPRESSION AND MAY BE USED IN OPERANDS LIKE ANY OTHER TERM. THE RESULTING VALUE OF THE QUESTION MARK OPERATOR EQUALS 1 IF THE VALUE OF ITS EXPRESSION MATCHES A USER FLAG THAT WAS SPECIFIED TO THE COMMAND DECODER AT RUN-TIME. OTHERWISE IT EQUALS 0. THIS OPERATOR IS USEFUL FOR CONTROLLING CONDITIONAL ASSEMBLY AND LISTING PARAMETERS WITHOUT HAVING TO MODIFY THE SOURCE FILE.

EXAMPLE: THE /2 OPTION WAS SPECIFIED TO THE COMMAND DECODER AT RUN-TIME.

```
. R XNAME
  *BIN, LOUT<SOURCE/2
```

THE SOURCE FILE CONTAINS THE FOLLOWING .LIST STATEMENTS:

```
. LIST      ?2-1
.
. LIST      1
```

AT THE FIRST .LIST STATEMENT, THE ?2 TERM EQUALS 1 SINCE /2 WAS SPECIFIED AT RUN-TIME. THE OPERAND (?2-1) EQUALS ZERO. THEREFORE LISTING IS INHIBITED UNTIL THE SECOND .LIST INSTRUCTION. AS THE OPERAND VALUE OF THIS STATEMENT IS 1, LISTING IS ENABLED AGAIN. NOTE THAT IF THE /2 OPTION WAS NOT SPECIFIED, THE INSTRUCTIONS AFTER THE FIRST .LIST WOULD BE INCLUDED IN THE "LOUT" FILE LISTING.

8. 2. 0 BINARY OPERATORS.

SIX SPECIAL CHARACTERS ARE USED TO PERFORM THE FOLLOWING BINARY OPERATIONS:

*	MULTIPLICATION
/	DIVISION
&	BOOLEAN AND
!	INCLUSIVE OR
+	ADDITION
-	SUBTRACTION

8.2.0

THE UNARY OPERATORS TAKE PRECEDENCE OVER THE BINARY OPERATORS DURING ASSEMBLY. THE * AND / OPERATORS ARE EXECUTED NEXT, THEN THE OTHER BINARY OPERATORS FROM LEFT TO RIGHT. BRACKETS, [AND], ARE USED TO CHANGE THE ORDER OF PRECEDENCE WHEN NECESSARY. A [IS A SHIFT/K ON TTY KEYBOARDS, AND A] IS A SHIFT/M.

EXAMPLE: IN THE STATEMENT BELOW THE OPERAND EXPRESSION IS EVALUATED IN THIS ORDER: [A* [-B]] + [[2/D] * [^C [^B101]]]

VALUE .EQU A*-B+2/D*^C^B101

ADDITION AND SUBTRACTION ARE ACCOMPLISHED BY TWO'S COMPLEMENT 16 BIT ARITHMETIC. NO CHECKS FOR OVERFLOW ARE MADE.

MULTIPLICATION IS ACCOMPLISHED BY REPEATED ADDITION. NO CHECKS FOR SIGN OR OVERFLOW ARE MADE.

DIVISION IS ACCOMPLISHED BY REPEATED SUBTRACTION. THE QUOTIENT IS THE NUMBER OF SUBTRACTIONS PERFORMED. THE REMAINDER IS NOT SAVED. NO CHECKS ARE MADE FOR SIGN. DIVISION BY ZERO RESULTS IN ZERO.

THE BOOLEAN AND FUNCTION (&) IS A BIT BY BIT LOGICAL AND OF TWO NUMBERS:

THE BOOLEAN INCLUSIVE OR (!) IS A BIT BY BIT LOGICAL OR OF TWO NUMBERS.

9. 0. 0 PSEUDO-OPERATORS.

PSEUDO-OPERATORS ARE INSTRUCTIONS TO THE ASSEMBLER WHICH ALLOW GREATER FLEXIBILITY IN PROGRAMMING.

A SUMMARY OF THE PSEUDO-OPS AND THEIR FUNCTIONS IS GIVEN IN THE APPENDIX.

9. 1. 0 ASSIGNMENT PSEUDO-OPS.

ASSIGNMENT PSEUDO-OPS ARE USED TO DEFINE VALUES, INPUT ASCII TEXT AND REASSIGN THE LOCATION COUNTER.

9. 1. 1 .EQU PSEUDO-OP.

THE .EQU IS USED TO ASSIGN A VALUE TO A SYMBOL. THIS SYMBOL VALUE CANNOT BE CHANGED ONCE DEFINED. .EQU IS USEFUL FOR ASSIGNING NAMES TO LOCATIONS WHICH ARE NOT LOADED BY THE OBJECT CODE.

EXAMPLE:

```
NAME1 .EQU 300*6
```

9. 1. 2 .SET PSEUDO-OP.

THE .SET IS USED EXACTLY LIKE THE .EQU EXCEPT THAT THE SYMBOL CAN BE REDEFINED WITH ANOTHER .SET AT ANY POINT IN THE PROGRAM:

EXAMPLE: THE FOLLOWING IS PERFECTLY LEGAL FOR A .SET BUT NOT AN .EQU.

```
NAME1 .SET 300*6
NAME1 .SET 22
```

NOTE THAT IT IS GOOD PRACTICE TO USE THE .EQU FOR ASSIGNMENTS RATHER THAN THE .SET EXCEPT (OF COURSE) WHERE THERE IS A SPECIFIC NEED TO REDEFINE A VALUE. THIS HELPS PREVENT THE ACCIDENTAL REDEFINITION OF A VALUE IN A PROGRAM.

9. 1. 3 .DINST PSEUDO-OP.

THE .DINST IS USED TO GIVE A MICROPROCESSOR OPERATOR ANOTHER NAME. THE ORIGINAL OPERATOR NAME WILL STILL BE VALID. NOTE THAT THE ASSEMBLER PSEUDO-OPS CANNOT BE RENAMED.

#9. 1. 3

EXAMPLE: THE MICROPROCESSOR INSTRUCTION "OPR" IS DEFINED AS "NEWOP". ANY FURTHER REFERENCES TO "NEWOP" IN THE PROGRAM WILL BE TREATED ACCORDING TO THE DEFINITION OF "OPR".

NEWOP .DINST OPR

"NEWOP" IS DEFINED TO BE THE EQUIVALENT TO THE MICROPROCESSOR INSTRUCTION "OPR" AND IS ADDED TO THE OPERATOR SET FOR THE REMAINDER OF THE ASSEMBLY.

REFERENCES TO USER DEFINED OPERATORS ARE NOT ALLOWED TO PRECEDE THEIR .DINST STATEMENT.

ASSEMBLER PSEUDO-OPS CANNOT BE USED IN EITHER THE LABEL OR OPERAND FIELDS OF ANY STATEMENT AND THEREFORE CANNOT BE DEFINED WITH THE .DINST STATEMENT.

LOCAL SYMBOLS CANNOT BE USED IN THE OPERATOR FIELDS, THEREFORE THEY SHOULD NOT BE USED WITH A .DINST STATEMENT.

9. 1. 4 .ORG PSEUDO-OP.

THE .ORG REASSIGNS THE LOCATION COUNTER.

THE LOCATION COUNTER WILL BE 0 AT THE START OF THE SOURCE INPUT.

THE .ORG OPERAND CANNOT BE FORWARD REFERENCED, (REFERRED TO A LABEL DEFINED FURTHER ON IN THE PROGRAM) AND CANNOT HAVE A LABEL.

9. 2. 0 DEFAULT RADIX PSEUDO-OPS.

INITIALLY, THE DEFAULT RADIX IS SET TO HEXADECIMAL SO THAT CONSTANTS ARE READ IN AS BASE 16 VALUES. (SEE MODIFICATION NOTES IF ANOTHER INITIAL DEFAULT RADIX IS DESIRED.)

AT ANY POINT IN THE PROGRAM, THE DEFAULT RADIX CAN BE REASSIGNED THROUGH USE OF THESE PSEUDO-OPS:

.BIN	;BINARY RADIX
.DECM	;DECIMAL RADIX
.HEX	;HEXADECIMAL RADIX
.OCT	;OCTAL RADIX

THE DEFAULT RADIX PSEUDO-OPS CANNOT HAVE AN OPERAND OR A LABEL.

ADDITIONALLY, THE RADIX OF INDIVIDUAL CONSTANTS CAN BE SPECIFIED BY THE USE OF THE ^B, ^D, ^H AND ^O INDICATORS. SEE SECTION # 5. 1. 0 . THESE INDICATORS DO NOT CHANGE THE DEFAULT RADIX.

9. 3. 0 DATA STORAGE PSEUDO-OPS.

THREE PSEUDO-OPS CAN BE USED TO STORE DATA. THEIR FORMAT IS:

```
LABEL      PSEUDO-OP      OPERAND, OPERAND, . . . . ; COMMENT
```

THE PSEUDO-OPS CAN HAVE AS MANY OPERANDS AS WILL FIT ON ONE 127 CHARACTER LINE.

EACH OPERAND CAN BE A SYMBOL, CONSTANT, OR EXPRESSION. COMMAS SEPARATE THE OPERANDS.

THE DOUBLE QUOTE (") CHARACTER IS USED DIFFERENTLY IN THE .BYTE COMMAND, BUT THE SINGLE QUOTE (') RETAINS ITS NORMAL FUNCTION.

9. 3. 1 .BYTE PSEUDO-OP.

THE .BYTE PSEUDO-OP STORES DATA IN SINGLE BYTES OF MEMORY. NUMERICAL BYTE VALUES CAN RANGE FROM -128 TO +255 (DECIMAL). NORMALLY, DOUBLE QUOTES AND SINGLE QUOTES ARE TREATED IDENTICALLY AND ARE USED TO FORM THE ASCII VALUE OF A SINGLE CHARACTER. HOWEVER, IN THE .BYTE PSEUDO-OP, THE DOUBLE QUOTE IS USED TO INDICATE TEXT STRINGS. DATA IS STORED SEQUENTIALLY AS IT IS PROCESSED, LEFT TO RIGHT. A TEXT STRING MUST BE CLOSED WITH A DOUBLE QUOTE.

EXAMPLE: THE ASCII VALUES OF THE TEXT ABC IS STORED:

```

          2 00      .ORG      200
200 41          .BYTE      "ABC", 0, 'B
201 42
202 43
203  0
204 42
```

THESE STATEMENTS WOULD BE INVALID:

```

      .BYTE      'ABC'      ; THE ' IS NOT FOR TEXT STRINGS
      .BYTE      "ABC      ; TEXT MUST END WITH A "
```

9. 3. 2 .DBYTE PSEUDO-OP.

THE .DBYTE IS SIMILAR TO THE .BYTE EXCEPT THAT IT STORES DOUBLE BYTE QUANTITIES. IT DOES NOT ACCEPT TEXT STRINGS. THE THE MOST SIGNIFICANT BYTE IS STORED FIRST, THEN THE LEAST SIGNIFICANT BYTE.

9. 3. 3 . ADDR PSEUDO-OP.

THE . ADDR PSEUDO-OP IS THE SAME AS THE . DBYTE PSEUDO-OP EXCEPT THAT THE LEAST SIGNIFICANT BYTE IS STORED FIRST. MANY MICROPROCESSORS USE THIS REVERSED FORMAT FOR ADDRESSES. FOR EXAMPLE:

```

      2 00      . ORG      200
200  1 32      . DBYTE   ^H3132 ; HEX CONSTANT
202  32 31      . ADDR    ^H3132 ; REVERSED BYTES

```

9. 3. 4 . ZERO PSEUDO-OP.

THE . ZERO PSEUDO-OP RESERVES THE NUMBER OF BYTES INDICATED BY THE OPERAND AND SETS THEM TO ZERO.

EXAMPLE: 16 ADDRESSES, 1 TO 10 (BASE 16) ARE ZEROED.

```

      0 1      . ORG      1
      1 0      . ZERO     10
      11 10    . BYTE     10

```

ONLY THE FIRST BYTE WILL BE PRINTED IN THE LISTING. THE LOCATION COUNTER IS ADVANCED. THE OPERAND OF . ZERO CANNOT BE FORWARD REFERENCED, (REFERRED TO A LABEL DEFINED FURTHER ON IN THE PROGRAM).

9. 4. 0 LISTING CONTROL DIRECTIVES.

THROUGH USE OF THE . LIST, . PAGE AND . TITLE PSEUDO-OPS, PLUS SEVERAL RUN-TIME OPTIONS, THE SOURCE PROGRAM CAN BE LISTED IN VARIOUS WAYS AT ASSEMBLY TIME.

NORMALLY, THE ASSEMBLER AUTOMATICALLY PAGES THE OUTPUT, ADDING A HEADER AT THE TOP OF THE PAGE. (NOTE THAT PAGE NUMBERS REPRESENT THE LISTING PAGE NUMBERS, NOT INPUT FILE PAGES.)

NOT ALL PSEUDO-OPS ARE LISTED IN THE OUTPUT. THE CONDITIONAL ASSEMBLY AND LISTING CONTROL PSEUDO-OPS ARE NOT LISTED UNLESS THE /P OPTION IS SPECIFIED. SEE RUN-TIME OPTIONS # 2. 4. 0 .

NORMALLY THE STATEMENTS WHICH ARE NOT ASSEMBLED DUE TO CONDITIONAL ASSEMBLY ARE NOT LISTED. USE OF THE /J COMMAND DECODER OPTION WILL ENABLE LISTING OF THESE STATEMENTS PLUS THE NORMALLY UNLISTED CONDITIONAL ASSEMBLY PSEUDO-OPS.

THE PAGINATION AND HEADING CAN BE SUPPRESSED THROUGH USE OF THE /H COMMAND DECODER OPTION.

9.4.0

IF THE OUTPUT DEVICE IS ONE WHICH DOES NOT PAGE ON A FORM FEED (A TTY), THE /T DECODER OPTION CAN BE USED TO CHANGE THE FORM FEED (WHICH NORMALLY STARTS A NEW PAGE) TO 3 CARRIAGE RETURN/LINE FEEDS SO THAT PAGES WILL BE SEPARATED BY 3 BLANK LINES IN THE LISTING.

WARNING MESSAGES ARE NORMALLY OUTPUT TO BOTH THE TERMINAL AND THE SOURCE LISTING. TO INHIBIT THESE MESSAGES, THE /W DECODER OPTION IS USED.

9.4.1 .LIST PSEUDO-OP.

A LIST FLAG IS USED DURING ASSEMBLY TO INDICATE WHETHER OR NOT THE STATEMENTS ARE TO BE LISTED. INITIALLY, THE FLAG IS ON AND STAYS ON UNLESS A .LIST PSEUDO-OP IS ENCOUNTERED.

A .LIST PSEUDO-OP CAN BE USED WITH OR WITHOUT AN OPERAND. A LABEL CANNOT BE USED WITH THE .LIST PSEUDO-OP.

WHEN A .LIST PSEUDO-OP WITHOUT AN OPERAND IS ENCOUNTERED, THE LIST FLAG IS INVERTED.

EXAMPLE:

	.ORG	200	;LIST FLAG INITIALLY ON
			;LISTED
VALUE	.SET	1	;LISTED
	.LIST		;LIST FLAG OFF
VALU2	.SET	70	;NOT LISTED
	.LIST		;LIST FLAG BACK ON

NOTE THAT UNLESS THE /P OPTION IS USED, THE .LIST OPERATOR ITSELF WILL NOT BE LISTED.

WHEN A .LIST PSEUDO-OP WITH AN OPERAND IS ENCOUNTERED, THEN LISTING IS INHIBITED IF THE OPERAND IS EQUAL TO ZERO. (THE LIST FLAG IS SET OFF). IF THE OPERAND IS NOT ZERO, LISTING IS ENABLED. (THE LIST FLAG IS SET ON).

9.4.2 .PAGE PSEUDO-OP.

INSERTING A .PAGE PSEUDO-OP IN THE PROGRAM WILL NORMALLY START A NEW PAGE BEGINNING WITH THE NEXT LINE. (THE .PAGE STATEMENT ITSELF IS NOT NORMALLY LISTED.) IF THE /P COMMAND DECODER OPTION IS USED, THE .PAGE STATEMENT WILL BE THE FIRST LINE OF THE NEW PAGE.

9.4.2

THE /H COMMAND DECODER OPTION INHIBITS THE .PAGE PSEUDO-OP.

THE .PAGE PSEUDO-OP CAN HAVE NO LABEL OR OPERAND.

9.4.3 .TITLE PSEUDO-OP.

THE .TITLE IS USED TO REPLACE THE HEADING WITH UP TO 32 CHARACTERS OF TEXT. ITS FORMAT IS:

.TITLE HEADING OF 32 CHARACTERS

THE FIRST CHARACTER AFTER THE .TITLE IS THE PSEUDO-OP DELIMITER WHICH CANNOT BE AN ALPHA-NUMERIC CHARACTER. THE DELIMITER IS CONSIDERED THE FIRST CHARACTER OF THE 32 CHARACTER GROUP AND WILL BE PRINTED OUT. ANY TEXT AFTER 32 CHARACTERS WILL BE IGNORED. TABS CAN BE USED IN THE HEADING.

THE /H COMMAND DECODER OPTION INHIBITS THE .TITLE PSEUDO-OP.

THE /P COMMAND DECODER ENABLES THE LISTING OF THE .TITLE PSEUDO-OP.

A SEMICOLON DOES NOT DELIMIT THE HEADING TEXT. COMMENTS CAN BE MADE ONLY AFTER THE 32 CHARACTER HEADING GROUP.

WHEN PLACED AT THE BEGINNING OF THE PROGRAM, THE .TITLE PSEUDO-OP WILL SET THE HEADING FOR THE FIRST PAGE. THE .TITLE MUST APPEAR BEFORE THE FIRST LINE TO BE LISTED.

EXAMPLE: THE FOLLOWING STATEMENTS WILL CAUSE THE HEADING OF THE FIRST PAGE TO BE "*MAIN PROGRAM".

```
                .TITLE*MAIN PROGRAM
VALUE          .EQU    1
                .LIST  VALUE
```

9.5.0 CONDITIONAL ASSEMBLY PSEUDO-OPERATORS.

THE .IFZERO, .IFNZRO, .IFDEF AND .IFNDEF OPERATORS ARE USED TO PROVIDE FOR THE CONDITIONAL ASSEMBLY IN A PROGRAM, SO THAT GROUPS OF STATEMENTS CAN BE ADDED (OR OMITTED) DURING THE ASSEMBLY PROCESS. EACH IS DESCRIBED INDIVIDUALLY IN THE SECTIONS THAT FOLLOW. ALL HAVE THE GENERAL FORM:

PSEUDO-OP OPERAND ; COMMENT

9. 5. 1 . IFZERO PSEUDO-OP.

IF THE OPERAND OF THE . IFZERO IS:

- EQUAL TO ZERO - ASSEMBLY IS UNAFFECTED.
- NOT EQUAL TO ZERO - STATEMENTS TO NEXT . ENDC ARE OMITTED.

THE OPERAND CANNOT BE FORWARD REFERENCED.

9. 5. 2 . IFNZRO PSEUD-OP.

IF THE OPERAND OF THE . IFNZRO IS:

- EQUAL TO ZERO - STATEMENTS TO NEXT . ENDC ARE OMITTED.
- NOT EQUAL TO ZERO - ASSEMBLY IS UNAFFECTED.

THE OPERAND CANNOT BE FORWARD REFERENCED.

9. 5. 3 . IFDEF PSEUDO-OP.

IF THE SYMBOL OPERAND OF THE . IFDEF IS:

- DEFINED - ASSEMBLY IS UNAFFECTED.
- NOT DEFINED - STATEMENTS TO NEXT . ENDC ARE OMITTED.

NOTE THAT . IFDEF WILL ACCEPT ONLY A SINGLE SYMBOL NAME AS THE OPERAND.

A SYMBOL IS CONSIDERED TO BE DEFINED IF IT HAS BEEN USED IN THE LABEL FIELD OF A STATEMENT PRECEEDING THE CONDITIONAL PSEUDO-OP.

9. 5. 4 . IFNDEF PSEUDO-OP.

IF THE SYMBOL OPERAND OF THE . IFNDEF IS:

- DEFINED - STATEMENTS TO NEXT . ENDC ARE OMITTED.
- NOT DEFINED - ASSEMBLY IS UNAFFECTED.

NOTE THAT ONLY A SINGLE SYMBOL NAME IS ALLOWED AS THE OPERAND.

A SYMBOL IS CONSIDERED TO BE DEFINED IF IT HAS BEEN USED IN THE LABEL FIELD OF A STATEMENT PRECEEDING THE CONDITIONAL PSEUDO-OP.

9. 5. 5 . ENDC PSEUDO-OP.

THIS PSEUDO-OP INDICATES THE END OF A CONDTIONAL ASSEMBLY GROUP.
EVERY CONDITIONAL PSUEDO-OP MUST BE PAIRED WITH A . ENDC.

9. 6. 0 . END PSEUDO-OP.

THIS INDICATES THE END OF THE SOURCE PROGRAM. IT CANNOT HAVE EITHER
A LABEL OR AN OPERAND. A WARNING MESSAGE WILL OCCUR IF THE . END
STATEMENT IS LEFT OFF.

#10. 0. 0 ERROR MESSAGES AND WARNINGS.

BOTH PASS #1 AND PASS #2 CAN GENERATE ERROR MESSAGES. THESE ARE
PRINTED ON THE CONSOLE DEVICE AS THEY OCCUR. IF A LISTING IS
SPECIFIED, PASS 3 WILL LIST THE ERROR MESSAGE ABOVE THE LINE IN
WHICH THE ERROR OCCURS.

ERROR MESSAGES WHICH ARE SENT TO THE CONSOLE HAVE THE FORM:

E: XX AT LABEL+N

WHERE "N" IS A DECIMAL NUMBER OF
LINES BEYOND THE STATEMENT WHICH
CONTAINED THE GIVEN LABEL. IF NO
LABEL WAS GIVEN, "N" IS THE NUMBER OF
LINES FROM THE BEGINNING LINE OF THE
PROGRAM.

IF THE BINARY OUTPUT FILE IS SENT TO THE CONSOLE, AND ERROR
MESSAGES OCCUR, THE OUTPUT FILE LINES AND ERROR MESSAGES WILL BE
INTERMIXED. USE OF THE /E OPTION WILL INHIBIT THE ERROR MESSAGES
TO THE CONSOLE SO THAT ONLY THE BINARY FILE IS OUTPUT. THIS IS
USEFUL WHEN A USER WOULD LIKE TO TRY OUT CERTAIN PARTS OF A PROGRAM
AND IS NOT YET CONCERNED WITH OTHER PARTS KNOWN TO HAVE ERRORS.

INDIVIDUAL ERROR MESSAGES ARE EXPLAINED IN TABLE #2 WHICH DIVIDES THE MESSAGES INTO THREE TYPES:

1) FATAL ERRORS- THESE ERRORS CAUSE THE IMMEDIATE EXIT TO THE OS/8 MONITOR. THE CURRENT OUTPUT FILE IS NOT CLOSED. /E WILL NOT INHIBIT FATAL ERROR MESSAGES. FATAL ERROR MESSAGES ARE ALWAYS SENT TO THE CONSOLE DEVICE.

2) WARNING MESSAGES INDICATE MINOR PROGRAM PROBLEMS. ASSEMBLY IS NOT HALTED. GOOD PROGRAMMING PRACTICES WILL ELIMINATE ALL WARNING MESSAGES.

3) NON-FATAL ERRORS - THE OCCURANCE OF A NON-FATAL ERROR WILL NOT HALT ASSEMBLY. THE ASSEMBLER ATTEMPTS TO DO AS MUCH OF THE LINE AS POSSIBLE. FOR EXAMPLE, IF THE OPERAND CANNOT BE EVALUATED, IT GIVES IT A VALUE OF ZERO, WRITES THE ERROR MESSAGE AND CONTINUES.

**** FATAL ERRORS ****

E:DF - DEVICE FULL:
FILE #N THERE IS NOT ENOUGH ROOM LEFT ON THE OUTPUT DEVICE FOR THE FILE. "N" INDICATES WHICH OF THE TWO OUTPUT FILES WAS IN ERROR.

E:LT - LOCAL SYMBOL TABLE OVERFLOW:
THIS ERROR OCCURS ONLY IF THE /K OPTION IS IN USE. CONVERSION OF SOME OF THE LOCAL SYMBOLS TO REGULAR SYMBOL NAMES WILL USUALLY SOLVE THIS PROBLEM. SEE THE NOTES ON THE /K RUN-TIME OPTION.

E:OE - OPEN ERROR IN OUTPUT FILE:
FILE #N AN ATTEMPT WAS MADE TO OPEN AN OUTPUT FILE ON AN INPUT-ONLY DEVICE. (PTR:, CDR:, ETC.) "N" INDICATES WHICH ONE OF THE TWO POSSIBLE OUTPUT FILES WAS IN ERROR.

E:PE - PHASE ERROR:
A LOCATION TAG HAS A DIFFERENT ADDRESS IN ONE PASS THAN IT HAD IN THE PREVIOUS PASS.

E:RE - READ ERROR:
FILE #N AN ERROR HAS OCCURRED WHILE READING FROM AN INPUT FILE DEVICE. "N" INDICATES WHICH ONE OF THE NINE POSSIBLE INPUT FILES HAD THE ERROR.

E:ST - SYMBOL TABLE OVERFLOW:
THE PROGRAM IS TOO LARGE. WHERE CONVENIENT, DIVIDE IT AND ASSEMBLE EACH PART SEPARATELY. ALSO REFER TO THE NOTES ON THE /K RUN-TIME OPTION.

E:WE - WRITE ERROR:
FILE #N AN ERROR HAS OCCURRED WHILE WRITING TO AN OUTPUT FILE DEVICE. "N" INDICATES WHICH ONE OF THE TWO OUTPUT FILES HAD THE ERROR.

**** WARNING MESSAGES ****

W:EF - NO .END STATEMENT:
THE LAST INPUT FILE MUST HAVE AN .END STATEMENT. THE ASSEMBLER PROCEEDS AS IF AN .END WERE PRESENT.

W:UC - ASSEMBLY WAS CONDITIONALLY INHIBITED AT THE END OF THE PROGRAM: EACH CONDITIONAL ASSEMBLY PSEUDO-OP MUST BE PAIRED WITH AN .ENDC STATEMENT.

**** NON-FATAL ERRORS ****

- E: BN - BAD NESTING OF BRACKETS:
EACH OPEN BRACKET MUST BE PAIRED WITH A CLOSED BRACKET.
- E: DR - DIGIT OUTSIDE OF RADIX:
THE CONSTANT CONTAINS A DIGIT NOT RECOGNIZED UNDER THE SPECIFIED RADIX. FOR EXAMPLE, THE DIGIT "2" IS NOT RECOGNIZED IN BINARY RADIX. THE CONSTANT WILL BE EVALUATED AS IF THAT DIGIT WERE ZERO.
- E: IL - ILLEGAL LABEL FIELD:
THE LABEL MAY NOT BE IN THE PROPER SYMBOL FORMAT, SEE SECTION #6. 2. 0 . ALSO, SOME PSEUDO-OPS CANNOT HAVE LABELS.
- E: IO - ILLEGAL OPERAND VALUE:
REFER TO THE SECTION ON THE STATEMENT'S OPERATOR TO DETERMINE THE ALLOWABLE OPERAND TERMS.
- E: LO - LINE INPUT OVERFLOW:
ONLY 127 CHARACTERS, NOT INCLUDING THE CARRIAGE RETURN AND LINE FEED, ARE ALLOWED IN AN INPUT LINE.
- E: LS - LOCAL SYMBOL SYNTAX ERROR:
THE CORRECT FORMAT FOR A LOCAL SYMBOL IS \$N WHERE "N" IS A DECIMAL NUMBER FROM 0 TO 255.
- E: ML - MULTIPLE LABEL DEFINITION:
THE SAME LABEL HAS A DIFFERENT VALUE AND IS USED WITH AN OPERATOR OTHER THAN A . SET PSEUDO-OP.
- E: MO - MISSING OR ILLEGAL MNEMONIC IN OPERATOR FIELD:
- E: OC - OPERAND TOO COMPLEX:
TOO MANY TERMS AND OPERATORS EXIST IN THE OPERAND. DIVIDE THE EXPRESSION USING THE . SET COMMAND.

EXAMPLE: THE FIRST EXPRESSION IS DIVIDED INTO THE TWO STATEMENTS FOLLOWING IT.

WORD	. EQU	[EXPR1] + [EXPR2]
TEMP	. SET	[EXPR1]
WORD	. EQU	TEMP + [EXPR2]

- E: OM - OPERAND MISSING.

E: OS - OPERAND SYNTAX ERROR.

E: PS - ILLEGAL PERMANENT SYMBOL USAGE IN OPERAND:
REFER TO THE APPENDICES TABLES TO SEE WHICH NAMES
ARE USED IN THE ASSEMBLER AND MICROPROCESSOR IN-
STRUCTION SETS AND RENAME YOUR SYMBOL SO THAT IT
WILL NOT CONFLICT.

E: TL - LABEL DEFINED TOO LATE:
ONLY ONE LEVEL OF FORWARD REFERENCING IS ALLOWED.

E: US - UNDEFINED SYMBOL:

NOTE: REFER TO SECTION #12. 0. 0 FOR ADDITIONAL ERROR MESSAGES WHICH
ARE SPECIFIC TO THE TYPE OF MICROPROCESSOR BEING USED.

#11. 0. 0 MODIFICATION NOTES.

VARIOUS MODIFICATIONS CAN BE MADE TO THE ASSEMBLER FOR GREATER
OPERATING CONVENIENCE. BEFORE MAKING ANY CHANGES, THE USER SHOULD
READ THE DESCRIPTION OF EACH OPTION CAREFULLY. NO CHECKS ON PATCH
VALIDITY ARE MADE. ALSO KEEP A RECORD OF ALL CHANGES SO THAT THE
STATUS OF THE CROSS-ASSEMBLER IS ALWAYS KNOWN.

MODIFICATIONS ARE MADE BY PATCHING LOCATIONS IN THE IMAGE (.SV)
FILE USING ODT. REFER TO THE OS/8 MANUAL FOR A DETAILED EXPLAIN-
ATION OF ODT OPERATION.

THE EXAMPLE BELOW SHOWS AN ODT PATCH BEING MADE TO FILE "XNAME.SV"
WHERE THE CONTENT OF LOCATION 10107 IS CHANGED FROM 3 TO 2.

```
. GET SYS: XNAME
. ODT
10107/0003 2
^C
. SA SYS: XNAME
```

#11. 1. 0

#11. 1. 0 CHANGING THE DEFAULT INPUT FILE EXTENSION (.MS).

PATCH LOCATION 10100 TO CONTAIN THE NEW 2 CHARACTER 6 BIT ASCII EXTENSION.

#11. 2. 0 CHANGING THE DEFAULT BINARY OUTPUT FILE EXTENSION (.MB)

PATCH LOCATION 10101 TO CONTAIN THE NEW 2 CHARACTER 6 BIT ASCII EXTENSION.

#11. 3. 0 CHANGING THE DEFAULT LISTING OUTPUT FILE EXTENSION (.LS).

PATCH LOCATION 10102 TO CONTAIN THE NEW 2 CHARACTER 6 BIT ASCII EXTENSION.

#11. 4. 0 CHANGING THE BASE YEAR DATE.

IN OS/8 ONLY 3 BITS ARE PROVIDED TO INDICATE THE CURRENT YEAR. THIS ALLOWS ONLY NUMBERS FROM 0 TO 7 WHICH MUST BE ADDED TO A BASE YEAR TO FORM THE ACTUAL YEAR NUMBER. IN 1978 AND AT ADDITIONAL 8 YEAR INTERVALS THE BASE YEAR MUST BE CHANGED TO PROVIDE THE PROPER DATE PRINTOUT. TO DO THIS, PATCH LOCATION 10104 TO CONTAIN THE TWO CHARACTER 6 BIT ASCII REPRESENTATION OF THE TWO LEAST SIGNIFICANT DIGITS OF THE YEAR.

BASE YEAR:	PATCH TO LOCATION 10104 (IN OCTAL).
1978	6770
1986	7066
1994	7164
2002	6062

SHOULD THIS PROGRAM SURVIVE UNTIL THE YEAR 2000 THE TWO MOST SIGNIFICANT DIGITS MAY BE CHANGED BY PATCHING LOCATION 10103 TO CONTAIN 6260.

#11. 5. 0 CHANGING THE DEFAULT RADIX. (HEXADECIMAL)

INITIALLY THE DEFAULT RADIX IS SET TO HEXADECIMAL. THIS MAY BE MODIFIED TO BINARY, OCTAL, OR DECIMAL BY PATCHING LOCATION 10105 FROM THE FOLLOWING TABLE.

RADIX:	PATCH LOCATION 10105 TO:
OCTAL	1
HEXADECIMAL	2
DECIMAL	3
BINARY	4

#11. 6. 0 GENERATING 8 BIT ASCII CHARACTERS WITHIN THE BINARY PROGRAM.

THE ASCII CHARACTERS GENERATED AS OPERANDS WITH THE QUOTE CHARACTERS ARE SEVEN BIT REPRESENTATIONS TYPICAL OF MOST MICROPROCESSOR SYSTEMS. TO GENERATE EIGHT BIT ASCII WITH THE EIGHTH BIT ALWAYS SET (AS IS DONE IN SOME PDP8 SOFTWARE), PATCH LOCATION 10106 TO CONTAIN 377. (ORIGINAL CONTENT WAS 177).

#11. 7. 0 RUNNING UNDER OS8 VERSION 2.

THE CROSS-ASSEMBLER IS SET UP TO USE THE OS/8 VERSION 3 METHOD FOR CORE SIZE DETERMINATION. IN OS/8 V3 THE CORE SIZE IS CONTAINED IN A MONITOR LOCATION. IN PREVIOUS VERSIONS, THE CORE SIZE MUST BE DETERMINED BY ACCESSING EACH FIELD OF MEMORY TO SEE IF IT EXISTS ON THE SYSTEM. THEREFORE, TO RUN THE CROSS-ASSEMBLER UNDER VERSION 2, PATCH LOCATION 10107 TO CONTAIN 2. (ORIGINAL CONTENT WAS 3).

#11. 8. 0 CHANGING THE NUMBER OF LINES PER PAGE. (6)

THE NORMAL NUMBER OF LINES PER PAGE IS SET AT 66. 6 OF THE 66 LINES ARE USED BY THE ASSEMBLER FOR THE HEADING AND MARGIN. TO ALTER THE NUMBER OF LINES ON A PAGE, PATCH LOCATION 10110 TO BE THE TOTAL POSITIVE LINES PER PAGE INCLUDING HEADING AND MARGIN.

#11. 9. 0

#11. 9. 0 CHANGING THE NUMBER OF CHARACTERS PER LINE. (72)

THE TOTAL NUMBER OF CHARACTERS PRINTED ON ONE LINE (EXCLUDING CARRIAGE RETURN AND LINE FEED) IS SET AT 72 (BASE 10). TO MODIFY THIS COUNT, PATCH LOCATION 10111 TO CONTAIN THE POSITIVE NUMBER OF CHARACTERS TO BE PRINTED ON A LINE (EXCLUDING THE CR AND LF).

#11. 10. 0 INITIAL FORM/FEED CONTROL.

SOME LINE PRINTER HANDLERS WHEN FIRST INITIALIZED WILL ISSUE AN AUTOMATIC FORM FEED. TO AVOID EJECTING AN ADDITIONAL PAGE EACH TIME THE ASSEMBLER IS CALLED, THE FIRST FORM FEED FROM THE HEADING HAS BEEN SUPPRESSED. TO REENABLE THIS FIRST FORM FEED, PATCH LOCATION 10112 WITH 214 (BASE 8).

#11. 11. 0 CHANGING LABEL DELIMITER (,).

TO PROVIDE COMPATIBILITY WITH OTHER ASSEMBLER FORMATS AN OPTIONAL LABEL DELIMITER WILL BE ACCEPTED. NORMALLY, THIS DELIMITER IS A COMMA, BUT IT CAN BE MODIFIED TO ANY OTHER NON-ALPHANUMERIC CHARACTER (EXCEPT THE SEMICOLON OR CARRIAGE RETURN). TO MODIFY THE DELIMITING CHARACTER PATCH LOCATION 10113 WITH THE 8 BIT ASCII VALUE FOR THE CHARACTER.

#11. 12. 0 CHANGING FROM 8 BIT TO 7 BIT ASCII IN THE OUTPUT FILES.

ALL ASCII OUTPUT TO THE BINARY (OBJECT) AND LISTING FILES IS IN 8 BIT ASCII FORMAT. TO OUTPUT 7 BIT ASCII FORMAT PATCH LOCATION 10114 TO CONTAIN 177. (ORIGINAL CONTENT WAS 377).

#11. 13. 0 CHANGING THE SENSE OF THE RUN-TIME OPTIONS.

EACH SLASH OPTION (EXCEPT /0 TO /9) MAY HAVE ITS SENSE INVERTED BY PATCHING THE LOCATIONS SHOWN IN THE FOLLOWING TABLE WITH THE DESCRIBED VALUE.

OPTION:	LOCATION:	STANDARD:	INVERTED:
/B	10116	7650	7640
/E	10117	7640	7650
/H	10120	7650	7640
/J	10121	7650	7640
/K	10122	7650	7640
/L	10123	0	1
/N	10124	7650	7640
/O	10125	7650	7640
/P	10126	7640	7650
/S	10127	7650	7640
/T	10130	7650	7640
/W	10131	7650	7640

#12. 0. 0 2650 CROSS-ASSEMBLER SPECIFICS.

THE FIRST ELEVEN SECTIONS OF THIS MANUAL HAVE PRESENTED SIERRA DIGITAL'S UNIVERSAL ASSEMBLER FORMAT AS IT IS APPLIED TO ALL CROSS-ASSEMBLERS IN THE X8 SERIES. THIS SECTION PRESENTS ADDITIONAL INFORMATION ON THE APPLICATION OF THE UNIVERSAL ASSEMBLER FORMAT TO A SPECIFIC CROSS-ASSEMBLER FOR THE 2650 MICROPROCESSOR. THE 2650 MICROPROCESSOR WAS DESIGNED BY SIGNETICS CORPORATION, 811 EAST ARGUES AVENUE, SUNNYVALE, CALIFORNIA 94086. THE 2650 IS PRODUCED BY SIGNETICS AND IS ALSO SECOND SOURCED BY ADVANCED MEMORY SYSTEMS, 1276 HAMMERWOOD, SUNNYVALE, CALIFORNIA 94086. NO ATTEMPT WILL BE MADE IN THIS MANUAL TO EXPLAIN THE OPERATION OF THE MICROPROCESSOR. EXCELLENT MANUALS COVERING THE OPERATION AND PROGRAMMING OF THE MICROPROCESSORS ARE AVAILABLE FROM THEIR MANUFACTURERS. SECTION #13 PRESENTS A SUMMARY OF THE INSTRUCTION MNEMONIC CODES DEFINED BY SIGNETICS AND RECOGNIZED BY OUR CROSS-ASSEMBLER.

#12. 1. 0 CROSS-ASSEMBLER FILE NAMES.

THE CROSS-ASSEMBLER IS PROVIDED ON FILE STRUCTURED MEDIA UNDER THE NAMES:

X2650.SV - FOR THE OS/8 SAVE IMAGE FILE.
X2650.BN - FOR THE OS/8 BINARY FORMAT FILE.

IT IS SUGGESTED THAT THE SAME NAMING CONVENTIONS BE USED WHEN LOADING THE CROSS-ASSEMBLER FROM PAPER TAPE.

#12. 2. 0 LISTING FORMAT.

THE LISTING FILE IS OUTPUT WITH THE OBJECT CODE PRINTED TO THE LEFT OF THE SOURCE CODE LINES. AS EACH MICROPROCESSOR INSTRUCTION MAY CODE INTO ONE, TWO, OR THREE BYTES, ROOM IS PROVIDED FOR THREE COLUMNS OF GENERATED OBJECT CODE PLUS A COLUMN FOR THE ADDRESS. THE ADDRESS AND OBJECT CODE ARE NORMALLY PRINTED IN HEXADECIMAL BUT THIS MAY BE CHANGED TO OCTAL WITH THE /O COMMAND DECODER OPTION. SOURCE LINES WHICH EXCEED THE PRINTOUT LIMIT WILL BE CONTINUED AT COLUMN 25 (STANDARD COMMENT TAB STOP) OF THE SOURCE PRINTOUT PORTION. TABS OCCURING IN THE SOURCE PROGRAM ARE CONVERTED TO THE PROPER NUMBER OF BLANK CHARACTERS BY THE ASSEMBLER. THIS IS DONE BY THE ASSEMBLER RATHER THAN THE DEVICE HANDLER OR DEVICE BECAUSE THE BEGINNING OF THE SOURCE PRINTOUT DOES NOT OCCUR ON A STANDARD TAB STOP.

#12. 3. 0 ADDITIONAL ERROR MESSAGES FOR THE 2650.

STANDARD ERRORS:

- E: II ILLEGAL INDIRECT ADDRESSING SPECIFIED.
A LEADING ASTRISK WAS USED IN AN OPERAND FOR
AN INSTRUCTION NOT ALLOWING INDIRECT
ADDRESSING.
- E: IP ILLEGAL OFF PAGE REFERENCE.
AN OFF PAGE ADDRESS WAS USED IN A NON-BRANCH
INSTRUCTION.
- E: RA RELATIVE ADDRESS IS OUT OF RANGE.
THE OPERAND ADDRESS WAS OUT OF RANGE FROM
THE REQUIRED -64 TO +63 (DECIMAL) BYTES FROM
THE FIRST LOCATION FOLLOWING THE RELATIVE
INSTRUCTION.
- E: RV BAD REGISTER VALUE FIELD.
THE VALUE ASSIGNED TO A REGISTER (OR
CONDITION CODE) SPECIFICATION FIELD DID NOT
FALL WITHIN THE ALLOWABLE VALUES FOR THE
INSTRUCTION.

#12. 4. 0 BINARY OUTPUT FILE (PIFBUG COMPATABLE).

THE OBJECT (BINARY OUTPUT) FILE CONSISTS OF ASCII TEXT REPRESENTING HEXADECIMAL NUMBERS IN THE FOLLOWING FORMAT:

LEADER STRINGS OF 100 NULL CHARACTERS PRECEED AND FOLLOW THE OBJECT OUTPUT. EACH LINE BEGINS WITH A COLON AND IS FOLLOWED BY A FOUR HEX DIGIT STARTING ADDRESS, A TWO HEX DIGIT BYTE COUNT, A TWO HEX DIGIT BLOCK CONTROL BYTE, UP TO 16 BYTES OF DATA (EACH 2 HEX DIGITS), AND A FINAL TWO HEX DIGIT BLOCK CONTROL BYTE.

EXAMPLE:

```
: AAAACCBDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDBB
```

WHERE:

AAAA IS THE HEXADECIMAL ADDRESS FOR STORING THE FIRST DATA BYTE. EACH ADDITIONAL DATA BYTE IS TO BE STORED IN SEQUENTIAL ADDRESSES.

CC IS THE TWO HEXADECIMAL DIGIT COUNT FOR THE NUMBER OF DATA BYTES (REPRESENTED BY PAIRS OF D'S) IN THE LINE. A COUNT OF ZERO INDICATES THE TERMINATION OF THE OBJECT OUTPUT. (ON TERMINATION OF THE OBJECT OUTPUT A STARTING ADDRESS OF 0000 IS PROVIDED).

BB IS A TWO HEXADECIMAL DIGIT BLOCK CONTROL BYTE. TWO BLOCK CONTROL BYTES OCCUR IN EACH RECORD, ONE FOR THE ADDRESS AND BYTE COUNT AND ONE FOR THE DATA BYTES. THE BLOCK CONTROL BYTE IS FORMED BY EXCLUSIVE-ORING EACH BYTE TO BE CHECKED WITH THE BLOCK CONTROL BYTE AND THEN ROTATING THE RESULT ONE BIT LEFT TO FORM THE NEW BLOCK CONTROL BYTE. BEFORE STARTING TO CHECK EACH FIELD THE BLOCK CONTROL BYTE IS SET TO ZERO.

DD REPRESENTS TWO HEXADECIMAL DIGITS FOR A BYTE OF OBJECT (BINARY) CODE. UP TO 16 BYTES MAY BE OUTPUT ON ONE LINE.

THE BINARY OUTPUT FILE CAN BE CHANGED TO BNPF FORMAT THROUGH THE USE OF THE /B RUN-TIME OPTION. SECTION #2. 4. 0 DESCRIBES THE BNPF OUTPUT.

#12. 5. 0 MISCELLANEOUS NOTES.

THE FOLLOWING SECTIONS CLARIFY SEVERAL TOPICS PERTAINING TO THE IMPLEMENTATION OF THE X2650 CROSS-ASSEMBLER.

#12. 5. 1 2650 PAGE BOUNDARIES.

PROGRAM ASSEMBLY DOES NOT FLOW ACROSS THE 8K BYTE PAGE BOUNDARIES FOR THE 2650. NO ERROR IS GIVEN IF THE PAGE BOUNDARY IS EXCEEDED. INSTEAD THE ASSEMBLER WRAPS THE PROGRAM COUNTER AROUND TO THE BEGINNING OF THE CURRENT PAGE. TO ASSEMBLE CODE ON ANOTHER 2650 PAGE AN .ORG STATEMENT MUST BE USED TO ORIGIN THE PROGRAM COUNTER ONTO THE NEW PAGE.

#12. 5. 2 BXA/BSXA INDEX REGISTER.

THE BSA AND BSXA INSTRUCTIONS REQUIRE THAT REGISTER 3 BE USED FOR INDEXING. THE INDEX REGISTER DESIGNATION MAY BE LEFT OFF AS THE INDEXING WITH REGISTER 3 IS IMPLIED BY THE INSTRUCTION DEFINITION.

```

BXA (*)ADR      ; EITHER FORM IS LEGAL
BXA (*)ADR, 3

```

IF AN INDEX REGISTER VALUE IS USED, IT MUST NOT BE FORWARD REFERENCED OR AN E:RV ERROR WILL OCCUR ON PASS 1.

#12. 5. 3 REGISTER SPECIFICATION SYNTAX.

MOST INSTRUCTIONS REQUIRE A REGISTER OR CONDITION CODE VALUE TO IMMEDIATELY FOLLOW THE INSTRUCTION MNEMONIC. FOR THESE INSTRUCTIONS THE VALUE MUST IMMEDIATELY FOLLOW THE MNEMONIC, BEING SEPARATED FROM IT ONLY BY A COMMA. THE VALUE MAY BE AN EXPRESSION BUT IT MUST CONTAIN NO BLANKS OR TABS. THE VALUE MUST BE SEPARATED FROM THE MAIN OPERAND BY AT LEAST ONE BLANK OR TAB. THE MAIN OPERAND EXPRESSION MAY CONTAIN BLANKS AND/OR TABS.

```

ADDI, 1+2      1 + 2      ; LEGAL SYNTAX
ADDI, 1 + 2    1 + 2      ; ILLEGAL SYNTAX

```

#12. 5. 4 2650 ADDRESS SPACE.

THE BASIC ARITHMETIC PACKAGE OF THE X2650 CROSS-ASSEMBLER CARRYS VALUES OUT TO 16 BITS. THE 2650 CAN HOWEVER, HANDLE ONLY 15 BIT (32K) ADDRESS VALUES. ANY .ORG OR MICROPROCESSOR INSTRUCTION REQUIRING AN ADDRESS VALUE WILL MASK OUT THE 16TH BIT WITHOUT ISSUING AN ERROR MESSAGE.

#12. 5. 5 ADDRESS DATA.

THE 2650 INDIRECT ADDRESSING MODE REQUIRES THAT TWO BYTE ADDRESSES BE SET UP IN MEMORY. THE .DBYTE PSEUDO-OP IS CONVIENIENT FOR DEFINING THESE DOUBLE BYTE ADDRESSES. THE .ADDR PSEUDO-OP SHOULD NOT BE USED AS IT GENERATES A REVERSED BYTE FORMAT TYPICAL OF OTHER MICROPROCESSOR TYPES.


```

        . PAGE
829 3B 9   CLEAR  BSTR, U  POSITN ; POSITION THE POINTERS
82B 24 FF  EORI, 0  ^0377 ; COMPLEMENT
82D 4D E8 59 ANDA, 0  *ATABLE, 1 ; MASK OUT SELECTED BIT
830 CD E8 59  STRA, 0  *ATABLE, 1 ; STORE RESULT BACK
833 17      RETC, U
        ;
        ; ROUTINE TO POSITION THE TABLE BYTE POINTER AND
        ; DECODE THE BIT POSITION.
834 D2      POSITN  RRL, 2      ; MULTIPLY BY 16 FOR EACH TABLE
835 D2      RRL, 2
836 D2      RRL, 2
837 D2      RRL, 2
838 46 F0    ANDI, 2  0F0
83A 86 80    ADDI, 2  ^L TABLES ; COMPUTE BASE ADDRESS
83C CA 1C    STRR, 2  ATABLE+1 ; OF 16 BYTE TABLE
83E 77 8     PPSL     8      ; ADD WITH CARRY
840 20      EORZ     0      ; CLEAR REG 0
841 84 10    ADDI, 0  ^M TABLES
843 C8 14    STRR, 0  ATABLE
845 1       LODZ     1
846 44 7     ANDI, 0  7
848 C 68 51  LODA, 0  $1, 0 ; GET BIT PATTERN
84B 51      RRR, 1      ; FORM INDEX TO BIT TABLE
84C 51      RRR, 1
84D 51      RRR, 1
84E 45 F     ANDI, 1  0F
850 17      RETC, U
        . BIN
851 1       $1       . BYTE  1, 10, 100, 1000
852 2
853 4
854 8
855 10      . BYTE  10000, 100000, 1000000, 10000000
856 20
857 40
858 80
859 0 0     ATABLE   . DBYTE  0
***** E: MO
        JUNK      ; SAMPLE ERROR
        . END
    
```

```

859 ATABLE      20 BKP1      829 CLEAR      0 IPORT1
  1 IPORT2     800 LOOP      2 M          3 OPORT1
  1 P          834 POSITN    820 SET      1080 TABLES
  3 U
    
```

ERRORS: 1

#13. 0. 0 MICROPROCESSOR INSTRUCTIONS SET.

THIS SECTION IS A SUMMARY OF THE INSTRUCTION SET OF THE 2650 MICROPROCESSOR AS DEFINED BY SIGNETICS. THE ASSEMBLY CODE FORMAT FOR EACH INSTRUCTION IS SHOWN WITH THE HEXADECIMAL OBJECT CODE. EACH INSTRUCTION WILL BE CODED INTO THE DESIGNATED NUMBER OF BYTES.

THE FOLLOWING ABBREVIATIONS ARE USED IN DESCRIBING THE ASSEMBLY CODE FORMAT:

R REGISTER NUMBER (0 TO 3).
DATA .. AN EIGHT BIT DATA QUANTITY.
PORT .. AN EIGHT BIT PORT ADDRESS.
ADR ... AN ADDRESS VALUE.
(*). . . AN OPTIONAL ASTRISK INDICATING INDIRECT ADDRESSING.
S AN INDEX REGISTER NUMBER (0 TO 3).
(,S) .. AN OPTIONAL COMMAND SIGN (PLUS OR MINUS) INDICATING AUTO-INCREMENT OR AUTO-DECREMENT.
V A CONDITION CODE VALUE (0 TO 3).

LOAD/STORE INSTRUCTIONS:

INSTRUCTION		MEANING	HEX CODE	BYTES
LODZ	R	LOAD REGISTER ZERO	00+R	1*
LODI, R	DATA	LOAD IMMEDIATE DATA	04+R	2
LODR, R	(*)ADR	LOAD RELATIVE	08+R	2
LODA, R	(*)ADR	LOAD ABSOLUTE	0C+R	3
LODA, 0	(*)ADR, X(, S)	LOAD ABSOLUTE, INDEXED	0C+X	3
STRZ	R	STORE REGISTER ZERO	CO+R	1*
STRR, R	(*)ADR	STORE RELATIVE	C8+R	2
STRA, R	(*)ADR	STORE ABSOLUTE	CC+R	3
STRA, 0	(*)ADR, X(, S)	STORE ABSOLUTE, INDEXED	CC+X	3

* LODZ 0 CODES INTO 60.
 STRZ 0 IS AN ILLEGAL INSTRUCTION.

ARITHMETIC INSTRUCTIONS:

INSTRUCTION		MEANING	HEX CODE	BYTES
ADDZ	R	ADD TO REGISTER ZERO	80+R	1
ADDI, R	DATA	ADD IMMEDIATE DATA	84+R	2
ADDR, R	(*)ADR	ADD RELATIVE	88+R	2
ADDA, R	(*)ADR	ADD ABSOLUTE	8C+R	3
ADDA, 0	(*)ADR, X(, S)	ADD ABSOLUTE, INDEXED	8C+X	3
SUBZ	R	SUBTRACT FROM REGISTER ZERO	A0+R	1
SUBI, R	DATA	SUBTRACT IMMEDIATE DATA	A4+R	2
SUBR, R	(*)ADR	SUBTRACT RELATIVE	A8+R	2
SUBA, R	(*)ADR	SUBTRACT ABSOLUTE	AC+R	3
SUBA, 0	(*)ADR, X(, S)	SUBTRACT ABSOLUTE, INDEXED	AC+X	3

LOGICAL INSTRUCTIONS:

INSTRUCTION		MEANING	HEX CODE	BYTES
ANDZ	R	AND TO REGISTER ZERO	40+R	1*
ANDI, R	DATA	AND IMMEDIATE DATA	44+R	2
ANDR, R	(*)ADR	AND RELATIVE	48+R	2
ANDA, R	(*)ADR	AND ABSOLUTE	4C+R	3
ANDA, 0	(*)ADR, X(, S)	AND ABSOLUTE, INDEXED	4C+X	3
IORZ	R	INCLUSIVE OR TO REGISTER ZERO	60+R	1
IORI, R	DATA	INCLUSIVE OR IMMEDIATE DATA	64+R	2
IORR, R	(*)ADR	INCLUSIVE OR RELATIVE	68+R	2
IORA, R	(*)ADR	INCLUSIVE OR ABSOLUTE	6C+R	3
IORA, 0	(*)ADR, X(, S)	INCLUSIVE OR ABSOLUTE, INDEXED	6C+X	3
EORZ	R	EXCLUSIVE OR TO REGISTER ZERO	20+R	1
EORI, R	DATA	EXCLUSIVE OR IMMEDIATE DATA	24+R	2
EORR, R	(*)ADR	EXCLUSIVE OR RELATIVE	28+R	2
EORA, R	(*)ADR	EXCLUSIVE OR ABSOLUTE	2C+R	3
EORA, 0	(*)ADR, X(, S)	EXCLUSIVE OR ABSOLUTE, INDEXED	2C+X	3

* ANDZ 0 IS AN ILLEGAL INSTRUCTION.

COMPARE INSTRUCTIONS:

INSTRUCTION		MEANING	HEX CODE	BYTES
COMZ	R	COMPARE TO REGISTER ZERO	E0+R	1
COMI, R	DATA	COMPARE IMMEDIATE DATA	E4+R	2
COMR, R	(*)ADR	COMPARE RELATIVE	E8+R	2
COMA, R	(*)ADR	COMPARE ABSOLUTE	EC+R	3
COMA, 0	(*)ADR, X(, S)	COMPARE ABSOLUTE, INDEXED	EC+X	3

ROTATE INSTRUCTIONS:

INSTRUCTION		MEANING	HEX CODE	BYTES
RRR, R		ROTATE REGISTER RIGHT	50+R	1
RRL, R		ROTATE REGISTER LEFT	D0+R	1

BRANCH INSTRUCTIONS:

INSTRUCTION	MEANING	HEX CODE	BYTES
BCTR, V (*)ADR	BRANCH ON CONDITION TRUE, REL	18+V	2
BCFR, V (*)ADR	BRANCH ON CONDITION FALSE, REL	98+V	2*
BCTA, V (*)ADR	BRANCH ON CONDITION TRUE, ABS	1C+V	3
BCFA, V (*)ADR	BRANCH ON CONDITION FALSE, ABS	9C+V	3*
BRNR, R (*)ADR	BRANCH ON REG NON-ZERO, REL	58+R	2
BRNA, R (*)ADR	BRANCH ON REG NON-ZERO, ABS	5C+R	3
BIRR, R (*)ADR	BRANCH ON INCREMENTING REG, REL	D8+R	2
BIRA, R (*)ADR	BRANCH ON INCREMENTING REG, ABS	DC+R	3
BDRR, R (*)ADR	BRANCH ON DECREMENTING REG, REL	F8+R	2
BDRA, R (*)ADR	BRANCH ON DECREMENTING REG, ABS	FC+R	3
BXA (*)ADR, 3	BRANCH INDEXED, ABSOLUTE	9F	3
ZBRR (*)ADR	ZERO BRANCH RELATIVE	9B	2

* BCFR AND BCFA CANNOT USE V=3.

SUBROUTINE BRANCH/RETURN INSTRUCTIONS:

INSTRUCTION	MEANING	HEX CODE	BYTES
BSTR, V (*)ADR	BRANCH TO SUBROUTINE ON CONDITION TRUE, RELATIVE.	38+V	2
BSFR, V (*)ADR	BRANCH TO SUBROUTINE ON CONDITION FALSE, RELATIVE.	B8+V	2*
BSTA, V (*)ADR	BRANCH TO SUBROUTINE ON CONDITION TRUE, ABSOLUTE.	3C+V	3
BSFA, V (*)ADR	BRANCH TO SUBROUTINE ON CONDITION FALSE, ABSOLUTE.	BC+V	3*
BSNR, R (*)ADR	BRANCH TO SUBROUTINE ON NON-ZERO REGISTER, RELATIVE.	78+R	2
BSNA, R (*)ADR	BRANCH TO SUBROUTINE ON NON-ZERO REGISTER, ABSOLUTE.	7C+R	3
BSXA (*)ADR, 3	BRANCH TO SUBROUTINE, INDEXED.	BF	3
RETC, V	RETURN FROM SUBROUTINE, COND.	14+V	1
RETE, V	RETURN FROM SUBROUTINE AND ENABLE INTERRUPT, CONDITIONAL.	34+V	1
ZBSR (*)ADR	ZERO BRANCH TO SUBROUTINE, RELATIVE.	BB	2

* BSFR AND BSFA CANNOT USE V=3.

PROGRAM STATUS INSTRUCTIONS:

INSTRUCTION		MEANING	HEX CODE	BYTES
LPSU		LOAD PROGRAM STATUS, UPPER.	92	1
LPSL		LOAD PROGRAM STATUS, LOWER.	93	1
SPSU		STORE PROGRAM STATUS, UPPER.	12	1
SPSL		STORE PROGRAM STATUS, LOWER.	13	1
CPSU	DATA	CLEAR PROG STATUS, UPPER, SELECTIVE.	74	2
CPSL	DATA	CLEAR PROGRAM STATUS, LOWER, SELECTIVE.	75	2
PPSU	DATA	PRESET PROGRAM STATUS, UPPER SELECTIVE.	76	2
PPSL	DATA	PRESET PROGRAM STATUS, LOWER, SELECTIVE.	77	2
TPSU		TEST PROGRAM STATUS, UPPER, SELECTIVE.	B4	2
TPSL	DATA	TEST PROGRAM STATUS, LOWER, SELECTIVE.	B5	2

INPUT/OUTPUT INSTRUCTIONS:

INSTRUCTION		MEANING	HEX CODE	BYTES
WRD, R		WRITE DATA	F0+R	1
REDD, R		READ DATA	70+R	1
WRTC, R		WRITE CONTROL	B0+R	1
REDC, R		READ CONTROL	30+R	1
WRTE, R	PORT	WRITE EXTENDED	D4+R	2
REDE, R	PORT	READ EXTENDED	54+R	2

MISCELLANEOUS INSTRUCTIONS:

INSTRUCTION		MEANING	HEX CODE	BYTES
HALT		HALT, ENTER WAIT STATE	40	1
DAR, R		DECIMAL ADJUST REGISTER	94+R	1
TMI, R	DATA	TEST UNDER MASK IMMEDIATE	F4+R	2
NOP		NO OPERATION	C0	1

APPENDIX A - RUN-TIME OPTIONS.

#14. 0. 0

/B - OUTPUT BINARY FILE IN BNPF FORMAT.
 /E - INHIBIT ERROR MESSAGES TO CONSOLE.
 /H - INHIBIT HEADINGS AND PAGINATION.
 /J - LIST UNASSEMBLED STATEMENTS AND CONDITIONAL
 ASSEMBLY PSEUDO-OPS.
 /K - EXPAND SYMBOL TABLE STORAGE INTO ADDITIONAL
 CORE.
 /L - OUTPUT LEADER (NULLS) IN BINARY FILE FOR EACH
 .ORG STATEMENT.
 /N - LIST ONLY THE SYMBOL TABLE.
 /O - OUTPUT LISTING IN OCTAL FORMAT INSTEAD OF IN
 HEXADECIMAL.
 /P - INCLUDE NORMALLY UNLISTED PSEUDO-OPS IN THE
 LISTING.
 /S - OMIT THE SYMBOL TABLE FROM THE LISTING.
 /T - REPLACE THE FORM/FEEED WITH 3 CR/LF'S.
 /W - INHIBIT WARNING MESSAGES.
 /O TO /9 - USER FLAGS, USED WITH THE ? OPERATOR.

APPENDIX B - INDICATOR SET.

* - MULTIPLICATION.
 / - DIVISION.
 & - BOOLEAN AND.
 ! - INCLUSIVE OR.
 + - ADDITION.
 - - SUBTRACTION.
 ^C - COMPLEMENT INDICATOR, (UPARROW B).
 ^B - BINARY RADIX INDICATOR, (UPARROW B).
 ^D - DECIMAL RADIX INDICATOR, (UPARROW D).
 ^H - HEXADECIMAL RADIX INDICATOR, (UPARROW H).
 ^O - OCTAL RADIX INDICATOR, (UPARROW O).
 ^L - LEAST SIGNIFICANT BYTE ACCESS OPERATOR,
 (UPARROW L).
 ^M - MOST SIGNIFICANT BYTE ACCESS OPERATOR,
 (UPARROW M).
 ; - COMMENT INDICATOR.
 " OR ' - ASCII CHARACTER INDICATOR.
 ? - USER FLAG OPERATOR.
 . - CURRENT LOCATION COUNTER, (PERIOD).

APPENDIX D - ERROR MESSAGES.

#14. 0. 0

E: BN - BAD NESTING OF BRACKETS.
E: DF - OUTPUT DEVICE FULL. (FATAL)
E: DR - DIGIT OUTSIDE OF RADIX.
E: II - ILLEGAL INDIRECT ADDRESSING SPECIFIED.
E: IL - ILLEGAL LABEL FIELD.
E: IP - ILLEGAL OFF PAGE REFERENCE.
E: IO - ILLEGAL OPERAND VALUE.
E: LO - LINE INPUT OVERFLOW.
E: LS - LOCAL SYMBOL SYNTAX ERROR.
E: LT - LOCAL SYMBOL TABLE OVERFLOW. (FATAL)
E: ML - MULTIPLE LABEL DEFINITION.
E: MO - MISSING OR ILLEGAL MNEMONIC IN OPERATOR FIELD.
E: OC - OPERAND TOO COMPLEX.
E: OE - OPEN ERROR IN OUTPUT FILE. (FATAL)
E: OM - OPERAND MISSING.
E: OS - OPERAND SYNTAX ERROR.
E: PE - PHASE ERROR, ADDRESS CONFLICT. (FATAL)
E: PS - ILLEGAL PERMANENT SYMBOL USAGE IN OPERAND.
E: RA - RELATIVE ADDRESS IS OUT OF RANGE.
E: RE - INPUT FILE READ ERROR. (FATAL)
E: RV - BAD REGISTER VALUE FIELD.
E: ST - SYMBOL TABLE OVERFLOW. (FATAL)
E: TL - LABEL DEFINED TOO LATE.
E: US - UNDEFINED SYMBOL.
E: WE - OUTPUT FILE WRITE ERROR. (FATAL)

W: EF - NO .END STATEMENT IN LAST FILE.
W: UC - UNINHIBITED CONDITIONAL ASSEMBLY IN EFFECT
AT ASSEMBLY END.

. ADDR - DOUBLE BYTE DATA STORAGE, REVERSED FORMAT.
. BIN - CHANGES DEFAULT RADIX TO BINARY.
. BYTE - SINGLE BYTE DATA STORAGE.
. DBYTE - DOUBLE BYTE DATA STORAGE.
. DECM - CHANGES DEFAULT RADIX TO DECIMAL.
. DINST - RENAMES A MICROPROCESSOR INSTRUCTION.
. END - PROGRAM TERMINATOR.
. ENDC - ENDS CONDITIONAL ASSEMBLY.
. EQU - ASSIGNS A PERMANENT VALUE TO A SYMBOL.
. HEX - CHANGES DEFAULT RADIX TO HEXADECIMAL.
. IFDEF - INCLUDE CODE TO .ENDC IF SYMBOL IS DEFINED.
. IFNDEF - INCLUDE CODE TO .ENDC IF SYMBOL IS NOT DEFINED.
. IFNZRO - INCLUDE CODE TO .ENDC IF OPERAND DOES NOT EQUAL 0.
. IFZERO - INCLUDE CODE TO .ENDC IF OPERAND EQUALS 0.
. LIST - PROVIDES SELECTIVE LISTINGS.
. OCT - CHANGES DEFAULT RADIX TO OCTAL.
. ORG - REASSIGNS THE CURRENT LOCATION COUNTER.
. PAGE - BEGINS NEW PAGE IN LISTING.
. SET - ASSIGNS A TEMPORARY VALUE TO A SYMBOL.
. TITLE - SPECIFIES HEADING.
. ZERO - ZEROS A SPECIFIED NUMBER OF BYTES.

