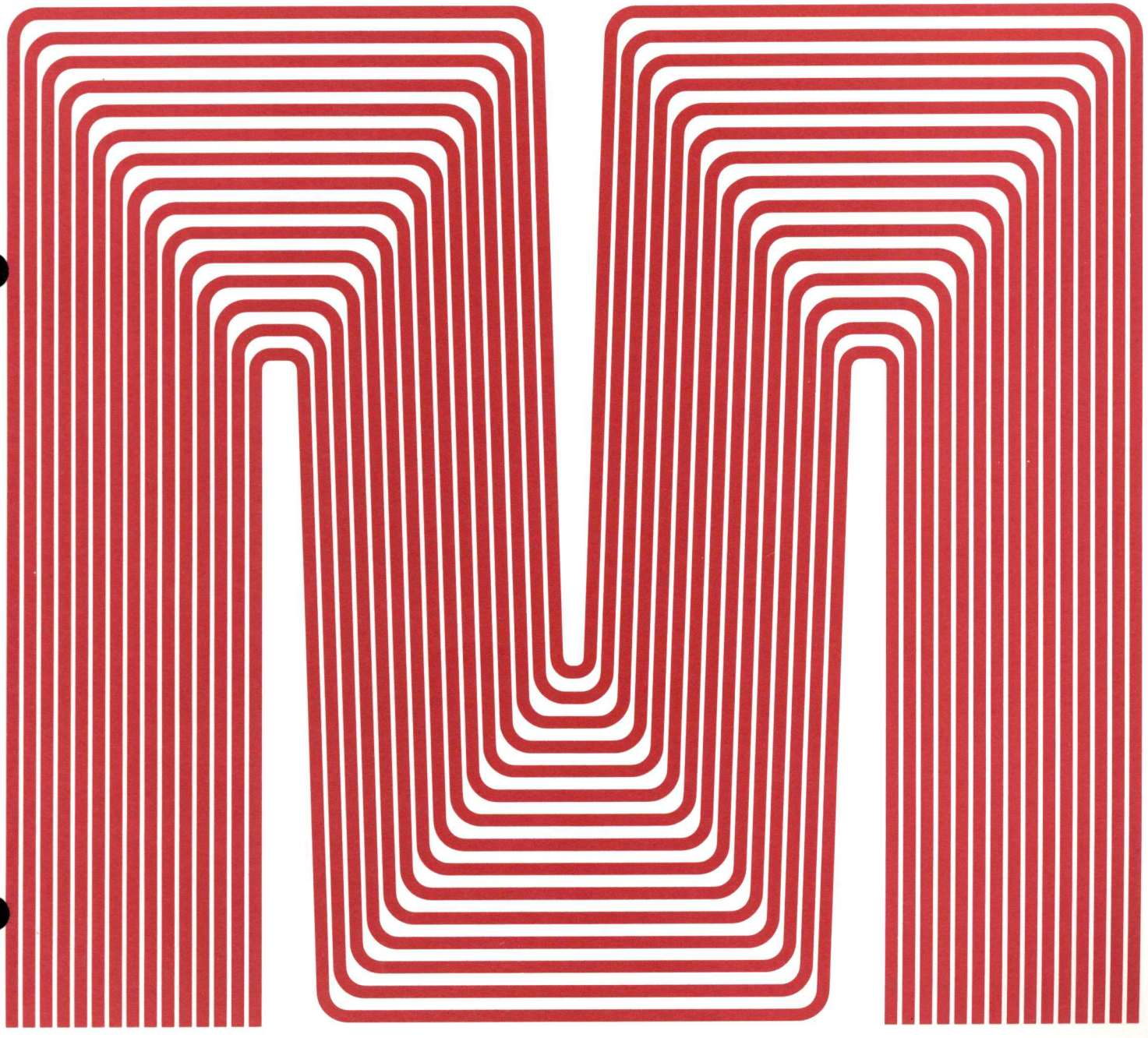
MicroSystems Inc.

Micro 800 Computer

AP800 Assembly Program



MICRO 800 COMPUTER

AP800 ASSEMBLY PROGRAM

69-1-0800-002

July 1969

Micro Systems Inc
644 East Young Street
Santa Ana, California 92705

TABLE OF CONTENTS

ILLUSTRATIONS

1. INTRODUCTION

AP800 is a symbolic assembly program for the MICRO 800 computer. The assembler provides for symbolic addressing and mnemonics for machine and assembler instructions. This program is written in FORTRAN IV and may be adapted to many computer systems. The MICRO 800 source program is entered by punch cards and the output of the assembler includes an assembly listing, read only storage diode map, and an object program card deck.

The AP800 assembly language includes the following features:

Address Arithmetic	–	Decimal and hexadecimal numbers, symbolic addresses, and arithmetic expressions.
Listing Control	–	The format of the listing which may be controlled with comment cards included.
Diagnostics	–	Diagnostics for source program errors included in the output listing.
Option Flags	–	Single letter flags to signify options to micro-commands.

2. SOURCE LANGUAGE

The source language is a sequence of symbolic instructions, called statements, which are punched on cards. Each statement is punched on a single card. Each statement may consist of from one to four entries: a name field, an operation field, an operand field, and a comment field. Columns 73-80 are normally used for identification or sequence numbers. Entries in the operand field are expressions which may consist of decimal numbers, hexadecimal numbers, and symbolic values.

STATEMENT FORMAT

Name Field

The name field entry is a symbol composed of from one to six characters starting with column 1 and terminating with the first blank. The first character of a symbol is alphabetic or period; subsequent characters may be alphabetic, numeric or a period. A name entry is usually optional and the type of instruction determines the legal content of the name field. The symbol takes on the current value of the assembler's location counter unless assigned another value by an assembler instruction. When an asterisk (*) appears in column 1 the remainder of the line is considered as comment and is not processed by the assembler except to place it on the listing.

Operation Field

The operation field entry is a mnemonic operation code specifying the machine or assembler instruction. The field begins in column 8 and is terminated by the first blank. Certain memory referencing instruction modes use special symbols suffixed to the mnemonic.

Operand Field

The operand field entries identify and describe data to be acted upon by instructions as, for example, memory locations, or literals. One or more operands may be written, depending on the needs of the instruction. Entries are separated by commas, and no blanks may appear in the field. The operand field starts in column 14. It is terminated by the first blank column.

Comments Field

Comments describing the information about the program may be inserted between the end of the operand field and column 72. All characters, including spaces, may be used in writing a comment.

OPERAND FIELD EXPRESSIONS

Expressions in the operand field are made up of one or more terms which are connected by + and – arithmetic operators. No parenthetical expressions are allowed. Each term of the

expression represents a value. Values may be assigned by the assembler program (symbols), or there may be inherent in the term itself (constants). The range of values depends on the operand and the instruction. Address expressions for relative type addressing are written as if they are not relative. The assembler will convert these expressions to a relative displacement.

Symbols

A symbol is composed of one to six characters. The first character must be alphabetic or period; subsequent characters may be numeric, alphabetic, or period. Imbedded blanks are not allowed and the assembler stops scanning the symbol with the first character which is not alphanumeric or a period. All symbols, except the special symbols * and **, used in an operand field, must be defined by a single appearance in the name field of statement within the program.

Special Symbols

The special symbol * represents the momentary value of the assembler's location counter. It may be used as any other symbol in an expression but must never appear in the name field. When used in the operand field of a multi-byte instruction it will assume the value of the address of the first byte of the instruction.

Constants

The values of the constant terms are not assigned by the assembler program but are inherent in the terms. There are two types of constant terms: decimal and hexadecimal.

a. Decimal Constant

   A decimal constant is an unsigned decimal number. The value must be less than 65,536.

b. Hexadecimal Constant

   A hexadecimal constant is an unsigned hexadecimal number of up to four characters written as a sequence of hexadecimal digits. The digits are enclosed in single quotation marks and preceded by the letter X. Each hexadecimal digit represents a four-bit binary number. The characters A through F are used to identify the hexadecimal integers 10 through 15.

3. MACHINE COMMANDS

Machine commands are expressed by a one or two character mnemonic code in the operation field. The required operands depend on the command type. The four syntax types are described below. Examples of the method of writing machine commands in the assembly language are shown in the sample listing in section 5.

Load Register Commands (Command 1)

All commands of this syntax type have two character mnemonics beginning with L, except for the Jump Command (JP). The second character is the register identifier character. The operand field of all commands of this type except Jump must contain a single operand which is an expression, whose value is less than 1024 and greater or equal to -256. It is evaluated modulo 256. The Jump command must contain an operand expression which has a positive value less than 1024.

Literal-File Commands

The commands of this syntax group (commands 2-6), have two character mnemonics and require two operands. The first operand is an expression which designates a file register (f) and must be in the range 0-15. The second operand (n) is an expression which must be less than 1024 and greater than or equal to -256. It is evaluted modulo 256.

Execute and Control Commands

The commands of this syntax group have operation code mnemonics identical to those of the next group, and require two operands. The first operand is an expression which designates a file register (f) and must be in the range 0-15. The second operand (c) is an expression which designates the option bits (7-4) and must be in the range 0-15.

Operate Class Commands other than Execute and Control

The commands of this syntax group have basic operation code mnemonics which are a single character. If the result of the operation is to be routed to a machine register the designator of that register is appended as a second character of the mnemonic. If the result is **not** to be placed in the designated file register, an * is appended to the mnemonic. The second character register designators are given in the table shown on the next page.

Register Code	Register Designator	Register
0		None
1	T	T Register
2	M	M Register
3	N	N Register
4	L	L Register: addresses 000-0FF and 200-2FF
5	K	L Register: addresses 100-1FF and 300-3FF
6	U	U Register
7	S	OR U Register into bits 15-8 of command (except Control commands)

The first operand is an expression which designates a file register (f) and must be in the range 0-15. Other operands are optional and if included, each consists of a single character which designates an option for the command. The allowable option flags for each command are given on the table on the next page and the meaning of each is given below. The order of writing the options is immaterial.

L	– Link Control
I	– Add one or insert one on shift
D	– Decrement one
T	– T register for operand
F	– Complement of T register for operand
H	– Half Cycle memory operation (otherwise full cycle)
R	– Right shift (otherwise left shift)
C	– Set condition flags

MICROCOMMANDS

Command	Mnemonic	Operand Field
Load T	LT	n
Load M	LM	n
Load N	LN	n
Load U	LU	n
Load Zero Control	LZ	n
Load Seven Control	LS	n
Jump	JP	n
Load File	LF	f,n
Add to File	AF	f,n
Test If Zero	TZ	f,n
Test If Not Zero	TN	f,n
Compare	CP	f,n
Execute	Er*	f,c
Control	Kr*	f,c
Add	Ar*	f,L,I,T,C
Increment	Ir*	f,L,C
Subtract	Sr*	f,L,D,T,C
Decrement	Dr*	f,L,C
Copy	Cr*	f,L,I,T,C
Read	Rr*	f,L,I,D,H
Write	Wr*	f,L,I,D,H
Logical OR	Or*	f,L,F,T,C
Move	Mr*	f,L,C
Exclusive-OR	Xr*	f,L,F,T,C
Logical AND	Nr*	f,L,F,T,C
Shift	Hr*	f,L,I,R,C

4. ASSEMBLER INSTRUCTIONS

Seven assembler instructions are included for control of the assembly process and the output listing.

ORG – Set Location Counter

The ORG assembler instruction alters the setting of the location counter. The name field entry, if any, will be assigned the value of the program counter after it is altered. The operand field of ORG must contain an expression whose value will be placed in the location counter. All symbols in the expression must have been previously defined when the instruction is first encountered. The next instruction which places object code in the program is forced to begin a new object card.

EQU – Equate Symbol

The EQU assembler instruction is used to define a symbol by assigning to it the value of the operand field. Any symbols appearing in the expression must have been previously defined when the instruction is first encountered. A name field entry must be present.

DC – Define Constant

The DC assembler instruction is used to provide constant data in memory. Each statement specified only one constant. The constant is written as an expression and is assembled as a 16-bit word in storage.

IDENT – Program Identification

The IDENT assembler instruction is used to identify the start of a program and to supply the program name which is located in the operand field. The IDENT must be the first statement in a source program.

END – End Assembly

The END assembler instruction terminates the assembly of a program and must be the last statement in a source program.

SPACE – Space Listing

The SPACE assembler instruction causes one or more blank lines to be inserted into the listing. The name field is disregarded by the assembler. The operand field contains an expression specifying the number of blank lines. If the spacing is beyond the end of the current page, the listing begins at the top of the next page.

EJECT – Start New Listing Page

The EJECT instruction causes the next line of the listing to appear at the top of the next page. The name and operand fields are disregarded by the assembler.

5. ASSEMBLY LISTING AND DIODE MAP

The output listing from AP800 contains the memory address, and contents of words in the object program. The source statement is printed side-by-side with the object code.

FORMAT

Printer Columns		Contents
8 – 11	–	Error flags
15 – 17	–	Storage address
21 – 24	–	Storage contents
31 – 110	–	Source statement

ERROR FLAGS

A – **Address Error**

This error occurs when an address expression in the operand field is incorrectly written or the value is out of range for one of the operands. An error flag will occur for each operand in error or out of range.

F – **Flag Error**

This error occurs when an operate class command has an option flag in the operand field which is not allowed for the command or is unrecognizable.

M – **Multidefined Symbol Error**

This error occurs when the symbol in the name field has been previously defined by appearing in the name field of another instruction.

N – **Name Field Error**

This error flag occurs when the symbol in the name field starts with a character other than alphabetic or period, or contains a non alphanumeric or non period character.

O – **Operation Mnemonic Error**

This error occurs when the assembler does not recognize the contents of the operation field starting in column 8. A zero value is assembled to allow patching.

U – **Undefined Symbol Error**

This error occurs when the symbol encountered in an expression of the operand field is not defined by an appearance in the name field.

SAMPLE LISTING

The sample listing on the next page shows the format of the listing and provides examples of how to write each instruction type, literals, constants, and assembler instructions. The six types of error conditions are also illustrated.

DIODE MAP

The read only storage diode map is printed if the control card following the END card contains a 1, 2, or 3 in column 1. The digit specifies the number of diode maps to be printed. The diode map for each 256 word read only storage board is placed on three pages of the assembly listing. The format of the map is the same as the physical layout of the ROS board. An X on the map indicates a 1-bit and that a diode is to be placed at the position of the X, while an 0 indicates a 0-bit and no diode.

Each of the 64 lines of the diode map for a board contains the diodes for four words. The address of the first word is printed at the left of the map. The four words are interleaved so that the same bit position in each of the four words are grouped together and printed as a cluster at four diode positions. The 16 bit positions are printed across the page and the sum of the number of diodes on the line is placed at the right of the map.

```
                                IDENT SAMPLE
                        * THIS SAMPLE PROGRAM SHOWS HOW TO WRITE VARIOUS COMMANDS.

                        ** LOAD REGISTER COMMANDS
      000   1112        START   LT      X#12#           LOAD T  -  HEXADECIMAL LITERAL
      001   1204                LM      4               LOAD M  -  DECIMAL LITERAL
      002   1308                LN      ALPHA+2         LOAD M  -  EXPRESSION LITERAL
      003   16AA                LU      X#AA#           LOAD U
N     004   160A        123456  LU      X#A#            SYMBOL IN NAME FIELD IS ILLEGAL
      005   1780                LS      X#80#           LOAD SEVEN CONTROL  -  HALT
                        ** JUMP COMMANDS
      006   1408        ALPHA   JP      *+2             JUMP IN PAGE ZERO
U     007   1400                JP      SAM             SYMBOL UNDEFINED
                                ORG     256             ORG ASSEMBLER INSTRUCTION - PAGE 1
      100   1502        PAGE1   JP      *+2             JUMP IN PAGE 1
      101   1C02                JP      PAGE2+2         JUMP TO PAGE 2
      102   1D02                JP      PAGE3+2         JUMP TO PAGE 3
0     103   0000                PP      2,2             OPERATION MNEMONIC IS ILLEGAL
                        **  FILE LITERAL COMMANDS
      104   2AFF                LF      10,X#FF#        LOAD FILE  -  HEXADECIMAL LITERAL
A     105   2200                LF      2               ERROR IN OPERAND FIELD
      106   2A02                LF      TEN,2           LOAD FILE  -  DECIMAL LITERAL
      107   3202                AF      2,2             ADD TO FILE
      108   4004                TZ      0,4             TEST IF ZERO
      109   5A0C                TN      TEN,X#C#        TEST IF NOT ZERO
      10A   65FE                CP      5,-2            COMPARE  -  NEGATIVE OPERAND OK
                        ** OPERATE COMMANDS WITH LEGAL OPTION FLAGS
      10B   0250        OPER    E       2,5             EXECUTE
      10C   7580                K       5,8             CONTROL
      10D   82F0                A       2,L,I,T,C       ADD  -  LINK,INCR,I REG, COND FLAG
      10E   8200                I       2,L,C           INCREMENT  -  FORM OF ADD
      10F   92F0                S       2,L,D,T,C       SUBTRACT  -  LINK,DECR,T REG,COND FLAG
      110   9200                D       2,L,C           DECREMENT  -  FORM OF SUBTRACT
      111   B2F0                C       2,L,I,T,C       COPY
      112   A260                R       2,L,D,H         READ MEMORY  -  LINK
      113   A260                R       2,L,I,D,H       READ MEMORY  -  L OR I OR D, HALF
      114   A270                W       2,L,I,D,H       WRITE MEMORY  -  L OR I OR D, HALF
      115   C2F0                O       2,L,F,T,C       OR  -  LINK,COMP T REG, TRUE T REG, COND
      116   C2F0                M       2,L,C           MOVE  -  LINK,COND FLAG
      117   D2F0                X       2,L,F,T,C       EXCLUSIVE-OR
      118   E2F0                N       2,L,F,T,C       AND
      119   F2F0                H       2,L,I,R,C       SHIFT  -  LINK,ONE,RIGHT,COND FLAG
                        ** VARIATIONS OF OPERATE COMMANDS
      11A   CA01                MT      TEN             MOVE FILE REG 10 TO T
      11B   854B                IN*     5               *  PREVENTS RESULT FROM GOING TO FILE
      11C   C518                M*      5,C             FILE 5 IS TESTED AND COND FLAGS SET
      11D   8069                CI*     0,T,I           THIS COMMAND INCREMENTS THE T REG
      11E   8543                IN      5               INCREMENT FILE REG 5 AND PLACE IN N REG
      11F   8A23                AN      TEN,T           FILE DESIGNATOR MAY BE EXPRESSION
      120   9B4B                DN*     11              FILE 11 MINUS ONE IS PLACED IN N REG
      121   8224                AL      2,T             JUMP IN PAGE 0 OR 2
      122   8245                IK      2               JUMP IN PAGE 1 OR 3
F     123   8541                IT      5,X             ILLEGAL FLAG
U     124   0500                E       5,L             NO FLAGS ON EXECUTE OR CONTROL
            000A        TEN     EQU     10
                                ORG     X#200#          ORG FOR PAGE 2
      200   10AB        PAGE2   DC      X#10AB#         COMMAND MADE BY CONSTANT
                                ORG     X#300#          ORG FOR PAGE 3
      300   150B        PAGE3   JP      OPER
                                END                     LAST CARD
```

FIGURE 1. SAMPLE LISTING

6. OBJECT PROGRAM CARD DECK

The AP800 assembly program generates a deck of cards which contain the binary object code. All information punched on the cards is in Hollerith code, with a single hexadecimal digit (four binary bits) punch in each column. This format allows easy visual reading of the cards after they are interpreted and permits rapid patching or generation of patches to the deck. Each card contains 16 program words. If all 16 words are zero, the card is not punched.

The cards have two fields as follows:

Columns 1-4	–	Load address
Columns 5-68	–	Object code, four columns per word

COMMENT AND EVALUATION SHEET
MICRO 800 COMPUTER
AP800 Assembly Program

Pub. No. 69–1–0800–002 July 1969

YOUR EVALUATION OF THIS MANUAL WILL BE WELCOMED BY MICRO SYSTEMS INC. ANY ERRORS, SUGGESTED ADDITIONS OR DELETIONS OR GENERAL COMMENTS MAY BE MADE BELOW. PLEASE INCLUDE PAGE NUMBER REFERENCE.

FROM NAME: _____

BUSINESS
ADDRESS: _____

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

FOLD ON DOTTED LINES AND STAPLE

STAPLE STAPLE

FOLD FOLD

First Class
Permit No. 1972
Santa Ana
California 92711

BUSINESS REPLY MAIL
NO POSTAGE NECESSARY IF MAILED IN THE U.S.A.

Postage Will Be Paid By

MICRO SYSTEMS INC.
644 East Young Street
Santa Ana, California 92705

FOLD FOLD

STAPLE STAPLE

MicroSystems Inc.

644 East Young Street
Santa Ana, California 92705
Telephone: (714) 540-6730
TWX: 910-595-1764