PolyMorphic Assembler System – 8K

OPERATING SYSTEM

The PolyMorphic Assembler System is a software package designed to run on the Poly-88 computer. Included in the system is an executive to handle memory and tape files, an Assembler, and a line oriented Editor. To use the system a minimum of 8K of memory should be available.

EXECUTIVE COMMANDS

control X	Cancel input line
EXEC	Execute a program
ASMB	Assemble a source file to object code
LIST	List current file
DELT	Delete line or lines of current file
####	Any four numeric digits enters editor
DISP	Display memory
RSEQ	Resequence current file
MNTR	Go to monitor
LOAD	Load source file
SAVE	Dump source file
DUMP	Dump memory to cassette
FREE	Display free memory space

To initialize the system, execute it at 2000H. To restart the system without initializing it, execute at 2003H.

The executive has one error message 'WHAT?' indicating an improper command or an error on parameters following the command. The other error that can happen is "CHECKSUM ERROR". This error can occur when a cassette error occurs.

EXEC ####
This command is used to execute a program at address ####.

LIST ####
This command is used to display the lines entered by the user into the file. The output consists of the lines in the file starting at line #### and the next 13 lines. If #### is not specified, then the next 13 lines are displayed. (It is recommended that you enter a zero if you want to display the start of the file.)

DELT #### ####
This command is used to delete lines entered by the user from the file. All lines starting at the first line and continuing up to and including the second number are deleted from the "current" file. If the second number is not specified then only the first is deleted.

LOADP /NAME/
LOADB /NAME/
Load the file with specified name from cassette. LOADP reads Polyphase tapes, LOADB reads byte format tapes. The names must be enclosed in delimeters (/) as shown, and be 3 characters or less. The syntax must be exactly as shown, with one space separating the B or P and the file delimiting /.

SAVEP /NAME/ LIN1 LIN2
SAVEB /NAME/ LIN1 LIN2
Save source text inclusive from line number specified as LIN1 to and including the line number specified as LIN2 as the file name given, in byte or Polyphase mode. LIN1 and LIN2 must be given as four digits, with one space between the ending file name delimiter / and the first line number, and one space between the line numbers.
Examples:

```
>SAVEP /MUNG/ 0010 0450
>SAVEB /CHAIN/ 0100 0300
```

DUMPP /NAME/ ADR1 ADR2
DUMPB /NAME/ ADR1 ADR2
Dump contents of memory in the address range specified by ADR1 and ARD2 to the named file in byte or Polyphase mode. ADR1 and ADR2 must be specified as four digits and must be separated by one space.
Examples:

```
>DUMPP /ASM/ 2000 2FFF
>DUMPB /ASM/ 2000 2FFF
```

FREE
This command prints the amount (in hexadecimal) of storage left.

ASMB (E) (S) #### ####
This command is used to assemble a source program located in the file area. The assembler performs the assembly, assigning addresses to the object code starting at the first number. On assembly the object code is placed in memory starting at location number 2. If number 2 is not specified, it is assumed to be the same value as number 1. During pass one errors detected will be displayed, and during pass two a complete listing is produced. If the option "E" is specified in the command, only those lines which contain errors are listed. If the optional "S" is specified, then the new symbol table will be added onto the previous symbol table. Otherwise it is started at the beginning of the symbol table space.
NOTE: If "E" and "S" are both specified "e" must be first or else it will be ignored.

TEXT EDITOR

The editor is a line oriented editor which enables the user to easily create program files in the System. Each line is prefaced by a fixed line number which provides for stable line referencing. Since line numbers can range from 0000 to 9999 (decimal)

there are 10,000 lines that can exist in each file. (if enough storage exists). As the user types lines on the keyboard, they are entered into the file area. The editor places all line numbers in sequence and automatically over-writes an existing line in the file if a new line with the same line number is entered by the user.

The editor does not automatically assign line numbers. The user must first, when entering a line of data, enter a line number which will be interpreted as a call to the editor. Valid line numbers must contain four digits; preceding zeros must be included. An entry to the editor is terminated by the carriage return key. No more than 64 characters may be input for one line. All lines are ordered by the ascending numeric sequence of their line numbers. If the user wishes to insert lines after the initial entry is made, it is suggested that the original line numbers be separated by five unit intervals.

ASSEMBLER

When the Assembler is given control by the executive, it proceeds to translate the Symbolic 8080 Assembly Language (source) program into 8080 machine (object) code. Features of the Assembler include:

Free format source input
Symbolic addressing, including forward references and relative symbolic references
Complex expressions may be used as arguments
Self defining constants
Multiple constant forms
Up to 256 eight character symbols
Reserved names for 8080 registers
ASCII character code generation
8 Pseudo Operations (assembler directives)

The Assembler translates lines in the file into object code. The second character following the line number is considered to be the first source code character position. Hence, the character immediately following the line number should normally be blank. Line numbers are not processed by the Assembler; they are merely reproduced on the listing.

STATEMENTS, COMMENTS, AND PSEUDO OPERATIONS

During pass 1, the Assembler allocates all storage necessary for the translated program and defines the values of all symbols used, by creating a symbol table. The storage allocated for the object code will begin at the first byte dictated by the first parameter in the original Executive ASMB command.

During pass 2, all expressions, symbols, and ASCII constants are evaluated to absolute values and are placed in allocated memory in the appropriate locations. The listing, also produced during pass 2, indicates exactly what data is in each location of memory.

STATEMENTS

Statements may contain either symbolic 8080 machine instructions or pseudo-ops. The structure of such a statement is

NAME OPERATION OPERAND COMMENT

The name-field, if present, must begin in assembler character position one. The symbol in the name-field can contain as many characters as the user wants; however, only the first 8 characters are used in the symbol table to uniquely define a symbol. All symbols in this field must begin with an alphabetic character and may contain no special characters.

The operation-field contains either an 8080 operation mnemonic or an Assembler pseudo-operation code.

The operand-field contains parameters pertaining to the operation-field. If two arguments are present, they must be separated by a comma.
Example:

```
0005 FLOP      MOV   M,B      THIS IS A COMMENT
0015 ;COMMENT LINE
0025           JMP   BEG
0035           CALL  FLOP
0045 BEG       ADI   8+6-4
0055           MOV   A,8
```

All fields are separated and distinguished from one another by the presence of a Tab (control-I) or one or more spaces.

The comment-field is for explanatory remarks. It is reproduced on the listing without processing. See example 0005. Comment lines must start with a semi-colon (;).

SYMBOLIC NAMES

To assign a symbolic name to a statement, one merely places the symbol in the name-field. To leave off the name-field the user skips two or more spaces after the line number and begins the operation field. If a name is attached to a statement, the assembler assigns it the value of the current location counter. The location counter always holds the address of the next byte to be assembled. The only exception to this is the EQU pseudo-operation. In this case a symbol in the name-field is assigned a value which is contained in the operand-field of the EQU pseudo-operation statement. Example:

```
0057 POTTA     EQU     128
```

assigns the value of 128 to the name POTTA. This data can then be used elsewhere in the program as: eg. ADI POTTA

Names are defined when they appear in the name-field. All defined names may be used as symbolic arguments in the argument-field. See

SYMBOLIC NAMES, cont.

examples on previous page.

In addition to user defined names, the assembler has reserved several symbols, the value of which is predetermined. These names may not be used by the user except in the operand-field. They are (with their value in parentheses):

A the accumulator (7) p the PSW
B Register B (0)
C Register C (1)
D Register D (2)
E Register E (3)
H Register H (4)
L Register L (5)
M Memory (through H, L) (6)
S Stack pointer (6)

In addition to the above reserved symbols, there is the single special character symbol ($). This symbol changes in values as the assembly progresses. It is always equated with the value of the program counter after the current instruction is assembled. It may only be used in the operand-field.

Examples:

```
    JMP  $     Means jump to the next location after
    MOV  A,B   instruction; i.e., the MOV instruction.

    LDA  $+5   Means load the data at the fifth location
    DB   0     after this location.  In this case the
    DB   1     data has the value of five.
    DB   2
    DB   3
    DB   4
    DB   5
```

RELATIVE SYMBOLIC ADDRESSING

If the name of a particular location is known, a nearby location may be specified using the known name and a numeric offset.
Example:

```
     JMP   BEG
     JPE   BEG+4
     CC    SUB1
     CALL  ADD1
 BEG MOV   A,B
     HLT
     MVI   V,'B'
     INR   B
```

In this example the instruction JMP BDG refers to the MOV A,B instruction. The instruction JPE BEG+4 refers to the INR B instruction. BEG+4 means the address BEG plus four bytes. This form of addressing can be used to locate several bytes before or after a named location.

CONSTANTS

The Assembler allows the user to write positive or negative numbers directly in the statement. They will be regarded as decimal constants and their binary equivalents will be used appropriately. All unsigned numbers are considered positive. Decimal constants can be defined using the descriptor "D" after the numeric value. (This is not required, as the default assignment is decimal.)

Hexadecimal constants may be defined using the descriptor "H" after a numeric value. (i.e. 10H, 10H, 3AH, 0F4H).
NOTE: A hexadecimal constant cannot start with the digits A-F. In this case, a leading 0 must be included. This enables the Assembler to differentiate between a numeric value and a mnemonic symbol.

CONSTANTS, cont.

ASCII constants may be defined by enclosing the ASCII character within single quote marks, i.e., 'C'. For double word constants, two characters may be defined within one quote string.

EXPRESSIONS

An expression is a sequence of one or more symbols, constants or other expressions separated by the arithmetic operators plus or minus. Examples:

```
PAM+3
ISAB-'A'+5
LOOP+32H-5
```

Expressions are calculated by using 16 bit arithmetic. All arithmetic is done modulo 65536. Single byte data cannot contain a value greater that 255 or less than -256. Any value outside this range will result in an assembler error.

PSEUDO-OPERATIONS

The pseudo-operations are written as ordinary statements, but they direct the assembler to perform certain functions which do not always develop 8080 machine code. The following section describes the pseudo-ops.

ORG

The ORG statement will set the program counter to the value in the operand-field. The label-field is optional but if present will be equated to the given operand-field.

END

The END statement informs the assembler that the last source statement has been read. The assembler will then start on pass 2, or terminate

END, cont.

the assembly and pass control back to the executive. This pseudo-op is not required as the assembler will stop when an end of file has been reached.

ENDS

The ENDS statement functions exactly like the END statement except it prints the symbol table at the end of pass 2.

LON

The LON will turn the print listing flag on. This results in a full listing from the LON statement until a LOFF statement or the end of assembly.

LOFF

The LOFF statement will turn the print flag off. This results in an "error only" listing until a LON statement or the end of assembly.

EQU

The EQU statement is used to equate a label with an expression.
Example: SAM EQU 12D equates SAM to be equal to 12

DS

The DS statement reserves the number of memory bytes specified by the operation-field.

DB

The DB statement generates a single byte constant specified by the operation-field.

DW

The DW statement is used to define two bytes of storage. The evaluated argument will be placed in the two bytes; high order 8 bits in the low order byte and low order 8 bits in the high order byte. This conforms to the Intel format for two byte addresses.

ASSEMBLER ERRORS

The following error flags are output on the assembler listing when the error occurs. Some of the errors are only output during pass one.

O	Opcode Error
L	Label Error
D	Duplicate Label Error
]	Missing Label Error
V	Value Error
U	Undefined Symbol
S	Syntax Error
R	Register Error
A	Argument Error

SYMBOL TABLE

The assembler normally allocates a symbol table of 100 (decimal) entries, each entry requiring 10 (decimal) bytes. Decreasing the size of the symbol table frees memory for use by the editor for storing the source text. To increase or decrease the contents of byte 2006H in the program, and restart the editor/assembler at 2000H. The contents of location 2006H must be greater than 1 and less than 255. This implies a maximum number of symbols at any time of 254.

POLYMORPHIC SYSTEMS SOFTWARE USER GROUP

PolyMorphic Systems would like to encourage the interchange of software between POLY 88 users. In order to do this we will distribute Cassettes of programs we feel are of interest to other users.

Should you develop a program that you would like to see made available to fellow users send us a cassette with a copy of the program in Byte format together with instructions (in Xeroxable form) on how to run the program. Programs we decide to distribute will be included in our price list and will be sold for 10 to 15 dollars. We accept no responsibility for the content or applicability of any program distributed in this manner, and offer this strictly as a service to POLY 88 users.

SOFTWARE USER FEED BACK

Program Name_____ Version_____

1. How received?

   ☐ Purchased from dealer Dealer Name?_____

   ☐ By mail factory direct

   ☐ Other

2. How much memory do you have on your system? _____K

3. How much do you expect to have?

   In six months _____K 1 year _____K ultimately _____K

4. How would you rate yourself as a computer user?

   ☐ Just starting

   ☐ Beginner

   ☐ Intermediate

   ☐ Wizard

   ☐ I remember the 650

5. Do you program primarily in

   ☐ BASIC ☐ Ass'y Language

   About how many lines is one of your average programs?_____

6. How would you rate the time used on your machine for the following:

   _____% Games _____% Maintenance _____% Building programs Other_____

7. How would you compare this program to others on the market?

   ☐ Better than most

   ☐ As good as most

   ☐ Worse than most Why?_____

8. What additional programs would you like to see?_____

   _____

Comments on reverse side

Return to PolyMorphic Systems, 460 Ward Drive, Santa Barbara, California 93111