POWERSOFT, INC. Cat. No. ABS0110

documentation of

ASSEMBLER

for the Apple II

The Powersoft (Apple II) 6502 Assembler is a Random Access Memory resident Assembler for the Apple II microcomputer system. The Assembler was designed to provide ease of use required by the novice, while providing capabilities powerful enough for the expert.

The Assembler facilitates the translation of symbolic language source programs into machine executable code. The resulting object code may then be recorded to tape for future execution by the computer.

The source code files read in by the Assembler must have been created using the Powersoft File Editor Program. In order to use this Assembler the user is expected to have a reasonable knowledge of 6502 machine language and the hexidecimal numbering system.

The Assembler may be run on an Apple II 8K to 48K microcomputer. It automatically adjusts its internal buffers to the memory size available. For ease of loading the Assembler program contains a small BASIC routine in front of it which allows the user to issue a BASIC Load Statement, to load in the Assembler program, and a Run Statement to start execution of the Assembler program.

Op-Code and Operand must be the same format as that shown in the "MCS6500 MICRO-COMPUTER FAMILY PROGRAMMING MANUAL" publication # 6500-50A from MOS Technology, Inc. 950 Rittenhouse Road, Norristown, PA 19401. It is strongly recommended that the user have a copy of this publication at his disposal. This documentation will not duplicate the ~~information found in that~~ publication.

The following notation conventions are followed in this text for brevity.

1. [] Square brackets enclose optional information.

2. CAPITALS represent information which must be present in a command.

3. lowercase letters represent user supplied values.

4. ƀ indicates manditory blank space required.

5. { } braces enclose information of different forms which may be used in place of each other.

LABEL

A LABEL is a symbolic name of a line code. LABELs are always optional unless the Op-Code is EQU. A LABEL is a string of alphanumeric characters no greater than six characters in length. The first character must be alphabetic. The LABEL must contain alphanumeric characters in the other five positions. No special characters may be used in LABELs.

OPCODEs

This Assembler supports all OPCODE mnemonics found in the previous mentioned publication. The Assembler also supports the PSEUDO-OPCODEs listed later.

NOTE: At this time the Assembler does not support Macros or Conditional assembly commands.

OPERAND

The OPERAND formats are the same as the assembly language form found in the previously mentioned 6502 programming manual. The OPERAND formats for the supported PSEUDO-OPCODEs are described later in this documentation.

In order to save memory in the symbol table buffer the user may use offsets after a symbolic name in the OPERAND. The offset must follow the name with no space. The offset must start with either a plus sign (+) or minus sign (-) followed by a decimal value of the offset. e.g. b̸LDAb̸TABEL+10 The example will substitute the address of TABEL plus ten bytes when the line is assembled.

COMMENT

All COMMENTs must start at least two spaces beyond the last character in the OPCODE or one space beyond the OPERAND. If a line has an asterisk (*) in the first column the entire line is treated as a COMMENT.

PSEUDO-OPCODEs

There are presently five PSEUDO-OPCODEs (assembler directives) which the Assembler will recognize. These assembler directives, although written like processor instructions are commands to the Assembler instead of the processor. They direct the Assembler to perform specific tasks during the assembly process, but have no meaning to the 6502 processor. The Assembler PSEUDO-OPCODEs are:

ORGb̸nnnn Sets the address reference counter to the value of nnnn. This will cause the Assembler to assign this address to the start of the assembled code. If this instruction is missing then the Assembler will assign the start address of the object code buffer to the assembled code. If the ORG command is used it should be the first instruction of the program, otherwise the resulting object code will be unpredictible.

TITLEb̸'xxxx.....' Sets the title line on each page of the listed output. The TITLE may be up to 23 characters in length enclosed in single quotes.

EQU $nnnn Sets the value of a LABEL to nnnn in the program, it can occur only once for any LABEL. The value must be a hex number preceeded by a dollar sign ($). ($nnnn is keyed in the OPERAND field.)

$$[\text{label}]\not{b}\text{DC}\not{b}\left\{\begin{array}{l} \text{A}\left\{\begin{array}{l}\text{'\$nnnn'}\\ \text{'label}\left[\begin{Bmatrix}+\\-\end{Bmatrix}\text{nnnnn}\right]\end{array}\right\} \\ \\ \text{C 'ascii string'}\\ \text{H 'hex string'}\end{array}\right\}$$

The A operand option causes a two byte address to be generated.
The C operand causes an ASCII string to be generated. This string may be up to 66 characters in length enclosed by single quotes.
The H operand option causes a hex string to be generated. In the string every two characters represents a byte. Up to 33 bytes of hex information may be enclosed in single quotes.

[label]ɃDSɃ	nnnn	This command reserves nnnn bytes of memory starting at the current value of the reference counter. nnnn is a decimal value.

During the execution of the Assembler the program will ask the user for certain information which it needs in order to give the user what he wants. The first question will ask the user which run time options are needed. The user may give one of three responses:

1. ALL This response will cause the Assembler to assume that the answer to all questions is yes and ~~that the~~ visual output is to be formatted for an 80 column printer.
2. NONE This response will cause the Assembler to assume that the answer to all questions is no and that the visual output is to be formatted for the CRT screen.
3. LIST This response causes all the questions to be asked and allows the user to tailor the Assembler to his needs.

The other questions asked if the LIST option is used are:

1. WRITE OBJECT TO TAPE Yes to this question causes the object file to be saved on tape.
2. LISTING Yes to this question causes a listing to be generated of the assembled code. No to this question inhibits all listings except the error file.
3. OUTPUT DEVICE (VIDEO OR PRINTER) This question wants to know which print format you wish.

Each time the Assembler is executed it will ask the following questions:

1. SOURCE IN MEMORY If no is answered here then the Assembler will read the source file in from tape.
2. ENTER DATE The user enters the date wanted on the listings.

After the assembly listing is complete the error file will then be listed. The numeric information indicates the line number on which the error occured and the alpha code tells the nature of the error.

A	Addressing Error	M	Opcode Missing
D	Duplicate Label	L	Invalid Label
P	Illegal Operand	B	Relative Branch Out Of Range
V	Symbol Table Full	U	Undefined Symbol

ASSEMBLER SOURCE CODE FORMAT

The basic format of a line of source code is as follows:

```
[LABEL]ɃOPCODEɃ[OPERAND]Ƀ[COMMENT]
```

GUARANTEE

POWERSOFT, INC. guarantees the playback of its pre-recorded tapes when purchased new, provided the playback head of the tape recorder used is properly aligned. All pre-recorded tapes are produced on the finest quality professional duplicating equipment available. *The program is recorded at least twice on the cassette.*