User Manual

Pullet 16 Assembler

Abstract
This user manual is going over useful information needed to use the Pullet16.

Adarius Adams, JeeJun Kim, Andrew Gray, Christopher Moyer, Vinay Penmet

General

With this assembler inputting ASCII data, and it should be outputting binary files. The hardware loads the machine instruction from whatever memory address is stored in the PC. It tries to decode, and then execute that instruction. The load, decode, execute sequence causes the PC to be incremented by one. Whatever is stored at that memory location is then loaded, decoded, and executed, etc. Memory referencing can be done either with direct addresses or with indirect address. The contents of a memory location are taken to be not the data but the address at which the data is to be found. The indirection indicator is an asterisk. For example:

```
STC    LOC
```

generates an instruction that will store the `ACC` contents at the location of `LOC` in memory, while

```
STC * LOC
```

generates an instruction that will fetch the contents of `LOC` as an address `ADDR` and the store the `ACC` contents at the location of `ADDR` in memory.

Formats

There are two formats for the machine instructions. The binary patterns that are machine code is as follows:

Machine Code Format I:

bits 0-2 opcode
bit 3 0 value indicates direct addressing, 1 indicates indirect
bits 4-15 memory address in hexadecimal

Machine Code Format II:

bits 0-2 opcode
bits 3 forced zero
bits 4-15 function selector code

****Note that these patterns are apart of the analog file to an a.out file****

Format I

Mnemonic opcode	**Binary opcode**	**Description**
BAN	000	Branch on ACC negative
SUB	001	Subtract contents of memory from ACC
STC	010	Store ACC in memory and then clear ACC
AND	011	And ACC with contents of memory
ADD	100	Add contents of memory to ACC
LD	101	Load ACC from contents of memory
BR	110	Unconditional branch

Format II

Mnemonic opcode	**Binary Opcode**	**Hex opcode**	**Description**
RD	1110 0000 0000 0001	E001	Read from standard input into ACC

STP	1110 0000 0000 0002	E002	STOP execution
WRT	1110 0000 0000 0003	E003	Write from ACC to standard output

Assembler Pseudo-op instructions:

- **ORG** - Set program counter to the value of the operand.
- **END** - End of input.
- **HEX** - Define a constant to be stored at the current PC location.
- **DS** - Define storage of *n* words beginning at the current PC location.

How To Use

In the terminal, the executable file for the Pullet16 Assembler will be named Aprog. To execute the program, use the terminal and type

```
./Aprog
```

The next sequence will be used for the input file for the assembler.

```
./Aprog ./inputfile
```

Remember that the input file should only contain lines of 16 ASCII 0's and 1's.

The sequence after that will be used for the output file. There will be a .bin of hex values and a .txt file of binary values for the output.

```
./Aprog ./inputfile outputfile
```

The last sequence in the terminal will be used for the log file, where all of the errors and logging will be.

```
./Aprog ./inputfile outputfile logfile.txt
```

Symbols

For a symbol to be valid, it must only be defined once, and the *first* character must be alphabetic; the *second* and/or *third* characters must be alphanumeric. In the symbol table below it will show their decimal locations as memory addresses.

```
SYMBOL TABLE
    SYM LOC FLAGS
SYM IDX  18
SYM N    20
SYM NN   21
SYM ONE  19
SYM TOP   3
```

Errors

The assembler will catch usual errors in the input, including:

- If an address is more than or equal to 4096 decimal, or if the address is less than 0.
- If a symbol is invalid or is defined more than once.
- If an opcode mnemonic is not a legal machine instruction or a pseudo-instruction.
- If a symbol used as a symbolic operand is not defined.
- If the input does not contain an END statement.
- If the hex operand is not legal.

All errors will be stated in the log file, and the program will keep assembling regardless of errors.