Assembler Linker Loader

Developed a 2 pass 8085 assembler, linker and loader in Python. It supports various new instructions (not in 8085 instruction set), looping capability and macros, which are converted to the 8085 assembly language accepted by GNUSim8085. It also detects various kinds of errors in syntax and semantics.

New Instructions defined

1. MUL Address1,Address2

The values stored in Memory Location Address1 and Address2 are multiplied and the result is stored in register B and C.

The Register C contains the lower bits and B contains the higher.

For Egs. if the output is BC 2 5

Answer = 2*256 + 5

2. SWAPR Reg1,Reg2

This instruction in used to swap the values stored in two registers.

Note: Accumalator can't be used for this instruction, Although the value of A remains preserved even after the use of instruction.

3. SWAPM Address1,Address2

This instruction in used to swap the values stored in two memory locations. The value in memory location of address1 is moved to address2 and that of address2 to address1.

No change to registers of flags done.

4. CCF

CCF is a Clear Carry Flag instuction used to Reset the Carry Flag. No change to any other flag done.

5. NOT Address

This is used to do the logical NOT of the value stored in memory location and the new value is stored in the same address.

6. DOUBLE Address

DOUBLE instruction is used to double the value stored in Memory location and the new value is stored in the same address.

7. MOVM Address1,Address2

The value stored in Address2 is moved to address1.

8. COMP Reg1,Reg2

Compares the values of two registers and the flags are modified accordingly.

9. LSTART, LEND

Looping

The user is given a feature to use loops just as similar to for loops in C programs.

For this two instructions are defined LSTART and LEND.

An instruction LSTART is given to start the loop followed by the name of the loop and three immediate values in the next line*.

These instructions are used to start and End the loop. The first line of the line inside the loop defines the name of the loop and three immediate integers which denote starting number, the increment number and the ending number.

For egs,

LSTART

abc 2,2,8

.

.

Instructions

.

.

LEND

All the instructions in the loop run for 3 times.

Because (8-2)/2 = 3.

10. **Macros**

In this software the user is also given a feature to implement Macros which is not a part of 8085 assembly language. The user is allowed to define a macro by saying MACRO ‘name’

Process

Linker

We run all the files through their own pass1 and pass2 making their symbol table and mnemonic table. Besides that we create an “EXTERN table” having all the references to external variables/symbols.

Then we combine the files and edit the symbol table to code to which we have reference by adding the offset created in memory by first file. We then replace the external symbols in the first file with their modified addresses.

Conversion of Code to 8085 Assembly Instruction set

All the code written is converted to the instruction set that a 8085 simulator can read and execute.

Macros are expanded, Linking is done, Loops are replaced by new instructions and we get a working Assembly Code.

Pass 1

- After the conversion of code to assembly(readable by 8085) the code is sent through Pass 1 procedure.
- A Symbol table is generated which stores the labels and its location counter.
- Another table, Mnemonics table is made which stores the mnemonics used with their opcodes and Addressed
- Extern Table is generated which stores the external variables linked to other files.

Pass 2

- After passing through Pass1 the code is sent through Pass 2 procedure.
- In this procedure Labels are replaced with their Location Counter.

Loading

- The user can load each file to different location at load time. After the inputs are taken, the locations offset are done and the final program is saved in a text file and it is 8085 readable.
- This program is copied and ran in gnu8085sim.