Assembler

How to Run:

Navigate to the Directory where Assembler.py is stored. Run the following command on the terminal. python Assembler.py <<path of the file to be assembled>>. The output will be generated at the same location as the input file.

Assumptions:

1. Variables and direct addressing to the memory location are handled.
2. END statement is necessary to produce the machine code.
3. The user can give comments in the Assembly code. The comment must be preceded by a hash (#). The statements after the # will be ignored during assembly.

Working

First Pass:

In the first pass, the assembler generates the symbol table by traversing the input assembly code. When a declarative statement of INP type or SAC type is read, the variable following the opcode is directly allotted an address. This address is determined by the initial start statement and the number of instructions given in the assembly code. Labels are identified using the branch statements which use the concept of forward referencing to put the label in the symbol table. Once the label definition is encountered, the symbol table is updated with the address of the label.

Memory Allocation:

Variables are allocated memory location after the length of the Assembly program. For example, if the program is of 20 lines then the variable memory allocation starts from 21 (in binary). In lines where a label has been used for referencing it gets translated into its Line number in Binary.

Second Pass:

The assembly code is read again and with the help of symbol table and opcode table the machine code is generated.

Error Handling:

1. Provided more no of operands than required.
2. Provided less no of operands than required.
3. Label not defined.
4. Variables not defined.
5. END statement missing.
6. Instruction not defined if label denotes a blank instruction.

Assembler Output:

Navigate to the Directory where input file is stored. The Assembler Generates a folder with the name <<input_filename>>_Assembled. The folder contains two text files

1. SymbolTable.txt
2. MachineCode.txt
3. OPCodeTable.txt

File Descriptions:

1.) SymbolTable.txt :

This file contains the symbol table generated during the first pass of the Assembler. It contains the Line number where the symbol was encountered, the type of the symbol, the actual symbol and the Address of the Symbol.

2.) MachineCode.txt :

This file contains the machine code generated by the Assembler. It contains the 12-bit instruction in binary number. Each line denotes a word in the memory.

3.) OPCodeTable.txt :

This file shows all the opcodes encountered during the traversal of the input file. It shows the offset along with the opcode and its equivalent in binary.

OPCODE TABLE

Opcode	Meaning	Assembly Opcode
0000	Clear accumulator	CLA
0001	Load into accumulator from address	LAC
0010	Store accumulator contents into address	SAC
0011	Add address contents to accumulator contents	ADD
0100	Subtract address contents from accumulator contents	SUB
0101	Branch to address if accumulator contains zero	BRZ
0110	Branch to address if accumulator contains negative value	BRN
0111	Branch to address if accumulator contains positive value	BRP
1000	Read from terminal and put in address	INP
1001	Display value in address on terminal	DSP
1010	Multiply accumulator and address contents	MUL
1011	Divide accumulator contents by address content. Quotient in R1 and remainder in R2	DIV
1100	Stop execution	STP