## COMPUTER ORGANISATIONS

## Assembler Project

* Ananya Singh (2019144)
* Mansi Singhal (2019370) (group 6)

Assembler is a program that converts assembly language into machine language that can be recognized by a processor. Each statement of an assembly language program gets translated to one machine instruction so the programmer has access to all features and instructions available in machine language.

For this project we have made a two pass assembler for 12 bit accumulator architecture where the first four bits show the opcode in binary representation and the rest 8 bits show the address. Our assembler is programmed in Python, it takes an input file with an assembly language program and converts it into machine language.

**Assumptions for the assembly language program**:

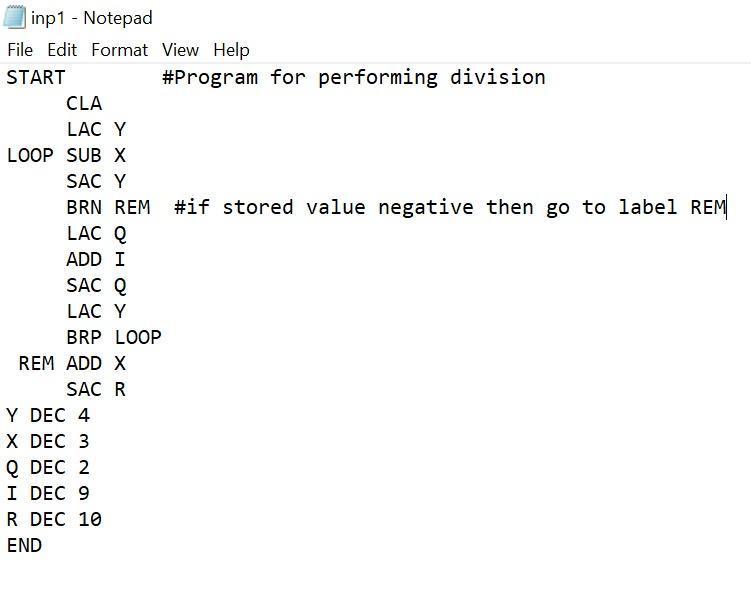
1. The variables must be declared at the end of the instructions, sequentially and in order of their appearance in the code by using statements like “X DEC 4”.
2. There must not be any literals, macros and procedures present in the program.
3. There must be “START” and “END” statements in the program to signify start and end of the program.
4. The program counter of the declaration statement of a variable will be the assigned as the address for the variable.
5. If there is a number mentioned after “START” then the program counter will start from that number.
6. The comments should start with a “#”.
7. There can only be one operand per instruction because it is a accumulator architecture based assembler. For CLA and STP there will be no operands.

**Error handling**:

1. If the opcode is invalid then the assembler throws an error.
2. If a label has not been defined then the assembler throws an error.
3. If a variable has not been declared then the assembler throws an error.
4. If there are multiple definitions of a label then the first definition is considered and the program is executed accordingly.
5. Defined label name cannot be used as a variable name and vice versa. An error will be thrown in this case.
6. If “START” and “END ” statements are not found then an error will be thrown.
7. If the number of operands supplied for an opcode is incorrect, an error will be thrown.
8. If the program counter for any statement exceeds 255(in decimal) then the assembler throws an error because 255 is the maximum decimal number that can be represented as a 8 bit binary number. So an address out of bound error will be thrown.

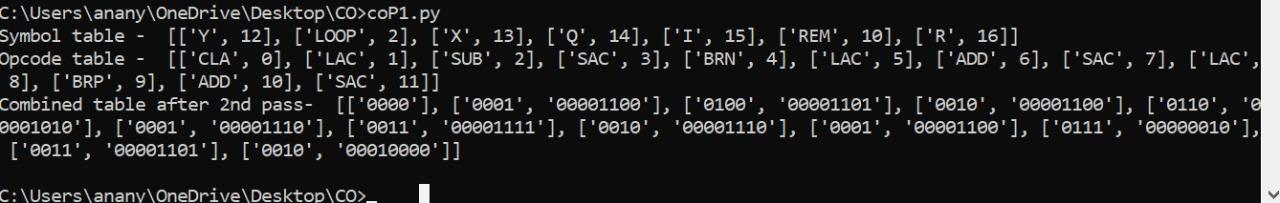
**Working of the assembler:**

Here is a sample input for which we will demonstrate how our assembler works:

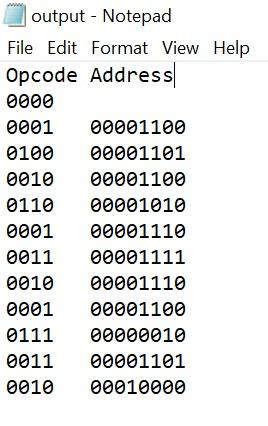


The assembler reads the input file in the first pass and makes the symbol table and opcode table.We have assigned the program counter of the declaration statement of variables as their address. If in the first pass all symbols are not found then the assembler signals error. The program is initialized with the number given after the “START” statement and if no number is given then it is initialized with zero.

In the second pass, the input file is read again and a combined table of the opcode in binary and corresponding address is made which is basically the machine code. Here are the mentioned tables for the sample input:



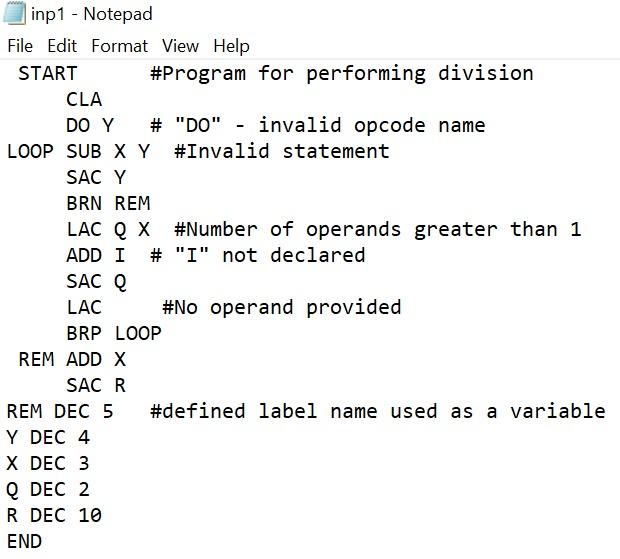
If the input file meets all the requirements and there are no errors found then the output file will display the corresponding machine code. The machine code for the sample input is as follows:



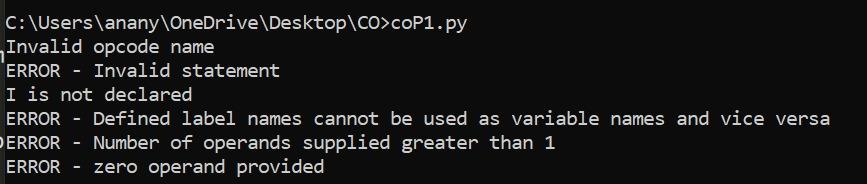
***Some other inputs:***

1. *Input file with some errors*

Input:

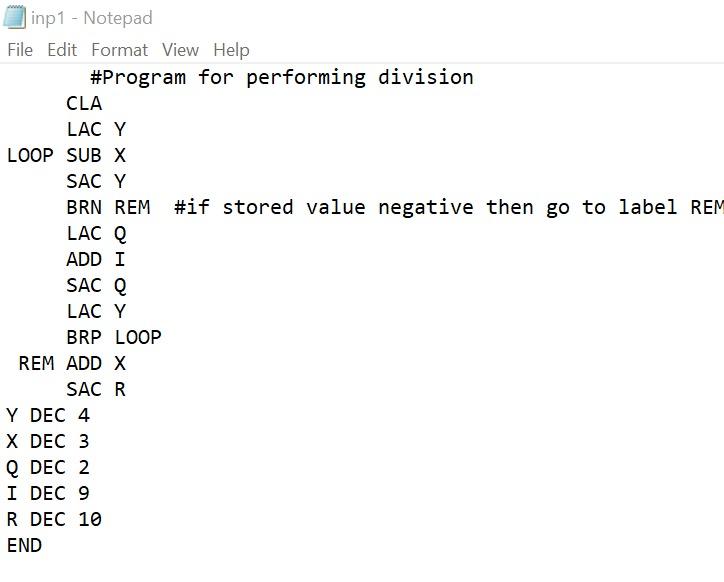


Output:

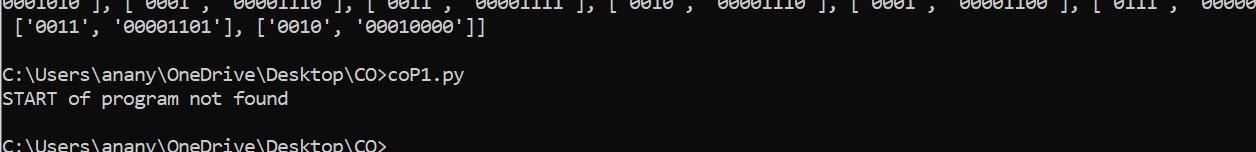


2. No *“START” statement :*

Input:

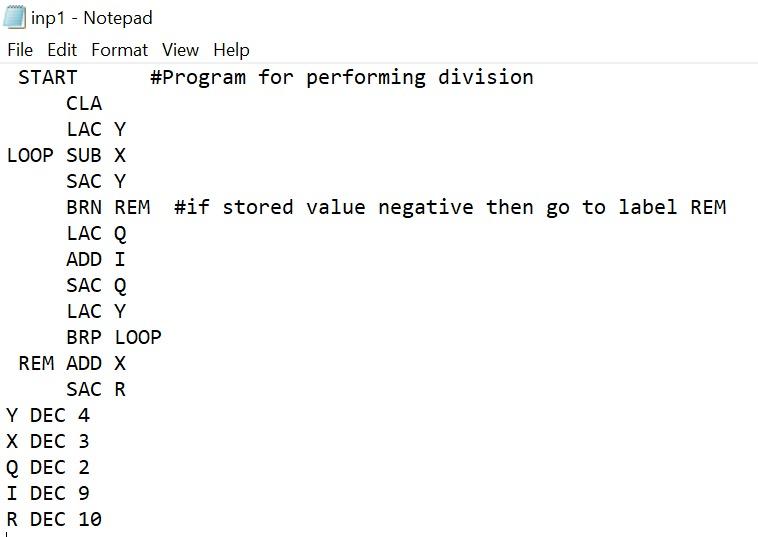


Output:

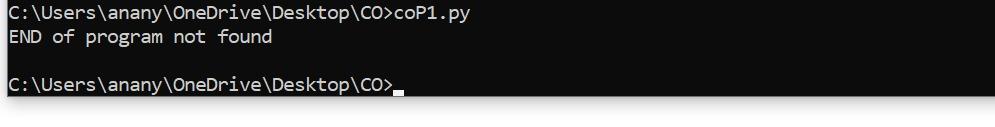


1. *No “END” statement*

Input:



Output:



*NOTE:*

*We have attached the code file,2 input files named “inp1” and “inp\_error” with errors in it, and an output file named “output”. After running the code file on command prompt, check the output file for the result.*