Course: CSE112 Computer Organization at IIITD

COMPUTER ORGANIZATION PROJECT - 1

Project Members:

1. Bhaskar Gupta
2. Rishav Kumar

Assumptions:

1. Clear Accumulator(CLR) clears the accumulator. No address and value is present after clearing the accumulator.
2. Comments could be added using "//". They are removed by the assembler for conversion to machine code.
3. No macros and procedures are to be assembled.
4. No literals are to be handled.
5. Only the opcodes given should be used. All these opcodes are pre-added as a dictionary in the program and are not read from a separate file.
6. Label cannot be an opcode and vice-versa.
7. Variable cannot be an opcode and vice-versa.
8. Label cannot be a variable and vice-versa.
9. Variables should only be defined once.
10. Number of instructions should not exceed 256 or else it will give an error.

Opcode Table

This table is used to check whether an instruction has an opcode or not.

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

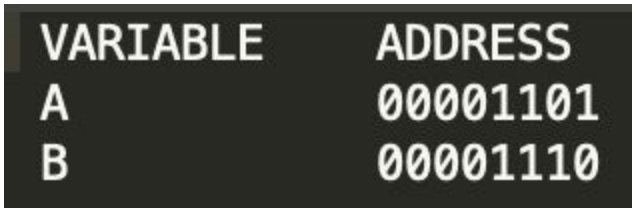
Assembly Test Code

This file is given as an input in both the passes. In the first pass, it checks for any kind of errors.

```
CLA
INP A
INP B
LAC A
SUB B
BRN L1
DSP A
CLA
BRZ L2
L1: DSP A
CLA
BRZ L2
L2: STP
```

Variable Table

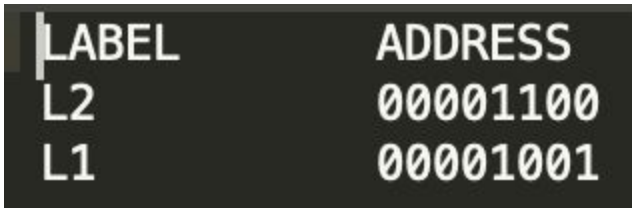
This table comprises the variable name and its local address.



VARIABLE	ADDRESS
A	00001101
B	00001110

Label Table

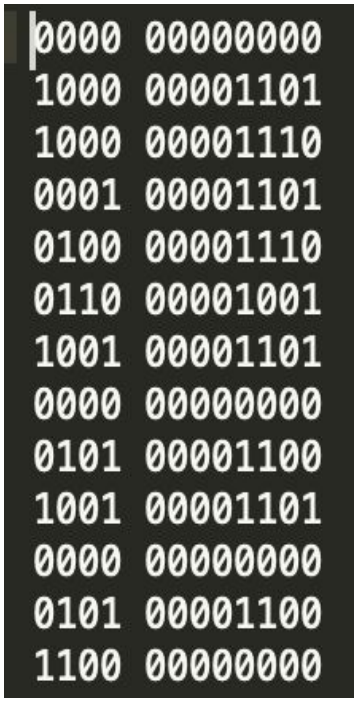
This table comprises the label name and its address in the input file.



LABEL	ADDRESS
L2	00001100
L1	00001001

Machine Code

This text file has the machine code of the input file.



```
0000 00000000
1000 00001101
1000 00001110
0001 00001101
0100 00001110
0110 00001001
1001 00001101
0000 00000000
0101 00001100
1001 00001101
0000 00000000
0101 00001100
1100 00000000
```

Error Handling

1. More than one opcode in Instruction

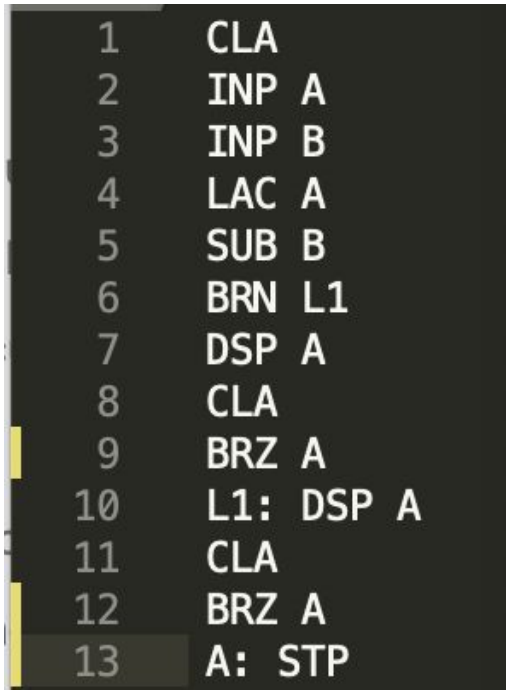
INP and DSP both are opcodes. But, a instruction cannot have more than one opcode so it will throw an error.

```
2   INP DSP       //INP A
```

```
Error found on line number 3: More than one Opcode found.
[Finished in 0.0s]
```

2. Label Name cannot be a Symbol Name

If we use a symbol name as a label name, it will throw an error as shown.




```
Error found on line number 9:A is a label type symbol.
Error found on line number 12:A is a label type symbol.
```

3. More than one Symbol provided

If we provide more than one symbol to an instruction, it will throw this error as shown.

```
5   SUB B A
```

```
ERROR on Line 5: More than one variable/label provided.
```

4. More than one Label provided

If we provide more than one label to an instruction, it will throw this error as shown.

```
6    BRN L1 L2
```

```
ERROR on Line 6: More than one variable/label provided.
```

5. No Opcode found in a Instruction

Each instruction should have an opcode but if it does not have one then it will give this error.

```
  5    A B
```

```
A B

Error found on line number 5: No Opcode found.
[Finished in 0.0s]
```

6. Insufficient Number of Arguments

If an opcode requires a variable and you don't provide it with a variable then it will throw this error.

```
  5    SUB
```

```
Error found on line number 5: Insufficient no. of arguments.
[Finished in 0.0s]
```

7. Formatting Error(Opcode occur after arguments in a Instruction)

If a variable is used before the opcode in an instruction then it will give the following error.

```
  3    B INP
```

```
Error found on line number 3: Formatting Error
[Finished in 0.0s]
```

8. Label Name cannot be a Opcode

If the label name is an opcode then it will give the following output.

```
10    INP: DSP A
```

```
Error found on line number 10: Label cannot be a opcode.
```

9. Memory Limit Exceeded

Overload triggered by more commands and variables processed than the maximum limit. Our assembler limit is 256 (0–255).

Pseudo Code

First Pass

If the first pass ends successfully, then only the second pass is executed.

1. Open input file to read the file line by line.

2. Check and remove the commented part in the line if any.

3. Check the presence of opcode in a line by using the function
   def opcodereturner(opcode,line):
   '''
   Parameter Type: Opcode Dictionary, Line/Word to check
   Return Type: String
   That is the opcode of the assembly opcode in that line.
   '''

4. Check if its a instruction or not by using the function
   def checkifinstruction(opcode,op):
   '''

Parameter Type: Opcode Dictionary, Word(Maybe Opcode)
Return Type: Boolean
Checks if the string is whether a instruction or not by
checking in the opcode dictionary keys.
'''

5. Count the no. of opcodes in an instruction using the function
   def no_of_opcodes(line,opcode):
   '''
   Parameter Type: Line, Opcode Dictionary
   Return Type: Int
   This function gives no. of opcodes in a line.
   '''

6. We have defined a boolean type variable "**error"** which is **False** in the beginning and turns **True** if an error occurs during the first pass. This stops the execution of the second pass from executing if "**error"** is equal to **True**.

Second Pass

1. This is only executed if "**error**" is equal to **False**.

2. All the Instructions are read again in the second pass.

3. Label Table and Variable Table are made in the second pass.

4. Whole assembly code is converted into machine code line by line and added into the file machinecode.txt.