Assignment 4 (Assembly Language Programming)

- **Assignment will be evaluated by the TAs.**
- **All codes must be properly documented and good code writing practice should be followed (carry marks).**
- **Copying is strictly prohibited. Any case of copying will automatically result in F for the whole course, irrespective of your performance in the other parts of the lab.**
- **You should work individually.**
- **You have to upload your code on Moodle. The deadline for submission is 17th February (Sunday). The assignment will be evaluated on Monday 18th February.**
- **Marks - 100**

Consider the following instruction set for a hypothetical machine. The machine supports only the register and immediate addressing modes. There are 32 registers in our machine, numbered R0 to R31, each having two bytes. R31 is the count register for LOOP. Program starts from the address location 0000H.

1) MOV 2) ADD 3) SUB 4) MUL 5) CMP 6) AND

7) OR 8) NOT 9) JMP 10) LOOP 11) HLT 12) JNZ

In this assignment, you are going to demonstrate the working of a two-pass assembler for the hypothetical machine! Write a program (in C) to implement the assembler. The input to your program would be an assembly language program written using instructions from the above set only. Your program should convert this input to a machine-level code. The conversion is done with two *passes*. In the first pass, the assembler generates Symbol Table and Machine-Opcode Table. The second pass of the assembler uses the tables generated in the first pass and generates the final machine level code. There are two Pseudo-Opcodes for our assembler (START, END). Note that these two pseudo opcodes do not need any further processing.

Program specifications:

Input: a text file containing the assembly code (should use the name "input.asm"). Also, the opcode-machine code mapping (shown below) and pseudo opcode tables (store and use them as separate text file).

1) MOV (0000) 2) ADD (0001) 3) SUB (0010) 4) MUL (0011)
5) CMP (0100) 6) AND (0101) 7) OR (0110) 8) NOT (0111)
9) JMP (1000) 10) LOOP (1001) 11) HLT (1010) 12) JNZ (1011)

Output: the following files (for the two passes)

Pass-1:

1) Symbol-Table File: It should have the symbol names along with its address, in the format <name, address>. Each line should contain only one <name, address> pair. The file name should be "symTable.txt".
2) Opcode-Table File: It should have the opcode names along with its machine code (from the opcode-machine code mapping table), in the format <opcode, machine code>. Each line should contain only one such pair. The file name should be "opTable.txt".

Pass-2:

1) The final machine code (should be named “output.o”).

Example: Consider the following input file.

```
       START
       MOV R2, 0003H
       MOV R31, 0004H
   L1: ADD R2, 0001H
       LOOP L1
       MUL R3
       AND R2, R9
       JMP L2
       OR R2, R5
   L2: HLT
       END
```

F**irst pass output**

1) Symbol-Table

L1:	0008H
L2:	0018H

2) Machine-Opcode Table

MOV	0000
ADD	0001
AND	0101
MUL	0011
JMP	1000
JNZ	1011
OR	0110
HLT	1010

Second pass will be having machine level code (also print the corresponding addresses as shown)

```
ADDRESS                  INSTRUCTION
0000                     0000 00010 00000000 00000011
0004                     0000 11111 00000000 00000100
0008                     0001 00010 00000000 00000001
000C                     0010 11111 00000000 00000001
0010                     1011 0008
0011                     0011 00001 00011
0013                     0101 00010 01001
0015                     1000 0018
0016                     0110 00010 00101
0018                     1010
```