CS222- Lab 4

Assembly language Programming

The goal of this is to familiarize the students with 8086 assembly language features.

Your assignment:
Using Dosbox environment, develop and Test the following (8086 ASM programs).

Develop test programs (2 example each) for each of the addressing modes below and verify the results.

1. **Register Addressing**
2. **Immediate Addressing**
3. **Direct Addressing**
4. **Register Indirect Addressing**
5. **Based Addressing**
6. **Indexed Addressing**
7. **Based Index Addressing**
8. **String Addressing**
9. **Direct I/O port Addressing (no need to test)**
10. **10. Indirect I/O port Addressing (no need to test)**
11. **Relative Addressing**
12. **Implied Addressing**

Ans:

P2.

Develop test programs (5 example each) for each of the case below and verify the results.

1. **Data Transfer Instructions**
2. **Arithmetic Instructions**
3. **Logical Instructions**
4. **String manipulation Instructions**
5. **Process Control Instructions**
6. **Control Transfer Instructions**

P1_3:

Write an assembly language program to find square and cube of a number

Ans here:

P1_4:

Write an assembly language program to find GCD of two numbers

Ans here:

P1_5:

Write an assembly language program to find largest and smallest number from a given set of numbers

Ans here:

Part 2: Machine-Level Representation of Programs 32/64 bit systems

Suppose we write a C code file code.c containing the following procedure definition:

Assume file name: code.c

```
int accum = 0;

 int sum(int x, int y)
 {
        int t = x + y;
        accum += t;
        return t;
 }
```

To see the assembly code generated by the C compiler, we can use the "-S" option on the command line:

unix> ***gcc -O1 -S code.c***

This will cause gcc to run the compiler, generating an assembly file code.s, and go no further. (Normally it would then invoke the assembler to generate an objectcode file.)
The assembly-code file contains various declarations including the set of lines:
sum:

```
pushl %ebp
movl %esp, %ebp
movl 12(%ebp), %eax
addl 8(%ebp), %eax
addl %eax, accum
popl %ebp
ret
```

If we use the '-c' command-line option, gcc will generate both compile and assemble the code:

unix> ***gcc -O1 -c code.c***

This will generate an object-code file code.o that is in binary format and hence cannot be viewed directly. Embedded within the 800 bytes of the file code.o is a 17-byte sequence having hexadecimal representation

55 89 e5 8b 45 0c 03 45 08 01 05 00 00 00 00 5d c3

To inspect the contents of machine-code files, a class of programs known as *disassemblers* can be used. These programs generate a format similar to assembly code from the machine code.With Linux systems, the program objdump (for "object dump") can serve this role given the '-d' command-line flag:
unix> *objdump -d code.o*

Suppose in file main.c we had the following function:

```
int main()
{
return sum(1, 3);
}
```

Then, we could generate an executable program prog as follows:

unix> ***gcc -O1 -o prog code.o main.c***

We can also disassemble the file prog:
unix> *objdump -d prog*

P2_1: Compare the code generate with
unix> *gcc -O1 -S -masm=intel code.c*
Ans. here:

:

P2_2: Test your own functions here:

Ans here

Function:

ASM code:

P2_2: Compile simple C programs and look at asm content:

Ans:

Submission :
Submit single doc/pdf file with above answers. Course work submission through cs322.iitp@gmail.com with subject: YourrollNo_Lab4. **Due on** 31st August 2018 , 5PM**.**