COMPUTER ORGANIZATION (CDA – 3103)

SPRING 2005

Lab # 4: Basic of Intel Assembly Programming

In this lab, we will discuss the basic of Intel Assembly Programming. For reference see following online material book on course website:

[1] PC Assembly Language, Paul A. Carter, October 10, 2004
[2] Subset of Intel Assembly Language

1 COMPUTER ORGANIZATION

[Kindly read the section 1.2 of the reference book given above.

The most important thing while writing an assembly program is to know:

1) The architecture of the target machine. This includes the general purpose and special purpose registers used for storing information in processor. It also includes the memory system that is attached with the processor and the interface between them.
2) The instruction set of the processor which is used by the assembly programmer for manipulating information with the processor and external system.

2 INSTRUCTION SET ARCHITECTURE

To know the instructions and their behavior, the best thing to start with is the instruction set manual which contains the assembly notation of all instructions, their source operator(s), destination operation(s) and their operations. It also indicates the impact of these instructions on other registers for showing some special behavior (overflow, underflow, divide-by-zero, etc ...)

3 A SAMPLE C TO ASSEMBLY CONVERSION

Let's say we have following program.

```
void main()
{
        int var1 , var2;
        int sum;

        var1 = 5;
        var2 = 10;
        sum = var1 + var2;
}
```

Now we have to convert each C instruction into corresponding assembly instruction. For this example I am using a hypothetical assembly language which is similar to Intel x86 assembly language.

Suppose we have following General Purpose Registers available to us.
AX, BX, CX and DX

```
void main()
{
        int var1 =0;        ------------------>  MOV AX , OxO
        int var2 =0;        ------------------>  MOV BX , OxO
        int sum = 0;        ------------------>  MOV CX , OxO

        var1 = 5;           ------------------>  MOV AX , Ox5
        var2 = 10;          ------------------>  MOV BX , Ox5

        sum = var1 + var2;  ------------------>  ADD CX, AX, BX
}
```

4 TRACING BEHAVIOR OF AN ASSEMBLY PROGRAM

Suppose we have following program.

```
MOV AX, 0x0                 =>      A = 0
MOV BX, 0x1                 =>      B = 0
MOV CX, VALUE               =>      B = some value

LABEL:

        MUL AX, AX, BX      =>      A = A * B
        ADD BX, BX, 0x1     =>      B = B + 1

        CMP.LE CX, 0x5 , LABEL  =>  if BX <= 5, then goto LABEL
```

Suppose we set VALUE = 5. Then this program is computing **5!**.