

EEE 410 – Microprocessors I

Fall 05/06 – Lecture Notes # 3

Outline of the Lecture

- Introduction to Assembly Programming (cont. from Lecture 2)
- Introduction to Program Segments

INTRODUCTION TO ASSEMBLY PROGRAMMING

ADD instruction

ADD destination, source ; add the source operand to destination

mnemonic operands

Ex:

```
MOV AL,24H ;move 24H into AL
MOV DL,11H ;move 11H into DL
ADD AL,DL ;AL=AL+DL (AL=35H) (DL =11H)
```

```
MOV CH,24H ;move 24H into CH
MOV BL,11H ;move 11H into BL
ADD CH,BL ;CH=CH+BL (CH=35H)
```

```
MOV CH,24H ;load one operand into CH
ADD CH,11H ;add the second operand to CH (CH=35H)
```

- If one register data is followed by an immediate data, it is called the *immediate operand*.

```
MOV CH,24H
ADD CH,11H
```

- 8-bit registers can hold FFH (255) maximum. Addition of larger numbers can be performed by the 16-bit nonsegment registers.

```
MOV AX,34EH
MOV DX,6A5H
ADD DX,AX ;DX=DX+AX (DX=9F3H)
MOV CX,34EH
ADD CX,6A5H ;CX=34EH+6A5=9F3H
```

INTRODUCTION TO PROGRAM SEGMENTS

Segment:

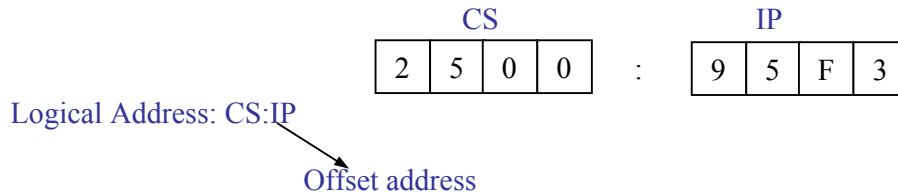
- A segment is an area of memory that includes up to 64K bytes and begins an address evenly divisible by 16 (such an address ends in 0H).
- Assembly Language Program consists of three segments:
 - *code segment* : contains the program code (instructions)
 - *data segment* : used to store data (information) to be processed by the program
 - *stack segment*: used to store information temporarily.

Logical and Physical Address

- **Physical Address** is the 20-bit address that actually put on the address bus. (in 8086)
 - Has a range of 00000H - FFFFFH
- **Offset Address** is a location within 64K byte segment range.
 - Has a range of 0000H - FFFFH
- **Logical Address** consists of segment address and offset address.

Addressing in Code segment

- To execute a program, the 8086 fetches the instructions from the code segment.
- The logical address of an instruction consists **CS** (Code Segment) and **IP**(instruction pointer)



- **Physical Address** is generated by shifting the CS one hex digit to the left and adding IP.

Example: CS:IP => 2500:95F3H

1. **Start with CS** 2500
2. **Shift left CS** 25000
3. **Add IP** 2E5F3 (25000+95F3)

The microprocessor will retrieve the instruction in turn memory locations starting from 2E5F3.

Ex: If CS=24F6H and IP=634AH, determine:

- a) The logical address is; 24F6:634A
- b) The offset address is; 634A
- c) The Physical address is; $24F60+634A=2B2AA$
- d) The lower range of the code segment: 24F6:0000 => $24F60+0000=24F60$
- e) The upper range of the code segment: 24F6:FFFF => $24F60+FFFF=34F5F$

Data segment

- The area of memory allocated strictly for data is called *data segment*.
- Just as the code segment is associated with CS and IP as segment register and offset. The data segment uses DS and an offset value. In 8086 BX, SI and DI are used to hold the offset address.

Ex: If DS=7FA2H and the offset is 438EH, determine:

- The physical address
- The lower range of the data segment
- The upper range of the data segment
- Show the logical address

- The Physical address is; $7FA20+438E= 83DAE$
- The lower range: $7FA20(7FA20+0000)$
- The upper range: $8FA1F(7FA20+FFFF)$
- The logical address is; $7FA2:438E$

Using the data segment

- Assume that a program is needed to add 5 bytes of data (25H, 12H, 15H, 1FH and 2BH)

```
One way:   MOV  AL,00H           ;initialize AL
           ADD  AL,25H
           ADD  AL,12H
           ADD  AL,15H
           ADD  AL,1FH
           ADD  AL,2BH           ; AL=25+12+15+1F+2B
```

} code and data are mixed

Other way: Assume that the offset for data segment begins at 200H

```
DS:0200 = 25
DS:0201 = 12
DS:0202 = 15
DS:0203 = 1F
DS:0204 = 2B
```

} within data segment

```
MOV  AL,0           ;clear AL
ADD  AL,[0200]      ;add the contents of DS:200 to AL
ADD  AL,[0201]      ;add the contents of DS:201 to AL
ADD  AL,[0202]      ;add the contents of DS:202 to AL
ADD  AL,[0203]      ;add the contents of DS:203 to AL
ADD  AL,[0204]      ;add the contents of DS:204 to AL
```

Little endian convention

Given 8-bit (1-byte) data, bytes are stored one after the other in the memory. However given 16-bit (2-bytes) of data how are data stored?

Ex: MOV AX,35F3H :load 35F3H into AX
 MOV [1500],AX : copy contents of AX to offset 1500H

In such a case the low byte goes to the low memory location and high byte goes to the high memory location.

DS:1500 = F3

DS:1501 = 35

This convention is called ***little endian convention***: This convention is used by Intel. The ***big endian convention*** is the opposite, where the high byte goes to the low address and low byte goes to the high address. Motorola microprocessor uses this convention.