

EEE 410 – Microprocessors I

Fall 04/05 – Lecture Notes # 5

Outline of the Lecture

- Flag Registers and bit fields
- 80x86 addressing modes.

FLAG REGISTERS AND BIT FIELDS

Flag Register and ADD instruction

The flag bits affected by the ADD instructions are: CF, PF, AF, ZF, SF and OF. The OF will be studied in Chapter 6.

Ex: Show how the flag register is affected by the addition of 38H and 2FH.

Solution: MOV BH,38H ;BH=38H
 ADD BH,2FH ;BH = BH + 2F = 38 + 2F= 67H

38	0011	1000
+ 2F	0010	1111
67	0110	0111

CF = 0 since there is no carry beyond d7
PF = 0 since there is odd number of 1's in the result
AF = 1 since there is a carry from d3 to d4
ZF = 0 since the result is not zero
SF = 0 since d7 of the result is zero

Ex: Show how the flag register is affected by the following addition

Solution: MOV AX,34F5H ;AX =34F5H
 ADD AX,95EBH ;AX = CAE0H

34F5	0011	0100	1111	0101
+ 95EB	1001	0101	1110	1011
CAE0	1100	1010	1110	0000

CF = 0 since there is no carry beyond d15
PF = 0 since there is odd number of 1s in the lower byte
AF = 1 since there is a carry from d3 to d4
ZF = 0 since the result is not zero
SF = 1 since d15 of the result is 1

➤ Note that the MOV instructions have no effect on the flag (Explain on the existing example)

Use of zero flag for looping

Zero flag is used to implement the program loops. Loop refers to a set of instructions that is repeated a number of times.

The following example shows the implementation of the loop concept in the program which adds 5 bytes of data.

```

Ex:          MOV  CX,05           ; CX holds the loop count
              MOV  BX,0200H       ; BX holds the offset data address
              MOV  AL,00          ; initialize AL
ADD_LP:      ADD  AL,[BX]         ; add the next byte to AL
              INC  BX             ; increment the data pointer
              DEC  CX             ; decrement the loop counter
              JNZ  ADD_LP         ; jump to the next iteration if the counter not
zero

```

80x86 ADDRESSING MODES

The CPU can access operands (data) in various ways, called addressing modes. In 80x86 there are 7 addressing modes

1. register
2. immediate
3. direct
4. register indirect
5. based relative
6. indexed relative
7. based indexed relative

1. Register addressing mode:

- involves the use of registers
- memory is not accessed, so faster
- source and destination registers must match in size.

```

Ex:          MOV  BX,DX
              MOV  ES,AX
              ADD  AL,BH
              MOV  AL,CX           ;not possible

```

2. Immediate addressing mode:

- source operand is a constant
- possible in all registers except segment and flag registers.

```

Ex:          MOV  BX,1234H       ; move 1234H into BX
              MOV  CX,223          ; load the decimal value 625 into CX
              ADD  AL,40H          ; AL=AL+33H

              MOV  DS,1234H       ;illegal           (ask what to do)

```

3. Direct addressing mode:

- address of the data in memory comes immediately after the instruction operand is a constant
- The address is the offset address. The offset address is put in a rectangular bracket

```

Ex:          MOV  DL,[2400]       ; move contents of DS:2400H into DL

```

Ex: Find the physical address of the memory location and its content after the execution of the following operation. Assume DS=1512H

```
MOV AL,99H
MOV [3518],AL
```

Physical address of DS:3518 => 15120+3518=18638H
The memory location 18638H will contain the value 99H

4. Register indirect addressing mode:

- The address of the memory location where the operand resides is held by a register.
- SI, DI and BX registers are used as the pointers to hold the offset addresses.
- They must be combined with DS to generate the 20-bit physical address

Ex: MOV AL,[BX] ; moves into AL the contents of the memory location pointed to by DS:BX

Ex: MOV CL,[SI] ; move contents of DS:SI into CL
 MOV [DI],AH ; move the contents of AH into DS:DI

5. Based relative addressing mode:

- BX and BP are known as the base registers. In this mode base registers as well as a displacement value are used to calculate the *effective address*.
- The default segments used for the calculation of Physical address (PA) are DS for BX, and SS for BP.

Ex: MOV CX,[BX]+10 ; move DS:BX+10 and DS:BX+11 into CX
 ; PA = DS (shifted left) +BX+10

- Note that, the content of the low address will go into CL and the high address contents will go into CH.
- There are alternative coding: MOV CX,[BX+10], MOV CX,10[BX]
- BX+10 is *effective address*

Ex: MOV AL,[BP]+5 ; PA = SS (shifted left) +BP+5

6. Indexed relative addressing mode:

- Indexed relative addressing mode works the same as the based relative addressing mode.
- Except the registers DI and SI holds the offset address.

Ex: MOV DX,[SI]+5 ;PA=DS(shifted left)+SI+5
 MOV CL,[DI]+20 ;PA=DS(shifted left)+DI+20

7. Based Indexed addressing mode:

- The combination of the based and indexed addressing modes.
- One base register and one index register are used.

Ex: MOV CL,[BX][DI]+8 ;PA=DS(shifted left)+BX+DI+8
 MOV CH,[BX][SI]+20 ;PA=DS(shifted left)+BX+SI+20
 MOV AH,[BP][DI]+12 ;PA=SS(shifted left)+BP+DI+12
 MOV AL,[BP][SI]+29 ;PA=SS(shifted left)+BP+SI+29

- Alternative coding
 MOV CL,[BX+DI+8]
 MOV CL,[DI+BX+8]

Offset Registers for various Segments

Segment register	CS	DS	ES	SS
Offset register(s)	IP	SI, DI, BX	SI, DI, BX	SP, BP

Segment Override:

MOV AL,[BX] DS:BX however MOV AL,ES:[BX]

MOV AX,[BP] SS:BP however MOV AX,DS:[BP]