University of Central Punjab
(Incorporated by Ordinance No. XXIV of 2002 promulgated by Government of the Punjab)
FACULTY OF INFORMATION TECHNOLOGY

Computer Organization and Assembly Language

Lab 04	
Topic	1. Indirect Addressing Mode with variations. 2. Flags (CF,ZF,SF)

Types of Addressing Modes

Direct A fixed offset is given in brackets and the memory at that offset is accessed. For example "mov [1234], ax" stores the contents of the AX registers in two bytes starting at address 1234 in the current data segment. The instruction "mov [1234], al" stores the contents of the AL register in the byte at offset 1234.	• Mov ax,[num1] ;reading • Mov [num2],ax ;writing
Based Register Indirect A base register is used in brackets and the actual address accessed depends on the value contained in that register. For example "mov [bx], ax" moves the two byte contents of the AX register to the address contained in the BX register in the current data segment. The instruction "mov [bp], al" moves the one byte content of the AL register to the address contained in the BP register in the current stack segment.	• Mov bx,var • Mov cx,[bx] • Mov [bx],ax
Indexed Register Indirect An index register is used in brackets and the actual address accessed depends on the value contained in that register. For example "mov [si], ax" moves the contents of the AX register to the word starting at address contained in SI in the current data segment. The instruction "mov [di], ax" moves the word contained in AX to the offset stored in DI in the current data segment.	• Mov si,var1 • Mov di,var2 • Mov [si], ax • Mov [di],bx • Mov cx,[si] • Mov dx,[di]

Based Register Indirect + Offset A base register is used with a constant offset in this addressing mode. The value contained in the base register is added with the constant offset to get the effective address. For example "mov [bx+300], ax" stores the word contained in AX at the offset attained by adding 300 to BX in the current data segment. The instruction "mov [bp+300], ax" stores the word in AX to the offset attained by adding 300 to BP in the current stack segment.	• mov [bx+3], ax • mov cl,[bp+5]
Indexed Register Indirect + Offset An index register is used with a constant offset in this addressing mode. The value contained in the index register is added with the constant offset to get the effective address. For example "mov [si+300], ax" moves the word contained in AX to the offset attained by adding 300 to SI in the current data segment and the instruction "mov [di+300], al" moves the byte contained in AL to the offset attained by adding 300 to DI in the current data segment.	• Mov [si+2],al • Mov bl,[di+4]
Base + Index One base and one index register is used in this addressing mode. The value of the base register and the index register are added together to get the effective address. For example "mov [bx+si], ax" moves the word contained in the AX register to offset attained by adding BX and SI in the current data segment. The instruction "mov [bp+di], al" moves the byte contained in AL to the offset attained by adding BP and DI in the current stack segment. Observe that the default segment is based on the base register and not on the index register. This is why base registers and index registers are named separately. Other examples are "mov [bx+di], ax" and "mov [bp+si], ax." This method can be used to access a two dimensional array such that one dimension is in a base register and the other is in an index register.	• mov [bx+si], ax • mov al,[bp+di]

Base + Index + Offset This is the most complex addressing method and is relatively infrequently used. A base register, an index register, and a constant offset are all used in this addressing mode. The values of the base register, the index register, and the constant offset are all added together to get the effective address. For example "mov [bx+si+300], ax" moves the word contents of the AX register to the word in memory starting at offset attained by adding BX, SI, and 300 in the current data segment. Default segment association is again based on the base register. It might be used with the array base of a two dimensional array as the constant offset, one dimension in the base register and the other in the index register. This way all calculation of location of the desired element has been delegated to the processor.	➢ mov [bx+si+100], ax

Execute every part of Question 1 in ***Nasm with Dosbox*** and observe the memory variables and register values.

Examples

a)

```
1  ; a program to add three numbers using indirect addressing
2  [org 0x100]
3  mov  bx, num1  ; point bx to first number
4  mov  ax, [bx]  ; load first number in ax
5  add  bx, 2     ; advance bx to second number
6  add  ax, [bx]  ; add second number to ax
7  add  bx, 2     ; advance bx to third number
8  add  ax, [bx]  ; add third number to ax
9  add  bx, 2     ; advance bx to result
10 mov  [bx], ax  ; store sum at num1+6
11 mov  ax, 0x4c00 ; terminate program
12 int  0x21
13
14 num1:         dw   5, 10, 15, 0
```

b) Indirect accessing of Word size

```
[org 0x100]

xor ax,ax
xor bx,bx
xor cx,cx
xor dx,dx

mov bx, var1
mov ax,[bx]
mov cx,[bx+2]
mov dx,[bx+4]
add ax,dx
mov [bx+2],ax

mov ax,0x4c00
int 21h

var1: dw 10
var2: dw 20h
var3: dw 13
```

c) Reading and writing in memory through indirect memory address.

```
[org 0x100]
```

```
xor ax,ax
xor bx,bx
xor cx,cx
xor dx,dx


mov bx, var1
mov ax,[bx]
mov cx,[bx+2]
add ax,cx

mov [bx],ax
mov [bx+2],  0
mov dx,[bx+3]
add dx,[bx+3]
mov [bx+3],dx


mov ax,0x4c00
int 21h

var1: dw 60
var2: db 5
var3: dw 100
```

Practice Tasks

Q1.Write a program to solve the following:

Use Indirect addressing mode to access memory variables:

Let

Var1=10
Var2=20
Var3=2
Var4=50
Var5=90

Save the sum of these (using **Indirect** addressing mode) Five variables (Var1+ Var2+ Var3+ Var4+Var5) in ax.

NOTE: Execute the code in sequence.

Q2. Write down the values of Carry Flag, Sign Flag and Zero Flags. Justify your values in flags with explanations.

Run these codes one by one.

a)
```
mov ax,10
mov bx,10
```

sub bx,ax

CF	
SF	
ZF	

Explanation:

b) mov ax,200

mov bx,100

add ax,bx

CF	
SF	
ZF	

Explanation:

c) mov ax,-50

CF	
SF	
ZF	

Explanation:

d) mov ax,100

mov bx,200

sub ax,bx

CF	
SF	
ZF	

Explanation: