Addressing Modes

Addressing Modes

- immediate operand
  - mov ax, 5
  - Every reasonable program needs some data in memory apart from constants
  - Constants cannot be changed, i.e. they cannot appear as the destination operand
  - there must be a mechanism in assembly language to store and retrieve data from memory.

. DATA DECLARATION

- “define byte” written as “db.”
- “define word” or “dw”
- We can associate a symbol with any address that we want to remember and use that symbol in the rest of the code

DIRECT ADDRESSING(ex2)

```
; a program to add three numbers using memory variables
 [org 0x0100]
        mov ax, [num1]         ; load first number in ax
        mov bx, [num2]         ; load second number in bx
        add ax, bx           ; accumulate sum in ax
        mov bx, [num3]         ; load third number in bx
        add ax, bx             ; accumulate sum in ax
        mov [num4], ax         ; store sum in num4
        mov ax, 0x4c00          ; terminate program
 int 0x21
 num1: dw 5
 num2: dw 10
num3: dw 15
 num4: dw 0
```

DIRECT ADDRESSING(ex3)

a program to add three numbers accessed using a single label

```
[org 0x0100]
    mov ax, [num1]          ;load first number in ax
    mov bx, [num1+2]            ; load second number in bx
    add ax, bx              ; accumulate sum in ax
    mov bx, [num1+4]            ; load third number in bx
    add ax, bx              ; accumulate sum in ax
    mov [num1+6], ax            ; store sum at num1+6
    mov ax, 0x4c00          ; terminate program
int 0x21
num1: dw 5
        dw 10
        dw 15
        dw 0
```

DIRECT ADDRESSING(ex4)

```
; a program to add three numbers accessed using a single label
 [org 0x0100]
      mov ax, [num1]        ; load first number in ax
      mov bx, [num1+2]          ; load second number in bx
      add ax, bx              ; accumulate sum in ax
      mov bx, [num1+4]          ; load third number in bx
      add ax, bx              ; accumulate sum in ax
      mov [num1+6], ax          ; store sum at num1+6
      mov ax, 0x4c00        ; terminate program
int 0x21
num1: dw 5, 10, 15, 0
```

DIRECT ADDRESSING(ex5)

```
; a program to add three numbers directly in memory
[org 0x0100]
    mov ax, [num1]          ; load first number in ax
    mov [num1+6], ax        ; store first number in result
    mov ax, [num1+2]        ; load second number in ax
    add [num1+6], ax        ; add second number to result
    mov ax, [num1+4]        ; load third number in ax
    add [num1+6], ax        ; add third number to result
mov ax, 0x4c00          ; terminate program
int 0x21
num1: dw 5, 10, 15, 0
```

ERRORS

- mov [num1+6], [num1] ; ILLEGAL
- “mov ax, bl”
- “mov [num1], [num2]

Add three numbers using byte variables(ex6)

If we change the directive in the last example from DW to DB, the program will still assemble and debug without errors, however the results will not be the same as expected?

```
; a program to add three numbers using byte variables
 [org 0x0100] mov al, [num1]        ; load first number in al
      mov bl, [num1+1]              ; load second number in bl
      add al, bl                    ; accumulate sum in al
      mov bl, [num1+2]              ; load third number in bl
      add al, bl                    ; accumulate sum in al
      mov [num1+3],al               ; store sum at num1+3
mov ax, 0x4c00 ; terminate program
int 0x21
num1: db 5, 10, 15, 0
```

SIZE MISMATCH ERRORS

- The instruction “mov [num1], 5” is legal but there is no way for the processor to know the data movement size in this operation.
- mov byte [num1], 5
- mov word [num1], 5
- The programmer is responsible for accessing the data as word if it was declared as a word and accessing it as a byte if it was declared as a byte

REGISTER INDIRECT ADDRESSING(ex7)

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

REGISTER INDIRECT ADDRESSING(ex8)

```
; a program to add ten numbers
[org 0x0100]
    mov bx, num1 ; point bx to first number
    mov cx, 10 ; load count of numbers in cx
    mov ax, 0 ; initialize sum to zero
    l1: add ax, [bx] ; add number to ax
    add bx, 2 ; advance bx to next number
    sub cx, 1 ; numbers to be added reduced
    jnz l1 ; if numbers remain add next
    mov [total], ax ; write back sum in memory
mov ax, 0x4c00 ; terminate program
int 0x21
num1: dw 10, 20, 30, 40, 50, 10, 20, 30, 40, 50
total: dw 0
```

REGISTER + OFFSET ADDRESSING

- ; a program to add ten numbers using register + offset addressing
- [org 0x0100]

```
mov bx, 0 ; initialize array index to zero
mov cx, 10 ; load count of numbers in cx
mov ax, 0 ; initialize sum to zero
l1: add ax, [num1+bx] ; add number to ax
add bx, 2 ; advance bx to next index
sub cx, 1 ; numbers to be added reduced
jnz l1 ; if numbers remain add next
mov [total], ax ; write back sum in memory
```

- mov ax, 0x4c00 ; terminate program
- int 0x21
- num1: dw 10, 20, 30, 40, 50, 10, 20, 30, 40, 50 total: dw 0

Class Activity(SEGMENT ASSOCIATION and ADDRESS WRAPAROUND)

- For example if BX=0100, SI=0200, and CS=1000 and the memory access under consideration is [cs:bx+si+0x0700], what would be the effective address?
- BX=9100, DS=1500 and the access is [bx+0x7000], is there is wraparound?

Segment Override Prefix

- Instruction opcode
- Mov ax ,[cs:bx] 2E8B07
- Mov ax ,[es:bx] 268B07
- Mov ax ,[ss:bx] 368B07
- Mov ax, [bx] 8B07