Classic Knowledge on x86 Addressing Modes

by : Antonius (@ringlayer)

http://www,ringlayer.net – https://ringlayer.wordpress.com

https://github.com/ringlayer?tab=repositories

8086 Addressing Modes

It's very important to understand addressing mode before understanding instruction sets. Here we provide several data addressing mode examples.
What we given here is not a complete list, since the purpose is just for understanding x86 instruction syntaxs:

- Immediate Addressing Mode
- Direct Offset / Displacement Addressing Mode
- Register Direct Addressing Mode
- Register Indirect Addressing Mode
- Indexed Addressing Mode
- Base Index Addressing Mode
- Base Index and Displacement Data Addressing Mode

Immediate Addressing Mode

Using immediate addressing mode means the data or operand immediately included in instruction. Examples:

```
mov ax,0h
mov al, 1b
mov ah, 1b
```

For example "mov ax, 11b" will move immediate value : 11 binary to ax register, those 2 bit will be moved to al register. So basically it's the same instruction as "mov al, 11b".
Before mov on 8 bit register al, when ah = 0h and al = 0h, before mov ax,11b :

```
al 8 bit register:
[0][0][0][0][0][0][0][0]
```

After mov ax,11b:

```
al 8 bit register:
[0][0][0][0][0][0][1][1]
```

Another example when ah = 0h and al=0h : "mov ax, 100h". Since 100h in binary is "100000000b" it will overflow 8 bit register al, hence the first bit : "1b" will be placed on ah register.
Before mov ax,100h:

```
al (accumulator low) 8 bit register :
[0][0][0][0][0][0][0][0]
```

```
ah (accumulator high) 8 bit register:
[0][0][0][0][0][0][0][0]
```

After "mov ax, 100h" :

```
al (accumulator low) 8 bit register :
[0][0][0][0][0][0][0][0]

ah (accumulator high) 8 bit register:
[0][0][0][0][0][0][0][1]
```

Direct Offset / Displacement Addressing Mode

Using direct addressing, we directly load an offset address of variable to a register.

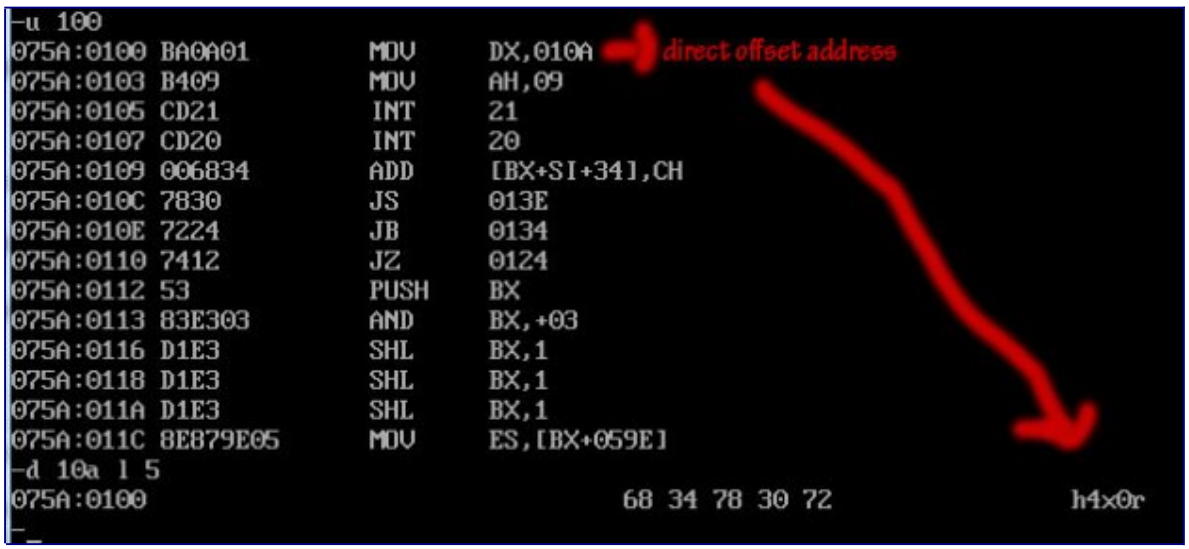
Example:

```
mov dx, ds:[10ah]
mov dx, offset variable
```

For example:

```
.model tiny
.data
    hax db 'h4x0r$'
.code
org 100h
start:
    mov dx, offset hax; -> direct offset address addressing
    mov ah, 9h
    int 21h
    int 20h
end start
```

Using direct offset addressing, we directly put offset address of string hax to dx register.
The rule is that if no segment register specified, default segment register that will be used is ds, and as any other transfer instruction operand' maximal size is size of destination.



We can see we directly transfer offset address of hax string which at offset address 10ah to dx register, if we dump 10ah about 5 bytes we can see it contains string "h4x0r" which previously defined before.

We can also use memory address as displacement (on masm we use disp keyword, in this example we use tasm without disp keyword):

```
;displacement mode sample
;simple code by : Antonius (sw0rdm4n)
;www.ringlayer.net
.model tiny
.data
        binx db 41h
.code
org 100h
start:
        mov ah,2h
        mov dx, ds:[10ah]
        int 21h
        int 20h
end start
```

"mov dx, ds:[10ah]" will move byte at address ds:10ah to register dx.

Register Direct Addressing Mode

Direct register addressing mode will use directly content of an operand register. Examples:

```
mov ax, bx; move bytes content of bx register to ax register
mov dl,al; move bytes content of al register to dl register
```

The rule is register size must be the same. If source is 8 bit regsiter, destination must be also 8 bit register. You won't put operand size larger than destination register.

Register Indirect Data Addressing Mode

Using register indirect addressing to access data means we use a register as a pointer to a memory address contains specific bytes.
Examples:

```
    mov dx, [bx]
    mov dx, cs:[bx] ; for different code segment
```

For example once instruction "mov dx, [bx]" executed, what happens is microprocessor will check offset address that is in bx register, for example bx register contains : "10e".
Microprocessor will suppose it as an offset address, Microprocessor then will fetch 2 bytes on offset 10eh and move it to dx register.

Example code:

```
;indirect register example
;made by sw0rdm4n
;www.ringlayer.net
.model tiny
.data
        hax db 01000001b
.code
org 100h
start:
        mov bx, 10eh
        mov dx, [bx]
        mov ah,02h
        int 21h
        mov dx, offset hax
```

```
        int 20h
end start
```

Actually instruction "mov dx, offset hax" is a junk, this is just for debugging purpose, just like printf(on user space c codes) and printk (on kernel space c codes).
At first "mov bx, 10eh" this instruction will move 10eh. 10eh will be supposed as offset address of variable hax.
mov dx, [bx], in this instruction, processor will assume bx as offset address pointer. Next, microprocessor will check offset address that recorded on bx register, within next sequece, microprocessor will get 2 bytes from 10eh and move it to dx register.

Indexed Addressing Mode

This mode uses special purpose index register such as si and di.

Examples:

```
     call cs:[di]
     add byte ptr cs:[di], 1b
     mov bx, cs:[si]
     mov al, byte ptr cs:[di]
     mov byte ptr cs:[di], al
     mov byte ptr cs:[di], 1b
     and byte ptr cs:[si], 11111111b
```

The possible combinations :

```
cs:  |
ds:  |    si / di
ss:  |
es:  |
```

If no segment register specified, by default ds will be used.

Example code:

```
;simple index addresing mode example
;made by Antonius (@sw0rdm4n)
;http://ringlayer.net
;compile with tasm 2.0 and tlink 3.0
.model tiny
.data
        sayhawatpu db 73h,61h,79h,68h,61h,77h,61h,74h,70h,75h ; string
sayhawatpu
        indexme db 10  dup (?)
 .code
 org 100h
 main:
        mov si, offset sayhawatpu
        xor cx, cx
        call _setvid
looper:
        mov bl, byte ptr cs:[si]; indexed addressing mode example
        mov byte ptr[indexme], bl
        mov dl, byte ptr[indexme]
        call _printout
        inc cx
        inc si
        cmp cx, 1010b
        jl looper
```

```
        int 20h
_printout proc near
        mov ah, 2h
        int 21h
        ret
_printout endp
_setvid proc near
        mov al, 0h
        mov ah,00h
        int 10h
        ret
_setvid endp
end main
```

On routine : "mov bl, byte ptr cs:[si]" it will move byte that pointed by offset address which recorded in source index.

What above codes did is simple, just print string "sayhawatpu" in vga video mode.

Base Indexed Addressing Mode

This addressing to access data uses combination of base register (bx and bp) and index register (si and di) to acquire data in memory.

Examples:

```
and cs:[bx+di], bx
jmp cs:[bx+di]
add byte ptr cs:[bx+di],1h
mov cs: [bx+di],100h
mov cs: [bp+si],bx
mov bx, cs: [bp+di]
mov bl, byte ptr cs:[bp+di]
```

The possible combinations :

```
cs:    |                |
ds:    |   bx+ / bp+    | si / di
ss:    |                |
es:    |                |
```

Example code:

```
;simple base and index addresing mode example
;made by Antonius (@sw0rdm4n)
;http://ringlayer.net
;compile with tasm 2.0 and tlink 3.0
.model tiny
.data
        ibmbio db
73h,75h,64h,61h,68h,20h,61h,64h,61h,20h,76h,61h,6bh,73h,69h,6eh ; sudah ada
vaksin
        indexme db 16  dup (?)
```

```
 .code
 org 100h
 main:
        mov si, offset ibmbio
        xor cx, cx
        call _setvid
looper:
        mov bp, 0h
        mov bl, byte ptr cs:[bp + si]; base and indexed addressing mode example
        mov byte ptr[indexme], bl
        mov dl, byte ptr[indexme]
        call _printout
        inc cx
        inc si
        cmp cx, 10000b
        jl looper
        int 20h
_printout proc near
        mov ah, 2h
        int 21h
        ret
_printout endp
_setvid proc near
        mov al, 13h
        mov ah,00h
        int 10h
        ret
_setvid endp
end main
```

On routine "mov bl, byte ptr cs:[bp + si]", we see base and indexed addressing mode example. What it does is moving a byte that pointed by offset address which
a result of calculation of bp + si into bl register.
bl register.

And again, what it does is simply print string in vga mode : "sudah ada vaksin"

Base Index and Displacement Data Addressing Mode

This mode is combination of indexed, base and displacement data addressing mode. Displacement can be 8 bit or 16 bit depends on the size of destination and purpose of routine.
The possible combinations :

```
cs:  |       |           |
ds:  | bx+ / | si+ / di+ | displacement
ss:  | bp+   |           |
es:  |       |           |
```

Examples:

```
mov dl, [bp + si + 1]
```

```
mov dx, [bx + si + 10h]
```

Example code:

```
;simple base and index addresing with displacement example
;made by Antonius (@sw0rdm4n)
;http://ringlayer.net
;compile with tasm 2.0 and tlink 3.0
.model tiny
.data
        ibmbio db 68h,61h,78h ; hax
        indexme db 3  dup (?)
 .code
 org 100h
 main:
        mov si, offset ibmbio
        call _setvid

        mov bp, 0h
        mov bl, byte ptr cs:[bp + si + 0h]; base indexed + displacement example
        mov byte ptr[indexme], bl
        mov dl, byte ptr[indexme]
        call _printout

        mov bp, 1h
        mov bl, byte ptr cs:[bp + si + 0h]; base indexed + displacement example
        mov byte ptr[indexme], bl
        mov dl, byte ptr[indexme]
        call _printout

        mov bp, 0h
        mov bl, byte ptr cs:[bp + si + 2h]; base indexed + displacement example
        mov byte ptr[indexme], bl
        mov dl, byte ptr[indexme]
        call _printout
        int 20h
_printout proc near
        mov ah, 2h
        int 21h
        ret
_printout endp
_setvid proc near
        mov al, 13h
        mov ah,00h
        int 10h
        ret
_setvid endp
end main
```

For example on this routine :"mov bl, byte ptr cs:[bp + si + 2h]" , this means to move a byte that's located from offset address calculation of bp + si + 2h