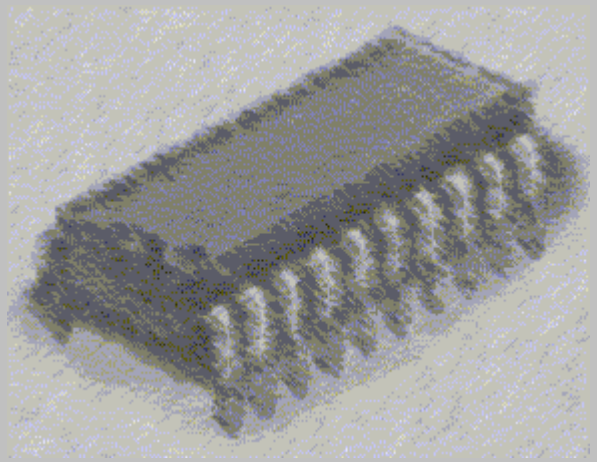
Lesson 3. Segmentation and addressing modes



Computer Structure and Organization

Graduated in Computer Sciences /
Graduated in Computer Engineering

Automatic Department

Contents

- Segmentation
- Addressing modes
- Relative addressing mode
- Shift instructions
- Logic instructions
- Accessing to a string character using:
  - Relative to base register addressing mode
  - Relative to index register addressing mode

i80x86 memory segmentation

- i8086 microprocessor has fourteen 16 bit registers. 16 bits allow to access to a maximum of 2^{16} = 64 K
- i8086 can access to 1 MB = 2^{20} memory size by dividing it in 64 K segments
- Two different informations must be known by program: data and code segment address and the offset inside it
- Phisical address can be easily calculate:

  BaseRegister x 10h + offset

- Segment registers are:
  - CS: code segment
  - DS: data segment
  - SS: stack segment
  - ES: extra segment (data)

Addressing modes

Addressing modes		μP 8086/88	Examples
Inmediate		Inmediate	MOV AX,15h
Direct	To register	Register	MOV AX,BX
	To memory	(with segmentation)	
	To page	Direct	MOV CX,ETIQUETA
Relative	To program counter	(for jumps only)	
	To register	Relative to base register	MOV [BX]+ARTÍCULO,AL
	To index register	Relative to index register	MOV DL,VECTOR[SI]
		Relative to index and base registers	MOV AH,[BX][SI]+ARRAY
	To stack	(relative to stack)	
Indirect		(it doesn't exist)	
Implícit		Some instructions only	

Relative addressing mode

- It's used to point to memory positions inside a segment
- Base and Index registers are used. DS is the register segment used when using BX. SS is used such us register segment if BP is used.
- i8086 has different relative addressing modes: relative to base register, relative to index register, relative to index and base and relative to offset, index and base registers.
- **Examples:**
  - ADD DX, [BX]
  - MOV DL, [SI]
  - SUB AX,[BP][SI]
  - XOR red[BX][DI], DX
  - AND AL,[BP+8]

Shift instructions (I)

- The whole of them have the same format
- **Format:** *ShiftInstruction* target, times
- **Description:**
- Shift specified target as many times as is indicated on the instruction. CL must store the number of times to shift target if it's more than one time.
- It's a local shift when using 8 bits registers (upper or lower bits of AX, BX, CX or DX registers)
- Shifted bits are copied on carry flag.
- i80x86 uses complement 2 representation systems. Therefore, arithmetic shifts will be performed in this representation system.
- Allowed shifts are: atithmetics, logicals, rotations and rotations through the carry flag to left or to right side.

Shift instructions (II)

- **Mnemonic:** SAL
- **Format:** SAL target, times
- **Description:** Specified target will be aritmetic shifted to left as many bits as times parameter indicates and filling with 0's by the right side.

- **Examples:**
  - MOV AX, 2 ; AX = 2 → 0000...00010
  - SAL AX, 1 ; AX = 4 → 0000...00100
  - MOV CL, 2 ; CL = 2
  - SAL AX, CL ; AX = 16 → 0000...10000

Shift instructions (III)

- **Mnemonic:** SAR
- **Format:** SAR target, times
- **Description:** Specified target will be aritmetic shifted to right as many bits as times parameter indicates and filling with MSB by the left side.

- **Examples:**
  - MOV AX, -16 ; AX = FFF0h (-16) → 1111...10000
  - SAR AX, 1 ; AX = FFF8h (-8) → 1111...11000
  - MOV CL, 2 ; CL = 2
  - SAR AX, CL ; AX = FFFEh (-2) → 1111...11110

Shift instructions (IV)

- **Mnemonic:** SHL
- **Format:** SHL target, times
- **Description:** Specified target will be aritmetic shifted to left as many bits as times parameter indicates and filling with 0's by the right side.

- **Examples:**
  - MOV AX, 0FFFFh ; AX = FFFFh (-1) → 1111...1111
  - SHL AX, 1 ; AX = FFFEh (-2) → 1111...1110
  - MOV CL, 2 ; CL = 2
  - SHL AX, CL ; AX = FFF8h (-8) → 1111...1000

Shift instructions (V)

- **Mnemonic:** SHR
- **Format:** SHR target, times
- **Description:** Specified target will be aritmetic shifted to right as many bits as times parameter indicates and filling with 0's by the left side.

- **Examples:**
  - MOV AX, -8 ; AX = FFF8h (-8) → 1111...11000
  - SHR AX, 1 ; AX = 7FFCh (32764) → 0111...11100
  - MOV CL, 2 ; CL = 2
  - SHR AX, CL ; AX = 1FFFh (8191) → 0011...11111

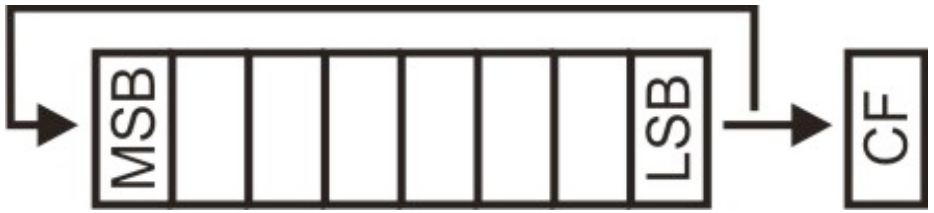
Shift instructions (VI)

- **Mnemonic:** ROL
- **Format:** ROL target, times
- **Description:** Specified target will be rotated to left as many bits as times parameter indicates. Outcoming bits will be introduced by the right side.

- **Examples:**
  - MOV AX, 8000h ; AX = 8000h → 1000...0000
  - ROL AX, 1 ; AX = 0001h → 0000...0001
  - MOV CL, 2 ; CL = 2
  - ROL AX, CL ; AX = 0004h → 0000...0100

Shift instructions (VII)

- **Mnemonic:** ROR
- **Format:** ROR target, times
- **Description:** Specified target will be rotated to right as many bits as times parameter indicates. Outcoming bits will be introduced by the left side.



- **Examples:**
  - MOV AX, 8001h ; AX = 8001h → 1000...0001
  - ROR AX, 1 ; AX = C000h → 1100...0000
  - MOV CL, 2 ; CL = 2
  - ROR AX, CL ; AX = 3000h → 0011...0000

Shift instructions (VIII)

- **Mnemonic:** RCL
- **Format:** RCL target, times
- **Description:** Specified target will be rotated to left throught carry flag as many bits as times parameter indicates. Outcoming bits will be introduced by the right side.

- **Examples:**
  - MOV AX, 8000h ; AX = 8000h y CF = 1 → 1000...0000 CF=1
  - RCL AX, 1 ; AX = 0001h → 0000...0001 CF=1
  - MOV CL, 2 ; CL = 2
  - RCL AX, CL ; AX = 0006h → 0000...0110 CF=0

Shift instructions (and IX)

- **Mnemonic:** RCR
- **Format:** RCR target, times
- **Description:** Specified target will be rotated to right throught carry flag as many bits as times parameter indicates. Outcoming bits will be introduced by the left side.

- **Examples:**
  - MOV AX, 8001h ; AX = 8001h y CF=1 → 1000...0001 CF=1
  - ROR AX, 1 ; AX = C000h → 1100...0000 CF=1
  - MOV CL, 2 ; CL = 2
  - ROR AX, CL ; AX = 7000h → 0111...0000 CF=0

Logical instructions (I)

- **Mnemonic:** AND
- **Format:** AND target, source
- **Description:**
- AND function will be performed with source and target parameters.
- Source and target must be of the same size

a	b	a AND b
0	0	0
0	1	0
1	0	0
1	1	1

Examples:

- AND AX, BX ; IF AX = 7777h y BX = 2222h then
  ; AX AND BX = 2222h
- AND AX, 1 ; IF AX = 4 then AX AND 1 = 0
- AND AX, 0FFFFh; Whatever AX is , AX AND 0FFFFh=AX

Logical instructions (II)

- **Mnemonic:** OR
- **Format:** OR target, source
- **Description:**
- OR function will be performed with source and target parameters.
- Source and target must be of the same size

a	b	a OR b
0	0	0
0	1	1
1	0	1
1	1	1

- **Examples:**
  - OR AX, BX ; IF AX = 7777h y BX = 2222h then AX ; OR BX = 7777h
  - OR AX, 1 ; IF AX = 4 then AX OR 1 = 5
  - OR AX, 0h ; Whatever AX is, AX OR 0h = AX

Logical instructions (III)

- **Mnemonic:** XOR
- **Format:** XOR target, source
- **Description:**
- XOR function will be performed with source and target parameters.
- Source and target must be of the same size

a	b	a XOR b
0	0	0
0	1	1
1	0	1
1	1	0

- **Examples:**
  - XOR AX, BX ; IF AX = 7777h y BX = 2222h thenAX XOR ; BX = 5555h
  - XOR AX, 1 ; IF AX = 4 then AX XOR 1 = 5
  - XOR AX, 0FFFFh; IF AX = 7777h AX XOR 0FFFFh = 8888h

Logical instructions (and IV)

- **Mnemonic:** NOT
- **Format:** NOT target
- **Description:**
- NOT function will be performed with target parameters.

a	NOT a
0	1
1	0

- **Example:**
  - NOT AX ; Si AX = 7777h entonces NOT AX= 8888h

String access by using relative to base register example

```
DOSSEG
.MODEL SMALL
.STACK 100h
.DATA
numeros  DB 1,2,3,4,5,6 ; String numbers

.CODE
Inicio:
MOV AX, @DATA
MOV DS, AX

LEA BX, numeros      ; DS:BX string address

MOV CX, 6            ; Number of times

Bucle:
  MOV DL, [BX]       ; Memory position is
                     ; stored on DL
  ADD DL, 7          ; We add 7 to it
  MOV [BX], DL       ; Store new value on
                      ;current memory
                      ; position
  INC BX             ; BX is incremented by 1
LOOP Bucle
MOV AH, 4Ch          ; program end service
INT 21h              ; requested

END Inicio
```

Program adds 7 to each number of the string

String access by using relative to index register example

```
DOSSEG
.MODEL SMALL
.STACK 100h
.DATA
numeros  DB  1,2,3,4,5,6; String numbers

.CODE
Inicio:
MOV AX, @DATA

MOV DS, AX

MOV  SI, 0      ; SI is initializated to 0
MOV  CX, 6      ; Number of times

Bucle:
    ADD numeros[SI], 7
    INC SI
LOOP Bucle
MOV AH, 4Ch     ; Program end service
INT 21h         ; requested

END Inicio
```

Program adds 7 to each number of the string