Tema 5.2:
Segmentación y modos de direccionamiento

- Segmentación
- Modos de direccionamiento
- Direccionamiento relativo
- Instrucciones de desplazamiento
- Instrucciones lógicas
- Ejemplo de acceso a una cadena mediante:
  - Direccionamiento relativo a base
  - Direccionamiento relativo mediante índice

Bibliografía básica

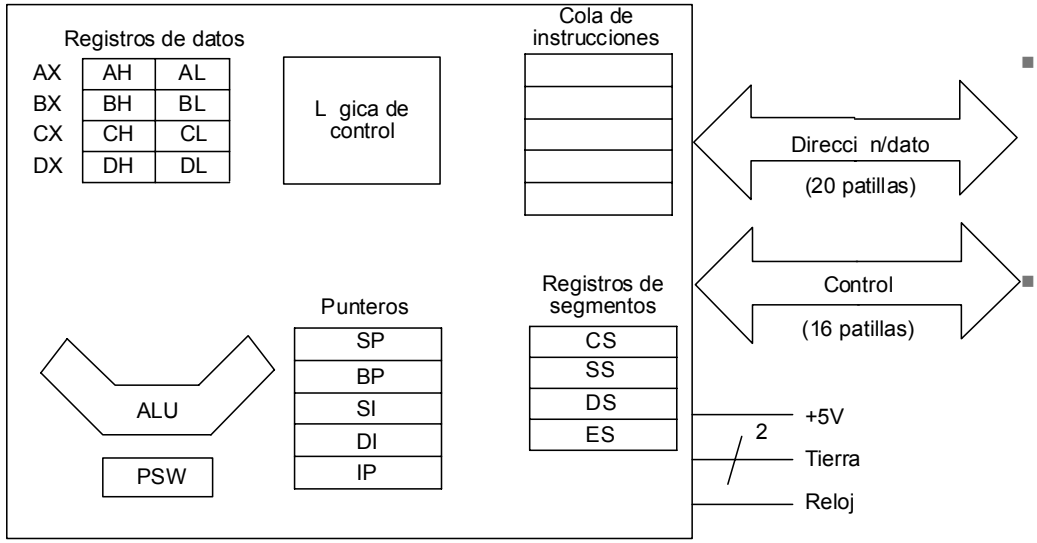
- 8088-8086/8087 programación ensamblador en entorno MS-DOS
  Miguel Angel Roselló.
  Ed. Anaya Multimedia

- Microprocesadores: el 8088 / 86
  Fernando Remiro Domínguez
  Agustín Martín García
  Ed. Akal-Bibiloteca tecnológica

- Lenguajes ensambladores
  R. Martínez Tomás. Ed. Paraninfo

- Lenguaje ensamblador de los 80x86
  Jon Beltrán de Heredia
  Editorial Anaya-Multimedia. 1996

Segmentación del i8086



- El microprocesador 8086 tiene catorce registros de 16 bits. Con 16 bits se puede acceder a 2^{16} o lo que es igual a 64 K
- El 8086 emplea un truco para acceder a 1 MB = 2^{20}
- El truco consiste en dividir el mega en trozos de 64 K que llama segmentos
- El programa en todo momento debe conocer en qué segmento están los datos o el código y cuál es la posición dentro del segmento

- El cálculo de la dirección física lo realiza según:
  - R.Base x 10h + desplazamiento
- Los registros de segmento son
  - CS: para el segmento de código
  - DS: para el segmento de datos
  - SS: para el segmento de pila
  - ES: segmento extra de datos
- Gracias a la segmentación se facilita la multiprogramación y existen zonas diferentes para el código, para los datos y para la pila

Modos de direccionamiento

Modos de direccionamiento			μP 8086/88	Ejemplos
Inmediato			Inmediato	MOV AX, 15H
	Directo	De registro	A registro	MOV AX, BX
		De memoria	(con segmentación)	
		De página base	Directo	MOV CX, ETIQUETA
	Relativo	Al contador de programa	(sólo saltos)	
		A un registro	Relativo a base	MOV [BX]+ARTÍCULO, AL
		A un registro índice	Mediante índice	MOV DL, VECTOR[SI]
			Mediante índice y base	MOV AH, [BX][SI]+ARRAY
		A pila	(Relativo a pila)	
Indirecto			(No existe)	
Implícito			Algunas instrucciones	

Direccionamiento relativo

- Se emplea para apuntar a direcciones de memoria dentro de un segmento
- Se emplean registros Base y registros Índices. Si BX se emplea como registro base, entonces el registro de segmento que se emplea es el registro DS. Si es BP el registro base, entonces el registro de segmento empleado es el de la pila SS
- Existen diferentes modos de direccionamiento indirecto en el 8086:

Ejemplos:

- ADD DX, [BX]
- MOV DL, [BP+6]
- XOR rojo[BX], D

Modo de direccionamiento	Sintaxis	Descripción.
Indirecto a registro	[BX] [BP][DI][SI]	Los registros contienen la dirección efectiva .
Relativo a base	desplaz[BX] desplaz[BP]	La dirección efectiva se calcula a partir del registro y se le suma el desplazamiento.
Mediante índice	desplaz[DI] desplaz[SI]	
Mediante índice y base	[BX][DI] [BX][SI] [BP][DI] [BP][SI]	La dirección efectiva se calcula a partir del registro base y el registro índice.
Mediante índice y base con desplazamiento.	Desp[BX][DI] Desp[BX][SI] Desp[BP][DI] Desp[BP][SI]	La dirección efectiva se calcula a partir del registro base, del índice y del desplazamiento.

Instrucciones de desplazamiento (I)

- **Nombre:** **SAL**
- **Formato:** **SAL destino, contador**
- **Descripción:**

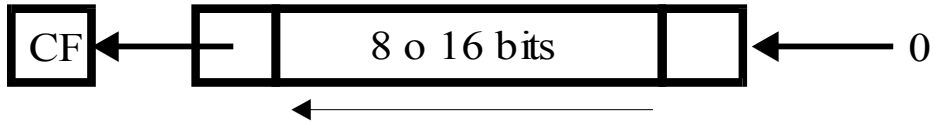
  Desplazará a la izquierda el operando destino tantos bits como indique el valor del contador, rellenando con ceros los bits que quedan libres por la derecha

  Si el valor de contador es mayor que uno, obligatoriamente deberemos especificarlo en el registro CL

  Si especifico un desplazamiento sobre un registro de ocho bits, la parte alta o la parte baja de los registros AX, BX, CX o DX, el desplazamiento será local a esos ocho bits. Copia el contenido del bit más significativo en el flag de acarreo

- **Ejemplos:**

```
MOV AX, 2        ; AX = 2
SAL  AX, 1       ; AX = 4
MOV CL, 2        ; CL = 2
SAL  AX, CL      ; AX = 16
```



Instrucciones de desplazamiento (II)

- **Nombre:** **SAR**
- **Formato:** **SAR destino, contador**
- **Descripción:**

  El formato de la instrucción es SAR destino, contador

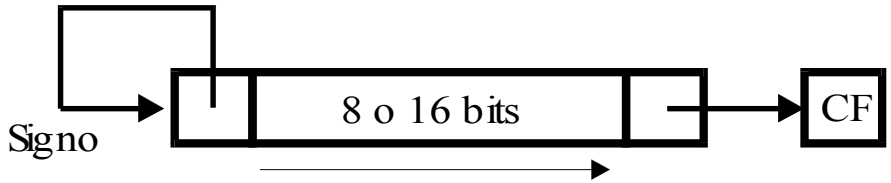
  Desplazará a la derecha el operando destino tantos bits como indique el valor del contador, rellenando con el bit de signo los bits que quedan libres por la izquierda

  Si el valor de contador es mayor que uno, obligatoriamente deberemos especificarlo en el registro CL

  Si especifico un desplazamiento sobre un registro de ocho bits, la parte alta o la parte baja de los registros AX, BX, CX o DX, el desplazamiento será local a esos ocho bits. Copia el contenido del bit menos significativo en el flag de acarreo

- **Ejemplos:**

```
MOV AX, 16      ; AX = 16
SAR AX, 1       ; AX = 8
MOV CL, 2       ; CL = 2
SAR AX, CL      ; AX = 2
```



Instrucciones de desplazamiento (III)

- **Nombre:** **SHL**
- **Formato:** **SHL destino, contador**
- **Descripción:**

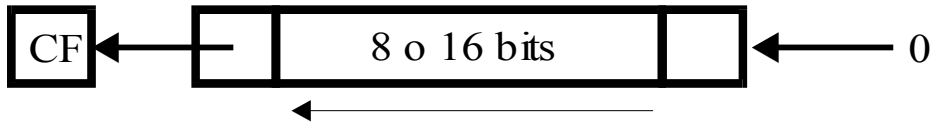
  Desplazará a la izquierda el operando destino tantos bits como indique el valor del contador, rellenando con ceros los bits que quedan libres por la derecha. Es un desplazamiento lógico. *El comportamiento es igual que el de la instrucción SAL*

  Si el valor de contador es mayor que uno, obligatoriamente deberemos especificarlo en el registro CL

  Si especifico un desplazamiento sobre un registro de ocho bits, la parte alta o la parte baja de los registros AX, BX, CX o DX, el desplazamiento será local a esos ocho bits. Copia el contenido del bit más significativo en el flag de acarreo

- **Ejemplos:**

```
MOV AX, 0FFFFh ; AX = FFFFh  (-1)
SHL AX, 1      ; AX = FFFEh (-2)
MOV CL, 2      ; CL = 2
SHL AX, CL     ; AX = FFF8h (-8)
```



Instrucciones de desplazamiento (IV)

- **Nombre:** **SHR**
- **Formato:** **SHR destino, contador**
- **Descripción:**

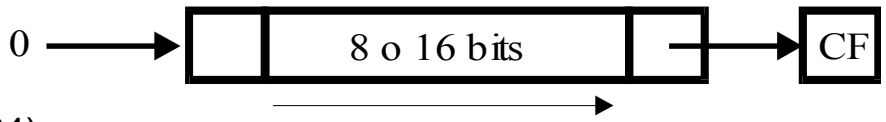
  Desplazará a la derecha el operando destino tantos bits como indique el valor del contador, rellenando con ceros los bits que quedan libres por la izquierda. Es un desplazamiento lógico

  Si el valor de contador es mayor que uno, obligatoriamente deberemos especificarlo en el registro CL

  Si especifico un desplazamiento sobre un registro de ocho bits, la parte alta o la parte baja de los registros AX, BX, CX o DX, el desplazamiento será local a esos ocho bits. Copia el contenido del bit menos significativo en el flag de acarreo

- **Ejemplos:**

```
MOV AX, 2       ; AX = FFF8h (-8)
SHR AX, 1       ; AX = 7FFCh (32764)
MOV CL, 2       ; CL = 2
SHR AX, CL      ; AX = 1FFFh (8191)
```



Instrucciones de desplazamiento (V)

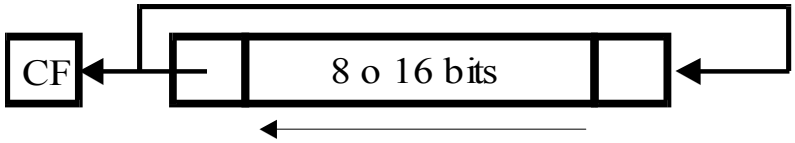
- **Nombre:** **ROL**
- **Formato:** **ROL destino, contador**
- **Descripción:**

  Rota los bits del operando destino hacia la izquierda tantos bits como indique el valor del contador, rellenando con los bits que saldrían por la izquierda los bits que quedan libres por la derecha

  Si especifico una rotación sobre un registro de ocho bits, la parte alta o la parte baja de los registros AX, BX, CX o DX, la rotación será local a esos ocho bits. Copia el contenido del bit más significativo en el flag de acarreo

- **Ejemplos:**

```
MOV AX, 8000h   ; AX = 8000h
ROL AX, 1       ; AX = 0001h
MOV CL, 2       ; CL = 2
ROL AX, CL      ; AX = 0004h
```



Instrucciones de desplazamiento (VI)

- **Nombre:** **ROR**
- **Formato:** **ROR destino, contador**
- **Descripción:**

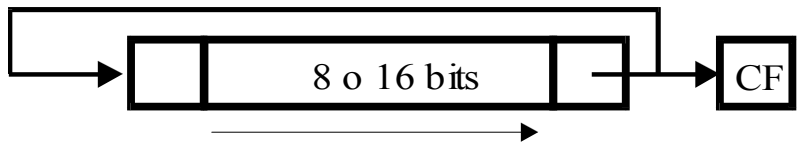
  Rota los bits del operando destino hacia la derecha tantos bits como indique el valor del contador, rellenando con los bits que saldrían por la derecha los bits que quedan libres por la izquierda

  Si deseamos rotar el operando una única vez a la derecha, se puede especificar directamente. Si el valor de contador es mayor que uno, obligatoriamente deberemos especificarlo en el registro CL

  Si especifico una rotación sobre un registro de ocho bits, la parte alta o la parte baja de los registros AX, BX, CX o DX, la rotación será local a esos ocho bits. Copia el contenido del bit menos significativo en el flag de acarreo

- **Ejemplos:**

```
MOV AX, 8001h   ; AX = 8001h
ROR AX, 1       ; AX = C000h
MOV CL, 2       ; CL = 2
ROR AX, CL      ; AX = 300
```



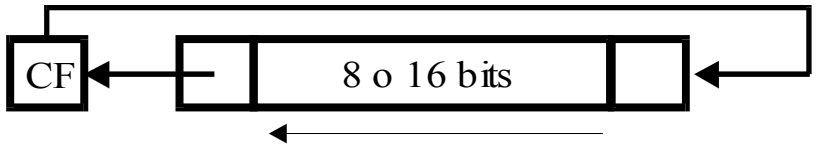
Instrucciones de desplazamiento (VII)

- **Nombre:** **RCL**
- **Formato:** **RCL destino, contador**
- **Descripción:**

  Rota los bits del operando destino hacia la izquierda a través del flag de acarreo tantos bits como indique el valor del contador

  Si deseamos rotar el operando una única vez a la izquierda, se puede especificar directamente. Si el valor de contador es mayor que uno, obligatoriamente deberemos especificarlo en el registro CL

  Si especifico una rotación sobre un registro de ocho bits, la parte alta o la parte baja de los registros AX, BX, CX o DX, la rotación será local a esos ocho bits



- **Ejemplos:**

```
MOV AX, 8000h   ; AX = 8000h y además suponemos el flag de acarreo a 1
RCL AX, 1       ; AX = 0001h
MOV CL, 2       ; CL = 2
RCL AX, CL      ; AX = 0006h y el flag de acarreo estará a 0
```

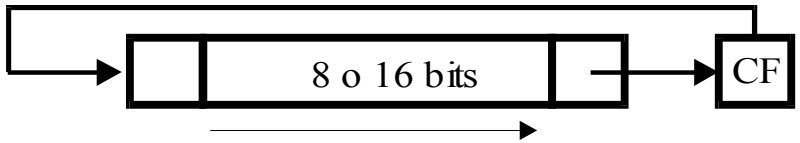
Instrucciones de desplazamiento (VIII)

- **Nombre:** **RCR**
- **Formato:** **RCR destino, contador**
- **Descripción:**

  Rota los bits del operando destino hacia la derecha tantos bits como indique el valor del contador a través del flag de acarreo

  Si deseamos rotar el operando una única vez a la derecha, se puede especificar directamente. Si el valor de contador es mayor que uno, obligatoriamente deberemos especificarlo en el registro CL

  Si especifico una rotación sobre un registro de ocho bits, la parte alta o la parte baja de los registros AX, BX, CX o DX, la rotación será local a esos ocho bits



- **Ejemplos:**

```
MOV AX, 8001h   ; AX = 8001h y suponemos el flag de acarreo a 1
ROR AX, 1       ; AX = C000h
MOV CL, 2       ; CL = 2
ROR AX, CL      ; AX = 7000h y el flag de acarreo estará a 0
```

Instrucciones lógicas (I)

- **Nombre:** **AND**
- **Formato:** **AND destino, origen**
- **Descripción:**

  Realiza a nivel de bits la operación lógica AND entre el origen y el destino

  Tanto el origen como el destino pueden ser operandos de 8 o de 16 bits, pero ambos del mismo tamaño

- **Ejemplos:**

```
AND AX, BX      ; Si AX = 7777h y BX = 2222h entonces AX AND BX = 2222h
AND AX, 1       ; AX = 4 entonces AX AND 1 = 0
AND AX, 0FFFFh  ; Sea cual sea el valor de AX, AX AND 0FFFFh = AX
```

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

Instrucciones lógicas (II)

- **Nombre:** **OR**
- **Formato:** **OR destino, origen**
- **Descripción:**

  Realiza a nivel de bits la operación lógica OR entre el origen y el destino

  Tanto el origen como el destino pueden ser operandos de 8 o de 16 bits, pero ambos del mismo tamaño

- **Ejemplos:**

```
OR AX, BX       ; Si AX = 7777h y BX = 2222h entonces AX OR BX = 7777h
OR AX, 1        ; Si AX = 4 entonces AX OR 1 = 5
OR AX, 0h       ; Sea cual sea el valor de AX, AX OR 0FFFFh = AX
```

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

Instrucciones lógicas (III)

- **Nombre:** **XOR**
- **Formato:** **XOR destino, origen**
- **Descripción:**

  Realiza a nivel de bits la operación lógica XOR entre el origen y el destino

  Tanto el origen como el destino pueden ser operandos de 8 o de 16 bits, pero ambos del mismo tamaño

- **Ejemplos:**

```
XOR AX, BX      ; Si AX = 7777h y BX = 2222h entonces AX XOR BX = 5555h
XOR AX, 1       ;  Si AX = 4 entonces AX XOR 1 = 5
XOR AX, 0FFFFh ; Si AX = 7777h AX XOR 0FFFFh = 8888h
```

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

Instrucciones lógicas (IV)

- **Nombre:** **NOT**
- **Formato:** **NOT destino**
- **Descripción:**

  Realiza a nivel de bits la operación lógica NOT del destino

  El destino pueden ser operandos de 8 o de 16 bits, pero ambos del mismo tamaño

- **Ejemplos:**

  NOT AX ; Si AX = 7777h entonces NOT AX= 8888h

a	NOT a
0	1
1	0

Ejemplo de acceso a una cadena mediante direccionamiento relativo a base

```
DOSSEG
.MODEL SMALL
.STACK 100h
.DATA
numeros  DB 1,2,3,4,5,6 ; Números de la cadena.
.CODE
Inicio:
MOV AX, @DATA        ; Indicamos donde se
MOV DS, AX           ; encuentran los datos

LEA BX, numeros ; En DS:BX dirección cadena
MOV CX, 6       ; Indicamos el número de dígitos
Bucle:
MOV DL, [BX]         ; Llevamos a DL el valor
                     ; del número
ADD DL, 7            ; Le sumamos 7
MOV [BX], DL         ; Dejamos el número
                     ;actualizado
INC BX               ; Incrementamos BX en 1
LOOP Bucle

MOV AH, 4Ch ;Solicitamos al sistema operativo
INT 21h     ;la terminación de nuestro programa
END Inicio
```

Ejemplo de acceso a una cadena mediante direccionamiento relativo mediante índice

```
DOSSEG
.MODEL SMALL
.STACK 100h
.DATA
numeros  DB  1,2,3,4,5,6; Números de la cadena.
.CODE
Inicio:
MOV AX, @DATA        ; Indicamos donde se
MOV DS, AX           ; encuentran los datos
MOV  SI, 0           ; Ponemos el registro SI a 0
MOV  CX, 6           ; Indicamos el número de dígitos
 Bucle:
   ADD numeros[SI], 7
      INC SI
      LOOP Bucle
MOV AH, 4Ch; Solicitamos al sistema operativo
INT 21h      ; la terminación de nuestro programa
END Inicio
```