

Instrucciones Lógicas y de Manipulación de Bits

Incluyen operaciones lógicas, corrimientos y rotaciones.

Instrucciones:

- AND
 - OR
 - XOR
 - TEST
 - NOT
 - SHL/SAL
 - SHR
 - SAR
 - ROL
 - RCL
 - ROR
 - RCR
-



Instrucciones Lógicas y de Manipulación de Bits

No se permite el uso de estas instrucciones en los registros de segmento.

Todas las instrucciones lógicas modifican el registro de banderas.

Las operaciones lógicas siempre ponen el bit de acarreo y el de sobreflujo en ceros, y el resto de los bits de banderas cambian para reflejar la condición del resultado.



AND

A * B = X		
0	0	0
0	1	0
1	0	0
1	1	1

Figura 6. Tabla de verdad para la operación AND (multiplicación lógica).

Tabla 18. Instrucciones AND

Instrucciones	Comentarios
AND AL,BL	AL = AL AND BL
AND CX,DX	CX = CX AND DX
AND CL,33H	CL = CL AND 33H
AND DI,4FFFFH	DI = DI AND 4FFFFH
AND AX,[DI]	AX = AX AND DS:[DI]
AND ARRAY[SI],AL	ARRAY[SI]=ARRAY[SI] AND AL

OR

A + B = X		
0	0	0
0	1	1
1	0	1
1	1	1

Figura 8. Tabla de verdad para la operación OR (suma lógica).

Tabla 19. Instrucciones OR

Instrucciones	Comentarios
OR AH,BL	AH = AH OR BL
OR SI,DX	SI = SI OR DX
OR DH,0A3H	DH = DH OR 0A3H
OR SP,990DH	SP = SP OR 990DH
OR DX,[BX]	DX = DX OR DS:[BX]
OR DATES[DI+2],AL	DATES[DI+2]=DATES[DI+2] OR AL

TEST

La instrucción **TEST** realiza una operación **AND**.

La diferencia es que la instrucción **AND** cambia el operando destino, mientras que la instrucción **TEST** no lo hace.

Una operación **TEST** sólo afecta la condición del registro de banderas, el cual indica el resultado de la prueba.

Por ejemplo, si se usa la instrucción **TEST** para probar un único bit, la bandera de cero **Z** será:

$Z = 1$ si el bit era 0

$Z = 0$ si el bit era 1



TEST

Tabla 21. Instrucciones TEST.

Instrucciones	Comentarios
TEST DL,DH	Realiza: DL AND DH
TEST CX,BX	Realiza: CX AND BX
TEST AX,04H	Realiza: AH AND 04



NOT

La instrucción **NOT** invierte todos los bit de un byte o palabra.
(Realiza el **complemento a uno**).

La instrucción **NEG** realiza el **complemento a dos** a un número, lo cual significa que el signo aritmético de un número cambia de positivo a negativo o de negativo a positivo.

La función NOT se considera una operación lógica y la función NEG se considera una operación aritmética.



NOT y NEG

Tabla 22. Instrucciones NOT y NEG.

Instrucciones	Comentarios
NOT CH	CH=CMPLT1(CH)
NEG CH	CH=CMPLT2(CH)
NEG AX	AX=CMPLT2(AX)
NOT TEMP	TEMP=CMPLT1(TEMP)
NOT BYTE PTR[BX]	[BX]=CMPLT1([BX])



XOR

$$A \oplus B = X$$

0	0	0
0	1	1
1	0	1
1	1	0

Figura 10. Tabla de verdad para la operación XOR.

Tabla 20. Instrucciones OR Exclusiva

Instrucciones	Comentarios
XOR CH,DH	CH = CH XOR DH
XOR SI,BX	SI = SI XOR BX
XOR CH,0EEH	CH = CH XOR 0EEH
XOR DI,0DDH	DI = DI XOR 0DDH
XOR DX,[SI]	DX = DX XOR DS:[DI]
XOR DATE[DI+2],AL	DATE[DI+2]=DATE[DI+2] XOR AL

Enmascaramiento

Consiste en realizar una operación lógica entre un dato y un cierto patrón.

Esto con el fin de conocer el estado de ciertos bits, o modificarlos de acuerdo a un patrón requerido.



Enmascaramiento (**AND**)

Poner los bits 3 y 7 en 0:

<i>Dato:</i>		xxxx	xxxx	(cualquier valor)
<i>Mascara:</i>	AND	0111	0111	(77h)
<i>Resultado:</i>		0xxx	0xxx	

Es importante recordar que:

0 *AND* cualquier-valor, es 0 lógico siempre

1 *AND* cualquier-valor, es cualquier-valor

0	0	0
0	1	0
1	0	0
1	1	1

Enmascaramiento (**AND**)

Conocer el estado de los bits 0 y 5:

<i>Dato:</i>		xxxx	xxxx	(cualquier valor)
<i>Mascara:</i>	AND	0010	0001	(21h)
<i>Resultado:</i>		00x0	000x	

Ejemplo en lenguaje C:

```
if( dato & 0x21 ) {  
    realizar-ciertas-acciones;  
}
```

Representación de 21h en lenguaje C

Operador **AND** lógico
en lenguaje C



Enmascaramiento (**OR**)

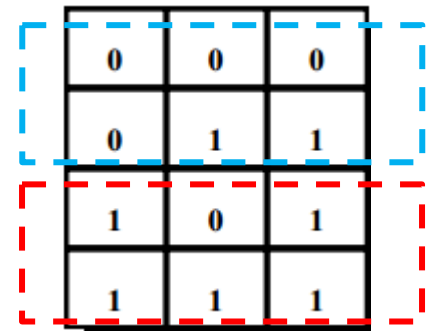
Poner en 1 lógico el nibble mas significativo:

<i>Dato:</i>		xxxx	xxxx	(cualquier valor)
<i>Mascara:</i>	OR		0000	(F0h)
<i>Resultado:</i>			xxxx	

Es importante recordar que:

1 OR cualquier-valor, es 1 lógico siempre

0 OR cualquier-valor, es cualquier-valor



0	0	0
0	1	1
1	0	1
1	1	1

Enmascaramiento (**XOR**)

Esta operación lógica se suele utilizar para invertir bits.

Invertir (complementar) el nibble menos significativo:

<i>Dato:</i>		xxxx	xxxx	(cualquier valor)
<i>Mascara:</i>	XOR	0000	1111 (0Fh)	
<i>Resultado:</i>		xxxx	$\bar{x} \bar{x} \bar{x} \bar{x}$	

Es importante recordar que:

1 XOR cualquier-valor, es el valor invertido

0 XOR cualquier-valor, es cualquier-valor

0	0	0
0	1	1
1	0	1
1	1	0

Corrimientos y Rotaciones



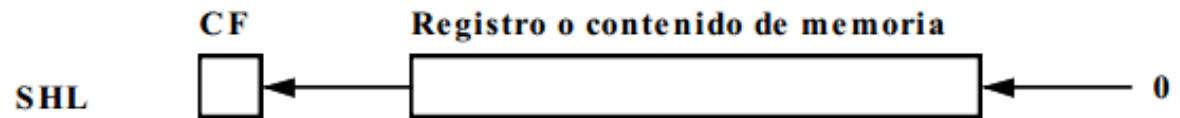
Corrimientos

Las instrucciones de corrimiento posicionan o mueven números a la izquierda o a la derecha dentro de un registro o localidad de memoria, excepto los registros de segmento.

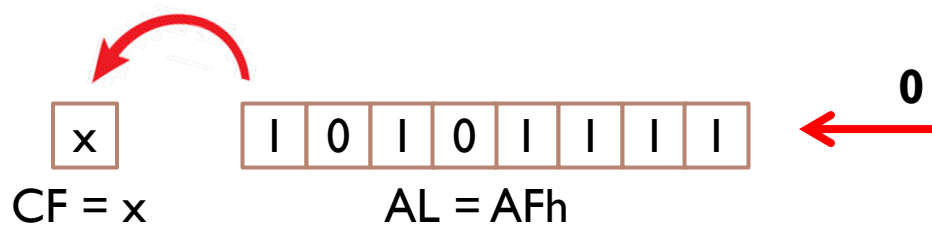


Corrimiento a la izquierda

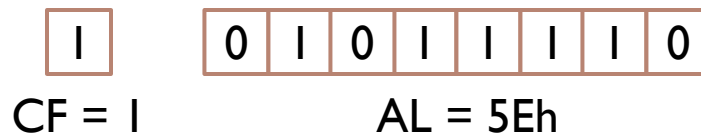
SHL: Shift Logical Left



Ejemplo:



SHL AL,I

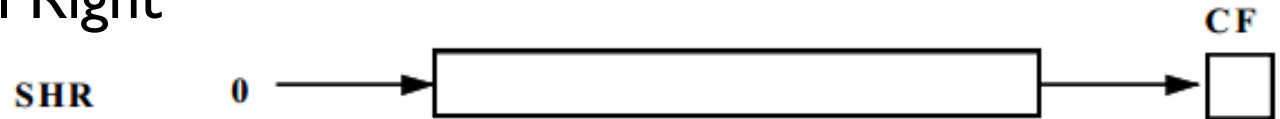


Carry Flag (CF)

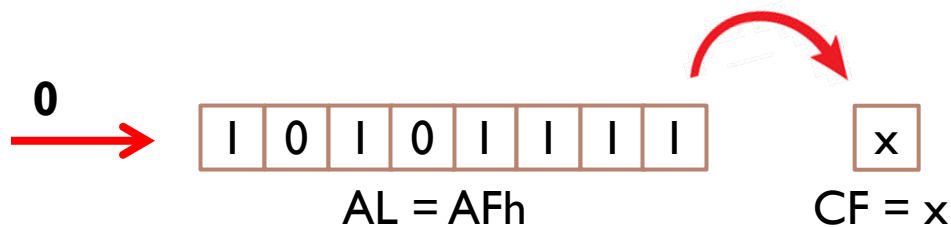


Corrimiento a la derecha

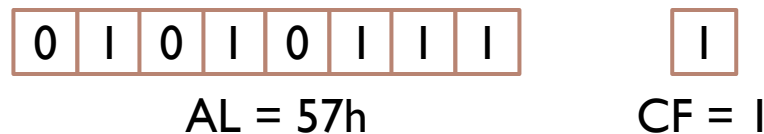
SHR: Shift Logical Right



Ejemplo:

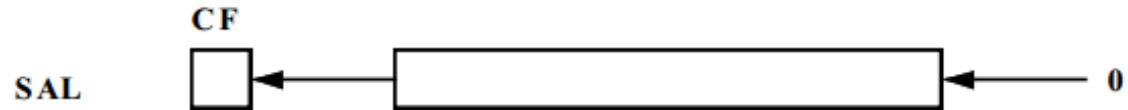


SHR AL,I

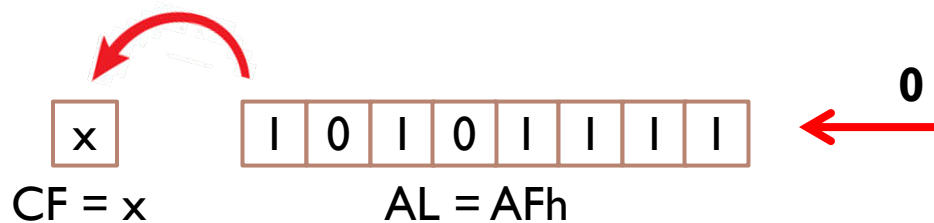


Corrimiento a la izquierda **aritmético**

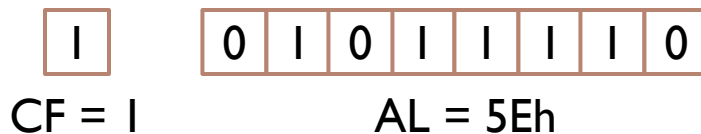
SAL: Shift Arithmetic Left



Ejemplo:



SAL AL,1

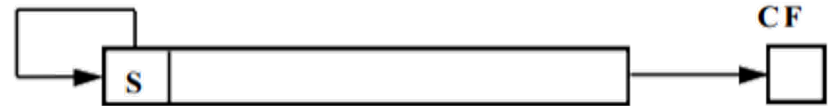


Carry Flag (CF)

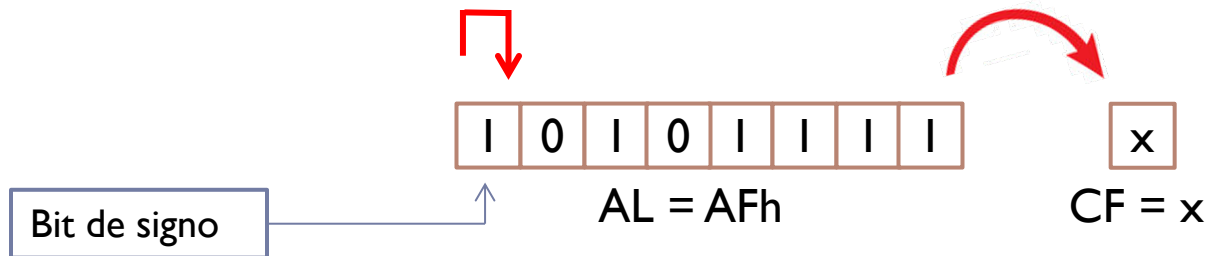
Corrimiento a la derecha **aritmético**

SAR: Shift Arithmetic Right

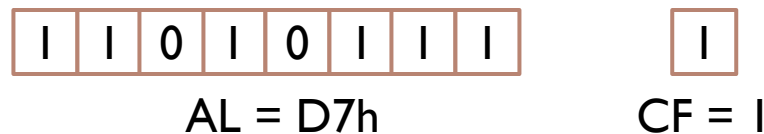
SAR



Ejemplo:



SAR AL,1



Corrimientos

Cuando se desea realizar un corrimiento de mas de un bit se usa el registro CL para indicar la cuenta de corrimientos. El registro CL no se modifica al ejecutarse la instrucción de corrimiento.

Ejemplos:

Hace un corrimiento de dos bits a la izquierda en el registro AX.

```
MOV CL,2  
SHL AX,CL
```

Hace un corrimiento aritmético de siete bits a la derecha en el registro BL.

```
MOV CL,7  
SAR BL,CL
```



Corrimientos

Tabla 23. Instrucciones de Corrimiento.

Instrucciones	Comentarios
SHL AX,1	Corrimiento de AX 1 lugar a la izquierda
SHR BX,1	Corrimiento de BX 1 lugares a la derecha
SAL DATA1,CL	Corrimiento aritmético de DATA CL lugares a la izquierda
SAR SI,CL	Corrimiento aritmético de SI CL lugares a la derecha



Corrimientos

Las operaciones de corrimientos también se pueden utilizar como operaciones aritméticas simples tales como:

Corrimiento a la izquierda: multiplicación por potencias de 2^n

Corrimiento a la derecha: división por potencias de 2^n

Ejemplos:

Multiplicación por potencias de 2^n :

$$2Dh * 2 = 0x5A$$

$$2Dh \ll 1 = 0x5A$$



Operador **Corrimiento
lógico a la izquierda**
en lenguaje C

$$36h * 4 = 0xD8$$

$$36h \ll 2 = 0xD8$$



Corrimientos

División por potencias de 2^n :

$2Dh / 2 = 0x16$

$2Dh \gg 1 = 0x16$



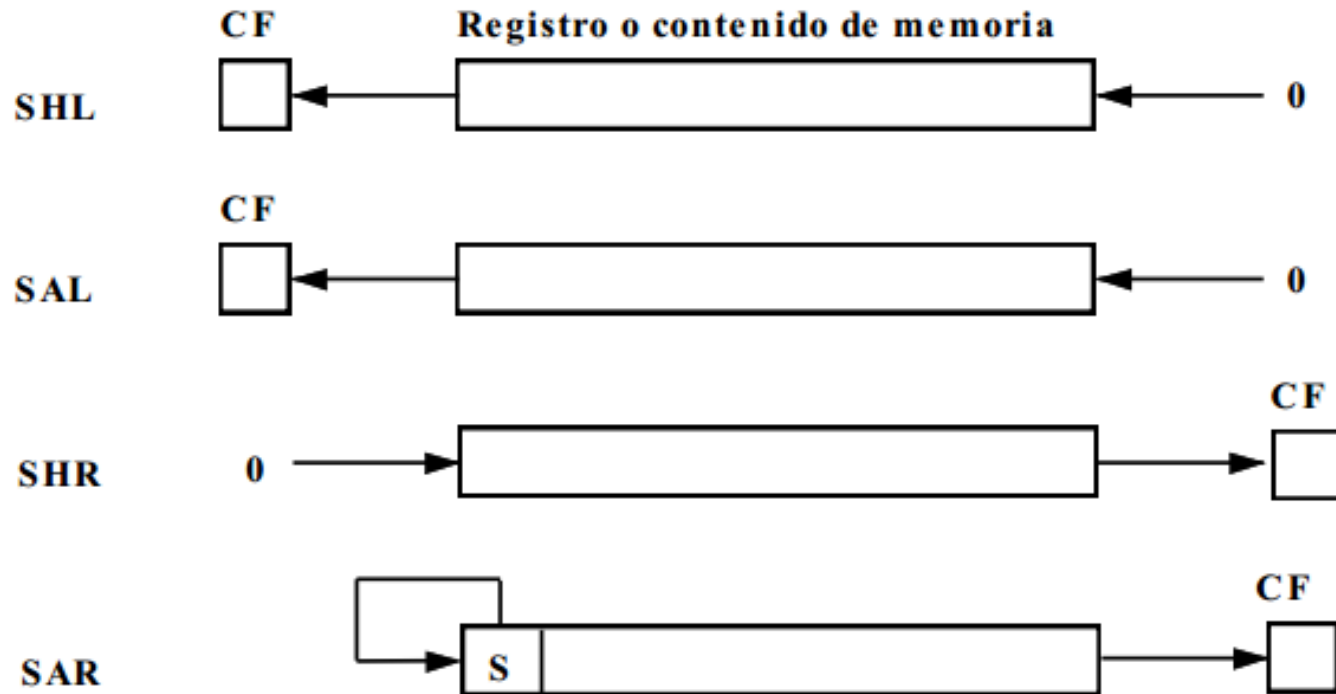
$36h / 8 = 0x06$

$36h \gg 3 = 0x06$

Operador **Corrimiento
lógico a la derecha**
en lenguaje C



Corrimientos



CF = Bandera de Acarreo



Rotaciones

Las instrucciones de rotación posicionan datos binarios mediante la rotación de la información en un registro o localidad de memoria ya sea de un extremo u otro o a través de la bandera de acarreo.

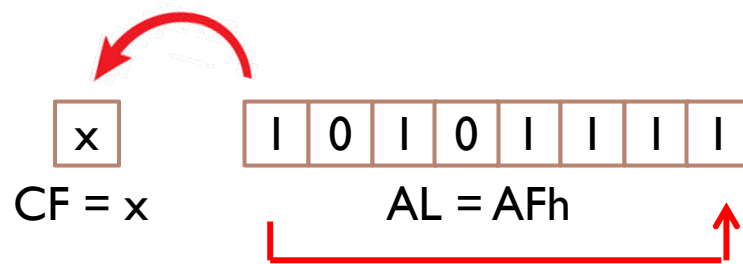
Al igual que con los Corrimientos, si se va a realizar una rotación de mas de un bit se tiene que usar al registro CL para indicar la cuenta de rotaciones.



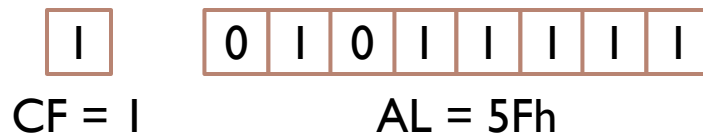
Rotación a la izquierda

ROL: Rotate Left

Ejemplo:



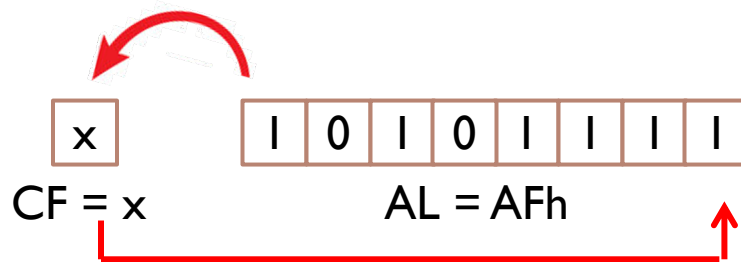
ROL AL,I



Rotación a la izquierda **con acarreo**

RCL: Rotate Left through Carry

Ejemplo:



RCL AL,1

1

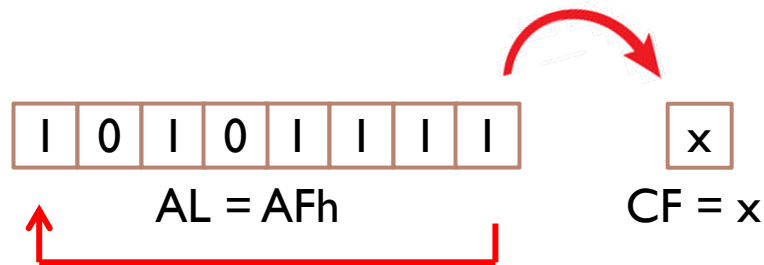
0	1	0	1	1	1	1	x
---	---	---	---	---	---	---	---

CF = 1 Si la bandera de acarreo previamente estaba en 1: **AL = 5Fh**
Si estaba en 0: **AL = 5Eh**

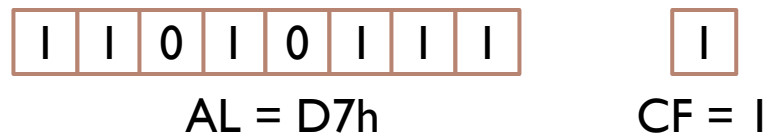
Rotación a la derecha

ROR: Rotate to Right

Ejemplo:



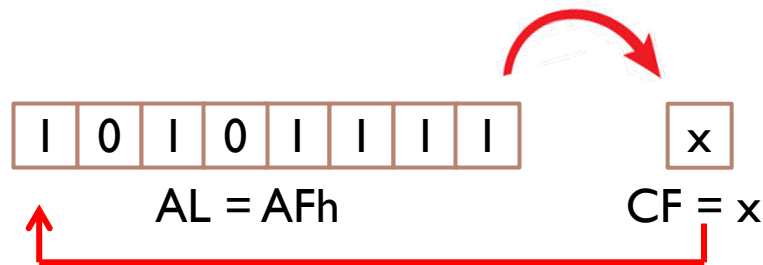
ROR AL,1



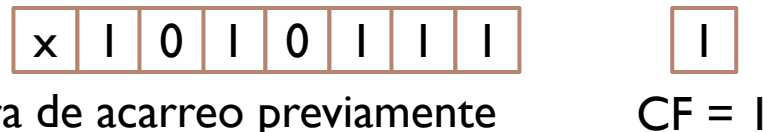
Rotación a la derecha **con acarreo**

RCR: Rotate Right through Carry

Ejemplo:



RCR AL, 1



Si la bandera de acarreo previamente estaba en 1: **AL = D7h**

Si estaba en 0: **AL = 57h**

Rotaciones

Tabla 24. Instrucciones de Rotación.

Instrucciones	Comentarios
ROL SI,1	Rota a SI 1 lugares a la izquierda
ROR AX,CL	Rota a AX CL lugares a la derecha
RCL BL,1	Rota a BL 1 lugares a la izquierda, a través de CF
RCR AH,CL	Rota a AH CL lugares a la derecha, a través de CF



Rotaciones

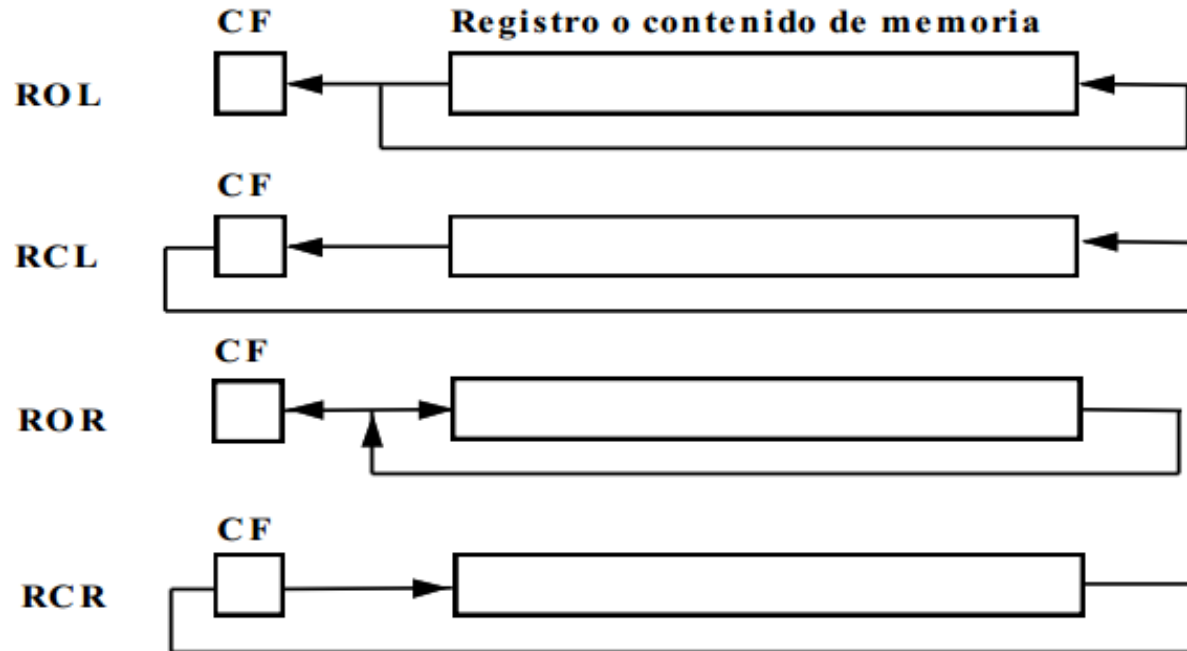


Figura 13. Conjunto de operaciones de rotación disponibles en el 8088.