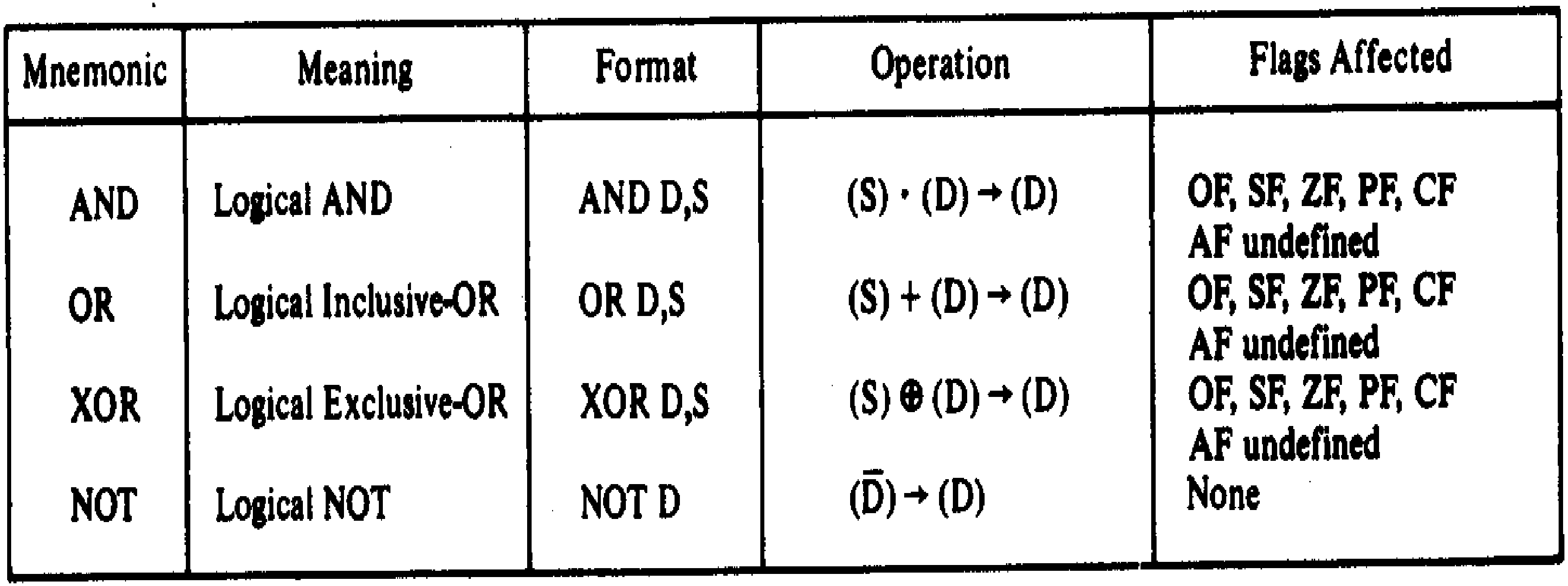
Logic Instructions

Logic Instructions

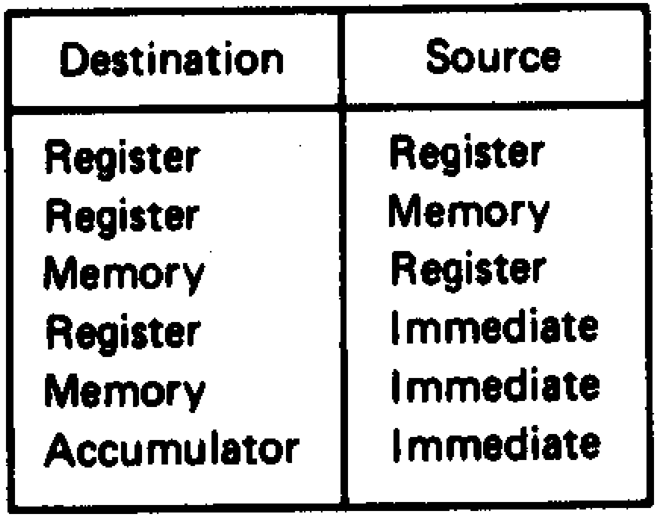
- The logic instructions include
  - AND
  - OR
  - XOR (Exclusive-OR)



Mnemonic	Meaning	Format	Operation	Flags Affected
AND	Logical AND	AND D,S	$(S) \cdot (D) \rightarrow (D)$	OF, SF, ZF, PF, CF AF undefined
OR	Logical Inclusive-OR	OR D,S	$(S) + (D) \rightarrow (D)$	OF, SF, ZF, PF, CF AF undefined
XOR	Logical Exclusive-OR	XOR D,S	$(S) \oplus (D) \rightarrow (D)$	OF, SF, ZF, PF, CF AF undefined
NOT	Logical NOT	NOT D	$(\bar{D}) \rightarrow (D)$	None

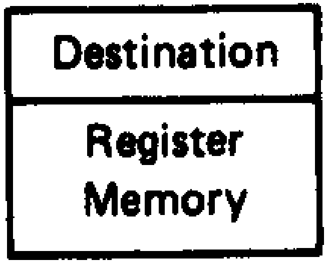
Logic Instructions (cont.)

- Logic instructions : AND, OR, XOR, NOT



Destination	Source
Register	Register
Register	Memory
Memory	Register
Register	Immediate
Memory	Immediate
Accumulator	Immediate

Allowed operands for AND, OR, and XOR instructions



Destination
Register
Memory

Allowed operands for NOT instruction

Logic Instructions (cont.)

- *EXAMPLE:*
  - Describe the results of executing the following instructions?

```
MOV AL, 01010101B
AND AL, 00011111B
OR   AL, 11000000B
XOR AL, 00001111B
NOT AL
```

- *Solution:*

$(AL)=01010101_2 \cdot 00011111_2= 00010101_2=15_{16}$

Executing the OR instruction, we get

$(AL)= 00010101_2 +11000000_2= 11010101_2=D5_{16}$

Executing the XOR instruction, we get

$(AL)= 11010101_2 \oplus 00001111_2= 11011010_2=DA_{16}$

Executing the NOT instruction, we get

$(AL)= (NOT)11011010_2 = 00100101_2=25_{16}$

$\oplus$

Logic Instructions (cont.)

- *EXAMPLE:*
  - Masking and setting bits in a register
- *Solution:*

Mask off the upper 12 bits of the word of data in AX

AND AX, $000F_{16}$

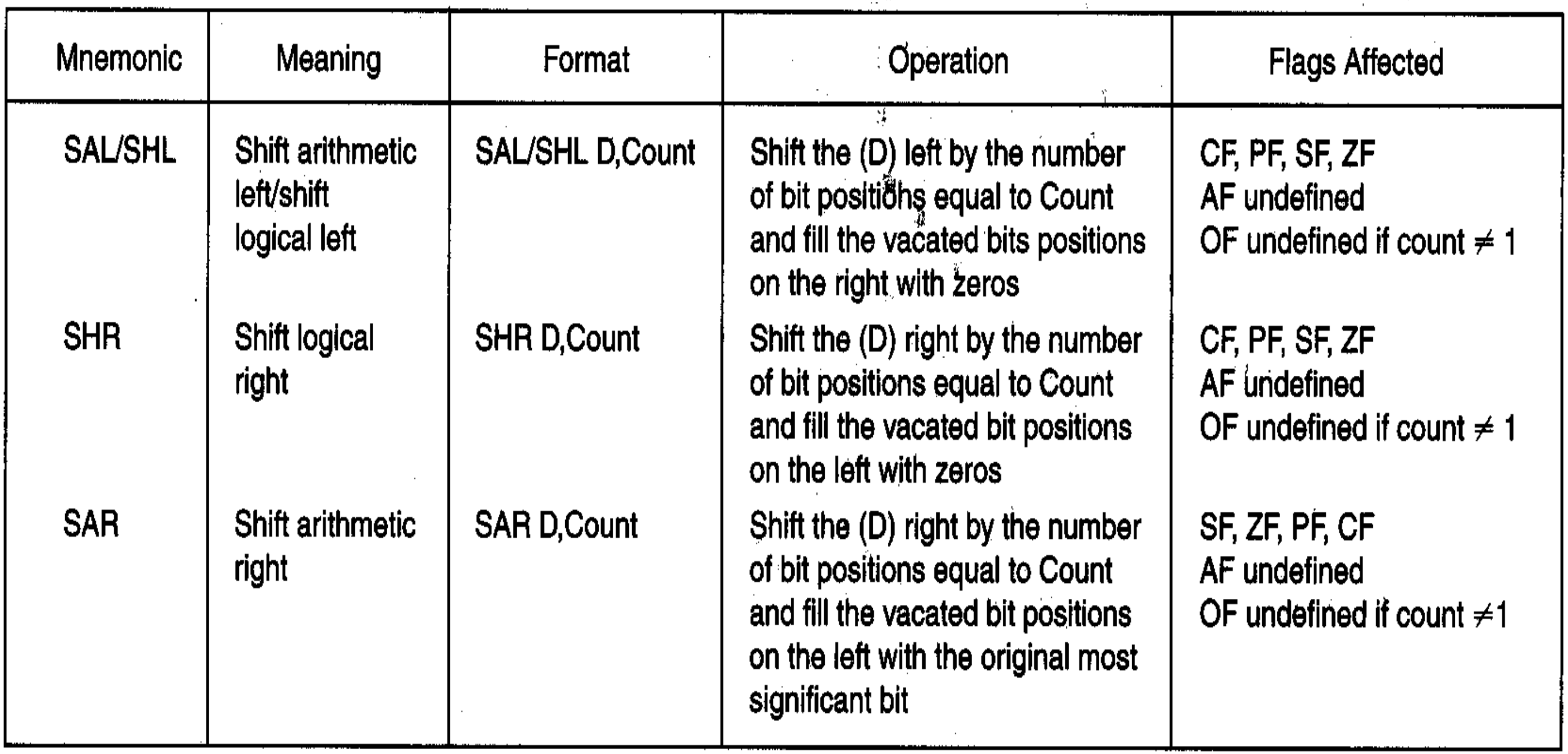
Setting B_4 of the byte at the offset address CONTROL_FLAGS

```
MOV   AL, [CONTROL_FLAGS]
OR    AL, 10H
MOV   [CONTROL_FLAGS], AL
```

Executing the above instructions, we get

$(AL) = XXXXXXXX_2 + 00010000_2 = XXX1XXXX_2$

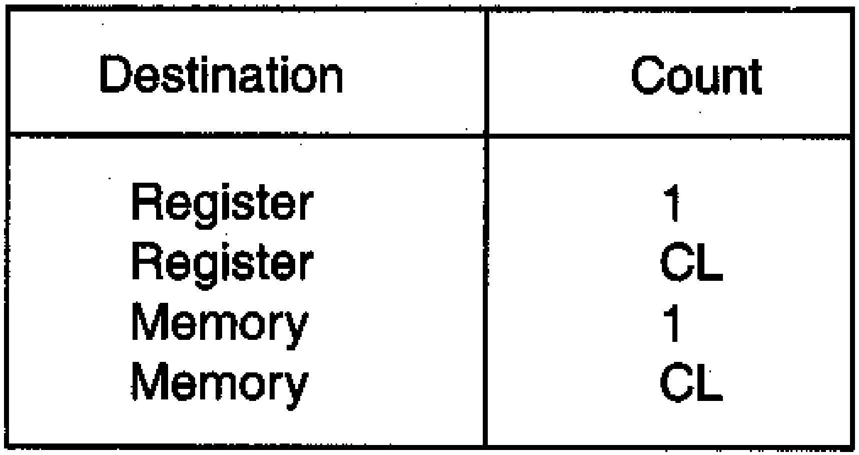
Shift Instructions



Mnemonic	Meaning	Format	Operation	Flags Affected
SAL/SHL	Shift arithmetic left/shift logical left	SAL/SHL D,Count	Shift the (D) left by the number of bit positions equal to Count and fill the vacated bits positions on the right with zeros	CF, PF, SF, ZF AF undefined OF undefined if count ≠ 1
SHR	Shift logical right	SHR D,Count	Shift the (D) right by the number of bit positions equal to Count and fill the vacated bit positions on the left with zeros	CF, PF, SF, ZF AF undefined OF undefined if count ≠ 1
SAR	Shift arithmetic right	SAR D,Count	Shift the (D) right by the number of bit positions equal to Count and fill the vacated bit positions on the left with the original most significant bit	SF, ZF, PF, CF AF undefined OF undefined if count ≠1

Shift Instructions (cont.)

- Shift instructions: SHL, SHR, SAL, SAR

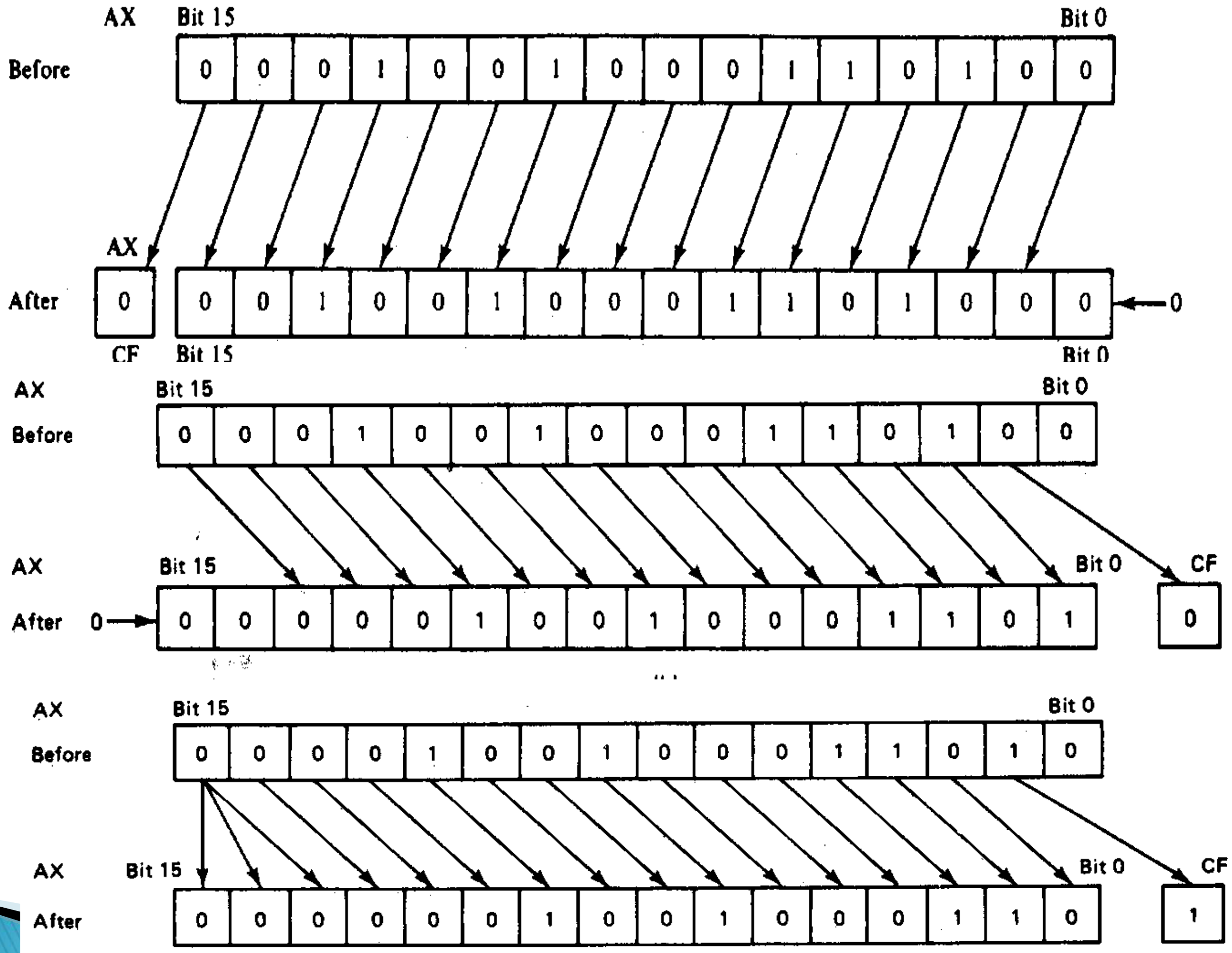


Destination	Count
Register	1
Register	CL
Memory	1
Memory	CL

Allowed operands for shift instructions

Shift Instructions (cont.)

- Shift instructions: SHL, SHR, SAL, SAR



SHL AX, 1

SHR AX, CL
(CL)=2

SAR AX, CL
(CL)=2

Shift Instructions (cont.)

- *EXAMPLE:*
  - Assume that CL contains 02_{16} and AX contains $091A_{16}$.
  - Determine the new contents of AX and the carry flag after the instruction SAR AX, CL is executed
- *Solution:*

(AX)=0000001001000110_2=0246_{16}

and the carry flag is (CF)=1_2

Shift Instructions (cont.)

- *EXAMPLE:*
  - Isolate the bit B3 of the byte at the offset address CONTROL_FLAGS.
- *Solution:*

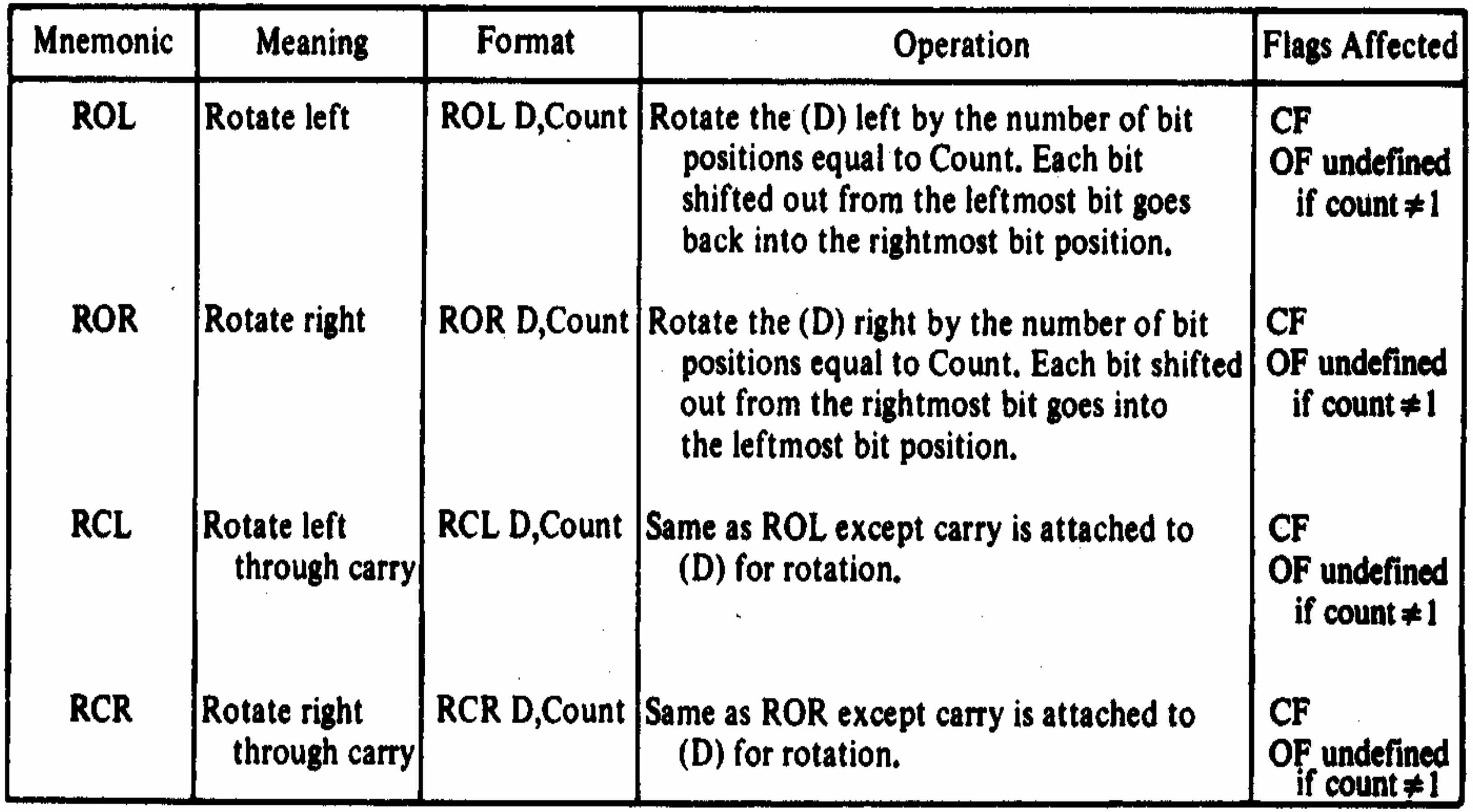
```
MOV AL, [CONTROL_FLAGS]
MOV CL, 04H
SHR AL, CL
```

Executing the instructions, we get

(AL)=0000$B_7B_6B_5B_4$ and (CF)=B_3

Rotate Instructions

- Rotate instructions: ROL, ROR, RCL, RCR

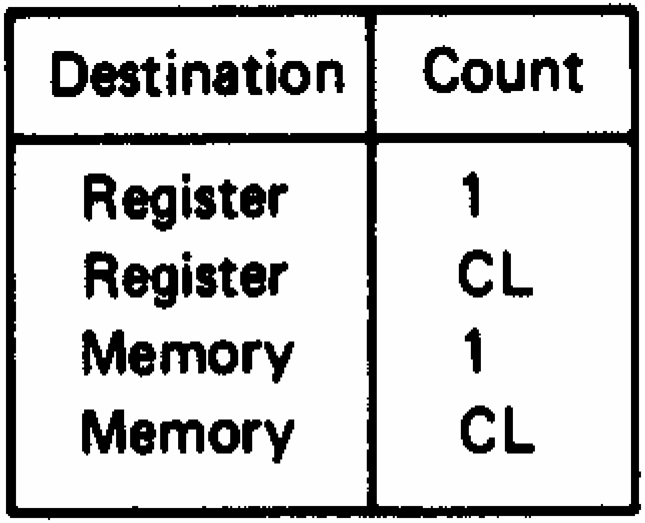


Mnemonic	Meaning	Format	Operation	Flags Affected
ROL	Rotate left	ROL D,Count	Rotate the (D) left by the number of bit positions equal to Count. Each bit shifted out from the leftmost bit goes back into the rightmost bit position.	CF OF undefined if count ≠ 1
ROR	Rotate right	ROR D,Count	Rotate the (D) right by the number of bit positions equal to Count. Each bit shifted out from the rightmost bit goes into the leftmost bit position.	CF OF undefined if count ≠ 1
RCL	Rotate left through carry	RCL D,Count	Same as ROL except carry is attached to (D) for rotation.	CF OF undefined if count ≠ 1
RCR	Rotate right through carry	RCR D,Count	Same as ROR except carry is attached to (D) for rotation.	CF OF undefined if count ≠ 1

(a)

Rotate Instructions (cont.)

- Rotate instructions: ROL, ROR, RCL, RCR

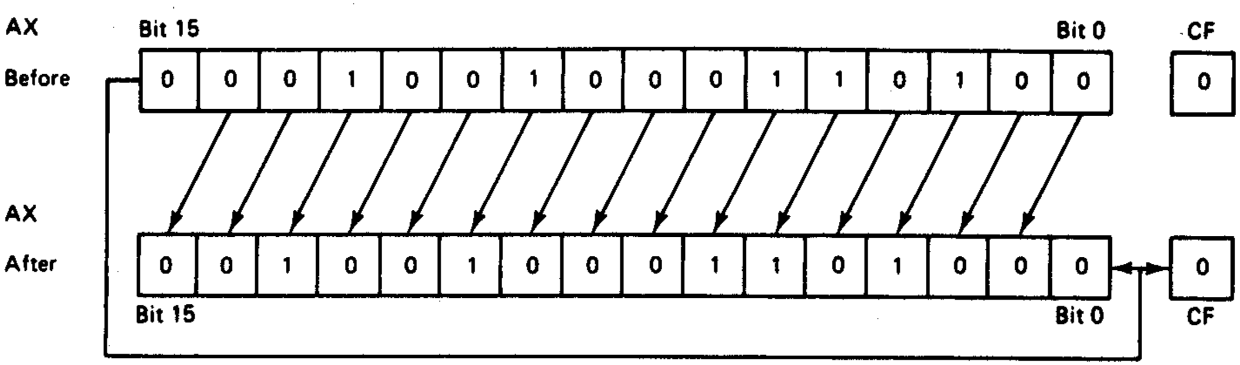


Destination	Count
Register	1
Register	CL
Memory	1
Memory	CL

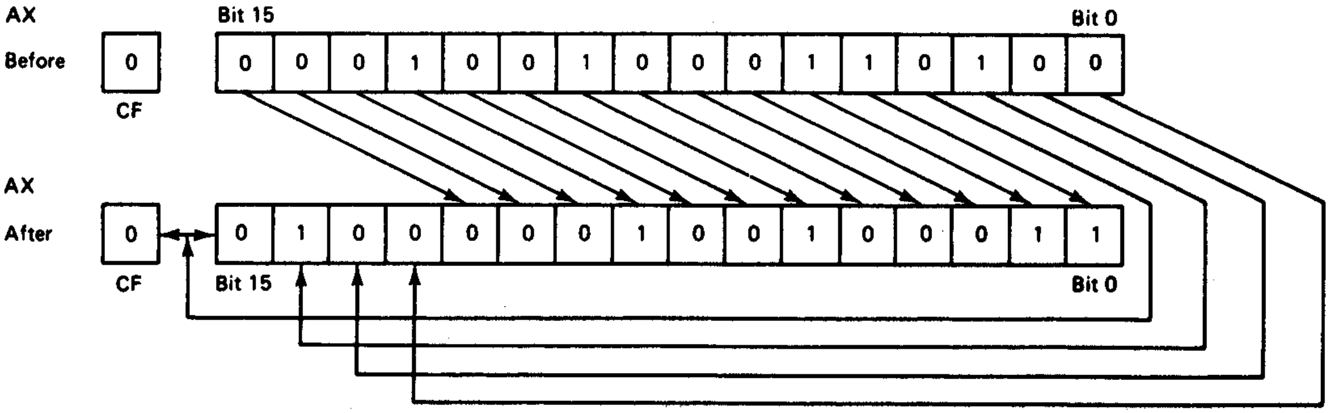
(b)

Rotate Instructions (cont.)

- Rotate instructions: ROL, ROR, RCL, RCR



ROL AX, 1

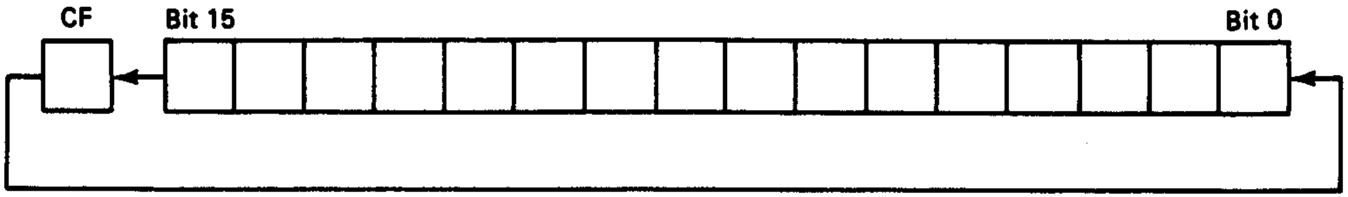


ROR AX, CL
(CL)=4

Rotate Instructions (cont.)

- Rotate instructions: ROL, ROR, RCL, RCR

For RCL, RCR, the bits are rotate through the carry flag



Rotate Instructions (cont.)

- *EXAMPLE:*
  - What is the result in BX and CF after execution of the following instructions?

    RCR BX, CL

    Assume that, prior to execution of the instruction, (CL)=04_{16}, (BX)=1234_{16}, and (CF)=0
- *Solution:*

  The original contents of BX are

  (BX) = $0001001000110100_2 = 1234_{16}$

  Execution of the RCR command causes a 4-bit rotate right through carry to take place on the data in BX, the results are

  (BX) = $1000000100100011_2 = 8123_{16}$

  (CF) = 0_2

Rotate Instructions (cont.)

- *EXAMPLE:*
  - Disassembly and addition of 2 hexadecimal digits stored as a byte in memory.
- *Solution:*
  - MOV AL, [HEX_DIGITS]
  - MOV BL, AL
  - MOV CL, 04H
  - ROR BL, CL
  - AND AL, 0FH
  - AND BL, 0FH
  - ADD AL, BL