The 8088/8086 Microprocessors

Programming, Interfacing, Software, Hardware and Applications

Logical & Shift instructions

Lecture on Logical & Shift Instructions

Logical instructions: The 88/86 processors has instructions to perform bit by bit logic operation on the specified source and destination operands.

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	$\overline{(D)} \rightarrow (D)$	None

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

Allowed operands for AND, OR, and XOR operations

Destination
Register
Memory

Allowed operand for NOT Operation

Logical Instructions

- AND
  - Uses any addressing mode except memory-to-memory and segment registers
  - Especially used to clear certain bits (masking)

    xxxx xxxx **AND** 0000 1111 = 0000 xxxx

    (clear the first four bits)
  - Examples: AND BL, 0FH

    AND AL, [345H]

- OR
  - Used to set certain bits

    xxxx xxxx **OR** 0000 1111 = xxxx 1111

    (Set the upper four bits)

- XOR
  - Used to invert certain bits

    xxxx xxxx XOR 0000 1111 = xxxxx'x'x'x'

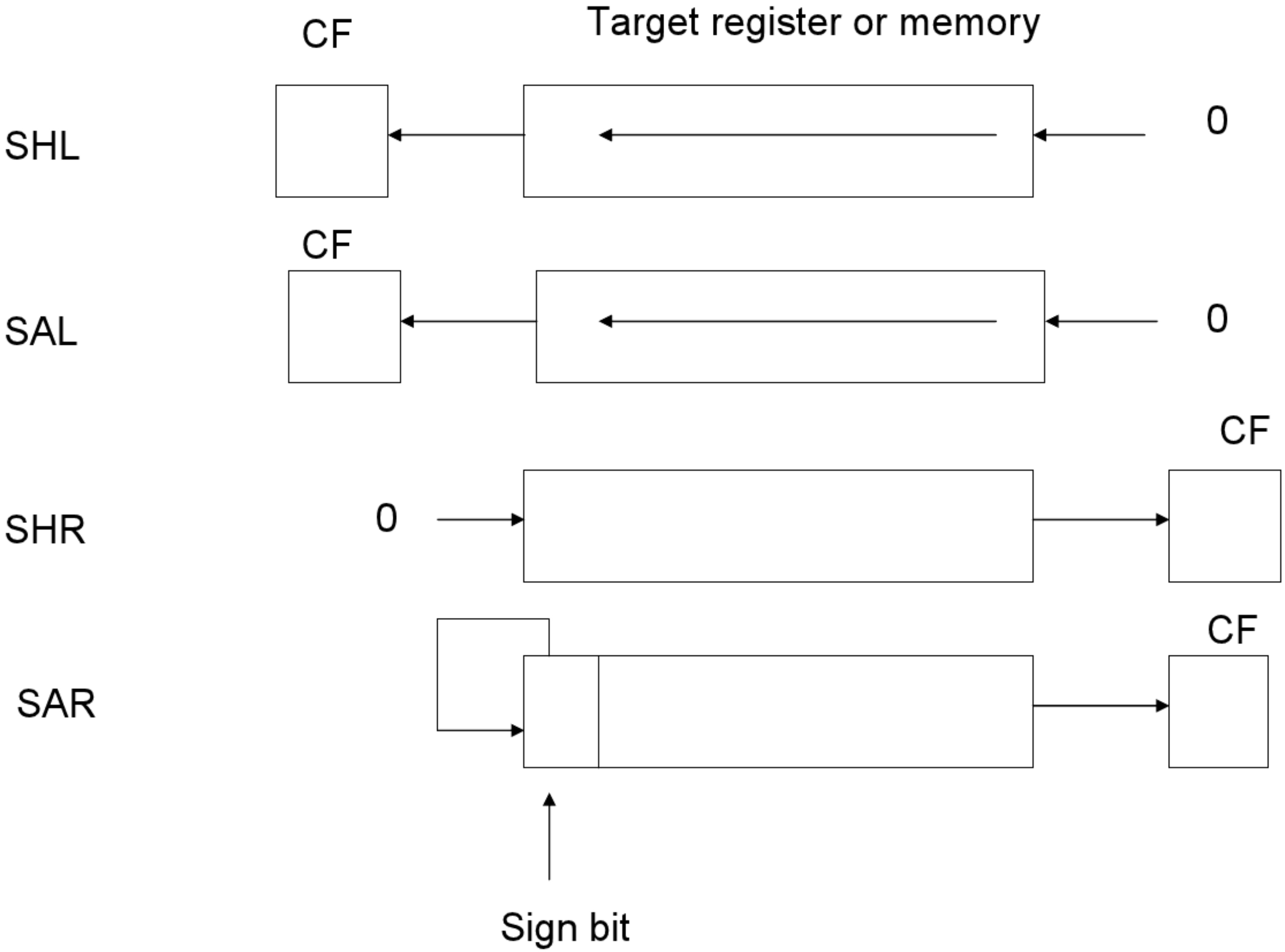
  -Example: Clear bits 0 and 1, set bits 6 and 7, invert bit 5 of register CL:

```
AND CL,  1111 1100B ; clear bits 0 and 1
OR CL, 1100 0000B; set bits 6 and 7
XOR CL, 0010 0000B; invert bit 5.
```

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 bits 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 bits positions on the left with the original most significant bit	CF,PF,SF,ZF AF undefined OF undefined if count ≠1

Destination	Count
Register	1
Register	CL
Memory	1
Memory	CL

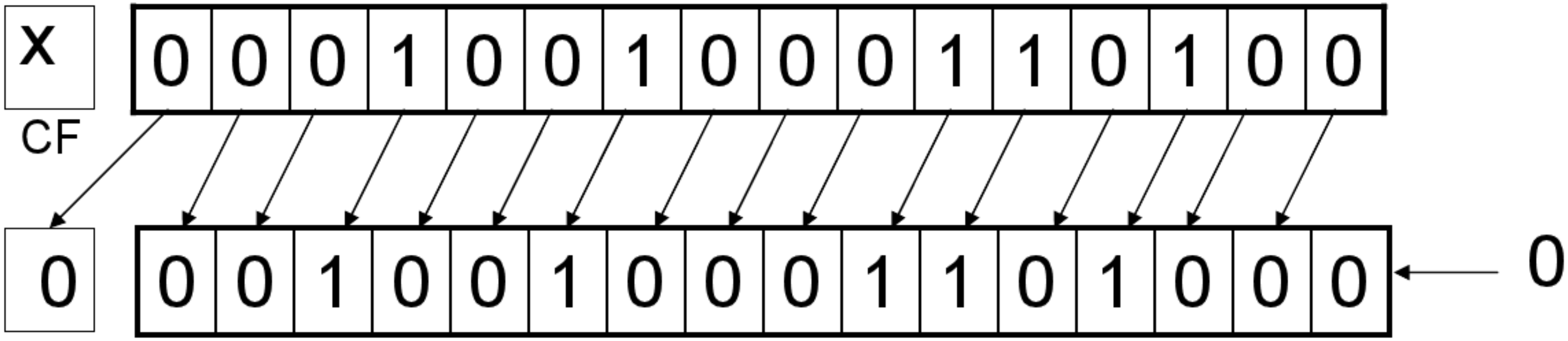

CF
Target register or memory
SHL
0
CF
SAL
0
CF
SHR
0
CF
SAR
Sign bit

Note that the amount of shift specified in the source operand can be defined Explicitly if it is one bit or should be stored in CL if more than 1.

EX: SHL AX, 1

causes the 16-bit register to be shifted 1-bit position to the left Where the vacated LSB is filled with zero and the bit shifted out of the MSB is saved in CF

Register AX before instruction SHL AX, 1

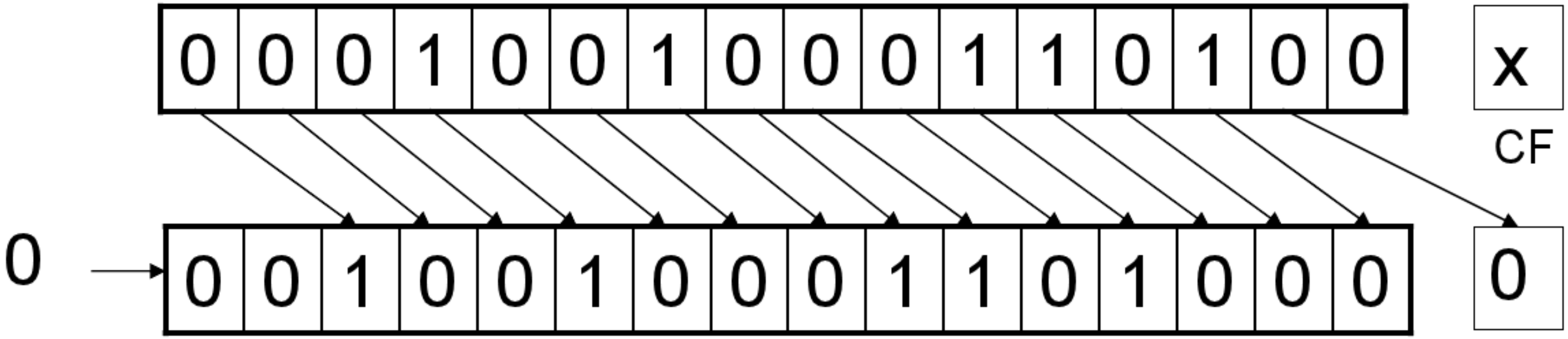


CF																	
X	0	0	0	1	0	0	1	0	0	0	1	1	0	1	0	0	
0	0	0	1	0	0	1	0	0	0	1	1	0	1	0	0	0	← 0

Shift arithmetic left SAL is exactly the same as SHL

```
EX:     MOV CL, 2H
        SHR AX, CL
```

The two MSBs are filled with zeros and the LSB is thrown away while the Second LSB is saved in CF.



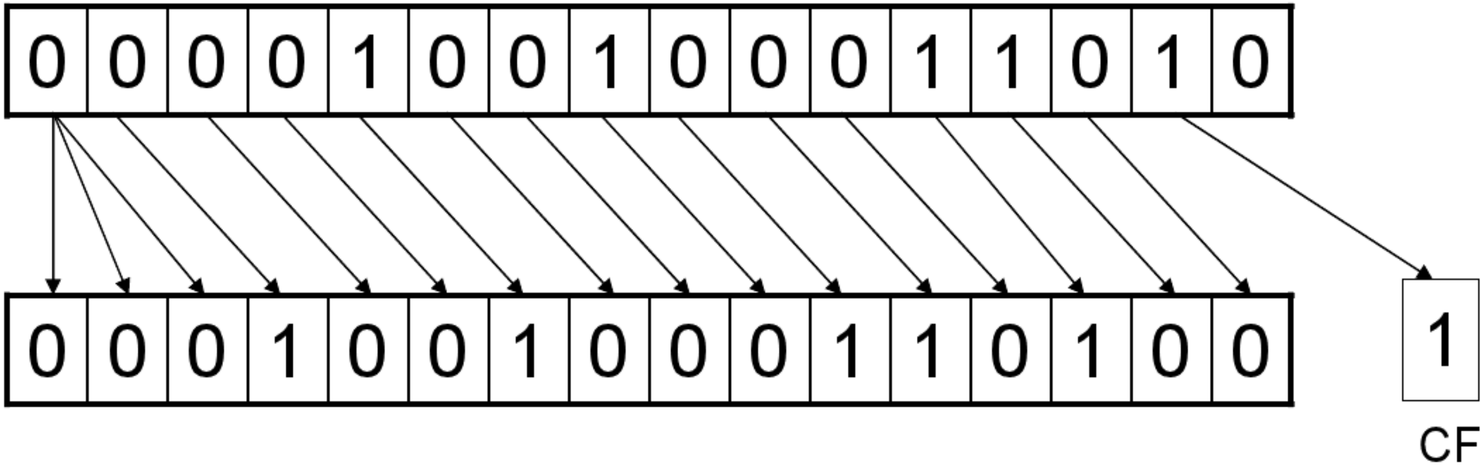
0 0 0 1 0 0 1 0 0 0 1 1 0 1 0 0 | x

CF

0 → 0 0 1 0 0 1 0 0 0 1 1 0 1 0 0 0 | 0

In Arithmetic shift to the right SAR the vacated bits at the left are filled with The value of the original MSB of the operand. Thus the original sign of the Number is maintained.

Ex: Assume (CL)= 2 and (AX)= 091AH. Determine the new contents of AX And CF after the instruction SAR AX, CL is executed.



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

1

CF

This operation is equivalent to division by powers of 2 as long as the bits shifted out of the LSB are zeros.

- Ex. Multiply AX by 10 using shift instructions

```
SHL AX, 1
MOV BX, AX
MOV CL,2
SHL AX,CL
ADD AX, BX
```

- Ex. What is the result of SAR CL, 1
  if CL initially contains B6H?
  DBH

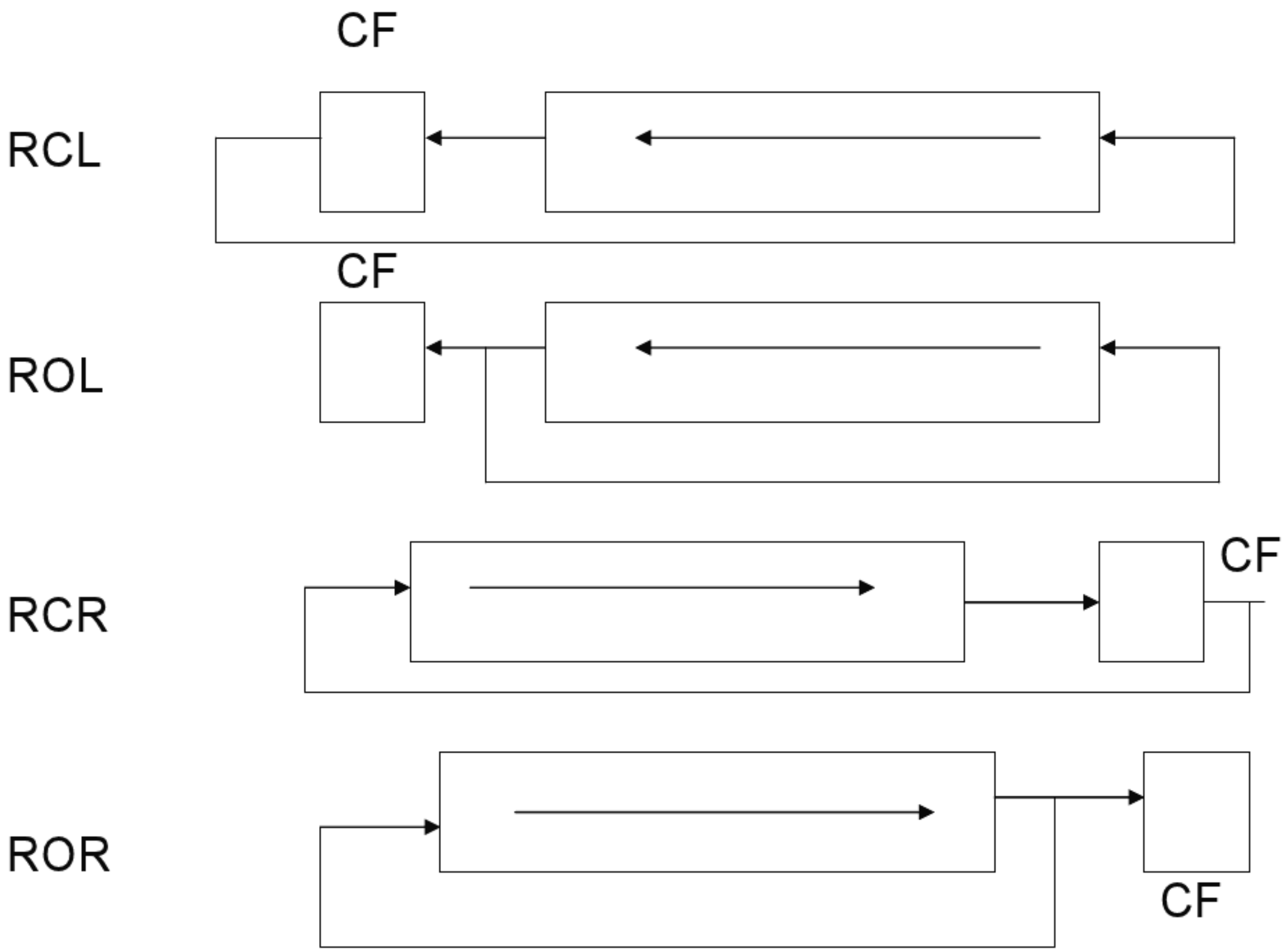
- Ex. What is the result of SHL AL, CL
  if AL contains 75H and CL contains 3?
  A8H

Rotate Instructions

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 left most 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 back 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

Destination	**Count**
Register	1
Register	CL
Memory	1
Memory	CL

Allowed Operands for rotate instruction



CF

RCL

CF

ROL

RCR

CF

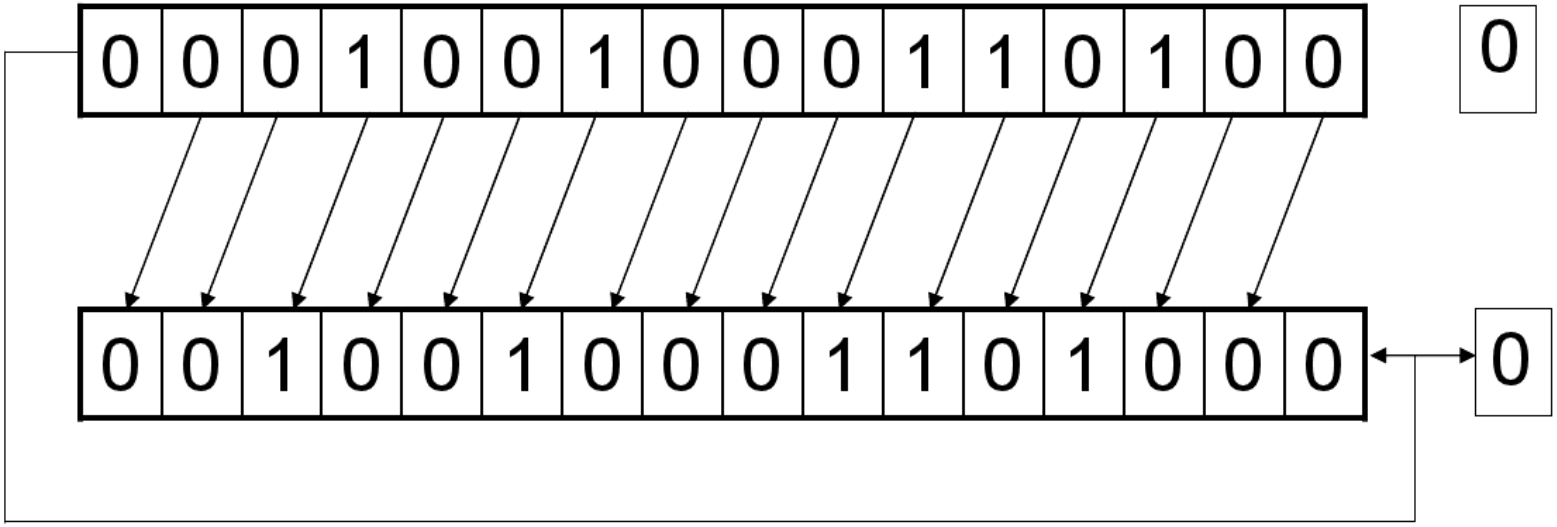
ROR

CF

Ex. What is the result of ROL BTRE PTR [SI], 1
if SI is pointing to a memory location that contains 41H? (82H)

EX: Assume (AX) = 1234H

What is the result of executing the instruction ROL AX, 1

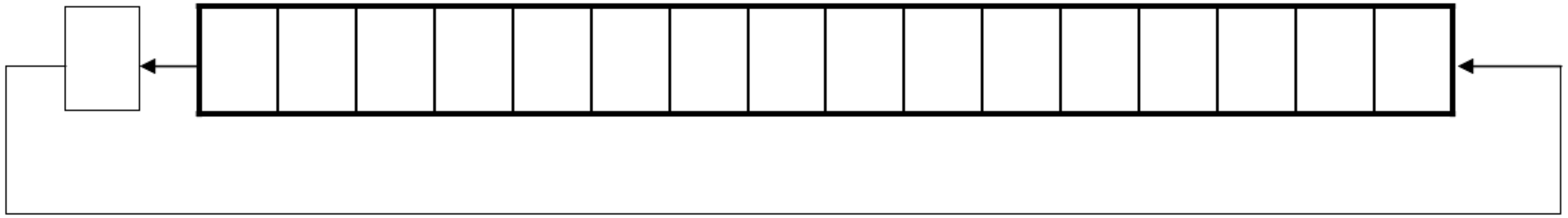


0	0	0	1	0	0	1	0	0	0	1	1	0	1	0	0	0
0	0	1	0	0	1	0	0	0	1	1	0	1	0	0	0	0

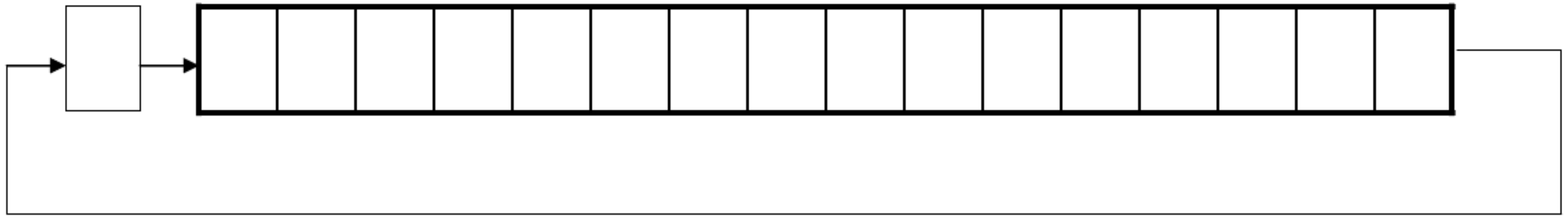
The original value of bit 15 which is 0 is rotated into CF and bit 0 of AX. All other bits have been rotated 1 bit position to the left.

Rotate right ROR instruction operates the same way as ROL except that Data is rotated to the right instead of left.

In rotate through carry left RCL and rotate through carry right RCR the bits rotate through the carry flag.



Rotation caused by RCL instruction



Rotation caused by RCR instruction

Example

Write a program that counts the number of 1's in a byte and writes it into BL

```
            DATA1 DB 97 ; 61h
            SUB BL, BL ; clear BL to keep the number of 1s
            MOV DL, 8 ; Counter
            MOV AL, DATA1
   AGAIN: ROL AL,1  ; rotate left once
            JNC NEXT ;   check for 1
            INC BL ; if CF=1 then add one to count
      NEXT: DEC DL ; go through this 8 times
            JNZ AGAIN ; if not finished go back
      NOP
```