Logic

Shift

and Rotation

Reference: Assembly Language Programming and Organization of the IBM PC – Charles Marut – Chapter 7

Shift

Content

- Introduction
- Types of Shift
  - Logical shift
    - Shift Left
    - Shift right
  - Arithmetical shift
    - Shift arithmetic left
    - Shift arithmetic right
- Reference

Introduction

Shift operation move the bits in a pattern, changing the position of the bits. They can move bits to the left or to the right. We can divide shift operation into two categories:

1. Logical Shift
2. Arithmetic Shift

Logical Shift

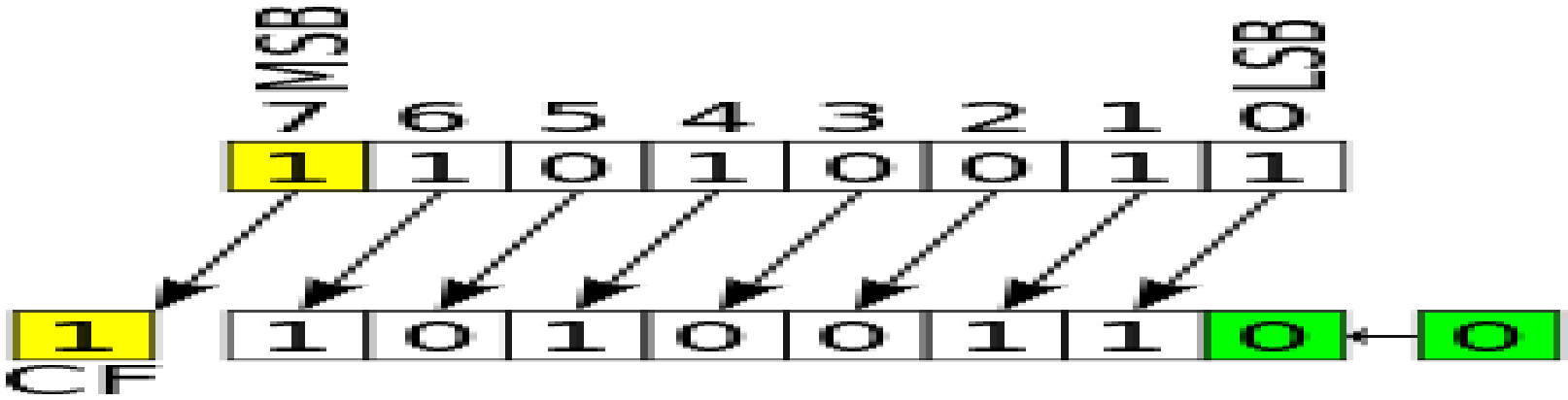
- A *logical shift* moves the bits within the cell one position to the right or to the left
- In a logical *right* shift, the least significant bit LSB is discarded and the most significant bit MSB is assigned 0.
- In a logical left shift, the LSB is assigned 0 and the msb is discarded.
- A shift instruction will have an operand that specifies how many times the one position shift is applied.

There are two kinds of Logical shift:

1. Shift Left
2. Shift Right

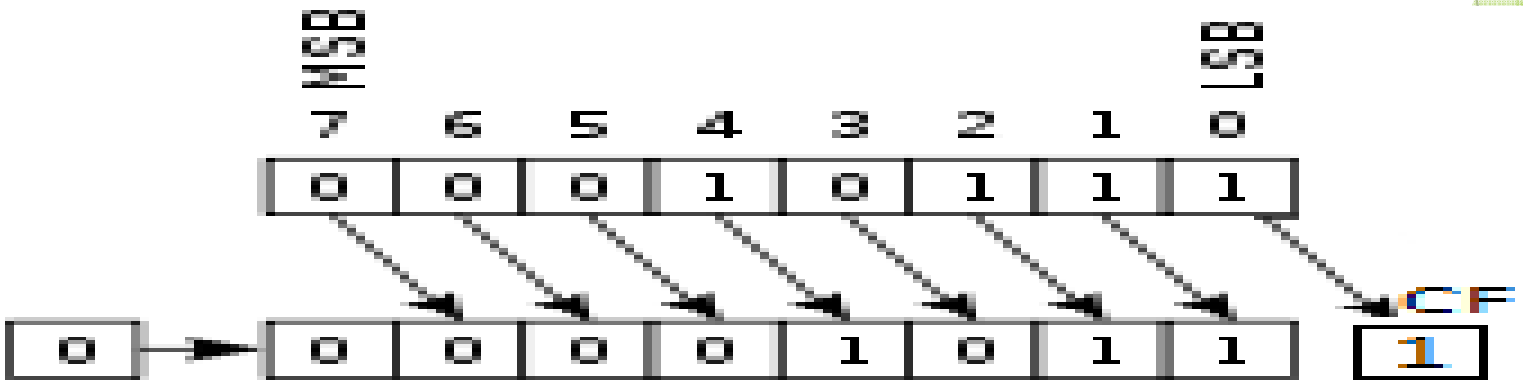
Left Shift

- A shift left logical of one position moves each bit to the left by one. The low-order bit LSB is replaced by a zero bit and the high-order bit MSB move to CF (Carry Flag)
- Shifting by two positions is the same as performing a one-position shift two times. Shifting by zero positions leaves the pattern unchanged. Shifting an N-bit pattern left by N or more positions changes all of the bits to zero.
- The picture shows the operation performed on eight bits. The original pattern is 11010011. The resulting pattern is 10100110.



Right shift

- A shift right logical of one position moves each bit to the right by one. The high-order bit MSB is replaced by a zero bit and the low-order bit LSB move to CF (Carry Flag)
- Shifting by two positions is the same as performing a one-position shift two times. Shifting by zero positions leaves the pattern unchanged. Shifting an N-bit pattern right by N or more positions changes all of the bits to zero.
- The picture shows the operation performed on eight bits. The original pattern is 00010111. The resulting pattern is 00001011.



Arithmetic shift

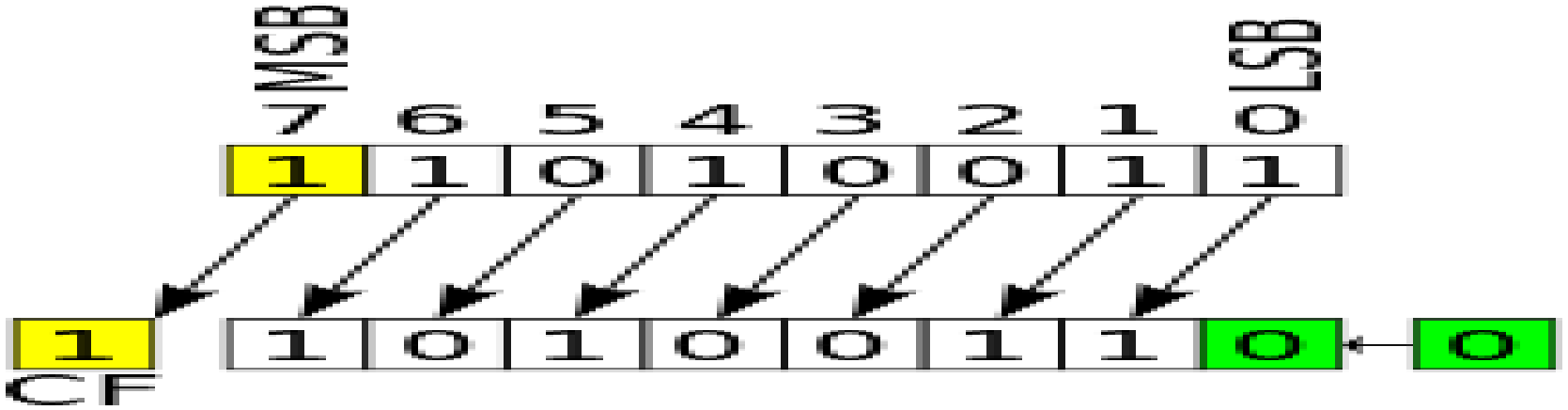
- Arithmetic shift operations assume that the bit pattern is a signed integer in two's complement format. Arithmetic right shift is used to divide an integer by two, while arithmetic left shift is used to multiply an integer by two.

  There are two kinds of arithmetic shift:

  1. Left arithmetic shift
  2. Right arithmetic shift

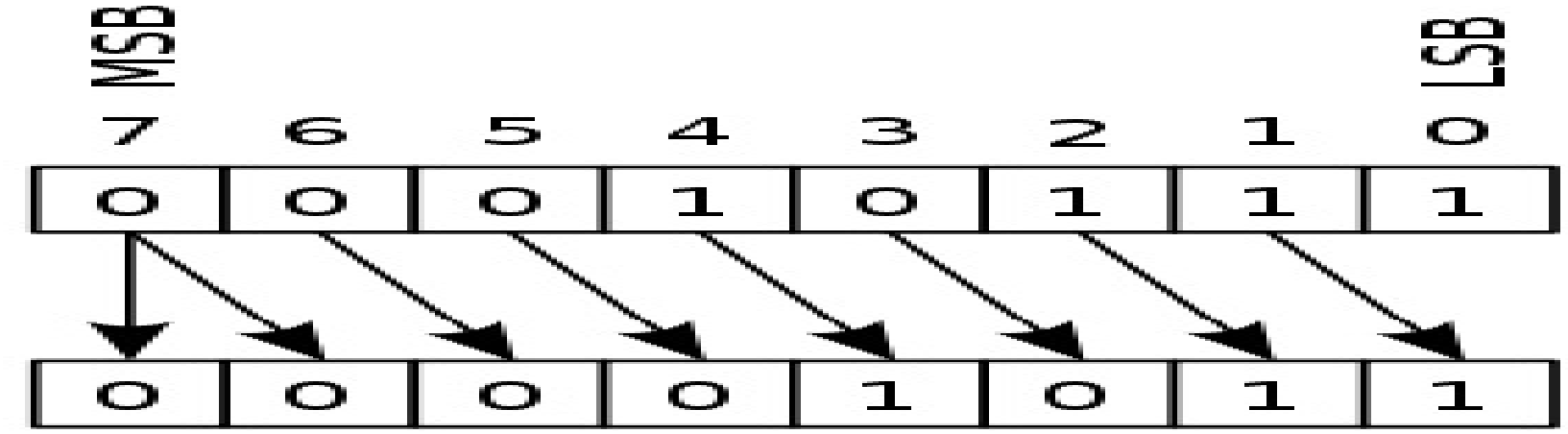
Left Arithmetic shift

- An shift arithmetic left is the same as a logical left shift
- A shift arithmetic left of one position moves each bit to the left by one. The low-order bit LSB is replaced by a zero bit and the high-order bit MSB move to CF (Carry Flag)
- A left arithmetic shift by *n* is equivalent to multiplying by 2^n . In 2's complement, positive or negative, a logical left shift, is equivalent to multiplication by two.
- The picture shows the operation performed on eight bits. The original pattern is 11010011. The resulting pattern is 10100110.



Shift Arithmetic Right

- A shift arithmetic right of one position moves each bit to the right by one. The high-order bit MSB is replaced by sign bit and the low-order bit LSB move to CF (Carry Flag)
- A shift arithmetic right is equivalent to integer division by two.
- In 2's complement, positive or negative, division by two is accomplished via an shift arithmetic right .
- The picture shows the operation performed on eight bits. The original pattern is 00010111. The resulting pattern is 00001011

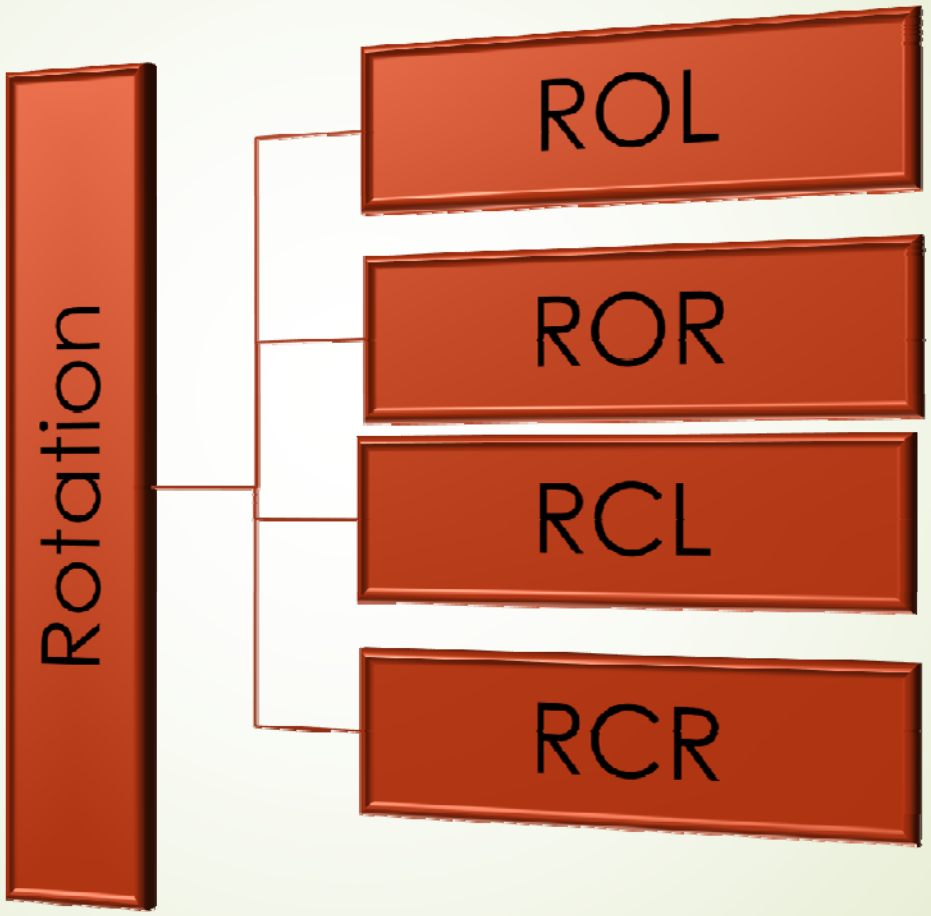


Rotation

Contents

- Types of Rotation
- ROL Instruction
- ROR Instruction
- Difference
- RCL Instruction
- RCR Instruction

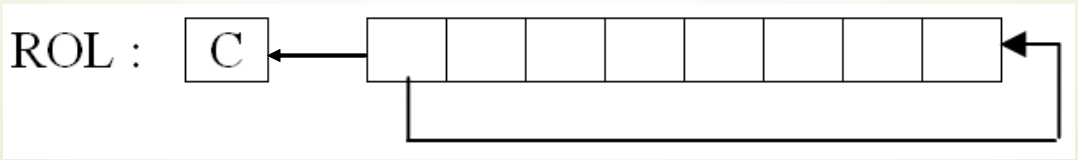
Types



ROL Instruction

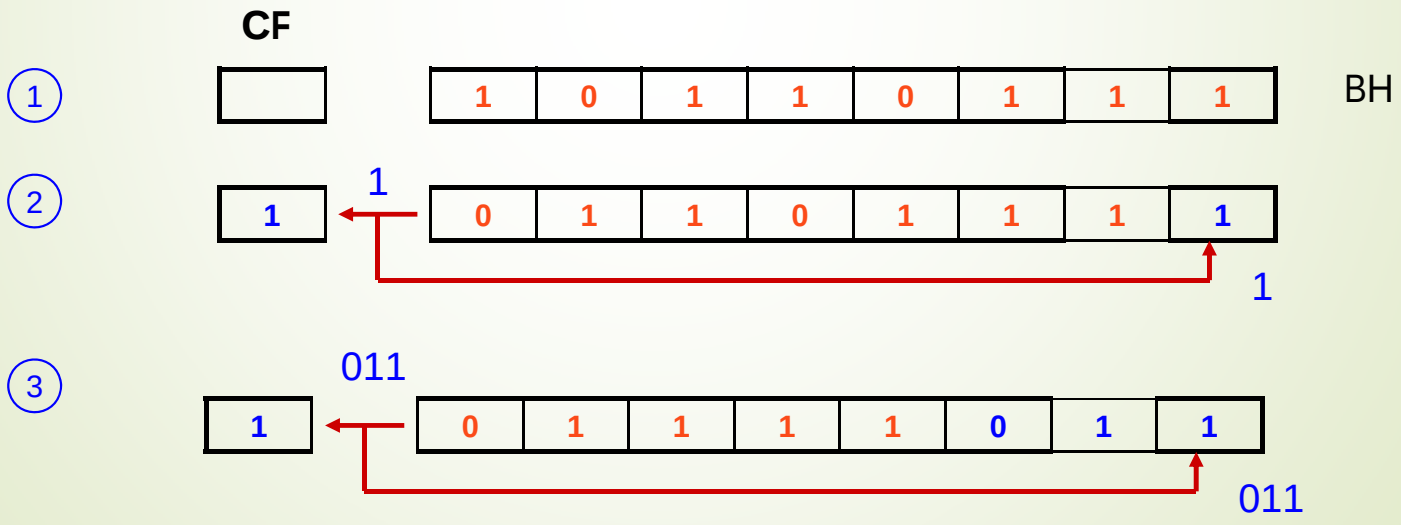
- ROL (rotate) shifts each bit to the left
- The highest bit is copied into both the Carry flag and into the lowest bit
- No bits are lost
- ROL is used for unsigned data

[label:]	ROL	register/memory, CL/*immediate*



Below are instances of the ROL instruction :

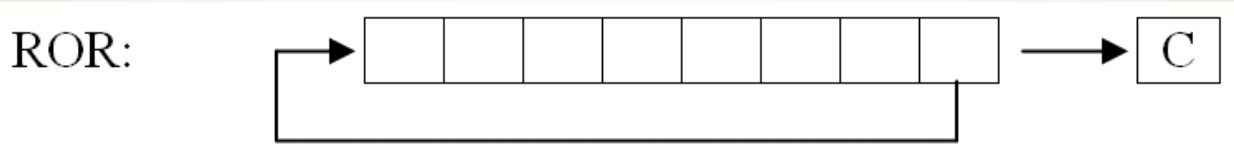
	Instruction	Comment	Binary	CF
1	MOV BL, 10110111B	; Initialize BH	10110111	
2	ROL BL, 01	; Rotate left 1	01101111	1
	MOV CL, 03	; Set rotate value		
3	ROL BL, CL	; Rotate left three more	01111011	1



ROR Instruction

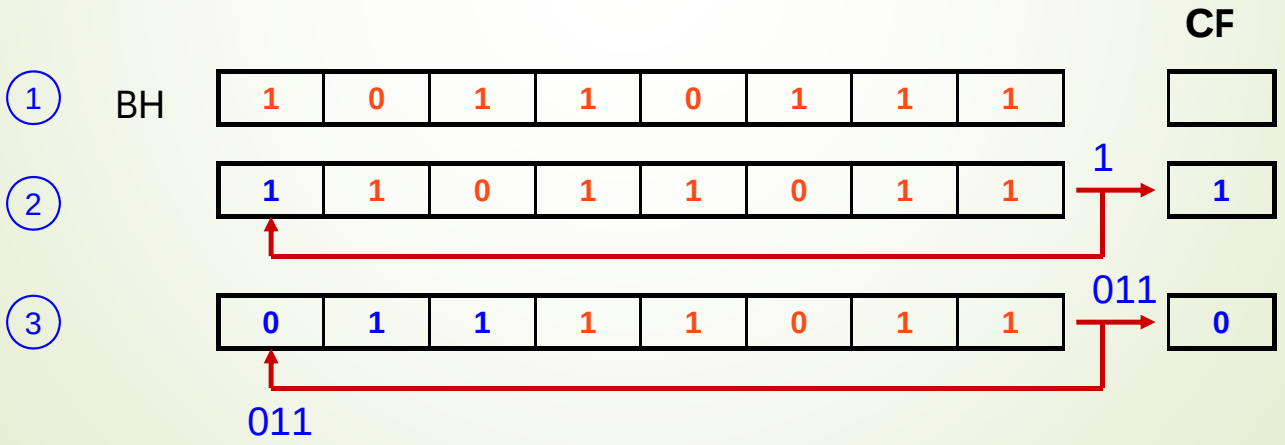
- ROR (rotate right) shifts each bit to the right
- The lowest bit is copied into both the Carry flag and into the highest bit
- No bits are lost
- ROR is for unsigned data

[label:]	ROR	register/memory, CL/*immediate*



A few examples on ROR:

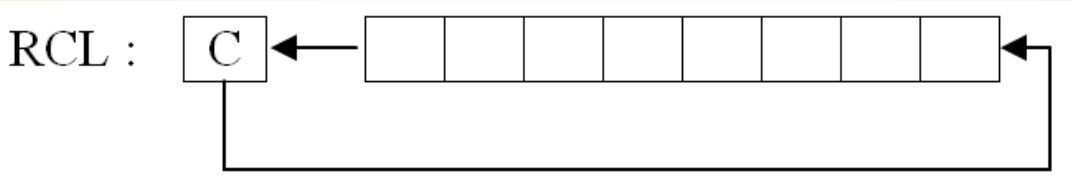
	Instruction	Comment	Binary	CF
1	MOV BL, 10110111B	; Initialize BH	10110111	
2	ROR BL, 01	; Rotate right 1	11011011	1
	MOV CL, 03	; Set rotate value		
3	ROR BL, CL	; Rotate right three more	01111011	0



RCL Instruction

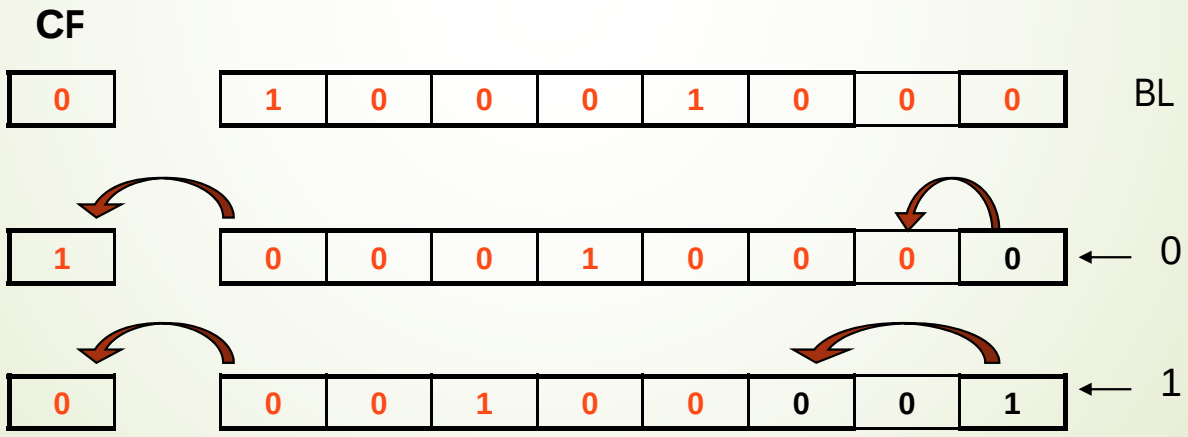
- RCL (rotate carry left) shifts each bit to the left
- Copies the Carry flag to the least significant bit
- Copies the most significant bit to the Carry flag
- RCL is for signed data

[label:]	RCL	register/memory, CL/*immediate*



Example:

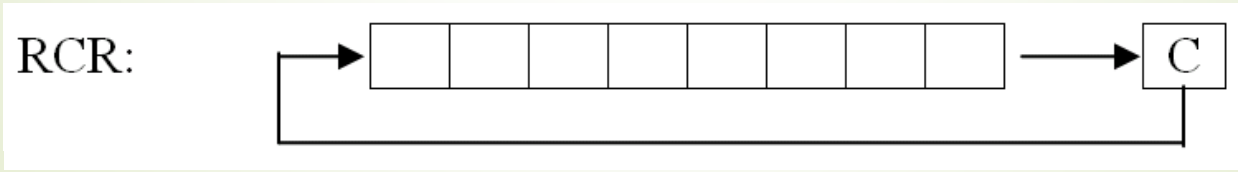
```
CLC                 ; CF = 0
MOV BL,88H          ; CF,BL = 0 10001000B
RCL BL,1            ; CF,BL = 1 00010000B
RCL BL,1            ; CF,BL = 0 00100001b
```



RCR Instruction

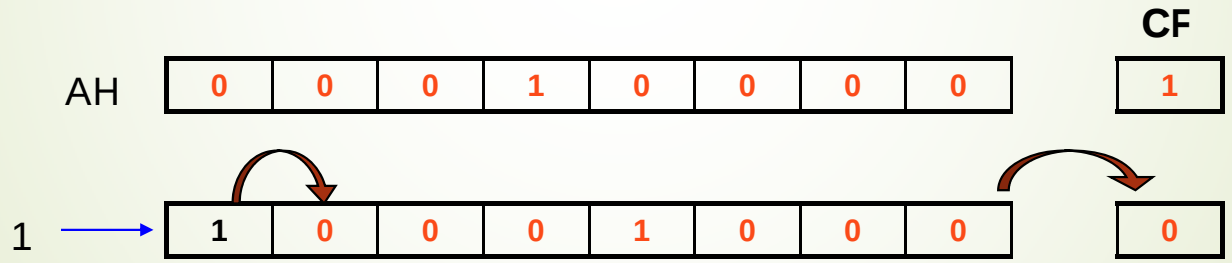
- RCR (rotate carry right) shifts each bit to the right
- Copies the Carry flag to the most significant bit
- Copies the least significant bit to the Carry flag

[label:]	RCR	register/memory, CL/*immediate*



Example:

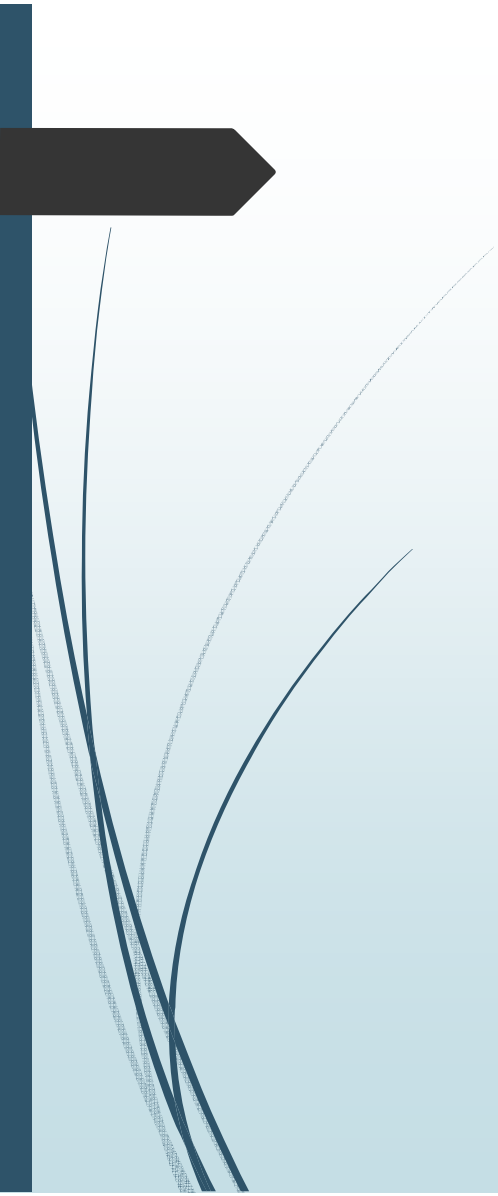
```
STC                     ; CF = 1
MOV AH,10H              ; CF,AH = 00010000 1
RCR AH,1                ; CF,AH = 10001000 0
```



Difference

- The difference between ROR and RCR is only the way of operation. In RCR, every bit that is rotated will enter the carry flag before entering the leftmost bit

- RCL works like just like ROL except that CF is part of the circle of bits being rotated.



The End