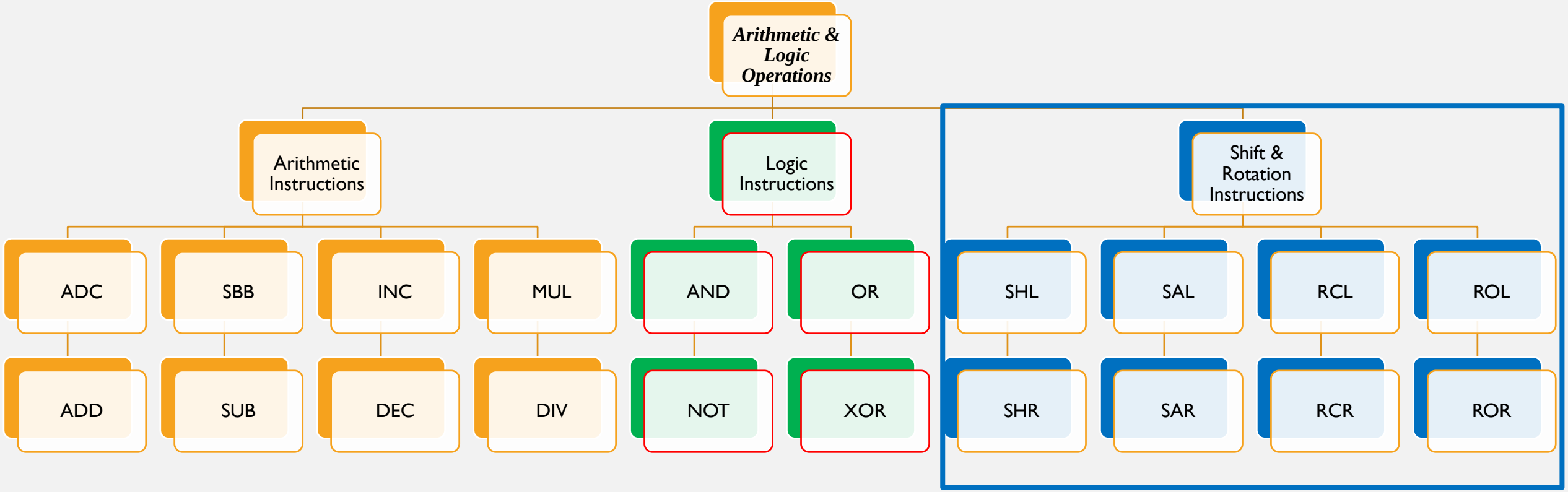
Shift & Rotation Instruction

REVIEW OF ASSEMBLY LANGUAGE

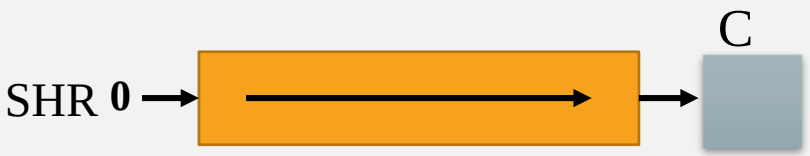
ARITHMETIC & LOGIC OPERATIONS



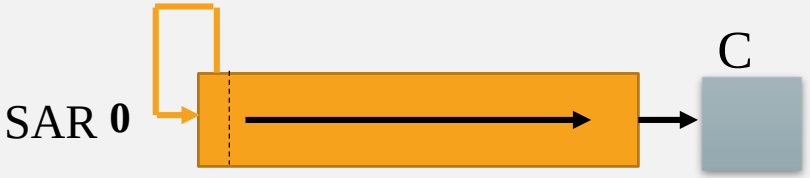
SHIFT INSTRUCTIONS

- Move (0) in the **Rightmost** of **unsigned** number
- Apply the Multiplication by 2 for **unsigned** number

- Move (0) in the **Rightmost** of **signed** number
- Apply the Multiplication by 2 for **signed** number



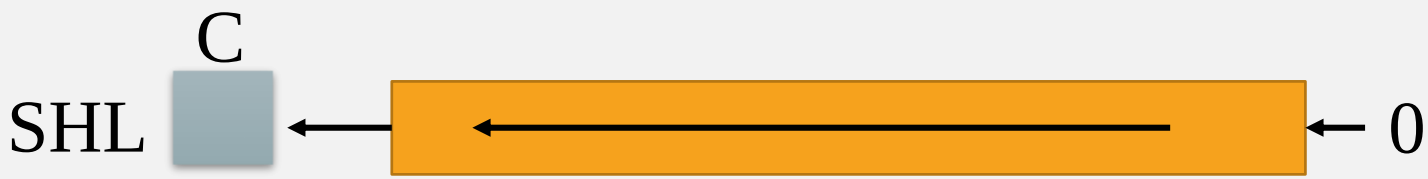
- Move (0) in the **leftmost** of **unsigned** number
- Apply the division by 2 for **unsigned** number



- Copy the Sign bit through the **signed** number
- Apply the division by 2 for **signed** number

LOGICAL SHIFT LEFT (SHL)

- Move (0) in the **Rightmost** of **unsigned** number
- Apply the Multiplication by 2 (by 2^{+n} [left shift]) for **unsigned** number



General Syntax

SHL Operand1, Count

Mem. , Immed.
Reg. , Immed.
Mem. , CL
Reg. , CL

Operand1 ← [B6 B5 B4 B3 B2 B1 B0 0]

CF ← B7

EX:

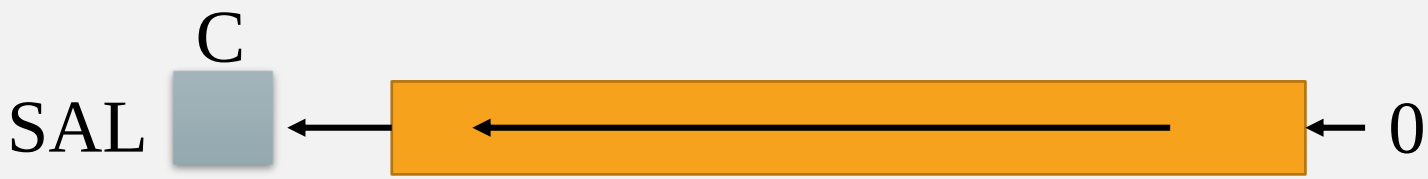
MOV DL,05 ; DL= 0101

SHL, DL,01 ; DL= 1010 ➔ DL=10

; 5 * 2^1= 5*2=10

SHIFT ARITHMETIC LEFT (SAL)

- Move (0) in the Rightmost of **signed** number
- Apply the Multiplication by 2 (by 2^{+n} [left shift]) for **signed** number



Operand1 ← [B6 B5 B4 B3 B2 B1 B0 0]

CF ← B7

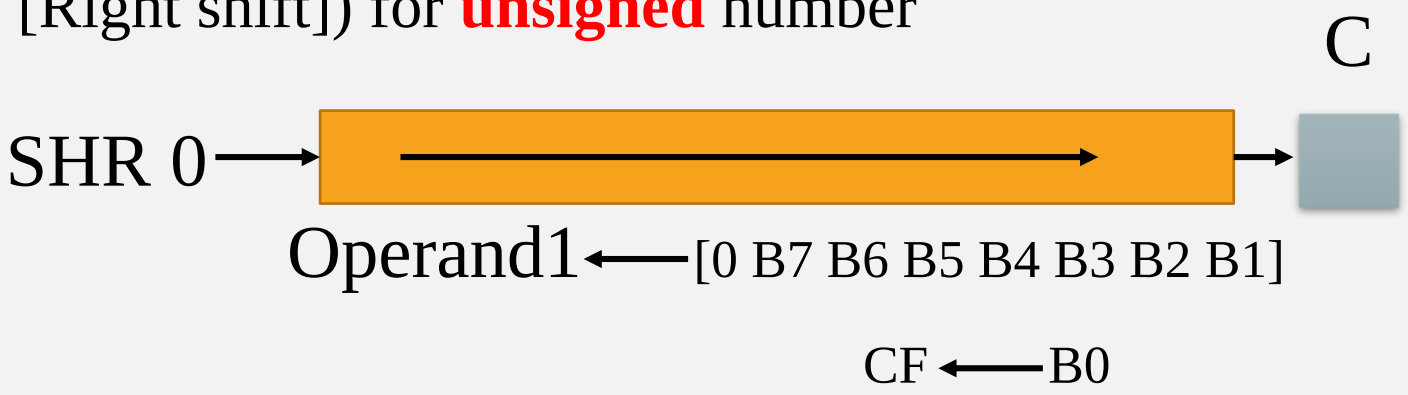
General Syntax

SAL Operand1, Count

Mem. , Immed.
Reg. , Immed.
Mem. , CL
Reg. , CL

LOGICAL SHIFT RIGHT (SHR)

- Move (0) in the leftmost of **unsigned** number
- Apply the division by 2 (by 2^{-n} [Right shift]) for **unsigned** number



General Syntax

SHR Operand1, Operand2

Mem. , Immed.
Reg. , Immed.
Mem. , CL
Reg. , CL

EX:

MOV DL,80 ; DL= 0101 0000

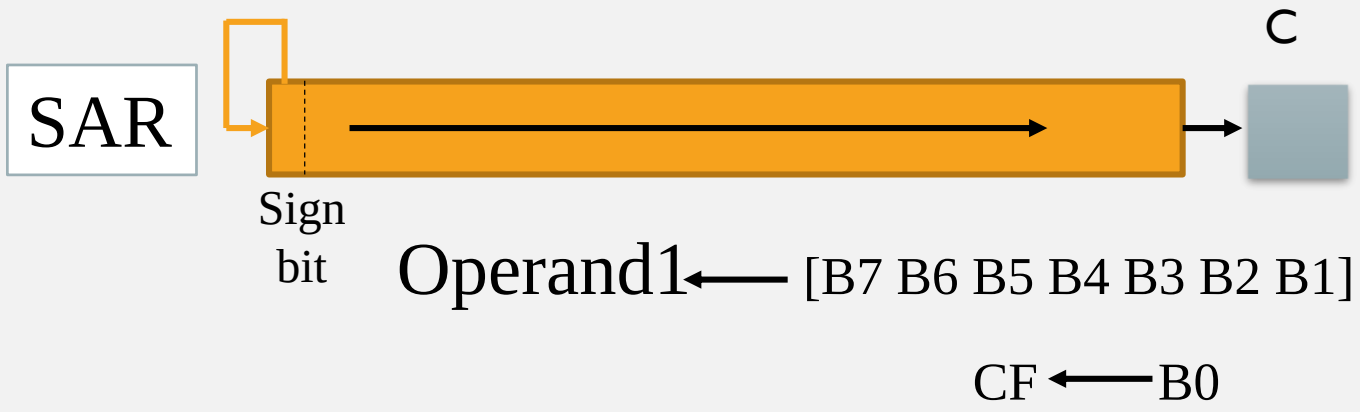
SHR, DL,01 ; DL= 0010 1000 ➔ DL=40

SHR, DL,02 ; DL= 0000 1010 ➔ DL=10

; $80 * 2^{-3} => \frac{80}{8} = 10$

SHIFT ARITHMETIC RIGHT (SAR)

- Copy the **sign** bit through the **signed** number
- Apply the Division by 2 for **signed** number



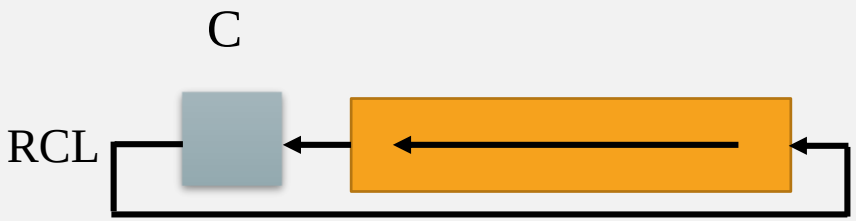
General Syntax

SAR Operand1, Count

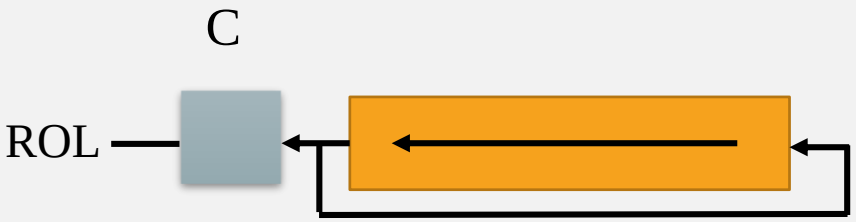
Mem. , Immed.
Reg. , Immed.
Mem. , CL
Reg. , CL

```
EX:
MOV  DL,-80  ; DL (+80) = 0101 0000 => 1011 0000b;
SAR  DL,1    ; DL = 1101 1000b = -40, CF = 0
SAR  DL,2    ; DL = 1111 0110b = -10, CF = 0
```

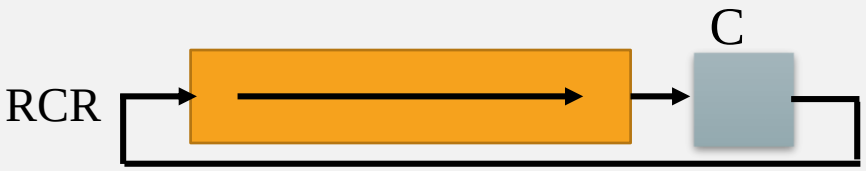
ROTATION INSTRUCTIONS



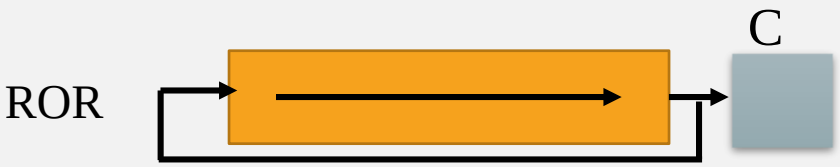
- **R**otate **C**arry **L**eft (**RCL**): Rotate left the bits **through** the CF**.**
- **MSB** goes into **CF** and **CF** goes into **LSB**.



- **Ro**tate **L**eft (**ROL**): Rotate left the bits into the CF.
- MSB goes into CF and MSB goes into LSB.



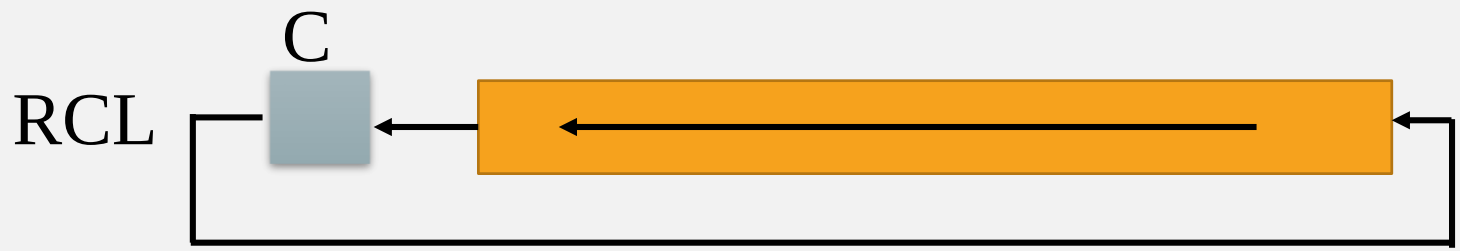
- **R**otate **C**arry **Right** (**RCR**): Rotate right the bits **through** the CF**.**
- **LSB** goes into **CF** and **CF** goes into MSB.



- **Ro**tate **Right** (**ROR**): Rotate right the bits into the CF.
- LSB goes into CF and LSB goes into MSB.

ROTATE CARRY LEFT (RCL)

- **R**otate **C**arry **L**eft (**RCL**): Rotate left the bits **through** the CF.
- MSB goes into CF and CF goes into LSB.



General Syntax

RCL Operand1, Count

Mem. , Immed.
Reg. , Immed.
Mem. , CL
Reg. , CL

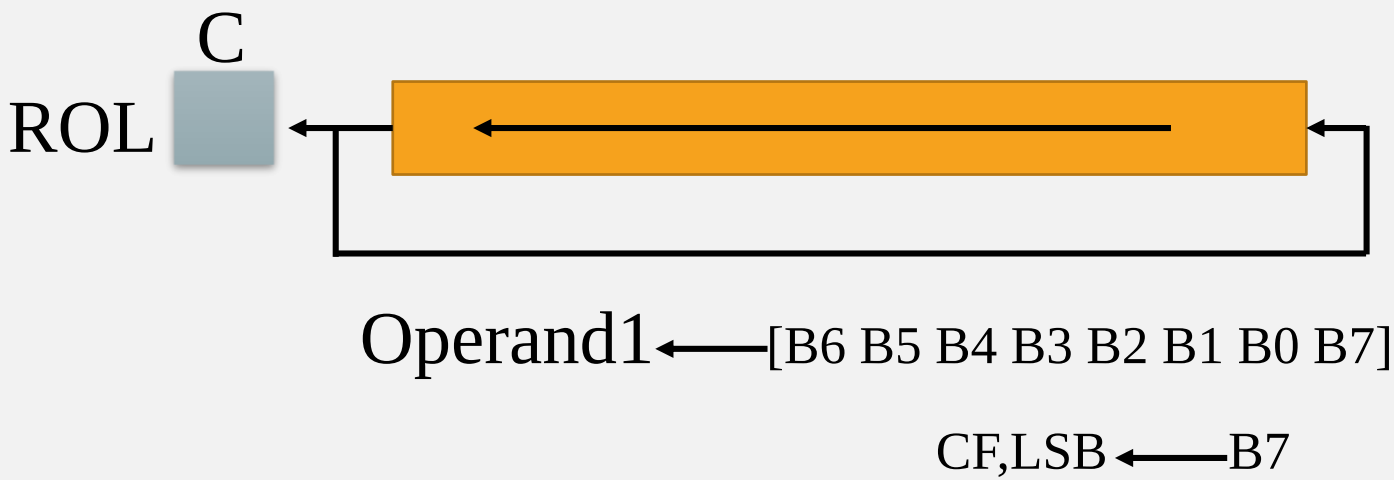
Operand1 ⟵ [B6 B5 B4 B3 B2 B1 B0 CF]

CF ⟵ B7

```
EX:
ClC                ; clear carry, CF = 0
MOV bl,88h         ;CF = 0 ,BL = 1000 1000b
RCL   bl,1         ;CF = 1 ,BL = 0001 0000b
RCL   bl,1         ;CF = 0 ,BL = 0010 0001b
```

ROTATE LEFT (ROL)

- **Ro**tate **L**eft (**ROL**): Rotate left the bits into the **CF**.
- **MSB** goes into **CF** and MSB goes into **LSB**.



General Syntax

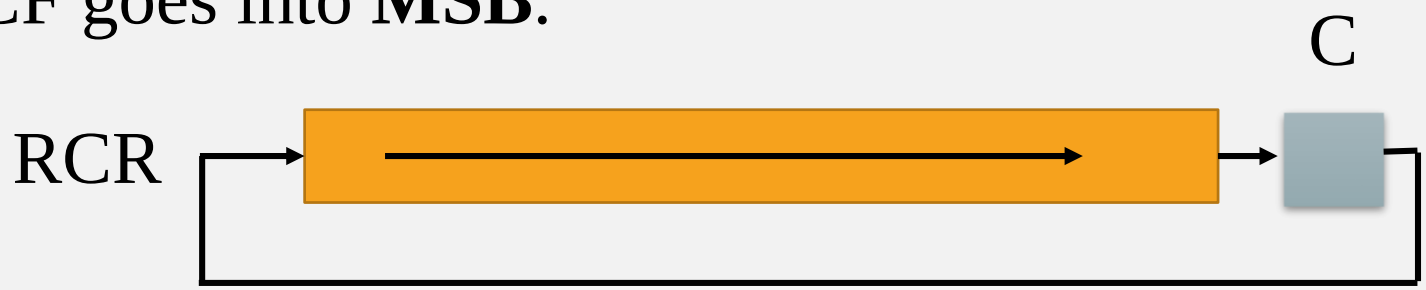
ROL Operand1, Count

Mem. , Immed.
Reg. , Immed.
Mem. , CL
Reg. , CL

```
EX:
MOV AL,11110000b        ; AL= 240
ROL  AL,1               ;AL =1110 0001b, CF = 1
```

ROTATE CARRY RIGHT (RCR)

- **R**otate **C**arry **R**ight (RCR): Rotate right the bits **through** the **CF**.
- LSB goes into **CF** and CF goes into **MSB**.



General Syntax

RCR Operand1, Operand2

Mem. , Immed.
Reg. , Immed.
Mem. , CL
Reg. , CL

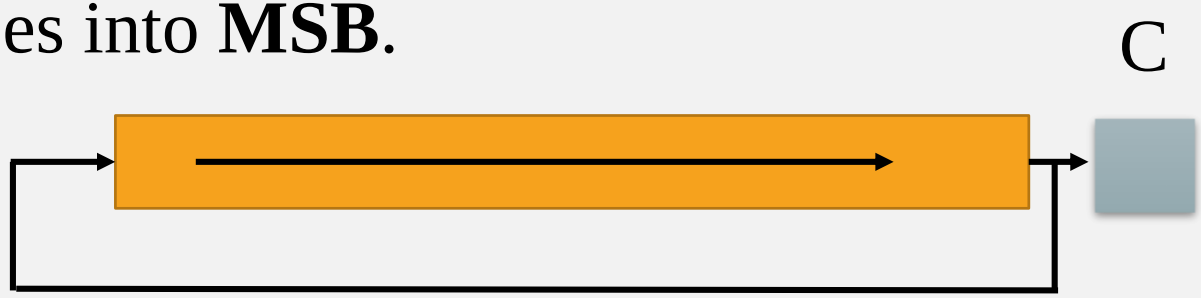
Operand1 ⟵ [CF B7 B6 B5 B4 B3 B2 B1]

CF ⟵ B0

```
EX:
STC              ; set carry, CF = 1
MOV ah,10h       ;CF = 1, AH = 0001 0000b
RCR  ah,1        ;CF = 0, AH = 1000 1000b
```

ROTATE RIGHT (ROR)

- **R**otate **C**arry **R**ight (RCR): Rotate right the bits **Into** the **CF**.
- LSB goes into **CF** and LSB goes into **MSB**.



General Syntax

ROR Operand1, Operand2

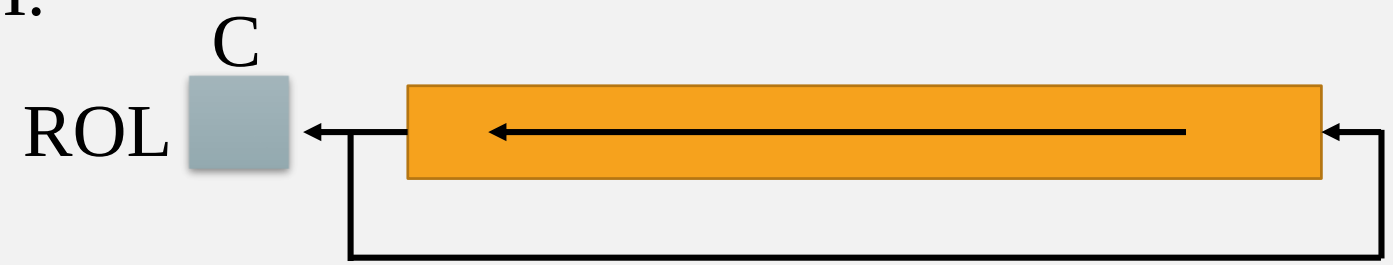
Mem. , Immed.
Reg. , Immed.
Mem. , CL
Reg. , CL

Operand1 ⟵ [B0 B7 B6 B5 B4 B3 B2 B1]

CF. MSB ⟵ B0

EXAMPLES

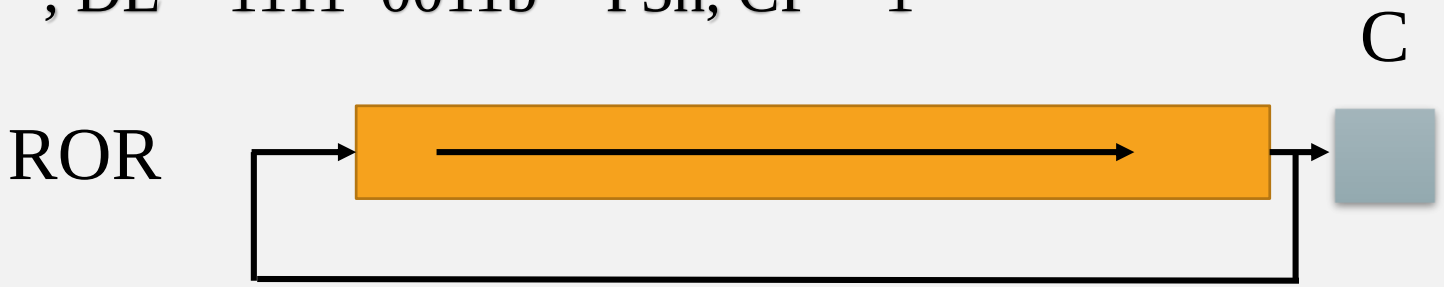
Ex01: Exchange the upper (bits 4–7) and lower (bits0–3) halves of a byte where the DL= 3FH.




```
MOV   DL,3Fh          ; DL =  0011 1111b
ROL   DL,4            ; DL = 1111  0011b = F3h, CF = 1
```

OR:

```
MOV   AL,3Fh          ; DL =  0011 1111b
ROR   DL,4            ; DL = 1111  0011b = F3h, CF = 1
```



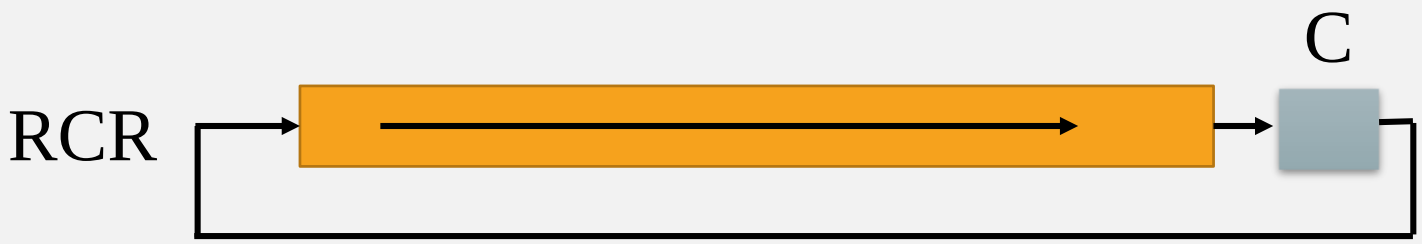
EXAMPLES

Ex02: Trace the following example and write down the values of each one.

Program	AL	BL	CL	DL
MOV AL, 10111111	10111111	?	?	?
MOV BL, 01010111	10111111	01010111	?	?
MOV CL, 0	10111111	01010111	0	?
AND AL,BL	00010111	01010111	0	?
OR AL, BL	01010111	01010111	0	?
JZ **EXIT**	01010111	01010111	0	?
NOT BL	01010111	10101000	0	?
AND AL,BL	00000000	10101000	0	?
JZ CHANGE				
MOV CL,2				
ROR AL, CL				
CHANGE: MOV DL,0	00000000	10101000	0	0
EXIT: MOV CL,1	00000000	10101000	1	0

EXAMPLES

Ex03: Make execution traces of the contents of each of the registers involved in the following program.



0	0000000010110111
Rotate with carry >>	5
1	0000000001011011
1	1000000000101101
1	1100000000010110
0	1110000000001011
1	0111000000000101

Program	BX	CL
MOV CL, 4		
CLC		
MOV BX, 1011 0111B		
RCR BX,1		
RCR BX,CL		

BX= 7005H

EXAMPLES

Ex04: Answer the following:

1. What does this program do?
2. What is the final result stored in BL register?

```
SUB BL,BL
MOV DL,8
MOV al, DATA1
AGAIN:
ROL al,1
JNC NEXT
INC BL
NEXT: DEC DL
JNZ AGAIN
HLT
DATA1 DB 97
```

THANK YOU