

BT Instruction

❖ BT = Bit Test Instruction

❖ Syntax:

BT *r/m16, r16*

BT *r/m32, r32*

BT *r/m16, imm8*

BT *r/m32, imm8*

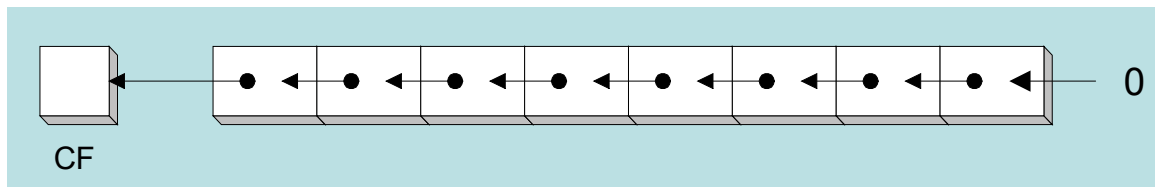
❖ Copies bit *n* from an operand into the Carry flag

❖ Example: jump to label L1 if bit 9 is set in AX register

```
bt AX, 9          ; CF = bit 9
jc L1             ; jump if Carry to L1
```

SHL Instruction

- ❖ SHL is the **Shift Left** instruction
 - ✧ Performs a logical left shift on the destination operand
 - ✧ Fills the lowest bit with **zero**
 - ✧ The **last bit shifted out from the left** becomes the **Carry Flag**



- ❖ Operand types for SHL:

```
SHL reg,imm8  
SHL mem,imm8  
SHL reg,CL  
SHL mem,CL
```

The shift **count** is either:

8-bit immediate *imm8*, or
stored in register *CL*

Only least sig. 5 bits used

Fast Multiplication

Shifting left 1 bit multiplies a number by 2

```
mov dl,5  
shl dl,1
```

Before: 0 0 0 0 0 1 0 1 = 5
After: 0 0 0 0 1 0 1 0 = 10

Shifting left n bits multiplies the operand by 2^n

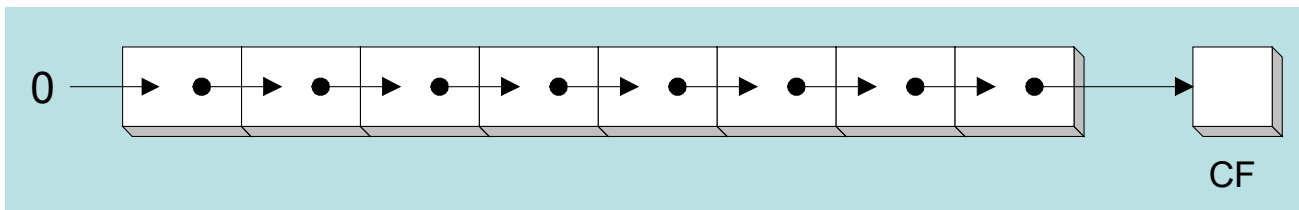
For example, $5 * 2^2 = 20$

```
mov dl,5    ; DL = 00000101b  
shl dl,2    ; DL = 00010100b = 20, CF = 0
```

SHR Instruction

❖ SHR is the **Shift Right** instruction

- ✧ Performs a logical right shift on the destination operand
- ✧ The highest bit position is filled with a **zero**
- ✧ The **last bit shifted out from the right** becomes the **Carry Flag**
- ✧ SHR uses the same instruction format as SHL



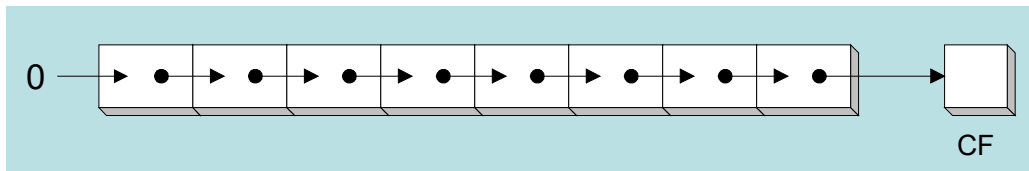
❖ **Shifting right** n bits **divides** the operand by 2^n

```
mov dl,80    ; DL = 01010000b
shr dl,1     ; DL = 00101000b = 40, CF = 0
shr dl,2     ; DL = 00001010b = 10, CF = 0
```

Logical versus Arithmetic Shifts

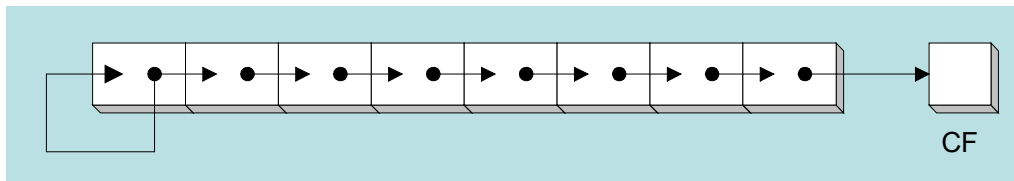
❖ Logical Shift

- ✧ Fills the newly created bit position with **zero**



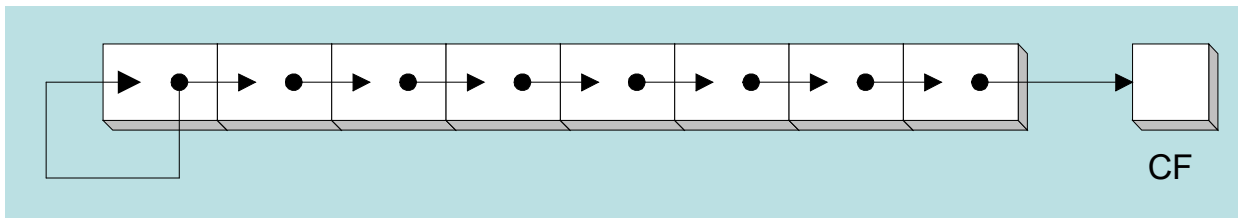
❖ Arithmetic Shift

- ✧ Fills the newly created bit position with a **copy of the sign bit**
- ✧ Applies only to **Shift Arithmetic Right** (SAR)



SAL and SAR Instructions

- ❖ SAL: **Shift Arithmetic Left** is identical to SHL
- ❖ SAR: **Shift Arithmetic Right**
 - ✧ Performs a right arithmetic shift on the destination operand



- ❖ SAR preserves the number's sign

```
mov dl,-80      ; DL = 10110000b
sar dl,1        ; DL = 11011000b = -40, CF = 0
sar dl,2        ; DL = 11110110b = -10, CF = 0
```

Effect of Shift Instructions on Flags

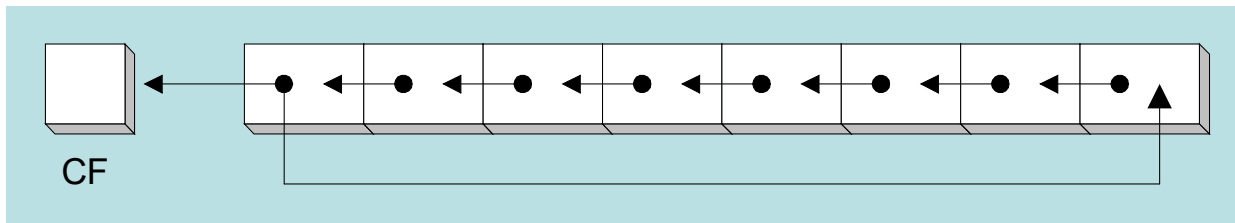
- ❖ The **CF** is the last bit shifted
- ❖ The **OF** is defined for single bit shift only
 - ✧ It is 1 if the sign bit changes
- ❖ The **ZF**, **SF** and **PF** are affected according to the result
- ❖ The **AF** is unaffected

ROL Instruction

❖ ROL is the **Rotate Left** instruction

- ✧ Rotates each bit to the left, according to the count operand
- ✧ Highest bit is copied into the Carry Flag and into the Lowest Bit

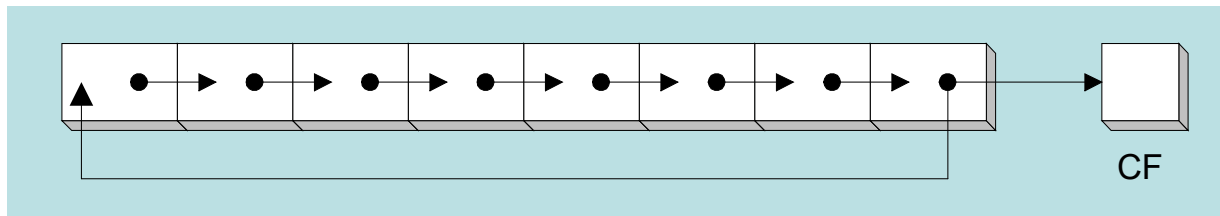
❖ No bits are lost



```
mov al,11110000b
rol al,1          ; AL = 11100001b, CF = 1
mov dl,3Fh        ; DL = 00111111b
rol dl,4          ; DL = 11110011b = F3h, CF = 1
```


ROR Instruction

- ❖ ROR is the **Rotate Right** instruction
 - ✧ Rotates each bit to the right, according to the count operand
 - ✧ Lowest bit is copied into the Carry flag and into the highest bit
- ❖ No bits are lost



```
mov al,11110000b
ror al,1           ; AL = 01111000b, CF = 0
mov dl,3Fh        ; DL = 00111111b
ror dl,4          ; DL = F3h, CF = 1
```