Integer Arithmetic
(Shift and Rotate Instructions)

Contents of Lecture

- Shift and Rotate instructions
- SHL Instruction
- SHR Instruction
- SAL and SAR Instructions:
- ROL Instruction
- ROR Instruction
- RCL and RCR Instruction
- **Shift and Rotate Applications(EXERCISES)**

References for this lecture

Assembly Language for x86 Processors (7th Edition). Chapter 7, Integer Arithmetic

Shift and Rotate instructions:

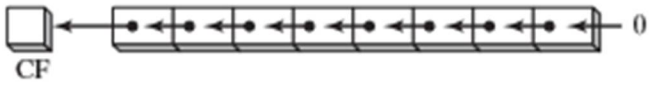
- Bit shifting means to move bits right and left inside an operand.
- All Shift and Rotate instructions affecting the Overflow and Carry flags

Shift and Rotate Instructions.

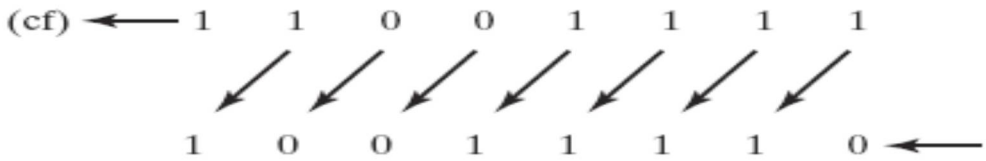
SHL	Shift left
SHR	Shift right
SAL	Shift arithmetic left
SAR	Shift arithmetic right
ROL	Rotate left
ROR	Rotate right
RCL	Rotate carry left
RCR	Rotate carry right
SHLD	Double-precision shift left
SHRD	Double-precision shift right

SHL Instruction:

- The SHL (shift left) instruction performs a logical left shift on the destination operand, filling the lowest bit with 0. The highest bit is moved to the Carry flag, and the bit that was in the Carry flag is discarded.



- **Example: If you shift 11001111 left by 1 bit**



- **Syntax:**

  SHL destination, shift-count

  The first operand in SHL is the destination and the second is the shift count

- The following lists the types of operands permitted by this instruction:
  - ✓ SHL reg, imm8
  - ✓ SHL mem, imm8
  - ✓ SHL reg, CL
  - ✓ SHL mem, CL

- X86 processors permit imm8 to be any integer between 0 and 255.
- The CL register can contain a shift count.
- Formats shown here also apply to the SHR, SAL, SAR, ROR, ROL, RCR, and RCL instructions.

- **Example:**

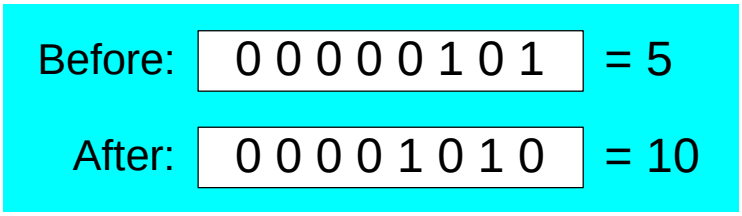
```
mov bl,8Fh          ; BL = 10001111b
shl bl,1            ; CF = 1, BL = 00011110b

mov al,10000000b
shl al,2            ; CF = 0, AL = 00000000b
```

Multiplication:

- Bitwise multiplication is performed when you shift a number's bits in a leftward direction (toward the MSB).
- For example, SHL can perform multiplication by powers of 2. Shifting any operand left by n bits multiplies the operand by 2^n.
- For example, shifting the integer 5 left by 1 bit yields the product of $5 * 2^1 = 10$:
- Commands:

```
Mov  dl,  5
shl  dl,  1
```

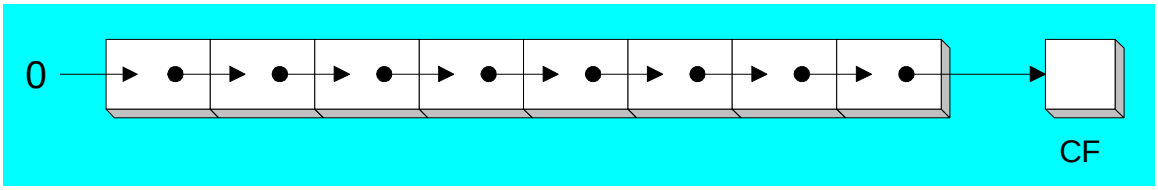


- If binary 00001010 (decimal 10) is shifted left by two bits, the result is the same as multiplying 10 by 2^2

```
mov  dl, 10     ; before:  00001010
shl  dl,  2     ; after:   00101000
```

SHR Instruction:

- The SHR (shift right) instruction performs a logical right shift on the destination operand, replacing the highest bit with a 0. The lowest bit is copied into the Carry flag, and the bit that was previously in the Carry flag is lost



- SHR uses the same instruction formats as SHL.

```
SHR    destination, shift-count
```

- In the following example, the 0 from the lowest bit in AL is copied into the Carry flag, and the highest bit in AL is filled with a zero:

```
mov  al,0D0h          ; AL = 11010000b
shr  al,1             ; AL = 01101000b, CF = 0
```

- In a multiple shift operation, the last bit to be shifted out of position 0 (the LSB) ends up in the Carry flag:

```
mov  al,00000010b
shr  al,2             ; AL = 00000000b, CF = 1
```

Bitwise division:

- Bitwise division is accomplished when you shift a number's bits in a rightward direction (toward the LSB). Shifting an unsigned integer right by n bits divides the operand by 2^n.
- In the following statements, we divide 32 by 2^1, producing 16:

```
mov  dl,32
shr  dl,1
```

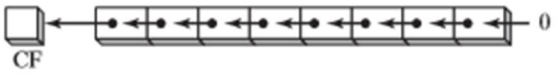
Before: 0 0 1 0 0 0 0 0 = 32
After: 0 0 0 1 0 0 0 0 = 16

- In the following example, 64 is divided by 2^3:

```
mov al,01000000b        ; AL = 64
shr al,3                ; divide by 8, AL = 00001000b
```

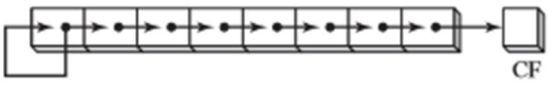
SAL and SAR Instructions:

- The SAL (shift arithmetic left) instruction works the same as the SHL instruction.
- For each shift count, SAL shifts each bit in the destination operand to the next highest bit position. The lowest bit is assigned 0. The highest bit is moved to the Carry flag, and the bit that was in the Carry flag is discarded:



- If you shift binary 11001111 to the left by one bit, it becomes 10011110:

- The SAR (shift arithmetic right) instruction performs a right arithmetic shift on its destination operand:



- The operands for SAL and SAR are identical to those for SHL and SHR. The shift may be repeated, based on the counter in the second operand:

```
SAR destination, count
```

- The following example shows how SAR duplicates the sign bit. AL is negative before and after it is shifted to the right:

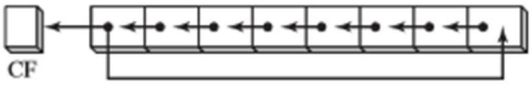
```
mov al,0F0h          ; AL = 11110000b (-16)
sar al,1             ; AL = 11111000b (-8), CF = 0
```

- **Signed Division** You can divide a signed operand by a power of 2, using the SAR instruction. In the following example, –128 is divided by 2^3. The quotient is –16:

```
mov dl,-128          ; DL = 10000000b
sar dl,3             ; DL = 11110000b
```

ROL Instruction:

- Bitwise rotation occurs when you move the bits in a circular fashion.
- The ROL (rotate left) instruction shifts each bit to the left. The highest bit is copied into the Carry flag and the lowest bit position.



- The instruction format is the same as for SHL.
- Bit rotation does not lose bits. A bit rotated off one end of a number appears again at the other end. Note in the following example how the high bit is copied into both the Carry flag and bit position 0:

```
mov   al,40h         ; AL = 01000000b
rol   al,1           ; AL = 10000000b, CF = 0
rol   al,1           ; AL = 00000001b, CF = 1
rol   al,1           ; AL = 00000010b, CF = 0
```

Multiple Rotations:

- When using a rotation count greater than 1, the Carry flag contains the last bit rotated out of the MSB position:

```
mov   al,00100000b
rol   al,3           ; CF = 1, AL = 00000001b
```

Exchanging Groups of Bits:

- You can use ROL to exchange the upper (bits 4–7) and lower (bits 0–3) halves of a byte.
- For example, 26h rotated four bits in either direction becomes 62h:

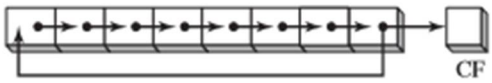
```
mov   al,26h
rol   al,4           ; AL = 62h
```

- When rotating a multibyte integer by four bits, the effect is to rotate each hexadecimal digit one position to the right or left. Here, for example, we repeatedly rotate 6A4Bh left four bits, eventually ending up with the original value:

```
mov   ax,6A4Bh
rol   ax,4        ; AX = A4B6h
rol   ax,4        ; AX = 4B6Ah
rol   ax,4        ; AX = B6A4h
rol   ax,4        ; AX = 6A4Bh
```

ROR Instruction:

- The ROR (rotate right) instruction shifts each bit to the right and copies the lowest bit into the Carry flag and the highest bit position.
- The instruction format is the same as for SHL



- In the following examples, note how the lowest bit is copied into both the Carry flag and the highest bit position of the result:

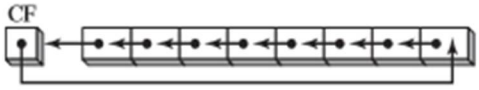
```
mov   al,01h      ; AL = 00000001b
ror   al,1        ; AL = 10000000b, CF = 1
ror   al,1        ; AL = 01000000b, CF = 0
```

- **Multiple Rotations** When using a rotation count greater than 1, the Carry flag contains the last bit rotated out of the LSB position:

```
mov   al,00000100b
ror   al,3        ; AL = 10000000b, CF = 1
```

RCL and RCR Instruction:

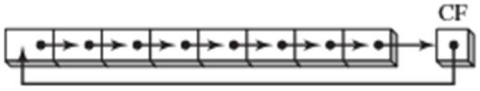
- The **RCL** (rotate carry left) instruction shifts each bit to the left, copies the Carry flag to the LSB, and copies the MSB into the Carry flag:



- Example:

```
mov bl,88h        ; CF=0,BL =  10001000b
rcl bl,1          ; CL=1,BL =  00010000b
rcl bl,1          ; CF=0,BL =  00100001b
```

- **RCR** Instruction The RCR (rotate carry right) instruction shifts each bit to the right, copies the Carry flag into the MSB, and copies the LSB into the Carry flag:



- Example:

```
mov ah,10h    ; CF=1,AH =  00010000b
rcr ah,1      ; CF=0,AH =  10001000b
```

Example:

Write a program count the numbers of bits containing one in register BX without changing the value of BX, save the number of ones in register AX.
For example:

BX = 1100 0101 1010 0011
AX = 8

Assembly code:

Shift and Rotate Applications (EXERCISES):

1. Input binary numbers from keyboard:

Convert the following algorithm to an assembly program that allow user to enter binary numbers by using logical operations.

```
Clear  BX ( BX will hold Binary values )
        Input a character (  '0'  OR '1')
        While  character < > CR DO
                Convert character to binary value
                Left shift BX
                Insert value into LSB of BX
                Input a character
End_While
```

✓ **How algorithm work if user enter 110:**

Clear BX :	BX = 0000 0000 0000 0000
Input character '1' , convert to 1	
Left shift BX:	BX = 0000 0000 0000 0000
Insert value into LSB of BX:	BX = 0000 0000 0000 0001
Input character '1', convert to 1	
Left shift BX:	BX = 0000 0000 0000 0010
Insert value into LSB of BX:	BX = 0000 0000 0000 0011
Input character '0' , convert to 0	
left shift BX :	BX = 0000 0000 0000 0110
Insert value into LSB of BX	
BX = 0000 0000 0000 0110	

//

2. Print binary number to screen:

Convert the following algorithm to an assembly program that allow user to print out binary numbers by using logical operations.

```
FOR 16 times Do
                Rotate left BX
                If   CF = 1 then
                        Output '1'
                else
                        Output   '0'
                end - if
END_FOR
```

3. Input hexadecimal numbers from keyboard:

Convert the following algorithm to an assembly program that allow user to enter hexadecimal numbers by using logical operations.

```
Clear       BX
            Input  Hex character
            While character <> CR Do
                  Convert character to Binary value
                  Left shift BX 4 Times
                  Insert value into lower 4 bits of BX
                  input a character
End_While
```

//

4. Print hexadecimal number to screen:

Write a program allow user to print out hexadecimal numbers by using logical operations.

```
For 4 times do
      MOV BH to DL
      Shift DL 4 times to Right
      If DL < 10   then
            Convert to character in 0 .....9
      Else
            Convert to character in A.....F
      End if
      Output character
      Rotate BX left 4 times
END For
```