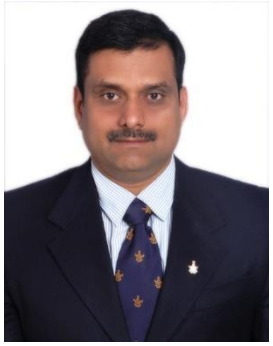
Microprocessors and Interfaces: 2021-22
Lecture 15
8086 Logical Instructions : Part-2

By Dr. Sanjay Vidhyadharan



Logical Instructions

The logic instructions include

- AND
- OR
- Exclusive-OR
- NOT
- NEG
- Shifts
- Rotates
- TEST (logical compare).

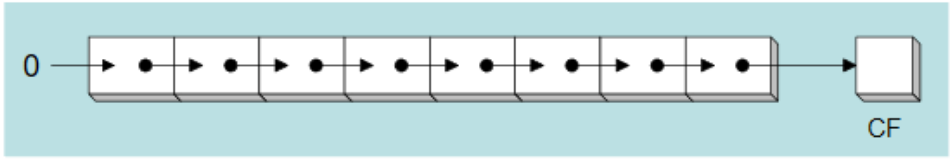
Shift

- Position or move numbers to the left or right within a register or memory location.
  - also perform simple arithmetic as multiplication by powers of 2^{+n} (left shift) and division by powers of 2^{-n} (right shift).
- The microprocessor's instruction set contains four different shift instructions:
  - two are **logical** (**SHL, SHR**);
  - two are **arithmetic** shifts (**SAL, SAR**)

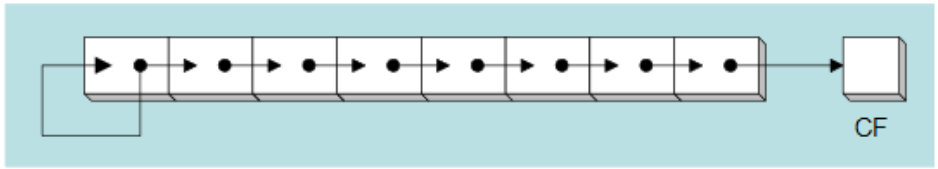
Shift

Logical vs Arithmetic Shifts

- A logical shift fills the newly created bit position with zero:



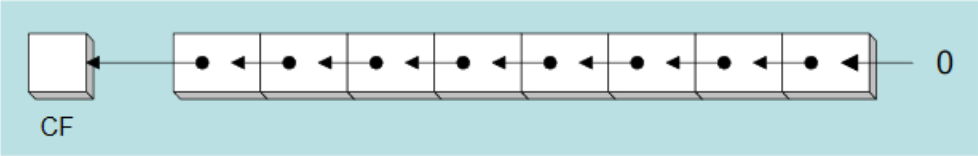
- An arithmetic shift fills the newly created bit position with a copy of the number's sign bit:



Shift

SHL Instruction

- The SHL (shift left) instruction performs a logical left shift on the destination operand, filling the lowest bit with 0.



- Operand types for SHL: SHL *destination,count*

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

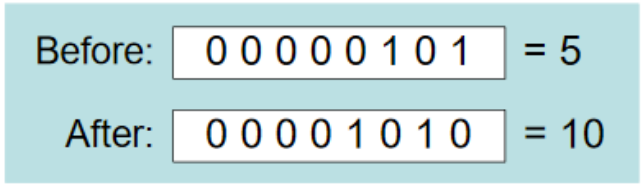
(Same for all shift and rotate instructions)

Shift

Fast Multiplication

Shifting left 1 bit multiplies a number by 2

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



Shifting left *n* bits multiplies the operand by 2^n

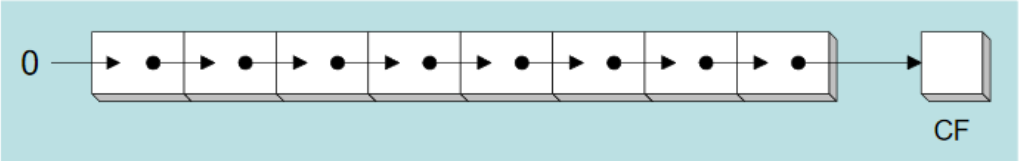
For example, $5 * 2^2 = 20$

```
mov dl,5
shl dl,2                    ; DL = 20
```

Shift

SHR Instruction

- The SHR (shift right) instruction performs a logical right shift on the destination operand. The highest bit position is filled with a zero.



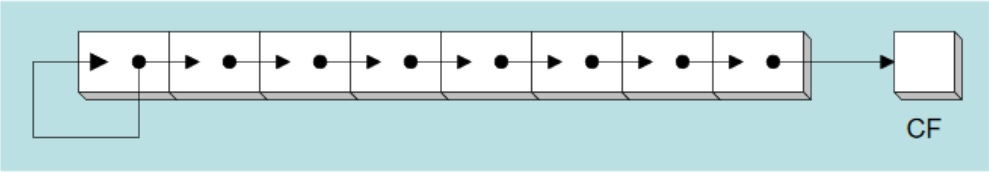
Shifting right *n* bits divides the operand by 2^n

```
mov dl,80
shr dl,1                    ; DL = 40
shr dl,2                    ; DL = 10
```

Shift

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.



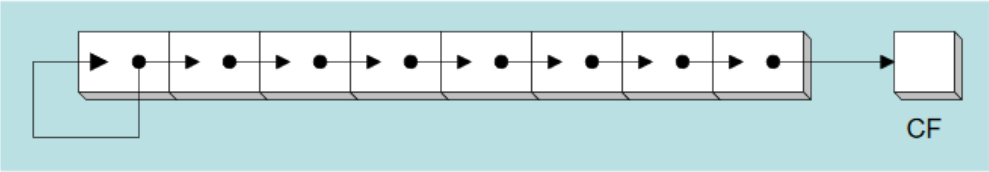
An arithmetic shift preserves the number's sign.

```
mov dl,-80
sar dl,1                    ; DL = -40
sar dl,2                    ; DL = -10
```

Shift

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.



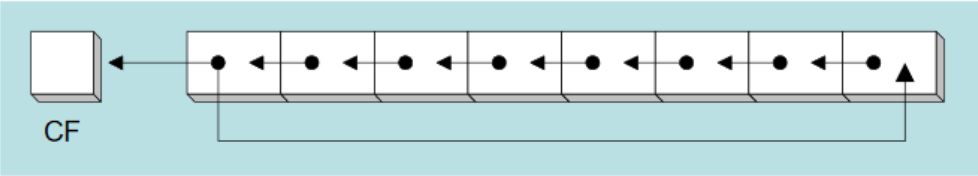
An arithmetic shift preserves the number's sign.

```
mov dl,-80
sar dl,1                    ; DL = -40
sar dl,2                    ; DL = -10
```

Rotate

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



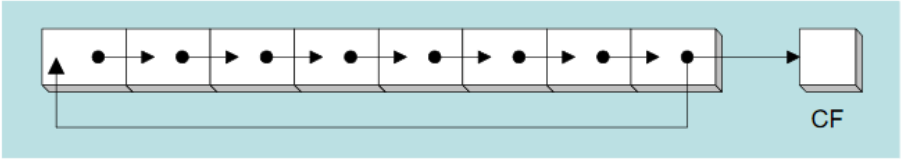

```
mov al,11110000b
rol al,1                    ; AL = 11100001b

mov dl,3Fh
rol dl,4                    ; DL = F3h
```

Flags Affected : CF
OF –If MSB changes –single bit rotate

Rotate

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




```
mov al,11110000b
ror al,1                    ; AL = 01111000b

mov dl,3Fh
ror dl,4                    ; DL = F3h
```

Flags Affected : CF
OF –If MSB changes –single bit rotate

Rotate

- ROL Destination, count
- ROR Destination, count

- Use CL for count greater than 1.
- (In 80386 onwards count greater than 1 can be directly given)

- ROL AX,1
- ROR BYTEPTR [SI], 1
- MOV CL, 04H
- ROL AX, CL
- ROL BYTEPTR [SI], CL
- ROL ECX, 12H 80386

Rotate

ASSUME BX=1111 0000 1001 1100, Swap the bytes of the BX register

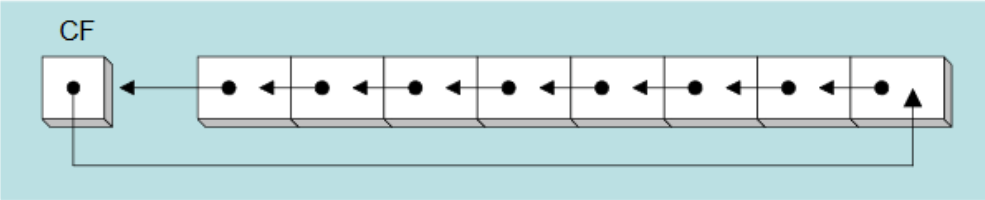
```
MOV CL, 08H
ROL BX, CL
```

```
MOV CL, 08H
ROR BX, CL
```

Rotate through Carry

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

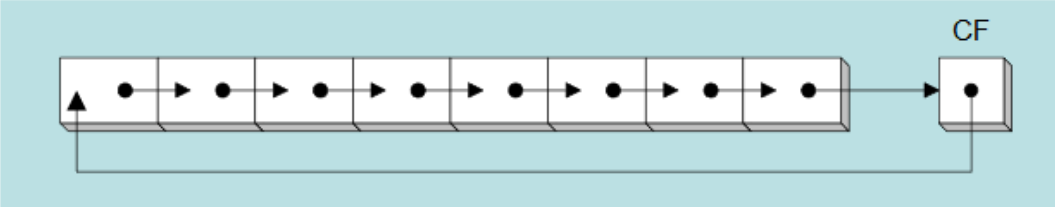



```
clc                         ; CF = 0
mov bl,88h                  ; CF,BL = 0 10001000b
rcl bl,1                    ; CF,BL = 1 00010000b
rcl bl,1                    ; CF,BL = 0 00100001b
```

Rotate through Carry

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




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

WRITE A PROGRAM THAT COUNTS THE NUMBER OF 1'S IN A BYTE IN LOCATION DATA1 AND WRITES IT INTO LOCATION RES1

```
        .Model Tiny
        .data
        DATA1 DB 0A7H
        RES1    DB    ?
        .code
        .startup
        SUB BL, BL                  ;clear BL & Carry Flag
        MOV DL, 8                   ;rotate total of 8 times
         MOV AL,DATA1
AGAIN: ROL AL,1                     ;rotate it once
        JNC NEXT                     ;check for 1
        INC BL                      ;if CF=1 then inc count
NEXT:   DEC DL                       ;go through this 8 times
        JNZ AGAIN                   ;if not finished go back
        MOV  RES1, BL
.exit
end
```

STRING COMPARISONS

- String instructions are powerful because they allow the programmer to manipulate large blocks of data with relative ease.
- Block data manipulation occurs with MOVS, LODS, STOS, INS, and OUTS.
- Additional string instructions allow a section of memory to be tested against a constant or against another section of memory.
  - **SCAS (string scan); CMPS (string compare)**

SCAS

Compares the AL with a byte of data in memory
Compares the AX with a word of data in memory
Compares the EAX with a doubleword of data in memory

Memory is ES: DI
Operands not affected flags affected(subtraction)

SCASB
SCASW
SCASD

Can be used with prefix
REPNE SCASB

- SCAS uses direction flag (D) to select auto- increment or auto-decrement operation for DI.

also repeat if prefixed by conditional repeat prefix

Write an ALP to find the displacement at which the data 0DH is present from an array of data stored from location DAT 1. The number of bytes of data in the array is 80.

```
. MODEL TINY
.DATA
DAT1 DB 80 DUP (?)
.CODE
.STARTUP
        MOV  DI, OFFSET  TEST STRING
        MOV  AL, 0DH
        MOV CX, 50H
        CLD
        REPNE  SCASB
.EXIT
END
```

- Scanning is repeated as long as bytes are not equal or the end of the string not reached.
- If 0DH is found DI will point to the next address

CMPS / CMPSB/ CMPSW

- Compares a byte in one string with a byte in another string or a word in one string with a word in another string

DS: SI with ES: DI

- Flags affected
- Direction flag used for auto increment or decrement
- Can be used with Prefix

Ex:

```
MOV SI, OFFSET STRING FIRST
MOV DI, OFFSET STRING SECOND
CLD
MOV CX, 100
REPE CMPSB
```

- Repeat until end of string or until compared bytes are equal

- REPE CMPSB
- REPNE CMPSB

Thankyou