SHIFT INSTRUCTIONS

SHIFT Instruction

- There are two kinds of shifts:
  - Logical
  - Arithmetic
- The *logical shift* is for unsigned operands.
- The *arithmetic shift* is for signed operands.
- Using shift instructions shifts the contents of a register or memory location right or left.
- The number of times (or bits) that the operand is shifted can be specified directly if it is once only, or through the CL register if it is more than once.

SHIFT Instruction

SHR: Shift right

- This is the logical shift right.
- The operand is shifted right bit by bit, and for every shift the LSB will go to the carry flag and the MSB is filled with zero.

Example: Show the result of SHR in the following:

```
MOV AL, 9AH
MOV CL, 3
SHR AL, CL
```

Solution:

9AH	=	1 0 0 1 1 0 1 0	
		0 1 0 0 1 1 0 1	CF = 0 (shifted once)
		0 0 1 0 0 1 1 0	CF = 1 (shifted twice)
		0 0 0 1 0 0 1 1	CF = 0 (shifted three times)

After shifting right three times, AL = 13H and CF = 0.

SHIFT Instruction

SHR: Shift right

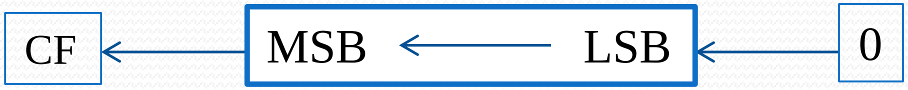
Example: Show the results of SHR in the following:

```
; from the data segment
TIMES       EQU   4
DATA1       DW    7777H
; from the code segment
MOV CL, TIMES
SHR DATA1, CL
```

Note: Solve by yourself and show the result.

SHIFT Instruction

SHL: Shift left



- Shift left is also a logical shift.
- It is the reverse of SHR.
- After every shift, the LSB is filled with zero and the MSB goes to CF.
- All the rules are the same as for SHR.

Example: Show the effects of SHL in the following:

```
MOV DH, 6
MOV CL, 4
SHL DH, CL
```

Solution:

	0 0 0 0 0 1 1 0	
CF = 0	0 0 0 0 1 1 0 0	(shifted left once)
CF = 0	0 0 0 1 1 0 0 0	
CF = 0	0 0 1 1 0 0 0 0	
CF = 0	0 1 1 0 0 0 0 0	(shifted left four times)

After the four shifts left, the DH register has 60H and CF = 0

COMPARE OF UNSIGNED NUMBERS

COMPARE of unsigned numbers

- The CMP instruction compares two operands and changes the flags according to the result of the comparison.
- The operands themselves remain unchanged.
- The destination operand can be in a register or in memory and the source operand can be in a register, in memory or immediate.
- All the CF, AF, SF, PF, ZF and OF flags reflect the result of the comparison.

Table 1: Flag settings for compare instruction

Compare Operands	CF	ZF
destination > source	0	0
destination = source	0	1
destination < source	1	0

COMPARE of unsigned numbers

Assume that there is a class of five people with the following grades: 69, 87, 96, 45 and 75. Find the highest grade.

```
        .MODEL SMALL
        .STACK 64
        .DATA
GRADES  DB      69, 87, 96, 45, 75
        ORG 0008H
HIGHEST DB      ?
        .CODE
MAIN    PROC FAR
        MOV AX, @DATA
        MOV DS, AX
        MOV CX, 5
        MOV BX, OFFSET GRADES
        SUB AL, AL
AGAIN:  CMP AL, [BX]
        JA NEXT ; CF=0 and ZF=0
        MOV AL, [BX]
NEXT:   INC BX
        LOOP AGAIN
        MOV HIGHEST, AL
        MOV AH, 4CH
        INT 21H
MAIN:   ENDP
        END MAIN
```

COMPARE of unsigned numbers

- Next example uses the CMP instruction to determine if an ASCII character is uppercase or lowercase.
- Note that small and capital letters in ASCII have the following values:

Letter	Hex	Binary	Letter	Hex	Binary
A	41	0100 0001	a	61	0110 0001
B	42	0100 0010	b	62	0110 0010
C	43	0100 0011	c	63	0110 0011
. . .	. . .	. . .	. . .	. . .	. . .
Y	59	0101 1001	y	79	0111 1001
Z	5A	0101 1010	z	7A	0111 1010

COMPARE of unsigned numbers

- As can be seen, there is a relationship between the pattern of lowercase and uppercase letters, as shown below for A and a:

A	0100 0001	41H
a	0110 0001	61H

- Notice that the only bit changes is ‘D5’. To change from lowercase to uppercase, D5 must be masked.
- Program first detects if the letter is in lowercase, and if it is, it is ANDed with 1101 1111 = DFH. Otherwise it is simply left alone.
- To determine if it is a lowercase letter, it is compared with 61H and 7AH to see if it is in the range *a* to *z*. Anything above or below this range should be left alone.

COMPARE of unsigned numbers

Lowercase to Uppercase Conversion

```
        .MODEL SMALL
        .STACK 64
        .DATA
DATA1   DB      ‘mY NAME is jOe’
        ORG 0020H
DATA2   DB      14 DUP (?)
        .CODE
MAIN    PROC FAR
        MOV AX, @DATA
        MOV DS, AX
        MOV SI, OFFSET DATA1
        MOV BX, OFFSET DATA2
        MOV CX, 14
```

```
BACK:   MOV AL, [SI]
        CMP AL, 61H
        JB OVER ; CF = 1
        CMP AL, 7AH
        JA OVER ; CF = 0 and ZF = 0
        AND AL, 11011111B
OVER:   MOV [BX], AL
        INC SI
        INC BX
        LOOP BACK
        MOV AH, 4CH
        INT 21H
MAIN    ENDP
        END MAIN
```

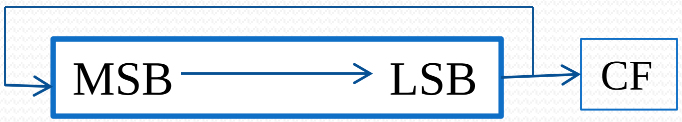
ROTATE INSTRUCTIONS

Rotate Instructions

- In many applications there is a need to perform a bitwise rotation of an operand.
- The rotation instructions ROR, ROL and RCR, RCL are designed specifically for that purpose.
- In rotate instructions, the operand can be in a register or memory.
- If the number of times an operand is to be rotated is more than 1, this is indicated by CL. This is similar to the shift instructions.
- There are two types of rotations: One is simple rotation of the bits of an operand and the other is a rotation through the carry.

Rotate Instructions

ROR: Rotate right



- In the rotate right, as bits are shifted from left to right they exit from the right end (LSB) and enter the left end (MSB).
- In addition, as each bit exits the LSB, a copy of it is given to the carry flag.
- In other words, in ROR the LSB is moved to the MSB and is also copied to CF.

Rotate Instructions

ROR: Rotate right

Example: Show the effects of ROR in the following:

MOV AL, 36H

MOV CL, 3

ROR AL, CL

Solution:

36H = 0 0 1 1 0 1 1 0

0 0 0 1 1 0 1 1 (CF = 0)

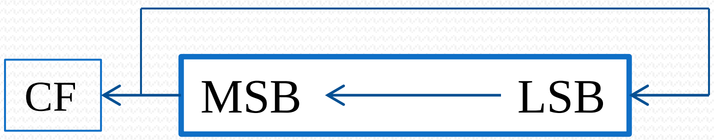
1 0 0 0 1 1 0 1 (CF = 1)

1 1 0 0 0 1 1 0 (CF = 1)

After rotating right three times, AL = C6H and CF = 1.

Rotate Instructions

ROL: Rotate left



- In the rotate left, as bits are shifted from right to left they exit the left end (MSB) and enter the right end (LSB).
- In addition, every bit that leaves the MSB is copied to the carry flag.
- In other words, in ROL the MSB is moved to the LSB and is also copied to CF, as shown in diagram.

Rotate Instructions

ROL: Rotate left

Example: Show the effects of ROL in the following:

```
MOV BH, 72H
MOV CL, 4
ROL BH, CL
```

Solution:

72H = 0 1 1 1 0 0 1 0

1 1 1 0 0 1 0 0 (CF = 0)

1 1 0 0 1 0 0 1 (CF = 1)

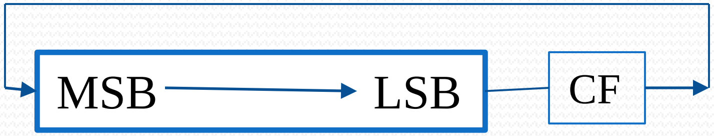
1 0 0 1 0 0 1 1 (CF = 1)

0 0 1 0 0 1 1 1 (CF = 1)

After rotating right four times, BH = 27H and CF = 1.

Rotate Instructions

RCR: Rotate right through carry



- In RCR, the bits are shifted from left to right, they exit the right end (LSB) to the carry flag, and the carry flag enters the left end (MSB).
- In other words, in RCR the LSB is moved to CF and CF is moved to the MSB.
- In reality, CF acts as if it is part of the operand.

Rotate Instructions

RCR: Rotate right through carry

Example: Show the effects of RCR in the following:

```
CLC        ; make CF = 0
MOV AL, 26H
MOV CL, 3
RCR AL, CL
```

Solution:

26H = 0 0 1 0 0 1 1 0

0 0 0 1 0 0 1 1 (CF = 0)

0 0 0 0 1 0 0 1 (CF = 1)

1 0 0 0 0 1 0 0 (CF = 1)

Result:

AL = 84H and CF = 1

Rotate Instructions

RCR: Rotate right through carry

Example: Show the effects of RCR in the following:

```
STC        ; make CF = 1
MOV BX, 37F1H
MOV CL, 5
RCR BX, CL
```

Solution:

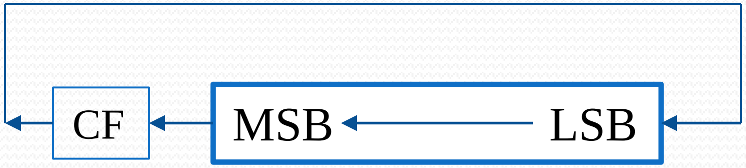
37F1H =	0011 0111 1111 0001	(CF = 1)
	1001 1011 1111 1000	(CF = 1)
	1100 1101 1111 1100	(CF = 0)
	0110 0110 1111 1110	(CF = 0)
	0011 0011 0111 1111	(CF = 0)
	0001 1001 1011 1111	(CF = 1)

Result:

BX = 19CFH and CF = 1

Rotate Instructions

RCL: Rotate left through carry



- In RCL, the bits are shifted from right to left, they exit the left end (MSB) and enter the carry flag, and the carry flag enters the right end (LSB).
- In other words, in RCL the MSB is moved to CF and CF is moved to the LSB.
- In reality, CF acts as if it is part of the operand.

Rotate Instructions

RCL: Rotate left through carry

Example: Show the effects of RCR in the following:

```
STC     ; make CF = 1
MOV BL, 15H
MOV CL, 2
RCL BL, CL
```

Solution:

15H =	(CF = 1)	0 0 0 1 0 1 0 1
	(CF = 0)	0 0 1 0 1 0 1 1
	(CF = 0)	0 1 0 1 0 1 1 0

Result:

BL = 56H and CF = 0