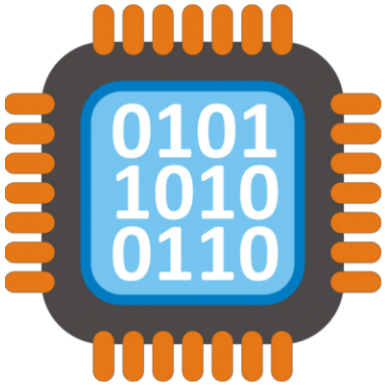
Secure Assembly Coding

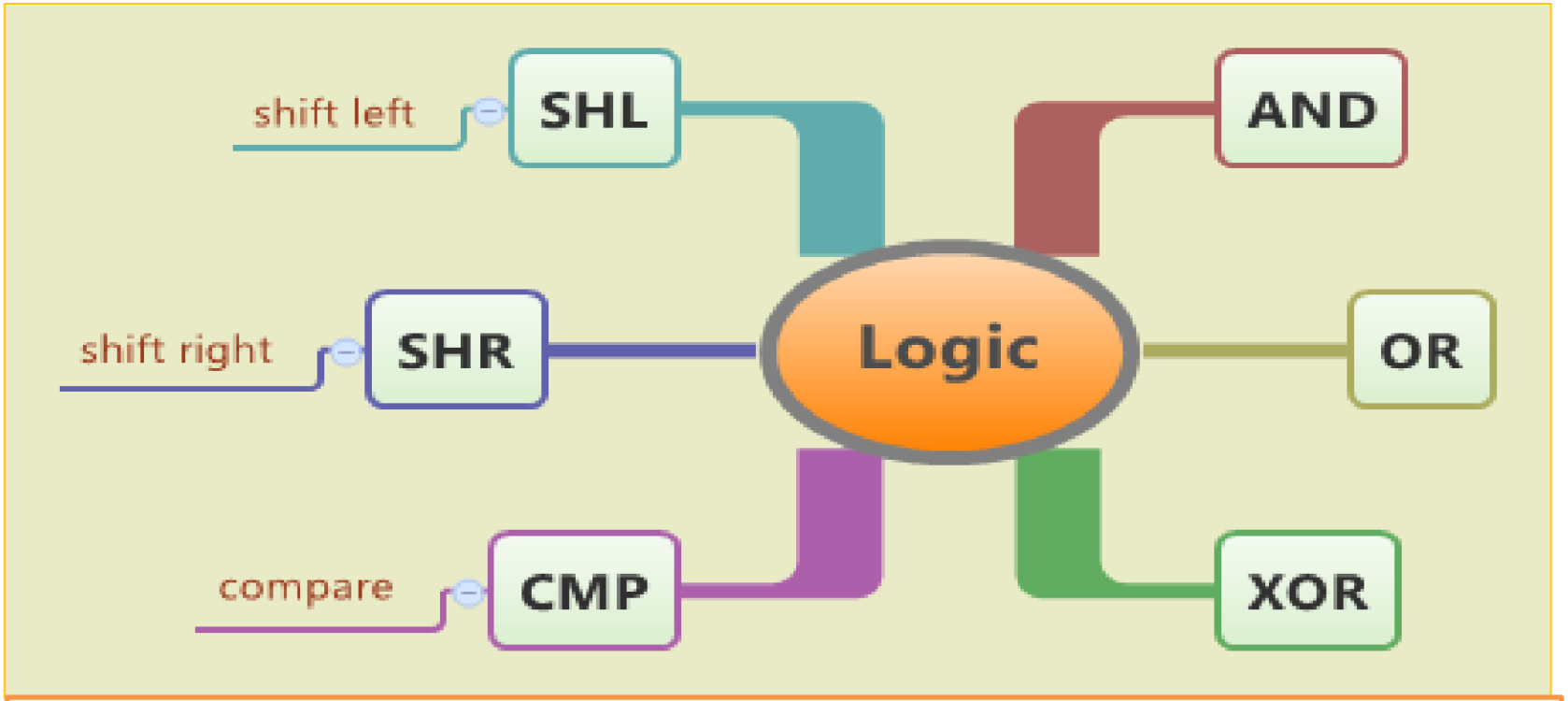
Week # 8 Lectures



Dr. Qasem Abu Al-Haija,
Department of Cybersecurity,

8086 Instruction Set

Group 3 : Bit Manipulation Instructions (logical).



Arithmetic Instructions:

AND, OR, XOR, NOT, TEST, SHL, SHR, SAL, SAR, ROL, ROR, RCL, RCR

8086 Instruction Set

Group 3 : Bit Manipulation Instructions (logical).

Logicals	
NOT mem/reg	NOT byte or word → *One's complement*
AND a, b	AND byte or word
OR a, b	OR byte or word
XOR a, b	Exclusive OR byte or word
TEST a, b	Test byte or word

Shifts	
SHL/SAL mem/reg, CNT	Shift logical/arithmetic left byte or word
SHR/SAR mem/reg, CNT	Shift logical/arithmetic right byte or word

Rotates	
ROL mem/reg, CNT	Rotate left byte or word
ROR mem/reg, CNT	Rotate right byte or word

a = "reg" or "mem," b = "reg" or "mem" or "data," CNT = number of times to be shifted.
If CNT > 1, then CNT is contained in CL. Zero or negative shifts and rotates are illegal.
If CNT = 1 then CNT is immediate data. Up to 255 shifts are allowed.

8086 Instruction Set

Group 3 : Bit Manipulation Instructions (logical).

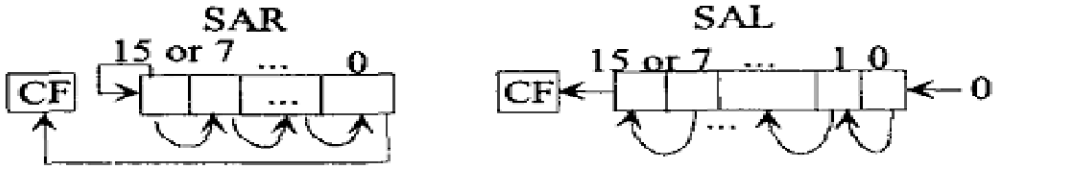
EX: TEST CL, 05H

Logically ANDs (CL) with 00000101. Does not store the result in CL, All flags are affected.

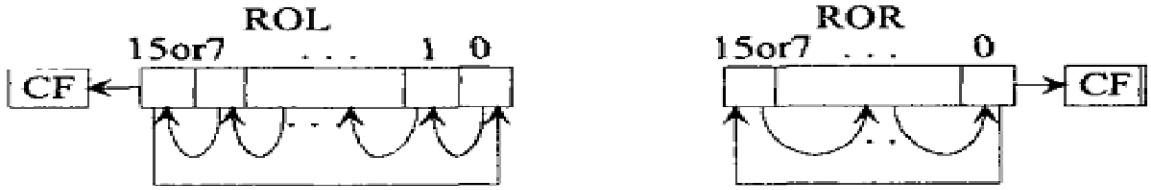
EX: Let AL=0111 1111 =7FH,

TEST AL, 80H; AL=7FH (unchanged), ZF=1 since (AL) AND (80H)=00H; SF=0; PF=1

EX: MOV CL,2 ; Shift count 2 is moved into CL
SHR DX,CL; Logically shifts (DX) twice to right



8086 SAR and SAL instructions



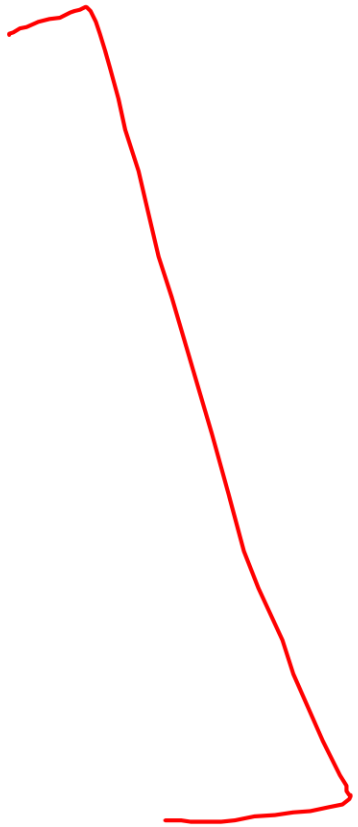
8086 ROR and ROL instructions

Bitwise instructions

- **Logical Instructions**
- **AND : Turns off specific bits (used with masking) .**
- **TEST : Same as AND , buts does not change destination .**
- **OR : Turns on specific bits .**
- **NOT : Complement all the bits .**
- **XOR : Complement specific bits .**

Shift and Rotate Instructions

- SHR : Shift Right
- SAR : Shift Arithmetic Right
- SHL : Shift Left
- SAL : Shift Arithmetic Left
- ROL : Rotate Left
- ROR : Rotate Right
- RCL : Rotate Carry Left
- RCR : Rotate Carry Right



Logical Instructions

AND Truth Table

Inputs		Output
A	B	Y = A.B
0	0	0
0	1	0
1	0	0
1	1	1

OR Truth Table

Inputs		Output
A	B	Y = A+B
0	0	0
0	1	1
1	0	1
1	1	1

XOR Truth Table

Inputs		Output
A	B	$Y = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

NAND Truth Table

Inputs		Output
A	B	$Y = \overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0

www.geekyshows.com

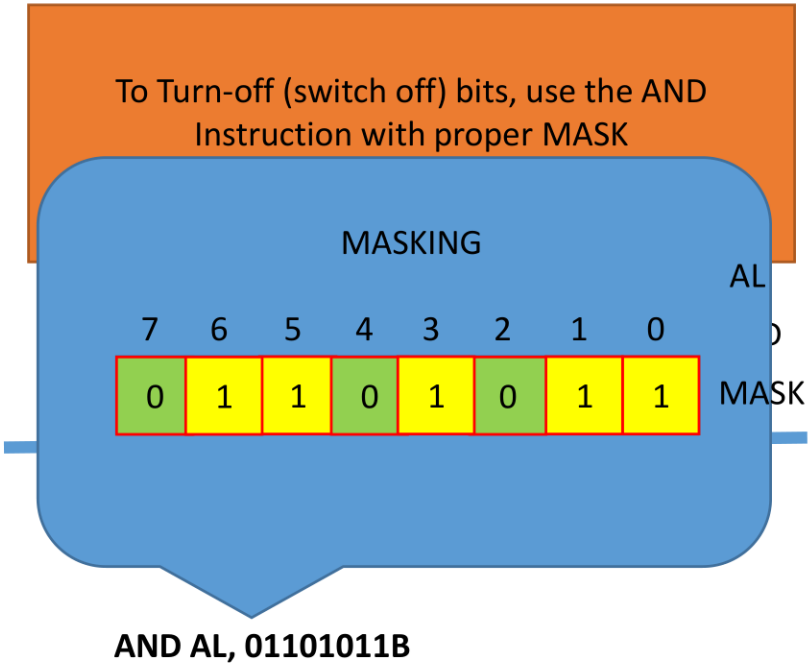
Logical Instructions

Given the Al Register as follows:

7	6	5	4	3	2	1	0
1	1	0	1	0	0	1	1

Perform the following Tasks on the AL Register:

1) **Turn off bits numbers 2, 4 , 7**



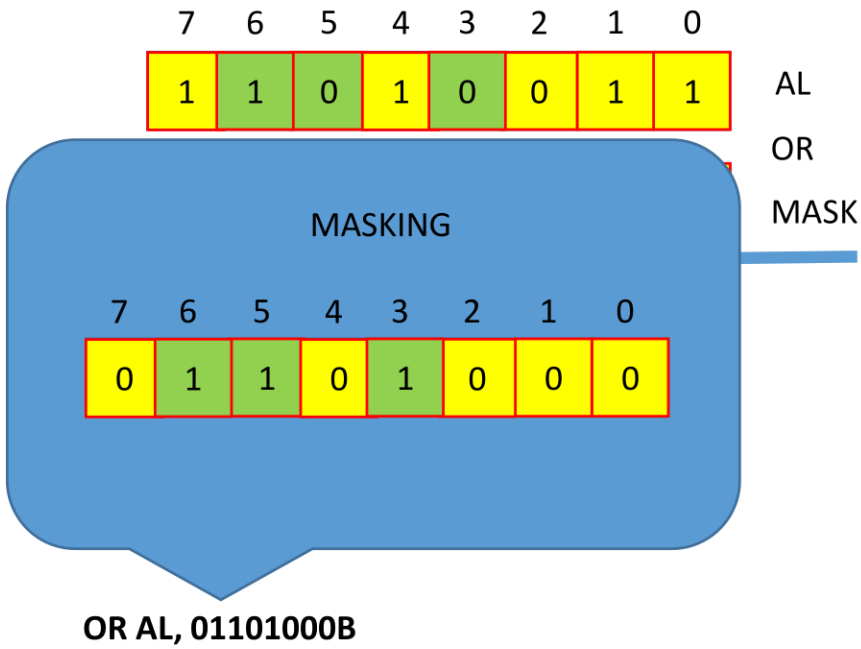
To Turn-on (switch on) bits, use the OR Instruction with proper MASK
Use 1 under the bit to turn on in the mask

Given the Al Register as follows:

7	6	5	4	3	2	1	0
1	1	0	1	0	0	1	1

Perform the following Tasks on the AL Register:

1) **Turn on bits numbers 3 , 5 , 6**



Given the Al Register as follows:

7	6	5	4	3	2	1	0
1	1	0	1	0	0	1	1

XOR Truth Table

Inputs		Output
A	B	$Y = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

3. Switch (Complement) bits

To Complement the value of a bit, use th
Instruction with proper MASK
Use 1 under the bit to complement in the

MASKING

7	6	5	4	3	2	1	0
1	0	0	1	1	0	0	1

XOR

XOR AL, 10011001B

0	1	0	0	1	0	1	0

Switch is also called toggle

EX) Given the AL Register as a status register , check bit number 3 . If bit number 3 equal zero then turn on the first 4 bits and switch off the remaining bits. Else complement the last 4 bits .

Given the Al Register as follows:

7	6	5	4	3	2	1	0
1	1	0	1	0	0	1	1

1) TEST Bit #3
   Test AL, 00001000B Mask
   JNZ ON ; if bit#3 = 0 then don't jump

0	0	0	0	0	0	0	0

2) NOTE: TEST does not change the value of AL
   Cont. next page

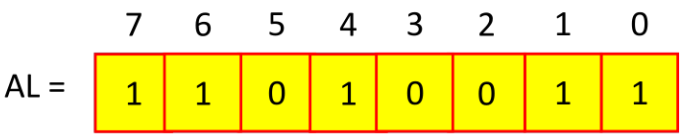
To test bit #3 use the TEST instruction with the following MASK:

0	0	0	0	1	0	0	0

The TEST instruction is the same as AND, except it does not change the destination Register

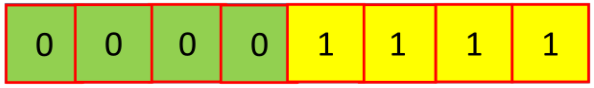
The result of the test for the entire register is ZERO, meaning that bit#3 was off….Now you can use JZ or JNZ to print proper message

Cont.



2) Turn on first 4 bits (0,1,2,3)
OR AL, 00001111B

3) Turn off last 4 bits (4,5,6,7)
AND AL, 00001111B



Al contains the same original value after the test… It is unchanged

The results in AL is that the first 4 bits are on

Now, to switch off the last 4 bits (4,5,6,7)
We use the AND instruction with the mask

Cont.

4) Else IF bit 3 is 1 then Complement the last 4 bits

	7	6	5	4	3	2	1	0
AL=	1	1	0	1	0	0	1	1

```
ON:     XOR     AL,11110000B
```

0	0	1	0	0	0	1	1

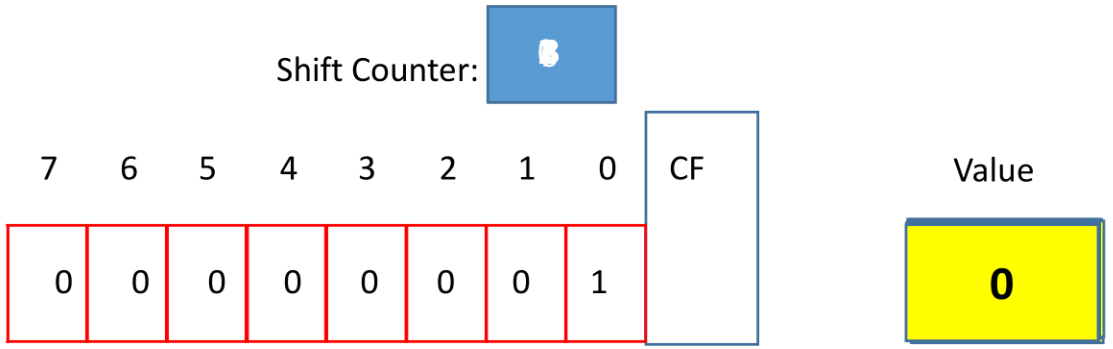
```
EXIT:   MOV     AX,4C00H
        INT     21H
```

To complement the last 4 bits (4,5,6,7)
We use the XOR instruction with the mask

1	1	1	1	0	0	0	0

Shift Right

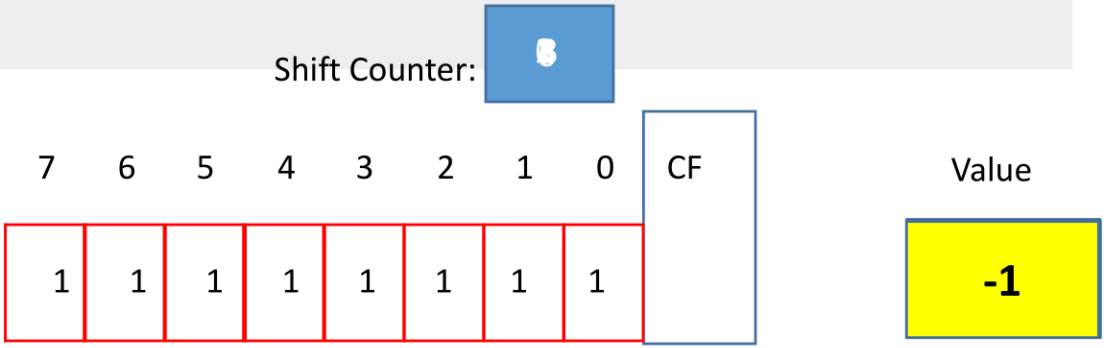
```
Syntax: SHR Register, Bits to be shifted

Example:  Mov AL, 173
          SHR AL, 8        ;Note every shift to the right is a division by 2
```



SAR : Shift Arithmetic Right

The SAR instruction stands for 'Shift Arithmetic Right'. This instruction shifts the mentioned bits in the register to the right side one by one, but instead of inserting the zeroes from the left end, the MSB is restored. The rightmost bit that is being shifted is stored in the Carry Flag (CF). Works with Signed Numbers.

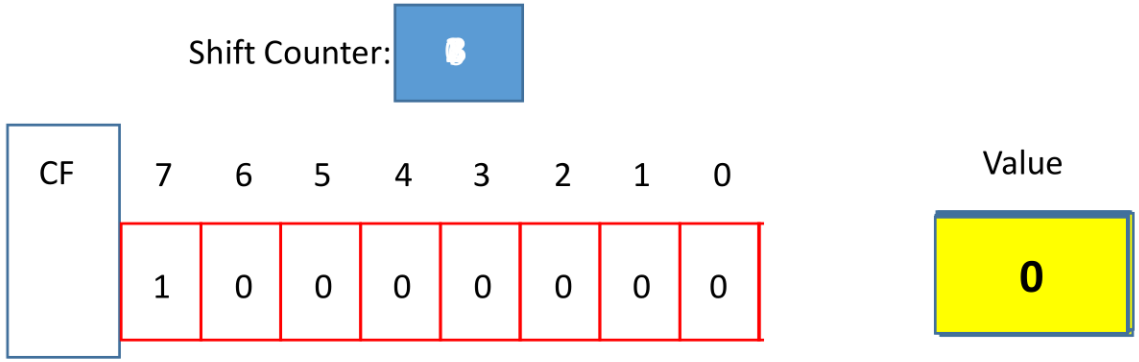
```
Syntax: SAR Register, Bits to be shifted

Example: Mov Al, -80
         SAR AL, 8
```



SHL : Shift Left

The SHL instruction is an abbreviation for 'Shift Left'. This instruction simply shifts the mentioned bits in the register to the left side one by one by inserting the same number (bits that are being shifted) of zeroes from the right end. The leftmost bit that is being shifted is stored in the Carry Flag (CF).

```
Syntax: SHL Register, Bits to be shifted
        MOV AL, 5
Example: SHL AL, 8   ; Note every shift to the left is similar to multiply by 2
```



SAL : Shift Arithmetic Left

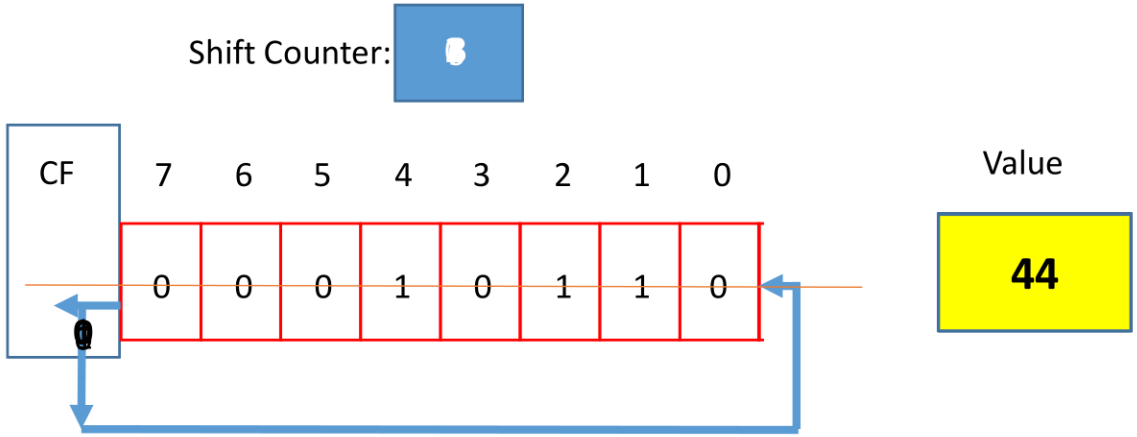
The SAL instruction is an abbreviation for ‘Shift Arithmetic Left’. This instruction is the same as SHL.

```
Syntax: SAL Register, Bits to be shifted

Example: SAL CL, 3
```

ROL : Rotate Left

The ROL instruction is an abbreviation for 'Rotate Left'. This instruction rotates the mentioned bits in the register to the left side one by one such that leftmost bit that is being rotated is again stored as the rightmost bit in the register, and it is also stored in the Carry Flag (CF).

```
Syntax: ROL Register, Bits to be shifted
Example: ROL AH, 8
```



Value

44

ROR : Rotate Right

The ROR instruction stands for ‘Rotate Right’. This instruction rotates the mentioned bits in the register to the right side one by one such that rightmost bit that is being rotated is again stored as the MSB in the register, and it is also stored in the Carry Flag (CF).

```
Syntax: ROR Register, Bits to be shifted

Example: ROR AH, 4
```

RCL : Rotate Carry Left

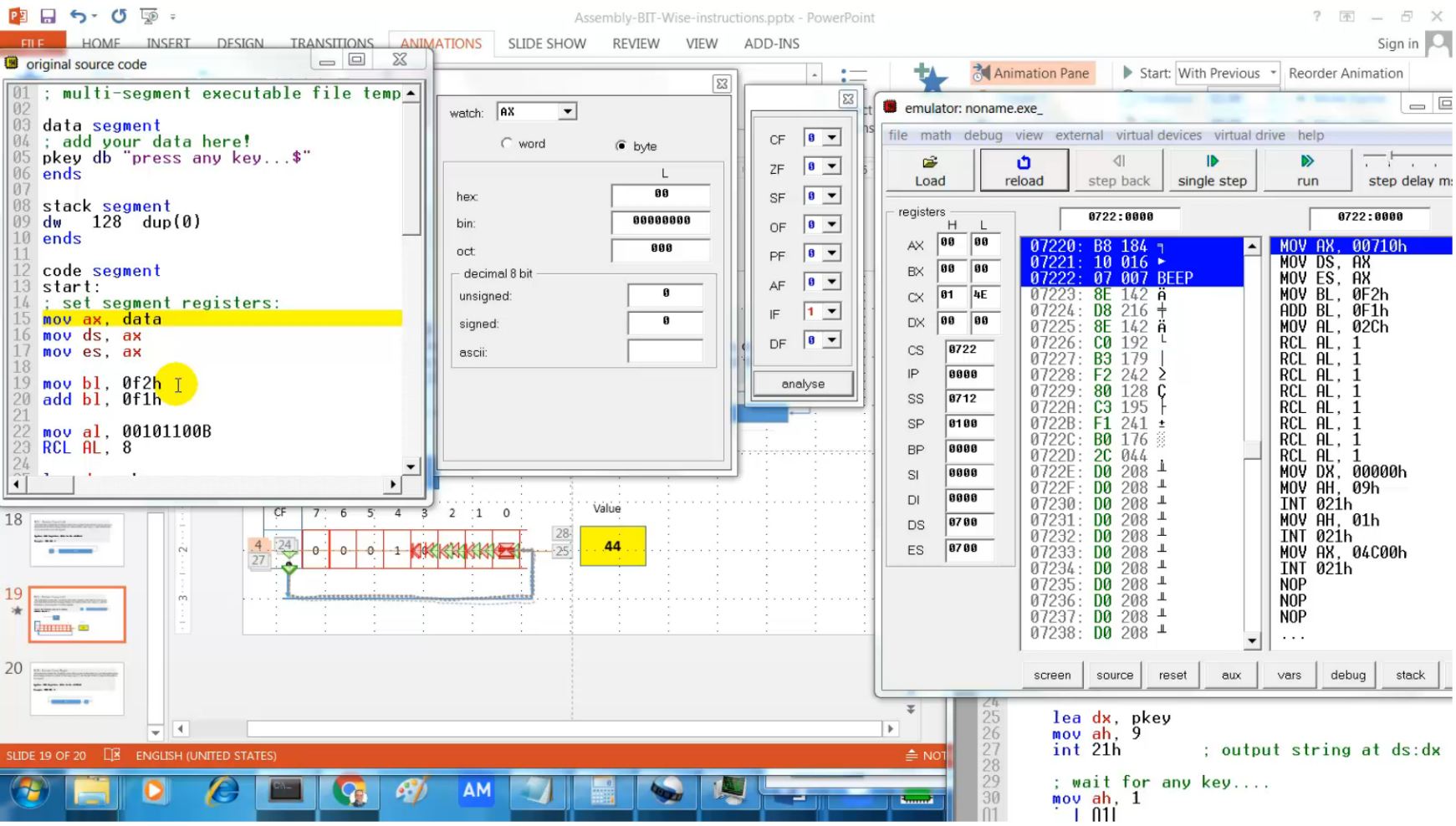
This instruction rotates the mentioned bits in the register to the left side one by one such that leftmost bit that is being rotated it is stored in the Carry Flag (CF), and the bit in the CF moved as the LSB in the register.

```
Syntax: RCL Register, Bits to be shifted

Example: RCL CH, 1
```

RCR : Rotate Carry Right

This instruction rotates the mentioned bits in the register to the right side such that rightmost bit that is being rotated it is stored in the Carry Flag (CF), and the bit in the CF moved as the MSB in the register.

```
Syntax: RCR Register, Bits to be shifted

Example: RCR BH, 6
```


Assembly-BIT-Wise-instructions.pptx - PowerPoint
original source code
; multi-segment executable file temp
data segment
; add your data here!
pkey db "press any key...$"
ends
stack segment
dw 128 dup(0)
ends
code segment
start:
; set segment registers:
mov ax, data
mov ds, ax
mov es, ax
mov bl, 0f2h
add bl, 0f1h
mov al, 00101100B
RCL AL, 8
watch:
word
byte
hex:
bin:
oct:
decimal 8 bit
unsigned:
signed:
ascii:
CF
ZF
SF
OF
PF
AF
IF
DF
analyse
Value
44
emulator: noname.exe_
file math debug view external virtual devices virtual drive help
Load
reload
step back
single step
run
step delay
registers
AX
BX
CX
DX
CS
IP
SS
SP
BP
SI
DI
DS
ES
07220: B8 184
07221: 10 016
07222: 07 007 BEEP
MOV AX, 00710h
MOV DS, AX
MOV ES, AX
MOV BL, 0F2h
ADD BL, 0F1h
MOV AL, 02Ch
RCL AL, 1
MOV DX, 00000h
MOV AH, 09h
INT 021h
MOV AH, 01h
INT 021h
MOV AX, 04C00h
INT 021h
NOP
screen
source
reset
aux
vars
debug
stack
lea dx, pkey
mov ah, 9
int 21h ; output string at ds:dx
; wait for any key....
mov ah, 1
SLIDE 19 OF 20
ENGLISH (UNITED STATES)

See other examples in the separate ppt file uploaded into Moodle

More Example_2

Main Sources for these slides

- *K. R. Irvine. Assembly Language for x86 Processors, 8th edition, Prentice-Hall (Pearson Education), June 2019. ISBN: 978-0135381656.*

- *B. Dang, A. Gazet, E. Bachaalany. Practical Reverse Engineering: x86, x64, ARM, Windows® Kernel, Reversing Tools, and Obfuscation. John Wiley & Sons, June 2014. ISBN: 978-1-118-78731-1*

- *Qasem Abu Al-Haija, "Microprocessor Systems", King Faisal University, Saudi Arabia*

- *Ghassan Issa, "Computer Organization", Petra University, Jordan.*

Thank you