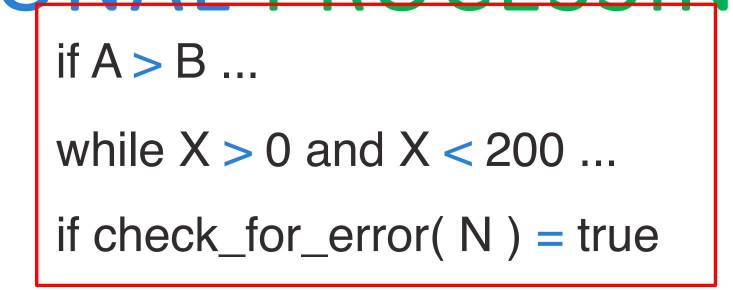
CSC 3210
Computer Organization and Programming

CHAPTER 6: CONDITIONAL PROCESSING



```
if A > B ...

while X > 0 and X < 200 ...

if check_for_error( N ) = true
```

Outline

- **Boolean and Comparison Instructions**
- Conditional Jumps
- Conditional Structures
- Conditional Control Flow Directives

Boolean and Comparison Instructions

- CPU Status Flags
- AND Instruction
- OR Instruction
- XOR Instruction
- NOT Instruction
  - Applications
- TEST Instruction
- CMP Instruction

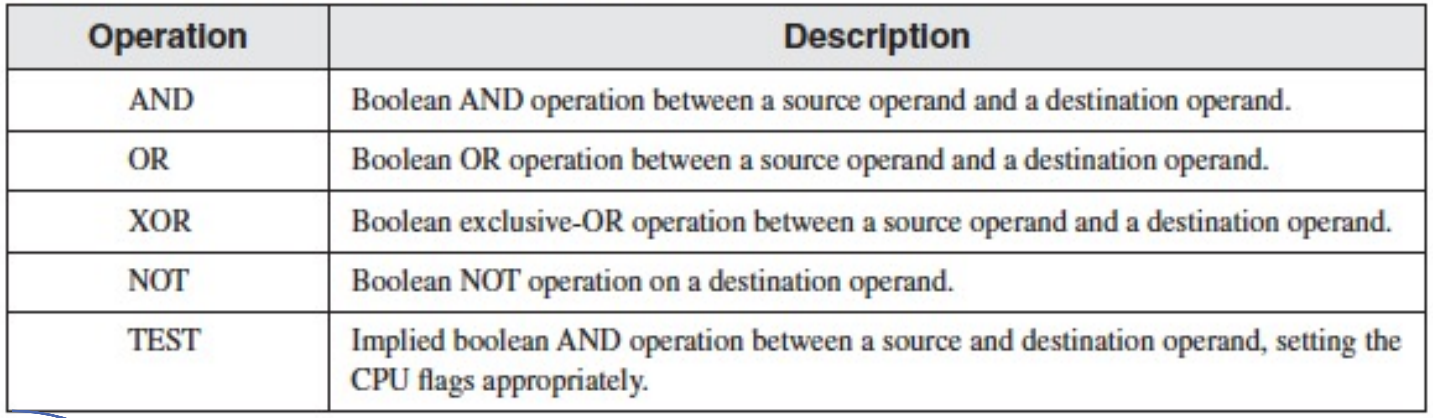
```
if A > B ...
while X > 0 and X < 200 ...
if check_for_error( N ) = true
```

Boolean operations are the core of all **decision statements** because they **affect** the **CPU status flags**.

Status Flags - Review

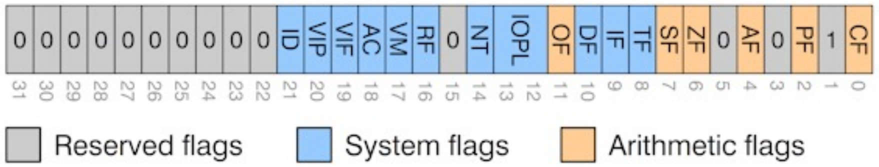
- **Boolean instructions** **affect** the Zero, Carry, Sign, Overflow, and Parity flags.

Table 6-1 Selected Boolean Instructions.



Operation	Description
AND	Boolean AND operation between a source operand and a destination operand.
OR	Boolean OR operation between a source operand and a destination operand.
XOR	Boolean exclusive-OR operation between a source operand and a destination operand.
NOT	Boolean NOT operation on a destination operand.
TEST	Implied boolean AND operation between a source and destination operand, setting the CPU flags appropriately.

eflags register



0	0	0	0	0	0	0	0	0	0	ID	VIP	VIF	AC	VM	RF	0	NT	IOPL		OF	DF	IF	TF	SF	ZF	0	AF	0	PF	1	CF
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

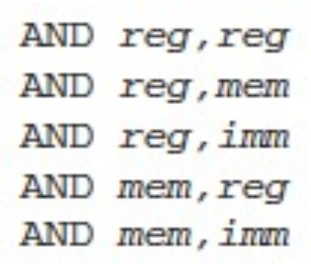
Status Flags - Review

- The Zero flag is set when the result of an operation equals zero.
- The Carry flag is set when an instruction generates a result that is too large (or too small) for the destination operand.
- The Sign flag is set if the destination operand is negative, and it is clear if the destination operand is positive.
- The Overflow flag is set when an instruction generates an invalid signed result.
- The Parity flag is set when an instruction generates an **even number** of **1** bits in the **low byte** of the destination operand.

AND Instruction

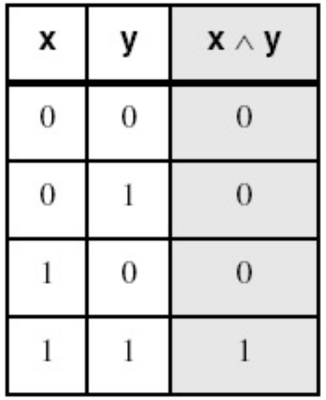
- Performs a Boolean AND operation between **each pair of matching bits** in two operands
- Syntax:

AND ***destination***, ***source*** (same operand types as MOV)



```
AND reg,reg
AND reg,mem
AND reg,imm
AND mem,reg
AND mem,imm
```

AND



x	y	x ∧ y
0	0	0
0	1	0
1	0	0
1	1	1

AND AL, BL

AND Instruction

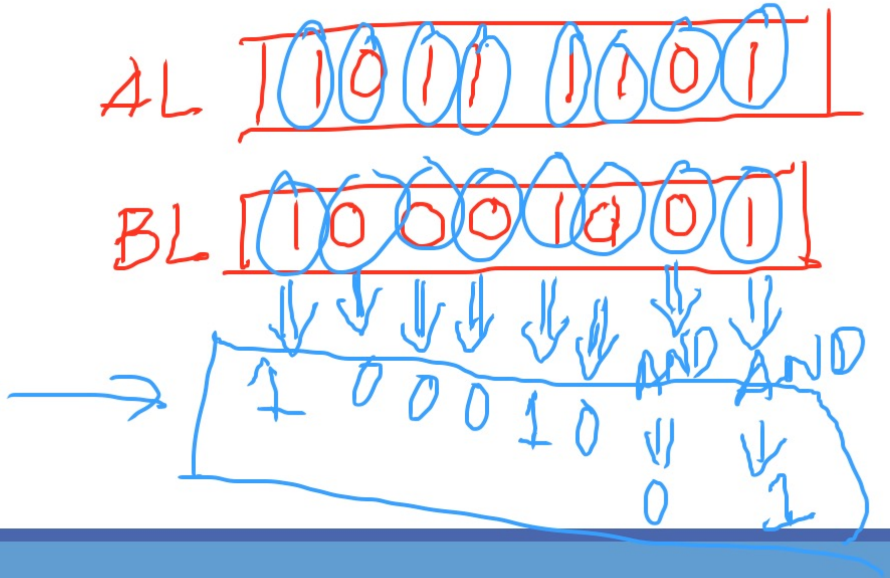
- Performs a Boolean AND operation between **each pair of matching bits** in two operands
- Syntax:

AND ***destination***, ***source*** (same operand types as MOV)

```
AND reg,reg
AND reg,mem
AND reg,imm
AND mem,reg
AND mem,imm
```

AND

x	y	x ∧ y
0	0	0
0	1	0
1	0	0
1	1	1



AND Instruction

0 0

- **Flags**
  - The AND instruction always clears the Overflow and Carry flags.
  - It modifies the Sign, Zero, and Parity flags
    - Consistent with the value assigned to the **destination operand**.

- **Example**:
  - Suppose the following instruction **results in** a value of Zero in the al register.
  - Thus, the Zero flag will be **set**:

```
and al,1Fh
```

AND Example

- **Task**: Jump to a label if an integer is even.
- **Solution**: AND the **lowest bit** with a 1,

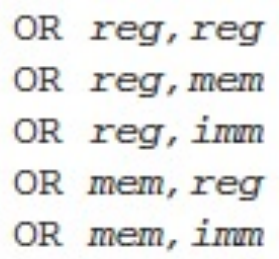
  If the result is Zero, the number was even.

```
mov ax,wordVal
And ax,1                ; low bit set?
jz  EvenValue           ; jump if Zero flag set
```

JZ (jump if Zero) is covered in Section 6.3.

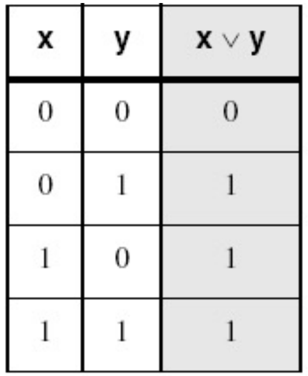
OR Instruction

- Performs a **Boolean OR** operation between each pair of matching bits in two operands
- **Syntax**: `OR destination, source`



```
OR  reg,reg
OR  reg,mem
OR  reg,imm
OR  mem,reg
OR  mem,imm
```

OR



x	y	x ∨ y
0	0	0
0	1	1
1	0	1
1	1	1

AL 1101 1000

AL 1101 1000

11011000

OR Instruction

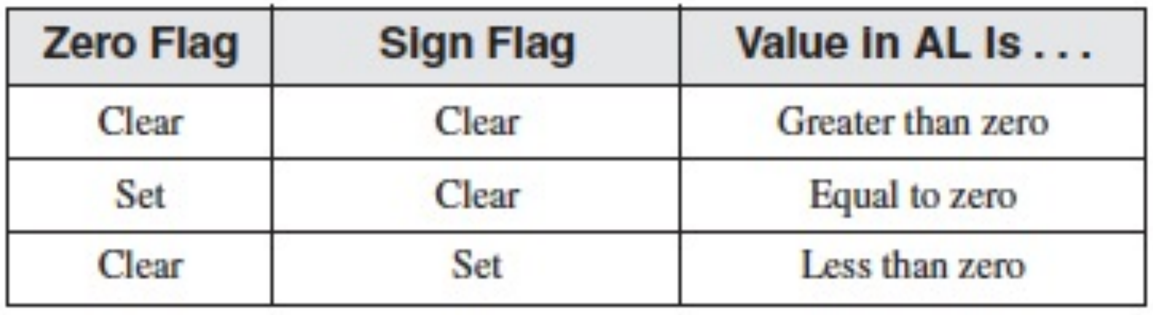
- **Flags**
  - The OR instruction always clears the Carry and Overflow flags.
  - It modifies the Sign, Zero, and Parity flags
    - consistent with the value assigned to the destination operand.

- **Example**: OR a number with itself (or zero) to **obtain certain information about its value**:

```
or al,al
```

OR Instruction

- The values of the Zero and Sign flags indicate the following about the contents of AL:



Zero Flag	Sign Flag	Value in AL is . . .
Clear	Clear	Greater than zero
Set	Clear	Equal to zero
Clear	Set	Less than zero

OR Example

- Task: Jump to a label if the value in AL is not zero.
- Solution: OR the byte with itself, then use the JNZ (jump if not zero) instruction.

```
or  al,al
jnz IsNotZero             ; jump if not zero
```

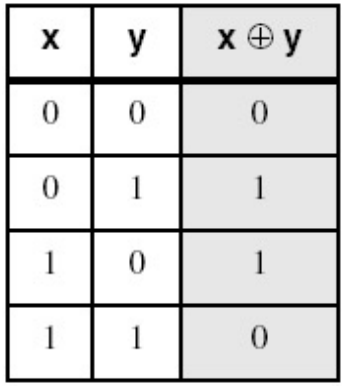
ORing any number with itself **does not change its value**.

XOR Instruction

- Performs a Boolean exclusive-OR operation between each pair of matching bits in two operands

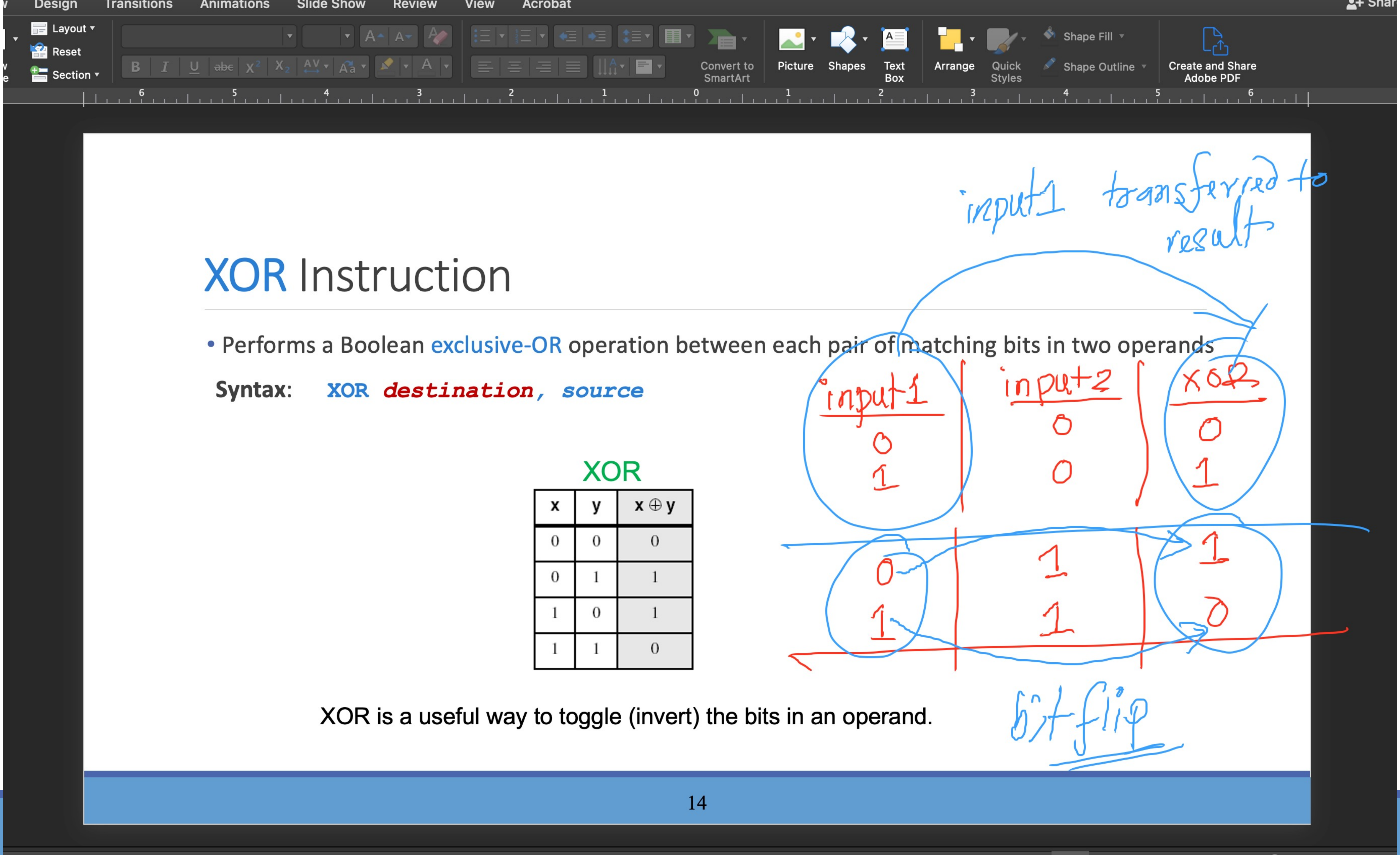
Syntax: `XOR destination, source`

XOR



x	y	x ⊕ y
0	0	0
0	1	1
1	0	1
1	1	0

XOR is a useful way to toggle (invert) the bits in an operand.



XOR Instruction

- Performs a Boolean exclusive-OR operation between each pair of matching bits in two operands

Syntax: `XOR destination, source`

XOR

x	y	x ⊕ y
0	0	0
0	1	1
1	0	1
1	1	0

XOR is a useful way to toggle (invert) the bits in an operand.

input1 transferred to result

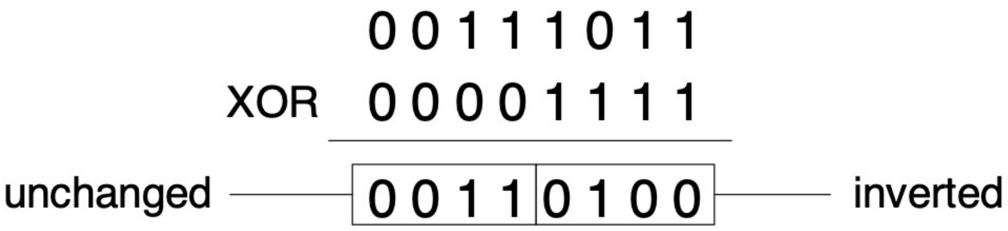
input1	input2	xor
0	0	0
1	0	1
0	1	1
1	1	0

bit flip

14

XOR Instruction

- **Example:**
  - **bit masking:**
    - A bit exclusive-ORed with 0 **retains its value**,
    - A bit exclusive-ORed with 1 is **toggle** (complemented).



0 0 1 1 1 0 1 1

XOR 0 0 0 0 1 1 1 1

unchanged 0 0 1 1 0 1 0 0 inverted

XOR is a useful way to toggle (invert) the bits in an operand.

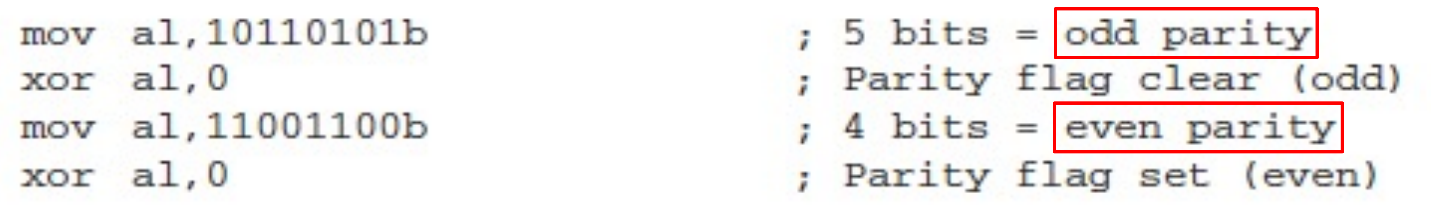
XOR

x	y	x ⊕ y
0	0	0
0	1	1
1	0	1
1	1	0

XOR Instruction

- **Flags**
  - The XOR instruction always clears the Overflow and Carry flags.
  - XOR modifies the Sign, Zero and Parity flags
    - consistent with the value assigned to the destination operand.

- **Example:**
  - **An effective way to check the parity of a number** without changing its value is to exclusive-OR the number with zero:

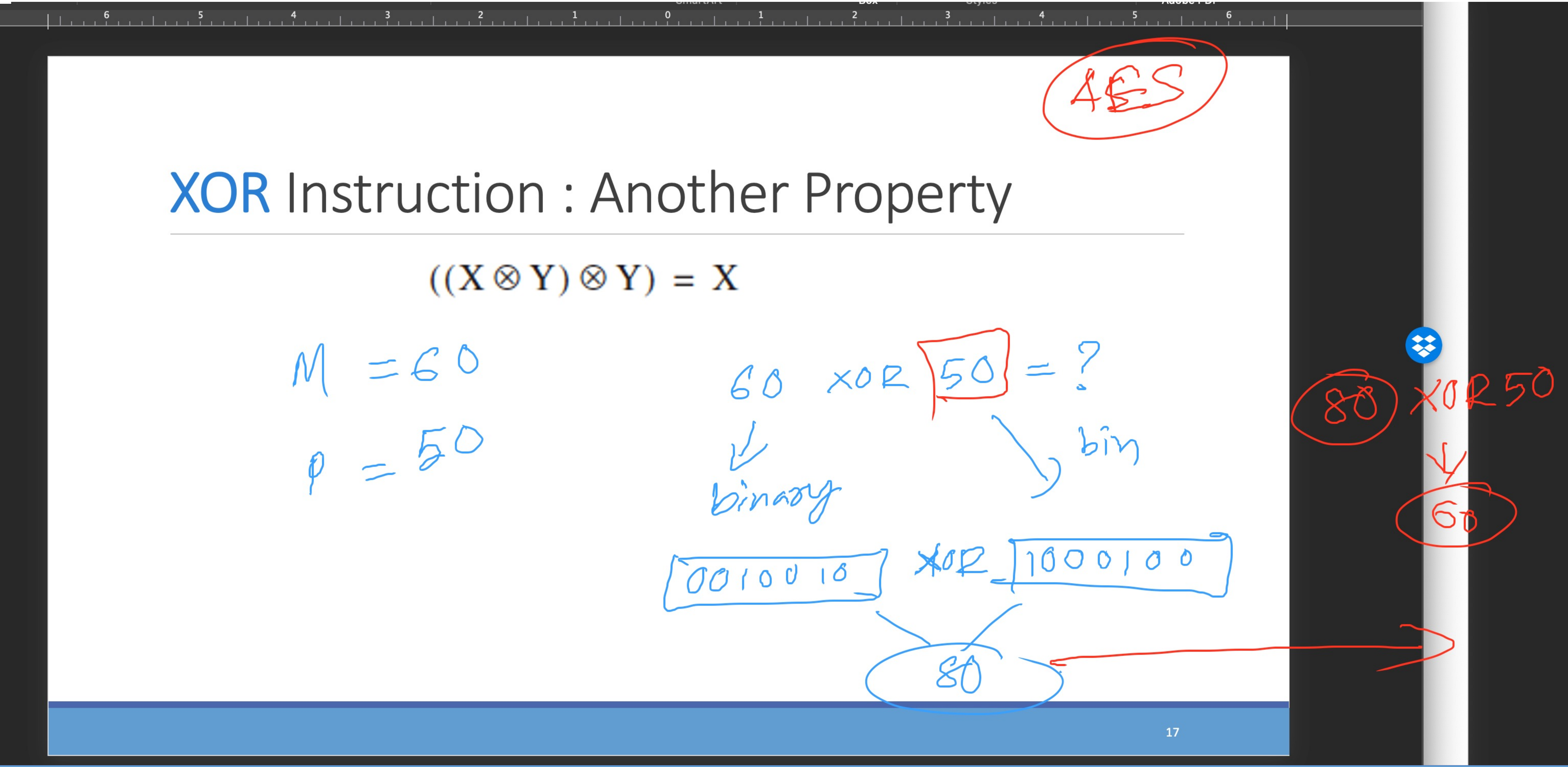


```
mov  al,10110101b               ; 5 bits = odd parity
xor  al,0                       ; Parity flag clear (odd)
mov  al,11001100b               ; 4 bits = even parity
xor  al,0                       ; Parity flag set (even)
```

The Parity flag (PF) is set when The least significant byte of the **destination** has an **even number of 1 bits**.

XOR Instruction : Another Property

$$((X \otimes Y) \otimes Y) = X$$



AES

XOR Instruction : Another Property

$$((X \otimes Y) \otimes Y) = X$$

M = 60

φ = 50

60 XOR $\boxed{50}$ = ?

binary

bin

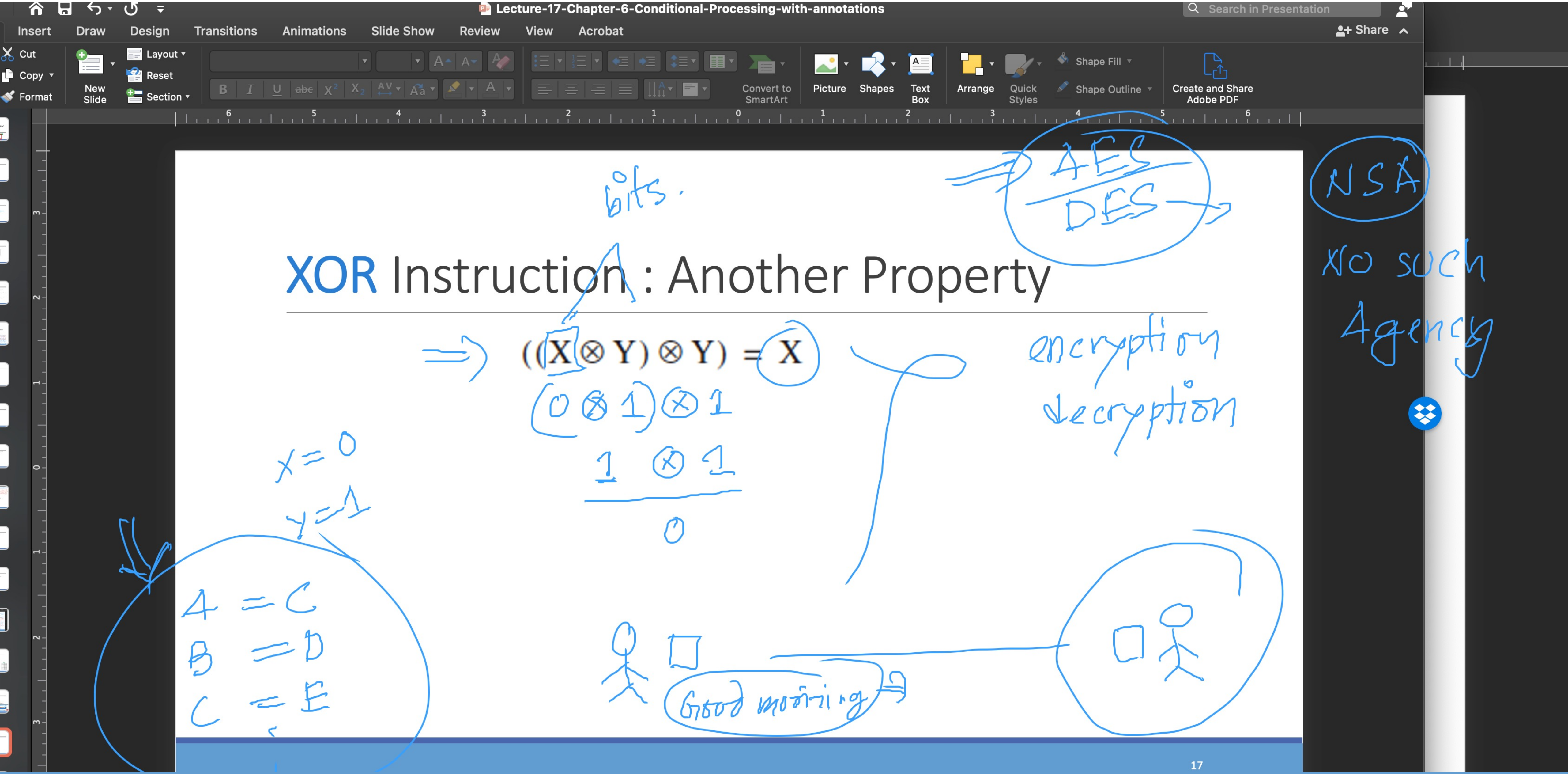
$\boxed{0010010}$ XOR $\boxed{1000100}$

80

80 XOR 50

60

17



bits.

AES
DES

NSA

No such Agency

XOR Instruction : Another Property

$$((X \otimes Y) \otimes Y) = X$$

$(0 \otimes 1) \otimes 1$

$1 \otimes 1$

0

encryption

decryption

$x = 0$

$y = 1$

A = C

B = D

C = E

Good morning

17

XOR Application: Encrypting a String

- The following loop uses the XOR instruction to transform every character in a string into a new value.

$$((X \otimes Y) \otimes Y) = X$$

```
KEY = 239                               ; can be any byte value between 1-255
BUFMAX = 128
.data
buffer  BYTE BUFMAX+1 DUP(0)
bufSize DWORD BUFMAX

.code
    mov ecx,bufSize                     ; loop counter
    mov esi,0                           ; index 0 in buffer
L1:
    xor buffer[esi],KEY                 ; translate a byte
    inc esi                             ; point to next byte
    loop L1
```

```
Enter the plain text: Attack at dawn.

Cipher text: «¢¢Äîä-Ä¢-ïÄÿü-Gs

Decrypted: Attack at dawn.
```