

# Conditional Processing

COE 205

Computer Organization and Assembly Language

Dr. Aiman El-Maleh

College of Computer Sciences and Engineering

King Fahd University of Petroleum and Minerals

[Adapted from slides of Dr. Kip Irvine: Assembly Language for Intel-Based Computers]

# Outline

- ❖ Boolean and Comparison Instructions
- ❖ Conditional Jumps
- ❖ Conditional Loop Instructions
- ❖ Translating Conditional Structures
- ❖ Indirect Jump and Table-Driven Selection
- ❖ Application: Sorting an Integer Array

# AND Instruction

- ❖ Bitwise AND between each pair of matching bits

**AND *destination, source***

- ❖ Following operand combinations are allowed

**AND *reg, reg***

**AND *reg, mem***

**AND *reg, imm***

**AND *mem, reg***

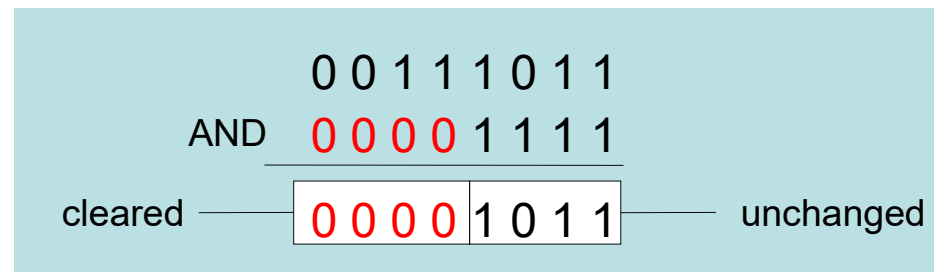
**AND *mem, imm***

Operands can be  
8, 16, or 32 bits  
and they must be  
of the same size

**AND**

x	y	$x \wedge y$
0	0	0
0	1	0
1	0	0
1	1	1

- ❖ AND instruction is  
often used to  
**clear selected bits**



# Converting Characters to Uppercase

- ❖ AND instruction can convert characters to uppercase

'a' = 0 1 **1** 0 0 0 0 1      'b' = 0 1 **1** 0 0 0 1 0

'A' = 0 1 **0** 0 0 0 0 1      'B' = 0 1 **0** 0 0 0 1 0

- ❖ Solution: Use the AND instruction to **clear bit 5**

```
    mov     ecx, LENGTHOF mystring
    mov     esi, OFFSET mystring
L1:  and     BYTE PTR [esi], 11011111b ; clear bit 5
      inc     esi
      loop   L1
```

# OR Instruction

❖ Bitwise OR operation between each pair of matching bits  
OR *destination, source*

❖ Following operand combinations are allowed **OR**

OR *reg, reg*

OR *reg, mem*

OR *reg, imm*

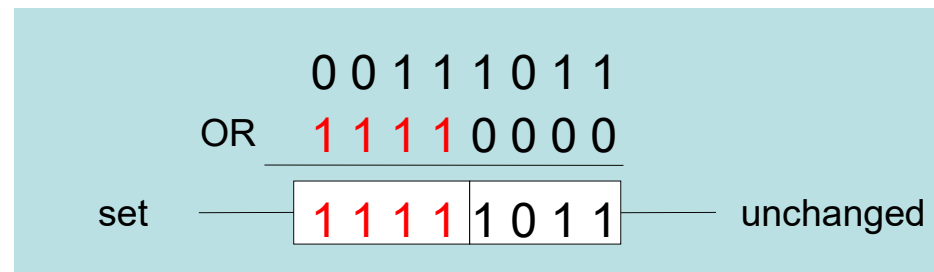
OR *mem, reg*

OR *mem, imm*

Operands can be  
8, 16, or 32 bits  
and they must be  
of the same size

x	y	$x \vee y$
0	0	0
0	1	1
1	0	1
1	1	1

❖ OR instruction is  
often used to  
**set selected bits**



# Converting Characters to Lowercase

- ❖ OR instruction can convert characters to lowercase

'A' = 0 1 **0** 0 0 0 0 1      'B' = 0 1 **0** 0 0 0 1 0

'a' = 0 1 **1** 0 0 0 0 1      'b' = 0 1 **1** 0 0 0 1 0

- ❖ Solution: Use the OR instruction to **set bit 5**

```
    mov     ecx, LENGTHOF mystring
    mov     esi, OFFSET mystring
L1:  or      BYTE PTR [esi], 20h          ; set bit 5
    inc     esi
    loop    L1
```

# Converting Binary Digits to ASCII

- ❖ OR instruction can convert a binary digit to ASCII

0 = 0 0 **0 0** 0 0 0 0      1 = 0 0 **0 0** 0 0 0 1

'0' = 0 0 **1 1** 0 0 0 0      '1' = 0 0 **1 1** 0 0 0 1

- ❖ Solution: Use the OR instruction to **set bits 4 and 5**

`or al,30h ; Convert binary digit 0 to 9 to ASCII`

- ❖ What if we want to convert an ASCII digit to binary?

- ❖ Solution: Use the AND instruction to **clear bits 4 to 7**

`and al,0Fh ; Convert ASCII '0' to '9' to binary`

# XOR Instruction

- ❖ Bitwise XOR between each pair of matching bits

**XOR *destination, source***

- ❖ Following operand combinations are allowed

**XOR *reg, reg***

**XOR *reg, mem***

**XOR *reg, imm***

**XOR *mem, reg***

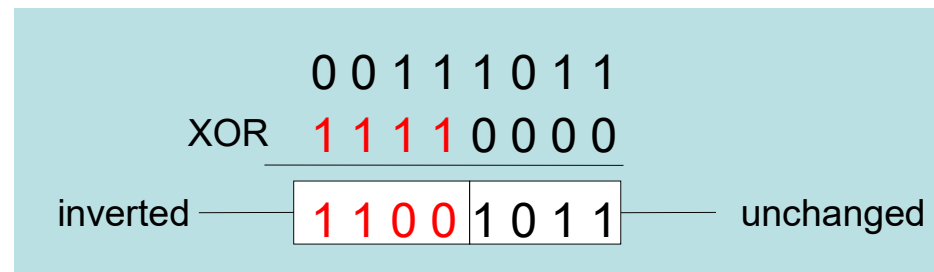
**XOR *mem, imm***

Operands can be  
8, 16, or 32 bits  
and they must be  
of the same size

**XOR**

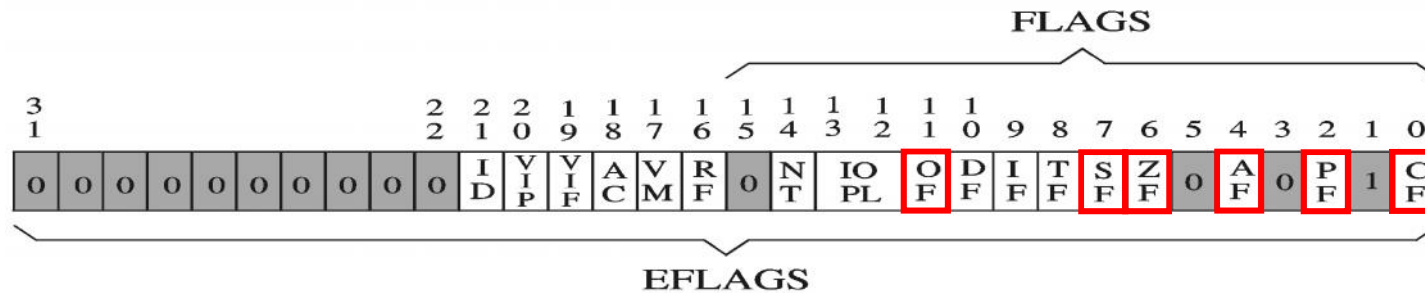
x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

- ❖ XOR instruction is  
often used to  
**invert selected bits**





# Affected Status Flags



The six status flags are affected

1. Carry Flag: **Cleared** by AND, OR, and XOR
2. Overflow Flag: **Cleared** by AND, OR, and XOR
3. Sign Flag: Copy of the **sign bit** in result
4. Zero Flag: Set when result is **zero**
5. Parity Flag: Set when parity in least-significant byte is **even**
6. Auxiliary Flag: **Undefined** by AND, OR, and XOR

# String Encryption Program

## ❖ Tasks:

- ✧ Input a message (string) from the user
- ✧ Encrypt the message
- ✧ Display the encrypted message
- ✧ Decrypt the message
- ✧ Display the decrypted message

## ❖ Sample Output

```
Enter the plain text: Attack at dawn.  
Cipher text: «ççÄîä-Äç-ïÄÿü-Gs  
Decrypted: Attack at dawn.
```

# Encrypting a String

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

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

```
    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
```

# Programming Assignments

- ❖ Write an assembly program that scans a string and converts all lower case characters to uppercase and all uppercase characters to lowercase. It should not change any other characters within the string.
- ❖ Write an assembly program that inputs a string, encrypts it using a XOR key and shows the cipher text on screen. It also decrypts the cipher text to show the original plain text.