

Week 7

Chapter 6: Conditional Processing

Chapter Overview

- **Boolean and Comparison Instructions**
- Conditional Jumps
- Conditional Loop Instructions
- Conditional Structures
- Application: Finite-State Machines
- Conditional Control Flow Directives

Boolean and Comparison Instructions

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

Status Flags - Review

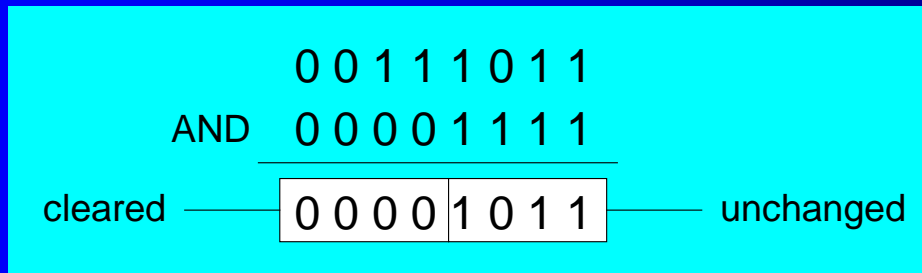
- The **Zero** flag is set when the result of an operation equals zero.
- The **Carry** flag is set when an instruction generates a result that is too large (or too small) for the destination operand.
- The **Sign** flag is set if the destination operand is negative, and it is clear if the destination operand is positive.
- The **Overflow** flag is set when an instruction generates an invalid signed result (bit 7 carry is XORed with bit 6 Carry).
- The **Parity** flag is set when an instruction generates an even number of 1 bits in the low byte of the destination operand.
- The **Auxiliary Carry** flag is set when an operation produces a carry out from bit 3 to bit 4

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)



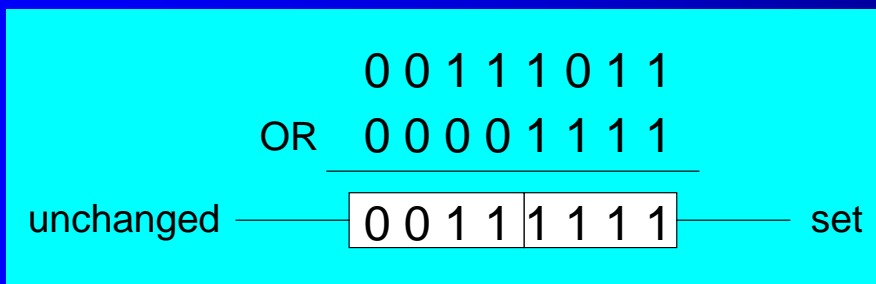
AND

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

OR Instruction

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

OR destination, source



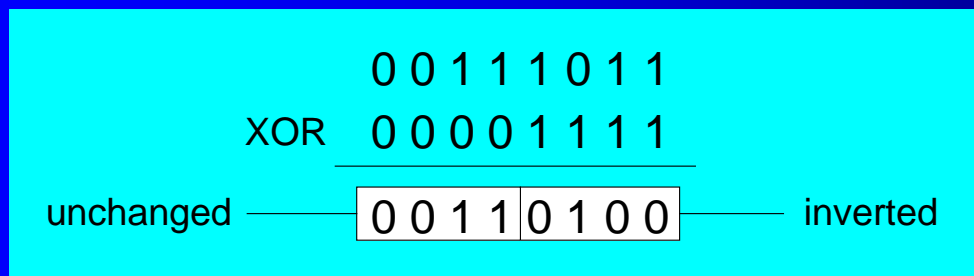
OR

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

XOR Instruction

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

XOR destination, source



XOR

x	y	$x \oplus y$
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.

NOT Instruction

- Performs a Boolean NOT operation on a single destination operand
- Syntax:

`NOT destination`

```
NOT  0 0 1 1 1 0 1 1
      ───────────
      1 1 0 0 0 1 0 0 ——— inverted
```

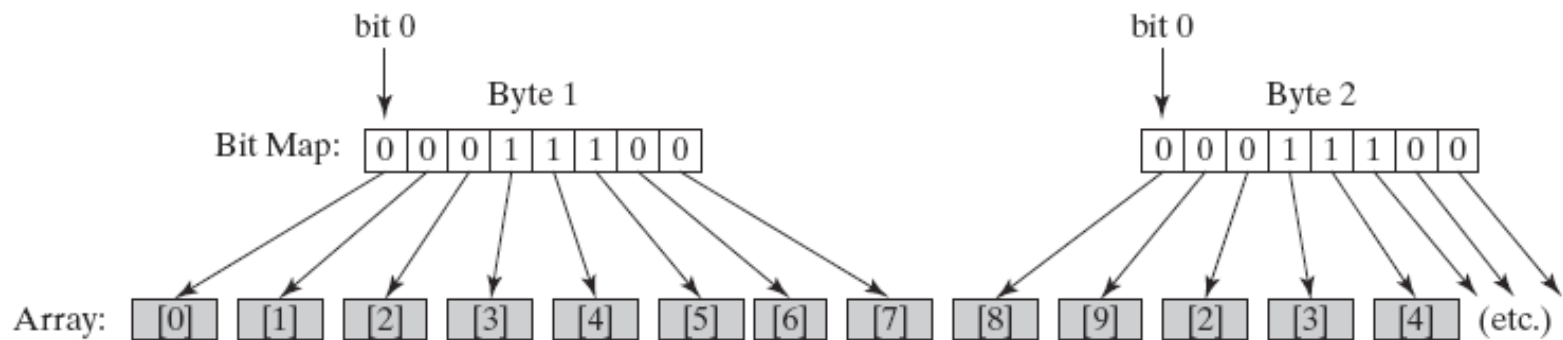
NOT

X	$\neg X$
F	T
T	F

Bit-Mapped Sets

- Binary bits indicate set membership
- Efficient use of storage
- Also known as *bit vectors*

FIGURE 6-1 Mapping Binary Bits to an Array.



Bit-Mapped Set Operations

- Set Complement

```
mov eax,SetX  
not eax
```

- Set Intersection

```
mov eax,setX  
and eax,setY
```

- Set Union

```
mov eax,setX  
or  eax,setY
```

Applications (1 of 5)

- Task: Convert the character in AL to upper case.
- Solution: Use the AND instruction to clear bit 5.

```
mov al,'a'           ; AL = 01100001b
and al,11011111b     ; AL = 01000001b
```

Applications (2 of 5)

- Task: Convert a binary decimal byte into its equivalent ASCII decimal digit.
- Solution: Use the OR instruction to set bits 4 and 5.

```
mov al,6                ; AL = 00000110b  
or  al,00110000b        ; AL = 00110110b
```

The ASCII digit '6' = 00110110b

Applications (3 of 5)

- Task: Turn on the keyboard CapsLock key
- Solution: Use the OR instruction to set bit 6 in the keyboard flag byte at 0040:0017h in the BIOS data area.

```
mov ax,40h                ; BIOS segment
mov ds,ax
mov bx,17h                ; keyboard flag byte
or BYTE PTR [bx],01000000b ; CapsLock on
```

This code only runs in Real-address mode, and it does not work under Windows NT, 2000, or XP.

Applications (4 of 5)

- 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.

Your turn: Write code that jumps to a label if an integer is negative.

Applications (5 of 5)

- 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.

TEST Instruction

- Performs a nondestructive AND operation between each pair of matching bits in two operands
- No operands are modified, but the Zero flag is affected.
- Example: jump to a label if either bit 0 or bit 1 in AL is set.

```
test al,00000011b  
jnz  ValueFound
```

- Example: jump to a label if neither bit 0 nor bit 1 in AL is set.

```
test al,00000011b  
jz   ValueNotFound
```


CMP Instruction (1 of 3)

- Compares the destination operand to the source operand
 - Nondestructive subtraction of source from destination (destination operand is not changed)
- Syntax: **CMP** *destination, source*
- Example: destination == source

```
mov al,5  
cmp al,5                ; Zero flag set
```

- Example: destination < source

```
mov al,4  
cmp al,5                ; Carry flag set
```

CMP Instruction (2 of 3)

- Example: destination > source

```
mov al,6  
cmp al,5                ; ZF = 0, CF = 0
```

(both the Zero and Carry flags are clear)

CMP Instruction (3 of 3)

The comparisons shown here are performed with signed integers.

- Example: destination > source

```
mov al,5  
cmp al,-2          ; Sign flag == Overflow flag
```

- Example: destination < source

```
mov al,-1  
cmp al,5           ; Sign flag != Overflow flag
```

Boolean Instructions in 64-Bit Mode

- 64-bit boolean instructions, for the most part, work the same as 32-bit instructions
- If the source operand is a constant whose size is less than 32 bits and the destination is the lower part of a 64-bit register or memory operand, all bits in the destination operand are affected
- When the source is a 32-bit constant or register, only the lower 32 bits of the destination operand are affected

What's Next

- Boolean and Comparison Instructions
- **Conditional Jumps**
- Conditional Loop Instructions
- Conditional Structures
- Application: Finite-State Machines
- Conditional Control Flow Directives

Conditional Jumps

- Jumps Based On . . .
 - Specific flags
 - Equality
 - Unsigned comparisons
 - Signed Comparisons
- Applications
- Encrypting a String
- Bit Test (BT) Instruction

Jcond Instruction

- A conditional jump instruction branches to a label when specific register or flag conditions are met
- Specific jumps:
 - JB, JC - jump to a label if the Carry flag is set
 - JE, JZ - jump to a label if the Zero flag is set
 - JS - jump to a label if the Sign flag is set
 - JNE, JNZ - jump to a label if the Zero flag is clear
 - JECXZ - jump to a label if ECX = 0

Jcond Ranges

- Prior to the 386:
 - jump must be within -128 to $+127$ bytes from current location counter
- x86 processors:
 - 32-bit offset permits jump anywhere in memory

Jumps Based on Specific Flags

Mnemonic	Description	Flags
JZ	Jump if zero	ZF = 1
JNZ	Jump if not zero	ZF = 0
JC	Jump if carry	CF = 1
JNC	Jump if not carry	CF = 0
JO	Jump if overflow	OF = 1
JNO	Jump if not overflow	OF = 0
JS	Jump if signed	SF = 1
JNS	Jump if not signed	SF = 0
JP	Jump if parity (even)	PF = 1
JNP	Jump if not parity (odd)	PF = 0

Jumps Based on Equality

Mnemonic	Description
JE	Jump if equal (<i>leftOp = rightOp</i>)
JNE	Jump if not equal (<i>leftOp \neq rightOp</i>)
JCXZ	Jump if CX = 0
JECXZ	Jump if ECX = 0

Jumps Based on Unsigned Comparisons

Mnemonic	Description
JA	Jump if above (if $leftOp > rightOp$)
JNBE	Jump if not below or equal (same as JA)
JAE	Jump if above or equal (if $leftOp \geq rightOp$)
JNB	Jump if not below (same as JAE)
JB	Jump if below (if $leftOp < rightOp$)
JNAE	Jump if not above or equal (same as JB)
JBE	Jump if below or equal (if $leftOp \leq rightOp$)
JNA	Jump if not above (same as JBE)

Jumps Based on Signed Comparisons

Mnemonic	Description
JG	Jump if greater (if $leftOp > rightOp$)
JNLE	Jump if not less than or equal (same as JG)
JGE	Jump if greater than or equal (if $leftOp \geq rightOp$)
JNL	Jump if not less (same as JGE)
JL	Jump if less (if $leftOp < rightOp$)
JNGE	Jump if not greater than or equal (same as JL)
JLE	Jump if less than or equal (if $leftOp \leq rightOp$)
JNG	Jump if not greater (same as JLE)

Applications (1 of 5)

- Task: Jump to a label if **unsigned** EAX is greater than EBX
- Solution: Use CMP, followed by JA

```
cmp eax,ebx  
ja  Larger
```

- Task: Jump to a label if **signed** EAX is greater than EBX
- Solution: Use CMP, followed by JG

```
cmp eax,ebx  
jg  Greater
```

Applications (2 of 5)

- Jump to label L1 if **unsigned** EAX is less than or equal to Val1

```
cmp eax,Val1  
jbe L1           ; below or equal
```

- Jump to label L1 if **signed** EAX is less than or equal to Val1

```
cmp eax,Val1  
jle L1
```

Applications (3 of 5)

- Compare unsigned AX to BX, and copy the larger of the two into a variable named **Large**

```
mov Large,bx
cmp ax,bx
jna Next
mov Large,ax
```

Next:

- Compare signed AX to BX, and copy the smaller of the two into a variable named **Small**

```
mov Small,ax
cmp bx,ax
jnl Next
mov Small,bx
```

Next:

Applications (4 of 5)

- Jump to label L1 if the memory word pointed to by ESI equals Zero

```
cmp WORD PTR [esi],0  
je  L1
```

- Jump to label L2 if the doubleword in memory pointed to by EDI is even

```
test DWORD PTR [edi],1  
jz   L2
```


Applications (5 of 5)

- Task: Jump to label L1 if bits 0, 1, and 3 in AL are **all set**.
- Solution: Clear all bits except bits 0, 1, and 3. Then compare the result with 00001011 binary.

```
and al,00001011b      ; clear unwanted bits
cmp al,00001011b      ; check remaining bits
je  L1                ; all set? jump to L1
```

Your turn . . .

- Write code that jumps to label L1 if **either** bit 4, 5, or 6 is set in the BL register.
- Write code that jumps to label L1 if bits 4, 5, and 6 are **all set** in the BL register.
- Write code that jumps to label L2 if AL has even parity.
- Write code that jumps to label L3 if EAX is negative.
- Write code that jumps to label L4 if the expression $(EBX - ECX)$ is greater than zero.

Encrypting a String

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

```
KEY = 239                ; can be any byte value
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
```

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

View the [Encrypt.asm](#) program's source code. Sample output:

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

BT (Bit Test) Instruction

- Copies bit *n* from an operand into the Carry flag
- Syntax: **BT** *bitBase*, *n*
 - *bitBase* may be *r/m16* or *r/m32*
 - *n* may be *r16*, *r32*, or *imm8*
- Example: jump to label L1 if bit 9 is set in the AX register:

```
bt AX,9          ; CF = bit 9
jc L1            ; jump if Carry
```