Microprocessors (0630371)
Fall 2010/2011 – Lecture Notes # 18

Conditional Jumps Instructions

- **No** high-level control structures in assembly language
- The most common way to transfer control in assembly language is to use a conditional jump. This is a two-step process:
  1. First test the condition.
  2. Then jump if the condition is true or continue if it is false.
- Conditional jump instructions can be divided into four groups:
  3. Jumps based on the value of a single arithmetic flag
  4. Jumps based on the value of CX or ECX
  5. Jumps based on comparisons of signed operands
  6. Jumps based on comparisons of unsigned operands
- Conditional Jump Instruction has the following syntax:

```
Jcond destination   ; cond is the jump condition
```

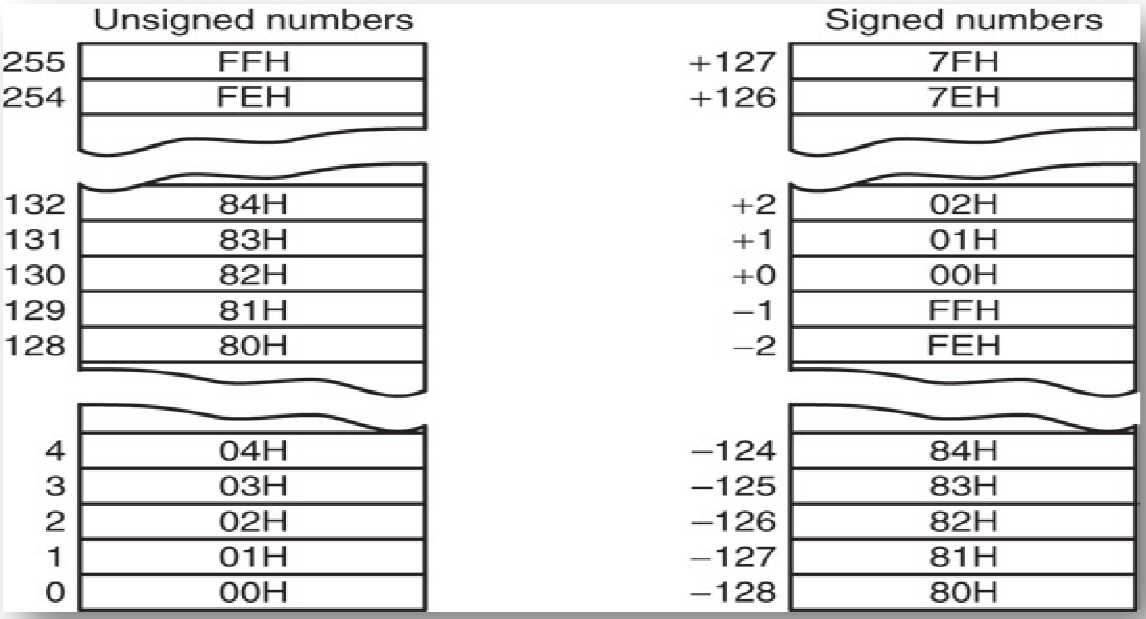
- The following is a list of jumps based on the **Zero, Carry, Overflow, Sign, and Parity** flags.

Mnemonic	Description	Flags
JZ, JE	Jump if Zero, Jump if Equal	ZF = 1
JNZ, JNE	Jump if Not Zero, Jump if Not Equal	ZF = 0
JC	Jump if Carry	CF = 1
JNC	Jump if No Carry	CF = 0
JO	Jump if Overflow	OF = 1
JNO	Jump if No Overflow	OF = 0
JS	Jump if Signed (Negative)	SF = 1
JNS	Jump if Not Signed (Positive or Zero)	SF = 0
JP, JPE	Jump if Parity, Jump if Parity is Even	PF = 1
JNP, JPO	Jump if Not Parity, Jump if Parity is Odd	PF = 0

- The following table shows the jumps based on the value of **CX** and **ECX**:

Mnemonic	Description
JCXZ	Jump if CX = 0
JECXZ	Jump if ECX = 0

Signed and unsigned numbers follow different orders.



- The following table shows a list of signed jumps based on comparisons of signed operands:

Mnemonic	Description	Condition Tested
JG, JNLE	Jump if Greater, Jump if Not Less or Equal	ZF = 0 and SF = OF
JGE, JNL	Jump if Greater or Equal, Jump if Not Less	SF = OF
JL, JNGE	Jump if Less, Jump if Not Greater or Equal	SF ≠ OF
JLE, JNG	Jump if Less or Equal, Jump if Not Greater	ZF = 1 or SF ≠ OF

- The following shows a list of unsigned jumps based on comparisons of unsigned operands:

Mnemonic	Description	Condition Tested
JA, JNBE	Jump if Above, Jump if Not Below or Equal	ZF = 0 and CF = 0
JAE, JNB	Jump if Above or Equal, Jump if Not Below	CF = 0
JB, JNAE	Jump if Below, Jump if Not Above or Equal	CF = 1
JBE, JNA	Jump if Below or Equal, Jump if Not Above	ZF = 1 or CF = 1

- All conditional jumps except two (JCXZ and JECXZ) use the processor flags for their criteria. Thus, any statement that sets or clears a flag can serve as a test basis for a conditional jump. The jump statement can be any one of 30 conditional-jump instructions

Programming Examples

Example 1: 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
```

Example 2: Write code that jumps to a label if an integer is negative.

- 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.

Example 3: jump to a label if either bit 0 or bit 1 in AL is set.

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

Example 4: jump to a label if neither bit 0 nor bit 1 in AL is set.

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

Example 5: Jump to a label if unsigned EAX is greater than EBX

- Solution: Use CMP, followed by JA

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

Example 6: Jump to a label if signed EAX is greater than EBX

- Solution: Use CMP, followed by JG

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

Example 7: Jump to label L1 if unsigned EAX is less than or equal to Val1

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

Example 8: Jump to label L1 if signed EAX is less than or equal to Val1

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

Example 9: 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:
```

Example 10: 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:
```

Example 11: Jump to label L1 if the memory word pointed to by ESI equals Zero

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

Example 12: Jump to label L2 if the doubleword in memory pointed to by EDI is even

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

Example 13: 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
```

Try to

- ✧ **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.**

Example 14:

```
TITLE Finding the Maximum of 3 Integers (max.asm)
     .686
     .MODEL flat, stdcall
     .STACK
     INCLUDE Irvine32.inc
     .data
     var1    DWORD  -30  ; Equal to FFFFFFE2 (hex)
     var2    DWORD  12
     var3    DWORD  7
     max1    BYTE    "Maximum Signed Integer = ",0
     max2    BYTE    "Maximum Unsigned Integer = ",0
     .code
     main PROC
         ; Finding Signed Maximum
         mov eax, var1
         cmp eax, var2
         jge L1
         mov eax, var2
     L1:
         cmp eax, var3
         jge L2
         mov eax, var3
     L2:
         lea  edx, max1
         call WriteString
         call WriteInt
         call Crlf
         ; Finding Unsigned Maximum
         mov eax, var1
         cmp eax, var2
         jae L3
         mov eax, var2
```

```
L3:
    cmp eax, var3
    jae L4
    mov eax, var3
L4:
    lea  edx, max2
    call WriteString
    call WriteHex
    call Crlf
    exit
main ENDP
END main
```

Example 15:
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**

To encrypt and decrypt the text , we use the following interesting property of xor instruction

$$((X \oplus Y) \oplus Y) = X$$

```
TITLE Encryption Program                (Encrypt.asm)
    ; This program demonstrates simple symmetric
    ; encryption using the XOR instruction.
    INCLUDE Irvine32.inc
    KEY = 239           ; any value between 1-255
    BUFMAX = 128        ; maximum buffer size
    .data
    sPrompt  BYTE  "Enter the plain text: ",0
    sEncrypt BYTE  "Cipher text:          ",0
    sDecrypt BYTE  "Decrypted:            ",0
    buffer   BYTE   BUFMAX+1 DUP(0)
    bufSize  DWORD  ?
    .code
    main PROC
         call InputTheString      ; input the plain text
         call TranslateBuffer     ; encrypt the buffer
         mov  edx,OFFSET sEncrypt ; display encrypted message
         call DisplayMessage
         call TranslateBuffer     ; decrypt the buffer
         mov  edx,OFFSET sDecrypt ; display decrypted message
         call DisplayMessage
         exit
    main ENDP
```

```
;-----------------------------------------------------
InputTheString PROC
; Prompts user for a plaintext string. Saves the string
; and its length.
; Receives: nothing
; Returns: nothing
;-----------------------------------------------------
     pushad
     mov  edx,OFFSET sPrompt  ; display a prompt
     call WriteString
     mov  ecx,BUFMAX          ; maximum character count
     mov  edx,OFFSET buffer   ; point to the buffer
     call ReadString          ; input the string
     mov  bufSize,eax         ; save the length
     call Crlf
     popad
     ret
InputTheString ENDP
;-----------------------------------------------------
DisplayMessage PROC
; Displays the encrypted or decrypted message.
; Receives: EDX points to the message
; Returns:  nothing
;-----------------------------------------------------
     pushad
     call WriteString
     mov  edx,OFFSET buffer   ; display the buffer
     call WriteString
     call Crlf
     call Crlf
     popad
     ret
DisplayMessage ENDP
;-----------------------------------------------------
TranslateBuffer PROC
; Translates the string by exclusive-ORing each
; byte with the encryption key byte.
; Receives: nothing
; Returns: nothing
     pushad
     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
     popad
     ret
TranslateBuffer ENDP
END main
```

Example 15: Sequential Search

```
; Receives:    esi = array address
;              ecx = array size
;              eax = search value
; Returns:     esi = address of found element
search PROC USES ecx
   jecxz notfound
L1:
   cmp  [esi], eax   ; array element = search value?
   je   found   ; yes? found element
   add  esi, 4  ; no? point to next array element
   loop L1
notfound:
   mov  esi, 0  ; if not found then esi = 0
found:
   ret          ; if found, esi = element address
search ENDP
```

Example 16: Scanning an Array

```
TITLE Scanning an Array              (ArryScan.asm)
; Scan an array for the first nonzero value.
INCLUDE Irvine32.inc
.data
intArray SWORD  0,0,0,0,1,20,35,-12,66,4,0
;intArray SWORD  1,0,0,0
;intArray SWORD  0,0,0,0
;intArray SWORD  0,0,0,1
noneMsg  BYTE "A non-zero value was not found",0
.code
main PROC
   mov   ebx,OFFSET intArray        ; point to the array
   mov   ecx,LENGTHOF intArray      ; loop counter
L1:
   cmp   WORD PTR [ebx],0      ; compare value to zero
   jnz   found       ; found a value
   add   ebx,2       ; point to next
   loop  L1          ; continue the loop
   jmp   notFound          ; none found
found:
   movsx eax,WORD PTR [ebx]    ; otherwise, display it
   call  WriteInt
   jmp   quit
notFound:
   mov   edx,OFFSET noneMsg    ; display "not found" message
   call  WriteString
quit:
   call  crlf
   exit
main ENDP
END main
```