Flow Control Instructions

Computer Organization and Assembly Language

Agenda

- Introduction
- Conditional Jumps
- JMP instruction
- Branching Structures
- Loops

Flow Control

- The jump and loop instructions transfer control to another part of program.
- This transfer can be unconditional or conditional depending on a particular combination of status flag settings.

Conditional Jumps

- *JNZ* is a conditional jump instruction. The syntax is
- Jxxx destination_label

Conditional Jumps

- If the condition for the jump is true, the next instruction to be executed is the one at destination_label, which may precede or follow the jump instruction.
- If the condition is false, the instruction Immediately following the jump executed.
- For JNZ, the condition Is that the result of the previous operation is not zero

Conditional Jumps

- To demonstrate the jump instructions the program below displays the IBM character set.

```
.model small
.stack 100h
.code
MAIN PROC
        MOV AH, 2
        MOV CX, 256    ; Loop counter, number of characters to display
        MOV DL, 0      ; Contains ASCII code of character to display, starting with 0
DISPLAY_LOOP:
        INT 21H        ; Output the character in DL
        INC DL         ; Update to next character
        DEC CX         ; Decrement loop counter
        JNZ DISPLAY_LOOP  ; Repeat the statements if CX is not 0
MAIN ENDP
END MAIN
```

Conditional Jumps

- To display the characters, a loop is used (JNZ DISPLAY_LOOP) instruction.
- Before entering the loop, AH Is initialized to 2 (single character display) and DL is set to 0, the initial ASCII code.
- CX Is the loop counter; it is set to 256 before entering the loop and is decremented after each character is displayed.
- The instruction that controls the loop is *JNZ (*Jump if Not Zero).
- If the result of the preceding Instruction (DEC CX) is not zero, the *JNZ* instruction transfers control to the instruction at label DISPLAY_LOOP.
- When CX finally contains 0, the loop ends.

CPU Implementing a Conditional Jump

- To implement a conditional jump, the CPU looks at the FLAGS register, which reflects the result of last instruction.
- If the condition for the jump (a combination of status FLAGS settings) are true, the CPU adjusts the IP to point to the destination label, so that the instruction at this label will be done next.
- If the jump condition is false, then IP is not altered; which skips the jump and executes the next instruction in code.
- In last example, CPU executes JNZ DISPLAY_LOOP by checking ZF.
  - If ZF = 0, control transfers to PRINT_LOOP; if ZF = 1, the program goes on to execute following instruction.

Conditional Jump Categories

- There are three categories;
- (1) signed jumps are used when a signed interpretation is being given to results,
- (2) the unsigned jumps are used for an unsigned interpretation, and
- (3) the single-flag jumps, which operate on settings/of individual flags.
- The jump instructions themselves do not affect the flags.

Signed Jumps

- Given below are signed jumps

Symbol	***Description***	***Condition for Jumps***
JG/JNLE	jump if greater than jump if not less than or equal to	ZF = 0 and SF = OF
JGE/JNL	jump if greater than or equal to jump if not less than or equal to	SF = OF
JL/JNGE	jump if less than jump if not greater than or equal	SF <> OF
JLE/JNG	jump if less than or equal jump if not greater than	ZF = 1 or SF <> OF

Unsigned Conditional Jumps

- Given below are unsigned jumps

Symbol	Description	Condition for Jumps
JA/JNBE	jump if above jump if not below or equal	CF = 0 and ZF = 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 equal jump if not above	CF = 1 or ZF = 1

Single-Flag Jumps

- Given below are *Single-Flag* jumps

Symbol	Description	Condition for Jumps
JE/JZ	jump if equal	ZF = 1
	jump if equal to zero	
JNE/JNZ	jump if not equal	ZF = 0
	jump if not zero	
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 sign negative	SF = 1
JNS	jump if nonnegative sign	SF = 0
JP/JPE	jump if parity even	PF = 1
JNP/JPO	jump if parity odd	PF = 0

CMP Instruction

- The jump condition is often provided by the CMP *(compare)* instruction.
- It has the form

  CMP destination, source
- This instruction compares destination and source by computing destination contents minus source contents.
- The result is not stored, but the flags are affected.
- The operands of CMP may not both be memory locations.
- Destination may not be a constant.
- CMP is like SUB, except that destination is not changed.

CMP Example

- For following instructions:

  CMP AX, BX

  JG BELOW

- where AX = 7FFFh, and BX = 0001.
- The result of CMP AX,BX is 7ffFh - 0001h = 7FFEh.
- The jump condition for JG Is satisfied (see the jump tables), because ZF = SF = OF = 0, so control transfers to label BELOW.

Interpreting Conditional Jumps

- In the last example, we determined by looking at the flags after CMP was executed that control transfers to label BELOW.
- This is how the CPU Implements a conditional jump.
- But a programmer can just use the name of the jump to decide if control transfers to the destination label. In the following,

```
CMP AX,BX
JG BELOW
```

- If AX is greater than BX (in a signed sense), then JG (jump if greater than) transfers to BELOW.

Conditional Jump without CMP

- The conditional jump can also work with other instructions apart from CMP. For example:

  DEC AX

  JZ NEXT

- If the contents of AX become 0, control transfers to NEXT.

Signed Versus Unsigned Jumps

- Each of the signed jumps corresponds to an analogous unsigned jump; for example, the signed jump JG and the unsigned jump JA.
- The table above on jumps shows that these jumps operate on different flags: the signed jumps operate on ZF, SF, and OF, while the unsigned jumps operate on ZF and CF.
- Using the wrong kind of jump can lead to incorrect results.
- Example: suppose we're giving a signed interpretation. If AX = 7FFFh, BX = 8000h, and we execute

  CMP AX,BX

  JA BELOW
- Even though 7FFFh > 8000h in a signed sense, the program does jump to BELOW, because 7FFFh < 8000h in an unsigned sense, and we are using the unsigned jump JA.

Unconditional Jump - JMP Instruction

- The JMP *(jump)* instruction causes a unconditional transfer of control. The syntax is

  JMP destination

- where destination is usually a label in the same segment as the JMP itself.
- JMP can be used to get around the range restriction of a conditional jump.

Branching Structures

- Branching structures enable a program to take different paths, depending on conditions.

Branching Structures - IF-THEN

- The IF-THEN structure may be- expressed in pseudocode as follows:

  IF *condition* is true

  THEN

  execute true-branch statements

  END_IF

- The *condition* is an expression that is true or false.
- If the condition is true, the true branch statement is executed.
- If the condition is false, nothing is done.

Branching Structures - Example

- Replace the number in AX by its absolute value.

```
CMP AX, 0  ; AX< 0
JNL ENDIF   ;no, exit
NEG AX
ENDIF
```

- The condition AX < 0 is expressed by CMP AX,O.
- If AX is not less than 0, nothing is done, so use a JNL (jump if not less) to jump around the NEG AX.
- If condition AX < 0 is true, the program goes on to execute NEG AX.

Branching Structures - IF-THEN-ELSE

- The IF-THEN structure may be expressed in pseudocode as follows:

```
IF condition is true
    THEN
        execute true-branch statements
END_IF
ELSE
        execute false-branch statements
END_ELSE
```

Branching Structures - IF-THEN-ELSE - Example

- Example: Register AL and BL both contain a value. Display the smaller of these values.
- Pseudocode:

```
IF AL <= BL
    THEN
    Display the character in AL
ELSE
    display the character in BL
END - IF
```

Branching Structures - IF-THEN-ELSE - Example

- Assembly code:

```
MOV AL, 'x'
MOV BL, 'y'
CMP AL, BL
JNBE ELSE_PART
        MOV DL, AL          ;IF BODY
        JMP DISPLAY
ELSE_PART:
        MOV DL, BL          ; ELSE BODY
DISPLAY:
        MOV AH, 2
        INT 21H
```

Branches with Compound Conditions

- The branching condition can also take multiple conditions
- condition_1 AND condition_2
- where condition_1 and condition_2 are either true or false.
- An AND condition is true if and only if condition_1 and condition_2 are both true.
- Likewise, if either condition is false, then the whole thing is false.

AND Conditions - Example

- Example: Read a character, and if it's an uppercase letter, display it.
- MOV AH, 1 ; To read a character
- INT 21H
- ;if ('A' <= char> and (char <= 'Z')
- CMP AL, ‘A’
- JNGE END_IF
- CMP AL, ‘Z’
- JNLE END_IF
- ; THEN DISPLAY THE CHARACTER
- MOV DL, AL
- MOV AH, 2
- INT 21H
- END_IF:

OR Conditions

- Condition_1 OR condition_2 is true if at least one of the conditions is true; it is only false when both conditions are false.
- **Example:** Read a character. If it's "y" or "Y", display it; otherwise, terminate the program.

OR Condition Example

- Code:

```
MOV AH, 1          ;Input character
INT 21H
CMP AL, 'y'
JE THEN_PART
CMP AL, 'Y'
JE THEN_PART
JMP END_IF         ;both false, terminate
THEN_PART:
    MOV AH, 2
    MOV DL, AL
    INT 21H
END_IF:
```

Looping Structures

- A **loop** is a sequence of instructions that is repeated.
- The number of times to repeat may be known in advance, or It may depend on conditions.
- ***FOR LOOP***
- This is a loop structure in which the loop statements are repeated a known number of times (a count-controlled loop).
- Pseudocode:

```
FOR loop_count_times DO
    Statements
END_FOR
```

LOOP Instruction

- The **LOOP** instruction can be used to implement a for loop.
- It has the form

  LOOP destination_label
- The counter for the loop is the register CX which is initialized to loop_count.
- Execution of the LOOP Instruction causes CX to be decremented automatically.
- If CX is not 0, control transfers to destination_label.
- If CX is 0, the next instruction after LOOP is executed.
- DestinatIon_label must precede the LOOP instruction by no more than 126 bytes.

FOR LOOP Example

- Example: Write a count-controlled loop to display a row of 20 stars.

```
    MOV CX, 20
    MOV AH, 2
    MOV DL, '*'
REPEAT:
    INT 21H
    LOOP REPEAT
```

WHILE LOOP

- Pseudocode:

  WHILE *condition DO*

  Statements

  END WHILE

- The *condition* is checked at the top of the loop.
- If true, the statements are executed; if false, the loop terminates.

WHILE LOOP Example

- Example: Write some code to count the number of characters in an input line.

```
        MOV DX, 0       ;       char counter
        MOV AH, 1
        INT 21H
WHILE_REPEAT:
        CMP AL, 0DH             ;0DH = Carriage Return
        JE END_WHILE
        INC DX
        INT 21H
        JMP WHILE_REPEAT
END_WHILE:
```