Flow Control Instructions

Label

- Labels are needed in situations where one instruction refers to another
- Labels end with a colon
- Label usually placed on a line by themselves
- They refer to the instruction that follows

Conditional Jumps

- If the condition for the jump is true, the next instruction to be executed is the one at destination_label
- If the condition is false, the instruction immediately following the jump is done next
- Syntax

  ```
  Jump_instruction destination_label
  ```

- Range
  - The destination label must precede or follow the jump instruction no more than 126 bytes

Conditional Jump

- If the conditions for the jump instruction that is the combination of status flag settings are true, the CPU adjusts the IP to point to the destination label so that the instruction at this label will be executed next
- If the jump condition is false, then IP is not altered
- There are three categories of conditional jumps-
  - Signed conditional jumps
  - Unsigned conditional jumps
  - Single-flag jumps

Signed Conditional Jump

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

Unsigned Conditional Jump

Symbol	Description	Condition for Jumps
JA/JNBE	Jump if above	CF=0 and ZF=0
	Jump if not below or equal	
JAE/JNB	Jump if above or equal	CF=0
	Jump if not below	
JB/JNAE	Jump if below	CF=1
	Jump if not above or equal	
JBE/JNA	Jump if below or equal	CF=1 or ZF=1
	Jump if not above	

Single-Flag Conditional Jump

Symbol	Description	Condition for Jumps
JE/JZ	Jump if equal	ZF=1
	Jump if equal to or 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

The CMP Instruction

- The jump condition is often provided by the CMP(compare) instruction
- Syntax

```
CMP destination, source
```

- Does the compare by subtracting the source from the destination
- The result is not stored
- Only the flags are affected
- The operands of CMP may not both be memory locations
- Destination operand may not be a constant

Example

- Suppose AX and BX contain signed numbers. Write some code to put the biggest one in CX
- Solution

```
MOV CX,AX
CMP BX,CX
JLE NEXT
MOV CX,BX
NEXT:
```

The JMP instruction

- The JMP instruction causes an unconditional jump
- JMP can be used to get around the range restriction of a conditional jump
- Syntax

```
JMP destination_label
```

High-level Language Branching Structures

- IF-THEN
  - Syntax

    IF condition is true

        THEN

            execute true branch statements

    END_IF

High-level Language Branching Structures

- IF-THEN-ELSE
  - Syntax

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

High-level Language Branching Structures

- CASE
  - It is a multiway branch structure that tests a register, variable or expression for particular values or range of values
  - Syntax

```
CASE expression
        values_1: statements_1
        values_2: statements_2
        .
        .
        values_n: statements_n
END_CASE
```

Example of IF-THEN

Replace the number in AX by its absolute value

Pseudocode Algorithm	Assembly Code
IF AX<0 THEN replace AX by –AX END_IF	CMP AX,0 JNL END_IF NEG AX END_IF:

Example of IF-THEN-ELSE

Suppose AL and BL contain extended ASCII characters. Display the one that comes first in the character sequence.

Pseudocode Algorithm	Assembly Code

IF AL<=BL
THEN
display the character in AL
ELSE
display the character in BL
END_IF
	MOV AH,2
	CMP AL,BL
	JNBE ELSE_
	MOV DL,AL
	JMP DISPLAY
	ELSE_:
	MOV DL,BL
	DISPLAY:
	INT 21H
	END_IF:

Example of CASE

If AX contains a negative number, put -1 in BX; if AX contains 0,put 0 in BX; if AX contains a positive number, put 1 in BX.

Pseudocode Algorithm	Assembly Code

CASE AX
  <0: put -1 in BX
  =0: put 0 in BX
  >0: put 1 in BX
END_CASE
	CMP AX,0
	JL NEGATIVE
	JE ZERO
	JG POSITIVE
	NEGATIVE:
	MOV BX,-1
	JMP END_CASE
	ZERO:
	MOV BX,0
	JMP END_CASE
	POSITIVE:
	MOV BX,1
	END_CASE:

High-level Language Branching Structures with Compound Conditions

- AND
  - An AND condition is true if and only if condition_1 and condition_2 are both true
  - Syntax

    ```
    condition_1 AND condition_2
    ```
- OR
  - An OR condition is true if at least one of condition between condition_1 and condition_2 are true
  - Syntax

    ```
    condition_1 OR condition_2
    ```

Example of AND

- Read a character and if it is an uppercase letter display it.

Pseudocode Algorithm	Assembly Code

Read a character
IF (‘A’>=character) and
(character<=‘Z’)
THEN
display character
END_IF
	MOV AH,1
	INT 21H
	
	CMP AL,’A’
	JNGE END_IF
	CMP AL,’Z’
	JNLE END_IF
	
	MOV DL,AL
	MOV AH,2
	INT 21H
	END_IF:

Example of OR

Read a character and if it's 'y' or 'Y' display it otherwise terminate the program.

Pseudocode Algorithm	Assembly Code

Read a character
IF (character='y') OR (character='Y')
THEN
display it
ELSE
terminate the program
END_IF
	MOV AH,1
	INT 21H
	CMP AL, 'y'
	JE THEN
	CMP AL,'Y'
	JE THEN
	JMP ELSE_
	THEN:
	MOV AH,2
	MOV DL,AL
	INT 21H
	JMP END_IF
	ELSE_:
	MOV AH,4CH
	INT 21H
	END_IF:

High-level Language Looping Structures

- FOR
  - This is a loop structure in which the loop statements are repeated a known number of times
  - Syntax

    ```
    LOOP destination_label
    ```

  - The counter for the loop is the register CX which is initialized to `loop_count`
  - Execution of LOOP instruction causes CX to be decremented automatically.

Example of FOR Loop

Write a count-controlled loop and display a row of 80 stars.

Pseudocode Algorithm	Assembly Code

FOR 80 times DO
display '*'
END_FOR
	MOV CX,80
	MOV AH,2
	MOV DL,'*'
	TOP:
	INT 21H
	LOOP TOP

The JCXZ Instruction

- It is used before the loop instruction to check the value of CX
- If CX is zero then it helps to terminate the program
- Syntax
  - `JCXZ destination_label`

High-level Language Looping Structures

- WHILE
  - This is a loop structure which is depend on a condition
  - Syntax

    WHILE condition DO

        statements

    END_WHILE
  - The condition is checked at the top of the loop
  - The loop executes as long as the condition is true

Example of WHILE Loop

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

Pseudocode Algorithm	Assembly Code

Initialize count to 0
Read a character
WHILE character <> carriage_return
DO
count=count+1
read a character
END_WHILE
	MOV DX,0
	MOV AH,1
	INT 21H
	WHILE_:
	CMP AL,0DH
	JE END_WHILE
	INC DX
	INT 21H
	JMP WHILE_
	END_WHILE:

High-level Language Looping Structures

- REPEAT
  - This is a conditional loop structure
  - Syntax

    REPEAT

      statements

    UNTIL condition
  - The condition is checked after the statements are executed

Example of REPEAT Loop

Write some code to read characters until a blank is read.

Pseudocode Algorithm	Assembly Code
REPEAT read a character UNTIL character is a blank	MOV AH,1 REPEAT: INT 21H CMP AL,' ' JNE REPEAT END_WHILE: