Microprocessor and Assembly Language CSC321

Sheeza Zaheer

Lecturer

COMSATS UNIVERSITY ISLAMABAD

LAHORE CAMPUS

Flow Control Instructions

OUTLINE

- **Flow Control Instructions**
  - Unconditional Jumps
  - Conditional Jumps
    - Signed Jumps
    - Unsigned Jumps
  - Branching Structures
  - Looping Structures
- **References**
  - **Chapter 6,** Ytha Yu and Charles Marut, “Assembly Language Programming and Organization of IBM PC

Unconditional Jump

- Syntax:

  JMP *destination_label*

- Purpose: To Transfer control to another part of the program.
- Example:

```
ORG 100h
.CODE
MOV AX, 2
MOV BX, 2
JMP LABEL_SUB
ADD AX, BX   ;this instruction will never execute
LABEL_SUB:
                SUB AX, BX
RET
```

Conditional Jump

- Syntax:

  Jxxx destination_label

  where xxx represents the condition
- If condition is **true**, the next instruction to be executed is the one at destination_label.
- If condition is **false**, the instruction immediately following the jump is done next.
- To implement a conditional jump, the CPU looks at the FLAG register (set by last instruction executed by the processor).
- JUMP instructions themselves do not affect flags.

Categories of Conditional Jump

- **Signed Jumps**: are used when a signed interpretation is being given to result.

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	SF = OF
JL/JNGE	Jump if less than Jump if not greater than or equal to	SF <> OF
JLE/JNG	Jump if less than or equal to Jump if not greater than	ZF = 1 and SF <> OF

Contd..

- **Unsigned Jumps**: are used for an unsigned interpretation of result

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

Contd.

- **Single-Flag Jumps**: which operate on settings of individual flags.

Symbol	Description	Condition for Jumps
JE/JZ	Jump If Equal Jump If Equal to Zero	ZF = 1
JNE/JNZ	Jump If Not Equal Jump If Not Equal to 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 Sign Negative	SF = 1
JNS	Jump If Sign Non Negative	SF = 0
JP/JPE	Jump if parity even	PF = 1
JNP/JPO	Jump if parity not even/jump if parity odd	PF = 0

CMP Instruction

- CMP (compare) instruction performs an implied subtraction of a source operand from destination operand. Neither operand is modified.

  CMP destination, source

- **FLAGS**

CF	ZF	CMP Result
1	0	Destination < Source
0	0	Destination > Source
0	1	Destination = Source

CMP Instruction Examples

- Destination < Source:

```
mov     ax, 5
cmp     ax, 10          ;CF = 1, ZF = 0
```

- Destination = Source

```
mov      ax, 1000
mov     cx, 1000
cmp     cx, ax          ; ZF = 1, CF = 0
```

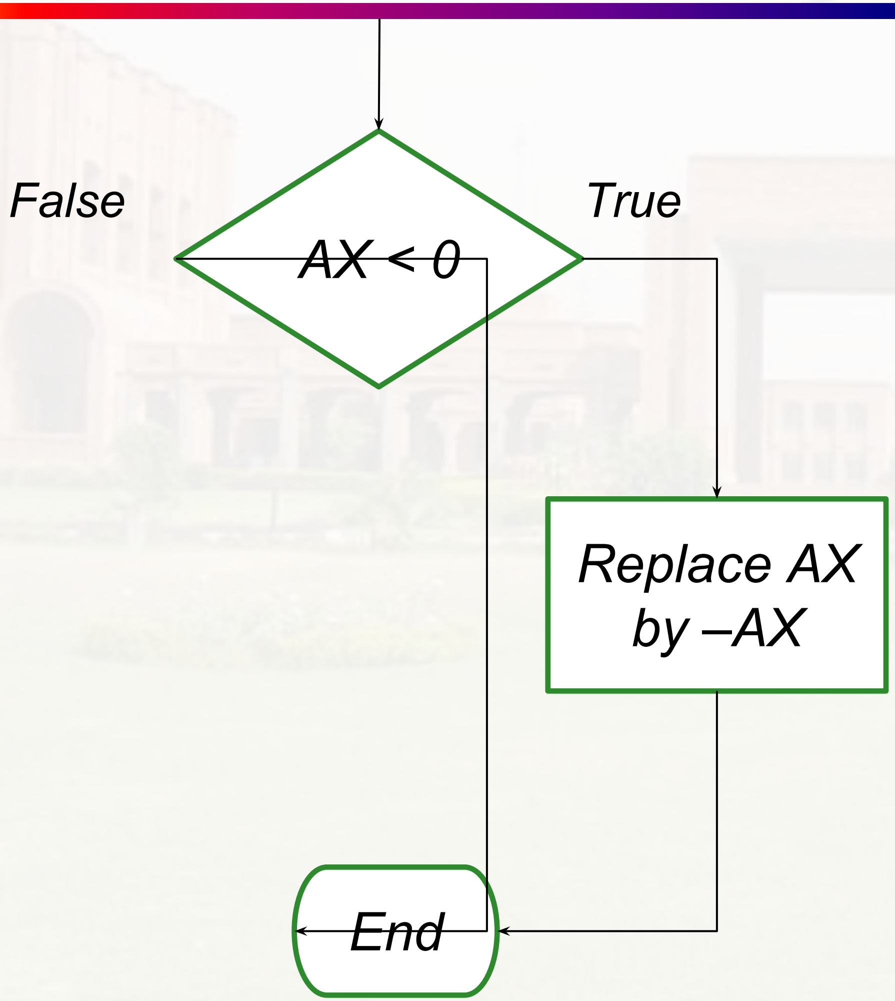
- Destination > Source

```
mov     si, 105
cmp     si, 0           ; ZF = 0 and CF = 0
```

High Level Language Structures

- Branching Structure
  - IF-THEN
  - IF-THEN-ELSE
  - CASE
  - Branching with Compound Conditions
    - AND CONDITIONS
    - OR CONDITIONS
- Looping Structures
  - FOR LOOP
  - WHILE LOOP
  - REPEAT LOOP

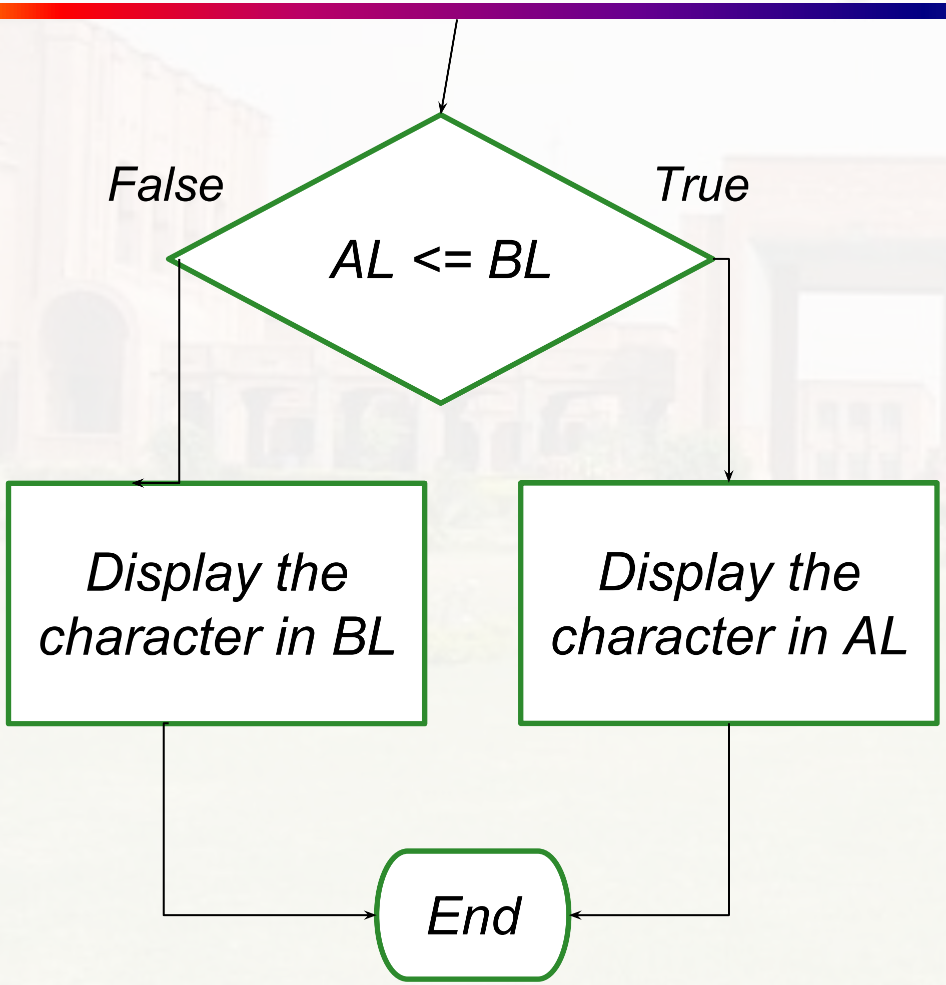
IF-THEN



;Code in Assembly Language:

```
ORG 100h
.CODE
MOV AX, FFFE
CMP AX, 0
JL IF1
JMP END_IF
IF1:
        NEG AX
END_IF:
        MOV AH, 4CH
        INT 21H
```

IF-THEN-ELSE



;Code in Assembly Language:

```
ORG 100h
.CODE
MOV AH, 1
INT 21H
MOV BL, AL ;first input in BL
INT 21H
;second input in AL
MOV AH, 2
CMP AL, BL ;If AL <= BL
JLE IF1
MOV DL, BL ;then
JMP DISPLAY
IF1:
      MOV DL, AL
DISPLAY:
      INT 21H
```

CASE

```
CASE AL
    < 0: put -1 in BL
    = 0: put 0 in BL
    > 0: put 1 in BL
END_CASE
```

```
;Code in Assembly Language:
ORG 100h
.CODE
MOV AH, 1
INT 21H         ;input in AL
CMP AL, 0       ;case
JL NEGATIVE
JE ZERO
JG POSITIVE
NEGATIVE:
        MOV BL, -1
        JMP END_CASE
ZERO:
        MOV BL, 0
        JMP END_CASE
POSITIVE:
        MOV BL, 1
        JMP END_CASE
END_CASE:
        MOV AH, 4Ch
        INT 21h
```

AND Condition

- An AND condition is true if and only if both conditions are true.
- Consider following Pseudo code:

```
Read a character (into AL)
IF (‘A’ <= character) and (character <= ‘Z’)
THEN
    display character
END_IF
```

Contd..

```
ORG 100h
.CODE
MOV AH, 1
INT 21H
;character in AL
CMP AL, ‘A’           ;if  AL >= ‘A’
JNGE END_IF
CMP AL, ‘Z’           ;and  AL <= ‘Z’
JNLE END_IF
MOV AH, 2
MOV DL, AL  ;then
INT 21H
END_IF:
      MOV AH, 4Ch
      INT 21H
```

OR Condition

- An OR condition is true if at least one of the conditions is true.
- Consider following Pseudo code:

```
Read a character (into AL)
IF (character = ‘y’) OR (character = ‘Y’)
THEN
    display it
ELSE
    terminate the program
END_IF
```

Contd..

```
ORG 100h
.CODE
MOV AH, 1
INT 21H             ;character in AL
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:
```

Looping Structures

- FOR LOOP

```
;initialize CX
TOP:
    ;body of the loop
    LOOP TOP
```

```
ORG 100h
.CODE
MOV CX, 80
MOV AH, 2
MOV DL, '*'
TOP:
    INT 21H
    LOOP TOP

MOV AH, 4Ch
INT 21H
```

Contd..

- FOR LOOP
  - Executed at least once.
  - If CX contains 0, the loop instruction decrements CX (CX = FFFFh) and the loop is then executed 65535 times!
  - To prevent this, use instruction JCXZ before the loop.

```
;initialize CX
JCXZ SKIP
TOP:
    ;body of the loop
    LOOP TOP
SKIP:
```

Contd..

- WHILE LOOP

```
WHILE condition DO
    ;statements
END_WHILE
```

- WHILE LOOP checks the terminating condition at the top of the loop, so don’t forget to initialize variables.
- Example:

```
Initialize count to 0
Read a character
WHILE character <> carriage return Do
    count = count + 1
    read a character
END_WHILE
```

Contd..

```
ORG 100h
.CODE
MOV DX, 0
MOV AH, 1   ;read first character
INT 21H
WHILE_:
    CMP AL, 0Dh
    JE END_WHILE
    INC DX
    INT 21H
    JMP WHILE_
END_WHILE:
MOV AH, 4Ch
INT 21H
```

Note: Requires 2 Jumps:

- *Conditional Jump at top*
- *JMP at the bottom*

Also, If terminating condition is false, loop is not executed.

Contd..

- REPEAT LOOP

```
REPEAT
    ;statements
UNTIL condition
```

- First statements are executed, then the condition is checked.
- If true, the loop terminates; if false, control branches to the top of the loop
- Example:

```
REPEAT
    Read a character
UNTIL character is a blank
```

Contd..

```
ORG 100h
.CODE
MOV AH, 1    ;read first character
REPEAT_:
      INT 21H
      CMP AL, ‘ ‘
      JNE REPEAT_
MOV AH, 4Ch
INT 21H
```

Note: Requires only one Conditional Jump at the end

Also, If terminating condition is false, still loop is executed at least once.

For Practice

- Example given in section 6.5
- Ch 6 Exercise: Q1, Q2, Q4