# Flow Control in Assembly Language

Abdul Haseeb
MS(DKE), Sukkur IBA University

# Overview

- **Jump:**
  - Signed
  - Unsigned
- **Branching:**
  - IF
  - IF-ELSE
  - CASE
- **Looping:**
  - FOR
  - WHILE
  - REPEAT-UNTIL

- Flow control can be:
  - Conditional
  - Non-Conditional

# Write the following code

```
.MODEL          SMALL
.STACK          100H
.CODE
MAIN            PROC
        MOV     AH,2
        MOV     CX,256
        MOV     DL,0
PRINT_LOOP:

        INT     21h             ;display a char
        INC     DL              ;increment ASCII code
        DEC     CX              ;decrement counter
        JNZ     PRINT_LOOP      ;keep going if CX not 0
;DOS exit
        MOV     AH,4CH
        INT     21h
MAIN    ENDP
        END     MAIN
```

# Conditional Jump

- JNZ is one of the examples
- Syntax:

```
Jxxx                    destination_label
```

# JNZ

- If the condition is true:
  - Instructions present in the label are executed
  - Labeled block may precede or follow Jump statement
- If the condition is false:
  - Instructions after the Jump statement are executed

# How CPU Implements a Jump

- CPU Looks in flag registers:
  - If the flag is true for Jump:
    - Set IP to Label
  - If the flag is false:
    - IP remains unaffected and continues to next instruction

**Table 6.1 Conditional Jumps**

***Signed Jumps***

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

## *Unsigned Conditional Jumps*

| *Symbol* | *Description* | *Condition for Jumps* |
|---|---|---|
| JA/JNBE | jump if above<br>jump if not below or equal | CF = 0 and ZF = 0 |
| JAE/JNB | jump if above or equal<br>jump if not below | CF = 0 |
| JB/JNAE | jump if below<br>jump if not above or equal | CF = 1 |
| JBE/JNA | jump if equal<br>jump if not above | CF = 1 or ZF = 1 |

## 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

- Jump condition is provided using CMP
- Computes difference among source and destination, Like SUB
- But unlike SUB, destination remains Unaffected

```
CMP destination, source
```

# Example

- Control goes to BELOW
- Because: ZF=0, SF=OF=0

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

where AX = 7FFFh, and BX = 0001.

7FFEh.

# As a programmer you can ignore, thinking about the flags

- Jump if AX is greater than BX

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

# Instructions

- If you are asked about signed interpretation, use signed jumps and vice versa

# JMP Instruction

- JMP(jump) causes unconditional transfer of control
- Syntax is:
  - JMP Destination

```
TOP:

;body of the loop
DEC   CX                ;decrement counter
JNZ   TOP               ;keep looping if CX > 0
MOV   AX,BX
```

# Perform the Following

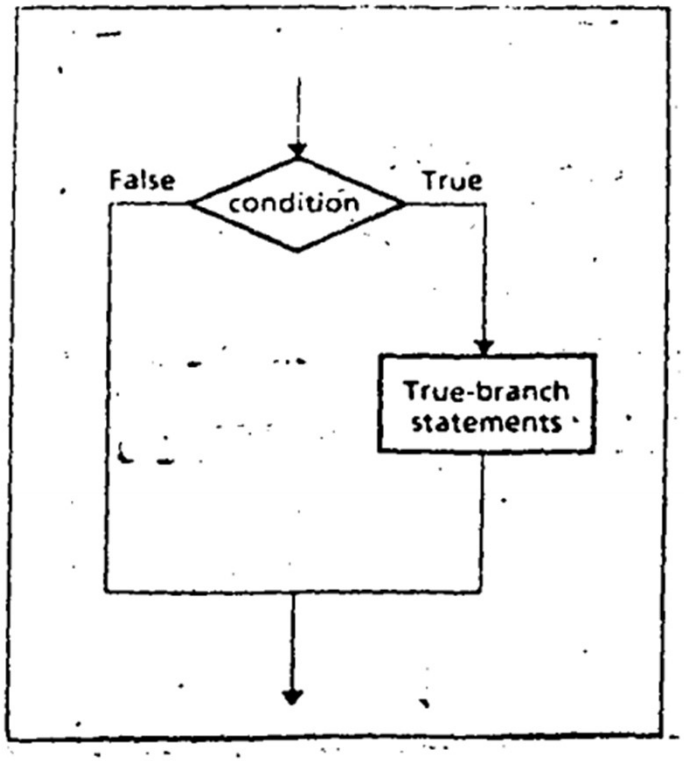
- Take Two Numbers as Input
- Display that number which is largest amongst the Two Numbers

# Branching Structure

- Allow programmer to take different decision based on evaluated condition

```
IF condition is true
 THEN
  execute true-branch statements
END_IF
```

# Branching Structure



# Branching Structure

**Example 6.2** Replace the number in AX by its absolute value.

**Solution:** A pseudocode algorithm is

```
IF AX < 0
 THEN
  replace AX by -AX
END_IF
```

# Branching Structure

It can be coded as follows:

```
;if AX < 0
                CMP   AX,0        ;AX < 0 ?
                JNL   END_IF      ;no, exit
;then
                NEG   AX          ;yes, change sign
END_IF:
```

# Program

- Check if AX is equal to 5, If yes then display the value

# If-else

***IF-THEN-ELSE***

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

# If-else

**Example 6.3** Suppose AL and BL contain extended ASCII characters. Display the one that comes first in the character sequence.

**Solution:**

```
IF AL <= BL
 THEN
  display the character in AL
 ELSE
  display the character in BL
END_IF
```

# If-else

It can be coded like this:

```
                MOV  AH,2       ;prepare to display
;if AL <= BL
                CMP  AL,BL      ;AL <= BL?
                JNBE  ELSE_     ;no, display char in BL
;then                           ;AL <= BL
                MOV DL,AL       ;move char to be displayed
                JMP DISPLAY     ;go to display
ELSE_:                          ;BL < AL
                MOV DL,BL

DISPLAY:
                INT 21h         ;display it
END_IF
```

# Program

- Check the value of AX, Register, if it is greater than 5:
  - Add 10 to the value and display result
  - Otherwise Increment the value by 1 and display the result

# CASE

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

# CASE

**Example 6.4** 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.

# CASE

**Solution:**

```
CASE AX
    <0: put -1 in BX
    =0: put 0 in BX
    >0: put 1 in BX
END_CASE
```

# CASE

**It can be coded as follows:**

```
;case AX
                CMP AX,0        ;test ax
                JL  NEGATIVE    ;AX < 0
                JE  ZERO        ;AX = 0
                JG  POSITIVE    ;AX > 0
NEGATIVE:
                MOV  BX,-1      ;put -1 in BX
                JMP  END_CASE   ;and exit
ZERO:
                MOV  BX,0       ;put 0 in BX
                JMP  END_CASE   ;and exit
POSITIVE:
                MOV  BX,1       ;put 1 in BX
END_CASE:
```

**Example 6.5** If AL contains 1 or 3, display "o"; if AL contains 2 or 4, display "e".

**Solution:**

```
CASE AL
   1,3: display 'o'
   2,4: display 'e'
END_CASE
```

The code is

```
;case AL
; 1,3:
              CMP  AL,1       ;AL = 1?
              JE   ODD        ;yes, display 'o'
              CMP  AL,3       ;AL = '3'
              JE   ODD        ;yes, display 'o'
; 2,4:
```

```
                JE    CDD         ;yes, display 'o'
; 2,4:
                CMP   AL,2        ;AL = 2?
                JE    EVEN        ;yes, display 'e'
                CMP   AL,4        ;AL = 4?
                JE    EVEN        ;yes, display 'e'
                JMP   END_CASE    ;not 1..4
CDD:                              ;display 'o'
                MOV   DL,'o'      ;get 'o'
                JMP   DISPLAY     ;go to display
EVEN:                             ;display 'e'
                MOV   DL,'e'      ;get 'e'
```

```
DISPLAY:

                MOV   AH,2
                INT   21H       ;display char
END_CASE:
```

# Task:

- Write down the code of case statements to check whether character in BX register is a vowel or consonant

# Branching with Compound Conditions

```
condition_1 AND condition_2
```

or

```
condition_1 OR condition_2
```

# Read a character and display it if it is between A-Z is a capital letter

```
;read a character
                MOV   AH,1        ;prepare to read
                INT   21H         ;char in AL
;if ('A' <= char) and (char <= 'Z')
                CMP   AL,'A'      ;char >= 'A'?
                JNGE  END_IF      ;no, exit
                CMP   AL,'Z'      ;char <= 'Z'?
                JNLE  END_IF      ;no, exit
;then display char
                MOV   DL,AL       ;get char
                MOV   AH,2        ;prepare to display
                INT   21H         ;display char
END_IF:
```

# Task

- Display a Number if it is between 1 and 7, otherwise display a message: out of range

# OR Condition

- Read a character if it is y or Y and display it, otherwise terminate the program

**Solution:**

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

```
;read a character
                MOV   AH,1          ;prepare to read
                INT   21H           ;char in AL
;if (character = 'y') or (character = 'Y')
                CMP   AL,'y'        ;char = 'y'?
                JE    THEN          ;yes, go to display it
                CMP   AL,'Y'        ;char = 'Y'?
                JE    THEN          ;yes, go to display it
                JMP   ELSE_         ;no, terminate
THEN:
                MOV   AH,2          ;prepare to display
                MOV   DL,AL         ;get char
                INT   21H           ;display it
                JMP   END_IF        ;and exit
ELSE_:
                MOV   AH,4CH
                INT   21H           ;DOS exit
END_IF:
```

# Task

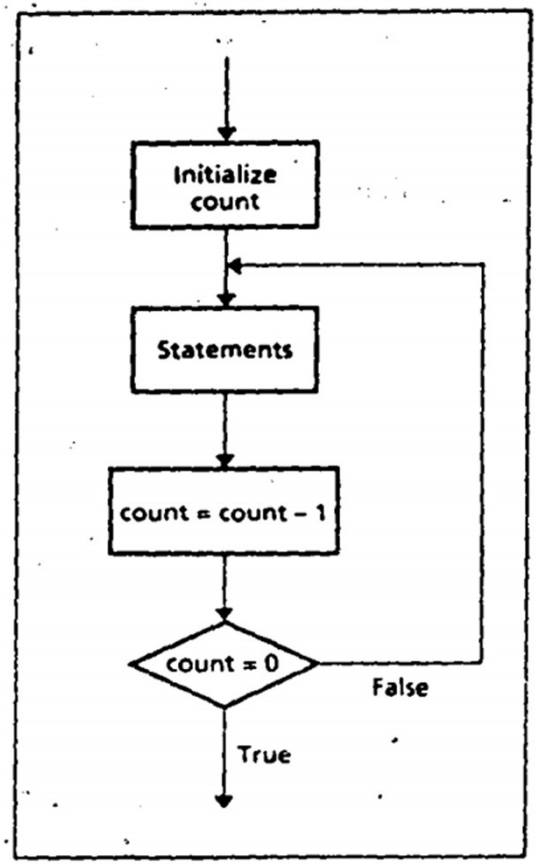
- Take user gender as input (M or F) ,make sure it’s a capital letter
- Take user age as input (between 1-9)
- If Gender is male or age is above 5, then Welcome should be displayed
- Otherwise Good Bye Should be displayed on screen

# Loop

- Sequence of Instructions which repeat:
  - Known Number of Times
  - Depend on conditions

# For Loop

- Counter Loop
- Number of repetitions are known in advance



# For Loop

- **LOOP** instruction:
  - Used to Implement for Loop
  - Syntax:
    - **LOOP** destination label
    - Destination label contains statements to be repeated
- Counter for the loop is CX Register
  - It is decremented after each iteration
  - Control transfers to destination label until CX becomes zero

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

# Executes at least once

**Solution:**

```
FOR 80 times DO
                display '*'
END_FOR
```

The code is

```
                MOV  CX,80      ;number of stars to display
                MOV  AH,2       ;display character function
                MOV  DL,'*'     ;character to display
TOP:
                INT  21h        ;display a star
                LOOP TOP        ;repeat 80 times
```

# Task

- Display first 6 Natural Numbers Using for loop

# What if CX is already zero?

Explore the answer, and JCXZ jump

# While Loop

- This Loop depends on the condition

**Example 6.9** Write some code to count the number of chara ters in : n input line.

**Solution:**

```
initialize count to 0
read a character
WHILE character <> carriage_return DO
 count = count + 1
 read a character
END_WHILE
```

The code is

```
                MOV   DX,0        ;DX counts characters
                MOV   AH,1        ;prepare to read
                INT   21H         ;character in AL
WHILE_:
                CMP   AL,0DH      ;CR?
                JE    END_WHILE   ;yes, exit
                INC   DX          ;not CR, increment count
                INT   21H         ;read a character
                JMP   WHILE_      ;loop back
END_WHILE:
```

# Task

- Write a while loop to take continuous input from user, until user presses 0

# Repeat Until Loop

**Example 6.10** Write some code to read characters until a blank is read.

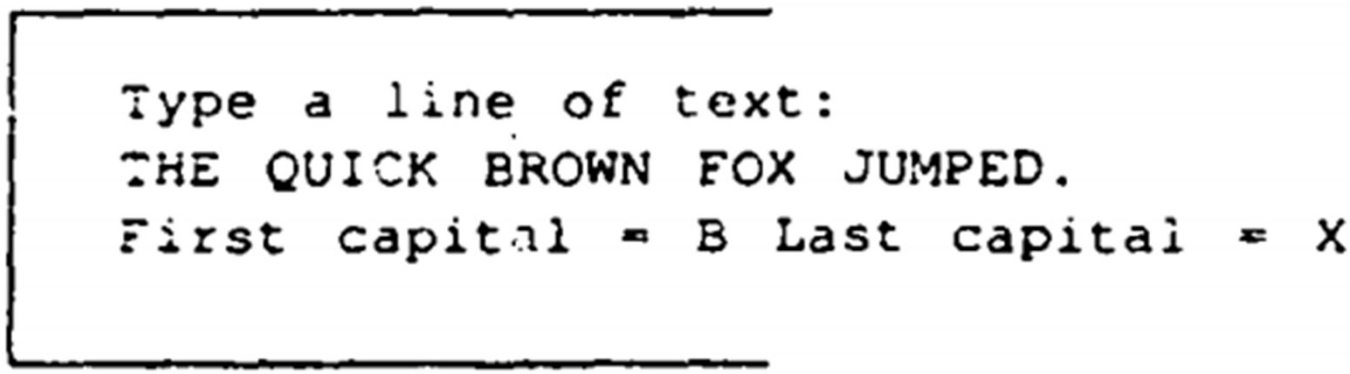
**Solution:**

```
REPEAT
 read a character
UNTIL character is a blank
```

The code is

```
                MOV   AH,1          ;prepare to read
REPEAT:
                INT   21H           ;char in AL
;until
                CMP   AL,' '        ;a blank?
                JNE   REPEAT        ;no, keep reading
```

# Lets solve a problem



# Possible solution

- Display the message
- Process the line of text
- Display the results

# Possible Solution

```
Read a character
WHILE character is not a carriage return DO
IF character is a capital letter (*)
 THEN
  IF character precedes first capital
   THEN
    first capital = character
  END_IF
  IF character follows last capital
   THEN
    last capital = character
   END_IF
END_IF
Read a character
END_WHILE
```

# Possible Solution

```
IF no capitals were typed,
 THEN
  display "No capitals"
 ELSE
  display first capital and last capital
END_IF
```

```
MOV   AH,9              ;display string function
LEA   DX,PROMPT         ;get opening message
INT   21H               ;display it
```

The message will be stored in the data segment as

```
PROMPT   DB       'Type a line of text:',0DH,0AH,'$'
```

Variables FIRST and LAST must have values before the WHILE loop is executed the first time. They can be initialized in the data segment as follows:

```
FIRST           DB              ']'
LAST            DB              '@'
```

The initial values "]" and "@" were chosen because "]" follows "Z" in the ASCII sequence, and "@" precedes "A". Thus the first capital entered will replace both of these values

```
                MOV  AH,1       ;read char function
                INT  21H        ;char in AL
WHILE_:
;while character is not a carriage return do
                CMP  AL,0DH     ;CR?
                JE   END_WHILE  ;yes, exit
;if character is a capital letter
                CMP  AL,'A'     ;char >= 'A'?
                JNGE END_IF     ;not a capital letter
                CMP  AL,'Z'     ;char <= 'Z'?
                JNLE END_IF     ;not a capital letter
;then
; if character precedes first capital
                CMP  AL,FIRST   ;char < FIRST?
                JNL CHECK_LAST  ;no, >=
;then first capital = character
                MOV  FIRST,AL   ;FIRST = char
; end_if
CHECK_LAST:
```

```
;if character follows last capital
                CMP   AL,LAST    ;char > LAST?
                JNG   END_IF     ;no, <=
;then last capital = character
                MOV   LAST,AL    ;LAST = char
; end_if
END_IF:
;read a character
                INT   21H        ;char in AL
                JMP   WHILE_     ;repeat loop
END_WHILE:
```

# Also Initialized in Data Segment

```
NOCAP_MSG       DB          'No capitals $'
CAP_MSG         DB          'First capital = '
FIRST           DB          ']'
                DB          ' Last capital = '
LAST            DB          '@ $'
```

```
                MOV   AH,9        ;display string function
;if no capitals were typed
                CMP   FIRST,']'   ;FIRST = ']'?
                JNE   CAPS        ;no, display results
;then
                LEA   DX,NOCAP_MSG
                JMP   DISPLAY
CAPS:
                LEA   DX,CAP_MSG
DISPLAY:
                INT   21H         ;display message
;end_if
```