CSE 331L / EEE 332L

Microprocessor Interfacing & Embedded System

Section: 5 & 6, Summer 2020

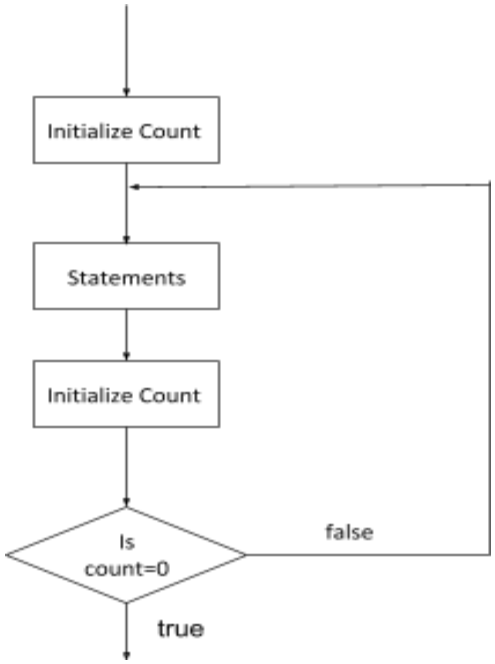
Lab- 04: Flow-Control Instructions

Example 1: Write a count-controlled loop to print a row of 80 '#'s

```
MOV DL, '#'
MOV AH, 2
MOV CX, 80

_LOOP:
INT 21H

LOOP _LOOP
```



Example2: Write a program to take 8 single-key inputs

```
MOV AH, 1
MOV CX, 8

INPUT:
INT 21H

LOOP INPUT
```

Example3: write a program to print the first five digits (0,1,2,3,4)

```
MOV AH, 2
MOV CX, 5
MOV DL, 30H

_LOOP:
INT 21H
INC DL

LOOP _LOOP
```

Example 4: Declare an array of size 10 without any initial data. Prompt the user to enter a line of text and store it into the array.

```
01
02 .MODEL SMALL
03 .STACK 100H
04 .DATA
05     MSG DB "Enter a text$"
06     ARRAY DB 10 DUP(?), '$'
07     NEWL DB 0AH, 0DH, '$'
08 .CODE
09     MOV AX, @DATA
10     MOV DS, AX
11
12     MOV CX, 10
13     MOV AH, 1
14     LEA SI, ARRAY
15
16     _IN:
17     INT 21H
18     MOV [SI], AL
19     INC SI
20
21     LOOP _IN
22
23     MOV AH, 9
24     LEA DX, NEWL
25     INT 21H
26     LEA DX, ARRAY
27     INT 21H
28
29     MOV AH, 4CH
30     INT 21H
31
```

Instructions:

Instruction	**Algorithm (= is assignment)**
XCHG	XCHG Destination, Source **Algorithm:** exchanges the contents between two registers, or a register and a memory location.
NEG	NEG Destination **Algorithm:** Destination = 2's complement of destination. Destination: register or memory location only.
CMP	CMP Destination, Source **Algorithm:** Computes destination-source to compare the operands and thus the flags are affected.

MUL (unsigned multiplication)	MUL Source (register/memory loc) **Algorithm (byte):** AX = AL x Source **Algorithm (word):** DX:AX = AX x Source (register/memory loc)
IMUL (unsigned multiplication)	IMUL Source (register/memory loc) **Algorithm (byte):** AX = AL x Source **Algorithm (word):** DX:AX = AX x Source
DIV (unsigned multiplication)	DIV divisor (register/memory loc) **Algorithm (byte):** AL (quotient) = AX / divisor AH (remainder) = AX % divisor **Algorithm (word):** AX (quotient) = (DX:AX) / divisor DX (remainder) = (DX:AX) % divisor
IDIV (unsigned multiplication)	IDIV divisor (register/memory loc) **Algorithm (byte):** AL (quotient) = AX / divisor AH (remainder) = AX % divisor **Algorithm (word):** AX (quotient) = (DX:AX) / divisor DX (remainder) = (DX:AX) % divisor

Logic Instructions

a	**b**	**a AND b**	**a OR b**	**a XOR b**	**NOT a**	**Instructions**
0	0	0	1	0	1	Opcode destination, source
0	1	0	1	1	1	
1	0	0	1	1	0	
1	1	1	0	0	0	

Example 5: Factorial of 5

```
.model small
.stack 100h
.data
    n db 5
.code
    mov ax, @data
    mov ds, ax

    xor cx, cx
    mov cl, n
    mov al, 1   ;holds the product

    factorial:
    mul cx

    loop factorial

    mov dx, ax
    mov ah, 2
    int 21h

    mov ah, 4ch
    int 21h
```

Example 6: division

```
.model small
.stack 100h
.data
    n db 4
.code
    mov ax, @data
    mov ds, ax

    mov ax, 25

    div n

    mov dl, al  ;quotient in dl
    mov dh, ah  ;remainder in dh

    mov ah, 2
    add dl, 30h
    int 21h     ;display quotient

    mov dl, 20h
    int 21h     ;print space

    mov dl, dh
    add dl, 30h
    int 21h     ;display remainder

    mov ah, 4ch
    int 21h
```