

LAB#06

LOOP

Objective

To understand the concepts of loop in assembly language

Theory

Loop

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 i.e. it's a count controlled loop.

FOR Loop

This is a loop structure in which the loop statements are repeated a known number of times.

Keyword: LOOP

A FOR loop is implemented using the LOOP instruction. The counter for the loop is the CX register, which is initialized to loop count, which is the number of times the loop is executed. Execution of the LOOP instruction causes CX to be decremented automatically. If CX becomes 0, the next instruction after loop is done.

Sample Program:

SOURCE CODE:

Object: Title a program that prints a character 100 times.

```
.model small  
.stack 100h  
.data  
.code
```

```
mov ah, 02h ;display a character  
mov cx, 100 ;number of times loop will execute  
mov dl, '*' ;ASCII code of character 0  
print: ;loop starts from here  
int 21h  
loop print ;executes the FOR loop
```

```
.exit  
endL
```

Lab Task:**Task#01:** Write a program to print ASCII characters.

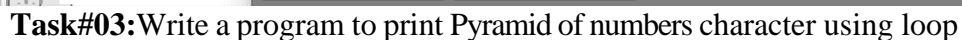
```
01 | model small
02 | .stack 100h
03 | .data
04 | .code
05 | main proc
06 |
07 |     mov cx,255
08 |     mov dx,00
09 |
10 |     L1:
11 |
12 |     mov ah,2
13 |     int 21h
14 |
15 |     inc dx
16 |
17 |     Loop L1
18 |
19 |     mov dl,10
20 |     mov ah,2
21 |     int 21h
22 |
23 |
24 |
25 |     mov dl,13
26 |     mov ah,2
27 |     int 21h
28 |
29 |     mov cx,255
30 |     mov dx,00
31 |
32 |     L2:
33 |
34 |     mov ah,2
35 |     int 21h
36 |
37 |     dec dx
38 |
39 |     Loop L2
40 |
41 |     mov ah,4ch
42 |     int 21h
43 |
44 |
45 |
46 |     main endp
47 | end main
48 | ret
49 |
50 | ret
51 |
```

The screenshot displays an x86 emulator interface with the following components:

- Registers Panel:** Shows the state of various registers. The Program Counter (PC) is at address 0004:0204. The Instruction Pointer (IP) is 0204. The Stack Pointer (SP) is 00FA. The Base Pointer (BP) is 0000. The Data Segment (DS) is 0700. The Extra Segment (ES) is 0700.
- Memory Panel:** Displays memory contents starting at address F400:0204. The memory contains a BIOS interrupt vector table for INT 021h, starting with the instruction IREI.
- Assembly Code Panel:** Shows the assembly code being executed. The code includes instructions like `int 21h`, `inc dx`, `Loop L1`, `mov dl,1`, `mov ah,2`, `mov cx,2`, `mov dx,0`, `L2:`, `mov ah,2`, `int 21h`, `dec dx`, `Loop L2`, `mov ah,4`, and `int 21h`.
- Emulator Screen:** A window titled "emulator screen (83x25 chars)" showing a black screen with some text and symbols.
- Controls:** Includes buttons for "Load", "reload", "step back", "single step", "run", "screen", "source", "reset", "aux", "vars", "debug", "stack", and "flags".

Task#02: Write a program to print A to Z character using loop

```
01 | model small
02 | .stack 100h
03 | .data
04 | .code
05 | main proc
06 |
07 |     mov cx,26
08 |     mov dx,65
09 |
10 |     L1:
11 |
12 |     mov ah,2
13 |     int 21h
14 |
15 |     inc dx
16 |
17 |     Loop L1
18 |
19 |
20 |
21 |
22 |     mov cx,26
23 |     mov dx,90
24 |
25 |     L2:
26 |
27 |     mov ah,2
28 |     int 21h
29 |
30 |     dec dx
31 |
32 |     Loop L2
33 |     mov ah,4ch
34 |     int 21h
35 |
36 |     main endp
37 | end main
38 |
```



```

0001 | model small
0002 | .stack 100h
0003 | .data
0004 | .code
0005 | main proc
0006 |
0007 |     mov cx,5
0008 |     mov dx,48
0009 |
0010 |     L1:
0011 |
0012 |     mov ah,2
0013 |     int 21h
0014 |
0015 |     inc dx
0016 |
0017 |     Loop L1
0018 |
0019 |     mov dl,10
0020 |     mov ah,2
0021 |     int 21h
0022 |
0023 |
0024 |
0025 |     mov dl,13
0026 |     mov ah,2
0027 |     int 21h
0028 |
0029 |     mov cx,4
0030 |     mov dx,48
0031 |
0032 |     L2:
0033 |
0034 |     mov ah,2
0035 |     int 21h
0036 |
0037 |     inc dx
0038 |
0039 |     Loop L2
0040 |
0041 |
0042 |
0043 |     mov dl,10
0044 |     mov ah,2
0045 |
0046 |     int 21h
0047 |     mov dl,13
0048 |     mov ah,2
0049 |     int 21h
0050 |
0051 |     mov cx,3
0052 |     mov dx,48
0053 |
0054 |     L3:
0055 |
0056 |     mov ah,2
0057 |     int 21h
0058 |
0059 |     inc dx
0060 |
0061 |     Loop L3
0062 |
0063 |     mov dl,10
0064 |     mov ah,2
0065 |     int 21h
0066 |
0067 |     mov dl,13
0068 |     mov ah,2
0069 |     int 21h
0070 |
0071 |     mov cx,2
0072 |     mov dx,48
0073 |
0074 |     L4:

```

0075
0076 **mov ah,2**
0077 **int 21h**
0078
0079 **inc dx**
0080
0081 **Loop L4**
0082 **mov dl,10**
0083 **mov ah,2**
0084 **int 21h**
0085
0086 **mov dl,13**
0087 **mov ah,2**
0088 **int 21h**
0089
0090 **mov cx,1**
0091 **mov dx,48**
0092
0093 **L5:**
0094
0095 **mov ah,2**
0096 **int 21h**
0097
0098 **inc dx**
0099
0100 **Loop L5**
0101
0102 **mov ah,4ch**
0103 **int 21h**
0104
0105 **main endp**
0106 **end ret**
0107 **main**

AutoSave Off

emulator: lab 00 part 04.exe

file math debug view external virtual devices virtual drive help

original sour... Load reload step back single step run step delay ms: 0

registers

	H	L
AX	4C	30
BX	00	00
CX	00	00
DX	00	31
CS	F400	
IP	0204	
SS	0710	
SP	00FA	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

F400:0204

F4200:	FF	255	RES
F4201:	FF	255	RES
F4202:	CD	205	=
F4203:	21	033	!
F4204:	CF	207	±
F4205:	00	000	NULL
F4206:	00	000	NULL
F4207:	00	000	NULL
F4208:	00	000	NULL
F4209:	00	000	NULL
F420A:	00	000	NULL
F420B:	00	000	NULL
F420C:	00	000	NULL
F420D:	00	000	NULL
F420E:	00	000	NULL
F420F:	00	000	NULL
F4210:	00	000	NULL
F4211:	00	000	NULL
F4212:	00	000	NULL
F4213:	00	000	NULL
F4214:	00	000	NULL
F4215:	00	000	NULL

BIOS DI
INT 021h
I RET
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
...

screen source reset aux vars debug stack flags

0090 **mov cx,1**
0091 **mov dx,48**

