Computer Organization & Assembly Language

Lab-6

Flag Register Carry parity Auxiliary zero sign trap interrupt direction and overflow flag in Assembly Language-1

What is Flag Register?

The FLAG register is **the status register that contains the current state of a CPU**.

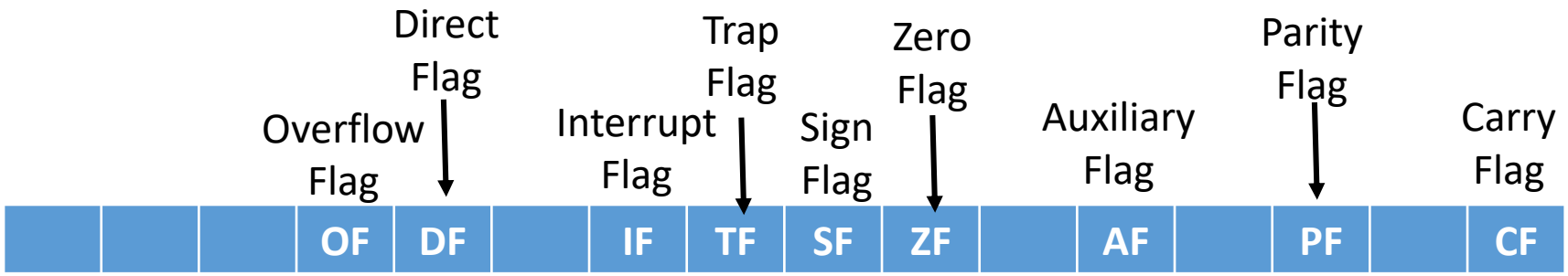
Why do we learn Flag Register?

Theoretically:

- To know how to control the operations of the CPU.
- To know how to handle the status of the operations.

Programmatically:

- To compare for conditional jump.
- To compare the numbers like less than, greater than, equal to etc.



There are always Nine (9) bits are useful

jump, unconditional jump, conditional jump and compare cmp, jmp in Assembly Language-2

Unconditional Jump:

Jump to label without any condition.

```
LabelName:
        Mov dx,'a'
        Mov ah,2
        INT 21h
Jmp LabelName
```

Label Rules:

1. Label can be placed at the beginning of the statement.
2. No reserve word will be assigned as a label such as: (Add, Sub, Mov etc.)
3. Colon (:) must be placed after Label Name while initializing. But not while calling.

jump, unconditional jump, conditional jump and compare cmp, jmp in Assembly Language-3

Conditional Jump:

Jump to label when condition occur.

Syntax:

Opcode Label

LabelName:

Mov ah,1

INT 21h

Mov dl,3

Cmp al,dl

JE LabelName

JE (Jump Equals to)

Cmp reg, reg
Cmp reg, constant
Cmp Reg, (memory address)

Cmp dl,al
Cmp dl,'3'
Cmp dl,($i)

All Comparison Conditions/Operators

JE	Jump Equal to	JZ	Jump Zero
JNE	Jump not equal to	JNZ	Jump Not Zero
JL	Jump less than	JB	Jump if below
JLE	Jump less than equal to	JBE	Jump below equal to
JG	Jump Greater than	JA	Jump if above
JGE	Jump Greater than equal to	JAE	Jump above equal to

Get an integer from user and display whether the number is even or odd in Assembly Language-4

```
.model small
.stack 100h
.data
ev db 'Even$'
od db 'Odd$'
.code
main proc
mov ax,@data
mov ds,ax
mov ah,1
int 21h
mov bl,2
div bl
cmp ah,0
je IsEven
mov dx,10
mov ah,2
int 21h
mov dx,13
mov ah,2
int 21h
mov dx,offset od
mov ah,9
int 21h
mov ah,4ch
int 21h
IsEven:
mov dx,10
mov ah,2
int 21h
mov dx,13
mov ah,2
int 21h
mov dx,offset ev
mov ah,9
int 21h
mov ah,4ch
int 21h
main endp
end main
```

DosBox Commands

- Edit Filename.asm (to create new file if not exists/open existing file)
- MASM Filename.asm; (to convert into object file using MASM assembler)
- LINK Filename.obj; (to convert object file into execution file using linker)
- To execute the exe file you just created,
  - Filename.exe (it will execute)

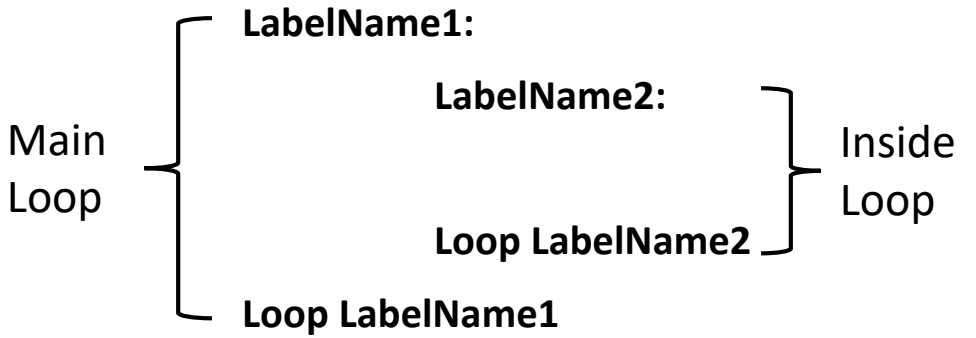
- NOTE: (Semicolon is mandatory while converting via assembler and linker only)

Nested Loops in Assembly Language-6

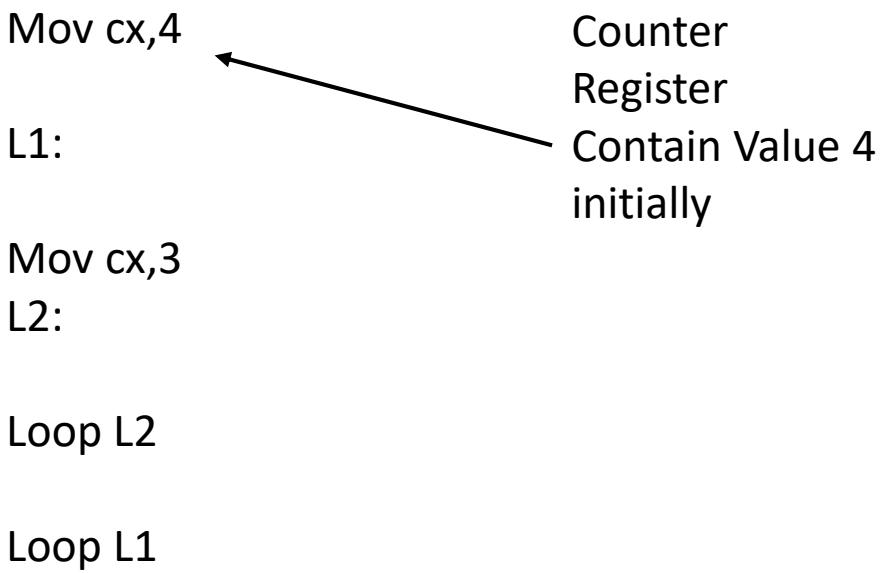
Loop within loop is called nested loop.

With the help of PUSH and POP we can use nested loop in assembly language

Syntax:



Example

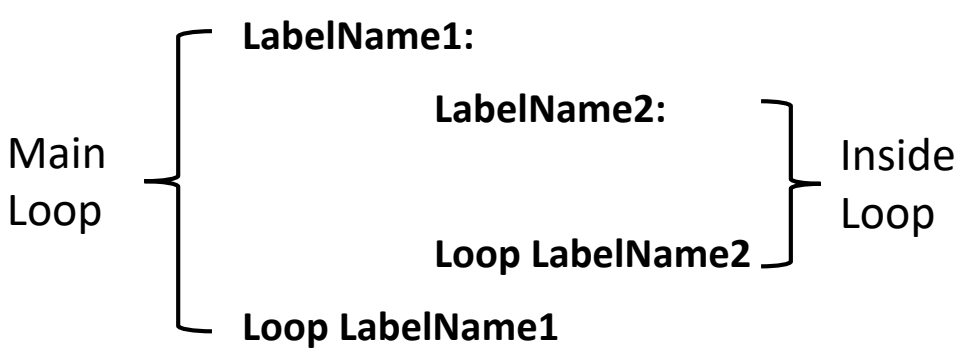


Nested Loops in Assembly Language-7

Loop within loop is called nested loop.

With the help of PUSH and POP we can use nested loop in assembly language

Syntax:



Example

```
Mov cx,4

L1:
PUSH CX

Mov cx,3
L2:

Loop L2

POP CX
Loop L1
```

The PUSH will save the initial value in the stack

The POP will set the last value of CX again.

Program to print the following pattern in Assembly Language-8

```
dosseg
.model small
.stack 100h
.data
.code
main proc
mov ax,@data
mov ds,ax
mov bx, 1
mov cx, 5
L1:
push cx
mov cx, bx
L2:
Mov dl, '*'
mov ah,2
int 21h
loop L2
mov dl,10
mov ah, 2
int 21h
mov dl,13
mov ah, 2
int 21h
inc bl
pop cx
loop L1
mov ah,4ch
int 21h
main endp
end main
```

```
*
**
***
****
*****
```

DosBox Commands

- Edit Filename.asm (to create new file if not exists/open existing file)
- MASM Filename.asm; (to convert into object file using MASM assembler)
- LINK Filename.obj; (to convert object file into execution file using linker)
- To execute the exe file you just created,
  - Filename.exe (it will execute)

- NOTE: (Semicolon is mandatory while converting via assembler and linker only)