Lab Session 03

Introduction to Assembly Language Programming

Flow control Instructions

Jump Instructions:
The jump instructions are used to transfer the flow of the process to the indicated operator.
Syntax….

```
Jxxx        destination_label
```

Compare instruction:
The compare instruction is used to compare two numbers. At most one of these numbers may reside in memory. The compare instruction subtracts its source operand from its destination operand and sets the value of the status flags according to the subtraction result. The result of the subtraction is not stored anywhere. Syntax…..

```
CMP         destination      Source
Example:    CMP         AX,BX
            JG          Bellow
```

#Program1: Write a program that will print all the Characters.

```
.modelsmall
.stack 100h
.data
.code
main proc

        mov ah,2
        mov cx, 256             ;no. of character to display
        mov dl,0                ;dl has ASCII code of null character
```

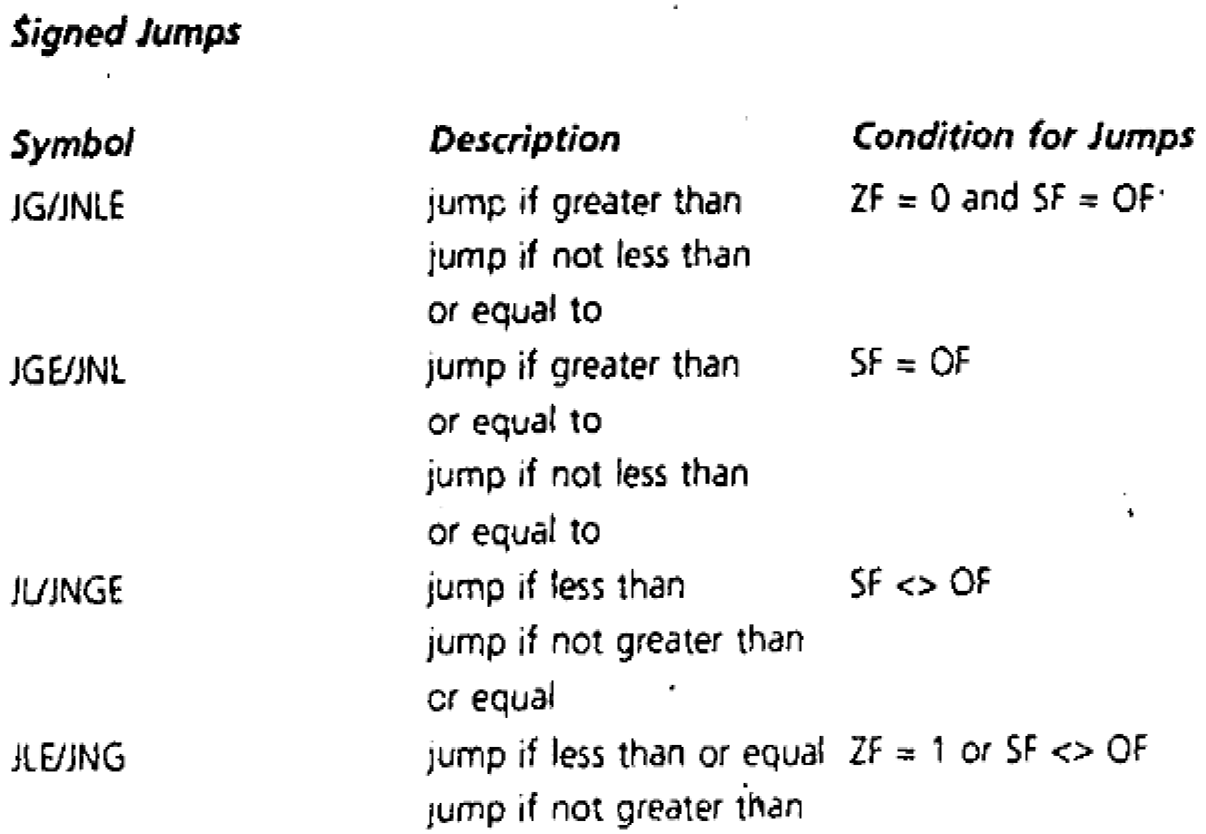
```
print_loop:
        int 21h
        inc dl                  ;increment ASCII code
        dec cx                  ;Decrement counter
        jnz print_loop          ; keep doing if CX not 0
:DOS exit
        mov ah,4ch
        int 21h
main endp
end main
```

#Excercise1: Write an assembly code to display a row of 80 stars.

#Program2: Write an assembly code for Printing ‘*’ depending on input number

```
.modelsmall
.stack 100h
.data
.code
main proc

        mov ah,1
        int 21h
        mov bl,al

        mov ah,2
        mov cx,bx
        mov dl,’*’

print_star:

        int 21h
        dec cx
        Jnz print_star
; return to DOS
        Mov ah,4ch
        int 21h                 ;DOS exit
Main endp
End main
```

Different Jump Instructions:



Signed Jumps

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

Unsigned Conditional Jumps

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

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

#Program3: Suppose AL and BL contain extended ASCII characters. Display the one that comes first in the character sequence.

```
.modelsmall
.stack 100h
.data
.code
main proc

        mov al,05h
        mov bl, 02h

        mov ah,2            ; prepare to display

        cmp al,bl           ; comparing AL with BL
         jge else           ; jump to else if BL is greater than or equal to AL

        mov dl,al
        jmp display

else:
        mov dl,bl

display:

     int 21h
main endp                   ;example if else
```

Exercise 2: write a Assembly code to find the biggest number between two number.

#Program 4: Write an assembly code that can identify a number is odd or even.

```
.model small
.stack 100h
.data
        msg             db          "Enter a number: $"
        msg1            db           10,13,"The number is ODD $"
        msg2            db           10,13,"The number is EVEN $"

.code
main proc
        mov ax, @data           ; initialize DS
         mov ds,ax

        lea dx,msg              ; printing msg
        mov ah,9
        int 21h

         mov ah,1               ;taking user input
        int 21h
        sub al,30h

        cmp al,1                ;compare number
         je ODD
        cmp al,3
        je ODD

        cmp al,2
        je EVEN
        cmp al,4
        je EVEN

        jmp END_CASE            ; if don't match then terminate
```

```
ODD:
        lea dx,msg1              ;printing msg1
         mov ah,9
        int 21h
EVEN:
         lea dx,msg2              ;printing msg2
         mov ah,9
         int 21h
END_CASE:

        mov ah,4ch
        int 21h
main endp                        ;example switch case
```

#Program5: Write an assembly code that identifies an Upper case letter.

```
.model small
.stack 100h
.data
msg1 db 'give your input: $'
msg2 db  0ah,0dh,'Its an upper case letter $'
msg3 db 0ah,0dh, 'Its not an upper case letter $'
.code
 main proc

        mov ax,@data
        mov ds,ax
        lea dx,msg1
        mov ah,9
        int 21h

        mov ah,1
        int 21h

         cmp al,'A'                 ;print upper case if the character is
         jnge end_if                ; char>='A' && char<='Z'
         cmp al,'Z'
         jnle end_if

        lea dx,msg2
        mov ah,9
        int 21h

        mov dl,al
        mov ah,2
        int 21h

        mov ah, 4ch
        int 21h

end_if:

           lea dx,msg3
           mov ah,9
           int 21h

          mov ah, 4ch
          int 21h

          main endp
          end main                  ;example of AND operation
```

#Practice 6: read a character from the user. If it “R” or “r” then display it. Otherwise terminate the program.

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

        msg1        db      “give your input: $”
        msg2        db      0dh,0ah,"the input is matched $"
        msg3        db      0dh,0ah, "sorry... next time $"

.code
main proc

       mov ax,@data
       mov ds,ax

       lea dx,msg1
       mov ah,9
       int 21h

       mov ah,1
       int 21h

       cmp al,'R'            ;comparing if the input is
       je then              ;input==’R’ || input==’r’
       cmp al,'r'
       je then
       jmp else

then:

       lea dx,msg2
       mov ah,9
       int 21h
```

```
         mov dl,al
        mov ah,2
        int 21h

        mov ah, 4ch
        int 21h

  else:

         lea dx,msg3
        mov ah,9
        int 21h

        mov ah, 4ch
        int 21h
 main endp
    end main                        ; OR operation
```

Good Luck ☺