Dave Jhared G. Paduada
BSCpE III - IF

JUMP INSTRUCTION		
CODE	INSTRUCTION	SCREENSHOT

`mov ax,1234h`
`jmp hello`
`mov bx,4567h`
`hello:`
`inc ax`
`mov ah,4ch`
`int 21h`
	JMP
	
	Jmp reg
		1st Jump
		registers
		H L
		AX 12 34
		BX 00 00
		CX 00 00
		DX 00 00
		CS 0100
		IP 0003
		SS 0100
		SP FFFE
		
		6th Jump
		registers
		H L
		AX 4C 35
		BX 00 00
		CX 00 00
		DX 00 00
		CS 0100
		IP 000D
		SS 0100
		SP FFFE
		
		2nd Jump
		registers
		H L
		AX 12 34
		BX 00 00
		CX 00 00
		DX 00 00
		CS 0100
		IP 0008
		SS 0100
		SP FFFE
		
		3rd Jump
		registers
		H L
		AX 12 35
		BX 00 00
		CX 00 00
		DX 00 00
		CS 0100
		IP 0009
		SS 0100
		SP FFFE
		
		4th Jump
		registers
		H L
		AX 4C 35
		BX 00 00
		CX 00 00
		DX 00 00
		CS 0100
		IP 000B
		SS 0100
		SP FFFE
		
		5th Jump
		registers
		H L
		AX 4C 35
		BX 00 00
		CX 00 00
		DX 00 00
		CS F400
		IP 0200
		SS 0100
		SP FFF8

```
.MODEL SMALL
.DATA
STRING1 DB 'ABOVE ', '$'
STRING2 DB 'BELOW ', '$'
.CODE
MOV AX, @DATA
MOV DS,AX


mov ax,1234h
mov bx,4567h
cmp ax,bx
;cmp bx,ax
ja HELLO
jb HI


HELLO:
MOV AH,09
MOV DX,OFFSET STRING1
INT 21H
JMP EXIT

HI:
MOV AH,09
MOV DX,OFFSET STRING2
INT 21H

EXIT:
MOV AH,04CH
INT 21H
```

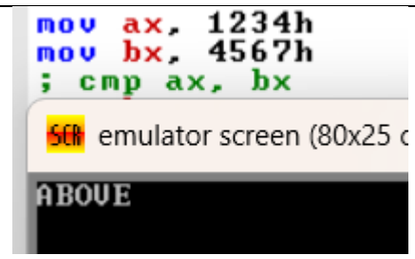
JA AND JB

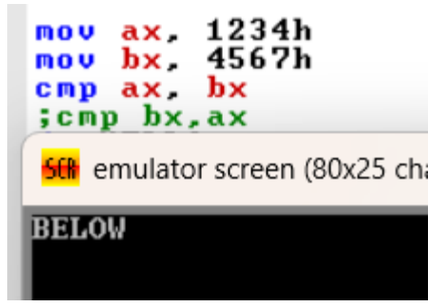
2 swap ax, bx
Bx , ax

Above and below SS

JA – Jump if Above
Prev comparison that the 1st operand is greater than the 2nd operand

JB – Jump if Below
1st Operand < 2nd Operand






```
.MODEL SMALL
.DATA
STRING1 DB 'EQUAL ', '$'
STRING2 DB 'NOT EQUAL ', '$'
.CODE
MOV AX, @DATA
MOV DS,AX


mov ax,1234h
mov bx,1234h
;mov bx,1235
cmp ax,bx
je HELLO
jne HI


HELLO:
MOV AH,09
MOV DX,OFFSET STRING1
INT 21H
JMP EXIT

HI:
MOV AH,09
MOV DX,OFFSET STRING2
INT 21H

EXIT:
MOV AH,04CH
INT 21H
```

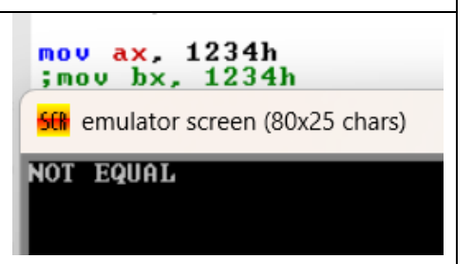
JE /JNE

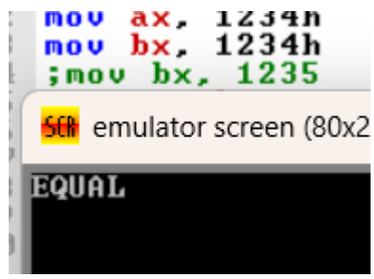
JE – Jump if Equal
If result of ZF 1 hence equal

JNE – Jump if not Equal

If result of ZF 0 hence not equal

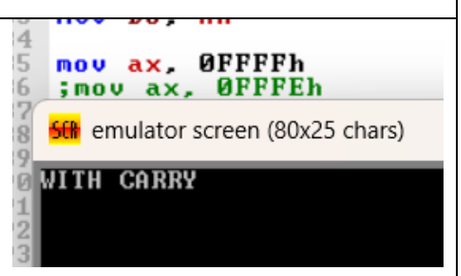
Check the Zero Flag after a comparison or arithmetic operation

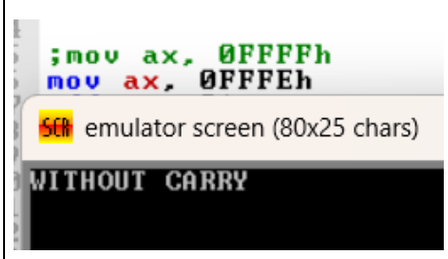





```
.MODEL SMALL
.DATA
STRING1 DB 'WITH CARRY ', '$'
STRING2 DB 'WITHOUT CARRY ', '$'
.CODE
MOV AX, @DATA
MOV DS,AX


mov ax,0FFFFh
;mov ax,0FFFEh
add ax,01
jc HELLO
jnc HI


HELLO:
MOV AH,09
MOV DX,OFFSET STRING1
INT 21H
JMP EXIT

HI:
MOV AH,09
MOV DX,OFFSET STRING2
INT 21H

EXIT:
MOV AH,04CH
INT 21H
```

JC / JNC



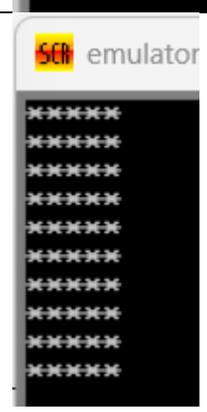


LOOP		
	LOOP	

```
.model small
.data
var1 db 'hello world',13,10,'$'
.code
mov ax,@data
mov ds,ax
mov ah,09
mov cx,20
ulit:
mov dx,offset var1
int 21h
loop ulit
mov ah,4ch
int 21h
```

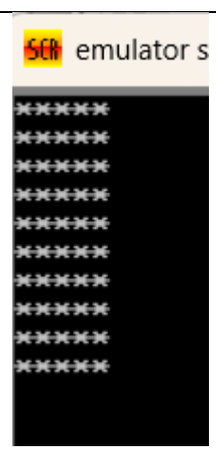
	DOUBLE LOOP	

```
.model small
.data
var1 db '*','$'
var2 db ' ',13,10,'$'
.code
mov ax,@data
mov ds,ax
mov ah,0
mov cl,10
ulit2:
  mov bl,cl
  mov cl,5
ulit:
mov ah,09h
mov dx,offset var1
int 21h
loop ulit
mov ah,09h
mov dx,offset var2
int 21h
  mov cl,bl
loop ulit2
mov ah,4ch
int 21h
```



	DOUBLE LOOP	

```
.model small
.data
var1 db '*','$'
var2 db ' ',13,10,'$'
.code
mov ax,@data
mov ds,ax
mov cx,10

ulit2:
push cx

    mov cl,5
    ulit:
        mov ah,09h
        mov dx,offset var1
        int 21h
    loop ulit

mov ah,09h
mov dx,offset var2
int 21h

pop cx
loop ulit2

mov ah,4ch
int 21h
```



```
.model small
.data
var1 db '*','$'
var2 db '',10,13,'$'
.code
mov ax,@data
mov ds,ax
mov cl,3
mov bl,1h
mov bh,0

ulit2:
     mov bh,cl
     mov cl,bl
ulit:
mov ah,09h
mov dx,offset var1
int 21h
loop ulit

inc bl
mov ah,09h
mov dx,offset var2
int 21h
  mov cl,bh
loop ulit2

mov ah,4ch
int 21h
```

DOUBLE LOOP


```
.model small
.data
var1 db '*','$'
var2 db '',10,13,'$'
.code
mov ax,@data
mov ds,ax
mov cx,3
mov bx,1
ulit2:
     push cx
     push bx
         pop cx
         ulit:
         mov ah,09h
         mov dx,offset var1
         int 21h
         loop ulit
         inc bx

     mov ah,09h
     mov dx,offset var2
     int 21h
     pop cx
loop ulit2

mov ah,4ch
int 21h
```

DOUBLE LOOP W/ PUSH AND POP

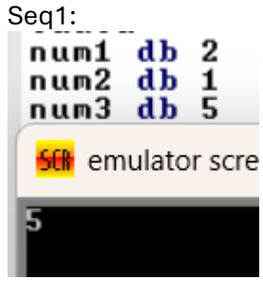
```
.model small
.data
num1 db 2
num2 db 1
num3 db 5
lar db '0',13,10,'$'
.code

start:
mov ax,@data
mov ds,ax
mov al,[num1]    ;save num1 sa al
mov bl,[num2]    ;save num2 sa bl
mov cl,[num3]    ;save num3 sa cl

cmp al,bl        ;compare 1 and 2
jg checkthird    ;jump para compare naman firstandthird
jmp checksecond  ;jump para compare naman sa secondandthird

checksecond:
cmp bl,cl            ;compare 2 and 3
jg secondlargest     ;jump para print second as largest
jmp thirdlargest     ;jump para print thrird as largest

checkthird:
cmp al,cl        ;compare 1 and 3
jg firstlargest  ;jump para print first as largest
jmp thirdlargest ;jump pag mas mali ung third

firstlargest:
mov al,[num1]   ;save sa AL ung laman ng variable na num1
or al,30h       ;use OR para maging ascii
lea bx,lar      ;kukunin address ng variable na LAR at isave sa BX
mov [bx],al     ;isave sa memory location ng LAR ung value ng AL
jmp print


secondlargest:
mov al,[num2]
or al,30h
lea bx,lar
mov [bx],al
jmp print


thirdlargest:
mov al,[num3]
or al,30h
lea bx,lar
mov [bx],al
jmp print


print:
mov ah,09
mov dx,offset lar
int 21h

mov ah,4ch
int 21h
```

Find the output:

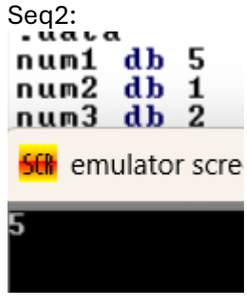
Change the value of num1,num2 and num3

Seq	n1	n2	n3
1	2	1	5
2	5	1	2
3	1	5	2

Seq1:



Seq2:



Seq3: