5th - Oct - 23

Flow Control :-

→ if else

→ loops - Repeat Statements
- Count - For loop
- Condition - while

→ FOR LOOP

→
```
Mov dl, '*'
Mov ah, 2
Mov CX, 5
repeat:
        int 21h
Loop repeat
```

CMP AX, DX

→ Display Numbers from (1) to 9.

→
```
MOV ah, 2
Mov dl, 31h
MOV CX, 9
Repeat:
        int 21h
        ADD dl, 1h   OR   INC dl
loop Repeat
```

→ Put the sum of 1, 2, 3 ... 9 into AX

→
```
MOV cx, 9
Mov AX, 30h
Repeat:
```

When CX=0, ZF=1

```
ADD AX, CX
Loop Repeat
```

Put the Sum of Sequence 50 terms in DX.
1, 5, 9, 13 ---.

```
MOV dx, 0
MOV CX, 50
Mov bx, 1
Repeat
    ADD dx, 4
loop repeat
```

Read a char & display it on the next line 80 times

```
MOV DX, 0AH, 0DH, '$'
MOV CX, 80
MOV AH, 1 ---> MOV DL, AL
Repeat: Mov ah, 2
        int 21h
        Mov ah, 9   ; Print String/Line feed
        int 21h
        loop Repeat
```

Read a five-char password and overwrite it by carrige return & display 5 X's

```
MOV DX, 0AH, 0DH, '$'
MOV CX, 5

Repeat:
    MOV AH, 1          ; Read char
    int 21h
    MOV DI, AL
    int 21h
    Mov Ah, 9          ; Print Line feed
    int 21h
    Mov dl, 'X'  → MOV ah, 2 ; Print char
    int 21h
Loop repeat.

    MOV dh, 0AH
    MOV ah, 2
    int 21h
    MOV dh, 0DH
    int 21h  MOV dl, dh
MOV ah, 2  MOV ah, 2
    int 21h.
```

CMP

→ CMP Destination - Source
Flags are Set
Example

→ CMP AX, BX ∴ AX-BX

→ JE / JZ ⇒ zf=1

↙ ↘

Jump Equal / Jump Zero

→ While_!

→ end_while:

while_!

CMP AX, BX

JE end_while:

if

→ JNE / JNZ ⇒ Zf=0

↙ ↘

Jump Not Equal Jump Not Zero

→ JL (Jump Less) ⇒ SF <> CF
if destination is < or > source
JL is executed

→ JG (Jump Greater) ⇒ zf=0 / SF = CF

→ JGE (Jump Greater Equal) ⇒ SF=OF

→ JLE (Jump Less Equal) ⇒ ZF=1 / SF≠CF

→ JMP (Jump)
→ Jump to Label

→ Display '*' 80 times through (While Loop)

```
MOV AX, 80
MOV BX, 1
Repeat:
    CMP AX, BX
    JE not_repeat
    MOV dl, '*'
    MOV ah, 2
    int 21h
    INC BX
    JMP Repeat
not_repeat:
    MOV ah, 4Ch
    int 21h
```

```
MOV AX, 80
CMP AX, BX
CMP AX, 1
```

Repeat_untill :-

```
MOV dl, '*'
MOV ah, 2
MOV ax, 80
MOV BX, 1
repeat:
    int 21h
    CMP AX, BX
    INC BX
    JE not_repeat
    JMP repeat
not_repeat.
```

JNE repeat

Take input from user untill they press Enter

```
mov ah, 1
mov dh, 0Ah
Repeat: int 21h
    CMP dh, al
    JNE repeat
not_repeat
```

JGE = Jmp Greater Equal

=> If AX < 0 Then
Put -1 IN BX
END IF

CMP AX , 0
JGE END_IF
MOV BX, -1
END_IF:

=> IF AL < 0 Then
put FFh in AH
Else
put 0 in AH
END_IF

CMP AL, 0
JGE else
if:
MOV AH, FFh
JMP END
else:
MOV AH, 0
end.

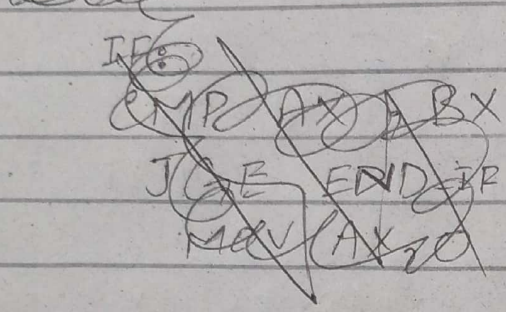
IF AX < BX Then
IF BX < CX Then
put 0 in AX
Else
put 0 in BX
END_IF
END_IF

CMP AX, BX
JGE END_IF
:
CMP BX, CX
JGE Else
MOV AX, 0
JMP END_IF
Else:
MOV BX, 0.
END_IF:

Question

⇒ IF AX < BX Then
put 0 in AX

Else
IF BX < CX Then
put 0 in CX
END-IF



CODE:

```
CMP AX , BX
JGE Else
MOV AX, 0
Else:
CMP BX , CX
JGE END-IF
MOV CX, 0
END-IF
```

~~CMP AX , BX~~

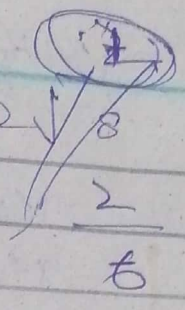
~~MOV AX , 0~~

JGE

BX < CX

AX < BX T
JGE Else
MOV AX, 0
Else:
CMP BX, CX
JGE END-IF
MOV CX, 0
END-IF

divisor = 2
dividend = 8
Q = 0



dividend ≥ divisor
inc Q
dividend = dividend - divisor
END-While

code:-

```
mov al, 2
mov bl, 8
mov cl, 0
Repeat:-
    CMP bl, al
    JL not-repeat
    INC cl
    SUB bl, al
    JMP repeat
Not-repeat
```

Mov dl, cl
Mov ah, 2
int 21h

bl = 0
bl = 0
0 = 0

al bl cl
4 x 2 = 0
0 + 4 = 4
4 + 4 = 8

Multiply :-

```
Mov al, 4
Mov bl, 2
Mov cl, 0
Repeat
    Dec bl
    ADD cl, al
    CMP bl, 0
    JNE repeat
Not-Repeat
Mov dl, cl
Mov ah, 2
int 21h
```

al bl cl
4 x 2 0
x 1 4
0 0 8
$

0 + 4 = 4
4 + 4 = 8

2, 5, ... 30

```
Mov DX, 2
Mov BX, 0
MOV CX, 30
Repeat
  ADD BX, DX
  ADD DX, 3
Loop Repeat
```

19-Oct-2023

=> IF AX < 0 then
replace AX by -AX

```
CMP AX, 0
    JNL end / JGE end
if:
    NEG AX
end:
```

=> if AL < BL then
Display AL
else
Display BL
end.

```
CMP AL, BL
  JGE
  MOV DL, AL
else MOV ah, 2
  MOV DL
```

```
CMP AL, BL
  JGE else
if:
  MOV DL, AL
  MOV ah, 2
  int 21h
  JMP end
else:
  MOV DL, BL
  MOV ah, 2
  int 21h
end.
```

IF AX < BX then
if BX < CX then
put 0 in AX
else
put 0 in BX
endif

```
CMP AX, BX
JNL end.

CMP BX, CX
MOV ax, 0
JNL else
JMP end
else:
MOV BX, 0
JMP end
end.
```

Read a char & display if if it is in uppercase.

if AL >= 'A' AND AL <= 'Z'
Display AL
end

```
MOV ah, 1
int 21h

CMP AL, 'A'
JL end.
CMP AL, 'Z'
JG end
```

```
if:
    MOV dl, Al
    Mov ah, 2
    int 21h
: endi
```

=> if AL == 'Y' OR AL == 'y' then
Display AL
end.

```
MOV ah, 1
int 21h
CMP AL, 'Y'
JE if
CMP AL, 'y'
JE if
JMP end.
if.
    MOV dl, Al
    MOV ah, 2
    int 21h
: end:
```