FAST-NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES PESHAWAR CAMPUS

NAME: Junaid Saeed

ROLL NO: 22P-9200

SECTION: BS(CS)-3D

COURSE: Coal

ASSIGNMENT # 4

SUBMITTED TO: Dr . Usman Abassi

<u>Question # 1 :</u>

<u>→</u>The screenshot and explanation of the code :

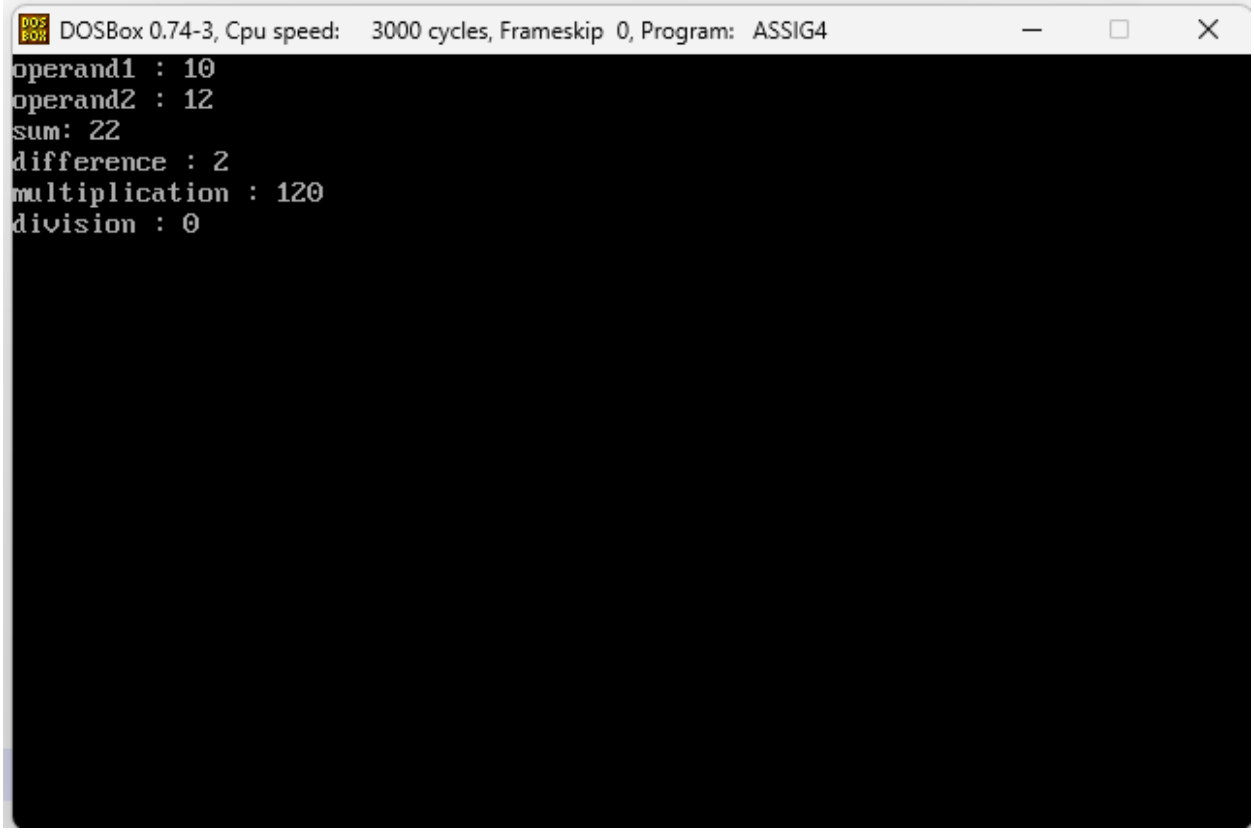
- The code performs arithmetic operations (addition, subtraction, multiplication, division) on predefined operands and displays the results on the screen. It uses stack operations to pass parameters and return values to/from functions. Specific memory locations hold messages and results, and it employs printing routines to display . Arithmetic functions ('sum', 'subtraction', 'mull', 'divv') manipulate operand values and store results. The ('printing') subroutine displays messages and numeric values at specified positions on the screen. ('clrscn') clears the screen by writing spaces to all text mode memory locations. The code concludes by invoking an interrupt to wait for a keypress ('int 0x21') and finally terminates the program ('int 0x4c00').

```
[org 0x0100]
 jmp start
 message1 :db  'operand1 : '
message2 :db  'operand2 : '
m3 :dw 11
message3 :db 'sum: '
m3: dw 5
message4: db 'difference : '
m4 :dw 13
message5: db 'multiplication : '
m5 :dw 17
message6: db 'division : '
m6 :dw 11

operand1: dw 10
operand2: dw 12
sum_result: dw 0
subtraction_result: dw 0
multiplication_result: dw 0
division_result: dw 0
cursor : dw 0

m1 : dw 17
h1 : dw 1565
 divv:
push bp
mov bp ,sp
push ax
push bx

mov ax , [bp+6]
mov bx ,[bp+4]

div bx
mov [division_result], ax
pop bx
pop ax
pop bp
ret



mul:

push bp
mov bp ,sp
push ax
push bx

mov ax , [bp+6]
mov bx ,[bp+4]
mul bx
mov [multiplication_result] , ax
pop bx
pop ax
pop bp


ret

sum:

push bp ;
mov bp,sp ;

mov ax , [bp+6]
mov bx ,[bp+4]
add ax ,bx
mov [sum_result] , ax

pop bp
ret

subtraction:
push bp ;
mov bp,sp ;
push ax
push bx
mov ax , [bp+6]
mov bx , [bp+4]

sub bx , ax
mov [subtraction_result], bx
pop bx
pop ax
pop bp
ret



printing:
push bp
mov bp ,sp
push ax
push bx
push es
push si
push cx
mov ax , 0xb800
mov es , ax
mov di ,  [bp+8]
mov si ,  [bp+6]
mov cx ,  [bp+4]

nextchar2 :
mov al , [si]
mov [es:di] , al
add di,2
add si, 1
loop nextchar2


pop cx
```

```
pop cx
pop si
pop es
pop bx
pop ax
pop bp
ret 6


clrscn:

      mov ax , 0xb800
      mov es , ax
      mov di , 0
      nextchar1:
      mov word [es:di] , 0x0720
      add di , 2
      cmp di,4000
      jne nextchar1
      ret




number:
push bp
mov bp, sp
push ax
push bx
push cx
push dx
mov ax , 0xb800
mov es , ax
mov ax , [bp+6]
mov bx , 10
mov cx , 0


nextprint :

 mov dx, 0
 div bx
 add dl , 0x30
 push dx
 inc cx
 cmp ax , 0
 jnz nextprint
 mov ax , 0xb800
 mov es , ax
 mov di , [bp+4]
 last:

 pop dx
 mov dh ,0x07
 mov [es:di] , dx
 add di,2


 loop  last
 pop dx
 pop cx
 pop bx
 pop ax
 pop bp
 ret 4

 line:

      push bp
      mov bp,sp
      push ax
      push bx
      mov bx,  [bp+4]
      mov cx , 160
      sub cx ,bx
      mov ax , 0xb800
      mov es , ax
      mov di , 0

      nextchar3:
      mov word [es:di] , 0x0720
      add di , 2
      cmp di,cx
      jne nextchar3
      pop bx
      pop ax
      pop bp
      ret 2




start:
   mov ax,[operand1]
  push ax
  mov ax ,[operand2]
  push ax;
  call sum
   call subtraction
  call mul1
  call divv
  call clrscn
```

```
mov ax , 160
mov [cursor],ax
push ax
mov ax ,message2
push ax
push word[m3]
call printing

mov ax , [operand2]
push ax
mov ax , [m3]
mov bx , 2
mul bx
add ax , 160
push ax
call number    ;2


mov ax , 320
push ax
mov ax ,message3
push ax
push word[m3]
call printing

mov ax , [sum_result]
push ax
mov ax , [m3]

mov bx , 2
mul bx
add ax , 320
push ax
call number

mov ax , 480
push ax
mov ax ,message4
push ax
push word[m4]
call printing

mov ax , [subtraction_result]
push ax
mov ax , [m4]

mov bx , 2
mul bx
add ax , 480
push ax
call number

mov ax , 640
push ax
mov ax ,message5
push ax
push word[m5]
call printing

mov ax , [multiplication_result]
push ax
mov ax , [m5]

mov bx , 2
mul bx
add ax , 640
push ax
call number


mov ax , 800
push ax
mov ax ,message6
push ax
push word[m6]
call printing

mov ax , [division_result]
push ax
mov ax , [m6]

mov bx , 2
mul bx
add ax , 800
push ax
call number
mov ah,0x01

int 0x21


mov ax, 0x4c00
int 0x21
```

→Final Output :-



→This is the final output of the code in wich I have displayed the all the given instructions as said in then question.

{ Thanks}