

Question No. 1

➤ Code Screen Shots

```

1  [org 0x0100]
2
3  jmp start
4
5  operand1 : dw 21
6  operand2 : dw 3
7
8  sum_result : dw 0
9  subtraction_result : dw 0
10 multiplication_result : dw 0
11 division_result : dw 0
12
13 topint1 : db 'Operand 1 : '
14 count1 : dw 12
15
16 topint2 : db 'Operand 2 : '
17 count2 : dw 12
18
19 topint3 : db 'Sum : '
20 count3 : dw 6
21
22 topint4 : db 'Difference : '
23 count4 : dw 13
24
25 topint5 : db 'Multiplication : '
26 count5 : dw 17
27
28 topint6 : db 'Division : '
29 count6 : dw 11
30
31 add1:
32
33     push bp
34     mov bp , sp
35
36     mov bx , [bp + 4]
37     mov ax , [bp + 6]
38
39     add ax , bx
40     mov [sum_result] , ax
41
42     mov sp , bp
43     pop bp
44
45     ret
46
47 subtract:
48     push bp
49     mov bp , sp
50
51     mov bx , [bp + 4]
52     mov ax , [bp + 6]
53
54     sub ax , bx
55     mov [subtraction_result] , ax
56
57     mov sp , bp
58     pop bp
59
60     ret
61
62
63 multiply:
64     push bp
65     mov bp , sp
66
67     mov bx , [bp + 4]
68     mov ax , [bp + 6]
69
70     mul bx
71     mov [multiplication_result] , ax
72
73     mov sp , bp
74     pop bp
75
76     ret
77
78
79 divide:
80     push bp
81     mov bp , sp
82
83     mov bx , [bp + 4]
84     mov ax , [bp + 6]
85
86     div bx
87     mov [division_result] , ax
88
89     mov sp , bp
90     pop bp
91
92     ret
93

```

```

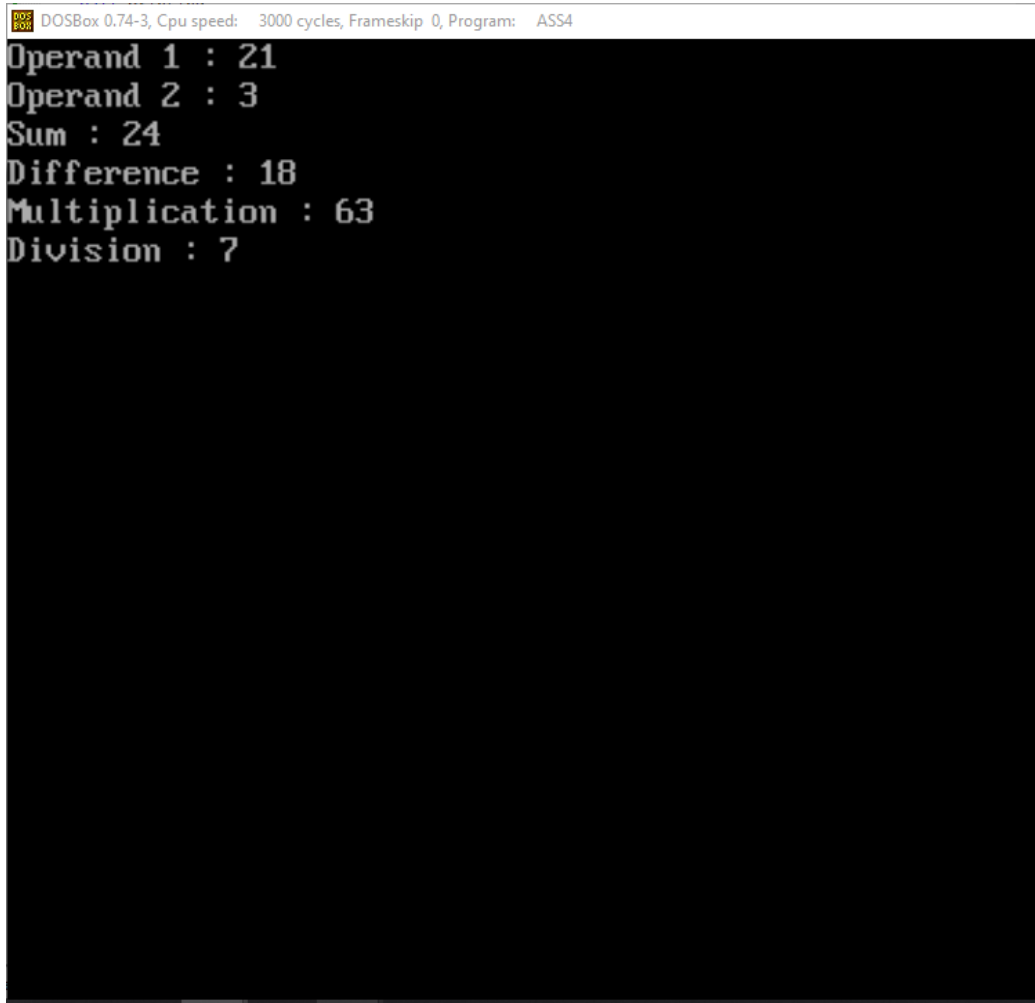
94      clrscreen:
95          mov ax , 0xb800
96          mov es , ax
97          mov bx , 0
98
99      loop1:
100
101          mov word [es : bx] , 0x072
102          add bx , 2
103
104          cmp bx , 4000
105          jne loop1
106
107          ret
108
109      printing:
110          mov ax , 0xb800
111          mov es , ax
112
113          push bp
114          mov bp , sp
115          mov cx , [bp + 4]
116          mov si , [bp + 6]
117          mov di , [bp + 8]
118
119          mov ah , 0x07
120
121
122      printingloop1 :
123          mov al , [si]
124
125          mov word [es : di] , ax
126          add si , 1
127
128          add di , 2
129
130          loop printingloop1
131
132
133          mov sp , bp
134          pop bp
135
136          ;-----
137
138          ret
139
140
141      printing2:
142
143          ;number printing
144          mov ax , 0xb800
145          mov es , ax
146
147          push bp
148          mov bp , sp
149
150          mov ax , [bp + 4]
151          mov bx , [bp + 6]
152          mov di , 10
153          mov cx , 0
154
155
156      printing2loop:
157
158          mov dx , 0
159          div di
160
161          add dl , 0x30
162          push dx
163
164          add cx , 1
165          cmp ax , 0
166
167
168          jnz printing2loop
169
170      printing2loop2:
171
172          pop dx
173          mov dh , 0x07
174
175          mov [es : bx] , dx
176          add bx , 2
177          sub cx , 1
178
179          cmp cx , 0
180          jnz printing2loop2
181
182          mov sp , bp
183          pop bp
184
185          ret 1
186
187

```

```
188      start:
189          push bx
190
191          mov ax, [operand1]
192          push ax
193
194          mov ax, [operand2]
195          push ax
196
197          call add1
198          call subtract
199          call multiply
200          call divide
201
202          call clrscreen
203
204          mov ax, 0 ;where to start printing
205          push ax
206
207          mov ax, toprint1
208          push ax
209
210          mov ax, [count1]
211          push ax
212
213          call printing
214
215          mov ax, [count1]
216          shl ax, 1
217          add ax, 0
218
219          push ax
220
221          mov ax, [operand1]
222          push ax
223
224          call printing2
225
226          mov ax, 160 ;where to start printing
227          push ax
228
229          mov ax, toprint2
230          push ax
231
232          mov ax, [count2]
233          push ax
234
235          call printing
236
237          mov ax, [count2]
238          shl ax, 1
239          shl ax, 1
240          add ax, 160
241
242          push ax
243
244          mov ax, [operand2]
245          push ax
246
247          call printing2
248
249          mov ax, 320 ;where to start printing
250          push ax
251
252          mov ax, toprint3
253          push ax
254
255          mov ax, [count3]
256          push ax
257
258          call printing
259
260          mov ax, [count3]
261          shl ax, 1
262          add ax, 320
263
264          push ax
265
266          mov ax, [sum_result]
267          push ax
268
269          call printing2
270
271          mov ax, 480 ;where to start printing
272          push ax
273
274          mov ax, toprint4
275          push ax
276
277          mov ax, [count4]
278          push ax
279
280          call printing
281
282          mov ax, [count4]
283          shl ax, 1
284          add ax, 480
285
286          push ax
```

```
287     mov ax , [subtraction_result]
288     push ax
289
290     call printing2
291
292
293     mov ax , 640 ;where to start printing
294     push ax
295
296     mov ax , toprint5
297     push ax
298
299     mov ax , [count5]
300     push ax
301
302     call printing
303
304     mov ax , [count5]
305     shl ax , 1
306     add ax , 640
307
308     push ax
309
310     mov ax , [multiplication_result]
311     push ax
312
313     call printing2
314
315     mov ax , 800 ;where to start printing
316     push ax
317
318     mov ax , toprint6
319     push ax
320
321     mov ax , [count6]
322     push ax
323
324     call printing
325
326     mov ax , [count6]
327     shl ax , 1
328     add ax , 800
329
330     push ax
331
332     mov ax , [division_result]
333     push ax
334
335     call printing2
336
337
338     pop ax
339     pop ax
340     pop bx
341
342
343
344     mov ah , 0x01
345     int 0x21
346
347     mov ax , 0x4c00
348     int 0x21
349
```

➤ Dos Box Screen Shot



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: ASS4
Operand 1 : 21
Operand 2 : 3
Sum : 24
Difference : 18
Multiplication : 63
Division : 7
```

➤ Explanation

The logic here is pretty simple but calling it again and again is a bit complicated so what i've done is copy the code from the assignment 3 question 1 that is calculating the sum subtraction multiplication and division of a given 2 operands , and storing that result in desired data labels. Additional here we need the character count of each string that we are printing so i've stored that in the data labels of each string named as count1 , count2 and so on that I'm printing. After which I've created two subroutines named printing and printing2 , printing is printing the string and printing 2 is printing the result . But first of all im calling the clrscr subroutine what it does is clear all the screen of the dos box and it actually not clearing it it is printing spaces on the whole dos box concept of physical address accessing of the video base we start from the first cell to the last cell and keep on printing spaces which actually clears the dos box .

Before calling the printing subroutine I'm passing ax that decides from where to start printing , add of the first string character and count of the string. The Count of the string act as a counter where to stop printing the characters one by one. After that string is printed it returns back and after which i again set the parameters for the second subroutine which is printing2 what it does it prints the number or you can call the result of sum , subtract and so on but

before calling it I set the parameters that is from where to start , this was one of the main problems faced as it was overwriting the string printed . So what I did was i didn't change the value of bx which was used in the printing subroutine for accessing the cells and like this the number is printed just in front of the string . The printing2 subroutine logic is different from the printing subroutine as it is printing the number. And we cant print the number directly we have to break it and push it into stack and then pop and print it. So we have to divide it by 10 . So what is happening is that take a number like 23 , 23 will be divided by 10 the remainder will be stored in dx which is 3 after which we will add 0x30 with dl which will convert it into ascii and push this value into the stack (as the number is broken down oppositely but we have to print it as 2 , 3 so stack is helpful to invert this and do our work). We will continue this process until the number has become 0 . After which we will start popping dx and printing the values this resembles printing the string somehow. But keep in mind after popping everytime we will move 0x07 into dh for the characteristics of the value to be printed.

Like this we have printed the first line now to print the second and other lines similarly we add 160 with previous line first cell in ax that we pass as parameters so that the second string and value is printed on the second line and similarly the address of the first string character and the count of the characters in the string will be sent as parameters rest the logic will remain the same.

In the same way printing on the next line we will add 160 with the previous line which is 160 and then push it to start printing from that specific location.

But keep in mind while printing the value what we do is to access that specific location we add the count of the character in the string multiply by 2 (which im doing by shifting left) and the line number for example (mov ax , [count2]

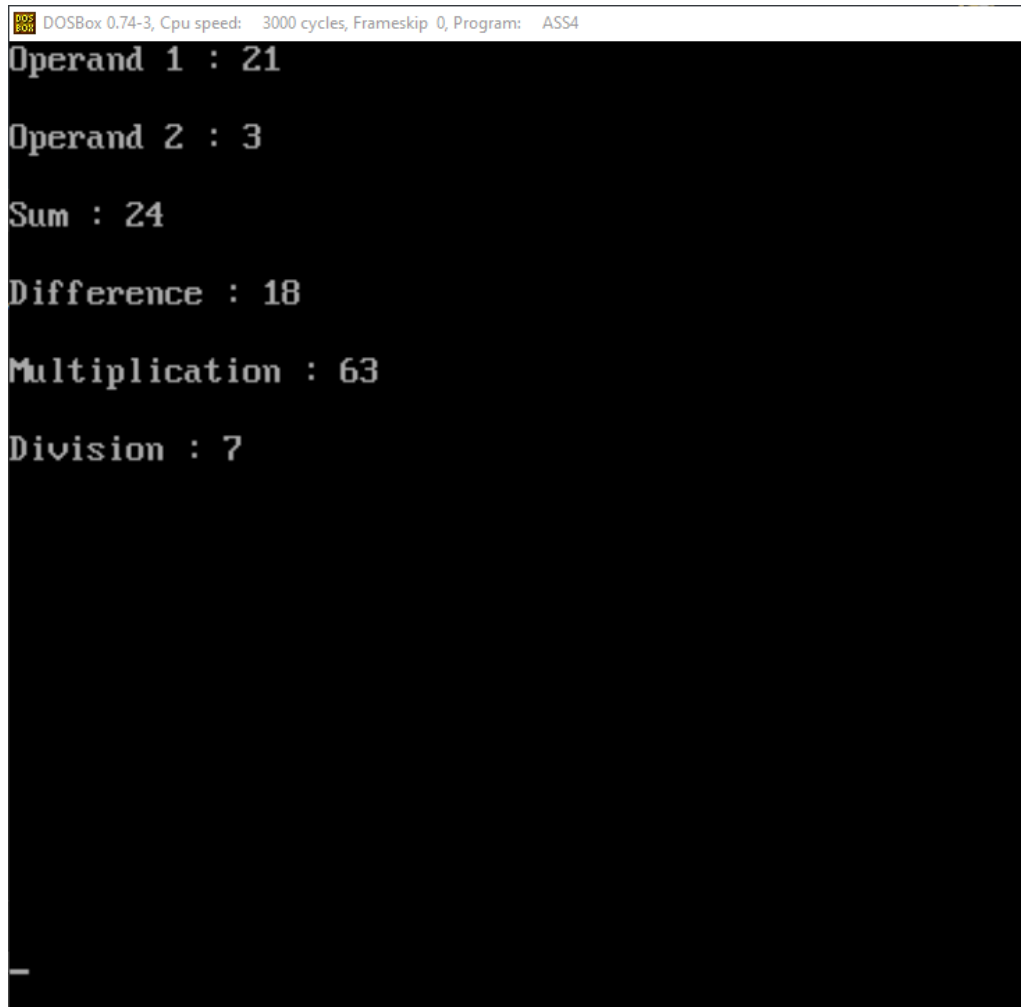
```
    shl ax , 1
    add ax , 160 )
```

what is happening here is that we are finding the place where to start printing the number this is written for printing the value on the second line infront of the string. And similarly we will add 160 with the previous line to switch to the next line (explained above) and multiply 2 with the count of characters in the string to skip that much so it doesn't overwrite that string printing before.

Continue this process and eventually we will achieve the printing as given.

➤ **Skipping A Line And Printing**

We Can also achieve this by skipping a line which is by 320 where we are adding 160 what it will do is it will skip a line.

A screenshot of a DOSBox window. The title bar at the top reads "DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: ASS4". The main window area has a black background with white text. The text is as follows:
Operand 1 : 21

Operand 2 : 3

Sum : 24

Difference : 18

Multiplication : 63

Division : 7

At the bottom left of the window, there is a small white horizontal line.