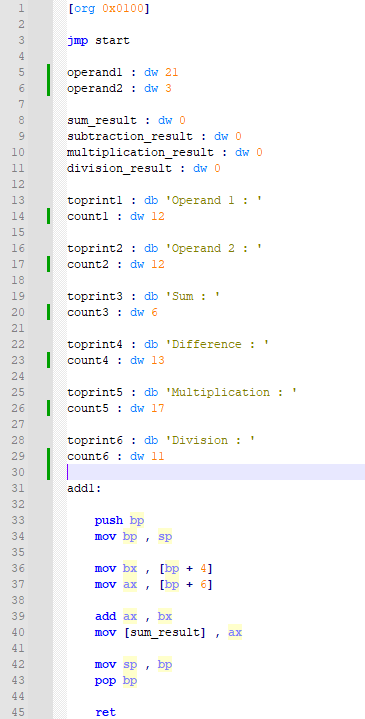
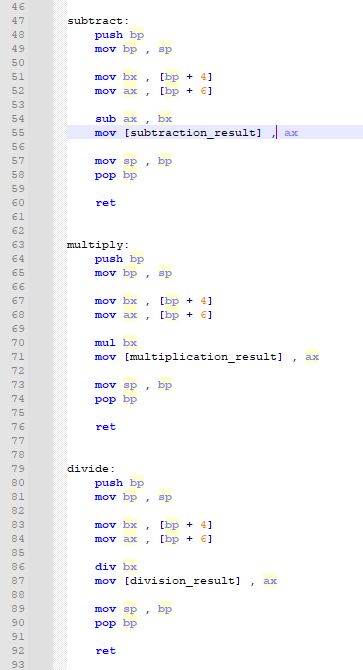
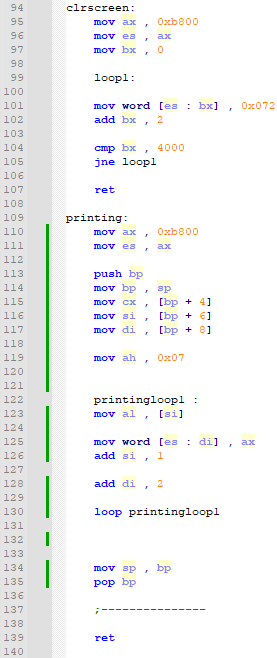
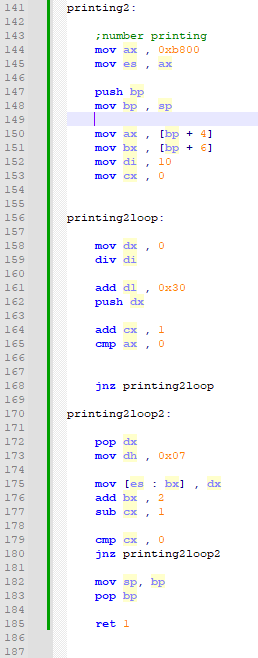
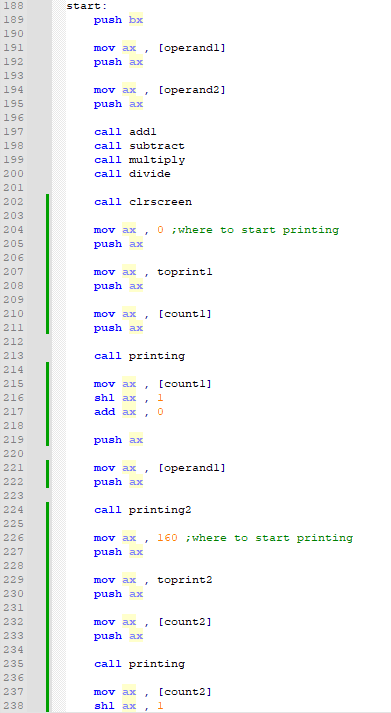
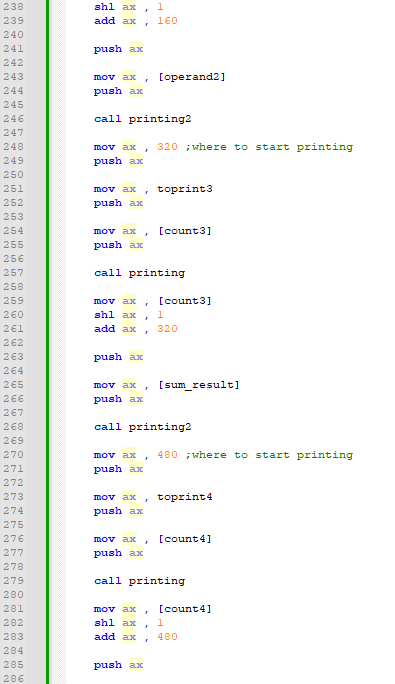
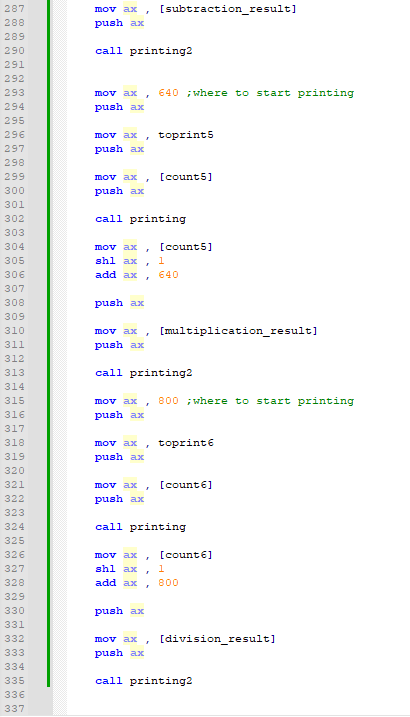
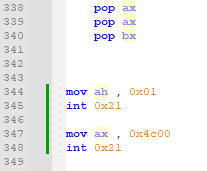
***Question No. 1***

* **Code Screen Shots**

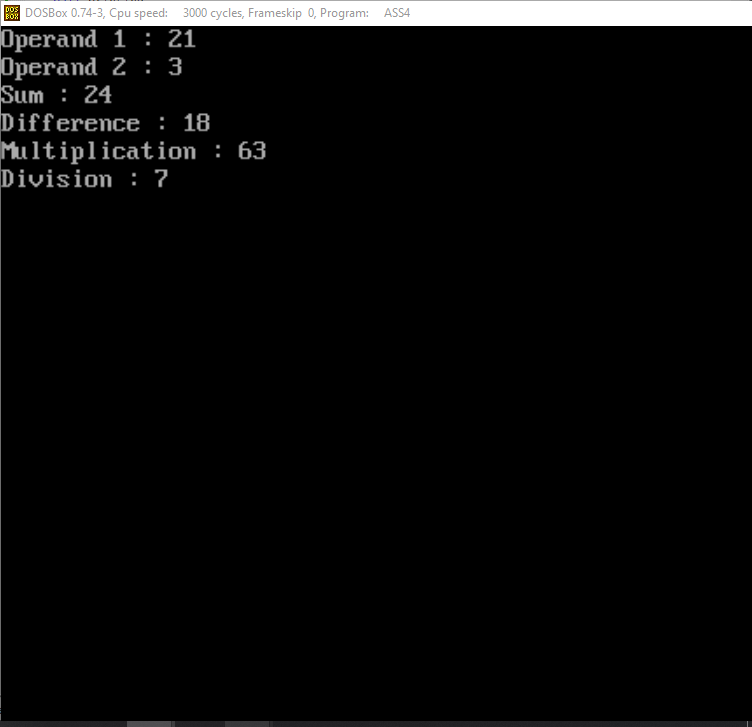








* **Dos Box Screen Shot**



* **Explanation**

The logic here is pretty simple but calling it again and again is a bit complicated so what ive 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 ive 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 clrscreen 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.

