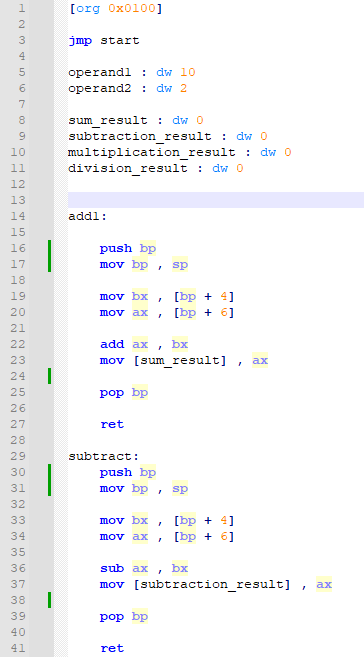
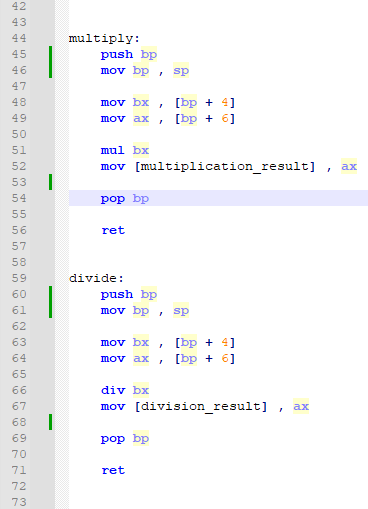
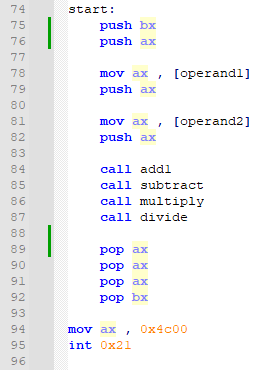
***Question No. 1***

* **Code Screen Shots**





* **Dos Box Screen Shots**



Before Calling The Subroutines



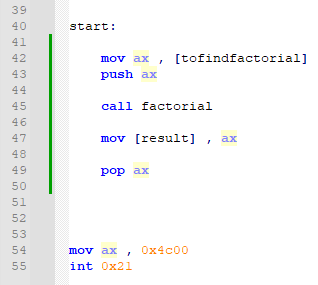
Just Before The Interrupt Command

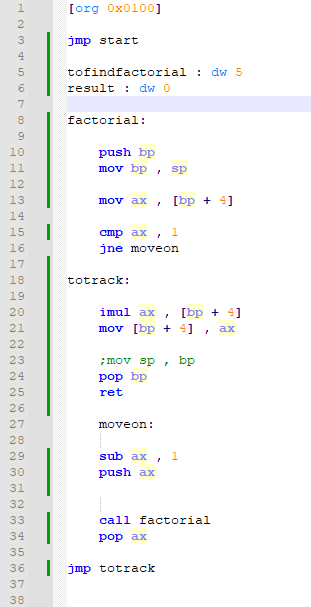
* **Explanation**

From main im passing the operands into the stack so they can be accessed by every subroutine after which im calling the the subroutines one by one and in each subroutine im accessing these operands that were pushed in main and through bp that has been pushed in each subroutine to access those operands and after the operation is done on operands I pop back bp which was pushed , store the result in the desired data location for storage and ret and similarly every subroutine performs on the same logic but the only difference is that the operation in each subroutine is different in main in last I pop back the values from the stack which were pushed initially to put back the initial values in the registers.

***Question No. 2***

* **Code Screen Shots**





* **Dos Box Screen Shots**



Before Calling the Factorial Subroutine For The Value 5

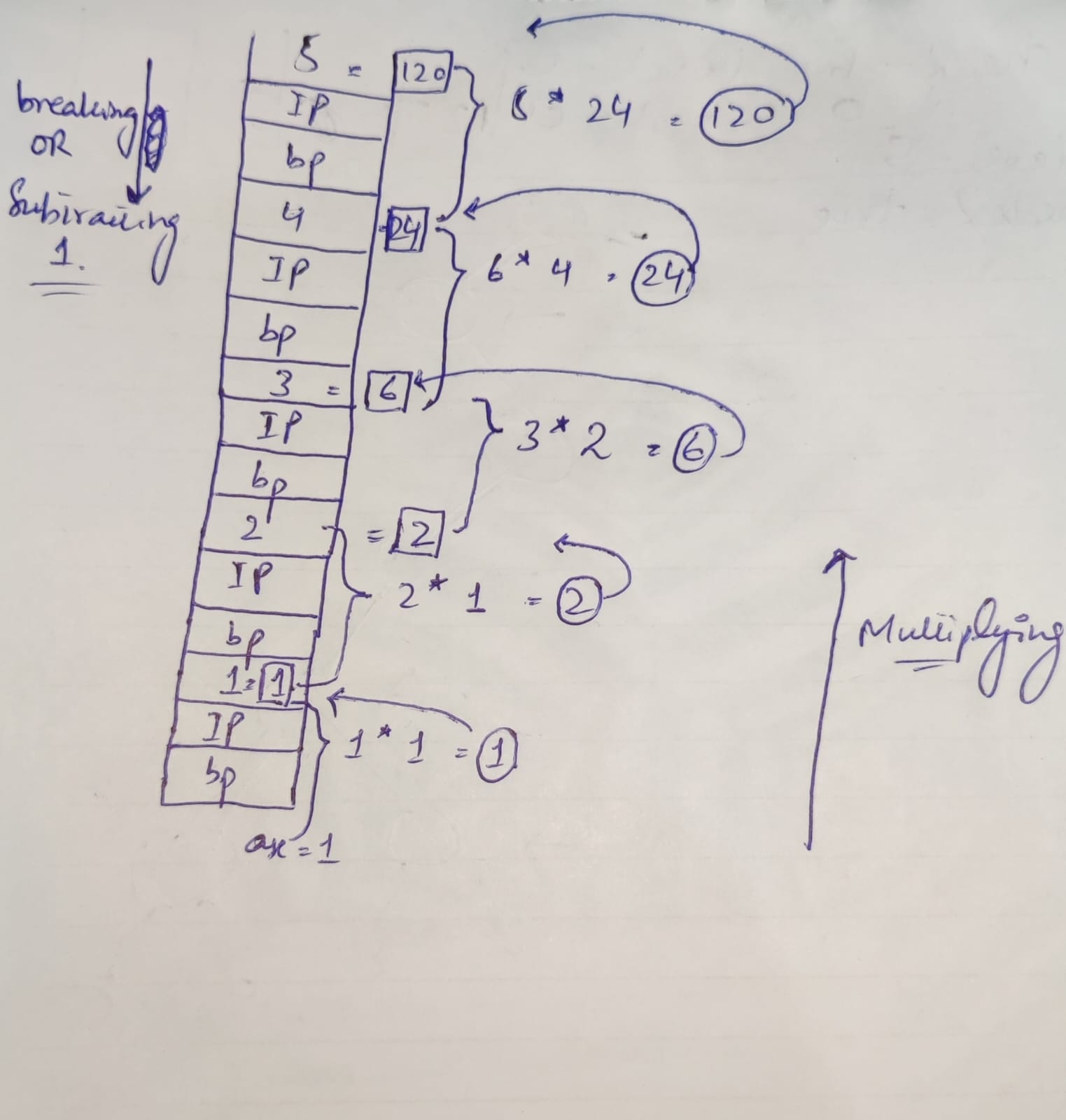


Just Before The Interrupt Command As The Factorial Of 5 Is Stored In The Data Label named as Result We Can see It Gives The Right answer Which Is 78 which’s decimal is 120

* **Explanation**

Im using recursion in this question as an open choice was given to us to do anything but with recursion so im finding the factorial of any given number. Here im testing it with the value of 5 so what is happening is that im passing the value 5 to the subroutine as a parameter and is it also in the stack as it was pushed initially so what does the subroutine does is that it accessed the value 5 from the stack store it into ax and decrement it by 1 becomes 4 and again pushed it into the stack and calls the subroutine it continues happening till it reaches the base condition which is if ax is 1 whenever it is equal to 1 it will multiply the current value of ax with the value of ax which was in the previous called subroutine through [bp + 4] and storing it into [bp + 4] and then returning back to the previously called subroutine and popping ax to get the value of ax in that subroutine and continue the process till it returns back into the main from where it was initially called also im using an unconditional jump totrack to put back the program in correct place after it returns from a recursive call

Like this im using proper usage of stack and recursion in assembly to get the factorial of a given number and store it into the data label named result.

A visual representation is also given below

Just Like this it is using recursion to find the factorial of a given number.