Procedure

Recall the functionalities of a simple calculator which can compute the basic arithmetic operation of addition, multiplication, subtraction and division on gi ven two numbers from the user and display the computed result. For division, you need to display both the quotient and remainder. In case of division, the first number will be divided by the second number whereas in case of subtraction, the first number will be subtracted from the second number. For division operation, you may assume that the second number will always be smaller or equal to first number.

You need not use loops. Once your desired operation is computed, your program will be closed.

However you need to use the procedure to perform your assignment.

<u>You need four procedures as addition, subtraction, division and multiplication</u>

So, what is a procedure?

Large problems can be divided into smaller tasks to make them more manageable
• A procedure is the ASM equivalent of a Java or C++ function
• Following is an assembly language procedure named Sample:

```
Sample:
   …..
   …..
   ret
```

A description of all tasks accomplished by the procedure.
•Receives: A list of input parameters; state their usage and requirements.
• Returns: A description of values returned by the procedure.
• Requires: Optional list of requirements called preconditions that must be satisfied before the procedure is called.

The CALL instruction calls a procedure
• pushes offset of next instruction on the stack
• copies the address of the called procedure into EIP
• The RET instruction returns from a procedure
• pops top of stack into EIP

A sample example

Example
Let us write a very simple procedure named sum that adds the variables stored in the ECX and EDX register and returns the sum in the EAX register −

```
section    .text
   global _start        ;must be declared for using gcc

_start:                    ;tell linker entry point
   mov     ecx,'4'
   sub     ecx, '0'

   mov     edx, '5'
   sub     edx, '0'

   call    sum           ;call sum procedure
   mov     [res], eax
   mov     ecx, msg
   mov     edx, len
   mov     ebx,1         ;file descriptor (stdout)
   mov     eax,4         ;system call number (sys_write)
   int     0x80          ;call kernel

   mov     ecx, res
   mov     edx, 1
   mov     ebx, 1             ;file descriptor (stdout)
   mov     eax, 4             ;system call number (sys_write)
   int     0x80          ;call kernel

   mov     eax,1         ;system call number (sys_exit)
   int     0x80          ;call kernel
sum:
   mov     eax, ecx
   add     eax, edx
   add     eax, '0'
   ret

section .data
msg db "The sum is:", 0xA,0xD
len equ $- msg

segment .bss
res resb 1
```

Command line will ask for the choice from the user. Please see the following input/output scenario for clarification

Input	Output
Please enter your choice: 1. Addition 2. Subtraction 3. Multiplication 4. Division 1 Enter two numbers 2 3	The result is 5
Please enter your choice: 1. Addition 2. Subtraction 3. Multiplication 4. Division 2 Enter two numbers 2 3	The result is -1
Please enter your choice: 1. Addition 2. Subtraction 3. Multiplication 4. Division 3 Enter two numbers 2 3	The result is 6
Please enter your choice: 1. Addition 2. Subtraction 3. Multiplication 4. Division 4 Enter two numbers 3 2	The result is Quotient 1 Remainder 1