Vivekanand Education Society's
Institute of Technology
Approved by AICTE & Affiliated to University of Mumbai

Artificial Intelligence and Data Science Department.

MP / Even Sem 2021-22 / Experiment 4.

YASH SARANG.
47 / D6AD.

EXPERIMENT - 4.

AIM: Assembly programming using Procedure.

THEORY:

Procedures or subroutines are very important in assembly language, as the assembly language programs tend to be large in size. Procedures are identified by a name. Following this name, the body of the procedure is described which performs a well-defined job. End of the procedure is indicated by a return statement.

Following is the syntax to define a procedure −

```
proc_name:
   procedure body
   ...
   ret
```

The procedure is called from another function by using the CALL instruction. The CALL instruction should have the name of the called procedure as an argument as shown below −

```
CALL proc_name
```

The called procedure returns the control to the calling procedure by using the RET instruction.

Program 1:

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
```

The output of program 1:

When the above code is compiled and executed,
it produces the following result −

The sum is: 9
