1]Write an ALP to compute fibocacci series for n inputs using procedures

Program –

```
section .data
   prompt db 'Enter n: '
   prompt_len equ $ - prompt
   msg db 'Series: '
   msg_len equ $ - msg
   space db ' '
   newline db 10
section .bss
   n resb 2
   num1 resb 2
   num2 resb 2
   result resb 2
section .text
global _start

write_proc:
   push ebp
   mov ebp, esp
   mov eax, 4
   mov ebx, 1
   mov ecx, [ebp+12]
   mov edx, [ebp+8]
   int 80h
   mov esp, ebp

  pop ebp
   ret 8

read_proc:
   push ebp
   mov ebp, esp
   mov eax, 3
   mov ebx, 0
   mov ecx, [ebp+12]
   mov edx, [ebp+8]
   int 80h
   mov esp, ebp
   pop ebp
   ret 8

add_proc:
   push ebp
   mov ebp, esp
   movzx eax, byte [ebp+12]
   sub al, '0'
   movzx ebx, byte [ebp+8]
   sub bl, '0'
```

```
    add eax, ebx
    add al, '0'
    mov [result], al
    mov esp, ebp
    pop ebp
    ret 8

_start:
    push prompt
    push prompt_len
    call write_proc
    push n
    push 2
    call read_proc
    push msg
    push msg_len
    call write_proc
    mov byte [num1], '0'
    mov byte [num2], '1'
    movzx ecx, byte [n]
    sub ecx, '0'
loop:
    push ecx
    push num1
    push 1
    call write_proc

    push space
    push 1
    call write_proc

    push dword [num1]
    push dword [num2]
    call add_proc
    mov al, [num2]
    mov [num1], al
    mov al, [result]
    mov [num2], al
    pop ecx
    dec ecx
    jnz loop
    push newline
    push 1
    call write_proc
    mov eax, 1
    mov ebx, 0
    int 80h
```

OUTPUT

```
atharv@Atharv:/mnt/c/Users/Athar/OneDrive/Documents/college/SEM4/MPMC/Labs/exp6$ ./1
Enter n: 5
Series: 0 1 1 2 3
```

2]Write an ALP to calculate a Fibonacci series using procedures

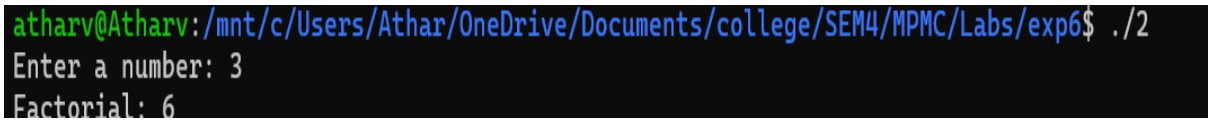
INPUT –

s

```
ection .data
   prompt db 'Enter a number: '
   prompt_len equ $ - prompt
   msg db 'Factorial: '
   msg_len equ $ - msg
   newline db 10

section .bss
   num resb 2
   result resb 1

section .text
global _start

write_proc:
   push ebp
   mov ebp, esp
   mov eax, 4
   mov ebx, 1
   mov ecx, [ebp+12]
   mov edx, [ebp+8]
   int 80h
   mov esp, ebp
   pop ebp
   ret 8

read_proc:
   push ebp
   mov ebp, esp
   mov eax, 3
   mov ebx, 0
   mov ecx, [ebp+12]
   mov edx, [ebp+8]
   int 80h
   mov esp, ebp
   pop ebp
   ret 8

_start:
   push prompt
   push prompt_len
   call write_proc

   push num
   push 2
   call read_proc

   push msg
   push msg_len
   call write_proc
```

```
    movzx eax, byte [num]
    sub al, '0'

    mov ebx, eax
    dec ebx

factorial_loop:
    test ebx, ebx
    jz done
    mul ebx
    dec ebx
    jmp factorial_loop

done:

    add al, '0'
    mov [result], al

    push result
    push 1
    call write_proc

    push newline
    push 1
    call write_proc

    mov eax, 1
    mov ebx, 0
    int 80h
```

OUTPUT –



3]Write an ALP to implement calculator functions using procedures

INPUT –

```
section .data
msg db ' ',10
msgLen equ $-msg
msg1 db 'Number 1: '
msg1Len equ $-msg1
msg2 db 'Number 2: '
msg2Len equ $-msg2
msg3 db 'Sum: '
msg3Len equ $-msg3
msg4 db 'Difference: '
msg4Len equ $-msg4
msg5 db 'Product: '
msg5Len equ $-msg5
msg6 db 'Quotient: '
msg6Len equ $-msg6
msg7 db 'Remainder: '
msg7Len equ $-msg7
msg8 db 'Power: '
msg8Len equ $-msg8

section .bss
num1 RESB 5
num2 RESB 5
sum RESB 5
diff RESB 5
prod RESB 5
quot RESB 5
rem RESB 5
power RESB 5

section .text
global _start

write_proc:
   mov eax, 4
   mov ebx, 1
   ret

read_proc:
   mov eax, 3
   mov ebx, 2
   ret

addition_proc:
   mov eax, [num1]
   sub eax, '0'
   mov ebx, [num2]
   sub ebx, '0'
   add eax, ebx
   add eax, '0'
   mov [sum], eax
   ret

subtraction_proc:
   mov eax, [num1]
   sub eax, '0'
   mov ebx, [num2]
   sub ebx, '0'
   sub eax, ebx
   add eax, '0'
   mov [diff], eax
   ret
```

```
multiplication_proc:
    mov eax, [num1]
    sub eax, '0'
    mov ebx, [num2]
    sub ebx, '0'
    mul ebx
    add eax, '0'
    mov [prod], eax
    ret

division_proc:
    mov al, [num1]
    sub al, '0'
    mov bl, [num2]
    sub bl, '0'
    div bl
    add al, '0'
    mov [quot], al
    add ah, '0'
    mov [rem], ah
    ret

    exponent_proc:
        mov al, [num1]
        sub al, '0'
        mov bl, [num2]
        sub bl, '0'
        mov cl, bl
        mov bl, al
        mov al, 1
    power_loop:
        cmp cl, 0
        je power_done
        mul bl
        dec cl
        jmp power_loop
    power_done:
        add al, '0'
        mov [power], al
        ret

_start:
    call write_proc
    mov ecx, msg1
    mov edx, msg1Len
    int 80h

    call read_proc
    mov ecx, num1
    mov edx, 5
    int 80h

    ; Read second number
    call write_proc
    mov ecx, msg2
    mov edx, msg2Len
    int 80h

    call read_proc
    mov ecx, num2
    mov edx, 5
    int 80h

    ; Addition
    call addition_proc
    call write_proc
    mov ecx, msg3
    mov edx, msg3Len
    int 80h
```

```
call write_proc
mov ecx, sum
mov edx, 1
int 80h

call write_proc
mov ecx, msg
mov edx, msgLen
int 80h

; Subtraction
call subtraction_proc
call write_proc
mov ecx, msg4
mov edx, msg4Len
int 80h

call write_proc
mov ecx, diff
mov edx, 1
int 80h

call write_proc
mov ecx, msg
mov edx, msgLen
int 80h

; Multiplication
call multiplication_proc
call write_proc
mov ecx, msg5
mov edx, msg5Len
int 80h

call write_proc
mov ecx, prod
mov edx, 1
int 80h

call write_proc
mov ecx, msg
mov edx, msgLen
int 80h

; Division
call division_proc
call write_proc
mov ecx, msg6
mov edx, msg6Len
int 80h

call write_proc
mov ecx, quot
mov edx, 1
int 80h

call write_proc
mov ecx, msg
mov edx, msgLen
int 80h

call write_proc
mov ecx, msg7
mov edx, msg7Len
int 80h

call write_proc
mov ecx, rem
mov edx, 1
```

```
int 80h

call write_proc
mov ecx, msg
mov edx, msgLen
int 80h

    ; Calculate power
    call exponent_proc
    call write_proc
    mov ecx, msg8
    mov edx, msg8Len
    int 80h

    call write_proc
mov ecx, power
mov edx, 1
int 80h

call write_proc
mov ecx, msg
mov edx, msgLen
int 80h

; Exit
mov eax, 1
mov ebx, 0
int 80h
```

OUTPUT –

```
atharv@Atharv:/mnt/c/Users/Athar/OneDrive/Documents/college/SEM4/MPMC/Labs/exp6$ ./3
Number 1: 3
Number 2: 2
Sum: 5
Difference: 1
Product: 6
Quotient: 1
Remainder: 1
Power: 9
```

CONCLUSION - Procedures were successfully implemented to complete the programs .