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

```
Program -
section .data
  prompt db 'Enter n: '
                                                  mov esp, ebp
  prompt_len equ $ - prompt
  msg db 'Series: '
                                                 pop ebp
  msg_len equ $ - msg
                                                  ret 8
  space db''
  newline db 10
                                                read_proc:
section .bss
                                                  push ebp
  n resb 2
                                                  mov ebp, esp
  num1 resb 2
                                                  mov eax, 3
  num2 resb 2
                                                  mov ebx, 0
  result resb 2
                                                  mov ecx, [ebp+12]
section .text
                                                  mov edx, [ebp+8]
                                                  int 80h
global start
                                                  mov esp, ebp
write_proc:
                                                  pop ebp
  push ebp
                                                  ret 8
  mov ebp, esp
  mov eax, 4
                                                add_proc:
  mov ebx, 1
                                                  push ebp
  mov ecx, [ebp+12]
                                                  mov ebp, esp
  mov edx, [ebp+8]
                                                  movzx eax, byte [ebp+12]
  int 80h
                                                  sub al, '0'
                                                  movzx ebx, byte [ebp+8]
```

sub bl, '0'

add eax, ebx push 1 add al, '0' call write proc mov [result], al mov esp, ebp push space push 1 pop ebp ret 8 call write_proc push dword [num1] _start: push prompt push dword [num2] push prompt_len call add proc call write_proc mov al, [num2] push n mov [num1], al push 2 mov al, [result] call read proc mov [num2], al push msg рор есх push msg_len dec ecx call write_proc jnz loop mov byte [num1], '0' push newline mov byte [num2], '1' push 1 movzx ecx, byte [n] call write proc sub ecx, '0' mov eax, 1 loop: mov ebx, 0 push ecx 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

push num1

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

INPUT –

S

ection .data	read_proc:
prompt db 'Enter a number: '	push ebp
prompt_len equ \$ - prompt	mov ebp, esp
msg db 'Factorial: '	mov eax, 3
msg_len equ \$ - msg	mov ebx, 0
newline db 10	mov ecx, [ebp+12]
	mov edx, [ebp+8]
section .bss	int 80h
num resb 2	mov esp, ebp
result resb 1	pop ebp
	ret 8
section .text	
global _start	_start:
	push prompt
write_proc:	push prompt_len
push ebp	call write_proc
mov ebp, esp	
mov eax, 4	push num
mov ebx, 1	push 2
mov ecx, [ebp+12]	call read_proc
mov edx, [ebp+8]	
int 80h	push msg
mov esp, ebp	push msg_len
pop ebp	call write_proc
ret 8	

movzx eax, byte [num] mov [result], al sub al, '0' mov ebx, eax push result dec ebx push 1 call write_proc factorial_loop: test ebx, ebx jz done push newline mul ebx push 1 dec ebx call write_proc jmp factorial_loop done: mov eax, 1 mov ebx, 0

OUTPUT -

add al, '0'

atharv@Atharv:/mnt/c/Users/Athar/OneDrive/Documents/college/SEM4/MPMC/Labs/exp6\$./2

int 80h

Enter a number: 3

Factorial: 6

3]Write an ALP to implement calculator functions using procedures

INPUT –	
section .data	global _start
msg db ' ',10	
msgLen equ \$-msg	write_proc:
msg1 db 'Number 1: '	mov eax, 4
msg1Len equ \$-msg1	mov ebx, 1
msg2 db 'Number 2: '	ret
msg2Len equ \$-msg2	
msg3 db 'Sum: '	read_proc:
msg3Len equ \$-msg3	mov eax, 3
msg4 db 'Difference: '	mov ebx, 2
msg4Len equ \$-msg4	ret
msg5 db 'Product: '	
msg5Len equ \$-msg5	addition_proc:
msg6 db 'Quotient: '	mov eax, [num1]
msg6Len equ \$-msg6	sub eax, '0'
msg7 db 'Remainder: '	mov ebx, [num2]
msg7Len equ \$-msg7	sub ebx, '0'
msg8 db 'Power: '	add eax, ebx
msg8Len equ \$-msg8	add eax, '0'
	mov [sum], eax
section .bss	ret
num1 RESB 5	
num2 RESB 5	subtraction_proc:
sum RESB 5	mov eax, [num1]
diff RESB 5	sub eax, '0'
prod RESB 5	mov ebx, [num2]
quot RESB 5	sub ebx, '0'
rem RESB 5	sub eax, ebx
power RESB 5	add eax, '0'

section .text

mov [diff], eax

ret

```
dec cl
multiplication_proc:
                                                                     jmp power_loop
  mov eax, [num1]
                                                                   power_done:
  sub eax, '0'
                                                                     add al, '0'
  mov ebx, [num2]
                                                                     mov [power], al
  sub ebx, '0'
                                                                     ret
  mul ebx
  add eax, '0'
                                                                 _start:
  mov [prod], eax
                                                                   call write_proc
  ret
                                                                   mov ecx, msg1
                                                                   mov edx, msg1Len
division_proc:
                                                                   int 80h
  mov al, [num1]
  sub al, '0'
                                                                   call read_proc
  mov bl, [num2]
                                                                   mov ecx, num1
  sub bl, '0'
                                                                   mov edx, 5
  div bl
                                                                   int 80h
  add al, '0'
  mov [quot], al
                                                                   ; Read second number
  add ah, '0'
                                                                   call write_proc
  mov [rem], ah
                                                                   mov ecx, msg2
  ret
                                                                   mov edx, msg2Len
                                                                   int 80h
  exponent_proc:
    mov al, [num1]
                                                                   call read_proc
    sub al, '0'
                                                                   mov ecx, num2
    mov bl, [num2]
                                                                   mov edx, 5
    sub bl, '0'
                                                                   int 80h
    mov cl, bl
    mov bl, al
                                                                   ; Addition
                                                                   call addition_proc
    mov al, 1
  power_loop:
                                                                   call write_proc
    cmp cl, 0
                                                                   mov ecx, msg3
    je power_done
                                                                   mov edx, msg3Len
    mul bl
                                                                   int 80h
```

call write_proc call write_proc mov ecx, prod mov edx, 1 mov ecx, sum int 80h mov edx, 1 int 80h call write_proc call write_proc mov ecx, msg mov ecx, msg mov edx, msgLen mov edx, msgLen int 80h int 80h ; Division ; Subtraction call division_proc call subtraction_proc call write_proc call write_proc mov ecx, msg6 mov ecx, msg4 mov edx, msg6Len mov edx, msg4Len int 80h int 80h call write_proc call write_proc mov ecx, quot mov ecx, diff mov edx, 1 mov edx, 1 int 80h int 80h call write_proc call write_proc mov ecx, msg mov edx, msgLen mov ecx, msg mov edx, msgLen int 80h int 80h call write_proc ; Multiplication mov ecx, msg7 call multiplication_proc mov edx, msg7Len int 80h call write_proc mov ecx, msg5

call write_proc

mov ecx, rem mov edx, 1

mov edx, msg5Len

int 80h

int 80h mov ecx, power mov edx, 1 int 80h call write_proc mov ecx, msg mov edx, msgLen call write_proc int 80h mov ecx, msg mov edx, msgLen ; Calculate power int 80h call exponent_proc call write_proc ; Exit mov ecx, msg8 mov eax, 1 mov edx, msg8Len mov ebx, 0 int 80h int 80h call write_proc

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 .