Name:Nischal Pradyoth
RollNo.:CS20B1109

Q1) Choose five numbers between 100 and 999. Write ALP by using conditional instructions for finding the largest of these five numbers and display the result as the output.

Code:

```
section .data
        msg db "The largest digit is: ";declaring all data
         len equ $- msg
        num1 dd 100
        num2 dd 183
        num3 dd 213
   num4 dd 300
   num5 dd 515
segment .bss
        largest1 resb 2;declaring in bss section as they get updated
        largest resb 2
        sum resb 40
        sum_len resd 1
section .text
 global _start ;must be declared for using gcc
_start: ;tell linker entry point
        mov ecx, [num1];here first we check the first 3 numbers and then next 2 numbers
        cmp ecx, [num2]
        jg check_third_num
         mov ecx, [num2]
        cmp ecx,[num3]
        jg second
        mov ecx,[num3]
        jmp second
  check_third_num:
        cmp ecx, [num3]
        jg second
         mov ecx, [num3]
         jmp second
second:         ; for num4,num5
   mov [largest1],ecx
   mov ecx,[num4]
   cmp ecx,[num5]
   jg final
   mov ecx,[num5]
   jmp final
final:;final updation
```

```
   cmp ecx,[largest1]
   jg _exit
   mov ecx,[largest1]
_exit:
          mov [largest], ecx

  mov eax,[largest]
         mov edi, sum              ; Argument: Address of the target string
  call int2str              ; Get the digits of EAX and store it as ASCII
  sub edi, sum
  mov [sum_len], edi

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

   ; Output sum
   mov eax, 4
   mov ebx, 1
   mov ecx, sum        ;printing digit
   mov edx, [sum_len]
   int 0x80
; Exit code
   mov eax, 1
   mov ebx, 0
   int 0x80

int2str:    ; Converts an positive integer in EAX to a string pointed to by EDI
   xor ecx, ecx
   mov ebx, 10
   .LL1:                ; First loop: Save the remainders
   xor edx, edx           ; Clear EDX for div
   div ebx              ; EDX:EAX/EBX -> EAX Remainder EDX
   push dx              ; Save remainder
   inc ecx              ; Increment push counter
   test eax, eax          ; Anything left to divide?
   jnz .LL1             ; Yes: loop once more
.LL2:               ; Second loop: Retrieve the remainders
   pop dx               ; In DL is the value
   or dl, '0'           ; To ASCII
   mov [edi], dl          ; Save it to the string
   inc edi              ; Increment the pointer to the string
```

```
    loop .LL2               ; Loop ECX times

    mov byte [edi], 0       ; Termination character
    ret
```

Q2) Choose a random number 20 ≤ R ≤ 100, write ALP by using loops for printing the numbers (i.e. from 1 to R) and display the result as the output.

Code:

```
segment .bss ;declaring data
    sum resb 40
    sum_len resd 1
    num resb 1
section .data
    msg db " ",0xa
    len equ $-msg
    msg2 db "The numbers are:"
    len2 equ $-msg2
section .text
 global _start ;must be declared for using gcc
_start: ;tell linker entry point
    mov ecx, msg2       ;printing in a new line
    mov edx, len2
    mov ebx, 1
    mov eax, 4
    int 0x80

          mov ecx,40
          mov eax, 1
l1:
    push ecx
    mov [num],eax
    mov edi, sum              ; Argument: Address of the target string
    call int2str              ; Get the digits of EAX and store it as ASCII
    sub edi, sum              ; EDI (pointer to the terminating NULL) - pointer to sum = length of the
string
    mov [sum_len], edi

    mov eax, 4
    mov ebx, 1
    mov ecx, sum       ;printing the number as a string
    mov edx, [sum_len]
    int 0x80
```

```
    mov ecx, msg        ;printing in a new line
    mov edx, len
    mov ebx, 1
    mov eax, 4
    int 0x80

mov eax, [num]    ;bring the content back to eax
inc eax           ; increment eax
pop ecx                     ; loop counter is restored in to ecx
loop l1

    ; Exit code
    mov eax, 1
    mov ebx, 0
    int 0x80
int2str:    ; Converts an positive integer in EAX to a string pointed to by EDI
    xor ecx, ecx
    mov ebx, 10
    .LL1:                ; First loop: Save the remainders
    xor edx, edx           ; Clear EDX for div
    div ebx               ; EDX:EAX/EBX -> EAX Remainder EDX
    push dx               ; Save remainder
    inc ecx               ; Increment push counter
    test eax, eax          ; Anything left to divide?
    jnz .LL1              ; Yes: loop once more
.LL2:                ; Second loop: Retrieve the remainders
    pop dx                ; In DL is the value
    or dl, '0'            ; To ASCII
    mov [edi], dl          ; Save it to the string
    inc edi               ; Increment the pointer to the string
    loop .LL2              ; Loop ECX times

    mov byte [edi], 0       ; Termination character
    ret
```

Q3) Write ALP for finding a factorial of number between 5 and 30 and display the result as the Output.

Code:

```
section .data       ;storing data
  msg db 'The factorial of 7 is:'
  len equ $-msg
section .bss
  sum resb 40
```

```
  sum_len resd 1
section .text
  global _start
_start:
  mov eax, 1
  mov ecx, 7
.L2:        ;loop for the factorial
          mul ecx
          dec ecx
          jnz .L2
mov edi, sum    ;this all is for printing the number as a string
    call int2str
    sub edi, sum
    mov [sum_len], edi

    mov ecx, msg
    mov edx, len
    mov ebx, 1
    mov eax, 4
    int 0x80

    ; Output result
    mov eax, 4
    mov ebx, 1
    mov ecx, sum
    mov edx, [sum_len]
    int 0x80
; Exit code
    mov eax, 1
    mov ebx, 0
    int 0x80

int2str:    ;function to push individual numbers into stack
    xor ecx, ecx
    mov ebx, 10
    .LL1:
    xor edx, edx
    div ebx
    push dx
    inc ecx
    test eax, eax
    jnz .LL1
.LL2:       ;function to pop those pushed numbers from stack
    pop dx
```

```
    or dl, '0'
    mov [edi], dl
    inc edi
    loop .LL2

    mov byte [edi], 0
    ret
```

Q4) Choose a positive value 5 ≤ P ≤ 20, write ALP by defining a P-element array containing P random values and do the sum of these values in the array and display the result as the output.

Code:

```
section .text
  global _start
_start:
  mov eax,6;taking size of the array
  mov ebx,0 ;taking ebx as 0 as it calculates sum
  mov ecx,array ;base address of x to ecx
top:  ;loop
  add ebx,[ecx]
  inc ecx
  dec eax
  jnz top
add ebx,'0';adding '0' for printing
mov [sum],ebx ;moving result

mov ecx,msg ;printing message
mov edx,len
mov eax,4
mov ebx,1
int 0x80

mov ecx,sum ;printing result
mov edx,1
mov ebx,1
mov eax,4
int 0x80

mov eax,1;exiting kernel
mov ebx,0
int 0x80

section .data ;storing data
  msg db 'The sum of the 6-sized array is:'
```

```
len equ $-msg
array db 1,2,3,4,5,-6
sum db 0
```

Result

CPU Time: 0.00 sec(s), Memory: 348 kilobyte(s)

compiled and executed in 0.925 sec(s)

```
The largest digit is: 515
```

CPU Time: 0.00 sec(s), Memory: 348 kilobyte(s)

compiled and executed in 1.069 sec(s)

```
The numbers are:1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
```

Result

CPU Time: 0.00 sec(s), Memory: 344 kilobyte(s)

compiled and executed in 0.979 sec(s)

```
The factorial of 7 is:5040
```

Result

CPU Time: 0.00 sec(s), Memory: 348 kilobyte(s)

compiled and executed in 1.089 sec(s)

```
The sum of the 6-sized array is:9
```