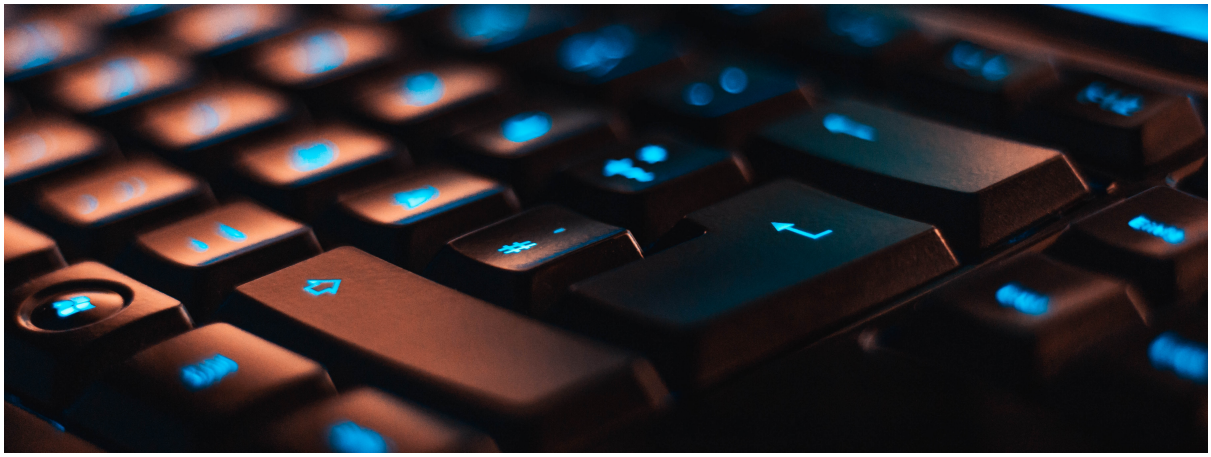


COA Lab Assignment 1

Roll No.: CS20B1016

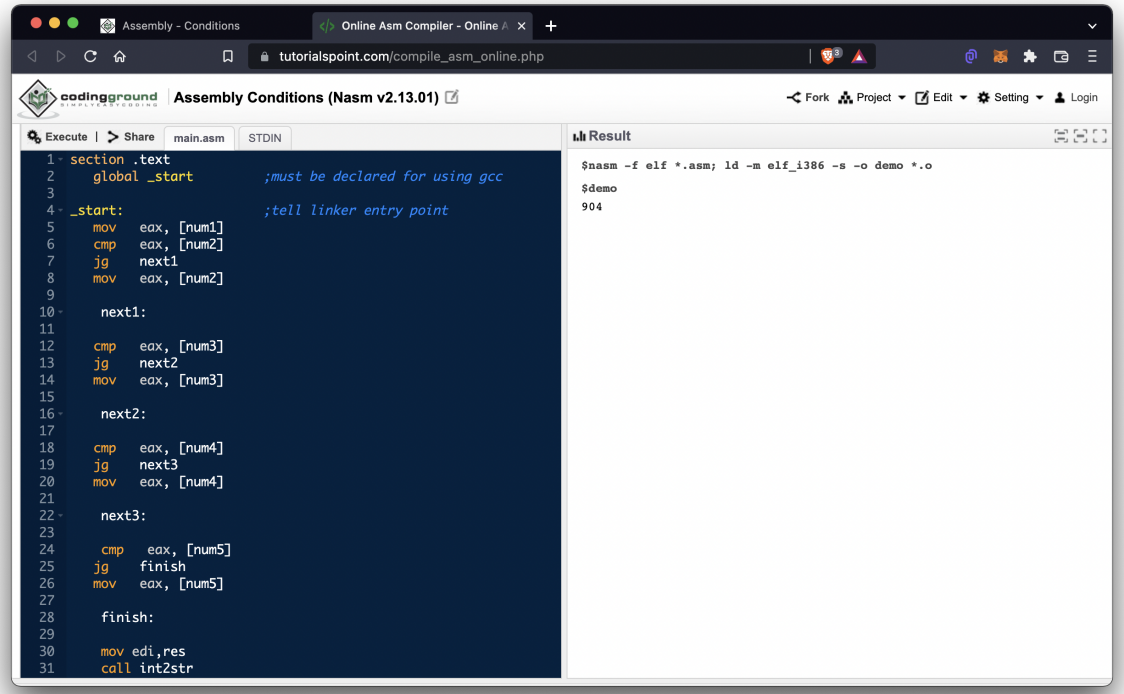
Name: V Nagasai

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.

Solution:

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

_start:                  ;tell linker entry point
   mov   eax, [num1]
   cmp   eax, [num2]
   jg    next1
   mov   eax, [num2]

  next1:

   cmp   eax, [num3]
   jg    next2
   mov   eax, [num3]

  next2:

   cmp   eax, [num4]
   jg    next3
   mov   eax, [num4]

  next3:

    cmp   eax, [num5]
   jg    finish
   mov   eax, [num5]

  finish:

  mov edi,res
```

```
  call int2str
  sub edi,res
  mov [len],edi

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

   mov   eax, 1
   int   80h

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                     ; RET: EDI points to the terminating NULL

section .data

   num1 dd 101
   num2 dd 201
   num3 dd 301
   num4 dd 505
   num5 dd 704

segment .bss
   res resd 20
   len resd 5
```

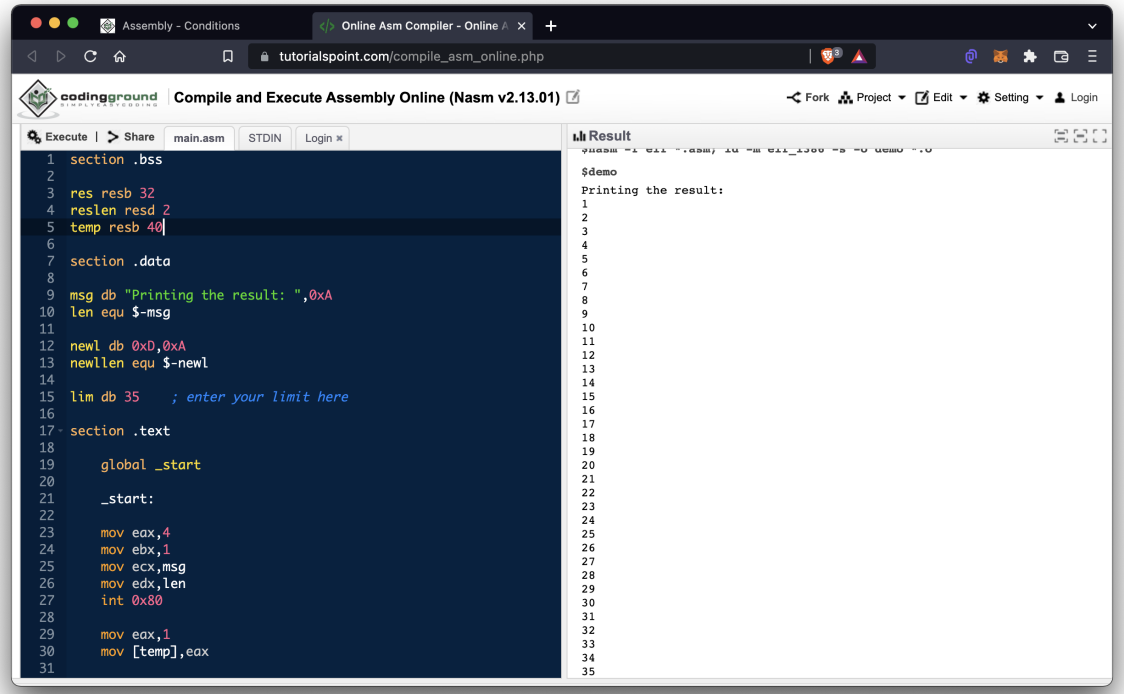


Q2. Choose a random number 20 ≤ R ≤ 100, write ALP by using loops for printing the numbers (i.e. from 1 to R) and display the result as the output.

Solution:

```
section .bss

res resb 32
reslen resd 2
temp resb 40

section .data

msg db "Printing the result: ",0xA
len equ $-msg

newl db 0xD,0xA
newllen equ $-newl

lim db 35    ; enter your limit here

section .text

    global _start

    _start:

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

    mov eax,1
    mov [temp],eax

    ll_1:
```

```
    mov edi, res                ; Argument: Address of the target string
    call int2str                ; Get the digits of EAX and store it as ASCII
    sub edi, res                ; EDI (pointer to the terminating NULL) - pointer to sum = length of the string
    mov [reslen], edi

    mov eax,4
    mov ebx,1
    mov ecx,res
    mov edx,[reslen]
    int 0x80

    mov eax,4
    mov ebx,1
    mov ecx,newl
    mov edx,newllen
    int 0x80

    mov eax,[temp]
    inc eax
    mov [temp],eax

    cmp eax,[lim]
    jg _exit

    jmp ll_1

    _exit:


; 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                     ; RET: EDI points to the terminating NULL


; 0xA -> new line
; 0xD -> carriage return
```

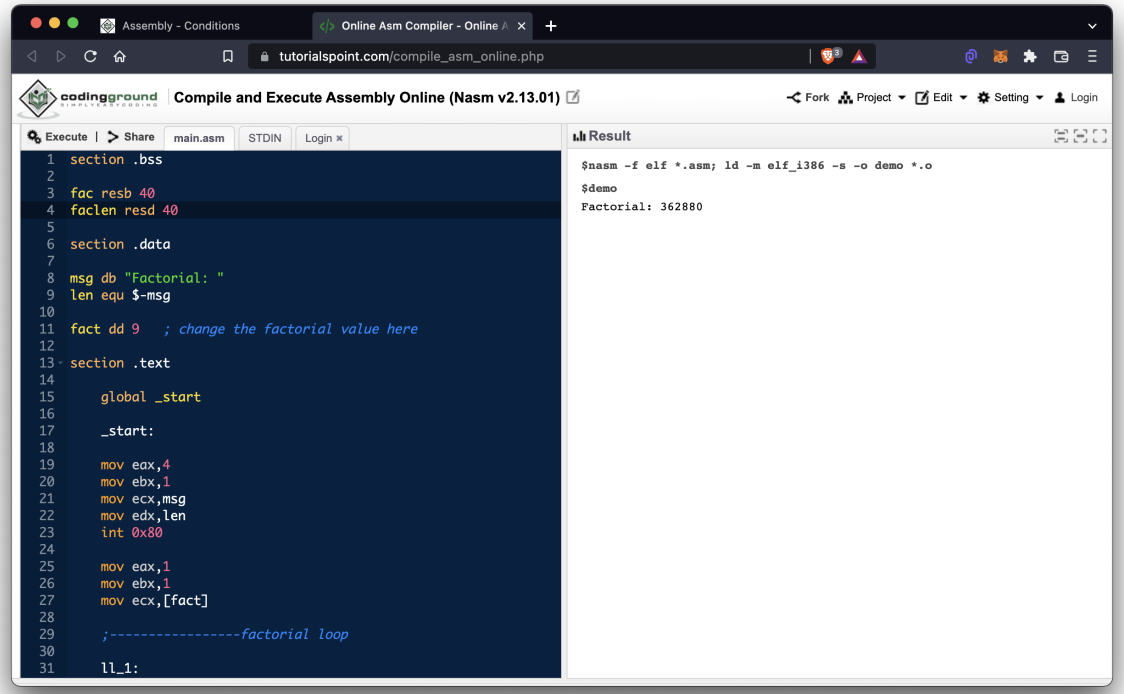


Q3) Write ALP for finding a factorial of number between 5 and 30 and display the result as the output.

Solution:

```
section .bss

fac resb 40
faclen resd 40

section .data

msg db "Factorial: "
len equ $-msg

fact dd 9   ; change the factorial value here

section .text

    global _start

    _start:

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

    mov eax,1
    mov ebx,1
    mov ecx,[fact]
```

```
    ;-----------------factorial loop

    ll_1:

    mul ebx
    inc ebx

    loop ll_1

    ;--------------------------------displaying result

        ; Convert the number to a string
    mov edi, fac                ; Argument: Address of the target string
    call int2str                ; Get the digits of EAX and store it as ASCII
    sub edi, fac                ; EDI (pointer to the terminating NULL) - pointer to sum = length of the string
    mov [faclen], edi


    mov eax, 4
    mov ebx, 1
    mov ecx, fac
    mov edx, [faclen]
    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                     ; RET: EDI points to the terminating NULL
```

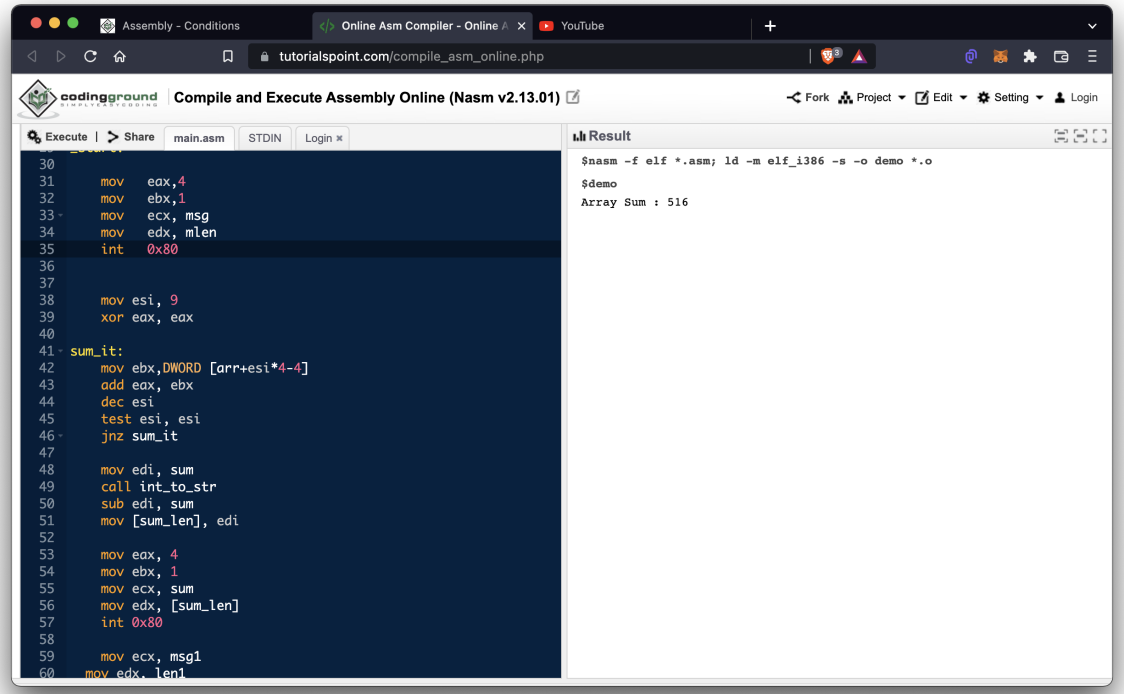


Q4) Choose a positive value 5 ≤ P ≤ 20, write ALP by defining a P-element array containing P random values and do the sum of these values in the array and display the result as the output.

Solution:

```
section .data

    global arr

    len db 9

    msg db "Array Sum : "
  mlen equ $ - msg

      msg1 db " ", 0xA
  len1 equ $- msg1

arr:
  dd 50
  dd 65
  dd 15
  dd 33
  dd 35
  dd 75
  dd 57
  dd 99
  dd 87


section .text

    global _start

_start:

  mov   eax,4
```

```
  mov   ebx,1
  mov   ecx, msg
  mov   edx, mlen
  int   0x80


  mov esi, 9
  xor eax, eax

sum_it:
  mov ebx,DWORD [arr+esi*4-4]
  add eax, ebx
  dec esi
  test esi, esi
  jnz sum_it

    mov edi, sum
    call int_to_str
    sub edi, sum
    mov [sum_len], edi

    mov eax, 4
    mov ebx, 1
    mov ecx, sum
    mov edx, [sum_len]
    int 0x80

    mov ecx, msg1
  mov edx, len1
  mov ebx, 1
  mov eax, 4
  int 0x80

  ; gracefull exit
  mov eax, 1
  mov ebx, 0
  int 0x80

int_to_str:
    ; 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

section .bss

    sum resb 40
    sum_len resd 1
```


Assembly - Conditions
Online Asm Compiler - Online A
YouTube
tutorialspoint.com/compile_asm_online.php
codingground
Compile and Execute Assembly Online (Nasm v2.13.01)
Fork
Project
Edit
Setting
Login
Execute
Share
main.asm
STDIN
Login
mov eax,4
mov ebx,1
mov ecx, msg
mov edx, mlen
int 0x80
mov esi, 9
xor eax, eax
sum_it:
mov ebx,DWORD [arr+esi*4-4]
add eax, ebx
dec esi
test esi, esi
jnz sum_it
mov edi, sum
call int_to_str
sub edi, sum
mov [sum_len], edi
mov eax, 4
mov ebx, 1
mov ecx, sum
mov edx, [sum_len]
int 0x80
mov ecx, msg1
Result
$nasm -f elf *.asm; ld -m elf_i386 -s -o demo *.o
$demo
Array Sum : 516