Name -: Diksha Chaudhari

Roll No -: 20U433

Subject -: ML

Division -: IV

Assignment No. 1

PROBLEM STATEMENT :

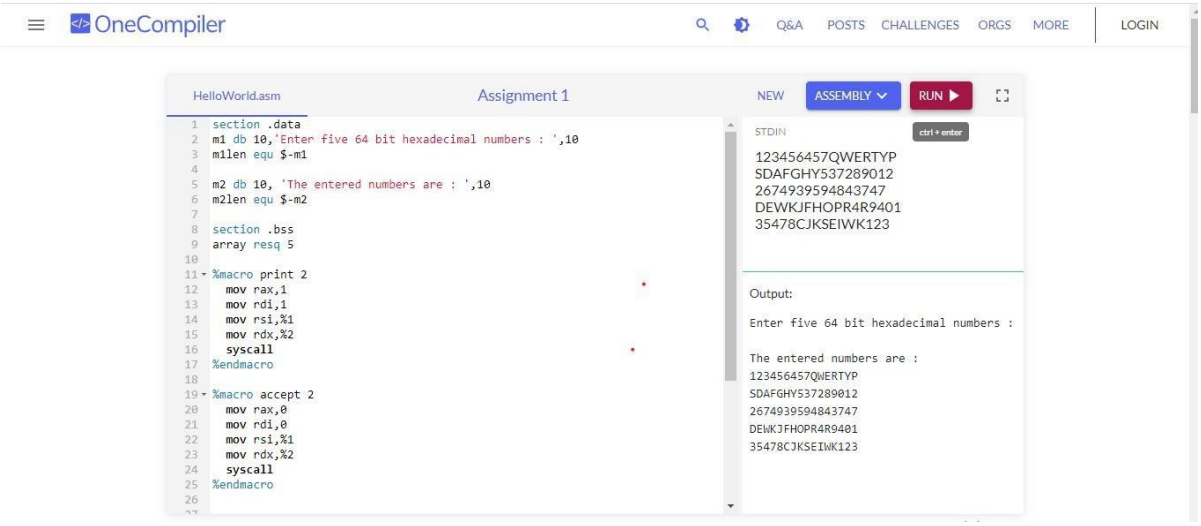
Write an X86/64 ALP to accept five 64 bit Hexadecimal numbers from user and store them in an array and display the accepted numbers.

PROGRAM:

```
section .data
m1 db 10,'Enter five 64 bit hexadecimal numbers : ',10
m1len equ $-m1

m2 db 10, 'The entered numbers are : ',10
m2len equ $-m2

section .bss
array resq 5

%macro print 2
mov rax,1 mov
rdi,1 mov
rsi,%1 mov
rdx,%2
syscall
%endmacro

%macro accept 2
mov rax,0 mov
rdi,0 mov rsi,%1
mov rdx,%2
syscall
%endmacro
```

```
section .text
global _start

_start:
print m1 , m1len
accept array, 84 print
m2 , m2len print
array, 84

mov rax,60
mov rdi,0
syscall
```

OUTPUT:



Assignment No. 2

PROBLEM STATEMENT :

Write an X86/64 ALP to count number of positive and negative numbers from the array.

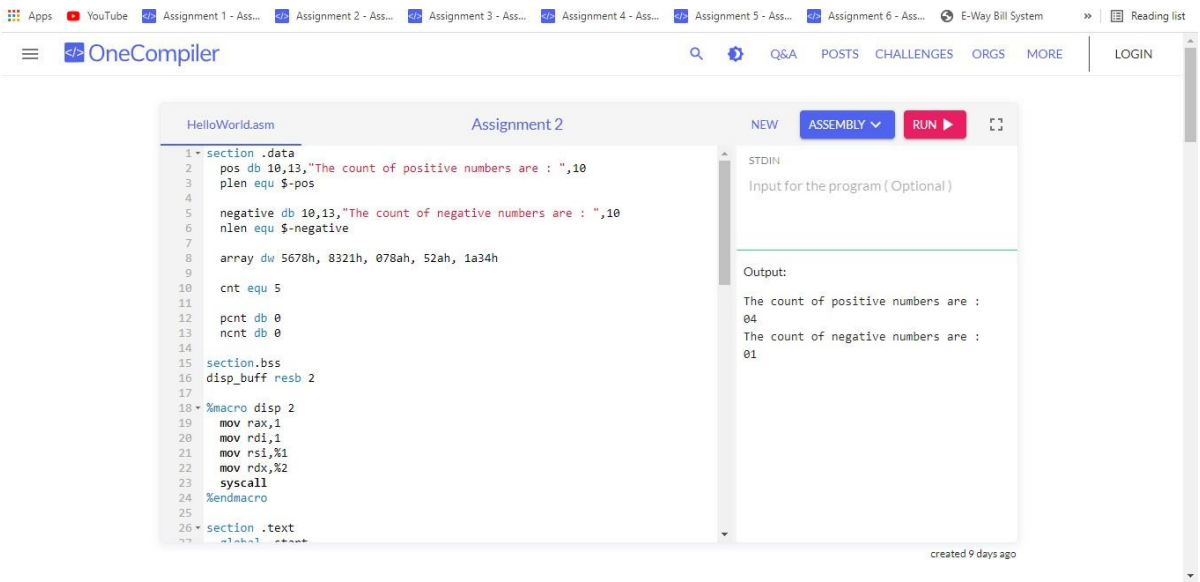
PROGRAM:

```
section .data
  pos db 10,13,"The count of positive numbers are : ",10
  plen equ $-pos

  negative db 10,13,"The count of negative numbers are : ",10
  nlen equ $-negative

  array dw 5678h, 8321h, 078ah, 52ah, 1a34h

  cnt equ 5

  pcnt db 0
  ncnt db 0

section.bss
disp_buff resb 2

%macro disp 2
  mov rax,1
        mov rdi,1
        mov rsi,%1
        mov rdx,%2
        syscall
%endmacro

section .text
        global _start

_start:
  mov esi,array
```

```
  mov ecx,cnt

back:
  BT word[esi],15
  jc x
  inc byte[pcnt]
  jmp skip

x:
  inc byte[ncnt]

skip:
  add esi,2
  loop back

  disp  pos,plen
  mov bl,[pcnt]
  call display

  disp negative,nlen
  mov bl,[ncnt]
  call display

  mov rax,60
  mov rdi,0
  syscall

display:
  mov ecx,2
  mov edi, disp_buff

up:
  rol bl,4
  mov al,bl
  and al,0fh
  cmp al,09h
  jbe y
  add al,07h

y:
```

```
add al,30h
mov [edi],al
inc edi
loop up
disp disp_buff,2

ret
```

OUTPUT:




```
section .data
  pos db 10,13,"The count of positive numbers are : ",10
  plen equ $-pos

  negative db 10,13,"The count of negative numbers are : ",10
  nlen equ $-negative

  array dw 5678h, 8321h, 078ah, 52ah, 1a34h

  cnt equ 5

  pcnt db 0
  ncnt db 0

section.bss
disp_buff resb 2

%macro disp 2
  mov rax,1
  mov rdi,1
  mov rsi,%1
  mov rdx,%2
  syscall
%endmacro

section .text
```

Assignment No. 3

PROBLEM STATEMENT :

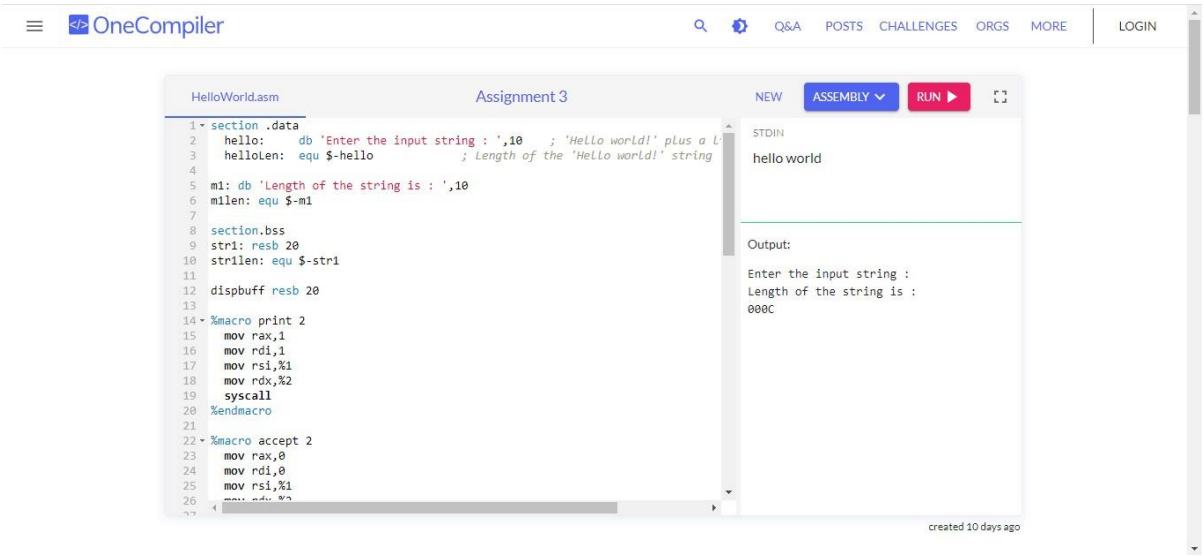
Write an X86/64 ALP to accept a string and to display its length.

PROGRAM:

```
section .data
        hello:     db 'Enter the input string : ',10
        helloLen:  equ $-hello

m1: db 'Length of the string is : ',10
m1len: equ $-m1

section.bss
str1: resb 20
str1len: equ $-str1

dispbuff resb 20

%macro print 2
  mov rax,1
        mov rdi,1
        mov rsi,%1
        mov rdx,%2
        syscall
%endmacro

%macro accept 2
        mov rax,0
        mov rdi,0
        mov rsi,%1
        mov rdx,%2
        syscall
%endmacro

section .text
        global _start
```

```
_start:
  print hello,helloLen
  print m1,m1len
  accept str1,str1len
  mov bx,ax
  call display

  mov rax,60
  mov rdi,0
  syscall

display:
  mov ecx,4
  mov edi,dispbuff

again:
  rol bx,4
  mov al,bl
  and al,0Fh
  cmp al,9
  jbe l1
  cmp al,'a'
  jb l2
  add al,57h
  jmp l3

l1:
  add al,30h
  jmp l3

l2:
  add al,37h

l3:
  mov[edi],al
  inc  edi
  loop again
  print dispbuff,4
  ret
```

OUTPUT:


OneCompiler
Q&A
POSTS
CHALLENGES
ORGS
MORE
LOGIN
HelloWorld.asm
Assignment 3
NEW
ASSEMBLY
RUN
section .data
hello: db 'Enter the input string : ',10 ; 'Hello world!' plus a l
helloLen: equ $-hello ; Length of the 'Hello world!' string
m1: db 'Length of the string is : ',10
m1len: equ $-m1
section.bss
str1: resb 20
str1len: equ $-str1
dispbuff resb 20
%macro print 2
mov rax,1
mov rdi,1
mov rsi,%1
mov rdx,%2
syscall
%endmacro
%macro accept 2
mov rax,0
mov rdi,0
mov rsi,%1
STDIN
hello world
Output:
Enter the input string :
Length of the string is :
000C
created 10 days ago

Assignment No. 4

PROBLEM STATEMENT :

Write an X86/64 ALP to perform non-overlapped block transfer without string specific instructions. Block containing data can be defined in the data segment.

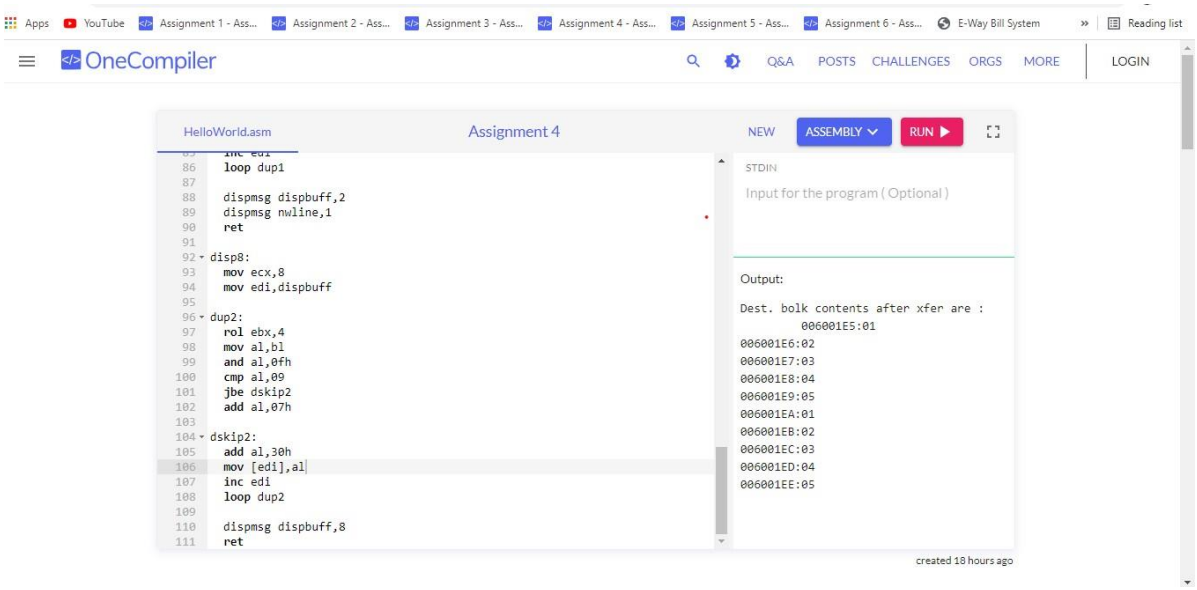
PROGRAM:

```
section .data
        dst_blk db 10,'Dest. bolk contents after xfer are :'
        dst_len  equ $-dst_blk

array db 01h,02h,03h,04h,05h
newarray times 5 db 0
cnt equ 05h
colon db ':'
nwline db 10

section .bss
choice resb 02
dispbuff resb 08

%macro dispmsg 2
  mov eax,04
  mov  ebx,01
  mov  ecx,%1
  mov edx,%2
  int 80h
%endmacro

%macro accept 2
        mov eax,03
        mov ebx,0
        mov ecx,%1
        mov edx,%2
        int 80h
```

```
%endmacro

section .text
        global _start

_start:
  menu:
  dispmsg dst_blk,dst_len
  dispmsg nwline,10

  mov  esi,array
  mov edi,newarray
  mov ecx,cnt

q:
  mov  al,[esi]
  mov [edi],al
  inc esi
  inc edi
  loop q

  mov esi,array
  xor rcx,rcx
  mov rcx,10

up2:
  push rcx
  mov ebx,esi
  call disp8
  dispmsg colon,1
  mov bl,[esi]
  call disp2
  inc esi
  pop rcx
  loop up2

  mov eax,01
mov ebx,00
int 80h
disp2:
```

```
  mov ecx,2
  mov edi,dispbuff

dup1:
  rol bl,4
  mov al,bl
  and al,0fh
  cmp al,09
  jbe dskip
  add al,07h

dskip:
  add al,30h
  mov [edi],al
  inc edi
  loop dup1

  dispmsg dispbuff,2
  dispmsg nwline,1
  ret

disp8:
  mov ecx,8
  mov edi,dispbuff

dup2:
  rol ebx,4
  mov al,bl
  and al,0fh
  cmp al,09
  jbe dskip2
  add al,07h

dskip2:
  add al,30h
  mov [edi],al
  inc edi
  loop dup2
 dispmsg dispbuff,8
 ret
```

OUTPUT:




```
  loop dup1

  dispmsg dispbuff,2
  dispmsg nwline,1
  ret

disp8:
  mov ecx,8
  mov edi,dispbuff

dup2:
  rol ebx,4
  mov al,bl
  and al,0fh
  cmp al,09
  jbe dskip2
  add al,07h

dskip2:
  add al,30h
  mov [edi],al
  inc edi
  loop dup2

  dispmsg dispbuff,8
  ret
```

Assignment No. 5

PROBLEM STATEMENT :

Write an X86/64 ALP to perform overlapped block transfer with string specific instructions. Block containing data can be defined in the data segment.

PROGRAM:

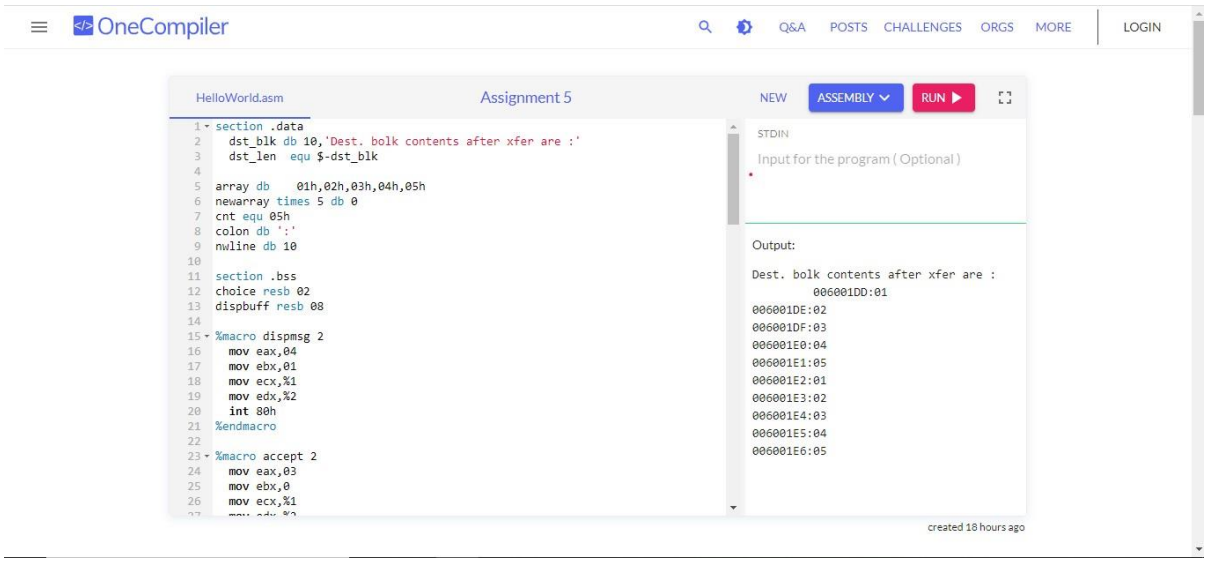
```
section .data
        dst_blk db 10,'Dest. bolk contents after xfer are :'
        dst_len  equ $-dst_blk

array db    01h,02h,03h,04h,05h
newarray times 5 db 0
cnt equ 05h
colon db ':'
nwline db 10

section .bss
choice resb 02
dispbuff resb 08

%macro dispmsg 2
  mov eax,04
  mov  ebx,01
  mov  ecx,%1
  mov edx,%2
  int 80h
%endmacro

%macro accept 2
        mov eax,03
        mov ebx,0
        mov ecx,%1
```

```
        mov edx,%2
        int 80h
%endmacro

section .text
        global _start

_start:
  menu:

  dispmsg dst_blk,dst_len
  dispmsg nwline,10

  mov  esi,array
  mov edi,newarray
  mov ecx,cnt

  rep movsb
  cld

  mov esi,array
  xor rcx,rcx
  mov rcx,10

up2:
  push rcx
  mov ebx,esi
  call disp8
  dispmsg colon,1
  mov bl,[esi]
  call disp2
  inc esi
  pop rcx
  loop up2

  mov eax,01
  mov ebx,00
```

```
  int 80h

disp2:
  mov ecx,2
  mov edi,dispbuff

dup1:
  rol bl,4
  mov al,bl
  and al,0fh
  cmp al,09
  jbe dskip
  add al,07h

dskip:
  add al,30h
  mov [edi],al
  inc edi
  loop dup1

  dispmsg dispbuff,2
  dispmsg nwline,1
  ret

disp8:
  mov ecx,8
  mov edi,dispbuff

dup2:
  rol ebx,4
  mov al,bl
  and al,0fh
  cmp al,09
  jbe dskip2
  add al,07h

dskip2:
```

```
add al,30h
mov [edi],al
inc edi
loop dup2

dispmsg dispbuff,8
ret
```

OUTPUT:



Assignment No. 6

PROBLEM STATEMENT :

Write an X86/64 ALP to find largest number from an array.

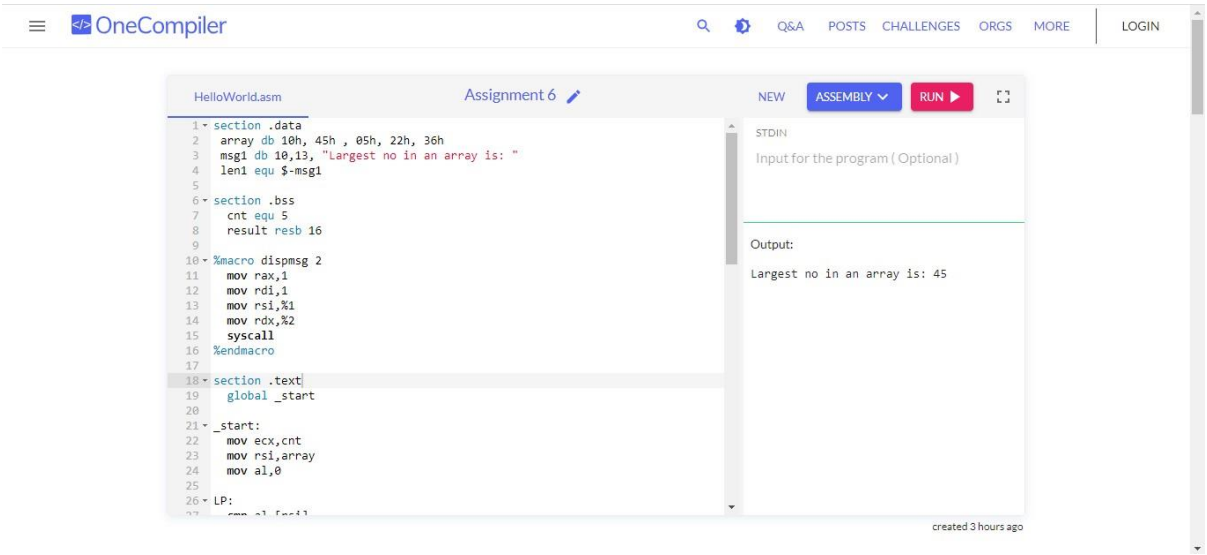
PROGRAM:

```
section .data
 array db 11h, 55h , 33h, 22h, 44h
 msg1 db 10,13, "Largest no in an array is: "
 len1 equ $-msg1

section .bss
  cnt equ 5
  result resb 16

%macro dispmsg 2
  mov rax,1
  mov rdi,1
  mov rsi,%1
  mov rdx,%2
  syscall
%endmacro

section .text
  global _start

_start:
  mov ecx,cnt
  mov rsi,array
  mov al,0

LP:
  cmp al,[rsi]
  jg skip
  xchg al,[rsi]
```

```
skip:
  inc rsi
  loop LP

call display

  mov rax,1
  mov rdi,1
  mov rsi,msg1
  mov rdx,len1
  syscall

  dispmsg result,2

  mov rax,60
  mov rdi,0
  syscall

display:
  mov rbx,rax
  mov rdi,result
  mov cx,2

up1:
  rol bl,04
  mov al,bl
  and al,0fh
  cmp al,09h
  jg add_37
  add al,30h
  jmp skip1

add_37:
  add al,37h

skip1:
```

```
  mov [rdi],al
  inc rdi
  dec cx
  jnz up1

ret
```

OUTPUT:



Assignment No. 7

PROBLEM STATEMENT :

Write an X86/64 ALP to detect protected mode and display the values of GDTR, LDTR, IDTR, TR and MSW Registers also identify CPU type using CPUID instruction.

PROGRAM:

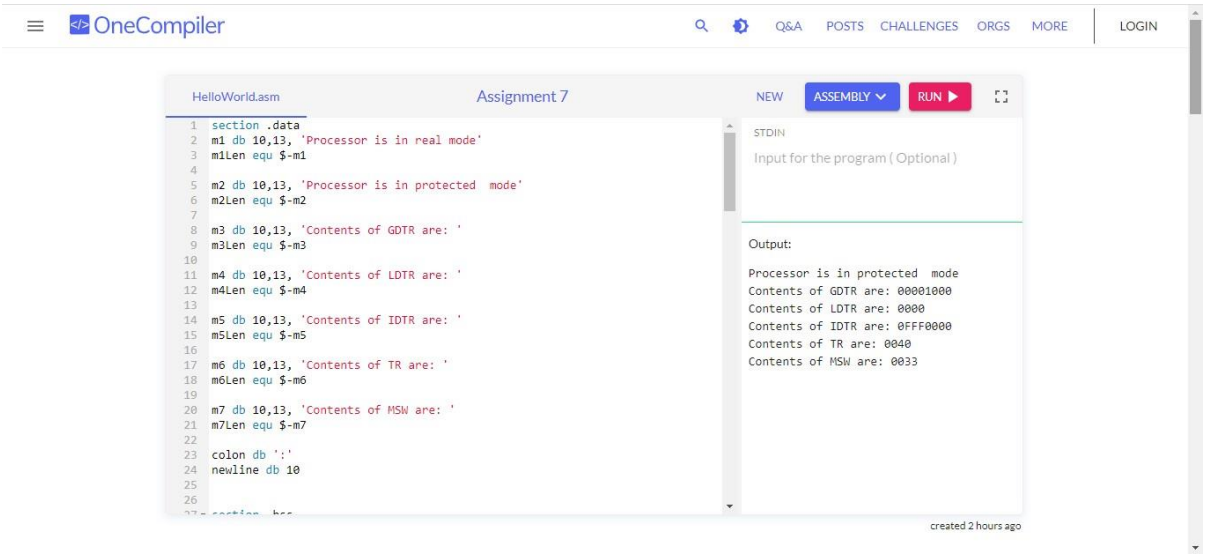
```
section .data
m1 db 10,13, 'Processor is in real mode'
m1Len equ $-m1

m2 db 10,13, 'Processor is in protected mode'
m2Len equ $-m2

m3 db 10,13, 'Contents of GDTR are: '
m3Len equ $-m3

m4 db 10,13, 'Contents of LDTR are: '
m4Len equ $-m4

m5 db 10,13, 'Contents of IDTR are: '
m5Len equ $-m5

m6 db 10,13, 'Contents of TR are: '
m6Len equ $-m6

m7 db 10,13, 'Contents of MSW are: '
m7Len equ $-m7

colon db ':'
newline db 10
```

```
section .bss
  gdt resw 1
      resd 1

  ldt resw 1

  idt resw 1
      resd 1

  tr resw 1

  msw resw 1

  disp_buff resb 8

%macro disp 2
  mov eax,04
  mov ebx,01
  mov ecx,%1
  mov edx,%2
  int 80h
%endmacro

section .text
global _start
_start:

  smsw ax
  mov[msw],ax
  bt ax,0
  jc prm
  disp m1,m1Len
  jmp exit

exit:
  mov eax,1
  mov ebx,0
```

```
  int 80h

prm:
  disp m2,m2Len

next1:
  sgdt[gdt]
  sldt[ldt]
  sidt[idt]
  str[tr]
  smsw[msw]

  disp m3,m3Len
    mov bx,[gdt+4]
    call disp4

    mov bx,[gdt+2]
    call disp4

  disp m4,m4Len
    mov bx,[ldt]
    call disp4
  disp m5,m5Len
    mov bx,[idt]
    call disp4

    mov bx,[idt+2]
    call disp4

  disp m6,m6Len
    mov bx,[tr]
    call disp4

  disp m7,m7Len
    mov bx,[msw]
    call disp4
```

```
  disp newline,1
   jmp exit

disp4:
  mov ecx,4
  mov edi, disp_buff

up1:
  rol bx,04
  mov al,bl
  and al,0fh
  cmp al,09
  jbe y
  add al,07h

y:
  add al,30h
  mov[edi],al
  inc  edi
  loop up1
  disp disp_buff,4
  ret
```

OUTPUT:



Assignment No. 8

PROBLEM STATEMENT :

Write an X86/64 ALP to perform multiplication of two 8-bit hexadecimal numbers. Use successive addition and add and shift method. Accept input from the user.

PROGRAM:

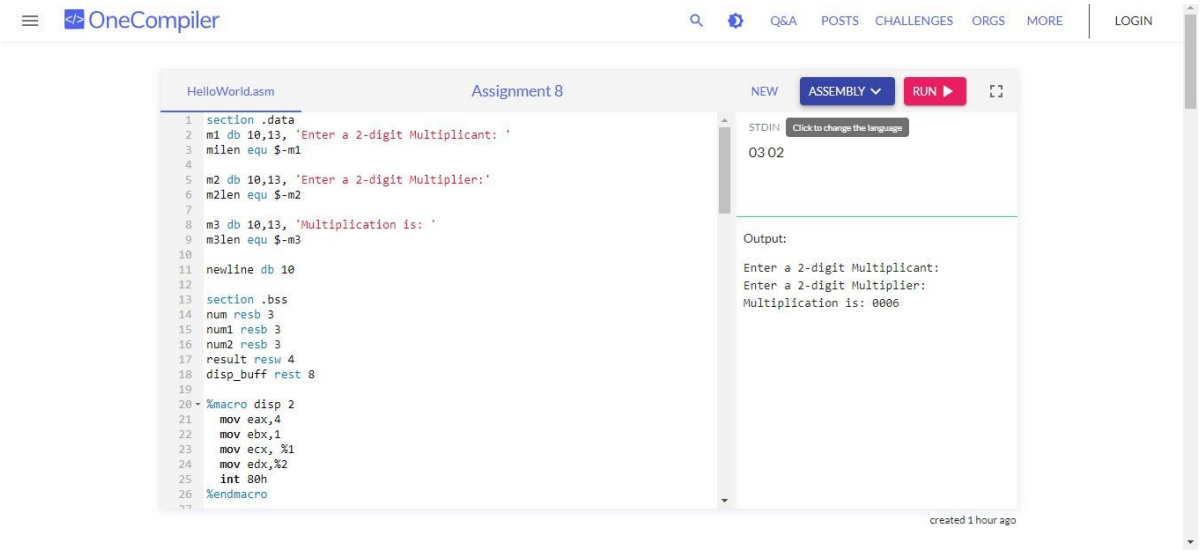
```
section .data
m1 db 10,13, 'Enter a 2-digit Multiplicant: '
milen equ $-m1

m2 db 10,13, 'Enter a 2-digit Multiplier:'
m2len equ $-m2

m3 db 10,13, 'Multiplication is: '
m3len equ $-m3

newline db 10

section .bss
num resb 3
num1 resb 3
num2 resb 3
result resw 4
disp_buff rest 8

%macro disp 2
  mov eax,4
  mov ebx,1
  mov ecx, %1
  mov edx,%2
  int 80h
%endmacro

%macro accept 2
```

```
  mov eax, 3
  mov ebx,0
  mov ecx,%1
  mov edx,%2
  int 80h
%endmacro

section .text
global _start
_start:

disp m1,milen
  accept num,3

  call convert

  mov [num1],bl

  disp m2,m2len
  accept num,3

  call convert
  mov [num2],bl

  xor rax,rax
  xor rcx, rcx

  mov al,[num1]
  mov bl,[num2]

back:
  add rcx,rax
  dec bl
  jnz back

  mov [result], rcx
  disp m3,m3len
  mov bx,[result]
  call disp2
```

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

convert:
  mov bx,00
  mov esi,num
  mov ecx,2

up:
  rol  bx,4
  mov al, [esi]
  cmp al,39h
  jbe x
  sub al,07h

x:
 sub al,30h
 add bl, al
 inc esi
 loop up
 ret

disp2:
 mov ecx,4
 mov edi,disp_buff

up1:
  rol bx,4
  mov al,bl
  and al,0fh
  cmp al,09
  jbe y
  add al,07h

y:
  add al,30h
  mov [edi], al
  inc edi
```

```
  loop up1
  disp disp_buff,4
ret
```

OUTPUT:



Assignment No. 9

PROBLEM STATEMENT :

Write an X86/64 ALP to find the factorial of a given integer number on a command line by using recursion. Explicit stack manipulation is expected in the code.

PROGRAM:

```
section .data
m1 db 10,"Enter No::"
m1len equ $-m1

m2 db 10,"Factorial is::"
m2len equ $-m2

newline db 10

section .bss
num1 resb 3
num2 resb 3
dispbuff resq 1

%macro cmn 4
  mov rax,%1
  mov rdi,%2
  mov rsi,%3
  mov rdx,%4
  syscall
%endmacro

section .text
global _start
_start:
  cmn 1,1,m1,m1len
```

```
  cmn 0,0,num1,3

  call convert
  mov [num2],bl

  cmn 1,1,m2,m2len
  xor rdx,rdx
  xor rax,rax

  mov bl,[num2]

  call proc_fact
  mov rbx,rax

  call display

exit:
  mov rax,60
  mov rdi,0
  syscall


proc_fact:

cmp bl,1
jg x
mov ax,1
ret

x:
  dec bl
  call proc_fact
  inc bl
  mul bl
  ret

display:
```

```
  mov ecx,16
  mov edi,dispbuff

again:
  rol rbx,4
  mov al,bl
  and al,0fh
  cmp al,09h
  jbe x1
  add al,09h

x1:
  add al,30h
  mov [edi],al
  inc edi
  loop again
  cmn 1,1,dispbuff,16
  cmn 1,1,newline,1
  ret

convert:
  mov rsi,num1
  mov cl,02h
  xor rax ,rax
  xor rbx,rbx

contc:
  rol bl,04h
  mov al,[rsi]
  cmp al,39h
  jbe skipc
  sub al,07h

skipc:
  sub al,30h
  add bl,al
  inc rsi
```

```
dec  cl
jnz contc
ret
```

OUTPUT: