MICROPROCESSOR LABORATORY

Journal

SEM - II

2020-21

Name :- Ojus Pravin Jaiswal

Roll No. :- SACO19108

Seat No. :- S191094290

Year :- SE

Division :- A

MICROPROCESSOR LABORATORY

Assignment No. 1

NAME :- OJUS PRAVIN JAISWAL
ROLL NO. :- SACO19108
DIVISION :- A

Assignment No. 1

Program :

```
%macro read_or_print 4
mov Rax, %1
mov Rdi, %2
mov Rsi, %3
mov Rdx, %4
syscall
%endmacro

%macro exit 0
mov rax,60
mov rdi,0
syscall
%endmacro

section .bss
array resd 200
counter resb 1

section .text
global _start
_start:

; Accept numbers in an array
mov byte[counter],09
mov rsi,array
loop:
read_or_print 0,0,rsi,17
add Rsi,17
dec byte[counter]
jnz loop

;display Contents of Array
mov byte [counter],09
mov rsi, array
loop1:
read_or_print 1,1,rsi,17
add Rsi, 17
dec byte [counter]
jnz loop1
exit
```

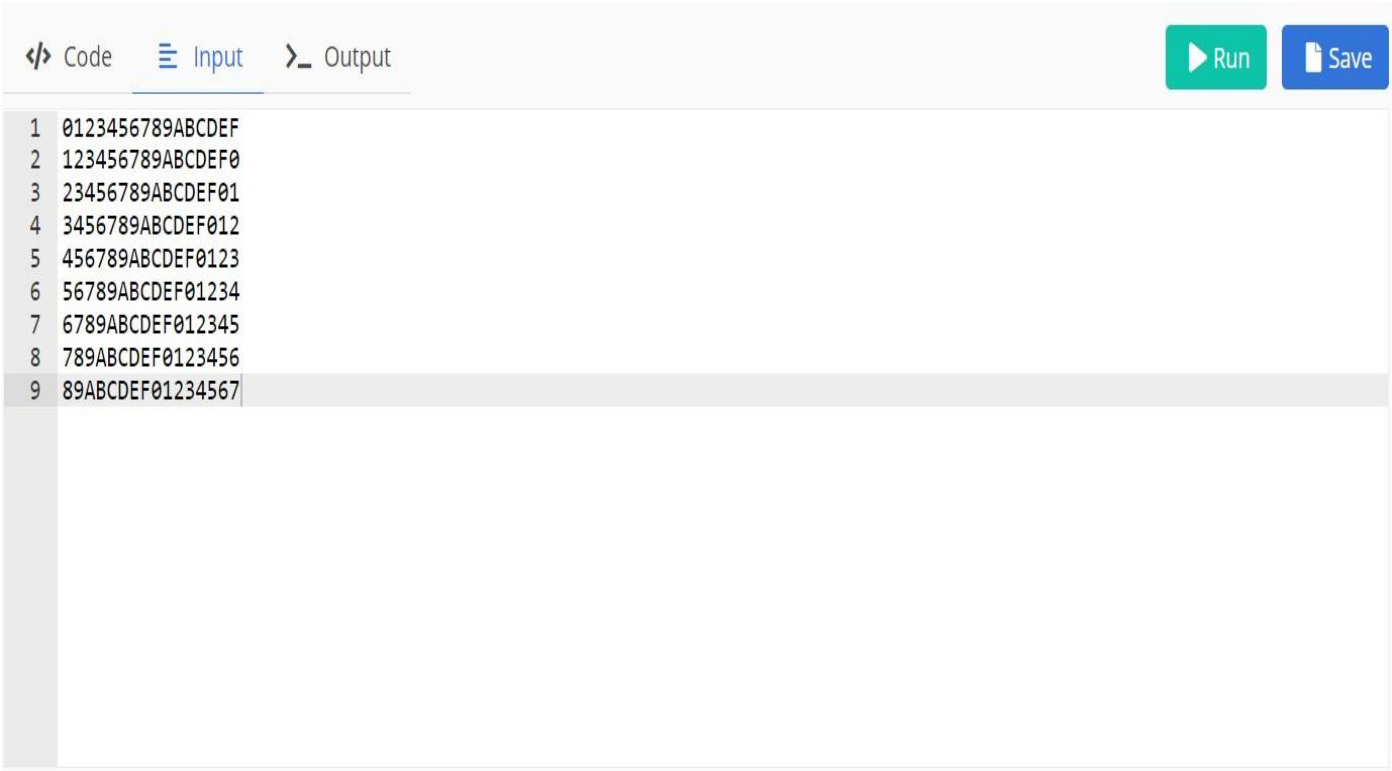
Code Input Output Run Save

```
%macro read_or_print 4
mov Rax, %1
mov Rdi, %2
mov Rsi, %3
mov Rdx, %4 syscall
%endmacro

%macro exit 0 mov rax,60 mov rdi,0 syscall
%endmacro

section .bss array resd 200
counter resb 1

section .text global _start
_start:

; Accept numbers in an array mov byte[counter],09
mov rsi,array loop:
read_or_print 0,0,rsi,17 add Rsi,17
dec byte[counter] jnz loop
```

Code Input Output Run Save

```
%macro exit 0 mov rax,60 mov rdi,0 syscall
%endmacro

section .bss array resd 200
counter resb 1

section .text global _start
_start:

; Accept numbers in an array mov byte[counter],09
mov rsi,array loop:
read_or_print 0,0,rsi,17 add Rsi,17
dec byte[counter] jnz loop

;display Contents of Array mov byte [counter],09 mov rsi, array
loop1:
read_or_print 1,1,rsi,17 add Rsi, 17
dec byte [counter] jnz loop1
exit
```

Input :



Output :

MICROPROCESSOR LABORATORY

ASSIGNMENT NO. 2

Name :- Ojus Pravin Jaiswal

Roll No. :- SACO19108

Division :- A

Assignment No. 2

Program :

```
%macro read_or_print 4
mov Rax, %1
mov Rdi, %2
mov Rsi, %3
mov Rdx, %4
syscall
%endmacro

%macro exit 0
mov rax,60
mov rdi,0
syscall
%endmacro

section .data
msg db 10,13,"Length of the String is:",10,13
msglen equ $-msg

section .bss
str1 resb 200
result resb 1
```

```
section .text
global _start
_start:

 read_or_print 0,0,str1,200
 call display
 exit

display:
        mov rsi,result+15
        mov rcx,16
 loop2:  mov rdx,0
        mov rbx,16
        div rbx
        cmp dl,09h
        jbe skip2
        add dl,07h
 skip2:  add dl,30h
        mov [rsi],dl
        dec rsi
        dec rcx
        jnz loop2
 read_or_print 1,1,msg, msglen
 read_or_print 1,1,result,16
 ret
```

Code Input Output Run Save

```
%macro read_or_print 4
mov Rax, %1
mov Rdi, %2
mov Rsi, %3
mov Rdx, %4
syscall
%endmacro

%macro exit 0
mov rax,60
mov rdi,0
syscall
%endmacro

section .data
msg db 10,13,"Length of the String is:",10,13
msglen equ $-msg

section .bss
str1 resb 200
```

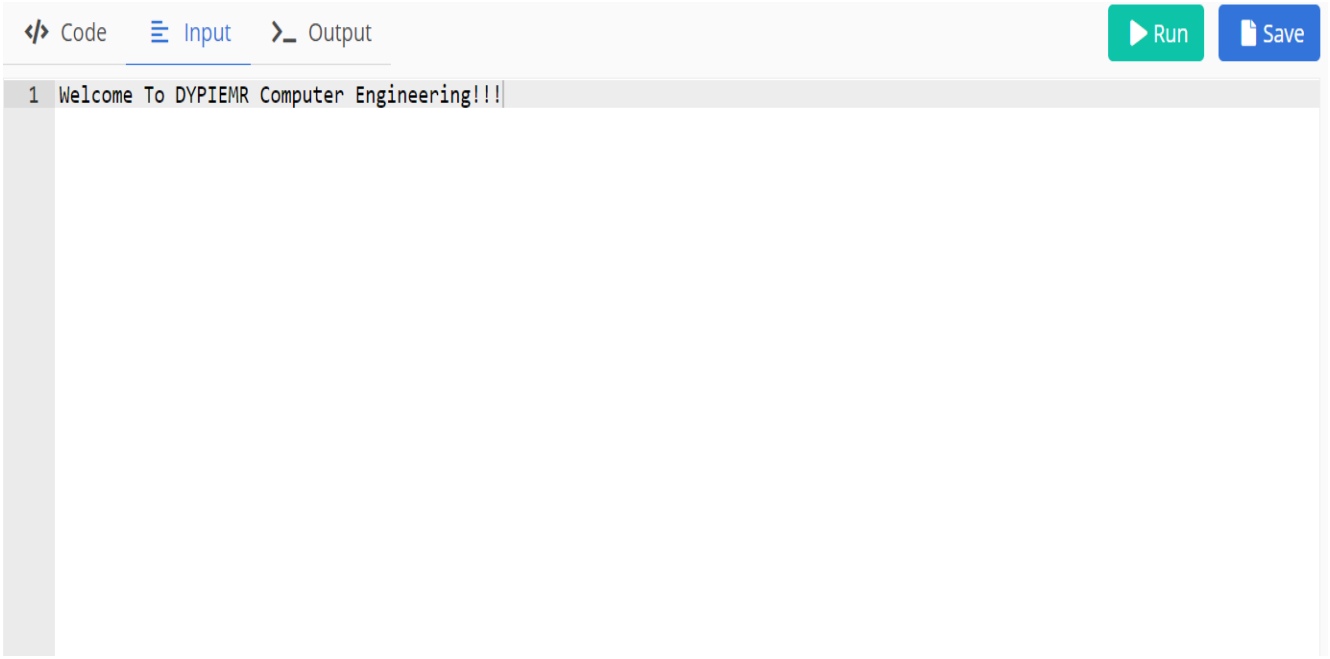
Code Input Output Run Save

```
result resb 17

section .text
global _start
_start:

read_or_print 0,0,str1,200
call display
exit

display:
mov rsi,result+15
mov rcx,16
loop2: mov rdx,0
       mov rbx,16
       div rbx
       cmp dl,09h
       add dl,07h
skip2: add dl,30h
       dec rsi
```

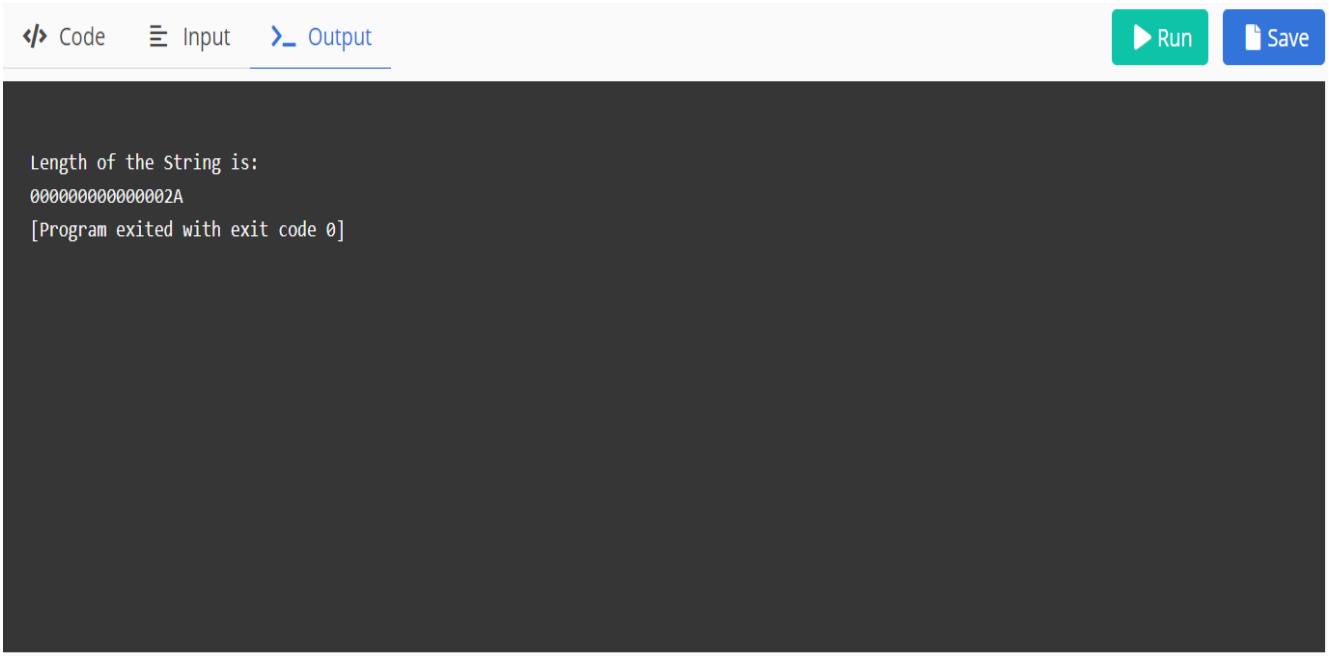
Code Input Output Run Save

```

read_or_print 0,0,str1,200
call display
exit

display:
mov rsi,result+15
mov rcx,16
loop2: mov rdx,0
       mov rbx,16
       div rbx
       cmp dl,09h
       add dl,07h
skip2: add dl,30h
       dec rsi
       dec rcx
       jnz loop2
read_or_print 1,1,msg,msglen
read_or_print 1,1,result,16
ret
```

Input :



Output :



MICROPROCESSOR LABORATORY

Assignment No. 3

Name :- Ojus Pravin Jaiswal
Roll No. :- SACO19108
Division :- A

Assignment No. 3

Program :

```
%macro read_or_print 4
mov Rax, %1
mov Rdi, %2
mov Rsi, %3
mov Rdx, %4
syscall
%endmacro

%macro exit 0
mov rax,60
mov rdi,0
syscall
%endmacro

section .data
array db 00h,10h,20h,30h,40h,50h,60h,70h
msg1 db 10,13,"The smallest element in the array is : ",10,13
msglen1 equ $-msg1
msg2 db 10,13,"The largest element in the array is : ",10,13
msglen2 equ $-msg2
```

```
section .bss
cnt resb 1
result resb 16

section .text
global _start
_start:

;Smallest Number
mov byte[cnt],08
mov rsi, array
mov al,0
up0:cmp al,[rsi]
      jl skip0
      xchg al,[rsi]
skip0:inc rsi
         dec byte[cnt]
         jnz up0

call display1

;Largest Number
mov byte[cnt],08
mov rsi,array
mov al,0
up1:cmp al,[rsi]
      jg skip1
```

```
        xchg al,[rsi]
skip1:inc rsi
            dec byte[cnt]
            jnz up1

call display2
exit

display1:
mov rsi,result+15
mov rcx,16
loop2:mov rdx,0
            mov rbx,16
            div rbx
            cmp dl,09h
            jbe skip2
            add dl,07h
skip2:add dl,30h
            mov [rsi],dl
            dec rsi
            dec rcx
            jnz loop2
read_or_print 1,1,msg1,msglen1
read_or_print 1,1,result,16
ret
```

```
display2:
mov rsi,result+15
mov rcx,16
loop3:mov rdx,0
        mov rbx,16
        div rbx
        cmp dl,09h
        jbe skip3
        add dl,07h
skip3:add dl,30h
        mov [rsi],dl
        dec rsi
        dec rcx
        jnz loop3
read_or_print 1,1,msg2,msglen2
read_or_print 1,1,result,16
ret
```

Code Input Output Run Save

```
%macro read_or_print 4
mov Rax, %1
mov Rdi, %2
mov Rsi, %3
mov Rdx, %4
syscall
%endmacro

%macro exit 0
mov rax,60
mov rdi,0
syscall
%endmacro

section .data
array db 00h,10h,20h,30h,40h,50h,60h,70h
msg1 db 10,13,"The smallest element in the array is : ",10,13
msglen1 equ $-msg1
msg2 db 10,13,"The largest element in the array is : ",10,13
msglen2 equ $-msg2
```

Code Input Output Run Save

```
section .bss
cnt resb 1
result resb 16

section .text
global _start
_start:

;Smallest Number
mov byte[cnt],08
mov rsi, array
mov al,0
up0:cmp al,[rsi]
    jl skip0
    xchg al,[rsi]
skip0:inc rsi
      dec byte[cnt]
      jnz up0

```

Code Input Output Run Save

```
call display1

;Largest Number
mov byte[cnt],08
mov rsi,array
mov al,0
up1:cmp al,[rsi]
    jg skip1
    xchg al,[rsi]
skip1:inc rsi
      dec byte[cnt]
      jnz up1

call display2
exit

display1:
mov rsi,result+15
mov rcx,16
loop2:mov rdx,0
```

Code Input Output Run Save

```
      mov rbx,16
      div rbx
      cmp dl,09h
      jbe skip2
      add dl,07h
skip2:add dl,30h
      mov [rsi],dl
      dec rsi
      dec rcx
      jnz loop2
read_or_print 1,1,msg1,msglen1
read_or_print 1,1,result,16
ret

display2:
mov rsi,result+15
mov rcx,16
loop3:mov rdx,0
      mov rbx,16
      div rbx
```

Code Input Output Run Save

```
72  read_or_print 1,1,result,16
73  ret
74
75  display2:
76  mov rsi,result+15
77  mov rcx,16
78  loop3:mov rdx,0
79        mov rbx,16
80        div rbx
81        cmp dl,09h
82        jbe skip3
83        add dl,07h
84  skip3:add dl,30h
85        mov [rsi],dl
86        dec rsi
87        dec rcx
88        jnz loop3
89  read_or_print 1,1,msg2,msglen2
90  read_or_print 1,1,result,16
91  ret
```

Output :

Code Input Output Run Save

```
The smallest element in the array is :
0000000000000000
The largest element in the array is :
0000000000000070
[Program exited with exit code 0]
```

MICROPROCESSOR LABORATORY

Assignment No. 4

NAME :- OJUS PRAVIN JAISWAL
ROLL NO. :- SACO19108
DIVISION :- A

Assignment No. 4

Program :

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

%macro exit 0
   mov rax, 60
   mov rdi,0
   syscall
%endmacro

section .data
   arr dq
0000000000000100h,0000000000000004h,000000000000003h,000000000000002h,000000000000001h
   n equ 5

   menu db 10d,13d,"**********MENU**********"
      db 10d,13d,"1. Addition"
      db 10d,13d,"2. Subtraction"
      db 10d,13d,"3. Multiplication"
```

```
        db 10d,13d,"4. Division"
        db 10d,13d,"5. Exit"
        db 10d,13d,"Enter your Choice : "
    menu_len equ $-menu

    m1 db 10d,13d,"Addition : "
    l1 equ $-m1
    m2 db 10d,13d,"Subtraction : "
    l2 equ $-m2
    m3 db 10d,13d,"Multiplication : "
    l3 equ $-m3
    m4 db 10d,13d,"Division : "
    l4 equ $-m4

section .bss
    answer resb 16
    choice resb 2

section .text
global _start:
_start:

    up:scall 1,1,menu,menu_len
       scall 0,0,choice,2

    cmp byte[choice],'1'
    je case1
    cmp byte[choice],'2'
```

```
    je case2
    cmp byte[choice],'3'
    je case3
    cmp byte[choice],'4'
    je case4
    cmp byte[choice],'5'
    je case5

    case1: scall 1,1,m1,l1
        call addition
        jmp up

    case2: scall 1,1,m2,l2
        call subtraction
        jmp up

    case3: scall 1,1,m3,l3
        call multiplication
        jmp up

    case4: scall 1,1,m4,l4
        call division
        jmp up

    case5:exit

addition:
    mov rcx,n
```

```
    dec rcx
    mov rsi,arr
    mov rax,[rsi]
up1:add rsi,8
    mov rbx,[rsi]
    add rax,rbx
    loop up1
    call display
ret

subtraction:
    mov rcx,n
    dec rcx
    mov rsi,arr
    mov rax,[rsi]
up2:add rsi,8
    mov rbx,[rsi]
    sub rax,rbx
    loop up2
    call display
ret

multiplication:
    mov rcx,n
    dec rcx
    mov rsi,arr
    mov rax,[rsi]
up3:add rsi,8
```

```
    mov rbx,[rsi]
    mul rbx
    loop up3
    call display
ret

division:
    mov rcx,n
    dec rcx
    mov rsi,arr
    mov rax,[rsi]
up4:add rsi,8
    mov rbx,[rsi]
    mov rdx,0
    div rbx
    loop up4
    call display
ret

display:
    mov rsi,answer+15
    mov rcx,16
cnt:mov rdx,0
    mov rbx,16
    div rbx
    cmp dl,09h
    jbe add30
    add dl,07h
```

add30:add dl,30h

mov [rsi],dl

dec rsi

dec rcx

jnz cnt

scall 1,1,answer,16

ret

Code Input Output Run Save

```
section .text
global _start:
_start:

    up:scall 1,1,menu,menu_len
       scall 0,0,choice,2

    cmp byte[choice],'1'
    je case1
    cmp byte[choice],'2'
    je case2
    cmp byte[choice],'3'
    je case3
    cmp byte[choice],'4'
    je case4
    cmp byte[choice],'5'
    je case5

    case1: scall 1,1,m1,l1
           call addition
```

Code Input Output Run Save

```
           jmp up

    case2: scall 1,1,m2,l2
           call subtraction
           jmp up

    case3: scall 1,1,m3,l3
           call multiplication
           jmp up

    case4: scall 1,1,m4,l4
           call division
           jmp up

    case5:exit

addition:
    mov rcx,n
    dec rcx
    mov rsi,arr
```

Code Input Output Run Save

```
    mov rax,[rsi]
up1:add rsi,8
    mov rbx,[rsi]
    add rax,rbx
    loop up1
    call display
ret

subtraction:
    mov rcx,n
    dec rcx
    mov rsi,arr
    mov rax,[rsi]
up2:add rsi,8
    mov rbx,[rsi]
    sub rax,rbx
    loop up2
    call display
ret
```

Code Input Output Run Save

```
multiplication:
    mov rcx,n
    dec rcx
    mov rsi,arr
    mov rax,[rsi]
up3:add rsi,8
    mov rbx,[rsi]
    mul rbx
    loop up3
    call display
ret

division:
    mov rcx,n
    dec rcx
    mov rsi,arr
    mov rax,[rsi]
up4:add rsi,8
    mov rbx,[rsi]
    mov rdx,0
```

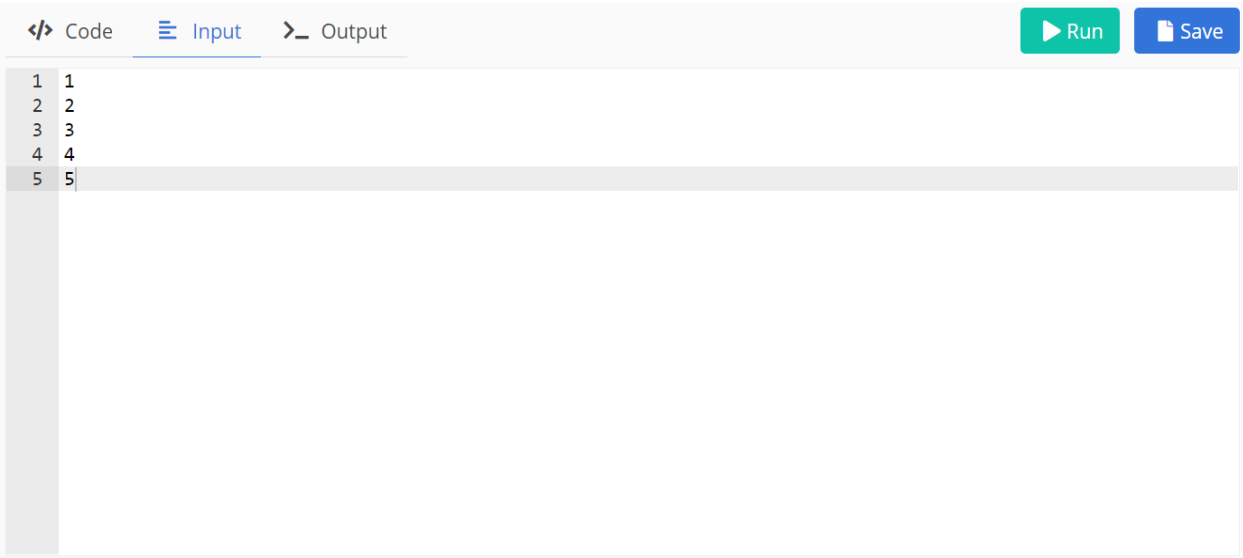
Code Input Output Run Save

```
    div rbx
    loop up4
    call display
ret

display:
    mov rsi,answer+15
    mov rcx,16
cnt:mov rdx,0
    mov rbx,16
    div rbx
    cmp dl,09h
    jbe add30
    add dl,07h
add30:add dl,30h
    mov [rsi],dl
    dec rsi
    dec rcx
    jnz cnt
    scall 1,1,answer,16
```

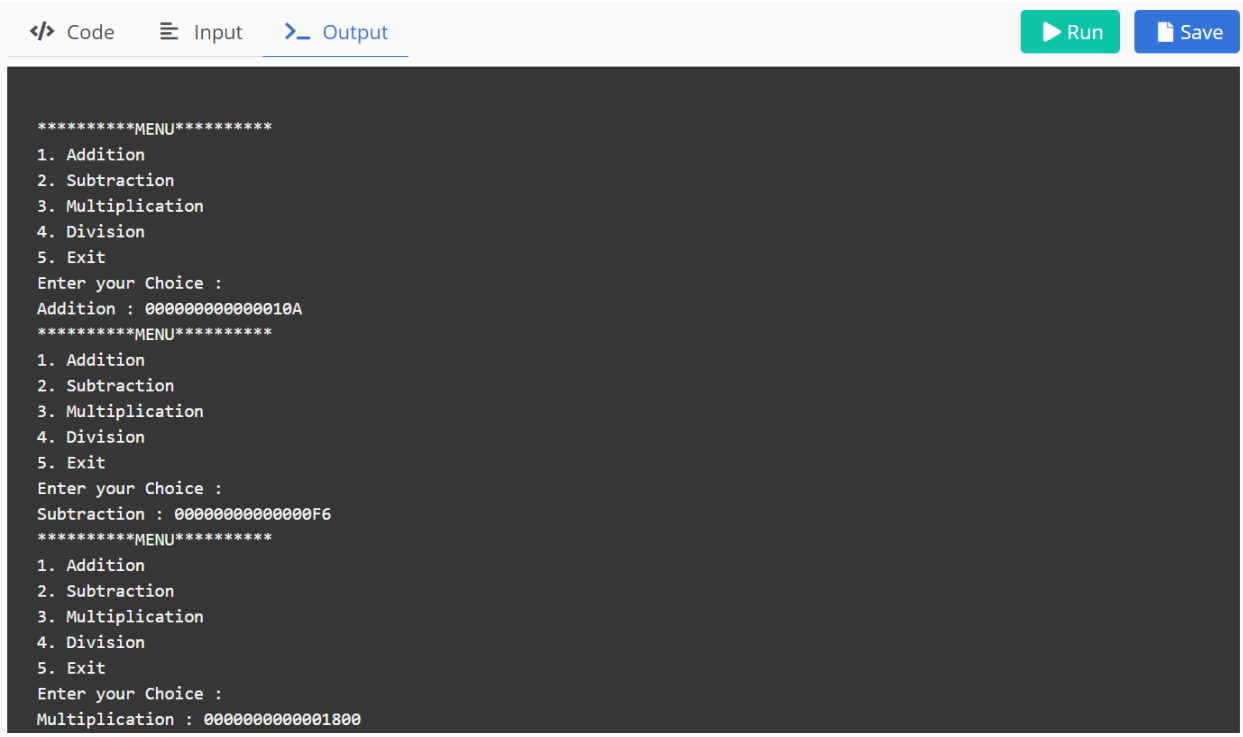
Code Input Output Run Save

```
    loop up4
    call display
ret

display:
    mov rsi,answer+15
    mov rcx,16
cnt:mov rdx,0
    mov rbx,16
    div rbx
    cmp dl,09h
    jbe add30
    add dl,07h
add30:add dl,30h
    mov [rsi],dl
    dec rsi
    dec rcx
    jnz cnt
    scall 1,1,answer,16
ret
```

Input :



Output :



Code Input Output Run Save

```
5. Exit
Enter your Choice :
Multiplication : 0000000000001800
**********MENU**********
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit
Enter your Choice :
Division : 000000000000000A
**********MENU**********
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit
Enter your Choice :
[Program exited with exit code 0]
```

MICROPROCESSOR LABORATORY

Assignment No. 5

NAME :- OJUS PRAVIN JAISWAL
ROLL NO. :- SACO19108
DIVISION :- A

Assignment No. 5

Program :

```
%macro print 2
mov Rax,1
mov  Rdi,1
mov Rsi,%1
mov Rdx,%2
syscall
%endmacro

%macro exit 0
mov rax,60
mov rdi,0
syscall
%endmacro

section .data
arr dq 00h,-10h,20h,-30h,40h,-50h,60h,-70h
n equ 8
pmsg db 10,13,"The count of positive elements in the array is : ",10,13
pmsglen equ $-pmsg
nmsg db 10,13,"The count of negative element in the array is : ",10,13
nmsglen equ $-nmsg
nwline db 10,13

section .bss
```

```
pcnt resq 1
ncnt resq 1
char_answer resb 16

section .text
global _start
_start:
mov rsi,arr
mov rdi,n
mov rbx,0
mov rcx,0

up:mov rax,[rsi]
    rol rax,1
    jc negative

positive:inc rbx
            jmp next

negative:inc rcx

next:add rsi,8
       dec rdi
       jnz up

       mov [pcnt],rbx
       mov [ncnt],rcx
```

```
        print pmsg,pmsglen
        mov rax,[pcnt]
        call display

        print nmsg,nmsglen
        mov rax,[ncnt]
        call display

        print nwline,1
        exit

display:
 mov rsi,char_answer+15
 mov rcx,16

 cnt:mov rdx,0
       mov rbx,16h
       div rbx
       cmp dl,09h
       jbe add30
       add dl,07h
 add30:add dl,30h
            mov [rsi],dl
            dec rsi
            dec rcx
            jnz cnt
 print char_answer,16
ret
```

Code Input Output Run Save

```
%macro print 2
mov Rax,1
mov Rdi,1
mov Rsi,%1
mov Rdx,%2
syscall
%endmacro

%macro exit 0
mov rax,60
mov rdi,0
syscall
%endmacro

section .data
arr dq 00h,-10h,20h,-30h,40h,-50h,60h,-70h
n equ 8
pmsg db 10,13,"The count of positive elements in the array is : ",10,13
pmsglen equ $-pmsg
nmsg db 10,13,"The count of negative element in the array is : ",10,13
```

Code Input Output Run Save

```
nmsglen equ $-nmsg
nwline db 10,13

section .bss
pcnt resq 1
ncnt resq 1
char_answer resb 16

section .text
global _start
_start:

mov rsi,arr
mov rdi,n
mov rbx,0
mov rcx,0

up:mov rax,[rsi]
   rol rax,1
   jc negative
```

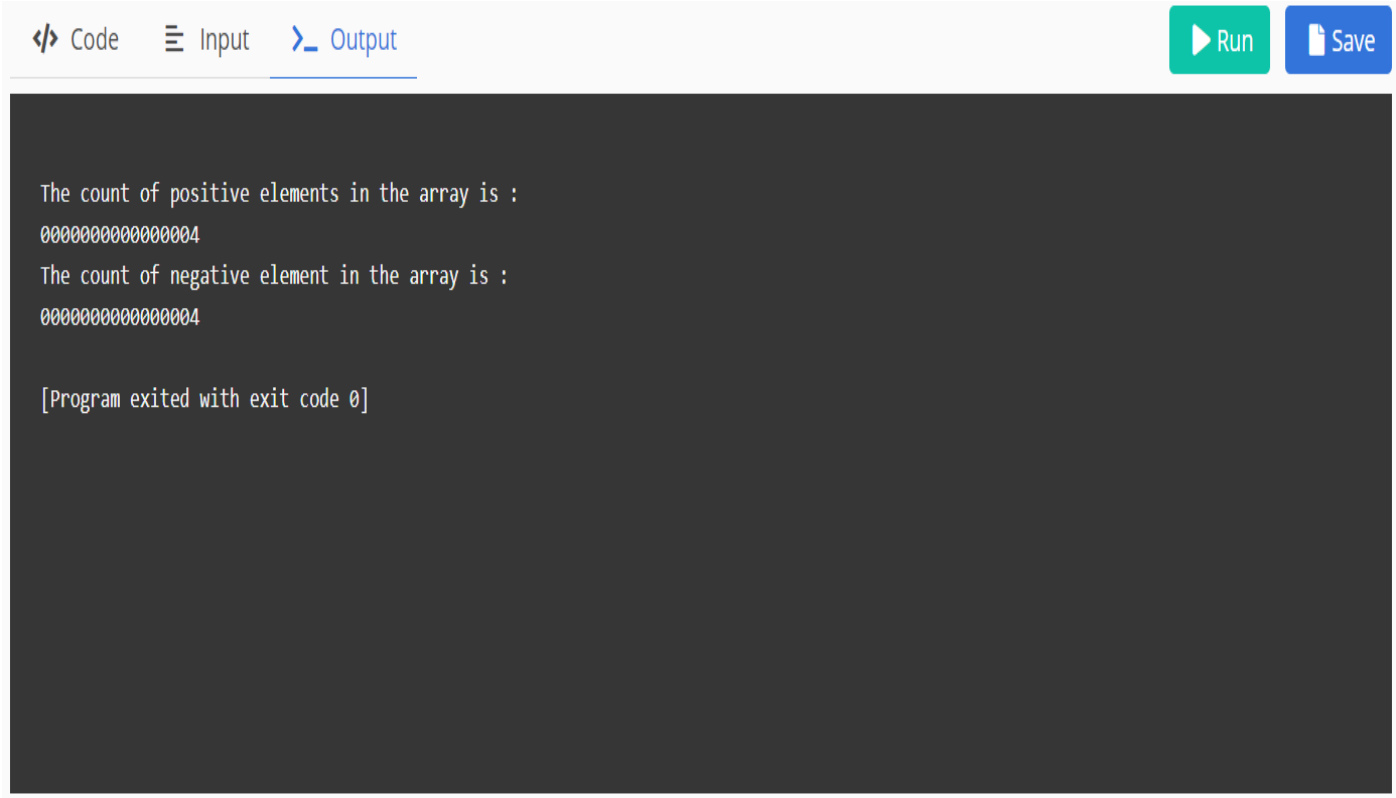
Code Input Output Run Save

```

positive:inc rbx
        jmp next

negative:inc rcx

next:add rsi,8
     dec rdi
     jnz up

mov [pcnt],rbx
mov [ncnt],rcx

print pmsg,pmsglen
mov rax,[pcnt]
call display

print nmsg,nmsglen
mov rax,[ncnt]
call display
```

Code Input Output Run Save

```
61
62  print nwline,1
63  exit
64
65  display:
66  mov rsi,char_answer+15
67  mov rcx,16
68  cnt:mov rdx,0
69      mov rbx,16h
70      div rbx
71      cmp dl,09h
72      jbe add30
73      add dl,07h
74  add30:add dl,30h
75        mov [rsi],dl
76        dec rsi
77        dec rcx
78        jnz cnt
79  print char_answer,16
80  ret
```

Output :



MICROPROCESSOR LABORATORY

ASSIGNMENT NO. 6

Name :- Ojus Pravin Jaiswal
Roll No. :- SACO19108
Division :- A

Assignment No. 6

Program :

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

section .data

   rmodemsg db 10,"Processor is in Real Mode"
   rmsg_len:equ $-rmodemsg

   pmodemsg db 10,"Processor is in Protected Mode"
   pmsg_len:equ $-pmodemsg

   gdtmsg db 10,"GDT Contents are :: "
   gmsg_len:equ $-gdtmsg

   ldtmsg db 10,"LDT Contents are :: "
   lmsg_len:equ $-ldtmsg

   idtmsg db 10,"IDT Contents are :: "
   imsg_len:equ $-idtmsg
```

```
    trmsg db 10,"Task Register Contents are :: "
    tmsg_len:equ $-trmsg

    mswmsg db 10,"Machine Status Word :: "
    mmsg_len:equ $-mswmsg

    promsg db 10,"Processor Information :: "
    promsg_len:equ $-promsg

    colmsg db ':'

    newline db 10

section .bss
    gdt resd 1
        resw 1
    ldt resw 1
    idt resd 1
        resw 1
    tr resw 1
    cr0_data resd 1
    dnum_buff resb 04

section .text
global _start:
_start:
```

```
    smsw eax
    mov [cr0_data],eax
    bt eax,0
    jc prmode
    disp rmodemsg,rmsg_len
    jmp nxt1

prmode:disp pmodemsg,pmsg_len

nxt1:sgdt[gdt]
     sldt[ldt]
     sidt[idt]
     str[tr]

     disp gdtmsg,gmsg_len

     mov bx,[gdt+4]
     call disp_num

     mov bx,[gdt+2]
     call disp_num

     disp colmsg,1

     mov bx,[gdt]
     call disp_num

     disp ldtmsg,lmsg_len
```

```
mov bx,[ldt]
call disp_num

disp idtmsg,imsg_len

mov bx,[idt+4]
call disp_num

mov bx,[idt+2]
call disp_num

disp colmsg,1

mov bx,[idt]
call disp_num

disp trmsg,tmsg_len

mov bx,[tr]
call disp_num

disp mswmsg,mmsg_len

mov bx,[cr0_data+2]
call disp_num

mov bx,[cr0_data]
call disp_num
```

```
    disp newline,1

    disp promsg,promsg_len

    mov eax,00h
    call disp_num
    cpuid
    call disp_num

exit: mov eax,01
     mov ebx,00
     int 80h

disp_num:
   mov esi,dnum_buff
   mov ecx,04
up1:rol bx,4
   mov dl,bl
   and dl,0fh
   add dl,30h
   cmp dl,39h
   jbe skip1
   add dl,07h
skip1:mov [esi],dl
     inc esi
     loop up1
disp dnum_buff,4
```

ret

Code Input Output Run Save

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

section .data

    rmodemsg db 10,"Processor is in Real Mode"
    rmsg_len:equ $-rmodemsg

    pmodemsg db 10,"Processor is in Protected Mode"
    pmsg_len:equ $-pmodemsg

    gdtmsg db 10,"GDT Contents are :: "
    gmsg_len:equ $-gdtmsg

    ldtmsg db 10,"LDT Contents are :: "
```

Code Input Output Run Save

```
    lmsg_len:equ $-ldtmsg

    idtmsg db 10,"IDT Contents are :: "
    imsg_len:equ $-idtmsg

    trmsg db 10,"Task Register Contents are :: "
    tmsg_len:equ $-trmsg

    mswmsg db 10,"Machine Status Word :: "
    mmsg_len:equ $-mswmsg

    promsg db 10,"Processor Information :: "
    promsg_len:equ $-promsg

    colmsg db ':'

    newline db 10

section .bss
    gdt resd 1
```

Code Input Output Run Save

```
41          resw 1
42      ldt resw 1
43      idt resd 1
44          resw 1
45      tr resw 1
46      cr0_data resd 1
47      dnum_buff resb 04
48
49  section .text
50  global _start:
51  _start:
52
53      smsw eax
54      mov [cr0_data],eax
55      bt eax,0
56      jc prmode
57      disp rmodemsg,rmsg_len
58      jmp nxt1
59
60  prmode:disp pmodemsg,pmsg_len
```

Code Input Output Run Save

```
61
62  nxt1:sgdt[gdt]
63       sldt[ldt]
64       sidt[idt]
65       str[tr]
66
67       disp gdtmsg,gmsg_len
68
69       mov bx,[gdt+4]
70       call disp_num
71
72       mov bx,[gdt+2]
73       call disp_num
74
75       disp colmsg,1
76
77       mov bx,[gdt]
78       call disp_num
79
80       disp ldtmsg,lmsg_len
```

Code Input Output Run Save

```
81       mov bx,[ldt]
82       call disp_num
83
84       disp idtmsg,imsg_len
85
86       mov bx,[idt+4]
87       call disp_num
88
89       mov bx,[idt+2]
90       call disp_num
91
92       disp colmsg,1
93
94       mov bx,[idt]
95       call disp_num
96
97       disp trmsg,tmsg_len
98
99       mov bx,[tr]
100      call disp_num
```

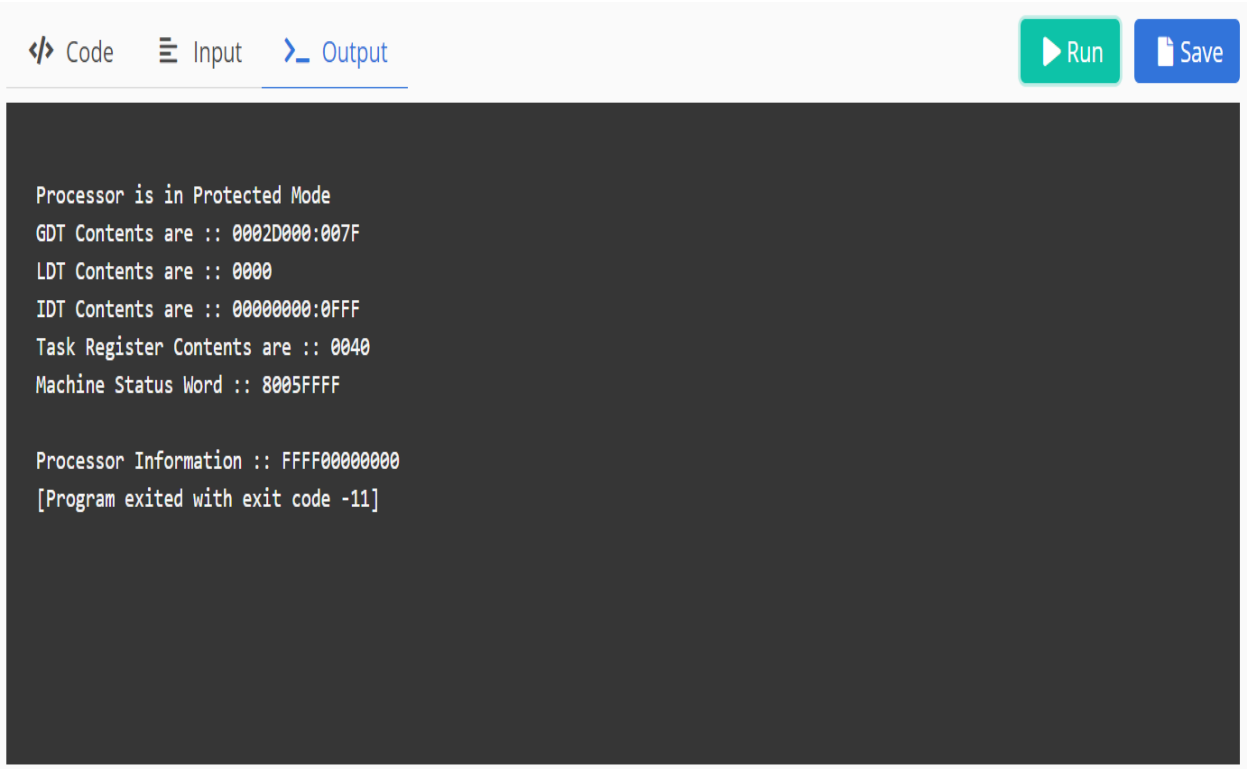
Code Input Output Run Save

```
101
102      disp mswmsg,mmsg_len
103
104      mov bx,[cr0_data+2]
105      call disp_num
106
107      mov bx,[cr0_data]
108      call disp_num
109
110      disp newline,1
111
112      disp promsg,promsg_len
113
114      mov eax,00h
115      call disp_num
116      cpuid
117      call disp_num
118
119  exit: mov eax,01
120        mov ebx,00
```

Code Input Output Run Save

```
118
119  exit: mov eax,01
120        mov ebx,00
121        int 80h
122
123  disp_num:
124      mov esi,dnum_buff
125      mov ecx,04
126  up1:rol bx,4
127      mov dl,bl
128      and dl,0fh
129      add dl,30h
130      cmp dl,39h
131      jbe skip1
132      add dl,07h
133  skip1:mov [esi],dl
134        inc esi
135        loop up1
136  disp dnum_buff,4
137  ret
```

Output :


Code
Input
Output
Run
Save
Processor is in Protected Mode
GDT Contents are :: 0002D000:007F
LDT Contents are :: 0000
IDT Contents are :: 00000000:0FFF
Task Register Contents are :: 0040
Machine Status Word :: 8005FFFF
Processor Information :: FFFF00000000
[Program exited with exit code -11]

MICROPROCESSOR LABORATORY

Assignment No. 7

NAME :- OJUS PRAVIN JAISWAL
ROLL NO. :- SACO19108
DIVISION :- A

Assignment No. 7

Program :

```
;Non Overlapped Block Transfer

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

%macro exit 0
mov rax,60
mov rdi,0
syscall
%endmacro

section .data
sblock db 10h,20h,30h,40h,50h
dblock times 5 db 0

msg1 db 10,13,"Before Non Overlapped Block Transfer :- ",10,13
msg1_len equ $-msg1
```

```
msg2 db 10,13,"After Non Overlapped Block Transfer :- ",10,13
msg2_len equ $-msg2

new_line db 10,13

smsg db 10, "Source Block is : "
smsg_len equ $-smsg

dmsg db 10,"Destination Block is : "
dmsg_len equ $-dmsg

space db " "

section .bss
Char_ans resb 2

Section .text
global _start
_start:

print msg1,msg1_len
print smsg,smsg_len
mov rsi,sblock
call block_display

print dmsg,dmsg_len
mov rsi,dblock
call block_display
```

```
call block_transfer

print new_line,1
print msg2,msg2_len
print smsg,smsg_len
mov rsi,sblock
call block_display

print dmsg,dmsg_len
mov rsi,dblock
call block_display
print new_line,1

exit

block_transfer:
mov rsi,sblock
mov rdi,dblock
mov rcx,5
up:
mov al,[rsi]
mov [rdi],al
inc rsi
inc rdi
dec rcx
jnz up
ret
```

```
block_display:
mov rbp,5
next:
mov al,[rsi]
push rsi
call display
print space,1
pop rsi
inc rsi
dec rbp
jnz next
ret

display:
Mov rsi,Char_ans+1
mov rcx,2
mov rbx,16
up1:
xor rdx,rdx
div rbx
cmp dl,09
jbe add30
add dl,07h
add30:
add dl,30h
mov [rsi],dl
dec rsi
```

dec rcx

jnz up1

print Char_ans,2

ret

Code Input Output Run Save

```
;Non Overlapped Block Transfer

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

%macro exit 0
mov rax,60
mov rdi,0
syscall
%endmacro

section .data
sblock db 10h,20h,30h,40h,50h
dblock times 5 db 0

```

Code Input Output Run Save

```
msg1 db 10,13,"Before Non Overlapped Block Transfer :- ",10,13
msg1_len equ $-msg1

msg2 db 10,13,"After Non Overlapped Block Transfer :- ",10,13
msg2_len equ $-msg2

new_line db 10,13

smsg db 10, "Source Block is : "
smsg_len equ $-smsg

dmsg db 10,"Destination Block is : "
dmsg_len equ $-dmsg

space db " "

section .bss
Char_ans resb 2

Section .text
```

Code Input Output Run Save

```
global _start
_start:

print msg1,msg1_len
print smsg,smsg_len
mov rsi,sblock
call block_display

print dmsg,dmsg_len
mov rsi,dblock
call block_display

call block_transfer

print new_line,1
print msg2,msg2_len
print smsg,smsg_len
mov rsi,sblock
call block_display

```

Code Input Output Run Save

```
print dmsg,dmsg_len
mov rsi,dblock
call block_display
print new_line,1

exit

block_transfer:
mov rsi,sblock
mov rdi,dblock
mov rcx,5
up:
mov al,[rsi]
mov [rdi],al
inc rsi
inc rdi
dec rcx
jnz up
ret

```

Code Input Output Run Save

```
block_display:
mov rbp,5
next:
mov al,[rsi]
push rsi
call display
print space,1
pop rsi
inc rsi
dec rbp
jnz next
ret

display:
Mov rsi,Char_ans+1
mov rcx,2
mov rbx,16
up1:
xor rdx,rdx
div rbx
```

Code Input Output Run Save

```
display:
Mov rsi,Char_ans+1
mov rcx,2
mov rbx,16
up1:
xor rdx,rdx
div rbx
cmp dl,09
jbe add30
add dl,07h
add30:
add dl,30h
mov [rsi],dl
dec rsi
dec rcx
jnz up1
print Char_ans,2
ret
```

Output :

Code Input Output Run Save

```
Before Non Overlapped Block Transfer :-

Source Block is : 10 20 30 40 50
Destination Block is : 00 00 00 00 00

After Non Overlapped Block Transfer :-

Source Block is : 10 20 30 40 50
Destination Block is : 10 20 30 40 50

[Program exited with exit code 0]
```

MICROPROCESSOR LABORATORY

ASSIGNMENT NO. 8

NAME :- OJUS PRAVIN JAISWAL
ROLL NO. :- SACO19108
DIVISION :- A

Assignment No. 8

Program :

```
;Overlapped Block Transfer

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

%macro exit 0
mov rax,60
mov rdi,0
syscall
%endmacro

section .data
sblock db 10h,20h,30h,40h,50h
dblock times 5 db 0

msg1 db 10,13,"Before Overlapped Block Transfer :- ",10,13
msg1_len equ $-msg1
```

```
msg2 db 10,13,"After Overlapped Block Transfer :- ",10,13
msg2_len equ $-msg2

new_line db 10,13

smsg db 10, "Source Block is : "
smsg_len equ $-smsg

dmsg db 10,"Destination Block is : "
dmsg_len equ $-dmsg

space db " "

section .bss
Char_ans resb 2

Section .text
global  _start
_start:

print msg1,msg1_len
print smsg,smsg_len
mov rsi,sblock
call block_display

print dmsg,dmsg_len
mov rsi,dblock-2
call block_display
```

```
call block_transfer

print new_line,1
print msg2,msg2_len
print smsg,smsg_len
mov rsi,sblock
call block_display

print dmsg,dmsg_len
mov rsi,dblock-2
call block_display
print new_line,1

exit

block_transfer:
mov rsi,sblock+4
mov rdi,dblock+4
mov rcx,5
up:
mov al,[rsi]
mov [rdi],al
dec rsi
dec rdi
dec rcx
jnz up
ret
```

```
block_display:
mov rbp,5
next:
mov al,[rsi]
push rsi
call display
print space,1
pop rsi
inc rsi
dec rbp
jnz next
ret

display:
Mov rsi,Char_ans+1
mov rcx,2
mov rbx,16
up1:
xor rdx,rdx
div rbx
cmp dl,09
jbe add30
add dl,07h
add30:
add dl,30h
mov [rsi],dl
dec rsi
```

dec rcx

jnz up1

print Char_ans,2

ret

Code Input Output Run Save

```
;Overlapped Block Transfer

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

%macro exit 0
mov rax,60
mov rdi,0
syscall
%endmacro

section .data
sblock db 10h,20h,30h,40h,50h
dblock times 5 db 0
```

Code Input Output Run Save

```
msg1 db 10,13,"Before Overlapped Block Transfer :- ",10,13
msg1_len equ $-msg1

msg2 db 10,13,"After Overlapped Block Transfer :- ",10,13
msg2_len equ $-msg2

new_line db 10,13

smsg db 10, "Source Block is : "
smsg_len equ $-smsg

dmsg db 10,"Destination Block is : "
dmsg_len equ $-dmsg

space db " "

section .bss
Char_ans resb 2

Section .text
```

Code Input Output Run Save

```
global  _start
_start:

print msg1,msg1_len
print smsg,smsg_len
mov rsi,sblock
call block_display

print dmsg,dmsg_len
mov rsi,dblock-2
call block_display

call block_transfer

print new_line,1
print msg2,msg2_len
print smsg,smsg_len
mov rsi,sblock
call block_display

```

Code Input Output Run Save

```
print dmsg,dmsg_len
mov rsi,dblock-2
call block_display
print new_line,1

exit

block_transfer:
mov rsi,sblock+4
mov rdi,dblock+4
mov rcx,5
up:
mov al,[rsi]
mov [rdi],al
dec rsi
dec rdi
dec rcx
jnz up
ret

```

Code Input Output Run Save

```
block_display:
mov rbp,5
next:
mov al,[rsi]
push rsi
call display
print space,1
pop rsi
inc rsi
dec rbp
jnz next
ret

display:
Mov rsi,Char_ans+1
mov rcx,2
mov rbx,16
up1:
xor rdx,rdx
div rbx
```

Code Input Output Run Save

```
ret

display:
Mov rsi,Char_ans+1
mov rcx,2
mov rbx,16
up1:
xor rdx,rdx
div rbx
cmp dl,09
jbe add30
add dl,07h
add30:
add dl,30h
mov [rsi],dl
dec rsi
dec rcx
jnz up1
print Char_ans,2
ret
```

Output :

Code Input Output Run Save

```
Before Overlapped Block Transfer :-

Source Block is : 10 20 30 40 50
Destination Block is : 40 50 00 00 00

After Overlapped Block Transfer :-

Source Block is : 10 20 30 40 50
Destination Block is : 40 50 10 20 30

[Program exited with exit code 0]
```

MICROPROCESSOR LABORATORY

ASSIGNMENT NO. 9

Name :- Ojus Pravin Jaiswal
Roll No. :- SACO19108
Division :- A

Assignment No. 9

Program :

```
%macro read_or_print 4
mov Rax, %1
mov Rdi, %2
mov Rsi, %3
mov Rdx, %4
syscall
%endmacro

%macro exit 0
mov rax,60
mov rdi,0
syscall
%endmacro

section .data
hmsg db 10,"Enter 4 digit Hex number :- "
hmsg_len equ $-hmsg

bmsg db 10,"Equivalent BCD number :- "
bmsg_len equ $-bmsg

errmsg db 10,"Please enter valid hex number !!!"
errmsg_len equ $-errmsg
```

```
new_line db 10,13

section .bss
buf  resb 5
char_ans resb 1

section .text
global _start
_start:

call HEX_BCD
read_or_print 1,1,new_line,1

exit

Accept:
read_or_print 0,0,buf,5
mov rcx,4
mov rsi,buf
xor bx,bx
up: shl bx,4
mov al,[rsi]
cmp al,'0'
jb error
cmp al,'9'
jbe sub30
cmp al,'A'
```

```
jb error
cmp al,'F'
jbe sub37
cmp al,'a'
jb error
cmp al,'f'
jbe sub57

sub57:sub al,20h
sub37:sub al,07h
sub30:sub al,30h

add bx,ax
inc rsi
dec rcx
jnz up
ret

error: read_or_print 1,1,errmsg,errmsg_len

exit

HEX_BCD:
read_or_print 1,1,hmsg,hmsg_len
call Accept
read_or_print 1,1,buf,5
mov ax,bx
mov bx,10
```

xor bp,bp

back: xor dx,dx

div bx

push dx

inc bp

cmp ax,0

jne back

read_or_print 1,1,bmsg,bmsg_len

back1:pop dx

add dl,30h

mov [char_ans],dl

read_or_print 1,1,char_ans,1

dec bp

jnz back1

ret

Code Input Output Run Save

```
%macro read_or_print 4
mov Rax, %1
mov Rdi, %2
mov Rsi, %3
mov Rdx, %4
syscall
%endmacro

%macro exit 0
mov rax,60
mov rdi,0
syscall
%endmacro

section .data
hmsg db 10,"Enter 4 digit Hex number :- "
hmsg_len equ $-hmsg

bmsg db 10,"Equivalent BCD number :- "
bmsg_len equ $-bmsg
```

Code Input Output Run Save

```
21
22 errmsg db 10,"Please enter valid hex number !!!"
23 errmsg_len equ $-errmsg
24
25 new_line db 10,13
26
27 section .bss
28 buf  resb 5
29 char_ans resb 1
30
31 section .text
32 global _start
33 _start:
34
35 call HEX_BCD
36 read_or_print 1,1,new_line,1
37
38 exit
39
40 Accept:
```

Code Input Output Run Save

```
41 read_or_print 0,0,buf,5
42 mov rcx,4
43 mov rsi,buf
44 xor bx,bx
45 up: shl bx,4
46 mov al,[rsi]
47 cmp al,'0'
48 jb error
49 cmp al,'9'
50 jbe sub30
51 cmp al,'A'
52 jb error
53 cmp al,'F'
54 jbe sub37
55 cmp al,'a'
56 jb error
57 cmp al,'f'
58 jbe sub57
59
60 sub57:sub al,20h
```

Code Input Output Run Save

```
61 sub37:sub al,07h
62 sub30:sub al,30h
63
64 add bx,ax
65 inc rsi
66 dec rcx
67 jnz up
68 ret
69
70 error: read_or_print 1,1,errmsg,errmsg_len
71
72 exit
73
74 HEX_BCD:
75 read_or_print 1,1,hmsg,hmsg_len
76 call Accept
77 read_or_print 1,1,buf,5
78 mov ax,bx
79 mov bx,10
80 xor bp,bp
```

Code Input Output Run Save

```
call Accept
read_or_print 1,1,buf,5
mov ax,bx
mov bx,10
xor bp,bp
back: xor dx,dx
div bx
push dx
inc bp
cmp ax,0
jne back

read_or_print 1,1,bmsg,bmsg_len
back1:pop dx
add dl,30h
mov [char_ans],dl
read_or_print 1,1,char_ans,1
dec bp
jnz back1
ret
```

Input :

Code Input Output Run Save

```
0026
```

Output :


Code
Input
Output
Run
Save
Enter 4 digit Hex number :- 0026
Equivalent BCD number :- 38
[Program exited with exit code 0]

Microprocessor Laboratory

Assignment No. 13

NAME :- OJUS PRAVIN JAISWAL
ROLL NO. :- SACO19108
DIVISION :- A

Assignment No. 13

Program :

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

%macro exit 0
	mov rax,60
	mov rdi,0
	syscall
%endmacro

;-----------------------------------------------------------------------------------------------------------

section .data
	m1		db 10d,13d,"Enter Input number : ",10d,13d
	l1		equ $-m1
	m2		db 10d,13d,"Factorial of Number (in hexadecimal) : ",10d,13d
	l2		equ $-m2
	m3		db 10d,13d,"Assignment No. : 13 To Calculate Factorial of Number.",10d,13d
	l3		equ $-m3
```

```
        m4              db
10d,13d,"==========================================================
==========",10d,13d
        l4              equ $-m4
        nline           db 10
        nline_len       equ $-nline

;-------------------------------------------------------------------------------------------------------------

section .bss
        numascii resb 16
        factorial resq 1
        answer resb 16

section .text
global _start
_start:
        read_or_print 1,1,m4,l4
        read_or_print 1,1,m3,l3
        read_or_print 1,1,m4,l4
        read_or_print 1,1,m1,l1    ; Display message
        read_or_print 0,0,numascii,17
        read_or_print 1,1,numascii,17
        call asciihextohex
        mov [factorial],rbx
        mov rcx,[factorial]
        call facto
        mov rax,00
        read_or_print 1,1,m2,l2     ;Display Message
```

```
        mov rax,qword[factorial]
        call display       ; displays a 8 digit hex number  in rax
        read_or_print 1,1,nline,nline_len
        exit

;-------------------------------------------------------------------------------------------------------

facto:
        push rcx
        cmp rcx,01
        jne ahead
        jmp exit2
ahead:  dec rcx
        mov rax,rcx
        mul qword[factorial]
        mov qword[factorial],rax
        call facto
exit2:  pop rcx

        ret

;-------------------------------------------------------------------------------------------------------

asciihextohex:

        mov rsi,numascii
        mov rcx,16
        mov rbx,0
```

```
        mov rax,0

loop1:
        rol rbx,04
        mov al,[rsi]
        cmp al,39h
        jbe skip1
        sub al,07h
skip1:
        sub al,30h
        add rbx,rax
        inc rsi
        dec rcx
        jnz loop1

        ret
;------------------------------------------------------------------------------------------------------------

display:
        mov rsi,answer+15
        mov rcx,16

loop2:
        mov rdx,0
        mov rbx,16
        div rbx
        cmp dl,09h
        jbe skip2
```

```
        add dl,07h
skip2:
        add dl,30h
        mov [rsi],dl
        dec rsi
        dec rcx
        jnz loop2
        read_or_print 1,1,answer,16
        ret
```

Code Input Output Run Save

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

%macro exit 0
        mov rax,60
        mov rdi,0
        syscall
%endmacro

;-----------------------------------------------------------------------------------------------------------

section .data
    m1      db 10d,13d,"Enter Input number : ",10d,13d
    l1      equ $-m1
    m2      db 10d,13d,"Factorial of Number(in hexadecimal) : ",10d,13d
```

Code Input Output Run Save

```
    l2      equ $-m2
    m3      db 10d,13d,"Assignment No. : 13 To Calculate Factorial of Number.",10d,13d
    l3      equ $-m3
    m4      db 10d,13d,"================================================================================",10d,13d
    l4      equ $-m4
    nline       db 10
    nline_len   equ $-nline

;-----------------------------------------------------------------------------------------------------------

section .bss
    numascii resb 16
    factorial resq 1
    answer resb 16

section .text
global _start
_start:
    read_or_print 1,1,m4,l4
    read_or_print 1,1,m3,l3
```

Code Input Output Run Save

```
    read_or_print 1,1,m4,l4
    read_or_print 1,1,m1,l1    ; Display message
    read_or_print 0,0,numascii,17
    read_or_print 1,1,numascii,17
    call asciihextohex
    mov [factorial],rbx
    mov rcx,[factorial]
    call facto
    mov rax,00
    read_or_print 1,1,m2,l2      ;Display Message
    mov rax,qword[factorial]
    call display          ; displays a 8 digit hex number  in rax
    read_or_print 1,1,nline,nline_len
    exit

;-----------------------------------------------------------------------------------------------------------

facto:
    push rcx
    cmp rcx,01
```

Code Input Output Run Save

```
    jne ahead
    jmp exit2
ahead:  dec rcx
    mov rax,rcx
    mul qword[factorial]
    mov qword[factorial],rax
    call facto
exit2:  pop rcx

    ret

;-----------------------------------------------------------------------------------------------------------

asciihextohex:

    mov rsi,numascii
    mov rcx,16
    mov rbx,0
    mov rax,0
```

Code Input Output Run Save

```
loop1:
    rol rbx,04
    mov al,[rsi]
    cmp al,39h
    jbe skip1
    sub al,07h
skip1:
    sub al,30h
    add rbx,rax
    inc rsi
    dec rcx
    jnz loop1

    ret
;-----------------------------------------------------------------------------------------------------------

display:
    mov rsi,answer+15
    mov rcx,16
```

Code Input Output Run Save

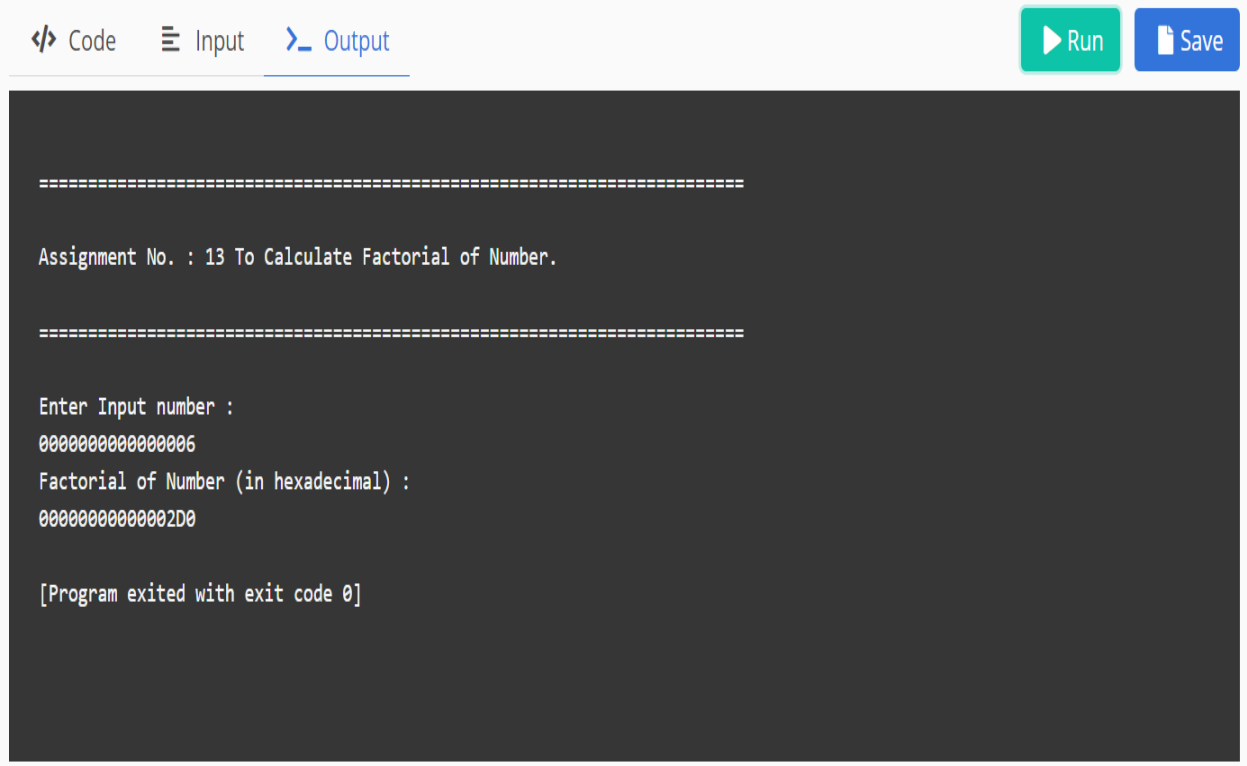
```
display:
    mov rsi,answer+15
    mov rcx,16

loop2:
    mov rdx,0
    mov rbx,16
    div rbx
    cmp dl,09h
    jbe skip2
    add dl,07h
skip2:
    add dl,30h
    mov [rsi],dl
    dec rsi
    dec rcx
    jnz loop2
    read_or_print 1,1,answer,16
    ret
```

Input :

Code Input Output Run Save

```
0000000000000006
```

Output :


Code
Input
Output
Run
Save
==========
Assignment No. : 13 To Calculate Factorial of Number.
==========
Enter Input number :
0000000000000006
Factorial of Number (in hexadecimal) :
00000000000002D0
[Program exited with exit code 0]