



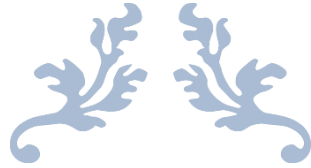


MICROPROCESSOR LABORATORY

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. :- SAC019108
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

```
1 %macro read_or_print 4
2   mov Rax, %1
3   mov Rdi, %2
4   mov Rsi, %3
5   mov Rdx, %4 syscall
6 %endmacro
7
8 %macro exit 0 mov rax,60 mov rdi,0 syscall
9 %endmacro
10
11 section .bss array resd 200
12 counter resb 1
13
14 section .text global _start
15 _start:
16
17 ; Accept numbers in an array mov byte[counter],09
18 mov rsi,array loop:
19 read_or_print 0,0,rsi,17 add Rsi,17
20 dec byte[counter] jnz loop
```

 Code  Input  Output

 Run

 Save

```
8 %macro exit 0 mov rax,60 mov rdi,0 syscall
9 %endmacro
10
11 section .bss array resd 200
12 counter resb 1
13
14 section .text global _start
15 _start:
16
17 ; Accept numbers in an array mov byte[counter],09
18 mov rsi,array loop:
19 read_or_print 0,0,rsi,17 add Rsi,17
20 dec byte[counter] jnz loop
21
22 ;display Contents of Array mov byte [counter],09 mov rsi, array
23 loop1:
24 read_or_print 1,1,rsi,17 add Rsi, 17
25 dec byte [counter] jnz loop1
26 exit
27 |
```

Input :

</> Code

≡ Input

 >_ Output

▶ Run

📄 Save

```
1 0123456789ABCDEF
2 123456789ABCDEF0
3 23456789ABCDEF01
4 3456789ABCDEF012
5 456789ABCDEF0123
6 56789ABCDEF01234
7 6789ABCDEF012345
8 789ABCDEF0123456
9 89ABCDEF01234567
```

Output :

</> Code

≡ Input

 >_ Output

▶ Run

📄 Save

```
0123456789ABCDEF
123456789ABCDEF0
23456789ABCDEF01
3456789ABCDEF012
456789ABCDEF0123
56789ABCDEF01234
6789ABCDEF012345
789ABCDEF0123456
89ABCDEF01234567
[Program exited with exit code 0]
```

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



```
1 %macro read_or_print 4
2   mov Rax, %1
3   mov Rdi, %2
4   mov Rsi, %3
5   mov Rdx, %4
6   syscall
7 %endmacro
8
9 %macro exit 0
10  mov rax,60
11  mov rdi,0
12  syscall
13 %endmacro
14
15 section .data
16 msg db 10,13,"Length of the String is:",10,13
17 msglen equ $-msg
18
19 section .bss
20 str1 resb 200
```

 Code  Input  Output

 Run

 Save

```
21 result resb 17
22
23 section .text
24 global _start
25 _start:
26
27 read_or_print 0,0,str1,200
28 call display
29 exit
30
31 display:
32 mov rsi,result+15
33 mov rcx,16
34 loop2: mov rdx,0
35         mov rbx,16
36         div rbx
37         cmp dl,09h
38         add dl,07h
39 skip2: add dl,30h
40         dec rsi
```

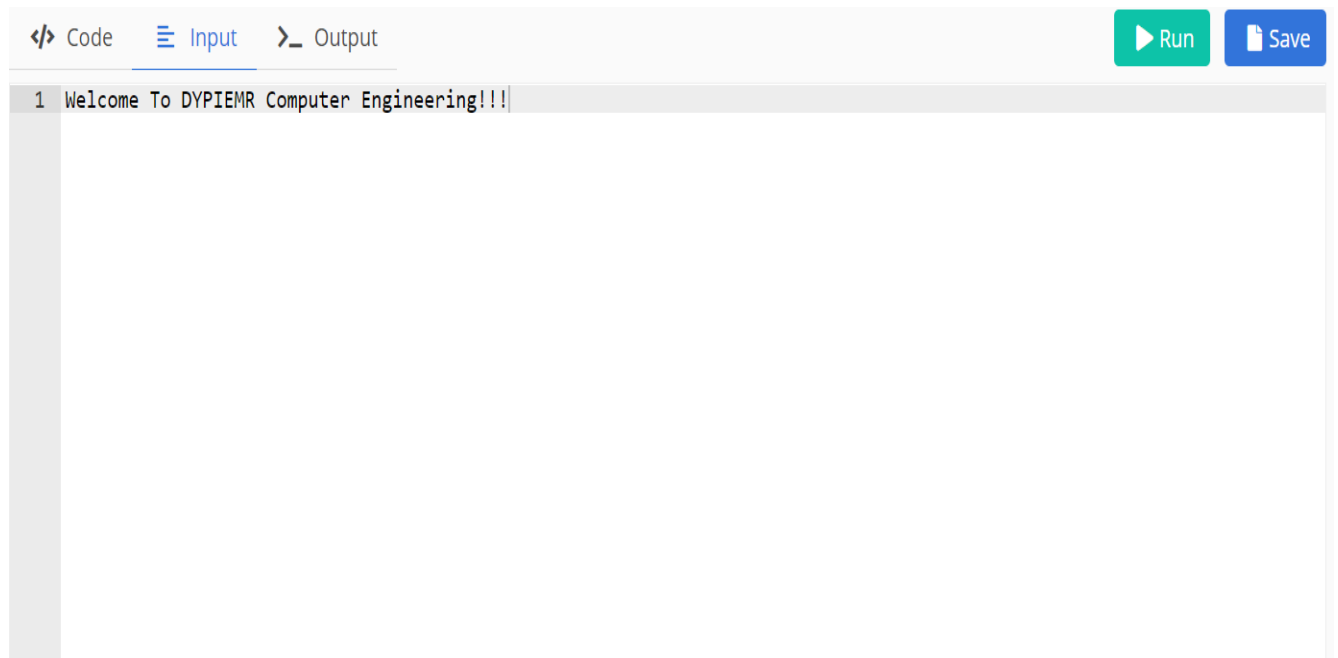
 Code  Input  Output

 Run

 Save

```
26
27 read_or_print 0,0,str1,200
28 call display
29 exit
30
31 display:
32 mov rsi,result+15
33 mov rcx,16
34 loop2: mov rdx,0
35         mov rbx,16
36         div rbx
37         cmp dl,09h
38         add dl,07h
39 skip2: add dl,30h
40         dec rsi
41         dec rcx
42         jnz loop2
43 read_or_print 1,1,msg,msglen
44 read_or_print 1,1,result,16
45 ret
```

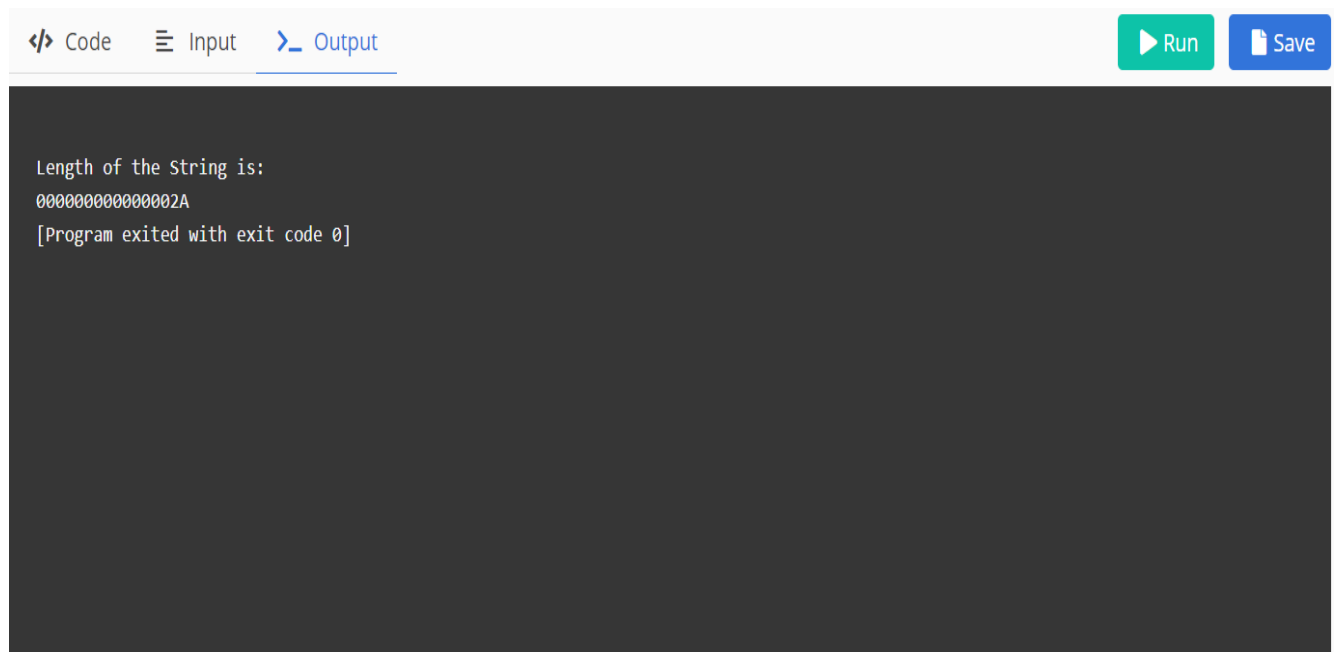
Input :



The screenshot shows an online code editor interface. At the top, there is a header bar with three tabs: 'Code' (selected), 'Input', and 'Output'. To the right of the tabs are two buttons: a green 'Run' button and a blue 'Save' button. The main area is a large text editor with a light gray background. A single line of text is entered: '1 Welcome To DYPIEMR Computer Engineering!!!'. The line number '1' is visible on the left side of the editor.

```
1 Welcome To DYPIEMR Computer Engineering!!!
```

Output :



The screenshot shows the same online code editor interface, but now the 'Output' tab is selected. The 'Code' and 'Input' tabs are still visible but not active. The 'Run' and 'Save' buttons remain. The main area is a dark gray text editor showing the output of the program. The output consists of three lines: 'Length of the String is:', '000000000000002A', and '[Program exited with exit code 0]'. The line numbers 1, 2, and 3 are visible on the left side of the editor.

```
1 Length of the String is:  
2 000000000000002A  
3 [Program exited with exit code 0]
```



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

```

1 %macro read_or_print 4
2   mov Rax, %1
3   mov Rdi, %2
4   mov Rsi, %3
5   mov Rdx, %4
6   syscall
7 %endmacro
8
9 %macro exit 0
10  mov rax,60
11  mov rdi,0
12  syscall
13 %endmacro
14
15 section .data
16 array db 00h,10h,20h,30h,40h,50h,60h,70h
17 msg1 db 10,13,"The smallest element in the array is : ",10,13
18 msglen1 equ $-msg1
19 msg2 db 10,13,"The largest element in the array is : ",10,13
20 msglen2 equ $-msg2

```

</> Code ≡ Input >_ Output

▶ Run

📄 Save

```
21
22 section .bss
23 cnt resb 1
24 result resb 16
25
26 section .text
27 global _start
28 _start:
29
30 ;Smallest Number
31 mov byte[cnt],08
32 mov rsi, array
33 mov al,0
34 up0:cmp al,[rsi]
35     jl skip0
36     xchg al,[rsi]
37 skip0:inc rsi
38     dec byte[cnt]
39     jnz up0
40
```

</> Code ≡ Input >_ Output

▶ Run

📄 Save

```
41 call display1
42
43 ;Largest Number
44 mov byte[cnt],08
45 mov rsi,array
46 mov al,0
47 up1:cmp al,[rsi]
48     jg skip1
49     xchg al,[rsi]
50 skip1:inc rsi
51     dec byte[cnt]
52     jnz up1
53
54 call display2
55 exit
56
57 display1:
58 mov rsi,result+15
59 mov rcx,16
60 loop2:mov rdx,0
```

</> Code ≡ Input >_ Output

▶ Run

📄 Save

```
61     mov rbx,16
62     div rbx
63     cmp dl,09h
64     jbe skip2
65     add dl,07h
66 skip2:add dl,30h
67     mov [rsi],dl
68     dec rsi
69     dec rcx
70     jnz loop2
71 read_or_print 1,1,msg1,msglen1
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
```


</> 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
```

```
000000000000000100h,00000000000000004h,0000000000000003h,0000000000000002h,0000000000  
00001h
```

```
    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
1  %macro scall 4
2      mov rax,%1
3      mov rdi,%2
4      mov rsi,%3
5      mov rdx,%4
6      syscall
7  %endmacro
8
9  %macro exit 0
10     mov rax, 60
11     mov rdi,0
12     syscall
13 %endmacro
14
15 section .data
16     arr dq 0000000000000100h,000000000000004h,000000000000003h,000000000000002h,000000000000001h
17     n equ 5
18
19     menu db 10d,13d,"*****MENU*****"
20         db 10d,13d,"1. Addition"
```

```
</> Code  Input  Output  Run  Save
21         db 10d,13d,"2. Subtraction"
22         db 10d,13d,"3. Multiplication"
23         db 10d,13d,"4. Division"
24         db 10d,13d,"5. Exit"
25         db 10d,13d,"Enter your Choice : "
26     menu_len equ $-menu
27
28     m1 db 10d,13d,"Addition : "
29     l1 equ $-m1
30     m2 db 10d,13d,"Subtraction : "
31     l2 equ $-m2
32     m3 db 10d,13d,"Multiplication : "
33     l3 equ $-m3
34     m4 db 10d,13d,"Division : "
35     l4 equ $-m4
36
37 section .bss
38     answer resb 16
39     choice resb 2
40
```


</> Code ≡ Input >_ Output

▶ Run

📄 Save

```
41 section .text
42 global _start:
43 _start:
44
45     up:scall 1,1,menu,menu_len
46         scall 0,0,choice,2
47
48     cmp byte[choice],'1'
49     je case1
50     cmp byte[choice],'2'
51     je case2
52     cmp byte[choice],'3'
53     je case3
54     cmp byte[choice],'4'
55     je case4
56     cmp byte[choice],'5'
57     je case5
58
59     case1: scall 1,1,m1,l1
60         call addition
```

</> Code ≡ Input >_ Output

▶ Run

📄 Save

```
61         jmp up
62
63     case2: scall 1,1,m2,l2
64         call subtraction
65         jmp up
66
67     case3: scall 1,1,m3,l3
68         call multiplication
69         jmp up
70
71     case4: scall 1,1,m4,l4
72         call division
73         jmp up
74
75     case5:exit
76
77 addition:
78     mov rcx,n
79     dec rcx
80     mov rsi,arr
```

</> Code ≡ Input >_ Output

▶ Run

📄 Save

```
81     mov rax,[rsi]
82 up1:add rsi,8
83     mov rbx,[rsi]
84     add rax,rbx
85     loop up1
86     call display
87     ret
88
89 subtraction:
90     mov rcx,n
91     dec rcx
92     mov rsi,arr
93     mov rax,[rsi]
94 up2:add rsi,8
95     mov rbx,[rsi]
96     sub rax,rbx
97     loop up2
98     call display
99     ret
100
```

</> Code ≡ Input >_ Output

▶ Run

📄 Save

```
101 ▾ multiplication:
102     mov rcx,n
103     dec rcx
104     mov rsi,arr
105     mov rax,[rsi]
106 ▾ up3:add rsi,8
107     mov rbx,[rsi]
108     mul rbx
109     loop up3
110     call display
111     ret
112
113 ▾ division:
114     mov rcx,n
115     dec rcx
116     mov rsi,arr
117     mov rax,[rsi]
118 ▾ up4:add rsi,8
119     mov rbx,[rsi]
120     mov rdx,0
```

</> Code ≡ Input >_ Output

▶ Run

📄 Save

```
121     div rbx
122     loop up4
123     call display
124     ret
125
126 ▾ display:
127     mov rsi,answer+15
128     mov rcx,16
129 ▾ cnt:mov rdx,0
130     mov rbx,16
131     div rbx
132     cmp dl,09h
133     jbe add30
134     add dl,07h
135 ▾ add30:add dl,30h
136     mov [rsi],dl
137     dec rsi
138     dec rcx
139     jnz cnt
140     scall 1,1,answer,16
```

</> Code ≡ Input >_ Output

▶ Run

📄 Save

```
122     loop up4
123     call display
124     ret
125
126 ▾ display:
127     mov rsi,answer+15
128     mov rcx,16
129 ▾ cnt:mov rdx,0
130     mov rbx,16
131     div rbx
132     cmp dl,09h
133     jbe add30
134     add dl,07h
135 ▾ add30:add dl,30h
136     mov [rsi],dl
137     dec rsi
138     dec rcx
139     jnz cnt
140     scall 1,1,answer,16
141     ret
```

Input :

</> Code

☰ Input

>_ Output

▶ Run

📄 Save

```
1 1
2 2
3 3
4 4
5 5
```

Output :

</> Code

☰ Input

>_ Output

▶ Run

📄 Save

```
*****MENU*****
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit
Enter your Choice :
Addition : 00000000000010A
*****MENU*****
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit
Enter your Choice :
Subtraction : 000000000000F6
*****MENU*****
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit
Enter your Choice :
Multiplication : 000000000001800
```

</> Code ≡ Input >_ Output

▶ Run

📄 Save

```
5. Exit
Enter your Choice :
Multiplication : 000000000001800
*****MENU*****
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit
Enter your Choice :
Division : 00000000000000A
*****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

```
1 %macro print 2
2 mov Rax,1
3 mov Rdi,1
4 mov Rsi,%1
5 mov Rdx,%2
6 syscall
7 %endmacro
8
9 %macro exit 0
10 mov rax,60
11 mov rdi,0
12 syscall
13 %endmacro
14
15 section .data
16 arr dq 00h,-10h,20h,-30h,40h,-50h,60h,-70h
17 n equ 8
18 pmsg db 10,13,"The count of positive elements in the array is : ",10,13
19 pmsglen equ $-pmsg
20 nmsg db 10,13,"The count of negative element in the array is : ",10,13
```

</> Code ≡ Input >_ Output

▶ Run

📄 Save

```
21 nmsglen equ $-nmsg
22 newline db 10,13
23
24 section .bss
25 pcnt resq 1
26 ncnt resq 1
27 char_answer resb 16
28
29 section .text
30 global _start
31 _start:
32
33 mov rsi,arr
34 mov rdi,n
35 mov rbx,0
36 mov rcx,0
37
38 up:mov rax,[rsi]
39     rol rax,1
40     jc negative
```

</> Code ≡ Input >_ Output

▶ Run

📄 Save

```
41
42 positive:inc rbx
43           jmp next
44
45 negative:inc rcx
46
47 next:add rsi,8
48       dec rdi
49       jnz up
50
51 mov [pcnt],rbx
52 mov [ncnt],rcx
53
54 print pmsg,pmsglen
55 mov rax,[pcnt]
56 call display
57
58 print nmsg,nmsglen
59 mov rax,[ncnt]
60 call display
```

</> Code

☰ Input

>_ Output

▶ Run

📄 Save

```
61
62 print newline,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 :

</> Code

☰ Input

>_ Output

▶ Run

📄 Save

```
The count of positive elements in the array is :
0000000000000004
The count of negative element in the array is :
0000000000000004

[Program exited with exit code 0]
```

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
```

```
mmsg db 10,"Machine Status Word :: "
```

```
mmsg_len:equ $-mmsg
```

```
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
1  %macro disp 2
2      mov rax,1
3      mov rdi,1
4      mov rsi,%1
5      mov rdx,%2
6      syscall
7  %endmacro
8
9  section .data
10
11      rmodemsg db 10,"Processor is in Real Mode"
12      rmsg_len:equ $-rmodemsg
13
14      pmodemsg db 10,"Processor is in Protected Mode"
15      pmsg_len:equ $-pmodemsg
16
17      gdtmsg db 10,"GDT Contents are :: "
18      gmsg_len:equ $-gdtmsg
19
20      ldtmsg db 10,"LDT Contents are :: "
```

```
</> Code  Input  Output  Run  Save
21      lmsg_len:equ $-ldtmsg
22
23      idtmsg db 10,"IDT Contents are :: "
24      imsg_len:equ $-idtmsg
25
26      trmsg db 10,"Task Register Contents are :: "
27      tmsg_len:equ $-trmsg
28
29      mswmsg db 10,"Machine Status Word :: "
30      mmsg_len:equ $-mswmsg
31
32      promsg db 10,"Processor Information :: "
33      promsg_len:equ $-promsg
34
35      colmsg db ':'
36
37      newline db 10
38
39  section .bss
40      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
```

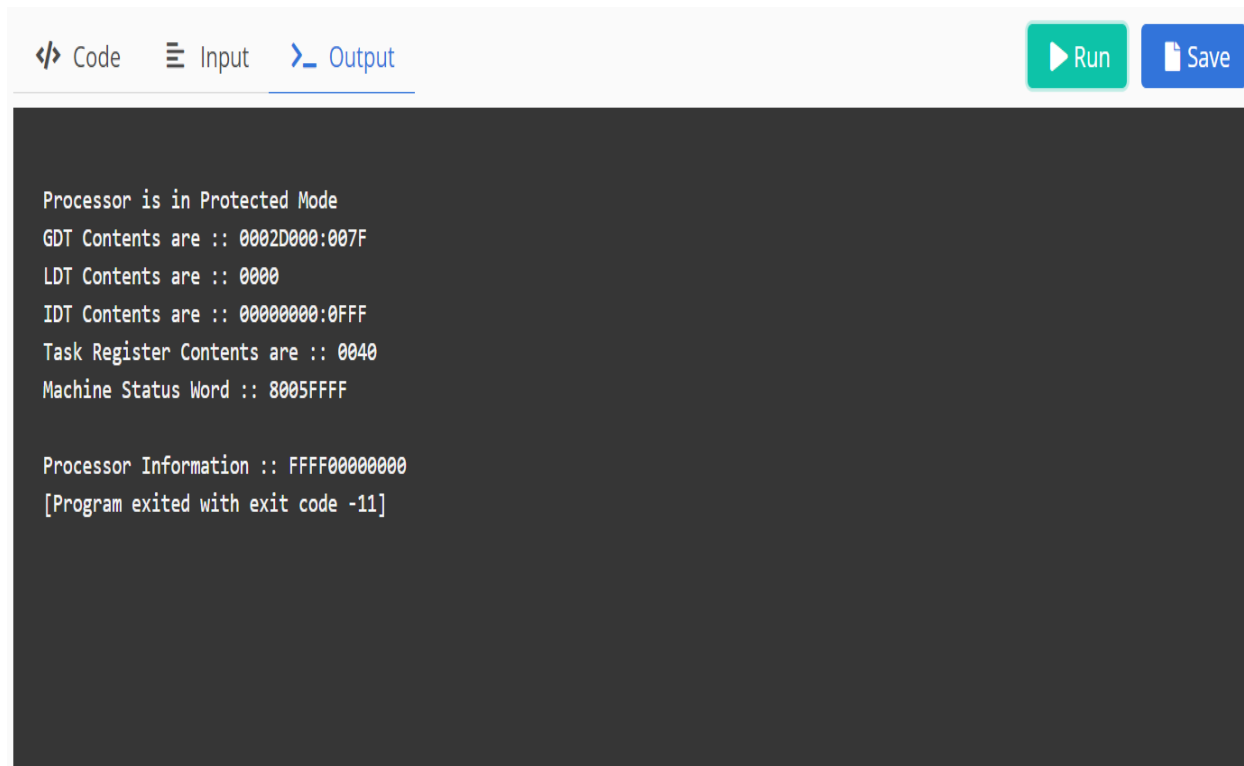
</> Code ≡ Input >_ Output

▶ Run

📄 Save

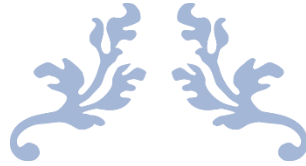
```
118
119 exit: mov eax,01
120       mov ebx,00
121       int 80h
122
123 disp_num:
124     mov esi,dnum_buff
125     mov ecx,04
126 up1:  rol bx,4
127     mov dl,bl
128     and dl,0fh
129     add dl,30h
130     cmp dl,39h
131     jbe skip1
132     add dl,07h
133 skip1: mov [esi],dl
134     inc esi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

Output :

A screenshot of a code editor interface. At the top, there is a header bar with three tabs: 'Code' (with a code icon), 'Input' (with a list icon), and 'Output' (with a document icon). The 'Output' tab is currently selected. To the right of the tabs are two buttons: a green 'Run' button with a play icon and a blue 'Save' button with a floppy disk icon. Below the header bar is a large dark gray area representing the output console. It contains several lines of white text showing system information and program status.

```
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
1 ;Non Overlapped Block Transfer
2
3 %macro print 2
4 Mov rax,1
5 Mov rdi,1
6 Mov rsi,%1
7 Mov rdx,%2
8 syscall
9 %endmacro
10
11 %macro exit 0
12 mov rax,60
13 mov rdi,0
14 syscall
15 %endmacro
16
17 section .data
18 sblock db 10h,20h,30h,40h,50h
19 dblock times 5 db 0
20
```

```
</> Code  Input  Output  Run  Save
21 msg1 db 10,13,"Before Non Overlapped Block Transfer :- ",10,13
22 msg1_len equ $-msg1
23
24 msg2 db 10,13,"After Non Overlapped Block Transfer :- ",10,13
25 msg2_len equ $-msg2
26
27 new_line db 10,13
28
29 smsg db 10, "Source Block is : "
30 smsg_len equ $-smsg
31
32 dmsg db 10,"Destination Block is : "
33 dmsg_len equ $-dmsg
34
35 space db " "
36
37 section .bss
38 Char_ans resb 2
39
40 Section .text
```

</> Code ≡ Input >_ Output

▶ Run

📄 Save

```
41 global _start
42 _start:
43
44 print msg1,msg1_len
45 print smsg,smsg_len
46 mov rsi,sblock
47 call block_display
48
49 print dmsg,dmsg_len
50 mov rsi,dblock
51 call block_display
52
53 call block_transfer
54
55 print new_line,1
56 print msg2,msg2_len
57 print smsg,smsg_len
58 mov rsi,sblock
59 call block_display
60
```

</> Code ≡ Input >_ Output

▶ Run

📄 Save

```
61 print dmsg,dmsg_len
62 mov rsi,dblock
63 call block_display
64 print new_line,1
65
66 exit
67
68 block_transfer:
69 mov rsi,sblock
70 mov rdi,dblock
71 mov rcx,5
72 up:
73 mov al,[rsi]
74 mov [rdi],al
75 inc rsi
76 inc rdi
77 dec rcx
78 jnz up
79 ret
80
```

</> Code ≡ Input >_ Output

▶ Run

📄 Save

```
81 block_display:
82 mov rbp,5
83 next:
84 mov al,[rsi]
85 push rsi
86 call display
87 print space,1
88 pop rsi
89 inc rsi
90 dec rbp
91 jnz next
92 ret
93
94 display:
95 Mov rsi,Char_ans+1
96 mov rcx,2
97 mov rbx,16
98 up1:
99 xor rdx,rdx
100 div rbx
```

```
</> Code  Input  Output  Run  Save
93
94 display:
95 Mov rsi,Char_ans+1
96 mov rcx,2
97 mov rbx,16
98 up1:
99 xor rdx,rdx
100 div rbx
101 cmp dl,09
102 jbe add30
103 add dl,07h
104 add30:
105 add dl,30h
106 mov [rsi],dl
107 dec rsi
108 dec rcx
109 jnz up1
110 print Char_ans,2
111 ret
112
```

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
1 ;Overlapped Block Transfer
2
3 %macro print 2
4 Mov rax,1
5 Mov rdi,1
6 Mov rsi,%1
7 Mov rdx,%2
8 syscall
9 %endmacro
10
11 %macro exit 0
12 mov rax,60
13 mov rdi,0
14 syscall
15 %endmacro
16
17 section .data
18 sblock db 10h,20h,30h,40h,50h
19 dblock times 5 db 0
20
```


```
</> Code  Input  >_ Output  Run  Save
21 msg1 db 10,13,"Before Overlapped Block Transfer :- ",10,13
22 msg1_len equ $-msg1
23
24 msg2 db 10,13,"After Overlapped Block Transfer :- ",10,13
25 msg2_len equ $-msg2
26
27 new_line db 10,13
28
29 smsg db 10, "Source Block is : "
30 smsg_len equ $-smsg
31
32 dmsg db 10,"Destination Block is : "
33 dmsg_len equ $-dmsg
34
35 space db " "
36
37 section .bss
38 Char_ans resb 2
39
40 Section .text
```

 Code  Input  Output

 Run

 Save




```
41 global _start
42 _start:
43
44 print msg1,msg1_len
45 print smsg,smsg_len
46 mov rsi,sblock
47 call block_display
48
49 print dmsg,dmsg_len
50 mov rsi,dblock-2
51 call block_display
52
53 call block_transfer
54
55 print new_line,1
56 print msg2,msg2_len
57 print smsg,smsg_len
58 mov rsi,sblock
59 call block_display
60
```

 Code  Input  Output

 Run

 Save

```
61 print dmsg,dmsg_len
62 mov rsi,dblock-2
63 call block_display
64 print new_line,1
65
66 exit
67
68 block_transfer:
69 mov rsi,sblock+4
70 mov rdi,dblock+4
71 mov rcx,5
72 up:
73 mov al,[rsi]
74 mov [rdi],al
75 dec rsi
76 dec rdi
77 dec rcx
78 jnz up
79 ret
80
```

 Code  Input  Output

 Run

 Save

```
81 block_display:
82 mov rbp,5
83 next:
84 mov al,[rsi]
85 push rsi
86 call display
87 print space,1
88 pop rsi
89 inc rsi
90 dec rbp
91 jnz next
92 ret
93
94 display:
95 mov rsi,Char_ans+1
96 mov rcx,2
97 mov rbx,16
98 up1:
99 xor rdx,rdx
100 div rbx
```

```
</> Code  Input  Output  Run  Save
92  ret
93
94  display:
95  Mov rsi,Char_ans+1
96  mov rcx,2
97  mov rbx,16
98  up1:
99  xor rdx,rdx
100 div rbx
101 cmp dl,09
102 jbe add30
103 add dl,07h
104 add30:
105 add dl,30h
106 mov [rsi],dl
107 dec rsi
108 dec rcx
109 jnz up1
110 print Char_ans,2
111 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

```
1 %macro read_or_print 4
2   mov Rax, %1
3   mov Rdi, %2
4   mov Rsi, %3
5   mov Rdx, %4
6   syscall
7 %endmacro
8
9 %macro exit 0
10  mov rax, 60
11  mov rdi, 0
12  syscall
13 %endmacro
14
15 section .data
16 hmsg db 10, "Enter 4 digit Hex number :- "
17 hmsg_len equ $-hmsg
18
19 bmsg db 10, "Equivalent BCD number :- "
20 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

```
76 call Accept
77 read_or_print 1,1,buf,5
78 mov ax,bx
79 mov bx,10
80 xor bp,bp
81 back: xor dx,dx
82 div bx
83 push dx
84 inc bp
85 cmp ax,0
86 jne back
87
88 read_or_print 1,1,bmsg,bmsg_len
89 back1: pop dx
90 add dl,30h
91 mov [char_ans],dl
92 read_or_print 1,1,char_ans,1
93 dec bp
94 jnz back1
95 ret
```

Input :

</> Code

☰ Input




>_ Output

▶ Run

📄 Save

```
1 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
1  %macro read_or_print 4
2      mov rax,%1
3      mov rdi,%2
4      mov rsi,%3
5      mov rdx,%4
6      syscall
7  %endmacro
8
9  %macro exit 0
10     mov rax,60
11     mov rdi,0
12     syscall
13 %endmacro
14
15 ;-----
16
17 section .data
18     m1      db 10d,13d,"Enter Input number : ",10d,13d
19     l1      equ $-m1
20     m2      db 10d,13d,"Factorial of Number(in hexadecimal) : ",10d,13d
21
```

```
</> Code  Input  Output  Run  Save
21     12      equ $-m2
22     m3      db 10d,13d,"Assignment No. : 13 To Calculate Factorial of Number.",10d,13d
23     l3      equ $-m3
24     m4      db 10d,13d,"===== ",10d,13d
25     l4      equ $-m4
26     nline   db 10
27     nline_len equ $-nline
28
29 ;-----
30
31 section .bss
32     numascii resb 16
33     factorial resq 1
34     answer resb 16
35
36 section .text
37 global _start
38 _start:
39     read_or_print 1,1,m4,l4
40     read_or_print 1,1,m3,l3
41
```

</> Code ≡ Input >_ Output

▶ Run

📄 Save

```
41 read_or_print 1,1,m4,14
42 read_or_print 1,1,m1,11 ; Display message
43 read_or_print 0,0,numascii,17
44 read_or_print 1,1,numascii,17
45 call asciihextohex
46 mov [factorial],rbx
47 mov rcx,[factorial]
48 call facto
49 mov rax,00
50 read_or_print 1,1,m2,12 ;Display Message
51 mov rax,qword[factorial]
52 call display ; displays a 8 digit hex number in rax
53 read_or_print 1,1,nline,nline_len
54 exit
55
56 ;-----
57
58 facto:
59 push rcx
60 cmp rcx,01
61
```

</> Code ≡ Input >_ Output

▶ Run

📄 Save

```
61 jne ahead
62 jmp exit2
63 ahead: dec rcx
64 mov rax,rcx
65 mul qword[factorial]
66 mov qword[factorial],rax
67 call facto
68 exit2: pop rcx
69
70 ret
71
72 ;-----
73
74 asciihextohex:
75
76 mov rsi,numascii
77 mov rcx,16
78 mov rbx,0
79 mov rax,0
80
81
```

</> Code ≡ Input >_ Output

▶ Run

📄 Save

```
81 loop1:
82 rol rbx,04
83 mov al,[rsi]
84 cmp al,39h
85 jbe skip1
86 sub al,07h
87 skip1:
88 sub al,30h
89 add rbx,rax
90 inc rsi
91 dec rcx
92 jnz loop1
93
94 ret
95 ;-----
96
97 display:
98 mov rsi,answer+15
99 mov rcx,16
100
101
```

</> Code

☰ Input

>_ Output

▶ Run

📄 Save

```
96
97 ▾ display:
98     mov rsi,answer+15
99     mov rcx,16
100
101 ▾ loop2:
102     mov rdx,0
103     mov rbx,16
104     div rbx
105     cmp dl,09h
106     jbe skip2
107     add dl,07h
108 ▾ skip2:
109     add dl,30h
110     mov [rsi],dl
111     dec rsi
112     dec rcx
113     jnz loop2
114     read_or_print 1,1,answer,16
115     ret
```

Input :

</> Code

☰ Input



>_ Output

▶ Run

📄 Save

```
1 0000000000000006
```

Output :

 Code  Input  Output

 Run

 Save

```
=====
```

```
Assignment No. : 13 To Calculate Factorial of Number.
```

```
=====
```

```
Enter Input number :
```

```
000000000000006
```

```
Factorial of Number (in hexadecimal) :
```

```
0000000000002D0
```

```
[Program exited with exit code 0]
```