
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
```

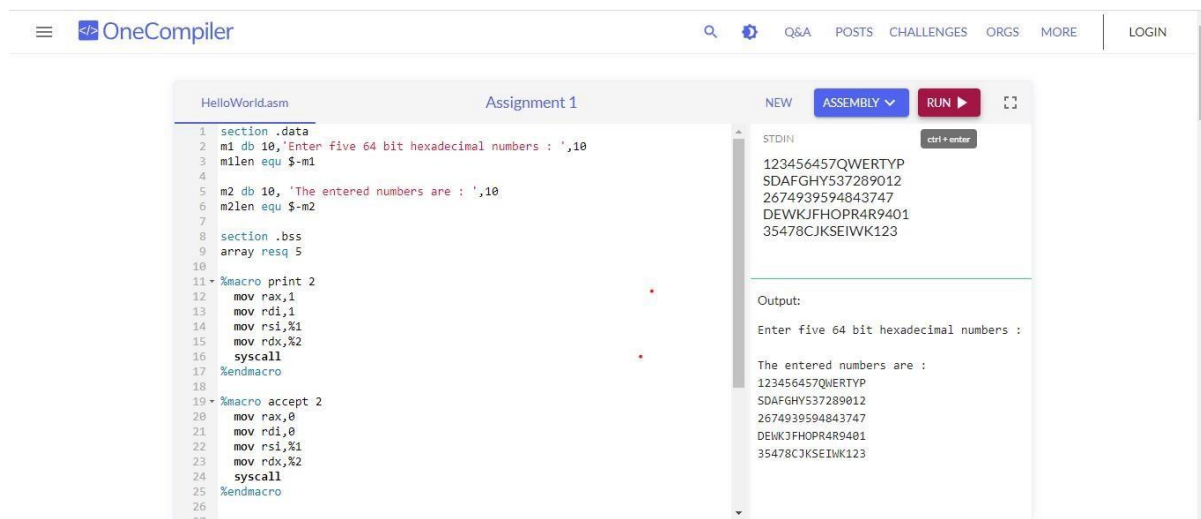
```
%macro accept 2
mov rax,0 mov
rdi,0 mov rsi,%1
mov rdx,%2
syscall
%endmacro
```

```
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:



The screenshot shows the OneCompiler web interface. The main editor displays an assembly file named 'HelloWorld.asm' with the following code:

```
1 section .data
2 m1 db 10,'Enter five 64 bit hexadecimal numbers : ',10
3 m1len equ $-m1
4
5 m2 db 10,'The entered numbers are : ',10
6 m2len equ $-m2
7
8 section .bss
9 array resq 5
10
11 * %macro print 2
12   mov rax,1
13   mov rdi,1
14   mov rsi,%1
15   mov rdx,%2
16   syscall
17 %endmacro
18
19 * %macro accept 2
20   mov rax,0
21   mov rdi,0
22   mov rsi,%1
23   mov rdx,%2
24   syscall
25 %endmacro
26
27
```

On the right side, the 'STDIN' input area contains the text: '123456457QWERTYP', 'SDAFGHY537289012', '2674939594843747', 'DEWKJFHOPR4R9401', '35478CJKSEIWK123'. Below this, the 'Output' section shows the program's execution results: 'Enter five 64 bit hexadecimal numbers :', 'The entered numbers are :', '123456457QWERTYP', 'SDAFGHY537289012', '2674939594843747', 'DEWKJFHOPR4R9401', '35478CJKSEIWK123'.

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:

The screenshot shows the OneCompiler web interface. The top navigation bar includes links for Apps, YouTube, and various assignments. The main editor area is titled 'HelloWorld.asm' and 'Assignment 2'. It contains the following assembly code:

```
1 section .data
2 pos db 10,13,"The count of positive numbers are : ",10
3 plen equ $-pos
4
5 negative db 10,13,"The count of negative numbers are : ",10
6 nlen equ $-negative
7
8 array dw 5678h, 8321h, 078ah, 52ah, 1a34h
9
10 cnt equ 5
11
12 pcnt db 0
13 ncnt db 0
14
15 section .bss
16 disp_buff resb 2
17
18 %macro disp 2
19 mov rax,1
20 mov rdi,1
21 mov rsi,%1
22 mov rdx,%2
23 syscall
24 %endmacro
25
26 section .text
27 global _start
```

On the right side, there is a 'STDIN' input field with the text 'Input for the program (Optional)'. Below it, the 'Output' section displays the program's results:

```
Output:
The count of positive numbers are :
04
The count of negative numbers are :
01
```

The interface also includes buttons for 'NEW', 'ASSEMBLY', and 'RUN', and a note at the bottom right stating 'created 9 days ago'.

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

```
1 section .data
2     hello: db 'Enter the input string : ',10 ; 'Hello world!' plus a L
3     helloLen: equ $-hello ; Length of the 'Hello world!' string
4
5     m1: db 'Length of the string is : ',10
6     m1len: equ $-m1
7
8 section .bss
9     str1: resb 20
10    str1len: equ $-str1
11
12    dispbuff resb 20
13
14 %macro print 2
15     mov rax,1
16     mov rdi,1
17     mov rsi,%1
18     mov rdx,%2
19     syscall
20 %endmacro
21
22 %macro accept 2
23     mov rax,0
24     mov rdi,0
25     mov rsi,%1
26     mov rdx,%2
```

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. block contents after xfer are :'\n
    dst_len equ $-dst_blk

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

Apps YouTube Assignment 1 - Ass... Assignment 2 - Ass... Assignment 3 - Ass... Assignment 4 - Ass... Assignment 5 - Ass... Assignment 6 - Ass... E-Way Bill System Reading list

OneCompiler

NEW ASSEMBLY RUN

HelloWorld.asm Assignment 4

```
85 inc ebx
86 loop dup1
87
88 dispmsg dispbuff,2
89 dispmsg newline,1
90 ret
91
92 disp8:
93 mov ecx,8
94 mov edi,dispbuff
95
96 dup2:
97 rol ebx,4
98 mov al,bl
99 and al,0fh
100 cmp al,09
101 jbe dskip2
102 add al,07h
103
104 dskip2:
105 add al,30h
106 mov [edi],al
107 inc edi
108 loop dup2
109
110 dispmsg dispbuff,8
111 ret
```

STDIN

Input for the program (Optional)

Output:

Dest. bolk contents after xfer are :

006001E5:01

006001E6:02

006001E7:03

006001E8:04

006001E9:05

006001EA:01

006001EB:02

006001EC:03

006001ED:04

006001EE:05

created 18 hours ago

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 :'\n
    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 newline,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 newline,1

ret

disp8:

mov ecx,8

mov edi,dispbuff

dup2:

rol ebx,4

mov al,bl

and al,0fh

cmp al,09

jbe dskip2

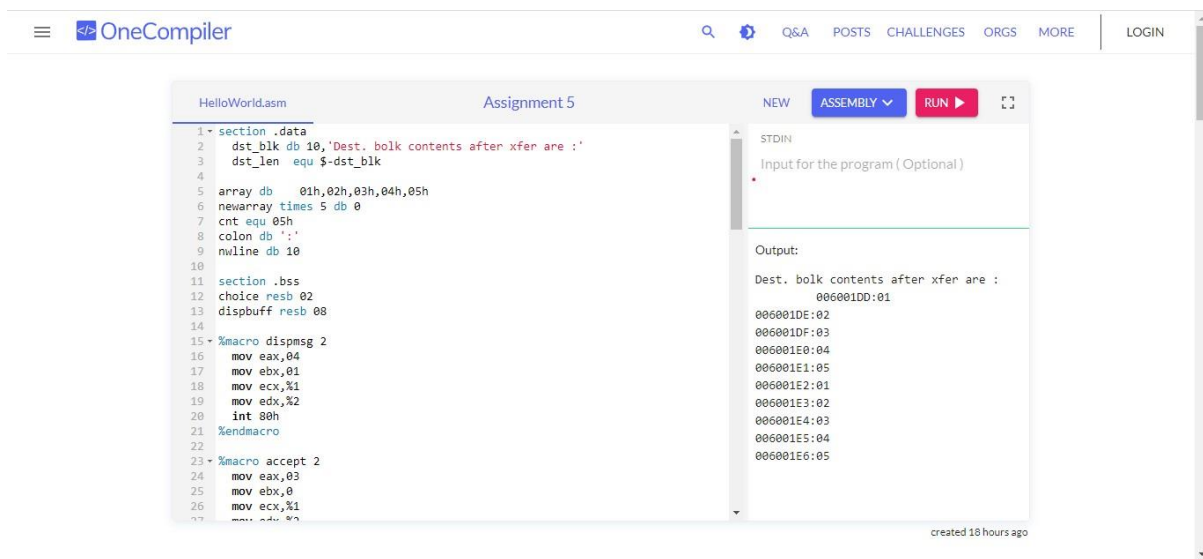
add al,07h

dskip2:

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

```
dispmsg dispbuff,8
ret
```

OUTPUT:



The screenshot shows the OneCompiler website interface. The main editor displays an assembly program named "HelloWorld.asm" under the tab "Assignment 5". The code includes data and bss sections, a macro for displaying a message, and a main function. The output window on the right shows the program's execution results.

```
1 section .data
2     dst_blk db 10,'Dest. bolk contents after xfer are : '
3     dst_len equ $-dst_blk
4
5     array db 01h,02h,03h,04h,05h
6     newarray times 5 db 0
7     cnt equ 05h
8     colon db ':'
9     newline db 10
10
11 section .bss
12 choice resb 02
13 dispbuff resb 08
14
15 %macro dispmsg 2
16     mov eax,04
17     mov ebx,01
18     mov ecx,%1
19     mov edx,%2
20     int 08h
21 %endmacro
22
23 %macro accept 2
24     mov eax,03
25     mov ebx,0
26     mov ecx,%1
27     mov edx,%2
```

Output:

```
Dest. bolk contents after xfer are :
006001D0:01
006001DE:02
006001DF:03
006001E0:04
006001E1:05
006001E2:01
006001E3:02
006001E4:03
006001E5:04
006001E6:05
```

created 18 hours ago

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:

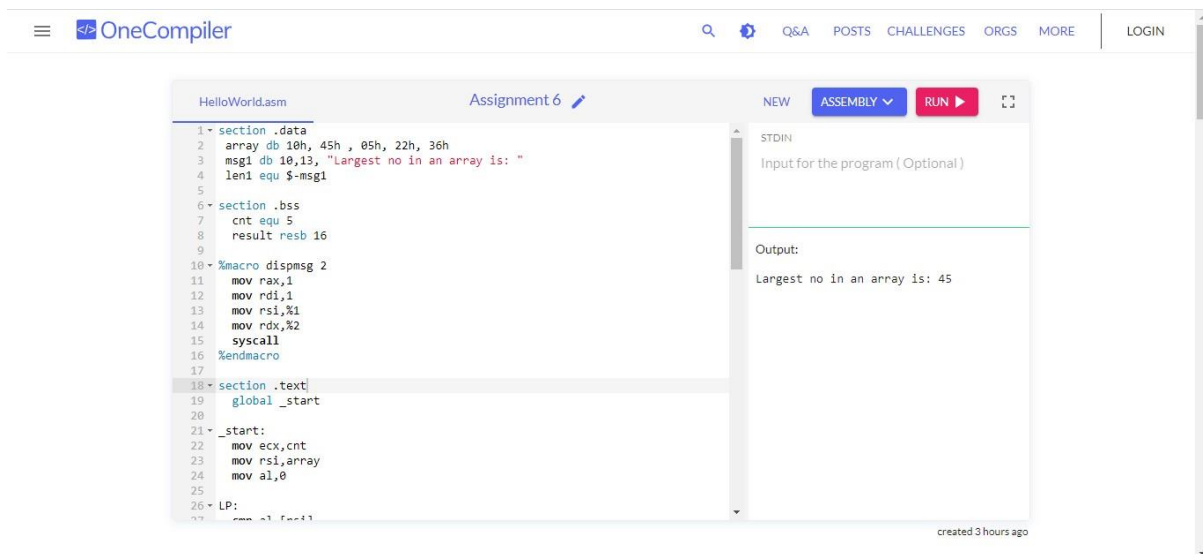
add al,37h

skip1:

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

ret

OUTPUT:



The screenshot shows the OneCompiler web interface. The top navigation bar includes a menu icon, the OneCompiler logo, and links for Q&A, POSTS, CHALLENGES, ORGS, MORE, and LOGIN. The main workspace is titled 'HelloWorld.asm' and 'Assignment 6'. It contains an assembly program that defines a data section with an array of 5 elements (10h, 45h, 05h, 22h, 36h) and a message 'Largest no in an array is: '. The program uses a macro to display the message and the largest number in the array (45). The output section shows the result: 'Largest no in an array is: 45'.

```
1 section .data
2 array db 10h, 45h, 05h, 22h, 36h
3 msg1 db 10,13, "Largest no in an array is: "
4 len1 equ $-msg1
5
6 section .bss
7 cnt equ 5
8 result resb 16
9
10 %macro dispmsg 2
11 mov rax,1
12 mov rdi,1
13 mov rsi,%1
14 mov rdx,%2
15 syscall
16 %endmacro
17
18 section .text
19 global _start
20
21 _start:
22 mov ecx,cnt
23 mov rsi,array
24 mov al,0
25
26 LP:
27 cmp al,0
```

Output:

Largest no in an array is: 45

created 3 hours ago

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

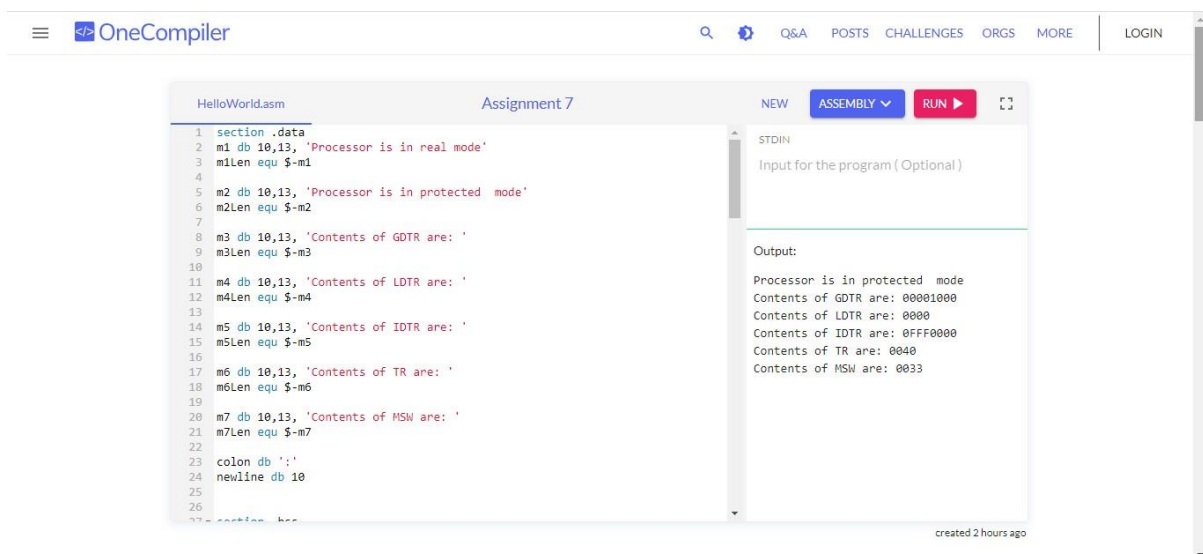
```
disp newline,1
jmp exit
```

```
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:



The screenshot shows the OneCompiler website interface. The top navigation bar includes a menu icon, the OneCompiler logo, a search icon, and links for Q&A, POSTS, CHALLENGES, ORGS, MORE, and LOGIN. The main content area displays an assembly program titled "HelloWorld.asm" under the heading "Assignment 7". The program code is as follows:

```
1 section .data
2 m1 db 10,13, 'Processor is in real mode'
3 m1len equ $-m1
4
5 m2 db 10,13, 'Processor is in protected mode'
6 m2len equ $-m2
7
8 m3 db 10,13, 'Contents of GDTR are: '
9 m3len equ $-m3
10
11 m4 db 10,13, 'Contents of LDTR are: '
12 m4len equ $-m4
13
14 m5 db 10,13, 'Contents of IDTR are: '
15 m5len equ $-m5
16
17 m6 db 10,13, 'Contents of TR are: '
18 m6len equ $-m6
19
20 m7 db 10,13, 'Contents of MSW are: '
21 m7len equ $-m7
22
23 colon db ':'
24 newline db 10
25
26
27 section .bss
```

On the right side, there are buttons for "NEW", "ASSEMBLY", and "RUN". Below these, the "STDIN" section shows "Input for the program (Optional)". The "Output" section displays the following results:

```
Processor is in protected mode
Contents of GDTR are: 00001000
Contents of LDTR are: 0000
Contents of IDTR are: 0FFF0000
Contents of TR are: 0040
Contents of MSW are: 0033
```

At the bottom right, it says "created 2 hours ago".

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:

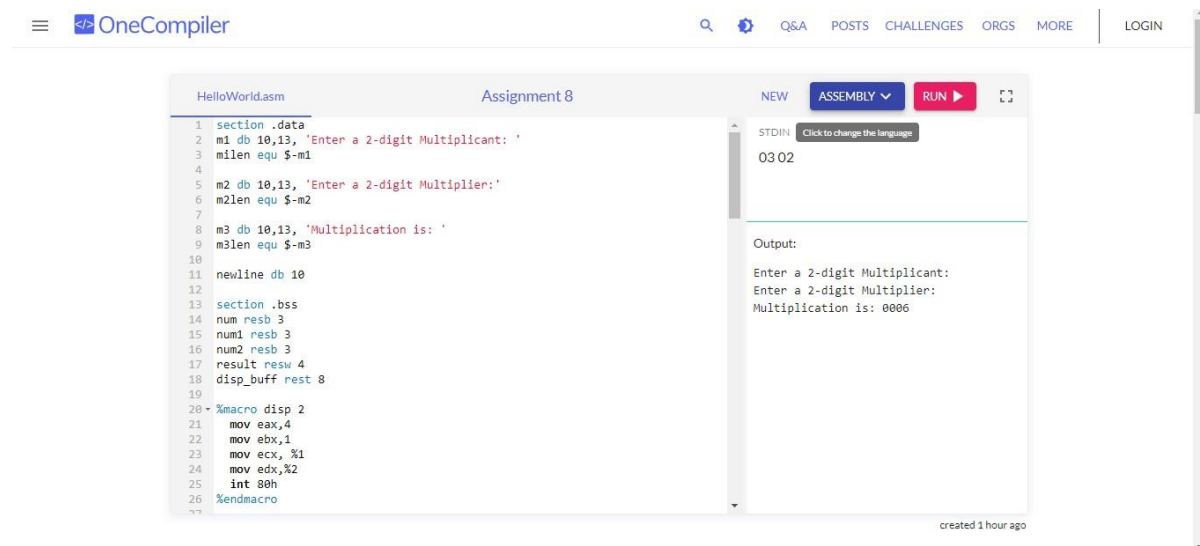
```
rol bx,4
mov al,bl
and al,0fh
cmp al,09
jbe y
add al,07h
```

y:

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

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

OUTPUT:



The screenshot shows the OneCompiler website interface. The main editor displays an assembly program named "HelloWorld.asm" with the following code:

```
1 section .data
2 m1 db 10,13, 'Enter a 2-digit Multiplicant: '
3 m1len equ $-m1
4
5 m2 db 10,13, 'Enter a 2-digit Multiplier:'
6 m2len equ $-m2
7
8 m3 db 10,13, 'Multiplication is: '
9 m3len equ $-m3
10
11 newline db 10
12
13 section .bss
14 num resb 3
15 num1 resb 3
16 num2 resb 3
17 result resw 4
18 disp_buff resb 8
19
20 %macro disp 2
21     mov eax,4
22     mov ebx,1
23     mov ecx,%1
24     mov edx,%2
25     int 80h
26 %endmacro
```

The right sidebar shows the "STDIN" input as "03 02" and the "Output" as:

```
Enter a 2-digit Multiplicant:
Enter a 2-digit Multiplier:
Multiplication is: 0006
```

The interface also includes a navigation bar with links for Q&A, POSTS, CHALLENGES, ORGS, MORE, and LOGIN, and a "NEW" button with a dropdown menu set to "ASSEMBLY". A "RUN" button is also present. The bottom right corner indicates the code was "created 1 hour ago".

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:

The screenshot shows the OneCompiler web interface. The main editor displays an assembly file named 'HelloWorld.asm' with the following code:

```
1 section .data
2 m1 db 10,"Enter No:."
3 m1len equ $-m1
4
5 m2 db 10,"Factorial is:."
6 m2len equ $-m2
7
8 newline db 10
9
10 section .bss
11 num1 resb 3
12 num2 resb 3
13 dispbuff resq 1
14
15 %macro cmn 4
16     mov rax,%1
17     mov rdi,%2
18     mov rsi,%3
19     mov rdx,%4
20     syscall
21 %endmacro
22
23 section .text
24 global _start
25 _start:
26     cmn 1,1,m1,m1len
27     cmn 0,0,num1,3
```

On the right side, the 'STDIN' input is '03'. The 'Output' section shows the program's execution results:

```
Output:
Enter No.:
Factorial is::0000000000000006
```

The interface also includes a top navigation bar with links for Q&A, POSTS, CHALLENGES, ORGS, MORE, and LOGIN. The bottom right corner indicates the code was 'created 1 hour ago'.