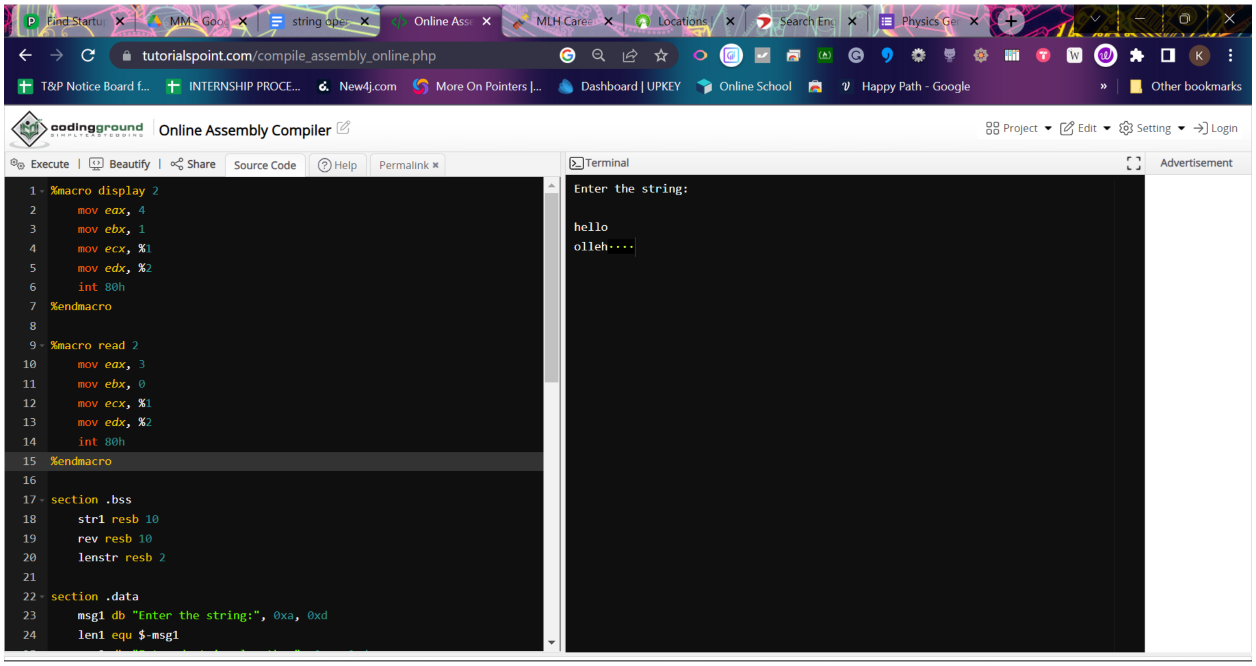
```
%macro display 2
   mov eax, 4
   mov ebx, 1
   mov ecx, %1
   mov edx, %2
   int 80h
%endmacro

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

section .bss
   str1 resb 10
   rev resb 10
   lenstr resb 2

section .data
   msg1 db "Enter the string:", 0xa, 0xd
   len1 equ $-msg1
   msg2 db "Entered string length: ", 0xa, 0xd
   len2 equ $-msg2
   msg3 db "Entered string's reverse string: ", 0xa, 0xd
   len3 equ $-msg3

section .text
   global _start
_start:
   display msg1, len1
   read str1, 10
   jmp reverse
   display msg3, len3
   jmp exit
reverse:
   mov esi, str1
   mov edi, rev
   mov cl, al
   add esi, eax
   dec esi
   L1:
        mov al, byte[esi]
        mov byte[edi], al
        inc edi
        dec esi
        dec cl
        jnz L1
        display rev,10
        int 80h

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



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

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

section .bss
    str1 resb 10
    rev resb 10
    lenstr resb 2

section .data
    msg1 db "Enter the string:", 0xa, 0xd
    len1 equ $-msg1
```

```
Enter the string:

hello
olleh····
```

Length of the string

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

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

section .bss
    str1 resb 10
    rev resb 10
    lenstr resb 2
    qumo resb 1
    rem resb 1

section .data
    msg1 db "Enter the String",0xa,0xd
    len1 equ $-msg1
    msg2 db "Enter String length is",0xa,0xd
    len2 equ $-msg2
    msg3 db "Enter String reverse is",0xa,0xd
    len3 equ $-msg3
    msg4 db "Enter the String",0xa,0xd
    len4 equ $-msg4


section .text
    global _start
_start:
    display msg1, str1
    read str1,10
    jmp len

len:
    dec al
    mov bl, 10
    div bl
    add al, 30h
    add ah, 30h
    mov [qumo],al
    mov [rem], ah
    display msg2,len2
    display qumo, 1
    display rem, 1
    int 80h
    jmp exit
```

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

Upper to lower

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

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

; upper to lower
section .bss
   str1 resb 10
   str2 resb 10
   rev resb 10
   lenstr resb 2
   quotient resb 1
   remainder resb 1

section .data
   msg1 db "Enter string: ", 10, 0
   len1 equ $-msg1
   msg2 db "String length: ", 10,0
   len2 equ $-msg2
   msg3 db "Reverse string; ", 10, 0
   len3 equ $-msg3

section .text
global _start
```

```
_start:
    display msg1, len1
    read str1, 10
    jmp lower
lower:
    dec al
    mov esi, str1
    mov edi, str2
    mov cl,al

up:
    mov al,byte[esi]
    cmp al, 'A'
    jb _store
    cmp al, 'Z'
    ja _store
    add al, ' '

_store:
    mov byte[edi], al
    inc edi
    inc esi
    dec cl
    jnz up

    display str2,10

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

Toggle

```
;toggle
%macro display 2
 mov eax, 4
 mov ebx, 1
 mov ecx, %1
 mov edx, %2
 int 80h
%endmacro

%macro read 2
```

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

section .bss
    str1 resb 10
    str2 resb 10
    rev resb 10
    lenstr resb 2
    quotient resb 2
    remainder resb 1

section .data
    msg1 db "Enter the String",0xa,0xd
    len1 equ $-msg1
    msg2 db "Enter String length is",0xa,0xd
    len2 equ $-msg2
    msg3 db "Enter String reverse is",0xa,0xd
    len3 equ $-msg3
    msg4 db "Enter the String",0xa,0xd
    len4 equ $-msg4

section .text
global _start
_start:
    display msg1, len1
    read str1, 10
    jmp toggle

toggle:
    dec al
    mov esi, str1
    mov edi, str2
    mov cl, al

above:
    mov al, byte[esi]
    cmp al, 'a'
    jae down
    cmp al,'A'
    jae down1
```

```
down:
   cmp al,'z'
   jbe tog1
down1:
   cmp al,'Z'
   jbe tog2

tog1:
   sub al, ' '
   mov byte[edi], al

tog2:
   add al,20h
   mov byte[edi],al

skip:
   inc esi
   inc edi
   dec cl
   jnz above

   display str2,10
   mov eax, 1
   mov ebx, 0
   int 80h
```