Reverse ,pallondrom, concate, **length**, L**ower to upper , u to l, toggle**, substring

Reverse Code

```
%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 "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 rever
      display msg3,len3
      jmp exit


rever:
      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
```

Length

```
%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,len1
      read str1,10
      ; jmp rever
      jmp len
      ; jmp exit
```

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
```

Lower to upper

```
%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
    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,len1
        read str1,10

        ;jmp lower
        jmp upper
        ; jmp rever
        ; jmp len
        ; jmp  rever
        ; jmp exit


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

_up:
   mov al,byte[esi]
   cmp al,61h
   jb _store

   cmp al,7Ah
```

```
   ja _store

   sub al,20h

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

display str2,10
```

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
    mov edx,%2
    int 80h

%endmacro



section .bss
    str1 resb 10
    str2 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,len1
        read str1,10

        jmp lower
        ; jmp upper
        ; jmp rever
        ; jmp len
        ; jmp  rever
        ; jmp exit

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

up:
  mov al,byte[esi]
  cmp al,41h
  jb _store
  cmp al,5Ah
  ja _store
  add al,20h
```

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

 display str2,10
```

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
   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,len1
      read str1,10

      jmp toggel
      ; jmp lower
      ; jmp upper
      ; jmp rever
      ; jmp len
      ; jmp  rever
      ; jmp exit

toggel:
   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,20h
  mov byte[edi],al
  jmp skip

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

skip:
  inc esi;
  inc edi;
  dec cl
  jnz above

 display str2,10
```

Palliondrom

```
%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 10
    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 the String Is not palliondrom",0xa,0xd
    len3 equ $-msg3
    msg4 db "Enter the String Is palliondrom",0xa,0xd
    len4 equ $-msg4

section .text

   global _start
   _start:

        display msg1,len1
        read str1,10
        jmp rever

rever:
        dec al
        mov byte[lenstr], al
        mov esi,str1
```

```
        mov edi,rev
        mov cl,byte[lenstr]
        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

        ;   jmp pall


pall:
        mov esi,str1
        mov edi,rev
        mov cl,byte[lenstr]

up2:
          mov al, byte[esi]
          cmp al, byte[edi]
          jne Npall
          inc esi
          inc edi
          dec cl
          jnz up2

Ypall:
 display msg4,len4
 jmp exit

Npall:
  display msg3,len3
  jmp exit

exit:
 mov eax,1
```

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

Concate 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
    str2 resb 10
    rev resb 10
    lenstr resb 10
    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 the String Is not palliondrom",0xa,0xd
    len3 equ $-msg3
    msg4 db "Enter the String Is palliondrom",0xa,0xd
    len4 equ $-msg4

section .text

   global _start
   _start:

        display msg1,len1
        read str1,10
        ; jmp rever

Concstr1:
        dec al
        mov byte[lenstr], al
        mov esi,str1
        mov edi,rev
        mov cl,byte[lenstr]
        L1:
          mov al,byte[esi]
          mov byte[edi],al
          inc edi
          inc esi
          dec cl
          jnz L1

Concstr2:

        display msg1,len1
        read str1,10
        dec al
        mov byte[lenstr], al
        mov esi,str1
```

```
        mov cl,byte[lenstr]
        L2:
          mov al,byte[esi]
          mov byte[edi],al
          inc edi
          inc esi
          dec cl
          jnz L2

        display rev,10



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