Práctica 6

# Objetivo

Identificar los modos de direccionamiento adecuados para manejo de memoria en aplicaciones de sistemas basados en microprocesador mediante la distinción de su funcionamiento, de forma lógica y responsable.

# Desarrollo

1. Copie el código del Listado 1 en un archivo llamado entrada.asm. Abra una terminal en Linux y ensamble el código con NASM por medio del comando: nasm -f elf entrada.asm

El cual generará el archivo objeto entrada.o.

Encadene el archivo por medio de uno de los siguientes comandos: a) En un sistema operativo de 32 bits: ld -s -o entrada entrada.o

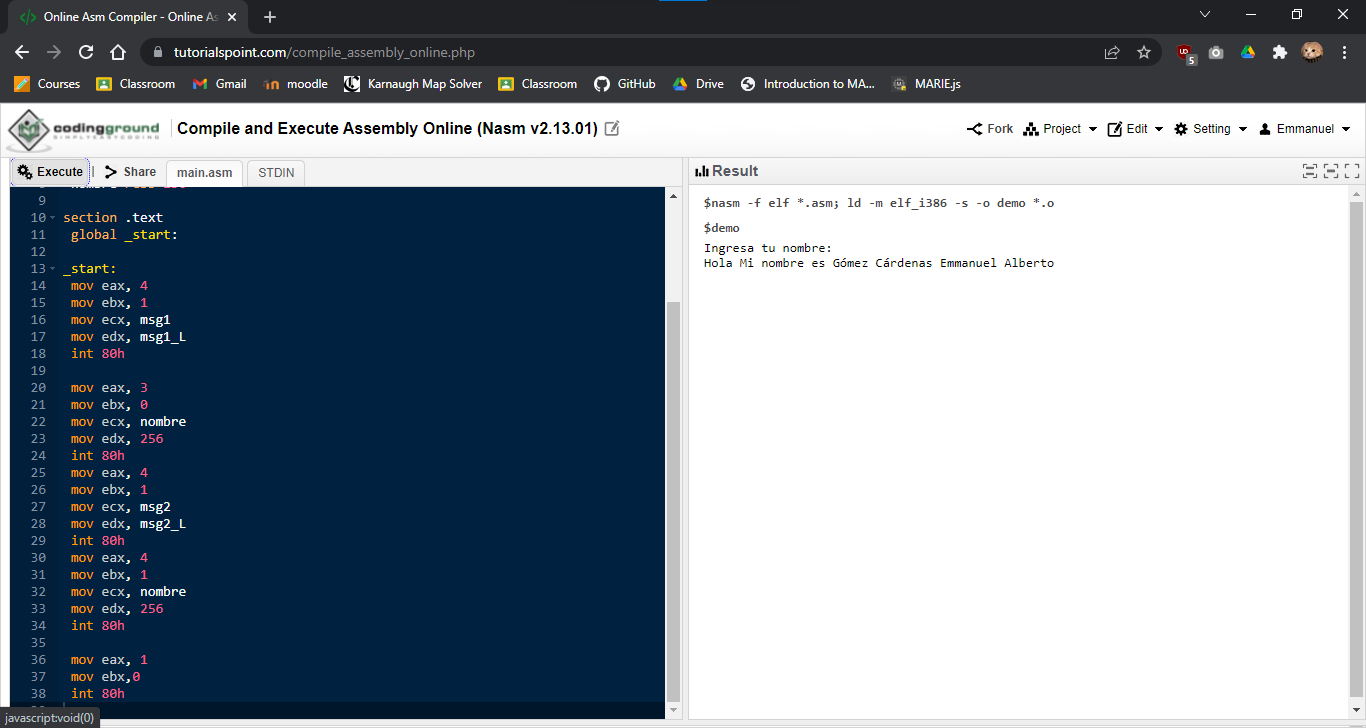
b) En un sistema operativo de 64 bits:

ld -m elf\_i386 -s -o entrada entrada.o

El cual generará el archivo ejecutable entrada. Ejecute el archivo por medio del comando:

./entrada

El programa solicita al usuario el ingreso de su nombre y despliega un mensaje de saludo.



1. Cree un programa llamado **P6.asm** que contenga las instrucciones necesarias para hacer lo que se indica a continuación:
   1. Reservar dos espacios en memoria no inicializados, uno de 64 bytes etiquetado como A y el otro de 1 byte etiquetado como N.
   2. Solicitar una cadena que se almacene en A.
   3. Copiar el primer carácter de A en la variable N. Use un modo de direccionamiento base.
   4. Reemplazar el tercer carácter A por un guion -, usando un modo de direccionamiento base con desplazamiento.
   5. Reemplazar el octavo carácter de A por una X usando un direccionamiento base con índice escalado.
   6. Copiar el segundo carácter de A y almacenarlo en los bits 15-8 del acumulador.
   7. Reemplazar el noveno carácter de A por el carácter en los bits 15-8 del acumulador, usando un direccionamiento base con índice escalado y desplazamiento.

Por cada inciso, despliegue en pantalla el nuevo valor de la variable modificada.

# Código

section .data ; Datos inicializados

    NL: db 13, 10

    NL\_L: equ $-NL

section .bss ; Datos no inicializados

   ;Reservar dos espacios en memoria no inicializados, uno de 64 bytes etiquetado como A.

   A resb 64

   ;El otro de 1 byte etiquetado como N.

   N resb 1

   ;Cadena temporal utilizada para imprimir

   cad resb 16

section .text

   global \_start:

\_start:

   ;Solicitar una cadena que se almacene en A.

   mov eax, 3  ;servicio

   mov ebx, 0  ;Entrada

   mov ecx, A  ;Cadena

   mov edx, 64 ;Caracteres

   int 80h

   call printA

   call printNl

   ;Copiar el primer carácter de A en la variable N. Use un modo de direccionamiento base.

   xor eax,eax     ;Limpiamos EAX

   mov ebx, A  ;Apuntamos las cadenas a manipular

   mov edx, N

   mov al, [ebx] ;obtenemos el primer caracter de A

   mov [edx], al ;Lo guardamos en N

   call printN

   call printNl

   call printNl

   ;Reemplazar el tercer carácter A por un guion -, usando un modo de direccionamiento base con desplazamiento

   mov byte [ebx+2], '-'

   call printA

   call printNl

   ;Reemplazar el octavo carácter de A por una X usando un direccionamiento base con índice escalado.

   mov esi,1

   mov byte [ebx+esi\*8], 'x'

   call printA

   call printNl

   ;Copiar el segundo carácter de A y almacenarlo en los bits 8-15 del acumulador

   xor eax,eax  ;Limpiamos los registros

   xor edx,edx

   mov ecx,8

   mov dl, [ebx+1]   ;Obtenemos el segundo caracter de A

   add al,dl   ;Lo guardaamos en A

   shl eax,cl   ;Lo recorremos para guardar el valor en los bits de 8 a 15

   call printHex

   call printNl

   ;Reemplazar el noveno carácter de A por el carácter en los bits 8-15 del acumulador, usando un direccionamiento base con índice escalado y desplazamiento.

   mov cl,8

   shr eax,cl

   mov [ebx+esi\*8+1], al

   call printA

   call printNl

   ;terminar proceso

   mov eax, 1

   mov ebx,0

   int 80h

   printA:            ;Imprime A

   pushad

   mov eax, 4

   mov ebx, 1

   mov ecx, A

   mov edx, 64

   int 80h

   popad

   ret

   printN:            ;Imprime N

   pushad

   mov eax, 4

   mov ebx, 1

   mov ecx, N

   mov edx, 1

   int 80h

   popad

   ret

   printNl:            ;Imprime un salto de linea

   pushad

   mov eax, 4

   mov ebx, 1

   mov ecx, NL

   mov edx, NL\_L

   int 80h

   popad

   ret

   printHex:

   pushad

   mov esi,cad

   mov edx, eax

   mov ebx, 0fh

   mov cl, 28

.nxt: shr eax,cl

.msk: and eax,ebx

   cmp al, 9

   jbe .menor

   add al,7

.menor:add al,'0'

   mov byte [esi],al

   inc esi

   mov eax, edx

   cmp cl, 0

   je .print

   sub cl, 4

   cmp cl, 0

   ja .nxt

   je .msk

.print: mov eax, 4

   mov ebx, 1

   sub esi, 8

   mov ecx, esi

   mov edx, 8

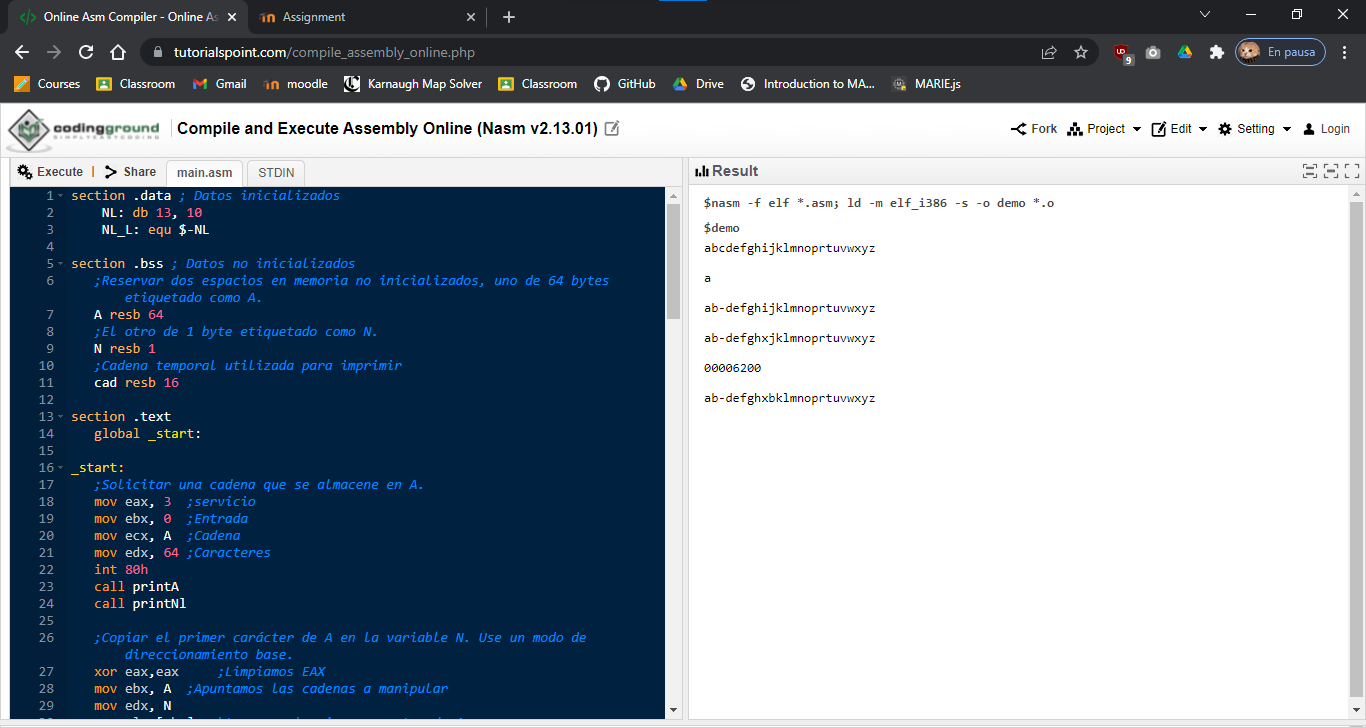
   int 80h

   call printNl

   popad

   ret

# 



# Conclusiones y comentarios

Es interesante como en esta versión de ensamblador es un poco más sencillo e intuitivo el pedir datos al usuario y la reservación de memoria respecto a tasm.

# Dificultades en el desarrollo

Debido a que la computadora utilizada para ejecutar la máquina virtual no es potente, la mayor complicación al momento de realizar la practica fueron las partes de ensamblar y ejecutar los códigos.