



## 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.

```
9
10 section .text
11 global _start:
12
13 _start:
14 mov eax, 4
15 mov ebx, 1
16 mov ecx, msg1
17 mov edx, msg1_L
18 int 80h
19
20 mov eax, 3
21 mov ebx, 0
22 mov ecx, nombre
23 mov edx, 256
24 int 80h
25 mov eax, 4
26 mov ebx, 1
27 mov ecx, msg2
28 mov edx, msg2_L
29 int 80h
30 mov eax, 4
31 mov ebx, 1
32 mov ecx, nombre
33 mov edx, 256
34 int 80h
35
36 mov eax, 1
37 mov ebx, 0
38 int 80h
```

**Result**

```
$nasm -f elf *.asm; ld -m elf_i386 -s -o demo *.o
$demo
Ingresa tu nombre:
Hola Mi nombre es Gómez Cárdenas Emmanuel Alberto
```

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

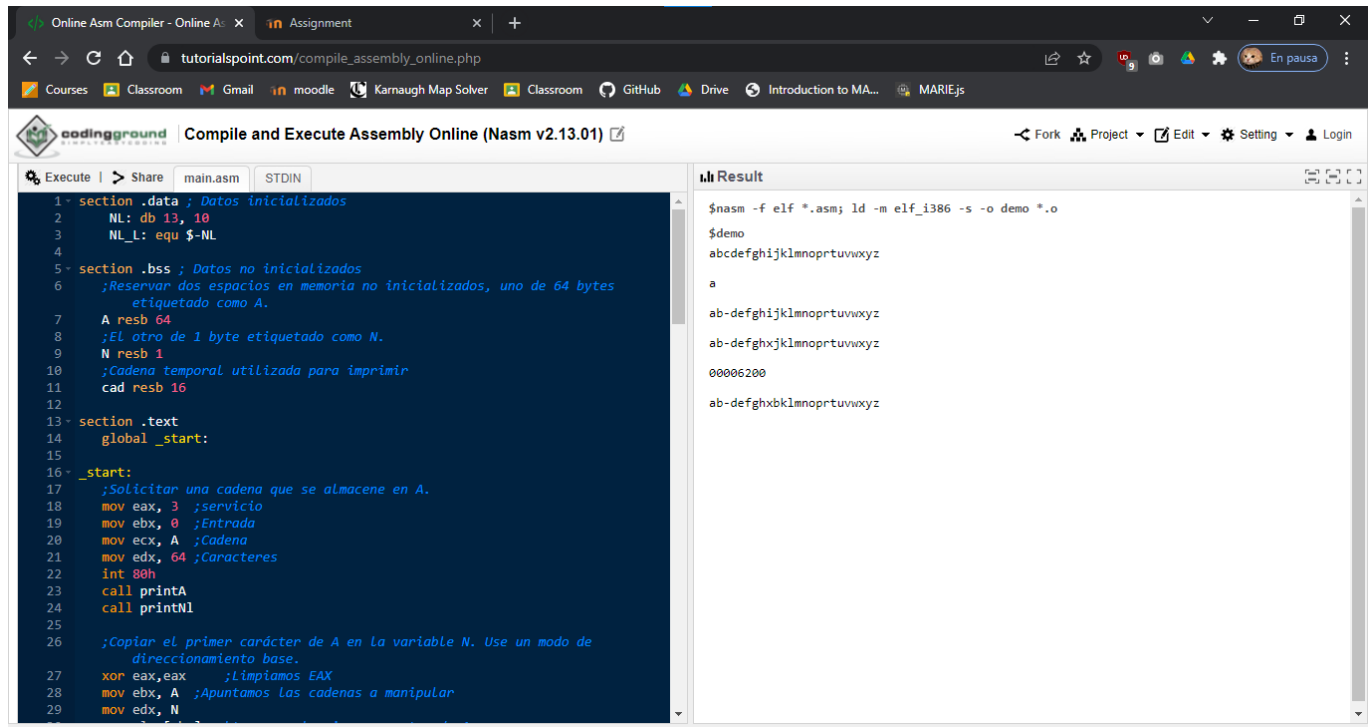
printN:                ;Imprime N
pushad
mov eax, 4
mov ebx, 1
mov ecx, N
mov edx, 1
int 80h
popad
ret

printNl:                ;Imprime un salto de linea
pushad
mov eax, 4
mov ebx, 1
mov ecx, NL
mov edx, NL_L
int 80h
popad
ret

prinHex:
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
```



The screenshot shows a web browser window with the URL `tutorialspoint.com/compile_assembly_online.php`. The page title is "Compile and Execute Assembly Online (Nasm v2.13.01)". The interface has a dark theme. On the left, there's a code editor with assembly code. On the right, there's a "Result" panel showing the output of the assembly process.

**Assembly Code (main.asm):**

```
1 section .data ; Datos inicializados
2   NL: db 13, 10
3   NL_L: equ $-NL
4
5 section .bss ; Datos no inicializados
6   ;Reservar dos espacios en memoria no inicializados, uno de 64 bytes
   etiquetado como A.
7   A resb 64
8   ;El otro de 1 byte etiquetado como N.
9   N resb 1
10  ;Cadena temporal utilizada para imprimir
   cad resb 16
11
12
13 section .text
14   global _start:
15
16 _start:
17   ;Solicitar una cadena que se almacene en A.
18   mov eax, 3 ;servicio
19   mov ebx, 0 ;Entrada
20   mov ecx, A ;Cadena
21   mov edx, 64 ;Caracteres
22   int 80h
23   call printA
24   call printNl
25
26   ;Copiar el primer carácter de A en la variable N. Use un modo de
   direccionamiento base.
27   xor eax, eax ;Limpiamos EAX
28   mov ebx, A ;Apuntamos las cadenas a manipular
29   mov edx, N
```

**Result:**

```
$nasm -f elf *.asm; ld -m elf_i386 -s -o demo *.o
$demo
abcdefghijklmnopqrstuvwxyz
a
ab-defghijklmnoprtuvvxyz
ab-defghxjklmnoprtuvvxyz
00006200
ab-defghxbklmnoprtuvvxyz
```

## 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.