



## Práctica 7

---

### Objetivo

Seleccionar los modos de direccionamiento adecuados para manejo de memoria y las instrucciones aritméticas y de transferencia de datos 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 Ej\_Lib.asm. Descargue de Moodle la biblioteca de funciones libpc\_io (archivos libpc\_io.a y pc\_io.inc). Abra una terminal en Linux y ensamble el código con NASM por medio del comando: `nasm -f elf`

Ej\_Lib.asm

El cual generará el archivo objeto Ej\_Lib.o.

Encadene el archivo por medio de uno de los siguientes comandos: a) En un sistema operativo de 32 bits:

```
ld -s -o Ej_Lib Ej_Lib.o libpc_io.a
```

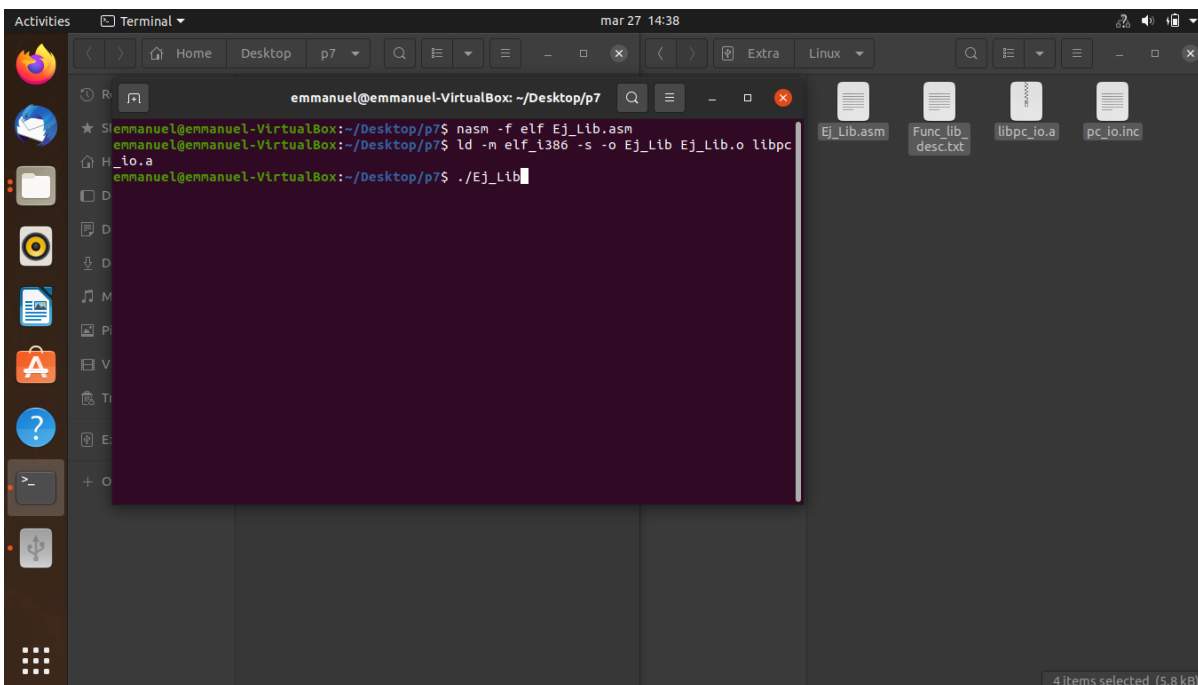
b) En un sistema operativo de 64 bits:

```
ld -m elf_i386 -s -o Ej_Lib Ej_Lib.o libpc_io.a
```

El cual generará el archivo ejecutable Ej\_Lib. Ejecute el archivo por medio del comando:

```
./Ej_Lib
```

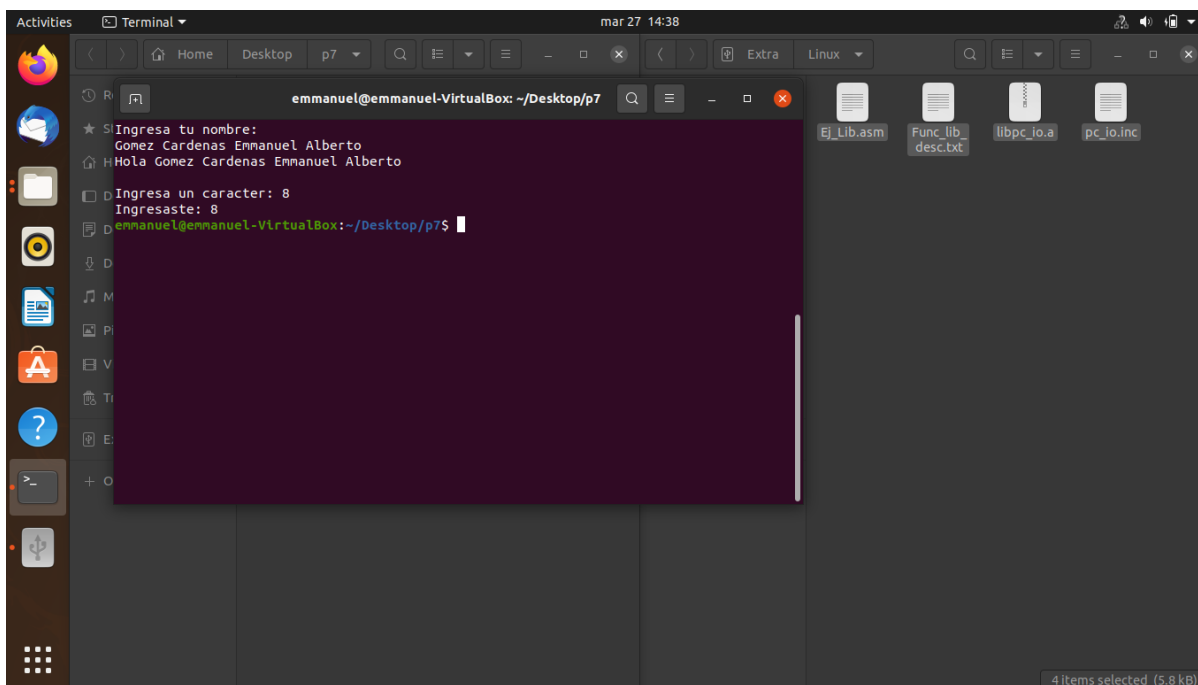
El programa solicita al usuario el ingreso de su nombre y despliega un mensaje de saludo. Posteriormente, solicita un carácter y lo despliega en pantalla.



A screenshot of a Linux terminal window titled "Terminal" with the date "mar 27 14:38". The terminal shows the following commands and output:

```
emmanuel@emmanuel-VirtualBox: ~/Desktop/p7
$ nasm -f elf Ej_Lib.asm
$ ld -m elf_i386 -s -o Ej_Lib Ej_Lib.o libpc
$ ./Ej_Lib
```

The terminal window is open over a file manager showing files: Ej\_Lib.asm, Func\_lib\_desc.txt, libpc\_io.a, and pc\_io.inc. The status bar at the bottom right indicates "4 items selected (5.8 kB)".



A screenshot of a Linux terminal window titled "Terminal" with the date "mar 27 14:38". The terminal shows the following interaction:

```
emmanuel@emmanuel-VirtualBox: ~/Desktop/p7
$ ./Ej_Lib
Ingresaste: 8
```

The terminal window is open over a file manager showing files: Ej\_Lib.asm, Func\_lib\_desc.txt, libpc\_io.a, and pc\_io.inc. The status bar at the bottom right indicates "4 items selected (5.8 kB)".

2. Cree un programa llamado **P7.asm** que contenga la rutina *printHex* de la Práctica 5, la cual recibe en EAX un valor que se quiere imprimir en formato hexadecimal. Agregue a **P7.asm** las instrucciones necesarias para hacer lo que se indica a continuación:
  - a) Reservar tres espacios en memoria no inicializados, uno de 64 bytes etiquetado como A, otro de 4 bytes etiquetado como N, y el último de 1 byte etiquetado como C.
  - b) Solicitar una cadena que se almacene en A.
  - c) Solicitar un carácter y almacenarlo en C.

- d) Reemplazar el catorceavo carácter de A por el carácter en C. Use un modo de direccionamiento base con índice escalado más desplazamiento.
- e) Almacenar 0xABF358C1 en N.
- f) Insertar N en la pila.
- g) Reste a N su matrícula (use el valor como si fuera hexadecimal).
- h) Almacenar 0x6F4395B2 en ECX.
- i) Intercambiar ECX y el valor de 32 bits en el tope de la pila.
- j) Aumentar 1 a ECX sin ADD o ADC.


3. Realizar lo que se indica a continuación:

- k) Reservar en memoria no inicializada, las siguientes variables 'a', 'b', 'c', 'd', 'e' y 'f' cada una de 4 bytes.
- l) Almacenar su matrícula como hexadecimal en a.
- m) Realizar lo siguiente (a+1) sin ADD o ADC y almacenarlo en a.
- n) Almacenar N en b.
- o) Realizar lo siguiente (b-1) sin SUB o SBB y almacenarlo en b.
- p) Almacenar en c el diecisieteavo carácter de A.
- q) Almacenar en eax 0xA0B0000.
- r) Almacenar en d los primeros 16 bits más significativos de eax en la parte menos significativa de d.
- s) Almacenar en e un 0x20.
- t) Almacenar en f un 0x10.
- u) Realizar lo siguiente (a+b) y almacenarlo en eax.
- v) Pasar los primeros 16 bits más significativos a la parte menos significativa de eax (antes XXXX0000, después 0000XXXX), poner en cero los demás.
- w) Multiplicar eax con c.
- x) Reducir eax a únicamente 16 bits con corrimientos a la derecha.
- y) Restar eax con d.
- z) Multiplicar eax con e.
- aa) Dividir el resultado del inciso z) con f.
- bb) Mostrar únicamente el resultado del inciso aa) con printf.

Por cada inciso, despliegue en pantalla el nuevo valor de la variable o registro modificado. Haga uso de la rutina *printf* para desplegar los valores numéricos, *puts* para desplegar cadenas y *putchar* para caracteres.

Agregar antes de cada impresión una cadena que indique que inciso es el que se está mostrando en consola, ejemplo "inciso z) 00XX00XX inciso aa) XXXX0000" etc.

## RESULTADO EN CONSOLA



```
!. /p7

Inciso b)
XXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXX

Inciso c)
A
A

Inciso d)
XXXXXXXXXXXXXXXXXXXXX
A
XXXXXXXXXXXXAXXXXXX

Inciso e)
ABF358C1

Inciso f)
ABF358C1

Inciso g)
AACD43B8

Inciso h)
6F4395B2
```

Inciso i)  
6F4395B2  
ABF358C1

Inciso j)  
ABF358C2

Inciso l)  
01261509

Inciso m)  
0126150A

Inciso n)  
AACD43B8

Inciso o)  
AACD43B7

Inciso p)  
XXXXXXXXXXXXAXXXXXX  
X

Inciso q)  
0A0B0000



Inciso r)  
0A0B0000  
0000A0B

Inciso s)  
00000020

Inciso t)  
00000010

Inciso u)  
AACD43B7  
0126150A  
ABF358C1

Inciso v)  
0000ABF3

Inciso w)  
0000ABF3  
00000058  
003B1B88

Inciso x)  
00003B1B

Inciso y)  
00003B1B  
0000A0B  
00003110

Inciso z)  
00003110  
00000020  
00000000  
00062200

Inciso aa  
00062200  
00000010  
00006220

Inciso bb  
00006220

## CÓDIGO

```
%include "pc_io.inc"
section .data
    NL: db 13, 10
    NL_L: equ $-NL
    ClauseText: db 10,13,"Inciso ",0
    ClauseLetter: db "b)",10,13,0
section .bss
    ;a) Reservar tres espacios en memoria no inicializados, uno de 64 bytes etiquetado como A, otro
    de 4 bytes etiquetado como N, y el último de 1 byte etiquetado como C.
    A resb 64 ;A = 64 bytes
    N resb 4 ;N = 4 bytes
    C resb 1 ;C = 1 bytes
    ;Cadena temporal utilizada para imprimir
    cad resb 16
    ;k) Reservar en memoria no inicializada, las siguientes variables 'a', 'b', 'c', 'd', 'e' y 'f'
    cada una de 4 bytes.
    a resb 4
    b resb 4
    c resb 4
    d resb 4
    e resb 4
    f resb 4

section .text
global _start:
_start:
; b) Solicitar una cadena que se almacene en A.
    call printClause
    mov eax, 3 ;servicio
    mov ebx, 0 ;Entrada
    mov ecx, A ;Cadena
    mov edx, 64 ;Caracteres
    int 80h
    mov edx,A
    call puts

; c) Solicitar un caracter y almacenarlo en C
    call printClause
    call getche
    mov [C],al
    mov al,[C]
    call putchar
    call printNl

; d) Reemplazar el catorceavo caracter de A por el caracter en C. Use un modo de direccionamiento
base con indice escalado mas desplazamiento.
    call printClause
    mov edx,A
    call puts
    mov edx,C
    call putchar
    call printNl
    xor eax,eax
    mov esi,6
```

```
mov ebx, A
mov al,[C]
mov [ebx+esi*2+1], al
mov edx,A
call puts

; e) Almacenar 0xABF358C1 en N
call printClause
mov eax, 0ABF358C1h
mov [N], eax
xor eax, eax ;Limpiamos para validar
mov eax,[N] ;Obtenemos N
call printHex ;Imprimimos N
call printNl

; f) Insertar N en la pila.
call printClause
mov eax,[N] ;Obtenemos N
push eax ;La empujamos a la pila
call printHex ;Imprimimos N
call printNl

; g) Reste a N su matricula (use el valor como si fuera hexadecimal)
call printClause
mov eax, 1261509h
sub [N],eax
mov eax,[N]
call printHex
call printNl

; h) Almacenar 0x6F4395B2 en ECX.
call printClause
mov ecx, 6F4395B2h
mov eax,ecx
call printHex
call printNl

; i) Intercambiar ECX y el valor de 32 bits en el tope de la pila.
call printClause
mov eax,ecx ;Movemos eax a ecx
call printHex ;Imprimimos ecx
pop eax
call printHex ;Imprimimos el valor de la pila
push ecx ;Empujamos ecx a la pila
mov ecx,eax ;Movemos eax a ecx
call printNl

; j) Aumentar 1 a ECX sin ADD o ADC.
call printClause
inc ecx ;Incrementamos ecx
mov eax, ecx
call printHex ;Imprimimos ecx
call printNl

; l) Almacenar su matrícula como hexadecimal en a.
```

```
add byte [ClauseLetter],1
call printClause
mov eax, 1261509h ;Ponemos la matricula en eax
mov [a],eax       ;Movemos eax a 'a'
call printHex     ;Imprimimos eax (a)
call printNl

; m) Realizar lo siguiente (a+1) sin ADD o ADC y almacenarlo en a.
call printClause
mov eax,[a] ;Obtenemos a
inc eax    ;Incrementamos eax 1 unidad
mov [a],eax ;Guardamos el valor de eax en 'a'
call printHex ;Imprimimos eax (a)
call printNl

; n) Almacenar N en b.
call printClause
mov eax, [N]
mov [b], eax
call printHex
call printNl

; o) Realizar lo siguiente (b-1) sin SUB o SBB y almacenarlo en b.
call printClause
mov eax, [b]
dec eax ;b-1
mov [b], eax
call printHex
call printNl

; p) Almacenar en c el diecisieteavo carácter de A.
call printClause
mov edx,A
call puts
xor eax,eax
mov ebx,A
mov al, [ebx+16]
mov [c],al
call putchar
call printNl

; q) Almacenar en eax 0xA0B0000.
call printClause
mov eax, 0A0B0000h
call printHex
call printNl

; r) Almacenar en d los primeros 16 bits más significativos de eax en la parte menos significativa de d.
call printClause
call printHex
mov cl,16
shr eax,cl
mov [d],ax
mov eax,[d]
```



```
    call printHex
    call printNl

; s) Almacenar en e un 0x20.
    call printClause
    mov eax,20h
    mov [e], al
    call printHex
    call printNl

; t) Almacenar en f un 0x10.
    call printClause
    mov eax,10h
    mov [f], al
    call printHex
    call printNl

; u) Realizar lo siguiente (a+b) y almacenarlo en eax.
    call printClause
    mov eax,[b]
    call printHex
    mov eax,[a]
    call printHex
    add eax,[b]
    call printHex

; v) Pasar los primeros 16 bits más significativos a la parte menos significativa de eax (antes
XXXX0000, después 0000XXXX), poner en cero los demás.
    call printClause
    mov cl,16
    shr eax,cl
    call printHex

; w) Multiplicar eax con c.
    call printClause
    call printHex
    push eax
    mov eax,[c]
    call printHex
    pop eax
    mul dword[c]
    push eax
    call printHex

; x) Reducir eax a únicamente 16 bits con corrimientos a la derecha.
    call printClause
    mov cl,8
    shr eax,cl
    call printHex

; y) Restar eax con d.
    call printClause
    call printHex
    push eax
    mov eax,[d]
```

```
    call printHex
    pop eax
    sub eax,[d]
    call printHex

; z) Multiplicar eax con e.
    call printClause
    call printHex
    push eax
    mov eax,[e]
    call printHex
    pop eax
    mul dword[e]
    push eax
    mov eax,edx
    call printHex
    pop eax
    call printHex

; aa)Dividir el resultado del inciso z) con f.
    mov byte[ClauseLetter], 'a'
    mov byte[ClauseLetter+1], 'a'
    call printClause
    call printHex
    push eax
    mov eax,[f]
    call printHex
    pop eax
    xor edx,edx
    div dword[f]
    call printHex

; bb)Mostrar únicamente el resultado del inciso aa) con printHex.
    mov byte[ClauseLetter+1], 'b'
    call printClause
    call printHex

    mov eax,1
    mov ebx,0
    int 80h

printHex:
    pushad
    mov esi,cad
    mov edx, eax
    mov ebx, 0fh
    mov cl, 28
.next: 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
printNl:
    pushad
    mov eax, 4
    mov ebx, 1
    mov ecx, NL
    mov edx, NL_L
    int 80h
    popad
    ret
printClause:
    pushad
    mov edx, ClauseText
    call puts
    mov edx, ClauseLetter
    call puts
    add byte [ClauseLetter], 1
    popad
    ret
```

### Conclusiones y comentarios

El uso de bibliotecas es de gran ayuda ya que gracias a estas podemos evitar tener que escribir código desde 0 cada vez, de esta manera podemos enfocarnos en escribir la parte que realmente nos interesa y ser más eficientes al escribir código.