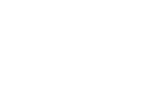


Práctica



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

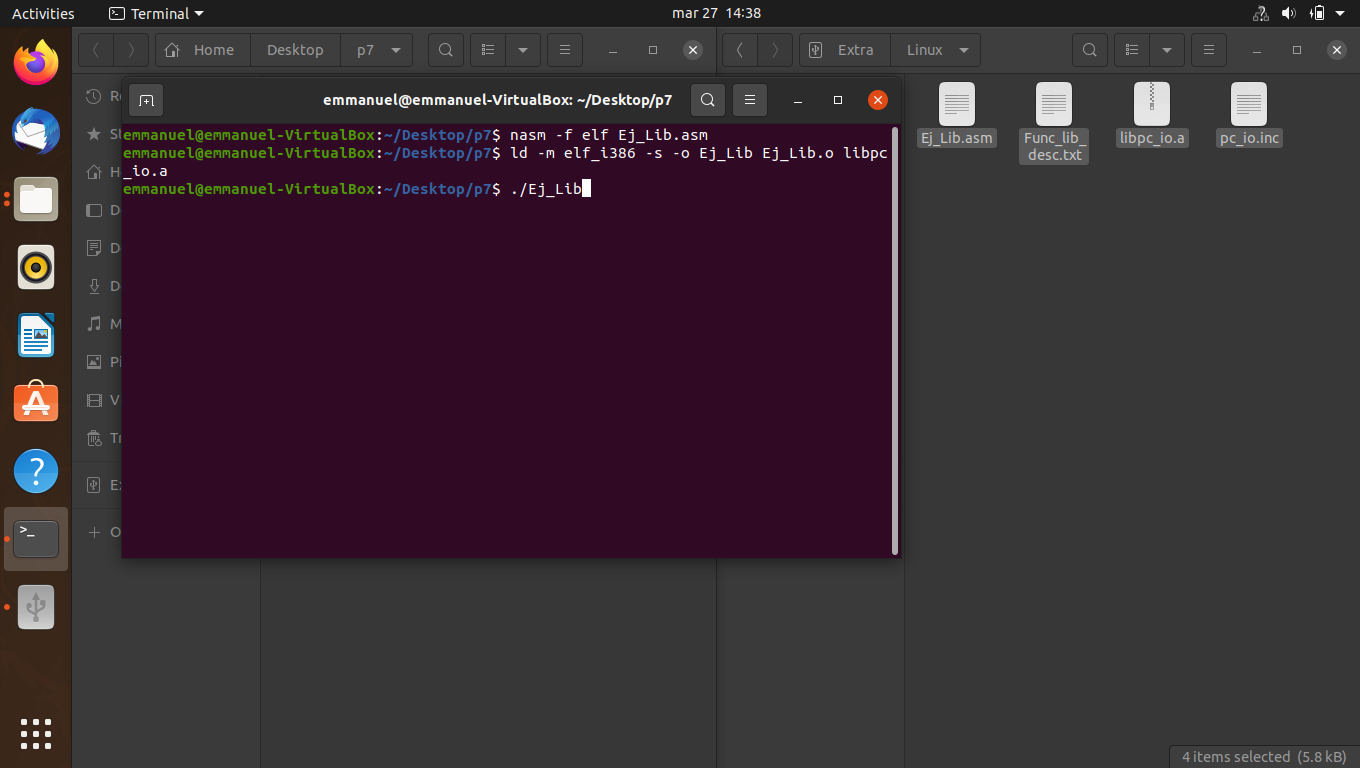
b) En un sistema operativo de 64 bits:

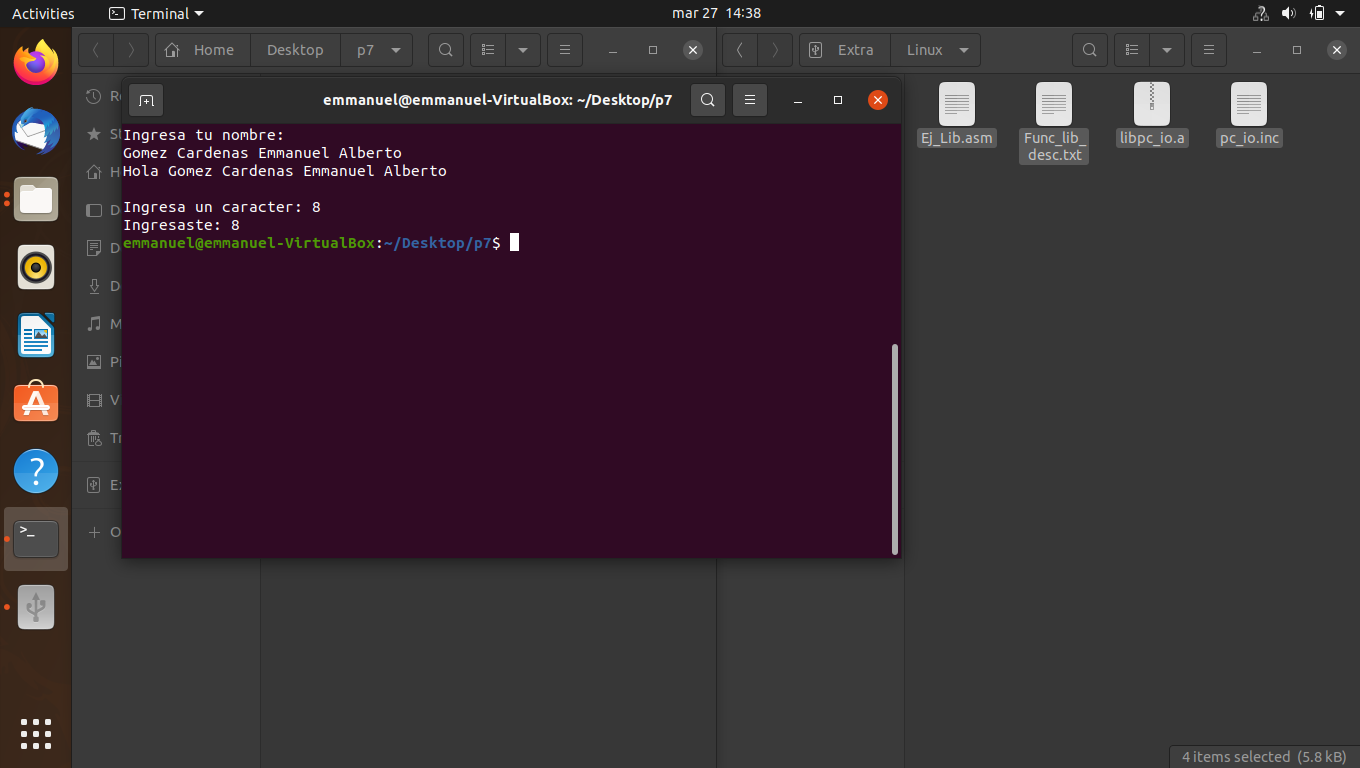
ld -m elf\_i386 -s -o Ej\_Lib Ej\_Lib.o libpc\_io.a

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.



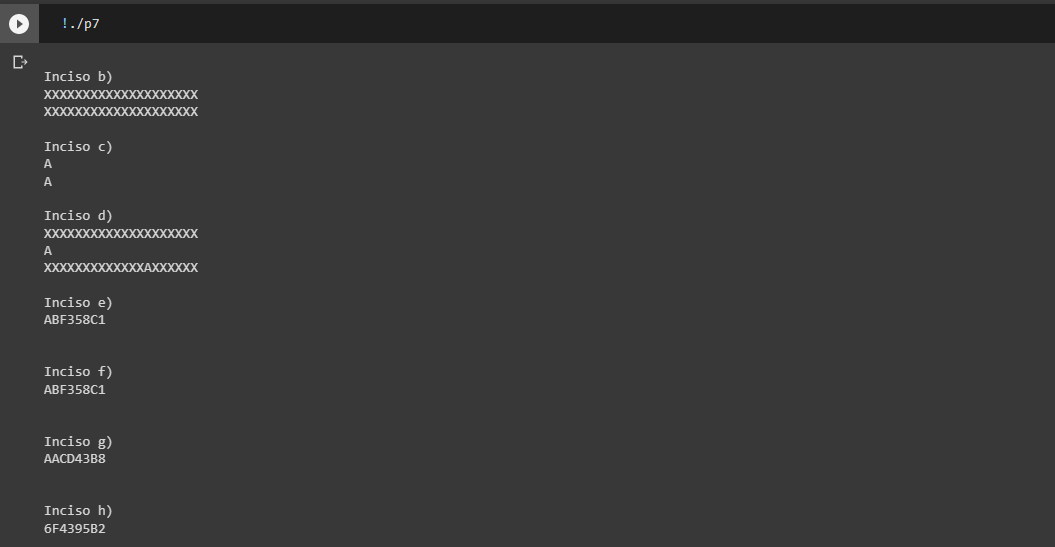


1. Cree un programa llamado **P7.asm** que contenga la rutina *printHex* de la Práctica 5, la cual recibe en EAX un valor que se quiere imprimir en formato hexadecimal. Agregue a **P7.asm** las instrucciones necesarias para hacer lo que se indica a continuación:
   1. Reservar tres espacios en memoria 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.
   2. Solicitar una cadena que se almacene en A.
   3. Solicitar un carácter y almacenarlo en C.
   4. Reemplazar el catorceavo carácter de A por el carácter en C. Use un modo de direccionamiento base con índice escalado más desplazamiento.
   5. Almacenar 0xABF358C1 en N.
   6. Insertar N en la pila.
   7. Reste a N su matrícula (use el valor como si fuera hexadecimal).
   8. Almacenar 0x6F4395B2 en ECX.
   9. Intercambiar ECX y el valor de 32 bits en el tope de la pila.
   10. Aumentar 1 a ECX sin ADD o ADC.

1. Realizar lo que se indica a continuación:
   1. Reservar en memoria no inicializada, las siguientes variables ‘a’, ‘b’, ‘c’, ‘d’, ‘e’ y ‘f’ cada una de 4 bytes.
   2. Almacenar su matrícula como hexadecimal en a.
   3. Realizar lo siguiente (a+1) sin ADD o ADC y almacenarlo en a.
   4. Almacenar N en b.
   5. Realizar lo siguiente (b-1) sin SUB o SBB y almacenarlo en b.
   6. Almacenar en c el diecisieteavo carácter de A.
   7. Almacenar en eax 0xA0B0000.
   8. Almacenar en d los primeros 16 bits más significativos de eax en la parte menos significativa de d.
   9. Almacenar en e un 0x20.
   10. Almacenar en f un 0x10.
   11. Realizar lo siguiente (a+b) y almacenarlo en eax.
   12. Pasar los primeros 16 bits más significativos a la parte menos significativa de eax (antes XXXX0000, después 0000XXXX), poner en cero los demás.
   13. Multiplicar eax con c.
   14. Reducir eax a únicamente 16 bits con corrimientos a la derecha.
   15. Restar eax con d.
   16. Multiplicar eax con e.
   17. Dividir el resultado del inciso z) con f.
   18. Mostrar únicamente el resultado del inciso aa) con printHex.

Por cada inciso, despliegue en pantalla el nuevo valor de la variable o registro modificado. Haga uso de la rutina *printHex* 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**

****

**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

.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

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.