



Universidad Autónoma de Baja California

FACULTAD DE CIENCIAS QUÍMICAS E INGENIERÍA



Organización y Arquitectura de las Computadoras.

Práctica No. 2: Introducción a ensamblador.

Alumno:

Joshua Osorio O. – 1293271

Docente:

José Isabel García R.

13/02/2026

Tabla de contenido

Objetivo	2
Descripción de la practica.....	2
Lista de materiales	2
Explicación paso a paso.....	6
Conclusiones.....	15
Dificultades	16

Objetivo

Distinguir las características de la organización y arquitectura del microprocesador de una computadora de propósito general, analizando sus recursos de hardware y software, para conocer capacidades y limitaciones de forma organizada y responsable.

Descripción de la practica

Lista de materiales

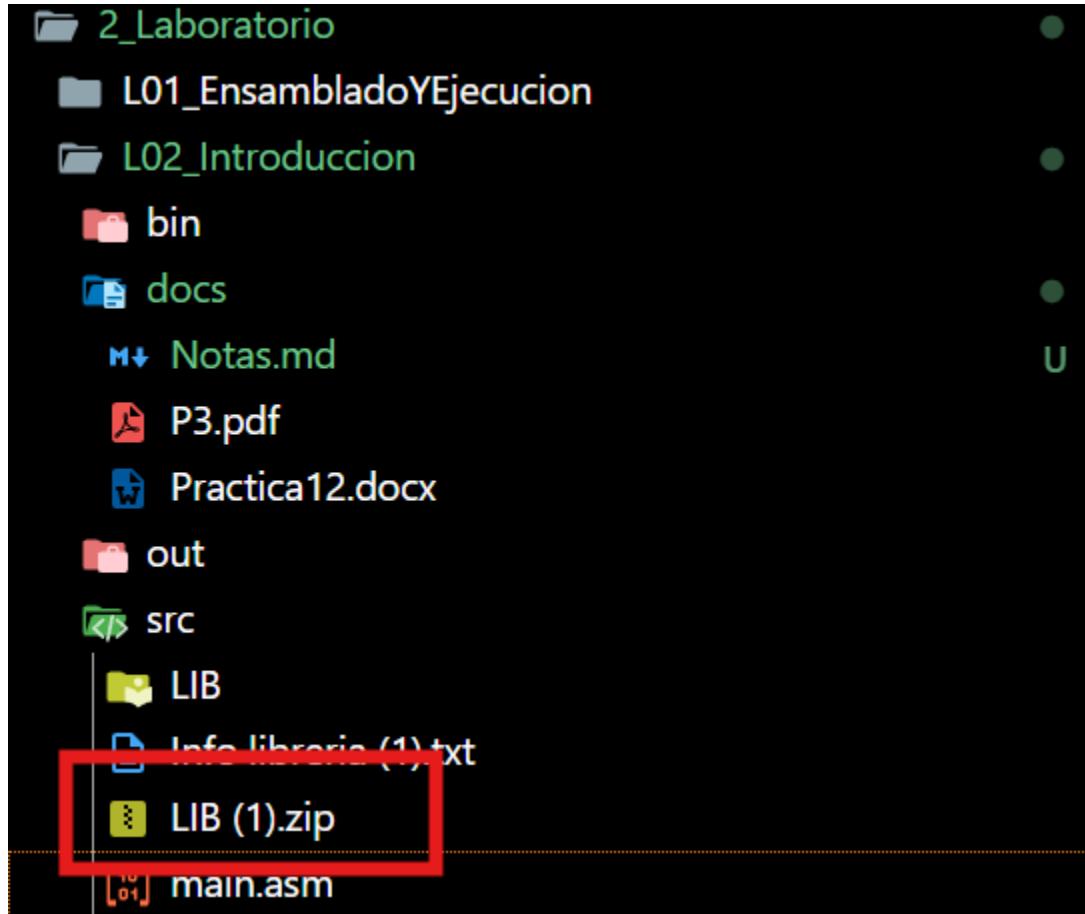
- Computadora
- Conexión a internet
- Git
- Vscode

Se cargó en GitHub el código de la práctica anterior y se verificó su correcto funcionamiento.

UABC_FCQI_ORGANIZACION_Y_ARQUITECTURA_DE_COMPUTADORAS.

The screenshot shows the Visual Studio Code interface. On the left, the code editor displays assembly code for `main.asm`. The code includes sections for data and bss, and defines global symbols `_start`, `msg1`, `msg2`, and `len`. The terminal window at the bottom shows the command-line process of assembling the code with `nasm -f elf32 -lsrc/LIB/src/main.asm -o out/main.o`, linking it with `ld -m elf_i386 -s -o bin/main out/main.o src/LIB/libpc_io.a`, and running the resulting executable `./bin/main`. The output of the program, "Hola Joshua", is also shown. The right side of the interface features the Explorer sidebar, which lists various files and folders related to the project, such as `Notas.md`, `P3.pdf`, and `Practica12.docx`.

Se descargó el archivo `lib.zip` desde Moodle, se descomprimió y se colocaron los dos archivos contenidos en la carpeta de trabajo de GitHub.



En la parte superior del archivo nombre_Practica.asm se agregó la instrucción:

```
%include "./pc_io.inc"
```

UABC_FCQI_ORGANIZACION_Y_ARQUITECTURA_DE_COMPUTADORAS.

```
1 %include "./pc_io.inc"
2
3 section .data ;Datos inicializados
4     msg1: db "Ingresa tu nombre",10,0
5     msg2: db "Hola ",0
6     msj: db 'Ingrese un digito (0-9)',0xA
7     len: equ $-msj
8
9 section .bss ;Datos no inicializados
10    num1 resb 1
11    nombre resb 256
12
13 section .text
14     global _start:
15
16 _start:
17     mov edx, msg1 ;Imprimir mensaje 1
18     call puts
19
```

El código se ensambló con el comando:

nasm -f elf nombre Practica.asm

Se enlazó el objeto generado con el comando:

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

```
PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS bash - L02_Introduccion + ⌂ ⌂ ... | [ ] X
@joeOsorio → /workspaces/36285_OrganizacionYArquitecturaDeComputadoras/2_Laboratorio/L02_Introduccion (main)
● $ nasm -f elf32 -Isrc/LIB src/main.asm -o out/main.o
@joeOsorio → /workspaces/36285_OrganizacionYArquitecturaDeComputadoras/2_Laboratorio/L02_Introduccion (main)
● $ ld -m elf_i386 -s -o bin/main out/main.o src/LIB/libpc_io.a
@joeOsorio → /workspaces/36285_OrganizacionYArquitecturaDeComputadoras/2_Laboratorio/L02_Introduccion (main)
● $ ./bin/main
Ingresa tu nombre
joshua
Hola joshua
@joeOsorio → /workspaces/36285_OrganizacionYArquitecturaDeComputadoras/2_Laboratorio/L02_Introduccion (main)
○ $
```

Finalmente, el programa se ejecutó con:

./nombre Practica

```

@joeOsorio → /workspaces/36285_OrganizacionYArquitecturaDeComputadoras/2_Laboratorio/L02_Introduccion (main)
● $ nasm -f elf32 -Isrc/LIB src/main.asm -o out/main.o
@joeOsorio → /workspaces/36285_OrganizacionYArquitecturaDeComputadoras/2_Laboratorio/L02_Introduccion (main)
● $ ld -m elf_i386 -s -o bin/main out/main.o src/LIB/libpc_io.a
@joeOsorio → /workspaces/36285_OrganizacionYArquitecturaDeComputadoras/2_Laboratorio/L02_Introduccion (main)
● $ ./bin/main
Ingresa tu nombre
joshua
Hola joshua
@joeOsorio → /workspaces/36285_OrganizacionYArquitecturaDeComputadoras/2_Laboratorio/L02_Introduccion (main)
○ $ [redacted]

> Codespaces: OAC ⑧ main* ⌂ ⑧ 0 △ 0 ⑧ 0
Lín. 18, col. 14 Tamaño de tabulación: 4 UTF-8 LF { } x86 and x86_64 Assem

```

Explicación paso a paso

En el código se deben agregar las siguientes partes:

1. En él .data agregar una constante de texto:

```

msj:    db  'Ingrese un digito (0-9)',0x0A
len:    equ $-msj

```

Esto define una cadena de texto con salto de línea (0x0A) y una constante len que guarda la longitud del mensaje.

2. En él .bss agregar una variable:

```

num1    resb 1

```

Aquí se reserva un byte para almacenar el valor ingresado.

3. En él .text se debe colocar el siguiente código:

Código1:

```

section .text
    mov eax, 4
    mov ebx, 1
    mov ecx, msj
    mov edx, len
    int 80h
    global _start:

```

¿Qué piensa usted que hará este código?

Carga el registro eax con un 4 decimal.

Carga el registro ebx con 1 decimal.

Carga el registro ecx con la dirección de la variable msj

Carga el registro edx con la dirección de la variable len.

Finaliza programa.

¿Qué hizo realmente este código?

UABC_FCQI_ORGANIZACION_Y_ARQUITECTURA_DE_COMPUTADORAS.

Coloque el código después del .text pero no cambio nada.

The screenshot shows the VS Code interface with the following details:

- Editors:** main.asm (highlighted), Notas.md, Notas.md
- Terminal Output:**

```
@joeosorio → /workspaces/36285_OrganizacionYArquitecturaDeComputadoras/2_Laboratorio/L02_Introduccion (main)
$ ld -m elf_i386 -s -o bin/main.out main.o src/LIB/libpc_io.a
@joeosorio → /workspaces/36285_OrganizacionYArquitecturaDeComputadoras/2_Laboratorio/L02_Introduccion (main)
$ nasm -f elf32 -lsrc/LIB src/main.asm -o out/main.o
@joeosorio → /workspaces/36285_OrganizacionYArquitecturaDeComputadoras/2_Laboratorio/L02_Introduccion (main)
$ ./bin/main
Ingrresa tu nombre
Joshua
Hola Joshua
@joeosorio → /workspaces/36285_OrganizacionYArquitecturaDeComputadoras/2_Laboratorio/L02_Introduccion (main)
$ []
```
- File Explorer:** Shows the project structure for L02_Introducción, including subfolders like L01_EstablecimientoYFuncionamiento, L02_Introducción, and L03_Tema.

Coloque el código después de _start: y aparece el mensaje.

Así que lo que realizo el Código fue:

Handwritten assembly code notes:

```
mov eax, 4      ;Carga instrucción del sistema.
mov ebx, 1      ;Indica que muestre algo en pantalla.
mov ecx, msj    ;Carga la dirección de mensaje.
mov edx, len    ;Carga la longitud del mensaje.
int 80h         ;Ejecuta La instrucción.
```

The screenshot shows the VS Code interface with the following details:

- Editors:** main.asm (highlighted), Notas.md, Notas.md
- Terminal Output:**

```
@joeosorio → /workspaces/36285_OrganizacionYArquitecturaDeComputadoras/2_Laboratorio/L02_Introduccion (main)
$ ld -m elf_i386 -s -o bin/main.out main.o src/LIB/libpc_io.a
@joeosorio → /workspaces/36285_OrganizacionYArquitecturaDeComputadoras/2_Laboratorio/L02_Introduccion (main)
$ nasm -f elf32 -lsrc/LIB src/main.asm -o out/main.o
@joeosorio → /workspaces/36285_OrganizacionYArquitecturaDeComputadoras/2_Laboratorio/L02_Introduccion (main)
$ ./bin/main
Ingrrese un digito (0-9)
Ingrresa tu nombre
Joshua
Hola Joshua
@joeosorio → /workspaces/36285_OrganizacionYArquitecturaDeComputadoras/2_Laboratorio/L02_Introduccion (main)
```
- File Explorer:** Shows the project structure for L02_Introducción, including subfolders like L01_EstablecimientoYFuncionamiento, L02_Introducción, and L03_Tema.

Código2:

UABC_FCQI_ORGANIZACION_Y_ARQUITECTURA_DE_COMPUTADORAS.

```
mov eax, 3
mov ebx, 0
mov ecx, num1
mov edx, 1
int 80h
```

¿Qué piensa usted que hará este código?

```
mov eax, 3      -> Carga una instrucción al sistema.
mov ebx, 0      -> Carga la instrucción de captura.
mov ecx, num1   -> Carga la dirección donde se almacenara.
mov edx, 1      -> Indica la longitud de la variable.
int 80h         -> Fin de la instrucción.
```

¿Qué hizo realmente este código?

```
mov eax, 3      ;carga instrucion al sistema.
mov ebx, 0      ;Instruccion captura.
mov ecx, num1   ;Direccion de variable.
mov edx, 1      ;Longitud de variable.
int 80h         ;Ejecuta instrucion.
```

The screenshot shows a code editor interface with a terminal window. The terminal window displays the assembly code and its execution:

```
2_Laboratorio > L02_Introducción > src > main.asm
15
16 start:
17     ;Codigo 1.
18     ; mov eax, 4      ;Carga instrucion del sistema.
19     ; mov ebx, 1      ;Indica que muestre algo en pantalla.
20     ; mov ecx, msg1  ;Carga la direccion de mensaje.
21     ; mov edx, len    ;Carga la longitud del mensaje.
22     ; int 80h        ;Finaliza instrucion.
23
24 ;codigo 2.
25 mov eax, 3      ;carga instrucion al sistema.
26 mov ebx, 0      ;Instruccion captura.
27 mov ecx, num1   ;Direccion de variable.
28 mov edx, 1      ;Longitud de variable.
29 int 80h         ;Ejecuta instrucion.
30
31 ;Inicio de codigo.
32 mov edx, msg1  ;Imprimir mensaje 1
33 call puts
```

TERMINAL OUTPUT:

```
@joeOsorio ~/workspaces/36285_OrganizacionYArquitecturaDeComputadoras/2_Laboratorio/L02_Introducción (main) $ nasm -f elf32 -Isrc/LIB src/main.asm -o out/main.o
@joeOsorio ~/workspaces/36285_OrganizacionYArquitecturaDeComputadoras/2_Laboratorio/L02_Introducción (main) $ ld -m elf_i386 -s -o bin/main out/main.o src/LIB/libpc_io.a
@joeOsorio ~/workspaces/36285_OrganizacionYArquitecturaDeComputadoras/2_Laboratorio/L02_Introducción (main) $ ./bin/main
1
Ingrresa tu nombre
Hola
@joeOsorio ~/workspaces/36285_OrganizacionYArquitecturaDeComputadoras/2_Laboratorio/L02_Introducción (main) $
```

The code editor shows the assembly file 'main.asm' with comments explaining each instruction. The terminal window shows the assembly code being compiled into an ELF32 executable, and then running it to print 'Hello' to the console.

UABC_FCQI_ORGANIZACION_Y_ARQUITECTURA_DE_COMPUTADORAS.

The screenshot shows the Visual Studio Code interface with the following details:

- Editor Area:** Displays the assembly code for `main.asm`. The code includes instructions for printing a message and reading input.
- Terminal:** Shows the command-line interface output for the assembly code execution.
- Explorer:** Shows the file tree for the workspace, including files like `src/main.asm`, `LIB/libpc.io.a`, and `Info libreria (1).txt`.

El compañero de la peña me ayudo a resolver la incógnita de porque se comporta raro el código, me cuestiono.

Código 3:

```
mov eax, 4  
mov ebx, 1  
mov ecx, num1  
mov edx, 1  
int 80h
```

¿Qué piensa usted que hará este código?

```
mov eax, 4      ;Carga instruccion para el sistema  
mov ebx, 1      ;Carga la instruccion de mostrar en pantalla.  
mov ecx, num1    ;Direccion de variable a imprimir  
mov edx, 1      ;Logitud de variable.  
int 80h         ;Ejecutar instrucción
```

Mostrar en pantalla lo capturado en la variable num1.

¿Qué hizo realmente este código?

UABC_FCQI_ORGANIZACION_Y_ARQUITECTURA_DE_COMPUTADORAS.

```
main.asm M | Q Extensión: Hex Editor
2_Laboratorio > L02_Introducción > src > main.asm
17 section .text
18
19     ; Código 1.
20     mov eax, 4      ;Carga instrucción del sistema.
21     mov ebx, 1      ;Indica que muestre algo en pantalla.
22     mov ecx, msg1  ;Carga la dirección de mensaje.
23     mov edx, len    ;Carga la longitud del mensaje.
24     int 80h          ;Finaliza instrucción.
25
26     ;Código 2.
27     mov eax, 3      ;carga instrucción al sistema.
28     mov ebx, 0      ;Instrucción captura.
29     mov ecx, num1  ;Dirección de variable.
30     mov edx, 2      ;Longitud de variable.
31     int 80h          ;Ejecuta instrucción.
32
33     ;Código 3.
34     mov eax, 4      ;Carga instrucción para el sistema
35     mov ebx, 1      ;Carga la instrucción de mostrar en pantalla.
36     mov ecx, num1  ;Dirección de variable a imprimir
37     mov edx, 1      ;Longitud de variable.
38     int 80h          ;Ejecutar instrucción
39
40     ; mov edx, msg1
41     ; call puts
42
43     ;Inicio de código base.
44     mov edx, msg1  ;Imprimir mensaje 1
45
46
PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS
@joseororio ~ /workspaces/36285_OrganizacionYArquitecturaDeComputadoras/2_Laboratorio/L02_Introducción (main) $ ./bin/main
Ingresé un digito (0-9)
Ingresé tu nombre
José
Hola Jioshua
@joseororio ~ /workspaces/36285_OrganizacionYArquitecturaDeComputadoras/2_Laboratorio/L02_Introducción (main) $
```

Mostró en pantalla lo capturado en la variable num1. Sin hacer enter y retorno.

```
Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda ↵ → 36285_OrganizacionYArquitecturaDeComputadoras [Codespaces: OAC]
main.asm M | Q Extensión: Hex Editor
2_Laboratorio > L02_Introducción > src > main.asm
17 section .text
18
19     ; Código 1.
20     mov eax, 4      ;Carga instrucción del sistema.
21     mov ebx, 1      ;Indica que muestre algo en pantalla.
22     mov ecx, msg1  ;Carga la dirección de mensaje.
23     mov edx, len    ;Carga la longitud del mensaje.
24     int 80h          ;Finaliza instrucción.
25
26     ;Código 2.
27     mov eax, 3      ;carga instrucción al sistema.
28     mov ebx, 0      ;Instrucción captura.
29     mov ecx, num1  ;Dirección de variable.
30     mov edx, 2      ;Longitud de variable.
31     int 80h          ;Ejecuta instrucción.
32
33     ;Código 3.
34     mov eax, 4      ;Carga instrucción para el sistema
35     mov ebx, 1      ;Carga la instrucción de mostrar en pantalla.
36     mov ecx, num1  ;Dirección de variable a imprimir
37     mov edx, 1      ;Longitud de variable.
38     int 80h          ;Ejecutar instrucción
39
40     ; mov edx, msg1
41     ; call puts
42
43     ;Inicio de código base.
44     mov edx, msg1  ;Imprimir mensaje 1
45
46
PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS
@joseororio ~ /workspaces/36285_OrganizacionYArquitecturaDeComputadoras/2_Laboratorio/L02_Introducción (main) $ nasm -f elf32 -Isrc/LIB src/main.asm -o main.o
@joseororio ~ /workspaces/36285_OrganizacionYArquitecturaDeComputadoras/2_Laboratorio/L02_Introducción (main) $ ld -m elf_i386 -s -o bin/main.out main.o src/LIB/libpc_io.a
@joseororio ~ /workspaces/36285_OrganizacionYArquitecturaDeComputadoras/2_Laboratorio/L02_Introducción (main) $ ./bin/main
Ingresé un digito (0-9)
Ingresé tu nombre
Jioshua
Hola Jioshua
@joseororio ~ /workspaces/36285_OrganizacionYArquitecturaDeComputadoras/2_Laboratorio/L02_Introducción (main) $
```

4. Del paso anterior se debe explicar textualmente que se piensa que hace cada sección de código, con sus propias palabras.

Código 1: Muestra en pantalla el contenido de la variable msg1 y da un salto de línea.

Código 2: Captura y guardar valor en variable 1

Código 3: Muestra el contenido de la variable num1

5. Ahora el código anterior lo vamos a modificar:

- a. El msg1 ahora debe quedar de la siguiente manera:

```
msg1: db 'Ingrrese un digito (0-9)',0x0
```

- b. El código del paso 3 se debe comentar y agregar el siguiente:

UABC_FCQI_ORGANIZACION_Y_ARQUITECTURA_DE_COMPUTADORAS.

¿Qué piensa usted que hará este código?

```
    mov edx, msj  
    call puts
```

Mueve la dirección al registro edx y luego la subrutina o función se encarga de mostrarlo en terminal.

¿Qué hizo realmente este código?

Mostro el mensaje en terminal, cargo la dirección de la variable msj en el registro edx y después con la función puts lo mostro en terminal.

¿Qué piensa usted que hará este código?

call getch()

Capturar algún carácter.

¿Qué hizo realmente este código?

Esta mostrando el ultimo contenido de al

En mi caso está guardando un 0.

¿Qué piensa usted que hará este código?

call putchar

¿Qué hizo realmente este código?

Muestra el contenido capturado por el teclado

UABC_FCQI_ORGANIZACION_Y_ARQUITECTURA_DE_COMPUTADORAS.

The screenshot shows a code editor interface with multiple tabs. The active tab is 'main.asm M'. The code in 'main.asm' contains assembly instructions for reading a character from the keyboard and printing it to the screen. The terminal window on the right shows the assembly code being assembled with nasm -f elf32 -Isrc, linking with libpc.io.a, and then running the resulting executable. The output shows the program asking for input, receiving '12', and then printing 'Hola'.

```
40 ; mov edx, nombre ;Dirección de variable a imprimir
41 ; mov edx, 1 ;Longitud de variable.
42 ; int 80h ;Ejecutar instrucción
43
44 ;Parte 5.b práctica
45 mov edx, msj
46 call puts
47
48 call getch
49 call putchar
50
51 mov ebx, nombre
52 ;capturar:
53 call getch
54 mov byte [ebx], al
55 inc ebx
56
57 cmp al, 10
58 jne capturar
59 mov byte [ebx], 0
60
61 mov edx, msg2
62 call puts
63
64 mov edx, nombre
65 call puts
66 call putchar
67
68 mov eax, 1 ;Carga la instrucción de salida de programa.
69 mov ebx, 0 ;Indica que terminó correctamente, como un return 0 en c.
70 int 80h ;Llamada a kerner con las anteriores mensajes.
71
72 mov edx, nombre
73 call puts
74
75 mov eax, 1 ;Carga la instrucción de salida de programa.
76 mov ebx, 0 ;Indica que terminó correctamente, como un return 0 en c.
77 int 80h ;Llamada a kerner con las anteriores mensajes.
```

TERMINAL

```
@joeOsorio → /workspaces/36285_OrganizacionYArquitecturaDeComputadoras/2_Laboratorio/L02_Introduccion (main) $ ./bin/main
@joeOsorio → /workspaces/36285_OrganizacionYArquitecturaDeComputadoras/2_Laboratorio/L02_Introduccion (main) $ ld -m elf_i386 -s -o bin/main.out/main.o src/LIB/libpc.io.a
@joeOsorio → /workspaces/36285_OrganizacionYArquitecturaDeComputadoras/2_Laboratorio/L02_Introduccion (main) $ ./bin/main
Ingrese un dígito (0-9)
12
Hola

@joeOsorio → /workspaces/36285_OrganizacionYArquitecturaDeComputadoras/2_Laboratorio/L02_Introduccion (main) $ ]
```

6. En la parte inferior del código, vamos a agregar la siguiente función:

salto:

```
mov al, 13
call putchar
mov al, 10
call putchar
ret
```

7. Ahora el código se modifica con salto y debe quedar de la siguiente manera:

¿Qué piensa usted que hará todo este código?

```
mov edx, msj
call puts
call salto
call getch
call salto
call putchar
call salto
```

Se carga la dirección en el registro edx.

Se muestra el mensaje con la rutina puts.

Se genera un salto de línea.

Se captura un carácter.

Se genera un salto de línea.

Se muestra lo capturado.

Se genera un salto de línea.

¿Qué hizo realmente todo este código?

Realizo lo pensado y mencionado anteriormente.

UABC_FCQI_ORGANIZACION_Y_ARQUITECTURA_DE_COMPUTADORAS.

The screenshot shows a terminal window with the following content:

```
> EDITORES ABIERTOS
  > 36285_ORGANIZACIONYARQUITECTURADECOMPUTADORAS [CODESPACES: OAC]
    2_Laboratorio
      > L02_Introduccion
        src
          LIB (1).zip
          main.asm
          taller.asm
          L03_Tema
          L04_Tema
          L05_Tema
          L06_Tema
        TERMINAL
          @joeOsorio → /workspaces/36285_OrganizacionYArquitecturaDeComputadoras/2_Laboratorio/L02_Introduccion (main) $ ./bin/main
            1
            Hola 1
          @joeOsorio → /workspaces/36285_OrganizacionYArquitecturaDeComputadoras/2_Laboratorio/L02_Introduccion (main) $ ./bin/main
            Ingrese un digito (0-9)

          2
          Hola 2
          @joeOsorio → /workspaces/36285_OrganizacionYArquitecturaDeComputadoras/2_Laboratorio/L02_Introduccion (main) $ []
          > ESQUEMA
          > LÍNEA DE TIEMPO
```

The assembly code in main.asm is:

```
55 ; call putchar
56
57
58 ; Parte 7
59 mov edx, msg1
60 call puts
61 call salto
62 call getch
63 call salto
64 call putchar
65 call salto
66
67
68 mov ebx, nombre
69 ; _____
70 capturar:
71     call getch
72     mov byte [ebx], al
73     inc ebx
74
75 cmp al, 10
76 jne capturar
77 mov byte [ebx], 0
78
79 mov edx, msg2
80 call puts
81
82 mov edx, nombre
83 call puts
84 call putchar
85 ;
```

8. Vamos y modificamos la función de salto:

```
salto:
  pushad
  mov al, 13
  call putchar
  mov al, 10
  call putchar
  popad
  ret
```

Agrega las banderas pusad y popad que indica donde inicia y donde termina la función.

¿Qué hizo realmente todo este código?

Agrego un salto después de mostrar el mensaje.

Captura un carácter.

Muestra el carácter.

Genera un salto y captura.

9. Ahora agregamos la funcionalidad de capturar una letra, por ejemplo la a y sumarle un 1 e imprimirla en terminal.

Agregamos el siguiente código después del último call salto

¿Qué piensa usted que hará todo este código?

```
mov ebx, num1
  mov byte[ebx], al
  add byte[ebx], 1
  mov al, byte[ebx]
  call putchar
```

Copia la dirección de num1 a ebx.

UABC_FCQI_ORGANIZACION_Y_ARQUITECTURA_DE_COMPUTADORAS.

Copia el contenido de 'al' al contenido de ebx casteando un byte.

Suma 1 al contenido de ebx casteado a byte.

Muestra el contenido.

¿Qué hizo realmente todo este código?

Realizo lo mencionado anteriormente sin embargo cabe destacar que putchar muestra el contenido de 'al'

The screenshot shows a terminal window with the following content:

```
36285_OrganizacionYArquitecturaDeComputadoras [CODESPACES: OAC]
2_Laboratorio > L02_Introduccion > src > main.asm
[main.asm] [Info libreria (1).txt] [Clase.asm] [Explorador]

EDITORIOS ABIERTOS
36285_OrganizacionYArquitecturaDeComputadoras [CODESPACES: OAC]
2_Laboratorio
L02_Introduccion
src
LIB (1).zip
main.asm
Taller.asm
L03_Tema
L04_Tema
L05_Tema
L06_Tema
TERMINAL
@joeOsorio ~ /workspaces/36285_OrganizacionYArquitecturaDeComputadoras/doras/_2_Laboratorio/L02_Introduccion (main) $ nasm -f elf32 -isrc /LIB/src/main.asm
@joeOsorio ~ /workspaces/36285_OrganizacionYArquitecturaDeComputadoras/doras/_2_Laboratorio/L02_Introduccion (main) $ ld -m elf_i386 -s -l o bin/main out/main.o src/LIB/libpc.io.a
@joeOsorio ~ /workspaces/36285_OrganizacionYArquitecturaDeComputadoras/doras/_2_Laboratorio/L02_Introduccion (main) $ ./bin/main
Ingrese un digito (0-9)

a
b
Hola

ESQUEMA
LINEA DE TIEMPO
```

10. Ahora para la prueba capture 3 y súmelo 1 como el caso anterior.

The screenshot shows a terminal window with the following content:

```
36285_OrganizacionYArquitecturaDeComputadoras [CODESPACES: OAC]
2_Laboratorio > L02_Introduccion > src > main.asm
[main.asm] [Info libreria (1).txt] [Clase.asm] [Explorador]

EDITORIOS ABIERTOS
36285_OrganizacionYArquitecturaDeComputadoras [CODESPACES: OAC]
2_Laboratorio
L02_Introduccion
src
LIB (1).zip
main.asm
Taller.asm
L03_Tema
L04_Tema
L05_Tema
L06_Tema
TERMINAL
@joeOsorio ~ /workspaces/36285_OrganizacionYArquitecturaDeComputadoras/doras/_2_Laboratorio/L02_Introduccion (main) $ ./bin/main
Ingrese un digito (0-9)

3
4
Hola

ESQUEMA
LINEA DE TIEMPO
```

11. Ahora capture otro número, modifique lo necesario para ello, y pruebe con una suma de $3 + 4 = 7$:

¿Qué piensa usted que hará todo este código?

```
mov ebx, num1
mov al, [ebx]
mov ebx, num2
add [ebx], al
mov al, byte[ebx]
call putchar
```

Copia la dirección de num1 al registro ebx.

UABC_FCFI_ORGANIZACION_Y_ARQUITECTURA_DE_COMPUTADORAS.

Copia el valor de ebx al registro ‘al’.

Copia la dirección de num2 a ebx.

Suma el contenido de 'al' al contenido de ebx.

Copia el contenido casteado a byte del registro ebx al registro 'al'.

Muestra el contenido en terminal del registro 'al'.

¿Qué hizo realmente todo este código?

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure under "36285_OrganizacionYArquitecturaDeComputadoras [Codespaces: OAC]". The "src" folder contains "LIB (1).zip", "main.asm", and "Taller.asm". Other folders like "L03_Tema", "L04_Tema", "L05_Tema", and "L06_Tema" are also listed.
- Terminal:** Displays the command-line interface with the following session:

```
@joeOsorio → /workspaces/36285_OrganizacionYArquitecturaDeComputadoras/doras/_Laboratorio/L02_Introduccion (main) $ ./bin/main
Ingrese un digito (0-9)

3
44
Hola

@joeOsorio → /workspaces/36285_OrganizacionYArquitecturaDeComputadoras/doras/_Laboratorio/L02_Introduccion (main) $
```
- Code Editor:** The main editor pane displays assembly code for "main.asm". The code includes various instructions like mov, add, call, and conditional jumps, along with comments explaining parts 9 and 11.
- Status Bar:** Shows the current workspace ("Taller.asm U"), file name ("main.asm M"), and other status information.

Muestra en terminal el contenido de msg1.

Se realizan 2 saltos.

Se captura y muestra el carácter.

Muestra el contenido 'al' con el 1 sumado que da 4 en ASCII

Muestra en terminal el contenido de msq1 y muestra lo capturado en nombre.

Obligatoriamente de cada paso debe tener las capturas del código y de la terminal.

Las preguntas van en relación con terminal, solo eso.

Conclusiones

Esta práctica de laboratorio permitió integrar los fundamentos del lenguaje ensamblador, destacando la manipulación directa de recursos de hardware como los registros y la memoria. A través de la declaración de variables en las secciones .data y .bss , y el uso de subrutinas (call puts, call getche, call putchar), se dominaron las operaciones básicas de entrada/salida. Un aprendizaje clave fue la relación entre la manipulación numérica de datos y su representación como caracteres mediante el código ASCII, como se evidenció al sumar 1 a un

valor capturado. En resumen, se logró el objetivo de distinguir las características de la organización y arquitectura del microprocesador, sentando bases sólidas para el desarrollo de programas eficientes a bajo nivel.

Dificultades

Tuve dificultades para definir una organización adecuada de carpetas para el repositorio, así como para encapsular y estructurar de mejor manera las bibliotecas proporcionadas por el docente.

Otra dificultad consistió en adaptar el ensamblaje y el enlazado de los archivos .asm a la estructura de carpetas propuesta, lo que derivó finalmente en el uso de los siguientes comandos.

```
$ nasm -f elf32 -lsrc/LIB src/main.asm -o out/main.o  
$ ld -m elf_i386 -s -o bin/main out/main.o src/LIB/libpc_io.a  
$ ./bin/main
```