



Universidad Autónoma de Baja California

FACULTAD DE CIENCIAS QUÍMICAS E INGENIERÍA



Organización y Arquitectura de las Computadoras.

Práctica No. 1: Introducción a ensamblador.

Alumno:

Joshua Osorio O. – 1293271

Docente:

José Isabel García R.

04/02/2026

Tabla de contenido

Objetivo	2
Descripción de la practica.....	2
Lista de materiales	4
Desarrollo.....	4
Investigación previa.....	4
¿Qué son: .data, .bss, .txt y directiva global?	4
¿Para qué sirven?.....	4
¿Cómo se usan?.....	5
¿Qué elementos van en cada sección?	5
Comandos más populares en Linux y como se usan.	5
Que son las extensiones de archivo relacionadas con asm.?	6
Evidencia sobre el repositorio y código realizado en github.	6
Conclusiones.....	8
Bibliografía	8

Objetivo

Conocer y dominar el uso de una máquina virtual con sistema operativo Linux analizando sus recursos de hardware y software, para conocer sus capacidades y limitaciones de forma organizada y responsable.

Descripción de la practica

Realizar una pequeña investigación en la arquitectura 386 de ensamblador sobre:

- .data
- .bss
- .txt
- directiva global

De cada una responder lo siguiente:

1. ¿Qué son?

UABC_FCQI_ORGANIZACION_Y_ARQUITECTURA_DE_COMPUTADORAS.

2. ¿Para qué sirven?
3. ¿Cómo se usan?
4. ¿Qué elementos van en cada sección?

Realizar una pequeña investigación sobre:

- Los comandos más populares en Linux y como se usan.
- Sobre que son las extensiones de archivo relacionadas con asm.

Realice los siguientes pasos:

1. Entrar a GitHub
2. Crear un repositorio nuevo con un archivo con nombre “Ejemplo_OAC.txt”.
3. Para activar codespaces se requiere dar clic en <> Code luego en Codespaces
4. Instalar nasm, con la siguiente instrucción:
sudo apt update
sudo apt install nasm -y
5. Revisar la versión de nasm instalado: nasm -v
6. Crear una carpeta cuyo nombre sea “ejemplo”: mkdir ejemplo
7. Cambiar del directorio /workspaces para que la carpeta nueva sea este, con:
cd /workspaces/"sustituir_con_nombre_repositorio"/ejemplo/
8. Comprobar que este cambio se haya realizado con: pwd.
9. Crear un archivo con extensión .asm y con nombre “ejemplo”: touch ejemplo.asm
10. Abrir Notepad para trabajar en ese editor de texto sin ayudas.
11. Ingresar el siguiente código en el archivo:

```
global _start
section .text
_start:
    ; sys_write(stdout, message, length)
    mov     eax,     4
    mov     ebx,     1
    mov     ecx,     message
    mov     edx,     length
    int     80h

    ; sys_exit(return_code)
    mov     eax,     1                ;sys_exit syscall
    mov     ebx,     0                ;return 0 (todo correcto)
    int     80h

section .data
message:    db      'Hello,    world!',    0x0A    ;mensaje y nueva linea
length:     equ     $-message                ;Obtenemos la longitud de la cadena
```

12. Cargar el código escrito en **Notepad** sin errores en el archivo ejemplo.asm en GitHub.
13. Ensamblar el archivo creado anteriormente: nasm -f elf ejemplo.asm

14. Enlazar el objeto creado: `ld -m elf_i386 -s -o ejemplo ejemplo.o`

15. Ejecutar el programa obtenido: `./ejemplo`

Lista de materiales

- Computadora
- Conexión a internet
- Git
- Vscod

Desarrollo

Investigación previa.

¿Qué son: .data, .bss, .txt y directiva global?

.text

Es la sección donde se almacena el código ejecutable del programa.

.data

Es la sección que contiene datos inicializados.

. bss

Es la sección que contiene variables no inicializadas.

Directiva global

Una directiva que hace visible un símbolo fuera del archivo.

¿Para qué sirven?

.text

Contiene las instrucciones máquina que la CPU ejecutará.

.data

Guarda variables globales o contantes cuyo valor inicial es conocido.

. bss

Reserva memoria son asignar valores; el sistema las inicializa en cero.

Directiva global

Permite que el enlazador reconozca el punto de entrada del programa.

¿Cómo se usan?

```
.text
    section .text

.data
    section .data
    mensaje: db "Hola mundo", 0

.bss
    section .bss
    buffer: resb 64

Directiva global
    global _start
```

¿Qué elementos van en cada sección?

```
.text
    Instrucciones del programa, Puntos de entrada como _start y llamadas al sistema

.data
    Cadenas, variables inicializadas y contantes.

.bss
    Buffers, arreglos y variables sin valor inicial.

Directiva global
    Nombres de funciones o etiquetas que deben ser accesibles externamente.
```

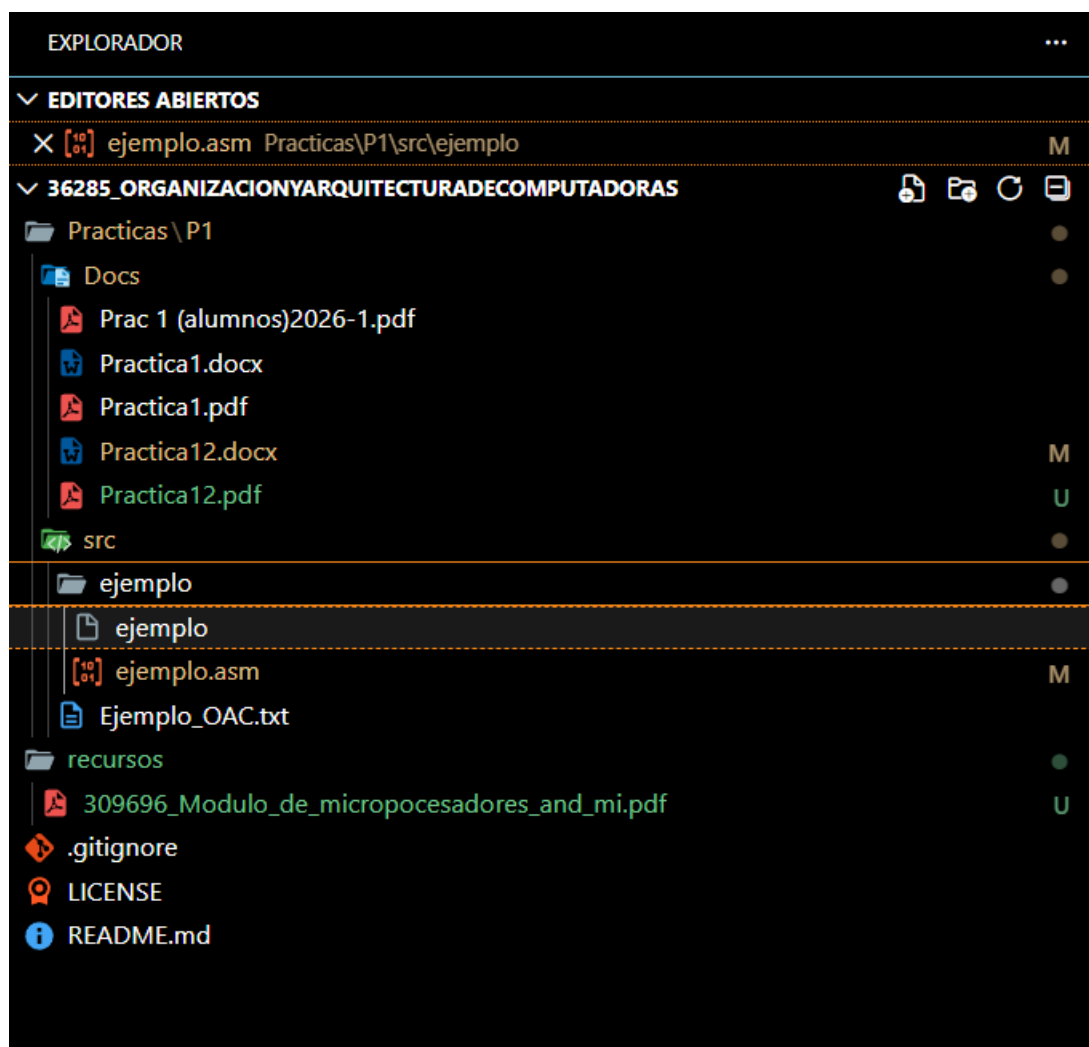
Comandos más populares en Linux y como se usan.

Comando	Función	Ejemplo
cd	Cambiar de directorio	cd carpeta
ls	Listar archivos	ls -l
pwd	Mostrar ruta actual	pwd
mkdir	Crear carpeta	mkdir ejemplo
touch	Crear archivo vacío	touch archivo.asm
rm	Eliminar archivos	rm archivo.asm
cp	Copiar archivos	cp archivo.asm b
mv	Mover / renombrar	mv archivo b
cat	Mostrar contenido	cat archivo.asm
sudo	Ejecutar como administrador	sudo apt update

Que son las extensiones de archivo relacionadas con asm.?

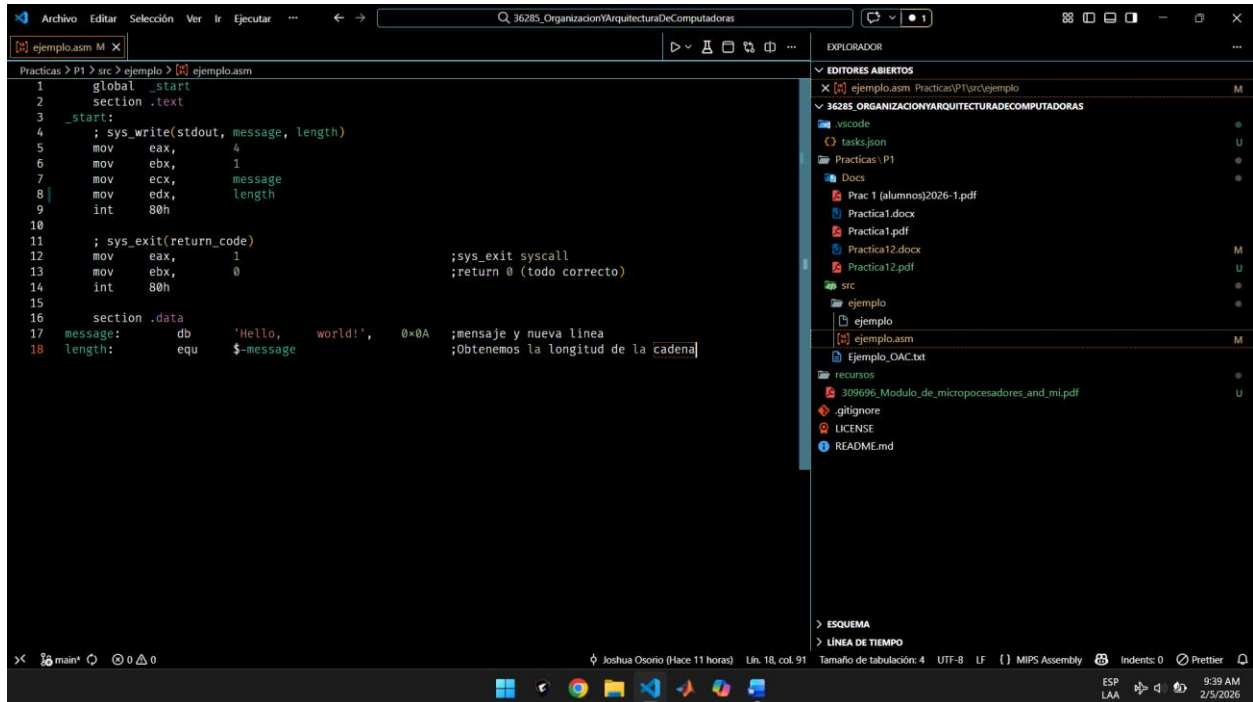
Extensión	Significado	Uso
.asm	Archivo de código ensamblador	Código fuente NASM
.s	Ensamblador AT&T	Usado por GCC
.o	Objeto compilado	Resultado de nasm -f elf
.bin	Binario puro	Código sin encabezados
.elf	Formato ejecutable Linux	Resultado de ld

Evidencia sobre el repositorio y código realizado en github.



Estructura básica del repositorio.

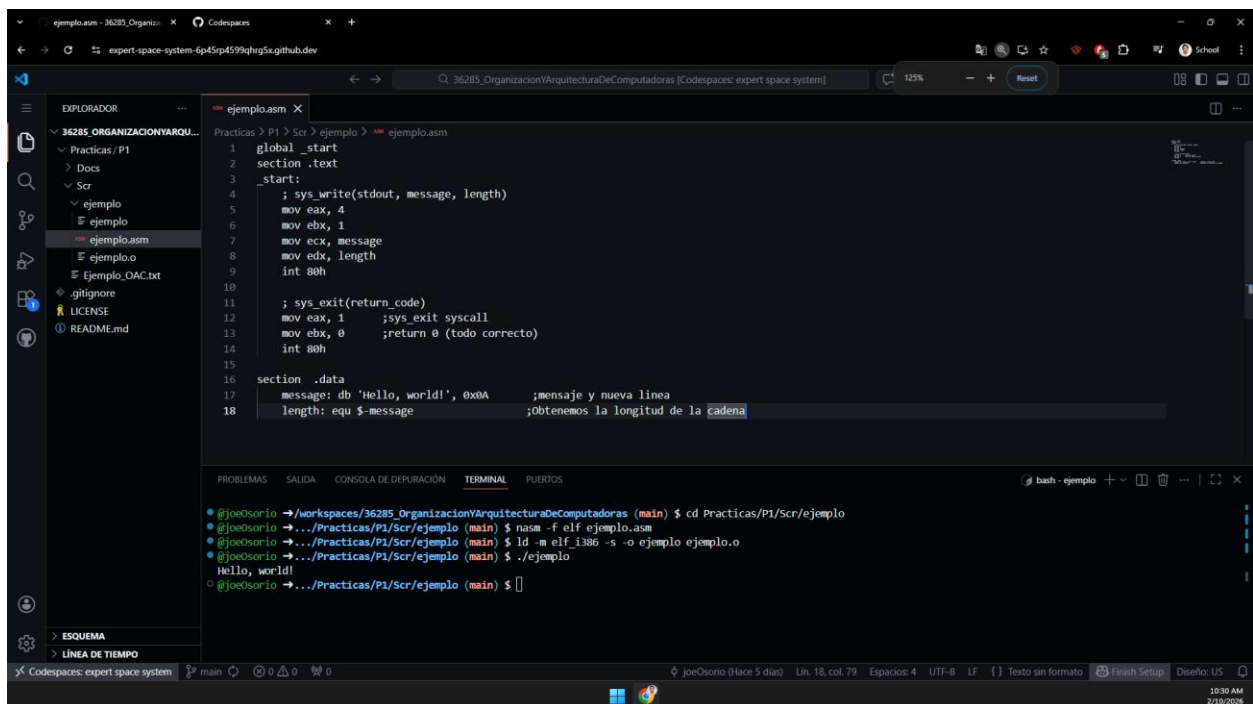
UABC_FCQI_ORGANIZACION_Y_ARQUITECTURA_DE_COMPUTADORAS.



The screenshot shows the Visual Studio Code editor with a file named `ejemplo.asm` open. The code is written in x86 assembly and includes comments in Spanish. The code defines a global `_start` function, sets up registers for `sys_write`, and prints the message "Hello, world!". It also includes a `_start` label and a `section .data` block for the message string.

```
1 global _start
2 section .text
3 _start:
4     ; sys_write(stdout, message, length)
5     mov eax, 4
6     mov ebx, 1
7     mov ecx, message
8     mov edx, length
9     int 80h
10
11     ; sys_exit(return_code)
12     mov eax, 1
13     mov ebx, 0
14     int 80h
15
16 section .data
17 message: db 'Hello, world!', 0x0A ;mensaje y nueva linea
18 length:  equ $-message           ;obtenemos la longitud de la cadena
```

Archivo .asm con código para imprimir Hola mundo.



The screenshot shows the Visual Studio Code editor with the same `ejemplo.asm` file open. The terminal window at the bottom shows the execution of the assembly program. The user has navigated to the directory `Practicas/P1/Scr/ejemplo` and run the following commands:

```
@joeOsorio → /workspaces/36285_OrganizacionYArquitecturaDeComputadoras (main) $ cd Practicas/P1/Scr/ejemplo
@joeOsorio → .../Practicas/P1/Scr/ejemplo (main) $ nasm -f elf ejemplo.asm
@joeOsorio → .../Practicas/P1/Scr/ejemplo (main) $ ld -m elf_i386 -s -o ejemplo ejemplo.o
@joeOsorio → .../Practicas/P1/Scr/ejemplo (main) $ ./ejemplo
Hello, world!
@joeOsorio → .../Practicas/P1/Scr/ejemplo (main) $ []
```

Ejecución de instrucciones para ensamblar y enlazar programa.

Conclusiones

En esta práctica nos introducimos al mundo de ensamblador, creando un repositorio en GitHub con una estructura simple, pero con la que trabajaremos a lo largo del curso. Además, con la investigación nos ayuda a familiarizarnos un poco más sobre la estructura básica del lenguaje ensamblador x86 realizando el primer hola mundo, relacionamos algunos comandos básicos.

Bibliografía

- [1] Roger Baig I Viñas, *Sistema operativo GNU/Linux básico*, 1.^a ed. 2003. Disponible en:
<https://redua.wordpress.com/wp-content/uploads/2008/04/868.pdf>