



# TECNOLÓGICO NACIONAL DE MÉXICO

## INSTITUTO TECNOLÓGICO DE TLAXIACO

---

### PRÁCTICA 2.- INTRODUCCIÓN A LA PROGRAMACIÓN EN ENSAMBLADOR

---

**Presenta:**

- Santiago Bautista Maribel 22620262

**Carrera:**

Ingeniería en Sistemas Computacionales

**Asignatura:**

Arquitectura de Computadoras

**Docente:**

Ing. José Alfredo Edward Osorio Salinas



Tlaxiaco, Oaxaca, a 8 de septiembre de 2024.

*"Educación, Ciencia y Tecnología, Progreso día con día"®*



## ÍNDICE

INTRODUCCIÓN .....	1
OBJETIVO.....	2
MATERIALES.....	2
DESARROLLO.....	2
Descripción: .....	2
4.1. Ejercicio 1. ....	2
4.1.1. Código fuente. ....	3
4.1.2. Captura de pantalla del programa en ejecución. ....	5
4.2. Ejercicio 2.....	5
4.2.1. Código fuente. ....	5
4.2.2. Captura de pantalla del programa en ejecución. ....	7
4.3 Ejercicio 3. ....	7
4.3.1. Código fuente. ....	8
4.3.2. Captura de pantalla del programa en ejecución. ....	10
4.4 Ejercicio 4 (EXTRA).....	10
4.4.1. Código fuente. ....	10
4.4.2. Captura de pantalla del programa en ejecución. ....	13
CONCLUSIÓN .....	14
REFERENCIAS BIBLIOGRÁFICAS .....	15

## INTRODUCCIÓN

En el presente trabajo se desarrolló una serie de códigos en lenguaje ensamblador a través de NASM y el bloc de notas. Al inicio se presenta el objetivo de esta práctica, posteriormente se presentan los materiales que utilizamos para llevarla a cabo, luego el desarrollo, en donde se da una breve descripción de cada código y las capturas de pantalla, por último, se encuentra una conclusión y las referencias bibliográficas.

El lenguaje ensamblador actúa como un intermediario entre el código máquina y los lenguajes de alto nivel, ofreciendo una comunicación más directa con el hardware. Cada familia de procesadores posee su propio lenguaje ensamblador, vinculado estrechamente a su arquitectura específica.

Un programa ensamblador convierte las instrucciones de lenguaje ensamblador a código máquina, facilitado por herramientas conocidas como ensambladores. Esta labor de traducción es necesaria para que se ejecuten componentes esenciales de un dispositivo, como los sistemas operativos y los controladores de dispositivo. Aunque el lenguaje ensamblador tiene una curva de aprendizaje acusada, proporciona una comprensión profunda de las operaciones a nivel de hardware.

El lenguaje ensamblador se suele emplear en los siguientes casos, manipulación directa del hardware, acceso a las instrucciones de los procesadores, resolución de situaciones críticas de rendimiento. (EDUCA EDTECH GROUP, 09)

## OBJETIVO

- ✓ El objetivo de esta práctica es que el alumno se familiarice con la programación en ensamblador, utilizando el lenguaje NASM. Para ello, se le pide que realice una serie de ejercicios que le permitan comprender los conceptos básicos de la programación en ensamblador, como la manipulación de registros, la utilización de instrucciones de salto condicional e incondicional, la utilización de instrucciones de entrada y salida, la utilización de instrucciones de manipulación de datos, entre otros.

## MATERIALES

- ✓ Computadora con el software NASM instalado.
- ✓ Editor de texto.

## DESARROLLO

### Descripción:

Para realizar esta práctica, el alumno deberá realizar los siguientes ejercicios:

- I. Realizar un programa en ensamblador que solicite al usuario dos números enteros y realice la suma de los mismos. El resultado de la suma deberá ser mostrado en pantalla.
- II. Realizar un programa en ensamblador que solicite al usuario un número entero y determine si el mismo es par o impar. El resultado deberá ser mostrado en pantalla.
- III. Realizar un programa en ensamblador que solicite al usuario un número entero y determine si el mismo es positivo, negativo o cero. El resultado deberá ser mostrado en pantalla.
- IV. **\*\*Extra +1:\*\*** Realizar un programa en ensamblador que solicite al usuario un número entero y determine si el mismo es primo o no. El resultado deberá ser mostrado en pantalla.

### Ejercicios:

#### 4.1. Ejercicio 1.

Realizar un programa en ensamblador que solicite al usuario dos números enteros y realice la suma de los mismos. El resultado de la suma deberá ser mostrado en pantalla.



#### 4.1.1. Código fuente.

*section .data*

*input\_msg db 'Introduce un numero: ', 0*

*result\_msg db 'El resultado es: ', 0*

*newline db 0x0A, 0 ; Salto de línea*

*section .bss*

*num1 resb 4 ; Espacio para el primer número*

*num2 resb 4 ; Espacio para el segundo número*

*sum resb 4 ; Espacio para la suma*

*section .text*

*global \_main*

*extern \_printf, \_scanf, \_ExitProcess*

*\_main:*

*; Pedir el primer número*

*push input\_msg*

*call \_printf*

*add esp, 4*

*; Leer el primer número (entero)*

*lea eax, [num1]*

*push eax*

*push format\_input*

*call \_scanf*

*add esp, 8*

*; Pedir el segundo número*

*push input\_msg*

*call \_printf*

*add esp, 4*



*; Leer el segundo número (entero)*

*lea eax, [num2]*

*push eax*

*push format\_input*

*call \_scanf*

*add esp, 8*

*; Sumar los dos números*

*mov eax, [num1]*

*add eax, [num2]*

*mov [sum], eax*

*; Mostrar el resultado*

*push result\_msg*

*call \_printf*

*add esp, 4*

*; Mostrar el valor de la suma*

*lea eax, [sum]*

*push dword [eax]*

*push format\_output*

*call \_printf*

*add esp, 8*

*; Saltar línea*

*push newline*

*call \_printf*

*add esp, 4*

*; Terminar el programa*

*push 0*

`call _ExitProcess`

`section .rodata`

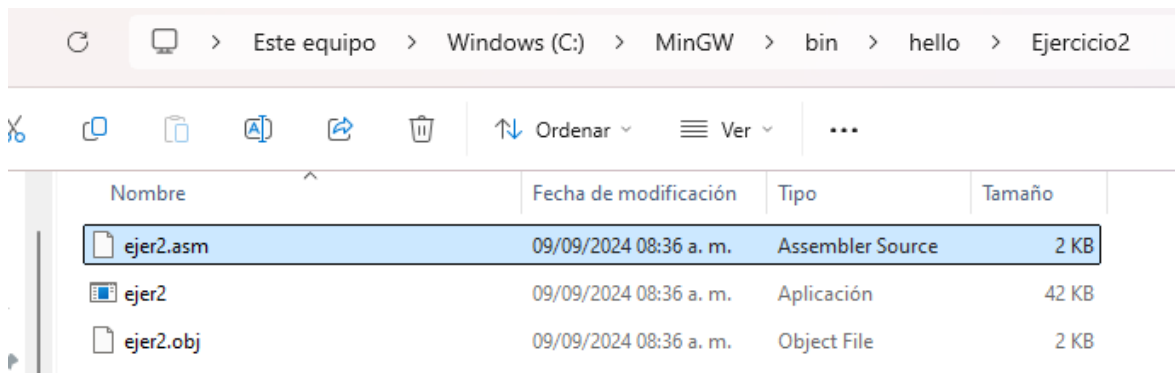
`format_input db '%d', 0 ; Leer números enteros`

`format_output db '%d', 0 ; Mostrar números enteros`

#### 4.1.2. Captura de pantalla del programa en ejecución.

```
C:\MinGW\bin\hello\Ejercicio2>C:\MinGW\bin\hello\Ejercicio2\ejer2.exe
Introduce un numero: 4
Introduce un numero: 5
El resultado es: 9
```

A continuación, se presenta los archivos que se crearon para la ejecución del código del ejercicio 1, en el cual me equivoqué en el nombre, el archivo debió llamarse ejercicio 1 .



Este equipo > Windows (C:) > MinGW > bin > hello > Ejercicio2				
Ordenar Ver ...				
Nombre	Fecha de modificación	Tipo	Tamaño	
ejer2.asm	09/09/2024 08:36 a. m.	Assembler Source	2 KB	
ejer2	09/09/2024 08:36 a. m.	Aplicación	42 KB	
ejer2.obj	09/09/2024 08:36 a. m.	Object File	2 KB	

#### 4.2. Ejercicio 2.

Realizar un programa en ensamblador que solicite al usuario un número entero y determine si el mismo es par o impar. El resultado deberá ser mostrado en pantalla.

##### 4.2.1. Código fuente.

`section .data`

`input_msg db 'Introduce un numero: ', 0`

`even_msg db 'El numero es par.', 0`

`odd_msg db 'El numero es impar.', 0`

`newline db 0x0A, 0 ; Salto de línea`

`section .bss`

`num resd 1 ; Reservar espacio para el número`

`section .text`



*global \_main*

*extern \_printf, \_scanf, \_ExitProcess*

*\_main:*

*; Pedir el número al usuario*

*push input\_msg*

*call \_printf*

*add esp, 4*

*; Leer el número*

*lea eax, [num]*

*push eax*

*push format\_input*

*call \_scanf*

*add esp, 8*

*; Verificar si es par o impar*

*mov eax, [num] ; Mover el número a eax*

*and eax, 1 ; Verificar el bit menos significativo*

*cmp eax, 0 ; Si el bit es 0, el número es par*

*je es\_par*

*; Si es impar, mostrar el mensaje correspondiente*

*push odd\_msg*

*call \_printf*

*add esp, 4*

*jmp fin*

*es\_par:*

*; Si es par, mostrar el mensaje correspondiente*

*push even\_msg*

*call \_printf*





*add esp, 4*

*fin:*

*; Saltar línea*

*push newline*

*call \_printf*

*add esp, 4*

*; Terminar el programa*

*push 0*

*call \_ExitProcess*

*section .rodata*

*format\_input db '%d', 0 ; Formato para leer enteros*

#### 4.2.2. Captura de pantalla del programa en ejecución.

```
C:\MinGW\bin\hello\Ejercicio2.2>C:\MinGW\bin\hello\Ejercicio2.2\PAR_IMPAR.exe
Introduce un numero: 6
El numero es par.

C:\MinGW\bin\hello\Ejercicio2.2>C:\MinGW\bin\hello\Ejercicio2.2\PAR_IMPAR.exe
Introduce un numero: 3
El numero es impar.
```

A continuación, se muestra la carpeta de archivos.

Nombre	Fecha de modificación	Tipo	Tamaño
PAR_IMPAR.asm	09/09/2024 08:56 a. m.	Assembler Source	2 KB
PAR_IMPAR	09/09/2024 08:59 a. m.	Aplicación	41 KB
PAR_IMPAR.obj	09/09/2024 08:57 a. m.	Object File	1 KB

#### 4.3 Ejercicio 3.

Realizar un programa en ensamblador que solicite al usuario un número entero y determine si el mismo es positivo, negativo o cero. El resultado deberá ser mostrado en pantalla.



#### 4.3.1. Código fuente.

*section .data*

*input\_msg db 'Introduce un numero: ', 0*

*pos\_msg db 'El numero es positivo.', 0*

*neg\_msg db 'El numero es negativo.', 0*

*zero\_msg db 'El numero es cero.', 0*

*newline db 0x0A, 0 ; Salto de línea*

*section .bss*

*num resd 1 ; Reservar espacio para el número*

*section .text*

*global \_main*

*extern \_printf, \_scanf, \_ExitProcess*

*\_main:*

*; Pedir el número al usuario*

*push input\_msg*

*call \_printf*

*add esp, 4*

*; Leer el número ingresado*

*lea eax, [num]*

*push eax*

*push format\_input*

*call \_scanf*

*add esp, 8*

*; Comparar el número*

*mov eax, [num] ; Mover el número a eax*

*cmp eax, 0 ; Comparar con 0*

*je es\_cero ; Si es igual a 0, saltar a es\_cero*



*jl es\_negativo ; Si es menor que 0, saltar a es\_negativo*

*; Si es positivo, mostrar el mensaje correspondiente*

*push pos\_msg*

*call \_printf*

*add esp, 4*

*jmp fin*

*es\_negativo:*

*; Si es negativo, mostrar el mensaje correspondiente*

*push neg\_msg*

*call \_printf*

*add esp, 4*

*jmp fin*

*es\_cero:*

*; Si es cero, mostrar el mensaje correspondiente*

*push zero\_msg*

*call \_printf*

*add esp, 4*

*fin:*

*; Saltar línea*

*push newline*

*call \_printf*

*add esp, 4*

*; Terminar el programa*

*push 0*

*call \_ExitProcess*

*section .rodata*

*format\_input db '%d', 0 ; Formato para leer enteros*

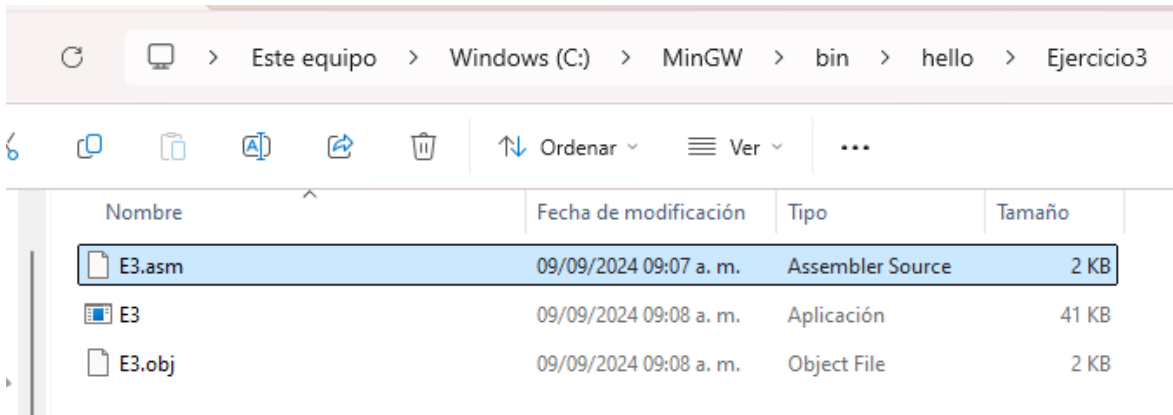
#### 4.3.2. Captura de pantalla del programa en ejecución.

```
C:\MinGW\bin\hello\Ejercicio3>C:\MinGW\bin\hello\Ejercicio3\E3.exe
Introduce un numero: 4
El numero es positivo.

C:\MinGW\bin\hello\Ejercicio3>C:\MinGW\bin\hello\Ejercicio3\E3.exe
Introduce un numero: -9
El numero es negativo.

C:\MinGW\bin\hello\Ejercicio3>C:\MinGW\bin\hello\Ejercicio3\E3.exe
Introduce un numero: 0
El numero es cero.
```

A continuación, se muestra la carpeta de los archivos creados.



Nombre	Fecha de modificación	Tipo	Tamaño
E3.asm	09/09/2024 09:07 a. m.	Assembler Source	2 KB
E3	09/09/2024 09:08 a. m.	Aplicación	41 KB
E3.obj	09/09/2024 09:08 a. m.	Object File	2 KB

#### 4.4 Ejercicio 4 (EXTRA).

Realizar un programa en ensamblador que solicite al usuario un número entero y determine si el mismo es primo o no. El resultado deberá ser mostrado en pantalla

##### 4.4.1. Código fuente.

*section .data*

*input\_msg db 'Introduce un numero: ', 0*

*prime\_msg db 'El numero es primo.', 0*

*not\_prime\_msg db 'El numero no es primo.', 0*

*newline db 0x0A, 0 ; Salto de línea*

*section .bss*

*num resd 1 ; Reservar espacio para el número*



*i resd 1 ; Variable de iteración para comprobar divisores*  
*remainder resd 1; Para almacenar el resto de la división*

*section .text*

*global \_main*

*extern \_printf, \_scanf, \_ExitProcess*

*\_main:*

*; Pedir el número al usuario*

*push input\_msg*

*call \_printf*

*add esp, 4*

*; Leer el número ingresado*

*lea eax, [num]*

*push eax*

*push format\_input*

*call \_scanf*

*add esp, 8*

*; Verificar si el número es menor o igual a 1*

*mov eax, [num]*

*cmp eax, 2*

*jl no\_es\_primo*

*; Inicializar i a 2*

*mov dword [i], 2*

*check\_prime:*

*; Comparar si  $i*i > num$*

*mov eax, [i]*

*imul eax, eax*



*cmp eax, [num]*

*jg es\_primo*

*; Calcular num % i*

*mov eax, [num]*

*mov ebx, [i]*

*xor edx, edx ; Limpiar edx para la división*

*div ebx ; eax / ebx, el resto queda en edx*

*cmp edx, 0*

*je no\_es\_primo ; Si el resto es 0, no es primo*

*; Incrementar i*

*inc dword [i]*

*jmp check\_prime*

*es\_primo:*

*; Mostrar el mensaje de que es primo*

*push prime\_msg*

*call \_printf*

*add esp, 4*

*jmp fin*

*no\_es\_primo:*

*; Mostrar el mensaje de que no es primo*

*push not\_prime\_msg*

*call \_printf*

*add esp, 4*

*fin:*

*; Saltar línea*

*push newline*

*call \_printf*

*add esp, 4*

*; Terminar el programa*

*push 0*

*call \_ExitProcess*

*section .rodata*

*format\_input db '%d', 0 ; Formato para leer enteros*

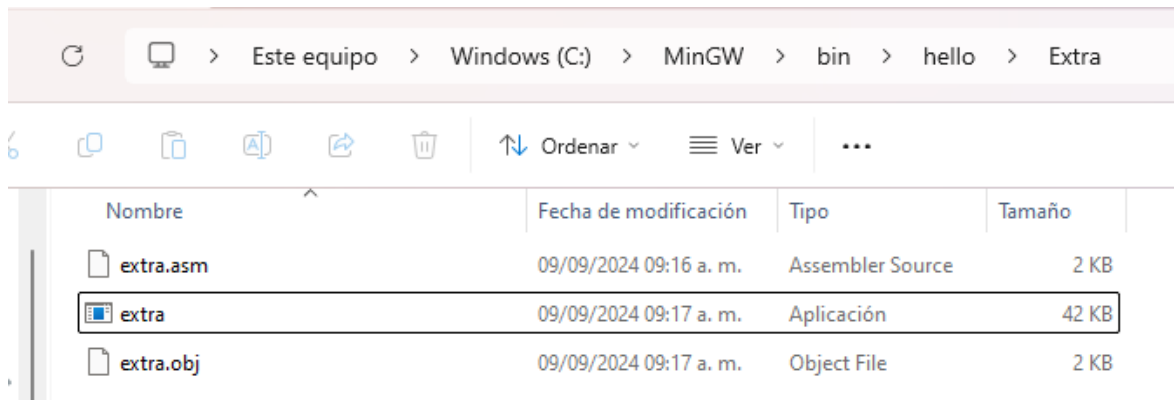
#### 4.4.2. Captura de pantalla del programa en ejecución.

```
C:\MinGW\bin\hello\Extra>C:\MinGW\bin\hello\Extra\extra.exe
Introduce un numero: 5
El numero es primo.

C:\MinGW\bin\hello\Extra>C:\MinGW\bin\hello\Extra\extra.exe
Introduce un numero: 8
El numero no es primo.

C:\MinGW\bin\hello\Extra>C:\MinGW\bin\hello\Extra\extra.exe
Introduce un numero: 2
El numero es primo.
```

A continuación, se muestra la carpeta de archivos, que fueron creados.



Este equipo > Windows (C:) > MinGW > bin > hello > Extra				
Ordenar Ver ...				
Nombre	Fecha de modificación	Tipo	Tamaño	
extra.asm	09/09/2024 09:16 a. m.	Assembler Source	2 KB	
extra	09/09/2024 09:17 a. m.	Aplicación	42 KB	
extra.obj	09/09/2024 09:17 a. m.	Object File	2 KB	



## CONCLUSIÓN

A través de esta práctica hemos podido aprender a utilizar nasm, utilizando diferentes comandos, lo primero que hicimos fue empezar a diseñar el código, por ejemplo sumar dos números, que fueran ingresados desde la consola, desde el bloc de notas, porque se nos hizo más fácil, aunque también se puede hacer en otras aplicaciones como Visual Studio 2022 con la extensión .asm, posteriormente abrimos nasm-shell, y accedimos a la carpeta donde teníamos guardado el archivo, esto con el ayuda del comando `cd..` , una vez dentro de la carpeta escribimos el siguiente comando `nasm -fwin32 Archivo.asm`, el cual nos permite crear un archivo de tipo .obj, finalmente se utilizó el siguiente comando `gcc Archivo.obj -o Archivo.exe`, este comando nos permite crear un ejecutable, el cual tenemos que arrastrar hasta la consola, y presionar enter. Gracias esto pudimos aprender a trabajar con comandos y direcciones de archivos, y conocer una la sintaxis del lenguaje ensamblador, la cual es algo extensa comparada a otros lenguajes como Java o C#, que es con los que hemos trabajado anteriormente.





## REFERENCIAS BIBLIOGRÁFICAS

EDUCA EDTECH GROUP. (2019 de Mayo de 09). *¿Qué es el lenguaje ensamblador (ASM)?* Obtenido de *¿Qué es el lenguaje ensamblador (ASM)?*:  
<https://www.educaopen.com/digital-lab/blog/software/lenguaje-ensamblador>