



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Facultad de ingeniería

Estructura y programación de computadoras

Grupo: 06

Manual de uso de ensamblador Z80

Semestre 2024-2

Gonzalez Cuellar Pablo Arturo

ÍNDICE

Manual de Usuario	3
Descripción general	3
Requisitos para la ejecución	3
Forma de ejecución	4
Ejecución	4
Errores a la hora de ensamblado	7
Restricciones del programa a ensamblar	9
Manual Técnico	10
Descripción General del Proyecto	10
Instalación y Configuración	10
Flujo general del programa del ensamblador	11
Flujo general del programa del traductor	11

Manual de Usuario

Descripción general

El documento abarca el funcionamiento y uso del programa que tiene como objetivo ensamblar programas con arquitectura del microprocesador Z-80. Éste programa al momento de ensamblar genera dos archivos con extensión “.LST” y “.HEX” correspondientes al programa ensamblado.

El programa “Ensamblador” en lenguaje python utiliza un “Traductor” que pasa de código ensamblador a código hexadecimal.

Éste código utiliza una tabla Hash como variante de las tablas de la arquitectura Z-80, dónde se encuentran todos los mnemónicos. Además, hace uso de una librería “re” para utilizar expresiones regulares.

Requisitos para la ejecución

El programa principal en cuestion y principal “Ensamblador.py”, necesita de las siguientes consideraciones para su correcto funcionamiento:

- Entorno de desarrollo **python** el cual es requisito **indispensable** ya que el desarrollo del programa se basa en este lenguaje de programación
- El archivo **MNEMÓNICOS.txt** el cual contendrá los mnemónicos necesarios para traducir nuestro programa y que el ensamblador lo reconozca.
- Un archivo con extensión “**.ASM**” debe de contener nuestro programa a ensamblar.
- En la misma carpeta donde estará el archivo “**MNEMÓNICOS.txt**” y el archivo con extensión “**.ASM**” deberán de estar los los programas “**Ensamblador.py**” y el “**Traductor.py**”.

Forma de ejecución

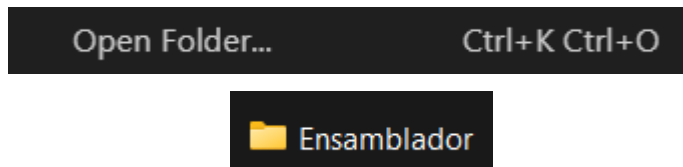
1. El programa podrá ejecutarse en alguna de las siguientes opciones
 - a. IDE de tu preferencia
 - b. Desde consola de comandos (cmd)

1.1. En cualesquiera de los dos casos se debe tener los archivos mencionados en “Requisitos para la ejecución” en el mismo directorio

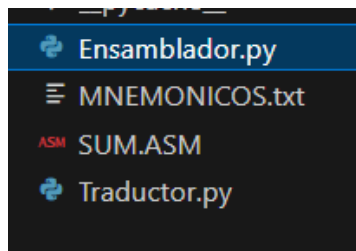
Ejecución

Para el caso “a” pondremos como ejemplo la ejecución en el IDE “Visual Studio Code”

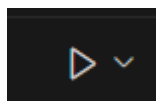
- I. Abrimos la carpeta donde se hallan los archivos necesarios para la ejecución.



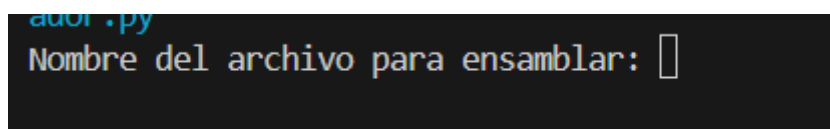
- II. Ya abierto, nos dirigimos a nuestro programa principal “Ensamblador”.



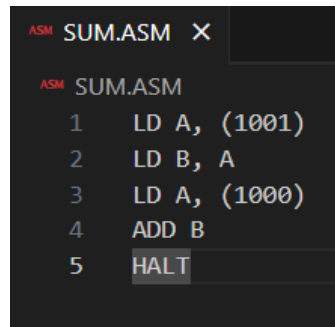
- III. Localizamos la opción que nos ofrece el IDE para correr el archivo antes mencionado



- IV. Automáticamente se abrirá una terminal de ejecución y en ella nos pedirá el archivo a ensamblar

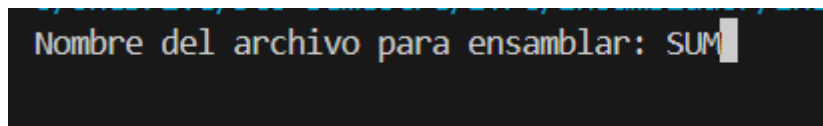


- V. Como ejemplo, en nuestra carpeta probaremos **"SUM.ASM"** el cual contiene un programa que realiza la suma del contenido de dos localidades de memoria



```
ASM SUM.ASM X
ASM SUM.ASM
1 LD A, (1001)
2 LD B, A
3 LD A, (1000)
4 ADD B
5 HALT
```

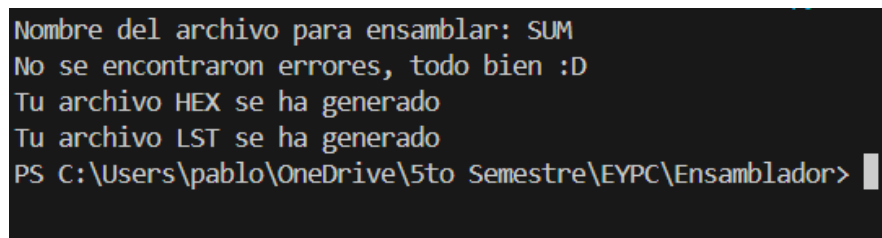
- VI. Para este caso, le pasamos como argumento a la terminal **solamente el nombre** del archivo a ensamblar **sin la extensión** del mismo, osea:



```
Nombre del archivo para ensamblar: SUM
```

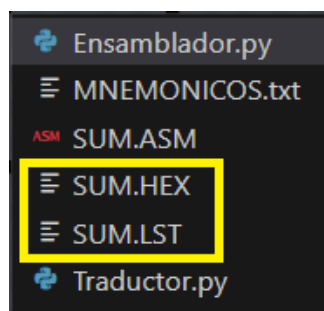
Y le damos **ENTER**

- VII. Automáticamente el programa comenzará el ensamblado si es que **NO** tiene errores (se revisarán los posibles errores de ejecución en el punto "*****") y mandará el siguiente mensaje:



```
Nombre del archivo para ensamblar: SUM
No se encontraron errores, todo bien :D
Tu archivo HEX se ha generado
Tu archivo LST se ha generado
PS C:\Users\pablo\OneDrive\5to Semestre\EYPC\Ensamblador>
```

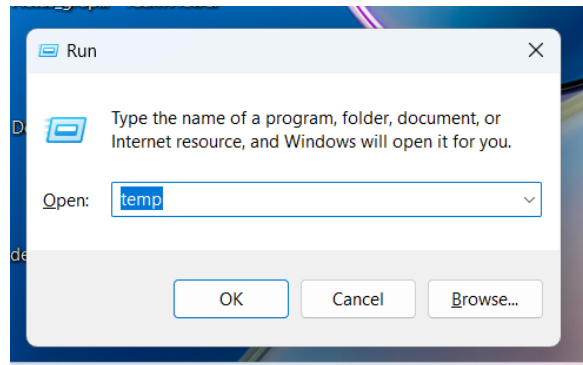
El cual quiere decir que el ensamblado fue exitoso y nos generará los archivos **".HEX"** y **".LST"**



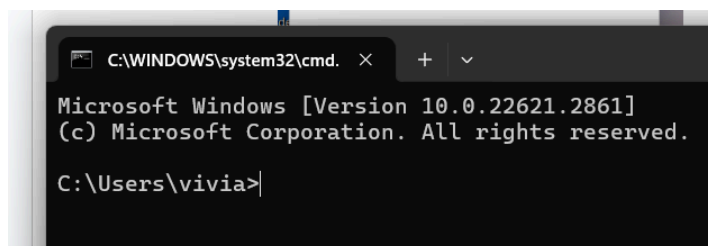
```
Ensamblador.py
MNEMONICOS.txt
ASM SUM.ASM
SUM.HEX
SUM.LST
Traductor.py
```

Para el caso “b” desde la consola de windows

- I. En el teclado pulsa la tecla “Windows” y la “R” al mismo tiempo. Debe de abrirse la siguiente ventana.



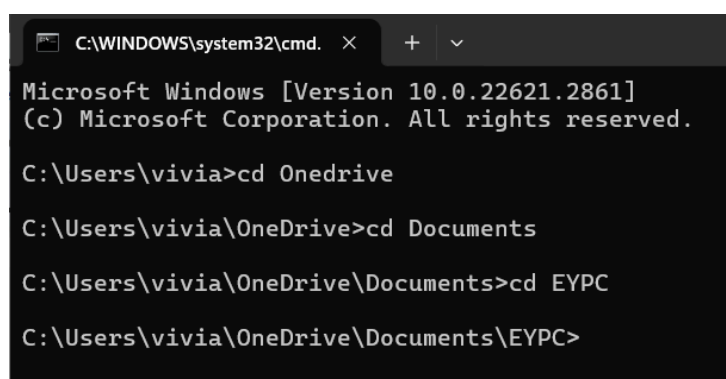
- II. En el rectángulo posicionado a un lado de “Abrir/Open” deberás de escribir cmd y darle al botón “Aceptar/OK” o pulsar la tecla “Enter”. Se abrirá la consola de comandos, debe de verse similar a lo siguiente:



- III. Debemos de ubicarnos en la carpeta donde se encuentran los 4 archivos entonces utilizaremos el siguiente comando las veces que sea necesaria hasta estar en donde están localizados:

-cd nombredelacarpeta

Aquí un ejemplo de cómo deberá de verse:



- IV. Para ejecutar el programa deberemos de poner la instrucción “**python.exe Ensamblador.py**”. Cuando se ejecute pedirá el nombre del archivo a ensamblar, SOLO debe de ponerse el nombre SIN la extensión “.asm” de lo contrario saltará un mensaje de error.

```
C:\Users\vivia\OneDrive\Documents\EYPC>python.exe Ensamblador.py
Nombre del archivo para ensamblar: MUL
No se encontraron errores, todo bien :D
Tu archivo HEX se ha generado
Tu archivo LST se ha generado
```

Errores a la hora de ensamblado

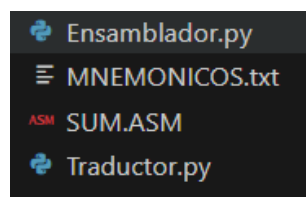
En esta sección del manual, se atenderán los posibles errores a la hora de la ejecución del programa

1. NO SE ENCUENTRA EL ARCHIVO “xxxxxx.ASM”

```
Nombre del archivo para ensamblar: ASM
No se encuentra el archivo de mnemonicos en el directorio
PS C:\Users\pablo\OneDrive\5to Semestre\EYPC\Ensamblador>
```

```
Nombre del archivo para ensamblar: SUM
SUM.ASM no localizado
PS C:\Users\pablo\OneDrive\5to Semestre\EYPC\Ensamblador>
```

Este error sucede cuando no se tienen cargados los archivos en la carpeta donde se está ejecutando el ensamblador, la carpeta se debe de ver así:



2. ARCHIVO “XXXXXX.ASM.ASM” NO ENCONTRADO

```
Nombre del archivo para ensamblar: SUM.ASM
SUM.ASM.ASM no localizado
PS C:\Users\pablo\OneDrive\5to Semestre\EYPC\Ensamblador>
```

Este error es causado debido a que se ingresó incorrectamente el **nombre** del programa a ensamblar, el cual no debe contener la extensión de la siguiente forma:

```
Nombre del archivo para ensamblar: SUM
```

3. NO SE PUDO GENERAR EL ARCHIVO ".HEX"

```
No se pudo generar el archivo HEX  
Tu archivo LST se ha generado
```

Este error sucede cuando hay un error de sintaxis en nuestro archivo **".ASM"**, el cual se muestra en la parte de arriba:

```
Nombre del archivo para ensamblar: SUM  
Hay errores en:  
ERROR DE SINTAXIS LD B,A  
Hay error en el código objeto  
No se pudo generar el archivo HEX  
Tu archivo LST se ha generado
```

4. ERROR DE SINTAXIS

Los errores de sintaxis ocurren cuando hay inconsistencias en la sintaxis de nuestro archivo **".ASM"**.

```
Hay errores en:  
ERROR DE SINTAXIS LD B,A  
Hay error en el código objeto
```

Estos errores se abordarán con más profundidad en el siguiente apartado.

Restricciones del programa a ensamblar

Para el archivo que se ensamblará se debe de tener en cuenta ciertas restricciones en su cuerpo, tales como:

- Las instrucciones/mnemónicos deben de estar en mayúsculas.
- El primer operando debe de estar separado del instrucción/mnemónico.
- Si hay una instrucción que use dos registros como operandos, se deberá seguir un formato:

LD {registro}, {registro}

- Las direcciones de memoria no pueden contener el indicador H de hexadecimal, solo los números de la misma dirección.
- Las etiquetas deben de estar totalmente en minúsculas y no deben de contener números "1,2,3,..".
- Las instrucciones aritméticas aceptan un SOLO argumento el cual será operado con lo que contenga el registro acumulador "A".

Manual Tecnico

Descripción General del Proyecto

- Componentes del proyecto:
 - **ensamblador.py**: Programa principal que orquesta el ensamblaje.
 - **Traductor.py**: Contiene las instrucciones y métodos necesarios para la traducción.
 - **Archivo .ASM**: Programa a ensamblar.
 - **MNEMÓNICOS.txt**: Tabla de mnemónicos con las instrucciones y su representación en código máquina.

Instalación y Configuración

- **Requisitos del sistema:**
 - **Python 3.12 (PREFERENTE)**

Para asegurar la correcta ejecución del ensamblador Z-80, se recomienda encarecidamente utilizar Python 3.12. Si bien el ensamblador puede funcionar en versiones anteriores de Python, la compatibilidad no está garantizada y se podrían experimentar problemas de rendimiento, compatibilidad y seguridad.
 - **EDITOR DE TEXTO / IDE**

Cualquier editor de texto avanzado o entorno de desarrollo integrado (IDE) compatible con Python, como Visual Studio Code, PyCharm, o Sublime Text.
 - **HERRAMIENTAS DE LA LINEA DE COMANDOS**

Acceso a la línea de comandos/terminal para ejecutar scripts y comandos de instalación.

Instalación de Python 3.12

1. Descargar Python 3.12:
2. Visite el sitio web oficial de Python en python.org.
 - a. Navegue a la sección de descargas y seleccione la versión 3.12 para su sistema operativo.
 - b. Instalar Python 3.12:

3. Siga las instrucciones de instalación proporcionadas para su sistema operativo.
4. Asegúrese de seleccionar la opción "Add Python to PATH" durante la instalación para facilitar la ejecución desde la línea de comandos.
5. Verificar la Instalación:

```
python --version
```

Flujo general del programa del ensamblador

Inicialización: Carga los mnemónicos desde un archivo y prepara la tabla hash.

Lectura del archivo fuente: Lee el archivo .ASM de entrada.

Primer pase: Llama a la función Pasadas para procesar el código fuente.

Generación de archivos:

- Si no hay errores, genera el archivo .HEX.
- Genera el archivo .LST independientemente de si hay errores o no.

Flujo general del programa del traductor

Lectura de línea: La función TRAD recibe una línea de código en ensamblador y extrae los mnemónicos y operandos.

Traducción de instrucciones: Convierte las instrucciones en ensamblador a código máquina utilizando las expresiones regulares y diccionarios de mnemónicos.

Cálculo de desplazamientos y direcciones: Maneja la traducción de direcciones y desplazamientos en el código ensamblador.

Complemento a dos: Si es necesario, calcula el complemento a dos de números binarios.