



TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE TLAXIACO

Práctica 7. Simulación de administración de memoria en la CPU.

Presenta:

Santiago Bautista Maribel 22620262

Carrera:

Ingeniería en Sistemas Computacionales

Asignatura:

Arquitectura de Computadoras

Docente:

Ing. Edward Osorio Salinas

Tlaxiaco, Oax., 12 de diciembre del 2024.



"Educación, Ciencia y Tecnología, Progresos día con día"®



ÍNDICE

INTRODUCCIÓN	3
OBJETIVO	3
MATERIALES	3
SOFTWARE	3
PROCEDIMIENTO	4
DESARROLLO	4
RESULTADOS.....	7
CONCLUSIÓN	7
REFERENCIAS.....	8



PRÁCTICA 7. SIMULACIÓN DE ADMINISTRACIÓN DE MEMORIA EN LA CPU.

INTRODUCCIÓN

OBJETIVO

El objetivo de esta práctica es simular el funcionamiento de un sistema operativo y realizar un resumen de los pasos que se llevan a cabo en la administración de memoria en la CPU. Estos pasos son:

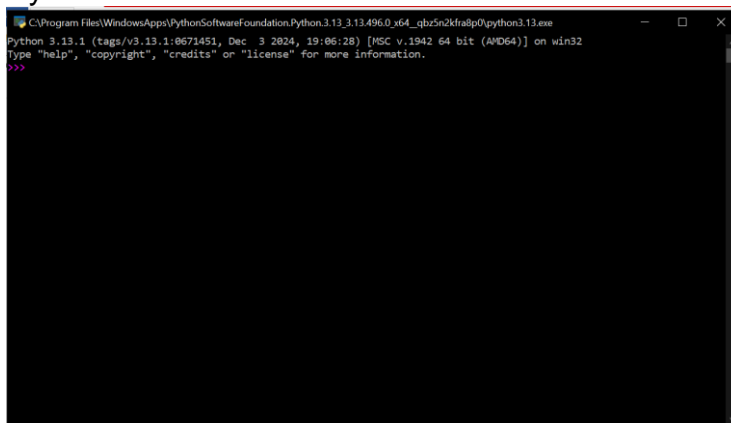
1. Cargar el programa en memoria.
2. Asignar espacio de memoria a las variables.
3. Asignar espacio de memoria a las instrucciones.
4. Ejecutar el programa.

MATERIALES

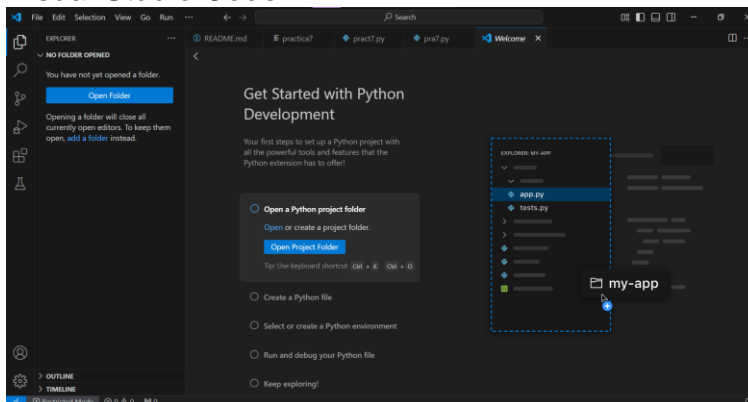
- Laptop
- Visual Studio Code previamente instalado.
- Python 3.13. previamente instalado.

SOFTWARE

Python 3.13.



Visual Studio Code



PROCEDIMIENTO

1. Cargar el programa en memoria.

Esto se puede simular con un diccionario que contenga las instrucciones del programa.

2. Asignar espacio de memoria a las variables.

Esto se puede simular con un array que contenga las variables del programa y que se asigne un espacio de memoria a cada una.

3. Asignar espacio de memoria a las instrucciones.

Esto se puede simular con un array que contenga las instrucciones del programa y que se ejecute una por una.

4. Ejecutar el programa.

Esto se puede simular con un ciclo que recorra las instrucciones del programa y las ejecute, asignando los valores de las variables a las instrucciones.

5. Mostrar el resultado

Al finalizar la ejecución del programa, se debe mostrar el resultado de las variables.

DESARROLLO

1. Cargar el programa en memoria

```
1  # Programa cargado en memoria
2  programa = {
3      0: "cargar 1, 5",
4      1: "cargar 2, 10",
5      2: "sumar 1, 2, 3",
6      3: "imprimir 3"
7  }
```

Este diccionario simula las instrucciones del programa cargado en memoria.

La claves (0, 1, 2, 3) representan las direcciones de memoria.

Los valores ("cargar 1, 5", "sumar 1, 2, 3", etc) son las instrucciones que el programa ejecutará.

2. Definir el espacio de memoria para variables

```
9  # Espacio de memoria para variables
10 variables = {
11     1: 0, # Variable 1
12     2: 0, # Variable 2
13     3: 0  # Variable 3 (resultado)
14 }
```

Este diccionario representa las variables del programa.

Las claves (1, 2, 3) son los identificadores de las variables en memoria.

Los valores (0) son los datos iniciales asignados a estas variables.

3. Convertir las instrucciones en una lista

```
16 # Lista de instrucciones a ejecutar
17 instrucciones = list(programa.values())
18
```

Convierte las instrucciones del diccionario `programa` en una lista para recorrerlas fácilmente.

4. Recorrer las instrucciones y ejecutarlas.

```
19 # Ciclo para ejecutar las instrucciones
20 for instruccion in instrucciones:
```

Un ciclo `for` recorre cada instrucción en la lista `instrucciones`.

En cada iteración, se evalúa el tipo de instrucción y se ejecuta el bloque de código correspondiente

🔧 Instrucción: "cargar"

```
21     if instruccion.startswith("cargar"):
22         _, var, valor = instruccion.split(" ")
23         var = int(var.strip(","))
24         valor = int(valor)
25         variables[var] = valor
```


- ✓ `instruccion.startswith("cargar")`: Verifica si la instrucción comienza con la palabra `cargar`.
- ✓ `instruccion.split(" ")`: Divide la instrucción en partes separadas por espacios.
- ✓ `var, valor = ...`: Extrae la variable y el valor.
- ✓ `var = int(var.strip(","))`: Quita la coma del final del número y lo convierte en entero.
- ✓ `valor = int(valor)`: Convierte el valor a entero.
- ✓ `variables[var] = valor`: Asigna el valor a la variable correspondiente en el diccionario `variables`.

🔧 Instrucción: "sumar"

```
26     elif instruccion.startswith("sumar"):
27         _, var1, var2, var3 = instruccion.split(" ")
28         var1 = int(var1.strip(","))
29         var2 = int(var2.strip(","))
30         var3 = int(var3)
31         variables[var3] = variables[var1] + variables[var2]
```

- ✓ `instruccion.startswith("sumar")`: Verifica si la instrucción comienza con la palabra `sumar`.

- ✓ `instruccion.split(" ")`: Divide la instrucción en partes.
- ✓ `var1, var2, var3, =...`: Extrae las variables involucradas.
- ✓ `var1 = int(var1.strip(","))`: Limpia y convierte `var1` a entero.
- ✓ `variables[var3] = variables[var1] + variables[var2]`: Suma los valores de las variables `var1` y `var2`, y almacena el resultado en `var3`.

 Instrucción: "imprimir"

```
32     elif instruccion.startswith("imprimir"):
33         _, var = instruccion.split(" ")
34         var = int(var)
35         print(f"Variable {var}: {variables[var]}")
36
```

- ✓ `instruccion.startswith("imprimir")`: Verifica si la instrucción comienza con la palabra `imprimir`.
- ✓ `instruccion.split(" ")`: Divide la instrucción en partes.
- ✓ `var = int(var)`: Convierte `var` a entero.
- ✓ `print(...)`: Imprime el valor de la variable indicada.

5. Mostrar el estado final de las variables.

```
37     # Mostrar el estado final de las variables
38     print("Estado final de las variables:", variables)
39
```

Imprime el estado final del diccionario `variables`, mostrando el valor de cada variable tras ejecutar todas las instrucciones.

RESULTADOS

```
Símbolo del sistema
Microsoft Windows [Versión 10.0.19045.5131]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\EQUIPO>cd Desktop

C:\Users\EQUIPO\Desktop>python pra7.py
Variable 3: 15
Estado final de las variables: {1: 5, 2: 10, 3: 15}

C:\Users\EQUIPO\Desktop>_
```

En cmd se ingresa el directorio del archivo el cual es Desktop, posteriormente se ejecuta el archivo utilizando `python pra7.py` el cual es el nombre de nuestro archivo, por lo cual se ejecuta el siguiente resultado:

```
Variable 3: 15
Estado final de las variables: {1: 5, 2: 10, 3: 15}
```

El resultado de nuestro programa en python realiza las siguientes instrucciones:

1. Se cargan las instrucciones en memoria.
2. Se inicializan las variables con valores predeterminados (0).
3. Se recorren las instrucciones y se ejecutan en orden:
 - ✚ **cargar**: Asigna valores iniciales.
 - ✚ **sumar**: Realiza operaciones aritméticas.
 - ✚ **imprimir**: Muestra el valor de una variable.
4. Se imprime el estado final de todas las variables.

CONCLUSIÓN

A través de la presente práctica aprendimos a utilizar Visual Studio Code junto con la consola de Python, consideramos que el lenguaje de programación Python es muy fácil de aprender, ya que tiene una sintaxis muy clara y sencilla, ya que anteriormente habíamos trabajado con Java y con C#, los cuales son más robustos,



otra característica que tiene es que permite la creación rápida de programas con menos líneas de código. Además, la simulación de la administración de memoria en Python que nosotros hicimos en la presente práctica se llevó a cabo mientras el programa se ejecutaba, donde se declararon variables, ciclos y listas de instrucciones.

REFERENCIAS

Gómez, J. J. (23 de Enero de 2023). *Variables en Python*. Obtenido de Variables en Python: <https://j2logo.com/python/tutorial/variables-python/>

Laern Microsoft. (12 de Diciembre de 2024). *Instalar y configurar Visual Studio Code para el desarrollo de Python*. Obtenido de Instalar y configurar Visual Studio Code para el desarrollo de Python: <https://learn.microsoft.com/es-es/training/modules/python-install-vscode/>

Wilkins, J. (29 de Abril de 2023). *El mejor manera para aprender Python: Tutorial de programación en Python para principiantes*. Obtenido de El mejor manera para aprender Python: Tutorial de programación en Python para principiantes: <https://www.freecodecamp.org/espanol/news/el-mejor-manera-para-aprender-python-tutorial-de-programacion-en-python-para-principiantes/>