INGENIERÍA EN SISTEMAS

Equipo 2

SIMULADOR DE ENSAMBLADOR Y

Memoria de una CPU simple

Lenguajes y Automatas II

SEPTIMO SEMESTRE

INTEGRANTES

de equipo

01 Gonzalez zenon Fernanda

02 Cauich Yah Yaret Emanuel

03 Méndez Cortez Carlos

Código ensamblador:

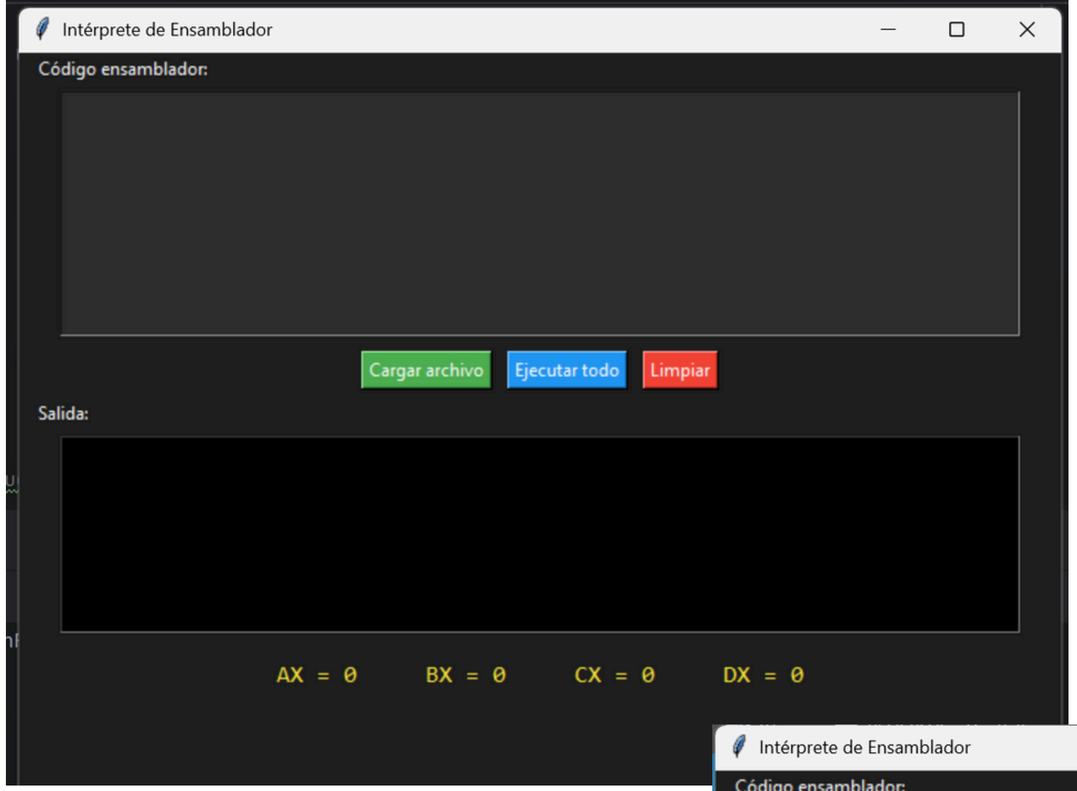
```
INC BX          ; BX = BX + 1 (3 + 1 = 4)
DEC AX          ; AX = AX - 1 (8 - 1 = 7)
PRINT BX        ; Mostrar BX
PRINT AX        ; Mostrar AX

CLR DX          ; DX = 0
PRINT DX        ; Mostrar DX

ADD BX, 10      ; BX = BX + 10 (4 + 10 = 14)
PRINT BX        ; Mostrar BX final
```

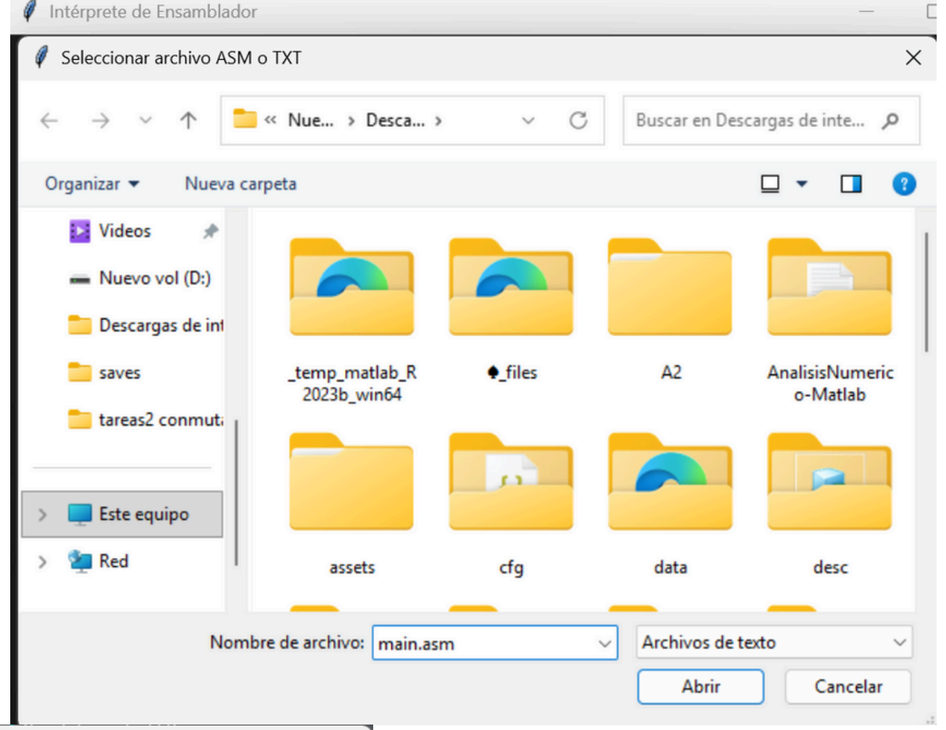
Cargar archivo | Ejecutar todo | Limpiar

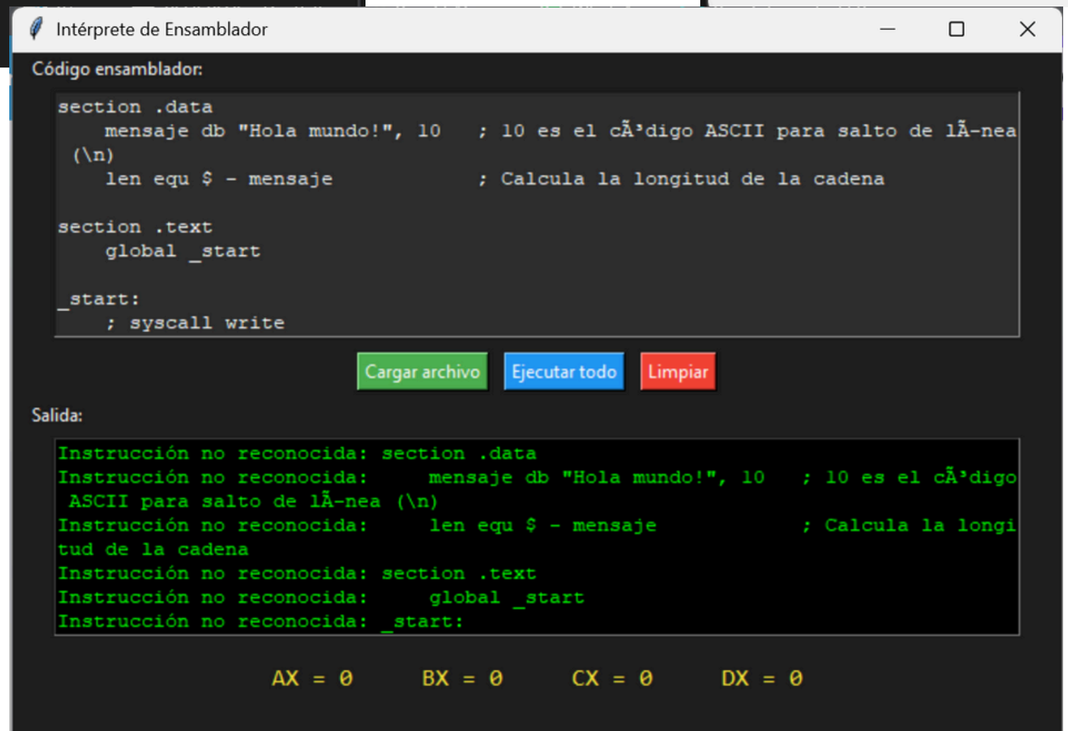
Salida:

```
AX = 8
CX = 8
BX = 4
AX = 7
DX = 0
BX = 14
```

AX = 7 BX = 14 CX = 8 DX = 0


Intérprete de Ensamblador
Código ensamblador:
Cargar archivo
Ejecutar todo
Limpiar
Salida:
AX = 0
BX = 0
CX = 0
DX = 0


Intérprete de Ensamblador
Seleccionar archivo ASM o TXT
Nue... › Desca... ›
Buscar en Descargas de inte...
Organizar
Nueva carpeta
Videos
Nuevo vol (D:)
Descargas de int
saves
tareas2 conmut
Este equipo
Red
_temp_matlab_R 2023b_win64
_files
A2
AnalisisNumerico-Matlab
assets
cfg
data
desc
Nombre de archivo:
main.asm
Archivos de texto
Abrir
Cancelar


Intérprete de Ensamblador
Código ensamblador:
section .data
mensaje db "Hola mundo!", 10 ; 10 es el cÃ³digo ASCII para salto de lÃ-nea (\n)
len equ $ - mensaje ; Calcula la longitud de la cadena
section .text
global _start
_start:
; syscall write
Cargar archivo
Ejecutar todo
Limpiar
Salida:
Instrucción no reconocida: section .data
Instrucción no reconocida: mensaje db "Hola mundo!", 10 ; 10 es el cÃ³digo ASCII para salto de lÃ-nea (\n)
Instrucción no reconocida: len equ $ - mensaje ; Calcula la longitud de la cadena
Instrucción no reconocida: section .text
Instrucción no reconocida: global _start
Instrucción no reconocida: _start:
AX = 0
BX = 0
CX = 0
DX = 0

Actividades

1. Definir un conjunto mínimo de instrucciones (MOV, ADD, SUB, INC, DEC, PRINT).
2. Programar un intérprete que lea instrucciones desde un archivo o lista de texto.
3. Implementar un cclo que procese cada línea e interprete su operación.
4. Mostrar resultados en pantalla conforme se ejecuten.
5. Probar con distintos programas de ejemplo

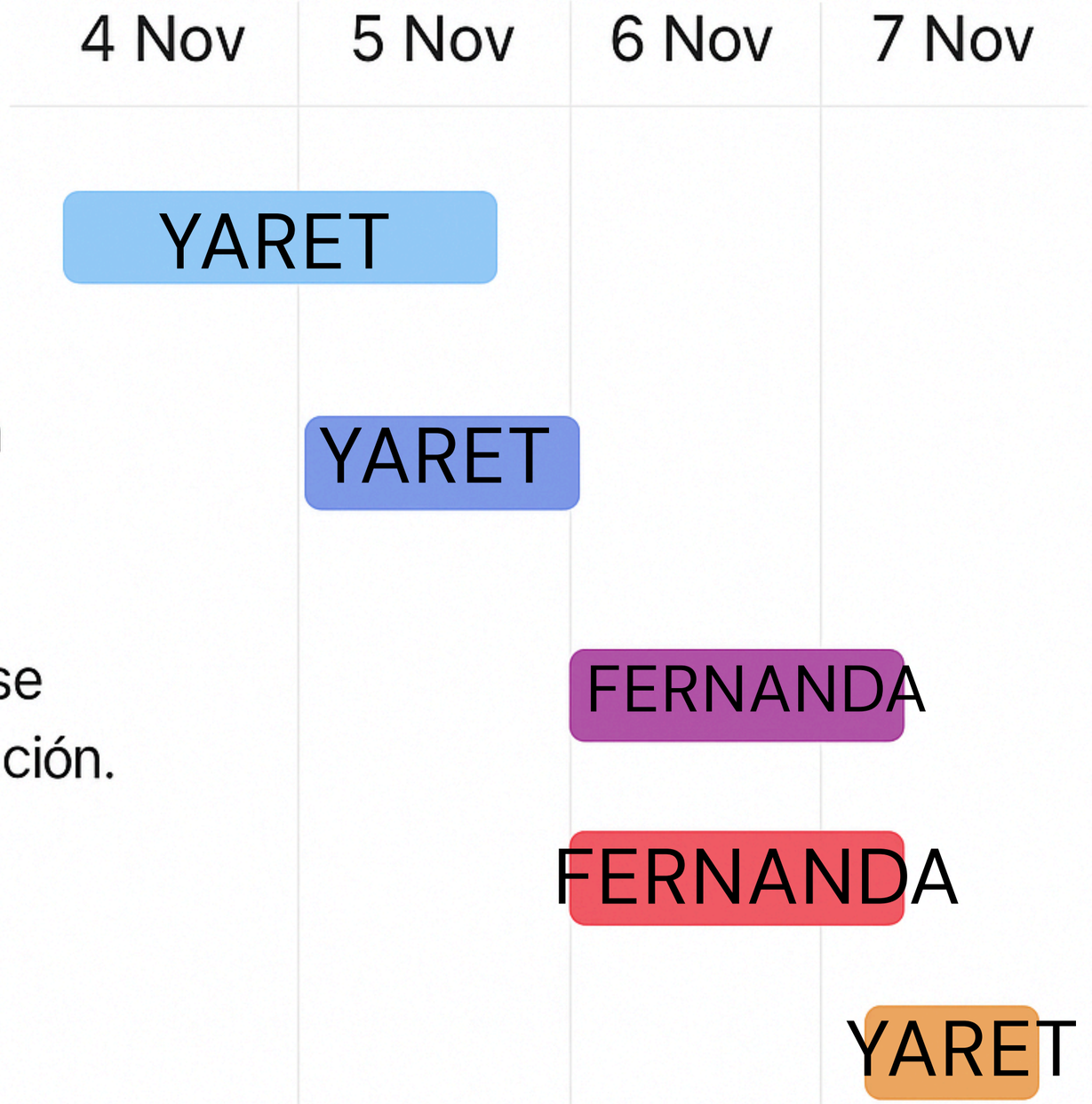


Diagrama de Gantt - Proyecto de Simulación e Intérprete

Semana	Actividad	Responsable	Fecha Inicio	Fecha Fin
20-24 oct	Leer sobre el proceso de compilación y las etapas que llevan al código objeto	Fernanda	20 oct	20 oct
20-24 oct	Investigar los conceptos: registro, ensamblador, lenguaje máquina y memoria	Yaret	21 oct	21 oct
20-24 oct	Realizar un esquema de transformación de instrucciones a código máquina	Carlos	22 oct	22 oct
20-24 oct	Diseñar documento de planeación del proyecto	Fernanda, Yaret, Carlos	23 oct	24 oct
27-31 oct	Investigar qué tipos de registros existen (AX, BX, CX, DX, PC, IR)	Fernanda	27 oct	27 oct
27-31 oct	Analizar cómo los registros almacenan y transfieren datos	Yaret	28 oct	28 oct
27-31 oct	Crear un programa que simule 4 registros principales	Carlos	29 oct	29 oct
27-31 oct	Implementar instrucciones básicas (MOV, CLR, etc.)	Fernanda	30 oct	30 oct
27-31 oct	Probar con varios valores y mostrar los resultados	Yaret	31 oct	31 oct
27-31 oct	Entrega final y diagrama actualizado	Carlos	3 nov	3 nov
4-7 nov	Definir un conjunto mínimo de instrucciones (MOV, ADD, SUB, INC, DEC, PRINT)	Fernanda	4 nov	5 nov
4-7 nov	Programar un intérprete que lea instrucciones desde un archivo o lista de texto	Yaret	5 nov	5 nov
4-7 nov	Implementar un ciclo que procese cada línea e interprete su operación	Carlos	6 nov	6 nov
4-7 nov	Mostrar resultados en pantalla conforme se ejecuten	Fernanda	6 nov	6 nov
4-7 nov	Probar con distintos programas de ejemplo	Yaret	7 nov	7 nov

Explicación de cómo se traduce cada instrucción

El intérprete desarrollado simula un pequeño procesador con cuatro registros principales: AX, BX, CX y DX.

Cada registro actúa como una celda de memoria que puede almacenar valores numéricos y ser modificada por medio de instrucciones escritas en lenguaje ensamblador.

El programa lee cada línea del archivo o área de texto, la separa en partes (operación y operandos), y ejecuta la acción correspondiente modificando el valor de los registros o mostrando información en pantalla.

A continuación se explica cómo se traduce cada instrucción en el lenguaje Python utilizado por el intérprete.

1. MOV REG, VALOR / MOV REG1, REG2

Función: Copia un valor numérico o el contenido de otro registro en el registro destino.

Traducción:

Si el segundo operando es un número, se convierte a entero y se asigna directamente.

Si es otro registro, se copia su valor.

```
registros[dest] = registros[src]  # si src es otro registro
registros[dest] = int(src)        # si src es un valor
```

Ejemplo:

MOV AX, 5 → AX = 5

MOV BX, AX → BX = AX = 5

2. ADD REG1, REG2 / ADD REG, VALOR

Función: Suma el contenido del segundo operando al primer registro.

Traducción:

```
registros[dest] += registros[src]  # si src es registro
registros[dest] += int(src)        # si src es valor
```

Ejemplo:

ADD AX, BX → AX = AX + BX

ADD BX, 10 → BX = BX + 10

3. SUB REG1, REG2 / SUB REG, VALOR

Función: Resta el segundo operando al primer registro.

Traducción:

```
registros[dest] -= registros[src]  # si src es registro
registros[dest] -= int(src)        # si src es valor
```

Ejemplo:

SUB CX, DX → CX = CX - DX

4. INC REG

Función: Incrementa el valor del registro en una unidad.

Traducción:

```
registros[reg] += 1
```

Ejemplo:

INC BX → BX = BX + 1

5. DEC REG

Función: Decrementa el valor del registro en una unidad.

Traducción:

registros[reg] -= 1

Ejemplo:

DEC AX → AX = AX - 1

6. CLR REG

Función: Limpia el registro, estableciendo su valor en cero.

Traducción:

registros[reg] = 0

Ejemplo:

CLR DX → DX = 0

7. PRINT REG

Función: Muestra el contenido actual del registro en la salida del programa.

Traducción:

print(f"{reg} = {registros[reg]}")

En la versión con interfaz gráfica, el resultado se envía a un cuadro de texto visible para el usuario.

Ejemplo:

PRINT AX → Muestra en pantalla: AX = 8