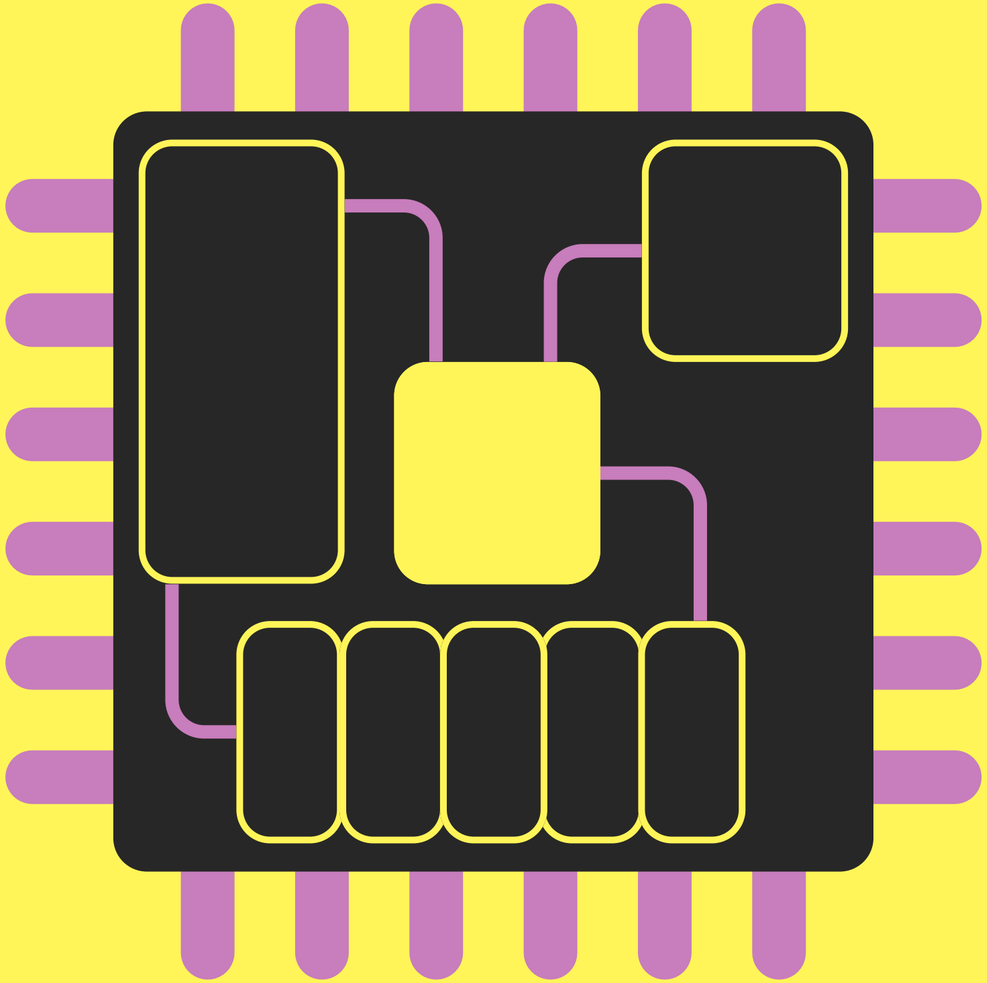
SISTEMAS OPERATIVOS

# Simulador de Procesos


- Miguel Angel Garcia Osorio
- Luis Alfonso Agudelo
- Julian Lara Aristizabal


# Descripción del Programa

El simulador de procesos es una implementación en C++ que replica el comportamiento de un sistema operativo utilizando el algoritmo de planificación Round Robin. El programa simula la ejecución concurrente de múltiples procesos, gestionando su cambio de contexto y la ejecución de instrucciones básicas.



# ¿COMO FUNCIONA?

1. Parser de Procesos (parser.cpp/.h)

- Función: Carga la configuración de procesos desde procesos.txt
- Responsabilidades:
  - Parsea los parámetros iniciales (PID, registros AX/BX/CX, quantum)
  - Carga las instrucciones específicas de cada proceso desde archivos individuales
  - Valida la integridad de los datos de entrada

2. Gestor de Instrucciones (instrucciones.cpp/.h)

- Función: Ejecuta las instrucciones de cada proceso
- Instrucciones soportadas:
  - ADD reg1, reg2/valor - Suma aritmética
  - SUB reg1, reg2/valor - Resta aritmética
  - MUL reg1, reg2/valor - Multiplicación
  - INC reg - Incremento en 1
  - JMP destino - Salto incondicional
  - NOP - No operación

3. Simulador Principal (simulador.cpp/.h)

- Función: Implementa el algoritmo Round Robin
- Funcionalidades:
  - Gestión de la cola de procesos
  - Cambio de contexto entre procesos
  - Control de quantum por proceso
  - Detección de terminación de procesos

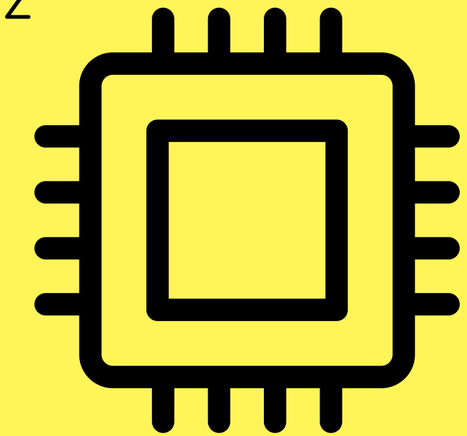
4. Estructura de Procesos (proceso.cpp/.h)

- Función: Define la estructura de datos de cada proceso
- Elementos:
  - PID (Process ID)
  - Program Counter (PC)
  - Registros (AX, BX, CX)
  - Quantum asignado
  - Estado actual (READY, RUNNING, WAITING, TERMINATED)
  - Vector de instrucciones

## Especificaciones del Sistema

Hardware de Prueba

- Procesador: 12th Gen Intel(R) Core(TM) i5-12500H @ 3.10 GHz
- Memoria RAM: 16 GB



## Requisitos

- Sistema operativo: Linux o Windows con WSL.
- Compilador: g++ versión 7 o superior.
- Herramientas: Terminal o consola para compilar y ejecutar.

# Análisis de Ejecución

**Procesos Cargados**

```
Se cargaron 3 procesos.
PID: 1, Quantum: 3, AX: 2, BX: 3, CX: 1, Instrucciones: 4
PID: 2, Quantum: 2, AX: 5, BX: 0, CX: 4, Instrucciones: 3
PID: 3, Quantum: 4, AX: 0, BX: 0, CX: 0, Instrucciones: 3
```

```
angel@ANGEL:~/Proyecto-1-Sistemas-Operativos$ ./procplanner
Se cargaron 3 procesos.
PID: 1, Quantum: 3, AX: 2, BX: 3, CX: 1, Instrucciones: 4
PID: 2, Quantum: 2, AX: 5, BX: 0, CX: 4, Instrucciones: 3
PID: 3, Quantum: 4, AX: 0, BX: 0, CX: 0, Instrucciones: 3

[Cambio de contexto]
Ejecutando Proceso 1 (Quantum = 3)
PC=0 | ADD AX, BX | RESULTADO = 5
PC=1 | INC AX | RESULTADO = 6
PC=2 | NOP | No hay resultado relevante
Proceso 1 pausa en PC=3

[Cambio de contexto]
Ejecutando Proceso 2 (Quantum = 2)
PC=0 | MUL AX, CX | RESULTADO = 20
PC=2 | JMP 2 | No hay resultado relevante
Proceso 2 pausa en PC=2

[Cambio de contexto]
Ejecutando Proceso 3 (Quantum = 4)
PC=0 | INC AX | RESULTADO = 1
PC=1 | NOP | No hay resultado relevante
PC=2 | SUB AX, 2 | RESULTADO = -1
Proceso 3 ha terminado.

[Cambio de contexto]
Ejecutando Proceso 1 (Quantum = 3)
PC=3 | SUB CX, 1 | No hay resultado relevante
Proceso 1 ha terminado.

[Cambio de contexto]
Ejecutando Proceso 2 (Quantum = 2)
PC=2 | ADD BX, 5 | RESULTADO = 5
Proceso 2 ha terminado.

Todos los procesos han terminado.
```

# Dificultades

La mayor dificultad fue sin duda la instrucción JMP, que cuando tenía como destino una instrucción anterior, terminamos en un ciclo infinito, esto nos tomó tiempo reconocerlo y saber como solucionarlo, se resolvió simplemente llevando un contador de instrucciones, y si se ve que se repite la instrucción mas de 10 veces, se acaba el proceso. Igualmente acoplarnos al C++ fue complejo al inicio, sin embargo con práctica e investigación logramos solucionarlo.

# Conclusión

La implementación del algoritmo de Round Robin demuestra ser una forma eficiente y equitativa de manejar procesos según su tiempo de ejecución. Por ello, concluimos este proyecto con un aprendizaje significativo sobre diversas piezas esenciales del funcionamiento de un procesador, como el ALU, el quantum y la asignación de proceso en general reforzando nuestra comprensión de la gestión y planificación de tareas en sistemas operativos.