<!--EAFIT-->

Simulación de cambio de contexto con planificación Round Robin {

<Por="Andres Toro, Fabian Buritica, Santiago Lafont"/>

}

Como funciona el codigo {

- El programa lee un archivo el cual contiene los procesos a ejecutar y sus registros. Luego lee las instrucciones del primer proceso mostrado y dependiendo su quantum ejecuta una cantidad n de instrucciones. Si el proceso no ha terminado guarda los valores de sus registros y hace un cambio de contexto, pasando a ejecutar el siguiente proceso. Esto se repite hasta que todos los procesos se hayan completado y ahí finaliza el programa.

}

procesos.txt

```
1   PID: 2, AX=1, BX=2, CX=3, Quantum=3
2   PID: 1, AX=0, BX=0, CX=0, Quantum=4
3
4
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Procesos cargados:
PID: 2, AX=1, BX=2, CX=3, Quantum=3, Estado=Listo
PID: 1, AX=0, BX=0, CX=0, Quantum=4, Estado=Listo

[Cambio de contexto]
Cargando estado de Proceso 2: PC=0, AX=1, BX=2, CX=3
Proceso 2 ejecutando instrucción 0: INC AX
Valores después de la instrucción: AX=2, BX=2, CX=3
Proceso 2 ejecutando instrucción 1: ADD BX, AX
Valores después de la instrucción: AX=2, BX=4, CX=3
Proceso 2 ejecutando instrucción 2: JMP 3
Valores después de la instrucción: AX=2, BX=4, CX=3

[Cambio de contexto]
Guardando estado de Proceso 2: PC=3, AX=2, BX=4, CX=3
Cargando estado de Proceso 1: PC=0, AX=0, BX=0, CX=0
Proceso 1 ejecutando instrucción 0: ADD AX, 5
Valores después de la instrucción: AX=5, BX=0, CX=0
Proceso 1 ejecutando instrucción 1: MUL BX, 2
Valores después de la instrucción: AX=5, BX=0, CX=0
Proceso 1 ejecutando instrucción 2: NOP
Valores después de la instrucción: AX=5, BX=0, CX=0
Proceso 1 ejecutando instrucción 3: INC CX
Valores después de la instrucción: AX=5, BX=0, CX=1
Proceso 1 ha terminado sus instrucciones.

[Cambio de contexto]
Cargando estado de Proceso 2: PC=3, AX=2, BX=4, CX=3
Proceso 2 ejecutando instrucción 3: NOP
Valores después de la instrucción: AX=2, BX=4, CX=3
Proceso 2 ha terminado sus instrucciones.
```

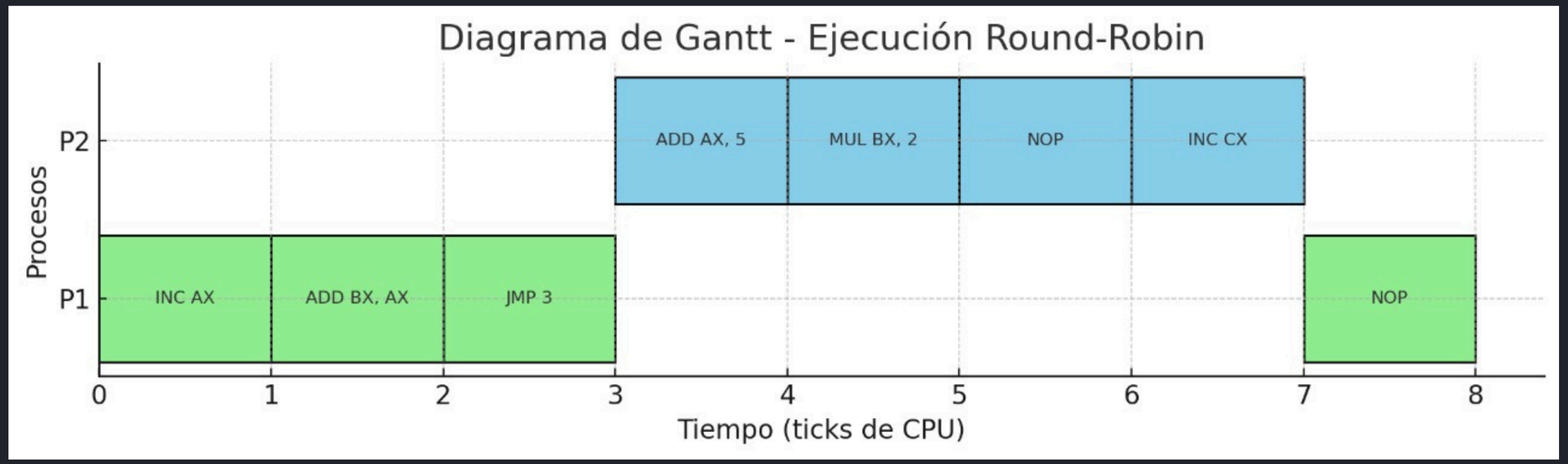
Diagrama de Gantt - Ejecución Round-Robin
Procesos
P2
P1
ADD AX, 5
MUL BX, 2
NOP
INC CX
INC AX
ADD BX, AX
JMP 3
NOP
0
1
2
3
4
5
6
7
8
Tiempo (ticks de CPU)

```
1   PID: 1, AX=0, BX=0, CX=0, Quantum=3
2   PID: 2, AX=1, BX=2, CX=3, Quantum=2
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Procesos cargados:
PID: 1, AX=0, BX=0, CX=0, Quantum=3, Estado=Listo
PID: 2, AX=1, BX=2, CX=3, Quantum=2, Estado=Listo

[Cambio de contexto]
Cargando estado de Proceso 1: PC=0, AX=0, BX=0, CX=0
Proceso 1 ejecutando instrucción 0: ADD AX, 5
Valores después de la instrucción: AX=5, BX=0, CX=0
Proceso 1 ejecutando instrucción 1: MUL BX, 2
Valores después de la instrucción: AX=5, BX=0, CX=0
Proceso 1 ejecutando instrucción 2: NOP
Valores después de la instrucción: AX=5, BX=0, CX=0

[Cambio de contexto]
Guardando estado de Proceso 1: PC=3, AX=5, BX=0, CX=0
Cargando estado de Proceso 2: PC=0, AX=1, BX=2, CX=3
Proceso 2 ejecutando instrucción 0: INC AX
Valores después de la instrucción: AX=2, BX=2, CX=3
Proceso 2 ejecutando instrucción 1: ADD BX, AX
Valores después de la instrucción: AX=2, BX=4, CX=3

[Cambio de contexto]
Guardando estado de Proceso 2: PC=2, AX=2, BX=4, CX=3
Cargando estado de Proceso 1: PC=3, AX=5, BX=0, CX=0
Proceso 1 ejecutando instrucción 3: INC CX
Valores después de la instrucción: AX=5, BX=0, CX=1
Proceso 1 ha terminado sus instrucciones.

[Cambio de contexto]
Cargando estado de Proceso 2: PC=2, AX=2, BX=4, CX=3
Proceso 2 ejecutando instrucción 2: JMP 3
Valores después de la instrucción: AX=2, BX=4, CX=3
Proceso 2 ejecutando instrucción 3: NOP
Valores después de la instrucción: AX=2, BX=4, CX=3
Proceso 2 ha terminado sus instrucciones.
```

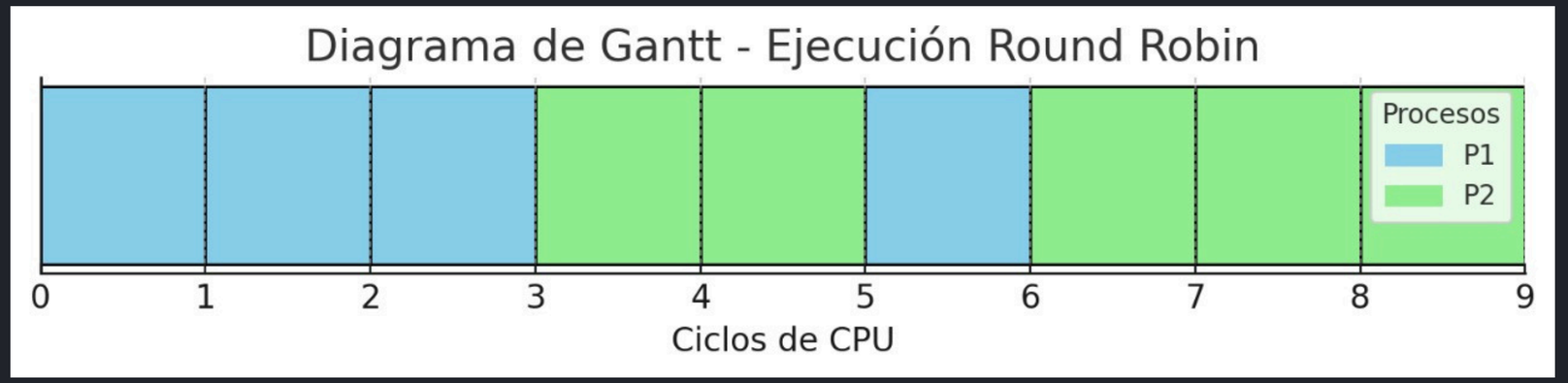
Diagrama de Gantt - Ejecución Round Robin
Procesos
P1
P2
0
1
2
3
4
5
6
7
8
9
Ciclos de CPU

Especificaciones del equipo {

Ram

40GB

Procesador

12th Gen Intel(R) Core(TM) i5-12450H (2.50 GHz)

Version del compilador

gcc (Ubuntu 13.3.0-6ubuntu2~24.04) 13.3.0

}

conclusiones

- si un proceso tiene quantum mayor, hará más trabajo seguido y sufrirá menos interrupciones; quantum pequeño aumenta el número de cambios de contexto pero reduce latencia para procesos que esperan.

- El tamaño del quantum importa
- Quantum grande → menos cambios de contexto, pero los demás esperan más.
- Quantum pequeño → más justo, pero más cambios de contexto.

- El orden de llegada influye
- El que empieza primero puede tener ventaja en el tiempo de respuesta.