

LABORATORIO DE SISTEMAS OPERATIVOS

PERÍODO ACADÉMICO: 2025 – B

EQUIPO:

PROFESOR: Ing. Marcela Saavedra MSc.

TIPO DE INSTRUMENTO: Guía de Laboratorio

TEMA: INTERBLOQUEOS O DEADLOCKS

ÍNDICE DE CONTENIDOS

1. OBJETIVOS	2
2. MARCO TEÓRICO.....	2
Detección de interbloqueo	2
3. PROCEDIMIENTO	3
3.1 Instalación de CPU-OS Simulator	3
3.2 Gráfico de asignación.....	3
3.3 Creación de un deadlock por medio de CPU-OS Simulator.....	3
3.4 Métodos de recuperación de una condición de bloqueo después de que se produzca..	11
3.5 Métodos de prevención de una condición de bloqueo	16
4. INFORME.....	24

ÍNDICE DE FIGURAS

Figura 1. Asignación de recursos a procesos	3
Figura 2. Ventana CPU Simulator.....	4
Figura 3. Compilador	4
Figura 4. Guardar los procesos	5
Figura 5. Compilación de procesos	5
Figura 6. Carga de procesos en memoria.....	6
Figura 7. Ventana de configuración	6
Figura 8. Procesos a estado listo	7
Figura 9. Uso de técnica RR.....	7
Figura 10. Simulación rápida	8
Figura 11. Ventana de recursos	8
Figura 12. Inicialización de simulación	9
Figura 13. Opción para mostrar estado de procesos	9
Figura 14. Liberación de recursos	11
Figura 15. Suspensión de procesos	14
Figura 16. Continuación de simulación.....	14
Figura 17. Técnicas de prevención de deadlocks	17

1. OBJETIVOS

- 1.1. Utilizar el simulador CPU Simulator para crear condiciones de bloqueo.
- 1.2. Implementar métodos para prevenir condiciones desbloqueo.

2. MARCO TEÓRICO

El interbloqueo es una situación que ocurre en el sistema operativo cuando cualquier proceso entra en un estado de espera porque otro proceso en espera está reteniendo el recurso demandado.

El interbloqueo es un problema común en el procesamiento múltiple donde varios procesos comparten un tipo específico de recurso mutuamente excluyente conocido como bloqueo suave o software.

Detección de interbloqueo

El programador de recursos puede detectar una ocurrencia de interbloqueo. Un programador de recursos ayuda al sistema operativo a realizar un seguimiento de todos los recursos que se asignan a diferentes procesos. Por lo tanto, cuando se detecta un punto muerto, se puede resolver utilizando los métodos que se indican a continuación:

Prevención de bloqueos:

Es importante evitar un punto muerto antes de que pueda ocurrir. El sistema verifica cada transacción antes de que se ejecute para asegurarse de que no conduzca a las situaciones de punto muerto. De tal manera que, incluso un pequeño cambio que puede conducir a un *deadlock* en el futuro se permite que se ejecute.

Es un conjunto de métodos para garantizar que al menos una de las condiciones no pueda mantenerse.

Ninguna acción preventiva:

Sin preferencia: un recurso puede ser liberado solo voluntariamente por el proceso que lo retiene después de que ese proceso haya terminado su tarea.

3. PROCEDIMIENTO

3.1 Instalación de CPU-OS Simulator

Descargar e instalar el simulador de:

<https://teach-sim.com/downloads/>

3.2 Gráfico de asignación

Problema: Se están ejecutando cuatro procesos. Se llaman P1 a P4. También hay cuatro recursos disponibles (sólo una instancia de cada uno). Se denominan R0 a R3. En algún momento de su existencia, cada proceso asigna un recurso diferente para su uso y lo mantiene para sí mismo para siempre. Posteriormente cada uno de los procesos solicita otro de los cuatro recursos.

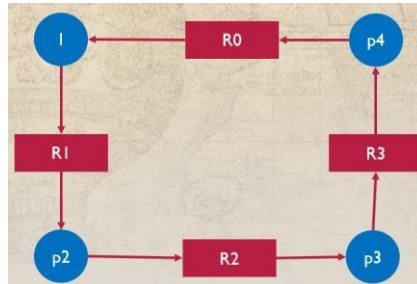


Figura 1. Asignación de recursos a procesos

3.3 Creación de un deadlock por medio de CPU-OS Simulator.

Paso 1: Abrir CPU-OS simulator.

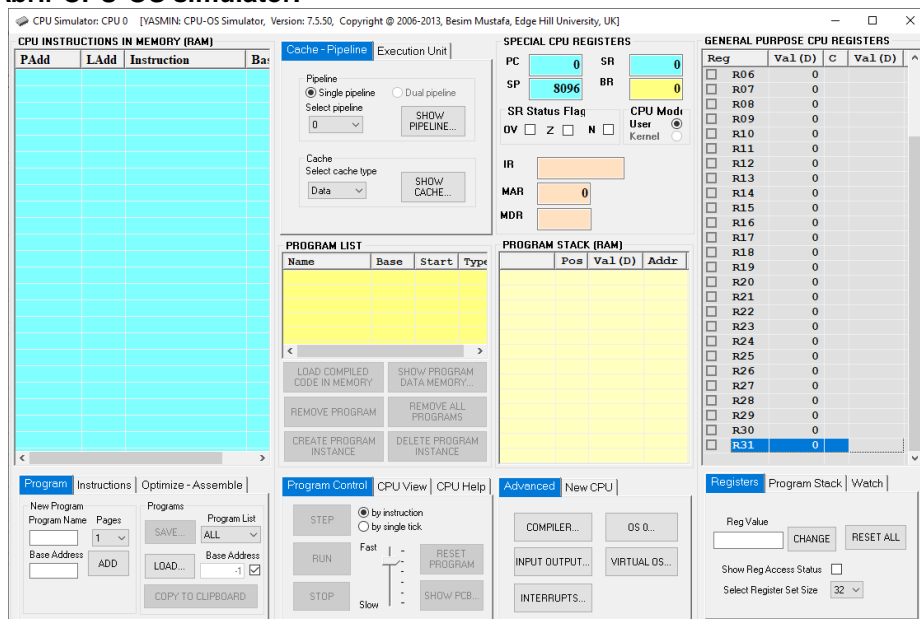


Figura 2. Ventana CPU Simulator

Paso 2: elegir la opción “compiler”, esto deberá abrir una ventana emergente.

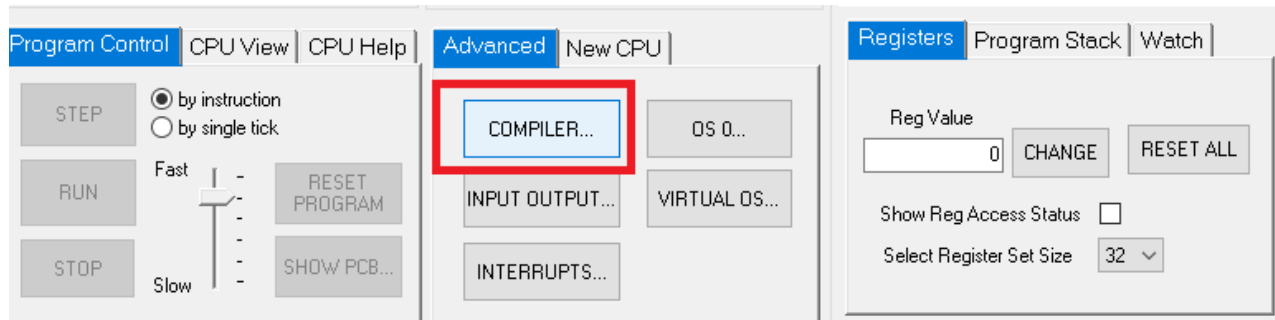


Figura 3. Compilador

Paso 3. Crear 4 procesos con recursos asignados y en espera de otro.

Colocar el código para cada proceso como el ejemplo de la figura en el Program Source y guardar como P.txt

P1.txt	R0 → P1 → R1
P2.txt	R1 → P2 → R2
P3.txt	R2 → P3 → R3
P4.txt	R3 → P4 → R0

```

program DeadlockP4
resource(0, allocate)
wait(3)
resource(1, allocate)
for n = 1 to 20
next
end
    
```

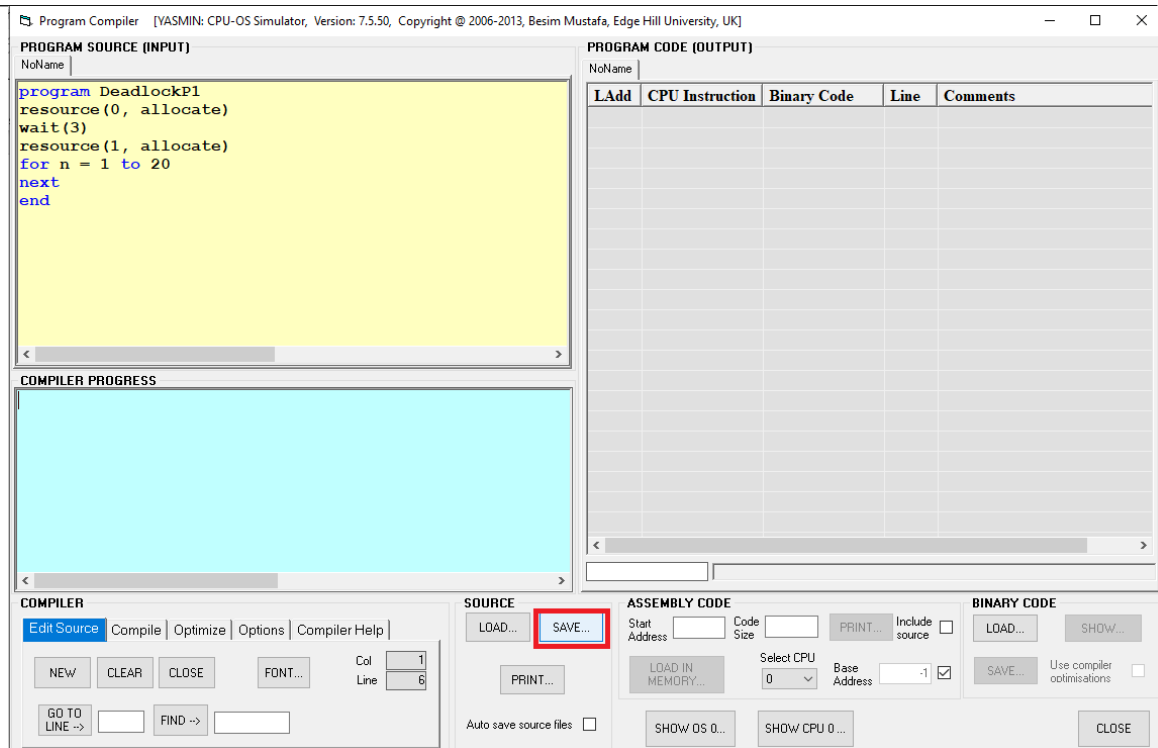


Figura 4. Guardar los procesos

Paso4. Compilar todos los programas creados

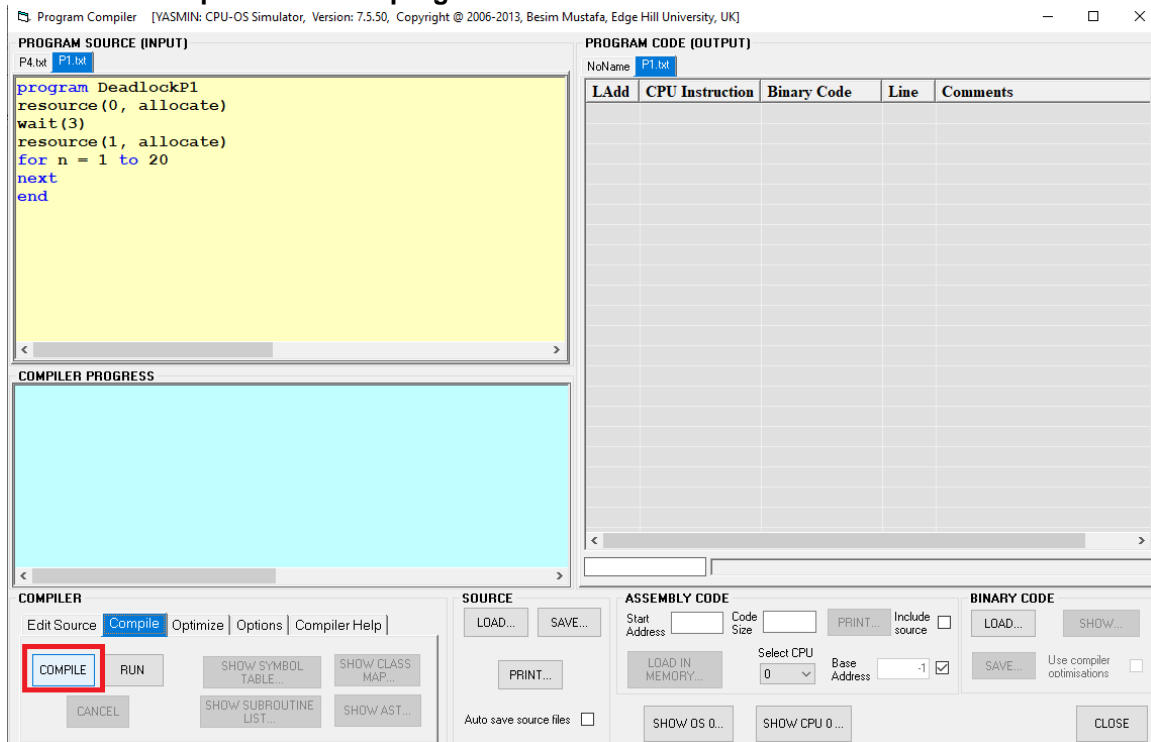


Figura 5. Compilación de procesos

Paso 5: Cargar los programas a memoria.

Escoger el programa y seleccionar la opción “load in memory”.

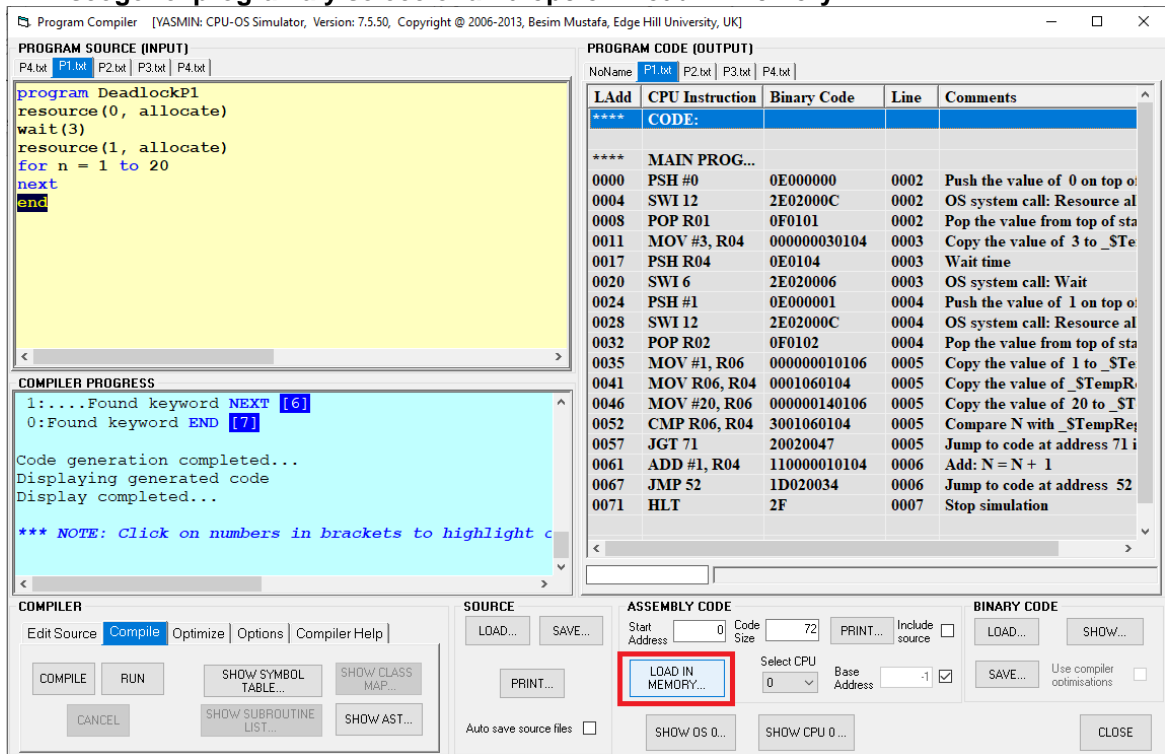


Figura 6. Carga de procesos en memoria

Paso 6: Seleccionar la opción “OS Simulator”.

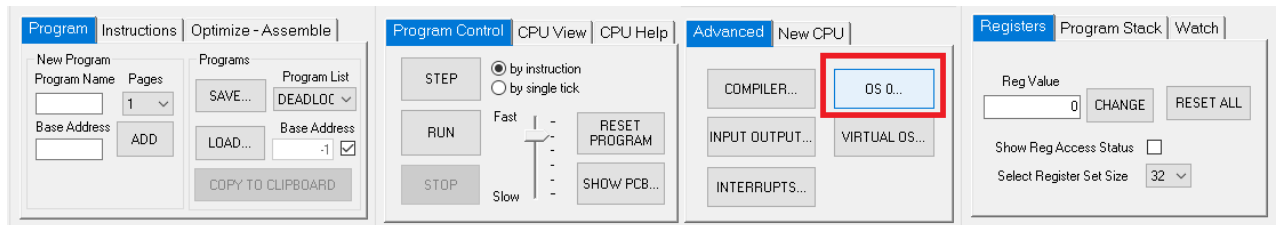


Figura 7. Ventana de configuración

Paso 7: Crear una instancia de cada programa.

Doble clic sobre cada nombre del programa en “Program list”, cuando se cree la instancia de los programas aparecerán en “READY PROCESSES (Ready Queue)”.

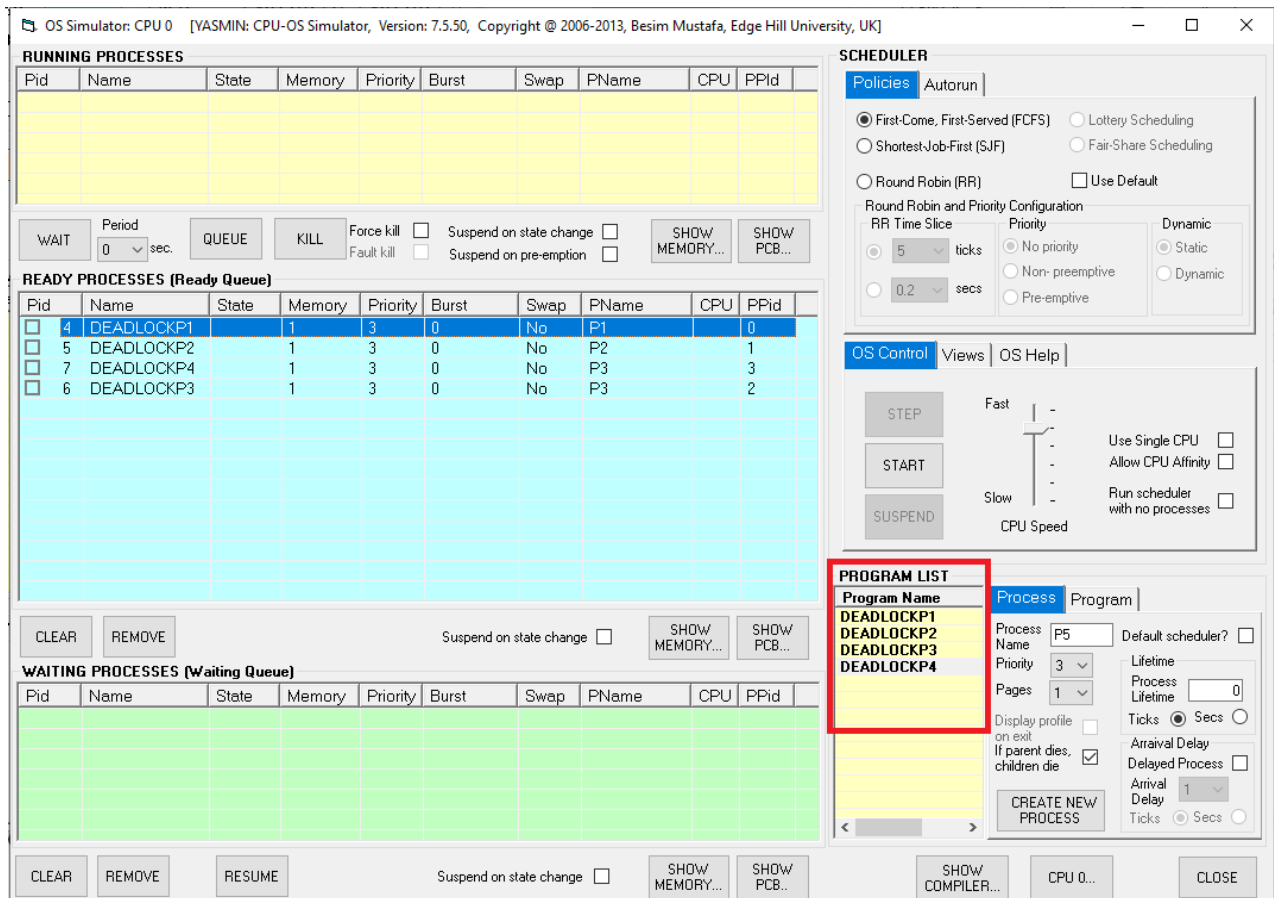


Figura 8. Procesos a estado listo

Paso 8: Seleccionar la técnica Round Robin en SCHEDULER → Policies.

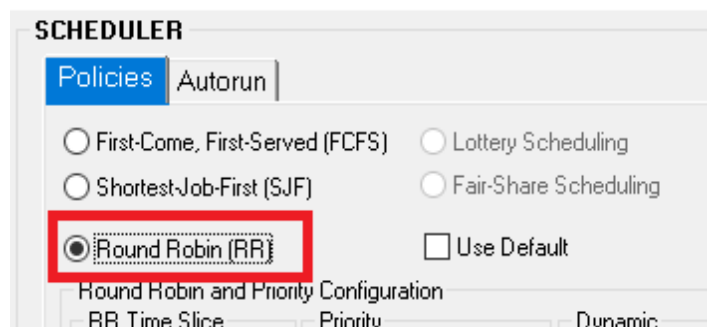


Figura 9. Uso de técnica RR

Paso 9: Colocar "CPU speed" a su máxima velocidad en OS Control.

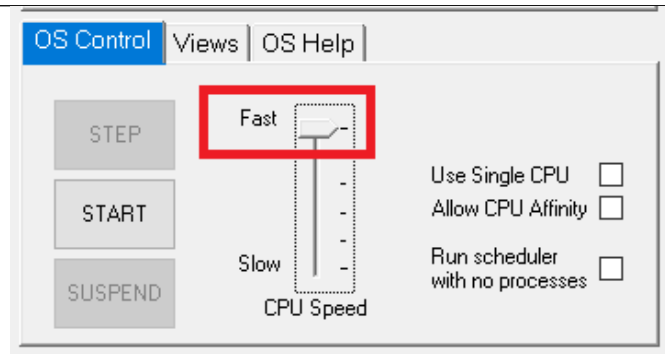


Figura 10. Simulación rápida

Paso 10: En la pestaña “Views” seleccionar “View resources”.
En la ventana emergente marcar Stay on top.



Figura 11. Ventana de recursos

Paso 11: Dentro de la ventana “OS control” seleccionar “Start” para comenzar la simulación y observar cómo cambian los estados de los recursos.

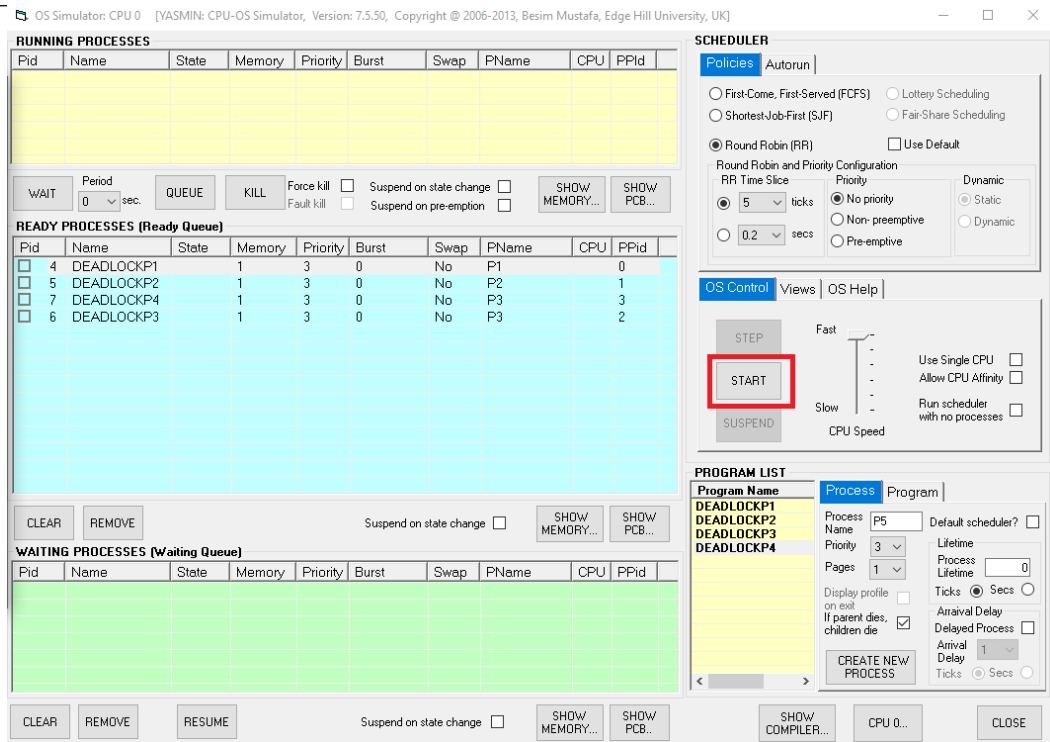


Figura 12. Inicialización de simulación

Mostrar el resultado que muestra la opción **SHOW DEADLOCKED PROCESSES** dentro de la vista de recursos.

Explicar detalladamente lo observado.

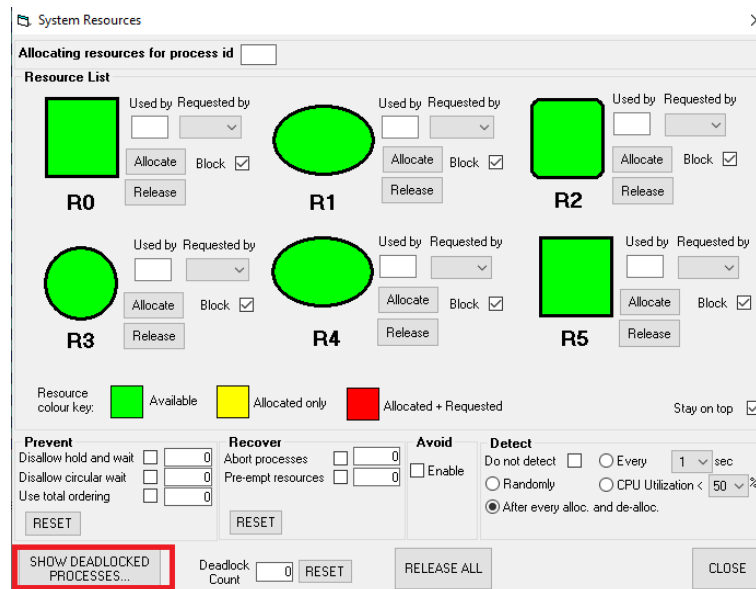
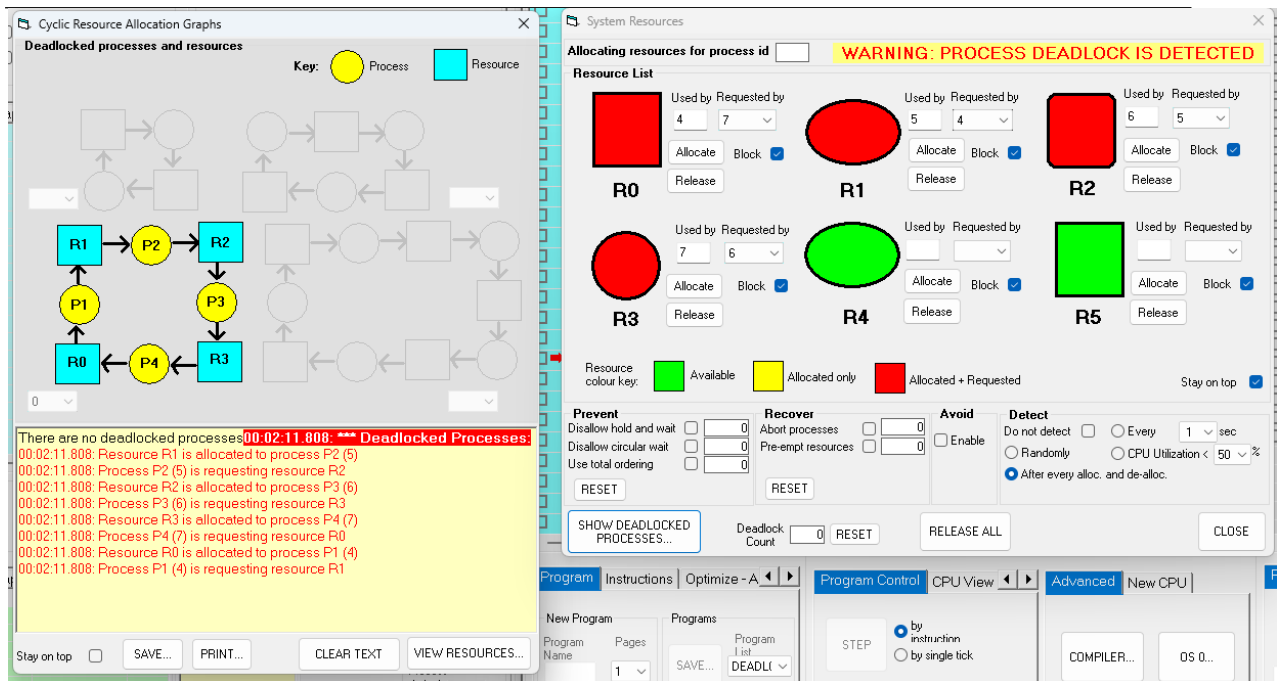


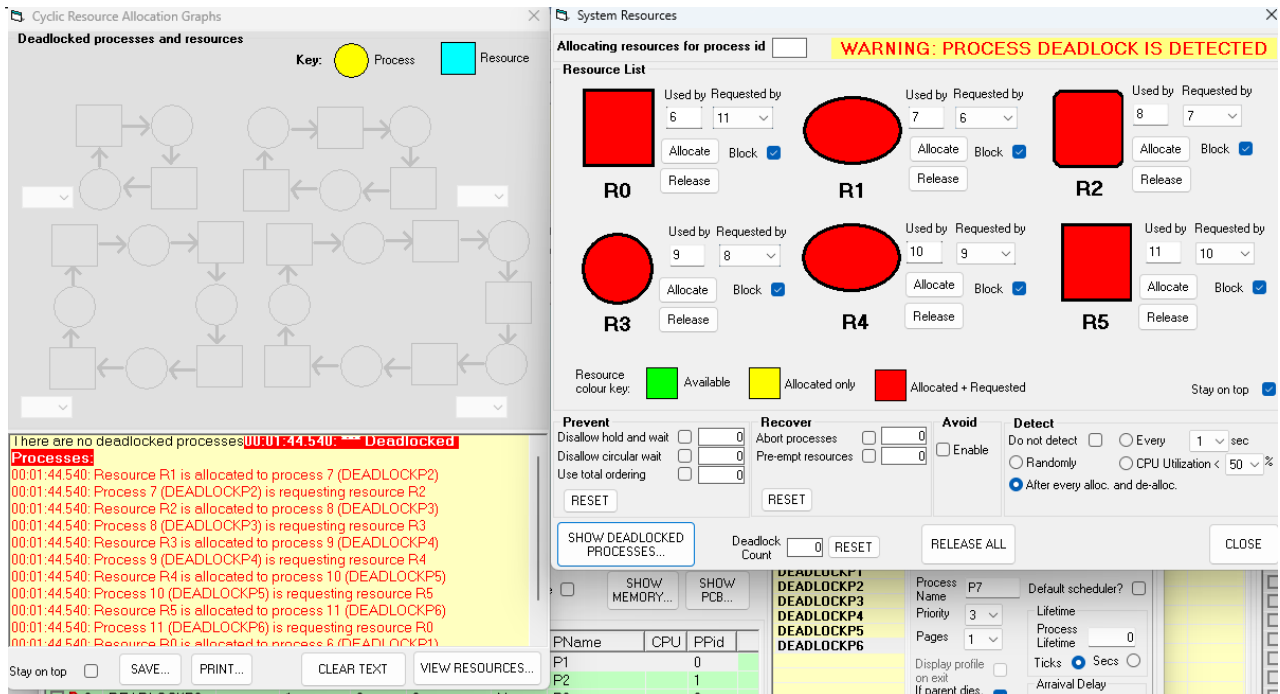
Figura 13. Opción para mostrar estado de procesos



En la imagen se puede notar que al momento de ejecutar los procesos, existe una espera circular, ya que existe un “bucle” en el que cada proceso espera un recurso “R” de otro

Repetir el procedimiento con 6 procesos y 6 recursos.

P1: R0 → R1
P2: R1 → R2
P3: R2 → R3
P4: R3 → R4
P5: R4 → R5
P6: R5 → R0



Cyclic Resource Allocation Graphs

Deadlocked processes and resources

Key: ● Process ■ Resource

There are no deadlocked processes.

Processes:

00:01:44.540: Resource R1 is allocated to process 7 (DEADLOCKP2)
 00:01:44.540: Process 7 (DEADLOCKP2) is requesting resource R2
 00:01:44.540: Resource R2 is allocated to process 8 (DEADLOCKP3)
 00:01:44.540: Process 8 (DEADLOCKP3) is requesting resource R3
 00:01:44.540: Resource R3 is allocated to process 9 (DEADLOCKP4)
 00:01:44.540: Process 9 (DEADLOCKP4) is requesting resource R4
 00:01:44.540: Resource R4 is allocated to process 10 (DEADLOCKP5)
 00:01:44.540: Process 10 (DEADLOCKP5) is requesting resource R5
 00:01:44.540: Resource R5 is allocated to process 11 (DEADLOCKP6)
 00:01:44.540: Process 11 (DEADLOCKP6) is requesting resource R0
 00:01:44.540: Resource R0 is allocated to process 6 (DEADLOCKP1)

System Resources

Allocating resources for process id

WARNING: PROCESS DEADLOCK IS DETECTED

Resource List

Resource	Used by	Requested by	Allocate	Block	Release
R0	6	11	<input type="button" value="Allocate"/>	<input checked="" type="checkbox"/>	<input type="button" value="Release"/>
R1	7	6	<input type="button" value="Allocate"/>	<input checked="" type="checkbox"/>	<input type="button" value="Release"/>
R2	8	7	<input type="button" value="Allocate"/>	<input checked="" type="checkbox"/>	<input type="button" value="Release"/>
R3	9	8	<input type="button" value="Allocate"/>	<input checked="" type="checkbox"/>	<input type="button" value="Release"/>
R4	10	9	<input type="button" value="Allocate"/>	<input checked="" type="checkbox"/>	<input type="button" value="Release"/>
R5	11	10	<input type="button" value="Allocate"/>	<input checked="" type="checkbox"/>	<input type="button" value="Release"/>

Resource colour key: ■ Available ■ Allocated only ■ Allocated + Requested

Prevent
 Disallow hold and wait ☐ 0
 Disallow circular wait ☐ 0
 Use total ordering ☐ 0

Recover
 Abort processes ☐ 0
 Pre-empt resources ☐ 0

Avoid
☐ Enable

Detect
 Do not detect ☐ Every 1 sec
☒ Randomly ☐ CPU Utilization < 50 %
☒ After every alloc. and de-alloc.

RESET

SHOW DEADLOCKED PROCESSES...

Deadlock Count: 0 RESET

RELEASE ALL

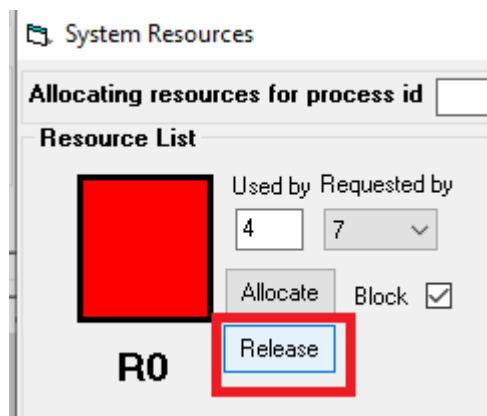
CLOSE

Stay on top ☐ SAVE... PRINT... CLEAR TEXT VIEW RESOURCES...

En el caso de los 6 procesos, aunque haya una ausencia de la imagen de los interbloqueos, podemos notar en el texto la espera circular que se notaba en el anterior ejemplo, solo que en este caso, existen mas procesos y recursos, podemos notar que hay un bloqueo en los 5 recursos, ya que como se mencionó antes, existe un bucle de espera entre cada uno

3.4 Métodos de recuperación de una condición de bloqueo después de que se produzca

Paso 1: En la ventana de Recursos del Sistema liberar el recurso R0.



System Resources

Allocating resources for process id

Resource List

Resource	Used by	Requested by	Allocate	Block	Release
R0	4	7	<input type="button" value="Allocate"/>	<input checked="" type="checkbox"/>	<input type="button" value="Release"/>

Figura 14. Liberación de recursos

Paso 2: Observar durante 5 minutos y documentar lo que ocurre.

Responder:

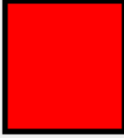
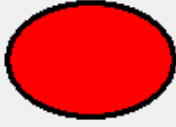

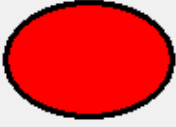


¿Se ha resuelto la condición de bloqueo? ¿Por qué?




System Resources

Allocating resources for process id

WARNING: PROCESS DEADLOCK IS DETECTED

Resource List

Resource	Used by	Requested by	Used by	Requested by	Used by	Requested by
	6	11		7	8	7
R0	<input type="text"/>	<input type="text"/>	R1	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="button" value="Allocate"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="button" value="Allocate"/>	<input checked="" type="checkbox"/>	<input type="button" value="Allocate"/>	<input type="checkbox"/>
<input type="button" value="Release"/>			<input type="button" value="Release"/>		<input type="button" value="Release"/>	
	9	8		10	9	10
R3	<input type="text"/>	<input type="text"/>	R4	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="button" value="Allocate"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="button" value="Allocate"/>	<input checked="" type="checkbox"/>	<input type="button" value="Allocate"/>	<input type="checkbox"/>
<input type="button" value="Release"/>			<input type="button" value="Release"/>		<input type="button" value="Release"/>	
	8	7		11	10	11
R2	<input type="text"/>	<input type="text"/>	R5	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="button" value="Allocate"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="button" value="Allocate"/>	<input type="checkbox"/>	<input type="button" value="Allocate"/>	<input type="checkbox"/>
<input type="button" value="Release"/>			<input type="button" value="Release"/>		<input type="button" value="Release"/>	

Resource colour key:
 Available
 Allocated only
 Allocated + Requested
Stay on top ☐

Prevent
Disallow hold and wait ☐
Disallow circular wait ☐
Use total ordering ☐

Recover
Abort processes ☐
Pre-empt resources ☐

Avoid
☐ Enable

Detect
Do not detect ☐
☐ Every sec
☐ Randomly
☐ CPU Utilization < %
☒ After every alloc. and de-alloc.

Deadlock Count

System Resources

Allocating resources for process id

Resource List

Resource	Used by	Requested by	Allocate	Block	Release
R0	<input type="text"/>	<input type="text"/>	<input type="button" value="Allocate"/>	<input type="checkbox"/>	<input type="button" value="Release"/>
R1	<input type="text"/>	<input type="text"/>	<input type="button" value="Allocate"/>	<input checked="" type="checkbox"/>	<input type="button" value="Release"/>
R2	<input type="text"/>	<input type="text"/>	<input type="button" value="Allocate"/>	<input type="checkbox"/>	<input type="button" value="Release"/>
R3	<input type="text"/>	<input type="text"/>	<input type="button" value="Allocate"/>	<input type="checkbox"/>	<input type="button" value="Release"/>
R4	<input type="text"/>	<input type="text"/>	<input type="button" value="Allocate"/>	<input checked="" type="checkbox"/>	<input type="button" value="Release"/>
R5	<input type="text"/>	<input type="text"/>	<input type="button" value="Allocate"/>	<input type="checkbox"/>	<input type="button" value="Release"/>

Resource colour key: ■ Available ■ Allocated only ■ Allocated + Requested

Stay on top ☐

Prevent

Disallow hold and wait ☐

Disallow circular wait ☐

Use total ordering ☐

Recover

Abort processes ☐

Pre-empt resources ☐

Avoid

☐ Enable

Detect

Do not detect ☐ ☐ Every sec

☐ Randomly ☐ CPU Utilization < %

☒ After every alloc. and de-alloc.

Se puede notar que los procesos se van liberando, pasando de rojo a amarillo y después a blanco, en la otra pantalla podemos notar que todos los procesos se han removido. Primero empieza liberando el recurso 0, lo cual causa una reacción en cadena que rompe la espera circular y por ende quita el deadlock

Paso 3: Repetir la condición de bloqueo.

Paso 4: En la ventana del OS Simulator, seleccionar un proceso de la cola de espera, luego seleccionar SUSPENDER y luego REMOVE.

OS Simulator: CPU 0 [YASMIN: CPU-OS Simulator, Version: 7.5.50, Copyright © 2006-2013, Besim Mustafa, Edge Hill University, UK]

RUNNING PROCESSES

Pid	Name	State	Memory	Priority	Burst	Swap	PName	CPU	PPid

WAIT 0 sec. Force kill ☐ Suspend on state change ☐ SHOW MEMORY... SHOW PCB...

READY PROCESSES (Ready Queue)

Pid	Name	State	Memory	Priority	Burst	Swap	PName	CPU	PPid

Suspend on state change ☐ SHOW MEMORY... SHOW PCB...

WAITING PROCESSES (Waiting Queue)

Pid	Name	State	Memory	Priority	Burst	Swap	PName	CPU	PPid
<input checked="" type="checkbox"/> 4	DEADLOCKP1		1	3	2	No	P1	0	
<input type="checkbox"/> 6	DEADLOCKP3		1	3	2	No	P2	2	
<input type="checkbox"/> 5	DEADLOCKP2		1	3	2	No	P2	1	
<input type="checkbox"/> 7	DEADLOCKP4		1	3	2	No	P4	3	

Suspend on state change ☐ SHOW MEMORY... SHOW PCB...

SCHEDULER

Policies

☐ First-Come, First-Served (FCFS) ☐ Lottery Scheduling
☐ Shortest-Job-First (SJF) ☐ Fair-Share Scheduling
☒ Round Robin (RR) ☐ Use Default

Round Robin and Priority Configuration
 RR Time Slice: ☐ 5 ticks ☐ 0.2 secs
 Priority: ☒ No priority ☐ Non-preemptive ☐ Pre-emptive
 Dynamic: ☒ Static ☐ Dynamic

OS Control Views OS Help

CPU Speed: Fast Slow

Use Single CPU ☐
 Allow CPU Affinity ☐
 Run scheduler with no processes ☐

PROGRAM LIST

Program Name: DEADLOCKP1, DEADLOCKP2, DEADLOCKP3, DEADLOCKP4

Process Name: P5, Priority: 3, Pages: 1

Display profile on exit ☐
 If parent dies, children die ☒

CREATE NEW PROCESS

Lifetime: Process Lifetime: 0, Ticks: ☒ Secs ☐ Arrival Delay: 1, Delayed Process: ☐

SHOW COMPILER... CPU 0... CLOSE

Figura 15. Suspensión de procesos

Paso 5: Seleccionar RESUME para continuar con el proceso.

OS Control Views OS Help

CPU Speed: Fast Slow

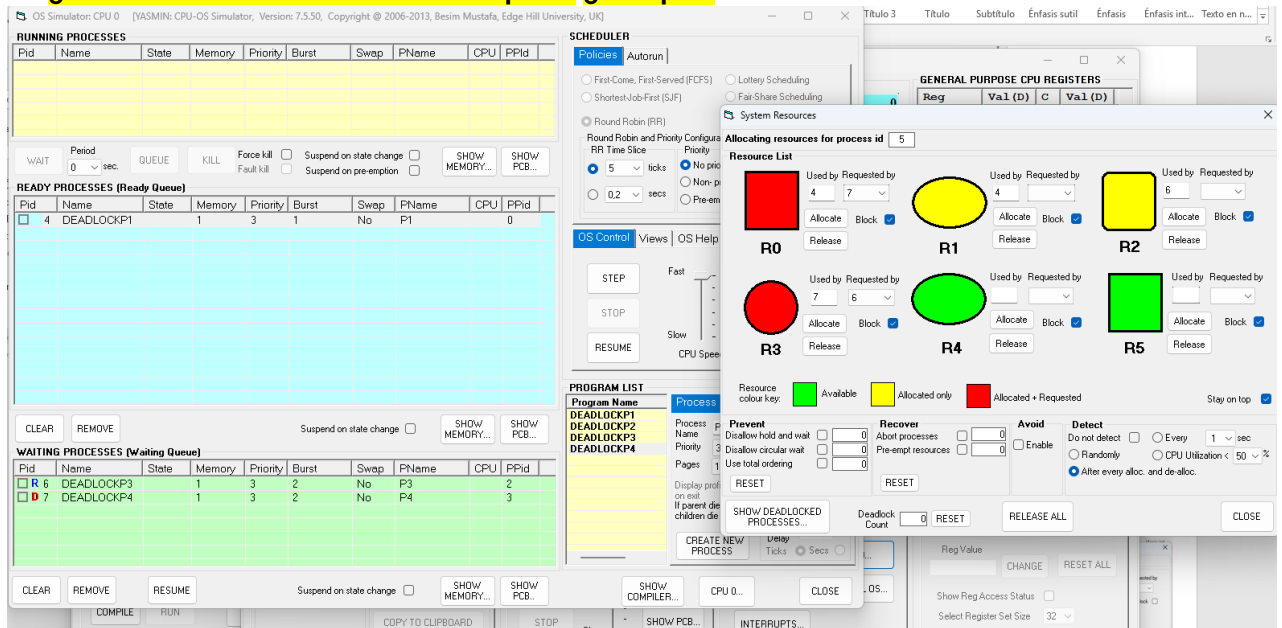
Use Single CPU ☐
 Allow CPU Affinity ☐
 Run scheduler with no processes ☐

Figura 16. Continuación de simulación

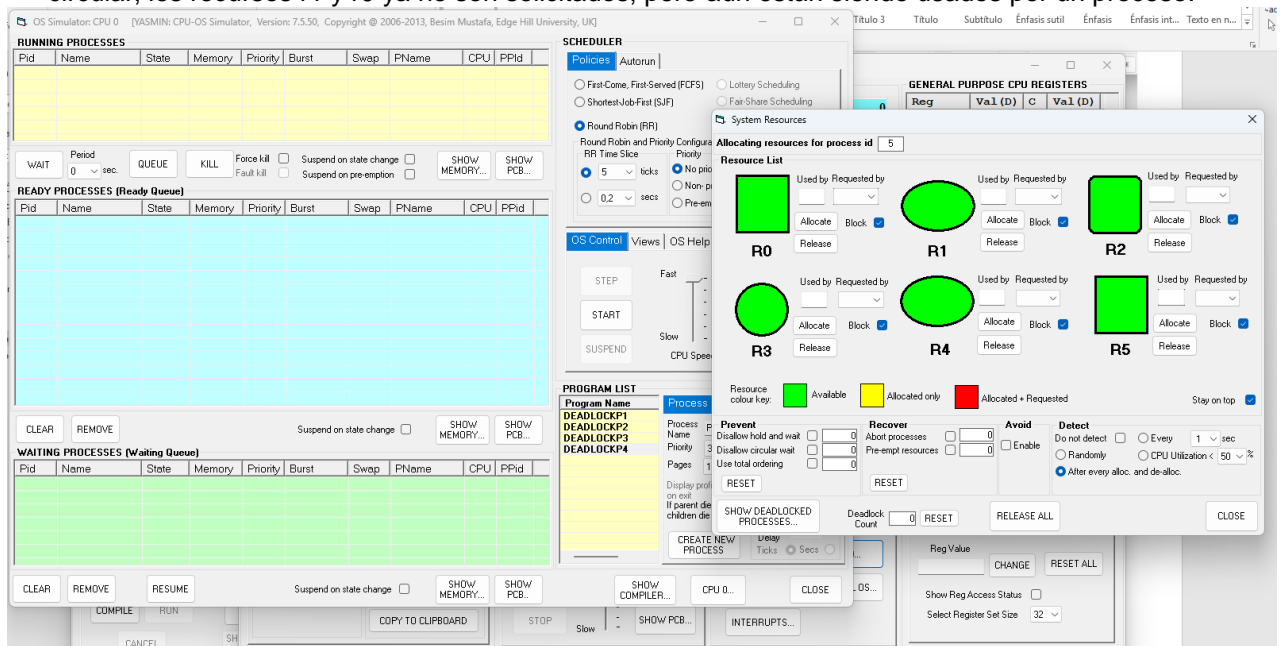
Paso 6: Observar durante 5 minutos y documentar lo que ocurre.

Responder:

¿Se ha resuelto la condición de bloqueo? ¿Por qué?

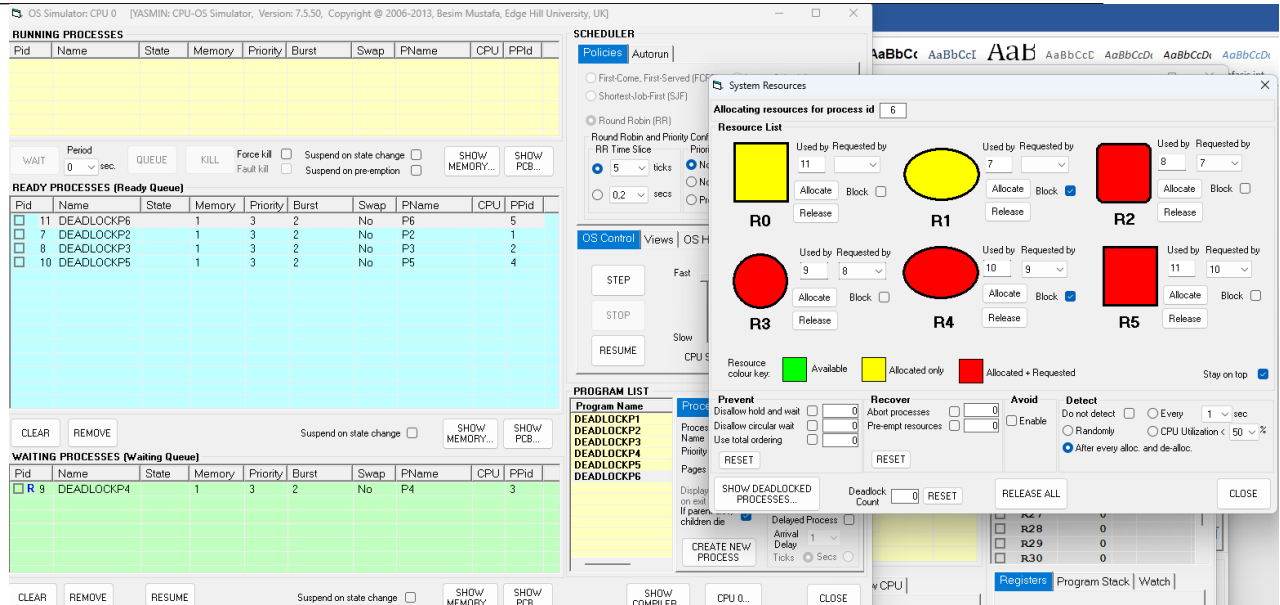


Al momento de remover y suspender el proceso se elimina el deadlock, ya que se rompió la espera circular, los recursos r1 y r0 ya no son solicitados, pero aún están siendo usados por un proceso.



Al reanudar los procesos notamos que ocurre la misma reacción en cadena vista en el ejemplo con los 6 procesos y recursos

Repetir el procedimiento con 6 procesos y 6 recursos.



OS Simulator: CPU 0 [YASMIN: CPU-OS Simulator, Version: 7.5.50, Copyright © 2006-2013, Besim Mustafa, Edge Hill University, UK]

RUNNING PROCESSES

Pid	Name	State	Memory	Priority	Burst	Swap	PName	CPU	PPid
11	DEADLOCKP6	1	3	2	No	P6	5		
7	DEADLOCKP2	1	3	2	No	P2	1		
8	DEADLOCKP3	1	3	2	No	P3	2		
10	DEADLOCKP5	1	3	2	No	P5	4		

READY PROCESSES (Ready Queue)

Pid	Name	State	Memory	Priority	Burst	Swap	PName	CPU	PPid
11	DEADLOCKP6	1	3	2	No	P6	5		
7	DEADLOCKP2	1	3	2	No	P2	1		
8	DEADLOCKP3	1	3	2	No	P3	2		
10	DEADLOCKP5	1	3	2	No	P5	4		

WAITING PROCESSES (Waiting Queue)

Pid	Name	State	Memory	Priority	Burst	Swap	PName	CPU	PPid
9	DEADLOCKP4	1	3	2	No	P4	3		

SCHEDULER

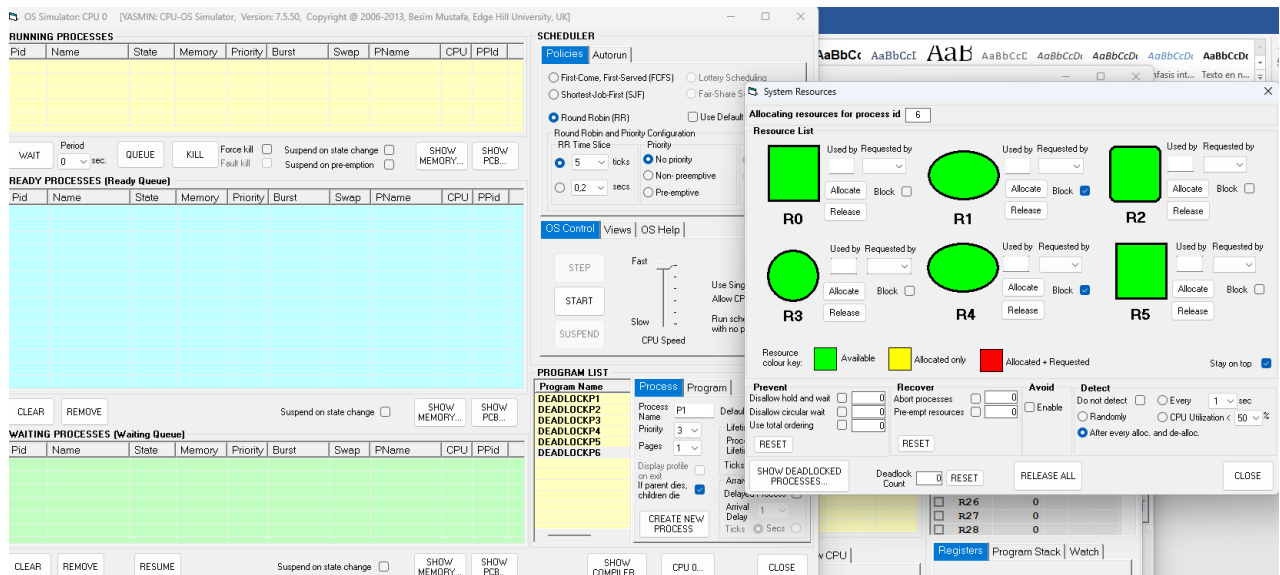
Resource List

Resource	Used by	Requested by	Allocated	Block	Release
R0	11	7	7		
R1	7	8	8		
R2	8	10	10		
R3	10	11	11		
R4	11	7	7		
R5	7	8	8		

PROGRAM LIST

Program Name	Process	Program
DEADLOCKP1	P1	Default
DEADLOCKP2	P2	Priority
DEADLOCKP3	P3	Priority
DEADLOCKP4	P4	Priority
DEADLOCKP5	P5	Priority
DEADLOCKP6	P6	Priority

Al momento de remover y suspender el proceso se elimina el deadlock, ya que se rompió la espera circular, los recursos r1 y r0 ya no son solicitados, pero aún están siendo usados por un proceso.



OS Simulator: CPU 0 [YASMIN: CPU-OS Simulator, Version: 7.5.50, Copyright © 2006-2013, Besim Mustafa, Edge Hill University, UK]

RUNNING PROCESSES

Pid	Name	State	Memory	Priority	Burst	Swap	PName	CPU	PPid
-----	------	-------	--------	----------	-------	------	-------	-----	------

READY PROCESSES (Ready Queue)

Pid	Name	State	Memory	Priority	Burst	Swap	PName	CPU	PPid
-----	------	-------	--------	----------	-------	------	-------	-----	------

WAITING PROCESSES (Waiting Queue)

Pid	Name	State	Memory	Priority	Burst	Swap	PName	CPU	PPid
-----	------	-------	--------	----------	-------	------	-------	-----	------

SCHEDULER

Resource List

Resource	Used by	Requested by	Allocated	Block	Release
R0	11	7	7		
R1	7	8	8		
R2	8	10	10		
R3	10	11	11		
R4	11	7	7		
R5	7	8	8		

PROGRAM LIST

Program Name	Process	Program
DEADLOCKP1	P1	Default
DEADLOCKP2	P2	Priority
DEADLOCKP3	P3	Priority
DEADLOCKP4	P4	Priority
DEADLOCKP5	P5	Priority
DEADLOCKP6	P6	Priority

Finalmente, cuando reanudamos el proceso notamos que se liberan totalmente todos los recursos, lo que quiere decir que se ha resuelto la condición de bloqueo, porque se ha eliminado un proceso que mantenía la espera circular, en este caso p1, eso causa que R0 deje de ser solicitado, entonces P6 ya no se queda esperando a dicho recurso, causando la misma reacción en cadena que el anterior ejercicio.

3.5 Métodos de prevención de una condición de bloqueo

Paso 1: Repetir el proceso para condición de bloqueo de 4 y 6 procesos, pero antes de seleccionar START habilitar la casilla “Disallow hold and wait”.

System Resources

Allocating resources for process id

Resource List

Resource	Used by	Requested by	Allocate	Block	Release
R0	<input type="text"/>	<input type="text"/>	<input type="button" value="Allocate"/>	<input checked="" type="checkbox"/>	<input type="button" value="Release"/>
R1	<input type="text"/>	<input type="text"/>	<input type="button" value="Allocate"/>	<input checked="" type="checkbox"/>	<input type="button" value="Release"/>
R2	<input type="text"/>	<input type="text"/>	<input type="button" value="Allocate"/>	<input checked="" type="checkbox"/>	<input type="button" value="Release"/>
R3	<input type="text"/>	<input type="text"/>	<input type="button" value="Allocate"/>	<input checked="" type="checkbox"/>	<input type="button" value="Release"/>
R4	<input type="text"/>	<input type="text"/>	<input type="button" value="Allocate"/>	<input checked="" type="checkbox"/>	<input type="button" value="Release"/>
R5	<input type="text"/>	<input type="text"/>	<input type="button" value="Allocate"/>	<input checked="" type="checkbox"/>	<input type="button" value="Release"/>

Resource colour key: ■ Available ■ Allocated only ■ Allocated + Requested Stay on top ☐

Prevent
 Disallow hold and wait ☒
 Disallow circular wait ☐
 Use total ordering ☐

Recover
 Abort processes ☐
 Pre-empt resources ☐

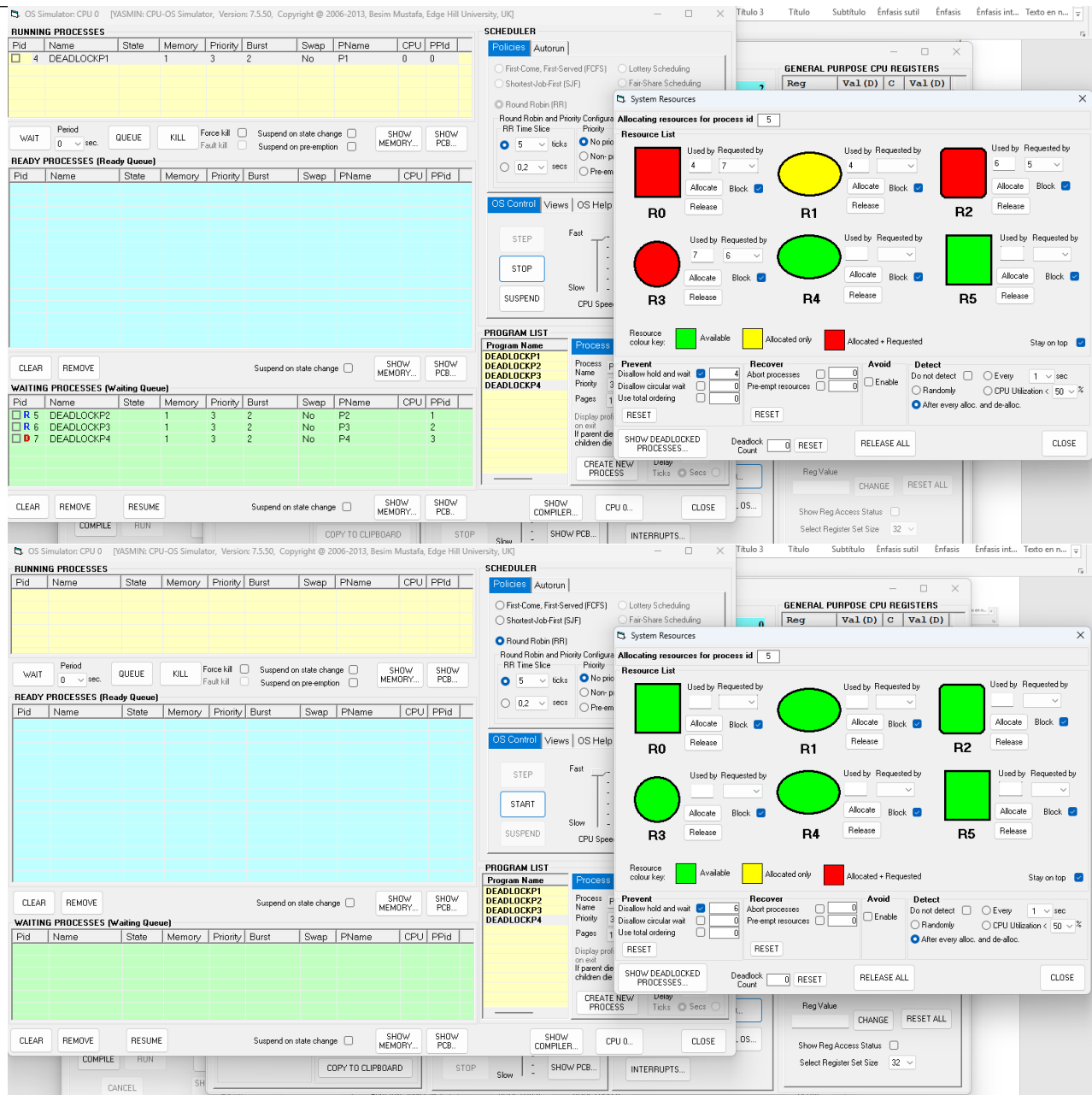
Avoid
☐ Enable

Detect
 Do not detect ☐ ☐ Every sec
☐ Randomly ☐ CPU Utilization < %
☒ After every alloc. and de-alloc.

Deadlock Count

Figura 17. Técnicas de prevención de deadlocks

procesos:



The screenshot displays the OS Simulator interface with the following components:

- Running Processes:** A table showing the current state of processes.

Pid	Name	State	Memory	Priority	Burst	Swap	PName	CPU	PPid
4	DEADLOCKP1	Running	1	3	2	No	P1	0	0
- Ready Processes (Ready Queue):** A table showing processes waiting for the CPU.

Pid	Name	State	Memory	Priority	Burst	Swap	PName	CPU	PPid
5	DEADLOCKP2	Ready	1	3	2	No	P2	1	1
6	DEADLOCKP3	Ready	1	3	2	No	P3	2	2
7	DEADLOCKP4	Ready	1	3	2	No	P4	3	3
- Waiting Processes (Waiting Queue):** A table showing processes waiting for resources.

Pid	Name	State	Memory	Priority	Burst	Swap	PName	CPU	PPid
5	DEADLOCKP2	Waiting	1	3	2	No	P2	1	1
6	DEADLOCKP3	Waiting	1	3	2	No	P3	2	2
7	DEADLOCKP4	Waiting	1	3	2	No	P4	3	3
- SCHEDULER:** A panel showing scheduling policies (First-Come, First-Served (FCFS), Shortest-Job-First (SJF), Round Robin (RR), etc.) and system resources.
- Resource List:** A panel showing the allocation of resources (R0, R1, R2, R3, R4, R5) to processes. It includes a 'Resource colour key' and a 'Prevent' section.
- GENERAL PURPOSE CPU REGISTERS:** A panel showing the state of CPU registers (Reg, Val, D, C, Val, D).

Notamos el mismo desarrollo que con los 6 procesos, fue bastante más rápido y el contador queda en 6, que como establecimos anteriormente, cuenta los posibles casos evitados en los que se pudo causar un deadlock.

6 procesos:

System Resources

Allocating resources for process id

Resource List

Used by

Requested by

11

Allocate

Block

☐

Release

R0

Used by

Requested by

7

6

Allocate

Block

☒

Release

R1

Used by

Requested by

7

Allocate

Block

☐

Release

R2

Used by

Requested by

Allocate

Block

☐

Release

R3

Used by

Requested by

Allocate

Block

☒

Release

R4

Used by

Requested by

10

Allocate

Block

☐

Release

R5

Resource colour key:

Available

Allocated only

Allocated + Requested

Stay on top

☒

Prevent

Disallow hold and wait ☒

Disallow circular wait ☐

Use total ordering ☐

RESET

Recover

Abort processes ☐

Pre-empt resources ☐

RESET

Avoid

☐ Enable

Detect

Do not detect ☐ ☐ Every sec

☐ Randomly ☐ CPU Utilization < %

☒ After every alloc. and de-alloc.

SHOW DEADLOCKED PROCESSES...

Deadlock Count RESET

RELEASE ALL

CLOSE

System Resources

Allocating resources for process id

Resource List

Resource	Used by	Requested by	Allocate	Block	Release
R0 (Green Square)	<input type="text"/>	<input type="text"/>	<input type="button" value="Allocate"/>	<input type="checkbox"/>	<input type="button" value="Release"/>
R1 (Green Oval)	<input type="text"/>	<input type="text"/>	<input type="button" value="Allocate"/>	<input checked="" type="checkbox"/>	<input type="button" value="Release"/>
R2 (Green Square)	<input type="text"/>	<input type="text"/>	<input type="button" value="Allocate"/>	<input type="checkbox"/>	<input type="button" value="Release"/>
R3 (Green Circle)	<input type="text"/>	<input type="text"/>	<input type="button" value="Allocate"/>	<input type="checkbox"/>	<input type="button" value="Release"/>
R4 (Green Oval)	<input type="text"/>	<input type="text"/>	<input type="button" value="Allocate"/>	<input checked="" type="checkbox"/>	<input type="button" value="Release"/>
R5 (Green Square)	<input type="text"/>	<input type="text"/>	<input type="button" value="Allocate"/>	<input type="checkbox"/>	<input type="button" value="Release"/>

Resource colour key: ■ Available ■ Allocated only ■ Allocated + Requested

Stay on top ☒

Prevent

Disallow hold and wait ☒

Disallow circular wait ☐

Use total ordering ☐

Recover

Abort processes ☐

Pre-empt resources ☐

Avoid

☐ Enable

Detect

Do not detect ☐ ☐ Every sec

☐ Randomly ☐ CPU Utilization < %

☒ After every alloc. and de-alloc.

Notamos que, a diferencia de los otros procesos, no hubo un deadlock, y “Disallow hold and wait”, subió su contador a 12. Lo que pasa al activar dicha opción previene que un proceso retenga un recurso mientras solicita otro. Gracias a eso, cada proceso debe solicitar todos los recursos que va a necesitar desde un inicio, por eso en el proceso se nota que van cambiando de color, porque están utilizando dichos recursos y finalmente los liberan antes de solicitar uno nuevo, por todo lo dicho, no hay posibilidad de que se genere espera circular y por ende no hay deadlock.

Finalmente, el contador subió a 12 porque este corresponde al conteo de deadlocks que pudieron ocurrir y refleja que el sistema está reintentando asignaciones bajo las condiciones que impone activar disallow hold and wait.

Paso 2: Repetir el proceso para condición de bloqueo de 4 y 6 procesos, pero antes de seleccionar START habilitar la casilla “Disallow circular wait”.

4 procesos:



OS Simulator: CPU 0 [YASMIN: CPU-OS Simulator, Version: 7.5.50, Copyright © 2006-2013, Besim Mustafa, Edge Hill University, UK]

RUNNING PROCESSES

Pid	Name	State	Memory	Priority	Burst	Swap	PName	CPU	PPid

READY PROCESSES (Ready Queue)

Pid	Name	State	Memory	Priority	Burst	Swap	PName	CPU	PPid

WAITING PROCESSES (Waiting Queue)

Pid	Name	State	Memory	Priority	Burst	Swap	PName	CPU	PPid
4	DEADLOCKP1	1	3	2	No	P1	0		
5	DEADLOCKP2	1	3	2	No	P2	1		
6	DEADLOCKP3	1	3	2	No	P3	2		
7	DEADLOCKP4	1	3	2	No	P4	3		

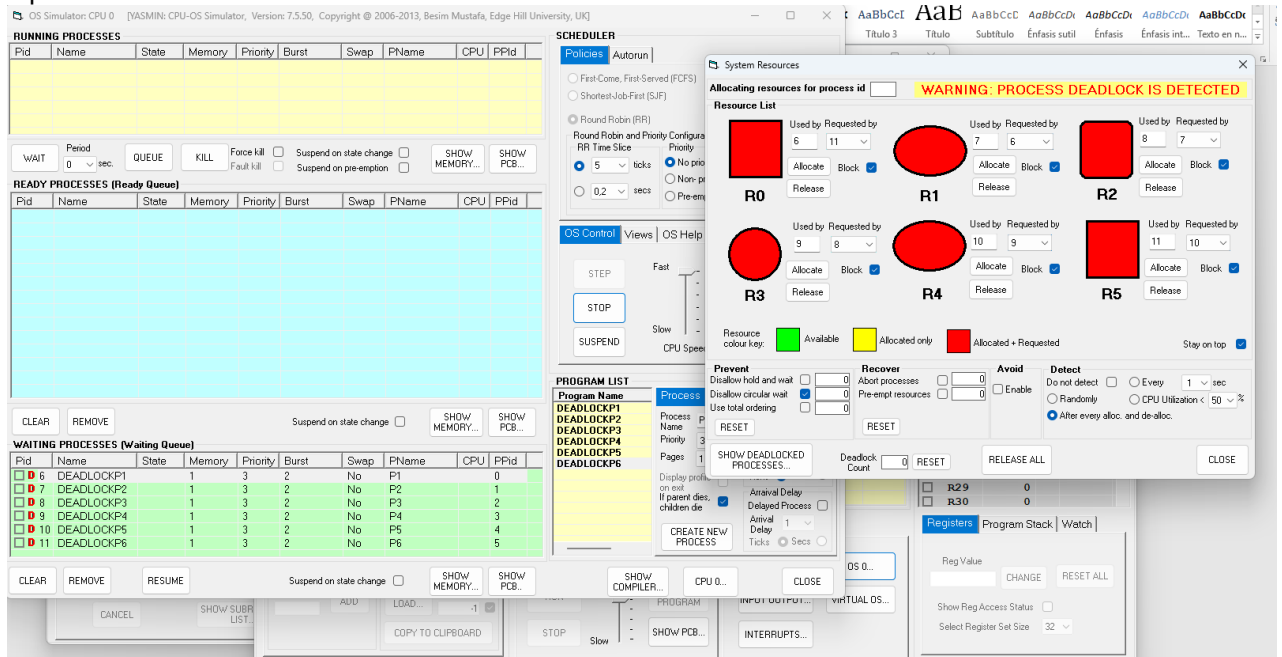
WARNING: PROCESS DEADLOCK IS DETECTED

Resource List:

Resource	Used by	Requested by	Allocated	Block	Release
R0	4	7			
R1	5	4			
R2	6	5			
R3	7	6			
R4					
R5					

Se vio un resultado bastante similar al de los 6 procesos, donde se causa el deadlock, pese a que se activa la opción que debería evitarlo.

6 procesos:



OS Simulator: CPU 0 [YASMIN: CPU-OS Simulator, Version: 7.5.50, Copyright © 2006-2013, Besim Mustafa, Edge Hill University, UK]

RUNNING PROCESSES

Pid	Name	State	Memory	Priority	Burst	Swap	PName	CPU	PPid

READY PROCESSES (Ready Queue)

Pid	Name	State	Memory	Priority	Burst	Swap	PName	CPU	PPid

WAITING PROCESSES (Waiting Queue)

Pid	Name	State	Memory	Priority	Burst	Swap	PName	CPU	PPid
6	DEADLOCKP1	1	3	2	No	P1	0		
7	DEADLOCKP2	1	3	2	No	P2	1		
8	DEADLOCKP3	1	3	2	No	P3	2		
9	DEADLOCKP4	1	3	2	No	P4	3		
10	DEADLOCKP5	1	3	2	No	P5	4		
11	DEADLOCKP6	1	3	2	No	P6	5		

WARNING: PROCESS DEADLOCK IS DETECTED

Resource List:

Resource	Used by	Requested by	Allocated	Block	Release
R0	6	11			
R1	7	6			
R2	8	7			
R3	9	8			
R4	10	9			
R5	11	10			

La opción disallow circular wait, aunque se supone debe prevenir deadlocks, en esta práctica ha fallado porque los procesos ya han adquirido recursos antes de activar esta opción, permitiendo que un ciclo exista en el momento de la verificación. Además, el simulador puede implementar una versión no estricta que solo evita ciclos inmediatos, pero no detecta ni rompe ciclos formados durante condiciones de carrera o en escenarios con múltiples instancias de recursos, esto explica porque aún se genera el deadlock.

Paso 3: Repetir el proceso para condición de bloqueo de 4 y 6 procesos, pero antes de seleccionar START habilitar la casilla “Use total ordering”.

4 procesos:



The screenshot shows the VASIMIN CPU-OS Simulator interface. The main window displays the state of the system with the following data:

Pid	Name	State	Memory	Priority	Burst	Swap	PName	CPU	PPid
5	DEADLOCKP2	Running	1	3	4	No	P2	0	1

Pid	Name	State	Memory	Priority	Burst	Swap	PName	CPU	PPid
7	DEADLOCKP4	Ready	1	3	4	No	P4	3	3
4	DEADLOCKP1	Ready	1	3	4	No	P1	0	0

Pid	Name	State	Memory	Priority	Burst	Swap	PName	CPU	PPid
6	DEADLOCKP3	Waiting	1	3	2	No	P3	2	2

The 'Resource List' window shows 5 resources (R0-R4) with their allocation and request status:

Resource	Used by	Requested by	Allocation	Block	Release
R0	4	5	1	Yes	Yes
R1	5	6	1	Yes	Yes
R2	6	7	1	Yes	Yes
R3	7	4	1	Yes	Yes
R4	4	6	1	Yes	Yes

The 'SCHEDULER' window shows the 'Policies' tab with 'Round Robin (RR)' selected. The 'OS Control' window shows the 'STOP' button. The 'PROGRAM LIST' window shows the list of processes: DEADLOCKP1, DEADLOCKP2, DEADLOCKP3, and DEADLOCKP4.

Podemos notar un comportamiento similar al ejemplo de los 6 procesos y recursos, con la diferencia en el contador de la opción, que como fue anteriormente establecido son los posibles casos en los que pudo haber un deadlock, este es de 3, debido a que son menos procesos y recursos los que estaban en ejecución

6 procesos:



RELEASE ALL

CLOSE



System Resources

Allocating resources for process id

Resource List

Resource	Used by	Requested by	Allocate	Block	Release
R0			<input type="button" value="Allocate"/>	<input type="checkbox"/>	<input type="button" value="Release"/>
R1			<input type="button" value="Allocate"/>	<input checked="" type="checkbox"/>	<input type="button" value="Release"/>
R2			<input type="button" value="Allocate"/>	<input type="checkbox"/>	<input type="button" value="Release"/>
R3			<input type="button" value="Allocate"/>	<input type="checkbox"/>	<input type="button" value="Release"/>
R4			<input type="button" value="Allocate"/>	<input checked="" type="checkbox"/>	<input type="button" value="Release"/>
R5			<input type="button" value="Allocate"/>	<input type="checkbox"/>	<input type="button" value="Release"/>

Resource colour key: ■ Available ■ Allocated only ■ Allocated + Requested

Stay on top ☒

Prevent

Disallow hold and wait ☐

Disallow circular wait ☐

Use total ordering ☒

Recover

Abort processes ☐

Pre-empt resources ☐

Avoid

☐ Enable

Detect

Do not detect ☐ ☐ Every sec

☐ Randomly ☐ CPU Utilization < %

☒ After every alloc. and de-alloc.

Notamos que al activar “Use total ordering”, todos los recursos son utilizados y solicitados (se nota por su cambio de color), pero nunca se causa un deadlock, mientras eso ocurre notamos que el contador de la opción sube a 5. Dicha opción establece un orden global para todos los recursos, cada proceso debe solicitar los recursos en dicho orden impuesto, esto causa que todos los recursos sean asignados y solicitados, sin producir un deadlock, ya que, elimina la posibilidad de espera circular, porque, un recurso de orden mayor, nunca podrá solicitar uno de orden menor, que otro proceso ya esté usando. El contador sube a 5, porque cuenta los posibles deadlocks evitados por esta regla que impone la opción activada.

4. INFORME

5. CONCLUSIONES Y RECOMENDACIONES

En conclusión, es fundamental conocer como funcionan los procesos dentro de un sistema operativo, conocer las razones por las que pueden entrar en un deadlock y saber cómo prevenirlos, este conocimiento lo obtuvimos con los incisos en los que se solicitaba activar opciones como “Disallow hold and wait”, remover un proceso, etc. Gracias a esto, tendremos los conocimientos necesarios para planificar cómo programar los procesos en futuros trabajos.

Gracias a la simulación con CPU-OS Simulator permitió visualizar de manera práctica cómo se generan, detectan y resuelven los interbloqueos, confirmando que la prevención mediante las opciones

como “Disallow hold and wait” o “Use total ordering” es efectiva. Este ejercicio refuerza la importancia de diseñar sistemas con estrategias que equilibren seguridad y eficiencia, aplicando técnicas adecuadas según el contexto para evitar bloqueos que comprometan la disponibilidad del sistema.