Sistemi Operativi e Laboratorio, Prova del 10/4/2018 – compito B

Nome: _____ Cognome: _____ Matricola:_____ corso:____ fila:___ posto: ___

Esercizio 1 (5 punti)

Un sistema con 5 processi (A, B, C, D, E) e risorse dei tipi R1, R2, R3, R4, rispettivamente di molteplicità [7, 7, 3, 6], utilizza l'algoritmo del banchiere per evitare lo stallo. Il sistema ha raggiunto lo stato (sicuro) mostrato nelle tabelle seguenti.

Assegnazione attuale				
	R1	R2	R3	R4
A			1	
B	2	1		3
C		2		2
D	1			
E	3	2	2	

Esigenza residua				
	R1	R2	R3	R4
A	0	2	0	1
B	0	2	3	1
C	0	0	2	0
D	2	1	2	0
E	0	0	1	0

Molteplicità			
R1	R2	R3	R4
7	7	3	6

Disponibilità			
1	2	0	1

Si considerino ora i seguenti casi (**in alternativa**):

a. Il processo D richiede un'istanza della risorsa R1
b. il processo A richiede due istanze della risorsa R2 (richiesta indivisibile)

In ognuno dei due casi, dire se la risorsa viene assegnata dal sistema operativo.

Soluzione

a) Stato raggiunto dopo l'ipotetica assegnazione di un'istanza di R1 al processo D :

Assegnazione attuale				
	R1	R2	R3	R4
A				
B				
C				
D				
E				

Esigenza residua (dopo l'assegnazione attuale)				
	R1	R2	R3	R4
A				
B				
C				
D				
E				

Molteplicità			
R1	R2	R3	R4
7	7	3	6

Disponibilità			

a1) Il processo _____ può/non può terminare; la disponibilità di {R1, R2, R3, R4} diviene _____
a2) Il processo _____ può/non può terminare; la disponibilità di {R1, R2, R3, R4} diviene _____
a3) Il processo _____ può/non può terminare; la disponibilità di {R1, R2, R3, R4} diviene _____
a4) Il processo _____ può/non può terminare; la disponibilità di {R1, R2, R3, R4} diviene _____
a5) Il processo _____ può/non può terminare; la disponibilità di {R1, R2, R3, R4} diviene _____

Stato sicuro? _____
Di conseguenza: risorsa assegnata? _____

b) Stato raggiunto dopo l'ipotetica assegnazione di due istanze di R2 al processo A:

Assegnazione attuale				
	R1	R2	R3	R4
A				
B				
C				
D				
E				

Esigenza residua (dopo l'assegnazione attuale)				
	R1	R2	R3	R4
A				
B				
C				
D				
E				

Molteplicità			
R1	R2	R3	R4
7	7	3	6

Disponibilità			

b1) Il processo _____ può/non può terminare; la disponibilità di {R1, R2, R3, R4} diviene _____
b2) Il processo _____ può/non può terminare; la disponibilità di {R1, R2, R3, R4} diviene _____
b3) Il processo _____ può/non può terminare; la disponibilità di {R1, R2, R3, R4} diviene _____
b4) Il processo _____ può/non può terminare; la disponibilità di {R1, R2, R3, R4} diviene _____
b5) Il processo _____ può/non può terminare; la disponibilità di {R1, R2, R3, R4} diviene _____

Stato sicuro? _____

Di conseguenza: risorsa assegnata? _____

Soluzione

a) Stato raggiunto dopo l'ipotetica assegnazione di un'istanza di R1 al processo D:

Assegnazione attuale				
	R1	R2	R3	R4
A			1	
B	2	1		3
C		2		2
D	2			
E	3	2	2	

Esigenza residua (dopo l'assegnazione attuale)				
	R1	R2	R3	R4
A	0	2	0	1
B	0	2	3	1
C	0	0	2	0
D	1	1	2	0
E	0	0	1	0

Molteplicità			
R1	R2	R3	R4
7	7	3	6

Disponibilità			
0	2	0	1

a1) Il processo A può terminare; la disponibilità di {R1, R2, R3, R4} diviene {0,2,1,1}
a2) Il processo E può terminare; la disponibilità di {R1, R2, R3, R4} diviene {3,4,3,1}
a3) Il processo B può terminare ; la disponibilità di {R1, R2, R3, R4} diviene {5,5,3,4}
a4) Il processo C può terminare ; la disponibilità di {R1, R2, R3, R4} diviene {5,7,3,6}
a5) Il processo D può terminare; la disponibilità di {R1, R2, R3, R4} diviene {7,7,3,6}

Stato sicuro? SI
Di conseguenza: risorsa assegnata? SI

b) Stato raggiunto dopo l'ipotetica assegnazione di due istanze di R2 al processo A:

Assegnazione attuale				
	R2	R4	R1	R3
A		2	1	
B	2	1		3
C		2		2
D	1			
E	3	2	2	

Esigenza residua (esclusa l'assegnazione attuale)				
	R2	R4	R1	R3
A	0	0	0	1
B	0	2	3	1
C	0	0	2	0
D	2	1	2	0
E	0	0	1	0

Molteplicità			
R1	R2	R3	R4
7	7	3	6

Disponibilità			
1	0	0	1

b1) Il processo A può terminare; la disponibilità di {R1, R2, R3, R4} diviene {1,2,1,1}
b2) Il processo E può terminare ; la disponibilità di {R1, R2, R3, R4} diviene {4,4,3,1}
b3) Il processo B può terminare; la disponibilità di {R1, R2, R3, R4} diviene {6,5,3,4}
b4) Il processo C può terminare ; la disponibilità di {R1, R2, R3, R4} diviene {6,7,3,6}
b5) Il processo D può terminare ; la disponibilità di {R1, R2, R3, R4} diviene {7,7,3,6}

Stato sicuro? SI
Di conseguenza: risorsa assegnata? SI

Esercizio 2 (5 punti)

Si consideri un processore che dispone dei seguenti registri:
- i registri speciali PC (program counter) e PS (program status) e lo stack pointer SP
- un banco di registri riservato allo stato utente, che comprende i registri generali R1, R2, R3,
- un ulteriore banco di registri riservato allo stato supervisore, che comprende i registri generali R'1, R'2, R'3 e lo stack pointer dello stack del nucleo SP'.

Il sistema operativo realizza i thread a livello kernel, con scheduling Round Robin e i thread costituiscono l'unica unità di schedulazione del sistema. Come meccanismi di mutua esclusione e di sincronizzazione, il sistema offre ai thread i lock e le variabili di condizione. Queste ultime sono implementate con semantica **MESA**.

Al tempo t sono presenti solo due thread: il thread T1, in stato di esecuzione, e il thread T2, che è bloccato sulla variabile di condizione COND. Nessun altro thread è presente nel sistema. Al tempo *t* il thread T1 esegue una istruzione SVC per invocare la chiamata di sistema COND.**signal**(&mutex). Al momento dell'esecuzione dell'istruzione SVC i registri del processore, i descrittori di T1 e T2 e lo stack del nucleo hanno i contenuti mostrati in tabella. Lo stack pointer del nucleo ha il valore 1016.

Il vettore di interruzione associato all'interruzione generata da SVC è 0425 e la parola di stato del nucleo è 275E.

Si chiede:

a) il contenuto dei descrittori, dei registri generali e speciali, dello stack del nucleo e lo stato del processore durante la fase di estrazione della prima istruzione della funzione di servizio;
b) il contenuto dei descrittori, dei registri generali e speciali, dello stack del nucleo e lo stato del processore durante la fase di estrazione dell'istruzione IRET con la quale termina la chiamata di sistema;
c) il contenuto dei descrittori, dei registri generali e speciali, dello stack del nucleo e lo stato del processore durante la fase di estrazione dell'istruzione eseguita subito dopo la IRET.

Descrittore di T1	
Stato	Esecuzione
PC	2E31
PS	16F2
SP	1873
R1	1234
R2	16CC
R3	1777

Descrittore di T2	
Stato	Bloccato
PC	B12C
PS	B6F2
SP	BFF5
R1	B5CC
R2	B000
R3	B011

Stack del nucleo	
……	……
1016	
1014	
1012	
1010	
100E	
100C	

Registri generali	
SP	AFFF
R1	AA2A
R2	A2CE
R3	A2CF

Registri stato supervisore	
SP'	1016
R1'	?
R2'	?
R3'	?

Processore: registri speciali					
PC	2F00	PS	16F2	stato	utente

Soluzione

a) contenuto dei descrittori, dei registri generali e speciali, dello stack del nucleo e lo stato del processore durante la fase di estrazione della prima istruzione della funzione di servizio:

Descrittore di T1	
Stato	
PC	
PS	
SP	
R1	
R2	
R3	

Descrittore di T2	
Stato	
PC	
PS	
SP	
R1	
R2	
R3	

Stack del nucleo	
……	……
1016	
1014	
1012	
1010	
100E	
100C	

Registri generali	
SP	
R1	
R2	
R3	

Registri stato supervisore	
SP'	
R1'	
R2'	
R3'	

Processore: registri speciali					
PC		PS		stato	

b) contenuto dei descrittori, dei registri generali e speciali, dello stack del nucleo e lo stato del processore durante la fase di estrazione dell'istruzione IRET con la quale termina la chiamata di sistema;

Descrittore di T1	
Stato	
PC	
PS	
SP	
R1	
R2	
R3	

Descrittore di T2	
Stato	
PC	
PS	
SP	
R1	
R2	
R3	

Stack del nucleo	
……	……
1016	
1014	
1012	
1010	
100E	
100C	

Registri generali	
SP	
R1	
R2	
R3	

Registri stato supervisore	
SP'	
R1'	
R2'	
R3'	

Processore: registri speciali					
PC		PS		stato	

c) Contenuto dei descrittori, dei registri generali e speciali, dello stack del nucleo e lo stato del processore durante la fase di estrazione dell'istruzione eseguita subito dopo la IRET.

Descrittore di T1	
Stato	
PC	
PS	
SP	
R1	
R2	
R3	

Descrittore di T2	
Stato	
PC	
PS	
SP	
R1	
R2	
R3	

Stack del nucleo	
……	……
1016	
1014	
1012	
1010	
100E	
100C	

Registri generali	
SP	
R1	
R2	
R3	

Registri stato supervisore	
SP'	
R1'	
R2'	
R3'	

Processore: registri speciali					
PC		PS		stato	

Soluzione

Sistemi Operativi e Laboratorio, Prova del 10/4/2018 – compito B

a) Contenuto dei descrittori, dei registri generali e speciali, dello stack del nucleo e lo stato del processore durante la fase di estrazione della prima istruzione della funzione di servizio:

Descrittore di T1	
Stato	Esecuzione
PC	2E31
PS	16F2
SP	1873
R1	1234
R2	16CC
R3	1777

Descrittore di T2	
Stato	Bloccato
PC	B12C
PS	B6F2
SP	BFF5
R1	B5CC
R2	B000
R3	B011

Stack del nucleo	
……	……
1016	2F00
1014	16F2
1012	
1010	
100E	
100C	

Registri generali	
SP	AFFF
R1	AA2A
R2	A2CE
R3	A2CF

Registri stato supervisore	
SP'	1012
R1'	?
R2'	?
R3'	?

Processore: registri speciali					
PC	0425	PS	275E	stato	supervisore

b) Contenuto dei descrittori, dei registri generali e speciali, dello stack del nucleo e lo stato del processore durante la fase di estrazione dell'istruzione IRET con la quale termina la chiamata di sistema;

Descrittore di T1	
Stato	Esecuzione
PC	2E31
PS	16F2
SP	1873
R1	1234
R2	16CC
R3	1777

Descrittore di T2	
Stato	Pronto
PC	B12C
PS	B6F2
SP	BFF5
R1	B5CC
R2	B000
R3	B011

Stack del nucleo	
……	……
1016	2F00
1014	16F2
1012	
1010	
100E	
100C	

Registri generali	
SP	AFFF
R1	AA2A
R2	A2CE
R3	A2CF

Registri stato supervisore	
SP'	1012
R1'	?
R2'	?
R3'	?

Processore: registri speciali					
PC	0425+?	PS	275E	stato	supervisore

c) Contenuto dei descrittori, dei registri generali e speciali, dello stack del nucleo e lo stato del processore durante la fase di estrazione dell'istruzione eseguita subito dopo la IRET.

Descrittore di T1	
Stato	Esecuzione
PC	2E31
PS	16F2
SP	1873
R1	1234
R2	16CC
R3	1777

Descrittore di T2	
Stato	Pronto
PC	B12C
PS	B6F2
SP	BFF5
R1	B5CC
R2	B000
R3	B011

Stack del nucleo	
……	……
1016	
1014	
1012	
1010	
100E	
100C	

Registri generali	
SP	AFFF
R1	AA2A
R2	A2CE
R3	A2CF

Registri stato supervisore	
SP'	1016
R1'	?
R2'	?
R3'	?

Processore: registri speciali					
PC	2F00	PS	16F2	stato	utente

Esercizio 3 (5 punti)

Uno spool di stampa di un sistema multithread che gestisce due stampanti fisiche, è organizzato con un thread gestore che gestisce le richieste di stampa da parte di altri thread $C_1,\ldots,C_n$ che vogliono stampare (thread clienti). I documenti da stampare vengono scambiati tra i clienti e il gestore tramite un buffer condiviso che può contenere un solo documento alla volta.
Il protocollo per la stampa funziona nel modo seguente:

- quando il generico cliente C_i vuol stampare, verifica se il buffer è occupato e in tal caso attende. Altrimenti C_i deposita il suo documento nel buffer condiviso e riattiva il gestore. Quindi C_i si mette in attesa del completamento della stampa. A stampa finita, legge l'esito della stampa e conclude la procedura.
- Il gestore esegue un ciclo infinito nel quale inizialmente si pone in attesa di un documento da stampare. Quando arriva un documento lo stampa, scrive l'esito della stampa nella variabile condivisa esitoStampa, e quindi riattiva il cliente che attendeva il completamento della stampa. Poi riattiva un eventuale thread cliente in attesa di depositare il documento da stampare nel buffer.

La soluzione proposta utilizza una variabile (`lock`) per la mutua esclusione e tre variabili di condizione:

- `condGestore` sulla quale si sospende il gestore quando non c'è nessun documento da stampare nel buffer;
- `condStampa` sulla quale si sospende il thread cliente che sta attendendo il completamento della stampa in corso;
- `condBuffer` sulla quale si sospendono i thread clienti quando il buffer è occupato.

Lo stato dello spool è inoltre rappresentato dalle seguenti variabili:

- `attesaRichieste` (inizializzata a `True`): è `False` se c'è un documento in attesa di essere stampato; è `True` altrimenti;
- `bufferOccupato` (inizializzata a `False`): è `True` se c'è nel buffer un documento in corso di stampa; è `False` altrimenti;
- `stampaInCorso` (inizializzata a `False`): è `False` se la stampa in corso è completata; è `True` altrimenti.

Sistemi Operativi e Laboratorio, Prova del 10/4/2018 – compito B

Si chiede di completare la seguente soluzione assumendo che la semantica delle variabili di condizione sia di tipo MESA.

Gestore:

```
[…]
while (True) {

       _____________________________________________
       while (attesaRichieste)

              _____________________________________________

       <stampa il documento nel buffer>
       esitoStampa = ……;   // scrive il risultato della stampa nella variabile esitoStampa

       bufferOccupato = False;
       stampaInCorso = False;
       attesaRichieste = True;

       _____________________________________________

       _____________________________________________

       _____________________________________________
       […]
       }
```

Generico thread cliente C_i:

```
[…]

_____________________________________________
while (bufferOccupato)

       _____________________________________________
<deposita documento da stampare in un buffer condiviso col gestore>
bufferOccupato = True;
stampaInCorso = True;
attesaRichieste = False;

_____________________________________________

while (stampaInCorso)

       _____________________________________________
<legge l'esito della stampa dalla variabile esitoStampa>

_____________________________________________
[…]
```

Soluzione

```
Gestore:
[…]
while (True) {
       lock.acquire();
       while (attesaRichieste)
              condGestore.wait(&lock);  // se non ci sono documenti da stampare attende

       <stampa il documento nel buffer>
       esitoStampa = ……;   // scrive il risultato della stampa nella variabile esitoStampa

       bufferOccupato = False;
       stampaInCorso = False;
       attesaRichieste = True;
       condStampa.signal();       // riattiva il thread che attende il completamento della stampa
       condBuffer.signal();       // riattiva un eventuale thread in attesa di stampare
       lock.release();
       […]
       }

Generico thread cliente Ci:
[…]
```

```
lock.acquire();
while (bufferOccupato)
      condBuffer.wait(&lock);   // se il buffer è occupato attende
<deposita documento da stampare in un buffer condiviso col gestore>
bufferOccupato = True;
stampaInCorso = True;
attesaRichieste  = False;
condGestore.signal();     // riattiva il gestore che avvia la stampa

while (stampaInCorso)
      condStampa.wait(&lock);   // attende il completamento della stampa
<legge l'esito della stampa dalla variabile esitoStampa>
lock.release();
[…]
```

Esercizio 4 (5 punti)

In un sistema in time sharing (con quanto di tempo e assegnazione del processore a rotazione tra i thread in stato di pronto) con thread realizzati a livello del Nucleo, sono presenti i thread T1, T2, T3, T4, T5 che condividono i semafori Sem1, Sem2 e una spinlock SL. Alla riattivazione dallo stato di attesa, il thread riattivato viene messo in fondo alla coda pronti.
Ad un certo tempo *t* è in esecuzione il thread T3, e gli altri thread sono nel seguente stato:

- Coda thread pronti: T4 → T5.
- T2 e T1 sono sospesi sul semaforo Sem2.

Inoltre, il semaforo Sem1 ha valore 1 e la spinlock SL ha valore 1 (BUSY).
Riempire la tabella seguente indicando quale thread è in esecuzione e quale è lo stato di Sem1, Sem2 e SL **al termine** di ciascuna sequenza, se, a partire dal tempo *t*, si verificano le sequenze di eventi A,..,E (ogni sequenza è da considerare in alternativa alle altre).

Soluzione

	SEQUENZA DI EVENTI	In Esec.	Coda pronti	Sem1: valore, coda	Sem2: valore, coda	Valore di SL
	STATO INIZIALE	**T3**	**T4 → T5**	**1, -**	**0, T2 → T1**	**BUSY**
A	il thread in esecuzione esegue V(Sem2); il thread in esecuzione esegue spinlockAcquire(SL); scade il quanto di tempo; il thread in esecuzione esegue spinlockRelease(SL);					
B	il thread in esecuzione esegue P(Sem2); il thread in esecuzione esegue V(Sem1); il thread in esecuzione esegue P(Sem1); il thread in esecuzione esegue spinlockAcquire(SL);					
C	il thread in esecuzione esegue P(Sem2); scade il quanto di tempo; il thread in esecuzione esegue P(Sem2); il thread in esecuzione esegue P(Sem2);					
D	il thread in esecuzione esegue spinlockRelease(SL); scade il quanto di tempo; il thread in esecuzione esegue spinlockAcquire(SL); il thread in esecuzione esegue P(Sem2);					
E	il thread in esecuzione esegue V(Sem1); il thread in esecuzione esegue V(Sem1); il thread in esecuzione esegue V(Sem1); il thread in esecuzione esegue P(Sem1);					

Sistemi Operativi e Laboratorio, Prova del 10/4/2018 – compito B

Soluzione

	SEQUENZA DI EVENTI	In Esec.	Coda pronti	Sem1: valore, coda	Sem2: valore, coda	Valore di SL
	STATO INIZIALE	**T3**	**T4 → T5**	**1, -**	**0, T2 → T1**	**BUSY**
A	il thread in esecuzione esegue V(Sem2); il thread in esecuzione esegue spinlockAcquire(SL); scade il quanto di tempo; il thread in esecuzione esegue spinlockRelease(SL);	T4	T5 → T2 →T3	1, -	0, T1	FREE
B	il thread in esecuzione esegue P(Sem2); il thread in esecuzione esegue V(Sem1); il thread in esecuzione esegue P(Sem1); il thread in esecuzione esegue spinlockAcquire(SL);	T4	T5	1, -	0, T2 → T1 → T3	BUSY
C	il thread in esecuzione esegue P(Sem2); scade il quanto di tempo; il thread in esecuzione esegue P(Sem2); il thread in esecuzione esegue P(Sem2);	-	-	1, -	0, T2 → T1 → T3 → T5 → T4	BUSY
D	il thread in esecuzione esegue spinlockRelease(SL); scade il quanto di tempo; il thread in esecuzione esegue spinlockAcquire(SL); il thread in esecuzione esegue P(Sem2);	T5	T3	1, -	0, T2 → T1→ T4	BUSY
E	il thread in esecuzione esegue V(Sem1); il thread in esecuzione esegue V(Sem1); il thread in esecuzione esegue V(Sem1); il thread in esecuzione esegue P(Sem1);	T3	T4 → T5	3, -	0, T2 → T1	BUSY

Esercizio 5 (3 punti)

In un sistema che implementa i thread a livello del nucleo, dire quali delle seguenti operazioni sono permesse anche quando il processore opera in stato UTENTE:

Operazione:	**Eseguibili anche in stato utente [SI/NO]**
Modificare lo stack pointer SP	
Disabilitare le interruzioni	
Modificare il valore di priorità di un thread	
Modificare lo stato (attesa/esecuzione/pronto) di un thread	
Copiare sullo stack corrente (quello puntato dal registro SP) il valore del program counter	
Eseguire una istruzione in linguaggio macchina di salto ad un indirizzo	
Eseguire l'istruzione IRET	
Modificare il valore dei registri base e limite	
Leggere il contenuto del vettore di interruzione	

Soluzione

Operazione:	**Eseguibili anche in stato utente [SI/NO]**
Modificare lo stack pointer SP	SI
Disabilitare le interruzioni	
Modificare il valore di priorità di un thread	
Modificare lo stato (attesa/esecuzione/pronto) di un thread	
Copiare il valore di PC sullo stack corrente (quello puntato dal registro SP)	SI
Eseguire una istruzione in linguaggio macchina di salto ad un indirizzo	SI
Eseguire l'istruzione IRET	
Modificare il valore dei registri base e limite	
Leggere il contenuto del vettore di interruzione	

Esercizio 6 (4 punti)

Dire quali stampe vengono prodotte dai processi che eseguono i seguenti frammenti di codice:

Frammento 1	**Frammento 2**	**Frammento 3**	**Frammento 4**

```
…
printf("DD");
a = fork();
if (a==0)
    printf("CC");
if (a>0)
    printf("EE");
if (a<0)
    printf("AA")
printf("BB")
…
```

```
…
a = fork();
printf("DD");
if (a<>0) printf("CC");
else {
   printf("AA");
   execl("prova",NULL);
   printf("EE");
}
…
```

```
…
printf("DD");
a = fork();
if (a==0) {
   b=fork();
   printf("CC");
}
else if (a>0)
   printf("EE");
else printf("BB");
}
…
```

```
…
printf("DD");
a = fork();
if (a==0) {
   printf("CC");
   execl("prova",NULL);
   printf("BB");
}
if (a<0)  exit()
if (a>0)
   printf("EE");
…
```

Frammento 1:

- il padre stampa: _____ _____;
- il figlio stampa: _____ _____;

Frammento 2:

- il padre stampa: _____ _____;
- il figlio stampa: _____ _____;

Frammento 3:

- il padre stampa: _____ _____;
- il figlio stampa: _____ _____;

Frammento 4:

- il padre stampa: _____ _____;
- il figlio stampa: _____ _____;

Soluzione

Frammento 1:

- Il processo padre stampa: DDEEBB se la prima fork ha successo. Altrimenti stampa DDAABB
- Se la fork ha successo il figlio stampa: CCBB

Frammento 2:

- Il processo padre stampa: DDCC
- Se la fork e la exec hanno successo, il processo figlio stampa DDAA
- Se la fork ha successo e la exec fallisce, il processo figlio stampa DDAAEE

Frammento 3:

- Il processo padre stampa: DDEE se la prima fork ha successo. Altrimenti stampa DDBB
- Se la prima fork ha successo il figlio stampa: CC
- Se entrambe le fork hanno successo il secondo figlio stampa CC

Frammento 4:

- Il processo padre stampa: DDEE se la prima fork ha successo. Altrimenti stampa DD
- Se la fork e la exec hanno successo il figlio stampa: CC
- Se la fork ha successo e la exec fallisce il figlio stampa: CCBB

Esercizio 7 (3 punti)

In un sistema che implementa i thread a livello utente con scheduling senza prerilascio sono presenti i processi P1, P2 e P3. Il processo P1 utilizza 3 thread: T11, T12 e T13; il processo P2 utilizza il solo thread T21 e il processo P3 utilizza il solo thread T31.

Sistemi Operativi e Laboratorio, Prova del 10/4/2018 – compito B

Ad un certo istante di tempo è in esecuzione il thread T11 del processo P1, e la coda pronti del processo P1 contiene (nell'ordine) i thread T21 e T31. Inoltre, il processo P2 è in stato di attesa sulla variabile di condizione COND, mentre il processo P3 è in stato di pronto. Infine, la variabile lock è occupata.
Assumendo che le variabili di condizione adottino una semantica **HOARE**, dire come evolve lo stato del sistema se si verificano **in alternativa** i seguenti eventi:

	Evento	Thread in esecuzione	Processi in attesa su COND
(a)	Il thread in esecuzione esegue **COND.signal()**		
(b)	Il thread in esecuzione esegue **COND.wait(&lock)**		
(c)	Il thread in esecuzione esegue **lock.acquire()**		
(d)	Il thread in esecuzione esegue **yield()**		

Soluzione

	Evento	Thread in esecuzione	Processi in attesa su COND
(a)	Il thread in esecuzione esegue **COND.signal()**	T21	-
(b)	Il thread in esecuzione esegue **COND.wait(&lock)**	T31	P2, P1
(c)	Il thread in esecuzione esegue **lock.acquire()**	T31	P2
(d)	Il thread in esecuzione esegue **yield()**	T12	P2