Sistemi Operativi e Laboratorio, Prova del ?/?/?

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

Esercizio 1

Quali delle seguenti operazioni possono essere eseguite da un processo in stato utente?

Soluzione

Operazione:	**Eseguibili dai processi in stato utente**
Invocare l'istruzione TSL (test and set lock-tipo read-modify-write)	X
Salvare i registri generali nel proprio descrittore di processo	
Invocare una chiamata di sistema	X
Leggere il contenuto della coda dei processi pronti	
Abilitare le interruzioni	
Invocare l'istruzione IRET (ritorno da un'interruzione)	

Esercizio 2

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 generali R1, R2, R3.
- un ulteriore registro riservato allo stato supervisore: lo stack pointer SP'

Il sistema riserva in memoria un'area per il vettore di interruzione e per lo stack del nucleo. Inoltre, il processore ha un comportamento standard e salva automaticamente il minimo indispensabile sullo stack.
Al tempo *t* il processo Pi in esecuzione invoca una chiamata di sistema yield() che provoca una commutazione di contesto a favore del processo Pj, che viene messo in esecuzione. Il vettore di interruzione associato alla chiamata di sistema ha il valore 0800 e la parola di stato del nucleo è 375E.
Quando l'interruzione viene riconosciuta, i registri del processore, i descrittori di Pi e Pj e lo stack del nucleo, che inizia alla locazione 8080, hanno i contenuti mostrati in figura.
L'interruzione determina l'intervento del nucleo, che esegue una funzione di servizio comprendente lo scheduler. Si chiede:

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

DESCRITTORE DI P_J	
Stato	Pronto
PC	BAB0
PS	16F2
SP	F1C0
R1	2200
R2	2211
R3	2233

DESCRITTORE DI P_I	
Stato	Esec
PC	AA11
PS	26F2
SP	DEC3
R1	1122
R2	1133
R3	1144

STACK del nucleo	
8080	
8079	
8078	
8077	
8076	
8075	
8074	.
8073	

REG. GENERALI	
R1	1199
R2	1188
R3	1177

PROCESSORE: Registri speciali							
PC	AB00		PS	36F2	SP	DEC0	SP'	8080

Soluzione

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

DESCRITTORE DI P_J	
Stato	Pronto
PC	BAB0
PS	16F2
SP	F1C0
R1	2200
R2	2211
R3	2233

DESCRITTORE DI P_I	
Stato	Esec
PC	AA11
PS	26F2
SP	DEC3
R1	1122
R2	1133
R3	1144

STACK del nucleo	
8080	**AB00**
8079	**36F2**
8078	
8077	
8076	
8075	
8074	
8073	

REG. GENERALI	
R1	1199
R2	1188
R3	1177

PROCESSORE: Registri speciali								
PC	**0800**		PS	**375E**	SP	DEC0	SP'	**8078**

b) contenuto dei descrittori, dei registri generali e speciali e dello stack del nucleo durante la fase di estrazione dell'istruzione IRET con la quale termina la funzione di servizio;

DESCRITTORE DI P_J		DESCRITTORE DI P_I		STACK del nucleo		REG. GENERALI	
Stato	**Esec**	Stato	**Pronto**	8080	**BAB0**	R1	**2200**
PC	BAB0	PC	**AB00**	8079	**16F2**	R2	**2211**
PS	16F2	PS	**36F2**	8078		R3	**2233**
SP	F1C0	SP	**DEC0**	8077			
R1	2200	R1	**1199**	8076			
R2	2211	R2	**1188**	8075			
R3	2233	R3	**1177**	8074			
				8073			
PROCESSORE: Registri speciali							
PC	**0800+??**	PS	375E	SP	DEC0	SP'	**8078**

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

DESCRITTORE DI P_J		DESCRITTORE DI P_I		STACK del nucleo		REG. GENERALI	
Stato	Esec	Stato	Pronto	8080		R1	2200
PC	BAB0	PC	AB00	8079		R2	2211
PS	16F2	PS	36F2	8078		R3	2233
SP	F1C0	SP	DEC0	8077			
R1	2200	R1	1199	8076			
R2	2211	R2	1188	8075			
R3	2233	R3	1177	8074			
				8073			
PROCESSORE: Registri speciali							
PC	**BAB0**	PS	**16F2**	SP	**F1C0**	SP'	**8080**

Esercizio 3

In un sistema con thread realizzati a livello kernel, dove l'unità schedulabile è il thread, sono presenti i processi P1 con thread P11, P12, P13, e P2 con thread P21 e P22. Il processore viene gestito in time sharing con quanti di tempo e coda pronti gestita in modo FIFO (politica Round Robin). Immediatamente prima del tempo *t* è in esecuzione il thread P13, i thread P21 e P12 sono sospesi sul semaforo SA e i rimanenti thread sono pronti, con il seguente ordinamento nella coda pronti: (primo) P22->P11 (ultimo). Inoltre nel sistema è presente il semaforo SB con valore 2. Riempire la seguente tabella indicando quale thread è in esecuzione se dal tempo t si verificano (in sequenza) i seguenti eventi:

Soluzione **P13** **0/P21,P12** **2/-** **(P22->P11)**

	Evento	Thread in esecuzione	Valore di SA / thread sospesi su SA	Valore di SB e thread sospesi su SB
(a)	Il thread in esecuzione esegue P(SA)	**P22**	**0 / P21, P12,P13**	**2 / -**
(b)	Il thread in esecuzione esegue V(SB)	**P22**	**0 / P21, P12,P13**	**3 / -**
(c)	Scade il quanto di tempo	**P11**	**0 / P21, P12,P13**	**3 / -**
(d)	Il thread in esecuzione esegue P(SB)	**P11**	**0 / P21, P12,P13**	**2 / -**

Esercizio 4

La segreteria studenti dell'Università di Pisa è dotata di 5 sportelli per il servizio agli studenti, e di una sala d'aspetto. Gli studenti entrano nella sala d'aspetto, attendono che vi sia uno sportello disponibile e, quando si verifica questo evento, lo individuano, lo raggiungono e ottengono il servizio voluto. Quindi liberano lo sportello e tornano in sala d'aspetto prima di uscire dall'ufficio.
Si consideri una formalizzazione del problema nel quale gli studenti sono thread di uno stesso processo, realizzati a livello del nucleo, che si sincronizzano mediante la variabile *lock* per la mutua esclusione, e la variabile di condizione *SportelloDisponibile*. i thread utilizzano inoltre la variabile *SportelliDisponibili*, di tipo intero e inizializzata al valore 5.

Sistemi Operativi e Laboratorio, Prova del ?/?/?

Si chiede di completare il seguente schema di soluzione e di indicare inoltre quale thread viene riattivato quando uno sportello viene liberato.

Soluzione

```
< lo studente entra nella sala d'aspetto>
lock.acquire();
while (SportelliDisponibili == 0) SportelloDisponibile.wait(lock);
//esiste uno sportello disponibile: lo studente lo individua e lo raggiunge//
SportelliDisponibili --;
lock.release();
< lo studente ottiene il servizio richiesto>
lock.acquire();
< lo studente lascia lo sportello e torna in sala d'aspetto>
SportelliDisponibili ++;
SportelloDisponibile.signal();
lock.release();
< lo studente lascia la sala d'aspetto>
```

Viene riattivato il thread: **uno a caso tra quelli in attesa. Se nessun thread è in attesa la signal non ha effetto.**

Esercizio 5

Si consideri il seguente frammento di codice:

```
…
printf("Pippo");
a = fork();
if (a!=0) {
        b=fork();
        if (b!= 0) { printf("Pluto"); exit(0);}
        else printf("Qui");
        }
else printf("Quo");
printf("Qua");
…
```

Il processo generato con la prima fork è il primo figlio.
Il processo generato con la seconda fork è il secondo figlio.
Dire cosa stampano il padre e i due figli nel caso in cui la prima fork abbia successo, e la seconda no.

Soluzione

Il **padre** stampa: "**PippoPluto**".
Il **primo** figlio stampa: "**QuoQua**".
Il **secondo** figlio **non viene generato e quindi non stampa niente**.

Esercizio 6

In un sistema con thread *realizzati a livello kernel,* dove l'unità schedulabile è il thread, sono presenti i processi P1 con thread P11, P12, P13, e P2 con thread P21, P22 e P23. Il processore viene gestito in time sharing con quanti di tempo e coda pronti gestita in modo FIFO (politica Round Robin). Ad un certo istante di tempo è in esecuzione il thread P11, i thread P21 e P23 sono sospesi sulla variabile di condizione *Condizione* (legata alla variabile lock *mux*) e i rimanenti thread sono pronti, con il seguente ordinamento nella coda pronti: (primo) P22->P12->P13 (ultimo). Infine, le variabili di condizione sono realizzate con semantica MESA.
Dire come sarà lo stato del sistema subito dopo che si verificano in sequenza i seguenti eventi:

Soluzione **P11** **P21,P23** **Pronti: P22->P12->P13**

	Evento	Thread in esecuzione	thread sospesi su Condizione	Stato di mux / thread sospesi su mux
(a)	Il thread in esecuzione esegue lock.release(mux)	**P11**	**P21,P23**	**Libero / -**
(b)	Scade il quanto di tempo	**P22**	**P21,P23**	**Libero / -**
(c)	Il thread in esecuzione esegue lock.acquire(mux)	**P22**	**P21,P23**	**Bloccato / -**
(d)	Il thread in esecuzione esegue Condizione.wait(mux)	**P12**	**P21, P23,P22**	**Libero / -**

Sistemi Operativi e Laboratorio, Prova del ?/?/?

Esercizio 7

Si consideri un sistema nel quale sono definiti il semaforo *sem* e i processi P1 (con priorità 3), P2 (con priorità 2) e P3 (con priorità 1). Lo scheduling avviene con una politica a priorità, in ordine decrescente di priorità e, a pari priorità, in ordine di arrivo. Al tempo *t* il processo P3 è in esecuzione e la coda pronti è vuota. Inoltre il semaforo *sem* ha valore 0 e la coda associata al semaforo contiene (nell'ordine) i processi P2->P1. A partire dal tempo *t* si verifica la seguente sequenza di eventi:

1) Il processo in esecuzione esegue *V(sem)*
2) il processo in esecuzione esaurisce il quanto di tempo;
3) il processo in esecuzione esegue *P(sem)*
4) il processo in esecuzione esegue *V(sem)*

Si chiede di specificare quale processo è in esecuzione dopo ogni evento e inoltre come si modificano il valore e la coda del semaforo *sem* e la *CodaPronti*.

Soluzione **P3** **vuota** **0** **P2->P1**

Sequenza di eventi	In Esecuzione	Coda Pronti	Valore di sem	Coda di sem
1) il processo in esecuzione esegue *V(sem)*	**P2**	**P3**	**0**	**P1**
2) scade il quanto di tempo	**P2**	**P3**	**0**	**P1**
3) il processo in esecuzione esegue *P(sem)*	**P3**	**vuota**	**0**	**P1 -> P2**
4) Il processo in esecuzione esegue *V(sem)*	**P1**	**P3**	**0**	**P2**

NOTA: alla scadenza del quanto di tempo di P2 (evento 2) rimane in esecuzione P2, perché non esistono altri processi pronti con priorità uguale o maggiore.

Esercizio 8

Un sistema con processi A, B, C, D, E e risorse dei tipi R1, R2, R3, R4, ha raggiunto lo stato mostrato nelle tabelle seguenti, che è uno stato sicuro:

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

Esigenza **Massima**				
	R1	R2	R3	R4
A	2			3
B	3	1	1	2
C	4	4	5	3
D	2	3	3	1
E	3	3	5	4

Molteplicità		
R1	R2	R3 | R4
6	4	7 | 7

Disponibilità		
0	1	1 | 4

Successivamente, i processi A e C eseguono **in sequenza** le seguenti richieste:

a) A richiede 2 istanze di R4 (richiesta multipla: deve essere soddisfatta integralmente)
b) C richiede 1 istanza di R2

Il gestore delle risorse applica l'algoritmo del banchiere per evitare lo stallo. Verificare se il gestore assegna le risorse richieste.

Soluzione

Stato raggiunto dopo l'assegnazione di 2 istanze di R4 al processo A:

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

Esigenza ***Residua***				
	R1	R2	R3	R4
A	0	0	0	**0**
B	2	1	0	2
C	2	4	4	3
D	2	1	0	1
E	2	2	4	2

Molteplicità		
R1	R2	R3 | R4
6	4	7 | 7

Disponibilità		
0	1	1 | **2**

Il processo **A può** terminare, il vettore disponibilità diventa: **2,1,1,5**
Il processo **B** può terminare, il vettore disponibilità diventa: **3,1,2,5**
Il processo **D** può terminare, il vettore disponibilità diventa: **3,3,5,5**
Il processo **E** può terminare, il vettore disponibilità diventa: **4,4,6,7**
Il processo **C** può terminare, il vettore disponibilità diventa: **6,4,7,7**

a) La richiesta lascia il sistema in uno stato sicuro? **SI** Può essere accettata? **SI**

Sistemi Operativi e Laboratorio, Prova del ?/?/?

Stato raggiunto dopo l'assegnazione di 1 istanza di R2 al processo C:

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

Esigenza ***Residua***				
	R1	R2	R3	R4
A	0	0	0	0
B	2	1	0	2
C	2	**3**	4	3
D	2	1	0	1
E	2	2	4	2

Molteplicità		
R1	R2	R3 | R4
6	4	7 | 7

Disponibilità		
0	**0**	1 | 2

Il processo **A può** terminare, il vettore disponibilità diventa: **2,0,1,5**

Nessun altro processo può terminare. Se accettata, **la richiesta lascerebbe il sistema in uno stato non sicuro**, quindi n**on viene accordata e il processo C viene sospeso.**

Esercizio 9

In un sistema con processi P1, P2, P3 e risorse R1, R2, R3 e R4, al tempo t si ha la seguente situazione:

- la disponibilità di R1 è 0, la disponibilità di R2 è 1, la disponibilità di R3 è 2, la disponibilità di R4 è 1;
- P1 possiede 2 esemplari di R2 e 1 di R3, e ha chiesto 2 esemplari di R1;
- P2 possiede 3 esemplari di R1 e 2 di R2, e ha chiesto 2 esemplare di R4;
- P3 possiede 2 esemplari di R4 e e chiede 2 esemplari di R3;

Le risorse non possono essere condivise dai processi; non è ammesso il prerilascio e i processi osservano il paradigma di "possesso e attesa". Inoltre il sistema prevede richieste multiple, con la convenzione che, in caso di richiesta di più esemplari di una risorsa che non siano tutti disponibili, il processo richiedente viene sospeso senza ottenerne alcuna.

Si domanda:

1. se il sistema è in stallo;
2. in caso affermativo, come si caratterizza l'attesa circolare.
3. in caso negativo, la sequenza di assegnazioni e rilasci che consente la terminazione di tutti i processi.

Soluzione

1) stallo? **NO**

2) caratterizzazione dell'attesa circolare: **Non c'e' attesa circolare**

3) sequenza che consente la terminazione di tutti i processi:

- P3 ottiene 2 di R3; quindi puo' terminare rilasciando 2 di R4 e 2 di R3

- P2 ottiene 2 di R4 e viene riattivato; puo' terminare rilasciando 2 di R4, 2 di R2 e 3 di R1

- P1 ottiene 2 di R1 e viene riattivato; puo' terminare rilasciando 2 di R1, 2 di R2 e 1 di R3