SISTEMI OPERATIVI, CORSI A e B - QUINTO APPELLO - 20/7/2006

Parte A

ESERCIZIO A-1 (4 punti)

Si consideri un processore che dispone dei seguenti registri:

- i registri speciali PC (program counter), PS (program status) e SP(stack pointer);
- i registri generali R1 e R2, utilizzati sia nello stato utente, sia nello stato supervisore

Inoltre riserva un'area di memoria (appartenente al nucleo) per il vettore di interruzione e per lo stack del nucleo. Il vettore di interruzione contiene, per ogni interruzione, un indirizzo nell'area di memoria del nucleo e una parola di stato.

- Al riconoscimento di un'interruzione, l'hardware salva i registri generali e speciali nello stack del nucleo e salta alla funzione di servizio dell'interruzione.
- Nella fase di esecuzione dell'istruzione IRET, l'hardware ripristina **tutti** i registri dallo stack del nucleo.

A un certo tempo è in esecuzione il processo P1 che esegue una chiamata di sistema con l'istruzione SVC. La primitiva eseguita con questo meccanismo sospende il processo P1 e fa passare in esecuzione il processo P2, che si trova nello stato di pronto.

All'istante in cui viene estratta l'istruzione SVC, i registri del processore, i descrittori di P1 e P2 e lo stack del nucleo hanno i contenuti mostrati in figura, che mostra anche il contenuto dell'elemento del vettore di interruzione associato alle interruzioni da programma.

Si chiede:

1) Lo stato del processore dopo l'esecuzione della SVC;
2) Il contenuto dei registri, dei descrittori e dello stack del nucleo dopo l'esecuzione della SVC;
3) Il contenuto dei registri, dei descrittori e dello stack del nucleo subito prima dell'esecuzione della IRET;
4) Il contenuto dei registri, dei descrittori e dello stack del nucleo subito dopo l'esecuzione della IRET;
5) Lo stato del processore dopo l'esecuzione della IRET

Descrittore di P1	
.....	.....
PC	AAAA
PS	BBBB
SP	CCCC
R1	DDDD
R2	EEEE

Descrittore di P2	
.....	.....
PC	3333
PS	2222
SP	1111
R1	0001
R2	0002

Stack del nucleo	
0FFF	ABCD
1000	
1001	
1002	
1003	
1004	
1005	

Registri	
PC	2000
PS	3000
SP	4000
R1	5555
R2	6666

INDIRIZZO	1000
PAROLA DI STATO	00FF
Vettore di interruzione	

Stack pointer del nucleo	0FFF

SOLUZIONE

1) Stato del processore: SUPERVISORE

Nelle tabelle seguenti: riportare solo i contenuti che cambiano

2)

Descrittore di P1	
.....	.....
PC	invariato
PS	invariato
SP	invariato
R1	invariato
R2	invariato

Descrittore di P2	
.....	.....
PC	invariato
PS	Invariato
SP	Invariato
R1	Invariato
R2	invariato

Stack del nucleo	
0FFF	ABCD
1000	2000
1001	3000
1002	4000
1003	5555
1004	6666

Registri	
PC	1000
PS	00FF
SP	1004
R1	??
R2	??

3)

Descrittore di P1	
.....	.....
PC	2000
PS	3000
SP	4000
R1	5555
R2	6666

Descrittore di P2	
.....	.....
PC	invariato
PS	Invariato
SP	Invariato
R1	Invariato
R2	invariato

Stack del nucleo	
0FFF	ABCD
1000	3333
1001	2222
1002	1111
1003	0001
1004	0002

Registri	
PC	?
PS	00FF
SP	1004
R1	??
R2	??

4)

Descrittore di P1	
.....	.....
PC	invariato
PS	Invariato
SP	Invariato
R1	Invariato
R2	invariato

Descrittore di P2	
.....	.....
PC	invariato
PS	Invariato
SP	Invariato
R1	Invariato
R2	invariato

Stack del nucleo	
0FFF	ABCD
1000	
1001	
1002	
1003	
1004	

Registri	
PC	3333
PS	2222
SP	1111
R1	0001
R2	0002

5) Stato del processore: UTENTE

SISTEMI OPERATIVI, CORSI A e B - QUINTO APPELLO - 20/7/2006

ESERCIZIO A-2 (4 punti)
Si considerino tre thread di uno stesso processo (A,B e C) implementati a livello utente. I thread, che sono eseguiti senza prerilascio, cooperano eseguendo il seguente codice:

```
1.     while (true) {
2.        TSL R1, X
3.        while (R1==1) {
4.              thread_yield();
5.              TSL R1,X}

6.        <accede al buffer condiviso...>
7.        thread_yield();
8.        <continua l'accesso al buffer condiviso...>

9.        X=0
10.       thread_yield();
11.       <altre elaborazioni...>
12.       <altre elaborazioni...>
13.       <altre elaborazioni...>
14.       <altre elaborazioni...>
15.    }
```

dove X è una variabile condivisa, che inizialmente ha il valore 0, e R1 è un registro del processore.
L'istruzione TSL, indivisibile, copia il contenuto di X in R1 e scrive 1 in X.
Supponendo che al momento della loro creazione:

- i thread iniziano ad eseguire il codice dalla riga 1
- il thread A è in esecuzione
- i thread B e C sono nella coda pronti, B in testa e C in coda

e supponendo che ogni riga sia eseguita in un tempo pari a 1 millisecondo, dire in quali istanti avviene una commutazione di contesto e quale thread viene messo in esecuzione nell'intervallo 0-40.

SOLUZIONE

Tempo	0	5	9	13	16	20	23	32	35	39				
Thread in esecuzione	A	B	C	A	B	C	A	B	C	A				

ESERCIZIO A-3 (3 punti)

In un sistema operativo ad ambiente locale, i processi A1, A2, ... ,Am (clienti) e i processi B1,B2,…,Bn (serventi) comunicano mediante primitive di comunicazione.
Per l'invio dei messaggi il sistema fornisce le primitive:

- *send(messaggio, destinatario)* che è asincrona e specifica il nome del processo destinatario.
- *send(messaggio, mailbox)* che è asincrona e specifica il nome della mailbox sulla quale depositare il messaggio.

Per la ricezione dei messaggi il sistema fornisce le primitive:

- *receive(messaggio, mittente)* che è bloccante e specifica il nome del processo dal quale si vuole ricevere un messaggio
- *receive(messaggio, mailbox)* che è bloccante, non specifica il mittente e riceve da una mailbox un messaggio di qualsiasi processo.
- *receive(messaggio)* che è bloccante, non specifica il mittente e riceve un messaggio di qualsiasi processo.

I messaggi contengono i campi *mitt* e *info*, il primo dei quali è il nome del mittente.
La richiesta di un generico cliente può essere servita da un servente arbitrario che poi invia la risposta al cliente.
Il generico cliente si sincronizza con un servente arbitrario e poi aspetta la risposta. Il processo servente che riceve la richiesta del cliente Ai la serve e quindi si sincronizza con Ai per inviare la risposta.
Completare il codice del generico processo cliente Ai e del generico processo servente Bj.

SOLUZIONE

Ai:

```
while (true) {
      mess = produce_mess();
      send(mess,mailbox);
      receive(risp);
      <consuma risposta>
}
```

Bj:

```
while (true) {
      receive(mess,mailbox);
      mittente=estrazione_mittente(mess);
      risposta= produci_risposta();
      send(risposta,mittente);
}
```

SISTEMI OPERATIVI, CORSI A e B - QUINTO APPELLO - 20/7/2006

ESERCIZIO A-4 (2 punti)

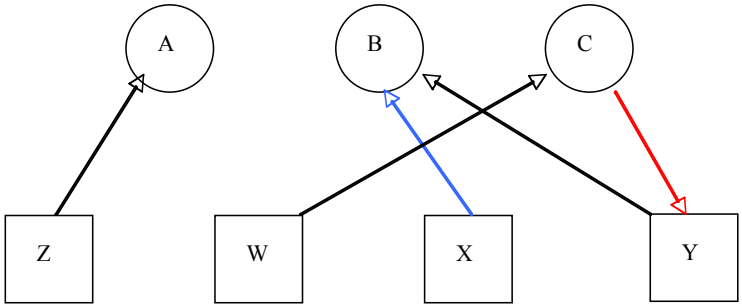
Un sistema con 3 processi (A, B e C) dispone di quattro risorse singole (X, Y, W, Z). Si supponga che a partire dal tempo iniziale avvenga la seguente sequenza di assegnazione/rilascio delle risorse:

1. A richiede X
2. B richiede Y
3. C richiede W
4. A richiede Z
5. B richiede X
6. C richiede Y
7. A rilascia X
8. B richiede Z
9. A richiede W

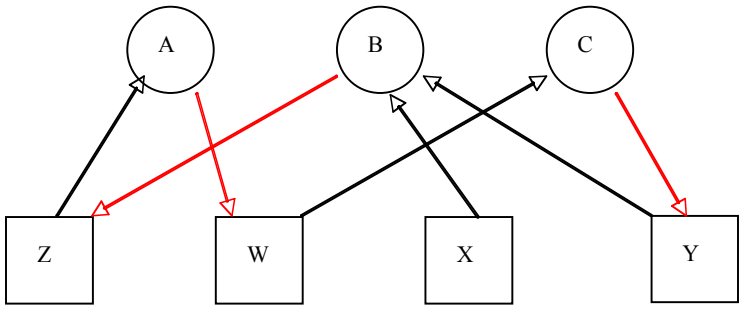
Tracciare il grafo di allocazione ai tempi 7 e 9 e dire se al termine della sequenza il sistema è in stallo.

SOLUZIONE

T=7



T=9



Il sistema è in stallo.

ESERCIZIO A- 5 (2 punti)

Dire quali delle seguenti affermazioni sono vere o false:

I thread di uno stesso processo implementati a livello utente non possono usare i semafori per sincronizzarsi	VERO
I thread di uno stesso processo implementati a livello utente non possono usare i semafori per gestire la mutua esclusione	VERO
I thread implementati a livello utente non possono invocare le chiamate di sistema	FALSO
I thread implementati a livello utente non hanno un descrittore di thread	FALSO
I thread implementati a livello del nucleo non hanno un proprio stack	FALSO
Il descrittore dei thread implementati a livello del nucleo contiene un puntatore all'area di memoria assegnata	FALSO

SISTEMI OPERATIVI, CORSI A e B - QUINTO APPELLO - 20/7/2006

Parte B

ESERCIZIO B-1 (4 punti)

Un disco con 2 facce, 100 settori per traccia e 100 cilindri ha un tempo di seek (proporzionale al numero di cilindri attraversati) pari a 1 ms per ogni cilindro. Il periodo di rotazione è di 3 msec: di conseguenza il tempo impiegato per percorrere un settore è 0,03 msec.
Inoltre si fanno le seguenti ipotesi :

- Nel caso peggiore, il tempo necessario per raggiungere il settore desiderato dopo un'operazione di seek è pari a un periodo di rotazione (3 msec);
- il disco dispone di un buffer della capacità di un settore.

Al tempo 0 la testina ha appena attraversato il settore 30 del cilindro 88 e sono pendenti le seguenti richieste di lettura o scrittura:

- cilindro 60 : settori 15 della faccia 0 e settore 16 della faccia 1
- cilindro 1 : settore 3 della faccia 1
- cilindro 88 : settore 5 della faccia 0

Successivamente arrivano i seguenti comandi:

- al tempo 2: cilindro 90 settore 7 della faccia 0.
- al tempo 30: cilindro 68 settore 12 della faccia 1.
- al tempo 55: cilindro 30 settore 40 della faccia 0.

Lo scheduling è effettuato con politica SSF. Calcolare il tempo necessario per eseguire tutte le operazioni. Il tempo di esecuzione di ogni operazione è uguale alla somma dell'eventuale tempo di *seek,* del ritardo rotazionale (tempo necessario per raggiungere il settore indirizzato) e del tempo di percorrenza del settore indirizzato. Quando si raggiunge un cilindro, i comandi pendenti devono essere eseguiti nell'ordine in cui sono elencati.
Per il ritardo rotazionale dopo un'operazione di *seek* si assume sempre il valore di caso peggiore, pari a un periodo di rotazione.

SOLUZIONE

op. su cilindro: 88 ; settore: 5
inizio: 0 ; seek: 0 ; rotazione: 2,22 ; percorrenza: 0,03 ; fine: 2,25

op. su cilindro: 90 ; settore: 7
inizio: 2,25 ; seek: 2 ; rotazione: 3 ; percorrenza: 0,03 ; fine: 7,28

Arrivato comando su cilindro 90

op. su cilindro: 60 ; settore: 15
inizio: 7,28 ; seek: 30 ; rotazione: 3 ; percorrenza: 0,03 ; fine: 40,31

op. su cilindro: 60 ; settore: 16
inizio: 40,31 ; seek: 0 ; rotazione: 0 ; percorrenza: 0,03 ; fine: 40,34

Arrivato comando su cilindro 68

op. su cilindro: 68 ; settore: 12
inizio: 40,34 ; seek: 8 ; rotazione: 3 ; percorrenza: 0,03 ; fine: 51,37

op. su cilindro: 1 ; settore: 3
inizio: 51,37 ; seek: 67 ; rotazione: 3 ; percorrenza: 0,03 ; fine: 121,4

Arrivato comando su cilindro 30

op. su cilindro: 30 ; settore: 40
inizio: 121,4 ; seek: 29 ; rotazione: 3 ; percorrenza: 0,03 ; fine: 153,43

SISTEMI OPERATIVI, CORSI A e B - QUINTO APPELLO - 20/7/2006

ESERCIZIO B-2 (4 punti)

In un sistema operativo che utilizza la rilocazione statica e gestisce la memoria con partizioni fisse, la memoria fisica ha un'ampiezza di 16 Mbyte ed è configurata con le seguenti partizioni:

- Partizione 1, ampiezza 1 Mbyte, riservata al Sistema Operativo;
- Partizione 2, ampiezza 1 Mbyte, disponibile per processi di ampiezza a con $a \leq 1$ Mbyte;
- Partizione 3, ampiezza 2 Mbyte, disponibile per processi di ampiezza a con $1 < a \leq 2$ Mbyte;
- Partizione 4, ampiezza 4 Mbyte, disponibile per processi di ampiezza a con $2 < a \leq 4$ Mbyte;
- Partizione 5, ampiezza 8 Mbyte, disponibile per processi di ampiezza a con $4 < a \leq 8$ Mbyte.

A partire dal tempo t=0 vengono generati nell'ordine i seguenti processi:

- t= 0: Processo P1, che occupa 1,7 Mbyte con tempo di completamento di 4 sec;
- t= 1: Processo P2, che occupa 0,9 Mbyte con tempo di completamento di 5 sec;
- t= 2: Processo P3, che occupa 4,7 Mbyte con tempo di completamento di 6 sec;
- t= 3: Processo P4, che occupa 5,8 Mbyte con tempo di completamento di 2 sec;
- t= 4: Processo P5, che occupa 0,1 Mbyte con tempo di completamento di 7 sec;
- t= 5: Processo P6, che occupa 7,2 Mbyte con tempo di completamento di 3 sec;
- t= 6: Processo P7, che occupa 1,5 Mbyte con tempo di completamento di 1 sec;
- t= 7: Processo P8, che occupa 1,9 Mbyte con tempo di completamento di 8 sec;

Il caricamento dei processi nelle rispettive partizioni avviene con politica FIFO, e l'istante di caricamento coincide con la terminazione del processo in esecuzione. Una volta caricato in memoria, ogni processo mantiene l'assegnazione fino alla sua terminazione. Si trascura il tempo necessario per il caricamento in memoria dei processi.
I processi caricati in memoria (incluso l'eventuale processo appena caricato) vengono schedulati con politica Shortest Job First (SJF) e si suppone che i processi, una volta in esecuzione, mantengano lo stato di esecuzione fino alla terminazione, senza mai sospendersi.
Si chiede a quale tempo avvengono il caricamento in memoria e l'uscita dal sistema di ogni processo.

SOLUZIONE

- Processo P1 Caricato a t= 0 Nella partizione 3 Inizia esec. a t= 0 Esce dal sistema a t= 4
- Processo P2 Caricato a t= 1 Nella partizione 2 Inizia esec. a t= 4 Esce dal sistema a t= 9
- Processo P3 Caricato a t= 2 Nella partizione 5 Inizia esec. a t= 10 Esce dal sistema a t= 16
- Processo P4 Caricato a t= 16 Nella partizione 5 Inizia esec. a t= 16 Esce dal sistema a t= 18
- Processo P5 Caricato a t= 9 Nella partizione 2 Inizia esec. a t= 21 Esce dal sistema a t= 28
- Processo P6 Caricato a t= 18 Nella partizione 5 Inizia esec. a t= 18 Esce dal sistema a t= 21
- Processo P7 Caricato a t= 6 Nella partizione 3 Inizia esec. a t= 9 Esce dal sistema a t= 10
- Processo P8 Caricato a t= 10 Nella partizione 3 Inizia esec. a t= 28 Esce dal sistema a t= 36

ESERCIZIO B-3 (3 punti)

In un file system di tipo Unix i blocchi hanno dimensione di 2 KB e i puntatori ai blocchi sono codificati con 32 bit (4 byte). Gli i-node contengono 15 puntatori diretti, due puntatori indiretti singoli, due puntatori indiretti doppi e due puntatori indiretti tripli.
Calcolare:

1. il massimo numero di byte indirizzabili con i soli puntatori diretti
2. il massimo numero di byte indirizzabili con i soli puntatori indiretti singoli
3. il massimo numero di byte indirizzabili con i soli puntatori indiretti doppi
4. il massimo numero di byte indirizzabili con i soli puntatori indiretti tripli
5. la massima dimensione (in byte) di un file
6. la massima dimensione (in blocchi) di un file

SOLUZIONE

I blocchi indiretti contengono $2^{11}/4 = 2^9 = 512$ indirizzi.

1. il massimo numero di byte indirizzabili con i soli puntatori diretti: 30 KB
2. il massimo numero di byte indirizzabili con i soli puntatori indiretti singoli: 2 MB
3. il massimo numero di byte indirizzabili con i soli puntatori indiretti doppi: 1 GB
4. il massimo numero di byte indirizzabili con i soli puntatori indiretti tripli: 512 GB
5. la massima dimensione (in byte) di un file: 513GB+ 2MB+ 30 KB
6. la massima dimensione (in blocchi) di un file: 268960783

SISTEMI OPERATIVI, CORSI A e B - QUINTO APPELLO - 20/7/2006

ESERCIZIO B-4 (2 punti)

Si consideri un sistema dove gli indirizzi logici hanno la lunghezza di 32 bit e le pagine logiche e fisiche hanno ampiezza di 2 kByte.Per la gestione della memoria con paginazione dinamica si utilizzano tabelle delle pagine a 2 livelli. La dimensione della tabella di primo livello è di 2^{10} elementi.
Negli elementi di ogni tabella di primo o secondo livello, che occupano 4 byte, 8 bit sono riservati agli indicatori (pagina caricata, riferita, modificata, ecc.) mentre i rimanenti rappresentano l'indice di un blocco fisico.
Si chiede:

1. la lunghezza del campo offset;
2. la lunghezza (numero di elementi) di ogni tabella di secondo livello;
3. lo spazio occupato in memoria dalla tabella di primo livello e da ogni tabella di secondo livello (numero di byte);
4. la massima dimensione della memoria fisica (numero di blocchi e di byte, espressi come potenze di 2).

SOLUZIONE

1. lunghezza del campo offset: 11 bit
2. lunghezza (numero di elementi) di ogni tabella di secondo livello: 2^{11}
3. spazio occupato in memoria dalla tabella di primo livello : $2^{10}*2^2 = 2^{12}$ (4 kByte)
4. spazio occupato in memoria da ogni tabella di secondo libello : $2^{11}*2^2 = 2^{13}$ (8 kByte)
5. massima dimensione della memoria fisica: blocchi 2^{24} ; byte. $2^{24}*2^{11} = 2^{35}$ (32 GByte)

ESERCIZIO B-5 (2 punti)

In un disco della capacità di 4 Gbyte (= 2^{32} byte) con blocchi di 4 Kbyte (= 2^{12} byte), è definito un file system FAT. Gli elementi della FAT sono in corrispondenza biunivoca con i blocchi fisici del disco. Ogni file è descritto da una lista concatenata di indirizzi di blocchi, realizzata sulla FAT. Ogni elemento della FAT è formato da un numero intero di byte e precisamente dal minimo numero di byte necessari per indirizzare l'intero disco.

1. Qual è la dimensione della FAT?
2. Qual è la lunghezza in byte di ogni elemento della FAT?
3. Qual è la lunghezza della FAT, espressa in byte?

SOLUZIONE

1. Dimensione della FAT: $2^{32} / 2^{12} = 2^{20}$ elementi;
2. I blocchi del disco sono 2^{20} , indirizzabili con 20 bit. Pertanto la lunghezza degli elementi della FAT è di 3 byte;
3. Numero di byte occupati dalla FAT: $3* 2^{20} = 3$ Mbyte