SISTEMI OPERATIVI, CORSI A e B - SECONDA VERIFICA INTERMEDIA - 31/5/2006

COGNOME .....NOME ..... MATRICOLA .....

ESERCIZIO A-1 (4 punti)

Si consideri un processore che dispone dei seguenti registri:
- i registri speciali PC (program counter) e PS (program status)
- un banco di registri generali, utilizzati sia in stato utente, sia in stato supervisore, che comprende i registri R1, R2, R3, R4 e lo stack pointer SP,

Il nucleo dispone di un proprio stack, che inizia all'indirizzo 1016 e si espande verso il basso, e di un proprio stack pointer SP' (valore 1017 quando lo stack è vuoto).
Il sistema realizza i thread a livello kernel e lo schedulatore gestisce esclusivamente i thread. Al tempo t sono presenti nel sistema (tra gli altri) il thread T_{ip} del processo P_i, che si trova in stato di esecuzione, e il thread T_{jq} del processo P_j, che occupa la prima posizione della Coda Pronti. Al tempo t, il thread T_{ip} esegue l'istruzione *SVC* per attivare la primitiva *ThreadYield*.
Immediatamente dopo il riconoscimento dell'interruzione generata dalla *SVC*, i registri del processore, i descrittori di T_{ip} e T_{jq} hanno i contenuti mostrati in figura. Lo stack del nucleo è vuoto e il puntatore SP' ha il valore 1017.
L'interruzione determina l'intervento del nucleo, che esegue la primitiva *ThreadYield* ,che a sua volta attiva lo scheduler.
Supponendo che il vettore di interruzione associato alla *SVC* sia 0425 e che la parola di stato del nucleo sia 275E, 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 primitiva;
b) il contenuto dei descrittori, dei registri generali e speciali e dello stack del nucleo immediatamente prima dell'attivazione dello scheduler, all'interno della primitiva;
c) 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 primitiva;
d) 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 T_{ip}	
Stato	Esec
PC	A12C
PS	16F2
SP	A275
R1	25CC
R2	0000
R3	0056
R4	AA38

DESCRITTORE DI T_{Jq}	
Stato	Pronto
PC	2E31
PS	16F2
SP	2873
R1	1234
R2	56CC
R3	0000
R4	0000

STACK DEL NUCLEO	
1017	----------
1016	
1015	
1014	
1013	
1012	
1011	
1010	
…….	

REGISTRI GENERALI	
SP	2997
R1	6649
R2	02CE
R3	D410
R4	73FF

PROCESSORE: Registri speciali				
PC	2F00		PS	16F2

STACK POINTER	
SP'	1017

SOLUZIONE

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

DESCRITTORE DI T_{ip}	
Stato	Esec
PC	Invariato
PS	Invariato
SP	Invariato
R1	Invariato
R2	Invariato
R3	Invariato
R4	Invariato

DESCRITTORE DI T_{Jq}	
Stato	Pronto
PC	Invariato
PS	Invariato
SP	Invariato
R1	Invariato
R2	Invariato
R3	Invariato
R4	Invariato

STACK DEL NUCLEO	
1017	----------
1016	2F00
1015	16F2
1014	
1013	
1012	
1011	
1010	

REGISTRI GENERALI	
SP	Invariato
R1	Invariato
R2	Invariato
R3	Invariato
R4	Invariato

PROCESSORE: Registri speciali				
PC	0425		PS	275E

STACK POINTER	
SP'	1015

COGNOME .....NOME ..... MATRICOLA .....

b) contenuto dei descrittori, dei registri generali e speciali e dello stack del nucleo immediatamente prima dell'attivazione dello scheduler, all'interno della primitiva;

DESCRITTORE DI T_{ip}		DESCRITTORE DI T_{Jq}		STACK DEL NUCLEO		REGISTRI GENERALI	
Stato	Esecuzione	Stato	Pronto	1017	----------	SP	Invariato
PC	2F00	PC	Invariato	1016	2F00	R1	??
PS	16F2	PS	Invariato	1015	16F2	R2	??
SP	2997	SP	Invariato	1014		R3	??
R1	6649	R1	Invariato	1013		R4	??
R2	02CE	R2	Invariato	1012			
R3	D410	R3	Invariato	1011			
R4	73FF	R4	Invariato	1010			
PROCESSORE: Registri speciali				STACK POINTER			
PC	0425 + ??	PS	275E	SP'	1015 + ??		

c) 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 primitiva;

DESCRITTORE DI T_{ip}		DESCRITTORE DI T_{Jq}		STACK DEL NUCLEO		REGISTRI GENERALI	
Stato	Pronto	Stato	Esecuzione	1017	----------	SP	2873
PC	Invariato	PC	Invariato	1016	2E31	R1	1234
PS	Invariato	PS	Invariato	1015	16F2	R2	56CC
SP	Invariato	SP	Invariato	1014		R3	0000
R1	Invariato	R1	Invariato	1013		R4	0000
R2	Invariato	R2	Invariato	1012			
R3	Invariato	R3	Invariato	1011			
R4	Invariato	R4	Invariato	1010			
PROCESSORE: Registri speciali				STACK POINTER			
PC	0425+ ??	PS	275E	SP'	1015		

d) 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 T_{ip}		DESCRITTORE DI T_{Jq}		STACK DEL NUCLEO		REGISTRI GENERALI	
Stato	Pronto	Stato	Esecuzione	1017	----------	SP	Invariato
PC	Invariato	PC	Invariato	1016		R1	Invariato
PS	Invariato	PS	Invariato	1015		R2	Invariato
SP	Invariato	SP	Invariato	1014		R3	Invariato
R1	Invariato	R1	Invariato	1013		R4	Invariato
R2	Invariato	R2	Invariato	1012			
R3	Invariato	R3	Invariato	1011			
R4	Invariato	R4	Invariato	1010			
PROCESSORE: Registri speciali				STACK POINTER			
PC	2E31	PS	16F2	SP'	1017		

COGNOME .....NOME ..... MATRICOLA .....

ESERCIZIO A-2 (4 punti)

Si considerino due thread linux A e B che scambiano messaggi attraverso un buffer condiviso a n posizioni, numerate da 0 a n-1. Ogni messaggio occupa una posizione del buffer. A tal fine A e B condividono un semaforo di mutua esclusione *mux* (inizializzato a 1), due variabili di condizione *NonPieno* e *NonVuoto* e una variabile *n_elem*, che rappresenta il numero di posizioni occupate nel buffer, e si sincronizzano con funzioni della libreria *pthreads.*
L'interazione tra i thread A e B si svolge nel modo seguente:

Thread A:

```
    while (true) {
    <produce il messaggio  mess >
    < se il buffer è pieno si sospende in attesa di un prelievo>
    <deposita mess nel buffer>
    n_elem++;
    < se n_elem==1 segnala di aver depositato, per l'eventuale riattivazione del thread B>
}
```

Thread B:

```
    while (true) {
    < se il buffer è vuoto si sospende in attesa di un deposito>
    <preleva un messaggio m >
    n_elem--;
    < se n_elem==n-2 segnala di aver prelevato, per l'eventuale riattivazione del thread A>
    <consuma m >
}
```

Inserire nel precedente schema le funzioni **pthread_mutex_lock**, **pthread_mutex_unlock**, **pthread_cond_wait**, **pthread_cond_signal** con i rispettivi argomenti.

SOLUZIONE

Thread A:

```
    a.1) while (true) {
    a.2) <produce  v>
    a.3) pthread_mutex_lock(&mux);
    a.4) if (n_elem == n-1) pthread_cond_wait(&NonPieno,&mux);
    a.5) <deposita v nel buffer>
    a.6) n_elem++;
    a.7) if (n_elem == 1) pthread_cond_signal(&NonVuoto);
    a.8) pthread_mutex_unlock(&mux);
}
```

Thread B:

```
    b.1) while (true) {
    b.2) pthread_mutex_lock(&mux);
    b.3) if (n_elem == 0) pthread_cond_wait(&NonVuoto,&mux);
    b.4) <preleva un valore v dal buffer>
    b.5) n_elem--;
    b.6) if (n_elem == n-2) pthread_cond_signal(&NonPieno);
    b.7) pthread_mutex_unlock(&mux);
    b.8) <consuma v>
}
```

COGNOME .....NOME ..... MATRICOLA .....

ESERCIZIO A-3 (3 punti)

In un sistema vengono generati 6 processi (A,B,C,D,E), con i tempi di arrivo, le priorità e le durate (in millisecondi) sotto specificate:

PROCESSO	TEMPO DI ARRIVO	PRIORITÀ	DURATA
A	0	2	10
B	36	3	7
C	8	1	24
D	18	3	6
E	28	4	10
F	32	2	11

Lo scheduling del processore avviene con una politica a prorità (che assegna il processore al processo che ha il valore più elevato di priorità e a pari priorità, al processo arrivato per primo) e con prerilascio. Si suppone che, una volta in esecuzione, ogni processo avanzi senza mai sospendersi.
Riempire la seguente tabella, utilizzando una riga per ogni evento che provoca la riassegnazione del processore e specificando, in ogni riga, il processo in esecuzione e la composizione della coda pronti subito dopo il verificarsi dell'evento.

Soluzione

TEMPO	EVENTO	PROC. IN ESEC.	CODA PRONTI	NOTE
0	Arriva A	A	Ø	
8	Arriva C	A	C	
10	Termina A	C	Ø	
18	Arriva D	D	C	C: tempo residuo 16
24	Termina D	C	Ø	
28	Arriva E	E	C	C: tempo residuo 12
32	Arriva F	E	F → C	
36	Arriva B	E	B → F → C	
38	Termina E	B	F → C	
45	Termina B	F	C	
56	Termina F	C	Ø	
68	Termina C	Ø	Ø	

ESERCIZIO A-4 (2 punti)

Si consideri un sistema nel quale sono definiti il semaforo *sem* e i processi P1 (con priorità 1), P2 (con priorità 1) e P3 (con priorità 2). Lo scheduling avviene con una politica a priorità, che prevede il prerilascio e assegna il processore al processo pronto di priorità più elevata (a pari priorità applica la politica FIFO). La politica applicata al semaforo è la FIFO.
Al tempo *t* il processo P1 è in esecuzione il processo P2 è pronto; inoltre il semaforo *sem* ha valore 0 e la sua coda contiene P3. Dopo il tempo *t* si verifica la seguente sequenza di eventi:

1) P1 esegue *wait (sem))*
2) il processo in esecuzione esegue *signal(sem);*
3) il processo in esecuzione esegue *wait (sem)*
4) il processo in esecuzione esegue *signal(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

Sequenza di eventi	In Esecuzione	Coda Pronti	Valore di *sem*	Coda di *sem*
1) P1 esegue *wait (sem)*	P2	∅	0	P3 → P1
2) il processo in esecuzione esegue *signal(sem)*	P3	P2	0	P1
3) il processo in esecuzione esegue *wait (sem)*	P2	∅	0	P1 → P3
4) Il processo in esecuzione esegue *signal(sem)*	P2	P1	0	P3

COGNOME .....NOME ..... MATRICOLA .....

ESERCIZIO A.5 (2 punti)

Dati tre processi A,B,C che osservano il paradigma di "possesso e attesa" e quattro risorse singole Q, R, S, utilizzabili in mutua esclusione e senza possibilità di prerilascio, supponiamo che il gestore assegni le risorse al processo richiedente alla sola condizione che la risorsa sia disponibile. Inizialmente tutte le risorse sono disponibili.
Si considerino la seguente sequenza di richieste e rilasci:

1) A richiede Q; 2) C richiede S; 3) C richiede Q 4) B richiede R; 5) B richiede S; 6) A rilascia Q;	7) A richiede S; 8) C richiede R; 9) A rilascia S; 10) C rilascia Q 11) B rilascia R

Mostrare come si evolve il sistema utilizzando la tabella riportata nello schema di soluzione, e rispondere alle seguenti domande:

a. la sequenza può essere interamente eseguita?
b. eventualmente, quali operazioni non possono essere eseguite e perché?
c. si raggiunge uno stallo?
d. eventualmente, quale operazione porta il sistema in stallo?

SOLUZIONE

Evoluzione del sistema:

Operazione	Eseguibile?	Processo A		Processo B		Processo C	
		Sospeso?	Risorse Assegnate	Sospeso?	Risorse Assegnate	Sospeso?	Risorse Assegnate
1	SI	NO	Q	NO	Ø	NO	Ø
2	SI	NO	Q	NO	Ø	NO	S
3	SI	NO	Q	NO	Ø	SI aspetta Q	S
4	SI	NO	Q	NO	R	SI	S
5	SI	NO	Q	SI aspetta S	R	SI	S
6	SI	NO	Ø	SI	R	NO	S, Q
7	SI	SI aspetta S	Ø	SI	R	NO	S, Q
8	SI	SI	Ø	SI	R	SI aspetta R	S, Q
9	NO	SI	Ø	SI	R	SI	S, Q
10	NO	SI	Ø	SI	R	SI	S, Q
11	NO	SI	Ø	SI	R	SI	S, Q

Quindi:

a. La sequenza non può essere interamente eseguita
b. le operazioni 9), 10) , 11) non possono essere eseguite, perchè i processi che dovrebbero eseguirle sono sospesi;
c. La sequenza provoca uno stallo
d. Lo stallo si raggiunge per effetto dell'operazione 8.

ESERCIZIO B.1 (4 PUNTI)

In un sistema che gestisce la memoria con paginazione a domanda sono presenti i processi A, B, C, D.
Lo stato di occupazione della memoria al tempo 10 è descritto dalla tabella *Core Map* . Gli elementi di questa tabella (in corrispondenza con i blocchi fisici) hanno i campi *Proc* (processo a cui è assegnato il blocco; il campo è vuoto se il blocco è libero); *Pag* (pagina del processo caricata nel blocco), *t* (tempo dell'ultimo riferimento alla pagina). I primi 6 blocchi sono riservati al sistema operativo.
Le tabelle delle pagine dei processi A e B sono mostrate in figura.
Per la gestione della memoria si utilizza un algoritmo di sostituzione LRU locale. Il *working set* assegnato ai processi A e B (inteso come numero di blocchi a disposizione del processo, anche se momentaneamente non occupati) ha dimensione 4 e contiene rispettivamente i blocchi: 6,9,14,22 per il processo A e i blocchi 7,8,19,20 per il processo B.

Proc								A	B	B	A	D		C		A	C	D		C	B		C	A	
Pag								5	5	3	3	0		3		7	5	6		9	2		8	6	
t								2	5	6	3	1		3		6	4	10		5	9		8	5	
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

Core Map al tempo 10

COGNOME .....NOME ..... MATRICOLA .....

Pagina	Blocco
0	-
1	-
2	-
3	9
4	-
5	6
6	22
7	14
Proc. A t= 10	

Pagina	Blocco
0	-
1	-
2	19
3	8
4	-
5	7
6	-
7	-
Proc. B t= 10	

Il processo A riferisce la pagina 3 al tempo 11, la pagina 2 al tempo 12, la pagina 0 al tempo 13 e la pagina 7 al tempo 14;
Il processo B riferisce la pagina 4 al tempo 15 e la pagina 0 al tempo 16.
Come si modificano i contenuti della *Core Map* e delle tabelle delle pagine dei processi A e B ai tempi 11, 12, 13, 14, 15 e 16 ?

SOLUZIONE

Core Map:
t= 11 elemento 9 nuovi valori dei campi: Proc = A ; Pag = 3 ; t = 11 ;
t= 12 elemento 6 nuovi valori dei campi: Proc = A ; Pag = 2 ; t = 12 ;
t= 13 elemento 22 nuovi valori dei campi: Proc = A ; Pag = 0 ; t = 13 ;
t= 14 elemento 14 nuovi valori dei campi: Proc = A ; Pag = 7 ; t = 14 ;
t= 15 elemento 20 nuovi valori dei campi: Proc = B ; Pag = 4 ; t = 15 ;
t= 16 elemento 7 nuovi valori dei campi: Proc = B ; Pag = 0 ; t = 16 ;

Proc								A	B	B	A	D		C		A	C	D		C	B	B	C	A	
Pag								2	0	3	3	0		3		7	5	6		9	2	4	8	0	
t								12	16	6	11	1		3		14	4	10		5	9	15	8	13	
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	

Core Map al tempo 16

Tabelle delle pagine

Pagina	Blocco
0	-
1	-
2	-
3	9
4	-
5	6
6	22
7	14
Proc. A t= 11	

Pagina	Blocco
0	-
1	-
2	6
3	9
4	-
5	-
6	22
7	14
Proc. A t= 12	

Pagina	Blocco
0	22
1	-
2	6
3	9
4	-
5	-
6	-
7	14
Proc. A t= 13	

Pagina	Blocco
0	22
1	-
2	6
3	9
4	-
5	-
6	-
7	14
Proc. A t= 14	

Pagina	Blocco
0	-
1	-
2	19
3	8
4	20
5	7
6	-
7	-
Proc. B t= 15	

Pagina	Blocco
0	7
1	-
2	19
3	8
4	20
5	-
6	-
7	-
Proc. B t= 16	

ESERCIZIO B.2 (4 punti)

Un disco con 2 facce, 250 settori per traccia e 150 cilindri ha un tempo di seek proporzionale al numero di cilindri attraversati e pari a 0,2 ms per ogni cilindro. Il periodo di rotazione è di 25 msec: conseguentemente il tempo impiegato per percorrere un settore è di 0,1 msec.
A un certo tempo (convenzionalmente indicato come t= 0) termina l'esecuzione dei comandi sul cilindro 111 e sono pervenute, nell'ordine, le seguenti richieste di lettura o scrittura:

- clindro 90, faccia 0, settore 100
- clindro 4, faccia 1, settore 120

- clindro 148, faccia 1, settore 5

Al tempo 70 arrivano tre ulteriori comandi di lettura:

- clindro 4, faccia 0, settore 120
- clindro 90, faccia 0, settore 60
- clindro 80, faccia 0, settore 60

Calcolare il tempo necessario per eseguire tutte queste operazioni supponendo che si adotti la politica di scheduling SCAN con fase di salita attiva al tempo 0.
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.
Il controllore è dotato di sufficiente capacità di buffering ed è sempre in grado di accettare senza ritardo i dati letti dal disco o quelli da scrivere sul disco.
Per il ritardo rotazionale dopo un'operazione di *seek* si assume sempre il valore di caso peggiore, pari a un intero periodo di rotazione (25 msec). Si assuma inoltre che i comandi sullo stesso cilindro vengano eseguiti in ordine FIFO.

SOLUZIONE

op. su cilindro:	148		5	100				
inizio:	0	seek:	7,4	rotazione:	25	percorrenza:	0,1	fine: 32,5
op. su cilindro:	90	settore:	100					
inizio:	32,5	seek:	11,6	rotazione:	25	percorrenza:	0,1	fine: 69,2
op. su cilindro:	4	settore:	5					
inizio:	69,2	seek:	17,2	rotazione:	25	percorrenza:	0,1	fine: 111,5
op. su cilindro:	4	settore:	120					
inizio:	111,5	seek:	0	rotazione:	11,4	percorrenza:	0,1	fine: 123
op. su cilindro:	80	settore:	5	faccia 1				
inizio:	123	seek:	15,2	rotazione:	25	percorrenza:	0,1	fine: 163,3
op. su cilindro:	90	settore:	5	faccia 0				
inizio:	163,3	seek:	2	rotazione:	25	percorrenza:	0,1	fine: 190,4

ESERCIZIO B.3 (3 punti)

Un file system di tipo FAT contiene la directory `pippo` che include i seguenti files:
`fileaa`, allocato sui blocchi 10,18,13,9,11
`filebb`, allocato sui blocchi 12,17,15,8,7,16
`filecc`, allocato sui blocchi 5,14,6,19,20,4,3

La directory `pippo` a sua volta è contenuta nel blocco 21.

Si chiede:
1) scrivere il contenuto della FAT
2) scrivere il contenuto della directory `pippo`.

SOLUZIONE

La FAT contiene:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
…	…	-1	3	14	19	16	7	11	18	-1	17	9	6	8	-1	15	13	20	4	-1	…

La directory `pippo` contiene:

```
fileaa, attributi del file, 10
filebb, attributi del file, 12
filecc, attributi del file, 5
```

ESERCIZIO B.4 (2 punti)

Si consideri un sistema dove gli indirizzi logici hanno la lunghezza di 40 bit e le pagine logiche e fisiche hanno ampiezza di 4 kByte. Per la gestione della memoria con paginazione a domanda si utilizzano tabelle delle pagine a 2 livelli. La tabella di primo livello comprende 2^{13} elementi.
Gli elementi di ogni tabella di primo o secondo livello occupano 4 byte, all'interno dei quali 4 bit sono riservati agli indicatori mentre i rimanenti codificano un indice di blocco fisico.
Si chiede:

1. la lunghezza del campo offset, in numero di bit;
2. la lunghezza delle tabelle di secondo livello (numero di elementi);
3. lo spazio occupato in memoria dalla tabella di primo livello (numero di byte);
4. lo spazio occupato in memoria da ogni tabella di secondo livello (numero di byte);
5. la massima dimensione della memoria virtuale, in numero di blocchi e di byte.

SOLUZIONE

1. Lunghezza del campo offset: **12** bit;
2. Lunghezza delle tabelle di secondo livello: $\mathbf{2^{15}}$ elementi
3. Spazio occupato in memoria dalla tabella di primo livello: $\mathbf{4*2^{13} = 32K\ byte}$;
4. Spazio occupato in memoria da ogni tabella di secondo livello: $\mathbf{4*2^{15} = 128K\ byte}$;
5. Massima dimensione della memoria fisica $\mathbf{2^{28}}$ blocchi; $\mathbf{2^{40} = 4T}$ byte.

ESERCIZIO B.5 (2 punti)

Si consideri un disco con 700 cilindri, 2 facce e 1000 settori per traccia.

a) Quanti settori contiene il disco?
b) Tradurre i seguenti indici di blocco nelle rispettive terne <cilindro, faccia, settore>:
1.233.991; 825.700; 12.918; 678.356; 1.329.655; 45.987.

SOLUZIONE

a) Il disco contiene 1.400.000 settori
b) Traduzione degli indici di blocco in terne:
cilindro = IndiceDiBlocco **div** (2*1000)
faccia = (IndiceDiBlocco **mod** (2*1000)) **div** 1000
settore = (IndiceDiBlocco **mod** (2*1000)) **mod** 1000

Indice di blocco	cilindro	faccia	settore
1.233.991	616	1	991
825.700	412	1	700
12.918	6	0	918
678.356	339	0	356
1.329.655	664	1	655
45.987	22	1	987