Sistemi Operativi e Laboratorio, Prova del 13/07/2016

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

Esercizio 1 (4 punti)

Si consideri un processore che dispone dei registri speciali PC (program counter) e PS (program status), dello stack pointer SP e dei registri generali R1, R2 e R3. In stato utente, ogni processo dispone di uno stack ad uso generale e di uno stack riservato per gestire le upcall chiamato *Signal Stack* (per semplicità assumiamo che ogni processo abbia un unico thread).

Per predisporre all'esecuzione della upcall in un processo, il nucleo utilizza il Signal Stack per salvare il contesto del processo. La upcall si conclude con l'istruzione RET che ripristina la normale esecuzione del processo.

Al tempo t, il nucleo invia una upcall al processo P che è associata alla funzione di gestione che inizia all'indirizzo 8000 (nello spazio di memoria del processo). Il signal stack del processo comincia alla locazione FF0A, il processo è in stato di pronto e il contenuto dei suoi registri speciali e generali è conservato nel suo descrittore come mostrato in tabella.

DESCRITTORE DI P	
Stato	Pronto
PC	2000
PS	16F2
SP	E000
R1	1111
R2	2222
R3	3333

Mostrare:

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 gestione della upcall;
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 RET con la quale termina la upcall;
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 RET.

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 gestione della upcall:

DESCRITTORE DI P	
Stato	Esecuzione
PC	2000
PS	16F2
SP	E000
R1	1111
R2	2222
R3	3333

SIGNAL STACK DI P	
FF0A	2000
FF09	E000
FF08	1111
FF07	2222
FF06	3333
FF05	
FF04	
FF03	

REGISTRI SP, R1,R2,R3	
SP	FF05
R1	??
R2	??
R3	??

PROCESSORE: Registri speciali e stato							
PC	8000		PS	16F2		Stato	Utente

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 RET con la quale termina la upcall:

DESCRITTORE DI P	
Stato	Esecuzione
PC	2000
PS	16F2
SP	E000
R1	1111
R2	2222
R3	3333

SIGNAL STACK DI P	
FF0A	2000
FF09	E000
FF08	
FF07	
FF06	
FF05	
FF04	
FF03	

REGISTRI SP, R1,R2,R3	
SP	FF08
R1	1111
R2	2222
R3	3333

PROCESSORE: Registri speciali e stato

PC	8000+??		PS	16F2		Stato	Utente

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 RETU.

DESCRITTORE DI P	
Stato	Esecuzione
PC	1010
PS	16F2
SP	E0AA
R1	3031
R2	3042
R3	3053

SIGNAL STACK DI P	
FF0A	
FF09	
FF08	
FF07	
FF06	
FF05	
FF04	
FF03	

REGISTRI SP, R1,R2,R3	
SP	E000
R1	1111
R2	2222
R3	3053

PROCESSORE: Registri speciali e stato							
PC	2000		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 gestione della upcall:

DESCRITTORE DI P	
Stato	
PC	
PS	
SP	
R1	
R2	
R3	

SIGNAL STACK DI P	
FF0A	
FF09	
FF08	
FF07	
FF06	
FF05	
FF04	
FF03	

REGISTRI SP, R1,R2,R3	
SP	
R1	
R2	
R3	

PROCESSORE: Registri speciali e stato							
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 RET con la quale termina la upcall:

DESCRITTORE DI P	
Stato	
PC	
PS	
SP	
R1	
R2	
R3	

SIGNAL STACK DI P	
FF0A	
FF09	
FF08	
FF07	
FF06	
FF05	
FF04	
FF03	

REGISTRI SP, R1,R2,R3	
SP	
R1	
R2	
R3	

PROCESSORE: Registri speciali e stato							
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 RETU.

DESCRITTORE DI P	SIGNAL STACK DI P	REGISTRI SP, R1,R2,R3

Sistemi Operativi e Laboratorio, Prova del 13/07/2016

Stato	
PC	
PS	
SP	
R1	
R2	
R3	

FF0A	
FF09	
FF08	
FF07	
FF06	
FF05	
FF04	
FF03	

SP	
R1	
R2	
R3	

PROCESSORE: Registri speciali e stato							
PC			PS			Stato	

Esercizio 2 (4 punti)

In un sistema operativo, i thread A (produttore) e B (consumatore) cooperano scambiandosi messaggi attraverso un buffer `buf` di `size` celle, ciascuna capace di contenere un messaggio. Lo stato del buffer è definito dalle variabili `ndata` che indica il numero di elementi presenti nel buffer, `last` e `front` che contengono gli indici degli elementi in testa e in coda al buffer, rispettivamente. Si chiede di scrivere la soluzione completando i frammenti rilevanti dei thread A e B nella scheda sottostante. Si utilizzino i meccanismi di lock e variabili di condizione (con semantica Mesa).

A (produttore):	B (consumatore):

While (true) {
item = produce_item();
_____
_____
_____
_____
_____
_____
_____
_____
}
	While (true) {
	_____
	_____
	_____
	_____
	_____
	_____
	_____
	_____
	consume(item);
	}

Variabili di lock usate:_____
variabili di condizione usate:_____

Soluzione

A (produttore):	B (consumatore):

While (true) {
item = produce_item();
lock.acquire();
while (ndata == size)
full.wait(lock);
buf[last % size] = item;
last=(last++)%size;
ndata--;
empty.signal(lock);
lock.release();
}
	While (true) {
	lock.acquire();
	while (ndata == 0)
	empty.wait(lock);
	item = buf[front % size];
	front=(front++)%size;
	ndata--;
	full.signal(lock);
	lock.release();
	consume(item);
	}

Variabili di lock usate: `lock`
variabili di condizione usate: `empty, full`

Sistemi Operativi e Laboratorio, Prova del 13/07/2016

Esercizio 3 (4 punti)

In un sistema che gestisce la memoria con paginazione a domanda, le pagine logiche e i blocchi fisici hanno una lunghezza di 2^6 byte e gli indirizzi logici hanno una lunghezza di 14 bit.
Per la gestione della memoria si utilizzano tabelle delle pagine a 2 livelli. Sia la tabella di primo livello che quelle di secondo livello contengono 2^4 elementi di 4 byte: pertanto ogni tabella occupa una pagina e la componente dell'indirizzo logico che individua le pagine è suddivisa in due parti di 4 bit ciascuna, denominate indice di primo livello (che indicizza la tabella di primo livello) e indice di secondo livello (che indicizza la tabella di secondo livello).
La tabella di primo livello è caricata permanentemente in memoria principale, in un blocco noto al sistema operativo, mentre le tabelle di secondo livello sono caricate a domanda.
L'elemento della tabella di primo livello di indice *i* è il descrittore della tabella di secondo livello di indice *i*, e contiene l'indicatore *P* di presenza della tabella ed eventualmente il blocco di memoria fisica nel quale la tabella medesima è caricata.
L'elemento della tabella di secondo livello di indice *j* è il descrittore della pagina *j* e contiene l'indicatore *P* di presenza della pagina ed eventualmente il blocco di memoria fisica nel quale la pagina è caricata.

Al tempo t è in esecuzione il processo P e i contenuti parziali della sua tabella delle pagine di primo livello e delle tabelle di secondo livello caricate in memoria principale sono mostrati in figura. Nella memoria fisica sono disponibili alcuni blocchi, che al verificarsi di *page faults* sono utilizzabili per caricare tabelle di secondo livello o pagine del processo. Questi blocchi sono ordinati nella coda <PrimoBlocco> → 90 → 91 → 92 → 93 → 94 → 95…….. e, in caso di *page fault*, sono utilizzati in questo ordine.

Indice 1° Liv	Blocco	P
0000	--	0
0001	--	0
0010	25	1
0011	--	0
0100	28	1
0101	--	0
0110	--	0
0111	33	1
…	…	…
Tabella: 1' livello		

Indice 2° Liv	Blocco	P
0000	--	0
0001	40	1
0010	--	0
0011	--	0
0100	--	0
0101	--	0
0110	41	1
0111	--	0
…….	…	…
Tabella: 2' livello Indice: 0010		

Indice 2° Liv	Blocco	P
0000	--	0
0001	--	0
0010	50	1
0011	--	0
0100	--	0
0101	--	0
0110	--	0
0111	--	0
…….	…	…
Tabella: 2' livello Indice: 0100		

Indice 2° Liv	Blocco	P
0000	63	1
0001	--	0
0010	56	1
0011	--	0
0100	--	0
0101	--	0
0110	--	0
0111	--	0
…….	…	…
Tabella: 2' livello Indice: 0111		

A partire dal tempo t il processo P accede alla memoria in sequenza con i seguenti indirizzi binari, nei quali è omessa la componente di 6 bit riservata all'offset nella pagina:

1. indirizzo 0000 0110
2. indirizzo 0010 0000
3. indirizzo 0000 0110
4. indirizzo 0111 0110
5. indirizzo 0111 0010
6. indirizzo 0110 0110

Per ciascun accesso alla memoria, si chiede: se l'accesso alla tabella di secondo livello determina page fault

- il blocco fisico che contiene (eventualmente dopo il caricamento in memoria) la tabella di secondo livello
- se l'accesso alla pagina riferita determina page fault
- blocco fisico che contiene (eventualmente dopo il caricamento in memoria) la pagina riferita.

Ipotesi: Quando una tabella di secondo livello è caricata in memoria, in ogni suo descrittore il bit di presenza vale 0.
Nell'intervallo di tempo considerato (accessi 1 .. 6) il gestore della memoria non scarica nessuna pagina.

Soluzione

1) Indirizzo 0000 0110

l'accesso alla tabella di 2° livello determina page fault?	
blocco fisico che contiene la tabella di secondo livello	
l'accesso alla pagina riferita determina page fault?	
blocco fisico che contiene la pagina riferita	

2) Indirizzo 0010 0000

l'accesso alla tabella di 2° livello determina page fault?	
blocco fisico che contiene la tabella di secondo livello	
l'accesso alla pagina riferita determina page fault?	
blocco fisico che contiene la pagina riferita	

3) Indirizzo 0000 0110

4) Indirizzo 0111 0110

Sistemi Operativi e Laboratorio, Prova del 13/07/2016

l'accesso alla tabella di 2° livello determina page fault?	
blocco fisico che contiene la tabella di secondo livello	
l'accesso alla pagina riferita determina page fault?	
blocco fisico che contiene la pagina riferita	

l'accesso alla tabella di 2° livello determina page fault?	
blocco fisico che contiene la tabella di secondo livello	
l'accesso alla pagina riferita determina page fault?	
blocco fisico che contiene la pagina riferita	

5) Indirizzo 0111 0010

l'accesso alla tabella di 2° livello determina page fault?	
blocco fisico che contiene la tabella di secondo livello	
l'accesso alla pagina riferita determina page fault?	
blocco fisico che contiene la pagina riferita	

6) Indirizzo 0110 0110

l'accesso alla tabella di 2° livello determina page fault?	
blocco fisico che contiene la tabella di secondo livello	
l'accesso alla pagina riferita determina page fault?	
blocco fisico che contiene la pagina riferita	

Soluzione

1) Indirizzo 0000 0110

l'accesso alla tabella di 2° livello determina page fault?	SI
blocco fisico che contiene la tabella di secondo livello	90
l'accesso alla pagina riferita determina page fault?	SI
blocco fisico che contiene la pagina riferita	91

2) Indirizzo 0010 0000

l'accesso alla tabella di 2° livello determina page fault?	NO
blocco fisico che contiene la tabella di secondo livello	25
l'accesso alla pagina riferita determina page fault?	SI
blocco fisico che contiene la pagina riferita	92

3) Indirizzo 0000 0110

l'accesso alla tabella di 2° livello determina page fault?	NO(*)
blocco fisico che contiene la tabella di secondo livello	90
l'accesso alla pagina riferita determina page fault?	NO
blocco fisico che contiene la pagina riferita	91

(*) Tabella caricata al passo 1) nel blocco 90

4) Indirizzo 0111 0110

l'accesso alla tabella di 2° livello determina page fault?	NO
blocco fisico che contiene la tabella di secondo livello	33
l'accesso alla pagina riferita determina page fault?	SI
blocco fisico che contiene la pagina riferita	93

5) Indirizzo 0111 0010

l'accesso alla tabella di 2° livello determina page fault?	NO
blocco fisico che contiene la tabella di secondo livello	33
l'accesso alla pagina riferita determina page fault?	NO
blocco fisico che contiene la pagina riferita	56

6) Indirizzo 0110 0110

l'accesso alla tabella di 2° livello determina page fault?	SI
blocco fisico che contiene la tabella di secondo livello	94
l'accesso alla pagina riferita determina page fault?	SI
blocco fisico che contiene la pagina riferita	95

Sistemi Operativi e Laboratorio, Prova del 13/07/2016

Esercizio 4 (4 punti)

Si consideri un File System simile a NTFS che alloca i file in sequenze contigue *(run)* individuate mediante coppie del tipo *(inizio, lunghezza),* dove *inizio* è l'indice del primo blocco fisico del *run* e *lunghezza* esprime il numero di blocchi che la compongono. Ogni file è descritto da un *Master Record* che contiene, oltre ad altri attributi, una o più coppie *(inizio, lunghezza).* Il File System è ospitato da un disco con *NCilindri= 100, NFacce= 2* e *NSettori= 200*. Il tempo necessario per percorrere un settore è di *0,1 msec* e il tempo di seek è proporzionale al numero di cilindri attraversati, dove il tempo di seek tra due cilindri adiacenti è pari a *1 msec*. La corrispondenza tra blocco fisico e indice di cilindro, faccia e settore è data dalle seguenti formule:

cilindro= blocco div *(NFacce*NSettori);*

faccia= (blocco mod *(NFacce*NSettori))* div *NSettori;*

settore= (blocco mod *(NFacce*NSettori))* mod *NSettori*.

In questo file system, si consideri un file che occupa 12 blocchi logici, allocati rispettivamente nei blocchi fisici: 600, 601, 602, 603, 604, 605, 21000, 21001, 21003, 1890, 1891, 1892.
Scrivere le coppie *(inizio, lunghezza)* che corrispondono ad ogni run del file:

numero run	***(inizio, lunghezza)***

Supponendo che la testina di lettura/scrittura del disco sia posizionata sul cilindro numero 88, e che dopo ogni operazione di seek la testina raggiunga il settore da leggere dopo metà del tempo rotazionale, calcolare il tempo necessario per leggere dal disco tutto il 3' run del file:

numero di blocchi da leggere:_____
indirizzo del primo blocco del run: _____
clilindro del primo blocco del run: _____
tempo di seek: _____
tempo rotazionale: _____
tempo di lettura dei blocchi: _____
tempo totale per leggere tutto il 3' run: _____

Soluzione

numero run	***(inizio, lunghezza)***
0	600,6
1	21000,2
2	21003,1
3	1890,3

numero di blocchi da leggere: **1**
indirizzo del primo blocco del run: **21003**
clilindro del primo blocco del run: **52**
tempo di seek: (88-52)* 1 msec = **36** msec
tempo rotazionale: **10** msec
tempo di lettura dei blocchi: **0,1** msec
tempo totale per leggere tutto il 3' run: **46,1** msec

Esercizio 5 (4 punti)

Un sistema gestisce il processore con una politica a code multiple (Multi-Level Feedback Queue - MFQ), con 10 classi di priorità di valori da 0 a 9 (valori maggiori corrispondono a priorità maggiore) e con quanto di tempo (QdT) di 10 *msec*. Quando un thread passa in esecuzione, gli viene assegnato un intero quanto di tempo, indipendentemente dal tempo consumato nel precedente turno di esecuzione.
La politica prevede la revoca del processore, che avviene *immediatamente* quando il processo in esecuzione esaurisce il quanto di tempo o quando viene riattivato un processo con priorità maggiore di quello in esecuzione.

Sistemi Operativi e Laboratorio, Prova del 13/07/2016

La priorità di ogni thread X è definita dinamicamente nel modo seguente:

- inizialmente si assegna un valore *normale*, pari a 5;
- se X ha utilizzato per intero il suo quanto di tempo, il suo valore di priorità viene decrementato di 1;
- se X non ha utilizzato per intero il quanto di tempo, il suo valore di priorità resta invariato;
- se X era bloccato per qualsiasi motivo e viene riattivato, il suo valore di priorità viene incrementato di 1;

Al tempo T, quando è in esecuzione il thread E, sono presenti 6 thread: A(6), B(5), C(4), D(3), E(3) ed F(2); la priorità di ogni thread è indicata tra parentesi. Sono inoltre definite le seguenti variabili:

- lock1: variabile di lock al tempo T libera
- lock 2: variabile di lock, al tempo T occupata e sulla sua coda di attesa è sospeso il thread A
- cond1: variabile di condizione, al tempo T sono sospesi sulla sua coda di attesa i thread C e B (in questo ordine)
- cond2: variabile di condizione, al tempo T nessun thread è sospeso su questa variabile

nell'ipotesi che avvengano IN ALTERNATIVA i seguenti sequenza di eventi, si utilizzi la tabella per indicare lo stato del sistema subito dopo ogni evento.

Evento	**Thread in esecuzione**	**Priorità thread**	**Thread bloccati**	**Thread pronti**
Stato al tempo T	E	A(6), B(5), C(4), D(3), E(3), F(2)	A(lock2), C(cond1), B(cond1)	D,F
Scade QdT				
E esegue cond1.signal				
E esegue lock2.release				
E esegue cond2.wait				
E esegue lock1.acquire				
In seguito ad un'interruzione il kernel esegue lock2.release				

Soluzione

Evento	**Thread in esecuzione**	**Priorità thread**	**Thread bloccati**	**Thread pronti**
Stato al tempo T	E	A(6), B(5), C(4), D(3), E(3), F(2)	A(lock2), C(cond1), B(cond1)	D,F
Scade QdT	**D**	A(6), B(5), C(4), D(3), **E(2),** F(2)	A(lock2), C(cond1), B(cond1)	**E**,F
E esegue cond1.signal	**C**	A(6), B(5), **C(5),** D(3), E(3), F(2)	A(lock2), B(cond1)	D,**E**,F
E esegue lock2.release	**A**	**A(7),** B(5), C(4), D(3), E(3), F(2)	C(cond1), B(cond1)	D,E,F
E esegue cond2.wait	**D**	A(6), B(5), C(4), D(3), E(3), F(2)	A(lock2), C(cond1), B(cond1), E(cond2)	F
E esegue lock1.acquire	**E**	A(6), B(5), C(4), D(3), E(3), F(2)	A(lock2), C(cond1), B(cond1)	D,F
In seguito ad un'interruzione il kernel esegue lock2.release	**A**	A(7), B(5), C(4), D(3), E(3), F(2)	C(cond1), B(cond1)	D,E,F

Esercizio 6 (3 punti)

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) il sistema è in stallo [si/no]? _____

2) se il sistema è in stallo caratterizzarne dell'attesa circolare:
_____

_____

3) se il sistema non è in stallo, individuare una sequenza di assegnazione e rilasci delle risorse 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 R3; quindi puo' terminare rilasciando 2 R4 e 2 R3
- P2 ottiene 2 R4 e viene riattivato; puo' terminare rilasciando 2 R4, 2 R2 e 3 R1
- P3 ottiene 2 R1 e viene riattivato; puo' terminare rilasciando 2 R1, 2 R2 e 1 R3

Esercizio 7 (3 punti)

In un sistema che definisce processi a thread singolo (pertanto schedula i processi) e che adotta una politica di schedulazione a code multiple (Multi-Level Feedback Queue - MFQ), sono definiti il semaforo *sem* e i processi A e B con priorità 2 e 1, rispettivamente. Al tempo *t* il processo A è in esecuzione, la coda pronti contiene il processo B e il semaforo *sem* ha valore 0. Dopo il tempo *t* si verifica la seguente sequenza di eventi:

1) A esegue la chiamata di sistema *fork()*, analoga a quella di UNIX, che genera il processo C;
2) Scade il quanto di tempo del processo in esecuzione;
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 le code dei processi pronti e il valore e la coda del semaforo *sem*.

Soluzione

Eventi	In Esecuzione	Coda priorità 2	Coda priorità 1	Valore di sem	Coda di sem
1) A esegue *fork()*	A (2)	C (2)	B (1)	0	-
2) Scade il quanto di tempo;	C (2)	A (2)	B (1)	0	-
3) il processo in esecuzione esegue *P(sem)*	A (2)		B (1)	0	C (2)

4) Il processo in esecuzione esegue *V(sem)*	A (2)	C (2)	B (1)	0	-

Soluzione

Eventi	In Esecuzione	Coda priorità 2	Coda priorità 1	Valore di sem	Coda di sem
1) A esegue *fork()*					
2) Scade il quanto di tempo;					
3) il processo in esecuzione esegue *P(sem)*					
4) Il processo in esecuzione esegue *V(sem)*					

Esercizio 8 (2 punti)

Quali delle seguenti operazioni possono essere eseguite da un thread **solo** in stato supervisore?

Operazione:	**Eseguibili dai processi in stato supervisore**
Ridurre la propria priorità	
Riattivare un processo sospeso	
Modificare lo stack pointer	
Eseguire l'istruzione TSL (test and set lock)	
Invocare una signal su una variabile di condizione	
Modificare il vettore di interruzione	

Soluzione

Operazione:	**Eseguibili dai processi in stato supervisore**
Ridurre la propria priorità	SI
Riattivare un processo sospeso	SI
Modificare lo stack pointer	
Eseguire l'istruzione TSL (test and set lock)	
Invocare una signal su una variabile di condizione	
Modificare il vettore di interruzione	SI

Esercizio 9 (2 punti)

Si consideri un sistema che gestisce la memoria con segmentazione, con spazio logico suddiviso in 4 segmenti con la seguente tabella dei segmenti:

segmento	Base	limite
0	3.234	870
1	12.100	1.200
2	4.500	78
3	10.000	999

Supponendo che il processo in esecuzione riferisca i seguenti indirizzi logici, formati da coppie contenenti l'indice di segmento e l'offset,

1. <0, 3.000>,
2. <0, 800>,
3. <1, 12.000>,
4. <1, 870>,
5. <2, 68>,
6. <3, 0>,

si chiede se ognuno dei precedenti indirizzi logici è legittimo e, in caso affermativo, di qual è il corrispondente indirizzo fisico.

Sistemi Operativi e Laboratorio, Prova del 13/07/2016

Soluzione

Indirizzo logico	Legittimo?	Indirizzo fisico
<0, 3.000>		
<0, 800>		
<1, 12.000>		
<1, 870>		
<2, 68>		
<3, 0>		

Soluzione

Indirizzo logico	Legittimo?	Indirizzo fisico
<0, 3.000>	NO	-
<0, 800>	SI	4.034
<1, 12.000>	NO	
<1, 870>	SI	12.970
<2, 68>	SI	4.568
<3, 0>	SI	10.000