SISTEMI OPERATIVI, CORSI A e B SECONDA VERIFICA INTERMEDIA - 5/6/2008 CORSO |A| |B|

COGNOME E NOME ..... MATRICOLA ..... AULA FILA POSTO

ESERCIZIO 1 (4 punti)

Un sistema simile a Unix ha a disposizione 30 Mbyte di memoria che gestisce con segmentazione, caricamento in partizioni variabili e swapping. Il sistema operativo è allocato permanentemente in una partizione con origine 0 e lunghezza 2 Mbyte.
Al tempo t=100 sono presenti in memoria due processi A e B che occupano le seguenti partizioni:

- il processo A, occupa tre partizioni:
  - una partizione A1 con origine 11 e lunghezza 2 per il codice
  - una partizione A2 con origine 7 e lunghezza 3 per i dati
  - una partizione A3 con origine 22 e lunghezza 1 per lo stack
- il processo B, che occupa tre partizioni:
  - una partizione B1 con origine 13 e lunghezza 3 per il codice
  - una partizione B2 con origine 17 e lunghezza 2 per i dati
  - una partizione B3 con origine 26 e lunghezza 1 per lo stack

Il gestore della memoria adotta politica *best fit* per l'assegnazione delle partizioni.
La politica di scheduling prevede che la sospensione di un processo comporti il suo immediato swap-out in memoria secondaria, mentre la riattivazione di un processo comporti sempre il suo immediato swap-in in memoria principale. Si assume che le operazioni di swap-in e swap-out vengano eseguite in un tempo trascurabile (i.e. istantaneamente).
Negli istanti successivi al tempo t avvengono (solo ed esclusivamente) i seguenti eventi rilevanti per il gestore della memoria:

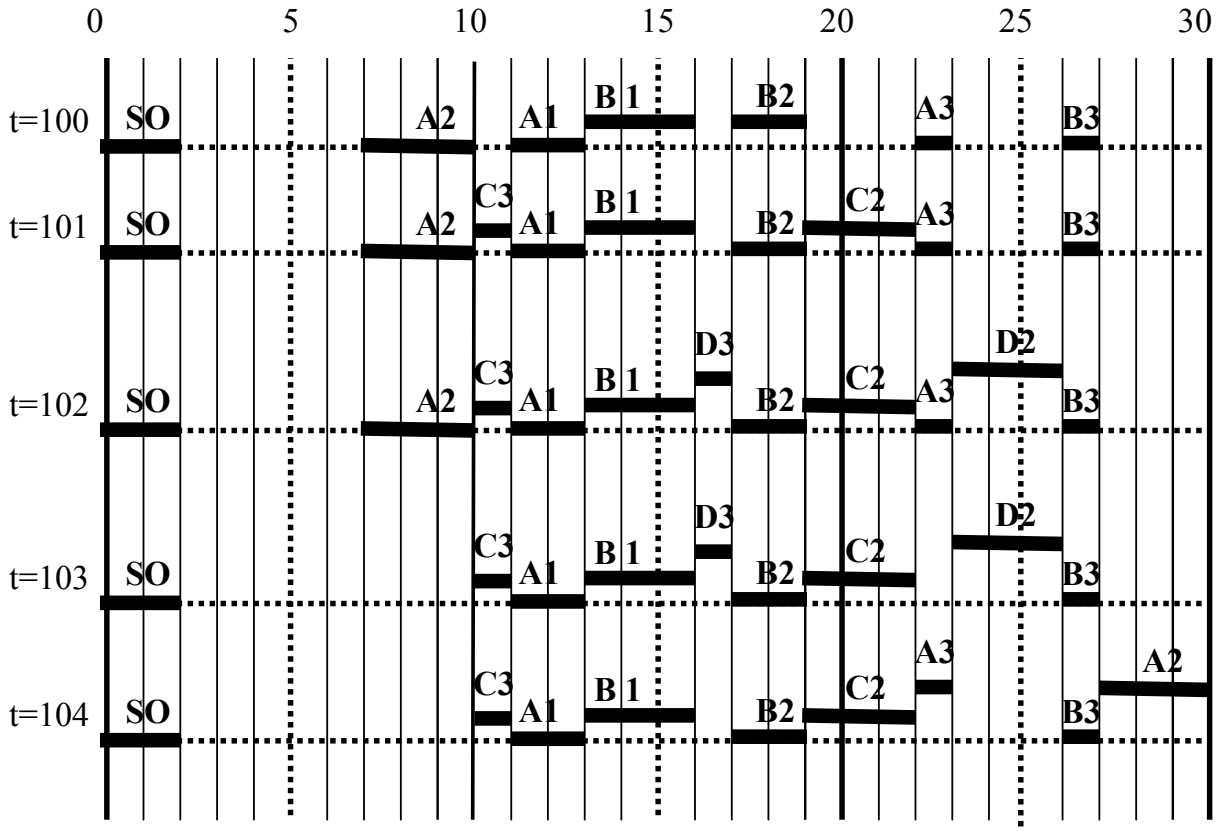
1. tempo t=101: A esegue una fork() che crea il processo figlio C
2. tempo t=102: A esegue una fork() che crea il processo figlio D
3. tempo t=103: A esegue una waitpid()
4. tempo t=104: il processo D esegue una exit()

Utilizzando il grafico sotto riportato, mostrare come evolve l'occupazione della memoria ai tempi 101,102,103 e 104.

SOLUZIONE

- **t= 101:** Si assegnano al processo C due partizioni, nelle quali si carica una copia dei dati e dello stack del processo padre. Il codice del processo C è condiviso con il padre e pertanto non si deve caricarne una copia.
- **t= 102:** Si assegnano al processo D due partizioni, nelle quali si carica una copia dei dati e dello stack del processo padre. Il codice del processo D è condiviso con il padre e pertanto non si deve caricarne una copia.
- **t= 103:** Il processo A si sospende, pertanto i segmenti A2 e A3 subiscono uno swap-out. Il segmento A1 resta caricato in memoria perché è condiviso con i processi C e D.
- **t= 104:** Il processo A viene riattivato pertanto vengono riallocati i segmenti A2 e A3; quindi il processo D termina, pertanto i segmenti D2 e D3 vengono deallocati.

SISTEMI OPERATIVI, CORSI A e B SECONDA VERIFICA INTERMEDIA - 5/6/2008 CORSO |A| |B|



ESERCIZIO 2 (4 punti)

Si consideri un sistema che gestisce la memoria utilizzando un algoritmo di sostituzione second chanche locale e che attribuisce a ogni processo un numero `MaxBlocchi` di blocchi a disposizione del working set. Questo numero è definito dinamicamente e viene incrementato o decrementato dal processo di sistema `WorkingSetManager`, che interviene periodicamente e applica la seguente politica:

- se esistono almeno 3 pagine con bit R=0, si esegue l'algoritmo second chance per rimuovere una pagina e si riduce di 1 il valore di `MaxBlocchi`;
- se tutte le pagine hanno il bit R=1, incrementa di 1 il valore di `MaxBlocchi`.

Al verificarsi di un errore di pagina si ha il seguente comportamento:

- se il numero di pagine attualmente caricate è minore di `MaxBlocchi`, la pagina riferita viene caricata assegnando il primo dei blocchi dalla lista `BlocchiDisponibili`.
- se il numero di pagine attualmente caricate è uguale a `MaxBlocchi`, viene scaricata la pagina prescelta dall'algoritmo di sostituzione e la pagina riferita viene caricata al suo posto.

A un certo istante, quando il tempo virtuale del processo A è t=10, la tabella delle pagine del processo A è la seguente:

Pagina	Blocco	Bit R (Riferita)
0	7	0
1	-	
2	-	
3	8	1
4	9	1
5	-	
6	15	0
7	-	
8	11	0
9	-	
Processo A		

Il valore di `MaxBlocchi` attualmente assegnato al processo A è `MaxBlocchi= 5`, i primi elementi della lista `BlocchiDisponibili` sono `20`→`25`→`16`→ ..., e il puntatore dell'algoritmo second chance riferisce la pagina logica 0 (l'algoritmo analizza le pagine in ordine crescente).

A partire da questo istante il processo A avanza ed esegue i seguenti riferimenti alla memoria:

- al tempo 10 riferisce la pagina 3
- al tempo 11 riferisce la pagina 8
- al tempo 12 riferisce la pagina 0
- al tempo 13 riferisce la pagina 9
- al tempo 14 riferisce la pagina 8

Dopo quest'ultimo riferimento interviene il processo `WorkingSetManager`, che applica la politica sopra definita.

Si chiede:

a) i caricamenti e gli eventuali scaricamenti di pagine eseguiti ai tempi 10, 11, 12, 13, 14;
b) la configurazione della tabella delle pagine del processo A subito prima dell'intervento del processo `WorkingSetManager`
c) il valore di `MaxBlocchi` assegnato al processo A dopo l'intervento del processo `WorkingSetManager`;
d) la configurazione della tabella delle pagine del processo A quando termina l'intervento del processo `WorkingSetManager`

SOLUZIONE

- tempo 10: scaricata la pagina - ; caricata la pagina - nel blocco -; riferita la pagina 3
- tempo 11: scaricata la pagina - ; caricata la pagina - nel blocco -; riferita la pagina 8
- tempo 12: scaricata la pagina - ; caricata la pagina - nel blocco -; riferita la pagina 0
- tempo 13: scaricata la pagina 6 ; caricata la pagina 9 nel blocco 15; riferita la pagina 9
- tempo 14: scaricata la pagina - ; caricata la pagina - nel blocco -; riferita la pagina 8

Il puntatore dell'algoritmo second chance indirizza la pagina logica 8.
La configurazione della tabella delle pagine del processo A subito PRIMA dell'intervento del processo `WorkingSetManager` è quindi:

Pagina	Blocco	Bit R (Riferita)
0	7	0
1	-	
2	-	
3	8	0
4	9	0
5	-	
6	-	
7	-	
8	11	1
9	15	1
Processo A		

Nuova configurazione della tabella delle pagine del processo A subito DOPO dell'intervento del processo `WorkingSetManager`:

Pagina	Blocco	Bit R (Riferita)
0	-	
1	-	
2	-	
3	8	0
4	9	0
5	-	
6	-	
7	-	
8	11	0
9	15	0
Processo A		

La pagina 0 è stata rimossa, il puntatore dell'algoritmo second chance indirizza la pagina 3 e `MaxBlocchi= 4`.

ESERCIZIO 3 (4 punti)

Si consideri un sistema che gestisce la memoria con paginazione dinamica con le seguenti caratteristiche:

- indirizzi logici di 24 bit e ampiezza dello spazio logico di ogni processo pari a 2^{24} byte;
- pagine logiche e blocchi fisici di 1 kByte;
- tabelle delle pagine a tre livelli; la tabella di primo livello comprende 2^6 elementi; quelle di secondo e di terzo livello hano dimensioni uguali tra loro;
- Tutti gli elementi della tabella di primo, secondo e terzo livello hanno lunghezza pari a 24 bit, di cui 12 sono indicatori e i restanti 12 codificano l'indice di un blocco fisico.;

In questo sistema è presente il processo P, che ha allocato nello spazio virtuale tre aree per il codice, dati e stack rispettivamente:

- codice: 100 Kbyte a partire dalla locazione 0
- dati: 32 Kbyte a partire dalla locazione 256* 1024
- stack: 8 Kbyte a partire dall'estremo superiore della memoria virtuale.

Si chiede:

1. la lunghezza del campo offset, in numero di bit;
2. la lunghezza delle componenti dell'indirizzo logico che indicizzano, rispettivamente, la tabella di primo, di secondo e di terzo livello, in numero di bit;
3. lo spazio occupato in memoria dalla tabella di primo livello e da ogni tabella di secondo e di terzo livello (in byte);
4. la massima dimensione della memoria fisica (numero di blocchi e di byte, espressi come potenze di 2);
5. Massimo spazio indirizzabile da una tabella di secondo livello e da una tabella di terzo livello;
6. il numero di tabelle di secondo livello utilizzate per la traduzione degli indirizzi del processo P (considerando le regioni dello spazio logico nelle quali sono collocati il codice, i dati e la pila del processo);

SOLUZIONE

1. lunghezza del campo offset : 10 bit;
2. lunghezza della componenti dell'indirizzo logico che indicizzano, la tabella di primo, secondo e terzo livello: rispettivamente 6, 4 e 4 bit;
3. spazio occupato in memoria dalla tabella di primo livello: $3* 2^6$ byte= 3*64 byte =192 byte ;

   spazio occupato in memoria da ogni tabella di secondo livello: $3* 2^4$ byte = 3*16 byte = 48 byte;

   spazio occupato in memoria da ogni tabella di terzo livello: $3* 2^4$ byte = 3*16 byte = 48 byte;
4. massima dimensione della memoria fisica : 2^{12} blocchi; $2^{12} * 2^{10} = 2^{22}$ byte (4 Mbyte);
5. Massimo spazio indirizzabile con una tabella di terzo livello: 2^4 KB=16Kbyte;

   Massimo spazio indirizzabile con una tabella di secondo livello : 2^4*2^4 KB = 256 KB;
6. numero di tabelle di secondo livello utilizzate per la traduzione degli indirizzi del processo P:

   Nota: ogni tabella di secondo livello indirizza la memoria a partire da un indirizzo logico multiplo di 256 KB

- il codice occupa 100 Kbyte a partire dall'indirizzo 0 → indirizzato da 1 tabella di secondo livello (quella di indice 0)
- lo stack occupa 8 Kbyte compresi negli ultimi 256 KB dello spazio logico → indirizzato da 1 tabella di secondo livello (quella di indice 2^6-1= 63)
- i dati occupano 32 kByte a partire dall'indirizzo 256 KB → indirizzati da 1 tabella di secondo livello (quella di indice 1).

ESERCIZIO 4 (4 punti)

Si consideri un sistema dove la memoria è gestita con paginazione a domanda, che interessa anche lo spazio virtuale del kernel. La pagine logiche e i blocchi fisici hanno un'ampiezza di 2^{10} byte (= 1 kByte). Si utilizza un algoritmo LRU locale.
Al tempo t= 100 la tabella delle pagine del *kernel* è mostrata a fianco. Per ogni pagina, il campo *P* contiene il bit di presenza in memoria. Non è indicato il campo di controllo relativo ai riferimenti passati.
Nel sistema è realizzato un File System FAT-16 dove:

- i blocchi del disco hanno un'ampiezza di 2^{10} byte (= 1 kByte);
- gli elementi della FAT codificano con 2 byte gli indirizzi dei blocchi del disco;
- La FAT è mappata nello spazio virtuale del kernel a partire dall'indirizzo logico 4096 (= 4* 2^{10});
- il file *pippo* è stato aperto dal processo P_i, al quale è stato restituito il *file descriptor* 5.
- la lunghezza corrente del file *pippo* è di 10.000 byte e la posizione corrente del puntatore di lettura è 3272 (= 3*2^{10} + 200)
- il blocco logico di indice 0 del file *pippo* è memorizzato nel blocco fisico 200 del disco (informazione contenuta nella *directory* interessata); i blocchi logici 1, 2, 3 e 4 sono memorizzati rispettivamente nei blocchi fisici 2078 (= 2* 2^{10}+ 30) ; 4196 (= 4* 2^{10}+ 100); 5150 (= 5* 2^{10}+ 30) e 4296 (= 4* 2^{10}+ 200);
- Al tempo t= 100 il processo P_i esegue l'operazione *read(5, &buffer, 200),*

Pag	Blocco	P
0	9100	1
1	9101	1
2	9102	1
3	9103	1
4	5178	1
5	7200	1
6	--	0
7	7250	1
8	--	0
9	--	0
10	--	0
11	7300	1
12	7301	1
13	--	0
14	--	0
15	7310	1
16	7312	1
17	7320	1
18	--	0
19	--	0
…….		

Tab. delle pagine del kernel

Si chiede:

1. la dimensione della FAT, in numero di byte, supponendo che il disco abbia la massima dimensione consentita;
2. il numero di elementi della FAT contenuti in una pagina logica della memoria;
3. quali sono le pagine logiche che contengono la FAT;
4. quanti e quali blocchi logici del file *pippo* si devono leggere per eseguire l'operazione *read(5, &buffer, 200);*
5. quali elementi della FAT devono essere letti a questo scopo;
6. a quali pagine logiche si deve accedere per leggere questi elementi;
7. quanti *page fault* vengono provocati da questi accessi.

SOLUZIONE

1. dimensione della FAT, in numero di byte: la FAT ha 2^{16} elementi, ciascuno di 2 byte; quindi occupa 2^{16}* 2= 2^{17} byte
2. il numero di elementi della FAT contenuti in una pagina logica: 2^{10}/ 2= 2^9 elementi
3. le pagine logiche occupate dalla FAT: sono quella di indice 4 e le successive 2^7- 1 pagine. Infatti la FAT occupa $2^{16}/2^9 = 2^7$ pagine logiche.
4. per eseguire l'operazione si deve leggere unicamente il blocco fisico corrispondente al blocco logico di indice 3. Infatti il puntatore di lettura, che ha valore 3*2^{10} + 200, è posizionato sul carattere 200 del blocco logico 3; l'ultimo dei 200 caratteri da leggere è il carattere 399 del medesimo blocco:
5. devono essere letti gli elementi gli elementi di indici 200 (individuato dalla directory), 2078 (= 4* 2^9+ 30; individuato dal precedente) e 4196 (= 8* 2^9+ 100; individuato dal precedente). Infatti è necessario raggiungere l'elemento di indice 4196 per conoscere l'indirizzo (5150 = 5* 2^{10}+ 30) del blocco fisico corrispondente al blocco logico 3.
6. si deve accedere alle seguenti pagine logiche:
   - pagina 4 (origine della FAT nello spazio logico; contiene l'elemento 200)
   - pagina 4+ 4= 8 (contiene l'elemento 2078 = 4* 2^9+ 30)
   - pagina 4+ 8= 12 (contiene l'elemento 4296 = 8* 2^9+ 200)
7. si verifica 1 *page fault,* per l'accesso alla pagine 8.

ESERCIZIO 5 (4 punti)

In un File System UNIX gli i-node contengono 8 indirizzi di blocchi diretti e 2 indirizzi indiretti, rispettivamente per l'indirizzamento indiretto semplice e per l'indirizzamento indiretto doppio.
I blocchi hanno ampiezza di 512 byte (= 2^9 byte) e gli indici di blocco sono codificati con 2 byte.
Si condideri un file la cui lunghezza corrente è di 135.700 byte e il cui puntatore di lettura ha attualmente il valore 5140.
Si chiede:

1) la massima capacità (espressa in byte) dei dischi indirizzabili da questo File System;
2) Il numero di indirizzi contenuti in ciascun blocco indiretto, di primo o di secondo livello;
3) La massima dimensione (espressa in byte) dei file in questo File System;
4) Con quale modalità viene raggiunto il blocco dati che contiene il carattere individuato dal puntatore di lettura;
5) Con quale modalità viene raggiunto il blocco dati che contiene il carattere EOF;
6) Quali sono i blocchi indiretti che contengono campi indirizzo non significativi e quale è il tasso di frammentazione interna (rapporto tra campi indirizzo non significativi e numero totale di campi indirizzo nel blocco) di questi blocchi

SOLUZIONE

1) Possono essere indirizzati 2^{16} blocchi di 2^9 byte ; la massima capacità disco è di 2^{25} byte (32 Mbyte);
2) Ogni blocco indiretto contiene $2^9 /2 = 256$ indirizzi ;
3) La massima dimensione dei file è pari a $8+ 2^8+ 2^{16}$ blocchi → $2^{12} + 2^{17}+ 2^{25}$ byte (> 32 Mbyte);
4) Il puntatore di lettura individua il carattere 5140 **mod** 512 = 20 del blocco logico 5140 **div** 512= 10; il blocco dati che ospita il blocco logico 10 è raggiunto attraverso l'indirizzo di indice 10-8= 2 del blocco indiretto individuato dall'indirizzo indiretto semplice.
5) Il carattere EOF occupa la posizione 135.700 **mod** 512= 20 del blocco logico 135.700 **div** 512= 265 ; il blocco dati che ospita il blocco logico 265 è raggiunto attraverso l'indirizzo di indice 265-(8+ 256)= 1 del blocco indiretto di primo livello individuato dall'indirizzo indiretto doppio (in altri termini, nel secondo dei blocchi indiretti di secondo livello);
6) I blocchi indiretti che contengono campi indirizzo non significativi sono:
   - il blocco indiretto di primo livello individuato dall'indirizzo indiretto doppio. L'unico campo indirizzo significativo è quello di indice 0; il tasso di frammentazione interna è (256- 1)/256;
   - il blocco indiretto di secondo livello individuato dal secondo campo indirizzo del blocco indiretto di primo livello a sua volta individuato dall'indirizzo indiretto doppio; i campi indirizzo significativi sono quelli di indice 0 e 1: il tasso di frammentazione interna è (256- 2)/256.

ESERCIZIO 6 (2 punti)

Dire quali delle seguenti tabelle di un processo Unix sono modificate dalla chiamata di sistema LSeek:

	SI/NO
process structure:	NO
user structure:	NO
Tabella dei file aperti dal processo:	NO
Tabella dei file aperti di sistema:	SI
Tabella dei file attivi:	NO

ESERCIZIO 7 (2 PUNTI)

Un disco è organizzato con *NCilindri= 100* (numerati da 0 a 99), *NFacce= 4* (numerate da 0 a 3) e *NSettori= 80* (numero di settori per traccia; numerati da 0 a 79). Ogni settore contiene 2^9= 512 byte, pari a un blocco.
A livello logico i blocchi sono individuati con *indici di blocco,* interi compresi nell'intervallo *[0, MaxBlocchi),* e sono allocati sul disco secondo la sequenza *cilindro, faccia, settore* (ad esempio, il blocco logico 401 corrisponde alla terna *(c= 1, f= 1, s= 1),* dove *c* è l'indice di cilindro, *f* è l'indice di faccia e *s* è l'indice di settore.
Si chiede:

1) la capacità del disco, in numero di blocchi e di byte
2) l'indice di blocco corrispondente alla terna *(c= 50, f= 2, s= 30)*;
3) la terna corrispondente al blocco 10.000

SOLUZIONE

1) Ogni cilindro contiene *NFacce*NSettori*= 4* 80 = 320 blocchi;

   la capacità del disco è : 100 * 320= 1000* 2^5 blocchi, pari a 1000* 2^{14} byte ~ 16 Mbyte

2) Dalla formula: *b= c*(NFacce*NSettori) + f*NSettori + s ,*

   l'indice di blocco corrispondente alla terna *(c= 50, f= 2, s= 30)* è:

   b= 50* 4 * 80 + 2* 80 + 30 = 16.190

3) Dalle formule:

   c= b div *(NFacce*NSettori);*

   f= (b mod *(NFacce*NSettori))* div *NSettori;*

   s= (b mod *(NFacce*NSettori))* mod *NSettori* ,

   la terna *(c, f, s)* corrispondente all'indice 10.000 è *(c= 31, f= 1, s= 0)*. Infatti:

   c= 10.000 div 320 = 31;

   f= (10.000 mod 320) div 80 = 80 div 80 = 1;

   s= (10.000 mod 320) mod 80 = 0

ESERCIZIO 8 (2 PUNTI)

Un disco con 100 cilindri, 4 facce e 80 settori per traccia ha un periodo di rotazione di 8 msec (di conseguenza il tempo impiegato per percorrere un settore è 0,1 msec) e un tempo di seek (proporzionale al numero di cilindri attraversati) pari a 0,5 ms per cilindro.
Al tempo 0 termina l'esecuzione dei comandi relativi al cilindro 65 e, in base alla sua politica di scheduling, il disco passa ad eseguire i comandi pendenti sul cilindro 40, che interessano:

- il settore 10 della faccia 0;
- il settore 25 della faccia 3;
- il settore 50 della faccia 1.

Questi comandi sono eseguiti nell'ordine che minimizza il ritardo rotazionale. La capacità di buffering del controllore del disco è sufficiente a garantire l'esecuzione senza ritardi di comandi che interessano settori consecutive.
Si chiede quanto tempo è necessario per completare l'esecuzione dei 3 comandi pendenti, a partire dall'inizio della fase di seek, supponendo che non arrivino altri comandi per il medesimo cilindro e che il primo settore raggiunto dalle teste di lettura/scrittura al termine della fase di seek sia quello di indice 30.

SOLUZIONE

- Tempo necessario per la fase di seek: (65-40)* 0,5= 12,5 msec;
- 1° comando eseguito: settore 50, faccia 1:
  ritardo rotazionale (50- 30)* 0,1= 2 msec; tempo di percorrenza 0,1 msec; totale 2,1 msec
- 2° comando eseguito: settore 10, faccia 0:
  ritardo rotazionale (80-51 +10)* 0,1= 3,9 msec; tempo di percorrenza 0,1 msec; totale 4 msec
- 3° comando eseguito: settore 25, faccia 3:
  ritardo rotazionale (25- 11)* 0,1= 1,4 msec; tempo di percorrenza 0,1 msec; totale 1,5 msec
- Tempo totale: 12,5+ 2,1+ 4+ 1,5= 20,1 msec.

ESERCIZIO 9 (2 PUNTI)

Un disco RAID di livello 4 è composto da 6 dischi fisici, numerati da 0 a 5. Le strip corrispondono a blocchi.
Il disco 5 è ridondante e il suo blocco di indice *i* contiene la parità dei blocchi di indice *i* dei dischi non ridondanti, cioè dei dischi 0, 1, 2, 3 e 4.
Al tempo t si leggono i blocchi di indice 10 dei dischi 3 e 5, che hanno i contenuti mostrati in tabella.

Disco 0	?	?	?	?	?	?	?	?
Disco 1	?	?	?	?	?	?	?	?
Disco 2	?	?	?	?	?	?	?	?
Disco 3	0	1	1	1	1	0	0	1
Disco 4	?	?	?	?	?	?	?	?
Disco 5	1	0	1	1	0	0	0	1

Contenuto del blocco 10 dei dischi 3 e 5 al tempo t

Al tempo t+ 1 si scrive la configurazione 1 1 0 0 1 0 1 1 nel blocco 10 dei disco 3. Si chiede:

1) Quali operazioni si devono eseguire per modificare il contenuto del blocco 10 del disco 3, e quali sono i dischi interessati;
2) Queste operazioni sono indipendenti?
3) il contenuto del blocco 10 dei dischi 3 e 5 dopo le operazioni eseguite al tempo t+1.

SOLUZIONE

1) Si devono eseguire le seguenti operazioni:
   - scrittura sul blocco 10 del disco 3 (per modificare il contenuto)
   - scrittura sul blocco 10 del disco 5 (per modificare la parità)
2) Queste operazioni sono indipendenti
3) Dopo le operazioni eseguite al tempo t+ 1, i contenuti dei blocchi 10 dei dischi 3 e 5 divengono:

Disco 0	?	?	?	?	?	?	?	?
Disco 1	?	?	?	?	?	?	?	?
Disco 2	?	?	?	?	?	?	?	?
Disco 3	1	1	0	0	1	0	1	1
Disco 3	?	?	?	?	?	?	?	?
Disco 5	0	0	0	0	0	0	1	1

Contenuto del blocco 10 dei dischi 3 e 5 dopo le operazioni eseguite al tempo t+1

ESERCIZIO 10 (2 punti)

In un sistema che gestisce la memoria con segmentazione a domanda, l'indirizzo logico è di 32 bit, dei quali i primi 4 bit codificano l'indice di segmento e i restanti 28 bit definiscono l'offset. Ogni elemento della tabella dei segmenti è formato da 7 byte, dei quali 4 bit contengono indicatori utili al gestore della memoria, 3 byte contengono il valore base, e i restanti bit contengono il valore limite.
Si chiede:

1. Il massimo numero di segmenti di un processo
2. La massima dimensione di un segmento (in byte)
3. La dimensione della tabella dei segmenti (in byte)
4. Il massimo valore possibile per l'indirizzo di origine di un segmento

SOLUZIONE

1. Il massimo numero di segmenti di un processo: 2^4 segmenti = 16 segmenti
2. La massima dimensione di un segmento (in byte): 2^{28} byte
3. La dimensione della tabella dei segmenti (in byte): 7*16=112 byte
4. Massimo valore possibile per l'indirizzo di origine di un segmento: 2^{24}