

2. Con riferimento alle tecniche che abbiamo visto per memorizzare il contenuto dei file sui blocchi del disco e di come il file-system ne tenga traccia, individuare quale tra le seguenti affermazioni è falsa.

- A. Nell'allocazione contigua è necessario conoscere a priori la dimensione massima del file in fase di creazione.
- B) Nell'allocazione concatenata (con liste collegate) è presente una certa perdita di spazio dovuto alla frammentazione interna.
- C. L'allocazione contigua è la soluzione che richiede meno memoria RAM ed il minor numero di accessi al disco per determinare il blocco in cui è memorizzato un arbitrario contenuto all'interno di un file.
- D. Usando una FAT per tenere traccia dei blocchi dei file non è necessario mantenere una ulteriore bitmap per tenere traccia dei blocchi liberi.
- E. Nell'allocazione che fa uso della tabella di allocazione dei file (FAT) la capacità del singolo blocco su disco può essere solo parzialmente sfruttata per memorizzare i contenuti del file, dovendo memorizzare il numero del blocco successivo.

3. Consideriamo un sistema che fa uso di memoria virtuale con le seguenti caratteristiche: uno spazio di indirizzamento virtuale da 1 Gb, un numero di pagina virtuale a 22 bit e un indirizzo fisico a 20 bit. Determinare esattamente quanti frame fisici ci sono in memoria.

$$\text{SPAZIO VIRTUALE : } 1 \text{ GB} = 2^{30}$$

$$m = 30$$

$$(30 - m) = 22 \Rightarrow m = 8$$

$$\text{Numero DI PAGINE : } 2^{m-m} \Rightarrow 2^{30-8} \Rightarrow 2^{22}$$

$$\text{DIMENSIONE DI UNA PAGINA : } \frac{2^{30}}{2^{22}} = 2^8 = 256$$

$$\text{NUMERO DI FRAME : } \frac{2^{20}}{2^8} = 2^{12} = 4096$$

4. Supponiamo di avere 3 processi che condividono una variabile x e che i loro pseudo-codici siano i seguenti:

P1:

```
wait(S)
x=x-2
signal(T)
wait(S)
x=x-1
signal(T)
```

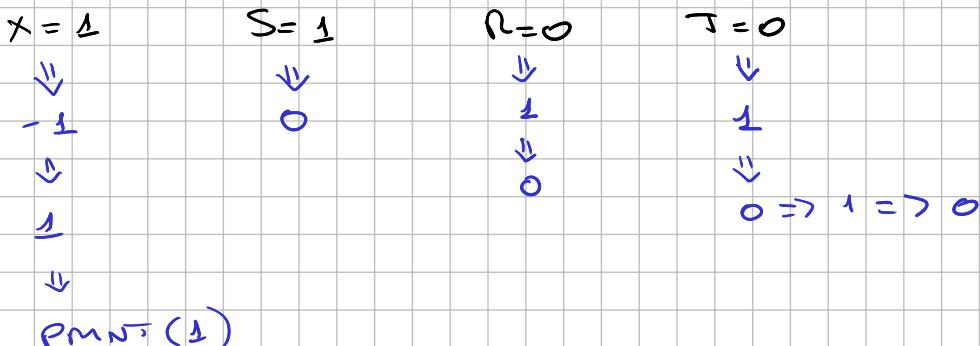
P2:

```
wait(R)
x=x+2
signal(T)
wait(R)
```

P3:

```
wait(T)
if (x<0) signal(R)
wait(T)
print(x)
```

Determinare l'output del processo P3 assumendo che il valore iniziale di x è 1 e che i 3 semafori abbiano i seguenti valori iniziali: S=1, R=0, T=0.



5. Supponiamo di avere un file-system che utilizza per tenere traccia dei file in esso memorizzati la seguente FAT e che prevede le seguenti voci all'interno della cartella radice:

FAT		cartella radice	
indice		nome	primo blocco
1	4		
2	3		
3	15		
4	5		
5	10		
6	12		
7	1		
8	2		
9	3		
10	-1		
...	...		

Indicare esattamente in quale blocco del disco (indicare il numero di blocco) è localizzabili l'offset 10100 all'interno del file pippo.txt. Indicare qual'è la dimensione minima presunta in byte dello stesso file. In tale calcolo tenere conto del fatto che un blocco del file-system è grande 4 kB.

Nota: gli offset sono espressi in byte e partono da 0.

$$\text{Dimensione blocco : } 4 \text{ KB} = 2^{12}$$

$$\text{OFFSET : } 10100$$

$$\frac{10100}{4096} = 2,4 \text{ (Trento Blocco)}$$



BLOCCHI DI "PIPO.TXT": 7 → 1 → 6 → 5 → 10

Dimensione minima: (4 KB + 4 blocchi) + 1 byte :

Dimensione massima: 4 KB . 5 blocchi

7. Supponiamo di avere un disco con 200 tracce (numerate da 0 a 199) la cui velocità di seek è di 1 traccia per ms. All'istante $t=0$ il sistema operativo sta servendo una richiesta sulla traccia 100 e in coda ci sono già le seguenti richieste per le tracce (50, 115, 180). Successivamente arrivano altre richieste all'istante $t=70$ per la traccia 150 e all'istante $t=130$ per la traccia 90. Si calcoli il tempo di ricerca complessivo (in ms) per servire tutte le richieste secondo la politica LOOK, iniziando in ordine ascendente (dalla traccia 0 verso la traccia 199) e trascurando la latenza rotazionale e il tempo di trasferimento. Indicare anche la sequenza di scheduling considerata.

$$\text{SEEK} = 1 \text{ ms}$$

$$t = 0$$

$$\text{TRACCIA} = 100$$

$$Q: 50, \underline{115}, 180$$

$$t = 15$$

$$\text{TRACCIA} = 115$$

$$Q: 50, \underline{180}$$

$$t = 70$$

$$\text{TRACCIA} = 170$$

$$Q: 50, \underline{130}, 180$$

$$t = 80$$

$$\text{TRACCIA} = 180$$

$$Q: 50, \underline{150}$$

$$t = 110$$

$$\text{TRACCIA} = 150$$

$$Q: 50$$

$$t = 130$$

$$\text{TRACCIA} = 130$$

$$Q: 50, 50$$

$$t = 170$$

$$\text{TRACCIA} = 90$$

$$Q: 50$$

$$t = 210$$

$$\text{TRACCIA} = 50$$

2. Supponiamo di avere cinque processi: P1, P2, P3, P4 e P5 con rispettive durate: 7, 5, 2, 4, 4 secondi. Supponiamo anche che i tempi di arrivo nella coda di scheduling dei vari processi sia, rispettivamente: 0, 3, 4, 6 e 9 secondi (ovvero, ad esempio, P4 arriva a tempo 6).

Assumendo di utilizzare l'algoritmo di scheduling Shortest Remaining Time Next, quale è il processo che sarà schedulato per ultimo? Riportare solo il nome del processo.

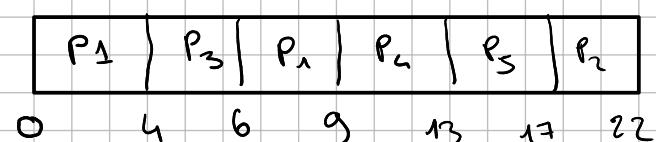
Processo	DURATA	ARRIVO
P1	7 (0)	0

P2	5	3
----	---	---

P3	2 (0)	4
----	-------	---

P4	4	6
----	---	---

P5	4	9
----	---	---



P1 P1 P3 P1 P4 P5 P2

0 3 4 6 9

4. Con riferimento all'algoritmo di scheduling SJF visto a lezione, individuare quale delle seguenti affermazioni è falsa.

- A. Non utilizza *preemption* ma può eventualmente essere adattato.
- B. È di difficile impiego nella pratica su sistemi interattivi.
- C. Il tempo di turnaround è ottimale in presenza di job con tempi di arrivo non nulli.
- D. È pensato per funzionare con processi prettamente non-interattivi (niente I/O).
- E. Necessita la conoscenza a priori del tempo che il job impiegherà per completarsi.

5. Supponiamo di utilizzare un semaforo mutex per coordinare l'operato di alcuni processi su alcune strutture condivise. Al termine delle operazioni da parte di tutti i processi, che valore ci aspettiamo di vedere per il semaforo?

Indicare il valore esatto.

1 vero falso

[1]

0 falso

Abbiamo i seguenti processi da elaborare con l'algoritmo FCFS:

Processo	Arrivo	Durata
P1	0	7
P2	2	4
P3	4	1
P4	5	4

P1 arriva al s0 e andrà subito in elaborazione e ci rimarrà fino al compleimento in s7. P2 arriverà a s2 e attenderà i 5s del rimanente completamento di P1 per andare in esecuzione ed essere completato all's9. Così via per gli altri, dove P3 e P4 attendono entrambi s7 e vengono completati in P3=8, P4=11.
 Tempi di attesa: P1=0, P2=5, P3=7, P4=7 (media 4,75)
 Tempi di completamento: P1=7, P2=9, P3=8, P4=11 (media 8,75)



$$\text{t.m.a.: } 0 + (7-2) + (11-4) + (12-5) = \\ 0 + 5 + 7 + 7 = 22.75$$

$$\text{t.m.c.: } 7 + (11-2) + (12-4) + (16-5) = \\ 7 + 9 + 8 + 11 = 45.75$$

Abbiamo i seguenti processi da elaborare con l'algoritmo SJF:

Processo	Arrivo	Durata
P1	0	7
P2	2	4
P3	4	1
P4	5	4

P1 arriva al s0 e andrà subito in elaborazione e ci rimarrà fino al complemento in s7. In quest'ultimo istante sono arrivati tutti gli altri processi, si sceglie quindi quello con durata minore che è P3 e viene eseguito, e così via.
 Ordine di elaborazione: P1, P3, P2, P4
 Tempi di attesa: P1=0, P2=6, P3=3, P4=7 (media 4)
 Tempi di completamento: P1=7, P2=10, P3=8, P4=11 (media 8)



$$\tau_{\text{m.a.}} : 0 + (7 - 4) + (8 - 2) + (11 - 5) \\ 0 + 3 + 6 + 7 = 16$$

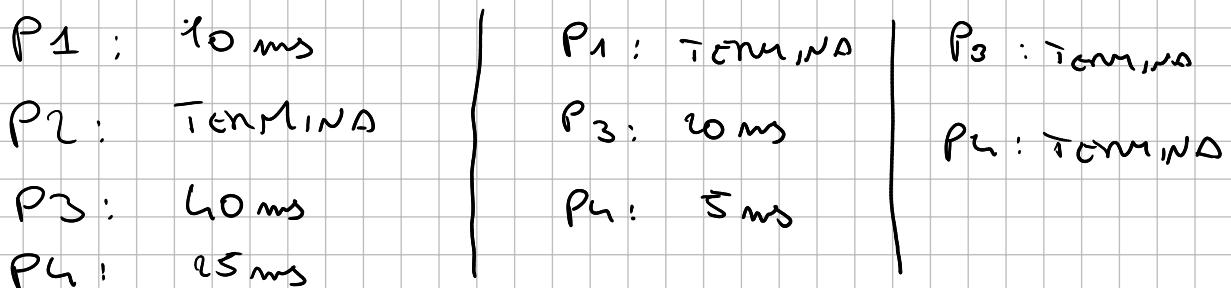
$$\tau_{\text{m.c.}} : 7 + (8 - 4) + (11 - 2) + (16 - 5) \\ 7 + 4 + 10 + 11 = 32$$

Abbiamo i seguenti processi in coda con relativa durata in millisecondi, e quanto di tempo stabilito di 20 ms:

Processi	P1	P2	P3	P4
Durata (millisecondi)	30	15	60	45

Verranno eseguiti nel seguente ordine:

- P1 (interrotto dopo 20 ms, ne rimangono altri 10)
- P2 (termina la propria esecuzione perché dura meno di 20 ms)
- P3 (interrotto dopo 20 ms, ne rimangono altri 40)
- P4 (interrotto dopo 20 ms, ne rimangono altri 25)
- P1 (termina la propria esecuzione perché necessitava di meno di 20 ms)
- P3 (interrotto dopo 20 ms, ne rimangono altri 20)
- P4 (interrotto dopo 20 ms, ne rimangono altri 5)
- P3 (termina la propria esecuzione perché necessitava di esattamente 20 ms)
- P4 (termina la propria esecuzione)



Supponiamo di utilizzare un algoritmo di scheduling preemptive come il Round-Robin: assumendo di avere un context switch effettuato in 5 ms e di usare quanti di tempo lunghi 50 ms, a quanto ammonta la percentuale di overhead per la gestione dell'interlacciamento dei processi?

$$\frac{5}{50+5} = 0,09 \cdot 100 = 9\%$$

formula: $\frac{C}{q+C}$

C: TEMPO CONTEXT SWITCH
 q: QUANTO DI TEMPO

9. Supponiamo di avere 3 processi che condividono una variabile x e che i loro pseudo-codici siano i seguenti:

P1: wait(S) $x=x-1$ signal(T)	P2: wait(R) $x=x+2$ signal(T)	P3: wait(T) if ($x>0$) signal(R) else signal(S) wait(T) print(x) signal(T)
--	--	---

Determinare l'output del processo P3 assumendo che il valore iniziale di x è -2 e che i 3 semafori abbiano i seguenti valori iniziali: S=0, R=1, T=0.

$$S = 0 \Rightarrow 0$$

$$R = 1 \Rightarrow 0$$

$$T = 0 = 1 \Rightarrow 0 \Rightarrow 1 \Rightarrow 0$$

$$x = -2 \Rightarrow 0 \Rightarrow -1 \Rightarrow \text{Point}(-1)$$

Con riferimento ad un sistema a thread che segue il modello "1-a-1", quale delle seguenti affermazioni è errata? Assumere l'utilizzo di uno scheduler con prelazione.

- A. L'accesso da parte di un thread ad una area dello spazio di indirizzamento che non è attualmente paginata, non implica il blocco di tutti gli altri thread dello stesso processo.
- B. E' necessaria la modalità supervisor della CPU per manipolare la tabella dei thread.
- C. Un thread che non rilascia spontaneamente la CPU può comunque bloccare gli altri thread dello stesso processo.
- D. Questo modello può essere utilizzato solo se supportato nativamente dal Sistema Operativo.
- E. L'overhead per il context switch tra thread è minore se i thread coinvolti appartengono allo stesso processo.

8. Con riferimento al problema dello scheduling quando applicato a sistemi multiprocessore, individuare quale delle seguenti affermazioni è errata.

- A. Nella multielaborazione asimmetrica solo un core può operare sulle strutture interne al kernel per la gestione dei processi.
- B. La migrazione guidata dei processi e quella spontanea sono due tecniche compatibili tra di loro.
- C. Nella multielaborazione simmetrica tutti i core sono considerati uguali.
- D. La gestione del bilanciamento del carico delle varie CPU è superflua in caso di code private di processi pronti per ogni CPU.
- E. La necessità di supportare la predilezione per una CPU serve principalmente per ottenere una migliore gestione della cache della memoria.

5. Abbiamo visto vari algoritmi di scheduling dei movimenti della testina di un disco rotazionale che permettono di ottimizzare vari parametri. In particolare ci siamo soffermati sul "seek time". Supponiamo di avere una determinata sequenza di richieste di I/O e che i numeri di cilindro associati a tale richieste siano, in ordine di arrivo, i seguenti: 73, 15, 11, 59, 80, 45, 90, 2. Assumiamo che: la testina, all'arrivo della prima richiesta (ovvero quella per il cilindro 73), sia già posizionata sul cilindro 50; che i cilindri siano in totale 100 e che siano numerati da 0 a 99. Supponendo di utilizzare l'algoritmo di scheduling "shortest seek time first", indicare la sequenza esatta che si andrebbe ad utilizzare e la distanza totale coperta dalla testina in termini di numero di cilindri traversati.

73, 15, 11, 59, 80, 45, 90, 2

INIZIO: 50

$$S_1: 45 \quad D: (50 - 45) = 5$$

$$S_2: 59 \quad D: (59 - 45) = 14$$

$$S_3: 73 \quad D: (73 - 59) = 14$$

$$S_4: 80 \quad D: (80 - 73) = 7$$

$$S_5: 90 \quad D: (90 - 80) = 10$$

$$S_6: 15 \quad D: (90 - 15) = 75$$

$$S_7: 11 \quad D: (15 - 11) = 4$$

$$S_8: 2 \quad D: (11 - 2) = 9$$

DISTANZA TOTALE: 138

6. Abbiamo visto come sfruttare il parallelismo anche sui dischi, in particolare abbiamo parlato delle varie configurazioni RAID (livelli RAID). Supponendo di essere interessati unicamente a migliorare le prestazioni nell'accesso ai nostri dati rispetto all'uso di una singola unità, quale livello RAID scegliereste?

- A. RAID 0 B. RAID 4 C. RAID 1 D. RAID 3 E. RAID 5

7. Supponiamo di utilizzare un file-system UNIX basato su i-node particolari che contengono i seguenti campi: 12 indirizzi diretti a blocchi di dati, 1 indirizzo ad un blocco indiretto singolo e 1 indirizzo ad un blocco indiretto doppio. Supponendo di avere numeri di blocchi a 32 bit e blocchi su disco da 1 kb, indicare esattamente la dimensione massima (in kb) supportata da un simile i-node. Esplicitare il calcolo utilizzato.

SOLUZIONE:

L'indirizzamento indiretto ha blocchi da 1kb pieno di indirizzi a 32 bit, quindi sono 12 blocchi di indirizzo diretto + 256 di indiretto + 256^2 di indiretto doppio = 65804 blocchi.

9. Tra gli algoritmi di sostituzione delle pagine abbiamo visto l'algoritmo di Aging. Tale algoritmo mantiene per ogni pagina un contatore binario che viene aggiornato periodicamente tenendo conto del bit di riferimento delle varie pagine. Supponiamo di avere 5 pagine in memoria e che a tempo t lo stato dei contatori binari sia il seguente:

pagina	contatore
A	00101000
B	10000001
C	01100000
D	00101011
E	00000010

In seguito, a tempo $t+1$, tali contatori sono aggiornati tenendo conto dello stato dei seguenti bit di riferimento (A:1, B:0, C:1, D:0, E:1) ed ancora, a tempo $t+2$, vengono nuovamente aggiornati con i bit di riferimento (A:0, B:1, C:0, D:0, E:1). Riportare nella risposta lo stato dei contatori dopo aver applicato questi due aggiornamenti ed indicare quale sarebbe la pagina destinata ad essere rimossa secondo l'algoritmo Aging se fosse necessario a tempo $t+3$.

t

A : 00101000
 B : 10000001
 C : 01100000
 D : 00101011
 E : 00000010

$t+1$

A : 10010100
 B : 01000000
 C : 10110000
 D : 00101010
 E : 10000001

$t+2$

A : 01001010
 B : 10100000
 C : 01011000
 D : 00001010
 E : 10000000

$t+3$ \rightarrow VIENE ELIMINATO

101	010	110	100	110
100	111	100	010	110
110	100	110	-	111
111	010	001	101	1100

$$\begin{array}{r}
 110 \\
 100 \\
 110 \\
 111 \\
 \hline
 011
 \end{array}$$

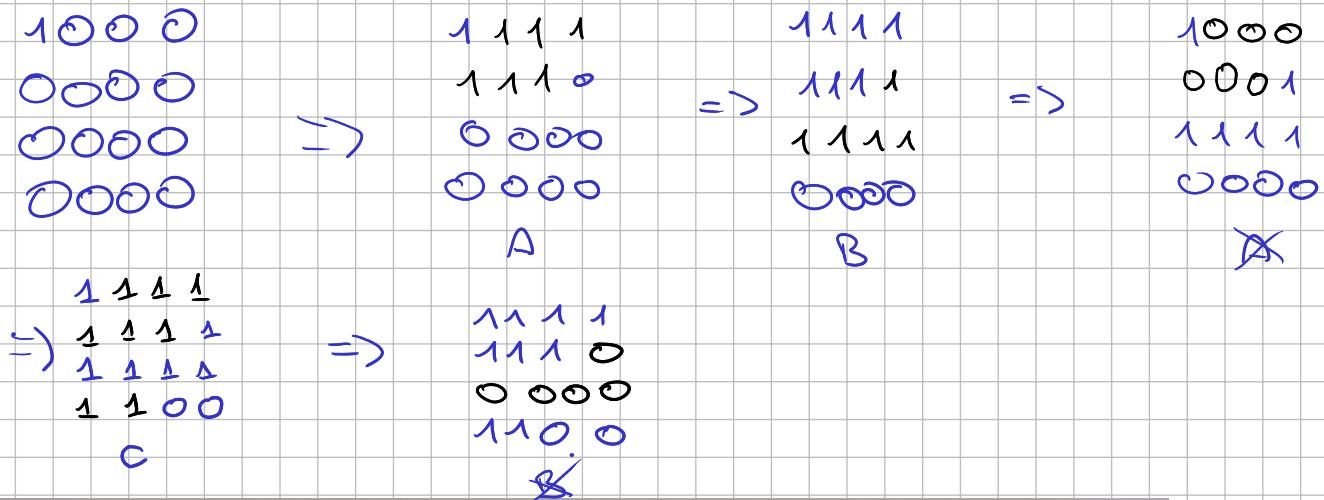
Per ricostruire il blocco mancante, devo fare solamente lo XOR fra gli elementi delle terza (3a) riga; quindi:

$$111 \text{ XOR } 110 = 001 \text{ XOR } 100 = 101 \text{ XOR } 110 = 011$$

2. I sistemi UNIX supportano sui propri file-system sia i collegamenti simbolici (soft-link) che quelli fisici (hard link). Con riferimento a ciò che si può fare attraverso le chiamate di sistema, individuare tra le seguenti affermazioni quella errata.

- A. Non è possibile creare un collegamento fisico ad un oggetto residente su un file-system diverso.
- B. L'i-node a cui si accede usando un collegamento fisico è lo stesso a cui si accede usando il riferimento originale.
- C. Non è possibile creare un collegamento simbolico ad una directory.
- D. E' sempre possibile creare un collegamento simbolico ad un file di dispositivo speciale.
- E. Non crea alcun problema creare un collegamento fisico ad un collegamento simbolico, purché quest'ultimo sia consistente.

9. Supponiamo di utilizzare una bitmap per tenere traccia dei blocchi liberi all'interno di un file-system e che tale bitmap, subito dopo la formattazione, appaia nel seguente modo: 1000 0000 0000 0000 (il primo blocco viene usato per la directory radice). Il sistema cercherà di allocare sempre il blocco libero con indice più piccolo. Riportare lo stato finale della bitmap dopo aver eseguito le seguenti operazioni sul file-system: creazione di un file A da 6 blocchi, creazione di un altro file B da 5 blocchi, cancellazione del file A, creazione di un altro file C da 8 blocchi e cancellazione del file B.



Assumiamo che il risultato di tale scansione sia il seguente:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	numeri di blocco
1	1	0	1	0	0	0	1	1	0	0	1	0	0	0	1	blocchi in uso
0	0	1	0	1	0	1	0	0	1	1	0	1	1	0	0	blocchi liberi

Individuare eventuali errori, correggerli (ricopriando sotto la tabella o intervenendo direttamente sulla tabella già presente) e descrivere brevemente il tipo di intervento fatto.

4. Individuare quale tra le seguenti condizioni non è un criterio da considerare per una buona soluzione al problema delle corse critiche.

- A. Quando un processo si mette in attesa per entrare nella propria regione critica bisogna poter quantificare in modo preciso il tempo massimo per cui resterà bloccato.
 - B. Non si dovrebbe impiegare busy waiting per gestire il blocco dei processi.
 - C. Nessun processo al di fuori della propria regione critica può bloccare altri processi.
 - D. Non si devono fare presupposti sulla velocità e sul numero di CPU a disposizione.
- Al più due processi possono trovarsi contemporaneamente nelle loro regioni critiche.

5. Con riferimento al problema dello scheduling quando applicato a sistemi multiprocessore, individuare quale delle seguenti affermazioni è errata.

- A. Non ha senso parlare di migrazione spontanea innescata da un processore in assenza di code private per ogni CPU.
- B. La necessità di supportare la predilezione per una CPU serve principalmente per ottenere una migliore gestione della cache della memoria.
- C. Nella multielaborazione simmetrica con code private per le CPU, se non si adottano politiche di migrazione, può succedere che una CPU rimanga senza fare nulla pur avendo altri processori occupati con parecchi lavori.
- D. Nella multielaborazione asimmetrica tutte le CPU operano regolarmente sulle strutture interne al kernel per la gestione dei processi.
- E. Nella multielaborazione simmetrica con code private per le CPU, il bilanciamento del carico delle varie CPU è un obiettivo che nega la possibilità di garantire la predilezione forte di un processo per un processore.

7. Supponiamo di avere una tabella delle pagine e che la MMU faccia uso di memoria associativa per velocizzarne l'accesso. Assumendo che un accesso alla memoria centrale richieda 150 nsec e che invece l'accesso al TLB necessiti di soli 20 nsec, calcolare il tempo medio effettivo di accesso ad una parola della memoria centrale assumendo una percentuale media di successi nell'uso della memoria associativa (TLB ratio) pari al 70%. In tale calcolo bisogna considerare anche il prelievo effettivo del dato richiesto dalla RAM.

Riportare il valore richiesto con la formula utilizzata.

TEMPO DI ACCESSO ALLA MEMORIA : 150 nsec

TEMPO DI ACCESSO ALLA TLB : 20 nsec

TLB RATIO : 70 %

TLB HIT : $150 + 20 = 170 \text{ nsec}$

TLB MISS : $150 + 20 + 150 = 320 \text{ nsec}$

T.M.A : $(0.7 \cdot 170) + (0.3 \cdot 320) = 119 + 96 = 133$

1. Abbiamo visto che la maggior parte dei file-system UNIX supportano nativamente i soft-link e gli hard-link. Individuare tra le seguenti affermazioni quella errata.

- A. Il numero di i-node di un soft-link è diverso da quello dell'i-node dell'oggetto riferito.
- B. Non comporta alcun problema creare un soft-link ad un altro soft-link purché quest'ultimo sia consistente.
- C. Per un utente non è possibile creare un hard-link ad una directory.
- D. E' sempre possibile creare un soft-link ad un file di dispositivo speciale.
- E. Non è possibile creare un soft-link ad un oggetto residente su un file-system diverso.

2. Supponiamo di avere un file-system che utilizza per tenere traccia dei file in esso memorizzati la seguente FAT e che prevede le seguenti voci all'interno della cartella radice:

FAT		cartella radice	
indice		nome	primo blocco
1	2
2	10	pippo.txt	3
3	7	hello.c	9
4	-1
5	0	fact.c	1
6	3
7	3		
8	-1		
9	8		
10	4		
...	...		

$$3 \rightarrow 8 \text{ kb}$$

$$2 \text{ kb} \times 3 = 6 \text{ kb} + 1 \text{ bit}$$

4 blocchi

Tenendo conto dei dati sopra riportati e del fatto che un blocco del file-system è grande 2 kb, rispondere alla seguenti due domande:

- quanti blocchi sono stati allocati per memorizzare il contenuto del file pippo.txt?
- qual è la dimensione minima (in byte) che possiamo supporre abbia il file pippo.txt?
Indicare, tra le possibili dimensioni (in byte), solo quella più piccola.

$$3 \Rightarrow 7 \Rightarrow 5 \Rightarrow 6 \Rightarrow -1 \quad (\text{4 blocchi})$$

$$(2 \cdot 4) = 8 \text{ kb} \quad \text{dimensione massima}$$

$$(2 \cdot 3) + 1 \text{ bit} = 6 \text{ kb} + 1 \text{ bit} \quad \text{dimensione minima}$$

Dati i seguenti dischi:

Disco 1	Disco 2	Disco 3
0010	1001	101x
1001	0101	110x
0110	0100	001x

1011
1100
0110

Trovare i bit di parità x

x = 011 dato che con lo xor risultano tali valori

$$x = 100$$

Esercizio Dispositivi-6

Un disco RAID di livello 4 è composto da 5 dischi fisici, numerati da 0 a 4. Le strip corrispondono a blocchi.

Il disco 4 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 e 3.

I blocchi di indice 5 dei dischi non ridondanti contendono rispettivamente:

Disco 0: 0 1 1 0 1 0 0 1

Disco 1: 0 1 0 0 1 1 0 1

Disco 2: 1 1 1 0 1 1 0 0

Disco 3: 0 1 1 1 1 0 0 1

Si chiede;

- 1) il contenuto del blocco di indice 5 del disco ridondante
- 2) come cambia il contenuto del blocco di indice 5 del disco ridondante se in seguito a una scrittura il contenuto del blocco 5 del disco 2 diviene 0 0 1 1 0 0 1 0

D_0	D_1	D_2	D_3	D_4
0	0	1	0	1
1	1	-1	1	0
1	0	1	1	1
0	0	0	1	1
1	1	1	1	0
0	1	1	0	0
0	0	0	0	0
1	1	0	1	1

D_0	D_1	D_2	D_3	D_4
0	0	0	0	0
1	1	0	1	1
1	0	1	1	1
0	0	1	1	0
1	1	0	1	1
0	1	0	0	1
0	0	1	0	1
1	1	0	1	1

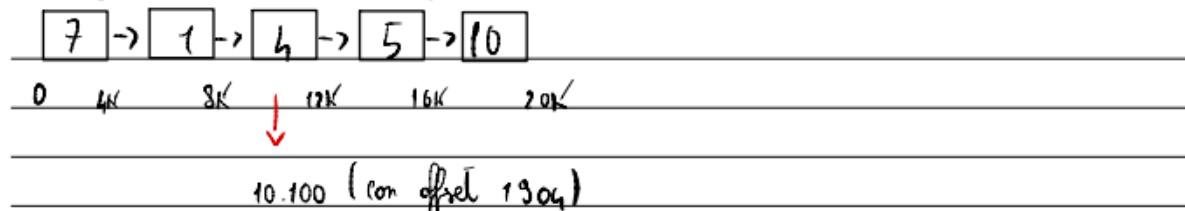
5. Supponiamo di avere un file-system che utilizza per tenere traccia dei file in esso memorizzati la seguente FAT e che prevede le seguenti voci all'interno della cartella radice:

FAT	cartella radice
indice	
1	4
2	3
3	15
4	5
5	10
6	12
7	1
8	2
9	3
10	-1
...	...

nome	primo blocco
...	...
pippo.txt	7
...	...

Indicare esattamente in quale blocco del disco (indicare il numero di blocco) è localizzabili l'offset 10100 all'interno del file pippo.txt. Indicare qual'è la dimensione minima presunta in byte dello stesso file. In tale calcolo tenere conto del fatto che un blocco del file-system è grande 4 kB.

Nota: gli offset sono espressi in byte e partono da 0.



Supponiamo di avere a disposizione l'istruzione TSL e di voler realizzare la mutua esclusione nell'accesso alle regioni critiche. Scrivere il codice in pseudoassembly delle procedure enter-region e leave-region.

MUTEX - LOCK

```

TSL  REG, MUTEX
CMP  REG, #0 // Se c'è già un thread in una critica
JE   OK
CALL THREA-YIELD
JMP  MUTEX-LOCK // Riprova perché non è zero (SPINLOCK)
OK: RET
    
```

MUTEX - UNLOCK

```

MOVE MUTEX, #0
RET
    
```

1. L'algoritmo di Aging per la sostituzione delle pagine prevede di mantenere un contatore C associato ad ogni pagina caricata in memoria. Tale contatore viene consultato nel momento in cui si deve scegliere quale pagina rimuovere dalla memoria: viene scelta quella con il contatore più basso.

Indicare esattamente qual'è l'aggiornamento periodico che viene effettuato su tale contatore.

- A. Somma del bit di referenziamento R al contatore C, con seguente shift a sinistra.
- B. Shift a sinistra di C e somma del bit di referenziamento R.
- C. Shift a sinistra di C ed inserimento del bit di referenziamento R come bit più significativo.
- D. Shift a destra di C ed inserimento del bit di referenziamento R come bit più significativo.
- E. Shift a destra di C ed inserimento del bit di modifica M come bit meno significativo.

6. Consideriamo il seguente codice in C una volta compilato ed eseguito su un sistema UNIX:

```
main() {  
    int pid, i;  
    for (i=0; i<=3; i++) {  
        pid=fork();  
        if (pid==0 && i!=2)  
            exit(0);  
    }  
    exit(0);  
}
```

Determinare il numero esatto di processi generati da tale codice (compreso quello direttamente creato dalla shell in seguito all'invocazione).

16

Facendo uso di una lista concatenata, che ha blocchi da 4kb e un numero di blocco di 32bit, quanti blocchi occorrono per memorizzare un file da 40kb?

Un blocco della lista è di 4KB, ovvero $4 \cdot 2^{10}$ byte.

Ogni blocco della lista concatenata ha un solo puntatore che specifica il numero del blocco successivo.

Questo numero è di 32bit, ovvero di 4byte.

Dunque ogni blocco della lista ha uno spazio per i dati di:

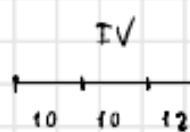
$$(4 \cdot 2^{10} - 4) \text{ byte} = (4 \cdot 1024 - 4) \text{ byte} = (4096 - 4) \text{ byte} = 4092 \text{ byte.}$$

Il file è di 40KB, quindi $(40 \cdot 2^{10}) \text{ byte} = 40960 \text{ byte.}$

La dimensione del file diviso lo spazio disponibile per ogni blocco è: $40960 / 4092 = 10,01$.

Sono serviti poco più di 10 blocchi, cioè 11 blocchi.

Supponiamo di utilizzare una tabella delle pagine a 2 livelli per la memoria virtuale. Assumendo che lo spazio di indirizzamento virtuale sia a 32 bit, che la tabella delle pagine di primo livello contenga 1024 voci e che le pagine virtuali abbiano una dimensione di 4 KB, determinare il numero di voci di ogni tabella delle pagine di secondo livello. Inoltre, indicare il numero di fetch alla memoria centrale necessari per determinare che un dato spaziamento all'interno dello spazio virtuale non è mappato in memoria (ignoriamo le fasi necessarie per la gestione del page fault). Indicare le due quantità richieste con le formule utilizzate per determinarle.



$\Rightarrow 4\text{GB}$ Spazio di ind. virtuale

Tab I LV \Rightarrow 1024 voci

Pagine $\Rightarrow 4\text{KB} \Rightarrow 2^{12}\text{B}$

IV $\Rightarrow 10\text{BIT} = 2^{\frac{10}{10}}$ # Voci Tab I LV

$10\text{BIT} = 2^{\frac{10}{10}}$ # Voci Tab II LV

$12\text{BIT} = 2^{\frac{12}{12}}$ B (per pagine tab)

1) 2^{10} voci Tabella II LV

\Rightarrow circa 4GB di spazio da ind. virt., ovvero 2^{32}B .

Dato che ci sono 1024 voci per la tabella di primo livello allora la dimensione delle tabelle di II LV è $2^{32}/2^{10} = 2^{22}\text{B}$, ovvero 4 MB.

Dato che ci sono pagine da 4KB, il numero di voci sarà:

$$4\text{MB}/4\text{KB} = 2^{22}/2^{12} = 2^{10} \text{ voci}, \text{ ovvero } 1024.$$

2) Numero accessi totali = numero di accessi Tab I LV + ... Tab II LV = 2

Supponiamo di avere 4 dischi da 512 MB impiegati per realizzare un volume RAID4. Avendo come riferimento la seguente visione parziale del contenuto dei dischi, ricostruire il contenuto dei blocchi mancanti (X, Y), esplicitando il calcolo effettuato. Indicare inoltre la capienza attesa in MB del volume RAID4 così ottenuto.

Inoltre la capacità ottenuta (in MB) del volume RAID-4 così ottenuto.			
disco 0	disco 1	disco 2	disco 3
10001111	0011011	0010011	101800111
X	01101101	11110100	01011111
11010100	00111010	01009111	10101001
01010111	1110011	Y	1100111
10101011	01010101	1011101	0100111

Nota: in questo esempio assumiamo che il blocco di base (stripe) abbia dimensione pari a 8 bit.

1000 1111	
XXXX XXXX	⇒ 11110000
1101 0100	
1010 1011	
0001 0011	
1111 0100	
0100 0111	
YYYY YYYY	⇒ 0001 1101
10111101	

$$\text{Volume} = \text{Vol Blocco} \cdot \# \text{ Blochi}$$

$$= 512 \text{ MB} \cdot 4$$

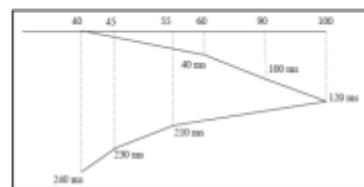
$$= 2048 \text{ MB}$$

(Es. con tempi di arrivo diversi) Si consideri un disco con un intervallo di tracce da 0 a 100, gestito con politica SCAN. Inizialmente la testina è posizionata sul cilindro 40; lo spostamento ad una traccia adiacente richiede 2 ms. Al driver di tale disco arrivano richieste per i cilindri 90, 45, 40, 60, 55, rispettivamente agli istanti 0 ms, 20 ms, 30 ms, 40 ms, 80 ms. Si trascuri il tempo di latenza. (1) In quale ordine vengono servite le richieste? (2) Il tempo di attesa di una richiesta è il tempo che intercorre dal momento in cui è sottoposta al driver a quando viene elettivamente servita. Qual è il tempo di attesa medio per le cinque richieste in oggetto?

Assumendo una direzione di movimento ascendente all'istante 0, le richieste vengono servite nell'ordine 60, 90, 55, 45, 40

Il tempo di attesa medio per le cinque richieste in oggetto è

$$\frac{(40-40)+(100-0)+(210-80)+(230-20)+(240-30)}{5} = \frac{0+100+130+210+210}{5} = 130 \text{ ms}$$



In un disco con blocchi di 2 Kbyte (= 2^{11} byte), è definito un file system FAT 16. Ogni elemento ha lunghezza di 2 byte e indirizza un blocco del disco. La copia permanente della FAT risiede nel disco a partire dal blocco di indice 0 e una copia di lavoro viene caricata in memoria principale all'avviamento del sistema operativo. Supponendo che la FAT sia dimensionata in base alla massima capacità di indirizzamento dei suoi elementi si chiede:

- 1) il numero di blocchi dati indirizzabili dalla FAT.
- 2) il numero di byte e di blocchi del disco occupati dalla FAT,
- 3) l'indice del primo blocco dati nel disco,
- 4) quale capacità (in blocchi e in byte deve avere il disco) per contenere tutti i blocchi dati indirizzabili.

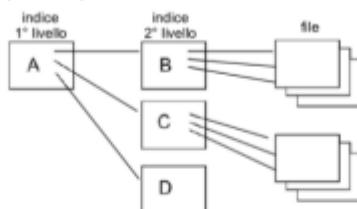
1. Il numero di blocchi dati indirizzabili dalla FAT è 2^{16}

2. La FAT ha 2^{16} elementi di 2 byte pertanto occupa 2^{17} byte è $2^6 = 64$ blocchi

3. Il primo blocco dati nel disco è quello di indice 64 (i blocchi precedenti sono riservati alla FAT)

4. Per contenere tutti i blocchi indirizzabili, il disco deve avere una capacità di almeno $2^{16} + 26$ blocchi e $227 + 217$ byte.

Illustrare il meccanismo di allocazione indicizzata dei file. Si faccia riferimento ad un file system con indici a due livelli, dimensione del blocco logico pari a 512 byte e indirizzi di 4 byte. Come si alloca un file di 1 Mb? Quanto blocchi servono? Come si accede al suo 400° blocco? Come si accede al byte 236.448? Qual è la dimensione massima di un file? Quanti blocchi occupa complessivamente?



L'allocazione indicizzata a due livelli segue questo schema:

Se il blocco logico è di 512 byte e gli indirizzi di 4 byte si hanno $512 / 4 = 128$ indirizzi per blocco.

Un file di 1 Mbyte occupa $1M / 512 = 2K = 2048$ blocchi ($1M = 1024 * 1024$)

2048 blocchi richiedono 2048 indici = $2048 / 128 = 16$ blocchi di 2° livello.

Complessivamente il file occupa $2048 + 16 + 1 = 2065$ blocchi.

Il blocco n. 400 è indicizzato dal blocco indice di 2° livello n. $400 / 128 = 3$, l'indirizzo è il 400 mod 128 = 16-esimo del blocco.

In generale, l'n-esimo indirizzo occupa i byte da $(n - 1) * 4$ a $(n - 1) * 4 + 3$.

Il blocco indice di secondo livello a sua volta è indicizzato dal 4° indirizzo (indirizzo n. 3) dell'indice di 1° livello, che si trova nei byte 12-15 del blocco.

L'accesso al byte 236.448 si risolve così: il byte occupa il blocco $236.448 / 512 = 461$ (contando da 0, quindi è il 462-esimo), e si trova all'offset $236.448 \bmod 512 = 416$. Trovato il blocco si procede come sopra.

La dimensione massima di un file è data dal numero massimo di indirizzi utilizzabili per indicizzarlo.

Nell'indice di 1° livello ci sono al massimo 128 puntatori, quindi ci sono al massimo 128 blocchi indice di 2° livello, ciascuno dei quali contiene 128 puntatori ai blocchi del file, per un totale di $128 \times 128 = 16384$ puntatori, e quindi blocchi del file. Il file può avere una dimensione massima di $16384 \times 512 = 8 \text{ Mbyte}$ (8.388.608 byte), e occupa complessivamente $16384 + 128 + 1 = 16513$ blocchi

In un sistema che usa paginazione, l'accesso al TLB richiede 150ns, mentre l'accesso alla memoria richiede 400ns. Quando si verifica un page fault, si perdono 8ms per caricare la pagina che si sta cercando in memoria. Se il page fault rate è il 2% e il TLB hit il 70%, indicare l'EAT ai dati.

Convertendo tutti i tempi in ms, abbiamo:

- tempo di accesso al TLB: $150 \text{ ns} = 150 \cdot 10^{-9} \text{ ms}$;
- tempo di accesso alla memoria: $400 \text{ ns} = 400 \cdot 10^{-9} \text{ ms}$;
- tempo di gestione del page fault: 8 ms;
- page fault rate: $2\% = 0,02$;
- TLB hit: $70\% = 0,7$.

Quindi

$$\begin{aligned}
 EAT &= TLB \text{ hit} \cdot (150 \cdot 10^{-9}) + (1 - \text{page fault rate}) \cdot (150 \cdot 10^{-9} + 400 \cdot 10^{-9}) + \text{page fault rate} \cdot \\
 &\quad (150 \cdot 10^{-9} + 8 + 400 \cdot 10^{-9}) \\
 &= 0,7 \cdot (150 \cdot 10^{-9}) + 0,28 \cdot (150 \cdot 10^{-9} + 400 \cdot 10^{-9}) + 0,02 \cdot (150 \cdot 10^{-9} + 8 + 400 \cdot 10^{-9}) \\
 &= 0,7 \cdot (1,5 \cdot 10^{-8}) + 0,28 \cdot (1,5 \cdot 10^{-8} + 4 \cdot 10^{-8}) + 0,02 \cdot (1,5 \cdot 10^{-8} + 8 + 4 \cdot 10^{-8}) \\
 &= 1,05 \cdot 10^{-8} + 1,54 \cdot 10^{-8} + 0,160011 \\
 &= 0,1627 \text{ ms}
 \end{aligned}$$

Shortest Remaining Time Next (SRTN, sistemi batch)

Supponiamo di avere i seguenti cinque processi con relative durate e tempi di arrivo:

processi	P1	P2	P3	P4	P5
durata	7	5	2	4	4
tempo di arrivo	0	3	4	6	9

Qual è il processo che sarà schedulato per ultimo?

Abbiamo detto che l'SRTN esegue prima i processi più veloci; essendo preemptive, nel momento in cui sta elaborando un processo ne arriva uno più veloce, sospende quello in corso e passa al più veloce. Quindi lo schema di esecuzione sarà (in grassetto, il processo a cui passa l'esecuzione):

t=0	t=3	t=4	t=6	t=9
P1=7	P1=4, P2=5	P1=3, P2=5, P3=2	P1=3, P2=5, P3=0, P4=4	P1=0, P2=5, P3=0, P4=4, P5=4

t=13	t=17	t=22
P1=0, P2=5, P3=0, P4=0, P5=4	P1=0, P2=5, P3=0, P4=0, P5=0	P1=0, P2=0, P3=0, P4=0, P5=0

Il processo schedulato per ultimo sarà P2.