1. Si consideri uno spazio di indirizzamento logico di otto pagine di 1024 parole ognuna, mappate su una memoria fisica di 32 frame.

   (a) Da quanti bit è costituito l'indirizzo logico?

   (b) Da quanti bit è costituito l'indirizzo fisico?

   Soluzione: siccome $8 = 2^3$, $1024 = 2^{10}$ e $32 = 2^5$, si ha:

   (a) indirizzo logico: 3 (n. di pagina) + 10 (offset nella pagina) = 13 bit,

   (b) indirizzo fisico: 5 (n. del frame) + 10 (offset nel frame) = 15 bit.

2. Si consideri la seguente *segment table*:

Segmento	Base	Lunghezza
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

   Quali sono gli indirizzi fisici corrispondenti ai seguenti indirizzi logici?

   - 0, 430
   - 1, 10
   - 2, 500
   - 3, 400
   - 4, 112

   Soluzione:

   - 0, 430 corrisponde all'indirizzo fisico 219+430=649 (indirizzo valido dato che 430<600);
   - 1, 10 corrisponde all'indirizzo fisico 2300+10=2310 (indirizzo valido dato che 10<14);
   - 2, 500 non corrisponde ad un indirizzo fisico valido dato che 500>100;
   - 3, 400 corrisponde all'indirizzo fisico 1327+400=1727 (indirizzo valido dato che 400<580);
   - 4, 112 non corrisponde ad un indirizzo valido dato che 112>96.

3. Si consideri un processo con text size = 2M, data size = 500K, stack size = 200K, process control block = 5K. La trap al kernel e ritorno impiega $1\mu sec$, e la CPU copia una parola di 4 byte, tra due locazioni di memoria, in 10 nsec.

   (a) Si dia una stima del tempo impiegato da una `fork()`, nel caso in cui il sistema operativo adotti la condivisione del codice ma non il copy-on-write.

   (b) Come sopra, ma con copy-on-write.

   (c) Per ottimizzare la memoria, conviene usare il copy-on-write sui segmenti o sulle pagine?

Soluzione:

(a) La quantità di memoria da copiare 500+200+5=705K, ossia 180.480 parole, per complessivi 1,8 msec, a cui bisogna aggiungere $1\mu sec$ per la system call (che non cambia sostanzialmente il valore).

(b) In questo caso, si tratta solo di copiare i 5K del PCB, quindi 1280*10 = 12800 nsec, a cui si aggiunge $1\mu sec$ per la system call, per un totale di $13,8\mu sec$.

(c) Sulle pagine, così basta allocare memoria soltanto per le pagine modificate e non gli interi segmenti. Questo può tuttavia causare più fault, e quindi un maggiore overhead per il meccanismo di trap al kernel.

4. Supponendo di avere un sistema con quattro frame e otto pagine, addottando una politica di rimpiazzamento FIFO, quanti page fault si verificheranno con la reference string seguente?

0 1 7 2 3 2 7 1 0 3

(Si assuma che i quattro frame siano inizialmente vuoti.) Ripetere il problema per LRU.

Soluzione: inizialmente FIFO caricherà nei quattro frame le pagine 0, 1, 7, 2 provocando quattro page fault. Poi scaricherà la pagina 0 e caricherà la pagina 3 (1 page fault) originando la coda 1, 7, 2, 3. I successivi riferimenti alle pagine 2, 7, 1 non provocheranno ulteriori page fault, lasciando la coda invariata. Il riferimento a 0 invece provocherà 1 page fault scaricando la pagina 1 e generando la coda 7, 2, 3, 0. Infine, l'ultimo riferimento alla pagina 3 non induce page fault, in quanto già presente in memoria. Quindi in totale si hanno 6 page fault. Simuliamo ora il funzionamento di LRU sulla reference string data:

	0	1	7	2	3	2	7	1	0	3
		0	1	7	2	3	2	7	1	0
			0	1	7	7	3	2	7	1
				0	1	1	1	3	2	7
					0	0	0	0	3	2
	P	P	P	P	P				P	P

Abbiamo quindi 7 page fault per LRU.

5. Si consideri un sistema con memoria paginata ad un livello, la cui page table sia mantenuta in memoria principale. Il tempo di accesso alla memoria principale sia $t = 50ns$.

(a) Qual è il tempo effettivo di accesso alla memoria?

(b) Aggiungendo un TLB, con tempo di accesso $\epsilon = 1ns$, quale hit rate dobbiamo avere per un degrado delle prestazioni del 5% rispetto a t?

(c) E con una paginazione a due livelli?

Soluzione:

(a) $EAT = 2t = 100$ns (50ns per recuperare il frame number, 50ns per l'indirizzo reale)

(b) Un degrado del 5% significa che $EAT = 1,05 * 50 = 52,5$ns. Ricordando che $EAT = \epsilon+\alpha t+(1-\alpha)(2t)$, abbiamo che $EAT = \epsilon+2t-\alpha t$, e quindi $\alpha = \frac{2t+\epsilon-EAT}{t} = \frac{100+1-52,5}{50} = 0,97$, ossia un hit rate del 97%.

(c) $EAT = 3t = 150$ns (50ns per l'accesso alla page table esterna, 50ns per l'accesso alla page table interna e 50ns per l'accesso al dato in memoria). Un degrado del 5% significa che $EAT = 1,05 * 50 = 52,5$ns. Ricordando che $EAT = \epsilon + \alpha t + (1-\alpha)(3t)$, abbiamo che $EAT = \epsilon + 3t - \alpha 2t$, e quindi $\alpha = \frac{3t+\epsilon-EAT}{2t} = \frac{150+1-52,5}{100} = 0,985$, ossia un hit rate del 98,5%.

6. (a) Si descriva brevemente il concetto di *working set* $WS(t, \Delta)$, all'istante t con intervallo Δ.

   (b) Si consideri la seguente stringa di riferimenti (partendo con $t = 0$):

   2 6 5 7 7 7 7 5 1 6 4

   Cosa è $WS(10, 8)$, ossia dopo l'ultimo accesso?

   (c) Nel precedente esempio, quanti page fault ci sono complessivamente con $\Delta = 4$ (supponendo che in ogni istante si mantenga in memoria esattamente il solo working set)?

Soluzione:

(a) Il working set è un'approssimazione della località del processo, ossia è l'insieme di pagine "attualmente" riferite. In generale $WS(t, \Delta)$=insieme delle pagine riferite negli accessi $[(t - \Delta + 1), t]$.

(b) $WS(10, 8) = \{1, 4, 5, 6, 7\}$

(c) 8 page fault. Basta fare la matrice seguente, facendo attenzione a togliere le pagine man mano che escono dal working set, e segnando fault quando bisogna (ri)mettercele:

2	6	5	7	7	7	7	5	1	6	4
2	6	5	7	7	7	7	5	1	6	4
	2	6	5	5	5		7	5	1	6
		2	6	6				7	5	1
			2						7	5
p	p	p	p				p	p	p	p

7. Si consideri un processo che generi la seguente stringa di riferimenti alle pagine virtuali:

   0 1 2 0 1 4 5 0 2 3

   (a) Se il processo ha 3 frame, gestiti con LRU, quanti page fault vengono generati?

   (b) Qual è il numero minimo di frame necessario per minimizzare i page fault?

Soluzione:

(a) Si generano 8 page fault:

	0	1	2	0	1	4	5	0	2	3
		0	1	2	0	1	4	5	0	2
			0	1	2	0	1	4	5	0
						2	0	1	4	5
							2	2	1	4
										1
	P	P	P			P	P	P	P	P

(b) Il minimo è 6 page fault (perché il processo accede a 6 pagine). Per determinare il numero minimo di frame per avere solo 6 page fault, si può procedere empiricamente, ripetendo la simulazione aumentando via via il numero di frame disponibili e fermandosi non appena si riscontrano solo 6 page fault sulla reference string data. Oppure si può sfruttare la *distance string*, che nel caso in questione risulta essere la seguente:

$$\infty \ \infty \ \infty \ 3 \ 3 \ \infty \ \infty \ 4 \ 5 \ \infty$$

Si ricorda che la distance string rappresenta la distanza fra la posizione di una pagina nel modello e la prima posizione, ovvero, quella nella prima riga della matrice (contando anche la casella di partenza) nel momento in cui la pagina stessa viene riferita. Se una pagina non è presente nella matrice, allora la sua distanza, quando viene riferita è ∞. Ad esempio nella simulazione della parte a) dell'esercizio, quando viene riferita inizialmente la pagina 0, la sua distanza è ∞ (dato che ancora non era presente nella matrice). Invece il secondo riferimento alla pagina 0 ha come distanza 3, in quanto la posizione precedentemente occupata dalla pagina 0 nella matrice si trova nella terza riga.

Indichiamo ora con C_i il numero di volte che il numero i compare nella distance string; nel caso in questione abbiamo: $C_1 = 0$, $C_2 = 0$, $C_3 = 2$, $C_4 = 1$, $C_5 = 1$, $C_\infty = 6$. Indicando poi con m il numero di frame e con n il numero più grande che compare nella distance string, indichiamo con $F_m = \Sigma_{k=m+1}^{n} C_k + C_\infty$ il numero di page fault che si verificano con m frame e con la reference string data. L'intuizione è la seguente: se ho a disposizione m frame i page fault saranno provocati dai riferimenti a pagine che "distano" almeno $m + 1$ dal top della matrice e dal numero di ∞ (ovvero da riferimenti a pagine non ancora presenti nel modello). Nel nostro caso abbiamo: $F_1 = 10$, $F_2 = 10$, $F_3 = 8$, $F_4 = 7$, $F_5 = 6$, quindi il numero minimo di frame che minimizza i page fault è 5.