

Cognome: _____ Nome: _____ Matricola: _____

Sistemi Operativi — A.A. 2007-2008, prova scritta dell'8 luglio 2008.

Usa questa pagina per la brutta, staccala, non consegnarla.

Sistemi Operativi — A.A. 2007-2008, prova scritta dell'8 luglio 2008.

Usa questa pagina per la brutta, staccala, non consegnarla.

Cognome: _____ Nome: _____ Matricola: _____

Sistemi Operativi — A.A. 2007-2008, prova scritta dell'8 luglio 2008.

Libri e appunti chiusi. Vietato comunicare con chiunque. Vietato l'uso di cellulari, calcolatrici, palmari e affini. Tempo a disposizione: 60 minuti.


1. Considera un sistema con architettura del kernel "execution within user process". In tale sistema sono presenti tre processi: **A**, **B**, **C**, inizialmente tutti e tre in coda ready nell'ordine **A** in testa poi **B** poi **C**. La politica di scheduling è **round robin** con quanto q .

A è cpu bound e genera sempre un page fault dopo $0.5q$, il page fault viene servito in tempo $0.3q$.

B è I/O bound (cpu burst trascurabili, i/o servito in $0.7q$).

C è puramente cpu bound e non provoca page faults.

Il processore esegue di volta in volta **A**, **B**, **C**, e inoltre, con tempi trascurabili, il codice per mode switching, dispatching, system call e interrupt handlers. Mostra schematicamente, nella seguente tabella, l'ordine con cui tali attività vengono eseguite (una sola croce per ciascuna colonna). Indica anche quali processi sono running, quali ready e quali bloccati in ciascun istante come indicato nell'esempio.

tempo 

proc. in user mode	A	x																		
	B																			
	C																			
mode switch		x																		
kernel mode	dispatching			x																
	system call i/o																			
	interrup handler per page fault			x																
	interrup handler per i/o																			
	interrup handler per q scaduto																			
stati processi	running	A	A	A																
	ready	B	B	B																
		C	C	C																
	in blocco			A																

2. Rispondi brevemente alle seguenti domande sulle politiche di page replacement.

Che significa page buffering?

Sistemi Operativi — A.A. 2007-2008, prova scritta dell'8 luglio 2008.

Che significa “minor page fault” in questo contesto?

Elenca i vantaggi di page buffering per la gestione delle pagine modificate?

3. Considera una architettura Pentium-like: pagina di 4 KB, paginazione a due livelli, pte 4 byte, root page table sempre in memoria. Il frammento di codice assembly mostrato è composto da 2 istruzioni che vengono eseguite consecutivamente. All'inizio dell'esecuzione la prima locazione libera puntata dallo stack pointer è 0xddaff001 e lo stack cresce **verso il basso, cioè verso indirizzi minori**. Calcola quanti page fault può generare al più ciascuna istruzione durante l'esecuzione del frammento in questione nelle fasi di fetch e di esecuzione. Considera le istruzioni eseguite di seguito e supponi che le pagine caricate dalle istruzioni precedenti permangano residenti durante l'esecuzione delle istruzioni successive.

Indirizzo	lunghezza	istruzione	Page faults dovuti a parti di page table non residenti		Page faults dovuti a codice o dati non residenti	
			fetch	execute	fetch	execute
0x11bffffd	5	carica nel registro A 4 byte a partire da 0xddaffffe				
0x11c00001	1	push del contenuto di A nello stack				

Supponi che il TLB sia inizialmente vuoto, cosa conterrà dopo l'esecuzione di tali istruzioni?

Cognome: _____ Nome: _____ Matricola: _____

Sistemi Operativi — A.A. 2007-2008, prova scritta dell'8 luglio 2008.

Come viene trattato il TLB in caso di process switch?

4. Descrivi l'algoritmo di **disk scheduling** “elevator”.

Rispetto a quale parametro elevator è unfair? Perché? C'è modo di rendere elevator fair?

Perché viene introdotto la tecnica del “**request merging**” e quali sono i vantaggi?