Homework HWT di Programmazione Concorrente
17 novembre — anno accademico 2016-2017

Cognome: Nome: Matricola:

Durata: 30 minuti. Non si può consultare documentazione. Usare solo ed esclusivamente i fogli che sono stati consegnati.

L'homework è composto di 15 domande a risposta multipla. Esiste una sola risposta esatta a domanda. La valutazione della singola domanda sarà così articolata: +1 risposta esatta; 0 nessuna risposta o risposta multipla; -0.5 per le risposte sbagliate. La valutazione complessiva si ottiene con:

$$\text{Valutazione} = 3 + 1 \times \#\text{esatte} - \frac{1}{2} \times \max(\#sbagliate - 3, 0).$$

Si precisa che non è necessario rispondere a tutte le domande e che non sono ammesse correzioni.

Domande

• Perché le sequenze di interleaving risultano un modello meno generale delle sequenze di esecuzione ammissibili?

- ◦ perché assumono l'indivisibilità delle istruzioni eseguite e che la piattaforma sia uni-processore
- ◦ assumono che l'esecuzione di una istruzione possa iniziare solo prima che la precedente sia terminata
- ◦ assumono che l'esecuzione di una istruzione possa terminare solo prima che la successiva sia cominciata
- ◦ assumono che la piattaforma sia multi-processore
- ◦ nessuna delle precedenti

• Si considerino questi due programmi formati da istruzioni indivisibili ed alcune possibili sequenze di interleaving risultanti da una loro esecuzione su di una architettura mono-processore. Sulla base delle condizioni di Bernstein, quali sequenze evidenzieranno la conseguente inteferenza

Primo Prg	*Secondo Prg*	*Sequenze di interleaving*
$x)T \leftarrow POSTI;$	$r)R \leftarrow POSTI - 1;$	$1)xryzst$
$y)T \leftarrow T + 1;$	$s)Q \leftarrow R;$	$2)xyzrst$
$z)POSTI \leftarrow T;$	$t)POSTI \leftarrow R;$	$3)rsxyzt$

- tutte
- solo la 1 non genera fenomeni di interferenza
- solo la 2 non genera fenomeni di interferenza
- solo la 3 non genera fenomeni di interferenza
- nessuna delle precedenti

• Quale delle seguenti affermazioni è conseguenza diretta dell'"Assunzione di Progresso Finito""?

- la particolare sequenza di interleaving adottata dagli esecutori fisici può essere prevista sulla base di argomentazioni probabilistiche
- è possibile basare la correttezza dei propri programmi concorrenti facendo ipotesi sulle sequenze di interleaving scelte dall'esecutore fisico
- è possibile evitare i fenomeni di interferenza inserendo delle pause artificiali di durata prestabilita
- è possibile basare la correttezza dei propri programmi concorrenti facendo ipotesi sulle sequenze di esecuzione ammissibili scelte dall'esecutore fisico
- nessuna delle precedenti

• Quali sono i vantaggi e le motivazioni dietro l'assunzione che tutte le istruzioni utilizzate, in qualsiasi linguaggio di programmazione di qualsiasi livello di astrazione, siano *non atomiche*, tranne esplicito avviso in senso contrario?

- risulta un modello adatto allo studio di sistemi di *flussi di esecuzione sequenziali debolmente connessi*
- risulta essenziale per poter dettagliare per ogni specifica piattaforma hardware/software di riferimento quali istruzioni sono veramente atomiche
- risulta un modello verosimile delle operazioni offerte dall'hardware moderno, che però induce a costruire soluzioni dipendenti dalla piattaforma utilizzata
- in questo modo possiamo supporre che gli aggiornamenti di strutture composite avvengano sempre atomicamente
- nessuna delle precedenti

• I fenomeni di interferenza si possono manifestare:

- se nessun flusso di esecuzione effettua scritture
- se i flussi di esecuzione eseguono solo istruzioni atomiche
- se gli stati transienti risultano visibili ma non sono acceduti da flussi di esecuzione diversi da quello che li producono
- se esistono flussi di esecuzione che producono stati transienti
- nessuna delle precedenti

• Per quale motivo nella implementazione delle primitive P e V di Dijkstra per piattaforme uniprocessore risulta superfluo utilizzare gli spin lock?

- perché l'indivisibilità delle due primitive è comunque ottenibile disabilitando le interruzioni
- per evitare di minare la reattività del sistema operativo utilizzato
- per non dover effettuare attese attive
- per garantire la fairness delle primitive
- nessuna delle precedenti

• Si consideri l'esecuzione concorrente da parte di $n > 1$ flussi di esecuzione su una architettura multi-processore di questo codice:
loop P(S) <<*sezione critica*>> V(S) **end loop**
I flussi eseguono indefinitivamente e ripetutamente la stessa identica sezione critica protetta da un semaforo binario di Dijkstra S implementato tramite spin-lock basato su una istruzione atomica di tipo *TestAndSet*. Si confronti il comportamento di questi flussi in presenza della semantica astratta delle due primitive di Dijkstra (P e V) ed in presenza della implementazione concreta:

- entrambe garantiscono la proprietà di fairness dell'esecuzione concorrente
- nessuna garantisce la proprietà di fairness dell'esecuzione concorrente

- la semantica astratta ma non quella concreta garantisce la proprietà di fairness dell'esecuzione concorrente
- la semantica concreta ma non quella astratta garantisce la proprietà di fairness dell'esecuzione concorrente
- nessuna delle precedenti

• Con riferimento all'implementazione delle primitive P e V di Dijkstra e confrontando le caratteristiche di ipotetiche implementazioni basate su attese attive oppure su attese passive e tecniche di contex-switch:

- quelle basate su attese attive divengono via via più convenienti all'aumentare della durata delle sezioni critiche e del livello di compezione
- quelle basate su attese passive e context-switch divengono via via più convenienti all'aumentare della durata delle sezioni critiche e del livello di compezione
- quelle basate su attese attive sono sempre più convenienti su piattaforme uni-processor
- quelle basate su attese passive e context-switch sono sempre più convenienti su piattaforme multi-processor
- nessuna delle precedenti

• Con riferimento alla chiamata di sistema `pthread_cond_wait(pthread_cond_t *cond, pthread_mutex_t *mutex)`, per quale motivo riceve come parametro anche il puntatore al mutex associato alla variabile condizione?

- per controllare che il mutex risulti libero al momento precedente l'invocazione e quindi già disponibile ad altri thread che volessero effettuare segnalazioni accedendo in mutua esclusione alla risorsa condivisa
- perché quando il controllo ritorna al thread che effettua la chiamata il mutex va sbloccato in quanto la segnalazione è oramai già avvenuta e catturata
- affinché il mutex sia sbloccato durante l'attesa passiva, di modo che anche altri thread che accedono in mutua esclusione alla risorsa sottostante possano effettuare segnalazioni sulla variabile condizione.
- affinché nessun thread acceda alla risorsa condivisa durante l'attesa passiva di modo da non corrompere il suo stato consistente visibile al momento della chiamata
- nessuna delle precedenti

• Quali sono gli svantaggi della tecnica di prevenzione dello stallo consistente nell'invalidare la condizione di non-prerilasciabilità delle risorse:

- non è applicabile a tutti i tipi di risorse
- la tecnica prevede di rilasciare forzatamente le risorse acquisite anche in assenza di competizione per il loro utilizzo
- il degrado di rendimento complessivo in assenza di competizione
- comporta il prerilascio delle sole risorse prerilasciabili già acquisite ogni qualvolta una richiesta viene rifiutata
- nessuna delle precedenti

• Si consideri una soluzione al problema dei cinque filosofi mangiatori in cui i filosofi richiedono incrementalmente le due forchette; quattro filosofi richiedono sistematicamente prima la forchetta di sx e poi quella di dx, mentre l'ultimo, mancino, richiede prima la forchetta di dx e poi quella di sx:

- la soluzione presenta pericolo di stallo
- la soluzione gode della proprietà di fairness
- la soluzione può comportare starvation
- la soluzione non consente il massimo livello di parallelismo
- nessuna delle precedenti

• Si consideri la soluzione al problema classico di P produttori, C consumatori, ed un buffer intermedio capace di ospitare M messaggi, che fa utilizzo di due semafori binari e due semafori a conteggio. Quali sequenze di interleaving sono consentite da questa soluzione?

- quelle in cui due produttori producono concorrentemente su celle distinte di un buffer già pieno
- quelle in cui due consumatori consumono concorrentemente da celle distinte di un buffer contenente almeno due messaggi
- quelle in cui due produttori producono concorrentemente sulla stessa cella di un buffer non pieno
- quelle in cui due consumatori consumono concorrentemente dalla stessa cella di un buffer non vuoto
- nessuna delle precedenti

• Si consideri la semantica delle regioni critiche condizionali, in uno scenario in cui un precedente f.d.e. sta uscendo dalla propria sezione critica, ci sono molteplici f.d.e. in attesa passiva su code di diverse condizioni, e, per finire, altri f.d.e. cercano di entrare dall'esterno della regione in sezione critica. Quale criterio prevede la semantica delle regioni critiche condizionali per scegliere un nuovo f.d.e. da mandare in sezione critica?

- si sceglie il primo f.d.e. della coda dei f.d.e. provenienti dall'esterno
- si sceglie non deterministicamente una condizione soddisfatta e si estrae il primo f.d.e. della sua coda
- si sceglie non deterministicamente una condizione, anche se non soddisfatta, e si estrae il primo f.d.e. della sua coda
- si sceglie non deterministicamente un f.d.e. tra quelli in cima ad una delle code la cui condizione risulta soddisfatta, oppure tra quelli provenienti dall'esterno.
- nessuna delle precedenti

• Quale dei seguenti è un limite dei monitor come strumento per la scrittura di codice concorrente?

- il basso livello di astrazione
- il loro utilizzo induce a disperdere il codice inerente il medesimo concetto
- il loro utilizzo induce a produrre soluzioni concorrenti conservative rispetto al livello di parallelismo raggiungibile
- la difficoltà di associare il concetto di monitor alle risorse condivise
- nessuna delle precedenti

• Con riferimento ai monitor muniti di primitive wait e signal di Hoare, quale delle seguenti affermazioni risulta corretta?

- le primitive di Hoare sono pensate per risolvere specificatamente problemi di competizione
- l'unica semantica corretta della signal è quella secondo la quale la signal comporta il rilascio immediato del monitor
- l'unica semantica corretta della signal è quella secondo la quale la signal non comporta il rilascio immediato del monitor
- l'unica semantica corretta della signal è quella secondo la quale la signal deve essere obbligatoriamente collocata come ultima istruzione di una entry-procedure
- nessuna delle precedenti