Elementi di Architettura e Sistemi Operativi

Bioinformatica - Tiziano Villa

16 Febbraio 2018

Nome e Cognome:
Matricola:
Posta elettronica:

problema	punti massimi	i tuoi punti
problema 1	10	
problema 2	5	
problema 3	5	
problema 4	10	
totale	30	

1. Si consideri il problema di sincronizzazione cosiddetto degli n filosofi, dove n filosofi siedono attorno a una tavola con una bacchetta tra ogni coppia di filosofi vicini.

   I filosofi sono numerati da 0 a $n-1$ e ad essi corrispondono processi diversi, cioe' ogni filosofo esegue $Pranza(i)$, dove i e' il numero del filosofo. Si assuma che ci sia un vettore di semafori, $Bacchetta[i]$ che rappresenta la bacchetta alla sinistra del filosofo i. Tutti i semafori sono inizializzati a 1.

   Si consideri la seguente soluzione:

```
void Pranza(int i) {
 Bacchetta[i].P();   /* prendi la bacchetta sinistra */
 Bacchetta[(i+1)%n].P();/* prendi la bacchetta destra */
 Mangia();
 Bacchetta[i].V();   /* lascia la bacchetta sinistra */
 Bacchetta[(i+1)%n].V();/* lascia la bacchetta destra */
}
```

   (a) Si spieghi come funziona questo codice di sincronizzazione.
   Traccia do soluzione.
   I semafori garantiscono la sincronizzazione nell'accesso alle bacchette.
   Si veda il libro di testo.

(b) Questa soluzione soddisfa le condizioni necessarie per lo stallo ? Si elenchino le condizioni di stallo, e per ogni condizione di stallo si verifichi se essa e' soddisfatta oppure o no in questa soluzione.

Traccia di soluzione.

- Mutua esclusione.
  I semafori sono inizializzati a 1, percio' ogni bacchetta puo' essere detenuta da un solo processo per volta.
- Niente prelazione.
  Le bacchette non possono essere tolte via da chi le detiene senza violare la semantica dei semafori su cui si basa il codice precedente.
- Possesso e attesa.
  In uno stallo, la seconda $P()$ nel codice proposto ha come effetto che il processo attenda mentre detiene la prima bacchetta ottenuta con la prima $P()$.
- Attesa circolare.
  Il filosofo i afferra la $Bacchetta[i]$ e aspetta che il filosofo $(i+1)\%n$ rilasci la $Bacchetta[(i+1)\%n]$.
  C'e' un ciclo chiuso dal filosofo $n-1$ quando afferra la $Bacchetta[n-1]$ e aspetta che il filosofo 0 rilasci la $Bacchetta[0]$.

(c) Questa soluzione puo' entrare in stallo ? Se no, si argomenti perche' no. Se si, si mostri una successione di chiamate del processo $Pranza()$ (indicando il relativo argomento) che porta allo stallo.

Traccia di soluzione.

Si, puo' entrare in stallo perche' le condizioni necessarie sono soddisfatte, come visto al punto precedente.

Si ha uno stallo quando si esegue $Pranza(0)$ e tale chiamata e' sospesa dopo la prima $P()$, poi si esegue $Pranza(1)$ e tale chiamata e' sospesa dopo la prima $P()$, poi si esegue $Pranza(2)$ e tale chiamata e' sospesa dopo la prima $P()$, poi si esegue $Pranza(3)$ e tale chiamata e' sospesa dopo la prima $P()$, poi si esegue $Pranza(4)$ e tale chiamata e' sospesa dopo la prima $P()$. A questo punto ogni filosofo ha afferrato la sua bacchetta sinistra, ma non ci sono altre bacchette libere, tutti i filosofi rimangono bloccati sulla seconda $P()$ in attesa che un altro filosofo rilasci la sua bacchetta, senza che nessuno possa mangiare e quindi rilasciare le bacchette detenute.

2. (a) Si spieghi come il meccanismo di traduzione degl'indirizzi da logici a fisici permette di proteggere la memoria di un processo da un altro processo.

   Traccia di soluzione.

   Si veda il libro di testo per una discussione sulla traduzione degl'indirizzi.

   Il sistema operativo garantisce che la medesima pagina fisica non sia assegnata a processi diversi. Soltanto il sistema operativo puo' definire la conversione da indirizzi logici a indirizzi fisici, impedendo ai singoli processi d'interferire con la memoria assegnata agli altri processi.

   (b) Generalmente, per una tavola delle pagine piena (cioe' si usano tutti i suoi elementi) succede che aumentando il numero di livelli della tavola delle pagine aumenta la dimensione della tavola delle pagine.

   Si mostri che questo non e' sempre vero se la tavola delle pagine e' sparsa (cioe' non si usano tutti i suoi elementi).

   Per concretizzare il ragionamento, si faccia riferimento a un indirizzo logico di 32 cifre binarie con pagine di 4 KB.

   Traccia di soluzione.

   Dato che le pagine sono di 4 KB servono 12 cifre binarie per indirizzare una parola al loro interno, e quindi rimangono $32 - 12 = 20$ cifre binarie per indirizzare le pagine, che sono in totale 2^{20}.

   Si consideri un processo che richiede solo una pagina (che ad es. utilizza solo riferimenti ad indirizzi molto alti).

   Una tavola delle pagine a un solo livello ha ancora bisogno di 2^{20} elementi (per ogni possibile indirizzo virtuale).

   Invece, una tavola delle pagine a 20 livelli in questo caso ha bisogno di una sola tavola delle pagine ad ogni livello costituita da 2 elementi, per un totale di 40 elementi.

3. Si considerino le seguenti due istruzioni di LC-3:

   `A: 0000111101010101`

   `B: 0100111101010101`

   Si spieghi in dettaglio che cosa fanno queste due istruzioni, sottolineando le somiglianze e differenze dei loro comportamenti.

   Traccia di soluzione

   Le istruzioni sono:

   ```
   A: BRnzp -171
   B: JSR -171
   ```

   Sia A che B cambiano incondizionatamente il contenuto del PC in $(PC + 1) - 171$ (si noti che la prima salta sempre perche' per forza una delle tre condizioni deve essere vera, indicando tutti e tre i casi di positivo, negativo o nullo). Tuttavia, B salva in $R7$ l'indirizzo di ritorno (cioe' $PC + 1$), mentre A non modifica $R7$.

4. (a) Si rappresentino la seguente funzione e il suo complemento con una SP (somma di prodotti) e una PS (prodotto di somme) minimizzate:
$F = \overline{a}\overline{b}\overline{c}\overline{d} + \overline{a}\overline{b}\overline{c}d + \overline{a}\overline{b}c\overline{d} + \overline{a}\overline{b}cd + a\overline{b}\overline{c}\overline{d} + a\overline{b}cd$.
Si mostrino le mappe di Karnaugh delle funzioni.
Traccia di soluzione.
SP: $F = \overline{a}\overline{b} + \overline{b}\,\overline{d}$.
SP: $\overline{F} = b + ad$.
PS: $F = \overline{b}(\overline{a} + \overline{d})$.
PS: $\overline{F} = (a + b)(b + d)$.
[La funzione proposta non dipende dalla variabile c, come le forme minime dimostrano.]

(b) Si progetti un circuito sequenziale che realizza un addizionatore seriale con la seguente specifica:

- Ci sono due variabili binarie in ingresso x e y, e una variabile binaria in uscita z.
- Le variabili in ingresso rappresentano due operandi binari da sommare letti a partire dalla cifra binaria meno significativa.
- La variabile in uscita rappresenta la somma binaria prodotta a partire dalla cifra binaria meno significativa.
- Si noti che non c'e' un limite prefissato al numero di cifre binarie degli addendi letti in ingresso.
- Per semplicita' si trascuri il riporto finale, quindi se i due ingressi hanno n cifre binarie, il circuito genera un'uscita con n cifre binarie (senza generare in uscita il riporto finale che corrisponderebbe alla $n+1$-esima cifra binaria dell'uscita).

i. Si progetti la macchina a stati finiti che modella la specifica (tipo Mealy) disegnando il grafo delle transizioni. S'indichi lo stato iniziale.

ii. Si minimizzi il numero degli stati della macchina proposta, applicando l'algoritmo di minimizzazione degli stati.

iii. Si scriva la tavola delle transizioni con gli stati futuri e le uscite e la si codifichi.

Traccia di soluzione.

Tabella delle transizioni della macchina a stati finiti:

```
00 sa sa 0
01 sa sa 1
11 sa sb 0
10 sa sa 1

00 sb sa 1
01 sb sb 0
11 sb sb 1
10 sb sb 0
```

iv. Supponendo di usare bistabili di tipo D, si derivino le equazioni minimizzate di eccitazione degl'ingressi dei bistabili e le equazioni minimizzate delle uscite. Si esegua e mostri la minimizzazione con le mappe di Karnaugh.

v. Si realizzi il circuito sequenziale corrispondente con bistabili di tipo D campionati sul fronte di salita, invertitori e porte NAND. Si etichettino con chiarezza i segnali.