

Elementi di Architettura e Sistemi Operativi

Bioinformatica - Tiziano Villa

26 Febbraio 2016

Nome e Cognome:

Matricola:

Posta elettronica:

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

1. Si consideri la seguente procedura di *trasferimento atomico* che rimuove un elemento da una coda e l'aggiunge ad un'altra. Il *trasferimento* deve apparire come *atomico*, cioè non ci deve essere un intervallo di tempo in cui un processo esterno possa determinare che un elemento è stato rimosso da una coda ma non ancora aggiunto a un'altra. La realizzazione deve essere concorrente, cioè deve permettere trasferimenti tra code multipli in parallelo.

```
void AtomicTransfer (Queue *queue1, *queue2) {
    Item thing; /* oggetto in trasferimento */

    queue1->lock.Acquire();
    thing = queue1->Dequeue();
    if (thing != NULL) {
        queue2->lock.Acquire();
        queue2->Enqueue(thing);
        queue2->lock.Release();
    }
    queue1->lock.Release();
}
```

Si argomenti se la procedura (i) funziona sempre, oppure (ii) non funziona mai, oppure (iii) talvolta funziona talvolta no. Per i casi (ii) o (iii) si proponga una soluzione.

Traccia di soluzione.

È il caso (iii): in certi casi non funziona, perché può (non necessariamente deve) condurre a un blocco. Quando un primo processo trasferisce un elemento dalla coda A a quella B, e un secondo processo trasferisce da B a A, si può avere un blocco se entrambi i processi acquisiscono il lucchetto sulla prima coda prima che l'altro processo lo acquisisca sulla seconda. Similmente per blocchi ciclici con più trasferimenti: ad es. un primo processo trasferisce un elemento dalla coda A a quella B, un secondo processo trasferisce da B a C e un terzo processo trasferisce da C a A, si può avere un blocco se tutti i processi acquisiscono il lucchetto sulla prima coda prima che alcun processo lo acquisisca sulla seconda.

La miglior soluzione per massimizzare la concorrenza è di acquisire entrambi i lucchetti all'inizio della procedura, con un ordinamento delle code per cui si acquisisce per prima la coda con l'indice più basso.

```

void AtomicTransfer (Queue *queue1, *queue2) {
    Item thing; /* oggetto in trasferimento*/

    if (queue1 < queue2) {
        queue1->lock.Acquire();
        queue2->lock.Acquire();
    } else { /* queue2 < queue1
        queue2->lock.Acquire();
        queue1->lock.Acquire();
    }
    thing = queue1->Dequeue();
    if (thing != NULL) {
        queue2->Enqueue(thing);
    }
    /* l'ordine di rilascio non importa */
    queue1->lock.Release();
    queue2->lock.Release();
}

```

Altre soluzioni meno efficienti in quanto a concorrenza: definire un lucchetto globale per l'intera procedura, o meglio definire un lucchetto globale solo per l'acquisizione dei due lucchetti delle code e poi rilasciarlo prima di togliere e aggiungere l'elemento alle code. Tuttavia non e' una soluzione molto concorrente: per es., se un processo deve aspettare per acquisire un lucchetto di una coda, allora dal momento che detiene il lucchetto globale, nessun altro processo puo' acquisire lucchetti fino a che il primo non ne completa l'acquisizione.

Un errore comune era dire che il codice originale non permetteva un trasferimento atomico, perche' ad es. un altro processo poteva accedere alla coda2 dopo che un elemento e' stato rimosso dalla coda1. Tuttavia, pur se l'elemento e' stato rimosso dalla coda1 e non ancora aggiunto alla coda2, tale processo esterno non puo' osservare la situazione, poiche' non puo' ottenere un lucchetto sulla coda1 e percio' non puo' sapere se l'elemento e' stato rimosso dalla coda1 oppure no. Quindi dal punto di vista del processo esterno, il trasferimento e' atomico, cioe' il processo esterno vede lo stato del sistema, o prima che si era effettuato il trasferimento, o dopo, ma non durante.

2. Siano dati 4 processi con durata dell'esecuzione e tempo d'arrivo espressi dalla seguente tabella:

| Processo | Durata | Arrivo |
|----------|--------|--------|
| P1 | 3 | 0 |
| P2 | 5 | 1 |
| P3 | 2 | 3 |
| P4 | 2 | 9 |

- (a) Si disegni lo schema GANTT (come nel libro di testo) che illustri l'esecuzione di questi processi con i seguenti algoritmi di schedulazione: FCFS (First Come First Serve), SRTF (Shortest-Remaining-Time-First, cioè Shortest-Job-First con prelazione), RR (Round-Robin) con quanto di tempo = 1.

Traccia di soluzione.

FCFS:

```
P1 P1 P1 P2 P2 P2 P2 P2 P3 P3 P4 P4
0  1  2  3  4  5  6  7  8  9 10 11
```

SRTF:

```
P1 P1 P1 P3 P3 P2 P2 P2 P2 P2 P4 P4
0  1  2  3  4  5  6  7  8  9 10 11
```

RR:

```
P1 P2 P1 P3 P2 P1 P3 P2 P2 P4 P2 P4
0  1  2  3  4  5  6  7  8  9 10 11
```

- (b) Per i tre algoritmi precedenti si calcoli il tempo di completamento di ciascun processo. Si definisca il tempo di completamento come la differenza tra il tempo di terminazione e il tempo d'arrivo.

Traccia di soluzione.

Il tempo di completamento e' la differenza tra il tempo di terminazione e il tempo d'arrivo, cioe' il tempo trascorso da quando il processo arriva nella coda a quando termina. Nella letteratura in inglese e' designato anche come TRT (turnaround time), cioe' tempo trascorso.

Per FCFS i tempi di completamento sono:

P1 3,

P2 7,

P3 7,

P4 3

Per SRTF i tempi di completamento sono:

P1 3,

P2 9,

P3 2,

P4 3

Per RR i tempi di completamento sono:

P1 6,

P2 10,

P3 4,

P4 3

- (c) Per i tre algoritmi precedenti si calcoli il tempo di attesa di ciascun processo. Si definisca il tempo di attesa come la differenza tra il tempo di completamento e la durata.

Traccia di soluzione.

Il tempo di attesa e' la differenza tra il tempo di completamento e la durata, cioe' e' il tempo trascorso in attesa nella coda senza eseguire.

Per FCFS i tempi di attesa sono:

P1 0,

P2 2,

P3 5,

P4 1

Per SRTF i tempi di attesa sono:

P1 0,

P2 4,

P3 0,

P4 1

Per RR i tempi di attesa sono:

P1 3,

P2 5,

P3 2,

P4 1

- (d) Per i tre algoritmi precedenti si calcoli il tempo di attesa medio per l'esecuzione dei quattro processi.

Traccia di soluzione.

Tempo di attesa medio per FCFS: $(0+2+5+1)/4 = 2$.

Tempo di attesa medio per SRTF: $(0+4+0+1)/4 = 1,25$.

Tempo di attesa medio per RR: $(3+5+2+1)/4 = 2,75$.

Si noti che SRTF ottiene il tempo di attesa medio minimo come dalla teoria.

3. Si consideri una memoria con una cache. Il tempo di accesso della cache T_c e' di 100 ns e il tempo di accesso della memoria T_m di 1200 ns. Se il tempo di accesso effettivo T_e del sistema memoria+cache e' del 10% maggiore del tempo di accesso della cache, quale deve essere la percentuale di successo S ?

Traccia di soluzione.

La formula per il tempo di accesso effettivo e'

$$T_e = S \times T_c + (1 - S)(T_m + T_c),$$

dove $T_c = 100 \text{ ns}$, $T_e = 1,1 \times T_c$, and $T_m = 1200 \text{ ns}$.

Sostituendo i valori si ha

$$1,1 \times 100 = 100S + (1 - S)(1200 + 100)$$

$$110 = 100S + 1300 - 1300S$$

$$1200S = 1190$$

$$S = 119/120 \approx 99,1\%.$$

4. Si consideri il seguente programma scritto nel linguaggio macchina LC-3.

```
                .ORIG    x3000
                LD        R1, NUM
START           ADD      R1, R1, #-1
                BRz       DONE
                JMP       START

DONE            TRAP      x25
NUM             .FILL     x000A
                .END
```

Si spieghi con chiarezza il funzionamento di tale programma, in base alle istruzioni del suo codice.

Traccia di soluzione

Il programma carica in R1 il numero 10 in esadecimale e poi lo decrementa di un'unità per volta fino al numero 0.

```
                .ORIG    x3000 ; il programma conta da 10 a 0
                LD        R1, NUM
START           ADD      R1, R1, #-1
                BRz       DONE
                JMP       START

DONE            TRAP      x25    ; ultima istruzione eseguibile
NUM             .FILL     x000A ; 10 in compl. a 2, esadecimale
                .END          ; pseudo-op, delimita
                               ; il sorgente del programma
```

5. Si progetti un circuito sequenziale che realizza la seguente specifica: prende in ingresso un intero in binario di lunghezza arbitraria, **una cifra binaria per volta** incominciando dalla cifra binaria piu' significativa, e produce il resto della divisione dell'intero per 3. Per esempio, se l'intero in ingresso e' $110101_2 = 53_{10}$, l'uscita deve essere $10_2 = 2_{10}$, poiche' $53/3$ da' resto 2 (si veda lo schema in Fig. 1).

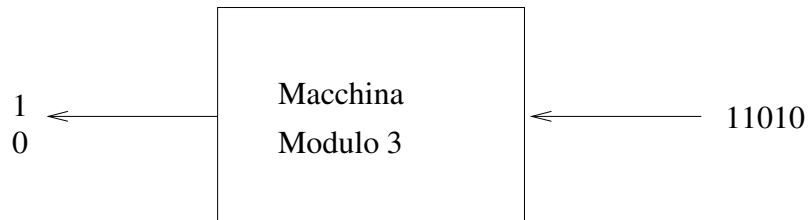


Figure 1: Schema ingresso-uscita della macchina modulo3.

- (a) Si disegni il grafo delle transizioni di una macchina a stati finiti che realizza la specifica. S'indichi lo stato iniziale.

Traccia. Il circuito ha un solo ingresso. Si supponga che lo stato stesso rappresenti la risposta. Quanti stati sono sufficienti ? Si noti che dopo che sono stati lette le prime n cifre binarie, la macchina sara' in uno stato che rappresenta il resto della divisione per 3 del numero rappresentato da quelle n cifre binarie.

Si veda la soluzione nel documento allegato.

- (b) Si minimizzi il numero degli stati della macchina proposta, applicando l'algoritmo di minimizzazione degli stati.

- (c) Si scriva la tavola delle transizioni con gli stati futuri e le uscite e la si codifichi.

- (d) Supponendo di usare bistabili di tipo D, si derivino le equazioni minimizzate di eccitazione degli ingressi dei bistabili e le equazioni minimizzate delle uscite. Si esegua e mostri la minimizzazione con le mappe di Karnaugh.

- (e) 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.