

Elementi di Architettura e Sistemi Operativi

Bioinformatica - Tiziano Villa

27 Settembre 2019

Nome e Cognome:

Matricola:

Posta elettronica:

problema	punti massimi	i tuoi punti
problema 1	9	
problema 2	6	
problema 3	5	
problema 4	10	
totale	30	

1. Si considerino le seguenti quattro varianti di soluzione con i semafori del problema del produttore-consumatore. Per ciascuna, si risponda alla domanda se e' corretta oppure no (esempio, puo' verificarsi un blocco), argomentando con chiarezza la risposta.

(a) Soluzione n. 1

```
Producer {} {  
    emptyBuffers.P();  
    mutex.P();  
    produce un'unita' di prodotto;  
    fullBuffers.V();  
    mutex.V();  
}
```

```
Consumer {} {  
    fullBuffers.P();  
    mutex.P();  
    consuma un'unita' di prodotto;  
    emptyBuffers.V();  
    mutex.V();  
}
```

Traccia di soluzione.

Corretta, anche se scambia i V rispetto a quella corretta proposta nelle dispense, che e' la Soluzione n. 4. Puo' essere meno efficiente della soluzione n. 4, in quanto potrebbe succedere che un produttore fosse interrotto tra `fullBuffers.V()` e `mutex.V()` e quindi che un consumatore pur essendoci un pieno (`fullBuffers.P()` eseguito con successo) non potrebbe consumare essendo il mutex ancora detenuto dal produttore; tuttavia prima o poi il produttore sarebbe riattivato, eseguirebbe `mutex.V()`, permettendo al consumatore di continuare oltre `fullBuffers.P()`, eseguire `mutex.P()` etc.

(b) Soluzione n. 2

```
Producer {} {  
    mutex.P();  
    emptyBuffers.P();  
    produce un'unita' di prodotto;
```

```

    fullBuffers.V();
    mutex.V();
}

Consumer {} {
    mutex.P();
    fullBuffers.P();
    consuma un'unita' di prodotto;
    emptyBuffers.V();
    mutex.V();
}

```

Traccia di soluzione.

Scorretta, può portare a un blocco. Se la coda è inizialmente piena, arriva un produttore che afferra la variabile mutex e poi deve aspettare un consumatore sul semaforo emptyBuffers.P(), ma nessun consumatore può consumare perché impossibilitato ad afferrare mutex detenuto dal produttore. Per cui il sistema si blocca. Si può arrivare a un blocco anche dalla condizione simmetrica in cui la coda è inizialmente vuota, un consumatore afferra mutex e si blocca su fullBuffers.B() e quindi non permette a un produttore di produrre etc.

(c) Soluzione n. 3

```

Producer {} {
    mutex.P();
    emptyBuffers.P();
    produce un'unita' di prodotto;
    fullBuffers.V();
    mutex.V();
}

Consumer {} {
    fullBuffers.P();
    mutex.P();
    consuma un'unita' di prodotto;
    mutex.V();
    emptyBuffers.V();
}

```

```
}
```

Traccia di soluzione.

Scorretta, può portare a un blocco. Argomento come nel caso precedente per la coda inizialmente piena e l'arrivo di un produttore che prende mutex e si ferma su emptyBuffers.P() non permettendo a un consumatore di consumare. In questo esempio non si ha il blocco a partire dalla coda vuota perché non c'è lo scambio di P nel consumatore.

(d) Soluzione n. 4

```
Producer {} {  
    emptyBuffers.P();  
    mutex.P();  
    produce un'unita' di prodotto;  
    mutex.V();  
    fullBuffers.V();  
}
```

```
Consumer {} {  
    fullBuffers.P();  
    mutex.P();  
    consuma un'unita' di prodotto;  
    mutex.V();  
    emptyBuffers.V();  
}
```

Traccia di soluzione.

Corretta, è la soluzione proposta nelle dispense.

In conclusione: per la correttezza, non si possono scambiare le operazioni P, ma si possono scambiare le operazioni V (per queste ultime al più con impatto sull'efficienza).

2. Si consideri un'organizzazione della memoria che combina la segmentazione e l'impaginazione con indirizzi logici di 32 cifre binarie divise come:

- (a) identificativo del segmento: 4 cifre binarie;
- (b) numero di pagina: 12 cifre binarie;
- (c) indirizzo nella pagina ("offset"): 16 cifre binarie.

Siano date la tavola dei segmenti e la mappa della memoria seguenti.

Tavola dei segmenti i cui elementi hanno i seguenti campi: numero di segmento, indirizzo alla tavola delle pagine, dimensione della tavola delle pagine:

#Segmento	Indirizzo tav.pag.	Dimensione tav.pag.
0	0x20000	0x4
1	0x30000	0x4
2	Invalido	---
3	0x10000	0x4
4	Invalido	---
...	Invalido	---
15	Invalido	---

Contenuto della memoria fisica:

Indirizzo	Valore
0x10000	0x10
	0xA
	0xB
	0x5
...	...
0x20000	0x3
	0x7
	0x5
	0x9
...	...
0x30000	0x6
	0x4
	0x8
	0x11

... | ...

Si spieghi il meccanismo di traduzione degli indirizzi logici in indirizzi fisici. Dati la tavola dei segmenti e il contenuto della memoria fisica mostrati prima, si determinino gli indirizzi fisici ottenuti dai seguenti indirizzi logici, argomentando la risposta:

- (a) 0x00001234
- (b) 0x10001234
- (c) 0x11111234
- (d) 0x20021234
- (e) 0x30031234

Traccia di soluzione.

Si divide l'indirizzo logico in tre campi da sinistra a destra con rispettivamente una, due e tre cifre esadecimali. Il primo campo seleziona uno dei 16 segmenti nella tavola dei segmenti, da cui si ricava l'indirizzo in memoria della tavola delle pagine di tale segmento. Il secondo campo è il numero di pagina logica che seleziona a partire dall'indirizzo precedente il numero di pagina fisica (nel nostro caso ogni tavola delle pagine contiene quattro elementi). Il terzo campo contiene l'indice nella pagina ed è la parte finale dell'indirizzo sia logico che fisico.

Ad es. l'indirizzo 0x00001234 si ripartisce come 0x 0 000 1234. Il primo campo 0 seleziona nella tavola dei segmenti il segmento 0 la cui tavola delle pagine si trova in memoria a partire dall'indirizzo 0x20000. Il secondo campo 000 seleziona nella tavola delle pagine (che si trova in memoria a partire dall'indirizzo 0x20000) la prima parola (all'indirizzo di memoria 0x 20000 + 0x 000 = 0x 20003) che contiene il numero della pagina fisica 0x 3 corrispondente alla pagina logica 0x 0. Il terzo campo 1234 è concatenato con il precedente per ottenere l'indirizzo fisico 0x 31234.

Ad es. l'indirizzo 0x30031234 si ripartisce come 0x 3 003 1234. Il primo campo 3 seleziona nella tavola dei segmenti il segmento 3 la cui tavola delle pagine si trova in memoria a partire dall'indirizzo 0x10000. Il secondo campo 003 seleziona nella tavola delle pagine (che si trova in memoria a partire dall'indirizzo 0x10000) la quarta parola (all'indirizzo di memoria 0x 10000 +

$0x\ 003 = 0x\ 10003$) che contiene il numero della pagina fisica $0x\ 5$ corrispondente alla pagina logica $0x\ 3$. Il terzo campo 1234 e' concatenato con il precedente per ottenere l'indirizzo fisico $0x\ 51234$.

Si noti che l'indice della pagina logica non deve superare la dimensione della pagina (succeded nel caso (c)). Inoltre il segmento deve corrispondere a un indirizzo della tavola delle pagine valido (il contrario succede nel caso (d)).

(a) $0x00001234 \longrightarrow 0x31234$

(b) $0x10001234 \longrightarrow 0x61234$

(c) $0x11111234$ il numero di pagina virtuale eccede la dimensione del segmento

(d) $0x20021234$ il segmento 2 e' invalido

(e) $0x30031234 \longrightarrow 0x51234$

3. Si consideri il seguente codice LC-3

```
    AND R0, R0, #0
    LDR R1, R5, #0

    BRz CASE_1
    ADD R1, R1, #-1
    BRz CASE_2
    BR CASE_DEF

CASE_1:
    ADD R1, R0, #3
    STR R1, R5, #-1

CASE_2:
    ADD R1, R0, #4
    STR R1, R5, #-1
    BR END_SWITCH

CASE_DEF:
    ADD R1, R0, #5
    STR R1, R5, #-1
    BR END_SWITCH

END_SWITCH:
    .
    .
    .
```


Si spieghi il suo funzionamento, sia commentando le singole istruzioni che la procedura complessiva.

Traccia di soluzione

```
    AND R0, R0, #0    ; azzera r0
    LDR R1, R5, #0

    BRz CASE_1        ; confronta x==0
    ADD R1, R1, #-1
    BRz CASE_2        ; confronta x==1
    BR CASE_DEF       ; salta al caso CASE_DEF

CASE_1:
    ADD R1, R0, #3
    STR R1, R5, #-1   ; y = 3

CASE_2:
    ADD R1, R0, #4
    STR R1, R5, #-1   ; y = 4
    BR END_SWITCH     ; salta via

CASE_DEF:
    ADD R1, R0, #5
    STR R1, R5, #-1   ; y = 5
    BR END_SWITCH     ; salta via

END_SWITCH:
    .
    .
    .
```

Si stia attenti alla semantica di LDR e STR.

LDR legge il valore in memoria all'indirizzo contenuto in R5 e lo salva in R1: $R1 \leftarrow mem[R5]$.

STR scrive in memoria il contenuto di R1 all'indirizzo precedente di 1 quello contenuto in R5: $mem[R5 - 1] \leftarrow R1$.

Se chiamiamo rispettivamente x e y le variabili cui puntano $R5$ e $R5 - 1$, (cioe' x stia per $mem[R5]$ e y per $mem[R5 - 1]$), allora possiamo dire che il frammento di codice LC-3 precedente e' la compilazione del seguente frammento di codice C:

```
int x;
int y;

switch (x) {
case 0:
    y = 3;

case 1:
    y = 4;
    break;

default:
    y = 5;
    break;
}
```

nell'ipotesi appunto che $R5$ punti alla variabile x (cioe' ne contenga l'indirizzo) e che $R5 - 1$ punti alla variabile y .

4. Si progetti un circuito sequenziale che funziona come contatore di Johnson a 4 cifre binarie secondo la sequenza seguente: $0000 \rightarrow 1111 \rightarrow 0010 \rightarrow 1101 \rightarrow 0100 \rightarrow 1011 \rightarrow 0110 \rightarrow 1001 \rightarrow 1000 \rightarrow 0111 \rightarrow 0000$ e via ripetendo.

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

Si minimizzi il numero degli stati della macchina proposta.

Traccia di soluzione.

Si veda la soluzione di un problema simile nelle dispense.

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

Tavola delle transizioni con la codifica naturale della specifica.

0000	1111
0001	----
0010	1101
0011	----
0100	1011
0101	----
0110	1001
0111	0000
1000	0111
1001	1000
1010	----
1011	0110
1100	----
1101	0100
1110	----
1111	0010

- (c) Supponendo di usare bistabili di tipo D, si derivino le equazioni minimizzate di eccitazione degl'ingressi dei bistabili e le equazioni minimizzate delle uscite.

- (d) Si realizzi il circuito sequenziale corrispondente con bistabili di tipo D campionati sul fronte di salita, invertitori e porte NAND (a 2, 3, o 4 ingressi). Si etichettino con chiarezza i segnali.