

# Elementi di Architettura e Sistemi Operativi

Bioinformatica - Tiziano Villa

10 Febbraio 2012

Nome e Cognome:

Matricola:

Posta elettronica:

problema	punti massimi	i tuoi punti
problema 1	6	
problema 2	7	
problema 3	7	
problema 4	10	
totale	30	

1. (a) S'indichino i tempi d'accesso tipici in nanosecondi alle seguenti memorie: registri di un processore, cache tra processore e memoria principale, memoria principale, memoria secondaria (disco).

Traccia di soluzione.

Numeri tratti dal libro di testo aggiornato al maggio 2009. Si possono trovare altrove numeri un po' diversi e/o piu' aggiornati. Tutti i numeri ragionevoli sono stati accettati.

Registri: 0,25-0,5 ns

Cache: 0,5-25 ns

Memoria principale: 80-250 ns

Disco: 5 000 000 ns.

- (b) S'indichino le dimensioni tipiche (in bytes) delle seguenti memorie: registri di un processore, cache tra processore e memoria principale, memoria principale, memoria secondaria (disco).

Traccia di soluzione.

Registri: < 1 KB

Cache: < 16 MB

Memoria principale: < 16 GB

Disco: > 100 GB.

(c) Si scriva e spieghi la formula che da' il tempo d'accesso medio a una memoria cache.

Se una memoria principale ha tempo d'accesso sui 100 ns e una cache sui 10 ns, se la percentuale di successo e' del 90%, qual e' il tempo d'accesso medio ?

Se una memoria principale ha tempo d'accesso sui 100 ns e una cache sui 10 ns, se la percentuale di successo e' del 99%, qual e' il tempo d'accesso medio ?

Traccia di soluzione.

Tempo d'accesso medio = Percentuale di successo x Tempo per successo + Percentuale d'insuccesso x Tempo per insuccesso.

Percentuale di successo del 90%:

$$90\% \times 10 + 10\% \times 110 = 20 \text{ ns.}$$

Percentuale di successo del 99%:

$$99\% \times 10 + 1\% \times 110 = 11 \text{ ns.}$$

2. Si consideri il seguente codice per risolvere il problema dei lettori e scrittori mediante semafori (*OKToRead*, *OKToWrite*, *Mutex*) e variabili di stato (*AR* numero lettori attivi, *WR* numero lettori in attesa, *AW* numero scrittori attivi, *WW* numero scrittori in attesa).

```
Semaphore OKToRead = 0; OKToWrite = 0; Mutex = 1;  
AR = WR = AW = WW = 0;
```

```
Lettore {  
    P (Mutex) ;  
    if ((AW + WW) == 0) {  
        V (OKToRead) ;  
        AR = AR + 1 ;  
    } else WR = WR + 1 ;  
    V (Mutex) ;  
    P (OKToRead) ;  
  
    leggere i dati ;  
  
    P (Mutex) ;  
    AR = AR - 1 ;  
    if (AR == 0 && WW > 0) {  
        V (OKToWrite) ;  
        AW = AW + 1 ;  
        WW = WW - 1 ;  
    }  
    V (Mutex) ;  
}
```

```

Scrittore {
    P(Mutex);
    if ((AW + AR + WW) == 0) {
        V(OKToWrite);
        AW = AW + 1;
    } else WW = WW + 1;
    V(Mutex);
    P(OKToWrite);

    scrivere i dati;

    P(Mutex);
    AW = AW - 1;
    if (WW > 0) {
        V(OKToWrite);
        AW = AW + 1;
        WW = WW - 1;
    } else while (WR > 0) {
        V(OKToRead);
        AR = AR + 1;
        WR = WR - 1;
    }
    V(Mutex);
}

```

(a) Si analizzi il codice precedente spiegandone il funzionamento.

Traccia di soluzione.

- Più lettori possono accedere al codice contemporaneamente, ma gli scrittori devono avere accesso esclusivo.
- I lettori possono procedere solo se non ci sono scrittori attivi o in attesa.
- Gli scrittori possono procedere solo se non ci sono lettori attivi o scrittori attivi o in attesa.
- Solo un processo alla volta manipola le variabili di stato con il semaforo *Mutex*.

(b) In caso di conflitto tra lettori e scrittori chi ha la priorit  ? Perche' ?

Traccia di soluzione.

Gli scrittori hanno la priorit . Ai lettori puo' essere negata la risorsa indefinitamente.

(c) E' necessario  $WW$  nella condizione del primo *if* dello *Scrittore* ? Perche' ?

Traccia di soluzione.

No, se  $WW \neq 0$  si deve avere  $AW \neq 0$  oppure  $AR \neq 0$ , poiche' se lo scrittore e' stato messo in attesa ci doveva essere o uno scrittore attivo o un lettore attivo.

(d) Puo' essere  $OKToRead > 1$  ? Puo' essere  $OKToWrite > 1$  ? Perche' ?

Traccia di soluzione.

Si alla prima, No alla seconda.



- (e) Si puo' garantire che il primo scrittore che esegue  $P(Mutex)$  sia anche il primo scrittore a scrivere ? Si risponda alla domanda sia nel caso che ci siano lettori attivi che nel caso in cui non ci siano lettori attivi. Si argomentino le risposte in dettaglio.

Traccia di soluzione.

La risposta e' NO.

Se ci sono lettori attivi, tutti gli scrittori che arrivano sono messi in attesa in una lista e chi sara' il primo dipendera' dal meccanismo di selezione dei processi dalla lista degli scrittori in attesa (quando i lettori attivi avranno finito).

La risposta e' NO anche nel caso che non ci siano lettori attivi e la spiegazione e' piu sottile, come segue. Il primo scrittore puo' essere fermato dopo  $V(Mutex)$ , e un secondo scrittore puo' arrivare ad eseguire  $P(OKToWrite)$  e proseguire senza problemi poiche' la variabile  $OKToWrite$  e' stata abilitata con l'operazione  $V(OKToWrite)$  dal primo scrittore, come mostrato in dettaglio di seguito.

Il primo scrittore esegue il seguente codice e poi supponiamo che sia sospeso esattamente dopo  $V(Mutex)$ :

```
Scrittore {
    P(Mutex);
    if ((AW + AR + WW) == 0) {
        V(OKToWrite);
        AW = AW + 1;
    } else WW = WW + 1;
    V(Mutex);
}
```

A questo punto si ha  $AW = 1$ ,  $WW = 0$ .

Il secondo scrittore esegue il suo codice

```
Scrittore {
    P(Mutex);
    if ((AW + AR + WW) == 0) {
        V(OKToWrite);
        AW = AW + 1;
    } else WW = WW + 1;
    V(Mutex);
    P(OKToWrite);
}
```

vede che  $OKToWrite = 1$  (grazie al primo scrittore), e prosegue accedendo in scrittura

```
P(OKToWrite);  
  
scrivere i dati;
```

poi continua, decrementando  $AW$  (incrementato dal primo scrittore), verificando se  $WW > 0$  - il che è vero perché lo stesso secondo scrittore ha incrementato la variabile  $WW$  - e quindi incrementando  $OKToWrite$  con  $V(OKToWrite)$  e  $AW$  (e decrementando  $WW$ ).

```
P(Mutex);  
AW = AW - 1;  
if (WW > 0) {  
    V(OKToWrite);  
    AW = AW + 1;  
    WW = WW - 1;  
} else while (WR > 0) {  
    V(OKToRead);  
    AR = AR + 1;  
    WR = WR - 1;  
}  
V(Mutex);  
}
```

Adesso con  $OKToWrite = 1$  il primo scrittore (quando riattivato) può proseguire come scrittore attivo con il resto del suo codice.

```
P(OKToWrite);  
  
scrivere i dati;  
  
P(Mutex);  
AW = AW - 1;  
if (WW > 0) {  
    V(OKToWrite);  
    AW = AW + 1;  
    WW = WW - 1;
```

```

    } else while (WR > 0) {
        V(OKToRead);
        AR = AR + 1;
        WR = WR - 1;
    }
    V(Mutex);
}

```

Si noti che il primo scrittore ha incrementato la variabile  $AW$  a cui poi "si riferisce" il secondo scrittore quando scrive; a sua volta il secondo scrittore ha incrementato la variabile  $WW$  a cui poi "si riferisce" il primo scrittore quando e' in attesa di scrivere (poi il secondo scrittore quando ha finito incrementa  $AW$  e decrementa  $WW$ , il che permette al primo scrittore di scrivere).

3. Si consideri un sistema a memoria virtuale con sostituzione delle pagine mediante l'algoritmo della seconda opportunita' ("second chance").

(a) Si descriva l'algoritmo della seconda opportunita' realizzato con una lista lineare con puntatori alla prima e ultima pagina.

Traccia: L'algoritmo della seconda opportunita' e' FIFO + cifra d'uso ("use bit" o "reference bit").

Traccia di soluzione.

- Ogni pagina fisica ha una cifra d'uso.
- Quando si deve rimpiazzare una pagina, si controlla la cifra d'uso della pagina in testa alla coda (la pagina caricata meno di recente):
  - Se la cifra d'uso e' 0, si elimina tale pagina e si inserisce la nuova pagina alla fine della coda.
  - Se la cifra d'uso e' 1, si azzerla la cifra d'uso e si sposta tale pagina alla fine della coda (cioe' si da' a tale pagina una seconda possibilita'), poi si passa alla pagina successiva in testa alla coda, e cosi' via.

(b) Supponendo di avere a disposizione quattro pagine fisiche ("frame"), e date le seguenti successioni di arrivi di pagina e di riferimenti a pagina, si mostri ad ogni passo la lista corrente creata dall'algoritmo suddetto, disegnando la lista con le pagine fisiche (lista lineare doppiamente concatenata) e puntatori alla prima e all'ultima pagina caricata, e indicando per ogni pagina fisica il valore della cifra d'uso ("use bit" o "reference bit").

- arriva la pagina B
- arriva la pagina A
- riferimento alla pagina A
- arriva la pagina D
- arriva la pagina C
- arriva la pagina F
- riferimento alla pagina D
- arriva la pagina E

Traccia di soluzione.

Nelle figure sotto, per "prima pagina" s'intende prima pagina caricata, e per "ultima pagina" s'intende ultima pagina caricata.

- arriva la pagina B

```

prima   /   ultima
pagina /   pagina
-----
|B u:0|
-----
```

- arriva la pagina A

```

prima           ultima
pagina          pagina
-----         -----
|B u:0| <--> |A u:0|
-----         -----
```

- riferimento alla pagina A

prima	ultima
pagina	pagina
-----	-----
B u:0	<-->  A u:1
-----	-----

- arriva la pagina D

prima		ultima
pagina		pagina
-----	-----	-----
B u:0	<-->  A u:1	<-->  D u:0
-----	-----	-----

- arriva la pagina C

prima			ultima
pagina			pagina
-----	-----	-----	-----
B u:0	<-->  A u:1	<-->  D u:0	<-->  C u:0
-----	-----	-----	-----

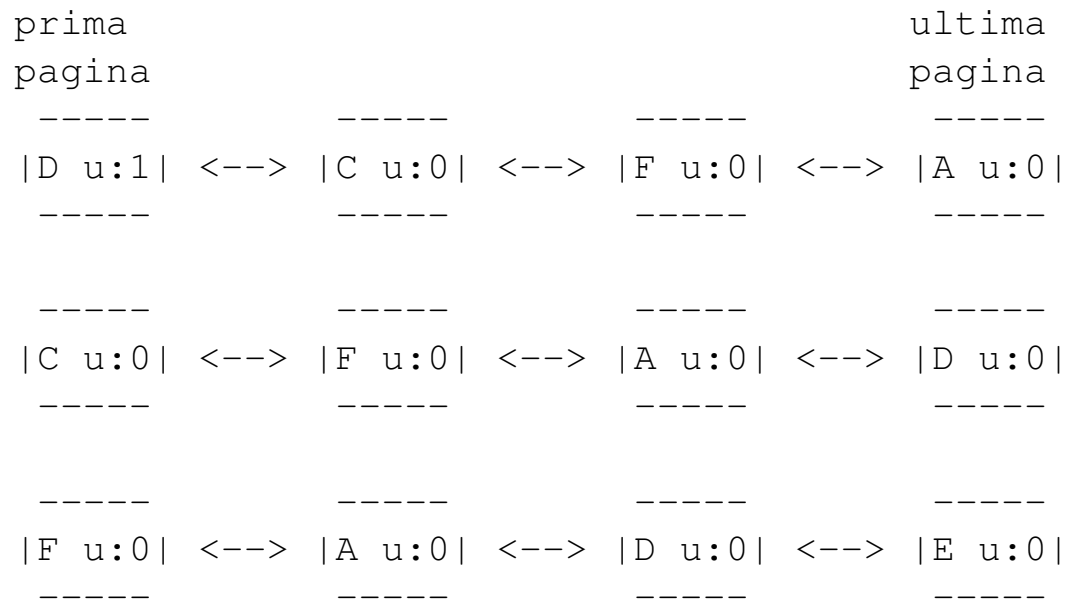
- arriva la pagina F

prima			ultima
pagina			pagina
-----	-----	-----	-----
A u:1	<-->  D u:0	<-->  C u:0	<-->  F u:0
-----	-----	-----	-----

- riferimento alla pagina D

prima			ultima
pagina			pagina
-----	-----	-----	-----
A u:1	<-->  D u:1	<-->  C u:0	<-->  F u:0
-----	-----	-----	-----

- arriva la pagina E



4. Si progetti un circuito sequenziale che realizza la seguente specifica:

- C'è un segnale binario d'ingresso  $X$  e un segnale binario d'uscita  $Z$ .
- L'uscita  $z$  vale 1 per un ciclo di orologio se e solo se all'ingresso è riconosciuta una sequenza composta di almeno due zeri seguiti da due uni (l'uscita  $z$  va a 1 in corrispondenza del secondo uno), altrimenti  $z$  vale 0.

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

Traccia di soluzione.

I SP SF U

0 s0 s1 0 # s0 stato iniziale

1 s0 s0 0

0 s1 s2 0

1 s1 s0 0

0 s2 s2 0

1 s2 s3 0

0 s3 s1 0

1 s3 s0 1

Si noti che la specifica originale

”L'uscita  $z$  vale 1 se e solo se sull'ingresso si sono presentati almeno due zeri seguiti esattamente da due uni (l'uscita  $z$  va a 1 in corrispondenza del secondo uno), altrimenti  $z$  vale 0.”

permetteva più interpretazioni. Si sono accettate tutte le interpretazioni ragionevoli.

Ad esempio, ci si poteva chiedere: l'uscita  $z$  va a 1 in corrispondenza del secondo uno della sequenza d'ingresso  $000^*110\dots$  e poi rimane a 1, oppure dopo ritorna a 0 e aspetta una nuova occorrenza di tale sequenza per diventare 1 in corrispondenza del secondo uno etc. ?



Una macchina che soddisfa la prima interpretazione e' la seguente

I SP SF U

0 s0 s1 0 # s0 stato iniziale

1 s0 s0 0

0 s1 s2 0

1 s1 s0 0

0 s2 s2 0

1 s2 s3 0

0 s3 s1 0

1 s3 s4 1

0 s4 s5 1

1 s4 s0 0

0 s5 s5 1

1 s5 s5 1

Nota che "esattamente", se interpretato alla lettera insieme con la richiesta di produrre 1 in corrispondenza del secondo uno, richiederebbe di "predire il futuro" per congetturare che la prossima cifra sara' 0 invece di 1. Per poter riconoscere che si sono visti esattamente due uni, bisognerebbe cioe' leggere in ingresso uno 0 subito dopo i due uni (per soddisfare la specifica che si sono visti esattamente due uni), ma in tal caso si potrebbe produrre  $z = 1$  solo in corrispondenza di tale 0 letto dopo i due uni.

- (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 degl'ingressi dei bistabili e le equazioni minimizzate delle uscite.

- (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.