

# Elementi di Architettura e Sistemi Operativi

Bioinformatica - Tiziano Villa

20 Ottobre 2017

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 seguente paradigma di sincronizzazione, nel caso di due processi che competono per entrare in una sezione critica. Si assuma che:
  - (a) Una sezione critica e' protetta se vi puo' accedere un solo processo per volta;
  - (b) la sincronizzazione e' equa se ogni processo ha la medesima possibilita' di accedere alla sezione critica.
  - (c) All'inizio vale

`flag = false;`

Nel libro di testo s'introduce la seguente istruzione *TestAndSet* disponibile in alcune architetture:

```
boolean TestAndSet(boolean *obiettivo) {  
    boolean valore = *obiettivo;  
    *obiettivo = true;  
    return valore;  
}
```

Tale istruzione si puo' utilizzare per imporre la mutua esclusione.

Si supponga che ogni processo esegua il seguente codice:

```
1. while (TestAndSet(&flag) == false)  
2.     ;  
3. si esegue nella sezione critica;  
4. flag = false;
```

Si risponda alle seguenti domande.

- (a) Come funziona questo meccanismo di sincronizzazione ?

Traccia di soluzione.

*TestAndSet* e' un'istruzione che atomicamente scrive 1 in una locazione di memoria (*flag*) ma restituisce il valore precedente di tale locazione. In questo modo si puo' proteggere una sezione critica di memoria utilizzando *flag* come un suo lucchetto e *TestAndSet* come chiave che apre il lucchetto: supponendo che la locazione *flag* argomento della *TestAndSet* sia inizializzata a falso, il primo processo che chiama la procedura *TestAndSet* acquisisce il lucchetto *flag* che e' falso, lo mette a vero, ma restituisce il vecchio valore falso di *flag*, mentre tutti gli altri processi che vorrebbero acquisire il lucchetto *flag* sono bloccati dal fatto che *flag* rimane vero sino a quando il primo processo non avra' terminato di eseguire nella sezione critica e al termine non avra' rimesso *flag* a falso, per dar modo a un altro processo in attesa (occupata) di acquisire il lucchetto *flag*.

- (b) Questo meccanismo di sincronizzazione garantisce la protezione della sezione critica ?

Se si, si argomenti il motivo. Se no, si mostri come correggere il codice.

Traccia di soluzione.

No. La condizione di sezione critica occupata e' indicata da *flag == true*, poiche' la procedura *TestAndSet* pone l'argomento a vero quando e' chiamata. Percio' un processo non aspetta quando la sezione critica e' gia' occupata da un altro processo. Per correggere il codice si sostituisca *TestAndSet(flag) == false* con *TestAndSet(flag) == true*.

```
1. while (TestAndSet(&flag) == true)
2.     ;
3. si esegue nella sezione critica;
4. flag = false;
```

- (c) Questo meccanismo di sincronizzazione e' equo ?

Traccia di soluzione.

Si. Il codice e' simmetrico, percio' ogni processo ha la medesima possibilita' di accesso.

2. Si consideri la seguente tavola dei segmenti:

| Segmento | Base | Lunghezza |
|----------|------|-----------|
| 0        | 219  | 600       |
| 1        | 2300 | 14        |
| 2        | 90   | 100       |
| 3        | 1327 | 580       |
| 4        | 1952 | 96        |

Si traducano i seguenti indirizzi logici nei corrispondenti indirizzi fisici.

(a)  $\langle 0, 430 \rangle$

(b)  $\langle 1, 10 \rangle$

(c)  $\langle 2, 500 \rangle$

(d)  $\langle 3, 400 \rangle$

(e)  $\langle 4, 112 \rangle$

Traccia di soluzione.

(a)  $219 + 430 = 649$

(b)  $2300 + 10 = 2310$

(c) indirizzamento illegale fuori segmento

(d)  $1327 + 400 = 1727$

(e) indirizzamento illegale fuori segmento

3. Si consideri il seguente frammento di programma scritto nel linguaggio macchina dell'architettura LC-3.

```
0x3000    AND R2, R2, #0
0x3001    ADD R1, R1, #-1
0x3002    ADD R1, R1, #-1
0x3003    ADD R1, R1, #-1
0x3004    BRn x3007
0x3005    ADD R2, R2, #1
0x3006    BRnzp x3001
0x3007    TRAP 0x25
```

Si spieghi il funzionamento di tale codice.

Quali sono i valori iniziali possibili del registro  $R1$  tali che il valore finale del registro  $R2$  sia 3 ?

Traccia di soluzione.

I valori possibili di  $R1$  sono: 9, 10, 11.

Il contenuto  $x$  di  $R1$  deve soddisfare le disequazioni:  $0 \leq x - 9 < 3$ .

La disequazione  $0 \leq x - 9$  stabilisce che si deve sottrarre 3 per almeno tre volte da  $R1$  per incrementare  $R2$  per almeno tre volte, cioè il contenuto di  $R2$  è almeno 3.

La disequazione  $x - 9 < 3$  stabilisce che non ci può essere un quarto incremento di  $R2$  perché dopo aver sottratto 3 per tre volte si ottiene un numero  $< 3$  e quindi si salta il quarto incremento di  $R2$ , cioè il contenuto di  $R2$  è al più 3.

4. Si consideri lo schematico allegato di un circuito sequenziale. Si costruisca la macchina a stati finiti corrispondente a tale circuito sequenziale.

Che cosa fa tale circuito ? Come si possono interpretare i segnali d'ingresso ?

Traccia di soluzione.

Le equazioni delle funzioni di stato futuro  $N_1$  e  $N_0$  sono:

$$N_1 = C'S_1 + C(D \oplus (S_1 \oplus S_0)) = C'S_1 + CDS'_0S'_1 + CDS_0S_1 + CD'S_0S'_1 + CD'S'_0S_1,$$

$$N_0 = C \oplus S_0 = CS'_0 + C'S_0.$$

La tavola della macchina a stati finiti estratta e':

| C | D | S1 | S0 | N1 | N0 |
|---|---|----|----|----|----|
| 0 | 0 | 0  | 0  | 0  | 0  |
| 0 | 1 | 0  | 0  | 0  | 0  |
| 1 | 0 | 0  | 0  | 0  | 1  |
| 1 | 1 | 0  | 0  | 1  | 1  |

|   |   |   |   |   |   |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 |

|   |   |   |   |   |   |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 |

|   |   |   |   |   |   |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 |

Si veda l'allegato per il grafo della macchina a stati finiti.

Il circuito realizza un contatore a due cifre con due ingressi  $C$  e  $D$ :  $C = 0$  indica che non si conta,  $C = 1$  indica che si conta;  $D = 0$  indica che si conta in avanti,  $D = 1$  indica che si conta all'indietro.