Architettura degli Elaboratori, 2008-09

Appello del 16 febbraio 2009

Domanda 1

Una unità di elaborazione U contiene un componente logico memoria M di capacità N = 16K interi non negativi di 32 bit. Riceve da U1 messaggi (A, IND), con A intero non negativo di 32 bit e IND di 14 bit, ed invia ad U2 messaggi (OUT) di 15 bit.

Una *prima versione* di U è espressa dal seguente microprogramma, dove *zero (M[IND] – A)* funge da codice operativo di due operazioni esterne equiprobabili, e la funzione booleana *f(x),* con *x* intero non negativo, è vera se e solo se *x* è una potenza di due ≥ 256 Mega:

0. (RDY1, zero(M[IND] – A) = 0 –) nop, 0;

   (= 1 1) reset RDY1, set ACK1, 0 → I, 0 → C, 1;

   (= 1 0) reset RDY1, set ACK1, A → M[IND], 0

1. (I_0, ACK2 = 0 –) f(M[I_m]) → H, 2;

   (= 1 0) nop, 1;

   (= 1 1) C → OUT, set RDY2, reset ACK2, 0;

2. (H = 0) I + 1 → I, 1;

   (= 1) I + 1 → I, C + 1 → C, 1

Si chiede una *seconda versione* di U tale che, esprimendo il tempo medio di elaborazione della prima versione come $T_1 \sim p\ N$, la seconda versione abbia un tempo medio di elaborazione $T_2 \sim q\ N$, con $q < p$.

Detto t_p il ritardo di una porta logica con al massimo 8 ingressi, le ALU e la memoria M hanno ritardo di $5t_p$.

Scrivere il microprogramma della seconda versione, realizzare *f(x)* e valutare le costanti *p* e *q*. Spiegare chiaramente come si è ragionato per definire la seconda versione.

Domanda 2

Si consideri il seguente programma che esegue il prodotto matrice-vettore:

```
int A[N][N], int B[N], int C[N];
    { for (i = 0; i < N; i ++)
        { C[i]] = 0;
          for (j = 0; j < N; j ++)
              C[i] = C[i] + A[i][j] * B[j]
        }
    }
```

Il programma è eseguito su un elaboratore la cui CPU contiene una cache primaria operante su domanda, completamente associativa, di capacità 64K parole e blocchi di 16 parole.

Spiegare come è costituito l'insieme di lavoro del programma, determinare sotto quali condizioni è possibile che questo risieda interamente in cache e, in tale ipotesi, determinare il numero medio di fault di cache.

Domanda 3

Si consideri il supporto a tempo di esecuzione di un comando *send* di LC, facente uso del metodo con indirizzi logici distinti per risolvere il problema delle strutture condivise riferite indirettamente.

Spiegare chiaramente la sequenza di azioni eseguite nel caso che il processo destinatario si trovi in attesa. Con il processo mittente di tipo interno, distinguere il caso che il processo destinatario sia interno o esterno.

Sintesi di soluzione

Quanto qui riportato è una estrema sintesi della soluzione, che <u>non</u> va presa a modello circa il modo di presentare l'elaborato, ma che ha solo lo scopo di mostrare il risultato. Per il modo di presentare l'elaborato si vedano le Esercitazioni 1, 2, 3 e le soluzioni delle due Prove di Verifica Intermedia.

Domanda 1

La prima versione ha i seguenti ritardi:

- ω_{PO}: $10t_p$ (domina la variabile di condizionamento *zero(…)*; M è indirizzata separatamente da IND e I_m),
- ω_{PC}: $2t_p$, σ_{PC}: $2t_p$,
- σ_{PO}: $8t_p$ (domina l'operazione elementare che usa $f(M[I_m])$, mentre gli incrementi hanno ritardo $7t_p$).

  L'espressione logica AND-OR della funzione *f(x)* contiene quattro termini AND a 32 variabili, ognuno contenente una variabile affermata in una delle quattro posizioni più significative e tutte le altre variabili negate. Il ritardo vale $3t_p$.

Quindi $\tau_1 = 21\ t_p$, $T_1 \sim 2N\ \tau_1/2 = 21\ N\ t_p$, $p = 21\ t_p$.

Poiché la prima versione usa la variabile di condizionamento complessa *zero(…)* che condiziona completamente ω_{PO}, l'uso della variabile di condizionamento semplice H non contribuisce ad abbassare il ciclo di clock ed anzi raddoppia il numero di cicli di clock.

La seconda versione, quindi, elimina H ed usa direttamente la variabile di condizionamento complessa $f(M[I_m])$, accorpando in una le due microistruzioni 1, 2, e *dimezzando il numero di cicli di clock*:

- ω_{PO}: $10t_p$ (domina ancora la variabile di condizionamento *zero(…)*),
- ω_{PC}: $2t_p$, σ_{PC}: $2t_p$,
- σ_{PO}: $7t_p$ (gli incrementi).

Quindi $\tau_2 = 20\ t_p$, $T_2 \sim N\ \tau_2/2 = 10\ N\ t_p$, $q = 10\ t_p$.

Domanda 2

L'insieme di lavoro è costituito da *w* blocchi che sono i blocchi di codice (1 – 2), un blocco di A ed un blocco di C (più eventualmente i blocchi del supporto più frequentemente riferiti), e da tutti gli N/σ blocchi di B sul quale si ha riuso ad ogni iterazione del loop più interno. La condizione affinché l'insieme di lavoro risieda in cache è

$$N + \sigma w < 64K$$

Inoltre, le istruzioni di LOAD su B sono annotate con "non deallocare".

Il numero di fault è dato da N^2/σ per A + N/σ (invece di N^2/σ se non si avesse B interamente in cache) per B + N/σ per C, quindi in totale ~ N^2/σ (invece di 2 N^2/σ).

Domanda 3

L'indicazione se il processo destinatario è di tipo interno o esterno è contenuta nel descrittore di canale oppure figura tra i parametri della procedura *send*.

Il descrittore di canale CH contiene, tra l'altro, un booleano Wait, la lunghezza L del messaggio, l'identificatore unico dell'oggetto variabile targa, *id_vtg*, e l'identificatore unico dell'oggetto PCBdest, *id_pcb*; *id_vtg* e *id_pcb* sono scritti in CH se e quando il destinatario si sospende).

Il ramo del supporto della *send* che trova Wait = true, effettua le seguenti azioni:

1. rimette Wait = false,
2. determina l'indirizzo logico nello spazio del destinatario dell'oggetto variabile targa usando *id_vtg* come indice o chiave di una tabella privata,
3. con un loop ripetuto L volte, copia il messaggio nella variabile targa,
4. se il destinatario è interno, determina l'indirizzo logico nello spazio del destinatario dell'oggetto PCBdest usando *id_pcb* come indice o chiave di una tabella privata, e concatena il PCB nella Lista Pronti, i cui puntatori sono identificatori unici di oggetti PCB; altrimenti, se il destinatario è esterno, invia, mediante una istruzione di STORE mappata nello spazio di I/O, una comunicazione di sveglia all'unità di I/O,
5. ritorna da procedura riabilitando le interruzioni.