Architetture degli elaboratori – Anno Accademico 2009-2010

Appello straordinario (vecchio ordinamento) – 2 Febbraio 2010

Specificare su tutti i fogli consegnati, nome, cognome, numero di matricola e corso (A o B) di appartenenza. Motivare adeguatamente risposte e soluzioni. I risultati, appena disponibili, saranno pubblicati sulle pagine web dei docenti.

Domanda 1

Un'unità di elaborazione U riceve da U_1 una coppia di indirizzi, INDA e INDB, ed un intero N. I due indirizzi rappresentano l'indirizzo base in memoria di due vettori, A e B, lunghi N parole.

L'unità U calcola un valore X definito come la somma di tutti gli A[i] e B[i] per cui A[i] ≥ B[i] e restituisce tale risultato alla unità U_1.

I vettori A e B risiedono in memoria. La gerarchia di memoria del sistema è costituita da una memoria cache, ad accesso diretto, operante su domanda, costituita da 16K blocchi di ampiezza 8 parole, e da una memoria principale da 1G parole, interallacciata con 4 moduli, e con tempo di accesso a ciascuno dei moduli pari a 20τ.

La cache è realizzata sullo stesso chip di U. I collegamenti inter-chip hanno una latenza di trasmissione pari a 4τ.

Con adeguate spiegazioni, si fornisca il microprogramma dell'unità U e si valuti il suo tempo medio di elaborazione in funzione di N e t_p, dove t_p è il ritardo di stabilizzazione di una porta logica con al più 8 ingressi. Il ritardo di una ALU vale 5 t_p.

Domanda 2

Dire se le seguenti affermazioni sono vere, false o vere sotto determinate condizioni, spiegando la risposta:

a) una unità di I/O, che sia implementata a livello firmware, non può utilizzare indirizzi logici;

b) per virtualizzare come processo esterno una unità di I/O, che sia implementata a livello firmware, è necessario che l'esecuzione di primitive per la cooperazione con processi interni sia delegata alla CPU;

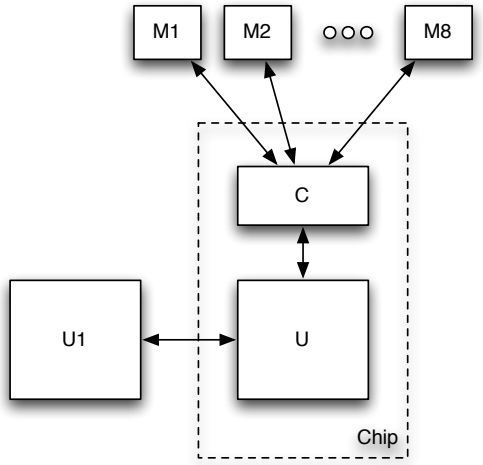
c) per permettere ad una unità di I/O di delegare alla CPU l'esecuzione di una funzionalità del supporto a tempo di esecuzione dei meccanismi di cooperazione tra processi, è necessario risolvere un problema di strutture dati condivise riferite indirettamente. Nel dare la risposta riferirsi esplicitamente ad almeno una specifica funzionalità;

d) si consideri il caso di una certa unità di I/O che utilizzi esclusivamente Memory Mapped I/O e che sia virtualizzata come processo esterno LC. Con riferimento all'implementazione di una primitiva *send* eseguita da tale unità: *1)* è impossibile effettuare la copia del messaggio direttamente nella variabile targa del processo interno destinatario; *2)* è possibile effettuare la copia del messaggio direttamente nella variabile targa del processo interno destinatario, ma ciò comporta un aumento della latenza della *send* stessa.

Traccia di soluzione (alcune spiegazioni sono da espandere leggermente)

Domanda 1

Il sistema ha la struttura illustrata in Figura.



Il calcolo di X avviene come segue:

```
int A[N], B[N], X;
{
// ricevi indirizzi base di A e B da U1
for(int i=0; i<N; i++) {
  if(A[i] >= B[i])
    X = X + A[i] + B[i];
}
// spedisci X a U1
}
```

Assumiamo che U1 attenda l'esito della calcolo richiesto prima di inviare eventuali altre richieste, quindi lo schema di interazione sarà a domanda/risposta. L'interfaccia fra U1 ed U è composta dai segnali di sincronizzazione RDY1 e RYDOUT1 e dai registri A, B ed N (in ingresso) e X (in uscita). L'interfaccia fra U e la cache è un'interfaccia di memoria standard ($RDYIN_C$, $RDYOUT_C$, DATAIN, DATAOUT, IND, OP, ESITO).

Il predicato if (A[i] >= B[i]) viene implementato come variabile di condizionamento complessa: questa scelta minimizza il numero di cicli di clock, mentre soluzioni con variabili di condizionamento semplici sarebbero caratterizzate da un numero di cicli di clock quasi doppio senza ridurre proporzionalmente la lunghezza del ciclo di clock. Per gli stessi motivi, per il calcolo di X conviene usare due ALU in cascata. Il microprogramma potrebbe essere il seguente:

0. (RDY_1=0) nop, 0;
   (=1) , reset RDY1, A →INDA, B→INDB, N-1→I, 0→XTEMP, INDA→IND,"read"→OP,set $RDYOUT_C$,1
1. ($RDYIN_C$,or(ESITO),I_0=0--) nop, 1
   (=100) reset $RDYIN_C$, set $RDYOUT_C$, INDB→IND, "read"→OP, INDA+1→INDA, DATAIN→AI, 2
   (=110) … // *trattamento eccezioni*
   (=--1) XTEMP→X, set $RDYOUT_1$, 0
2. ($RDYIN_C$,or(ESITO),segno (AI-DATAIN)=0--) nop,2
   (=100) AI+DATAIN+XTEMP→XTEMP, INDB+1→INDB, I-1→I, INDA→IND, "read"→OP, set $RDYOUT_C$, reset $RDYIN_C$, 1
   (=101) INDB+1→INDB, I-1→I, INDA→IND, "read"→OP, set $RDYOUT_C$, reset $RDYIN_C$, 1

(=11-) … // *trattamento eccezioni*

Il tempo medio di elaborazione deve tener conto degli accessi in cache e dei fault di cache. Sia il vettore A che il vettore B vengono letti sequenzialmente e quindi genereranno ognuno N/σ fault, *a patto che l'indirizzo base di A e di B mappino su blocchi diversi della cache ad indirizzamento diretto ed indipendentemente dalla loro effettiva lunghezza N*.

In totale verranno eseguite

$$2N + 2$$

microistruzioni. Per semplicità approssimiamo il valore con 2N. All'interno di queste microistruzioni si genereranno 2N accessi in memoria. Tenendo conto che la cache è ad indirizzamento diretto, ogni accesso (in caso di successo) richiederà 2τ. Quindi il tempo medio di elaborazione, al netto dei fault di cache, sarà di

$$T_{id} = 2N\tau + 4N\tau = 6N\tau$$

Dobbiamo inoltre considerare il tempo necessario a trattare i fault in cache T_{fault}.

A e B sono in sola lettura, e quindi il tempo necessario al trattamento del singolo fault può essere stimato come il tempo necessario a trasferire un blocco di cache dalla memoria principale interallacciata. Con le solite convenzioni tale tempo può essere stimato in

$$T_{fault} = 2\ T_{tr} + (\sigma/m)\ \tau_M + m\ \tau = 8\ \tau + 40\ \tau + 4\ \tau = 52\ \tau$$

Quindi il tempo medio di elaborazione può essere stimato in

$$T = T_{id} + T_{fault} = 6N\ \tau + (2N/\sigma)\ 52\tau = 6N\ \tau + 13N\ \tau = 19N\ \tau$$

La lunghezza del ciclo di clock può essere valutata secondo il procedimento standard:

$T_{\omega PO}=5t_p$ per via della ALU utilizzata nella microistruzione 2 per testare A[i]≥B[i];

$T_{\sigma PO}=12t_p$ per via della somma AI+DATAIN+XTEMP→XTEMP che richiede 2 ALU in cascata ed un commutatore all'ingresso di X;

$T_{\omega PC}$, $T_{\sigma PC}$ valgono $2t_p$ (4 variabili di condizionamento e 2 bit per lo stato, quindi i termini AND hanno meno di 8 ingressi. I termini in OR avranno ancora meno di otto ingressi).

Avremo quindi

$$\tau = 5t_p + \max\ \{2t_p + 12t_p, 2t_p\} + t_p = 20t_p$$

Domanda 2

a) Falso. L'uso di indirizzi logici è ortogonale al fatto che una funzionalità sia usata a livello firmware o assembler.

b) Falso in generale, in quanto il supporto delle primitive può essere scritto interamente a microprogramma. Solo nel caso che l'unità sia già "congelata", e si voglia aggiungere la possibilità di eseguire una primitiva non già prevista, è necessario il meccanismo di delega.

c) Vero. Esempio 1: l'esecuzione di una send via delega alla CPU; le strutture condivise riferite indirettamente sono il descrittore di canale e il messaggio oppure la variabile targa. Esempio 2: la sveglia di un processo interno; la struttura condivisa riferita indirettamente è il PCB del processo interno da svegliare.

d) 1) Falso, se la variabile targa del processo interno destinatario è allocata fisicamente nella memoria di tale unità di I/O.

2) Come detto al punto *1)*, la fattibilità è assicurata dal fatto che la variabile targa sia fisicamente allocata nella memoria dell'unità di I/O. La latenza della *send* è più alta nelle architetture più comuni nelle quali la struttura di interconnessione tra CPU e I/O ha una banda limitata (bus); spesso il suo accesso da parte del processore centrale non è soggetto a caching oppure, se previsto il caching, il trasferimento dei blocchi ha una latenza troppo alta per essere efficiente. Però, nessuna penalità è introdotta, dal punto di vista delle prestazioni, se la struttura di interconnessione tra CPU e unità di I/O ha la stessa banda della struttura tra CPU e memoria principale o cache secondaria.