Architettura degli Elaboratori, 2008-09

Appello del 23 gennaio 2009

Domanda 1

Una unità di elaborazione U è connessa ad una gerarchia di memoria costituita da una memoria principale MEM di capacità 1G parole (32 bit) e da una memoria CACHE di capacità 64K parole. U e l'unità che contiene la memoria CACHE appartengono allo stesso chip.

MEM è interallacciata con 8 moduli ed ha ciclo di clock uguale a 5 volte quello di U. CACHE ha blocchi di 8 parole, opera su domanda ed è indirizzata con il metodo diretto. Tutti i collegamenti inter-chip hanno latenza di trasmissione uguale a 5 volte il ciclo di clock di U.

MEM contiene solo oggetti di tipo array, ognuno di 1M interi.

U comunica in ingresso con una unità U_{Master} e in uscita con le unità U_0, …, U_7.

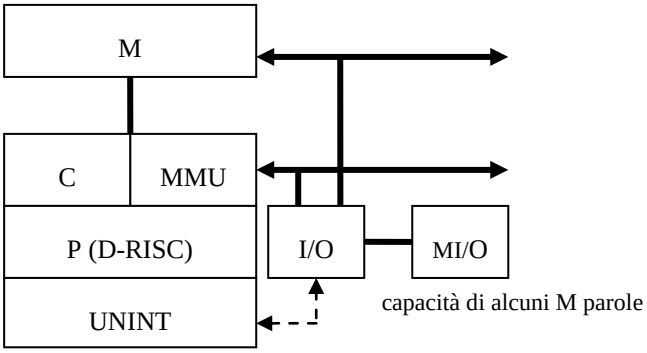
U riceve in ingresso messaggi (IDA, IDB, J) dove IDA e IDB sono gli identificatori unici di due array A e B, e J è un valore di 2 bit. U calcola il numero NUM di elementi di A e B, con lo stesso indice, che hanno il byte J-esimo uguale. Detto Q il valore di tale byte, NUM è inviato all'unità U_H con H = Q *div* 32.

É noto il ritardo t_p di una porta logica con al massimo 8 ingressi. Una ALU ha ritardo uguale a $5t_p$.

Valutare il tempo medio di elaborazione di U in funzione di t_p, spiegando adeguatamente la risposta.

Domanda 2

Un elaboratore general-purpose ha la seguente architettura:



I/O è una unità di I/O realizzata a livello firmware.

Un processo APPL opera su due array, A e B, ognuno di 4K interi. Il valore di A è ottenuto da I/O; il valore di B, risultato di una certa elaborazione su A, è inviato alla stessa I/O.

I/O è capace di trasferire blocchi di dati di ampiezza massima 4K parole. I/O *non* è virtualizzata da un processo driver.

Si distinguono i seguenti casi:

1. I/O è implementato come processo esterno. Si assume l'esistenza di un livello con linguaggio LC;
2. I/O *non* è implementato come processo esterno.

Nei due casi:

a) spiegare le azioni svolte a livello firmware da I/O ed a livello assembler dal processore, affinché abbia luogo la necessaria cooperazione tra APPL e I/O;

b) spiegare quali strutture dati condivise e condivise riferite indirettamente sono utilizzate, e su quali supporti di memoria è necessario o conveniente allocarle.

Sintesi di soluzione

Vengono date spiegazioni molto sintetiche riferite agli aspetti più significativi. Si vedano le Esercitazioni ed i riferimenti al materiale didattico per spiegazioni più esaurienti.

Domanda 1

La presenza della gerarchia di memoria di ripercuote solo sulle prestazioni di U, non sulla sua realizzazione. L'unità cache è vista all'interfaccia da U, per cui non è assolutamente necessario conoscerne l'implementazione interna.

Il tempo di accesso in cache, visto da U, vale $t_c = 1\tau$, come reso possibile dal metodo diretto.

Il tempo medio di elaborazione di U è valutato come:

$$T = T_{id} + N_{fault} \; T_{trasf}$$

L'*insieme di lavoro* della computazione consiste dei due blocchi correnti di A e B. Posto $N = 1M$, si ha:

$$N_{fault} = 2\,\frac{N}{\sigma} = \frac{N}{4}$$

$$T_{trasf} = 2\;T_{tr} + \tau_M + \sigma\,\tau = 23\,\tau$$

$$N_{fault} \; T_{trasf} = 5{,}85\,N\,\tau$$

T_{id}. si ricava dalla progettazione di U.

MEM contiene 1K vettori di 1M ciascuno, quindi gli identificatori IDA e IDB sono di 10 bit. Gli indirizzi base di A e B, di 30 bit, sono ottenuti concatenando 20 zeri agli identificatori. NUM ed il registro I contatore di passi sono di 21 bit. Si fa uso di controllo residuo per ricavare il byte J-esimo di una parola e per operare parametricamente sulle interfacce con U_0, …, U_7.

La *scelta della variabile di condizionamento*, che esprime l'uguaglianza del byte J-esimo di ogni coppia di elementi di A e B con lo stesso indice, cade sulla soluzione con variabile di condizionamento *complessa* (uscita della funzione zero di una ALU che esegue la differenza dei due byte). Questa soluzione minimizza il tempo medio di elaborazione: poiché gli operandi della differenza sono ottenuti con accessi in memoria, una soluzione con variabile di condizionamento in un registro non può essere implementata con l'anticipo di un ciclo di clock per formarne il valore. <u>*Nota: questa spiegazione è necessaria*</u>.

Con simboli ovvi, ed essendo H dato dai tre bit più significativi di Q (H = Q *div* 32), il microprogramma è:

0. (RDYMaster = 0) nop, 0;

   (= 1) reset RDYMaster, set ACKMaster, IDA° venti_zeri → IA, IDB° venti_zeri → IB,

   J → JJ, 0 → I, 0 → NUM, 1

1. (I_0, ACKOUT [H] = 0 –) IA → IND, set RDYOUT_C, 2;

   (= 1 0) nop, 1;

   (= 1 1) reset ACKOUT [H], set RDYOUT [H], NUM → OUT [H], 0

2. (RDYIN_C = 0) nop, 2;

   (= 1) reset RDYIN_C, DATAIN [JJ] → Q, IA + 1 → IA, IB → IND, set RDYOUT_C, 3

3. (RDYIN_C, *zero (Q – DATAIN [JJ])* = 0 –) nop, 3;

   (= 1 0) reset RDYIN_C, NUM + 1 → NUM, I + 1 → I, IB + 1 → IB, 1;

   (= 1 1) reset RDYIN_C, I + 1 → I, IB + 1 → IB, 1

Quindi:

$$T_{id} \sim N\,(3\tau + 2\,t_c) = 5\,N\,\tau$$

$$\varepsilon_{cache} = \frac{T_{id}}{T_{id} + T_{fault}} = \frac{5}{10{,}85} = 0{,}46$$

Per la valutazione del ciclo di clock:

$T_{\omega PO} = T_{Alu} + T_K = 7\,t_p$ — il commutatore è all'uscita di DATAIN e comandato da JJ; la ALU ha in ingresso solo Q e l'uscita del suddetto commutatore, ed esegue solo la sottrazione: è quindi verificata la condizione necessaria per la correttezza;

$T_{\omega PC} = T_{\sigma PC} = 2\,t_p$

$T_{\sigma PO} = T_{Alu} + 2\,T_K = 9\,t_p$ — oltre alla ALU sopra detta, la PO ne contiene altre tre (numero minimo per permettere il parallelismo nelle microperazioni), delle quali una ha in ingresso un commutatore a due ingressi (ad esempio, IA, IB).

$$\tau = 19\,t_p$$

$$T = 10{,}85\,N\,\tau = 2{,}16 \times 10^8\,t_p$$

Domanda 2

L'architettura rende possibile realizzare la condivisione di memoria tra CPU e I/O sia attraverso MI/O (*Memory Mapped I/O*) che attraverso M (*DMA*). La memoria comune viene utilizzata per tutte le strutture dati necessarie alla cooperazione:

a) strutture dati per il supporto a tempo di esecuzione delle send/receive di LC usate dai processi APPL e I/O,

b) strutture dati utilizzate per programmare esplicitamente la cooperazione tra il processo APPL e l'unità I/O.

a) La struttura del programma LC prevede

- un canale CH1 da APPL ad I/O per effettuare la richiesta, di tipo pura sincronizzazione: asincrono di 1 posizione;
- un canale CH2 da I/O a APPL per inviare il valore dell'array (4K), con variabile targa A: asincrono di 1 posizione;
- un canale CH3 da APPL ad I/O per inviare il valore di B (4K): asincrono di 1 posizione.

L'interazione attraverso i canali CH1, CH2 è a domanda e risposta, per cui APPL si porrà certamente in attesa nella receive. Quindi, con l'implementazione ottimizzata del supporto, il valore dell'array verrà sempre copiato da I/O direttamente nella variabile targa A.

Tutta la cooperazione, richiesta dal problema, è quindi ottenuta attraverso la programmazione in LC e dal supporto a tempo di esecuzione di LC, che è implementato a livello assembler per i processi interni ed a livello firmware, direttamente nel microprogramma, per l'unità di I/O. *Le azioni svolte sono quelle tipiche del supporto: si veda il materiale didattico*.

I descrittori di canale possono essere allocati *fisicamente* sia in MI/O che in M (*nessuna differenza si nota nel codice di APPL e nel microcodice di I/O*), quindi meglio se in M per facilitare il caching da parte della CPU. Lo stesso vale, a maggior ragione, per la variabile targa A. La variabile targa B conviene allocarla in MI/O, in modo da rendere agevole ad I/O l'utilizzo delle parole di tale struttura man mano che vengono scritte nella sua memoria e, contemporaneamente, inviate al dispositivo associato.

b) Non disponendo di LC, occorre programmare esplicitamente una sequenza di azioni equivalente a quella che, nella soluzione *a)*, è effettuata dal supporto a tempo di esecuzione delle primitive usate. Il realtà, rispetto al caso *a)*, sarà ora possibile apportare delle semplificazioni dovute alla conoscenza specifica del problema.

Metteremo in evidenza le strutture condivise (*sh*) e quelle condivise riferite indirettamente (*ind-sh*). Nel secondo caso il termine *riferimento* sta a significare indirizzo logico oppure identificatore unico di oggetto, a seconda che si faccia uso, rispettivamente, del metodo a indirizzi logici coincidenti oppure a indirizzi logici distinti, in questo secondo metodo l'indirizzo logico, nello spazio del processo che deve accedere all'oggetto, viene ricavato da un tabella privata indicizzata dall'identificatore unico: *si veda il materiale didattico*.

Azioni per implementare la cooperazione:

1. APPL esegue tre "STORE di I/O" (nello spazio logico mappato nella spazio fisico di MI/O: vedere il significato di questa frase) per passare ad I/O il riferimento ad A (ind-sh) e il riferimento a PCB_APPL (ind-sh) e per segnalare la richiesta(sh). La terza scrittura in MI/O è equivalente anche ad una segnalazione di sveglia I/O (I/O si trova in attesa attiva, oppure sta ricevendo dati dal dispositivo e li sta scrivendo in MI/O);
2. I/O, una volta disponibili le 4K parole dal dispositivo, le copia in A usandone il riferimento, quindi invia una interruzione con messaggio (sveglia, riferimento a PCB_APPL). In una locazione condivisa X *nota* (sh), I/O ha prima copiato il riferimento alla propria variabile BI/O (ind-sh) in cui ricevere il risultato di APPL;
3. lo handler dell'interruzione sulla CPU effettua la sveglia di APPL, ponendolo in stato di pronto;
4. quando APPL torna in esecuzione, esegue la computazione per trasformare il valore di A nel valore di B; quindi, accedendo alla locazione condivisa X *nota* (sh), ottiene il riferimento alla variabile BI/O (ind-sh) e vi copia il valore di B, effettuando, mediante una STORE di I/O (sh), anche una segnalazione di sveglia I/O (in anticipo rispetto alla copia di B, se I/O è progettata in maniera consistente).