Architettura degli Elaboratori

a.a. 2013/14 - secondo appello, 1 luglio 2014

Riportare nome, cognome, numero di matricola e corso A/B

Domanda 1

Un'unità di elaborazione U, operante su stream, ha il compito di valutare, per l'array di interi *A[N]* associato ad ogni elemento dello stream d'ingresso, il numero *S* di componenti aventi valore maggiore di zero. Il valore di S è l'elemento dello stream di uscita. N = 64K. In dettaglio, le specifiche sono le seguenti:

1. sullo stesso chip di U sono presenti una cache primaria C1 e una cache secondaria C2, associative su insiemi e operanti su domanda, con γ_1 = 32K, σ_1 = 8, γ_2 = 256K , σ_2 = 128;
2. memoria esterna M di capacità 4G, interallacciata con m = 8, $\tau_M = 20\tau$ (τ = ciclo di clock di U), e struttura di interconnessione ad albero con latenza di trasmissione dei collegamenti uguale a 1τ;
3. la computazione fa riferimento a un insieme di 256 array *A*;
4. l'elemento dello stream di ingresso è un identificatore unico intero di un array *A*;
5. non esiste una regola a priori su come i 256 array sono allocati, cioè il microprogramma deve essere indipendente dalla specifica allocazione dei 256 array.

a) Con l'obiettivo di massimizzare la banda ideale di U, scriverne il microprogramma, caratterizzare il funzionamento della gerarchia di memoria, e valutare la banda ideale di U in funzione di N e τ.

b) Valutare la banda effettiva di U nel caso che la banda richiesta sia uguale a $1/6N\tau$.

Domanda 2

L'istruzione `LOAD Ra, Rb, Rc` viene utilizzata in un dato programma in modalità Memory Mapped I/O. Con riferimento a tale istruzione:

a) spiegarne la semantica nel caso specifico (Memory Mapped I/O);

b) spiegare come la sua presenza influenza la fase di compilazione e di creazione del processo;

c) spiegarne in dettaglio il funzionamento dell'interprete firmware per un'architettura di CPU pipeline, distinguendo i casi in cui venga usata o non venga usata la gerarchia di memoria;

d) determinarne: *i)* il tempo di servizio ideale, *ii)* la latenza che va considerata nella valutazione di dipendenze logiche indotte dall'istruzione stessa.

Domanda 3

Spiegare l'implementazione di una memoria RAM di capacità N usando indirizzi di N bit, con il vincolo che uno e un solo bit dell'indirizzo valga 1, e determinarne il tempo di accesso.

Soluzione

Domanda 1

La corrispondenza tra identificatori di array (*id_A* – 8 bit) e i rispettivi indirizzi base di memoria (*ind_A* – 32 bit) viene data attraverso una tabella realizzata come vettore *TAB[256]*. La tabella può essere implementata da una memoria RAM di 256 registri interna a U, oppure può essere allocata in memoria: nel secondo caso, essendo soggetta a riuso, rimane permanentemente in C1 con una penalità trascurabile (2τ) nella trasformazione iniziale di *id_A* in *ind_A*. Nel seguito verrà usata la soluzione con memoria di registri nella PO di U.

Il microprogramma di U è scritto con accessi a domanda-risposta a C1, indipendentemente dal funzionamento e dalle prestazioni della gerarchia di memoria che dipendono dalla progettazione di C1 e C2. Per massimizzare la banda ideale di U:

i) visto il ridotto overhead per ogni accesso a C1, il microprogramma di U deve minimizzare il numero di cicli di clock, in particolare eseguendo nella stessa microistruzione il calcolo sull'elemento di A letto e la richiesta di lettura del prossimo elemento;

ii) occorre minimizzare la penalità dovuta ai fault di cache agendo sulla progettazione della gerarchia M – C2 – C1, visto che la computazione non è predisposta a ottimizzazioni sull'uso della gerarchia di memoria su domanda.

i) Microprogramma:

// Interfacce: ingresso *(id_A, RDYIN, ACKIN)*, uscita *(OUT* su 17 bit, *RDYOUT, ACKOUT)*, verso C1 *(IND, OP, RDYOUTC)*, da C1 (DATAIN, RDYINC). Viene usato un contatore di passi *I* a 16 bit, inizializzato a *N–2*, decrementato ad ogni passo, e testato per zero per eseguire diversamente l'ultimo passo che comporta l'ultimo accesso //

0. (RDYIN = 0) nop, 0;

   (= 1) reset RDYIN, set ACKIN, TAB[id_A] → ind_A, TAB[id_A] → IND, read → OP; set RDYOUTC, N – 2 → I, 0 → S, 1

1. (RDYINC, $DATAIN_0$, or(DATAIN), or(I) = 0 – – –) nop, 1;

   (= 1 0 1 1) reset RDYINC, **S + 1 → S**, I – 1 → I, ind_A + 1 → ind_A, ind_A + 1 → IND, read → OP; set RDYOUTC, 1;

   (= 1 0 1 0) reset RDYINC, **S + 1 → S,** ind_A + 1 → IND, read → OP, set RDYOUTC, 2;
   // richiesto l'ultimo accesso //

   (= 1 0 0 1, 1 1 – 1) reset RDYINC, I – 1 → I, ind_A + 1 → ind_A, ind_A + 1 → IND, read → OP; set RDYOUTC, 1;

   (= 1 0 0 0, 1 1 – 0) reset RDYINC, ind_A + 1 → IND, read → OP; set RDYOUTC, 2;
   // richiesto l'ultimo accesso //

2. (RDYINC, $DATAIN_0$, or(DATAIN), ACKOUT = 0 – – –, 1 – – 0) nop, 2;

   (= 1 0 1 1) reset RDYINC, **S + 1 → OUT**, set RDYOUT, reset ACKOUT, 0;

   (= 1 0 0 1, 1 1 – 1) reset RDYINC, **S → OUT**, set RDYOUT, reset ACKOUT, 0

Il tempo di calcolo interno in assenza di fault di cache è dato da:

$$T_0 = N\,(\tau + t_{c\,1}) = 3\,N\,\tau$$

ii) Per quanto riguarda l'effetto dei fault di cache, A gode solo della proprietà di località, per cui il working set è costituito dal blocco corrente di A: verranno generati N/σ_2 fault in C2 e N/σ_1 fault in C1.

Per massimizzare la banda, la gerarchia di memoria è progettata in modo da esplicitare il parallelismo tra M-C2 e C2-C1: il funzionamento di C2 prevede che, *per ogni blocco-C2*,

- il primo blocco-C1 ottenuto da M sia inoltrato subito anche a C1,
- la lettura dei rimanenti blocchi-C1 da M a C2 avvenga in parallelo all'elaborazione di U con C1.

Quindi:

$$T_{fault} = \frac{N}{\sigma_2}\,T_{trasfM-C2}(\boldsymbol{\sigma_1}) + \frac{N}{\sigma_1}\,T_{trasfC\,2-C1}(\sigma_1)$$

dove, essendo $\sigma_1 = m$:

$$T_{trasfM-C2}(\boldsymbol{\sigma_1}) = 2\,T_{tr} + \tau_M + (\tau + T_{tr})\,lg_2 m$$

Poiché richieste consecutive di blocchi-C1 da C1 a C2 avvengono ogni $3\sigma_1\tau = 24\tau > \tau_M$, per ogni blocco C2 i blocchi-C1 richiesti (dopo il primo) si troveranno già in C2, quindi:

$$T_{trasfC\,2-C1}(\sigma_1) = 2\,\sigma_1\,\tau$$

Si ha:

$$T_{fault} \sim 2.2\,N\,\tau$$

Il tempo di calcolo interno di U vale:

$$T = T_0 + T_{fault} = 5.2\,N\,\tau$$

Essendo molto maggiore della latenza di comunicazione del risultato finale, questo è anche il tempo di servizio ideale di U.

La banda offerta:

$$B_{off} = \frac{1}{5.2\,N\,\tau}$$

è quindi maggiore della banda richiesta, per cui la banda effettiva di U è uguale alla banda richiesta:

$$B = \frac{1}{6\,N\,\tau}$$

Domanda 2

a) I registri generali RG[Ra] e RG[Rb] contengono valori interi che, sommati, danno luogo ad un indirizzo logico al quale, nella funzione di traduzione degli indirizzi del processo, viene fatto corrispondere un indirizzo fisico della memoria locale di una determinata unità di I/O. Il valore del dato, letto a tale indirizzo, viene scritto nel registro generale RG[Rc].

b) Sia X l'informazione MV[RG[Ra] + RG[Rb]]. In fase di compilazione, una volta allocato X nella memoria virtuale all'indirizzo logico prescelto, e inizializzati opportunamente i registri generali, nel file di configurazione viene registrato che l'informazione a tale indirizzo logico deve essere fisicamente allocata nella memoria locale di una determinata unità di I/O. In generale, questa corrispondenza riguarderà l'intera pagina MV-M contenente X.

Inoltre, nel file di configurazione è anche registrato se tale pagina è *inizializzata* al valore presente nel file eseguibile, e se è *condivisa*.

In fase di creazione del processo, consultando il file di configurazione:

- si provvede alla corretta inizializzazione della Tabella di Rilocazione usando l'identificatore della pagina fisica di I/O allocata alla pagina logica;
- se la pagina è inizializzata nel file eseguibile: se non è condivisa, oppure è condivisa ma non ancora caricata per altri processi, si provvede a comunicare alla fase di caricamento che la pagina logica deve essere copiata nella pagina fisica della memoria locale di I/O indicata dalla Tabella di Rilocazione.

c) IU invia l'istruzione ad EU incrementando il semaforo associato a RG[Rc], calcola l'indirizzo logico se e quando RG[Ra] e RG[Rb] sono aggiornati, e invia la richiesta di lettura a MMU_D.

Si distinguono due soluzioni architetturali:

1. Se l'architettura prevede che ogni accesso in memoria debba essere effettuato tramite cache, MMU_D traduce l'indirizzo e (se non c'è fault di pagina MV-M) passa la richiesta di lettura, contenente l'indirizzo fisico, all'unità cache dati primaria CD, la quale invia il dato letto a EU. Se CD genera fault di blocco-C1, la richiesta di *trasferimento blocco* è inoltrata da CD (a C2 e da questa) all'unità di I/O il cui identificatore è riconosciuto come un campo dell'indirizzo fisico (la parte più significativa, eccetto bit iniziali che distinguono la memoria principale dal complesso delle memorie di I/O).

   [Va da sé che effettuare caching di informazioni caratterizzate da sola località e presenti in memorie di I/O può rivelarsi inefficiente, a meno che il sottosistema di I/O e la struttura di interconnessione non siano realizzate a larga banda analogamente alla memoria principale: questo aspetto però non è centrale agli effetti di quanto richiesto].

2. Se l'architettura (con sottosistema di I/O e struttura d'interconnessione tradizionali) prevede che la cache possa essere disabilitata, e questa opzione è applicata all'istruzione in oggetto, allora MMU_D instrada la richiesta di *singola* lettura all'unità di I/O il cui identificatore è riconosciuto come un campo dell'indirizzo fisico. Trovandoci in un'architettura CPU pipeline, MMU_D non attende deterministicamente il dato a domanda

e risposta, ma, una volta inoltrata la richiesta, porta avanti altre richieste da IU se possibile (funzionamento in-order FIFO – analogamente all'unità EU_Master per operazioni lunghe): la disponibilità del dato da scrivere in RG[Rc] non interessa l'istruzione Load di per sé, ma interessa eventuali successive istruzioni sulle quali la Load induca una dipendenza logica. Quando il dato sarà ricevuto (nondeterministicamente), viene inoltrato a EU (via CD).

[L'unità di I/O è automaticamente autorizzata (arbitraggio implicito) a scrivere il dato sul Bus. Non più di una richiesta di lettura alla volta può essere inoltrata da MMU_D, né può essere accettata alcuna interruzione, prima che MMU_D abbia ricevuto il risultato della lettura].

d) Il tempo di servizio ideale rimane uguale a uno slot temporale *t*, come per ogni altra istruzione e in particolare come per ogni Load, in quanto, in ognuna delle architetture 1 e 2, nessun ritardo è aggiunto all'elaborazione di MMU_D e CD. In particolare, nella soluzione 2 MMU_D non funziona a domanda e risposta, e la richiesta all'unità di I/O può essere inoltrata via un'apposita unità buffer sullo stesso chip CPU, in modo da non pagare, nel tempo di servizio ideale, la latenza della comunicazione su un collegamento esterno.

Il punto *ii)* si riferisce alla latenza L_s da considerare della valutazione del ritardo Δ dovuto a dipendenze logiche IU-EU. La Load in oggetto può indurre una dipendenza IU-EU, nel qual caso L_s entra nella valutazione di Δ_1; oppure una dipendenza EU-EU, nel qual caso, se la dipendenza influenza a sua volta una dipendenza IU-EU, L_s entra nella valutazione di Δ_2.

Nella soluzione 1 la latenza è quella di una normale Load in assenza di fault di cache:

$$L_s = t$$

e quindi

$$L_{pipe} = 0$$

La penalità (T_{fault}) dovuta ai fault di cache non fa parte di L_s e viene valutata nella determinazione del tempo di completamento del programma.

Nella soluzione 2, la latenza del servente è incrementata dal ritardo di un accesso a domanda e risposta a una memoria esterna:

$$L_{pipe} = 2(\tau + T_{tr}) + \tau_{MI/O}$$

$$L_s = t + 2(\tau + T_{tr}) + \tau_{MI/O}$$

Come detto sopra, il fatto che MMU_D possa proseguire ad elaborare altre richieste prima di ricevere il risultato, e che quindi tale latenza possa non essere pagata interamente, rientra nelle usuali ottimizzazioni statiche o dinamiche per l'architettura di CPU pipeline.

Domanda 3

In base alle proprietà della funzione commutatore con variabili di controllo C e ingressi X_0 … X_{n-1},

$$\bigvee_{i=0}^{n-1} p_i(C) \wedge X_i$$

dove $p_i(C)$ è vero se e solo se la configurazione di C è in corrispondenza biunivoca con *i*, il commutatore per la lettura ha un unico livello di porte AND: la *i*-esima porta ha in ingresso il bit *i*-esimo dell'indirizzo e la parola della locazione di indirizzo *i* (*w* porte AND a 2 ingressi per ogni locazione, con *w* lunghezza di parola).

Il selezionatore per la scrittura ha un unico livello di porte AND: la *i*-esima porta ha in ingresso il bit *i*-esimo dell'indirizzo e il segnale β di abilitazione alla scrittura della locazione di indirizzo *i*.

Se N_0 è il massimo numero di ingressi per porta, la funzione di OR del commutatore per la lettura è realizzata con un albero di $\lceil \log_{N_0} N \rceil$ livelli. Quindi, il tempo di accesso, uguale al ritardo di stabilizzazione del commutatore, è:

$$t_a = \left(1 + \lceil \log_{N_0} N \rceil\right) t_p$$