Architettura degli Elaboratori

Appello del 8 giugno 2012

Riportare su tutti i fogli consegnati nome, cognome, numero di matricola, corso A/B, e la sigla NEW (per nuovo ordinamento), oppure OLD-0 (per vecchio ordinamento, nuovo programma), oppure OLD-1 (per vecchio ordinamento, vecchio programma). I risultati verranno pubblicati sulle pagine web del corso/dei docenti appena disponibili.

Domanda 1 (tutti)

a) Una unità contiene 256 registri indipendenti di 32 bit, e riceve in ingresso un booleano OP, una parola X di 32 bit e un identificatore di uno dei registri. Se OP = 0 scrive X nel registro identificato. Se OP = 1 invia in uscita il risultato della *reduce* della somma applicata a tutti i registri. Realizzare l'unità con il minimo tempo di elaborazione e valutare tale tempo in funzione del ritardo di t_p di una porta logica con al più 8 ingressi, con $T_{ALU} = 5t_p$, fornendo adeguate spiegazioni.

b) Come sopra, ma con i 256 registri facenti parte di 4 moduli di un componente logico memoria con organizzazione sequenziale e tempo di accesso da determinare in funzione di t_p.

Domanda 2 (tutti)

Valutare l'insieme di lavoro della seguente computazione, dove $N = 4K$ e h è un parametro d'ingresso:

```
int A[N][N];
    ∀ i = 0 .. N – 1:
        ∀ j = 0 .. N – 1:
            if A[i][j] ≠ A[h][j]
                then A[i][j] = A[i][j] – A[h][j]
```

per una gerarchia di memoria memoria principale – cache primaria. Valutare in funzione di N il tempo speso in fault di cache, nell'ipotesi che la capacità della cache sia di 16K, i blocchi siano di 16 parole e la memoria principale sia realizzata da una singola unità con ciclo di clock uguale a 20τ, con $T_{tr} = 7\tau$.

Domanda 3 (NEW, OLD-0)

Dare una descrizione dettagliata dell'Unità Esecutiva parallela di una architettura D-RISC pipeline scalare in-order. Dalla descrizione, ricavare il tempo di servizio ideale dell'Unità Esecutiva. Determinare inoltre il tempo di servizio effettivo in assenza di degradazioni delle prestazioni del programma e il tempo di servizio effettivo in presenza di tali degradazioni.

Domanda 3 (OLD-1)

Pronunciarsi sulla verità o meno delle seguenti affermazioni:

a) dato un processo con n strutture dati private e m condivise, per quanto riguarda l'allocazione di tali strutture il suo spazio di indirizzamento consta di $n + m$ puntatori alle strutture stesse;

b) ogni volta che un processo modifica una struttura dati S, la modifica deve essere riportata anche nella propria memoria virtuale;

c) ogni volta che un processo modifica una struttura dati X condivisa, la modifica deve essere riportata anche nelle memorie virtuali di tutti i processi che condividono X;

d) la Tabella di Rilocazione di un processo viene trasferita interamente all'interno della MMU all'atto della commutazione di contesto;

e) la Tabella di Rilocazione di un processo viene trasferita parzialmente all'interno della MMU all'atto della commutazione di contesto.

Sintesi della soluzione

(da completare con spiegazioni ulteriori)

Domanda 1 (tutti)

a) Essendo i registri tutti indipendenti, la reduce è eseguibile in un singolo ciclo di clock mediante un albero binario di 255 ALU, che garantisce il minimo tempo di elaborazione possibile, proporzionale a $lg_2 N$:

0. (RDYIN, OP, ACKOUT = 0 - - , 1 1 0) nop, 0;

   (= 1 0 -) reset RDYIN, set ACKIN, X → REG[ID], 0;

   (= 1 1 1) reset RDYIN, set ACKIN, reset ACKOUT, set RDYOUT, *reduce* (REG, +) → OUT, 0

L'unità è implementata da una singola rete sequenziale di Moore, con ritardo nullo della funzione delle uscite.

La scrittura è implementata con un selezionatore avente ritardo $2t_p$ (due livelli AND) cui va a sommarsi il ritardo di stabilizzazione t_p del segnale di abilitazione $\beta_1 = RDYIN\ \overline{OP}$.

Il calcolo della reduce ha ritardo uguale a

$$T_{calc\text{-}reduce} = lg_2 N * T_{ALU} = 40t_p.$$

Le variabili di controllo per le interfacce di ingresso ($\beta_2 = RDYIN\ \overline{OP} + RDYIN\ OP\ ACKOUT$) e di uscita ($\beta_3 = RDYIN\ OP\ ACKOUT$) si stabilizzano in parallelo alla scrittura o alla reduce.

In conclusione, la funzione di transizione dello stato interno si stabilizza in $40t_p$, da cui

$$T = \tau = 41t_p$$

b) Non è più possibile l'esecuzione della reduce in un singolo ciclo di clock. Sfruttando la memoria modulare, con un albero binario a due livelli di ALU, l'algoritmo iterativo della fase di calcolo della reduce viene eseguito in 256/4 = 64 cicli di clock (ovviamente, la prima operazione esterna è ancora eseguita in un ciclo di clock). Usando un contatore I a 7 bit:

0. (RDYIN, OP, = 0 -) nop, 0;

   (= 1 0) reset RDYIN, set ACKIN, X → M[ID], 0;

   (= 1 1) reset RDYIN, set ACKIN, 0 → I, 0 → S, 1

1. (I_0, ACKOUT = 0 -) S + *reduce* (M_0[I], M_1[I], M_2[I], M_3[I], +) → S, I + 1 → I, 1;

   (= 1 0) nop, 1;

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

(potevano ovviamente essere risparmiati i cicli di clock dell'inizializzazione e della terminazione).

In lettura la memoria M fornisce quattro parole in parallelo indirizzando i quattro moduli in modo del tutto indipendente, mentre per la scrittura viene indirizzata l'intera memoria a singolo accesso. Il tempo di accesso vale $3\ t_p$.

Essendo certamente $T_{\omega PC} = T_{\sigma PC} = 2\ t_p$ e $T_{\omega PO} = 0$, si ha:

$$T_{\sigma PO} = t_a + T_{reduce} + T_{ALU} + T_{K\_S} = t_a + 3T_{ALU} + T_{K\_S} = 20\ t_p$$

$$\tau = 23\ t_p$$

$$T_0 = \tau$$

$$T_1 = 66\,\tau = 1510\ t_p$$

(riducibile a $64\ \tau = 1472\ t_p$).

Domanda 2 (tutti)

Essendoci riuso della riga *A[h][*]*, l'insieme di lavoro è rappresentato dal codice (presumibilmente un blocco), dal generico blocco per riferire *A[i][j]* e da tutti i blocchi ($N/\sigma = 256$) della riga *A[h][*]*. Il numero di fault di cache è dato da

$$N_{fault} = N^2/\sigma$$

Il tempo impiegato per trasferire un blocco è:

$$T_{trasf} = 2\,T_{tr} + \sigma\,\tau_M + \tau = 335\,\tau$$

da cui la penalità per i fault di cache:

$$T_{fault} = N_{fault} * T_{trasf} \sim 21\,N^2\,\tau$$

Domanda 3 (NEW, OLD-0)

Per la descrizione si veda il materiale didattico. Da dettagliare:

- realizzazione delle operazioni lunghe in pipeline,
- realizzazione e struttura dell'unità EU_Master incaricata della distribuzione, della lettura, modifica e controllo di consistenza dei registri generali e dei registri in virgola mobile, dell'esecuzione diretta delle operazioni corte su interi e delle LOAD.

EU_Master, *presa a sé stante*, ha un tempo di servizio *ideale* di un ciclo di clock, in quanto, in presenza di registri aggiornati, è in grado di eseguire in parallelo nella stessa microistruzione tutte le azioni richieste (dimostrare).

Il tempo di servizio ideale di ogni stadio del pipeline, *preso a sé stante*, è ancora un singolo ciclo di clock.

A causa della latenza delle comunicazioni, tutte le unità della struttura pipeline hanno tempo di servizio effettivo uguale a due cicli di clock ($t = 2\tau$). Inoltre, il tempo di servizio *effettivo non può essere inferiore al tempo di interarrivo* a EU, che dipende dalla probabilità che una istruzione sia eseguita dalla EU:

$$\frac{T}{prob_{aritm} + prob_{load}}$$

In assenza di degradazioni sul tempo medio di servizio per istruzione dello specifico programma, questo vale:

$$\frac{t}{prob_{aritm} + prob_{load}}$$

In presenza di degradazioni:

$$\frac{T}{prob_{aritm} + prob_{load}} = \frac{t\,(1 + \lambda) + \Delta}{prob_{aritm} + prob_{load}}$$

dove Δ rappresenta il ritardo medio dovuto alle dipendenze logiche IU-EU, sulle quali hanno eventualmente impatto alcune dipendenze logiche EU-EU:

$$\Delta = \Delta_1 + \Delta_2$$

[Il tempo di interarrivo rappresenta un lower bound del tempo di servizio effettivo della EU, comunque un valore sufficientemente attendibile. In realtà, per ottenere una valutazione formalmente corretta, al tempo di interarrivo andrebbe sommato il ritardo medio dovuto anche a quelle dipendenze logiche EU-EU *che non rientrano in* Δ_2: queste, infatti, pur non impattando sulle dipendenze IU-EU, contribuiscono ad aumentare il tempo di servizio di una *EU in-order*. Tale fenomeno potrebbe avere effetto sul tempo di servizio per istruzione della CPU solo se il fattore di utilizzazione della EU divenisse maggiore di uno. **]**

Domanda 3 (OLD-1)

a) *dato un processo con n strutture dati private e m condivise, per quanto riguarda l'allocazione di tali strutture il suo spazio di indirizzamento consta di n + m puntatori alle strutture stesse.* – Falso: se fosse vero non sarebbe possibile indirizzare correttamente, mediante indirizzi logici, strutture dati costituite da più di una parola;

b) *ogni volta che un processo modifica una struttura dati S, la modifica deve essere riportata anche nella propria memoria virtuale* – Falso: sarebbe inutile e dannoso; al più le modifiche vanno riportate nelle copia in memoria secondaria in caso di deallocazione dinamica e successiva riallocazione o, per i risultati che il processo deve restituire, all'atto della terminazione. Inoltre, vale il punto *c)* seguente;

c) *ogni volta che un processo modifica una struttura dati X condivisa, la modifica deve essere riportata anche nelle memorie virtuali di tutti i processi che condividono X.* – Falso: tali strutture sono, per definizione, presenti in singola copia in memoria principale. Nelle memorie virtuali tali strutture "non hanno valore", nel senso che , agli effetti del processo, è necessario e sufficiente poterle indirizzare;

d) *la Tabella di Rilocazione di un processo viene trasferita interamente all'interno della MMU all'atto della commutazione di contesto.* – Falso: il problema non è tanto la dimensione della Tabella, quanto il fatto che, con alta probabilità, solo un sottoinsieme (piccolo) delle entrate viene utilizzata durante un'esecuzione (spiegare perché). Quindi, un'allocazione dinamica della Tabella in MMU comporta un numero di accessi in memoria (molto) minore (al più uguale) a quello di un'allocazione statica;

e) *la Tabella di Rilocazione di un processo viene trasferita parzialmente all'interno della MMU all'atto della commutazione di contesto.* – Vero solo a condizione che la parte parzialmente trasferita all'atto della commutazione di contesto sia relativa a una o più entrate della Tabella necessarie per rilocare gli indirizzi logici che riferiscono la Tabella stessa. In altre soluzioni (indirizzo fisico della Tabella passato alla MMU) nessuna entrata della Tabella viene trasferita in MMU all'atto della commutazione di contesto.