Architettura degli Elaboratori

Appello straordinario per studenti fuori corso – 3 aprile 2013

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

Si raccomanda di attenersi strettamente e puntualmente alle domande, fornendo spiegazioni chiare e sintetiche.

Domanda 1

Un'unità di elaborazione U riceve contemporaneamente in ingresso una *n*-upla di interi (X_0, X_1, …, X_{n-1}) e calcola su di essi un'espressione del tipo X_0 op_1 X_1 op_2 X_2 op_3 … op_{n-1} X_{n-1}, dove ogni op_j può essere un'operazione di addizione o di sottrazione. Il risultato è inviato in uscita.

Detti, come di solito, k e τ i fattori di $T = k\tau$ (tempo di calcolo interno o tempo di servizio ideale senza tenere conto delle comunicazioni), si possono avere varie realizzazioni di U con valori diversi di k e di τ.

Dimostrare che la realizzazione con il minimo T è quella avente τ massimo. Per semplicità supporre che le operazioni op_1 op_2 op_3 … op_{n-1} siano ordinate totalmente come indicato.

Fare un esempio.

Domanda 2

Un'unità di elaborazione U riceve da U1 uno stream illimitato di valori interi x. Sia $A[10^4]$ un array di interi. Per ogni x, U calcola il numero di occorrenze di x in A e invia il risultato a U2. I collegamenti tra le unità hanno latenza di trasmissione uguale a 5 cicli di clock.

Si considerino due diverse versioni di U:

1) A è contenuto in una memoria di registri interna a U,
2) A è contenuto in una memoria esterna M. U utilizza una gerarchia di memoria M-C, con C cache sullo stesso chip di U, associativa, di capacità 16K.

Dimostrare che, a parità di ciclo di clock, il rapporto tra il tempo di servizio ideale della versione 2) e quello della versione 1) è uguale a 3.

Domanda 3

Per una qualunque gerarchia di memoria a due livelli M2-M1, spiegare perché la funzione che lega la probabilità di fault alla dimensione della pagina ha un minimo.

Dire se questo è vero anche nel caso che tutti i dati siano caratterizzati da riuso.

Domanda 4

Si consideri una CPU pipeline la cui l'Unità Esecutiva contiene unità funzionali in pipeline: una per operazioni intere lunghe, una per operazioni reali corte e una per operazioni reali lunghe.

Viene eseguito un programma di benchmark costituito da sole istruzioni aritmetiche.

a) Dimostrare che l'Unità Esecutiva data *non può* essere collo di bottiglia se la CPU è scalare.

b) Dimostrare che l'Unità Esecutiva data *può* essere collo di bottiglia se la CPU è superscalare a m vie ($m > 1$). In questo caso dire se esiste una realizzazione dell'Unità Esecutiva capace di eliminare il collo di bottiglia.

Soluzione

Domanda 1

La soluzione con una sola microistruzione ($k = 1$) è quella con T minimo e τ massimo: tutta l'espressione è realizzata mediante una cascata di $n - 1$ ALU. U è una singola rete sequenziale il cui ciclo di clock è dato dal ritardo della funzione di transizione dello stato interno:

$$T = \tau = (n-1)\, T_{ALU} + \delta$$

Esempio: per A + B + C –D si ha $T = 3T_{ALU} + \delta$.

La soluzione con ciclo di clock minimo e $k = n - 1$ è caratterizzata da:

$$T' = (n-1)\, \tau' = (n-1)\, (T_{\omega PC} + T_{ALU} + T_K + \delta) > T$$

dove il commutatore K è quello all'ingresso dell'unica ALU utilizzata nella PO.

Esempio: per A + B + C –D si ha $T' = 3\tau' = 3\,(4t_p + T_{ALU} + \delta)$.

Tutte le soluzioni intermedie, consistenti nel decomporre l'espressione in sottoespressioni ognuna eseguita in una microistruzione, sono migliori dell'ultima, ma sempre con tempo di calcolo > T.

[Va da se che, in assenza del vincolo sull'ordinamento totale delle operazioni, la soluzione con unico ciclo di clock avrebbe tempo di calcolo logaritmico in n, ma questo non cambia i termini del confronto.]

Domanda 2

La latenza delle comunicazioni di U con U2 è data da:

$$L_{com} = 2\,(\tau + T_{tr}) = 12\tau$$

Il tempo di servizio ideale di U è dato da:

$$T_{id} = \max(T_{calc}\,, L_{com}) = T_{calc}$$

dove T_{calc} è il tempo di calcolo interno $O(N)$ per eseguire la computazione su ogni x e sull'intero array A.

Per la versione 1):

$$T_{id-1} = T_{calc-1} = N\,\tau$$

Per la versione 2), l'insieme di lavoro della gerarchia di memoria è dato da tutto l'array A, in quanto A è costante e utilizzato per tutti gli elementi dello stream. Il *riuso* su A è esplicitabile, in quanto $N < \gamma_C$. Dopo il primo elemento dello stream, A risiederà permanentemente in cache, associando alla richiesta di accesso in memoria l'opzione "non deallocare". Poiché il tempo di servizio ideale è il valore *medio* per il generico elemento dello stream, e lo stream è illimitato, il trasferimento M-C dei blocchi di A per il primo elemento dello stream (T_{fault}) ha peso nullo.

Il microprogramma è un ciclo di N iterazioni, nell'i-esima delle quali viene effettuata la richiesta di lettura alla memoria esterna (un ciclo di clock) e atteso il dato $A[i]$ (due cicli di clock: cache associativa). L'elaborazione (test sul confronto ed eventuale incremento) viene effettuata nello stesso ciclo di clock in cui si richiede l'accesso ad $A[i+1]$.Quindi:

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

Domanda 3

La funzione ha un minimo legato allo sfruttamento della *località* presente nei programmi.

Sia una computazione operante su *n* strutture dati contemporaneamente, *dove ognuna gode solo della proprietà di località*. La località dell'*intera* computazione viene garantita se M1 è in grado di contenere almeno *n* pagine, una per ogni struttura. Per $\sigma > \gamma_1/n$, M1 non è in grado di garantire la località dell'intera computazione e la probabilità di fault cresce al crescere di σ. Se $\sigma < \gamma_1/n$, la probabilità di fault decresce al crescere di σ in quanto non viene ancora sfruttata al meglio la località di ogni singola struttura.

[Ovviamente, ogni computazione ha il suo *n*, quindi per ogni sistema non è possibile prevedere a priori un valore ottimo in assoluto di σ, che viene stimato mediamente su grossi campioni di programmi].

Se tutti i dati sono caratterizzati da riuso, e γ_1 è tale per cui il riuso è esplicitabile, esiste ancora un valore ottimo di σ in quanto nella fase transitoria, in cui tutte le *n* struttura sono trasferite in M1, vale ancora il principio di località (anche se il valore ottimo è meno vincolante sulle prestazioni, in quanto appunto legato all'ottimizzazione della sola fase transitoria).

Domanda 4

Ogni unità funzionale ha tempo di servizio ideale uguale al tempo di servizio ideale dell'intera CPU, cioè $Tid = t$ ($= \tau$ oppure $= 2\tau$ a seconda dell'implementazione delle comunicazioni).

Detto T_A il tempo di interarrivo a EU, per una determinata architettura è garantito (per costruzione) che EU Master abbia tempo di servizio ideale $\leq T_A$. Per il teorema dei serventi multipli, il tempo di interarrivo all'unità funzionale *i*-esima è quindi dato da:

$$T_{A-i} = \frac{T_A}{p_i}$$

dove p_i è la probabilità che l'operazione richiesta sia eseguita dall'unità funzionale *i*-esima. Affinché EU non sia collo di bottiglia è necessario che nessuna unità funzionale sia collo di bottiglia, quindi:

$$\forall\, i\colon\ \rho_i \ \leq 1 \quad \rightarrow \quad \forall\, i\colon\ t \ \leq \frac{T_A}{p_i}$$

a) Il tempo di interarrivo è uguale a *t*, in quanto nel benchmark in oggetto non esistono degradazioni delle prestazioni (l'architettura funziona come un pipeline puro). La relazione precedente è soddisfatta per ogni *i* e per qualunque valore di p_i, al limite anche $p_i = 1$ (tutte le istruzioni dello stream usano la stessa unità funzionale).

b) Il tempo di interarrivo è uguale a *t/m*, quindi la relazione precedente non può essere soddisfatta per tutte le unità funzionali.

Formalmente il collo di bottiglia può essere eliminato disponendo di *m* copie identiche di ogni unità funzionale, così che il tempo di servizio di ogni operazione divenga *t/m*.