# Architettura degli Elaboratori

a.a. 2012/13 – quarto appello, 20 gennaio 2014

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

## Domanda 1

*a)* Dimostrare che, detto $n$ il massimo numero d'ingressi per porta logica, se il numero di frasi condizionali distinte del microprogramma di un'unità di elaborazione è minore o uguale a $n$ allora è possibile progettare le funzioni della Parte Controllo di tale unità con ritardo di stabilizzazione minore o uguale a $2t_p$.

*b)* Dire se la seguente affermazione è vera o falsa, spiegando chiaramente la risposta: "condizione necessaria e sufficiente affinché un'unità di elaborazione abbia tempo di elaborazione uguale a un ciclo di clock è che la trasformazione dai valori d'ingresso ai valori di uscita sia una funzione pura".

## Domanda 2

Per una CPU D-RISC con istruzioni in virgola fissa e mobile, architettura pipeline scalare e Unità Esecutiva parallela, dimostrare che è possibile progettare l'unità EU Master con tempo di servizio ideale, senza tenere conto dell'impatto delle comunicazioni, uguale a un ciclo di clock. Il funzionamento è FIFO out-of-order con al massimo una istruzione in attesa.

La *dimostrazione* deve essere data basandosi sul funzionamento dettagliato di EU Master per tutte le funzionalità previste.

## Domanda 3

La seguente computazione:

*int A[N], B[N], C[N], D[N], X[N];*

$\forall$ *i = 0 .. N – 1:*

$\forall$ *j = 0 .. N – 1:*

*X[i] = F(A[i], B[i], C[j], D[j])*

con $N = 10^4$, è eseguita su un'architettura con le seguenti caratteristiche:

- CPU D-RISC con ciclo di clock $\tau$;
- cache dati primaria C1 su domanda, write-through, con capacità 32K e blocchi di ampiezza $\sigma_1$;
- cache secondaria C2 on chip, su domanda, con blocchi di ampiezza $\sigma_2 = q\ \sigma_1$;
- memoria principale M interallacciata con $m$ moduli e ciclo di clock $\tau_M$;
- interconnessione da M a CPU con struttura ad albero binario e latenza di trasmissione dei collegamenti $T_{tr}$.

Ricavare, in funzione di $N$, $\tau$, $\tau_M$, $m$, $\sigma_1$, $q$, $T_{tr}$ e $\varepsilon$, l'espressione del tempo di completamento in assenza di fault di cache per un determinato valore $\varepsilon$ dell'efficienza relativa della gerarchia di memoria per quel programma. Si assuma che il meccanismo write-through non abbia influenza sulle prestazioni e che il numero di fault di cache per il codice sia trascurabile. Non è nota la funzione $F$ né il suo tempo di servizio.

# Soluzione

## Domanda 1

**a)** Le frasi condizionali distinte sono in corrispondenza biunivoca con le combinazioni distinte delle variabili dello stato interno di PC e delle variabili d'ingresso di PC (variabili di condizionamento). Quindi, l'espressione logica di qualunque variabile dello stato di uscita e dello stato successivo è esprimibile come OR di al più *n* termini AND ognuno dei quali ha meno di *n* variabili (d'ingresso e dello stato interno) specificate (funzione basta su logaritmi, comunque limitata superiormente da *n*). Che i termini in OR siano al più *n* è provato dal fatto che il numero di frasi in cui una variabile di controllo o dello stato successivo vale uno è al più uguale (upper bound asintotico) al numero delle frasi stesse.

**b)** Falsa. La condizione è solo sufficiente e, *inoltre*, a condizione che si riesca a realizzare una rete combinatoria corrispondente alla funzione.

Infatti, affinché il tempo di elaborazione (tempo di servizio ideale senza considerare l'impatto delle comunicazioni) sia uguale a un ciclo di clock occorre che il microprogramma sia descritto da una sola microistruzione. Questo è certamente possibile (*condizione sufficiente*) se la trasformazione ingresso-uscita è una funzione pura e se si riesce a realizzare una rete combinatoria corrispondente:

0. …. $F(IN) \rightarrow OUT, 0$

Ma anche in presenza di uno stato interno che condizioni la trasformazione ingresso-uscita è possibile scrivere il microprogramma mediante una sola microistruzione (*negazione della condizione necessaria*), a condizione che tanto la trasformazione (ingresso, stato)-uscita, quanto l'aggiornamento dello stato interno siano esprimibili come funzioni pure e che si riesca a realizzarle mediante reti combinatorie:

0. …. $F(IN, S) \rightarrow OUT, G(IN, S) \rightarrow S, 0$

## Domanda 2

Per dimostrare l'asserto occorre progettare EU Master mediante il microprogramma, seppure in una versione ad alto livello che evidenzi la struttura in termini di nondeterminismo e parallelismo.

L'unità EU Master è descrivibile da un microprogramma costituito da una sola microistruzione nella quale vengono testate contemporaneamente le seguenti variabili di condizionamento:

- RDY da IU per la ricezione dell'istruzione,
- RDY da DM per la ricezione del dato di una LOAD,
- RDY dalle unità funzionali per la ricezione dei risultati da scrivere nei registri generali e nei registri in virgola mobile,
- Codice operativo (8 variabili di condizionamento),
- OR applicato all'uscita dei semafori associati agli operandi di una istruzione aritmetica: 2 variabili di condizionamento riconosciute parametricamente in funzione degli indirizzi dei primi due registri riferiti dall'istruzione,
- Come sopra per i semafori associati agli operandi dell'eventuale istruzione in attesa,
- Condizione di attesa di un'istruzione sull'aggiornamento dei registri sorgente,
- ACK verso le unità funzionali,
- ACK verso IU.

Il funzionamento nondeterministico deve essere realizzato eseguendo *in parallelo nello stesso ciclo di clock* tutte le azioni corrispondenti agli eventi non deterministici: ricezione di una nuova istruzione da IU e ricezione di un risultato dalle unità funzionali.

Ricevendo un'istruzione da IU, nello stesso ciclo di clock viene decodificata e vengono testati i semafori associati ai registri riferiti. Se l'istruzione è LOAD è significativo anche il test di RDYDM. Le azioni conseguenti sono l'esecuzione locale (LOAD o aritmetico-logica corta su interi), o l'inoltro della tupla (operazione, valori operandi) all'unità funzionale corrispondente, o la messa in attesa dell'istruzione utilizzando un apposito registro e opportune informazioni associare ai registri sorgente dei quali si attende l'aggiornamento, in ogni caso aggiornando il semaforo del registro destinazione. Nella situazione di (una sola) istruzione in attesa, l'unità non si blocca (FIFO out-of-order) e passa a ricevere la prossima istruzione da IU.

*Nello stesso ciclo di clock* può anche essere ricevuto un valore dalle unità funzionali: in parallelo alle azioni di cui sopra (*nondeterminismo implementato come parallelismo*) sono effettuati l'inoltro del risultato a IU, l'aggiornamento del registro e del relativo semaforo, il test del valore precedente del semaforo, e l'eventuale "sveglia" di un'istruzione in attesa: questa istruzione è ora nelle condizioni di essere eseguita in un ciclo di clock per cui valgono tutte le considerazioni di cui sopra.

Se gli eventi nondeterministici fossero serviti in sequenza il tempo di servizio ideale sarebbe maggiore di un ciclo di clock. Ad esempio, servendo solo la richiesta dalle unità funzionali verrebbe ritardato il servizio di una nuova istruzione da IU.

In conclusione, è stato dimostrato che in qualunque ciclo di clock vengono eseguite in parallelo tutte le azioni conseguenti a tutti agli eventi non deterministici verificatisi sulle varie interfacce d'ingresso, da cui l'asserto.

## Domanda 3

Il tempo di completamento $T_{c0}$ in assenza di fault di cache si esprime a partire dall'efficienza relativa della gerarchia di memoria per quel programma:

$$\epsilon = \frac{T_{c0}}{T_{c0} + T_{fault}}$$

Quindi, per $\varepsilon < 1$ ($T_{fault} > 0$):

$$T_{c0} = \frac{\varepsilon}{1 - \varepsilon} T_{fault}$$

dove, funzionando sia C1 che C2 su domanda:

$$T_{fault} = N_{fault\ -C2} * T_{trasfM\ -C2} + N_{fault\ -C1} * T_{trasfC\ 2-C1}$$

Occorre esprimere la funzione:

$$T_{fault} = T_{fault}\ (N, \tau, \tau_M, m, \sigma_1, q, T_{tr})$$

L'insieme di lavoro dei dati del programma è costituito dal blocco corrente di A, dal blocco corrente di B, da tutto C e da tutto D, in quanto A, B sono caratterizzati da sola località e C, D da località e riuso. Essendo *N* ampiamente inferiore alla capacità della cache dati primaria (e quindi, a maggior ragione, ampiamente minore della capacità di C2), il riuso è effettivamente esplicitabile utilizzando l'annotazione "non deallocare" nelle istruzioni di LOAD su C[j] e D[j]. Quindi il numero di fault per C1 e C2 è dato da:

$$N_{fault\ -C1} = 4\ \frac{N}{\sigma_1} \qquad\qquad N_{fault\ -C2} = 4\ \frac{N}{q\sigma_1}$$

Gli accessi a X sono in sola scrittura, quindi non richiedono trasferimento di blocco, e, per ipotesi, il meccanismo write-through non ha impatto sul tempo di completamento.

Con una struttura ad albero binario, la latenza di trasferimento di un blocco di ampiezza $\sigma_2 = q\sigma_1$ da M a C2 è espresso da:

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

e la latenza di trasferimento di un blocco di ampiezza $\sigma_1$ da C2 a C1:

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

In conclusione:

$$T_{c0} = \frac{\varepsilon}{1-\varepsilon}\,4\,\frac{N}{\sigma_1}\left\{\frac{2T_{tr}}{q} + \frac{\sigma_1}{m}\tau_M + \frac{\tau + T_{tr}}{q}\ lg_2 m + 2\,\sigma_1\,\tau\right\}$$