CALCOLATORI ELETTRONICI M

ARCHITETTURA DEI CALCOLATORI ELETTRONICI M

24/01/2022 – Online – Esercizio 1 & 2

Tempo disponibile: 150 minuti
– Link al testo in formato pdf
– Link ai fogli per lo svolgimento Es.1, Es.2

Si fornisca la soluzione al compito riempiendo il foglio di calcolo disponibile ai link sovra riportati e consegnandolo allegandolo al quiz. In caso non sia possibile usare il foglio di calcolo si inviino sempre allegandole al quiz le foto alla soluzione cartecea. Inserire in tutti i nomi dei fili Cognome_Nome_Matricola altrimenti il file non verrà valutato.

Esercizio 1 - Descrizione Sistema

Un RV32IMAFD con Tck = T dispone di **tre** unità funzionali, A, M e D "multiciclo" capaci di eseguire le seguenti istruzioni su operandi in virgola mobile:

A: FADD (in 2T) M: FMUL (in 3T) D: FDIV (in 4T).

Si consideri il seguente frammento di codice e si faccia l'ipotesi che in T1 si abbia Fi=i per ogni valore di i compreso tra 0 e 31.

```
fdiv.s   f2, f9, f3
fmul.s   f1, f9, f3
fadd.s   f4, f9, f3
fdiv.s   f5, f1, f2
fadd.s   f5, f3, f1
fdiv.s   f4, f5, f3
```

1. Si stimi il numero di colpi di clock al di sotto del quale non è possibile scendere nell'esecuzione del codice assegnato, qualunque sia il numero di RS, CRB e stadi di Fetch e Decode disponibili, nell'ipotesi che esista una sola unità di tipo **A**, una sola di tipo **D** e una sola di tipo **M** (tre unità funzionali in tutto) e si motivi la risposta *(In questa stima non si tenga conto delle dipendenze nel codice) (**1 punto**)*

2. Si disegni il grafo delle dipendenze e si deduca il numero minimo di periodi di clock necessario a eseguire il codice assegnato tenendo conto solo delle dipendenze trovate ***(1 punto)***.

3. Si mostri la dinamica dell'esecuzione nel caso della CPU considerata nel punto 1 con 1 CDB, uno stadio di IF e uno di ID, e **2 RS** per ognuna delle tre unità funzionali (*si ipotizzi che, in caso di conflitti sul CDB, la fase di WB dell'unità D abbia la priorità sulle altre**) (3 punti).***

4. E' possibile ridurre il numero di periodi di clock necessario ad eseguire il codice assegnato rispetto al valore risultante dalla risposta al punto 3? Si punti al numero minore possibile di periodi di clock e al numero minimo di modifiche, supponendo di poter apportare solo le seguenti modifiche all'architettura:

- o aumento di RS da due a tre in una o più unità funzionali
- o raddoppio o triplicazione del CDB per poter eseguire due o tre fasi di WB per clock
- o aggiunta di una o più unità funzionali con due RS

In caso di raddoppio di una unità funzionale, si ipotizzi di fare lo scheduling a rotazione sulle due unità funzionali uguali.

Quali modifiche converrebbe apportare? Si motivi la risposta e si disegni la nuova dinamica di esecuzione. ***(3 punti)***.

5. Si disegni il film del registro f4 nel caso della dinamica di esecuzione di cui al punto 4. ***(2 punti)***.

Esercizio 2 - Descrizione Sistema

Un sistema multicore S è composto da 2 cores RV64GC dotati ciascuno di una cache L1 dati privata a 2 vie da 4KiB e linee da 64 bytes, gestita con stato MESI e con politica di scrittura Write Allocate in caso di miss.

Nel sistema S esegue un'applicazione che svolge il calcolo della somma degli elementi di due array x e y contenenti ciascuno 512 float a doppia precisione memorizzati a partire dall'indirizzo 0x0010 1000 e 0x0020 2000 rispettivamente. La memoria contiene anche un ulteriore array, composto da 512 float a doppia precisione e denominato z e memorizzato a partire dall'indirizzo 0x0030 3000.

Al fine di eseguire il calcolo del vettore z l'applicazione in esecuzione su S esegue in sequenza lo stesso blocco di codice sui due core. Prima il codice esegue sul core0 per il calcolo dei primi 256 elementi di z e poi esegue sul core1 per il calcolo dei successivi 256 elementi di z.

RISC-V assembly (ridotto)

BLOCCO B

```
00| .loop:
01|     fld     f0, 0(x10)   # f0 = x[i]
02|     fld     f1, 0(x11)   # f1 = y[i]
03|     fadd.d  f0, f0, f1
04|     fsd      f0, 0(x12)   # z[i] = f1
05|     addi    x10, x10, 8
06|     addi    x11, x11, 8
07|     addi    x12, x12, 8
08|     addi    x13, x13, -1
09|     bnez    x13, .loop
10| .return:
11|     ret
```

Il codice di fianco corrisponde al compilato del blocco di codice che implementa il calcolo della somma tra i due vettori.
In questo codice:

- x10 contiene l'indirizzo del valore x[i]
- x11 contiene l'indirizzo del valore y[i]
- x12 contiene l'indirizzo del valore z[i], dove il risultato del della somma degli elementi dei due vettori viene salvato.
- x13 tiene il conto delle iterazioni ancora da eseguire.
- f0, f1 sono registri floating-point temporanei.

Quesiti:

2.a) Facendo riferimento al sistema S, si completi la mappa delle memoria con i tre vettori x, y e z. Calcolare l'indirizzo del primo e ultimo elemento degli array x, y e z considerato da ognuno dei cores. ***(Punti.3)***

Si analizzi la dinamica della cache dati, e, tenendo ben presente che il sistema ha un solo caching agent, si risponda in modo preciso, schematico, conciso e tabellare ai seguenti quesiti: ***(14 Punti)***

2.b) Quali sono gli indici di set e linea, e i tag associati agli array x, y e z? Quante linee di cache occuperanno del loro insieme? Possono le porzioni dell'array x,y e z associate al core 0 essere contenute interamente nelle sua cache L1 privata?

2.c) Si consideri la dinamica della cache nel calcolo dell'elemento z[0] svolto dal core0 - disegnando lo stato MESI, il contenuto della cache e il valore del bit LRU dopo ogni operazione di accesso alla memoria (Load e Store).

2.d) Si mostri lo stato MESI, il contenuto e il valore del bit LRU della cache per il solo core 0, ignorando i calcoli svolti dal core1. Si riporti lo stato della prima e dell'ultima linee di cache per il core.
Si indichi il numero di miss, il numero di accessi, si calcoli la miss rate e si calcoli il numero di cicli di WB.

2.e) Come cambierebbero i cicli di WB e la miss rate se la politica di scrittura in caso di miss fosse Write Around?

2.f) Si mostri lo stato MESI, il contenuto e il valore del bit LRU della cache per entrambi i core al fine dell'esecuzione del codice sul core1. Facendo l'ipotesi che il core1 esegua dopo il termine dell'esecuzione sul core0. Si riporti lo stato della prima e dell'ultima linea di cache.