CALCOLATORI ELETTRONICI M

ARCHITETTURA DEI CALCOLATORI ELETTRONICI M

11/01/2022 – 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.

```
fadd.s  f1, f4, f8
fdiv.s  f3, f8, f4
fmul.s  f2, f1, f5
fmul.s  f1, f3, f4
fmul.s  f3, f2, f7
fadd.s  f1, f3, f11
```

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 CRB, 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 CRB, 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:

- aumento di RS da due a tre in una o più unità funzionali
- raddoppio o triplicazione del CRB per poter eseguire due o tre fasi di WB per clock
- 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 f3 nel caso della dinamica di esecuzione di cui al punto 4. ***(2 punti)***.

Esercizio 2 - Descrizione Sistema

Un sistema multicore S è composto da 4 cores RV64GC dotati ciascuno di una cache L1 dati privata a 2 vie da 8KiB complessivi e linee da 32 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 media degli elementi di un array x contenente 4096 float a singola precisione memorizzati a partire dall'indirizzo 0x010A 0000. La memoria contiene anche un ulteriore array, composto da 4 float a singola precisione e denominato y e memorizzato a partire dall'indirizzo 0x010B 0000, ed una variabile float a singola precisione z memorizzata all'indirizzo 0x010C 0000.

Al fine di eseguire il calcolo della media l'applicazione in esecuzione su S esegue in sequenza due blocchi di codice:

B0) Il primo (B0) viene eseguito in parallelo, su tutti e 4 i cores ma su 4 porzioni diverse dell'array x. Ciascuna istanza del blocco di codice svolgerà la sommatoria di una porzione da 1024 elementi del vettore x. Le accumulazioni parziali sono memorizzate negli elementi di y. L'accumulazione della porzione j-esima viene eseguita dal core j-esimo e memorizzata al termine del blocco in y[j] (fsw).

RISC-V assembly (ridotto)

BLOCCO B0

```
00| .loop:
01|       flw   f1, 0(x10)  # f1 = x[i]
02|       fadd.s f0, f0, f1
03|       addi  x10, x10, 4
04|       addi  x12, x12, -1
05|       bnez  x12, .loop
05| .return:
07|       fsw   f0, 0(x11)  # *y = f0
08|       ret
```

Il codice di fianco corrisponde al compilato del blocco di codice **B0** che implementa il calcolo dell'accumulazione parziale.
In questo codice:

- x10 contiene l'indirizzo del valore x[i]
- x11 contiene l'indirizzo contenuto in y[j], dove il risultato dell'accumulazione viene salvato.
- x12 tiene il conto delle iterazioni ancora da eseguire.
- f0 e f1 sono registri floating-point temporanei.

B1) Il secondo blocco di codice (B1) è seriale ed esegue solo sul core 3. La routine attende la terminazione di tutti i 4 tasks che compongono la routine (B1), ed effettua l'accumulazione delle somme parziali contenute negli elementi dell'array y. Quest'ultimo valore viene poi diviso per 4096 che corrisponde al numero di elementi.

RISC-V assembly (ridotto)

SEZIONE SERIALE (B1)

```
00| .loop:
01|       flw   f0, 0(x10)  # f0 = y[j]
02|       fadd.s f8, f8, f0
03|       addi   x10, x10, 4
04|       addi   x8, x8, -1
05|       bnez   x8, .loop
06|       fdiv.s  f8, f8, f9  # sum = sum/4096
07| .return:
08|       fsw    f8, 0(x11)   # *z = f0
09|       ret
```

Il codice di fianco corrisponde al compilato del blocco di codice **B1** che quella porzione seriale di codice che esegue su un solo core, colleziona i risultati delle computazioni eseguite in parallelo sui quattro cores e li somma e ne fa la media per ottenere il risultato finale.
In questo codice:

- x10 contiene l'indirizzo del valore y[j].
- x11 contiene l'indirizzo contenuto in z, dove il risultato media viene salvato.
- x8 tiene il conto delle iterazioni ancora da eseguire.
- f0 e f8 sono registri floating-point temporanei.
- f9 contiene il numero 4096, pari al numero totale degli elementi

Quesiti:

2.a) Facendo riferimento al sistema S, si completi la mappa delle memoria con i due vettori x, y e la variabile z. Calcolare l'indirizzo del primo e ultimo elemento degli array x e y 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 4 porzioni dell'array x essere contenute interamente nelle cache dei 4 cores?

2.c) Si consideri la dinamica della cache nel calcolo di y[0] - blocco B0 - svolto solo 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) per la prima e l'ultima iterazione del loop. Si indichi il numero di miss, il numero di accessi, si calcoli la miss rate. Quant'è l'aritmetic intensity associata al blocco di codice B0?

2.d) Si determini il numero di cicli di WB nonché lo stato MESI, il contenuto e il valore del bit LRU della cache al termine dell'esecuzione parallela del blocco B0. Al fine della risoluzione del quesito si assuma che l'ultima istruzione del blocco parallelo (lsw) sia eseguita da tutti i processori con il seguente ordine: Core-0, Core-1, Core-2, Core-3. Si riporti lo stato delle prime due ed ultime due linee di cache. Se non incluse si riportino le eventuali linee di cache che contengono gli elementi dell'array y.

2.e) Come cambierebbe lo stato della cache 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 il core 3 alla fine dell'esecuzione della porzione seriale del blocco di codice. Si riporti lo stato delle prime due ed ultime due linee di cache. Se non incluse si riportino le eventuali linee di cache che contengono gli elementi dell'array y e della variabile z.