Architettura degli Elaboratori Mod. 2
Fac-simile d'esame - Soluzioni

Filippo Bergamasco

02/05/2022

Soluzione 1

Notiamo che l'indirizzo è composto da 8 cifre esadecimali, il che, suggerisce che la dimensione totale dell'indirizzo è di 32 bit.

Calcoliamo il numero di bit necessari per l'OFFSET e ricordiamoci che l'indirizzamento dei dati avviene al byte. Dunque, sapendo che il block size è di 4 words ossia 128 bit, ricaviamo il corrispondente numero di Byte: 128 bit = 16 Byte = 2^4 Byte. Di conseguenza ci serviranno 4 bit di OFFSET (ultima cifra esadecimale dell'indirizzo).

1. Nel primo caso il TAG è composto da 16 bit (4 cifre esadecimali).

   Calcoliamo il numero di bit necessari per l'INDEX. Sapendo che l'indirizzo è a 32 bit.

   $$32 - 16 - 4 = 12$$

   Con 12 bit di INDEX (3 cifre esadecimali dell'indirizzo) abbiamo

   $$\#linee = 2^{12}$$

   Ricaviamo ora il numero di blocchi nella cache, dividendo il totale dei dati per il block size, ossia:

   $$\#blocks = \frac{64 * 2^{10}}{16} = \frac{2^{16}}{2^4} = 2^{12}$$

   Il livello di associativita' della cache è :

   $$assoc = \frac{\#blocks}{\#linee} = \frac{2^{12}}{2^{12}} = 1$$

   La cache è ad accesso diretto.

   L'indirizzo è pertanto suddiviso come segue:

	TAG	INDEX	OFFSET
0x	1337	001	0

2. Nel secondo caso il TAG è composto da 20 bit (5 cifre esadecimali dell'indirizzo) e l'offset non cambia.

   Il numero di bit dell'INDEX sono di conseguenza $32-20-4=8$ esprimibili in 2 cifre esadecimali. Con 8 bit di INDEX abbiamo:

   $$\#linee = 2^8$$

   Il numero di blocchi rimane invariato ma l'associatività cambia:

   $$assoc = \frac{\#blocks}{\#linee} = \frac{2^{12}}{2^8} = 2^4$$

   La cache è una 16-way associative.

   L'indirizzo è pertanto suddiviso come segue:

   ```
        TAG     INDEX    OFFSET
   0x  13370      01      0
   ```

Soluzione 2

(a) Le dipendenze RAW sono le seguenti:

- 1 -> 2 sul register file $9
- 2 -> 3 sul register file $9
- 1 -> 3 sul register file $9
- 4 -> 5 sul register file $7
- 5 -> 6 sul register file $4

(b) Il diagramma temporale è il seguente:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
lw	IF	ID	EXE	MEM	WB													
addi		IF	ID	<ID>	<ID>	EXE	MEM	WB										
sw			IF	<IF>	<IF>	ID	<ID>	<ID>	EXE	MEM	WB							
addi						IF	<IF>	<IF>	ID	EXE	MEM	WB						
sub									IF	ID	<ID>	<ID>	EXE	MEM	WB			
bne										IF	<IF>	<IF>	ID	<ID>	<ID>	EXE	MEM	WB

Figure 1: Diagramma Temporale

(c) Considerando che il numero di cicli necessari per svolgere un ciclo di loop è: $5+4+4+4+4+3=28$ e la funzione di loop viene eseguita 8 volte consecutivamente, il CPI è dato da:

$$CPI = \frac{\#cicli}{\#istruzioni} = \frac{28 * 8}{6 * 8} = 4$$

Il diagramma temporale, nel caso il processore sia in grado di eseguire forwarding, è il seguente:

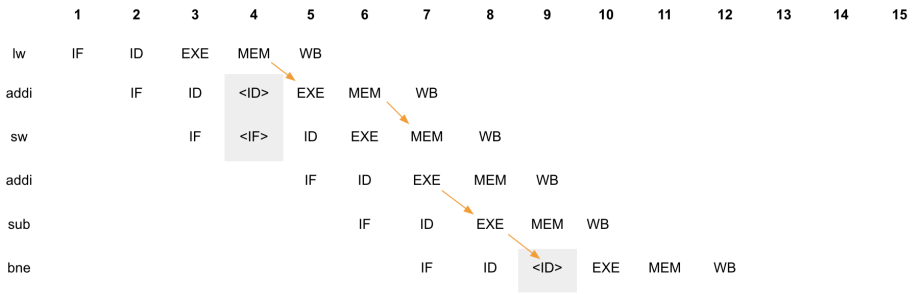

Figure 2: Diagramma Temporale, forwarding in arancione

Soluzione 3

```
    adr x0, n       // x0 = &n
    ldr w0, [x0]    // w0 = Mem[x0]. w0 lo usiamo per memorizzare
                    //               la variabile n

    mov w1, #0      // w1 lo usiamo per memorizzare il valore di r
    mov w2, #0      // w2 lo usiamo per il contatore i
    cmp w2, w0
    b.ge exit_for
init_for:

    ands w3, w2, #1 // w3 = w2 & 1
    b.ne skip1
    add w1, w1, w2
skip1:
    add w2, w2, #1  // incremento contatore
    cmp w2, w0
    b.lt init_for

exit_for:
    adr x0, r       // x0=&r
    str w1, [x0]    // Mem[x0] = w1
```

Soluzione 4

Una possibile soluzione è la seguente:

```
    .global palindroma
palindroma:

    mov x1, x0
    ldrb w2, [x0]
    cmp w2, #0
    b.eq return1

loop1:
    add x1, x1, #1
    ldrb w2, [x1]
    cmp w2, #0
    b.ne loop1
    add x1, x1, #-1

loop2:
    ldrb w2, [x1]
    ldrb w3, [x0]
    cmp w2, w3
    b.ne return0
    add x0,x0,#1
    add x1,x1,#-1
    cmp x0, x1
    b.le loop2

return1:
    mov x0, #1
    ret
return0:
    mov x0, #0
    ret
```

Risposte alle domande di teoria

1. Un esempio di località spaziale è dato dall'accesso sequenziale agli elementi di un array. Un esempio di località temporale è invece il contatore di un ciclo. Aumentare la dimensione del blocco permette di sfruttare meglio la località spaziale. A parità di dimensione della cache, aumentare il livello di associatività ci permette di sfruttare la località temporale perché riduce i conflitti e quindi la probabilità che un dato (che potrebbe servirci in futuro) venga rimpiazzato da un altro dato in conflitto sullo stesso blocco.
2. Nei bus sincroni vi è una linea di clock condivisa tra gli elementi connessi

al bus. Il protocollo di comunicazione sfrutta questo clock per arbitrare gli accessi al bus. E' generalmente più veloce e performante ma richiede bus corti. Nei bus asincroni, invece, il clock non è condiviso tra gli elementi del bus. Sono richieste pertanto linee aggiuntive e protocolli di handshaking per sincronizzare le comunicazioni. Permette la comunicazione tra periferiche con velocità diverse ed è usato nei BUS standard come USB, SATA, PCIe, etc.

3. Le eccezioni in una CPU con pipeline costituiscono una forma di Control Hazard. Comportano le seguenti problematiche:

- Occorre annullare le istruzioni già entrate nella pipeline prima che si verificasse l'eccezione
- Occorre modificare lo stadio EXE di modo che possa mettere in stallo la CPU (le eccezioni aritmetiche vengono generate in questo stadio)
- Eccezioni multiple possono essere generate nello stesso ciclo di clock da stadi diversi. E' necessario quindi un sistema di priorità nella gestione delle eccezioni