Architettura degli Elaboratori (modulo II e B) (Compito 21 Gennaio 2010)

Usare un foglio separato per rispondere alle domande seguenti, specificando nell'intestazione: **Titolo del corso** (*Architettura degli Elaboratori – modulo II*, oppure *Architettura degli Elaboratori B*, oppure *Integrazione*), **Data esame**, **Cognome e Nome**, **Matricola**.

Esercizio 1

Considerare una memoria virtuale con indirizzo virtuale di 32 b. Supporre che ogni pagina contenga 2 KB, che l'indirizzo fisico sia di 34 b, e che gli ingressi della page table abbiano un valid bit come primo bit.

- Calcolare il numero di ingressi della page table e la dimensione in Byte.
- Considerando la seguente porzione della page table (il numero a sinistra indica il numero di ingresso), individuare a quali pagine fisiche si riferiscono primi tre ingressi. Tradurre, usando tali informazioni, l'indirizzo virtuale `0x00000AFF` nel corrispondente indirizzo fisico.

0	0xA42000
1	0x855000
2	0x1FF010

Sempre riferendoci allo stesso sottosistema di memoria, si consideri l'esistenza di una cache, associativa a 2 vie, la cui parte dati è di 1 MB.

- Se l'Index è uguale a 13 b, qual è la dimensione di ciascun blocco? Qual è la dimensione dellaa Tag

Soluzione

Le pagine hanno dimensione 2048 B = 2^{11} B. Sono quindi necessari 11 bit di offset. Da qui si ottiene che i bit usati per indirizzare le pagine virtuali (numero di pagina virtuale) sono 32-11 = 21 e che ci sono 2^{21} ingressi nella page table. Ogni ingresso della page table deve contenere 34-11 = 23 b per la pagina fisico, più il bit di validità, quindi un totale di 24 b = 3 B. La dimensione della page table è quindi $2^{21} \cdot 3$ B = 6 MB.

Per rispondere al secondo punto, è sufficiente controllare se i 3 ingressi sono validi (primo bit) e, in tale caso, eliminare il primo bit per ottenere la pagina corrispondente. Otteniamo che solo i primi due sono validi e corrispondono quindi alle pagine fisiche `0x242000` e `0x055000`. Il terzo ingresso ha invece bit di validità uguale a zero e non corrisponde di conseguenza a nessun indirizzo fisico.
Per tradurre `0x00000AFF` in indirizzo fisico dobbiamo estrarre i primi 21 bit ed utilizzarli per accedere alla page table. In binario, tale numero risulta essere `0000 0000 0000 0000 0000 1010 1111 1111`. Otteniamo quindi che la pagina virtuale è la seconda (ingresso numero 1) che ha come corrispondente pagina fisica `0x055000`, ovvero `000 0101 0101 0000 0000 0000`. Per ottenere la traduzione è sufficiente "appendere" i bit di offset alla pagina fisica ottenendo: `000010101010000000000 01011111111` cioè `00 0010 1010 1000 0000 0010 1111 1111`. Tale indirizzo può essere scritto in esadecimale come: `0x02A802FF`.

La dimensione di ogni entry della cache è $\frac{Size\ cache}{2\ vie \cdot 2^{Index}} = \frac{2^{20}}{2^{14}} = 2^6 = 64$ B.
Quindi $Offset$ è 6 b = $\log 64$. Tag = 34 - $Offset$ - $Index$ = $34 - 6 - 13 = 15$ b.

Esercizio 2

Si consideri il seguente programma:

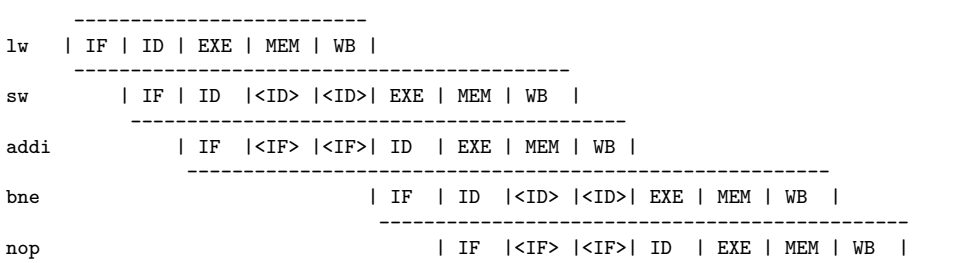
```
loop:   lw   $3, a($4)
        sw   $3, b($4)
        addi $4, $4, 4
        bne  $4, $6, loop
```

dove `$4` è inizializzato a 0 e `$6` a 16000.

- Calcolare il CPI nel caso di esecuzione sulla pipeline a 5 stadi vista a lezione, senza tecniche di forwarding. Supporre che il register file consenta di scrivere nella prima parte del clock e di leggere nella seconda parte dello stesso ciclo si clock, e che successivamente alla `bne` venga inserita una `nop` (la `nop` è funzionale alla tecnica del delayed branch).
- Calcolare il miglioramento della performance (speed-up) nel caso in cui vengano introdotte tecniche di forwarding (riorganizzare il codice se possibile).

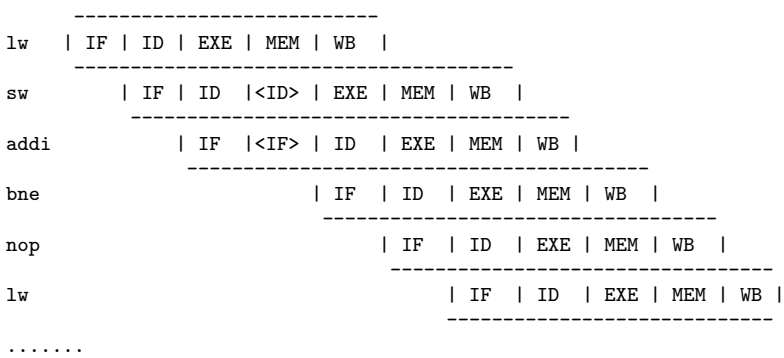
Soluzione

Abbiamo una dipendenza tra la `lw` e la successiva `sw` sul registro `$3`. Inoltre abbiamo una dipendenza tra la `addi` e la successiva `bne` e su `$4`. Otteniamo quindi la seguente esecuzione:



Ogni loop richiede quindi 9 cicli di clock. Poichè vengono eseguiti 16000 / 4 = 4000 volte, otteniamo un numero totale di cicli di $9 \cdot 4000 = 36000$ (i 3 cicli di svuotamento della diventano irrilevanti rispetto ai 36000 necessari per il loop). Poiché le istruzioni sono 4 * 4000 = 16000 otteniamo CPI = 36000 / 16000 = 2.25.

Purtroppo non possiamo ottimizzare il codice rischedulando le istruzioni. Il forwarding consente invece di evitare tutti gli stalli tranne quello dopo la `lw`:



In questo caso un loop richiede 6 cicli di clock. Non considerando il caricamento otteniamo un CPI = 6/4 = 1.5. Lo speedup si calcola semplicemente facendo il rapporto tra i CPI (IC e Freq sono invariati), ovvero $Speedup = 2.25/1.5 = 1.5$
Possiamo ottimizzare ulteriormente con il loop unrolling.