Architettura degli Elaboratori (mod. II/B, 2^o Compitino) (1 Giugno 2011)

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

Esercizio 1

Sia dato un sistema di memoria virtuale paginata, con indirizzo virtuale di 32 b e fisico di 32 b, con pagine di 2048 B. Sia la Page Table e sia la TLB contengono, per ogni ingresso, 3 ulteriori bit (Valid, Dirty, Reference).
Calcolare la dimensione della Page Table.
Se la TLB ha 128 ingressi, calcolare il numero di bit necessari per memorizzare ciascuna entry della TLB (considerando i 3 bit aggiuntivi) rispetto alle organizzazioni: completamente associativa, diretta, e associativa a 4 vie.

Soluzione

No. di bit per indirizzare una pagina: $\log 2048 = log 2^{11} = 11$
#Numero di Pagina virtuale: 32 - 11 = 21 b.
#Numero di Pagina fisica: 32 - 11 = 21 b.
Dimensione di ciascun ingresso della Page Table: #Numero di Pagina fisica + 3 = 21+3 = 24 b = 3 B.
Dimensione della Paga Table: $2^{\#\text{Numero di Pagina virtuale}} * 3\text{ B} = 2^{21} * 3 = 6$ MB.

TLB diretta: INDEX = $log 128 = 7$ b, per cui TAG = #Numero di Pagina virtuale - INDEX = 21-7 = 14 b. Quindi ogni entry è composta da TAG + #Numero di Pagina fisica + 3 = 14+21+3=38 b.
TLB 4-way: INDEX = $\log 128/4 = 5$ b, per cui TAG = #Numero di Pagina virtuale - INDEX = 21-5=16 b. Quindi ogni entry è composta da TAG + #Numero di Pagina fisica + 3 = 16+21+3 = 40 b.
TLB compl. associativa: Ogni entry è composta da #Numero di Pagina virtuale + #Numero di Pagina fisica + 3 = 21+21+3=45.

Esercizio 2

Considerare il seguente loop, eseguito dalla CPU *multi-ciclo*[1] vista a lezione.

```
      addi $s0, $a0, 1024*4  # $a0 è l'indirizzo iniziale di un array di word
loop: lw   $t1, 0($a0)
      add  $t1, $t1, $a1
      sw   $t1, 0($a0)
      addi $a0, $a0, 4
      bne  $a0, $s0, loop
```

- Assumendo che dati e codice siano presenti in cache all'inizio dell'esecuzione, calcolare IC e CPI medio. Cosa viene calcolato al 18^o ciclo di clock?
- Se tutto l'array, ovvero i dati acceduti dalle lw/sw, fosse presente in memoria ma non in cache, ricalcolare il numero di cicli totali e il CPI, considerando che ogni blocco della cache è grande 64 B, mentre il cache penalty è di 40 cicli.
- Considerare il codice precedente come il corpo di una procedura con due parametri in ingresso, passati secondo le solite convenzioni sui registri. Quali sono questi due parametri, e cosa rappresentano nel codice stesso? Completare la funzione con gli eventuali salvataggi dei registri sullo stack secondo le convenzioni, e il ritorno.

Soluzione

Il loop interno viene eseguito per 1024 volte.
$IC = 1 + 5 * 1024 = 5121$
$no\_cicli = 4 + 1024 * (4 + 5 + 4 + 4 + 3) = 4 + 1024 * 20 = 4 + 20480 = 20484$.
$CPI = cicli/IC = 20484/(1 + 5 * 1024) = 20484/5121 = 4$.
Al 18^o ciclo viene calcolato il primo passo dell'istruzione `addi`. Ovvero viene letta l'istruzione e viene incrementato il PC di 4 unità.
Se consideriamo la cache, rispetto alle `lw` abbiamo 1 miss e 15 hit, poiché ogni blocco della cache è di 16 word. In corrispondenza delle `sw` abbiamo tutti hit, poiché la word acceduta è stata caricata precedentemente dalla `lw`. I miss sono quindi $\frac{cicli\ del\ loop}{16} = 1024/16 = 64$. Quindi i cicli spesi sono:
$no\_cicli = 20484 + 64 * 40 = 23044$.
$CPI = cicli/IC = 23044/(1 + 5 * 1024) = 23044/5121 = 4,5$.

[1] Tutte le istruzioni impiegano 4 cicli, eccetto le `lw` (5 cicli in caso di hit) e le `beq/bne` (3 cicli).

Infine, i due parametri della funzione sono `$a0` e `$a1`, che corrispondono rispettivamente all'indirizzo iniziale di un array di word di 1024 elementi, e al valore da sommare a tutti gli interi dell'array.
Il codice con i salvataggi dei registri è:

```
 foo: addi $sp, $sp, -4
      sw   $s0, 0($sp)       # $s0 callee-save
      addi $s0, $a0, 1024*4  # $a0 è l'indirizzo iniziale di un array di word
loop: lw   $t1, 0($a0)
      add  $t1, $t1, $a1
      sw   $t1, 0($a0)
      addi $a0, $a0, 4
      bne  $a0, $s0, loop
      lw   $s0, 0($sp)
      addi $sp, $sp, 4
      jr   $31
```