Architettura degli Elaboratori 2012-13

Esercizi e quesiti - Parte 3

I seguenti esercizi e quesiti sono proposti per verificare la preparazione e comprensione della materia. La loro validità è tanto maggiore quanto più sono svolti di pari passo con il programma del corso e in modo critico. È fondamentale che le risposte siano date in forma chiara e rigorosa, usando i concetti e la terminologia del corso, evitando ambiguità e spiegazioni fuori tema.

Si raccomanda, agli studenti interessati, di chiarire ogni dubbio e di verificare le risposte con il docente.

1) Esercizi su compilazione, ottimizzazione e valutazione delle prestazioni

Riprendere gli esercizi della *Raccolta 2* (esercizi 4, 5, 6, 7) e risolverli per architetture con:

- **a)** Cache
- **b)** CPU Pipeline scalare
- **c)** CPU Pipeline superscalare
- **d)** CPU con Simultaneous Multithreading

Analogamente alle parti precedenti del corso, qualunque altro esempio di algoritmo noto è valido per questa classe di esercizi.

2) Si consideri un elaboratore con memoria cache. Pronunciarsi sulla verità o meno delle seguenti affermazioni, spiegando la risposta:

a) la dimensione ottimale del blocco è indipendente dal programma;

b) quanto più grande è la dimensione del blocco, tanto migliori sono le prestazioni dei programmi;

c) l'insieme di lavoro di un processo è costituito dal suo codice e dalle strutture dati utilizzate;

d) il riuso dei dati di un programma può essere riconosciuto direttamente dall'unità cache;

e) la progettazione dell'unità cache è diversa a seconda che il sistema preveda o non preveda le ottimizzazioni legate al riuso dei dati.

3) Si consideri una gerarchia di memoria memoria principale – memoria cache su domanda, senza ulteriori livelli di cache e con memoria principale avente banda di elaborazione uguale a una parola per tempo di accesso.

Dimostrare che, in tali ipotesi, programmi che godano solo della proprietà di località hanno un tempo di completamento maggiore o uguale rispetto al caso in cui l'architettura non preveda cache.

Applicare il risultato ottenuto al caso in cui il livello superiore sia la memoria di I/O.

4) Si consideri la seguente computazione operante sugli array di N interi A e B, con F funzione nota:

$$\forall\ i = 0\ ..\ N-1 : B[i] = F(A[i])$$

Spiegare cosa c'è da aspettarsi circa le differenze sul tempo di completamento implementando la computazione sulle seguenti architetture: S1 basata su una unità di elaborazione dedicata, S2 basata

su una CPU convenzionale, S3 basata su una CPU pipeline. Tutte le architetture hanno lo stesso ciclo di clock e la stessa gerarchia memoria principale – cache. A e B sono allocati in memoria.

5) Per una architettura CPU pipeline, spiegare la struttura, il funzionamento, ed eventuali ottimizzazioni dinamiche, dei sottosistemi

a) Unità Istruzioni,

b) Unità Memoria Dati,

distinguendo esplicitamente ogni classe di istruzione. Sulla base di tali descrizioni, determinare il tempo di servizio ideale delle unità suddette.

Analogamente per una CPU superscalare a due vie.

6) Si discutano le caratteristiche dell'Unità Esecutiva Master per una Unità Esecutiva contenente tre unità pipeline dedicate, rispettivamente, alla moltiplicazione e divisione in virgola fissa (4 stadi), addizione e sottrazione (4 stadi) e moltiplicazione e divisione (8 stadi) in virgola mobile. In particolare:

- le interazioni con Unità Istruzioni e con Memoria Dati,
- la gestione dei registri generali e in virgola mobile,
- l'implementazione delle dipendenze sui dati.

Sulla base di tale descrizione, determinare il tempo di servizio ideale dell'Unità Esecutiva Master.

7) Altre computazioni per esercizi tipo 1):

a)
```
int A[N][N];
∀ i = 0 .. N – 1:
        ∀ j = 0 .. N – 1:
                if A[i][j] ≠ A[h][j]
                        then A[i][j] = A[i][j] – A[h][j]
```

b)
```
int A[N], B[N], C[N];
∀ i = 0 .. N – 1:
        { C[i] = 0;
          ∀ j = 0 .. N – 1:
                C[i] = if (B[j] < 0)
                                then C[i] – B[j] + A[i] / 2048
                                else C[i] + B[j] – A[i] % 4096
        }
```

c)
```
int A[N];
∀ i = 0 .. N – 1:
        if A[i % C] = 0   then A[i] = 0   else A[i] = A[i] ∗ A[i] + 1
```

8) Si consideri il seguente programma D-RISC:

```
        LOAD   RA, 0, Ra
        LOAD   RB, 0, Rb
        LOAD   RC, 0, Rc
        LOAD   RD, 0, Rd
        IF <   Rc, Rd, THEN
        DIV    Ra, Rc, Rx
        INCR   Rx
        GOTO   CONT
THEN:   MUL    Ra, Rb, Ra
        DIV    Rc, Rd, Rc
        ADD    Ra, Rc, Rx
CONT:   STORE  RX, 0, Rx
        END
```

a) Con riferimento ad una CPU pipeline scalare con Unità Esecutiva parallela, avente unità pipeline in virgola fissa a quattro stadi, fornirne una versione ottimizzata e valutarne il tempo di completamento in funzione del ciclo di clock e della probabilità che si verifichi il predicato dell'istruzione IF.

b) Analogamente per una CPU superscalare a due vie.

9) Si consideri il problema della ricerca di un elemento in una lista realizzata mediante la tecnica dei doppi puntatori. Ogni elemento della lista è formato da una parola VAL che contiene il valore dell'elemento seguita dai puntatori NEXT e PREV rispettivamente all'elemento successivo e precedente nella lista.

Si compili in assembler D-RISC una procedura, che accetta due parametri in ingresso (*l'indirizzo della struttura dati contenente il puntatore* al primo elemento della lista, e il valore da ricercare) e restituisce un intero (0 non trovato, 1 trovato) dopo aver rimosso (se presente) il valore cercato dalla lista. Si assume che l'elemento ricercato, se presente, sia presente in un sola posizione della lista, non in quella iniziale, e che il passaggio dei parametri avvenga mediante registri.

Si fornisca il tempo di completamento per la procedura in funzione della lunghezza della lista (N), assumendo che nella metà dei casi la ricerca avvenga per un valore non presente nella lista.

Distinguere due casi:

a) lista privata,

b) lista condivisa.

L'architettura della CPU è pipeline. La cache dati primaria è completamente associativa, su domanda, di capacità 32K parole e blocchi di 8 parole. La cache secondaria è on-chip e contiene l'intera lista.

10) Si consideri la computazione corrispondente al seguente algoritmo (con *M* potenza di 2 e >> 1):

int A[M];

$\forall$ *i = 0 .. M – 1:*

$\forall$ *j = i .. M – 1:*

A[i] = if pari(A[i] + A[j]) then A[i] + A[j] else A[i] – A[j]

Implementarla mediante una unità di elaborazione U connessa ad una gerarchia di memoria memoria principale – cache. La cache, residente sullo stesso chip di U, è associativa, su domanda, Write-Trough, con capacità $\gamma = M/4$ e blocchi di σ parole. La memoria principale è interallacciata con $m = 8$ moduli e ciclo di clock uguale a $p\tau$ (con $p >> 1$ e $p << M$, e τ ciclo di clock di U). I collegamenti interchip hanno latenza di trasmissione uguale a τ. U riceve da una unità U_M l'indirizzo base di A e restituisce alla stessa U_M una segnalazione di fine elaborazione.

Scrivere il microprogramma di U e valutare il tempo medio di elaborazione in funzione di M, σ, p e t_p (una ALU ha ritardo $5t_p$). Spiegare le principali scelte di progetto e la valutazione delle prestazioni.

11) Si consideri una unità cache primaria C con metodo di indirizzamento diretto e scritture con il metodo write-through. Il chip CPU contiene anche una cache secondaria. La MMU non ha interfacce verso la memoria esterna.

Scrivere il microprogramma dell'unità C, esclusa la fase di gestione del fault di blocco, e, sulla base di tale microprogramma, mostrarne la parte Operativa e valutarne il ciclo di clock e il tempo di servizio utilizzando simboli opportuni.

Completare il microprogramma dell'unità C per quanto riguarda anche la gestione del fault di blocco, e, sulla base di tale microprogramma, valutare il tempo di trasferimento di un blocco dalla cache secondaria alla cache primaria.

12) Una unità U è interfacciata con due unità P_{in} e P_{out} e con un sottosistema di memoria M di capacità 2G parole.

U riceve da P_{in} una parola X da 32 bit, la considera divisa in due campi (i 20 bit più significativi nel campo A e i 12 bit meno significativi nel campo B) ed invia all'unità P_{out} la parola Y ottenuta sostituendo in X A con il valore TAB[A].

La tabella TAB è contenuta interamente nella memoria M a partire da un indirizzo contenuto nel registro INDTAB interno ad U. Per velocizzare il calcolo, U contiene una memoria C da 64 posizioni organizzata come una cache ad accesso diretto con $\sigma = 1$. Il calcolo di TAB[A] avviene utilizzando C: nel caso in cui C non contenga l'entry di TAB relativa ad A, U provvede ad aggiornare l'informazione in C accedendo alla tabella TAB in M prima di inviare Y a P_{out}.

Sia $4t_p$ il tempo di accesso di C e $20t_p$ il tempo di accesso a M, sullo stesso chip di U, e 5tp il tempo di stabilizzazione della ALU. Si fornisca il microprogramma dell'unità U avendo cura di ottimizzare il tempo necessario alla traduzione di X in Y e si determinino T e τ.