Università degli Studi di Milano - Corso Architettura degli elaboratori e delle reti – Turno 1 - Prof. Borghese

Cognome e nome dello studente:

Matricola: ***Anno di corso e turno:***

A.A. 2002-2003 – Prova d'esame del 26 settembre 2003

1. Data la seguente funzione: y = a + bc + a + d + bcd:
   A) sintetizzare il circuito che implementa la funzione e definire il cammino critico [2],
   B) scrivere la funzione nella prima forma canonica (somma di prodotti) [1].

2. Disegnare un sommatore su 4 bit, con anticipazione di riporto [3].

3. Disegnare la porta di scrittura di un Register File costituito da 32 registri, ciascuno di 4 bit [3].

4. Tradurre in linguaggio Assembly il seguente segmento di codice [3]:

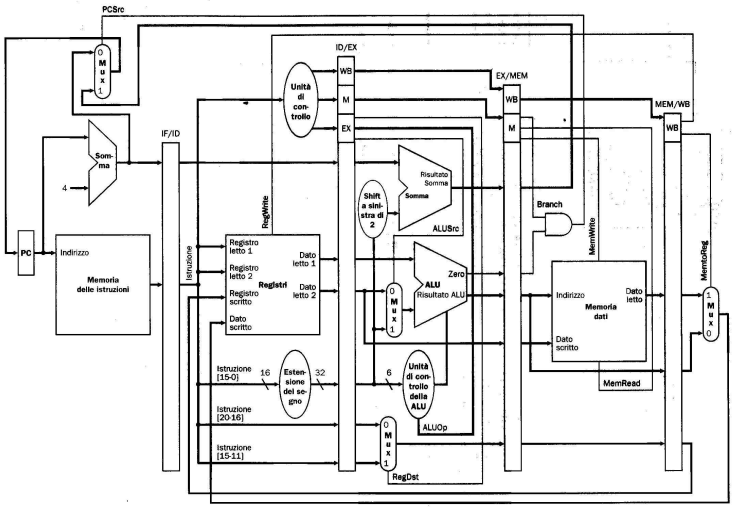
```
t0 = 2;
for (i=0; i<s1; i++1
{           t1 = t1 + 1;
            s2 = somma(s3, t2);
}
s4 = t0 + s2;
```

Dove le variabili t vengono associate ai registri $t, e le variabili s ai registri $s.

Facendo riferimento alla CPU con Pipe-line, qui a fianco, disegnare le modifiche necessarie per eseguire correttamente il segmento di codice precedente [6].

Definire cos'è uno stallo di una pipe-line [2].

5. Definire una CPU super-scalare e darne uno schema a blocchi [4].



6 Si consideri una macchina degli anni 80. La MISS_PENALTY è di 6 cicli di clock, mentre tutte le istruzioni impiegano 4.5 cicli di clock se si ignorano i MISS (stalli della memoria). Ipotizzando un MISS_RATE del 9% e che vi siano in media 2 riferimenti alla memoria per ogni istruzione.
Qual è l'impatto sulle prestazioni quando viene inserita la cache reale rispetto ad una cache ideale?
Qual è l'impatto sulle prestazioni tra il caso di cache reale e senza inserimento della cache?. [3].

7. Definire il contenuto del registro ALUOut nello schema di CPU multi-ciclo al termine del 2° stadio di esecuzione (decodifica) dell'istruzione: add $s0, $t0, $t1, sapendo che *nel processore MIPS i registri:* $zero, $at, $s0, $s1, $s2, $s3, $s4, $t0, $t1 corrispondono rispettivamente ai registri $0, $1, $16, $17, $18, $19, $20, $8, $9 e che l'istruzione **add** ha il seguente formato: 0x0 rs rt td 0x0 0x20 [3].

8. Gestione di un link in modalità circuit switching [1].

9. Disengare lo schema di interfacciamento delle periferiche sul bus, nel caso di arbitraggio del bus in daisy chain e con arbitro centrale con priorità [3].