Esercitazione di ricapitolazione – parte II

Potete trovare molti esempi di programmi o spezzoni di programma Assembly e Macchine a Stati Finiti nei temi di esame passati. Altri ne potete trovare nelle slide e nel materiale delle esercitazioni. Seguono alcuni esercizi di ricapitolazione.

I. Definire che cos'è una ISA. [1].

II. Definire i formati delle istruzioni e darne un esempio [4].

III. Definire i tipi di istruzioni messi a disposizione dall'ISA del MIPS R3000 [2].

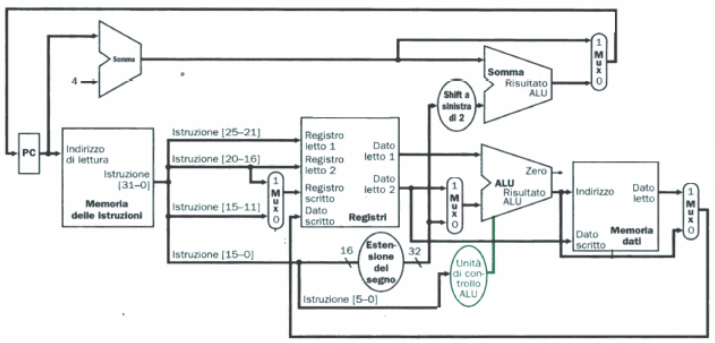
IV. Cosa succede in un'operazione di push o di pop? Quale parte dell'architettura interessa? [1]

V. Modalità di accesso ai dati nella memoria principale (RAM) nelle architetture MIPS [2].

VI. Qual è lo spazio indirizzabile della memoria da un'istruzione di load/store?

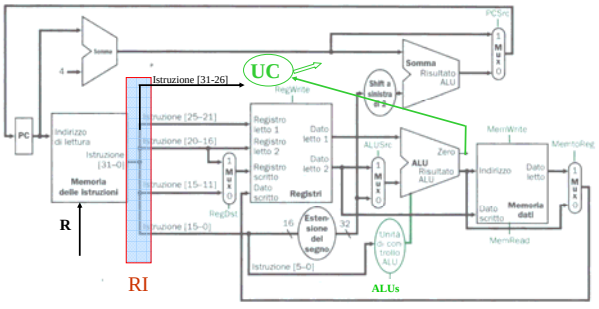
1. Descrivere il ciclo di esecuzione di un'istruzione. Di quali informazioni abbiamo bisogno in ciascuna fase?

2. Quali sono i segnali di controllo necessari per fare funzionare questa CPU? Quanti cicli di clock sono necessari per eseguire un'istruzione? Quanto valgono i segnali di controllo per eseguire l'istruzione lw $s0, 40($s1)? Quali sono le unità funzionali principali associate alle varie fasi di esecuzione? Indicare sul grafico la quantità che viene calcolata dalle varie unità funzionali.



3. Progettare un controllore a 2 livelli della ALU, sapendo che le operazioni consentite sono: addi, andi, ori, add, sub, and, or, lw, sw, beq. Il primo livello riceverà in input solamente il codice operativo, il secondo livello riceverà in input l'uscita del primo livello ed il contenuto del campo funct.

4. Seguire il data_path ed il control_path per i diversi tipi di istruzioni nella CPU a ciclo singolo riportata qui a fianco. Quali segnali di controllo **non** occorre specificare per un'istruzione di lw? E per un'istruzione di add? E per un'istruzione di addi? Per un'istruzione di beq?



5. Specificare quali sono gli operandi su cui agiscono le 3 ALU e da dove provengono. Cosa contiene l'ingresso 0 del Mux del secondo operando della ALU quando sto eseguendo l'istruzione: add $t0, $t1, $t2, sapendo che $t0 = $7 e che Funct = 0x20. Cosa contiene l'uscita del Mux all'ingresso della porta di scrittura del register file nell'istruzione beq $t0, $t1? E nell'istruzione sw $t0, 0($t1)? E nell'istruzione: add $t0, $t1, $t2?

6. Quando conviene una CPU con pipeline rispetto ad una CPU a ciclo singolo?

7. Come viene gestito un salto dalla CPU? Disegnare il circuito che gestisce i salti.

8. Scrivere il codice binario generato dalla seguente situazione. Abbiamo tre procedure: ProcA, ProcB, ProcC. Il segmento codice delle tre procedure è di 1Kbyte ciascuno il segmento dati rispettivamente di 1Kbyte, 2Kbyte, 4Kbyte.
Le prime cinque istruzioni di ProcA sono:
0: lw $t0, 24($gp)
4: beq $t0, $t1, Loop:
8: j Fine:
Loop: 12 add $t0, $t2, $t3
Fine: 16 sub $t0, $t1, $t2
Le prime cinque istruzioni di ProcB sono:
0: lw $t0, 4($gp)
4: bne $t0, $t1, Loop:
8: j Fine:
Loop: 12 add $t0, $t2, $t3
Fine: 16 sub $t0, $t1, $t2
Le prime cinque istruzioni di ProcC sono:
0: lw $t0, 4($gp)
4: bne $t0, $t1, Loop:
8: j Fine:
Loop: 12 sw $t0, 48($t1)
Fine: 16 sub $t0, $t1, $t2
Specificare la Symbol Allocation Table risultante dalla compilazione.

9. Cosa è una CPU con pipe-line? Una pipe-line consente l'esecuzione più veloce di un'istruzione rispetto ad una CPU a singolo ciclo? Di quanto aumenta la velocità di esecuzione in una CPU con pipeline? Una CPU con pipeline richiede più o meno unità funzionali di una CPU a ciclo singolo? Motivare le risposte.

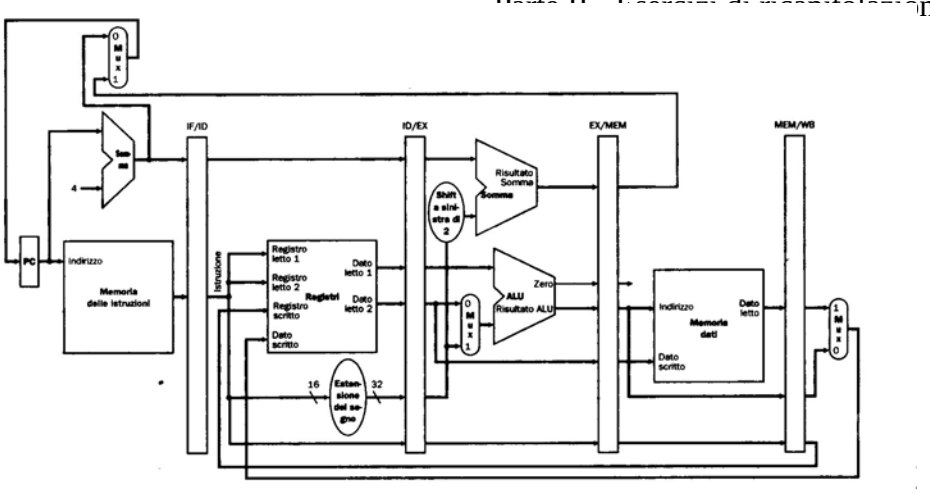
10. Modificare la CPU di cui sopra in modo tale che diventi compatibile all'esecuzione in pipe-line (senza gestione di hazard). Modificare la CPU in due modi diversi: tenendo conto dei segnali di controllo e non tenendone conto. Quali diventano i registri? Cosa contengono? Quali passi di esecuzione separano? Da quanti bit sarà costituito ciascun registro? Cos'è uno stallo?

11. Definire cosa si intende per architettura CISC e RISC, e quali sono i rispettivi vantaggi e svantaggi.

12. Cosa rappresenta un hazard? Quando si verifica? Fare un esempio per ogni tipo di hazard.

13. Visualizzare con uno schema temporale e con un esempio, quali sono le dipendenze tra le istruzioni che provocano un hazard sui dati o un hazard sul controllo.

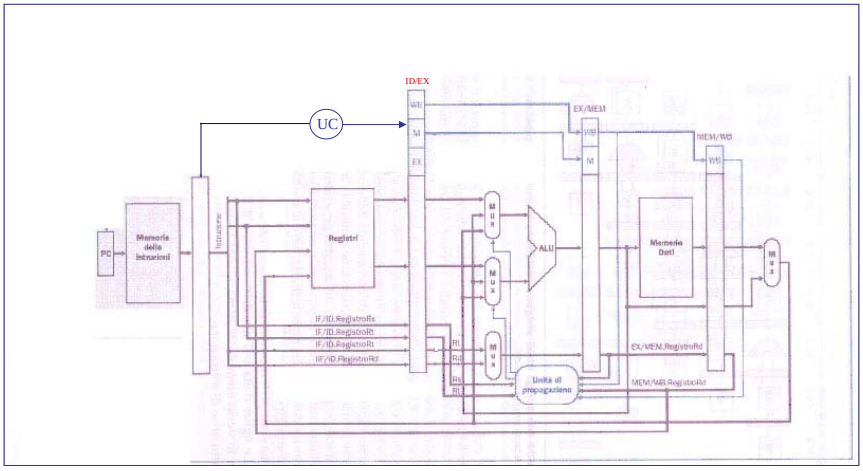
14. Dato lo schema a fianco, quale sarà il contenuto dei registri di pipeline (stato), quali saranno i segnali di controllo attivi e quali indifferenti, al termine (prima della commutazione del clock) del terzo stadio dell'istruzione in bold (sub):



add $t0, $t1, $t2
sub $t3, $t3, $t5
beq $t6, $t0, 16
add $t0, $t1, $t3
sapendo che $t0 = 7, i codici operativi di add e sub = 0, beq = 4; il codice Funct della add è 32 e della sub è 34.

15. Modificare lo schema della CPU con pipeline riportato sopra per potere gestire: un hazard sui dati dovuto ad istruzioni aritmetico-logiche. Scrivere le condizioni logiche che vengono utilizzate per identificare questo hazard e le funzioni logiche che servono a risolverlo.

16. Modificare lo schema della CPU, in modo tale da potere gestire:
a) Un hazard dovuto ad un'istruzione di load.
b) Un hazard sul controllo dovuto ad una beq.
Dare un esempio di codice in cui questi due hazard si verificano. E spiegare la dipendenza tra dati che origina l'hazard.



17. Dato lo schema qui a fianco, specificare il contenuto di tutti i registri ed i segnali di controllo negli stadi 1, 2, 3, 4, 5 di esecuzione della lw:
add $t0, $t1, $t2
addi $t0, $t1, 64
beq $t3, $t4, 16
lw $t3, 0($t0)
add $t6, $t6, $t7
add $t4, $t5, $t3
sapendo che i codici operativi di add, addi, beq, lw sono rispettivamente: 0, 8, 4, 35. Il codice funct della add è 32 e che $t0 = $7.

18. Modificare la CPU disegnata sopra in modo tale da potere gestire gli hazard generati da una lw. Specificare tutti i segnali di controllo.

19. Modificare la CPU disegnata sopra in modo tale da potere gestire gli hazard generati da una branch. Specificare tutti i segnali di controllo.

20. A cosa servono i registri causa e stato? Cosa si intende per mascheramento di un interrupt?

21. Descrivere tutte le operazioni necessarie alla gestione di un interrupt o eccezione.

22. Modificare la CPU di cui sopra, perchè sia in grado di gestire le eccezioni di overflow e di codice operativo non valido. Cosa è un'eccezione e un'interruzione? Che ruolo hanno i registri EPC, il registro Causa, Stato? Cos'è la maschera di interruzione? Quali sono le due modalità di risposta ad un'interruzione o eccezione? Cosa è il coprocessore 0? Come viene gestito? Cos'è un exception handler? Quali sono i passi da eseguire per servire in modo corretto un'eccezione? Quali sono i passi da eseguire per servire in modo corretto un interrupt? Ci sono differenze?

23. Definire i tipi di istruzioni disponibili. Quali di queste frasi sono corrette?
1) Tutte le istruzioni aritmetico-logiche sono di tipo R.
2) Le istruzioni di accesso alla memoria sono di tipo I.
3) Le istruzioni di salto sono di tipo I.
4) Le istruzioni con due operandi sorgenti su 32 bit sono di tipo R.
5) Le istruzioni di tipo R sono tutte istruzioni aritmetico-logiche.
6) Le istruzioni di tipo R hanno 2 operandi sorgenti su 32 bit.