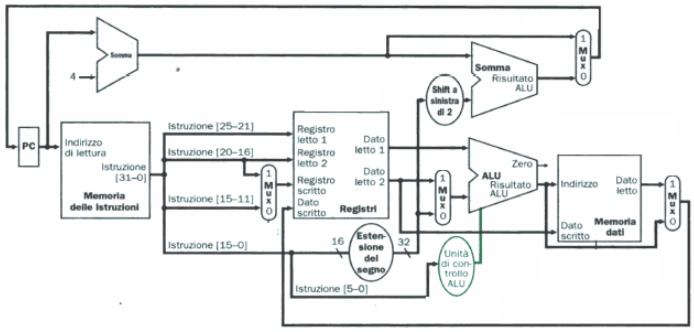
Esercitazione di ricapitolazione – parte III

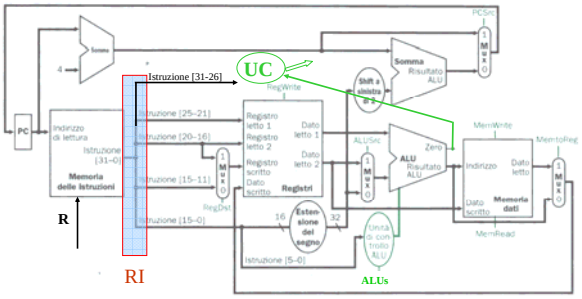
1. Descrivere il ciclo di esecuzione di un'istruzione. Di quali informazioni abbiamo bisogno in ciascuna fase?

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?

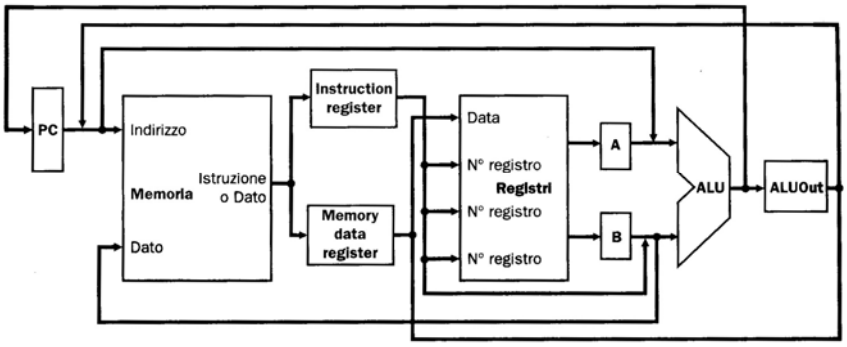


5. Specificare quali sono gli operandi su cui agiscono le 3 ALU e da dove provengono. Cosa contiene l'ingresso 0 del Mux del secondo operando della ALU quando sto eseguendo l'istruzione: add $t0, $t1, $t2, sapendo che $t0 = $7 e che Funct = 0x20. Cosa contiene l'uscita del Mux all'ingresso della porta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

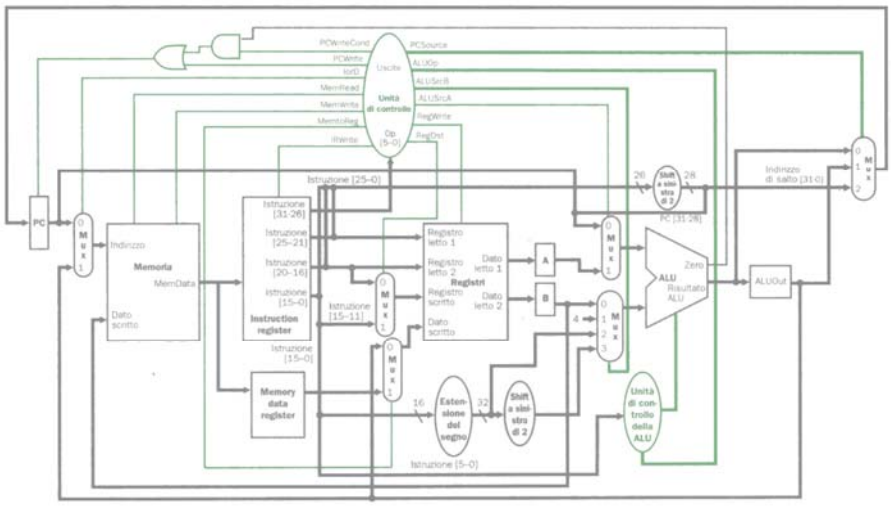
6. Quando conviene utilizzare una CPU multi-ciclo e quando una a CPU a ciclo singolo? Quando conviene una CPU con pipeline rispetto ad una CPU a ciclo singolo o multi-ciclo?

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

8. Data la CPU in figura:
Indicare quali sono i registri ed il loro ruolo.
Indicare l'inizio e la fine delle cinque fasi del ciclo di un'istruzione.
Specificare tutti i segnali di controllo necessari al suo funzionamento.
Specificare anche per quali segnali di controllo si può utilizzare il segnale di clock per quali non si può utilizzare e perchè.



10. Specificare il contenuto (lo stato) dei registri A e B, del registro ALUOut, del Registro Istruzioni, del MDR e del PC al termine del terzo stadio di esecuzione (prima della commutazione del clock) delle istruzioni:
lw $t0, 0($t1)
sw $t0, 0($t1)
add $t0, $t1, $t2
beq $t0, $t1, label
addi $t0, $t1, 16
Sapendo che i codici operativi sono rispettivamente: 35, 43, 0, 4, 8, che il campo funct della add è 32, che il registo $t0 è il registo $7.
Specificare tutti i segnali di controllo che sono attivi per le cinque istruzioni nelle diverse fasi di esecuzione. Per quali segnali di controllo è possibile utilizzare il segnale di clock?



11. Cosa succede nel quarto stadio di un'istruzione di branch? Cosa succede nel quindo stadio di esecuzione di una sw? Quando il segnale di scrittura del PC è attivo? Scrivere la funzione logica del segnale che comanda la scrittura del PC.

12. Sintetizzare la macchina a stati finiti (STG) che controlla la CPU disegnata sopra per le seguenti istruzioni: tipo R, beq, j, lw, sw, addi, subi, slti.

13. 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? 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?

14. 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? E di una CPU multi-ciclo? Motivare le risposte.

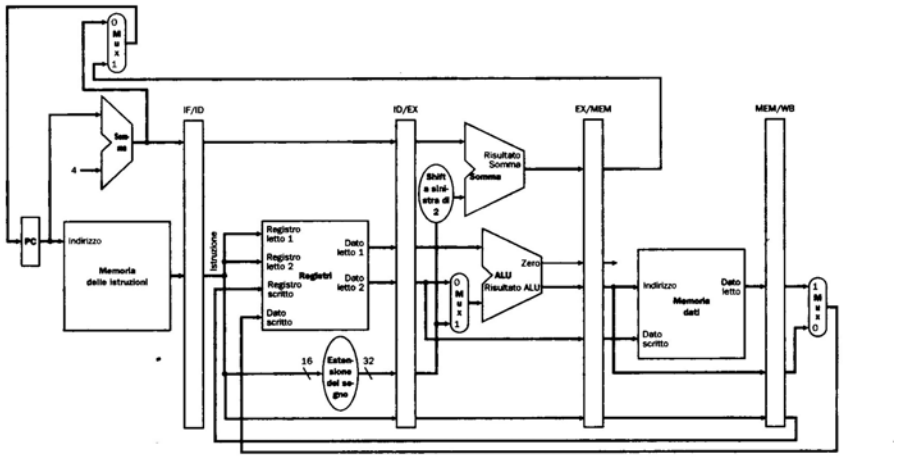
15. Modificare la CPU di cui sopra in modo tale che diventi compatibile all'esecuzione in pipe-line (senza gestione di hazard). Modificare la CPU in due modi diversi: tenendo conto dei segnali di controllo e non tenendone conto. Quali diventano i registri? Cosa contengono? Quali passi di esecuzione separano?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

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

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

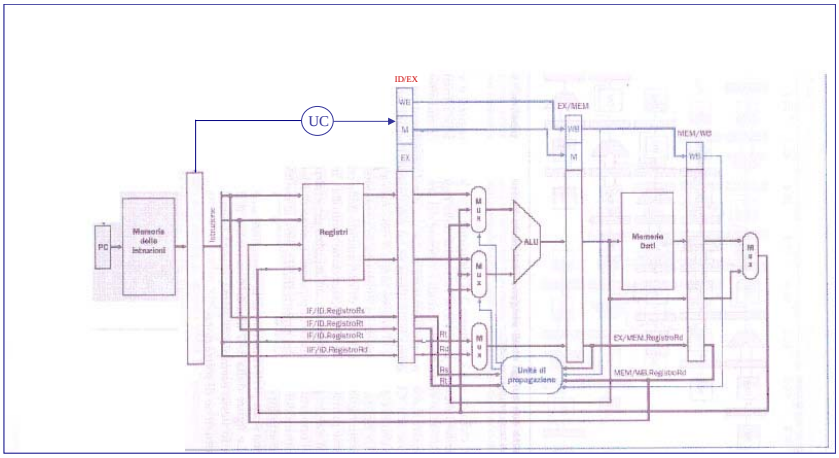
18. Visualizzare con uno schema temporale e con un esempio, quali sono le dipendenze tra le istruzioni che provocano un hazard sui dati o un hazard sul controllo.

17. 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.



20. 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.

21. 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.



22. 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.

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

24. Modificare la CPU disegnata sopra in modo tale da potere gestire gli hazard generati da una branch.

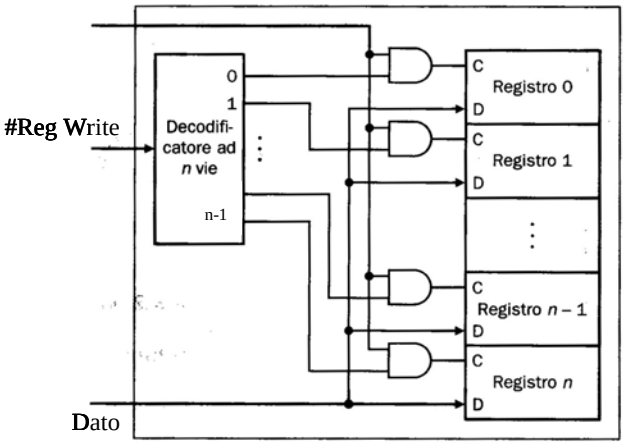
Specificare tutti i segnali di controllo.

25. Cos'è il branch prediction buffer? Cosa si intende per pipeline superscalare, superpipeline e scheduling dinamico? Che cosa sono le reorder station? E le reservation station? A cosa serve l'operazione di renaming dei registri? E' corretto dire che internamente la CPU di un Pentium IV ha un'ISA RISC e perchè? Quante cache primarie ha un Pentium IV e perchè? Cosa si intende per Pipeline superscalare?

26. Cosa si intende per esecuzione speculativa? Cos'è un branch prediction buffer? Come si può migliorare il funzionamento di una pipeline per la gestione dei salti condizionati?

27. Definire lo schema di massima dell'Architettura della pipe-line di un Pentium IV e descrivere quali sono i componenti principali e quali possono essere i problemi principali.

28. Dato il register file disegnato qui a fianco, spiegare come funziona in una singolo ciclo, nella quale occorre potere scrivere e leggere dati diversi nello stesso ciclo di clock, ed in una pipe-line, nella quale quello che scrivo in un ciclo di clock è anche quello che voglio che venga letto. Come funziona il register file in una multi-ciclo?



29. Scrivere una routine di Kernel (in assembly) di risposta ad un interupt di livello 0 nel MIPS. La routine stamperà il contenuto dei registri acausa e stato.

30. Scrivere la stessa routine di cui sopra per la gestione di un'eccezione.

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