


Architettura degli elaboratori - II
Le architetture multi-ciclo

Prof. Alberto Borghese
Dipartimento di Informatica
alberto.borghese@unimi.it

Università degli Studi di Milano


Sommario

Principi ispiratori di una CPU multi-ciclo. Le fasi di fetch e decodifica

Esecuzione multi-ciclo delle istruzioni R, lw/sw, beq, j

Unità di controllo multi-ciclo

Problemi

•Duplicazione della Memoria e triplicazione della ALU.
Tuttavia le unità funzionali sono utilizzate in fasi diverse del ciclo di esecuzione di un'istruzione (e.g. Memoria istruzioni in fase di fetch, Memoria dati in fase di Lettura/scrittura).

•Durata uguale per istruzioni che richiedono tempi molto diversi.
Il clock deve essere impostato secondo il cammino critico.

- Possibile soluzione: variare la durata del clock per ogni istruzione.
Non si sa quanto deve durare l'istruzione fino alla fase di decodifica.
Non praticabile.

Esecuzione in un singolo ciclo di clock

Assumiamo: memoria (2ns), ALU e sommatori (2ns), lettura/scrittura registri (1ns), decodifica (2ns), nessun ritardo, tempi trascurabili per gli altri elementi della CPU, componenti indipendenti possono lavorare in parallelo.

Istruzione	Memoria istruzioni	Lettura registri Decodifica	Operazione ALU	Memoria dati	Write back	Totale
Tipo R	2	2	2	0	1	7ns
lw	2	2	2	2	1	9ns
sw	2	2	2	2	0	8ns
beq	2	2	2	0	0	6ns
j	2	2	0	0	0	4ns

•La durata del ciclo di clock deve essere pari al percorso più lungo (cammino critico).
Percorso più lungo dovuto ad istruzione di caricamento (lw)

Valutazione della prestazione della CPU a singolo ciclo

Dipende dal programma.

	lw	sw	beq	j	R	fp (add)	fp (mul)	Durata Clock (max)	Durata media
Durata	9ns	8ns	6ns	4ns	7ns	12ns	20ns		
Caso I	24%	12%	18%	2%	44%			9ns	7.36ns
Caso II	31%	21%	5%	2%	27%	7%	7%	20ns	8.98ns

In ogni caso, un'implementazione a clock singolo porta ad uno spreco di tempo notevole.

Come gestire istruzioni di durata diversa?

Quando è efficiente l'implementazione a singolo ciclo?

Clock di durata variabile è una soluzione?

Non viene risolto il problema della duplicazione delle unità funzionali.

Caratteristiche CPU multi-ciclo

Spezza l'istruzione in più passi, dove ciascun passo impiega lo stesso tempo.

Il clock non sincronizza più l'intera istruzione ma solamente il singolo passo.

Le istruzioni possono essere eseguite in un numero diverso di cicli di clock.

Consente di riutilizzare le unità funzionali (in cicli di clock diversi).

Richiede l'aggiunta di HW addizionali (registri di memoria temporanea). Questi devono memorizzare lo stato delle unità funzionali, cioè l'informazione che può servire ai passi successivi e che rischia di essere sovrascritta dal riuso dell'unità funzionale.

L'unità di controllo diventa una FSM.

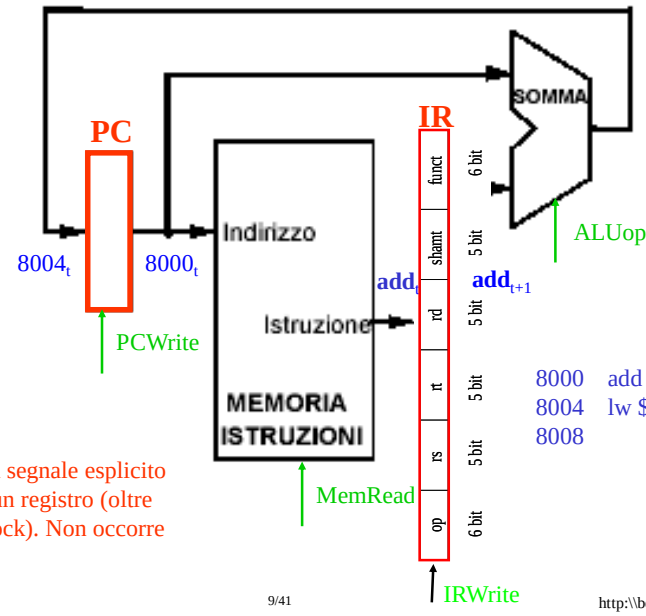
Sommario

Principi ispiratori di una CPU multi-ciclo. Le fasi di fetch e decodifica

Esecuzione multi-ciclo delle istruzioni R, lw/sw, beq, j

Unità di controllo multi-ciclo

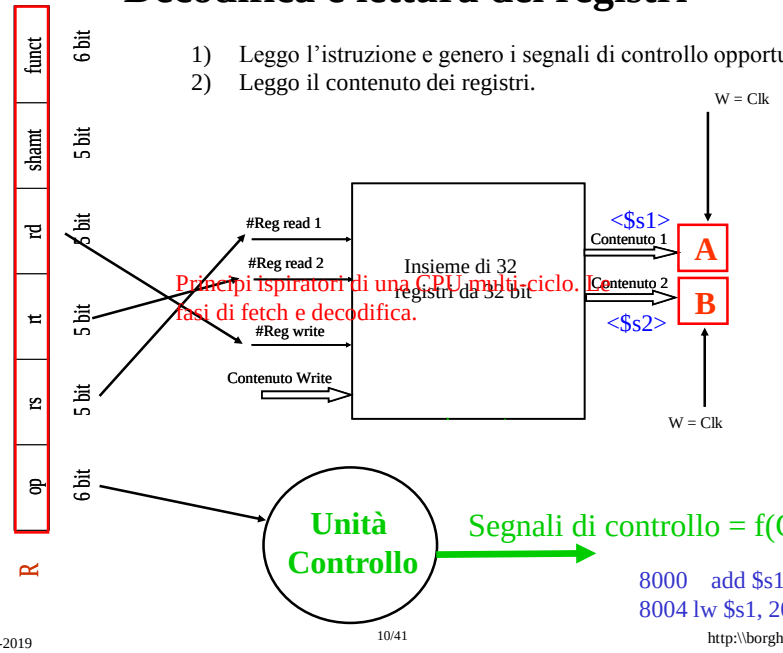
Circuito della fase di fetch multi-ciclo



8000 add $s1, $s2, $s3
8004 lw $s1, 20($s2)
8008

NB: Occorre un segnale esplicito per scrivere in un registro (oltre al segnale di clock). Non occorre per la lettura.

Decodifica e lettura dei registri

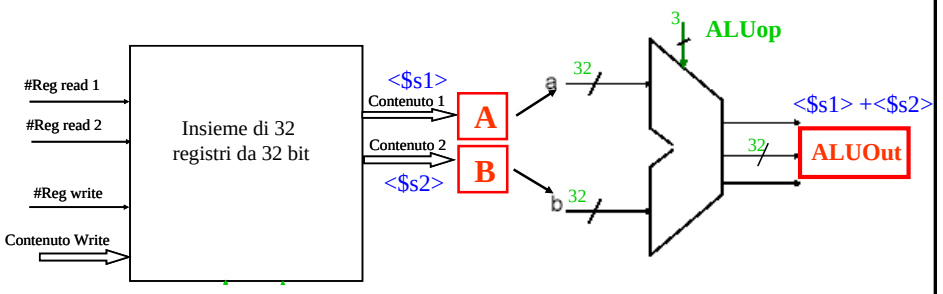


1) Leggo l'istruzione e genero i segnali di controllo opportuni.
2) Leggo il contenuto dei registri.

Principi ispiratori di una CPU multi-ciclo. Le fasi di fetch e decodifica.

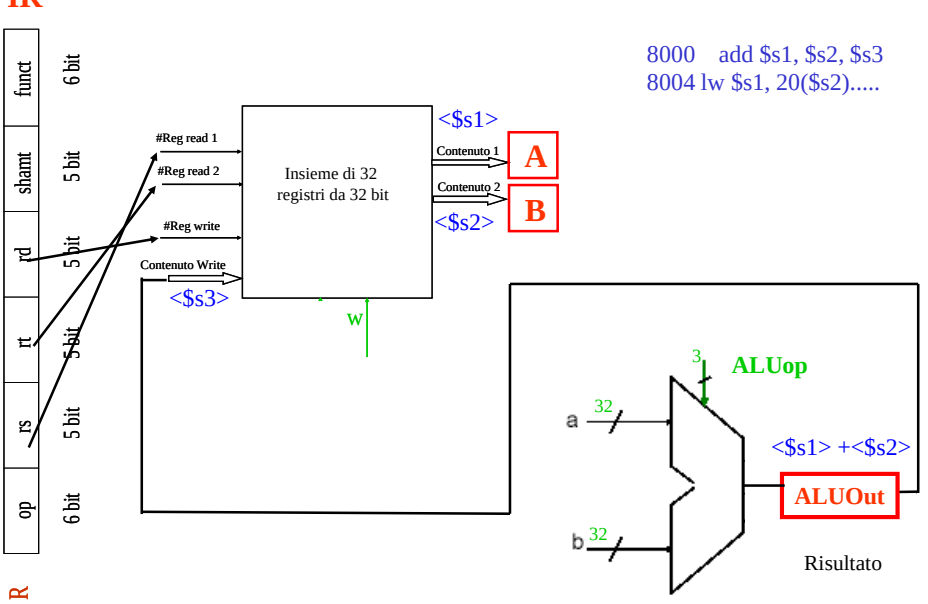
8000 add $s1, $s2, $s3
8004 lw $s1, 20($s2).....

Fase di calcolo (tipo R)



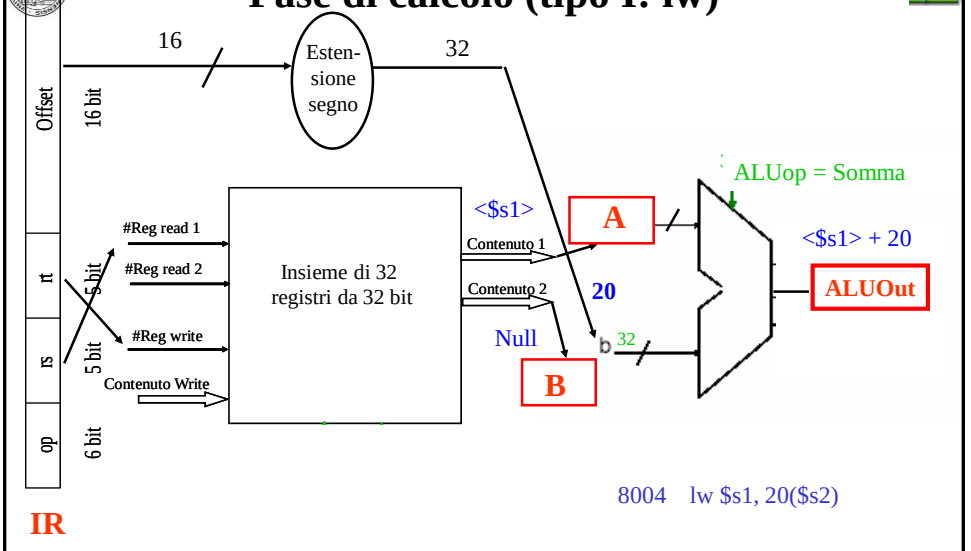
8000 add $s1, $s2, $s3
8004 lw $s1, 20($s2).....

Fase di Write back (tipo R)

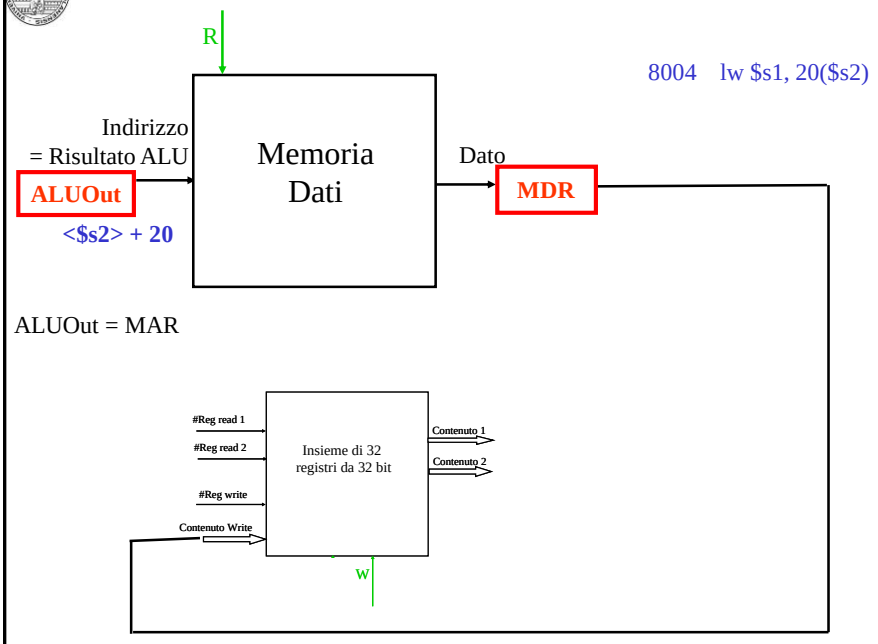


8000 add $s1, $s2, $s3
8004 lw $s1, 20($s2).....

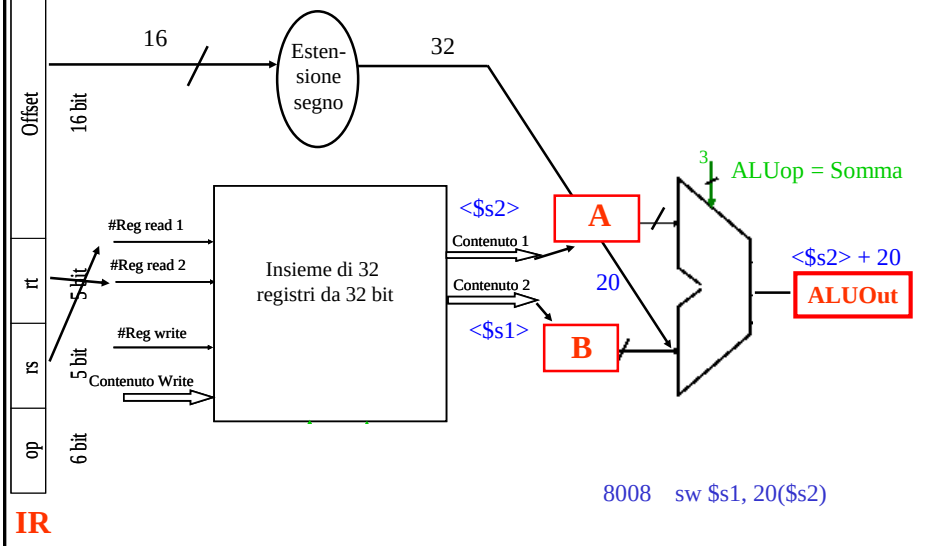
Fase di calcolo (tipo I: lw)



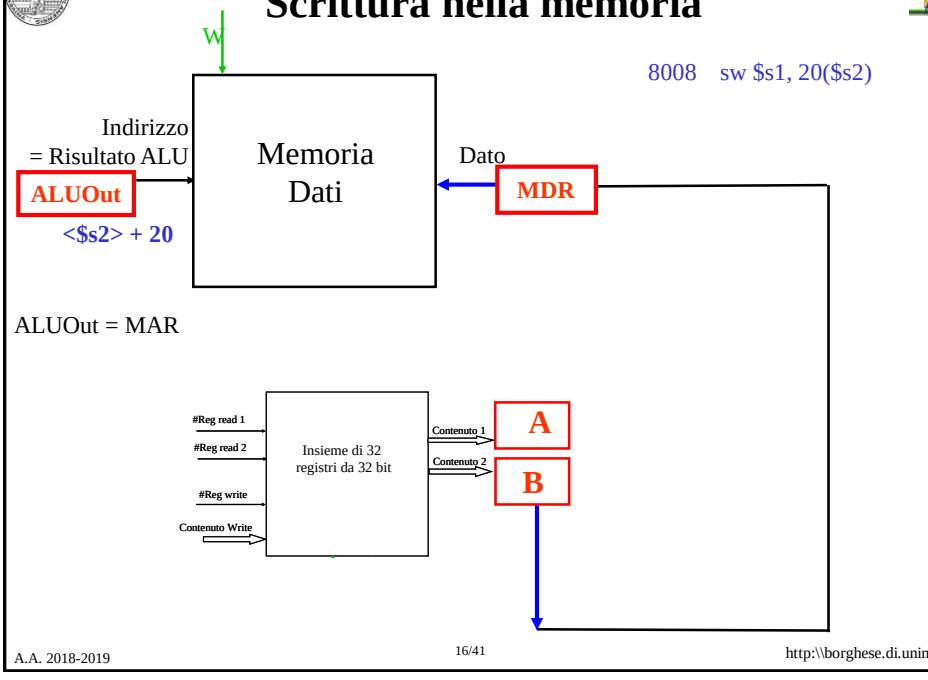
Lettura della memoria + write-back



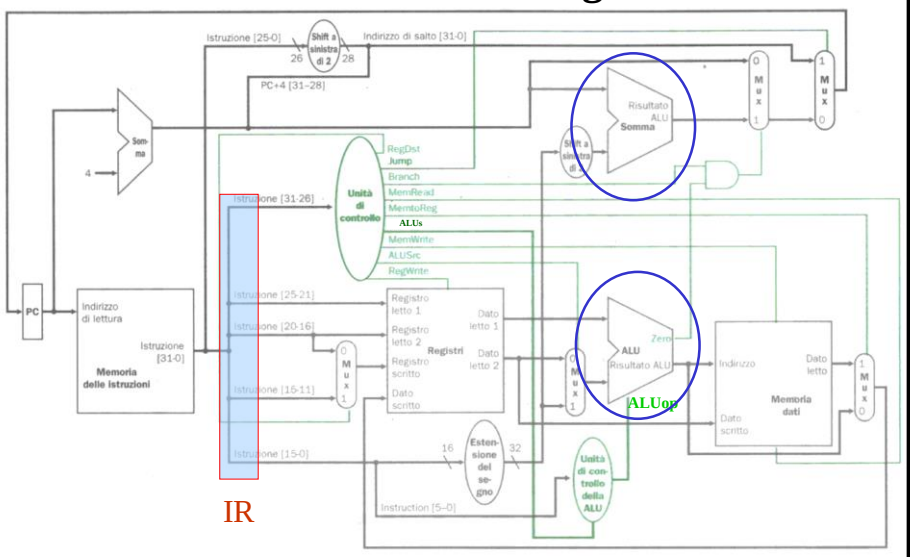
Fase di calcolo (tipo I: sw)



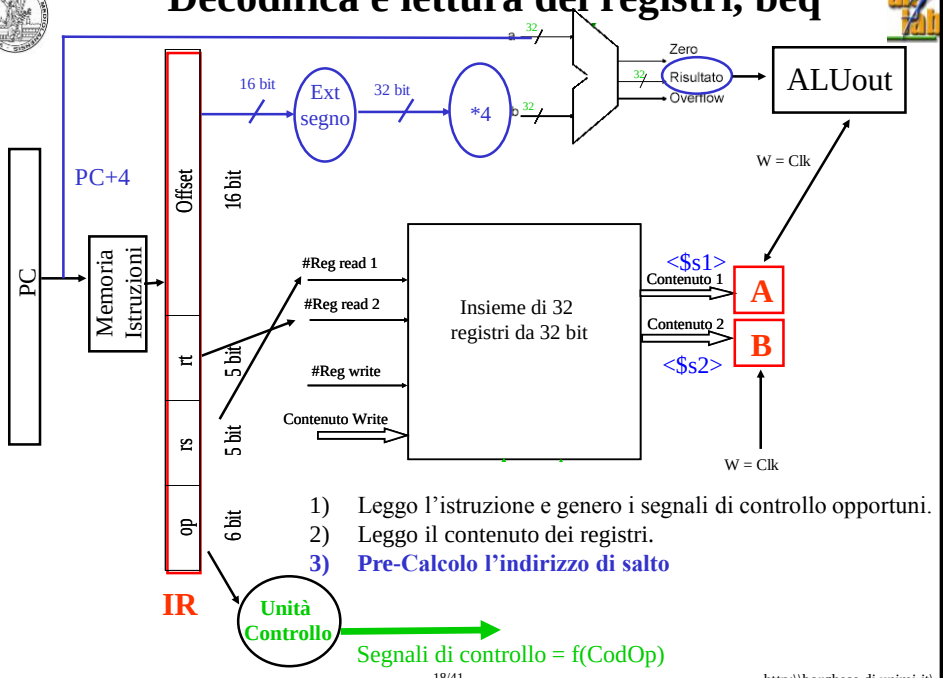
Scrittura nella memoria



CPU + UC a ciclo singolo

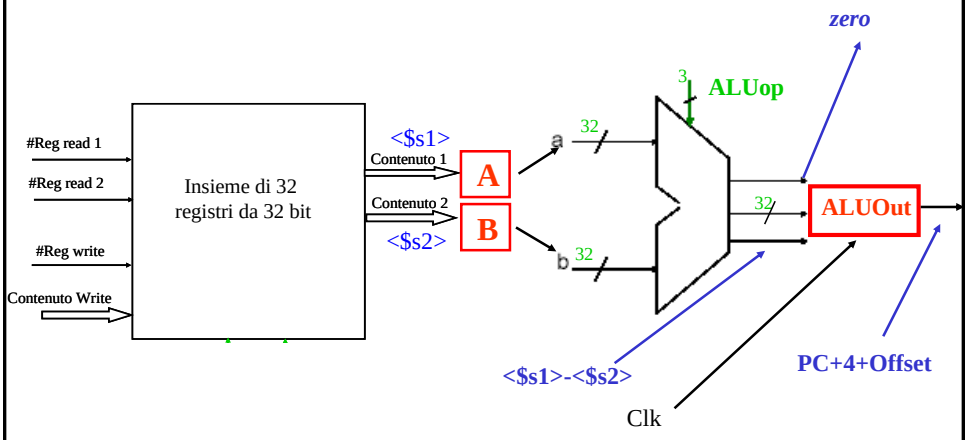


Decodifica e lettura dei registri, beq

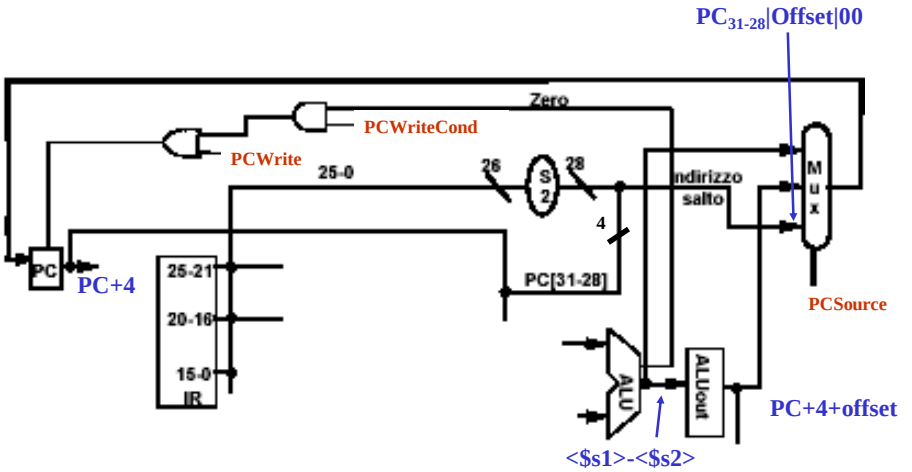


1) Leggo l'istruzione e genero i segnali di controllo opportuni.
2) Leggo il contenuto dei registri.
3) **Pre-Calcolo l'indirizzo di salto**

Fase di calcolo (beq)



CPU multi-ciclo: i salti



Mux sceglie tramite PCSource tra:
- Indirizzo ottenuto sommando 4.
- Indirizzo ottenuto dall'ALU (beq).
- Indirizzo ottenuto dal campo offset dell'IR (jump).

PCWrite, e PC Source sono coordinati
PCWriteCond e PCSource sono coordinati.

Confronto delle 2 CPU

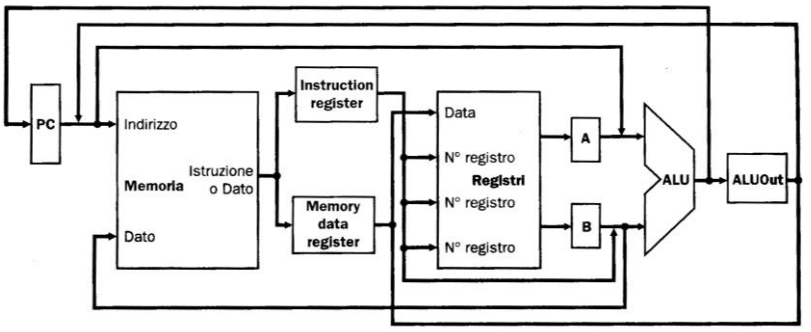
Due Memorie. Si possono compattare a patto di creare 2 registri. 1 registro istruzione (RI) ed un registro dati (MDR).

3 ALU. Si può utilizzare una unica ALU se viene utilizzata in 3 fasi diverse.

Fase 1: + 4.

Fase 2 Offset + rs (pre-calcolo l'indirizzo di salto).

Fase 3. Operazione di tipo R.



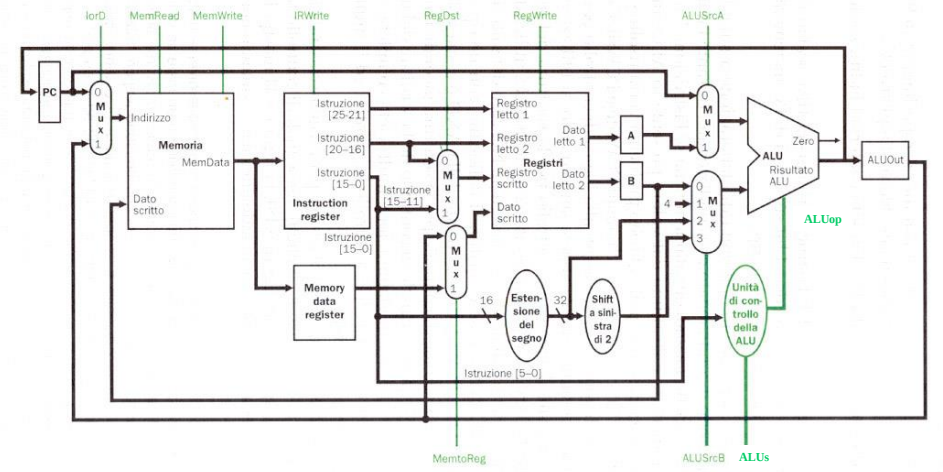
Sommario

Principi ispiratori di una CPU multi-ciclo. Le fasi di fetch e decodifica

Esecuzione multi-ciclo delle istruzioni R, lw/sw, beq, j

Unità di controllo multi-ciclo

CPU multi-ciclo per R, lw/sw, beq



Analisi dell'utilizzo dei registri

PC – Sincronizza il ciclo di esecuzione di un'istruzione. Viene aggiornato in fase di fetch (+4). In questo caso contiene l'indirizzo dell'istruzione successiva. Può essere aggiornato anche in fase di decodifica (jump) oppure in fase di esecuzione (beq).

RI – Viene aggiornato alla fine della FF. Mantiene per tutte le fasi, le informazioni sull'istruzione: numero di registri, offset, funct, shmt, opcode. Questa informazione viene utilizzata da:

- Istruzioni di tipo beq, jump in fase di Decodifica.
- Istruzioni di tipo R e lw, in fase di Decodifica e WriteBack.
- Istruzioni di tipo sw, in fase di Decodifica.

A e B – Vengono aggiornati sempre. Il loro valore cambia solo alla fine della DECOD (nelle altre fasi l'uscita del register file non cambia). Mantiene il contenuto dei registri rs e rt. Il contenuto del 2o registro può non essere utilizzato (lw).

ALUOut – Viene aggiornato sempre. Nei primi 3 stadi assume rispettivamente il valore di:

- FF (contiene PC + 4).
- DECOD (Contiene PC + Offset * 4).
- EXEC (Contiene Reg Read 1 + (Reg Read 2 *oppure* Offset * 4).

MDR – Viene aggiornato solo nelle istruzioni lw/sw.

Register File – Viene aggiornato solo nella fase di WriteBack nelle istruzioni R e lw.

Scrittura nei registri della CPU multi-ciclo

Tutti i registri interni della CPU vengono scritti ad ogni colpo di clock. Non c'è bisogno di sintetizzare nella UC il segnale di scrittura dei registri.

TRANNE CHE PER l'IR ed il PC.

Tutti i registri possono essere riscritti ad ogni passo dell'istruzione, tranne l'IR che deve mantenere il suo valore per tutta la durata dell'esecuzione di un'istruzione ed il PC che viene aggiornato in fase di fetch e nei salti.

Register File e Memoria vengono scritti solamente nelle fasi di WB e Memoria rispettivamente.

Segnali di controllo

Nome del segnale	Valore	Effetto
ALUSrcA (1 bit)	0	Il primo operando è il valore attuale del PC
	1	Il primo operando proviene dalla prima porta di lettura del Register File
ALUSrcB (2 bit)	00	Il secondo operando proviene dalla seconda porta di lettura del RF
	01	Il secondo operando è la costante + 4
	10	Il secondo operando è l'estensione del segno del campo offset
	11	Il secondo operando proviene dall'estensione del segno e dallo shift a sx di due posizioni, dell'offset
IorD	0	L'indirizzo della memoria proviene dal PC
	1	L'indirizzo della memoria proviene dalla ALU (ALUOut)
PCSource	00	In PC viene scritta l'uscita della ALU (PC+4)
	01	In PC viene scritta il contenuto di ALUOut (indirizzo di una branch)
	10	In PC viene scrittto l'indirizzo di destinazione della jump
PCWrite	0	Nessuno
	1	Viene scritto il registro PC. L'indirizzo scritto in PC è controllato da PCSource
PCWriteCond	0	Nessuno
	1	Il PC viene scritto se anche l'uscita Zero della ALU è affermata

Riassunto dell'esecuzione

Nome del passo	Azioni per Istruzioni di Tipo R	Azioni per istruzioni di accesso alla memoria	Azioni per salti condizionati	Azioni per salti non condizionati
Fetch	IR = Memory[PC] PC = PC + 4			
decodifica & Prelievo dati dai registri	A = Reg[IR[25-21]] B = Reg[IR[20-16]] ALUOut = PC+4 + (sign_extend(IR[15-0] << 2)			
Esecuzione	ALUOut = A oper B	ALUOut = A + sign_extend(IR[15-0])	If (A == B) then PC = ALUOut	PC = PC[31-28] || (IR[25-0]<<2)
Memoria o WriteBack		Load: MDR = Memory[ALUOut] Store: Memory[ALUOut] = B		
WriteBack	Reg([IR[15-11]] = ALUOut	Load: Reg[IR[20-16]] = MDR		

Le istruzioni richiedono da 3 a 5 cicli di clock

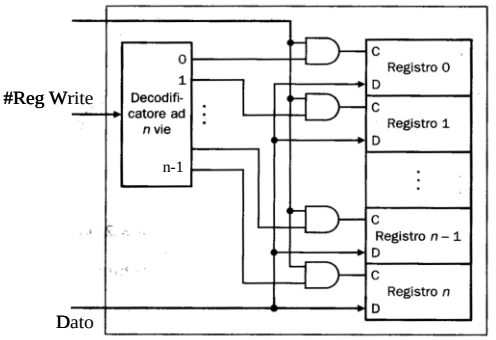
Altri segnali di controllo

Scrittura del Register File – solamente nelle istruzioni con fase di WriteBack.

Scrittura / lettura Memoria – solamente nelle istruzioni di accesso alla memoria.

Scrittura nei registri A, B, ALUOut – ad ogni colpo di clock.

Scrittura del IR – Avviene durante la fase di fetch.



Segnali di controllo

Passo esecuzione	IorD	MemRead	MemWrite	IRWrite	ALUScrA	ALUSrcB	ALUs	PCSource	PCWrite	PCWriteCond	RegWrite	RegDst	MemtoReg
Fase fetch													
Decodifica													
Calcolo – lw													
Memoria – lw													
WB – lw													
Calcolo – sw													
Memoria – sw													
Calcolo – R													
WB – R													
Calcolo – beq													
Calcolo – j													

Segnali di controllo

Passo esecuzione	IorD	MemRead	MemWrite	IRWrite	ALUScrA	ALUSrcB	ALUs	PCSource	PCWrite	PCWriteCond	RegWrite	RegDst	MemtoReg
Fase fetch	0	1	0	1	0	01	00	00	1	X	0	X	X
Decodifica	X	0	0	0	0	11	00	X	0	0	0	X	X
Calcolo – lw	X	0	0	0	1	10	00	X	0	0	0	X	X
Memoria – lw	1	1	0	0	X	X	X	X	0	0	0	X	X
WB – lw	X	0	0	0	X	X	X	X	0	0	1	0	1
Calcolo – sw	X	0	0	0	1	10	00	X	0	0	0	X	X
Memoria – sw	1	0	1	0	X	X	X	X	0	0	0	X	X
Calcolo – R	X	0	0	0	1	00	10	X	0	0	0	X	X
WB – R	X	0	0	0	X	X	X	X	0	0	1	1	0
Calcolo – beq	X	0	0	0	1	00	01	01	0	1	0	X	X
Calcolo – j	X	0	0	0	X	X	X	10	1	X	0	X	X

Sintesi della FSM della CPU

•Stato – passo di esecuzione.
•Uscita – segnali di controllo.
•Ingressi – Codice operativo.

I valori dei segnali di controllo dipendono:
•dal passo dell'istruzione (stato)

Il passo successivo dell'istruzione (stato prossimo) dipende:
• dal codice operativo (ingresso).
• dal passo presente (stato presente).

•Uscita = f(Stato)
•Stato_prossimo = f(Ingressi, Stato_presente)

Macchina a Stati Finiti (di Moore)

La Macchina di Moore è definita, in teoria degli automi, dalla quintupla :

< X, I, Y, f(.), g(.) >

X: insieme degli stati (in numero finito).

I: alfabeto di ingresso: tutti i simboli che si possono presentare in ingresso. Se abbiamo n ingressi, avremo 2^n possibili simboli da leggere in ingresso (configurazioni).

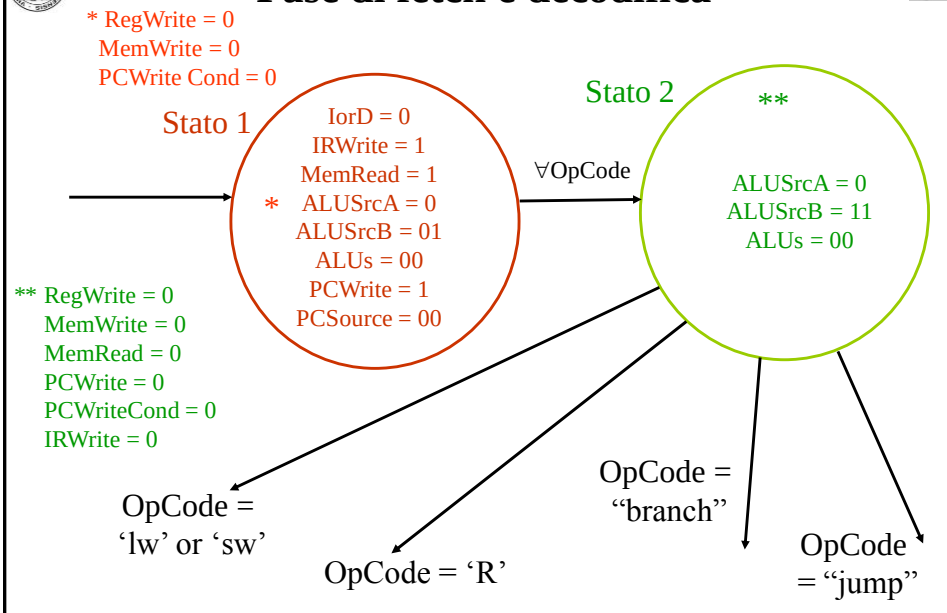
Y: alfabeto di uscita: tutti i simboli che si possono generare in uscita. Se abbiamo m uscite, avremo 2^m possibili simboli da presentare in uscita (configurazioni).

f(.): funzione stato prossimo: X' = f(X,I). Definisce l'evoluzione della macchina nel tempo. L'evoluzione è deterministica.

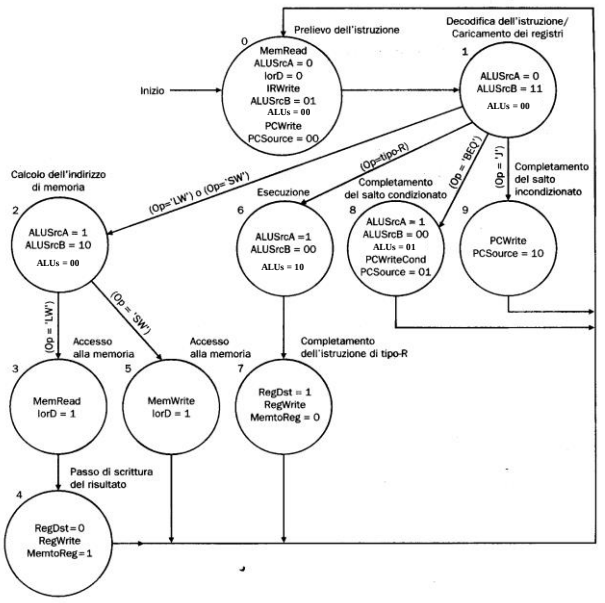
g(.): funzione di uscita: Y= g(X) nelle macchien di Moore.

Per il buon funzionamento della macchina è previsto uno stato iniziale, al quale la macchina può essere portata mediante un comando di reset.

Fase di fetch e decodifica



FSM - STG



FSM - STT

Stato (Passo di esecuzione)	OpCode = R	OpCode = sw	OpCode = lw	OpCode = beq	OpCode = j	IorD	MemRead	MemWrite	IRWrite	ALUScrA	ALUSrcB	ALUs	PCSource	PCWrite	PCWriteCond	RegWrite	RegDst	MemtoReg
Fase fetch - 0						0	1	0	1	0	01	00	00	1	X	0	X	X
Decodifica - 1						X	0	0	0	0	11	00	X	0	0	0	X	X
Calcolo – lw/sw - 2						X	0	0	0	1	10	00	X	0	0	0	X	X
Memoria – lw - 3						1	1	0	0	X	X	X	X	0	0	0	X	X
WB – lw - 4						X	0	0	0	X	X	X	X	0	0	1	0	1
Memoria – sw – 5						1	0	1	0	X	X	X	X	0	0	0	X	X
Calcolo – R - 6						X	0	0	0	1	00	10	X	0	0	0	X	X
WB – R - 7						X	0	0	0	X	X	X	X	0	0	1	1	0
Calcolo – beq .- 8						X	0	0	0	1	00	01	01	0	1	0	X	X
Calcolo – j - 9						X	0	0	0	X	X	X	10	1	X	0	X	X

FSM - STT

Stato (Passo di esecuzione)	OpCode = R	OpCode = sw	OpCode = lw	OpCode = beq	OpCode = j	IorD	MemRead	MemWrite	IRWrite	ALUScrA	ALUSrcB	ALUs	PCSource	PCWrite	PCWriteCond	RegWrite	RegDst	MemtoReg
Fase fetch - 0	1	1	1	1	1	0	1	0	1	0	01	00	00	1	X	0	X	X
Decodifica - 1	6	2	2	8	9	X	0	0	0	0	11	00	X	0	0	0	X	X
Calcolo – lw/sw - 2	X	5	3	X	X	X	0	0	0	1	10	00	X	0	0	0	X	X
Memoria – lw - 3	X	X	4	X	X	1	1	0	0	X	X	X	X	0	0	0	X	X
WB – lw - 4	X	X	0	X	X	X	0	0	0	X	X	X	X	0	0	1	0	1
Memoria – sw – 5	X	0	X	X	X	1	0	1	0	X	X	X	X	0	0	0	X	X
Calcolo – R - 6	7	X	X	X	X	X	0	0	0	1	00	10	X	0	0	0	X	X
WB – R - 7	0	X	X	X	X	X	0	0	0	X	X	X	X	0	0	1	1	0
Calcolo – beq .- 8	X	X	X	0	X	X	0	0	0	1	00	01	01	0	1	0	X	X
Calcolo – j - 9	X	X	X	X	0	X	0	0	0	X	X	X	10	1	X	0	X	X

FSM – sintesi della funzione di uscita

Stato $s_0\ s_1\ s_2\ s_3$	IorD	MemRead	MemWrite	IRWrite	ALUScrA	ALUSrcB	ALUs	PCSource	PCWrite	PCWriteCond	RegWrite	RegDst	MemtoReg
Fase fetch - 0000	0	1	0	1	0	01	00	00	1	X	0	X	X
Decodifica - 0001	X	0	0	0	0	11	00	X	0	0	0	X	X
Calcolo–lw/sw - 0010	X	0	0	0	1	10	00	X	0	0	0	X	X
Memoria – lw - 0011	1	1	0	0	X	X	X	X	0	0	0	X	X
WB – lw - 0100	X	0	0	0	X	X	X	X	0	0	1	0	1
Memoria – sw – 0101	1	0	1	0	X	X	X	X	0	0	0	X	X
Calcolo – R - 0110	X	0	0	0	1	00	10	X	0	0	0	X	X
WB – R - 0111	X	0	0	0	X	X	X	X	0	0	1	1	0
Calcolo – beq .- 1000	X	0	0	0	1	00	01	01	0	1	0	X	X
Calcolo – j - 1001	X	0	0	0	X	X	X	10	1	X	0	X	X

Esempi:
Y (RegWrite) = (Stato == 7) OR (Stato == 4)
Y (ALUSrcA) = (Stato == 8) OR (Stato == 6) OR (Stato == 2)

FSM – codifica della STT (stato futuro)

Passo esecuzione	OpCode = R 000000	OpCode = sw 101011	OpCode = lw 100011	OpCode = beq 000100	OpCode = j 000010
Fase fetch - 0000	0 0 0 1	0 0 0 1	0 0 0 1	0 0 0 1	0 0 0 1
Decodifica - 0001	0 1 1 0	0 0 1 0	0 0 1 0	1 0 0 0	1 0 0 1
Calcolo – lw/sw - 0010	X X X X	0 1 0 1	0 0 1 1	X X X X	X X X X
Memoria – lw - 0011	X X X X	X X X X	0 1 0 0	X X X X	X X X X
WB – lw - 0100	X X X X	X X X X	0 0 0 0	X X X X	X X X X
Memoria – sw – 0101	X X X X	0 0 0 0	X X X X	X X X X	X X X X
Calcolo – R - 0110	0 1 1 1	X X X X	X X X X	X X X X	X X X X
WB – R - 0111	0 0 0 0	X X X X	X X X X	X X X X	X X X X
Calcolo – beq .- 1000	X X X X	X X X X	X X X X	0 0 0 0	X X X X
Calcolo – j - 1001	X X X X	X X X X	X X X X	X X X X	0 0 0 0

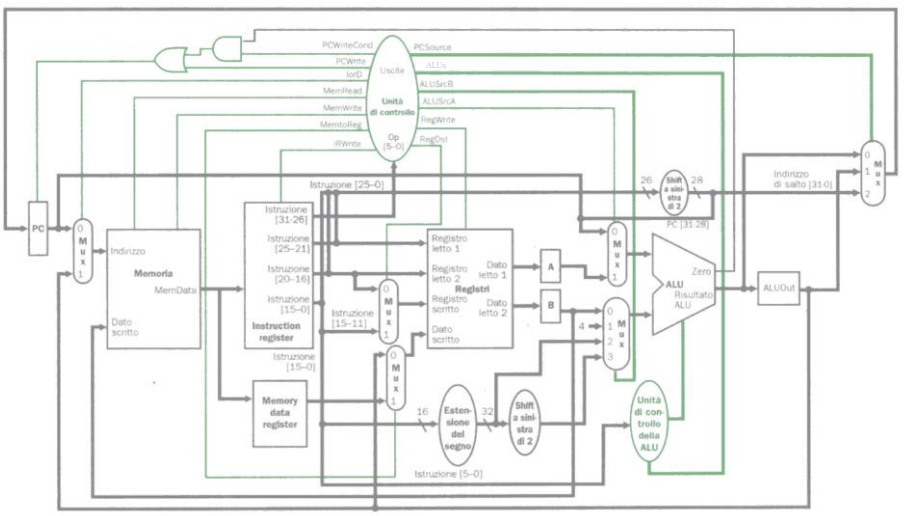
FSM – circuito dello stato futuro

Non dipende da OpCode

Esempio: $S_o(t+1)=(\bar{S}_o\bar{S}_1\bar{S}_2\bar{S}_3)+(\bar{S}_o\bar{S}_1\bar{S}_2\bar{S}_3)(\bar{i}_o i_1 \bar{i}_2 \bar{i}_3 \bar{i}_4 \bar{i}_5)+(\bar{S}_o\bar{S}_1\bar{S}_2\bar{S}_3)(\bar{i}_o i_1 \bar{i}_2 \bar{i}_4 i_5)+(\bar{S}_o\bar{S}_1\bar{S}_2\bar{S}_3)$

Passo esecuzione	OpCode = R 000000	OpCode = sw 101011	OpCode = lw 100011	OpCode = beq 000100	OpCode = j 000010
Fase fetch - 0000	0 0 0 1	0 0 0 1	0 0 0 1	0 0 0 1	0 0 0 1
Decodifica - 0001	0 1 1 0	0 0 1 0	0 0 1 0	1 0 0 0	1 0 0 1
Calcolo – lw/sw - 0010	X X X X	0 1 0 1	0 0 1 1	X X X X	X X X X
Memoria – lw - 0011	X X X X	X X X X	0 1 0 0	X X X X	X X X X
WB – lw - 0100	X X X X	X X X X	0 0 0 0	X X X X	X X X X
Memoria – sw – 0101	X X X X	0 0 0 0	X X X X	X X X X	X X X X
Calcolo – R - 0110	0 1 1 1	X X X X	X X X X	X X X X	X X X X
WB – R - 0111	0 0 0 0	X X X X	X X X X	X X X X	X X X X
Calcolo – beq .- 1000	X X X X	X X X X	X X X X	0 0 0 0	X X X X
Calcolo – j - 1001	X X X X	X X X X	X X X X	X X X X	0 0 0 0

CPU multi-ciclo



Sommario

I segnali di controllo della CPU multi-ciclo

Sintesi dell'Unità di Controllo come Macchina a Stati Finiti