



L'unità di controllo di CPU multi-ciclo

Prof. Alberto Borghese
Dipartimento di Scienze dell'Informazione

borgnese@dsi.unimi.it

Università degli Studi di Milano

Riferimento sul Patterson: Sezione C3



Sommario

I segnali di controllo della CPU multi-ciclo

Sintesi dell'Unità di Controllo come Macchina a Stati Finiti



Confronto delle 2 CPU



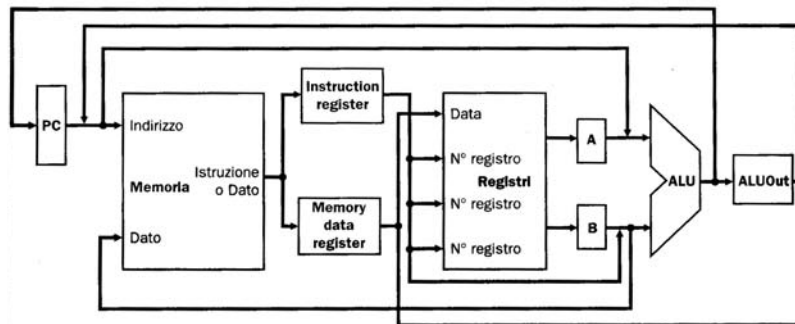
Due Memorie. Si possono compattare a patto di creare 2 registri. 1 registro istruzione (RI) ed un registro dati (MDR).

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

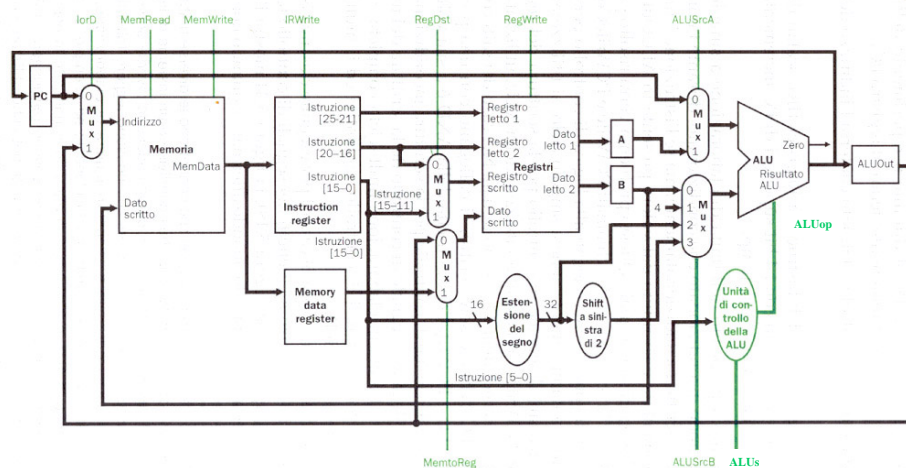
Fase 1: + 4.

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

Fase 3. Operazione di tipo R.

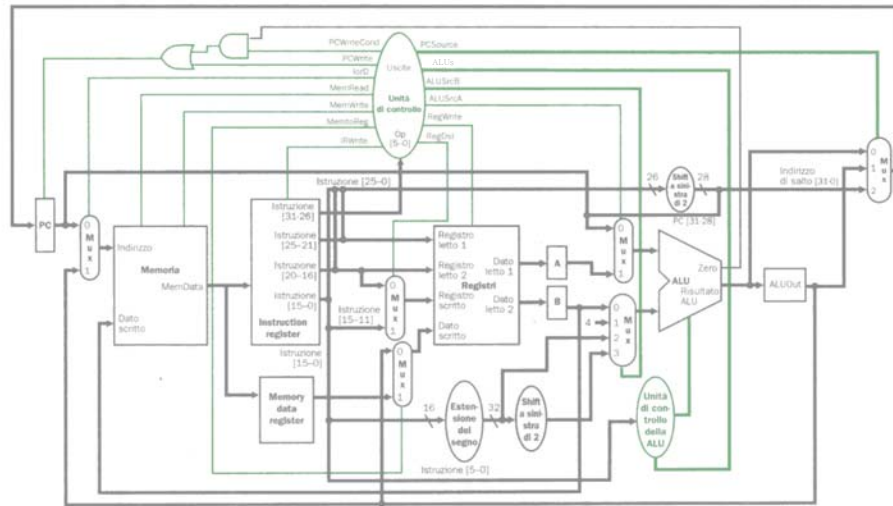


CPU multi-ciclo per R, lw/sw, beq





CPU multi-ciclo



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 scritto 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

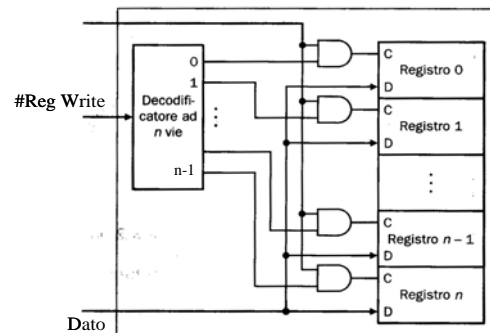


Scrittura del Register File – solamente nelle istruzioni con fase di WriteBack.

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.





Sommario



I segnali di controllo della CPU multi-ciclo

Sintesi dell'Unità di Controllo come Macchina a Stati Finiti



Segnali di controllo



Passo esecuzione	IoRD	MemRead	MemWrite	IRWrite	ALUScrA	ALUSrcB	ALUs	PCSource	PCWrite	PCWriteCond	RegWrite	RegDst	MemoReg
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	ALUScrB	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 :

$$\langle X, I, Y, f(\cdot), g(\cdot) \rangle$$

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 macchine 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



* RegWrite = 0
MemWrite = 0
PCWrite Cond = 0

Stato 1

IorD = 0
IRWrite = 1
MemRead = 1
* ALUSrcA = 0
ALUSrcB = 01
ALUs = 00
PCWrite = 1
PCSource = 00

Stato 2

**
ALUSrcA = 0
ALUSrcB = 11
ALUs = 00

\forall OpCode

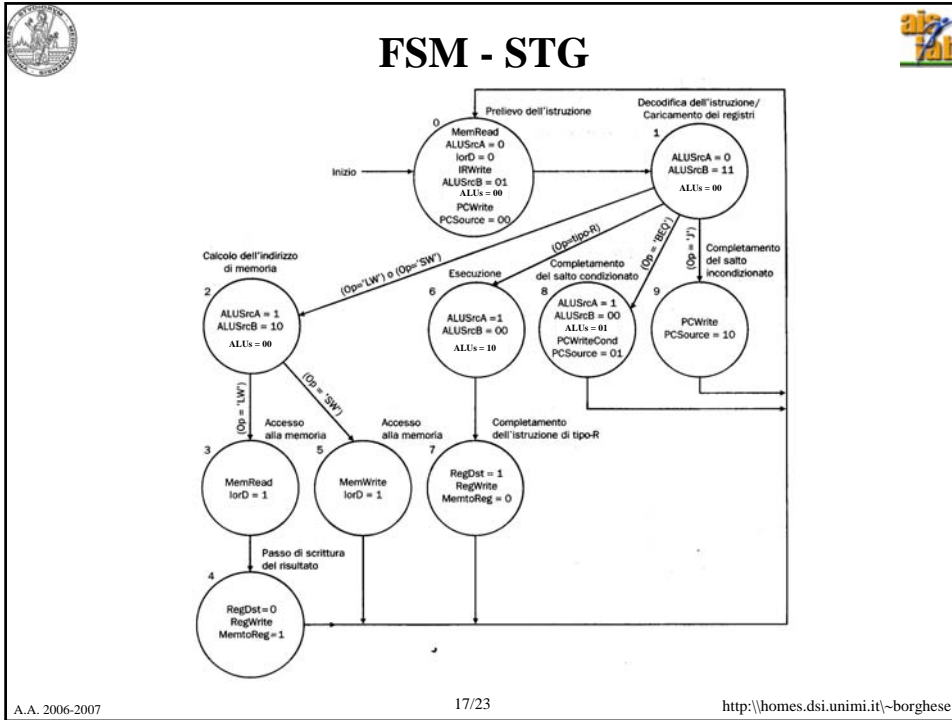
** RegWrite = 0
MemWrite = 0
MemRead = 0
PCWrite = 0
PCWriteCond = 0
IRWrite = 0

OpCode =
'lw' or 'sw'

OpCode = 'R'

OpCode =
"branch"

OpCode =
"jump"



FSM - STT

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

A.A. 2006-2007 18/23 <http://homes.dsi.unimi.it/~borgnese>



FSM - STT



Stato (Passo di esecuzione)	OpCode = R	OpCode = sw	OpCode = lw	OpCode = beq	OpCode = j	lOrd	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 $s_0 s_1 s_2 s_3$	lOrd	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:
 $Y(\text{RegWrite}) = (\text{Stato} == 7) \text{ OR } (\text{Stato} == 4)$
 $Y(\text{ALUSrcA}) = (\text{Stato} == 8) \text{ OR } (\text{Stato} == 6)$
 $\text{OR } (\text{Stato} == 2)$



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	0001	0001	0001	0001	0001
Decodifica - 0001	0110	0010	0010	1000	1001
Calcolo – lw/sw - 0010	XXXX	0101	0011	XXXX	XXXX
Memoria – lw - 0011	XXXX	XXXX	0100	XXXX	XXXX
WB – lw - 0100	XXXX	XXXX	0000	XXXX	XXXX
Memoria – sw – 0101	XXXX	0000	XXXX	XXXX	XXXX
Calcolo – R - 0110	0111	XXXX	XXXX	XXXX	XXXX
WB – R - 0111	0000	XXXX	XXXX	XXXX	XXXX
Calcolo – beq .- 1000	XXXX	XXXX	XXXX	0000	XXXX
Calcolo – j - 1001	XXXX	XXXX	XXXX	XXXX	0000



FSM – circuito dello stato futuro



Non dipende da OpCode

$$\text{Esempio: } S_0(t+1) = (S_0 S_1 S_2 S_3) + (S_0 S_1 S_2 S_3)(i_0 i_1 i_2 i_3 i_4 i_5) + (S_0 S_1 S_2 S_3)(i_0 i_1 i_2 i_4 i_5) + (S_0 S_1 S_2 S_3)$$

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



Sommario



I segnali di controllo della CPU multi-ciclo

Sintesi dell'Unità di Controllo come Macchina a Stati Finiti