


L'unità di controllo di CPU a singolo ciclo

Prof. Alberto Borghese
Dipartimento di Scienze dell'Informazione
borghese@dsi.unimi.it

Università degli Studi di Milano

Riferimento sul Patterson: capitolo 5 (fino a 5.4) e C1, C2.


Sommario

CPU che gestisce istruzioni di tipo R, lw/sw, branch

Controllore della ALU

Unità di Controllo Principale

Tipi di istruzioni dell'ISA

Consideriamo istruzioni di tipo R, di tipo lw/sw, salto condizionato:

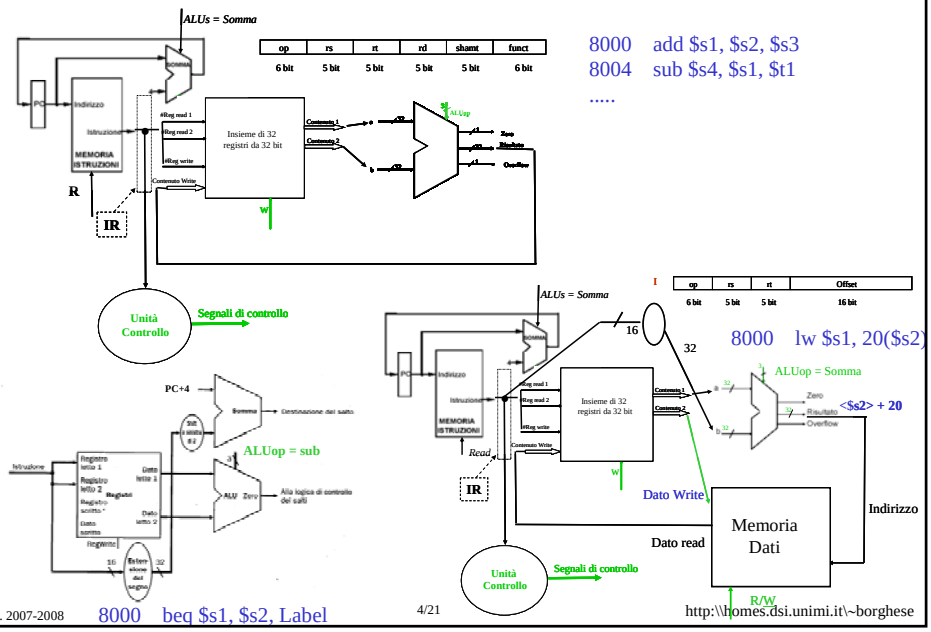
	31-26	25-21	20-16	15-11	10-6	5-0
R	op	rs	rt	rd	shamt	funct

	31-26	25-21	20-16	15-0
lw/sw	35/43	rs	rt	offset

	31-26	25-21	20-16	15-0
beq	4	rs	rt	indirizzo

Architettura RISC:
Campo Op sempre contenuto nei primi 6 bit, inseguito Op[5-0].
Registri da leggere, rs ed rt, in posizione 25-21 e 20-16. Vengono sempre letti dal Register File.
Registro base per lettura / scrittura, rs, sempre in posizione 25-21.
Offset sempre in posizione 15-0.
Registro destinazione, rd (15-11) per le istruzioni di tipo R, rt (20-16) per le istruzioni lw.

CPU per l'esecuzione di istruzioni di tipo R/lw/beq



Osservazioni

Il ciclo di esecuzione di un'istruzione si compie in un unico ciclo di clock.

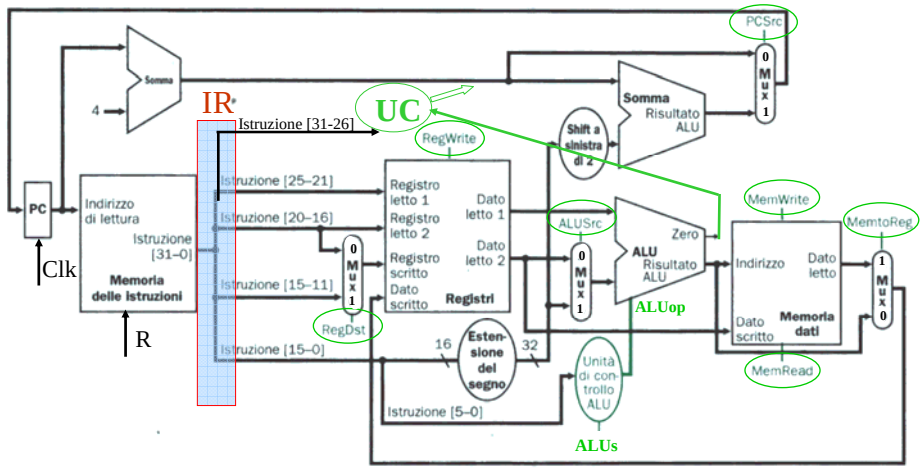
⇩

Ogni unità funzionale può essere utilizzata 1 sola volta.

⇩

Duplicazione Memoria: Memoria dati e memoria istruzioni.
Triplicazione ALU: 3 ALU: 2 sommatori + 1 general purpose.

Schema generale (lw/sw + R + beq)



Controllore della ALU: ALUs = f(..)
Controllore della CPU: Segnali_Controllo = f(OpCode)
W = Clock

Sommario

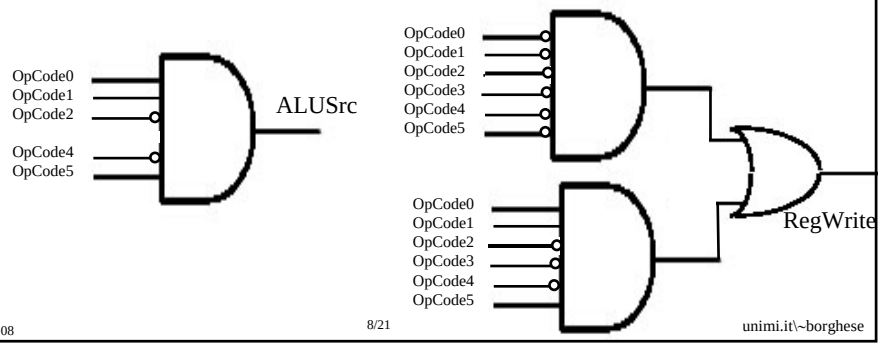
CPU che gestisce istruzioni di tipo R, lw/sw, branch

Unità di Controllo Principale

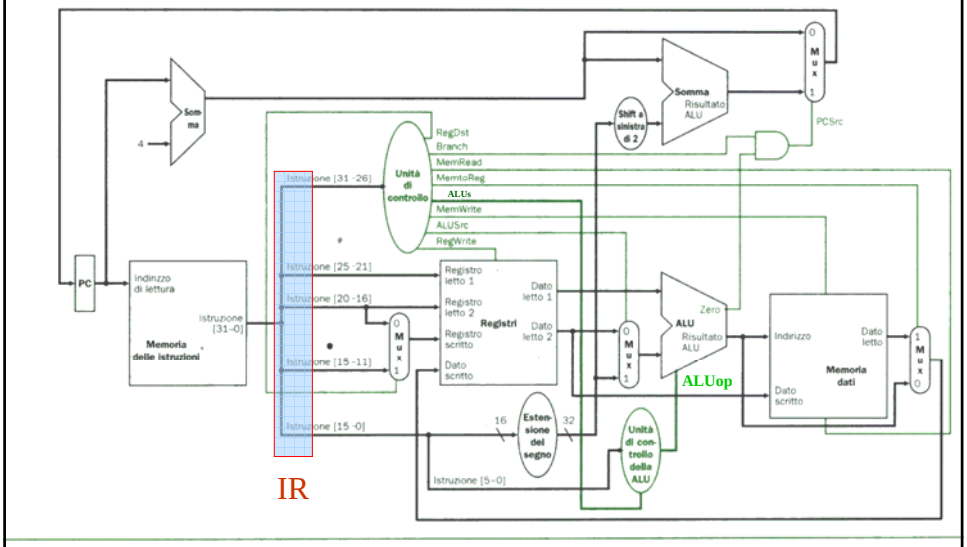
Controllore della ALU

Controllo del data-path

Istruzione (OpCode)	RegDst	ALUSrc	Memto Reg	Reg Write	Mem Read	Mem Write	Branch	ALUs
R (000000)	1	0	0	1	X	0	0	10
lw (100011)	0	1	1	1	1	0	0	00
sw (101011)	x	1	x	0	0	1	0	00
beq (000100)	x	0	x	0	X	0	1	01



CPU + UC



Segnali di controllo su 1 bit

Nome del segnale	Effetto quando è negato	Effetto quando è affermato
RegDst	Il numero del registro destinazione proviene dal campo rt (R2, bit 20-16)	Il numero del registro destinazione proviene dal campo rd (bit 15-11)
RegWrite	Nessuno	Nel registro specificato all'ingresso registro scritto del Register File, viene scritto il valore presente all'ingresso Dato Scritto
ALUSrc	Il secondo operando della ALU proviene dalla seconda uscita in lettura del Register File	Il secondo operando della ALU è la versione estesa (con segno) del campo offset
Branch	Il valore del PC viene sostituito dall'uscita del sommatore che calcola PC+4 (condizionato all'uscita di ALU)	Il valore del PC viene sostituito dall'uscita del sommatore che calcola la destinazione del salto (condizionato all'uscita di ALU)
MemRead	Nessuno	Il contenuto della cella di memoria dati indirizzata dal MAR è posto nel MDR
MemWrite	Nessuno	Il contenuto in ingresso al MDR, viene memorizzato nella cella il cui indirizzo è caricato nel MAR
MemtoReg	Il valore inviato all'ingresso Dato al Register File proviene dalla ALU	Il valore inviato all'ingresso Dato al Register File proviene dalla memoria

L'istruzione jump

J	Op Code = 2	Indirizzo

6 bit 26 bit

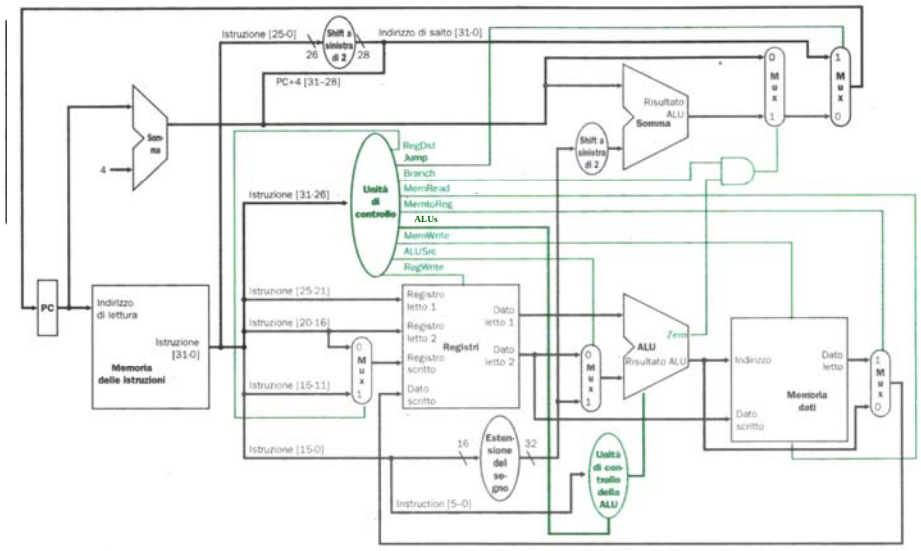
j 80040

L'indirizzo di salto sarà determinato in due passi:
A) Calcolo di Indirizzo = Indirizzo * 4.
B) Deteminazione dell'indirizzo di salto come:

Base (PC)	0100 1000 0011 0001 1011 1011 1011 10 11 +
Nuovo indirizzo	1000 0110 0111 0000 0000 0001 00 00 =
Indirizzo salto	0100 1000 0110 0111 0000 0000 0001 00 00

L'indirizzo è un numero positivo (posizione in memoria assoluta).

CPU + UC completa (aggiunta di jump)



Controllo del data-path

Istruzione (OpCode)	Reg Dst	ALU Src	Memto Reg	Reg Write	Mem Read	Mem Write	Branch	ALUs	Jump
R (000000)	1	0	0	1	X	0	0	10	0
lw (100011)	0	1	1	1	1	0	0	00	0
sw (101011)	x	1	x	0	0	1	0	00	0
beq (000100)	x	0	x	0	X	0	1	01	0
J (000010)	x	x	x	0	X	0	0	xx	1

Segnali di controllo su 1 bit

Nome del segnale	Effetto quando è negato	Effetto quando è affermato
RegDst	Il numero del registro destinazione proviene dal campo rt (R2, bit 20-16)	Il numero del registro destinazione proviene dal campo rd (bit 15-11)
RegWrite	Nessuno	Nel registro specificato all'ingresso registro scritto del Register File, viene scritto il valore presente all'ingresso Dato Scritto
ALUSrc	Il secondo operando della ALU proviene dalla seconda uscita in lettura del Register File	Il secondo operando della ALU è la versione estesa (con segno) del campo offset
Branch	Il valore del PC viene sostituito dall'uscita del sommatore che calcola PC+4 (condizionato all'uscita di ALU)	Il valore del PC viene sostituito dall'uscita del sommatore che calcola la destinazione del salto (condizionato all'uscita di ALU)
MemRead	Nessuno	Il contenuto della cella di memoria dati indirizzata dal MAR è posto nel MDR
MemWrite	Nessuno	Il contenuto in ingresso al MDR, viene memorizzato nella cella il cui indirizzo è caricato nel MAR
MemtoReg	Il valore inviato all'ingresso Dato al Register File proviene dalla ALU	Il valore inviato all'ingresso Dato al Register File proviene dalla memoria
Jump	Il valore del PC viene preso il PC è quello della branch oppure PC+4	Il valore del PC viene impostato al valore ottenuto dal campo dato della jump

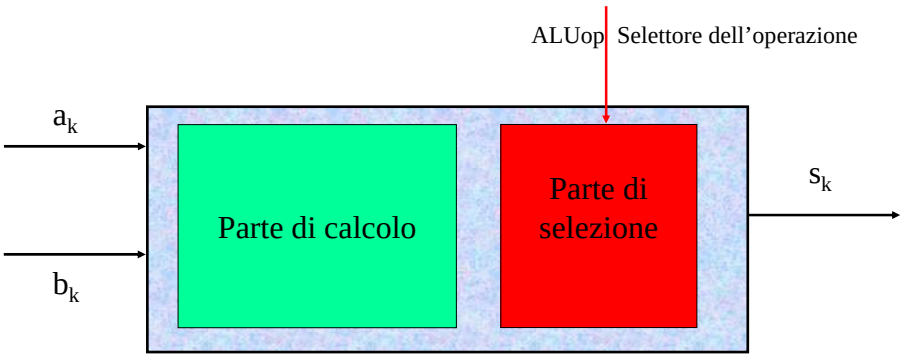

Sommario

CPU che gestisce istruzioni di tipo R, lw/sw, branch

Unità di Controllo Principale

Controllore della ALU


Struttura a 2 livelli di una ALU

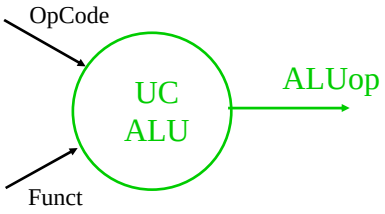


Le operazioni consentite dalla ALU (selezionate tramite ALUop):

and	000
or	001
add	010
sub	110
slt	111

UC e ALU

Data l'istruzione, l'UC deve inviare il comando opportuno alla ALU.
Campo Op Code
Campo Funct



Le operazioni consentite dalla ALU:
- and 000
- or 001
- add 010
- sub 110
- slt 111

Input: 6 bit

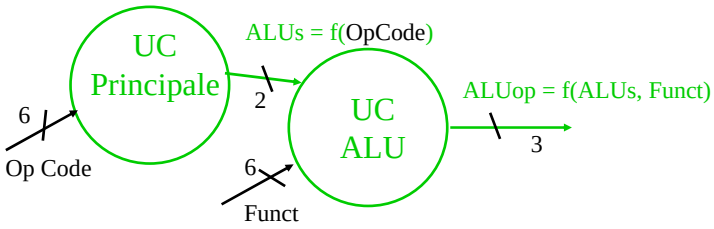
Quali operazioni devono essere eseguite per le diverse istruzioni:
- R -> Dipende dal campo funct
- lw -> Somma
- sw -> Somma
- beq -> Differenza

ALUop = f(OpCode, Funct)

Output: 3 bit

Controllo gerarchico

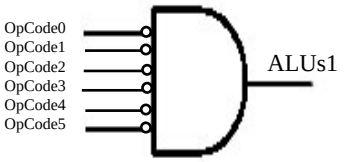
Le operazioni consentite dalla ALU:
- and 000
- or 001
- add 010
- sub 110
- slt 111

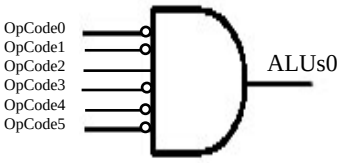


If (OpCode == R) then
Funct → ALUop
Else
OpCode → ALUop

Controllo della ALU

Istr	OpCode						ALUs	
lw	1	0	0	0	1	1	**0**	**0**
sw	1	0	1	0	1	1	**0**	**0**
beq	0	0	0	1	0	0	**0**	**1**
add	0	0	0	0	0	0	**1**	**0**
sub	0	0	0	0	0	0	**1**	**0**
and	0	0	0	0	0	0	**1**	**0**
or	0	0	0	0	0	0	**1**	**0**
slt	0	0	0	0	0	0	**1**	**0**





Sintetizzo i 2 bit come SOP

ALUs = f(OpCode)

Controllo della ALU

Istr	OpCode						ALUs		Funct						ALUop		
lw	1	0	0	0	1	1	**0**	**0**	x	x	x	x	x	x	**0**	**1**	**0**
sw	1	0	1	0	1	1	**0**	**0**	x	x	x	x	x	x	**0**	**1**	**0**
beq	0	0	0	1	0	0	**0**	**1**	x	x	x	x	x	x	**1**	**1**	**0**
add	0	0	0	0	0	0	**1**	**0**	1	0	0	0	0	0	**0**	**1**	**0**
sub	0	0	0	0	0	0	**1**	**0**	1	0	0	0	1	0	**1**	**1**	**0**
and	0	0	0	0	0	0	**1**	**0**	1	0	0	1	0	0	**0**	**0**	**0**
or	0	0	0	0	0	0	**1**	**0**	1	0	0	1	0	1	**0**	**0**	**1**
slt	0	0	0	0	0	0	**1**	**0**	1	0	1	0	1	0	**1**	**1**	**1**

SOP

ALUop = f(ALUs, Funct)

Sommario

CPU che gestisce istruzioni di tipo R, lw/sw, branch

Controllore della ALU

Unità di Controllo Principale