



Architettura degli Elaboratori I

Corso di Laurea Triennale in Informatica

Università degli Studi di Milano

Dipartimento di Informatica "Giovanni Degli Antoni"

Edizione 2 (Cognomi H-Z), A.A. 2024-2025, Nicola.Basilico@unimi.it

Riepilogo segnali di controllo

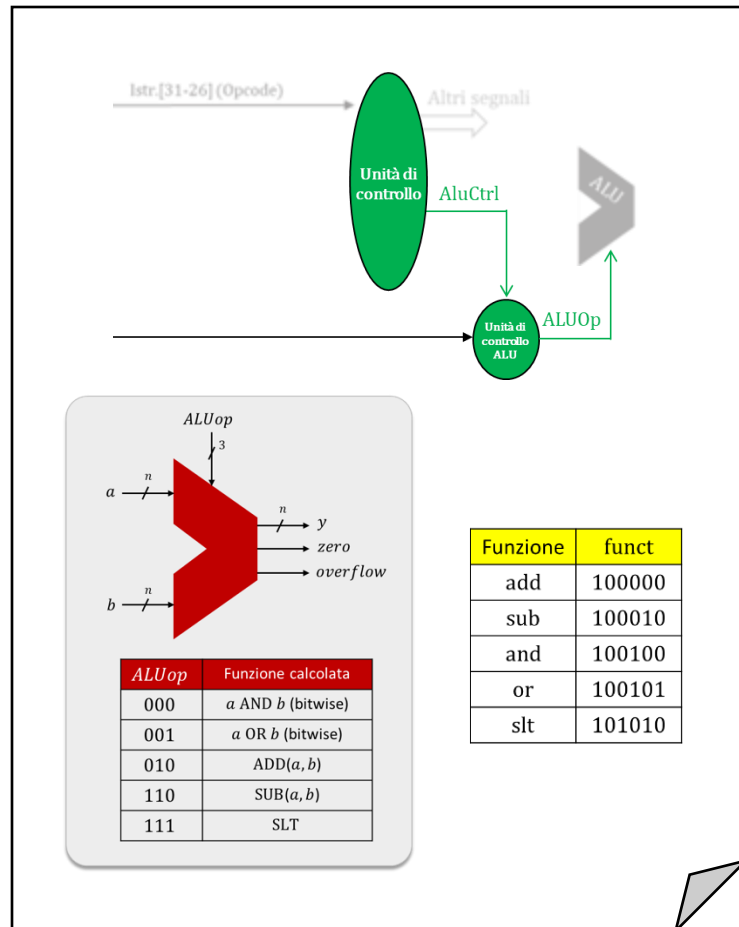
Segnali di lettura/scrittura: hanno un effetto solo quando posti a **1**

RegWrite	Attiva la scrittura nel Register File: il dato D viene scritto nel registro indirizzato da WriteAddr
MemRead	Attiva la lettura in memoria : sull'uscita DataRead viene posto il valore della parola di memoria che ha indirizzo Addr
MemWrite	Attiva la scrittura in memoria : il dato D viene scritto nella parola di memoria che ha indirizzo Addr

Segnali di selezione

	Se posto a 0	Se posto a 1
ALUSrc	Il secondo operando della ALU viene dal Register File: uscita DataRead ₂	Il secondo operando della ALU è l'estensione su 32 bit del campo IMMEDIATO dell'istruzione (bit 15-0)
MemToReg	Il dato D da scrivere nel Register File proviene dall'uscita della ALU	Il dato D da scrivere nel Register File dall'uscita di lettura DataRead della memoria
RegDst	L'indirizzo del registro destinazione è estratto dal campo r_t dell'istruzione (bit 20-16)	L'indirizzo del registro destinazione è estratto dal campo r_d dell'istruzione (bit 15-11)
Branch AND Zero	L'indirizzo da scrivere in PC è $PC + 4$ (a meno che Jump non sia posto a 1)	L'indirizzo da scrivere in PC è $PC + 4$ a cui si somma il valore del campo IMMEDIATO dell'istruzione (bit 15-0) esteso su 32 bit e moltiplicato per 4
Jump	L'indirizzo da scrivere in PC è $PC + 4$ (a meno che Branch AND Zero non sia posto a 1)	L'indirizzo da scrivere in PC ottenuto aggiungendo al valore del campo PSEUDO-INDIRIZZO dell'istruzione (bit 0-15) due zeri a destra e i 4 bit più significativi di PC a sinistra

Unità di controllo ALU



- **AluCtrl**: comunica all'unità di controllo della ALU quale tipo di istruzione la CPU sta eseguendo, ci interessano solo le istruzioni aritmetico-logiche, gli accessi a memoria e la branch; con jump la ALU non è necessaria
- **ALUOp**: configura la ALU perché esegua una delle operazioni supportate

AluCtrl	ALUOp
Istruzione Aritmetico-Logica (tipo R)	La ALU fa l'operazione indicata da funct
Accesso a memoria (lw o sw)	La ALU fa una somma
Branch on equal	La ALU fa una differenza

- Iniziamo scegliendo una codifica per **AluCtrl** il cui valore dipende da Opcode

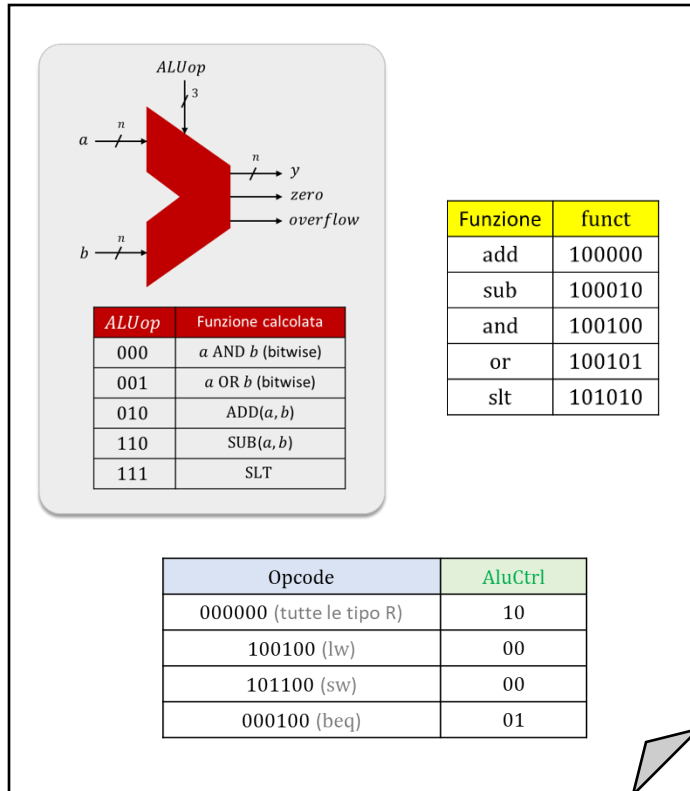
Opcode	AluCtrl
000000 (tutte le tipo R)	10
100011 (lw)	00
101011 (sw)	00
000100 (beq)	01

AluCtrl = 10 → Aritmetico-logica

AluCtrl = 00 → Accesso a memoria

AluCtrl = 01 → Branch

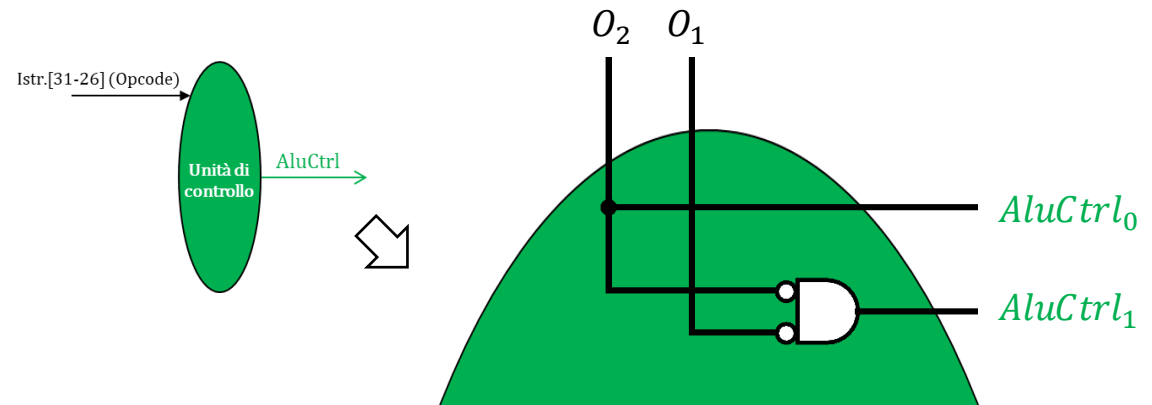
Unità di controllo ALU



	Opcode						AluCtrl	
	<i>O</i> ₅	<i>O</i> ₄	<i>O</i> ₃	<i>O</i> ₂	<i>O</i> ₁	<i>O</i> ₀	AluCtrl ₁	AluCtrl ₀
A-R →	0	0	0	0	0	0	1	0
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
j →	0	0	0	0	1	0	<i>x</i>	<i>x</i>

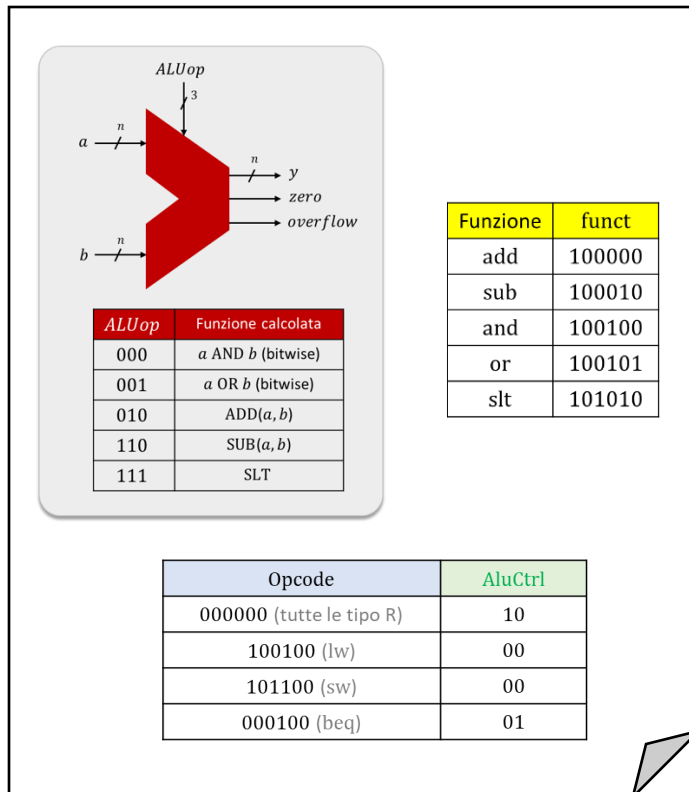
$$\text{AluCtrl}_0 = O_2$$

$$\text{AluCtrl}_1 = \overline{O_2} \overline{O_1}$$



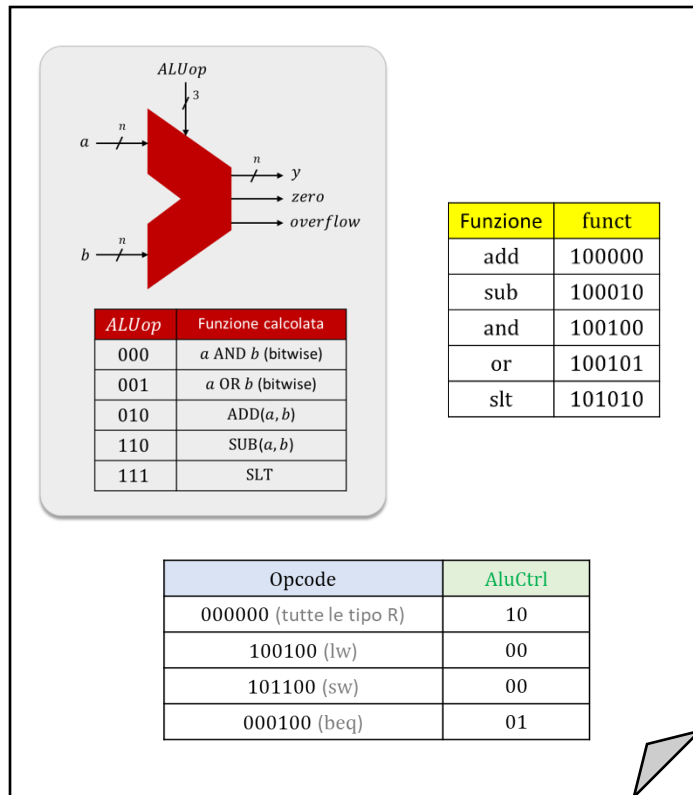
Unità di controllo ALU

- Ora possiamo sintetizzare l'unità di controllo della ALU che, dato il segnale di controllo **AluCtrl** e funct, determina **ALUOp**



	<i>AluCtrl</i>		funct						<i>ALUOp</i>		
	<i>AluCtrl</i> ₁	<i>AluCtrl</i> ₀	<i>f</i> ₅	<i>f</i> ₄	<i>f</i> ₃	<i>f</i> ₂	<i>f</i> ₁	<i>f</i> ₀	<i>ALUOp</i> ₂	<i>ALUOp</i> ₁	<i>ALUOp</i> ₀
lw	0	0	<i>x</i>	<i>x</i>	<i>x</i>	<i>x</i>	<i>x</i>	<i>x</i>	0	1	0
sw	0	0	<i>x</i>	<i>x</i>	<i>x</i>	<i>x</i>	<i>x</i>	<i>x</i>	0	1	0
beq	0	1	<i>x</i>	<i>x</i>	<i>x</i>	<i>x</i>	<i>x</i>	<i>x</i>	1	1	0
add	1	0	1	0	0	0	0	0	0	1	0
sub	1	0	1	0	0	0	1	0	1	1	0
and	1	0	1	0	0	1	0	0	0	0	0
or	1	0	1	0	0	1	0	1	0	0	1
slt	1	0	1	0	1	0	1	0	1	1	1

Unità di controllo ALU



	AluCtrl		funcnt							ALUOp		
	AluCtrl ₁	AluCtrl ₀	f ₅	f ₄	f ₃	f ₂	f ₁	f ₀		ALUOp ₂	ALUOp ₁	ALUOp ₀
lw	0	0	x	x	x	x	x	x		0	1	0
sw	0	0	x	x	x	x	x	x		0	1	0
beq	0	1	x	x	x	x	x	x		1	1	0
add	1	0	1	0	0	0	0	0		0	1	0
sub	1	0	1	0	0	0	1	0		1	1	0
and	1	0	1	0	0	1	0	0		0	0	0
or	1	0	1	0	0	1	0	1		0	0	1
slt	1	0	1	0	1	0	1	0		1	1	1

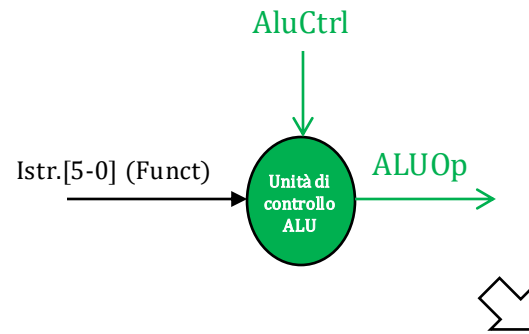
$$AluCtrl_i \rightarrow C_i$$

$$ALUOp_0 = C_1 \overline{C_0} f_5 \overline{f_4} (\overline{f_3} f_2 \overline{f_1} f_0 + f_3 \overline{f_2} f_1 \overline{f_0})$$

$$ALUOp_1 = (\overline{C_1} + C_0) + (\overline{f_5} + f_4 + f_3 + \overline{f_2} + f_1)$$

$$ALUOp_2 = \overline{C_1} C_0 + C_1 \overline{C_0} (f_5 \overline{f_4} \overline{f_2} f_1 \overline{f_0})$$

Unità di controllo ALU

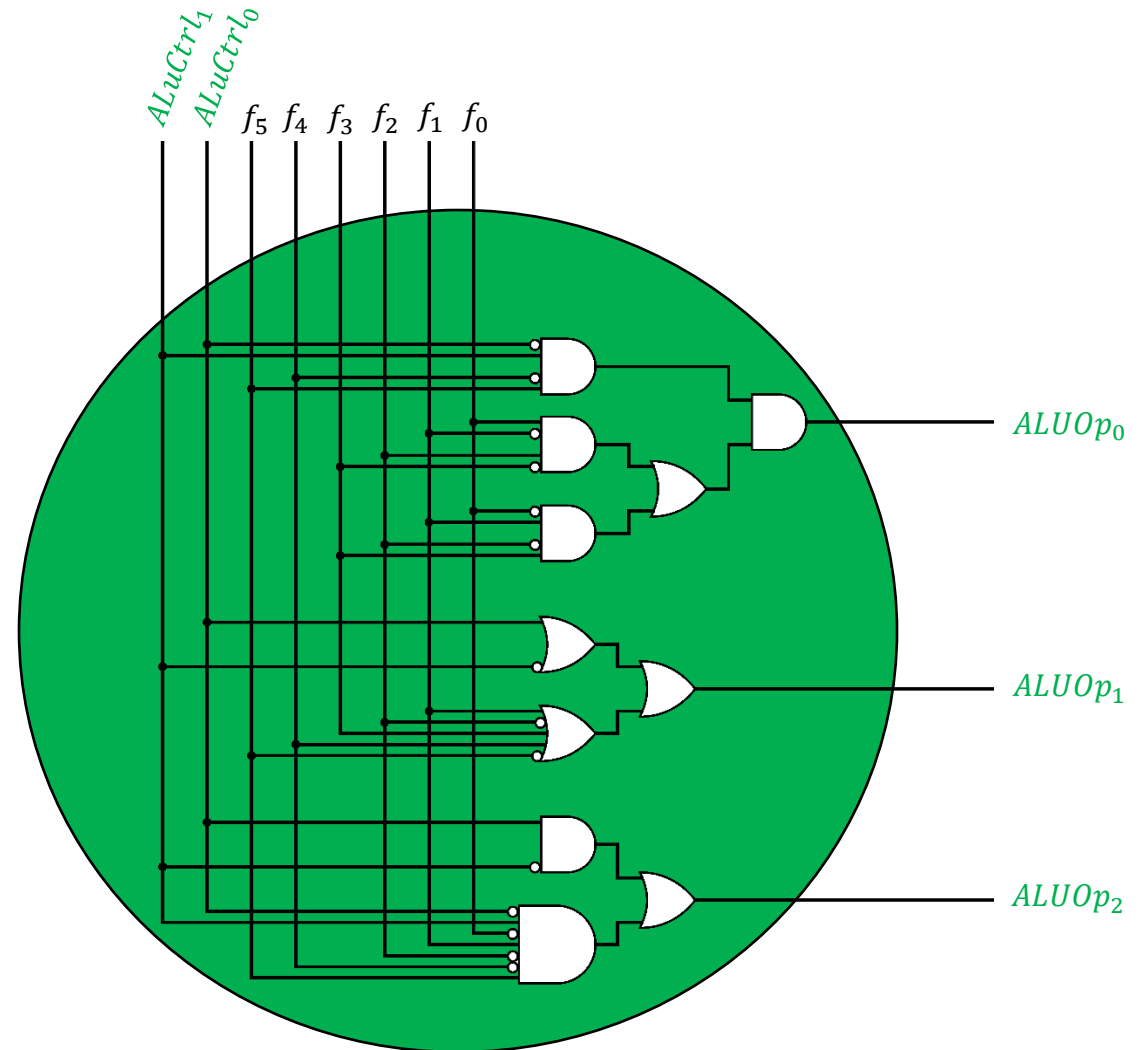


	<i>AluCtrl</i>		funct							<i>ALUOp</i>		
	<i>AluCtrl₁</i>	<i>AluCtrl₀</i>	<i>f₅</i>	<i>f₄</i>	<i>f₃</i>	<i>f₂</i>	<i>f₁</i>	<i>f₀</i>		<i>ALUOp₂</i>	<i>ALUOp₁</i>	<i>ALUOp₀</i>
lw	0	0	x	x	x	x	x	x		0	1	0
sw	0	0	x	x	x	x	x	x		0	1	0
beq	0	1	x	x	x	x	x	x		1	1	0
add	1	0	1	0	0	0	0	0		0	1	0
sub	1	0	1	0	0	0	1	0		1	1	0
and	1	0	1	0	0	1	0	0		0	0	0
or	1	0	1	0	0	1	0	1		0	0	1
slt	1	0	1	0	1	0	1	0		1	1	1

$$ALUOp_0 = C_1 \overline{C_0} f_5 \overline{f_4} (\overline{f_3} f_2 \overline{f_1} f_0 + f_3 \overline{f_2} f_1 \overline{f_0})$$

$$ALUOp_1 = (\overline{C_1} + C_0) + (\overline{f_5} + f_4 + f_3 + \overline{f_2} + f_1)$$

$$ALUOp_2 = \overline{C_1} C_0 + C_1 \overline{C_0} (f_5 \overline{f_4} \overline{f_2} f_1 \overline{f_0})$$



Unità di controllo principale

	O_5	O_4	O_3	O_2	O_1	O_0	Reg Write	Reg Dst	ALU Src	Mem Read	Mem Write	Mem ToReg	Branch	Jump
A-R	0	0	0	0	0	0	1	1	0	0	0	0	0	0
lw	1	0	0	0	1	1	1	0	1	1	0	1	0	0
sw	1	0	1	0	1	1	0	x	1	0	1	x	0	0
beq	0	0	0	1	0	0	0	x	0	0	0	x	1	0
j	0	0	0	0	1	0	0	x	x	0	0	x	x	1

$$\text{RegWrite} = \overline{O_2}\overline{O_1} + O_5\overline{O_3}$$

$$\text{MemRead} = O_5\overline{O_3}$$

$$\text{Branch} = \overline{O_5}O_2$$

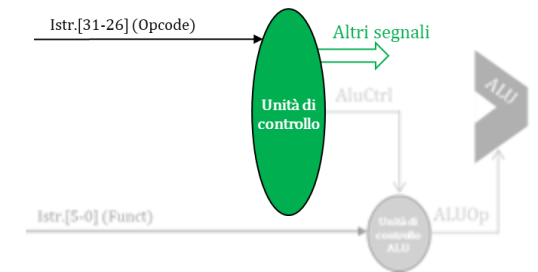
$$\text{RegDst} = \overline{O_2}\overline{O_1}$$

$$\text{MemWrite} = O_3$$

$$\text{Jump} = O_1\overline{O_0}$$

$$\text{ALUSrc} = O_5$$

$$\text{MemToReg} = O_5\overline{O_3}$$



Unità di controllo finale

