Architettura degli Elaboratori

Il Processore:
l'Unità di Controllo della ALU

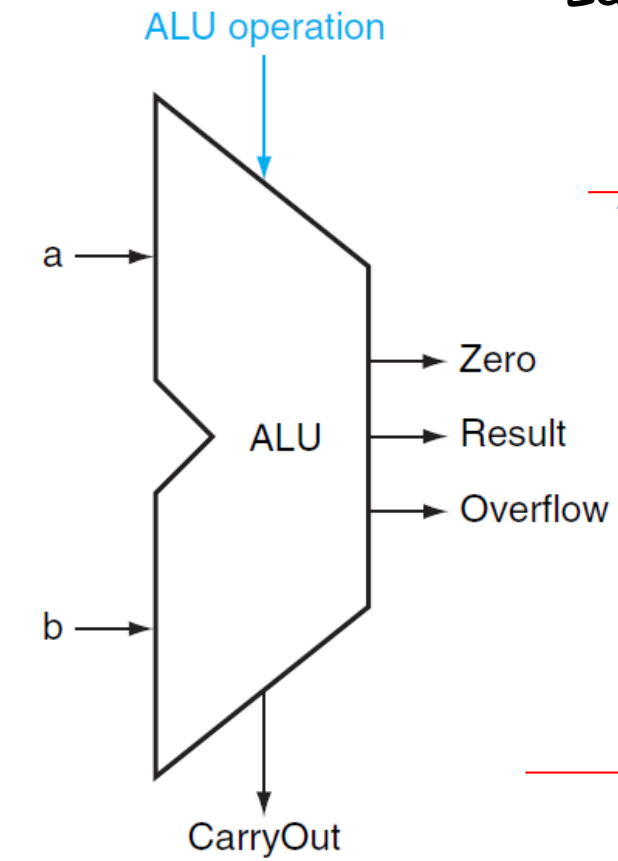
Barbara Masucci
UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA
DIPARTIMENTO DI ECCELLENZA

Punto della situazione

- Abbiamo visto come costruire l'Unità di Elaborazione Dati (Datapath) del processore MIPS

- Nella lezione di oggi e nella prossima, aggiungeremo all'Unità di Elaborazione Dati costruita la relativa Unità di Controllo
  - Inizieremo oggi con la costruzione di una piccola unità, detta Unità di Controllo della ALU
    - Genera i segnali di controllo per la ALU
  - Nella prossima lezione costruiremo l'Unità di Controllo principale
    - Genera, oltre a tutti gli altri segnali di controllo, anche gli input per l'unità di controllo della ALU

La ALU del MIPS



La ALU prevede le 6 seguenti combinazioni dei suoi 4 segnali di controllo

~~Ainvert (1 bit),~~ Bnegate (1 bit), Operation (2bit)

ALU control lines	Function
0000	AND
0001	OR
0010	add
0110	subtract
0111	set on less than
~~1100~~	~~NOR~~

Ignoriamo l'istruzione NOR (e il segnale Ainvert) perché non fa parte del set di istruzioni considerate

La ALU del MIPS

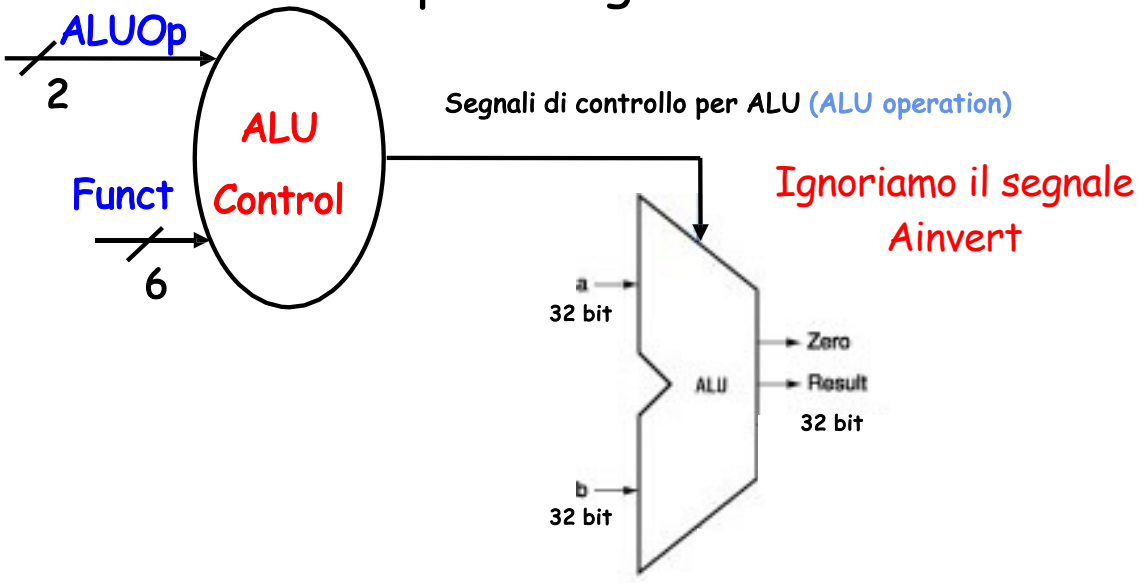
- La ALU deve eseguire
  - Una somma, per calcolare l'indirizzo di memoria, quando sono eseguite le istruzioni lw e sw (formato I)
  - Una sottrazione, per eseguire l'istruzione beq (formato I)

  - Una delle 5 operazioni (AND, OR, add, sub, slt) in funzione del campo funct, per le istruzioni di tipo R

La ALU Control

Per generare i segnali di controllo per la ALU viene utilizzata una piccola unità di controllo (ALU Control o Contollore ALU) che

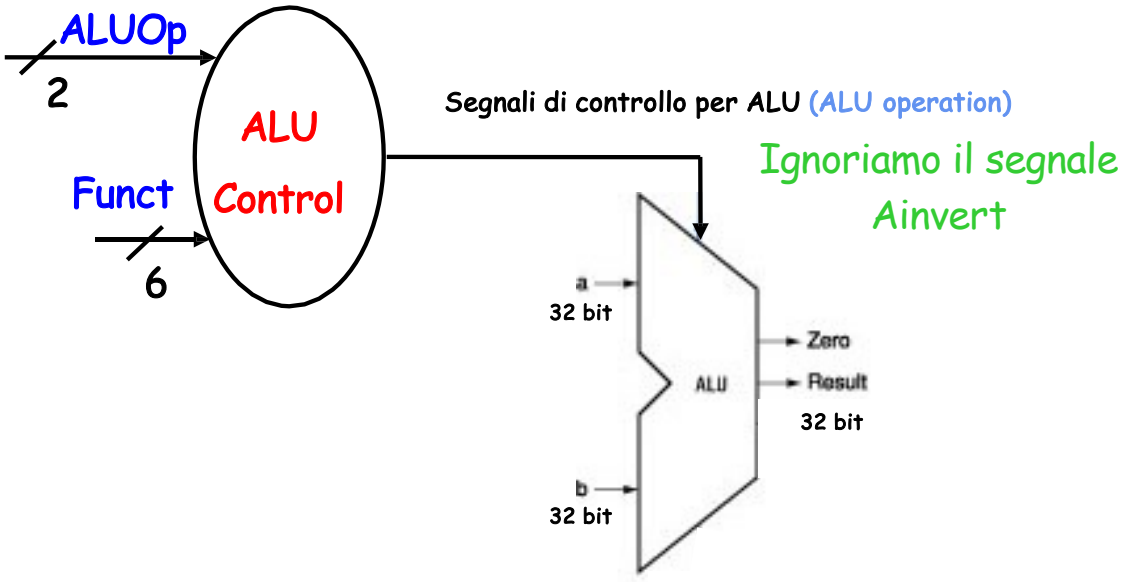
- Riceve in ingresso il campo funct dell'istruzione e un campo di controllo su 2 bit, detto ALUOp, che indica l'operazione da eseguire
- Fornisce in output i segnali di controllo della ALU



La ALU Control

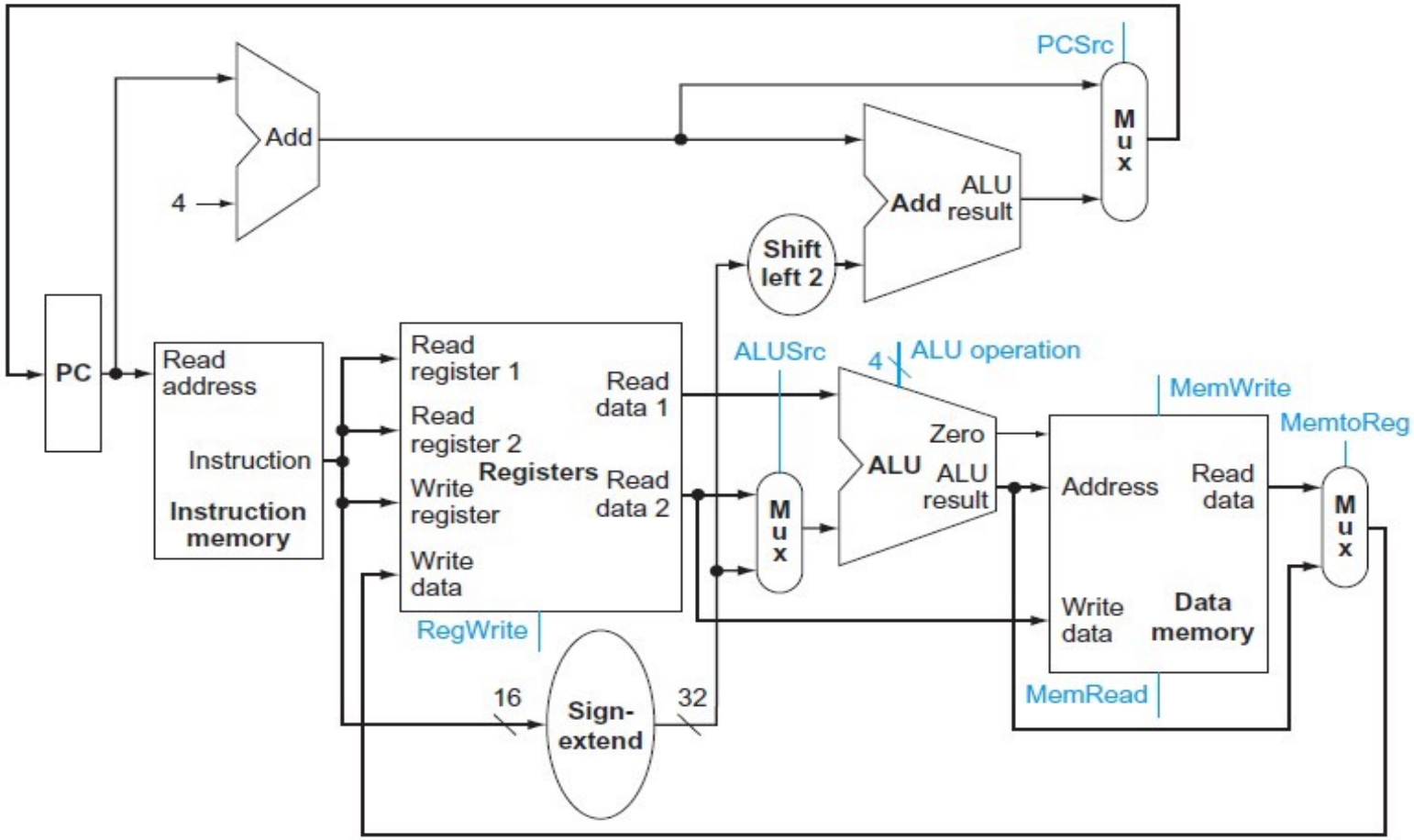
Il segnale di input ALUOp per la ALU Control varia a seconda dell'istruzione da eseguire

- Per lw e sw, ALUOp=00 (operazione per l'ALU: somma)
- Per beq, ALUOp=01 (operazione per l'ALU: sottrazione)
- Per istruzioni in formato R, ALUOp=10 (operazione per l'ALU: dipende dal campo funct)



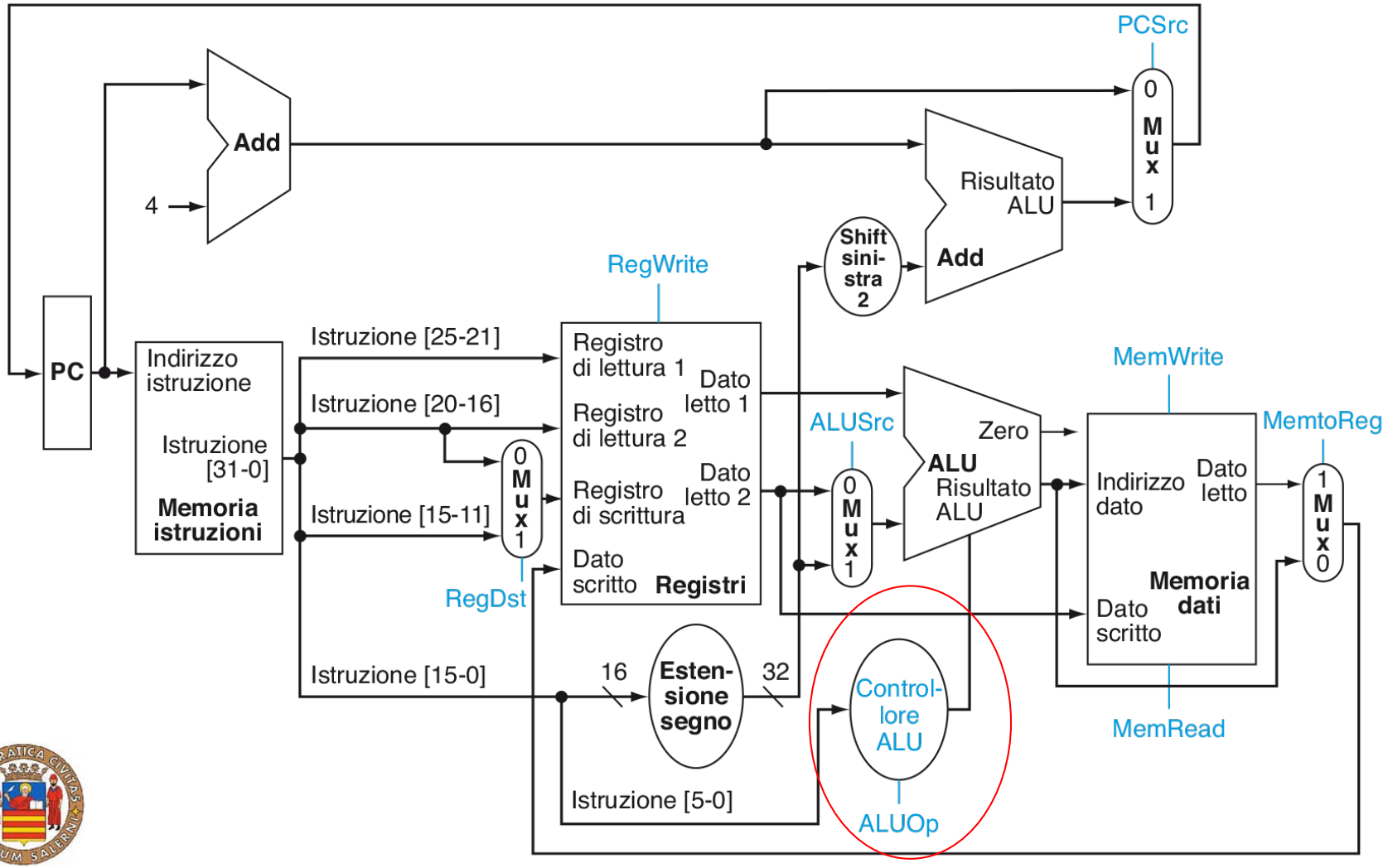
Implementazione a ciclo singolo

All'Unità di Elaborazione costruita...



Implementazione a ciclo singolo

...aggiungiamo la ALU Control



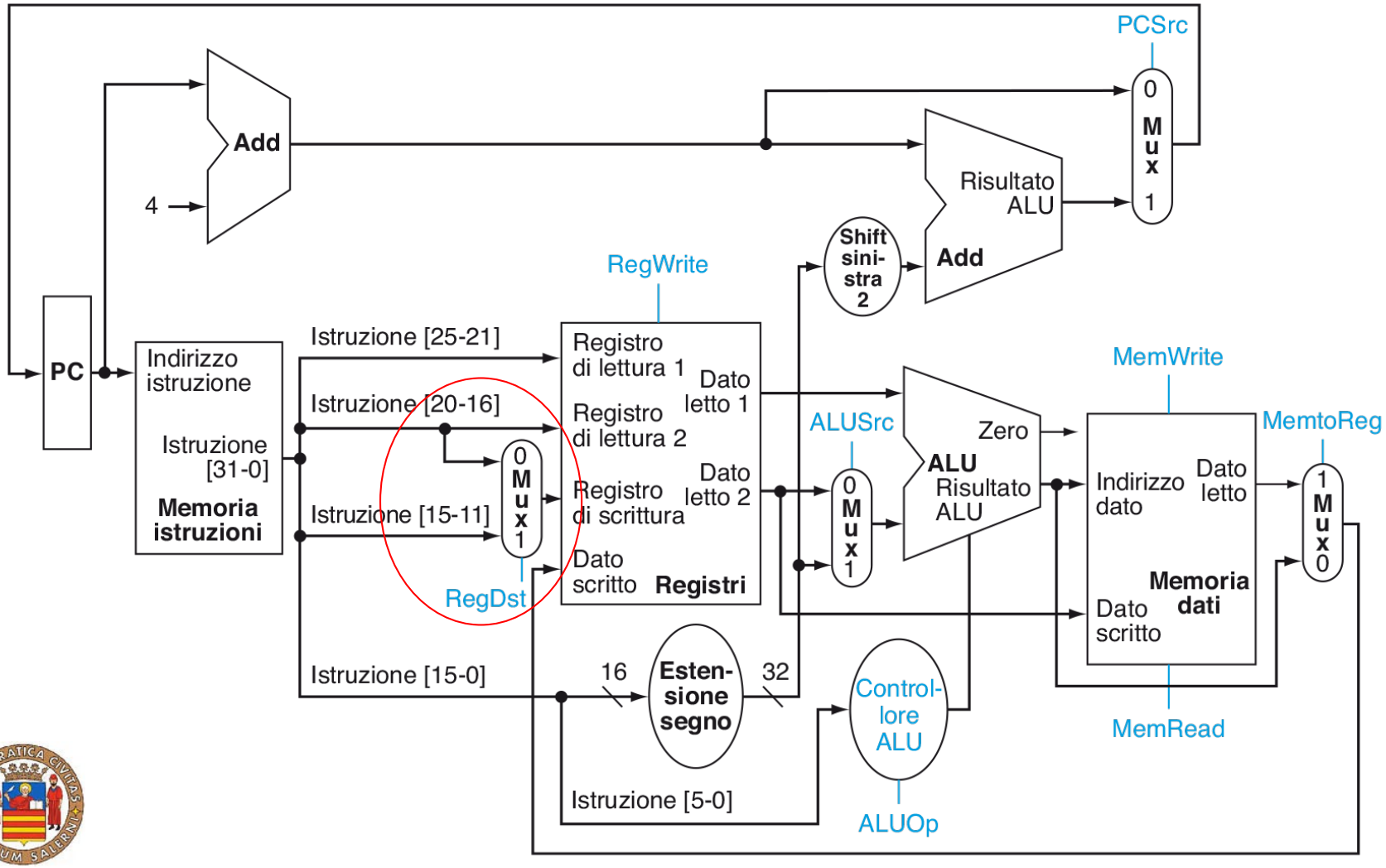
Implementazione a ciclo singolo

- In realtà nella slide precedente la ALU Control non è l'unica aggiunta
- E' stato aggiunto anche un multiplexer per selezionare il registro di scrittura (target)
  - La necessità sorge dal fatto che il registro target si trova in posizioni diverse a seconda del tipo dell'istruzione
    - Posizione dei bit da 15 a 11 per istruzioni di tipo R

    - Posizione dei bit da 20 a 16 per l'istruzione lw

registro target

Implementazione a ciclo singolo



La ALU Control

- L'ultima colonna mostra i valori da assegnare ai segnali di controllo della ALU in funzione di
  - ALUOp (2 bit)
  - Campo funct (6 bit)

Mai 11

Inizia sempre per 10
Non finisce mai per 11

Codice operativo istruzione	ALUOp	Operazione eseguita dall'istruzione	Campo funzione	Operazione dell'ALU	Ingresso di controllo alla ALU
Lw	00	load di 1 parola	XXXXXX	somma	0010
Sw	00	store di 1 parola	XXXXXX	somma	0010
Branch equal	01	salto condizionato all'uguaglianza	XXXXXX	sottrazione	0110
Tipo R	10	somma	100000	somma	0010
Tipo R	10	sottrazione	100010	sottrazione	0110
Tipo R	10	AND	100100	AND	0000
Tipo R	10	OR	100101	OR	0001
Tipo R	10	set less than	101010	set less than	0111

Quando ALUOp=00 oppure 01, l'operazione che la ALU deve eseguire non dipende dal campo funct: tale valore è indifferente e viene indicato con XXXXXX

La ALU Control

Come implementare la corrispondenza tra i 2 bit di ALUOp e i 6 bit del campo funct con i 4 bit dei segnali di controllo della ALU?

- Innanzitutto, il campo funct viene considerato solo quando ALUOp=10
- Inoltre, ci interessa solo un piccolo sottoinsieme dei 2^6 valori possibili che possono essere assunti dal campo funct
- Possiamo utilizzare un piccolo circuito logico che riconosca il sottoinsieme dei valori possibili e imposti i bit di controllo della ALU in modo corretto
  - Per costruire questo circuito, scriviamo innanzitutto la tavola di verità della funzione Operation (4 bit, di cui consideriamo solo gli ultimi tre)
  - Poiché la tavola ha 2+6=8 variabili, ci sono 2^8=256 righe, ma molte di queste possono essere omesse perché corrispondenti ad input che non possono presentarsi (indifferenti)

Tavola di verità per Operation

	ALUOp		Funct field						Operation
	ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0	
Somma per lw, sw	0	0	X	X	X	X	X	X	0010
Sub per beq	X	1	X	X	X	X	X	X	0110
add	1	X	X	X	0	0	0	0	0010
sub	1	X	X	X	0	0	1	0	0110
and	1	X	X	X	0	1	0	0	0000
or	1	X	X	X	0	1	0	1	0001
slt	1	X	X	X	1	0	1	0	0111

Siccome ALUOp non è mai 11, la tabella contiene le combinazioni 00, X1 e 1X

Inoltre, poiché quando si utilizza il campo Funct, F5F4 è sempre 10, questi due bit sono sostituiti con XX

ALUOp=00 somma per lw e sw

ALUOp=01 sottrazione per beq

ALUOp=10 formato R spec. dal campo funct

Tavola di verità per Operation

- Dalla tavola di verità nella slide precedente possiamo omettere il primo bit di output e considerare solo gli altri tre: Operation2, Operation1, Operation0
  - Per ciascuno di questi output possiamo considerare solo le righe della tavola in corrispondenza dei quali l'output vale 1
  - Inoltre, possiamo compattare le tabelle risultanti, considerando solo gli input non indifferenti

Tavola di verità per Operation2

Input Output

ALUOp		Funct field						Operation
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0	
0	0	X	X	X	X	X	X	010
X	1	X	X	X	X	X	X	110
1	X	X	X	0	0	0	0	010
1	X	X	X	0	0	1	0	110
1	X	X	X	0	1	0	0	000
1	X	X	X	0	1	0	1	001
1	X	X	X	1	0	1	0	111

Operation2

0
1
0
1
0
0
1

La prima riga di input in cui Operation2=1 è riportata giù (con ALUOp1=X)
Le altre 2 righe di input in cui Operation2=1, fra tutte quelle che iniziano per 1X, sono tutte e sole quelle in cui F1 = 1

ALUOp		Function code fields					
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0
X	1	X	X	X	X	X	X
1	X	X	X	X	X	1	X

Tavola di verità per Operation1

Input							Output
ALUOp		Funct field					
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0	Operation
0	0	X	X	X	X	X	X	010
0	1	X	X	X	X	X	X	110
1	X	X	X	0	0	0	0	010
1	X	X	X	0	0	1	0	110
1	X	X	X	0	1	0	0	000
1	X	X	X	0	1	0	1	001
1	X	X	X	1	0	1	0	111

Le prime 2 righe di input in cui Operation1=1 sono tutte e sole quelle con ALUOp1 = 0
Le altre 3 righe di input in cui Operation1=1, fra tutte quelle che iniziano per 1X, sono tutte e sole quelle in cui F2 = 0

ALUOp		Function code fields					
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0
0	X	X	X	X	X	X	X
X	X	X	X	X	0	X	X

Tavola di verità per Operation0

Input								Output
ALUOp		Funct field						Operation
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0	
0	0	X	X	X	X	X	X	010
0	1	X	X	X	X	X	X	110
1	X	X	X	0	0	0	0	010
1	X	X	X	0	0	1	0	110
1	X	X	X	0	1	0	0	000
1	X	X	X	0	1	0	1	001
1	X	X	X	1	0	1	0	111

Fra tutte le righe con ALUOp = 1X:
la prima riga di input in cui Operation0 = 1, è l'unica con F0 = 1;
la seconda riga di input in cui Operation0 = 1 è l'unica con F3 = 1

ALUOp		Function code fields					
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0
1	X	X	X	X	X	X	1
1	X	X	X	1	X	X	X

Tavole di verità compresse per i tre bit di Operation

Tavola di verità per Operation 2

ALUOp		Function code fields					
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0
X	1	X	X	X	X	X	X
1	X	X	X	X	X	1	X

Tavola di verità per Operation 1

ALUOp		Function code fields					
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0
0	X	X	X	X	X	X	X
X	X	X	X	X	0	X	X

Tavola di verità per Operation 0

ALUOp		Function code fields					
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0
1	X	X	X	X	X	X	1
1	X	X	X	1	X	X	X

Funzioni corrispondenti

Operation2 = ALUOp0+ALUOp1· F1

ALUOp		Function code fields					
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0
X	1	X	X	X	X	X	X
1	X	X	X	X	X	1	X

Operation1 = $\overline{ALUOp1}$ + $\overline{F2}$

ALUOp		Function code fields					
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0
0	X	X	X	X	X	X	X
X	X	X	X	X	0	X	X

Operation0 = ALUOp1·F0+ALUOp1·F3 = ALUOp1·(F0+F3)

ALUOp		Function code fields					
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0
1	X	X	X	X	X	X	1
1	X	X	X	1	X	X	X

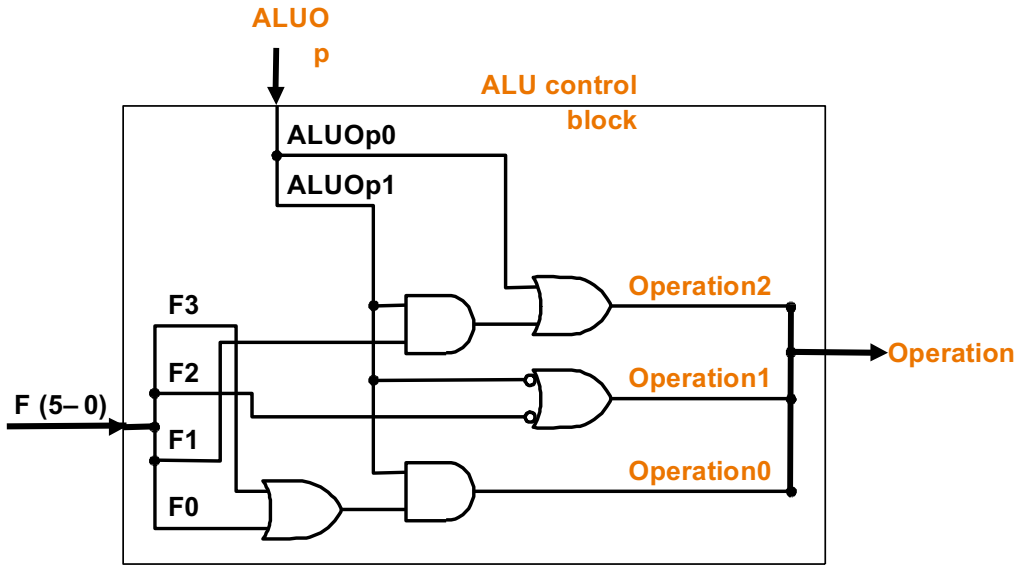
Un circuito per Operation

Ecco il circuito che realizza le tre funzioni

$$Operation2 = ALUOp0 + ALUOp1 \cdot F1$$

$$Operation1 = \overline{ALUOp1} + \overline{F2}$$

$$Operation0 = ALUOp1 \cdot F0 + ALUOp1 \cdot F3 = ALUOp1 \cdot (F0 + F3)$$



La ALU Control

- In definitiva, l'Unità di Controllo della ALU per il processore MIPS nell'implementazione a ciclo singolo
  - Riceve in ingresso il campo funct (6 bit) dell'istruzione e il segnale di controllo ALUOp (2 bit)
  - Genera i segnali di controllo Operation2, Operation1, Operation0 per la ALU

Riepilogo e riferimenti

- Unità di Controllo per la ALU
  - [PH] par. 4.4 (prima parte)
  - Appendice D.1, D.2 (ottimizzazione del circuito per Operation)