Corso di
Architettura degli Elaboratori
a.a. 2023/2024

Il livello logico digitale:
Circuiti logici digitali di base

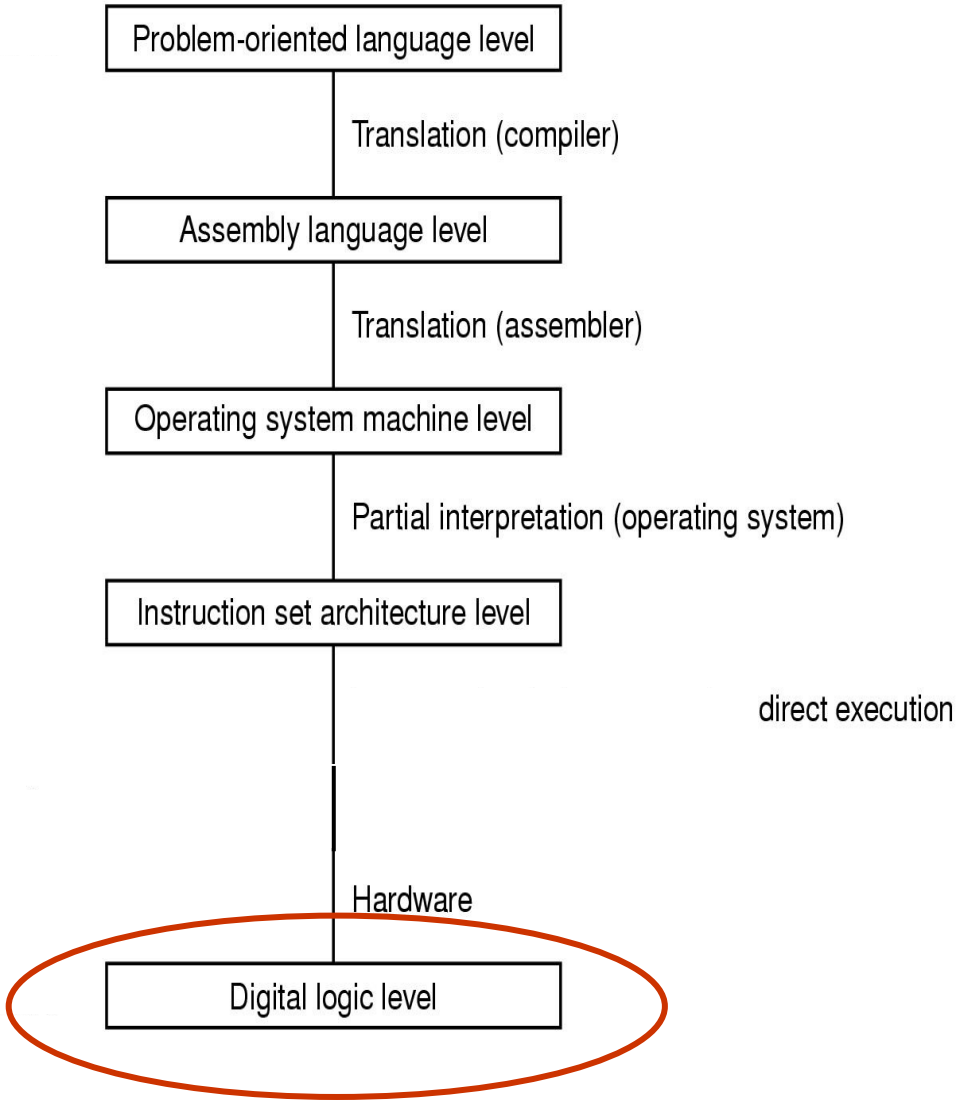
Livello della logica digitale

Livello della *Logico-Digitale*

Costituenti di base del computer:

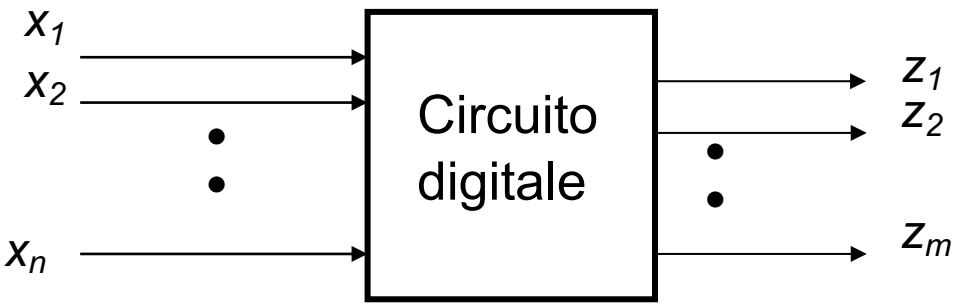
- porte
- registri
- memoria

Sotto questo livello ci sono i dispositivi (funzionamento interno delle porte: transistor)



Circuiti digitali

- Gli elementi di base con cui si sono costruiti i calcolatori si chiamano **circuiti digitali** (o reti digitali)
- Sono dispositivi che utilizzano **solo due valori logici**: 0 (segnale tra 0 e 1 volt) e 1 (segnale tra 2 e 5 volt).
  - I valori di tensione possono anche essere altri.
- Un circuito digitale trasforma segnali (binari) di **ingresso** $x_1, x_2, \ldots, x_n$ nei segnali (binari) di **uscita** $z_1, z_2, \ldots, z_m$.



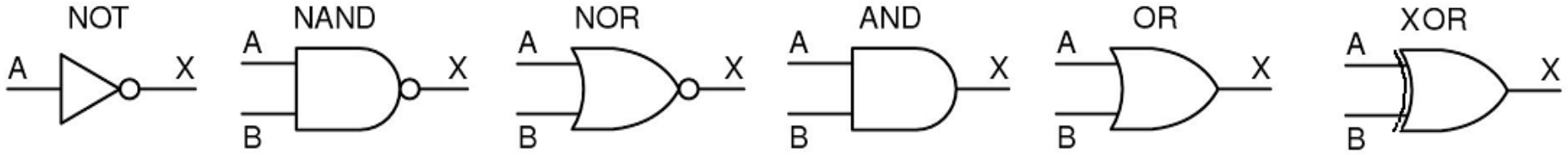
Porte logiche

- I circuiti sono detti ***combinatori*** quando l'uscita è funzione esclusivamente dell'ingresso; sono detti ***sequenziali*** quando l'uscita è funzione oltre che dell'ingresso anche di uno ***stato***.

- Gli elementi primitivi dei circuiti digitali sono chiamati ***porte logiche*** e calcolano alcune funzioni di questi segnali a due valori.

- Questi dispositivi si basano sul fatto che si può far funzionare un ***transistor*** come un interruttore binario molto veloce.

Porte logiche



A	X
0	1
1	0

(a)

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

(b)

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

(c)

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

(d)

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

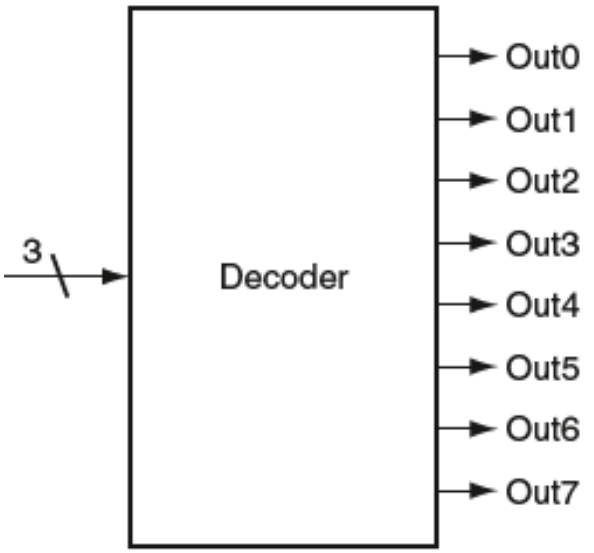
(e)

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

(f)

Circuiti combinatori: decoder

- ***Decoder***: prende un numero di *n* bit come ingresso e lo usa per selezionare (mettere a 1, asserire) una delle 2^n linee di uscita
- Può essere utilizzato per attivare una certa componente (vedi ALU più avanti), oppure un banco di memoria, ecc.
- Interpretiamo gli ingressi A B C (o I2 I1 I0) come le cifre di un numero in base 2 con A (I2) quella più significativa

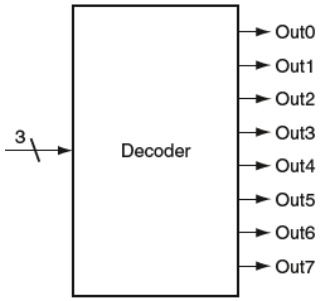


a. A 3-bit decoder

Inputs			Outputs							
I2	I1	I0	Out7	Out6	Out5	Out4	Out3	Out2	Out1	Out0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

b. The truth table for a 3-bit decoder

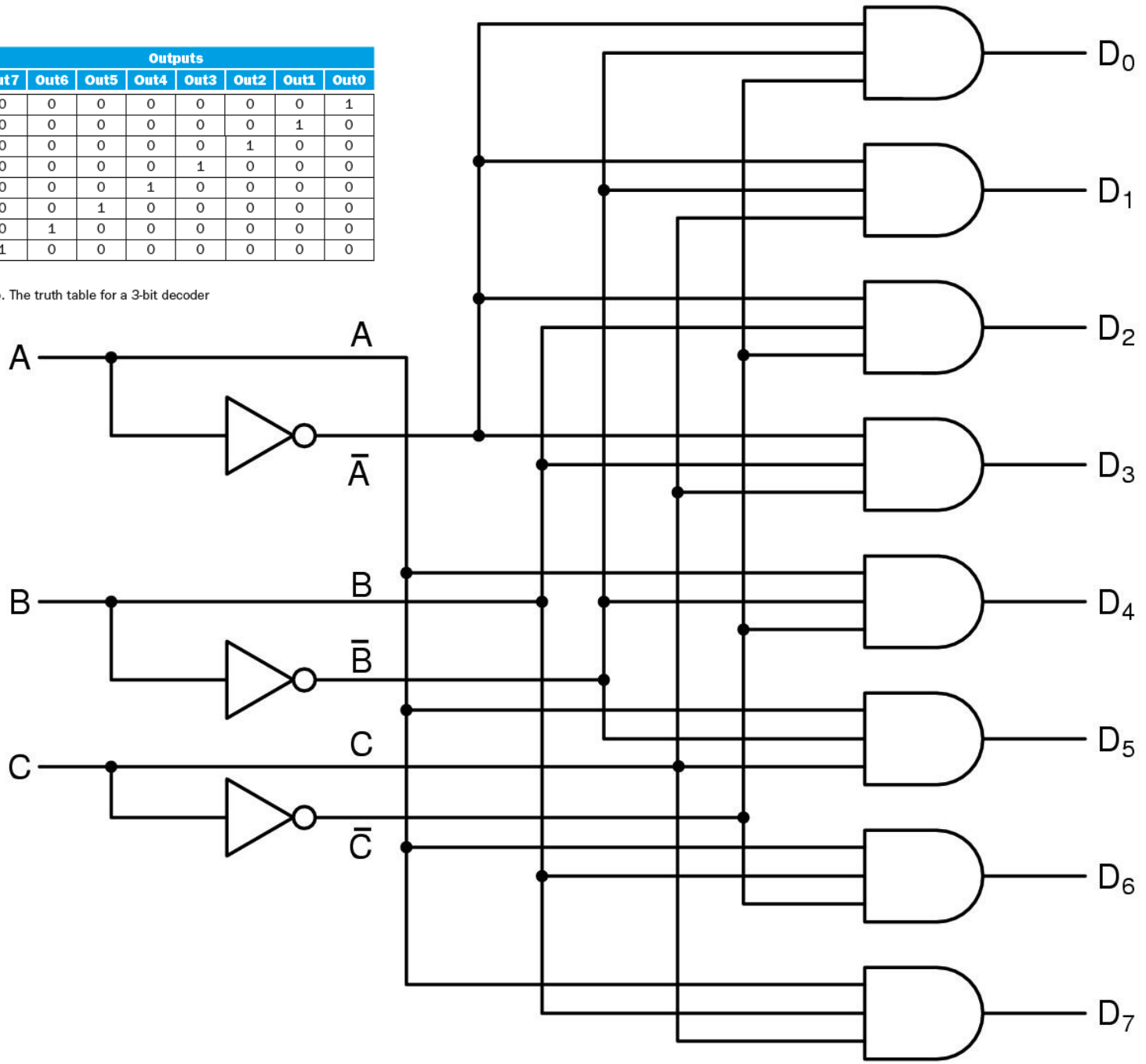
Circuiti combinatori: decoder



a. A 3-bit decoder

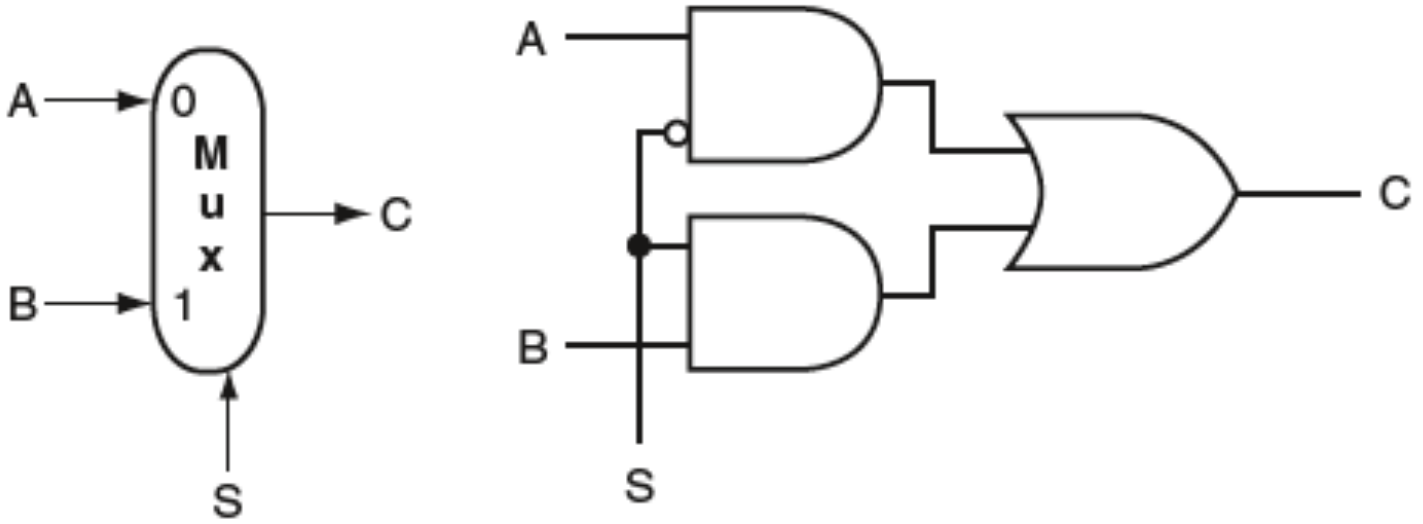
Inputs			Outputs							
12	11	10	Out7	Out6	Out5	Out4	Out3	Out2	Out1	Out0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

b. The truth table for a 3-bit decoder



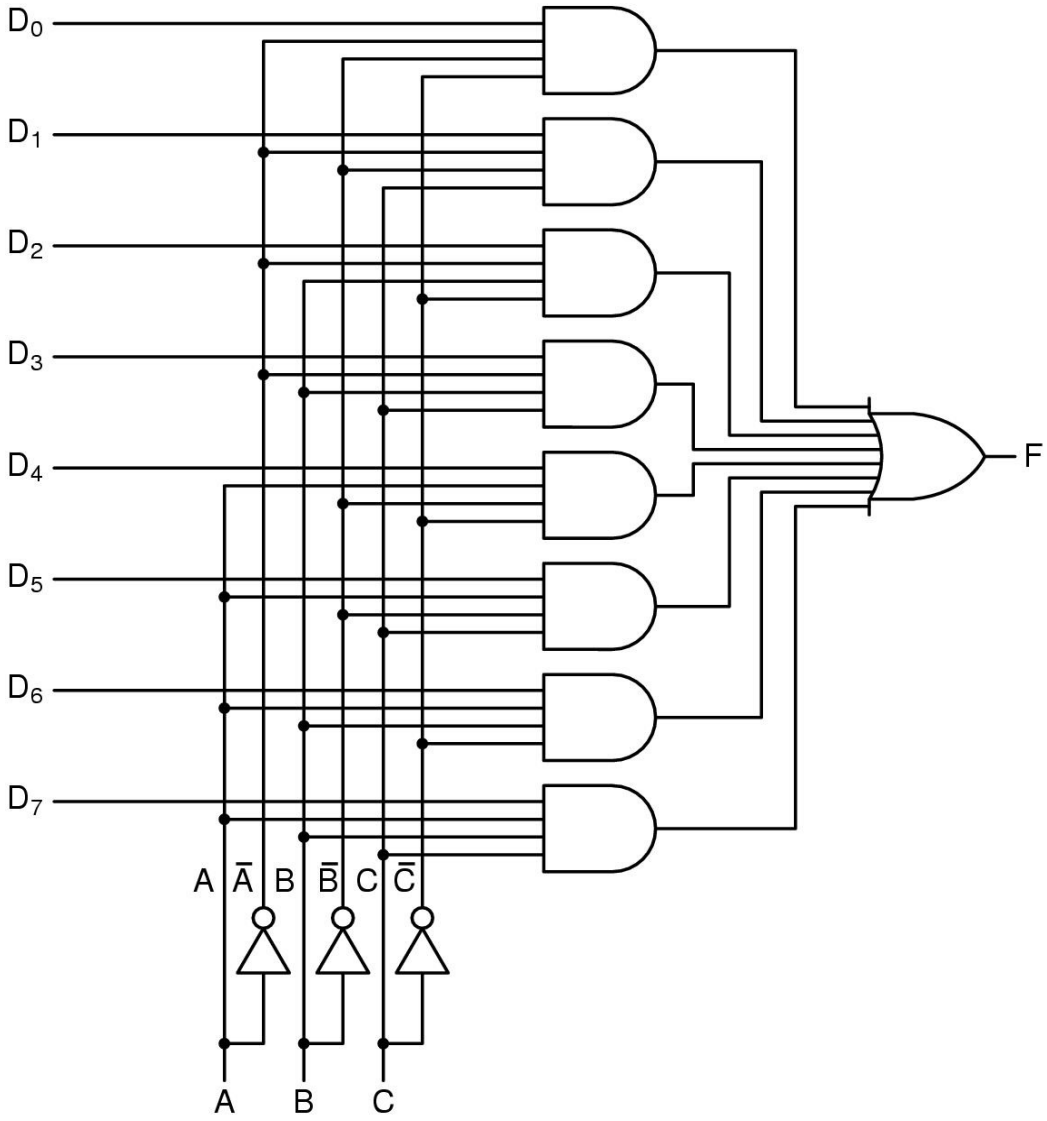
Circuiti combinatori: multiplexer

- ***Multiplexer (o <u>selettore</u>)***:
  2 ingressi, 1 uscita e *1* ingresso di controllo
- La linea di controllo determina quale dei *2* ingressi deve essere selezionato per essere inviato all'uscita
- Esempio con 2 ingressi (A e B), un'uscita (C) ed un ingresso di controllo (S)

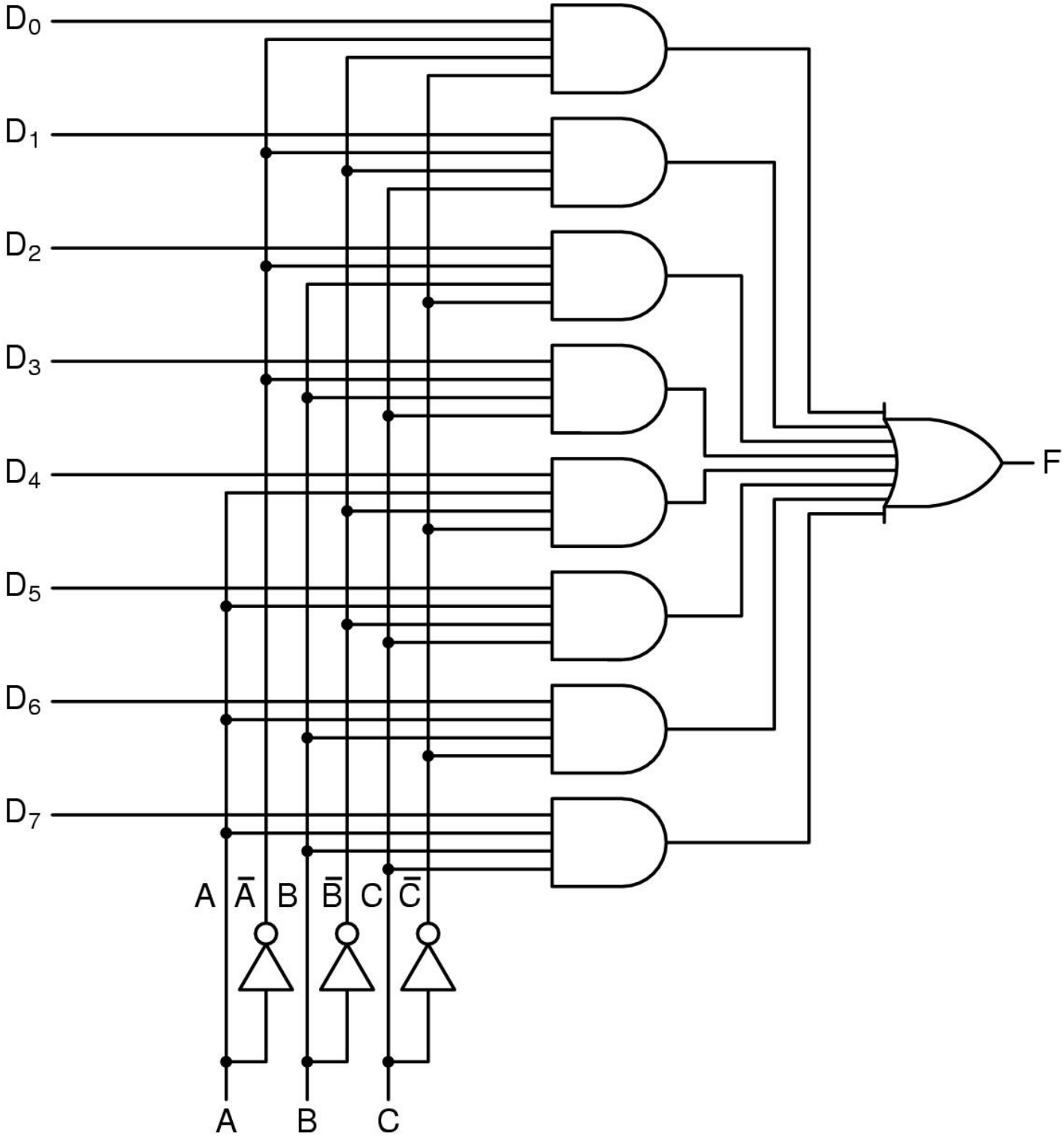


Circuiti combinatori: multiplexer

- ***Multiplexer***: In generale, possiamo avere 2^n ingressi, 1 uscita e n ingressi di controllo
- Le linee di controllo determinano quale dei 2^n ingressi deve essere selezionato per essere inviato all'uscita
- Esempio con 8 ingressi (D_0,.. D_7), un uscita (F) e tre ingressi di controllo (A,B e C)



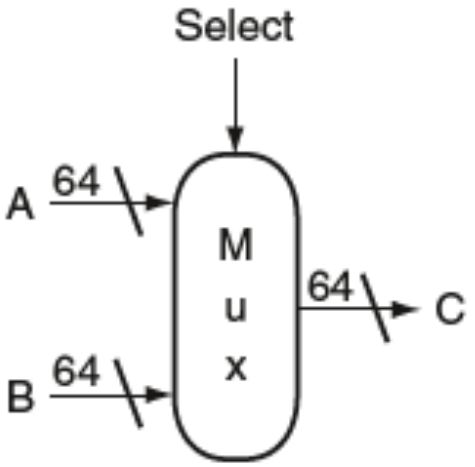
Multiplexer

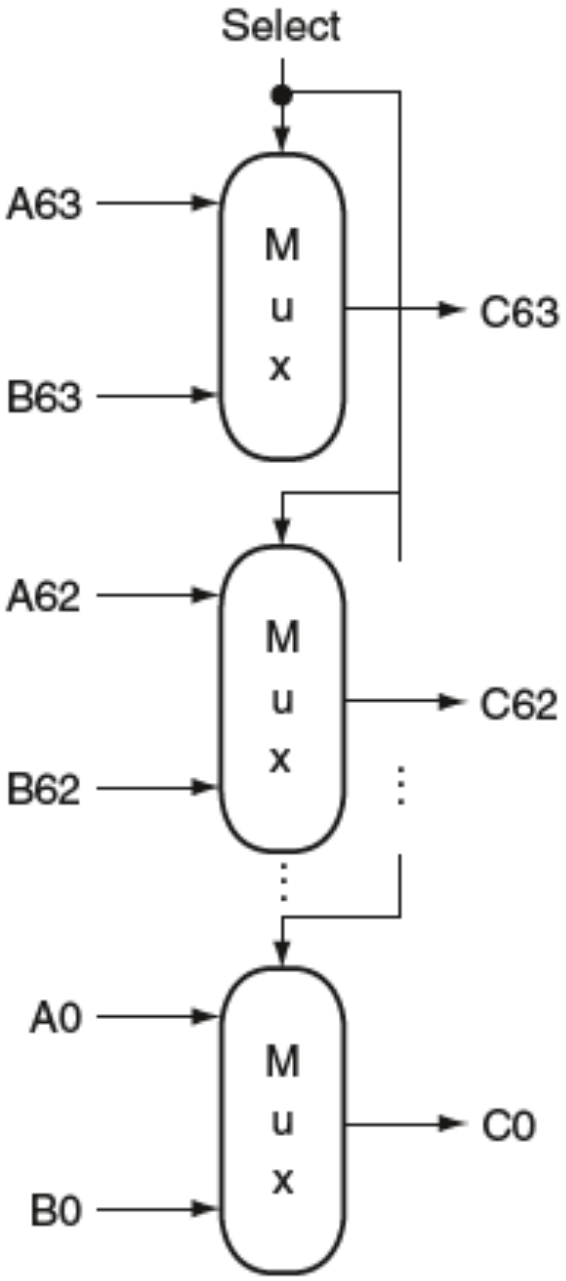


Un multiplexer è composto da un decoder dove ogni AND accoglie un ingresso e l'OR finale

Circuiti combinatori: multiplexer

- ***Multiplexer (o <u>selettore</u>)***: *2* ingressi da 64 linee ciascuno, *1* uscita da 64 linee e *1* ingresso di controllo
- La linea di controllo determina quale dei *2* ingressi da 64 linee deve essere selezionato per essere inviato all'uscita da 64 linee
- Esempio con 2 ingressi (A e B), un uscita (C) ed un ingresso di controllo (Select)

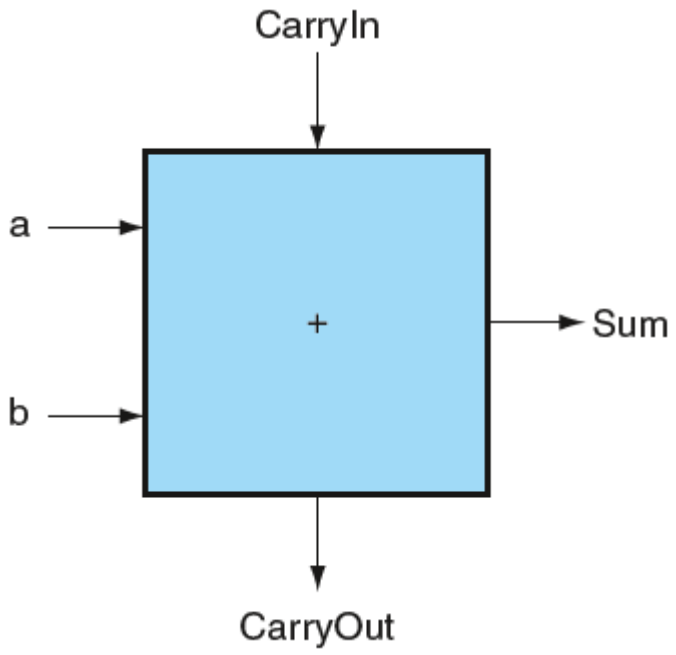




Circuiti numerici: addizionatori

- riceve in ingresso due bit (cifre in base 2) da sommare, a e b nello schema
- riceve in ingresso un bit (cifra in base 2) di riporto, CarryIn nello schema
- restituisce un bit (cifra in base 2) in uscita che rappresenta il risultato, Sum nello schema
- restituisce un bit (cifra in base 2) in uscita che rappresenta il riporto, CarryOut nello schema

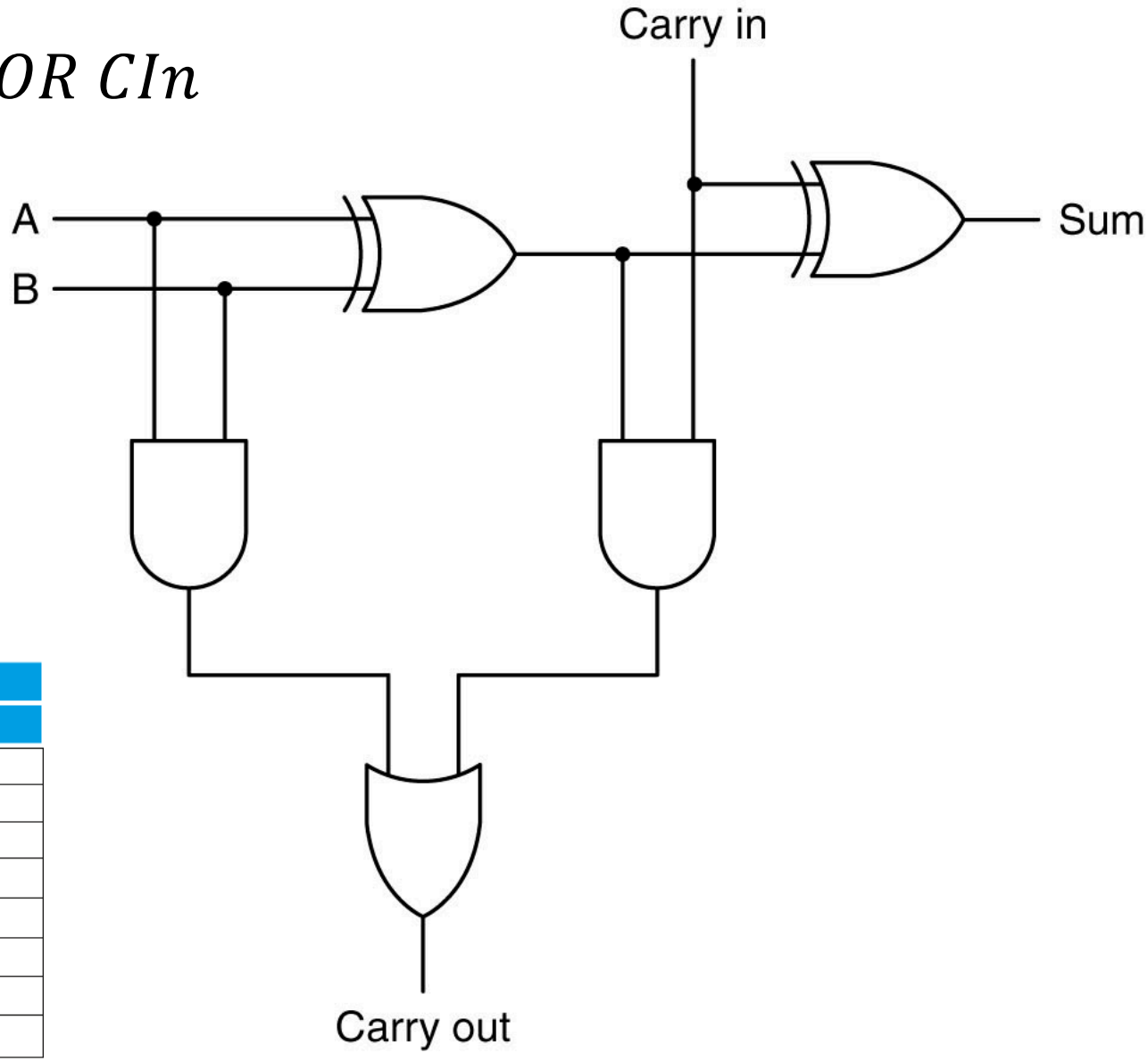
Inputs			Outputs	
a	b	CarryIn	CarryOut	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



Circuiti numerici: addizionatori con XOR

$$C_{Out} = CIn(a\ XOR\ \ b) + a\ b$$

$$Sum\ =\ (a\ XOR\ b)\ XOR\ CIn$$

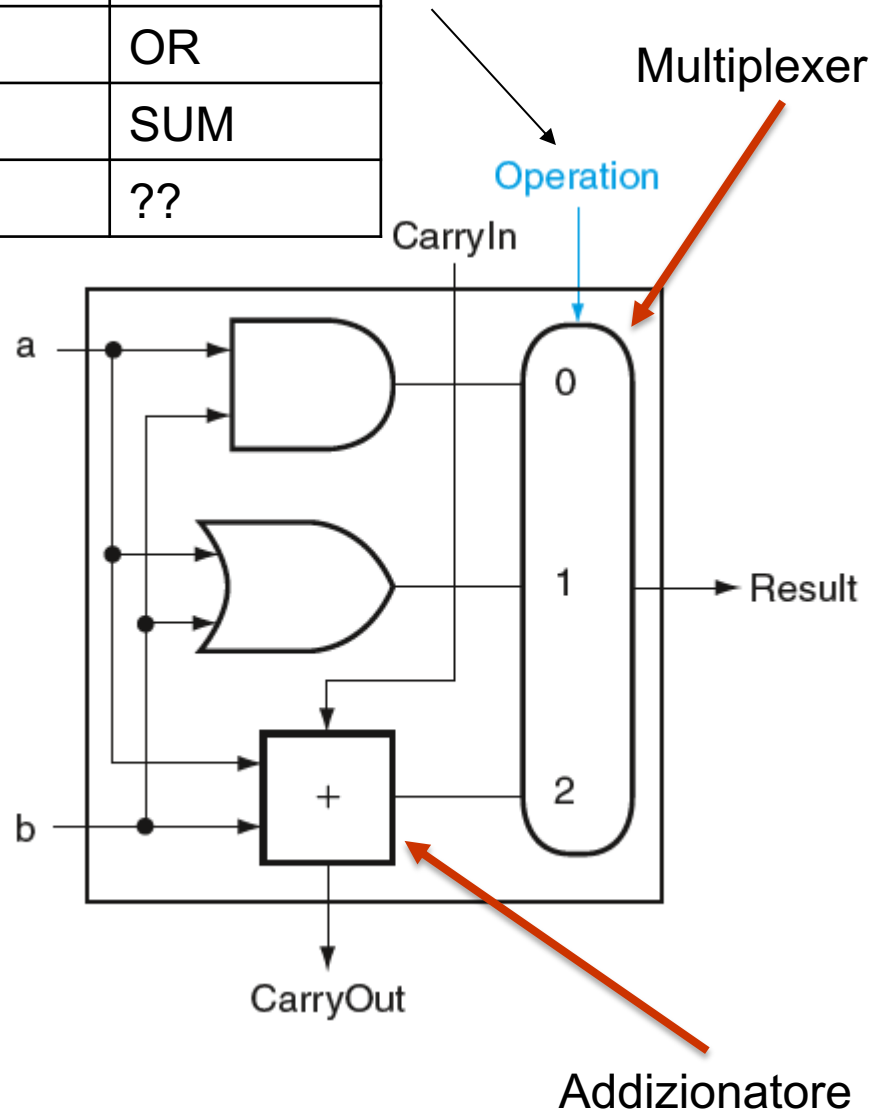


Inputs			Outputs	
a	b	CarryIn	CarryOut	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

RISC-V: ALU ad 1 bit

- Dati a e b è in grado di calcolare:
  - Result = a AND b
  - Result = a OR b
  - Result = a + b (somma)
- Un ingresso di controllo (Operation) seleziona l'operazione desiderata. Dovendo selezionare 3 possibili ingressi in realtà abbiamo 2 linee Operation di controllo.
- Un terzo ingresso fornisce in CarryIn per la somma
- Una seconda uscita rappresenta il CarryOut

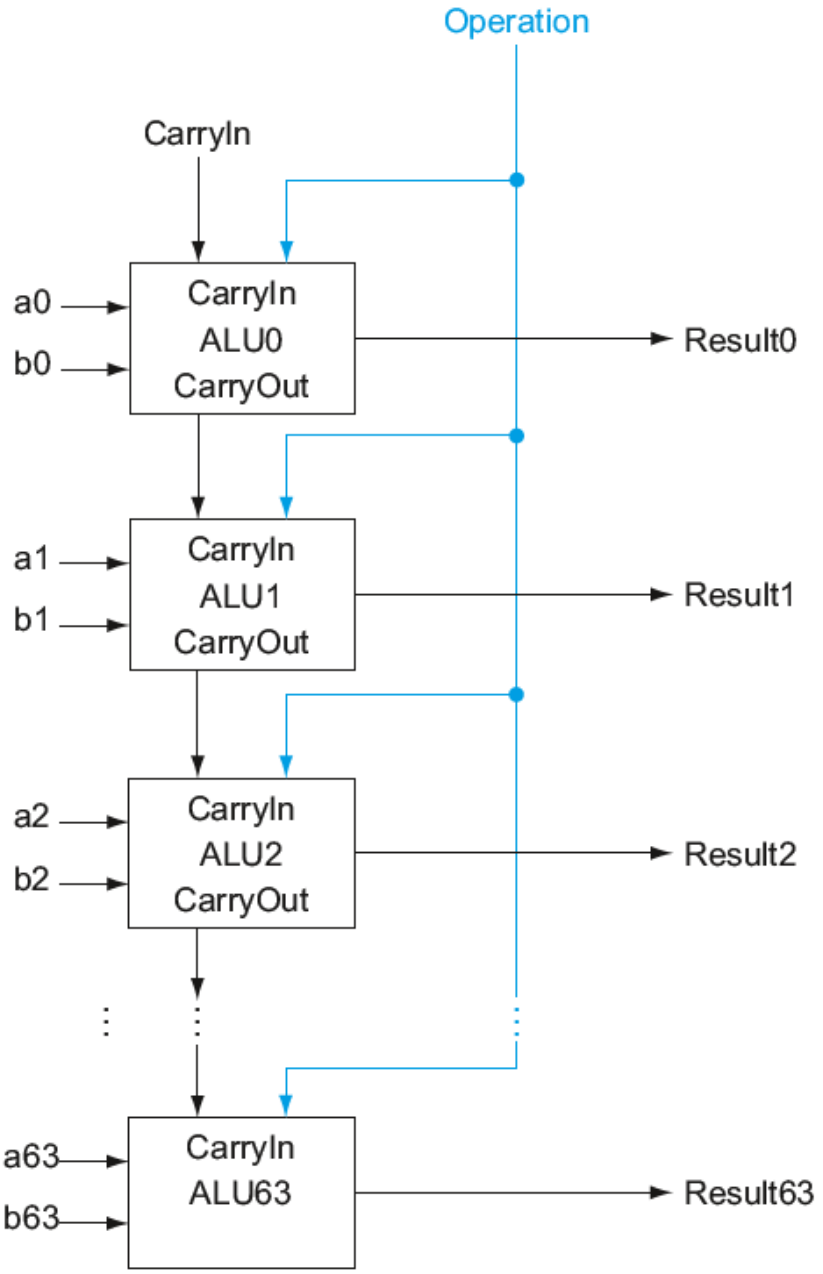
Operation	Function
00	AND
01	OR
10	SUM
11	??



RISC-V: ALU a 64 bit

- Dati a e b su 64 bit è in grado di calcolare:
  - Result = a AND b (bit a bit)
  - Result = a OR b (bit a bit)
  - Result = a + b (somma su 64 cifre)
- Un ingresso di controllo (Operation) seleziona l'operazione desiderata. Dovendo selezionare 3 possibili ingressi in realtà abbiamo 2 linee Operation di controllo.

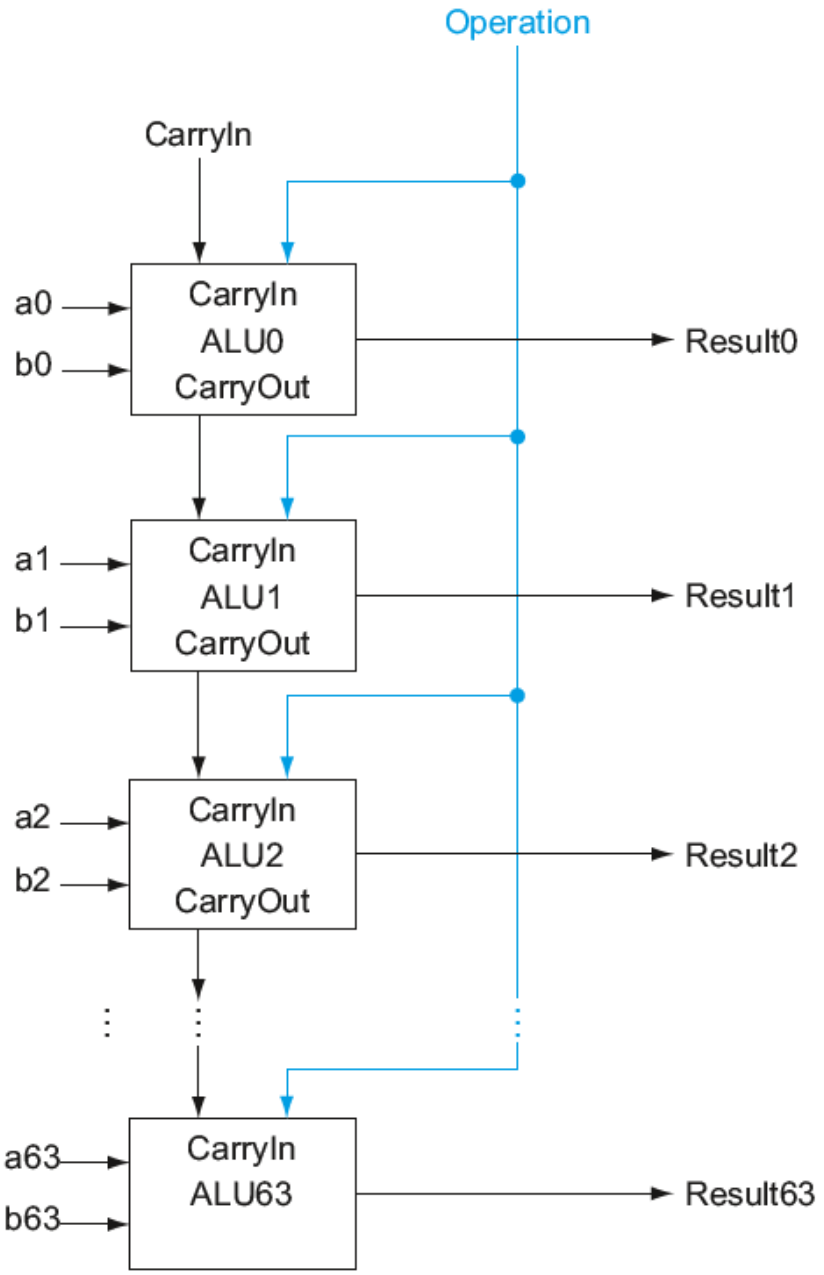
Operation	Function
00	AND
01	OR
10	SUM
11	??



RISC-V: ALU a 64 bit

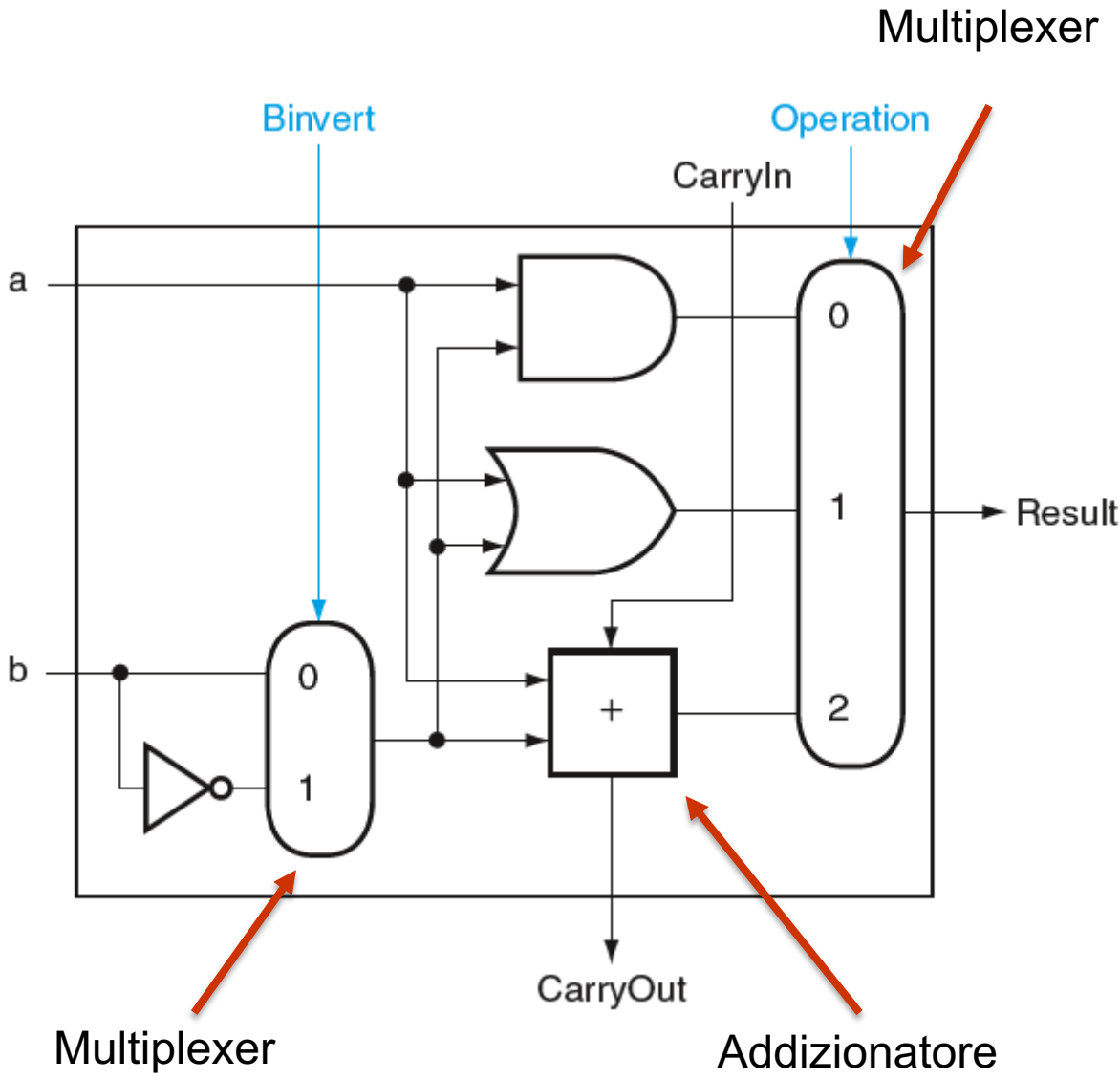
- Una ALU per operandi a e b ad n bit si ottiene utilizzando n ALU ad 1 bit
- Ogni ALU esegue l'operazione sulla coppia di bit degli operandi a e b nella stessa posizione (bit slice)
- Per sommare due operandi a e b di n bit il CarryOut dell'ALU per il bit in posizione i diventa il CarryIn del bit in posizione i + 1
- Il CarryIn del bit meno significativo (a0 e b0) può essere usato come segnale di incremento per calcolare a + b + 1 (servirà per le sottrazioni)

Operation	Function
00	AND
01	OR
10	SUM
11	??



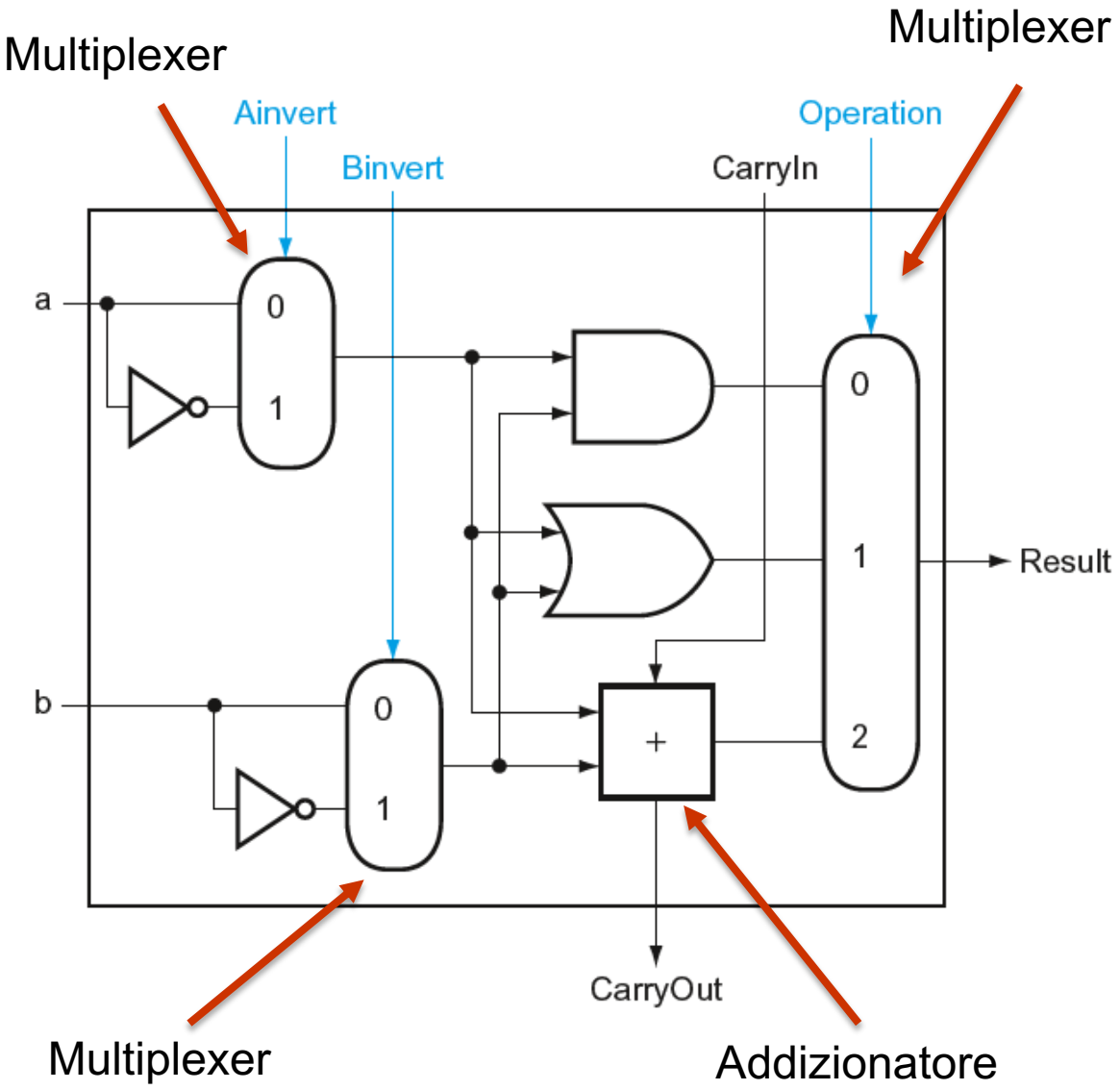
RISC-V: ALU ad 1 bit (sub)

- Dati a e b è in grado di calcolare le funzioni precedenti con b o $\bar{b}$
- Un ingresso di controllo aggiuntivo (Binvert) seleziona l'operazione desiderata
- Serve, ad esempio, per le sottrazioni (a - b)
  - Ricordare il complemento a 2
  - Con l'uso di $\bar{b}$
  - CarryIn = 1 per il bit meno significativo



RISC-V: ALU ad 1 bit (nor)

- Dati a e b è in grado di calcolare le funzioni precedenti con a o $\bar{a}$
- Un ingresso di controllo aggiuntivo (Ainvert) seleziona l'operazione desiderata
- Serve, ad esempio, per funzioni NOR: $\overline{(a \text{ OR } b)}$ con l'uso di $(\bar{a} \text{ AND } \bar{b})$



RISC-V: ALU per slt

- L'istruzione Set on Less Than (slt) restituisce 1 se rs1 < rs2 e 0 altrimenti
- Il valore di tutti i bit in uscita tranne quello *meno significativo* deve essere 0
- Il valore del bit *meno significativo* dipende dal confronto

- Idea: **usare la sottrazione**

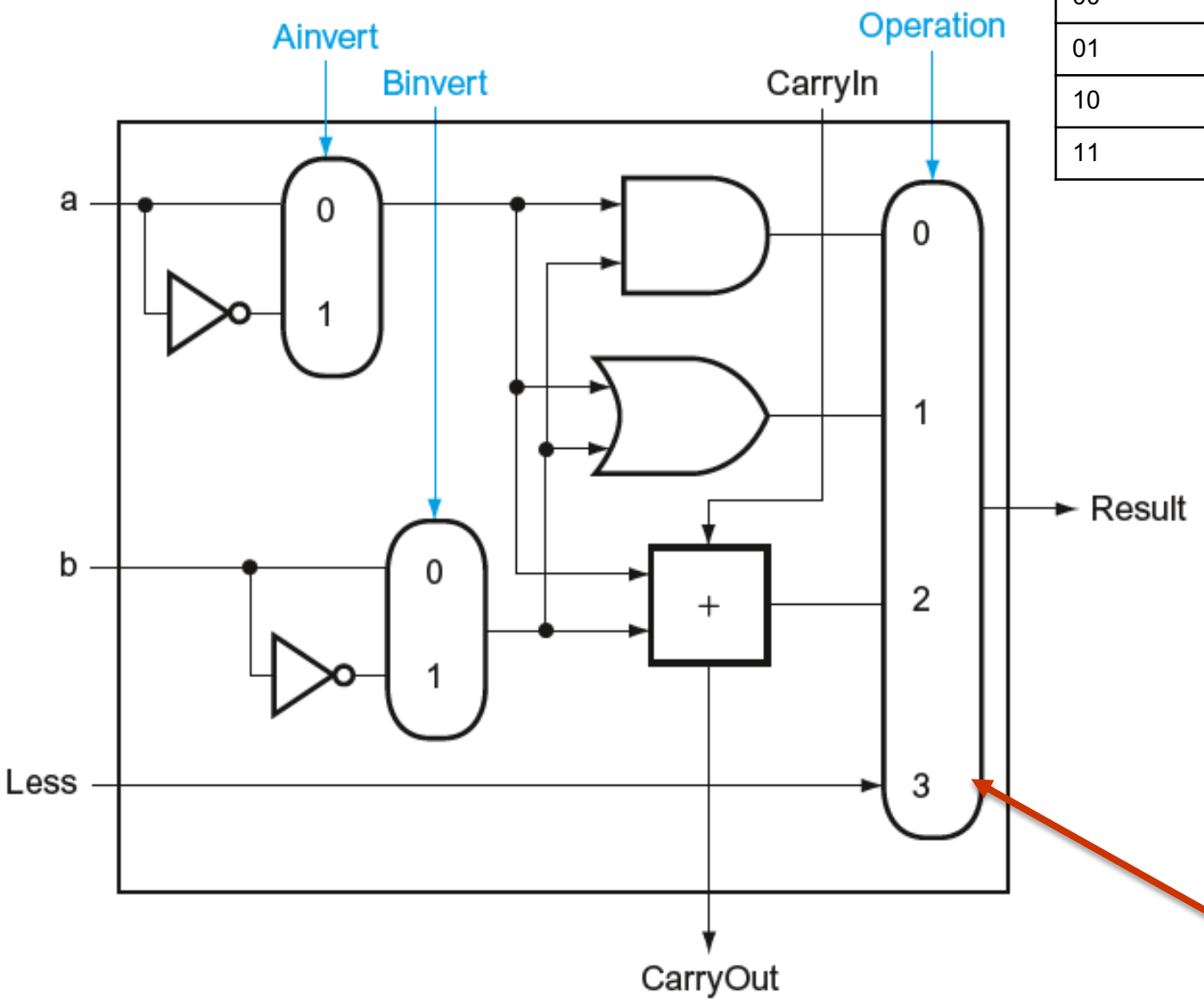
(a - b) < 0 implica a < b

- Per vedere se **(a - b) è negativo** verifico se il **bit più significativo vale 1**

RISC-V: ALU per slt

- ALU estesa per ognuno dei 63 bit meno significativi degli operandi

Operation	Function
00	AND
01	OR
10	SUM
11	LESS

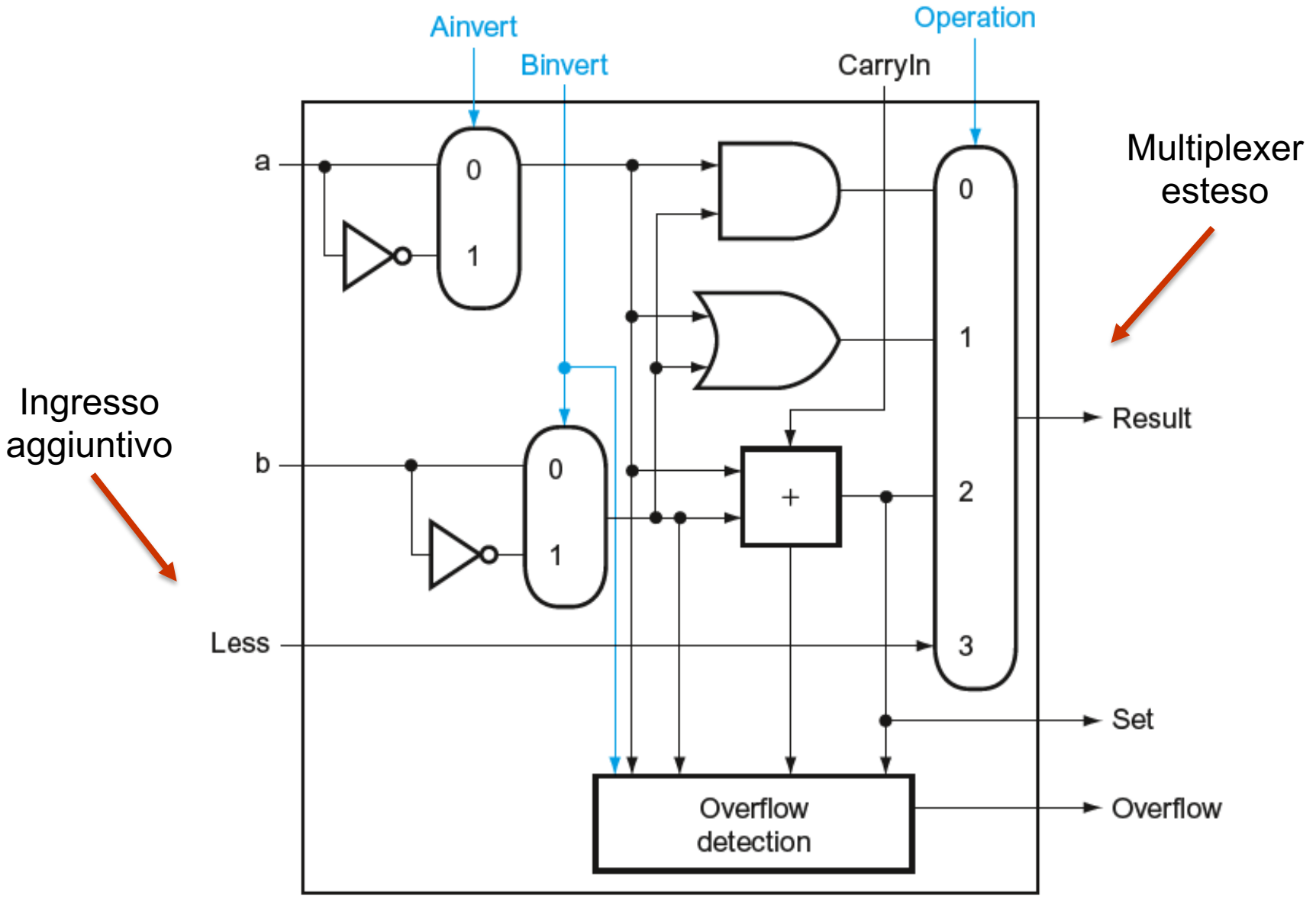


Ingresso aggiuntivo

Multiplexer esteso

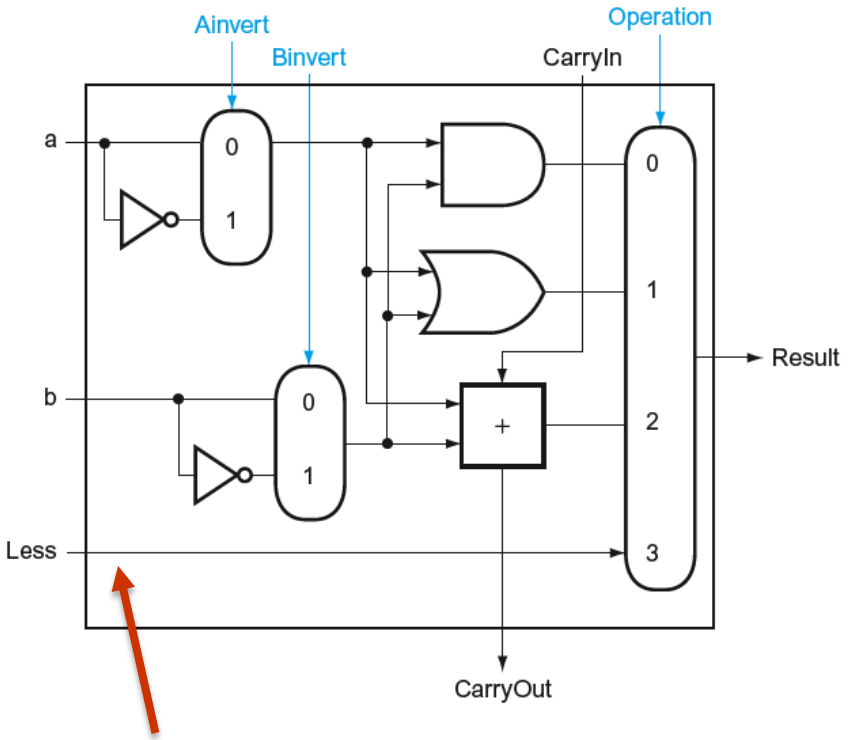
RISC-V: ALU per slt

- ALU estesa per il 64-esimo bit, quello più significativo degli operandi

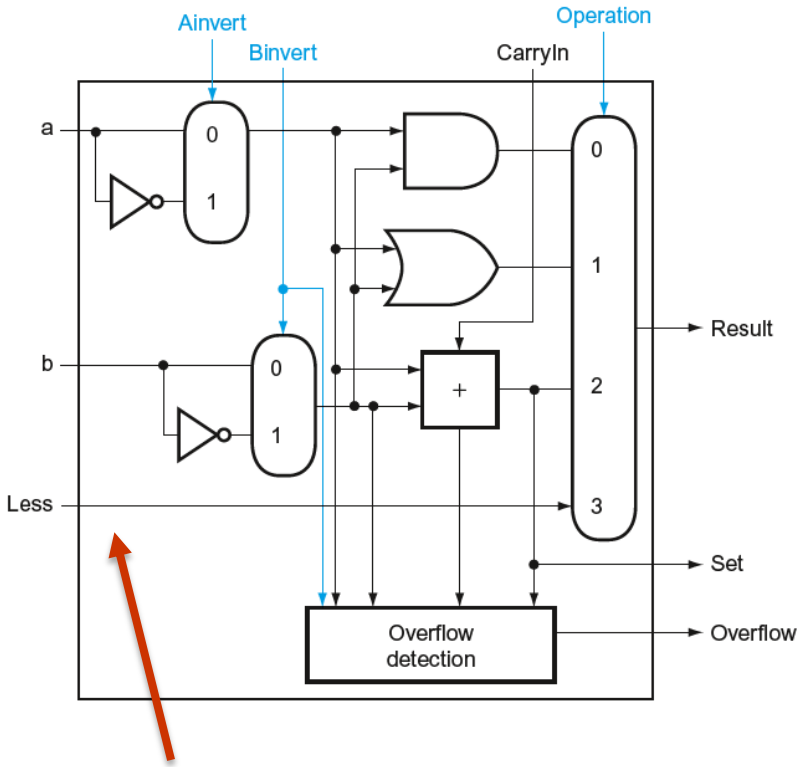


RISC-V: ALU per slt

- L'ingresso Less dell'ALU per i bit da 1 a 63 deve essere uguale a 0
- L'ingresso Less dell'ALU per il bit 0 (il meno significativo) deve essere uguale al bit di segno del risultato della differenza a – b
  - ovvero il bit della somma dell'ALU del bit 63
- Per questo l'ALU per il bit 63 ha un'uscita Set per il risultato dell'addizionatore



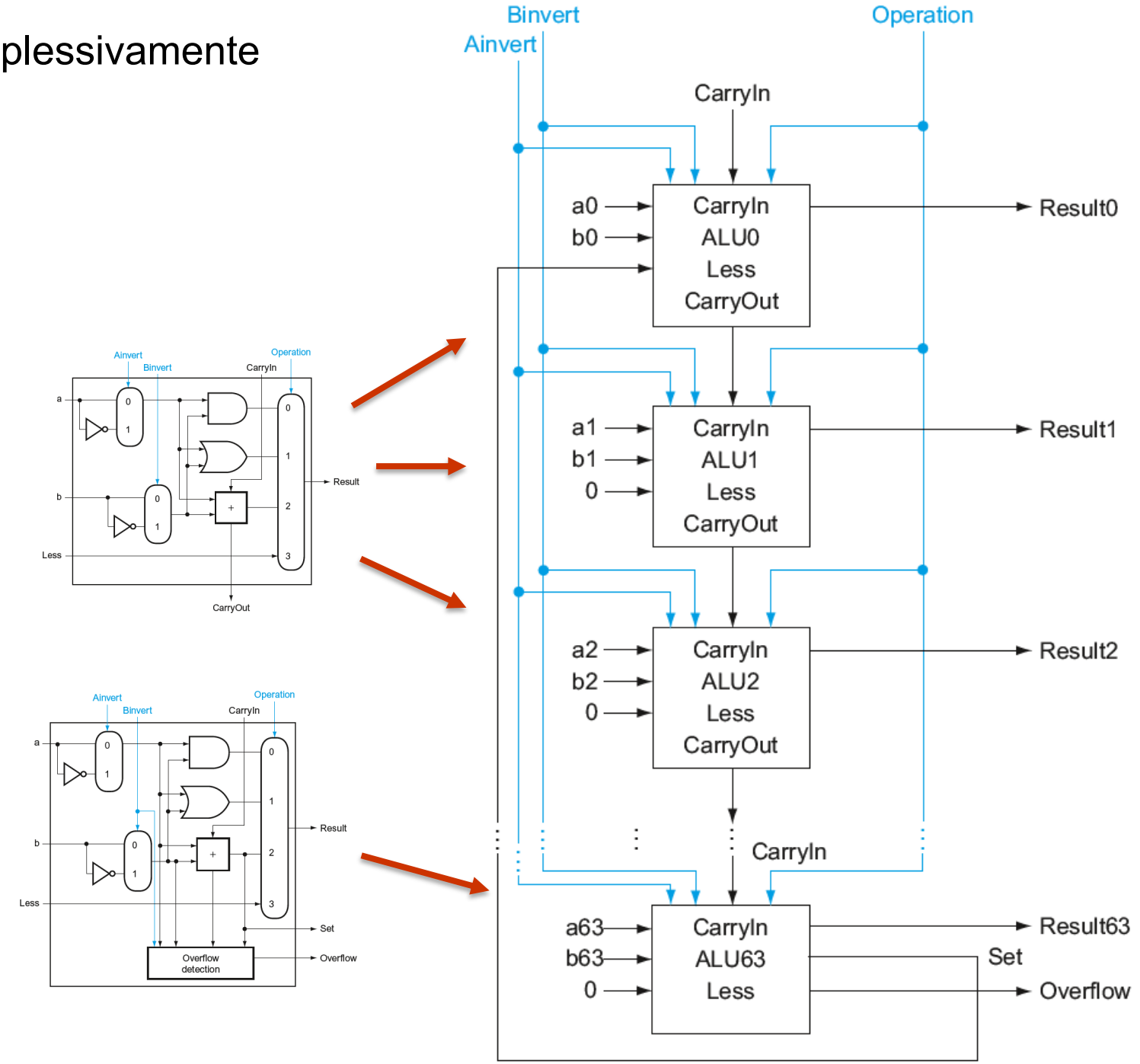
- Ingresso a 0 per tutti tranne che per il bit meno significativo



- Ingresso a 0

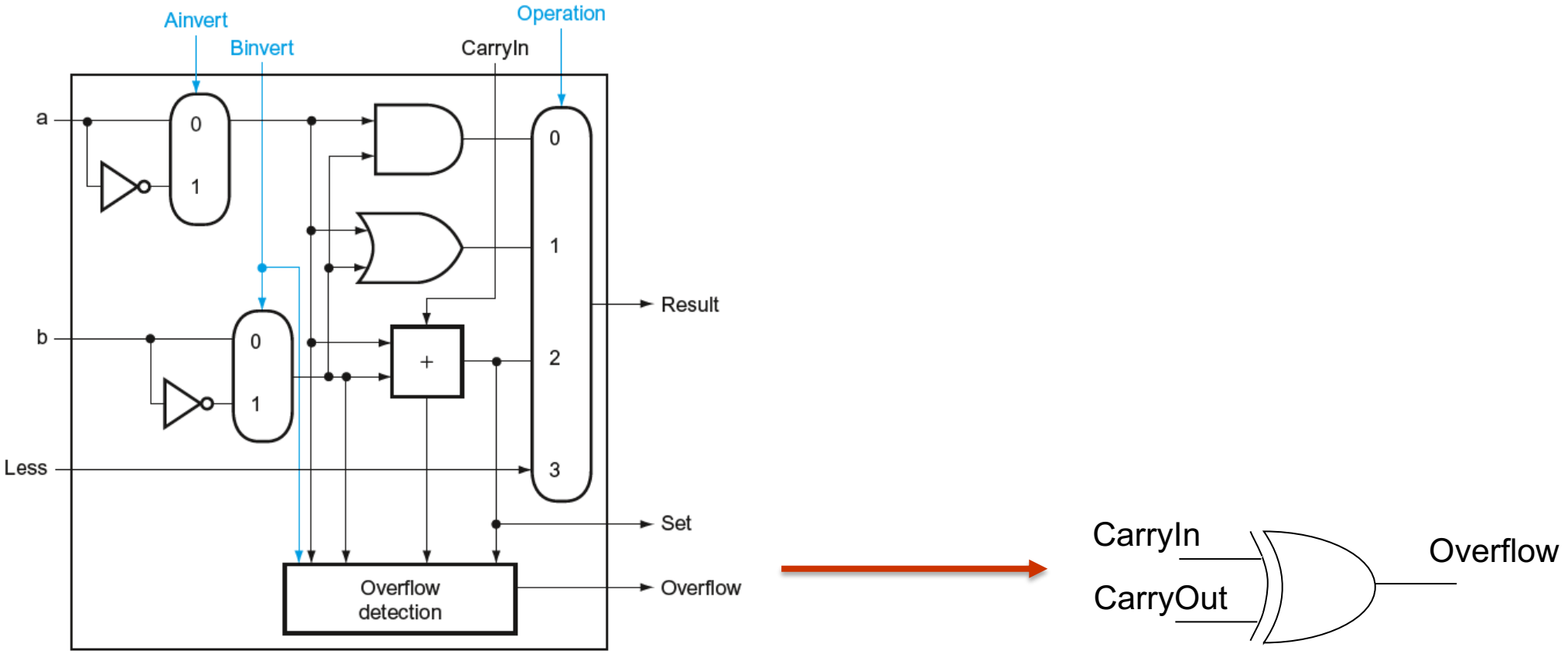
RISC-V: ALU per slt

- Complessivamente



RISC-V: overflow

- Figura: Nella somma o differenza di due interi con segno in complemento a due, abbiamo overflow se I due operandi hanno lo stesso segno ma il risultato ha segno opposto.
- Ricordate la teoria del complemento a 2? Si ha overflow quando il CarryIn ed il CarryOut del bit più significativo sono discordi.



Controllo operazioni dell'ALU

- Ogni volta che vogliamo che l'ALU esegua sottrazioni dobbiamo asserire (porre a 1) sia CarryIn sia Binvert

Operazione	Funzione	Binvert	CarryIn0
0	AND	0	0
1	OR	0	0
2	ADD	0	0
2	SUB	1	1
3	SLT	1	1

CarryIn0 = Binvert

Usiamo una sola linea chiamata **Bnegate**

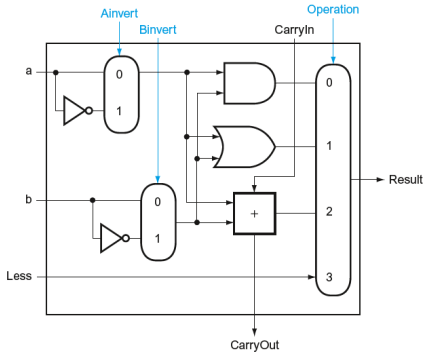
RISC-V: ALU per `beq`

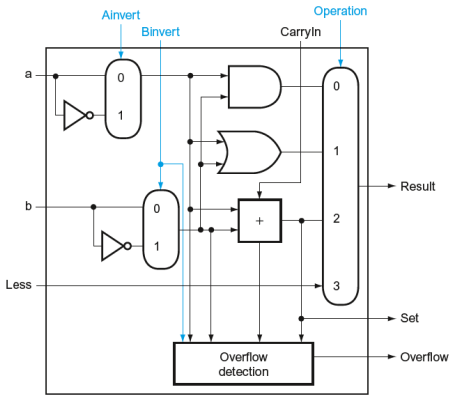
- L'istruzione Branch if Equal (beq) realizza un salto se due registri sono uguali
- Si realizza il salto eseguendo (a - b) e controllando se il risultato è uguale a 0
- L'ALU fornisce il valore 1 quando tutti i bit del risultato sono a zero, in quanto:

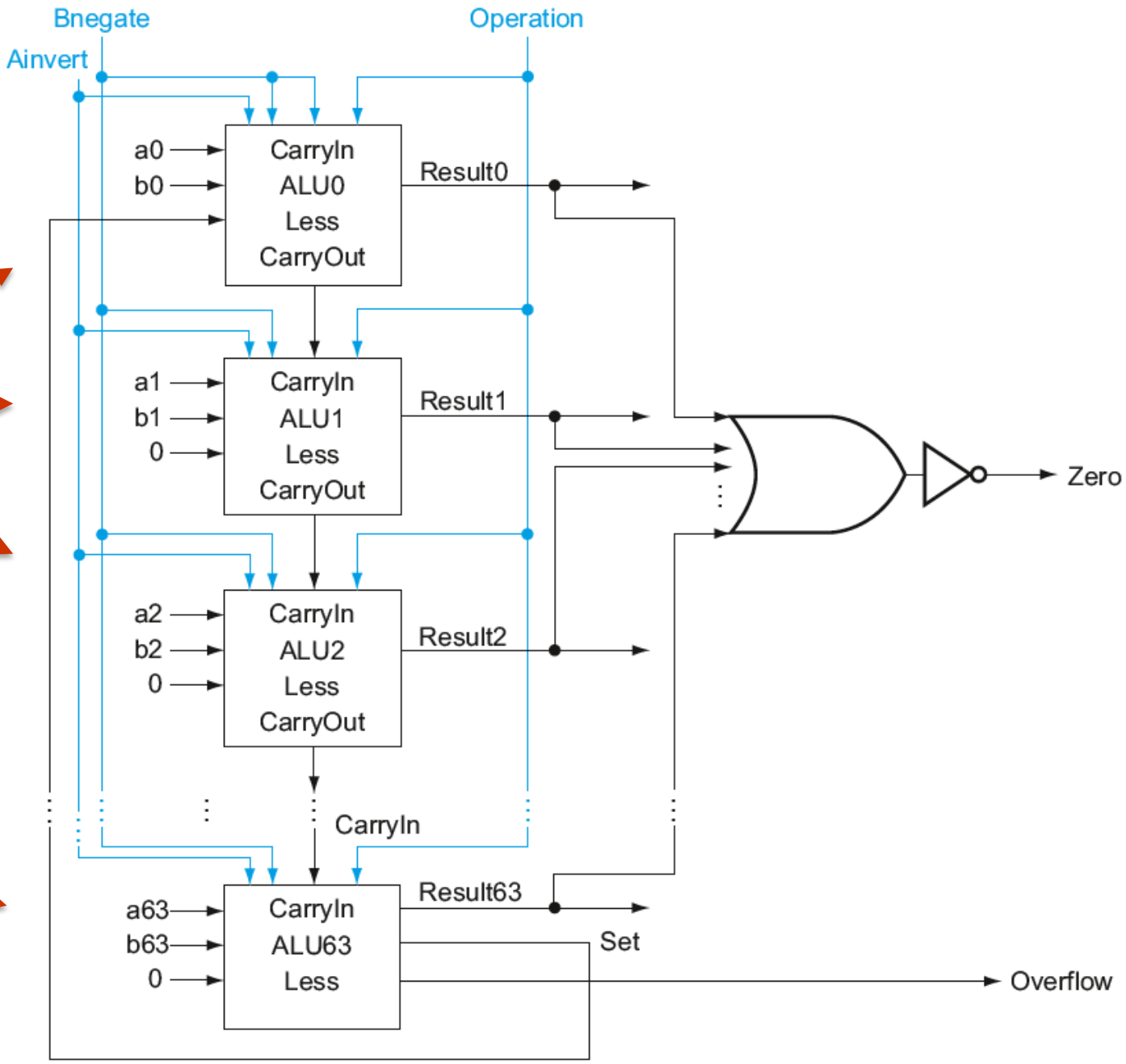
```
(a-b)==0 -> a==b
```

"Zero" vale 1 quando il risultato (a-b) è zero!

RISC-V: ALU per `beq`







Controllo operazioni dell'ALU

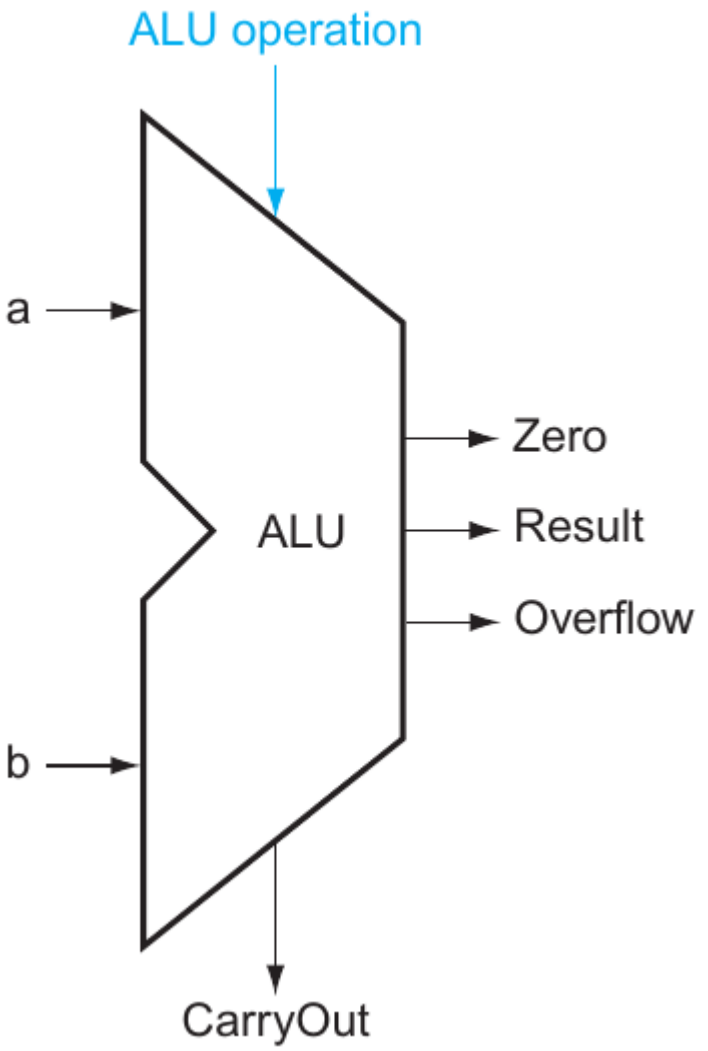
- Per il controllo dell'ALU, possiamo pensare alla combinazione di
  - 1 bit per l'ingresso Ainvert
  - 1 bit per l'ingresso Bnegate
  - 2 bit per gli ingressi Operation
- come a linee di controllo a 4 bit per fare in modo che l'ALU esegua la somma, la sottrazione, l'AND, l'OR, la NOR o la `slt`

Ainvert	Bnegate	Operation
0	0	00
0	0	01
0	0	10
0	1	10
0	1	11
1	1	00

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

RISC-V: ALU

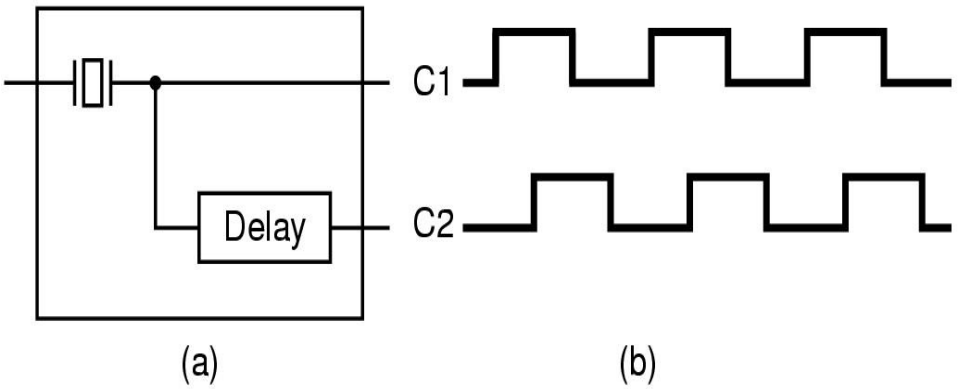
- Il simbolo comunemente usato per rappresentare una **ALU**
- Anche comunemente usato per indicare un **addizionatore**, quindi è prassi indicare esplicitamente con una scritta quale di questi due componenti si intende.

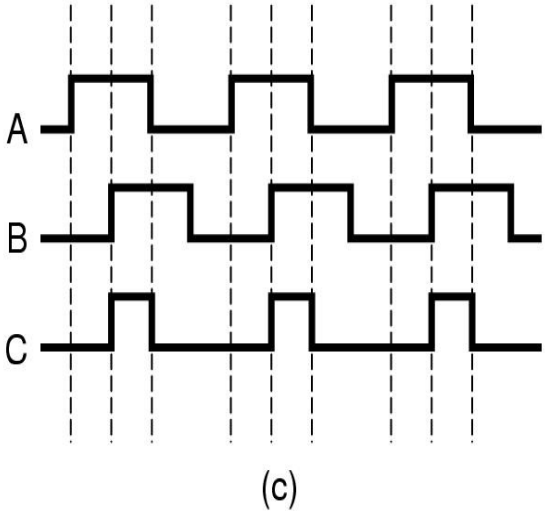


ALU control lines	Function
0000	AND
0001	OR
0010	add
0110	subtract
0111	set less than
1100	NOR

Clock

- ***Clock***: un circuito che emette una serie di impulsi con una specifica larghezza e intermittenza
- ***Tempo di ciclo di clock***: intervallo fra i fronti corrispondenti di due impulsi consecutivi
- Fronte di salita di C1
  fronte di discesa di C1
  fronte di salita di C2
  fronte di discesa di C2





500 MHz = 2 nsec di tempo
di ciclo di clock

Clock

Tempo di ciclo di clock

Il tempo si misura nei calcolatori in sottomultipli di secondo:

- **1 ms** (millisecondo) = $1*10^{-3}$ sec.
- **1 µs** (microsecondo) = $1*10^{-6}$ sec.
- **1 ns** (nanosecondo) = $1*10^{-9}$ sec.

Frequenza = 1/Tempo di ciclo

Frequenza di clock

La frequenza specifica il numero di periodi di clock per unità di tempo (ovvero per secondo).
La frequenza si calcola come inverso del tempo di ciclo di clock.
L'unità di misura è **l'Hertz**, di cui si utilizzano per i calcolatori, i multipli:

1 KHz (KiloHertz) = $1*10^3$ Hertz
1 MHz (MegaHertz) = $1*10^6$ Hertz
1 GHz (GigaHertz) = $1*10^9$ Hertz