



La ALU

Prof. Alberto Borghese
Dipartimento di Scienze dell'Informazione

borgnese@di.unimi.it

Università degli Studi di Milano

Riferimento sul Patterson: sezione C.5



Sommario

ALU ad 1 bit

ALU a 32 bit

Comparazione, Overflow, Test di uguaglianza

Tecnologie di costruzione di una ALU



Funzione della ALU



E' integrata nel processore, all'inizio degli anni 90 è stata rivoluzionaria la sua introduzione con il nome di co-processore matematico.

Esegue le operazioni aritmetico-logiche.

E' costituita da circuiti combinatori. Utilizza i blocchi di base già visti.

Opera su parole (MIPS 32 bit).

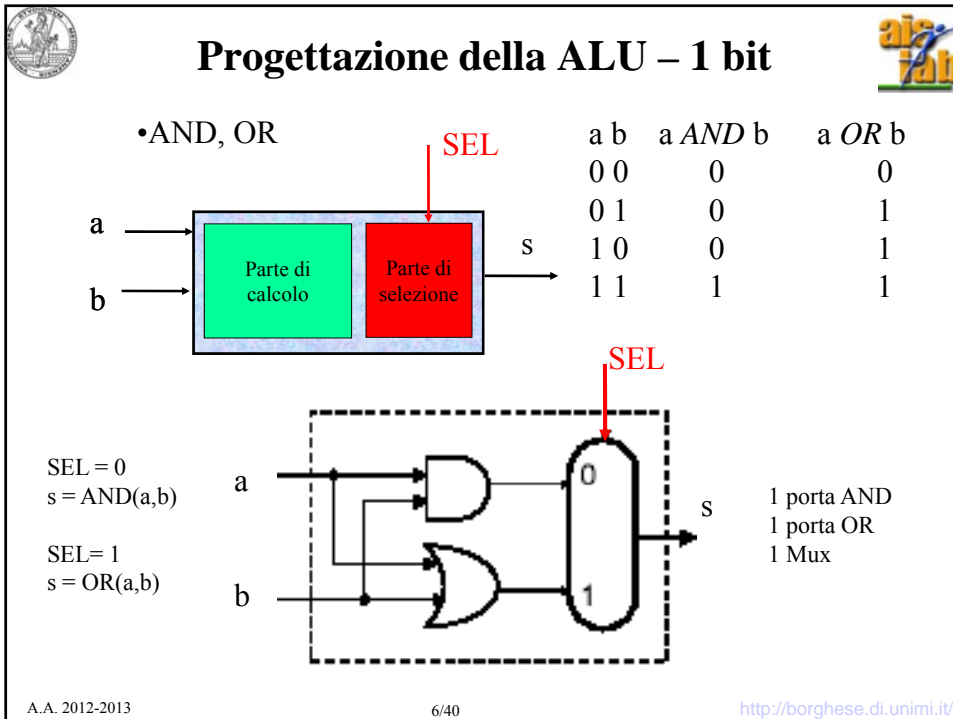
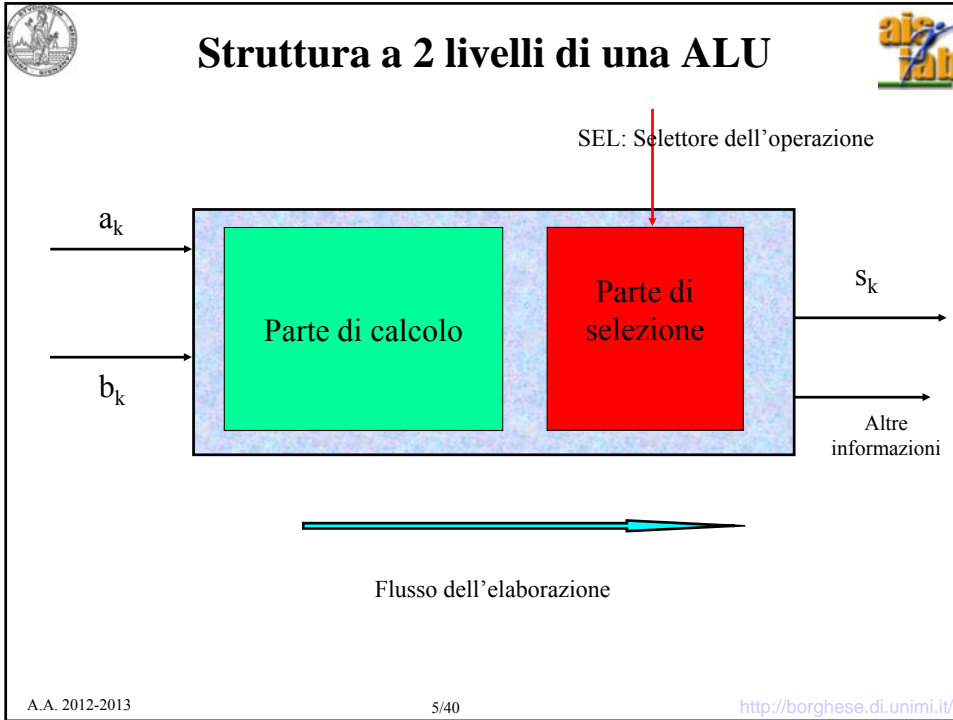
Le ALU non compaiono solamente nei micro-processori.



Problematiche di progetto



- Velocità (Riporto).
- Costo.
- Precisione.
- Affidabilità
- Consumo.

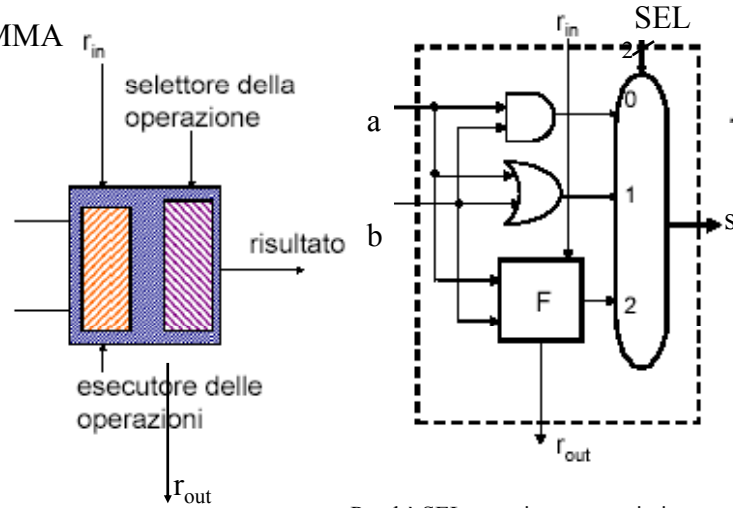




La nuova struttura della ALU – 1 bit



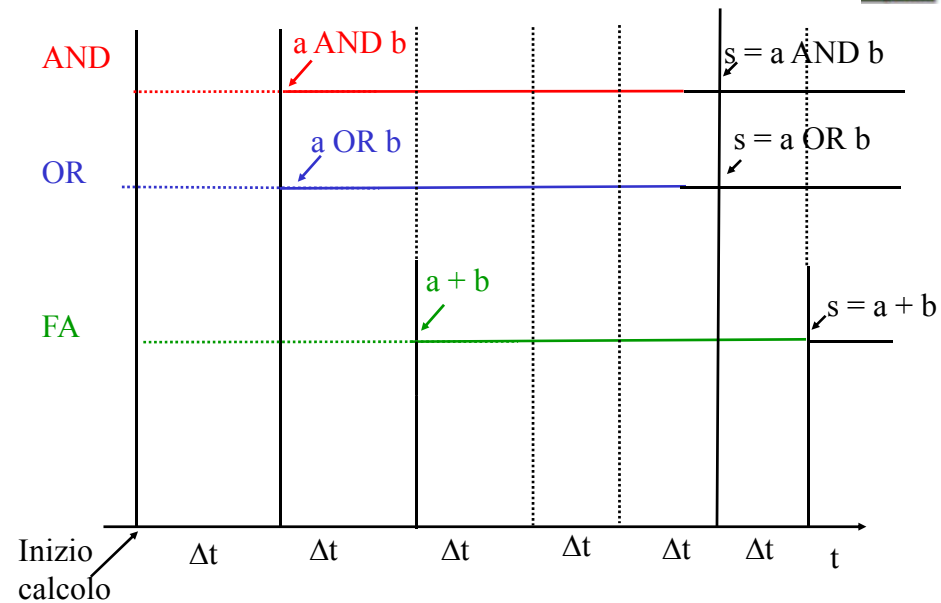
- AND
- OR
- SOMMA



Perchè SEL non viene messo in ingresso?



I cammini critici all'interno della ALU





Valutazione ALU a 1 bit



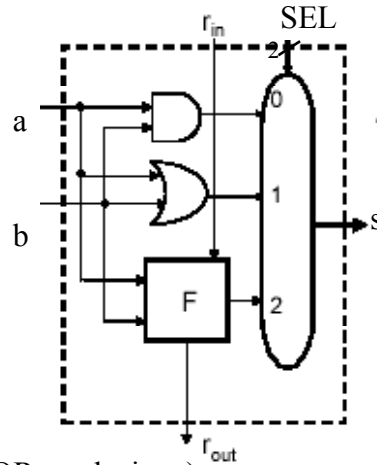
- AND
- OR
- SOMMA

Complessità 1° livello: $5 + 2 = 7$

Complessità 2° livello: 8

CC 1° livello: 2 per s_{out} , 3 per r_{out}

CC 2° livello: 4 (2 AND + 2 OR del mux)



CC complessivo: 2 (calcolo) + 2 (OR – selezione)

Gli AND del selettore sono attivati in parallelo ai circuiti di calcolo



Sommario



ALU ad 1 bit

ALU a 32 bit

Comparazione, Overflow, Test di uguaglianza

Tecnologie di costruzione di una ALU



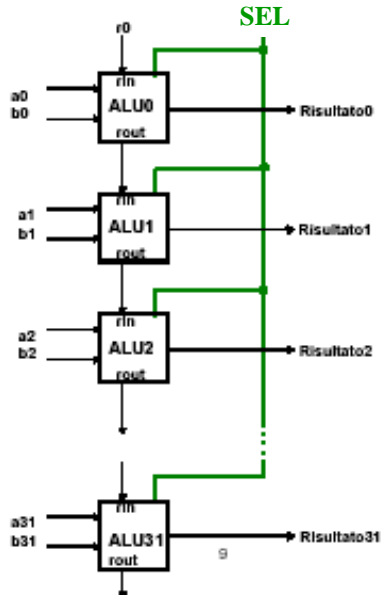
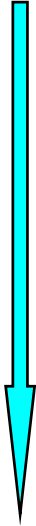
ALU a 32 bit



Come collegare le
ALU ad 1 bit?

Flusso di calcolo

Perchè non si può
parallelizzare?



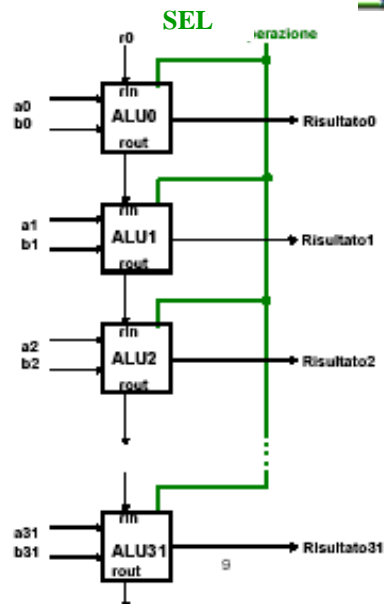
Valutazione ALU a 32 bit



Complessità: $15 \times 32 =$
480 porte logiche

Cammino critico: $3 \times 31 +$
 $2 (s_{out}) +$
 $[1 + 2]$ (selezione) =
98 porte logiche

per 4 operazioni





Sottrazione



In complemento a 2 diventa un'addizione: $a - b = a + \bar{b} + 1 = 1 + a + \bar{b}$

Esempio: $s = 3 - 4$; su 3 bit

3 -> 011	011 +
-4 -> 100 in complemento a 2	100 =
-1 -> 111 in complemento a 2	111

Posso scrivere il numero negativo in complemento a 2 come somma:

	4 -> 100	numero positivo: b
Passo I - Complemento a 1	011+	complemento a 1: \bar{b} +
Passo II - Sommo + 1	1=	sommo 1: 1=
Risultato - Complemento a 2	100	risultato -b

Posso quindi scrivere: $-b = \bar{b} + 1$



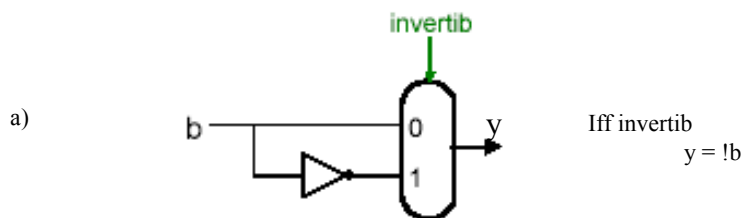
Sottrazione



In complemento a 2 diventa un'addizione: $a - b = a + \bar{b} + 1$

Serve:

- un inverter (NOT).
- la costante 1

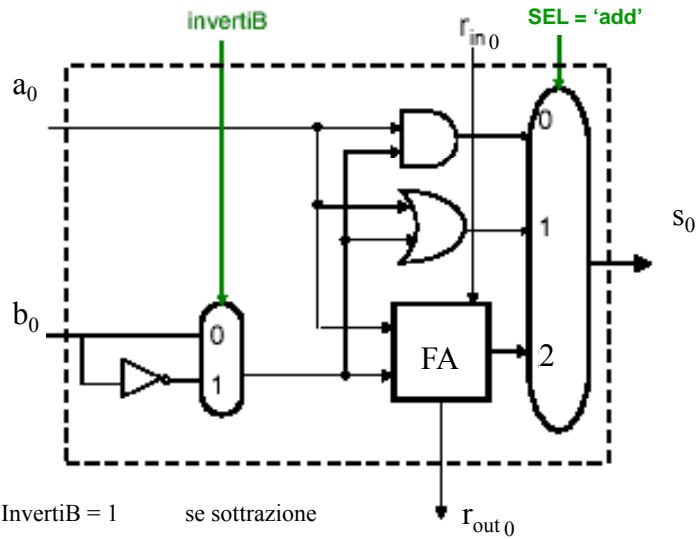


Aggiunge 2 porte logiche al cammino critico.

- Da dove prendo la costante 1?



Sottrazione - ALU₀



$r_{in}(0) = \text{InvertiB} = 1$ se sottrazione

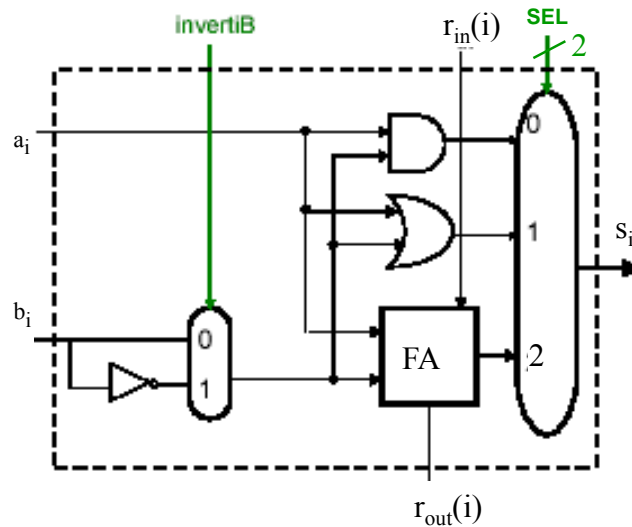
(occorre utilizzare un full adder anche per il bit meno significativo con $r_{in0} = 1$).
Effettuo quindi la somma di 1 con la somma della prima coppia di bit.



Sottrazione - ALU_i



- AND
- OR
- SOMMA
- SOTTRAZIONE



$r_{in}(i) = r_{out}(i-1)$ $i = 1, 2, 3, \dots, 31$
 $\text{InvertiB} = 1$

$i \neq 0$
se sottrazione



Operazioni particolari - ALU_i

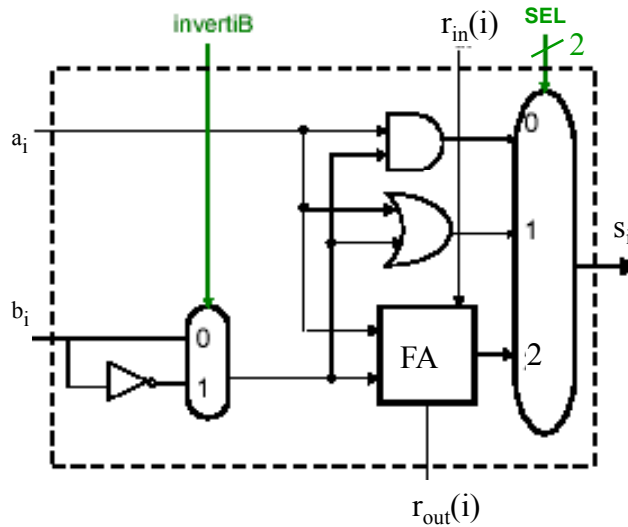


E' possibile programmare questa ALU per eseguire

a AND !b

oppure:

a OR !b



InvertiB = 1
SEL = AND, OR

A.A. 2012-2013

17/40

<http://borghese.di.unimi.it/>



Sottrazione: ALU a 32 bit



$r_{in}(0) = \text{InvertiB} = 1$
se sottrazione

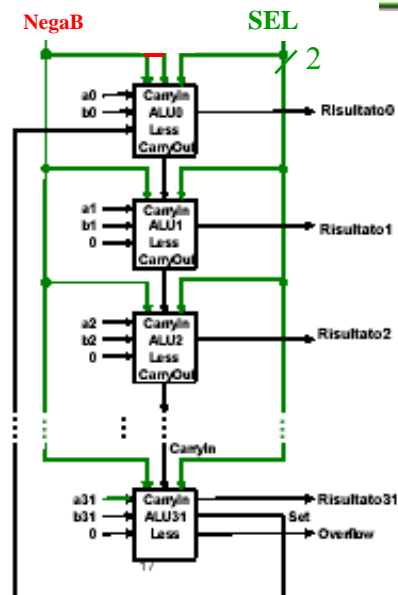
- AND
- OR
- SOMMA
- SOTTRAZIONE

From_UC	SEL	r_0	InvertiB
And	And	0	0
Or	Or	0	0
Somma	Add	0	0
Sottr.	Add	1	1

InvertiB e r_0 sono lo stesso segnale, si può ancora ottimizzare.

$r_{in}(0)$ entra solo in ALU₀

A. InvertiB entra in tutte le ALU_i



10

<http://borghese.di.unimi.it/>



Sommario



ALU ad 1 bit

ALU a 32 bit

Comparazione, Overflow, Test di uguaglianza

Tecnologie di costruzione di una ALU



Confronto



Fondamentale per **dirigere il flusso** di esecuzione (test, cicli...)

if a **less_than** b then
 s = 1

if a < b then
 s = 1

if (a - b) < 0 then
 s = 1



Come sviluppare la comparazione - I?



```

if (a - b) < 0 then
  s = 1
  else
  s = 0

```

```

if differenza(a,b) < 0 then
  s = 1
  else
  s = 0

```

Si controlla che il primo bit del risultato della somma (bit di segno) sia = 1.

Esempio: $s = 3 - 4$; su 3 bit

3 -> 011
 -4 -> 100 in complemento a 2
 -1 -> 111 in complemento a 2



Come sviluppare la comparazione - II?

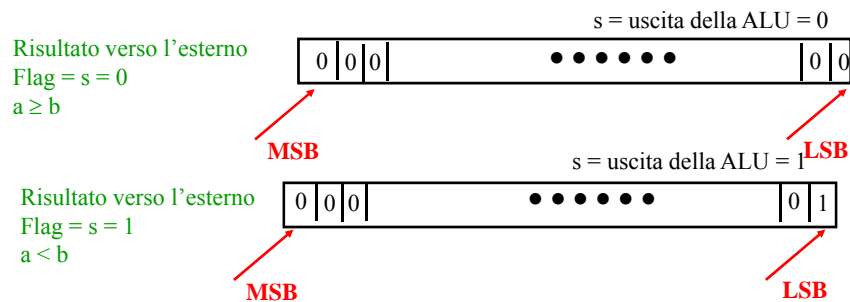


```

if (a - b) < 0 then
  s = 1
  else
  s = 0

```

Viene impostato un flag = 1 se $a < b$.
 Valori possibili in uscita della ALU: {0, 1}

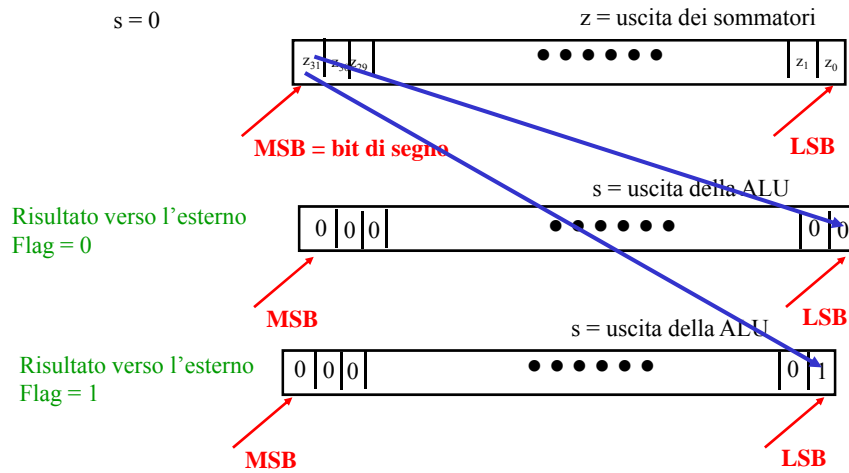




Come sviluppare la comparazione (riassunto)?



if $(a - b) < 0$ then
 $s = 1$
 else
 $s = 0$



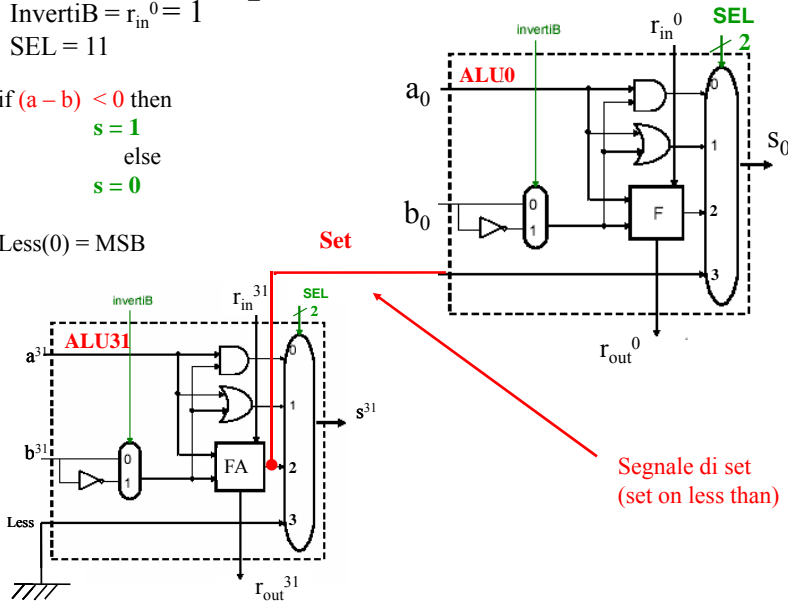
Comparatore - ALU⁰ : ALU³¹



InvertiB = $r_{in}^0 = 1$
 SEL = 11

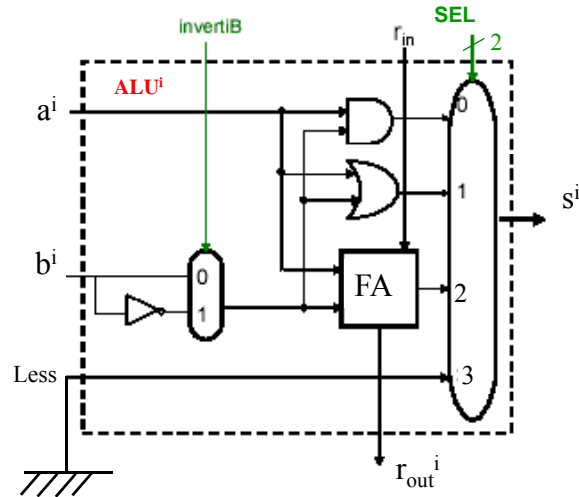
if $(a - b) < 0$ then
 $s = 1$
 else
 $s = 0$

Less(0) = MSB





Comparatore - ALUⁱ



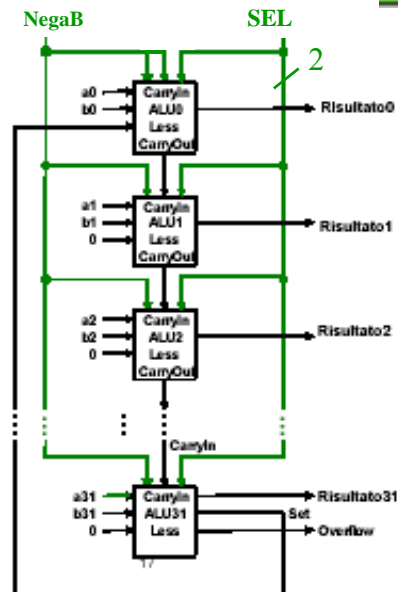
Less(i) = 0 $i = 1, 2, 3, \dots, 31$ $i \neq 0$



Comparazione - ALU completa a 32 bit



- AND
- OR
- SOMMA
- SOTTRAZIONE
- COMPARAZIONE
 - ($a < b$)
 - ($a > b$)





Overflow decimale



$a + b = c$ - codifica su 2 cifre,

$a = 12, b = 83 \Rightarrow$ Not Overflow.

```

012+
083=
-----
095

```

$a = 19, b = 83 \Rightarrow$ Overflow

```

019+
083=
-----
102

```

Si può avere overflow con la differenza?



Overflow binario



$a + b = c$ - codifica su 4 cifre binarie, di cui 1 per il segno.

No overflow

```

0010+  2+
0011=  3=
-----
0101   5

```

```

1010+  -6+
1111=  -1=
-----
1001   -7

```

Overflow

```

0010+  2+
0111=  7=
-----
1001   -7??

```

```

1010+  -6+
1011=  -5=
-----
0101   +5??

```

Quindi si ha overflow nella somma quando:

$a + b = s, \quad a > 0, b > 0 \quad \text{MSB di } a \text{ e } b = 0, \text{ MSB di } s = 1.$

$a + b = s, \quad a < 0, b < 0 \quad \text{MSB di } a \text{ e } b = 1, \text{ MSB di } s = 0.$

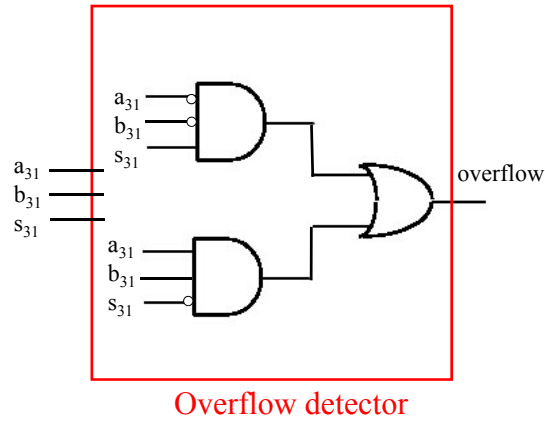


Circuito di riconoscimento dell'overflow



Lavora sui MSB

a_{31}	b_{31}	s_{31}	overflow
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0



$$\text{Overflow} = \neg a_{31} b_{31} s_{31} + a_{31} b_{31} \neg s_{31}$$

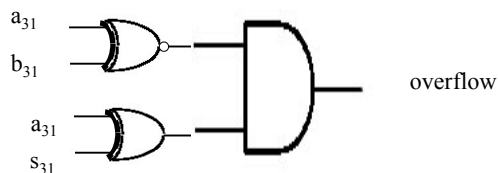


Circuito di riconoscimento dell'overflow



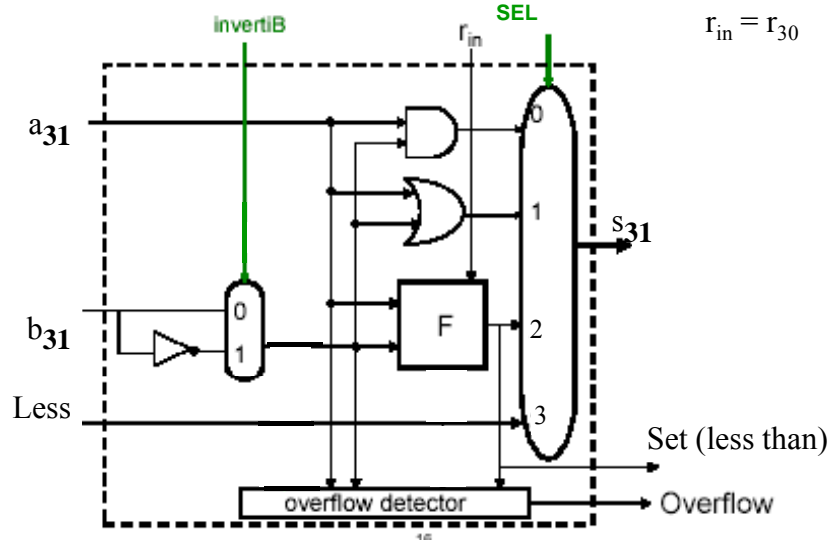
$$\text{Overflow} = (a_{31} \oplus b_{31}) \oplus s_{31}$$

if ($a_{31} == b_{31}$) then
 if ($a_{31} \neq s_{31}$) then
 OVERFLOW





ALU₃₁



Uguaglianza – prima possibilità



beq rs, rt label

iff $(rs - rt) = 0$, salta.

A_0	B_0	C_0	A_1	B_1	C_1
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	1	1	1

$$C = C_0 C_1 \dots C_{n-1}$$

$$C_k = \overline{a_k \oplus b_k}$$

Input: a, b su 32 bit

Output: {0,1}.

Per 32 bit occorrono le seguenti porte a 2 ingressi:

32 XOR

31 AND

Si può fare di meglio sfruttando il sommatore già presente nella ALU



Operazione di uguaglianza



beq rs, rt label iff $(rs - rt) = 0$, salta.

Occorre quindi:

- Impostare una differenza.
- Effettuare l'OR logico di tutti i bit somma.
- L'uscita dell'OR logico = 0 i due numeri sono uguali.

Input: a, b su 32 bit

Output: $\{0,1\}$. In questo caso viene dedicata una linea di uscita.

$$\text{Zero} = \text{NOR}(FA_0, FA_1, FA_{N-1})$$

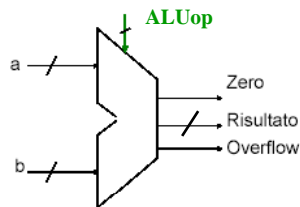


ALU a 32 bit: struttura finale

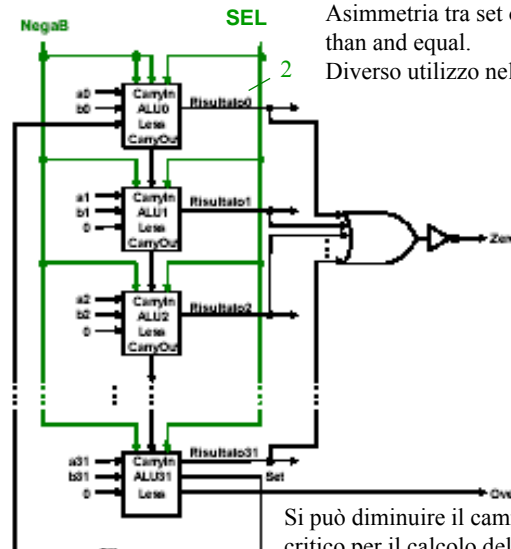


Operazioni possibili:

- AND
- OR
- Somma / Sottrazione
- Comparazione
- Test di uguaglianza



Sono evidenziate
solamente le variabili
visibili all'esterno.



Si può diminuire il cammino critico per il calcolo dello Zero aumentando la complessità



Sommario



ALU ad 1 bit

ALU a 32 bit

Comparazione, Overflow, Test di uguaglianza

Tecnologie di costruzione di una ALU



Approcci tecnologici alla ALU.



Tre approcci tecnologici alla costruzione di una ALU (e di una CPU):

- **Approccio strutturato.** Analizzato in questa lezione.
- **Approccio hardware programmabile (e.g. PAL).** Ad ogni operazione corrisponde un circuito combinatorio specifico.
- **Approccio ROM.** E' un approccio esaustivo (tabellare). Per ogni funzione, per ogni ingresso viene memorizzata l'uscita. E' utilizzabili per funzioni molto particolari (ad esempio di una variabile). Non molto utilizzato.
- **Approccio firmware (firm = stabile), o microprogrammato.** Si dispone di circuiti specifici solamente per alcune operazioni elementari (tipicamente addizione e sottrazione). Le operazioni più complesse vengono sintetizzate a partire dall'algoritmo che le implementa.



Approccio ROM alla ALU



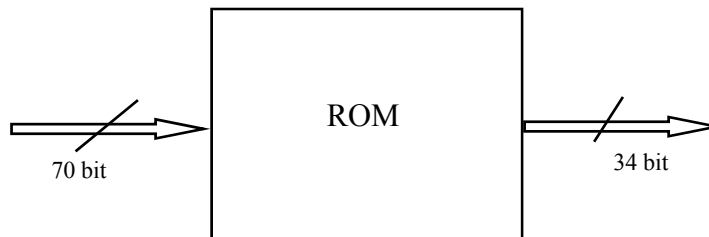
Input:

- A n bit
- B n bit
- SEL k bit
- InvertiB: 1 bit
- Totale: $2*n + k + 1$ bit

Output:

- S n bit
- zero 1 bit
- overflow 1 bit
- Totale: $n + 2$ bit

Per dati su 32 bit, 5 operazioni:



Capacità della ROM: $2^{68} = 2.95.. \times 10^{20}$ parole di 34 bit!

A.A. (Capacità della ROM per dati su 4 bit, 5 operazioni: $2^{12} = 4,098$ parole di 6 bit) di.unimi.it/



L'approccio firmware



Nell'approccio firmware, viene inserita nella ALU una unità di controllo e dei registri. L'unità di controllo attiva opportunamente le unità aritmetiche ed il trasferimento da/verso i registri. Approccio "controllore-datapath".

Viene inserito un microcalcolatore dentro la ALU.

Il primo microprogramma era presente nell'IBM 360 (1964).



L'approccio firmware vs hardware



La soluzione HW è più veloce ma più costosa per numero di porte e complessità dei circuiti.

La soluzione firmware risolve l'operazione complessa mediante una sequenza di operazioni semplici. E' meno veloce, ma più flessibile e, potenzialmente, adatta ad inserire nuove procedure.

La soluzione HW è percorsa per le operazioni frequenti: la velocizzazione di operazioni complesse che vengono utilizzate raramente non aumenta significativamente le prestazioni (legge di Amdahl).



Sommario



ALU ad 1 bit

ALU a 32 bit

Comparazione, Overflow, Test di uguaglianza

Tecnologie di costruzione di una ALU