


La ALU

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

Università degli Studi di Milano


Sommario

Funzione della ALU.

ALU su 1 bit: operazioni logiche + somma.

ALU su 32 bit ed implementazione di sottrazione, confronto e test di uguaglianza.

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.

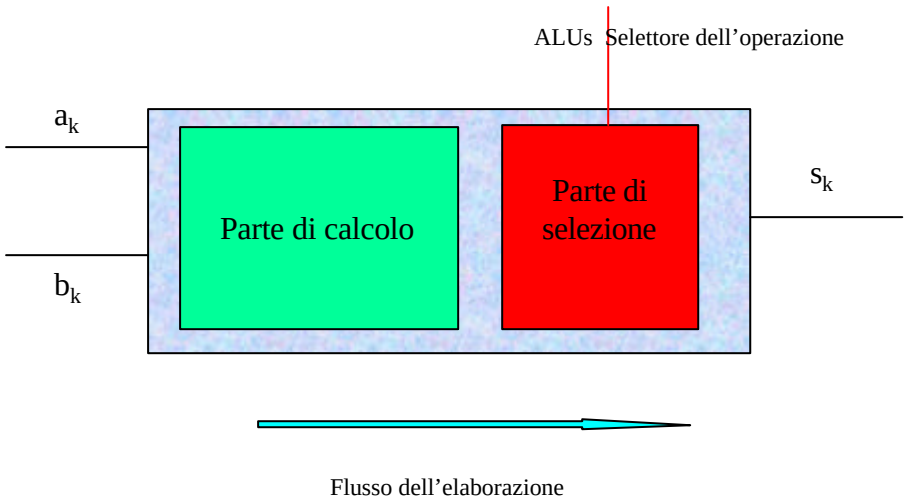
Sommario

Funzione della ALU.

ALU su 1 bit: operazioni logiche + somma.

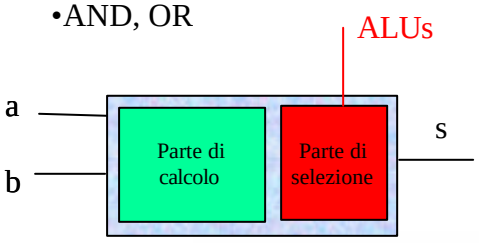
ALU su 32 bit ed implementazione di sottrazione, confronto e test di uguaglianza.

Struttura a 2 livelli di una ALU

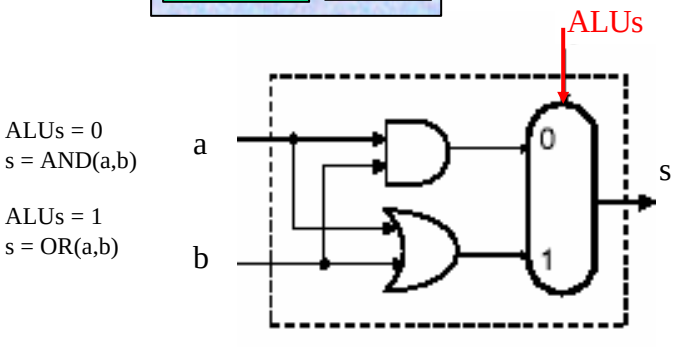


Progettazione della ALU

•AND, OR



a b	a *AND* b	a *OR* b
0 0	0	0
0 1	0	1
1 0	0	1
1 1	1	1



ALUs = 0
s = AND(a,b)

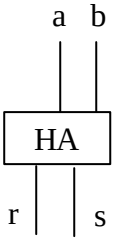
ALUs = 1
s = OR(a,b)

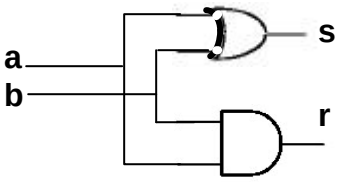
1 porta AND
1 porta OR
1 Mux

(Half) Adder ad 1 bit

Tabella della verità della somma:

a b	somma	riporto
0 0	0	0
0 1	1	0
1 0	1	0
1 1	0	1





$s = a \oplus b$

$r = ab$

La somma è diventata un'operazione logica!

Full Adder ad 1 bit

Tabella della verità della somma completa:

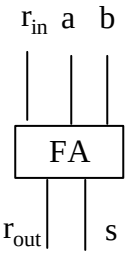
a	b	r_{in}	somma	riporto
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

$$s = m_1 + m_2 + m_4 + m_7$$
$$r = m_3 + m_5 + m_6 + m_7$$

$$s = \bar{a}\,\bar{b}\,r_{in} + a\,\bar{b}\,\bar{r}_{in} + \bar{a}\,b\,\bar{r}_{in} + a\,b\,r_{in} =$$

$$= (a \oplus b)\,\bar{r}_{in} + \overline{(ab + ab)}\,r_{in} =$$

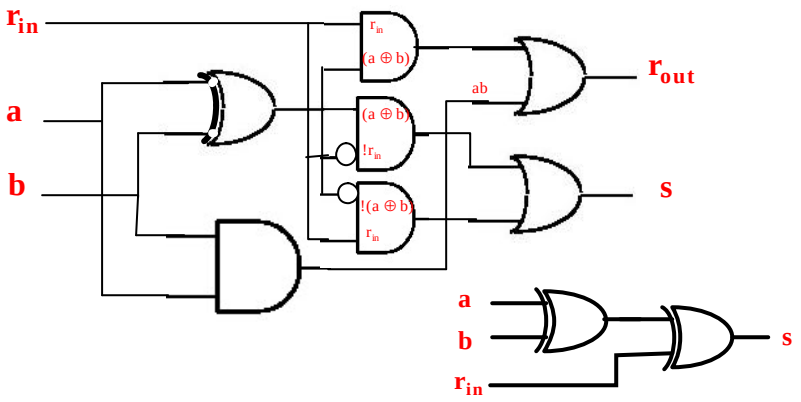
$$= (a \oplus b)\,\bar{r}_{in} + \overline{(a \oplus b)}\,r_{in}$$



$$r_{out} = a\,b\,\bar{r}_{in} + \bar{a}\,b\,r_{in} + a\,\bar{b}\,r_{in} + a\,b\,r_{in} = ab + (a \oplus b)\,r_{in}$$

Implementazione circuitale

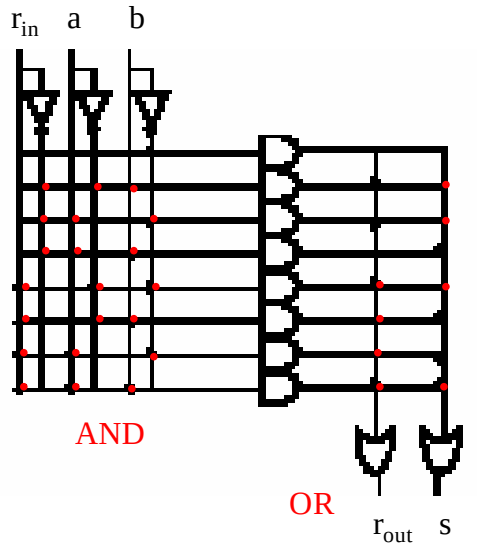
$$s = (a \oplus b)\overline{r_{in}} + \overline{(a \oplus b)}\,r_{in} = (a \oplus b) \oplus r_{in}$$
$$r_{out} = ab + (a \oplus b)r_{in}$$



Implementazione mediante PLA

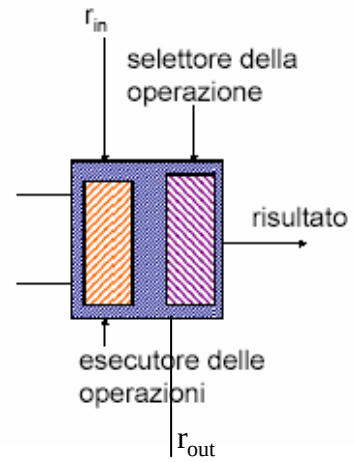
a	b	r_{in}	somma	r_{out}
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

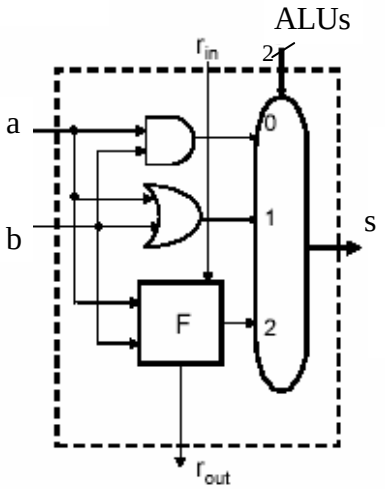
SOP: costruisco i mintermini e li sommo.



La nuova struttura della ALU

- AND
- OR
- SOMMA



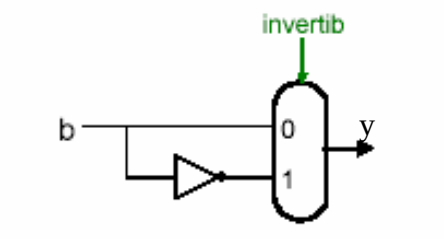


Sottrazione

In complemento a 2 diventa un'addizione: a – b = a + !b + 1

Serve:
a) un inverter (NOT).
b) la costante 1

a)



Iff invertib
y = !b

b) Da dove prendo la costante 1?

Sommario

Funzione della ALU.

ALU su 1 bit: operazioni logiche + somma.

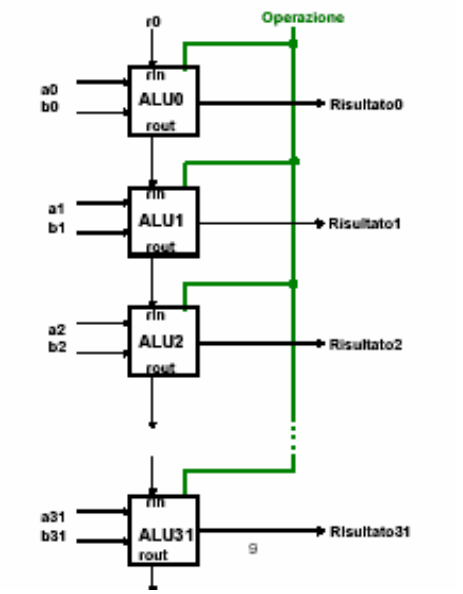
ALU su 32 bit ed implementazione di sottrazione, confronto e test di uguaglianza.

ALU a 32 bit

Come collegare le ALU ad 1 bit?

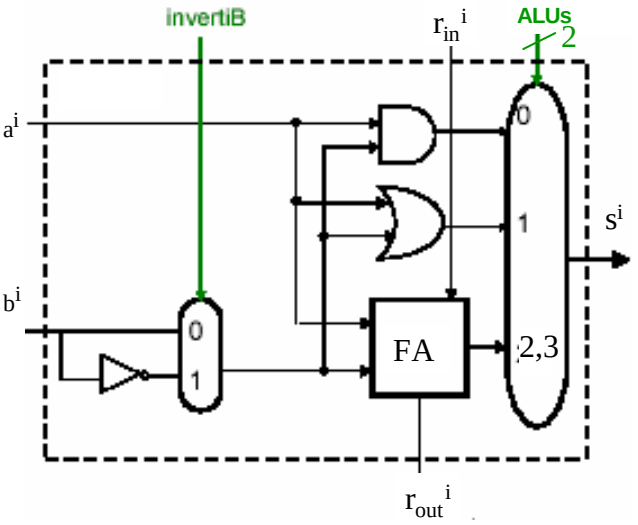
Flusso di calcolo

Perchè non si può parallelizzare?



Sottrazione - ALU_i

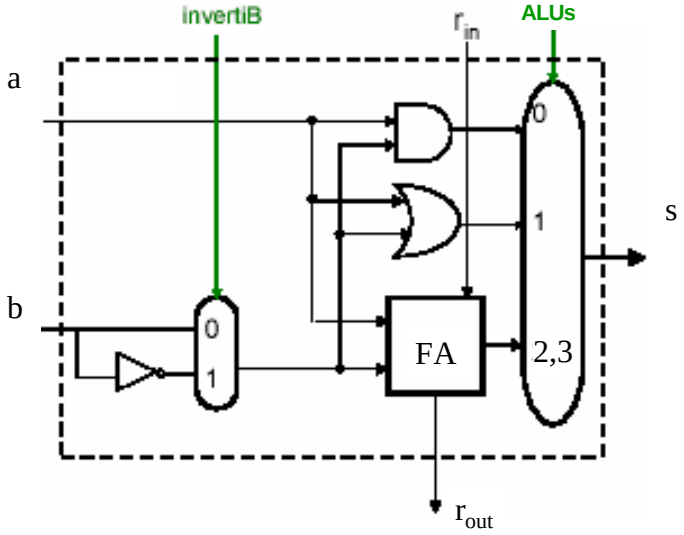
- AND
- OR
- SOMMA
- SOTTRAZIONE



$r_{in}(i) = r_{out}(i-1)$ $i = 1,2,3,....31$ $i \neq 0$

InvertiB = 1 sse ALUs = sottrazione

Sottrazione - ALU_0



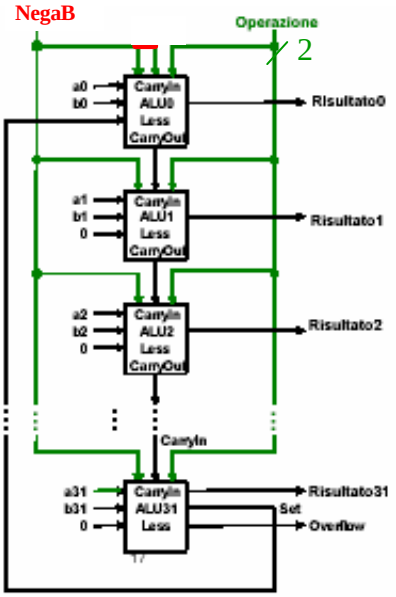
$r_{in}(0)$ = InvertiB = 1 sse ALUs = sottrazione

Sottrazione: ALU a 32 bit

$r_{in}(0)$ = InvertiB = 1
sse ALUs = sottrazione

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

- AND
- OR
- SOMMA
- SOTTRAZIONE



Confronto

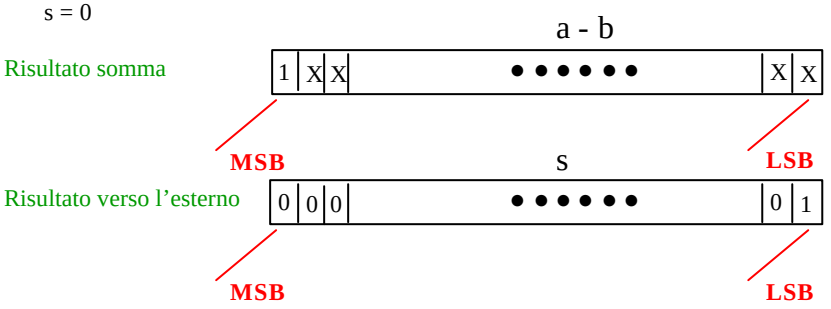
Fondamentale per dirigere il flusso di esecuzione (test, cicli....)

if a < b then
s = 1

if (a – b) < 0 then
s = 1

Come sviluppare la comparazione?

if (a – b) < 0 then
s = 1
else
s = 0



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

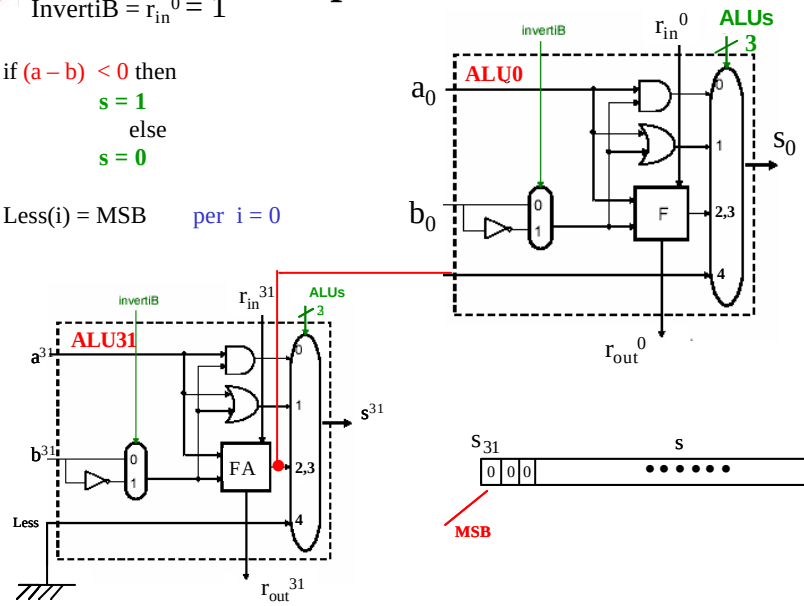
Occorre quindi:
- Impostare una differenza.
- Inviare l'uscita del sommatore del MSB all'input LESS di ALU_0.
(L'uscita del MSB viene anche chiamata segnale di *set*).

Comparatore – ALU^0

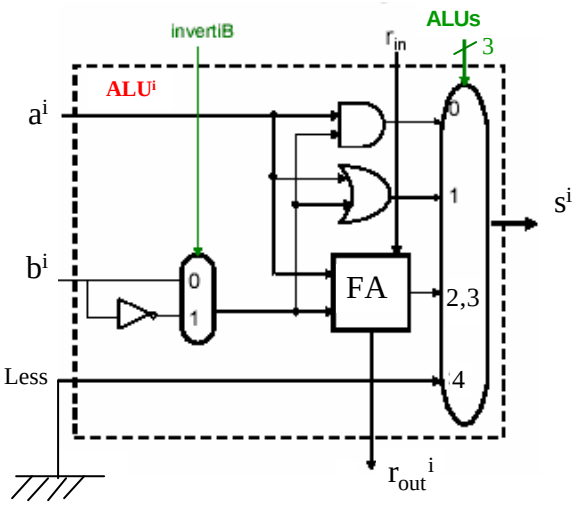
InvertiB = $r_{in}^{0} = 1$

if (a – b) < 0 then
s = 1
else
s = 0

Less(i) = MSB per i = 0



Comparatore - ALU^i



Less(i) = 0 i = 1,2,3, ...31 i ? 0

Overflow

Esempio decimale:
a + b = c - codifica su 2 cifre, a = 19, b = 83 => Overflow.

19 + 83 = (1)02

Supponiamo sia definito il bit di segno possiamo riscrivere:

019 + 083 = 102.

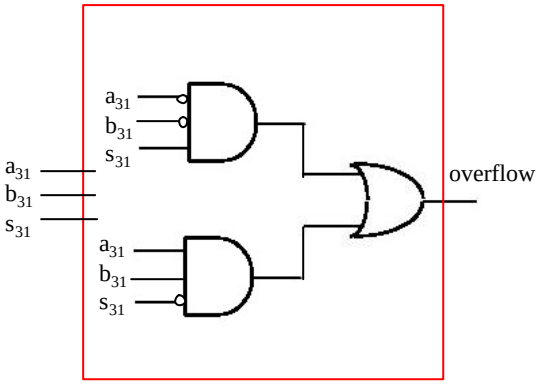
Quindi si ha overlow nella somma quando:
a + b = s, a > 0, b > 0 MSB di a e b = 0, MSB di s = 1.
a + b = s, a < 0, b < 0 MSB di a e b = 1, MSB di s = 0.

Si può avere overflow con la differenza?

Circuito di riconoscimento dell'overflow

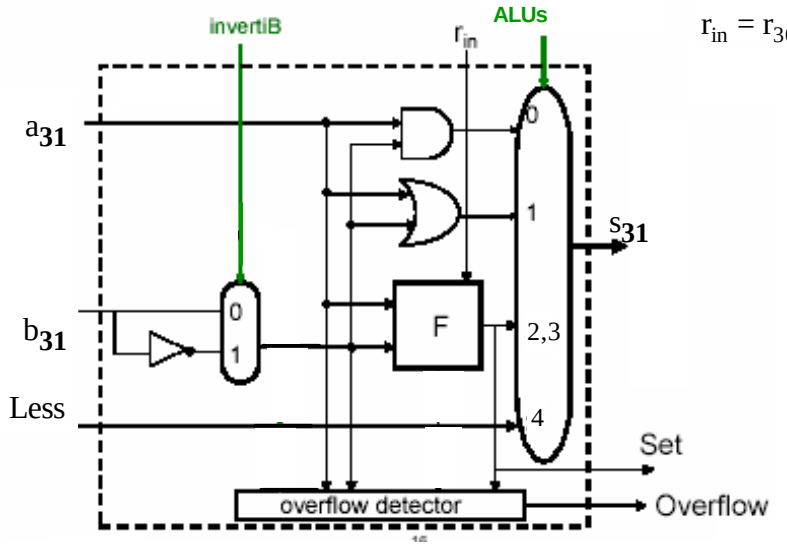
Lavora sui MSB

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



Overflow detector

ALU_{31}



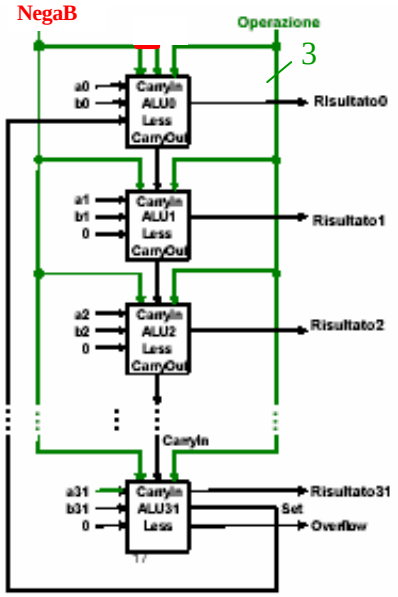
$r_{in} = r_{30}$

Comparazione - ALU completa a 32 bit

$r_{in}(0)$ = InvertiB = 1
sse ALUs = sottrazione

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

- AND
- OR
- SOMMA
- SOTTRAZIONE
- COMPARAZIONE



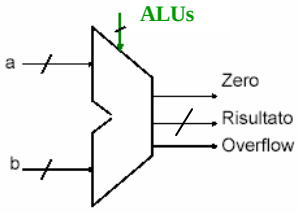
Operazione di uguaglianza

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

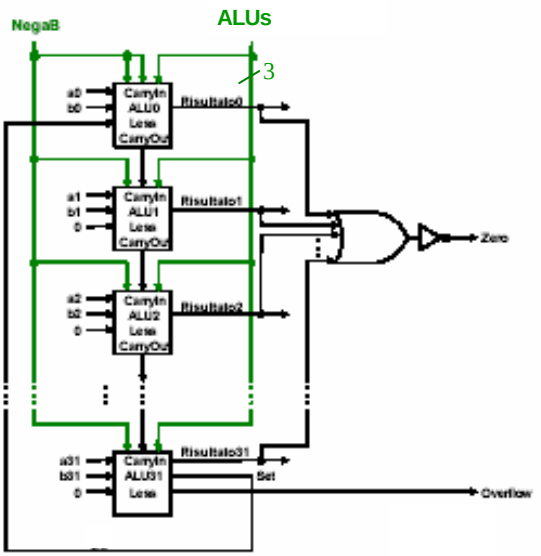
Occorre quindi:
- Impostare una differenza.
- Effettuare l'OR logico di tutti i bit somma.
- L'uscita dell'OR logico = 0 i due numeri sono uguali.

ALU a 32 bit: struttura finale

Operazioni possibili:
- AND
- OR
- Somma / Sottrazione
- Comparazione
- Test di uguaglianza



Sono evidenziate solamente le variabili visibili all'esterno.



Sommario

Funzione della ALU.

ALU su 1 bit: operazioni logiche + somma.

ALU su 32 bit ed implementazione di sottrazione, confronto e test di uguaglianza.