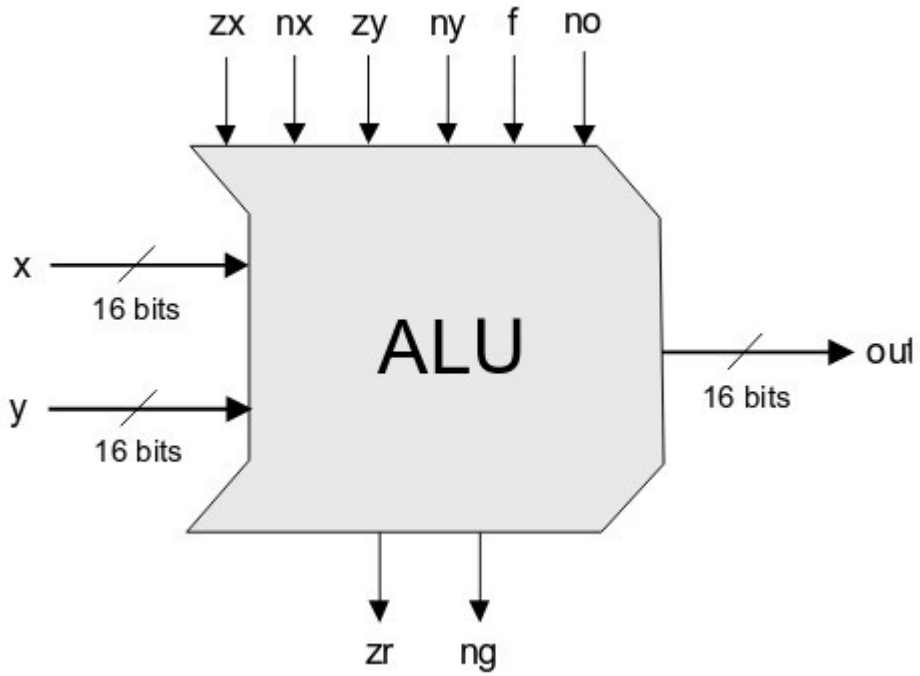
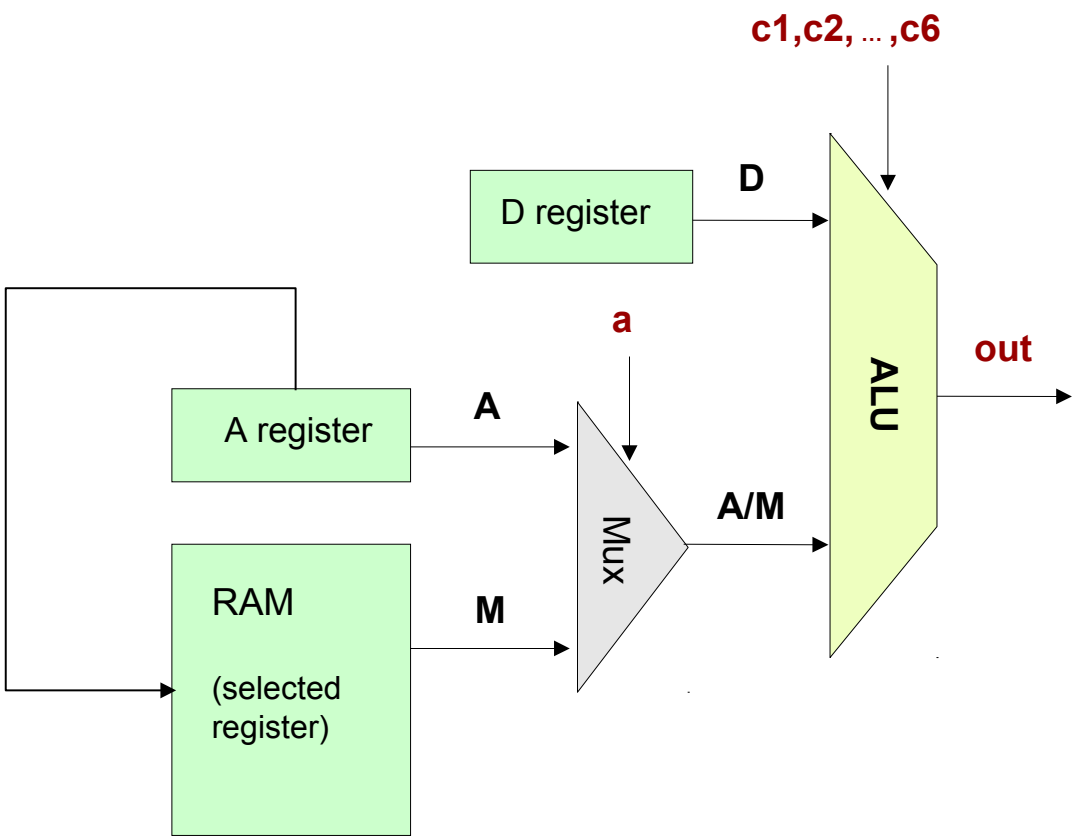
La ALU dell'architettura Hack



Prof. Ivan Lanese

L'architettura del processore Hack



- La ALU esegue diversi tipi di operazioni, di volta in volta identificate dai "control bit" (c1, c2,..,c6), su due parole da 16 bit. Il risultato (una parola da 16 bit) viene trasmesso sul bus "out"

Richiamo riguardo alle somme binarie

Assumendo un sistema a 4 bit (senza segno):

```
  0 0 0 1          1 1 1 1
    1 0 0 1          1 0 1 1
  + 0 1 0 1        + 0 1 1 1
  ---------        ---------
  0 1 1 1 0        1 0 0 1 0
```

no overflow

overflow

- Algoritmo: esattamente quello delle somme decimali

Rappresentazione numeri in complemento a 2

0	0000		
1	0001	1111	-1
2	0010	1110	-2
3	0011	1101	-3
4	0100	1100	-4
5	0101	1011	-5
6	0110	1010	-6
7	0111	1001	-7
		1000	-8

- I numeri vengono rappresentati in complemento a 2 (qui un esempio a 4 bit, HACK ha invece un'architettura a 16 bit)
- La codifica dei numeri positivi inizia con 0
- La codifica dei numeri negativi inizia con 1
- Metodo standard di conversione del segno: complemento bit-a-bit e somma 1
- Metodo veloce di conversione del segno: partendo da destra si tengono tutti gli 0 ed il primo 1, si complementa il resto

Esempio: 2 - 5 = 2 + (-5) =

$$\begin{array}{r} 0\,0\,1\,0 \\ +\,1\,0\,1\,1 \\ \hline 1\,1\,0\,1 \end{array} \quad = \; -3$$

Trucco per eseguire sottrazioni in complemento a 2

0	0000		
1	0001	1111	-1
2	0010	1110	-2
3	0011	1101	-3
4	0100	1100	-4
5	0101	1011	-5
6	0110	1010	-6
7	0111	1001	-7
		1000	-8

- Abbiamo visto che il cambio di segno del numero x si ottiene facendo !x+1, dove !x è il complemento bit-a-bit del numero x
- Ora consideriamo x-y:
  - x-y =
    -(-x+y) =
    -(!x+1+y) =
    -(!x+y)-1 =
    !(!x+y)+1-1 =
    !(!x+y)

Esempio: 2 - 5 = !(!2+5) = 1 1 0 1

+ 0 1 0 1

(1) 0 0 1 0 ! 1 1 0 1 = -3

Tecnica per eseguire incrementi

0	0000		
1	0001	1111	-1
2	0010	1110	-2
3	0011	1101	-3
4	0100	1100	-4
5	0101	1011	-5
6	0110	1010	-6
7	0111	1001	-7
		1000	-8

- Abbiamo visto che possiamo calcolare x-y come !(!x+y)
- Ora consideriamo:
  - x+1 =
    x-(-1) =
    !(!x+(-1))
- Per verificare:
  - -x = !x + 1 implica !x =-x - 1

<u>Esempio:</u> -4 + 1= !(!(-4)+(-1)) = 0 0 1 1

+ 1 1 1 1

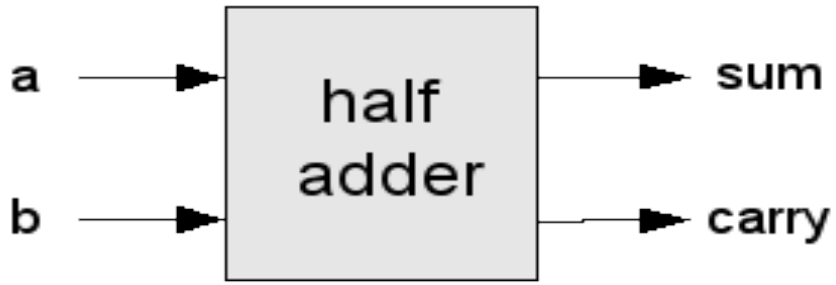
(1) 0 0 1 0 ! 1 1 0 1 = -3

Adder

- Adder: circuito progettato per sommare due interi
- Implementazione usuale:
  - Half adder: progettato per sommare 2 bit
  - Full adder: progettato per sommare 3 bits
  - Adder: progettato per sommare 2 interi da *n* bit

Half adder (progettato per sommare 2 bit)

a	b	sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

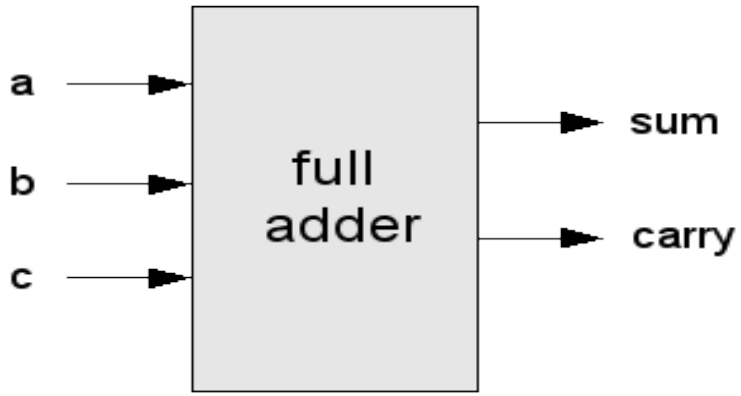


Implementazione:

approccio standard (forma canonica, mappa di Karnaugh)
oppure verificare se esistono porte classiche equivalenti

Full adder (progettato per sommare 3 bit)

a	b	c	sum	carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



Implementazione:

approccio standard (forma canonica, mappa di Karnaugh)

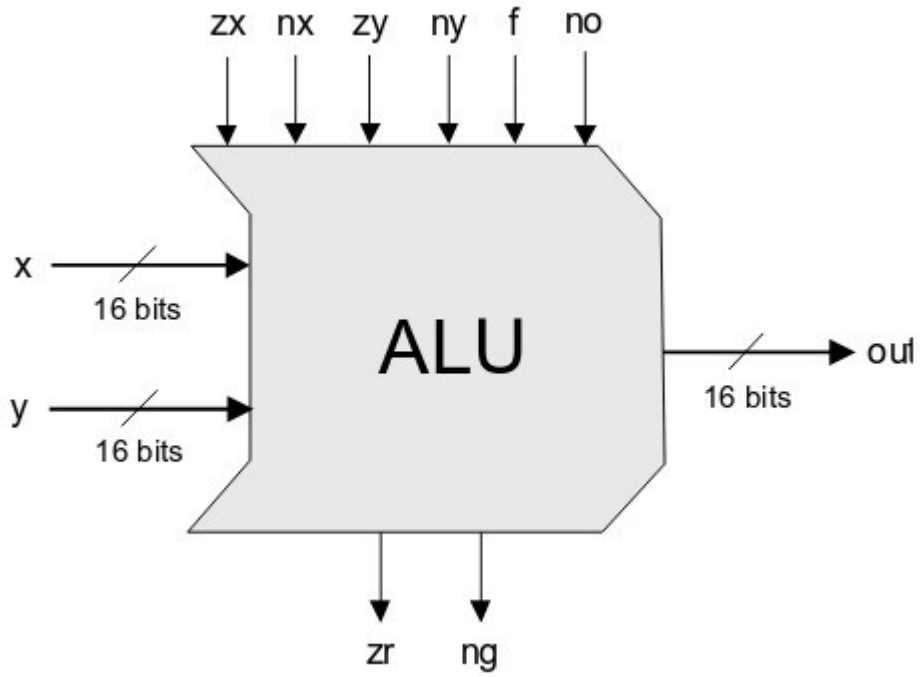
oppure utilizzare degli half adder

n-bit Adder (per l'architettura HACK somma due interi da 16 bit)

...	1	0	1	1	a
					+
...	0	0	1	0	b
...	1	1	0	1	out

Implementazione: Half Adder seguito da sequenza di Full Adder

La ALU (architettura Hack)



out(x, y, control bit) =

x+y, x-y, y-x,

0, 1, -1,

x, y, -x, -y,

!x, !y,

x+1, y+1, x-1, y-1,

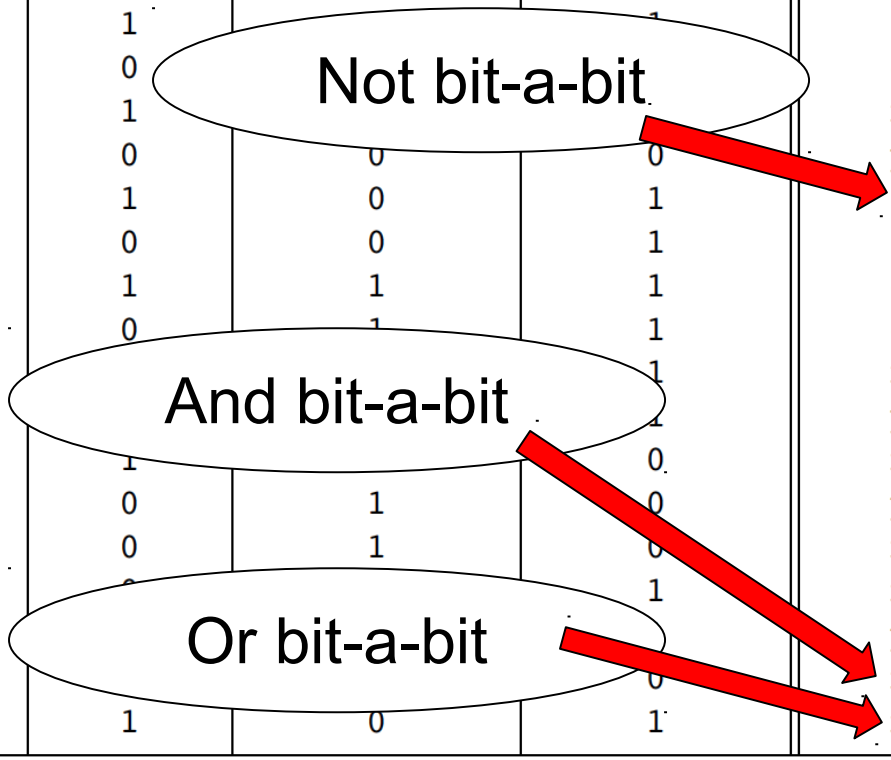
x&y, x|y

Uso dei control bit (architettura Hack)

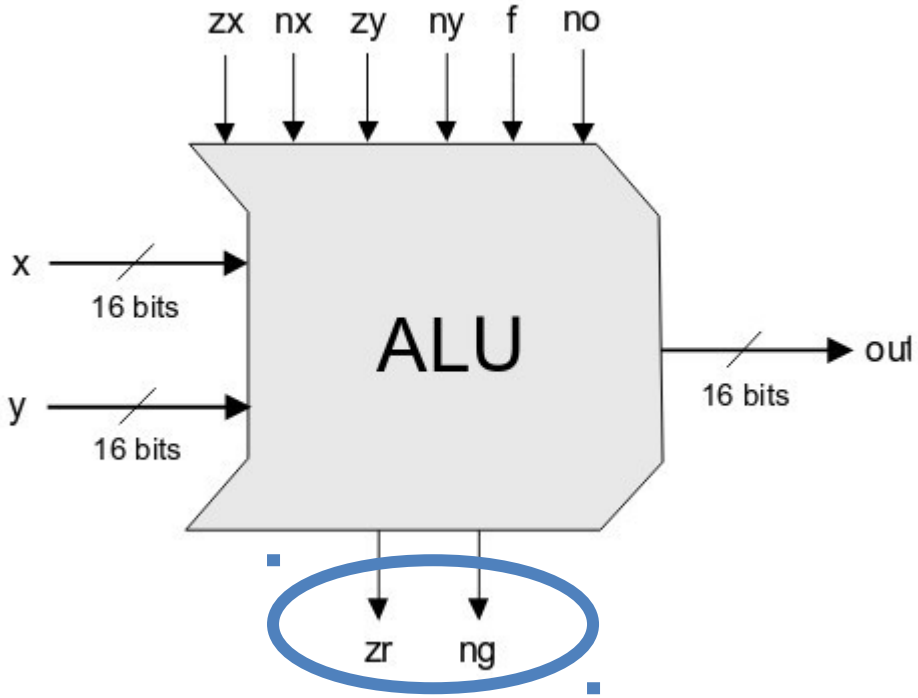
These bits instruct how to preset the x input		These bits instruct how to preset the y input		This bit selects between + / And	This bit inst. how to postset out	Resulting ALU output
zx	nx	zy	ny	f	no	out=
if zx then x=0	if nx then x=!x	if zy then y=0	if ny then y=!y	if f then out=x+y else out=x&y	if no then out=!out	f(x,y)=
1	0	1	0	1	0	0
1	1	1	1	1	1	1
1	1	1	0	1	0	-1
0	0	1	1	0	0	x
1	1	0	0	0	0	y
0	0	1	1	0	1	!x
1	1	0	0	0	1	!y
0	0	1	1	1	1	-x
1	1	0	0	1	1	-y
0	1	1	1	1	1	x+1
1	1	0	1	1	1	y+1
0	0	1	1	1	0	x-1
1	1	0	0	1	0	y-1
0	0	0	0	1	0	x+y
0	1	0	0	1	1	x-y
0	0	0	1	1	1	y-x
0	0	0	0	0	0	x&y
0	1	0	1	0	1	x|y

Uso dei control bit (architettura Hack)

These bits instruct how to preset the x input		These bits instruct how to preset the y input		This bit selects between + / And	This bit inst. how to postset out	Resulting ALU output
zx	nx	zy	ny	f	no	out=
if zx then x=0	if nx then x=!x	if zy then y=0	if ny then y=!y	if f then out=x+y else out=x&y	if no then out=!out	f(x,y)=
1	0	1	0	1	0	0
1	1	1	1	1	1	1
1	1	1	0	1	0	-1
0	0	1	1	0	0	x
1	1	0	0	0	0	y
0	0	1	1	0	1	!x
1	1	0	0	0	1	!y
0	0	1	1	1	1	-x
1	1	0	0	1	1	-y
0	1	1	1	1	1	x+1
1	1	0	0	1	1	y+1
0	0	1	1	0	0	x-1
1	1	0	0	1	0	y-1
0	0	0	0	1	0	x+y
0	1	0	1	0	1	x-y
0	0	0	1	0	1	y-x
0	0	0	0	0	0	x&y
0	1	0	1	0	1	x|y



I bit di stato della ALU (architettura Hack)



La ALU ha due bit di stato:
zr: se out==0 allora zr=1, altrimenti zr=0
ng: se out<0 allora ng=1, altrimenti ng=0

Questi bit verranno usati quando completeremo la realizzazione del processore Hack