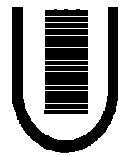


Architettura dei Calcolatori

Prof. Enrico Nardelli



Università degli Studi di Roma “Tor Vergata”

Virtual CPU (Eniac): parte 4

In dettaglio...

Vedremo cosa accade all' interno di vCPU
durante l'esecuzione di diverse operazioni :

Load (N.B. 'LOAD 0' non setta il bit ZE a 1)

Somma

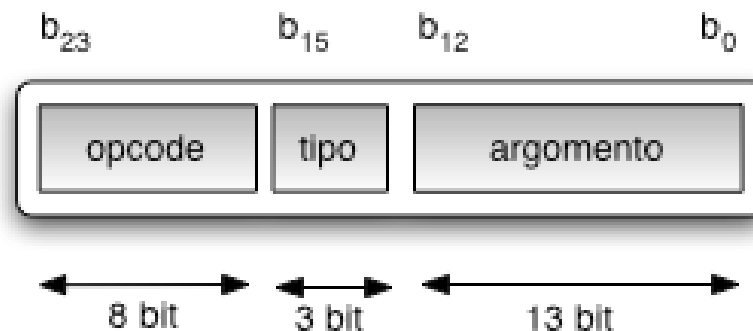
Prodotto

Store

Ricordiamo il formato dell'istruzione

- ❑ Fissato a 24 bit.
- ❑ 8 bit dedicati al codice operativo ($b_{23} \dots b_{16}$).
 - $b_{23} = 1$ istruzione di tipo α o β .
- ❑ 3 bit per il tipo dell'argomento ($b_{15} \dots b_{13}$).
- ❑ 13 per l'argomento ($b_{12} \dots b_0$).

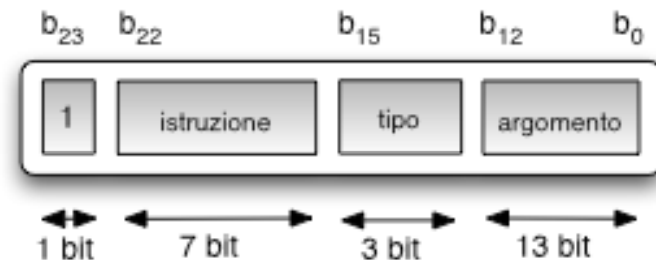
Formato dell'istruzione



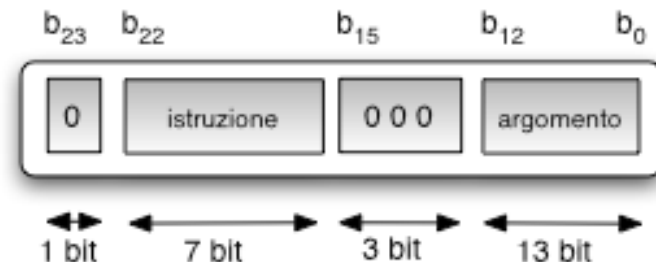
Ricordiamo le 2 Classi di istruzioni

- La differenza tra le due categorie viene formalizzata per mezzo della specifica di due classi di istruzioni: la classe α e la classe β .
- Le due classi si distinguono per mezzo dell'MSB dell'istruzione: il bit b_{23} .
- Nella classe β i bit $b_{15} \dots b_{13}$ sono sempre assegnati a zero in quanto inutilizzati. La tipologia dell'argomento è insita nel codice operativo.

Classe α



Classe β



Esempio : LOAD

LOAD 'numero'

LOAD 5 $\rightarrow$ 1 01 00 000 000 000000000000101

$b_{23} = 1$ istruzioni di tipo α

$b_{22}b_{21}$ = categoria

$b_{20} \dots b_{16}$ codificano la specifica operazione all'interno della categoria (per questa categoria si usano solo i bit b_{20} b_{19})

	b_{22}	b_{21}
Prodotti	1	1
Somme	1	0
Memorizzazione	0	1
Logiche ed altro	0	0

	b_{20}	b_{19}
ST	1	1
LD	0	0

Istruzioni classe α

(tipo di argomento variabile)

Codifica del tipo di argomento

- Come detto la tipologia dell'argomento delle istruzioni di tipo α viene specificata dai 3 bit di tipo $b_{15} \dots b_{13}$
- b_{15} identifica la natura dell'argomento dell'istruzione:
 - 1 indica che l'argomento è un nome di registro
 - 0 indica che è un numero naturale
- Gli altri due bit, $b_{14}b_{13}$, assumono di conseguenza significati diversi in base alla natura dell'argomento

Esempio : LOAD @

LOAD @5

LOAD @5 → 1 01 00 000 001 000000000000101

LOAD @BX → 1 01 00 000 101 000000000000001

$b_{23} = 1$ istruzioni di tipo α

$b_{22}b_{21}$ = categoria

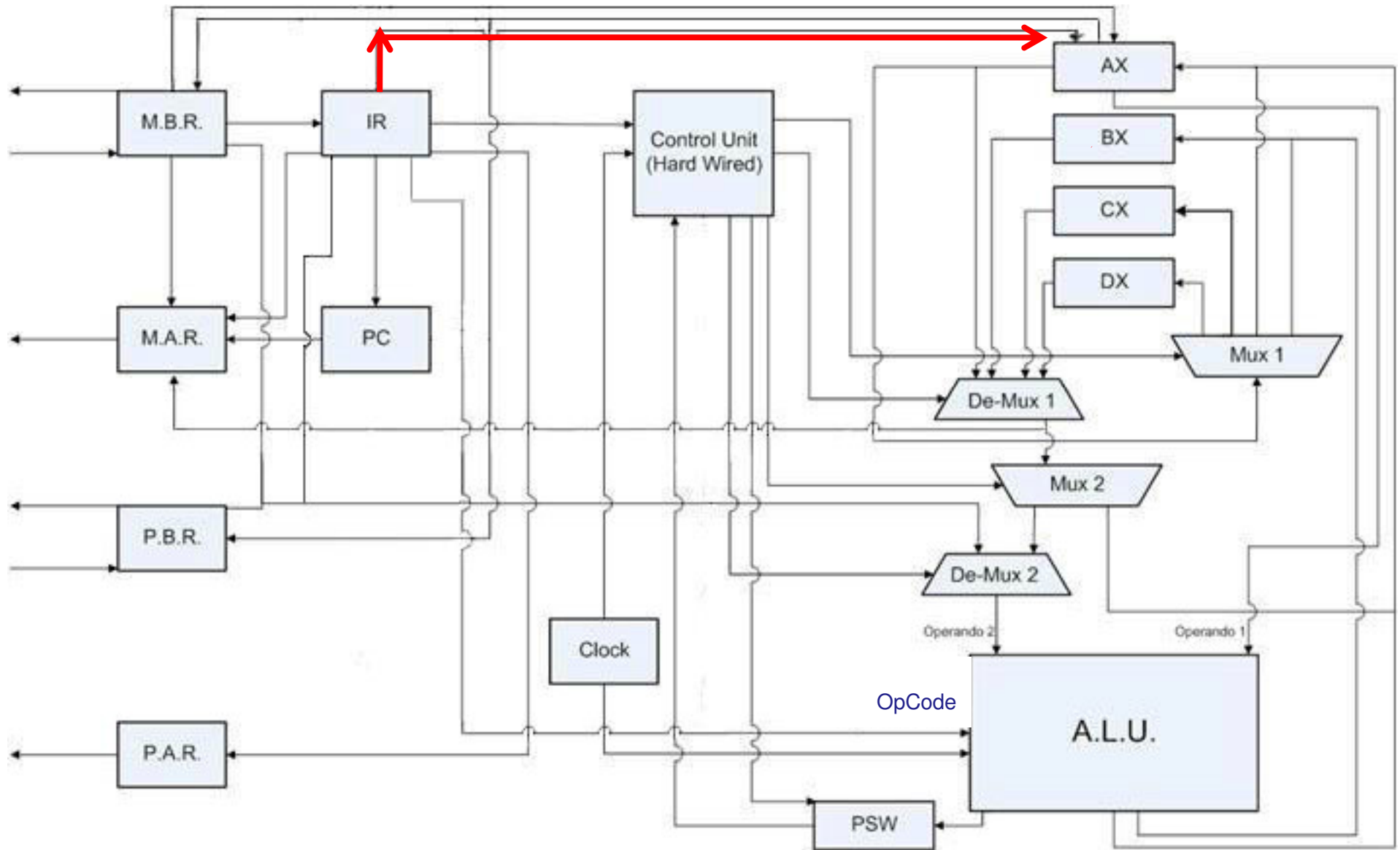
$b_{20}...b_{16}$ codificano la specifica operazione all'interno della categoria (per questa categoria si usano solo i bit b_{20} b_{19})

	b_{22}	b_{21}
Prodotti	1	1
Somme	1	0
Memorizzazione	0	1
Logiche ed altro	0	0

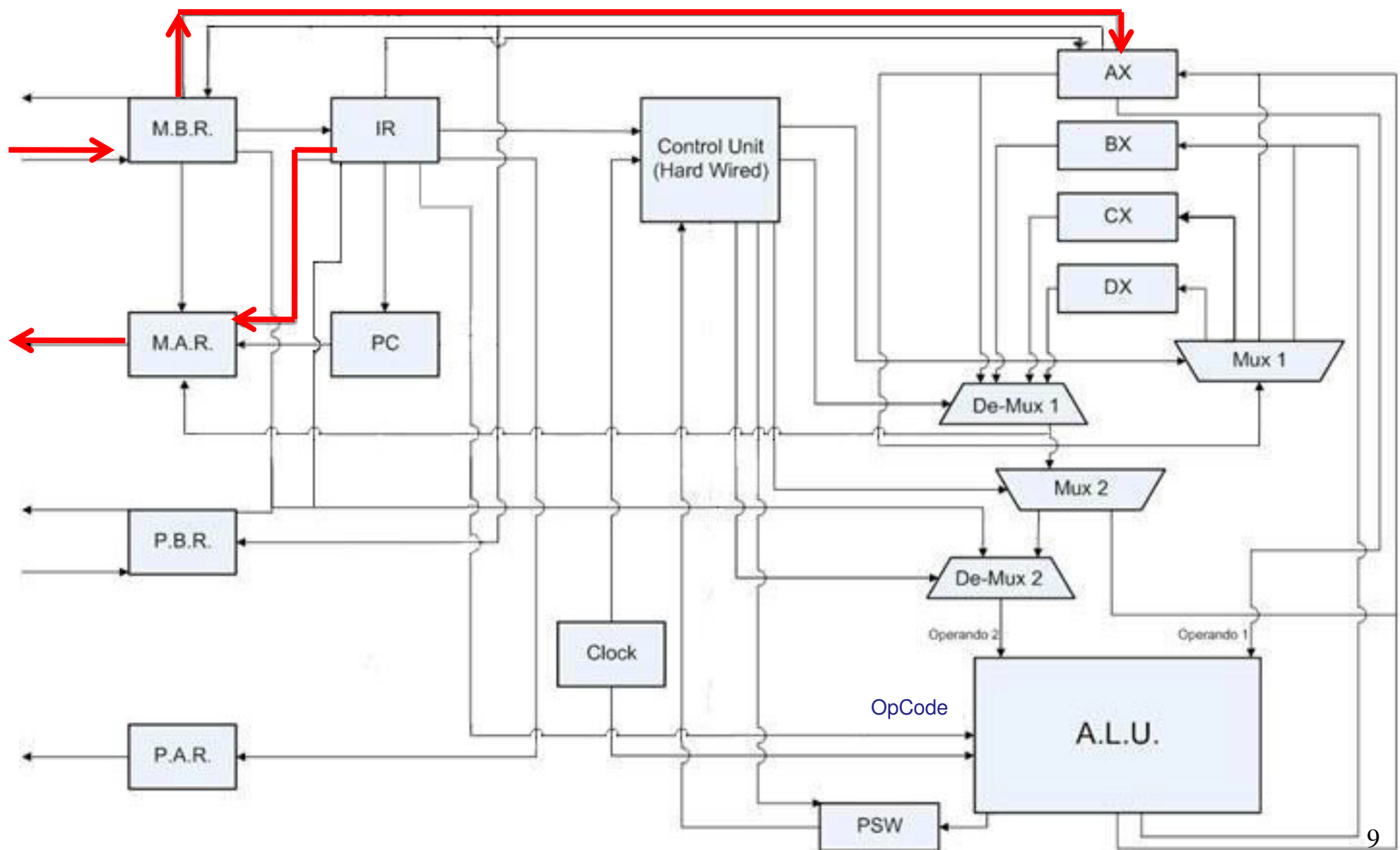
	b_{20}	b_{19}
ST	1	1
LD	0	0

	b_{12}	$b_{11}...b_3$	b_2	b_1	b_0
AX	0	0...0	0	0	0
BX	0	0...0	0	0	1
CX	0	0...0	1	1	0
DX	0	0...0	1	1	1
PC	0	1...1	1	1	1

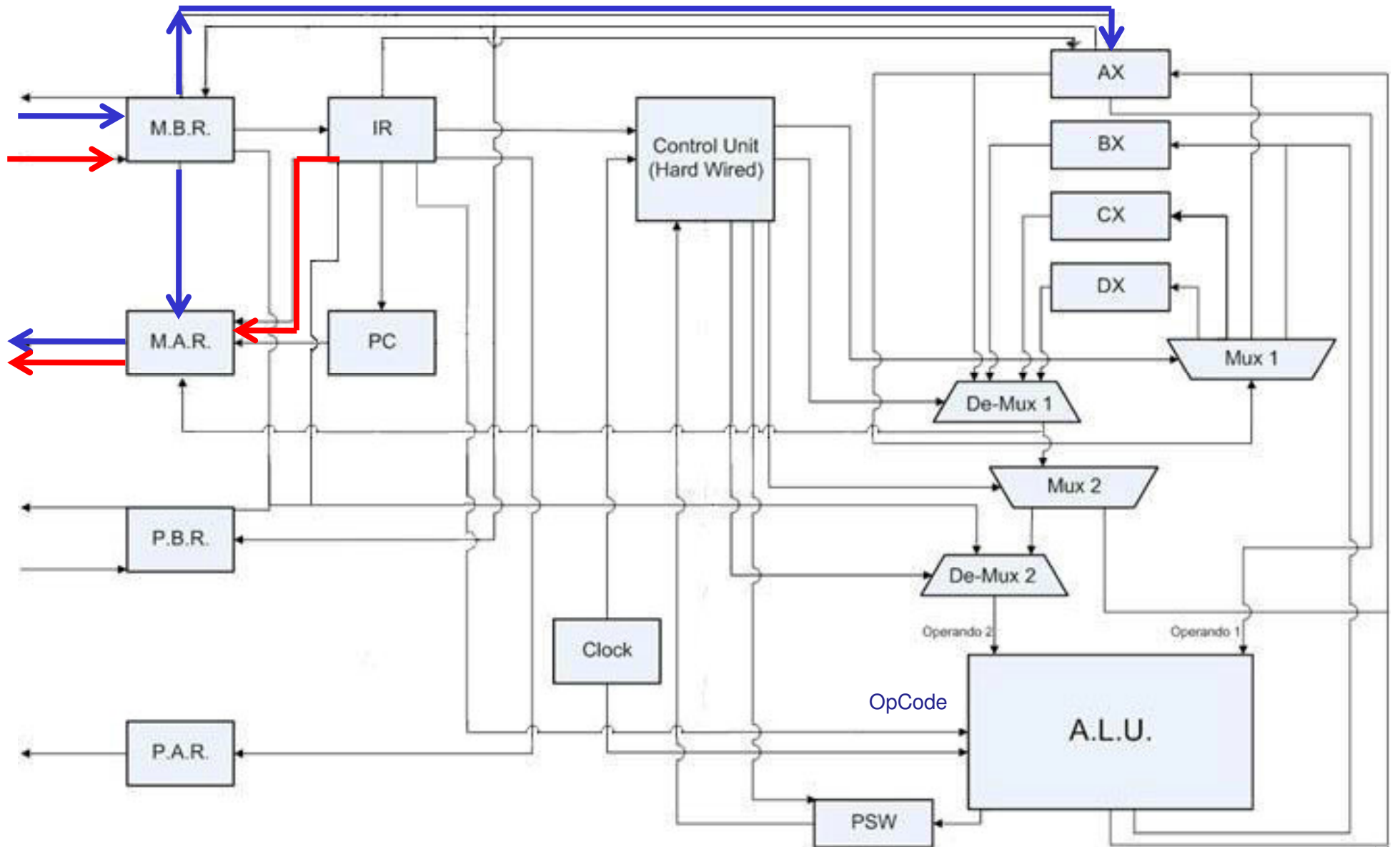
LOAD 5



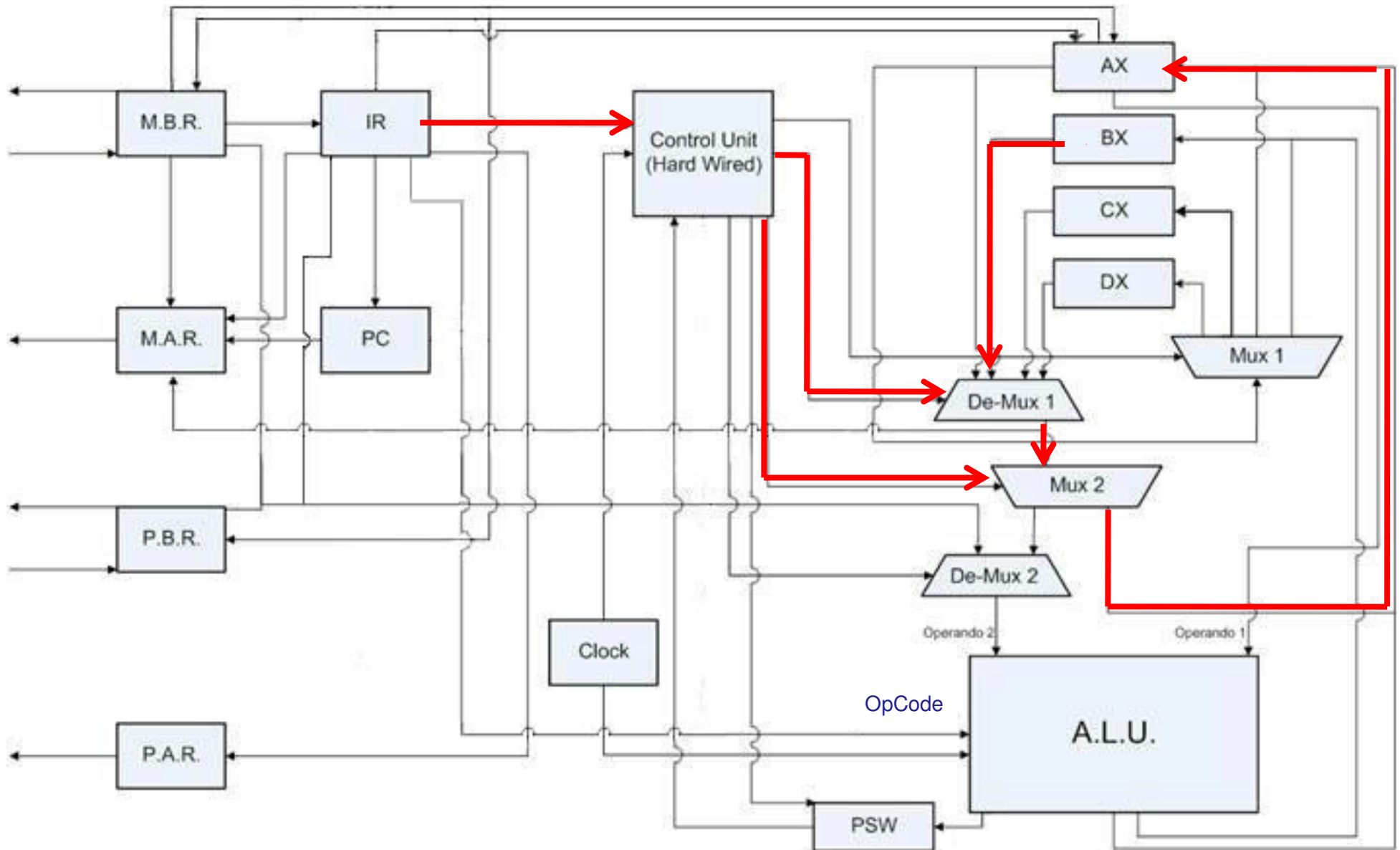
LOAD @5



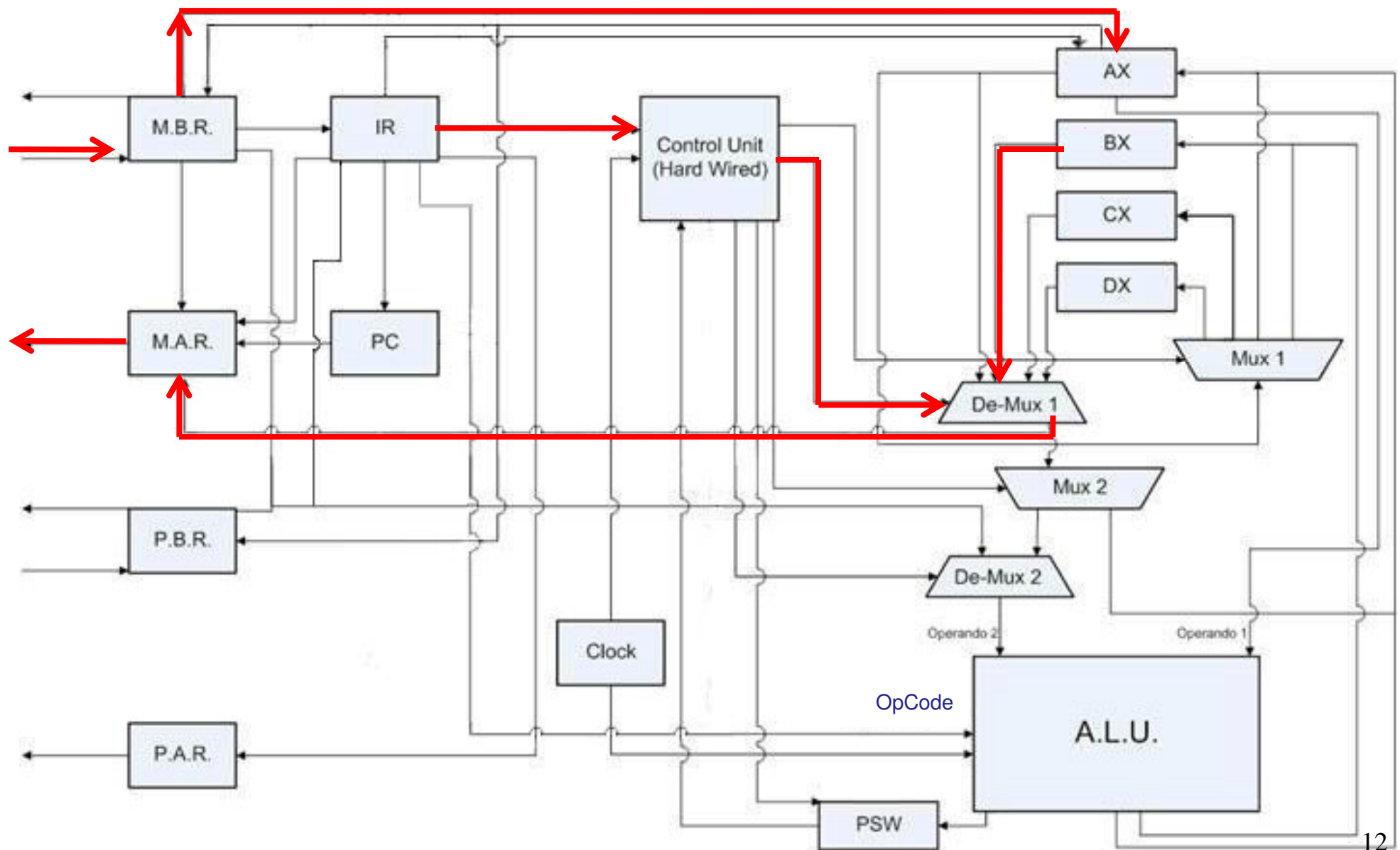
LOAD @@5



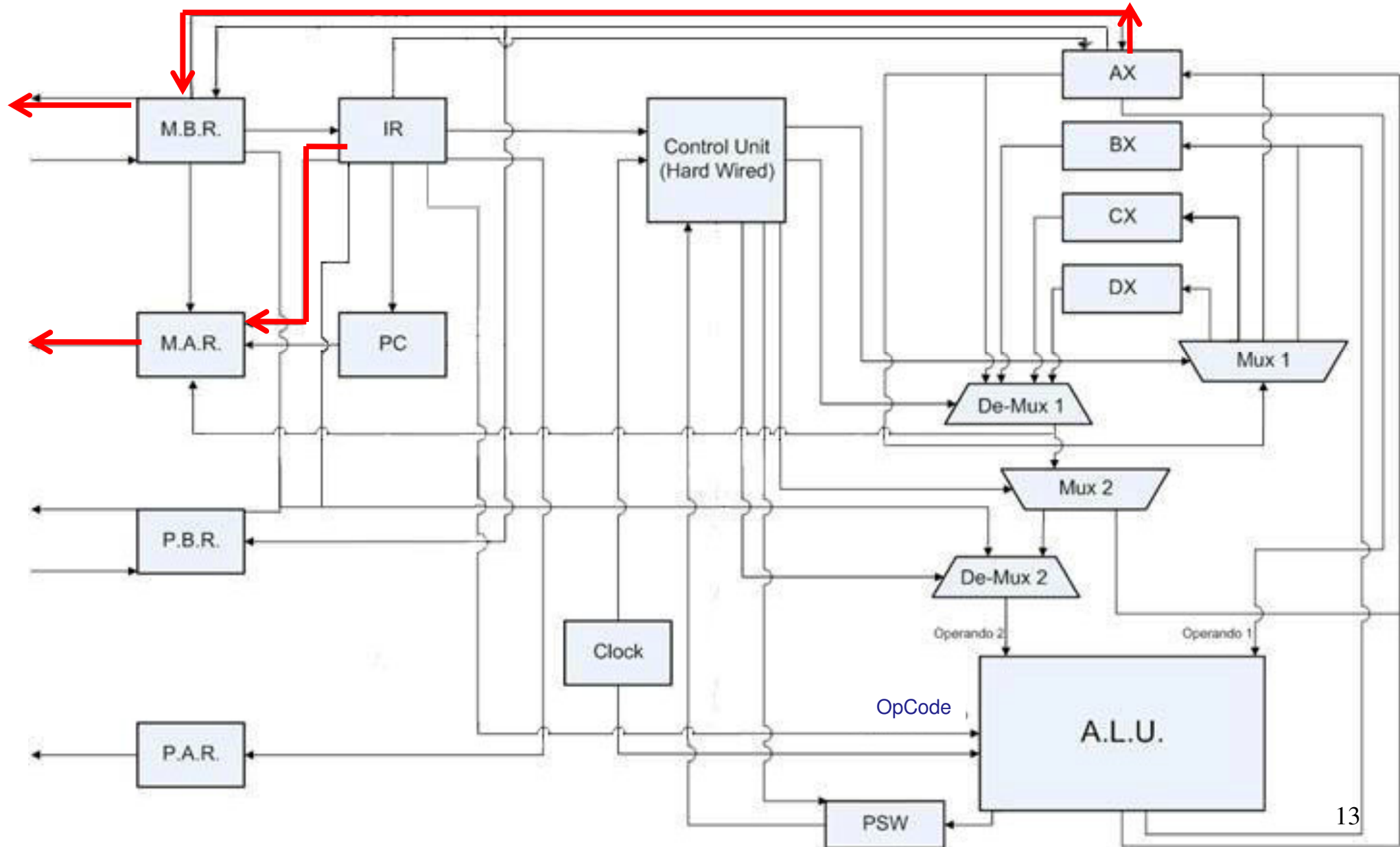
LOAD BX



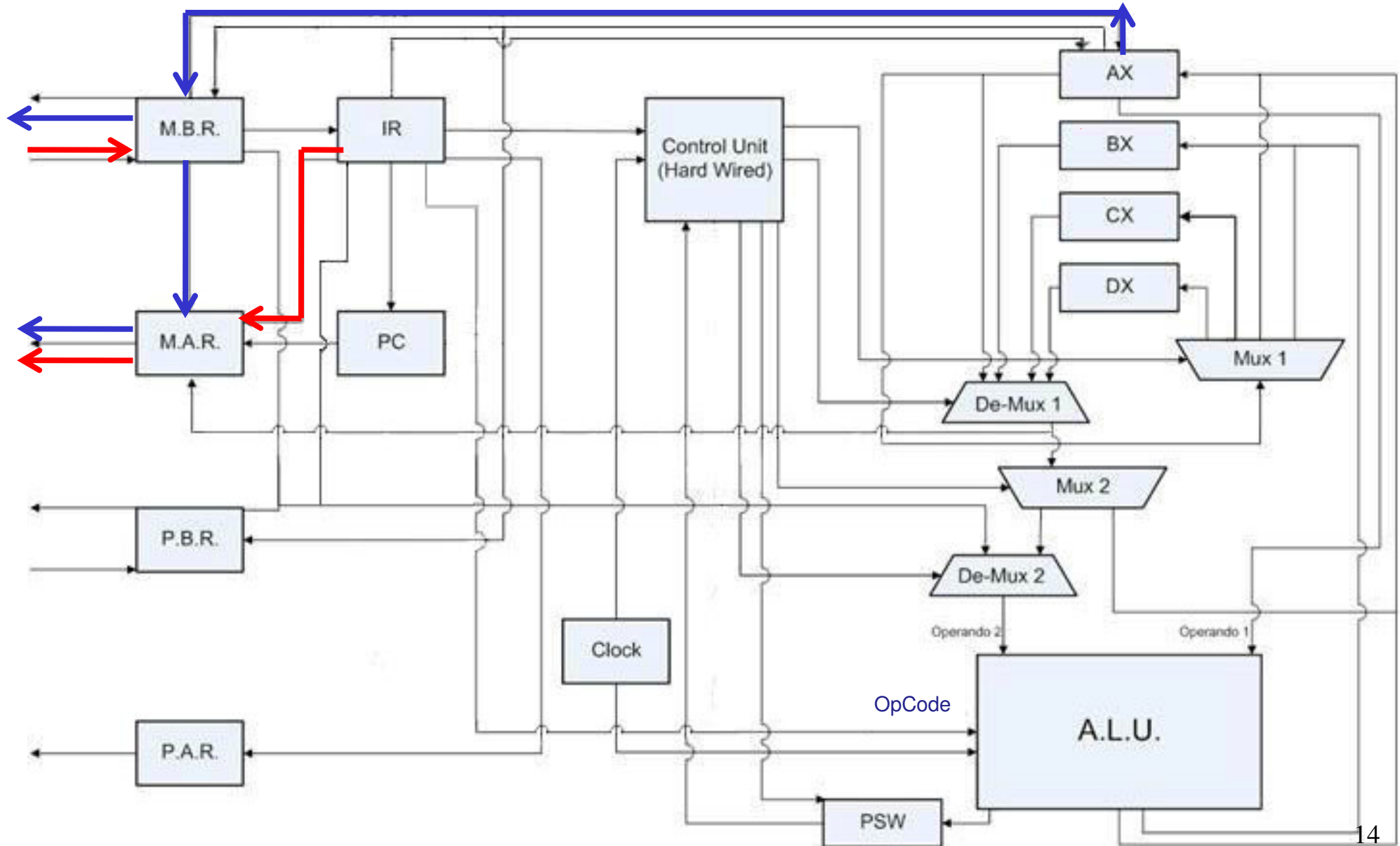
LOAD @BX



STORE @6



STORE @@6



Esempio : SOMMA

ADD 7 → 1 10 01 000 000 000000000000111

$b_{23} = 1$ istruzioni di tipo α

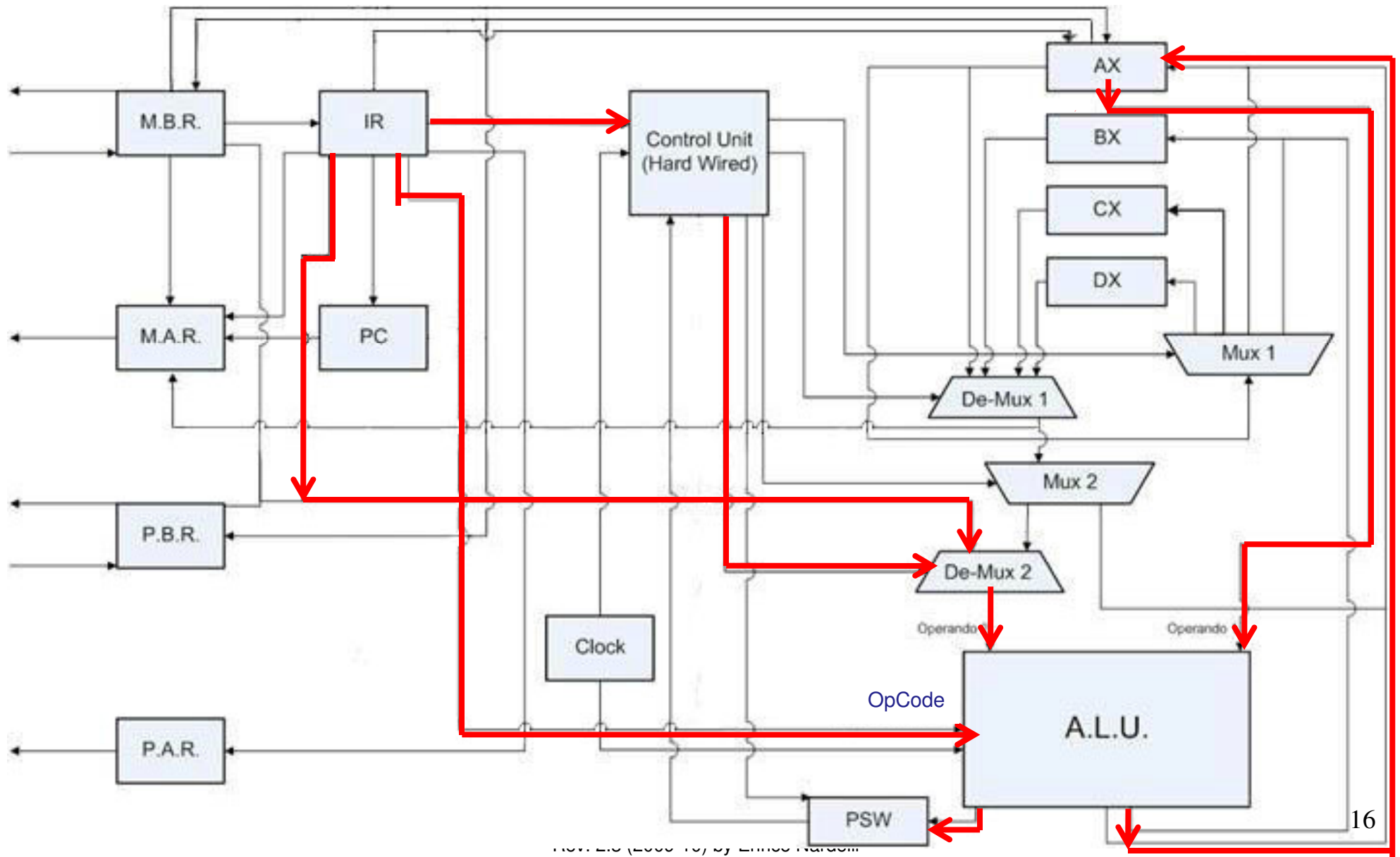
$b_{22}b_{21}$ = categoria

$b_{20} \dots b_{16}$ codificano la specifica operazione all'interno della categoria (per questa categoria si usano solo i bit $b_{20} b_{19}$)

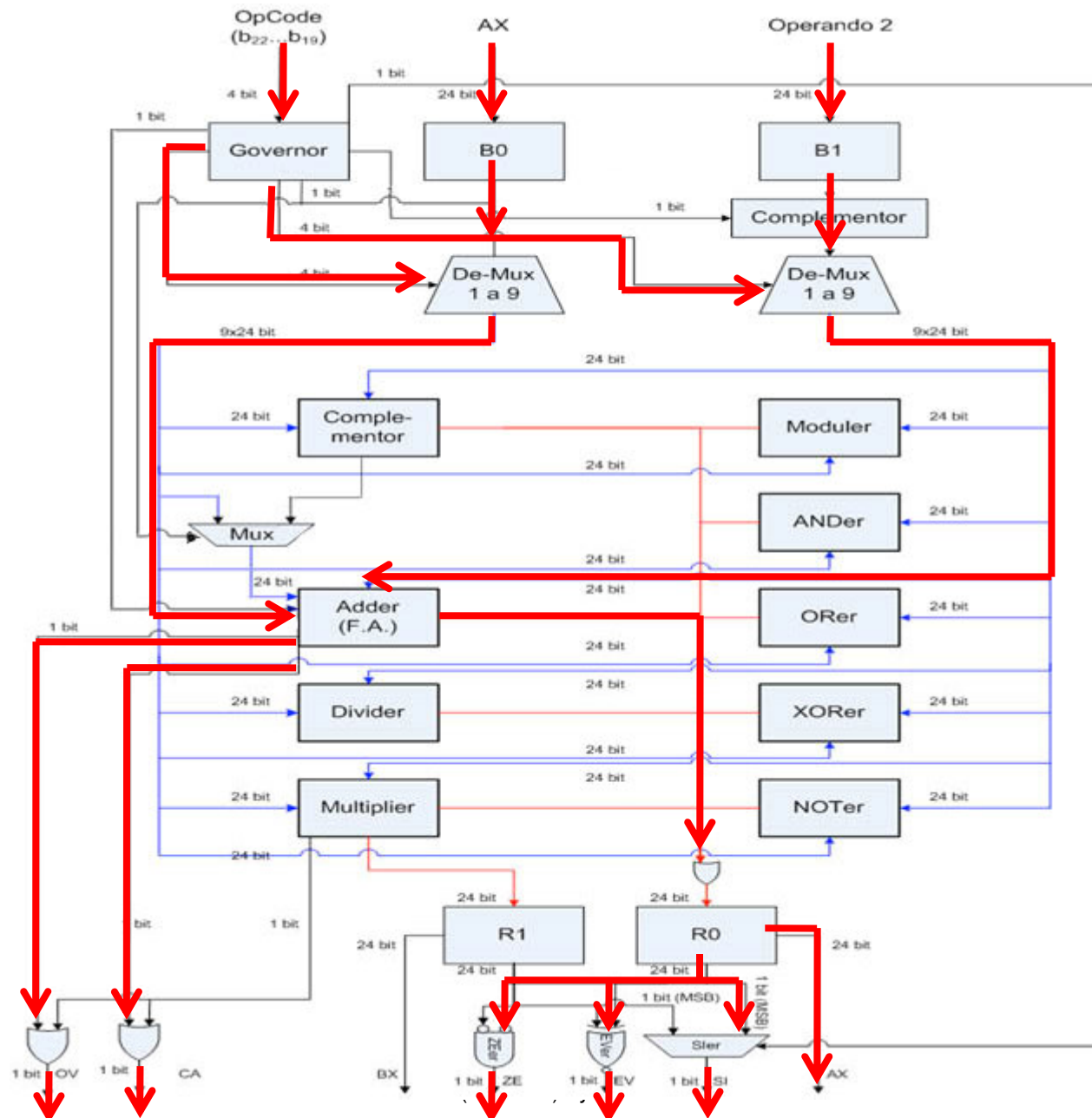
	b_{22}	b_{21}
Prodotti	1	1
Somme	1	0
Memorizzazione	0	1
Logiche ed altro	0	0

	b_{20}	b_{19}
INC	1	1
DEC	1	0
ADD	0	1
SUB	0	0

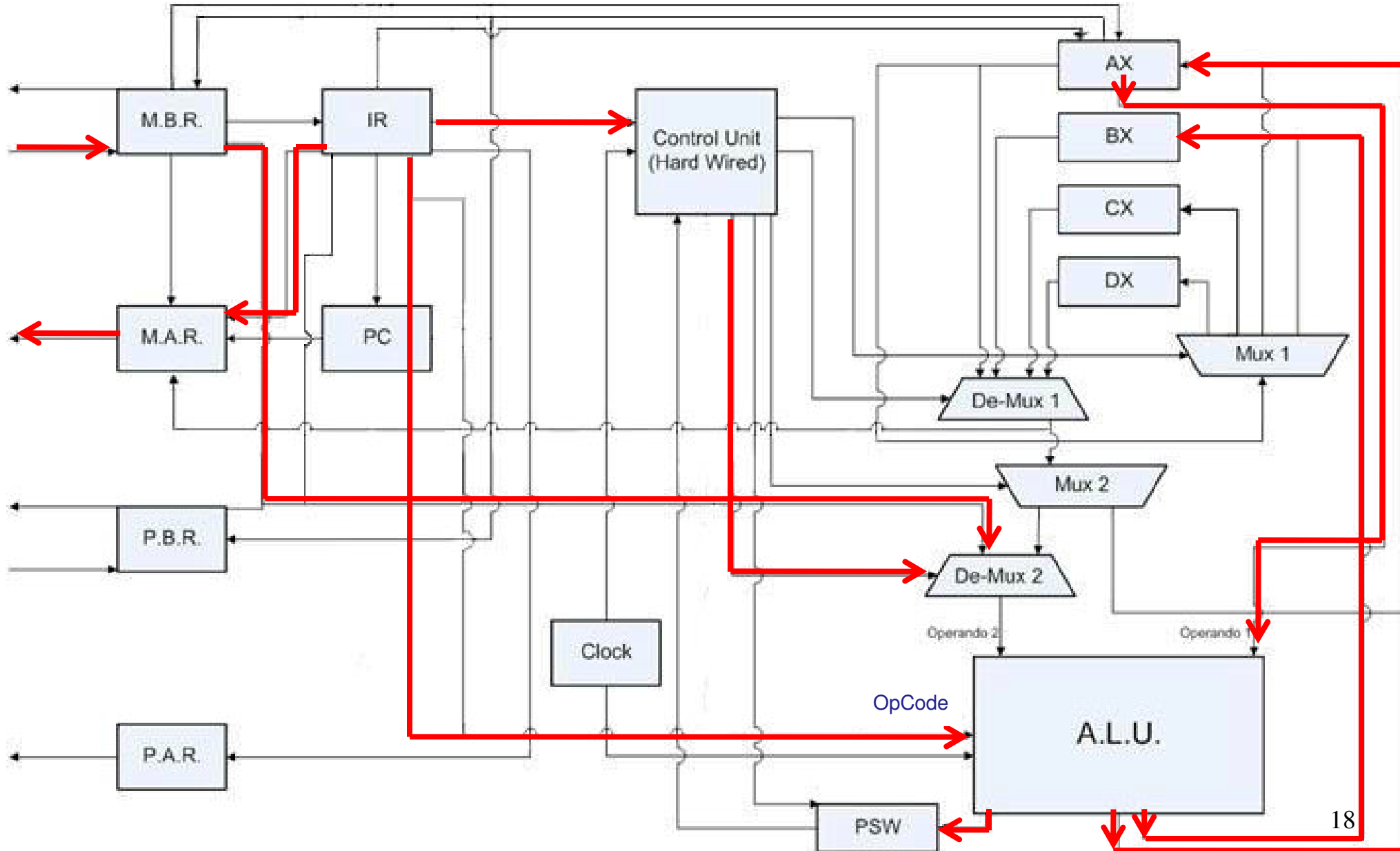
ADD 5



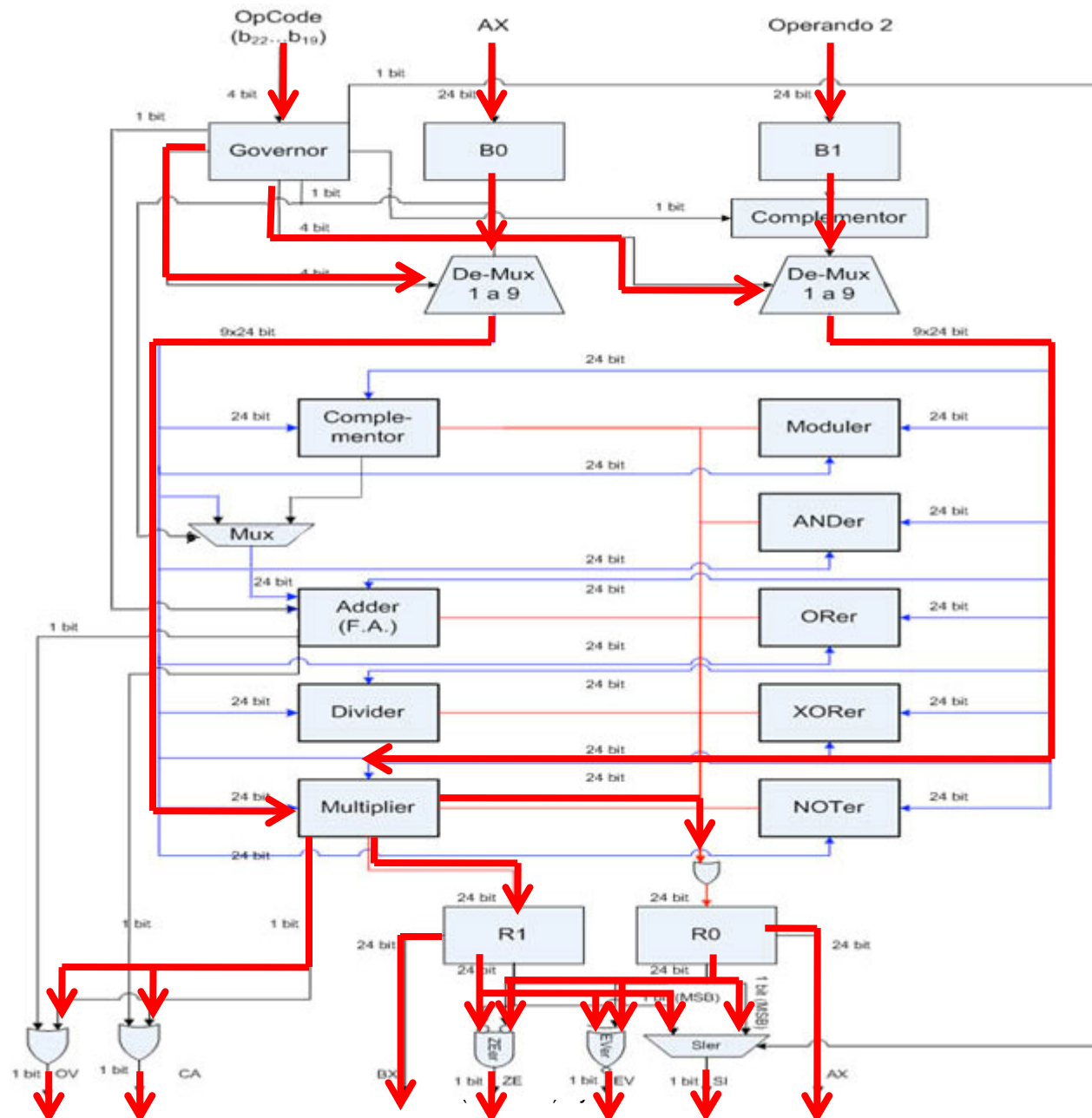
'ADD ...' : Cosa accade all'interno dell' ALU



MUL @5



'MUL @5' : Cosa accade all'interno dell' ALU



Un semplice esempio

Programma che calcola (senza istruzioni condizionate) il cubo del numero contenuto nella cella di memoria 5

Address	Content
0	
1	LOAD @5
2	MUL @5
3	MUL @5
4	STORE @6
5	3
6	

Cosa accade ...

- ▣ ... dopo che ho calcolato il cubo di 3?
- ▣ ... se calcolo il cubo di 2000 ?
- ▣ ... se calcolo il cubo di 9000 ?

Cosa accade in vCPU...

Address

Content

0

1

LOAD @5

2

MUL @5

3

MUL @5

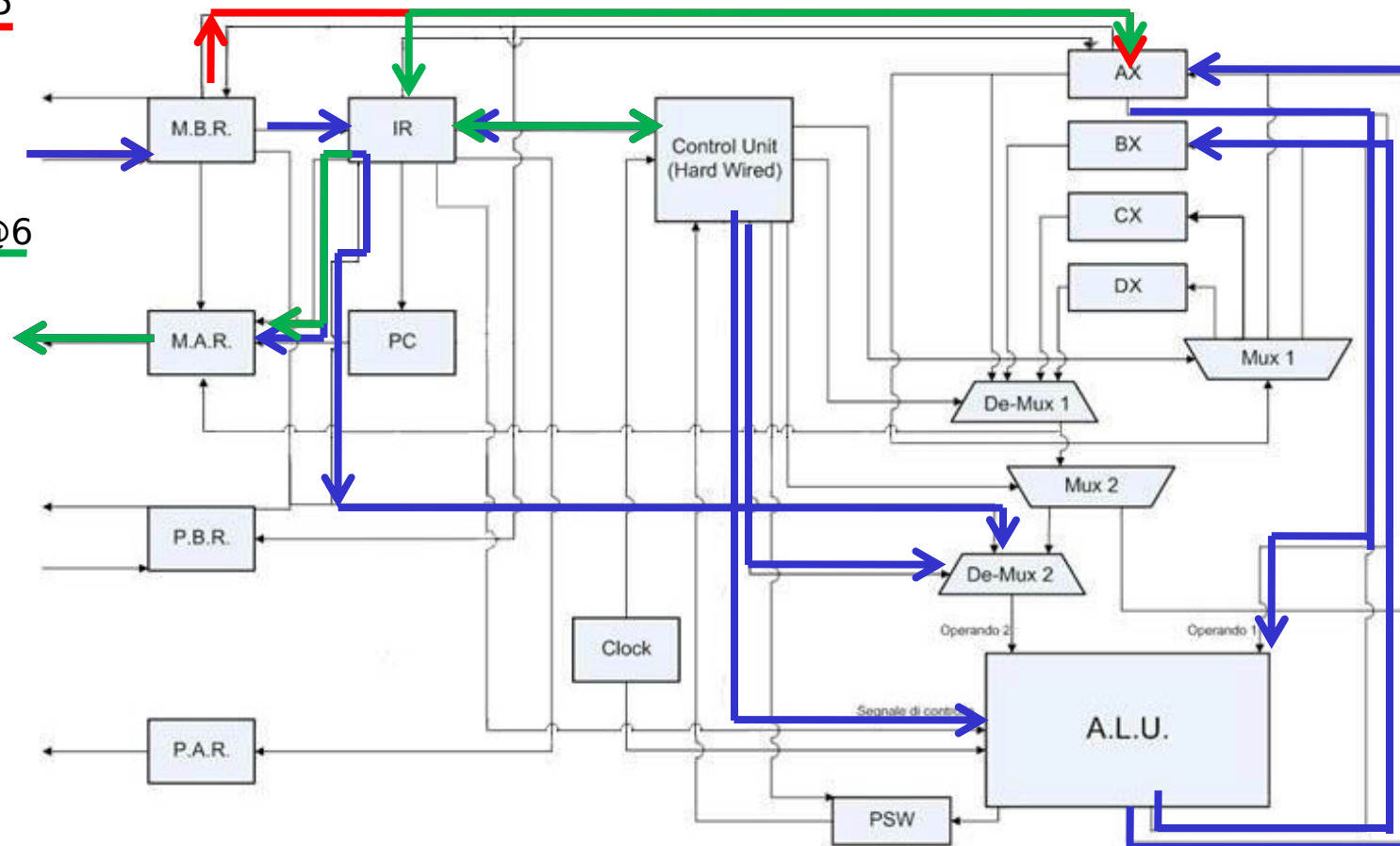
4

STORE @6

5

3

6



Ulteriori esempi...

Cosa accade a fronte della esecuzione delle diverse operazioni:

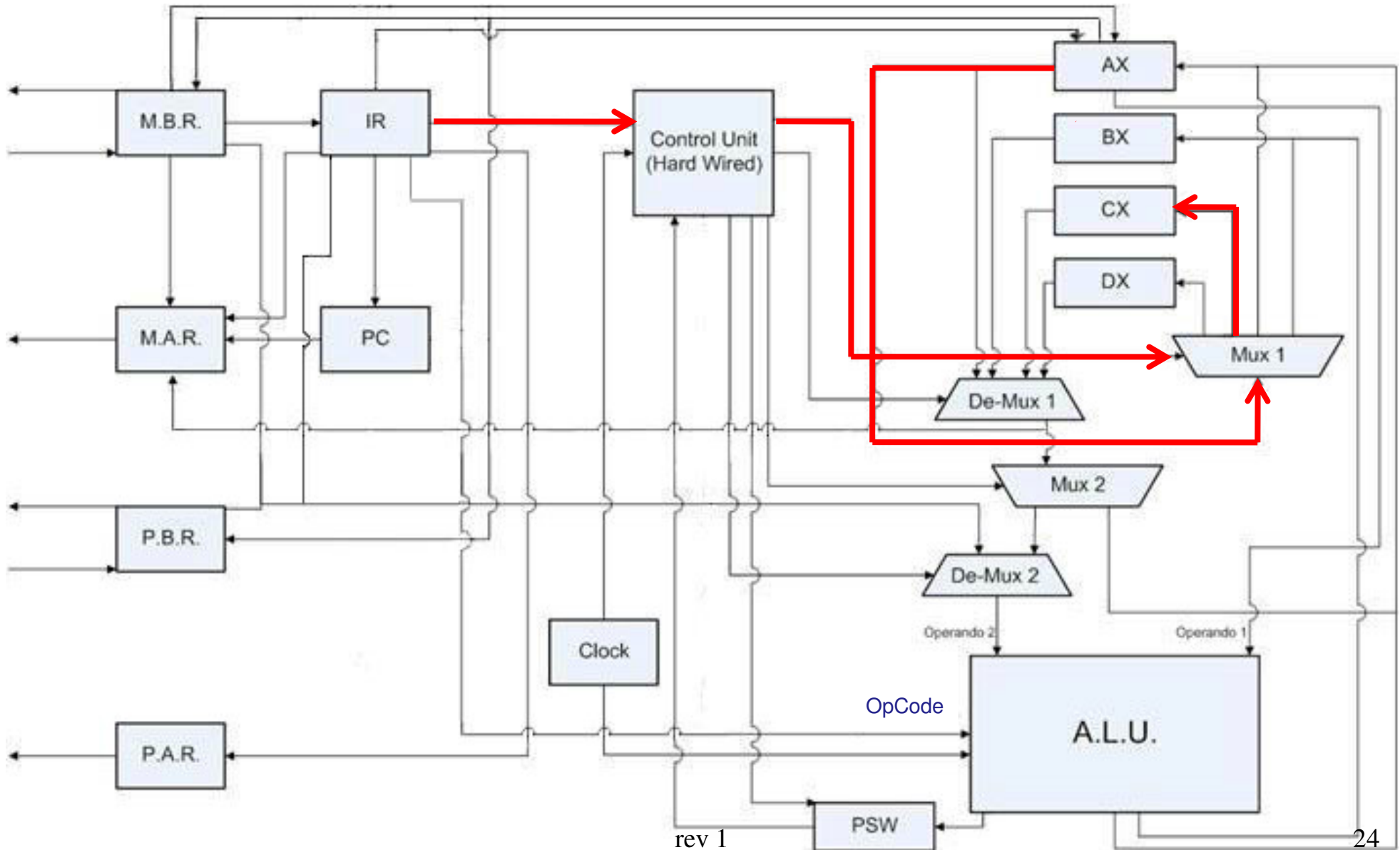
all'interno di vCPU

- Memorizzazione
- Sottrazione
- Salto condizionato e non

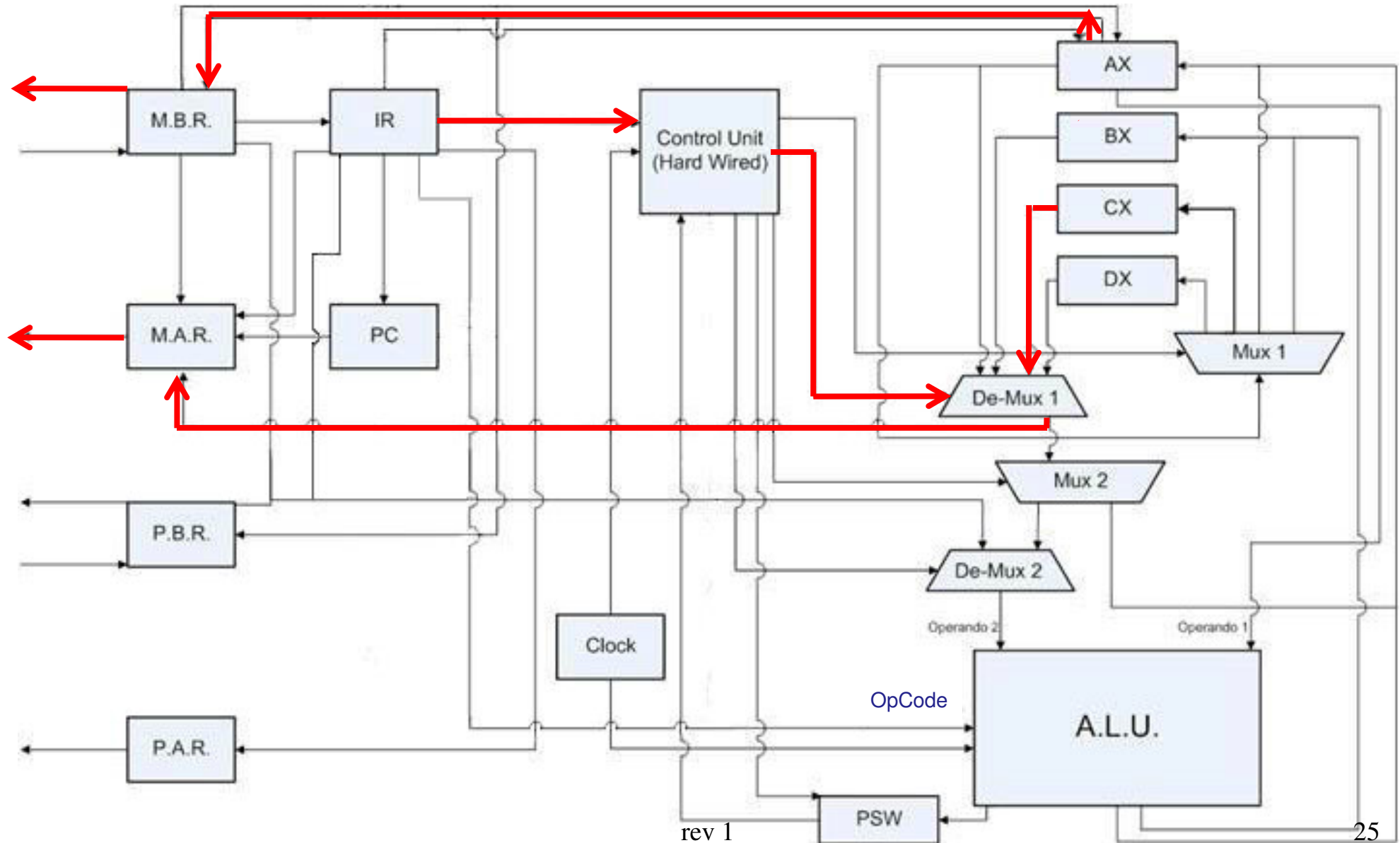
all'interno della ALU

- Incremento e decremento
- Negazione logica

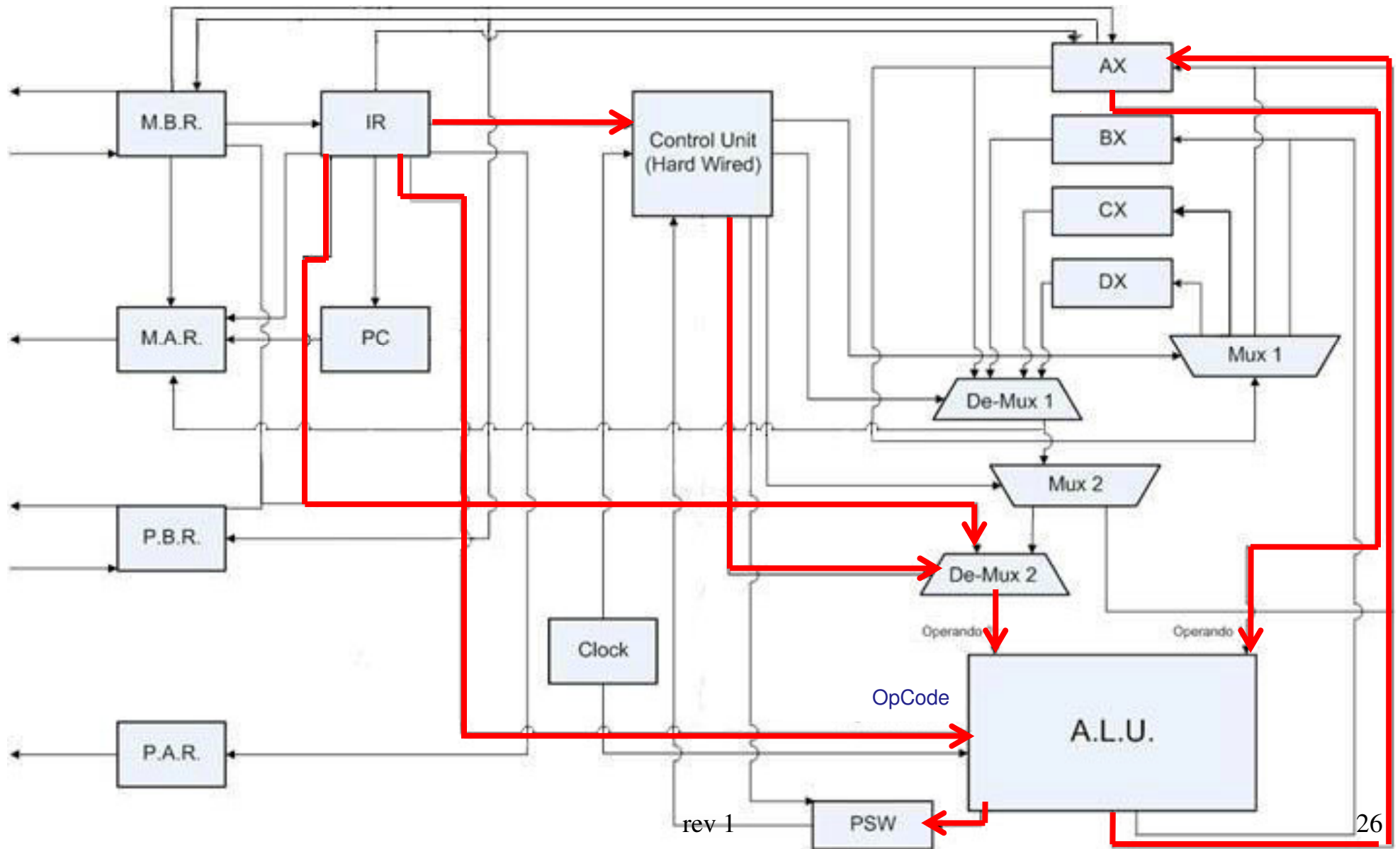
STORE CX



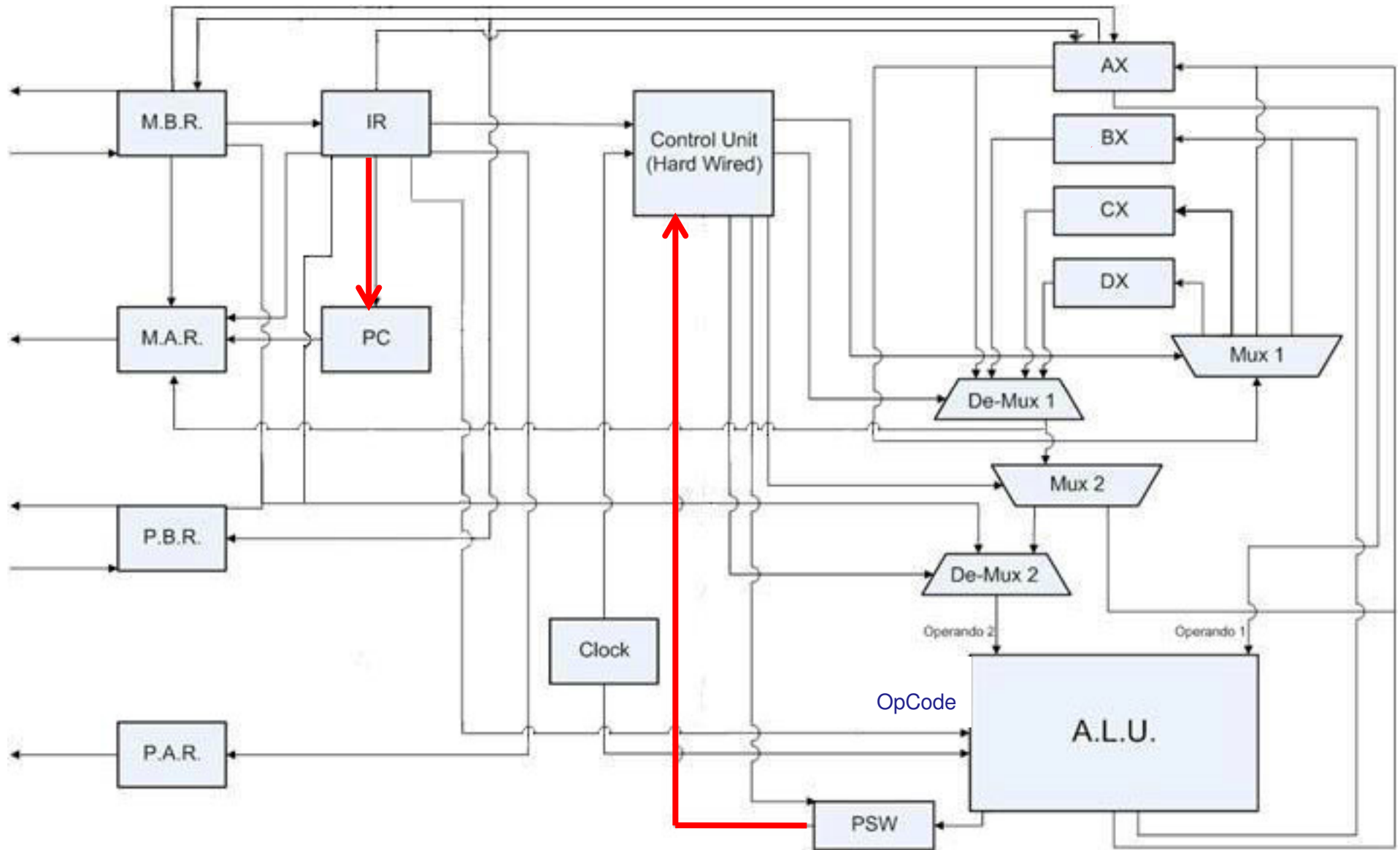
STORE @CX



SUB 3



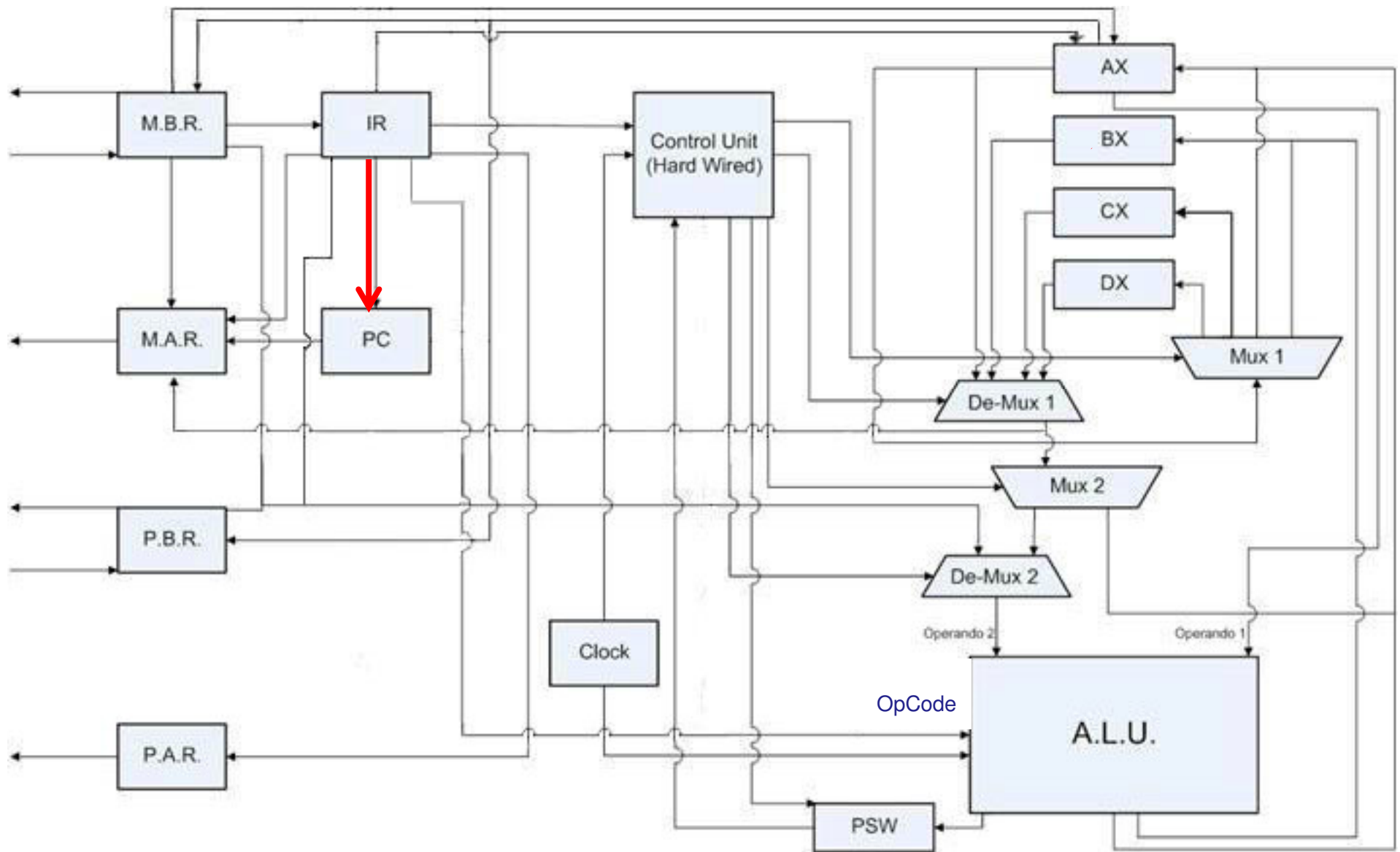
JZ 21



rev 1

Rev. 2.3 (2009-10) by Enrico Nardelli

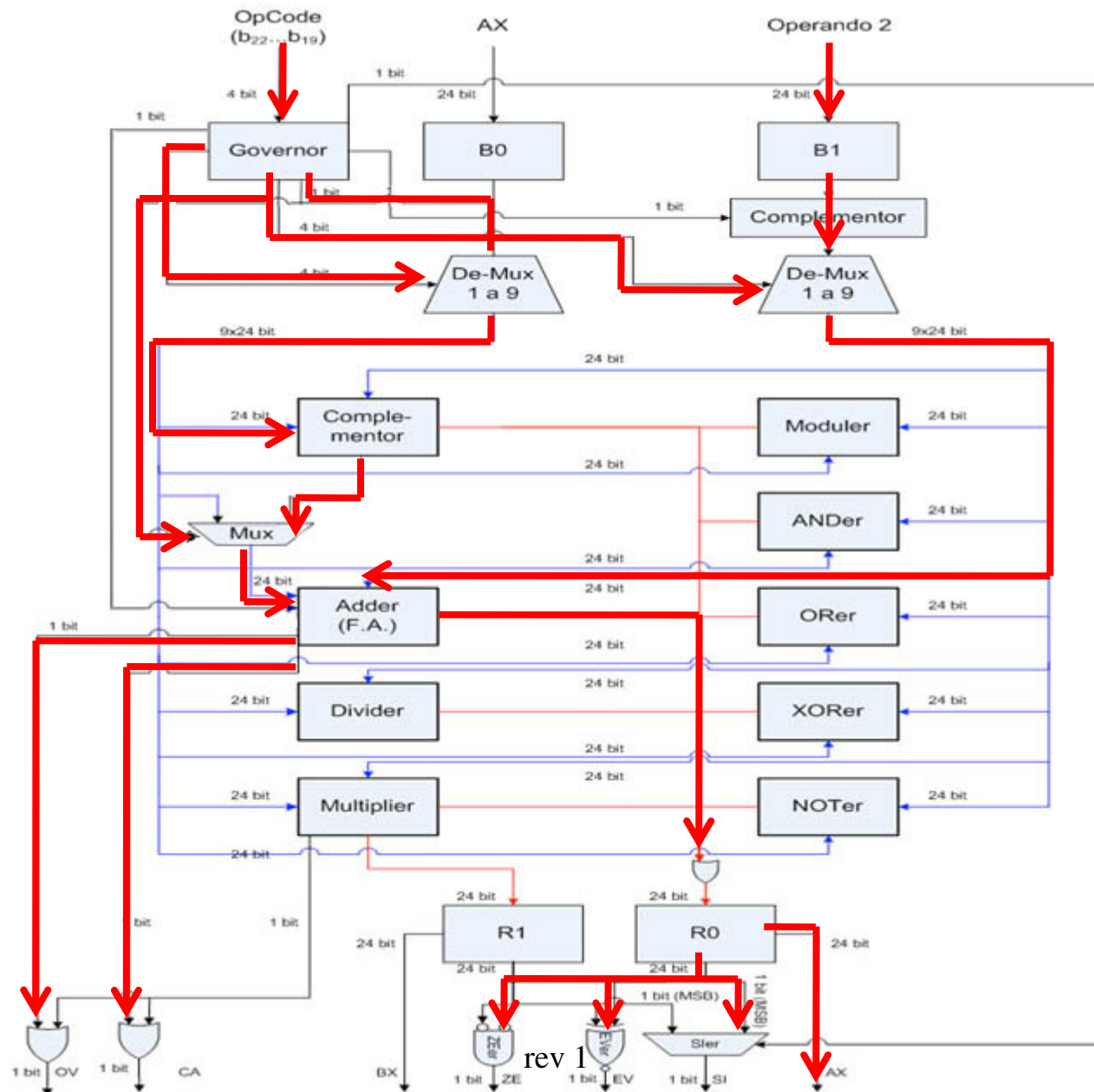
JMP 11



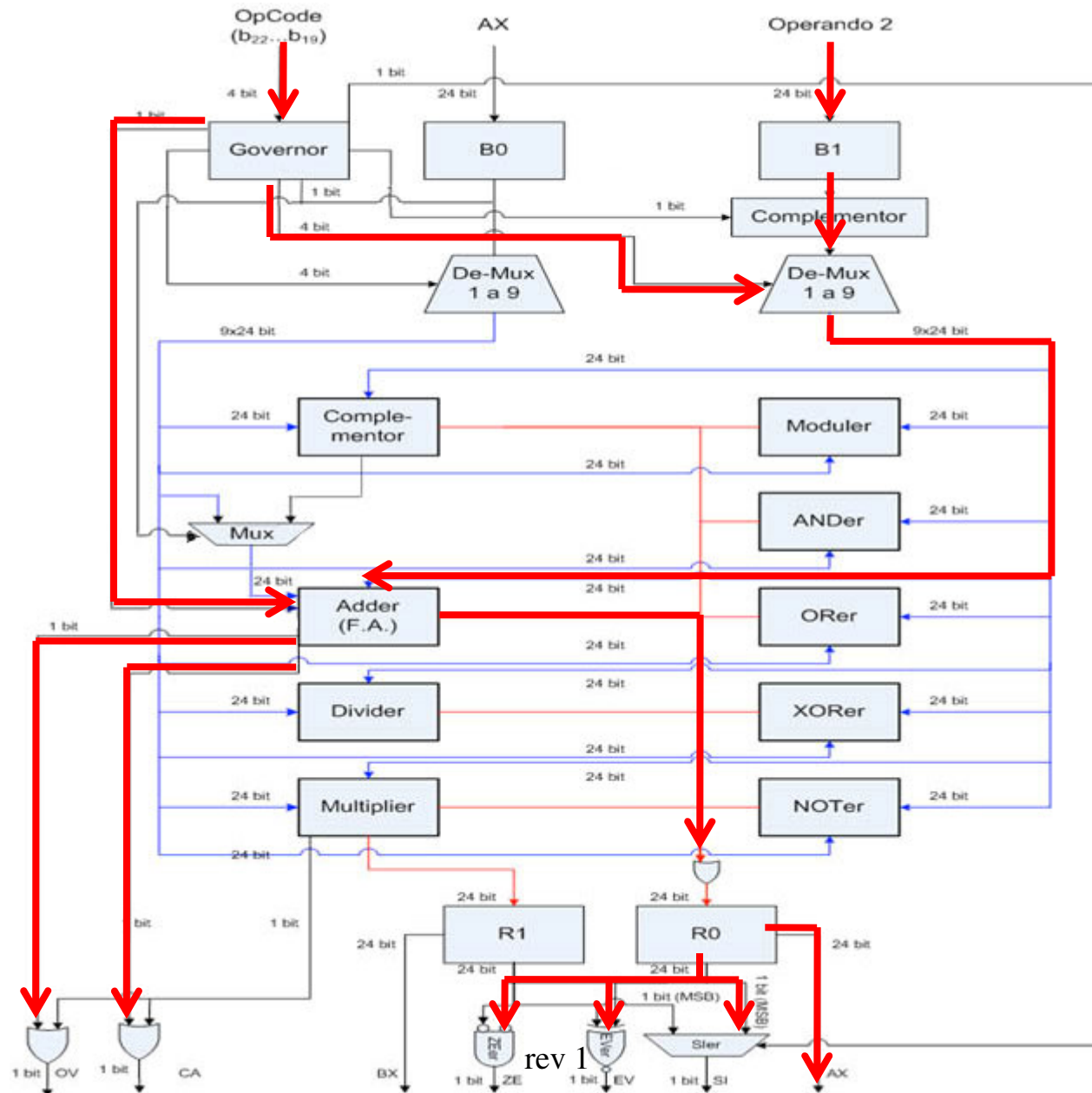
rev 1

Rev. 2.3 (2009-10) by Enrico Nardelli

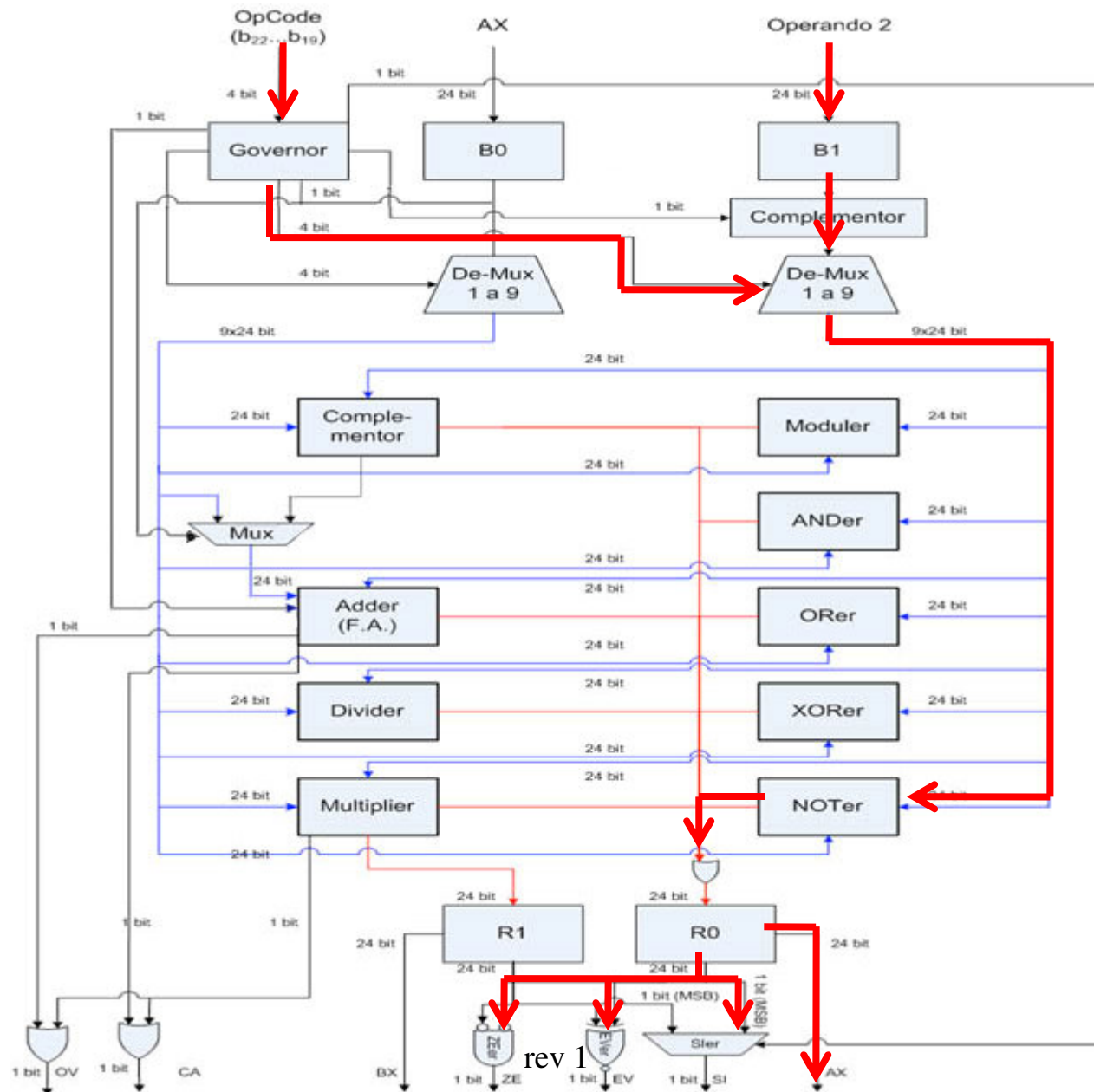
'DEC ...' : Cosa accade all'interno dell' ALU



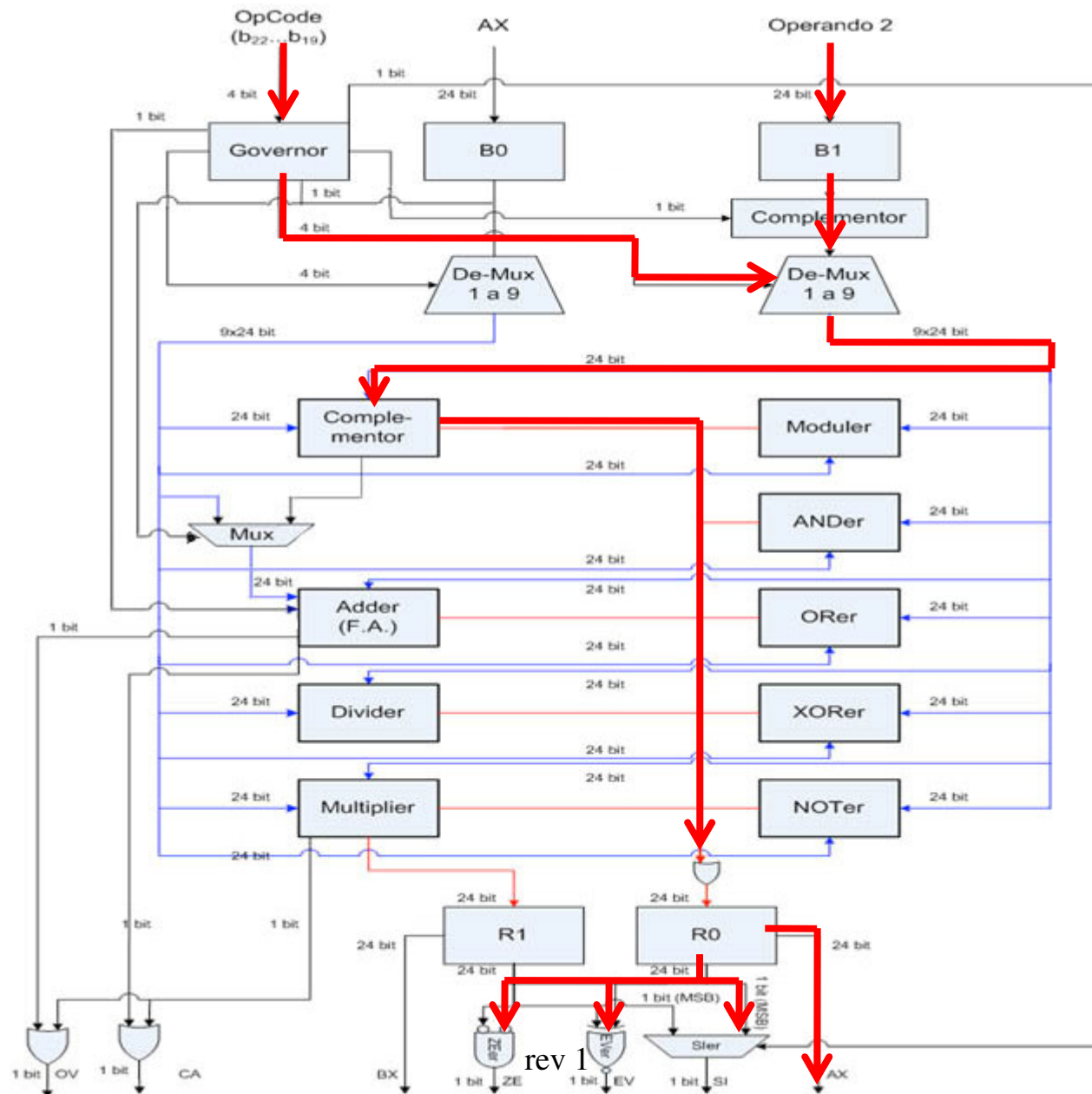
'INC ...' : Cosa accade all'interno dell' ALU



‘NOT ...’ : Cosa accade all’interno dell’ ALU



'NEG ...' : Cosa accade all'interno dell' ALU



Eniac

- Emulatore di una CPU virtuale progettata e realizzata dal Dott. Mauro Codella e Dott. Dario Dussoni nell'ambito delle loro Tesi di Laurea con il Prof. Enrico Nardelli.
- Attualmente il software è reperibile alla seguente URL :

<http://sourceforge.net/projects/eniac/>