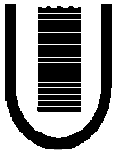
Virtual CPU – Eniac
parte 2

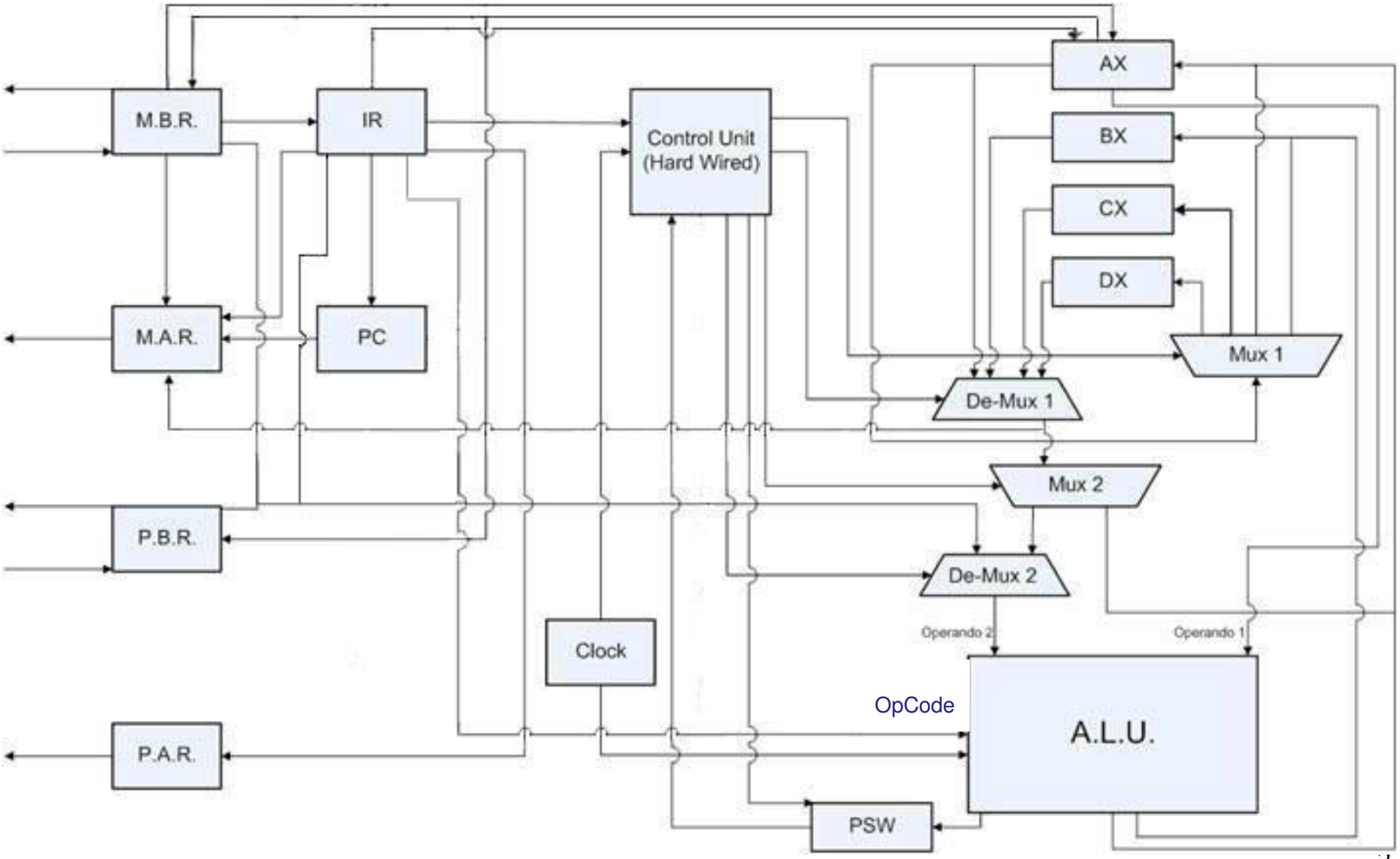


Università degli Studi di Roma “Tor Vergata”

Dr.ssa Veronica Marchetti

Rev. EN - 15gen08

Dove eravamo rimasti



I Flag

- Bit del registro PSW.

- Utilizzati per prendere decisioni in base all'esito di istruzioni precedenti, così da poter cambiare il flusso del programma
- Sono la rappresentazione di una parte dello stato finale a cui è arrivata l'ALU nell'effettuazione dell'ultima operazione

I Flag

- Semplici - il meccanismo che ne determina il valore dipende solo dallo stato finale dell'ALU, NON da quale operazione è stata effettuata

- Complessi - vengono configurati anche in base a criteri legati alle operazioni
  - in particolare le operazioni coinvolte sono quelle di addizione e sottrazione.
  - (Vedremo successivamente la loro interpretazione a livello aritmetico)

I Flag

Flag semplici

- SI = 1 se l'ultima operazione ha generato un risultato in cui il bit più significativo = 1,

  0 altrimenti;

- ZE =1 se il risultato dell'ultima operazione è zero,

  0 altrimenti;

- EV =1 se la somma dei bit a 1 del risultato dell'ultima operazione è pari (*even*) – Attenzione! non implica che il risultato sia pari!,

  0 altrimenti.

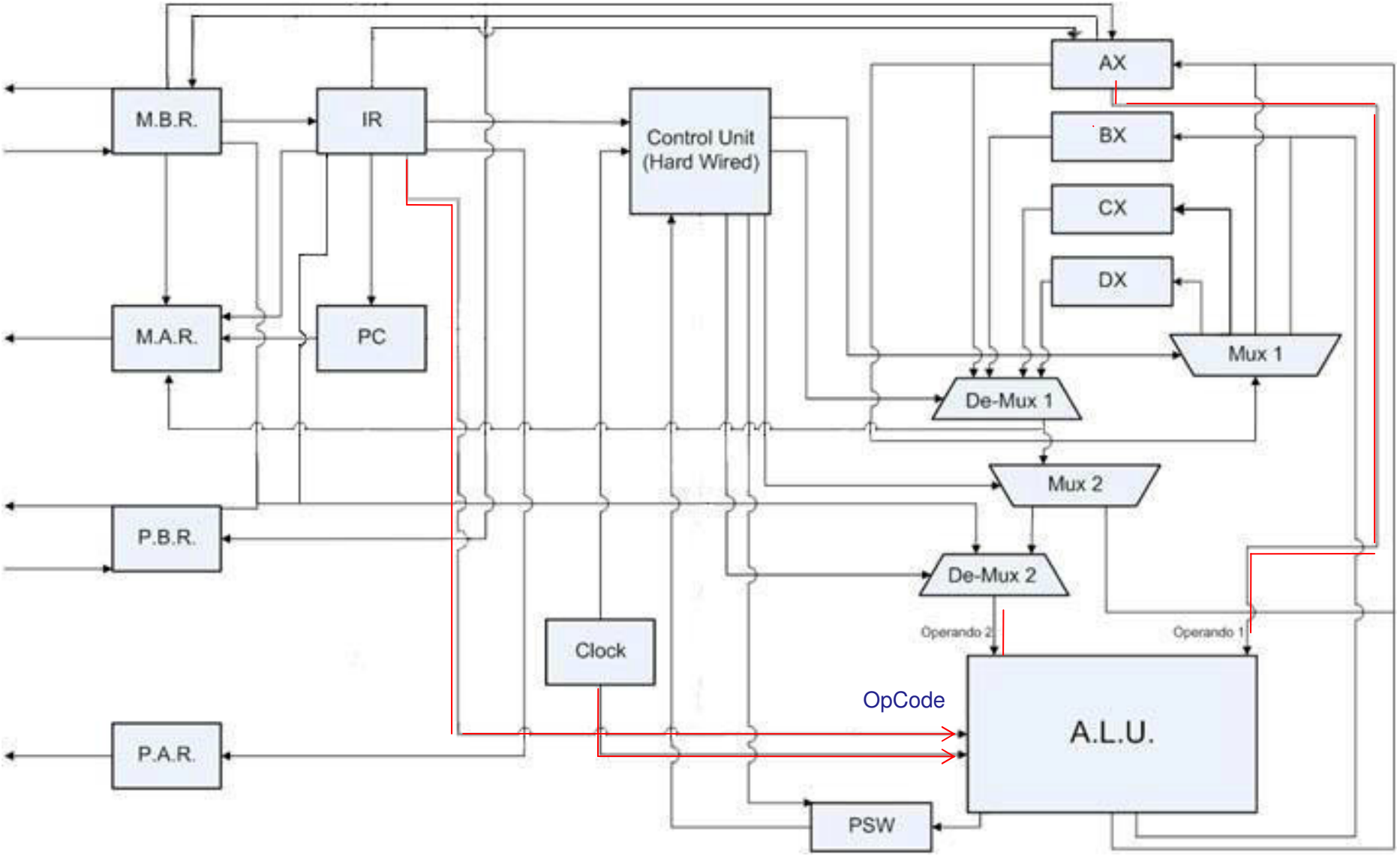
La ALU e le sue funzionalità

- Operazioni possibili:

  somma, sottrazione, divisione, moltiplicazione, negazione, modulo, and, or, or esclusivo (xor) e not.

- Ha quattro ingressi
  - Due di dati
  - Uno di controllo
  - Uno per il clock

Ingressi


M.B.R.
IR
Control Unit
(Hard Wired)
AX
BX
CX
DX
M.A.R.
PC
Mux 1
De-Mux 1
Mux 2
P.B.R.
De-Mux 2
Operando 2
Operando 1
Clock
OpCode
A.L.U.
P.A.R.
PSW

La ALU e le sue funzionalità

- Scelta progettuale
  - Il secondo operando e il segnale di controllo sono specificati nell'istruzione
  - Il primo operando è sempre AX.
  - L'uscita va sempre in AX ma, nel caso specifico della moltiplicazione, è distribuita in BX e AX.
  - Insieme al risultato escono anche i valori che configurano i flag nel registro PSW.
- il segnale di controllo specifica l'operazione da eseguire
- I due operandi in ingresso hanno tanti bit quanti una parola di memoria,
- Il risultato può essere della dimensione o di una parola o di due parole di memoria

Comprendere la struttura della ALU di vCPU

- Descrizione incrementale (comprensione graduale)

Gli elementi :

- **Adder**,
  - implementato per mezzo di un Full Adder, in grado di effettuare delle addizioni
  - Gli operandi dell'operazione vengono inizialmente memorizzati nei registri interni alla ALU, B0 e B1 (memorizzano rispettivamente il valore di AX e l'operando specificato nell'istruzione)
  - La presenza di un circuito di complementazione a 2, indicato nella figura come **Complementor**, permette inoltre di eseguire le sottrazioni.

Comprendere la struttura della ALU di vCPU

- Il modulo **Governor :**
  - decodifica l'opcode in input
  - attiva la circuiteria per effettuare l'operazione richiesta.

    Infatti quando l'opcode identifica una sottrazione, il Governor attiva il Complementor che trasforma il secondo operando nel suo complemento a 2 e rende così possibile effettuare la sottrazione dal primo operando mediante l'Adder.

    (3) - (5)

    (3) + (compl a 2 di 5)

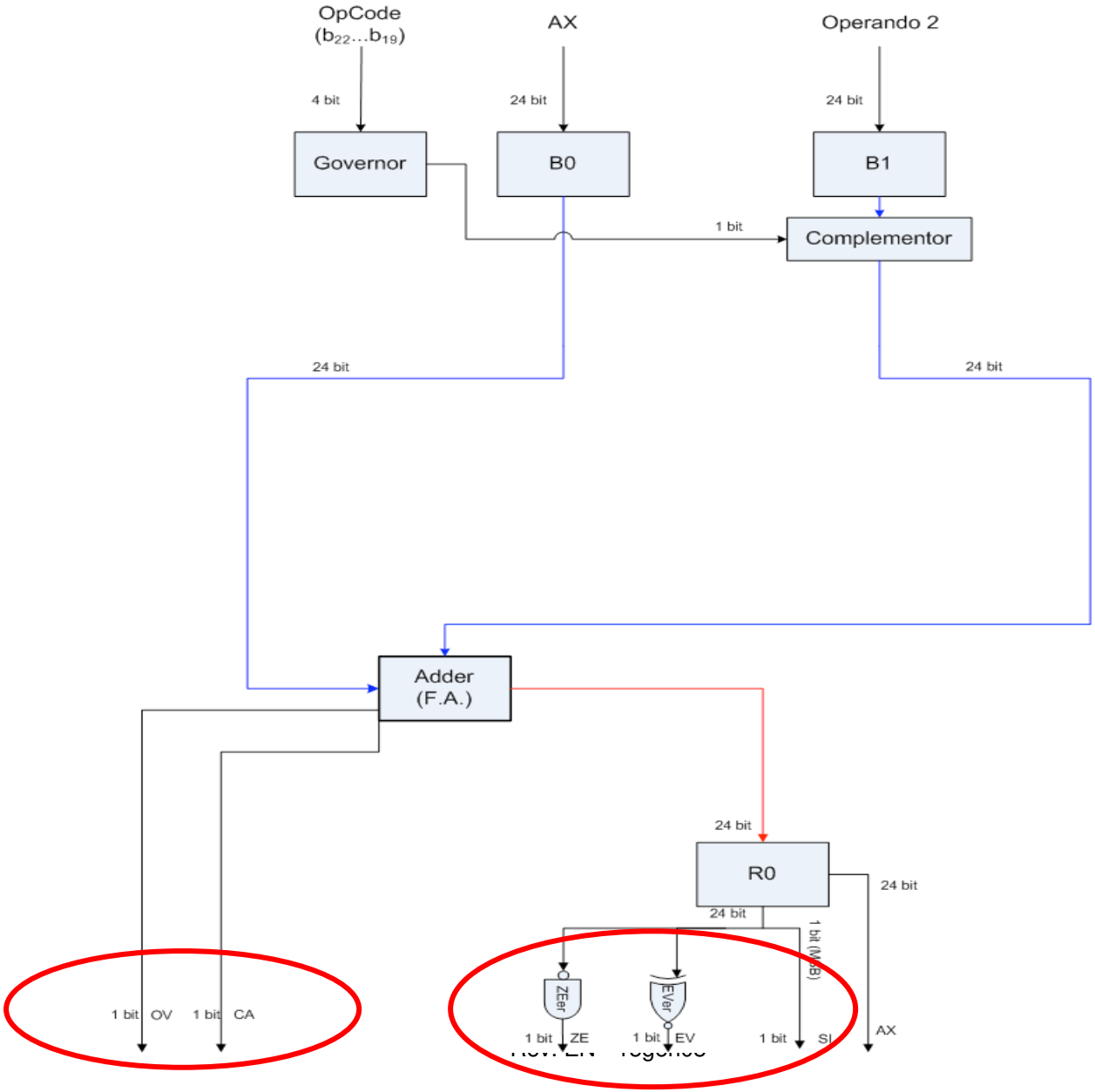
    ovvero $(0011)_2 + (1011)_2$

Comprendere la struttura della ALU di vCPU

- Risultato viene memorizzato nel registro R0 da dove vengono prelevati i valori necessari per il calcolo dei flag *semplici*
- Flag semplici dell' Adder
  - SI coincide con il bit più significativo in R0 (segno)
  - ZE con l'AND di tutti i bit in R0 negati
  - EV con lo XOR negato di tutti i bit in R0.
- Flag complessi dell'Adder
  - CA = 1 se il bit di riporto dell'Adder vale 1 nell'ultima operazione,

    0 altrimenti.
  - OV = 1 se l'ultima operazione ha generato un risultato il cui bit più significativo differisce da entrambi i bit più significativi degli operandi, 0 altrimenti;
- Il registro R0 e' inoltre connesso ad AX.

Una semplice ALU in grado di effettuare somme e sottrazioni



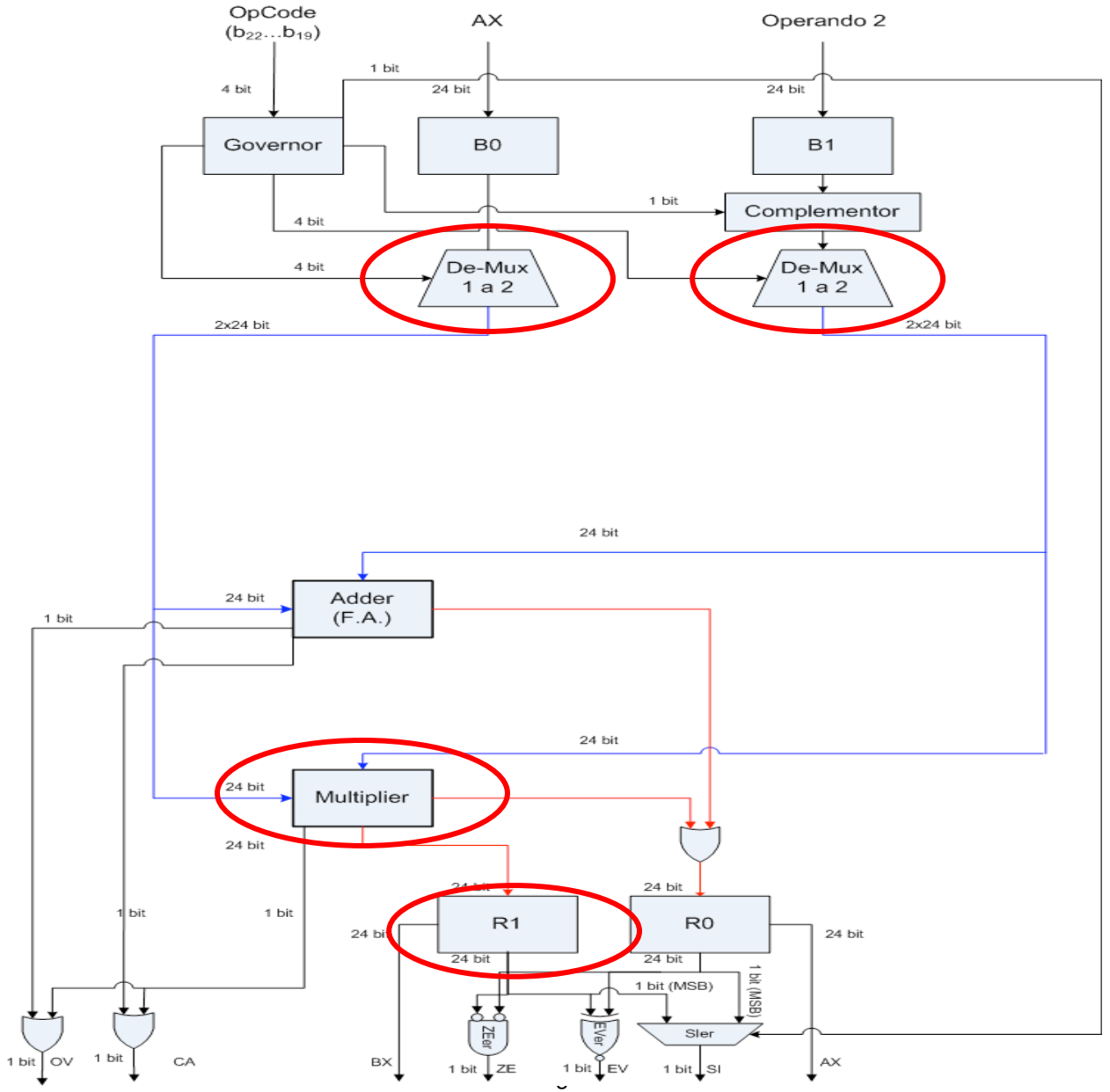
ALU somme sottrazioni e moltiplicazioni

Successivo livello di complessità: aggiunta di un modulo in grado di effettuare le operazioni di moltiplicazione: il **Multiplier**

- introduzione di due de-multiplexer, guidati dal Governor, in grado di indirizzare gli operandi verso il modulo d'interesse (Adder o Multiplier).
- La natura dell'operazione di moltiplicazione comporta l'introduzione di un secondo registro per la memorizzazione del risultato: il registro R1.

  il Multiplier memorizza la parte meno significativa del risultato in R0 e la piu' significativa in R1

Una semplice ALU in grado di effettuare somme e sottrazioni e moltiplicazioni

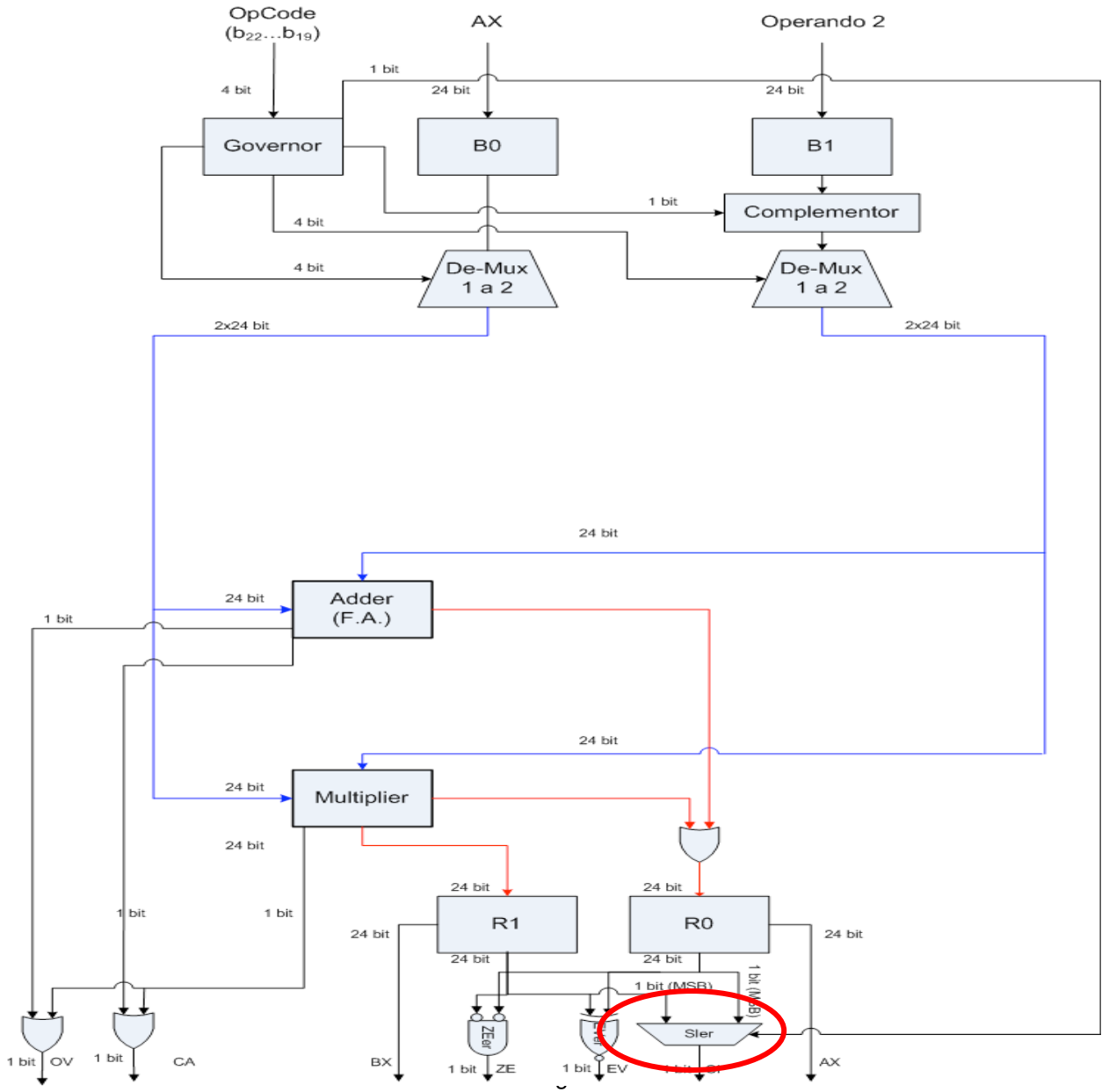


ALU somme sottrazioni e moltiplicazioni

- Conseguentemente all'introduzione del nuovo registro, la logica per il calcolo dei flag semplici cambia.
  - Per ZE ed EV vengono considerati anche i bit in R1
  - Per SI viene utilizzato un de-multiplexer, pilotato dal Governor, in grado di selezionare il bit più significativo che definirà valore del flag.

    In particolare se l'operazione da effettuare è una moltiplicazione, verrà utilizzato il bit più significativo in R1, altrimenti quello in R0.

Una semplice ALU in grado di effettuare somme e sottrazioni e moltiplicazioni



ALU somme sottrazioni e moltiplicazioni

- Come l'Adder anche il Multiplier genera dei flag complessi
  - OV: viene assegnato a 0 quando i bit in R1 e il bit più significativo in R0 sono tutti uguali tra di loro,

    1 altrimenti.
  - CA: viene assegnato secondo lo stesso criterio usato per OV.

- Per poter avere solo un bit in uscita sia per il flag OV che per CA, sono state introdotte due porte OR in grado di riunire i segnali uscenti dai moduli Adder e Multiplier in uno solo.

ALU somme sottrazioni e moltiplicazioni

- Tenendo presente che :
  - le porte OR hanno output 1 nel caso in cui almeno uno degli ingressi valga 1
  - grazie ai de-multiplexer collegati con gli operandi i moduli Adder e Multiplier non possono essere attivi contemporaneamente

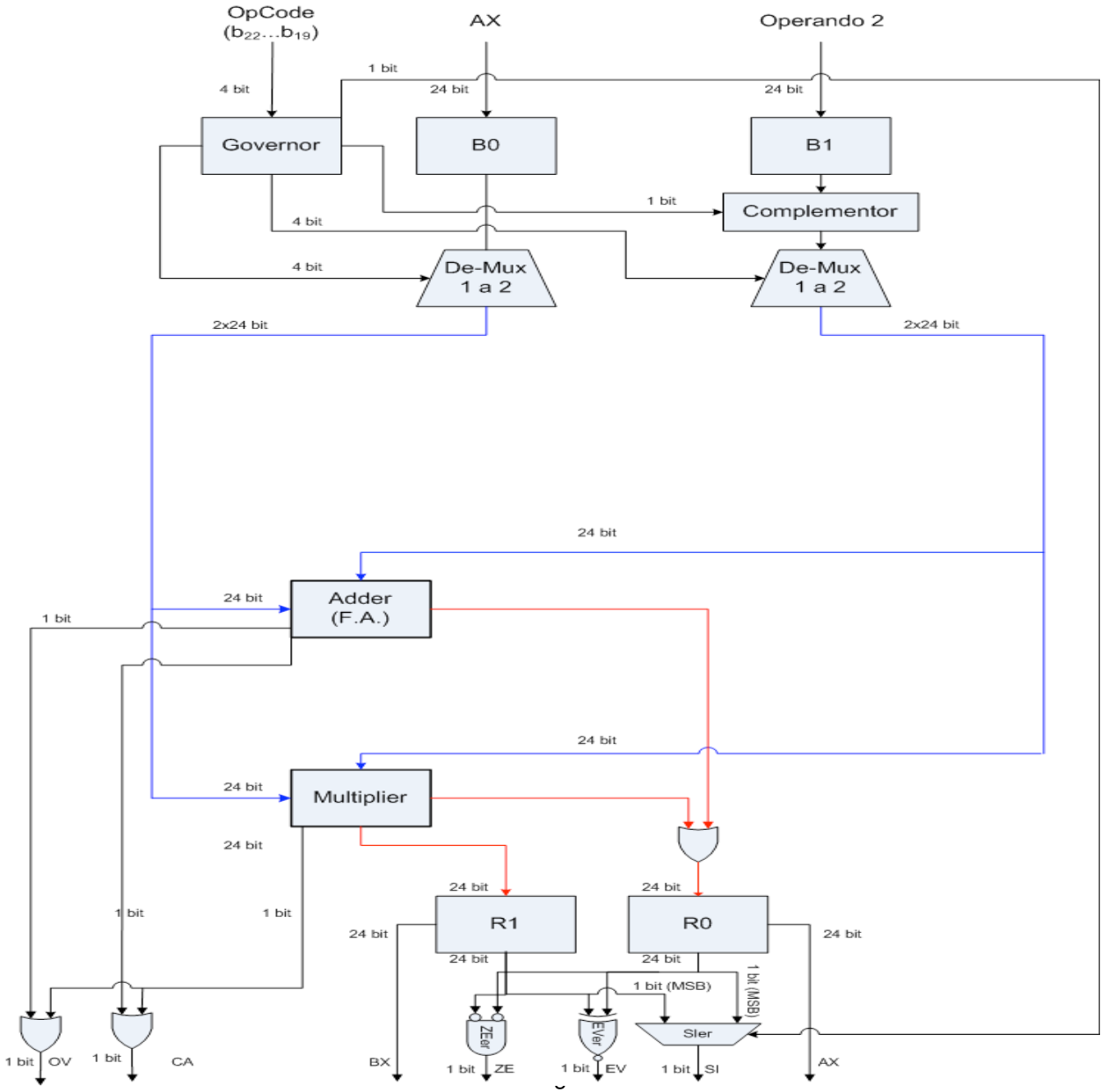
i flag OV e CA in uscita dalla ALU hanno sempre il valore generato dal modulo attivato.

- Analogamente ai flag complessi, anche il registro R0 riceve in ingresso l'OR di tutte le uscite dei moduli.

ALU somme sottrazioni e moltiplicazioni

- Come visto prima, grazie ai de-multiplexer, può essere attivo solo un modulo alla volta e il segnale in uscita dal modulo attivato, in OR con tutte le uscite a 0 degli altri moduli, coincide proprio con il risultato dell'operazione richiesta.
- Conseguenza di ciò è che non si ha la necessità di avere dei segnali di abilitazione per ogni modulo.

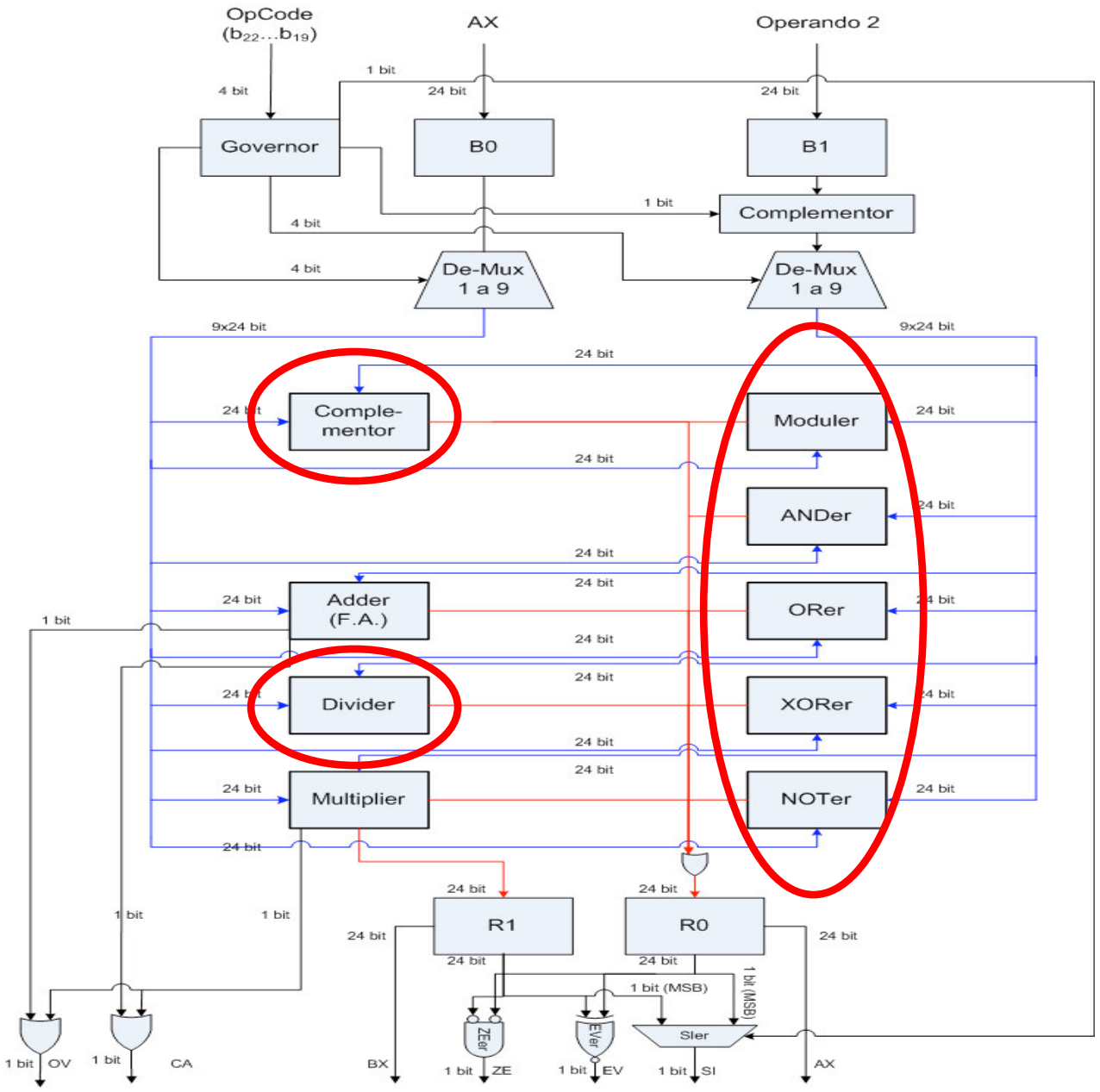
Una semplice ALU in grado di effettuare somme e sottrazioni e moltiplicazioni



La ALU di vCPU

- Completiamo ora la descrizione della struttura della ALU di vCPU, con la presentazione degli altri moduli necessari per implementare il resto delle operazioni
  - Divisione
  - Operazioni booleane

La ALU di vCPU



La ALU di vCPU

Elenco completo dei moduli presenti :

- **Adder** effettua la somma di due interi relativi.

  R0 = Operando1 + Operando2

- **Complementor** restituisce l'opposto di un intero relativo.

  R0 = –Operando1

La ALU di vCPU

- **Multiplier** effettua il prodotto di due interi relativi.

  R1,R0 = Operando1 Operando2

- **Divider** effettua la divisione tra due interi relativi.

  R0 = Operando1/Operando2

- **Moduler** effettua il modulo della divisione tra due interi relativi.

  R0 = Operando1 mod Operando2

- **ANDer** effettua l'AND logico bit a bit tra due interi relativi.

  R0 = Operando1 AND Operando2

La ALU di vCPU

- **ORer** effettua l'OR logico bit a bit tra due interi relativi.

  R0 = Operando1 OR Operando2

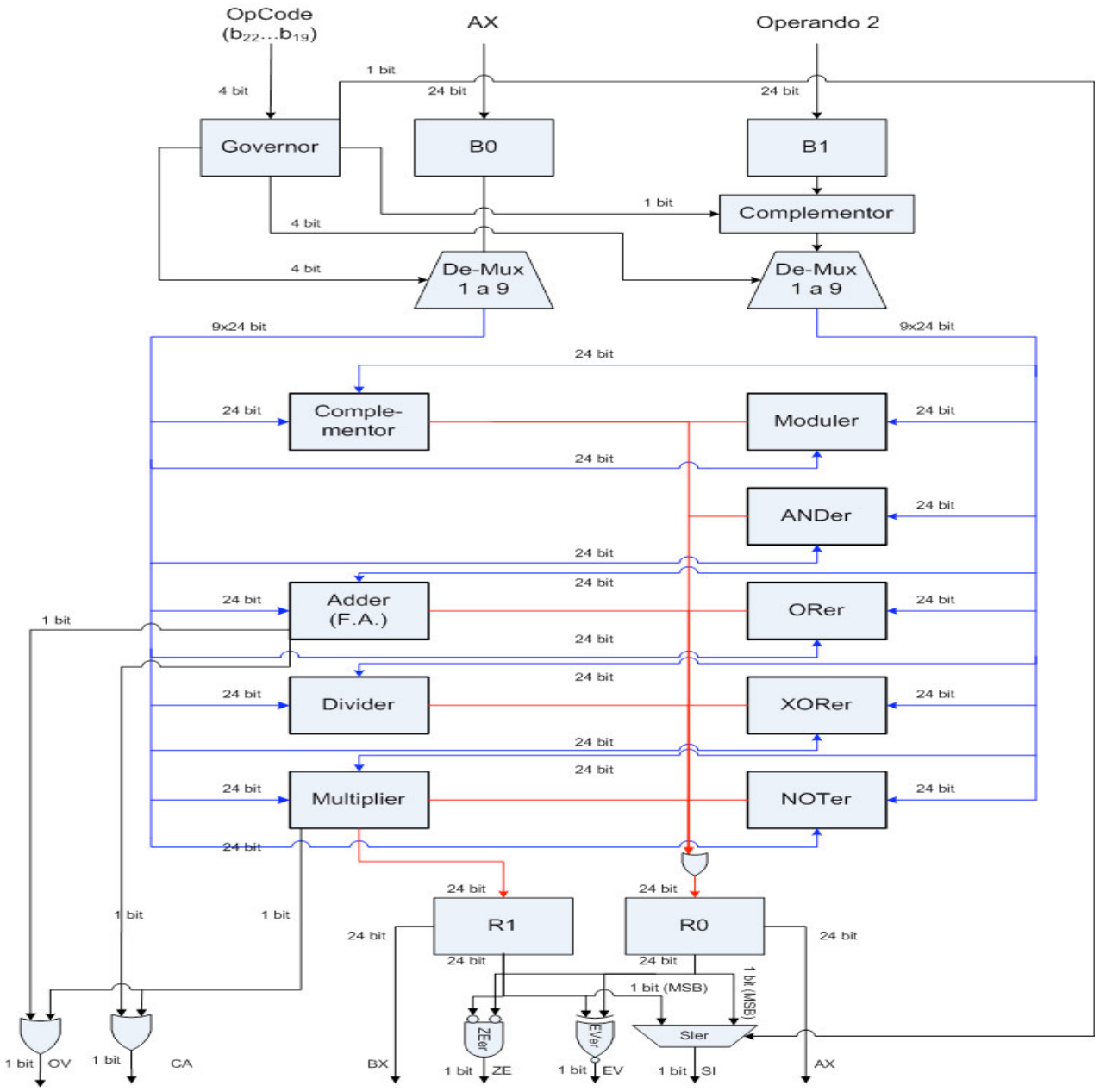
- **XORer** effettua l'OR esclusivo bit a bit tra due interi relativi.

  R0 = Operando1 XOR Operando2

- **NOTer** effettua il NOT logico bit a bit di un intero relativo.

R0 = Operando1

La ALU di vCPU



La ALU di vCPU

- La struttura del Multiplier e del Divider, e dei relativi componenti interni, non verrà esaminata in maggior dettaglio.

- Per ulteriori informazioni relative a tale struttura si rimanda alla tesi del Dott. Mauro Codella.

Le istruzioni di vCPU

- Torniamo ora ad analizzare in dettaglio le istruzioni di vCPU vedendo prima che valori possono essere rappresentati
  - Questa descrizione è ovviamente dipendente dalla struttura che è stata definita per la ALU

Rappresentazione degli interi

- vCPU è dotata di istruzioni in grado di elaborare solo interi relativi.
- codifica adottata è quella del complemento a due.

vantaggi del complemento a due rispetto ad altre notazioni

- Ha una sola rappresentazione dello zero
- Permette di effettuare le operazioni aritmetiche utilizzando la stessa circuiteria in grado di compiere operazioni aritmetiche su interi senza segno

Complemento a due (*)

- E' il metodo più diffuso per la rappresentazione dei numeri relativi in informatica
- E' inoltre una operazione di negazione
- Ragione della diffusione: i circuiti di addizione e sottrazione non devono esaminare il segno di un numero rappresentato con questo sistema per determinare quale operazione sia necessaria.
- Il bit iniziale rappresenta il segno

  (se 1 il numero e' negativo – per ottenere il modulo complemento e sommo 1)
- Il complemento a due di un numero negativo restituisce il numero positivo pari al valore assoluto

Complemento a due (*)

Esempio: Rappresentiamo il numero -5 con 8 bit in complemento a due.

- Parto dalla rappresentazione in binario del numero 5.

  0000 0101 (5)

  La prima cifra è 0, il numero è positivo

- Inverto i bit (ottengo il complemento a uno)

  1111 1010

- Aggiungendo uno ottengo il complemento a due

  1111 1011 (-5)

  Il primo bit (è 1) evidenzia che il numero è negativo

- Il risultato è la rappresentazione in complemento a due di un numero binario.

Rappresentazione degli interi

- Un numero intero con segno è rappresentato da

$$b_n\ b_n{-}1\ \ldots\ b_1\ b_0$$

dove $b_i \in \{0, 1\}$ tali che

$$-b_n 2^n + b_{n-1} 2^{n-1} + \ldots + b_1 2^1 + b_0 2^0 = X$$

Si rappresentano quindi tutti i numeri relativi che vanno da -2^n a 2^n-1 inclusi.

in cui il bit significativo ha un peso negativo.

Rappresentazione degli interi

La rappresentazione con segno richiede di definire a priori la grandezza dell' intervallo di rappresentazione, cioè il numero di bit dedicati alla rappresentazione del numero stesso.

Un numero rappresentato in complemento a due usando N bit si può rappresentare anche utilizzando N+K bit aggiungendo prima del MSB per K volte il valore del MSB stesso.

$(-3)_{10} = (101)_2$ con 3 bit

$(-3)_{10} = (11101)_2$ con 5 bit

Il formato delle istruzioni di vCPU

Presentiamo in dettaglio il formato delle istruzioni e i criteri utili per assegnare gli opcode alle varie operazioni

Definiamo ora formalmente i termini argomento e operando

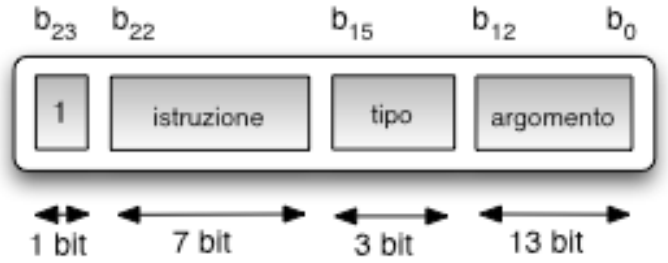
- Argomento: e' il valore dei bit $b_{12}...b_0$ presenti nell'istruzione;
- Operando: e' il valore effettivo utilizzato dall'operazione intrapresa dall'istruzione

Come visto le istruzioni hanno un argomento che può essere di diversa natura

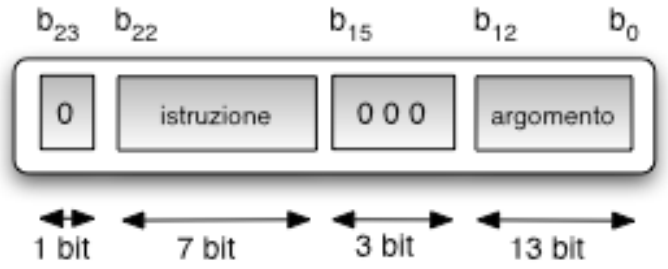
Classi di istruzioni

- 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 $b_{15}\ldots b_{13}$ sono sempre assegnati a zero in quanto inutilizzati. La tipologia dell'argomento è insita nel codice operativo.

Classe α



Classe β



Il formato delle istruzioni di vCPU

In vCPU ci sono 5 tipi di argomento :

- **immediato**: l'operando coincide con l'argomento.
- **registro**: l'operando è il valore contenuto nel registro specificato come argomento.
- **indirizzo diretto**: l'operando è il valore contenuto nella cella di memoria specificata come argomento.

Il formato delle istruzioni di vCPU

- **indirizzo indiretto**: l'operando è il valore contenuto nella cella di memoria il cui indirizzo e' contenuto nella cella di memoria specificata come argomento.
- **registro indiretto**: l'operando è il valore contenuto nella cella di memoria il cui indirizzo è contenuto nel registro specificato come argomento.

Il prefisso "@" indica che il valore effettivo usato dall'istruzione non corrisponde direttamente all'argomento, ma si ottiene da esso.

Il formato delle istruzioni di vCPU

Esempi di questi cinque tipi di argomenti, per una ipotetica istruzione SOMMA ACC che somma il valore dell'accumulatore a quello specificato dall'argomento sono:

- **Immediato**: SOMMA ACC 100
- **Indirizzo diretto**: SOMMA ACC @100
- **Indirizzo indiretto**: SOMMA ACC @@100
- **Registro**: SOMMA ACC BX
- **Registro indiretto**: SOMMA ACC @BX

Ricordiamo che il registro accumulatore in vCPU è il registro AX

Le istruzioni di vCPU

- Sono suddivise nelle seguenti categorie, prescindendo dalla loro classe, per essere analizzate in base alla loro funzione:
  - **Aritmetico logiche** sono le istruzioni che effettuano operazioni sui dati: sono sia aritmetiche che logiche.
  - **Memorizzazione** sono le istruzioni che permettono il flusso di dati dai registri alla memoria e viceversa.
  - **Test, salto e controllo del flusso del programma** sono le istruzioni che possono deviare il flusso del programma.
  - **I/O** sono le istruzioni che permettono il flusso dei dati da e verso l'esterno

Nella prossima lezione…

- Vedremo continueremo a vedere in dettaglio le istruzioni e il loro formato

- Faremo esercizi utilizzando l'emulatore Eniac.

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/