

La CPU

Liceo G.B. Brocchi

Classi prime Scientifico - opzione scienze applicate

Bassano del Grappa, Ottobre 2022

Prof. Giovanni Mazzocchin

La CPU

- La **CPU** (**C**entral **P**rocessing **U**nit) è la componente preposta a:
 - leggere dati e istruzioni dalla memoria centrale
 - eseguire le istruzioni
 - scrivere i risultati delle elaborazioni in memoria o sulle periferiche di output
- Una CPU è composta da 3 elementi:
 - **ALU**: un circuito integrato composto principalmente dalle combinazioni specifiche delle porte logiche che abbiamo visto. Operazioni come la *somma binaria con riporto*, o lo XOR di due byte vengono effettuate dalla ALU
 - **CU** (**C**ontrol **U**nit): governa e impartisce gli ordini di esecuzione alla ALU
 - **Registri**: piccole aree di memoria ad accesso molto veloce, utilizzate per memorizzare provvisoriamente i dati necessari per eseguire un'istruzione

I registri



Registri di uso speciale

- Il **PC (Program Counter)**: contiene l'indirizzo della prossima istruzione da eseguire

			indirizzi di memoria	istruzioni
esecuzione prima istruzione	registro PC		000000	add(op1, op2)
	000000		000001	xor(op1, op2)
esecuzione seconda istruzione	registro PC		000010	sub(op1, op2)
	000001		000011	or(op1, op2)
esecuzione terza istruzione	registro PC		000100	or(op1, op2)
	000010		000101	and(op1, op2)
			000111	mult(op1, op2)
			001000	nand(op1, op2)

Registri di uso speciale

- Il **SR (Status Register)**: detto anche **registro dei flag**
- Ogni bit contenuto in questo registro ha un significato legato allo stato del processore
 - Questi bit dotati di significati specifici vengono chiamati **flag**

Status Register			
ZF (Zero Flag)	CF (Carry Flag)	SF (Sign Flag)	Overflow Flag
1	0	0	0

1: L'ultima operazione aritmetica ha prodotto come risultato 0

0: L'ultima operazione aritmetica **non** ha prodotto un riporto

0: L'ultima operazione aritmetica **non** ha prodotto un numero negativo

0: L'ultima operazione aritmetica ha prodotto un overflow

Registri di uso speciale

- Il **SP (Stack Pointer)**: contiene l'indirizzo della cima dello **stack**. Lo stack è un'area di memoria particolare, organizzata come una pila (struttura LIFO (last-in, first-out)). L'utilizzo di quest'area di memoria diventerà evidente quando programmeremo. Su una pila sono possibili due operazioni: **push** (inserimento in cima) e **pop** (rimozione dalla cima)



Registri di uso speciale

- Il **IR (Instruction Register)**: contiene il codice operativo (**opcode**) dell'istruzione corrente:
 - se l'istruzione corrente è **add a, b**, questo registro contiene un codice binario che rappresenta **add**
 - se l'istruzione corrente è **xor a, b**, questo registro contiene un codice binario che rappresenta **xor**
- Ogni istruzione ha il proprio codice. Ogni processore ha le proprie istruzioni con i propri codici. Ecco un esempio semplificato:

Istruzione	Opcode
add	00
sub	01
xor	10
and	11

Registri di uso generale

- I registri di uso generale, i cui nomi e il cui numero dipendono dall'architettura del processore, memorizzano temporaneamente gli operandi e i dati utilizzati dalle istruzioni
- Ipotizziamo che esistano due registri di uso generale, che chiamiamo **AX** e **BX**, di dimensione 1 byte (8 bit). Una possibile istruzione macchina che somma il contenuto di **AX** e **BX** è:

add AX, BX

- Se AX contiene 00000001 e BX contiene 00000011, il risultato dell'istruzione sarà 00000100 (*ma dove verrà memorizzato questo risultato?*)

L'Unità di Controllo (CU)

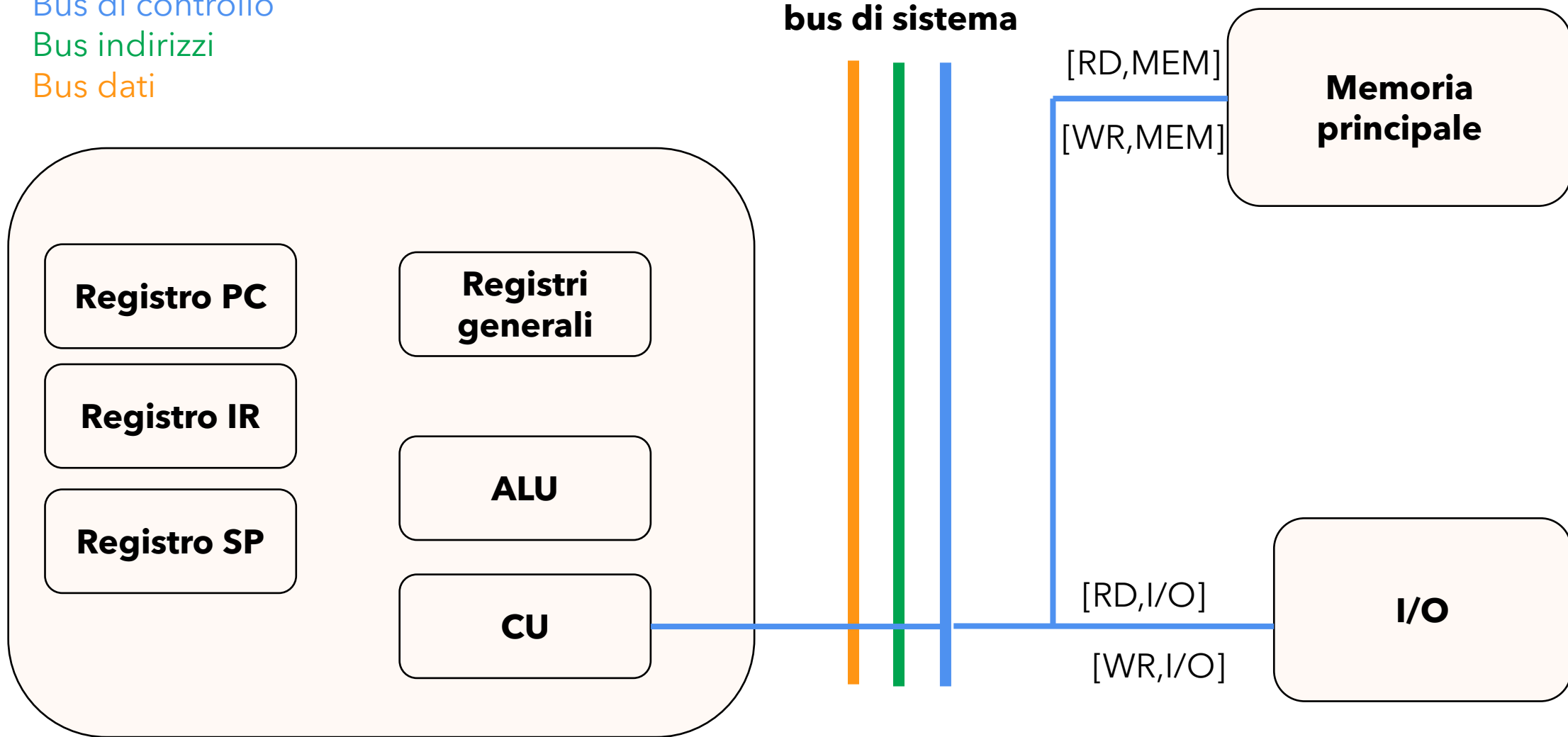
- La CU coordina le operazioni svolte dalla CPU
- In particolare, gestisce il trasferimento di dati tra processore e memoria, e tra processore e I/O inviando sul *bus di controllo* determinati segnali, fatti grosso modo così:
 - **[RD, MEM]**: significa «io processore voglio leggere dalla memoria principale»
 - **[WR, MEM]**: significa «io processore voglio scrivere nella memoria principale»
 - **[RD, I/O]**: significa «io processore voglio leggere da un dispositivo di input»
 - **[WR, I/O]**: significa «io processore voglio scrivere su un dispositivo di output»

Architettura di una generica CPU

Bus di controllo

Bus indirizzi

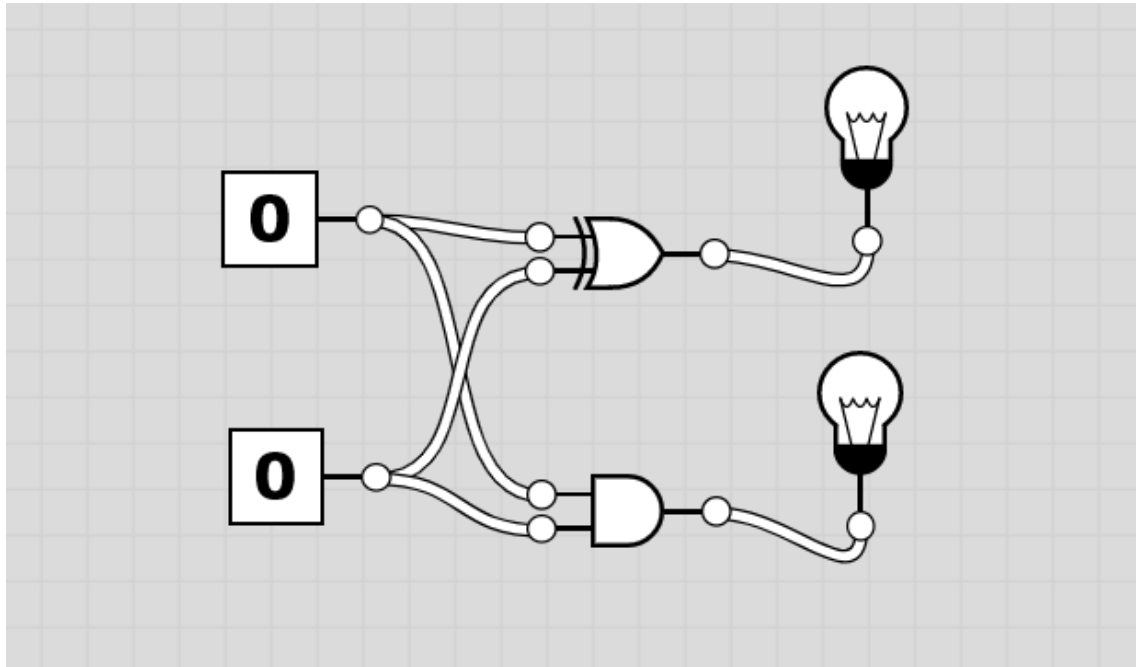
Bus dati



La ALU

- La ALU è costituita da **reti combinatorie** ossia da porte logiche collegate in modo tale da eseguire tutte le operazioni aritmetico/logiche gestite dal microprocessore, come l'addizione e la sottrazione binarie, l'and tra stringhe binarie etc...
- Vediamo un esempio di rete combinatoria che potrebbe far parte di una ALU generica, partendo dal problema specificato qui sotto.
- *Problema: dobbiamo collegare alcune porte logiche in modo da produrre una rete combinatoria in grado di eseguire la somma di 2 bit, memorizzando l'eventuale riporto*

Dentro la ALU: il semi-sommatore (half-adder)

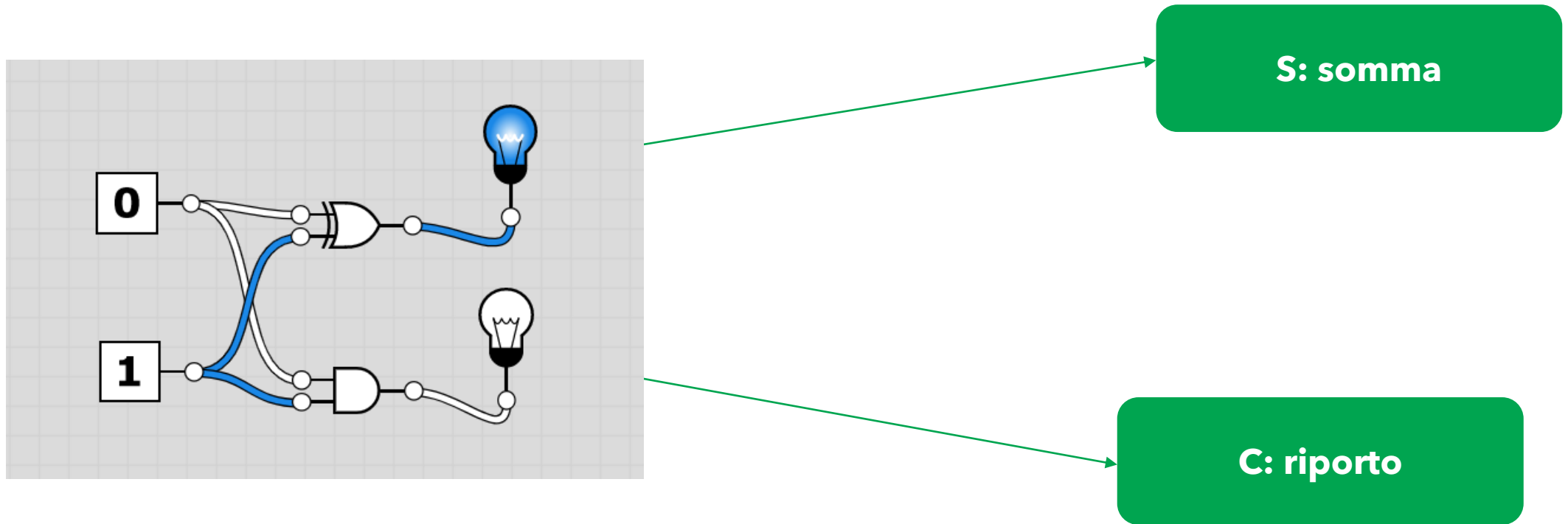


S: somma

C: riporto

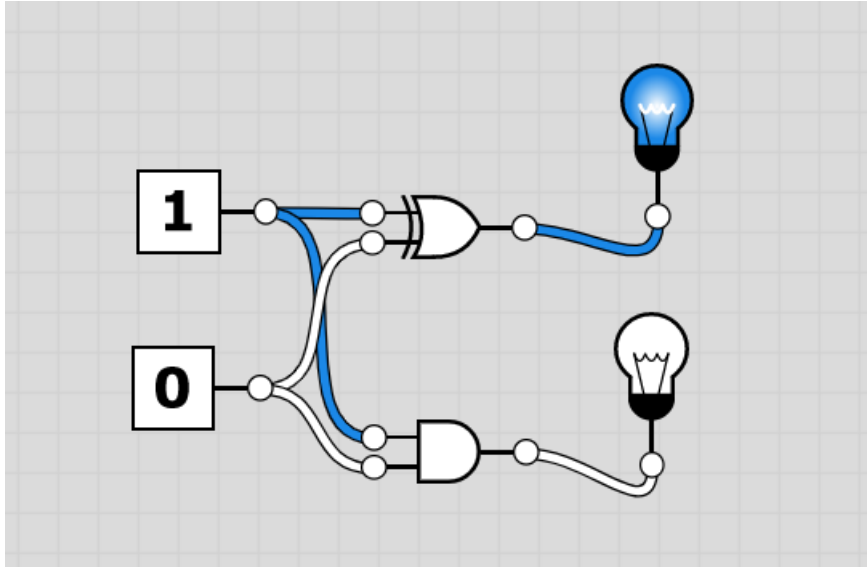
$0 + 0 = 0$, con riporto di 0

Dentro la ALU: il semi-sommatore (half-adder)



0 + 1 = 1, con riporto di 0

Dentro la ALU: il semi-sommatore (half-adder)

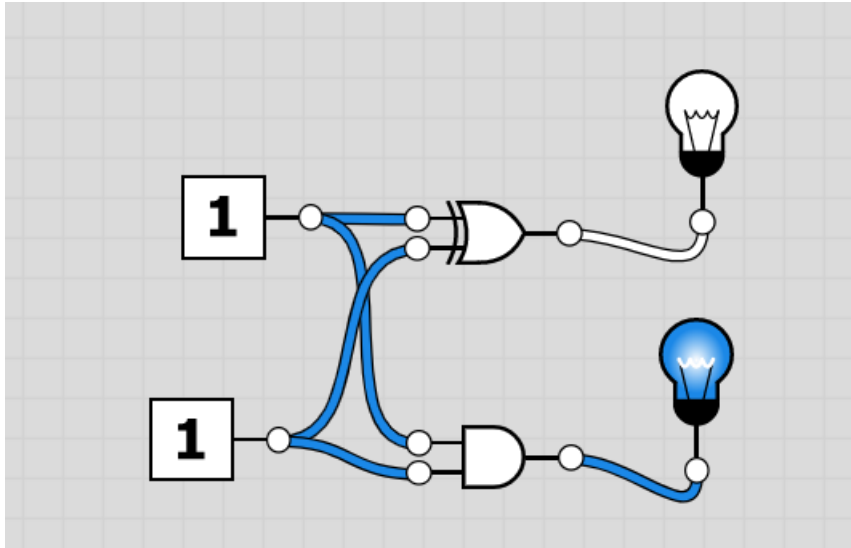


S: somma

C: riporto

1 + 0 = 1, con riporto di 0

Dentro la ALU: il semi-sommatore (half-adder)



S: somma

C: riporto

1 + 1 = 0, con riporto di 1

Il ciclo di esecuzione delle istruzioni

- Il processore agisce secondo una sequenza di passi rigida, eseguita ciclicamente (significa che quando la sequenza viene terminata, essa riprende dal primo passo, e così via fino all'arresto della macchina)

FETCH: la CU pone sul bus indirizzi il valore del registro PC, e imposta sul bus controllo [RD, MEM]

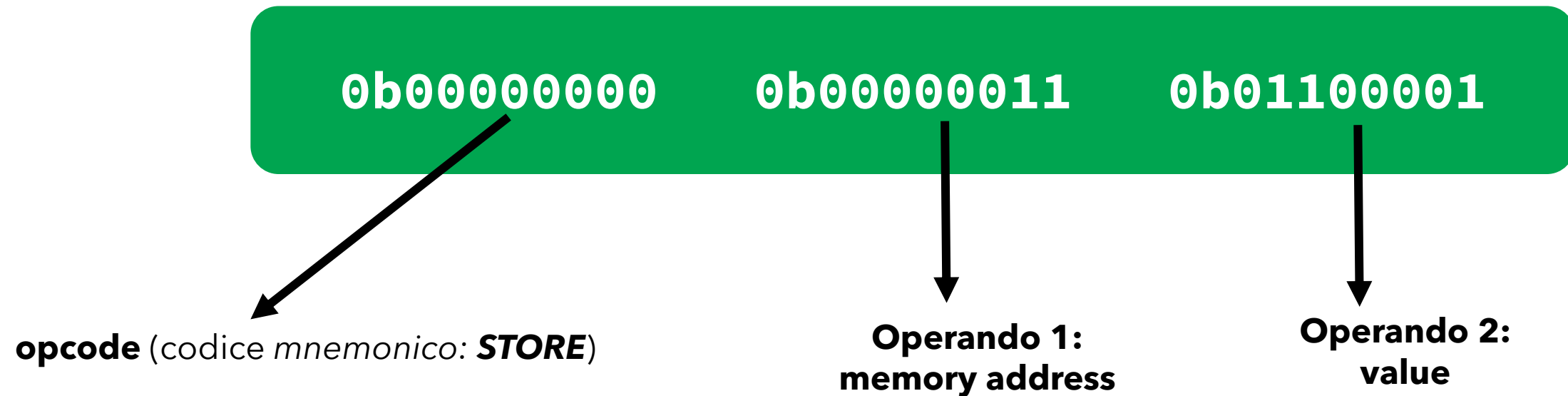
DECODE: la CU legge l'*opcode* dell'istruzione e in base all'*opcode* stesso determina quanti parametri servono, ossia quanto è lunga l'istruzione complessiva da prelevare. Gli operandi caricati dalla memoria vengono copiati nei registri

EXECUTE: viene eseguita l'operazione vera e propria, sulla base dell'*opcode* e degli operandi

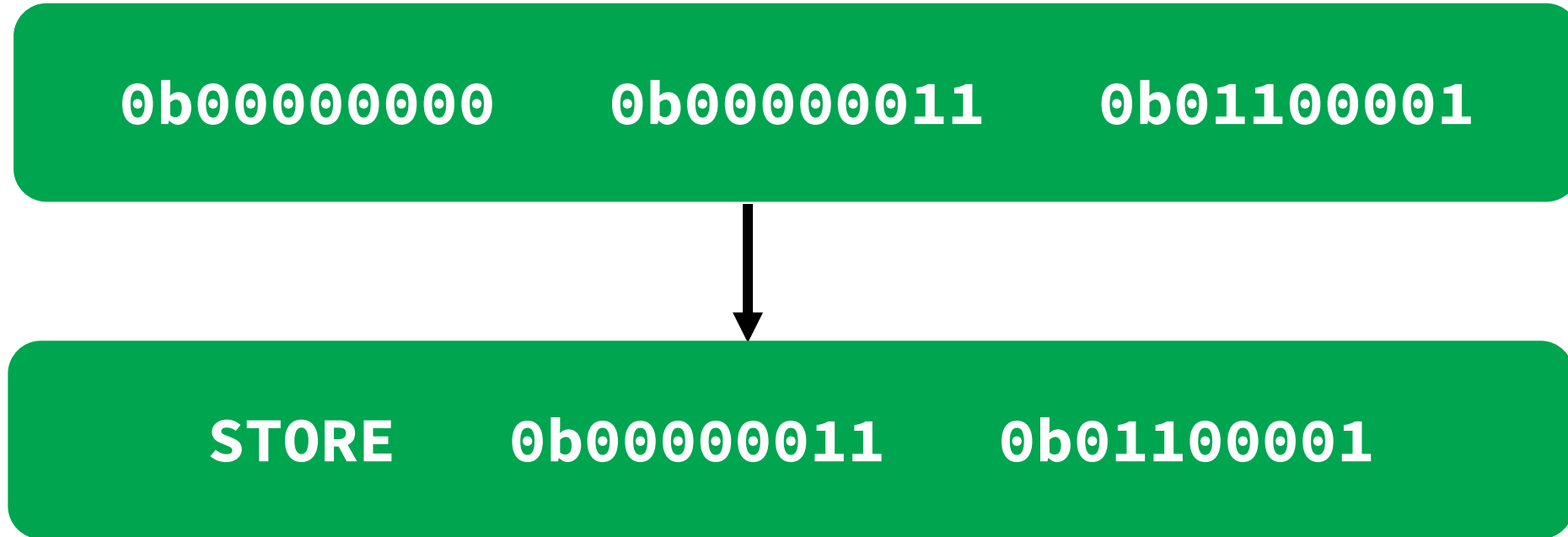
STORE: al termine dell'istruzione, se ci sono risultati, questi vengono scritti in memoria, o verso l'I/O

Instruction Set Architecture

- Ogni processore ha il suo set di istruzioni specifico, denominato **ISA (Instruction Set Architecture)**
- Ad ogni istruzione corrisponde un *microprogramma cablato* in ALU, ossia un circuito in grado di effettuare l'operazione specifica (eg. addizione, sottrazione, operazioni logiche)
- Ecco un esempio di istruzione macchina di una **possibile ISA**:



Instruction Set Architecture



Significato dell'istruzione: *salva nella memoria principale, all'indirizzo specificato dall'operando 1 (0b000000011), il valore specificato dall'operando 2 (0b01100110)*

Tutta l'istruzione precedente è memorizzata nella RAM, e occupa 3 byte:
1 byte per l'opcode, 1 byte per il primo operando e 1 byte per il secondo operando

NB: stiamo parlando di un ISA astratta, non del linguaggio macchina di un microprocessore esistente (*Intel, ARM, AMD etc...*)

Instruction Set Architecture

8-bit memory addresses	0b00000000	0b00000001	0b00000010	0b00000011	0b0000010
content					
other memory locations					
8-bit memory addresses	0b00010000	0b00010001	0b00010010		
content	0b00000000	0b00000011	0b01100001		

STORE

operand 1
(memory address)

operand 2
(value)

PC: 0b00010000 → il programma inizia all'indirizzo 0b00010000

Instruction Set Architecture

Stato della memoria dop l'esecuzione dell'istruzione STORE

8-bit memory addresses	0b00000000	0b00000001	0b00000010	0b00000011	0b0000010
content				0b01100001	
other memory locations					
8-bit memory addresses	0b00010000	0b00010001	0b00010010		
content	0b00000000	0b00000011	0b01100001		

STORE

operand 1

operand 2

*È stato scritto il valore
specificato da operand
2 nella locazione di
memoria specificata da
operand 1*

Instruction Set Architecture

STORE 0b000000011 0b01100001

Di quanto deve essere incrementato il contenuto del PC (Program Counter) dopo l'esecuzione di questa istruzione?