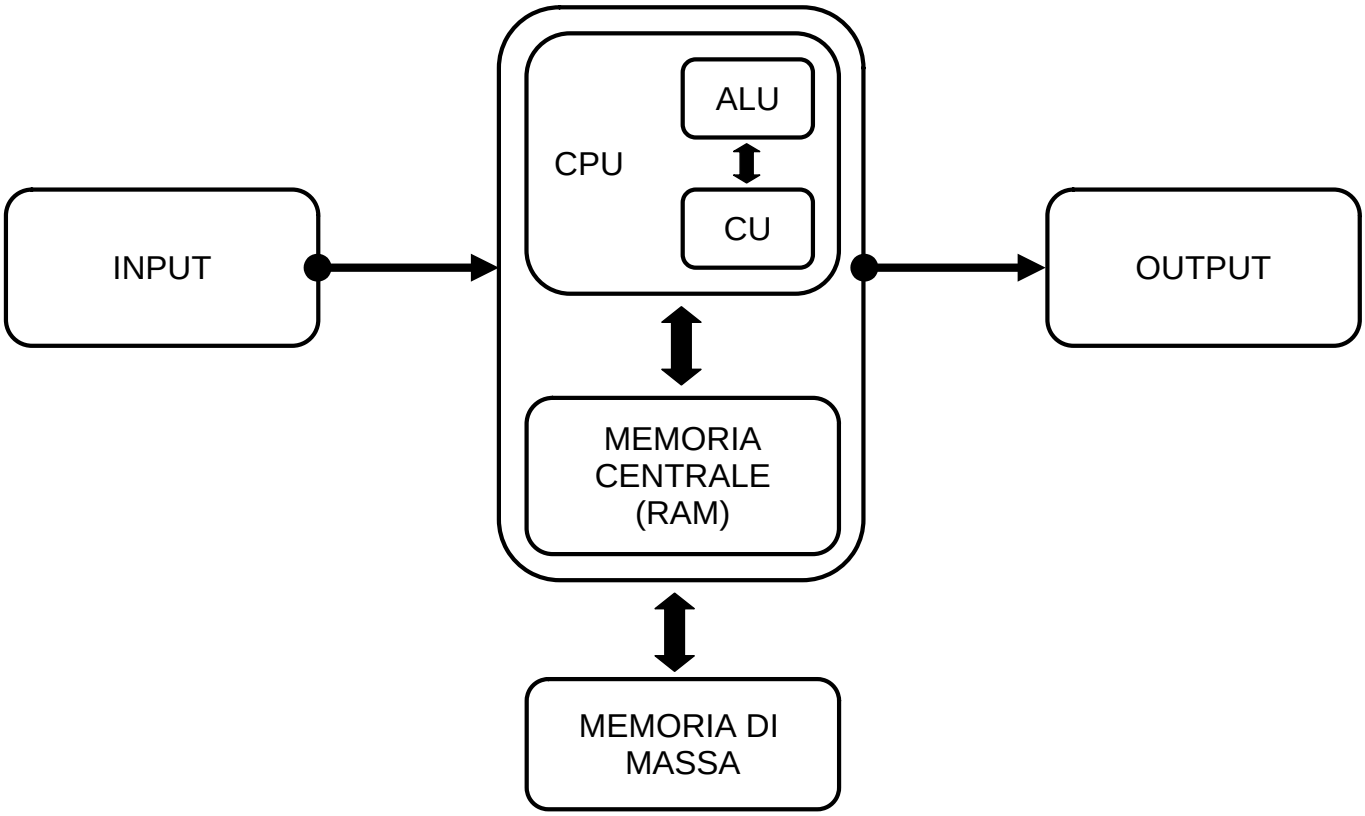
ARCHITETTURA DI UN ELABORATORE

componenti e struttura



periferiche di input

Dispositivi attraverso cui si immettono dati nel sistema {: .fragment}

memoria centrale

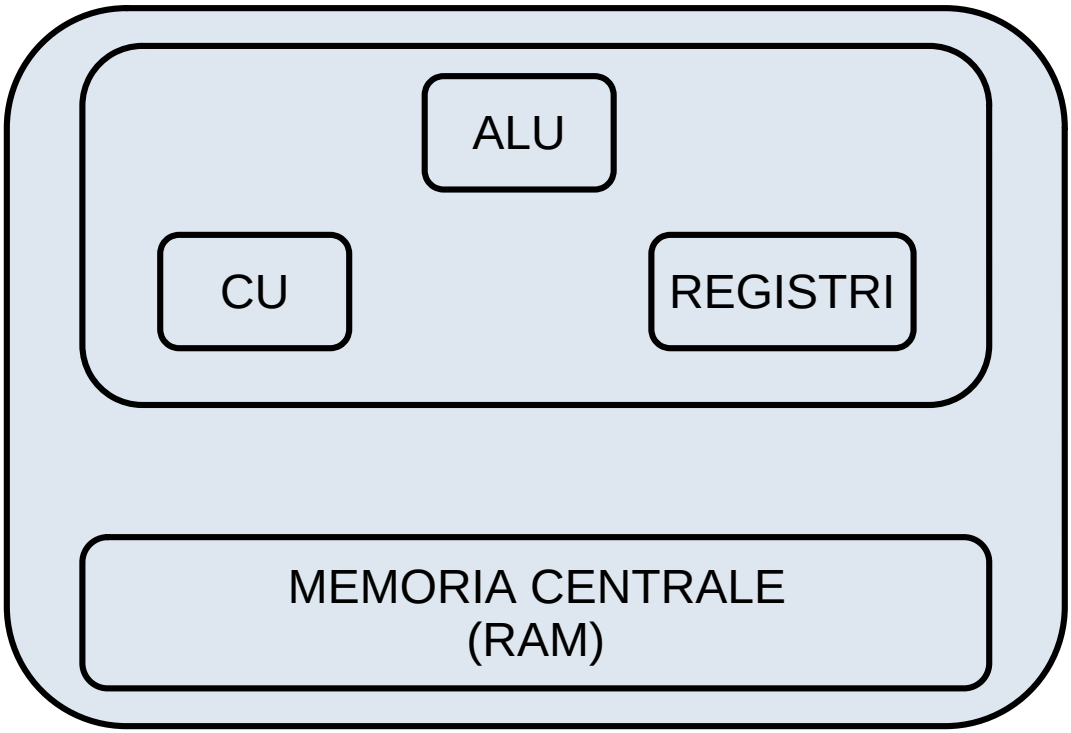
Dispositivo in cui viengono memorizzati i dati e i programmi

cpu

Central Processing Unit, composta da:

- ALU (Arithmetic Logic Unit): esegue calcoli aritmetici {: .fragment}
- CU (Control Unit): gestisce le istruzioni {: .fragment}
- REGISTRI: Memorizza dati e istruzioni prelevati dalla memoria centrale {: .fragment}

--



---

periferiche di output

Dispositivi attraverso cui il sistema fornisce i risultati delle elaborazioni

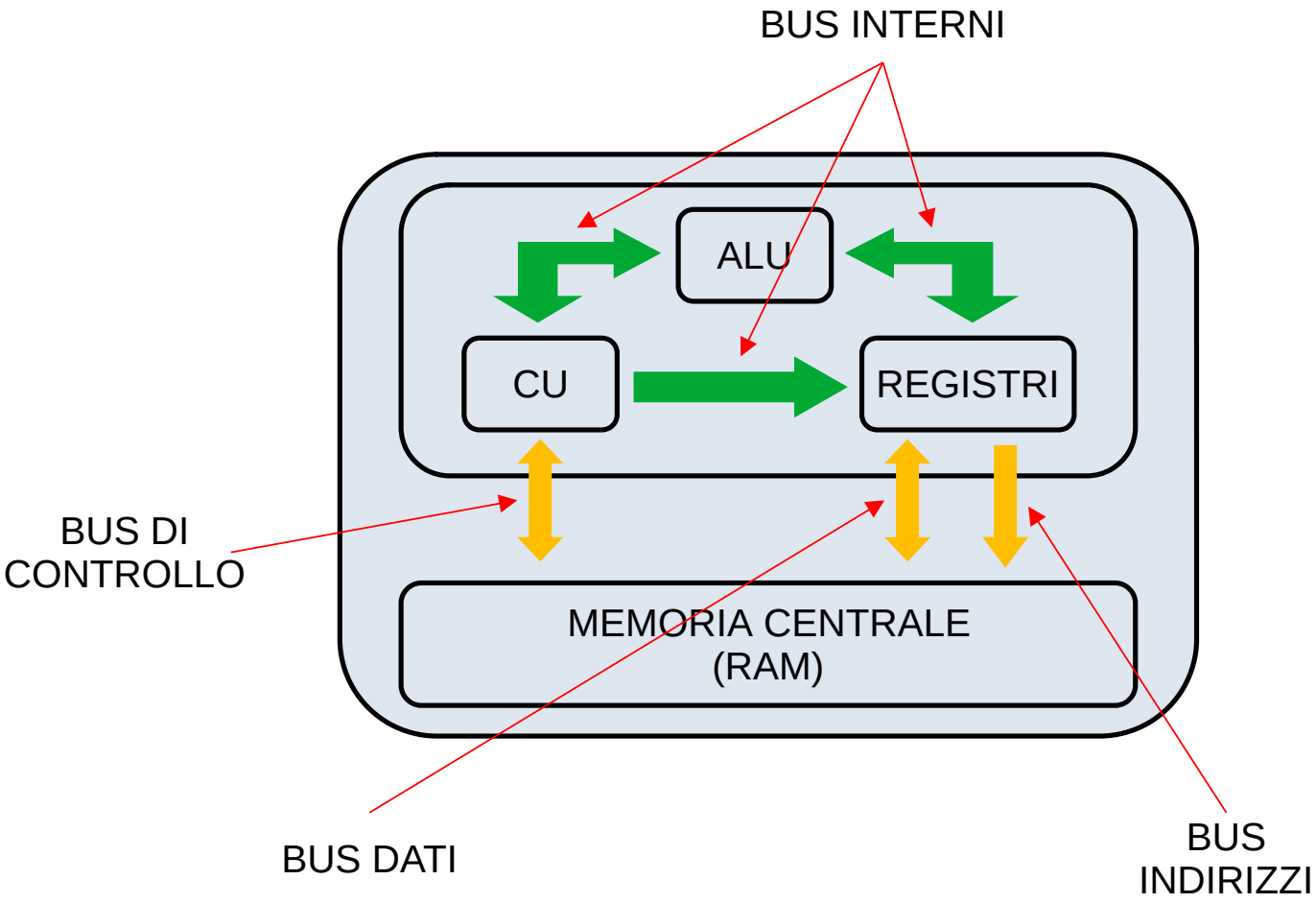
---

memoria di massa

Dispositivo che memorizza dati e programmi in modo permanente

---

STRUTTURA DI UNA CPU



---

bus

- Canali di comunicazione che i componenti usano per scambiarsi i dati {: .fragment}
- Formati da piste conduttive che trasportano impulsi elettrici (bit) {: .fragment}
- Hanno dimensione diversa: 16, 32, oggi 64 {: .fragment}

---

registri

- memorie interne alla CPU {: .fragment}
- la ALU esegue le istruzioni (calcoli) contenute nei registri su dati anch'essi contenuti nei registri {: .fragment}
- velocissime, ma di dimensioni ridotte (32bit nell'architettura x86, 64 bit nell'architettura x64) {: .fragment}
- Sono poche, generalmente 16 o 32 registri *general purpose* e altrettanti registri specifici {: .fragment}

---

alu

- Esegue operazioni aritmetiche {: .fragment}
- Svolge un numero limitato di operazioni logiche fra gruppi di bit {: .fragment}

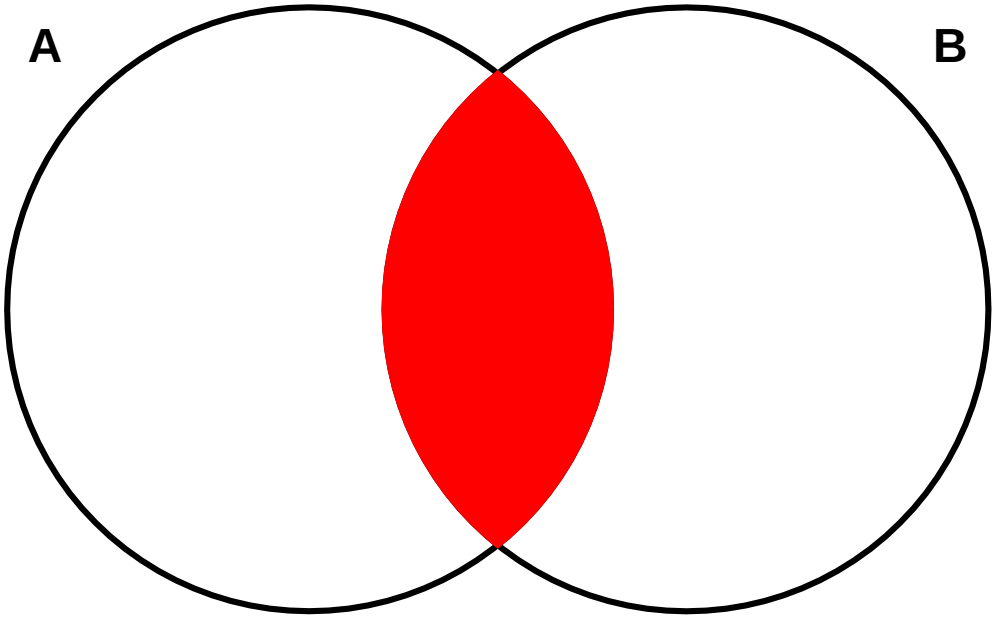
AND

A	B	$A \land B$

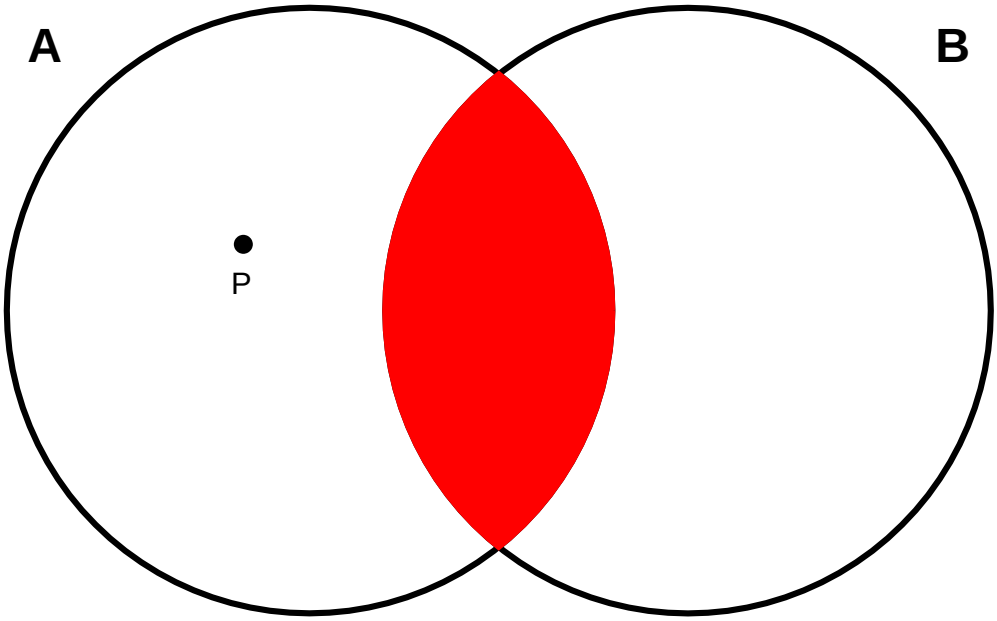
A	B	$A \land B$
0	0	0
0	1	0
1	0	0
1	1	1

---

intersezione



--



--

il punto p appartiene ad A? Si (1)

il punto p appartiene a B? No (0)

$$ 1 \land 0 \implies 0 $$

---

esempio di operazione AND bit a bit:

```
1 0 1 0 0 1 0 1 AND
0 1 0 1 1 1 0 1 =

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

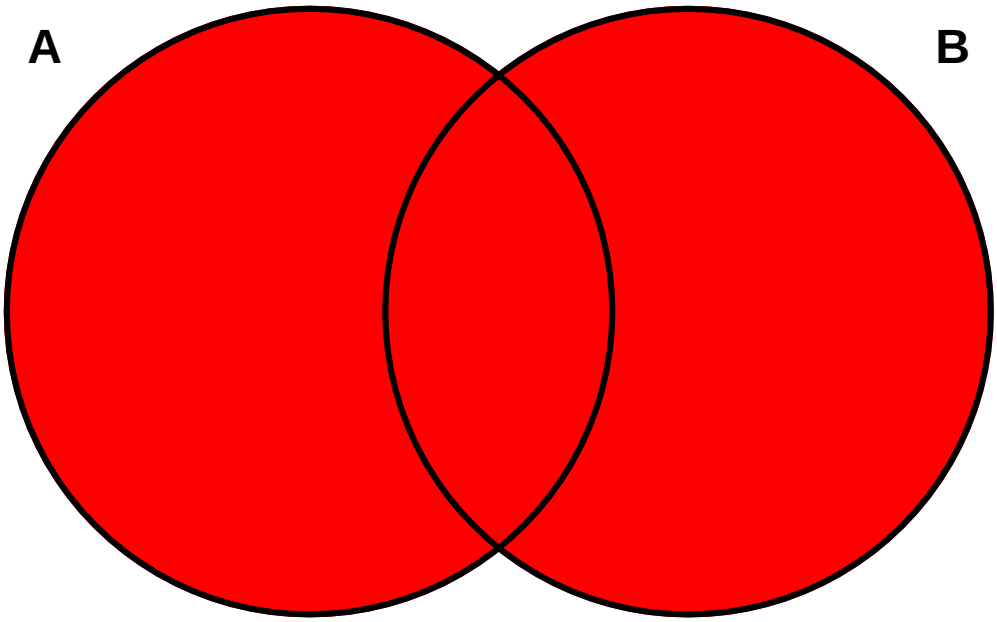
L'operazione è su gruppi di bit: `byte` {: .fragment}

OR

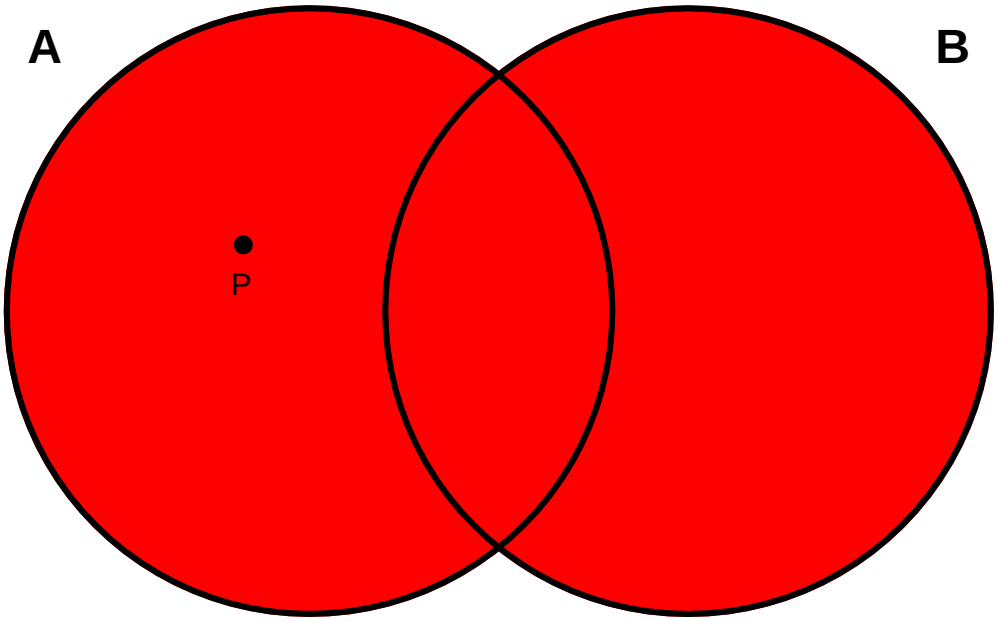
A	B	$A \lor B$
0	0	0
0	1	1
1	0	1
1	1	1

---

unione



--



--

il punto p appartiene ad A? Si (1)

il punto p appartiene a B? No (0)

$$ 1 \lor 0 \implies 1 $$

CU

La **Control Unit** coordina tutte le azioni necessarie ad eseguire le `ISTRUZIONI`

---

schematizzando...

- La CPU delle macchine moderne è costituita da:
  - ALU (esegue i calcoli) {: .fragment}
  - CU (coordina le operazioni) {: .fragment}
  - Registri (memorizzano istruzioni e dati) {: .fragment}
- Le istruzioni sono contenute in sequenza nei `programmi` {: .fragment}

---

esempio

programma SOMMA

--

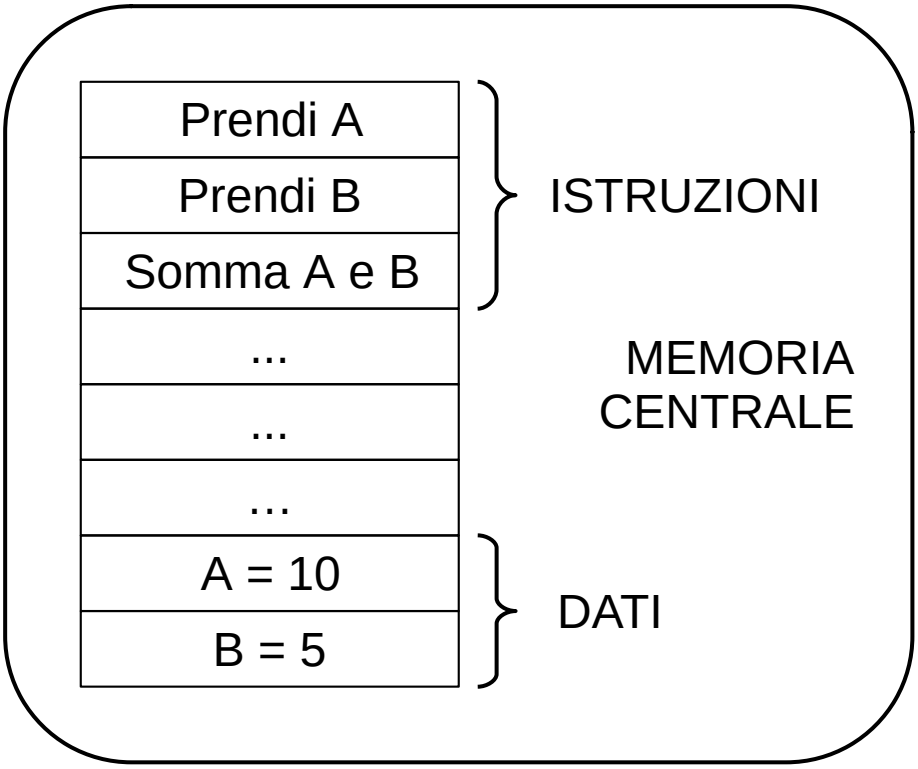
istruzioni

- Prendi A
- Prendi B
- Somma A e B

Nella memoria centrale (RAM) ci sarà:

- il programma, con l'elenco delle istruzioni {: .fragment}
- i dati, cioè gli addendi della somma {: .fragment}

--



---

il compito della CPU è quello di andare a prendere ognuna delle istruzioni e dei dati, eseguire le operaizoni e restituire il risultato

---

più in dettaglio...

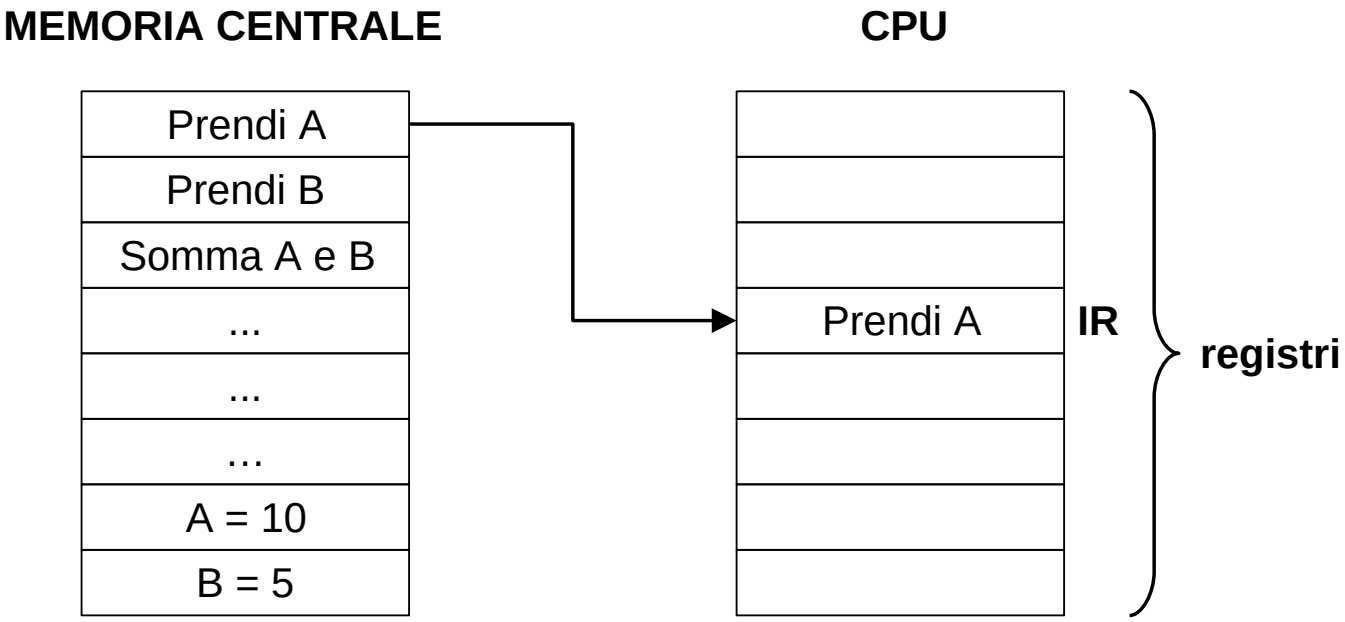
- La CPU prende l'istruzione (`fetch`)
- La interpreta (`decode`)
- La esegue (`execute`)

---

fetch

È l'operazione con cui la **CU** va a prelevare un'istruzione dalla memoria centrale e la sposta dentro la CPU, in un registro specifico che si chiama `instruction register` (**IR**)

--



---

decode

È l'operazione con cui la CU interpreta l'istruzione, ovvero capisce cosa deve fare e si prepara ad agire di conseguenza

--

in pratica...

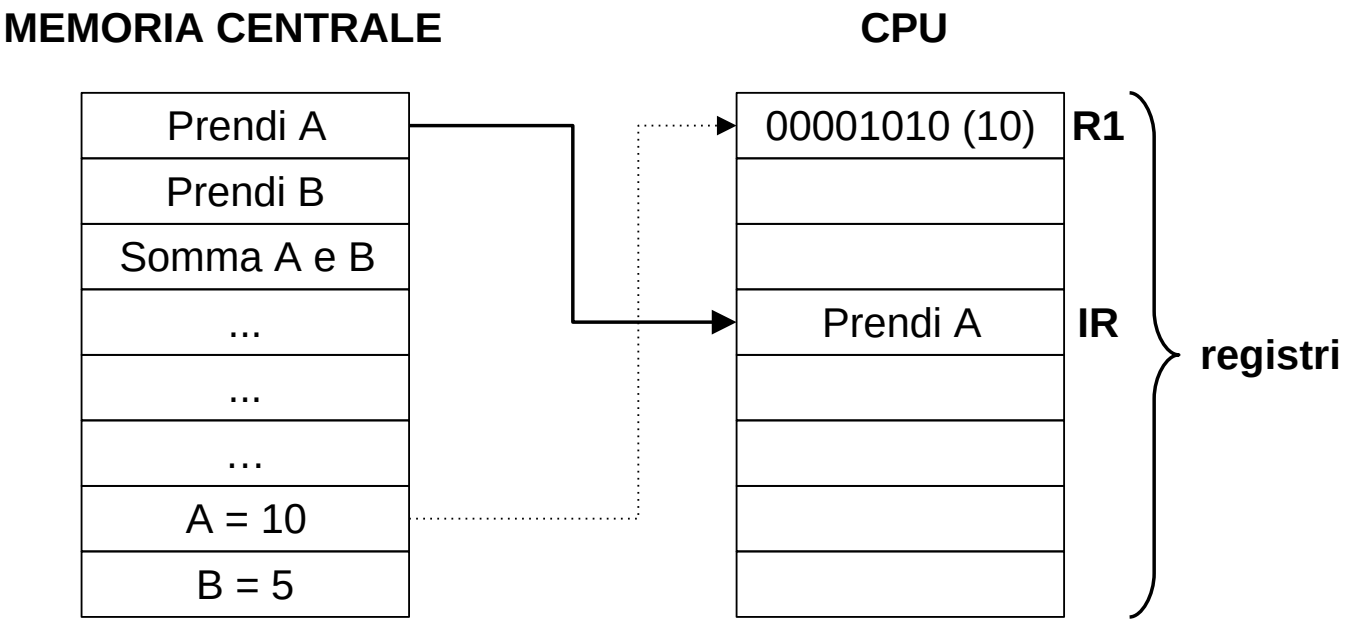
`Prendi A` = Leggi il valore contenuto in A e copialo in **R1** (Registro 1)

---

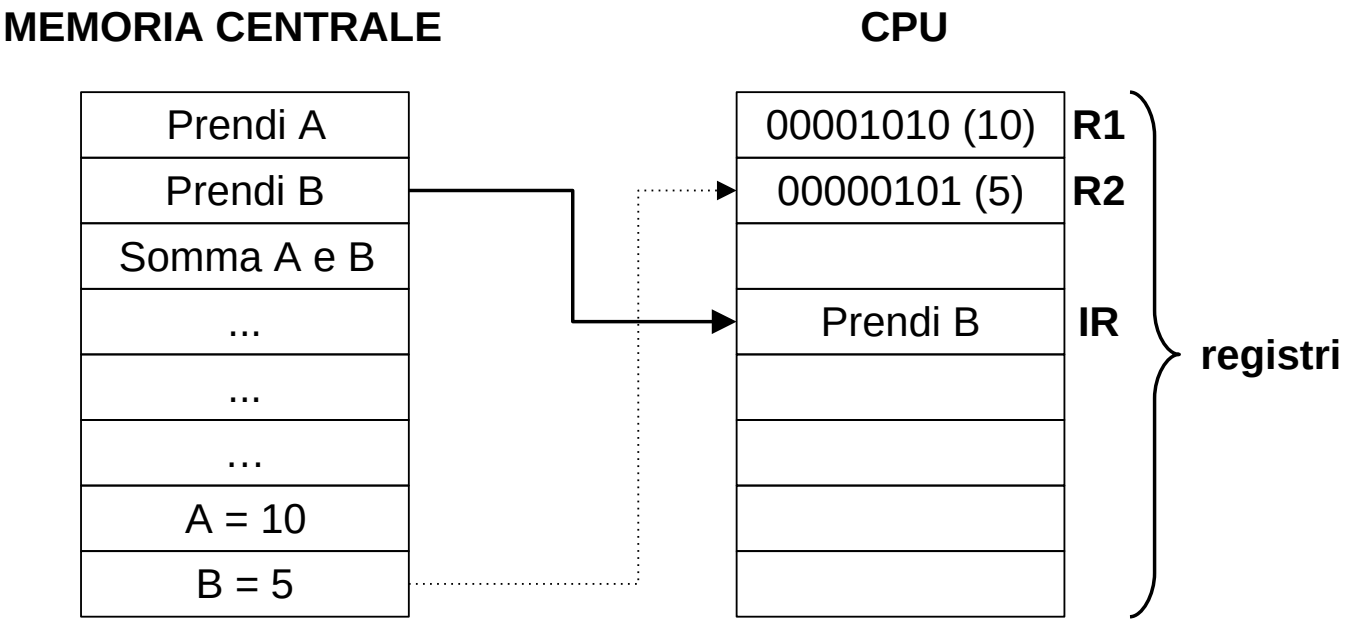
execute

La CU esegue l'istruzione chiamando in causa ALU e i registri, se necessario

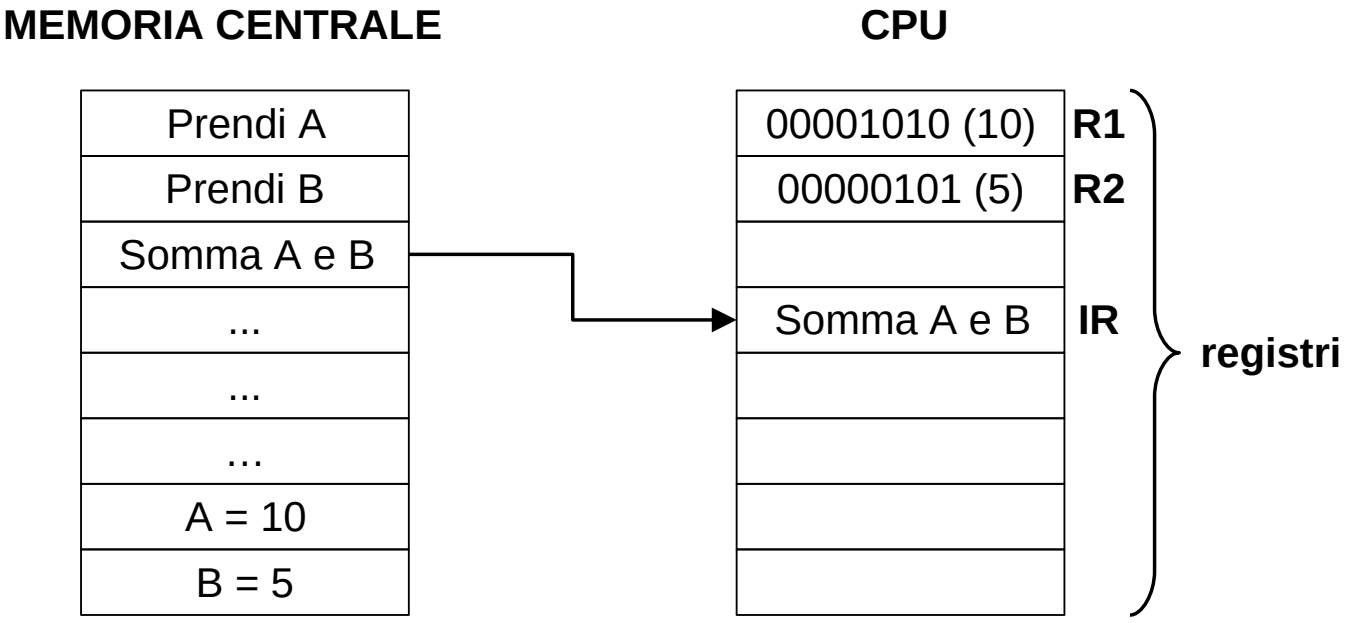
--



E poi...



E infine...



---

decode

`Somma A e B` = Somma il contenuto dei registri **R1** e **R2** e scrivi il risultato in **R3**

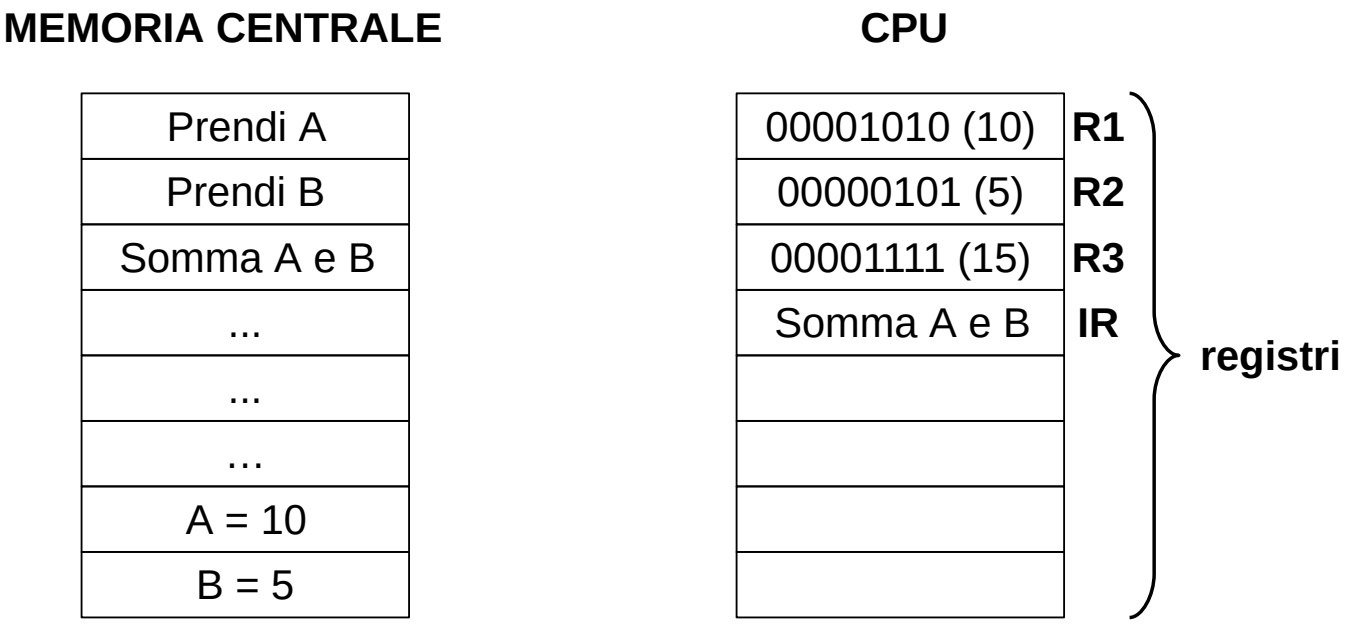
---

execute

```
0 0 0 0 1 0 1 0 +
0 0 0 0 0 1 0 1 =

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

--



---

decode più in dettaglio

- Il processore parla un linguaggio complesso, fatto di valori binari che vengono scambiati dai registri. Per rendere la programmazione più agevole ogni processore è dotato di un **instruction set** {: .fragment}
- L'instruction set è l'insieme delle operazioni che un processore può svolgere. Viene espresso in un linguaggio semplificato in modo che i programmatori possano impartire comandi al processore senza dover scrivere manualmente complicate operazioni tra i registri {: .fragment}
- ad esempio l'istruzione **ADD(R1, R2)** permette di sommare il contenuto dei registri *senza* specificare quali operazioni logiche deve eseguire la ALU per produrre il risultato {: .fragment}

--

Assembly

- Le istruzioni presenti nell'instruction set permettono di scrivere software usando un linguaggio di programmazione chiamato **ASSEMBLY** {: .fragment}
- Linguaggio di basso livello, che permette di usare solo istruzioni molto semplici {: .fragment}
- Ci sono due diverse architetture basate sull'istruction set: {: .fragment}
  - RISC (Reduced Instruction Set Computer): poche istruzioni, molto veloce
  - CISC (Complex Instruction Set Computer): molte istruzioni, meno veloce, ma più potente

---

CLOCK

- La velocità di un processore si misure in GigaHertz (GHz) e viene chiamata `frequenza di clock` {: .fragment}
- Questo valore esprime il numero di cicli al secondo {: .fragment}
- Un ciclo è una sequenza `fetch-decode-execute` {: .fragment}

- La sincronizzazione del lavoro di tutti i transistor all'interno della CPU avviene per mezzo di un `oscillatore al quarzo` (come quelli usati negli orologi) {: .fragment}

---

oscillatore al quarzo

- Circuito contenente un piccolo frammento di materiale cristallino {: .fragment}
- Quando esposto a un campo elettrico, produce un segnale elettrico che oscilla con una frequenza precisa tra due livelli di tensione {: .fragment}
- I transistor quindi cambiano stato tante volte al secondo, quant'è la frequenza di clock {: .fragment}
- I cicli di clock determinano la velocità del processore {: .fragment}