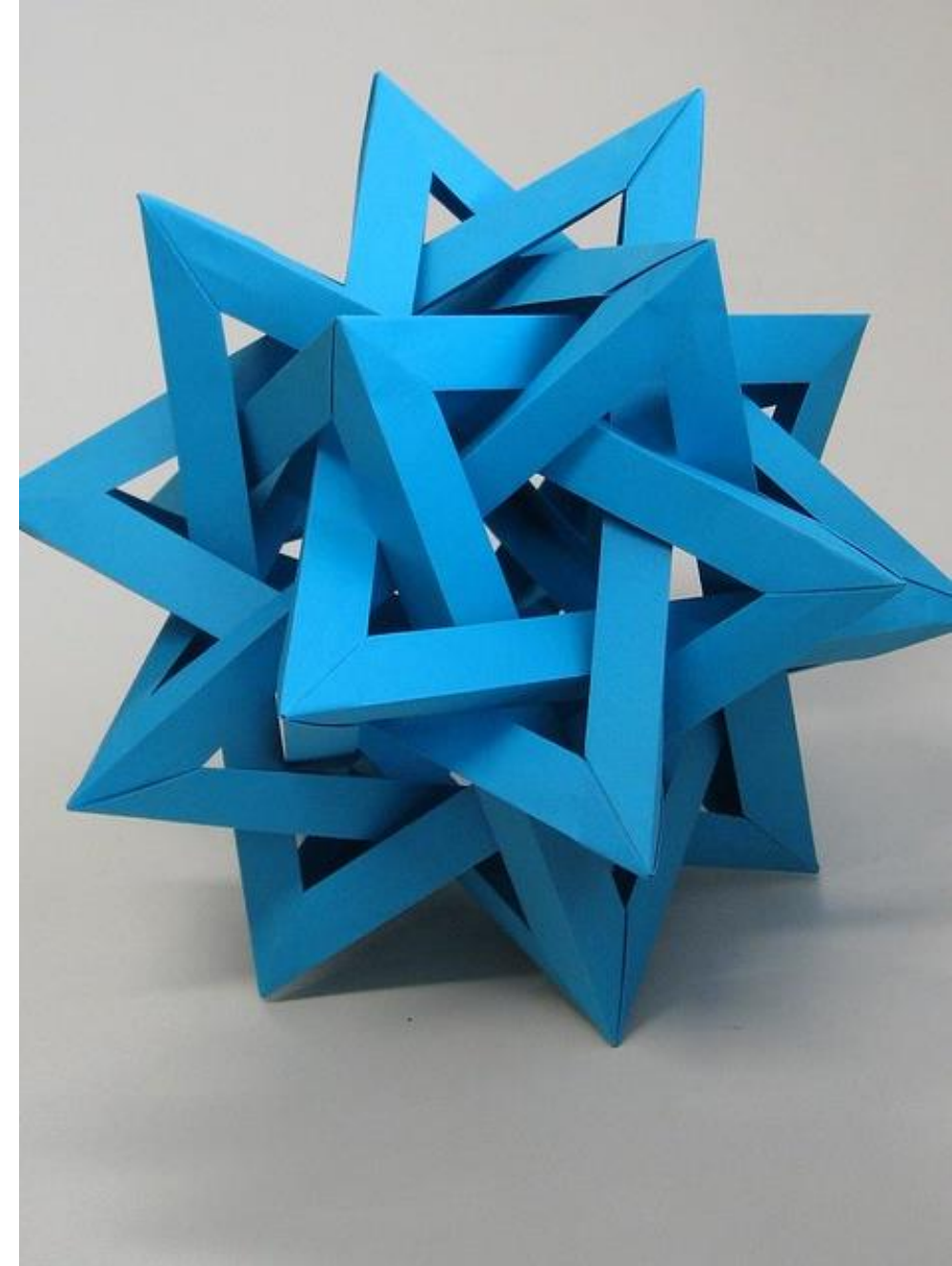




# Unità T2: Architettura degli elaboratori

---



[This Photo](#) by Unknown Author is licensed under [CC BY-SA](#)

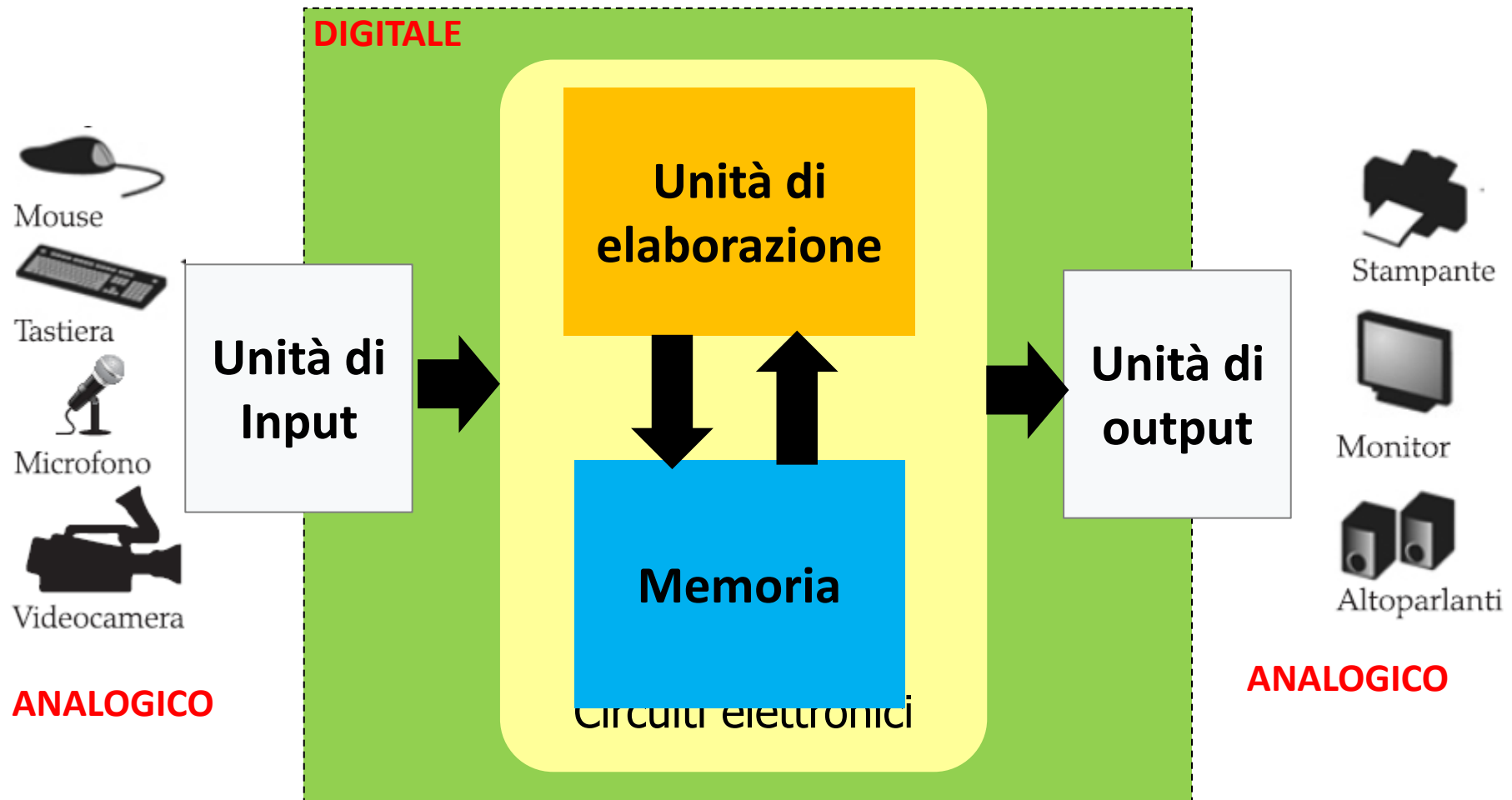
# Architettura degli elaboratori

---

# Architettura di un elaboratore

- Per comprendere il processo di programmazione, è necessario conoscere almeno per grandi linee gli elementi costitutivi di un computer.
- Faremo riferimento al tipico PC, anche se tutti i calcolatori hanno sostanzialmente la stessa struttura.

# I blocchi fondamentali dell'elaboratore



# Componenti fondamentali

## ■ Unità di I/O

- Interfaccia da/verso utente
- Implicano un cambio di dominio fisico
  - Umano = analogico, asincrono, non elettrico
  - Calcolatore = digitale, sincrono, elettrico
  - Necessarie opportune conversioni

## ■ Unità di elaborazione

- Contiene i circuiti per l'esecuzione delle 'istruzioni'
- "microprocessore"

## ■ Memoria

- Memorizza in modo permanente dati e programmi
- Necessaria per l'elaborazione per motivi di efficienza

**CIRCUITI  
(dentro la  
"scatola")**

# Il microprocessore

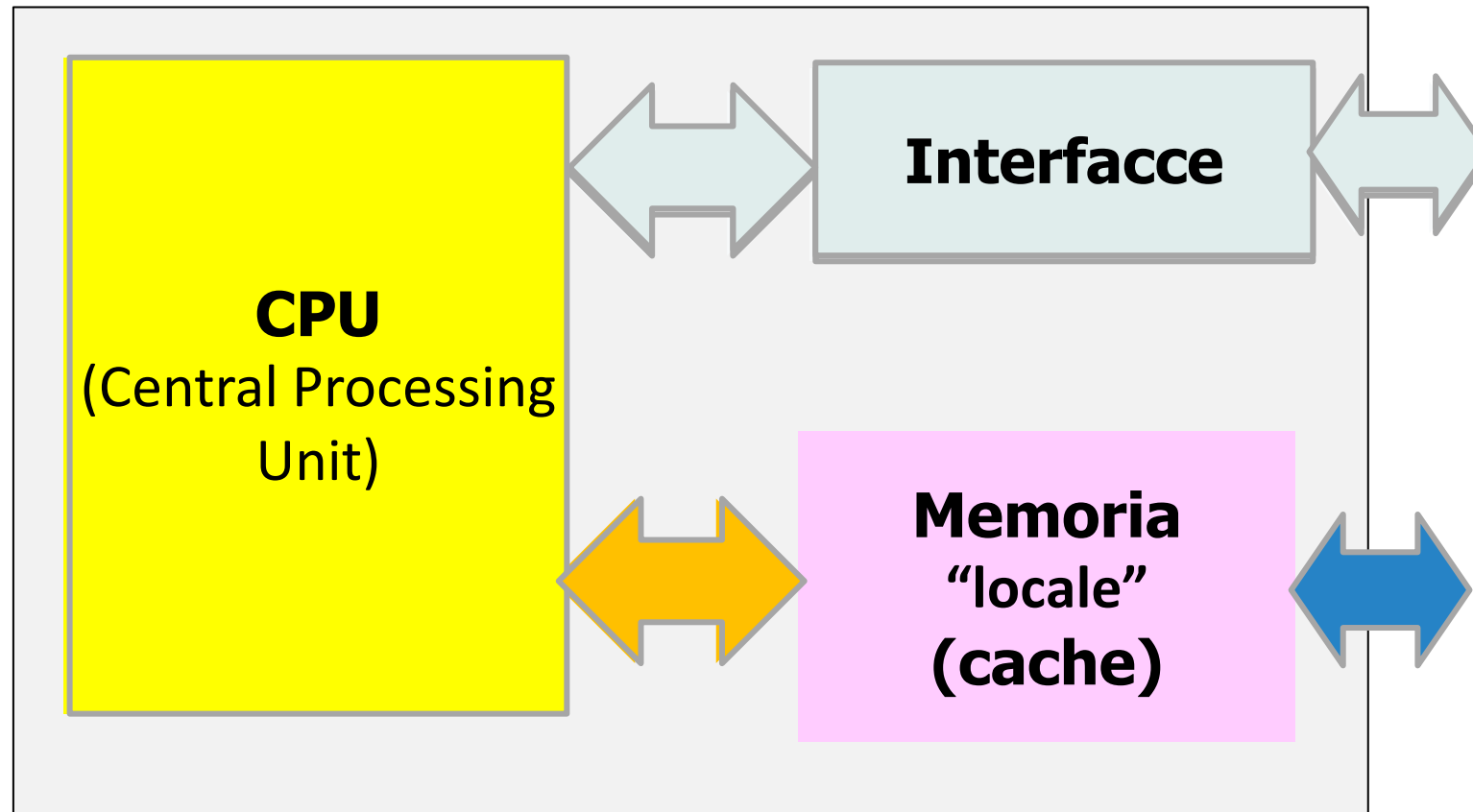
---

# Microprocessore

- Il microprocessore è il circuito che fisicamente esegue TUTTE le istruzioni
- Contiene quindi:
  - Tutti i circuiti per eseguire le operazioni di base su numeri interi, reali e operazioni logiche
  - Opportuni circuiti per il “coordinamento” dell’esecuzione delle istruzioni (per es. il loro sequenziamento, controllo degli errori)
  - Interfacce per spostare dati da/verso la memoria
  - Interfacce per spostare dati da/verso unità di I/O
- Ha (in linea di principio) limitate capacità di memorizzare dati e/o istruzioni
  - Lo stretto necessario per eseguire le operazioni
  - Ma per motivi di efficienza (v. dopo) una parte della memoria è “ospitata” nel microprocessore

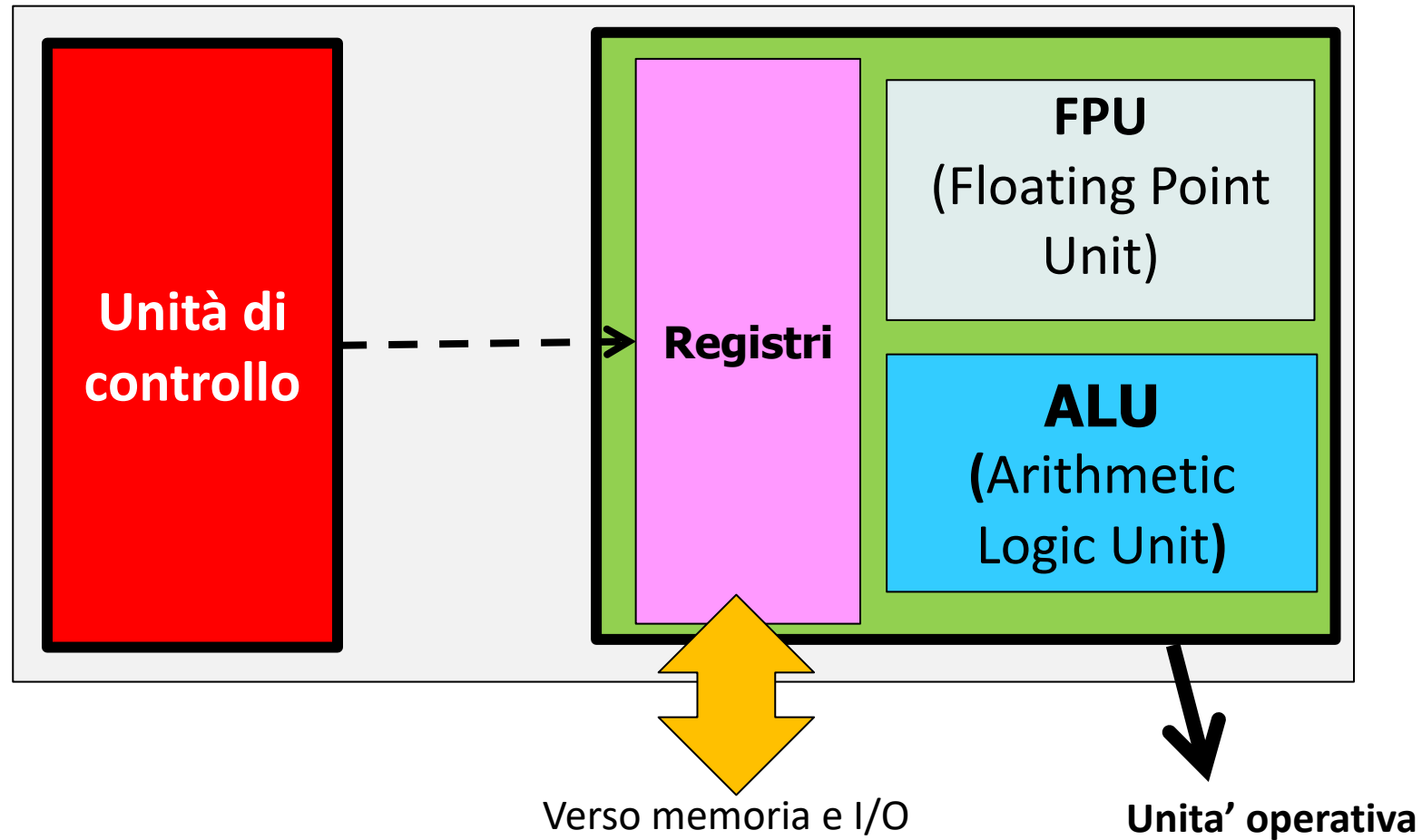


# Struttura del microprocessore





# La CPU

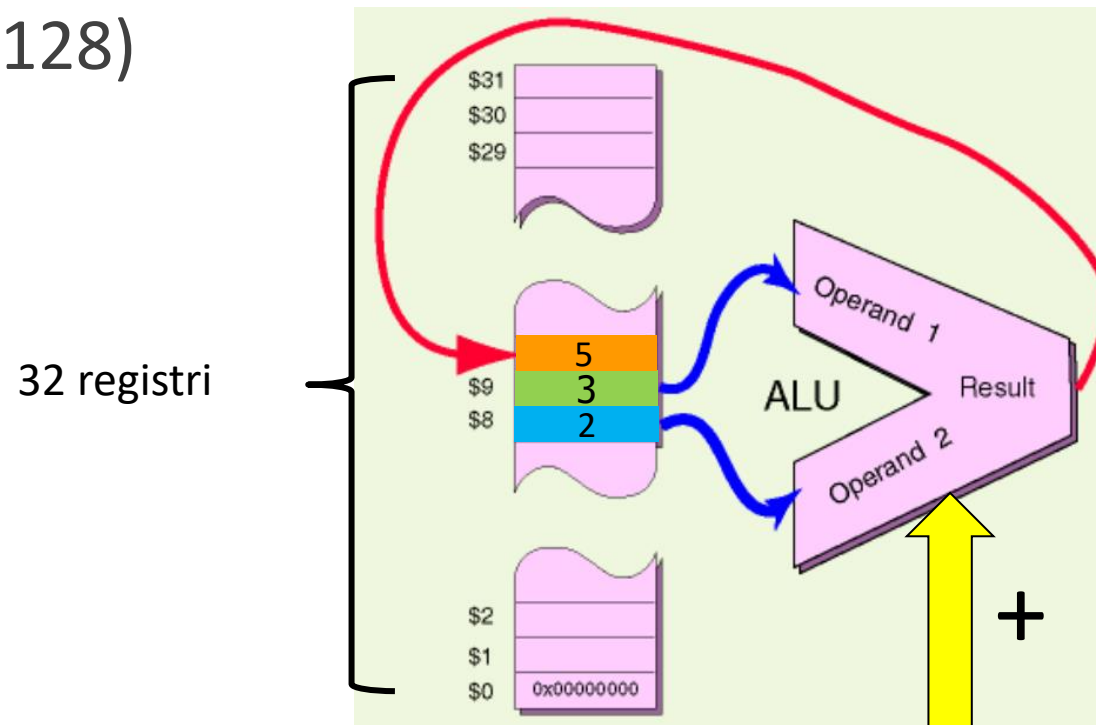


# Unità operativa

- Svolge tutte le elaborazioni richieste (aritmetiche, logiche, grafiche?, ...).
- È composta di:
  - ALU (Arithmetic Logic Unit)
  - Registri istruzioni e dati
  - FPU (Floating Point Unit)
  - Registro dei Flag

# Registri

- Elementi di memoria locale usati per conservare temporaneamente dei dati (es. risultati parziali) o istruzioni  
Ogni trasferimento da processore a memoria e viceversa avviene tra registri e memoria
- Numero limitato (8...128)



# Flag

- Registro che contiene un insieme di bit che segnalano determinati stati dell'insieme delle unità di calcolo e alcune informazioni sul risultato dell'ultima operazione eseguita
- Utilizzati per implementare alcune condizioni
- Alcuni flag significativi
  - Zero: Segnala se il risultato dell'operazione è o no zero.
  - Segno: indica il segno del risultato dell'operazione precedente
  - Overflow: indica se il risultato dell'operazione precedente eccede i limiti della rappresentazione

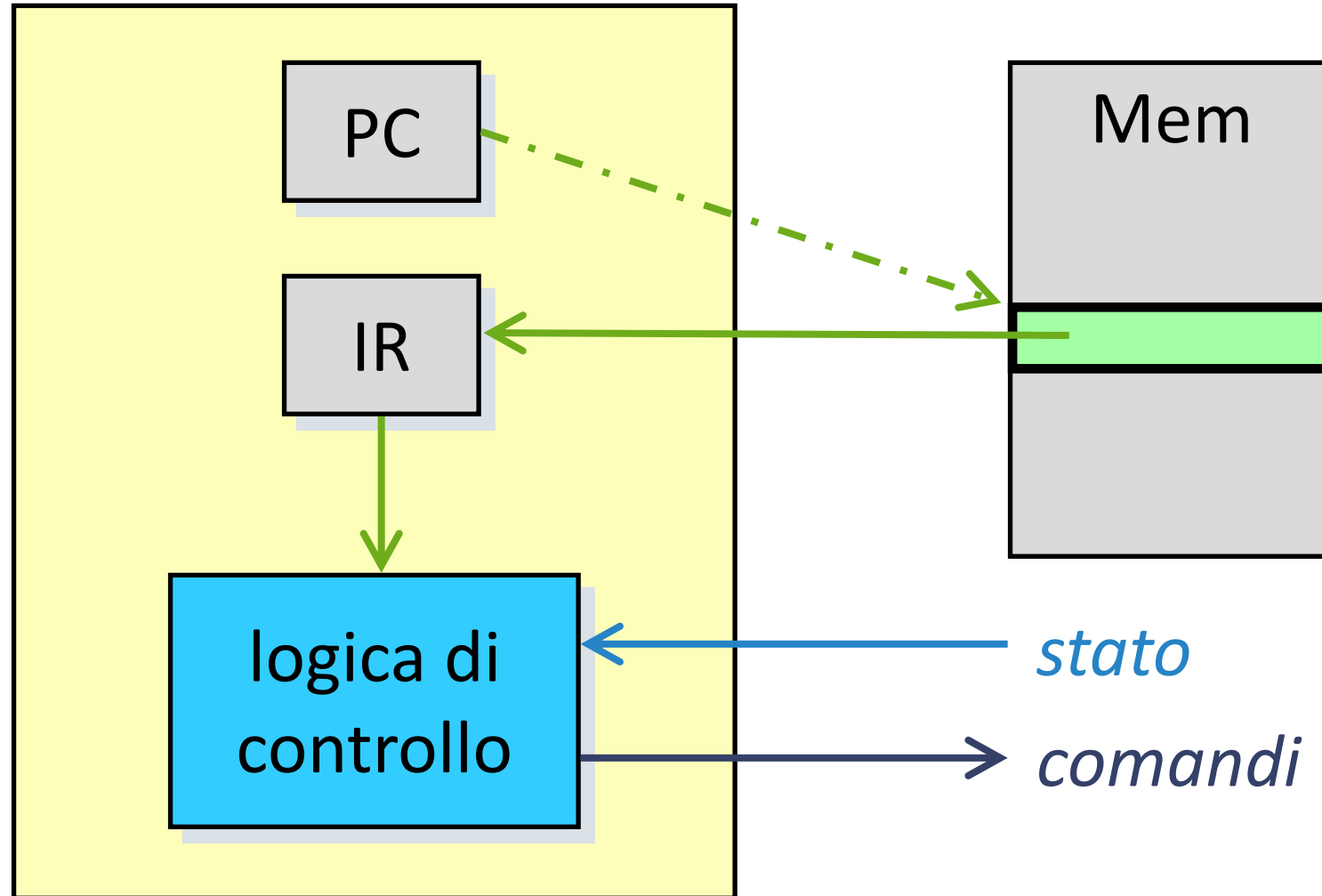
# ALU e FPU

- ALU
  - Svolge tutti i calcoli (aritmetici, logici, confronti) su numeri interi
- FPU:
  - Svolge i calcoli su numeri reali
- Notevole differenza nel tempo di esecuzione!
  - Il rapporto dipende dallo specifico processore, un'operazione FPU è tipicamente più lenta di 5-50 volte più lenta di un'operazione ALU.

# Unità di controllo

- È il cuore dell'elaboratore:
- In base alle istruzioni contenute nel programma che esegue ed allo stato di tutte le unità decide l'operazione da eseguire ed emette gli ordini relativi

# Unità di controllo: schema funzionale



# Componenti dell'UC

- PC (Program Counter)  
registro che indica sempre l'indirizzo della cella di memoria che contiene la prossima istruzione da eseguire
- IR (Instruction Register)  
registro che memorizza temporaneamente l'operazione corrente da eseguire
- Logica di controllo  
interpreta il codice macchina in IR per decidere ed emette gli ordini che le varie unità devono eseguire



# Esecuzione di un'istruzione

- Tre fasi:

**FETCH**     $IR \leftarrow M[PC]$   
               $PC \leftarrow PC + 1$

**DECODE**    $ordini \leftarrow \text{decode}(IR)$

**EXECUTE**   attiva i blocchi  
                  interessati  
                  dall'istruzione



- $IR \leftarrow M[PC]$ : preleva dalla memoria l'istruzione nella posizione indicata da PC
- $PC \leftarrow PC + 1$ : incrementa il valore di PC (al passo successivo conterrà la prossima istruzione da eseguire)

# Un collegamento ai flowchart...

- Conoscendo ora i blocchi fondamentali, siamo in grado di valutare quali operazioni sono da considerarsi “elementari” (e sono usabili dentro i blocchi di un DDF)

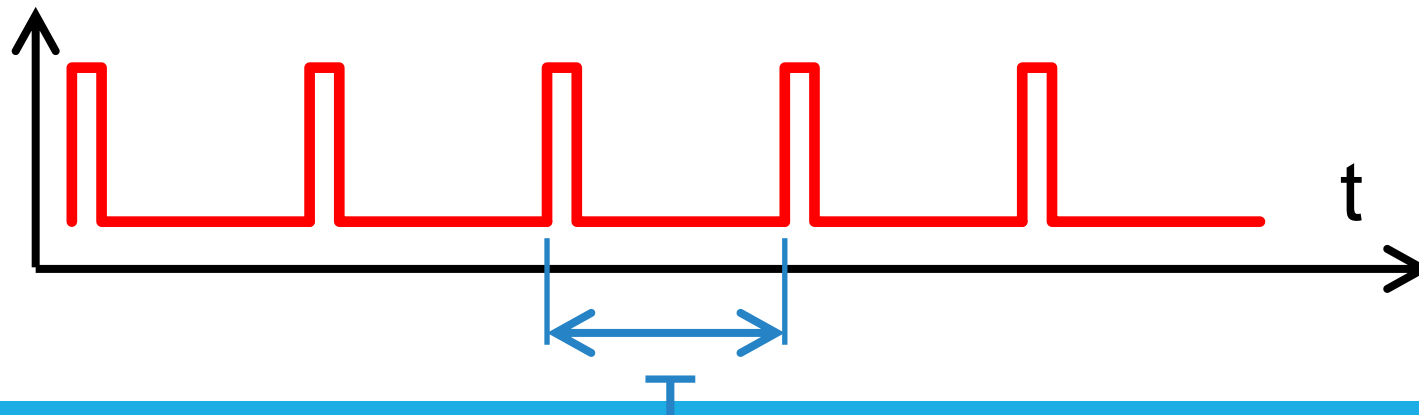
Categoria	Operazioni specifiche
Operazioni aritmetiche (ALU)	+, -, *, /, resto tra numeri interi
Operazioni aritmetiche (FPU)	+, -, *, /, resto tra numeri reali
Operazioni logiche (ALU)	Operazioni su quantita' logiche (unione, intersezione, negazione)
Confronti (ALU)	<, >, =, <=, >=, !=
CPU + Memoria	Trasferimento dati dalla/verso memoria
CPU + unita' di I/O (+ Memoria)	Lettura/scrittura da/su dispositivo

# Un breve riassunto...

- Sistema di elaborazione =  
Unità di I/O + Unità centrale (CPU) + Memoria
- CPU = Unità operativa + Unità di controllo
- Unità operativa:
  - Svolge i calcoli (contiene i circuiti che li eseguono)
  - Contiene alcuni registri  
("parcheggi" per i dati da e verso memoria e I/O)
- Unità di controllo:
  - Governa l'esecuzione delle istruzioni (che legge dalla memoria)
  - Procede secondo tre fasi principali

# Il clock

- Ogni elaboratore contiene un elemento di temporizzazione (detto clock ) che genera un riferimento temporale comune per tutti gli elementi costituenti il sistema di elaborazione.
- $T$  = periodo di clock
  - unità di misura = s
- $f$  = frequenza di clock ( =  $1/T$  )
  - unità di misura =  $s^{-1}$  = Hz (cicli/s)



# Tempistica delle istruzioni

- Un ciclo-macchina è l'intervallo di tempo in cui viene svolta una operazione elementare ed è un multiplo intero del periodo del clock
- L'esecuzione di un'istruzione richiede un numero intero di cicli macchina, variabile a seconda del tipo di istruzione
  - Esempio

Tipo	n. Cicli
ALU	1
FPU	15
MEM	5
I/O	100

- La prestazione complessiva del programma dipenderà dal mix di istruzioni!!!

# La memoria

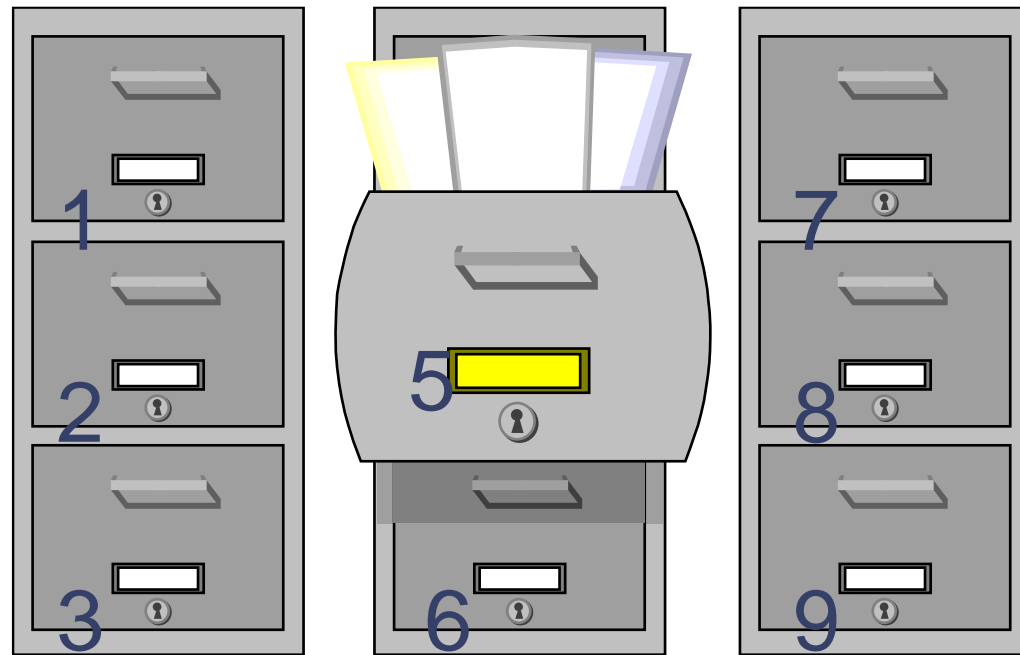
---

# Memoria

- Memorizza i dati e le istruzioni necessarie all'elaboratore per operare
- Perché è necessaria???
  - Per motivi intrinseci di memorizzazione (memoria di massa)
  - Per motivi di efficienza
- Caratteristiche:
  - indirizzamento
  - parallelismo
  - accesso (sequenziale o casuale)
  - Struttura gerarchica

# Indirizzamento

- La memoria è organizzata in celle (mimima unità accessibile direttamente). Ad ogni cella di memoria è associato un indirizzo (numerico) per identificarla univocamente.





# Parallelismo

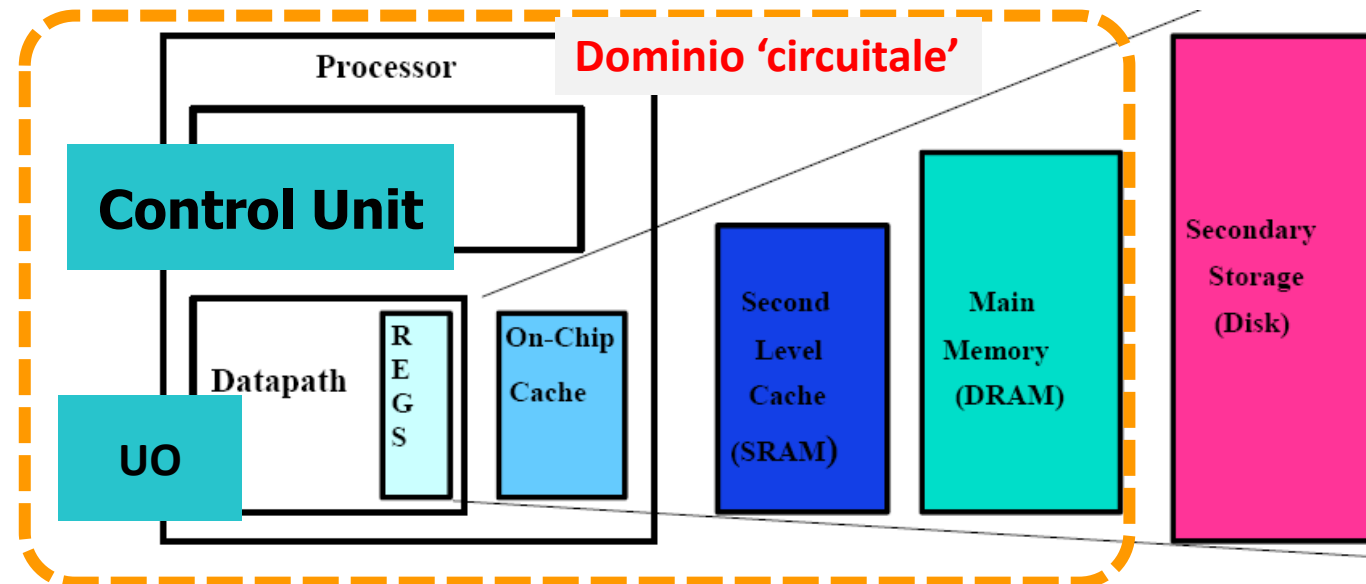
- Ogni cella di memoria contiene una quantità fissa di bit (word):
  - identica per tutte le celle (di una certa unità di memoria)
  - accessibile con un'unica istruzione
  - è un multiplo del byte
    - Tipicamente la dimensione della cella di memoria coincide con quella dei registri per consistenza e semplicità

# Gerarchia di memoria

- Idealmente la memoria dovrebbe essere
  - Più grande possibile
  - Più economica possibile
  - Più veloce possibile
  - Mantenere le informazioni indefinitamente (anche in assenza di alimentazione) – non volatili
- Una simile memoria non esiste (ancora)
  - Le memorie veloci hanno un costo relativamente alto e sono volatili
  - Le memorie non volatili costano relativamente poco ma sono relativamente lente
- Necessaria una gerarchia di memoria

# Gerarchia di memoria

- Per ottimizzare tempo medio di accesso e costo medio, si organizza la memoria a livelli
  - Più vicino al processore le memorie più veloci, costose (e anche volatili, per motivi tecnologici)
  - Più lontano dal processore le memorie più lente, economiche e non volatili



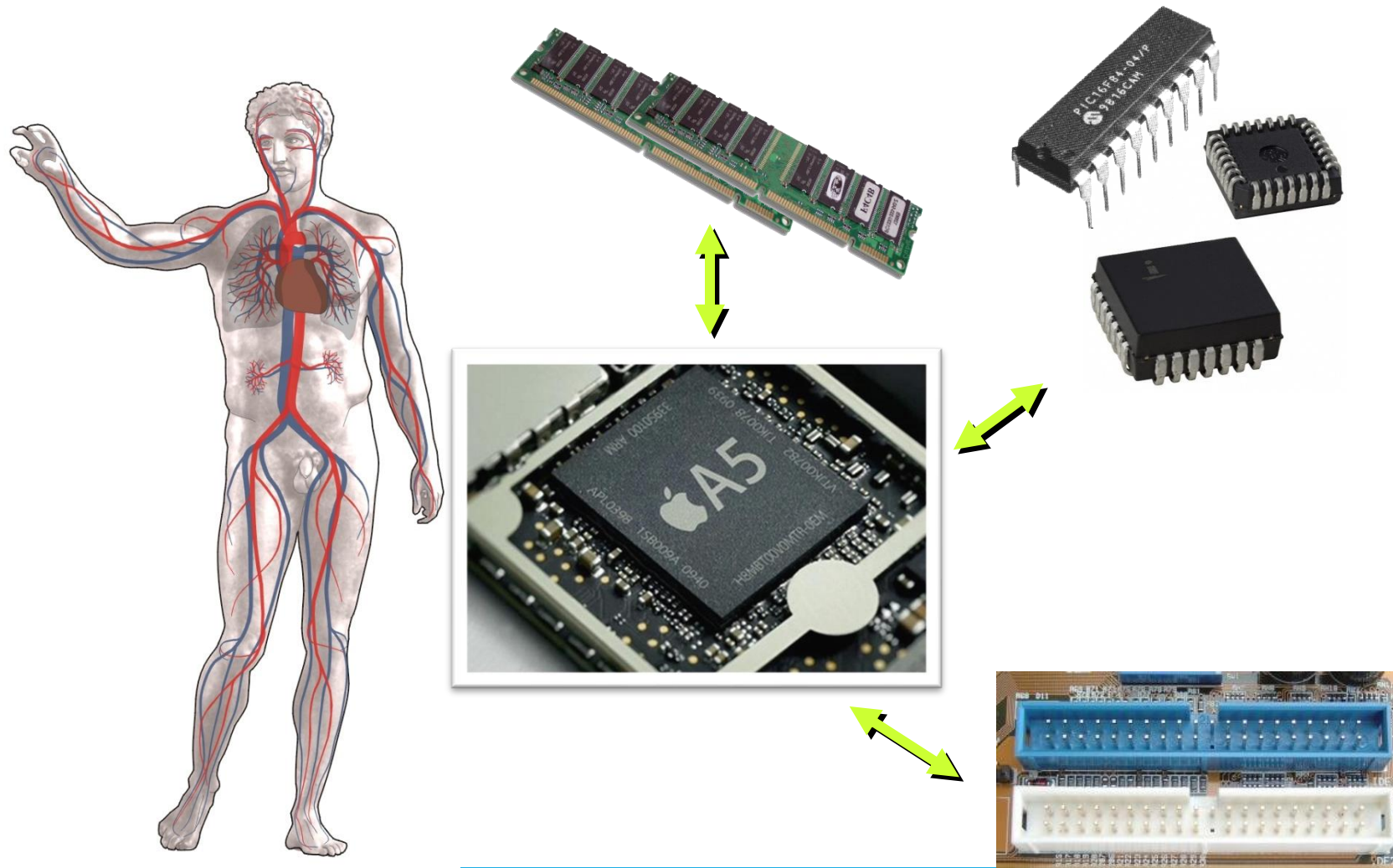
# Confronto tra memorie

Metrica	SRAM (cache)	DRAM	FLASH	Disco
Dimensione	MB (1-16)	GB (4-16)	GB (16-512)	TB (>1)
Velocità	1-5 ns	50-150 ns	~10ms*	~10ms
Costo	10-5/byte	10-8/byte	10-9/byte	10-10 \$/byte
Persistenza	Volatile	Volatile	Non volatile	Non volatile

# Le interconnessioni (bus)

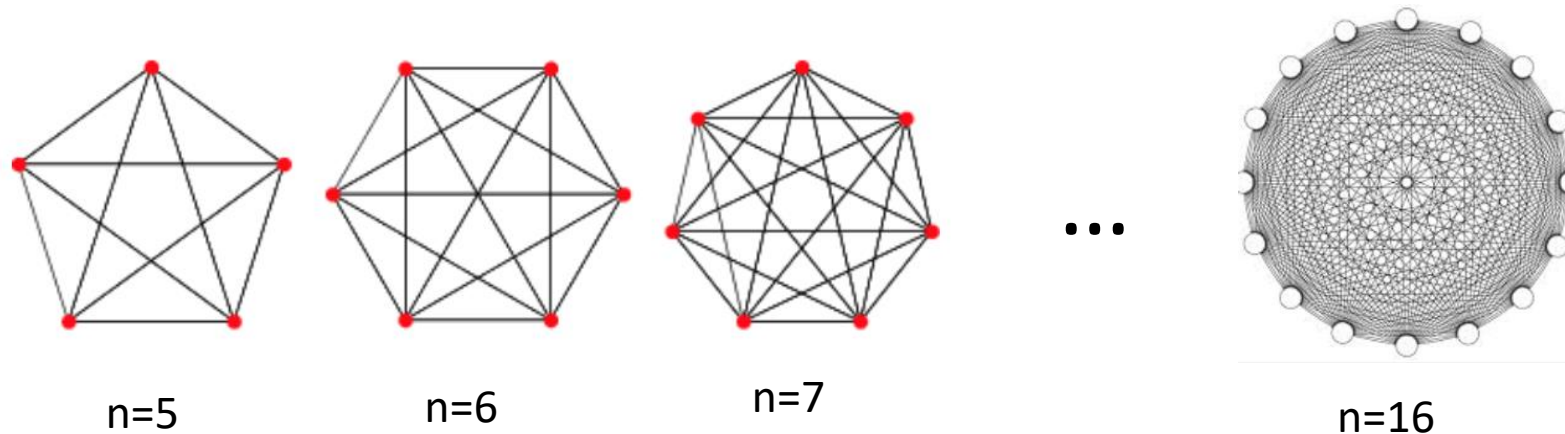
---

# I Bus (sistema circolatorio del PC)



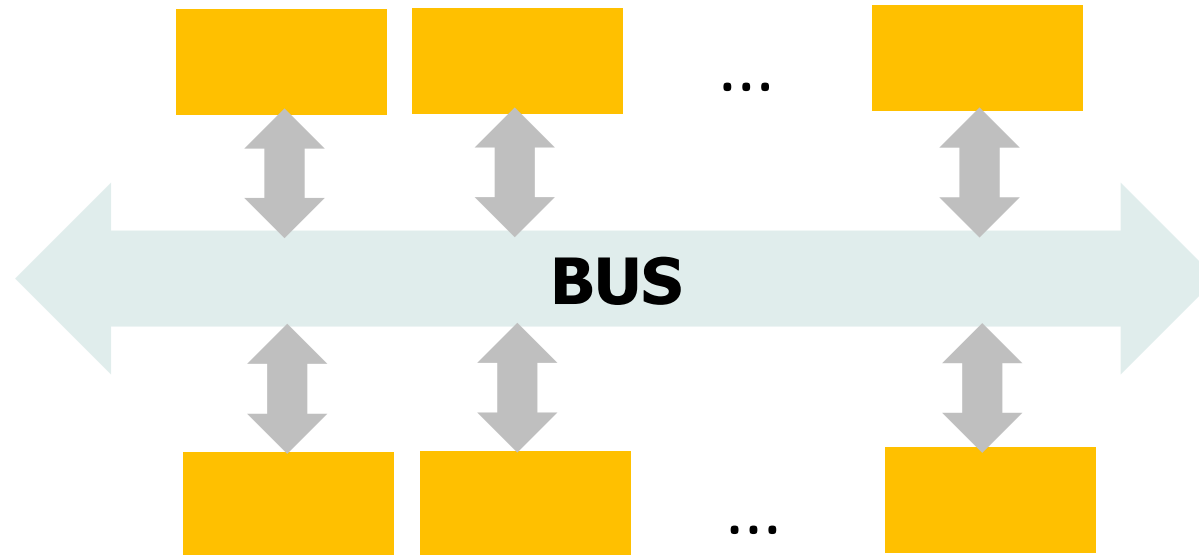
# I bus

- Come connettere  $n$  unità (CPU con memorie e vari controllori di I/O) in modo efficiente?
- La connessione punto-punto non è pratica!
  - Il numero di connessioni cresce con il quadrato del numero di unità da connettere!



# I bus

- Usiamo un'unica linea di connessione per connettere tutti i componenti
  - Fisicamente “agganciati” a questa linea





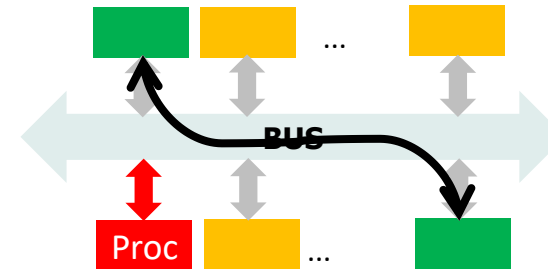
# I bus

## ■ Vantaggi:

- costi ridotti di produzione
- Estendibilità (scalabilità)
  - aggiunta di nuovi dispositivi molto semplice
- Standardizzabilità
  - regole per la comunicazione da parte di dispositivi diversi

## ■ Svantaggi:

- Lentezza
  - utilizzo in mutua esclusione del bus
- Sovraccarico del processore (CPU)
  - funge da “master” sul controllo del bus

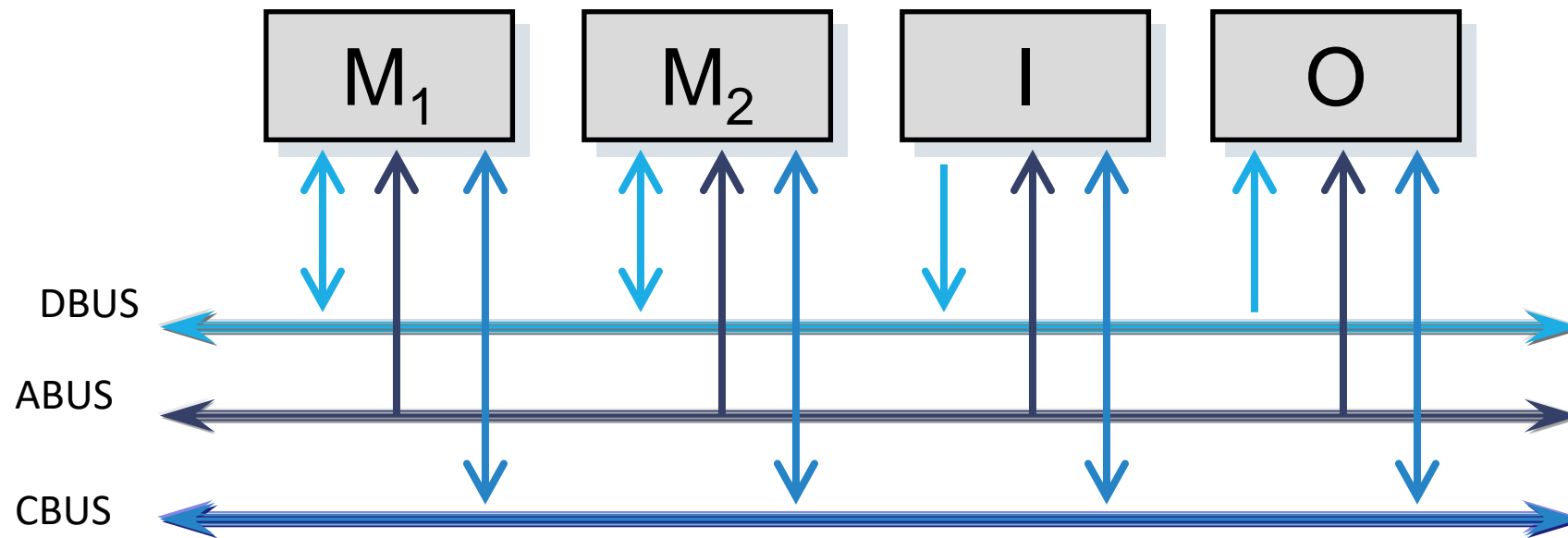


# Caratteristiche di un bus

- Trasporto di un solo dato per volta
- Frequenza = n. di dati trasportati al secondo
- Ampiezza = n. di bit di cui è costituito un singolo dato
- Se mal dimensionato, potrebbe essere un collo di bottiglia

# Tipi fondamentali di bus

- Un singolo bus è suddiviso in tre “sotto bus”, detti:
  - bus dati (DBus)
  - bus degli indirizzi (ABus)
  - bus di controllo (CBus)



# Massima memoria interna (fisicamente presente)

- La dimensione dell'Abus determina il max numero di celle di memoria indirizzabili
- La dimensione del Dbus “indica” la dimensione di una cella di memoria
- $\text{max mem} = 2^{|Abus|} \times |Dbus| \text{ bit}$
- Esempio (Abus da 20 bit, Dbus da 16 bit):
  - $\text{max mem} = 2^{20} \times 2 \text{ byte} = 2 \text{ MB}$
  - ossia 1 M celle di memoria, ognuna da 2 byte

# Una vista di insieme

