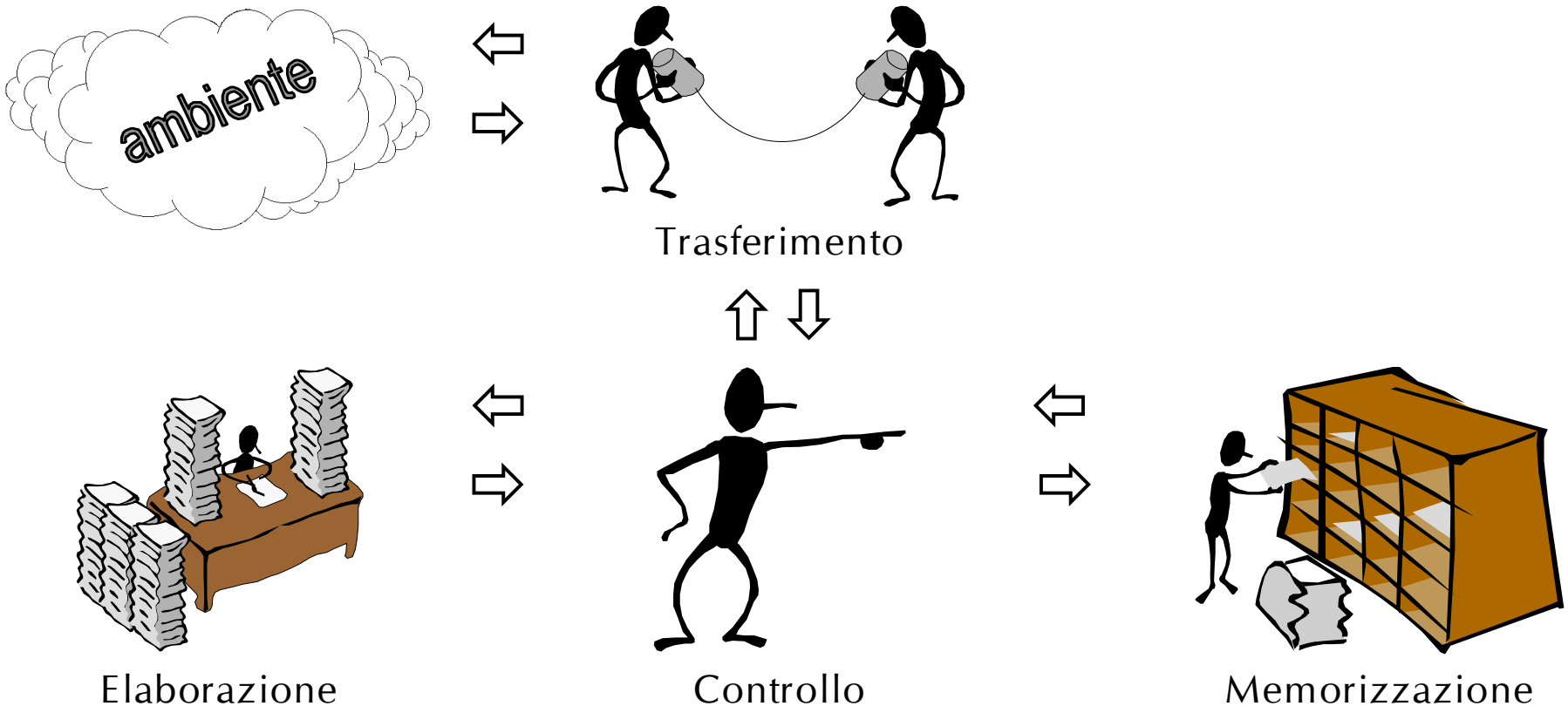
Le infrastrutture hardware

Il processore
La memoria centrale
La memoria di massa
Le periferiche di I/O

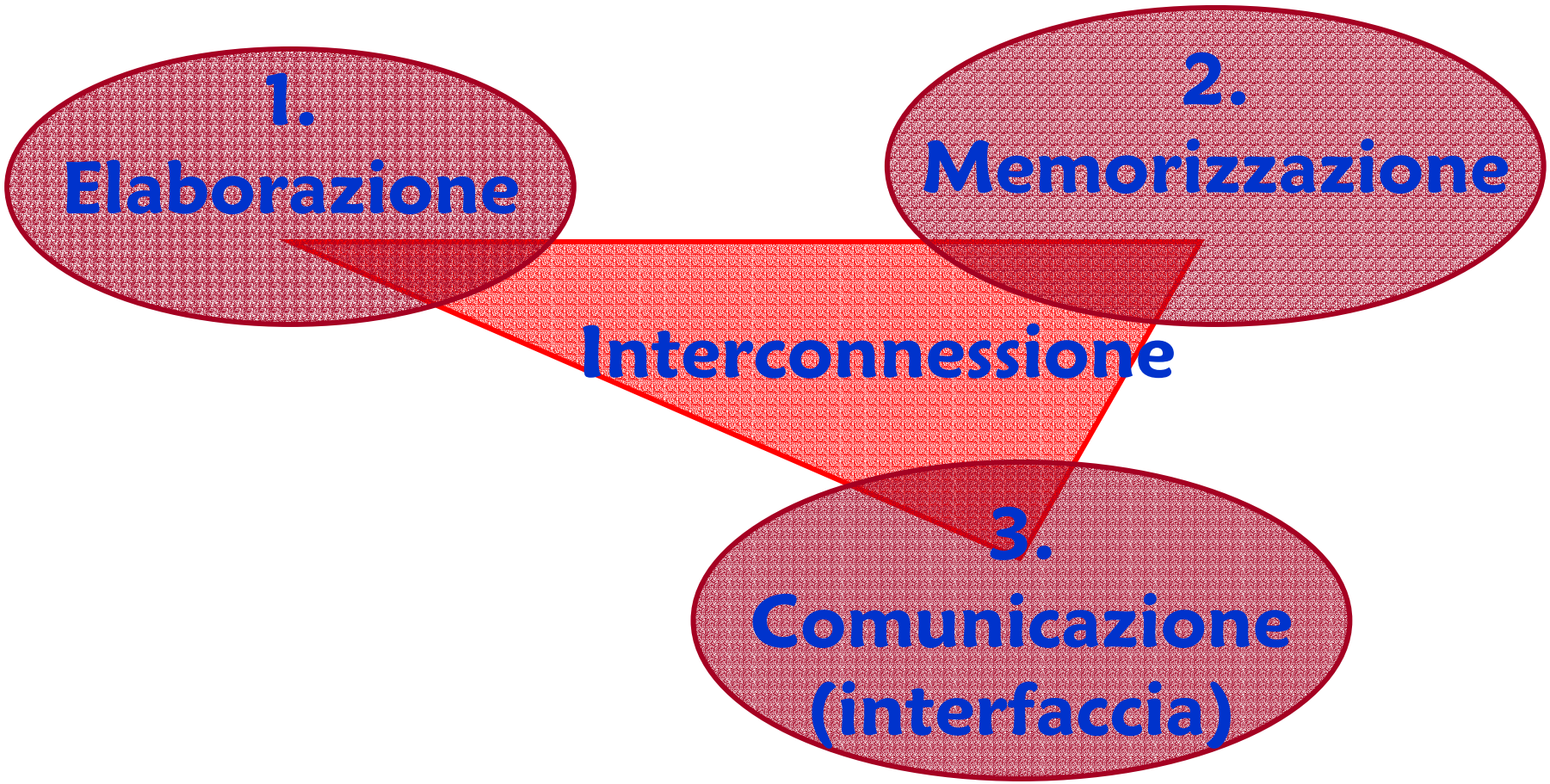
Funzionalità di un calcolatore



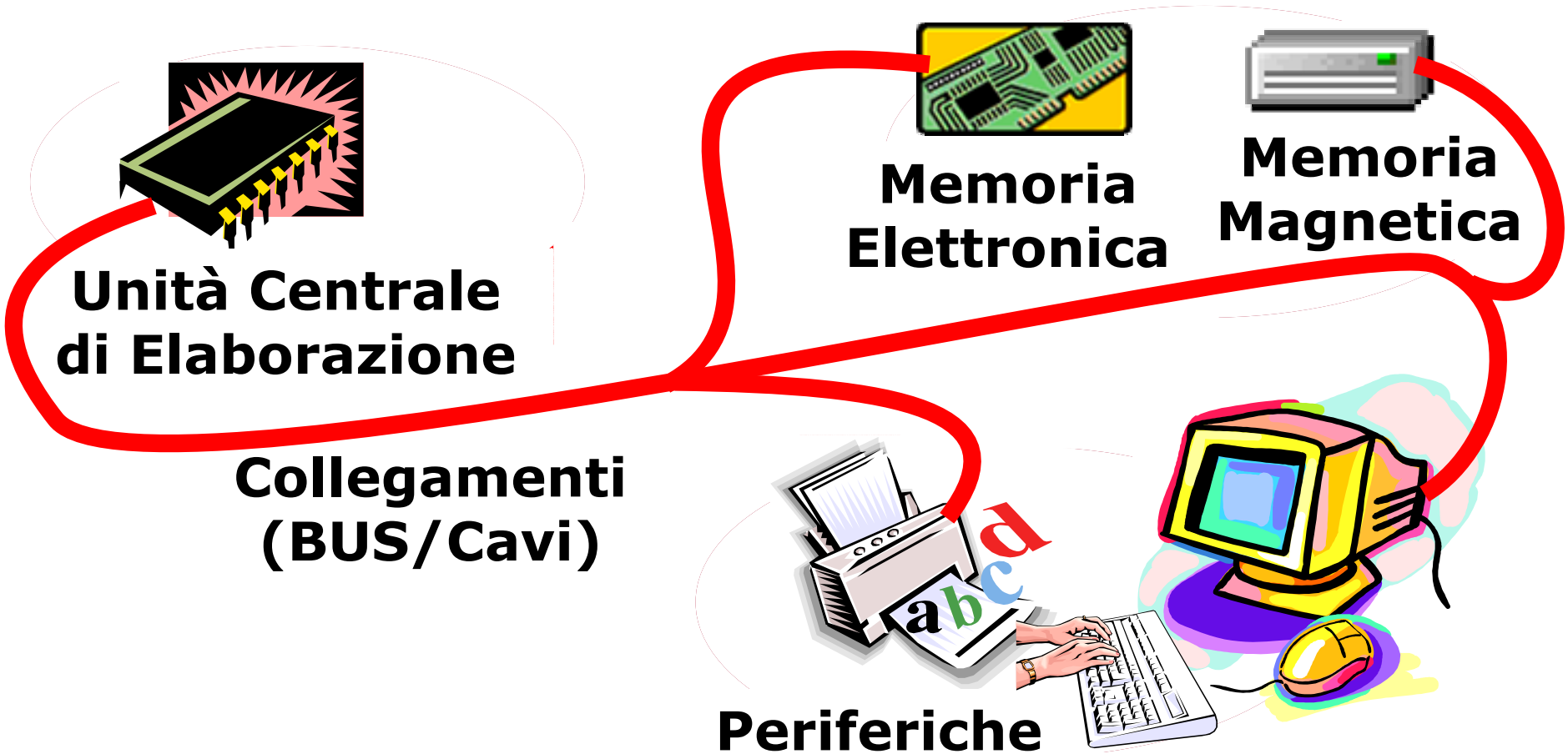
Caratteristiche dell'architettura

- **Flessibilità**
  - adatta a svolgere diverse tipologie di compiti
- **Modularità**
  - ogni componente ha una funzione specifica
- **Scalabilità**
  - ogni componente può essere sostituito con uno equivalente
- **Standardizzazione**
  - componenti facilmente sostituibili in caso di malfunzionamento
- **Riduzione dei costi**
  - grazie alla produzione su larga scala
- **Semplicità**
  - di installazione ed esercizio del sistema

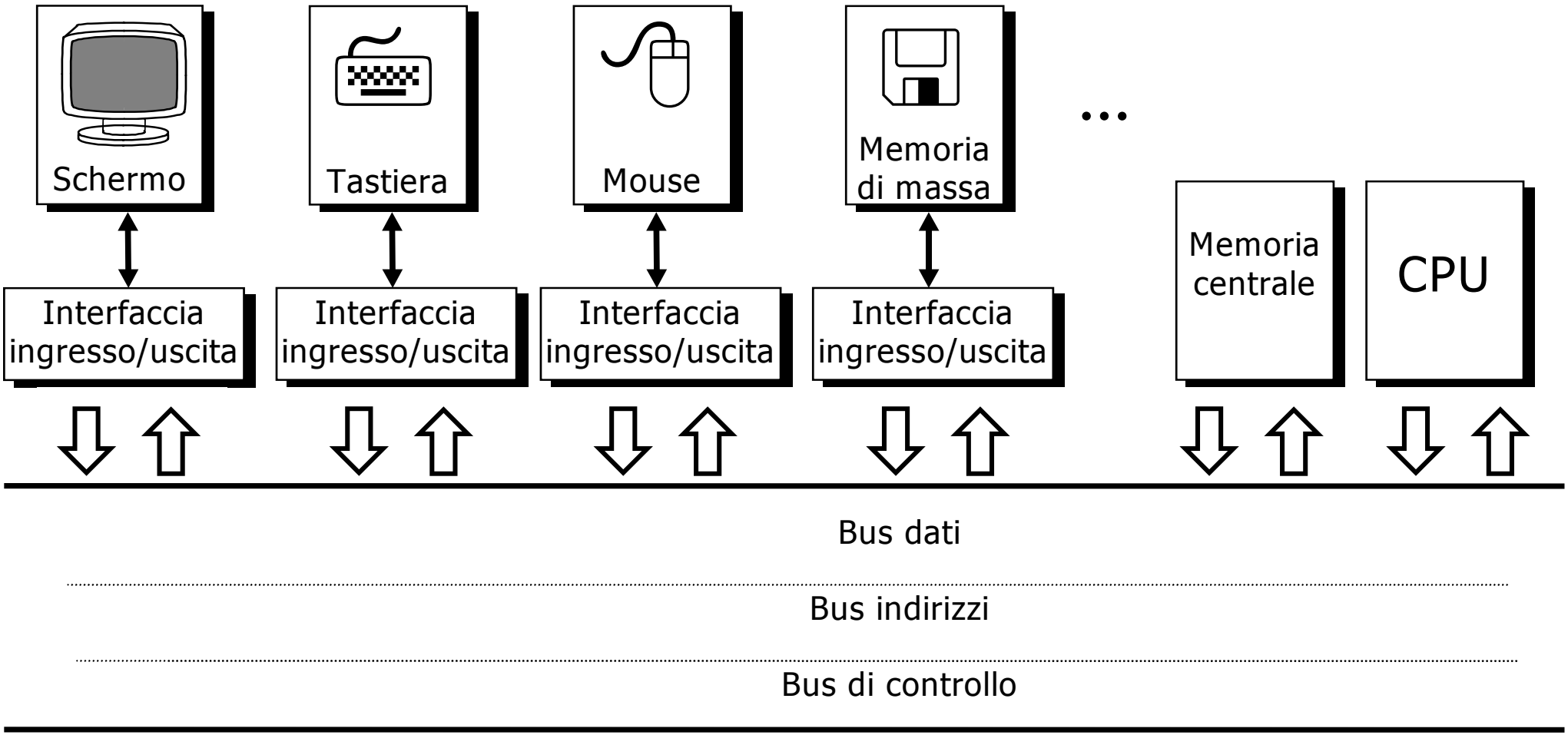
Il calcolatore: modello concettuale



Il calcolatore: modello architetturale



Lo schema di riferimento

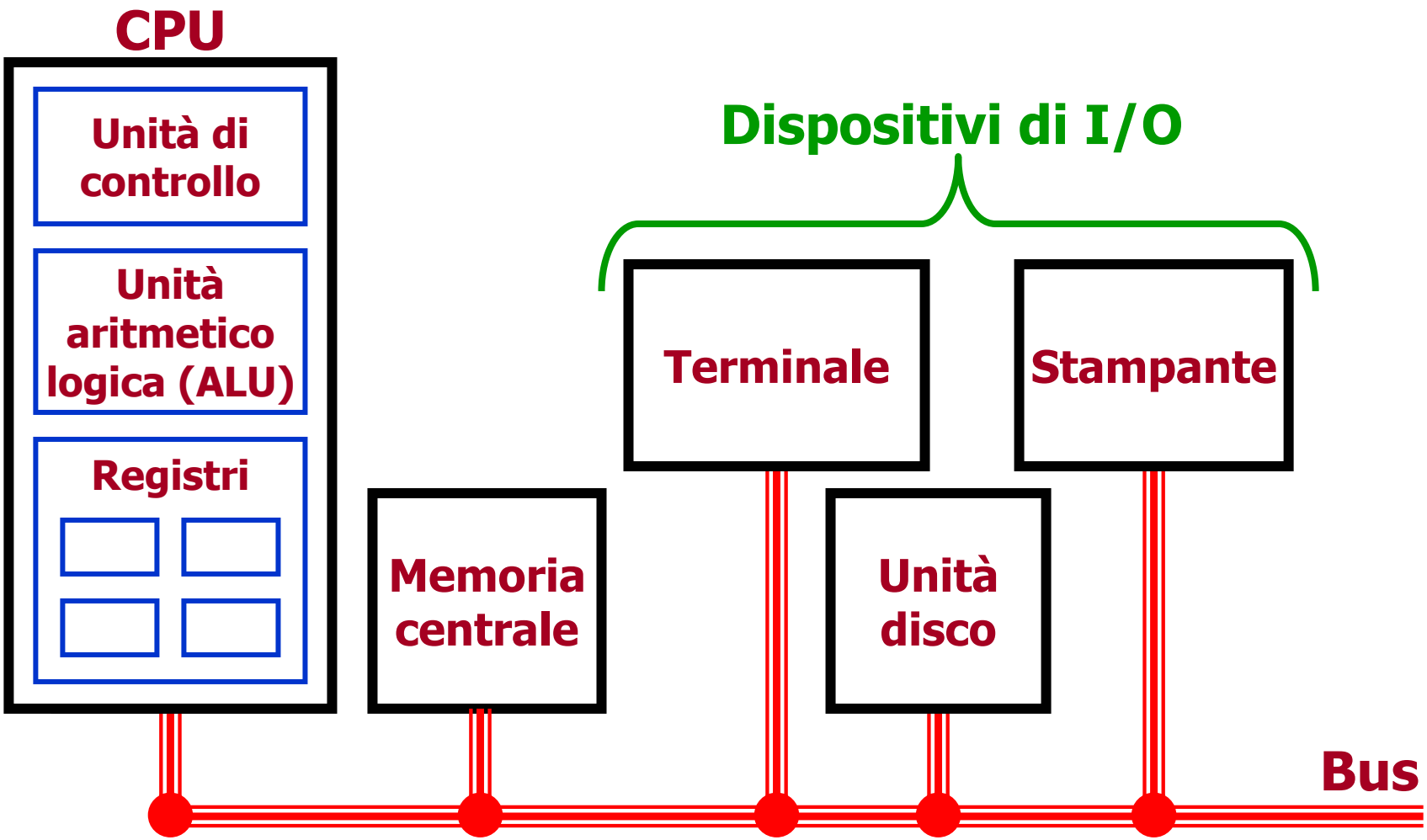


Caratteristiche del collegamento a BUS

- Semplicità
  - un'unica linea di connessione → costi ridotti di produzione
- Estendibilità
  - aggiunta di nuovi dispositivi molto semplice
- Standardizzabilità
  - regole per la comunicazione da parte di dispositivi diversi
- Lentezza
  - utilizzo in mutua esclusione del bus
- Limitatà capacità
  - al crescere del numero di dispositivi collegati
- Sovraccarico del processore (CPU)
  - perchè funge da *master* sul controllo del bus

Unità centrale di elaborazione
CPU

Organizzazione tipica di un calcolatore "bus oriented"



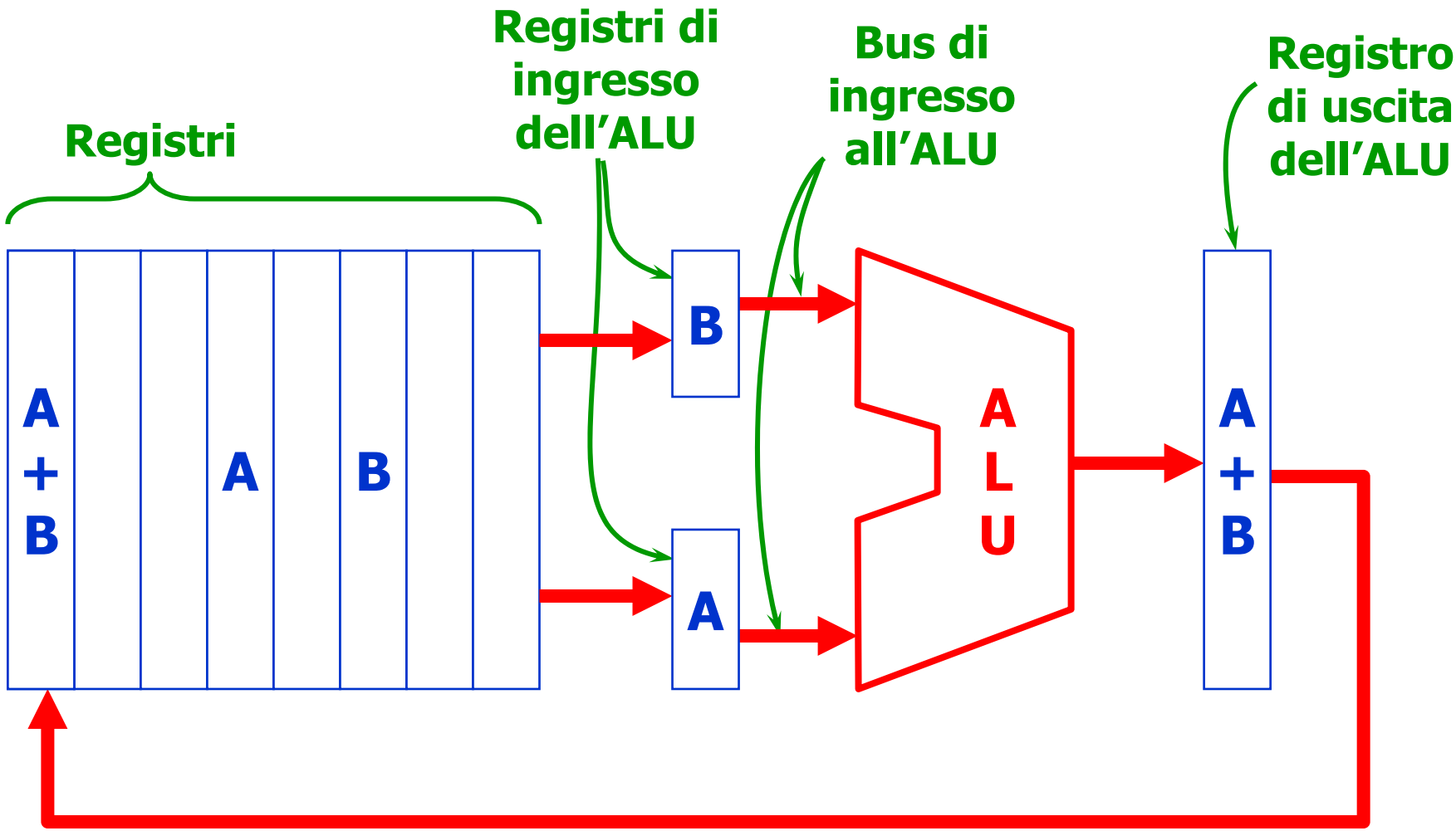
Tre tipologie di istruzioni

- Istruzioni aritmetico-logiche (Elaborazione dati)
  - Somma, Sottrazione, Divisione, …
  - And, Or, Xor, …
  - Maggiore, Minore, Uguale, Minore o uguale, …
- Controllo del flusso delle istruzioni
  - Sequenza
  - Selezione semplice, a due vie, a n vie, …
  - Ciclo a condizione iniziale, ciclo a condizione finale, …
- Trasferimento di informazione
  - Trasferimento dati e istruzioni tra CPU e memoria
  - Trasferimento dati e istruzioni tra CPU e dispositivi di ingresso/uscita (attraverso le relative interfacce)

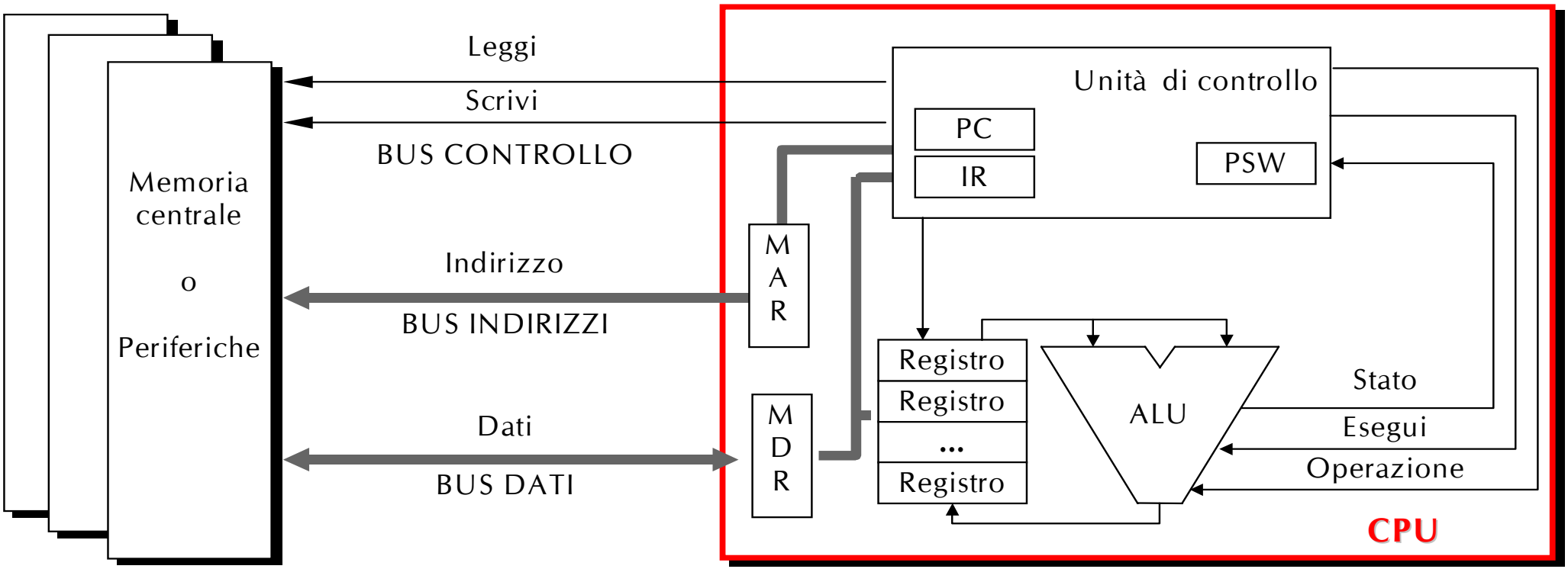
Elementi di una CPU

- **Unità di controllo**
  - legge le istruzioni dalla memoria e ne determina il tipo.
- **Unità aritmetico–logica**
  - esegue le operazioni necessarie per eseguire le istruzioni.
- **Registri**
  - **memoria ad alta velocità** usata per risultati temporanei e informazioni di controllo;
  - il **valore massimo** memorizzabile in un registro è determinato dalle **dimensioni** del registro;
  - esistono registri di uso generico e registri specifici:
    - **Program Counter** (**PC**) – qual è l'istruzione successiva;
    - **Instruction Register** (**IR**) – istruzione in corso d'esecuzione;
    - **...**

Struttura del "data path"



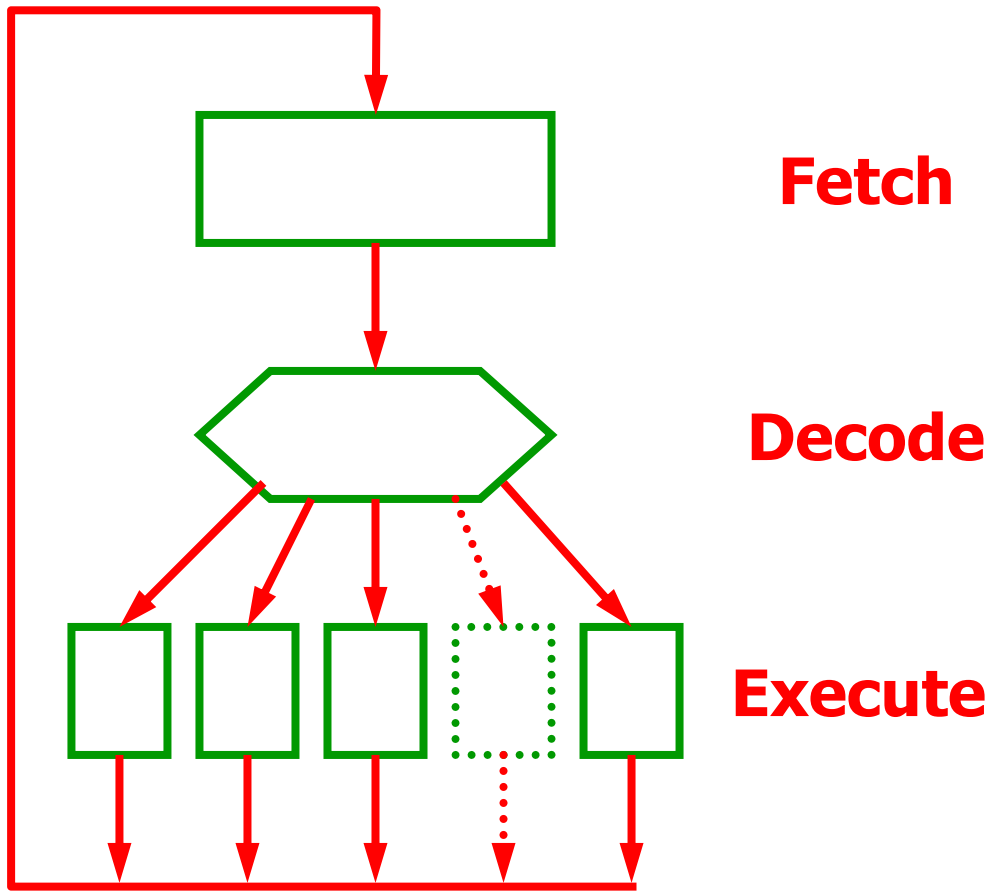
La struttura della CPU



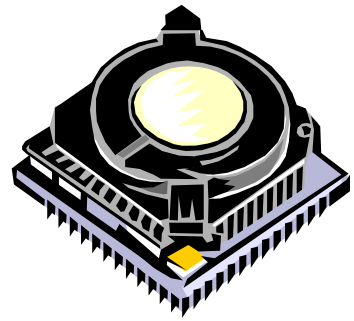
Esecuzione delle istruzioni

- Ciclo **Fetch–Decode–Execute** (**leggi–decodifica–esegui**)
  1. Prendi l'**istruzione corrente** dalla memoria e mettila nel **registro istruzioni** (**IR**).
  2. **Incrementa** il **program counter** (**PC**) in modo che contenga l'indirizzo dell'istruzione successiva.
  3. Determina il tipo dell'istruzione corrente (**decodifica**).
  4. Se l'istruzione usa una parola in memoria, determina dove si trova.
  5. Carica la parola, se necessario, in un registro della CPU.
  6. **Esegui** l'istruzione.
  7. Torna al punto 1 e inizia a eseguire l'istruzione successiva.

Ciclo Fetch–Decode–Execute



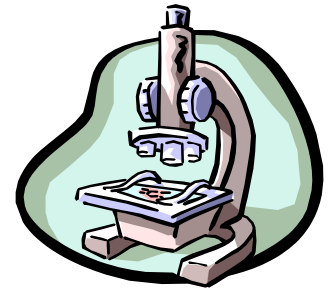
CPU



- In grado di eseguire solo istruzioni codificate in **linguaggio macchina**
- Ciclo Fetch – Decode - Execute

1. Prendi l'istruzione corrente dalla memoria e mettila nel registro istruzioni (IR) (***fetch***)
2. Incrementa il Program Counter (PC) in modo che contenga l'indirizzo dell'istruzione successiva
3. Determina il tipo di istruzione da eseguire (***decode***)
4. Se l'istruzione necessita di un dato in memoria determina dove si trova e caricalo in un registro della CPU
5. Esegui l'istruzione (***execute***)
6. Torna al punto 1 e opera sull'istruzione successiva

Evoluzione delle CPU



CPU	Anno	Frequenza (MHz)	Dimensione registri / bus dati	Numero di transistor
8086	1978	4.77 — 12	8 / 16	29 000
80286	1982	8 — 16	16 / 16	134 000
80386	1986	16 — 33	32 / 32	275 000
80386 SX	1988	16 — 33	32 / 16	275 000
80486	1989	33 — 50	32 / 32	1 200 000
Pentium	1993	60 — 200	32 / 64	3 100 000
Pentium II	1997	233 — 400	32 / 64	7 500 000
Pentium III	1999	450 — 1133	32 / 64	24 000 000
Pentium 4	2000	1600 — 2000	32 / 64	42 000 000

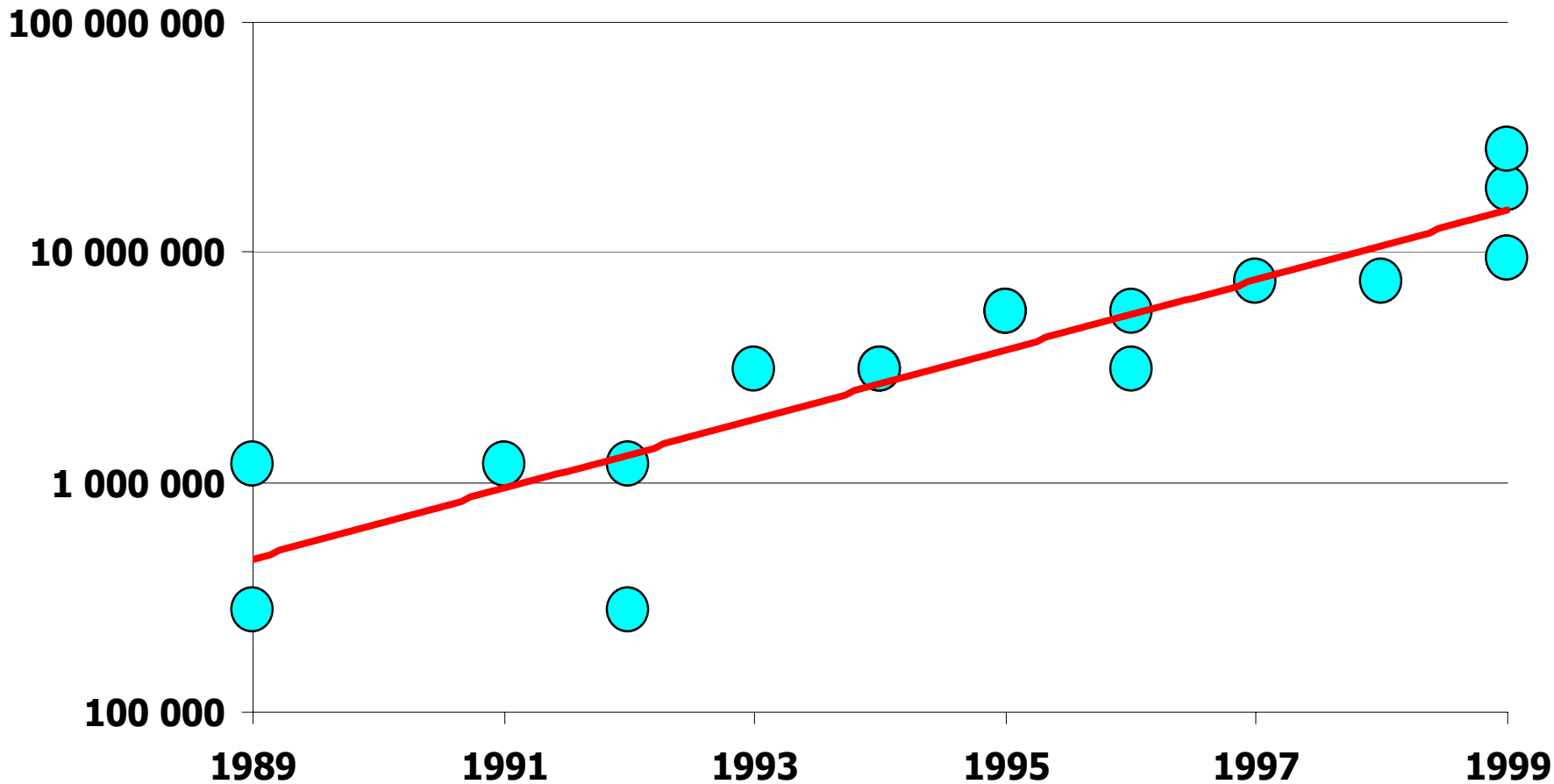
Legge di Moore

Osservazione fatta da Gordon Moore nel 1965:

il numero dei transistor per cm^2 raddoppia ogni X mesi

In origine X era 12. Correzioni successive hanno portato a fissare **X=18**. Questo vuol dire che c'è un incremento di circa **il 60% all'anno**.

Transistor [CPU Intel]



Legge di Moore e progresso

- Il progresso della tecnologia provoca un **aumento del numero di transistor** per cm^2 e quindi per chip.
- Un maggior numero di transistor per chip permette di produrre **prodotti migliori** (sia in termini di prestazioni che di funzionalità) **a prezzi ridotti**.
- I prezzi bassi stimolano la nascita di **nuove applicazioni** (e.g. non si fanno video game per computer da milioni di $).
- Nuove applicazioni aprono **nuovi mercati** e fanno nascere **nuove aziende**.
- L'esistenza di tante aziende fa **crescere la competitività** che, a sua volta, stimola il **progresso della tecnologia** e lo sviluppo di **nuove tecnologie**.

Approfondimento: incrementare le prestazioni con il parallelismo

Parallelismo

- La **frequenza di clock**
  - influenza direttamente il tempo di ciclo del data path e quindi le prestazioni di un calcolatore;
  - è limitata dalla tecnologia disponibile.
- Il **parallelismo** permette di migliorare le prestazioni senza modificare la frequenza di clock. Esistono due forme di parallelismo:
  - **parallelismo a livello delle istruzioni** (architetture **pipeline** o architetture **superscalari**);
  - **parallelismo a livello di processori** (**Array computer**, **multiprocessori** o **multicomputer**).

Architettura pipeline

- Organizzazione della CPU come una "**catena di montaggio**"
  - la CPU viene suddivisa in "**stadi**", ognuno dedicato all'esecuzione di un compito specifico;
  - l'esecuzione di un'istruzione richiede il **passaggio attraverso** (tutti o quasi tutti) **gli stadi della pipeline**;
  - in un determinato istante, **ogni stadio esegue la parte di sua competenza di una istruzione**;
  - in un determinato istante, esistono **diverse istruzioni contemporaneamente in esecuzione**, una per ogni stadio.

Esempio di pipeline /1

Pipeline in **cinque stadi**:

S1. **lettura istruzioni** dalla memoria e loro caricamento in un apposito buffer;

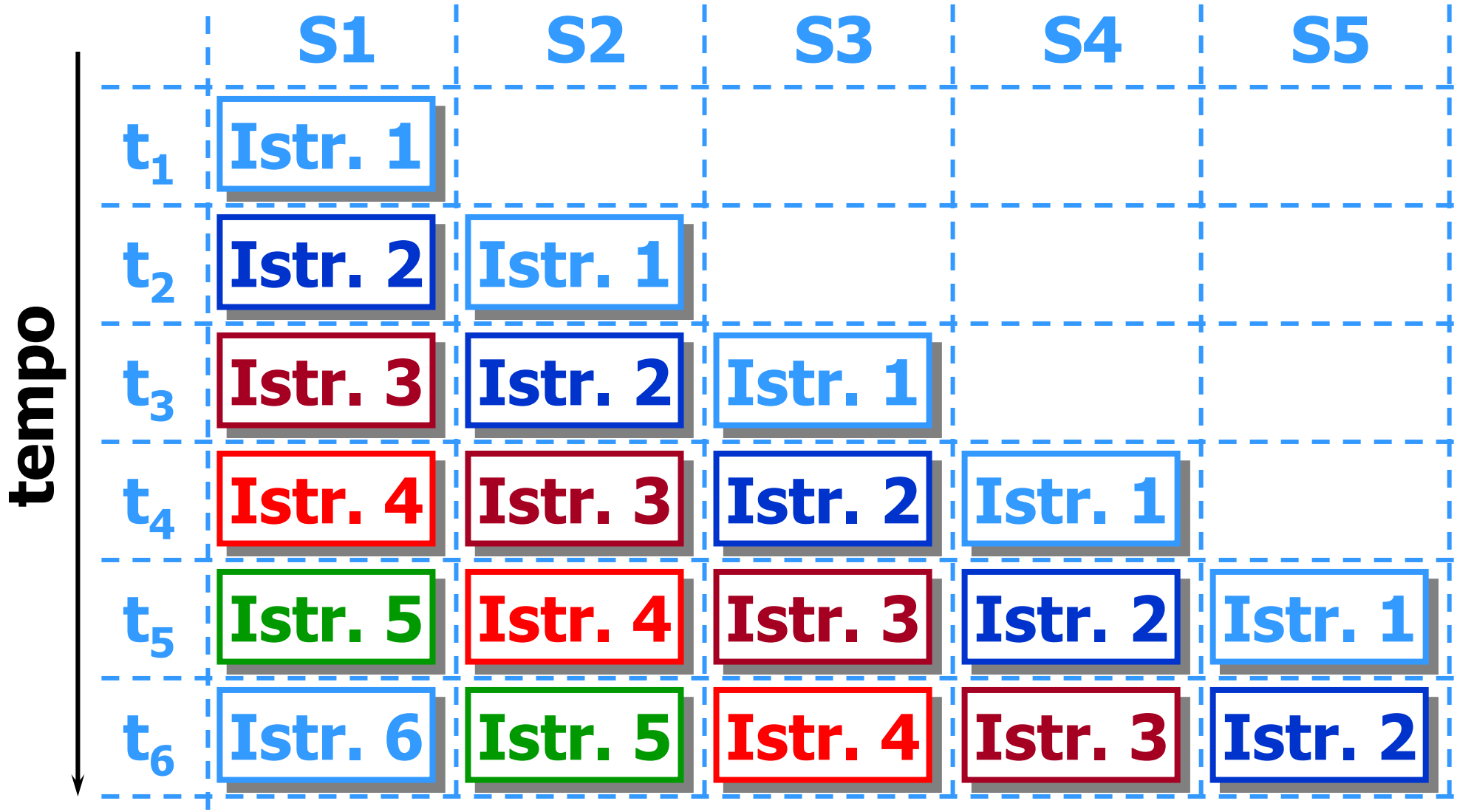
S2. **decodifica** dell'istruzione per determinarne il tipo e gli operandi richiesti;

S3. individuare e **recuperare gli operandi** dai registri o dalla memoria;

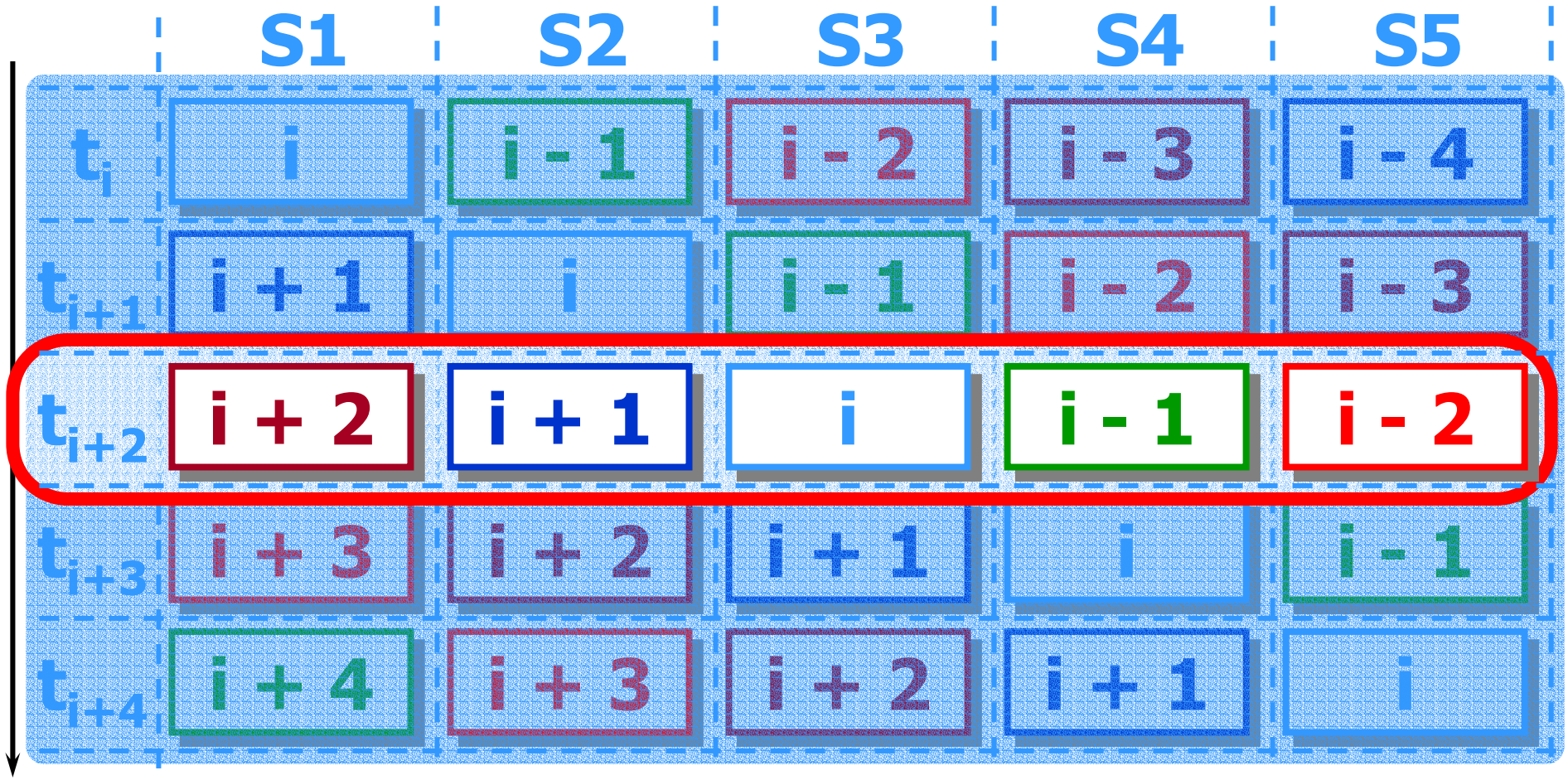
S4. **esecuzione** dell'istruzione, tipicamente facendo passare gli operandi per il data path;

S5. **invio dei risultati** al registro appropriato.

Esempio di pipeline /2

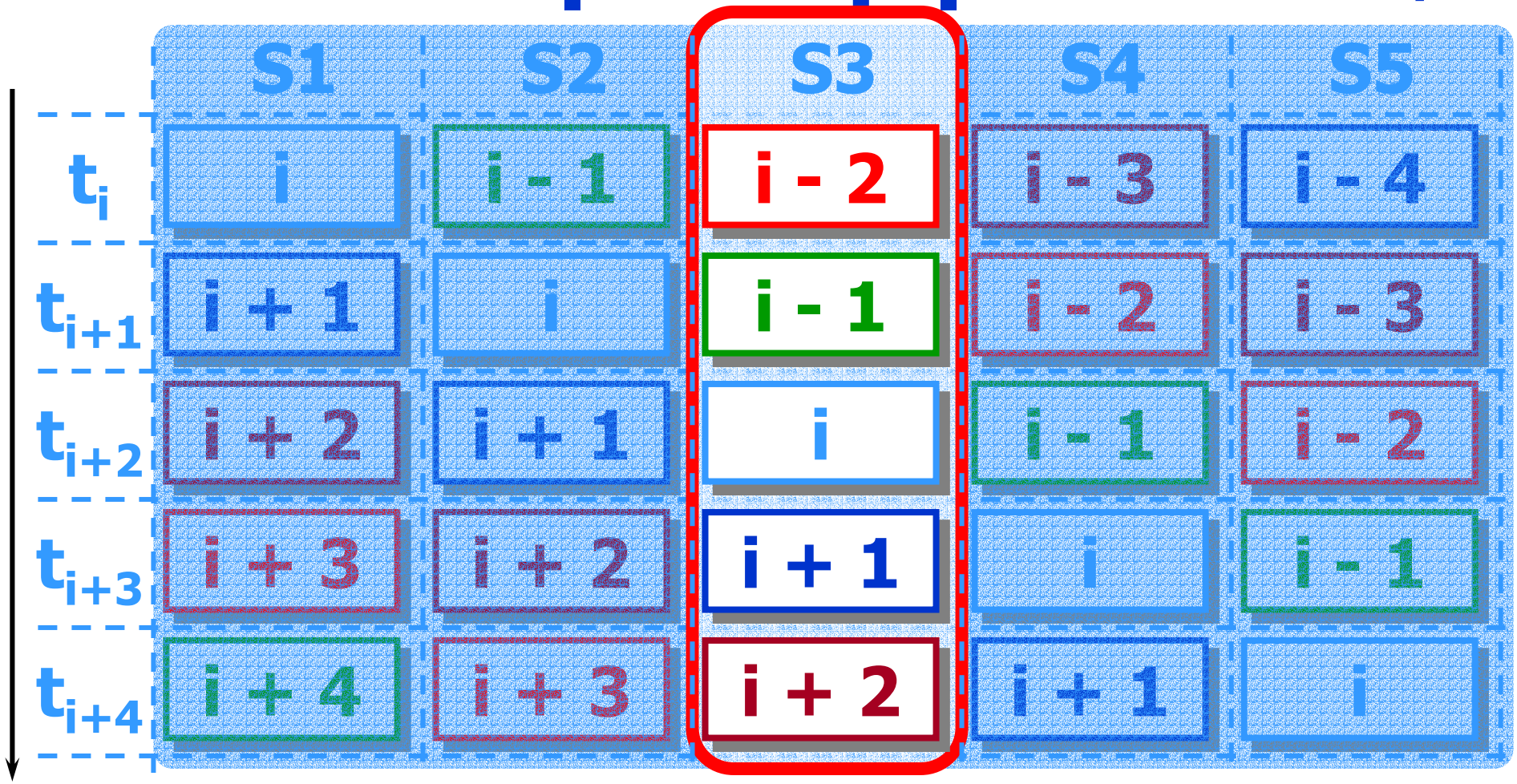


Esempio di pipeline /3



All'istante t_{i+2} ci sono 5 istruzioni in esecuzione

Esempio di pipeline /4



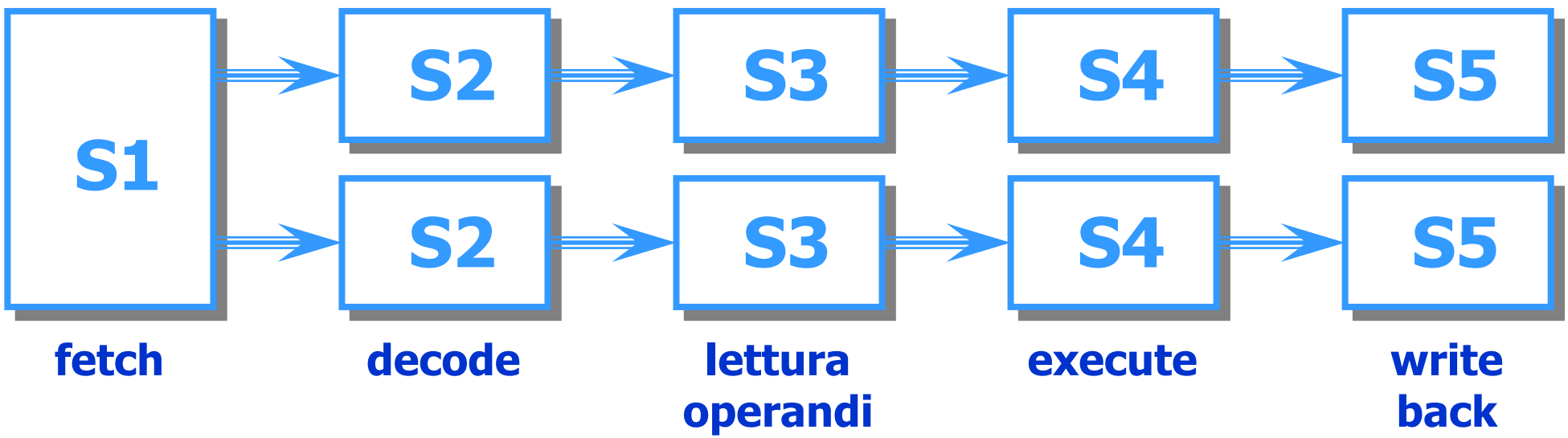
Lo stadio S3 esegue la parte di sua competenza di istruzioni successive l'una all'altra.

Prestazioni di una pipeline

- Il tempo di esecuzione (**latenza**) della singola istruzione non diminuisce, anzi **aumenta**
  - il tempo di attraversamento (latenza) della pipeline corrisponde al numero degli stadi (**N**) moltiplicato per il tempo di ciclo (**T**);
  - il tempo di ciclo è limitato dallo stadio più lento!
- **Aumenta** il numero di istruzioni completate nell'unità di tempo (**throughput**)
  - si completa **un'istruzione a ogni ciclo di clock**;
  - l'**incremento** di throughput è quasi **proporzionale al numero degli stadi!**

Architetture superscalari

- Vista la disponibilità di un maggior numero di transistor si inseriscono **più pipeline** nella stessa CPU
  - aumenta il parallelismo perché è possibile eseguire contemporaneamente diversi **flussi di istruzioni**;
  - è necessario garantire che non ci siano **conflitti** tra le istruzioni che vengono eseguite insieme; di solito il controllo è affidato al compilatore.



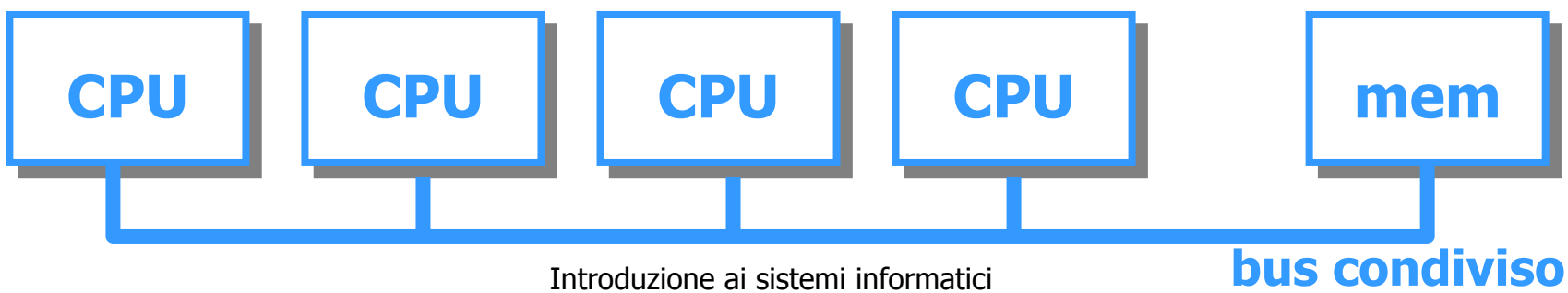
Architetture superscalari

In alternativa si possono **replicare le unità funzionali**

- rappresentano lo stadio più lento della pipeline (in genere richiedono diversi cicli di clock);
- è più semplice evitare i **conflitti** tra le diverse istruzioni.

Multiprocessori

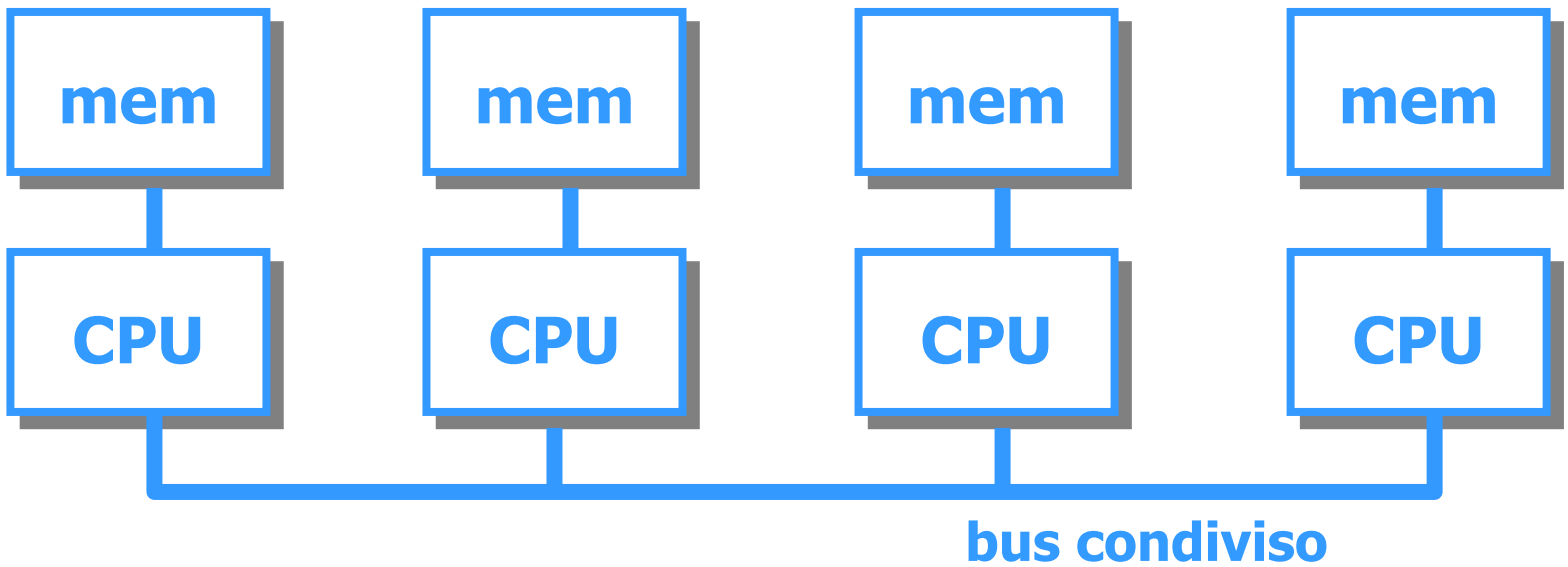
- **Diverse CPU** condividono una **memoria comune**:
  - le CPU debbono **coordinarsi** per accedere alla memoria;
  - esistono diversi schemi di collegamento tra CPU e memoria, quello più semplice prevede che ci sia un **bus condiviso**;
    - se i processori sono veloci il **bus** diventa un **collo di bottiglia**;
    - esistono soluzioni che permettono di migliorarne le prestazioni, ma si adattano a sistemi con un **numero limitato di CPU** (<20).
- La **memoria condivisa** rende più semplice il **modello di programmazione**:
  - si deve **parallelizzare l'algoritmo**, ma si può trascurare la **"parallelizzazione" dei dati**.



Multicalcolatori

- Sistemi composti da tanti calcolatori collegati fra loro
  - ogni calcolatore è dotato di una **memoria privata** e non c'è **memoria in comune**;
  - **comunicazione** tra CPU basata su **scambio di messaggi**.
- Non è efficiente collegare ogni calcolatore a tutti gli altri, quindi vengono usate topologie particolari:
  - **griglie** a 2/3 dimensioni, **alberi** e **anelli**;
  - i messaggi, per andare da fonte a destinazione, spesso devono passare da uno o più calcolatori intermedi o **switch.**
  - **Tempi di trasferimento** dei messaggi dell'ordine di alcuni **microsecondi** sono comunque facilmente ottenibili.
- Sono stati costruiti multicalcolatori con ~10.000 CPU.

Struttura di un multicomputer



La memoria

La memoria

- **Supporto alla CPU**: deve fornire alla CPU dati e istruzioni il più rapidamente possibile
- **Archivio**: deve consentire di archiviare dati e programmi garantendone la conservazione e la reperibilità anche dopo elevati periodi di tempo
- Diverse esigenze:
  - **velocità** per il supporto alla CPU
  - **non volatilità** ed **elevate dimensioni** per l'archivio
- Diverse tecnologie
  - **elettronica**: veloce, ma costosa e volatile
  - **magnetica** e **ottica**: non volatile ed economica, ma molto lenta

Criteri di caratterizzazione di una memoria

- Velocità
  - tempo di accesso (quanto passa tra una richiesta e la relativa risposta)
  - velocità di trasferimento (quanti byte al secondo si possono trasferire)
- Volatilità
  - cosa succede quando la memoria non è alimentata?
  - per quanto tempo i dati vi rimangono immagazzinati?
- Capacità
  - quanti byte può contenere? qual è la dimensione massima?
- Costo (per bit)
- Modalità di accesso
  - diretta (o casuale): il tempo di accesso è indipendente dalla posizione
  - sequenziale: il tempo di accesso dipende dalla posizione
  - mista: combinazione dei due casi precedenti
  - associativa: indicato il dato, la memoria risponde indicando l'eventuale posizione che il dato occupa in memoria.