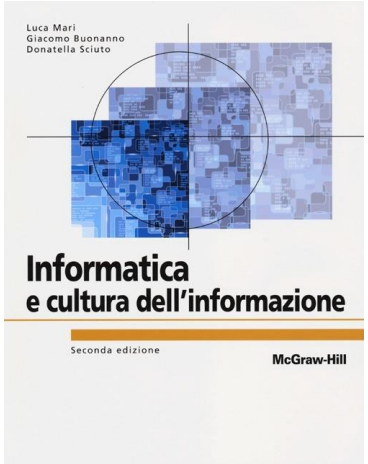
Programma del corso



L. Mari, G. Buonanno, D. Sciuto
"Informatica e cultura dell'informazione"
(seconda edizione)
McGraw-Hill ©, 2013

- Programma di massima (50 ore)
  - Introduzione alla scienza e alla tecnologia dell'informazione
  - **Modello e architettura dei calcolatori**
  - Tecniche di codifica dell'informazione
  - Il sistema operativo
  - Reti di calcolatori e sicurezza della comunicazione

Obiettivi di apprendimento

- Modello concettuale, funzionale e architetturale di un calcolatore
  - Funzione di CPU, memoria, periferiche, bus
- Struttura e principi di funzionamento della CPU e della memoria centrale

Struttura di base dei calcolatori

- I primi calcolatori di concezione analoga a quella attuale furono sviluppati sfruttando dispositivi elettromeccanici e poi valvole termoioniche
- Il passaggio ai transistor e la sempre maggiore miniaturizzazione dei componenti ha permesso un incremento rapidissimo delle capacità di elaborazione

- La struttura di base dei calcolatori non è cambiata
  - Operazioni semplici eseguite ad alta velocità
  - Possibilità di essere programmati per eseguire più istruzioni in successione
  - E' l'elevata velocità a rendere utile il calcolatore

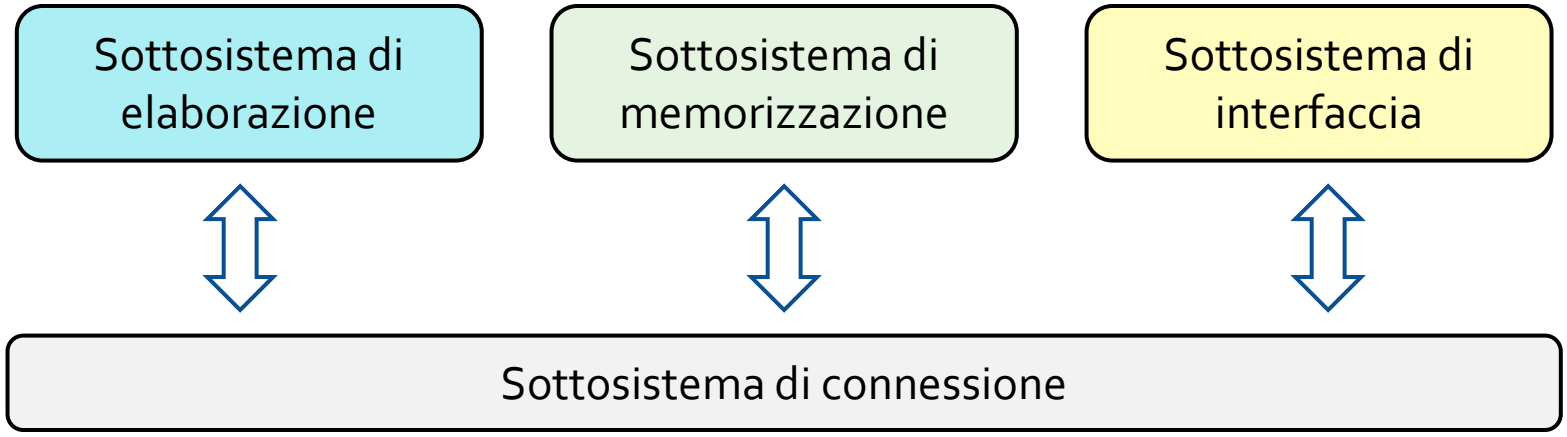
Caratteristiche generali

- Flessibilità
  - Architettura non specializzata per un solo tipo di utilizzo ma adatta a svolgere diversi compiti
- Modularità e scalabilità
  - A ogni componente viene demandato lo svolgimento di una funzione specifica del sistema complessivo
  - La standardizzazione delle interfacce fra i componenti (il modo in cui i componenti sono collegati e il modo in cui comunicano) facilita la sostituzione in caso di malfunzionamenti
  - Un componente può essere sostituito con uno funzionalmente equivalente (se necessario in grado di fornire prestazioni migliori)
- Abbattimento dei costi
  - Grazie alla produzione su vasta scala dei componenti
- Semplicità di installazione e di esercizio del sistema

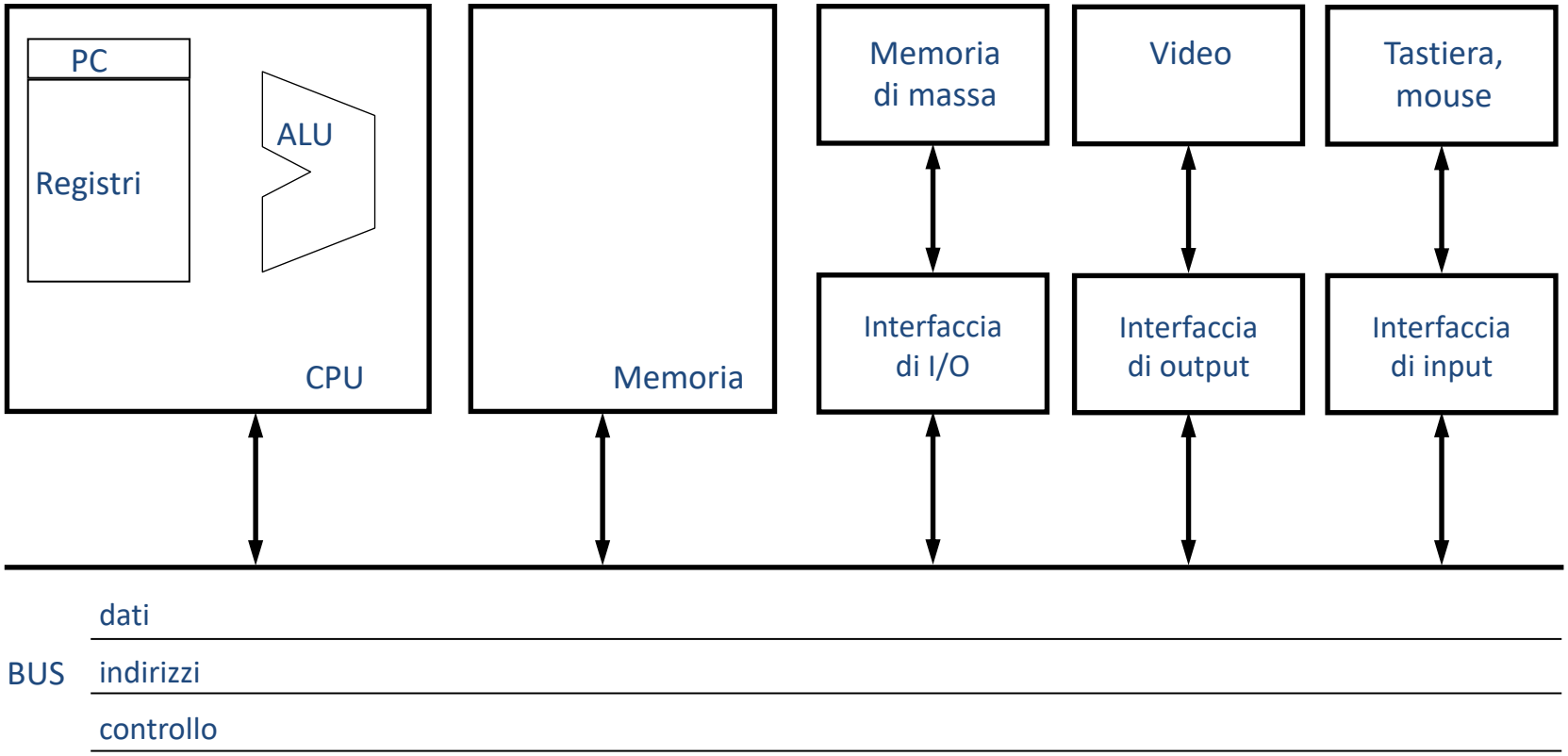
Architettura di Von Neumann

Descrizione top-down (di tipo gerarchico) del calcolatore, visto inizialmente come formato da pochi componenti generali

- "Architettura di von Neumann"
  - Architettura alla base dei moderni calcolatori
  - Ipotesi fondamentale: dati e istruzioni codificati in uno stesso formato, quindi possono essere memorizzati usando un unico dispositivo
  - Basato su quattro sottosistemi:



Lo schema di riferimento



Lo schema di riferimento

- Alla CPU sono affidate operazioni di controllo del flusso di esecuzione e di elaborazione vera e propria
- La memoria ospita dati e istruzioni per le elaborazioni della CPU e anche i dati per i trasferimenti da/verso i dispositivi di ingresso/uscita
  - La memoria può essere vista come un insieme di celle adiacenti, ciascuna identificata da un indirizzo univoco, in cui possono essere scritti o letti dati e istruzioni
- Il calcolatore interagisce con l'esterno attraverso dispositivi ausiliari, le periferiche, connesse al dispositivo mediante interfacce di ingresso/uscita
- Il collegamento dei sottosistemi è basato sui concetti di bus e di master-slave

Bus

- Il bus è un insieme di linee alla quali sono connessi tutti i componenti del calcolatore
  - Bus dati: per il trasferimento di dati fra i componenti (es. fra CPU e memoria)
  - Bus indirizzi: su cui viene indicato l'indirizzo di memoria da cui i dati devono essere letti o scritti
  - Bus di controllo: su cui transitano i segnali di controllo che permettono di specificare quali dispositivi sono coinvolti nel trasferimento
- La CPU ha il ruolo di gestione dell'intero sistema e agisce come "master" rispetto al bus
  - Controlla l'accesso al bus delle periferiche, che operano in modalità "slave" (non possono accedere autonomamente al bus)
  - Le linee di controllo vengono utilizzate per questo coordinamento

Clock

I dispositivi elettronici digitali (inclusa la CPU) sono dotati di un segnale periodico di sincronizzazione: il "clock"

- Agisce come un metronomo e scandisce le operazioni del dispositivo elettronico
- In particolare, il clock della CPU scandisce le operazioni di controllo ed esecuzione del processore
  - Ad esempio un clock di 1 GHz invia un miliardo di impulsi al secondo → un'operazione ogni nanosecondo
  - Maggiore è la frequenza di clock, maggiore è la frequenza di esecuzione delle istruzioni
  - Il completamento di un'istruzione richiede più cicli di clock
- Permette ai dispositivi di comunicare correttamente sul bus
  - E' necessario adeguarsi al dispositivo più lento collegato al bus ("possibile collo di bottiglia")

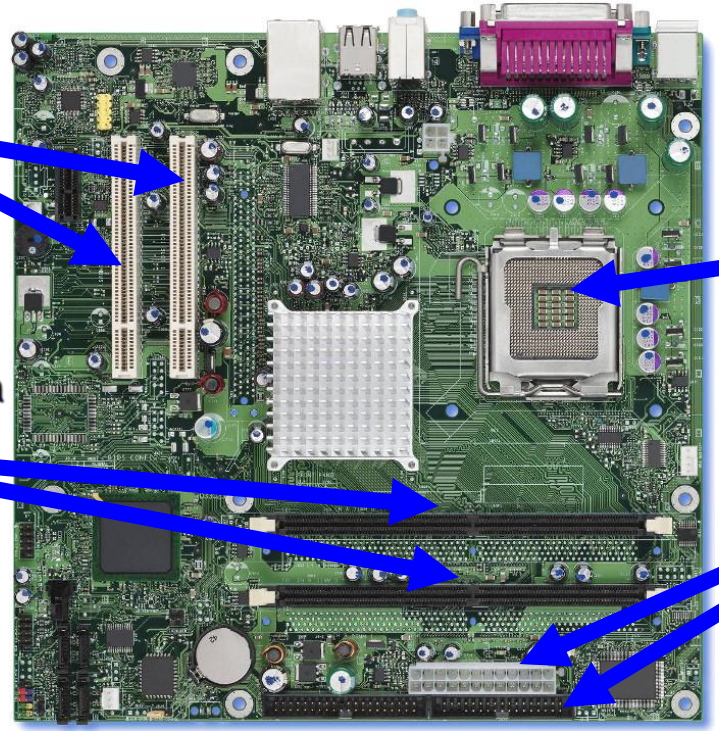
Bus

- L'uso di un unico insieme di linee per il bus è stato un approccio inizialmente vincente per
  - Semplicità ed economicità
  - Estendibilità e standardizzazione
- A fronte però di svantaggi sempre più vincolanti
  - Lentezza
  - Sovraccarico della CPU
- Questo approccio è diventato nel tempo un "collo di bottiglia" per il sistema
  - Introdotti meccanismi per liberare la CPU dal controllo del bus
  - Introdotti bus aggiuntivi per velocizzare i trasferimenti (ad esempio con la memoria o la scheda video)
- Approccio a bus singolo resta valido a livello didattico

Scheda madre

I circuiti del bus sono solitamente integrati nella scheda madre (motherboard)

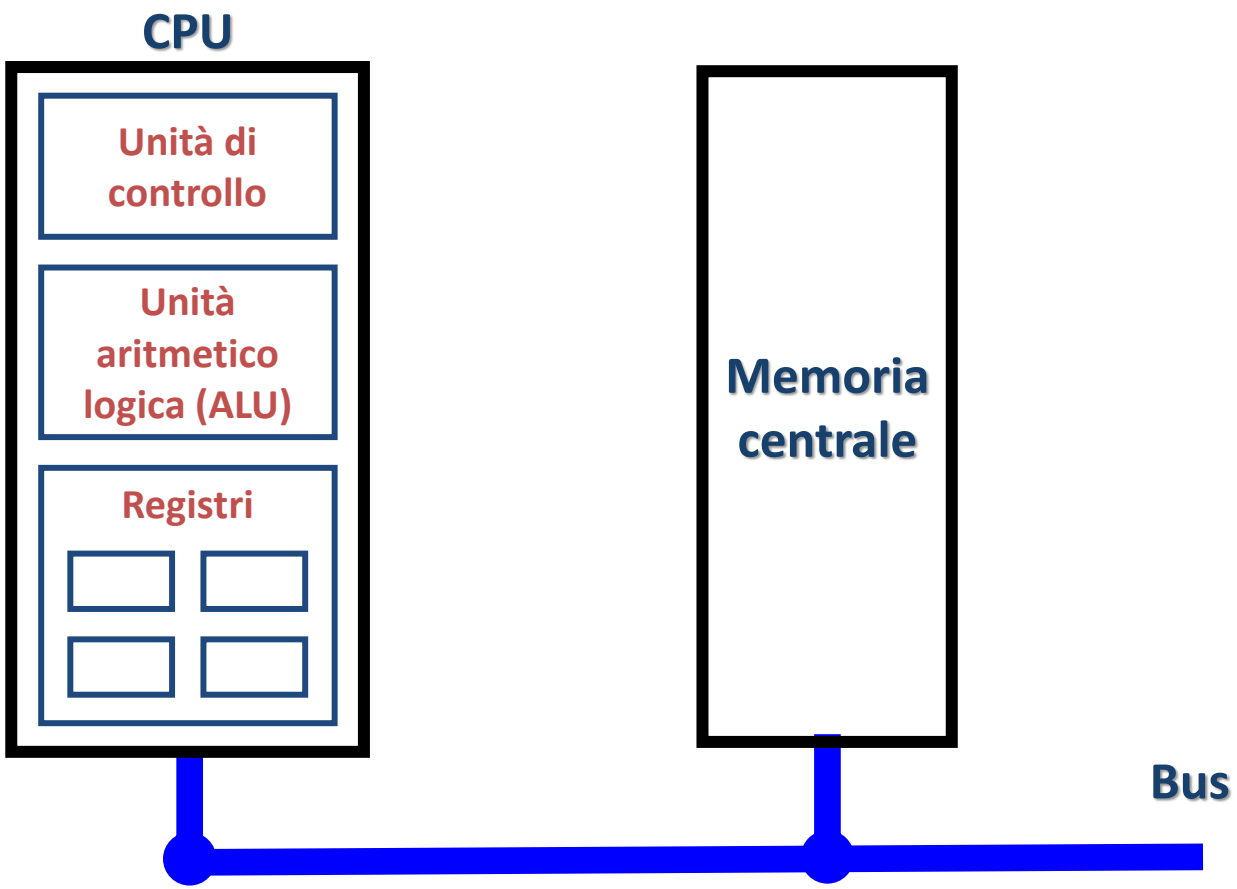
- Scheda elettronica su cui sono montati direttamente o collegati mediante connettori i vari componenti (CPU, memoria centrale, periferiche di I/O)



CPU

Unità centrale di elaborazione

Il processore (CPU)



Il processore (CPU)

Un calcolatore basato sull'architettura di Von Neumann esegue un programma sulla base dei seguenti principi:

- Dati e istruzioni sono memorizzati in una stessa memoria che permette sia la scrittura che la lettura
- I contenuti della memoria sono indirizzati in base alla loro posizione, indipendentemente dal tipo di dato o istruzione contenuto
- Le istruzioni vengono eseguite in modo sequenziale

La legge di evoluzione di un processore è semplice:

- *Prelevare, decodificare, ed eseguire istruzioni nell'ordine in cui sono memorizzate nella memoria principale, fino a quando non viene incontrata un'istruzione che stabilisce esplicitamente un ordine diverso da quello sequenziale per la prossima istruzione da prelevare ed eseguire; a partire da tale istruzione proseguire con le stesse modalità e così via all'infinito.*

Il processore (CPU)

Il linguaggio per cui la CPU si comporta da *esecutore* è detto linguaggio macchina:

- Le istruzioni scritte in linguaggio macchina sono piuttosto rudimentali
  - Il numero di operandi è limitato (in genere non più di due)
  - Il numero di operazioni previste è ridotto
  - Il concetto di tipo di dato è quasi assente

Codifica delle istruzioni

Servono regole di codifica per la rappresentazione delle istruzioni in memoria principale

In generale, per rappresentare informazioni in un calcolatore si adotta sempre una *codifica binaria*

- Alfabeto costituito da due soli simboli, convenzionalmente rappresentati con '0' e '1', detti perciò *simboli binari*
- Un'informazione (dato o istruzione) viene rappresentata con una sequenza di *n* simboli binari
- Il numero *n* di simboli binari utilizzati dipende dal numero di scelte possibili per il dato o istruzione da rappresentare:
  - Se è scelto da un insieme di due elementi → è sufficiente un simbolo
  - Se è scelto da un insieme di quattro elementi → due simboli binari
  - Ogni volta che raddoppio gli elementi, ho bisogno di un simbolo binario in più
  - In generale, con n simboli binari rappresento fino a 2^n elementi di un insieme

Codifica delle istruzioni

- Ciascun simbolo binario viene memorizzato a livello fisico mediante un dispositivo in grado di assumere in modo stabile due possibili configurazioni (*dispositivo bistabile)*
- La memoria è costituita da un insieme di dispositivi bistabili, chiamati *bit (binary digit)*
  - I bit sono raggruppati in successioni di 8, chiamate *byte* e che costituiscono una *cella* di memoria
  - Un indirizzo di memoria permette di accedere in modo univoco ad una specifica *cella* di memoria
  - Gruppi di N celle consecutive formano una *parola (word),* dove N dipende dall'architettura (es. architettura a 32 bit → N = 32/8 = 4)
  - E' possibile scrivere/leggere in/da memoria una parola (N celle) con un'unica operazione attraverso il sistema di interconnessione

Codifica delle istruzioni

Anche le istruzioni devono essere rappresentate mediante una opportuna codifica

- Ad ogni istruzione è associato un codice operativo univoco (*opcode*)
- Le istruzioni possono essere raggruppate in tre classi
  - Istruzioni aritmetico-logiche
  - Istruzioni per il trasferimento dati
  - Istruzioni di controllo

Codifica delle istruzioni

Esempio di codifica dell'opcode su 8 bit:

Aritmetico-logiche

Codice	Istruzione
0111 1100	add
0111 1101	sub
0111 1110	and
...	...

Trasferimento dati

Codice	Istruzione
1110 1000	load
1111 1000	store
...	...
...	...

Controllo

Codice	Istruzione
0100 1001	if_eq
0100 1000	call
0100 1100	return
...	...

Codifica delle istruzioni

- Un'istruzione deve anche specificare i dati o i riferimenti ai dati necessari per l'esecuzione dell'istruzione stessa
  - Operandi
  - Dove scrivere il risultato
- Ad esempio nell'addizione (`add`) si devono specificare
  - Due operandi (gli addendi)
  - Dove scrivere il risultato
- Il numero di operandi dipende dall'istruzione (es. il calcolo del valore assoluto richiede un solo operando)

Codifica delle istruzioni

- L'operando può essere specificato
  - Indicando da dove deve essere prelevato (sorgente)
    - La sorgente può essere un registro del processore o un indirizzo di memoria
  - Direttamente nella stessa istruzione (operando immediato)
- La destinazione può essere specificata
  - Indicando un registro o un indirizzo di memoria

Alcuni esempi di formati per istruzioni in linguaggio macchina:

Codice operativo	Destinazione	Sorgente 1	Sorgente 2

Codice operativo	Destinazione	Sorgente 1	Operando

Codice operativo	Destinazione	Operando

Codifica delle istruzioni

- Il linguaggio *assembly* è un linguaggio di programmazione in corrispondenza uno a uno con il linguaggio macchina
  - Permette di utilizzare una rappresentazione in forma mnemonica
  - Maggiore leggibilità per il programmatore rispetto al formato binario
  - Tradotto automaticamente in linguaggio macchina da programma assemblatore (*assembler*)

Esempio di rappresentazione di un'istruzione

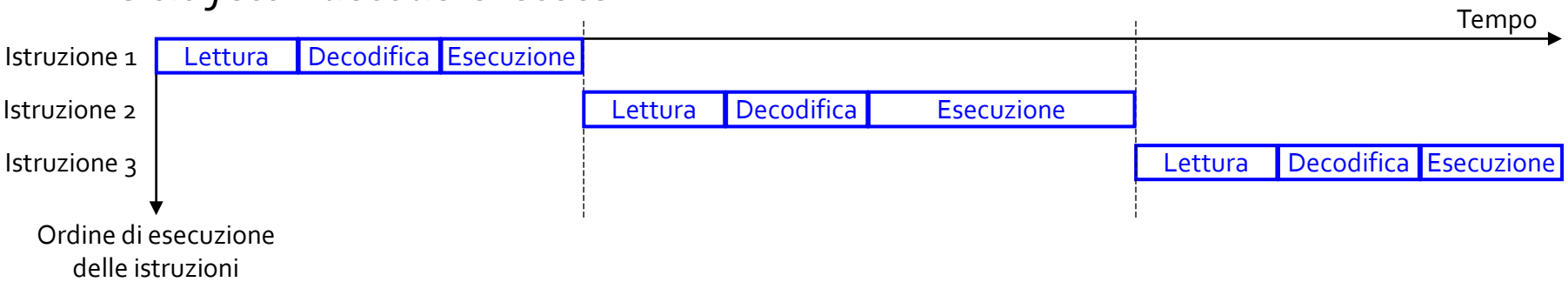
Linguaggio	Codice operativo	Destinazione	Sorgente 1	Sorgente 2
Linguaggio macchina	0111 1100	00001	00010	00011
Linguaggio assembly	add	R01	R02	R03

Fasi della CPU

Il comportamento della CPU può essere descritto come l'esecuzione ciclica di tre fasi, scandita dagli impulsi di sincronizzazione del clock:

- Lettura (*fetch*)
  - Trasferimento dalla memoria alla CPU della prossima istruzione da eseguire
- Decodifica (*decode*)
  - Riconoscimento dell'istruzione e identificazione delle operazioni da svolgere
- Esecuzione (*execute*)
  - Svolgimento delle operazioni corrispondenti all'istruzione

Ciclo fetch-decode-execute

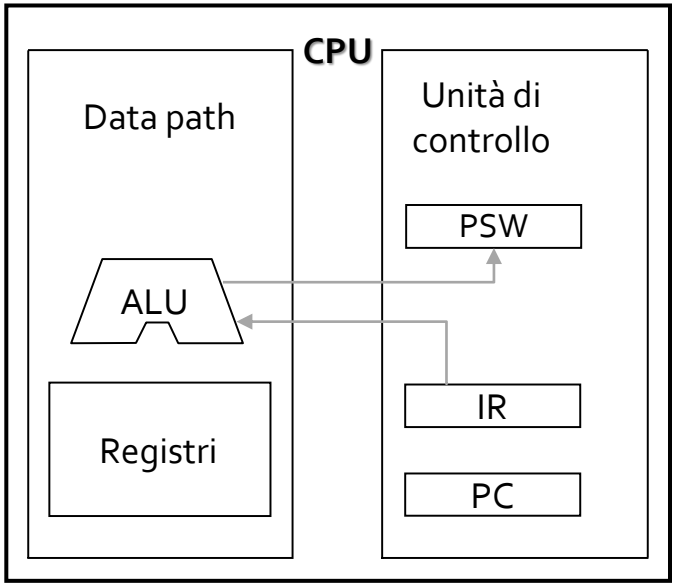


Le componenti della CPU

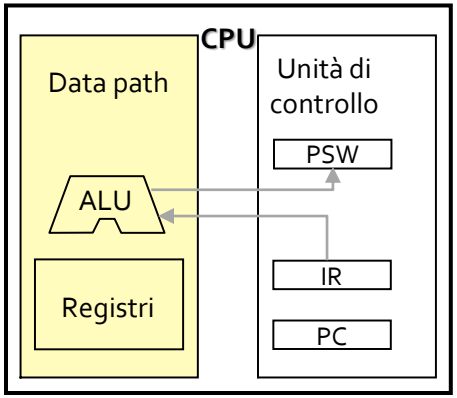
In un processore si possono identificare due componenti principali:

- Data path (o percorso dei dati)
  - Parte che si occupa dell'effettiva elaborazione dei dati
  - Comprende:
    - Una o più unità aritmetico-logiche, dette ALU (Arithmetic Logic Unit)
    - Alcune unità di memorizzazione temporanea, i registri, memoria ad alta velocità usata per operandi e risultati temporanei
- Unità di controllo
  - Coordina le operazioni di tutto il processore (anche del data path)
  - Regola il flusso dei dati e indica quali registri debbano essere collegati agli ingressi e all'uscita della ALU
  - Invia alla ALU il codice dell'istruzione da eseguire
  - Riceve indicazioni sull'esito dell'elaborazione appena eseguita dalla ALU e gestisce opportunamente queste informazioni
  - Comprende alcuni registri di uso specifico
    - Program Counter (PC) – Indirizzo dell'istruzione successiva in memoria
    - Instruction Register (IR) – Contiene l'istruzione in corso d'esecuzione
    - Process Status Word (PSW) – registro di stato
    - …

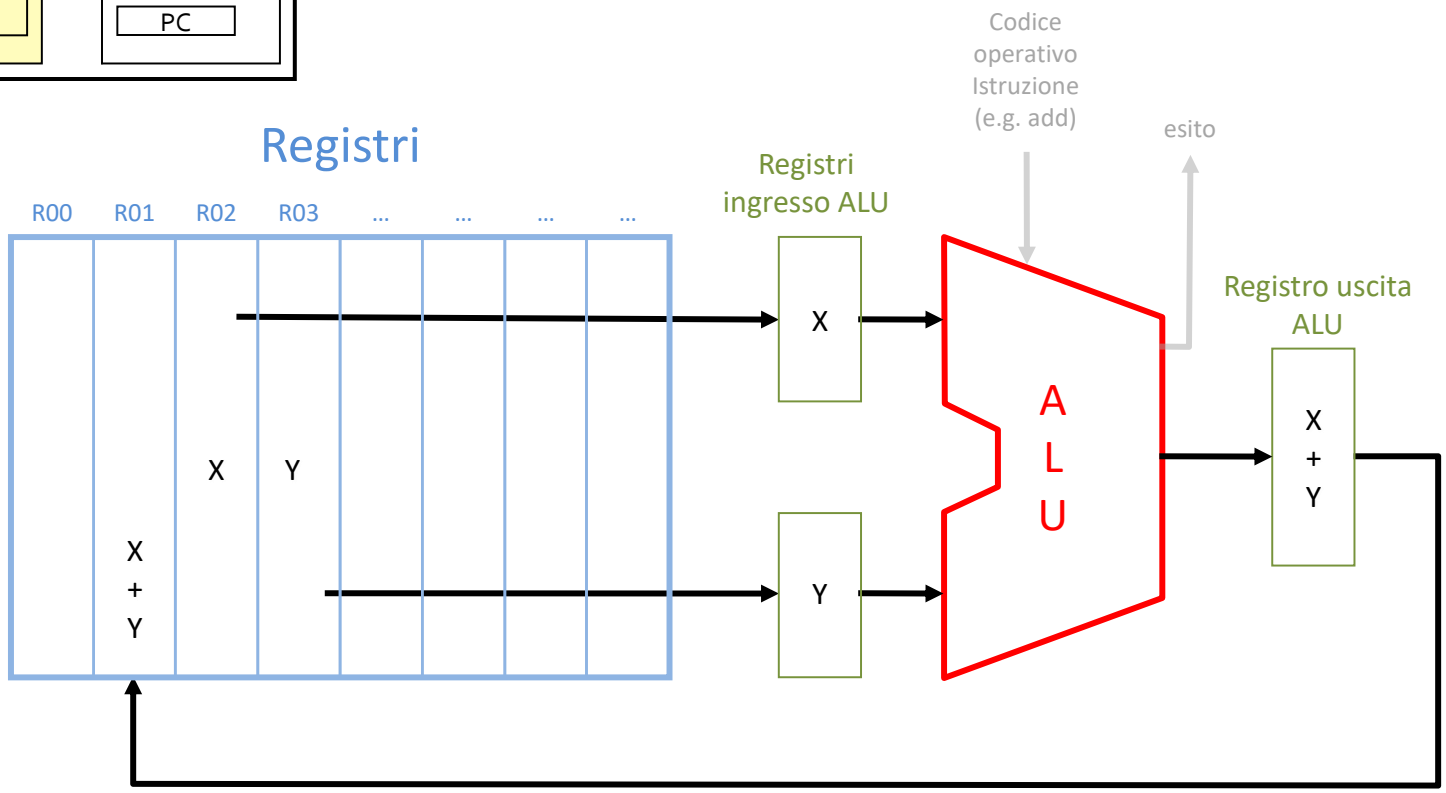
Data path e unità di controllo



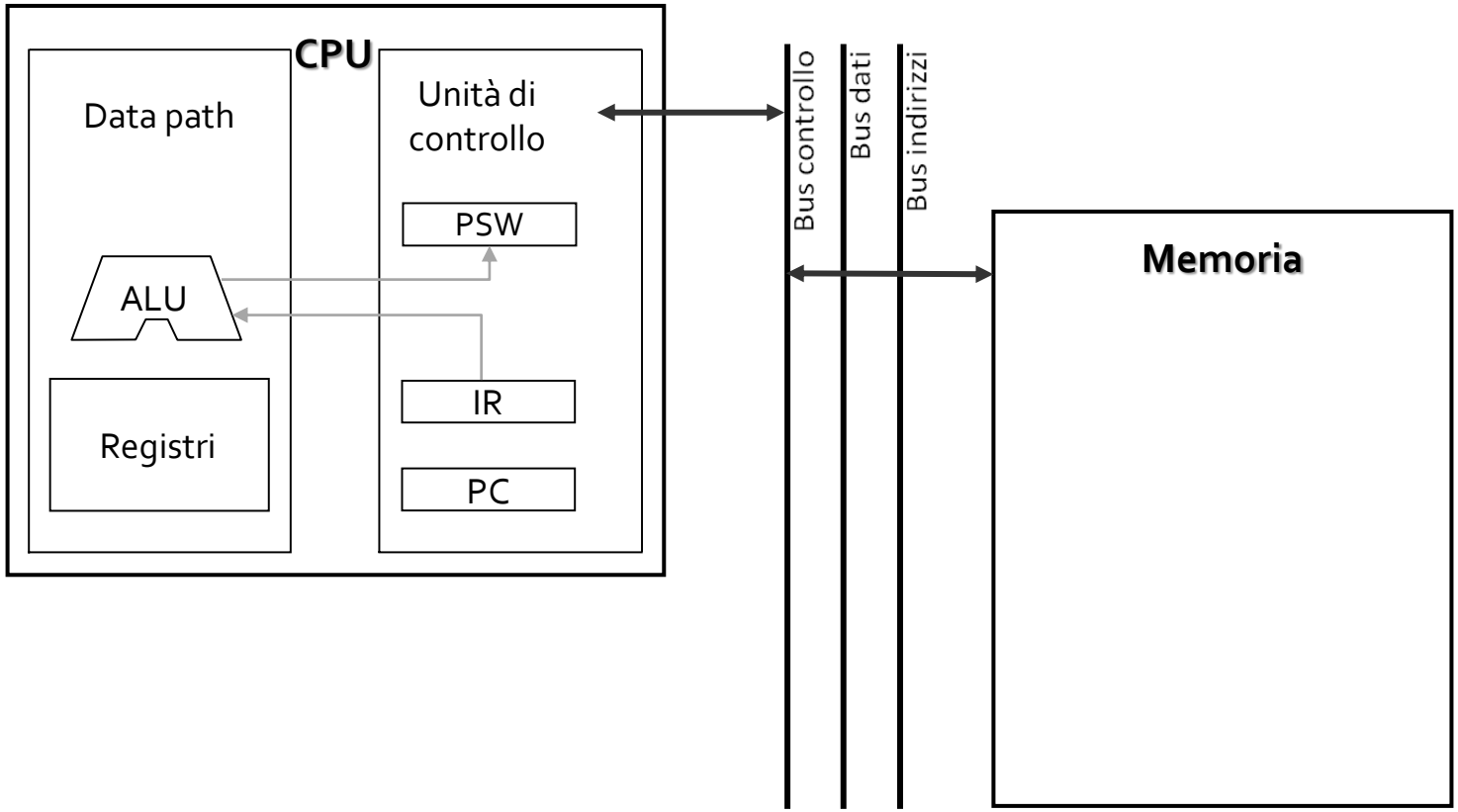
Data path – esempio esecuzione istruzione



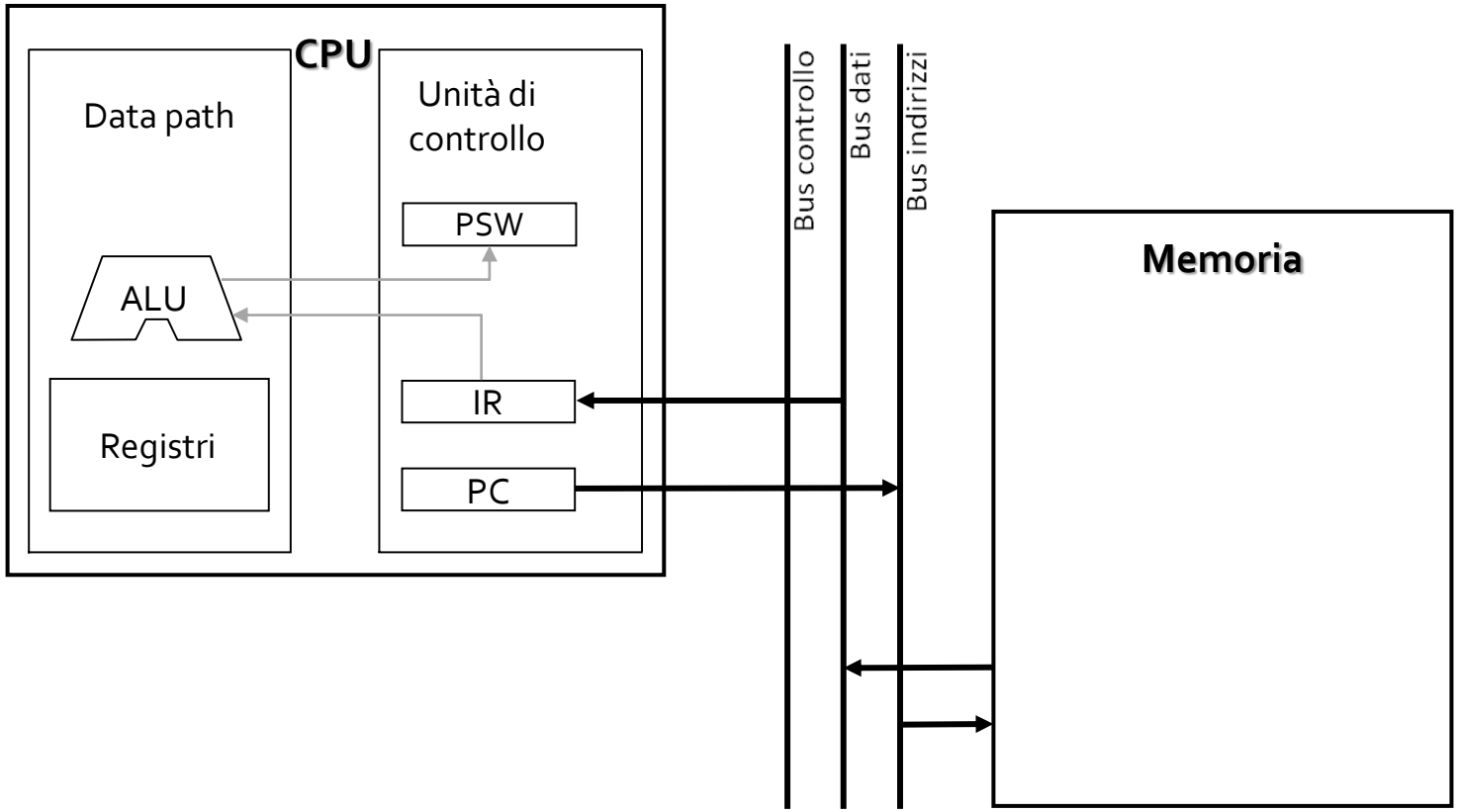
Codice operativo	Destinazione	Sorgente 1	Sorgente 2
add	R01	R02	R03



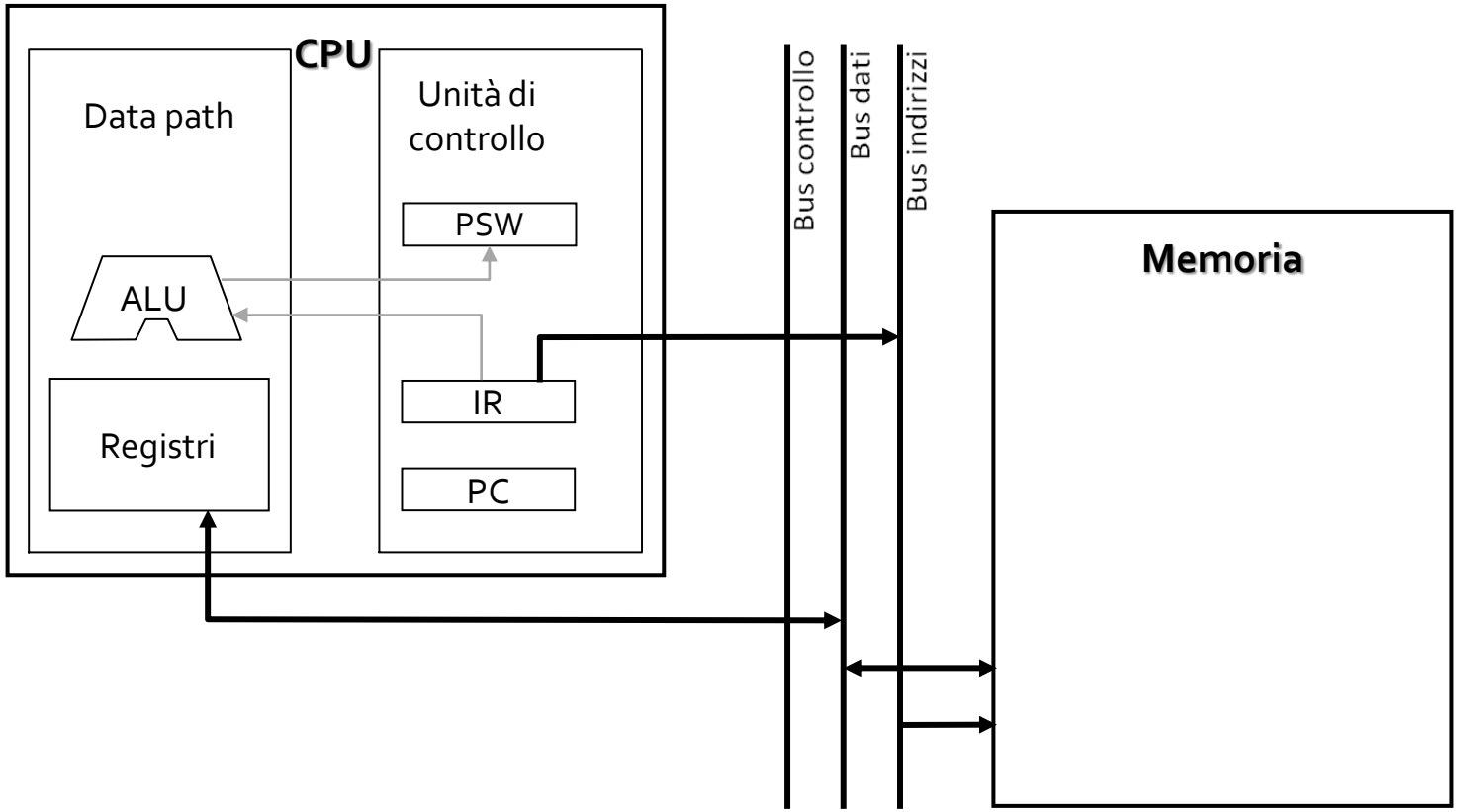
Interazione con la memoria – controllo



Interazione CPU-memoria – lettura istruzione



Interazione CPU-memoria – trasfer. dati



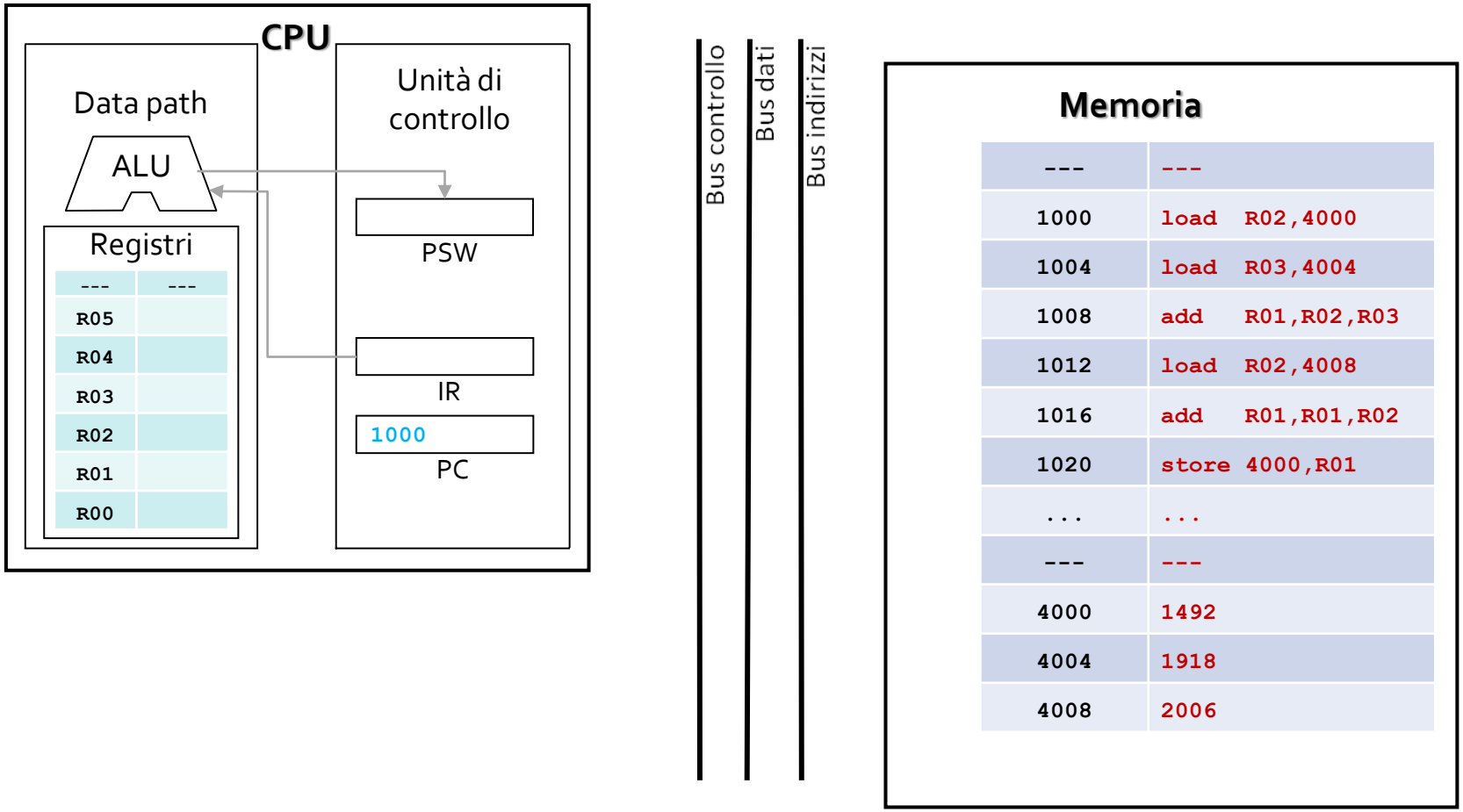
Esempio sequenza di istruzioni

Ipotizziamo che:

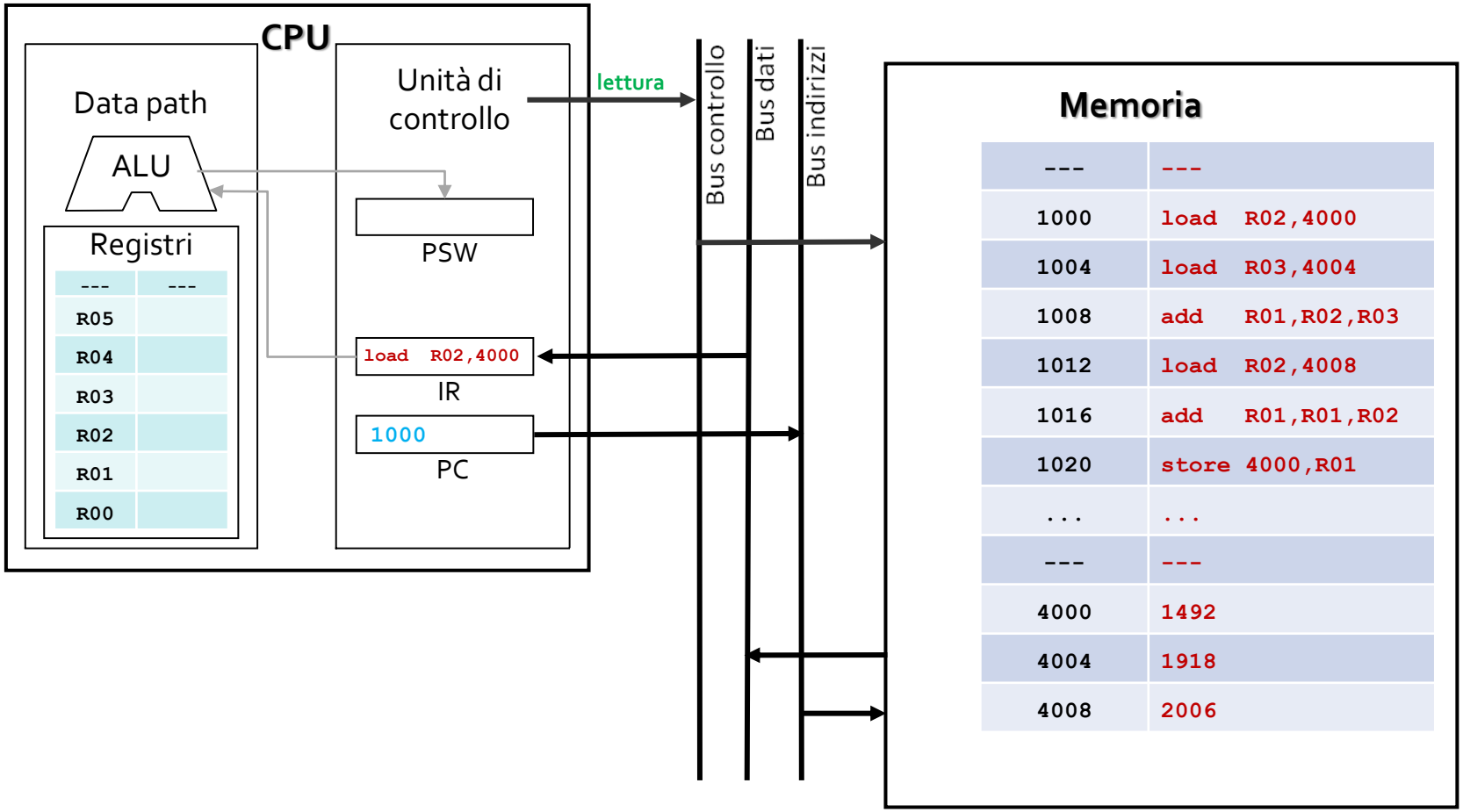
- Si debba eseguire l'istruzione di "alto livello" **A ⟵ A + B + C**
  (assegna alla variabile A la somma del contenuto delle variabili A, B e C)
- Le variabili A, B e C si trovino rispettivamente all'indirizzo di memoria centrale 4000 (A), 4004 (B) e 4008 (C)
- Le corrispondenti istruzioni in linguaggio macchina si trovino all'indirizzo 1000, 1004, 1008, … (come riportato nella tabella sottostante)

Indirizzo	Istruzione	Significato
1000	`load R02,4000`	*trasferisce il contenuto dell'indirizzo 4000 (A) nel registro R02*
1004	`load R03,4004`	*trasferisce il contenuto dell'indirizzo 4004 (B) nel registro R03*
1008	`add R01,R02,R03`	*somma il contenuto dei registri R02 e R03 e scrive il risultato in R01*
1012	`load R02,4008`	*trasferisce il contenuto dell'indirizzo 4008 (C) nel registro R02*
1016	`add R01,R01,R02`	*somma il contenuto dei registri R01 e R02 e scrive il risultato in R01*
1020	`store 4000,R01`	*trasferisce il contenuto del registro R01 nell'indirizzo 4000 (A)*
...	...	...

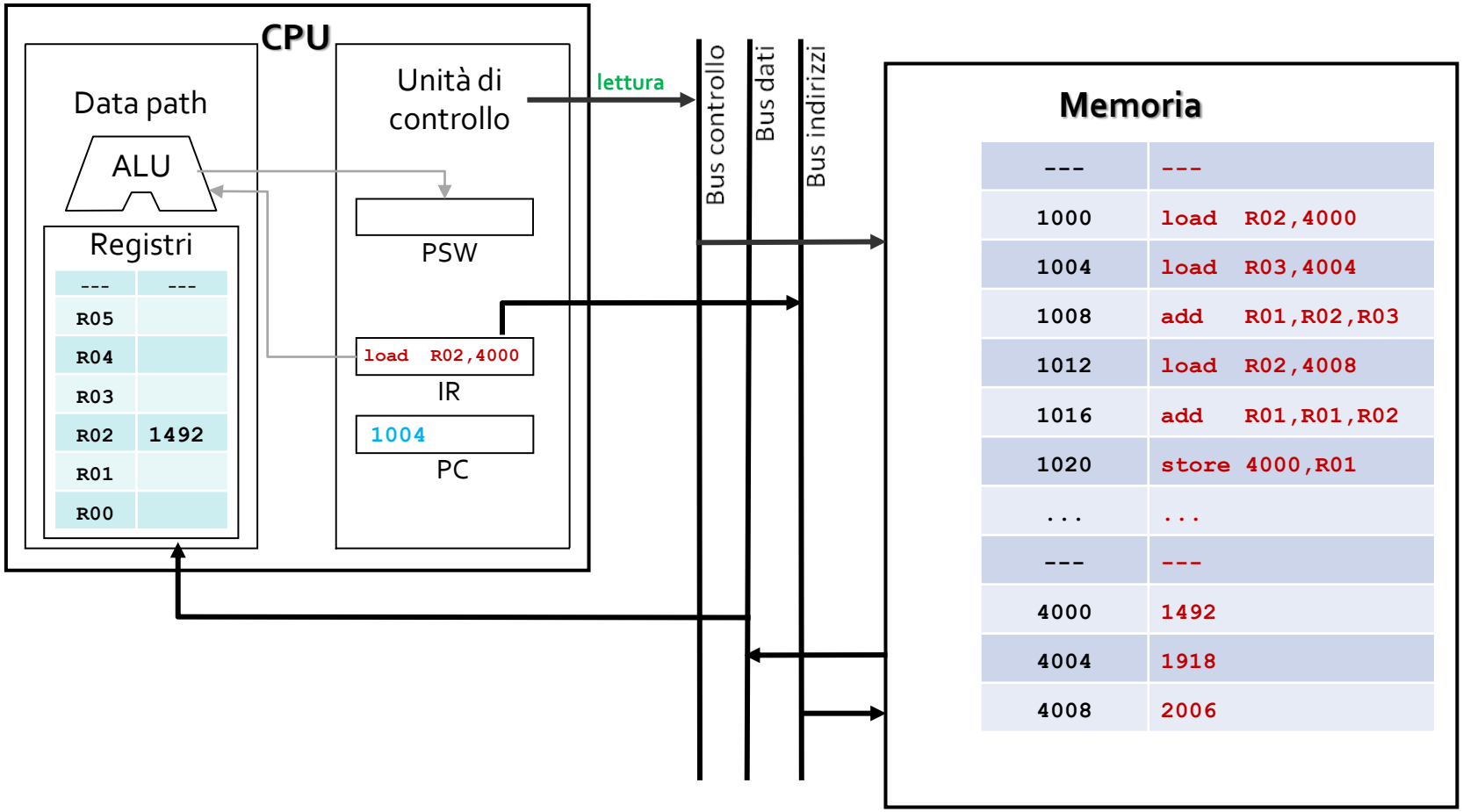
Situazione iniziale



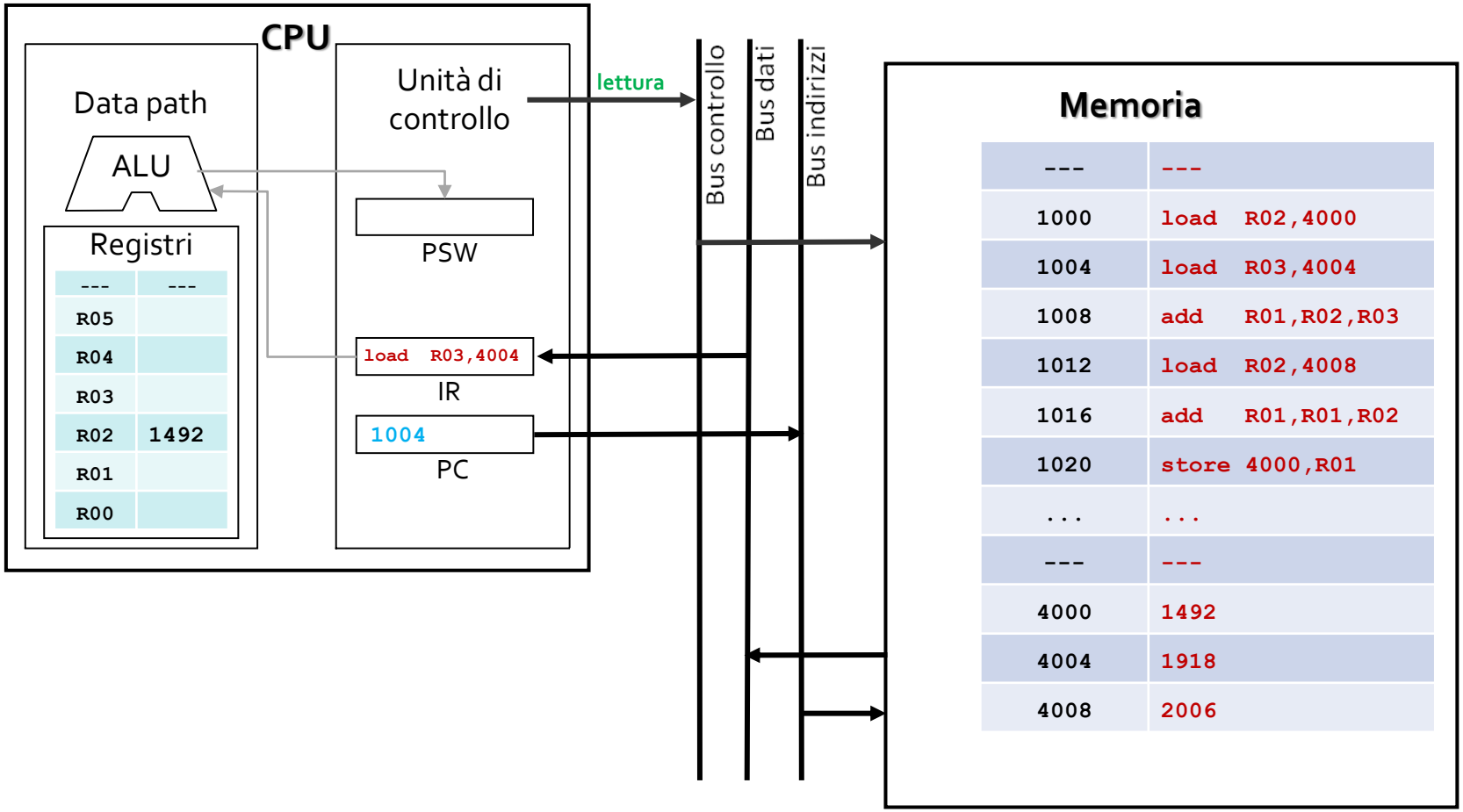
Lettura istruzione 1000



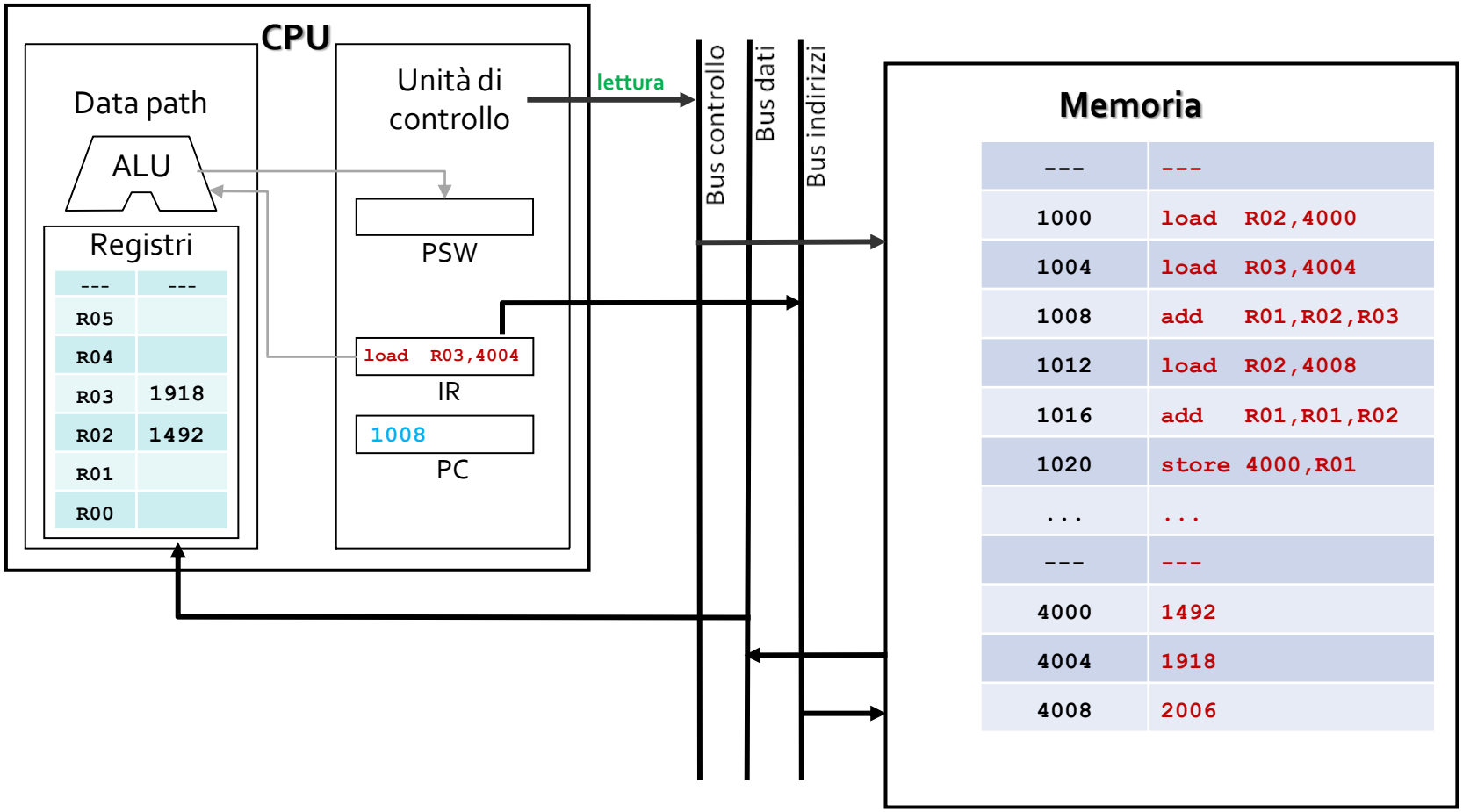
Esecuzione istruzione 1000



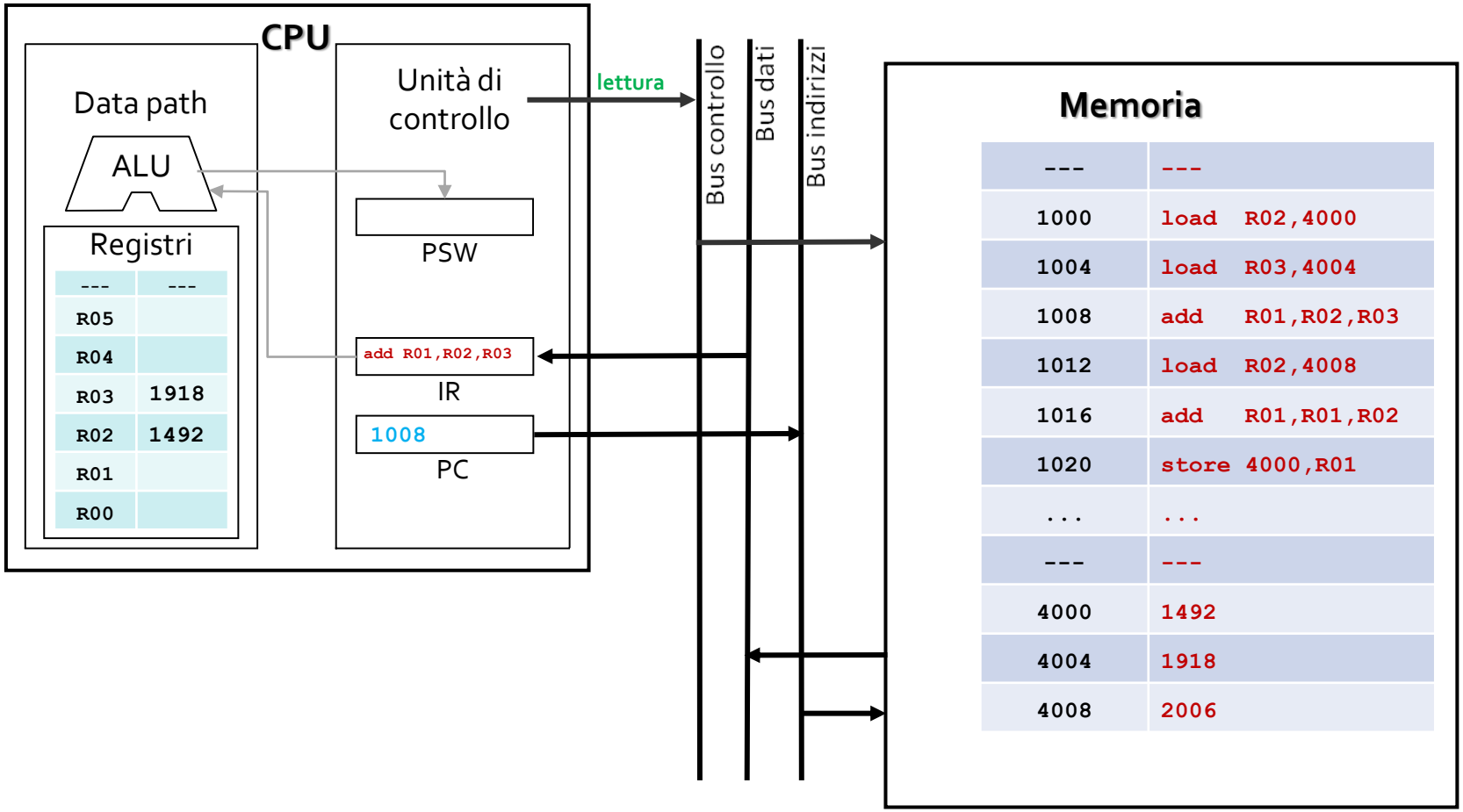
Lettura istruzione 1004



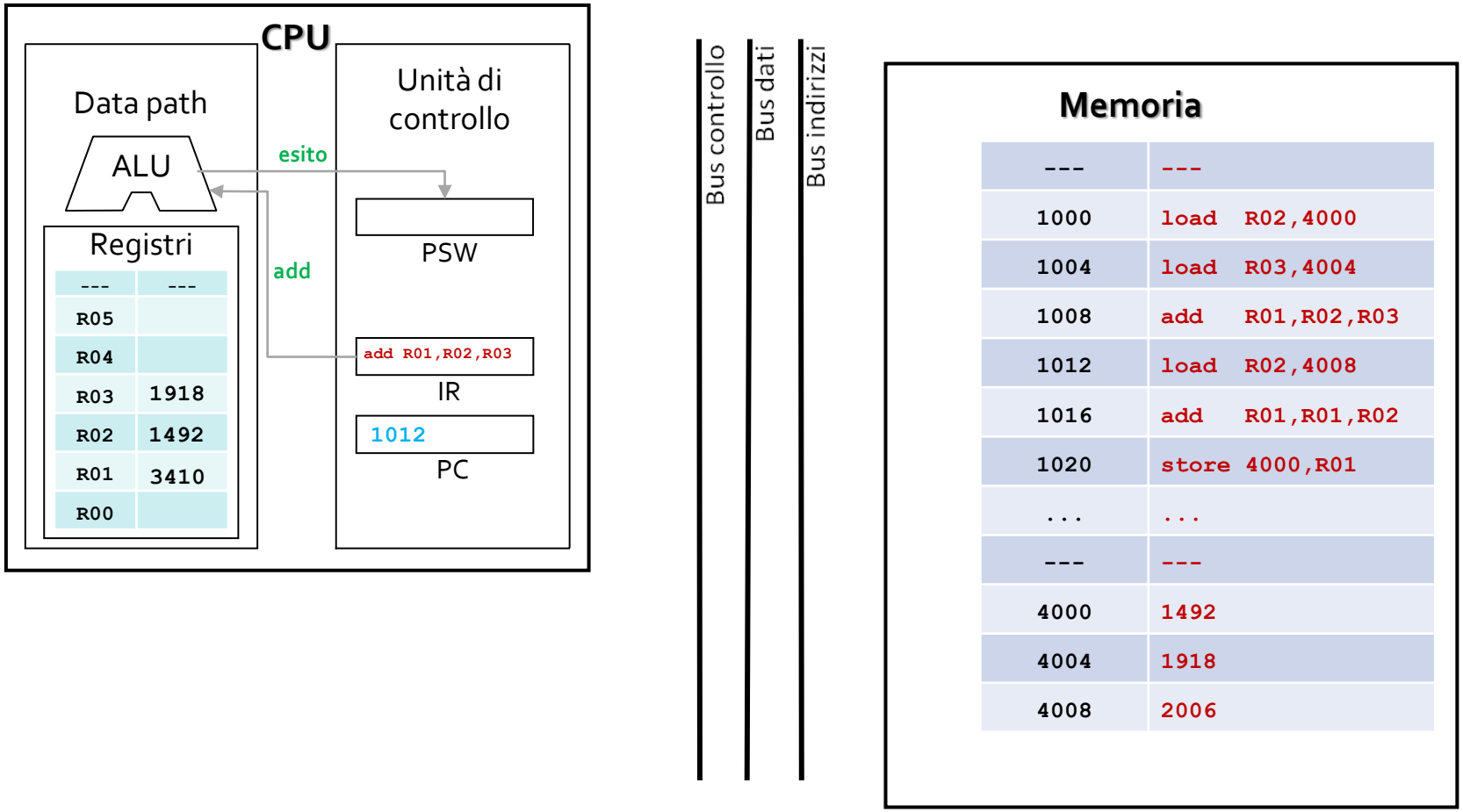
Esecuzione istruzione 1004



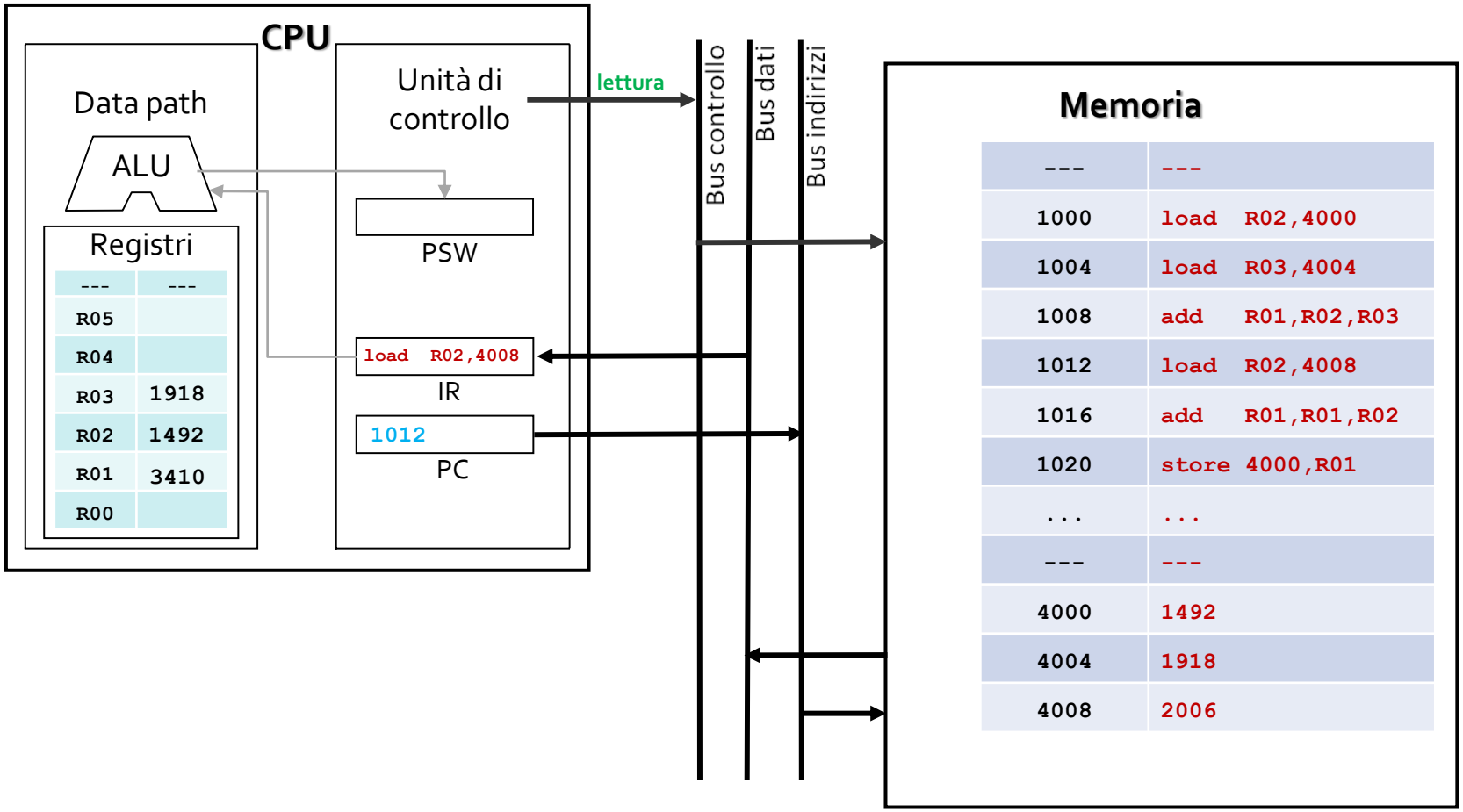
Lettura istruzione 1008



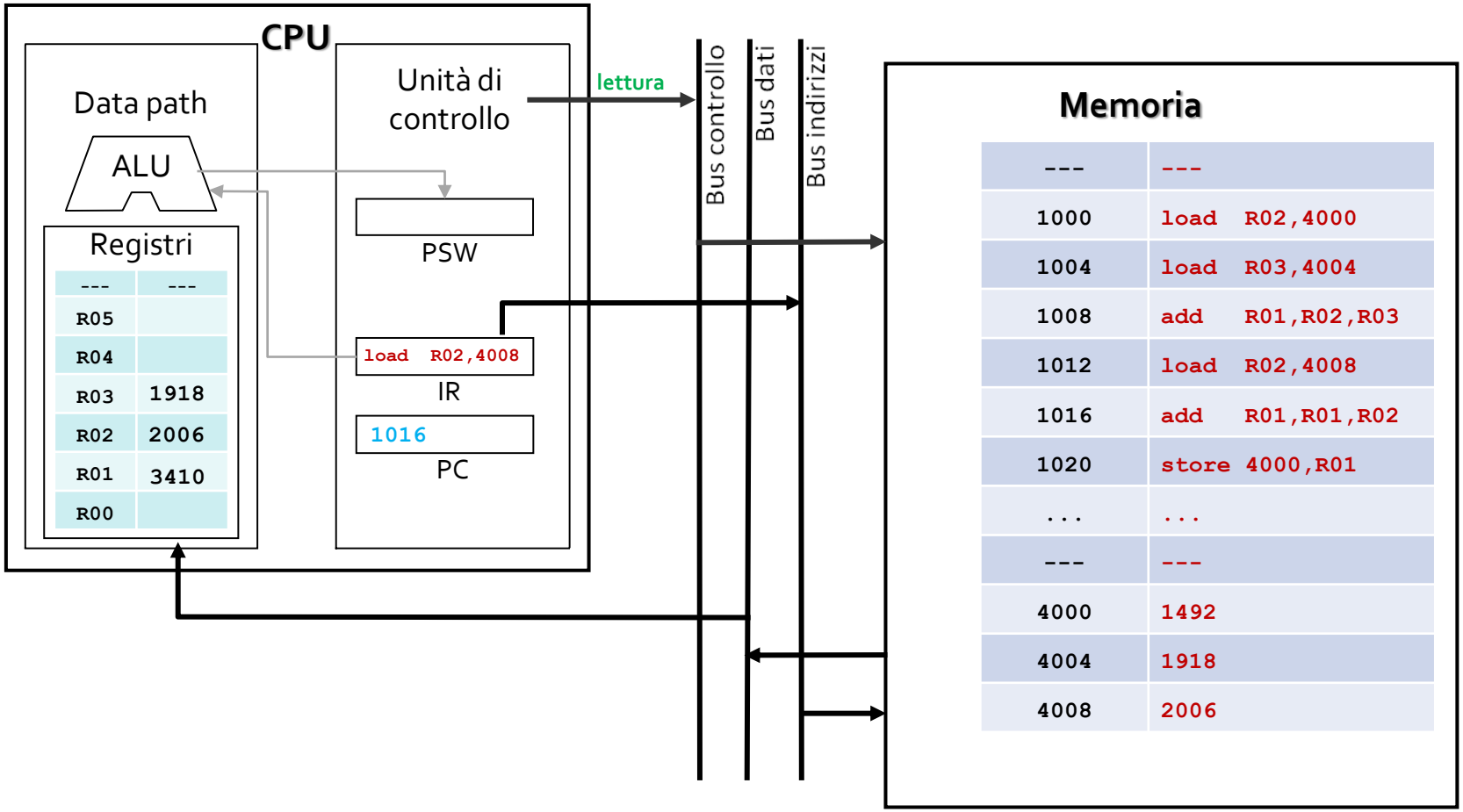
Esecuzione istruzione 1008



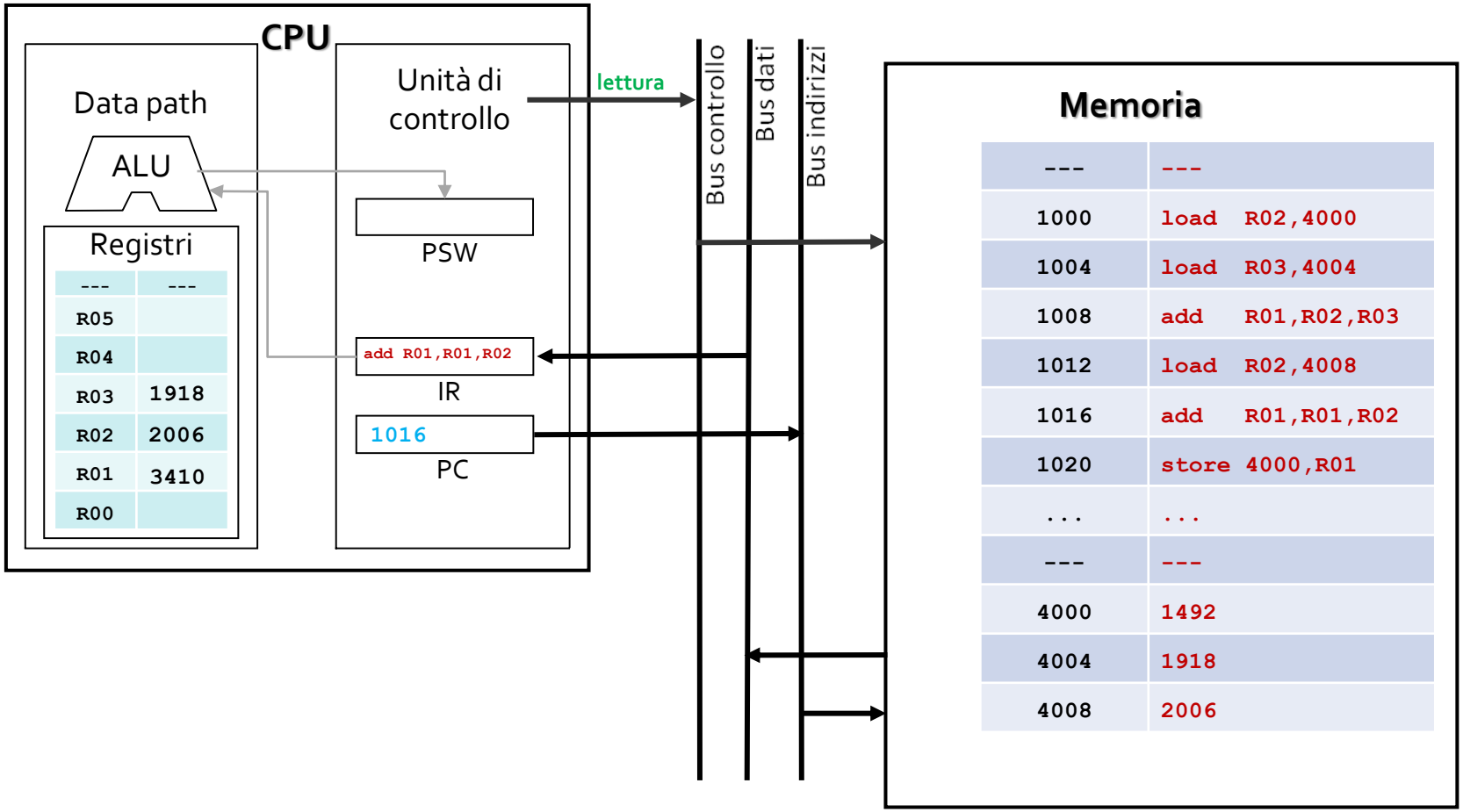
Lettura istruzione 1012



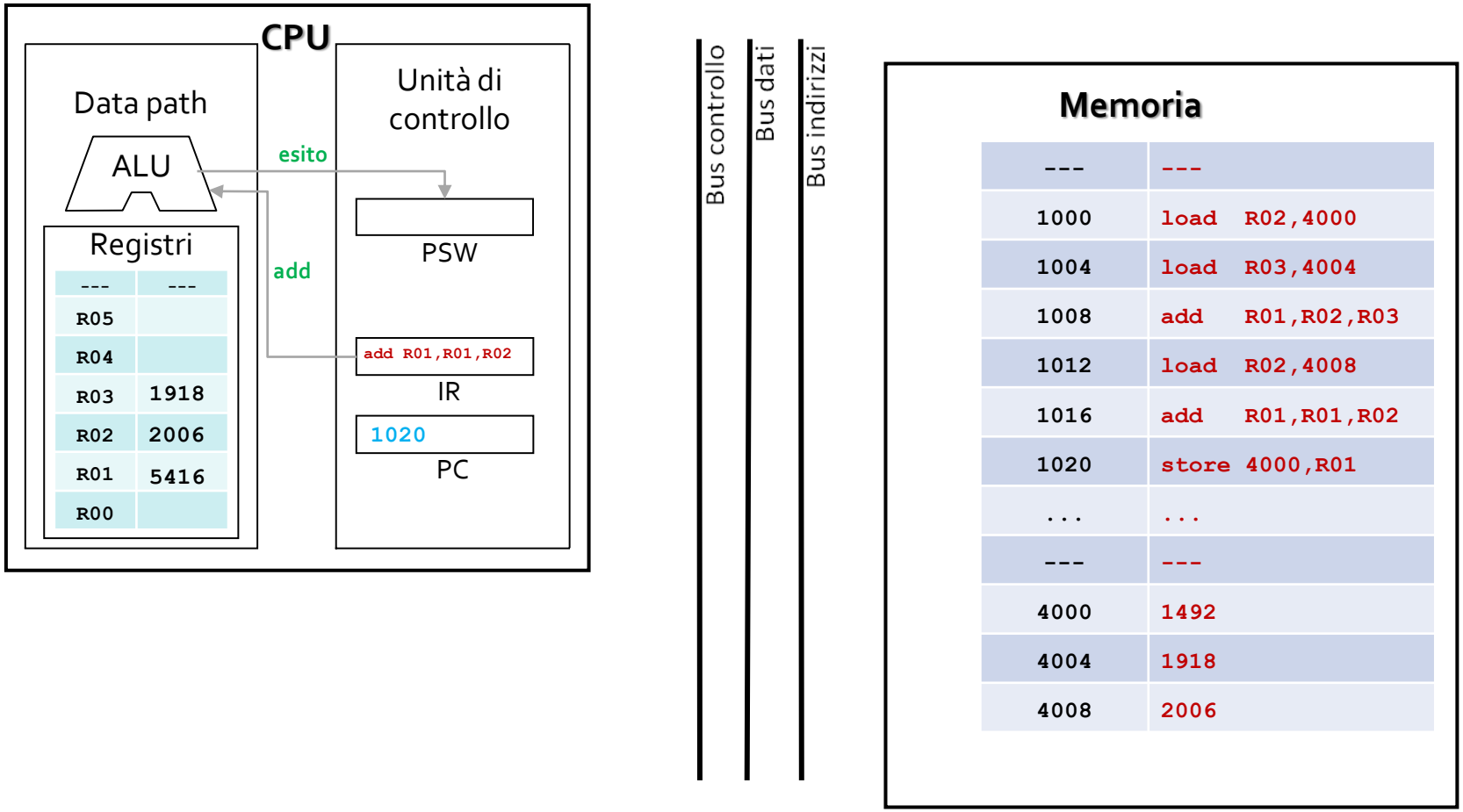
Esecuzione istruzione 1012



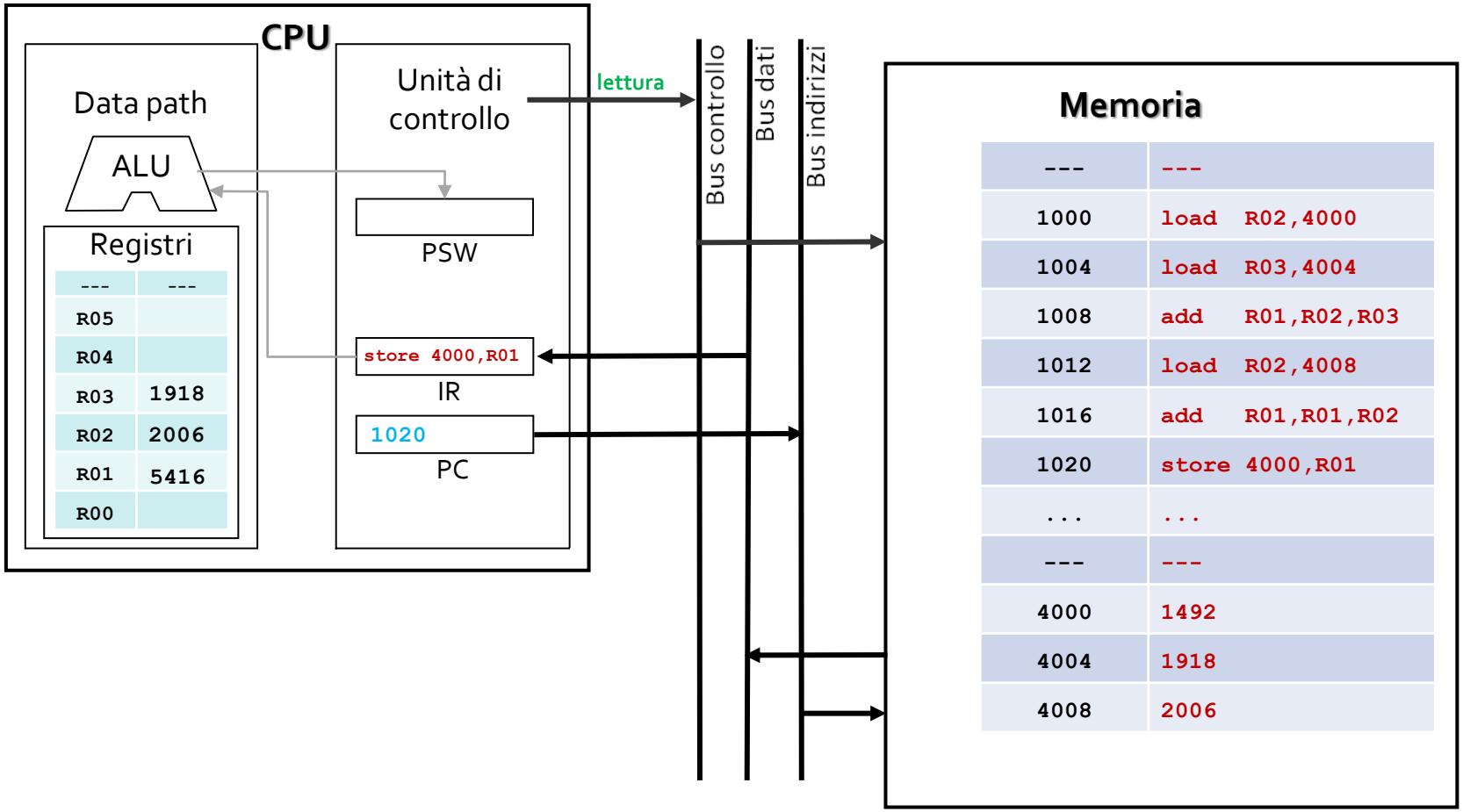
Lettura istruzione 1016



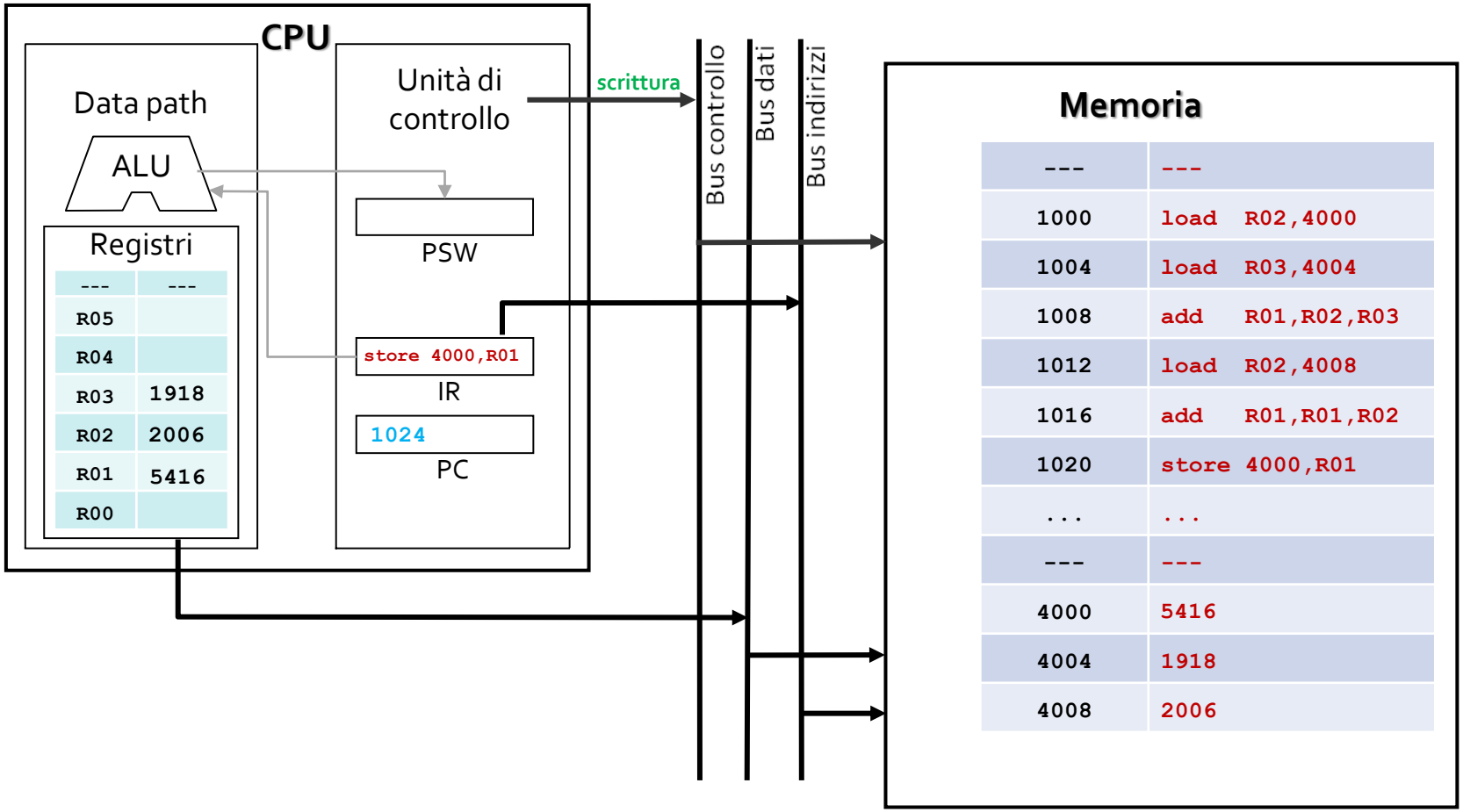
Esecuzione istruzione 1016



Lettura istruzione 1020



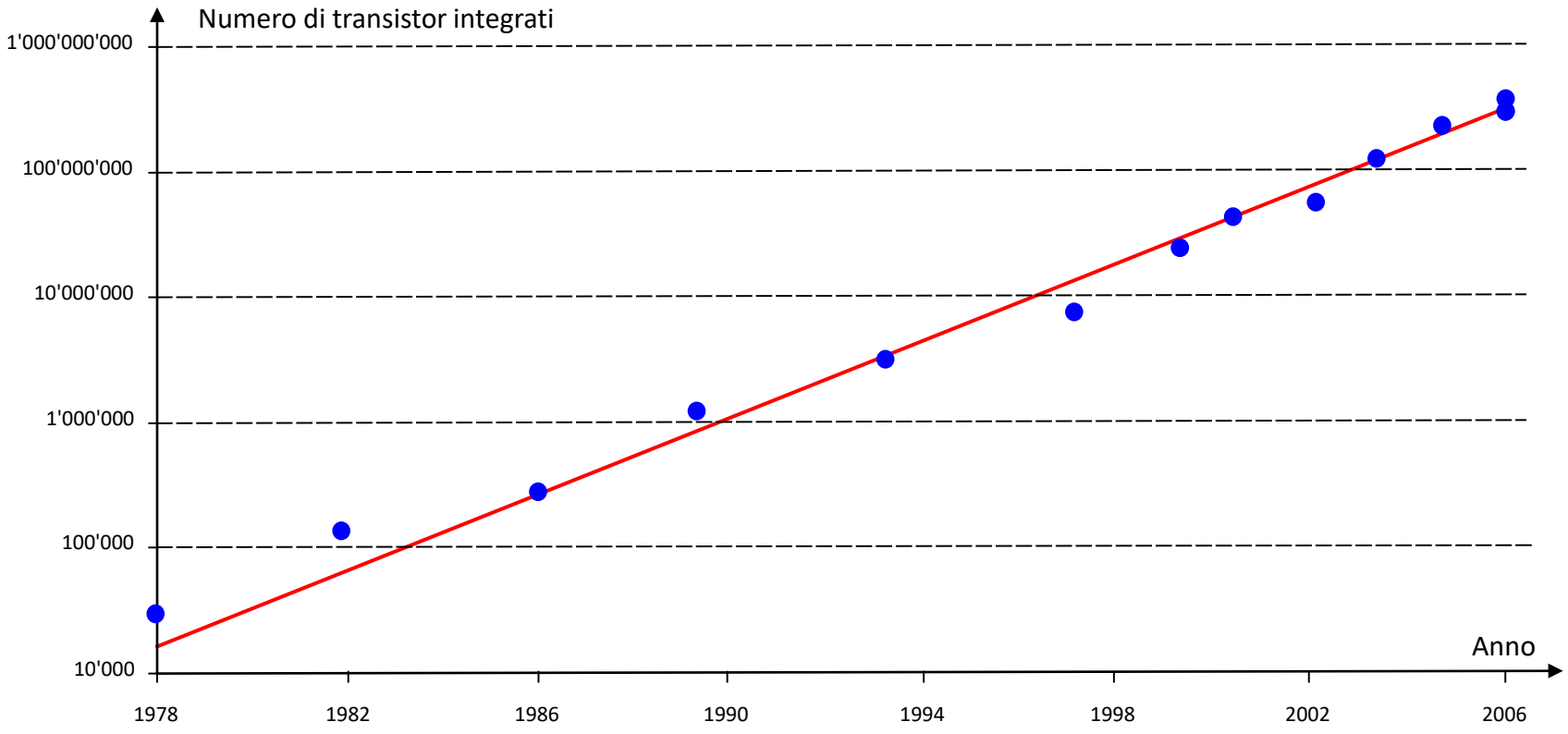
Esecuzione istruzione 1020



Evoluzione delle CPU

Legge di Moore (1965)

- Constatazione empirica di Gordon Moore (uno dei soci fondatori di Intel)
- Raddoppio numero di transistor integrati ogni 24 mesi



Evoluzione delle CPU

Famiglia CPU	Anno	Frequenza (MHz)	Dimensione registri (bit)	Numero di transistor
8086	1978	4.77 – 12	8	29 000
80286	1982	8 – 16	16	134 000
80386	1986	16 – 33	32	275 000
80486	1989	16 – 50	32	1 200 000
Pentium	1993	60 – 200	32	3 100 000
Pentium II	1997	233 – 400	32	7 500 000
Pentium III	1999	450 – 1133	32	24 000 000
Pentium 4	2000 – 2004	1300 – 3800	32	42-125 000 000
Pentium D	2005 – 2006	2800 – 3600	64	230-376 000 000
Core 2 Duo	2006	1666 – 3200	64	291 000 000
Core i7	2008 –	2666 – 3600	64	781 000 000

MEMORIA

Memoria centrale vs Memoria di massa

- La memoria centrale è destinata a contenere le istruzioni e i dati dei programmi in esecuzione
  - Funge da "memoria operativa" per il processore
  - Deve essere veloce per sfruttare il più possibile le potenzialità del processore
  - Non deve essere troppo costosa
- La tecnologia disponibile permette di avere memorie veloci a costi contenuti solo nella forma di memorie *volatili*
  - Il contenuto viene perso se il computer non è più alimentato (spegnimento)
  - Si usano circuiti a semiconduttori ad accesso diretto (Random Access Memory – RAM, diventato ormai sinonimo di memoria centrale)
- Per mantenere i dati in modo in modo permanente, invece, si sfrutta la *memoria di massa*, realizzata con tecnologie differenti
  - Non volatile
  - Costo per bit minore (quindi può essere più capiente)
  - Più lenta

Memoria per l'avvio del sistema

Nel sistema è presente anche una memoria di tipo *non volatile* destinata a contenere le istruzioni che il processore deve eseguire alla sua accensione

- Dal punto di vista architetturale è considerata parte della memoria centrale
  - Si trova solitamente su un chip dedicato collegato alla scheda madre
- Usata dalla CPU al momento dell'accensione per leggere la procedura di avvio (detta anche *firmware* o *BIOS*)
  - L'esecuzione della procedura di avvio è detta *bootstrap*
- La memoria utilizzata è di tipo elettronico (flash) e può essere riscritta per aggiornare il firmware
  - E' detta ROM (Read Only Memory), perché nei primi computer si usavano tecnologie non riscrivibili

Struttura della memoria centrale

Successione di elementi bistabili (*bit – binary digits*), capaci di assumere due configurazioni indicate convenzionalmente con '0' e '1'

- I bit della memoria centrale sono raggruppati in *celle*
  - In quasi tutti i calcolatori le celle sono composte da 8 bit (*byte*)
  - La cella è l'unità minima indirizzabile
- Le celle sono raggruppate in *parole (word)*
  - Nei moderni calcolatori una *parola* è costituita da 8 celle (8 byte – 64 bit)
  - Stessa dimensione dei registri del processore e del bus dati
  - E' possibile effettuare la lettura/scrittura di una parola in un'unica operazione

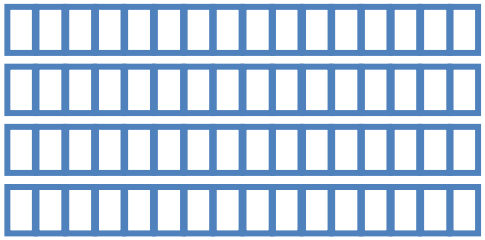
Struttura della memoria centrale

La cella è l'unità minima indirizzabile, ossia che può essere identificata in modo univoco mediante un indirizzo specificato sul bus indirizzi

- Se usiamo n bit per l'indirizzo possiamo indirizzare 2^n byte
  - Indirizzo a 16 bit → 64 KB indirizzabili
  - Indirizzo a 32 bit → 4 GB indirizzabili
  - Indirizzo a 64 bit → 16 exabytes (16 miliardi di GB o 16 milioni di terabyte)!

Esempio di indirizzi di parola, con 16 bit per parola e indirizzi su 16 bit

```
0000 0000 0000 0000
0000 0000 0000 0010
0000 0000 0000 0100
0000 0000 0000 0110
```



4 parole da 16 bit e
indirizzi a 16 bit

Prestazioni della memoria

- Tempo di accesso (*access time*)
  - Intervallo tra il momento in cui una richiesta di accesso dalla CPU arriva alla memoria e l'istante in cui la memoria rende disponibile il dato (lettura) o segnala che ha completato la scrittura del dato
- Velocità di trasferimento (*transfer rate*)
  - Misurata in bit al secondo o byte al secondo, quantità di dati trasferibili nell'unità di tempo
  - Si riferisce ad accessi multipli sequenziali
- Capacità
  - Quantità di dati che si possono memorizzare
  - Misurata in:
    - Kilobyte (KB) → 2^{10} byte → 1024 byte
    - Megabyte (MB) → 2^{20} byte → 1 048 576 byte
    - Gigabyte (GB) → 2^{30} byte → 1 073 741 824 byte
    - Terabyte (TB) → 2^{40} byte → 1 099 511 627 776 byte

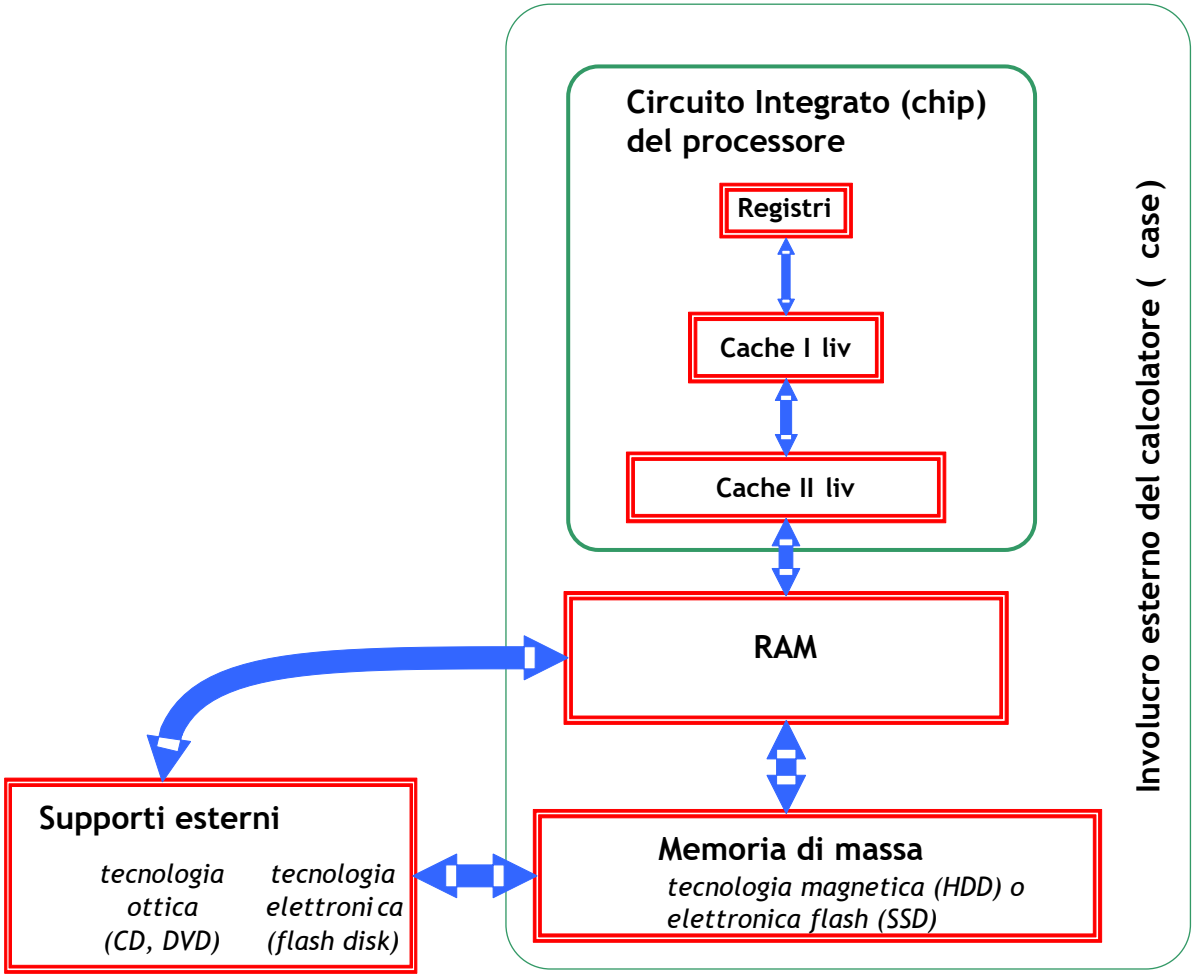
Esempi (ordini di grandezza)

- Tempo di accesso
  - Memoria di massa HDD (hard disk drive), disco magnetico
    → ordine dei *millisecondi*
  - Memoria di massa SSD (solid state drive), tecnologia flash (elettronica)
    → decine o centinaia di *microsecondi* (almeno 10 volte più veloce vs HDD)
  - Memoria RAM DDR4
    → *nanosecondi* (più di mille volte più veloce vs SSD, un milione di volte più veloce vs HDD)
- Velocità di trasferimento sequenziale
  - HDD magnetico → circa 100 MB/s
  - SSD (bus SATA) → circa 500 MB/s
  - SSD NVMe (bus PCI express) → circa 3000 MB/s
  - RAM DDR4 → circa 20 GB/s

Organizzazione gerarchica della memoria

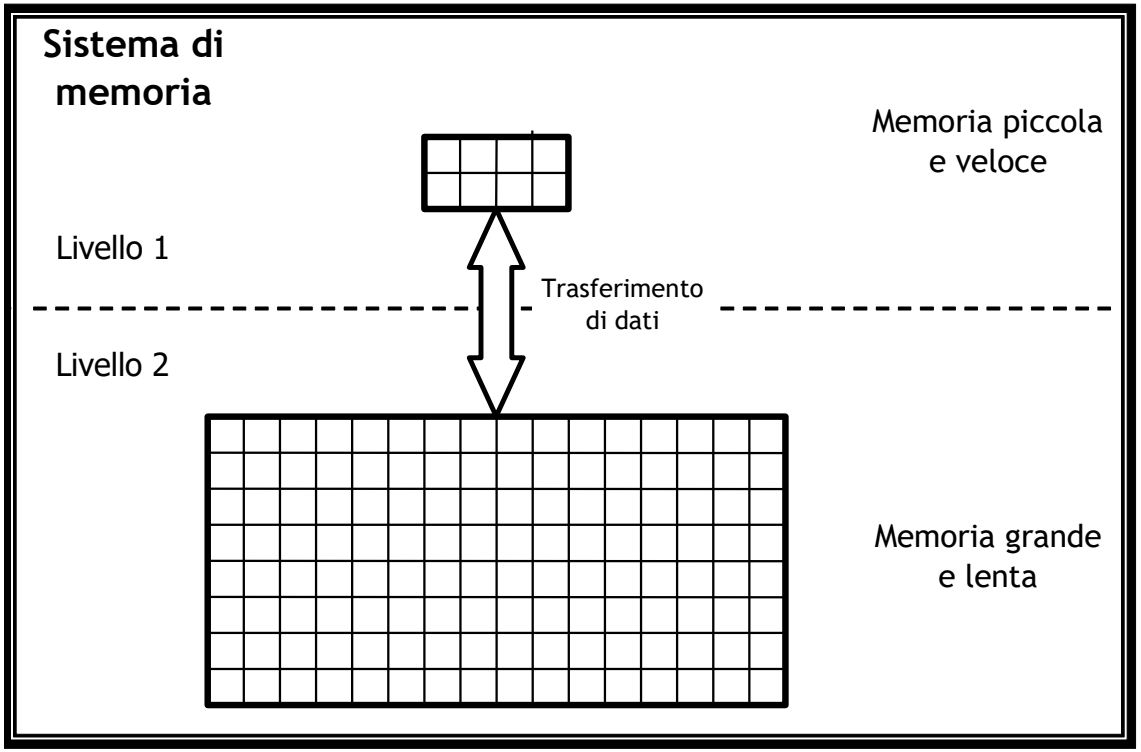
- Le CPU sono sempre state più veloci delle memorie
  - L'aumento di integrazione ha consentito di realizzare CPU sempre più efficienti e veloci
- L'accesso alla memoria passa attraverso il bus
  - La frequenza di funzionamento del bus è molto più bassa di quella della CPU
  - E' possibile fare memorie molto veloci se stanno nel chip della CPU, ma sono piccole e costose

Gerarchia a più livelli



Organizzazione gerarchica della memoria

- Le diverse tecnologie disponibili sono combinate in una gerarchia di memorie, in modo da ottimizzare le prestazioni complessive
- Esempio semplificato con due livelli:
  - L'obiettivo è quello di combinare una memoria piccola e veloce (MPV) con una memoria grande e lenta (MGL) per ottenere una memoria grande e veloce



Organizzazione gerarchica della memoria

- Organizziamo la memoria in blocchi di dimensioni prefissate
  - Quando la CPU deve leggere o scrivere una parola
    - Viene controllato se il blocco contenente la parola si trova nella memoria veloce
    - Solo se necessario il blocco viene trasferito dalla memoria lenta a quella veloce

- Questa tecnica ha successo solo se abbiamo un'alta probabilità di trovare i dati di cui abbiamo bisogno nella memoria piccola e veloce
  - La frequenza con cui si trova il dato nella MPV è detta *hit rate*
    - In questi casi il tempo di accesso è detto *hit time* ed è quello della memoria veloce
  - La frequenza con cui è necessario accedere alla MGL è detta *miss rate*
    - In questi casi il tempo di accesso è detto *miss penalty* ed è quello della memoria lenta
  - Il tempo medio di accesso è dato da:
    hit rate * hit time + miss rate * miss penalty
  - Se la hit rate è alta (es. 99%), il tempo medio di accesso tende all'hit time

Principio di località

Il principio di *località* permette di ottenere un hit rate elevato

- Il comportamento medio dei programmi in termini di accesso alla memoria è facilmente prevedibile

Più specificatamente si parla di:

- Località *spaziale*
  - Quando si accede all'indirizzo X, è molto probabile che gli accessi successivi richiedano celle vicine ad X
    - Le istruzioni vengono in genere lette in modo sequenziale da locazioni consecutive della memoria
    - I dati sono organizzati in strutture e "vicini" tra loro
- Località *temporale*
  - Quando si accede all'indirizzo X, è molto probabile negli accessi successivi si richieda di nuovo la cella X
    - Cicli di istruzioni accedono ripetutamente alle stesse locazioni di memoria
    - Istruzioni vicine tendono ad utilizzare le stesse variabili

Principio di località

Come si sfrutta la località:

- Località spaziale
  - Quando devo caricare un dato in MPV, carico anche i dati vicini (blocco)
- Località temporale
  - Conservo i dati in MPV il più possibile
  - Se devo rimpiazzare un blocco (la memoria MPV è piena), sostituisco il blocco usato meno recentemente (least recently used – LRU)

Due esempi con unità di misura generiche

- Ipotesi di partenza
  - Hit time = 1 unità
  - Miss penalty = 10 unità

- Esempio 1 – hit rate 10% → 1/10
  - Il tempo medio di accesso sarà 0.1*1 + 0.9*10 = 9.1 unità

- Esempio 2 – hit rate 99% → 99 /100
  - Il tempo medio di accesso sarà 0.99*1 + 0.01*10 = 1.09 unità