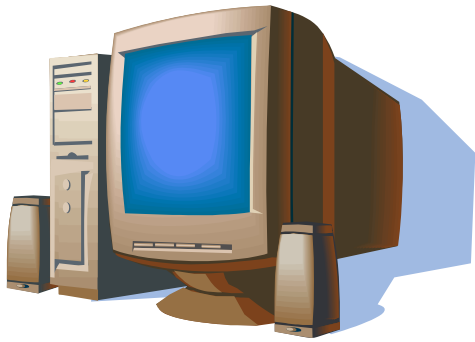
# Programma del corso

- *Introduzione agli algoritmi*
- *Rappresentazione delle Informazioni*
- ***Architettura del calcolatore***
- *Elementi di Programmazione*

# Cos'è un Calcolatore?

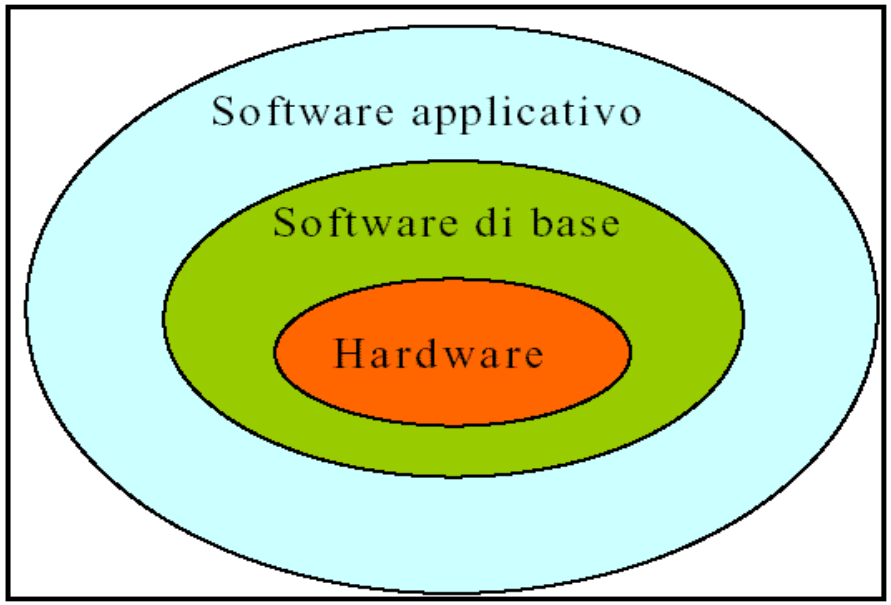
- Un **computer** (calcolatore) è una macchina in grado di accettare informazioni provenienti dall'esterno, di effettuare su di esse operazioni aritmetiche e logiche e quindi di fornire risultati in forma comprensibile



- Per svolgere ciascuna di queste funzioni possiede dei **dispositivi idonei**

# Architettura del calcolatore

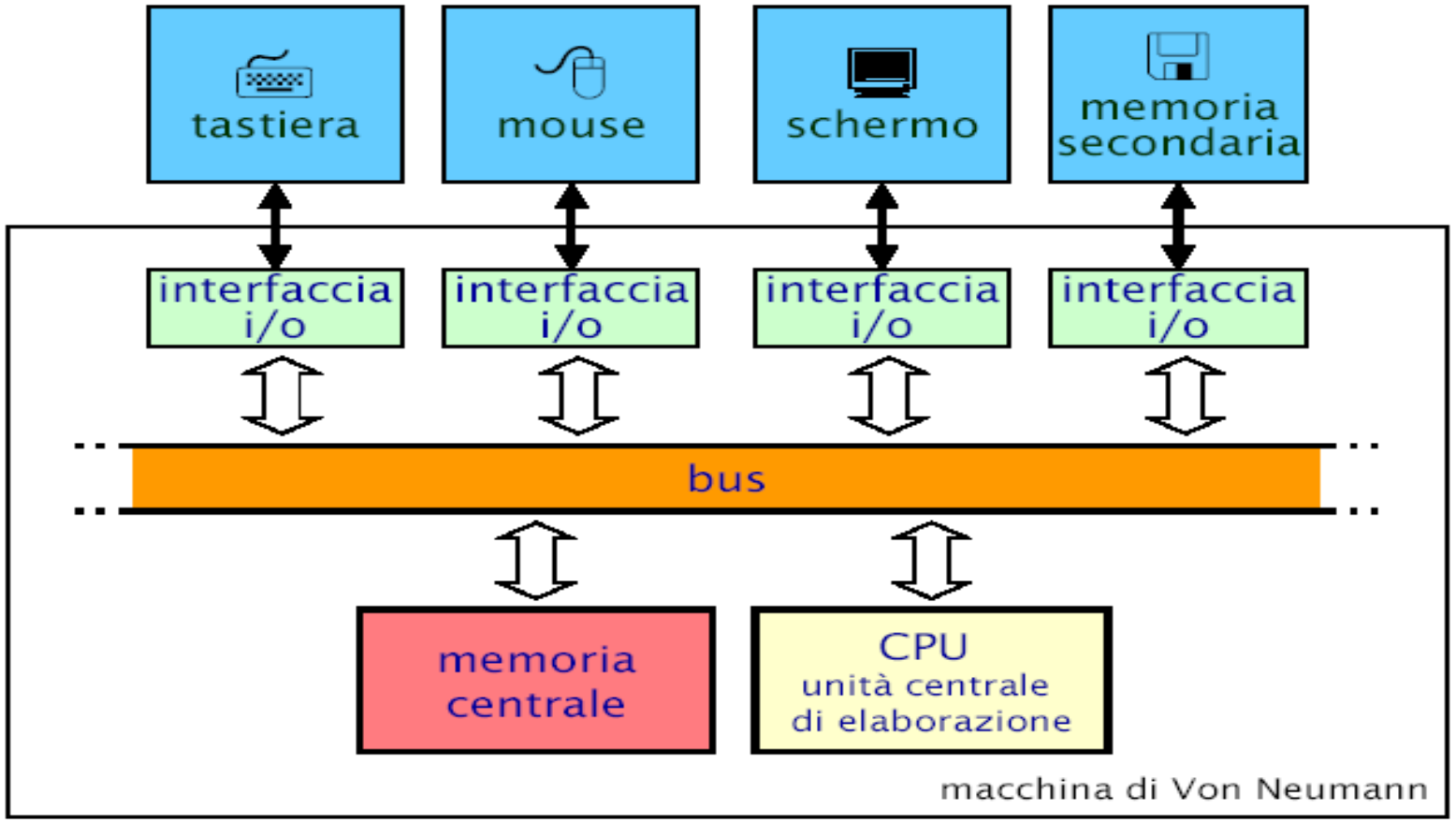
- La prima decomposizione di un calcolatore è relativa a due macro-componenti:
  - **Hardware**
  - **Software**



# Architettura del calcolatore

- L'architettura dell'**hardware** di un calcolatore reale è molto complessa
- La **macchina di von Neumann** è un modello semplificato dei calcolatori moderni
  - **John von Neumann,** matematico ungherese, progettò, verso il 1945, il primo calcolatore con programmi memorizzabili anziché codificati mediante cavi e interruttori

# Macchina di Von Neumann



# Macchina di Von Neumann

E' composta da 4 tipologie di componenti funzionali:

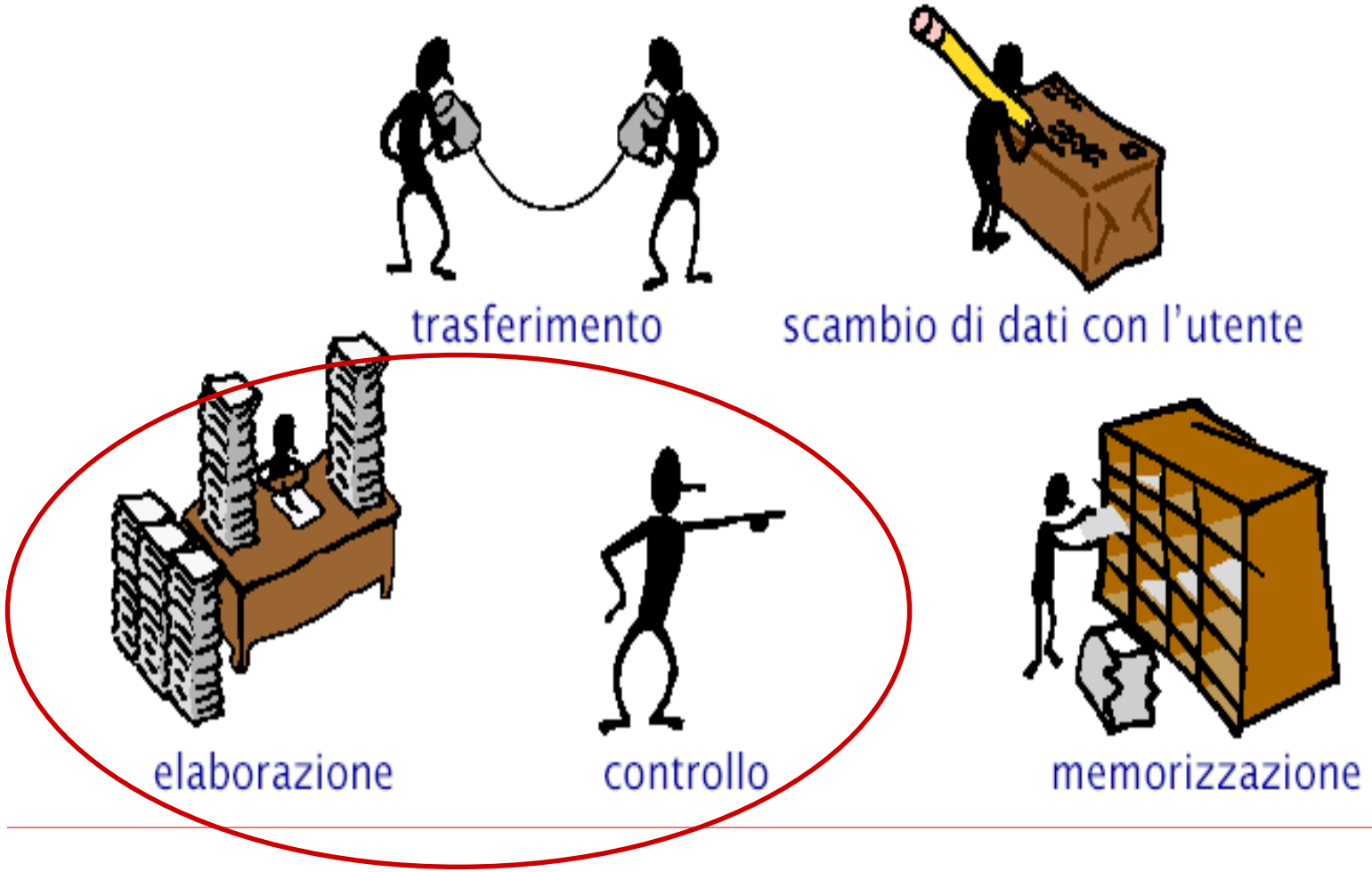
- **Unità centrale di elaborazione** (**CPU**)
  - esegue istruzioni per l'elaborazione dei dati
  - esegue operazioni aritmetiche o logiche (**ALU**)
  - svolge anche funzioni di controllo
- **Memoria centrale (RAM)**
  - memorizza e fornisce l'accesso a dati e programmi
- **Interfacce di ingresso e uscita**
  - componenti di collegamento con le periferiche del calcolatore
- **Bus**
  - svolge la funzionalità di trasferimento di dati e di informazioni di controllo tra le varie componenti funzionali

# Macchina di Von Neumann

Il funzionamento di un calcolatore è descrivibile in termini di poche componenti **(macro-unità)** funzionali

- ogni macro-unità è specializzata nello svolgimento di **una tipologia omogenea** di funzionalità
- ***Eccezione***: **l'Unità Centrale di Elaborazione (CPU)**, che svolge sia funzionalità di elaborazione che di controllo

# Macchina di Von Neumann




Svolto dalla Unità Centrale di Elaborazione (CPU)

# Elaborazione

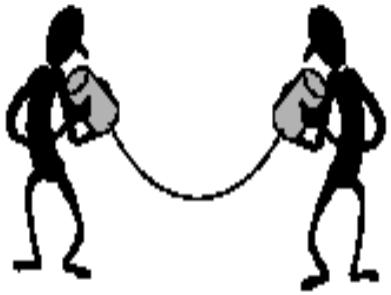
- Un calcolatore sa svolgere poche tipologie di operazioni elementari ma in modo **molto efficiente**
  - un calcolatore può eseguire centinaia di milioni di istruzioni al secondo

- L'elaborazione dei **dati** viene svolta dall'**unità aritmetico-logica (ALU)**, che è un componente dell'unità centrale di elaborazione (**CPU** – **Central Processing Unit**)

# Elaborazione

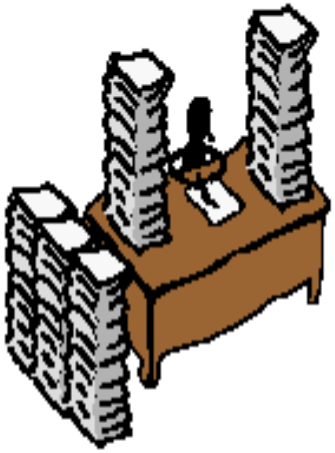
- Le **istruzioni** di un programma corrispondono ad operazioni elementari di elaborazione
  - operazioni aritmetiche
  - operazioni relazionali (confronto tra dati)
  - operazioni su caratteri e valori di verità
  - altre operazioni numeriche

# Macchina di Von Neumann



trasferimento

scambio di dati con l'utente



elaborazione

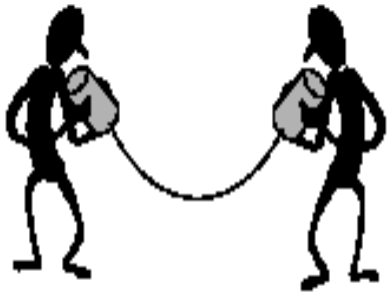
controllo

memorizzazione

# Controllo

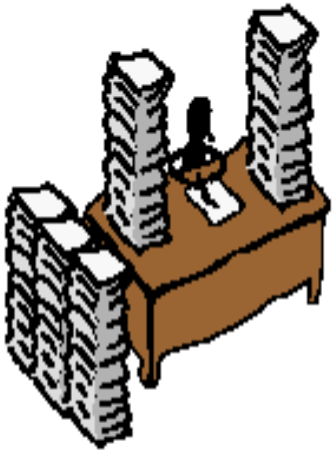
- Il coordinamento tra le varie parti del calcolatore è svolto dall'**Unità di Controllo**
  - è un componente **dell'unità centrale di elaborazione (CPU)**
  - ogni componente dal calcolatore esegue solo le azioni che gli vengono richieste dall'unità di controllo
- Il **controllo** consiste nel coordinamento dell'esecuzione temporale delle operazioni
  - sia internamente all'unità di elaborazione sia negli altri elementi funzionali

# Macchina di Von Neumann



trasferimento

scambio di dati con l'utente



elaborazione

controllo

memorizzazione

# Memorizzazione

- Un calcolatore memorizza
  - i **dati**, che rappresentano informazioni di interesse
  - i **programmi** per l'elaborazione dei dati
- La **memoria** è l'unità responsabile della memorizzazione dei dati
- Una unità di memoria fornisce **due** sole **operazioni**
  - memorizzazione di un valore (**scrittura**)
  - accesso al valore memorizzato (**lettura**)

# 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.

# Indirizzi di memoria

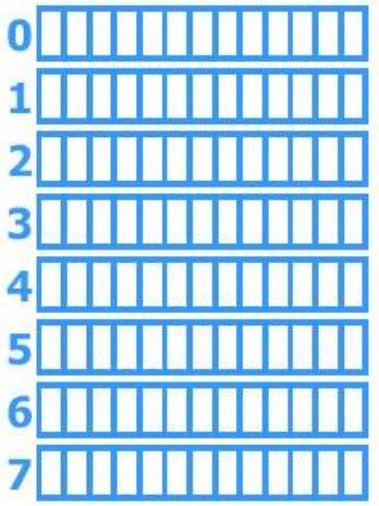
- I bit nelle memorie sono raggruppati in **celle**:
  - tutte le celle sono formate dallo **stesso numero di bit**;
  - una cella composta da **k bit**, è in grado di contenere una qualunque tra **$2^k$ combinazioni** diverse di bit.
- Ogni cella ha un **indirizzo**:
  - serve come accesso all'informazione;
  - in una memoria con **N celle** gli indirizzi vanno da **0** a **N–1**.
- **La cella è l'unità indirizzabile più piccola.**
  In quasi tutti i calcolatori è di **8 bit** (un **byte**).
- I byte vengono raggruppati in **parole** (che oggi sono di **32/64 bit**), su cui la CPU esegue le operazioni.

# Organizzazione della memoria

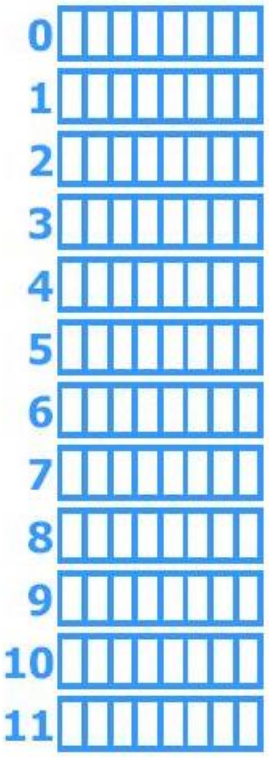
- Anche gli indirizzi della memoria sono rappresentati come numeri binari:
  - un indirizzo di **M** bit consente di indirizzare $\mathbf{2^M}$ celle;
  - per 6 o 8 celle bastano 3 bit, per 12 celle ne servono 4;
  - il **numero di bit nell'indirizzo** determina il **numero massimo di celle indirizzabili** nella memoria ed è indipendente dal numero di bit per cella (una memoria con $2^{12}$ celle richiede sempre 12 bit di indirizzo, quale che sia la dimensione di una cella).
- Una memoria può essere organizzata in diversi modi:
  - con 96 bit possiamo avere 6 celle di 16 bit (6*16=96), o 8 celle di 12 bit (8*12=96) o 12 celle di 8 bit (12*8=96).

# Organizzazione della memoria


6 parole da 16 bit




8 parole da 12 bit




12 parole da 8 bit

# Dispositivi di memorizzazione

**Memorie d'uso**
**Ram** (*Random Access Memory* o Memoria ad accesso casuale
**Rom** (*Read Only Memory* o memoria di sola lettura; si attiva all'accensione del Computer)

**Memorie di Massa**
Hard Disk
Floppy Disk
Cd –rom
Unità di Back-up
Penne USB

# Memoria centrale

# Memoria centrale (o principale)

- E' la componente del calcolatore in cui vengono immagazzinati e da cui vengono acceduti i dati e i programmi (solitamente di tipo **RAM** – Random Access Memory)

- E' la memoria che può essere acceduta direttamente dal processore
  - è costituita da sequenze di **celle** (o **locazioni**)
  - ogni cella può contenere una quantità fissata di memoria (numero di bit), detta **parola** di memoria

# Altre informazioni sulla RAM: la PAROLA o WORD

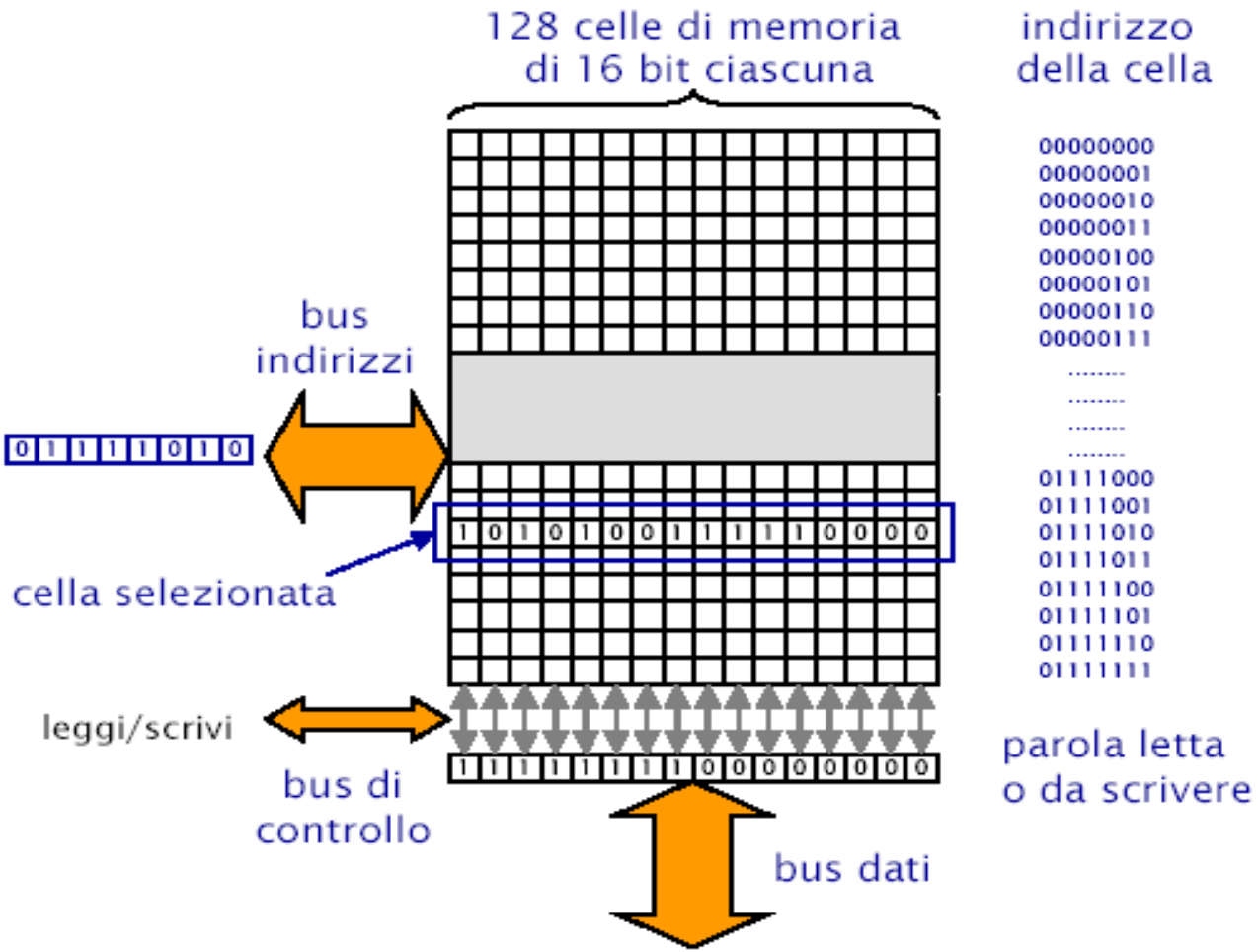
- La **parola (word)** di un computer: quanti bit possono essere letti/scritti/usati dalla CPU con un unico accesso alla memoria (16, 32, 64, 128 bit)

- Tra le altre caratteristiche, più grande è la **parola**, maggiore è la "potenza" del computer

# Memoria centrale

- Ogni cella è caratterizzata da
  - un **indirizzo**, che è un numero che identifica la cella e ne consente l'accesso
  - un **valore**, che è la sequenza di bit memorizzata dalla cella
- La memoria fornisce le operazioni di
  - **lettura**: consultazione del valore di una cella con un dato indirizzo
  - **scrittura**: modifica del valore di una cella con un dato indirizzo

# Struttura della RAM



# Dimensioni della RAM

- **Spazio di indirizzamento**: insieme o numero delle celle indirizzabili direttamente

- Il numero di celle indirizzabili e' una potenza di due. Con:
  - 16 bit si indirizzano $2^{16}$ celle = 65.536 celle
  - 32 bit si indirizzano $2^{32}$ = 4.294.967.296 celle
  - …..

# Dimensioni tipiche della RAM

- Nei Personal Computer:
  - 1 GByte, 2 Gbyte, etc
  - **una volta era un lusso avere 64 KB**
- Nei Mainframe/Workstations:
  - 4, 8, 16... GByte
- Ricordatevi che la memoria e' espandibile (fino ad un certo limite)

# Proprietà della RAM

- La RAM e' **veloce**
  - per leggere/scrivere una cella ci vogliono, in media 5--30 nanosecondi (millesimi di milionesimi di secondo = $30 * 10^{-9}$ s)
- la RAM e' **volatile**
  - e' fatta di componenti elettronici, e se togliete l'alimentazione perdete tutto
- La RAM e' **costosa** (relativamente)

# Memorie ROM

- Le memorie **ROM** (read only memory)
  - permettono **solo** la **lettura** dei dati
  - sono **persistenti** (mantengono il suo contenuto anche quando non c'è alimentazione)
  - in questa memoria si trovano i programmi che servono per l'avvio della macchina, i cosiddetti programmi di sistema e il **BIOS** (Basic Input Output System) sistema di base per il controllo di entrata ed uscita (cioè il **FIRMWARE**)

# Memorie secondarie

- Dette anche **Memoria di massa**
  - memorizza ***grandi masse*** di dati
  - i dati memorizzati "sopravvivono" all'esecuzione dei programmi
  - **non può** essere acceduta direttamente dalla CPU
    - i dati di una memoria secondaria per essere elaborati dal processore devono passare nella memoria centrale

# Caratteristiche delle memorie secondarie

- **non volatilità**
  - i dati memorizzati non si perdono allo spegnimento del calcolatore (perché memorizzati in forma magnetica o ottica anziché elettronica)
- **grande capacità**
  - capacità maggiore (anche di diversi ordini di grandezza) rispetto alla memoria centrale
- **bassi costi**
  - il costo per bit di una memoria secondaria è minore (di diversi ordini di grandezza) rispetto alla memoria centrale
- **bassa velocità di accesso**
  - tempi di accesso maggiori (di qualche ordine di grandezza) rispetto a quelli della memoria principale

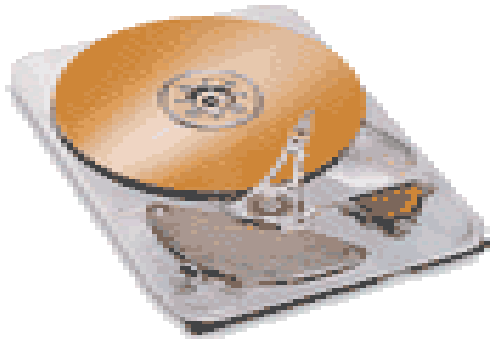
# La memoria secondaria

- Programmi e dati risiedono normalmente in memoria secondaria

- Quando si lancia un programma questo viene copiato dalla memoria secondaria in memoria primaria. Questa operazione si chiama **caricamento**

# Dischi magnetici: l'HARD DISK

- E' fatto di supporti magnetici permanenti, gestiti mediante dispositivi meccanici
- Tempi di accesso dell'ordine dei micro/millisecondi
- Spazio disponibile:
  - 80, 20, 160, …, 500 Gigabyte
    **(una volta era un lusso avere 10 Megabyte)**

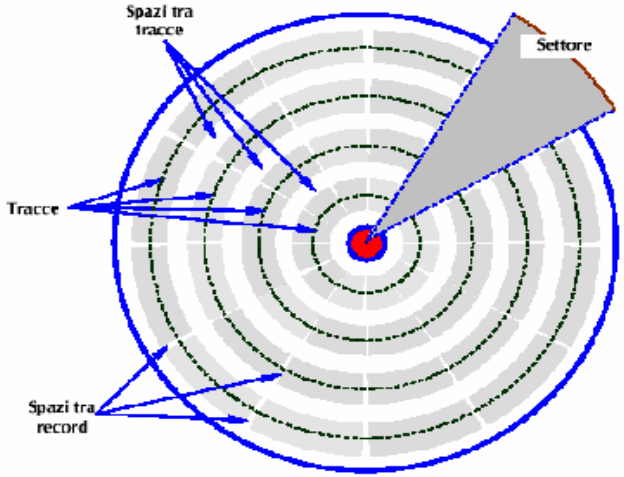


# Dischi magnetici: l'HARD DISK

- Nell'hard disk la memoria e' organizzata in blocchi di dimensione fissa (512B, 1KB,2KB,..) **indirizzabili direttamente**
- La lettura/scrittura del disco avviene sempre in blocchi, per risparmiare tempo (pensate al tempo perso se si dovesse leggere un byte per volta!)
- Il disco e' quindi **formattato** in blocchi

# Dischi magnetici: l'HARD DISK

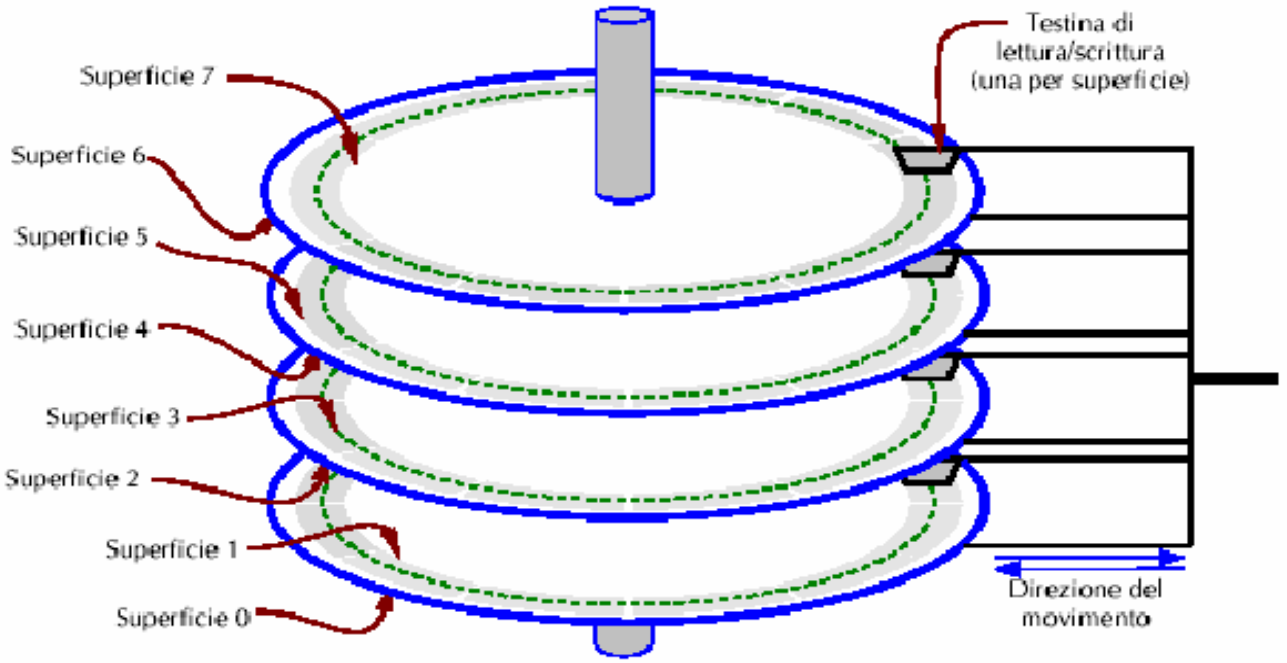
- Un disco consiste in un certo numero di ***piatti*** con due superfici che ruotano attorno ad un perno centrale
  - ogni superficie dispone di una propria testina di lettura / scrittura




- Le superfici sono organizzate in cerchi concentrici (***tracce***) e in spicchi di ugual grandezza (***settori***)
  - un bit corrisponde ad uno stato di polarizzazione (positiva o negativa) del materiale magnetico che costituisce i dischi

# Dischi magnetici: l'HARD DISK

Le tracce equidistanti dal centro formano un **cilindro**.



# Memoria primaria vs memoria secondaria

| RAM | HARD DISK |
| --- | --- |
| veloce (nanosec) | lenta(microsec) |
| piccola (pochi Gigabyte) | grande (pochi Terabyte) |
| volatile | permanente |

Notate che, in teoria, il computer potrebbe funzionare con la sola ram o il solo hard disk

# Altre memorie secondarie…

# Dischi magnetici: floppy disk

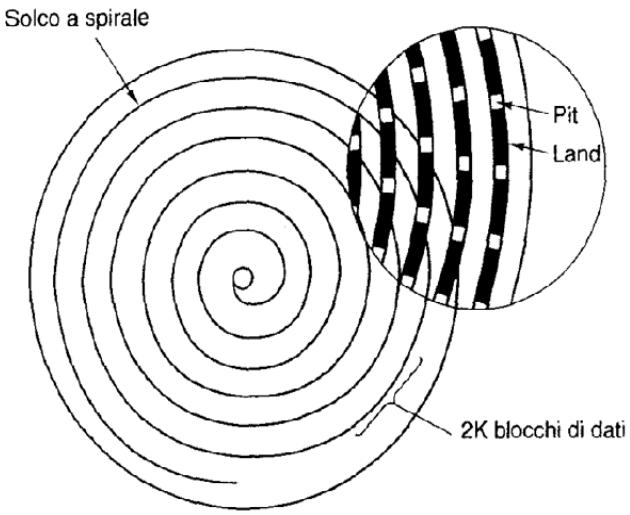
- Sono dischi magnetici di piccola capacità, portatili, usati per trasferire informazioni (file) tra computer diversi.
- Sono costituiti da un unico disco con due superfici.
- Storicamente ne sono stati creati vari tipi identificati dal loro diametro (3.5, 5.25 e 8 pollici).
  - oggi sopravvivono solo dischetti da 3.5” (1.4 Mbyte)

# Dischi ottici

- La superficie di un disco presenta una successione di tratti disposti secondo un'unica traccia a spirale
  - **pit**: tratto di superficie avvallata
  - **land**: tratto di superficie liscia

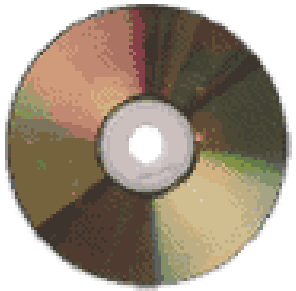
  } **riflettono raggi luminosi in modo diverso**

- Il passaggio da pit a land (e viceversa) rappresenta 1 mentre l'assenza di variazione rappresenta 0



# Dispositivi ottici

- **CD-ROM** (Compact Disk): sono esattamente gli stessi CD usati per la musica
- la sigla ROM (Read Only Memory) indica il fatto che i dati, una volta scritti su CD, sono indelebili e potranno essere soltanto letti
- la capacità tipica è di 650 MByte (che nei CD audio corrisponde a 74 minuti di registrazione), ma esistono anche modelli leggermente più capienti.



# Dispositivi ottici

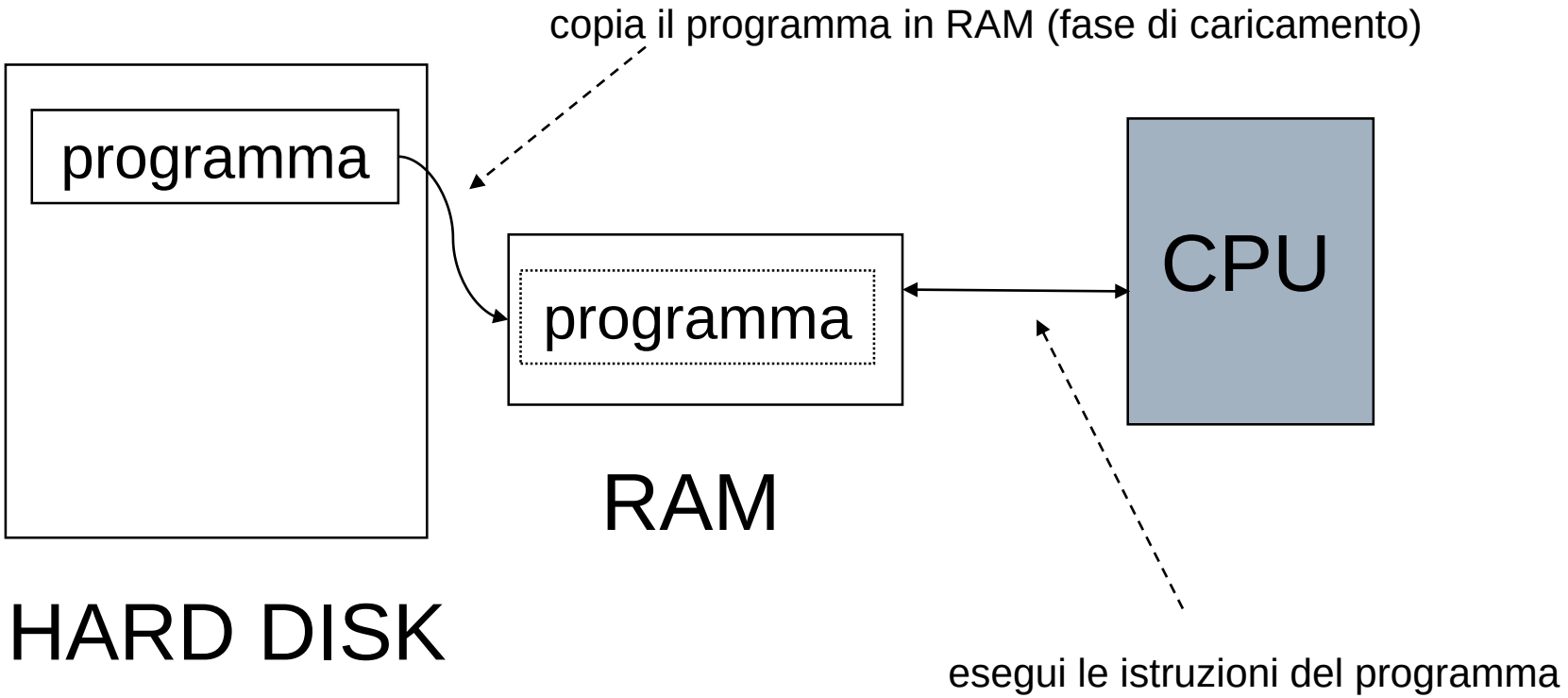
- **DVD (Digital Versatile Disk)**: Esteriormente sono in tutto simili ai CD-ROM, ma possono contenere da 9 a 17 GByte (cioè fino a 25 volte la capacità di un normale CD).
- Sono usati da alcuni anni soprattutto per i film digitali, tuttavia possono benissimo contenere anche i normali dati come i CD-ROM.
- Per leggere i DVD occorre un lettore CD appropriato (i normali drive per CD non sono in grado di farlo). Il lettore DVD è invece sempre in grado di leggere anche i normali CD-ROM.

# Il processore - CPU
# (CENTRAL PROCESSING UNIT)

# … abbiamo visto

- **Programmi** e **dati** risiedono in file memorizzati in memoria secondaria
- Per essere eseguiti (i programmi) e usati (i dati) vengono copiati nella memoria primaria
- La **CPU** e’ in grado di eseguire le istruzioni di cui sono composti i programmi

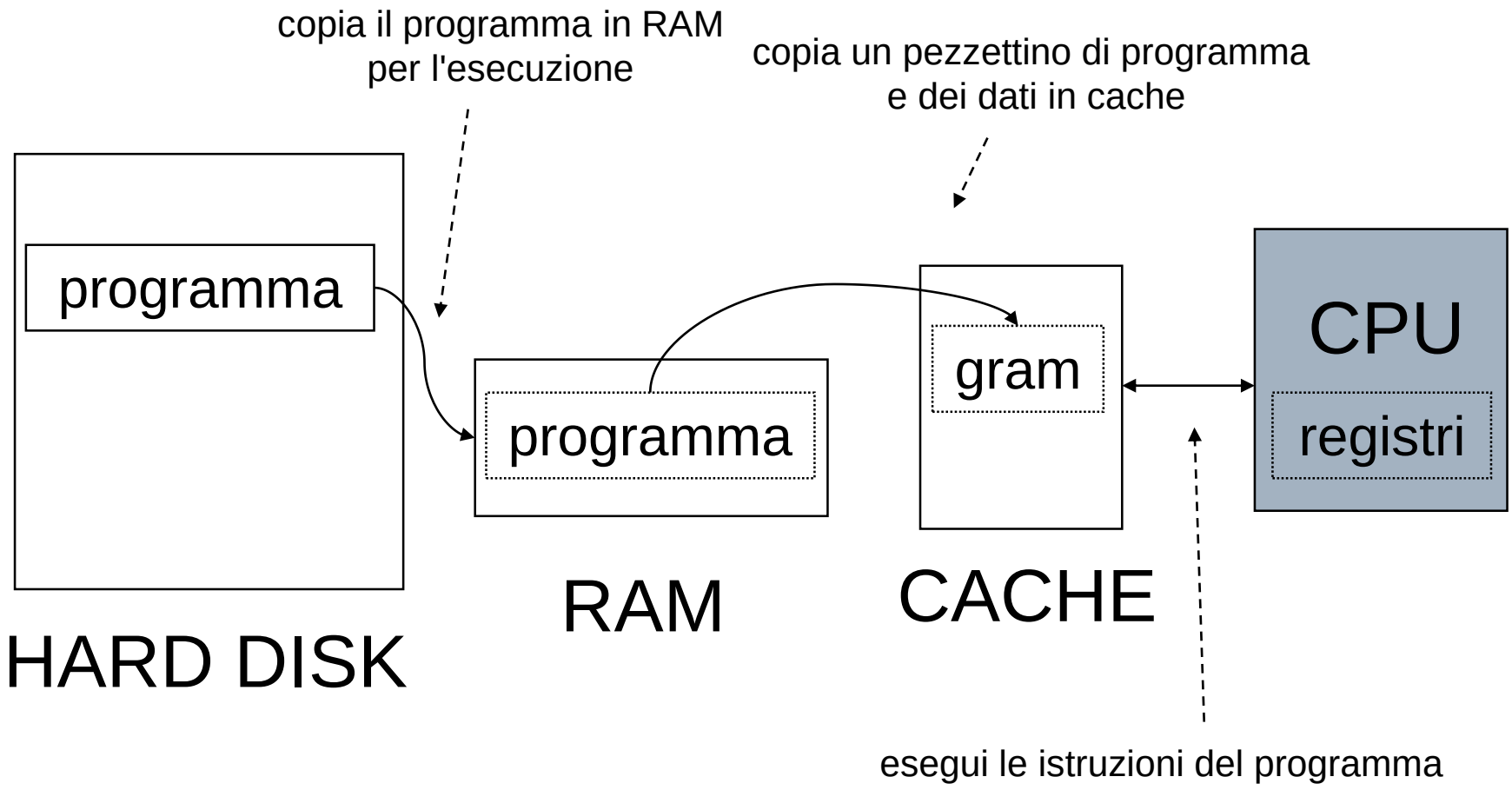
# Funzionamento



# MEMORIA CACHE

- Livello di **memoria intermedio** tra i registri e la ram
- Per memorizzare i dati usati piu' spesso senza doverli recuperare in memoria
- Valori tipici: 512KB, 1MB, 2MB
- **Interna** o **esterna** alla CPU
- Influisce moltissimo sulle prestazioni e il costo della CPU (e quindi del computer)
- I computer attuali hanno spesso **più livelli** di cache (Es: L1, L2, L3)

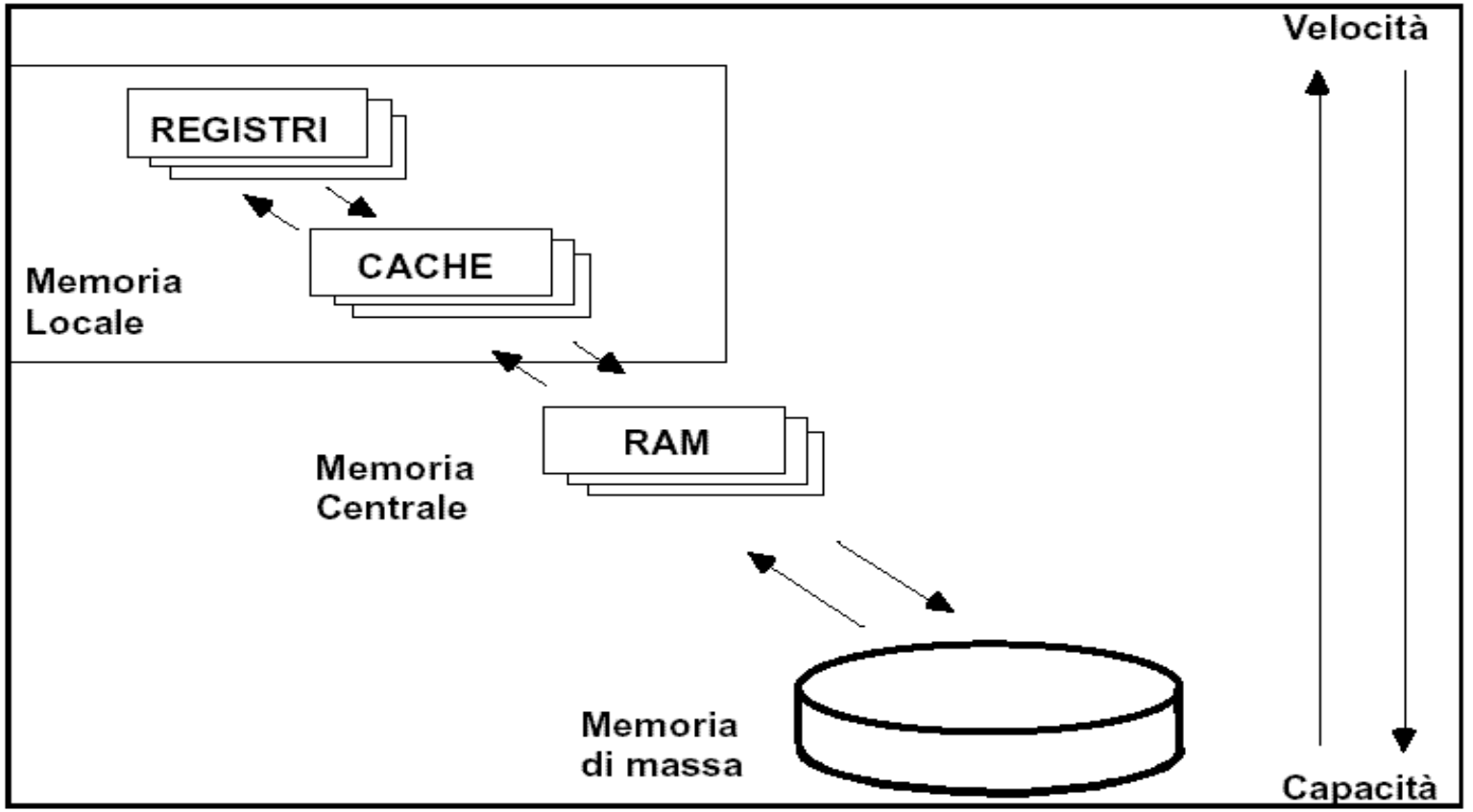
# MEMORIA CACHE



# Memorie di un computer

| | | Tempi di accesso |
|---|---|---|
| **Registri** | < 1 KByte | 100 * picosecondi |
| **Cache** | < 4 MByte | nanosecondi |
| **RAM** | < 16 Gbyte | 10 * nanosec |
| **Hard disk** | > 200 GByte | 10 * microsec |
| **Dischi ottici** | 650MB-17GB | micro/millisecondi |
| **Nastri** | > 10 GByte | 10 * millisecondi |

# Gerarchia delle memorie



# Caratteristiche dei microprocessori

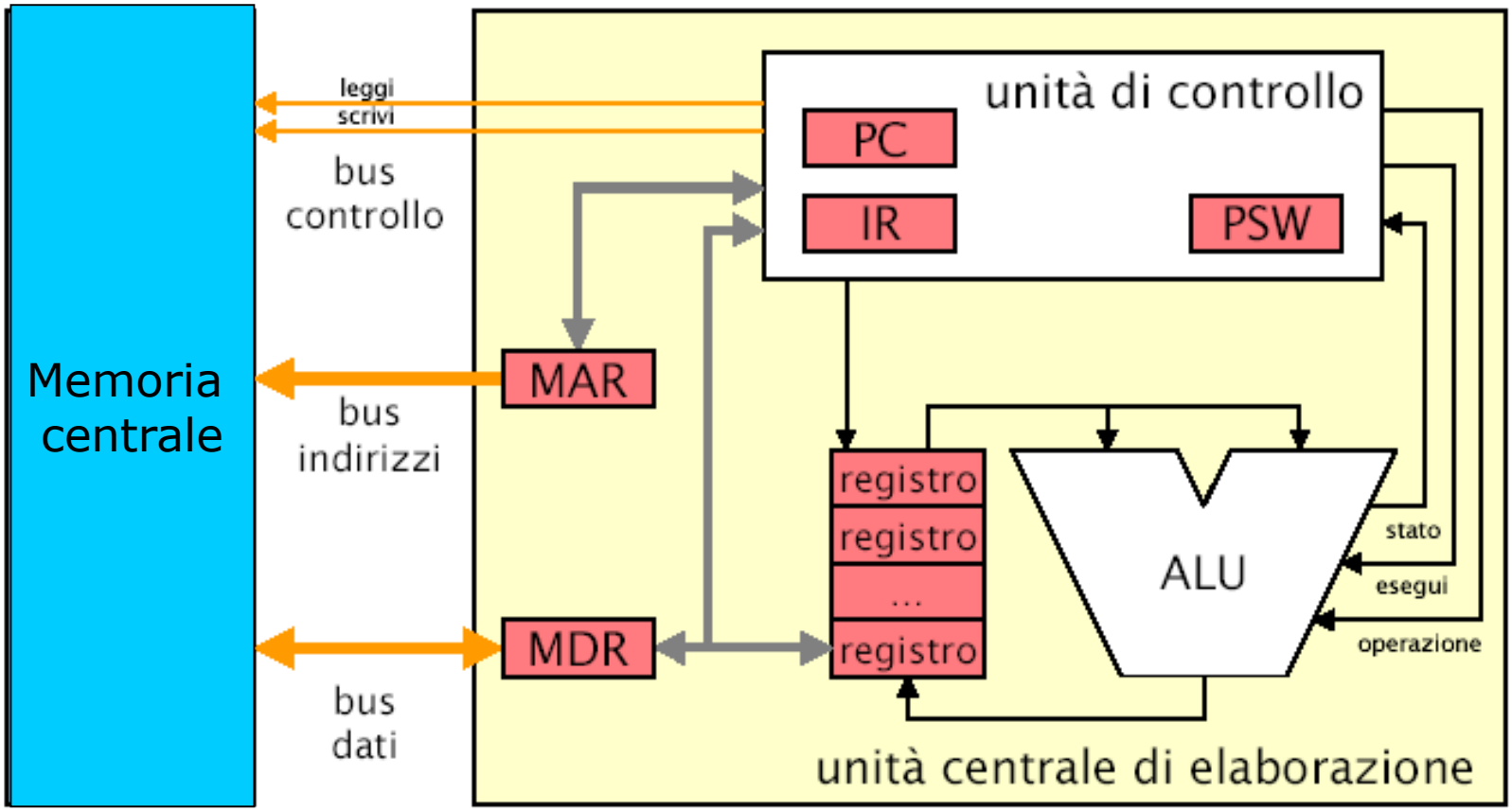
- **Repertorio di istruzioni**
  - L'insieme delle istruzioni che costituiscono il linguaggio macchina del processore
- **Frequenza di clock**
  - l'esecuzione di una istruzione può richiedere più cicli macchina
- **Ampiezza del bus**
  - numero di bit nel bus interno del processore
- **Co-processori**
  - processori specializzati per operazioni complesse (*es*: co-processore matematico)
- **Memoria cache**
  - una memoria veloce locale al processore, che consente di accedere più velocemente ai dati da elaborare

# Frequenza di clock

La frequenza con cui si eseguono i cicli di esecuzione è scandita dal **clock** (orologio interno)

- ad ogni impulso di clock l'unità di controllo esegue un ciclo di esecuzione (**ciclo macchina**, v. dopo)
- la velocità di elaborazione di un microprocessore dipende dalla frequenza del suo clock (1, 1.5,2.2, ... GHz) (es.: 1GHz = 1 miliardo di cicli al secondo). Attualmente si parla solo di GHz

# Struttura del processore (CPU)



# Le componenti della CPU: I REGISTRI

- Sono piccole unita' di memoria (2, 4, 8 byte) con tempi di accesso molto piu' bassi delle celle della memoria primaria
- Ospitano le informazioni necessarie per eseguire l'istruzione corrente
- In numero molto limitato (10, 20, 64) si dividono in registri **speciali** e **generali**

# Le componenti della CPU: I REGISTRI SPECIALI

- **Il Program Counter (PC)**
  - Contiene l'indirizzo in memoria centrale della **prossima istruzione** da eseguire
  - All'inizio dell'esecuzione di un programma viene caricato con l'indirizzo della prima istruzione di quel programma
  - Ad ogni istruzione eseguita il PC viene modificato per contenere l'indirizzo della istruzione successiva

# Le componenti della CPU: I REGISTRI SPECIALI

- **L'Instruction Register (IR)**
  - Contiene l'istruzione correntemente in esecuzione
  - La CPU legge l'istruzione contenuta nell'instruction register e la esegue

- **IL Registro di stato (PSW)**
  - Descrive lo stato corrente della esecuzione
  - Segnala eventuali errori (ad es.: overflow)

# Le componenti della CPU: I REGISTRI SPECIALI

- **Registro Indirizzi Memoria (MAR)**
  - contiene l'indirizzo della cella da cui leggere o in cui scrivere un dato
- **Registro Dati Memoria (MDR)**
  - contiene il dato letto dalla memoria o da scrivere in memoria

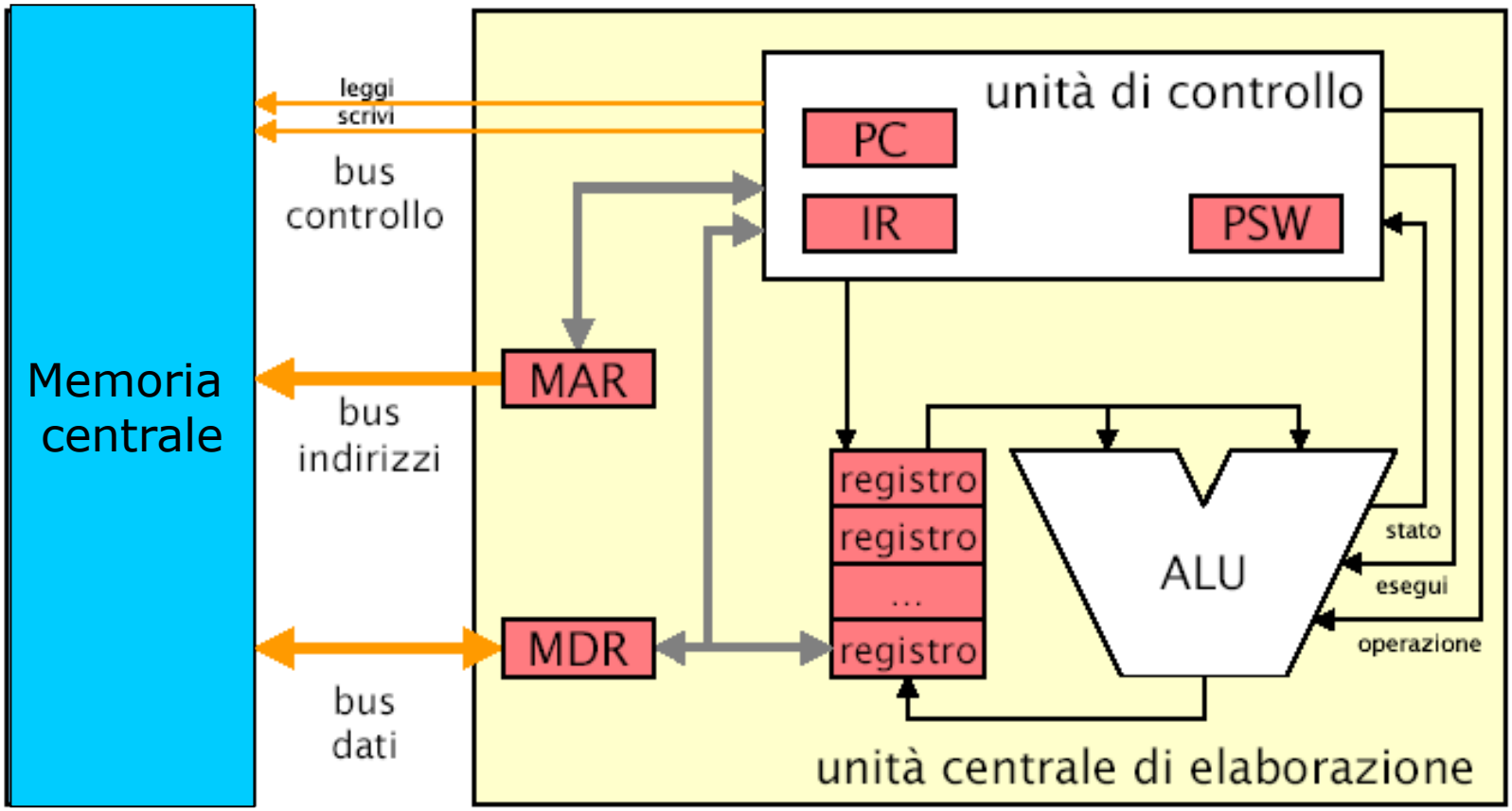
# Le componenti della CPU: I REGISTRI GENERALI

- **I registri generali**
  - in numero di 8, 16, 64
  - sono usati come memorie temporanee per contenere gli operandi delle istruzioni e i risultati parziali durante l'esecuzione delle istruzioni

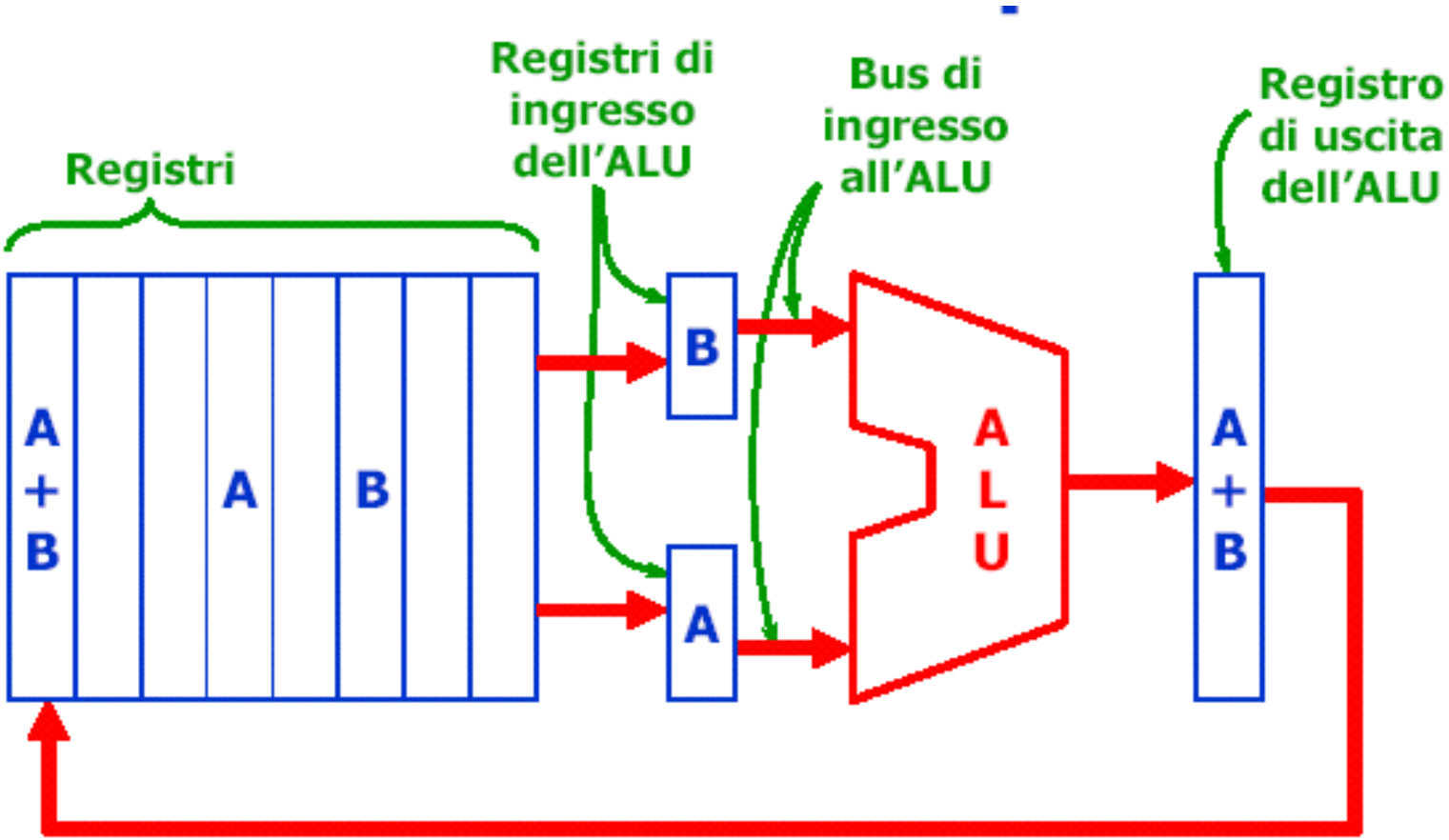
# Le componenti della CPU: ARITHMETIC-LOGIC UNIT (ALU)

- Si occupa di eseguire le operazioni di tipo aritmetico/logico: somme, confronti…
- Preleva gli operandi dai / deposita il risultato delle operazioni nei: registri generali
- A volte e' affiancata da un **co-processore matematico**

# Struttura del processore



# Le componenti della CPU:
# ARITHMETIC-LOGIC UNIT (ALU)



# Linguaggio macchina e assembler

- **Linguaggio macchina** :

  linguaggio comprensibile direttamente dal processore della macchina (binario)

- **Assembler** :

  versione simbolica del linguaggio macchina in cui i nomi delle operazioni e degli operandi sono indicati con codici simbolici (Es: ADD=Somma; LOAD=Carica in memoria, etc)

- **Assemblatori** :

  programmi che traducono il codice simbolico in sequenze di 0 e 1

# Il linguaggio macchina e la codifica delle istruzioni assembler

Si vuole scrivere un programma che sommi la paga base e la paga straordinaria di un impiegato per calcolare la paga lorda

| LINGUAGGUO MACCHINA | LING. ASSEMBLER | ALTO LIVELLO (es: Visual Basic) |
|---|---|---|
| +1300042774<br>+140593419<br>+1200274027 | `LOAD  pagabase`<br>`ADD  straordinario`<br>`STORE pagalorda` | `pagaLorda = pagaBase +`<br>`Straordinario` |

Esempio di scrittura di uno **stesso** spezzone di programma in diversi linguaggi di programmazione: Linguaggio macchina, Linguaggio Assembler, Visual Basic

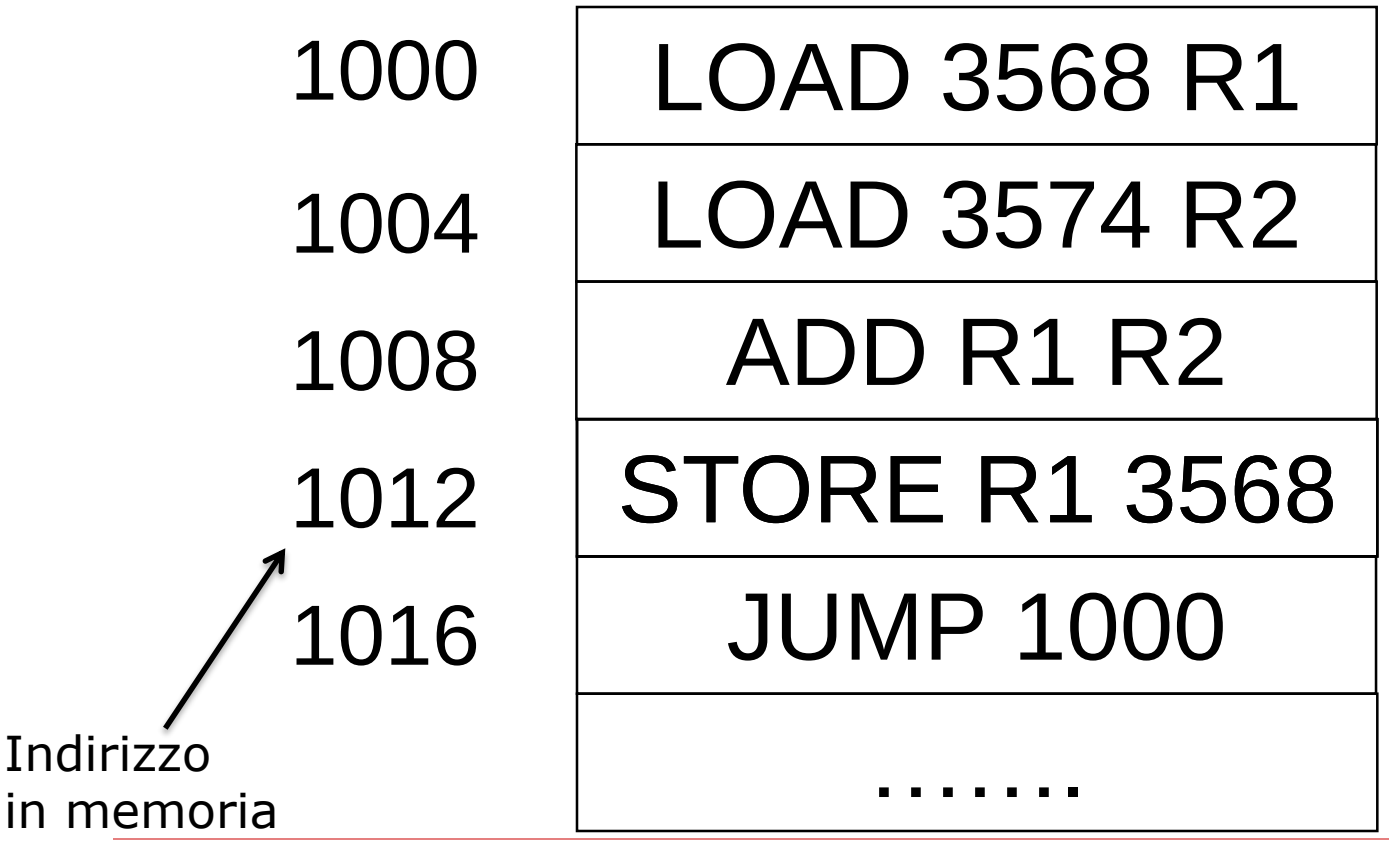
# Il processore - CPU (CENTRAL PROCESSING UNIT)

- Si occupa di **eseguire** i programmi
- I programmi che la CPU è in grado di eseguire sono scritti in **linguaggio macchina**
- **Istruzioni macchina**:

| **Codice istruzione** | **argom. 1** | **argom. 2** |
|---|---|---|

- 16 o 32 bit di lunghezza
- gli argomenti possono mancare

# Un programma in linguaggio macchina (ASSEMBLER)



# Il set di istruzioni macchina

- Ogni tipo di processore e' in grado di eseguire un numero limitato (40/100) di istruzioni

- Combinando in modo diverso sequenze anche molto lunghe di istruzioni (i programmi) si possono far fare al computer tantissime cose completamente diverse

# L’elaborazione delle informazioni: le istruzioni della CPU

- Alcune tipi di istruzioni

| | |
|---|---|
| Istruzioni aritmetiche | Istruzioni di trasferimento dati |
| Istruzioni di confronto | Istruzioni di salto |

- Il funzionamento della CPU è determinato dall’insieme delle istruzioni che essa esegue

# Istruzioni di trasferimento dati

Operazioni che spostano dei dati da una posizione all'altra

Cella di memoria → Registro
Registro → Cella di memoria
Cella di memoria → Cella di memoria
Registro → Registro

Esempi:
Carica il Registro R con il contenuto della cella di memoria X
Memorizza il contenuto del registro R nella cella di memoria X
Copia il contenuto della cella di memoria X nella cella di memoria Y

# Operazioni aritmetiche

Innescano un calcolo da parte dell'unità aritmetico logica . comprendo operazioni aritmetiche + - • /, e operazioni logiche: AND, OR NOT

A secondo del set di istruzioni gli operandi possono risiedere in memoria  o essere in un registro dell'ALU

Esempi

Somma il contenuto della cella X al contenuto della cella X  e metti il risultato nella cella Z

Somma il contenuto della cella X al contenuto della cella Y e metti il risultato nella cella Y

Somma il contenuto della cella X al contenuto del registro R e metti il risultato nel registro R

# Operazioni di confronto

Confrontano due valori e sistemano un indicatore sulla base del risultato del confronto

La maggior parte dei sistemi hanno un registro speciale nel processore dove vengono registrati i bit di operazioni di confronto

Esempio

Confronta il contenuto della cella di memoria X con il contenuto della cella Y
e poni i codici a valori opportuni

# Istruzioni di Salto (controllo del flusso di programma)

Il normale modo di operare ella macchina di von Neumann è sequenziale:

istruzione all'indirizzo i
istruzione all'indirizzo i+1
...

N.B. Se ogni istruzione occupa 1 parola di memoria

Esistono operazioni che possono alterare il normale *flusso di controllo* sequenziale

Esempio
Prendi la prossima istruzione incondizionamente dalla parola di memoria X

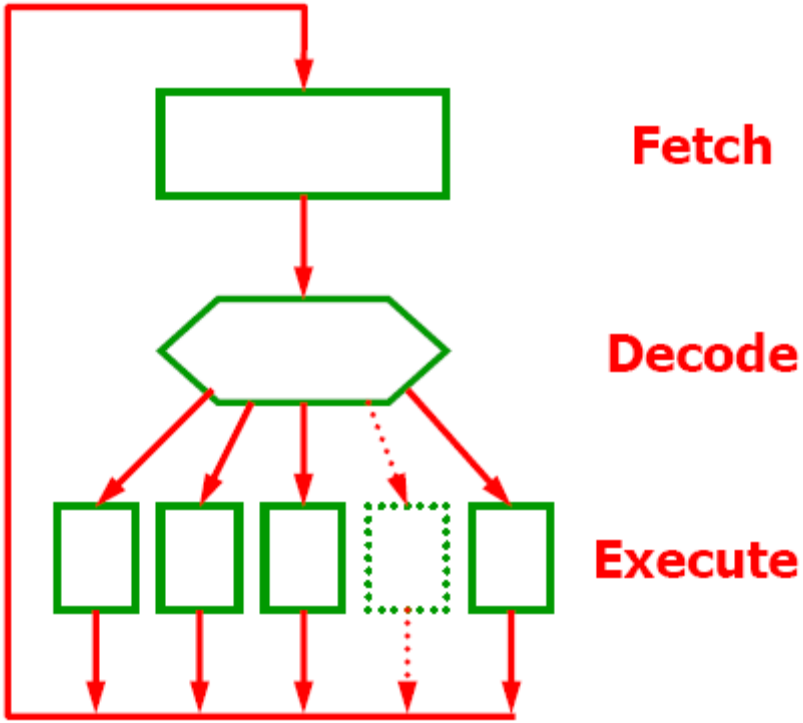
# Le componenti della CPU: LA CONTROL UNIT (CU)

- L'unità di controllo del processore (CPU) esegue una istruzione svolgendo le seguenti **tre operazioni** di base:
  - **Fetch** (lettura)
  - **Decode** (decodifica)
  - **Execute** (esecuzione)
- Un programma è eseguito reiterando il ciclo *fetch-decode-execute* (**ciclo macchina in GHz**) per eseguire ordinatamente le sue istruzioni

# Il ciclo fetch-decode-execute

- La CPU è un dispositivo che opera in modo ciclico ripetendo per ogni istruzione del programma (dalla prima all'ultima):
  - Lettura (fetch): acquisizione della memoria di un'istruzione del programma (e incremento del PC)
  - Decodifica (decode): identificazione del tipo di istruzione da eseguire
  - Esecuzione (execute): esecuzione delle operazioni necessarie per il completamento dell'istruzione

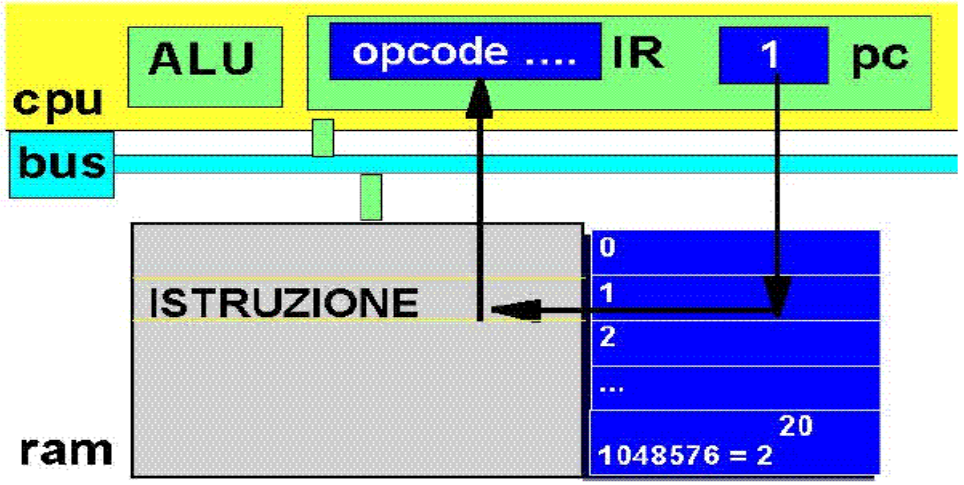
# Il ciclo Fetch-Decode-Execute



# Ciclo *fetch-decode-execute*

## 1) FETCH:

- si accede alla prossima istruzione, riferita dal registro contatore dell'istruzione (PC)
- si porta tale istruzione dalla memoria centrale al Registro Istruzioni (IR)

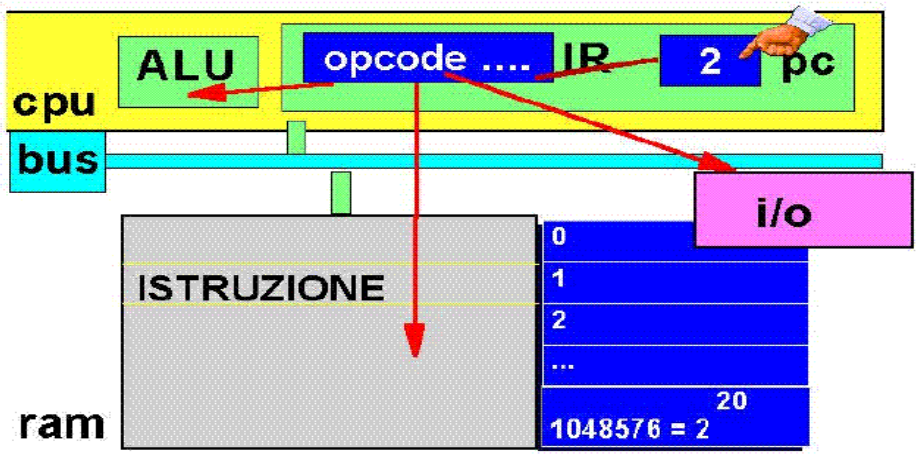


# Ciclo *fetch-decode-execute*

- **2) DECODE**: decodifica dell'istruzione
  - si individua il tipo dell'operazione e gli operandi (dati) usati
  - si trasferiscono i dati nei registri opportuni

# Ciclo *fetch-decode-execute*

- **3) EXECUTE**: esecuzione dell'istruzione
  - si incrementa il registro contatore dell'istruzione (PC)
  - ciascuna azione viene richiesta al componente opportuno



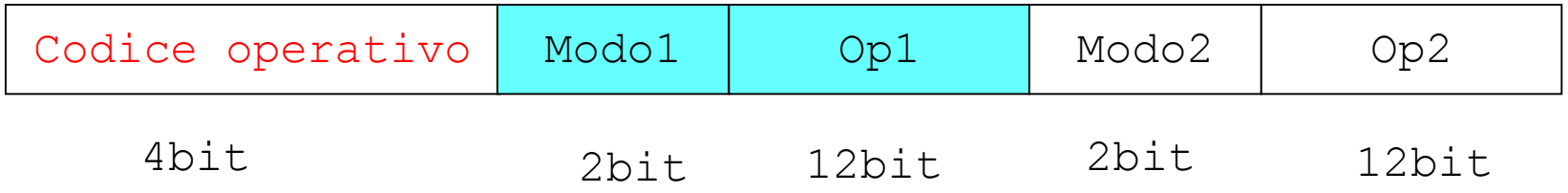
# Il linguaggio macchina e la codifica delle istruzioni assembler

1° A ciascuna istruzione macchina viene associato un intero senza segno (codice operativo)

2° L'istruzione può coinvolgere uno o più operandi

3° Tipicamente la codifica di una istruzione e' lunga come una parola o multipli della parola per poterla leggere dalla memoria in modo più efficiente :

-es : con parole di 4 byte (32 bit)

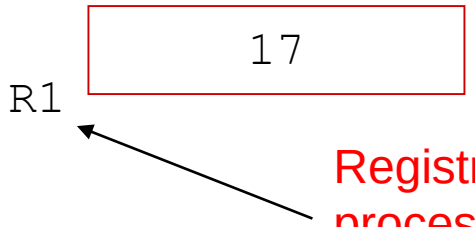


# Esempio: come si realizza in assembler l'operazione *x=y+2*

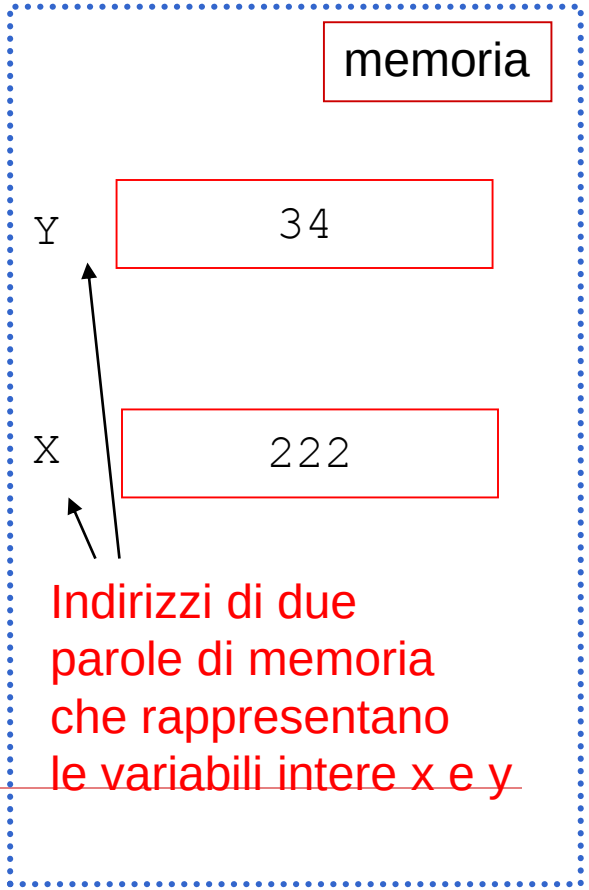
assembler

```
LOAD   Y, R1
ADD     2, R1
STORE  R1, X
```

Legge il valore in Y e lo scrive in R1




Registro interno del processore (variabile temporanea su cui lavorare)




Indirizzi di due parole di memoria che rappresentano le variabili intere x e y

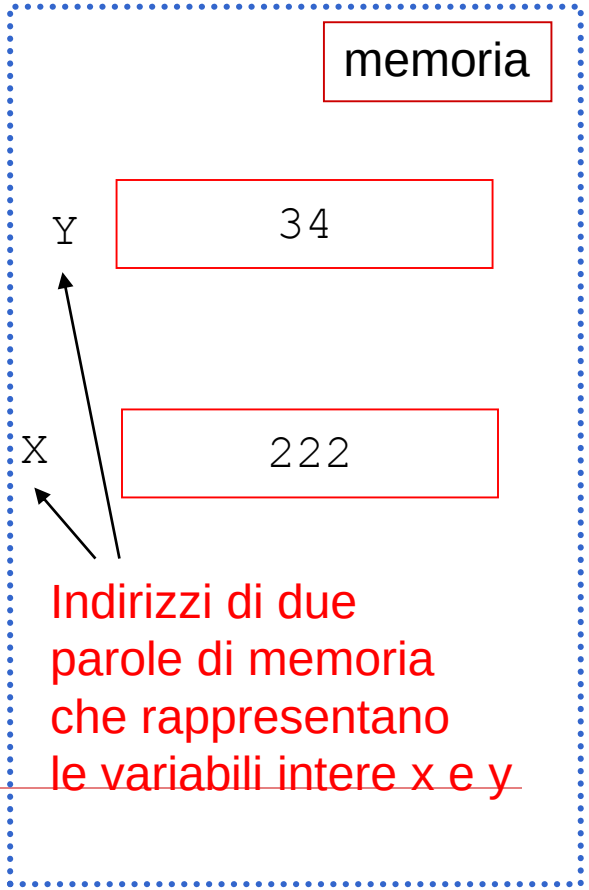
# Esempio: come si realizza in assembler l'operazione *x=y+2*

assembler

```
LOAD   Y, R1
ADD     2, R1
STORE  R1, X
```

Aggiunge 2 a R1


Registro interno del processore (variabile temporanea su cui lavorare)



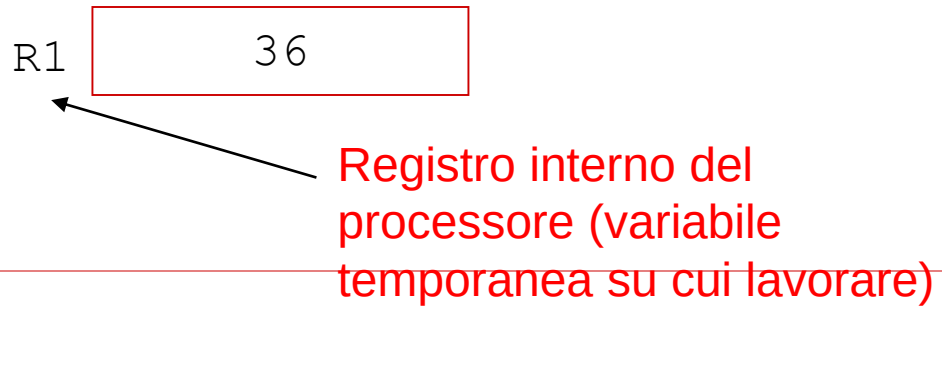

Indirizzi di due parole di memoria che rappresentano le variabili intere x e y

# Esempio: come si realizza in assembler l'operazione *x=y+2*

assembler

```
LOAD   Y, R1
ADD     2, R1
STORE  R1, X
```

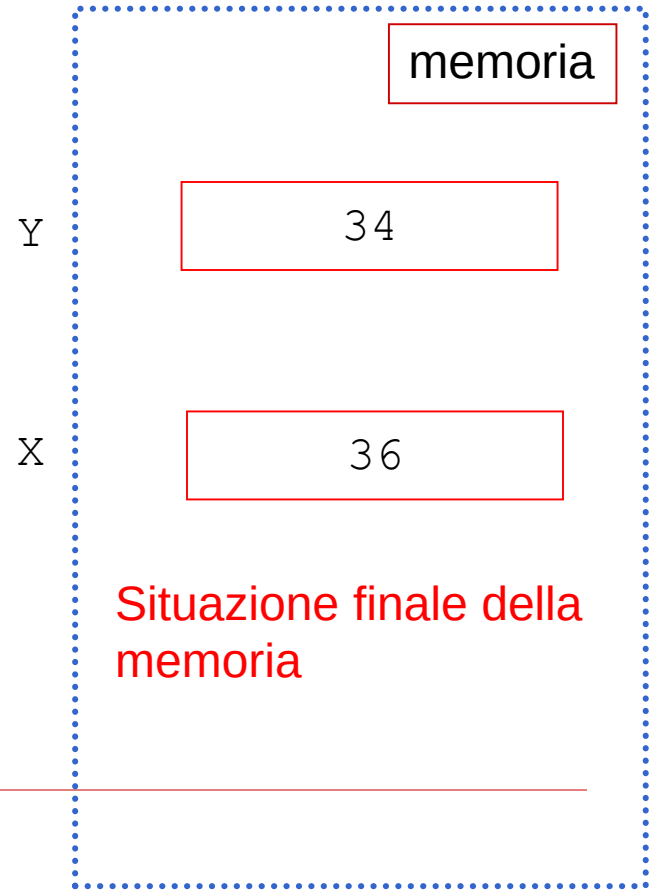
Scrive il contenuto di R1 nella parola di indirizzo X



# Esempio: come si realizza in assembler l'operazione *x=y+2*

assembler

```
LOAD   Y, R1
ADD     2, R1
STORE  R1, X
```

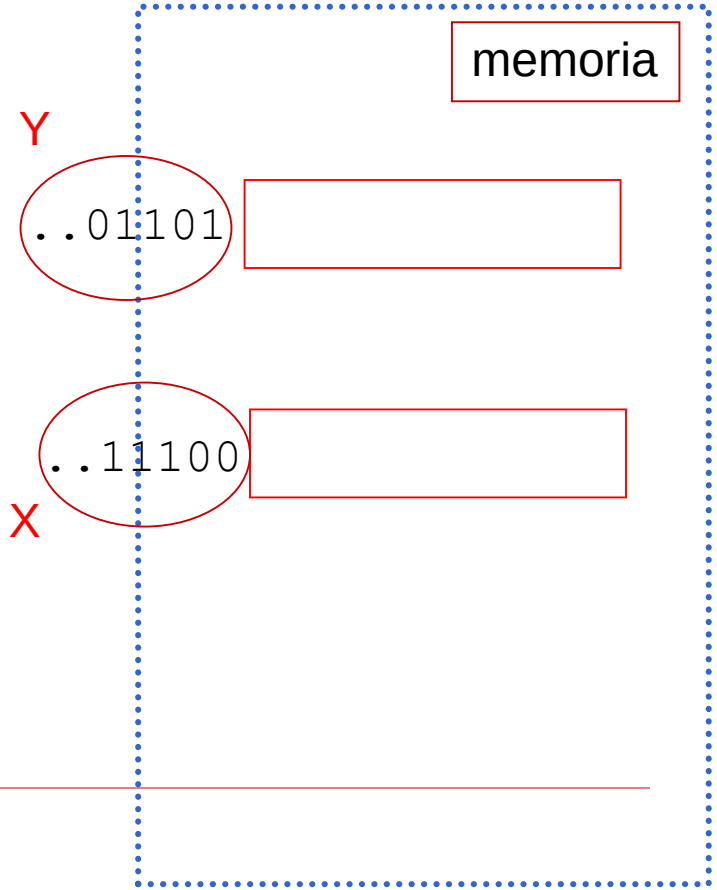
# Il linguaggio macchina

- Traduzione binaria (in linguaggio macchina) di

  LOAD  Y, R1

  ADD  2, R1

  STORE  R1, X
- Prima di tutto dobbiamo decidere quale è il vero indirizzo di X e Y



# Il linguaggio macchina

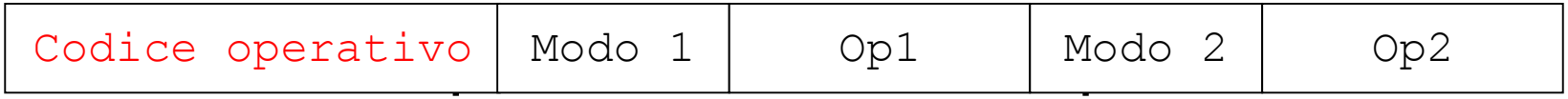
- Codifica binaria di

  LOAD ..01101, R1

  ADD 2, R1

  STORE R1, ..11100

- Ogni operazione si codifica secondo un formato stabilito da chi costruisce l'hardware



# Il linguaggio macchina

- i campi del formato :


È la codifica dell'operazione da eseguire
Ad esempio:

```
ADD     0001: AGGIUNGE IL VALORE INDICATO DA OP1 IN OP2
LOAD    0110: CARICA IN OP2(dest) IL VALORE IN OP1(sorg)
STORE   0111: SALVA IN OP2(dest) IL VALORE IN OP1(sorg)
```

# Il linguaggio macchina

- Vediamo i vari campi del formato :

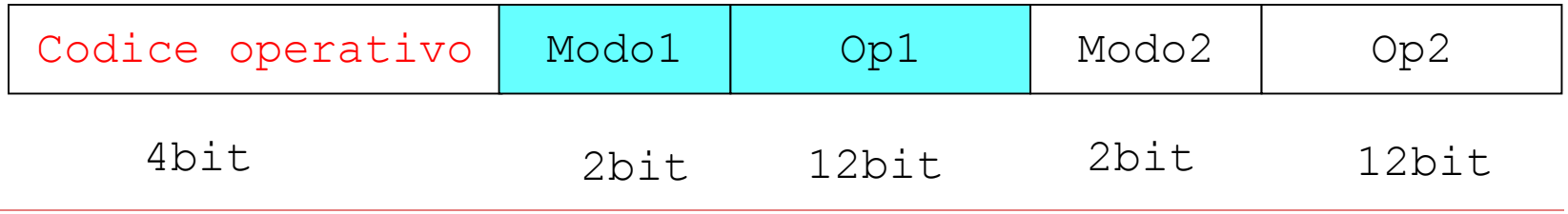

È la codifica primo operando, MODO1
dice a cosa si riferisce OP1 (modalità di indirizzamento)
es:

```
se MODO1 = 00 l'operando è nel registro
              interno del processore
              (e OP1 è il numero del registro)
se MODO1 = 01 l'operando è in memoria
              (e OP1 è l'indirizzo)
se MODO1 = 10 l'operando è immediato, cioè
              OP1 è direttamente il valore
              dell'operando
```

Lo stesso vale per il secondo operando!

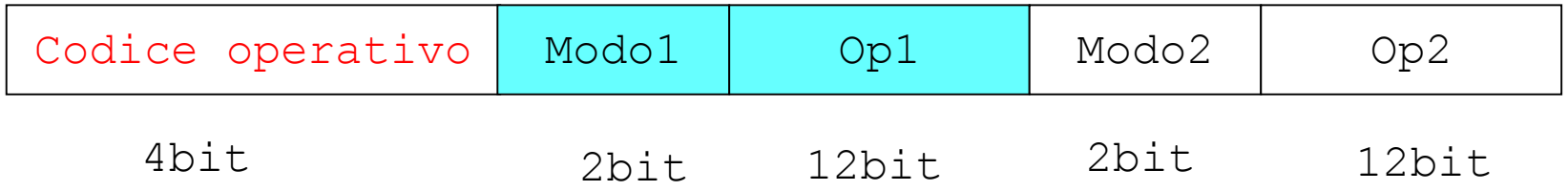
# Il linguaggio macchina

- Tipicamente la codifica di una istruzione e' lunga come una parola o multipli della parola per poterla leggere dalla memoria in modo più efficiente:
  - es : con parole di 4 byte (32 bit)



# Il linguaggio macchina

- Problema …..
  - es : con 12 bit posso indirizzare 'solo' $2^{12}$ parole di memoria centrale (RAM) !




Cioè posso avere al massimo una RAM di 4K parole … se ne ho di più devo inventarmi codifiche diverse….

# E' possibile combinare semplici istruzioni in linguaggio macchina per portare a termine alcune operazioni algoritmiche di alto livello

- Codifica binaria di
  LOAD ..01101, R1
  ADD 2, R1
  STORE R1, ..11100

| MODI |
|---|
| 00 registro |
| 01 memoria |
| 10 immediato |
| **CODICI OPERATIVI** |
| ADD 0001 |
| LOAD 0110 |
| STORE 0111 |

| Codice operativo | Modo1 | Op1 | Modo2 | Op2 | |
|---|---|---|---|---|---|
| 4bit | 2bit | 12bit | 2bit | 12bit | |
| 0110 | 01 | ..01101 | 00 | ..00001 | load |
| 0001 | 10 | ..00010 | 00 | ..00001 | add |
| 0111 | 00 | ..00001 | 01 | ..11100 | store |

# E’ possibile combinare semplici istruzioni in linguaggio macchina per portare a termine alcune operazioni algoritmiche di alto livello