

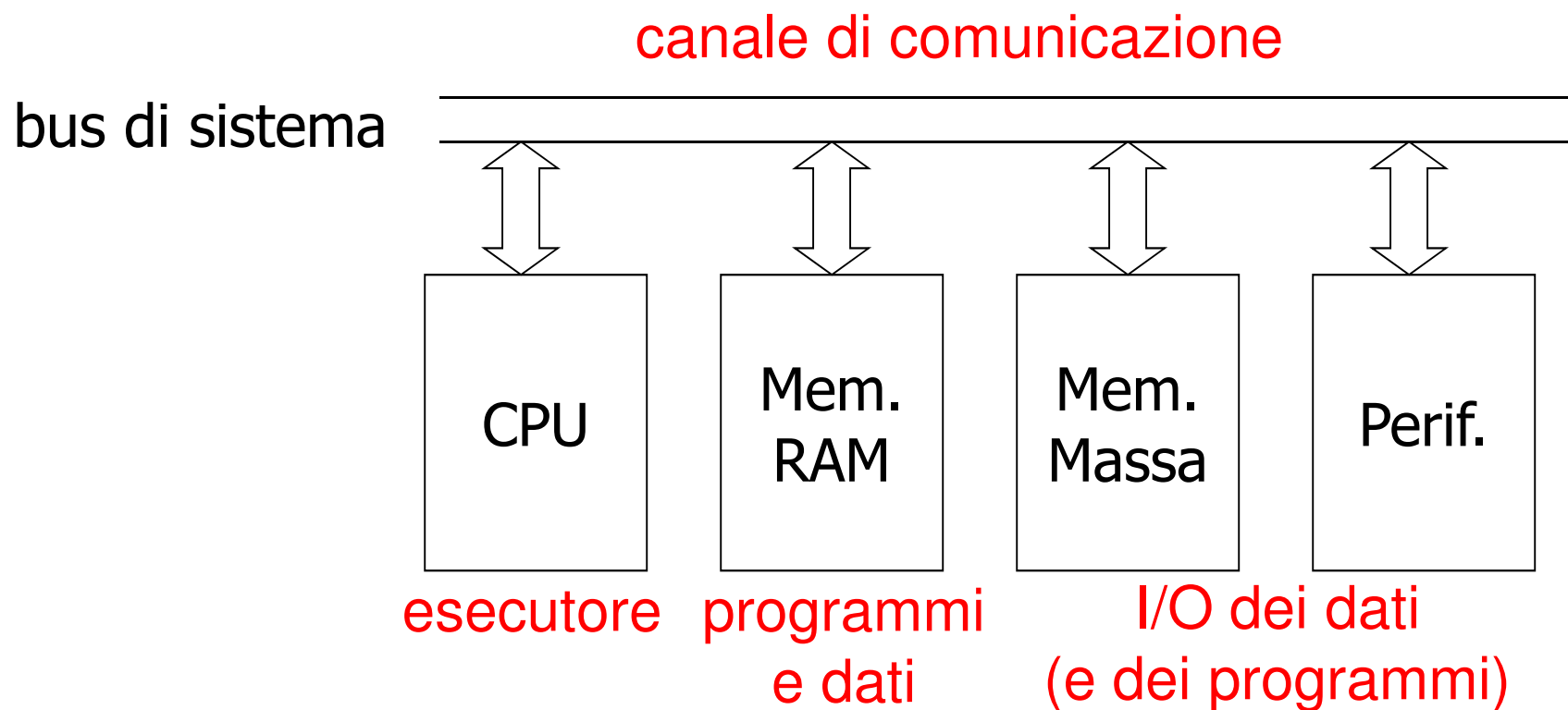
Fondamenti di Informatica

Allievi Automatici

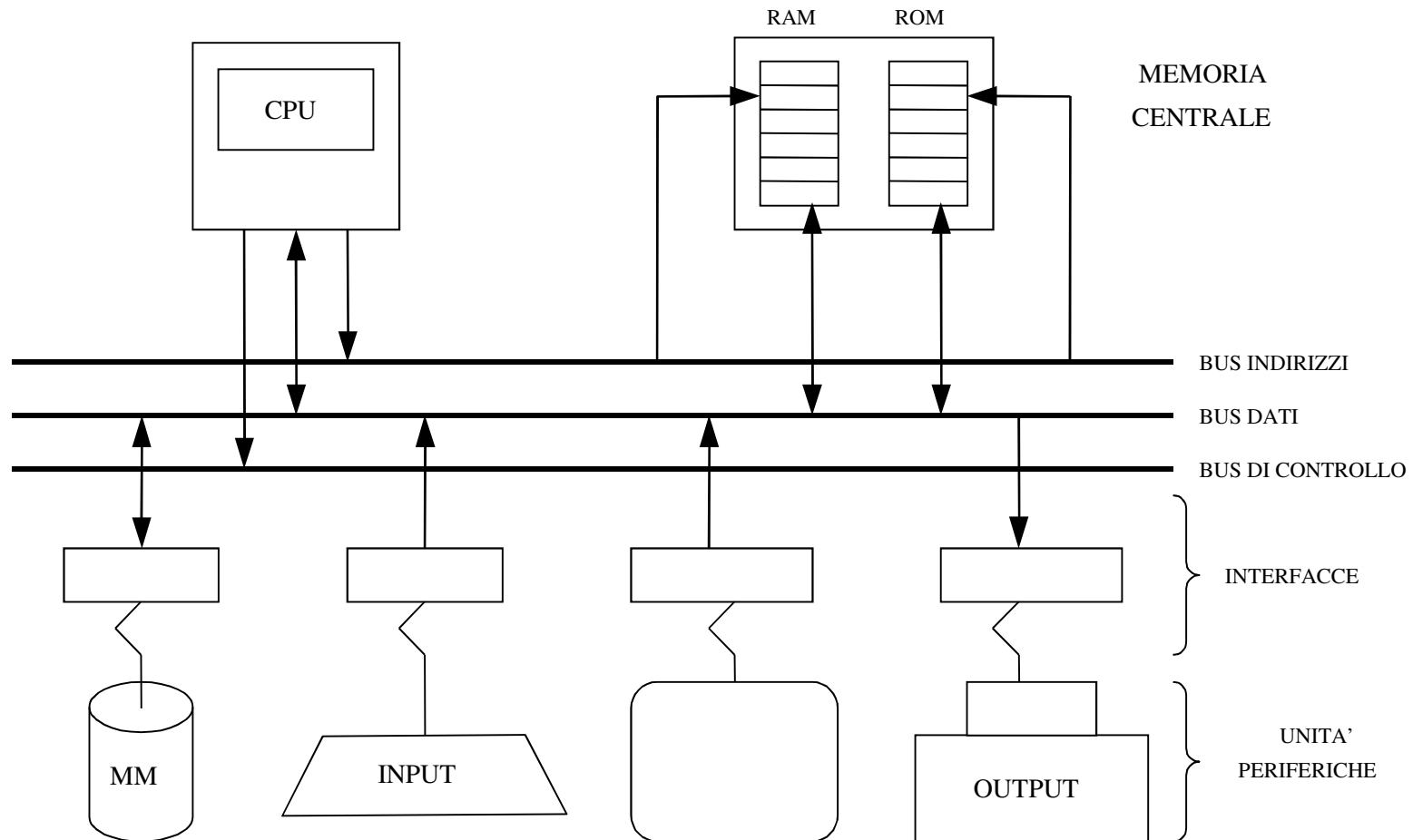
A.A. 2015-16

Architettura del Calcolatore

La macchina di von Neumann



Architettura del calcolatore

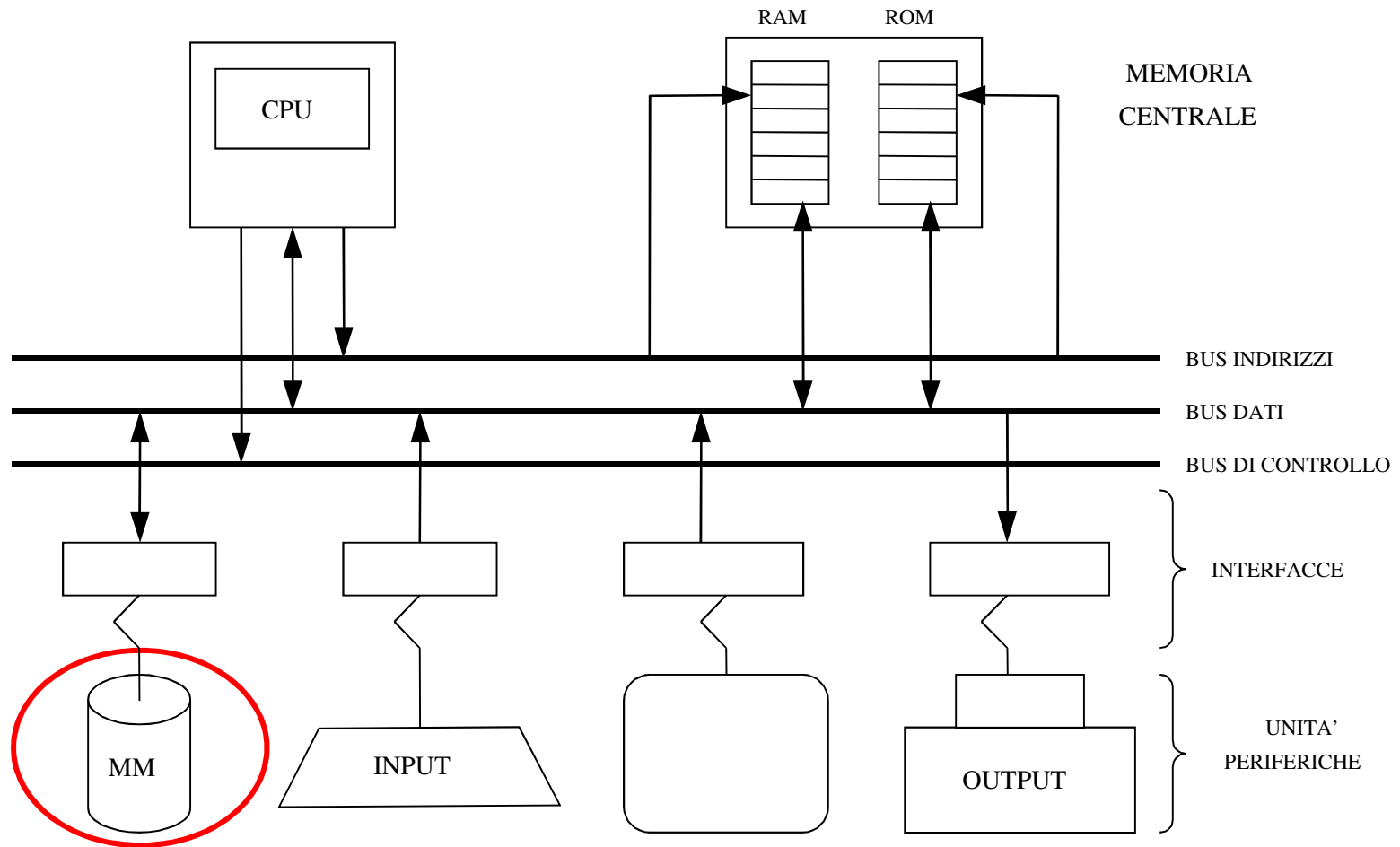


Elementi funzionali

- **Unità di elaborazione, o CPU**
 - elabora dati, coordina trasferimento dei dati
 - esegue i programmi, cioè interpreta ed esegue le loro istruzioni
- **Memoria Centrale**
 - memorizza dati e programmi
 - capacità limitata (esempio: 4 GByte)
 - volatile
 - accesso all'informazione molto rapido
- **Bus di sistema**
 - collega e consente scambio di dati

- **Memoria secondaria o memoria di massa**
 - memorizza grandi quantità di dati e programmi
 - persistente
 - accesso molto meno rapido della RAM
 - Su PC: hard disk (es. 750 GB), CD-ROM (700 MB), DVD (4,7 GB), pendrive usb (es. 8 GB), scheda microSD (32 GB)
- **Unità periferiche (I/O)**
 - comunicazione con l'ambiente esterno
 - **terminali**, con tastiera, mouse, video, **stampanti**
 - l'ambiente esterno non è sempre un utente umano (impianti industriali, robot, strumenti di controllo)
 - Sensori e attuatori

Architettura del calcolatore

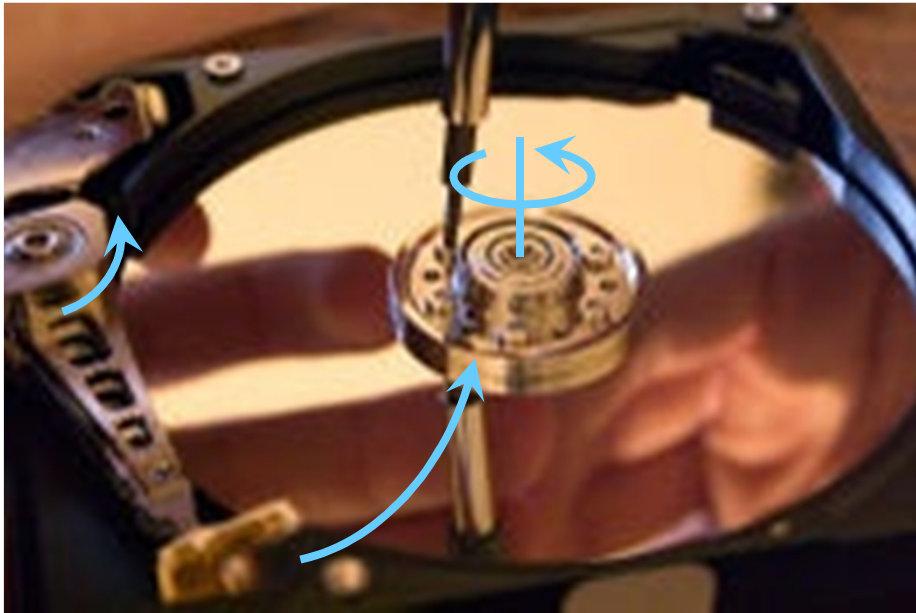
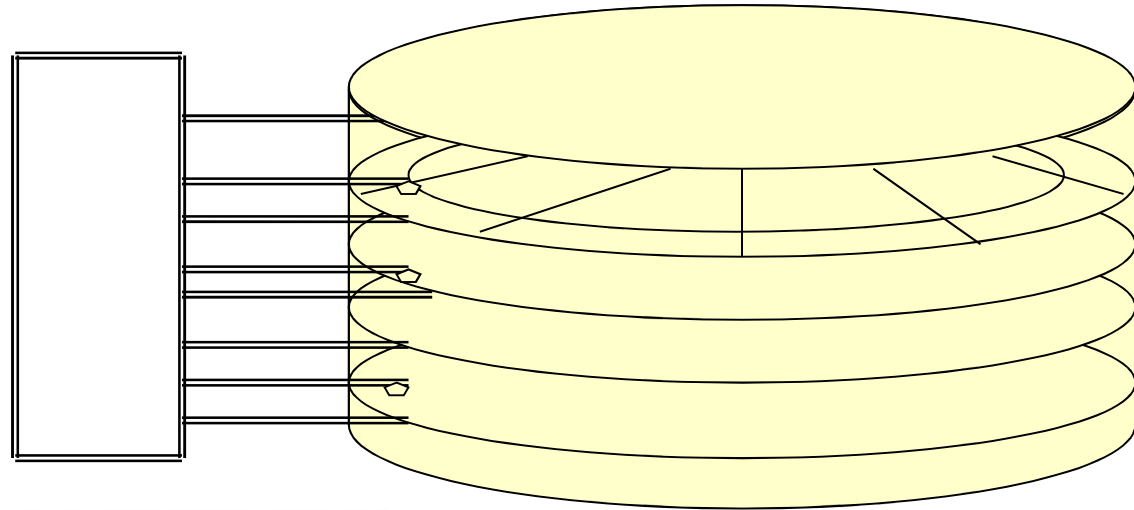


Le memorie di massa

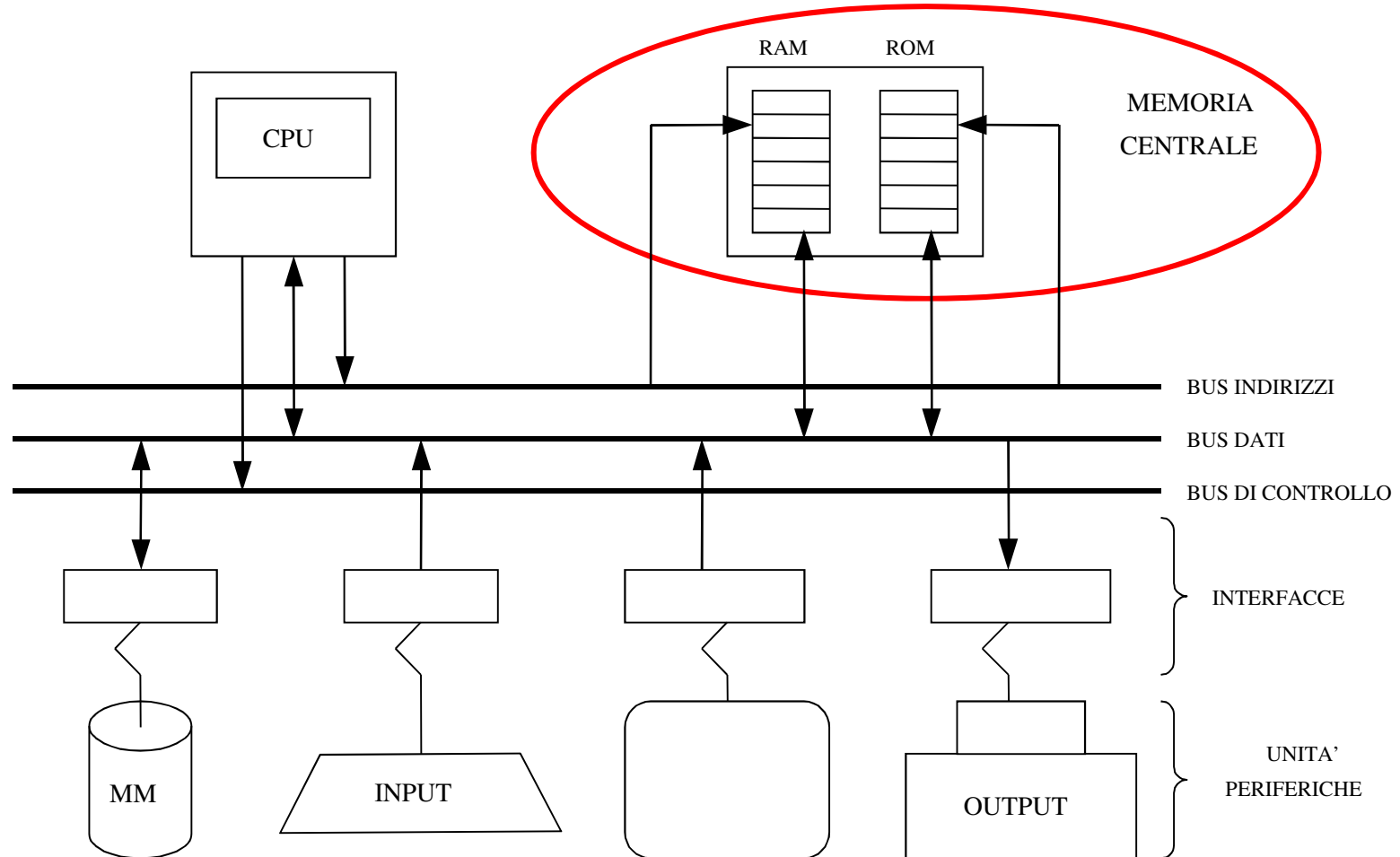
- Informazione memorizzata su supporto magnetico (o ottico o altro...) di costo contenuto
- Memoria permanente: le informazioni sono raggruppate in *file* gestiti dal *sistema operativo*.
- Dischi rigidi: tempi di accesso dell'ordine delle *decine* di **millisecondi**; floppy disk: dell'ordine delle *centinaia* di millisecondi
- Nastri e Cartucce: per memorizzare informazioni “storiche” (back-up)
- CD-ROM e DVD-ROM: sola lettura (700 MB e 4,7 GB)
 - CD-RW e DVD-RW: anche riscrivibili

Struttura dei dischi rigidi (Hard Disk)

- **testine**
- **superfici**
- **tracce**
- **cilindri**
- **settori**



Architettura del calcolatore



La memoria centrale

- Contiene le istruzioni e i dati su cui la CPU può operare
 - Contiene, cioè, sia i **dati** sia i **programmi**
 - Tutta l'informazione, per poter essere elaborata, deve passare dalla memoria centrale (e successivamente caricata in uno dei registri della CPU)
- Rispetto alla memoria di massa
 - è memoria a breve-medio periodo
 - è volatile (si perde allo spegnendo la macchina)

La memoria centrale

- Dimensioni ridotte
 - ordine dei GByte (ma fino a non molto tempo fa, MByte)
- Tempi di accesso
 - ordine delle decine di **nanosecondi**
 - circa 1 milione di volte più veloce delle memorie di massa
- È un insieme ordinato di parole (**celle**)
 - 1 parola = n elementi binari (8, 16, 32, 64 bit)
 - La posizione di ogni parola è identificata da un indirizzo
- *Capacità di indirizzamento* (numero di celle)
 - definita dalle dimensioni di bus indirizzi e registro indirizzi

RAM e ROM

- RAM: Random Access Memory
 - Le celle sono indirizzabili in un ordine qualunque (accesso random = diretto, non "casuale")
 - Il tempo di accesso non dipende dalla cella
- ROM: Read-Only memory
 - Per programmi protetti e definiti dal costruttore
 - Il BIOS (Basic I/O System) che carica in memoria il sistema operativo quando la macchina viene accesa
 - Ne esistono di diversi tipi
 - “Erasable”, “Programmable”, (**EPROM**)
 - Una via intermedia tra Hardware e Software (Firmware)

Tecnologie per la memoria centrale

■ La memoria RAM

- È realizzata mediante circuiti a transistori
- È modificabile (leggibile e scrivibile) ma deve essere continuamente alimentata per mantenere le informazioni (volatile)
- All'accensione il suo contenuto è una sequenza casuale di 0 e 1

■ La memoria ROM

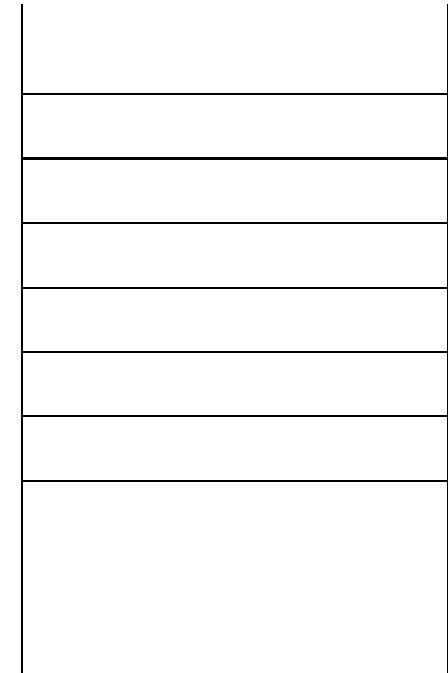
- È solo leggibile: le informazioni sono di solito scritte in modo permanente dal costruttore
- È caricata al momento della produzione del calcolatore
- Vi si accede ogni qualvolta questo viene acceso
- Contiene il **bootstrap**, un programma contenente le prime istruzioni che la CPU deve eseguire

Indirizzamento della memoria centrale

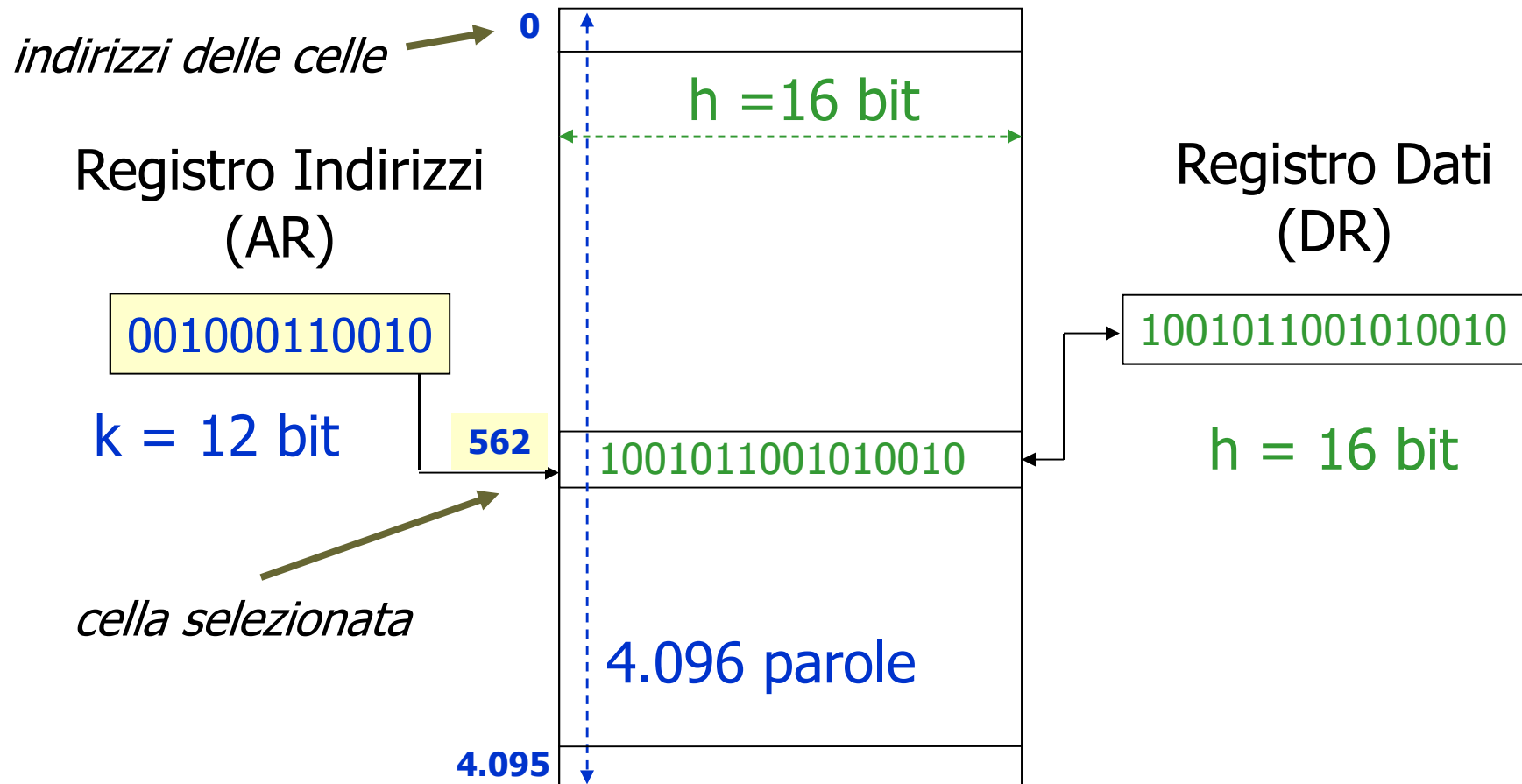
- Esiste un registro (*registro indirizzi - AR*) della CPU per indirizzare la memoria
- È un registro di k bit: può indirizzare 2^k celle
 - Con 10 bit si indirizzano 1.024 celle
 - 1 “kilo-parole”
 - Con 20 bit, 1.048.576 parole
 - 1 “mega” parole
 - 30(giga), 40(tera), 50(peta), 60(exa)

Celle

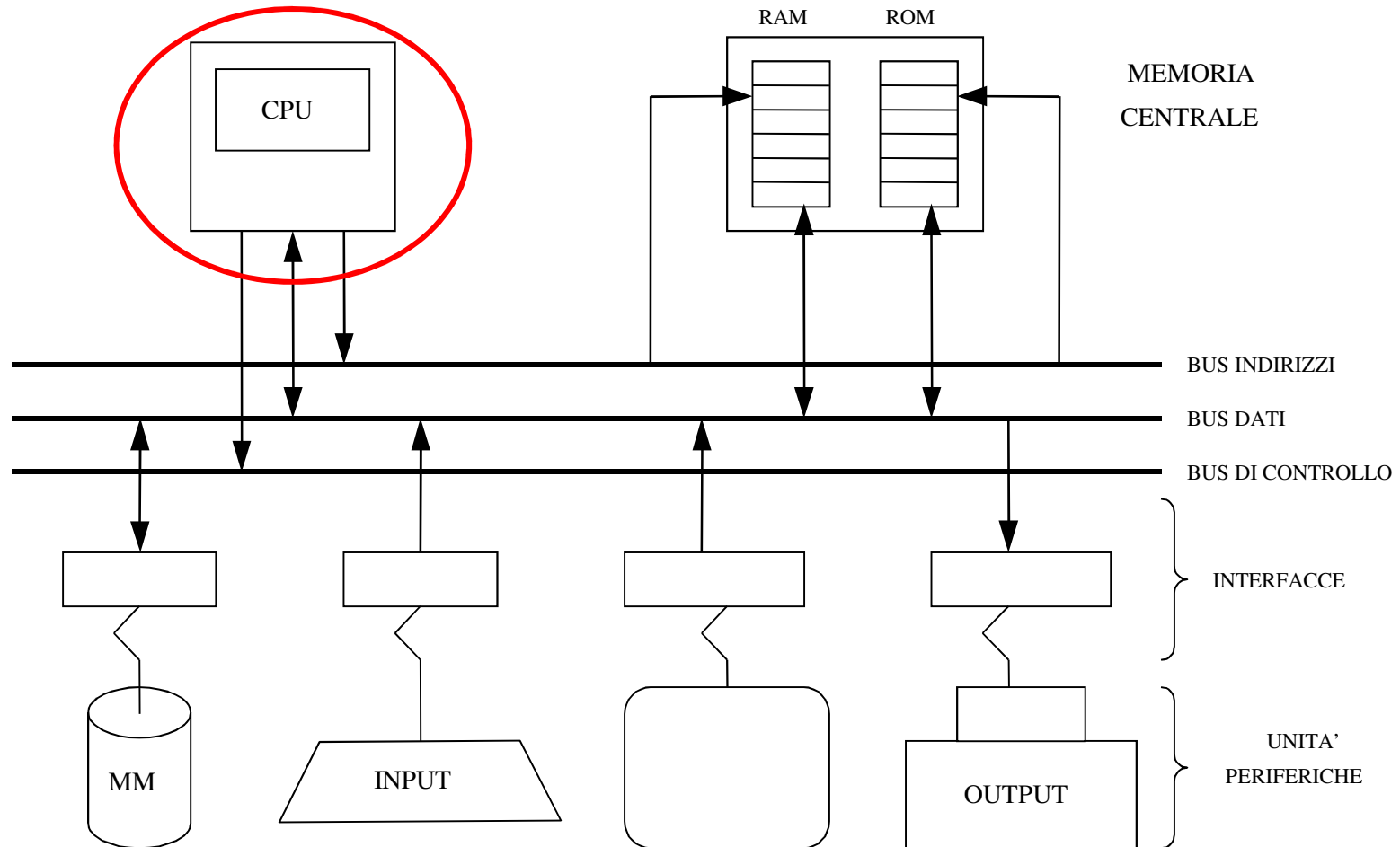
- hanno un **indirizzo**
- contengono **parole**



Esempio di RAM da 8 KByte = $h \times 2^k$ bit



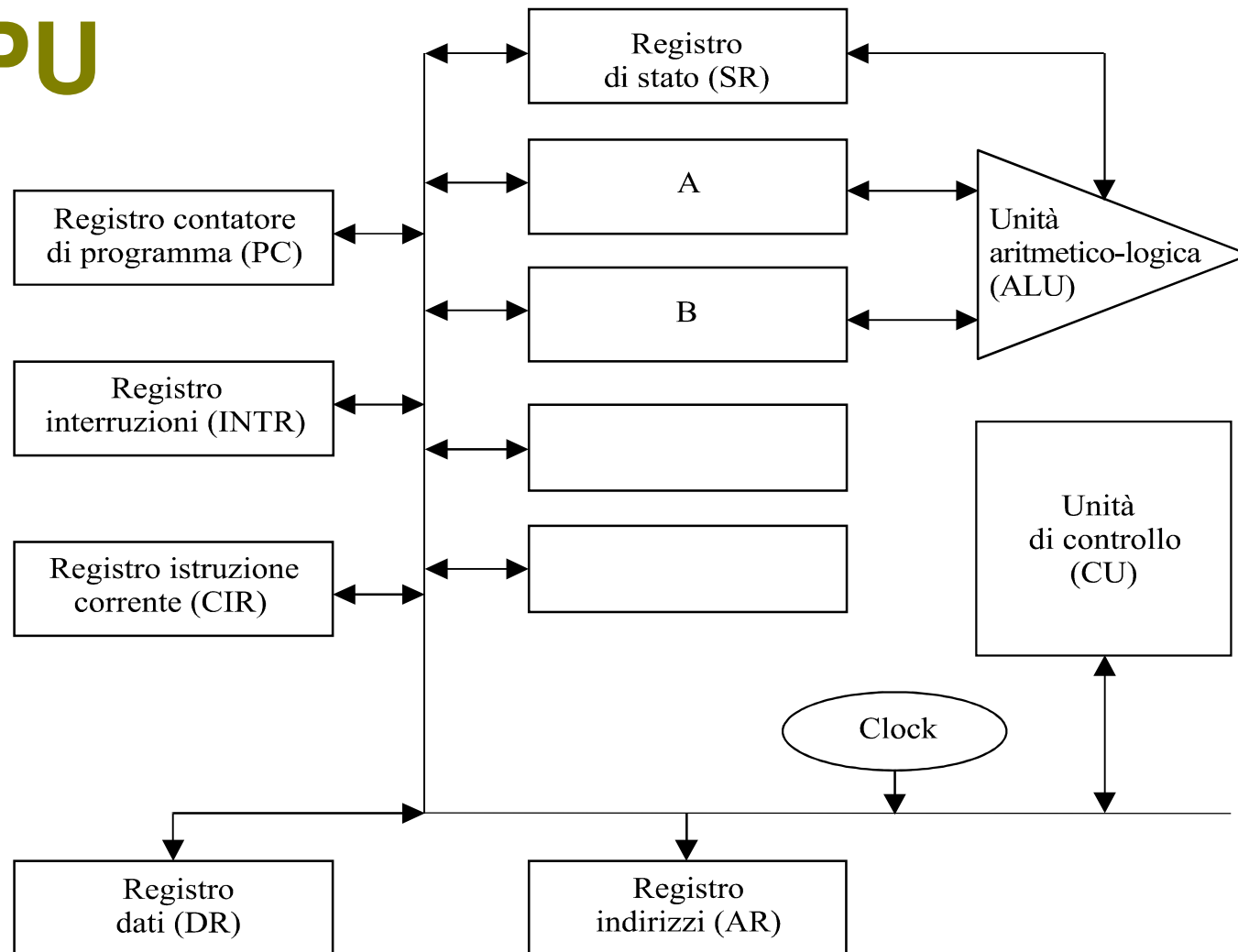
Architettura del calcolatore



La CPU

- Contiene gli elementi circuitali che regolano il funzionamento del calcolatore:
 - **L'unità di controllo** è responsabile della **decodifica** e dell'**esecuzione** delle istruzioni. "Dirige" l'azione delle altre parti
 - **L'orologio di sistema (clock)** permette di sincronizzare le operazioni, temporizzando il funzionamento del calcolatore
 - **L'unità aritmetico-logica (ALU)** realizza le operazioni aritmetiche e logiche eventualmente richieste per l'esecuzione dell'istruzione. È **priva di facoltà di scelta**
 - **I registri** sono piccole memorie velocemente accessibili, utilizzate per memorizzare risultati parziali o informazioni necessarie al controllo. L'insieme dei valori contenuti nell'insieme di tutti i registri in un dato istante dell'elaborazione è detto **contesto**

La CPU





Formato istruzioni in linguaggio macchina

- Costituite (ovviamente) da sequenze di 0 e 1
 - campo **codice operativo** (*obbligatorio*) specifica l'operazione da eseguire
 - campo **operandi** (*facoltativo*) indica i dati da utilizzare (*gli operandi possono essere uno o due*)
 - Può contenere direttamente il *valore*, o l'indirizzo della cella che contiene il valore (*riferimento a una variabile*)
 - **FORMATO ISTRUZIONE:**

| | |
|----------------|----------|
| Cod. operativo | Operandi |
|----------------|----------|

Il ciclo “preleva-esegui”

- La memoria RAM contiene le istruzioni
- Ciclicamente:

 **Preleva** la prossima istruzione
Decodificala ed **eseguila** 

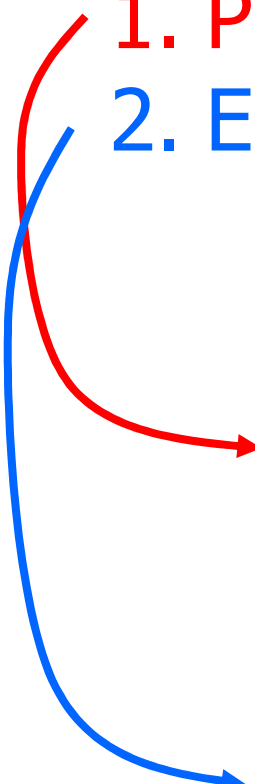
- Istruzioni
 - di **elaborazione** dei dati
 - di **trasferimento** dei dati
- Funzionamento scandito dall'orologio di sistema (“clock”)

Fasi del ciclo di CPU

- **Prelievo (o Fetch):**
 - PC contiene l'indirizzo della prossima istruzione
 - acquisizione prossima istruzione da memoria (che viene scritta nel CIR, Registro Istr. Corrente)
 - incremento del registro PC
- **Decodifica:** interpretazione codice operativo
- **Esecuzione:** attivazione dell'esecuzione pertinente all'operazione corrente
 - Operazioni della ALU sugli operandi (valori o indirizzi)
 - Operazione di salto (modifica PC)

Il ciclo “preleva-esegui”

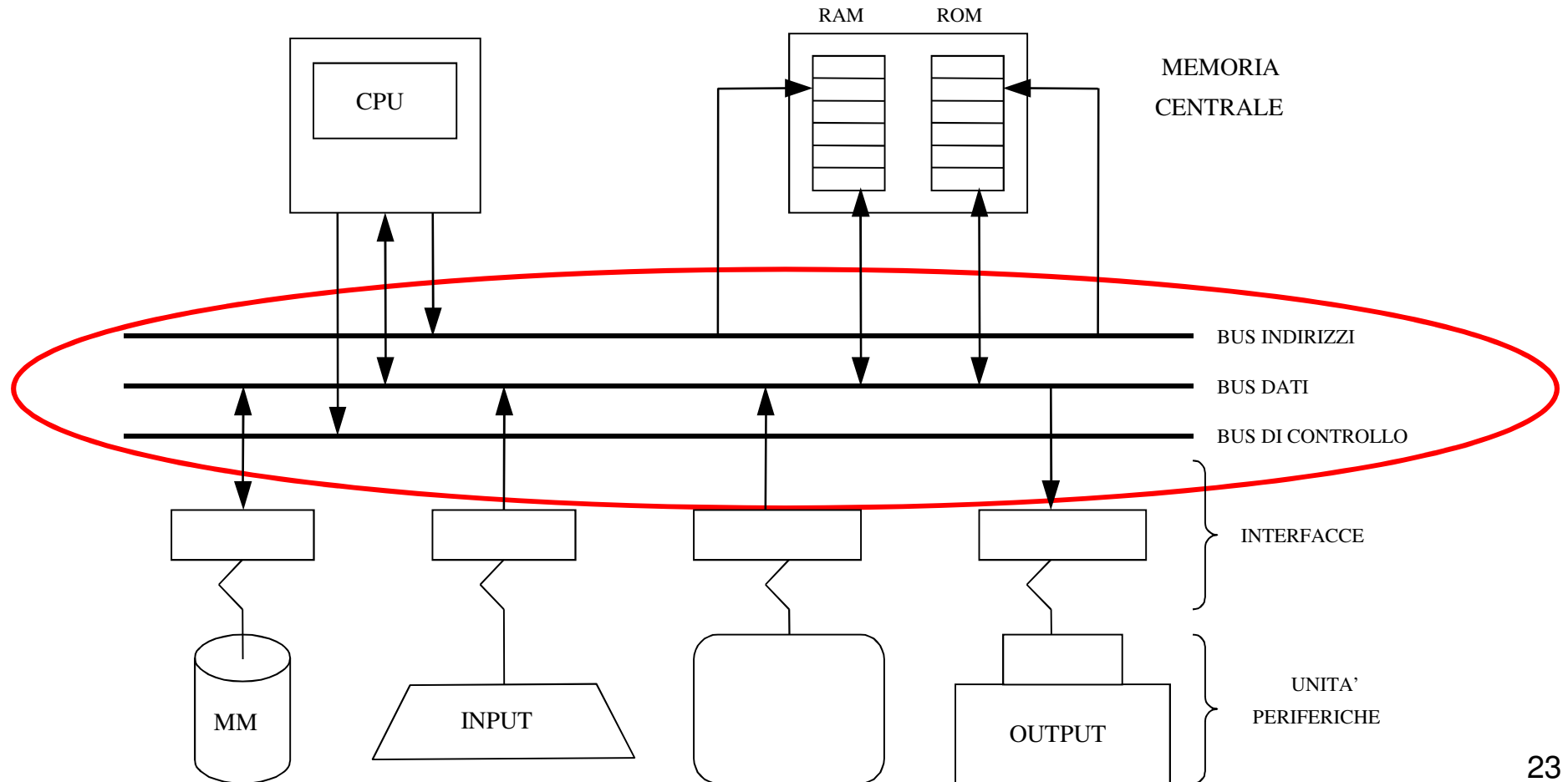
1. Preleva la prossima istruzione
2. Esegui l'istruzione prelevata



1. Contenuto di PC → AR
2. Lettura da memoria centrale → DR
3. Trasferimento da DR → CIR
4. Incremento di PC

5. Interpretazione ed esecuzione di CIR

Architettura del calcolatore



Il bus di sistema

- Insieme di connettori (conduttori elettrici) che trasportano bit di informazioni collegando fra di loro l'unità di elaborazione, la memoria e le varie interfacce di ingresso/uscita
- I trasferimenti sono gestiti dalla CPU (modalità **master/slave**) e si chiamano ***cicli*** del bus, che con la sua capacità ne determina la velocità

Componenti del bus di sistema

- **Bus Dati**
 - trasferisce dati da master a slave e viceversa
- **Bus Indirizzi**
 - trasferisce indirizzi, per esempio l'indirizzo di un dato dal registro indirizzi alla memoria, per accedere al dato stesso
- **Bus di Controllo**
 - Da master a slave: codice dell'istruzione da eseguire (per es. lettura da disco)
 - Da slave a master: informazioni sul successo dell'operazione

Cicli di bus: esempi

- **Operazione di lettura da memoria centrale (LOAD):**
 - la CPU carica l'indirizzo della parola di memoria nel registro indirizzi e lo trasmette alla memoria via **bus indirizzi**
 - la CPU invia il comando di Read Memory sul **bus di controllo**
 - la memoria trasmette sul **bus dati** il contenuto della parola verso il registro dati
 - la memoria segnala al processore sul **bus di controllo** che l'operazione è stata completata con successo: il dato si trova **nel registro dati**

Cicli di bus: esempi

- **Operazione di scrittura in memoria centrale (STORE):**

- la CPU carica indirizzo della parola di memoria dove si vuole scrivere nel registro indirizzi e lo trasmette alla memoria via **bus indirizzi**
- la CPU carica nel registro dati la parola da scrivere in memoria
- la CPU invia il comando di Write Memory sul **bus di controllo**
- la CPU trasmette sul **bus dati** il contenuto del registro dati verso l'indirizzo di memoria segnalato
- la memoria segnala al processore sul **bus di controllo** che l'operazione è stata completata con successo: il dato si trova **nella parola di memoria** destinazione

| Cod. binario | Cod. simbolico | Significato |
|---------------------|-----------------------|--|
| 0000 | LOADA | <i>poni in A il valore della cella <op> della RAM</i> |
| 0001 | LOADB | <i>poni in B il valore della cella <op> della RAM</i> |
| 0010 | STOREA | <i>poni nella cella <op> della RAM il valore in A</i> |
| 0011 | STOREB | <i>poni nella cella <op> della RAM il valore in B</i> |
| 0110 | ADD | <i>poni in A il valore di A+B restituito dalla ALU</i> |
| 0111 | DIF | <i>poni in A il valore di A-B restituito dalla ALU</i> |
| 1010 | JMP | <i>la prossima istr. da eseguire è nella cella <op></i> |
| 1011 | JMPZ | <i>se A=0, la prossima istr. da eseguire è in <op></i> |
| 1100 | NOP | <i>istruzione nulla: nessun effetto [...HA senso]</i> |
| 1101 | HALT | <i>fine del programma</i> |
| 0100 | READ | <i>leggi (I/O) un valore e ponilo nella cella <op></i> |
| 0101 | WRITE | <i>scrivi (I/O) il valore contenuto nella cella <op></i> |
| 1000 | LDCA | <i>poni in A la costante (il valore numerico) <op></i> |
| 1001 | LDCB | <i>poni in B la costante (il valore numerico) <op></i> |
| ... | ... | |

Piccolo esempio di programma assembler

| <i>Cella di mem.</i> | <i>Cod. Oper.</i> | <i>Operando</i> |
|-----------------------------|--------------------------|------------------------|
| 0 | READ | 8 |
| 1 | READ | 9 |
| 2 | LOADA | 8 |
| 3 | LOADB | 9 |
| 4 | ADD | - |
| 5 | STOREA | 10 |
| 6 | WRITE | 10 |
| 7 | HALT | - |
| 8 | | (INT) |
| 9 | | (INT) |
| 10 | | (INT) |

Che cosa fa ?

Si noti che il primo valore letto si perde nel processore (sovrascritto per effetto della ADD) ma resta in memoria centrale (cella 8)

Esecuzione dei programmi

Esempio: calcolare l'espressione $(x + y) - (z + w)$

1. poni in memoria centrale (rispettivamente nelle celle 16, 17, 18 e 19) i valori di x , y , z , e w
2. esegui l'addizione di z e w :
 1. copia da cella 18 a registro A
 2. copia da cella 19 a registro B
 3. somma i due registri (operazione eseguita dalla ALU che lascia il risultato nel registro A)
3. immagazzina risultato (che ora è nel registro A) nella cella 20

4. esegui l'addizione di x e y :
 1. copia da cella 16 a registro A;
 2. copia da cella 17 a registro B;
 3. somma i registri (operazione eseguita dalla ALU che lascia il risultato nel registro A);
5. esegui la sottrazione di $(x + y)$ e $(z + w)$:
 1. copia da cella 20 a registro B;
 2. sottrai il contenuto dei due registri (operazione eseguita dalla ALU che lascia il risultato nel reg. A);
6. scrivi il risultato sul dispositivo di uscita:
 1. copia da registro A a cella 20;
 2. copia da cella 20 a registro dati della periferica
7. arresta l'esecuzione del programma

Il programma in assembler

0-3: acquisizione dei dati

16-20: spazio per rappresentare i dati

- le cinque variabili simboliche X, Y, Z, W, RIS menzionate nel programma. Ognuna identifica una cella di memoria

- I nomi simbolici sono più comodi degli indirizzi, ma bisogna supporre che qualcuno (il compilatore!) gestisca la corrispondenza con un numero opportuno di celle allocate per rappresentare i dati

4-6 e 8-10: Somme ($Z+W$ e $X+Y$)

7: salvataggio del risultato intermedio

11: recupero del risultato intermedio

14: visualizzazione del risultato

| | | |
|----|------------|----------|
| 0 | READ | X |
| 1 | READ | Y |
| 2 | READ | Z |
| 3 | READ | W |
| 4 | LOADA | Z |
| 5 | LOADB | W |
| 6 | ADD | |
| 7 | STOREA | RIS |
| 8 | LOADA | X |
| 9 | LOADB | Y |
| 10 | ADD | |
| 11 | LOADB | RIS |
| 12 | DIF | |
| 13 | STOREA | RIS |
| 14 | WRITE | RIS |
| 15 | HALT | |
| 16 | ...int.... | (X)..... |
| 17 | ...int.... | (Y)..... |
| 18 | ...int.... | (Z)..... |
| 19 | ...int.... | (W)..... |
| 20 | ...int... | (RIS)... |

Il programma in linguaggio macchina

indirizzi delle celle

codici operativi

*operando significativo
(indirizzo della cella RIS)*

operando non significativo

| | | |
|-----|-------|---------------|
| 0. | 0100 | 0000000010000 |
| 1. | 0100 | 0000000010001 |
| 2. | 0100 | 0000000010010 |
| 3. | 0100 | 0000000010011 |
| 4. | 0000 | 0000000010010 |
| 5. | 0001 | 0000000010011 |
| 6. | 0110 | 0000000000000 |
| 7. | 0010 | 0000000010100 |
| 8. | 0000 | 0000000010000 |
| 9. | 0001 | 0000000010001 |
| 10. | 0110 | 0000000000000 |
| 11. | 0001 | 0000000010100 |
| 12. | 0111 | 0000000000000 |
| 13. | 0010 | 0000000010100 |
| 14. | 0101 | 0000000010100 |
| 15. | 1101 | 0000000000000 |
| 16. | | |
| 17. | | |
| 18. | | |
| 19. | | |
| 20. | | |

istruzioni

dati

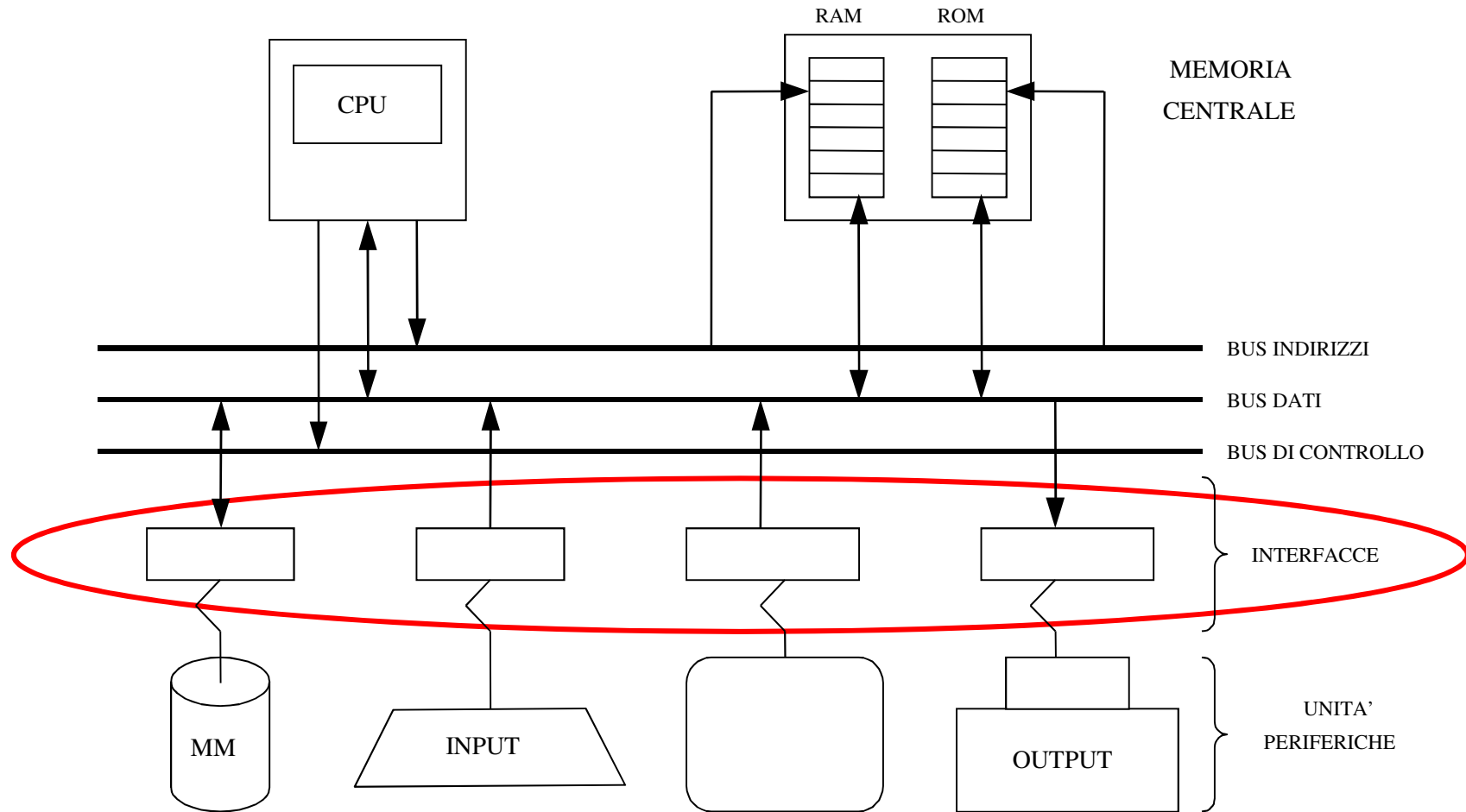
Il programma in C

```
int main() {  
    int x, y, z, w, ris;  
    scanf("%d%d%d%d", &x, &y, &z, &w);  
    ris = (x+y)-(z+w);  
    printf("\n\n Il risultato e' : %d", ris);  
    return 0;  
}
```

Notiamo che:

- il compilatore deve decidere quanto spazio allocare per le variabili (e *dove* allocarle – prima o dopo le istruzioni?)
- La corrispondenza posizionale tra i %d e le variabili si traduce in un preciso ordine delle istruzioni READ in assembler

Architettura del calcolatore



Interfacce di I/O (verso le periferiche)

- Consentono il collegamento tra elaboratore e periferiche
- Possono essere dispositivi elettromeccanici
- Contengono *registri* per
 - inviare comandi alla periferica
 - *registro comandi periferica (PCR)* – collegato al bus di controllo
 - scambiare dati
 - *registro dati della periferica (PDR)* – collegato al bus dati
 - controllare il funzionamento della periferica
 - *registro stato periferica (PSR)*: pronto, occupato, errore, ...

Il sistema operativo

- Programma di grandi dimensioni e notevole complessità, che permette all'utente di interagire con il calcolatore
- Sviluppo di un sistema operativo: centinaia di anni-uomo
- Poca teoria, e solo in alcune aree, perché...
 - storicamente: prima lo sviluppo tecnologico, poi la teorizzazione
 - aspetti non di calcolo ma di ottimizzazione, gestione di interazione, gestione di malfunzionamenti etc.
 - il progetto di un sistema operativo è un problema intrinsecamente complesso e interdisciplinare
- Complessità estrema: studio per **livelli di macchine astratte** per ragionare a diversi livelli di astrazione
- Funzioni:
 - Supporto per la programmazione
 - Meccanismi di Ingresso / Uscita (I/O)
 - Gestione archivi (file)