Lezione 10

L'architettura di riferimento

Proff. A. Borghese, F. Pedersini

Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano

Sommario

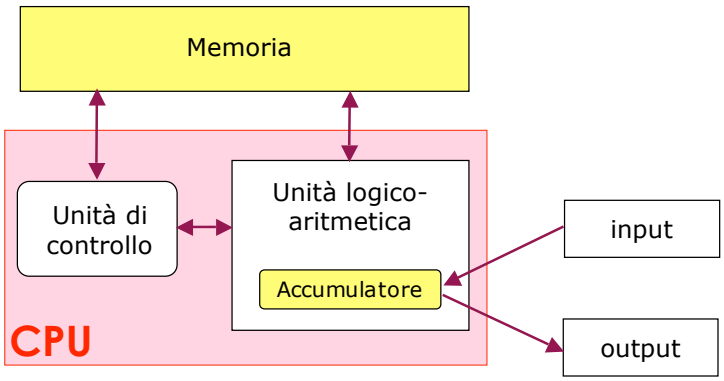
- Architettura di Von Neumann, architetture moderne
- Il ciclo di esecuzione di un'istruzione.
- All'esterno della CPU: il bus, la memoria.

Architettura di Von Neumann

- **Architettura di VON NEUMANN:**
  - Dati e Istruzioni sono contenute in una memoria leggibile e scrivibile
    - Accesso al contenuto della memoria, in base alla sua posizione (Indirizzo), non in funzione del tipo di dato (istruzione/dato).
  - Esecuzione **sequenziale** da un'istruzione alla seguente
  - I risultati di un'operazione sono sempre contenuti nell' accumulatore

ALU + UC incorporate nella CPU

Von Neumann: Architettura ad Accumulatore



Tipi di architetture

Architetture moderne:

- **Accumulator** (1 register = 1 indirizzo di memoria)

1 address	**add A**	acc ← acc + mem[A]
1+x address	**addx A**	acc ← acc + mem[A + x]

- **Stack** (posso operare solo sui dati in cima allo stack)

0 address	**add**	tos ← tos + next

- **Register-addressed Memory** (tanti indirizzi di memoria quanti sono i registri: indirizzamento indiretto)

2 address	**add A B**	EA(A) ← EA(A) + EA(B)
3 address	**add A B C**	EA(A) ← EA(B) + EA(C)

- **Load/Store** (posso operare solamente sui dati contenuti nei registri. Devo prima caricarli dalla memoria).

3 address:	**add Ra Rb Rc**	Ra ← Rb + Rc
	load Ra [Rb]	Ra ← mem[Rb]
	store Ra [Rb]	mem[Rb] ← Ra

Architetture LOAD/STORE

- Registri ad uso generale insufficienti a memorizzare tutte le variabili di un programma
  - Esempio: 32 registri da 32 bit ciascuno
  - Assegnata una locazione di memoria ad ogni variabile nella quale trasferire il contenuto del registro quando questo deve essere utilizzato per contenere un'altra variabile.

- *Architetture **LOAD/STORE:***
  - Gli **operandi dell'ALU possono provenire soltanto dai registri** contenuti nella CPU e **non** possono provenire direttamente **dalla memoria**.
  - Sono necessarie apposite istruzioni di:

    *caricamento **(LOAD)*** dei dati da memoria ai registri;

    *memorizzazione **(STORE)*** dei dati dai registri alla memoria.

CISC vs. RISC

- Le 10 istruzioni più frequenti:

  IDEA: le istruzioni semplici richiedono la maggior parte del tempo di elaborazione

  → Ottimizzo le istruzioni semplici, trascuro il resto

  → Set di istruzioni ridotto, solo istruzioni semplici **RISC**

Rank	Instruction	Average Execution Frequency (%)
1	load	22%
2	conditional branch	20%
3	compare	16%
4	store	12%
5	add	8%
6	and	6%
7	sub	5%
8	move register-register	4%
9	call	1%
10	return	1%
	Total	96%

Architetture RISC

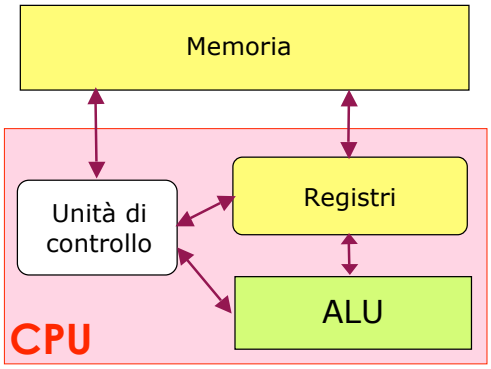
- **RISC**: **R**educed **I**nstruction **S**et **C**omputer
- IDEA: eseguire soltanto istruzioni semplici
  - Operazioni complesse scomposte in serie di istruzioni semplici da eseguire in un ciclo-base ridotto, ma ottimizzato.
  - Architettura semplice: ad esempio, Load-Store – gli operandi dell'*ALU* possono provenire solo dai registri, *non* dalla memoria
- Vantaggi:
  - *CPU* semplice → si riducono i tempi di esecuzione delle singole istruzioni
  - Dimensione **fissa** delle istruzioni → semplice gestione della fase di prelievo ***(fetch)*** e della decodifica delle istruzioni

Architetture di tipo CISC

- **CISC**: **C**omplex **I**nstruction **S**et **C**omputer
  Elevata complessità delle istruzioni eseguibili
  Elevato numero di istruzioni disponibili
- Numerose modalità di indirizzamento per gli operandi dell'ALU
  - possono provenire da registri oppure da memoria
  - Indirizzamento diretto, indiretto, con registro base, ecc.
- Dimensione variabile delle istruzioni a seconda della modalità di indirizzamento di ogni operando
  - complessità di gestione della fase di prelievo o fetch in quanto a priori non è nota la lunghezza dell'istruzione da caricare.
- Elevata complessità della CPU
  - Rallentano i tempi di esecuzione delle operazioni.
  - Elevata profondità dell'albero delle porte logiche, utilizzato per la decodifica.

CPU – elementi principali

- **Registri:**
  - Banco di registri di uso generale (Register File)
    - Memoria ad accesso rapido, in cui memorizzare i dati di utilizzo più frequente
  - Program Counter (PC)
    - Contiene l'indirizzo dell'istruzione corrente da aggiornare durante l'evoluzione del programma, in modo da prelevare dalla memoria la corretta sequenza di istruzione
  - Instruction Register (IR)
    - Contiene l'istruzione in corso di esecuzione.
- **Arithmetic Logic Unit – (ALU)**
  - Effettua le operazioni aritmentico logiche fra registri, oppure direttamente dalla memoria (modalità di indirizzamento)
- **Unità di controllo**
  - "Cervello" della CPU
  - Coordina i flussi di informazione, in funzione dell'istruzione in corso
  - Seleziona l'operazione opportuna della ALU.



Sommario

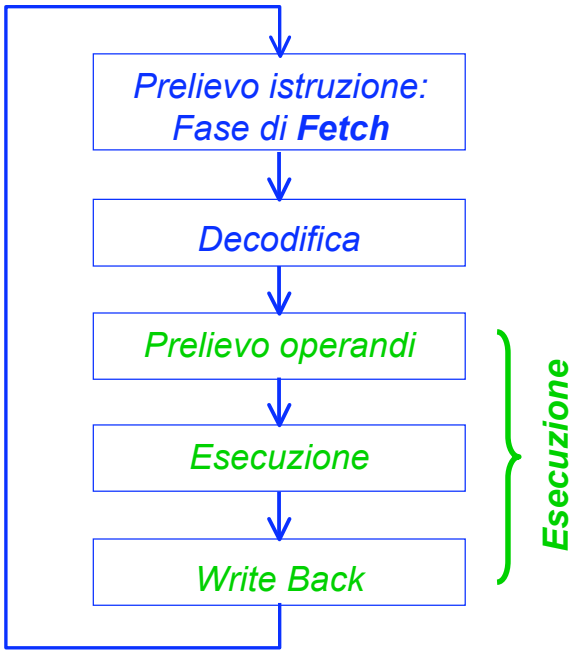
- Architettura di Von Neumann, architetture moderne.
- La CPU – il ciclo di esecuzione di un'istruzione.
- L'organizzazione della memoria. Il bus.

Unità centrale di elaborazione (CPU)

- *Compito della CPU: eseguire in sequenza le istruzioni che costituiscono il programma assegnato all'elaboratore.*
  - *ciclo di esecuzione di istruzioni*

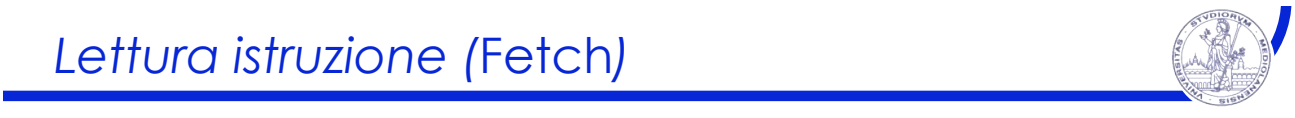
- Eseguire un'istruzione significa:
  - acquisire l'istruzione da eseguire: **FETCH**
  - interpretare l'istruzione e i dati a disposizione: **DECODIFICA**
  - eseguire calcoli / scelte / trasferimenti informazione **ESECUZIONE**

Ciclo esecuzione istruzione



- **Istruzioni e dati** risiedono nella **memoria principale**
- L'esecuzione di un programma inizia quando il registro **PC** punta alla prima istruzione del programma
- **MIPS**: suddivisione dell'esecuzione in tre sottofasi

Lettura istruzione (Fetch)



- Fase di **FETCH**
  - Il segnale di controllo per la **lettura** viene inviato alla memoria
  - Trascorso il tempo necessario all'accesso in memoria, la parola indirizzata (istruzione) viene letta dalla memoria e trasferita nel registro **IR**

Decodifica dell'istruzione

- DECODIFICA: l'istruzione contenuta nel registro IR viene decodificata per essere eseguita
- Predisposizione della CPU all'esecuzione dell'istruzione:
- Apertura delle vie di comunicazione appropriate
  - Lettura/scrittura di registri
  - Lettura/scrittura verso memoria
  - Lettura/scrittura dall'esterno (I/O)
  - …

Recupero degli operandi

- Se l'istruzione è, ad esempio, aritmetico-logica, è necessario recuperare gli operandi
  - possono essere nei registri di uso generale oppure in memoria
- Architetture a **registri**
  - Se un operando risiede in memoria, deve essere prelevato caricando l'indirizzo dell'operando e attivando un ciclo di READ della memoria.
  - L'operando letto dalla memoria viene trasferito alla ALU, che esegue l'operazione.
- Architetture **LOAD/STORE**
  - Le istruzioni di caricamento dalla memoria sono separate da quelle aritmetico/logiche.

Esecuzione dell'istruzione

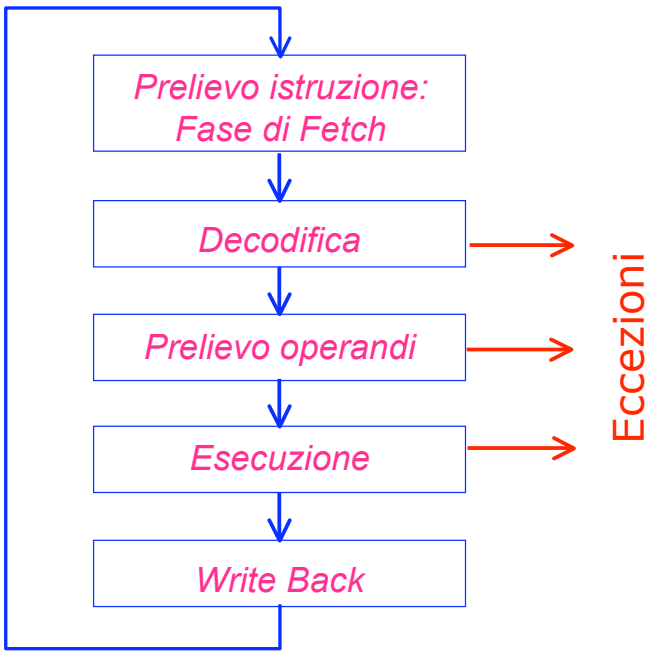
- Vengono selezionati i circuiti combinatori appropriati per l'esecuzione dell' istruzione e determinate in fase di decodifica.
- Esempio: `add $s3, $s2, $s1`
  - **Fase di fetch**: Caricamento dell'istruzione di **add**
  - **Decodifica**: Preparazione CPU a svolgere una somma.
  - **Lettura dati**: Indirizzamento adeguato del Register File.
  - **Esecuzione**: Esecuzione della somma.

Write-back: *Riscrittura risultato*

- Write-back:
  - Il risultato dell'operazione può essere memorizzato nei registri ad uso generale oppure in memoria.
  - Se il risultato dell'operazione deve essere posto in memoria, esso viene inviato al bus dati. L'indirizzo della posizione di memoria in cui scrivere il risultato viene bus indirizzi e si attiva un ciclo di scrittura (WRITE) della memoria.

- Chiusura ciclo:
  - Mentre viene eseguita un'istruzione, il contenuto del PC viene incrementato in modo da puntare alla prossima istruzione da eseguire.
  - Non appena è terminata l'esecuzione dell'istruzione corrente, si preleva l'istruzione successiva dalla memoria (fetch)

Ciclo di esecuzione di un'istruzione



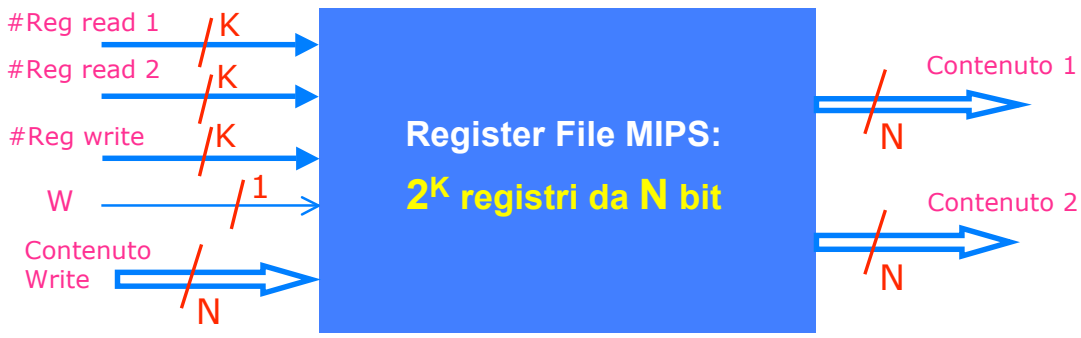
- Il normale flusso di esecuzione di un programma può essere interrotto da *Eccezioni*:
  - un segnale di interruzione esterno *(INTERRUPT)* per una richiesta di intervento.
  - Un segnale di errore di calcolo (es. divisione per 0)

Sommario

- Architettura di Von Neuman. Architetture moderne.
- Il ciclo di esecuzione di un'istruzione.
- L'organizzazione della memoria. Il bus.

Registri della CPU: Register File

- **Register file**: elemento di una CPU che contiene tutti i registri di uso generale (general purpose registers)
  - Banco di registri: 2^K **registri** da **N bit** ciascuno
  - Solitamente, **2 porte di uscita, 1 in ingresso**
- Operazioni:
  - SELEZIONE: fornendo in ingresso il numero del registro (#reg)
  - LETTURA: non modifica il contenuto del registro selezionato
  - SCRITTURA: Inserisce <ContenutoWrite> nel registro selezionato (comando: W)

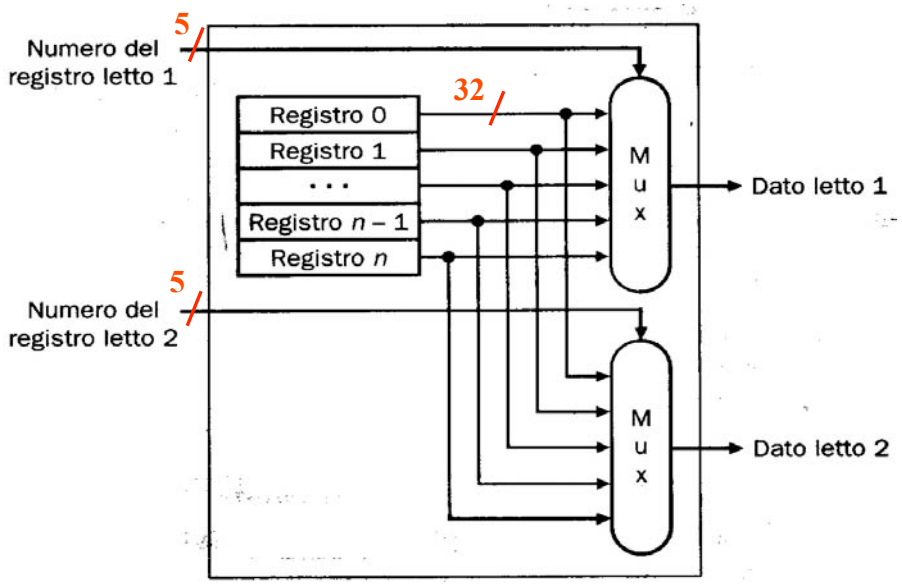


Register file (CPU **MIPS**)

- **Register File**: famiglia di CPU **"MIPS"**:
  - **32 registri da 32 bit**
  - **1** linea di **ingresso**
  - **2** linee di **uscita**

Register File MIPS: Porta di LETTURA

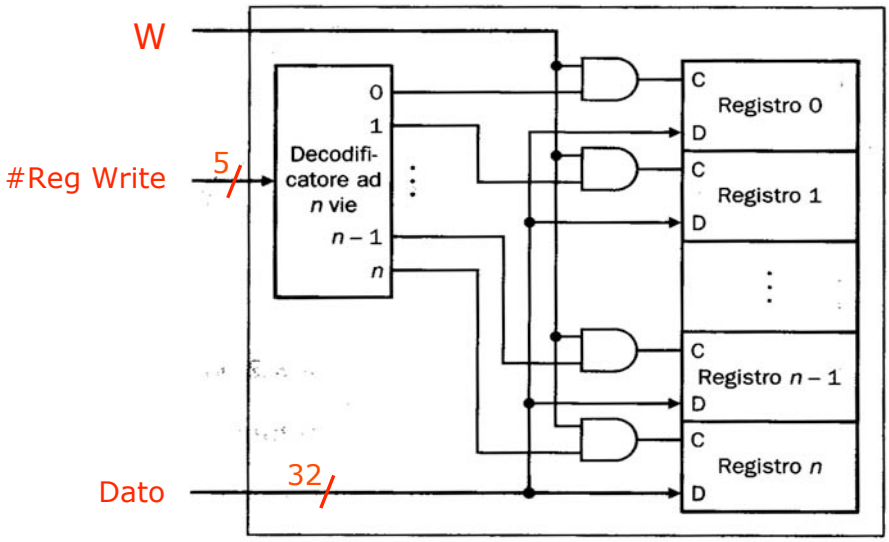
- 2 MUX di selezione registro
  - 2 registri possono essere letti contemporaneamente



Register file MIPS: Porta di SCRITTURA

- **Ingresso CLK:** (Selezione) AND **W**
  - Se no ad ogni CLK scriverei nei registri

- **Ingresso D:**
  - Dato (32 bit)



Indirizzi nella memoria principale

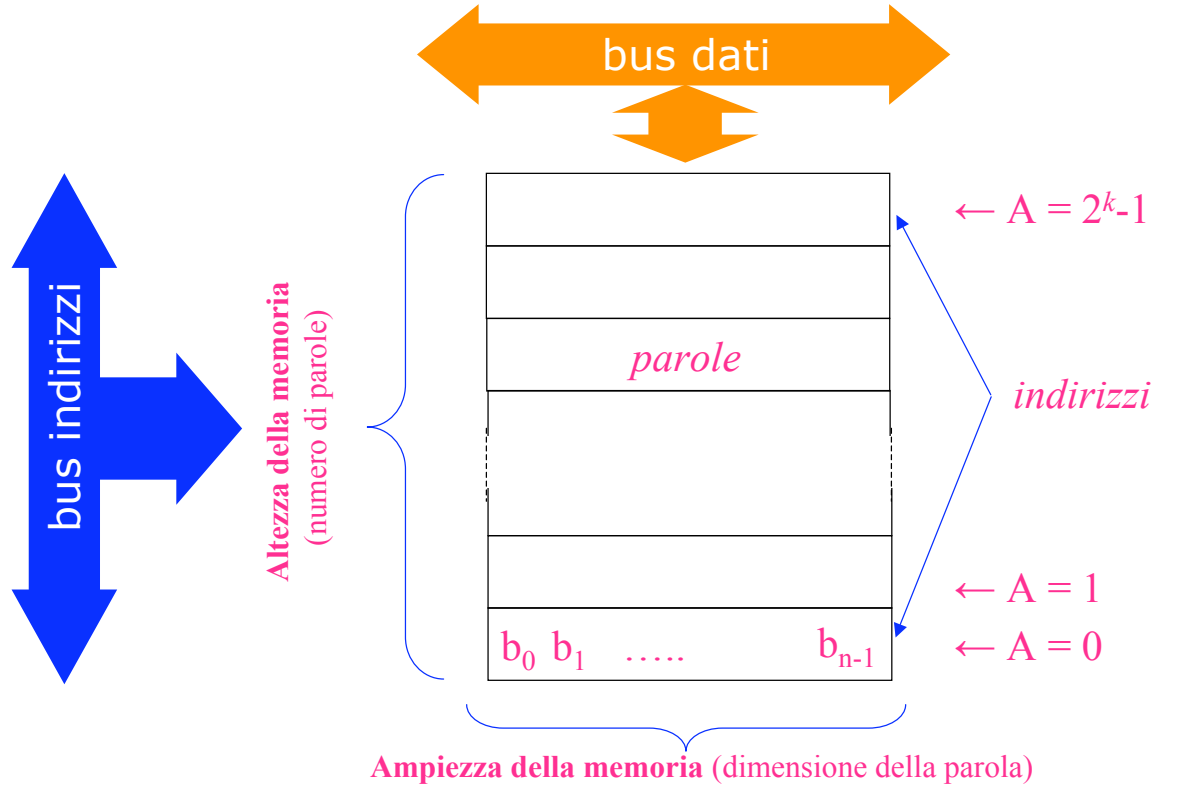
- La memoria è organizzata in ***2^K parole (word)*** di ***N bit***
  - ***N: ampiezza*** *della memoria*
  - ***2^K: altezza*** *della memoria*

- In genere, la dimensione della parola di memoria coincide con la dimensione dei registri della *CPU (CPU word)*
  - Le operazioni di *load/store* avvengono in un **singolo ciclo**

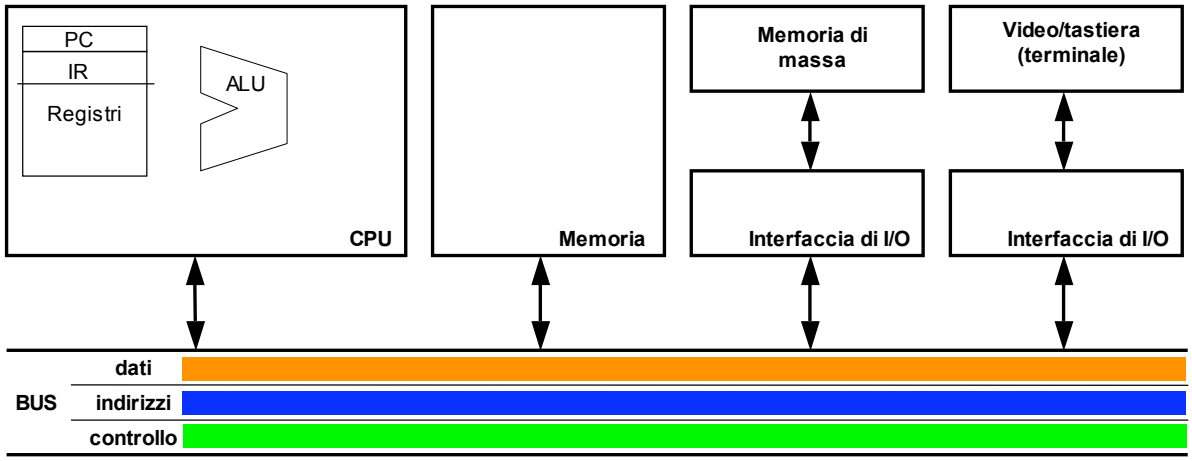
- **Capacità della memoria =** *(N.words)* x *(dim.word [bytes])*
  - ***k = 32 → (2^{32}= 4 Gwords)*** x ***(32 bit = 4 bytes) = 16 Gbytes***

- *Random Access Memory –* ***RAM***
  - **Il tempo di accesso alla memoria** è **fisso e indipendente dalla posizione della parola** alla quale si vuole accedere.

Indirizzi nella memoria principale



Collegamento tra componenti mediante BUS



Connessione *a nodo comune: BUS*

Tutte le unità del calcolatore sono connesse al bus

Bus di sistema

- *BUS: infrastruttura di comunicazione tra le diverse unità del calcolatore.*

  Generalmente composto da tre parti:

  - ***Bus dati***: le linee per trasferire dati e istruzioni da/verso i dispositivi.
  - ***Bus indirizzi***: su cui la *CPU* trasmette l'indirizzo di memoria da cui prelevare/depositare il dato nel caso di lettura/scrittura dalla memoria.
  - ***Bus di controllo***: informazioni ausiliarie per la corretta definizione delle operazioni da compiere e per la sincronizzazione tra *CPU* e memoria

- *Esempio: lettura dalla memoria*

  1. *La CPU fornisce l'indirizzo della parola desiderata sul bus indirizzi*
  2. *viene richiesta l'operazione di **lettura** attivando il bus di controllo.*
  3. *Quando la memoria ha reso disponibile la parola richiesta, il dato viene trasferito sul bus dati e la CPU può prelevarlo dal bus dati ed utilizzarlo nelle sue elaborazioni*