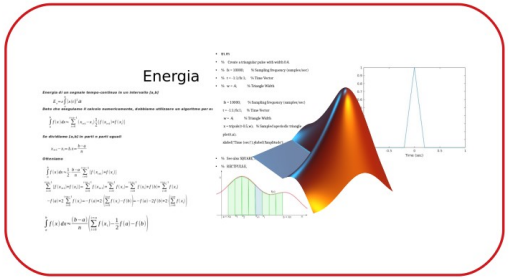
UNIVERSITÀ DEGLI STUDI DI TRIESTE

Programmazione Informatica

Ingegneria Civile e Ambientale
Ingegneria Navale

Teoria 4 – Architettura dei calcolatori. HW, SW, Sistemi Operativi



Antonio Fiumara
antonio.fiumara@dia.units.it

A. A. 2025-26

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Struttura del Corso

Gli argomenti trattati nel corso saranno:

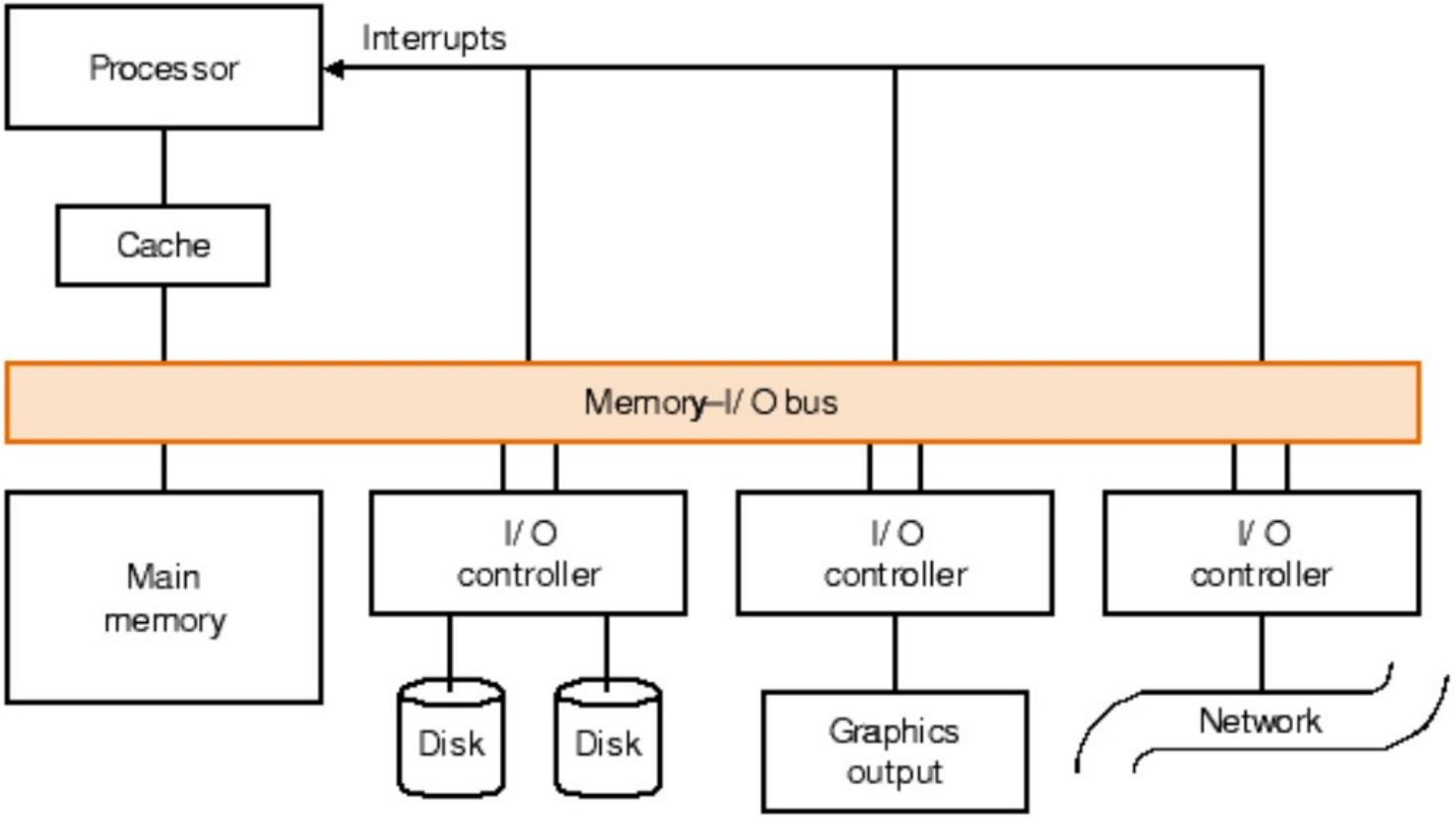
- Lezione Teoria 1 – Introduzione, presentazione del corso, codifica dell'informazione
- Lezione Teoria 2 – Rappresentazione dell'informazione in un computer
- Lezione Teoria 3 – Algebra booleana, introduzione e applicazioni nella programmazione
- **Lezione Teoria 4 – Architettura dei calcolatori. HW, SW, Sistemi Operativi**
- Lezione Teoria 5 – Algoritmi e strutture dati
- Lezione Teoria 6 – Implementazione di algoritmi notevoli (ordinamento , ricerca, ecc)
- Lezione Programmazione - Matlab 1 – Introduzione, installazione, comandi di base, matrici
- Lezione Programmazione - Matlab 2 – Input/Output (I/O), strutture di controllo e funzioni
- Lezione Programmazione - Matlab 3 – Sistemi Lineari e Algebra lineare
- Lezione Programmazione - Matlab 4 – Analisi matematica
- Lezione Programmazione - Matlab 5 – Funzioni nel dominio del tempo e della frequenza
- Lezione Programmazione - Matlab 6 – Analisi ed elaborazione dei segnali
- Lezione Programmazione - Matlab 7 – Ripasso e approfondimenti
- Esonero esame Programmazione Matlab (3/12/25)

Introduzione

Architettura di un calcolatore elettronico

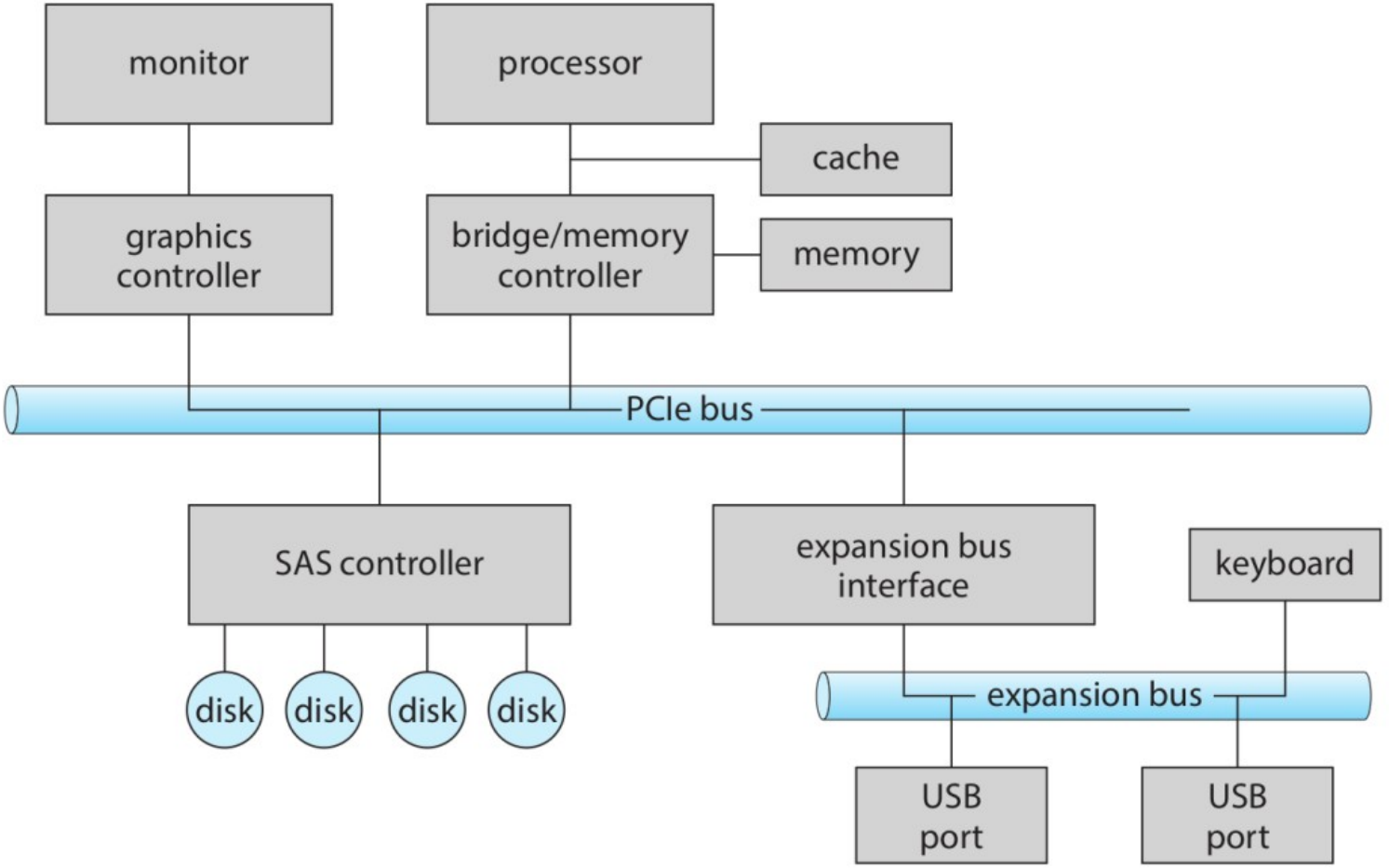
UNIVERSITÀ DEGLI STUDI DI TRIESTE

Architettura di un calcolatore



UNIVERSITÀ DEGLI STUDI DI TRIESTE

Architettura di un calcolatore – I/O System



UNIVERSITÀ DEGLI STUDI DI TRIESTE

Componenti principali di un calcolatore elettronico

1. Il **processore** esegue istruzioni
2. La **memoria principale** (RAM/ROM) contiene i dati e la codifica del programma
3. Dispositivi/periferiche di **input/output** (I/O). Esempi:
   - tastiera (I)
   - mouse (I)
   - schermo (O)
   - microfono (I)
   - touch screen (I/O)
   - stampante (O)
   - altoparlante (O)
   - memoria di massa/disco (I/O)
   - interfaccia di rete (I/O)
4. Interconnessione fra le varie componenti (**Memory bus – I/O bus**)

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Componenti principali di un calcolatore elettronico

1. Il processore esegue istruzioni

Codifica delle istruzioni: cosa sono le istruzioni e come vengono codificate e rappresentate?

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Componenti principali di un calcolatore elettronico

Abbiamo visto che i dati (testo, immagini, numeri, etc) sono tradotti in sequenza di byte tramite opportuna codifica

Anche le istruzioni devono essere codificate.

Esempio: codifica delle ricette (vedi in seguito)

Un programma è una sequenza di istruzioni

Ogni istruzione può essere codificatata in sequenza di byte

(programma è sequenza di istruzioni)+(istruzione è sequenza di byte)=

Un programma può quindi essere codificato in sequenza di bytes: file eseguibile

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Codifica delle istruzioni

Codifica degli ingredienti

Codice (Hex)	Ingrediente
00	acqua
01	olio
02	sale
03	pasta
04	passata di pomodoro
05	aglio

Codifica dei recipienti

Codice (Hex)	Recipienti
00	pentola
01	padella
02	scolapasta
03	piatto

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Codifica delle istruzioni

Codifica delle “istruzioni”

Codice (Hex)	Istruzione
00 x y z	versare y grammi di ingrediente x in recipiente z
01 x y	versare contenuto del recipiente x in recipiente y
02 x	spostare recipiente x sul fuoco
03 x	togliere recipiente x dal fuoco
04 x	spostare recipiente x sul lavandino
05 x	mescolare recipiente x
06 x	aspettare x minuti
07 x	mangiare contenuto del recipiente x

x, y, z sono detti operandi dell’istruzione

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Codifica delle istruzioni

Esercizio 1:
a cosa corrisponde il seguente codice?
00 00 FF 00 00 00 FF 00 02 00 06 0A 00 02 0A 00 00 03
64 00 06 0A 04 02 01 00 02 07 02

suggerimento: prima individuare le singole istruzioni nel codice

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Codifica delle istruzioni

Esercizio 1:
a cosa corrisponde il seguente codice?
00 00 FF 00 **02** 00 **06** 3B **00** 03 3F 00 **00** 02 23 00 **06** 0A **01** 00 02 **01** 02 03 **00** 01 03 **05** 03 **07** 03

suggerimento: prima individuare le singole istruzioni nel codice

Codice (Hex)	Istruzione
00 x y z	versare y grammi di ingrediente x in recipiente z
01 x y	versare contenuto del recipiente x in recipiente y
02 x	spostare recipiente x sul fuoco
03 x	togliere recipiente x dal fuoco
04 x	spostare recipiente x sul lavandino
05 x	mescolare recipiente x
06 x	aspettare x minuti
07 x	mangiare contenuto del recipiente x

Codice (Hex)	Recipienti
00	pentola
01	padella
02	scolapasta
03	piatto
Codice (Hex)	**Ingrediente**
00	acqua
01	olio
02	sale
03	pasta
04	passata di pomodoro
05	aglio

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Codifica delle istruzioni

Esercizio 2: come si codifica la ricetta della pasta al pomodoro

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Codifica delle istruzioni

Analogia Cucina/Calcolatore

Cucina	Calcolatore
Cuoco	Processore
Recipienti, ingredienti	dati
Cucina, Dispensa	Memoria (contiene dati e istruzioni)
Ricetta	Programma

- Il linguaggio di programmazione permette di descrivere un programma: Italiano (in cucina), C, Java, Matlab (con il calcolatore)
- Il compilatore traduce un programma espresso in un certo linguaggio di programmazione, in linguaggio macchina (01 F5 05 04 . . .)

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Memoria principale

La memoria del computer è costituita dai dispositivi nei quali vengono memorizzati e dati e i programmi. Possiamo distinguere in

- **Memoria interna alla CPU**
  - Registri
  - Cache
  - Caratterizzata da alta velocità (comparabile con quella del processore) e limitate dimensioni
- **Memoria primaria o centrale**
  - La memoria centrale contiene i dati e i programmi in corso di elaborazione ed è composta dalla RAM (random access memory) e dalla ROM, una memoria a sola lettura (ROM, read only memory)
- **Memoria secondaria o periferica**
  - Dispositivi di memoria che possono essere letti o scritti nei quali è possibile memorizzare dati e programmi in modo permanente.

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Memoria principale

La memoria centrale contiene i dati e i programmi in corso di elaborazione

- RAM (random access memory)
  - è un tipo di memoria **volatile** caratterizzata dal permettere l'accesso **diretto** a qualunque indirizzo di memoria. La RAM si contrappone alla memoria ad accesso sequenziale, in cui i dati sono disposti in modo sequenziale e per accedervi è necessario scorrere su tutti i dati precedenti.
  - L’accesso diretto implica che il tempo necessario per leggere o scrivere un dato è indipendente dall’indirizzo di memoria
  - Volatile significa che i dati vengono persi se si toglie l’alimentazione
  - Ha una dimensione: quantità di dati che possono essere memorizzati (in Byte o multipli di Byte)
  - tempo di accesso: dipende dalla tecnologia e varia a seconda dell’operazione (lettura più veloce della scrittura)

https://it.wikipedia.org/wiki/RAM

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Memoria principale

La memoria principale (RAM) contiene un lungo elenco di bytes che codificano:
- dati del programma
- codifica delle istruzioni da eseguire

in ordine sparso (ma noto) ogni byte ha un indirizzo che permette di accedervi

Indirizzo	Dato
0	00
1	c2
2	c7
3	58
...	...

si dice: “la locazione 3 di memoria contiene il byte 58”

- Accesso sia in lettura che scrittura mediante l’indirizzo
- “Random-Access” perché offre le stesse prestazione indipendentemente dalla locazione di memoria a cui si accede e per distinguerle dalla memoria ad accesso sequenziale

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Memoria principale

Caratteristiche della RAM:
1) ogni byte può essere scritto o letto specificando il suo indirizzo
2) è volatile: quando non viene più alimentata perde il suo contenuto
3) ha una dimensione: quanti dati possono essere memorizzati (in Byte o suoi multipli)
4) tempo di accesso: varia a seconda dell'operazione (lettura più veloce della scrittura)
Quanti bit sono necessari per codificare l'indirizzo di una memoria di 4 GiB?

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Memoria principale

Caratteristiche della RAM:
1) ogni byte può essere scritto o letto specificando il suo indirizzo
2) è volatile: quando non viene più alimentata perde il suo contenuto
3) ha una dimensione: quanti dati possono essere memorizzati (in Byte o suoi multipli)
4) tempo di accesso: varia a seconda dell'operazione (lettura più veloce della scrittura)

Quanti bit sono necessari per codificare l'indirizzo di una memoria di 4 GiB?

4 GiB = 4×2^{30} bytes = 2^{32} bytes

La memoria contiene 2^{32} bytes. Servono quindi 32 bit per indirizzare i byte del suo contenuto

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Read Only Memory

ROM

Considerato che il contenuto della RAM viene cancellato ad ogni spegnimento, come possiamo memorizzare dei dati che siano usati durante all’accesione (fase di boot)?
Read-Only Memory (ROM)
non è volatile: mantiene il suo contenuto anche se non alimentata

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Read Only Memory

è memoria **non volatile** in cui i dati sono memorizzati tramite collegamenti elettronici fisici e stabili e vengono mantenuti anche in assenza di alimentazione

- È ad accesso diretto (come la RAM)
- Il suo contenuto non è modificabile durante il normale funzionamento
- La memorizzazione dei dati e programmi può essere attuata, con diverse tecniche, in fase di progettazione, prototipazione, costruzione e in alcuni casi, anche attraverso opportuni dispositivi elettronici.
- Esistono diverse tipologie di ROM, alcune di queste sono cancellabili e riscrivibili con appositi strumenti
  - EPROM (Erasable Programmable ROM)
  - EEPROM (Electrically Erasable Programmable ROM)
  - FLASH ROM, basata sulla memoria a stato solido non volatile denominata Flash Memory

https://it.wikipedia.org/wiki/RAM

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Organizzazione della memoria

La memoria centrale è organizzata in “celle di memoria” raggiungibili da un “indirizzo di memoria”.

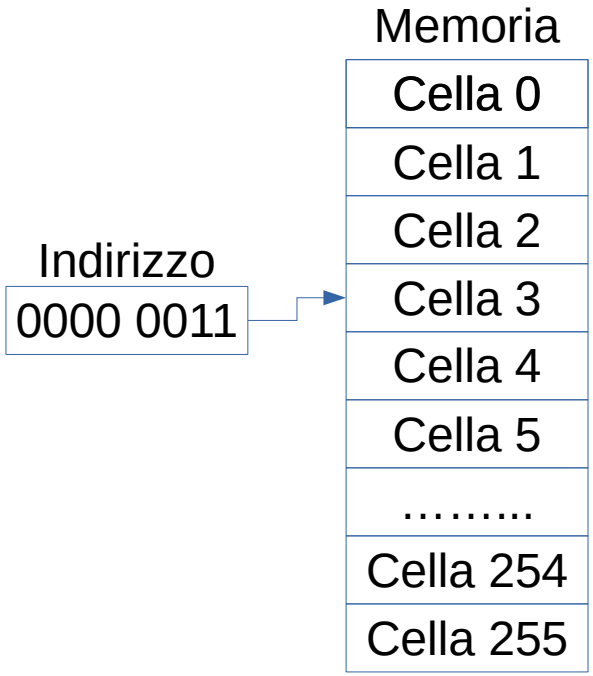
- Le celle di memoria sono costituite da Byte o da multipli di Byte
  - 8 bit (1 byte)
  - 16 bit (2 byte)
  - 32 bit (4 byte)
  - 64 bit (8 byte)
- L’indirizzo di memoria deve essere codificato in modo tale da poter “indirizzare” ogni cella di memoria
- Esistono diverse modalità di indirizzamento della memoria, noi per semplicità faremo riferimento all’indirizzamento lineare
- L’indirizzo codificato in binario puro su n bit può indirizzare 2^n celle di memoria

  Ad esempio, se un computer ha una memoria di 4 GiB = 4×2^{30} byte = 2^{32} byte,

  la memoria contiene 2^{32} byte. Per l’indirizzo servono quindi 32 bit.

UNIVERSITÀ DEGLI STUDI DI TRIESTE

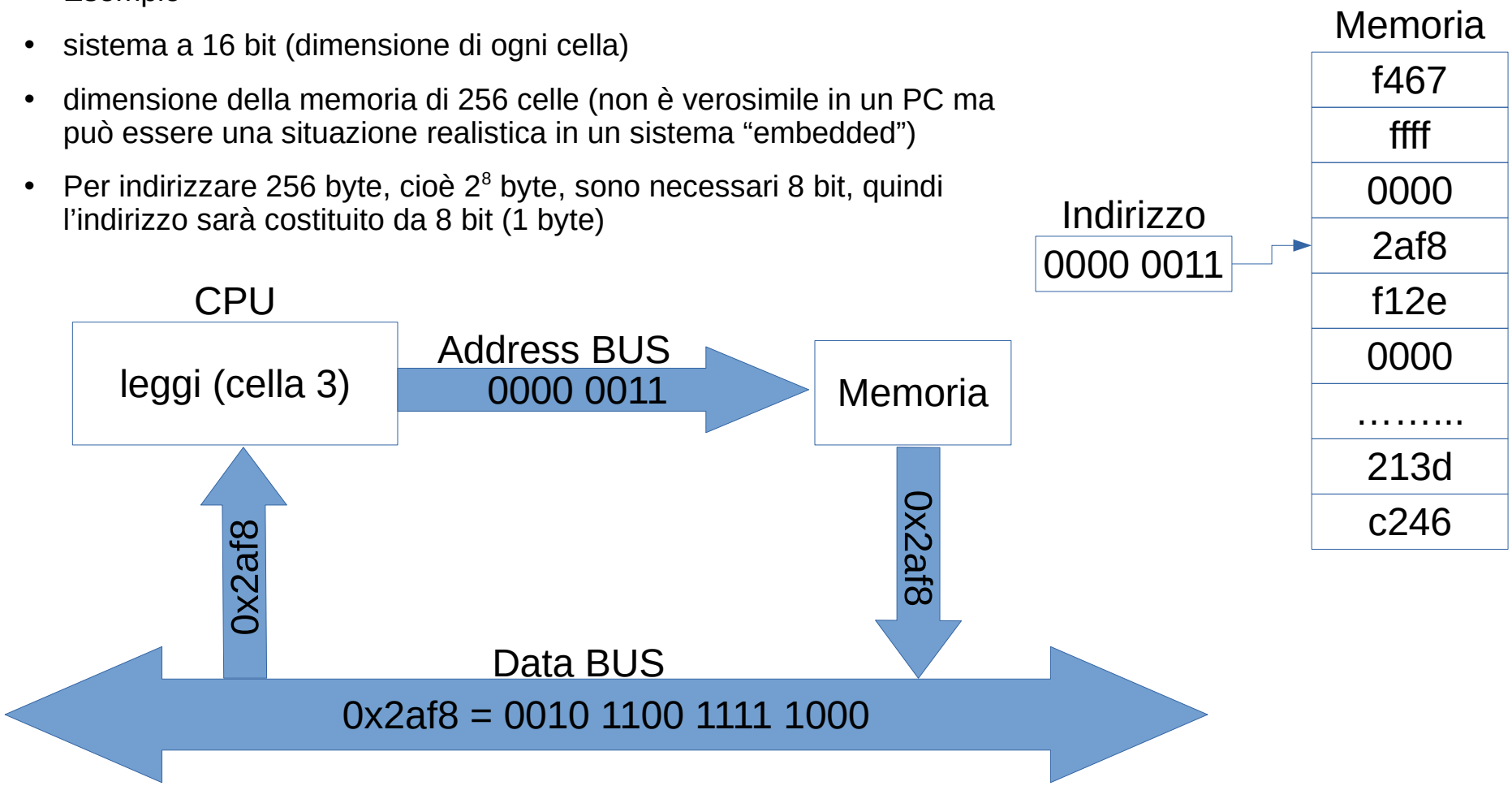
Organizzazione della memoria



Organizzazione della memoria

Esempio

- sistema a 16 bit (dimensione di ogni cella)
- dimensione della memoria di 256 celle (non è verosimile in un PC ma può essere una situazione realistica in un sistema “embedded”)
- Per indirizzare 256 byte, cioè 2^8 byte, sono necessari 8 bit, quindi l’indirizzo sarà costituito da 8 bit (1 byte)



UNIVERSITÀ DEGLI STUDI DI TRIESTE

Organizzazione della memoria

Dispositivi di memoria che possono essere letti o scritti nei quali è possibile memorizzare dati e programmi in modo permanente.

- Dischi
  - Magnetici: dischi fissi (Hard disk), floppy disk, nastri
  - Ottici: CD-ROM, DVD-ROM
  - Magneto-ottici
- Flash memory: memoria a stato solido con tempi di accesso relativamente veloci
  - CompactFlash, SmartMedia, MultiMediaCard, Memory Stick
  - Secure Digital
  - MicroSD
  - xD-Picture Card
  - Chiavette USB
  - Disk on module (SSD)

3,5 pollici SSD.
Operating system concepts. Abraham Silberschatz e altri

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Processore (CPU)

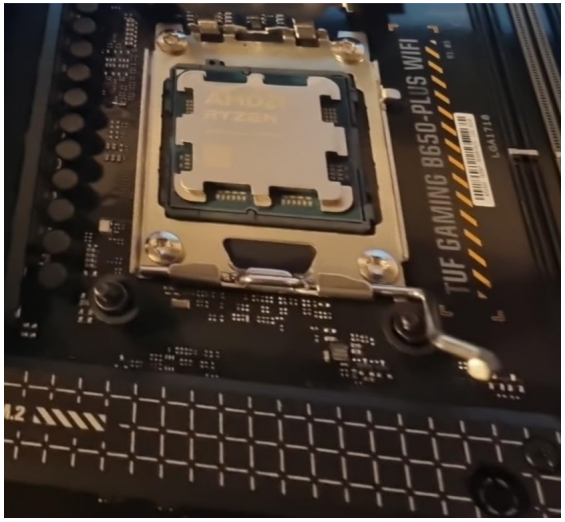
- Il processore esegue continuamente istruzioni (codificate) prelevate dalla memoria
- Per motivi di efficienza (velocità), le operazioni non si possono operare direttamente sulla memoria
- non esiste un'istruzione "aggiungi il contenuto dell'indirizzo x al contenuto dell'indirizzo y"
- Per svolgere una tale operazione, il processore ha a disposizione dei registri
- i registri si possono vedere come particolari memorie interne al processore, molto più veloci della RAM (come un "piano di appoggio" in cucina)
- i registri non hanno un indirizzo (come le locazioni di memoria), ma sono usati con il loro nome. Esempio (su Intel processors): EAX, EBX, etc.
- L'operazione precedente, quindi, si può svolgere così:
  - 1 "copia il contenuto dell'indirizzo x nel registro EAX"
  - 2 "aggiungi il contenuto del registro EAX al contenuto dell'indirizzo y"
- Esistono altri registri che svolgono altre funzioni: registri di stato, registri per l'accesso alla memoria, etc.

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Processore

Per svolgere la propria funzione (la continua esecuzione di istruzioni), un processore è dotato delle seguenti componenti

- un insieme di registri (elementi di memoria)
- alcune unità che svolgono le operazioni:
- Arithmetic Logic Unit (ALU), per svolgere operazioni come la somma, operazioni binarie (AND, OR, NOT, XOR, etc.)
- Floating Point Unit, per svolgere operazioni con numeri in virgola mobile (altro tipo di numeri rappresentabile)
- Control Unit, per controllare l'esecuzione delle istruzioni



UNIVERSITÀ DEGLI STUDI DI TRIESTE

Processore

- Un registro particolare, il Program Counter (PC), viene utilizzato per eseguire le istruzioni
- Il Program Counter contiene l'indirizzo di memoria da cui prelevare la prossima istruzione (ricorda: un'istruzione è una sequenza di byte)
- Dal momento dell'accensione, fino al suo spegnimento il processore esegue per sempre il seguente ciclo:
  - 1 fase di fetch: preleva l'istruzione dalla locazione di memoria indicata dal registro PC e la copia in un suo registro interno (Instruction Register)
  - 2 fase di decode: decodifica l'istruzione (deve capire cosa fare) e incrementa il PC di un numero uguale alla lunghezza dell'istruzione decodificata (ricorda: nell'esempio della cucina le istruzioni avevano lunghezza variabile da 2 a 4 byte)
  - 3 fase di execute: esegue l'istruzione e torna al primo passo
- Le istruzioni hanno bisogno degli operandi per essere eseguite
- All'accensione del calcolatore il PC vale 0: il processore inizia a prelevare, decodificare ed eseguire dalla prima locazione di memoria
- Esempio: illustrare le varie fasi con la ricetta

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Processore

Processore: velocità
La velocità di esecuzione è determinata dalla frequenza del clock del processore
Il clock si misura in x Hertz e corrisponde al numero di cicli di clock in un secondo: un clock di 3 GHz corrisponde a 3×10^9 cicli di clock al secondo
Il numero di cicli necessari per eseguire una singola istruzione varia a seconda di vari fattori fra cui:

- la tipologia di istruzione (Es: fare una moltiplicazione richiede più cicli di un AND)
- la disponibilità degli operandi (Es: usare operandi che si trovano nei
- registri è più veloce rispetto all'uso di operandi in memoria)
- la tipologia di ISA (Es: su RISC la singola operazione è più veloce che su CISC)
- Il numero di cicli richiesto per svolgere un'operazione
  -su architettura CISC (complessa) varia da 2 a 20, circa
  -su architettura RISC (semplice) varia da 1 a 4, circa

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Processore architettura RISC e CISC

Possono esistere modi diversi per codificare le istruzioni:
1) si può scegliere una codifica “ricca” in cui codifico tante istruzioni
2) oppure una codifica “povera” con poche istruzioni codificate
(ricorda: esempio di codifica “ricca” dei caratteri cinesi, rispetto a codifica “povera” dei caratteri latini)

Complex Instruction Set Computer (CISC)	Reduced Instruction Set Computer (RISC)
tante istruzioni codificate	poche istruzioni codificate
stesso programma scritto con poche (complesse) istruzioni	stesso programma scritto con tante (semplici) istruzioni
lenta esecuzione di una istruzione complessa	veloce esecuzione di una istruzione semplice
Esempio di processore CISC: Intel Core i7 (tipicamente laptop/desktop computers)	Esempio di processore RISC: ARM (tipicamente su smartphone)

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Processore

ISA, linguaggi, compilatore
La scelta della codifica delle istruzioni determina l'**Instruction Set Architecture (ISA)**
CISC e RISC sono due famiglie di ISA
la scelta dell'ISA è influenzata da motivi commerciali, caratteristiche delle applicazioni, etc.
la codifica delle istruzioni si chiama anche linguaggio macchina

- scrivere un programma in linguaggio macchina è decisamente faticoso e "error-prone"
- I linguaggi di programmazione permettono di esprimere un programma in modo più naturale rispetto al linguaggio macchina
- Esempi di linguaggi di programmazione: C, Java, Pascal, Fortran, etc.
- Il compilatore traduce un programma scritto in un linguaggio di programmazione, in codice eseguibile (sequenza di codifiche delle istruzioni)

Domanda:
1 Dato un programma, conterrà un numero maggiore di istruzioni il codice compilato sopra una ISA CISC oppure RISC?

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Linguaggi di programmazione

Un linguaggio di programmazione è un insieme strutturato di regole e simboli che permettono ai programmatori di scrivere istruzioni comprensibili dal computer.
L'obiettivo di questi linguaggi è rendere più facile e intuitivo il processo di creazione di programmi che realizzano operazioni specifiche.
Come abbiamo visto, le istruzioni sono tradotte in linguaggio macchina ed eseguite dalla CPU.

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Linguaggi di programmazione - Classificazione

- **Linguaggi di alto livello:** Sono linguaggi progettati per essere comprensibili dagli esseri umani. La loro sintassi è vicina al linguaggio naturale, rendendo lo sviluppo di software più veloce ed efficiente. Offrono funzionalità come la gestione automatica della memoria e librerie standard per semplificare lo sviluppo. Sono (quasi) indipendenti dall'hardware e necessitano di essere tradotti in linguaggio macchina tramite un compilatore o un interprete.

- **Linguaggi di basso livello**: Sono più vicini al linguaggio macchina e offrono un maggiore controllo diretto sull'hardware. Rispetto ai linguaggi di alto livello, sono più difficili da imparare e utilizzare, ma consentono ottimizzazioni più fini del codice. Sono specifici per ogni architettura e sono eseguiti più velocemente.

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Linguaggi di programmazione - Classificazione

Compilatori e Interpreti

Per tradurre il **codice sorgente** (scritto dal programmatore) in codice eseguibile dalla CPU, sono necessari strumenti che eseguano questa traduzione.
Un **compilatore** è un programma che prende in ingresso il codice sorgente di un programma e lo traduce completamente in linguaggio macchina prima che il programma venga eseguito.
Questa traduzione crea un file **eseguibile**, che può essere lanciato in un secondo momento senza la necessità di ritradurre il codice.
Un **interprete** è un programma che legge e interpreta il codice sorgente linea per linea o istruzione per istruzione, traducendo ed eseguendo il codice in tempo reale senza produrre un file eseguibile separato.
Esempi di linguaggi compilati sono C, Fortran, Pascal, C++, Visual Basic. I linguaggi interpretati includono: Matlab, Python, JavaScript, Perl, PHP.
Esistono anche linguaggi ibridi come Java, che vengono prima compilati in bytecode e poi interpretati da una macchina virtuale (JVM).

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Linguaggi di programmazione - Classificazione

Caratteristica	Compilatore	Interprete
Modalità di traduzione	Traduce l'intero programma in linguaggio macchina prima dell'esecuzione.	Traduce ed esegue il programma istruzione per istruzione.
Tempo di esecuzione	Il codice tradotto viene eseguito in modo indipendente dal compilatore, quindi l'esecuzione è molto veloce.	L'esecuzione è generalmente più lenta, perché ogni istruzione deve essere tradotta al momento dell'esecuzione.
File eseguibile	Genera un file eseguibile	Non genera un file eseguibile. L'interprete deve essere presente ogni volta che si esegue il programma.
Diagnosi di errori	Gli errori vengono rilevati prima dell'esecuzione, durante la fase di compilazione.	Gli errori vengono rilevati durante l'esecuzione, istruzione per istruzione.
Efficienza	Poiché tutto il codice è tradotto in un'unica volta, i programmi compilati tendono a essere più efficienti in fase di esecuzione.	L'esecuzione può essere più lenta, poiché ogni istruzione viene interpretata "al volo".

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Compilazione

1. Analisi lessicale (Lexer): il compilatore legge il codice sorgente e lo scompone in unità più piccole chiamate token. Ogni token rappresenta un simbolo di base del linguaggio, come una parola chiave (ad esempio if, while), un identificatore (come il nome di una variabile), un operatore o un delimitatore (ad esempio ;).
2. Analisi sintattica (Parser): il compilatore verifica se la sequenza di token generata dall'analisi lessicale segue le regole grammaticali (sintassi) del linguaggio di programmazione. Costruisce una struttura ad albero chiamata albero sintattico o albero di derivazione che rappresenta la struttura gerarchica del codice.
3. Analisi semantica: il compilatore verifica il significato del codice, ovvero se ogni istruzione ha un senso nel contesto del programma. Ad esempio, verifica che le variabili siano dichiarate prima di essere utilizzate, che i tipi di dati siano compatibili e che le operazioni aritmetiche siano applicate correttamente.

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Compilazione

4. Ottimizzazione: è una fase facoltativa in cui il compilatore cerca di migliorare il codice generato, riducendone la dimensione o migliorandone le prestazioni. L'ottimizzazione può essere applicata sia a livello di codice intermedio (durante la traduzione) sia a livello di codice macchina finale.
5. Generazione del codice: il compilatore genera il codice oggetto o codice macchina. Questo è il codice che sarà eseguito direttamente dalla CPU. In questa fase, vengono assegnati indirizzi di memoria alle variabili e le istruzioni vengono tradotte in sequenze binarie.
6. Linking: il compilatore collega insieme vari pezzi di codice (come librerie o funzioni definite in file separati) per formare il file eseguibile finale. Questa fase è particolarmente importante per i linguaggi come C e C++, dove il programma può essere suddiviso in più file sorgente.

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Interprete

Un interprete, a differenza di un compilatore, traduce il codice sorgente linea per linea o istruzione per istruzione, ed esegue immediatamente il codice tradotto senza produrre un file eseguibile. Anche se in generale i programmi interpretati sono più lenti di quelli compilati, gli interpreti offrono alcuni vantaggi significativi.
Il codice sorgente interpretato può essere eseguito su qualsiasi piattaforma che abbia l'interprete appropriato. Non è necessario ricompilare il programma per ogni architettura hardware.
Poiché il codice viene eseguito linea per linea, è possibile interrompere l'esecuzione in qualsiasi momento e correggere eventuali errori senza dover ricompilare tutto il programma.
Un linguaggio interpretato è uno strumento ideale per la sperimentazione e rende la fase di debugging molto semplice.

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Interprete

Non è necessario attendere la fase di compilazione. Gli interpreti sono spesso utilizzati in linguaggi di scripting (come Matlab, Python e JavaScript) per eseguire rapidamente piccoli programmi o script.

Per eseguire un programma interpretato, è necessario che l'interprete sia presente sulla macchina. Il codice non può essere eseguito autonomamente.

UNIVERSITÀ DEGLI STUDI DI TRIESTE

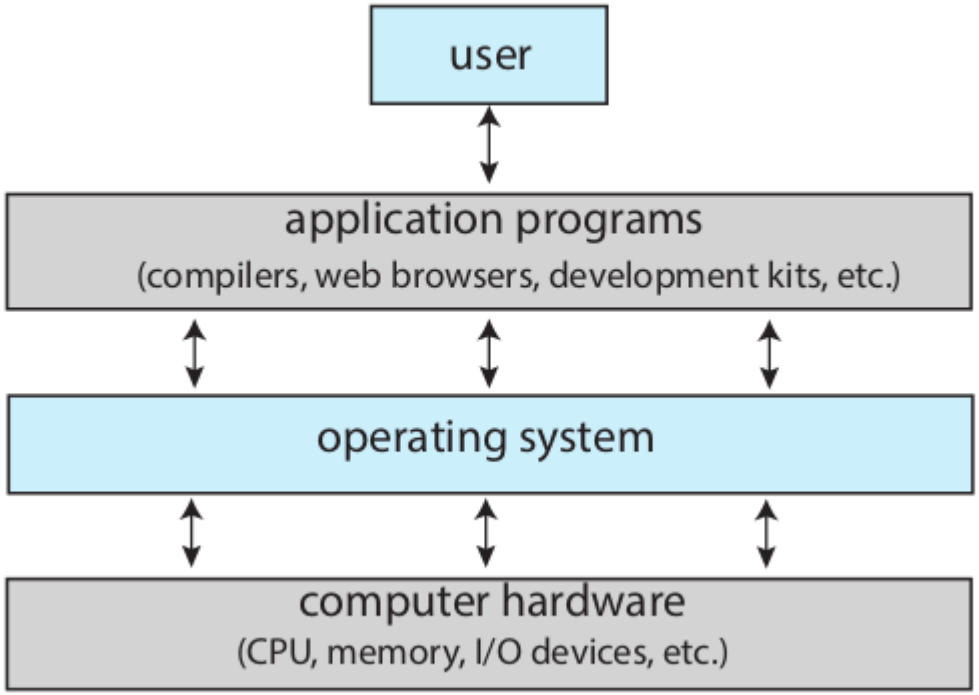
Paradigma di programmazione

Modello di sviluppo software che definisce come i problemi dovrebbero essere risolti e come i programmi dovrebbero essere strutturati. Esistono diversi paradigmi che influenzano la sintassi e la semantica dei linguaggi di programmazione.

- **Programmazione imperativa**: i programmi consistono in una sequenza di istruzioni che il computer esegue passo dopo passo. Si basa sull'uso di variabili, cicli e strutture di controllo per modificare lo stato del programma (Matlab, C, C++, Java, Python).
- **Programmazione dichiarativa**: invece di specificare esattamente come risolvere un problema, si descrive il risultato desiderato e il linguaggio determina come raggiungere tale risultato. Si concentra più sul "cosa fare" piuttosto che sul "come farlo" (SQL, Prolog).
- **Programmazione orientata agli oggetti**: i programmi sono organizzati attorno agli oggetti, che rappresentano entità reali o astratte e contengono dati e metodi (Java, C++, Python, Matlab).
- **Programmazione procedurale**: organizza il codice in una serie di istruzioni o procedure, spesso chiamate funzioni o subroutine, che vengono eseguite in modo sequenziale per raggiungere un obiettivo o risolvere un problema.

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Sistema operativo



UNIVERSITÀ DEGLI STUDI DI TRIESTE

Funzioni del sistema operativo

Un sistema operativo è un software che gestisce l'hardware di un computer. Esso fornisce inoltre una base per i programmi applicativi e funge da intermediario tra l'utente del computer e l'hardware del computer.

- Esistono molti modi di affrontare questi compiti e una vasta tipologia di sistemi operativi.
- I sistemi operativi sono ormai diffusi in molti ambiti,
  - Personal computer
  - Automobili
  - Elettrodomestici
  - Smartphone,
  - Tablet
  - Smart TV
  - Server aziendali
  - Server di cloud computing
- Una responsabilità fondamentale di un sistema operativo è quella di allocare le "risorse" del computer ai programmi e, nei sistemi "multi tasking", gestire la priorità e la sincronizzazione dei tasks

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Funzioni del sistema operativo

Personal Computer

- Linux
- Unix like
- Microsoft Windows
- Apple MacOs
- BSD
- Google Chrome OS (derivato da Linux)

Smartphone e tablet

- Apple IOS
- Android
- Microsoft Windows Phone
- Sailfish OS
- Symbian OS

https://it.wikipedia.org/wiki/Sistema_operativo

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Energia

Energia di un segnale tempo-continuo in un intervallo [a,b]

$$E_x = \varepsilon \int_a^b |x(t)|^2 dt$$

Dato che eseguiamo il calcolo numericamente, dobbiamo utilizzare un algoritmo per e:

$$\int_a^b f(x)dx \approx \sum_{i=0}^{i=n-1} (x_{i+1}-x_i)\frac{1}{2}[f(x_{i+1})+f(x_i)]$$

Se dividiamo [a,b] in parti n parti uguali

$$x_{i+1}-x_i = \Delta x = \frac{b-a}{n}$$

Otteniamo

$$\int_a^b f(x)dx \approx \frac{1}{2}\cdot\frac{b-a}{n}\sum_{i=0}^{i=n-1}[f(x_{i+1})+f(x_i)]$$

$$\sum_{i=0}^{i=n-1}[f(x_{i+1})+f(x_i)] = \sum_{i=0}^{i=n-1} f(x_{i+1}) + \sum_{i=0}^{i=n-1} f(x_i) = \sum_{i=0}^{i=n-1} f(x_i) + f(b) + \sum_{i=0}^{i=n-1} f(x_i)$$

$$-f(a)+2\sum_{i=0}^{i=n-1} f(x_i) = -f(a)+2\left(\sum_{i=0}^{i=n} f(x_i) - f(b)\right) = -f(a)-2f(b)+2\left(\sum_{i=0}^{i=n} f(x_i)\right)$$

$$\int_a^b f(x)dx \approx \frac{(b-a)}{n}\left(\sum_{i=0}^{i=n} f(x_i) - \frac{1}{2}f(a) - f(b)\right)$$

- tri.m
- % Create a triangular pulse with width 0.4.
- % fs = 10000; % Sampling frequency (samples/sec)
- % t = -1:1/fs:1; % Time Vector
- % w = .4; % Triangle Width

```
fs = 10000;      % Sampling frequency (samples/sec)
t = -1:1/fs:1;   % Time Vector
w = .4;          % Triangle Width
x = tripuls(t-0.5,w);  % Sampled aperiodic triangle
plot(t,x);
xlabel('Time (sec)');ylabel('Amplitude')
```

- % See also SQUARE, S
- % RECTPULSE,