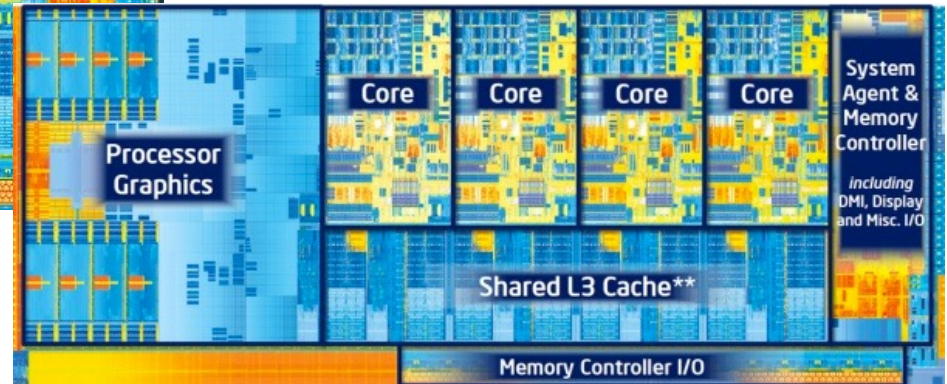
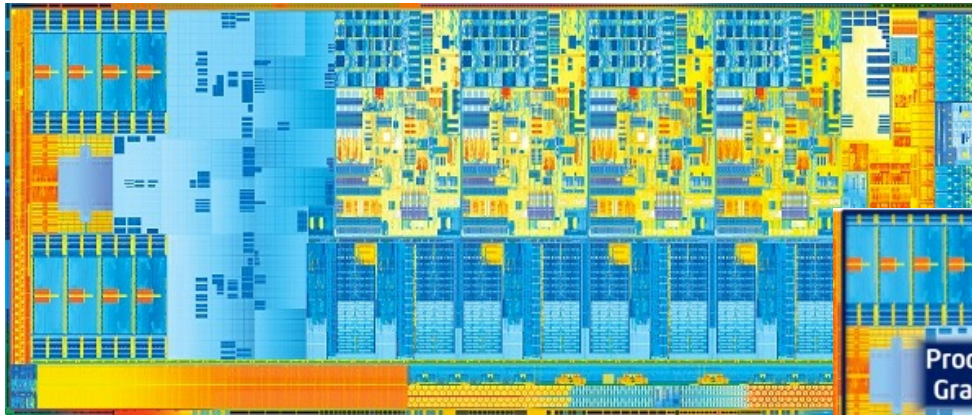


ARCHITETTURA DEGLI ELABORATORI (Gruppo B, matricole ultima cifra 5-9)



Canalizzazioni e docenti

- Gruppo A (matricola 0-4)
 - Docente: prof. Matteo Comin comin@dei.unipd.it
- Gruppo B (matricola 5-9)
 - Docente: prof. Francesco Silvestri silvestri@dei.unipd.it

Ricevimento studenti: su appuntamento da concordare via e-mail

Chi sono? ... Francesco Silvestri

- Formazione:
 - 2006-2009: PhD Università di Padova in Ingegneria Informatica
 - 2007: Visiting scholar University of Texas @ Austin
 - 2010-2016: Post-doc Università di Padova e IT University of Copenhagen
 - 2016-2019: Ricercatore Università di Padova
 - Dal 2019: Professore associato Università di Padova
- Didattica:
 - Architettura degli elaboratori (ing. inf.)
 - Big Data Computing (comp. eng.),
- Ricerca:
 - *Big data algorithms*: come estrarre informazioni da big-data?
 - *High performance algorithms*: come sfruttare le architetture di calcolo per i big-data?
- Real life:
 - 3 bambini + 1 moglie



Argomenti

- **Basi:**

- Rappresentazione dell'informazione
- Reti logiche

- **Architettura di un elaboratore:**

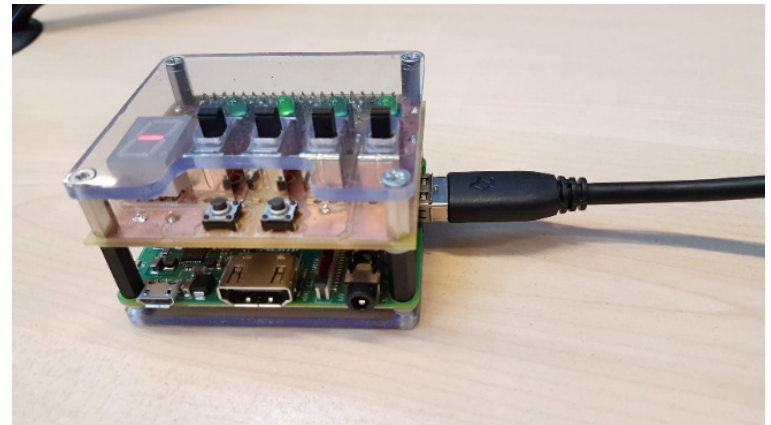
- Componenti: Processore, memoria, I/O
- Istruzioni macchina e linguaggio assembly

- **Organizzazione del processore:**

- Cache
- Memoria virtuale
- Gestione I/O
- Pipeline

- **Caso di studio:** processore ARM

- **Attività di laboratorio:** uso del linguaggio assembly su Raspberry PI



Moodle del corso

- <https://stem.elearning.unipd.it/course/view.php?TOADD>
- Sito del corso:
 - informazioni generali;
 - news dai docenti;
 - forum studenti/docenti;
 - calendario delle lezioni;
 - materiale didattico (slide delle lezioni, quiz, materiale aggiuntivo, esercizi e temi svolti);

Lezioni

- **Inizio:** 26 Febbraio 2024
- **Fine:** 14 Giugno 2024
- **Lezioni:**
 - Lunedì 12.30 → P1
 - Giovedì 12.30 → P3
 - Venerdì 12.30 → P3
- **Laboratorio:**
 - Giovedì 12.30 in aula Taliercio
 - Indicativamente inizio a fine marzo
 - Attività in gruppi di 2-3 persone

Esame

- **Tipologia:** prova teoria + prova programmazione
- **2 prove intermedie** (“compitini”)
 - I° compitino a metà corso;
 - II° compitino congiuntamente al I appello;
 - Voti sufficienti in **entrambe** le prove intermedie:
 - sostituiscono la parte di teoria
 - 1 punto bonus
 - Chi non supera entrambe le prove intermedie deve sostenere di nuovo **tutta la parte di teoria** (no voti parziali)
- **Prova programmazione:** solo durante gli appelli
 - I appello: II prova intermedia + prova programmazione
 - II, III, IV appello: prova teoria + prova programmazione

Prove di accertamento ed esami

- Prima prova intermedia (teoria, prima parte):
 - TBA (circa 15-19 Aprile)
- Seconda prova intermedia + prima prova programmazione:
 - I appello: 24 giugno 2024
- Teoria (tutto programma) + seconda prova programmazione:
 - II appello: 15 luglio 2024
 - III appello: 17 settembre 2024
 - IV appello TBA (gennaio/febbraio 2025).
- Tutti i voti conseguiti saranno validi fino a febbraio 2025.

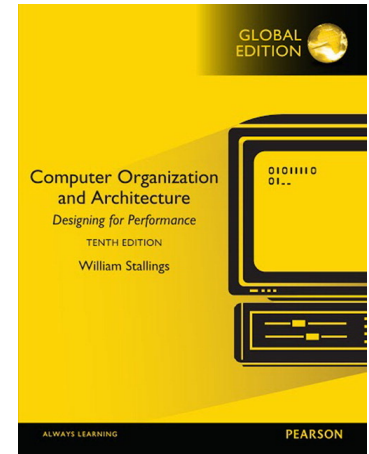
Studio

- Didattica: 6 ore a settimana, 72 ore totali
- CFU: 9
- Studio a casa: 153 ore
- Fortemente consigliato:
 - Passare la teoria con i **COMPITINI**
 - Rimanere al passo con lo studio
- Prima di iscriversi ai compitini:
 - Esercitarsi con i Quiz di preparazione all'esame su moodle

Testi

TESTO DI RIFERIMENTO

- William Stallings. **Computer organization and architecture designing for performance.** Pearson education, 10 or 11 edition.



PER CONSULTAZIONE

- L. Hennessy, D.A. Patterson. *Computer Architecture – A Quantitative Approach*. Morgan-Kaufmann, 2011.
- D.A. Patterson, J.L. Hennessy. *Computer Organization and Design ARM Edition: The Hardware Software Interface*. Morgan Kaufmann, 2016.
- R. E. Bryant, R. O'Hallaron. *Computer Systems: A Programmer's*. Pearson, 2015.

Perché studiare Architettura degli Elaboratori?

- Correttezza
- Prestazioni
- Efficienza energetica
- Sicurezza

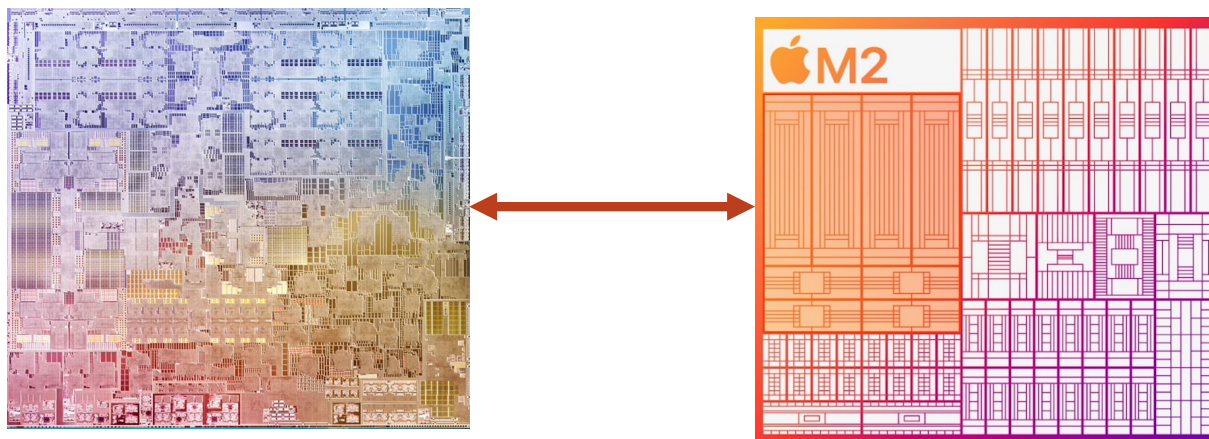
Gestire sistemi complessi

- Astrazione
- Disciplina
- Le tre “Y”

Da: S. L. Harris and D. M. Harris. Digital Design and Computer Architecture

Astrazione

- L'**astrazione** è il processo di identificazione delle caratteristiche principali di un sistema o oggetto, ignorando i dettagli non essenziali o irrilevanti.
 - L'astrazione è usata per scomporre sistemi complessi in parti più piccoli e facili da gestire
- L'astrazione è cruciale nell'architettura degli elaboratori perché permette la realizzazione di sistemi complessi che sono più facili capire e gestire.



Disciplina

- **Disciplina**: intenzionalmente ridurre le scelte implementative.
 - Aumenta la produttività nei livelli più alti di astrazione.

- Esempio: disciplina digitale
 - Tensioni discrete invece di continue
 - Circuiti digitali:
 - Sottoinsieme di circuiti analogici
 - Più semplici dei circuiti analogici
 - Maggior facilità nello sviluppo di sistemi complessi.
 - Tendenza nel sostituire sistemi analogici con sistemi digitali



Le tre Y

- **Hierarchy**

- Sistemi divisi in moduli e sotto-moduli
- Struttura divide-and-conquer

- **Modularity**

- Moduli con comportamenti ben definiti:
 - Il tipo di funzioni
 - Interfaccia con altri moduli

- **Regularity**

- Scomposizione gerarchica in moduli uniformi
- Minor moduli da progettare, più facilità nella sostituzione

Concetto di astrazione

Livelli di astrazione utilizzati per progettare e per descrivere un sistema di calcolo:

- L+5 Algoritmi
- L+4 Programmi
- L+3 Sistema Operativo
- L+2 Architettura
- L+1 Organizzazione (o micro architettura)
- L 0 Logica Circuitale
- L-1 Circuiti analogici
- L-2 Fisica

Stratificazione a livelli

- **L+5 Algoritmi / strutture dati:**
 - Procedure per risolvere un particolare problema computazionale. Focus su vincoli computazionali.
- **L+4 Programmi:**
 - Implementazione tramite opportuni linguaggi di opportuni algoritmi/strutture dati. Focus su struttura del codice, portabilità, librerie,...
- **L+3 Sistema Operativo:**
 - Sistema che permette la coesistenza di programmi indipendenti (processi) e l'uso condiviso delle risorse fisiche.
- **L+2 Architettura**
 - Definizione delle istruzioni base del computer e degli altri dispositivi fisici che compongono il sistema

Stratificazione a livelli

- **L+1 Organizzazione (o micro architettura)**
 - Funzionamento dei singoli componenti fisici del sistema in termini di interconnessione e spostamento di informazioni tra circuiti logici elementari
- **L 0 Logica Circuitale**
 - Realizzazione dei circuiti logici elementari basato sulla logica Booleana.
- **L-1 Circuiti analogici**
 - Progettazione dei circuiti logici elementari in termini di transistor o dispositivi optoelettronici.
- **L-2 Fisica**
 - Progettazione dei dispositivi elettronici integrati sulla base delle caratteristiche fisiche dei semiconduttori.

Stratificazione a livelli e corsi

- L+5 Algoritmi → **Dati e Algoritmi**
- L+4 Programmi → **Fondamenti di informatica/Dati e Algoritmi**
- L+3 Sistema Operativo → **Sistemi operativi**
- L+2 Architettura
- L+1 Organizzazione (o micro architettura)
- L 0 Logica Circuitale
- L-1 Circuiti analogici
- L-2 Fisica → **Corsi di elettronica**

Stratificazione a livelli e corsi

Livelli di astrazione utilizzati per progettare e per descrivere un sistema di calcolo:

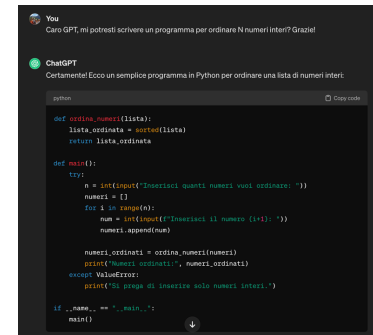
- L+5 Algoritmi
- L+4 Programmi
- L+3 Sistema Operativo
- L+2 Architettura
- L+1 Organizzazione (o micro architettura)
- L 0 Logica Circuitale
- L-1 Circuiti analogici
- L-2 Fisica



Astrazione nei linguaggi di programmazione

Troviamo astrazione anche nei linguaggi di programmazione. Ad esempio:

- LLM
- R
- Python
- Java
- C/C++/
- Assembly
- Linguaggio macchina
- The matrix



The screenshot shows a chat window with a user asking for a Python program to sort a list of integers. ChatGPT responds with a code snippet that defines a function to sort a list and a main function that prompts the user for input and prints the sorted list.

```
python
def ordina_numeri(lista):
    lista_ordinata = sorted(lista)
    return lista_ordinata

def main():
    try:
        n = int(input("Inserisci quanti numeri vuoi ordinare: "))
        numeri = []
        for i in range(n):
            num = int(input(f"Inserisci il numero {i+1}: "))
            numeri.append(num)

        numeri_ordinati = ordina_numeri(numeri)
        print("Numeri ordinati: ", numeri_ordinati)
    except ValueError:
        print("Va' forte di inserire solo numeri interi.")

if __name__ == '__main__':
    main()
```

