Studiare con l'IA, imparare con la testa

Metodo, memoria e responsabiiità dell'apprendimento

Materiale per workshop guidato

recupero Architettura degli Elaboratori

Nota Bene:

Il contenuto è stato valutato e approvato dal prof. Corrado Santoro

Se sei qui, significa che stai affrontando uno dei percorsi universitari più stimolanti di sempre: **l'Informatica**. In questo mondo, capire come imparare bene vale quanto conoscere gli algoritmi o programmare un compilatore. Questa dispensa nasce con un'idea semplice:

mostrarti come usare i Large Language Models (LLM) – come ChatGPT, Gemini, Claude –per **studiare meglio**, **capire più a fondo** e **verificare le tue conoscenze** in ogni insegnamento fondamentale del tuo corso di laurea. Non per copiare. Non per trovare scorciatoie. Ma per *diventare una versione più potente di te stessə*.

Perché usare i LLM quando studi informatica?

Per tre motivi molto forti:

1. Capisci più velocemente

Un LLM può darti:

- spiegazioni semplici dei concetti più astratti
- esempi concreti
- metafore
- confronti tra idee simili
- schemi visivi

Tutto questo accelera la comprensione, soprattutto nelle materie più "dure" come Architettura degli Elaboratori, Analisi o Algebra Lineare.

2. Puoi rielaborare in modi diversi

Gli LLM ti aiutano a:

- trasformare lezioni e appunti in mappe cognitive
- creare tabelle comparative

- costruire glossari dei termini più importanti
- generare domande guida per leggere i testi

È come avere un assistente che rende più chiaro ciò che già possiedi, senza sostituirti.

3. Ti alleni e ti valuti meglio

Qui gli LLM sono incredibili:

- possono generare esercizi graduati di difficoltà
- spiegarti passo passo come risolverli
- mostrarti errori comuni
- offrirti quiz per il ripasso
- aiutarti a fare “retrieval practice”, la tecnica più efficace per imparare davvero

Studiare informatica significa anche *saper affrontare problemi* — e gli LLM sono eccellenti sparring partner.

Cosa NON devono fare gli LLM

Questa dispensa ti aiuta anche a non cadere nelle trappole più comuni.

Gli LLM **non sono**:

- una fonte assoluta e infallibile
- un modo per evitare la fatica
- un sostituto dello studio personale
- un generatore di elaborati da “copia-incolla”

Sono strumenti potenti *se guidati con attenzione*, come un buon compilatore: se gli dai istruzioni sbagliate, il risultato sarà sbagliato.

Come usare questa dispensa

La dispensa è organizzata così:

- una breve introduzione al metodo **Smart Learning Design**, che ti mostra *come* strutturare lo studio usando gli LLM;
- una sezione con alcuni **insegnamenti** del CdL in Informatica, con esempi concreti di prompt, esercizi, schemi e strategie;
- una parte finale dedicata alla **metacognizione**: cosa hai imparato, cosa devi ancora rafforzare, come migliorare.
-

SEZIONE 1 — Come imparare con l'AI: il metodo Smart Learning Design (SLD)

Perché ti serve un metodo

Studiare informatica non è solo memorizzare definizioni o imparare il codice a memoria. Significa *capire*, *legare concetti*, *risolvere problemi*, *spiegare*, *applicare*. Per fare tutto questo in modo efficace ti serve una cosa che troppo spesso manca:
un metodo strutturato per imparare.
Il metodo **Smart Learning Design (SLD)** ti accompagna in 8 passi semplici, che puoi applicare a qualsiasi insegnamento — da Programmazione I ad Analisi Matematica II.
E la cosa bella?
Ogni passo può essere potenziato dai LLM, se impari a usarli bene (ed è proprio il senso di questa dispensa).

Le 8 fasi del metodo SLD

(per ogni fase, troverai: cosa significa + come usarla nel tuo studio + un esempio di prompt)

Cosa vuoi imparare?

La domanda più difficile di tutte. Ma se non chiarisci il tuo obiettivo, studierai in modalità *vagabonda*.

Qui userai il LLM per definire il tuo **perché**, cioè cosa vuoi sapere o saper fare.

Prompt esempio:

"Aiutami a definire un obiettivo SMART per lo studio di Architettura degli Elaboratori. Fammi una domanda alla volta."

Esplorare

È la fase del *primo contatto*: fai domande semplici, ingenue, "da curioso".

Qui lasci lavorare il tuo Sistema 1 (intuizione, curiosità) e usi il LLM per:

- capire i concetti base
- chiedere metafore
- cercare esempi
- avere confronti e analogie
- visualizzare schemi

Prompt esempio:

“Spiegami in modo semplice cosa sono gli autovalori. Fammi poi una metafora e un esempio pratico.”

Rielaborare

Ora devi *costruire senso*.

Non basta aver letto o ascoltato: devi trasformare il materiale in **strutture**.

Qui userai il LLM per:

- creare mappe cognitive
- sintetizzare appunti
- trasformare informazioni in tabelle
- costruire glossari
- generare domande guida

Prompt esempio:

“Organizza questi appunti in una tabella chiara sui tipi di strutture dati. Poi genera 10 domande guida.”

Applicare

È il momento di sporcarsi le mani. In informatica, applicare significa *provare*, *sbagliare*, *capire dove*, *correggere*.

Il LLM qui ti aiuta a:

- scomporre esercizi complessi in piccoli step
- mostrarti errori comuni
- verificare se un passaggio è corretto
- simulare ciò che faresti in laboratorio

-

Prompt esempio:

“Scomponi questo esercizio di Programmazione II in passaggi chiari. Non darmi la soluzione: fammi procedere uno step alla volta.”

Discutere

Imparare significa anche vedere **punti di vista diversi**.

Il LLM può:

- aiutarti a formulare tesi e antitesi
- mostrarti fallacie logiche
- farti ragionare su alternative

Prompt esempio:

“Quali sono i principali argomenti *pro* e *contro* le memorie dinamiche?”

Produrre

Qui dimostri davvero ciò che sai: scrivi, spieghi, progetti, implementi.

Il LLM ti aiuta a:

- strutturare un elaborato
- generare outline
- commentare codice
- migliorare chiarezza e precisione

Prompt esempio:

“Aiutami a progettare la struttura di una relazione sulla pipeline di un processore a 5 stadi.”

Consolidare

È il momento del ripasso intelligente. Non basta “rileggere”: devi recuperare attivamente.

Il LLM può:

- generare quiz personalizzati
- simulare un mini-esame
- creare flashcard

Prompt esempio:

“Fammi 10 quiz a risposta multipla sui protocolli di rete, dal più facile al più difficile.”

Cos’ho imparato? (Metacognizione)

L’informatica è piena di concetti che tornano, si intrecciano, si ricombinano.
Qui ti fermi e ti chiedi: “Cosa ho capito davvero? Cosa devo rinforzare?”
Il LLM può guidarti con domande di riflessione.

Prompt esempio:

“Fammi 5 domande per capire se ho davvero compreso la differenza tra paginazione e segmentazione.”

ARCHITETTURA DEGLI ELABORATORI

A. Come un LLM può essere di supporto in questo corso

Architettura può sembrare “astratta”, ma è in realtà molto visiva e strutturata. Un LLM può esserti utile per:

- capire architetture complesse attraverso **schemi e diagrammi** (pipeline, bus, memoria)
- ottenere **metafore e analogie** per chiarire concetti difficili (cache, ALU, branching)
- confrontare tecnologie (CISC vs RISC, pipeline vs superscalare)
- scomporre problemi di laboratorio
- spiegare passo passo il funzionamento di un ciclo istruzione

È come avere un assistente che ti restituisce il concetto con il filtro più adatto a te.

B. ESPLORARE – Capire i concetti fondamentali

Prompt semplici e diretti per chi vuole orientarsi:

Prompt 1 – “Spiegamelo semplice”

“Spiegami in modo molto chiaro cos’è la pipeline di un processore. Fai esempi e usa una metafora.”

Prompt 2 – “Fammi capire cosa è davvero importante”

“Quali sono i concetti fondamentali di Architettura degli Elaboratori che uno studente deve conoscere?”

Prompt 3 – “Confronti utili”

“Confronta memorie statiche e dinamiche in una tabella.”

Prompt 4 – “Visualizzazioni”

“Mostrami uno schema testuale del ciclo di funzionamento di una CPU.”

C. RIELABORARE – Trasformare quello che sai

Qui rendi più solido ciò che stai imparando.

Prompt 1 – “Fammi una mappa cognitiva”

“Genera una mappa testuale che organizzi: CPU, memoria, bus, I/O, pipeline e cache.”

Prompt 2 – “Glossario intelligente”

“Crea un glossario dei 15 termini più importanti di Architettura degli Elaboratori, con definizioni brevi.”

Prompt 3 – “Domande guida per studiare”

“Fammi 10 domande guida per capire meglio la gerarchia della memoria.”

D. APPLICARE – Esercizi step-by-step

Gli studenti spesso trovano difficile il passaggio dalla teoria alla pratica.

Prompt 1 – “Esercizio scomposto”

“Scomponi questo esercizio sulla pipeline in passaggi chiari. Non darmi la soluzione finale: Cosa accade quando un processore RISC esegue l’istruzione **STR R4,[R2,#12]**”

Prompt 2 – “Debug concettuale”

“Questi passaggi sono corretti per descrivere un ciclo di fetch-decode-execute? Indica eventuali errori.”

Prompt 3 – “Errori comuni”

“Quali sono gli errori più frequenti che gli studenti fanno quando studiano la gerarchia della memoria?”

E. RIPASSARE – Quiz, flashcard, mini-esami

Qui il LLM diventa un tutor per il recupero attivo.

Prompt 1 – Quiz progressivi

“Fammi 10 domande a risposta chiusa sulla memoria cache, da facile a difficile.”

Prompt 2 – Mini-esame orale

“Simula un orale di Architettura degli Elaboratori. Fai una domanda per volta.”

Prompt 3 – Flashcard

“Genera 20 flashcard sui concetti principali di CPU, bus e memoria.”

F. Mini-esercizi con difficoltà crescente –

LIVELLO 1 — BASE

Perfetti per iniziare, capire i concetti fondamentali e verificare la comprensione iniziale.

Esercizio 1 — Il ciclo di Von Neumann

Obiettivo: capire le fasi base del fetch-decode-execute.

Prompt da inserire nella dispensa:

“Spiegami il ciclo di Von Neumann come se lo dovessi raccontare a uno studente delle superiori. Poi fammi 3 domande per verificare se ho capito.”

Esercizio 2 — I componenti principali della CPU

Obiettivo: identificare ALU, CU e registri.

Prompt:

“Genera uno schema testuale che descrive ALU, Control Unit e registri. Poi chiedimi di completare una tabella con le loro funzioni.”

Esercizio 3 — Pipeline: capire cosa succede

Obiettivo: familiarizzare con i 5 stadi classici.

Prompt:

“Descrivi cosa succede all’istruzione `ADD R1, R2, R3` in ciascuno dei 5 stadi della pipeline. Usa un linguaggio semplice.”

LIVELLO 2 — INTERMEDIO

Perfetti per consolidare, applicare e collegare più concetti.

Esercizio 4 — Conflitti di pipeline (hazards)

Obiettivo: riconoscere data hazard, control hazard e structural hazard.

Prompt:

“Dammi un esempio concreto di ciascun tipo di hazard nella pipeline. Poi fammi risolvere 3 esercizi dove devo identificare l’hazard corretto.”

Esercizio 5 — Cache mapping

Obiettivo: capire direct mapped, fully associative e set associative.

Prompt:

“Confronta i tre tipi di mapping in una tabella. Poi dammi 5 indirizzi di memoria e chiedimi in quale blocco andrebbero in una cache direct mapped.”

LIVELLO 3 — AVANZATO

Perfetti per studenti che vogliono sfidarsi, prepararsi a esami o a un orale.

Esercizio 6 — Performance della pipeline

Obiettivo: calcolare speedup, ciclo di clock, throughput.
Prompt:
“Dammi un esercizio dove devo calcolare lo speedup ottenuto pipelining un processore non pipeline. Guidami con domande, non darmi mai la soluzione finché non rispondo.”

Esercizio 7 — Branch Prediction

Obiettivo: ragionare sul misprediction penalty.
Prompt:
“Fammi un problema avanzato sul branch prediction, includendo tassi di misprediction e penalty.”

Esercizio 8 — Trade-off progettuali

Obiettivo: ragionare in profondità su scelte architetturali.
Prompt:
“Simula un esercizio orale: ‘Se aumenti la dimensione della cache, quali sono i pro e contro dal punto di vista di latenza, costo e consumo energetico?’ Fammi prima proporre una risposta, poi correggila.”

Argomento 1 – Sistemi di numerazione

Contenuti teorici

- Sistema **decimale**, **binario**, **esadecimale**
- Rappresentazione posizionale dei numeri
- Conversioni:
  - **Binario Decimale**
  - **Binario Esadecimale**
  - **Decimale Binario**
- **Somma binaria** con e senza riporto

Esercizi proposti

Esercizio 1 – Conversione binario → decimale

Convertire i seguenti numeri binari in **decimale**:

1. 101011_2
2. 11010_2
3. 100111_2
4. 111001_2
5. 1000001_2

Suggerimento: usare i pesi delle potenze di 2.

Esercizio 2 – Conversione decimale → binario

Convertire i seguenti numeri **decimali** in binario:

1. 13_{10}
2. 27_{10}
3. 42_{10}
4. 58_{10}
5. 100_{10}

Suggerimento: applicare il metodo delle **divisioni successive per 2**.

Esercizio 3 – Conversione binario esadecimale

A) Binario → esadecimale

Convertire:

1. 10101110_2
2. 11110000_2
3. 11001001_2

Suggerimento: dividere il binario a gruppi di **4 bit**.

Esercizio 4 – Conversioni combinate

Convertire:

1. $5A_{16}$ in decimale
2. 1110101_2 in esadecimale
3. 94_{10} in binario
4. 11001100_2 in decimale
5. $2F_{16}$ in binario

Suggerimento: eseguire conversioni **intermedie** (ad esempio HEX → BIN → DEC).

Esercizio 5 – Somma binaria

Eseguire le seguenti **addizioni binarie**:

1. $1011_2 + 1101_2$
2. $10010_2 + 01101_2$
3. $11101_2 + 10111_2$
4. $110011_2 + 001101_2$
5. $101010_2 + 010111_2$

Suggerimento: ricordare le regole base:

A	B	Riporto	Somma
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Indicazioni metodologiche per gli studenti

Come studiare i sistemi di numerazione

1. **Comprendere il meccanismo posizionale**

- Ogni cifra pesa una potenza della base.
- In binario: potenze di **2**
- In esadecimale: potenze di **16**

2. **Memorizzare le corrispondenze fondamentali**

Esadecimale Binario:

HEX	BIN
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100

HEX	BIN
D	1101
E	1110
F	1111

Allenarsi molto sulle somme

- Spesso è la parte più “meccanica”, ma fondamentale per capire il funzionamento delle ALU.

Esercizi interattivi

Binary Practice

- https://www.binarypractice.com/
    - Ottimo per conversioni e somme.

RapidTables – Converter

- https://www.rapidtables.com/convert/number/
    - Per verificare risultati.

Argomento 2 – Rappresentazione dell’informazione

Contenuti trattati

- **Bit, byte, word, double word**
- **Interi senza segno su n-bit**
- **Rappresentazione in complemento a 2 (interi con segno)**
- **Rappresentazione dei caratteri – ASCII**

Esercizi proposti

Esercizio 1 – Bit, Byte, Word, Double Word

Rispondere alle seguenti domande:

1. Quanti bit ci sono in:
    - a) 3 byte
    - b) 5 word da 16 bit
    - c) 2 double word da 32 bit
2. Quanti byte servono per memorizzare:
    - a) una sequenza di 200 bit?
    - b) 10 numeri interi da 32 bit?
3. Un sistema a 64 bit utilizza word da 64 bit:
    - Quanti byte occupa una word?
    - Quante word posso memorizzare in 1 KB?

Obiettivo: comprendere le dimensioni delle unità base di informazione.

Esercizio 2 – Interi positivi senza segno su n-bit

1. Qual è il **massimo valore rappresentabile** usando:
    - a) 8 bit
    - b) 12 bit
    - c) 16 bit
2. Rappresentare su **8 bit senza segno** i numeri:
    - 25
    - 87
    - 145
3. Quale valore decimale rappresenta:
    - 10101101_2
    - 00011110_2

Esercizio 3 – Complemento a 2 (n-bit)

Parte A – Intervallo

1. Trovare l’intervallo di numeri rappresentabili su:

- a) 8 bit
- b) 12 bit

Parte B – Codifica

Rappresentare su **8 bit in complemento a 2**:

1. +45
2. −18
3. −63
4. +90

Procedura:

1. Scrivere il valore assoluto in binario
2. Invertire i bit
3. Aggiungere 1

Parte C – Decodifica

Convertire in **decimale**:

1. 11101011_2
2. 10011110_2
3. 01100101_2

Esercizio 4 – Codifica ASCII

1. Scrivere il codice ASCII (in **binario e esadecimale**) dei caratteri:
   - 'A'
   - 'a'
   - '0'
   - '9'
   - '?'
2. Decodificare i seguenti byte ASCII:
   - 01001000_2
   - 01000001_2
   - 00110010_2

3. Qual è la stringa ASCII associata alla sequenza:
4. 48 65 6C 6C 6F

   (in esadecimale)

Indicazioni metodologiche

Strategie di studio

1. **Prima capire i modelli**
   - senza segno → solo valori positivi
   - complemento a 2 → positivi **e negativi**
2. **Allenarsi molto con le conversioni**
3. **Fare sempre verifiche inverse**
   - Ricavare il decimale dal binario ottenuto per verificare la correttezza.

Contenuti per approfondimento

Materiale teorico

Sistemi di numerazione e rappresentazione

- https://it.wikipedia.org/wiki/Rappresentazione_dei_numeri_al_calcolatore

Complemento a due

- https://it.wikipedia.org/wiki/Complemento_a_due

Virgola fissa

- https://it.wikipedia.org/wiki/Virgola_fissa

Virgola mobile – IEEE 754

- https://it.wikipedia.org/wiki/IEEE_754

Codifica ASCII

- https://it.wikipedia.org/wiki/ASCII

Simulatori e strumenti

- **IEEE 754 converter**
  https://www.h-schmidt.net/FloatConverter/IEEE754.html
- **Binary & Complemento a 2 practice**
  https://www.binarypractice.com/
- **ASCII Table interattiva**
  https://www.asciitable.com/

Video YouTube

Cerca:

- *“complemento a due spiegazione”*
- *“virgola mobile IEEE 754”*
- *“floating point architecture explained”*
- *“ASCII encoding explained”*

Canali consigliati:

- Ingegneria Informatica
- UniBo LMS
- Computerphile (inglese)

Argomento 3 – Algebra booleana e funzioni logiche

Contenuti

- **Definizione di algebra booleana**
- Operazioni fondamentali:
  - **Somma logica (OR)**
  - **Prodotto logico (AND)**
  - **Inversione (NOT)**
- **Tabelle di verità**
- **Teoremi di De Morgan**
- **Funzioni logiche**
- **Analisi di funzioni logiche**
- **Sintesi di funzioni logiche**

- **Mappe di Karnaugh**

Esercizi Proposti

Esercizio 1 – Operazioni fondamentali e tabelle di verità

Parte A – Tavole base

Costruire le **tabelle di verità** delle seguenti operazioni:

1. A+B (OR)
2. $\underline{A} \times B$ (AND)
3. A (NOT)

Parte B – Espressioni logiche

Compilare la tabella di verità della funzione:

$F(A, B, C) = \overline{AB}+AC$

Obiettivo: comprendere il passaggio da espressione booleana a comportamento logico.

Esercizio 2 – Teoremi di De Morgan

1. Applicare i teoremi di De Morgan e semplificare:

a) $\overline{A+B}$
b) $\overline{AB}$
c) $\overline{\underline{A} + B + C}$
d) ABC

2. Trasformare le seguenti espressioni usando **solo porte NAND**:

a) F = A + B
b) F = AB

Richiamo teorico:

$\overline{A+B} = \overline{A} \cdot \times \overline{B}$

$\overline{AB} = \overline{A} + \overline{B}$

Esercizio 3 – Analisi di funzioni logiche

Data la funzione:

$F(A,B,C) = \overline{A}BC + A\overline{B} + AB$

Costruire la **tabella di verità completa**

Scrivere:

- Forma **lista di mintermini**
- Forma **Somma di Prodotti**

Obiettivo: passare dall'espressione al comportamento funzionale.

Esercizio 4 – Sintesi di funzioni

Data la seguente tabella di verità:

A	**B**	**C**	**F**
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1

A	B	C	F
1	1	0	0
1	1	1	1

Scrivere:

- Mintermini per cui F=1
- Espressione **Somma di Prodotti**

Semplificare l’espressione ottenuta

Obiettivo: passare da tabella a circuito ottimizzato.

Esercizio 5 – Mappe di Karnaugh

Data la tabella della verità dell’esercizio precedente, disegnare la **mappa di Karnaugh a 3 variabili.**

Individuare:

- Raggruppamenti validi
- Termini semplificati

Scrivere la **forma logica minima**

Obiettivo: ridurre porte e complessità circuitale.

Indicazioni metodologiche

Linee guida

Procedere sempre così:

Espressione → Tabella di verità → Karnaugh → Espressione minima

Saper distinguere:

- **Analisi**
    - Ho l’equazione → trovo cosa fa

- **Sintesi**
  - Ho il comportamento → costruisco l’equazione

Suggerimenti operativi Karnaugh

- Raggruppare celle adiacenti in potenze di 2:
  - 1, 2, 4, 8...
- Preferire gruppi **più grandi possibile**
- I gruppi **possono sovrapporsi**
- I bordi della mappa **sono contigui**

Risorse web di approfondimento

Teoria

- **Algebra booleana**
  - https://it.wikipedia.org/wiki/Algebra_booleana
- **De Morgan**
  - https://it.wikipedia.org/wiki/Leggi_di_De_Morgan
- **Mappe di Karnaugh**
  - https://it.wikipedia.org/wiki/Mappa_di_Karnaugh

Simulazioni ed esercizi

- **Logisim Evolution(simulatore circuiti)**
  - https://github.com/logisim-evolution/logisim-evolution
- **BoolSimplifier**
  - https://www.boolean-algebra.com/
- **Mapa di Karnaugh interattiva**
  - https://www.dcode.fr/karnaugh-map

Video didattici

Cerca su YouTube:

- *“algebra booleana esercizi svolti”*
- *“mappe di Karnaugh spiegazione”*
- *“De Morgan NAND gates”*

Canali consigliati:

- Ingegneria Informatica Uni
- UniBo LMS
- Computerphile