

Architettura Degli Elaboratori

Indice:

Lezione 1.....	1
26 settembre 2025.....	1
Lezione 2.....	2
3 ottobre 2025.....	2
Lezione 3	4
10 ottobre 2025.....	4
Lezione 4.....	6
17 ottobre 2025.....	6
Lezione 5.....	7
24 ottobre 2025.....	7
Lezione 6.....	8
31 ottobre 2025.....	8

Lezione 1

26 settembre 2025

Il linguaggio dei computer è formato solo da 0 e 1

Non capiscono i linguaggi di programmazione

I bit sono i valori 0 e 1

Se ho 8 bit viene chiamato byte

I computer utilizzano il sistema binario perché hanno solo due stati: acceso (1) e spento (0)

Il numero più a sinistra è il bit più significativo (MSD) mentre il numero più a destra c'è il bit meno significativo (LSD)

Quando la base non è ovvia bisogna scriverla

Se devo convertire un numero dopo la virgola da decimale a binario devo dare una precisione quindi dico quante cifre voglio prendere in considerazione dopo la virgola nel risultato finale, moltiplico il numero con la virgola per 2 e se il numero prima della virgola è 1 lo metto da parte e azzerò il numero prima della virgola e lo moltiplico per 2

Esempio:

$$0,87 \times 2 = 1,74 \text{ m1} = 1$$

$$0,74 \times 2 = 1,48 \text{ m2} = 1$$

$$0,48 \times 2 = 0,96 \text{ m3} = 0$$

$$0,96 \times 2 = 1,92 \text{ m4} = 1$$

Se ho un numero intero separato dalla virgola utilizzo l'algoritmo del resto e della moltiplicazione

I bit sono le cifre del numero binario

I numeri ottali vanno da 00 a 07

I numeri esadecimali vanno da 0 a 9 e appena finisce il 9 si inizia da A e si finisce con F

Per trasformare da binario ad ottale si divide il numero binario in gruppi da 3 bit a partire dalla virgola e ogni cifra ottale viene codificata dai 3 bit

Esempio (vedi slide prof pagina 20)

Per trasformare da binario ad esadecimale si divide il numero binario in gruppi da 4 bit a partire dalla virgola e ogni cifra esadecimale viene codificata dai 4 bit

Esempio (vedi slide prof pagina 21)

Per trasformare in ottale/esadecimale a binario scompongo il numero in binario

Esempio (vedi slide prof pagina 22)

Il numero più grande che posso rappresentare in binario devono essere tutti 1

L'overflow si verifica quando la somma non è rappresentabile quindi se noi abbiamo 4 bit la somma ci crea 5 bit

Se il sottraendo è maggiore del minuendo si ha una differenza negativa

Per capire quanti bit servono per rappresentare qualcosa devo fare il logaritmo di quel qualcosa mentre se devo capire quanti numeri possono essere rappresentati da quel tot di bit devo fare la potenza di $2^n - 1$

Esempio:

- Se voglio rappresentare 10 numeri diversi (da 0 a 9):
 $n = \log_2(10) \approx 3,32 \rightarrow$ servono 4 bit, perché
con 3 bit rappresento solo 8 numeri (da 0 a 7),
mentre con 4 bit ne rappresento 16 (da 0 a 15).
- Se ho 5 bit:
 $N = 2^5 = 32 \rightarrow$ posso rappresentare 32 numeri,
cioè da 0 a 31.

Lezione 2

3 ottobre 2025

Nella rappresentazione a complemento a 2 utilizzo il bit più significativo per vedere che modulo è

Per trasformare da decimale a binario i numeri negativi faccio $2^n - \text{il numero negativo}$

Esempio:

$$-3 = 2^n - 3$$

Per comodità utilizzo il complemento a 2 quindi cambio il segno quindi prendo il numero positivo e lo rappresento in bit e poi fino al primo 1 lascio tutto uguale per poi invertire i bit quindi 0 diventa 1 e viceversa

Per semplificare i calcoli se ho una sequenza di 1 in complemento a 2 incomincio a calcolare dall'1 più a destra

Esempio:

1111 0000 0000 0000 0000 0000 0001

$$(-2^{28} + 2^0)$$

1111 1111 1111 1111 1111 1111 1110 1001

$$(-2^5 + 2^3 + 2^0)$$

Nel complemento a 2 la somma e la sottrazione sono identiche a quelle con i numeri senza segno

L'underflow si verifica quando il numero negativo è più piccolo di quello che possiamo rappresentare

Alcune persone e libri utilizzano il termine overflow anche se si tratta di underflow

Quando abbiamo la somma fra un numero positivo e negativo non avremmo mai un overflow perché la somma di un positivo e un negativo rimane sempre nell'intervallo

Quando c'è un riporto finale nel complemento a 2 viene ignorato

Se faccio la somma tra due negativi e due positivi può dare overflow

Capisco se c'è overflow dal segno della soluzione. Se sto sommando due positivi e il risultato da un numero negativo allora ho un overflow. Stessa cosa con i negativi quindi se sommo due numeri negativi mi danno un numero positivo allora ho un overflow

Esempio: (vedere le slide della prof)

Nel complemento a 2 se i bit più significati dei due addendi sono uguali a quello della somma allora non ho un overflow

Se io ho un numero binario con la virgola mobile posso scriverlo mettendo la virgola dove voglio e poi moltiplico 2^n dove n è la posizione della virgola e se n è negativo sposto la virgola a destra mentre se ho n positivo la virgola si sposta a sinistra

Esempio:

$$-10110,101 = -1,0110101 \times 2^4$$

$$0,001101 = 1,101 \times 2^{-3}$$

Con i numeri reali non si può utilizzare il complemento a 2

La virgola mobile si può rappresentare con la formula: (vedi slide prof)

Ci sono due tipi di precisione per rappresentare i numeri reali dentro alla CPU:

- Singola precisione (float) 32 bit
- Doppia precisione (double) 64 bit

Nella singola precisione abbiamo i primi 23 bit che servono a rappresentare il numero dopo la virgola poi abbiamo 8 bit che servono a rappresentare l'esponente e l'ultimo bit serve a rappresentare il segno

Mentre nella doppia precisione abbiamo i primi 20 bit che rappresentano il numero dopo la virgola mentre 11 bit servono per rappresentare l'esponente e l'ultimo bit serve per rappresentare il segno (vedere slide prof)

Esempio:

Numero = -37,75

Binario = 100101, 11

Forma normalizzata = -1, 0010111 x 2⁵

Mantissa = 0010111

Esponente = 101 (5)

Segno = 1

Lo standard IEEE 754 viene utilizzato per rappresentare dei numeri con la virgola mobile

Con lo standard IEEE 754 con la precisione singola (32 bit) si utilizza la codifica polarizzata a 127 in modo tale da avere sempre un numero positivo quindi sommo al mio esponente 127

Mentre con la precisione doppia (64 bit) si utilizza la codifica polarizzata a 1023 quindi sommo al mio esponente 1023

Esempio (vedi slide prof)

Le potenze da applicare per trasformare il numero dopo la virgola da binario a decimale devo utilizzare le potenze negative quindi 2⁻ⁿ

Lezione 3

10 ottobre 2025

Il calcolatore ha la CPU (central processing unit), è formato da 2 parti, va a parlare con la memoria e ha un ingresso e un'uscita

Il calcolatore ha anche una memoria e contiene due tipi di dati che sono:

- Programmi che verranno eseguiti sulla CPU;
- Dati che servono a far eseguire il programma;

La CPU è formata da:

- Control Unit (Unità di Controllo): è la parte che coordina e controlla tutte le operazioni del processore. Decide cosa deve fare la CPU in ogni momento, interpreta le istruzioni del programma e ordina ai vari componenti (come l'ALU e i registri) di eseguire le operazioni necessarie.
- Datapath: è la parte dove vengono eseguite le operazioni. Comprende l'ALU (Arithmetic Logic Unit), i registri, e tutti i circuiti necessari per eseguire i calcoli e trasferire i dati durante l'esecuzione del programma.

Gli algoritmi servono a descrivere passo per passo come risolvere quel problema

L'algoritmo può essere anche scritto in linguaggio umano non per forza in linguaggio macchina

Per far capire l'algoritmo all'esecutore bisogna scriverlo in un linguaggio di programmazione

L'algoritmo e il linguaggio di programmazione sono ad alto livello

Il sistema operativo è un programma diverso dalle normali applicazioni, perché funziona come controllore del sistema: gestisce le risorse del computer (CPU, memoria, dispositivi di input/output) e decide quali risorse assegnare ai vari programmi in esecuzione.

Nella parte più bassa del calcolatore troviamo i transistor, che sono i componenti fondamentali dell'hardware: servono per realizzare le porte logiche, cioè i circuiti base che permettono di eseguire operazioni logiche e aritmetiche.

Per progettare un hardware bisogna seguire alcuni passaggi:

1. Definire cosa deve fare il circuito (la sua funzione).
2. Descrivere il comportamento con l'algebra booleana, che permette di esprimere le operazioni logiche.
3. Costruire il circuito logico corrispondente, cercando di ottimizzarlo (ridurre la complessità e il numero di porte).
4. Effettuare la mappatura tecnologica, cioè tradurre l'algebra booleana in componenti fisici (porte logiche, transistor, ecc.).
5. Verificare il funzionamento del circuito, per assicurarsi che l'hardware esegua correttamente le operazioni previste.

Le porte logiche sono strumenti matematici e circuitali che servono a descrivere e comprendere il funzionamento dei circuiti digitali.

Nella logica binaria ha solo 2 valori: 0 e 1, abbiamo degli operatori logici che sono AND, OR e il NOT

La logica binaria è molto simile all'aritmetica binaria infatti l'AND è simile al prodotto mentre l'OR è simile alla somma per questo motivo con l'AND si utilizza il simbolo della moltiplicazione mentre per l'OR si utilizza il simbolo dell'addizione

Il NOT è la negazione quindi se io ho $X = 0$ diventa $X = 1$ e viceversa

L'AND funziona così:

$X \mid Y$	$Z = XY$
0 0	0

0	1	0
1	0	0
1	1	1

L'OR funziona così:

X		Y		Z = X+Y
0		0		0
0		1		1
1		0		1
1		1		1

Le porte logiche hanno i seguenti simboli (vedere slide prof)

Lo XOR funziona così:

X		Y		Z = X+Y
0		0		0
0		1		1
1		0		1
1		1		0

Utilizziamo le identità per trovare il circuito logico migliore quindi molto più veloce a fare i calcoli e meno porte logiche

Per misurare il costo di un circuito si utilizzano due costi:

- Numero di termini (porte AND)
- Numero di letterali

Il letterale è una variabile o il negato di una variabile

Esempio di letterale: $F = XYZ$, i letterali sono X, Y, Z

Esempio di termini: $F = XYZ + XZ$, i termini sono XYZ e XZ

Lezione 4

17 ottobre 2025

Una funzione booleana può essere rappresentata tramite una tabella di verità, nella quale ogni combinazione possibile di input corrisponde a un unico valore di output (vero o falso, cioè 1 o 0).

Per rappresentare in modo matematico le funzioni booleane si usano delle strutture standard chiamate forme canoniche, che permettono di scrivere la funzione in maniera ordinata e univoca.

Le principali forme canoniche sono:

- Somma di prodotti (forma canonica disgiuntiva)
- Prodotto di somme (forma canonica congiuntiva)

Sono state scelte perché è molto più facile semplificare i circuiti e producono dei circuiti molto piccoli

Esistono due tipi di forme canoniche e sono:

- SOP (Sum Of Products) = somma di prodotti di letterali
- POS (Product Of Sums) = prodotto di somme di letterali

Mintermini si riferisce alle somme di prodotti mentre mastermini sono più utili per i prodotti di somme

Un mintermine è un prodotto (AND) di tutte le n variabili della funzione, dove ogni variabile appare come letterale (cioè nella forma diretta o negata).

Un maxtermine è una somma (OR) di tutte le n variabili della funzione, dove ogni variabile appare come letterale (diretta o negata).

Nella SOP (Sum of Products) le variabili che valgono 0 vengono negate, mentre quelle che valgono 1 non si negano.

Nella POS (Product of Sums) le variabili che valgono 0 non si negano, mentre quelle che valgono 1 vengono negate.

Per ottenere la forma POS, si prendono dalla tabella di verità tutte le combinazioni in cui $f = 0$.

Per ottenere la forma SOP, si prendono dalla tabella di verità (o dalla K-map) tutte le combinazioni in cui $f = 1$.

Con 3 variabili, bisogna raggruppare gli 1 adiacenti che hanno in comune le stesse variabili (vedi esercizio).

Con 4 variabili, si raggruppano gli 1 in blocchi di dimensione pari a potenze di 2 (1, 2, 4, 8, ...), cercando di lasciare meno 1 isolati possibile.

Si può usare l'effetto "Pac-Man", cioè considerare gli estremi della mappa come adiacenti: si può "uscire" da un lato e "rientrare" dall'altro per formare gruppi più grandi.

Lezione 5

24 ottobre 2025

Un implicante è un gruppo di celle con valore 1 in una K-map (mappa di Karnaugh) che sono adiacenti tra loro. Quando raggruppi questi 1 adiacenti, ottieni un implicante che semplifica l'espressione booleana

Un implicante primo è un implicante che non può essere ulteriormente semplificato. In altre parole, un implicante primo è il gruppo più grande possibile di celle con valore 1 che non può essere esteso senza perdere la sua capacità di coprire i mintermini della funzione booleana.

Un implicante primo essenziale è un implicante primo che copre almeno una cella che non può essere coperta da nessun altro implicante. In altre parole, un implicante primo essenziale è un implicante che contiene un mintermine unico (cioè una cella 1 che non può essere coperta da altri gruppi).

Gli implicanti con don't care (DC) sono le celle X che trovi nelle K-map. Quando hai tre 1 adiacenti e una X adiacente, puoi trasformare quella X in 1 per formare un gruppo di 4 celle, ma non puoi mai fare un gruppo composto solo da X. Le X (don't care) ti permettono di semplificare i gruppi, ma un implicante deve sempre coprire almeno una cella con 1. Le celle X possono essere trattate come 1 per estendere i gruppi, ma non possono mai essere l'unico elemento di un implicante.

Lezione 6

31 ottobre 2025

Il decoder è un circuito combinatorio che effettua una decodifica, cioè trasforma un codice binario compatto in una forma meno compatta (più estesa).

In pratica, a partire da un numero binario in ingresso, attiva una sola uscita corrispondente a quel valore.

Esempio:

Se l'ingresso è 011 (che corrisponde al numero 3 in decimale), il decoder attiva solo l'uscita numero 3.

In sintesi, parte da un codice compatto e lo espande.

L'encoder è il circuito inverso del decoder. Serve a rappresentare un'informazione in modo più compatto, cioè trasforma più linee di ingresso in un numero binario più corto.

In sintesi, parte da più segnali e li codifica in un codice binario compatto.

Il multiplexer è un circuito combinatorio che sceglie quale ingresso far passare in uscita in base a un segnale di selezione (S).

È paragonabile a un if/else nel linguaggio di programmazione.

Esempio:

- Se $S = 0$, in uscita passa l'ingresso I_0 .
- Se $S = 1$, in uscita passa l'ingresso I_1 .

In generale, un multiplexer con n bit di selezione può scegliere tra 2^n ingressi.

Half adder è il circuito combinatorio che segue la somma di due bit

Nel sommatore logico (half adder o full adder), abbiamo due uscite: Sum e Carry (vedere slide prof)

- Sum rappresenta il risultato della somma dei bit di ingresso, ed è calcolato tramite l'operatore XOR:
→ $\text{Sum} = A \oplus B$
(vale 1 solo se uno dei due ingressi è 1, ma non entrambi)
- Carry rappresenta il riporto della somma (cioè il "resto" che va alla posizione successiva) ed è calcolato con un AND:

→ $\text{Carry} = A \cdot B$
(vale 1 solo se entrambi gli ingressi sono 1)

Se abbiamo il riporto abbiamo bisogno di sommare 3 bit quindi utilizziamo il circuito combinatorio chiamato full adder

Il circuito full adder si può rappresentare come un half adder in cascata

I circuiti sequenziali sono circuiti logici in cui una parte dell'uscita ritorna come ingresso. Questo permette al circuito di ricordare informazioni sullo stato precedente, ed è per questo che si dice che hanno memoria.

Il valore di questa memoria è detto stato del circuito.

I circuiti sequenziali si dividono in due categorie principali:

- Sincroni:
 - o Evolvono nel tempo discreto, scandito da un segnale di clock comune.
 - o Il clock coordina tutte le operazioni, rendendo il comportamento prevedibile e ordinato.
- Asincroni:
 - o Non utilizzano un clock; la loro evoluzione dipende dagli ingressi e avviene in tempo continuo.
 - o Sono più veloci ma anche molto più complessi da progettare, perché più sensibili ai ritardi dei segnali.

Il latch è un blocco di memoria elementare usato per memorizzare un singolo bit. In un latch è possibile sia scrivere che leggere il valore memorizzato.

Il latch Set-Reset (SR latch) è un tipo base di latch, costruito con due porte NAND (o NOR) collegate in modo incrociato (vedere slide del prof)
Serve per impostare (Set) o azzerare (Reset) il bit memorizzato.

La combinazione vietata è $S = 1$ e $R = 1$, perché porta entrambe le uscite (Q e \bar{Q}) a 0, violando la regola secondo cui \bar{Q} deve essere sempre il contrario di Q .

In pratica, quando $S = 1$ e $R = 1$, il circuito entra in una condizione instabile e non è possibile sapere quale valore manterrà dopo.

Il latch D (detto anche data o delay latch) è una versione semplificata del latch SR e serve per memorizzare un singolo bit (0 o 1) (vedere slide del prof)
Ha un solo ingresso chiamato D e un segnale di controllo chiamato Enable (o Clock).
Il latch D risolve il problema delle combinazioni non valide del latch SR.

Quando Enable = 1, il latch legge il valore presente su D e lo porta in uscita (Q) → l'uscita segue l'ingresso.

Quando Enable = 0, il latch mantiene il valore precedente in memoria.

