Appunti di Fondamenti di Informatica

Giacomo Simonetto

Primo semetre 2023-24

Sommario

Appunti del corso di Fondamenti di Informatica della facoltà di Ingegneria Informatica dell'Università di Padova.

Indice

1 Storia dell'informatica **3**

2 Computer **4**

3 Modello di Von Neumann **4**
3.1 Central Processing Unit o CPU 4
3.2 Memoria primaria e secondaria 5
3.3 Dispositivi di I/O 6

4 Sistemi operativi **7**
4.1 Unix - Linux 8

5 Rappresentazione delle informazioni nei calcolatori **9**
5.1 Rappresentazione in sistema posizionale 9
5.2 Rappresentazione in modulo - segno 10
5.3 Rappresentazione in complemento a 2 11
5.4 Rappresentazione in virgola fissa 12
5.5 Rappresentazione in virgola mobile 13
5.6 Base esadecimale e ottale 13
5.7 ASCII e UNICODE 13

6 Introduzione alla programmazione **14**
6.1 Algoritmo 14
6.2 Computational Thinking 14
6.3 Programmazione 14
6.4 Linguaggi di programmazione 14

7 Java **15**
7.1 Introduzione 15
7.2 Struttura di un programma 15
7.3 Variabili 15
7.4 Metodi 15
7.5 Classi e oggetti 15
7.6 Classe String 15
7.7 Classe Scanner 15
7.8 Selezioni 15
7.9 Iterazioni 16

8 Strutture dati **16**
8.1 Array 16
8.2 Matrici 16

1 Storia dell'informatica

fine 1800	Si hanno i primi tentativi di **ricerca di un linguaggio formale**. La matematica è un sistema formale completo? Esiste un procedimento meccanico (passo-passo, finito) per dimostrare se una proposizione sia vera o falsa? Il primo tentavo di *"formalizzazione della matematica"* viene svolto da David Hilbert, con cui si scopre che la matematica possiede 23 problemi di formalizzazione chiamati *"23 problemi di Hilbert"*. La risposta alla prima domanda risale al 1931 quando Goedel, con il *"teorema di incompletezza"* conferma che la matematica non è un sistema formale.
1936	Church, Turing e Kleene elaborano dei formalismi meccanici tra cui la **Macchina di Turing** e la **Tesi di Church-Turing** che sostiene che tutto ciò che è computabile è computabile dalla macchina di Turing universale. La capacità computazionale tra una macchina di Turing e un computer odierno è la stessa (ecceto per il fatto che la macchina di Turing prevedeva uno spazio di archiviazione illimitato), cambia solo la velocità computazionale. Entrambe le macchine risolvono gli stessi problemi, ovvero tutti quelli che si possono risolvere con un algoritmo.
1943	Si arriva a costruire l'**ENIAC**, il primo computer (general purpose) della storia. Si programmava esclusivamente in binario, i circuiti si basavano sulle valvole termoioniche e occupava un palazzo di 5 piani.
1948	Walter Brattain, John Bardeen e William Shockley creano il primo **transistor** (MOSFET) in grado di sommare due bit. Grazie a ciò, durante gli anni '50 si riesce a ridurre le dimensioni dei computer a un piano.
1958	John Backus della IBM sviluppa il primo linguaggio di programmazione di alto livello **Fortran** per programmare uno dei computer sviluppati dall'IBM. Le novità erano quelle di poter programmare in un linguaggio simile all'inglese e l'introduzione delle selezioni e dei cicli.
1966	Viene formulato il **Teorema di Jacopini-Bohm**: qualsiasi algoritmo è implementabile utilizzando le strutture fondamentali di sequenza, selezione e ripetizione. In alrre parole ha senso investire nell'informatica come strumento per risolvere problemi di tipo algoritmico.
1969	Viene inventato l'**Internet** (a carattere).
1970-71	Niklaus Wirth inventa il **PASCAL**, il primo linguaggio strutturato in cui scompaiono il go-to, ma a differenza dei precedenti, non può essere usato per scrivere sistemi operativi.
1970-71	Federico Faggin sviluppa il **primo microprocessore**.
1973	Dennis Ritchie inventa il linguaggio **C**, simile al Pascal, ma con la possibilità di impiegarlo per sviluppare sistemi operativi.
1977	Steve Jobs e la Apple inventano il **primo personal computer**.
1979	Bjarne Stroustrup sviluppa il **C++**, ovvero il C con il paradigma a oggetti.
1979	Come risposta alla Apple, la IBM crea il suo primo PC. Non credendo nei PC, non volendo perdere tempo e non avendo un proprio sistema operativo, la IBM si rivolge alla Microsoft (nata nel 1974) chiedendole di sviluppare un sistema operativo per microprocessori. La Microsoft sviluppa **MS-DOS** (Microsoft Disk Operating System) chiedendo 50 euro per copia (praticamente nulla). Dopo 6 anni vengono vendute 300 milioni di copie e il ricavato viene investito per sviluppare Windows.
1991	Nasce **internet a interfaccia grafica** ed insieme ad esso c'è la necessità di avere programmi in grado di girare indipendentemente dal sistema operativo (Win, Mac OS, Unix). Si sviluppano le prime **Virtual Machine** in grado di e codici in grado di girare su qualsiasi macchina, grazie alle Virtual Machine.
1994	Linus Torvald pubblica la prima versione stabile del kernel **Linux** (creato nel 1991).
1995	James Gosling nella Sun Microsystems sviluppa il linguaggio **Java**, dotato della particolarità di generare un codice compilato in grado di essere eseguito su qualsiasi macchina grazie alla Java Virtual Machine.

2 Computer

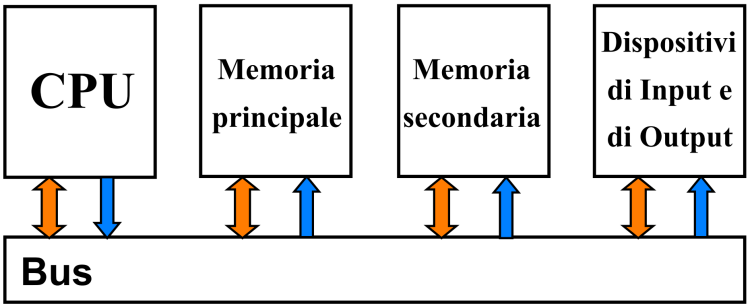
Per computer, o calcolatore, si intende un sistema di elaborazione e memorizzazione di informazioni che opera sotto il controllo di un programma. È composto da hardware (parte fisica) e software (programmi e dati). I dati possono essre di diverso tipo (immagini, testi, audio, video, ...) e sono rappresentati elettricamente in 0 e 1.

Esistono diversi tipi di computer (workstation, smartphone, ...) che possono svolgere diversi tipi di impieghi (elettrodomestici, giochi, fotografie, ...).

3 Modello di Von Neumann

Il modello di Von Neumann è una rappresentazione dell'archietttura di un elaboratore. Prevede la presenza di 4 blocchi: la CPU, la memoria primaria, la memoria secondaria e i dispositivi di I/O collegati insieme grazie al BUS.

Inoltre sono presenti due diversi flussi di informazioni: quello di dati è bidirezionale (nello schema è rappresetato dalle frecce arancioni), mentre quello degli indirizzi e dei segnali di controllo è unidirezionale con direzione CPU → altri blocchi (nello schema è rappresentato dalle frecce blu).



3.1 Central Processing Unit o CPU

La Central Processing Unit ha il compito di:

- individuare ed eseguire le istruzioni
- elaborare dati attraverso la ALU (Unità Logico Aritmetica)
- reperire dati di input e restituire dati di output

Componenti

È costituita da tre blocchi:

Control Unit	o *CU*, gestisce l'esecuzione dei programmi e i flussi di dati
ALU	o *Arithmetic Logical Unit*, elabora le espressioni logiche e algebriche
Registri	memorie temporanee per dati che devono essere subito elaborati: - l'**Accumulator**, o *ACC*, che memorizza i dati elaborati o che stanno per essere elaborati dalla *ALU* - il **Program Counter**, o *PC*, che memorizza l'indirizzo di memoria dell'istruzione successiva da eseguire - l'**Instruction Register**, o *IR*, che memorizza l'istruzione da decodificare - il **Memory Data Register**, o *MDR*, che memorizza i dati/le istruzioni lette o che stanno per essere scritte nella memoria primaria - il **Memory Adress Register**, o *MAR*, che memorizza l'indirizzo di memoria dell'istruzione da eseguire o del dato da utilizzare

Funzionamento

La CPU ha funzionamento ciclico che si divide in tre fasi. La velocità di una CPU, chiamata frequenza di clock è espressa in cicli al secondo (dell'ordine dei GHz) ed è scandita dal *Clock*. La velocità massima è dovuta ai limiti fisici della tecnologia disponibile.

1° fase	**fetch**	- viene letto l'indirizzo dell'istruzione da eseguire dal PC e viene salvato nel MAR - viene incrementato il PC in modo che punti all'istruzione successiva - viene letta e caricata l'istruzione prima nel MDR poi nell'IR
2° fase	**decode**	- la CU decodifica l'istruzione salvata nell'IR - se necessario viene caricato nel MAR l'indirizzo del dato da elaborare o della posizione in cui scrivere il dato elaborato
3° fase	**execute**	- viene eseguita l'istruzione: - se necessario viene caricato nel MDR il dato referenziato dal MAR - il dato può essere salvato nell'ACC o impiegato in un'operazione logico-algebrica eseguita dalla ALU - il risultato viene salvato nell'ACC - oppure il dato memorizzato nell'ACC viene scritto nell'indirizzo di memoria contenuto nel MAR

Limiti e parallelismo

I limiti della CPU sono principalmente due: la frequenza di clock e l'impossibilità di eseguire un'istruzione, finché non viene completata quella precedente. Per superare il secondo problema si sono cercate soluzioni come il parallelismo. Esistono due tipi di parallelismo:

- **parallelismo a livello di istruzioni**:
  detto anche pipeline o multiscalari, consiste nel suddividere il ciclo di un processore in 5 stadi (lettura, decodifica, recupero operandi, caricamento, esecuzione, invio risultati) e di eseguire contemporaneamente più istruzioni su stadi diversi. In questo modo non è necessario aspettare che l'esecuzione dell'istruzione precedente termini per iniziare quella della successiva, ma è sufficiente che sia completato il primo stadio.
- **parallelismo a livello di processori**:
  consiste nell'avere più processori che lavorano contemporaneamente in grado di eseguire più istruzioni nello stesso momento. In base all'archiettura di distinguono in multiprocessori (se sono presenti più processori che condividono la stessa memoria) o multicomputer (se sono più processori, ciascuno con la propria memoria dedicata, collegati tra loro).

3.2 Memoria primaria e secondaria

La memoria ha il compito di memorizzare dati e programmi, sia in maniera temporanea, che permanente.

Struttura

La memoria è composta da celle chiamate allocazioni di memoria. Ogni allocazione può contenere un preciso numero di bit. Un bit (abbreviazione di Binary Digit) è l'unità minima di dimensione della memoria e corrisponde allo spazio occupato da 0 o 1. Il bit è un sottomultiplo del byte, 1byte = 8bit. Il byte è l'unità minima di accesso singolo alla memoria ed è l'unità base per la misura della dimensione dello spazio di archiviazione.

Memoria primaria

La memoria primaria è la più veloce delle due, ma anche la più costosa. Ne esistono due tipi:

RAM	o *Random Access Memory*, memoria volatile, dotata della caratteristica di avere un tempo di accesso ad una cella indipendente dal luogo in cui essa si trova (tempo di accesso "casuale"). Viene impiegata per salvare dati temporanei derivati dall'esecuzione di programmi.
ROM	o *Read Only Memory*, memoria permanente di sola lettura in cui vengono salvati i programmi necessari all'avvio della macchina, es. BIOS (*Basic Input Output System*)
Cache	o *memoria di località*, memoria estremamente veloce che permette di memorizzare celle di memoria che potenzialmente potrebbero tornare utili nelle future elaborazioni. Esistono due tipi di località: - località temporale: accedere alla stessa cella in tempi vicini - località spaziale: accedere a celle limitrofe

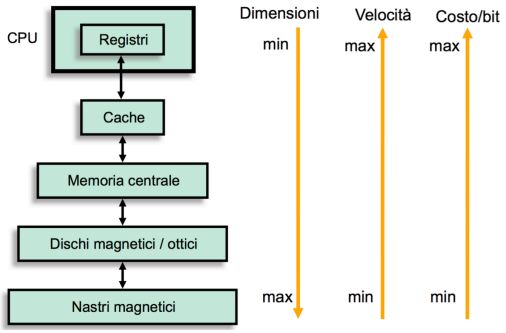
Memoria secondaria

Memoria non volatile, più lenta e molto meno costosa della memoria primaria. È riservata all'archiviazione di file, dati, programmi (tra cui anche il sistema operativo) che vengono trasferiti nella *RAM* al momento dell'esecuzione. Esistono diversi supporti di archiviazione di memoria secondaria:

- *HDD* o disco magnetico
- *SSD* o disco a stato solido (solid state drive)
- dischi ottici come *CD*, *DVD*, *Blue-Ray*
- chiavette USB
- nastri magnetici, impiegati per l'archiviazione di documenti, sono molto lenti, ma hanno costo molto basso ed elevata capacità di archiviazione

Gerarchie di memoria

Maggiore è la dimensione, minore è la velocità ed il costo.



3.3 Dispositivi di I/O

Permettono l'interazione dell'essere umano con la macchina. Comprendono mouse, tastiera, touchpad, schermo, stampante, ... Le operazioni relative ai dispositivi i I/O sono:

polling	o controllo da programma, consiste nel ripetuto e periodico controllo dello stato dei dispositivi
interrupt	richiama l'attenzione della CPU attraverso un'interruzione del flusso di esecuzione
DMA	o *Direct Memory Access*, dispositivo indipendente dalla CPU che gestisce il flusso di dati dei dispositivi di I/O ed alleggerisce il carico della CPU (la CPU indica solo indirizzi e dati da spostare)

4 Sistemi operativi

Cos'è e a cosa serve

Un sistema operativo, abbreviato *SO*, è un insieme di software che fornisce all'utente una serie di comandi e servizi per usufruire della potenza di calcolo di un elaboratore elettronico, inoltre garantisce l'operatività di base di un elaboratore, coordinando e gestendo le risorse hardware di elaborazione e memorizzazione, le periferiche, le risorse/attività software e facendo da interfaccia con l'utente, senza il quale quindi non sarebbe possibile l'utilizzo del computer stesso e dei programmi. Ogni sistema operativo è legato ad uno specifico hardware.

Bootstrap

Il bootstrap è la fase in cui viene avviato il sistema operativo, generalmente all'avvio del computer. La procedura di bootstrap è memorizzata nella *ROM*, all'avvio del computer:

- la CPU legge le istruzioni dalla ROM
- recupera il sistema operativo dal disco (memoria secondaria)
- carica il sistema operativo nella RAM
- avvia l'esecuzione dei programmi che ne permettono il funzionamento

Struttura a cipolla

Il sistema operativo è organizzato su più strati (come una cipolla), ciascuno con la caratteristica di poter interfacciarsi soltanto con quelli più interni. Questo garantisce modularità, flessibilità e più facile manutenzione. Gli strati, dal più interno sono:

nucleo o core	gestisce le risorse fisiche, comunica con l'hardware ed è scritto in linguaggio macchina
gestore I/O	gestisce i dispositivi di input e output e si occupa di trasferire i dati tra le diverse memorie del computer
gestore memoria	gestisce l'allocazione delle memorie durante l'esecuzione dei programmi
gestore archiviazione	anche chiamato filesystem, organizza la struttura si archiviazione dei file
interfaccia utente	permette all'utente di interagire con la macchina attraverso un'interfaccia grafica (GUI) o a linea di comando (CLI)

Comandi e linguaggi di controllo

Ogni sistema operativo possiede un linguaggio di controllo, ovvero un insieme di comandi che permette di interfacciarsi con il sistema operativo, eseguire operazioni o programmi, controllare le attività in corso e lo stato della macchina. I comandi sono impartiti dall'utente attraverso il terminale o attraverso l'interazione con l'interfaccia grafica. In Windows i comandi riprendono il vecchio sistema MS-DOS.

4.1 Unix - Linux

Introduzione

Linux è un sistema operativo sviluppato nel 1994 da Linus Torvald, basandosi su UNIX. Unix è un sistema operativo proprietaro, Linux è la corrispettiva versione di Unix, ma open source.

Utenti e permessi

Da sempre Linux e Unix sono sistemi multiutente, ovvero ciascun file ha un utente proprietario e ogni utente più accedere e modificare solo dove è permesso. L'utente che non ha limitazioni è chiamato `root`.

Filesystem

Il filesystem è organizzato con una struttura ad albero capovolto, in cui la cartella, o *directory*, principale, che contiente tutti i file e le directory del sistema, è chiamata `root`. Ogni elemento nel filesystem è raggiungibile attraverso un percoso chiamato *path*. Il percorso della cartella `root` è `\`.

I path si distinguono in:

- **percorso assoluto**:
  ovvero il percorso che separa la cartella `root` dal file in questione, inizia con `\`, ovvero il simbolo della cartella `root`, cioè con `/`, es. `/user/nomeutente/home/desktop/file.txt`
- **persorso relativo**:
  ovvero il percorso che separa una cartella diversa dalla `root` dal file in questione, inizia con il nome della cartella di partenza, es. `desktop/file.txt` rispetto alla `home`

Il percorso per accedere alla stessa cartella è `./`, quello per accedere alla cartella di livello superiore è `../`

Shell o CLI

La Shell è l'interfaccia utente a linea di comando. In Linux/Unix sono presenti diverse shell: *bash*, *csh*, *ksb*, *zsh*, in base alla distribuzione utilizzata (in Windows è quella di *MS-DOS*).

I comandi della CLI si dividono in *builtin*, che sono presenti di default nell'OS, ed *esterni* che possono essere installati in un secondo momento dall'utente.

In Linux/Unix sono presenti dei metacaratteri come il simbolo `*`, che rappresenta una sequenza di uno o più caratteri, e il simbolo `?`, che rappresenta un singolo carattere.

Alcuni comandi di Linux/Unix sono:

`cd`	change directory
`ls`	list files and subdirectory of the current directory
`cp`	copy file
`mv`	move or rename file
`rm`	remove file
`mkdir`	make new directory
`rmdir`	remove directory
`kill`	end process
`sudo`	per eseguire comandi dall'utente root
`man`	manuale
`appropos`	ricerca comandi
`whatis`	descrizione comando

5 Rappresentazione delle informazioni nei calcolatori

In un calcolatore elettonico, le informazioni sono memorizzate ed elaborate in formato binario (0 e 1). Bisogna convertire ciascun tipo di dato (testo, numeri, immagini, audio, video) in notazione binaria. Di seguito alcuni esempi:

numeri naturali $\mathbb{N}$	rappresentazione secondo il sistema posizionale
numeri interi $\mathbb{Z}$	rappresentazione modulo - segno rappresentazione in complemento a due
numeri reali $\mathbb{R}$	rappresentazione in virgola fissa rappresentazione in virgola mobile (singola e doppia precisione)
caratteri	tabella ASCII o Unicode

Dato che un computer può elaborare soltanto un numero finito di informazioni, mentre i numeri sono infiniti, significa che ci sarà un valore massimo e un valore minimo rappresentabile. Inoltre viene introdotto un errore dato dal fatto che non tutti i nmeri hanno un numero di cifre limitate (es. π o $\sqrt{2}$).

5.1 Rappresentazione in sistema posizionale

Sfrutta il principio che ogni cifra possiede un peso dato dalla posizione relativa nel numero. Nel sistema decimale ogni cifra ha, come peso, una potenza di 10, in quello binario si usano le potenze di 2.

Rappresentazione

DEC	234_{10}	$= 2 \cdot 10^2 + 3 \cdot 10^1 + 4 \cdot 10^0$
BIN	11101010_2	$= 1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0$

base b num con n cifre $\quad a_{n-1}a_{n-2}\dots a_0 = \sum_{k=0}^{n-1} a_k \cdot b^k \quad$ con $\begin{matrix} a_k = k\text{-esima cifra} \\ b^k = \text{peso della } k\text{-esima cifra} \end{matrix}$

Valori rappresentabili

Il range di valori rappresentabili è $[0, 2^n - 1]$.

Conversione decimale - binario

Per eseguire la conversione dal sistema decimale a quello binario, bisogna usare l'algoritmo di conversione:

```
while (numero != 0)
resto[i] = numero % base
numero = numero / base
```

Il numero converito si ottiene giustapponendo i resti ottenuti al contrario, in modo che l'ultimo resto è la cifra più significativa e il primo resto è quella meno significativa.

Conversione binario - decimale

Per convertire un numero dal sistema binario a quello decimale, basta associare ciascuna cifra al suo peso in potenza di 2 e sommare i valori ottenuti.

$$\begin{aligned} 11101010_2 &= 1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 \\ &= 1 \cdot 128 + 1 \cdot 64 + 1 \cdot 32 + 0 \cdot 16 + 1 \cdot 8 + 0 \cdot 4 + 1 \cdot 2 + 0 \cdot 1 \\ &= 128 + 64 + 32 + 8 + 2 \\ &= 234_{10} \end{aligned}$$

5.2 Rappresentazione in modulo - segno

Con il sistema posizionale non è possibile rappresentare valori negativi, per cui viene introdotta la rappresentazione modulo-segno. Tale sistema prevede di riservare il primo bit al segno del numero ed i restanti per il modulo rappresentato con il sistema decimale.

Rappresentazione

num ≥ 0	$+12_{10} = +1 \cdot 1100_2 = 0 + 1100_2 = 01100_{2MS}$
num < 0	$-12_{10} = -1 \cdot 1100_2 = 1 + 1100_2 = 11100_{2MS}$

Valori rappresentabili

Il range di valori rappresentabili è $\left[-\left(2^{n-1}-1\right);+\left(2^{n-1}-1\right)\right]$. Si osserva che lo 0_{10} possiede due rappresentazioni: 10000_{2MS} e 00000_{2MS} per $n = 4$.

Conversioni

Per le conversioni da sistema decimale a binario e viceversa si ricorre al procedimento illustrato nel sistema porizionale, con l'unica particolarità di avere uno `0` se il numero è positivo ed un `1` se il numero è negativo.

Criticità

Questo sistema di rappresentazione non viene utilizzato in quanto l'algoritmo per eseguire somme (e sottrazioni) è poco efficiente e complesso.

5.3 Rappresentazione in complemento a 2

Con la rappresentazione in complemento a 2 è possibile rappresentare numeri interi positivi e negativi, eliminando la doppia rappresentazione dello zero e semplificando l'algoritmo di somma (e differenza), avendo sempre i positivi e lo 0 che inziano per `0` e i negativi che iniziano per `1`.

Rappresentazione

num ≥ 0	$+12_{10} = 0 + 1100_2 = 01100_{C2}$
num < 0	$-12_{10} = 10100_{C2} \quad \rightarrow \quad -12 + 32 = 20_{10} = 10100_2$

Valori rappresentabili

Il range di valori rappresentabili è $\left[-2^{n-1}; 2^{n-1} - 1\right]$.

Conversione decimale - binario

Per i numeri positivi, compreso lo 0, si impiega il classico sistema posizionale (aggiungendo uno `0` davanti al numero per il segno), mentre per i numeri negativi è necessario:

1. sommare 2^n con n numero di cifre in binario in modo da rendere il numero positivo
2. convertire il risultato secondo il sistema posizionale (se i conti sono giusti il numero inizierà per `1`)

$$
\begin{aligned}
&+12_{10} = 0 + 1100_2 = 01100_{C2} \\
&-12_{10} \rightarrow -12 + 2^5 = 20_{10} = 10100_2 \\
&-12_{10} = 11100_{C2}
\end{aligned}
$$

In alternativa se il numero è negativo

1. convertire il modulo secondo il sistema posizionale
2. aggiungerci uno `0` davanti
3. invertire le cifre (gli `0` diventano `1` e gli `1` diventano `0`)
4. sommarci 1 (se i conti sono giusti il numero inizierà per `1`)

$$
\begin{aligned}
&-12_{10} \rightarrow 1100_2 \rightarrow 00011 \rightarrow 10011 \rightarrow 10011 + 1 \rightarrow 10100 \\
&-12_{10} = 10100_{C2}
\end{aligned}
$$

Conversione binario - decimale

Per i numeri che iniziano per `0` basta eseguire la conversione per sistema posizionale, per quelli che inizano con `1` bisogna converire il numero secondo il sistema posizionale e toglierci 2^n con $n =$ cifre del numero.

$$
\begin{aligned}
01100_{C2} &= 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^3 + 0 \cdot 2^1 + 0 \cdot 2^0 \\
&= 8 + 4 \\
&= 12_{10}
\end{aligned}
$$

$$
\begin{aligned}
10100_{C2} &= 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^3 + 0 \cdot 2^1 + 0 \cdot 2^0 \quad - 2^5 \\
&= 16 + 4 \quad - 32 \\
&= 28 - 32 \\
&= -12_{10}
\end{aligned}
$$

5.4 Rappresentazione in virgola fissa

La rappresentazione in virgola fissa riprende il principio del sistema posizionale, usando potenze con esponenti negativi per le cifre dopo la virgola.

Rappresentazione

DEC	$234,56_{10}$	$= 2 \cdot 10^2 + 3 \cdot 10^1 + 4 \cdot 10^0 + 5 \cdot 10^-1 + 6 \cdot 10^-2$
BIN	$11101010,1001_2$	$= 1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 +$ $+1 \cdot 2^-1 + 0 \cdot 2^-2 + 0 \cdot 2^-3 + 1 \cdot 2^-3$

base b num con n, m cifre

$$a_n a_{n-1} \dots a_0, a_1 a_2 \dots a_m = \sum_{k=0}^{n} a_k \cdot b^k + \sum_{k=1}^{m} a_k \cdot b^{-k}$$

con $a_k = k$-esima cifra, b^k, b^{-k} =peso della k-esima cifra

Valori rappresentabili

Il range di valori rappresentabili è $0 + [2^{-m}, 2^n - 1]$, con n cifre intere, m cifre decimali.

Conversione decimale - binario

Per eseguire la conversione dal sistema decimale a quello binario, è necessario dividere la parte intera da quella dopo la virgola. Per la prima basta convertirla con l'algoritmo visto per il sistema posizionale, mentre per la parte decimale è necessario applicare il seguente "algoritmo":

```
while (numero != 0)
    parteIntera[i] = parteIntera di numero
    numero = numero * base
```

Per ottenere il numero convertito è necessario prendere le parti intere in ordine (senza invertirli). Si osserva che il risultato potrebbe essere un numero illimitato.

Conversione binario - decimale

Per convertire un numero dal sistema binario a quello decimale, basta associare ciascuna cifra al suo peso in potenza di 2 e sommare i valori ottenuti.

$$\begin{aligned}
11101010,1001_2 &= 1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 + \\
&\quad + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 0 \cdot 2^{-3} + 1 \cdot 2^{-4} \\
&= 1 \cdot 128 + 1 \cdot 64 + 1 \cdot 32 + 0 \cdot 16 + 1 \cdot 8 + 0 \cdot 4 + 1 \cdot 2 + 0 \cdot 1 + \\
&\quad + 1 \cdot \frac{1}{2} + 0 \cdot \frac{1}{4} + 0 \cdot \frac{1}{8} + 1 \cdot \frac{1}{16} \\
&= 128 + 64 + 32 + 8 + 2 + 0,5 + 0,0625 \\
&= 234,5625_{10} \\
&\approx 234,56_{10}
\end{aligned}$$

Criticità

Non tutti i numeri sono rappresentabili in un numero finito di cifre. Alcuni numeri che nel sistema decimale hanno un numero finito di cifre, nel sistema binario potrebbero essere illimitati, per cui la loro rappresentazione potrebbe essere un'approssimazione. Inoltre è poco efficiente in quando per rappresentare numeri molto grandi, la parte decimale sarebbe poco significativa e i bit riservati a tale parte si potrebbero usare per la parte intera. Viceversa per numeri prossimi allo 0.

5.5 Rappresentazione in virgola mobile

Utilizza la notazione esponenziale con mantissa ed esponente che permette maggiore flessibilità per numeri molto grandi e numeri prossimi allo 0. Lo standard *IEEE 754* prevede due rappresentazioni: a singola e a doppia precisione.

Rappresentazione

DEC	$234,56_{10}$	$= 0,23456 \cdot 10^4$
BIN	$11101010,1001_2$	$= 0,111010101001 \cdot 2^8$

Suddivisione in bit

Per lo standard a singola precisione, la suddivione dei bit in segno mantissa esponente 1 + 23 + 8.
Per lo standard a doppia precisione, la suddivione dei bit in segno mantissa esponente 1 + 52 + 11.

Valori rappresentabili

Il range di valori rappresentabili, sia a singola che a doppia precisione, sono molto ampi ($\pm 2^{127}$ a singola), la precisione che si valuta è la distanza tra due numeri binari con la virgola adiacenti.

Per la singola precisione, avere 23 bit di mantissa, significa che posso suddividere l'intervallo da $0,0 \ldots 1 \cdot 2^n, 0,1 \ldots 1 \cdot 2^n$ in 2^{23} parti. Queste parti saranno èiù fitte per esponenti bassi e meno fitte per esponenti alti.

Per lo standard a singola precisione si ha che la differenza tra due numeri adiacenti $\delta = 2^{23} \cdot 2^E$, dove E è l'esponente binario dei numeri da rappresentare che varia da $[-127, +128]$.

Per lo standard a singola precisione si ha che la differenza tra due numeri adiacenti $\delta = 2^{52} \cdot 2^E$, dove E è l'esponente binario dei numeri da rappresentare che varia da $[-1023, +1024]$.

Valori limite

Alcune combinazioni di mantissa ed esponente sono catalogate per valori particolari come:

- 0: esponente minimo, mantissa 0
- ∞: esponente massimo, mantissa 0
- NaN: esponente massimo, mantissa 1 (*Not A Number*)

Criticità

Non tutti i numeri sono rappresentabili con un numero finito di cifre, ma vengono introdotte approssimazioni, come nel caso di 4,35 (che ha una rappresentazione binaria illimitata). Per questo motivo, quando si fanno confronti tra valori in virgola mobile è necessario introdurre un errore entro cui due numeri sono uguali.

Inoltre nel caso in cui si lavora con numeri molto grandi a cui vengono sommati numeri molto piccoli, si rischia di non avere abbastanza precisione per eseguire correttamente la somma, rischiando di perdere il valore dell'addendo minore. Questo avviene soprattutto quando l'addendo più piccolo è minore della precisione del numero più grande.

5.6 Base esadecimale e ottale

5.7 ASCII e UNICODE

6 Introduzione alla programmazione

6.1 Algoritmo

Un algoritmo è un metodo di risoluzione di un problema che:

- deve essere eseguibile
- non deve essere ambiguo
- deve concludersi in un numero finito di passi

6.2 Computational Thinking

Per computational thinking, o pensiero computazionale, si intende l'insieme delle abilità che permettono di astrarre il problema e tradurlo in algoritmo. Comprende le tecniche di astrazione/risoluzione di problemi algoritmici tra cui la decomposizione di problemi complessi e la modularità.

6.3 Programmazione

definizione, algortmo, errori, ...

6.4 Linguaggi di programmazione

Linguaggio macchina

Linguaggi assembly e di basso livello

Linguaggi di alto livello

7 Java

7.1 Introduzione

portabilità, bytecode, memoria, JVM ...

7.2 Struttura di un programma

classi, metodi, oggetti, librerie, pacchetti, commenti, variabili, literals, stringhe, costanti, strutture logiche, ...

7.3 Variabili

Tipi di dato

char, byte, short, int, long, float, double

Promozione e casting

promozione, casting esplicito, ...

Operazioni

assegnazione, somma, moltiplicazione, differenza, divisione, modulo, confronti, precedenze

Costanti

final

Literals

numeri, stringhe, interpretazione

7.4 Metodi

main, parametri, overloading

7.5 Classi e oggetti

Struttura

variabili d'istanza, metodi, riferimenti e allocazione, this metodi costruttori, di accesso, di modifica, ...

Incapsulamento

public, private, static, non-static incapsulamento e accessibilità

Pacchetti e organizzazione

pacchetti, classi e import, documentazione API

7.6 Classe String

utilizzo, metodi, ...

7.7 Classe Scanner

utilizzo, metodi, ...

7.8 Selezioni

utilizzo, struttura, else if, annidamento, else sospeso, espressioni logiche, ... switch case

7.9 Iterazioni

while, do-while, for, cicli annidati, break, continue

8 Strutture dati

adt, liste e array

8.1 Array

definione, struttura, creazione, passaggio come parametro

Algoritmo di copia

copia di un array

Algoritmo di ridimensionamento

ridimensionamento di un array

Array riempiti a metà

dimensione logica vs dimensione relative

8.2 Matrici

array bidimensionali