Introduzione alla programmazione (bozza)

1. Sistemi a microprocessore

Un computer è un dispositivo le cui parti scambiano ed elaborano flussi di cifre binarie (bit) a gruppi di otto (1 byte = 8 bit). Questi bit possono rappresentare diverse informazioni:

- numeri
- caratteri
- suoni
- immagini in movimento
- immagini fisse
- **istruzioni**

⇒ … 0 1 1 0 1 0 0 1 1 1 0 1 0 0 1 1 …

Prendiamo ad es. una calcolatrice: alcuni tasti generano sequenze di bit che rappresentano numeri, altri generano codici interni che rappresentano istruzioni.

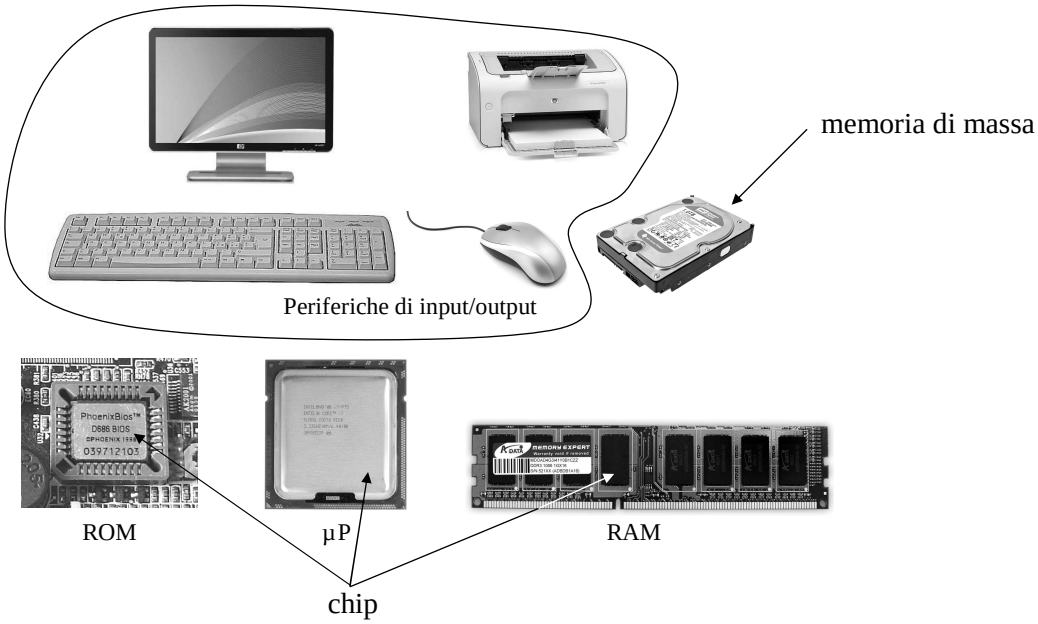
La parte fisica di un computer, ovvero i vari oggetti che lo compongono, viene denominata **hardware** (significato originario: “ferramenta”).

L’insieme delle istruzioni che, con o senza l’intervento dell’utilizzatore, determinano il funzionamento della macchina viene indicato con il termine **software**.

Un computer è basato su un sistema a microprocessore, che a sua volta include i seguenti dispositivi:

- **Microprocessore** (o **CPU**): insieme dei circuiti che leggono ed eseguono le istruzioni del software, eseguendo calcoli o più in generale operazioni logiche.
- **Memoria ROM**. Memoria permanente a sola lettura. Nei sistemi a uso specifico (elettrodomestici, stampanti, lettori audio/video, ecc.), ospita le istruzioni per svolgere i compiti prefissati; nei computer ospita le istruzioni per avviare il sistema e riconoscere le periferiche di base (**BIOS**, *Basic Input/Output System*). In entrambi i casi si parla di **firmware**, da *firm*, azienda o ditta produttrice.

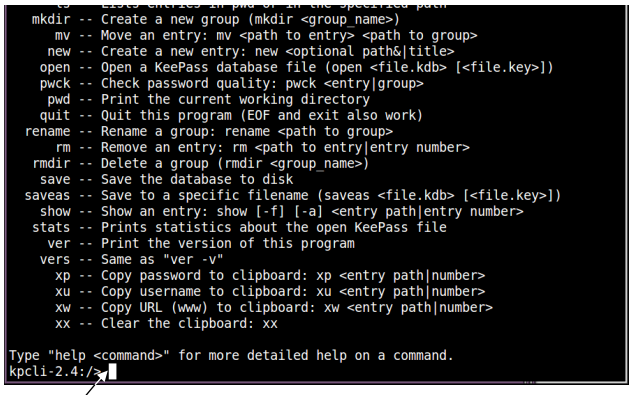
- **Memoria RAM**. Memoria volatile a lettura/scrittura. Ha la funzione di memorizzare temporaneamente, e cioè solo mentre il sistema è in funzione, le informazioni che devono essere più velocemente disponibili per svolgere i compiti previsti.
- **Periferiche di input/output**. Sono i dispositivi che consentono di interagire o scambiare informazioni con il microprocessore. Esempi di periferiche di input: pulsantiera, tastiera, mouse, scanner, webcam; esempi di periferiche di output: spia, display, sistema video, stampante. Alcune periferiche hanno funzioni sia di input, sia di output: sistema audio, scheda di rete.
- **Memoria di massa**. Memoria di grande dimensioni (hard-disk, pendrive, disco DVD) che ha il compito di ospitare il software le informazioni memorizzate dagli utenti.



Il microprocessore, la ROM e la RAM sono basati su circuiti elettrici estremamente miniaturizzati denominati **microchip**, o più semplicemente **chip**. Gli attuali microprocessori contengono alcuni miliardi di transistor, cosa che si traduce in circa un miliardo di porte logiche.

Le memorie di massa ospitano il **sistema operativo** (Windows, OS X, Android, iOS, etc.), ovvero il sistema di istruzioni che costituisce la base per utilizzare i software specifici, denominati **programmi applicativi** (Word, Excel, Chrome, Paint, Photoshop, etc.). Le principali funzioni dell'OS (*Operative System*) sono:

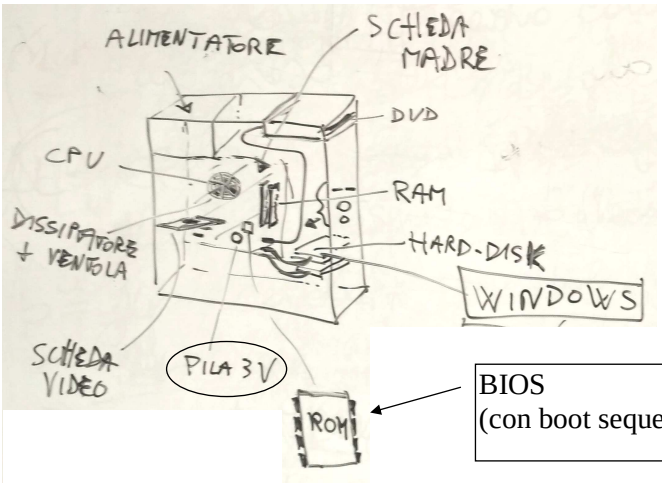
- Fornire all'utente una interfaccia per interagire con la macchina. Questa interfaccia può essere di tipo **CLI** (Command Line Interface) o **GUI** (Graphic User Interface, con icone, menù etc.)
- Fornire un sistema per archiviare e gestire i file.
- Installare i driver di periferica (software che pilota il funzionamento di un certo dispositivo) in modo che tutti gli applicativi possano farne uso (ad. es. mouse, tastiera, stampante etc.).
- Fornire un certo numero di servizi che consentano un ordinato funzionamento degli applicativi.



prompt dei comandi CLI GUI

2. Avvio di un computer

La fase di avvio di un computer viene comunemente denominata **boot** (abbreviazione di *bootstrap*). Alcune impostazioni del BIOS possono essere modificate dall'utente e mantenute nel tempo grazie a una pila a bottone interna che alimenta anche l'orologio del computer. Una delle impostazioni è la **sequenza di boot** (o *boot sequence*), un elenco che determina l'ordine in cui saranno esaminate le memorie di massa alla ricerca del sistema operativo.



BIOS
(con boot sequence)

Main	Advanced	Security	Power	Boot	Exit

Boot-time Diagnostic Screen:	[Disabled]	Item Specific Help
QuickBoot Mode:	[Enabled]	Displays the diagnostic screen during boot
Restore On AC/Power Loss:	[Stay Off]	
On LAN:	[Stay Off]	
First Boot Device	[Removable Devices]	sequenza di boot (modificabile)
Second Boot Device	[Hard Drive]	
Third Boot Device	[ATAPI CD-ROM]	
Fourth Boot Device	[Network Boot]	
▶Hard Drive		all'interno di queste voci è possibile modificare l'ordine di ricerca di hard-disk, dispositivi removibili, etc., nel caso ne fossero presenti due o più dello stesso tipo
▶Removable Devices		
▶Removable Format		

L'avvio di un computer avviene nelle seguenti fasi:

1. Lettura ed esecuzione del programma BIOS contenuto nella ROM, ovvero:
   - caricamento di driver universali per garantire il funzionamento di una base hardware costituita almeno da tastiera, USB, video e memorie di massa;
   - fase POST (*Power On Self Test*) con eventuali messaggi di errore sonori o a video;
   - Scansione delle memorie di massa secondo la sequenza di boot alla ricerca di un sistema avviabile.
2. Caricamento del sistema operativo.
3. Login dell'utente.
4. Uso dei programmi applicativi.

3. Linguaggi di alto livello

Le istruzioni che possono essere accettate da un microprocessore sono scritte in forma di numeri binari. Pertanto un programma è rappresentato da una sequenza di byte che, espressa in esadecimale, assume una forma del tipo

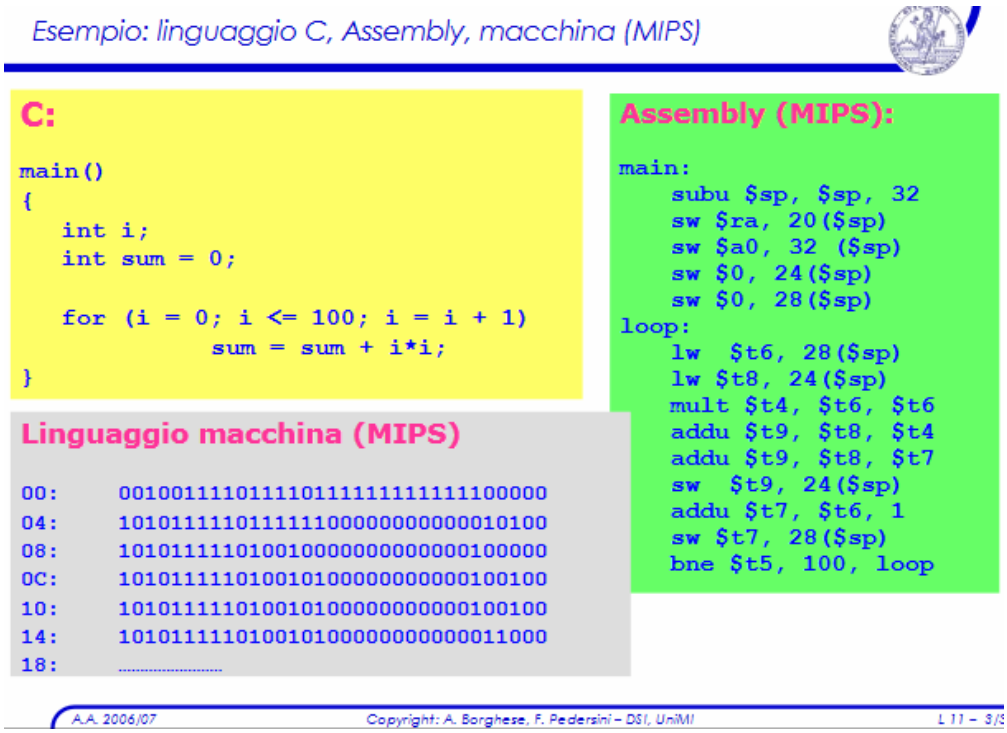
A3 E2 24 5C …

Scrivere un linguaggio direttamente nella sua forma binaria, denominata **linguaggio macchina**, richiede programmatori altamente specializzati; inoltre il programma scritto risulta difficilmente leggibile e modificabile, e molto faticosamente adattabile a piattaforme con linguaggio macchina diverso. Il primo passo fatto per superare questi ostacoli è stato quello di sostituire ai codici binari istruzioni simboliche (**linguaggio Assembly**); successivamente le istruzioni simboliche (ad es. ADD per sommare, MULT per moltiplicare) devono essere trasformate in linguaggio macchina da un software assemblatore.

	READ	X	0	0100	1000
	READ	Y	1	0100	1001
	LOADA	X	2	0000	1000
	LOADB	Y	3	0000	1001
	MUL		4	1000	
	STOREA	X	5	0010	1000
	WRITE	X	6	0101	1000
	HALT		7	1101	0000
X	INT		8	0000	0000
Y	INT		9	0000	0000

Assembly Linguaggio macchina

Successivamente sono stati ideati i **linguaggi di alto livello**, come C e C++, che hanno una struttura logico-sintattica completamente svincolata dalla piattaforma hardware e dal linguaggio macchina. Questo è possibile demandando a un software specifico il compito di convertire il linguaggio di alto livello in linguaggio macchina.



Per tradurre un linguaggio di alto livello in linguaggio macchina esistono due modi possibili:

- Compilazione (es. C, C++, Pascal): il software **compilatore** traduce l'intero programma in linguaggio macchina e ne produce la versione compilata che sarà effettivamente utilizzata.
- Interpretazione (Pearl, Javascript, Python): il software **interprete** traduce ed esegue il programma istruzione per istruzione.

Principali vantaggi dei linguaggi di alto livello:

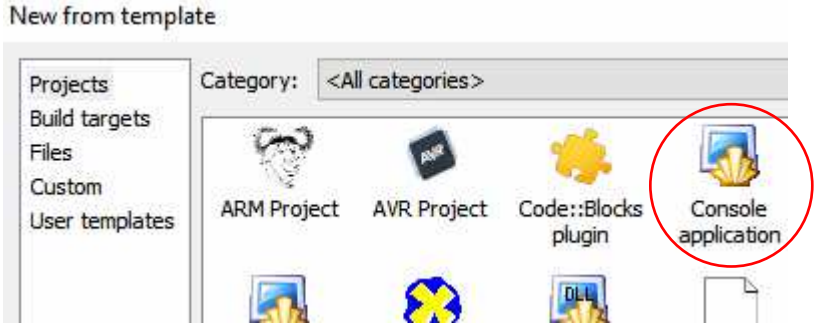
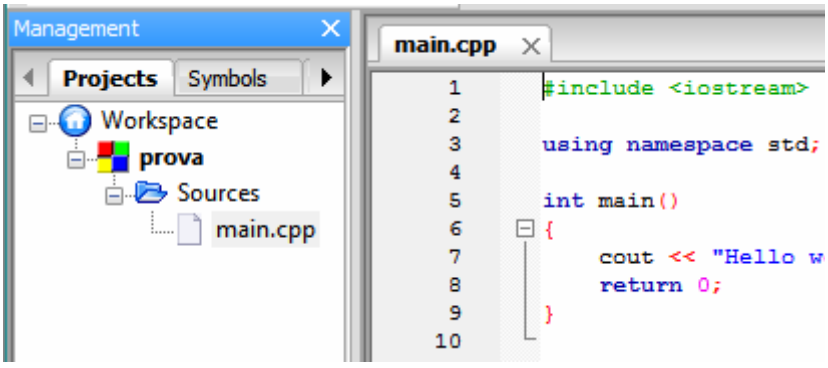
- facilità di scrittura, lettura e modifica del programma, in particolare per correggere gli errori;
- il codice non dipende dalla piattaforma hardware/software che lo deve eseguire: ad es. un programma in C++ può essere eseguito su sistemi Intel o AMD e con diversi sistemi operativi, a patto di disporre del compilatore specifico per la piattaforma utilizzata.

4. Ambiente di sviluppo C++

Un programmatore C++ deve disporre di un software IDE (*Integrated Development Environment* ovvero **Ambiente di Sviluppo Integrato**). I principali compiti di un ambiente di sviluppo:

- fornisce un blocco note evoluto mediante il quale scrivere il programma (che in sé è un file di testo semplice);
- compila il codice, segnalando eventuali errori formali nel codice;
- consente il **debugging**, e cioè l'eliminazione dei *bug* (in italiano "baco", errore di programmazione): permette di eseguire il programma una istruzione alla volta controllando nel contempo i valori delle variabili interessate.

Per realizzare un programma di tipo "*console*", cioè a riga di comando, con Codeblocks:

1. Cliccare	
2. Cliccare "Console application" e Scegliere la cartella in cui memorizzare i file	
3. Scrivere il programma nel file *main.cpp*	
4. Compilare ed eseguire il programma (pulsante *build and run*)	

Se non ci sono errori che impediscono la compilazione nella cartella *bin/Debug* viene salvato il programma compilato con estensione *exe*.