



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

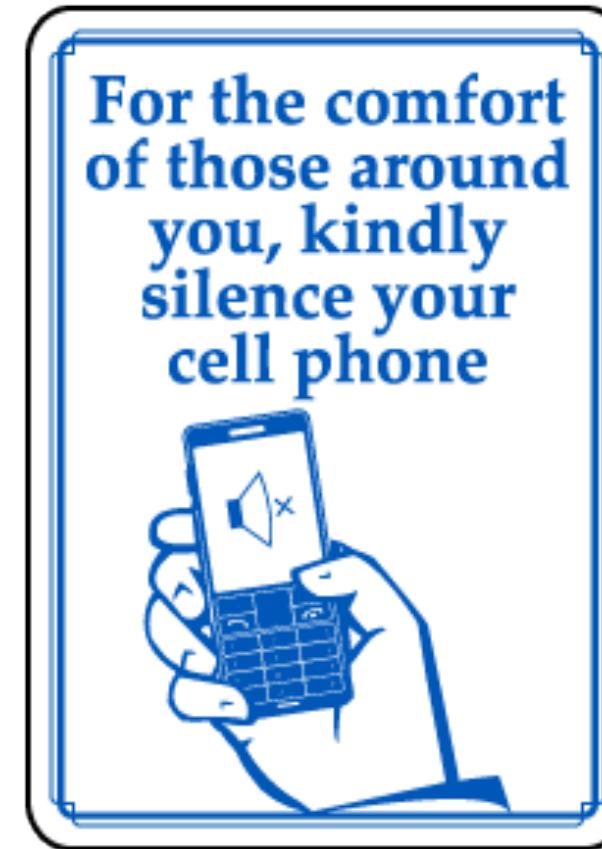
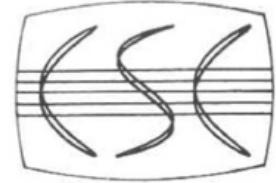
Sergio Canazza
canazza@dei.unipd.it - <http://www.dei.unipd.it/~canazza/>



UNIVERSITÀ
DEGLI STUDI
DI PADOVA



DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE



Docente

- Prof. Sergio Canazza
- Ricerca -> direttore del Centro di Sonologia Computazionale;
per i curiosi:
 - <http://www.dei.unipd.it/~canazza>
 - <http://csc.dei.unipd.it>
- Didattica -> Dipartimento di Ingegneria dell'Informazione
- Recapiti
 - Telefono: 049 827 7790
 - E-mail (meglio!): canazza@dei.unipd.it
 - Ufficio: III piano DEI/G, Dipartimento di Ingegneria
dell'Informazione, Via Gradenigo 6/A
- Orario di ricevimento: giovedì 15:30 – 16:30
- Molto consigliato: appuntamento via e-mail

Richiesta di colloquio



- Illustrate **bene** il problema in un messaggio di **posta elettronica**, eventualmente allegando algoritmo o codice Java
 - **Per rispondere in tempi contenuti, mi devo preparare in anticipo!**
- Scrivete **nome, cognome e matricola**, specificate anche il **corso di laurea**
 - Non sempre è necessario un colloquio, a volte riesco a rispondere adeguatamente via e-mail

Richiesta di colloquio

- Mi devo comprare un computer nuovo, mi consiglia?
 - NO
- Non frequento, mi spiega i cicli?
 - NO
- Ero a lezione, non ho capito i cicli, me li rispiega?
 - NO
- Dopo aver studiato con dedizione i cicli, cercando di chiarire i dubbi con i colleghi e facendo ricerche in biblioteca, non ho capito come funziona questo particolare ciclo **while**, ne possiamo parlare insieme?
 - CERTAMENTE

Studenti del Canale 3

▼ Insegnamenti PRIMO ANNO Corsi di Laurea (Triennali)

1) Matricole e ripetenti

Per gli studenti del **primo anno** e per i **ripetenti**, la suddivisione in canali viene effettuata in base al corso di laurea di appartenenza e all'ultima cifra del numero di matricola. Gli studenti del corso di laurea in **Ingegneria dell'Informazione** che intendono **rifrequentare** gli insegnamenti del primo anno devono seguire la suddivisione indicata per il corso di laurea in Ingegneria dell'Automazione e dei Sistemi.

Ingegneria Biomedica:

- Ultima cifra del numero di matricola da 0 a 4: **Canale 1**
- Ultima cifra del numero di matricola da 5 a 9: **Canale 2**

Ingegneria dell'Automazione e dei Sistemi:

- Ultima cifra del numero di matricola da 0 a 4: **Canale 2**
- Ultima cifra del numero di matricola da 5 a 9: **Canale 3** ←
- Tutte le matricole del curriculum in lingua inglese: **Information Engineering**

Ingegneria delle Telecomunicazioni, Internet e Multimedia:

- Tutte le matricole: **Canale 1**

Ingegneria Elettronica:

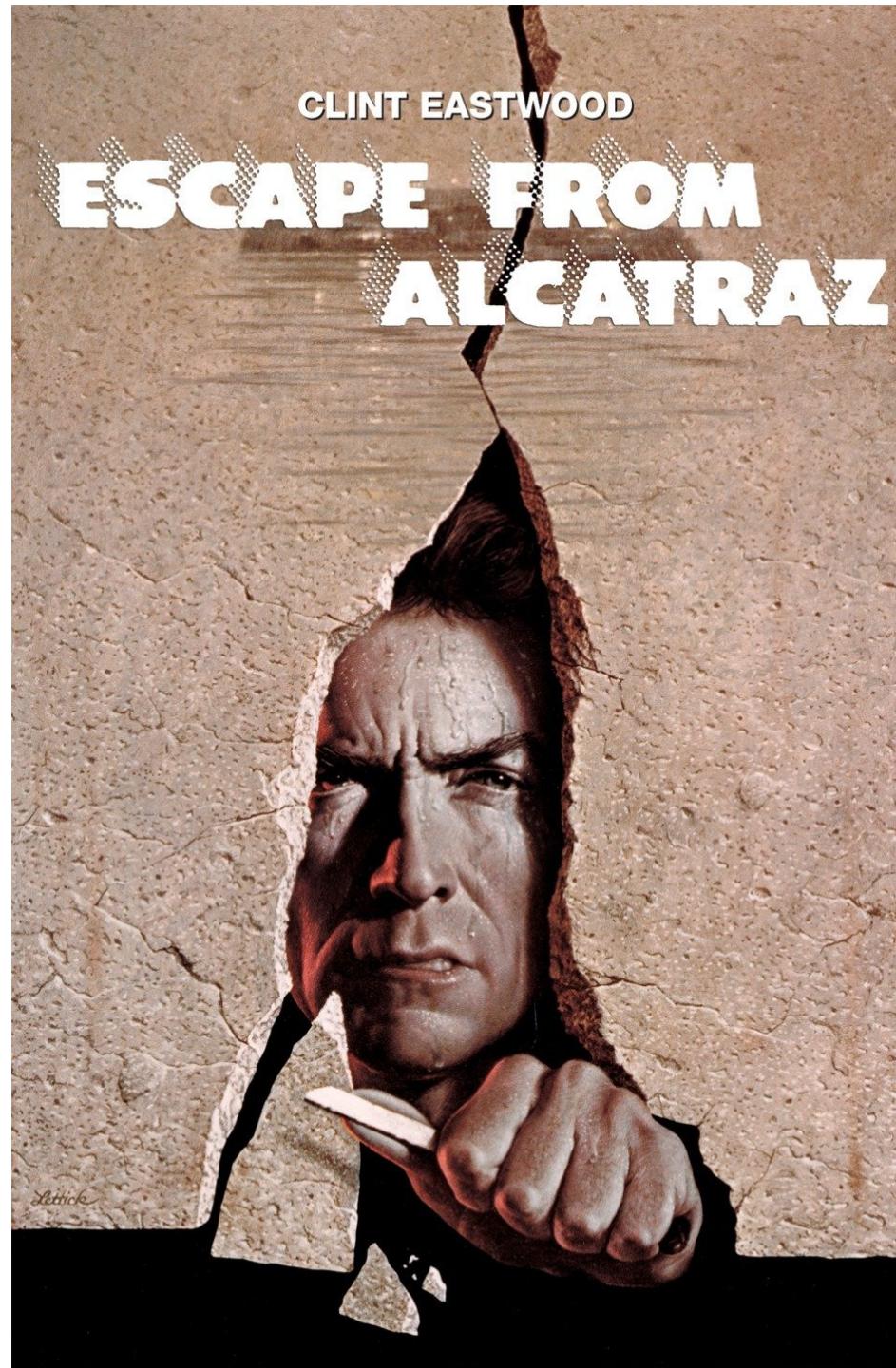
- Tutte le matricole: **Canale 1**

Ingegneria Informatica:

- Ultima cifra del numero di matricola da 0 a 4: **Canale 2**
- Ultima cifra del numero di matricola da 5 a 9: **Canale 3** ←

L'orario delle lezioni è visibile sul **Portale studenti** selezionando la Scuola di Ingegneria, il proprio corso di laurea, l'anno di corso, il canale di appartenenza e la **settimana di lezione**.

<https://stem.elearning.unipd.it/course/view.php?id=2552>



Posso cambiare canale?

- Per ottenere il cambio di canale occorre fare richiesta **motivata** al Presidente del Consiglio di Corso di Studi, tramite le segreterie didattiche
 - Chiedere ai docenti non serve a nulla!
 - L'attribuzione del nuovo canale va poi comunicata al docente del canale “ricevente”
 - Le richieste vengono accolte solo in casi veramente “eccezionali”

Orario

- Lezioni in aula P3 tenute dal docente
 - durata **effettiva 90** minuti, solitamente **senza pausa**
- Esercitazioni in laboratorio Taliercio **non assistite**:
 - Un turno di 3 ore ogni settimana
 - Le esercitazioni sono **guidate**:
 - Il docente NON è presente, ma propone sul sito esercizi da svolgere
 - Sarà spesso (non sempre) presente un *Tutor Junior*

Orario

- Dal 02/10/2023:
 - Lunedì 08:30 – 10:30, Aula P3
 - Mercoledì 16:30 – 18:30, Aula P3
 - Giovedì 16:30 – 18:30, Aula P3
 - Venerdì 10:30 – 12:30, Aula P3
- Laboratorio: dal **25/10/2023**
- Mercoledì 10:30 – 13:30 (Aula Taliercio, Fiera)

Calendario

- Lezioni: 02 ottobre 2023 – 20 gennaio 2024
 - Questo insegnamento comprende 96 ore accademiche
- Vacanze di Natale
 - 23 dicembre 2023 (compreso) – 7 gennaio 2024 (compreso)
- Altre festività:
 - 1 novembre 2023, mercoledì
 - 8 dicembre 2023, venerdì
- Può succedere che, con un certo preavviso, alcune lezioni vengano annullate/recuperate
 - Consultare sempre moodle!



Siti da consultare

...per informazioni amministrative, tecniche, manifesti degli studi, piani di studio, orari, ecc.:

- Moodle area STEM: <http://stem.elearning.unipd.it>
- Sito di Ateneo: <http://www.unipd.it>
- Sito dell'**Aula Didattica Taliercio**:
<http://www.adt.unipd.it>
 - Aula informatica dove si svolgono esercitazioni settimanali
- **Sito del corso** su Moodle STEM (slide, esercizi, laboratori...):
 - **INP3053372 - FONDAMENTI DI INFORMATICA (B)
2023-2024 - Prof. Sergio Canazza Targon**

16:22 Ven 23 set stem.elearning.unipd.it 63%

Unipd | Portale Didattica | Orari | Uniweb | Webmail | Italiano (it) ▾

≡ 800 ANNI UNIVERSITÀ DEGLI STUDI DI PADOVA

Non sei collegato. (Login)

Macroarea STEM

Benvenuti nella nuova piattaforma Moodle!

CATEGORIE DI CORSO

- DIBIO Dipartimento di Biologia
- DIPARTIMENTO DI BIOLOGIA - DiBio
- Dipartimento di Fisica e Astronomia Galileo Galilei
- DIPARTIMENTO DI FISICA E ASTRONOMIA "GALILEO GALILEI" - DFA
- DIPARTIMENTO DI GEOSCIENZE
- ICEA DIPARTIMENTO DI INGEGNERIA CIVILE, EDILIZIA E AMBIENTALE DEPARTMENT OF CIVIL, ENVIRONMENTAL AND ARCHITECTURAL ENGINEERING
- DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE
- dii DIPARTIMENTO DI INGEGNERIA INDUSTRIALE

Cerca corsi

en 23 set stem.elearning.unipd.it

800 ANNI UNIVERSITÀ DEGLI STUDI DI PADOVA

Benvenuto nella piattaforma Moodle dell'Università degli Studi di Padova!

Descrizione dell'accesso

È un tuo diritto possedere un account email unipd. Il formato degli indirizzi email è:

Studenti: nome.cognome@studenti.unipd.it
Docenti e personale: nome.cognome@unipd.it

Con una sola password puoi accedere a tutti i servizi Single Sign On (SSO). Potrai quindi accedere a Moodle e controllare la tua posta con le stesse credenziali.

Per effettuare il login usando il SSO, clicca semplicemente sull'immagine del SSO qui a destra.

Non hai ancora ottenuto la tua email unipd?
Segui queste istruzioni: [Guida al Single Sign On](#)

Sei un docente o un altro utente senza credenziali SSO?

Effettua il login senza Single Sign On

Alcuni corsi possono consentire l'accesso agli ospiti
[Login come ospite](#)

Login con SSO



I miei corsi

INP3053372 - FONDAMENTI DI INFORMATICA (B) 2023-2024

INP3053372 - FONDAMENTI DI INFORMATICA (A) 2022-2023

INP3053372 - FONDAMENTI DI INFORMATICA (B) 2022-2023

INP3053372 - FONDAMENTI DI INFORMATICA (C) 2022-2023

INQ1097764 - FOUNDATIONS OF COMPUTER SCIENCE 2022-2023

INQ0091622 - COMPUTER ENGINEERING FOR MUSIC AND MULTIMEDIA 2022-2023

VLAB-DEI 2022-2023

Ricerca commerciale (contratti conto terzi)

Progetti europei collaborativi

Dopo aver fatto login, è necessario **iscriversi ai singoli insegnamenti**

Iscrizione al corso in Moodle

- L'iscrizione al corso tramite Moodle è NECESSARIA per
 - Accedere al materiale didattico
 - Accedere alle esercitazioni di laboratorio
 - Essere aggiornati in merito a eventuali modifiche al calendario delle lezioni
 - Sostenere l'esame
 - È l'unico modo che abbiamo per "contare" gli studenti interessati all'insegnamento... importante per organizzare il laboratorio e gli appelli d'esame
- **Studenti di anni precedenti che debbano ancora sostenere l'esame DEVONO ISCRIVERSI DI NUOVO in Moodle**

Informazioni sulle lezioni - Appunti delle lezioni

Le lezioni frontalini dell'insegnamento di Fondamenti di Informatica, canale 2, iniziano il 2 ottobre 2023.

Questo sito Moodle del corso verrà "popolato" continuamente con altre informazioni utili, per cui vi invito a controllarlo frequentemente.

Orario delle lezioni:
 Lunedì 08:30 - 10:30, Aula P3, complesso Paolotti, Via Belzoni, 7
 Mercoledì 16:30 - 18:30, Aula P3, complesso Paolotti, Via Belzoni, 7
 Giovedì 16:30 - 18:30, Aula P3, complesso Paolotti, Via Belzoni, 7
 Venerdì 10:30 - 12:30, Aula P3, complesso Paolotti, Via Belzoni, 7

Vengono aggiunte le seguenti lezioni:
 lun 2 ottobre 10:30-12:30; ven 6 ottobre 8:30-10:30; lun 9 ottobre 10:30-12:30; lun 16 ottobre 10:30-12:30

Le lezioni di:
 mercoledì 4 ottobre; giovedì 5 ottobre; mercoledì 11 ottobre; giovedì 12 ottobre
 previste dalle 16:30 alle 18:30, sono anticipate dalle 14:30 alle 16:30

Per l'apprendimento degli argomenti trattati, è anche gratuitamente a disposizione il MOOC di Fondamenti di Informatica da me realizzato per la piattaforma FEDERICA dell'Università Federico II di Napoli.
 Connetersi alla piattaforma:
<https://ims.federica.eu/enrol/index.php?id=393>
 cliccare su "iscriviti" (se già non l'avevi fatto), compilare il modulo.
 Il mio MOOC è ricevibile tramite titolo (Fondamenti di Informatica) e/o docente (Sergio Canazza), e corrisponde al codice: ED9C1977

Si segnalano (e si consiglia la fruizione di) tutte le lezioni, che potranno costituire un ripasso dell'intero programma.

Attività Laboratoriali

In questa sezione si trova materiale utile per un proficuo studio della materia (esercizi, temi d'esame), ma che NON sostituisce in alcun modo la frequenza delle lezioni (in aula o in streaming).

Le attività laboratoriali sono terminate il 10 gennaio 2023.

Il giorno 27 gennaio 2023 ci sarà una sessione di Q&A online, dalle ore 15:00 alle ore 17:00 (si veda il forum Annunci per maggiori informazioni).

Le attività di laboratorio sono svolte in Aula Taliercio. Saranno resi disponibili esercizi (con soluzione) settimanalmente. Durante l'orario laboratoriale (**martedì 8:30-11:20**), per quanto possibile, si cercherà di dare assistenza adeguata tramite collaboratori del docente (Tutor Junior) per aiutare nello svolgimento degli esercizi.

Per le esercitazioni casalinghe del corso è sufficiente un computer qualsiasi, non servono potenze di calcolo rilevanti, e i software da installare (Java Virtual Machine, JVM e un editor di testo) è gratuito e liberamente scaricabile. Per chi desidera fare pratica da casa con l'ambiente Linux, sono a disposizione delle "Macchine Virtuali" (VM) con software già pre-installato (editor di testi e JDK) in ambiente Linux CentOS. Per accedervi, dovete prenotarne preventivamente l'utilizzo collegandovi a:
<https://vlabbooking.vdi.it.unipd.it/>

Trovate una guida che descrive come effettuare la procedura di iscrizione in alto a destra ("Manuale").

Questa guida riassume sia la fase di prenotazione, sia di utilizzo della VM. Questo sito include materiale sul progetto T.2020 relativo a queste Virtual Machine.

Accesso a Moodle

Informazioni sulle lezioni - Appunti delle lezioni

Le informazioni sulle lezioni sono state aggiornate.

Esami

Il 31 maggio 2016 è stato modificato l'art. 22 del Regolamento carriere studenti, reperibile all'indirizzo <http://www.unipd.it/regolamenti-studenti>, con effetti a partire dalla sessione d'esami di settembre 2016.

Invito tutti a leggere con attenzione le nuove disposizioni, che qui riassumo (relativamente agli insegnamenti che, come questo, utilizzano la "registrazione online"):

- dopo la pubblicazione del voto finale in UniWeb, lo studente ha 7 giorni di tempo per decidere se accettare o rifiutare il voto ottenuto (non più, quindi, i mesi che si avevano prima...)
- se decide di rifiutare il voto, deve farlo tramite il consueto sistema on-line (UniWeb); entro i 7 giorni può anche cambiare idea e "annullare il rifiuto"
- trascorsi i 7 giorni, tutti i voti che non sono stati rifiutati saranno verbalizzati (non esiste più l'azione di "accettazione del voto", è implicita se il voto non viene rifiutato); i voti che sono stati rifiutati "non esistono più" e non potranno essere "reclamati" in seguito

Chi, per qualsiasi motivo, avesse urgenza di verbalizzare il voto prima della scadenza dei termini, DEVE COMUNICARLO AL DOCENTE NEL MOMENTO IN CUI FA L'ESAME.

In questa sezione si trovano alcuni temi d'esame con relativa proposta di soluzione.

Esami

ESAMI (temi d'esami)

ESAMI (temi d'esami) (old)

Consegna documento di riconoscimento

Esercitazione di laboratorio ...

Questionario a Risposte ...

Soluzioni del Questionario ...

Prima simulazione d'esam...

Simulazione test: prima part...

Esercizi

File di lavoro per l'esercizio 1

Seconda simulazione d'esam...

ESAMI (temi d'esami)

Nascosta agli studenti

Visita CSC

Nascosta agli studenti

Sondaggio conoscitivo

Nascosta agli studenti

Questionario intermedio sull'organizzazione e l'efficacia dell'insegnamento

Nascosta agli studenti

Apertura: venerdì, 10 novembre 2023, 11:57
 Chiusura: mercoledì, 15 novembre 2023, 09:57

Secondo questionario sull'organizzazione e l'efficacia dell'insegnamento

Nascosta agli studenti

Chiusura: mercoledì, 20 dicembre 2023, 08:57

Iscrizione al corso

- Per partecipare alle attività didattiche del corso (lezioni, laboratorio, **esame finale**) è **necessario**
 - iscriversi al corso dal sito del corso su Moodle
 - L'iscrizione è valida fino al termine delle attività didattiche dell'anno accademico (settembre 2024) e va ripetuta l'anno successivo se l'esame non è stato superato.

Frequenza alle lezioni

- Regolamento didattico
 - La frequenza alle lezioni **non è obbligatoria**
- Frequenza alle lezioni in aula
 - **Importante**
- Frequenza alle esercitazioni in laboratorio
 - **Consigliata**
 - Chi non frequenta i laboratori fa molta più fatica a superare l'esame
- Analisi statistica su corsi di anni precedenti

Obiettivi formativi

- Quanto bisogna studiare:
 - Risposta ovvia: **molto**
 - Risposta sincera: **molto**
 - Risposta tecnica: dipende dai **crediti formativi universitari**
- Questo è un corso da **12 CFU** (1 CFU=25 ore di studio)
 - Ore complessive: $12 \times 25 = 300$ di cui
 - lezione ed esercitazione in aula: **96**
 - laboratorio (spesso assistito da un tutor): **48**
 - **studio individuale: 156 (circa 10 ore settimanali da ora sino all'esame)**
 - Lo studio individuale comprende anche pratica al calcolatore (in aggiunta al laboratorio)

Obiettivi del Corso

- Il Corso ha l'obiettivo di
 - presentare le **basi teoriche** dell'informatica
 - proporre un **approccio ingegneristico** e progettuale alla programmazione
 - fornire una **visione professionale** alla risoluzione dei problemi
 - **utilizzare concretamente** le caratteristiche di programmazione del linguaggio Java

Programma (1)

- Organizzazione di un elaboratore. Unità centrale di elaborazione, memoria centrale, dispositivi di memoria di massa, dispositivi di ingresso e uscita.
- Il sistema operativo (Linux), sommario delle funzioni, processi, multiprogrammazione.
- Rappresentazione dell'informazione, sistemi di numerazione e conversioni.
- Linguaggi di programmazione. Esecuzione di un programma tramite compilazione/interpretazione. Macchina virtuale Java.
- Concetto di algoritmo, introduzione all'analisi degli algoritmi, esemplari di un problema e loro taglie. Misura della complessità: nel caso peggiore e nel caso medio. Notazione asintotica O-grande.
- Ricorsione, eliminazione della ricorsione.

Programma (2)

- Strutture dati e algoritmi, il concetto di tipo di dato astratto, un'interfaccia Java come definizione di un tipo di dato astratto, realizzazione di un tipo di dato astratto mediante una classe.
- Array, liste, pile e code, realizzazione mediante un array o una catena di celle.
- Ricerca di un elemento in un array e in una lista. Ricerca per bisezione in un array ordinato.
- Tabelle, dizionari, semplice realizzazione di un dizionario mediante un array parzialmente riempito o una lista.
- Algoritmi di ordinamento, ordinamento per selezione, inserzione, mergesort.

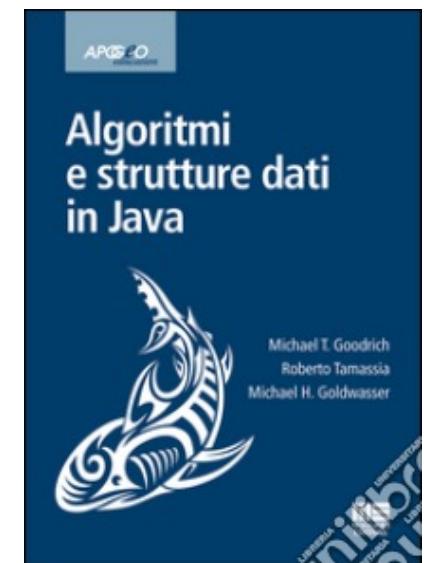
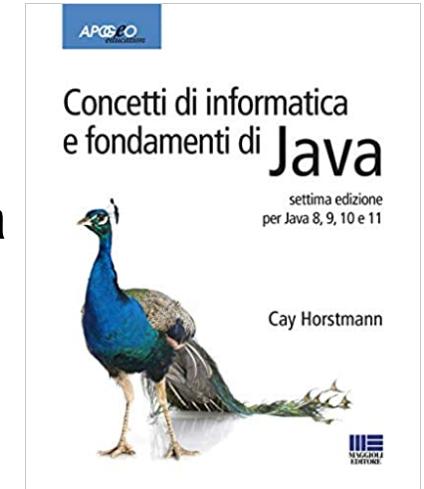
Programma (3)

- Il linguaggio di programmazione Java.
- Tipi di dati elementari e oggetti, riferimenti, operatori ed espressioni, istruzioni di controllo, classi e interfacce.
Campi e metodi di un classe.
- Introduzione alla programmazione ad oggetti.
- Polimorfismo ed ereditarietà.
- Operazioni di ingresso e uscita da ingresso e da uscita standard, operazioni di ingresso e uscita da file di testo.
- Gestione elementare degli errori.

- **Realizzazione di semplici progetti di programmazione nei laboratori didattici**

Libro di testo e libri consigliati

- Libro di testo:
 - Cay Horstmann, "Concetti di informatica e fondamenti di Java - Per Java 8, 9, 10 e 11", **settima Edizione**, Ed. Apogeo, ISBN: 8891639435, 2020
 - Per la parte del corso dedicata allo studio delle strutture di dati, i primi 6 capitoli del volume:
 - M.T. Goodrich, R. Tamassia, M.H. Goldwasser, "Algoritmi e strutture dati in Java", 2015
 - Unico **prerequisito**: dimestichezza con l'uso di un PC e concetti di base del suo funzionamento.





Esami

- Calendario: <http://esami.dei.unipd.it>
 - 1o appello: 30/01/2024 (più 31 e 1/02)
 - 2o appello: 21/02/2024 (più 22 e 23)
 - 3o appello: 3/07/2023 (più 4)
 - 4o appello: 3/09/2023 (più 4)
- Modalità
 - due prove da sostenere **consecutivamente**
 - prova scritta (teoria), prova pratica (esercizi di programmazione)
 - Va superata la prima prova per accedere alla successiva
- In questo insegnamento **NON** ci sono “prove di accertamento” intermedie

Esami

- Il 31 maggio 2016 è stato modificato l'art. 22 del Regolamento carriere studenti, reperibile all'indirizzo:
<http://www.unipd.it/regolamenti-studenti>
- con effetti a partire dalla sessione d'esami di settembre 2016:
 - dopo la pubblicazione del voto finale in UniWeb, lo studente ha 7 giorni di tempo per decidere se accettare o rifiutare il voto ottenuto (non più, quindi, i mesi che si avevano prima...)
 - se decide di rifiutare il voto, deve farlo tramite il consueto sistema online (UniWeb); entro i 7 giorni può "annullare il rifiuto"
 - trascorsi i 7 giorni, tutti i voti che non sono stati rifiutati saranno verbalizzati (non esiste più l'azione di "accettazione del voto", è implicita se il voto non viene rifiutato); i voti che sono stati rifiutati "non esistono più" e non potranno essere "reclamati" in seguito
 - Chi, per qualsiasi motivo, avesse urgenza di verbalizzare il voto prima della scadenza dei termini, DEVE COMUNICARLO AL DOCENTE NEL MOMENTO IN CUI FA L'ESAME.

Conviene rifiutare un voto?

- Quanto influisce il voto V di un singolo esame sul voto di laurea F ?
 - Dipende da
 - Numero di crediti C del corso
- $F = k + (110/30) * [(180 - C) * M + C * V] / 180$
- $F_{V1} - F_{V2} = (110/30) * C * (V1 - V2) / 180$
 - Non dipende da M degli altri esami...
 - $\Delta F = \Delta V * C * 0.02037$
 - $\Delta F = \Delta V * 0.24$ (per $C = 12$)

Conviene rifiutare un voto?

- Passando da 18 a 30 nel voto di Fondamenti di Informatica, il voto finale di laurea aumenta di 2.88
 - Ma di solito l'alternativa non è tra 18 e 30...
- Il voto finale di laurea aumenta di 0.24 per ogni punto in più nel voto di un esame
 - Rifiutando 22 e prendendo poi 25 si aumenta il voto finale di laurea di poco più di mezzo punto
 - con gli arrotondamenti, si aumenta o di un punto o di... niente
 - Ma rifiutando 22 e prendendo poi 19...



Tasso di disoccupazione confortante

– Disoccupazione

	LM Ingegneria @ UniPD	LM Ingegneria @ Italia	LM @ UniPD	LM @ Italia
a 1 anno	6.6 %	10.9 %	14.3 %	21.4 %
a 3 anni	1.2 %	4.0 %	8.3 %	12.3 %

– Tempo per trovare primo lavoro: **2.8 mesi**

Scuola	Tasso di abbandono dal I al II anno (%)					
	Laurea			Laurea Magistrale a ciclo unico		
	2016/17	2017/18	2018/19	2016/17	2017/18	2018/19
Agraria e Medicina Veterinaria	20,6	20,1	23,1	5,5	13,6	14,0
Economia e Scienze Politiche	17,9	19,5	15,9			
Giurisprudenza	13,5	19,7	16,4	21,5	28,3	21,4
Ingegneria	31,7	31,4	31,3	8,0	14,1	10,5
Medicina e Chirurgia	16,9	18,9	18,2	15,9	6,4	7,6
Psicologia	12,3	13,4	11,8			
Scienze	29,5	27,5	23,7			
Scienze Umane, Sociali e del Patrimonio Culturale	24,6	23,6	24,2	7,1	7,8	8,2
TOTALE	23,9	24,1	23,4	15,1	14,6	12,8

Per il calcolo del tasso di abbandono sono state considerate le coorti di immatricolati 2016/17 per le lauree e le coorti a.a. 2013/14 e a.a. 2014/15 per le lauree magistrali a ciclo unico (rispettivamente di durata 6 e 5 anni).

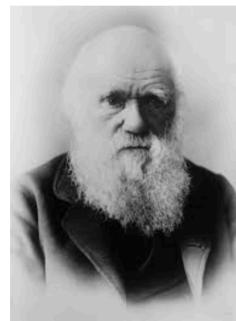
Scuola	Tasso di laureati regolari (%)								
	Laurea			Laurea Magistrale			Laurea Magistrale a ciclo unico		
	2016/17	2017/18	2018/19	2016/17	2017/18	2018/19	2016/17	2017/18	2018/19
Agraria e Medicina Veterinaria	35,5	35,4	33,8	44,8	52,9	57,8	47,4	38,2	37,3
Economia e Scienze Politiche	41,6	43,0	43,8	28,7	39,2	30,7			
Giurisprudenza	11,2	10,9	17,2				8,8	7,2	16,5
Ingegneria	21,4	24,0	25,2	26,3	27,4	31,7	1,2	8,3	3,4
Medicina e Chirurgia	57,5	57,0	58,6	71,9	70,4	68,1	45,3	42,8	59,5
Psicologia	58,1	59,4	56,5	52,4	53,9	57,6			
Scienze	28,9	30,2	29,1	48,1	50,5	52,8			
Scienze Umane, Sociali e del Patrimonio Culturale	33,7	31,6	33,8	31,0	28,2	24,3	43,1	50,6	55,4
TOTALE	35,9	36,4	36,4	39,4	41,1	42,3	29,7	30,4	42,6



Il modello formativo di Ingegneria

- Tipicamente Darwiniano

"*Survival of the fittest*"



*"It is not the strongest
of the species that
survives, nor the most
intelligent, but the one
most responsive to
change"*
(Charles Darwin, 1809)

- Prendiamo studenti da un ambiente fortemente controllato...
 - scuole secondarie di 2° livello ("superiori")
- e li scaraventiamo in un ambiente radicalmente diverso
 - praticamente privo di controllo
- Tipicamente gli studenti:
 - sono liberi di **frequentare** o no
 - sono liberi di **studiare** o no
 - hanno **interazione minima** coi docenti e con gli altri allievi
 - spesso frequentano le peggiori **strutture** di tutto il percorso formativo
- Il fatto che molti abbandonino non è sorprendente
 - **non si adattano** al cambiamento **radicale** che imponiamo loro
- Il modello del 1° anno viene poi riproposto anche negli anni successivi e alla LM
 - quelli che "sopravvivono", per **adattarsi al cambiamento** **impiegano più tempo** del dovuto a laurearsi

Qualche consiglio (1)

- A chi non ha mai avuto a che fare con calcolatori, programmazione, ecc.:
 - Questo insegnamento **non ha prerequisiti**, chiunque lo può seguire con profitto, MA:
 - È **fondamentale** prendere immediatamente confidenza con i *ferri del mestiere*
 - **Installare e usare** da subito l'ambiente di programmazione per il linguaggio Java
 - Svolgere con regolarità gli **esercizi proposti nei laboratori**



Qualche consiglio (2)

- A chi ha già una solida esperienza di programmazione:
 - Questo insegnamento **non** ha lo scopo di formare dei programmatore professionisti
 - Studieremo anche: ricorsione, algoritmi di ricerca e ordinamento, analisi della complessità di un algoritmo, tipi di dati astratti
 - Attenzione quindi a non considerarvi promossi in partenza!

Qualche consiglio (3)

- Non considerate questo insegnamento come **estraneo** al vostro indirizzo di studio
 - I contenuti dell'insegnamento vi serviranno:
 - Nel prosieguo del vostro corso di studi
 - Per la vostra tesi di laurea
 - Nel mondo del lavoro
- ... Inoltre chi non supera questo insegnamento **non si laurea!**

Laboratorio: dove e quando

- Aula didattica Taliercio: <http://www.adt.unipd.it>
 - Via Niccolò Tommaseo, 59
- Turni di **3 ore**, parte delle quali assistite da Tutor
- Primo laboratorio: **martedì 25 ottobre 10:30**
 - Lezione del Tutor su concetti introduttivi:
<http://www.adt.unipd.it/usoaula/Introduzione/index.php3>



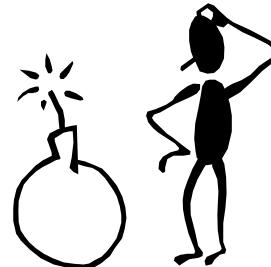
Prendere confidenza con il computer

- Per alcuni di voi il laboratorio didattico di questo corso costituisce (forse) il primo approccio con un computer
- I primi passi da compiere per acquisire confidenza con il nuovo strumento di lavoro sono:
 - accedere al computer (***fare log-in***)
 - capire i concetti di ***file*** e di cartella (***directory***)
 - scrivere un programma Java
 - individuare il compilatore e l'interprete Java
 - compilare un programma Java da ***terminale***
 - eseguire programmi



Legenda di alcuni simboli grafici (di mio utilizzo)

Errori frequenti



Consigli per la produttività



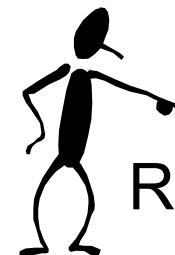
Suggerimenti per la qualità



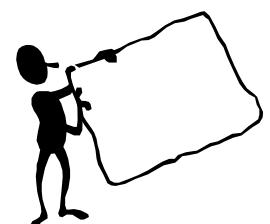
Argomenti avanzati



Accenni ad argomenti
che verranno
approfonditi in seguito



Regole di sintassi Java



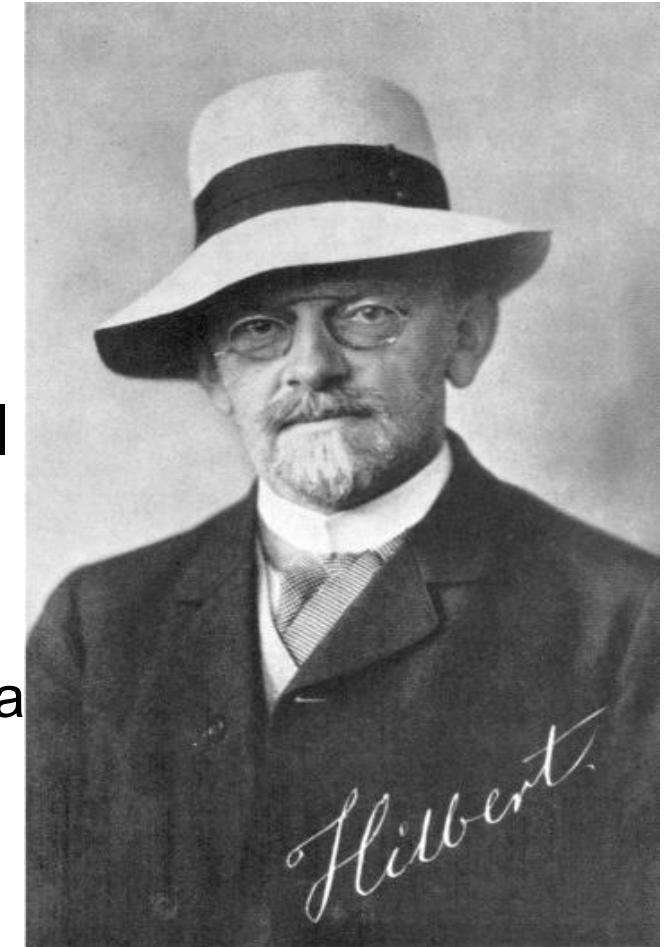
Note di cronaca

Le prime domande

- Cos'è l'informatica?
- Cos'è un computer?
- Cos'è un programma?
- Cos'è la programmazione?
- Cos'è un algoritmo?

Hilbert e il formalismo

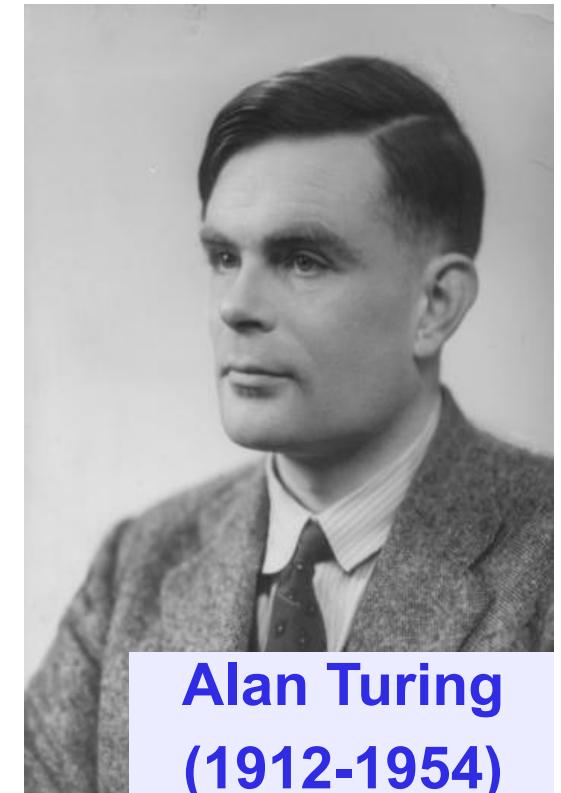
- Alla fine dell'800 la matematica si interroga sui propri fondamenti
- David Hilbert propone 23 problemi matematici fondamentali, da risolvere nel corso del XX secolo
 - Il *problema della decisione*: esiste un metodo, universalmente valido, che permetta di stabilire con sicurezza la verità o la falsità di un qualsiasi enunciato della logica formale?
 - Hilbert è fiducioso: in matematica non esiste l'*Ignorabimus!*



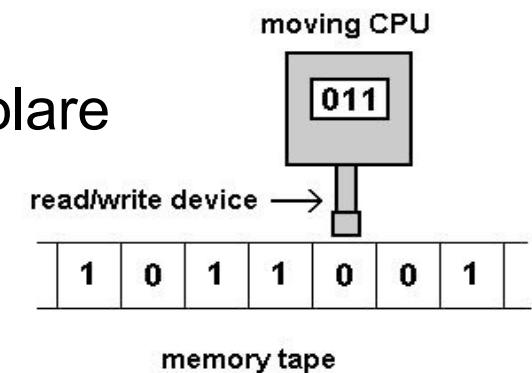
David Hilbert
(1862-1943)

Turing, Church: teoria degli algoritmi

- Alan Turing e Alonzo Church dimostrano (~1936) che tale procedura non esiste!
 - Esistono proposizioni “indecidibili”
- Viene introdotto il concetto di *algoritmo*
 - Una sequenza di operazioni “elementari” che compongono una procedura di calcolo
- Viene definita la macchina di Turing
 - Un computer immaginario
 - Una macchina universale, in grado di calcolare ogni funzione calcolabile



Alan Turing
(1912-1954)



Turing, Church: teoria della complessità

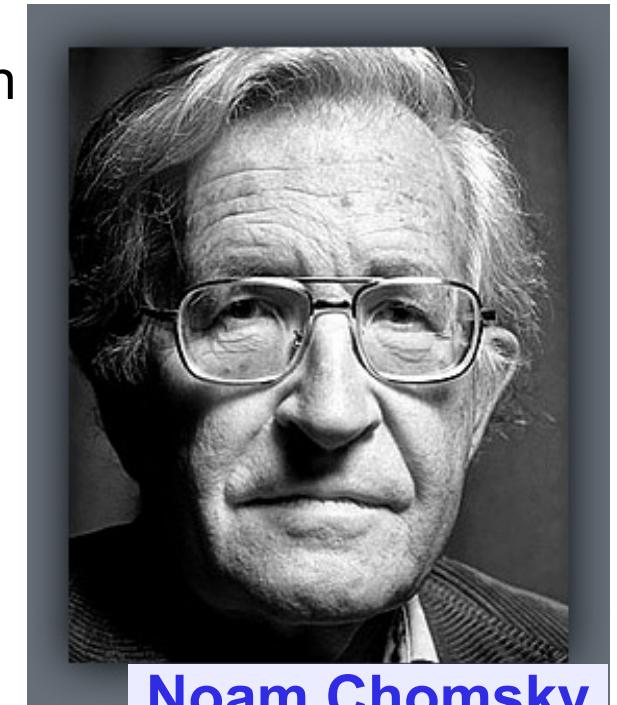
- Tra le proposizioni decidibili – o funzioni calcolabili – o problemi risolubili – alcune sono più “facili” di altre
 - Esistono diverse **classi di complessità**, corrispondenti al tempo necessario per risolvere i problemi
 - Alcuni problemi non sono risolubili in pratica, perché possono essere risolti solo in tempi immensi (miliardi di anni)



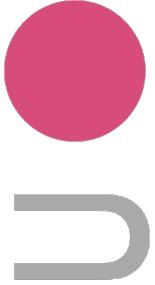
Alonzo Church
(1903-1995)

Chomsky: i linguaggi formali

- Nel '900 linguisti e matematici cercano di formulare un approccio strutturale allo studio del linguaggio
 - **Linguaggi formali:** definiti da insiemi di produzioni grammaticali che li generano
- Chomsky identifica quattro tipi di linguaggi formali, caratterizzati da diverse grammatiche
 - Universali, sensibili al contesto, liberi da contesto, regolari
 - Servono a classificare linguaggi informatici e automi

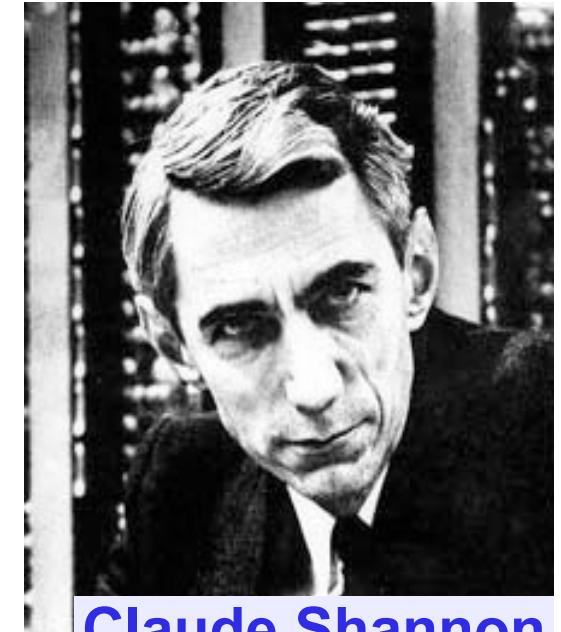


Noam Chomsky
(1928)



Shannon: la teoria dell'informazione

- Pone basi teoriche per la progettazione di **circuiti digitali** (reti logiche)
 - Costituiti da **bit** (binary digit), variabili che assumono solamente i valori 0 e 1
- Fonda la **teoria dell'informazione**
 - Problema di ricostruire in maniera affidabile informazioni trasmesse da un **mittente** e affette da **rumore**
 - Teoria alla base della costruzione di **reti di calcolatori**



Claude Shannon
(1916-2001)

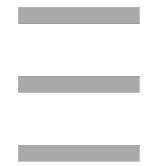
Informazione ed entropia

Pensavo di chiamarla informazione, ma la parola era fin troppo usata, così decisi di chiamarla incertezza. Quando discussi della cosa con John Von Neumann, lui mi disse che avrei dovuto chiamarla entropia, per due motivi: "Innanzitutto, la tua funzione d'incertezza è già nota nella meccanica statistica con quel nome. In secondo luogo, e più significativamente, nessuno sa cosa sia con certezza l'entropia, così in una discussione sarai sempre in vantaggio"

(Claude Shannon)

- L'informazione è ciò che, per un osservatore posto in una situazione in cui si hanno almeno due occorrenze possibili, supera un'incertezza e risolve un'alternativa
- In una comunicazione, che avviene attraverso un alfabeto di simboli finito, l'informazione viene associata a ciascun simbolo trasmesso e viene definita come la riduzione di incertezza che si poteva avere a priori sul simbolo trasmesso
- L'informazione collegata a un simbolo (incertezza associata a un esperimento) è definita:
 $I = -\log_2 p_i$
dove p_i è la probabilità di trasmissione di quel simbolo
- Si definisce bit la quantità di incertezza di un esperimento che ha soli due esiti equiprobabili





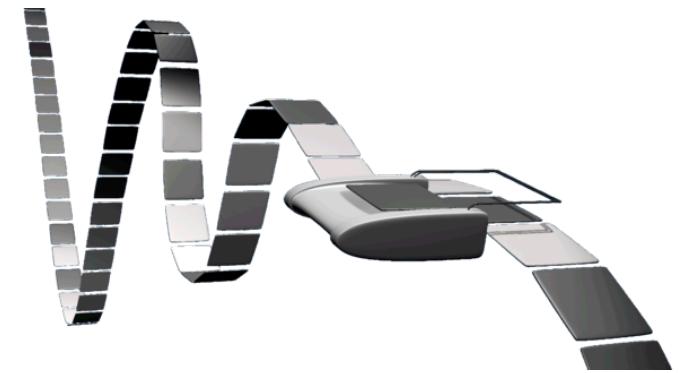
Informatica

«L'informatica non riguarda i computer più di quanto l'astronomia riguardi i telescopi» (Edsger Wybe Dijkstra)

- Inform(ation autom)atique: Philippe Dreyfus nel 1962 introduce il termine *informatique* per definire il trattamento automatico dell'informazione.
- Studio sistematico dei processi algoritmici che descrivono e trasformano l'informazione, per es.:
 - Studio dei linguaggi formali e degli automi
 - Studio della complessità computazionale, in particolar modo per la minimizzazione del numero di istruzioni da eseguire per la risoluzione di un problema e per la ricerca di algoritmi approssimati per risolvere problemi NP-hard
 - Crittologia, la scienza che studia i metodi per rendere un messaggio incomprensibile a chi non sia in possesso di una chiave di lettura del messaggio stesso
 - Teoria dei codici, utilizzata per la compressione dati (codifica di sorgente) o per aumentare l'integrità dei dati (codifica di canale)
 - Ricerca operativa, per fornire strumenti matematici di supporto alle attività decisionali



Il computer



- Dal latino *computare* (calcolare)
- Dispositivo fisico che implementa il funzionamento di una macchina di Turing
 - Sistema formale
 - Sequenza di caselle infinita in cui sono registrati simboli di un alfabeto finito
 - Dotata di una testina di lettura e scrittura
 - A ogni istante si può trovare in uno stato di un insieme finito di stati.
 - Dotata di un repertorio finito di istruzioni che determinano la sua evoluzione in conseguenza dei dati iniziali
 - L'evoluzione si sviluppa per passi successivi che corrispondono a una sequenza discreta di istanti successivi
 - Ogni passo dell'evoluzione viene determinato dallo stato attuale **s** nel quale la macchina si trova e dal carattere **c** che la testina di I/O trova sulla casella del nastro su cui è posizionata e si concretizza nell'eventuale modifica del contenuto della casella, nell'eventuale spostamento della testina di una posizione verso destra o verso sinistra e nell'eventuale cambiamento dello stato
 - Le azioni effettuate a ogni passo sono determinate dall'istruzione che ha come prime due componenti **s** e **c**; le altre tre componenti dell'istruzione forniscono nell'ordine il nuovo stato, il nuovo carattere e una richiesta di spostamento verso sinistra, nullo o verso destra.
- Elaboratore: enfasi sulle capacità di elaborazione
- Ordinateur: enfasi sulle capacità di organizzare i dati
- Computer (calcolatore): discendente delle calcolatrici

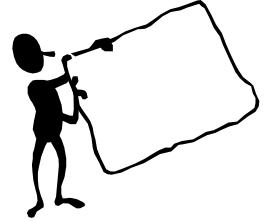
Il computer

- Un computer è una macchina che
 - ***memorizza dati*** (numeri, parole, immagini, suoni...)
 - ***interagisce con dispositivi*** (schermo, tastiera, mouse...)
 - ***esegue programmi***
- Ogni programma svolge una diversa funzione, anche complessa
 - impaginare testi o giocare a scacchi
 - I programmi sono sequenze di istruzioni che il computer esegue e di decisioni che il computer prende per svolgere una certa attività

L'uso dei calcolatori

"Che bisogno ha una persona di tenersi un computer in casa?"
(Kenneth Olsen, fondatore della Digital, 1977)

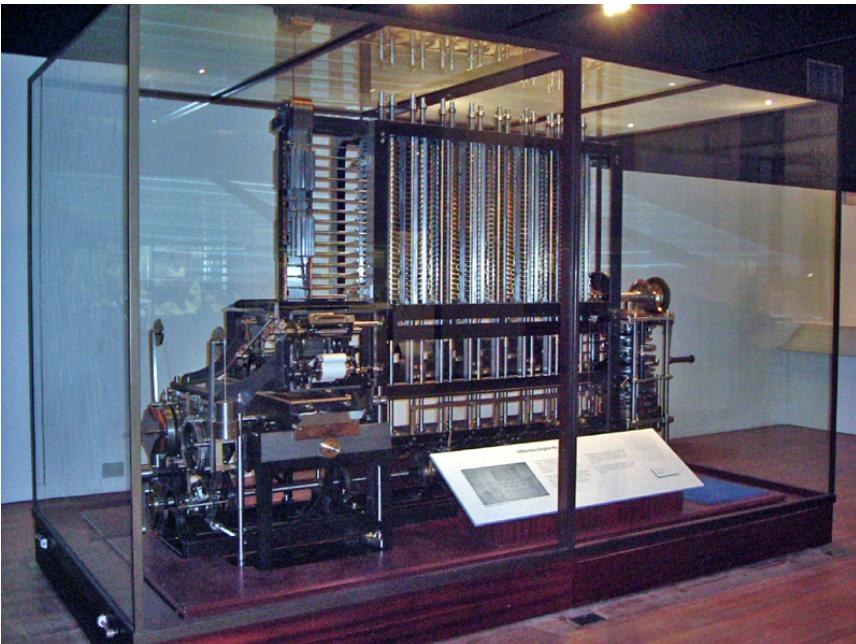
- A casa:
 - Elettrodomestici
 - Immagini digitali nei televisori, suono digitale nei compact disk
 - Dispositivi indossabili (occhiali, orologi)
- Mezzi di trasporto
- Sistemi informativi (basi di dati): supermercati, biblioteche, sul personale di un'organizzazione ecc.
- Fabbriche: sensori + elaboratori + attuatori
- Pagamento elettronico
- Office automation
- Comunicazione molti a molti
- Applicazioni medicali
- Simulazioni scientifiche



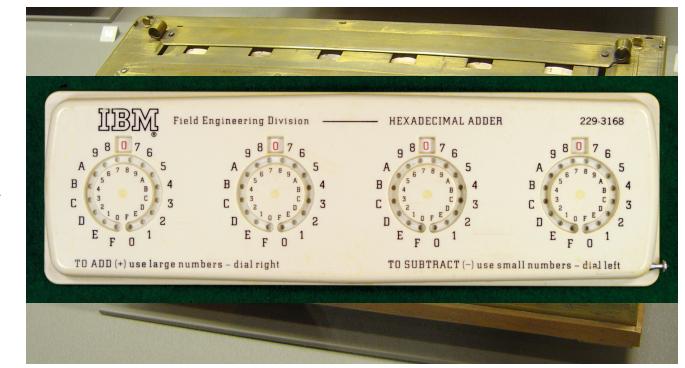
Un po' di storia

- Circa 2400 a.C.: invenzione del abaco
- 1621 d.C.: invenzione del regolo
- 1642: Blaise Pascal crea la prima macchina meccanica per il calcolo delle somme (pascalina)

differenziale di Babbage (1849)
(tabulare funzioni polinomiali per approssimare funzioni trigonometriche e logaritmiche)



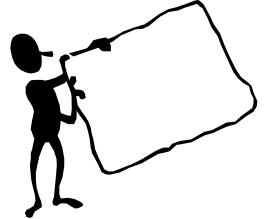
pascalina



Antikythera (II secolo a.c.)
Strumento per predire eclissi e il movimento della luna



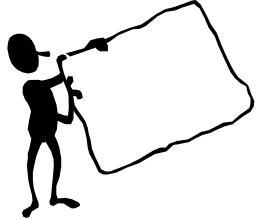
Un po' di storia



"Questa invenzione dell'energia elettrica è un fallimento totale"
(Erasmus Wilson, presidente dello Stevens Institute of Technology, 1879)

- 1843: Ada Lovelace (la prima programmatrice al mondo) pubblica le proprie annotazioni
- 1890: Viene utilizzata l'elettricità in un progetto di elaborazione dei dati (schede perforate)
- 1900: Prima macchina automatica a schede perforate
- 1944: Primo computer elettromeccanico

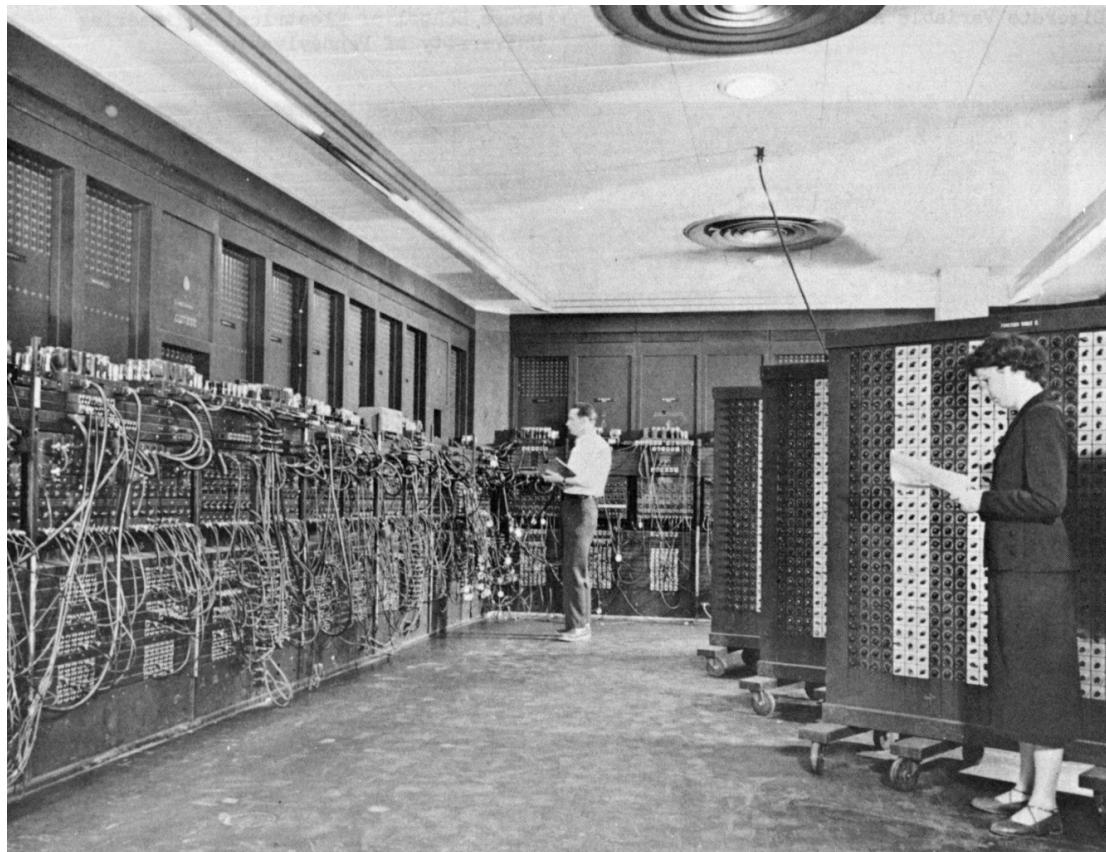




Un po' di storia

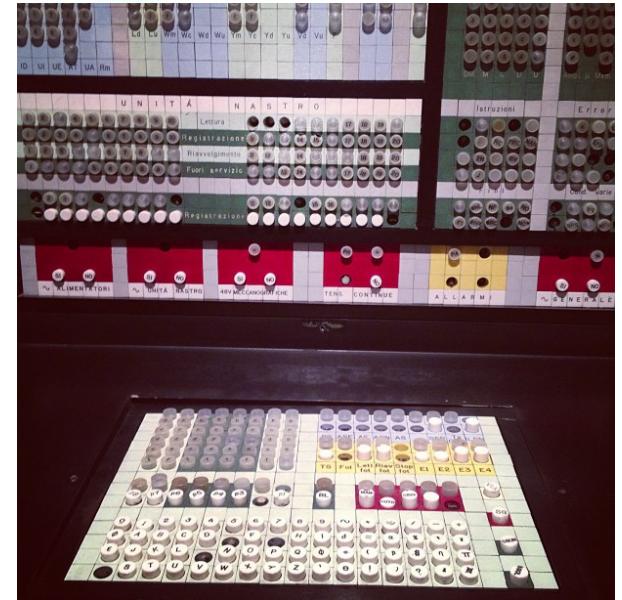
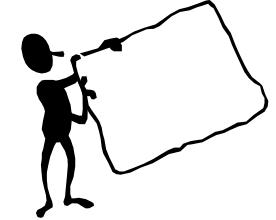
L'unità di calcolo sull'ENIAC è dotata di 18.000 valvole e pesa 30 tonnellate, ma può darsi che in futuro i computer abbiano soltanto 1000 tubi e pesino soltanto una tonnellata e mezza
(da *Popular Mechanics*, 1949)

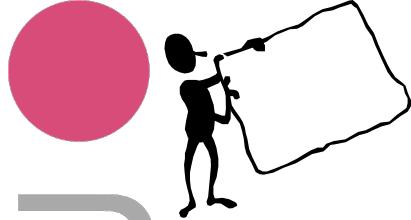
- 1946: Primo computer elettronico negli Stati Uniti
 - ENIAC: *Electronic Numerical Integrator And Computer*



Un po' di storia

- 1952: Il computer UNIVAC prevede correttamente l'elezione del presidente americano Eisenhower
- 1957: Olivetti ELEA 9003. **Primo computer** commerciale totalmente a **transistor** del mondo
- 1969: Nasce le rete ARPANET che darà l'origine a Internet
- 1967: La prima calcolatrice portatile Sharp EL-8
- 1971: Intel 4004. Primo microprocessore
- 1977: Il computer Apple][(primo personal computer prodotto su scala industriale)





U

II

U
II
Engineering

Un po' di storia

"640 Kbytes should be enough for anybody"
(Bill Gates, 1981)



- 1982: Computer portatili, compact disk
- 1982: La prima stampante laser, il *desktop publishing*
- 1985: Telefoni cellulari, il computer pervasivo
- 1993: Desktop multimediali
- 1994: Trasmissione wireless per il computer portatili
- 1998: Inizia la transizione dalle videocassette ai DVD
- 2001: Apple iPOD, *Discoteca 2.0: dal concept al DB*
- 2006: Nintendo Wii, *Il corpo reintrodotto*
- 2010: Apple iPAD, *Biblioteca 2.0? dalla narrazione al DB*

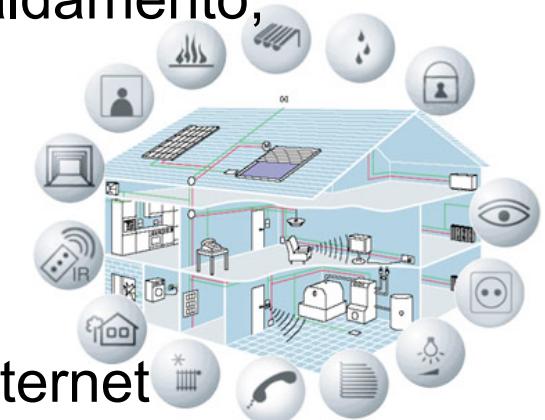
Direzioni dello sviluppo dei computer

“Penso che nel mondo ci sia mercato forse per 4 o 5 computer”

(Thomas Watson, Presidente della IBM, 1943)

“Abbiamo un computer qui a Cambridge, ce n'è uno a Manchester e uno al laboratorio nazionale di fisica. Immagino che sarebbe giusto averne uno anche in Scozia, ma non di più. (Douglas Hartree, fisico inglese 1951)

- (almeno) Tre direzioni dello sviluppo dell'HW:
 - Miniaturizzazione (pervasivi)
 - Potenza di calcolo (multimediali)
 - Economia (immersi), IoT
 - Quantum Computing
 - *Informatica pervasiva e immersa*: i “microcontrollori” nei dispositivi elettronici “intelligenti”:
 - Negli elettrodomestici (forno, lavatrice, TV, irrigazione, allarme, illuminazione, ...)
 - Nelle automobili, negli oggetti indossabili
 - AI
 - Convergenza con la telecomunicazione
 - Per esempio, la TV/il cellulare con accesso

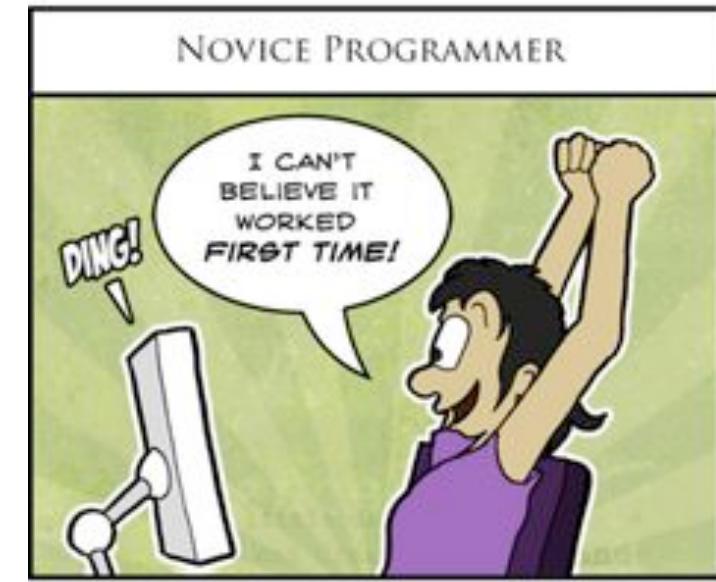


Cos'è un programma?

- Nonostante i programmi siano molto sofisticati e svolgano funzioni molto complesse, le istruzioni di cui sono composti sono ***molto elementari***, e.g.:
 - estrarre un numero da una posizione della memoria
 - sommare due numeri
 - inviare la lettera **A** alla stampante
 - accendere un punto rosso in una posizione dello schermo
 - se un dato è negativo, proseguire il programma da una certa istruzione anziché dalla successiva (**decisione**)
- L'**elevatissimo numero** di istruzioni presenti in un programma e la loro esecuzione ad **altissima velocità** garantisce un'interazione fluida

Cos'è la programmazione?

- Un programma descrive al computer, in estremo dettaglio, la sequenza di passi necessari per svolgere un particolare compito
- L'attività di progettare e realizzare un programma è detta programmazione
 - *Programmatore: un organismo biologico che trasforma caffina e pizza in software*
- **Scopo di questo corso** è fornire la competenza per scrivere semplici programmi usando il linguaggio di programmazione Java



Cos'è la programmazione?

- **Usare** un computer **non** richiede alcuna attività di programmazione
 - così come per guidare un automobile non è necessario essere un meccanico
- Al contrario, un **informatico professionista** solitamente svolge una intensa attività di programmazione, anche se la programmazione non è l'unica competenza che deve avere
 - La programmazione è una parte importante dell'informatica, ed è un'attività che **in genere** affascina gli studenti e li motiva allo studio

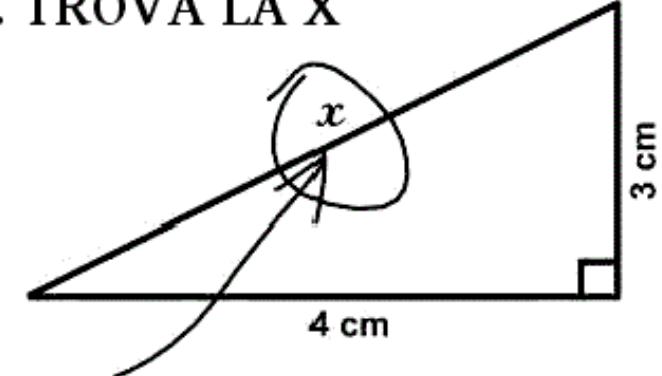
The life of a programmer:



YES! The code is working again!
(And I've got no fucking clue why!)

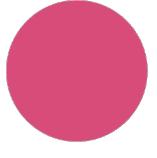
Cos'è un algoritmo?

3. TROVA LA X



ECCOLA QUI

- Quale tipo di problemi è possibile risolvere con un computer?
 - Dato un insieme di fotografie di paesaggi, qual è il paesaggio **più rilassante**?
 - Avendo depositato ventimila euro in un conto bancario che produce il 5% di interessi all'anno, capitalizzati annualmente, quanti anni occorrono affinché il saldo del conto arrivi al doppio della cifra iniziale?
- Il primo problema non può essere risolto dal computer. **Perché?**



Cos'è un algoritmo?

- Il primo problema non può essere risolto dal computer perché non esiste una **definizione** di **paesaggio rilassante** che possa essere usata per confrontare ***in modo univoco*** due paesaggi diversi
- ***Un computer può risolvere soltanto problemi che potrebbero essere risolti anche manualmente***
 - **è solo molto più veloce, non si annoia, non fa errori**
- Il secondo problema è certamente risolvibile manualmente, facendo un po' di calcoli...

Cos'è un algoritmo?

- Si dice **algoritmo** la **descrizione** di un metodo di soluzione di un problema che
 - sia **esegibile** (sequenza di passi eseguibili)
 - sia **priva di ambiguità**
 - arrivi a una conclusione con un **numero finito** di passi
- Un computer può risolvere soltanto i problemi per i quali sia noto un algoritmo
- Etimologia
 - Muḥammad ibn Mūsā al-Khwārizmī
(c. 780 – c. 850)
 - Matematico, astronomo, geografo persiano



Un esempio di algoritmo

Problema: Avendo depositato 20000€ in un conto bancario che produce il 5% di interessi all'anno, capitalizzati annualmente, quanti anni occorrono affinché il saldo arrivi al doppio della cifra iniziale?

Algoritmo:

1. Anno attuale è 0 e saldo attuale è 20000€
2. Ripetere i successivi passi 3 e 4 finché il saldo è minore di 40000€, poi passare al punto 5
3. Aggiungere 1 al valore dell'anno attuale
4. Il nuovo saldo attuale è il valore del saldo attuale moltiplicato per 1.05 (aggiungiamo il 5%)
5. Il risultato è il valore dell'anno attuale

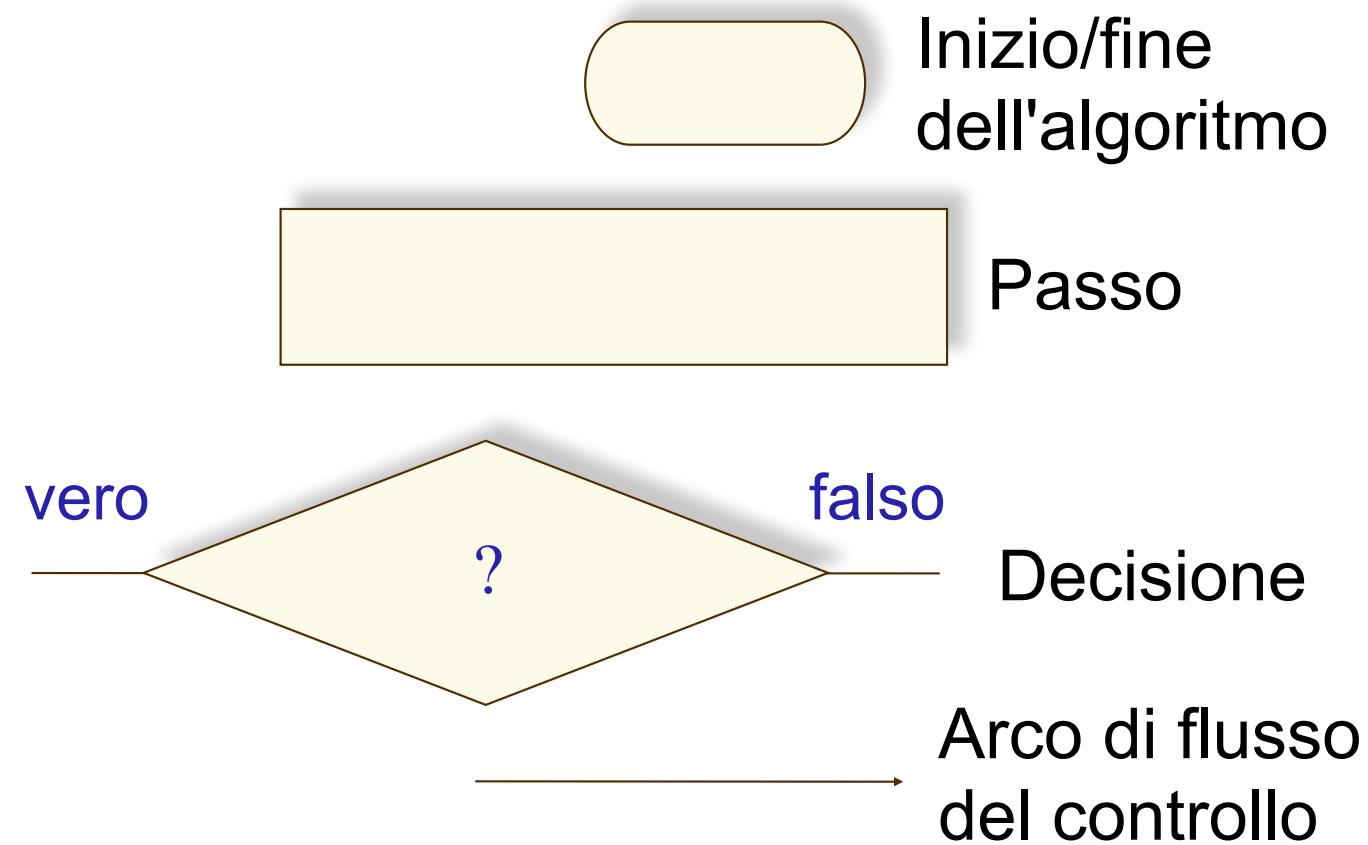
anni	saldo
0	20.000,00
1	21.000,00
2	22.050,00
3	23.152,50
4	24.310,13
5	25.525,63
6	26.801,91
7	28.142,01
8	29.549,11
9	31.026,56
10	32.577,89
11	34.206,79
12	35.917,13
13	37.712,98
14	39.598,63
15	41.578,56

Un esempio di algoritmo

- Il metodo di soluzione proposto
 - è **non ambiguo**, perché fornisce precise istruzioni su cosa bisogna fare ad ogni passaggio e su quale deve essere il passaggio successivo
 - è **eseguibile**, perché ciascun passaggio può essere eseguito concretamente (se, ad esempio, il metodo di soluzione dicesse che il tasso di interesse da usare al punto 4 è variabile in dipendenza da fattori economici futuri, il metodo non sarebbe eseguibile...)
 - **arriva a conclusione in un tempo finito**, perché per ogni passo il saldo aumenta

Diagrammi di flusso

- Semplici algoritmi sono a volte rappresentati graficamente per mezzo di diagrammi di flusso.
- Simboli usati:



A cosa servono gli algoritmi

- L'identificazione di un algoritmo è un requisito **indispensabile** per risolvere un problema (con il computer o senza)
- La scrittura di un programma per risolvere un problema consiste, in genere, nella traduzione di un algoritmo in un qualche linguaggio di programmazione
 - Prima di scrivere un programma, **è necessario individuare e descrivere un algoritmo**
- La definizione di algoritmi e la misura della loro efficienza è una parte importante dell'informatica (e anche di questo corso)
- **Computational thinking!**
 1. **Abstraction**: Problem formulation;
 2. **Automation**: Solution expression;
 3. **Analyses**: Solution execution and evaluation



L'architettura di un calcolatore

- Per capire i meccanismi di base della programmazione è necessario conoscere gli **elementi hardware** (fisici) che costituiscono un calcolatore
- I calcolatori hanno un'architettura che può essere ricondotta (più o meno facilmente) a uno **stesso modello**
 - Server, personal computer, laptop, tablet, smartphone, smartwatch...
 - le eccezioni più importanti sono alcune macchine a elaborazione parallela e la computazione quantistica



Il modello di John von Neumann

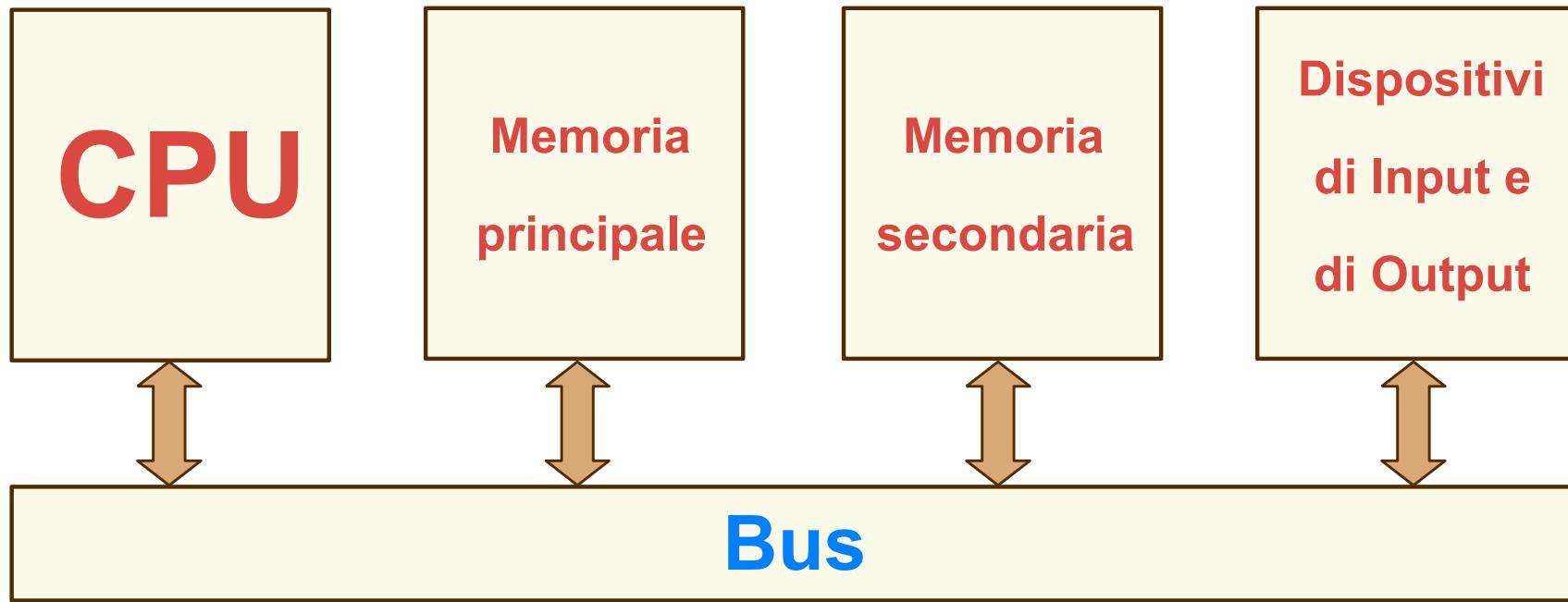
- Primo documento che descrive una macchina elettronica nella cui memoria vengono registrati dati e programma:
 - John von Neumann, *First draft of a report on the EDVAC*, Moore School of Electrical Engineering, University of Pennsylvania, June 30, 1945
- L'architettura dei moderni processori è molto simile a quella descritta nel documento:
Macchine di von Neumann



John von Neumann
(1903-1957)

Il modello di John von Neumann

- L'architettura di von Neumann è composta da **quattro blocchi** comunicanti tra loro per mezzo di un **bus**, un canale di scambio di informazioni



Caratteristiche del collegamento a BUS

Vantaggi

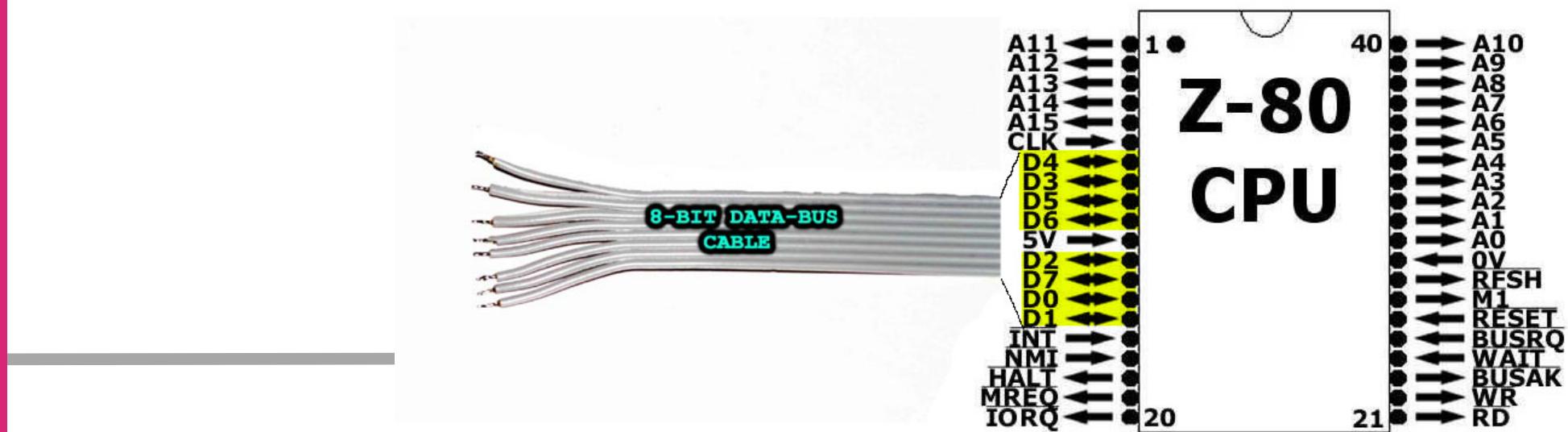
- **Semplicità:** unica linea di connessione implica costi ridotti
- **Estendibilità:** aggiunta di nuovi dispositivi molto semplice
- **Standardizzabilità:** regole precise di comunicazione tra dispositivi diversi

Svantaggi

- **Lentezza:** il bus è utilizzabile solo in mutua esclusione
- **Limitata capacità:** al crescere del n. di dispositivi collegati
- **Sovraccarico del processore:** la **CPU** funge infatti da *master* sul controllo del bus

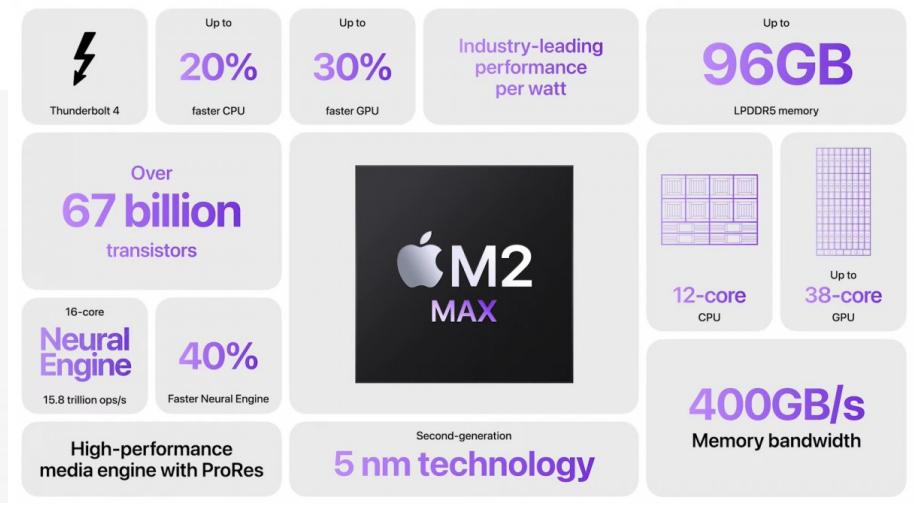
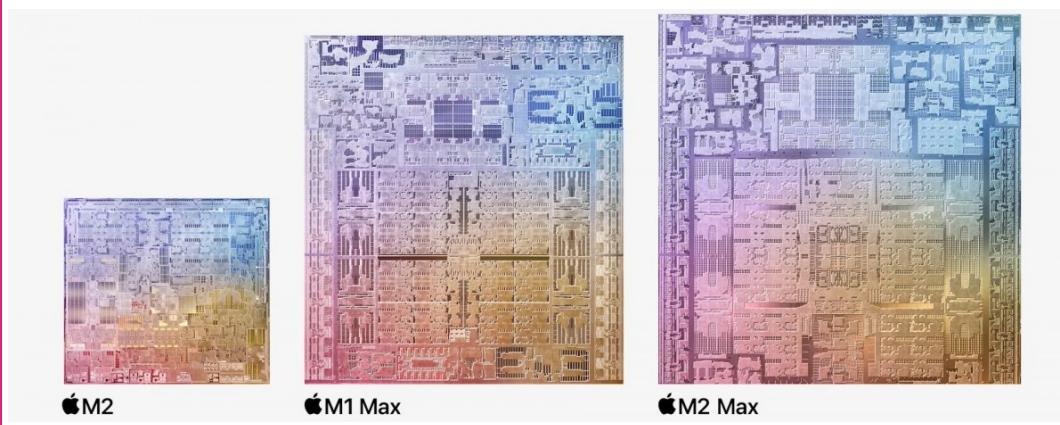
Il bus nel modello di von Neumann

- Il bus è in realtà costituito da tre bus distinti
 - bus dei *dati*
 - bus degli *indirizzi*
 - bus dei *segnali di controllo*
- Sul bus dei dati viaggiano dati **da e verso la CPU**
- Sugli altri bus viaggiano indirizzi e segnali di controllo che **provengono soltanto dalla CPU**



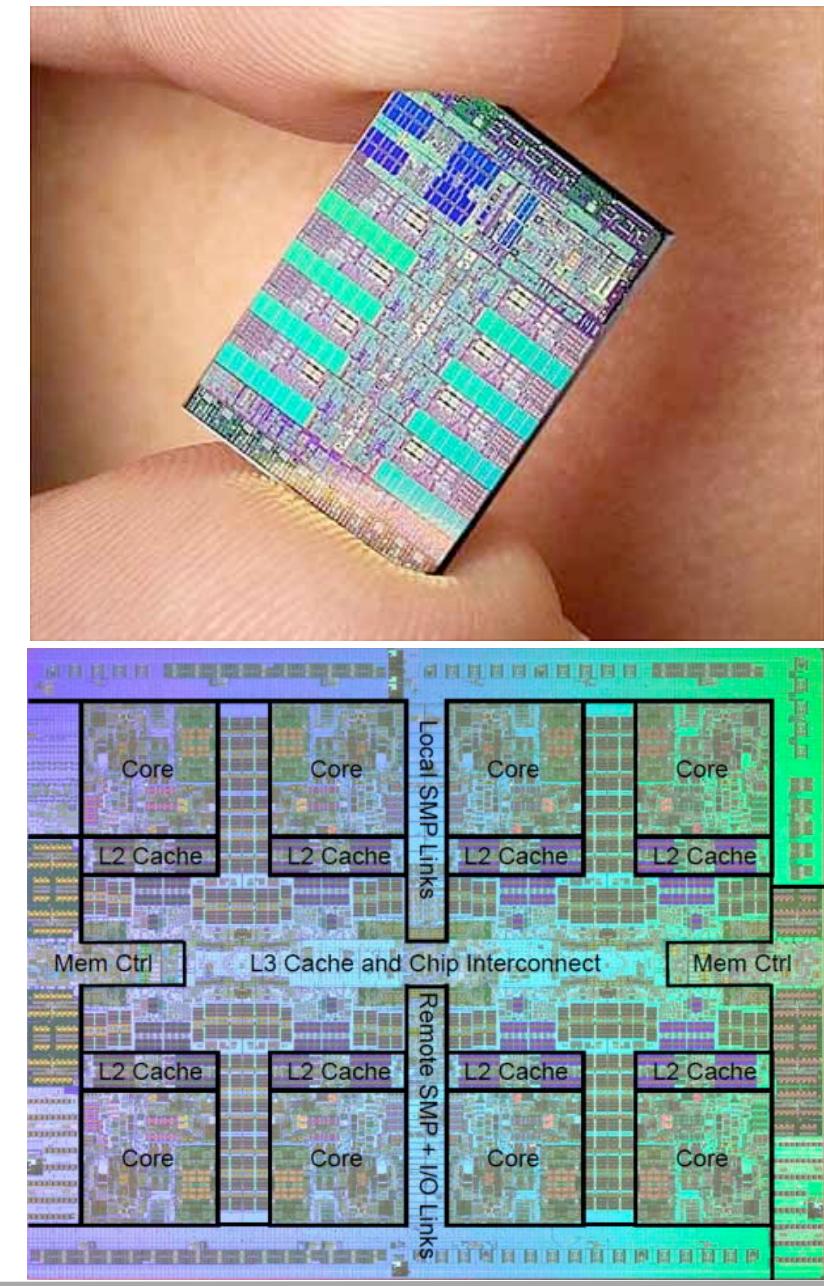
L'unità centrale di elaborazione

- L'unità centrale di elaborazione (**CPU**, *Central Processing Unit*) è il cuore del computer
 - individua ed esegue le istruzioni del programma
 - effettua elaborazioni aritmetiche e logiche con la sua unità logico-aritmetica (**ALU**, *Arithmetic-Logic Unit*)
 - reperisce i dati dalla memoria esterna e da altri dispositivi periferici e ve li rispedisce dopo averli elaborati
 - è costituita da uno o più **chip** (microprocessori)



Il chip della CPU

- Un chip, o ***circuito integrato***, è un componente elettronico con connettori metallici esterni (*pin*) e collegamenti interni (*wire*), costituito principalmente di silicio e alloggiato in un contenitore plastico o ceramico (*package*)
- I collegamenti interni di un chip sono molto complicati; per esempio, il chip Power10 di IBM (2021) è composto fino a 240 cores, 64TB RAM, 7nm, e costituito da ***18 miliardi di transistor***
- Fugaku (Japan): 7.6 milioni di core con 537000 TFlops di picco

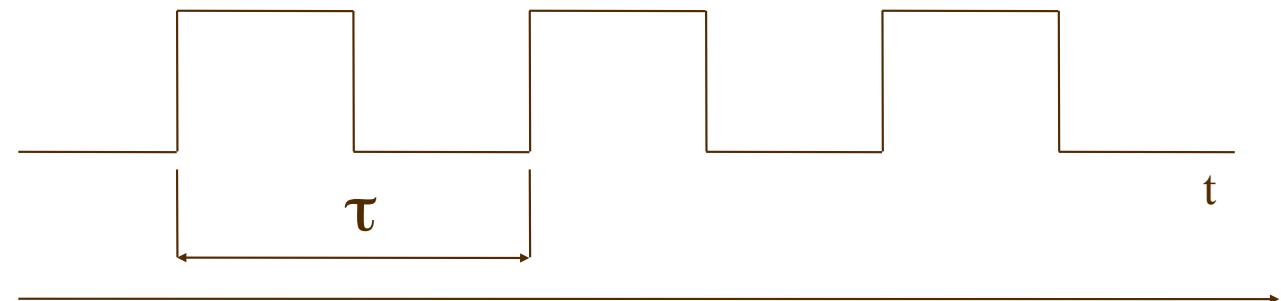


L'unità centrale di elaborazione

- Dal punto di vista logico, la CPU è costituita da tre parti principali
 - l'**unità logico-aritmetica (ALU)**, che esegue operazioni aritmetiche (addizioni, sottrazioni, moltiplicazioni, divisioni) e logiche.
 - l'unità di controllo, che ne governa il funzionamento
 - un insieme di **registri**, che sono spazi ad accesso molto veloce per la memorizzazione temporanea di dati e istruzioni da eseguire
- Il funzionamento della CPU è **ciclico** e il periodo di tale ciclo viene scandito dall'orologio di sistema (**clock**), che produce un segnale periodico

Il segnale di clock nella CPU

- La **frequenza** del clock, costituisce una delle caratteristiche tecniche più importanti della CPU
- si misura in **cicli al secondo** o **Hz (Hertz)**
 - Ad esempio: **f = 3 GHz** (gigaHz), $3 \times 10^9 = 3$ miliardi di cicli al secondo



$f = 1/\tau$ τ si dice periodo
segnale periodico: $f(t) = f(t + \tau)$

Ciclo di funzionamento della CPU

- Cicli di funzionamento sono composti da tre fasi
 - **accesso (fetch)**: lettura dell'istruzione da eseguire e sua memorizzazione nel **registro istruzione**
 - **decodifica (decode)**: decodifica dell'istruzione da eseguire
 - **esecuzione (execute)**: esecuzione dell'istruzione
- Ciclo **fetch-decode-execute**
- La posizione dell'istruzione a cui si accede durante la fase di *fetch* è contenuta in un registro speciale detto *program counter* (PC)
 - viene incrementato di un'unità a ogni ciclo, in modo da **eseguire istruzioni in sequenza**

Incrementare le prestazioni: Parallelismo

- La **frequenza di clock** influenza direttamente il tempo di esecuzione delle istruzioni ed è limitata dalla tecnologia disponibile.
- Il **parallelismo** permette di migliorare le prestazioni senza modificare la frequenza di clock. Esistono due forme di parallelismo:
 - **parallelismo a livello delle istruzioni** (architetture pipeline e superscalari)
 - **parallelismo a livello di processori** (multiprocessori e multicompiler).

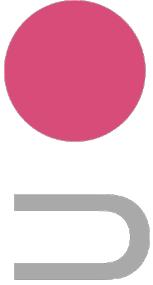
Architettura pipeline

Consiste nell'organizzare la CPU come una “catena di montaggio”

- la CPU viene suddivisa in **stadi**, ognuno dedicato all'esecuzione di un compito specifico
- l'esecuzione di una istruzione richiede il **passaggio attraverso** tutti o alcuni degli **stadi della pipeline**
- in un certo istante, ogni stadio esegue la parte di istruzione di sua “competenza”
- in un certo istante esistono diverse **istruzioni contemporaneamente in esecuzione**, una per stadio.

Un esempio: pipeline a 5 stadi

- S1. **lettura istruzioni** dalla memoria e loro caricamento in un apposito buffer
- S2. **decodifica** dell'istruzione per determinarne il tipo e gli operandi richiesti
- S3. individuazione e **recupero** degli operandi dai registri o dalla memoria
- S4. **esecuzione** dell'istruzione
- S5. **invio dei risultati** all'apposito registro.



Multiprocessori e multicalcolatori

- Nei **Multiprocessori** diverse CPU condividono una **memoria comune**:
 - le **CPU devono** coordinarsi per accedere alla memoria
 - esistono diversi schemi di collegamento tra CPU e memoria; quello più semplice prevede ci sia un **bus condiviso**.
- Nei **Multicalcolatori** si utilizzano più calcolatori, ognuno dei quali dotato di una **memoria privata**:
 - la comunicazione tra CPU è basata su **scambio di messaggi**
 - si è arrivati a costruire multicalcolatori con ~10000 CPU.

La memoria del computer

- Ci sono due tipi di memoria: **primaria** e **secondaria**
- Entrambe servono a **immagazzinare dati** e **programmi** all'interno del computer, con ruoli diversi
- La memoria è suddivisa in **celle** o locazioni di memoria, ognuna delle quali ha un **indirizzo**
- Ogni cella contiene un numero predefinito di **bit**
 - un **bit** è un dato elementare che può assumere due valori, *convenzionalmente* chiamati **zero** e **uno**
 - un insieme di otto bit si chiama **byte** ed è di norma l'unità di misura della memoria (ad es. 2 Gbyte), nonchè l'unità minima di memoria accessibile singolarmente.

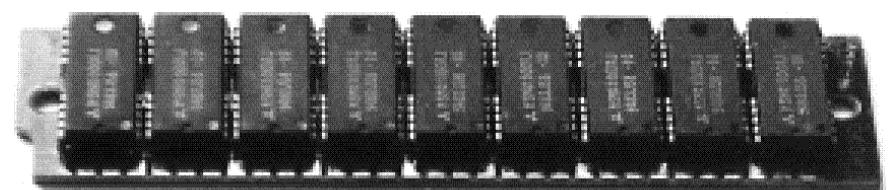
Bit, Unità di Misura e Byte

- La parola bit può avere diversi significati:
 - Il bit è l'**unità di misura dell'informazione**
 - **bit = binary digit**, una cifra in un sistema di numerazione in base 2 ... (eg, numero binario di 32 bit)
 - **bit = dispositivo bistabile** che memorizza un simbolo di un alfabeto binario... (eg, un registro di 12 bit)
- Unità di misura della memoria
 - Nel Sistema Internazionale (SI) e nel senso comune
 - $k=10^3=1000$, $M=10^6=1000000$, $G=10^9=1000000000$
 - Per IEC/ISO (International Electrotechnical Committee)
 $ki \text{ (kibi)} = 2^{10} = 1024$, $Mi \text{ (Mebi)} = 2^{20} = 1\ 048\ 576$
 $Gi \text{ (Gibi)} = 2^{30} = 1\ 073\ 741\ 824$
 - Quindi: 1 Gbyte = 10^9 byte, Gibyte = 2^{30} byte



La memoria primaria (o centrale)

- La memoria **primaria** è **veloce** ma **costosa**
- È costituita da **chip di memoria** realizzati con la stessa tecnologia (al silicio) utilizzata per la CPU
 - memoria **di sola lettura** (**ROM**, *Read-Only Memory*)
 - memoria ad accesso casuale (**RAM**, *Random Access Memory*)
 - dovrebbe chiamarsi memoria **di lettura e scrittura**, perché in realtà anche la ROM è ad accesso casuale, mentre ciò che le distingue è la possibilità di scrivervi
 - **accesso casuale** significa che **il tempo per accedere ad un dato non dipende dalla sua posizione nella memoria**



Spazio di indirizzamento

- Il processore può
 - **Scrivere** un numero in una cella di memoria
 - **Leggere** un numero da una cella di memoria
- In entrambi i casi deve specificare l'**indirizzo** della cella
 - L'indirizzo è un numero → sequenza di bit.
 - Per ragioni di rapidità il processore deve poterlo comunicare in “un colpo solo” alla memoria → l'indirizzo deve avere una **lunghezza prefissata**
 - Limitando tale lunghezza, limito anche il numero di indirizzi che posso esprimere.
 - Es.: indirizzo 16 bit → 2^{16} celle = 64 MB
 - Es.: indirizzo 32 bit → 2^{32} celle = 4 GB
 - Il numero di celle indirizzabili è detto **spazio di indirizzamento**
 - Alcuni PC usano 32 bit per gli indirizzi
 - Max 4 GB di memoria
 - Computer usano già da tempo 64 bit

La memoria ROM

- La memoria di sola lettura, **ROM**
 - conserva i dati ed i programmi in essa memorizzati anche quando il computer viene spento
 - è una memoria **non volatile**
 - Viene scritta una volta sola direttamente alla fabbricazione (distribuzione di firmware)
 - contiene i programmi necessari all'avvio del computer, programmi che devono essere **sempre disponibili**
 - nei PC, tali programmi prendono il nome di **BIOS** (Basic Input/Output System)
- Alla categoria delle ROM appartengono anche
 - PROM (Programmable ROM) con dispositivo speciale
 - EPROM (Erasable PROM) con raggi ultravioletti
 - EEPROM (Electrical EPROM) multiple scritture

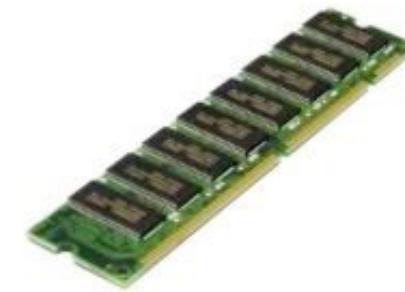
La memoria RAM

- La memoria ad accesso casuale, **RAM**
 - è una memoria che consente la **lettura** e la **scrittura** dei dati e dei programmi in essa contenuti
 - contiene dati in fase di modifica e programmi che non devono essere sempre disponibili
 - perde i dati quando si spegne il computer
 - è una **memoria volatile**
- Esistono vari tipi di RAM, che differiscono per tecnologie costruttive, costi, prestazioni, ...
 - SRAM – Static RAM
 - DRAM – Dynamic RAM
 - SDRAM – Syncronous DRAM
 - DDR1-DDR5 - Double Data Rate

Il principio di località: la cache

- **Località spaziale:** “quando si accede all’indirizzo A, è *molto probabile* che gli accessi successivi richiedano **celle vicine ad A.**”
- **Località temporale:** “quando si accede all’indirizzo A, è *molto probabile* che gli accessi richiedano **di nuovo** la cella A.”
- Si utilizza quindi una memoria che consenta accessi estremamente veloci su blocchi utilizzati di recente. Questa memoria è la **cache**: veloce ma molto costosa, quindi piccola.

Packaging

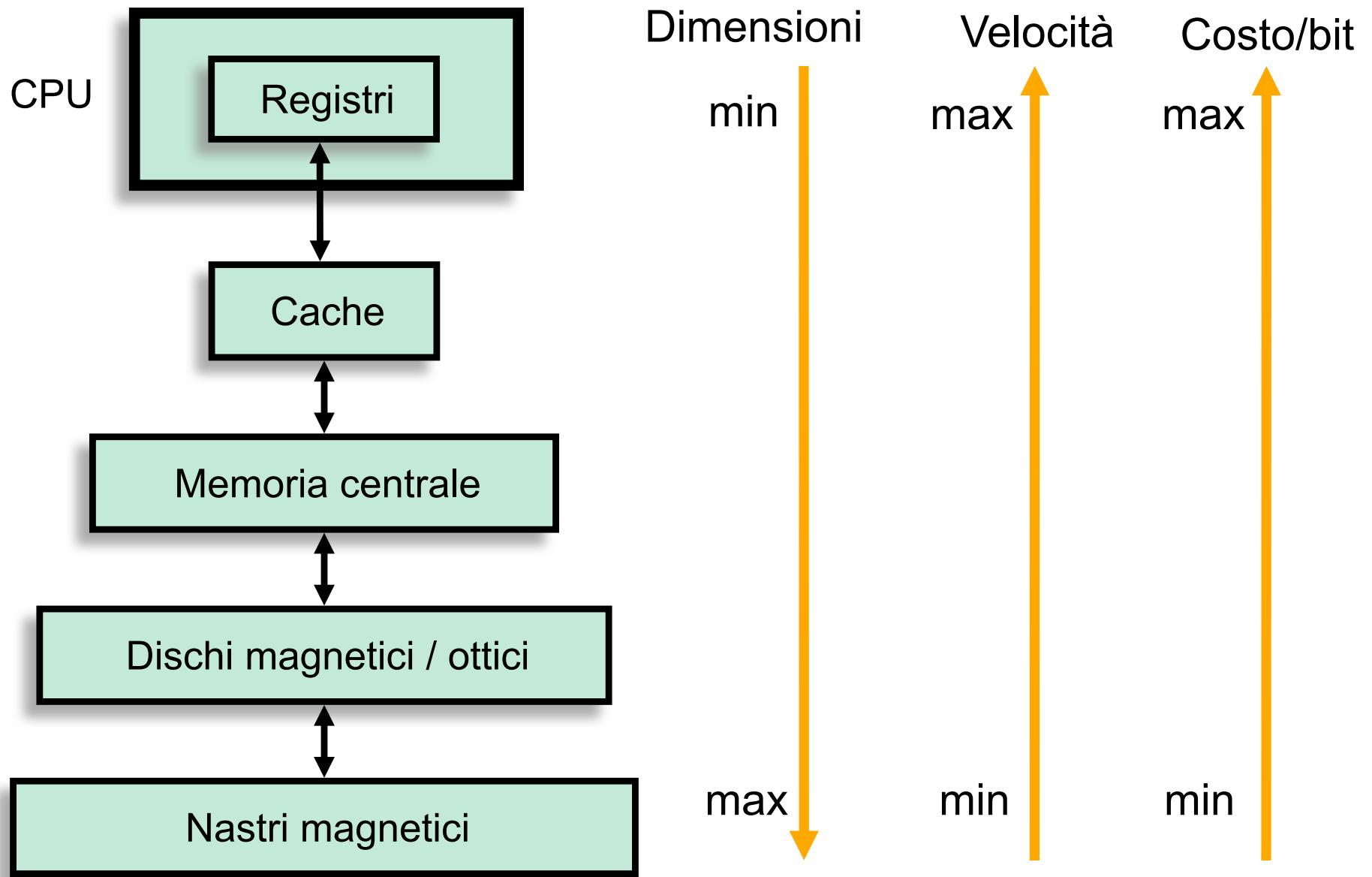


- Fino all'inizio degli anni '90 la memoria veniva prodotta, acquistata ed installata su chip singoli
- Oggi si monta un gruppo di chip, tipicamente 8 o 16, su una piccola scheda stampata
 - SIMM (Single Inline Memory Module): la fila di connettori si trova da un solo lato della scheda
 - DIMM (Dual Inline Memory Module): i connettori si trovano su ambedue i lati della scheda
- SIMM e DIMM sono spesso dotate di un codice di rilevazione o correzione degli errori.

La memoria secondaria (o di massa)

- La memoria secondaria
 - è un supporto di memoria **non volatile**
 - è circa 100 volte **meno costosa** della memoria primaria (circa $0.2 \cdot 10^{-9}$ €/byte contro $25 \cdot 10^{-9}$ €/byte)
 - è molto **più lenta** (tempi di accesso in lettura/scrittura) della memoria primaria
- Programmi e dati risiedono nella memoria secondaria e vengono caricati nella RAM quando necessario, per poi tornarvi aggiornati se e quando necessario
- È di solito un **disco rigido** (o disco fisso, **hard disk**)
 - Formato da piatti rotanti rivestiti di materiale magnetico, con testine di lettura/scrittura
 - Attuali dimensioni: **Terabyte** (tebybite, ovvero 2^{40} byte)

Gerarchia di memorie



Dischi magnetici

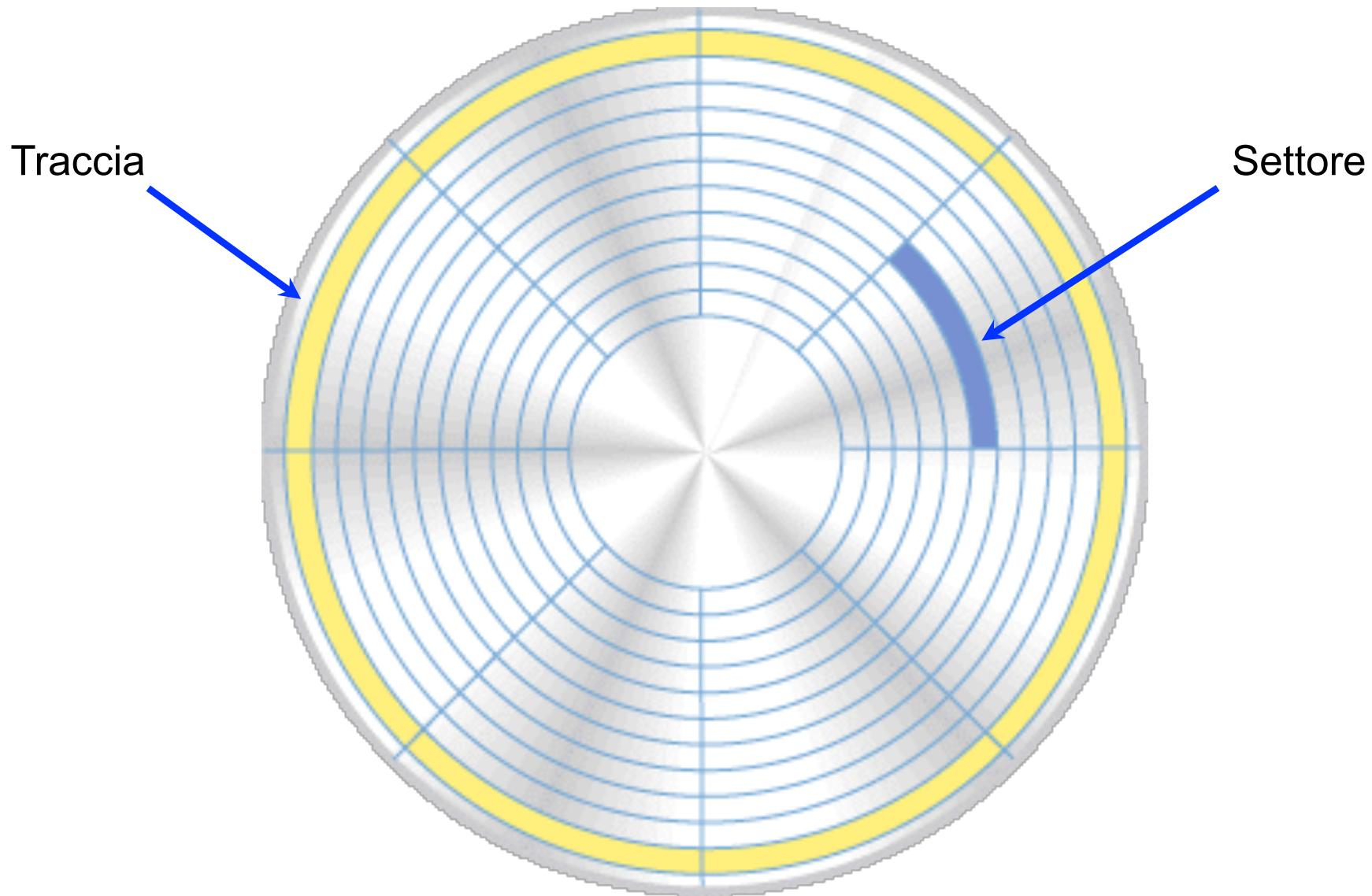


- Sono **piatti**, generalmente di alluminio, ricoperti di materiale **ferromagnetico**
- **Fattore di forma** (diametro)
 - sempre più piccolo, consente velocità di rotazione maggiori
 - 3.5 pollici per sistemi desktop e fino ad 1 pollice per i portatili.
- **Testina** (strumento di lettura/scrittura)
 - è sospesa appena sopra la superficie magnetica
 - **scrittura**: il passaggio di corrente attraverso la testina magnetizza la superficie
 - **lettura**: il passaggio sopra un'area magnetizzata induce una corrente nella testina.

Tracce e settori

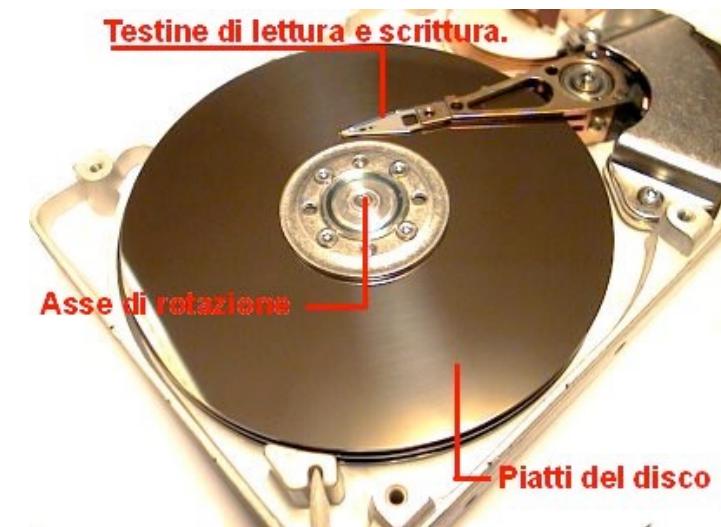
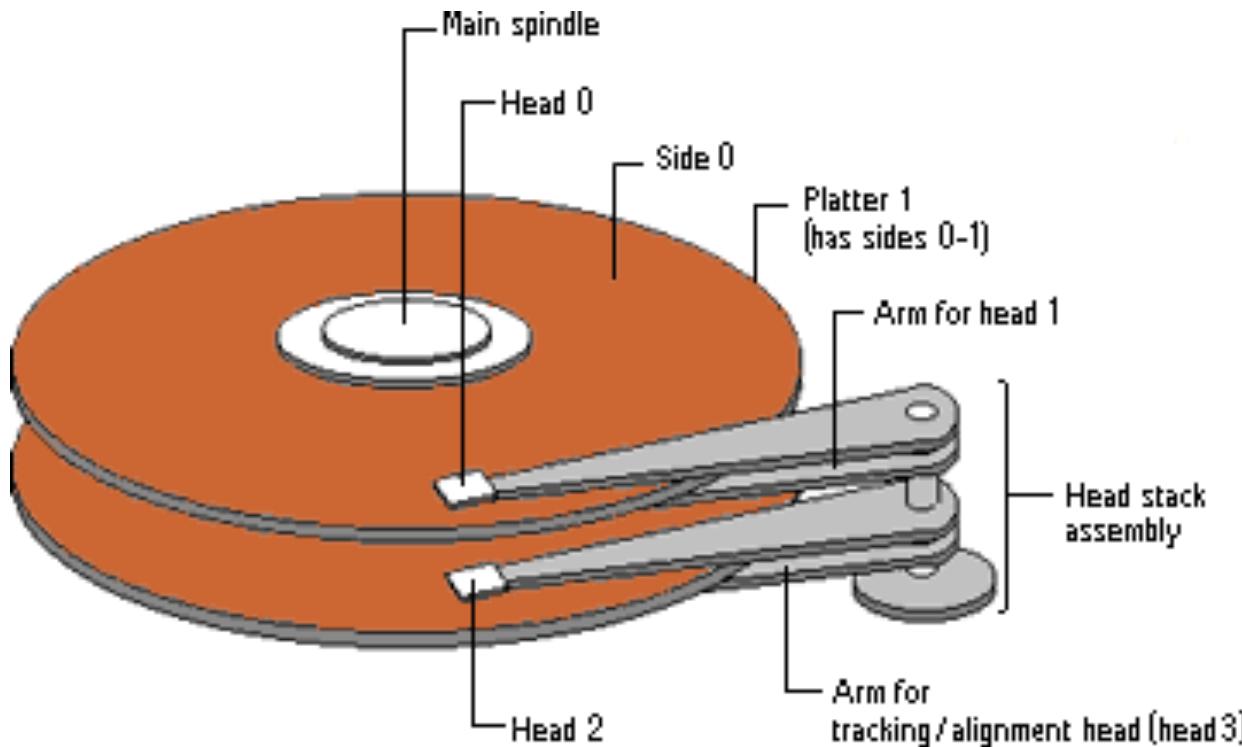
- **Traccia (track)**: sequenza circolare di bit scritta mentre il disco compie una rotazione completa. Tra una traccia e l'altra c'è un piccolo spazio separatore (**gap**).
- **Settore (sector)**: parte di una traccia corrispondente ad un *settore circolare* del disco
 - un settore contiene 512 byte di dati, preceduti da un *preamble* e seguiti da un codice di correzione degli errori
- Altre sottostruuture sono i **cluster** e i **blocchi**.
- **Formattazione**: operazione che predisponde tracce e settori per la lettura/scrittura
 - circa il 15% dello spazio si perde in gap, preamboli e codici di correzione errori.

Tracce e settori

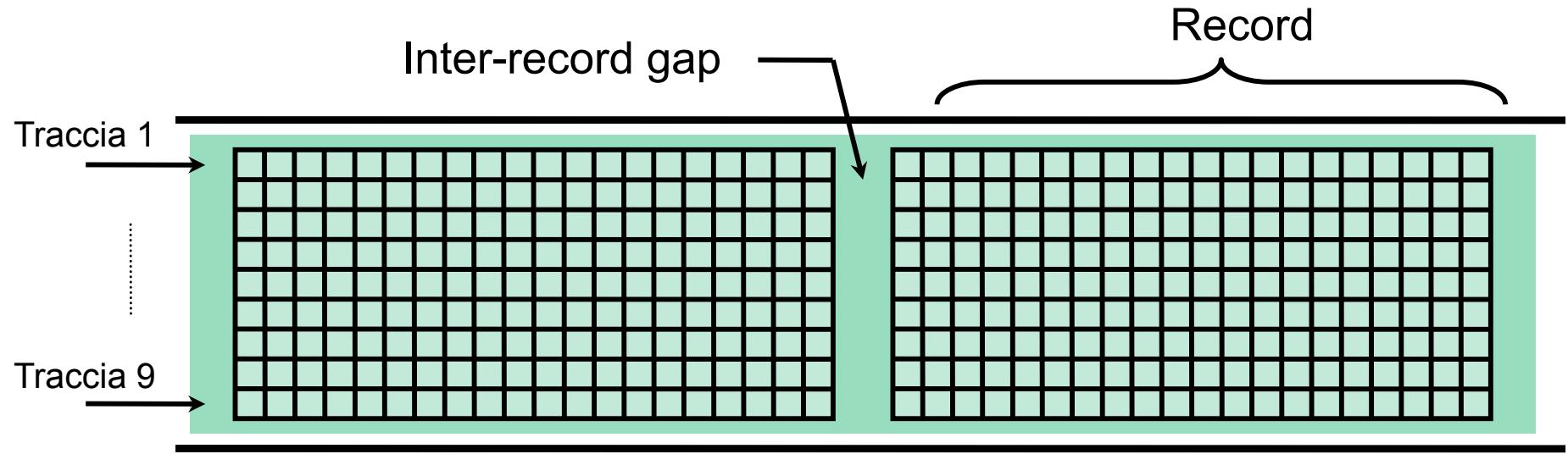


Hard Disk

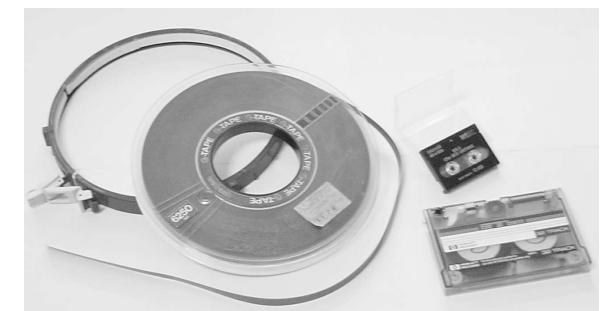
L'Hard Disk è composto da più superfici (piatti) e da una testina di lettura per superficie.



Nastri magnetici (LTO, DLT)



- Capacità di diversi TB
- Accesso sequenziale ai dati
- Molto lenti
- Utili solo per operazioni di backup



Floppy disk



- **Funzioni:** distribuzione del software su larga scala (con l'avvento dei PC) e archiviazione dati
- Struttura analoga a quella di un disco magnetico
 - il disco si **ferma** quando non è operativo
 - l'**avvio della rotazione** comporta un **ritardo** di $\frac{1}{2}$ secondo
- Caratteristiche tipiche di un floppy da 3.5":
 - capacità: **1.44 MB**
 - tracce × settori: **80 × 18**
 - giri per minuto (RPM): **300**
 - velocità di trasferimento: **500 Kbps.**

TODAY I SHOWED MY NEPHEW
AN OLD 1.4MB FLOPPY DISC,
AND HE SAID:



„COOL!!! SOMEONE 3D
PRINTED THE SAVE ICON!“

Dischi ottici



- Lettura ottica basata sulla riflessione (o sulla mancata riflessione) di un raggio laser sul supporto
- Densità di registrazione più alta che nei dischi magnetici
- Creati in origine per registrare programmi televisivi, successivamente applicati ai calcolatori.
- Tipologie:
 - CD-ROM, CD-R, CD-RW, DVD, DVD-RAM, HD-DVD, Blu-Ray...

Compact Disk (CD)

- Supporto proposto nel 1980 da Philips e Sony per sostituire i dischi musicali in vinile
- Standard di fabbricazione IS-10149:
 - diametro: **12 cm**; spessore: 1.2 mm e foro di 15 mm al centro
 - produzione:
 - laser ad alta potenza che brucia fori di 0.8 µm in un **disco master** (le depressioni sono dette **pit** e le aree tra i pit sono dette **land**)
 - dal master si ricava uno stampo
 - nello stampo viene iniettata una resina liquida di **policarbonato** che forma un CD con la stessa sequenza di pit del master
 - sul policarbonato viene depositato uno strato molto sottile di **alluminio riflettente**
 - copertura con strato protettivo e poi con una etichetta.

Lettura di un CD

- Un **laser a bassa potenza** emette una luce infrarossa sul disco
- I **pit** appaiono come **cunette** su una superficie piatta
- I passaggi **pit/land** o **land/pit** indicano un **1**, e la loro assenza indica uno **0**
- Pit e land sono impressi in una **spirale unica** che compie 22.188 giri attorno al disco
- La lettura viene effettuata a **velocità costante** (120 cm/sec), molto meno della velocità di lettura degli **hard disk**.

La memoria secondaria

▪ **Memoria flash** in rapidissima diffusione

- Permanente e riscrivibile (EEPROM), organizzata a blocchi
- Priva di parti mobili, resistente a sollecitazioni ed urti,
- Leggera e piccola, indicata per la trasportabilità (fotocamere digitali, lettori di musica, cellulari, ecc.)
- Sostituto per hard disk di computer portatili
- Controindicazioni
 - Costi alti, anche se in costante diminuzione con il progredire della tecnologia
 - Numero di scritture possibile è altissimo ma limitato

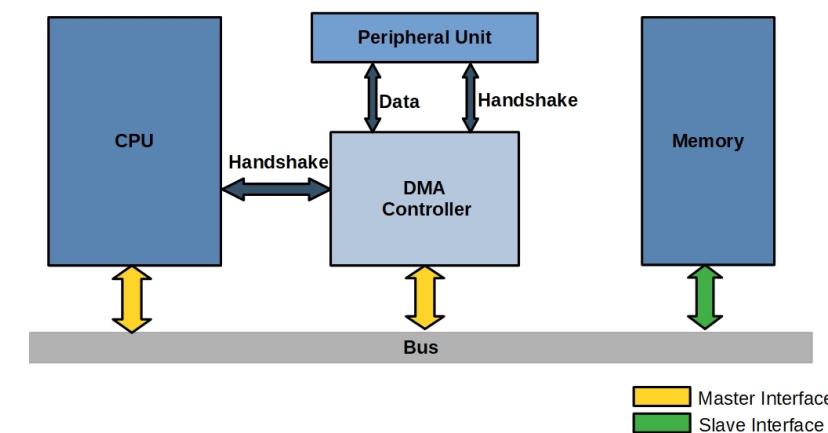


Dispositivi di ingresso/uscita

- L'interazione fra l'utente e il calcolatore avviene mediante **dispositivi periferici di Input/Output (I/O)**
 - Tradizionali dispositivi di input sono la **tastiera**, il **mouse** (dispositivo di puntamento), il **microfono** (per impartire comandi vocali), il **joystick**, lo **scanner** (per la digitalizzazione di documenti e di immagini)
 - Tradizionali dispositivi di output sono lo **schermo** (monitor), le **stampanti**, gli **altoparlanti**

Input / Output

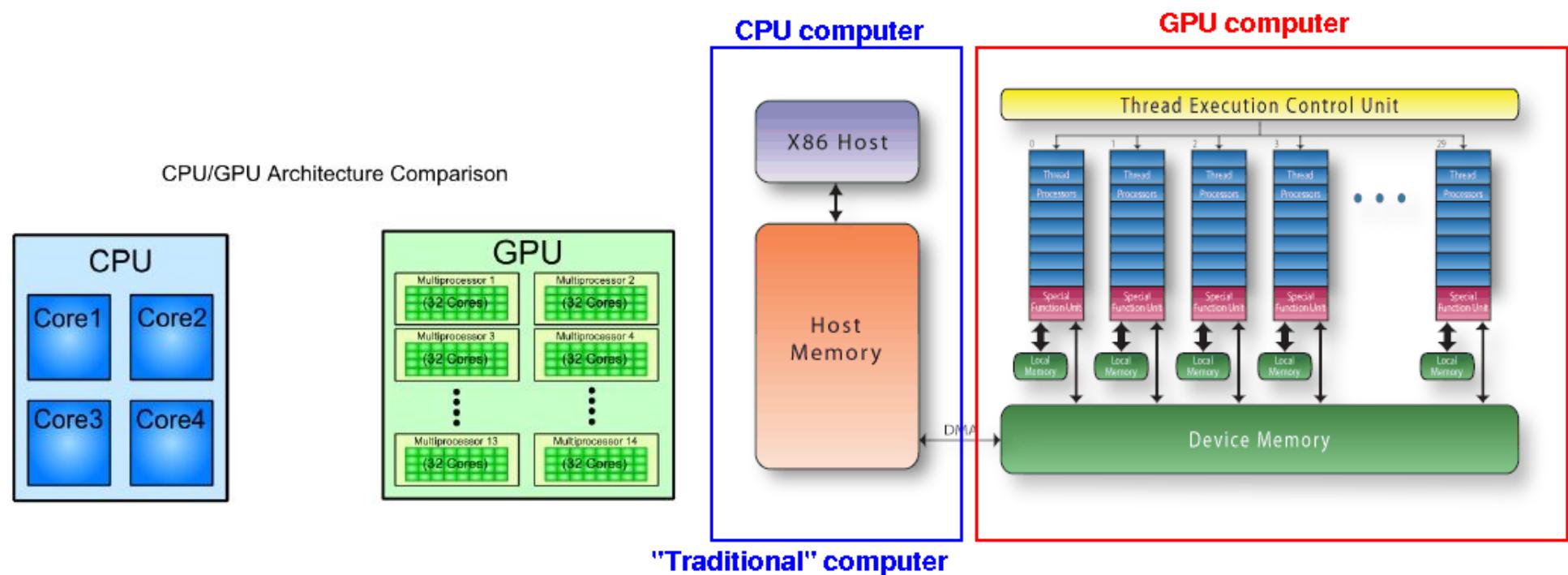
- Le operazioni di I/O si possono effettuare in **3 modalità**:
 - Controllo da programma**: si utilizzano cicli di codice (*polling*) che esaminano periodicamente lo stato dell'interfaccia hardware su cui leggere/scrivere (i.e. una stampante), fino a completamento dell'operazione.
 - Interrupt**: il dispositivo periferico è in grado di notificare alla CPU il completamento di una operazione “richiamando la sua attenzione” attraverso una *interruzione* del flusso di esecuzione.
 - Direct Memory Access (DMA)**: uno specifico componente hardware (*DMA controller*), solleva la CPU dalla necessità di intervenire direttamente al trasferimento di ogni dato in I/O. La CPU comunica non con la periferica ma col DMA controller imponendogli l'inizio dell'operazione e comunicandogli l'indirizzo di memoria in cui il dato da trasferire è immagazzinato.



Input / Output

- CUDA (Compute Unified Device Architecture) è un'architettura hardware per l'elaborazione parallela creata da NVIDIA. Tramite l'ambiente di sviluppo per CUDA: scrivere applicazioni capaci di eseguire calcolo parallelo sulle GPU. I linguaggi di programmazione disponibili nell'ambiente di sviluppo CUDA sono estensioni dei linguaggi più diffusi: C, Python, Fortran, Java e MATLAB.

Linear algebra; Signal and image processing (FFTs); Neural networks and deep learning



Collegamento: le porte

- Collegamento **fisico** tramite le **porte** (anche dette **interfacce**)
 - Le porte sono le “prese” a cui si collegano i dispositivi
 - Sono attaccate alla scheda madre
 - Alcune sono **esterne** (visibili al di fuori del case), alcune sono **interne**
 - **Periferiche esterne** si possono attaccare/staccare, maneggiabili dall’utente
 - **Periferiche interne** sono pensate per non essere scollegate; si possono però sostituite con relativa facilità
 - Diversi tipi di porte; si differenziano per:
 - **Quantità** di informazioni che lasciano passare (numero di bit)
 - **Velocità** di trasferimento delle informazioni
 - Alcune porte permettono anche di **alimentare** i dispositivi

Porte Standard e USB



- **Interfaccia Seriale:** trasporta **1 bit** per volta, ha velocità massima = 115 Kbps e si usa per periferiche lente, come mouse e modem esterni.
- **Interfaccia parallela:** trasporta **8 bit** alla volta, ha velocità massima = 150 KB/sec e si usa(va) per stampanti, scanner e unità di backup (nastri, Zip).
- **Universal Serial Bus (USB):** definito con l'intento di sostituire le attuali porte seriali e parallele
 - velocità = **12 Mbit/sec**, collega fino a **127** periferiche in **cascata**
 - **alimenta** direttamente periferiche a basso consumo (tastiere, mouse) ed è completamente (?) **plug & play**
 - **USB 2.0** del 1999 arriva fino a **480 Mbps**
 - **USB 3.0** - 4,8 Gbit/s - Settembre 2007
 - **USB 3.1** 10 Gbit/s - gennaio 2013 (**USB-C**, 2014)
 - **USB 3.2** 20 Gbit/s - luglio 2017
 - **USB 4** 40 Gbit/s - agosto 2019
 - **Wireless USB** - onde radio

Tipi di porte



- Ciascun tipo di porta ha una **presa** riconoscibile a cui corrisponde uno specifico **connettore**.

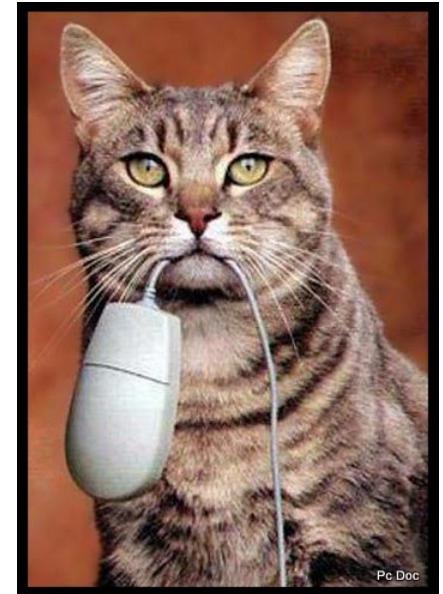
In figura:

- 6 USB 2.0
- 1 Firewire
- Gigabit LAN
- eSATA
- PS/2 (tastiera)



- eSATA e SATA

Mouse



Pc Doc

- È una periferica di tipo “point and click”
- Nasce nel 1964 per poter utilizzare la prima interfaccia grafica (“a finestre”).
- Negli anni si è evoluto, sostituendo la classica “pallina” con un dispositivo ottico (optical mouse) che permette un utilizzo più versatile e perdendo la “coda” (cordless mouse).



Input: il mouse



- Manda diversi tipi di segnali al processore:
 - **Pressione** di un tasto
 - **Rilascio** di un tasto
 - **Spostamento** rispetto alla posizione precedente
- Questa informazione viene elaborata e tradotta:
 - in un corrispondente **movimento del puntatore** sullo schermo per gli eventi di **spostamento**
 - in un **segnale di stato** in caso di **pressione** e/o **rilascio**
 - Alcune combinazioni di eventi acquisiscono significati particolari:
 - Ad es. pressione+rilascio+pressione+rilascio, se eseguito velocemente, corrisponde ad un **“doppio click”**
 - La trasformazione dalla combinazione iniziale all’evento composto viene operata dal software

Dispositivi di puntamento

- **Mouse meccanici/ottici**
 - Il classico mouse a forma di topo
 - Il principio è lo stesso, cambia la tecnologia di registrazione dello spostamento
 - mouse ottico più preciso e meno deteriorabile
- **Trackball**
 - Invece che spostare il mouse si fa ruotare una pallina girevole
- **Touchpad**
 - Per comunicare lo spostamento si muove il dito su una superficie sensibile
- **Airmouse** —————→
- **Pointing stick (o TrackPoint)**
 - Simile ad un minuscolo joystick



Input: la tastiera



- La pressione di un tasto genera un **interrupt** attraverso cui viene attivato il gestore degli interrupt della tastiera che decodifica il tasto premuto. Al rilascio del tasto di verifica un secondo interrupt.
- Questa informazione viene elaborata e trasformata:
 - nel **carattere** corrispondente se si tratta di un tasto-carattere (anche in base allo **stato** corrente)
 - in un **segnale di stato** se si tratta di un tasto con funzione particolare (es. SHIFT, CTRL, frecce, ecc.)
- Molte tastiere moderne contengono tasti associati direttamente a funzioni di alto livello
- Tastiere diverse; principali distinzioni:
 - Ordine delle lettere (Italia/USA, QWERTY)
 - Numero di tasti (numero tipico: 108)

Input: lo scanner



- Immagine su carta
- Immagine digitale
- Produce una **bitmap**
 - anche se il foglio contiene del testo, lo scanner fa una “foto” e produce un’immagine
- Parametro importante: **risoluzione di scansione**
 - come per la stampante si misura in **DPI**, numero di punti che lo scanner legge per ciascun pollice
 - ciascun punto viene poi convertito in un pixel
 - La risoluzione di scansione influenza la risoluzione digitale dell’immagine
 - Es.: immagine di 10x10cm, scansione a 300 DPI
 - 10 cm sono circa 4 pollici, quindi 1200x1200 punti
 - 1200x1200 punti ->1200x1200 pixel.

Output: il monitor



- Il monitor **mostra un'immagine** che viene elaborata dal computer
- L'immagine viene **aggiornata** spesso -> appare animata
- Parametri importanti nella scelta di un monitor:
 - **Risoluzione** massima dell'immagine mostrata
 - **Profondità** del colore massima dell'immagine mostrata
 - **Frequenza** di aggiornamento dell'immagine
 - **Dimensione** fisica del monitor
 - **Tipologia** di monitor

Monitor



■ CRT (Cathode Ray Tube):

- un “cannone” spara un raggio di elettroni contro uno **schermo fosforescente**; il raggio viene **deflesso** per coprire tutti i punti dello schermo, una riga per volta
- un’immagine a tutto schermo viene completata **30/60/100 volte al secondo (refresh rate)**

■ LCD/TFT (Liquid Crystal Display/Thin Film Transistor):

- sono schermi piatti e leggeri, composti di **due lastre parallele di vetro** nella cui intercapedine sono contenuti i **cristalli liquidi**, illuminati da una luce situata dietro lo schermo
- i cristalli liquidi scorrono come un liquido e hanno una struttura tridimensionale
- un **campo elettrico** modifica l’allineamento molecolare dei cristalli e quindi le **proprietà ottiche**.

Risoluzione e Profondità

- La CPU scrive l'immagine da visualizzare in una **memoria speciale** associata al video
 - Video-RAM (VRAM)
 - L'immagine è **codificata**
 - Il monitor visualizza l'immagine contenuta nella VRAM
- **Risoluzione** e **profondità** massime
 - determinate dalla **dimensione della memoria video**
 - più memoria video = maggiore risoluzione / profondità
 - maggiore risoluzione, scrivania più grande!
- In parte risoluzione e profondità dipendono anche dalle capacità fisiche del video (dimensioni e possibilità di visualizzare più colori)

Stampanti B&W - 1

Stampanti ad aghi



- **Funzionamento:** la testina di stampa contiene fino a 24 aghi, e ogni ago è azionato da un'elettrocalamita; mentre la testina si muove, l'azione combinata degli aghi compone i caratteri
- **Caratteristiche:** sono **economiche** e **affidabili**, ma **lente**, **rumorose** e con **grafica di bassa qualità**.

Stampanti a getto d'inchiostro



- **Funzionamento:** al posto degli aghi ci sono **ugelli** collegati a serbatoi di inchiostro di diversi colori; mentre la testina si muove, gli ugelli spruzzano gocce di inchiostro per comporre i caratteri
- **Caratteristiche:** risoluzioni fino a 1440 dpi (dots per inch); **economiche**, **silenziose**, di **buona qualità** ma sono **lente** e le cartucce sono **costose**.

Stampanti B&W - 2



Stampanti laser

- Un raggio laser infrarosso viene modulato dalla sequenza di pixel che deve essere impressa sul foglio.
- Viene deflesso da uno specchio rotante su un tamburo fotosensibile elettrizzato che si scarica dove colpito dalla luce.
- L'elettricità statica attira una fine polvere di materiali sintetici e pigmenti, il toner, che viene trasferito sulla carta.
- Il foglio passa sotto un rullo fusore riscaldato ad elevata temperatura, che fonde il toner facendolo aderire alla carta.
- Caratteristiche: alta qualità e flessibilità, buona velocità e costi contenuti; utilizzano una tecnologia simile a quella delle fotocopiatrici.

Stampanti a colori

Stampanti a getto d'inchiostro

- Caratteristiche: permettono buoni risultati per la grafica a colori ma risultati mediocri per le fotografie. Per risultati migliori si usano inchiostri a base asciutta e inchiostri a base di pigmenti.



Stampanti a inchiostro solido

- Caratteristiche: si utilizza un inchiostro speciale a base di cera in quattro blocchi solidi, ma è richiesto un lungo tempo di avviamento per sciogliere l'inchiostro.

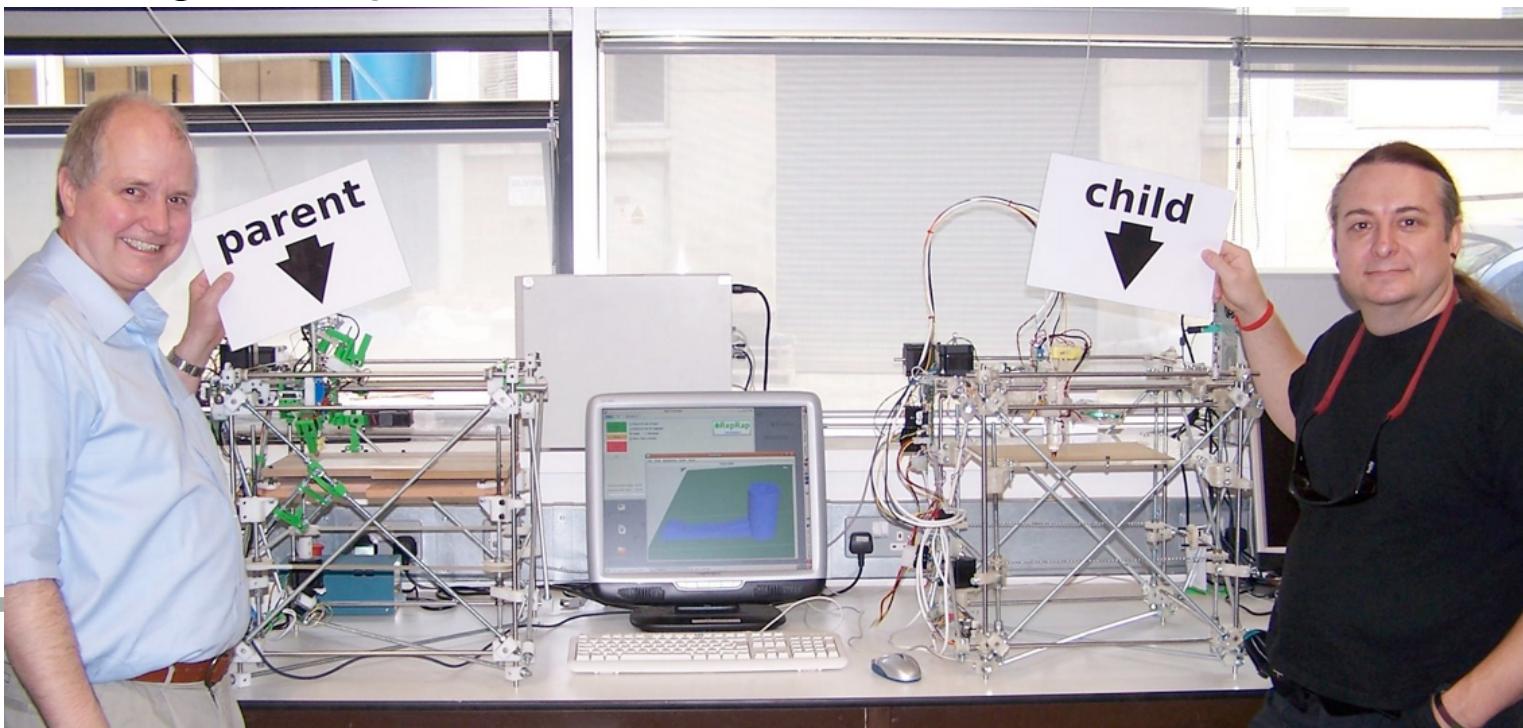
Stampanti laser a colori

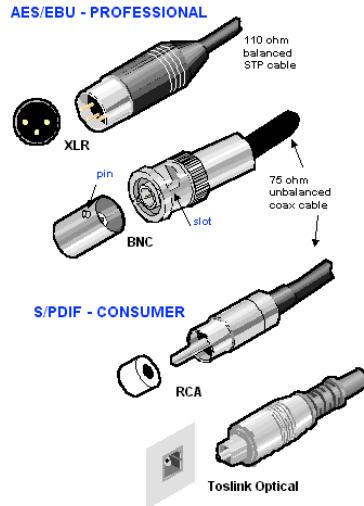
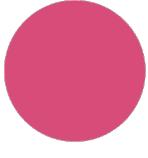
- Caratteristiche: funzionano come il modello monocromatico, ma generano quattro immagini, una per ogni colore del toner, C, Y, M, K. Costose, veloci, alta qualità e producono immagini stabili nel tempo.

Stampante 3D

- RepRap è una macchina autoreplicante, può stampare i pezzi necessari per assemblare una nuova stampante.
- Possiamo stimare che una famiglia che usi una RepRap per produrre 20 oggetti domestici ogni anno (ovvero circa lo 0,02% di tutti i prodotti fabbricabili potenzialmente con una di queste macchine, se usata 24h/24 7g/7) risparmi una somma che varia dai 300 ai 2000 dollari americani:
- "...la conclusione di questo studio è che una RepRap è già un investimento economicamente attraente per la famiglia americana media di oggi."

Fonte: B.T. Wittbrodt et al., Life-cycle economic analysis of distributed manufacturing with open-source 3-D





Output: scheda audio



- È una periferica **interna** o **esterna**
- Ha il compito di trasformare informazioni digitali in **suoni**
- Esternamente mostra solo il jack per inserire le cuffie o gli altoparlanti
 - Uscita di linea, uscita amplificata, ottica, S/PDIF, BNC, MADI, MIDI
 - Può essere dotata di altoparlanti, solitamente di scarsa qualità
 - Alcuni monitor contengono anche altoparlanti per cui è possibile collegarli anche all'uscita della scheda audio

Dispositivi di ingresso/uscita

- Importante dispositivo di I/O: connessione a una **rete** per scambiare dati con sistemi remoti
 - può essere diretta (ad alta velocità), tramite una rete locale, o remota (a velocità più bassa), tramite una linea telefonica e un modem, via radio (senza fili o wireless)



- Nuovi dispositivi di input/output “**multimodale**”
 - Touchscreen, accelerometri, sensori di luce, giroscopi, output tattile (buzz), ...

- La scheda madre (**motherboard**) è la base su cui sono montati **CPU** e **RAM**, collegati fra loro dai **bus**.

La scheda madre

A BALANCE OF FEATURES

The APPLE-1 SYSTEM is a **fully assembled, tested & burned-in** microprocessor board using the 6502 microprocessor. The board contains processor & support hardware; **complete video electronics** for a 40 character/line, 24 line video display; **on-board RAM capacity of 8K BYTES**; software system monitor in PROM; and fully regulated power supplies. The Apple attaches directly to an ASCII encoded keyboard and a video monitor, allowing the efficient entry and examination of programs in hexadecimal notation. The use of the new **16-pin 4K RAM chips** results in low power and high density memory, which can be upgraded to the 16K chips when they become available (32K bytes on-board RAM!!)

A fast (1 kilobaud) cassette interface is available and includes a tape of **Apple Basic**. And ... Yes, Folks, **Apple Basic is Free!**

APPLE COMPUTER CO.

APPLE-1 \$666.66
*includes 4K bytes RAM

Micro Interface	<ul style="list-style-type: none"> • 6502 Microprocessor • Full video display electronics - 40 char/line, 24 line. • Outputs composite video. • Has ASCII keyboard interface on-board. • Cassette interface board available. FAST - 1 Kilobaud.
Memory	<ul style="list-style-type: none"> • Uses 16-pin 4K Dynamic RAMS. • 8K BYTE RAM capacity on-board! • Upgradable to 16K RAM chips. • Software system monitor in PROM
Basic	<ul style="list-style-type: none"> • Apple Basic ... pseudo-compiled, FAST, FREE.
Power	<ul style="list-style-type: none"> • Fully regulated power supplies on-board.

DEALER INQUIRIES INVITED

APPLE COMPUTER COMPANY

770 Welch Road, Suite 154
Palo Alto, California 94304
Phone: (415) 326-4248

CIRCLE NO. 42 ON INQUIRY CARD

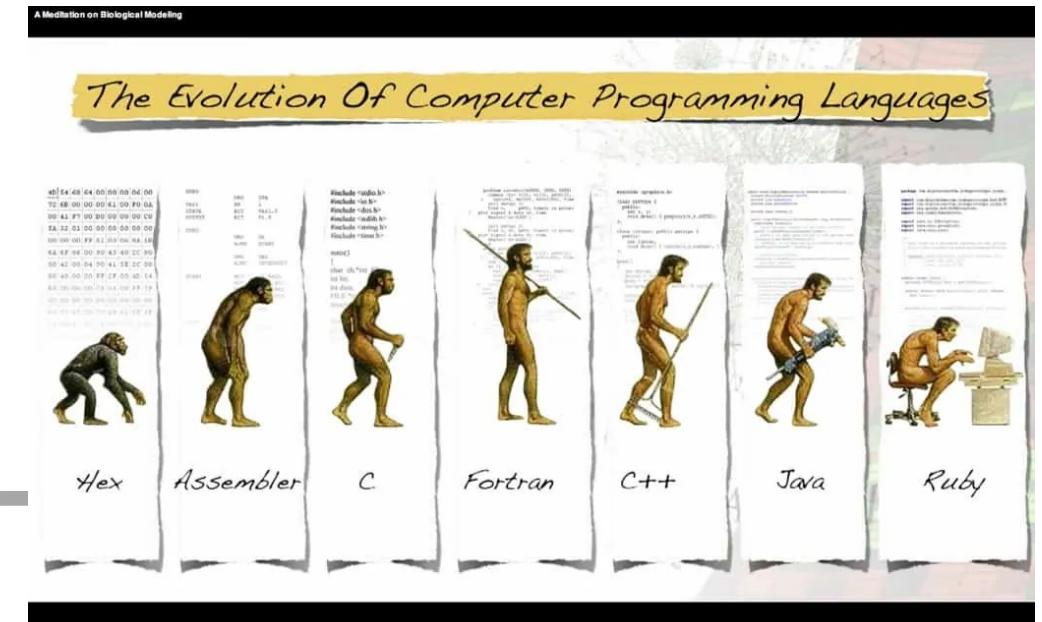
JULY 1976

È tutto chiaro? ...

1. Dove viene memorizzato un programma che non sia attualmente in esecuzione?
2. Quale parte del computer esegue le operazioni aritmetiche, come addizioni e moltiplicazioni?

Le istruzioni macchina

- Le istruzioni elementari eseguite da un calcolatore (i.e. dalla sua CPU) si chiamano ***istruzioni macchina***
- L'insieme di istruzioni macchina (**instruction set**) è **specifico** di una particolare CPU: quello di un Pentium™ è diverso da quello di uno SPARC™
- Una particolare CPU è la cosiddetta ***macchina virtuale Java*** (JVM, *Java Virtual Machine*)
 - la JVM non è una vera CPU... ma per il momento possiamo considerarla tale...



Le istruzioni macchina

- Esempio di alcune istruzioni macchina:
 - carica in un registro il valore contenuto nella posizione di memoria 40
 - carica in un altro registro il valore 100
 - se il primo valore è maggiore del secondo, prosegui con l'istruzione contenuta nella posizione di memoria 240, altrimenti con l'istruzione che segue quella attuale
- La codifica delle istruzioni macchina avviene sotto forma di **configurazioni di bit** conservate in memoria (che possono essere interpretate come numeri interi)
- Le precedenti istruzioni per la JVM diventano

21 40 16 100 163 240



Le istruzioni macchina

- In tutte le CPU, le istruzioni macchina si possono suddividere nelle categorie (i nomi delle istruzioni sono solo degli esempi):
 - **trasferimento dati**, tra registri e memoria principale
 - LOAD (verso un registro), STORE (verso la memoria)
 - **operazioni aritmetiche e logiche**, eseguite dalla ALU
 - **aritmetiche**: ADD, SUB, MUL, DIV
 - **logiche**: AND, OR, NOT
 - **salti**, per alterare il flusso di esecuzione sequenziale (viene modificato il Program Counter)
 - **incondizionato** (JUMP): salta in ogni caso
 - **condizionato**: salta solo se un certo valore è zero (JZ) o se è maggiore di zero (JGZ)

Le istruzioni macchina

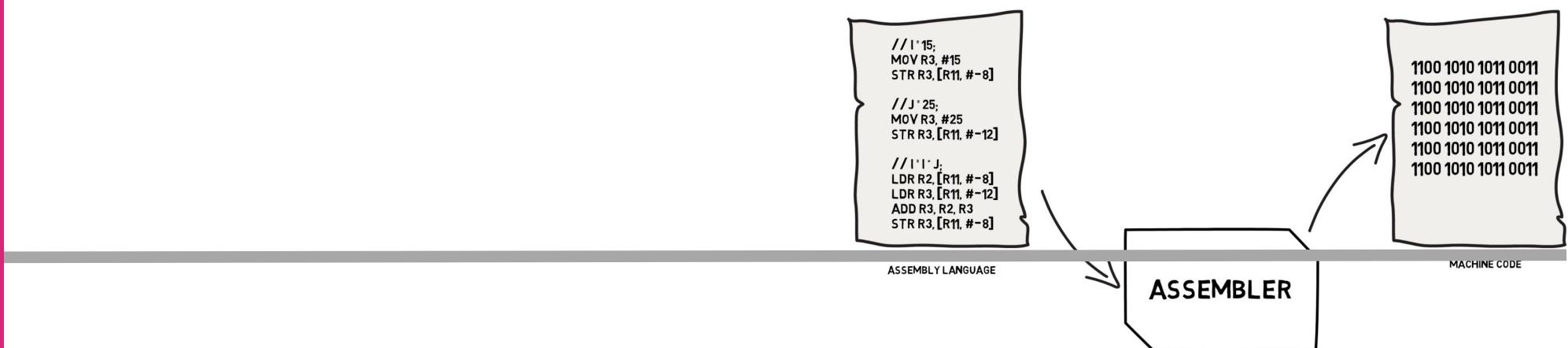
- Per eseguire un programma in un calcolatore è necessario scrivere *all'interno della memoria primaria le configurazioni di bit corrispondenti alle istruzioni macchina del programma*
- Per fare ciò è necessario conoscere tutti i codici numerici delle istruzioni macchina
 - 21 40 16 100 163 240 ...
- Questa operazione lunga e noiosa (che veniva eseguita agli albori dell'informatica) è stata presto automatizzata da *un programma in esecuzione sul computer stesso, detto assemblatore (assembler)*

L'assemblatore

- Utilizzando l'assemblatore, il programmatore scrive il programma mediante dei **nomi abbreviati** (**codici mnemonici**) per le istruzioni macchina, molto più facili da ricordare
 - esempio precedente per la JVM

iload	40
bipush	100
if_icmpgt	240

- L'uso di nomi abbreviati è assai più agevole, e il programma assemblatore si occupa poi di **tradurre** il programma in configurazioni di bit
- Tali linguaggi con codici mnemonici si dicono **linguaggi assembly** (uno diverso per ogni CPU)



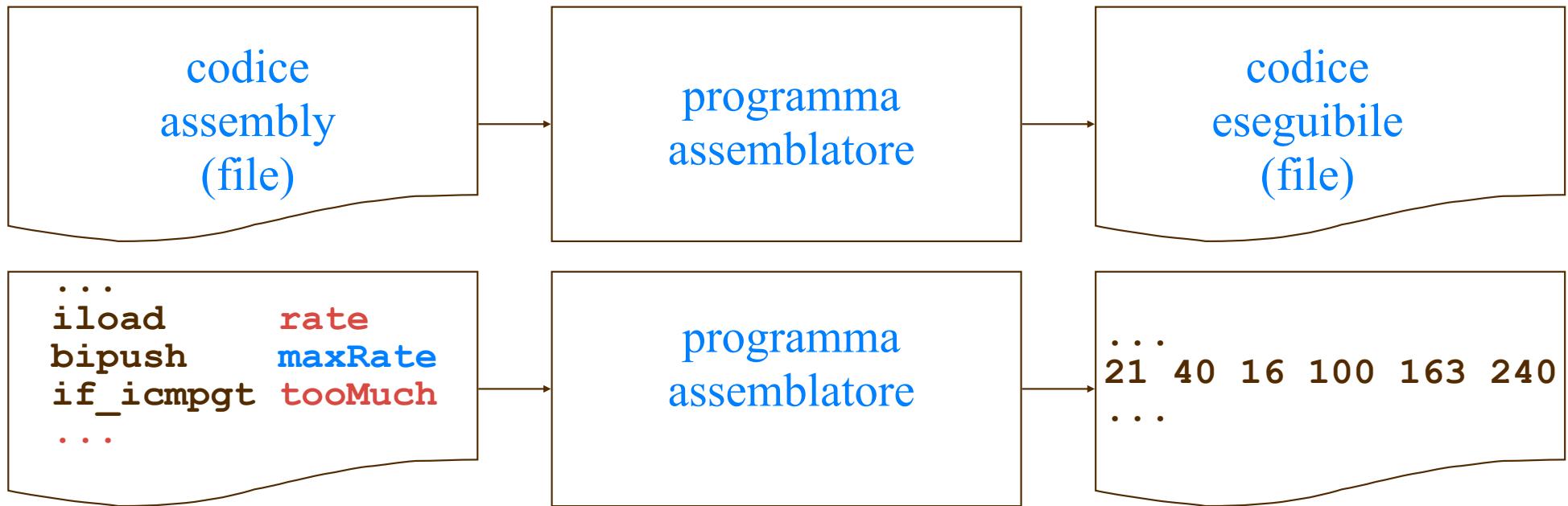
L'assemblatore

- Un'altra caratteristica molto utile dell'assemblatore è quella di poter assegnare dei *nomi* agli *indirizzi di memoria* e ai *valori numerici* e di usarli nelle istruzioni

```
iload      rate  
bipush    maxRate  
if_icmpgt tooMuch
```

- In questo modo il programma è *molto più leggibile*, perché viene evidenziato il *significato* degli indirizzi di memoria e dei valori numerici
- C'è corrispondenza biunivoca fra insieme di istruzioni assembly e instruction set del processore

L'assemblatore



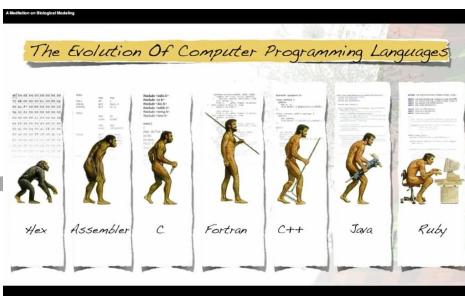
- Il programma assemblatore traduce istruzioni assembly in istruzioni macchina (corrispondenza biunivoca)
- Il programma assemblatore riceve in ingresso un **file** contenente codice in linguaggio assembly e produce in uscita un **file** contenente istruzioni macchina.

I linguaggi assembly

- **Vantaggio:** rappresentarono un grosso passo avanti rispetto alla programmazione in linguaggio macchina
- **Problema:** occorrono *molte istruzioni* per eseguire anche le operazioni più semplici
- **Problema:** le istruzioni di uno stesso programma **cambiano** al cambiare della CPU
 - ➡ ■ è molto costoso scrivere programmi che possano funzionare su diverse CPU, perché praticamente bisogna riscriverli completamente

Linguaggi ad alto livello

- Negli anni '50 furono inventati i primi linguaggi di programmazione ***ad alto livello***
 - **FORTRAN**: primo “vero” linguaggio
 - **BASIC, COBOL**
 - Anni '60 e '70: programmazione strutturata
 - **Pascal** (Niklaus Wirth, 1968)
 - **C** (Brian Kernigham e Dennis Ritchie, 1970-75)
 - Anni '80 e '90, programmazione orientata agli oggetti
 - **C++** (Bjarne Stroustrup, 1979)
 - **Java** (James Gosling e Patrick Naughton, 1991)
- Il programmatore esprime la sequenza di operazioni da compiere, senza scendere al livello di dettaglio delle istruzioni macchina



```
if (rate > 100)  
System.out.print("Troppo");
```

Linguaggi ad alto livello

- Un programma, detto **compilatore**, legge il programma in linguaggio ad alto livello e genera il corrispondente programma nel linguaggio macchina di una certa CPU
- Linguaggi sono **indipendenti dalla CPU**
 - ma il prodotto della compilazione (codice eseguibile), non è indipendente dalla CPU
 - occorre compilare il programma con un diverso compilatore per ogni CPU sulla quale lo si vuole eseguire
- Si dice **codice sorgente** il codice scritto in un linguaggio ad alto livello
- Si dice **codice eseguibile** il codice scritto nel linguaggio macchina

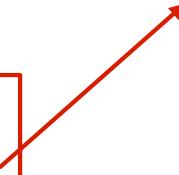


Linguaggi ad alto livello

- I linguaggi di programmazione, creati dall'uomo, hanno una sintassi molto più rigida di quella dei linguaggi naturali, per agevolare il compilatore
- Il compilatore segnala gli ***errori di sintassi***

```
if (rate > 100) System.out.print("Troppo");
```

C'è una parentesi chiusa di troppo



- e ***non tenta di capire***, come farebbe un utente umano, perché sarebbe molto difficile verificare le ***intuizioni*** del compilatore
 - meglio la segnalazione di errori!***

Linguaggi ad alto livello

- Esistono molti linguaggi di programmazione ad alto livello, così come esistono molte lingue
- L'esempio seguente è in linguaggio Java

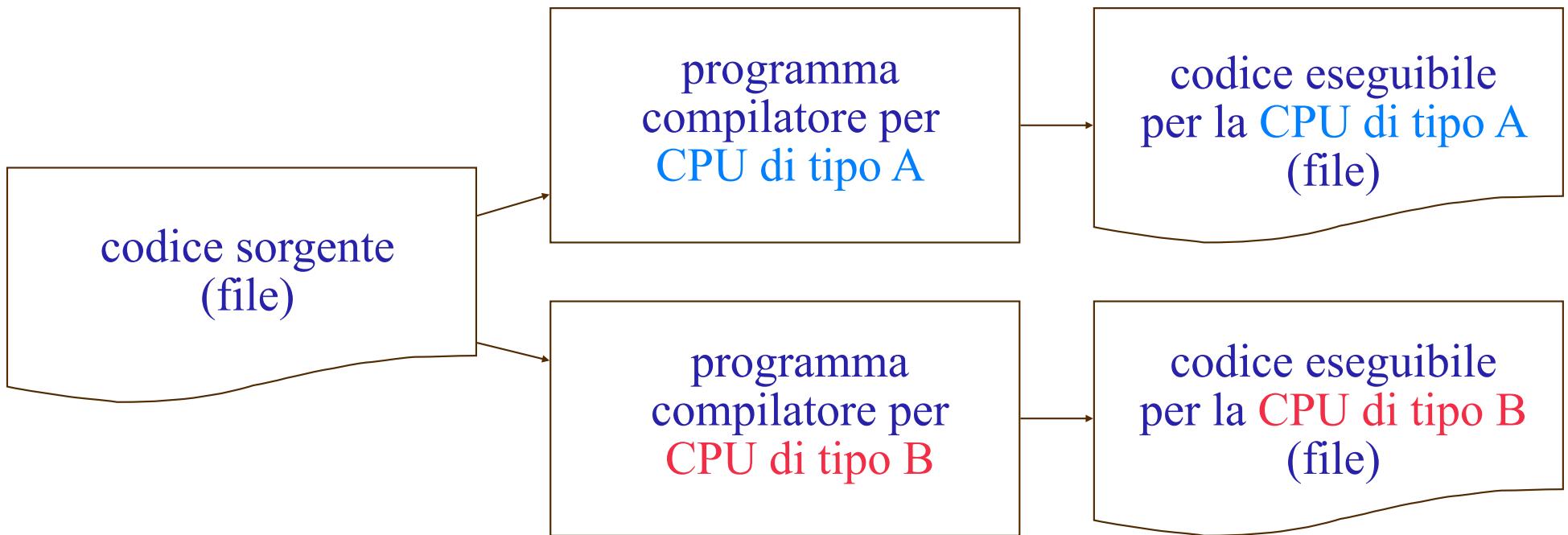
```
if (rate > 100) System.out.print("Troppo");
```

- e questo è l'equivalente in linguaggio Pascal

```
if rate > 100 then write ('Troppo');
```

- La sintassi di un linguaggio viene scelta dai progettisti del linguaggio stesso, come compromesso fra leggibilità, facilità di compilazione e coerenza con altri linguaggi

Compilatore



- Il compilatore traduce codice in linguaggio ad alto livello in codice eseguibile di uno specifico processore
 - riceve in ingresso un file contenente codice in linguaggio ad alto livello e produce in uscita un file contenente codice eseguibile
- Lo stesso codice sorgente può essere tradotto in codice eseguibile di processori diversi, con compilatori diversi



Il linguaggio Java

- Nato nel 1991 in Sun Microsystems,
- da un gruppo di progettisti guidato da Gosling e Naughton
 - ... che consumarono molto caffè durante la progettazione del linguaggio
- Progettato per essere **semplice** e **indipendente dalla CPU** (o, come anche si dice, dall'hardware, dalla piattaforma o dall'architettura)
- Il primo prodotto del progetto Java fu un **browser**, **HotJava**, presentato nel 1994, che poteva scaricare programmi (detti **applet**) da un server ed eseguirli sul computer dell'utente, **indipendentemente** dalla sua piattaforma



Il linguaggio Java

- Il linguaggio Java è stato adottato da moltissimi programmatore perché
 - è più **semplice** del suo diretto concorrente, C++
 - consente di **scrivere e compilare una volta ed eseguire dovunque** (cioè su tutte le piattaforme)
“compile once, execute everywhere”
 - ha una **ricchissima libreria standard** che mette a disposizione dei programmatore un insieme vastissimo di funzionalità **standard**, indipendenti anche dal sistema operativo

La libreria di classi standard

- Per effettuare molte operazioni comuni
 - Per esempio: scrivere sulla shell (“standard output”)
- è necessario ***interagire con il sistema operativo***
 - operazioni ***di basso livello*** che richiedono conoscenze specifiche
- Queste operazioni, per chi utilizza Java, sono state già realizzate dagli autori del linguaggio, che hanno scritto delle classi apposite (per esempio, **System**)
- Il bytecode di queste classi si trova all'interno di ***librerie standard***, che sono ***raccolte di classi***
- ***Non è necessario avere a disposizione il codice sorgente di queste classi, né capirlo!*** Comodo...

Il linguaggio Java per gli studenti

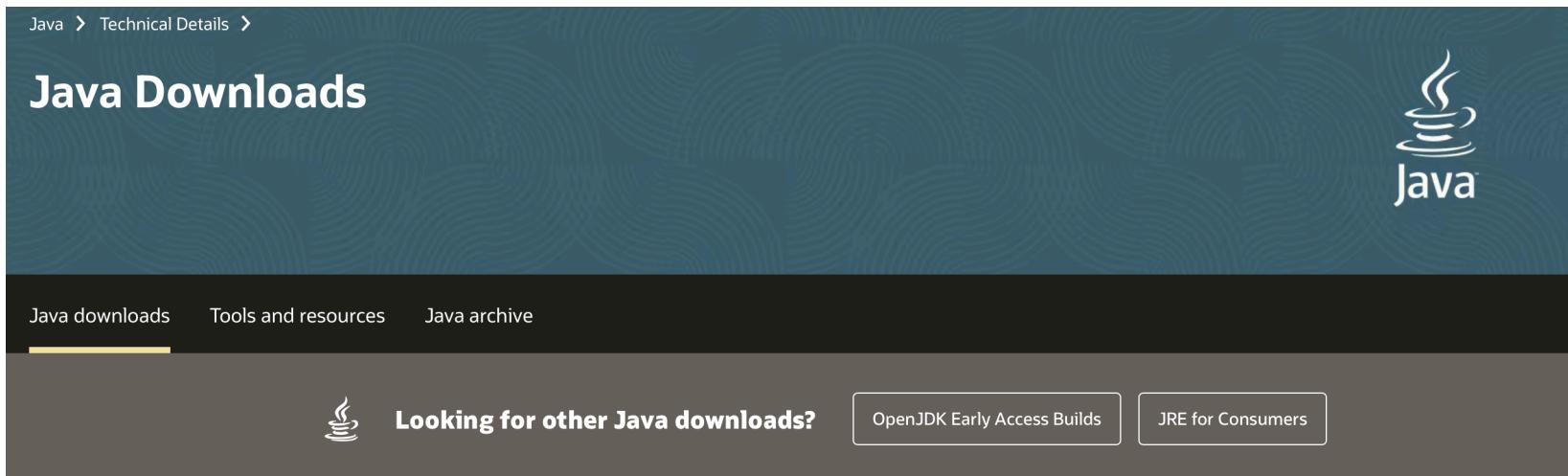
- Dato che Java non è stato progettato per la didattica
 - *non è molto semplice scrivere programmi Java molto semplici!*
- Anche per scrivere programmi molto semplici è necessario conoscere parecchi dettagli “tecnici”, un potenziale problema nelle prime lezioni
- Adotto lo stile didattico di *usare alcuni costrutti sintattici senza spiegarli o approfondirli*, rimandando tali fasi al seguito



Java Development Kit (JDK)

- Da **scaricare** e **installare** sul vostro PC!

<https://www.oracle.com/java/technologies/javase-downloads.html>



Java 21 and Java 17 available now

JDK 21 is the latest long-term support release of Java SE Platform.

[Learn about Java SE Subscription](#)

[JDK 21](#) [JDK 17](#) [GraalVM for JDK 21](#) [GraalVM for JDK 17](#)

JDK Development Kit 21 downloads

JDK 21 binaries are free to use in production and free to redistribute, at no cost, under the [Oracle No-Fee Terms and Conditions \(NFTC\)](#).

JDK 21 will receive updates under the NFTC, until September 2026, a year after the release of the next LTS. Subsequent JDK 21 updates will be licensed under the [Java SE OTN License](#) (OTN) and production use beyond the [limited free grants](#) of the OTN license will [require a fee](#).

Le fasi della programmazione

- L'attività di programmazione si esegue in tre fasi
 - scrittura del programma (**codice sorgente**)
 - compilazione del codice sorgente
 - creazione del **codice eseguibile** (codice macchina)
 - esecuzione del programma
- Per scrivere il codice sorgente si usa un **editor di testo**, salvando (memorizzando) il codice in un file
 - **HelloTester.java**

Individuare il compilatore Java

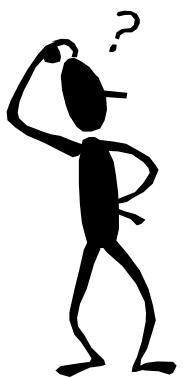
- Il modo di utilizzo del compilatore Java dipende dal sistema operativo
 - si seleziona con il mouse un'icona sullo schermo
 - si seleziona una voce in un menu di comandi
 - **si compone il nome di un comando sulla tastiera**
 - si utilizza un ambiente integrato per lo sviluppo software (IDE, *Integrated Development Environment*)
- Nel nostro corso useremo i comandi da tastiera del **JDK (*Java Development Kit*)**

La compilazione del sorgente

- Compilando il codice sorgente di un programma (gli enunciati in linguaggio Java) si ottiene un particolare formato di **codice eseguibile**, detto **bytecode**, che è codice macchina per la Java Virtual Machine (JVM)

- L'istruzione

```
javac HelloTester.java  
genera il file HelloTester.class
```



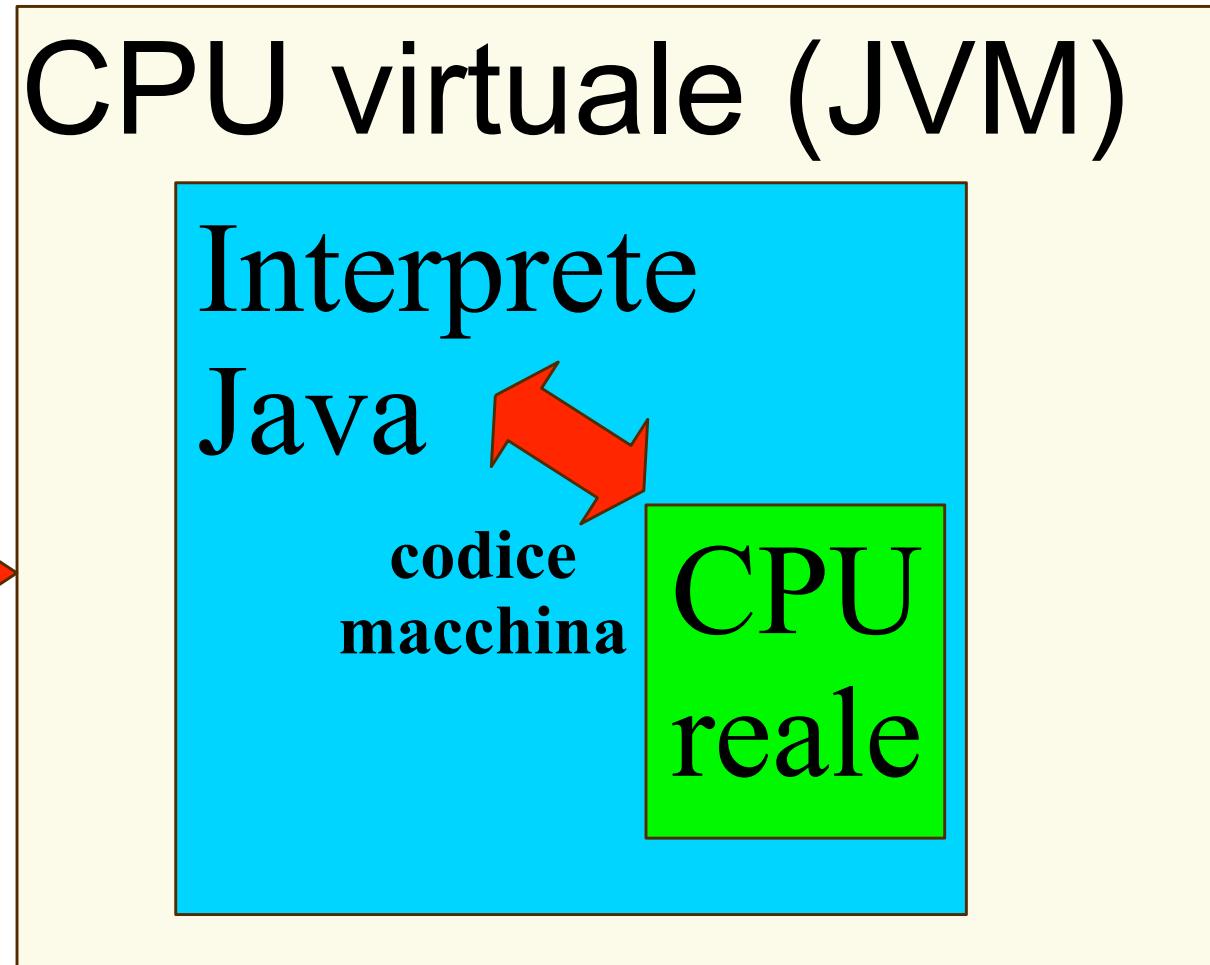
- Quindi il bytecode non è codice direttamente eseguibile, dato che la JVM non esiste...
- Il file con il codice bytecode contiene una **traduzione** delle istruzioni del programma

L'esecuzione del programma

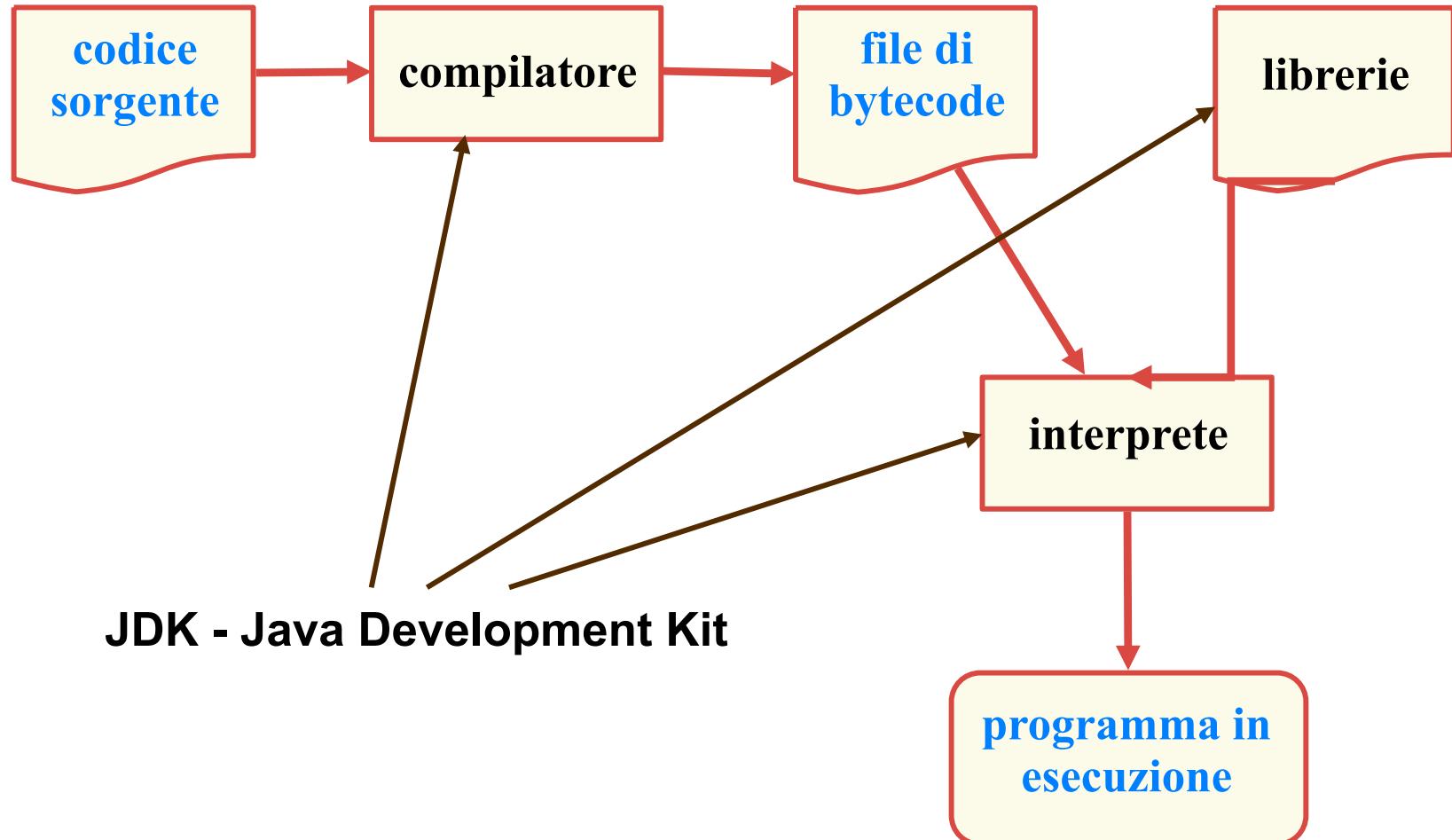
- Per eseguire un programma si usa l'**interprete Java**, un programma eseguibile sul computer dell'utente che
 - carica in memoria principale (RAM) il bytecode del programma (**HelloTester**)
 - avvia il programma
 - carica successivamente altri file di bytecode che sono eventualmente necessari durante l'esecuzione (per esempio dalla libreria standard)
 - traduce “**al volo**” le istruzioni del bytecode in istruzioni macchina della CPU reale (che esegue il codice)
- L'istruzione
 - `java HelloTester`
 - esegue il programma **HelloTester**

La Java Virtual Machine

bytecode



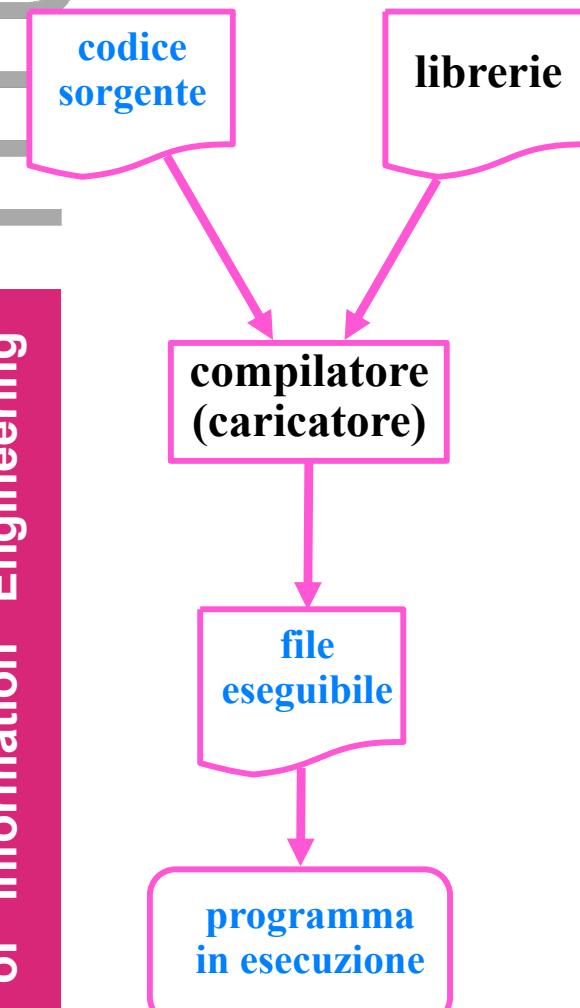
Il processo di programmazione in Java



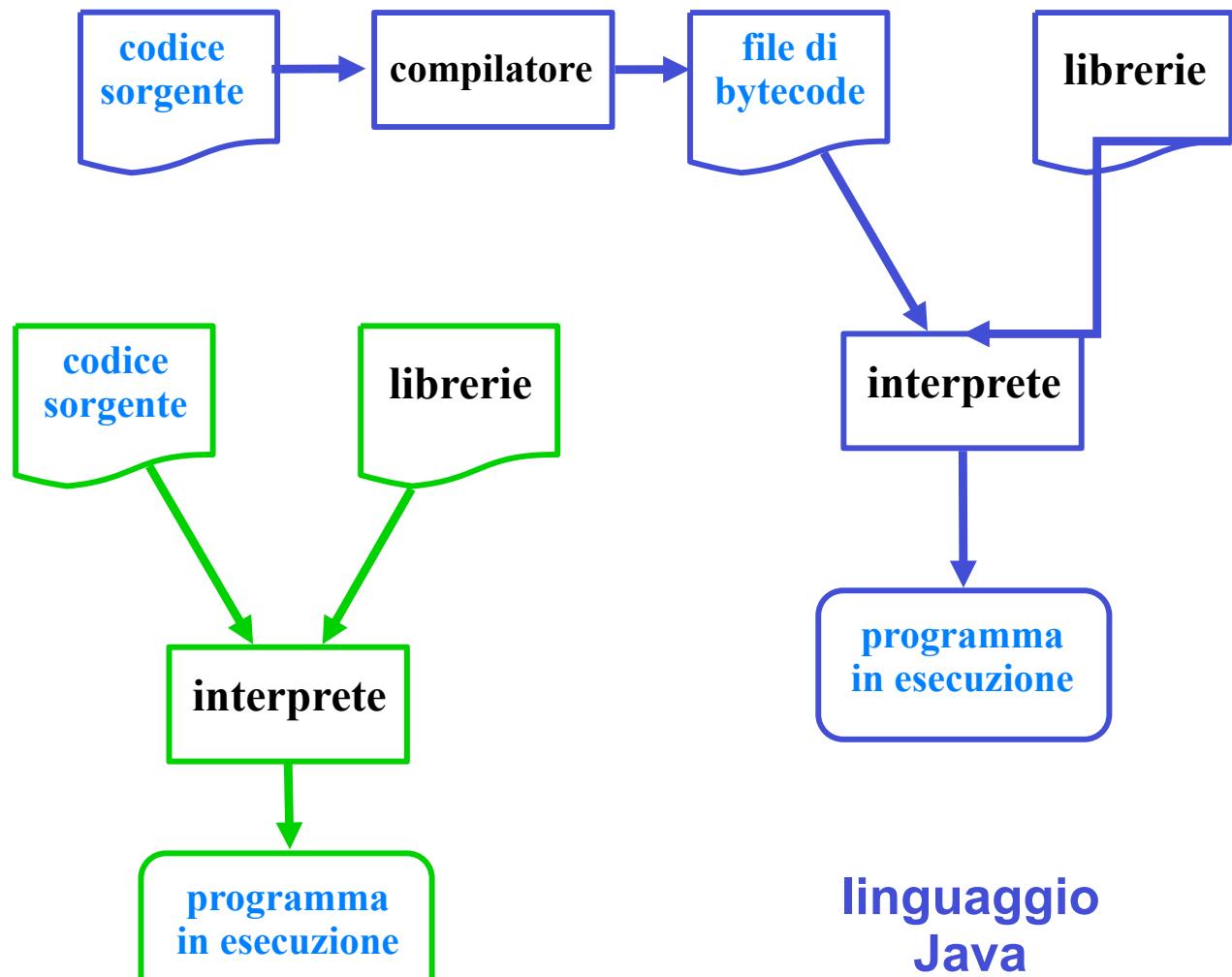
Compilatore e/o interprete

- Per passare dalla scrittura del file sorgente in linguaggio Java all'esecuzione del programma su una particolare CPU, si usano il **compilatore e l'interprete**
- Con la maggior parte degli altri linguaggi di programmazione ad alto livello, invece, si usa soltanto il **compilatore oppure l'interprete**
 - con **linguaggi compilati** come Pascal, C e C++, si usa il **compilatore** per creare un file eseguibile, contenente codice macchina, a partire da file sorgenti (con l'eventuale ausilio di un **caricatore**, *loader*, che interagisce con una libreria)
 - con **linguaggi interpretati** come BASIC e PERL, l'**interprete** traduce "al volo" il file sorgente in codice eseguibile e lo esegue, senza creare un file eseguibile

Il processo di programmazione



linguaggi compilati



linguaggi interpretati

linguaggio Java

Compilatore e/o interprete

- Il fatto che un linguaggio sia compilato o interpretato influisce fortemente su quanto è
 - facile eseguire lo stesso programma su computer aventi diverse CPU (**portabilità**)
 - veloce l'esecuzione di un programma (**efficienza**)
- Entrambi questi aspetti sono molto importanti nella fase di scelta di un linguaggio di programmazione da utilizzare in un progetto
- Il linguaggio Java, da questo punto di vista, è un **linguaggio misto**, essendo sia compilato sia interpretato, in fasi diverse

Portabilità

- I programmi scritti in un linguaggio **interpretato** sono portabili
- I programmi scritti in un linguaggio **compilato**
 - sono portabili a livello di file sorgente, ma è necessario compilare il programma su ogni diversa CPU
 - non sono portabili a livello di file eseguibile, perché esso contiene codice macchina per una particolare CPU
- *I programmi scritti in linguaggio Java sono portabili*, oltre che a livello di file sorgente, anche *a un livello intermedio*, il livello del bytecode
 - possono essere compilati una sola volta ed eseguiti da *interpreti diversi* su diverse CPU



Efficienza

- I programmi scritti in un linguaggio interpretato sono poco efficienti
 - l'intero processo di traduzione in linguaggio macchina deve essere svolto a ogni esecuzione
- I programmi scritti in un linguaggio compilato sono molto efficienti
 - l'intero processo di traduzione in linguaggio macchina viene svolto prima dell'esecuzione, una volta per tutte
- I programmi scritti in linguaggio **Java** hanno un'**efficienza intermedia**
 - parte del processo di traduzione viene svolto una volta per tutte (dal compilatore) e parte viene svolto a ogni esecuzione (dall'interprete)

Portabilità ed efficienza

- Se si vuole soltanto la portabilità, i linguaggi interpretati sono la scelta migliore
- Se si vuole soltanto l'efficienza, i linguaggi compilati sono la scelta migliore
- Se si vogliono perseguire ***entrambi gli obiettivi***, come quasi sempre succede, ***il linguaggio Java può essere la scelta vincente***



U

È tutto chiaro? ...

- 1. Quali sono i due principali vantaggi del linguaggio Java?
- 2. Quanto tempo occorre per imparare a utilizzare l'intera libreria di Java?

Compilare un semplice programma



I CAN'T KEEP CALM
AND CARRY ON

I'M A PROGRAMMER AND

I GET 21 ERRORS
IN A 20 LINES PROGRAM CODE

Il (solito...) primo programma Java

- Tradizionalmente, il primo programma che si scrive quando si impara un linguaggio di programmazione ha il compito di visualizzare sullo schermo un semplice saluto

Hello, World!

- Scriviamo il programma nel file
 - **HelloTester.java**
 - usando un editor di testi

Me:

I am good in C language.

Interviewer:

Then write "Hello World" using C.

Me:

c c ccccc c c c
c c c c c c c
ccccccccc c c c
c c c c c c c
c c ccccccc ccccccc ccccccc
c c ccccccc ccccccc c
c c c c c c c
cc c cc c c c c c
c c c c c c c c
c c ccccccc c c c c c
c c c c c c c c

Il nostro primo programma Java

```
public class HelloTester
{
    public static void main(String[] args)
    {
        // visualizza un messaggio sullo schermo
        System.out.println("Hello, World!");
    }
}
```

Occorre fare molta attenzione

- il testo va inserito esattamente come è presentato (per ora...)
- ***maiuscole e minuscole sono considerate distinte***
- il file ***deve*** chiamarsi **HelloTester.java**

Il nostro primo programma Java

- A questo punto **compiliamo** il programma

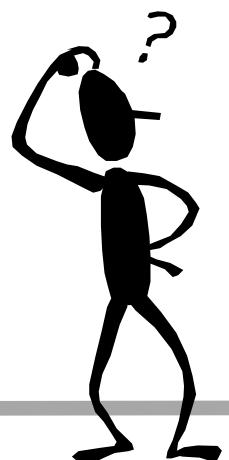
```
javac HelloTester.java
```

- e il compilatore genera il file **HelloTester.class**
- Ora **eseguiamo** il programma

```
java HelloTester
```

- ottenendo la visualizzazione del messaggio di saluto sullo schermo

```
Hello, World!
```



Analisi di HelloTester: classi

- La prima riga

```
public class HelloTester
```

- **definisce** una nuova **classe**
- Le classi sono **fabbriche di oggetti** e rappresentano un concetto fondamentale in Java, che è un linguaggio di programmazione **orientato agli oggetti** (**OOP**, **Object-Oriented Programming**)
- Per il momento, consideriamo gli **oggetti** come **elementi da manipolare in un programma Java**



Analisi di HelloTester: classi

- La **parola chiave** **public** indica che la classe **HelloTester** può essere utilizzata da tutti (in seguito vedremo **private**...)

```
public class HelloTester
```



- Una **parola chiave** è una **parola riservata** del linguaggio che va scritta esattamente così com'è e che non può essere usata per altri scopi
- La parola chiave **class** indica che inizia la **definizione** di una **classe**
- Ciascun **file sorgente** (parte di un programma Java) può contenere **una sola classe pubblica**, il cui nome **deve coincidere** con il nome del file

Le parole chiave di Java

Cfr. Appendice C

abstract	continue	for	new	switch
assert	default	if	package	synchronized
boolean	do	goto	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while

Analisi di HelloTester: metodi

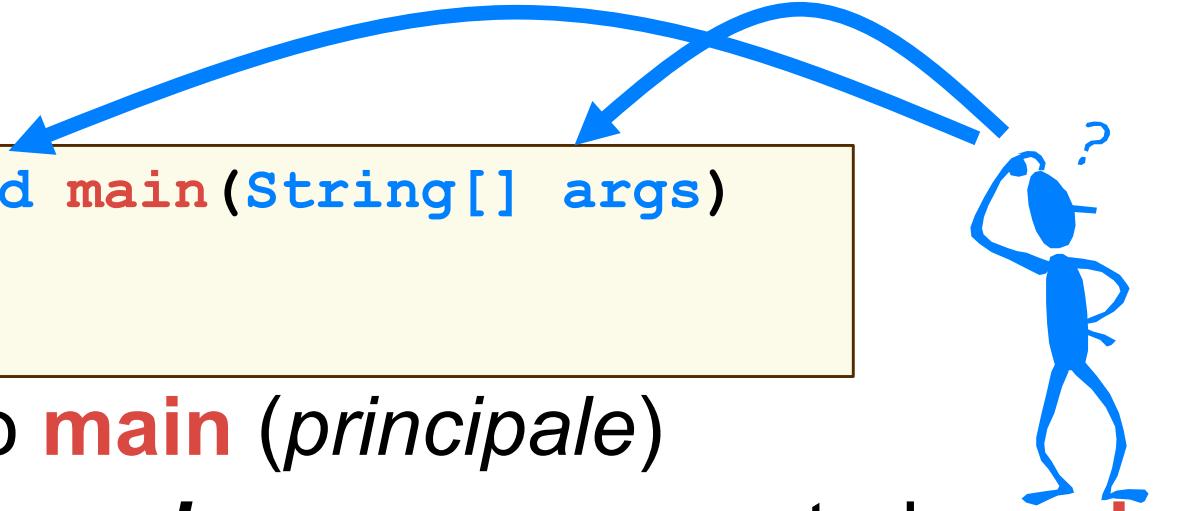
- Per ora non usiamo le classi come fabbriche di oggetti, ma come contenitori di **metodi**
- Un **metodo** serve a definire una sequenza di istruzioni o **enunciati** che **descrive come svolgere un determinato compito** (in altri linguaggi i metodi si chiamano *funzioni* o *procedure*)
- Un metodo **deve** essere inserito in una classe, quindi le classi rappresentano il meccanismo principale per l'organizzazione dei programmi



Analisi di HelloTester: metodi

- La costruzione

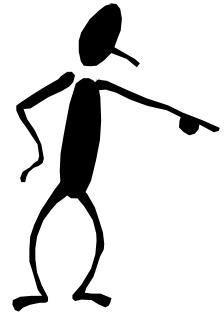
```
public static void main(String[] args)  
{ ... }
```



- definisce il metodo **main** (*principale*)
- Un'applicazione Java **deve** avere un metodo **main**
- Anche qui, **public** significa *utilizzabile da tutti*
- Invece, **static** significa che il metodo **main** non è associato a una istanza della classe, ma solo alla classe stessa



Sintassi: programma semplice



- Sintassi:

```
public class NomeClasse
{ public static void main(String[] args)
    { enunciati
    }
}
```

- Scopo: eseguire un programma semplice, descritto da **enunciati** e contenuto nel file **NomeClasse.java**
- Nota: la parte in **blu** viene per ora considerata una **infrastruttura necessaria**, approfondita in seguito

Analisi di HelloTester: commenti

- Nel programma sono presenti anche dei **commenti**, che vengono **ignorati** dal compilatore, ma che rendono il programma molto più comprensibile
 - // visualizza un...
- Un commento **inizia con una doppia barra //** e **termina alla fine della riga**
- Nel commento si può scrivere **qualsiasi cosa**
- Se il commento si deve estendere **per più righe**, è molto scomodo usare tante volte la sequenza //
- Si può iniziare un commento con /* e terminarlo con */

/// questo e' un commento
/// lungo,inutile...
/// ... e anche scomodo

/*
questo e' un commento
lungo ma inutile...
*/

Analisi di HelloTester: enunciati

- Gli enunciati del **corpo** di un metodo (*gli enunciati contenuti tra le parentesi graffe*) vengono eseguiti uno alla volta ***nella sequenza in cui sono scritti***
 - Ogni enunciato termina con il carattere ;
 - Il metodo **main** del nostro esempio ha un solo enunciato, che visualizza una riga di testo

```
System.out.println("Hello, World!");
```
 - Ma **dove** la visualizza? Un programma può inserire testo in una finestra, scriverlo in un file o anche inviarlo a un altro calcolatore attraverso Internet...

Analisi di HelloTester: oggetti

```
System.out.println("Hello, World!");
```

- Nel nostro caso la destinazione è l'**output standard** (o “flusso di uscita standard”)
 - è una proprietà di ciascun programma che dipende dal sistema operativo del computer
- In Java, l'output standard è rappresentato da un **oggetto** di nome **out**
 - come ogni metodo, **anche gli oggetti devono essere inseriti in classi**: **out** è inserito nella classe **System** della **libreria standard**, che contiene oggetti e metodi da utilizzare per accedere alle **risorse di sistema**
 - per **usare** l'oggetto **out** della classe **System** si scrive

```
System.out
```



Analisi di HelloTester: l'oggetto out

```
System.out.println("Hello, World!");
```

- *System* è una classe della **java platform API** che è una collezione di classi pronte per essere usate, fornite dal progettista del linguaggio insieme al linguaggio Java
- **Application Program Interface (API)** significa Interfaccia per la programmazione di applicazioni
- *System.out* e' un oggetto definito nella classe *System* di tipo *PrintStream*
- *PrintStream* è un'altra classe della **java platform API**



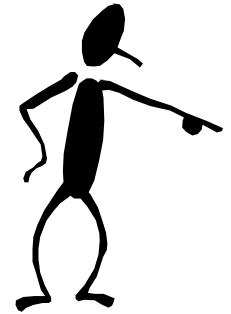
Analisi di HelloTester: oggetti

```
System.out.println("Hello, World!");
```

- Quando si usa un oggetto, bisogna specificare **cosa** si vuol fare con l'oggetto stesso
 - in questo caso vogliamo **usare un metodo** dell'oggetto **out**, il metodo **println**, che stampa una riga di testo
 - per **usare** il metodo **println** dell'oggetto **System.out** si scrive **System.out.println(parametri)**
 - la coppia di parentesi tonde racchiude le informazioni necessarie per l'esecuzione del metodo (**parametri**)
- A volte il carattere **punto** significa “usa un oggetto di una classe”, altre volte “usa un metodo di un oggetto”: dipende dal contesto...



Sintassi: invocazione di metodo



- Sintassi: `oggetto.nomeMetodo(parametri)`
- Scopo: invocare il metodo **nomeMetodo** dell'**oggetto**, fornendo **parametri** se sono richiesti
- Nota: se non sono richiesti parametri, le parentesi tonde devono essere indicate ugualmente
- Nota: se ci sono due o più parametri, essi vanno separati l'uno dall'altro con una **virgola**, all'interno delle parentesi tonde

Analisi di HelloTester: stringhe

- Una sequenza di caratteri racchiusa tra virgolette si chiama stringa (*string*)

"Hello, World!"

- C'è un motivo per dover racchiudere le stringhe tra virgolette
 - come farebbe altrimenti il compilatore a capire che volete, ad esempio, stampare la parola **class** e non che volete iniziare la definizione di una nuova classe?
- Il metodo **println** può anche stampare numeri

```
System.out.println(7); System.out.println(3 + 4);
```
- C'è anche il metodo **print**, che funziona come **println** ma **non va a capo al termine della stampa**

Una nota stilistica

- Diversamente da altri linguaggi (es. FORTRAN) il linguaggio Java consente una ***disposizione del testo a formato libero***: gli spazi e le interruzioni di riga (“andare a capo”) non sono importanti, tranne che per separare parole
- Sarebbe quindi possibile scrivere

```
public class HelloTester{public static void  
    main (String[ ] args) { System.out.  
        println("Hello, World!") ; } }
```

- Bisogna però ***fare attenzione alla leggibilità!***

È tutto chiaro? ...

1. Come si può modificare il programma HelloTester in modo che visualizzi le parole “Hello” e “World!” su due righe consecutive?
2. Il programma continua a funzionare anche senza la riga che inizia con // ?
3. Cosa viene visualizzato dai seguenti enunciati?

```
System.out.print("My lucky number is");
```

```
System.out.println(3 + 4 + 5);
```

