

Fondamenti di Programmazione

Nicola Galesi

Dipartimento di Informatica

Pagina Personale: www.di.uniroma1.it/~galesi

Pagina del Corso: www.di.uniroma1.it/fond.html

Algoritmi

Un **algoritmo** é la descrizione di un procedimento che porta alla soluzione di un determinato compito.

Viene specificato mediante una serie di regole non ambigue che devono potere essere eseguite meccanicamente, senza cioè doverne capire il significato.

Esempi di algoritmi

- L'algoritmo di Euclide per calcolare l'mcd
- Una ricetta
- L'algoritmo per moltiplicare due numeri in colonna

Algoritmi & Macchine

un algoritmo, essendo un processo meccanico, può essere eseguito su una **macchina**.

Vi sono macchine in grado di eseguire solo determinati algoritmi.....



Pascalina....

La macchina inventata da Blaise Pascal nel 1642, in grado di sommare due numeri

Algoritmi & Macchine

..... e **macchine universali**, ad es. i **computer**, capaci di eseguire qualunque algoritmo che sia specificato in un linguaggio comprensibile alla macchina



IBM Mare Nostrum

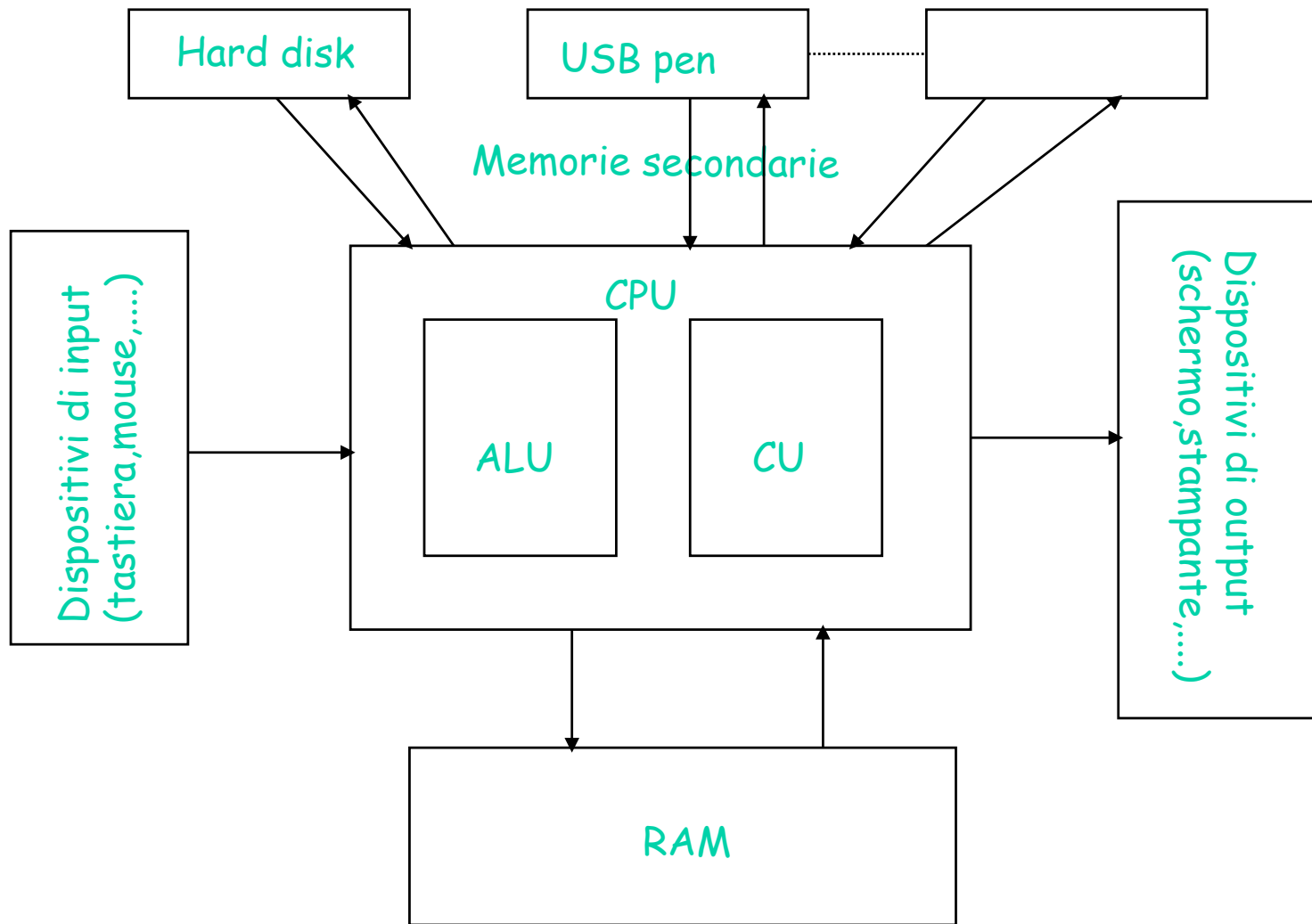
3564 Processori

Architettura di un Calcolatore

Un calcolatore, secondo il modello astratto di Von Neumann è composto da tre parti:

- Una memoria detta **RAM** (Random Access memory) che contiene dati e programmi (istruzioni macchina) che il calcolatore esegue.
- Un'unità di calcolo detta **CPU** (central Processing Unit) suddivisa in una
 - **ALU** (Arithmetic Logic Unit) esegue operazioni logico aritmetiche
 - **CU** (Control Unit) preleva le istruzioni dalla RAM e le rende eseguibili dalla ALU.
- Unità di comunicazione con l'esterno

Architettura di un Calcolatore



Memoria Principale

La RAM o memoria principale può immaginarsi come un insieme di "cassette" o "celle":

- Ogni cella ha un indirizzo che la identifica univocamente e non varia
- Ogni cella contiene un valore.

Entrambi indirizzo e valore sono numeri binari

La CU può accedere a qualunque cella direttamente ,
da qui il nome RAM.

Il contenuto della memoria non viene preservato. È **volatile**.

Unità di misura

bit	0/1
byte	8 bit
KB (kilo byte)	2^{10} bytes (1024)
MB (mega byte)	2^{20} bytes
GB (mega byte)	2^{30} bytes
TB (tera byte)	2^{40} bytes

Programmare

Iniziamo dal livello più basso (macchina hw):

A livello di hardware, programmare vuol dire specificare una sequenza di istruzioni-macchina per implementare un determinato compito.

Esempio di istruzioni macchina

INP x (ricevi un valore dall'input e salvalo nella cella con indirizzo x)

MOV x y (muovi il contenuto della cella x nella cella y)

ADD x y (somma il contenuto delle celle x e y e copialo in x)

Programmare

Chiaramente per programmare su un calcolatore compiti complessi (come un video gioco, la gestione di una base di dati, un programma di simulazione di volo) abbiamo bisogno di poter specificare istruzioni "piu complesse", altrimenti diventa un compito improbo e non conveniente.

Linguaggi di alto livello.

Il **C** e' un linguaggio di alto livello che ci consente di scrivere programmi complessi.

DOMANDA. Come fa il calcolatore ad eseguire un programma scritto in un linguaggio di alto livello ??

Programmare

Esistono programmi che fanno parte del

Software di Base

che si occupano di tradurre un programma scritto in un linguaggio di alto livello in una sequenza di istruzioni-macchina comprensibile dal calcolatore per essere eseguiti

- **Compilatore** è un programma che legge il programma sorgente e lo trasforma in un programma in linguaggio macchina eseguibile dal calcolatore
- **Interprete** è un programma che esegue, trasformandole in istruzioni macchina, le istruzioni del programma sorgente una dopo l'altra, senza occuparsi dell'intero programma

Linguaggi

Per definire un linguaggio abbiamo bisogno di specificare:

- Un **alfabeto**: insieme dei simboli con cui formare i termini del linguaggio
- La **sintassi**, ovvero una grammatica che definisca le regole per la produzione di frasi ben formate del linguaggio
- Una **semantica**, un insieme di regole con cui dare un significato alla frasi ben formate del linguaggio

Linguaggi

Fraasi ben formate in linguaggio naturale

Giovanni mangia un gelato.

Leopardi ha scritto lo Zibaldone.

Fraasi sintatticamente "scorrette" (non ben formate)

Cane il mangia gelato

Lucia la piove la sera

...

Fraasi sintatticamente corrette ma semanticamente scorrette (almeno nel senso comune)

Il cane parla di se

.....

Linguaggi di programmazione

- frasi sintatticamente corrette = programmi
- regole del linguaggio = sintassi
- semantica = esecuzione del programma

Possono esistere dei programmi perfettamente corretti
la cui esecuzione non ha alcun senso

Ricordate che per eseguire il programma dobbiamo
trasformarlo in un programma eseguibile dalla macchina

- compilatori
- interpreti

Compilatori ed Interpreti

Un programma di un linguaggio di alto livello come C, JAVA è contenuto in un file, è dunque una sequenza di bits e non può essere direttamente eseguibile dalla macchina.

Per eseguire il programma vi sono due possibilità a seconda del linguaggio usato

- **interprete**: un programma che legge il programma e lo esegue istruzione dopo istruzione.
- **compilatore**: un programma che traduce il programma sorgente in un programma in un linguaggio macchina che può essere eseguito dalla macchina

Compilatori ed Interpreti

In generale gli interpreti sono meno efficienti perché devono tradurre ogni istruzione in linguaggio macchina.

D'altra parte i compilatori, traducendo il programma sorgente in un programma in linguaggio macchina

Producono codice che dipende dalla macchina e per tanto non eseguibile su tutte le macchine. In ogni macchina abbiamo bisogno di ricompilare il programma.

Il **C** è un linguaggio di programmazione che richiede un compilatore.

FILES e codifica binaria

L'unico e (più conveniente) metodo di rappresentazione dell'informazione utilizzato da dispositivi elettronici come i computer è basato sul bit 0/1 che corrisponde allo stato di aperto/chiuso in cui può trovarsi una porta.

Quindi per memorizzare dei testi occorre poter codificare i caratteri mediante sequenze di 0 e 1, vale a dire di numeri (in rappresentazione binaria).

Chiaramente tra i caratteri includiamo anche i caratteri che permettono di scrivere numeri come il carattere "9".

Codice ASCII e UNICODE

Negli anni 60 venne definito il codice ASCII che ad ogni carattere della lingua inglese (piu altri caratteri speciali, come il CR, \$,% ecc.) associava un numero che permetteva quindi di rappresentarlo nella memoria di un calcolatore come la sequenza di 0 e 1 rappresentazione binaria di quel carattere.

Il codice ASCII è stato poi esteso per includere caratteri accentati e finalmente più recentemente è stato superato dal codice UNICODE (a 16 bit in seguito a 32 bit ...) che permette la rappresentazione di oltre 60000 caratteri, inclusi i più di 20000 ideogrammi cinesi

Esempio

Carattere

codice UNICODE

9

0000 0000 0011 1001

Tipologie di Files

Files di Testo

Un file di testo è formato da una sequenza di bit che rappresentano caratteri e quindi un testo (per esempio in italiano). Come facciamo per visualizzare il testo effettivamente scritto?

Esistono degli appositi programmi detti **Editori di Testo**, che consentono di **aprire** i files di testo e di visualizzarne il contenuto sotto forma di caratteri. In ambiente **UNIX**, editor di testo sono ad esempio **vi**, **Emacs** ed altri.

Files Eseguibili

Un file eseguibile è anch'esso una sequenza di bit che però rappresentano una sequenza di istruzioni in linguaggio macchina che il computer attraverso la CPU può direttamente eseguire. Il **Sistema Operativo** contiene gli appositi programmi che sono in grado di riconoscere un file eseguibile ed effettivamente eseguirlo.

Tipologie di Files

Tutti i file in fin dei conti sono sequenze di bits.

Come fa il SO a sapere che certi file sono di testo ed altri eseguibili. ??

Di fatto no non lo sa. I programmi del SO che devono eseguire programmi sono in grado di riconoscere se un programma è un file eseguibile e in quel caso lo eseguono. Oppure riconoscono che è un file di testo e in quel caso il programma appropriato è in grado di aprirlo e visualizzarlo correttamente.

Nomi di file ed estensioni

Ad ogni file daremo un nome che ci permetterà di riconoscerlo.

A volte oltre al nome i file hanno anche una **estensione**.

L'estensione permette di identificare da che tipo di programmi il file può essere usato.

Nomefile.estensione

Alcuni Esempi

Un file di testo ha estensione **.txt**

Un file eseguibile ha estensione (in Windows) **.exe**

Un file che contiene un programma C ha estensione **.c**

Seconda parte

Linguaggi di Programmazione
Linguaggio C

Scrivere un Programma Perche?

Per risolvere un problema, per semplificare gestioni, ecc.

Come fare ? Passi:

- Analizzare il problema e trovare un **algoritmo**, cioè la sequenza di passi che porta alla sua soluzione.
- Tradurre l'algoritmo in un **programma** scritto in un linguaggio di programmazione di alto livello.

A volte il programma può essere formato a sua volta da molti programmi.

Linguaggi di programmazione

Ci sono diversi tipi di linguaggi di alto livello:

- **imperativi**, basati sul concetto di istruzione di assegnamento e di uso e modifica del contenuto di strutture dati. Esempi: Fortran, Pascal, C
- **Funzionali**. Basati sul concetto di funzione. Ad esempio il LISP
- **logici**, basati sul concetto di predicato logico. Ad esempio il PROLOG
- **Ad Oggetti**, basati sul concetto di dato come oggetto in grado di comunicare con altri oggetti. Esempi C++, JAVA

Linguaggio C

Il C è un linguaggio di programmazione che richiede un compilatore.

I programmi scritti in C passano normalmente attraverso 6 fasi prima di essere eseguiti:

- Editare
- Preelaborare
- Compilare
- Linkare (collegare)
- Caricare
- Eseguire

Editare

Una volta identificato l'algoritmo per risolvere un problema dobbiamo scriverlo in C.

La prima fase di scrittura di un programma C consiste quindi nello scrivere il programma in un file di testo. A tal scopo si userà un qualunque editor di testo. Ad esempio in LINUX potete usare Emacs. Prima di tutto dobbiamo dare un nome al file che conterrà il nostro programma. Il file DEVE avere estensione .c

Esempio: dalla shell di LINUX lanciare

> **emacs prova.c &**

Nota. Emacs aiuta nella scrittura di un programma in C

Preelaborare

Una volta scritto il programma, ed una volta lanciato il compilatore, prima ancora di compilare il programma

> `cc prova.c` (oppure `gcc prova.c`)

.....viene eseguito il programma detto **preprocessore** che esegue una serie di comandi, specificati nel programma C e noti come **direttive del preprocessore**. Tali comandi consentono, come vedremo piú avanti nel corso operazioni

Come:

- Inclusioni di altri file nel programma
- Sostituzioni di simboli speciali in parti del programma

Compilazione

Una volta scritto il programma, possiamo

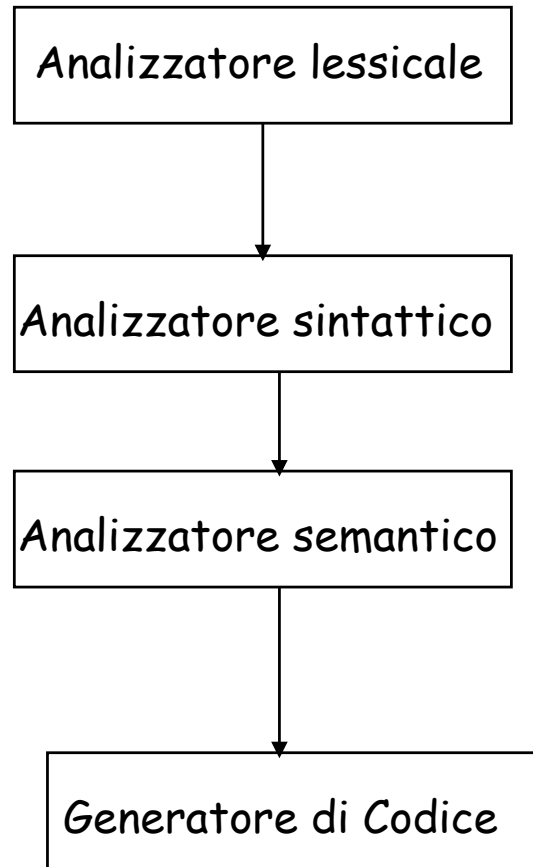
COMPILARE il PROGRAMMA

per produrre il codice oggetto, e il codice eseguibile.

Quando lanciamo il compilatore, dopo aver eseguito il preprocessore si attivano una serie di programmi che consentono di verificare la correttezza del programma da noi scritto e che generano il codice oggetto solo nel caso in cui tutto sia risultato corretto.

Vediamo in generale come è fatto un compilatore....

Struttura di un compilatore



Errori e warnings

Spesso il compilatore si fermerà avendo trovato degli errori sintattici e non. Produrrà sullo schermo una serie di messaggi che vanno considerati come linee guida per riuscire a scovare dove è stato commesso l'errore o gli errori.

A volte l'errore può semplicemente essere un errore sintattico (una parentesi mancante), a volte potrebbe essere un errore del tipo di un riferimento ad un oggetto non dichiarato nel programma.....

Consiglio

Un buon stile di programmazione e molta pratica aiuta ad essere veloci nell'interpretare il compilatore e scovare velocemente gli eventuali errori commessi.

Linkare

Terminata la compilazione, ed una volta andata a buona fine, lo stesso comando

> **cc**

Provvederà ad eseguire il link del codice oggetto con quello di funzioni mancanti.

I programmi scritti in C contengono tipicamente dei riferimenti a funzioni definite altrove (in librerie standard del C, in librerie definite da altri programmi C). Di conseguenza il codice oggetto prodotto dal compilatore conterrà dei "buchi" dovuti a queste parti mancanti.

Il linker si occupa di collegare il codice oggetto con tali oggetti per produrre un codice eseguibile.

Se tutto è andato bene il compilatore creerà un file

a.out

che sarà direttamente eseguibile sul computer e che sarà l'immagine eseguibile del nostro programma.

Loading (Caricamento)

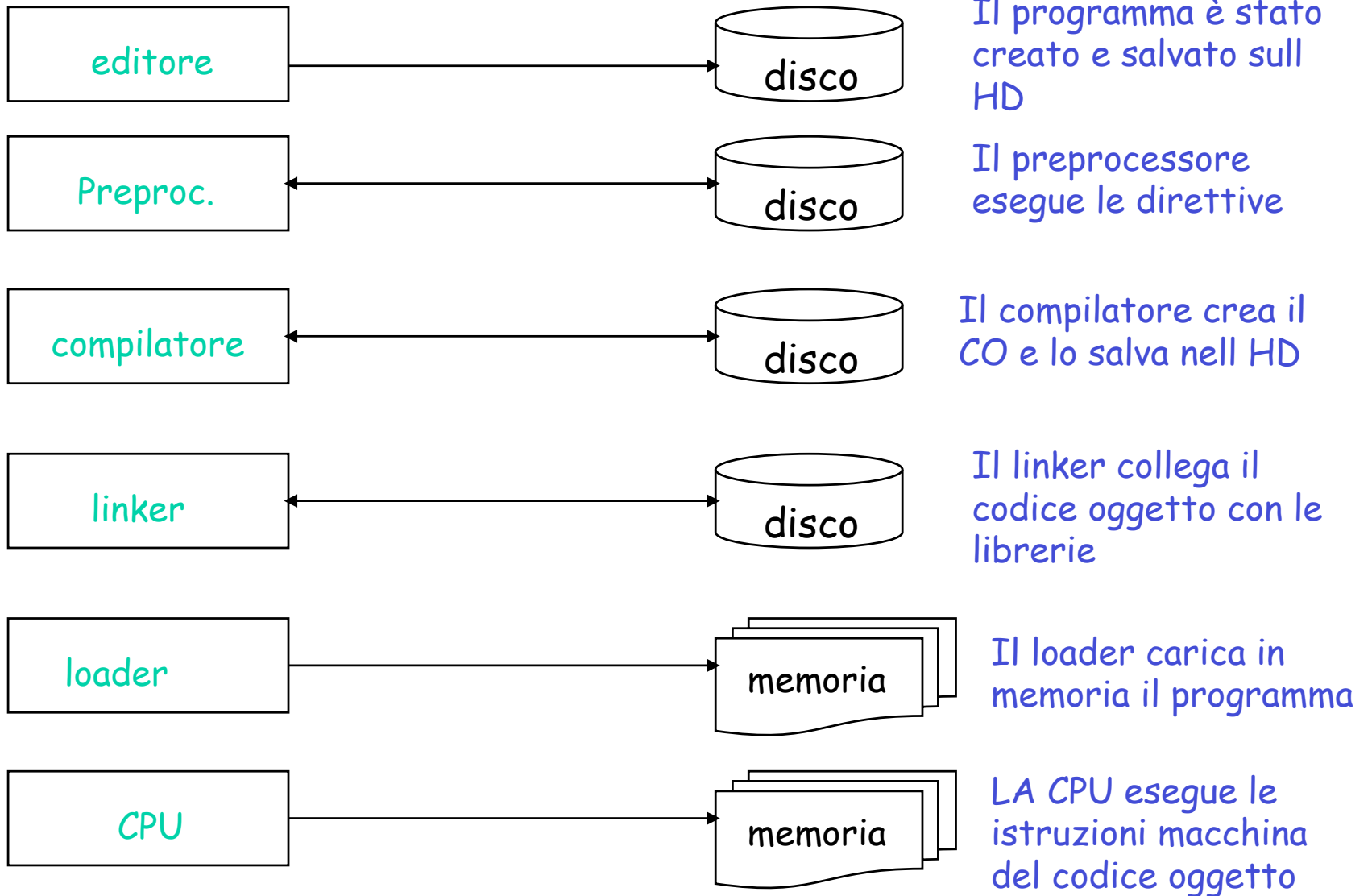
Naturalmente prima di poter essere eseguito un programma deve essere caricato nella memoria principale. A tale compito provvederà il loader.

Esecuzione

Finalmente per poter eseguire il nostro programma dobbiamo digitare il nome del file eseguibile sul prompt

> a.out

Riepilogo



I Programmi comunicano !

Generalmente risolvere un problema vuol dire cercare una procedura che a partire da certi **dati iniziali** sia in grado di produrre dei **dati finali** .

I programmi quindi, essendo delle istruzioni per risolvere un problema, devono poter comunicare con l'esterno.

Devono:

- poter **ricevere dei dati dall'esterno**, ovvero ricevere di dati **in input**
- poter mostrare dei risultati, cioè dare dei **risultati in uscita** o **in output**.

Flussi di comunicazione

In C esistono dei dispositivi che consentono di identificare l'input e l'output. Certe funzioni in C ricevono l'input dal dispositivo chiamato

stdin (standard input)

che generalmente è associato con **la tastiera**.

Mentre i dati in uscita sono inviati al dispositivo di output detto

stdout (standard output)

Che generalmente è associato con lo **schermo**.

Esiste un terzo dispositivo, generalmente anche associato allo schermo che consente di visualizzare i messaggi di errori e che si chiama

stderr (standard error)

Un primo Esempio

```
#include <stdio.h>

int main(void)
{
    printf("Primo Stupido Programma\n");
    return 0;
}
```

Questo programma ha l'effetto di stampare sullo schermo la stringa "Meglio tardi che mai"

Un primo Esempio

Scrivete questo programma in un file con estensione .c (es prova.c)
usando un text editor (esempio emacs)

Compilarlo con

> `cc prova.c` (o `gcc prova.c`)

Errori ????

No !! Ok !!

Allora eseguitelo con il comando

> `a.out`

Analisi

```
#include <stdio.h>
int main(void)
{
    printf("Primo Stupido Esempio\n");
    return 0;
}

#include <stdio.h>
```

`#include <stdio.h>`

Si tratta di una direttiva del preprocessore. Le direttive del processore iniziano con il carattere #.

Il preprocessore trovando la direttiva `include`, include tutto il file `stdio.h`, che è un file che fa parte delle librerie del C e che contiene informazioni relative alla funzione `printf`.

I simboli `<...>` sono usati per indicare che il file `stdio.h` si trova in una directory predefinita dove il preprocessore cercherà per difetto.

L'estensione `.h` di questo file indica che è un file "header", cioè di quei file che vengono di solito messi in testa ad un programma contenendo le definizioni di funzioni usate nel programma .

```
int main(void)
```

Questa è la prima riga della definizione della funzione main.

int e void sono parole riservate del C. Possiamo anticipare che main è una funzione e che la parola riservata int ci sta dicendo che la funzione main **restituisce un numero intero**, mentre il void tra le due parentesi tonde indica che main **non riceve alcun dato in entrata**, ovvero è una funzione priva di parametri.

NOTA BENE

Ogni programma C **DEVE contenere** una funzione che si chiama `main`.
L'esecuzione di ogni programma C inizia **SEMPRE** con l'esecuzione di questa funzione.

Le parentesi {...}

Le parentesi graffe {...} racchiudono il **corpo della funzione main**, cioè delimitano le istruzioni che definiscono la funzione main.

Sono usate per racchiudere il corpo di tutte le funzioni.

Ed inoltre consentono di raggruppare più istruzioni, in modo da considerarle come una sola macro-istruzione.

`printf()`

`printf` è una funzione definita nella libreria `stdio.h` che consente di stampare sullo schermo

`" Primo stupido Esempio\n"`

È una stringa. In C una stringa, cioè una sequenza di caratteri, è definita racchiudendo i caratteri tra i doppi apici. I due caratteri finali, cioè `\n`, rappresentano in C un carattere speciale che consente di andare a capo, cioè saltare ad una nuova linea (newline in inglese)

```
printf(" Primo stupido esempio\n")
```

È una chiamata alla funzione printf. In C il nome di una funzione provoca una chiamata alla funzione. Tra parentesi si possono includere eventuali **parametri**, cioè dati in ingresso alla funzione, che devono essere previsti dalla sua definizione.

```
printf(" Primo stupido esempio\n");
```

Questa è una **ISTRUZIONE**. In particolare questa istruzione è una chiamata alla funzione printf. Si noti il ";" finale. Molte istruzioni in C terminano con un ";". Dimenticare un ";" dove necessario causa un errore riconosciuto dal compilatore.

Return 0;

Questa è una istruzione di return. Essa fa sì che la funzione in cui si trova, in questo caso la funzione main, restituisca il valore che la segue, vale a dire 0, al sistema operativo.

Il SO può eventualmente usare questa informazione.

Se una funzione ha un tipo (cioè restituisce un valore), come in questo, sarebbe un errore dimenticare l'istruzione di return. Non metterla causerebbe la segnalazione di un "warning" (avviso) da parte del compilatore.

Un secondo esempio

```
#include <stdio.h>

int main(void)
{
    printf("Meglio tardi");
    printf("che mai");
    printf("\n");
    return 0;
}
```

Un terzo esempio

```
#include <stdio.h>

int main(void)
{
    printf("Meglio tardi\n");
    printf("che mai\n");
    printf("\n");
    return 0;
}
```