18/09/2020

Lezioni caricate su moodle con algoritmi più o meno chiari. Nel finesettimana.

Il linguaggio C è un evoluzione di linguaggi preesistenti, che erano cosviluppati con il c.

Evoluzione del c, con alcune convenzioni, alcune caratteristiche dal 1989 al 1999 => la realease infatti sarà un C-Anno di rilascio.

Noi svilupperemo il C-99.

Il linguaggio era pensato per fare tutto, era una grande innovazione. E ci fecero davvero tutto. Il suo utilizzo si svolge su più livelli della programmazione high and low.

importante al tempo era la portabilità del codice da una macchina all'altra: integrazione o compilazione.

Il C viene classificato come medio alto livello come il FORTRANT

---

confronto C++:

- nel c++ abbiamo obj. oriented e data abstraction
- nella programmazione avanzata si vedrà il c++

COME COLMARE IL BUCO DAL LINGUAGGIO DI ALTO LIVELLO E LA MACCHINA:

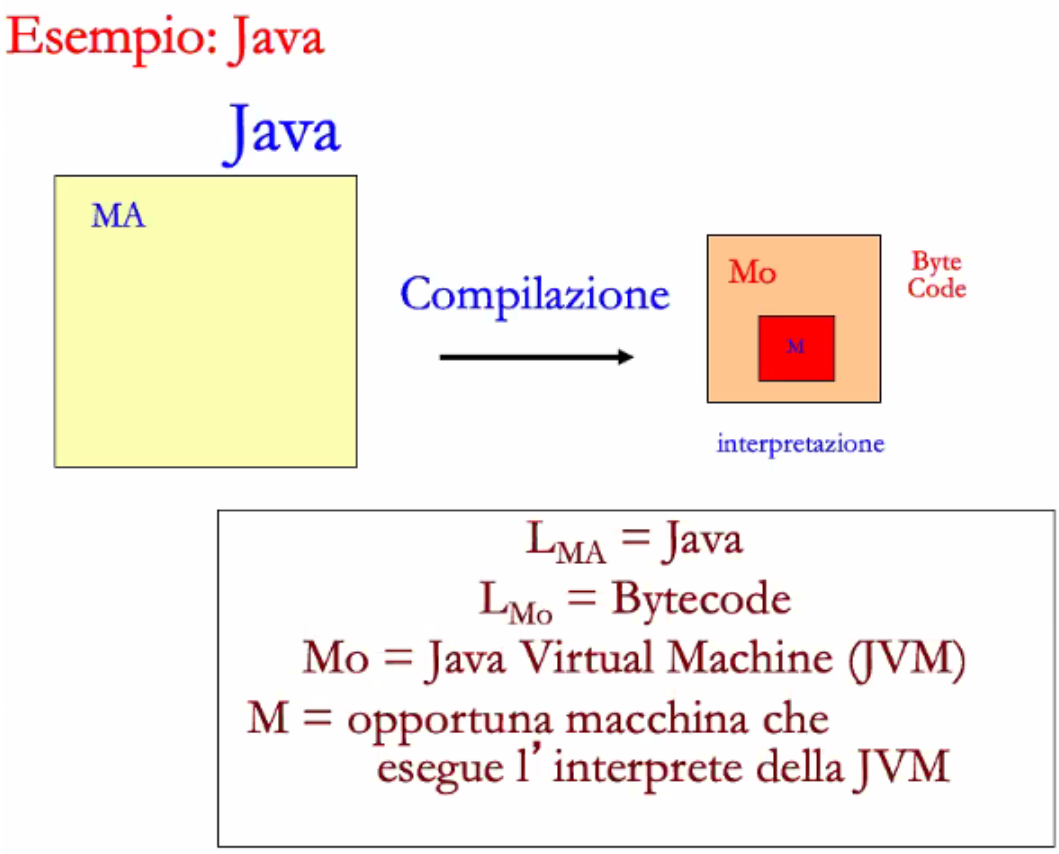
- compilazione: si passa per una o più fasi di traduzione con linguaggi intermedi, o con fasi intermedi che creano assembramenti di codici diversi per arrivare al linguaggio macchina.
    - questo richiede un certo tempo
    - potrebbe creare errori, non andare a buon fine
    - può avere delle fasi all'interno di questa fase per ottimizzare il codice.
- interpretazione: non c'è distanza tra codice e macchina,
    - solo decodifica delle istruzioni, l'esecuzione è più lenta => non c'è ottimizzazione perchè c'è un ciclo di fetch-code-execute
    - ci sono linguaggi che sono prima interpretati e poi compilati.
    - l'interpretazione è semplice e flessibile, sotto c'è la macchina che conosce il set di istruzioni che traduce
    - non c'è bisogno di avere un programma completo per lavorare. Es. stampa variabile => io lo faccio
    - nel C il programma potrebbe non essere eseguito, mentre nel linguaggio interpretato se non viene eseguito è perchè c'è un errore.

ci sono linguaggi macchina che non vengono interpretati: concetto di Macchina Astratta => alla macchina fisica è assegnato un set di istruzioni stesse, Ml => macchina virtuale che ha la capacità di calcolo, capacità di memorizzare, capacità di alloccare risorse: è concettualmente come un calcolatore. Ci sarà un interprete, ci sarà qualcuno che si occupa della gestione della memoria. Ciclo di instruzioni che l'interprete esegue sulla macchina stessa e servono per la traduzione e compilazione.

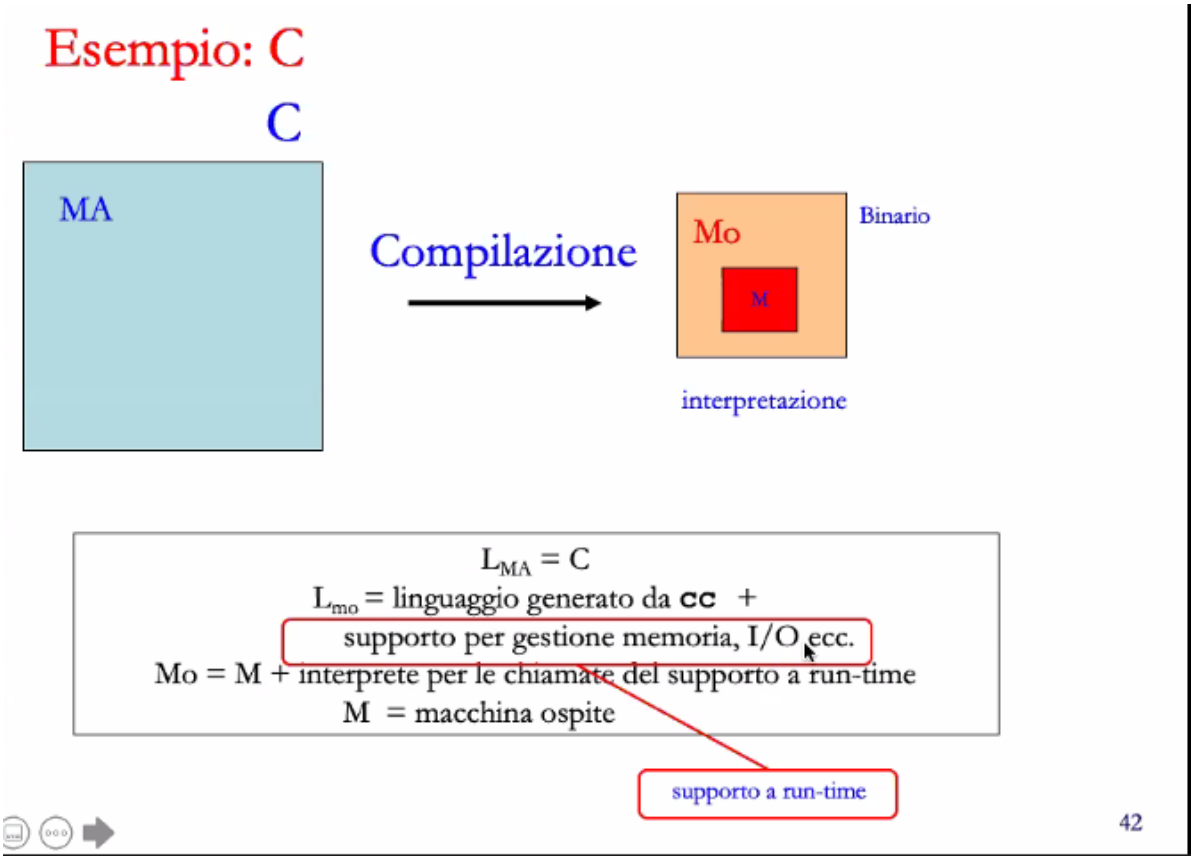
Come posso implementare il linguaggio:

- implementazione interpretativa pura:
  - fetch-execute-code
- implementazione compilativa pura:
  - lontananza fra L e Lo

Esempio java:



Caso del C:

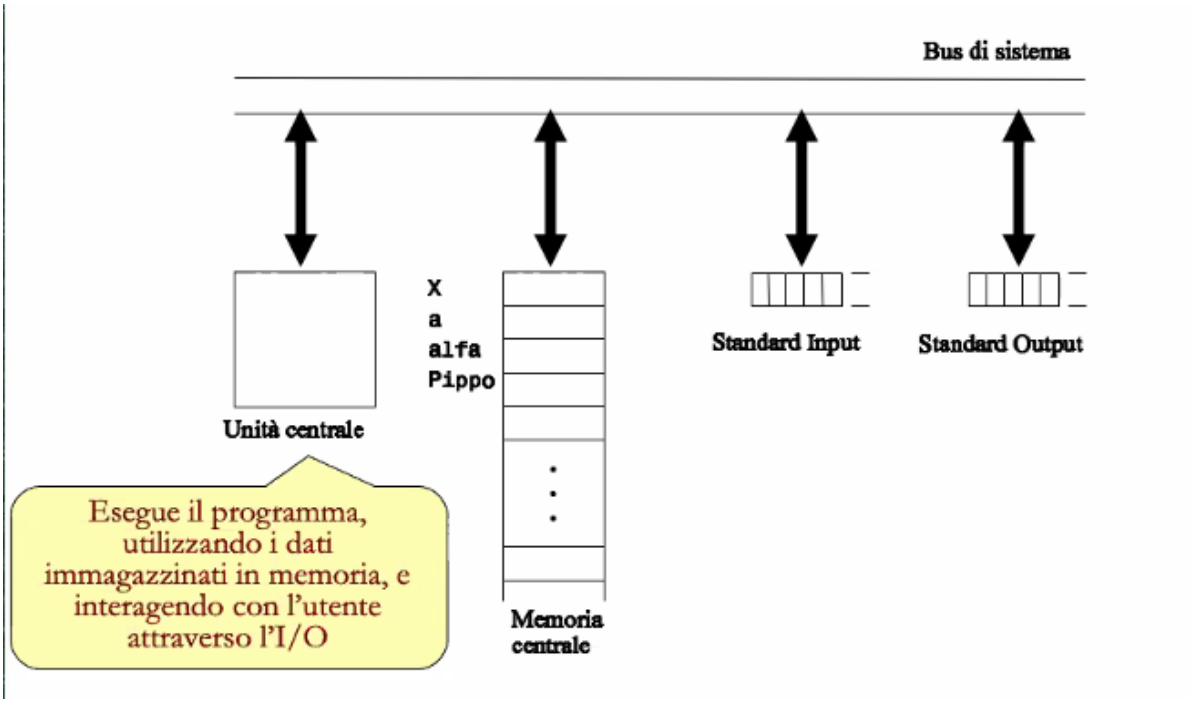


- sono due implementazione del linguaggio in modo statico: quando uno va a implementare il linguaggio che porta dalla macchina astratta al linguaggio ospite
    - poi ci sono tecniche che vanno per la maggiore in base a quello che si fa, si usa

Fase di compilazione/Sviluppo di un programma:

- processo che si usa per sviluppare programmi dai più semplici ai più complessi
    - prima parte di scrittura del programma: editor
    - da adesso partono una serie di programmi che ti permetteranno di arrivare al linguaggio macchina ospite (legge e interpreta il tuo programma):
        - prepocessor: pulizia, trasformazione lessicale
        - compilation
        - linking: collegare altri file, programmi che sono necessari per il funzionamento del programma (#include <stdio.h>) => programmi con funzioni necessarie al funzionamento del programma
        - Execute: programma eseguito da fetch-code-execute
    - potrebbe andare male? si, magari un link non funziona bene, non compilato correttamente, messaggi di errore e status dal compilatore.

ARCHITETTURA MACCHINA ASTRATTA:



- 2 periferiche per questo corso principalmente: stdinput-stdoutput
- bus di sistema astratto che mette in comunicazione questi elementi
- StdIn: tastiera - Staout: output
- Memoria: rispetto alla memoria centrale di basso livello è l'introduzione delle variabili
    - no familiarità con le variabili matematiche
    - organizzare la memoria nella macchina astratta del C è che si vuole denotare un elemento della memoria centrale con un identificatore, che sia leggibile, interpretabile e che sia utile da ricordare da altre funzione.
    - Qualcuno si deve occupare di linkare la variabile a un luogo di memoria.
    - che tipo di dati possiamo memoriazzare:
        - numeri, caratteri, stringhe, composizioni tra queste
        - gli oggetti non semplici come li mappo e li utilizzo?
        - gli oggetti non finiti e non semplici, perchè non ci sono limiti agli oggetti.
    - capacità di denotare spazio in memoria con identificatori, denotare elementi di un insieme infinito
    - come si gestisce questa cosa? tipo dizionario di indirizzi fisici e identificatori? ma come facciamo con più parole, numero arbitrario di numeri e parole
        - non solo uso di variabili statiche, ma anche dinamiche
        - referenziale variabile => conoscere indirizzi tramite identificatore
    - problema di creare un lessico, cioè la lista delle parole ammesse.
        - C case sensitive
        - *Space, Tab & Newline* sono ignorati se non sono parte di una stringa
    - identificatore della variabile, per dare un nome alla variabile
        - come poter dare il nome a tutti questi oggetti
        - keywords: parole riservate da C, 32 chiavi
            - ambiguità non ammessa, perchè errore o warning avvisato dal precompiler
            - conflitto
            - alcuni definiscono un nome della funzione, alcune delle variabile e poi ci sono altri usi

- specifica formale del linguaggio del C, nel testo di Ritchie o in Ansi C
- regole di sintassi e semantica si studieranno
- importanza dell'identificatore:
  - identificare le variabile, distinguere cose
  - caso in cui si possono usare stessi identificatori all'interno del programma, di base comunque si usano identificatori per scopi diversi.
  - la variabile si definisce all'interno della macchina fisica, un contenitore: può reindirizzare => dire la parola e il contenuto della variabile
  - alcune cose assolutamente non matematiche x=x+1 => in matematica porta a una contraddizione
- regole identificatore:
  - successione di lettere-cifre e underscore, nient'altro
  - il primo simbolo dell'identificatore non è mai un numero
  - CASE SENSITIVE!!
- ci sono identificatori che vengono asseriti e poi assegnati nel linking:
  - intestazione main (int ....) {}
- Variabili nel linguaggio sono usate per memorizzare.
- per modificare le variabili
- left- right value =>

- dichiarazione variabili:
  - strumento che da lo spazio nella memoria centrale, con un costrutto => la dichiarazione di una variabile
  - prende questa variabile, li aggiunge alla lista di tutti gli altri identificatori
  - cerca di capire di quanta memoria ha bisogno: solo una? più di una?
    - variabili complesse con più memoria

            - se chiedi troppa memoria per una variabile si può segnlare
            - **int** => parola chiave
                - **int** x; **int** x=3*2;
            - ci devono essere sempre due cose: il tipo della variabile e il nome
                - il tipo serve per capire di quanta memoria tu abbia bisogno => internamente ci sarà una mappatura per immagazzinare una richiesta di una variabile di quel tipo
                - nome della variabile
    - ISTRUZIONI DI I/O:
        - indicazione che serve per il programma per raggiungere tutto l'input e output per il programma, usando api
        - il modo di comunicare viene esteso a altre librerie
            - leggere dati scanf => leggere da stdiN
            - scrivere da printf => scrivere in out
        - & operatori utilizzabile

- gli operatori hanno priorità all'interno di un espressione:
    - ci sono le tabelline di priorità
- associatività => quando due operatori hanno la stessa priorità, come si risolve l'espressione con quell'operatore?
    - bisogna vedere il tipo di associatività da usare
- si dichiarano 4 variabili ABCD => 4*int
    - a=b=c=d=5 => l'assegnamento associa a destra, quindi
    - (a=(b=(c=(d=5)))) => più interna è quella del valore a destra, quindi spiegato il 'si assegna a destra'
    - associatività dell'assegnamento
- dimostrazione netbeans
    - alloccamento variabili => si riservano due parole => cosa c'è li dentro
        - non si sa cosa ci sia li dentro => non si può contrare su cosa ci sia dentro, si vuole sapere? si prova a stampare cosa ci sia dentro
            - valore inderterminato => se il programma girasse di nuovo non si può dire che c'è quel valore
            - per questo si fanno le assegnazioni
        - se commento l'assegnazione del codice
        - uso della '&' => serve per scrivere e leggere
        - i valori di int e float sono indefinite => le variabili vanno inzializzate e questo va fatto programmabilmente.
        -