Assemblatore – Linker – Loader

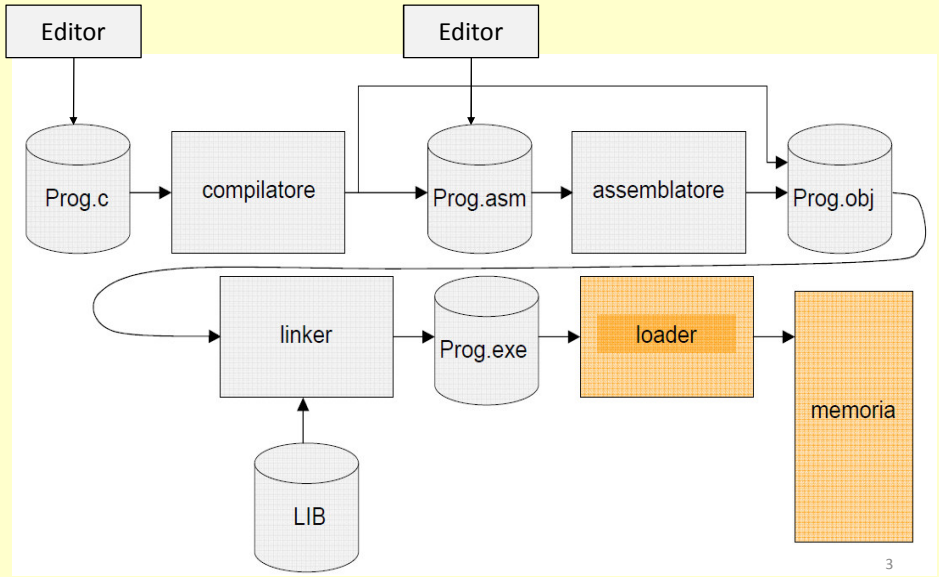
Dal programma sorgente all'eseguibile

- Prima di essere eseguito, un programma attraversa le seguenti fasi:
  - Traduzione
    - Compilazione
    - Assemblaggio
  - Collegamento
  - Caricamento in memoria
- I prodotti delle varie fasi sono ospitati in files
  - .c
  - .asm
  - .obj
  - .exe

High Level Language

Machine Language

Dal programma sorgente all'eseguibile



Compilazione

- Nella prima fase, il programma ad alto livello viene tradotto nel linguaggio assembler utilizzando un apposito programma detto ***compilatore***
- Dopo la fase di compilazione, il programma scritto il linguaggio assembler viene tradotto in linguaggio macchina utilizzando un apposito programma detto ***assemblatore*** (assembler)
- Spesso con il termine compilazione si indica l'intero processo di traduzione da linguaggio ad alto livello a linguaggio macchina (essendo l'assemblatore spesso integrato con il compilatore)

L'assemblatore

Traduce un modulo in assembler in un **file oggetto** (.obj)

- Interpreta ed esegue **le pseudoistruzioni direttive** e **dichiarative**
- legge e traduce tutte **le istruzioni assembly** che costituiscono il programma,
- traduce **in linguaggio macchina** i codici operativi, i dati e le etichette, controllandone la **correttezza sintattica**
- genera **un file oggetto**.

Per poter tradurre in linguaggio macchina le istruzioni di salto, l'assemblatore deve poter determinare gli indirizzi corrispondenti alle etichette usate nei salti; per questo costruisce una ***tabella dei simboli (Symbol Table).***

La***Tabella dei simboli*** contiene le corrispondenze tra nomi delle etichette (simboli) ed gli indirizzi di memoria.

Il file oggetto prodotto dall'assemblatore assume che il modulo parta dalla locazione di memoria 0.

Formato istruzione assembly

Label :	*cod. operat.*	*Operando/i*	*; commento*

Esempio di source code

ILC = Instruction Location Counter

contiene il valore numerico associato alle label

Symbol table

Simbolo	*valore ILC*	*altre informaz.*
……..	……	……..
CICLO	124	- lunghezza del data field associato al simbolo,
BINGO	128	- bit di rilocazione,
		- se il simbolo è accessibile al di fuori della procedura
……..	……..	………..

Opcode table

Codice	*opcode hex.*	*lungh. in byte*	*classe…..*
MOV	5A	3	1
ADD	4A	2	2
JE	1C	2	3
SUB	19	2	2
MUL	D2	2	2
…….	….	…	…..

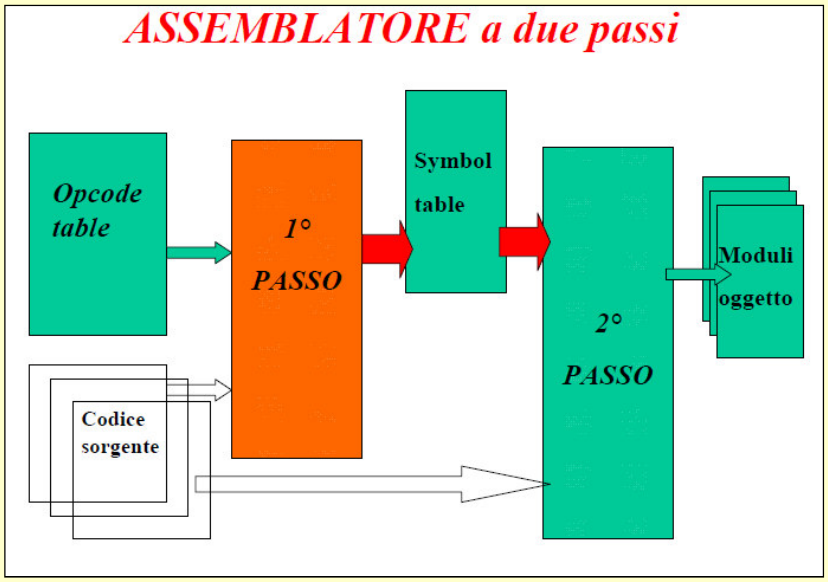
Problema delle forward reference !

ASSEMBLATORE A DUE PASSI:

1 °Passo : - ***legge e traduce tutte le istruzioni assembly*** che costituiscono il programma usando la **tabella dei codici operativi** (**OpCode table**)

- ***individuazione di tutti i nomi (le label)*** *che* compaiono come riferimento simbolico di dati o di istruzioni
- ***creazione di una Symbol Table*** che contiene le label con la loro posizione relativa all'interno del programma

2 °Passo : - ***traduzione dei codici mnemonici delle istruzioni, degli operandi e delle label***, mediante la consultazione della **symbol table** costruita nel 1° passo.



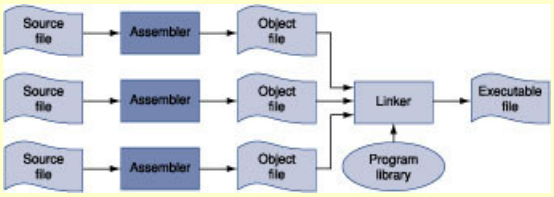
Il linker

Il ***linker*** (link editor o collegatore) ha il compito di **collegare** tra loro vari moduli che compongono lo stesso programma, per esempio:

- Programma sorgente suddiviso in più file che vengono compilati separatamente creando diversi file oggetto
- Utilizzo di funzioni di libreria

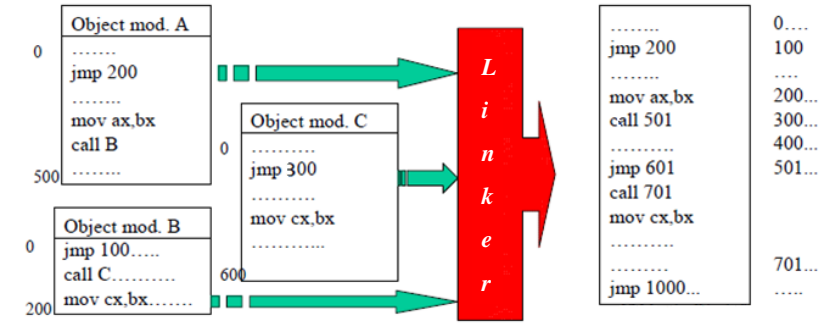
Il **linker** :

- **collega** tra loro i file contenenti il **codice oggetto** dei vari moduli che costituiscono il programma, unendovi anche il codice delle funzioni di libreria utilizzate,
- **e produce un file contenente il codice eseguibile**, che viene memorizzato su disco

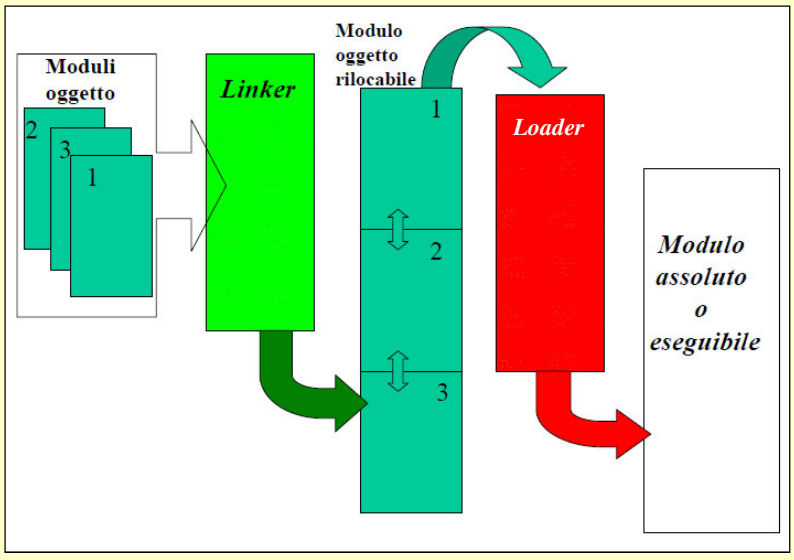


Il linker:

- **crea una tabella con tutti i moduli oggetto e la loro lunghezza**
- **assegna un indirizzo di inizio (di load) ad ogni modulo oggetto**
- **riloca gli spazi di indirizzamento dei vari moduli (Rilocazione) :** in tutte le istruzioni che contengono un indirizzo di memoria somma una "costante di rilocazione" uguale all'indirizzo di inizio del modulo in cui la istruzione è contenuta
- **assegna** nelle istruzioni di chiamata a procedure esterne **l'indirizzo di partenza di queste procedure** (**External Reference)**



Il linker e il loader



Il loader

Programma che **carica nella memoria centrale**, a partire da un determinato indirizzo fisico stabilito dal Sistema Operativo, **il programma binario eseguibile** prodotto dal linker e memorizzato su disco, determinando il legame tra indirizzi logici e fisici (binding).

Il loader si occupa di far partire il programma

- Legge l'intestazione per determinare la dimensione dei segmenti codice e dati
- Crea uno spazio di memoria sufficiente per codice e dati
- Copia istruzioni e dati in memoria
- Copia nello stack i parametri (se presenti) passati al main
- Inizializza i registri e lo stack pointer
- Salta ad una procedura di inizializzazione che copia i parametri nei registri appositi dallo stack e poi invoca la procedura di inizio del programma (main() in C)

Dal programma sorgente all'eseguibile