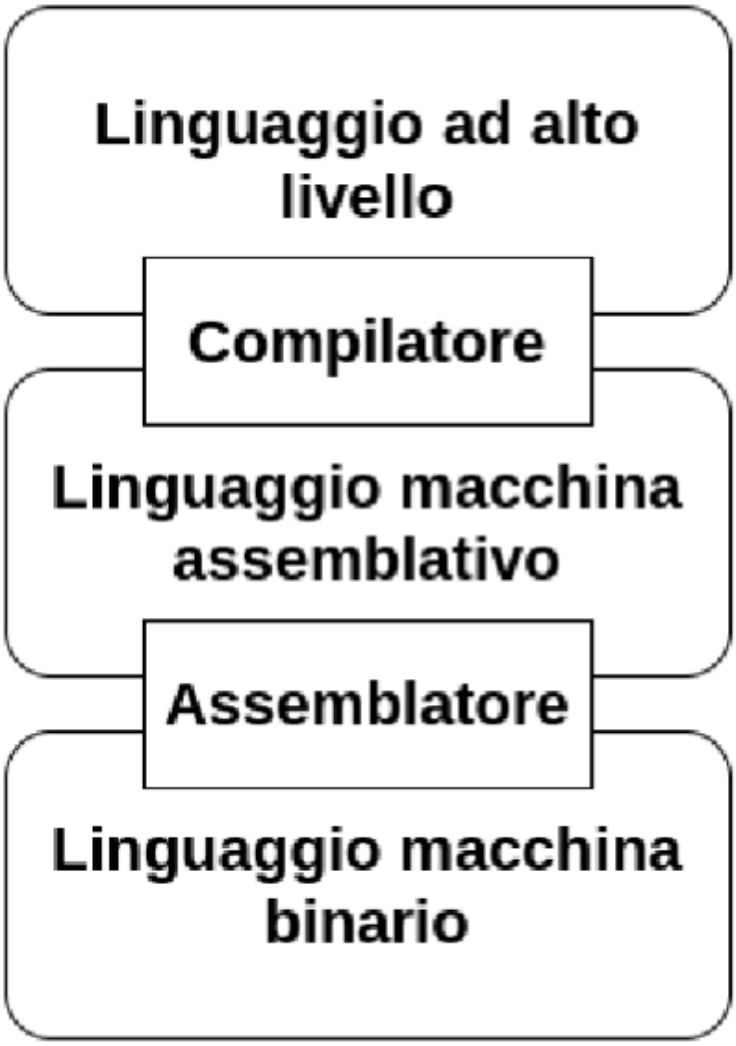
Assembly, come funzionano i linguaggi di programmazione e Sistema Operativo

Il programmatore scrive i programmi in **LINGUAGGIO ASSEMBLATIVO (ASSEMBLY)**

Il programma assemblativo viene tradotto in sequenze binarie dall' **ASSEMBLATORE (ASSEMBLER)**

Il **COMPILATORE** traduce il codice ad alto livello in codice assemblativo



GENERAZIONE PROGRAMMA OGGETTO

1. Il **COMPILATORE trasforma una serie di file sorgenti di linguaggio ad alto livello in file sorgenti assembly**

2. L **'ASSEMBLATORE trasforma una serie di file sorgenti assembly** (generati dal compilatore o scritti direttamente da un programmatore) **in file oggetto**

3. **Il LINKER collega assieme vari file oggetto e file di libreria in un unico programma oggetto** (un file oggetto potrebbe usare dei sottoprogrammi definiti in altri file oggetto o di libreria)

PROCESSO DI COMPILAZIONE DI ASSEMBLY

1. Analisi lessicale: identificazione dei token / keyword
2. analisi sintattica: identifica eventuali errori grammaticali come per esempio l'utilizzo di una variabile non allocata
3. traduzione istruzioni
4. risoluzione delle etichette
5. generazione file oggetto
6. ottimizzazione (passo opzionale)

ASSEMBLATORE A 2 PASSI

1. Primo passo
    1. raccoglie i nomi delle etichette
    2. raccoglie i valori associati alle etichette
    3. raccoglie indirizzi
    4. creazione tabelle simboli
2. Secondo passo
    1. si scorre il codice sorgente sostituendo i simboli con i valori in tabella
    2. crea il codice oggetto

VANTAGGI
è modulare
gestione accurata dei simboli
migliora il debugging

SVANTAGGI
è più lento

LOADER
Loader statico → carica tutto il programma in memoria
Loader dinamico → carico solo le parti del programma che devo usare
Bootstrap loader → carico il sistema operativo in memoria all'avvio del PC

LINKER
è un programma essenziale nel processo di trasformazione in codice binario, questo lo fa generando un file oggetto che collega tutti gli altri file oggetti creati

LIBRERIE
sono contenitori di file che raggruppano oggetti tutti dello stesso tipo e sono create dall'Archiver.
Nel file di libreria sono inserite le informazioni per permettere al Linker di risolvere i riferimenti a nomi esterni in altri programmi.

COMPILATORE

1. Normalmente i programmi vengono scritti in linguaggi ad alto livello molto espressivi
2. Il COMPILATORE trasforma un file sorgente scritto in linguaggio ad alto livello in un file scritto in assembly
3. Il compilatore automatizza molti compiti del programmatore assembly (gestione delle aree di attivazione per esempio)
4. Un compilatore che riorganizza le istruzioni per ottimizzare il codice viene detto OTTIMIZZANTE
5. Un programma ad alto livello può chiamare sottoprogrammi presenti in altri file assembly o scritti in altri linguaggi (il linker gestirà i collegamenti)

DEBUGGER
Il compilatore è in grado di rilevare errori sintattici e nomi sconosciuti nel codice sorgente, ma non errori di programmazione (bug)

Il DEBUGGER è un programma che ci permette di eseguire il programma oggetto ed interrompere la sua esecuzione in qualsiasi istante per valutarne il corretto funzionamento, controllando lo stato della memoria e dei registri

Il debuger può essere eseguito in 2 modalità:
trace mode: il programma viene eseguito passo-passo, interrompendosi dopo ogni istruzione

breakpoint: l'esecuzione del programma si interrompe in punti di osservazione specifici definiti dal programmatore

Passi di esecuzione in Trace mode:
• Si genera un'eccezione al termine dell'esecuzione di ogni istruzione del programma
• Il Debugger viene lanciato come routine di servizio dell'istruzione
• Il programmatore controlla lo stato del programma
• Una volta che il programmatore seleziona il comando per continuare l'esecuzione viene effettuato un rientro dall'interruzione e viene eseguita l'istruzione successiva

Passi di esecuzione in usando i breakpoint:
• Quando il Debugger è in esecuzione, il programmatore può scegliere dei punti di osservazione (breakpoint) dove interrompere il porgramma
• Il Debugger sostituisce e mette da parte le istruzioni in corrispondenza dei breakpoint con speciali interruzioni software (Trap)
• Il programma viene eseguito normalmente fino ad arrivare alla prima Trap, dove l'esecuzione passa al Debugger
• Una volta che il programmatore seleziona il comando per continuare l'esecuzione il Debugger riprende l'esecuzione del programma e reinserisce la trap

SISTEMA OPERATIVO
Il Sistema Operativo (SO) gestisce il coordinamento generale di tutte le attività del calcolatore (esecuzione concorrente di programmi, gestione degli I/O, gestione dell'accesso alla memoria, gestione dell'interfaccia utente, etc.)
• Il SO è formato da un insieme di routine essenziali che risiedono nella memoria centrale e un insieme di programmi di utilità che risiedono su disco e vengono caricati in memoria centrale per essere eseguiti
• **Durante l'inizializzazione del sistema, un processo di avvio (boot-strapping) viene usato per caricare in memoria una porzione iniziale del SO**

Sistemi operativi capaci di eseguire più programmi contemporaneamente sono chiamati concorrenti o multitasking

Multitasking del SO
• Per comunicare con programmi, memoria e periferiche di I/O, il SO fa uso di un sistema di interruzioni
• Il SO fornisce una serie di routine di servizio di interruzioni software per svariati compiti, ognuna con il proprio vettore di interruzione
• Per gestire programmi concorrenti il SO ne divide l'esecuzione in quanti di tempo (time slicing)
• Un contatore lancia un interruzione ogni quanto di tempo τ, e lancia la routine SCHEDULER per scegliere il prossimo programma da eseguire
• I programmi possono trovarsi in 3 stati: RUNNING, RUNNABLE e BLOCKED

I componenti del sistema operativo sono:
Kernel → core del sistema operativo
Interfaccia utente
1. CLI → command line interface
2. GUI graphic user interface
Servizi di sistema → gestiscono e forniscono servizi in background