Catena programmativa

📖 A partire dai file sorgente scritti dal programmatore, è possibile ottiene un programma eseguibile dalla macchina. La struttura che effettua tale processo è detta catena programmativa.

1. Diversi **source files** vengono **tradotti** in linguaggio macchina da un assembler, che si occupa anche di gestire le pseudoistruzioni.
2. Si ottengono così gli **object files** (.obj), che vengono **uniti**, insieme alla libreria che ne definisce i comandi, **da un linker**.
3. Il **linker produce l'eseguibile** ricercato.

PROCESSO DI ASSEMBALGGIO

Per ogni source file, l'assembler costruisce una tabella contenente i simboli utilizzati al suo interno.
L'**assembler traduce** i **codici mnemonici** (per esempio, add, sub, sw, lw...) **in codice**
binario in base al tipo di istruzione. I riferimenti simbolici ai registri vengono associati al loro valore numerico *(per esempio, $t0 → $8 → 01000)*. In caso **un'istruzione non abbia** un **corrispondente** nella tabella generata, essa è detta **unresolved** (non risolta).
L'assembler deve **leggere** il **programma due volte**, dato che l'uso delle labels che indicano parti di codice che possono essere scritte più avanti del riferimento ad essa.
Ogni assemblaggio comincia a partire dall'indirizzo 0, il primo disponibile nel segmento text della memoria (*0x00400000)*. Tale impostazione può causare errori, poiché, in presenza di più codici sorgente, i dati text possono essere sovrascritti.
Pertanto, **i riferimenti assoluti, come gli offset, devono essere rilocati**.
Il **relocation info** è **l'elenco** dei **punti** del codice in **cui** è **necessario modificare** tali **riferimenti**.
Le **etichette globali**, che si riferiscono e servono ad ogni specifico source code, **non** vengono **subito risolte**; tali etichette, che rimangono non risolte, vengono poi gestite direttamente dal linker.

LINKER

📖 Il linker ha il compito di unire i vari file .obj processati dall’assembler, gestisce i riferimenti globali e, mediante i relocation records (contenenti le relocation info), genera l’eseguibile.

LOADER

📖 Il loader ha il compito di leggere l’eseguibile per determinarne la lunghezza ed allocare il necessario spazio in memoria (nel segmento static data e nel segmento text). Successivamente, copia le istruzioni ed i dati nella RAM, inserisce nello stack i parametri da passare, si occupa dell’inizializzazione dei registri e della posizione dello stack pointer.
Il loader richiama una procedura di startup: essa passa i parametri necessari e
richiama il programma principale. Una volta che tale programma restituisce il
controllo alla procedura di startup, essa termina il programma attarverso una syscall
exit.

DEBUGGER

📖 Il debugger è un tool che aiuta a riscontrare gli errori presenti nel codice. Esso permette di eseguire step-by-step un programma e di controllare in ogni
momento i valori delle variabili ed i risultati delle espressioni.
E’ possibile stabilire dei breakpoint, ovvero un’interruzione dell’esecuzione in punti
predefiniti del programma, e dei watchpoint, ovvero breakpoint condizionati dalla
modifica di valori di preselezionate variabili.

Catena programmativa in sintesi

Il **programma** viene **passato** al **COMPILER**, che lo **traduce da** un linguaggio di **alto-livello** (C++/C#) **a** linguaggio **macchina** o assembly. Poi questo **codiceviene passato** all ’**ASSEMBLER** che lo **traduce in** istruzioni **binarie** , generando così un **file OBJ** (object). Il tutto **passa al LINKER** che **collega** il **file OBJ** ad altri file OBJ e file di libreria, **generando** così un **file eseguibile** (.EXE); allafine c’è la **fase** di **LOADER** che **carica** il **programma** sul computere può essere eseguito.