Appunti di Calcolatori Elettronici
Modello di macchina multilivello

Introduzione

Un **calcolatore digitale** è sostanzialmente una macchina in grado di risolvere dei *problemi* eseguendo le *istruzioni* che gli vengono fornite. Una *sequenza di istruzioni* che descrive come eseguire un certo *compito* è detta **programma**.

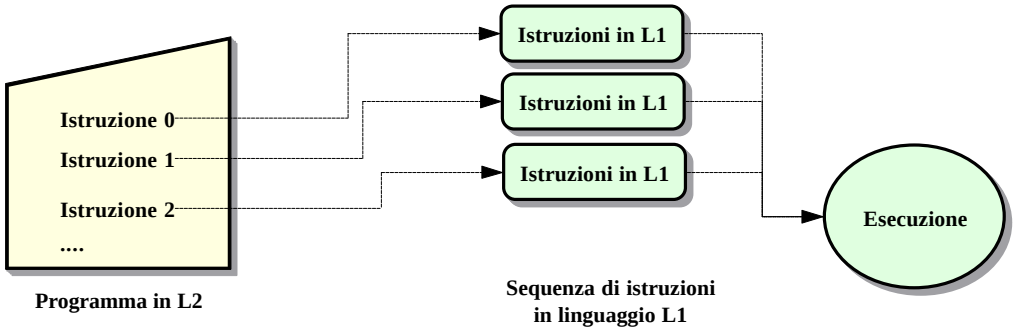
Un calcolatore digitale è dotato di un numero elevatissimo di **circuiti elettronici**: questi sono in grado di riconoscere ed eseguire "direttamente" solo un insieme limitato di **istruzioni semplici**; perciò, tutti i programmi, per poter essere eseguiti, devono essere convertiti (ed è sempre possibile farlo) in sequenze di tali istruzioni semplici ([1]). L'insieme delle istruzioni semplici di un calcolatore forma un *linguaggio* che permette all'utente di comunicare con il calcolatore: questo linguaggio prende il nome di **linguaggio macchina**.

Nel progettare un calcolatore, in base all'uso che se ne intende fare, si tende a semplificare il più possibile le istruzioni del linguaggio macchina, anche e soprattutto per ridurre la complessità ed il costo dell'elettronica da utilizzare. Tuttavia, lo svantaggio di questa elevata semplicità sta nella difficoltà, per l'utente, di utilizzarlo. Il problema viene allora risolto progettando un nuovo insieme di istruzioni, che risulti più conveniente da usare: mentre il linguaggio macchina verrà indicato nel seguito con **L1** (e viene spesso detto **linguaggio built-in**), questo secondo insieme di istruzioni verrà indicato come linguaggio **L2**.

[1] Generalmente, le istruzioni di base sono di 3 tipi fondamentali: somma di 2 numeri, controllo di un numero per vedere se è uguale a 0, spostamento di dati da una parte della memoria del calcolatore ad un'altra.

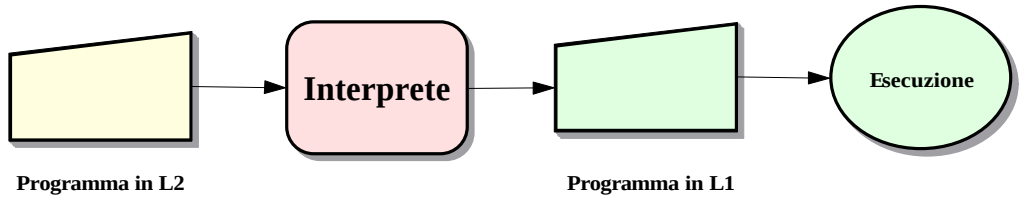
Il linguaggio L2 può essere a sua volta utilizzato in due modi differenti. Infatti, dato che un calcolatore può eseguire sempre e soltanto istruzioni in L1, si tratta di vedere in quali modi viene effettuato il *passaggio* da L2 ad L1:

- il primo metodo consiste nel sostituire ad ogni istruzione in L2 l'equivalente sequenza di istruzioni in L1: con questo metodo, detto di **traduzione**, il calcolatore esegue il nuovo programma in L1 anziché il programma originale in L2;



Metodo di traduzione

- il secondo metodo è invece un po’ più complesso: bisogna infatti scrivere un programma, in linguaggio L1, che riceva in input i programmi in L2 e li esegua esaminando una istruzione dopo l'altra ed eseguendo per ognuna la sequenza equivalente di istruzioni in L1. Questo secondo metodo è detto di **interpretazione** ed il programma in L1 che lo realizza prende il nome di **interprete**.



Metodo di interpretazione

In generale, quindi, *con la traduzione si passa prima al programma equivalente in L1 e poi lo si esegue; al contrario, con l'interpretazione, non viene generato alcun programma intermedio: ogni istruzione in L2 viene esaminata, decodificata e*

immediatamente eseguita. La caratteristica comune è, ovviamente, il fatto per cui il calcolatore esegue solo istruzioni in L1.

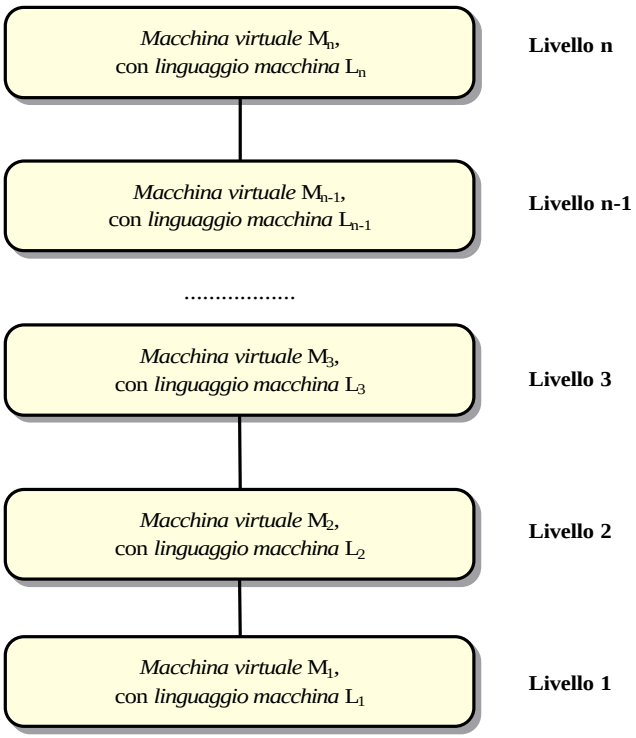
Ovviamente, per rendere pratici i processi di traduzione o di interpretazione, è necessario che i linguaggi L1 ed L2 non differiscano troppo tra loro: tuttavia, questo fa' sì che anche L2 risulti non particolarmente comodo da usare per l'utente. Di qui l'idea (e spesso la necessità) di inventare anche un terzo insieme di istruzioni, questa volta più orientato all'utente che non alla macchina di quanto non avvenisse per L2. Indicheremo questo linguaggio con **L3**.

In generale, questa invenzione di una *intera serie di linguaggi*, ognuno più conveniente del suo predecessore, può proseguire teoricamente all'infinito; nella realtà, il processo si arresta quando si ottiene un linguaggio adatto alle proprie esigenze. Ogni linguaggio usa il suo predecessore come base, per cui è possibile considerare un calcolatore che usa questa tecnica come una serie di **strati** (o *livelli*), posti uno sopra l'altro e corrispondenti ciascuno ad un dato linguaggio. Evidentemente, il linguaggio di livello più basso è il più semplice ma anche il più scomodo (si tratta del *linguaggio macchina*), mentre invece quello di livello più alto è il più complicato ma anche più "familiare" per l'utente.

Linguaggi, livelli e macchine virtuali

Ogni macchina è dotata di un proprio **linguaggio macchina** (L1), cioè l'insieme di tutte le istruzioni che essa può eseguire "direttamente". In questo senso possiamo dire che *una macchina definisce un linguaggio*. Vale però anche il viceversa, nel senso che *ogni linguaggio definisce una macchina*, intesa come quella macchina che può eseguire direttamente tutti i programmi scritti in quel linguaggio. Chiaramente, a linguaggi complessi (quelli *ad alto livello*, per esempio) corrisponderanno macchine complesse e costose, ma questo non impedisce di pensare che tali macchine possano comunque esistere.

Possiamo allora visualizzare un calcolatore dotato di **N livelli** (cioè di N linguaggi) come N differenti **macchine VIRTUALI**, ognuna relativa ad un livello. Nel seguito, useremo i termini "livello" e "macchina virtuale" in modo intercambiabile.



Schematizzazione di una macchina a più livelli

Solo quei programmi scritti in linguaggio L1 potranno essere eseguiti direttamente dai circuiti elettronici di una macchina. Ogni programma che sia scritto in un linguaggio diverso da L1 dovrà subire o una serie di *traduzioni* fino ad arrivare ad un programma in L1 oppure dovrà essere *interpretato* da un interprete di livello inferiore, fino ad un interprete di livello L1. Chiaramente, però, una persona che scrive programmi per la macchina virtuale di livello N non si deve preoccupare degli interpreti e dei traduttori sottostanti: la struttura della macchina garantisce che questi programmi saranno in qualche modo eseguiti.

La struttura a livelli delle macchine odierne

Nei calcolatori moderni si possono generalmente individuare i seguenti livelli:

- il livello più basso (**livello 0**, detto anche **livello della logica digitale**) è quello dell' **hardware**, ossia dei circuiti elettrici ed elettronici di cui il calcolatore si

compone. Tali dispositivi eseguono direttamente i programmi scritti nel linguaggio macchina del livello superiore, cioè il livello 1 ([2]);

- il livello successivo (**livello 1**) è propriamente il livello del *linguaggio macchina*: a partire da questo livello si introduce il concetto di *programma* come sequenza di istruzioni da eseguire; in particolare, nel livello 1 c'è un programma, detto **microprogramma**, che si occupa di interpretare le istruzioni del livello 2. Per questo motivo, il livello 1 viene detto **livello della microprogrammazione**. Da notare che, per definizione stessa di tale livello, non possono esistere due calcolatori aventi lo stesso livello 1: ci possono essere, e ci sono, similitudini, ma mai l'uguaglianza;
- il **livello 2** è noto come **livello della macchina standard**. Non tutti i calcolatori dispongono del livello 1: in questi casi, i programmi di livello 2 vengono eseguiti dai circuiti di livello 0 senza traduzioni o interpretazioni intermedie;
- al di sopra del livello 2 c'è il **livello 3**, che presenta alcune particolari caratteristiche: intanto, esso si differenzia dal livello 2 in quanto fornisce un insieme di nuove istruzioni, una organizzazione della memoria differente, la capacità di eseguire più programmi in parallelo ed altro; tuttavia, gran parte delle istruzioni del livello 3 compaiono anche nel livello 2. Ecco perchè potremmo dire che si tratta di un **livello ibrido**. Le nuove capacità aggiunte a livello 3 vengono eseguite da un particolare interprete, il quale fa operare il livello 2: si tratta del **sistema operativo**. In altre parole, le istruzioni del livello 3 identiche a quelle del livello 2 vengono eseguite dal livello 2, ossia vengono interpretate direttamente dal microprogramma (livello 1) corrispondente; le altre istruzioni vengono invece interpretate dal sistema operativo. Il livello 3 è noto perciò come **livello del sistema operativo**;
- mentre tra i livelli finora esaminati ci sono fondamentalmente notevoli affinità, il passaggio dal livello 3 al **livello 4** è molto più brusco e netto. Lo scopo dei primi livelli (da 0 a 3) non è certo un utilizzo diretto da parte del programmatore medio, bensì il funzionamento dei *traduttori* e degli *interpreti* che supportano i livelli superiori. Al contrario, tali livelli superiori sono concepiti per un uso diretto da parte del programmatore, il quale li utilizza per

[2] Ci sarebbe in verità anche un livello più basso del livello 0, il cosiddetto **livello di dispositivo** che però riguarda il campo della ingegneria elettronica e non ci interessa.

la risoluzione dei propri problemi. La differenza forse più evidente è la seguente: mentre i linguaggi dei livelli 1, 2 e 3 sono costituiti solo da numeri (ottimi per le macchine ma pessimi per le persone), i linguaggi dal livello 4 in su cominciano a contenere *stringhe* ed *abbreviazioni* sicuramente più congeniali per l'uomo. Il livello 4 è noto come **livello del linguaggio Assembler**: si tratta fondamentalmente di una *forma simbolica* di uno dei linguaggi sottostanti. In pratica, questo livello consente di scrivere programmi formalmente analoghi a quelli dei livelli inferiori, ma comunque meno ostici per l’utente. I programmi del livello 4 vengono prima tradotti nei linguaggi di livello 1, 2 e 3 e poi interpretati per l'esecuzione. I programmi dediti alla traduzione sono gli **assemblatori**;

- al di sopra del livello 4 c'è il **livello dei linguaggi simbolici ad alto livello**, ossia di linguaggi molto vicini al linguaggio naturale dell'uomo. Programmi scritti in questi linguaggi vengono solitamente tradotti in programmi di livello 4 o 3 dai cosiddetti **compilatori**.

Evoluzione delle macchine a più livelli

Esaminiamo velocemente lo sviluppo storico delle macchine a più livelli.

I primi **calcolatori digitali** (1940) possedevano soltanto 2 livelli: il livello di *macchina standard*, in cui era fatta tutta la programmazione, e quello della *logica digitale*, che eseguiva i programmi.

Nel 1951 si passò a calcolatori a 3 livelli, al fine soprattutto di semplificare l'hardware: ogni macchina aveva adesso un *interprete* la cui funzione era quella di eseguire i programmi del linguaggio della macchina standard tramite la loro interpretazione.

Nel giro di pochi anni si passò al livello degli *assemblatori* e dei *compilatori*, allo scopo di facilitare il compito dei programmatori.

Agli inizi, i calcolatori erano un qualcosa su cui i programmatori potevano e dovevano operare personalmente per far funzionare i propri programmi. Questo implicava che essi dovessero conoscere a fondo la macchina e la soluzione di eventuali problemi. Di conseguenza, era più il tempo in cui si cercava di individuare i guasti che non il tempo dedicato alle esecuzioni vere e proprie dei programmi.

Dopo 10 anni (1960), si tentò di ridurre questa perdita di tempo automatizzando il lavoro degli operatori: venne introdotto un programma, detto **sistema operativo**, il quale, una volta ricevuti i programmi in input dal programmatore, si occupava di leggerli ed eseguirli. La sofisticazione dei sistemi operativi fu molto rapida: nuove istruzioni, opzioni e caratteristiche furono aggiunte al livello della macchina standard, finché non costituirono un nuovo livello, quello appunto del sistema operativo.

I primi sistemi operativi non facevano altro che leggere i *pacchi di schede* forniti dai programmatori e stampavano gli output su una stampante: questo tipo di organizzazione era conosciuta come sistema **bacth** (o **a lotti**) ed era generalmente piuttosto lenta. Più tardi, invece, furono sviluppate delle versioni dei sistemi operativi che permettevano a diversi programmatori di comunicare direttamente con il calcolatore. In questo tipo di sistemi, venivano usati dei cavi (generalmente di tipo "telefonico") per collegare i terminali al calcolatore centrale. In questo modo, il programmatore poteva inviare i propri programmi ed osservare i risultati (in tempi molto rapidi) da qualunque luogo. Questi sistemi furono e sono tuttora chiamati **sistemi time-sharing**.

Hardware, software e firmware

Come più volte sottolineato, i programmi scritti nel linguaggio macchina (livello 1) di un calcolatore possono essere direttamente eseguiti dai circuiti elettronici (livello 0) del calcolatore stesso, senza l'intervento di alcun interprete o intermediario. Questi circuiti elettronici, insieme a quelli che costituiscono la memoria ed i dispositivi di I/O, costituiscono il cosiddetto **hardware** del calcolatore.

Il cosiddetto **software** è invece l'insieme degli algoritmi e delle loro rappresentazioni per il calcolatore, ossia i **programmi**. Tali programmi possono essere rappresentati su vari supporti (schede perforate, cassette magnetiche, film fotografici o tanti altri mezzi), ma la vera essenza del software è l'insieme di istruzioni che costituiscono i programmi e non certo i mezzi fisici su cui sono registrati.

Una specie di "forma intermedia" tra hardware e software è costituita dal cosiddetto **firmware**: si tratta del software che viene "incorporato" nei dispositivi elettronici digitali durante la loro costruzione. Si ricorre al firmware quando si

prevede di cambiare i programmi molto raramente o addirittura mai, oppure anche quando è importante che un dato programma non venga perso, ad esempio per cadute di corrente. Nella maggior parte dei calcolatori, i microprogrammi sono firmware.

Equivalenza tra hardware e software

Il concetto di **equivalenza** tra hardware e software esprime semplicemente il fatto che *ogni operazione eseguita dal software può essere costruita direttamente nell'hardware, così come ogni istruzione eseguita dall'hardware può essere simulata nel software*.

La decisione se mettere certe funzioni nel software o nell'hardware è basilare nel progetto dei calcolatori ed è influenzata da alcuni importanti fattori, come il costo, la velocità, l'affidabilità, la frequenza prevista negli eventuali cambiamenti ed altro.

Sui primi calcolatori, la distinzione tra hardware e software era chiara e netta: l'hardware eseguiva alcune istruzioni semplici, come la *somma* (ADD) ed il *salto* (JUMP), mentre tutto il resto era programmato esplicitamente via software. Così facendo, quando un programma doveva moltiplicare due numeri, il programmatore doveva scrivere la propria *procedura di moltiplicazione* oppure usarne una *predefinita di libreria*. Quando, però, fu evidente che alcune operazioni venivano eseguite molto frequentemente, si passò a realizzare circuiti specializzati per l'effettuazione di tali operazioni direttamente a livello hardware, il che le rendeva molto più veloci. Ci fu così una ovvia tendenza a "spostare le operazioni verso il basso", ossia appunto verso l'hardware.

Quando invece arrivò l'era della **microprogrammazione** e dei *calcolatori a più livelli*, nacque la tendenza esattamente opposta. Mentre, nei primi calcolatori, l'istruzione ADD veniva eseguita direttamente dall'hardware, sui *calcolatori microprogrammati* essa veniva interpretata da un microprogramma eseguito al livello più basso, il quale seguiva una serie di passi, del tipo seguente: preleva l'istruzione, determinane il tipo, localizza i dati da sommare, prendi i dati dalla memoria, esegui l'addizione e memorizza il risultato. Questo è un classico esempio di una funzione che viene "spostata verso l'alto", dal livello dell'hardware (livello 0) al livello della microprogrammazione (livello 1).

A questo punto, è evidente che, con lo sviluppo delle macchine a più livelli, ogni progettista deve decidere cosa porre ad ogni livello delle proprie macchine; si tratta

perciò di una ulteriore generalizzazione del problema di scegliere se implementare una funzione in hardware o in software.

A quest'ultimo proposito, si ritiene interessante elencare alcune funzioni caratteristiche dei moderni calcolatori che vengono eseguite dall'hardware oppure da microprogrammi ma che inizialmente erano programmate esplicitamente a livello di macchina standard (livello 2):

- istruzioni per la moltiplicazione e la divisione di interi;
- istruzioni per l'aritmetica in virgola mobile;
- istruzioni per l'aritmetica in doppia precisione (che cioè coinvolge numeri con il doppio delle cifre significative);
- istruzioni per la chiamata ed il ritorno delle procedure;
- istruzioni per il conteggio;
- modalità per velocizzare calcoli che coinvolgono vettori;
- modalità per permettere che i programmi vengano collocati in memoria dopo che hanno già iniziato la loro esecuzione (concetto di *rilocazione* dei programmi);
- clock per scandire il tempo di esecuzione dei programmi;
- sistemi di interruzione, che segnalano ad esempio al calcolatore quando una operazione di input o di output è terminata;
- capacità di sospendere un programma e cominciarne un altro (concetto di *commutazione di processi*).

Autore: **Sandro Petrizzelli**
e-mail: sandry@iol.it
sito personale: http://users.iol.it/sandry