Calcolatori Elettronici: introduzione

G. Lettieri

3 Marzo 2017

1 Introduzione e richiami

La Figura 1 mostra l'architettura generale di un calcolatore. Nel corso approfondiremo ogni componente. I principali argomenti di cui parleremo sono:

- protezione;
- interruzioni;
- memoria virtuale.

Questi tre argomenti ci permetteranno di parlare della *multiprogrammazione*: come eseguire "contemporaneamente" più programmi su un sistema che ha un solo processore. Studieremo tutti i dettagli che ci permetteranno di realizzare un (semplificato, ma funzionante) *nucleo di sistema operativo multiprogrammato*. Raffineremo inoltre l'architettura dell'elaboratore parlando di cache, DMA (Direct Memory Access), bus PCI. Infine, esamineremo la struttura interna di un processore moderno in grado di eseguire le istruzioni in parallelo, fuori ordine e speculativamente.

Prima di parlare dei nuovi argomenti, cerchiamo di richiamare e fissare quanto già dovrebbe essere noto.

1.1 La memoria

È un sistema organizzato in "celle", ciascuna composta di un certo numero di bit che è uguale per tutte le celle. Ogni cella ha un indirizzo unico che la identifica.

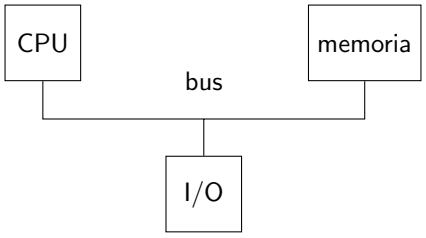

Figura 1: Architettura generale di un calcolatore.

Gli indirizzi sono normalmente dei numeri in sequenza. Il nome "indirizzo" è stato scelto per ricordare gli indirizzi delle case: le celle della memoria sono case, allineate lungo una stessa via, e l'indirizzo è dunque il loro numero civico. Sia gli indirizzi che gli abitanti sono numeri: è importante non confondere le due cose.

Punti (tutti già noti) da tenere a mente:

- ogni cella contiene più precisamente una sequenza di bit; questa può essere sempre interpretata (da chi usa la memoria) come un numero naturale espresso in base due, ma il significato del contenuto di una cella dipende esclusivamente dall'uso che se ne fa;
- la memoria sa eseguire soltanto due tipi di operazioni: lettura e scrittura;
- sa eseguire una sola operazione per volta;
- per eseguire una lettura si deve specificare l'indirizzo della cella che si vuole leggere (la memoria risponde con il contenuto della cella);
- per eseguire una scrittura si deve specificare l'indirizzo della cella che si vuole scrivere, e il nuovo contenuto della cella (la memoria sostituisce il vecchio contenuto con il nuovo);
- la memoria è completamente passiva; non cambia il suo stato se non quando qualcuno ordina una operazione di scrittura;
- *tutte* le celle della memoria contengono *sempre* qualcosa, anche prima di eseguire qualunque scrittura: questo perché ogni bit è memorizzato da un dispositivo che può assumere solo uno tra due stati (0 o 1); all'accensione ci saranno zeri e uno a caso;
- ciascuna operazione richiede un tempo all'incirca costante, indipendentemente da quali altre operazioni sono state richieste precedentemente (memoria ad accesso casuale).

1.2 I/O (senza interruzioni e DMA)

Anche se i dispositivi di ingresso/uscita sono tanti e vari, tutte le possibili interazioni sono trasformante in operazioni di lettura o scrittura. Queste operazioni coinvolgono particolari celle, dette registri o porte di I/O. Ogni dispositivo (tastiera, stampante, ...) sarà collegato al sistema tramite una *interfaccia*, all'interno della quale si trovano i registri. Indipendentemente dall'interfaccia a cui appartiene, ogni registro di I/O ha un indirizzo che lo identifica univocamente, esattamente come le celle della memoria. Le operazioni di lettura e scrittura sui registri sono del tutto simili alle analoghe operazioni sulla memoria. La differenza è che ciascuna operazione può avere effetti collaterali, come causare la stampa di un carattere sulla carta di una stampante. Anche la lettura I/O di

un registro può avere effetti collaterali, come cambiare lo stato interno di una periferica. Per esempio, leggere il codice dell'ultimo tasto premuto dal registro di ingresso della tastiera fa sì che la tastiera smetta di memorizzare quel codice; la prossima volta che il registro verrà letto, la tastiera restituirà il codice del *nuovo* ultimo tasto premuto (se ve ne sono). Ciò è completamente diverso dalla memoria, in cui il contenuto delle celle cambia solo se vi si scrive e due operazioni di lettura consecutive restituiranno sempre lo stesso valore. I dispositivi di I/O, inoltre, possono cambiare il loro stato interno in seguito alla loro interazione con il mondo esterno al calcolatore (per esempio, la tastiera memorizza il codice di un nuovo tasto se qualcuno lo preme). Dall'interno del calcolatore, il mutamento di stato di una periferica è visibile esclusivamente dal fatto che cambia il contenuto dei registri della sua interfaccia. Le interfacce, inoltre, (in assenza dei meccanismi di interruzione e DMA) aspettano passivamente che qualcuno legga il nuovo valore dei registri.

1.3 Il bus (senza DMA)

Il bus connette tra loro i vari dispositivi in modo che ciascuno possa comunicare con ciascun altro. La comunicazione deve avvenire sempre tramite operazioni di lettura o scrittura. Non tutte le possibili comunicazioni sono utilizzate. In assenza di DMA, ci si limita a queste:

1. la CPU legge o scrive sulla memoria;
2. la CPU legge o scrive sull'I/O.

Le operazioni devono essere eseguite una per volta. In ogni operazione è l'indirizzo che permette di capire se è coinvolta la memoria o l'I/O. Il discrimine può essere operato o riservando indirizzi diversi alla memoria e all'I/O, oppure definendo *spazi di indirizzamento* separati (vie diverse, nella metafora degli indirizzi delle case). Nel secondo caso, ogni operazione deve specificare anche lo spazio di indirizzamento coinvolto (memoria o I/O). Ogni operazione è vista da tutti da tutti i dispositivi collegati al bus, e ciascuno di essi deve autonomamente capire se l'operazione è rivolta a lui oppure no. Per farlo, deve confrontare l'indirizzo dell'operazione con gli indirizzi a lui riservati. Se l'operazione non è rivolta a lui, deve ignorarla. Se nessun dispositivo riconosce l'indirizzo, possono succedere cose diverse in dipendenza dal tipo di bus: noi faremo l'ipotesi che le operazioni vengano comunque completate, con le scritture che non hanno effetto e le letture che restituiscono un valore casuale.

1.4 La CPU (senza interruzioni e protezione)

È un sistema che sa eseguire una sequenza di *istruzioni*. Le istruzioni operano sullo *stato* della CPU e/o interagiscono con l'esterno ordinando operazioni di lettura o scrittura sul bus. Lo stato della CPU è contenuto in un insieme di registri. I registri sono funzionalmente simili alle celle della memoria, con la differenza che si trovano all'interno della CPU. Le istruzioni sono codificate in

numeri e contenute in memoria. Uno dei registri della CPU contiene l'indirizzo della prossima istruzione da eseguire (Instruction Pointer, IP). La CPU, da quando la accendiamo a quando la spegniamo, fa solo ed esclusivamente le seguenti cose:

1. preleva dalla memoria l'istruzione puntata dall'IP;
2. la decodifica, la esegue e aggiorna l'IP;
3. torna al punto 1.

(In seguito aggiungeremo qualcosa relativamente alle interruzioni, ma il concetto cambierà molto poco.) Il modo in cui viene aggiornato l'IP al punto 2 dipende dall'istruzione eseguita. Per la maggior parte delle istruzioni, l'IP viene semplicemente incrementato in modo che punti all'istruzione che segue in memoria, secondo l'ordine degli indirizzi. Un programma, dunque, è per lo più una sequenza di azioni da eseguire una dopo l'altra (come la parola "programma" normalmente indica in altri contesti). Alcune istruzioni possono cambiare l'IP per eseguire dei salti. L'istruzione `hlt` ferma il processore (possiamo pensare che non modifichi l'IP).

Punti (tutti già noti) da tenere a mente:

1. tutto ciò che il processore fa, lo fa perché sta prelevando o eseguendo una istruzione;
2. tranne che quando esegue `hlt`, il processore non smette mai di eseguire istruzioni;
3. le istruzioni sono quelle del linguaggio macchina del processore; il processore non è in grado di eseguire nient'altro;
4. tutto quello che vede il processore è il suo stato corrente e l'istruzione da eseguire; il processore non ricorda le istruzioni passate e non si aspetta particolari istruzioni future:
   - tutto ciò che resta di una istruzione alla fine della sua esecuzione è l'effetto che essa ha avuto sullo stato dei registri del processore, delle celle di memoria, o sui registri di I/O;
   - dualmente, tutto ciò che una istruzione vede quando comincia la sua esecuzione è ciò che si trova nei registri del processore, nelle celle di memoria o nei registri di I/O.

1.5 Hardware e software

Quanto esposto sopra è tutto ciò che l'hardware sa fare. Tutto ciò che un calcolatore fa, per quanto complicato possa apparire, deve ridursi a questo. È il software (il programma contenuto in memoria ed eseguito dalla CPU) a orchestrare questi comportamenti elementari in modo da ottenerne di più complessi.

In particolare, le istruzioni che la CPU è in grado di eseguire sono tante ma finite: la CPU sa eseguire solo quelle e nient'altro. Il programmatore deve pensare alle istruzioni come alle possibili mosse degli scacchi, e allo stato del calcolatore (contenuto dei registri della CPU, della memoria e dei registri di I/O) come alla scacchiera: per giocare la partita bisogna sapere quali sono le mosse permesse, e usarle per raggiungere il proprio scopo.

Alcuni trovano difficoltà nel capire se una certa cosa è fatta in hardware o in software. La difficoltà può nascere dal fatto che è comunque l'hardware ad eseguire il software, quindi, in un certo senso, tutto è sempre fatto in hardware. D'altro canto, l'hardware non fa niente se non gli è stato ordinato dal software, quindi tutto sembra essere fatto in software. In generale, una cosa è fatta in software se è richiesto l'intervento consapevole da parte di un programmatore affinché la cosa avvenga. Possiamo usare delle euristiche per aiutarci a decidere in molti casi:

- una cosa che non è completamente svolta da una delle operazioni *elementari* che l'hardware è in grado di eseguire, ed è invece programmata con *almeno due istruzioni della CPU*, va sicuramente considerata come "fatta in software". In particolare, tutto ciò che accade nel sistema durante il prelievo e l'esecuzione di una singola istruzione va considerato come "fatto in hardware" (il programmatore non ha nessun controllo su ciò che avviene nell'hardware mentre è in esecuzione una singola istruzione). Viceversa, una operazione di moltiplicazione programmata usando addizioni, traslazioni e cicli va sicuramente considerata come "fatta in software".

- anche una cosa che si svolge con un'unica istruzione elementare, e che però fa parte di un meccanismo più ampio, può essere considerata come "fatta in software" se il programmatore si deve ricordare di inserire tale istruzione nel suo programma affinchè il meccanismo più ampio funzioni. (come vedremo in seguito, rientra in questo caso l'operazione di risposta ad una richiesta di interruzione).