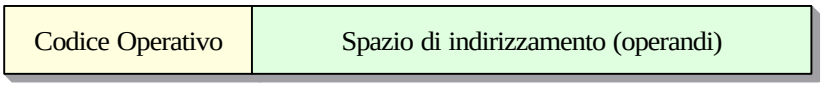
Appunti di Calcolatori Elettronici
Le istruzioni

I formati delle istruzioni

Un **programma** è costituito da una sequenza di **istruzioni**, ognuna delle quali specifica una determinata azione. La prima parte di ogni istruzione prende il nome di **codice operativo** e serve appunto a specificare quale azione si deve eseguire. Successivamente, molte istruzioni contengono i dati da utilizzare oppure indicano la locazione di tali dati (in ogni caso, si parla di **operandi**): ad esempio, una istruzione che confronta due caratteri per vedere se sono uguali, deve specificare quali sono i caratteri da confrontare. Tutto ciò che riguarda la specificazione di dove si trovano gli operandi da usare prende il nome di **indirizzamento**.

Formato generico di una **istruzione**



A seconda della macchina presa in considerazione, le istruzioni possono avere tutte la stessa lunghezza (caso ormai molto raro) oppure lunghezze diverse (caso più frequente). Inoltre, le istruzioni possono essere più corte, uguali o più lunghe di una parola.

Criteri generali di progettazione dei formati delle istruzioni

Dovendo scegliere il formato (o i formati) delle istruzioni di una macchina, i fattori da considerare sono molteplici. Il principio forse più importante da tenere presente è che *le istruzioni brevi sono migliori di quelle più lunghe*, per una serie di motivi:

- un programma costituito da N istruzioni a 16 bit occupa solo la metà dello spazio di memoria di un programma con N istruzioni a 32 bit. Un primo motivo importante riguarda perciò l'occupazione di memoria;
- un secondo motivo è legato al fatto che ogni memoria ha una particolare *velocità di trasferimento* (misurata in **bps**, ossia bit al secondo), determinata dalla tecnologia e dalla progettazione della memoria stessa ([1]), e tale velocità può influenzare la velocità di esecuzione delle istruzioni: se la velocità di trasferimento di una memoria è di T bps e se la lunghezza media delle istruzioni è di R bit, allora la memoria potrà trasportare al massimo T/R istruzioni al secondo verso il processore e questo valore influenzerà la velocità di esecuzione delle istruzioni; in particolare, se il tempo richiesto dal solo processore per eseguire una istruzione (una volta cioè che questa è stata prelevata dalla memoria) è lungo a confronto con il tempo di prelevamento, allora la velocità della memoria è meno importante e quindi lo stesso vale per la lunghezza delle istruzioni; al contrario, con le CPU veloci, la memoria è spesso il *collo di bottiglia*, per cui aumentare il numero di istruzioni prelevate al secondo può essere di importanza fondamentale;
- un altro criterio di progettazione è che le istruzioni devono avere uno "spazio" sufficiente per esprimere tutte le operazioni desiderate: è impossibile avere una macchina che può compiere 2^N istruzioni ma con istruzioni più corte di N bit, proprio perché non ci sarebbe abbastanza spazio nel codice operativo per indicare quale istruzione è necessaria;
- in una macchina, è inoltre auspicabile che la lunghezza della parola sia un multiplo intero della lunghezza di carattere: se il codice del carattere ha k bit,

[1] La *velocità di trasferimento* di una memoria è il numero di bit al secondo che si possono leggere dalla memoria stessa, per cui una memoria veloce può offrire al processore (oppure ad un dispositivo di I/O) più bit al secondo di una memoria lenta.

la lunghezza della parola dovrebbe essere *k, 2k, 3k, 4k* e così via; in caso contrario, sarebbe sprecato dello spazio quando vengono memorizzati dei caratteri. Questi limiti imposti sulla lunghezza della parola dal codice del carattere influiscono evidentemente sulla lunghezza delle istruzioni: infatti, una istruzione può occupare un numero intero di byte o parole oppure un numero intero di istruzioni deve essere contenuto in una parola;

- un ultimo criterio riguarda il numero di bit corrispondenti ad un *campo di indirizzamento*. Per comprendere questo criterio, si ritiene opportuno fare un esempio: consideriamo il progetto di una macchina in cui un carattere abbia lunghezza 8 bit e in cui la memoria centrale debba avere capacità di 2^{16} caratteri; il progettista potrebbe allora scegliere di assegnare indirizzi consecutivi alle unità di 8 bit oppure di 16 bit oppure di 24 bit o di 32 bit o altro. Ad esempio, supponiamo che si debba scegliere tra unità da 8 bit e unità da 32 bit: nel primo caso, si dovrebbe avere una memoria da 2^{16} byte (numerati da 0 a 65535), mentre nel secondo caso di dovrebbe avere una memoria da 2^{14} parole (numerate da 0 a 16383). Scegliendo l'indirizzamento delle parole da 32 bit, ci sarebbe il problema che il programma dovrebbe non solo prelevare le parole contenenti i caratteri, ma anche estrarre il carattere da ciascuna parola, richiedendo perciò un numero maggiore di istruzioni, con conseguente aumento sia dello spazio occupato dal programma sia del tempo di esecuzione; al contrario, l'organizzazione a 8 bit fornirebbe un indirizzo per ogni carattere, facilitando il tutto. D'altra parte, a parità di capacità complessiva di memoria, la scelta dei 32 bit richiederebbe indirizzi a 14 bit, mentre invece la scelta degli 8 bit richiederebbe indirizzi a 16 bit; avere indirizzi più brevi significa istruzioni più brevi, il che a sua volta comporta sia meno spazio occupato dai programmi sia anche meno tempo di prelevamento dalla memoria. Non solo, ma volendo adottare indirizzi da 16 bit anche quando si indirizzano parole da 32 bit, si otterrebbe un aumento della capacità della memoria di un fattore 4 rispetto a quella permessa dall'organizzazione ad 8 bit. Questo esempio aiuta dunque a capire che, per ottenere una migliore **risoluzione di memoria** (ossia la possibilità di indirizzare "pezzi" più piccoli della memoria), bisogna pagare il prezzo di indirizzi più lunghi, che, in generale, significano anche istruzioni più lunghe.

Cenni all’indirizzamento

E’ possibile classificare le istruzioni di un calcolatore a seconda del *numero di operandi* che usano. A tal proposito, è subito importante precisare che un gruppo di registri (numerati) della CPU costituiscono una memoria ad alta velocità e definiscono quindi uno spazio di indirizzamento che si affianca a quello costituito dalla memoria vera e propria. Quindi, una istruzione che somma il contenuto del registro R1 con quello del registro R2 dovrebbe essere classificata come una istruzione con due operandi, dato che essa deve specificare quali registri sommare esattamente nello stesso modo in cui una istruzione che somma due parole di memoria deve specificare quali parole.

Le istruzioni più comuni sono quelle che specificano uno, due o anche tre operandi. Su molte macchine, del resto, si usa un unico operando, in quanto uno speciale registro, detto **accumulatore**, fornisce l’altro operando. Su queste macchine, l’indirizzo specificato è di solito quello di una parola di memoria *m* in cui si trova l’operando. Ad esempio, l’istruzione per l’addizione che specifica l’indirizzo *m* ha il seguente effetto:

```
accumulatore := accumulatore + memoria[m]
```

Le istruzioni di somma a due indirizzi usano invece un indirizzo per la sorgente e l’altro per la destinazione, per cui la sorgente viene sommata alla destinazione:

```
destinazione := destinazione + sorgente
```

Le istruzioni a tre indirizzi specificano infine due sorgenti ed una destinazione:

```
destinazione := sorgente1 + sorgente2
```

Premesse queste considerazioni generali, dobbiamo prestare maggiore attenzione a come i bit del campo di indirizzamento vengono interpretati per trovare l’operando cui fanno riferimento. Il caso più intuitivo è quello in cui essi contengono l’indirizzo di memoria dell’operando, ma ci sono una serie di alternative che esamineremo nei prossimi paragrafi.

Indirizzamento immediato

Il modo più semplice per specificare un operando in una istruzione è che la parte di indirizzamento dell'istruzione contenga effettivamente l'operando stesso invece di un indirizzo o di qualunque altra informazione che descrivano la posizione dell'operando. Si parla in questo caso di **indirizzamento immediato** e **operando immediato**, proprio perché quest'ultimo viene prelevato dalla memoria insieme all'istruzione stessa ed è dunque subito disponibile all'uso.

L'indirizzamento immediato ha dunque l'indubbio pregio di non richiedere un accesso alla memoria aggiuntivo per prelevare l'operando. Ha però anche il difetto di limitare il valore dell'operando ad un numero che sia contenuto nel campo di indirizzamento: se tale campo è lungo N bit, l'operando potrà valore al più 2^N-1.

A livello di curiosità, segnaliamo che le CPU Intel non hanno un modo di indirizzamento per gli operandi immediati, ma hanno un gruppo esteso di istruzioni in cui uno degli operandi è immediato.

Indirizzamento diretto

In questo caso, la posizione dell'operando viene specificata indicando l'indirizzo della parola di memoria che lo contiene. Senza scendere, per ora, nei dettagli di come un calcolatore possa riconoscere gli operandi immediati da quelli diretti, ci limitiamo a dire che esistono due possibili strategie: usare codici operativi differenti oppure usare speciali modi di indirizzamento per ogni tipo di operando.

Indirizzamento tramite registri

L'indirizzamento di registri è concettualmente uguale all'indirizzamento diretto, in quanto il campo di indirizzamento contiene il numero del registro in cui è contenuto l'operando. D'altra parte, abbiamo già osservato che una macchina dotata di registri ha in effetti due spazi di indirizzamento, uno corrispondente appunto ai registri e l'altro alla memoria; di conseguenza, si può pensare ad un indirizzo su tale macchina come composto da due parti: un bit viene usato per indicare se si desidera un registro oppure una parola di memoria, mentre i restanti bit forniscono, rispettivamente, il numero del registro o l'indirizzo della parola di memoria.

E' ovvio, però, che i registri sono molti meno delle parole di memoria, per cui l'uso di un campo di indirizzamento di uguale lunghezza nei due casi sarebbe uno spreco:

vengono quindi generalmente usate istruzioni di diverso formato per *operandi di registro* e *operandi di memoria*. Se ci fosse una istruzione di registro per ogni istruzione che indirizza la memoria, avremmo metà dei codici operativi per gli operandi di registro e metà per gli operandi di memoria; la distinzione tra i codici operativi avverrebbe tramite un bit per designare quale spazio di indirizzamento usare. In alternativa, tale bit potrebbe essere spostato nel campo di indirizzamento.

Le macchine sono progettate con registri per due fondamentali ragioni:

- i registri sono più veloci della memoria centrale;
- essendo pochi, i registri richiedono meno bit per essere identificati.

Il problema viene invece dal numero di registri, che complica non poco la programmazione in quanto bisogna prendere delle decisioni su quali operandi e su quali risultati intermedi devono essere contenuti nei pochi registri a disposizione e su quali invece nella memoria centrale.

Indirizzamento indiretto

Abbiamo detto che l’*indirizzamento diretto* è uno schema in cui l’indirizzo contenuto nell’istruzione specifica quale parola di memoria o quale registro contiene l’operando. Al contrario, l’**indirizzamento indiretto** è uno schema in cui l’indirizzo contenuto nell’istruzione specifica quale parola di memoria o quale registro contiene non l’operando, ma l’indirizzo dell’operando. Ad esempio, consideriamo una istruzione che serve a caricare il registro R1 tramite l’operando indiretto contenuto nella locazione 1510 “puntata” dalla locazione 1000:

- per prima cosa, il contenuto della locazione 1000 viene copiato in un registro interno della CPU;
- tale contenuto (1510) viene assunto non come operando (come avverrebbe nel caso di indirizzamento diretto), ma come indirizzo della locazione contenente effettivamente l’operando: l’operando viene quindi prelevato dalla locazione 1510 e portato in R1. Il contenuto della locazione 1000 viene detto **puntatore**, per evidenti motivi.

Alcune macchine permettono anche l'**indirizzamento indiretto a più livelli**: si tratta, in pratica, del meccanismo per cui un puntatore viene usato per trovare una parola di memoria che punta a sua volta ad un'altra parola e così via.

E' interessante notare che gli indirizzamenti immediato, diretto, indiretto e indiretto a più livelli mostrano una "progressione" nel *numero di accessi alla memoria* (intendo con questa espressioni anche gli accessi a registro):

- l'indirizzamento immediato no richiede accessi alla memoria, dato che l' operando è prelevato insieme all'istruzione;
- l'indirizzamento diretto richiede un accesso alla memoria, per prelevare l'operando;
- l'indirizzamento indiretto richiede due accessi alla memoria, uno per il puntatore e l'altro per l'operando;

l'indirizzamento indiretto a più livelli richiede almeno tre accessi alla memoria, due o più per i puntatori e l'ultimo per l'operando vero e proprio.

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