Livello di architettura dell’insieme d’istruzioni

Aniello Minutolo

Definizione e proprietà del livello ISA

Sommario

1. Definizione e proprietà del livello ISA.
2. Modelli di memoria.
3. Registri.

Il livello ISA

Il livello di architettura dell'insieme d'istruzioni (**ISA**, Instruction Set Architecture) definisce il modello di memoria, quali registri ci sono, quali sono i tipi di dati e d'istruzioni disponibili, e così via.

Il livello **ISA** si trova tra il livello della microarchitettura e il livello del sistema operativo:

- storicamente, fu il primo livello a essere sviluppato, e costituiva infatti l'unico livello presente;
- ancor oggi si usa riferirsi al livello come all'**architettura** di una macchina o altre volte (in modo improprio) come al **linguaggio assemblativo.**

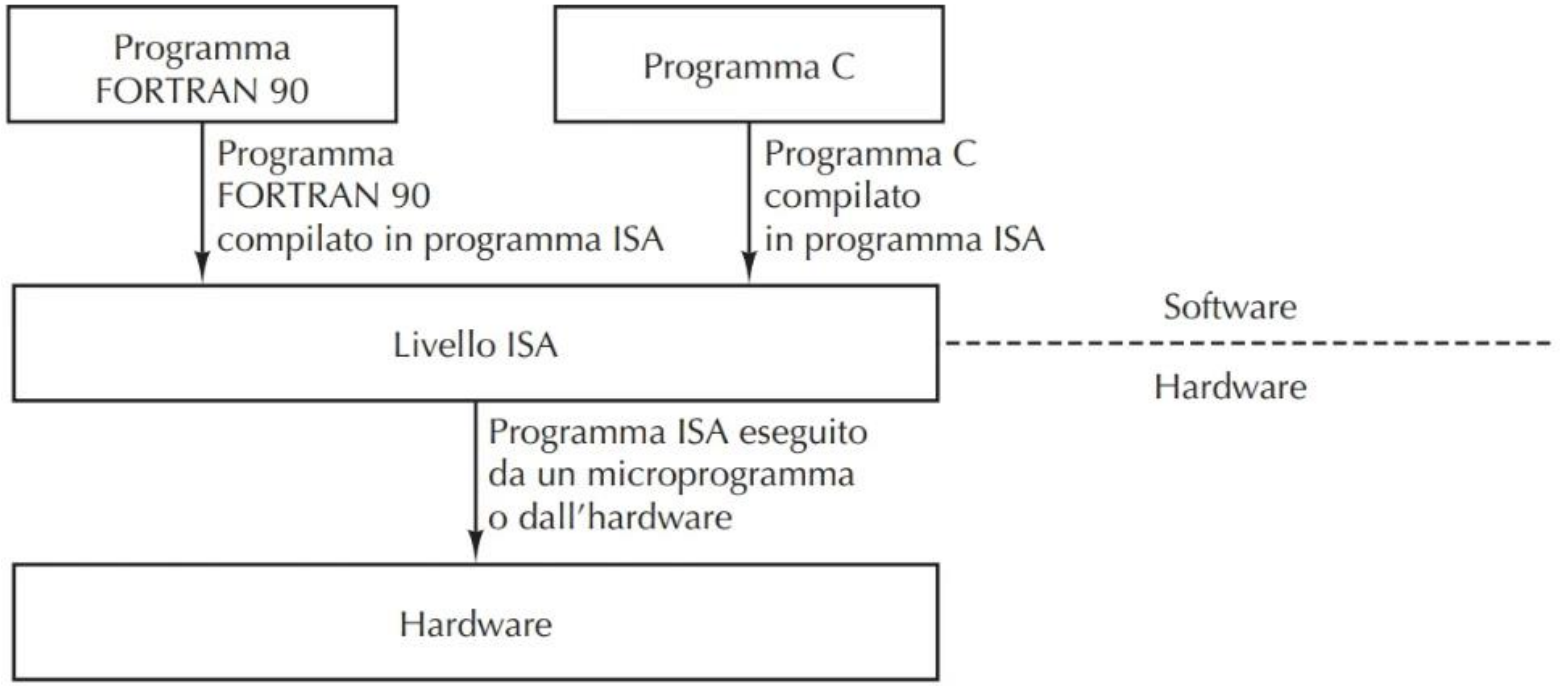
Il livello ISA

Il livello ISA definisce l'aspetto che la macchina assume agli occhi di un programmatore in linguaggio macchina.

Siccome si programma raramente in linguaggio macchina, in realtà il livello **ISA** definisce l'**interfaccia** tra i **compilatori** e l'**hardware**, fungendo da linguaggio comprensibile a entrambi:

- il codice di livello ISA è spesso l'output di un compilatore.

Il livello ISA



Il livello **ISA** definisce l'**interfaccia** tra i **compilatori** e l'**hardware** ed è il linguaggio che entrambi possono comprendere:

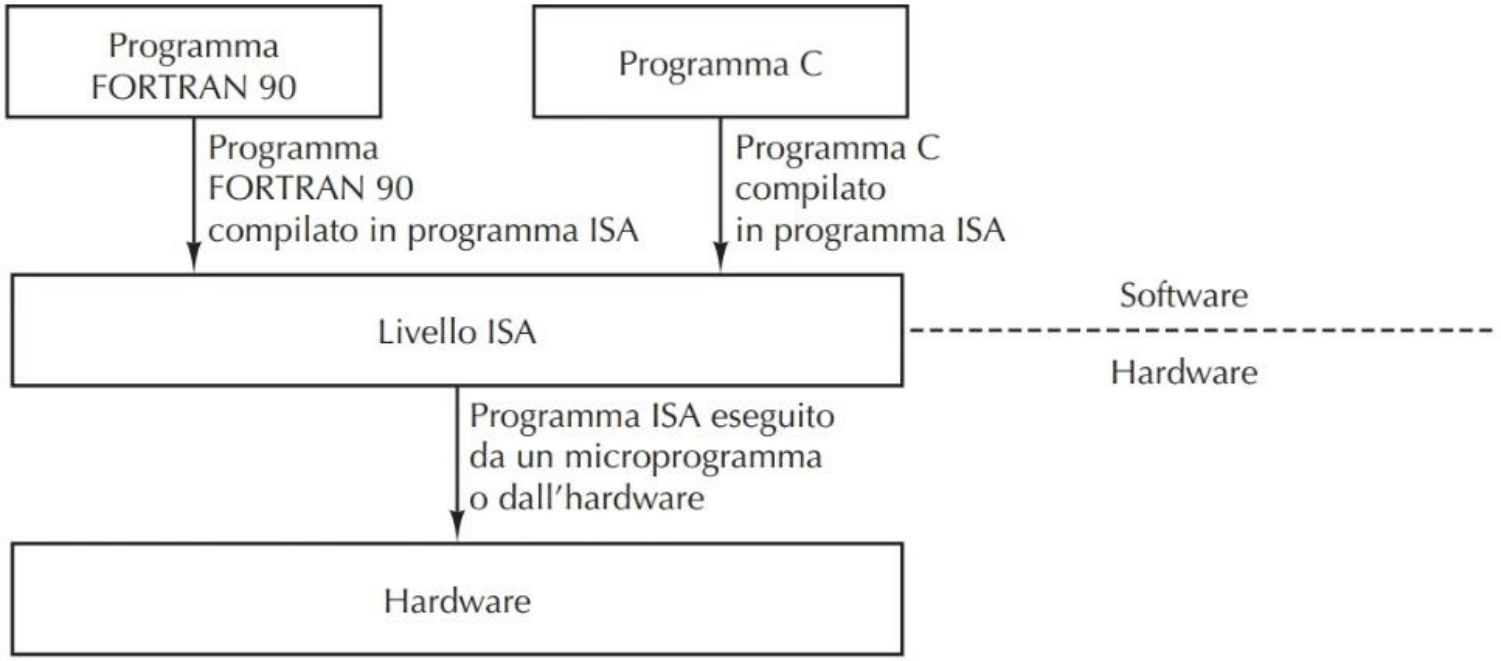
- i computer sono progettati per poter eseguire programmi scritti in C, C++, Java o in altri linguaggi d'alto livello, invece che in uno solo;
- l'hardware viene costruito in modo da poter eseguire direttamente i programmi scritti in una forma intermedia comune, il livello ISA;
- i progettisti di sistemi partono da programmi scritti in vari linguaggi d'alto livello per poi tradurli in programmi di livello ISA.

Il livello ISA

- La fase di progettazione di una nuova macchina richiede la consultazione sia dei progettisti del compilatore, sia dei progettisti dell’hardware, al fine di individuare le caratteristiche desiderate per il livello ISA.
- Un buon livello ISA deve essere efficiente sia per l'implementazione hardware che per la generazione di codice da parte dei compilatori, ed è cruciale per garantire prestazioni ottimali e la compatibilità tra diverse generazioni di hardware.

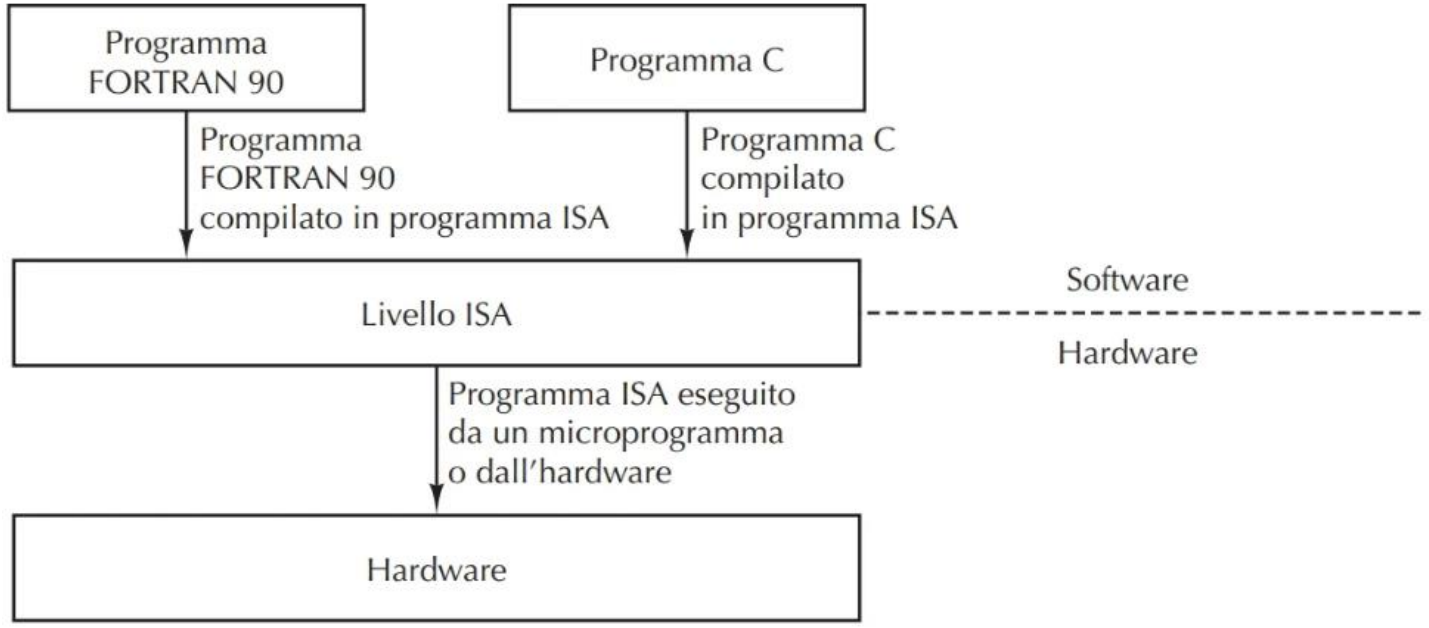
Vantaggi di un livello ISA comune

- Un **ISA** comune permette l'esecuzione di programmi scritti in diversi linguaggi su un unico hardware.
- L'approccio di tradurre linguaggi d'alto livello in **ISA** garantisce prestazioni superiori rispetto all'interpretazione diretta.
- L'**ISA** funge da linguaggio intermedio comprensibile sia dai compilatori che dall'hardware.



Fattori di un buon progetto del livello ISA

- Un **ISA** ben progettato deve essere efficiente da implementare con le tecnologie attuali e future.
- La regolarità e completezza delle istruzioni disponibili favorisce una compilazione pulita e ottimizzata.
- Un **ISA** di qualità può migliorare le prestazioni fino al 25% rispetto a progetti equivalenti ma meno curati.



Retrocompatibilità e vincoli di mercato

Le richieste del mercato hanno reso la **retrocompatibilità** un requisito fondamentale per lo sviluppo di nuove macchine, poiché i clienti non vogliono abbandonare il software esistente:

- i progettisti sono spesso liberi di modificare il livello **hardware** a patto che garantiscano un livello **ISA** retrocompatibile con i modelli precedenti;
- i progettisti possono optare per un progetto di microprogrammazione o per l'esecuzione diretta, o aggiungere pipeline, funzionalità superscalari o qualunque altra cosa a livello hardware, **ammesso che si mantenga la retrocompatibilità con il livello ISA precedente;**
- l'obiettivo è assicurare che i vecchi programmi girino sul nuovo processore e la sfida diventa la progettazione di macchine migliori soggette ai vincoli di retrocompatibilità.

Le richieste del mercato limitano l'innovazione negli **ISA**, rendendo difficile l'introduzione di nuove architetture.

Visibilità del livello ISA

- Il livello ISA è definito come l'aspetto della macchina visibile a un programmatore in linguaggio macchina o a un compilatore.
- Il fatto che la microarchitettura sia o meno microprogrammata (o che disponga di pipeline, o che sia superscalare, e così via) non fa parte del livello ISA, perché non è visibile al progettista del compilatore.
- In realtà questa osservazione non è del tutto corretta, poiché alcune di queste proprietà influenzano le prestazioni, il che è visibile a chi scrive il compilatore.

Istruzioni del livello ISA

La caratteristica principale del livello ISA è l'insieme d'istruzioni macchina che definisce, che specifica ciò che la macchina è in grado di fare.

Tipicamente, il livello ISA comprende sempre alcune istruzioni anche se possono presentarsi in forme diverse:

- STORE e LOAD finalizzate al trasferimento di dati dai registri alla memoria e viceversa.
- MOVE per la copia di dati tra registri.
- Istruzioni aritmetiche, booleane e di confronto dei dati con eventuale salto condizionato dal risultato del confronto.

Documenti di definizione formale

Alcune architetture come **ARM v7** hanno documenti formali che specificano il livello **ISA** in dettaglio:

- i documenti di definizione includono sezioni normative che impongono requisiti e sezioni informative per la comprensione.

Le specifiche **ISA** garantiscono che diverse implementazioni hardware eseguano lo stesso codice con gli stessi risultati.

Esempio: specifiche IA-32

- Intel ha rilasciato le specifiche complete dell'**ISA IA-32** solo verso la fine degli anni '90, dopo iniziali resistenze.
- Il documento di riferimento per il **Core i7** conta 4161 pagine, riflettendo la complessità dell'**ISA x86.**
- Le specifiche **ISA** sono essenziali per garantire la compatibilità tra diverse implementazioni dello stesso set di istruzioni.

Modalità di esecuzione

- Un’altra proprietà importante del livello ISA è che la maggior parte dei processori è dotata di almeno due modalità d’esecuzione: **kernel** per il sistema operativo e **utente** per le applicazioni.
- La modalità **kernel** serve a eseguire il sistema operativo e permette l’esecuzione di tutte le istruzioni.
- La modalità **utente** ha lo scopo di eseguire i programmi applicativi e non consente l’esecuzione di certe istruzioni “delicate” (come quelle che manipolano direttamente la cache).

Modelli di memoria

Organizzazione della memoria

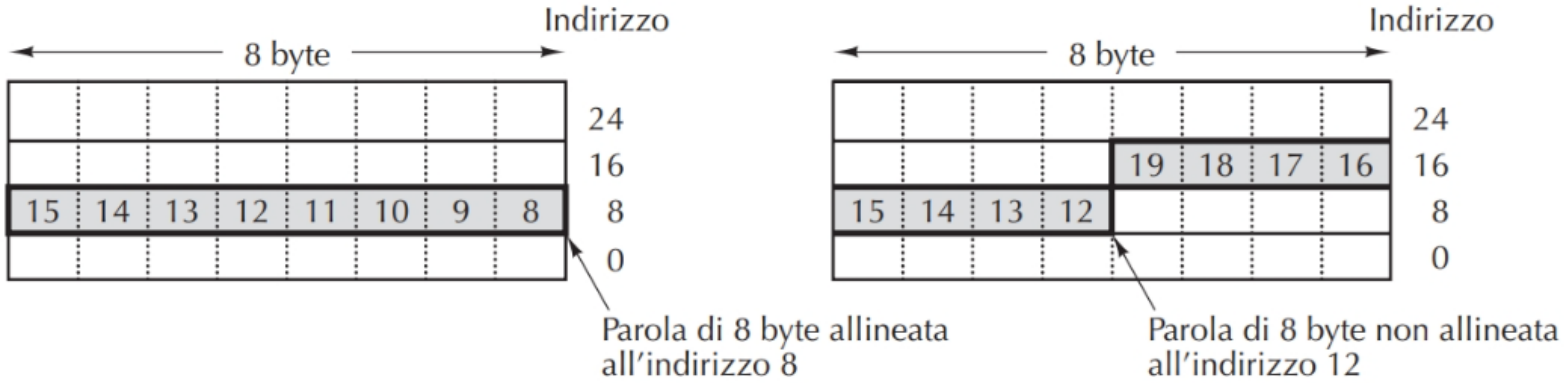
Tutti i computer suddividono la memoria in celle indirizzate in modo consecutivo:

- la dimensione più comune delle celle è di 8 bit (1 byte), ma in passato sono state usate celle di dimensioni diverse, da 1 a 60 bit.

I byte sono spesso raggruppati in parole di 4 byte (32 bit) o di 8 byte (64 bit), con istruzioni specifiche per manipolarle:

- l'uso di byte come unità base è legato al numero di bit usato per la rappresentazione di caratteri ASCII e Unicode.

Allineamento della memoria



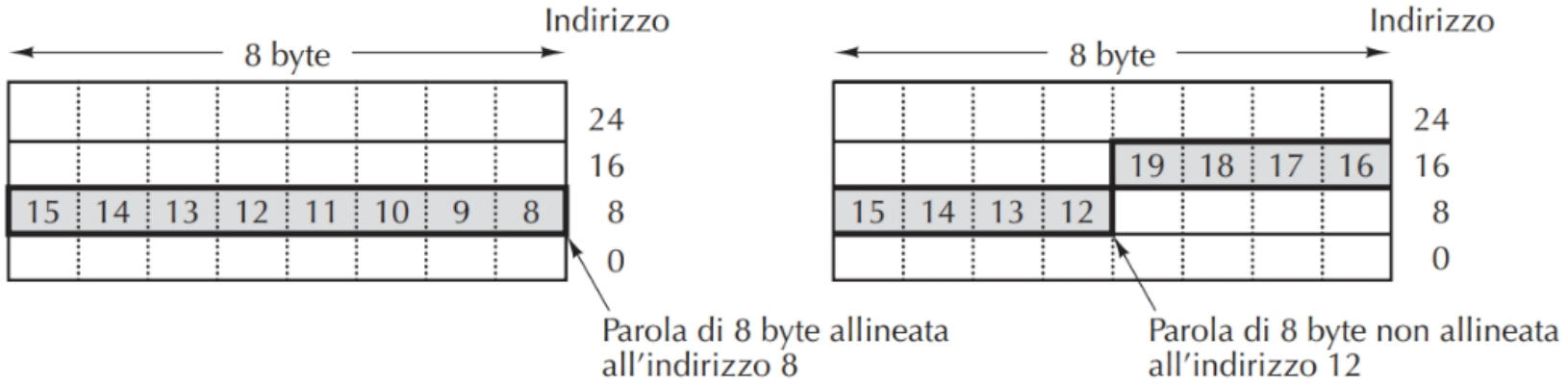
Molte architetture richiedono che le parole siano allineate a indirizzi multipli della loro dimensione

- ad esempio, una parola di 8 byte può cominciare all'indirizzo 0, 8 o 16, ma non all'in-dirizzo 4 o 6.

Allineamento della memoria

Spesso l'allineamento è richiesto perché in tal modo la memoria riesce a funzionare in modo più efficiente, ma può causare complicazioni in caso di accessi non allineati:

- per esempio il Core i7 preleva dalla memoria 8 byte alla volta per mezzo di un'interfaccia DDR3 che supporta solamente accessi allineati ai 64 bit;
- il Core i7 non potrebbe referenziare indirizzi di memoria non allineati, perché l'interfaccia di memoria richiede indirizzi multipli di 8.



Spazi degli indirizzi

Gran parte dei processori a livello ISA dispone di un solo spazio lineare degli indi-rizzi, che si estende dall'indirizzo 0 fino a un certo massimo, generalmente 2^{32} o 2^{64} byte.

Spazi degli indirizzi

Alcune architetture dispongono di spazi degli indirizzi separati per le istruzioni e per i dati, il che rende più complessa l'architettura ma presenta diversi vantaggi:

1. ad esempio, è possibile referenziare 2^{32} byte di programma e 2^{32} byte di dati usando indirizzi di soli 32 bit;
2. poiché le scritture avvengono sempre nello spazio dei dati diviene impossibile sovrascrivere il programma accidentalmente, il che elimina una possibile sorgente di bachi di programma;
3. più difficili gli attacchi dei malware, perché il software "maligno" non può accedere al programma (non può nemmeno indirizzarlo.

Spazi degli indirizzi

Si noti che disporre di spazi degli indirizzi separati per dati e istruzioni non è lo stesso che disporre di una cache separata di primo livello

- nel primo caso il numero totale di indirizzi viene raddoppiato e gli accessi agli indirizzi portano a risultati differenti, a seconda che avvengano nello spazio dei dati o delle istruzioni;
- nel caso della cache separata c'è un solo spazio degli indirizzi, soltanto che cache differenti ne contengono porzioni differenti.

Semantica della memoria

Un ulteriore aspetto del modello di memoria a livello ISA è la semantica della memoria.

La semantica della memoria definisce l'ordine e la visibilità degli accessi in lettura e scrittura:

- in architetture **senza riordino** delle microistruzioni è ragionevole aspettarsi che un’istruzione LOAD, eseguita dopo un’istruzione STORE e sullo stesso indirizzo, restituisca il valore appena memorizzato;
- in architetture **con riordino** delle microistruzioni, il comportamento della memoria può esibire comportamenti inattesi.

Semantica della memoria

I progettisti possono scegliere tra **diverse semantiche,** dal modello più restrittivo a quello più libero:

- i modelli di memoria intermedi bilanciano prestazioni e semplicità, bloccando solo alcune dipendenze critiche.

Una **semantica più semplice** prevede la serializzazione di tutte le richieste di accesso a memoria, così che ciascuna viene completata prima che venga emessa la successiva:

- questa strategia degrada le prestazioni.

Semantica della memoria

Una **semantica più complessa** prevede che non ci sia alcuna garanzia, e che sia il programma a dover eseguire un'apposita istruzione **SYNC** per bloccare l'emissione di nuove operazioni di memoria finché le precedenti non risultino completate:

- questa scelta genera un grosso carico di lavoro supplementare per il compilatore, che deve conoscere il funzionamento della microarchitettura, ma assicura ai progettisti hardware la massima libertà nell'ottimizzazione dell'utilizzo della memoria.

Registri

Visibilità dei registri

In genere i registri visibili a livello microarchitetturale, quali ad esempio il TOS e il MAR, non sono visibili a livello ISA.

Tutti i computer però dispongono di alcuni registri, come il program counter e il puntatore allo stack, che sono visibili anche a livello ISA:

- i loro compito è il controllo dell'esecuzione del programma, il contenimento dei risultati temporanei o altro.

D'altro canto i registri visibili a livello ISA sono sempre visibili a livello della microarchitettura, perché è lì che sono implementati.

Categorie principali dei registri a livello ISA

I registri del livello ISA possono essere divisi in due categorie principali: **registri specializzati** e **registri d’uso generale.**

I **registri specializzati** comprendono il program counter, il puntatore allo stack e altri registri dedicati a funzioni specifiche.

I **registri d’uso generale** sono destinati invece a contenere le variabili locali più importanti e i risultati parziali del calcolo:

- la loro funzione principale è di consentire un accesso rapido a dati usati recentemente (in pratica per evitare accessi in memoria);
- le macchine RISC hanno tipicamente almeno 32 registri d'uso generale per ridurre gli accessi alla memoria, ma nei nuovi progetti la tendenza è di incrementarne il numero.

Registri d'uso generale

I registri d’uso generale di alcune macchine sono del tutto equivalenti e intercambiabili, e la scelta del registro da usare non ha importanza:

- il compilatore può quindi scegliere indifferentemente se usare R1 o R25 per mantenere un risultato temporaneo.

I registri d’uso generale di altre macchine possono invece essere in qualche modo specializzati:

- per esempio, il Core i7 ha un registro chiamato EDX utilizzabile come registro d’uso generale, ma che è anche destinato a ricevere la metà del prodotto in una moltiplicazione e la metà del dividendo in una divisione.

Convenzioni d'uso dei registri

Anche quando i registri d’uso generale sono completamente intercambiabili, è usuale che i sistemi operativi o i compilatori adottino convenzioni nel modo di utilizzarli allo scopo di facilitare l'interoperabilità tra codice generato da diversi compilatori:

- per esempio alcuni registri possono contenere i parametri di chiamata a una procedura e altri essere usati come registri di lavoro.

Convenzioni d'uso dei registri

Laddove esistono convenzioni su come vadano usati i registri di un sistema, è consigliabile che i compilatori e i programmatori del linguaggio assemblativo le rispettino:

- il mancato rispetto delle convenzioni può causare problemi, come la sovrascrittura accidentale di valori importanti.

Registri in modalità kernel

Oltre ai registri del livello ISA visibili ai programmi dell'utente, esiste un numero sostanziale di registri specializzati visibile solo in **modalità kernel** e che controlla cache, memoria, dispositivi di I/O e altre funzionalità hardware della macchina:

- possono essere impiegati solo dal sistema operativo, perciò i compilatori e gli utenti non hanno bisogno di riconoscerli.

Questi registri specializzati sono essenziali per il sistema operativo ma non sono accessibili dalle applicazioni utente.

Il registro di flag

Il registro di **flag**, detto anche **PSW** (Program Status Word), è un registro ibrido tra la modalità kernel e quella utente:

- contiene vari bit di controllo necessari alla CPU, tra cui i più importanti sono i **codici di condizione**, che vengono impostati a ogni ciclo dell'ALU e riflettono lo stato del risultato dell'operazione più recente.

Il registro di flag

Alcuni bit tipici che rappresentano codici di condizione sono:

- N – posto a 1 dopo risultato negativo;
- Z – posto a 1 dopo risultato uguale a zero;
- V – posto a 1 se il risultato ha causato un overflow;
- C – posto a 1 se il risultato ha causato un riporto oltre l'ultimo bit più significativo;
- A – posto a 1 se si è verificato un riporto oltre il terzo bit;
- P – posto a 1 se il risultato è pari (parità nulla).

Il registro di flag

I codici di condizione sono importanti perché sono utilizzati dalle istruzioni di confronto e di salto condizionato:

- per esempio, l'istruzione **CMP** sottrae due operandi e imposta il codice di condizione **Z** in base al risultato della differenza;
- l'istruzione **BEQ** (branch on equal, "salta se uguali") effettua il salto se il bit **Z** ha valore 1.

Il registro di flag

Il registro PSW non contiene soltanto codici di condizione, ma il resto del contenuto varia da macchina a macchina:

- alcuni campi addizionali molto comuni sono la modalità di macchina (cioè, utente o kernel), i bit di traccia (usati nel debugging), il livello di priorità della CPU e lo stato di attivazione degli interrupt.

Spesso il PSW è leggibile in modalità utente, ma alcuni dei suoi campi possono essere scritti solo in modalità kernel (per esempio il bit di modalità kernel/utente).

Bibliografia

Libro di testo

- Andrew S. Tanenbaum e Todd Austin. Architettura dei calcolatori. Un approccio strutturale. 6/ED. Anno 2013. Pearson Italia spa. (Disponibile nella sezione “Biblioteca”)

Fonte argomenti e immagini

- Capitolo 5, Paragrafo 5.1, pp. 353-360