

L'avvento delle architetture RISC

E' di dominio pubblico che i più recenti e potenti microprocessori, il PowerPC sviluppato da Motorola ed IBM, l'Alpha di DEC/Compaq, il chip SPARC ultra della SUN, sono **architetture RISC**.

RISC è una sigla che deriva da "**Reduced Instruction Set Computer**" e contrappone le architetture delle CPU di questo tipo a quelle che si erano venute affermando, come conseguenza della accesa competizione tra i progettisti per offrire agli utenti repertori di istruzioni sempre più ricchi, per una efficiente programmazione in linguaggio assemblativo (linguaggio macchina in forma simbolica).

Dopo l'avvento delle CPU RISC le architetture precedenti sono state rinominate di tipo **CISC** (**Complete Instruction Set Computer**).

La dimostrazione che la velocità di una CPU RISC, a parità di tecnologia di integrazione, poteva essere di oltre un ordine di grandezza superiore a quella di una CPU "single chip" tradizionale, si deve a diversi gruppi di ricerca americani, sia universitari che di industrie.

Retrospettiva storica

Gli studi sulle architetture RISC erano partite dalle difficoltà che si avevano a scrivere sistemi operativi e compilatori per strutture che avevano un linguaggio assemblativo con istruzioni di lunghezza variabile.

Le moderne architetture RISC derivano essenzialmente da tre progetti di ricerca.

Nel **1975** l'**IBM** per prima lanciò il progetto **IBM 801** che non è mai arrivato in produzione ma ha influenzato sia il progetto di ricerca dell'Università di **Berkeley**, che dette vita a due successive versioni di CPU sperimentali, **RISC-I** e **RISC-II**, da cui deriva l'attuale architettura **SPARC** adottata dalla **Sun Microsystems**, che quello poco successivo dell'Università di **Stanford**, il progetto **MIPS**.

Quest'ultimo progetto dette vita alla omonima ditta, **MIPS Computer Company**, che costruì e commercializzò nei primi anni 80, il chip **MIPS R2000**, su cui la **Digital Equipment Corporation (DEC)** basò la sua prima linea di computer RISC, denominati **DECstation** per distinguerli da quelli CISC che erano denominati **VAXstation**.

La famiglia PowerPC

Come di solito avviene, il nome di un chip si riferisce ad una famiglia di processori. In questa il capostipite è il **PowerPC 601** da cui sono derivati il tipo **603**, il **604** ed il **620** che sono evoluzioni del progetto iniziale, nate dalla necessità di ottenere prestazioni sempre più elevate.

Nel caso del pwerPC le maggiori prestazioni sono state ottenute sia attraverso miglioramenti strutturali che attraverso aggiornamenti della tecnologia di integrazione su silicio.

L'architettura **PowerPC** è nata all'inizio degli anni '90 dallo sforzo congiunto di **IBM, Motorola e Apple**.

E' interessante notare che Motorola ed Apple erano in quegli anni impegnati, la prima a produrre la famiglia di processori CISC 68.000, che era giunta sul mercato al 68.040 e la seconda a produrre il Macintosh Quadra che sul 68.040 era basato.

L'adesione della Motorola al programma PowerPC ha comportato che l'estrema evoluzione dell'architettura CISC della casa, il **68.060**, non è mai entrato nel mercato dei processori per elaborazione (general purpose computer), restando confinato nelle applicazioni per controlli industriali.

Qualche notizia strutturale

Il PowerPC è un esempio significativo di architettura RISC perché ai suoi progettisti è stata lasciata piena libertà di interpretare la nuova filosofia strutturale senza ricercare raccordi con precedenti progetti.

- L'architettura si avvale di **due unità di elaborazione separate**. Un'unità di calcolo per l'aritmetica intera (**Integer Unit**) che esegue le operazioni logiche e quelle su operandi interi, mentre l'unità per l'aritmetica in virgola mobile (**Floating Point Unit**) esegue le operazioni su numeri reali.
- Le due unità lavorano coordinate da una unità che gestisce le istruzioni (**Instruction Unit**), prelevandole dalla memoria e distribuendole fra le due unità aritmetiche.
- La struttura essenziale del processore è completata da una **cache memory** che, benché trasparente dal punto di vista logico generale ha una funzione determinante sui tempi di elaborazione del processore.

Essenzialità del set di istruzioni, esempio

Una delle caratteristiche tipiche dell'architettura RISC del PowerPC è che *le uniche istruzioni a poter accedere alla memoria* sono LOAD e STORE, che, a loro volta, usano pochissime modalità di indirizzamento.

Più in generale, **in un processore che adotta la filosofia RISC, una singola istruzione esegue una singola operazione, che richiede un calcolo su operandi contenuti in registri, oppure un accesso alla memoria.**

Il compito di assemblare queste operazioni in sequenze che realizzino operazioni più complesse è lasciato al compilatore. Ciò permette al compilatore di trarre il massimo vantaggio dalle opportunità di incremento delle prestazioni disponibili in un processore dotato di "*pipeline*".

Istruzioni di lunghezza fissa: 32 bit

- Codificata in forma comprensibile alla macchina, **ciascuna istruzione è rappresentato da una configurazione univoca di 32 bit contenuta in una parola.**

Questa configurazione specifica l'operando (o gli operandi) e l'operazione che deve essere eseguita.

- **La restrizione ad una sola parola per istruzione è un'altra caratteristica comune a tutte le architetture RISC.**

Questo semplice formato di istruzione ha anche lo scopo di facilitare la realizzazione di processori dotati di struttura "*pipeline*".

- **E' ovvio che la scelta di questo formato istruzione limita la quantità di informazione che può essere data in un'istruzione.**

E impossibile includere un valore di 32 bit da usare come indirizzo di memoria o come operando immediato. Il numero di bit disponibili per codificare un dato immediato è influenzato da tutto ciò che necessita di essere specificato nella parola dell'istruzione. Vedremo l'effetto di questa restrizione man mano che esamineremo le varie istruzioni e modalità di indirizzamento.

Tipi di istruzioni

Le istruzioni dei PowerPC possono essere suddivise in cinque gruppi:

1. **Istruzioni Load e Store**, che trasferiscono dati fra la memoria principale e i registri della CPU.
2. **Istruzioni aritmetiche e logiche su numeri interi**, che operano su dati presenti nei registri di uso generale da R0 a R31 e su operandi immediati; tali istruzioni non possono accedere direttamente alla memoria.
3. **Istruzioni per il controllo del flusso**, come le istruzioni di salto condizionato, che determinano la sequenza di esecuzione del programma.
4. **Istruzioni in virgola mobile**, che operano su dati nei registri in virgola mobile, che vanno da FR0 a FR31.
5. **Istruzioni di controllo del processore**, necessarie per coordinare le operazioni dei vari componenti del sistema, quali i dispositivi di I/O e le memorie cache.

Modi di indirizzamento

indirizzamento indicizzato immediato: L'indirizzo effettivo dell'operando è la somma del contenuto di un registro indicato nell'istruzione e di uno spiazzamento di 16 bit con segno, X, anch'esso fornito con l'istruzione.

L'operando è specificato nella forma $X(Rsorg)$ e l'indirizzo effettivo viene calcolato come: $A_{eff} = X + [Rsorg]$.

dove Rsorg è uno qualsiasi dei registri di uso generale da R1 a R31.

Se viene usato 0 in luogo di Rsorg, come in X(0), il valore 0 viene usato al posto del contenuto del registro per calcolare A_{eff} .

In questo modo, l'indirizzo effettivo diviene semplicemente X, esteso in segno a 32 bit. L'estensione del segno significa che il bit più significativo del numero di 16 bit viene replicato negli altri 16 bit più significativi.

indirizzamento indicizzato a registro: L'indirizzo effettivo è la somma del contenuto di due registri di uso generale indicati nell'istruzione.

L'indirizzo effettivo è calcolato come: $A_{eff} = [Ri] + [Rj]$

Se viene usato 0 in luogo di Ri, il valore 0 viene usato al posto del contenuto del registro; in questo caso, l'indirizzo effettivo è [Rj].

I codici assembler del PowerPC

Sebbene il repertorio di istruzioni del PowerPC sia ridotto, **ciascuna istruzione esiste in molte varianti, perché le possibili dimensioni degli operandi, che possono essere byte, mezze parole, parole o doppie parole, combinandosi con i modi di indirizzamento ed altre possibilità accessorie fa crescere notevolmente le possibilità a disposizione del programmatore.**

Le caratteristiche specifiche di ciascuna istruzione sono esplicitate nel codice mnemonico.

Due istruzioni di Load o Store, che effettuino la stessa operazione con due modalità d'indirizzamento diverso, hanno un codice mnemonico differente.

CISC v/s RISC

- | | |
|---|---|
| <input type="checkbox"/> Accessi in memoria: solo LOAD & STORE | <input type="checkbox"/> L'accesso alla memoria è consentito a molte istruzioni |
| <input type="checkbox"/> Modi di indirizzamento limitati | <input type="checkbox"/> Molti modi di indirizzamento |
| <input type="checkbox"/> Istruzioni della stessa lunghezza (facilmente decodificabili) | <input type="checkbox"/> Istruzioni di diversa lunghezza |
| <input type="checkbox"/> Le istruzioni fanno operazioni elementari (rapidità di esecuzione e semplicità di controllo) | <input type="checkbox"/> Le istruzioni fanno operazioni elementari ma anche complesse (velocità ridotta, complessità del controllo) |
| <input type="checkbox"/> Le istruzioni lavorano con la memoria e sono lente | <input type="checkbox"/> Le istruzioni lavorano con la memoria e sono lente |
| <input type="checkbox"/> Le istruzioni impegnano un solo ciclo | <input type="checkbox"/> Le istruzioni impegnano più cicli |
| <input type="checkbox"/> Le istruzioni lavorano su registri (quindi sono veloci) | <input type="checkbox"/> Il controllo deve essere micro programmato (e quindi più lento) |
| <input type="checkbox"/> Il controllo può essere cablato (e quindi più veloce) | <input type="checkbox"/> L'esecuzione in pipeline risulta più complessa anche per la lunghezza variabile delle istruzioni |
| <input type="checkbox"/> È possibile adottare una pipeline più lunga nell'esecuzione (e più efficiente) | <input type="checkbox"/> La programmazione in Assembler è più semplice |
| <input type="checkbox"/> Si possono usare molti registri nella CPU | |
| <input type="checkbox"/> Il compilatore può ottimizzare la traduzione di linguaggi ad alto livello | |