Architettura degli elaboratori - lezione 2

Appunti di Davide Vella 2024/2025

Claudio Schifanella
claudio.schifanella@unito.it
Link al moodle :
https://informatica.i-learn.unito.it/course/view.php?id=3106

21/02/2025

Contenuti

11. Note per l'esame

CPU e i suoi componenti

Unità di controllo

Si occupa di capire di che tipo di istruzione si sta per fare e poi (dopo aver settato i bit di controllo in modo opportuno) viene eseguita l'istruzione.

ALU

Circuito combinatorio che si occupa di eseguire le istruzioni mandate dall'unità di controllo.

Registri

Program Counter :

Contiene un indirizzo (NON l'istruzione) della prossima istruzione. Da qui capiamo che il programma che vogliamo eseguire si trova sulla RAM. Il programma arriva alla RAM quando viene eseguito partendo dal disco rigido.

Program register :

Contiene il codice macchine dell'istruzione che stiamo eseguendo.

Controller

Circuito che si occupa di comunicare con una parte fisica. C'é ne uno per la RAM, uno per i display port, GPU... (nell'attuale configurazione e un'architettura specifica (questa detta sopra è intel)). Una volta c'erano due controller, uno chiamato "North bridge" e "South Bridge".

Data path

È l'organizzazione interna

Avvio di un programma

Quando avviamo un programma, allochiamo lo spazio sulla RAM per :

- Il codice
- Lo stack
- L'heap

Bus

Un mezzo di comunicazione, formato da fili elettrici. Si occupa di far comunicare le varie parti hardware del computer (CPU, RAM, disco rigido...). La velocità del bus è data dalla larghezza del bus e dalla velocità del clock. Quelli interni alla bus sono più veloce (generalmente) rispetto a quelli che collegano CPU, RAM...

Esecuzione delle istruzioni

La CPU esegue ogni istruzione del livello 1 (ISA) per mezzo di una serie di passi elementari :

1. Prendi l'istruzione seguente dalla memoria e la mette nel registro delle istruzioni
2. Cambia il program counter per indicare l'istruzione seguente
3. Determina il tipo dell'istruzione appena letta (unità di controllo)
4. Se l'istruzione usa una parola in memoria, determina dove si trova (non avviene nell'architettura RISC-V)
5. Metti la parola, se necessario, in un registro della CPU
6. Esegui l'istruzione (a dipendenza dell'istruzione può essere eseguita tutta dalla ALU o dalla ALU e un altro circuito)
7. Torna al punto 1 e inizia a eseguire l'istruzione successiva
   (Per tutto il corso, si pensa di avere sempre 1 macchina, 1 cpu, 1 core).

Ruolo del software

- Organizzazione a livelli
- Software applicativo
- Software di sistema

Ciclo di vita del software

Esiste prima un programma. Tramite il compilatore diventa un programma assembly. Ora attraverso l'assemblatore diventa linguaggio macchina. Facendo "gcc -S (programma)" otteniamo un file .S, ovvero in assembly (che usa l'istruction register per la CPU del computer che stiamo usando, noi useremo quella di RISC-V).

Prestazioni

Le prestazioni si misurano in :

- Tempo di esecuzione / tempo di risposta : tempo fra l'inizio e il completamento di un programma.
- Throughput o la larghezza di banda : il numero di task eseguiti nell'unità di tempo.
  Queste cose sono influenzate sia dal processore, ma anche dal tipo di task che vogliamo

eseguire. Ancora possiamo vedere :

- Tempo di esecuzione della CPU : il tempo che la CPU utilizza nella computazione richiesta da un certo task, che si divide in :
  - Tempo di CPU utente : tempo effettivo speso dalla CPU sulla computazione della richiesta di un programma
  - Tempo di CPU di sistema : tempo speso dalla CPU per eseguire le funzioni del sistema operativo richieste per l'esecuzione di un programma
- Cloak : dispositivo che genera un segnale (onda quadra) che serve a sincronizzare i componenti.
- Ciclo di clock : detto anche colpo, colpo di clock, periodi di clock
- Periodo/frequenza di clock : durata di un ciclo di clock (250 picosecondi o 4GHz). Architetture diverse, prendono cicli di clock diversi per eseguire una stessa istruzione (CPI - cicli per istruzione). Una frequenza di clock più elevata rende più veloce l'esecuzione di un'istruzione. (ad esempio le istruzioni "load" hanno un tempo di esecuzione più elevato). Per calcolare il tempo di CPU facciamo :

$$\text{Tempo di CPU} = \frac{\text{numero di istruzioni} \times \text{CPI}}{\text{frequenza di clock}}$$

Limiti fisici

Esistono alcuni limiti per la CPU, non possiamo aumentare, ad esempio, la velocità di clock all'infinito. Alcuni limiti sono :

- costi
- fisici
- potenza elettrica

RISC-V (reduced instruction set computer)

Proposto e sviluppato dal 2010 da : Andrew waterman, Yunsup Lee, Krste Asanovic. Gli obbiettivi erano :

- hardware semplice
- compilatori efficienti
- massimizzare costi e prestazioni
- minimizzare il consumo energetico
- **Standard aperto** (molto importante)
  RISC-V si adatta bene sia per microcontrollori si per supercomputer.

ISA

L'ISA è semplice e può essere estesa con delle estensioni :

estensione	descrizione
i	integer
m	
a	
f	
d	
g	
c	

Principi di progettazione

1. La semplicità favorisce la regolarità
2. Minori le dimensioni, maggiore è la velocità
3. Un buon progetto richiede buoni compromessi
   RISC-V ha poche istruzioni e semplici, che però se vengono ben usate posso eseguire tutti i programmi. CISC al contrario ha tante istruzioni, anche complesse, che possono eseguire sì tutti i programmi, ma prendono più tempo.

Nota

RISC-V non è un azienda, ne una CPU, è una struttura di CPU.

Registri e memoria

RISC-V ha la seguente struttura :

- 32 registri per gli interi (es : RV64I ha i registri a 64 bit, per RV32I (quella che viene usata durante il corso) sono a 32 bit). Questi registri si chiamano $x_0, x_1, \ldots, x_{31}$.
- 32 registri per i numeri in virgola mobile
- Registro program counter
- 4096 control status registers
- Memoria centrale (ad esempio, 2^{64} indirizzi nella RV64I, 2^{32} indirizzi nella RV32I) (ogni indirizzo ha n)

Ruoli dei registri

I registri hanno dei ruoli specifici :

- x0 : zero (utili per ottimizzare, molte volte inizializziamo una variabili a 0, questo lo rende più veloce).
- x1 : return address
- ...
- x10-x17 : registri usati per il passaggi di paramenti nelle procedure e valori di ritorno
- x5-x7, x28-x31 : registri temporanei, non salvati in caso di chiamata
- x8-x9, x18-27 : registri ... il contenuto va preservato se utilizzati dalla procedura chiamata

Istruzioni aritmetiche

Tutte le istruzioni aritmetiche hanno esattamente 3 operando, l'ordine di questi operando è fisso. Alcuni esempi :

- a = b + c in C, diventa "add a, b, c" in RISC-V assembly
- a = b - c in C, diventa "sub a, b, c" in RISC-V assembly
  Nota, sopra in realtà non esistono variabili, ma i registri, quindi (ad esempio) :
- a = b + c in C, diventa "add x5, x20, x21" in RISC-V assembly
- a = b - c in C, diventa "sub x5, x20, x21" in RISC-V assembly

Nota

- byte : 8 bit
- half word : 16 bit
- word : 32 bit
- doubleword : 64 bit
-

Domande poste durante la lezione (forse utili per esame)

Note per l'esame

Dal lato pratico, all'esame verrà valutato quanto riusciamo a rendere efficiente il programma (usare meno volte possibile la ram ad esempio e quante più volte i registri).