MIPS & SPIM

Modulo del Corso di Architettura degli Elaboratori

Nicola Paoletti

Università di Camerino
Scuola di Scienze e Tecnologie

10 Aprile 2013
AA 2012/2013

Informazioni utili (1/2)

- **Email:** nicola.paoletti@unicam.it
- **Orario:** Mercoledì, 10:00-11:00
- **Pagina web del corso:**
  `http://www.nicolapaoletti.com/teaching/`
- **Pagina Facebook del corso:**
  `http://www.facebook.com/mipsSpimUNICAM`

Informazioni utili (2/2)

- **Modalità d'esame:** Progetto + Relazione
- **Materiale didattico e d'approfondimento:**
  - Slides e materiale aggiuntivo reperibili sulla pagina web del corso

Obiettivi del corso

- Architettura HW del processore MIPS32
- Set di istruzioni del processore MIPS32
- Il tool SPIM, un simulatore MIPS32
- Valutazione delle prestazioni (*benchmarking*)

Progetti

Tipologie:

- Progetto MIPS/SPIM
- Progetto Benchmarking
- ...

Linee guida:

1. Si può presentare il progetto individualmente o in coppie
2. Copia cartacea + copia digitale della relazione
3. Il progetto scelto va comunicato al Prof. Leonardo Pasini via mail (leonardo.pasini@unicam.it)

Riepilogo

1 Introduzione

Concetti fondamentali (1/2)

Linguaggio macchina

Il linguaggio basato su valori numerici utilizzato dai computer per memorizzare ed eseguire programmi. Alfabeto: $\{0, 1\}$.

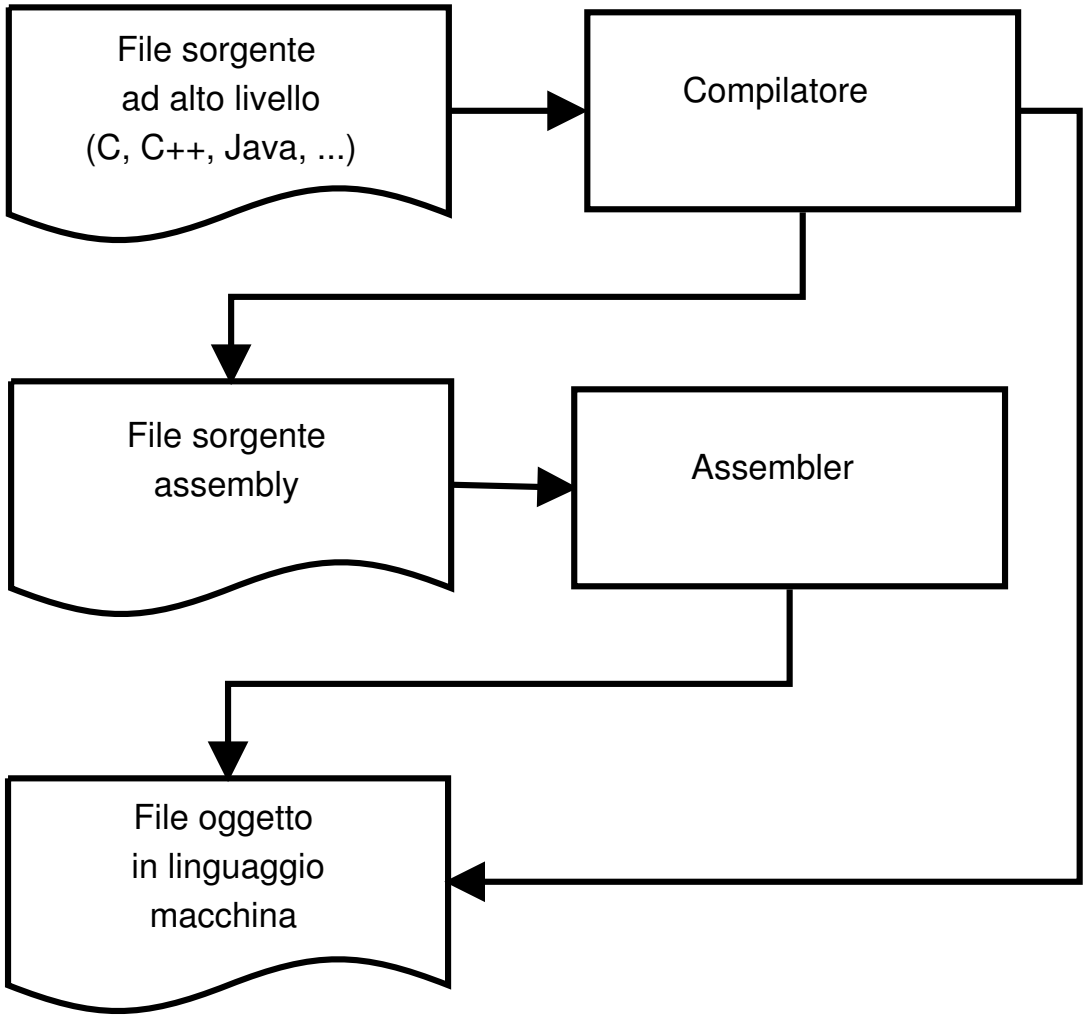
Linguaggio assembly

Rappresentazione simbolica del linguaggio macchina; utilizza simboli invece di numeri per rappresentare istruzioni, registri e dati. Più leggibile rispetto al linguaggio macchina.

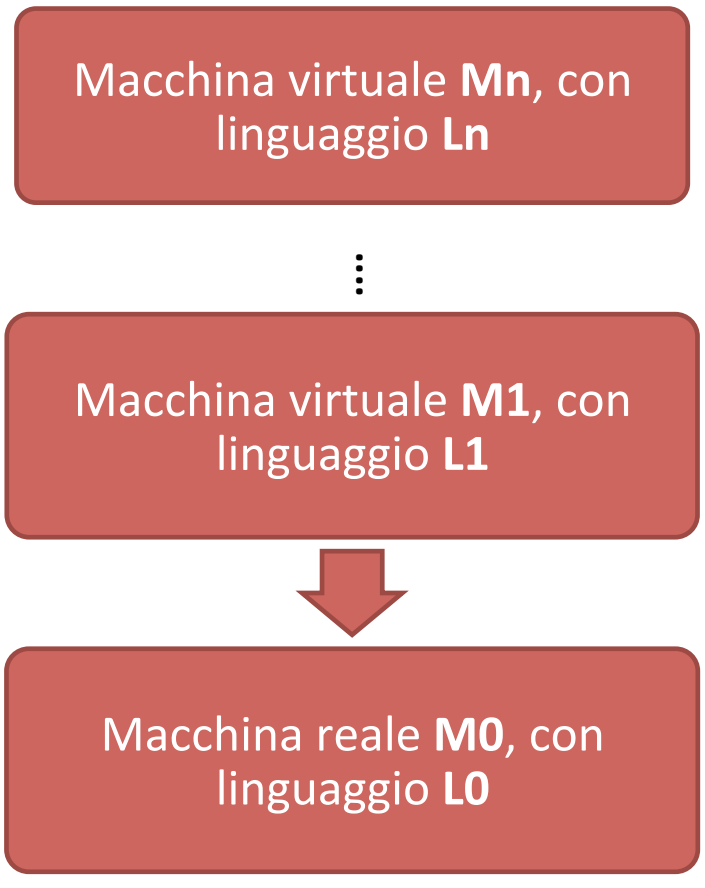
Linguaggio ad alto livello

Astrae i dettagli hardware; si avvicina al linguaggio naturale.

Concetti fondamentali (2/2)



Organizzazione multilivello del calcolatore (1/2)



I programmi in Ln vengono o interpretati da un interprete a livello n-1, o vengono tradotti al linguaggio macchina di Mn-1

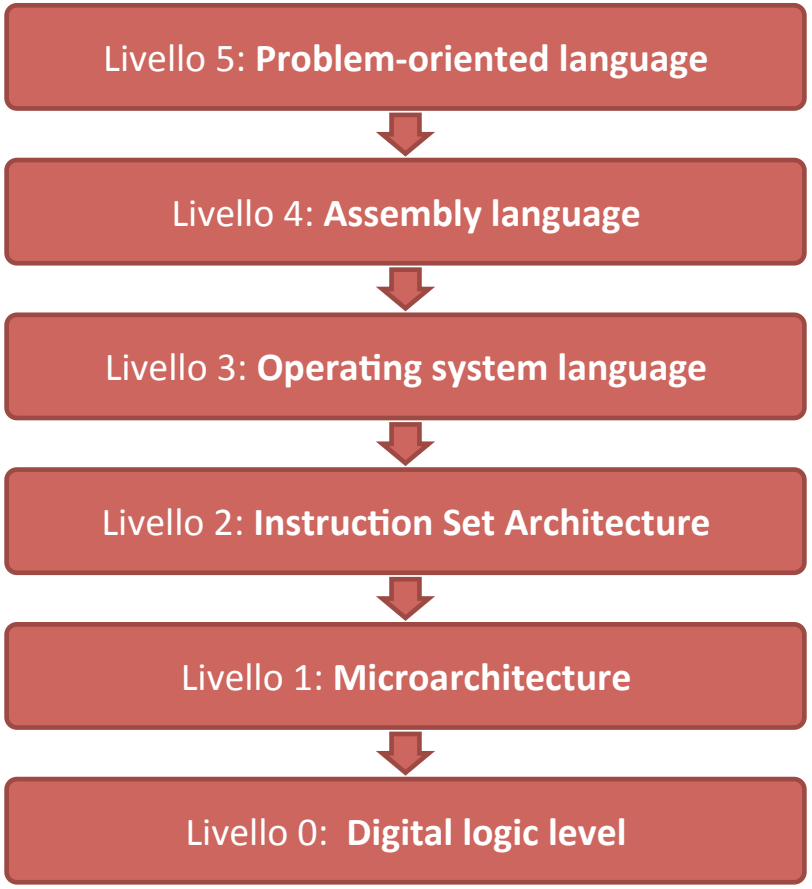
I programmi in L1 vengono o interpretati da un interprete a livello 0, o vengono tradotti al linguaggio macchina di M0

I programmi in L0 possono essere eseguiti direttamente dall'hardware (linguaggio macchina)

User-oriented

Machine-oriented

Organizzazione multilivello del calcolatore (2/2)



Linguaggio assembly (1/2)

- Solleva il programmatore dal lavorare direttamente con sequenze di bit.
- Associa ad ogni comando nel linguaggio macchina (codice operativo) una sequenza di caratteri (forma mnemonica)
- Le locazioni di memoria e i registri possono essere indirizzati con identificatori testuali, anzichè attraverso i loro indirizzi binari.

Linguaggio assembly (2/2)

- Dipende direttamente dall'hardware → ogni CPU o famiglia di CPU ha un suo proprio assembly
- **Pro:** efficienza; controllo completo sull'hw.
- **Contro:** scarsa portabilità; scrittura del codice lenta e costosa.

Esempio

Alto livello: $d = s + t$

Assembly:

Macchina:

Esempio

Alto livello:	$d = s + t$
Assembly:	`add d, t, s`
Macchina:	

Esempio

Alto livello:	$d = s + t$
Assembly:	`add d, t, s`
Macchina:	000000ss sssttttt ddddd000 00100000

Architetture RISC (1/2)

RISC (Reduced Instruction Set Computer) è una strategia di progettazione per microprocessori caratterizzata da

- operazioni semplici eseguibili in
- tempi rapidi e uniformi;
- tanti registri per memorizzare risultati intermedi.

È opposto alla filosofia **CISC** (Complex Instruction Set Computer), in cui si hanno istruzioni più complesse con tempi di esecuzione diversi.

Cenni storici (1/2)

Anni '70

Memorie costose e compilatori inefficienti
→ costruzione di chip in grado di eseguire in hardware istruzioni anche molto complesse
→ architetture CISC

La più celebre CISC ISA (Instruction Set Architecture) è l'Intel x86, che copre tutta la gamma dei processori Intel dagli anni '70 fino ad oggi (Intel Pentium, Core Duo, Core Quad, Core Extreme, Core I3, Core I5, Core I7 . . .)

Cenni storici (2/2)

Anni '80

- Memorie più economiche e compilatori più efficienti
- Divario tra velocità CPU e velocità memorie
- Il 90% del tempo, il processore utilizza sempre un sottoinsieme ristretto di istruzioni
- Esecuzione diretta solo di queste poche istruzioni, lasciando al compilatore l'onere di spezzettare le istruzioni più complesse
- Limitare gli accessi in memoria centrale con poche e semplici modalità di accesso (load/store) e aumentando il numero di registri nel processore.
- $\rightarrow$ architetture RISC

Piattaforme e architetture RISC

- **ARM:** architettura leader nei dispositivi mobile e embedded (iPod, iPhone, iPad, Blackberry, Windows Mobile, Gameboy Advance, Nintendo DS, . . .)
- **Power Architecture:** IBM supercomputer, Apple PowerPC, Nintendo Gamecube e Wii, Xbox 360, Playstation 3, . . .
- **SPARC e UltraSPARC:** Oracle (Sun Microsystems) server
- . . .
- **MIPS:** è l'architettura che studieremo in questo corso. Molti prodotti commerciali sono basati su MIPS, tra cui Playstation, Playstation 2, PSP, e Nintendo 64.

RISC vs CISC

	CISC	**RISC**
Approccio	HW complesso, SW semplice	HW semplice, SW complesso
Codice	Codice compatto; HW si occupa della decodifica di istruzioni complesse	Dimensione del codice aumenta; HW più semplice
Hardware	Pochi registri; Centinaia di istruzioni macchina; Tante modalità di indirizzamento	Molti registri; solo accessi load/store; qualche decina di istruzioni soltanto