Prof. Mariacarla Staffa
a.a. 2022/2023

Laboratorio di Architettura Degli Elaboratori

Introduzione al Corso

About me and my lectures …

- Lectures given by Prof. **Mariacarla Staffa**
- Phone: +39 081 5476580
- email: mariacarla.staffa@uniparthenope.it

- Attività di Ricerca
  - Artificial Intelligence
  - Cognitive Robotics
  - Design and Development of Intelligent agents/robots

Ricevimento Studenti

Martedì: 9:00-11:00

In presenza: Aula 428, piano, IV, lato Nord, CDN (NA)

Da Remoto su TEAMS, codice: 9qzro55

Course disclaimer

This is the 1st edition of this course, there will be lectures you'll like and lectures you won't, there'll be topics clearly explained other not, there will be teaching styles you'll enjoy while others will just bore you.

Keep with me until the end and help me in improving the course so next edition will be marvelous and unforgettable!

Obiettivi

- I due moduli integrati (Architettura dei Calcolatori e Laboratorio di Architettura dei Calcolatori, 6+6 CFU, **esame unico**) hanno l'obiettivo di illustrare gli aspetti fondamentali dell'organizzazione e della architettura dei moderni calcolatori elettronici. Il corso di Laboratorio di Architettura, in particolare, tratta della progettazione digitale di reti combinatorie e sequenziali e di sviluppo di programmi assembly.

Quando

Mercoledì dalle 14:00 alle 16:00

Giovedì dalle 9:00 to 11:00

Dove

Aula 2

Programma Sintetico

- Rappresentazione dell'informazione
- Algebra di Boole
- Reti Combinatorie
- Processori ARM/MIPS
- Linguaggio assembly
- Sviluppo di programmi assembly

Materiale Didattico

- Le presentazioni multimediali (formato .pdf e/o .pptx) di tutte le lezioni saranno disponibili sul sito internet del corso:

https://elearning.uniparthenope.it/

- Emulatori software.

ESAME

- L'obiettivo della verifica è quantificare, per ogni studente, il livello di apprendimento raggiunto riguardo gli argomenti elencati nel programma di "Architettura dei Calcolatori e Laboratorio di Architettura dei Calcolatori".
- La procedura di verifica consiste:
  - In una prova teoria (50% del voto) inerente gli argomenti trattati nel suddetto programma mediante test a risposta multipla su https://elearning.uniparthenope.it .
  - Una prova pratica individuale che sottopone all'allievo quesiti riguardo tanto l'analisi e sintesi delle reti logiche (20% del voto), quanto lo sviluppo di codice Assembler per sistemi MISC/ARM (30% del voto).
  - La complessità dei diversi quesiti proposti in ciascuna prova scritta è di pari livello dei quesiti trattati durante le esercitazioni assistite previste come attività di laboratorio.

Libri di Testo

- A.S Tanenbaum, T. Austin – Architettura dei Calcolator. 6° Edition. Pearson
- S. Harris - Digital Design And Computer Architecture: ARM Edition. M. Kaufmann Editor.

  - Testi Consigliati: W. Stallings - Architettura e organizzazione dei calcolatori (progetto e prestazioni). Pearson Italia, 2004 (traduzione italiana della sesta edizione).

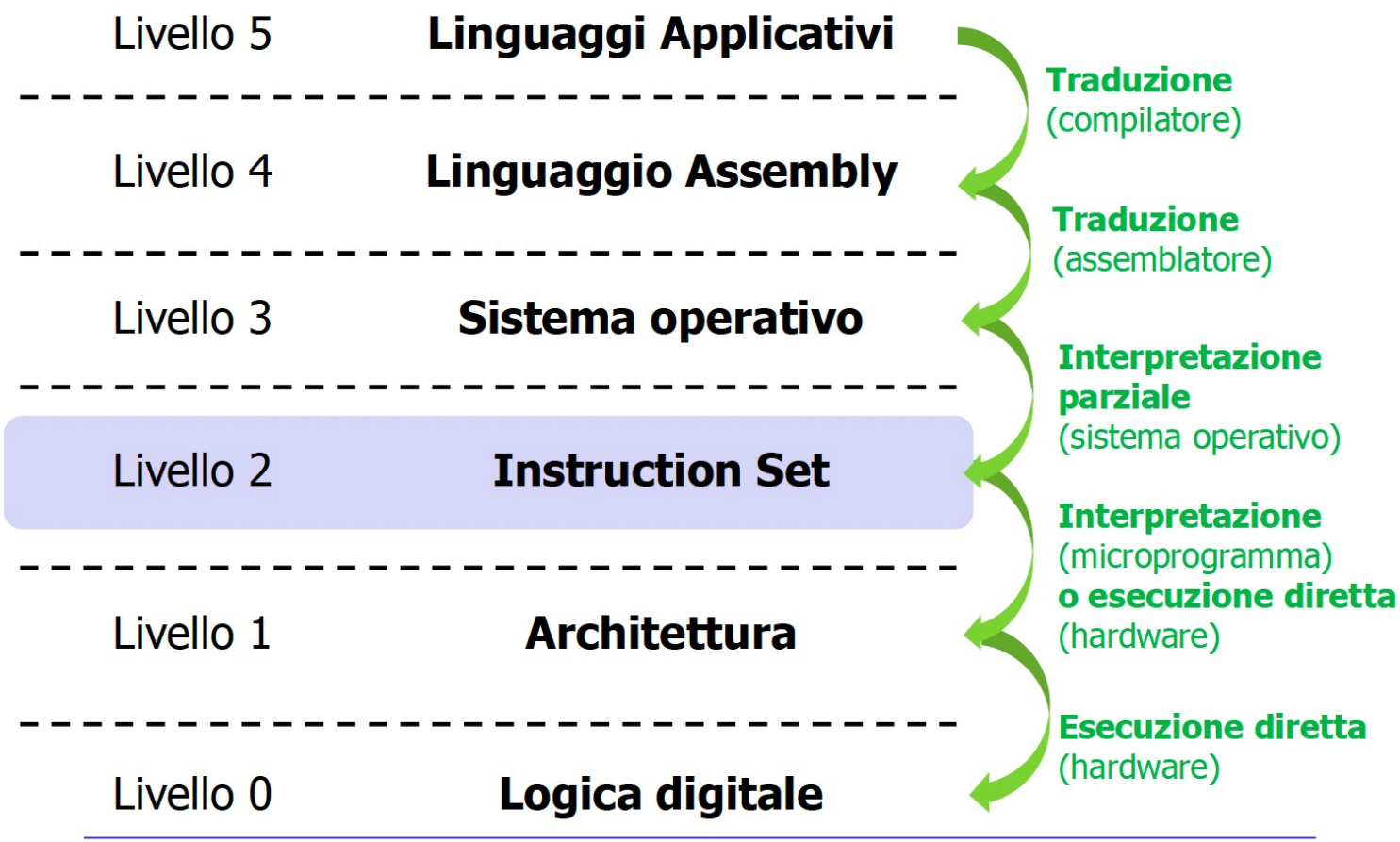
Instruction Set

- L'***instruction set***, in informatica ed elettronica, è l'**insieme di istruzioni macchina** che descrive quegli aspetti, visibili a basso livello al programmatore, dell'architettura di un calcolatore, definita in inglese come ***instruction set architecture*** o in acronimo **ISA**.

- Si tratta di fatto dell'insieme di istruzioni base che il processore può compiere e che costituiscono dunque il suo linguaggio macchina, a partire dal quale vengono scritti i relativi programmi nei vari linguaggi di programmazione a più alto livello di astrazione. Computers con microarchitetture differenti possono condividere lo stesso instruction set. Ad esempio, l'Intel Pentium e l'AMD Athlon implementano versioni quasi identiche dell'instruction set x86,

- Un'ISA è una specifica dell'insieme di tutti quei codici binari (opcode) che rappresentano i comandi implementati nativamente da un particolare design di CPU. L'insieme degli opcode di una specifica ISA è detto anche linguaggio macchina della ISA. Una ISA può anche essere emulata da un interprete software. Poiché l'emulatore deve effettuare una traduzione da una ISA ad una ISA differente, questa soluzione è in generale più lenta rispetto ad una ISA implementata in hardware.

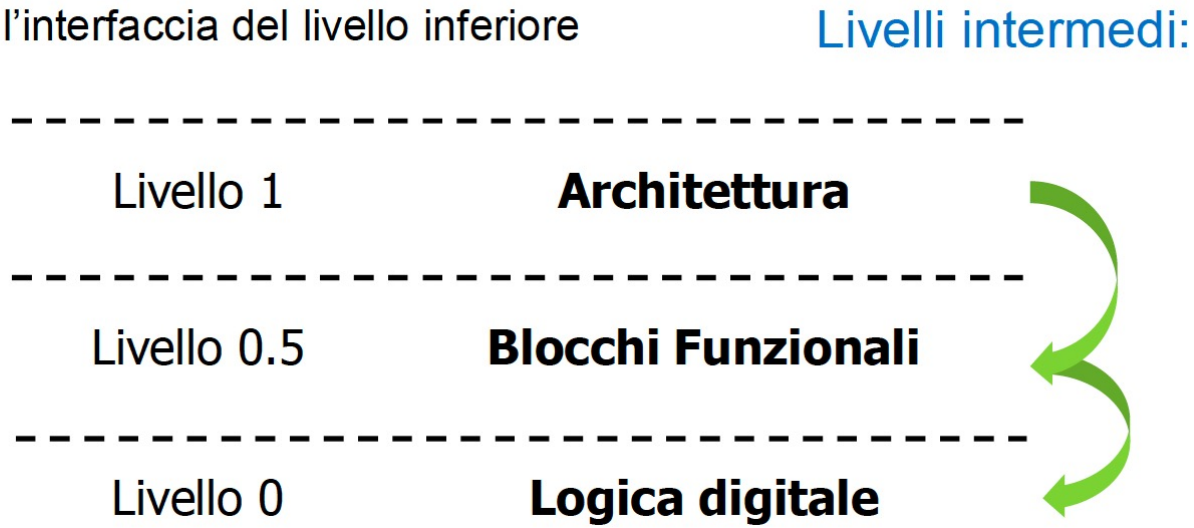
Il Livello Instruction Set Architecture



Livelli di Astrazione

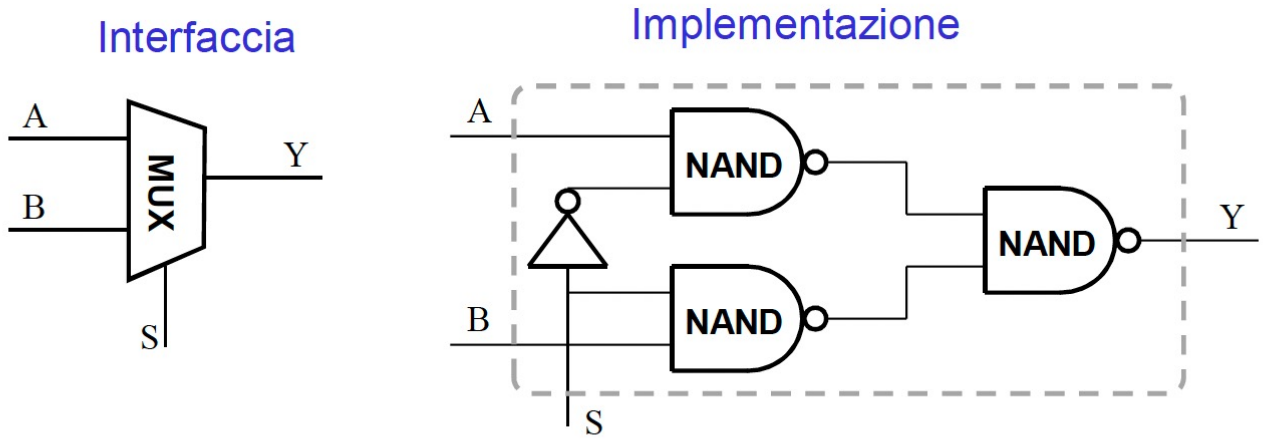
Ciascun livello consiste di:

- Un'interfaccia
  - cos'è visibile dall'esterno
  - usata dal livello superiore
- Un'implementazione
  - come lavora internamente
  - usa l'interfaccia del livello inferiore



Livelli di Astrazione

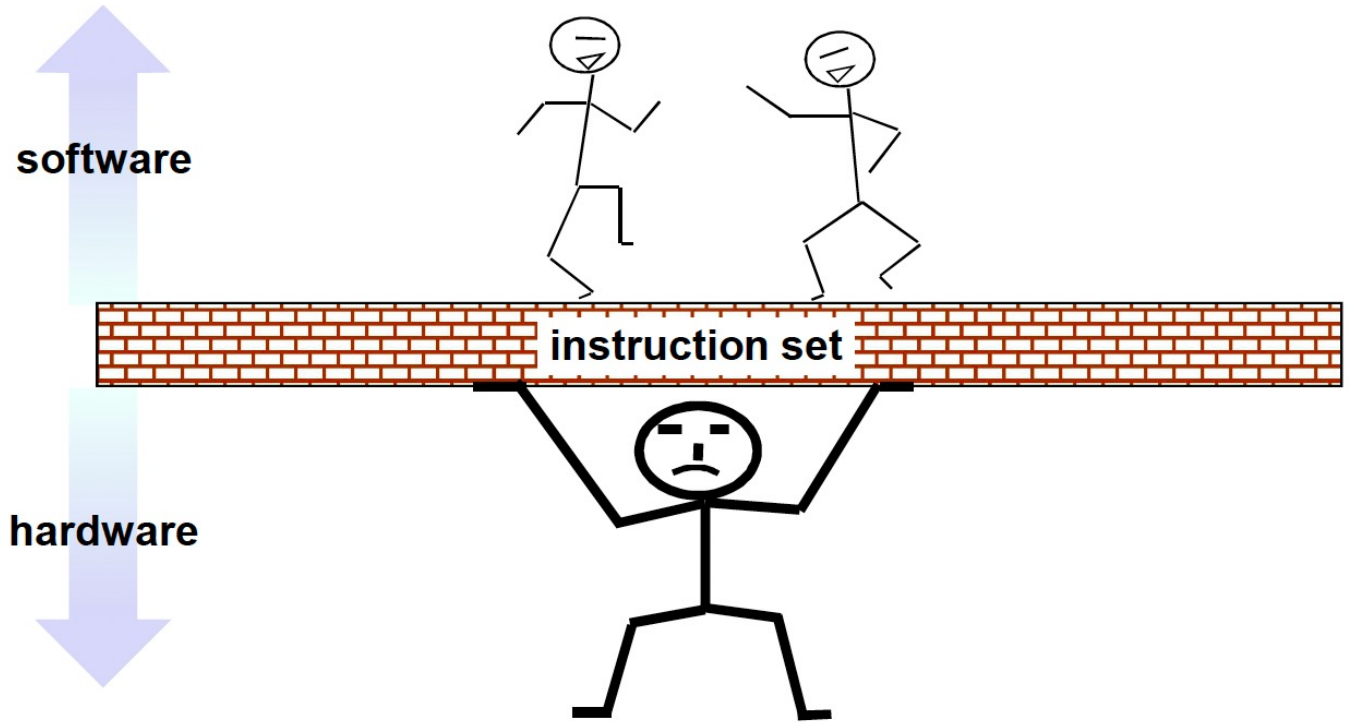
Esempio:
livello dei Blocchi Funzionali



ISA Instruction Set Architecture

- Il livello visto dal programmatore assembly o dal compilatore.
- Comprende:
  - **Instruction Set**
    (quali operazioni possono essere eseguite?)
  - **Instruction Format**
    (come sono fatte le istruzioni?)
  - **Data storage**
    (dove sono posizionati i dati?)
  - **Addressing Modes**
    (come si accede ai dati?)
  - **Exceptional Conditions**
    (come vengono gestiti i casi eccezionali?)
- Una corretta comprensione dell'ISA è importante non solo per i progettisti di compilatori e di sistemi operativi, ma anche per i programmatori.

Il Set di Istruzioni: un interfaccia Critica



Il Set di Istruzioni: un interfaccia Critica

- È un difficile compromesso fra:
  - massimizzare le prestazioni
  - massimizzare la semplicità di uso
  - minimizzare i costi di produzione
  - minimizzare i tempi di progettazione
- Definisce la sintassi e la semantica del linguaggio

- In informatica esistono due tipi popolari di architetture basate sul set di istruzioni:
  - CISC (Complex Instruction Set Computing)
  - RISC (Reduced Instruction Set Computing).

RISC (Reduced Instruction Set Computing)

RISC (computer con set di istruzioni ridotto) è un computer che utilizza un'unità di elaborazione centrale (CPU) che implementa il principio di progettazione del processore delle istruzioni semplificate. Il design dell'architettura RISC semplifica e accelera notevolmente l'elaborazione dei dati riducendo al minimo il numero di istruzioni memorizzate in modo permanente nel microprocessore e si basa maggiormente su istruzioni non residenti, ad esempio codice o programmi software. RISC è probabilmente la tecnologia a microprocessore più veloce ed efficiente disponibile oggi.

RISC (Reduced Instruction Set Computing)

- I chip o microprocessori RISC sfruttano il fatto che la maggior parte delle istruzioni per i processi informatici sono relativamente semplici e di conseguenza i computer sono progettati per gestire rapidamente quelle semplici istruzioni. L'architettura RISC è un miglioramento dell'architettura CISC (complex Instruction Set Computing) utilizzata nei chip Intel Pentium originali. Sebbene Intel abbia lentamente integrato la tecnologia RISC nei suoi chip, sono ancora per lo più basati su CISC.

Vantaggi RISC

- RISC è più un approccio generale all'elaborazione che un insieme specifico di regole, pertanto, diversi processori e sistemi basati su RISC funzionano in modi differente.
- Grazie alla sua semplicità, dà la possibilità di utilizzare lo spazio sui microprocessori.
- La velocità dell'operazione può essere massimizzata e il tempo di esecuzione può essere ridotto al minimo.
- Può essere facilmente progettato rispetto al CISC.
- Le prestazioni sono migliori grazie al set di istruzioni semplificato.
- Le istruzioni RISC eseguono un'istruzione per ciclo di clock.
- Ha un'unità separata per la memoria e quindi è possibile risparmiare spazio su disco.

CISC (Complex Instruction Set Computing)

- **CISC** è un tipo di design a microprocessore. L'architettura CISC contiene un ampio set di istruzioni per computer che vanno da molto semplici a molto complesse e specializzate. Sebbene il progetto calcoli istruzioni complesse nel modo più efficiente, si è successivamente scoperto che molte istruzioni piccole e brevi potevano calcolare istruzioni complesse in modo più efficiente.
- Il microprocessore PowerPC, utilizzato nella workstation RISC system / 6000 di IBM e nei computer Macintosh è un microprocessore RISC. I microprocessori Pentium di Intel sono microprocessori CISC. RISC prende ciascuna delle istruzioni più lunghe e complesse da un progetto CISC e la riduce a più istruzioni che sono più brevi e più veloci da elaborare.

Vantaggi CISC

- È facile aggiungere nuovi comandi nel chip senza modificare la struttura del set di istruzioni poiché l'architettura utilizza un hardware generico per eseguire i comandi.
- In genere è più facile scrivere codice assembly CISC rispetto al codice assembly RISC.
- CISC supporta la microprogrammazione.
- Gli operandi in memoria vengono manipolati dalle istruzioni.
- I CISC hanno un gran numero di istruzioni ben definite, un fattore che rende i linguaggi di alto livello facili da progettare e implementare.
- Ha più numero di codici di indirizzamento e meno numero di registri.

Differenza tra Architettura RISC e CISC

BASE DI CONFRONTO	RISC	CISC
Numero di modalità di indirizzamento	RISC ha meno modalità di indirizzamento e la maggior parte delle istruzioni nel set di istruzioni ha un registro per registrare la modalità di indirizzamento.	CISC ha molte diverse modalità di indirizzamento e può quindi essere utilizzato per rappresentare in modo più efficiente le dichiarazioni del linguaggio di programmazione di livello superiore.
Unità di microprogrammazione	È un'unità di programmazione cablata.	Dispone di un'unità di microprogrammazione.
Esempi	System / 360, PDP-11, VAX, AMD, Motorola 68000 e PC desktop su CPU Intel x86.	DEC Alpha, AMD 29000, ARC, Atmel AVR, Blackfin, Intel i860 e i960, MIPS, Motorola 88000, PA-RISC, power (incluso PowerPC), SuperH, SPARC e ARM.

Differenza tra Architettura RISC e CISC

Codifica delle istruzioni	Vengono utilizzate le codifiche a lunghezza fissa delle istruzioni. Esempio: in IA32, generalmente tutte le istruzioni sono codificate come 4 byte.	Vengono utilizzate codifiche a lunghezza variabile delle istruzioni. Esempio: la dimensione dell'istruzione IA32 può variare da 1 a 15 byte.
Operazioni aritmetiche e logiche	Le operazioni aritmetiche e logiche utilizzano solo operandi di registro.	Le operazioni aritmetiche e logiche possono essere applicate sia alla memoria che agli operandi del registro.
Il set di istruzioni	Il set di istruzioni è ridotto, cioè ha solo poche istruzioni nel set di istruzioni. Molte di queste istruzioni sono molto primitive.	Il set di istruzioni ha una varietà di istruzioni differenti che possono essere utilizzate per operazioni complesse.

Differenza tra Architettura RISC e CISC

Applicazione	Viene utilizzato in applicazioni di fascia alta come elaborazione video, telecomunicazioni ed elaborazione di immagini.	Viene utilizzato in applicazioni di fascia bassa come sistemi di sicurezza, automazione domestica ecc.
Processore	I suoi processori hanno semplici istruzioni che richiedono circa un ciclo di clock.	Il suo processore ha istruzioni complesse che richiedono più clock per l'esecuzione.
Pipelining del processore	I suoi processori sono altamente pipeline.	I processori sono normalmente poco pipeline oppure non pipeline.

Differenza tra Architettura RISC e CISC

Programmi di attuazione	I programmi di implementazione sono esposti ai programmi a livello di macchina.	I programmi di implementazione sono nascosti dai programmi a livello di macchina.
Modalità di indirizzamento complesse	Le modalità di indirizzamento complesse vengono sintetizzate utilizzando il software.	Supporta già modalità di indirizzamento complesse.
Complessità	La complessità sta nel micro programma.	La complessità di RISC risiede nel compilatore che esegue il programma.
Prestazione	Le prestazioni sono ottimizzate con maggiore attenzione al software.	Le prestazioni sono ottimizzate con maggiore attenzione all'hardware.

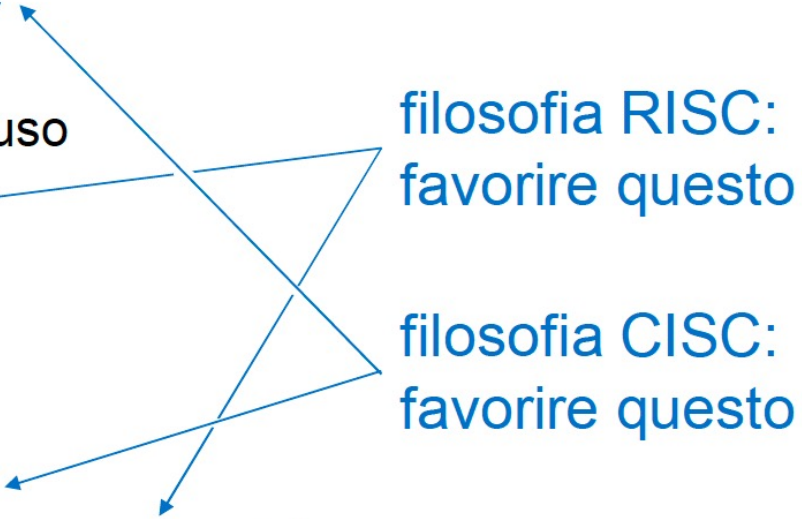
Differenza tra Architettura RISC e CISC

Memoria esterna	Non richiede memoria esterna per i calcoli.	Richiede una memoria esterna per i calcoli.
Decodifica delle istruzioni	La decodifica delle istruzioni è semplice.	La decodifica delle istruzioni è complessa.
Espansione del codice	L'espansione del codice può essere un problema.	L'espansione del codice non è un problema.
Tempo di esecuzione	Il tempo di esecuzione è molto inferiore.	Il tempo di esecuzione è molto alto.
Registri	Sono presenti più set di registri.	È presente un solo set di registri.
Unità di memoria	Non ha unità di memoria e utilizza un hardware separato per implementare le istruzioni.	Ha un'unità di memoria per implementare istruzioni complesse.
Dimensione del programma	RISC ha una grande dimensione del programma.	I CISC hanno un programma di piccole dimensioni.

Criteri di Definizione di un IS

- Esiste un'infinità di insiemi di istruzioni *equivalenti*
  che permettono di scrivere codice per eseguire… qualsiasi cosa*

- La scelta di un insieme delle istruzioni prende in considerazione:
  - la semplicità della realizzazione
    - cioè dell'HW richiesto
  - l'espressività e la semplicità di uso
    - la potenza delle istruzioni
    - la facile programmabilità
  - l'efficienza
    - velocità di esecuzione
    - cioè: (vel di ogni istruzione)
      x (quante se ne rendono necessarie)



* …purché sia computabile, e non richieda troppe risorse.
Vedi corsi di informatica teorica!

Criteri di Definizione di un IS

- Oltre a questi criteri obiettivi,
  molte altre forze più capricciose
  hanno plasmato gli Istruction Sets oggi usati
  - La loro storia
    (soprattutto il bisogno di back compatibility)
    (non tutte le scelte sono logiche a posteriori)
  - Fattori economici / legali
  - Considerazioni tecniche pertinenti al contesto del loro sviluppo

Gli Instruction Set più popolari

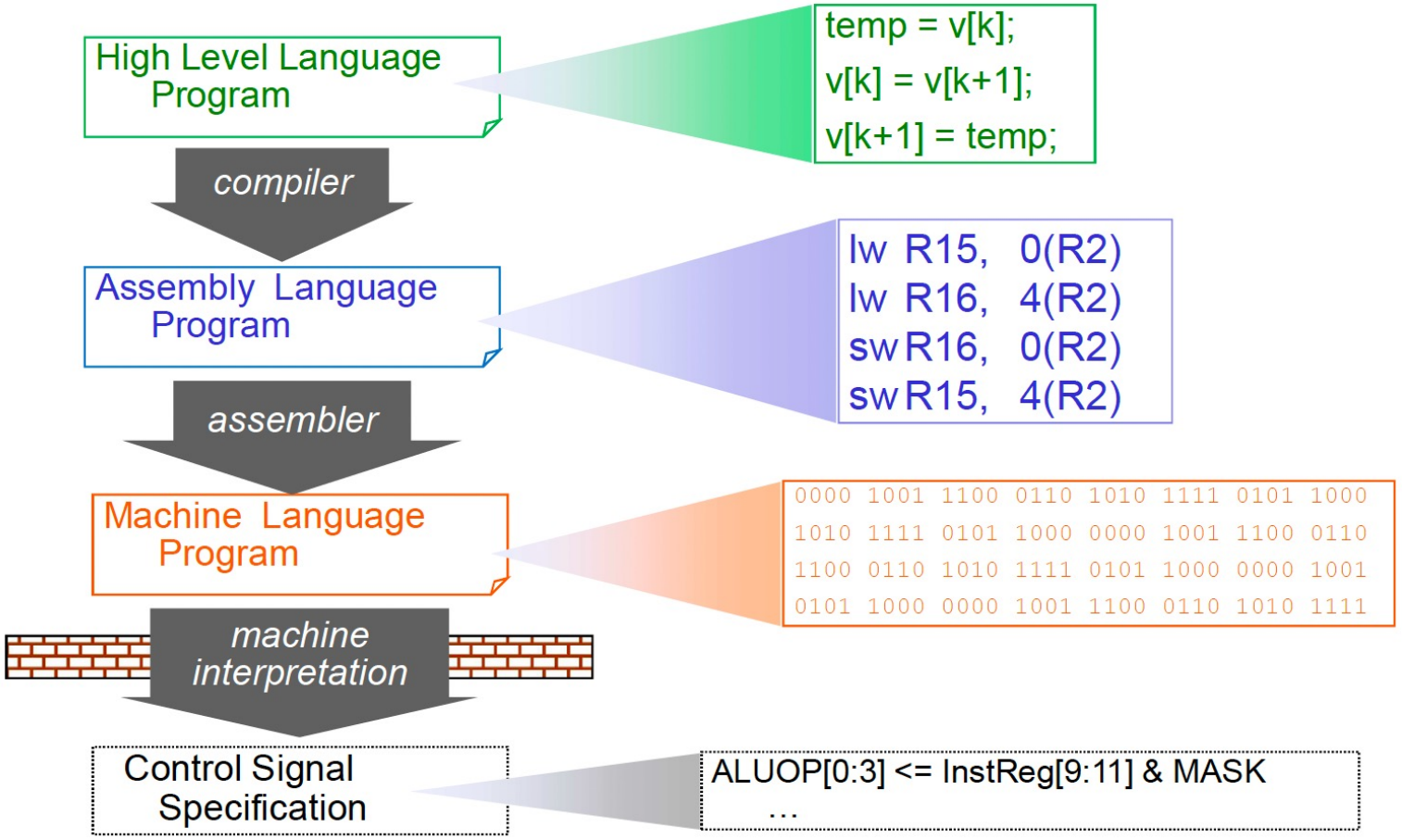
MIPS
ARM
} filosofia RISC

X86
X64
} filosofia CISC

Organizzazione della Macchina

- L'organizzazione della macchina è la vista del computer del progettista logico. Comprende:
  - Funzionalità e prestazioni delle unità funzionali (registri, ALU, ecc.)
  - Modalità di connessione delle unità funzionali
  - Come l'informazione viene comunicata tra i componenti
  - Controllo dei trasferimenti di informazioni
  - Coordinamento delle unità funzionali per la realizzazione dell'ISA.
- Tipicamente l'organizzazione è concepita per supportare un dato set di istruzioni.
- Tuttavia, per progettare un buon insieme di istruzioni è importante capire come potrebbe essere implementata l'architettura.

Il Codice da un livello all'altro



Programmi in Linguaggio Macchina (programmi binari)

- Un programma (software) scritto in linguaggio macchina (di uno specifico Instruction Set, per uno specifico SO)

- Consiste in :
  - **DATI**
    tenuti in un blocco di memoria
  - **ISTRUZIONI**
    tenuti in un altro blocco di memoria
  - per convenzione, si usano due aree di memoria diverse, dedicate a ciascuna di queste due cose

- Le istruzioni sono:
  - quelle dell'Instruction Set
  - o invocazioni di funzioni del Sistema Operativo («system calls») (funzioni nel «kernel» di quel SO, per es per aprire un file)

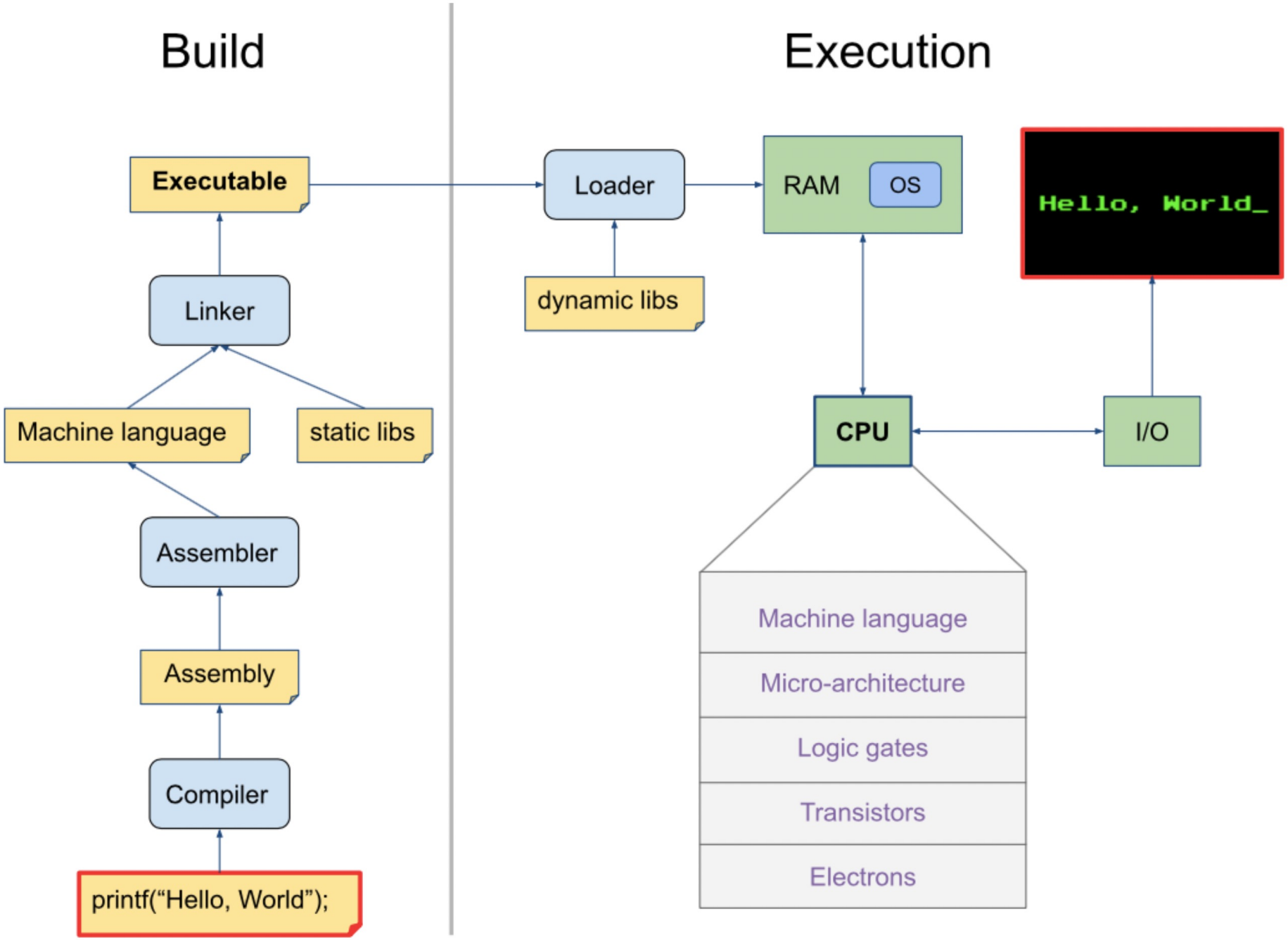
Programmi in Linguaggio Macchina (programmi binari)

- Quindi un Programma Binario è **specifico** per un Instruction Set e un Sist Op
  - (cioè per una «platform»)
  - gira solo su una macchina con un architettura che implementa quell'Instruction Set e fornito di quel SO
- Alcuni dettagli pratici dei binari più comuni:
  - Programmi per Win64:
    Instruction Set x64, e SO Windows
  - Programmi per Win32:
    Instruction Set x86, e SO Windows
    ma sono eseguibili anche da architetture x64, perché è back-compatible (seppur con una efficienza leggermente minore, perché non si sfrutta la maggiore potenza dell'x64). Nota: non viceversa.
  - Programmi per MacOS:
    possono contenere versioni del binario per due Instruction Set: x64 e x86 in modo da essere eseguiti in modo ottimale su entrambe le architetture
  - Programmi per Linux
- Il sorgente (in linguaggio ad alto livello) è indipendente dalla piattaforma (è «cross-platform») perché può invece essere compilato per piattaforme diverse

Eseguire Programmi binari

- Caricare il programma (DATI + ISTRUZIONI) in memoria
- Avviare il programma

- Sarebbe estremamente difficile scrivere programmi binari direttamente
- Durante questo corso saliremo di due livelli, e scriveremo i nostri programmi in un linguaggio leggermente più ad alto livello: Assembly


Build
Execution
Executable
Linker
Machine language
static libs
Assembler
Assembly
Compiler
printf("Hello, World");
Loader
dynamic libs
RAM
OS
Hello, World_
CPU
I/O
Machine language
Micro-architecture
Logic gates
Transistors
Electrons

Prima del Linguaggio Assembly…

- Rappresentazione dell'Informazione
- Rappresentazione Binaria e Strutture di Calcolo
- Il Livello Hardware: Reti Logiche
- Il Livello della Macchina Assembler