POLITECNICO DI MILANO

High Performance Processors and Systems

Class introduction

- A bird's eye view on the HPPS 2010 class -

Donatella Sciuto: sciuto@elet.polimi.it

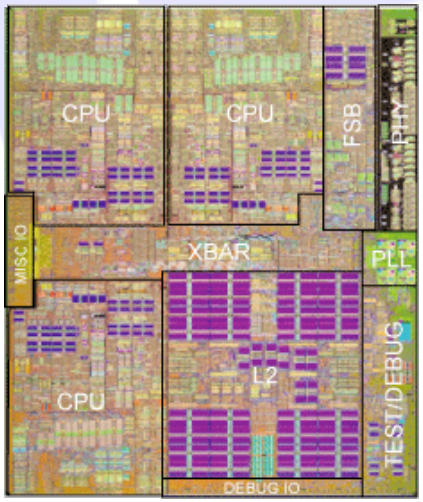
Marco D. Santambrogio: marco.santambrogio@polimi.it

HPPS

One core to rule them all

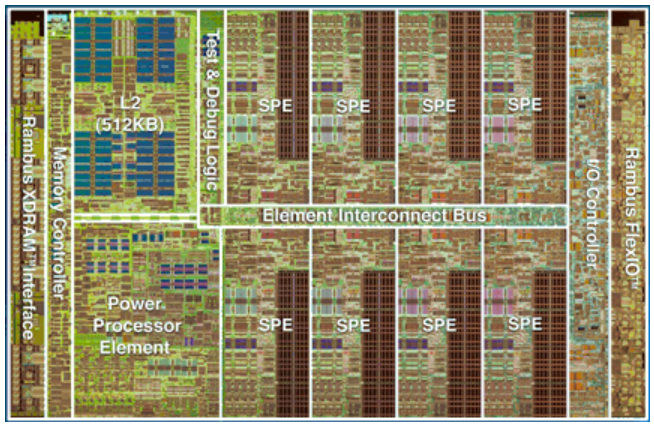
Xenon: XBOX360

- Three symmetrical cores
  - each two way SMT-capable and clocked at 3.2 GHz



Cell: PS3

- Cell is a heterogeneous chip multiprocessor
  - One 64-bit Power core
  - 8 specialized co-processors



Top 10 Reasons to Attend the HPPS Class!

1. Transform massive silicon resources into performance benefits
2. HPPS and CA impacts every other aspect of electrical engineering and computer science
3. Investigate computer architecture innovative challenging researches
4. Do cross-cutting research, where our imagination is the only real limit
5. Design novel computer architecture
6. Build portable systems where the system automatically optimizes for different platforms
7. Design reconfigurable, controllable, observable components
8. It is exciting!
9. It has never been more exciting!
10. Do I really need a 10?

Outline

- Computer Architecture: Overview
  - Introduction
  - Technologies and Performance
  - How to turn lead into gold
- Class organization
- End?...

Introduction

Architecture: Which Definition?

- Abstract architecture
  - the ***functional specification*** of a computer
- Concrete architecture (aka microarchitecture)
  - an ***implementation*** of an abstract architecture.

- Abstract architecture: a “black box” specification of a machine – can be seen:
  - From the programmer’s point of view
    - we deal with a ***programming model***, equivalent to description of the ***machine language***;
  - From the designer’s point of view
    - we deal with a ***hardware model*** (a black-box description for the designer: must include additional information, e.g., interface protocols etc.).

Where Do We Start

- Background: the “Von Neumann paradigm” (and the Harvard alternative)
- Extension to a “reactive paradigm” – still Von Neumann!

- An architectural paradigm
  - Composition of hardware and program execution mode;
  - Does not include software, but implies the execution mode of object code!

What is “Computer Architecture”

Computer Architecture =

Instruction Set Architecture +

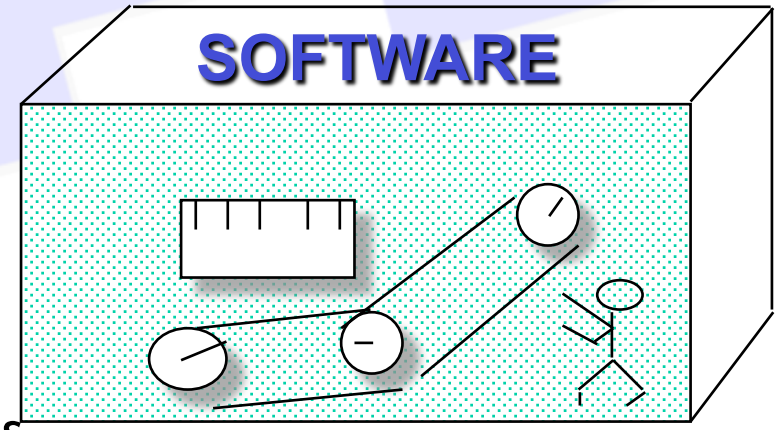
Machine Organization + …..

Instruction Set Architecture (subset of Computer Arch.)

... the attributes of a [computing] system as seen by the programmer, *i.e.* the conceptual structure and functional behavior, as distinct from the organization of the data flows and controls the logic design, and the physical implementation.

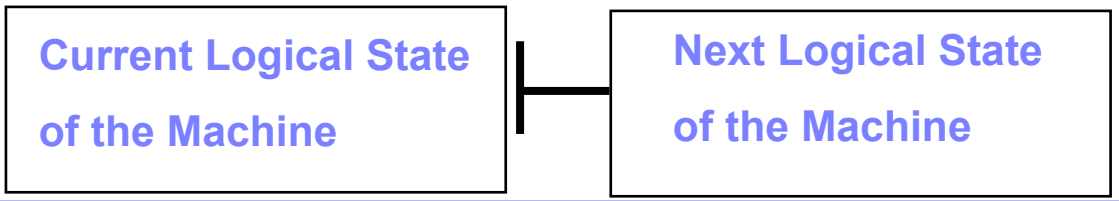
Amdahl, Blaaw, and Brooks, 1964

- Organization of Programmable Storage
- Data Types & Data Structures: Encodings & Representations
- Instruction Set
- Instruction Formats
- Modes of Addressing and Accessing Data Items and Instructions
- Exceptional Conditions

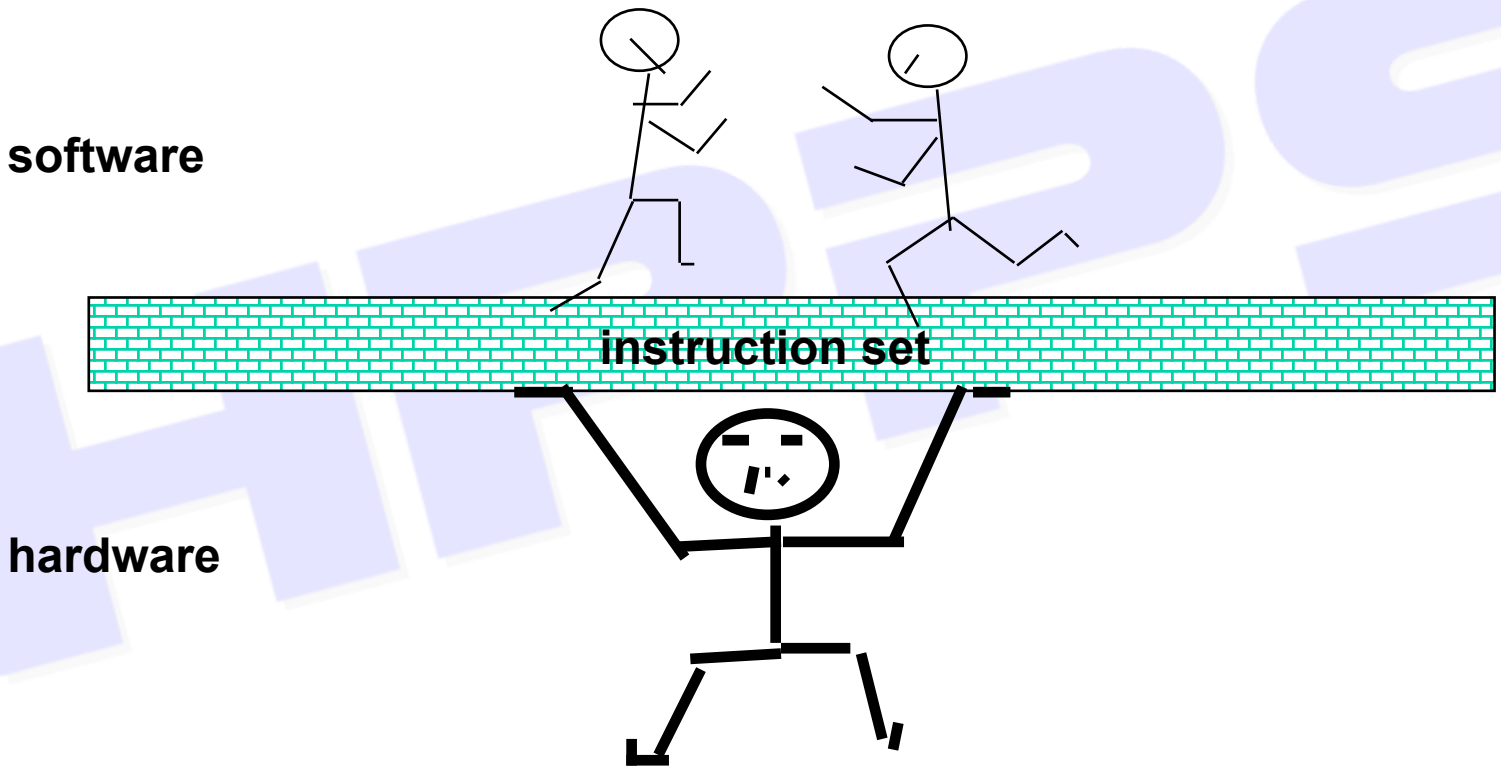


Elements of an ISA

- Set of machine-recognized data types
  - bytes, words, integers, floating point, strings, . . .
- Operations performed on those data types
  - Add, sub, mul, div, xor, move, ….
- Programmable storage
  - regs, PC, memory
- Methods of identifying and obtaining data referenced by instructions (addressing modes)
  - Literal, reg., absolute, relative, reg + offset, …
- Format (encoding) of the instructions
  - Op code, operand fields, …



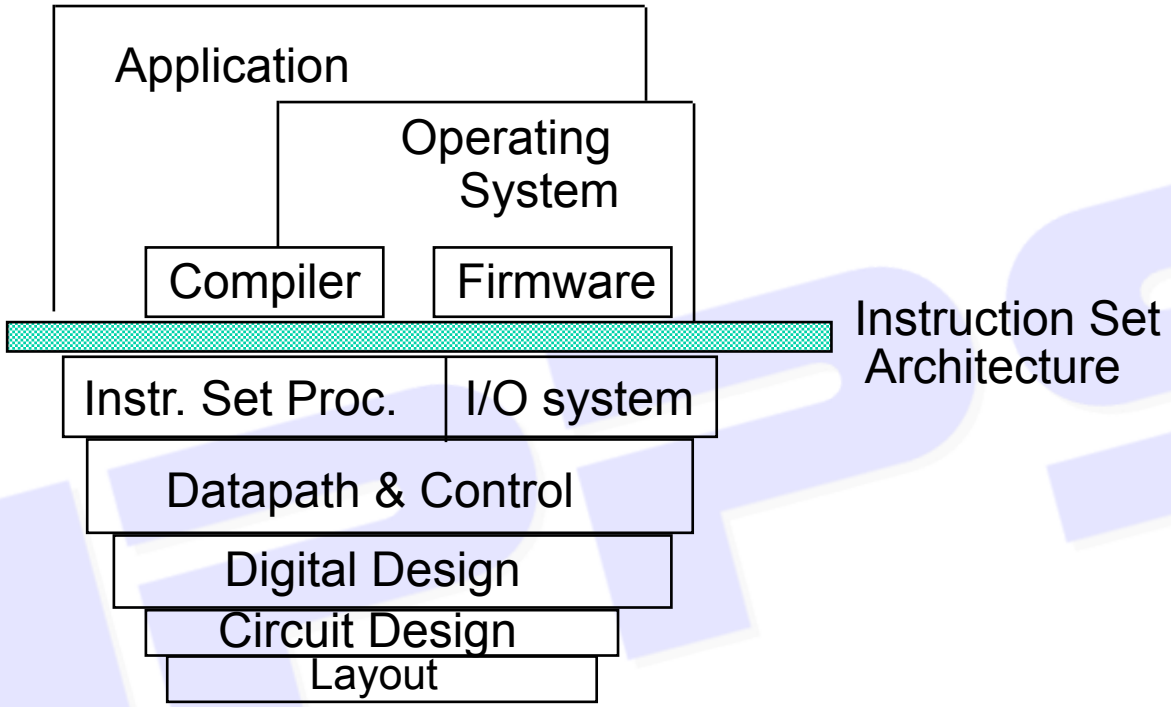
The Instruction Set: a Critical Interface



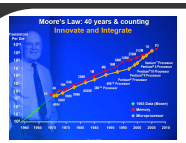
Computer Architecture's Changing Definition

- 1950s to 1960s: Computer Architecture Course: Computer Arithmetic
- 1970s to mid 1980s: Computer Architecture Course: Instruction Set Design, especially ISA appropriate for compilers
- 1990s: Computer Architecture Course: Design of CPU, memory system, I/O system, Multiprocessors, Networks
- 2020s: Computer Architecture Course: Self adapting systems? Self organizing structures? DNA Systems/ Quantum Computing?

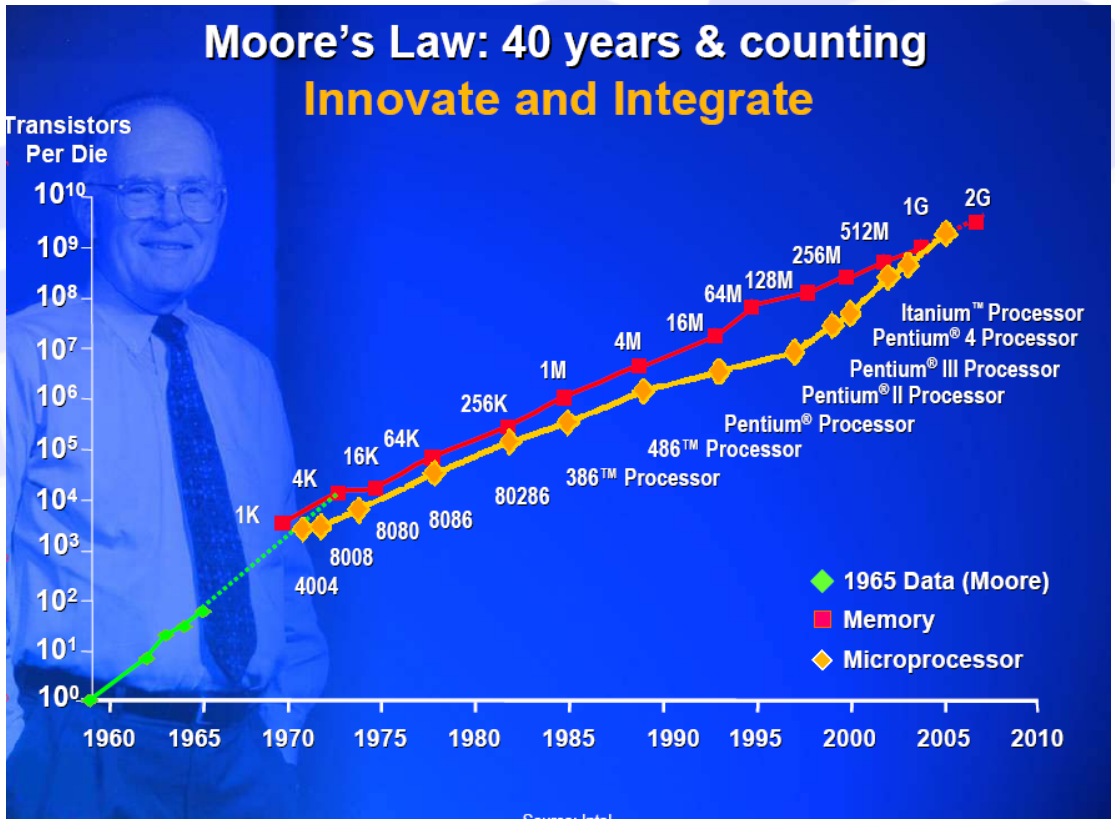
What is "Computer Architecture"?



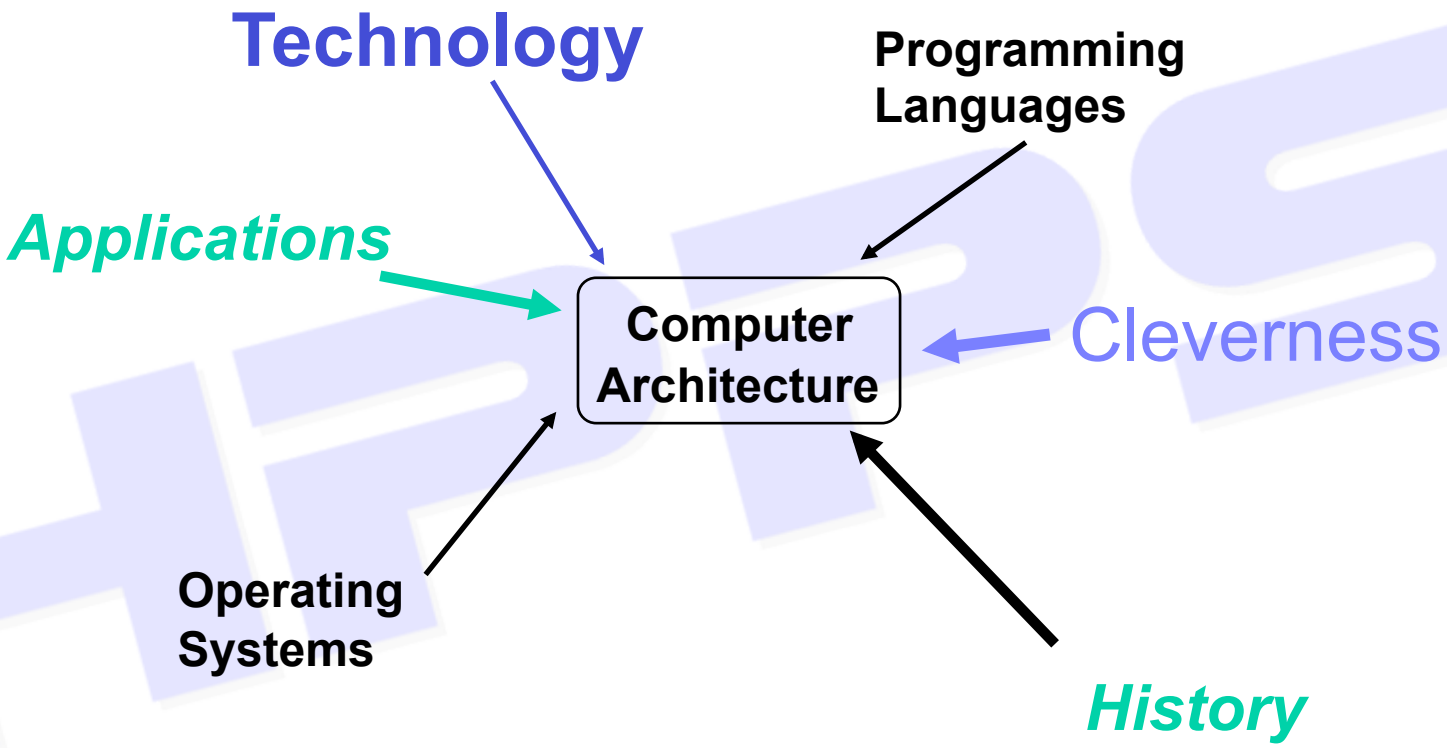
- Coordination of many *levels of abstraction*
- Under a rapidly changing set of forces
- Design, Measurement, *and* Evaluation



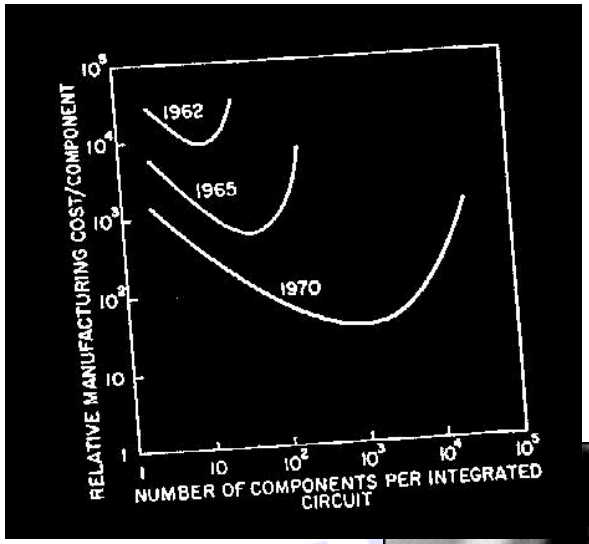
Technologies and Performance

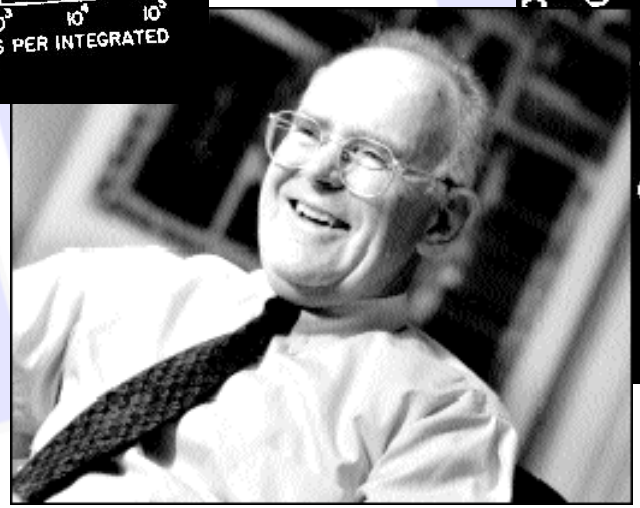


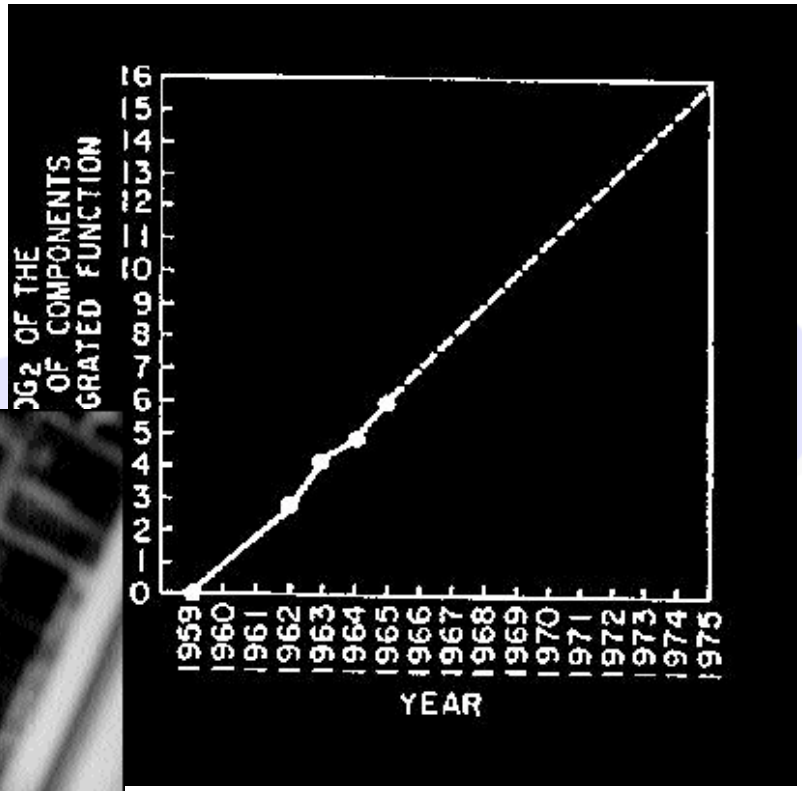
Forces on Computer Architecture



Amazing Underlying Technology Change



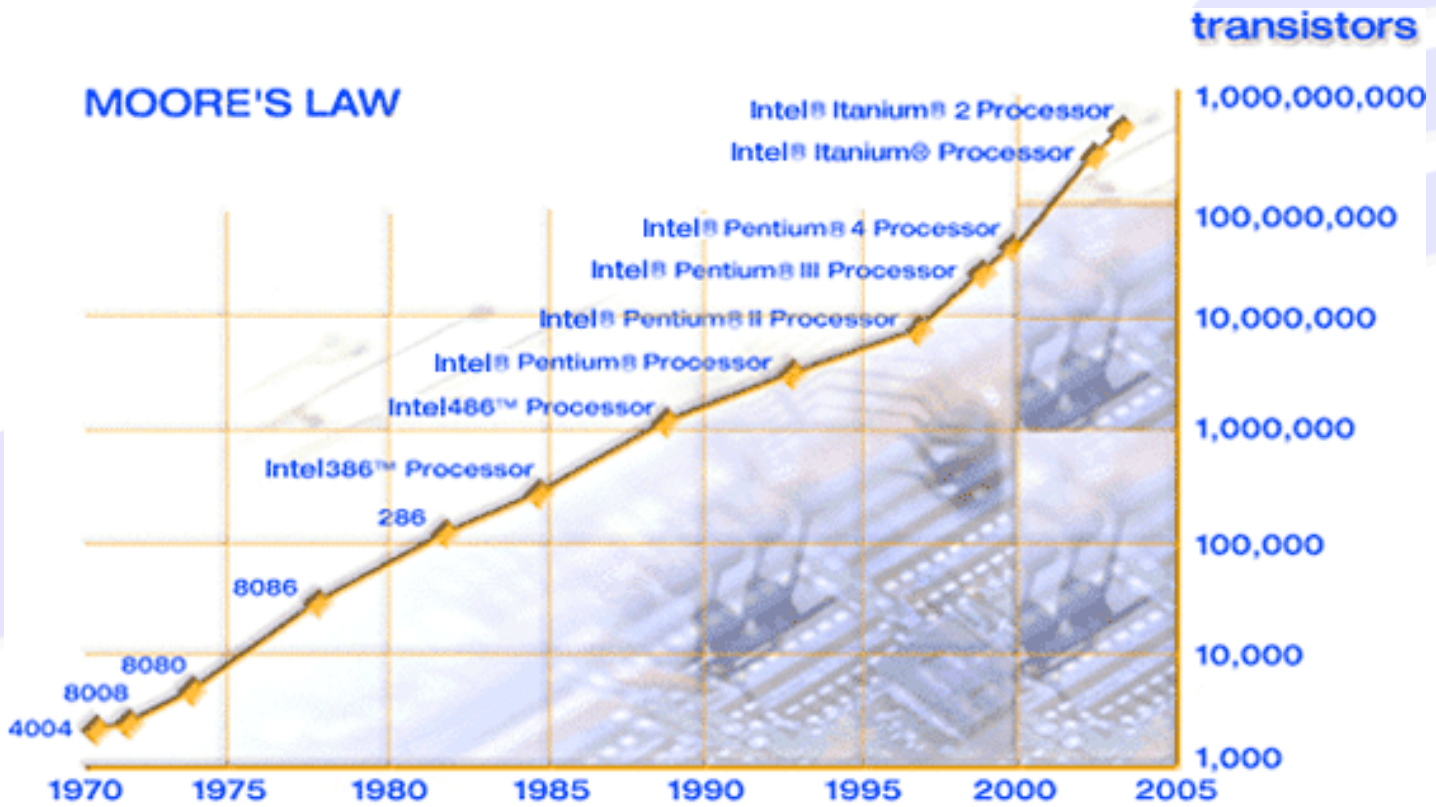




- "Cramming More Components onto Integrated Circuits" Gordon Moore, Electronics, 1965
  http://www.intel.com/technology/silicon/mooreslaw/
- **the number of transistors on a chip doubles about every two years**

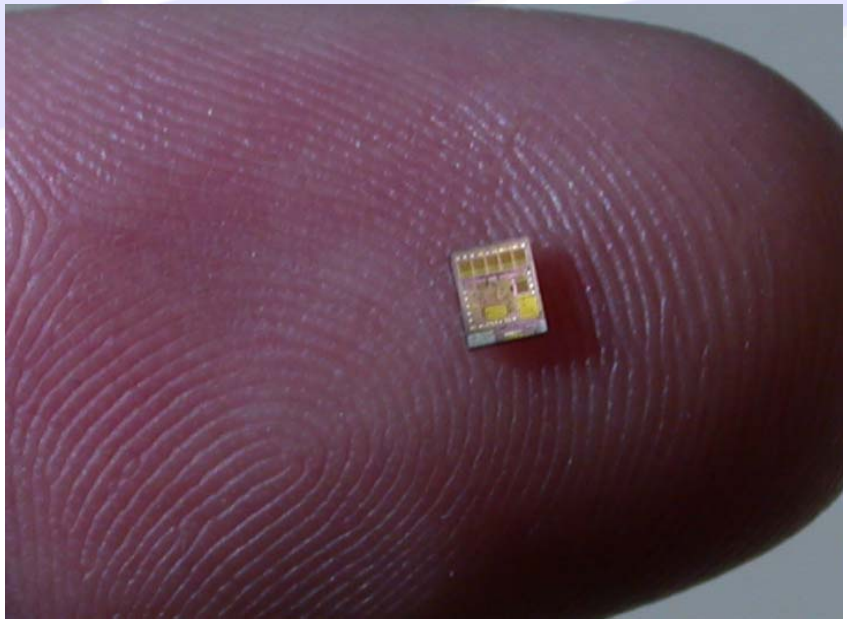
Moore's law in Intel

- **Moore's Law Means More Performance.** Processing power, measured in millions of instructions per second (MIPS), has risen because of increased transistor counts



It's not just about bigger and faster!

- Complete computing systems can be tiny and cheap
- System on a chip
- Resource efficiency
  - Real-estate, power, pins, …



Integrated Approach

- What really matters is the functioning of the complete system, i.e. hardware, runtime system, compiler, and operating system

- In networking, this is called the "End to End argument"
  - Computer architecture is not just about transistors, individual instructions, or particular implementations
  - Original RISC projects replaced complex instructions with a compiler + simple instructions

Definition: Performance

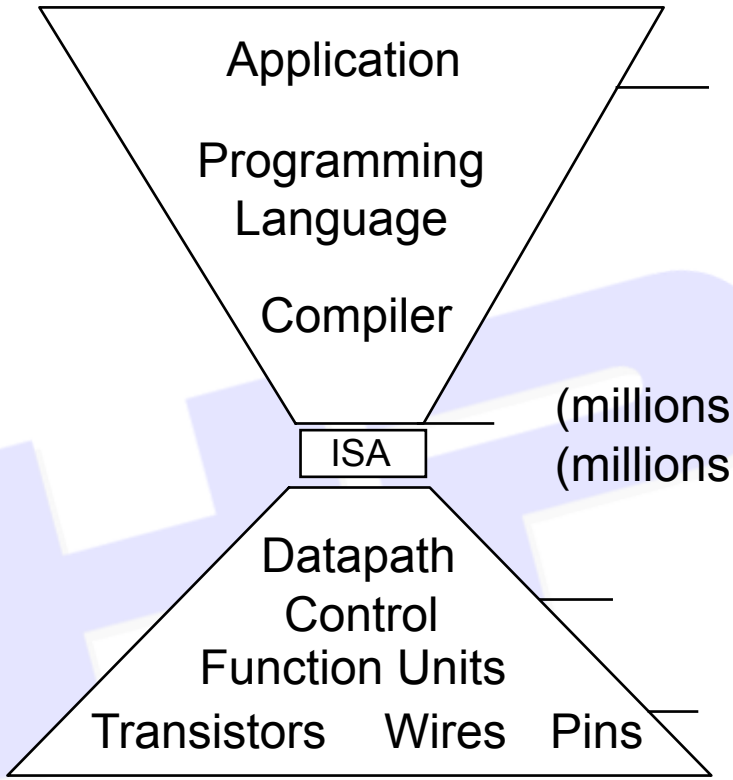
- Performance is in units of *things* per sec
  - bigger is better

- If we are primarily concerned with response time
  - Performance (x) = 1/execution time (x)

"X is n times faster than Y" means

$$n = \frac{\text{Performance(X)}}{\text{Performance(Y)}} = \frac{\text{Execution\_time(Y)}}{\text{Execution\_time(X)}}$$

Metrics of Performance



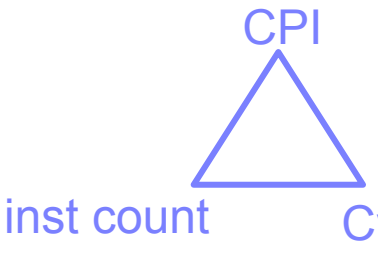
Answers per day/month

(millions) of Instructions per second: MIPS
(millions) of (FP) operations per second: MFLOP/s

Megabytes per second

Cycles per second (clock rate)

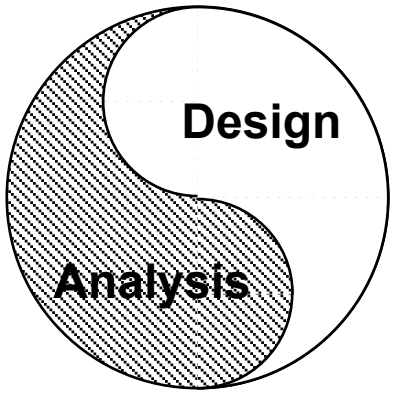
Components of Performance



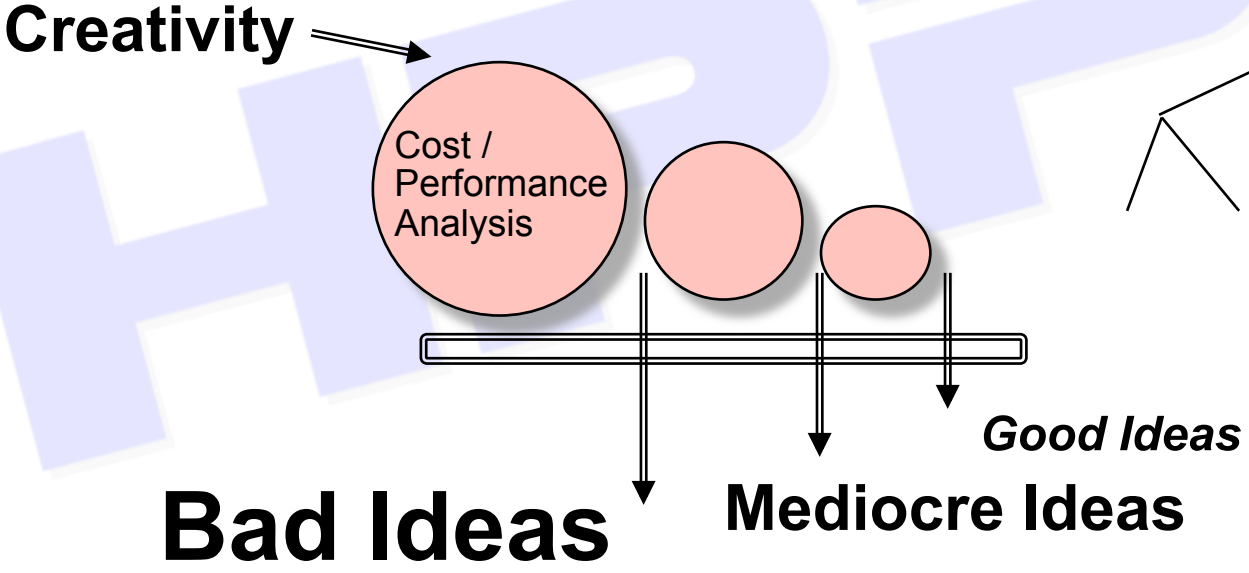
$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

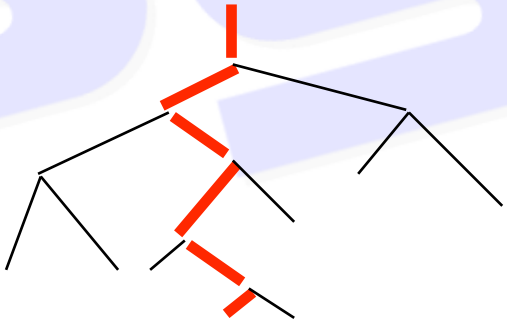
	Inst Count	CPI	Clock Rate
Program	X		
Compiler	X	(X)	
Inst. Set.	X	X	
Organization	X		X
Technology			X

Measurement and Evaluation

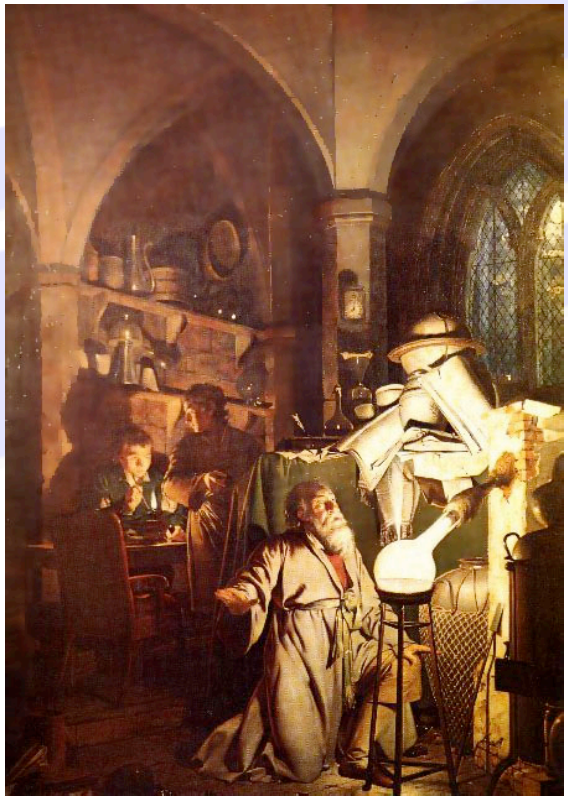


- Architecture is an iterative process
  - searching the space of possible designs
  - at all levels of computer systems





How to turn lead into gold



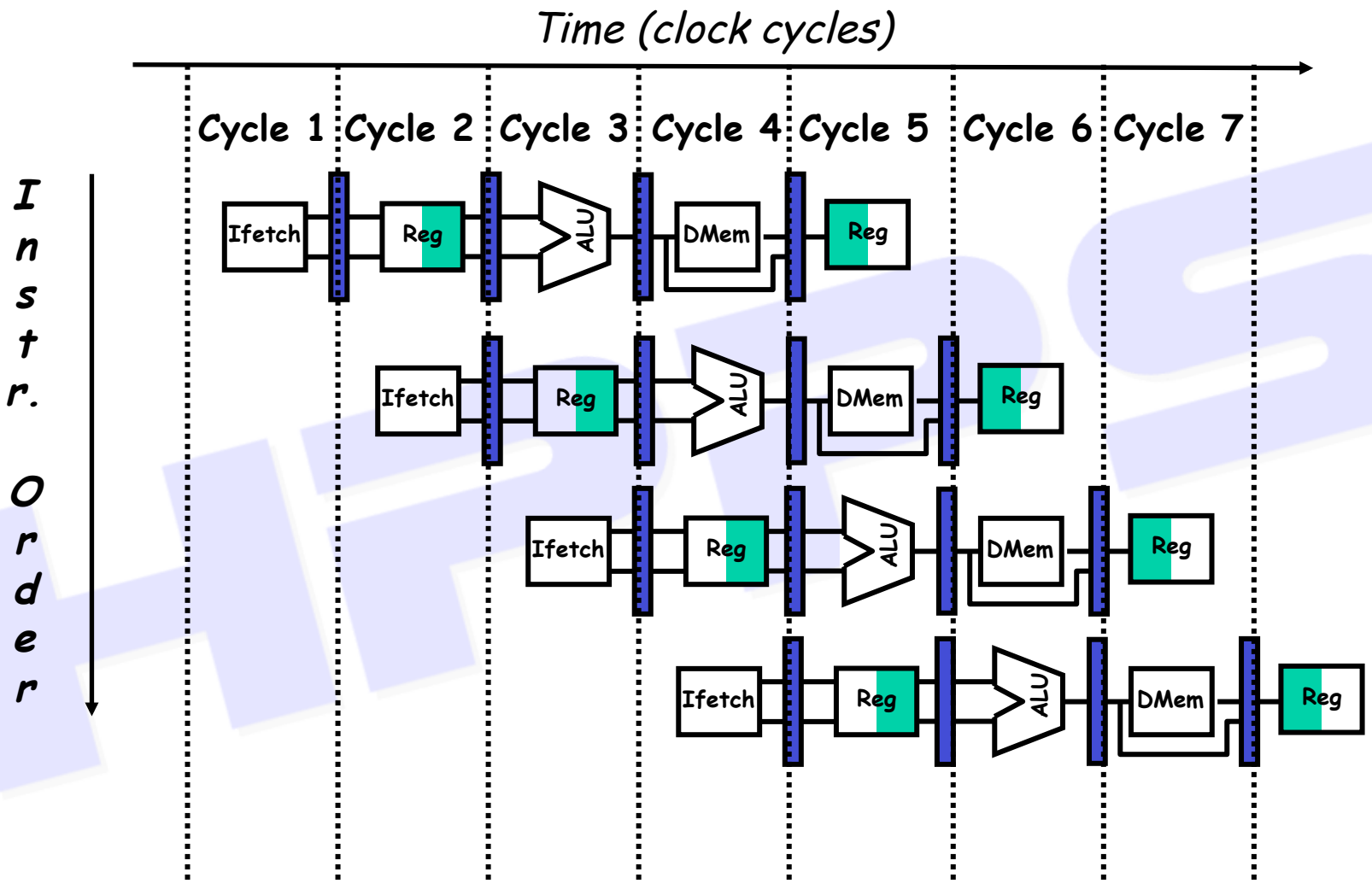
The Alchemist (1771), Joseph Wright of Derby
(Derby Museum and Art Gallery, Derby, UK)

How do you turn more stuff into more performance?

- Do more things at once
- Do the things that you do faster

- Beneath the ISA illusion….

Pipelined Instruction Execution



Limits to pipelining

- Maintain the von Neumann “illusion” of one instruction at a time execution

- Hazards prevent next instruction from executing during its designated clock cycle
  - Structural hazards: attempt to use the same hardware to do two different things at once
  - Data hazards: Instruction depends on result of prior instruction still in the pipeline
  - Control hazards: Caused by delay between the fetching of instructions and decisions about changes in control flow (branches and jumps).

Progression of ILP

- 1st generation RISC – pipelined
  - Full 32-bit processor fit on a chip ->issue almost 1 IPC
    - Need to access memory 1+x times per cycle
  - Floating-Point unit on another chip
  - Cache controller a third, off-chip cache
  - 1 board per processor -> multiprocessor systems
- 2nd generation: superscalar
  - Processor and floating point unit on chip (and some cache)
  - Issuing only one instruction per cycle uses at most half
  - Fetch multiple instructions, issue couple
    - Grows from 2 to 4 to 8 …
  - How to manage dependencies among all these instructions?
  - Where does the parallelism come from?
- VLIW
  - Expose some of the ILP to compiler, allow it to schedule instructions to reduce dependences

Modern ILP

- Dynamically scheduled, out-of-order execution
- Current microprocessor fetch 10s of instructions per cycle
- Pipelines are 10s of cycles deep
  - many 10s of instructions in execution at once
- Grab a bunch of instructions, determine all their dependences, eliminate dep's wherever possible, throw them all into the execution unit, let each one move forward as its dependences are resolved
- Appears as if executed sequentially
- On a trap or interrupt, capture the state of the machine between instructions perfectly
- Huge complexity

Have we reached the end of ILP?

- Multiple processor easily fit on a chip
- Every major microprocessor vendor has gone to multithreading
  - Thread: loci of control, execution context
  - Fetch instructions from multiple threads at once, throw them all into the execution unit
  - Intel: hyperthreading,
  - Concept has existed in high performance computing for 20 years (or is it 40? CDC6600)
- Vector processing
  - Each instruction processes many distinct data
  - Ex: MMX
- Raise the level of architecture – many processors per chip

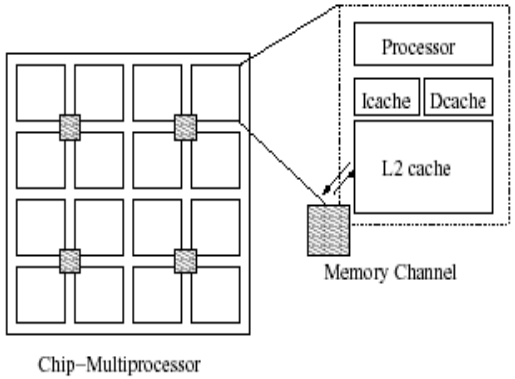
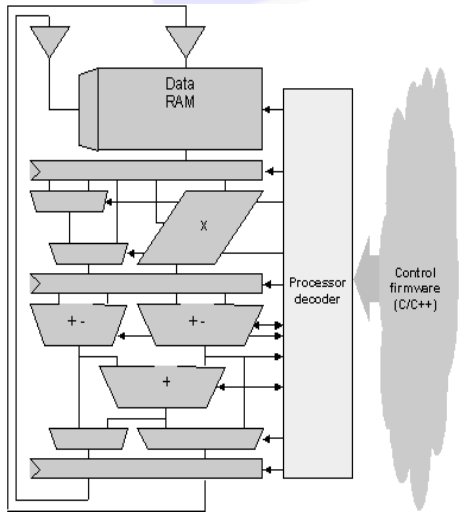


Figure 1. Chip-multiprocessor model.



Tensilica Configurable Proc

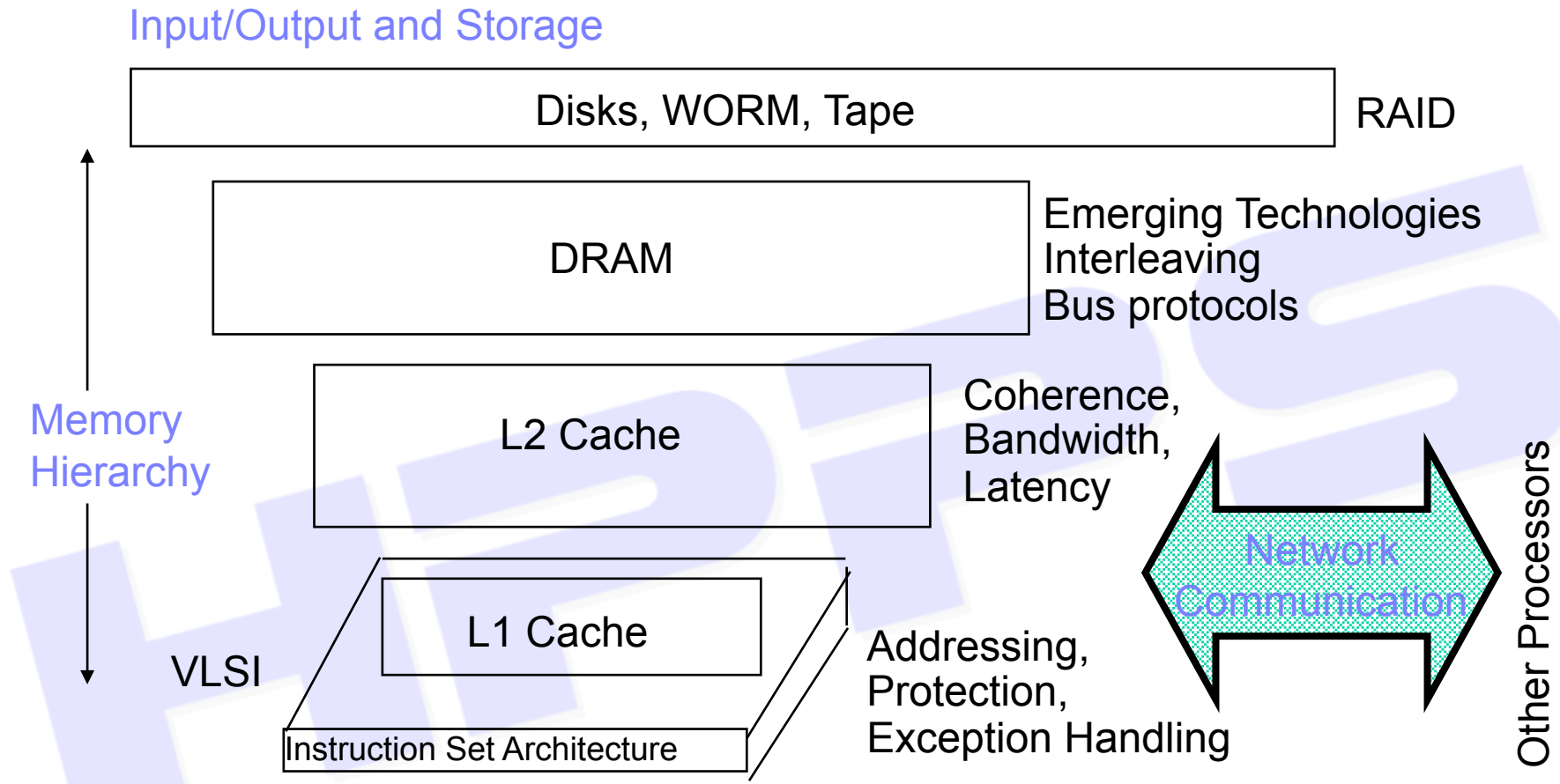
When all else fails - guess

- Programs make decisions as they go
  - Conditionals, loops, calls
  - Translate into branches and jumps (1 of 5 instructions)
- How do you determine what instructions for fetch when the ones before it haven't executed?
  - Branch prediction
  - Lot's of clever machine structures to predict future based on history
  - Machinery to back out of mis-predictions
- Execute all the possible branches
  - Likely to hit additional branches, perform stores

  ⇒speculative threads

  ⇒What can hardware do to make programming (with performance) easier?

Computer Architecture Topics



Computer Architecture Topics: Beyond ILP

- ILP architectures (superscalar, VLIW...):
  - Support fine-grained, instruction-level parallelism;
  - Fail to support large-scale parallel systems;
- Multiple-issue CPUs are very complex, and returns (as far as extracting greater parallelism) are diminishing
  - extracting parallelism at higher levels becomes more and more attractive.
- A further step: ***process- and thread-level parallel architectures.***
- To achieve ever greater performance: ***connect multiple microprocessors in a complex system.***

Computer Architecture Topics: Parallel Architectures

- Definition: “A parallel computer is a collection of processing elements that cooperates and communicate to solve large problems fast"

  Almasi and Gottlieb, Highly Parallel Computing, 1989

- The aim is to replicate processors to add performance vs design a faster processor.

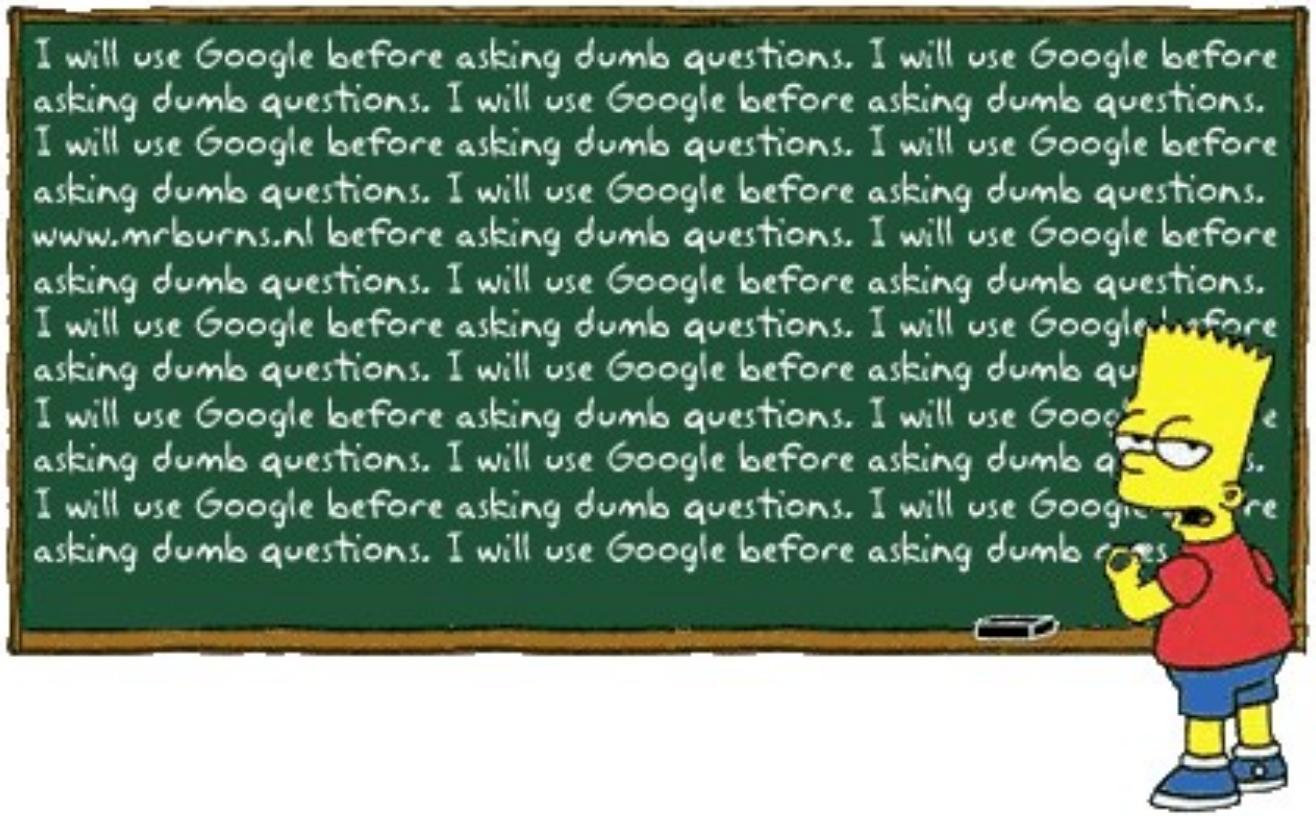
- Parallel architecture extends traditional computer architecture with a communication architecture
  - abstractions (HW/SW interface)
  - different structures to realize abstraction efficiently

Computer Architecture Topics: Self-Aware Computing

- Self-Aware Computing: systems that can observe their runtime behavior, learn, and take actions to meet desired goals

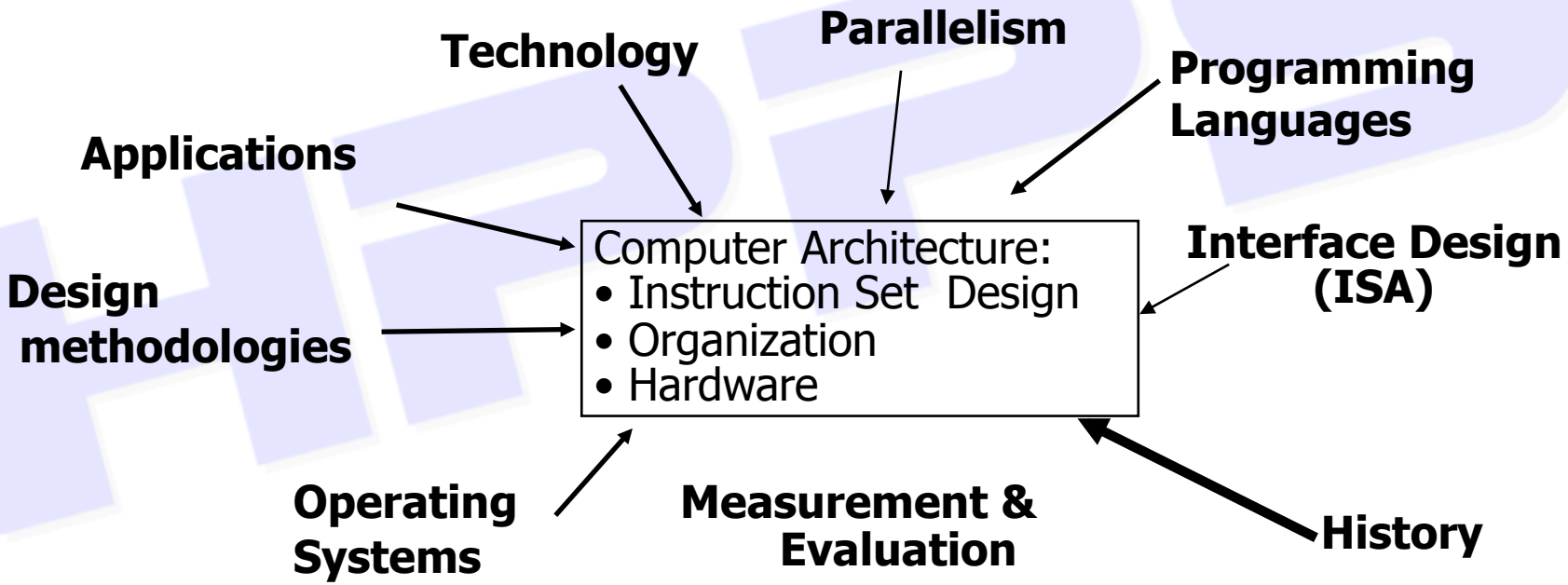
- Expected Impacts
  - Self-aware OS handles the complexity
    - Programmer specifies goals, the OS figures out how to meet them
    - System detects and handles errors as needed
  - Translate massive silicon resources into performance benefits
  - Increase software adaptability, it automatically optimizes for different platforms (i.e., remote briefing, adaptive audio system i.e., home theater)
  - System is dynamic and adaptable to changing requirements or operating conditions (i.e., cloud computing, mobile devices)

Class Organization



Course Focus

- Understand design techniques, machine structures, technology factors, evaluation methods, design of computer architectures



Topics

- Measuring performance
- Internal Parallelism in processors:
  - Pipelining
  - Instruction level parallelism inside processors
  - How to get more performance? Superscalar processors
- Process level parallelism
  - Thread-level
  - Multiprocessor architectures: introduction
- Cache memories
  - Very short review of addressing schemes
  - How to measure performance

Topic Coverage

- Textbook: Hennessy and Patterson, *Computer Architecture: A Quantitative Approach*, 4th Ed.

- Slides, papers and forums
  - Can be found in CORSIONLINE (http://corsi.metid.polimi.it/col/)

Lectures

- Tuesday: 3.30 pm – 17.00 pm – N12
- Thursday: 3.30 pm – 17.00 pm – EG3

- Special eventes
  - March 11: Project Proposals
  - May 4: No class – Official Project Hours
  - May 6: First Midterm
  - May 10 - 14: Project Reviews
  - June 15: Second Midterm
  - June 17 and 22: Project Presentations

Exams

- Written exam
  - Midterms:
    - 1st: May 6, 2010
    - 2nd: June 15, 2010
- Mark = 35% (1st midterm) + 35% (2nd midterm) +
  + 30% (projects)
- (!UIC mark) Possibility of oral exam on request (from the student or by us…)

Contacts and Office Hours

- Donatella Sciuto

  - Contact:
    - email: sciuto@elet.polimi.it
    - office: 3662 (DEI, first floor)
  - Office Hours:
    - Monday (after 2.30 pm), book an appointment via email

- Marco D. Santambrogio

  - Contact:
    - email: marco.santambrogio@polimi.it
    - skype: marco.santambrogio
    - office: 3492 (DEI, first floor)
  - Office Hours:
    - Whenever you want && I can

END?

Are you ready to see how deep the rabbit-hole goes?...