Computer Architecture

Course Information

- Course Homepage

  LMS system: https://lms.ncu.edu.tw/

- Instructor: 鄭旭詠

  e-mail : breeze.cheng@gmail.com

- Teaching Assistant:

  劉亞昇 anderson08121995@gmail.com

Course Information

Textbook

- J. L. Hennessy and D. A. Patterson, Computer Architecture: A Quantitative Approach, fifth Edition, Morgan Kaufmann Publishing Co.

- D. A. Patterson and J. L. Hennessy , Computer Organization & Design: The hardware/software interface, Morgan Kaufmann Publishing Co.

Grading Policy

- 15% Quiz and Class Participation
- 28% Mid-term1
- 28% Mid-term2
- 29% Final Examination

Reference Resources

- David Patterson
  - Electrical Engineering and Computer Sciences, University of California, Berkeley
  - http://www.eecs.berkeley.edu/~pattrsn
  - http://www-inst.eecs.berkeley.edu/~cs252

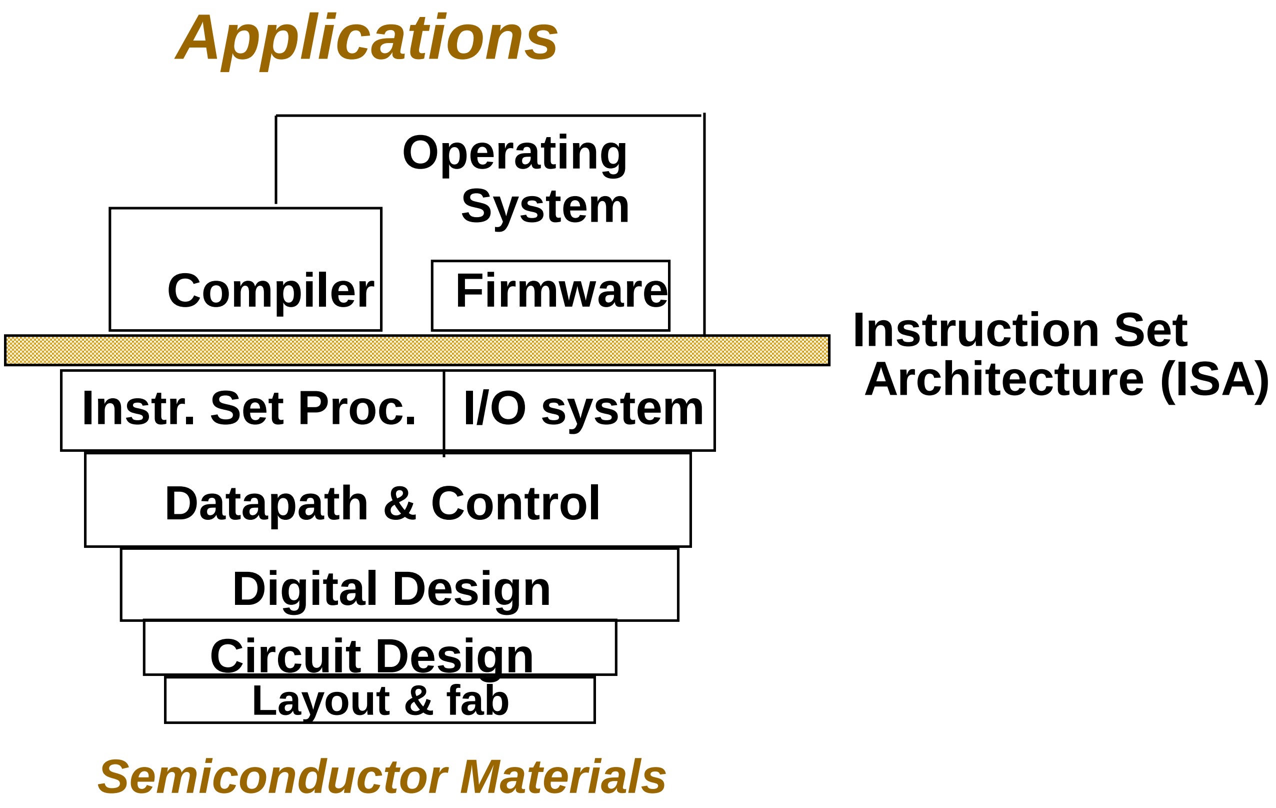
- Shih-Hao Hung, National Taiwan University

  Graduate Computer Architecture

  http://www.csie.ntu.edu.tw/~hungsh/CA/ca.htm

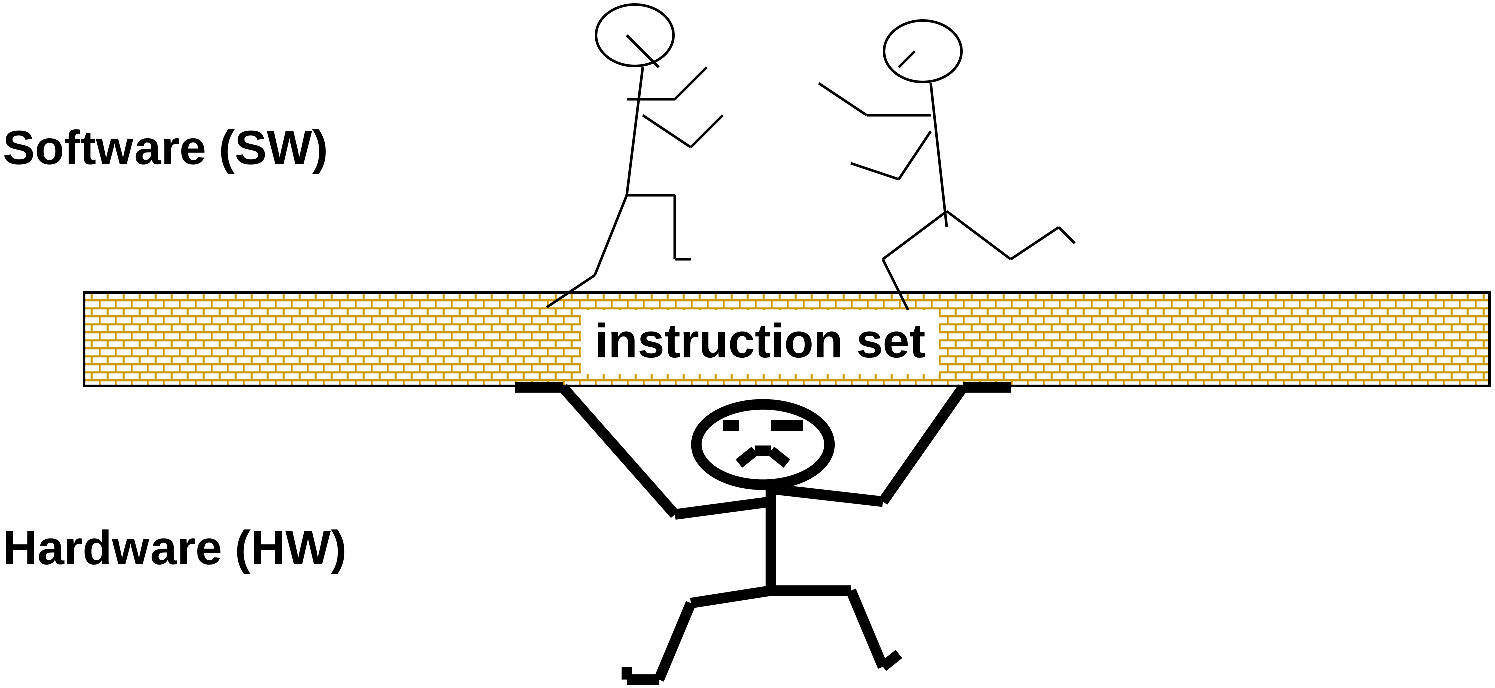
- David E. Culler, UC-Berkeley
  http://www.cs.berkeley.edu/~culler/courses/cs252-s05/

What is "Computer Architecture"?



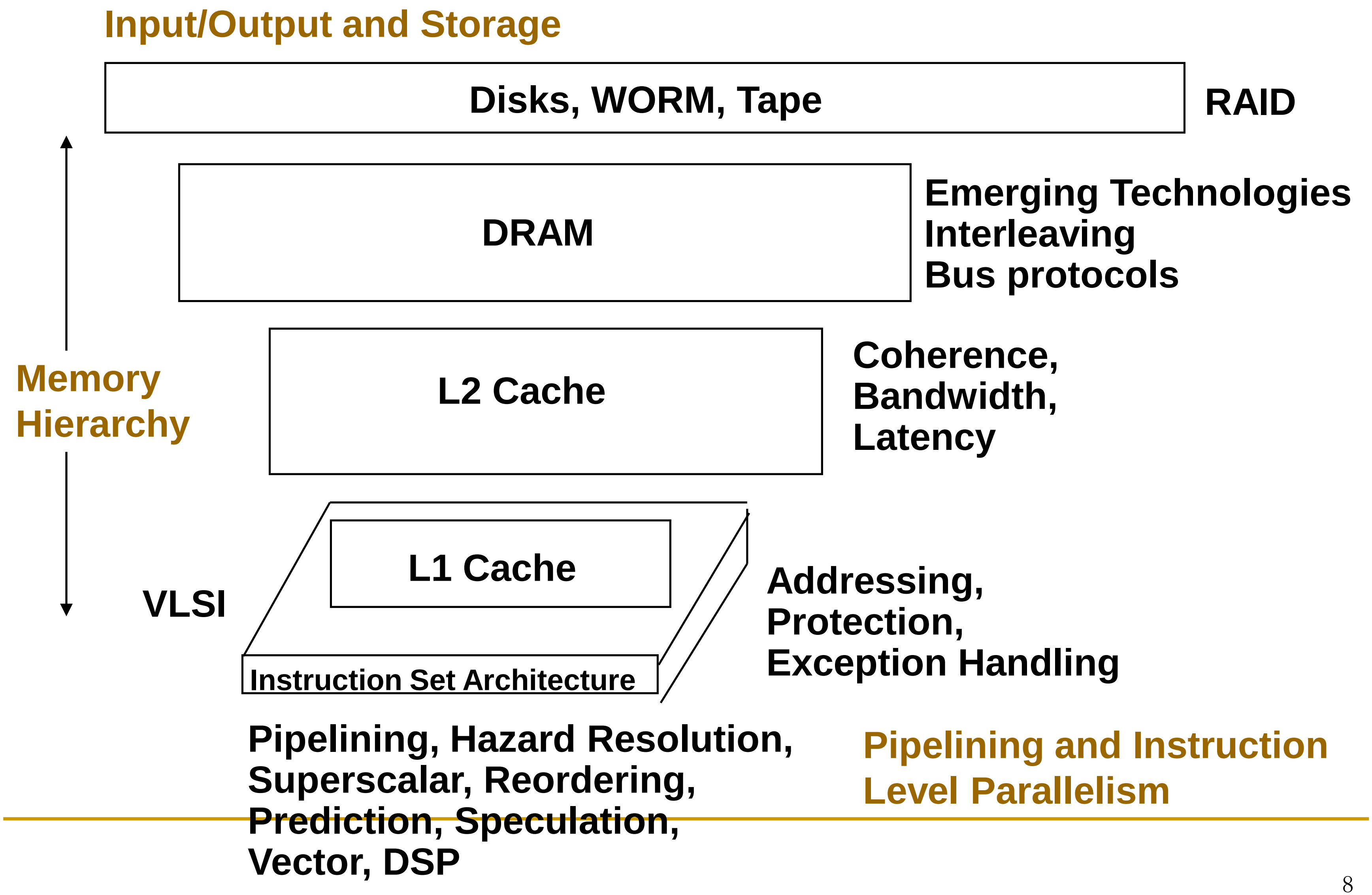
- Coordination of many *levels of abstraction*
- Under a rapidly changing set of forces
- Design, Measurement, *and* Evaluation

The Instruction Set: a Critical Interface



- Properties of a good abstraction
  - Lasts through many generations (portability)
  - Used in many different ways (generality)
  - Provides convenient functionality to higher levels
  - Permits an efficient implementation at lower levels

Computer Architecture Topics



Input/Output and Storage

Disks, WORM, Tape

RAID

DRAM

Emerging Technologies
Interleaving
Bus protocols

Memory
Hierarchy

L2 Cache

Coherence,
Bandwidth,
Latency

L1 Cache

VLSI

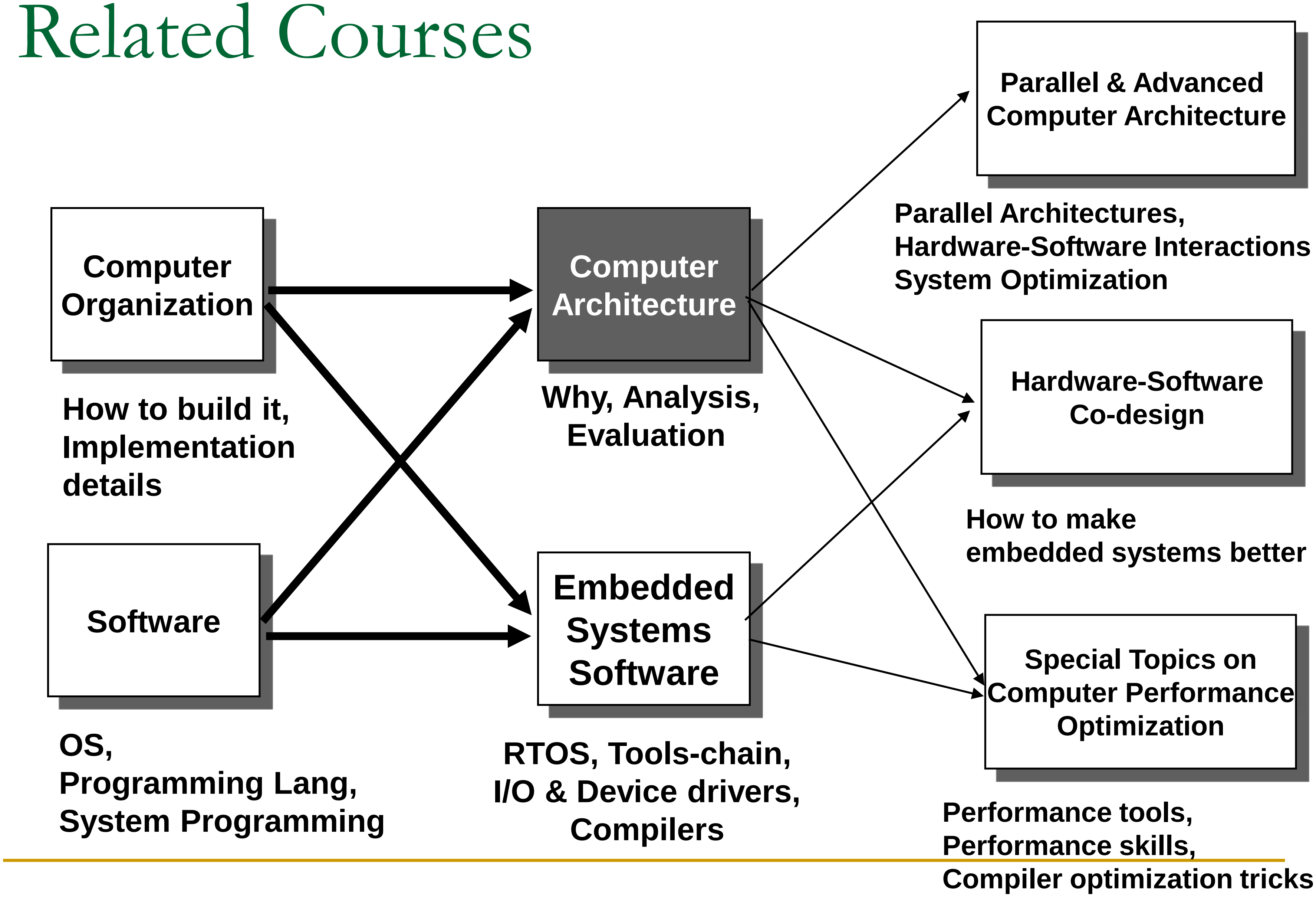
Instruction Set Architecture

Addressing,
Protection,
Exception Handling

Pipelining, Hazard Resolution,
Superscalar, Reordering,
Prediction, Speculation,
Vector, DSP

Pipelining and Instruction
Level Parallelism

Related Courses



Computer Organization

How to build it, Implementation details

Software

OS, Programming Lang, System Programming

Computer Architecture

Why, Analysis, Evaluation

Embedded Systems Software

RTOS, Tools-chain, I/O & Device drivers, Compilers

Parallel & Advanced Computer Architecture

Parallel Architectures, Hardware-Software Interactions System Optimization

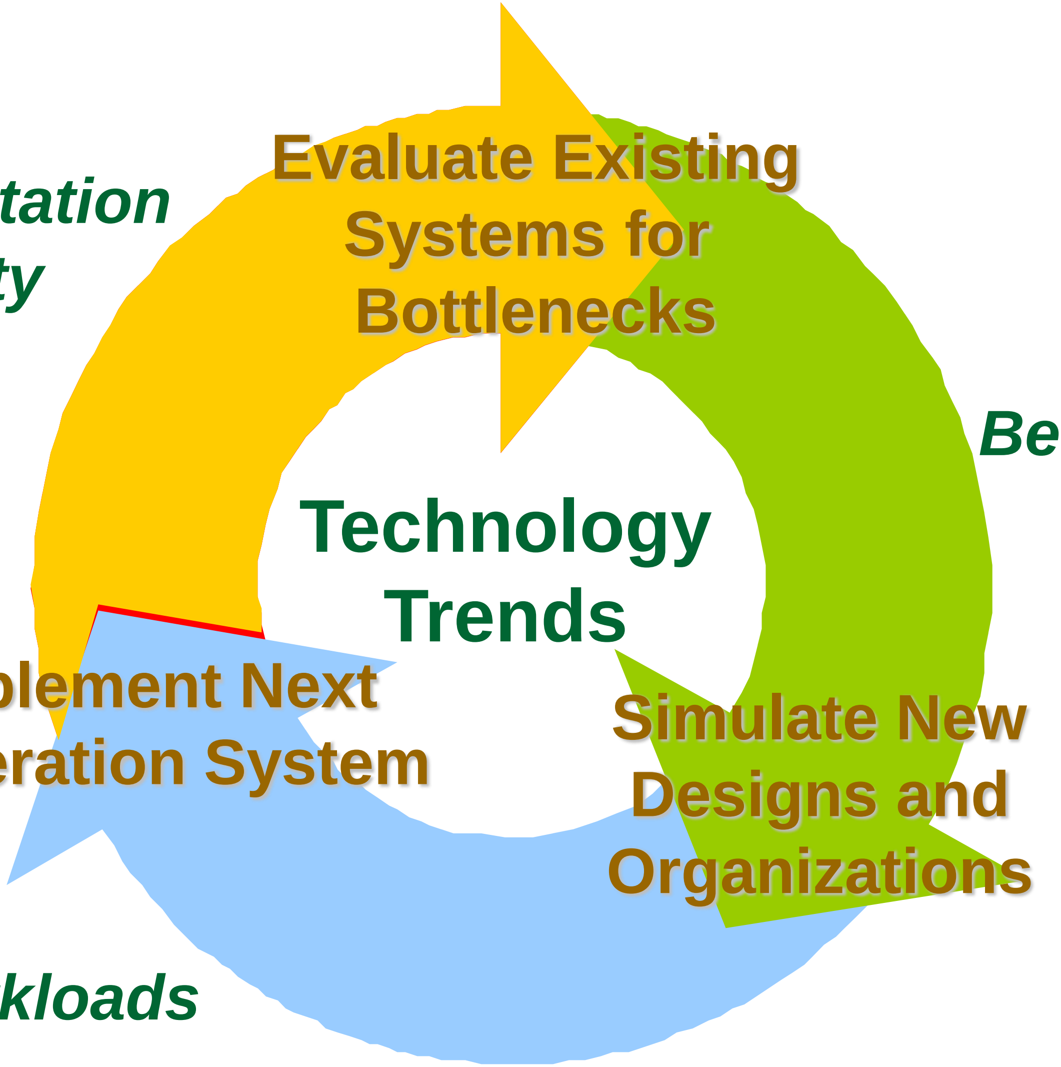
Hardware-Software Co-design

How to make embedded systems better

Special Topics on Computer Performance Optimization

Performance tools, Performance skills, Compiler optimization tricks

Computer Engineering Methodology



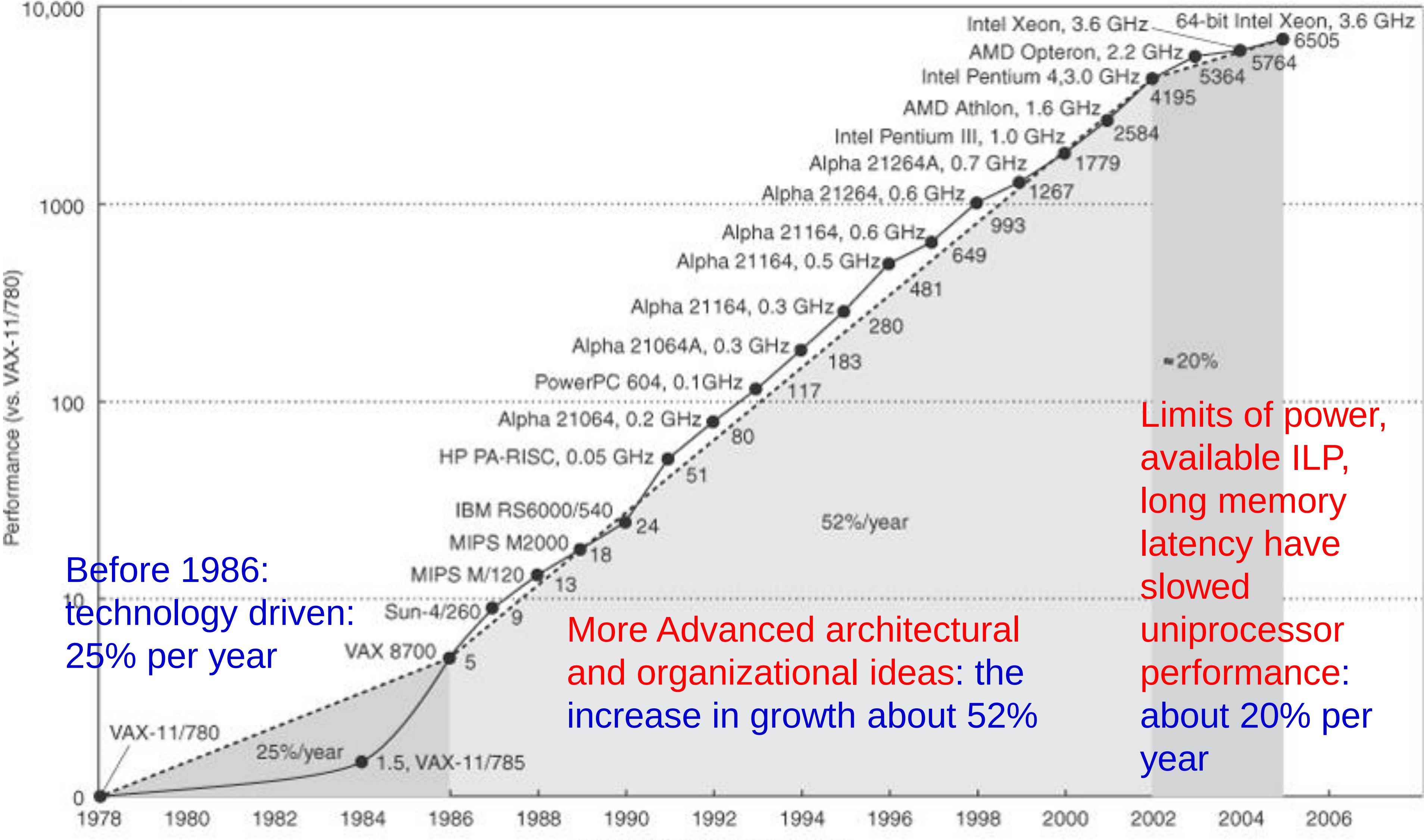
Evaluate Existing Systems for Bottlenecks
Technology Trends
Simulate New Designs and Organizations
Implement Next Generation System

Implementation Complexity

Benchmarks

Workloads

Growth in Process Performance since the mid-1980s



Before 1986: technology driven: 25% per year

More Advanced architectural and organizational ideas: the increase in growth about 52%

Limits of power, available ILP, long memory latency have slowed uniprocessor performance: about 20% per year

© 2007 Elsevier, Inc. All rights reserved.

Crossroads: Conventional Wisdom in Comp. Arch

- Old Conventional Wisdom: Power is free, Transistors expensive
- New Conventional Wisdom: “Power wall” Power expensive (Maximum power dissipation of air-cooled chips)

- Old CW: Sufficiently increasing Instruction Level Parallelism via compilers, innovation (Out-of-order, speculation, VLIW, …)
- New CW: “ILP wall” law of diminishing returns on more HW for ILP

- Old CW: Multiplies are slow, Memory access is fast
- New CW: “Memory wall” Memory slow, multiplies fast (200 clock cycles to DRAM memory, 4 clocks for multiply)

- Old CW: Uni-processor performance 2X / 3 yrs
- New CW: multiple “cores”

Crossroads

- In 2004, Intel canceled its high-performance uni-processor projects and joined IBM and Sun
- Declaring that the road to higher performance would be via multiple processor per chip rather than faster uni-processors
- Little Instruction Level Parallelism (ILP) left to exploit efficiently => switch to Thread Level Parallelism (TLP) and Data Level Parallelism (DLP)

- ILP: Compiler and hardware conspire to exploit ILP implicitly without programmer's attention
- TLP and DLP: explicitly parallel, requiring the programmer to write parallel code to gain performance

Syllabus

2. 23	Course Overview
3. 2	Fundamentals of Computer Design
3. 9	Fundamentals of Computer Design
3. 16	Instruction Set Principles and Pipelining
3. 23	Advanced Pipelining and Instruction Level Parallelism (ILP)
3. 30	Midterm 1
4. 6	校際活動週
4. 13	Advanced Pipelining and ILP, scoreboard
4. 20	Scoreboard, Tomasulo
4. 27	Branch Prediction
5. 4	Multi-Processers
5. 11	Midterm 2
5. 18	Memory Hierarchy
5. 25	Memory Hierarchy, Cache
6. 1	Cache, Virtual Memory
6. 8	I/O, Storage
6. 15	Final Exam