
EEC 581 Computer Architecture

Lec 1 - Introduction

Chansu Yu
Electrical and Computer Engineering
Cleveland State University

Acknowledgement ...

•Class notes are from

- David Patterson
- Electrical Engineering and Computer Sciences
- University of California, Berkeley

- <http://www.eecs.berkeley.edu/~pattsn>
- <http://www-inst.eecs.berkeley.edu/~cs252>

Outline

- **Computer Science at a Crossroads**
- **Computer Architecture v. Instruction Set Arch.**
- **What Computer Architecture brings to table**

8/30/2007

3

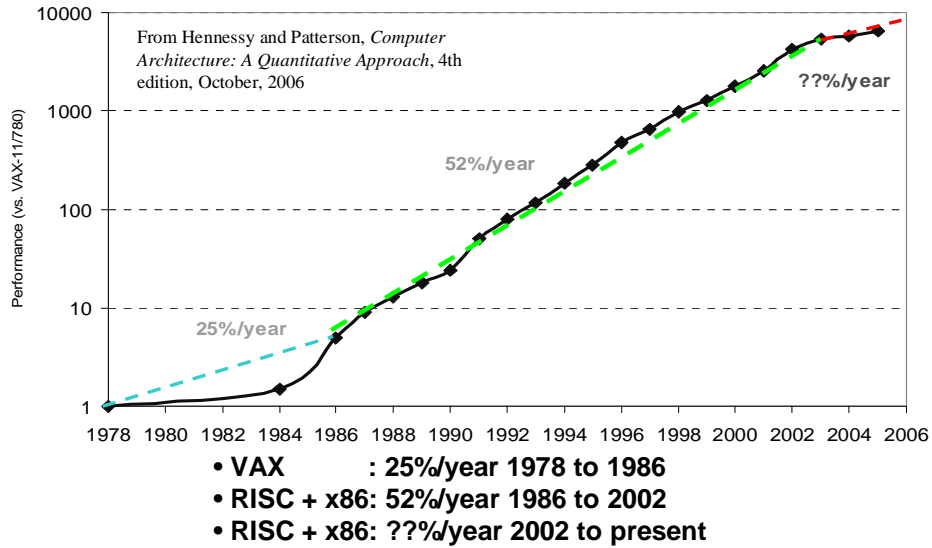
Crossroads: Conventional Wisdom in Comp. Arch

- **Old Conventional Wisdom: Power is free, Transistors expensive**
 - **New Conventional Wisdom: “Power wall” Power expensive, Xtors free (Can put more on chip than can afford to turn on)**
 - **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: Uniprocessor performance 2X / 1.5 yrs**
 - **New CW: Power Wall + ILP Wall + Memory Wall = Brick Wall**
 - Uniprocessor performance now 2X / 5(?) yrs
- ⇒ **Sea change in chip design: multiple “cores” (2X processors per chip / ~ 2 years)**
- » More simpler processors are more power efficient

8/30/2007

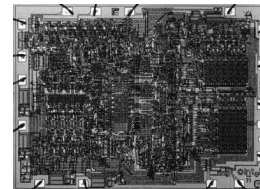
4

Crossroads: Uniprocessor Performance



Sea Change in Chip Design

- Intel 4004 (1971): 4-bit processor, 2312 transistors, 0.4 MHz, 10 micron PMOS, 11 mm² chip
- RISC II (1983): 32-bit, 5 stage pipeline, 40,760 transistors, 3 MHz, 3 micron NMOS, 60 mm² chip
- 125 mm² chip, 0.065 micron CMOS = 2312 RISC II+FPU+lcache+Dcache
 - RISC II shrinks to ~ 0.02 mm² at 65 nm
 - Caches via DRAM or 1 transistor SRAM (www.t-ram.com) ?
 - Proximity Communication via capacitive coupling at > 1 TB/s ? (Ivan Sutherland @ Sun / Berkeley)



• Processor is the new transistor?

8/30/2007

6

Déjà vu all over again?

- **Multiprocessors imminent in 1970s, '80s, '90s, ...**
- **"... today's processors ... are nearing an impasse as technologies approach the speed of light.."**

David Mitchell, *The Transputer: The Time Is Now* (1989)
- **Transputer was premature**
 - ⇒ Custom multiprocessors strove to lead uniprocessors
 - ⇒ Procrastination rewarded: 2X seq. perf. / 1.5 years
- **"We are dedicating all of our future product development to multicore designs. ... This is a sea change in computing"**

Paul Otellini, President, Intel (2004)
- **Difference is all microprocessor companies switch to multiprocessors (AMD, Intel, IBM, Sun; all new Apples 2 CPUs)**
 - ⇒ Procrastination penalized: 2X sequential perf. / 5 yrs
 - ⇒ Biggest programming challenge: 1 to 2 CPUs

8/30/2007

7

Problems with Sea Change

- **Algorithms, Programming Languages, Compilers, Operating Systems, Architectures, Libraries, ... not ready to supply Thread Level Parallelism or Data Level Parallelism for 1000 CPUs / chip,**
- **Architectures not ready for 1000 CPUs / chip**
 - Unlike Instruction Level Parallelism, cannot be solved by just by computer architects and compiler writers alone, but also cannot be solved *without* participation of computer architects
- **This edition (4th Edition of textbook *Computer Architecture: A Quantitative Approach*) explores shift from Instruction Level Parallelism to Thread Level Parallelism / Data Level Parallelism**

8/30/2007

8

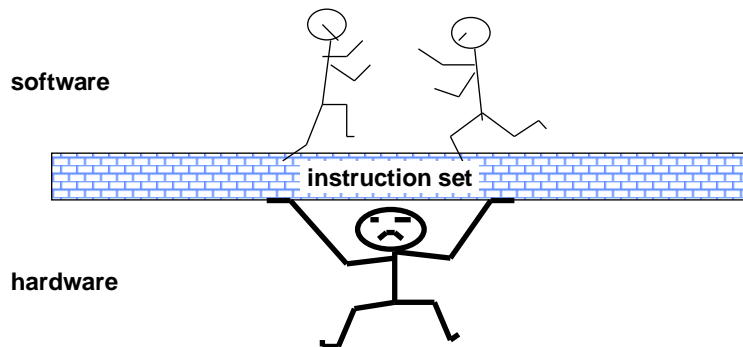
Outline

- Computer Science at a Crossroads
- **Computer Architecture v. Instruction Set Arch.**
- **What Computer Architecture brings to table**

8/30/2007

9

Instruction Set Architecture: Critical Interface

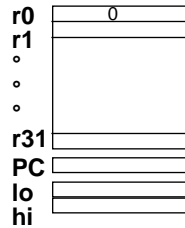


- **Properties of a good abstraction**
 - Lasts through many generations (portability)
 - Used in many different ways (generality)
 - Provides convenient functionality to higher levels
 - Permits an efficient implementation at lower levels

8/30/2007

10

Example: MIPS



Programmable storage

2^{32} x bytes
 31 x 32-bit GPRs (R0=0)
 32 x 32-bit FP regs (paired DP)
 HI, LO, PC

Data types ?

Format ?

Addressing Modes?

Arithmetic logical

Add, AddU, Sub, SubU, And, Or, Xor, Nor, SLT, SLTU,
 Addl, AddIU, SLTI, SLTIU, Andl, Orl, Xorl, *LUI*
 SLL, SRL, SRA, SLLV, SRLV, SRAV

Memory Access

LB, LBU, LH, LHU, LW, LWL, LWR
 SB, SH, SW, SWL, SWR

Control

J, JAL, JR, JALR
 BEq, BNE, BLEZ, BGTZ, BLTZ, BGEZ, BLTZAL, BGEZAL

32-bit instructions on word boundary

8/30/2007

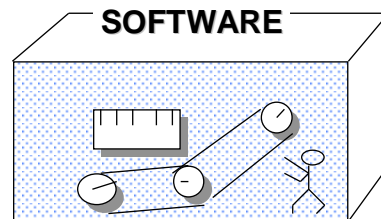
11

Instruction Set Architecture

"... 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, Blaauw, and Brooks, 1964

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



8/30/2007

12

ISA vs. Computer Architecture

- **Old definition of computer architecture = instruction set design**
 - Other aspects of computer design called implementation
 - Insinuates implementation is uninteresting or less challenging
- **Our view is computer architecture >> ISA**
- **Architect's job much more than instruction set design; technical hurdles today *more* challenging than those in instruction set design**
- **Since instruction set design not where action is, some conclude computer architecture (using old definition) is not where action is**
 - We disagree on conclusion
 - Agree that ISA not where action is (ISA in CA:AQA 4/e appendix)

8/30/2007

13

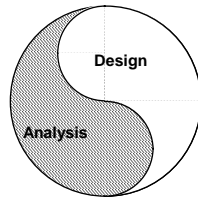
Comp. Arch. is an Integrated Approach

- **What really matters is the functioning of the complete system**
 - hardware, runtime system, compiler, operating system, and application
 - In networking, this is called the “End to End argument”
- **Computer architecture is not just about transistors, individual instructions, or particular implementations**
 - E.g., Original RISC projects replaced complex instructions with a compiler + simple instructions

8/30/2007

14

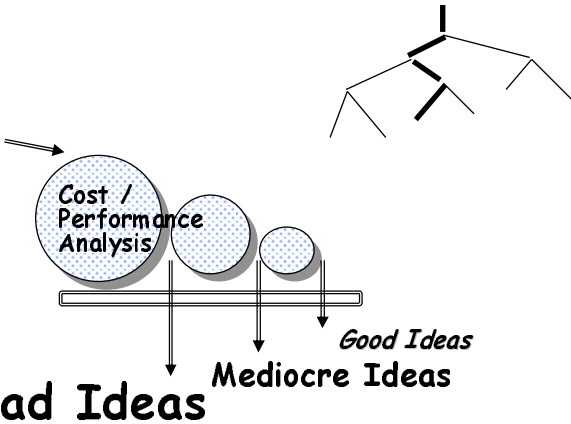
Computer Architecture is Design and Analysis



Architecture is an iterative process:

- Searching the space of possible designs
- At all levels of computer systems

Creativity



8/30/2007

15

Outline

- Computer Science at a Crossroads
- Computer Architecture v. Instruction Set Arch.
- **What Computer Architecture brings to table**

8/30/2007

16

What Computer Architecture brings to Table

- **Other fields often borrow ideas from architecture**
- **Quantitative Principles of Design**
 1. Take Advantage of Parallelism
 2. Principle of Locality
 3. Focus on the Common Case
 4. Amdahl's Law
 5. The Processor Performance Equation
- **Careful, quantitative comparisons**
 - Define, quantity, and summarize relative performance
 - Define and quantity relative cost
 - Define and quantity dependability
 - Define and quantity power
- **Culture of anticipating and exploiting advances in technology**
- **Culture of well-defined interfaces that are carefully implemented and thoroughly checked**

8/30/2007

17

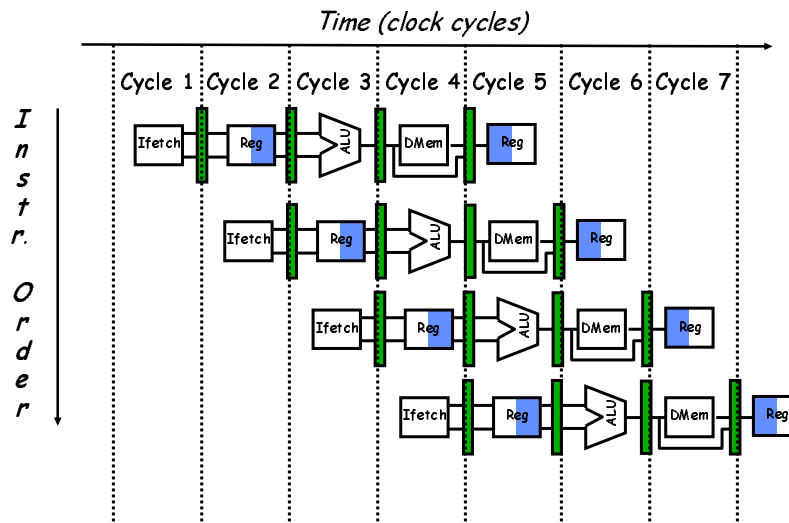
1) Taking Advantage of Parallelism

- **Increasing throughput of server computer via multiple processors or multiple disks**
- **Detailed HW design**
 - Carry lookahead adders uses parallelism to speed up computing sums from linear to logarithmic in number of bits per operand
 - Multiple memory banks searched in parallel in set-associative caches
- **Pipelining: overlap instruction execution to reduce the total time to complete an instruction sequence.**
 - Not every instruction depends on immediate predecessor ⇒ executing instructions completely/partially in parallel possible
 - Classic 5-stage pipeline:
 - 1) Instruction Fetch (Ifetch),
 - 2) Register Read (Reg),
 - 3) Execute (ALU),
 - 4) Data Memory Access (Dmem),
 - 5) Register Write (Reg)

8/30/2007

18

Pipelined Instruction Execution

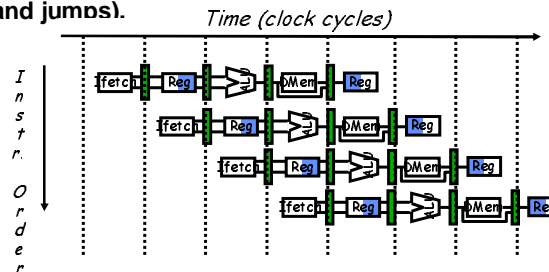


8/30/2007

19

Limits to pipelining

- 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).

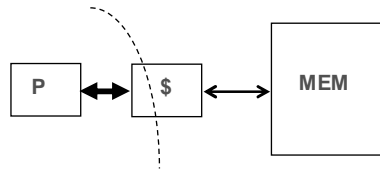


8/30/2007

20

2) The Principle of Locality

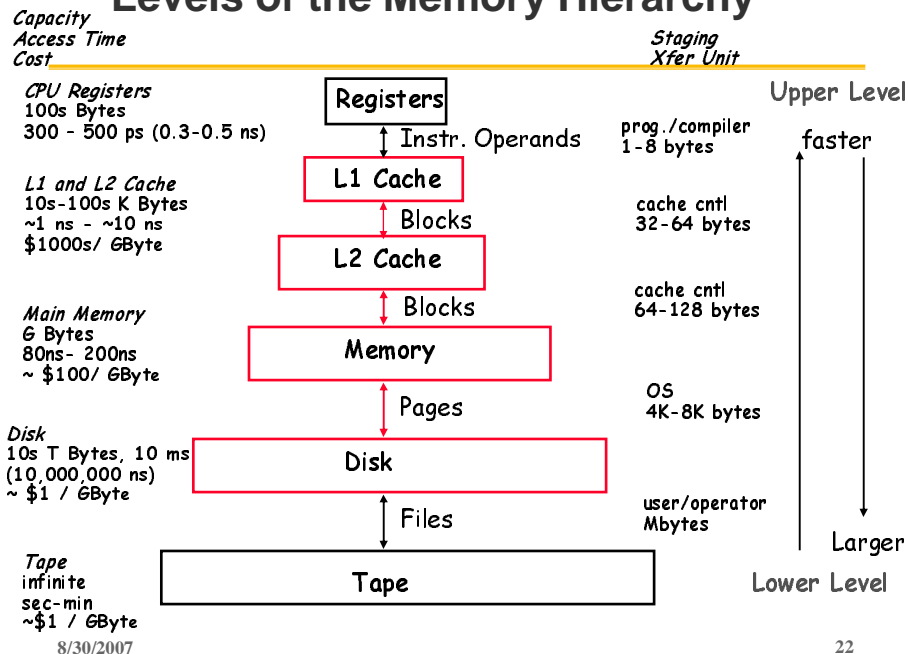
- **The Principle of Locality:**
 - Program access a relatively small portion of the address space at any instant of time.
- **Two Different Types of Locality:**
 - **Temporal Locality** (Locality in Time): If an item is referenced, it will tend to be referenced again soon (e.g., loops, reuse)
 - **Spatial Locality** (Locality in Space): If an item is referenced, items whose addresses are close by tend to be referenced soon (e.g., straight-line code, array access)
- **Last 30 years, HW relied on locality for memory perf.**



8/30/2007

21

Levels of the Memory Hierarchy



8/30/2007

22

3) Focus on the Common Case

- **Common sense guides computer design**
 - Since its engineering, common sense is valuable
- **In making a design trade-off, favor the frequent case over the infrequent case**
 - E.g., Instruction fetch and decode unit used more frequently than multiplier, so optimize it 1st
 - E.g., If database server has 50 disks / processor, storage dependability dominates system dependability, so optimize it 1st
- **Frequent case is often simpler and can be done faster than the infrequent case**
 - E.g., overflow is rare when adding 2 numbers, so improve performance by optimizing more common case of no overflow
 - May slow down overflow, but overall performance improved by optimizing for the normal case
- **What is frequent case and how much performance improved by making case faster => Amdahl's Law**

8/30/2007

23

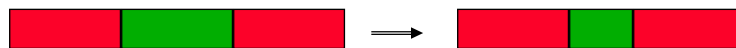
4) Amdahl's Law

$$\text{ExTime}_{\text{new}} = \text{ExTime}_{\text{old}} \times \left[(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}} \right]$$

$$\text{Speedup}_{\text{overall}} = \frac{\text{ExTime}_{\text{old}}}{\text{ExTime}_{\text{new}}} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

Best you could ever hope to do:

$$\text{Speedup}_{\text{maximum}} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}})}$$



8/30/2007

24

Amdahl's Law example

- New CPU 10X faster
- I/O bound server, so 60% time waiting for I/O

$$\text{Speedup}_{\text{overall}} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

$$= \frac{1}{(1 - 0.4) + \frac{0.4}{10}} = \frac{1}{0.64} = 1.56$$

- Apparently, its human nature to be attracted by 10X faster, vs. keeping in perspective its just 1.6X faster

8/30/2007

25

5) Processor performance equation



inst count Cycle time

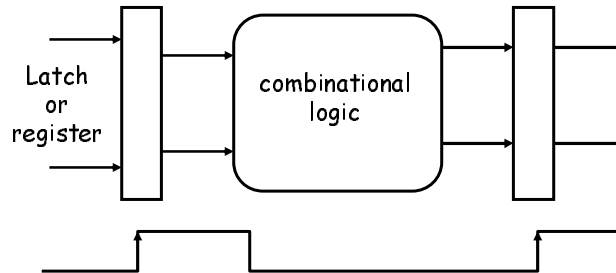
$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

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

8/30/2007

26

What's a Clock Cycle?



- **Old days: 10 levels of gates**
- **Today: determined by numerous time-of-flight issues + gate delays**
 - clock propagation, wire lengths, drivers

8/30/2007

27

And in conclusion ...

- **Computer Architecture >> instruction sets**
- **Computer Architecture skill sets are different**
 - 5 Quantitative principles of design
 - Quantitative approach to design
 - Solid interfaces that really work
 - Technology tracking and anticipation
- **EEC 581 to learn new skills, transition to research**
- **Computer Science at the crossroads from sequential to parallel computing**
 - Salvation requires innovation in many fields, including computer architecture
- **Read Chapter 1, then Appendix A !**

8/30/2007

28