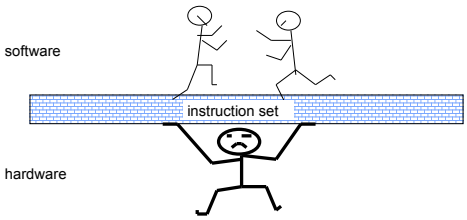
CSE 490/590 Computer Architecture

Intro II & Trends

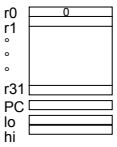
Steve Ko
Computer Sciences and Engineering
University at Buffalo

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

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, AddI, AddIU, SLTI, SLTIU, AndI, OrI, XorI, *LUI*
- SLL, SRL, SRA, SLLV, SRLV, SRAV

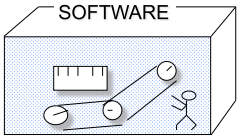
Memory Access
- LB, LBU, LH, LHU, LW, LWL,LWR
- SB, SH, SW, SWL, SWR

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

32-bit instructions on word boundary

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

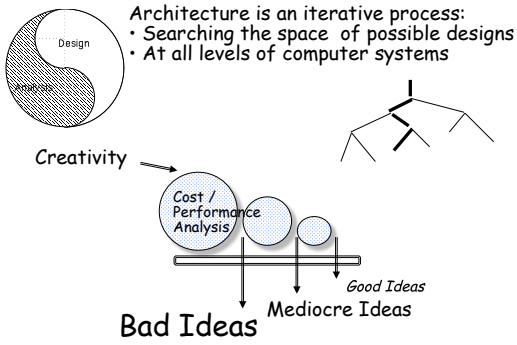
ISA vs. Computer Architecture

- Old definition of computer architecture = instruction set design
  - Other aspects of computer design called implementation
  - Insinuates implementation is uninteresting or less challenging
- New 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

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

Computer Architecture is Design and Analysis



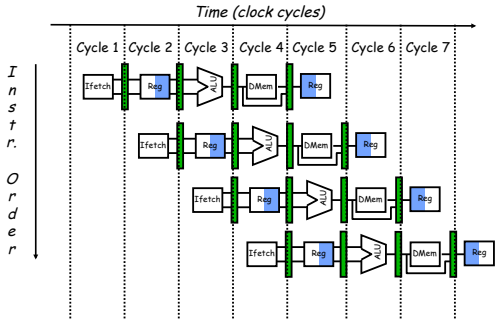
What Computer Architecture brings to Table

- 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, quantify, and summarize relative performance
  - Define and quantify relative cost
  - Define and quantify dependability
  - Define and quantify power

1) Taking Advantage of Parallelism

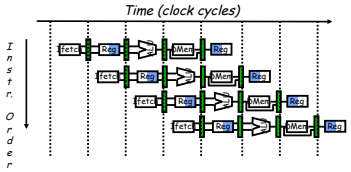
- Increasing throughput of server computer via multiple processors or multiple disks
- Detailed HW design
  - Carry lookahead adders uses parallelism to speed up computing sums from linear to logarithmic in number of bits per operand
  - Multiple memory banks searched in parallel in set-associative caches
- 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)

Pipelined Instruction Execution



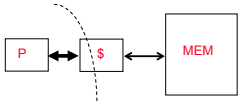
Limits to pipelining

- 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).

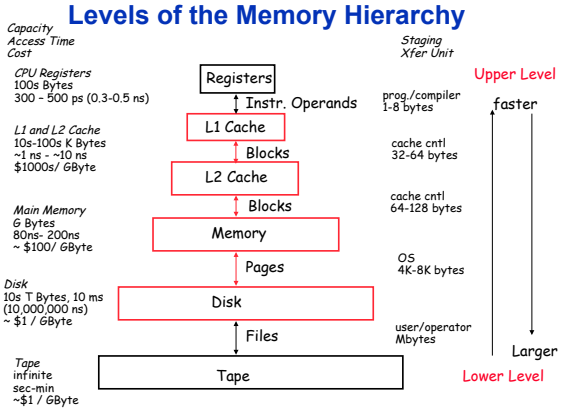


2) The Principle of Locality

- 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.



Levels of the Memory Hierarchy



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

4) Amdahl's Law

$$ExTime_{new} = ExTime_{old} \times \left[\left(1 - Fraction_{enhanced}\right) + \frac{Fraction_{enhanced}}{Speedup_{enhanced}} \right]$$

$$Speedup_{overall} = \frac{ExTime_{old}}{ExTime_{new}} = \frac{1}{\left(1 - Fraction_{enhanced}\right) + \frac{Fraction_{enhanced}}{Speedup_{enhanced}}}$$

Best you could ever hope to do:

$$Speedup_{maximum} = \frac{1}{\left(1 - Fraction_{enhanced}\right)}$$

Amdahl's Law example

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

$$Speedup_{overall} = \frac{1}{\left(1 - Fraction_{enhanced}\right) + \frac{Fraction_{enhanced}}{Speedup_{enhanced}}}$$

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

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

5) Processor performance equation

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

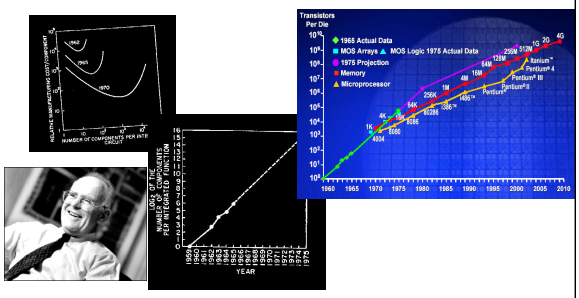
	Inst Count	CPI	Clock Rate
Program	X		
Compiler	X	(X)	
Inst. Set.	X	X	
Organization	X		X
Technology			X

CSE 490/590 Administrivia

- Jangyoung Kim's Office Hours
  - Office: 232 Bell
  - Office Hours: MWF, 1pm – 2pm
- Please check the web page: http://www.cse.buffalo.edu/~stevko/courses/cse490/spring11
- Don't forget: Recitations start from next week!
- Disclaimer: Lecture notes are heavily based on UC Berkeley's materials.

Technology Trends

Moore's Law: 2X transistors / "year"



- "Cramming More Components onto Integrated Circuits"
  - Gordon Moore, Electronics, 1965
- # on transistors / cost-effective integrated circuit double every N months (12 ≤ N ≤ 24)

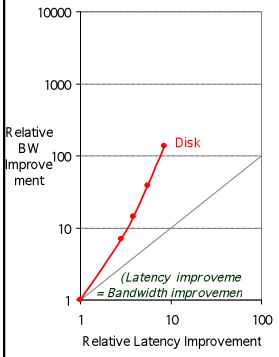
Tracking Technology Performance Trends

- Drill down into 4 technologies:
  - Disks,
  - Memory,
  - Network,
  - Processors
- Compare ~1980 Archaic (Nostalgic) vs. ~2000 Modern (Newfangled)
  - Performance Milestones in each technology
- Compare for Bandwidth vs. Latency improvements in performance over time
- Bandwidth: number of events per unit time
  - E.g., M bits / second over network, M bytes / second from disk
- Latency: elapsed time for a single event
  - E.g., one-way network delay in microseconds, average disk access time in milliseconds

Disks: Archaic(Nostalgic) v. Modern(Newfangled)

• CDC Wren I, 1983	• Seagate 373453, 2003	
• 3600 RPM	• 15000 RPM	(4X)
• 0.03 GBytes capacity	• 73.4 GBytes	(2500X)
• Tracks/Inch: 800	• Tracks/Inch: 64000	(80X)
• Bits/Inch: 9550	• Bits/Inch: 533,000	(60X)
• Three 5.25" platters	• Four 2.5" platters (in 3.5" form factor)	
• Bandwidth: 0.6 MBytes/sec	• Bandwidth: 86 MBytes/sec	(140X)
• Latency: 48.3 ms	• Latency: 5.7 ms	(8X)
• Cache: none	• Cache: 8 MBytes	

Latency Lags Bandwidth (for last ~20 years)

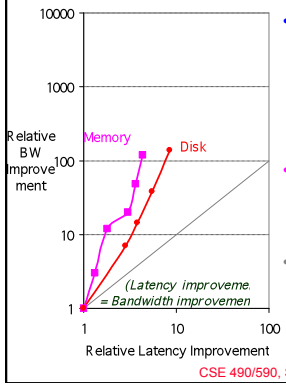


- Performance Milestones
- Disk: 3600, 5400, 7200, 10000, 15000 RPM (8x, 143x)
  (latency = simple operation w/o contention
  BW = best-case)

Memory: Archaic (Nostalgic) v. Modern (Newfangled)

• 1980 DRAM (asynchronous)	• 2000 Double Data Rate Synchr. (clocked) DRAM	
• 0.06 Mbits/chip	• 256.00 Mbits/chip	(4000X)
• 64,000 xtors, 35 mm^2	• 256,000,000 xtors, 204 mm^2	
• 16-bit data bus per module, 16 pins/chip	• 64-bit data bus per DIMM, 66 pins/chip	(4X)
• 13 Mbytes/sec	• 1600 Mbytes/sec	(120X)
• Latency: 225 ns	• Latency: 52 ns	(4X)
• (no block transfer)	• Block transfers (page mode)	

Latency Lags Bandwidth (last ~20 years)



- Performance Milestones
- Memory Module: 16bit plain DRAM, Page Mode DRAM, 32b, 64b, SDRAM, DDR SDRAM (4x,120x)
- Disk: 3600, 5400, 7200, 10000, 15000 RPM (8x, 143x)

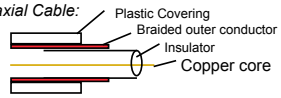
(latency = simple operation w/o contention
BW = best-case)

LANs: Archaic (Nostalgic)v. Modern (Newfangled)

- Ethernet 802.3
- Year of Standard: 1978
- 10 Mbits/s link speed
- Latency: 3000 μsec
- Shared media
- Coaxial cable

- Ethernet 802.3ae
- Year of Standard: 2003
- 10,000 Mbits/s link speed (1000X)
- Latency: 190 μsec (15X)
- Switched media
- Category 5 copper wire

Coaxial Cable:

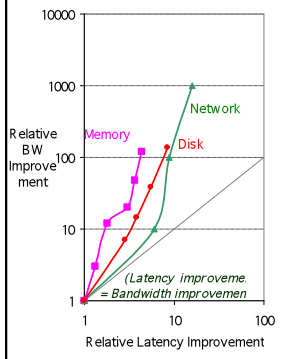


"Cat 5" is 4 twisted pairs in bundle

Twisted Pair:

Copper, 1mm thick, twisted to avoid antenna effect

Latency Lags Bandwidth (last ~20 years)

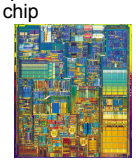


- Performance Milestones
- Ethernet: 10Mb, 100Mb, 1000Mb, 10000 Mb/s (16x,1000x)
- Memory Module: 16bit plain DRAM, Page Mode DRAM, 32b, 64b, SDRAM, DDR SDRAM (4x,120x)
- Disk: 3600, 5400, 7200, 10000, 15000 RPM (8x, 143x)

(latency = simple operation w/o contention
BW = best-case)

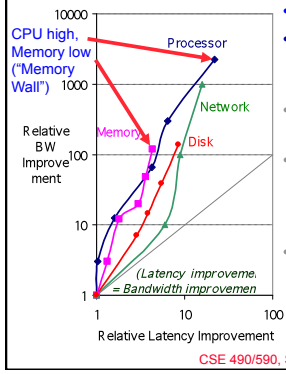
CPUs: Archaic (Nostalgic) v. Modern (Newfangled)

- 1982 Intel 80286
- 12.5 MHz
- 2 MIPS (peak)
- Latency 320 ns
- 134,000 xtors, 47 mm^2
- 16-bit data bus, 68 pins
- Microcode interpreter, separate FPU chip
- (no caches)



- 2001 Intel Pentium 4
- 1500 MHz (120X)
- 4500 MIPS (peak) (2250X)
- Latency 15 ns (20X)
- 42,000,000 xtors, 217 mm^2
- 64-bit data bus, 423 pins
- 3-way superscalar, Dynamic translate to RISC, Superpipelined (22 stage), Out-of-Order execution
- On-chip 8KB Data caches, 96KB Instr. Trace cache, 256KB L2 cache

Latency Lags Bandwidth (last ~20 years)



- Performance Milestones
- Processor: '286, '386, '486, Pentium, Pentium Pro, Pentium 4 (21x,2250x)
- Ethernet: 10Mb, 100Mb, 1000Mb, 10000 Mb/s (16x,1000x)
- Memory Module: 16bit plain DRAM, Page Mode DRAM, 32b, 64b, SDRAM, DDR SDRAM (4x,120x)
- Disk : 3600, 5400, 7200, 10000, 15000 RPM (8x, 143x)

Rule of Thumb for Latency Lagging BW

- In the time that bandwidth doubles, latency improves by no more than a factor of 1.2 to 1.4
  (and capacity improves faster than bandwidth)
- Stated alternatively:
  Bandwidth improves by more than the square of the improvement in Latency

6 Reasons Latency Lags Bandwidth

1. Moore's Law helps BW more than latency
   - Faster transistors, more transistors, more pins help Bandwidth
     - » MPU Transistors: 0.130 vs. 42 M xtors (300X)
     - » DRAM Transistors: 0.064 vs. 256 M xtors (4000X)
     - » MPU Pins: 68 vs. 423 pins (6X)
     - » DRAM Pins: 16 vs. 66 pins (4X)
   - Smaller, faster transistors but communicate over (relatively) longer lines: limits latency
     - » Feature size: 1.5 to 3 vs. 0.18 micron (8X,17X)
     - » MPU Die Size: 35 vs. 204 mm^2 (ratio sqrt ⇒ 2X)
     - » DRAM Die Size: 47 vs. 217 mm^2 (ratio sqrt ⇒ 2X)

6 Reasons Latency Lags Bandwidth (cont'd)

2. Distance limits latency
   - Size of DRAM block ⇒ long bit and word lines ⇒ most of DRAM access time
   - Speed of light and computers on network
   - 1. & 2. explains linear latency vs. square BW?
3. Bandwidth easier to sell ("bigger=better")
   - E.g., 10 Gbits/s Ethernet ("10 Gig") vs. 10 μsec latency Ethernet
   - 4400 MB/s DIMM ("PC4400") vs. 50 ns latency
   - Even if just marketing, customers now trained
   - Since bandwidth sells, more resources thrown at bandwidth, which further tips the balance

6 Reasons Latency Lags Bandwidth (cont'd)

4. Latency helps BW, but not vice versa
   - Spinning disk faster improves both bandwidth and rotational latency
     - » 3600 RPM ⇒ 15000 RPM = 4.2X
     - » Average rotational latency: 8.3 ms ⇒ 2.0 ms
     - » Things being equal, also helps BW by 4.2X
   - Lower DRAM latency ⇒ More access/second (higher bandwidth)
   - Higher linear density helps disk BW (and capacity), but not disk Latency
     - » 9,550 BPI ⇒ 533,000 BPI ⇒ 60X in BW

6 Reasons Latency Lags Bandwidth (cont'd)

5. Bandwidth hurts latency
   - Queues help Bandwidth, hurt Latency (Queuing Theory)
   - Adding chips to widen a memory module increases Bandwidth but higher fan-out on address lines may increase Latency
6. Operating System overhead hurts Latency more than Bandwidth
   - Long messages amortize overhead; overhead bigger part of short messages

Summary of Technology Trends

- For disk, LAN, memory, and microprocessor, bandwidth improves by square of latency improvement
  - In the time that bandwidth doubles, latency improves by no more than 1.2X to 1.4X
- Lag probably even larger in real systems, as bandwidth gains multiplied by replicated components
  - Multiple processors in a cluster or even in a chip
  - Multiple disks in a disk array
  - Multiple memory modules in a large memory
  - Simultaneous communication in switched LAN
- HW and SW developers should innovate assuming Latency Lags Bandwidth
  - If everything improves at the same rate, then nothing really changes
  - When rates vary, require real innovation

Acknowledgements

- These slides heavily contain material developed and copyright by
  - Krste Asanovic (MIT/UCB)
  - David Patterson (UCB)
- And also by:
  - Arvind (MIT)
  - Joel Emer (Intel/MIT)
  - James Hoe (CMU)
  - John Kubiatowicz (UCB)
- MIT material derived from course 6.823
- UCB material derived from course CS252