Warmup Exercise

- Consider a binary tree
- Left & right pointers
- Integer value keys
- Initialized to be fully balanced

Question #1:
- The average lookup time for tree of size 1024 (1K = 2^10) is 50ns
- What about for a tree of size 1,048,576 (1M = 2^20)?

Question #2:
- For each item in a tree, look it up (repeatedly)
- What is the expected distribution of lookup times over all items
  - For a tree with height \( h \)
  - That is, what does the histogram of lookup times look like?

```c
while (node != NULL) {
    if (node->m_data == value) {
        return node;
    } else if (node->m_data < value) {
        node = node->m_right;
    } else {
        node = node->m_left;
    }
}
```

Today’s Agenda

- Course overview and administrivia
- Motivational experiments
- What is computer architecture anyway?
  - ...and the forces that drive it
Pervasive Idea: Abstraction and Layering

- **Abstraction**: only way of dealing with complex systems
  - Divide world into objects, each with an...
  - **Interface**: knobs, behaviors, knobs \(\rightarrow\) behaviors
  - **Implementation**: "black box" (ignorance+apathy)
  - Only specialists deal with implementation, rest of us with interface
  - Example: car, only mechanics know how implementation works
- **Layering**: abstraction discipline makes life even simpler
  - Divide objects in system into layers, layer \(n\) objects...
  - Implemented using interfaces of layer \(n - 1\)
  - Don't need to know interfaces of layer \(n - 2\) (sometimes helps)
- **Inertia**: a dark side of layering
  - Layer interfaces become entrenched over time ("standards")
  - Very difficult to change even if benefit is clear (example: Digital TV)
- **Opacity**: hard to reason about performance across layers

Abstraction, Layering, and Computers

- Computers are complex, built in layers
  - Several **software** layers: assembler, compiler, OS, applications
  - **Instruction set architecture (ISA)**
  - Several **hardware** layers: transistors, gates, CPU/Memory/Io
  - 99% of users don't know hardware layers implementation
  - 90% of users don't know implementation of any layer
  - That's okay, world still works just fine
  - But sometimes it is helpful to understand what's "under the hood"

CIS 240: Abstraction and Layering

- Build computer bottom up by raising level of abstraction
- Solid-state semi-conductor materials \(\rightarrow\) transistors
- Transistors \(\rightarrow\) gates
- Gates \(\rightarrow\) digital logic elements: latches, muxes, adders
  - Key insight: number representation
- Logic elements \(\rightarrow\) datapath + control = processor
  - Key insight: stored program (instructions just another form of data)
  - Another one: few insns can be combined to do anything (software)
- Assembly language \(\rightarrow\) high-level language
- Code \(\rightarrow\) graphical user interface

Beyond CIS 240

- CIS 240: Introduction to Computer Systems
  - Bottom-up overview of the entire hardware/software stack
  - Follow on courses look at individual pieces in more detail
- CIS 380: Operating Systems
  - A closer look at system level software
- CIS 277, 330, 341, 350, 390, 391, 455, 460, 461, 462... A closer look at different important application domains
- **CIS 371: Computer Organization and Design**
  - A closer look at hardware layers
Why Study Hardware?

- It’s required (translation: “it’s good for you”, we think)
- Real world impact
  - Without computer architecture there would be no computers
- Penn legacy
  - First “computer” (ENIAC) was built here
  - “computer” = general-purpose stored-program computer
- Get a hardware job
  - Intel, AMD/ATI, IBM, Sun/Oracle, NVIDIA, ARM, HP, TI, Samsung, Microsoft...
- Be better at a software job
  - Apple, Google, Microsoft, etc.
- Go to grad school

Hardware Aspect of CIS 240 vs. CIS 371

- Hardware aspect of CIS 240
  - Focus on one toy ISA: LC4
  - Focus on functionality: “just get something that works”
  - Instructive, learn to crawl before you can walk
  - Not representative of real machines: 240 hardware is circa 1975
- CIS 371
  - De-focus from any particular ISA
  - Focus on quantitative aspects: performance, cost, power, etc.

CIS 371 Topics

- Review of CIS 240 level hardware
  - Instruction set architecture
  - Single-cycle datapath and control
- New
  - Performance, cost, and technology
  - Fast arithmetic
  - Pipelining and superscalar execution
  - Memory hierarchy and virtual memory
  - Multicore
  - Power & energy

Course Goals

- Three primary goals
  - Understand key hardware concepts
    - Pipelining, parallelism, caching, locality, abstraction, etc.
  - Hands-on design lab
  - A bit of scientific/experimental exposure and/or analysis
    - Not found too many other places in the major
- My role:
  - Trick you into learning something
CIS371 Administrivia

- Instructor
  - Prof. Milo Martin (milom@cis), Levine 606
- “Lecture” TAs
  - Christian DeLozier & Abhishek Udupa
- “Lab” TAs
  - TBD
- Contact e-mail:
  - cis371@cis.upenn.edu (goes to me and lecture TAs)
- Lectures
  - Please do not be disruptive (I’m easily distracted as it is)
- Information on assignments, labs, exams, grading
  - Forthcoming

The CIS371 Lab

- Lab project
  - “Build your own processor” (pipelined 16-bit CPU for LC4)
  - Use Verilog HDL (hardware description language)
    - Programming language compiles to gates/wires not insns
  - Implement and test on FPGA (field-programmable gate array)
    - Instructive: learn by doing
    - Satisfying: ”look, I built my own processor”
- No scheduled lab sessions
  - But you’ll need to use the hardware in the lab for the projects

Lab Logistics

- K-Lab: Moore 204
  - Home of the boards, computers, and later in semester … you
  - Good news/bad news: 24 hour access, keycode for door lock
  - “Lab” TA Office hours, project demos here, too
- Tools
  - Digilent XUP-V2P boards
  - Xilinx ISE
  - Warning: all such tools notorious for being buggy and fragile
- Logistics
  - All projects must run on the boards in the lab
  - Boards and lockers handout … sometime in next few weeks

CIS371 Resources

- Three different web sites
  - Course website: syllabus, schedule, lecture notes, assignments
    - http://www.cis.upenn.edu/~cis371/
  - “Piazza”: announcements, questions & discussion
  - The way to ask questions/clarifications
  - Can post to just me & TAs or anonymous to class
  - As a general rule, no need to email me directly
    - Please sign up!
  - “Blackboard”: grade book, turning in some assignments
    - https://courseweb.library.upenn.edu/
- Textbook
  - New this year: available online from Penn library!
  - Course will largely be lecture note driven
Coursework (1 of 2)

- A few homework assignments – **individual work**
  - Written questions, occasional short programming
  - Due at beginning of class
  - 2 total "grace" periods, hand in late, no questions asked
    - One period is to next class (Tue -> Thr, Thr -> Tue)
    - Max of one late period per assignment
    - **Why?** solutions posted after next class

- 4 labs – **all done in groups of 3**
  - Lab 0: getting started, tools intro
  - Lab 1: arithmetic unit & register file
  - Lab 2: single-cycle LC4
  - Lab 3: pipelined LC4: bypassing, branch prediction, superscalar

Coursework (2 of 2)

- **Exams**
  - In-class midterm (TBD)
  - Cumulative final exam (time & date set by registrar)

- Attend two research seminars
  - Of four or five at 3pm on Tue/Thr throughout semester
  - Or watch the recorded video online
  - Turn in short writeup

- Class participation

Grading

- **Tentative grade contributions:**
  - Homework assignments: 15%
  - Labs: 30%
  - Research seminars: 2% x 2 = 4%
  - Class participation: 1%
  - Exams: 50%
    - Midterm: 17%
    - Final: 33%

- **Historical grade distribution**
  - Median grade: B+
  - 2011: A’s: 40%, B’s: 50%, C’s: 7%, D/F’s: 3%
  - 2009: A’s: 40%, B’s: 40%, C’s: 15%, D/F’s: 5%

Academic Misconduct

- Cheating will **not** be tolerated

- **General rule:**
  - Anything with your name on it must be **YOUR OWN** work
  - Example: individual work on homework assignments

- **Possible penalties**
  - Zero on assignment (minimum)
  - Fail course
  - Note on permanent record
  - Suspension
  - Expulsion

- **Penn’s Code of Conduct**
  - [http://www.vpul.upenn.edu/osl/acadint.html](http://www.vpul.upenn.edu/osl/acadint.html)
Full Disclosure

- Potential sources of bias or conflict of interest

- Most of my funding governmental (your tax $$$ at work)
  - National Science Foundation (NSF)
  - DARPA & ONR

- My non-governmental sources of research funding
  - NVIDIA (sub-contract of large DARPA project)
  - Intel
  - Sun/Oracle (hardware donation)

- Collaborators and colleagues
  - Intel, IBM, AMD, Oracle, Microsoft, Google, VMWare, ARM, etc.
  - (Just about every major computer hardware company)

Recap: CIS 371 in Context

- Prerequisite: CIS 240
  - Absolutely required as prerequisite
  - Focused on “function”
  - Exposure to logic gates and assembly language programming

- The “lecture” component of the course:
  - Mostly focuses on “performance”
  - Some coverage of “experimental evaluation”

- The “lab” component of the course:
  - Focuses on “design”
  - Design a working processor

Computer Science as an Estuary

Where does CIS71 fit into computer science?

Engineering, some science

- Engineering
- Design
- Handling complexity
- Real-world impact
- Examples: Internet, microprocessor

Science

- Experiments
- Hypothesis
- Examples: Internet behavior, Protein-folding supercomputer
- Human/computer interaction

Mathematics

- Limits of computation
- Algorithms & analysis
- Cryptography
- Logic
- Proofs of correctness

Other Issues

- Public policy, ethics, law, security

Experimental Motivation
Limits of Abstraction: Question #1

• Question#1:
  • The average lookup time for tree of size 1024 (1K) is 50ns
  • What is the expected lookup time for a tree of size 1048576 (1M)?

• Analysis (from what you know from 121, 240, 320):
  • 1024 is $2^{10}$, 1048576 is $2^{20}$
  • Binary search is $O(\log n)$
  • Based on that, it will take roughly twice as long to lookup in a $2^{20}$ tree than a $2^{10}$ tree
  • Expected time: 100ns

• Let’s evaluate this **experimentally**
  • Experiment: create a balanced tree of size $n$, lookup a random node 100 million times, find the average lookup time, repeat
Average Instructions per Lookup

![Graph showing instructions per lookup](image)

So number of instructions isn’t the problem

Question #1 Discussion

- Analytical answer assuming $O(\log n)$
  - $2^{10}$ to $2^{20}$ will have 2x slowdown
- Experimental result
  - $2^{10}$ to $2^{20}$ has a 10x slowdown
- 5x gap in expected from experimental!
- What is going on?
  - Modern processor have “fast” and “slow” memories
  - Fast memory is called a “cache”
  - As tree gets bigger, it doesn’t fit in fast memory anymore
  - Result: average memory access latency becomes slower

Limits of Abstraction: Question #2

- Question #2:
  - What is the expected distribution of lookup times?
  - That is, for a tree with height $h$, what is the histogram of repeatedly looking up a random value in the tree?

- Analysis:
  - 50% of nodes are at level $n$ (leaves), slowest
  - 25% of nodes are at level $n-1$, a bit faster
  - 12.5% of nodes are at level $n-2$, a bit faster yet
  - 6.25%, 3%, 1.5%

- Let’s evaluate this experimentally
  - Experiment: create a balanced tree of size $2^{19}$, for each node, lookup it up 100 million times (consecutively), calculate lookup time for each node, create a histogram

What about runtime? (not instructions)
What is going on here?

Min leaf: 25

One at: 62 (max)

Several at 56

Tree size is 2

Fastest and Slowest Leaf Nodes (Core2)

- Expectation:
  - Let’s just consider the leaves
  - Same depth, similar instruction count -> similar runtime

- Some of the fastest leaves (all ~24):
  - LLLLLLLLLLLLLLLLLL
  - LLLLLLLLLLLLLLLLLLLLLL (or any with one "R")
  - LLRRLLRRLLRRLLRRLL
  - LLRRLLRRLLRRLLRRLRR
  - LLRRRRRRRRRRRRRRRR
  - was worst than average (~41)

- Some of the slowest leaves:
  - RRRRRRRRRRRRRRRRR
  - RRRRLRRRRRLLLRRLL (~62)
  - RRRRLRRRRRLLLRRLL (~56)
  - Some of the slowest leaves:
  - RRRRLRRRRRLLLRRLLL (~56)
Question #2 Discussion

• Analytical expectation
  - 50%, 25%, 12.5%, 6.25%, 3%, 1.5%...
  - All leaf nodes with similar runtime

• Experimental result
  - Significant variation, position in tree matters
  - All “left” is fastest, all “right” is slow, but not the slowest
  - Pattern of left/right seems to matter significantly

• What is going on?
  - “Taken” branches are slower than “non-taken” branches
  - Modern processors learn and predict branch directions over time
    • Can detect simple patterns, but not complicated ones
  - Result: exact branching behavior matters

What is Computer Architecture?

“Computer Organization”

• “Digital Systems Organization and Design”
  • Don’t really care about “digital systems” in general
  • “Computer Organization and Design”

• Computer architecture
  • Definition of ISA to facilitate implementation of software layers
  • The hardware/software interface

• Computer micro-architecture
  • Design processor, memory, I/O to implement ISA
  • Efficiently implementing the interface

• CIS 371 is mostly about processor micro-architecture
  • Confusing: architecture also means micro-architecture
What is Computer Architecture?

“Computer Architecture is the science and art of selecting and interconnecting hardware components to create computers that meet functional, performance and cost goals.” - WWW Computer Architecture Page

• An analogy to architecture of buildings…

What is Computer Architecture?

The role of a building architect:

Materials
- Steel
- Concrete
- Brick
- Wood
- Glass

Goals
- Function
- Cost
- Safety
- Ease of Construction
- Energy Efficiency
- Fast Build Time
- Aesthetics

Plans

Construction

Buildings
- Houses
- Offices
- Apartments
- Stadiums
- Museums

The role of a computer architect:

“Technology”
- Logic Gates
- SRAM
- DRAM
- Circuit Techniques
- Packaging
- Magnetic Storage
- Flash Memory

Goals
- Function
- Performance
- Reliability
- Cost/Manufacturability
- Energy Efficiency
- Time to Market

Plans

Manufacturing

Computers
- Desktops
- Servers
- Mobile Phones
- Supercomputers
- Game Consoles
- Embedded

Important differences: age (~60 years vs thousands), rate of change, automated mass production (magnifies design)

Computer Architecture Is Different…

• Age of discipline
  - 60 years (vs. five thousand years)

• Rate of change
  - All three factors (technology, applications, goals) are changing
  - Quickly

• Automated mass production
  - Design advances magnified over millions of chips

• Boot-strapping effect
  - Better computers help design next generation
Design Constraints

- **Functional**
  - Needs to be correct
  - And unlike software, difficult to update once deployed
  - What functions should it support (Turing completeness aside)

- **Reliable**
  - Does it continue to perform correctly?
  - Hard fault vs transient fault
  - Google story - memory errors and sun spots
  - Space satellites vs desktop vs server reliability

- **High performance**
  - "Fast" is only meaningful in the context of a set of important tasks
  - Not just "Gigahertz" – truck vs sports car analogy
  - Impossible goal: fastest possible design for all programs

Design Goals

- **Low cost**
  - Per unit manufacturing cost (wafer cost)
  - Cost of making first chip after design (mask cost)
  - Design cost (huge design teams, why? Two reasons...)
  - (Dime/dollar joke)

- **Low power/energy**
  - Energy in (battery life, cost of electricity)
  - Energy out (cooling and related costs)
  - Cyclic problem, very much a problem today

- **Challenge: balancing the relative importance of these goals**
  - And the balance is constantly changing
  - No goal is absolutely important at expense of all others
  - Our focus: performance, only touch on cost, power, reliability

Shaping Force: Applications/Domains

- Another shaping force: applications (usage and context)
  - Applications and application domains have different requirements
    - Domain: group with similar character
    - Lead to different designs

- **Scientific**: weather prediction, genome sequencing
  - First computing application domain: naval ballistics firing tables
  - Need: large memory, heavy-duty floating point
  - Examples: CRAY T3E, IBM BlueGene

- **Commercial**: database/web serving, e-commerce, Google
  - Need: data movement, high memory + I/O bandwidth
  - Examples: Sun Enterprise Server, AMD Opteron, Intel Xeon

More Recent Applications/Domains

- **Desktop**: home office, multimedia, games
  - Need: integer, memory bandwidth, integrated graphics/network?
  - Examples: Intel Core 2, Core i7, AMD Athlon

- **Mobile**: laptops, mobile phones
  - Need: low power, integer performance, integrated wireless
  - Laptops: Intel Core 2 Mobile, Atom, AMD Turion
  - Smaller devices: ARM chips by Samsung and others, Intel Atom

- **Embedded**: microcontrollers in automobiles, door knobs
  - Need: low power, low cost
  - Examples: ARM chips, dedicated digital signal processors (DSPs)
  - Over 1 billion ARM cores sold in 2006 (at least one per phone)

- **Deeply Embedded**: disposable “smart dust” sensors
  - Need: extremely low power, extremely low cost
Application Specific Designs

- This class is about **general-purpose CPUs**
  - Processor that can do anything, run a full OS, etc.
  - E.g., Intel Core i7, AMD Athlon, IBM Power, ARM, Intel Itanium

- In contrast to **application-specific chips**
  - Or **ASICs** (Application specific integrated circuits)
    - Also application-domain specific processors
    - Implement critical domain-specific functionality in hardware
      - Examples: video encoding, 3D graphics
    - General rules
      - Hardware is less flexible than software
      - Hardware more effective (speed, power, cost) than software
      - Domain specific more "parallel" than general purpose
    - But general mainstream processors becoming more parallel

- Trend: from specific to general (for a specific domain)

---

Technology Trends

“Technology”

- Basic element
  - Solid-state **transistor** (i.e., electrical switch)
  - Building block of **integrated circuits (ICs)**

- What’s so great about ICs? Everything
  + High performance, high reliability, low cost, low power
  + Lever of mass production

- Several kinds of integrated circuit families
  - **SRAM/logic**: optimized for speed (used for processors)
  - **DRAM**: optimized for density, cost, power (used for memory)
  - **Flash**: optimized for density, cost (used for storage)
  - Increasing opportunities for integrating multiple technologies

- Non-transistor storage and inter-connection technologies
  - Disk, optical storage, ethernet, fiber optics, wireless

---

Constant Change: Technology

<table>
<thead>
<tr>
<th>“Technology”</th>
<th>Applications/Domains</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic Gates</td>
<td>Desktop</td>
</tr>
<tr>
<td>SRAM, DRAM</td>
<td>Servers</td>
</tr>
<tr>
<td>Circuit Techniques</td>
<td>Mobile Phones</td>
</tr>
<tr>
<td>Packaging</td>
<td>Supercomputers</td>
</tr>
<tr>
<td>Magnetic Storage</td>
<td>Game Consoles</td>
</tr>
<tr>
<td>Flash Memory</td>
<td>Embedded</td>
</tr>
</tbody>
</table>

Goals
- Performance
- Reliability
- Cost/Manufacturability
- Energy Efficiency
- Time to Market

- Absolute improvement, **different rates of change**
- New application domains enabled by technology advances
Funny or Not Funny?

Technology Trends

- **Moore’s Law**
  - Continued (up until now, at least) transistor miniaturization

- Some technology-based ramifications
  - Absolute improvements in density, speed, power, costs
  - SRAM/logic: density: ~30% (annual), speed: ~20%
  - DRAM: density: ~60%, speed: ~4%
  - Disk: density: ~60%, speed: ~10% (non-transistor)
  - Big improvements in flash memory and network bandwidth, too

- **Changing quickly and with respect to each other!!**
  - Example: density increases faster than speed
  - Trade-offs are constantly changing
  - Re-evaluate/re-design for each technology generation

Technology Change Drives Everything

- Computers get 10x faster, smaller, cheaper every 5-6 years!
  - A 10x quantitative change is qualitative change
  - Plane is 10x faster than car, and fundamentally different travel mode

- New applications become self-sustaining market segments
  - Recent examples: mobile phones, digital cameras, mp3 players, etc.

- Low-level improvements appear as discrete high-level jumps
  - Capabilities cross thresholds, enabling new applications and uses

Moore’s Law - 1965

Today: $2^{30}$ transistors
Revolution I: The Microprocessor

- **Microprocessor revolution**
  - One significant technology threshold was crossed in 1970s
  - Enough transistors (~25K) to put a 16-bit processor on one chip
  - Huge performance advantages: fewer slow chip-crossings
  - Even bigger cost advantages: one “stamped-out” component

- Microprocessors have allowed new market segments
  - Desktops, CD/DVD players, laptops, game consoles, set-top boxes, mobile phones, digital camera, mp3 players, GPS, automotive

- And replaced incumbents in existing segments
  - Microprocessor-based system replaced supercomputers, “mainframes”, “minicomputers”, etc.

First Microprocessor

- **Intel 4004 (1971)**
  - Application: calculators
  - Technology: 10000 nm
  - 2300 transistors
  - 13 mm²
  - 108 KHz
  - 12 Volts
  - 4-bit data
  - Single-cycle datapath

Pinnacle of Single-Core Microprocessors

- **Intel Pentium4 (2003)**
  - Application: desktop/server
  - Technology: 90nm (1/100x)
  - 55M transistors (20,000x)
  - 101 mm² (10x)
  - 3.4 GHz (10,000x)
  - 1.2 Volts (1/10x)
  - 32/64-bit data (16x)
  - 22-stage pipelined datapath
  - 3 instructions per cycle (superscalar)
  - Two levels of on-chip cache
  - data-parallel vector (SIMD) instructions, hyperthreading

Tracing the Microprocessor Revolution

- How were growing transistor counts used?
  - Initially to widen the datapath
    - 4004: 4 bits → Pentium4: 64 bits
  - ... and also to add more powerful instructions
    - To amortize overhead of fetch and decode
    - To simplify programming (which was done by hand then)
Revolution II: Implicit Parallelism

- Then to **extract implicit instruction-level parallelism**
  - Hardware provides parallel resources, figures out how to use them
  - Software is oblivious

- Initially using pipelining ...
  - Which also enabled increased clock frequency

- ... caches ...
  - Which became necessary as processor clock frequency increased

- ... and integrated floating-point
- Then deeper pipelines and branch speculation
- Then multiple instructions per cycle (superscalar)
- Then dynamic scheduling (out-of-order execution)

- We will talk about these things

Pinnacle of Single-Core Microprocessors

  - Application: desktop/server
  - Technology: 90nm (1/100x)
  - 55M transistors (20,000x)
  - 101 mm² (10x)
  - 3.4 GHz (10,000x)
  - 1.2 Volts (1/10x)
  - 32/64-bit data (16x)
  - 22-stage pipelined datapath
  - 3 instructions per cycle (superscalar)
  - Two levels of on-chip cache
  - data-parallel vector (SIMD) instructions, hyperthreading

Modern Multicore Processor

- Intel Core i7 (2009)
  - Application: desktop/server
  - Technology: 45nm (1/2x)
  - 774M transistors (12x)
  - 296 mm² (3x)
  - 3.2 GHz to 3.6 GHz (~1x)
  - 0.7 to 1.4 Volts (~1x)

- 128-bit data (2x)
- 14-stage pipelined datapath (0.5x)
- 4 instructions per cycle (~1x)
- Three levels of on-chip cache
- data-parallel vector (SIMD) instructions, hyperthreading
  - Four-core multicores (4x)

Revolution III: Explicit Parallelism

- Then to support **explicit data & thread level parallelism**
  - Hardware provides parallel resources, software specifies usage
  - Why? diminishing returns on instruction-level-parallelism

- First using (subword) vector instructions..., Intel’s SSE
  - One instruction does four parallel multiplies

- ... and general support for multi-threaded programs
  - Coherent caches, hardware synchronization primitives

- Then using support for multiple concurrent threads on chip
  - First with single-core multi-threading, now with multi-core

- Graphics processing units (GPUs) are highly parallel
  - Converging with general-purpose processors (CPUs)?
To ponder...

Is this decade’s “multicore revolution” comparable to the original “microprocessor revolution”?

Technology Disruptions

- Classic examples:
  - The transistor
  - Microprocessor
- More recent examples:
  - Multicore processors
  - Flash-based solid-state storage
- Near-term potentially disruptive technologies:
  - Phase-change memory (non-volatile memory)
  - Chip stacking (also called 3D die stacking)
- Disruptive “end-of-scaling”
  - “If something can’t go on forever, it must stop eventually”
  - Can we continue to shrink transistors for ever?
  - Even if more transistors, not getting as energy efficient as fast

Managing This Mess

- Architect must consider all factors
  - Goals/constraints, applications, implementation technology

- Questions
  - How to deal with all of these inputs?
  - How to manage changes?

- Answers
  - Accrued institutional knowledge (stand on each other’s shoulders)
  - Experience, rules of thumb
  - Discipline: clearly defined end state, keep your eyes on the ball
  - Abstraction and layering

Recap: Constant Change

"Technology"
- Logic Gates
- SRAM
- DRAM
- Circuit Techniques
- Packaging
- Magnetic Storage
- Flash Memory

Applications/Domains
- Desktop
- Servers
- Mobile Phones
- Supercomputers
- Game Consoles
- Embedded

Goals
- Function
- Performance
- Reliability
- Cost/Manufacturability
- Energy Efficiency
- Time to Market