CS 61C:
Great Ideas in Computer Architecture
Moore's Law, Components

Instructor:
Randy H. Katz
http://inst.eecs.Berkeley.edu/~cs61c/fa13

New-School Machine Structures (It's a bit more complicated!)

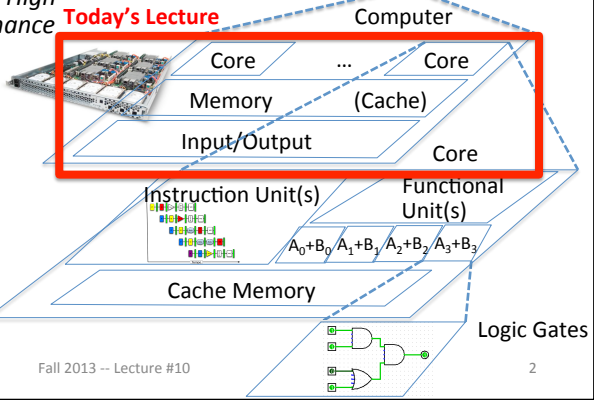
Software

- Parallel Requests
  Assigned to computer
  e.g., Search "Katz"
- Parallel Threads
  Assigned to core
  e.g., Lookup, Ads
- Parallel Instructions
  >1 instruction @ one time
  e.g., 5 pipelined instructions
- Parallel Data
  >1 data item @ one time
  e.g., Add of 4 pairs of words
- Hardware descriptions
  All gates @ one time
- Programming Languages

Harness Parallelism & Achieve High Performance

Hardware

Today's Lecture



Agenda

- Review
- Moore's Law
- Administrivia
- ARM and MIPS
- Technology Break
- Components of a Computer

Agenda

- Review
- Moore's Law
- Administrivia
- ARM and MIPS
- Technology Break
- Components of a Computer

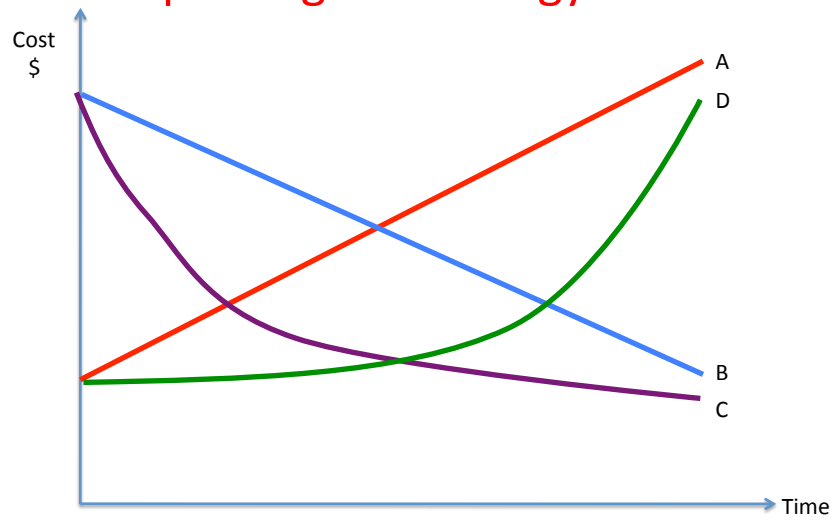
Review

- Assemblers can enhance machine instruction set to help assembly-language programmer
- Translate from text that easy for programmers to understand into code that machine executes efficiently: Compilers, Assemblers
- Linkers allow separate translation of modules
- Interpreters for debugging, but slow execution
- Hybrid (Java): Compiler + Interpreter to try to get best of both
- Compiler Optimization to relieve programmer

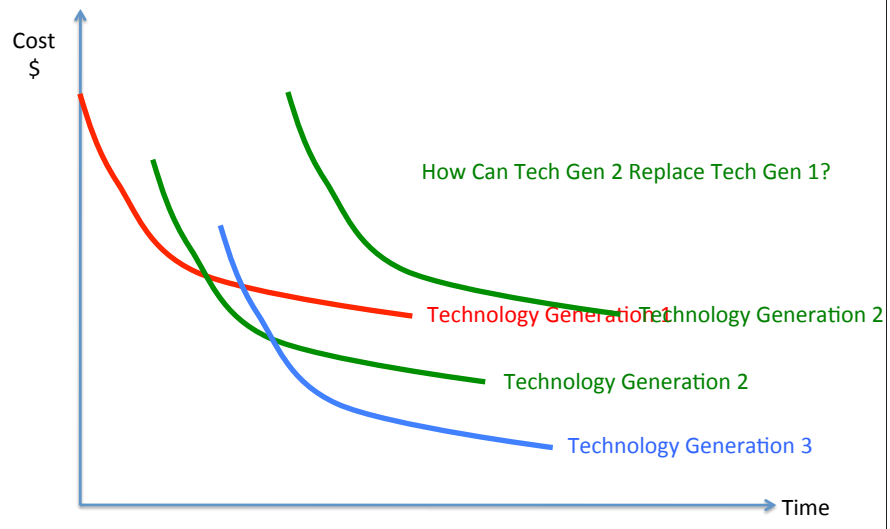
Agenda

- Review
- Moore's Law
- Administrivia
- ARM and MIPS
- Technology Break
- Components of a Computer

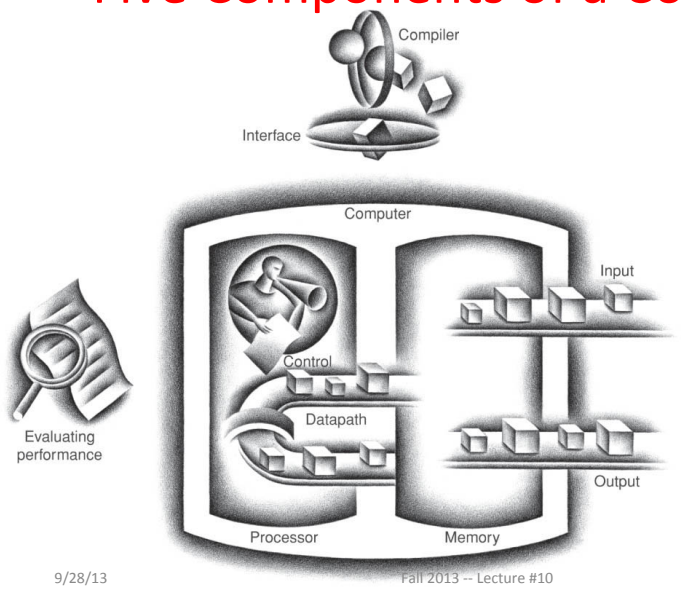
Technology Cost over Time: What does Improving Technology Look Like?



Technology Cost over Time Successive Generations

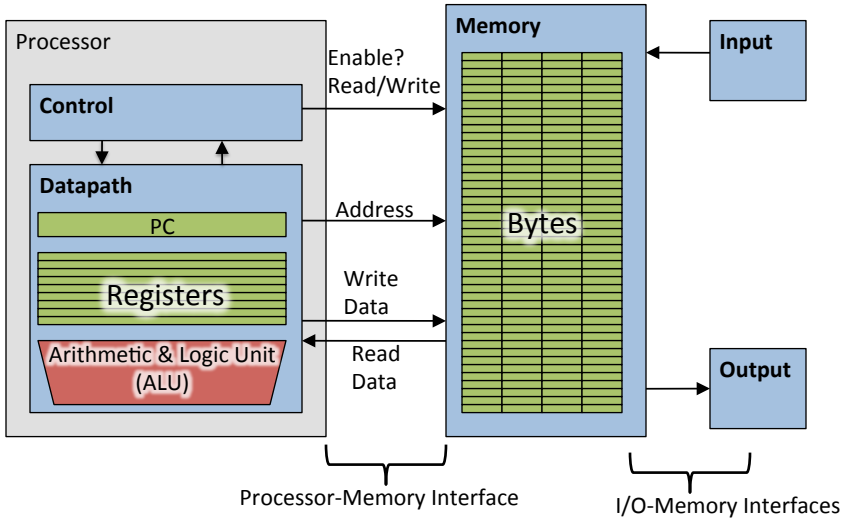


Five Components of a Computer



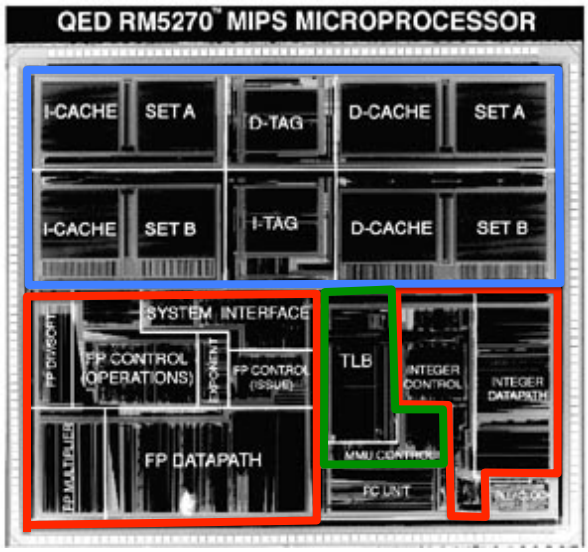
- Control
- Datapath
- Memory
- Input
- Output

Components of a Computer



Reality Check: Typical MIPS Chip Die Photograph

Protection-oriented Virtual Memory Support



Performance Enhancing On-Chip Memory (iCache + dCache)

Floating Pt Control and Datapath

Integer Control and Datapath

Types of Memory

Volatile (needs power to hold state)

- Static RAM (SRAM), built from bistables that use local positive feedback to hold value
- Dynamic RAM (DRAM), holds values on capacitors that must be periodically refreshed

Non-Volatile (holds state without power)

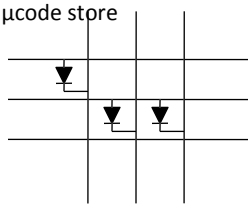
- Read-Only Memory (ROM) – holds fixed contents
- Magnetic memory – Core, plus newer MRAM
- Flash memory – can be written only 10,000's times

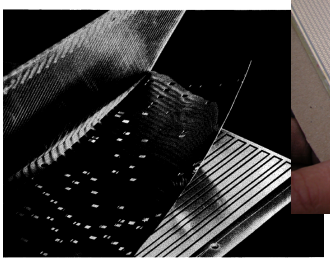
Early Read-Only Memory Technologies

Punched cards, From early 1700s through Jaquard Loom, Babbage, and then IBM

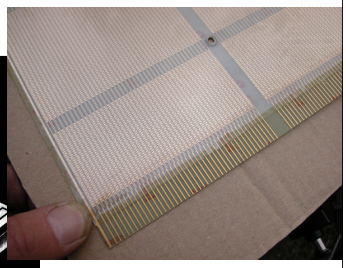
Punched paper tape, instruction stream in Harvard Mk 1

Diode Matrix, EDSAC-2 µcode store





IBM Card Capacitor ROS

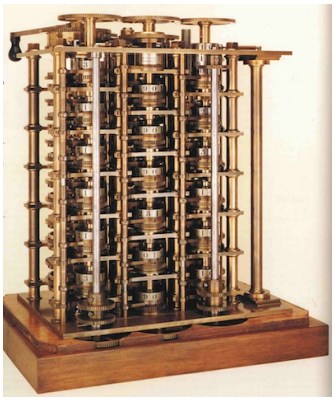


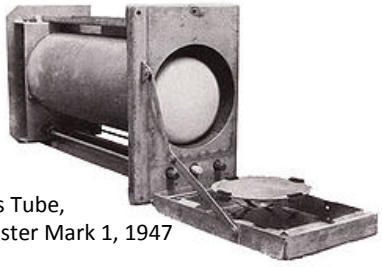
IBM Balanced Capacitor ROS

14

Early Read/Write Memory Technologies

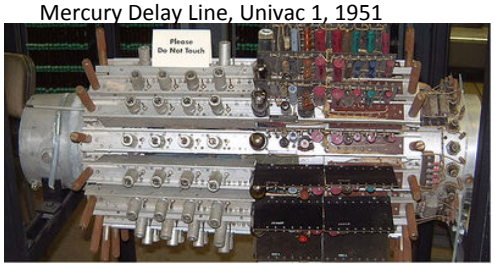
Babbage, 1800s: Digits stored on mechanical wheels





Williams Tube, Manchester Mark 1, 1947

Mercury Delay Line, Univac 1, 1951

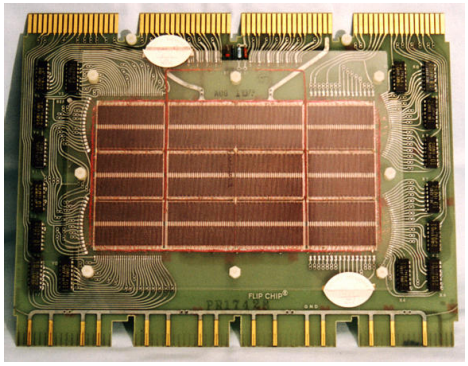


15

Core Memory

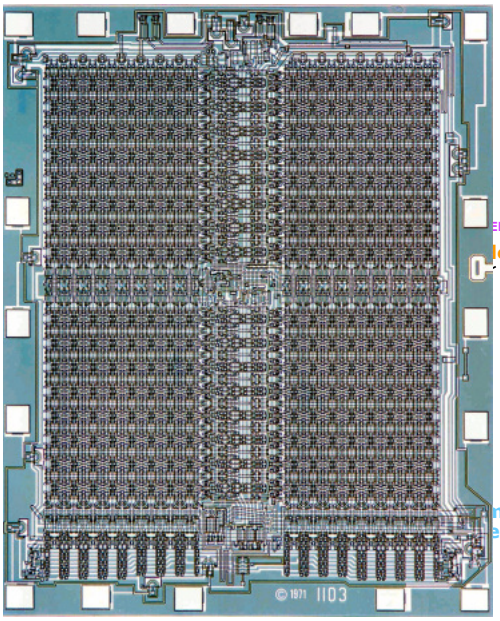
- Core memory was first large scale reliable main memory
  - invented by Forrester in late 40s/early 50s at MIT for Whirlwind project
- Bits stored as magnetization polarity on small ferrite cores threaded onto two-dimensional grid of wires
- Coincident current pulses on X and Y wires would write cell and also sense original state (destructive reads)

- Robust, non-volatile storage
- Used on space shuttle computers until recently
- Cores threaded onto wires by hand (25 billion a year at peak production)
- Core access time ~ 1μs



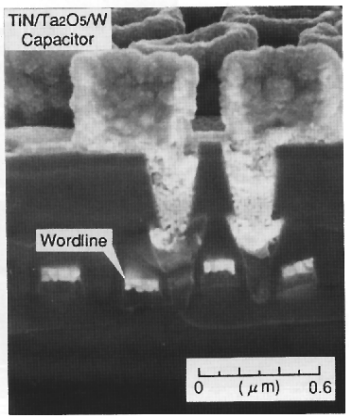
DEC PDP-8/E Board, 4K words x 12 bits, (1968)

One-Transistor Dynamic RAM [Dennard, IBM]

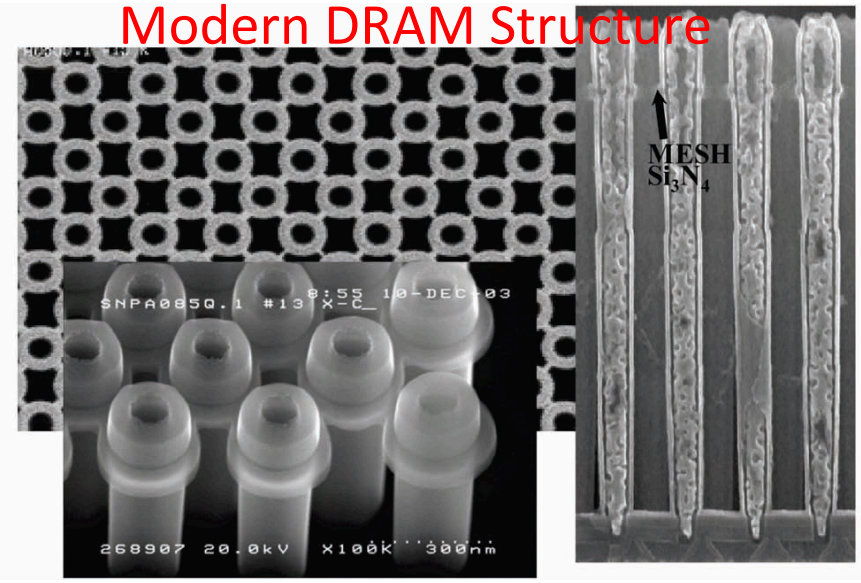


Intel formed to exploit market for semiconductor memory

First commercial DRAM was Intel 1103, held 1Kb in 1970



Modern DRAM Structure

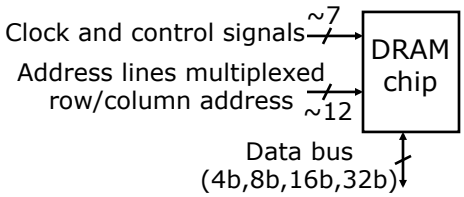


[Samsung, sub-70nm DRAM, 2004]

18

DRAM Packaging

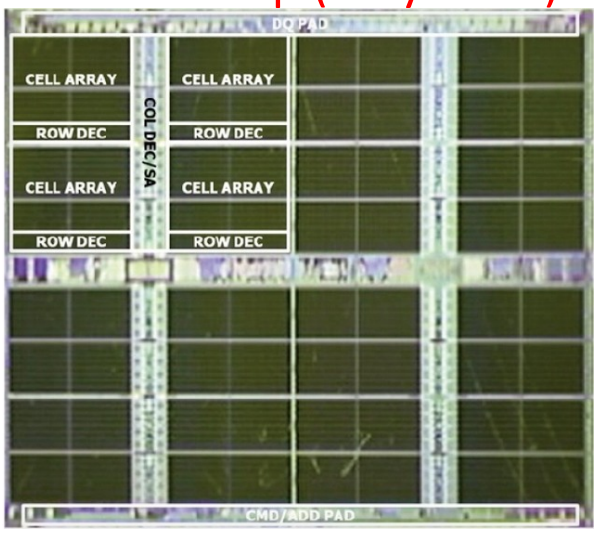
(Laptops/Desktops/Servers)



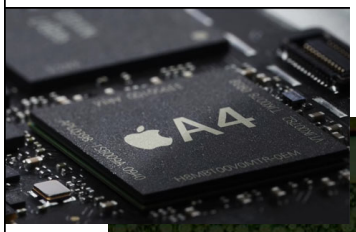
- DIMM (Dual Inline Memory Module) contains multiple chips with clock/control/address signals connected in parallel (sometimes need buffers to drive signals to all chips)
- Data pins work together to return wide word (e.g., 64-bit data bus using 16x4-bit parts)

19

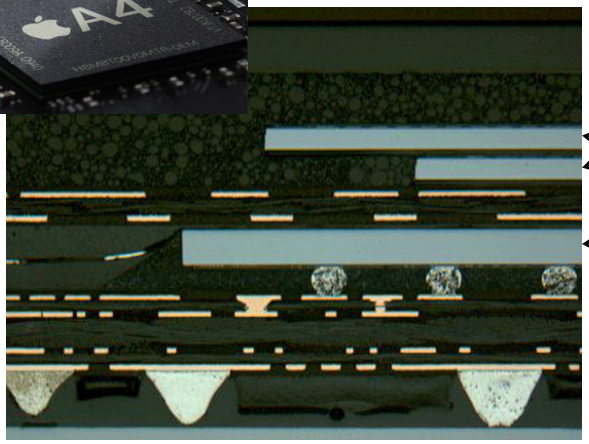
Reality Check: Samsung LPDDR2 4Gb DRAM Chip (May 2013)



DRAM Packaging, Mobile Devices



[Apple A4 package on circuit board]



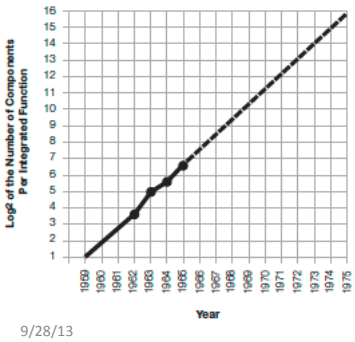
[Apple A4 package cross-section, iFixit 2010]

Moore's Law

"The complexity for minimum component costs has increased at a rate of roughly a factor of two per year. …That means by 1975, the number of components per integrated circuit for minimum cost will be 65,000." *(from 50 in 1965)*

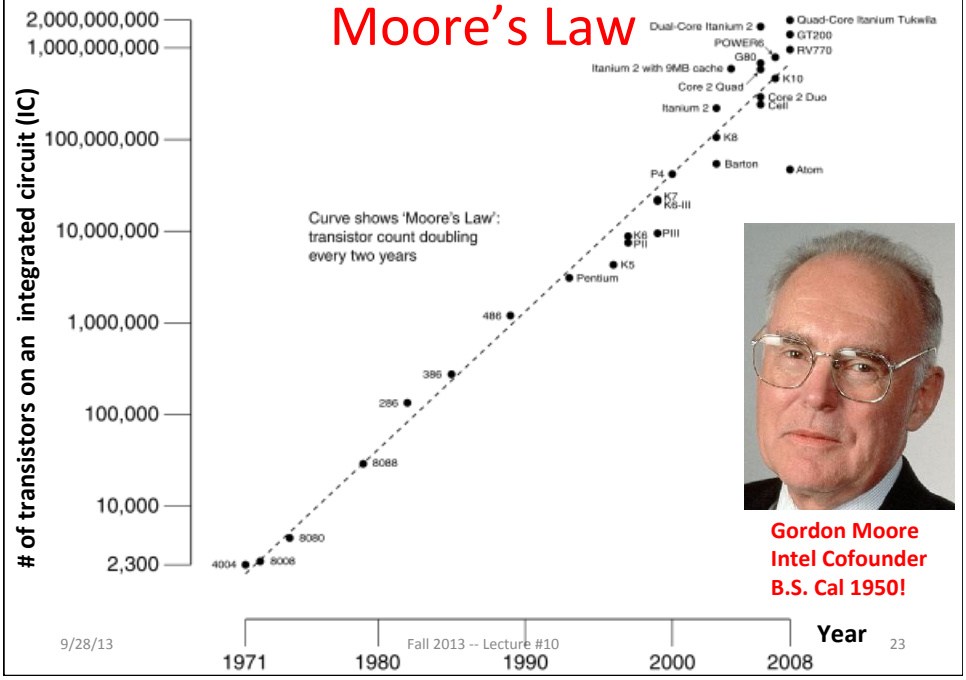
Gordon Moore, "Cramming more components onto integrated circuits," *Electronics*, Volume 38, Number 8, April 19, 1965

"Integrated circuits will lead to such wonders as home computers--or at least terminals connected to a central computer--automatic controls for automobiles, and personal portable communications equipment. The electronic wristwatch needs only a display to be feasible today."



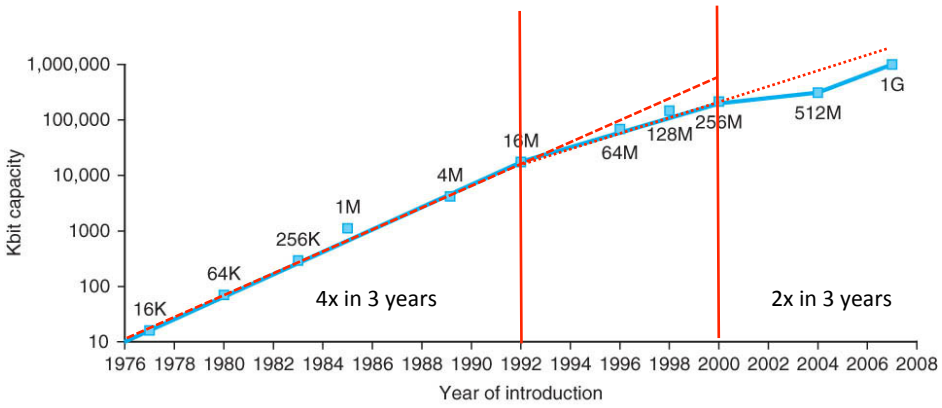
Predicts: 2X Transistors / chip every 2 years

Moore's Law



Gordon Moore
Intel Cofounder
B.S. Cal 1950!

Memory Chip Size

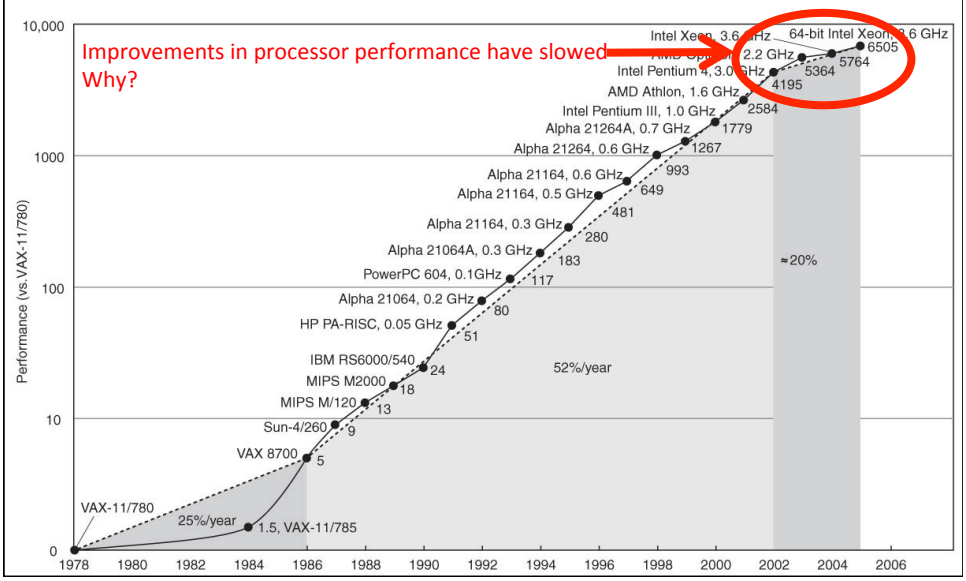


Growth in memory capacity slowing

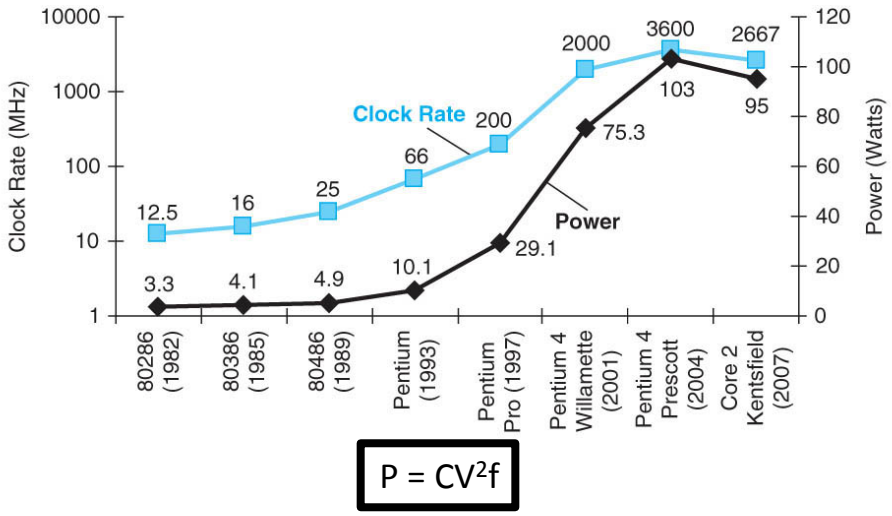
End of Moore's Law?

- It's also a law of investment in equipment as well as increasing volume of integrated circuits that need more transistors per chip
- Exponential growth cannot last forever
- More transistors/chip will end during your careers
  - 2022?
  - (When) will something replace it?

Technology Trends: Uniprocessor Performance (SPECint)



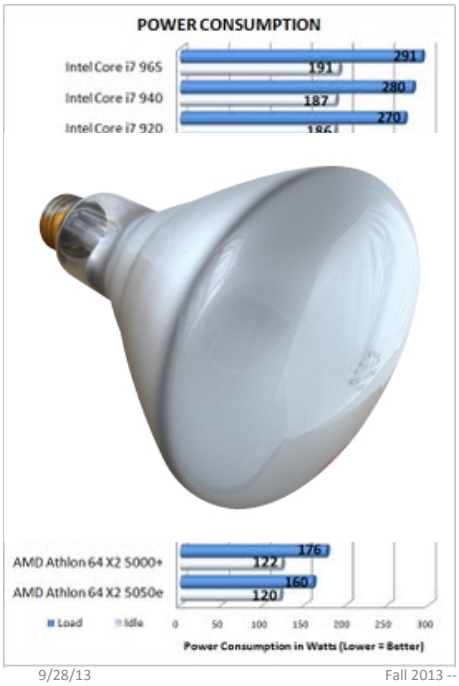
Limits to Performance: Faster Means More Power



$P = CV^2f$

$P = C V^2 f$

- Power is proportional to Capacitance * Voltage^2 * Frequency of switching
- What is the effect on power consumption of:
  - "Simpler" implementation (fewer transistors)?
  - Smaller implementation (shrunk down design)?
  - Reduced voltage?
  - Increased clock frequency?



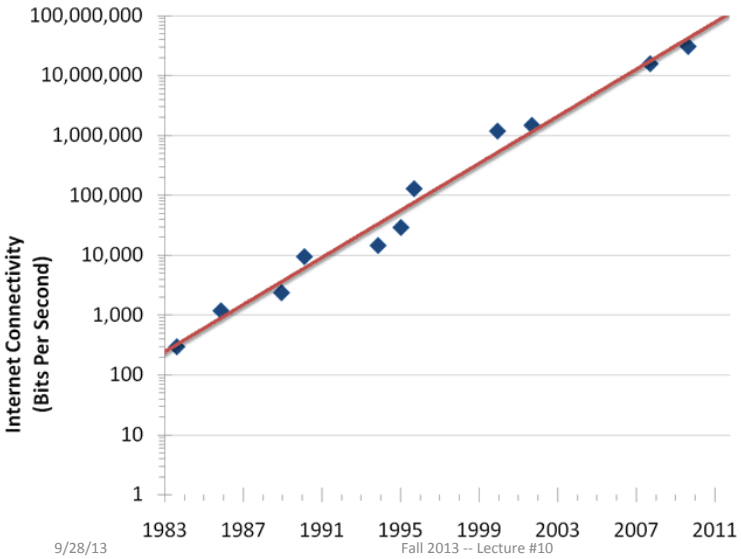
Doing Nothing Well—NOT!

- Traditional processors consume about two thirds as much power at idle (doing nothing) as they do at peak
- Higher performance (server class) processors approaching 300 W at peak
- Implications for battery life?

Computer Technology: Growing, But More Slowly

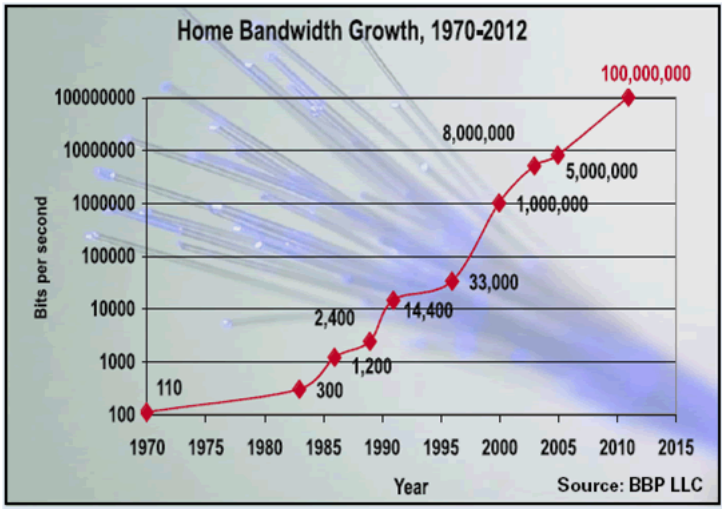
- Processor
  - Speed 2x / 1.5 years (since '85-'05) [***slowing***!]
  - Now +2 cores / 2 years
  - When you graduate: 3-4 GHz, 6-8 Cores in client, 10-14 in server
- Memory (DRAM)
  - Capacity: 2x / 2 years (since '96) [***slowing***!]
  - Now 2X/3-4 years
  - When you graduate: 8-16 GigaBytes
- Disk
  - Capacity: 2x / 1 year (since '97)
  - 250X size last decade
  - When you graduate: 6-12 TeraBytes
- Network
  - Core: 2x every 2 years
  - Access: 100-1000 mbps from home, 1-10 mbps cellular

Internet Connection Bandwidth Over Time

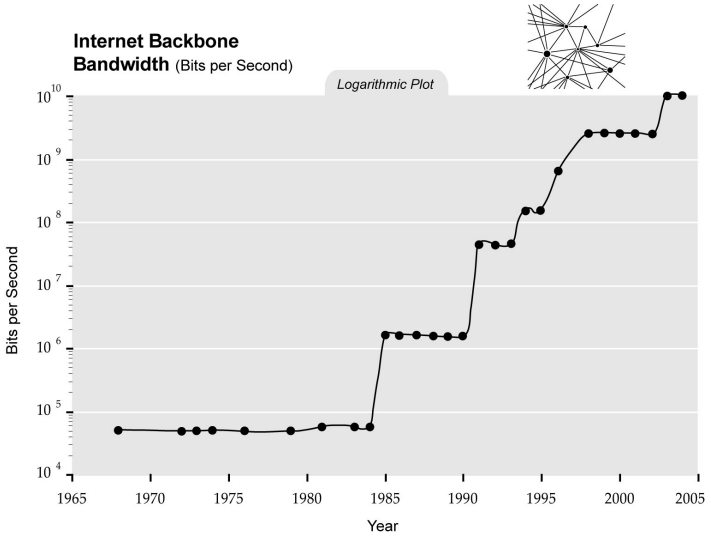


50% annualized growth rate per year

Internet Connection Bandwidth Over Time



Internet Connection Bandwidth Over Time



Agenda

- Review
- Moore's Law
- Administrivia
- ARM and MIPS
- Technology Break
- Components of a Computer

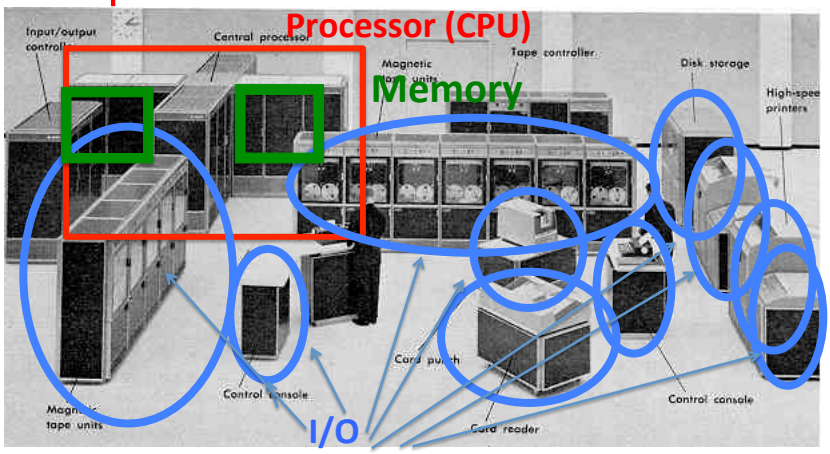
Administrivia

- Lab #5, HW #4, Project #2-1 posted
- Midterm on the horizon:
  - No discussion during exam week
  - TA Review: TBD
  - Exam: Th, 10/17, 6-9 PM, 1 Pimentel, 10 Evans, 155 Dwinelle
  - Small number of special consideration cases, due to class conflicts, etc.—contact me

Agenda

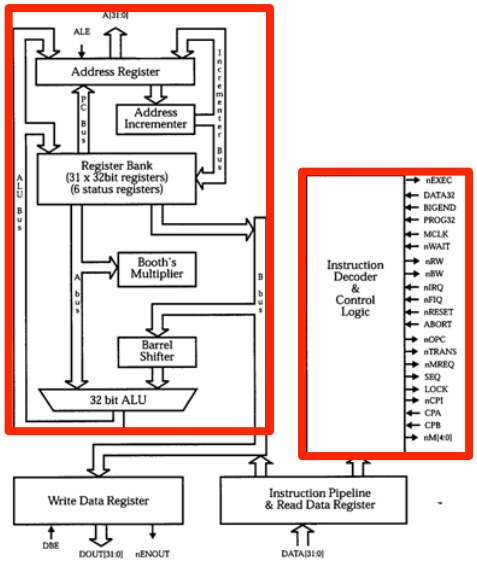
- Review
- Moore's Law
- Administrivia
- Components of a Computer
- Technology Break
- Components of a Computer

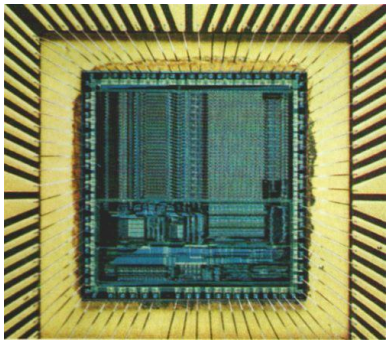
Computer Eras: Mainframe 1950s-60s



"Big Iron": IBM, UNIVAC, … build $1M computers for businesses => COBOL, Fortran, timesharing OS

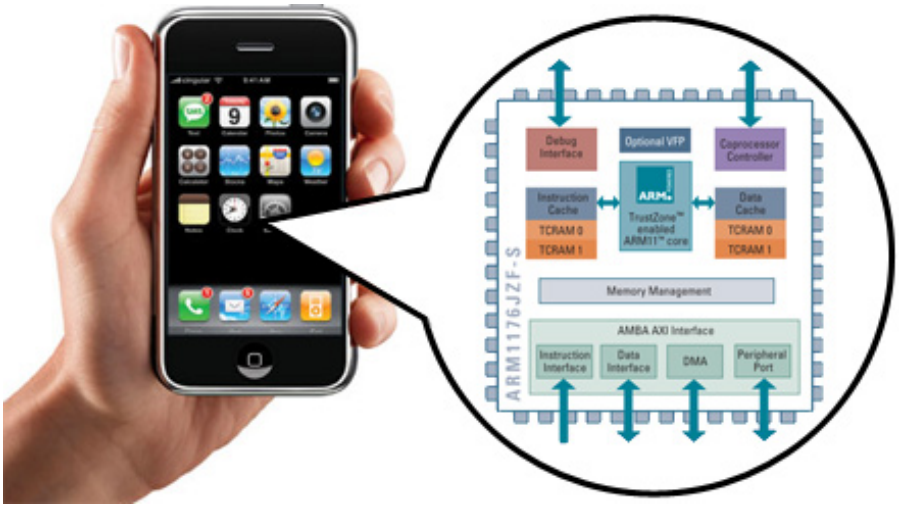
ARM Architecture





- http://en.wikipedia.org/wiki/ARM_architecture

The ARM Inside the iPhone



Flash Card Quiz

How many ARM processors in an iPhone?

One

Two

More than two, less than eight

At least eight

Flash Card Quiz

Which of following statements is true?

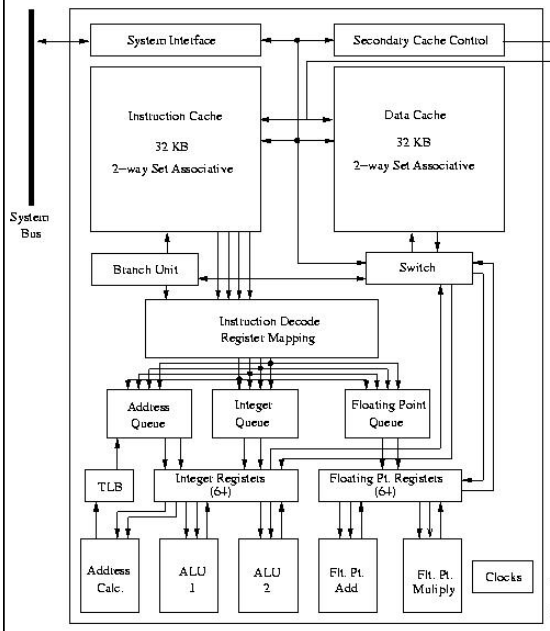
Need a different compiled version of C "strlen" for each ARM core in an iPhone

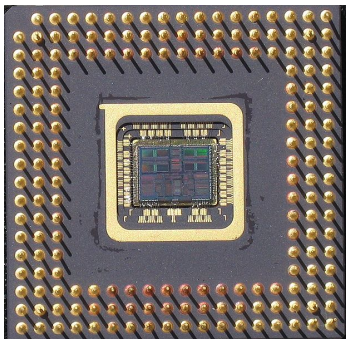
All processors in an iPhone execute the ARM ISA

Different ARM cores are optimized for the I/O device they control

Every ARM core can access all the memory in an iPhone

Example MIPS Block Diagram

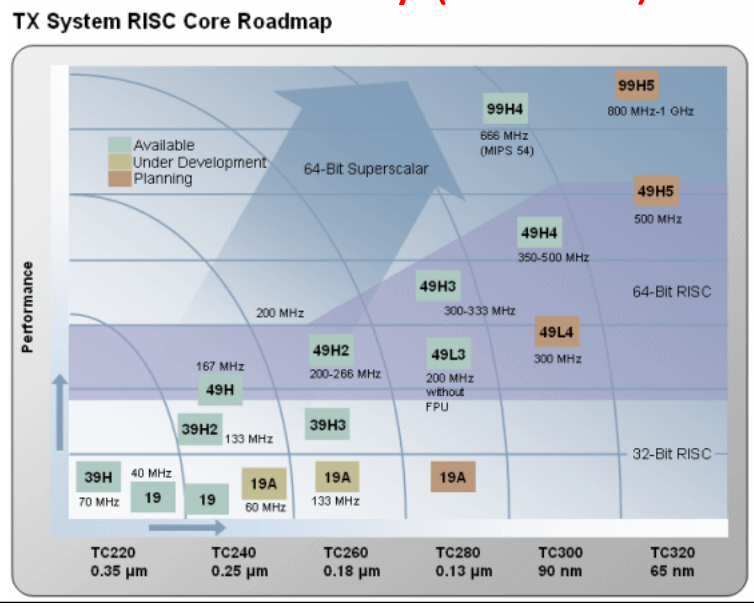




TLB: Translation Look Aside Buffer

ALU: Arithmetic & Logical Unit

A MIPS Family (Toshiba)



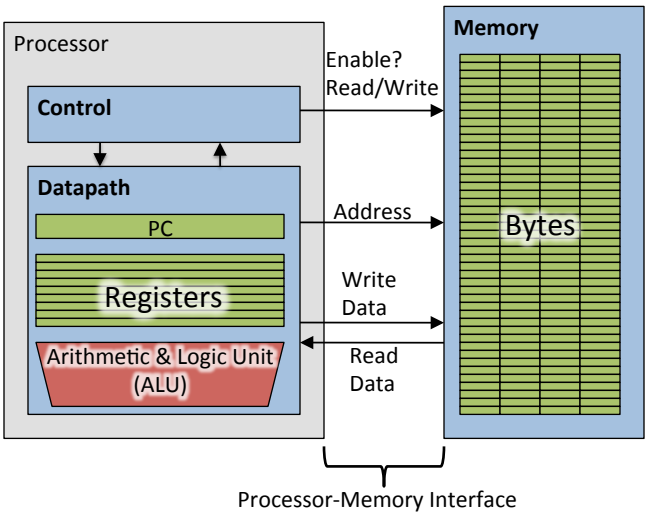
Agenda

- Review
- Moore's Law
- Administrivia
- ARM and MIPS
- Technology Break
- Components of a Computer

Agenda

- Review
- Moore's Law
- Administrivia
- ARM and MIPS
- Technology Break
- Components of a Computer

Components of a Computer



The Processor

- *Processor* (CPU): the active part of the computer, which does all the work (data manipulation and decision-making)
  - *Datapath*: portion of the processor which contains hardware necessary to perform operations required by the processor ("the brawn")
  - *Control*: portion of the processor (also in hardware) which tells the datapath what needs to be done ("the brain")

Phases of Instruction Execution

- Can break up the process of "executing an instruction" into *stages* or *phases*, and then connect the phases to create the whole datapath
  - Smaller phases are easier to reason about and design
  - Easy to optimize (change) one phase without touching the others

Project 2 Warning

- You are going to write a simulator in C for MIPS, implementing these 5 phases of execution

Stages of the Datapath : Overview

- Problem: a single, atomic block which "executes an instruction" (performs all necessary operations beginning with fetching the instruction) would be too bulky and inefficient
- Solution: break up the process of "executing an instruction" into *stages* or *phases*, and then connect the phases to create the whole datapath
  - Smaller phases are easier to design
  - Easy to optimize (change) one phase without touching the others

Phases of the Datapath (1/5)

- There is a wide variety of MIPS instructions: so what general steps do they have in common?
- Phase 1: *Instruction Fetch*
  - No matter what the instruction, the 32-bit instruction word must first be fetched from memory (the cache-memory hierarchy)
  - Also, this is where we Increment PC (that is, PC = PC + 4, to point to the next instruction: byte addressing so + 4)
- Simulator: Instruction = Memory[PC]; PC+=4;

Phases of the Datapath (2/5)

- Phase 2: *Instruction Decode*
  - Upon fetching the instruction, we next gather data from the fields (decode all necessary instruction data)
  - First, read the opcode to determine instruction type and field lengths
  - Second, read in data from all necessary registers
    - For add, read two registers
    - For addi, read one register
    - For jal, no reads necessary

Simulator for Decode Phase

```
Register1 = Register[rsfield];
Register2 = Register[rtfield];
if (opcode == 0) …
   else if (opcode >5 && opcode <10) …
        else if (opcode …) …
            else if (opcode …) …
```

- Better C statement for chained if statements?

Phases of the Datapath (3/5)

- Phase 3: *ALU* (Arithmetic-Logic Unit)
  - Real work of most instructions is done here: arithmetic (+, -, *, /), shifting, logic (&, |), comparisons (slt)
  - What about loads and stores?
    - lw $t0, 40($t1)
    - Address we are accessing in memory = the value in $t1 PLUS the value 40
    - So we do this addition in this stage
- Simulator: Result = Register1 op Register2;
  Address = Register1 + Addressfield

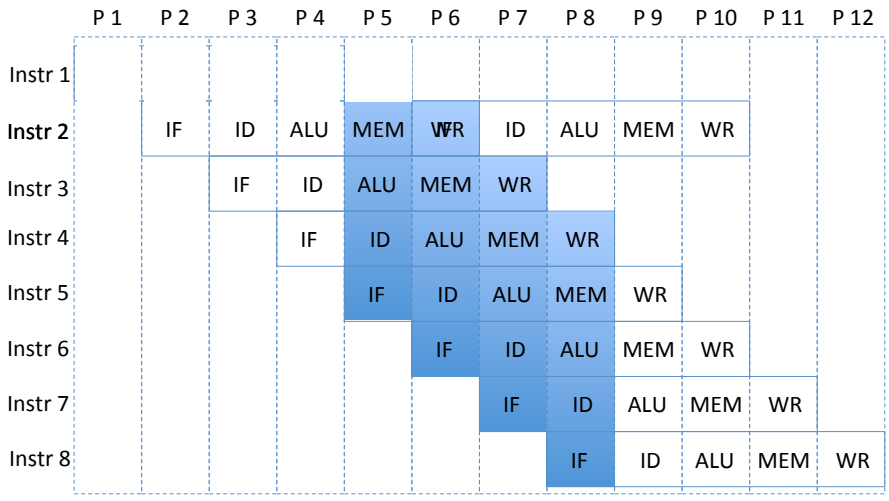
Phases of the Datapath (4/5)

- Phase 4: *Memory Access*
  - Actually only the load and store instructions do anything during this phase; the others remain idle during this phase or skip it all together
  - Since these instructions have a unique step, we need this extra phase to account for them
  - (As a result of the cache system, this phase is expected to be fast: talk about next week)
- Simulator: Register[rtfield] = Memory[Address]
  or Memory[Address] = Register[rtfield]

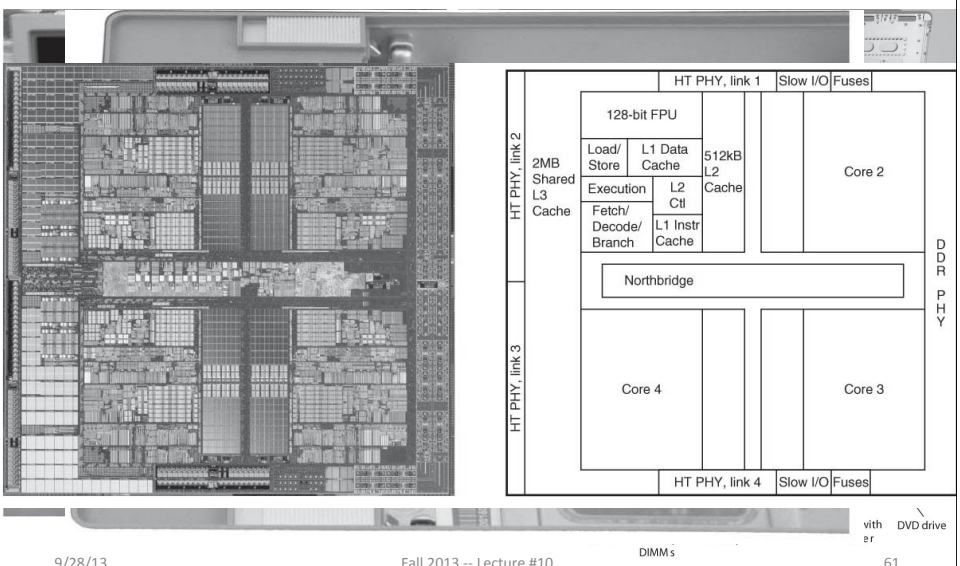
Phases of the Datapath (5/5)

- Phase 5: *Register Write*
  - Most instructions write the result of some computation into a register
  - E.g.,: arithmetic, logical, shifts, loads, slt
  - What about stores, branches, jumps?
    - Don't write anything into a register at the end
    - These remain idle during this fifth phase or skip it all together
- Simulator: Register[rdfield] = Result

Instruction Level Parallelism

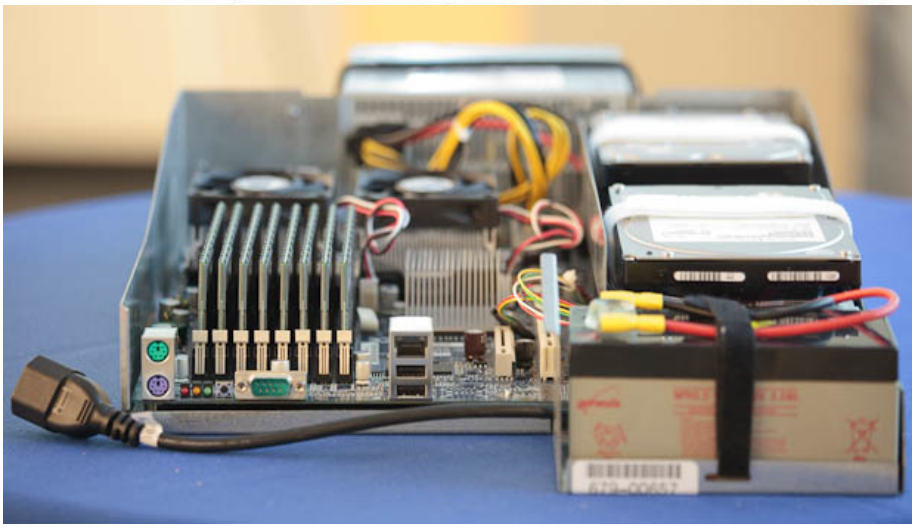


Laptop Innards



Server Internals

Server Internals



And in Conclusion, …

- Key Technology Trends and Limitations
  - Transistor doubling BUT *power* constraints and *latency* considerations limit performance improvement
  - (Single Processor) computers are about as fast as they are likely to get, exploit parallelism to go faster
- Five Components of a Computer
  - Processor/Control + Datapath
  - Memory
  - Input/Output: Human interface/KB + Mouse, Display, Storage … evolving to speech, audio, video
- Architectural Family: One Instruction Set, Many Implementations