PH-chp1a-LEC

COSC-520, Computer and Systems Architecture, 2012, spring

Prof. Richard K. Squier
St. Mary's, Room 339
202-687-6027
squier@cs.georgetown.edu

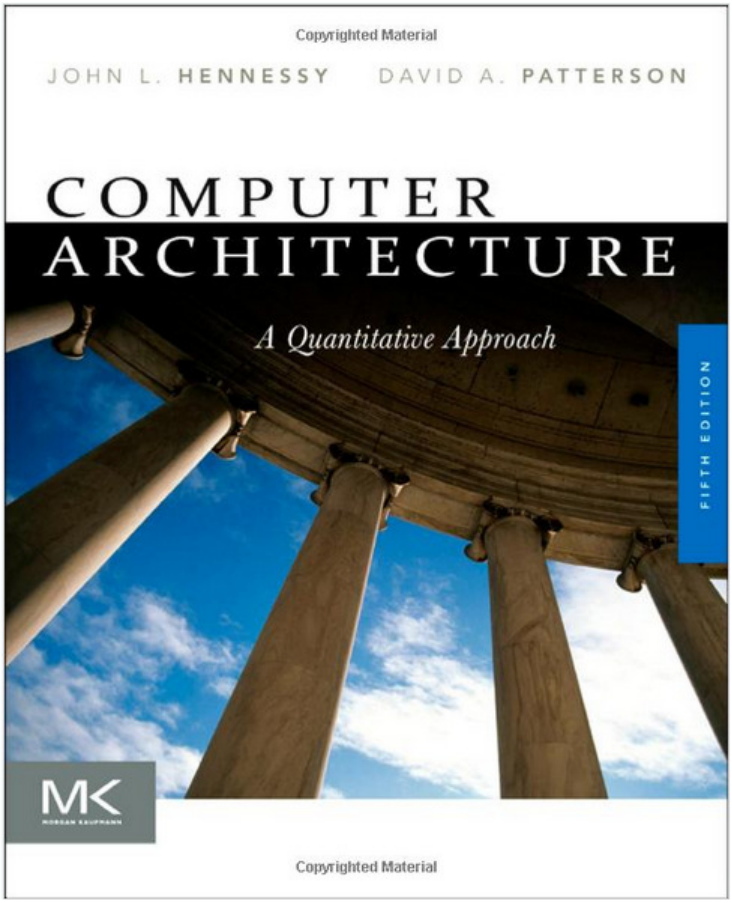
Office hours: Mon/Wed 14:00-15:00, and by appt., or drop in.
Lecture: Tue/Thu 09:30-10:45, White-Gravenor, Room 208

Textbook:

Hennessy & Patterson

Computer Architecture: A Quantitative Approach
5th Edition

Morgan-Kaufmann, 2012
ISBN-13: 978-0-12-374-750-1



Supplemental text:

Patt & Patel

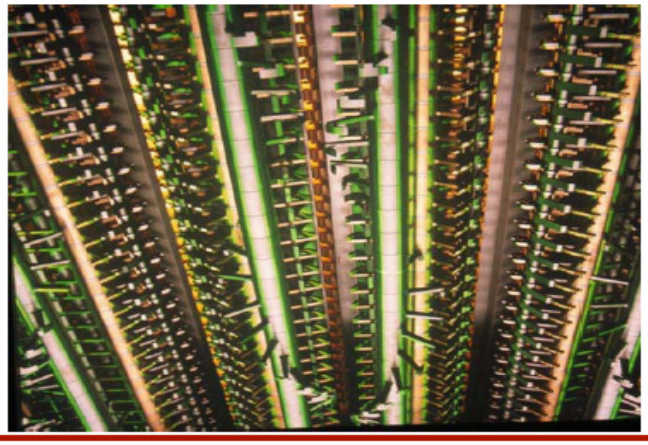
Introduction to Computing Systems
2nd Edition, 2nd Printing

McGraw-Hill, 2004
ISBN-10: 0-07-246750-9

Current State of the World

- Electronic systems dominate almost everything
  - And most of these systems use processors and memory
- Why?
  - Break this question into three questions
    - Why electronics?
    - Why use digital integrated circuits (ICs) to build electronics?
    - Why use processors in ICs?
- Why use electronics
  - Electrons are easy to move / control
  - Easier than the current alternatives
- Result is that we move information / not real physical stuff
  - Think phone, email, fax, TV, WWW, etc.

Mechanical Alternative to Electronics



Picture of a version of the Babbage difference engine built by the Museum of Science UK

"The calculating section of Difference Engine No. 2, has 4,000 moving parts (excluding the printing mechanism) and weighs 2.6 tons. It is seven feet high, eleven feet long and eighteen inches in depth"

Also,

water,

Gasses,

QM,

→ all physical processes?

What is information?

- Building electronics
  - Started with tubes, then miniature tubes
  - Transistors, then miniature transistors
  - Components were getting cheaper, more reliable but
    - There is a minimum cost of a component (storage, handling ...)
    - Total system cost was proportional to complexity

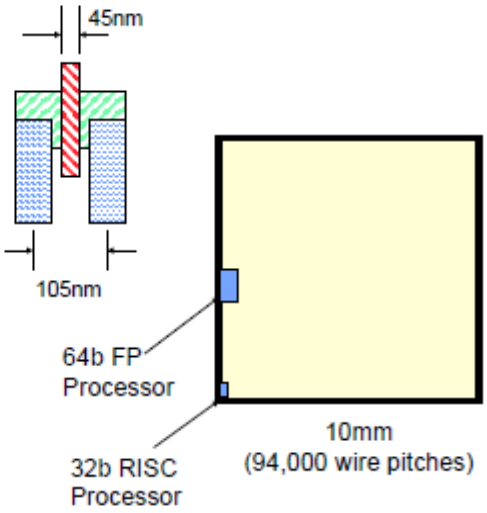
Complexity ↑
cost per unit ↑

- Integrated circuits changed that
  - Devices that integrate multiple transistors
  - Print a circuit, like you print a picture,
    - Create components in parallel
    - Cost no longer depended on # of devices
  - What happens as resolution goes up?

Printing
= investment in design ↑
but
cost per unit ↓ !

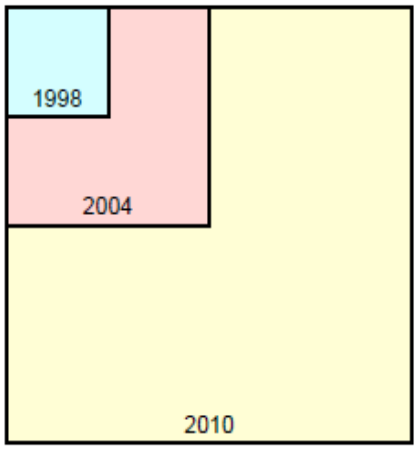
Sense of Scale

- What fits on a chip today?
- Mainstream logic chip
  - 10mm on a side (100mm²)
  - 45nm drawn gate length
  - 105nm wire pitch
  - 10 wires levels
- For comparison
  - 32b RISC integer processor
    - 1K x 2K wire grids
    - 4400 processors
  - SRAM
    - About 4 x 4 grids / bit
    - 552 M SRAM cells
  - DRAM
    - 1 x 2 grids / bit
    - 4.4 B cells



94,000 x 94,000 grid

- Chip density doubles every 3 years
  - What can you do with this?
- More devices ⇒ harder to design

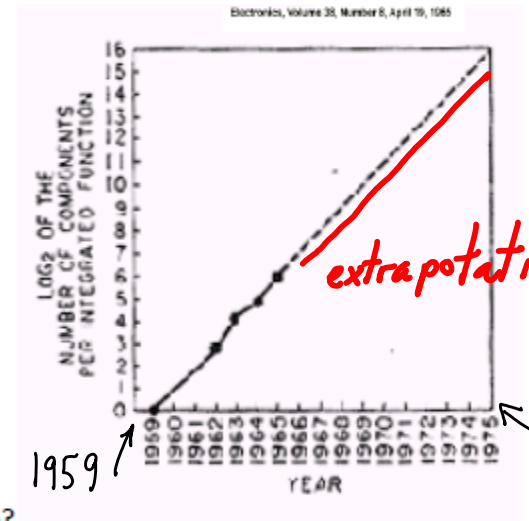


4 x density / 6 yr

→ 4 x complexity

The Famous Moore's Law

- Devices get smaller
  - Get more devices on a chip
  - Devices get faster
- Initial graph from 1965 paper
  - Prediction: 2x density per year
  - Not too many data points
- Slowed down to 2x
  - Every 1.5 to 2 years?
- Is Moore's Law really a Law?
- What does it say about performance?

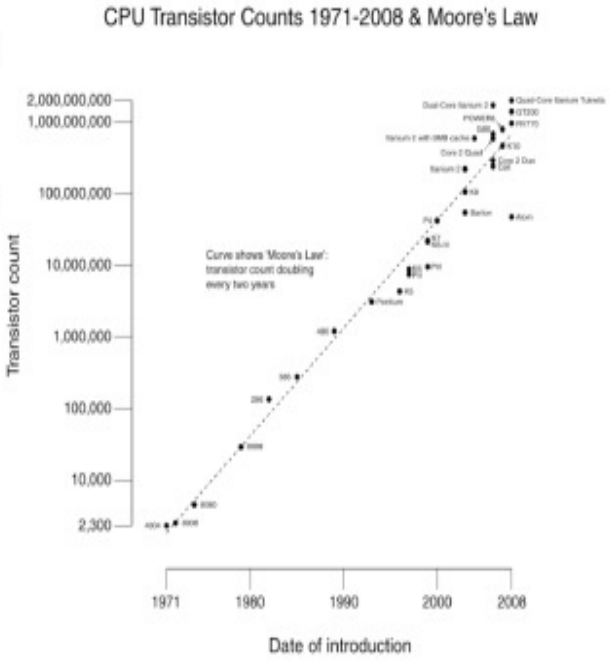


$\frac{2^0 \rightarrow 2^{16} \left(\frac{TR}{cm^2}\right)}{16\ yr}$

~ 200%/yr

Microprocessor Scaling

- Intel 8080 (1975)
  - 3,5k transistors @ 200KHz
- Intel Pentium4 (2003)
  - 42M transistors @ 3.4GHz
- Performance changed from 0.06MIPS to >1,000MIPS
- Complexity scaling has been largely hidden from software



2,300 → 2 B (TR)

~ 35 yr

~ $\frac{\times 10^6}{35\ yr}$

~ 48%/yr

Overall $\frac{1 \rightarrow 2B}{48\ yr}$ → ~ 54%/yr

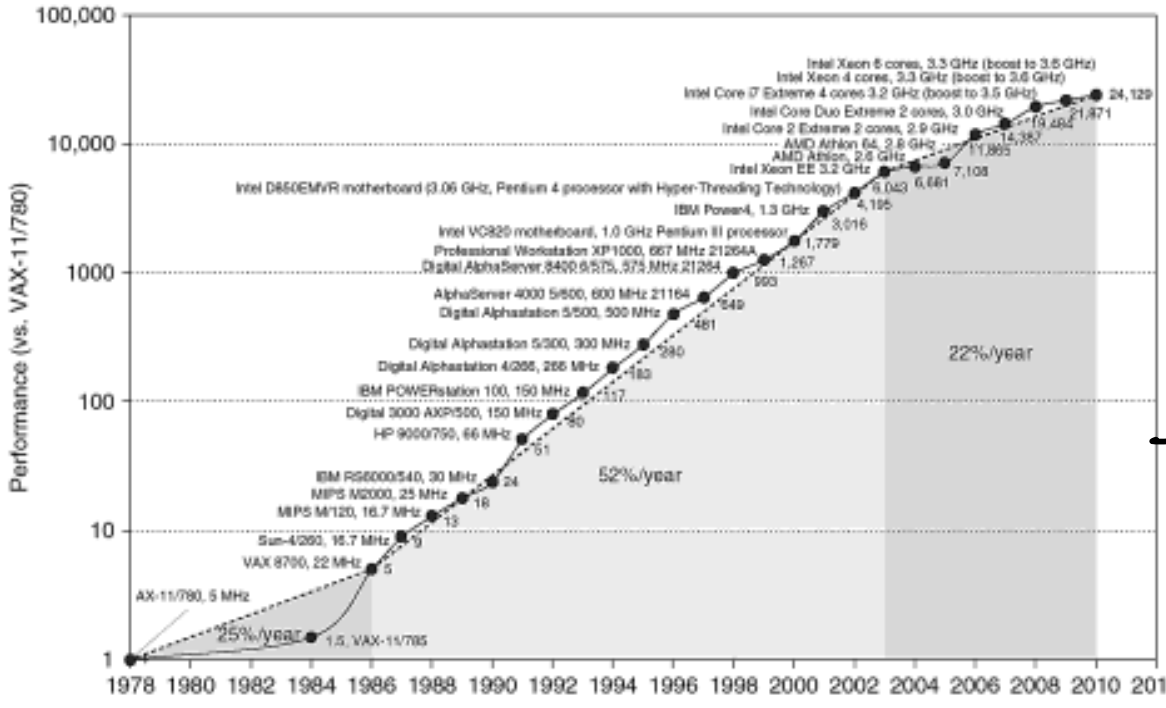


Figure 1.1 Growth in processor performance since the late 1970s. This chart plots performance relative to the VAX 11/780 as measured by the SPEC benchmarks (see Section 1.8). Prior to the mid-1980s, processor performance

Related to Performance?

$\frac{1 \rightarrow 24{,}000}{32\ yr}$

→ 37%/yr

Why?

- clocks ↑
- complexity ↑
- keeping all parts busy ↓

Design space

measures $/op, W/op, J/op

delay/op → -$/op (penalties)

Feature	Personal mobile device (PMD)	Desktop	Server	Clusters/warehouse-scale computer	Embedded
Price of system	$100–$1000	$300–$2500	$5000–$10,000,000	$100,000–$200,000,000	$10–$100,000
Price of microprocessor	$10–$100	$50–$500	$200–$2000	$50–$250	$0.01–$100
	1.8 B	0.35 B	0.02 B		19 B (2010)
Critical system design issues	Cost, energy, media performance, responsiveness	Price-performance, energy, graphics performance	Throughput, availability, scalability, energy	Price-performance, throughput, energy proportionality	Price, energy, application-specific performance

Figure 1.2 A summary of the five mainstream computing classes and their system characteristics. Sales in 2010 included about 1.8 billion PMDs (90% cell phones), 350 million desktop PCs, and 20 million servers. The total number of embedded processors sold was nearly 19 billion. In total, 6.1 billion ARM-technology based chips were shipped in 2010. Note the wide range in system price for servers and embedded systems, which go from USB keys to network routers. For servers, this range arises from the need for very large-scale multiprocessor systems for high-end transaction processing.

The Complexity Problem

- Complexity is the limiting factor in modern chip design
  - Two problems

1. How do you make use of all that IC resources?
   - Uberappliance
     - Cellphone, PDA, iPod, mobile TV, video camera
   - Too many applications to cast all into hardware logic
   - Takes too long to finish the design
2. How do you make sure it works?
   - Verification problem
   - How do you fix bugs?

- Only way to survive complexity:
  - Hide complexity in "general-purpose" components
  - "Reuse" components

Programmable Components aka Processors

- An old approach to solve the complexity problem
  - Build a generic device and customize with memory
    - Through a process called programming ☺
  - (Re)use device in a large number of systems
  - Best way to do this is with a general purpose processor

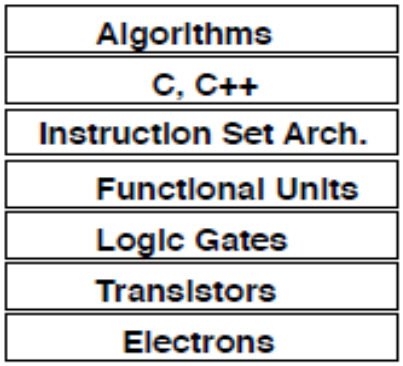
- Processor complexity grows with technology
  - But software model stays roughly the same
    - C, C++, Java, … run on Pentium 2, 3, 4, M, Core, Core 2, …
    - True for sequential programs
  - This is getting much tougher to do
    - Recent hardware developments require software model changes
    - Multi-core processors

Key to Complexity: Nice Interfaces

- Use abstraction to hide complexity
  - Define an interface to allow people to use features without needing to understand all the implementation details

- Works for hardware and software

- Stable interfaces allows people to optimize below and above it



Major Topics

- Hardware-software interface
  - Machine language and assembly language programming
  - Compiler optimizations and performance
- Processor design
  - Pipelined processor design, multiple pipes, out-of-order, speculation
- Memory hierarchy
  - Caches
- Virtual memory & operating systems support
- I/O devices and systems

Key Ideas in Computer Systems

- Pipelining
- Parallelism
- Caching
- Indirection
- Amortization
- Out-of-order execution
- Speculation

- Widely applicable across hardware & software in computer systems
  - Learn them here; use them everywhere…

Reality Check #4
Efficiency Matters

- Most modern systems are constrained by cost and energy
  - You want a cheap cell phone that does a lot with a long battery life
  - You want an scalable and cost-efficient data center

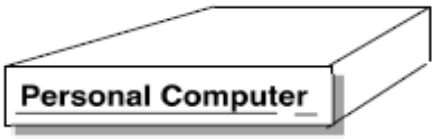
- Hence, efficiency improvements can have a huge impact
  - Efficiency measured in performance/$, performance/Watt, …
  - This is why Google builds its own servers, Apple has a chip design group, …
  - Significant space for improvement in both large and small systems
    - But typically requires a system-level approach

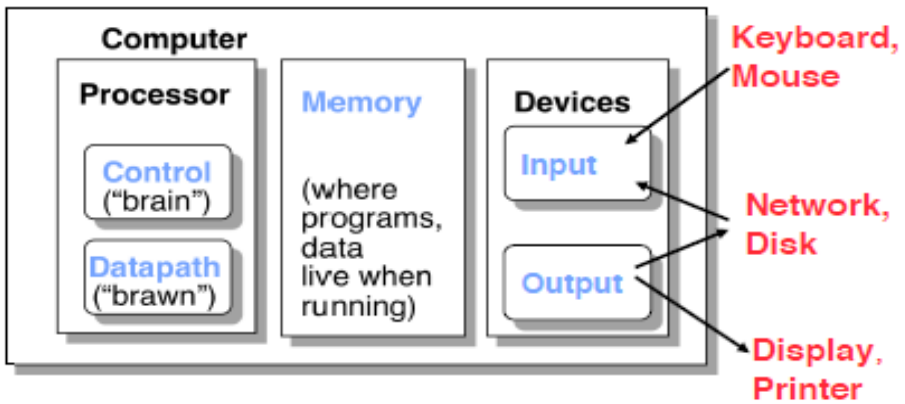
What is a Computer System?

- Depends (a little) on what type of computer system
- We probably mostly think about PC systems

- Actually most computers look like this…

5 components of any Computer





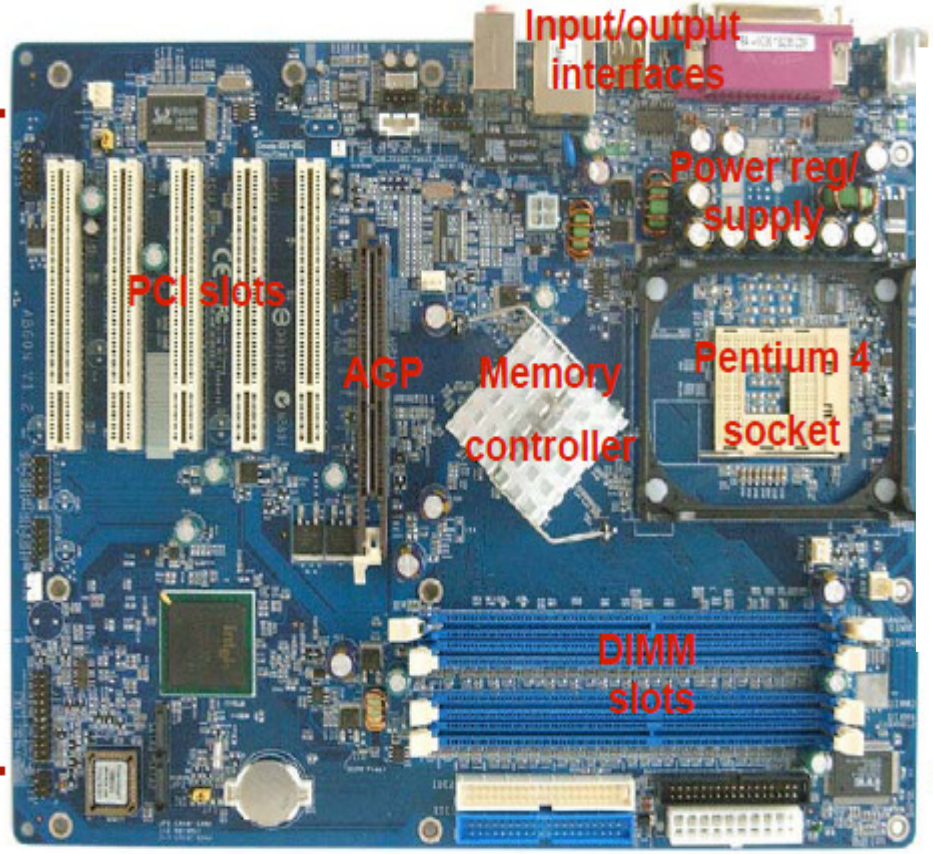
What is in a Computer System?

- Each system is different, but generally have similar parts:
- Must have:
  - Processor, Memory
  - Interface to outside world (I/O)
- Generally have:
  - Cache memory
  - System bus
  - Memory controller
  - I/O bus

MIPS Processor Board

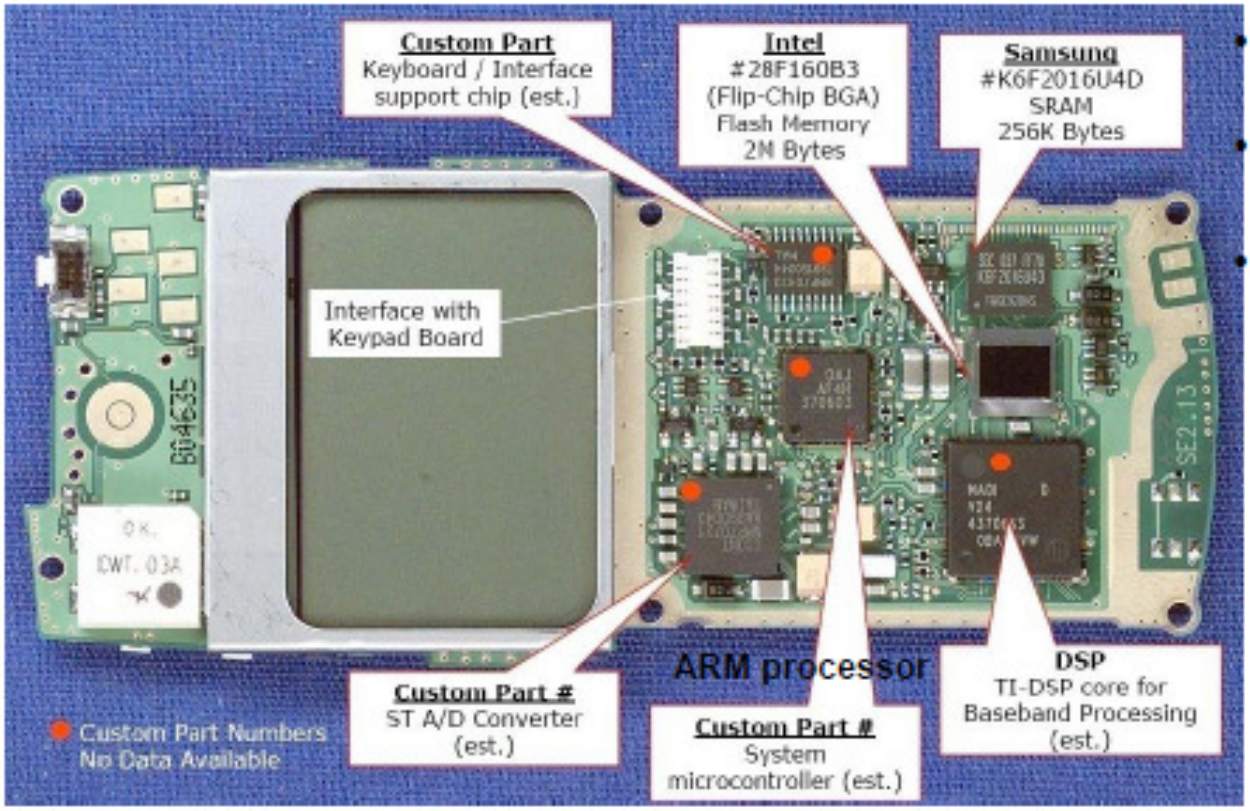
- R3000 CPU (120K transistors)
- R3010 FPU
- 32 KB Instruction cache
- 32 KB Data cache
- 256 KB secondary cache
- Memory controller chips

PC Motherboard



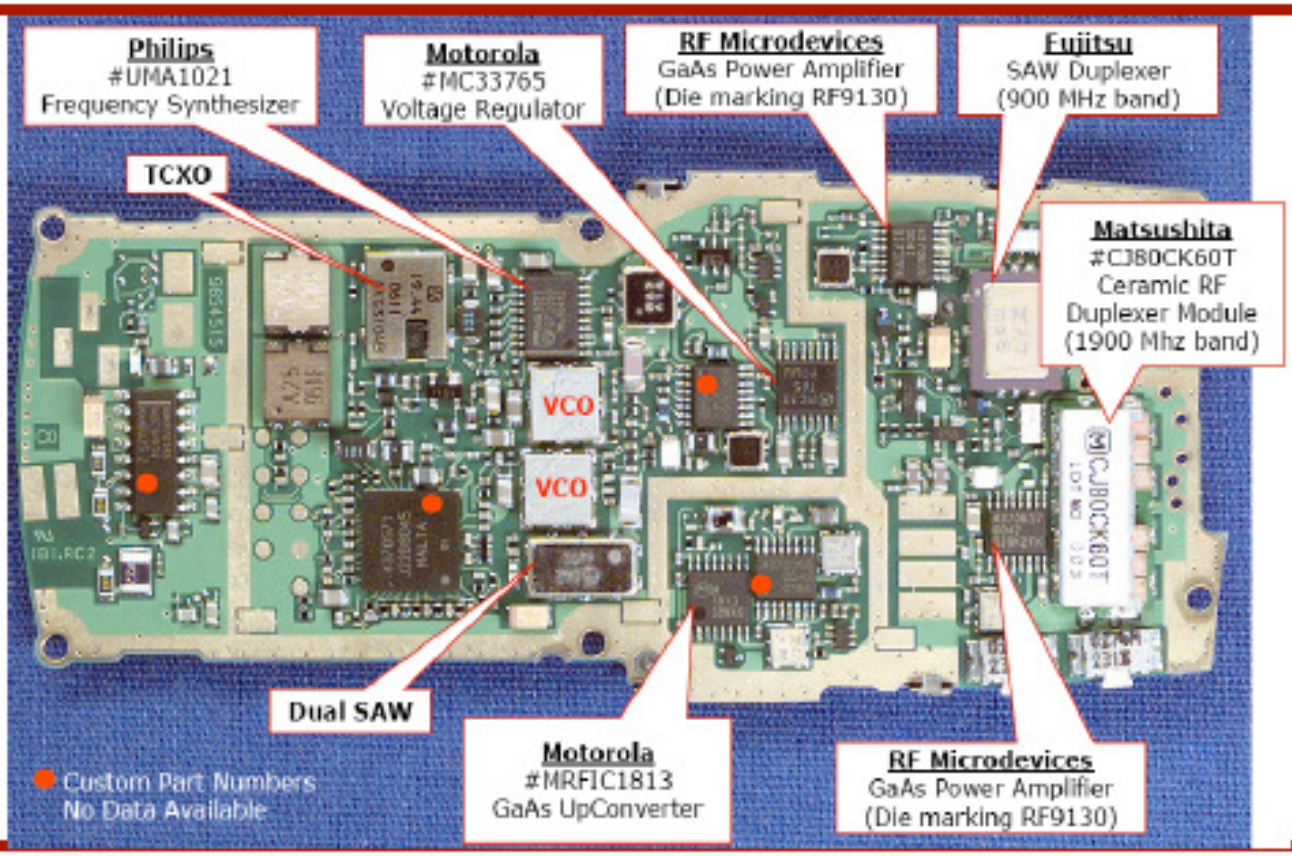
- Pentium 4 2.66 GHz
  - 8KB Data cache, 12 KB Instruction cache
  - 512 KB L2 Cache
  - 533 MHz System Bus
  - 68 Watts
- Memory system
  - 4 DDR DIMM slots
  - Up to 4 GB
- I/O interfaces
  - Ethernet
  - USB
  - Serial ATA (disk)
  - Serial port
  - Parallel port

Digital Cell Phone (Nokia 8260)
Front Side



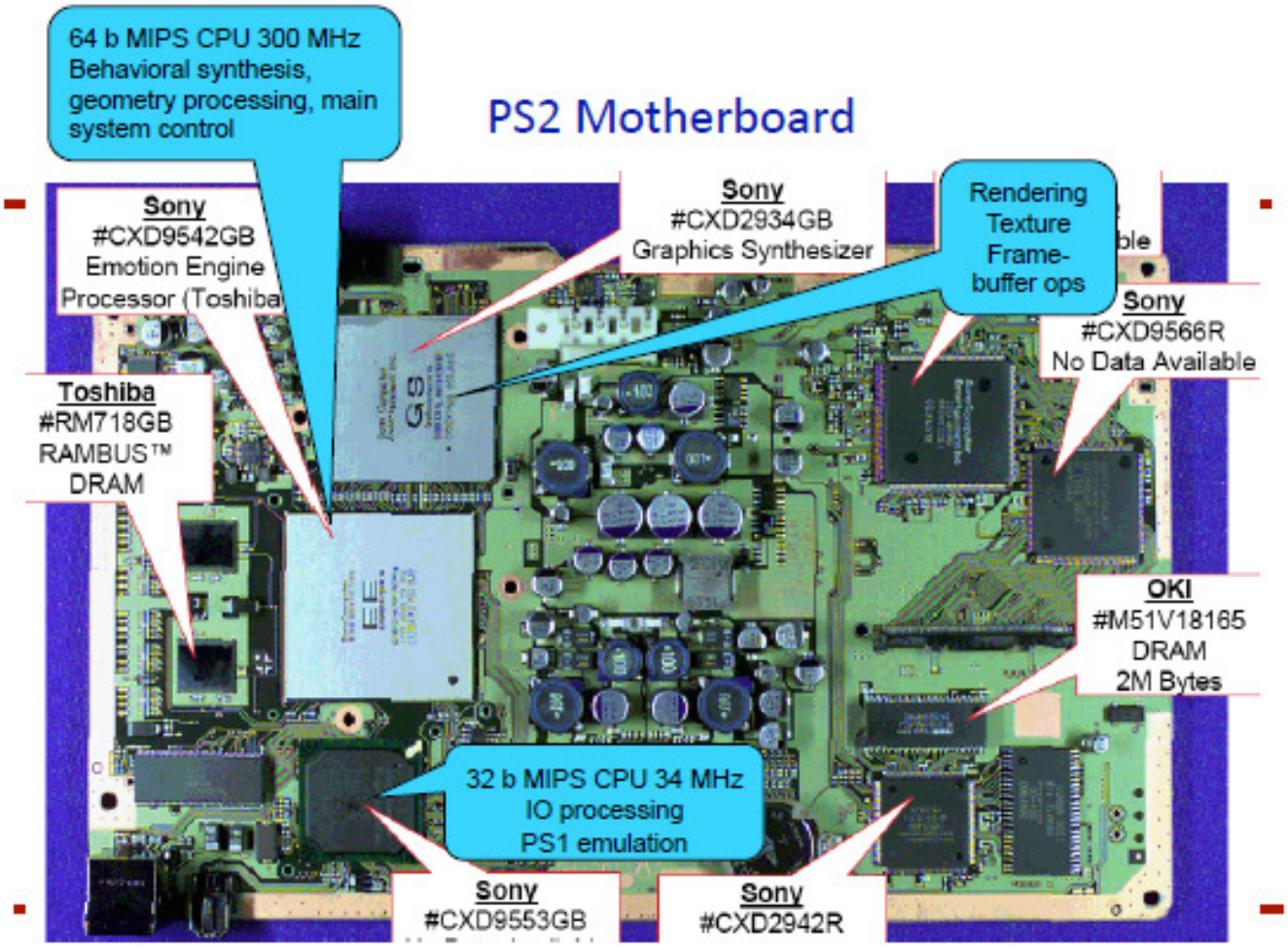
- Battery 900 mAhr
- 3.5 hr talk
  - ~ 1 W
- 8 days standby
  - ~ 1mW

Back Side



Courtesy of Portelligent

PS2 Motherboard



Courtesy of Portelligent

Tools and Documents:

---- https://svn.cs.georgetown.edu/svn/projects2/520-2012spring/ (250-374-developer, y(&qwqsq)
Your branch, course documents (syllabus, lecture notes, HW)

---- https://svn.cs.georgetown.edu/svn/projects/ (250-374-developer, y(&qwqsq)
LC3tools/
PennSim.jar
Electric.jar
MIPStools/
MIPSassembler-MAR_4_1.jar

---- cygwin (unix environment for MS Windows)
(binaries via setup.exe from cygwin web site)
+ defaults + tools (make, grep, sed, awk, gzip, iverilog, subversion, ...)

---- Subversion (svn, commandline client)

---- Verilog
Icarus Verilog (iverilog, binaries and source code online)

---- LC3trunk/src
lcc (C compiler targeted to LC3 assembly language)
lc3tools
lc3as (LC3 assembler)
lc3pre (pre-processor for LC3 assembly language)
obj2bin (LC3 machine code conversion to verilog-readable format)

---- LC3trunk/src/Modules/
int.asm, putc.asm, ... (source code for part of our LC3 OS)
putc-driver.asm, ... (source code to test modules)

---- LC3trunk/src/os-src/
lc3os-bare.asm (skeleton source code for your LC3 OS)

---- LC3trunk/src/Assembly_test_code/
testALLinstr.asm, ... (code to test LC3 execution)

---- LC3trunk/Makefile
(How to build LC3 tools, LC3 assembly, C compiling, verilog conversion, ...)

---- LC3trunk
See the READMEs in LC3trunk/, LC3trunk/lib/, LC3trunk/src/, ...
See LC3trunk/lib/*.jelib for Electric-based implementation of LC3

THINGS TO DO:
---- set your path for your bin directory:
cd ~
vi .bash_profile
PATH=/Users/squier/myBranch/trunk/bin:${PATH}
source .bash_profile; echo $PATH;

[Note: no spaces, " ", in path. If your directory path has spaces, use links:]

ln -s /Users/squier/My\ Stupid\ Path\ To\ My\ Branch myBranch