

Introduction

CPE380/CS380, Fall 2024

Hank Dietz

<http://aggregate.org/hankd/>

Course Overview

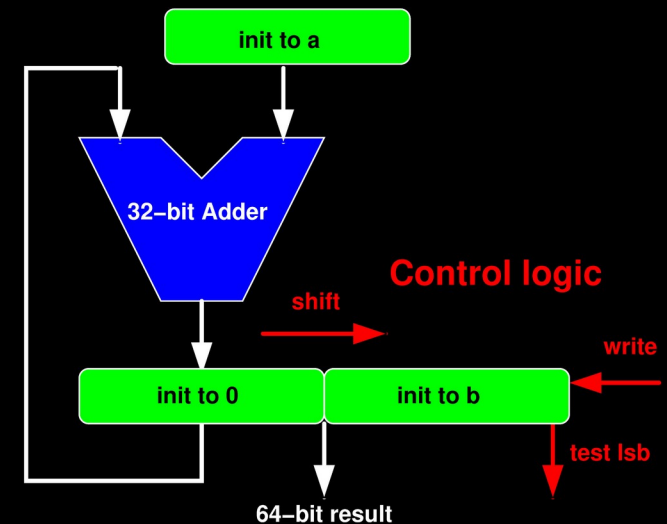
- You know how to write a simple program... from **CS** courses
- You know how to build simple combinatorial and sequential logic circuits from **ECE** courses (especially CPE282 or EE280/EE281)
- **This course fills the gap between the two:**
 - So you can **better specify & use** that stuff
 - So you can **create** the stuff in between
 - There will be implementations in **Verilog**

Verilog 32-bit Multiplier

```
module mul(ready, c, a, b, reset, clk);

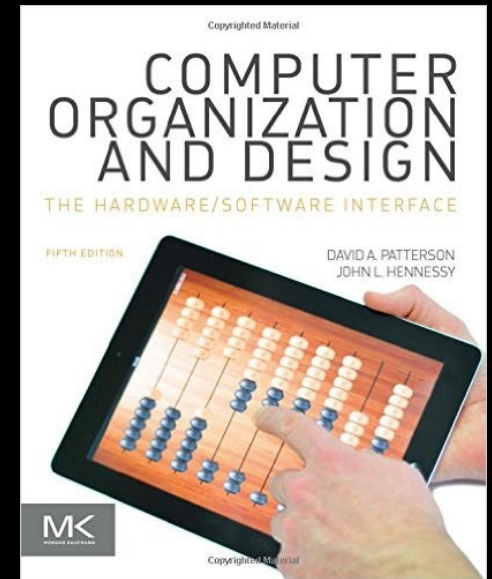
parameter BITS = 32;
input [BITS-1:0] a, b;
input reset, clk;
output reg [BITS*2-1:0] c;
output reg ready;
reg [BITS-1:0] d;
reg [BITS-1:0] state;
reg [BITS:0] sum;

always @(posedge clk or posedge reset) begin
  if (reset) begin
    ready <= 0;
    state <= 1;
    d <= a;
    c <= {{BITS{1'b0}}, b};
  end else begin
    if (state) begin
      sum = c[BITS*2-1:BITS] + d;
      c <= (c[0] ? {sum, c[BITS-1:1]} :
            (c >> 1));
      state <= {state[BITS-2:0], 1'b0};
    end else begin
      ready <= 1;
    end
  end
end
endmodule
```



Textbook

- The text is:
Computer Organization & Design, 5th Edition: The Hardware/Software Interface by Patterson & Hennessy
- You can use any MIPS edition from 2nd – 6th, but we'll reference sections from the 5th
- We will not assign problems from the text
- Lots of additional materials at the course URL and presented in class... **text is reference only**



Grading & Such

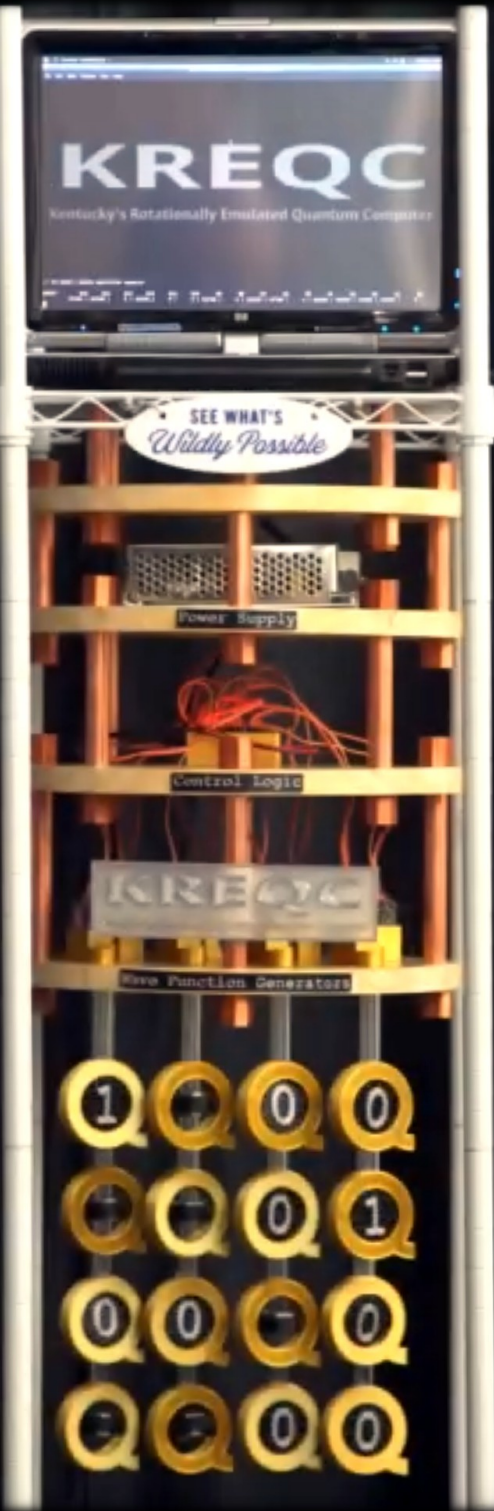
- One individual Verilog project, ~15%
- Three team projects, ~10% each
- Three homework assignments, ~10% each
- In-person final exam, ~25%
- Material from **lectures**, the **text** as cited, **canvas**, or from the **course URL**:
<http://aggregate.org/CPE380/>
- You are expected to regularly attend class
- I try not to curve much; always in your favor

Course Content

Lectures	Topic
1	Introduction
3	Verilog (individual project)
3	Multi-cycle machine (team project)
3	Machine & assembly languages (homework)
2	Single-cycle machine (team project)
3	Integer & float arithmetic (homework)
5	Pipelined machine (team project)
4	Memory hierarchy and I/O (homework)
2	Parallel processing and performance
1	<i>reserved for schedule slippage</i>
1	Review for final exam

Me (and why I'm biased)

- **Hank Dietz**, ECE Professor and James F. Hardyman Chair in Networking
- Office: **203 Marksbury**
- Research in parallel compilers & architectures:
 - Built 1st Linux PC cluster supercomputer
 - Antlr, AFNs, SWAR, FNNs, MOG, ...
 - Various awards & world records for best price/performance in supercomputing
- Lab: **108/108A Marksbury** – I have **TOYS!**



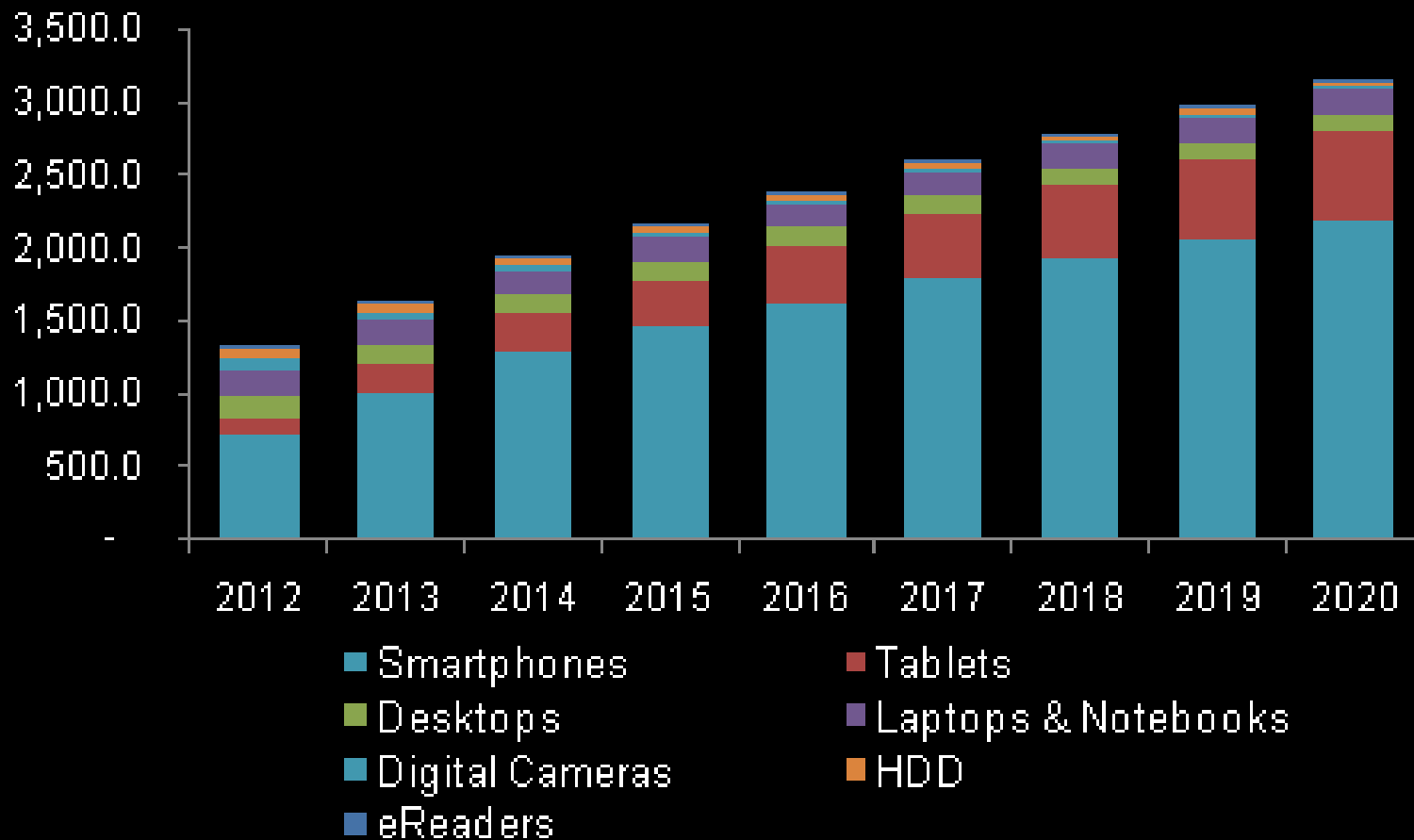
Electrical & Computer Engineering



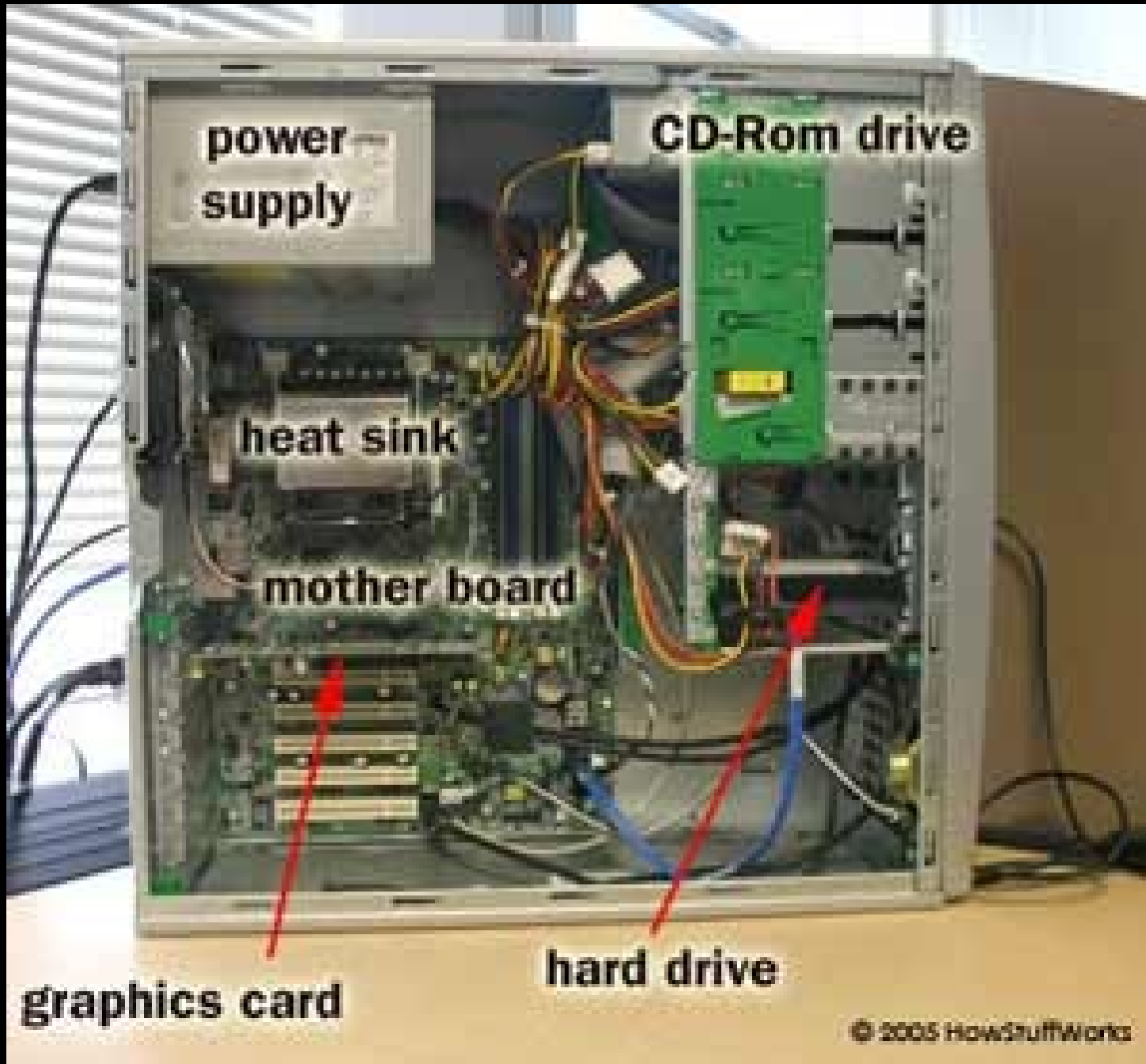
Let's Talk About Computers

- Embedded computers, IoT (Internet of Things)
- Personal Mobile Devices (PMDs)... usually “smart phones” and tablets
- Personal Computers (PCs)
- Servers
- Supercomputers
- Clusters, Farms, Grids, and Clouds (Warehouse Scale Computers – WSC, Software as a Service – SaaS)

M-Unit Sales, Global Personal Electronics



What's Inside?







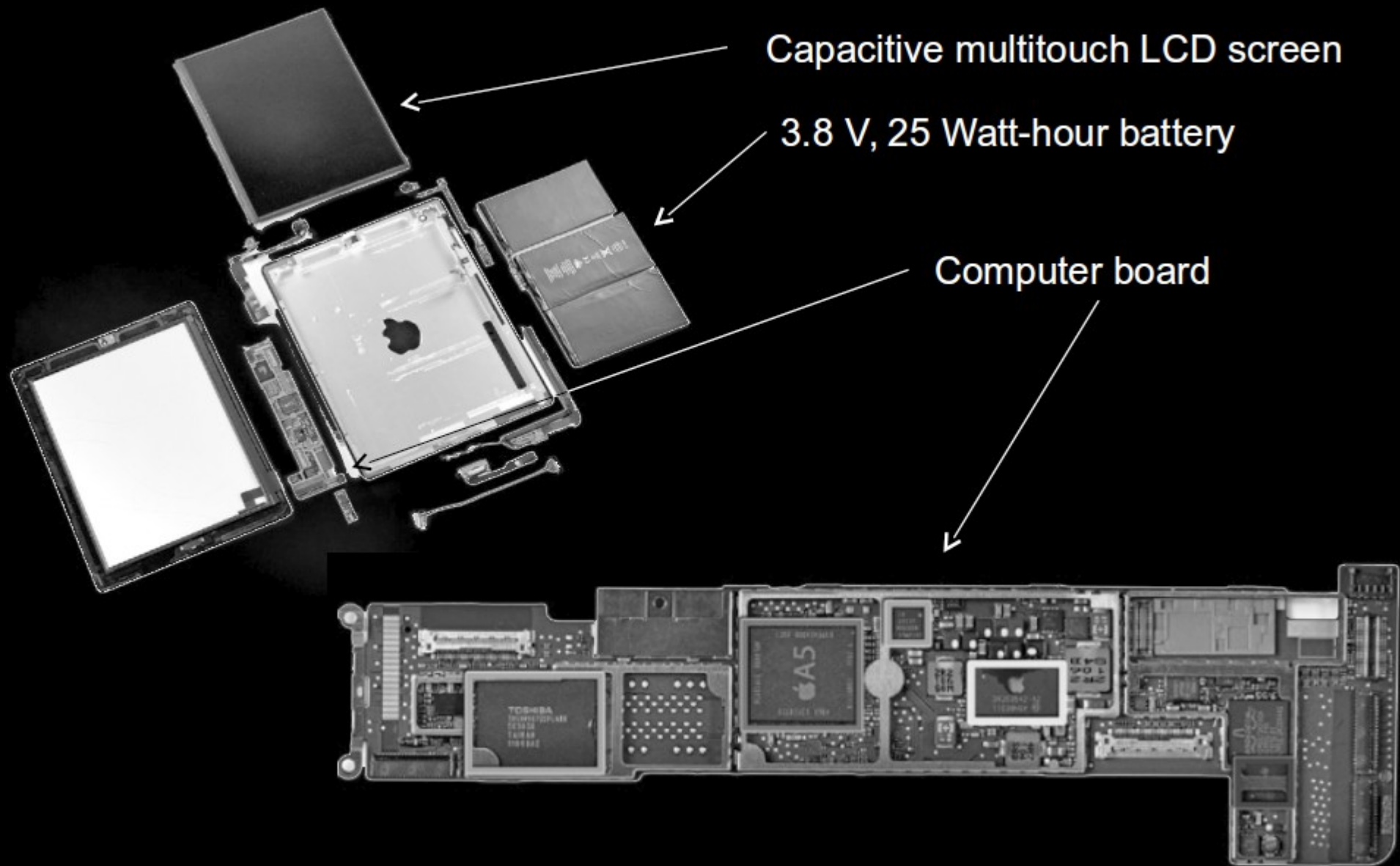
LCD

heat sink

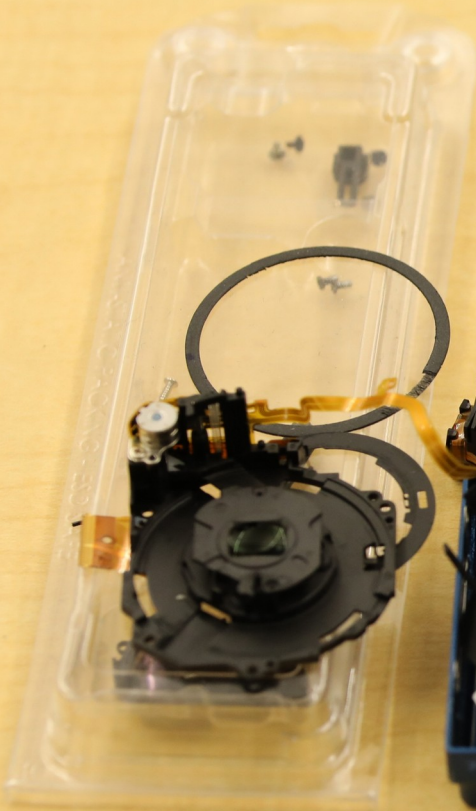
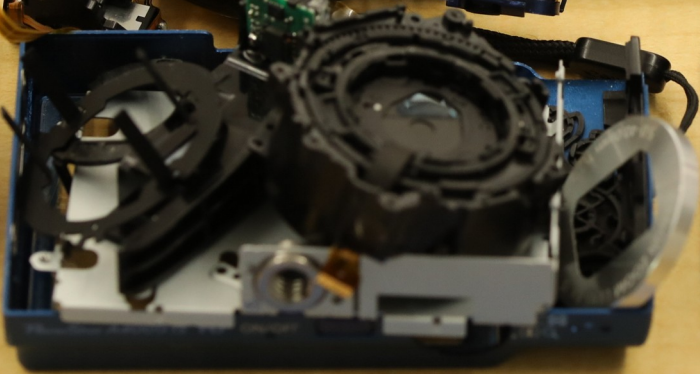
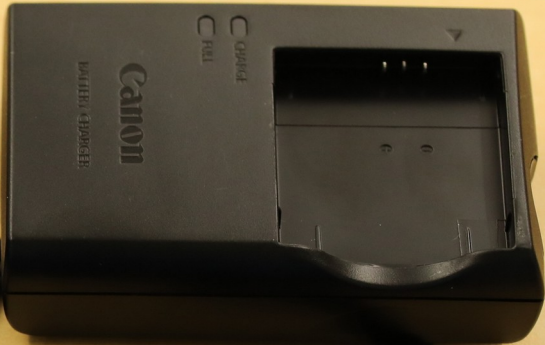
mother board

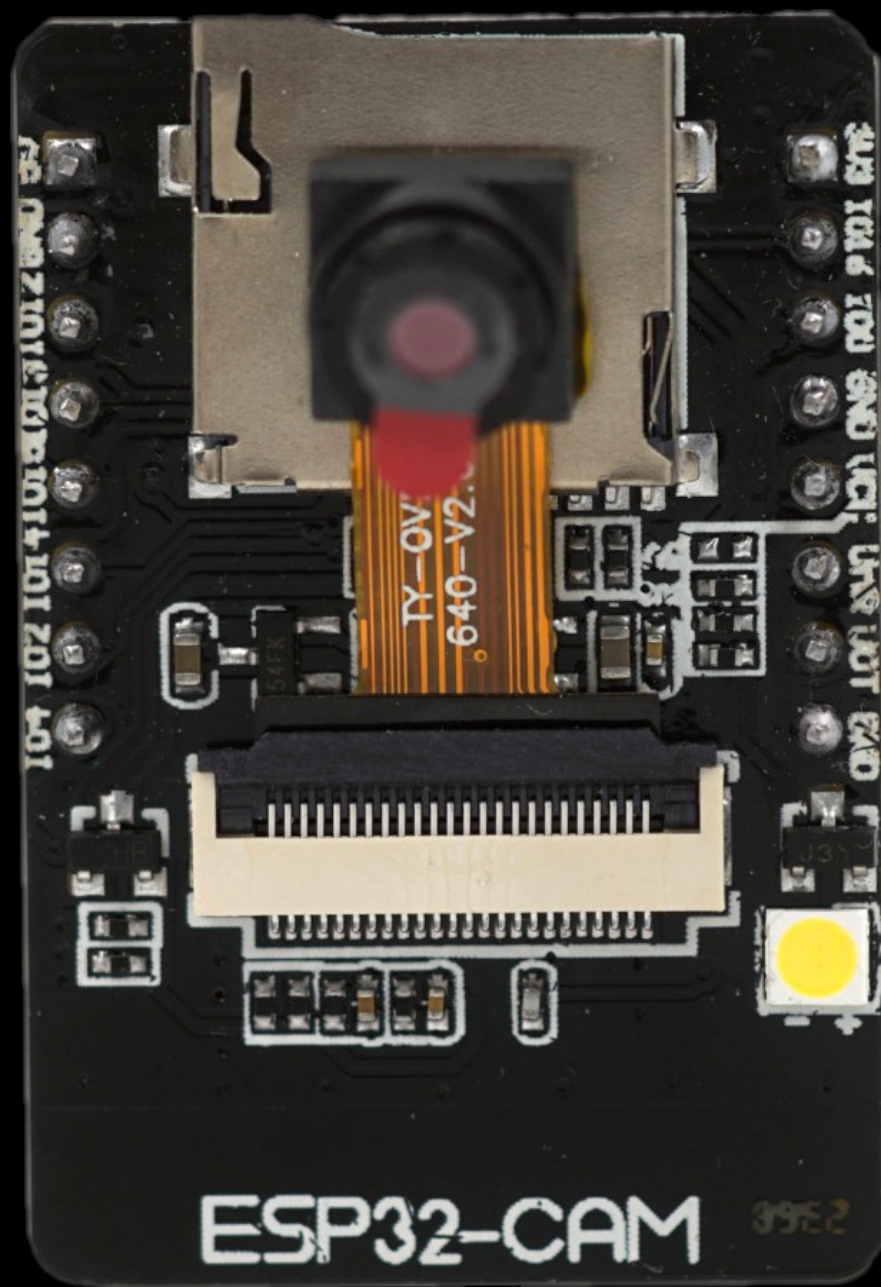
processor

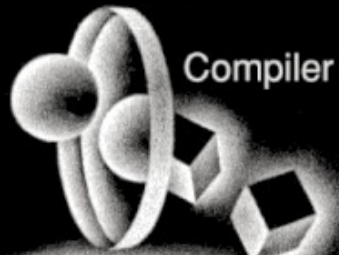
graphics chip









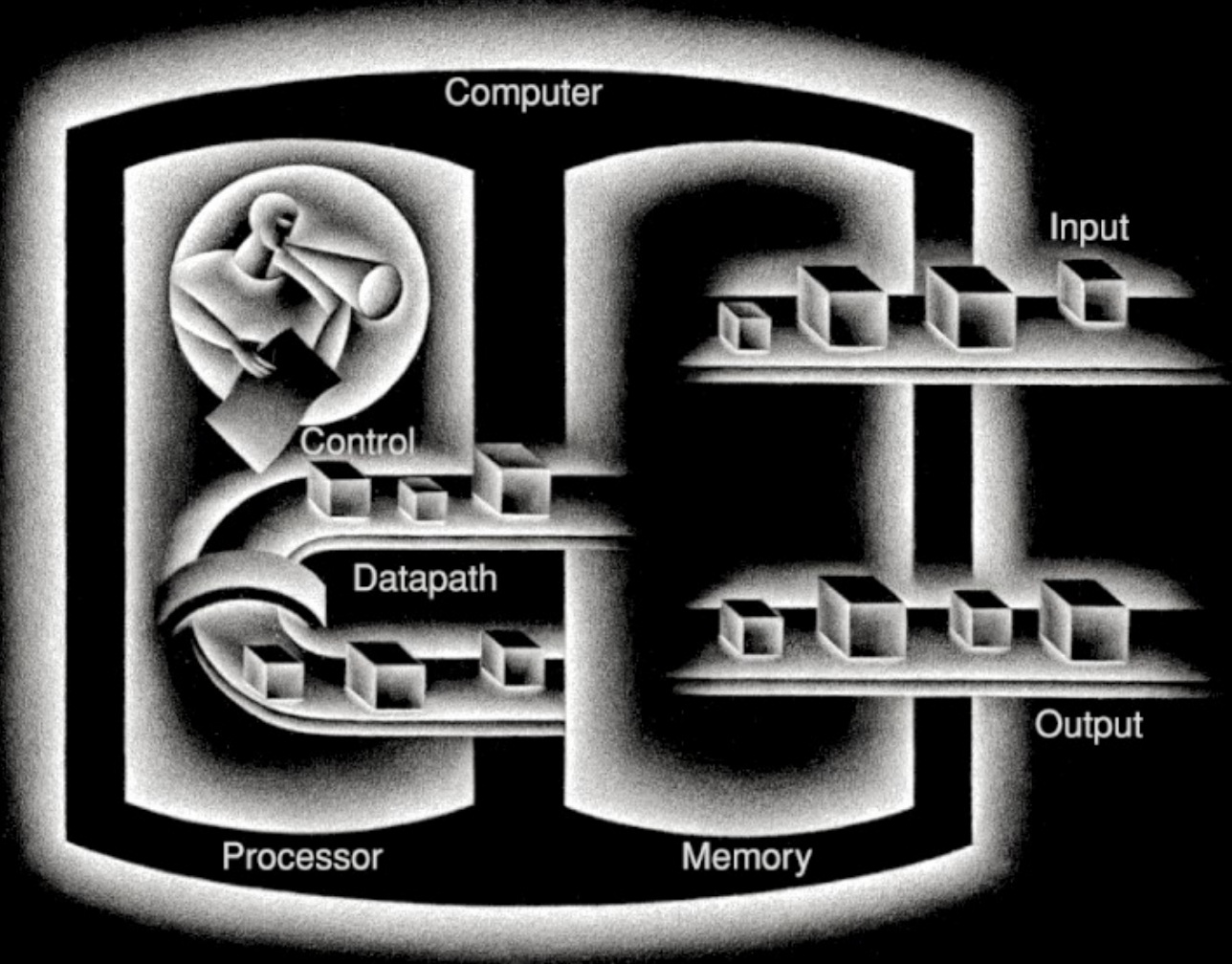


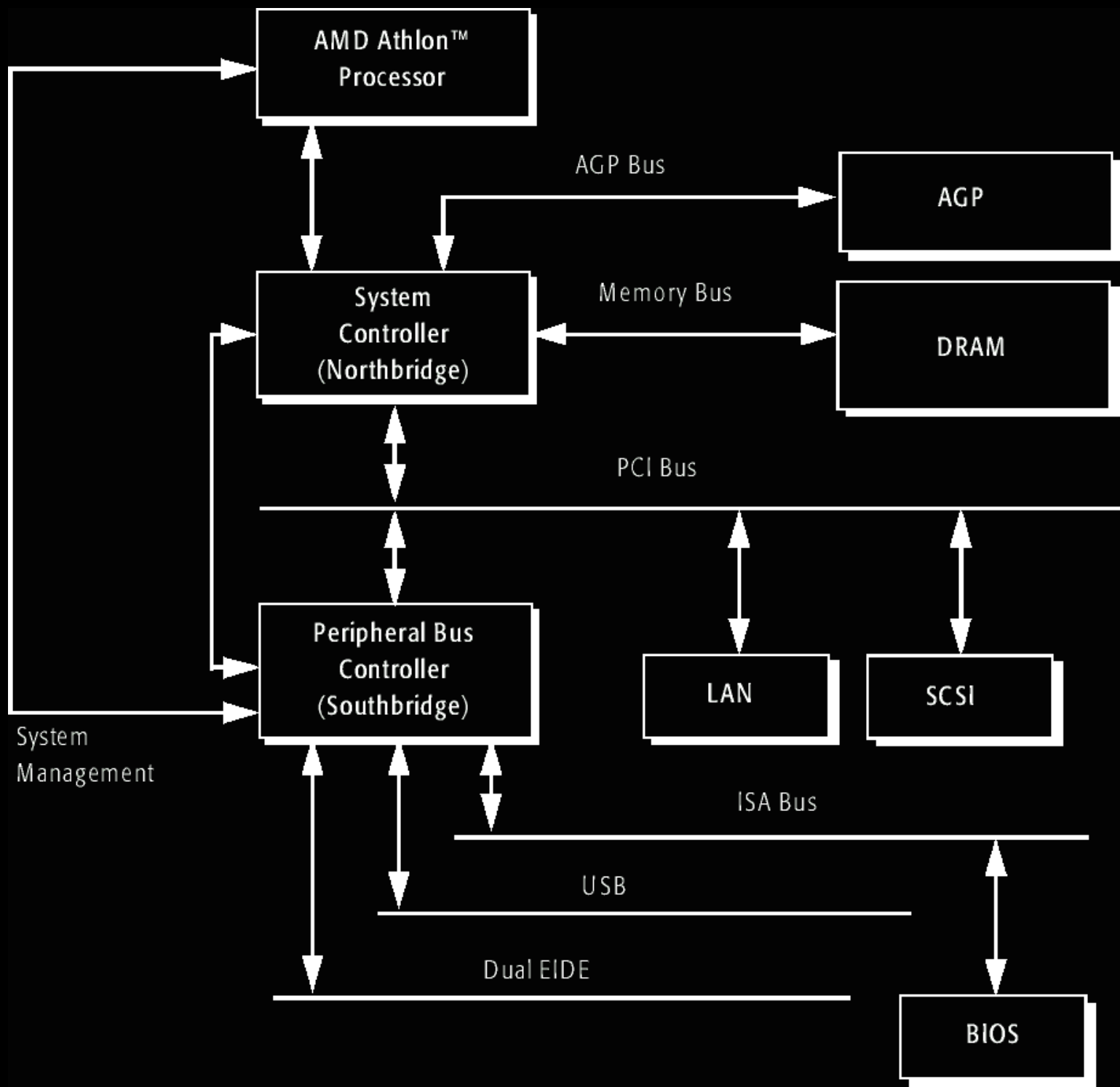
Compiler

Interface



Evaluating performance





Processor Terminology

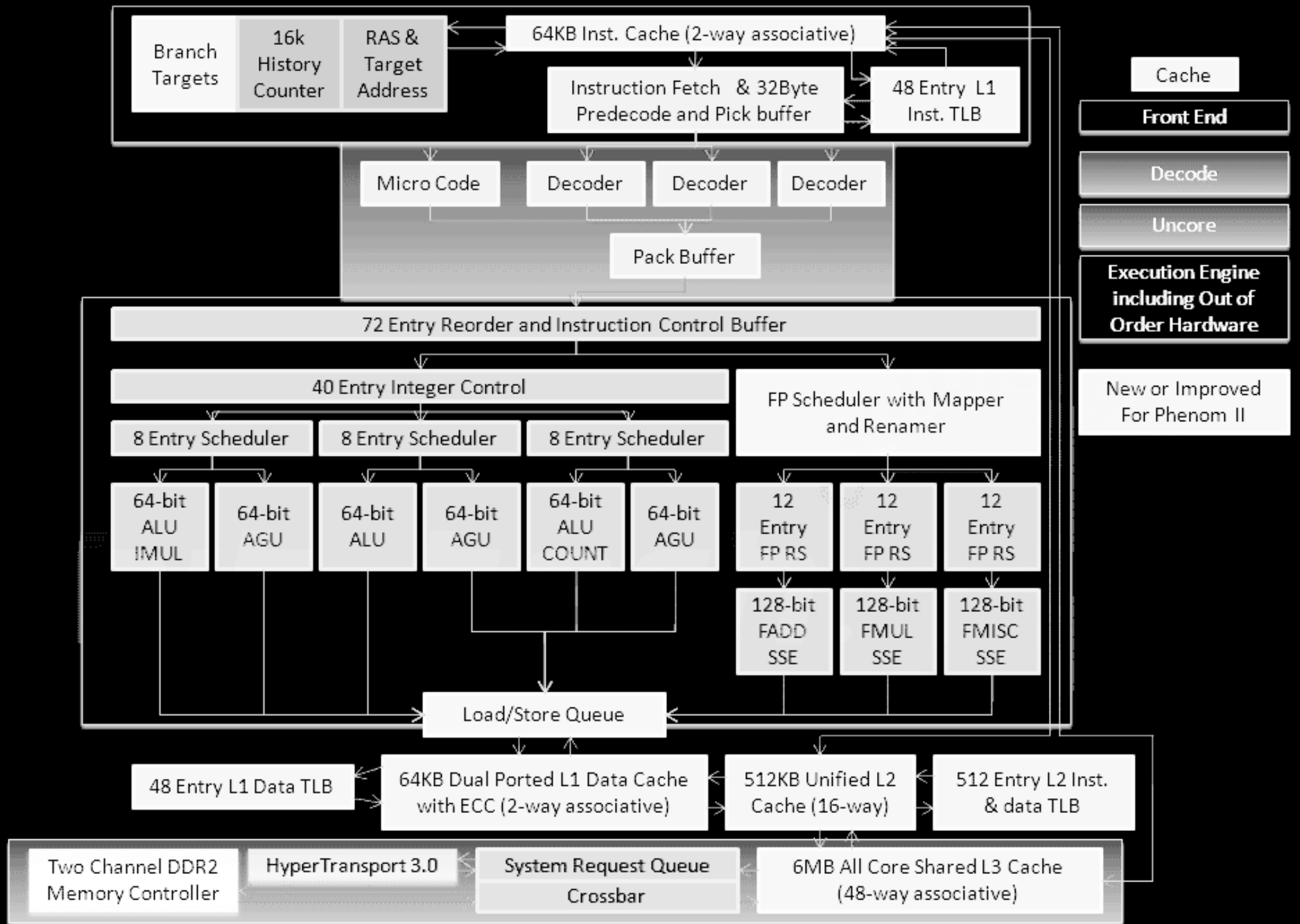
- CPU – Central Processing Unit
- PE, Core – Processing Element
- Processor – CPU or chip containing PEs
- “Computer Family” – same ISA
- x86, IA32, x64/AMD64 – Intel 386-based ISAs
- MIPS, ARM, SPARC – other common ISAs
- DSP – Digital Signal Processor
- GPU – Graphics Processing Unit
- Tensor – Matrix support for neural networks
- Quantum – Combinatorial use of **superposition**

Complexity is Increasing!

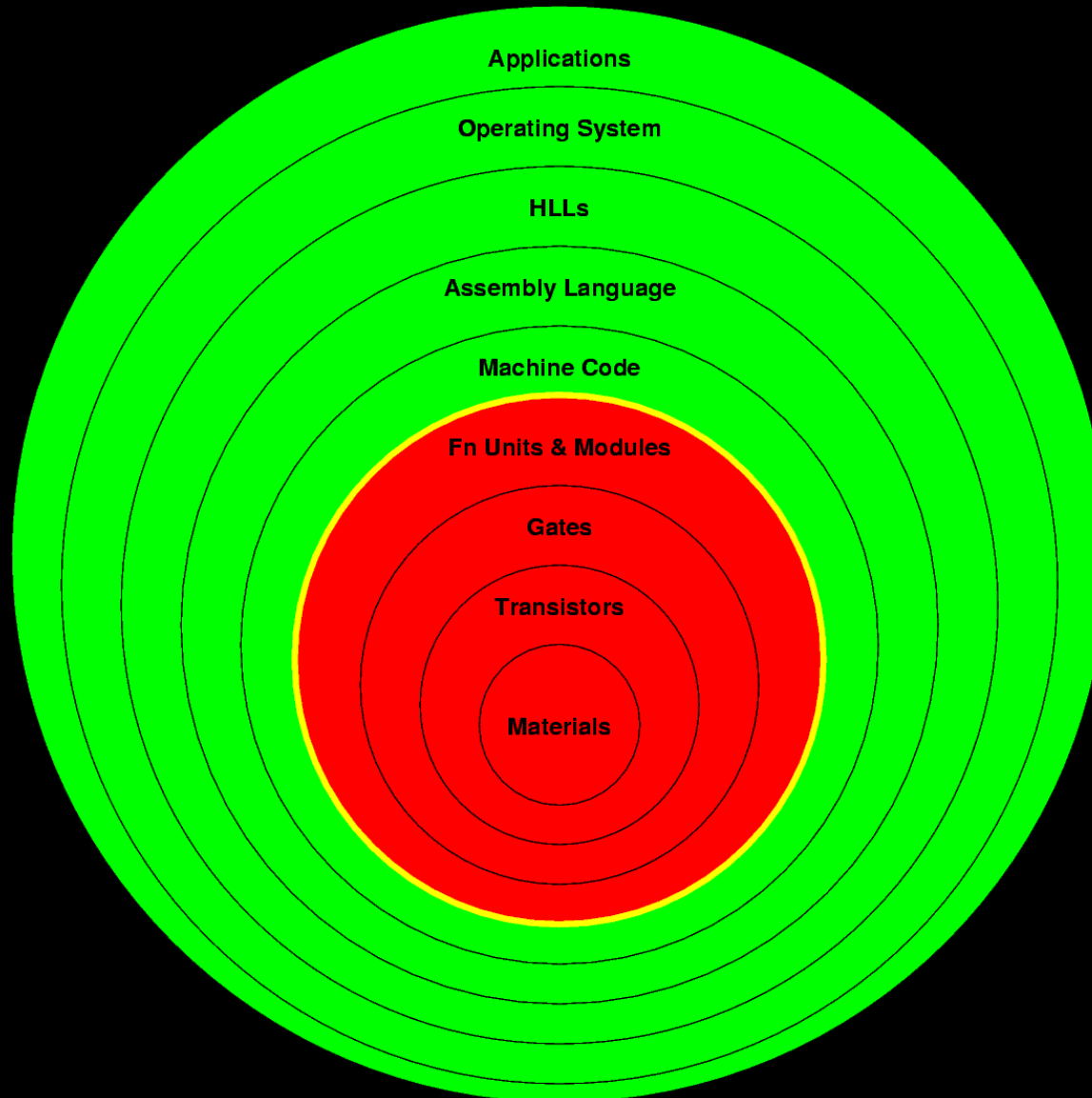
- Lots of things you use every day have **BILLIONS** of components!
- You don't live long enough to know it all



Frontier Supercomputer:
8,730,112 cores, 1.102 Exaflop/s



Abstraction “Onion”



Software Layers

- Applications...
- Operating Systems (OS)...
- High-Level Languages (HLLs)
Aka, High Order Languages (HOLs)
 - Designed for humans to write & read
 - Modularity
 - Abstract data types, type checking
 - Assignment statements
 - Control constructs
 - I/O statements

Instruction Set Architecture

- ISA defines HW/SW interface
- **Assembly Language**
 - Operations match hardware abilities
 - Relatively simple & limited operations
 - Mnemonic (human readable?)
- **Machine Language**
 - Bit patterns – 0s and 1s
 - Actually executed by the hardware

High-level
language
program
(in C)

```
swap(int v[], int k)
{int temp;
  temp = v[k];
  v[k] = v[k+1];
  v[k+1] = temp;
}
```

↓
Compiler

Assembly
language
program
(for MIPS)

```
swap:
  muli $2, $5, 4
  add  $2, $4, $2
  lw   $15, 0($2)
  lw   $16, 4($2)
  sw   $16, 0($2)
  sw   $15, 4($2)
  jr   $31
```

↓
Assembler

Binary machine
language
program
(for MIPS)

```
000000001010000100000000000011000
000000000000110000001100000100001
100011000110001000000000000000000
100011001111001000000000000000100
101011001111001000000000000000000
101011000110001000000000000000100
0000001111100000000000000000001000
```

Hardware Layers

- Function-block organization
- Gates & Digital Logic (CPE282 stuff)
- Transistors
 - Used as bi-level (saturated) devices
 - Amplifiers, not just on/off switches
- Materials & Integrated Circuits
 - Implementation of transistors, etc.
 - Analog properties

Who Does What?

- Instruction Set Design, by *Architect*
 - Machine & Assembly Languages
 - **“Computer Architecture”**
 - **Instruction Set Architecture** / Processor
- Computer Hardware Design, by *Engineer*
 - Logic Design & Machine Implementation
 - **“Processor Architecture”**
 - **“Computer Organization”**

How To Use Layers

- Things are too complex to “know everything”
- Need to know only layers adjacent
 - Makes design complexity reasonable
 - Makes things reusable
- Can **tunnel** to lower layers
 - For efficiency
 - For special capabilities

8 Great Ideas

- Design for Moore's Law
- Abstraction
- Make the common case fast
- Pipelining
- Parallelism
- Prediction
- Hierarchy of memories
- Dependability via redundancy



SI Terminology Of Scale

1000^1	kilo	k	1000^{-1}	milli	m
1000^2	mega	M	1000^{-2}	micro	u
1000^3	giga	G	1000^{-3}	nano	n
1000^4	tera	T	1000^{-4}	pico	p
1000^5	peta	P	1000^{-5}	femto	f
1000^6	exa	E			

- 1000^x vs. 1024^x
- 1 **Byte (B)** is 8-10 bits (**b**), 4 bits in a **Nybble**
- Hertz (**Hz**) is frequency (vs. period)

Conclusion

- LOTS of stuff to know about...
focus of this course is the basic stuff around the ISA and its implementation
- A lot of computer system design is about how to build efficient systems despite incredibly high and rapidly increasing system complexity
- Look at the history references on the WWW:
not to memorize who, what, when, & where,
but to *see trends...*