Introduction

EE685, Fall 2024

Hank Dietz

http://aggregate.org/hankd/

University of Kentucky

Course Overview

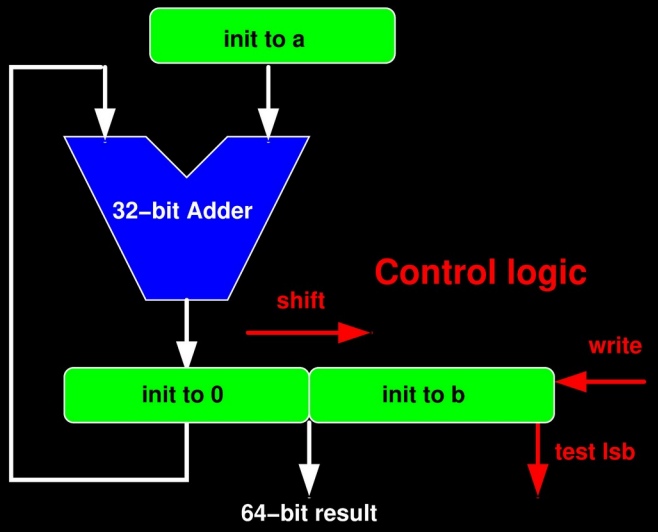
- Sort-of like CPE380/480, but broader & deeper
- **There will be *Verilog***, and you'll design stuff
- A lot more of the ***advanced*** stuff
  - Fancy things inside processors
  - Lots of memory & parallel architecture
- Higher-level models and/or simulation

Verilog 32-bit Multiplier

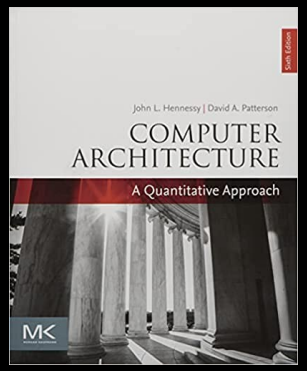
```
module mul(ready, c, a, b, reset, clk);

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 Architecture, 6th Edition: A Quantitative Approach,
  by Patterson & Hennessy
- Same text as was used in CPE480…
  and *optionally* referenced here
- Lots of additional materials…

Grading & Such

- Four assignments, ~10% each
- Midterm exam, ~20%
- Final exam, ~40%

- Material cited from the text, from lectures, from canvas, or from the course URL:
  `http://aggregate.org/EE685/`

- You are expected to regularly attend class
- I try not to curve much; always in your favor

Course Content

- Precise content depends on you:
  - How many of you took CPE380? When?
  - How many of you took CPE480? When?

This course is sort-of **++CPE480**...

- The syllabus is posted, but it is a little vague because things will vary significantly based on the backgrounds of students in the course

Course Content

Lectures	Topic
1	Introduction
3	Verilog (project)
6	Pipelined RISC machine (project)
3	Instruction-level parallelism
1	Review for midterm
1	In-class midterm exam
3	Advanced processor internals (assignment)
5	Memory hierarchy and protection (assignment)
4	Scalable parallel processing and networking
1	Review for final exam

Me (and why I'm biased)

- Hank Dietz, ECE Professor and
  James F. Hardymon 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**!

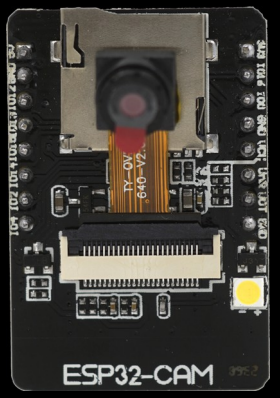

KREQC
Kentucky's Rotationally Emulated Quantum Computer
SEE WHAT'S
Wildly Possible
Power Supply
Control Logic
KREQC
Wave Function Generator


College of Engineering
Electrical & Computer Engineering

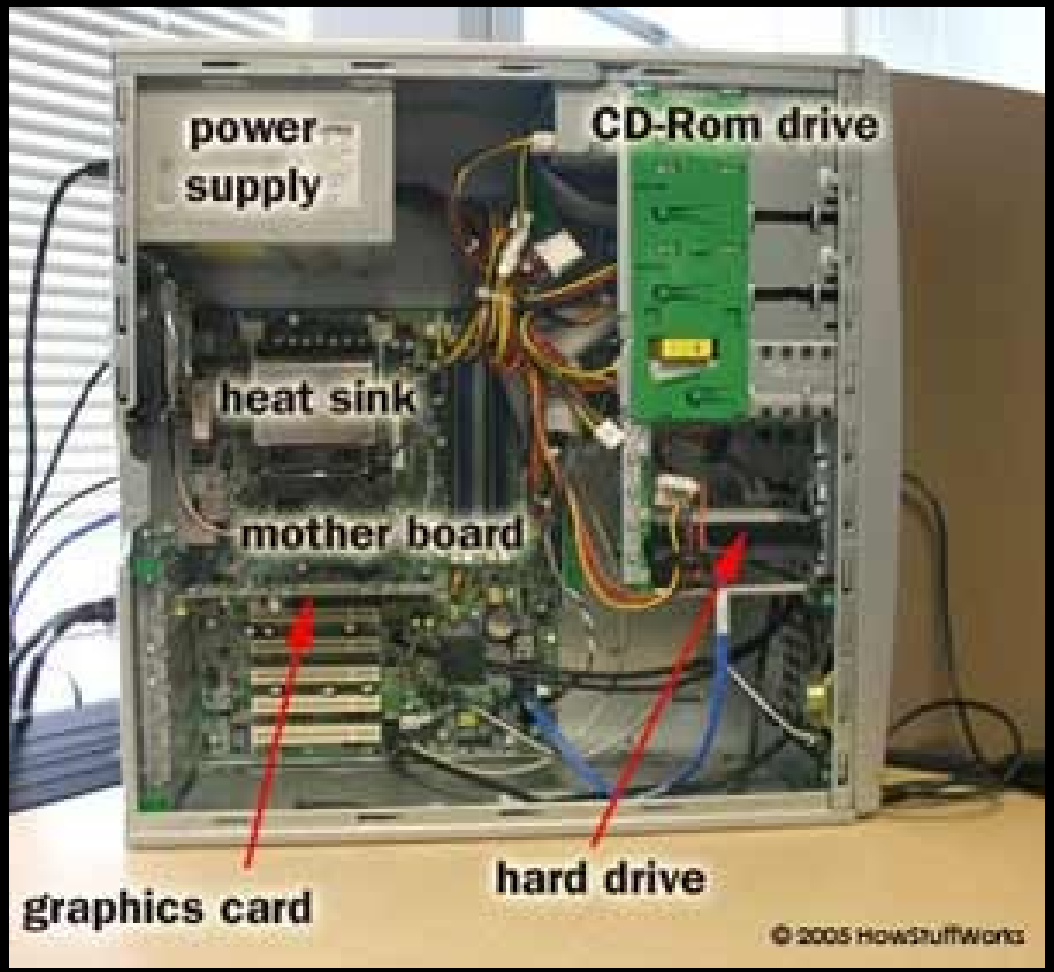

UK ECE

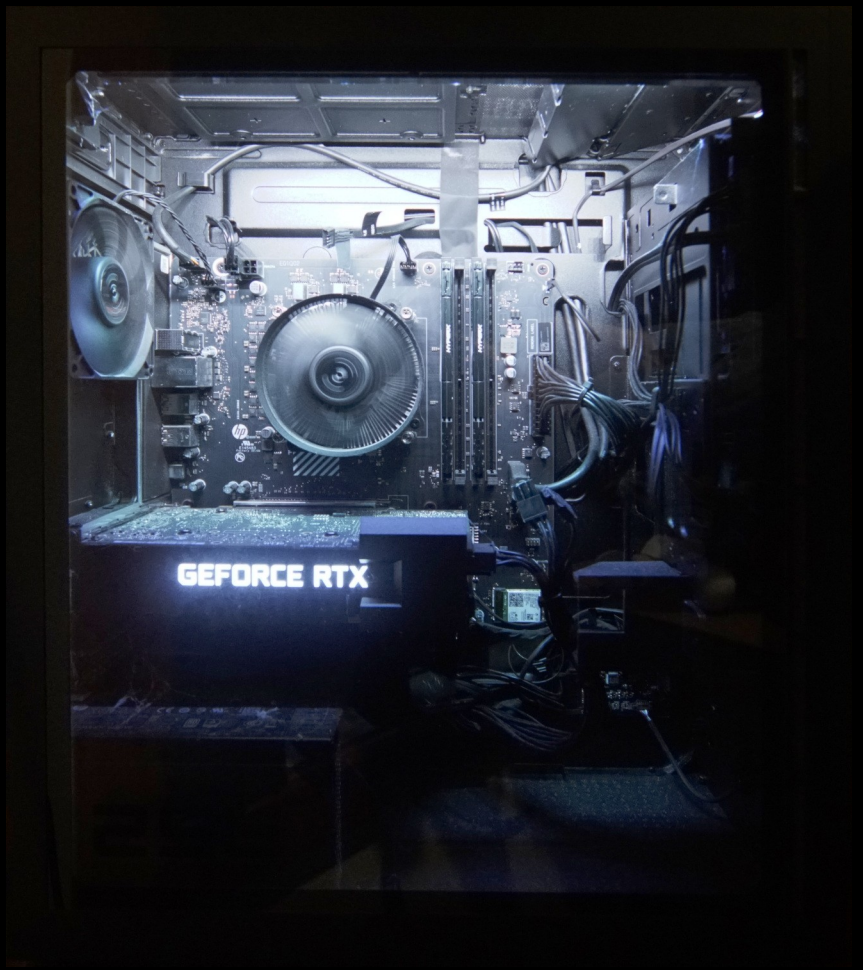
What is Advanced stuff?

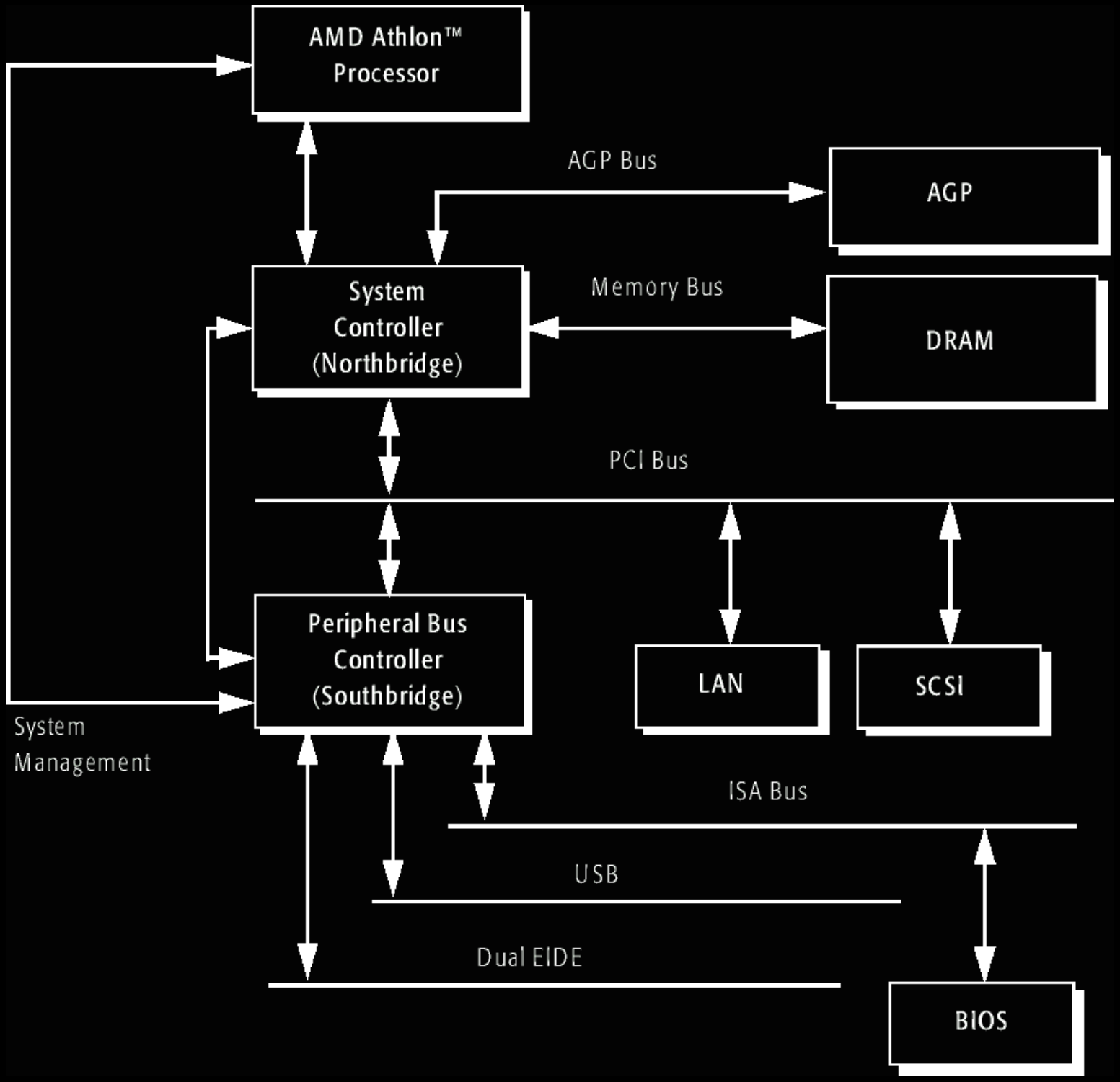
Frontier Supercomputer:
8,730,112 cores, 1.102 Exaflop/s

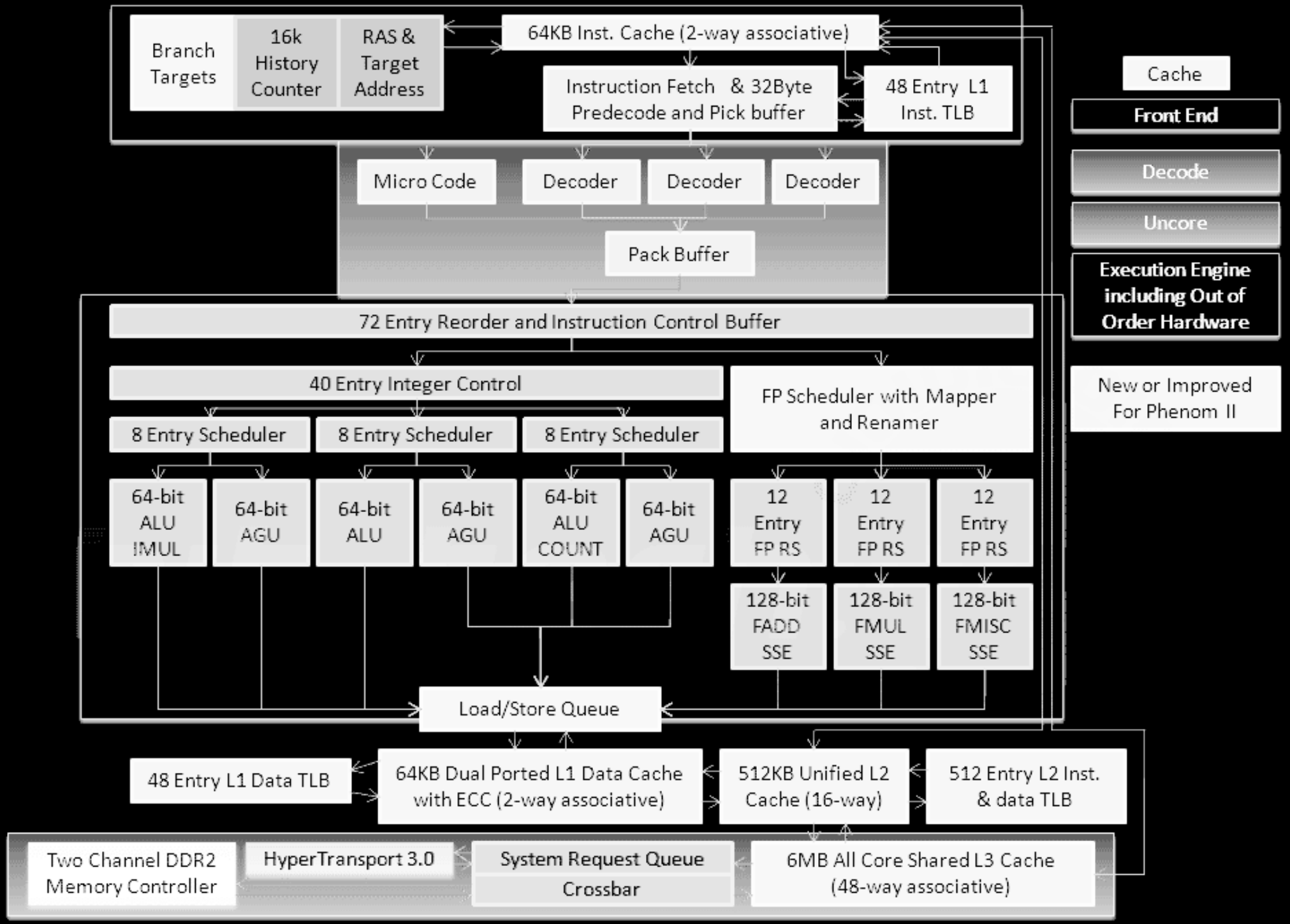


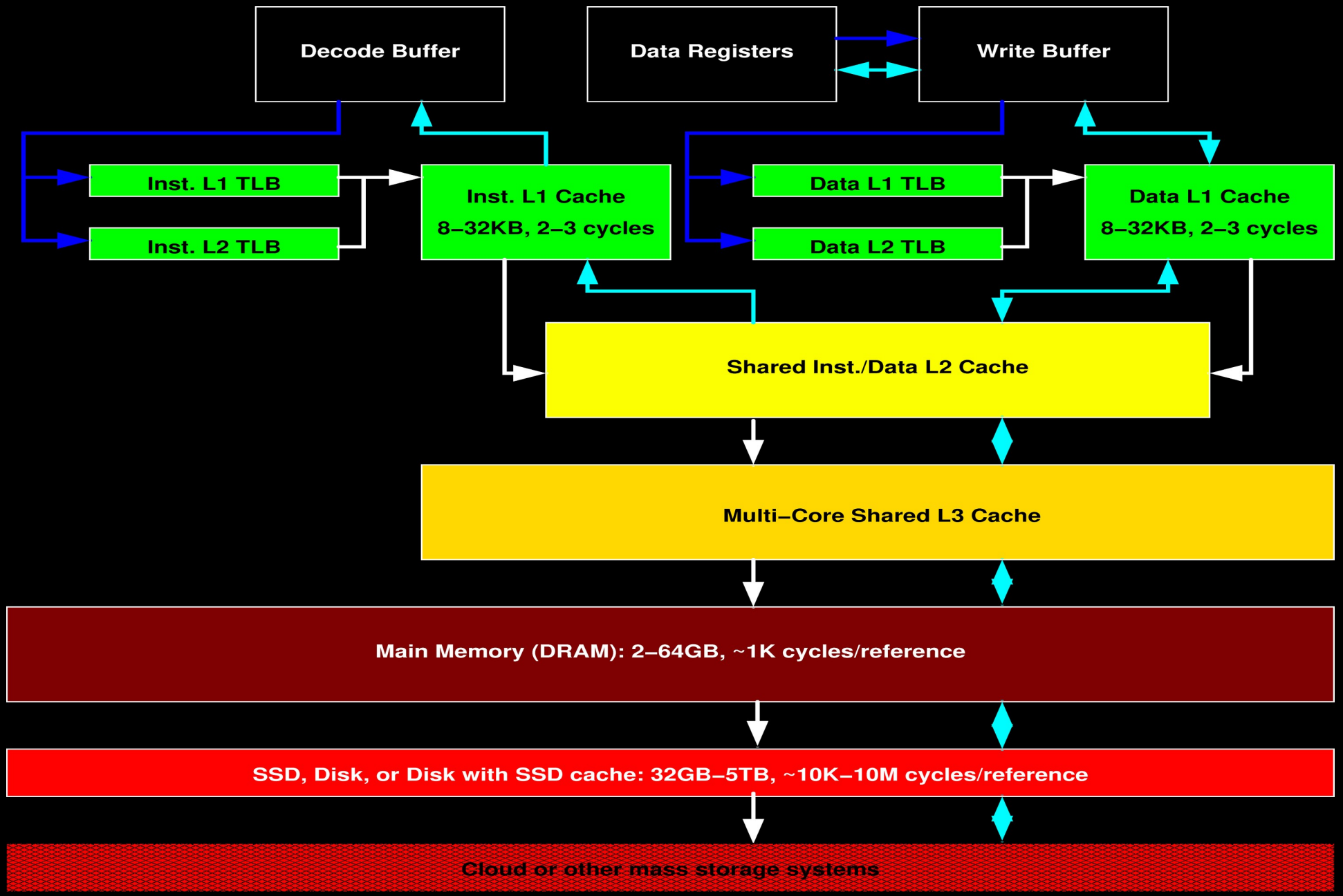
What's inside a PC?



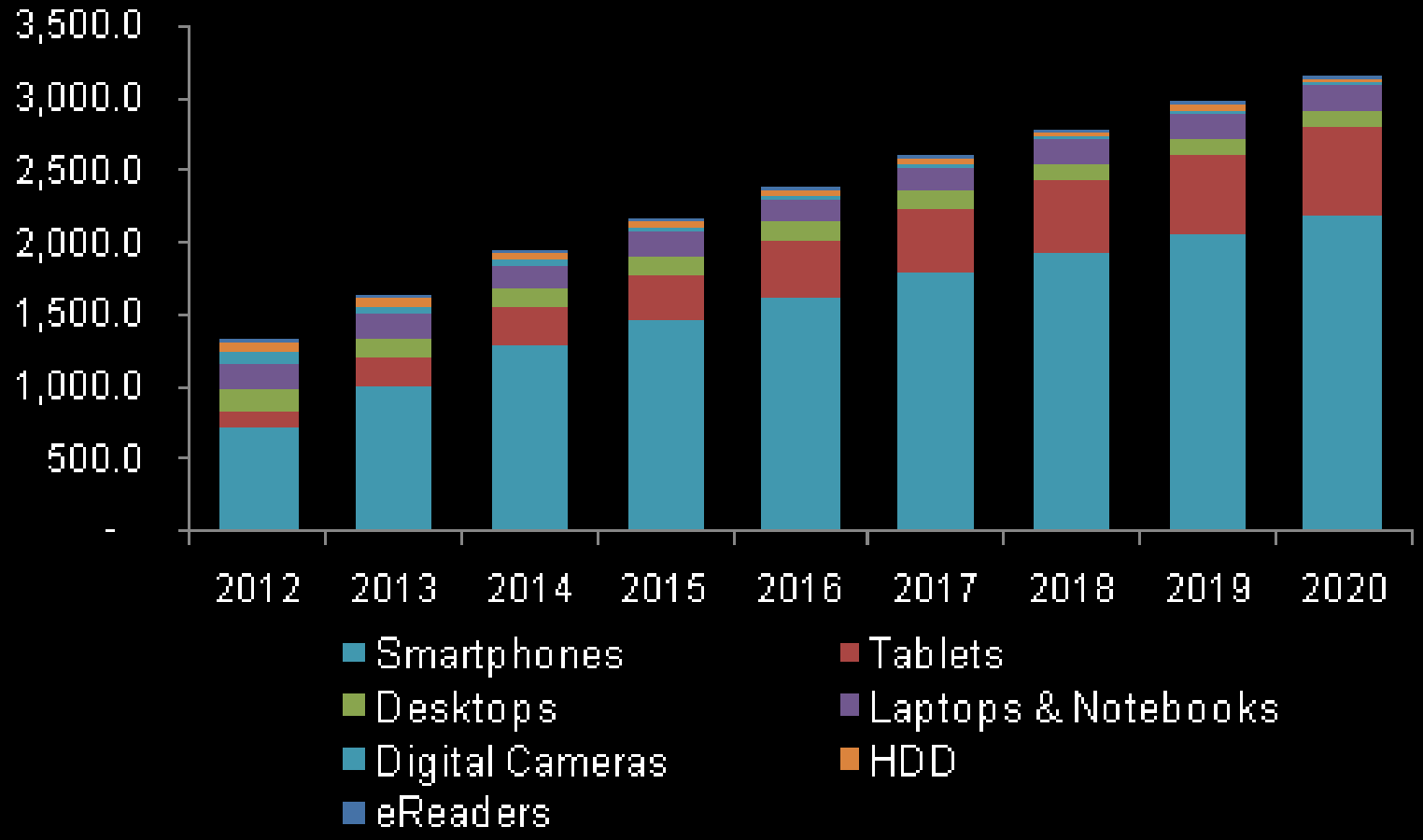



AMD Athlon™
Processor
AGP Bus
AGP
System
Controller
(Northbridge)
Memory Bus
DRAM
PCI Bus
Peripheral Bus
Controller
(Southbridge)
LAN
SCSI
System
Management
ISA Bus
USB
Dual EIDE
BIOS

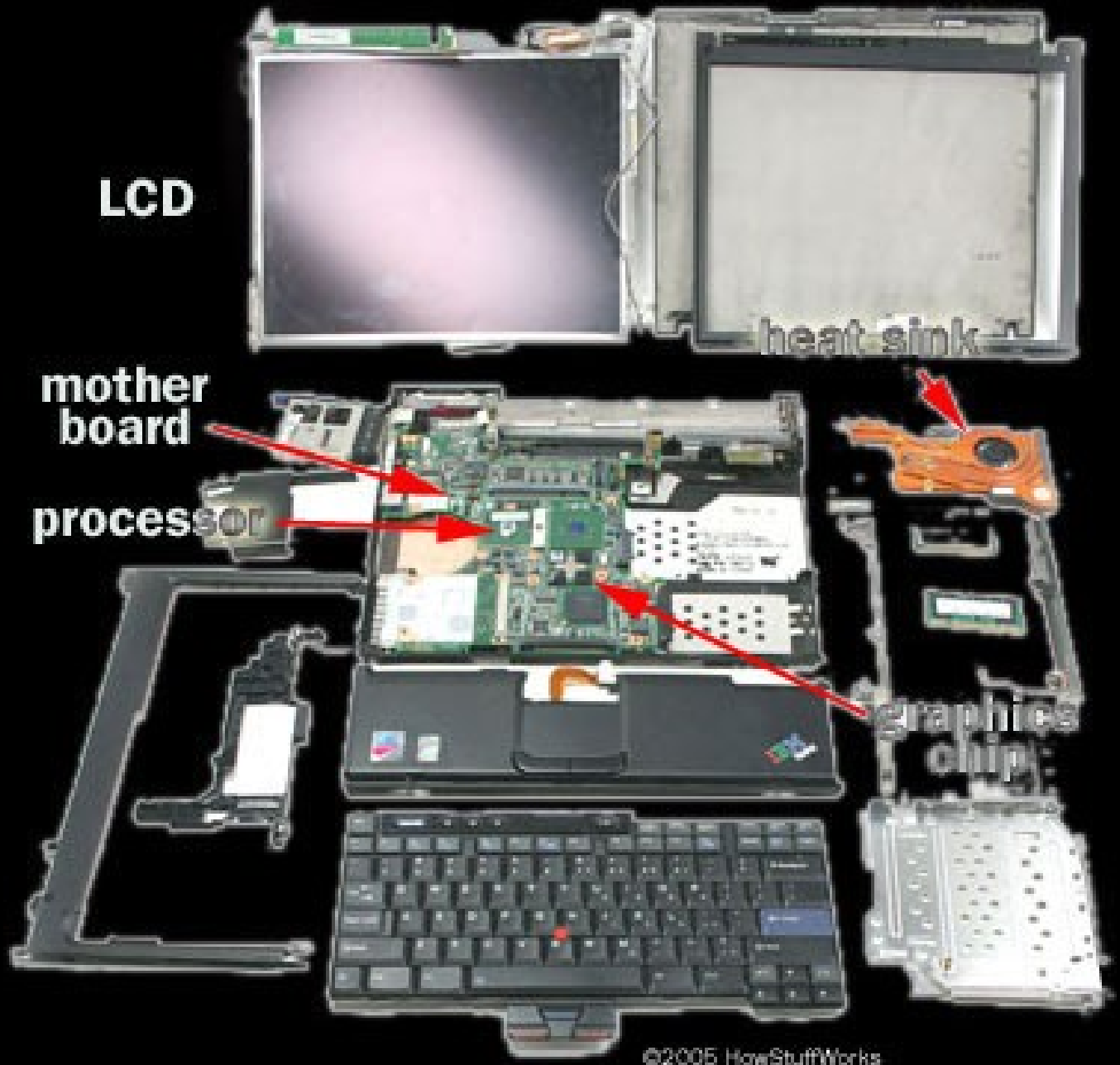

Branch Targets
16k History Counter
RAS & Target Address
64KB Inst. Cache (2-way associative)
Instruction Fetch & 32Byte Predecode and Pick buffer
48 Entry L1 Inst. TLB
Micro Code
Decoder
Decoder
Decoder
Pack Buffer
72 Entry Reorder and Instruction Control Buffer
40 Entry Integer Control
8 Entry Scheduler
8 Entry Scheduler
8 Entry Scheduler
FP Scheduler with Mapper and Renamer
64-bit ALU IMUL
64-bit AGU
64-bit ALU
64-bit AGU
64-bit ALU COUNT
64-bit AGU
12 Entry FP RS
12 Entry FP RS
12 Entry FP RS
128-bit FADD SSE
128-bit FMUL SSE
128-bit FMISC SSE
Load/Store Queue
48 Entry L1 Data TLB
64KB Dual Ported L1 Data Cache with ECC (2-way associative)
512KB Unified L2 Cache (16-way)
512 Entry L2 Inst. & data TLB
Two Channel DDR2 Memory Controller
HyperTransport 3.0
System Request Queue
Crossbar
6MB All Core Shared L3 Cache (48-way associative)
Cache
Front End
Decode
Uncore
Execution Engine including Out of Order Hardware
New or Improved For Phenom II

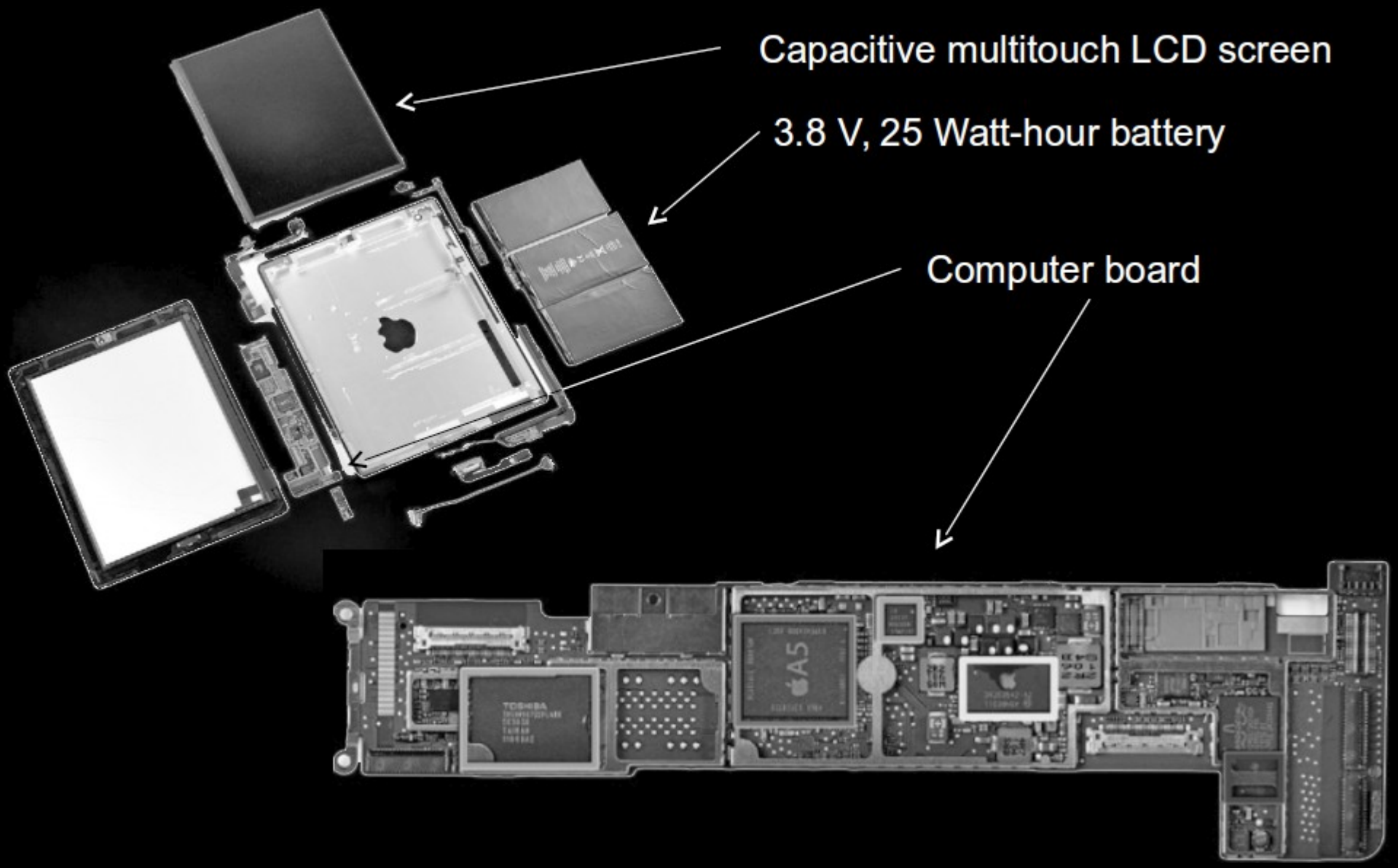

Decode Buffer
Data Registers
Write Buffer
Inst. L1 TLB
Inst. L2 TLB
Inst. L1 Cache
8–32KB, 2–3 cycles
Data L1 TLB
Data L2 TLB
Data L1 Cache
8–32KB, 2–3 cycles
Shared Inst./Data L2 Cache
Multi–Core Shared L3 Cache
Main Memory (DRAM): 2–64GB, ~1K cycles/reference
SSD, Disk, or Disk with SSD cache: 32GB–5TB, ~10K–10M cycles/reference
Cloud or other mass storage systems

M-Unit Sales, Global Personal Electronics

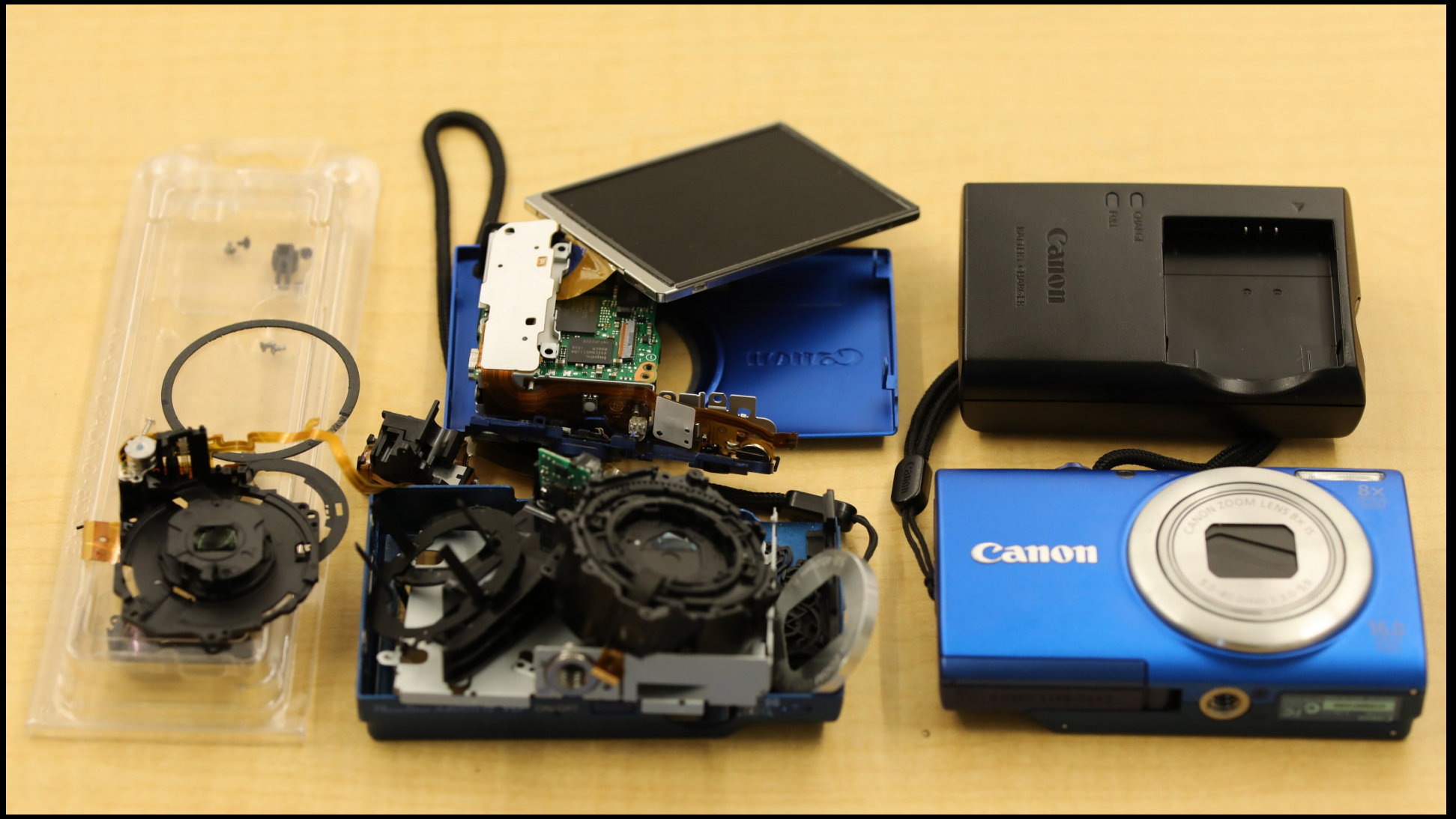


https://www.grandviewresearch.com/industry-analysis/personal-consumer-electronics-market

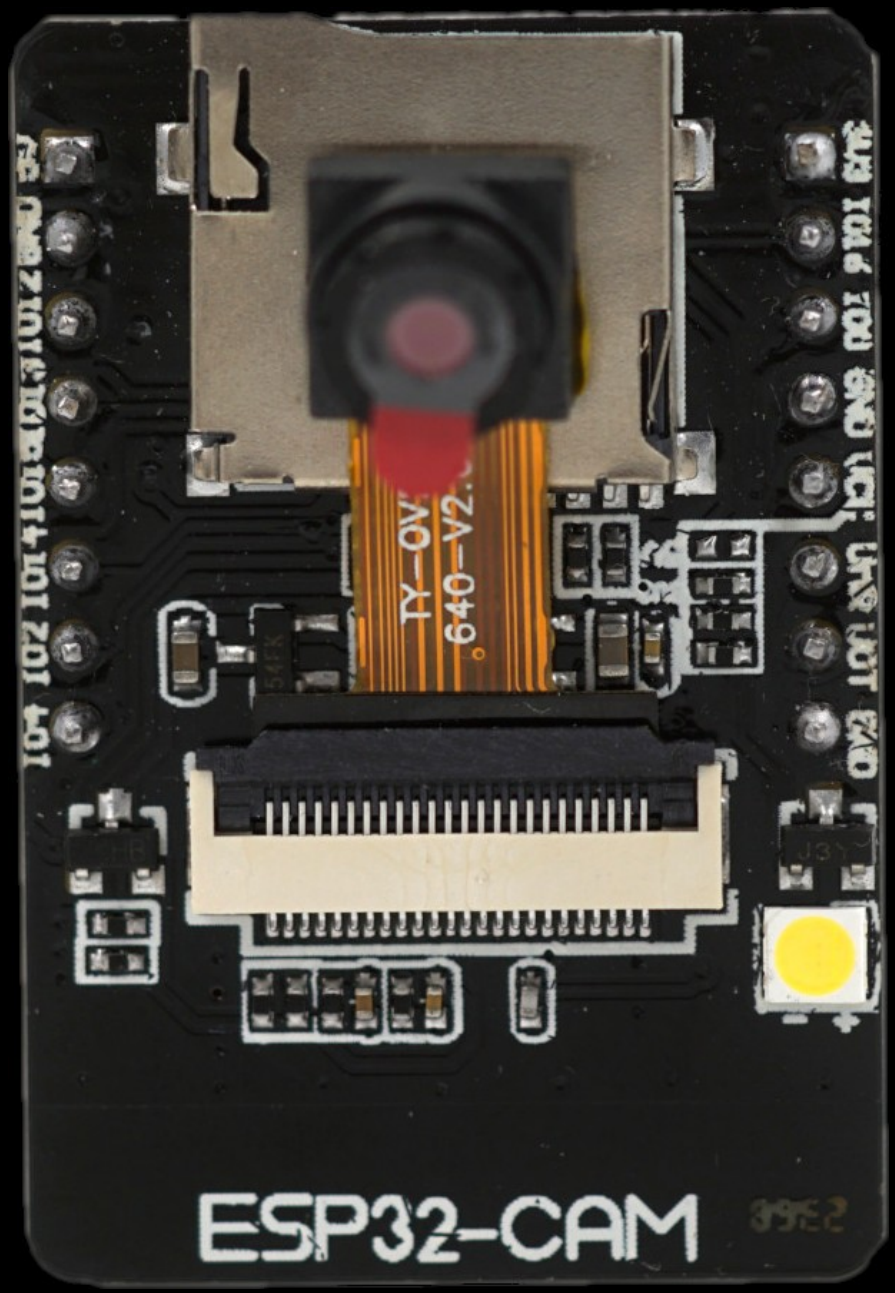

LCD
heat sink
mother board
processor
graphics chip
©2005 HowStuffWorks


Capacitive multitouch LCD screen
3.8 V, 25 Watt-hour battery
Computer board
A5
TOSHIBA


DIGIC 4


Canon
Canon
Canon
8×

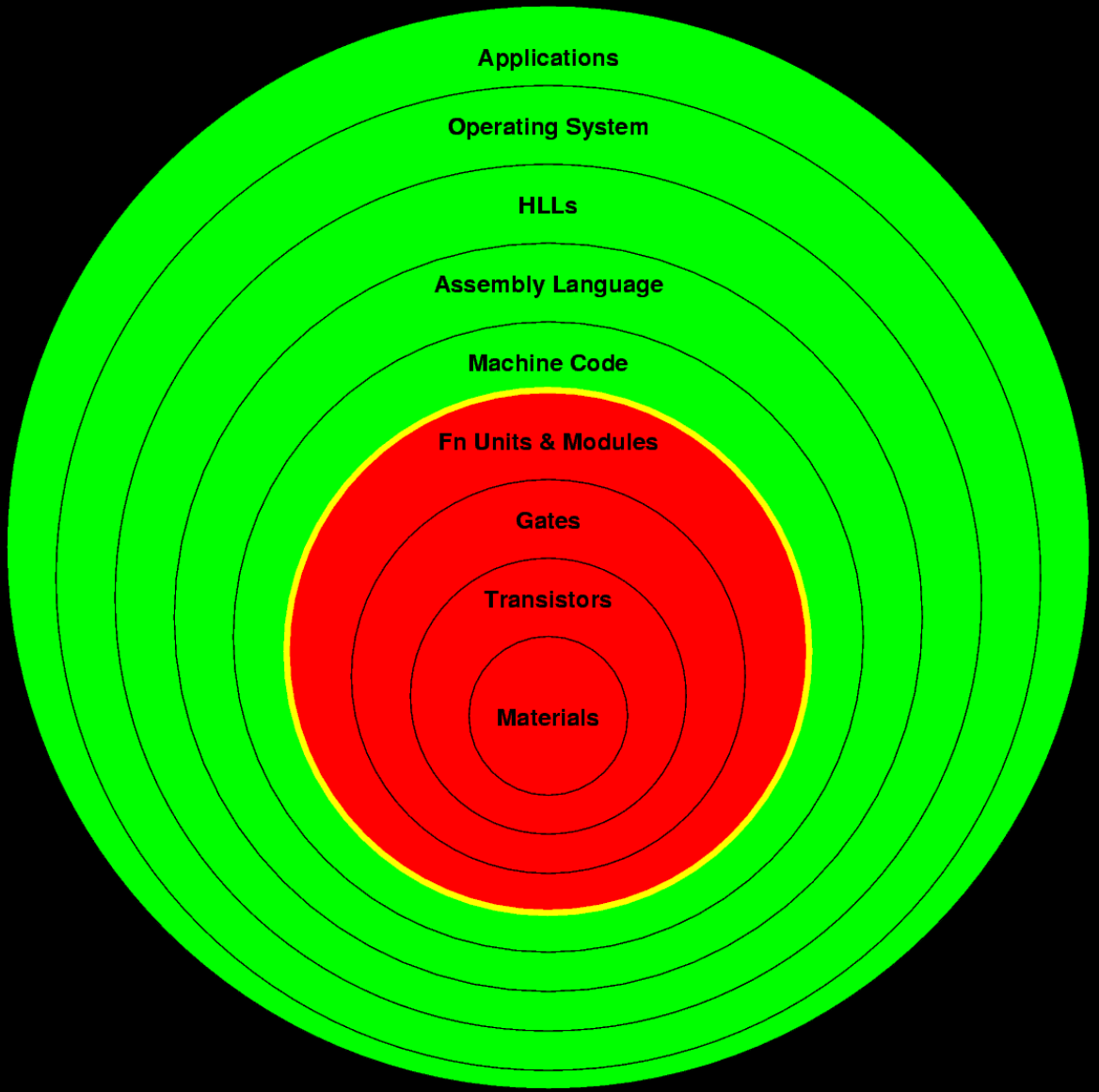

FlashAir
W-02
Wireless LAN
10
SDHC
32GB
TOSHIBA


ESP32-CAM

Complexity is Increasing!

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

Abstraction “Onion”

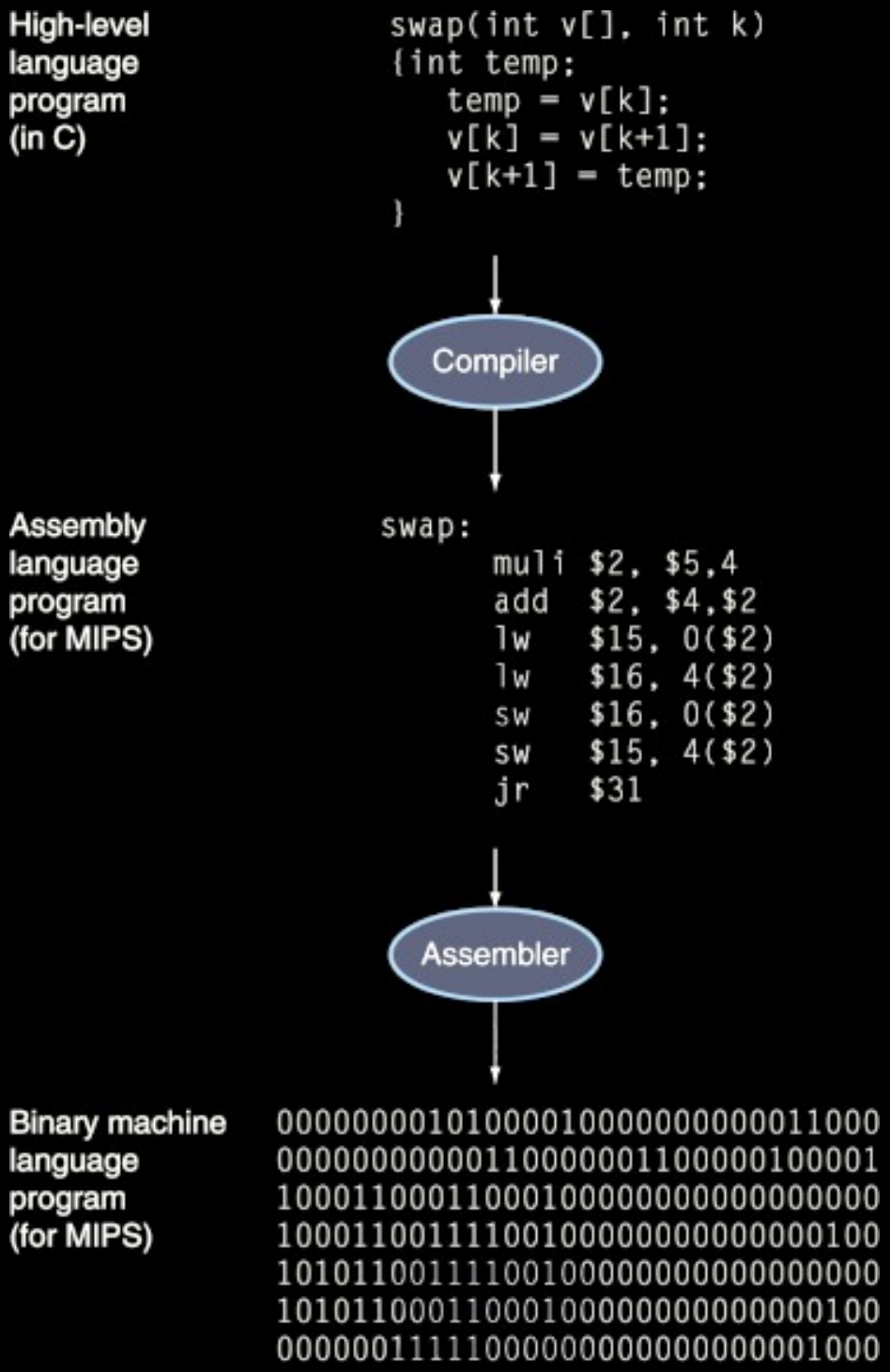


Software Layers

- 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)
00000000101000010000000000011000
00000000000011000000110000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000

Hardware Layers

- Function-block organization (CPE380 stuff)
- 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?

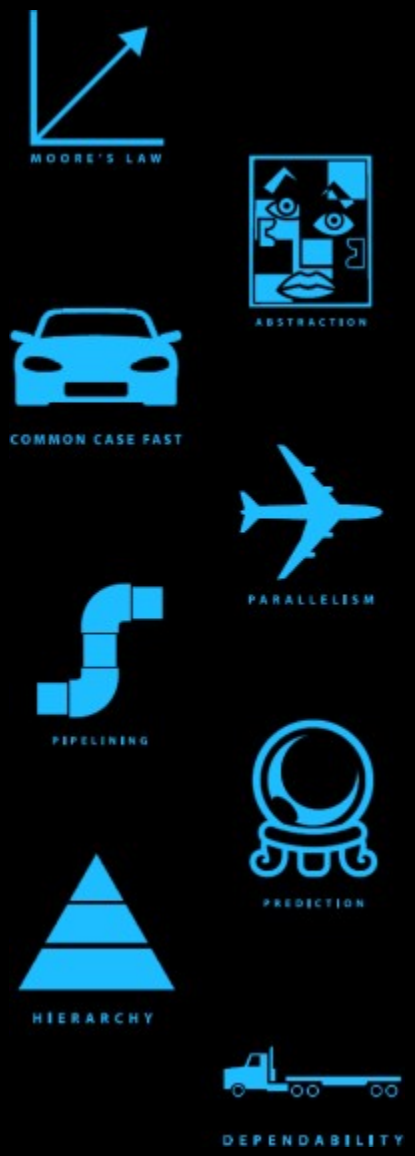
- Instruction Set Design, by *Architect*
  - Machine & Assembly Languages
  - "**Computer Architecture**"
  - **Instruction Set Architecture** / Processor
- Computer Hardware Design, by *Engineer*
  - Logic Design & Machine Implementation
  - "**Processor Architecture**"
  - "Computer Organization"

How does EE685 see this?

- A lot like CPE380/CPE480…
- Like CPE380, Verilog runs through it
- Focus is on the hardware/software interface
  - Models that guide design and use
  - Modules that enable or boost performance

8 Great Ideas

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



Main topics for EE685

- Start with Verilog
- Review pipelined RISC (where CPE380 ends)
- Add instruction-level parallelism (SWAR, VLIW, SuperScalar…)
- Add advanced processor internals (floating point, interrupts)
- Add memory heirarchy (caches, virtualization)
- Add scalable parallelism (SIMD/GPU, MIMD, networking)