Introduction

CPE380/CS380, Fall 2025

Hank Dietz

`http://aggregate.org/hankd/`

University of Kentucky

Course Overview

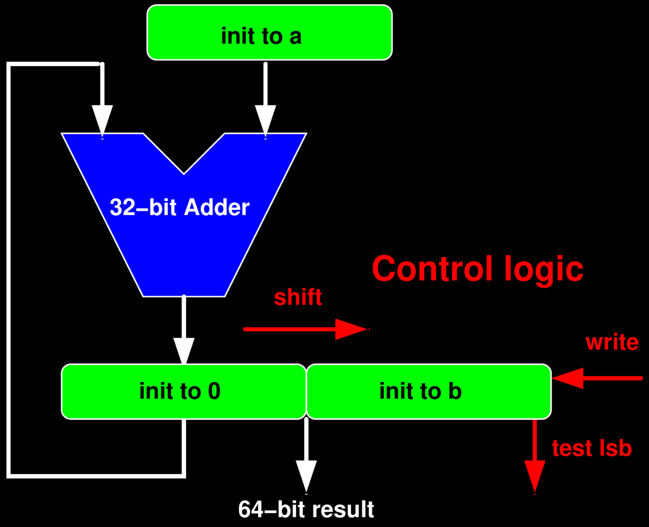
- You know how to write a simple program... from CS courses
- You know how to build simple combinatorial and sequential logic circuits from ECE courses (especially CPE282 or EE280/EE281)
- This course fills the gap between the two:
  - So you can better **specify & use** that stuff
  - So you can **create**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

Verilog 32-bit Multiplier

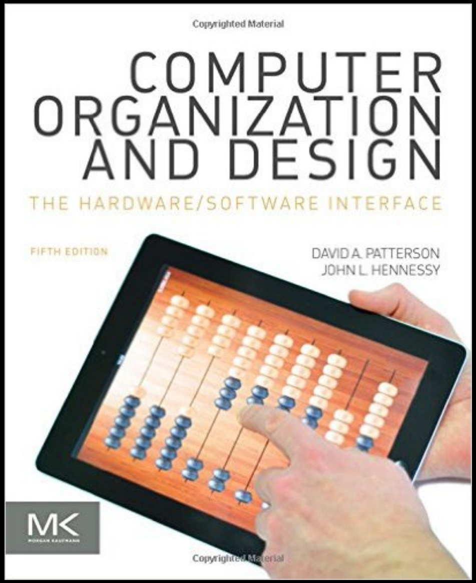
```
module mul(ready, c, a, b, reset, clk);

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, ~10%
- Three team projects, ~10% each
- Four homework assignments, ~10% each
- In-person final exam, ~20%
  (course grade limited to 1 letter above final)
- 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

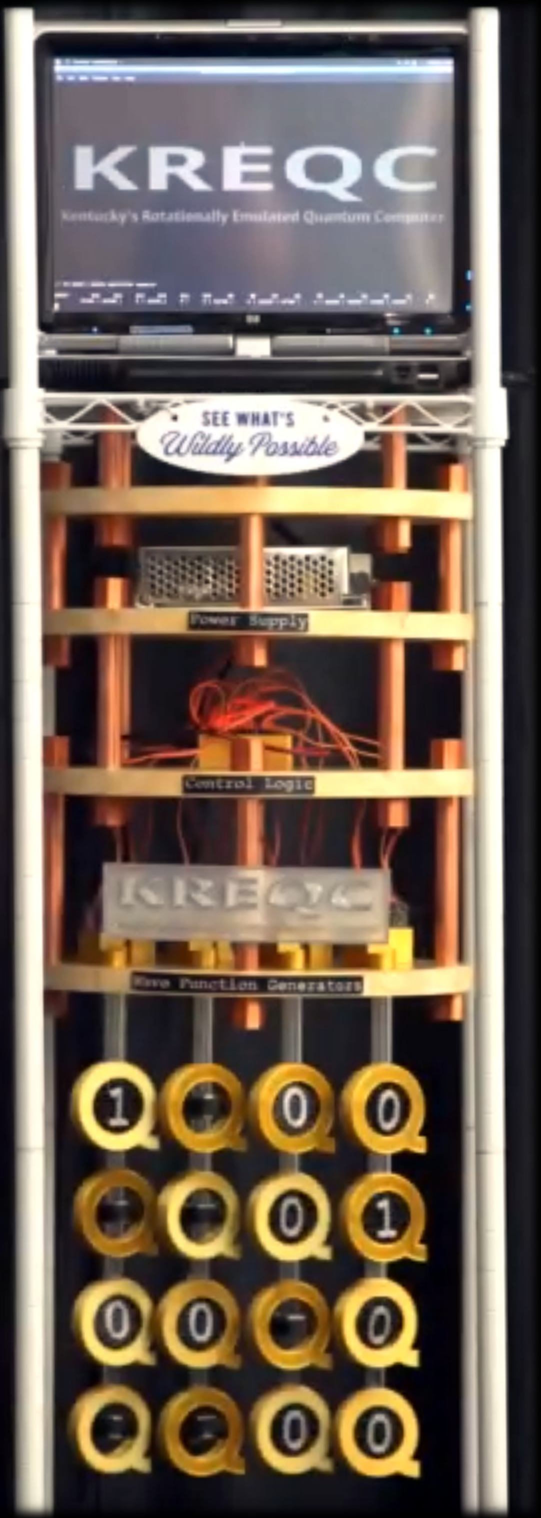
Lectures	Topic
1	Introduction
3	Verilog (individual project)
3	Multi-cycle machine (team project)
3	Machine & assembly languages (homework)
2	Single-cycle machine (team project)
3	Integer & float arithmetic (homework)
4	Pipelined machine (team project)
4	Memory hierarchy and I/O (homework)
3	Parallel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

Schedule Notes

- Projects are deliberately pushed as early as possible to reduce time pressure
- Some topics may be given more or less time depending on how students are doing
- I will be presenting research at IEEE/ACM SC (Supercomputing) conference, so we will not have regular class meetings **11/18** & **11/20**

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**!

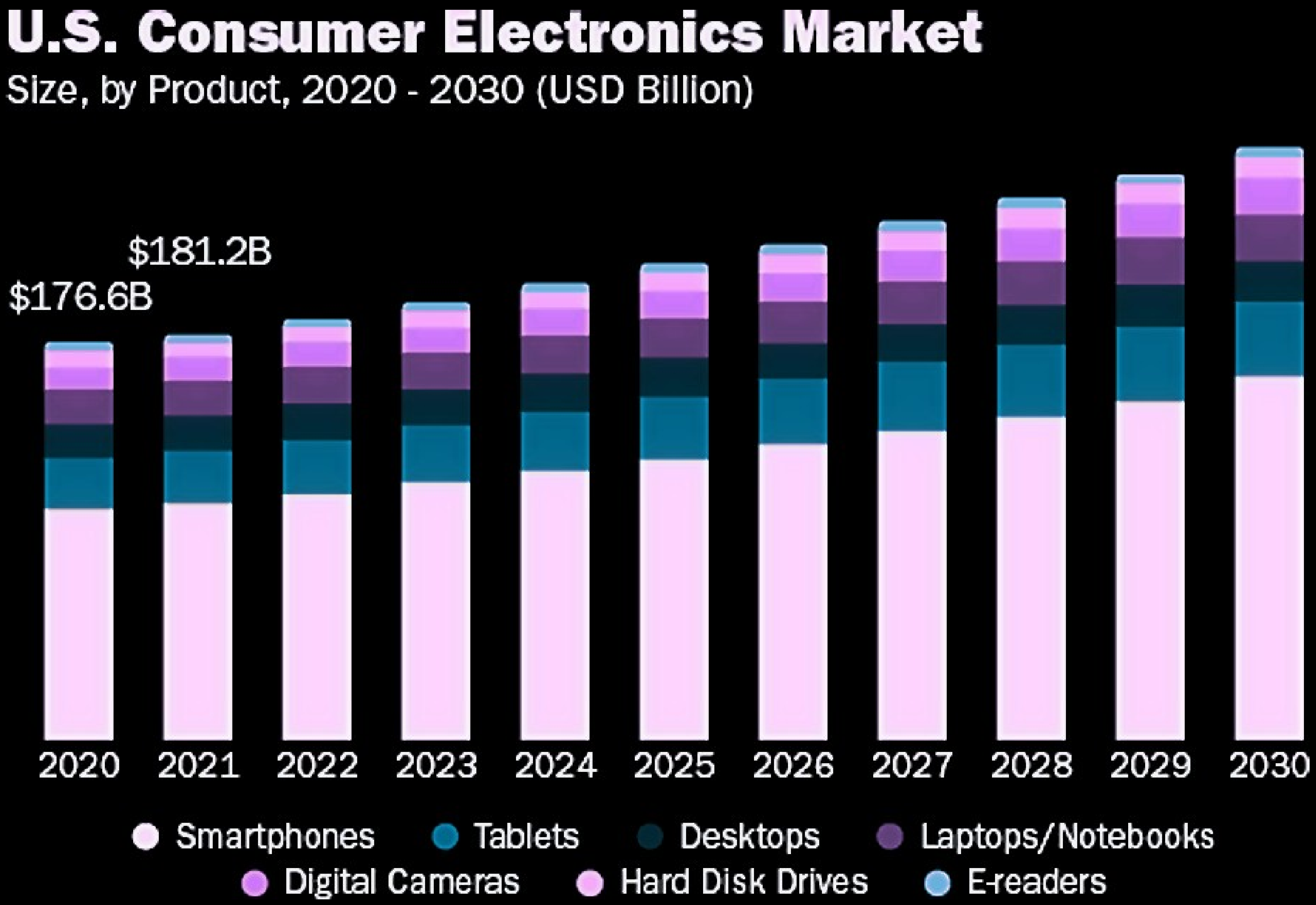

KREQC
Kentucky's Rotationally Emulated Quantum Computer
SEE WHAT'S Wildly Possible
Power Supply
Control Logic
KREQC
Wave Function Generators

College of Engineering
Electrical & Computer Engineering

nak
UK ECE
nak

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)

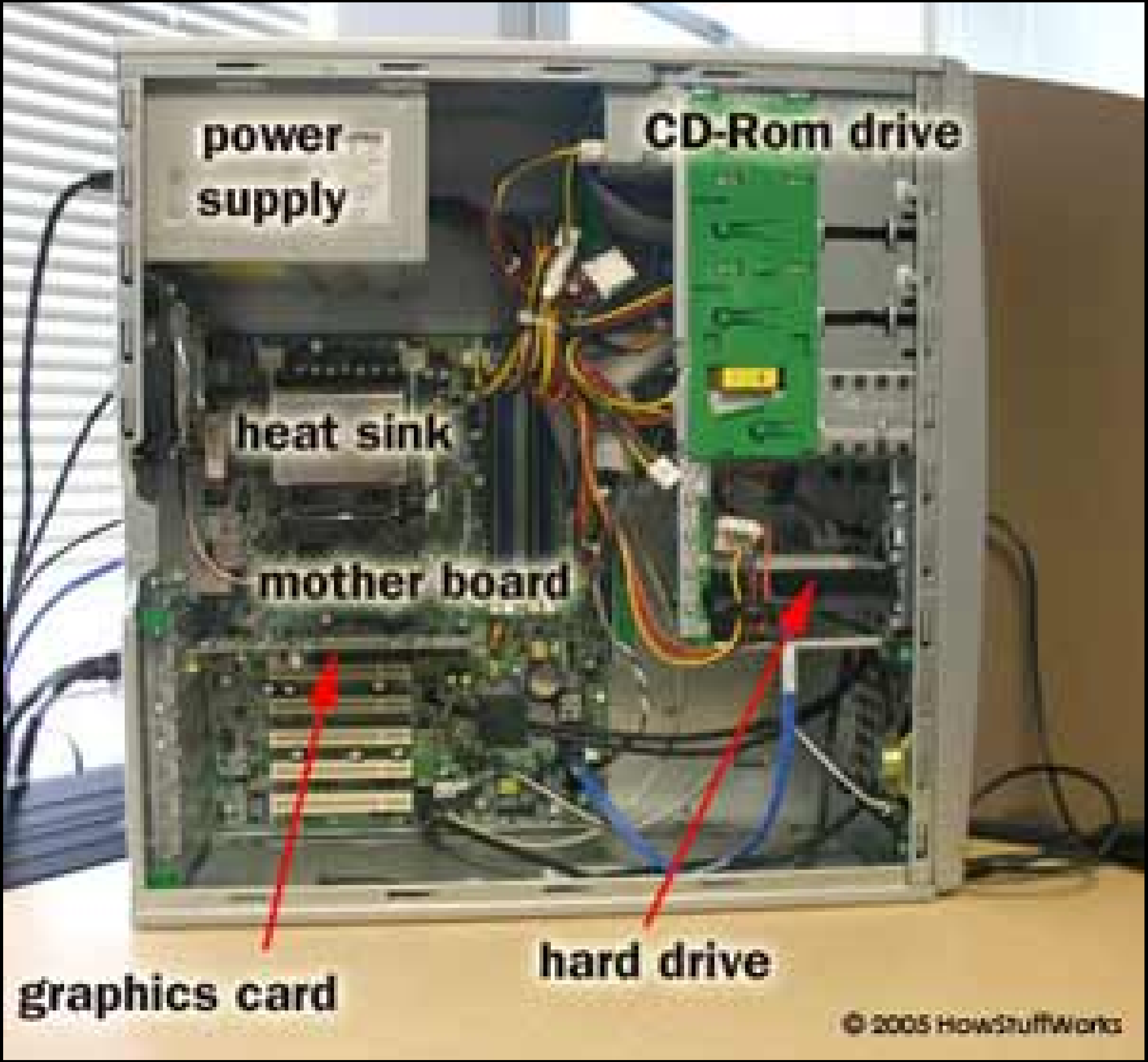


U.S. Consumer Electronics Market

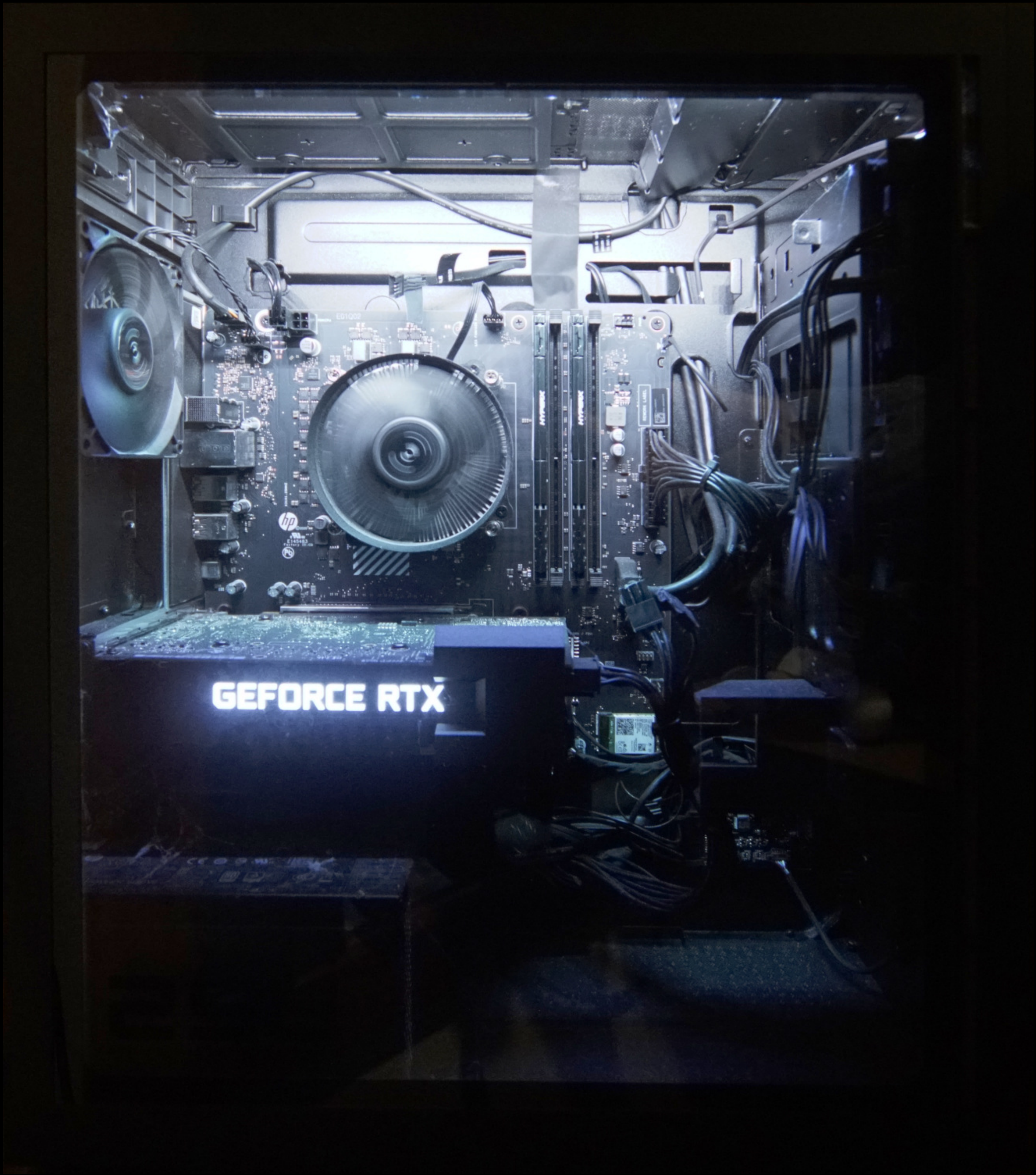
Size, by Product, 2020 - 2030 (USD Billion)

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

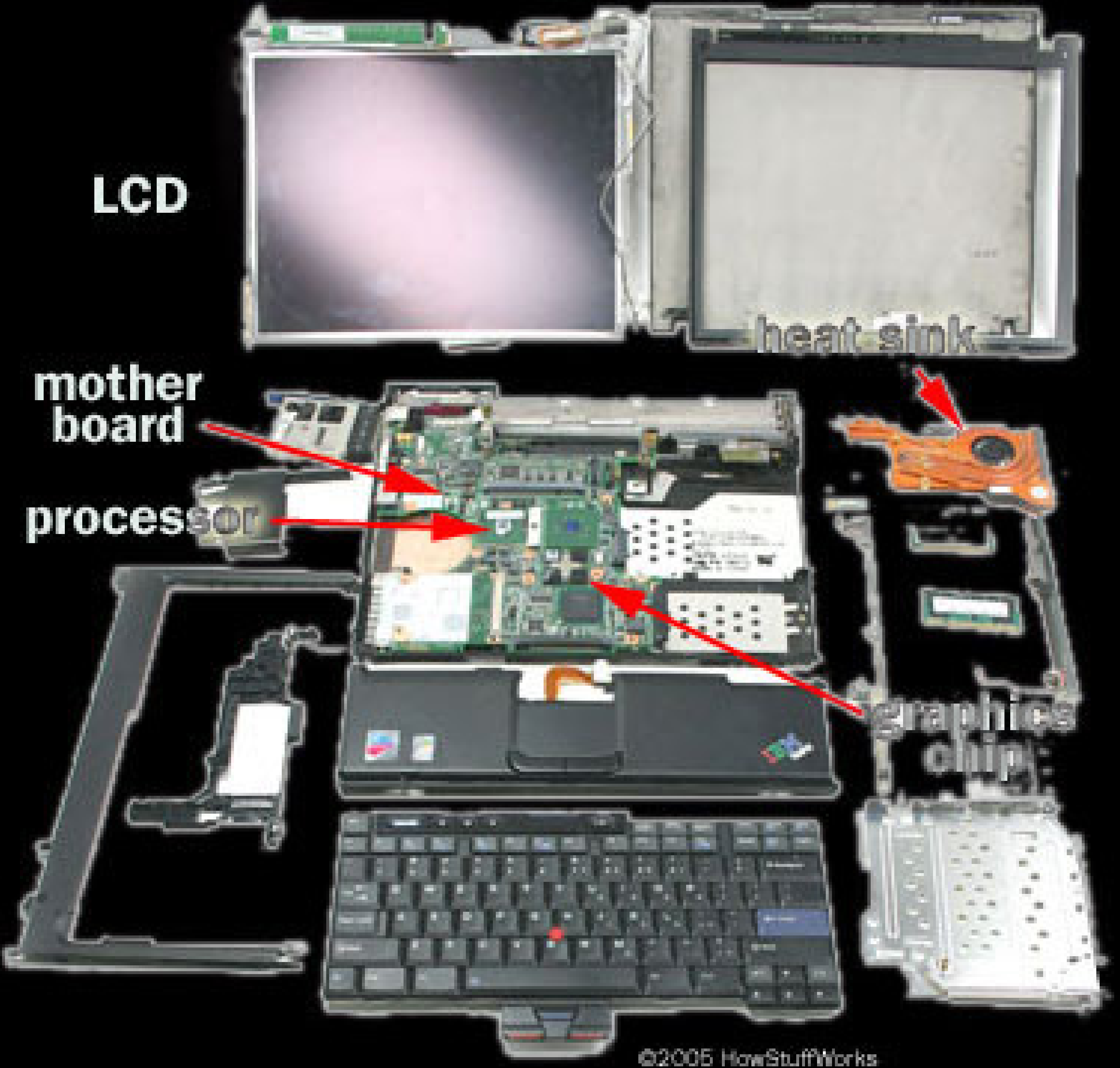
What's Inside?

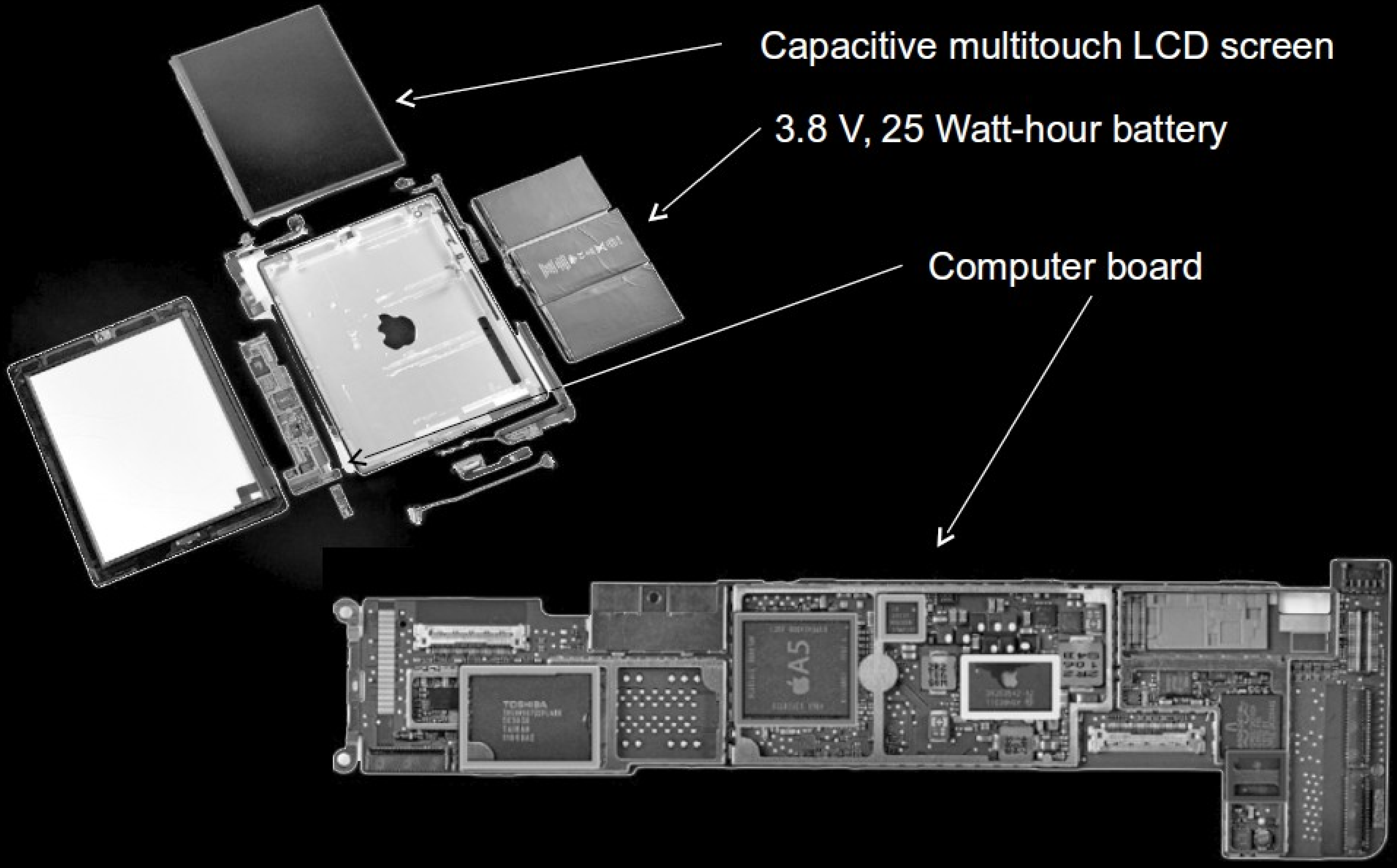


© 2005 HowStuffWorks



GEFORCE RTX


LCD
heat sink
mother board
processor
graphics chip
©2005 HowStuffWorks



Capacitive multitouch LCD screen

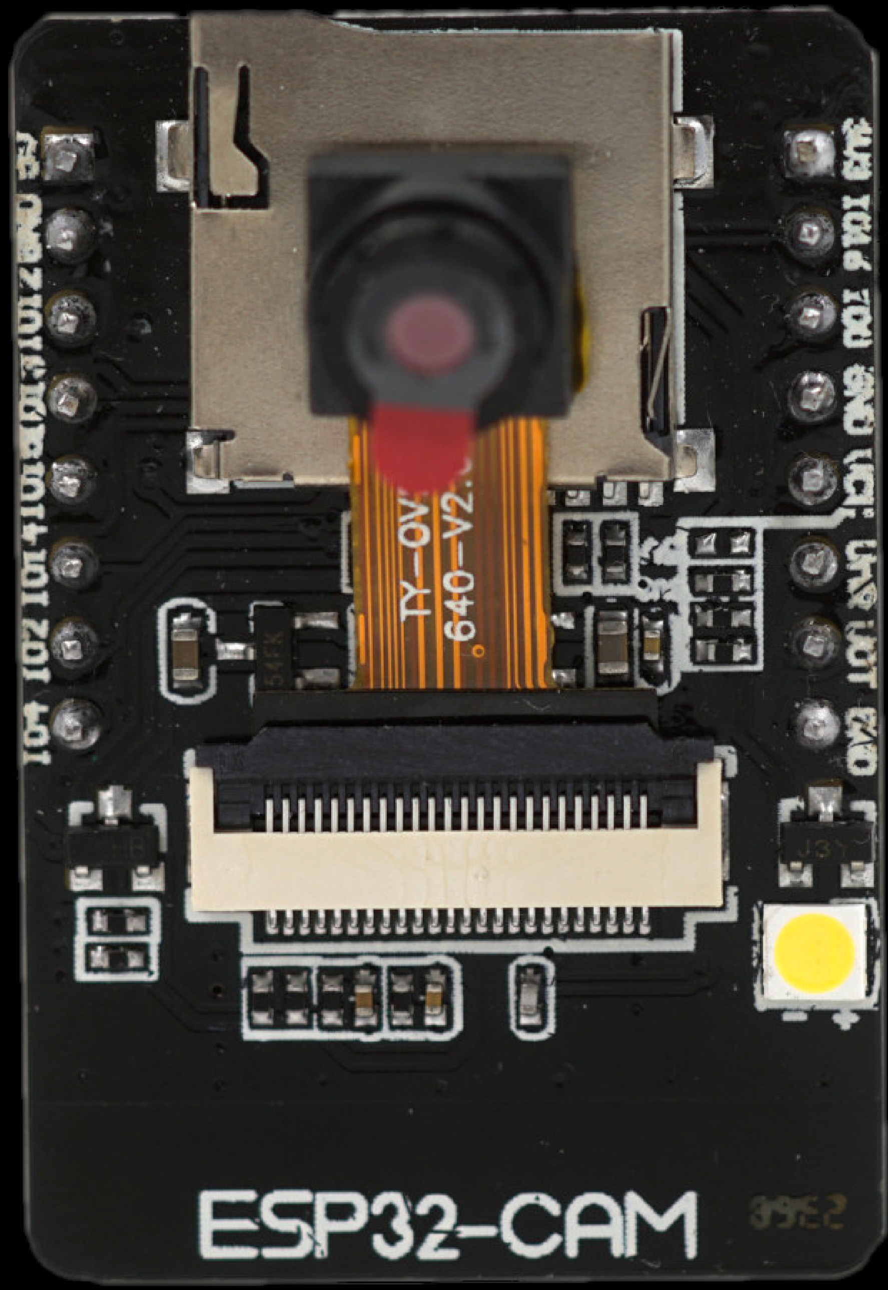
3.8 V, 25 Watt-hour battery

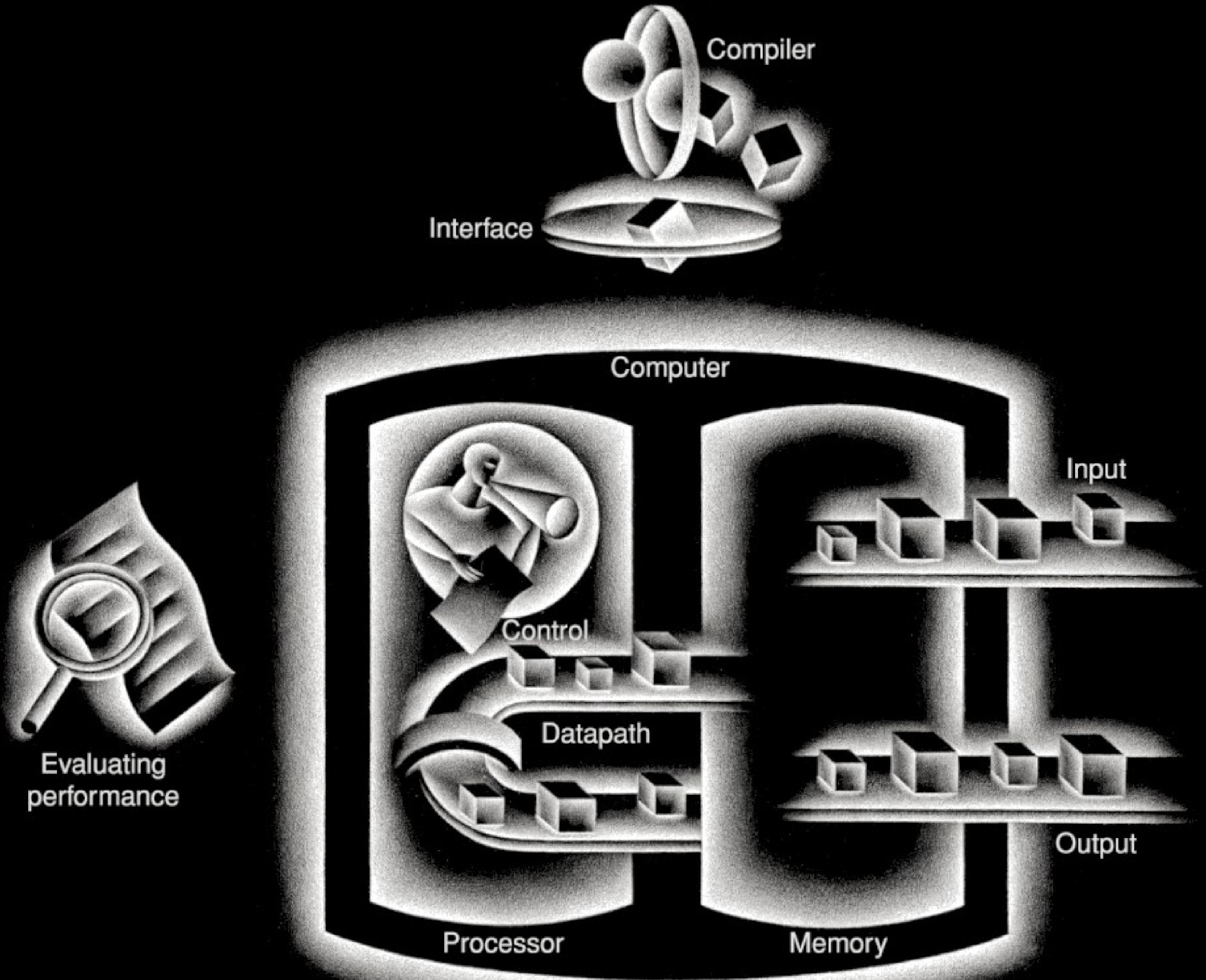
Computer board

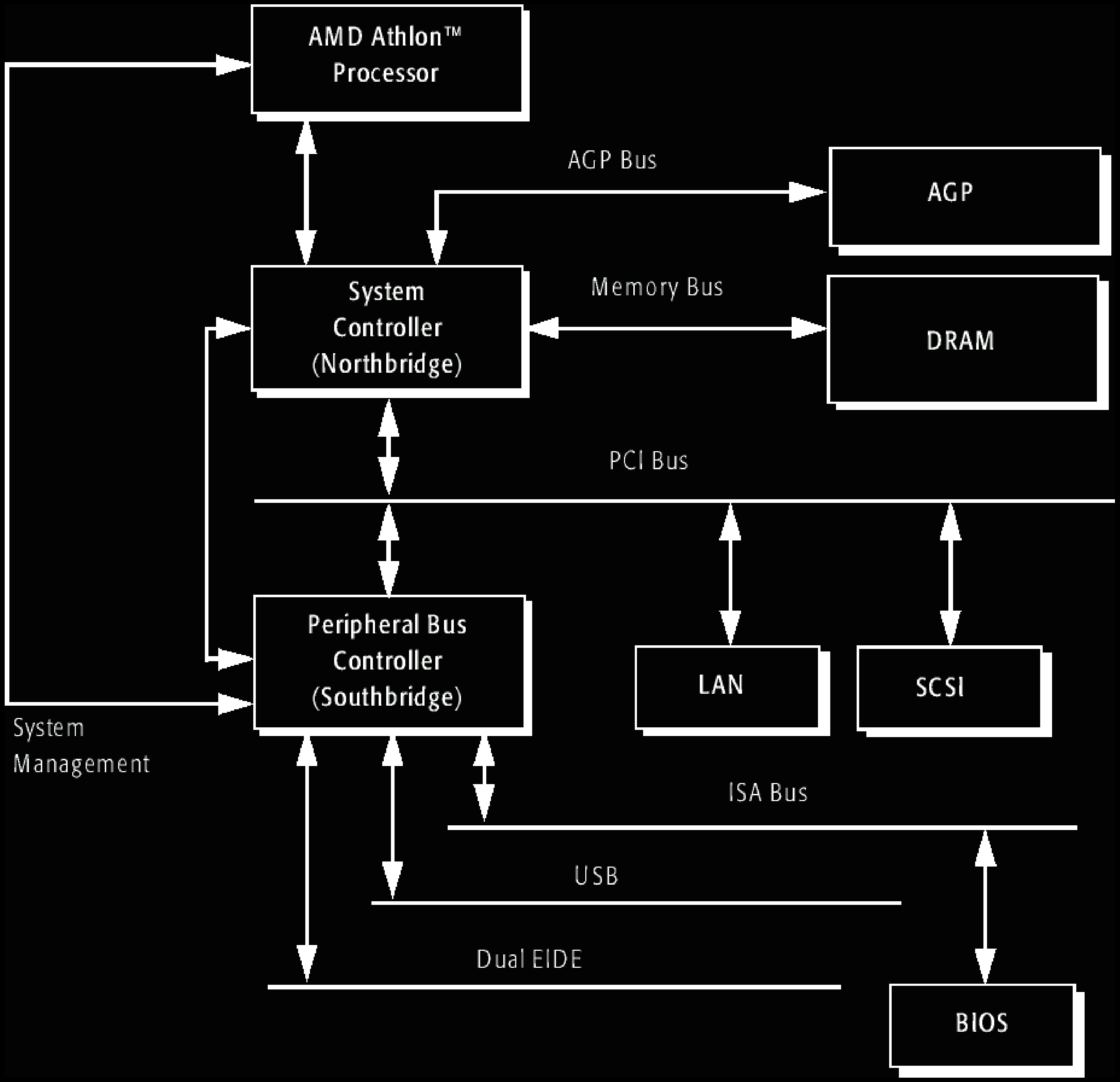
hynix
CANON
DiGiC4
CARD/BATT. OPEN

Canon
BATTERY CHARGER
CHARGE
FULL
Canon
8x
CANON ZOOM LENS 8x IS
Canon

FlashAir
W-02
Wireless LAN
32GB
TOSHIBA


ESP32-CAM


Compiler
Interface
Computer
Input
Control
Evaluating performance
Datapath
Output
Processor
Memory


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

Processor Terminology

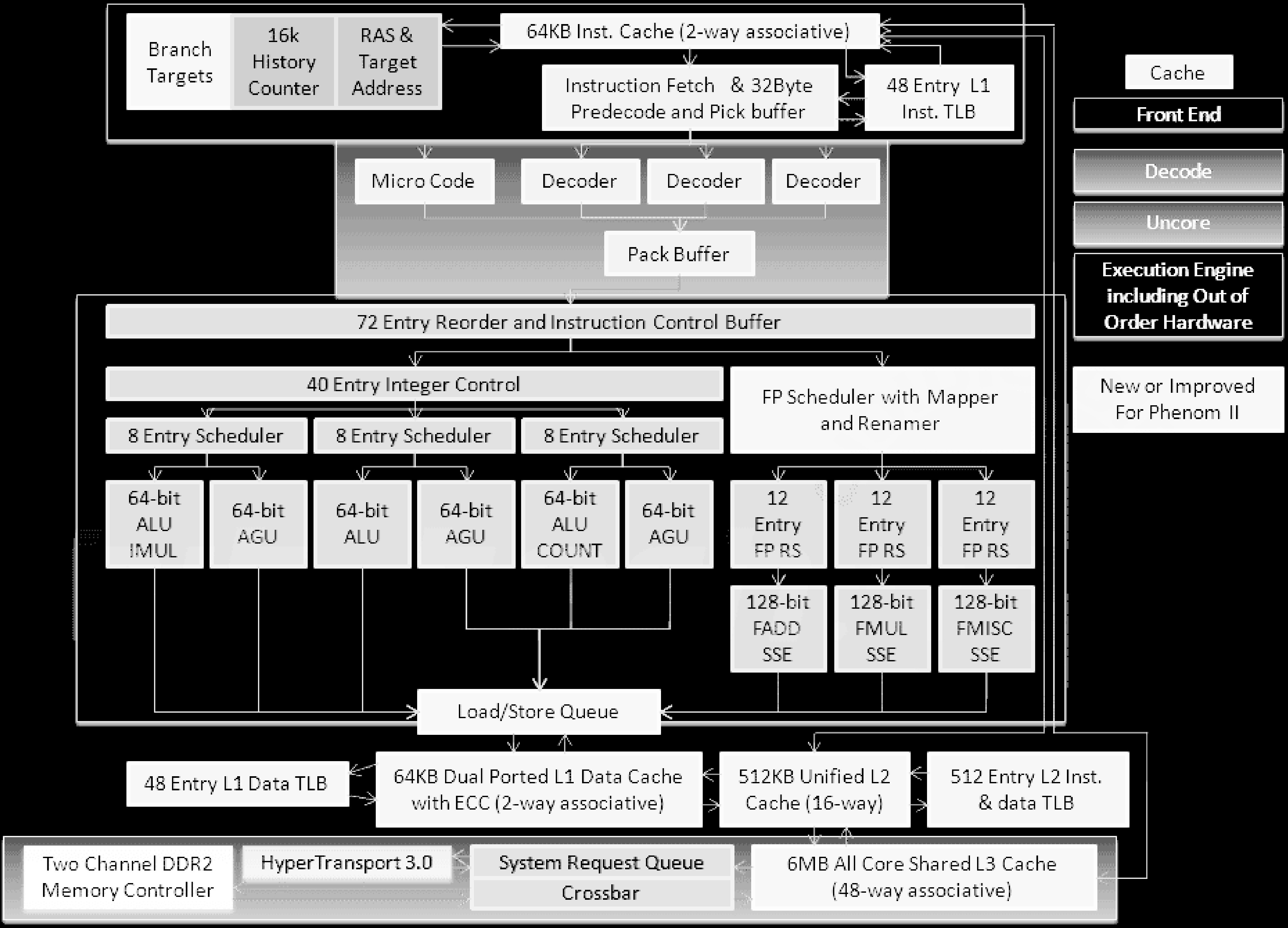
- CPU – Central Processing Unit
- PE, Core – Processing Element
- Processor – CPU or chip containing PEs
- “Computer Family” – same ISA
- x86, IA32, x64/AMD64 – Intel 386-based ISAs
- MIPS, ARM, SPARC – other common ISAs
- DSP – Digital Signal Processor
- GPU – Graphics Processing Unit
- Tensor – Matrix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

Lawrence Livermore National Laboratory
U.S. DEPARTMENT OF ENERGY
NNSA
Hewlett Packard Enterprise
AMD
Red Hat
EL CAPITAN
ADVANCED SIMULATION & COMPUTING
HPE Cray
Mike Carter Photography

El Capitan supercomputer:
11,039,616 cores, 2.746 Exaflop/s
Cost approx. $600M, 29.6 MW power



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

FP Scheduler with Mapper and Renamer

8 Entry Scheduler

8 Entry Scheduler

8 Entry Scheduler

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

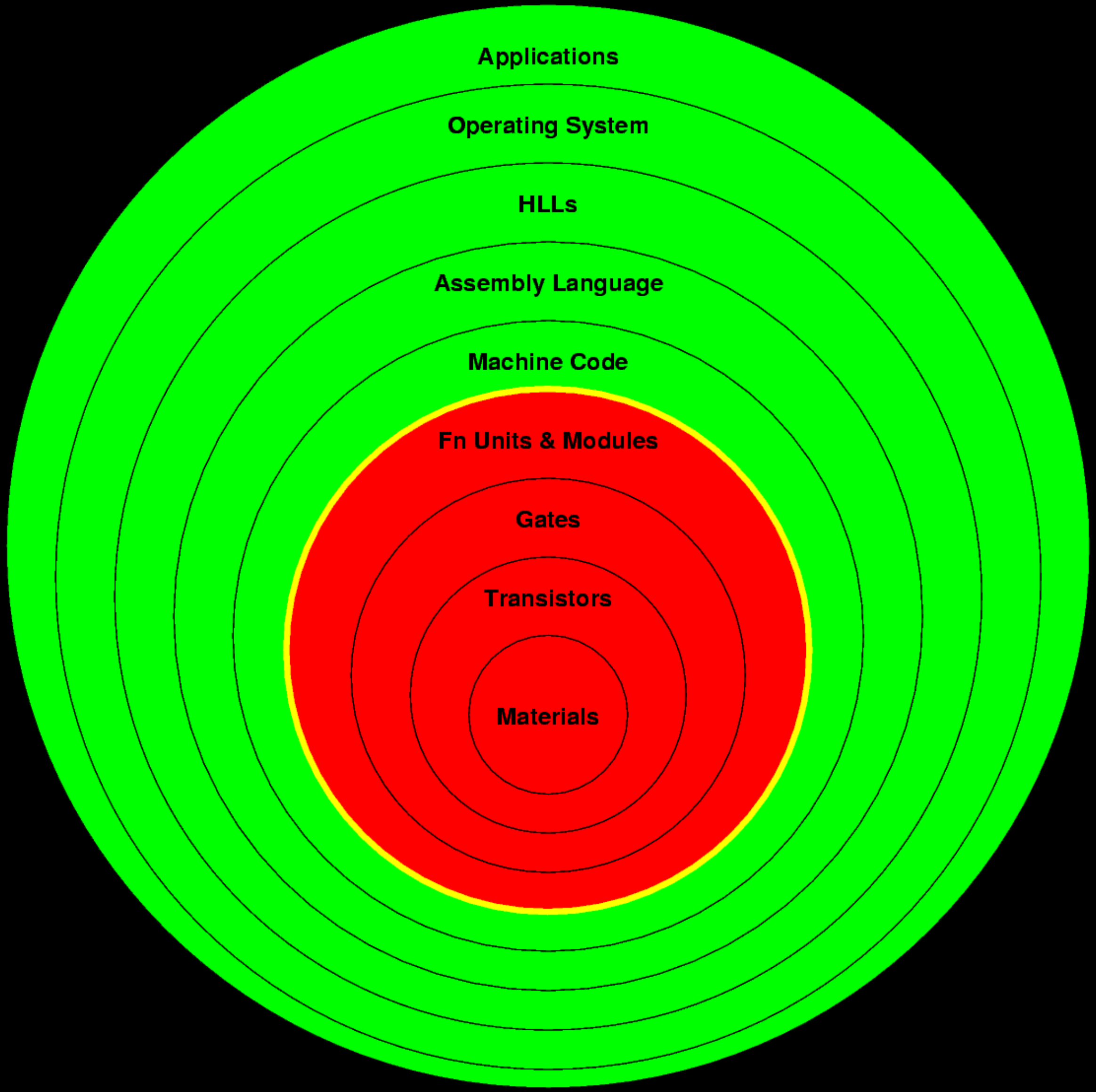
Decode

Uncore

Execution Engine including Out of Order Hardware

New or Improved For Phenom II

Abstraction "Onion"

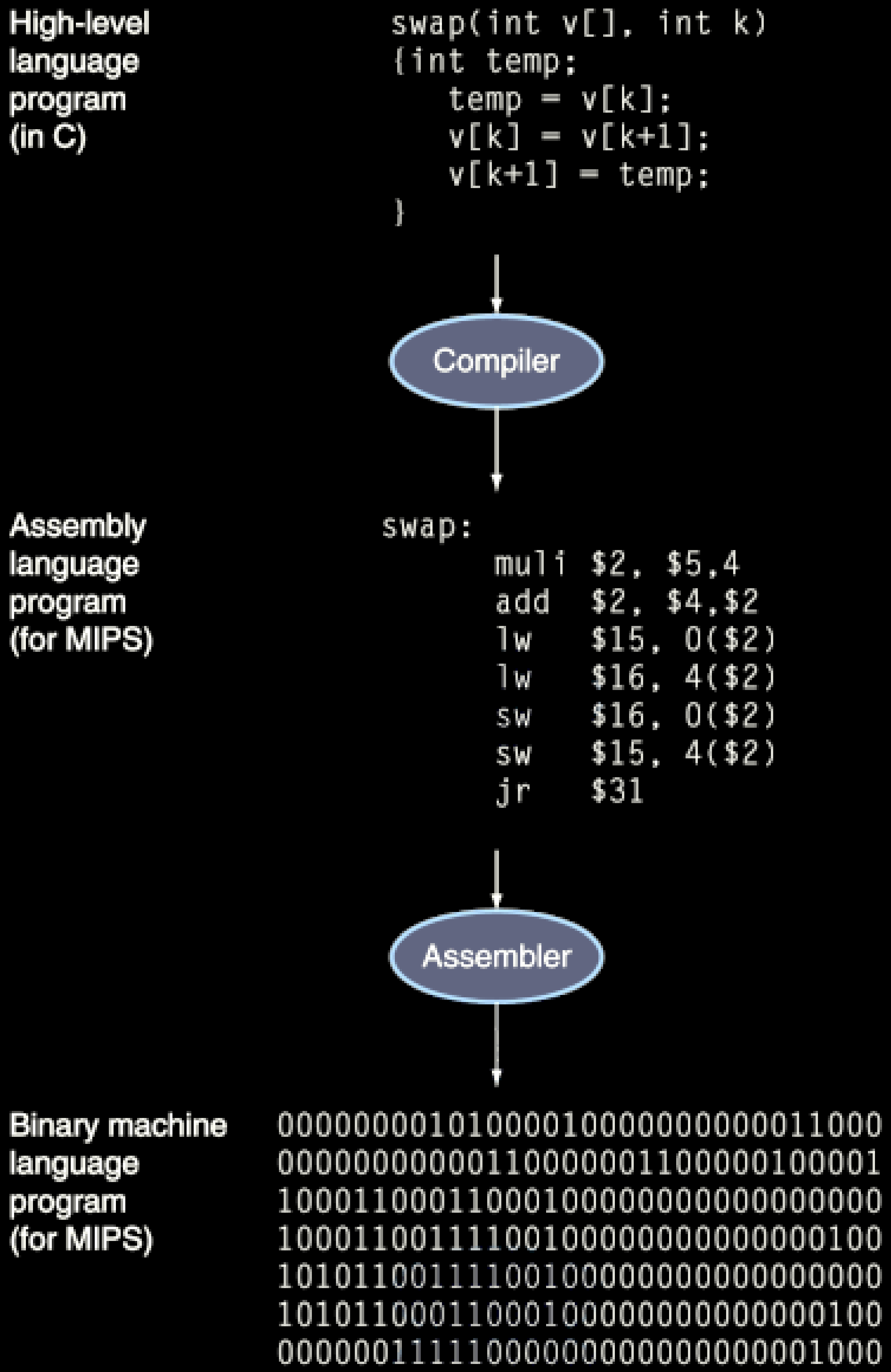


Software Layers

- 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;
}
```

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
```

Binary machine
language
program
(for MIPS)

```
00000000101000010000000000011000
00000000000110000001100000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000
```

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?

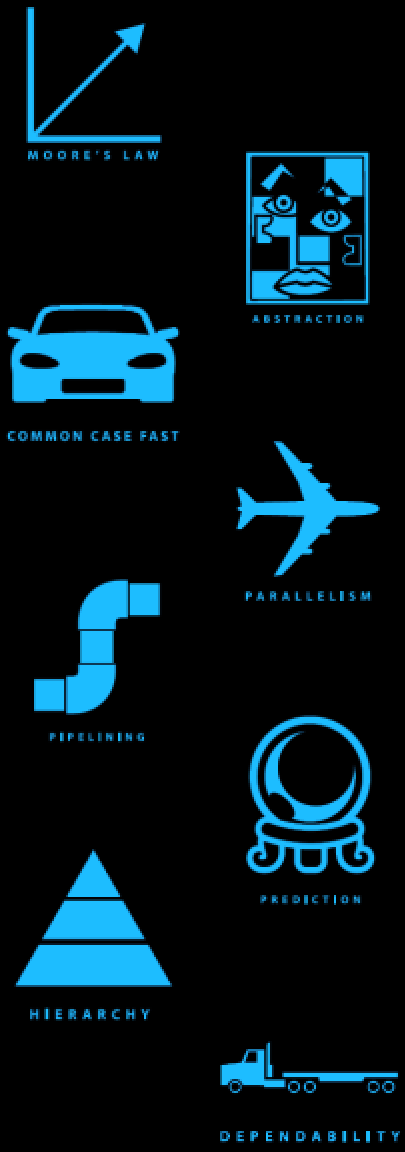
- Instruction Set Design, by *Architect*
  - Machine & Assembly Languages
  - “**Computer Architecture**”
  - **Instruction Set Architecture** / Processor
- Computer Hardware Design, by *Engineer*
  - Logic Design & Machine Implementation
  - “**Processor Architecture**”
  - “Computer Organization”

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***...