



Lecture 1: Computer Architecture

Jangwoo Kim (SNU) : jangwoo@snu.ac.kr

and many other professors (see below!)

Slides developed in part by Profs. Austin, Brehob, Falsafi, Hill, Hoe, Lipasti, Martin, Roth, Shen, Smith, Sohi, Tyson, Vijaykumar, and Wenisch @ Carnegie Mellon University, University of Michigan, Purdue University, University of Pennsylvania, University of Wisconsin, POSTECH and SNU.



Key Topics

- ◆ What is a computer?
(or what makes a computer?)
- ◆ How a computer works?
- ◆ How to make a computer?
- ◆ What should we know about computer?

*You should be able to answer these questions
at the end of this semester.*



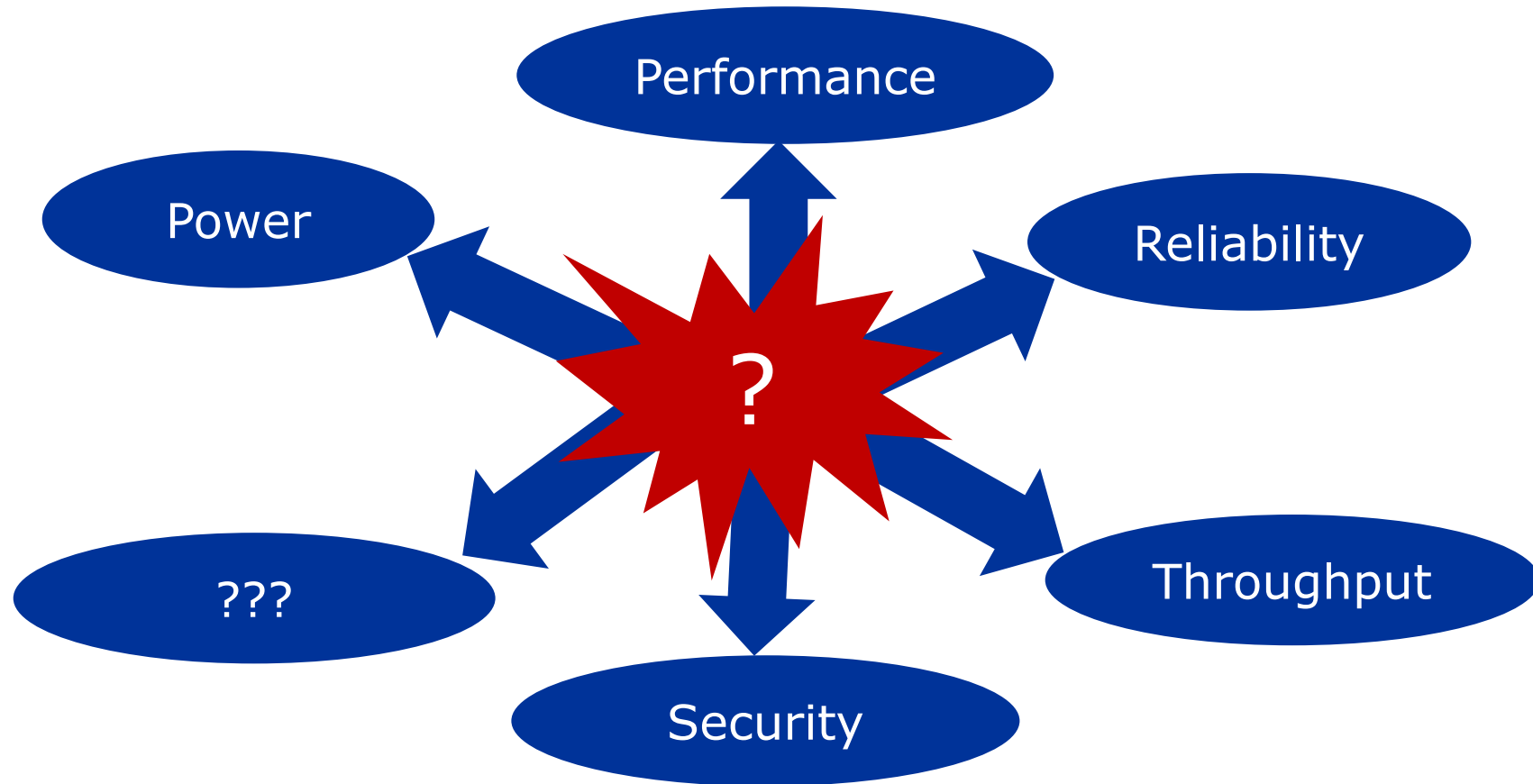
Say you are a CPU architect @ Intel.

What should you really do?





Design issues to be considered



Very difficult to find an optimal design point



Performance: how to make it fast?

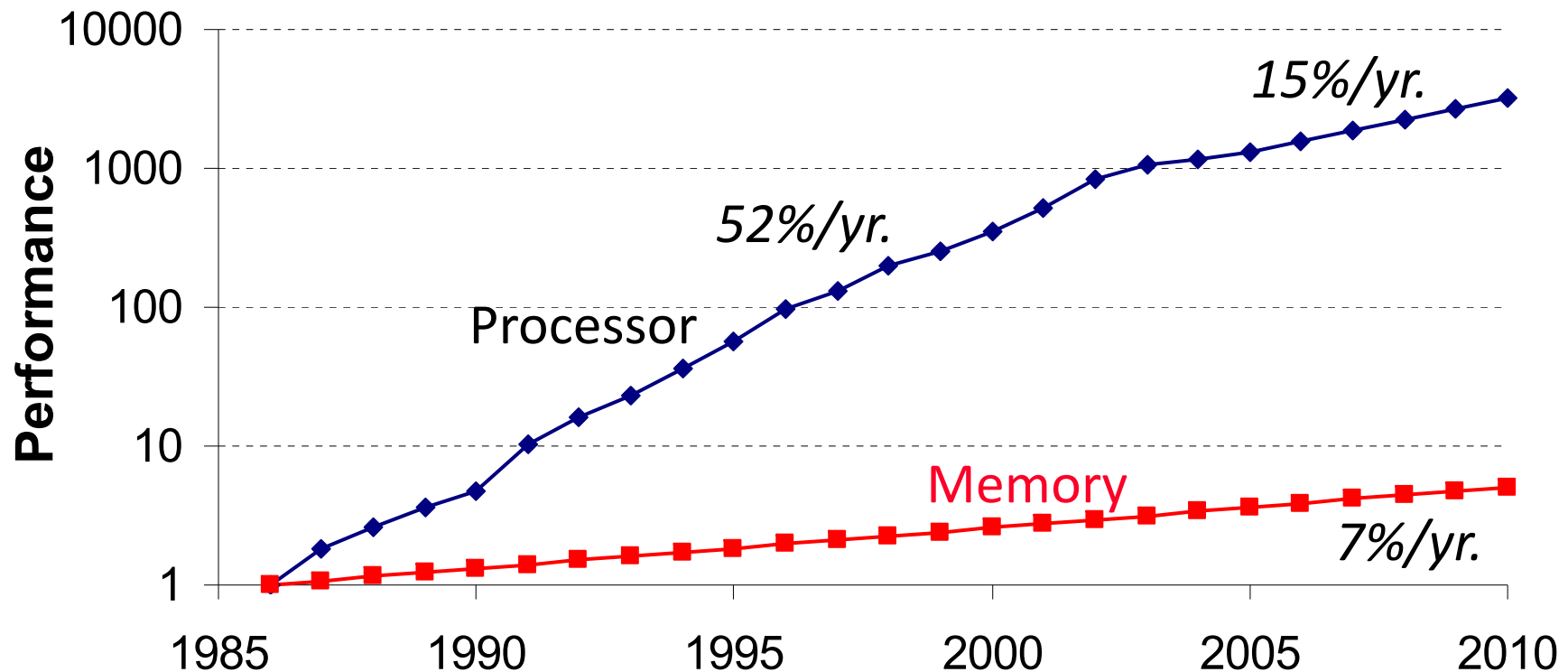
◆ Performance law

$$Latency = (\# \text{ cores}) * (1 / \text{frequency}) * (\# \text{ of instructions}) * CPI$$

- # of cores ← smaller circuit, smaller cache
 - single core, two cores, etc.
- Frequency ← deep pipelining, high voltage
 - 1Ghz, 2Ghz, etc.
- # of 'dynamic' instructions ← good ISA, good compiler
 - How many instructions to execute for this program?
- CPI or IPC ($= 1 / CPI$) ← more calculators, more logic
 - clock per instruction: how many clocks to execute one instruction?



Memory: getting relatively slower

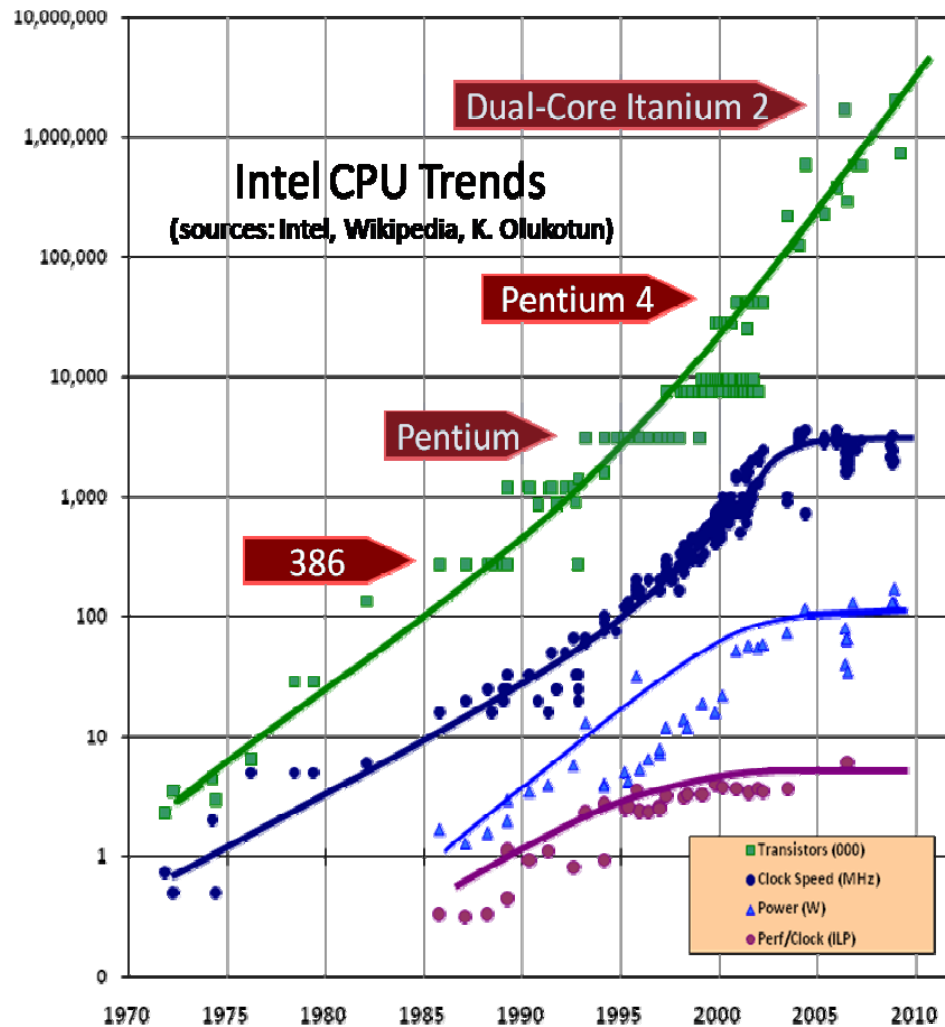


Source: Hennessy & Patterson, Computer Architecture: A Quantitative Approach, 4th ed.

Today: 1 mem access \approx 500 arithmetic ops
“Von Newman computer bottleneck”



Power: computers can burn out



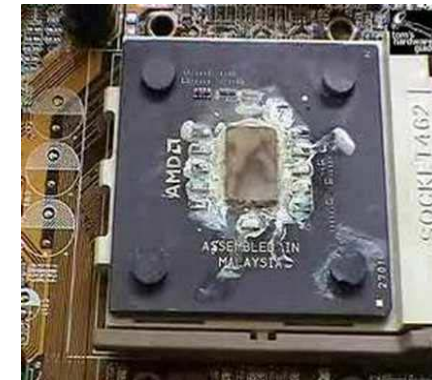
What is the use
of extra transistors?

of transistors per chip

Clock speed

Power

Perf. per clock

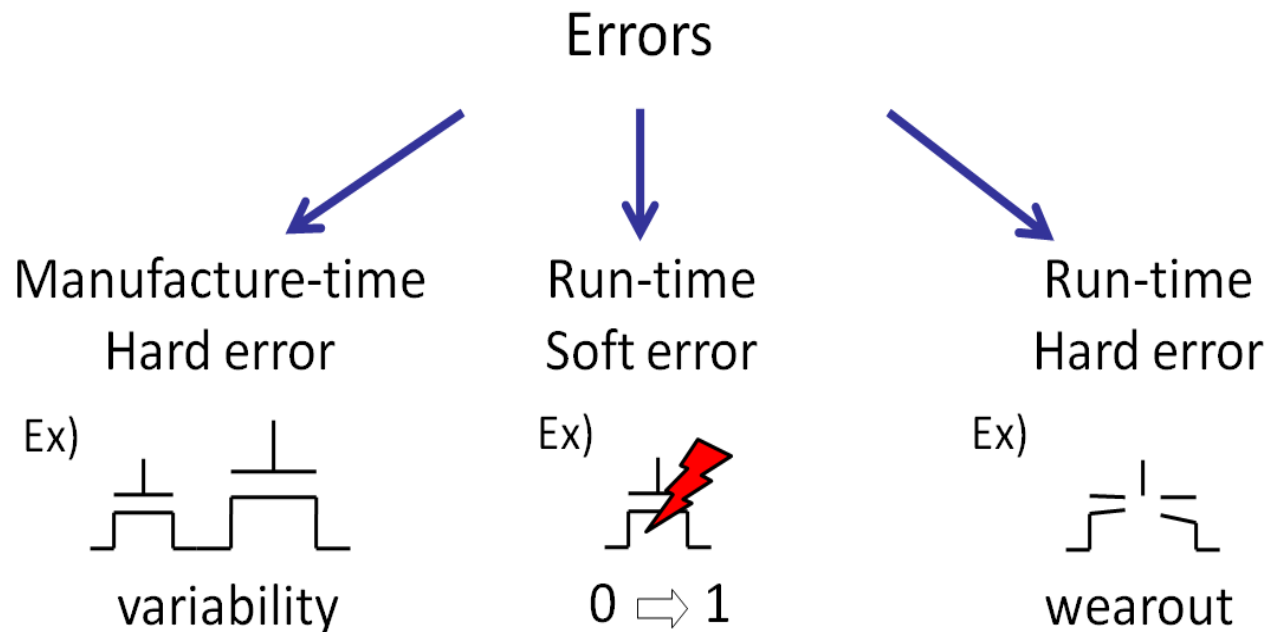


can't build a faster CPU
due to the power ceiling



Reliability: computers can fail

- ◆ Computer gets broken quite often (e.g., $1+1 \rightarrow 1000$)



Your computers are fixing these errors for you.
You just don't see it.

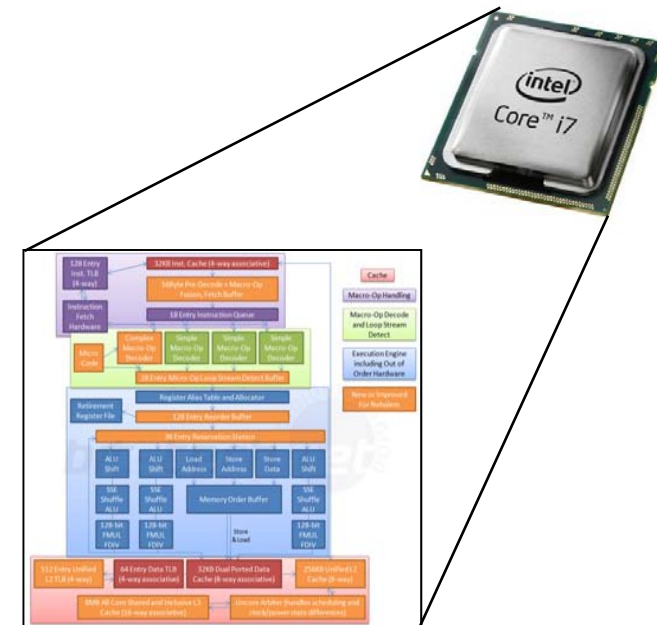
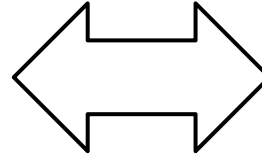


Modeling: what is the best design?

◆ Simulation is the KEY.

Simulators are S/Ws written in C/C+, etc for

- Design exploration
- Design modeling
- Design evaluation
 - Performance
 - Power
 - Reliability
 - ...
- Design feedback



No simulation → No new system

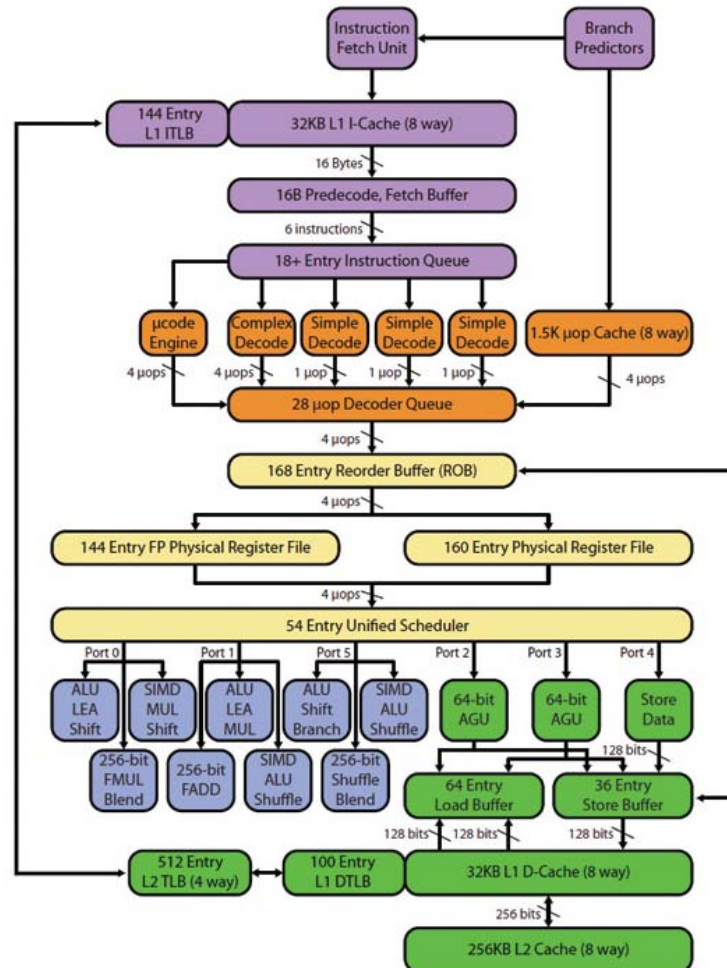


Computer architects should make
“all these” difficult decisions!

If you do everything well, ...



Architect's new CPU design!

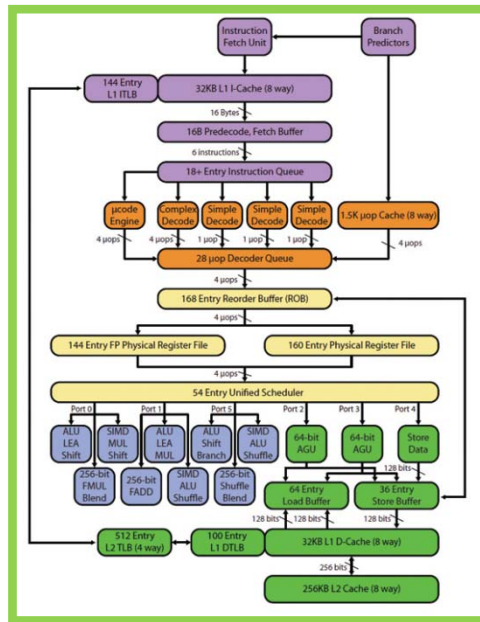


- ◆ Core pipeline
 - CISC→RISC translation
 - Hyper-threading (a.k.a. SMT)
 - Branch prediction
 - Out-of-order execution
 - 168 Entry ROB
 - 144~160 Registers
 - 54 Entry Issue Queue
- ◆ Multi-core/multi-thread
 - 2 thread per core
 - 4-6 cores per chip
- ◆ Cache
 - 32KB L1 i-cache
 - 32KB L1 d-cache
 - 256KB L2 cache
 - 8-12MB L3 cache

This is 'micro-architecture' of Intel Sandy Bridge CPU.



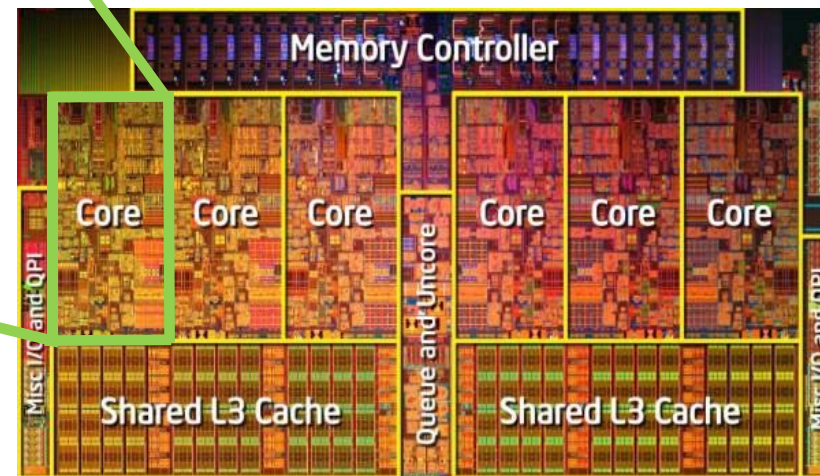
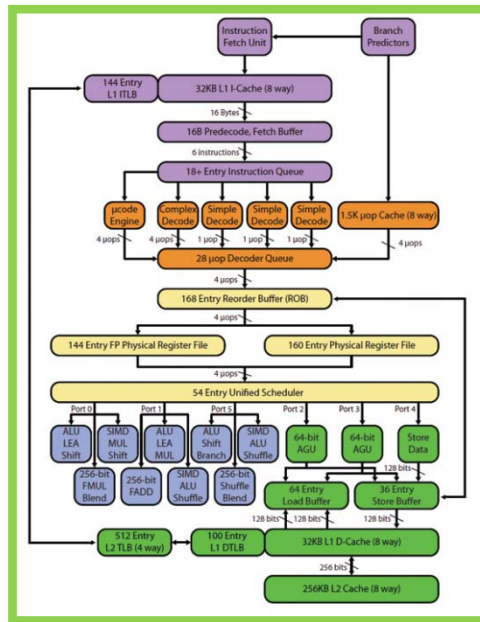
Let's put the design on the chip



Simulation (in C/C++)



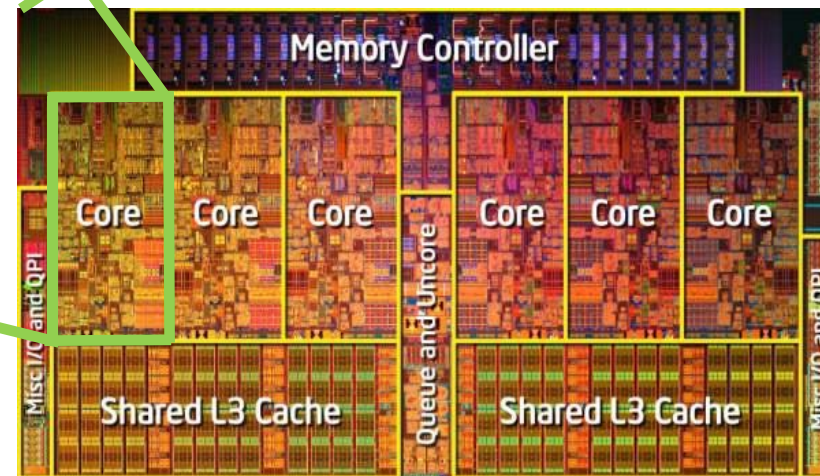
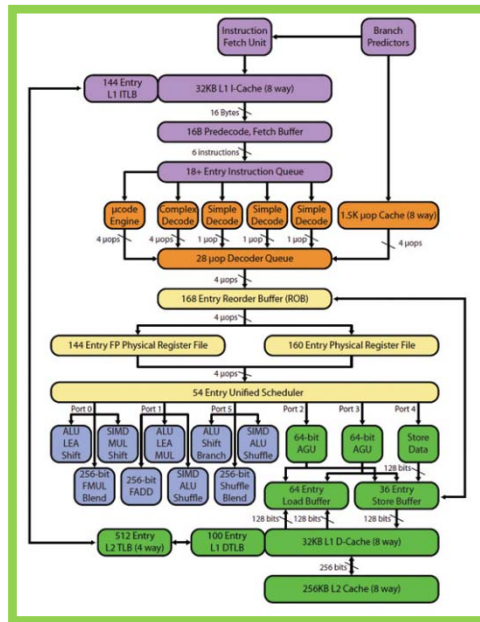
Let's put the design on the chip



Simulation (in Verilog)



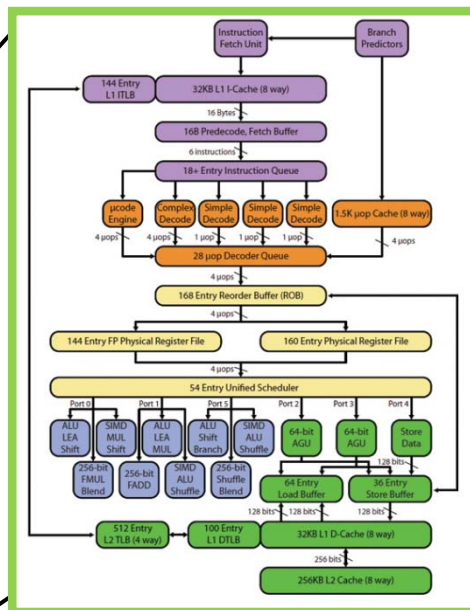
Let's put the design on the chip



C/C++ → Verilog → Packaging



Intel Sandy Bridge CPU



- ◆ Intel i7 Nehalem
- ◆ 12 threads on 6 cores
- ◆ 32 KB L1 i-cache, d-cache,
- ◆ 256KB L2 cache
- ◆ 8-12MB L3 cache

Finally, you have a new CPU.



Architects are the best engineers

- ◆ They know “everything”
 - Application
 - Operating System
 - HW Architecture
 - VLSI circuits

- ◆ They can do “everything”
 - Assembly programming
 - C/C++ programming
 - Verilog programing

As a result,
every company wants to hire computer architects.

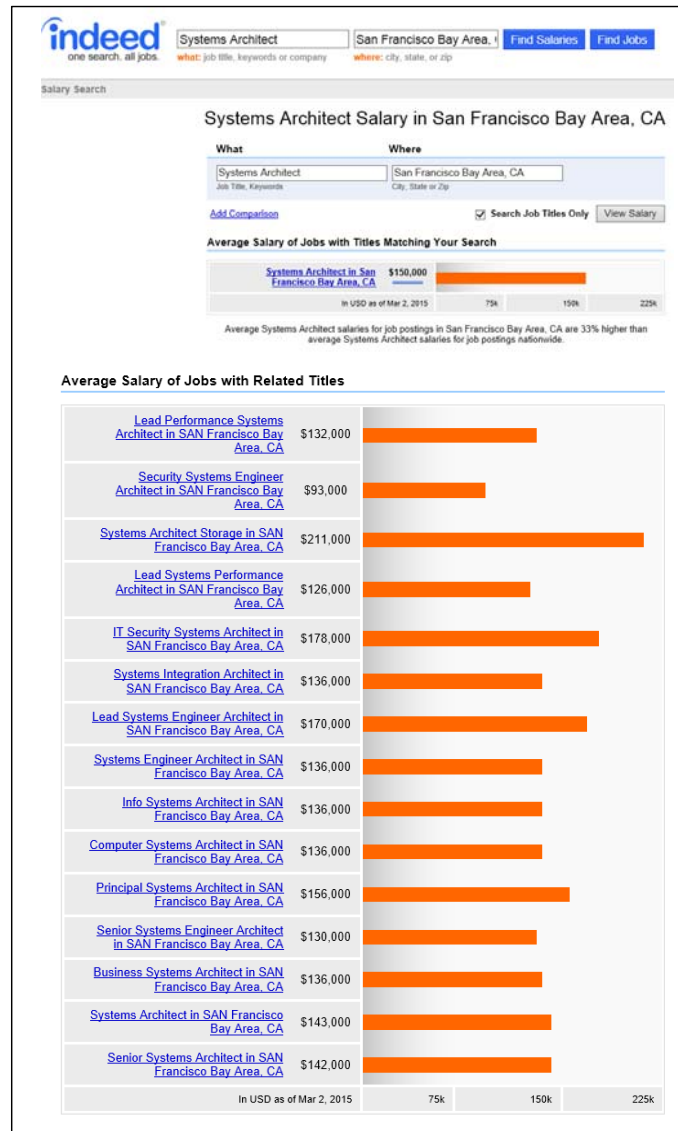


© Kim 2018 -- Portions © Austin, Brehob, Falsafi, Hill, Hoe, Lipasti, Martin, Roth, Shen, Smith, Sohi, Tyson, Vijaykumar, Wenisch

Good News



This training pays back



Annual salary of system engineers

\$150,000+ (year)

Major companies pay more

\$250,000+ (year)

Even grad-student interns make

\$10,000+ (month)



You are in the right class

◆ **This course** will cover

- Good knowledge on computer architecture and system
- Hands-on skills in C/C++ programming (for simulation)
- Hands-on skills in Verilog programming (for prototyping)
- Read advanced research papers (for research)

You will successfully compete with top-notch students.

◆ You will follow a wonderful career

- Top-tier industry (e.g., Intel, IBM, Google, MS, Facebook)
- Top-tier academia (e.g., grad schools, research labs)

Your efforts will pay back in real world.



You can follow computer/HW track!

◆ Follow the given curriculum

– core-architecture courses

- System programming
- **Digital system design**
- **Computer architecture**
- **Advanced computer architecture**
- Parallel computer architecture
- VLSI system design

} What I teach

– almost core-architecture courses

- Operating system
- Compiler design
- Computer network
- Computer security
- Some mathematics and statistics ...



If you do very good in this course

- ◆ Your training quality is “officially” guaranteed

- I will write strong recommendation letters
 - Global IT companies
 - Top-tier foreign universities
- I will consider recruiting you to my lab
 - For only 2-3 seats per year
 - Will work on top-tier graduation projects


There are many, many success stories



Bad News



Intel's Job Posting



[For Business](#) | [For Home](#) | [Products](#) | [Support](#) | [About Intel](#)

[IT Center](#) | [Developer Center](#) | [Partners](#) | [Technology](#)

Search

Communities

[Jobs at Intel Home](#) > [Candidate Help Desk](#) >

Job Search

My Jobpage

[Back to prior page](#)[Printable Format](#)

Power Performance Architect - 623631

Description

In this position you will be responsible for developing next generation of power/performance analysis tools, developing new methodologies, building prototype proof of concepts for microarchitecture features, and working with others teams to make them production worthy for conducting quantitative performance and power analysis. You will have opportunity to help defining new product microarchitecture features working with performance, power, and architecture, design teams to meet landing zone goals for new products.

Qualifications

Minimum Qualifications:
You should possess a PhD degree in EE, CE, or CS with experience/significant thesis work in architecture and performance/power analysis. You should have proficiency in C/C++ programming and experience with UNIX* development environment including Perl scripting language. You should be able to effectively work in a team, have strong initiative to deliver high quality results, and enjoy exploring breadth of technical options.

Additional qualifications that could be useful for this position include:
Experience in power and analytical/simulation performance modeling and simulation environments
Experience related to direct product development in design/validation/software development

Job Category : Engineering

Primary Location : USA-Oregon, Hillsboro

Full/Part Time : Full Time

Job Type : College Graduate



Google's Job Posting

☆ Platforms Performance Engineer

Mountain View, CA, USA

Hardware Engineering · Full-time

Know someone who would be interested?

APPLY NOW

Find connections

Know someone at Google? Reach out to them

Google's software engineers develop products that connect, explore, and interact with information at massive scale, and bring fresh ideas from all areas, including system design, networking and data processing, UI design and mobile. You will work on a specific project or projects as you and our fast-paced team display leadership qualities and continue to push technology forward.

With your technical expertise you will design, develop, test, deploy, maintain and improve our products.

You will be working on new platform performance bottlenecks with special optimizations. These will include improvements in the software stack.

Responsibilities

- Characterize end to end system performance
- Model and analyze key Google system design.
- Understand and optimize for key features as well as novel customer requirements.
- Work with the infrastructure and hardware vendors to bring Google products to market.
- Work with the infrastructure and hardware vendors to understand and that the capabilities.

Minimum qualifications

- BS degree in Computer Science or Electrical Engineering degree, or equivalent practical experience.
- Experience in C/C++, Scripting. Experience with computer architecture and new hardware design, including memory/storage systems.
- Experience with simulation and performance trade-off analysis for systems software.
- Experience in hardware bring up, Linux kernels, application software, and RTL design.

Preferred qualifications

- MS/PhD in Computer Engineering, Electrical Engineering, or Computer Science.
- Experience with large-scale distributed systems, networking, and related software infrastructure.
- Experience with silicon architecture and design with the ability to root cause performance bottlenecks.
- Self-driven and good problem solving; work across different teams across the hardware/software stack.
- Proficient with measurement tools related to profiling systems/storage performance, health, utilization.
- Proficient with statistical analysis tools including machine learning.



You must know “everything”

It means A LOT OF WORK!



Hours of efforts per week

- ◆ Attend lectures
 - ~2.5 hrs / week
- ◆ Read book & handouts
 - ~3 hrs/ week
- ◆ Homework assignments
 - ~5 hrs / week
- ◆ Verilog assignments (**your “mandatory” Tuesday lab**)
 - ~5 hrs / week
- ◆ Programming assignments
 - ~5 hours / week
- ◆ Studying, etc
 - ‘n’ extra hrs / week

} **undecided yet**

Expect to spend ~15-20 hours / week on this class.



Hours of efforts per week

- ◆ Attend lectures
 - ~2.5 hrs / week
- ◆ Read book & handouts
 - ~3 hrs / week

This is “global standard”

All top-tier US universities expect such efforts for their architecture courses

undecided yet

- ◆ Studying, etc
 - ‘n’ extra hrs / week

Expect to spend ~15-20 hours / week on this class.



If you are NOT READY for this load

- ◆ This course is **NOT for Everyone**
 - Don't overestimate your will
 - Must use whatever time you find for STUDY
 - Don't underestimate the load of system/HW courses
 - E.g., Architecture, Operating System, Verilog Prototyping
- ◆ What happens
 - 20+% eventually dropped and make their lab partners frustrated
 - 90+% of exchange students either dropped or failed (and complained)

WARNING:

**If you are not ready,
you must consider taking "the other session"**



© Kim 2018 -- Portions © Austin, Brehob, Falsafi, Hill, Hoe, Lipasti, Martin, Roth, Shen, Smith, Sohi, Tyson, Vijaykumar, Wenisch

Course Info



Instructor: My Past Work in Industry

- ◆ As a CPU/system design architect

- CPU development @ **Sun Microsystems**

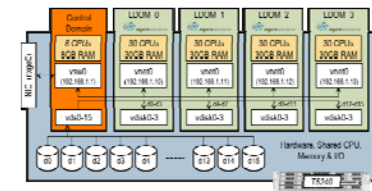
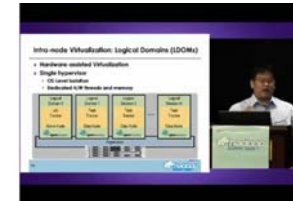
- Ultra SPARC “Yosemite Fall” CPU
 - T4/T5 SPARC server



- The world’s highest-throughput many-core systems

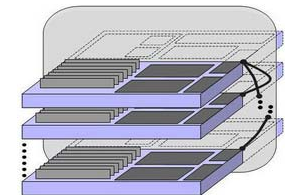
- Cloud operating system @ **Oracle**

- Cost-effective datacenter design
 - Highly scalable Hadoop engine



- System Simulation @ **Carnegie Mellon**

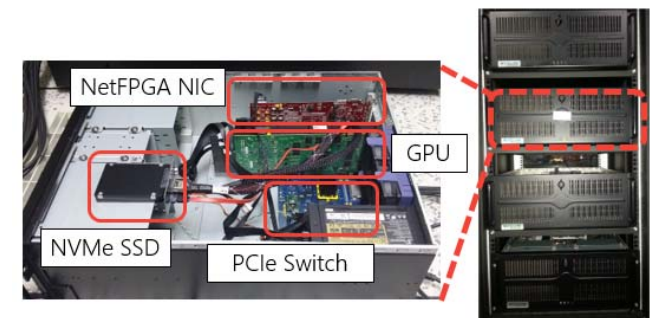
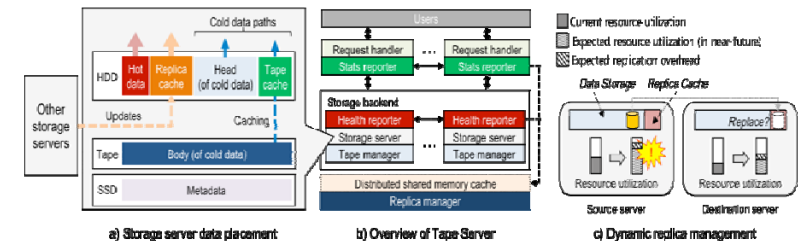
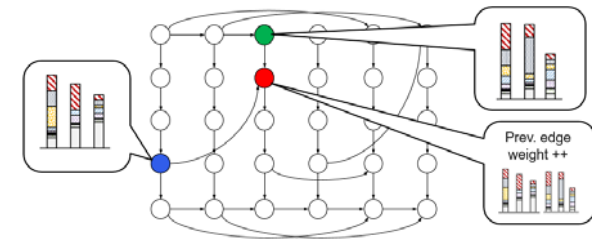
- Flexus: open-source full-system simulator
 - TRUSS: a reliable and scalable, DSM server





Instructor: My Recent Work in Academia

- ◆ We are doing “hard-core” system research!
 - **System Performance Modeling**
 - Fast and accurate performance analysis
 - Pinpoint performance bottlenecks
 - **System Software**
 - Dynamic resource management
 - QoS guarantee at minimum costs
 - **System Prototyping**
 - Enable both scale-up and scale-out
 - Improve big-data processing significantly!





Course Info

◆ Instructor

- Jangwoo Kim jangwoo@snu.ac.kr @ R902-B301

◆ TAs

- Pyeongsu Park pyeongsu@snu.ac.kr
 - Eunjin Baek ebaek@snu.ac.kr
 - Suheon Bae sheon.bae@snu.ac.kr
 - Hunjun Lee hunjunlee7515@snu.ac.kr
 - All of TAs ece332-ta@hpcs.snu.ac.kr
 - and **Anyone (=my grad students)**
in High Performance Computer System (HPCS) Lab
- R851.4-B301
or
R1114-B301

◆ URL

- **eTL** (lectures, labs, and Q&A board)



Course Info (continued)

◆ Textbook



- P&H, 5e // three versions: ARMS, RISC-V & **MIPS**
(computer organization & design by Patterson & Hennessy, 5th edition)

We use this version

◆ Grading (for now)

- | | |
|-------------|-----|
| - Midterm | 30% |
| - Final | 30% |
| - HW & Quiz | 10% |
| - Project | 30% |



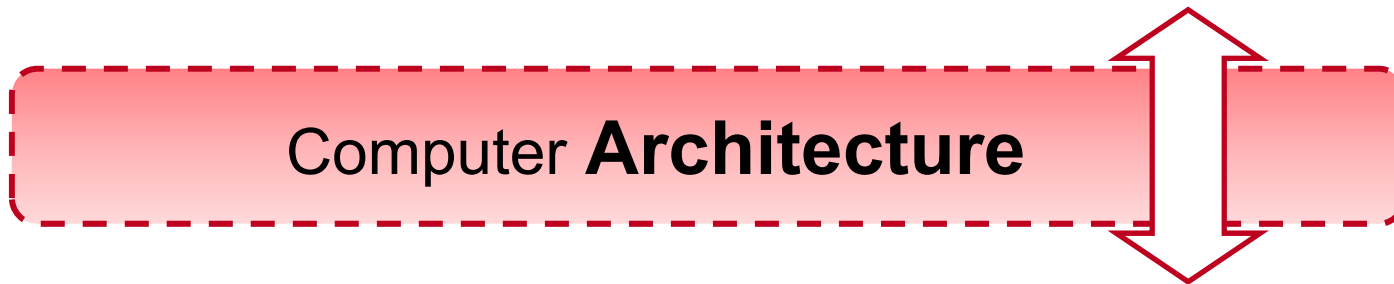
Let's start!



What is Computer Architecture?

◆ System Programming

- “C” as a model of computation & x86 as a machine language
- Interact with the computer hardware through OS
- What about the details **below the abstraction?**



◆ Digital System

- Digital logic as a model of computation
- Gates and wires as building blocks
- What about the details **above this abstraction?**



What is Computer Architecture?

- ◆ System Programming
- ◆ Computer Architecture
 - Computer Architecture
 - How to present precisely **the functionality** of a computer
 - The most abstract “design spec” for HW engineers
 - Computer Organization (a.k.a. Microarchitecture)
 - How to design (e.g., **better performance, lower power?**)
 - How to evaluate
 - How to tune
 - Computation Structure
 - Digital representations
 - Processing, storage and I/O elements
- ◆ Digital Design



What is a Computer?

◆ Computer

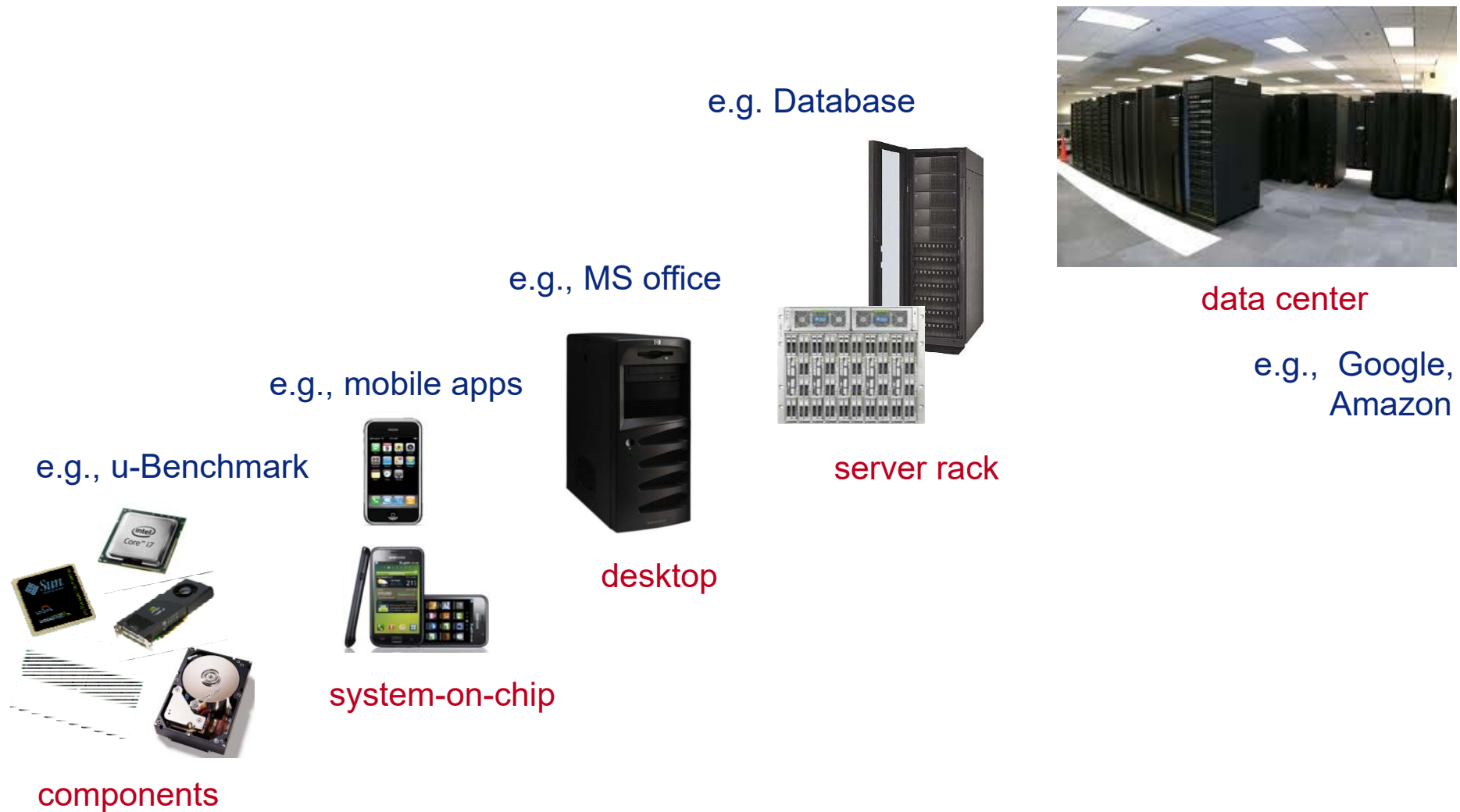
: one that computes; *specifically* : a programmable usually electronic device that can store, retrieve, and process data

-- Merriam-Webster English Dictionary



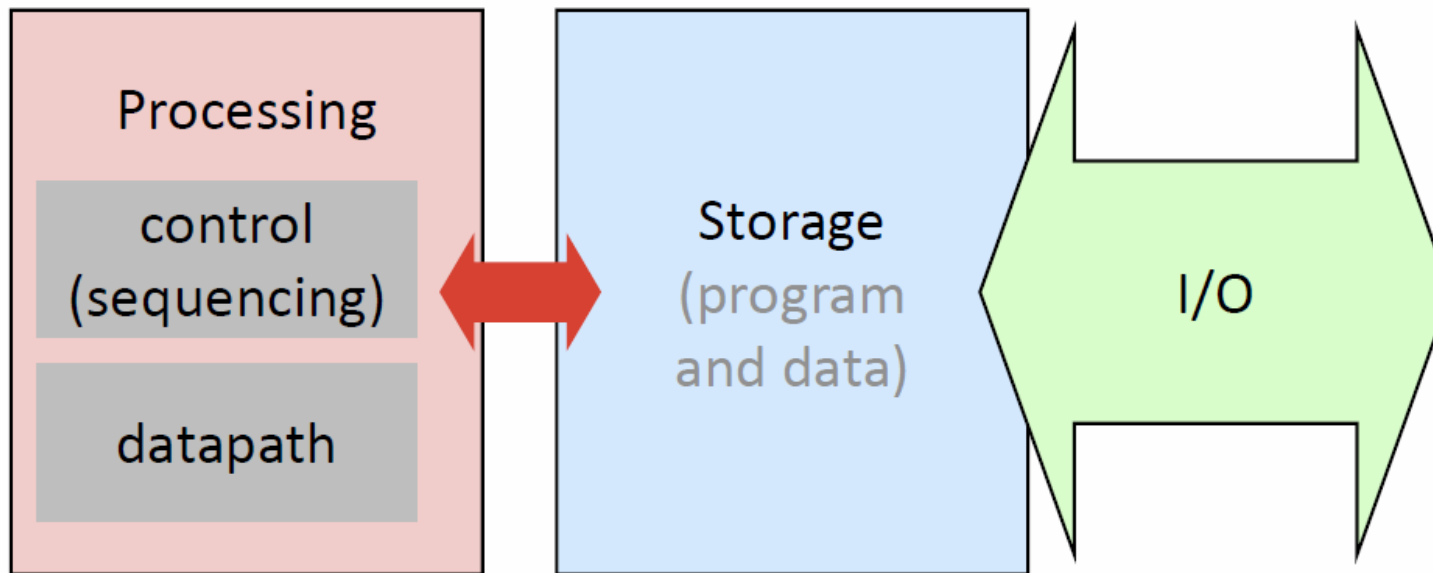


Computers?





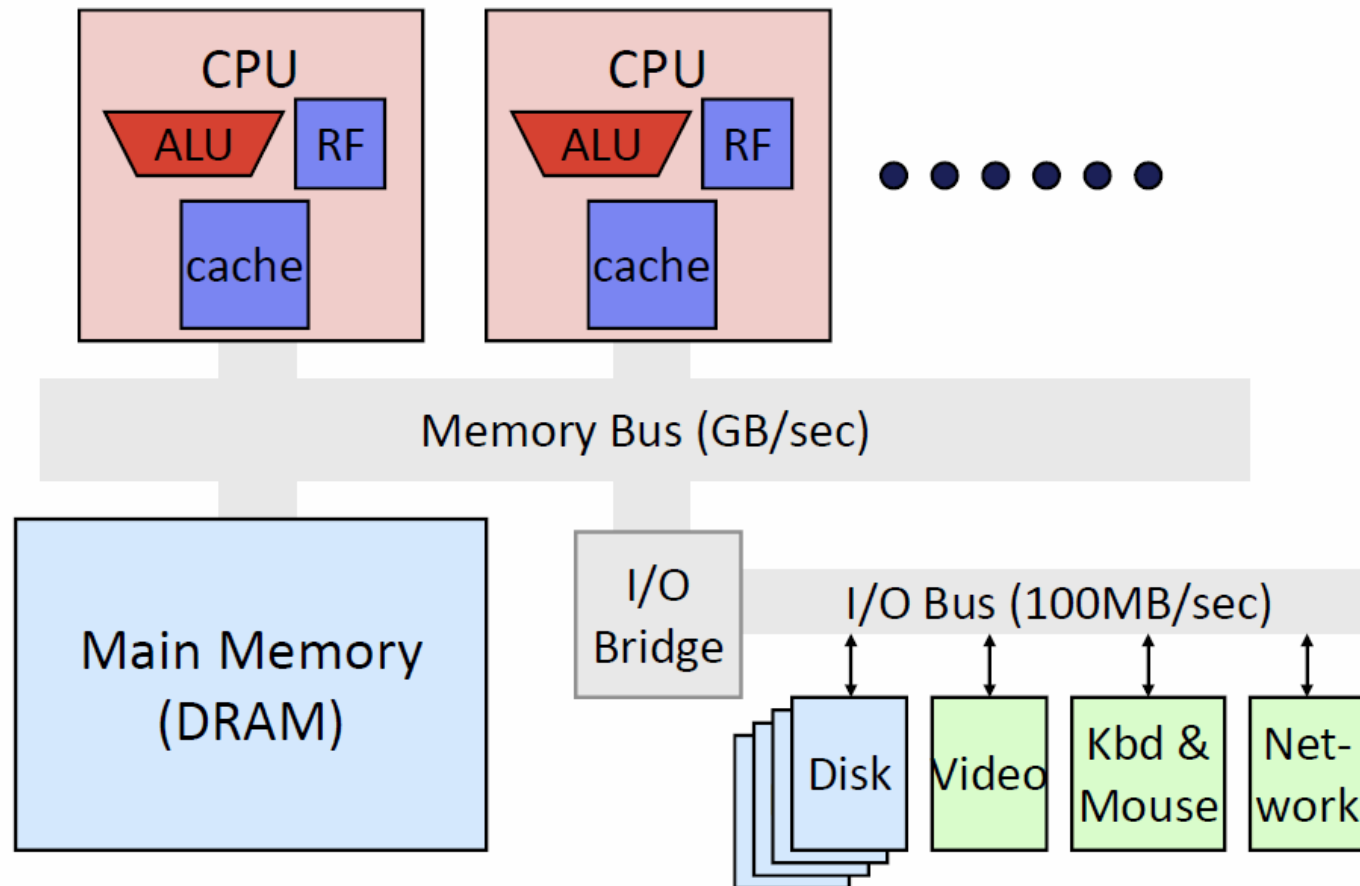
So, what makes a computer a computer?



Von Newman Architecture (or stored-program computer)
:: both program and data are stored in memory



A Typical Computer Organization





Computer System Abstractions

This
Course



Applications
Compilers
OS
Architecture (ISA)
Microarchitecture
Digital Design
Circuits
Devices/Physics

← Users & programs

← Prog. languages

← Resource utilization

← **HW/SW interface**

← **Data & control path**

← **Registers, ALU, ..**

← **Digital logic**

← Transistors, signals

← Atoms, electrons



Computer Architecture Analogy

◆ Architecture Level

- Car: driving manual & operation manual
// you don't have to be a car mechanic to drive a car.
- Computer : ?
// you don't have to be a circuit designer to use a computer.

◆ Microarchitecture (implementation) Level

- A particular car design has a certain configuration of electrical and mechanical components
- A particular computer design has a certain configuration of datapath and control logic units.

◆ Realization level

- Car: metal sheets, cranks, shafts, gears,
- Computer: gates, wires, semiconductor, transistors, ...



Wait. We keep saying # of transistors...

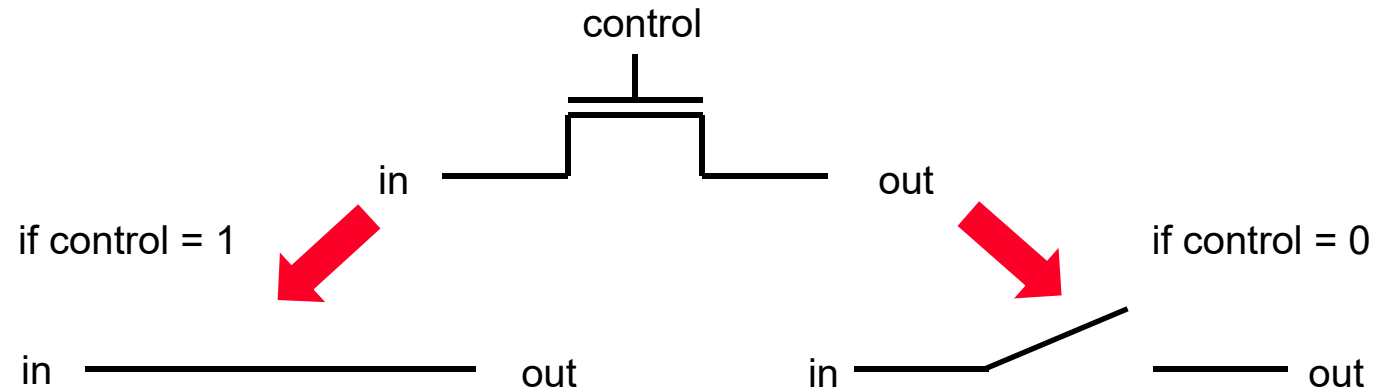
What is a “transistor” by the way?



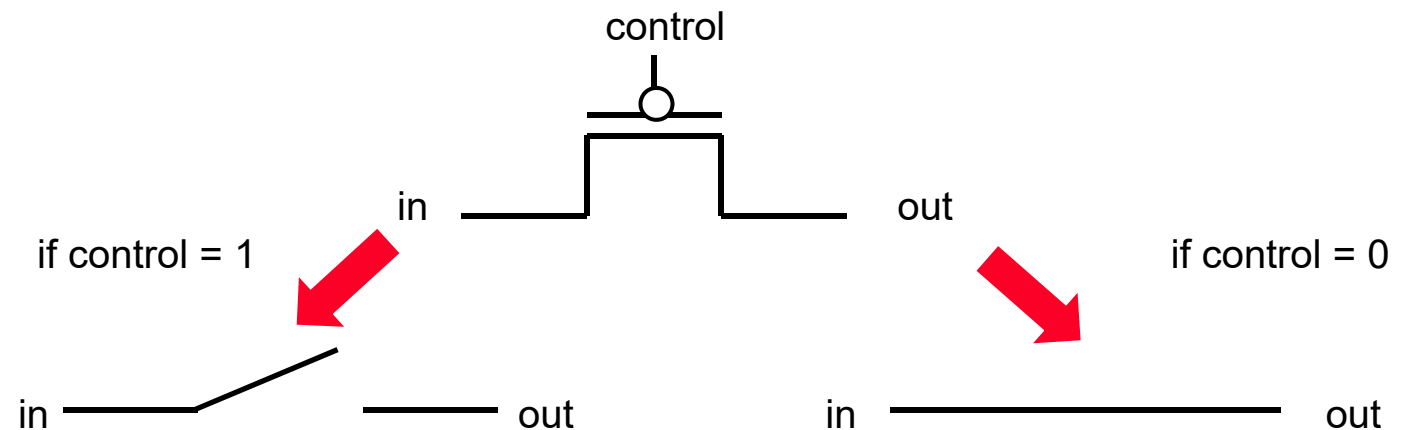
Digital circuits = a lot of switches

- ◆ Transistor, vacuum tube,
 - switch to connect or disconnect for 0 (gnd) and 1 (Vdd)

TURN ON @ 1



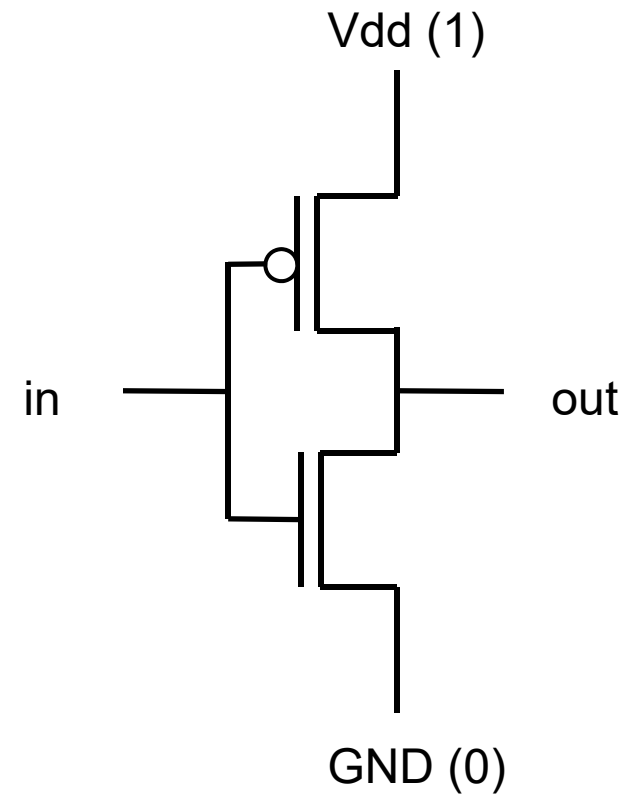
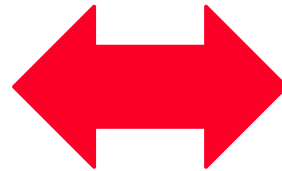
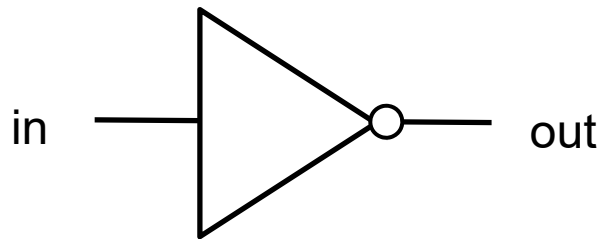
TURN ON @ 0





Inverter

- ◆ $\text{out} = \sim \text{in}$
 - If $\text{in} = 0$, $\text{out} = 1$
 - If $\text{in} = 1$, $\text{out} = 0$

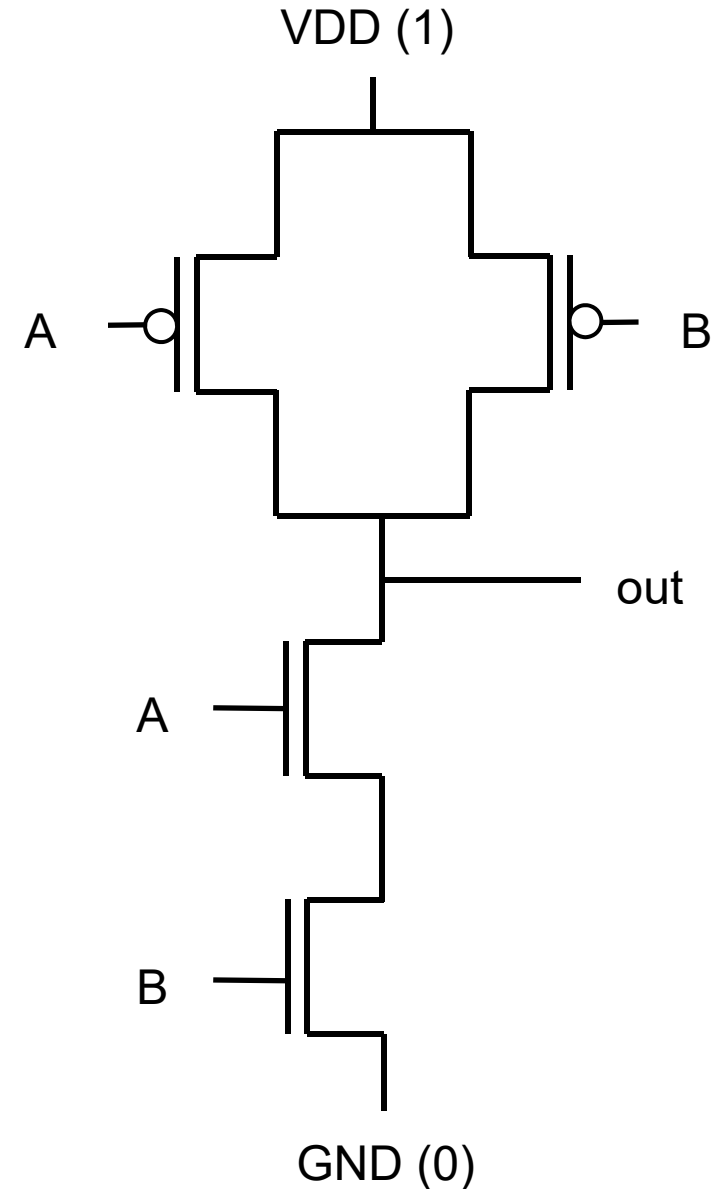
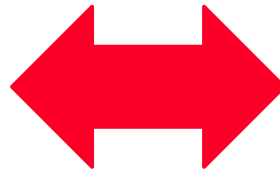
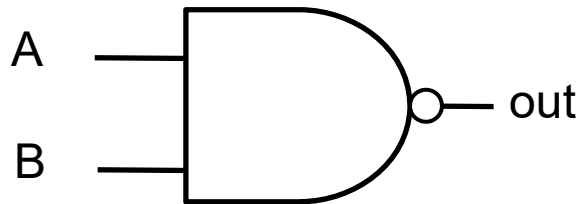




NAND = not AND

◆ $\text{out} = A \text{ nand } B = \neg(A \text{ and } B)$

- If $A = 1$ & $B = 1$, $\text{out} = 0$
- If otherwise, $\text{out} = 1$



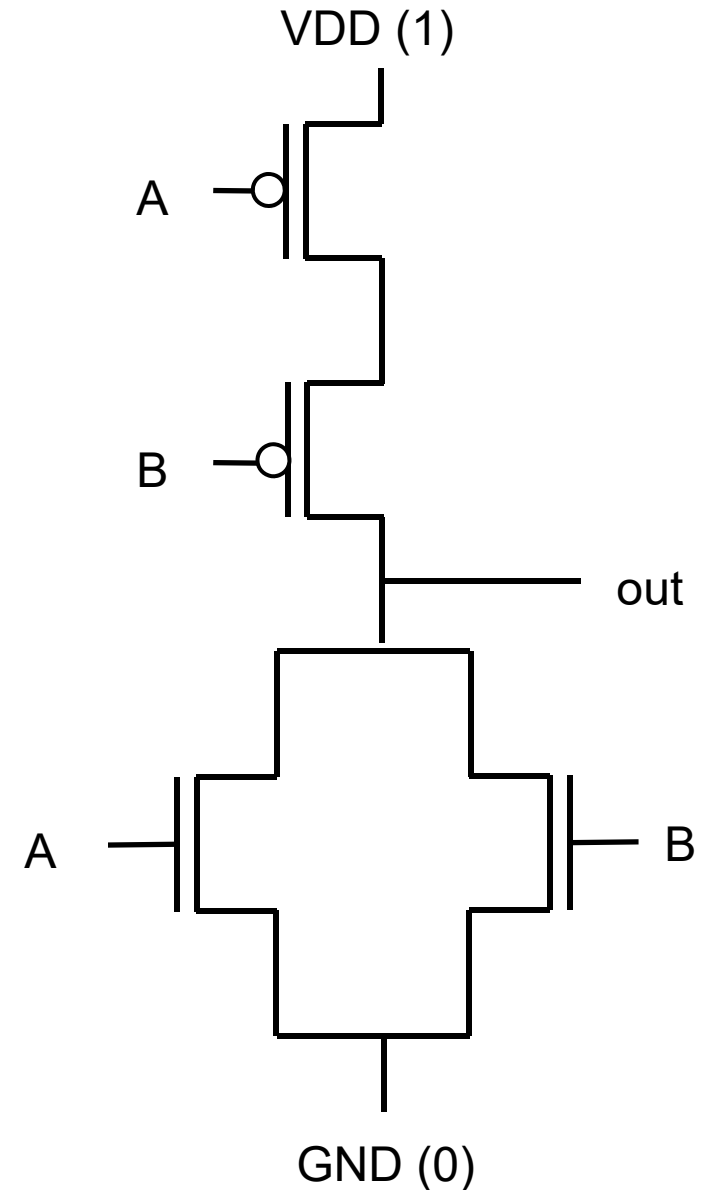
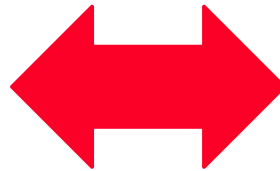
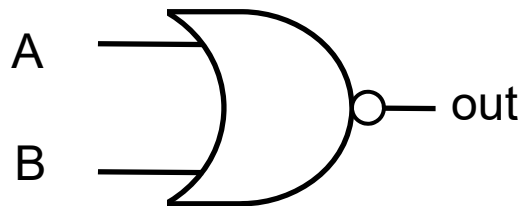
*You can make AND
if you add INV at the end.*



NOR = not OR

◆ $\text{out} = A \text{ nor } B = \neg(A \text{ or } B)$

- If $A = 0$ & $B = 0$, $\text{out} = 1$
- If otherwise, $\text{out} = 0$



*You can make OR
if you add INV at the end.*



If you have INV, NAND, NOR..

You have **everything** to build a digital logic device and memory.

- Combinational circuits
- Sequential circuits

CPU, memory and ... EVERYTHING!!

*However, this is NOT what we cover from this class.
(take digital design courses offered in EE)*



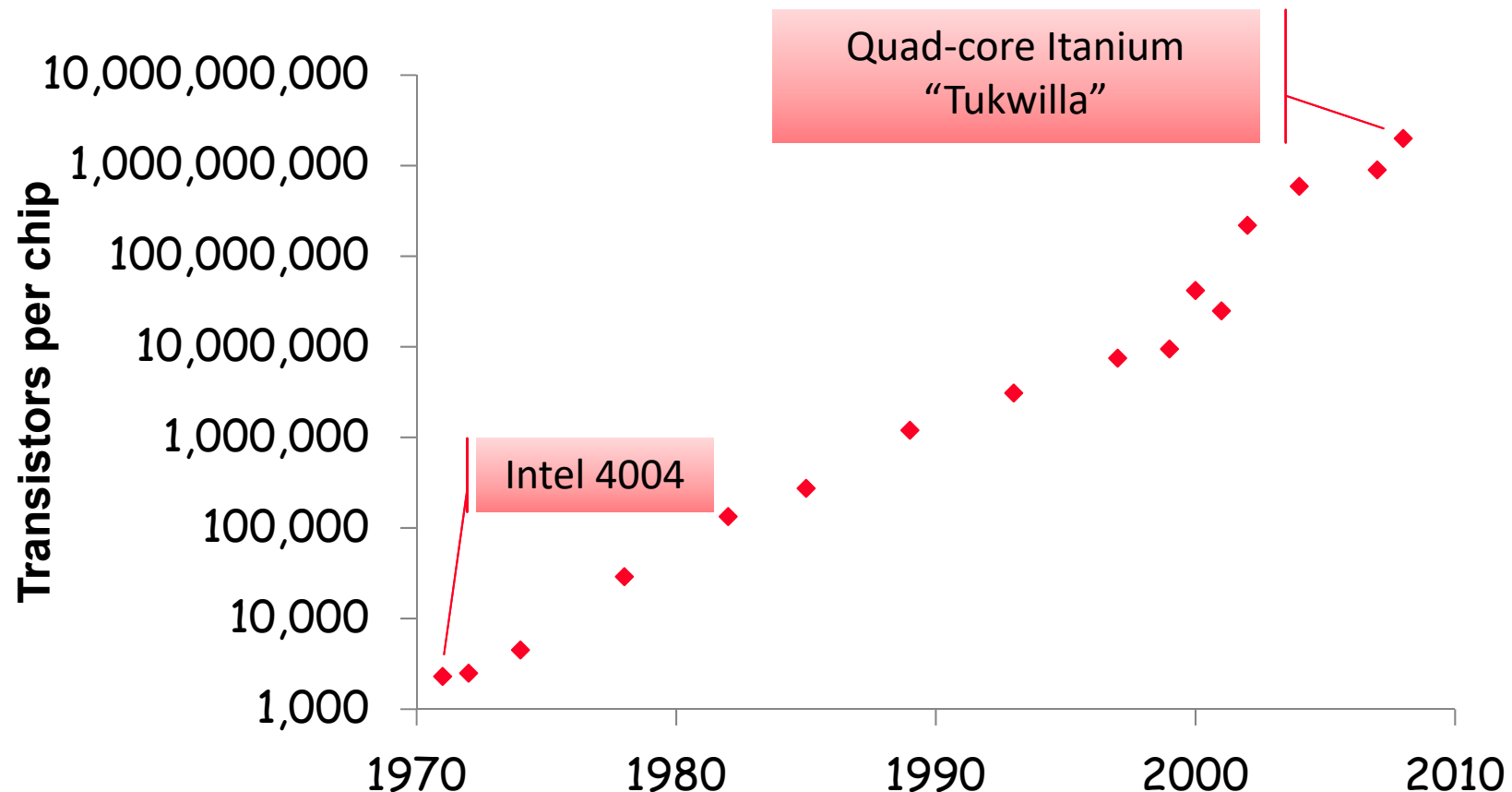
What is Moore's Law?

What is Dennard Scaling?



Moore's Law

- The number of transistors on a chip roughly doubles every two years -



- Transistor density continues to increase 35% per year
- Another decade of scaling is still likely



Evolution of Single-Chip

	1970's	1980's	1990's	2010 (extrapolated)
Transistor Count	10K-100K	100K-1M	1M-100M	1B
Clock Frequency	0.2-2MHz	2-20MHz	20M-1GHz	10GHz
Instruction/Cycle	< 0.1	0.1-0.9	0.9- 2.0	10 (?)
MIPS/MFLOPS	< 0.2	0.2-20	20-2,000	100,000

- ◆ OK. # of transistors getting increased as such.

Is CPU getting 2x faster every two year? If not, why?



Dennard scaling

- ◆ Dennard Scaling is about “performance” scaling
 - Moore’s Law tells me transistors will become smaller
 - Dennard Scaling tells me, as transistors became smaller
 - They must switch faster, use less power but achieve the same power density.
- **should lower supply voltage, current, and capacitance**

Device or Circuit Parameter	Scaling Factor
Device dimension t_{ox}, L, W	$1/k$
Doping concentration N_a	k
Voltage V	$1/k$
Current I	$1/k$
Capacitance eA/t	$1/k$
Delay time per circuit VC/I	$1/k$
Power dissipation per circuit VI	$1/k^2$
Power density VI/A	1

<Dennard’s scaling rule>

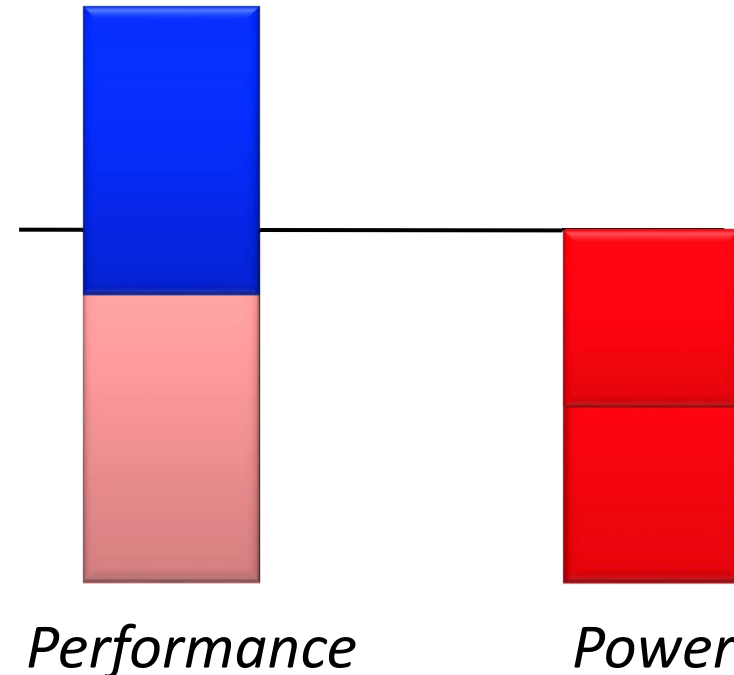


Parallelism is a solution?

$$\text{Power} = CV^2F \quad F \propto V$$

Scale clock frequency to 80%
by reducing V .

Now add a second core!



Same power budget, but 1.6x performance!

But:

- Must “parallelize” application, which is usually impossible
- Remember Amdahl’s Law! (=make common-case fast)



Historical Perspectives: prelude to modern computer architecture



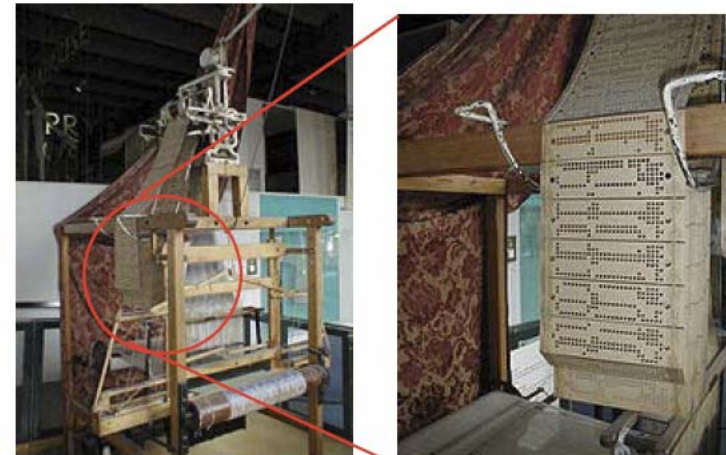
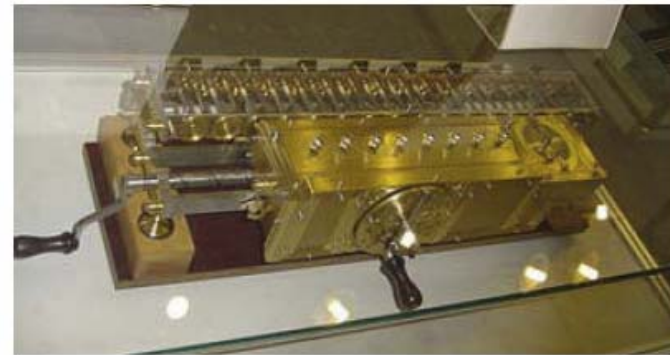
Pitfalls of Computer Technology Forecasting

- ◆ “I think there is a world market for maybe five computers.”
Thomas Watson, *IBM Chairman*, 1943
- ◆ “There is no reason anyone would want a computer in their home.” Ken Olsen, *DEC founder*, 1977
- ◆ “DOS addresses only 1 MB of RAM because we cannot imagine any applications needing more.” *Microsoft*, 1980
- ◆ “640K ought to be enough for anybody.” *Bill Gates*, 1981
- ◆ “The 32-bit machine would be an overkill for a personal computer.” Sol Libes, *ByteLines*



Beginnings of Digital Computing

- ◆ Industrial Revolution era's "hi-tech" in mechanization
 - Steam engines
 - Mechanical calculators
 - Jacquard's loom:
gears, pulley's,
chain and punch cards





Charles Babbage (1791-1871)

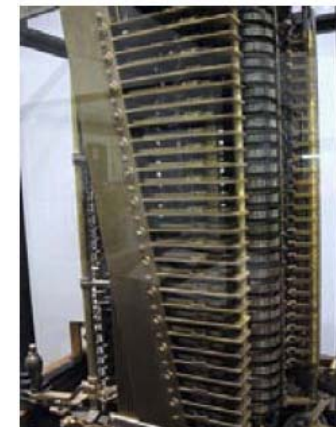
◆ Difference Engine, 1823: **a special-purpose computer**

- Evaluated polynomial functions by Newton's method of successive differences (requiring only additions)
- Eventually built by Georg and Edvard Shueetz in 1855



◆ Analytical Engine, 1833: **a general-purpose computer**

- **Programmed** by punch cards, “**assembly language**” included loops and branches
- 1000 word data storage, punch card I/O
- Unfortunately never completed (may not be feasible ever in mechanical forms..)
- Would have been 10x30 meters, powered by a steam engine (if it was possible)





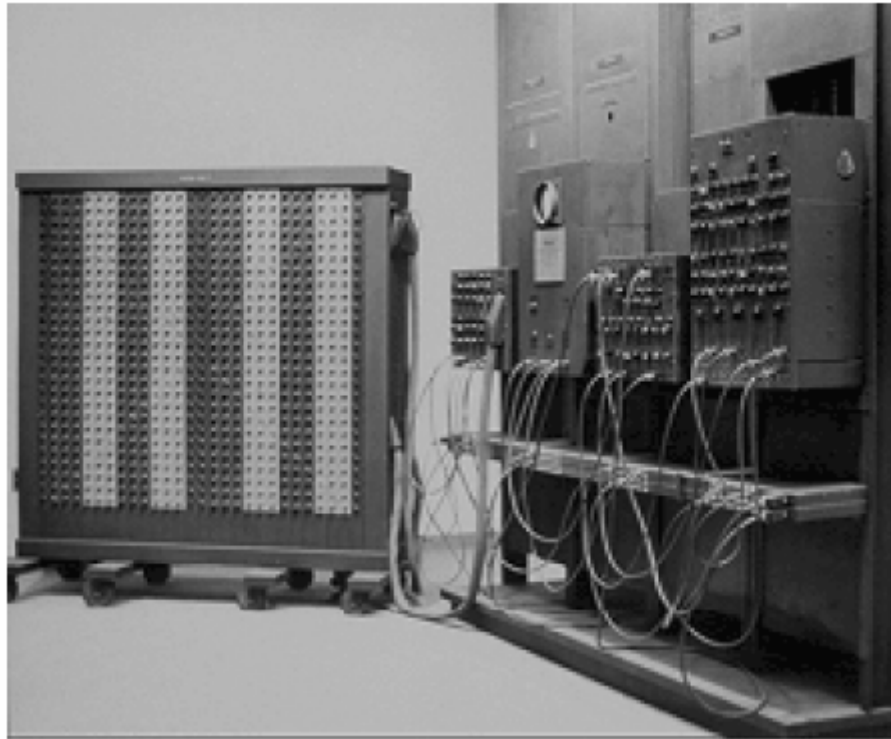
100 Years of Technology Changes

- ◆ Mechanical, 1800s
 - Gears, chains, pulleys, and steam power
 - Punch cards
- ◆ Electromechanical, early 1900s
 - Switches, relays
 - “Acoustic” mercury delay lines
 - e.g. Harvard/IBM Mark 1, Aiken 1939-1944, 50ft long, 5 ton, 750K pars, 3~6 sec per addition
- ◆ Electrical, mid 1990s and on
 - Plugboards, vacuum tubes, CRTs
 - Later came DRUM memory, core memory, transistors,



ENIAC, 1946

Eckert and Mauchly, U of Penn



from The ENIAC Museum
<http://www.seas.upen.edu/~museum>

- ◆ The first programmable electronic digital computer
- ◆ 18,000 vacuum tubes
- ◆ 30 ton, 25 x 2.6 meter
- ◆ 1900 additions per second
- ◆ 20 10-decimal-digit words (100-word core by 1952)
- ◆ programmed by **3000 switches** in the function table and plug-cables (became **stored-program** in 1948 following **von Neumann's** advise)



Proliferation 40s and 50s

- ◆ From Moore School of Engineering, U of Penn
 - ENIAC, Eckert & Mauchly, 1946
 - EDVAC, von Neumann, 1944~1952
 - EDSAC, Wilkes, 1949 (first stored-program built)
 - IAS, Bigelow, 1952
 - ORDVAC, SEAC, MANICA, JOHNICA, ILLIAC, ..
- ◆ They are not alone
 - ABC, Atanasoff and Berry, 1939~1942
 - Z3, Z4, Konrad Zuse, late 1930's ~ early 1940's
 - Colossus, Alan Turing 1943
 - Harvard Mark I, Aiken
- ◆ Don't forget software advances
 - FORTRAN (1954), **C (1973)**, **C++ (1983)**, JAVA (1995)



Commercialization the 50s

- ◆ UNIVAC, 1951, the first commercial computer
 - sold only 48 units (~\$1M)
 - ◆ IBM 701, 1952
 - sold 19 units
 - leased at \$12K per month
 - ◆ IBM 650, 1953
 - sold ~2000 units (\$200K~\$400K)
 - ◆ **IBM System/360, 1964**
 - A family of **binary-compatible** computer
 - 19 combinations of varying speed and memory capacity (from \$200K ~ \$2M)
 - Still lives on today as the “highly-profitable” IBM z900 series (Chief architect: Gene Amdahl, Project leader: Fred Brooks)
- “architecture” concept
- Redefined Industry!!
-
- A blue arrow points from the text “architecture” concept to the IBM System/360, 1964 entry. A red arrow points from the text Redefined Industry!! to the same entry.



Cheaper or faster in 60s and 70s

◆ Minicomputers

- DEC PDP-8, 1965, \$20K, size of large refrigerators
- less powerful than “mainframes”, 10x cheaper
- departmental computers, timesharing --- PDP-11 and VAXs enjoyed extreme popularity in the 70s and 80s

◆ Supercomputers

- performance at all costs!! (ECL, liquid-cooling, hand-built, ..)
- biggest customers: national security, nuclear weapons, cryptography, (also aerospace, petroleum, automotive, pharmaceutical, sciences)
- Seymour Cray, 1925~1996
 - Worked on UNIVAC and later CDC6600 & 7600
 - Pioneered many high-performance techniques in use today
 - Cray Research Inc. → Cray Computer Corp → Cray Inc.



Early Examples



DEC PDP 8, 1963
(an early mini)



Xerox Alto, 1973
An early “PC” with mouse & GUI



Cray 3, 1993



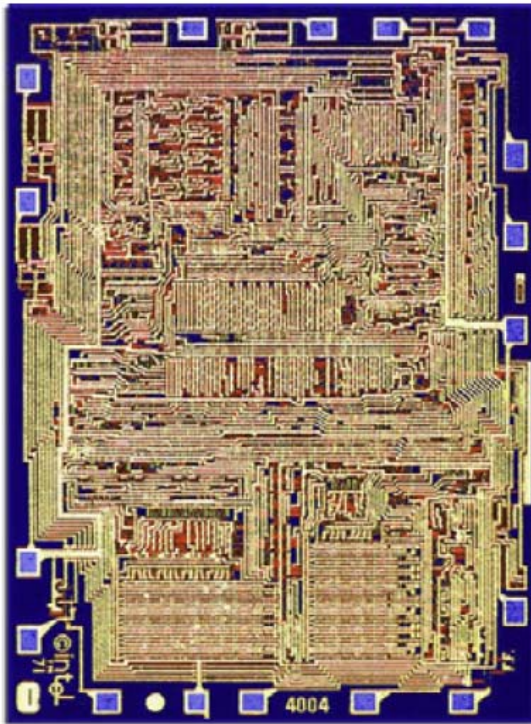
90KW: liquid cooled by “Fluorinert”

15 GFLOPS (1 sec on Cray 3 = 67 years on ENIAC)

cost: \$30M → sold only one machine → company went bankrupt.



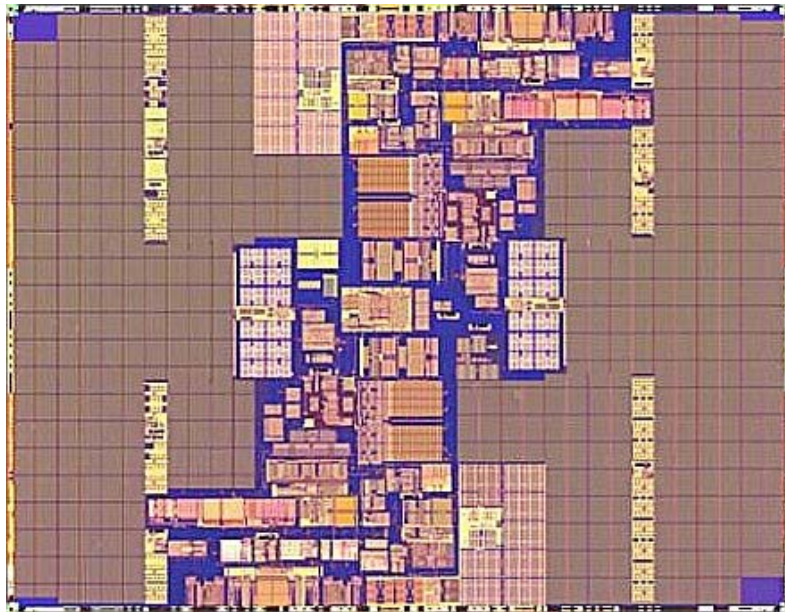
The “Killer Micros” from 70s and on



- ◆ Intel 4004, first single chip CPU
- ◆ 4-bit processor for a calculator
- ◆ 2,300 transistors
- ◆ 16-pin DIP package
- ◆ 740kHz
- ◆ ~100K operations per second



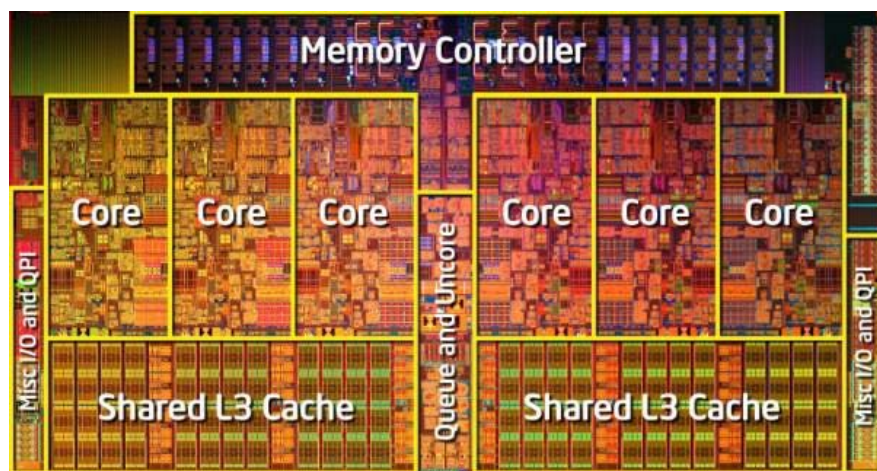
Intel Itanium (Montecito) 2004



- ◆ Intel Montecito (dual-core)
- ◆ 64-bit processor
- ◆ 1.7 billion transistors
- ◆ 1.7 Ghz, issue up to 8 instructions per cycle
- ◆ 26 MB cache!



Intel i5/i7/Xeon (Nehalem) 2008

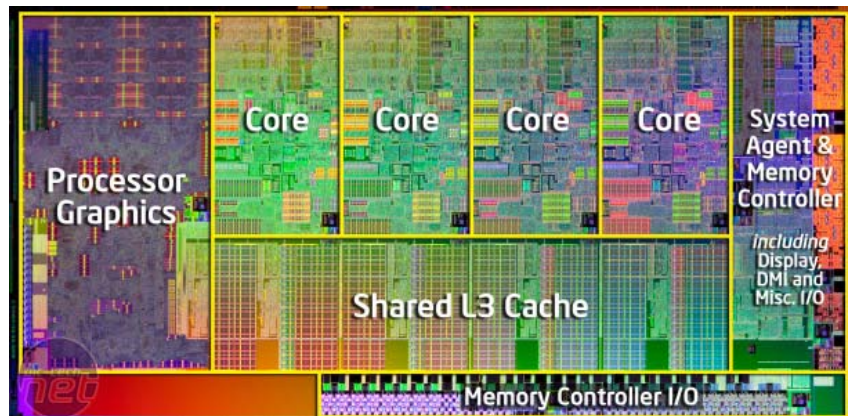


- ◆ 45nm technology
 - 32nm shrink → Westmere (2010)
- ◆ Caches
 - 64KB L1 per core
 - 256KB L2 per core
 - Up to 12MB L3 per chip
- ◆ 20-24 pipeline stages





Intel i5/i7/Xeon (Sandy Bridge) 2011

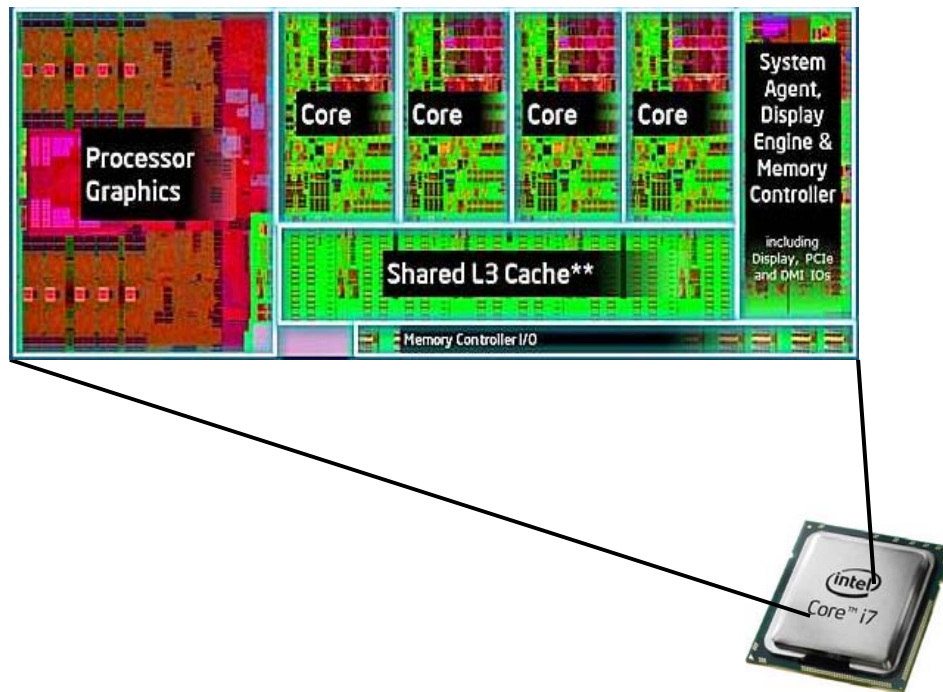


- ◆ 32nm technology
 - 22nm shrink → Ivy Bridge (2012)
- ◆ Caches
 - 64KB L1 per core
 - 256KB L2 per core
 - Up to 20MB L3 per chip
- ◆ **GPU on chip!**





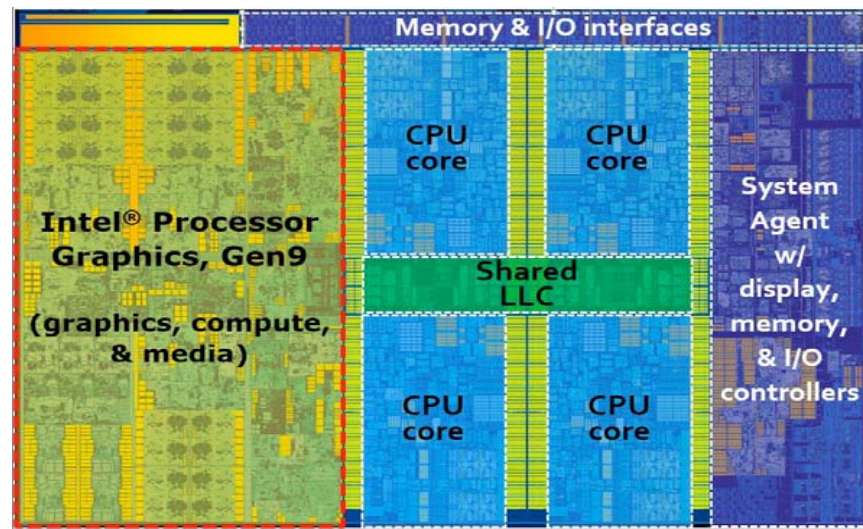
Intel i5/i7/Xeon (Haswell) 2013



- ◆ 22nm technology
 - Up to **5.5B transistors** per chip
 - 14nm shrink → Broadwell (2014)
- ◆ Cores
 - 2-4 (mainstream)
 - 2-18 (xeon)
- ◆ Caches
 - 64KB L1 per core
 - 256KB L2 per core
 - Up to 40MB L3 per chip
 - **Up to 128MB eDRAM L4**



Intel i5/i7/Xeon (Sky Lake) 2015



- ◆ 14nm technology
 - 10nm shrink → Kaby Lake (?)
- ◆ Cores
 - 2-4 (mainstream)
 - 2-18 (xeon)
- ◆ Caches
 - 64KB L1 per core
 - 256KB L2 per core
 - Up to 8MB L3 per chip
 - Up to 128MB eDRAM L4
- ◆ **Advanced I/O support**
 - e.g., 16 PCIe 3.0 lanes on chip



Question?

Announcements

Reading: P&H Ch. 1 & Ch. 2 (MIPS ver.)

Handouts: None