

Introduction to Computer Systems Architecture

Fall 2012

Huzefa Rangwala, PhD

Email: rangwala@cs.gmu.edu

<http://www.cs.gmu.edu/~hrangwal>

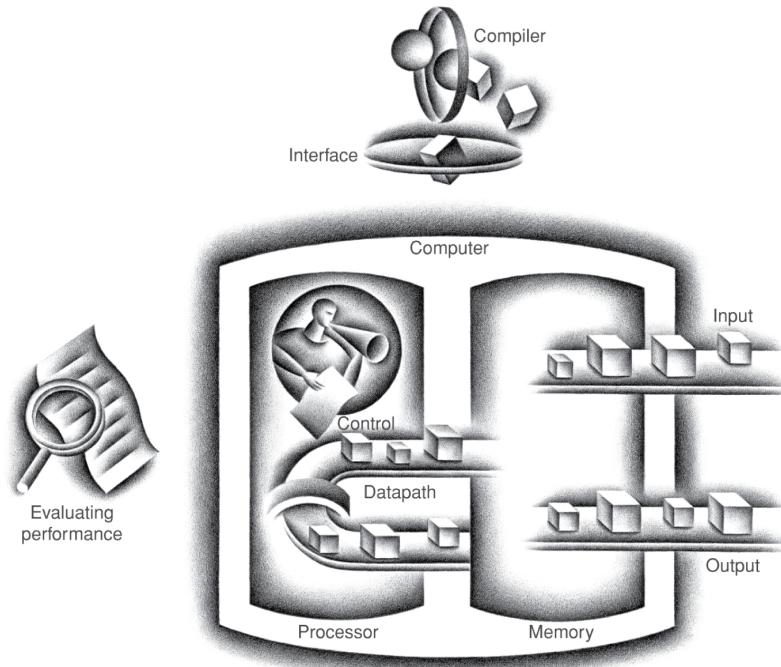
*Slides adapted from Computer Organization and Design by Patterson and Henessey

Roadmap

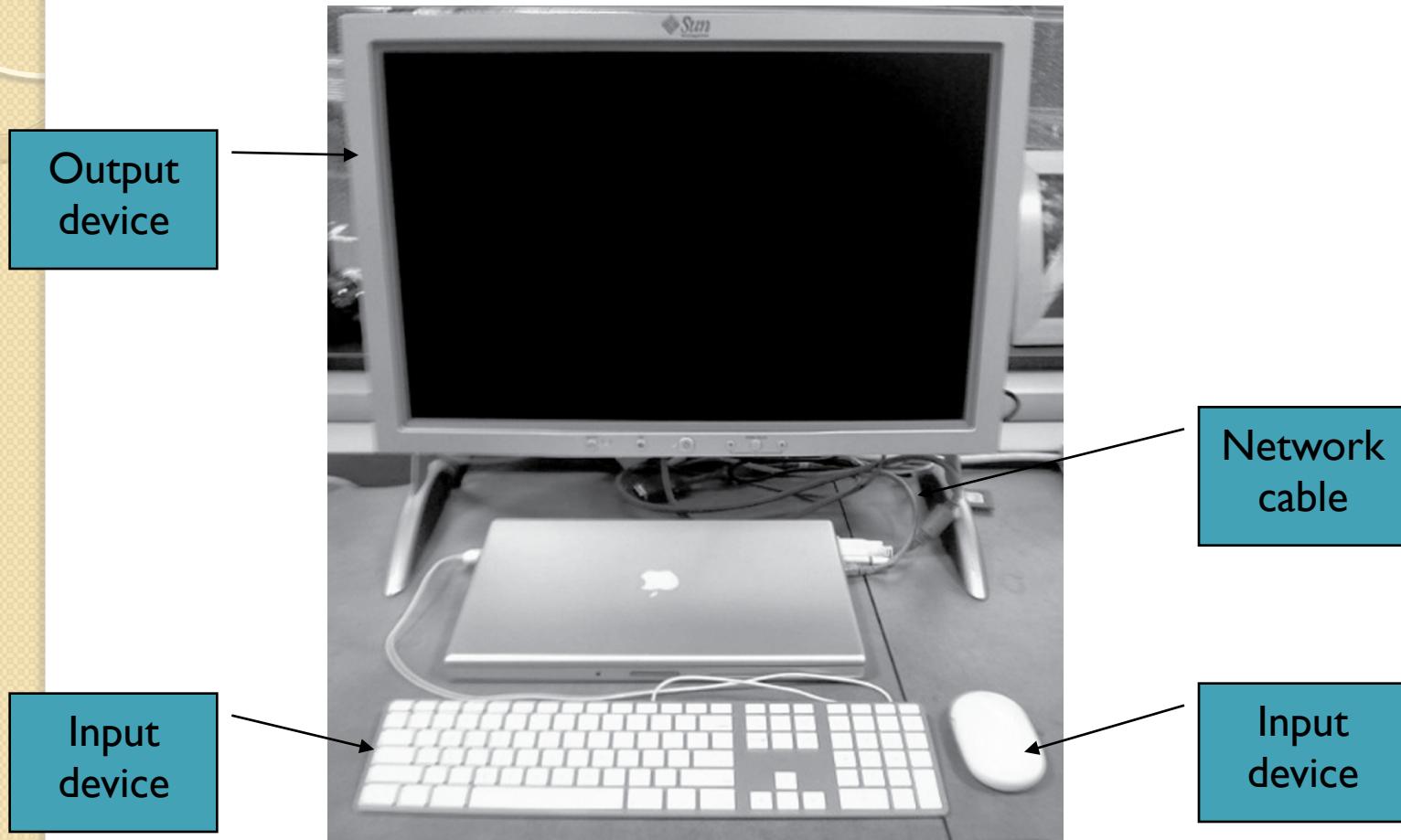
- Identify Components of a Computer ?
 - I trust you can do that.
- Computer Performance Metrics.
- Exercises in class.

Components of a Computer

The BIG Picture

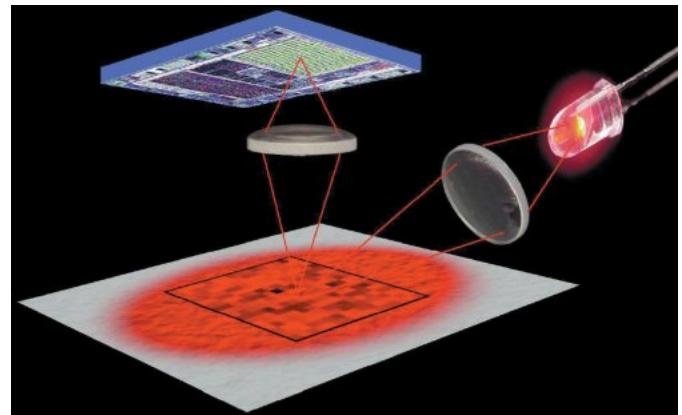


Anatomy of a Computer



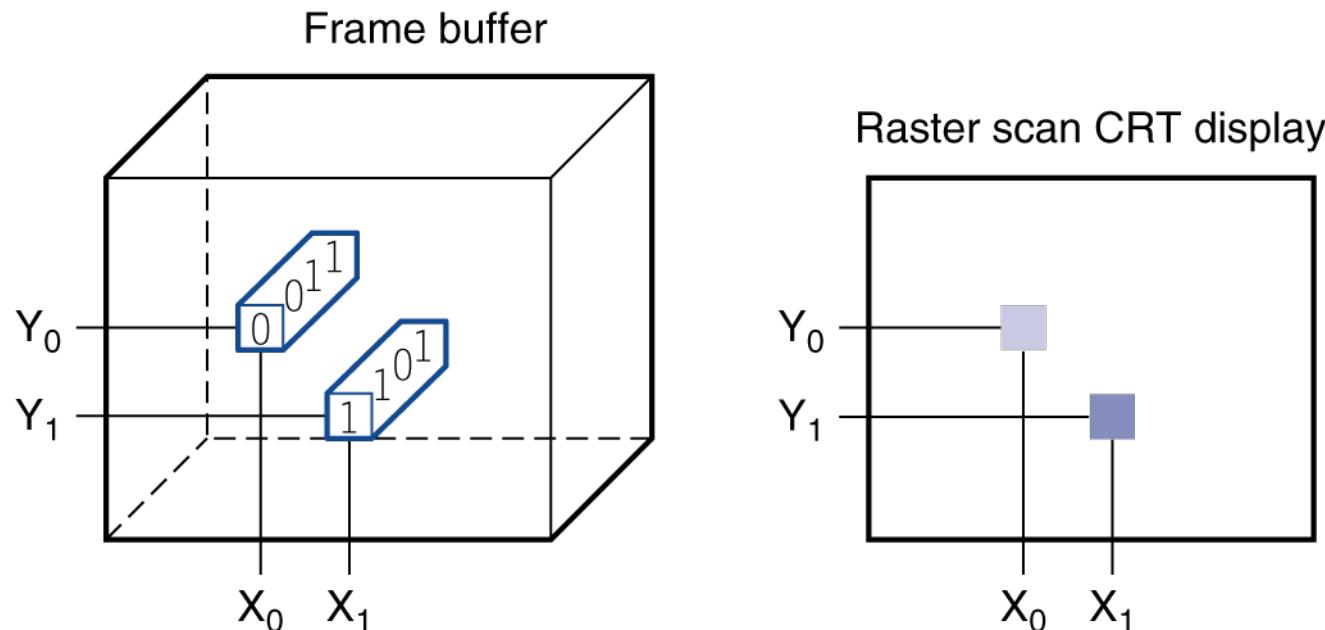
Anatomy of a Mouse

- Optical mouse
 - LED illuminates desktop
 - Small low-res camera
 - Basic image processor
 - Looks for x, y movement
 - Buttons & wheel
- Supersedes roller-ball mechanical mouse

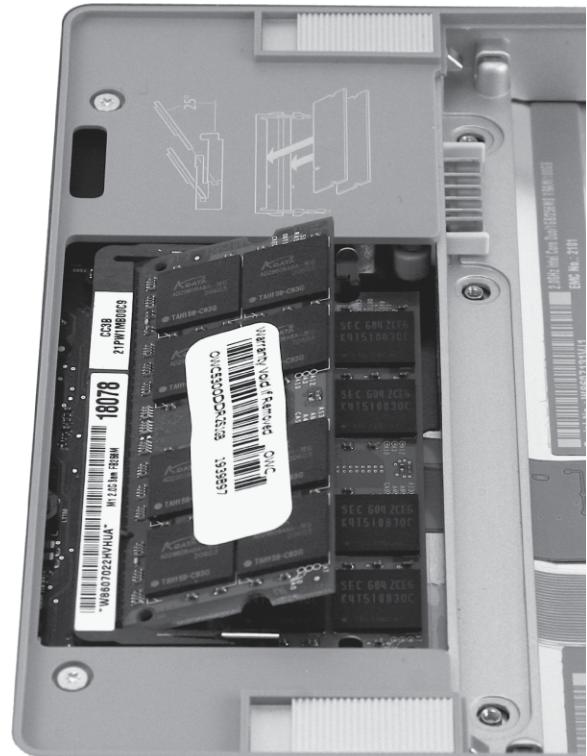
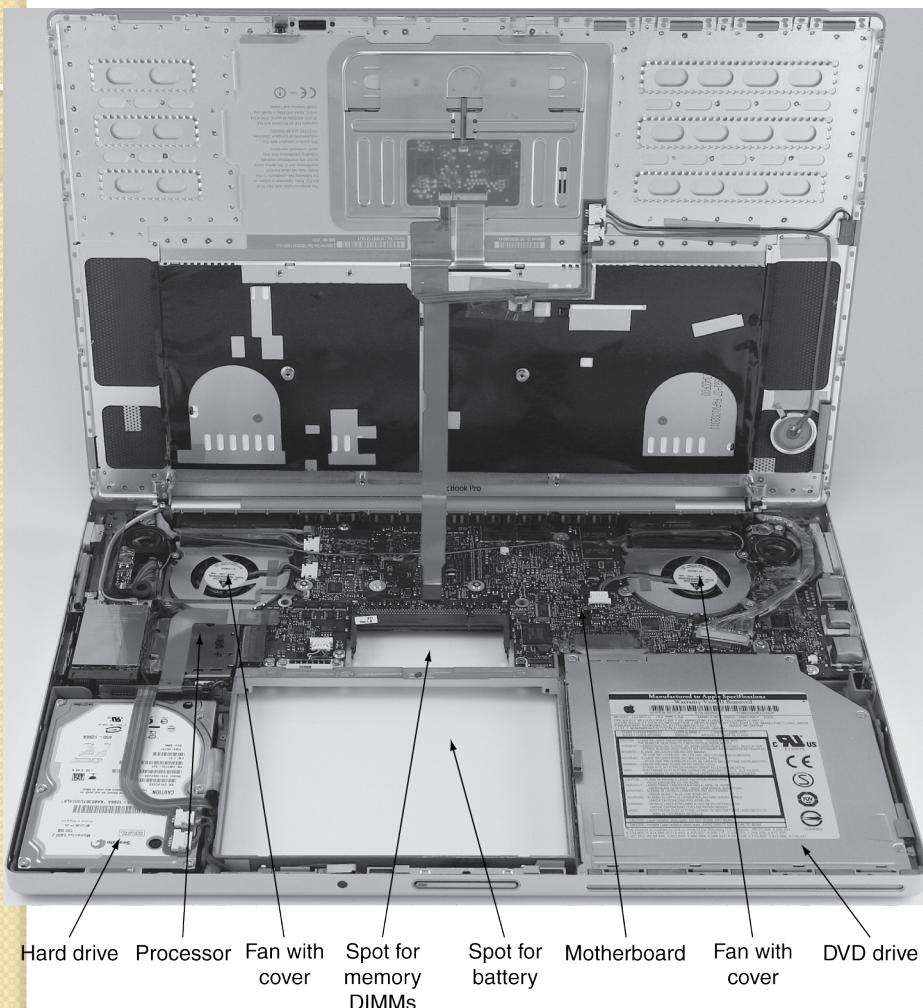


Through the Looking Glass

- LCD screen: picture elements (pixels)
 - Mirrors content of frame buffer memory



Opening the Box

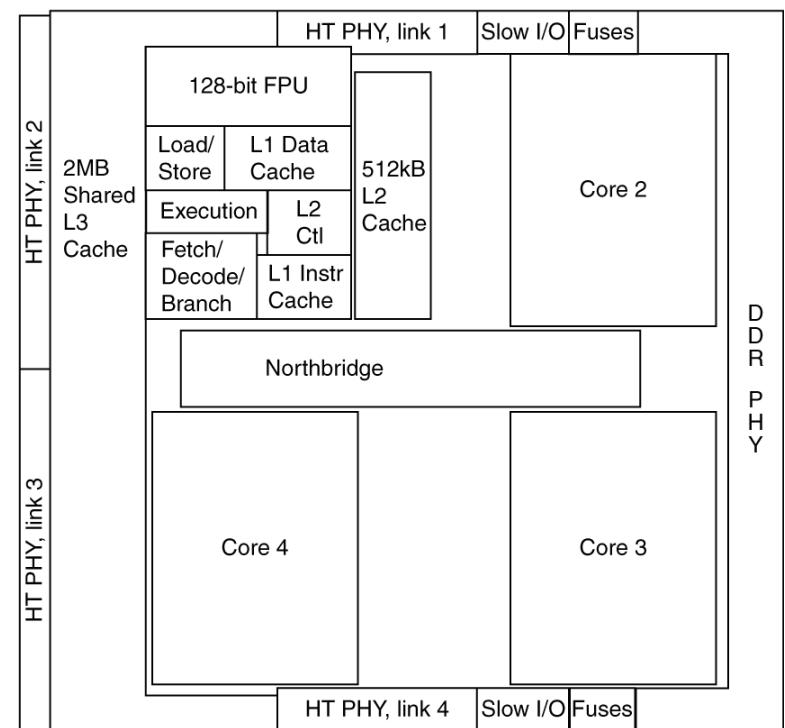
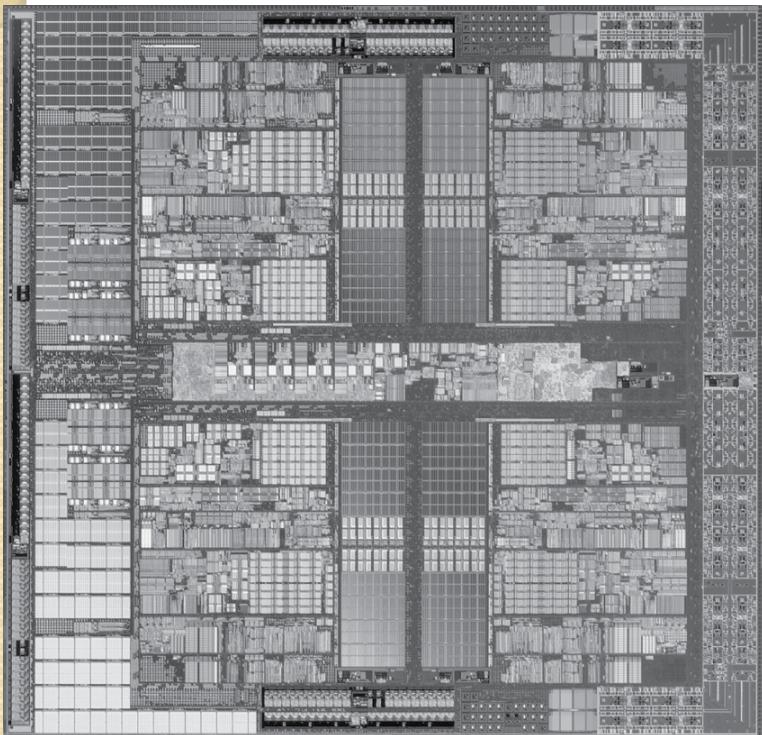


Inside the Processor (CPU)

- Datapath: performs operations on data
- Control: sequences datapath, memory, ...
- Cache memory
 - Small fast SRAM memory for immediate access to data

Inside the Processor

- AMD Barcelona: 4 processor cores



Abstractions - ISA

The BIG Picture

- Abstraction helps us deal with complexity
 - Hide lower-level detail
- Instruction set architecture (ISA)
 - The hardware/software interface
- Application binary interface
 - The ISA plus system software interface
- Implementation (Hardware obeys instruction)
 - The details underlying and interface

Define ISA

An abstract interface between the hardware and lower(st)-level software that encompasses all the information necessary to write a machine language program; that will run correctly including instructions, memory, I/O, registers, etc.

A Safe Place for Data

- Volatile main memory or primary memory
 - Loses instructions and data when power off
 - Caches – SRAM
 - DRAM
- Non-volatile secondary memory
 - Magnetic disk
 - Flash memory



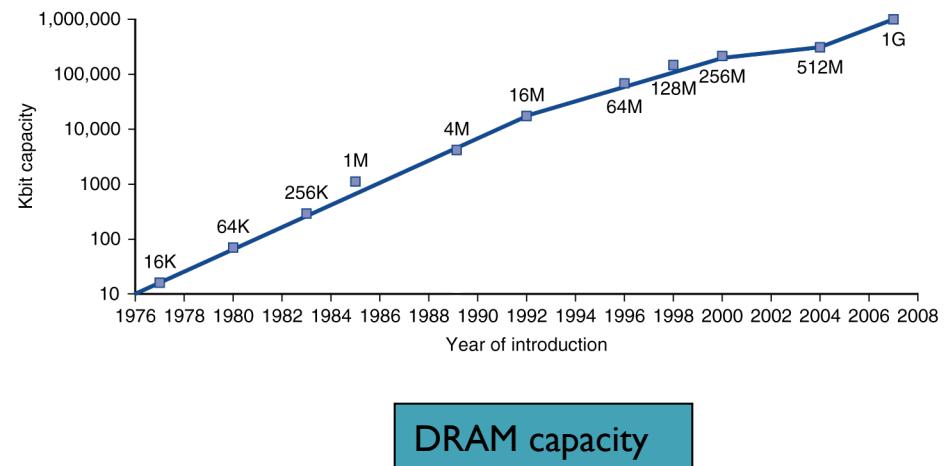
Networks

- Communication and resource sharing
- Local area network (LAN): Ethernet
 - Within a building
- Wide area network (WAN): the Internet
- Wireless network: WiFi, Bluetooth



Technology Trends

- Electronics technology continues to evolve
 - Increased capacity and performance
 - Reduced cost



Year	Technology	Relative performance/cost
1951	Vacuum tube	1
1965	Transistor	35
1975	Integrated circuit (IC)	900
1995	Very large scale IC (VLSI)	2,400,000
2005	Ultra large scale IC	6,200,000,000



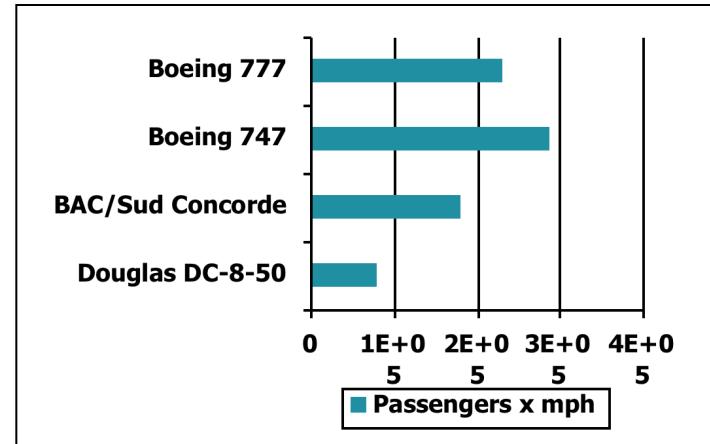
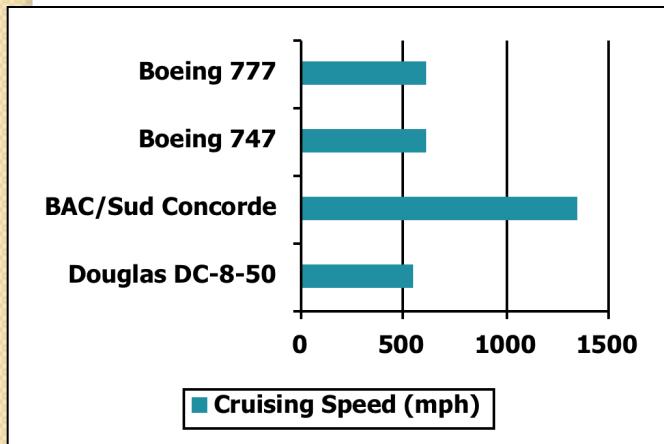
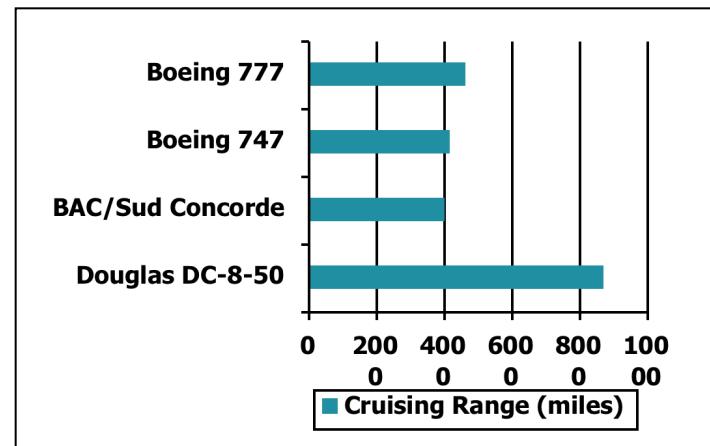
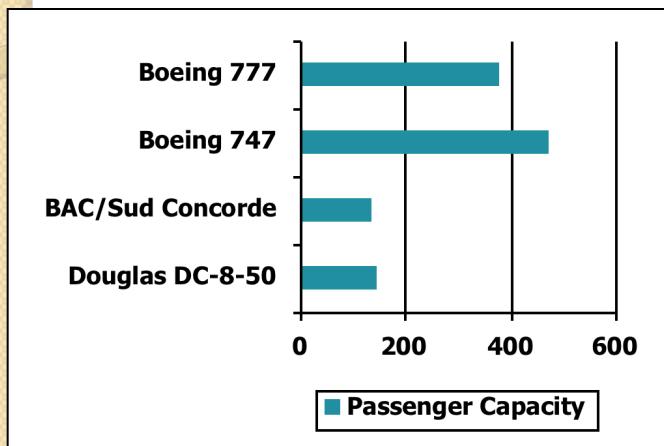
PERFORMANCE METRICS

Performance Analogy



Defining Performance

- Which airplane has the best performance?



What is important for computer performance ?

- One computer is better than another ?

Response Time and Throughput

- Response time
 - How long it takes to do a task
- Throughput
 - Total work done per unit time
 - e.g., tasks/transactions/... per hour
- How are response time and throughput affected by
 - Replacing the processor with a faster version?
 - Adding more processors?

* Thought Question: ?

When should we care about response time
and when about throughput ? Why ?

- a) Individual Desktop PC
- b) Embedded processor
- c) Google Mail Server

We'll focus on response time for now...

Relative Performance

- Define Performance = I/Execution Time
- “X is n time faster than Y”

$$\begin{aligned}\text{Performance}_x / \text{Performance}_y \\ = \text{Execution time}_y / \text{Execution time}_x = n\end{aligned}$$

Relative Performance

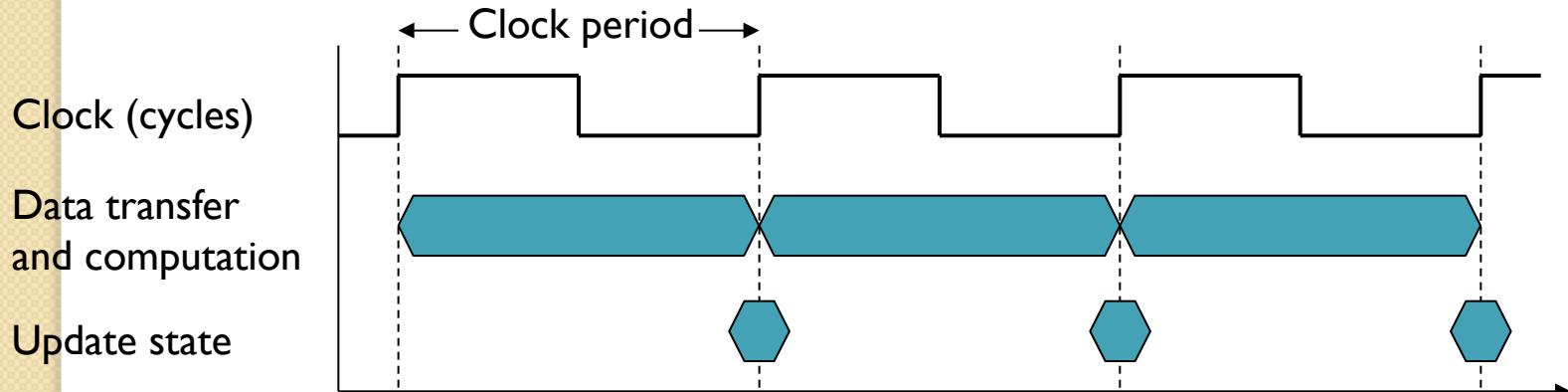
- Example: time taken to run a program
 - 10s on A, 15s on B
 - So A is **???** Times faster than B ?
- $\text{Execution Time}_B / \text{Execution Time}_A = 15s / 10s = 1.5$
- So A is 1.5 times faster than B

Measuring Execution Time

- Elapsed time/Wall clock/Response time
 - Total response time, including all aspects
 - Processing, I/O, OS overhead, idle time
 - Determines system performance
- CPU time
 - Time spent processing a given job
 - Discounts I/O time, other jobs' shares
 - Comprises user CPU time and system CPU time
 - Different programs are affected differently by CPU and system performance

CPU Clocking

- Operation of digital hardware governed by a constant-rate clock.
- Clock determines when events take place.



- **Clock period:** duration of a clock cycle
 - e.g., $250\text{ps} = 0.25\text{ns} = 250 \times 10^{-12}\text{s}$
- **Clock frequency (rate):** cycles per second
 - e.g., $4.0\text{GHz} = 4000\text{MHz} = 4.0 \times 10^9\text{Hz}$

CPU Time

CPU Time = CPU Clock Cycles × Clock Cycle Time

$$= \frac{\text{CPU Clock Cycles}}{\text{Clock Rate}}$$

- Performance improved by
 - Reducing number of clock cycles.
 - Increasing clock rate.
 - Hardware designer must often trade off either the number of clock cycles required for a program or the length of a clock cycle.

CPU Time Example

- Computer A: 2GHz clock, 10s CPU time
- Designing Computer B
 - Aim for 6s CPU time
 - Can do faster clock, but causes $1.2 \times$ clock cycles
- How fast must Computer B clock be?

$$\text{Clock Rate}_B = \frac{\text{Clock Cycles}_B}{\text{CPU Time}_B} = \frac{1.2 \times \text{Clock Cycles}_A}{6s}$$

$$\begin{aligned}\text{Clock Cycles}_A &= \text{CPU Time}_A \times \text{Clock Rate}_A \\ &= 10s \times 2\text{GHz} = 20 \times 10^9\end{aligned}$$

$$\text{Clock Rate}_B = \frac{1.2 \times 20 \times 10^9}{6s} = \frac{24 \times 10^9}{6s} = 4\text{GHz}$$

Instruction Count and CPI

Clock Cycles = Instruction Count \times Cycles per Instruction

CPU Time = Instruction Count \times CPI \times Clock Cycle Time

$$= \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock Rate}}$$

- Instruction Count for a program
 - Determined by program, ISA and compiler
- Average cycles per instruction
 - Determined by CPU hardware
 - If different instructions have different CPI
 - Average CPI affected by instruction mix

Instruction Count and CPI

Clock Cycles = Instruction Count × Cycles per Instruction

CPU Time = Instruction Count × CPI × Clock Cycle Time

$$= \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock Rate}}$$

- Instruction Count for a program
 - Determined by ?, ? And ?
- Average cycles per instruction
 - Determined by ?

CPI Example

- Computer A: Cycle Time = 250ps, CPI = 2.0
- Computer B: Cycle Time = 500ps, CPI = 1.2
- Same ISA
- Which is faster, and by how much?

$$\text{CPU Time}_A = \text{Instruction Count} \times \text{CPI}_A \times \text{Cycle Time}_A$$

$$= I \times 2.0 \times 250\text{ps} = I \times 500\text{ps}$$

A is faster...

$$\text{CPU Time}_B = \text{Instruction Count} \times \text{CPI}_B \times \text{Cycle Time}_B$$

$$= I \times 1.2 \times 500\text{ps} = I \times 600\text{ps}$$

$$\frac{\text{CPU Time}_B}{\text{CPU Time}_A} = \frac{I \times 600\text{ps}}{I \times 500\text{ps}}$$

$$= 1.2$$

...by this much

Extra Exercise #1

- Refer to Handout!

CPI Example [Instruction Mix]

- Alternative compiled code sequences using instructions in classes A, B, C

Class	A	B	C
CPI for class	1	2	3
Instruction Count in sequence 1	2	1	2
Instruction Count in sequence 2	4	1	1

- Sequence 1: IC = 5
 - Clock Cycles
 $= 2 \times 1 + 1 \times 2 + 2 \times 3$
 $= 10$
 - Avg. CPI = $10/5 = 2.0$
- Sequence 2: IC = 6
 - Clock Cycles = ???
 - Avg. CPI = ???

CPI in More Detail

- If different instruction classes take different numbers of cycles

$$\text{Clock Cycles} = \sum_{i=1}^n (\text{CPI}_i \times \text{Instruction Count}_i)$$

- Weighted average CPI

$$\text{CPI} = \frac{\text{Clock Cycles}}{\text{Instruction Count}} = \sum_{i=1}^n \left(\text{CPI}_i \times \frac{\text{Instruction Count}_i}{\text{Instruction Count}} \right)$$

Relative frequency

Extra Exercise 2

- Refer to handout!

Minimum CPI (Discussion)

- What is the minimum CPI ?
- Can we have a CPI of 0.5?

Today ..

- Recap: Performance
 - In-class exercises
- Power Trends
- Multiprocessors
- Manufacturing Integrated Circuits
- SPEC Benchmark
- Million Instructions Per Seconds (MIPS)

Performance Summary

The BIG Picture

$$\text{CPU Time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

- Performance depends on
 - Algorithm: affects IC, possibly CPI
 - Programming language: affects IC, CPI
 - Compiler: affects IC, CPI
 - Instruction set architecture: affects IC, CPI, clock rate

CPU Execution Time ?

CPU Time

Clock Cycles = Instruction Count × Cycles per Instruction

CPU Time = Instruction Count × CPI × Clock Cycle Time

$$= \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock Rate}}$$

Exercise I from Handout

Computer A has an overall CPI of 1.3 and can be run at a clock rate of 600MHz. Computer B has a CPI of 2.5 and can be run at a clock rate of 750 Mhz. We have a particular program we wish to run. When compiled for computer A, this program has exactly 100,000 instructions. How many instructions would the program need to have when compiled for Computer B, in order for the two computers to have exactly the same execution time for this program?

Ex 2

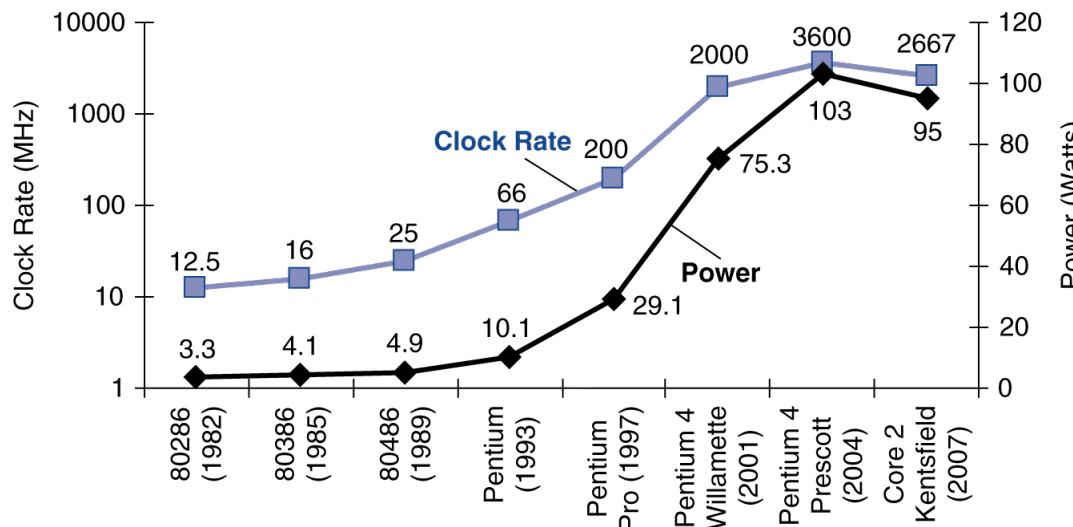
	A	B	C	D	Clock Rate
P1	1	2	3	4	1.5GHz
P2	2	2	2	2	2.0GHz

- A) Given a program with $10E+6$ instructions divided into classes as follows: 10% of class A, 20% of class B, 50% of class C and 20% of class D, which implementation is faster ?
- B) What is the global CPI for each implementation ?
- C) Find the number of clock cycles needed in each case.

Minimum CPI (Discussion)

- What is the minimum CPI ?
- Can we have a CPI of 0.5?

Power Trends



- In CMOS IC technology

$$\text{Power} = \text{Capacitive load} \times \text{Voltage}^2 \times \text{Frequency}$$

×30

5V → 1V

×1000

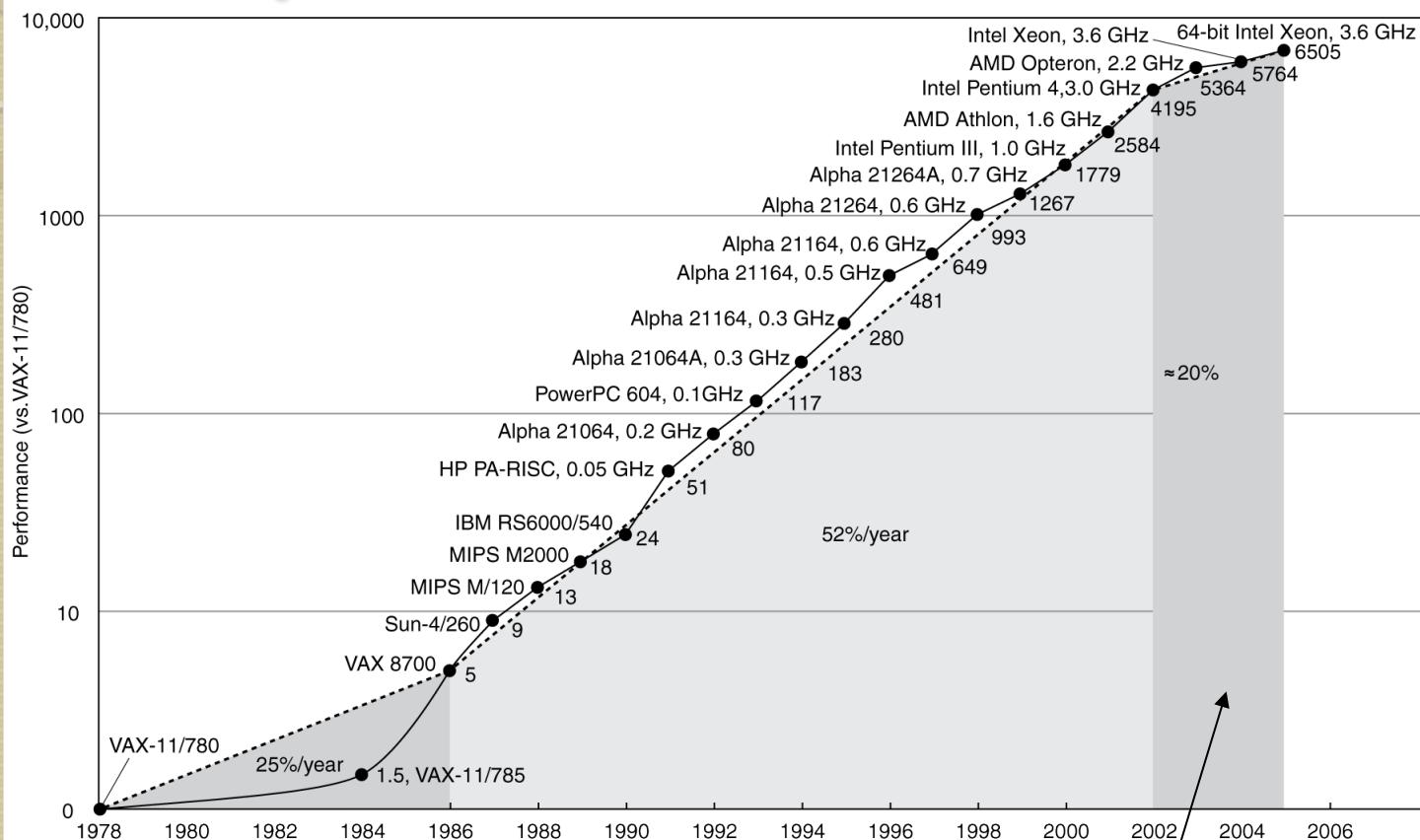
Reducing Power (WPS)

- Suppose a new CPU has
 - 85% of capacitive load of old CPU
 - 15% voltage and 15% frequency reduction
 - What will be the ratio of new power to old power

$$\frac{P_{\text{new}}}{P_{\text{old}}} = \frac{C_{\text{old}} \times 0.85 \times (V_{\text{old}} \times 0.85)^2 \times F_{\text{old}} \times 0.85}{C_{\text{old}} \times V_{\text{old}}^2 \times F_{\text{old}}} = 0.85^4 = 0.52$$

- The power wall
 - We can't reduce voltage further
 - We can't remove more heat
- How else can we improve performance?

Uniprocessor Performance



Constrained by power, instruction-level parallelism, memory latency



How else can we improve performance?

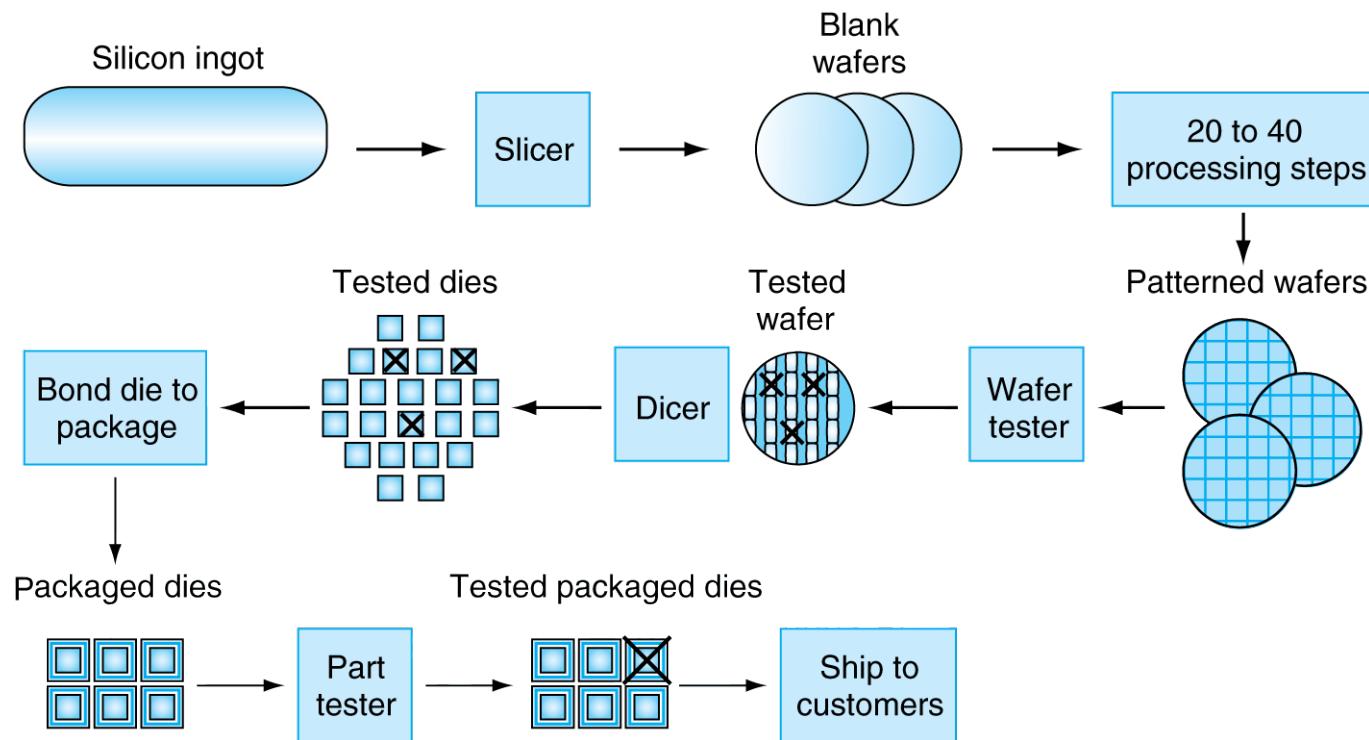
Multiprocessors

- Multicore microprocessors
 - More than one processor per chip
- What do you think are the challenges of dealing with multiprocessors ?

Challenges of Multicore Multiprocessors.

- Requires explicitly parallel programming
 - Compare with instruction level parallelism
 - Hardware executes multiple instructions at once
 - Hidden from the programmer
 - Hard to do
 - Programming for performance
 - Load balancing
 - Optimizing communication and synchronization

Manufacturing IC (FYI)



- Yield: proportion of working dies per wafer

Workload & Benchmark

Define **Workload**: A set of programs run on a computer that is either the actual collection of programs run by a user or constructed from real programs to approximate such a mix. A typical workload specifies both the programs and the relative frequencies.

Define **Benchmark**: A program selected for use in comparing computer performance.

SPEC CPU Benchmark

- Programs used to measure performance
 - Supposedly typical of actual workload
- Standard Performance Evaluation Corp (SPEC)
 - Develops benchmarks for CPU, I/O, Web, ...
- SPEC CPU2006
 - Elapsed time to execute a selection of programs
 - Negligible I/O, so focuses on CPU performance
 - Normalize relative to reference machine
 - Summarize as geometric mean of performance ratios
 - CINT2006 (integer) and CFP2006 (floating-point)

$$\sqrt[n]{\prod_{i=1}^n \text{Execution time ratio}_i}$$

CINT2006 for Opteron X4 2356

Name	Description	IC×10 ⁹	CPI	Tc (ns)	Exec time	Ref time	SPECratio
perl	Interpreted string processing	2,118	0.75	0.40	637	9,777	15.3
bzip2	Block-sorting compression	2,389	0.85	0.40	817	9,650	11.8
gcc	GNU C Compiler	1,050	1.72	0.47	24	8,050	11.1
mcf	Combinatorial optimization	336	10.00	0.40	1,345	9,120	6.8
go	Go game (AI)	1,658	1.09	0.40	721	10,490	14.6
hmmer	Search gene sequence	2,783	0.80	0.40	890	9,330	10.5
sjeng	Chess game (AI)	2,176	0.96	0.48	37	12,100	14.5
libquantum	Quantum computer simulation	1,623	1.61	0.40	1,047	20,720	19.8
h264avc	Video compression	3,102	0.80	0.40	993	22,130	22.3
omnetpp	Discrete event simulation	587	2.94	0.40	690	6,250	9.1
astar	Games/path finding	1,082	1.79	0.40	773	7,020	9.1
xalancbmk	XML parsing	1,058	2.70	0.40	1,143	6,900	6.0
Geometric mean							11.7

High cache miss rates

SPEC Power Benchmark

- Power consumption of server at different workload levels
 - Performance: ssj_ops/sec (# of operations per second)
 - Power: Watts (Joules/sec)

$$\text{Overall ssj_ops per Watt} = \left(\sum_{i=0}^{10} \text{ssj_ops}_i \right) \Bigg/ \left(\sum_{i=0}^{10} \text{power}_i \right)$$

SPECpower_ssj2008 for X4

Target Load %	Performance (ssj_ops/sec)	Average Power (Watts)
100%	231,867	295
90%	211,282	286
80%	185,803	275
70%	163,427	265
60%	140,160	256
50%	118,324	246
40%	920,35	233
30%	70,500	222
20%	47,126	206
10%	23,066	180
0%	0	141
Overall sum	1,283,590	2,605
$\Sigma ssj_ops / \Sigma power$		493

Pitfall: Amdahl's Law

- Improving an aspect of a computer and expecting a proportional improvement in overall performance

$$T_{\text{improved}} = \frac{T_{\text{affected}}}{\text{improvement factor}} + T_{\text{unaffected}}$$

- Example: multiply accounts for 80s/100s
 - How much improvement in multiply performance to get 5× overall?

Pitfall: Amdahl's Law

- Improving an aspect of a computer and expecting a proportional improvement in overall performance

$$T_{\text{improved}} = \frac{T_{\text{affected}}}{\text{improvement factor}} + T_{\text{unaffected}}$$

- Example: multiply accounts for 80s/100s
 - How much improvement in multiply performance to get 5× overall?

$$20 = \frac{80}{n} + 20$$

- Can't be done!

- Corollary: make the common case fast

Fallacy: Low Power at Idle

- Look back at X4 power benchmark
 - At 100% load: 295W
 - At 50% load: 246W (83%)
 - At 10% load: 180W (61%)
- Google data center
 - Mostly operates at 10% – 50% load
 - At 100% load less than 1% of the time
- Consider designing processors to make power proportional to load

MIPS as a Performance Metric

- MIPS: Millions of Instructions Per Second

$$\begin{aligned}\text{MIPS} &= \frac{\text{Instruction count}}{\text{Execution time} \times 10^6} \\ &= \frac{\text{Instruction count}}{\frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}} \times 10^6} = \frac{\text{Clock rate}}{\text{CPI} \times 10^6}\end{aligned}$$

Exercise:

Measurement	Computer A	Computer B
Instruction Count	10 billion	8 billion
Clock Rate	4 Ghz	4 Ghz
CPI	1.0	1.1

- A) Which computer is faster ?
- B) Which computer has higher MIPS ratings ?

Exercise:

Measurement	Computer A	Computer B
Instruction Count	10 billion	8 billion
Clock Rate	4 Ghz	4 Ghz
CPI	1.0	1.1

- A) Which computer is faster ? --- Computer B
- B) Which computer has higher MIPS ratings ? --- Computer A

Pitfall: MIPS as a Performance Metric

- MIPS: Millions of Instructions Per Second
 - Doesn't account for
 - Differences in ISAs between computers
 - Differences in complexity between instructions

$$\begin{aligned}\text{MIPS} &= \frac{\text{Instruction count}}{\text{Execution time} \times 10^6} \\ &= \frac{\text{Instruction count}}{\frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}} \times 10^6} = \frac{\text{Clock rate}}{\text{CPI} \times 10^6}\end{aligned}$$

- CPI varies between programs on a given CPU

Concluding Remarks

- Cost/performance is improving
 - Due to underlying technology development
- Hierarchical layers of abstraction
 - In both hardware and software
- Instruction set architecture
 - The hardware/software interface
- Execution time: the best performance measure
- Power is a limiting factor
 - Use parallelism to improve performance