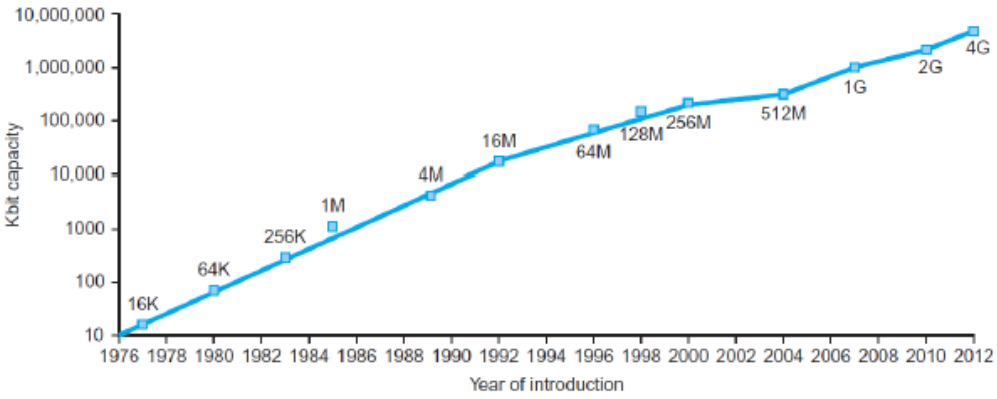
COMPUTER ARCHITECTURE

MEASURING PERFORMANCE

COMPUTER TECHNOLOGY

- Rapidly changing field:
  - vacuum tube -> transistor -> IC -> VLSI
  - doubling every 1.5 years:
    - memory capacity
    - processor speed (due to advances in technology and hardware organization)
  - cute example: if Boeing had kept up with IBM we could *fly from Bangkok to Newyork City in 10 minutes*
  - Things we'll be learning:
  - how computers work, what's a good design, what's not
  - issues affecting modern processors (e.g., caches, pipelines)

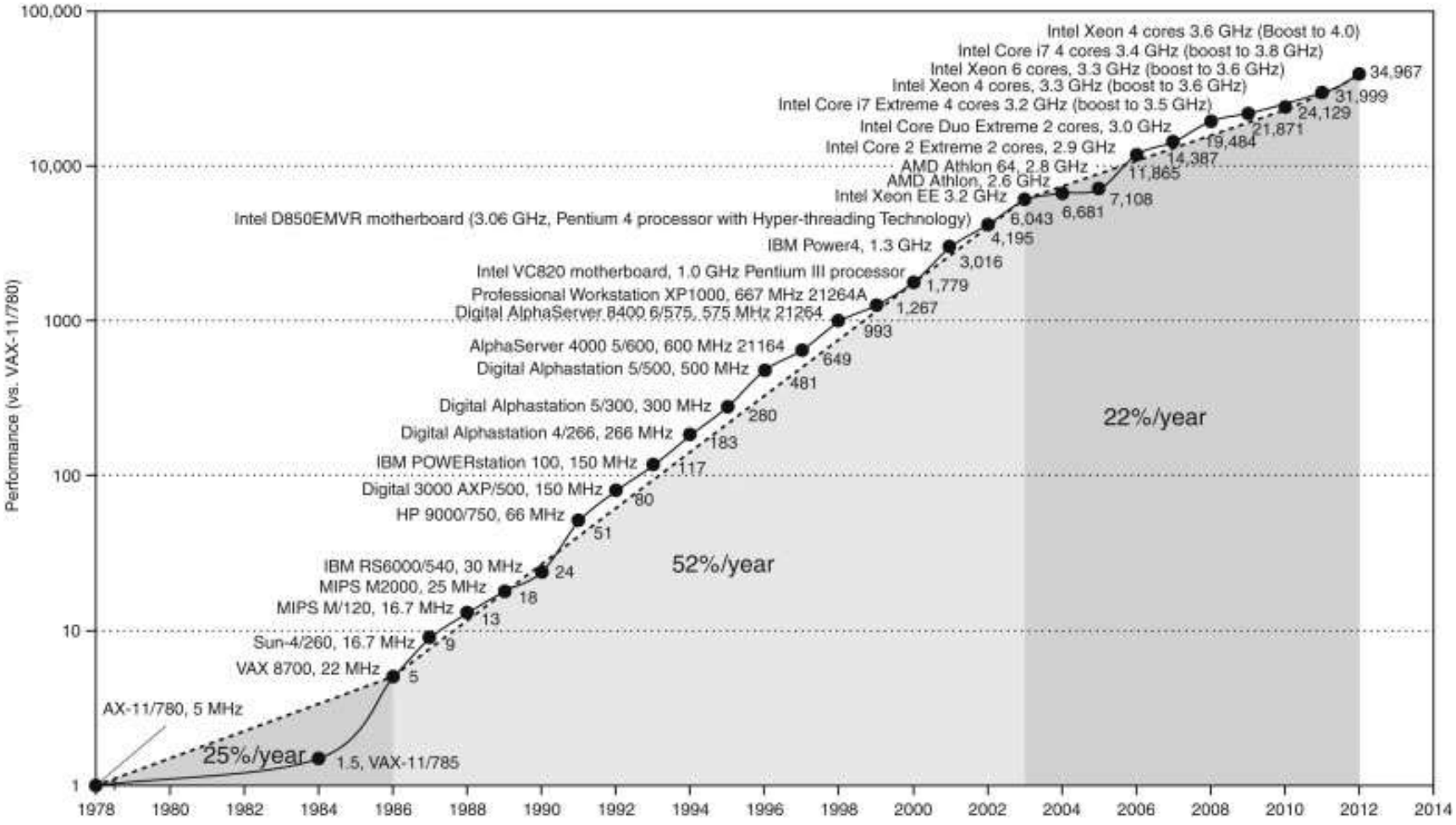
- Electronics technology continues to evolve
  - Increased capacity and performance
  - Reduced cost



DRAM capacity

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
2013	Ultra large scale IC	250,000,000,000

POWER WALL


Performance (vs. VAX-11/780)
100,000
10,000
1000
100
10
1
1978
1980
1982
1984
1986
1988
1990
1992
1994
1996
1998
2000
2002
2004
2006
2008
2010
2012
2014
Intel Xeon 4 cores 3.6 GHz (Boost to 4.0)
Intel Core i7 4 cores 3.4 GHz (boost to 3.8 GHz)
Intel Xeon 6 cores, 3.3 GHz (boost to 3.6 GHz)
Intel Xeon 4 cores, 3.3 GHz (boost to 3.6 GHz)
Intel Core i7 Extreme 4 cores 3.2 GHz (boost to 3.5 GHz)
Intel Core Duo Extreme 2 cores, 3.0 GHz
Intel Core 2 Extreme 2 cores, 2.9 GHz
AMD Athlon 64, 2.8 GHz
AMD Athlon, 2.6 GHz
Intel Xeon EE 3.2 GHz
Intel D850EMVR motherboard (3.06 GHz, Pentium 4 processor with Hyper-threading Technology)
IBM Power4, 1.3 GHz
Intel VC820 motherboard, 1.0 GHz Pentium III processor
Professional Workstation XP1000, 667 MHz 21264A
Digital AlphaServer 8400 6/575, 575 MHz 21264
AlphaServer 4000 5/600, 600 MHz 21164
Digital Alphastation 5/500, 500 MHz
Digital Alphastation 5/300, 300 MHz
Digital Alphastation 4/266, 266 MHz
IBM POWERstation 100, 150 MHz
Digital 3000 AXP/500, 150 MHz
HP 9000/750, 66 MHz
IBM RS6000/540, 30 MHz
MIPS M2000, 25 MHz
MIPS M/120, 16.7 MHz
Sun-4/260, 16.7 MHz
VAX 8700, 22 MHz
AX-11/780, 5 MHz
34,967
31,999
24,129
21,871
19,484
14,387
11,865
7,108
6,681
6,043
4,195
3,016
1,779
1,267
993
649
481
280
183
117
80
51
24
18
13
9
5
1.5, VAX-11/785
22%/year
52%/year
25%/year

MICROPROCESSOR PERFORMANCE

- Running out of ideas to improve single thread performance
- Power wall makes it harder to add complex features
- Power wall makes it harder to increase frequency
- Additional performance provided by: more cores, occasional spikes in frequency, accelerators

PERFORMANCE

- Performance is the key to understanding underlying motivation for the hardware and its organization
- Measure, report, and summarize performance to enable users to
  - make intelligent choices
  - see through the marketing hype!
- Why is some hardware better than others for different programs?
- What factors of system performance are hardware related? (e.g., do we need a new machine, or a new operating system?)
- How does the machine's instruction set affect performance?

WHAT DO WE MEASURE? DEFINE PERFORMANCE….

Airplane	Passengers	Range (mi)	Speed (mph)
Boeing 737-100	101	630	598
Boeing 747	470	4150	610
BAC/Sud Concorde	132	4000	1350
Douglas DC-8-50	146	8720	544

- How much faster is the Concorde compared to the 747?
- How much bigger is the Boeing 747 than the Douglas DC-8?

- *So which of these airplanes has the best performance?!*

COMPUTER PERFORMANCE: TIME, TIME, TIME!!!

- ***Response Time (elapsed time, latency):***
  - how long does it take for *my* job to run?
  - how long does it take to execute (start to
  - finish) *my* job?
  - how long must *I* wait for the database query?

Individual user concerns...

- ***Throughput:***
  - how *many* jobs can the machine run at once?
  - what is the *average* execution rate?
  - how *much* work is getting done?

Systems manager concerns…

- *If we upgrade to a machine with a new processor what do we increase?*
- *If we add a new machine to the lab what do we increase?*

Performance

Wall Clock Time

Elapsed Time

- counts everything from start to finish
- a useful number, but often not good for comparison purposes
- **elapsed time = CPU time + wait time**

CPU Time

- I/O or time spent running other programs not counted
- CPU time = user CPU time + system CPU time

User CPU Time

- CPU execution time or, simply, execution time
- **Time spent executing the lines of code that are in our program**

DEFINITION OF PERFORMANCE

- For some program running on machine X:

$$\text{Performance}_X = 1 / \text{Execution time}_X$$

- *X is n times faster than Y* means:

$$\text{Performance}_X \ / \ \text{Performance}_Y = n$$

CLOCK CYCLES

- Instead of reporting execution time in seconds, we often use *cycles*. In modern computers hardware events progress cycle by cycle: in other words, each event, e.g., multiplication, addition, etc., is a sequence of cycles
- *Clock ticks* indicate start and end of cycles
- *cycle time* = time between ticks = seconds per cycle
- *clock rate* (*frequency*) = cycles per second
- (1 Hz. = 1 cycle/sec, 1 MHz. = 10^6 cycles/sec)
- *Example*: A 200 MHz. clock has a cycle time:

$$\frac{1}{200 \times 10^6} = 5 \times 10^{-9} \text{seconds}$$

PERFORMANCE EQUATION I

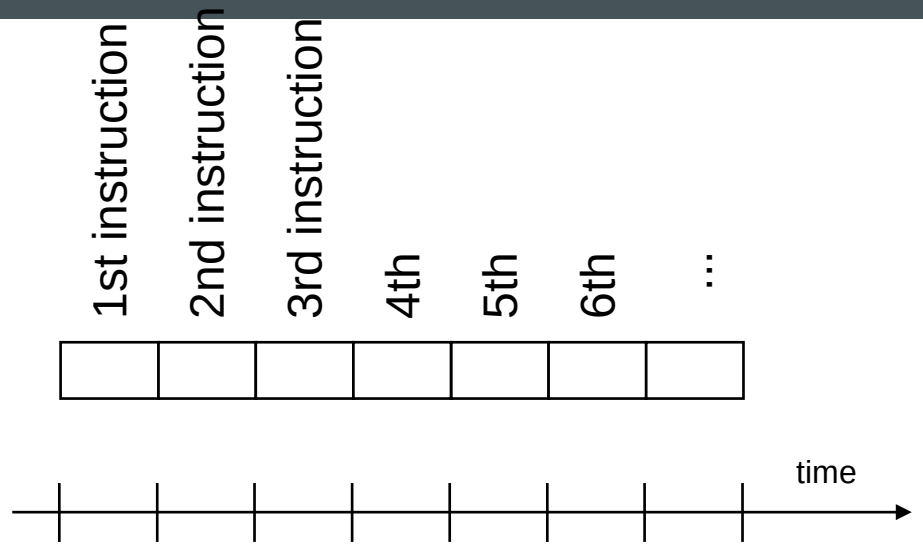
$$\frac{\text{seconds}}{\text{program}} = \frac{\text{cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{cycle}}$$

equivalently

CPU execution time for a program = CPU clock cycles for a program × Clock cycle time

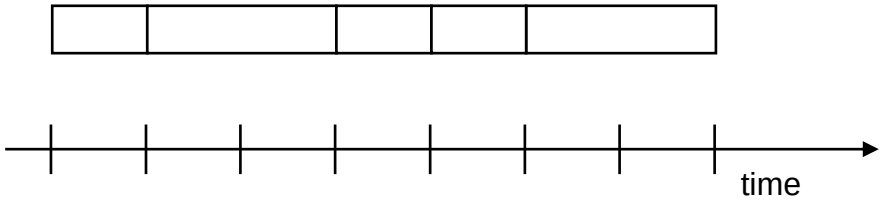
- So, to improve performance one can either:
  - reduce the number of cycles for a program, or
  - reduce the clock cycle time, or, equivalently,
  - increase the clock rate

HOW MANY CYCLES ARE REQUIRED FOR A PROGRAM?



- Could assume that # of cycles = # of instructions
  - *This assumption is incorrect!* Because:
    - Different instructions take different amounts of time (cycles)
    - Why…?

HOW MANY CYCLES ARE REQUIRED FOR A PROGRAM?



- Multiplication takes more time than addition
- Floating point operations take longer than integer ones
- Accessing memory takes more time than accessing register

EXAMPLE

- Our favorite program runs in 10 seconds on computer A, which has a 400MHz. clock. Calculate the number of clock cycles taken by the program.

EXAMPLE

- Our favorite program runs in 10 seconds on computer A, which has a 400MHz. clock. Calculate the number of clock cycles.
- We are trying to help a computer designer build a new machine B, that will run this program in 6 seconds. The designer can use new (or perhaps more expensive) technology to substantially increase the clock rate, but has informed us that this increase will affect the rest of the CPU design, causing machine B to require 1.2 times as many clock cycles as machine A for the same program.

- *What clock rate should we tell the designer to target?*

SOLUTION

Assume the program takes X cycles and the desired rate is Y

(X/ 400/10^6) / (1.2X/ Y)

= 10/6

therefore Y = 800 MHz

TERMINOLOGY

- A given program will require:
  - some number of instructions (machine instructions)
  - some number of cycles
  - some number of seconds
- We have a vocabulary that relates these quantities:
  - *cycle time* (seconds per cycle)
  - *clock rate* (cycles per second)
  - **(*average*) *CPI* (cycles per instruction)**
    - a floating point intensive application might have a higher average CPI

PERFORMANCE EQUATION II

CPU execution time for a program = Instruction count for a program × average CPI × Clock

- *Derive the above equation from Performance Equation I*
- CPU execution time = IC * CPI * Clock cycles
  - IC : Instruction count
  - *CPI : Clock cycles per instruction*

CPI EXAMPLE 1

- Suppose we have two computers
  - Computer A has a clock cycle time of 250 ps and a CPI of 2.0 for some program
  - Computer B has a clock cycle time of 500 ps and a CPI of 1.2 for the same program.
- Which machine is faster for this program, and by how much?

SOLUTION

We know that each computer executes the same number of instructions for the program; let's call this number I. First, find the number of processor clock cycles for each computer:

$$\text{CPU clock cycles}_A = I \times 2.0$$
$$\text{CPU clock cycles}_B = I \times 1.2$$

Now we can compute the CPU time for each computer:

$$\begin{aligned}\text{CPU time}_A &= \text{CPU clock cycles}_A \times \text{Clock cycle time} \\ &= I \times 2.0 \times 250 \text{ ps} = 500 \times I \text{ ps}\end{aligned}$$

Likewise, for B:

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

Clearly, computer A is faster. The amount faster is given by the ratio of the execution times:

$$\frac{\text{CPU performance}_A}{\text{CPU performance}_B} = \frac{\text{Execution time}_B}{\text{Execution time}_A} = \frac{600 \times I \text{ ps}}{500 \times I \text{ ps}} = 1.2$$

We can conclude that computer A is 1.2 times as fast as computer B for this program.

AMDAHL'S LAW

- the performance improvement to be gained from using some faster mode of execution is limited by the **fraction of the time** the faster

$$\text{Speedup} = \frac{\text{Performance for entire task using the enhancement when possible}}{\text{Performance for entire task without using the enhancement}}$$

Alternatively,

$$\text{Speedup} = \frac{\text{Execution time for entire task without using the enhancement}}{\text{Execution time for entire task using the enhancement when possible}}$$

AMDAHL'S LAW

- Performance Improvement depends on two factors:
  - The fraction of the computation time in the original computer that can be converted to take advantage of the enhancement
  - The improvement gained by the enhanced execution mode, that is, how much faster the task would run if the enhanced mode were used for the entire program

$$\text{Execution time after improvement} = \frac{\text{Execution time affected by improvement}}{\text{Amount of improvement}} + \text{Execution time unaffected}$$

AMDAHL'S LAW

$$\text{Execution time}_{\text{new}} = \text{Execution time}_{\text{old}} \times \left((1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}} \right)$$

The overall speedup is the ratio of the execution times:

$$\text{Speedup}_{\text{overall}} = \frac{\text{Execution time}_{\text{old}}}{\text{Execution time}_{\text{new}}} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

EXAMPLE

Let a program have 40 percent of its code enhanced to run 2.3 times faster. What is the overall system speedup S?

SOLUTION

Let a program have 40 percent of its code enhanced to run 2.3 times faster. What is the overall system speedup S?

$\text{Fraction}_{\text{Enhanced}} = 0.4$

$\text{Speedup}_{\text{Enhanced}} = 2.3$

$$\text{Speedup}_{\text{Overall}} = \frac{1}{(1-0.4)+\frac{0.4}{2.3}} = 1.292$$

COMMON CASE: AMDAHL'S LAW

- *Example*:
  - Suppose a program runs in 100 seconds on a machine, with multiply responsible for 80 seconds of this time.
  - *How much do we have to improve the speed of multiplication if we want the program to run 5 times faster?*

AMDAHL'S LAW

$$\text{Execution time after improvement} = \frac{80 \text{ seconds}}{n} + (100 - 80 \text{ seconds})$$

Since we want the performance to be five times faster, the new execution time should be 20 seconds, giving

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

$$0 = \frac{80 \text{ seconds}}{n}$$

- Design Principle: *Make the common case fast*

PRACTICE

- Suppose we enhance a machine making all floating-point instructions run five times faster. The execution time of some benchmark before the floating-point enhancement is 10 seconds.
  - *What will the speedup be if half of the 10 seconds is spent executing floating-point instructions?*
- We are looking for a benchmark to show off the new floating-point unit described above, and want the overall benchmark to show a speedup of 3. One benchmark we are considering runs for 100 seconds with the old floating-point hardware.
  - *How much of the execution time would floating-point instructions have to account for in this program in order to yield our desired speedup on this benchmark?*

SUMMARY

- Performance specific to a particular program
  - total execution time is a consistent summary of performance
- For a given architecture performance increases come from:
  - increases in clock rate
  - improvements in processor organization that lower CPI
  - compiler enhancements that lower CPI and/or instruction count
- expecting improvement in one aspect of a machine's performance to affect the total performance

NEXT CLASS …

- Number Representation
- **Before next class: Revise**
  - Binary Representation
  - Binary to Decimal and Decimal to Binary conversion
  - Binary Addition