Course Introduction; Chapter 1

- Rapidly changing field:
  - Vacuum tube -> transistor -> IC -> VLSI
  - Doubling every 1.5 years:
    - Memory capacity.
    - Processor Performance (Due to advances in technology and organization).

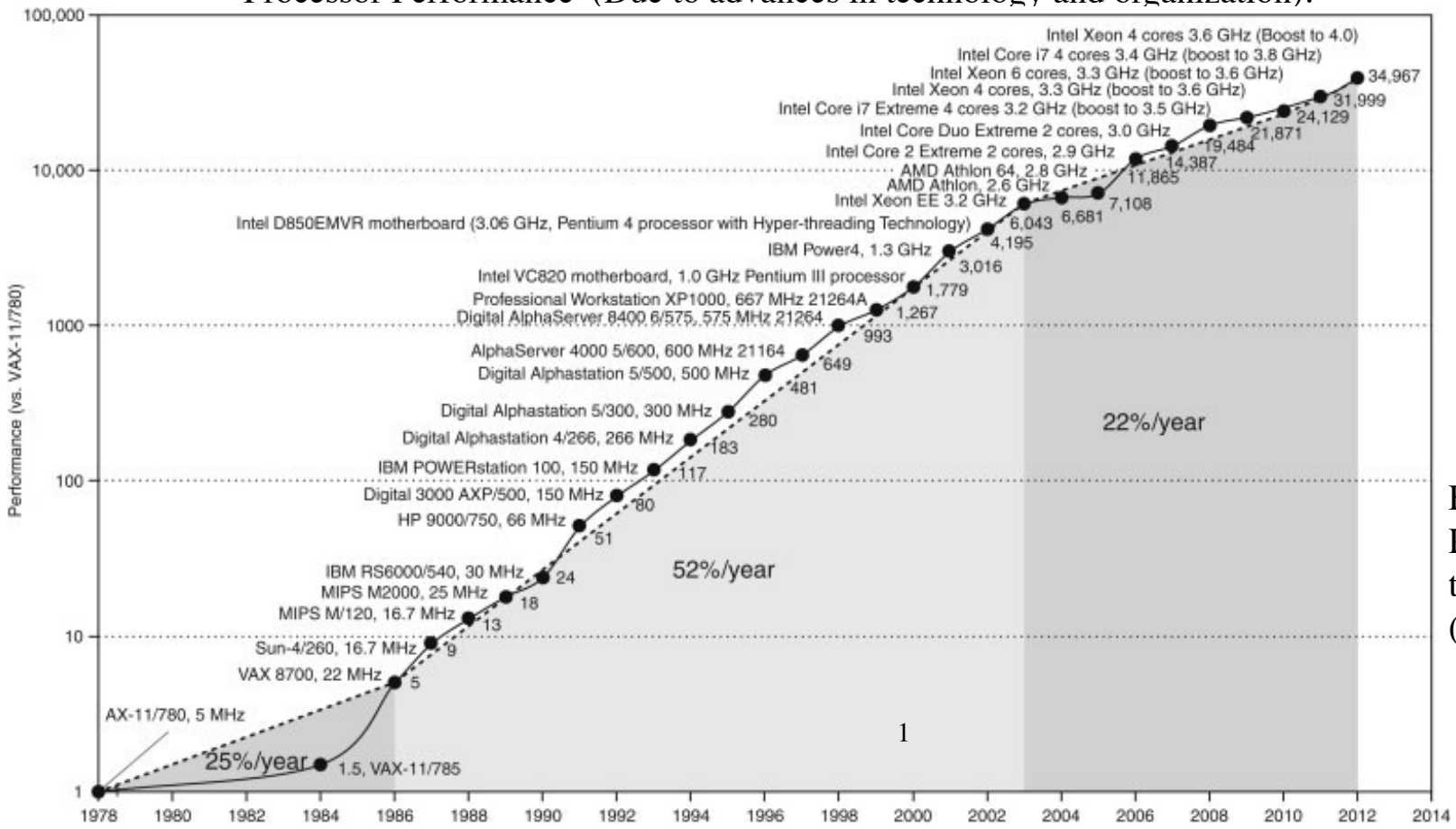
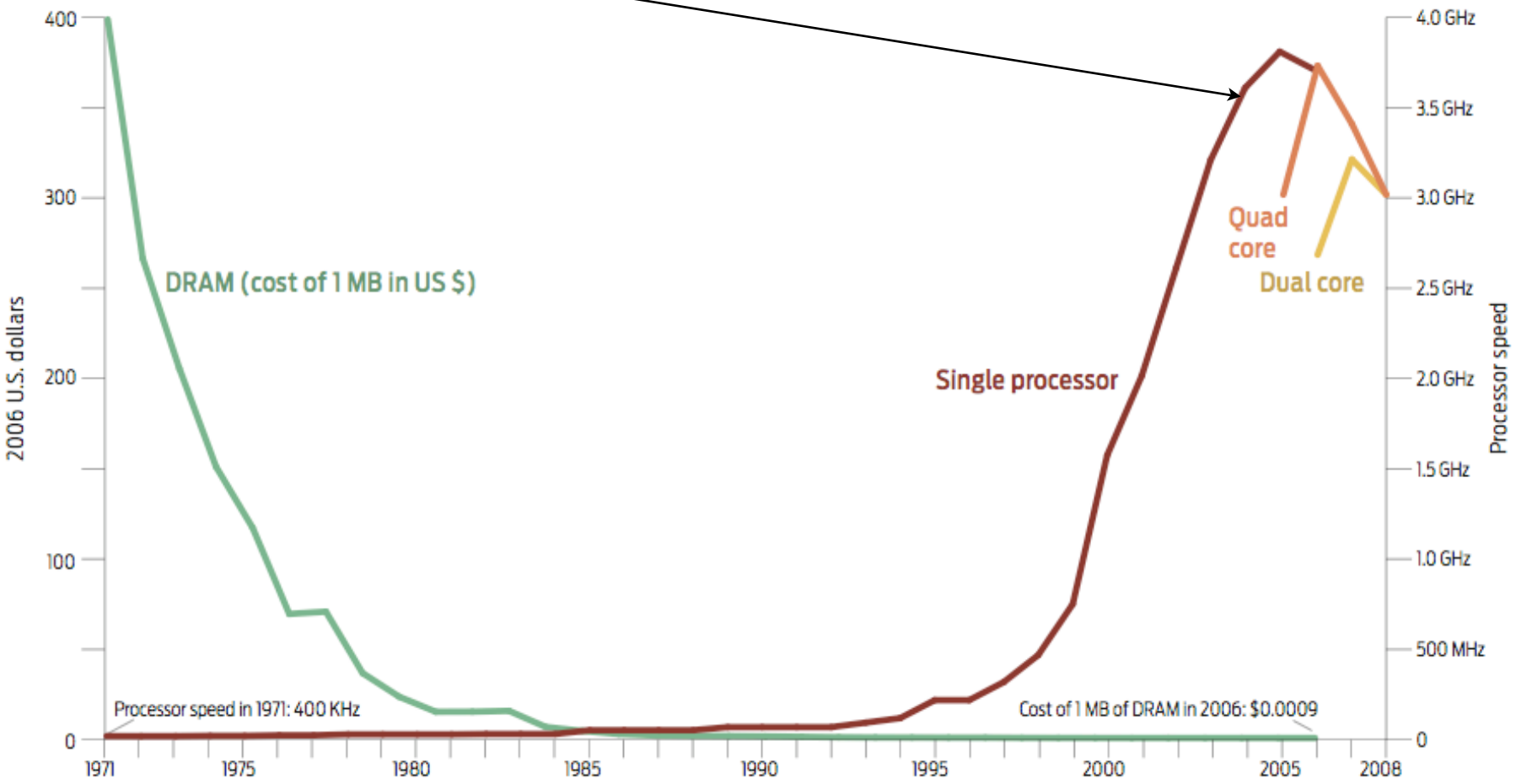


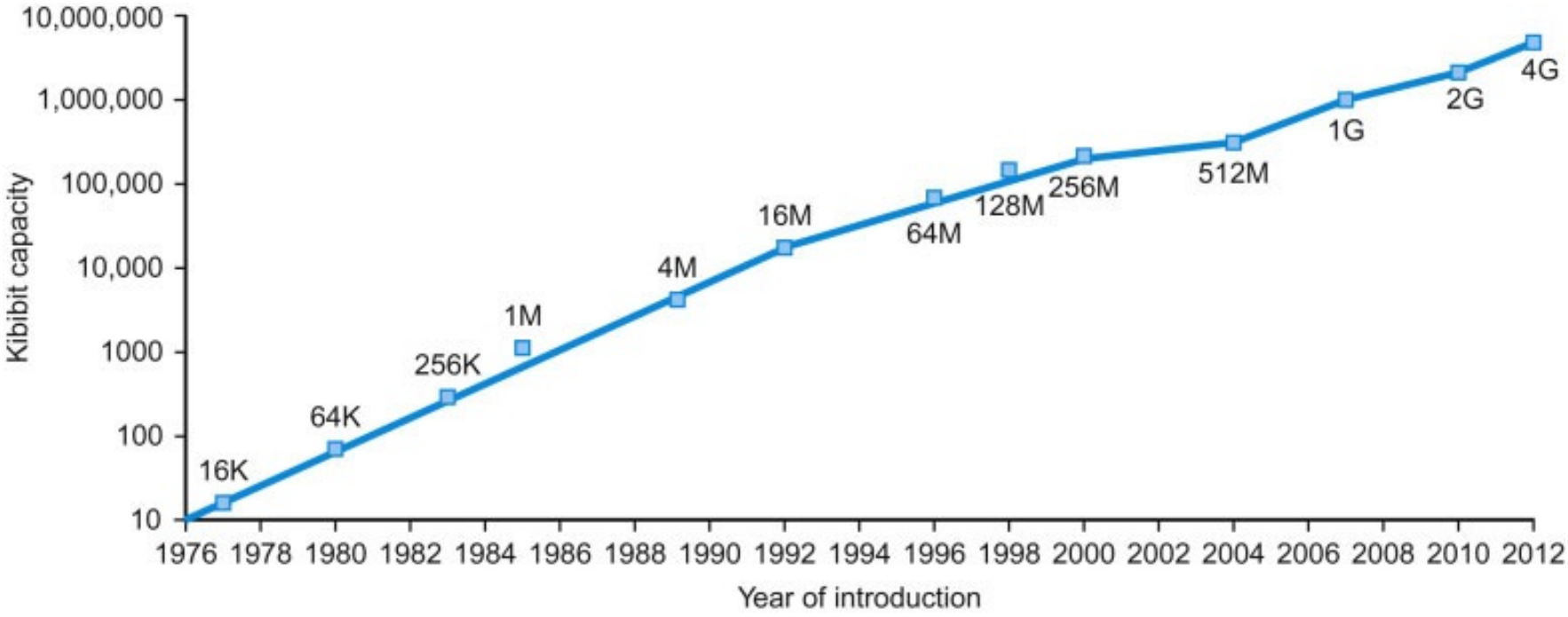
Figure 1.16, page 42. Performance compared to a VAX-11/780 (1978).

- But, there is a problem: Too much heat!

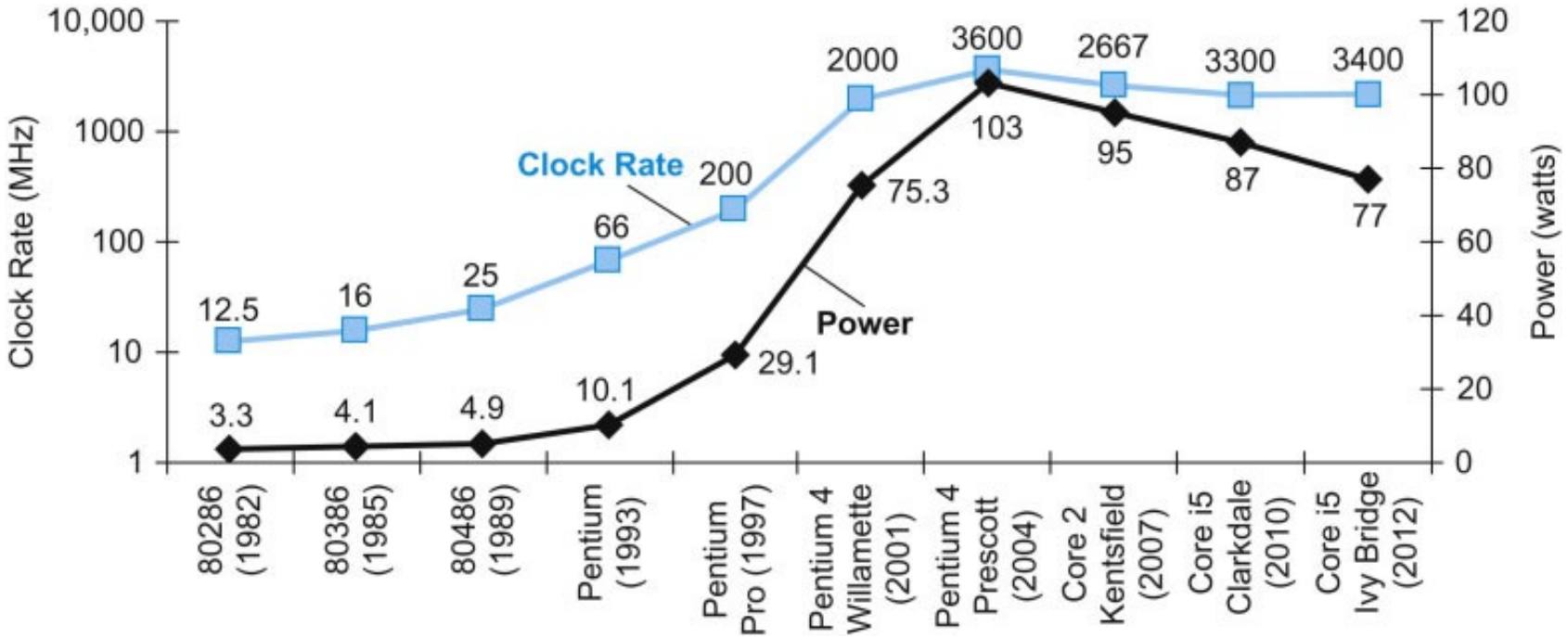


IEEE Spectrum, May 2008, page 56, http://spectrum.ieee.org/computing/hardware/37-years-of-moores-law.

- Memory growth:
  - Figure 1.11, page 25.
  - Capacity quadrupled every 3 years for about 20 years.
  - Rate has slowed to doubling every two to three years.



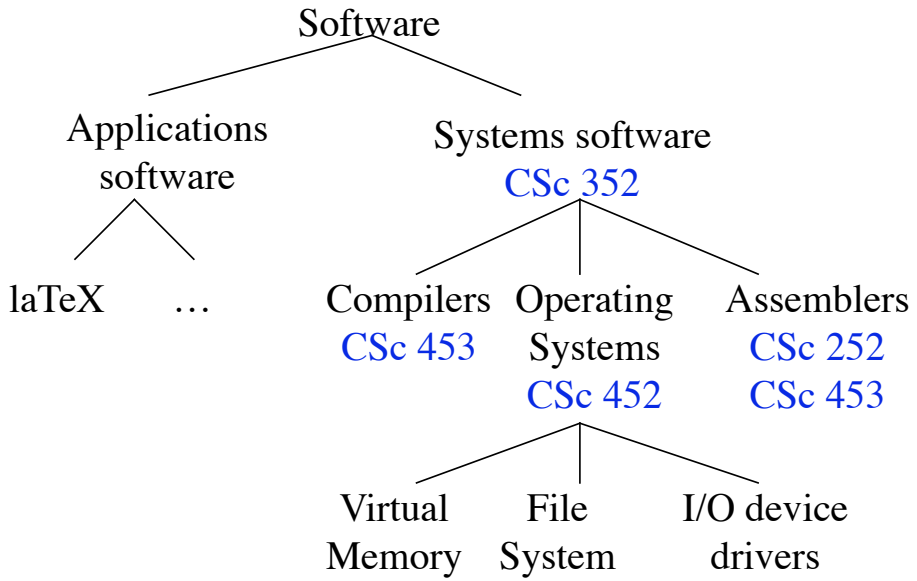
- Power vs. Clock Rate:
  - Clock rate is a rough measure of the performance of a processor
    - Valid mainly when comparing the same processor.
  - Power is the energy needed to drive the processor, and the heat that has to be dissipated by the computer.



- What this course covers: Two major topics.
  - Architecture:
    - Representing numbers (and other things) in binary form.
    - Design of a (small) Central Processing Unit (CPU)
    - Improving performance: pipelining and memory cache.
  - Assembly language programming:
    - Basics: if's, loops, arrays.
    - Functions.
    - Recursion and/or structures.

Why learn this stuff?

- It's a required course :-)
- Useful in understanding other topics in CS.
  - CSc 352 — Systems Programming and Unix.
  - CSc 452 — Principles of Operating Systems.
  - CSc 453 — Compilers and Systems Software.
  - CSc 425 — Principles of Networking.
  - CSc 422 — Parallel and Distributed Computing.
- You want to build software people use (need performance).
- You work with embedded applications.
- You need to make a purchasing decision or offer "expert" advice.
- Learn the lingo (cache, MBytes, GBytes, MHz, GHz, pipelining, super-scalar, vector, etc.)



What is a Computer?

- Components:
  - Input (mouse, keyboard, joystick, …).
  - Output (display, printer, …).
  - Memory (disk drives, DRAM, SRAM, SDRAM, DDR Ram, CD, flash, …).
  - Network.

- Our primary focus: the processor (Central Processing Unit, CPU).
  - Datapath (how data moves within the CPU) and control.
  - Implemented using billions of transistors.
  - Impossible to understand by looking at each transistor.
  - We need:

Abstractions:

- Delving into the depths reveals more information.
- Abstractions omit unneeded detail.
  - Help us cope with complexity.
  - What are some of the details that appear in these familiar abstractions?

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;
} /* swap */
```

C Compiler

Assembly language program (for MIPS)

```
swap:
    sll  $t0, $a1, 2
    add  $t0, $a0, $t0
    lw   $t7, 0($t0)
    lw   $s0, 4($t0)
    sw   $s0, 0($t0)
    sw   $s0, 4($t0)
    jr   $ra
```

Assembler

Binary machine language program (for MIPS)

```
000000 00000 00101 01000 00010 000000
000000 00100 01000 01000 00000 100000
100011 01000 01111 0000000000000000
100011 01000 10000 0000000000000100
101011 01000 10000 0000000000000000
101011 01000 01111 0000000000000100
000000 11111 00000 00000 00000 001000
```

Figure 1.4, page 15 (modified)

Instruction Set Architecture:

- A very important abstraction.
  - Interface between hardware and low-level software.
  - Standardizes instructions, machine language bit patterns, etc.
  - Advantage: different implementations of the same architecture.
  - Disadvantage: sometimes prevents using new innovations.
  - True or False: Binary compatibility is extraordinarily important?

- Modern instruction set architectures:
  - 80x86/PentiumXX, Intel iCore, ARM, PowerPC, MIPS, UltraSparc, …

Where We Are Headed:

- Binary representation of numbers and other things (Chapter 3).
- A specific instruction set architecture (Chapter 2 and Appendix A).
  - Introduction to assembly language programming.
  - Integer arithmetic.
  - Flow control.
  - Procedure calls.
- Arithmetic and how to build an ALU (Appendix B).
  - Data representations.
  - Floating point.
- Constructing a processor to execute our instructions (Chapter 4).
- Pipelining to improve performance (Chapter 4).
- Caching to improve memory performance (Chapter 5).

How to Do Well in this Course:

- Study the lecture notes (both those I provide and those you add). Ask questions about what you do not understand.

- Read the text. Ask questions about what you do not understand.

- Work out the assignments (programs & homework assignments). Do so early! Ask questions about what you do not understand.

- Review returned quizzes, homework assignments, and programs. Ask questions about what you do not understand.

- Read and use the Piazza.

Episode 7: New Allies, New Enemies
Star Wars - V: The Empire Strikes Back
NPR Original Radio Drama