CS 356 Unit 0

Class Introduction

Basic Hardware Organization

What is This Course About?

- Introduction to Computer Systems
  - a.k.a. Computer Organization or Architecture
- Filling in the "systems" details
  - How is software generated (compilers, libraries) and executed (OS, etc.)
  - How does computer hardware work and how does it execute the software I write?
- Lays a foundation for future CS courses
  - CS 350 (Operating Systems), ITP/CS 439 (Compilers), CS 353/EE 450 (Networks), EE 457 (Computer Architecture)

Today's Digital Environment

Why is System Knowledge Important?

- Increase productivity
  - Debugging
  - Build/compilation
- High-level language abstractions break down at certain points
- Improve performance
  - Take advantage of hardware features
  - Avoid pitfalls presented by the hardware
- Basis of understanding security and exploits

What Will You Learn

- Binary representation systems
- Assembly
- Processor organization
- Memory subsystems (caching, virtual memory)
- Compiler optimization and linking

Syllabus

20-Second Timeout

- Who Am I?
  - Teaching faculty in EE and CS
  - Undergrad at USC in CECS
  - Grad at USC in EE
  - Work(ed) at Raytheon
  - Learning Spanish (and Chinese?)
  - Sports enthusiast!
    - Basketball
    - Baseball
    - Ultimate Frisbee?

ABSTRACTIONS & REALITY

Abstraction vs. Reality

- Abstraction is good until reality intervenes
  - Bugs can result
  - It is important to underlying HW implementations
  - Sometimes abstractions don't provide the control or performance you need

Reality 1

- `int`s are not integers and `float`s aren't reals
- Is $x^2 >= 0$?
  - Floats: Yes
  - Ints: Not always
    - 40,000*40,000 = 1,600,000,000
    - 50,000*50,000 = -1,794,967,296
- Is (x+y)+z = x+(y+z)?
  - Ints: Yes
  - Floats: Not always
    - (1e20 + -1e20) + 3.14 = 3.14
    - 1e20 + (-1e20 + 3.14) = around 0

Reality 2

- Knowing some assembly is critical
- You'll probably never write much (any?) code in assembly as compilers are often better than even humans at optimizing code
- But knowing assembly is critical when
  - Tracking down some bugs
  - Taking advantage of certain HW features that a compiler may not be able to use
  - Implementing system software (OS/compilers/libraries)
  - Understanding security and vulnerabilities

Reality 3

- Memory matters!
  - Memory is not infinite
  - Memory can impact performance more than computation for many applications
  - Source of many bugs both for single-threaded and especially parallel programs
  - Source of many security vulnerabilities

Reality 4

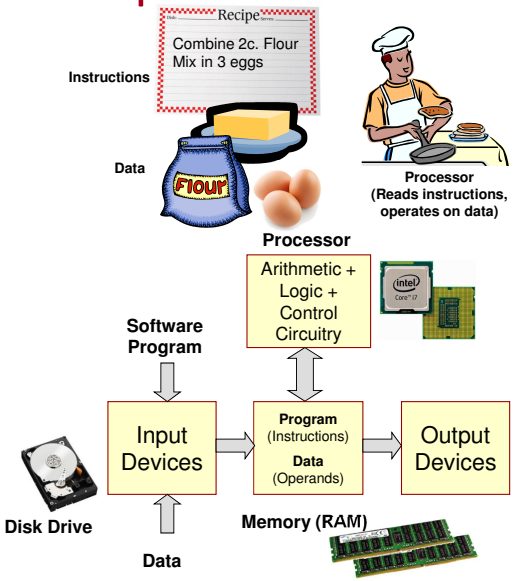
- There's more to performance than asymptotic complexity
  - Constant factors matter!
  - Even operation counts do not predict performance
    - How long an instruction takes to execute is not deterministic…it depends on what other instructions have been execute before it
  - Understanding how to optimize for the processor organization and memory can lead to up to an order of magnitude performance increase

Drivers and Trends

COMPUTER ORGANIZATION AND ARCHITECTURE

Computer Components

- Processor
  - Executes the program and performs all the operations
- Main Memory
  - Stores *data* and *program* (*instructions)*
  - Different forms:
    - RAM = read and write but volatile (lose values when power off)
    - ROM = read-only but non-volatile (maintains values when power off)
  - Significantly slower than the processor speeds
- Input / Output Devices
  - Generate and consume data from the system
  - MUCH, MUCH slower than the processor



Architecture Issues

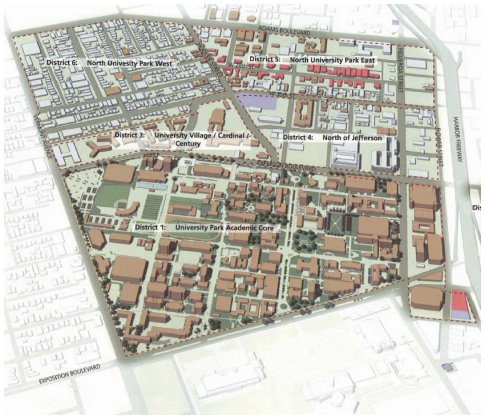
- Fundamentally, computer architecture is all about the different ways of answering the question:

  "What do we do with the ever-increasing number of transistors available to us"

- Goal of a computer architect is to take increasing transistor budgets of a chip (i.e. Moore's Law) and produce an equivalent increase in computational ability

Moore's Law, Computer Architecture & Real-Estate Planning

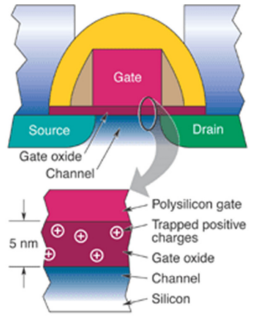
- Moore's Law = Number of transistors able to be fabricated on a chip grows exponentially with time
- Computer architects decide, "What should we do with all of this capability?"
- Similarly real-estate developers ask, "How do we make best use of the land area given to us?"

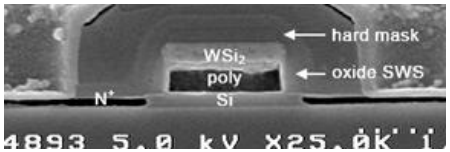


USC University Park Development Master Plan
http://re.usc.edu/docs/University%20Park%20Development%20Project.pdf

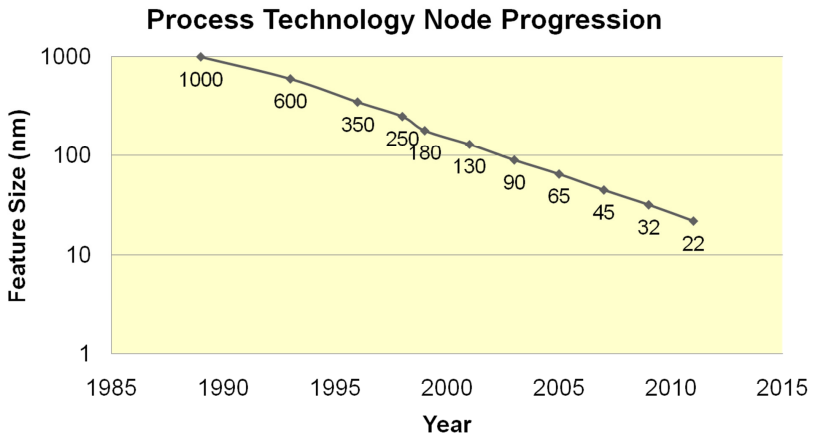
Transistor Physics

- Cross-section of transistors on an IC
- Moore's Law is founded on our ability to keep shrinking transistor sizes
  - Gate/channel width shrinks
  - Gate oxide shrinks
- Transistor feature size is referred to as the implementation "technology node"

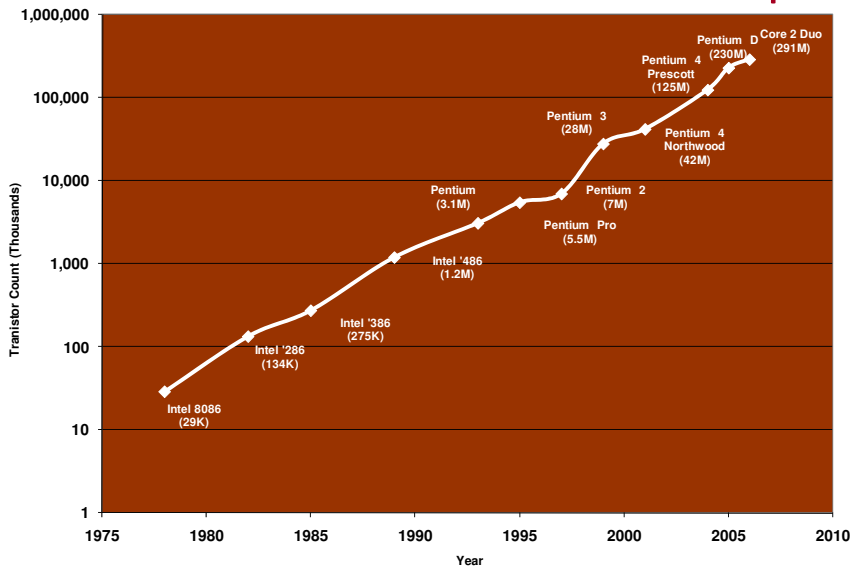




Technology Nodes



Growth of Transistors on Chip

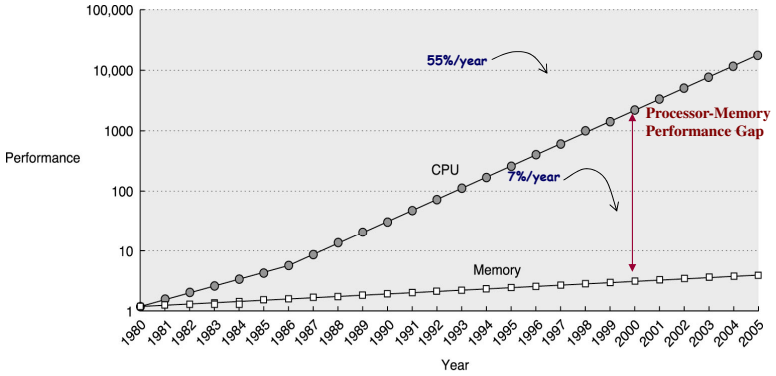


Implications of Moore's Law

- What should we do with all these transistors
  - Put additional simple cores on a chip
  - Use transistors to make cores execute instructions faster
  - Use transistors for more on-chip cache memory
    - Cache is an on-chip memory used to store data the processor is likely to need
    - Faster than main-memory (RAM) which is on a separate chip and much larger (thus slower)

Memory Wall Problem

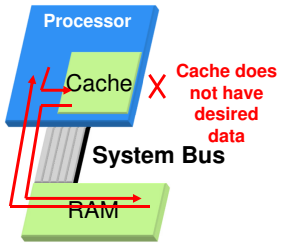
- Processor performance is increasing much faster than memory performance

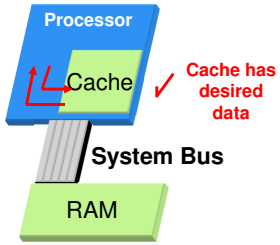


Hennessy and Patterson, *Computer Architecture – A Quantitative Approach* (2003)

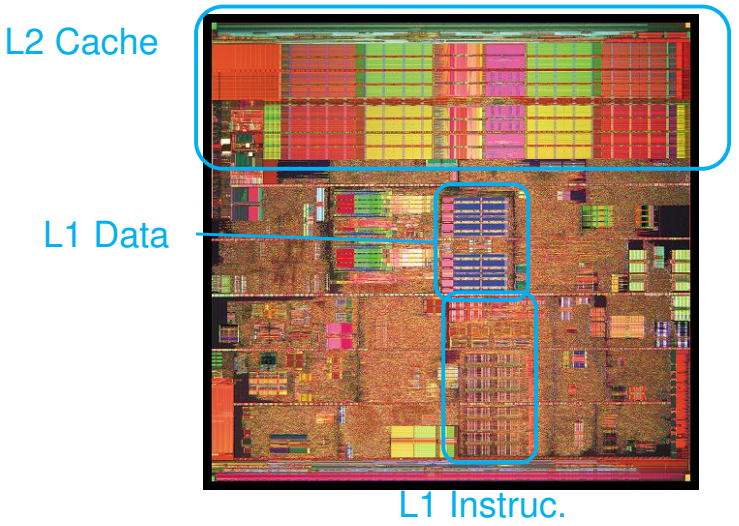
Cache Example

- Small, fast, on-chip memory to store **copies** of recently-used data
- When processor attempts to access data it will check the cache first
  - If the cache has the desired data, it can supply it quickly
  - If the cache does not have the data, it must go to the main memory (RAM) to access it

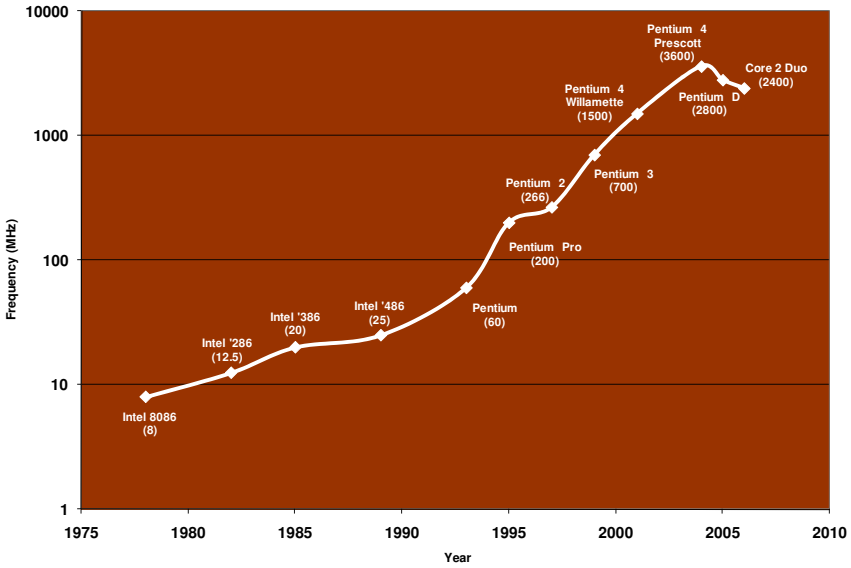




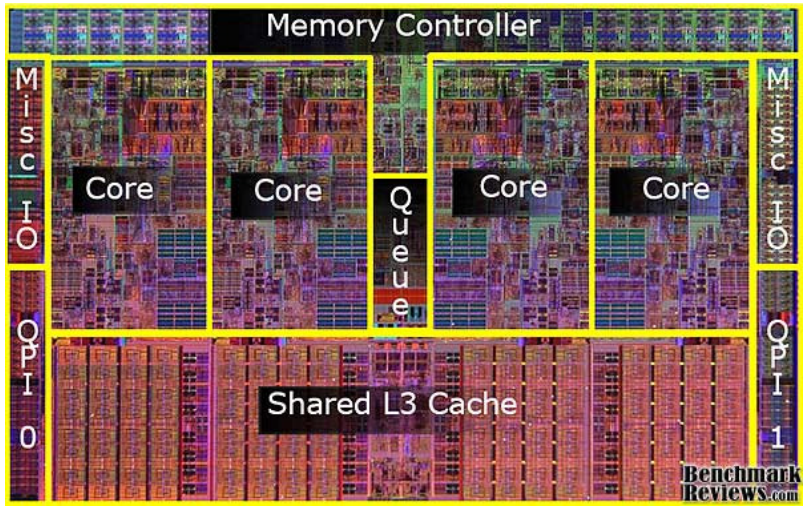
Pentium 4



Increase in Clock Frequency



Intel Nehalem Quad Core



Progression to Parallel Systems

- If power begins to limit clock frequency, how can we continue to achieve more and more operations per second?
  - By running several processor cores in parallel at lower frequencies
  - Two cores @ 2 GHz vs. 1 core @ 4 GHz yield the same theoretical maximum ops./sec.
- For various applications like graphics and computationally intensive workloads this is taken to an extreme by GPUs

GPU Chip Layout

- 2560 Small Cores
- Upwards of 7.2 billion transistors
- 8.2 TFLOPS
- 320 Gbytes/sec

Source: NVIDIA

Photo: http://www.theregister.co.uk/2010/01/19/nvidia_gf100/

Intel Haswell Quad Core