EEL-4713C
Computer Architecture
Lecture 1

Ann Gordon-Ross

Larsen Hall 221

Administrative matters

Instructor: Ann Gordon-Ross (Dr. Ann)
Larsen 221; Office hours: TBD and by appointment
http://www.ann.ece.ufl.edu; ann@ece.ufl.edu

TA: Scott Arnold, scottsarnold@ufl.edu; Office hours: TBD

Web Page: Sakai

Email: Start subject with [EEL 4713] (don't send email via Sakai)

Course files: On Sakai and the directory linked off my webpage

Schedule: Pay special attention to the course schedule, linked off Sakai and in the directory linked of my webpage

Text: Computer Organization & Design
The Hardware / Software Interface (Revised 4th Edition)
by Patterson and Hennessy, Morgan Kauffman Publishers

Overview

- Computer architecture is an exciting field
  - Computer architects are always on the cutting edge
  - Designing several future generations of processors now
- Exciting time to be in computer architecture!
  - Paradigm shift from single-core to multi-core
  - But this class focuses on single-core
  - Multi-core architecture is just a collection of single cores, so must know single-core architecture first.
- Computer architects have a different design philosophy as compared to software designers

What is this class about?

- Computer Architecture:
  - Instruction sets: how are microprocessors programmed?
  - Organization: how does data flow in the microprocessor?
  - Hardware design: how are logic components implemented?

What is this class about?

- Computer Architecture:
  - Instruction sets: how are microprocessors programmed?
  - Hardware/software interface: How are instruction sets designed? How does it impact the design of microprocessors and the software running on them?
  - Example: Apple's move from PowerPC to "x86" (Intel)
    - Enabled greater choice in terms of processor configurations
    - Software migration was a major issue; addressed with "binary translation" software (Rosetta)

What is this class about?

- Computer Architecture:
  - Instruction sets: how are microprocessors programmed?
  - Organization: how does data flow in the microprocessor?
  - Instruction set defines the behavior for each and every instruction supported by a microprocessor; there are multiple organizations that can satisfy the functional behavior, and tradeoffs involved
  - How are the major components of the data path organized and controlled?
  - Example: Intel Pentium 4 vs. Core Duo
    - Additional CPU "core", plus changes in the pipeline design
    - "Wider" instruction issue (4 vs. 3), shorter pipeline
    - "Conroe is nothing like any previous Pentium 4 products. In fact, it's based on the mobile Core Duo design which is in itself based on Pentium M, which is based on the Pentium 3 architecture. So Intel has actually done a bit of a U-turn." (trustedreviews.com)

What is this class about?

- Computer Architecture:
  - Instruction sets: how are microprocessors programmed?
  - Organization: how does data flow in the microprocessor?
  - Hardware design: how are logic components implemented?
    - CMOS, transistor size scaling; power/performance tradeoffs
    - "The Core-based Intel Xeon is so power efficient, that Apple engineers were able to remove the liquid cooling system from the previous Power-PC based model" (apple.com)

What is this class about?

- Computer Architecture:
  - Instruction sets: how are microprocessors programmed?
  - Organization: how does data flow in the microprocessor?
  - Hardware design: how are logic components implemented?
- The process of designing complex digital logic systems
  - Based on knowledge of instruction sets and organization covered in class, you will design a micro-processor using VHDL

What should you expect to achieve in this class?

° In-depth understanding of the inner-workings of modern computers, their evolution, and trade-offs present at the hardware/software boundary.
  - Insight into fast/slow operations that are easy/hard to implement in hardware
    - Tradeoffs between these designs
  - Computer architecture design process
- Hands-on experience with the design process in the context of a large, complex hardware system
- From functional specification to control and datapath implementation and simulation
- Using modern CAD tools and methodologies (VHDL)

Course structure

° Class syllabus:
  - Also refer to policies document for information on academic honesty and late assignments

° Book to be used as supplement for lectures
  - When a topic is covered in class, not all details will be presented.
  - I expect you to read on your own to learn those details

° Additional reading materials

° Key ingredient to success:
  - Read material *before* lecture

° Grading:
  - Assignments (labs/homeworks) – 50%
  - Exams (two midterms, second one is not cumulative) – 50%
    - Tentative dates
      - Midterm 1 – Tentatively 2/16
      - Midterm 2 – Last day of class 4/24

Course Structure

° Lectures (approximate):
  - 2 weeks on Introduction and ISA/MIPS (Chapters 1 and 2)
  - 3 weeks on basic RISC datapath/control design
  - 2 weeks on pipelined processor design
  - 2 weeks on number systems and performance evaluation
  - 2 weeks on memory systems
  - 2 weeks on input/output
  - 1 week on parallelism and other advanced topics
  - 4 extended lab period lectures or special topics

° Slides and reading assignments posted on Sakai or off of course files repository linked off my webpage
  - Acknowledgement:
    - The slides used in class, unless otherwise noted, are adapted from David Patterson's lecture slides

Assignments

° No late assignments will be accepted, no matter what

° Demo in lab, Sakai turn in due Saturday after demo (except for assign 6)

° Three sections:
  - Setup section
    - Get started with tools used
  - Textbook section
    - Helps you keep up with material for exams
  - Lab section
    - Hands-on design experience

° Dos and Donts
  - While studying together in groups is encouraged to foster discussion and learning, all work submitted *must* be your own
    - Not your neighbors, partners, past years' students, from the web, etc. not even with citation
  - Plagiarism will result in an F in the ***course!***

Labs/project

- The project is a major component of this class
  - Goal: expose you to the process of designing a microprocessor
  - Weekly labs will build up to the final processor design
  - Challenging but rewarding
- Throughout this class you will design a MIPS microprocessor:
  - To the extent that it can be simulated within a VHDL-based hardware development framework
  - Starting with the major components of a MIPS datapath
  - Integrate the components and control logic into a processor implementing a subset of MIPS
- Your tools:
  - VHDL and Altera Quartus II
  - Proficiency with these is key to success

Internet companions

- EEL-4713 Web site - Sakai:
  - Lecture slides
  - Assignments
  - Announcements
  - Software documentation, tutorials
  - Discussion forum
  - Course schedule
  - All course files are linked off of my webpage, Sakai may simply refer you to that directory at times
- Computer architecture simulators, Grid appliance
  - Assignments will guide you through usage of tools as appropriate

Next lectures

- Assignment #1 is posted, can start working on it this week in lab
- Reading for the next few lectures: chapters 1 and 2
- Computer Abstractions and Technology
  - Textbook, chapter 1
- Instruction set architectures
  - Textbook, Chapter 2
  - Sections 2.1-2.8, 2.10, 2.12-2.13, 2.18-2.20

What is "Computer Architecture"

Computer Architecture =

Instruction Set Architecture (ISA) + Machine Organization

Classic computer organization:

John von Neumann

Stored program computer

Read instruction and data from memory; decode and execute; write results back to memory

Five key components:

Input, Output, Memory, Datapath and Control

Abstraction layers

User

- High-level language (e.g. C++, Java)
- Low-level language (Assembly)

Software

Hardware

- Register-level transfer (Datapath)
- Basic logic gates (AND, OR)
- Devices (CMOS transistors)

Hardware organization

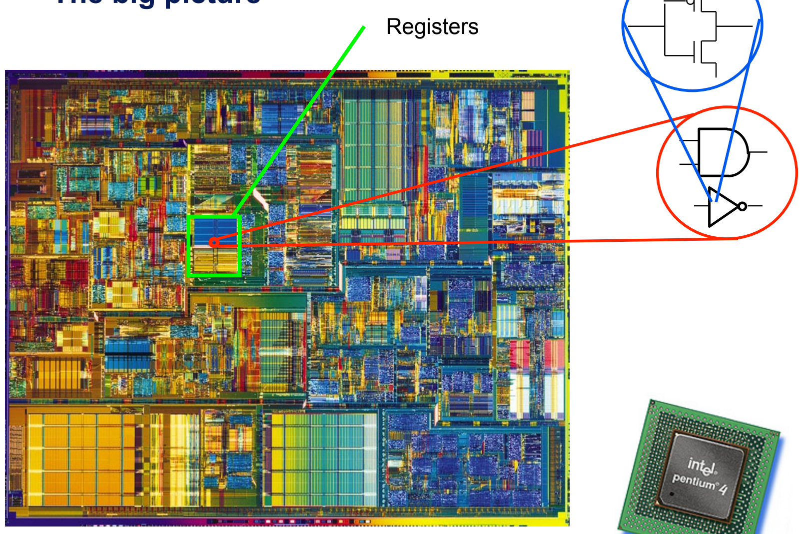
Tradeoff: support an efficient implementation, while providing a standard interface to software

---

Hardware

- Register-level transfer (Datapath)
- Basic logic gates (AND, OR)
- Devices (CMOS transistors)

The big picture



The Pentium™ 4 (~40M transistors)

Software interface

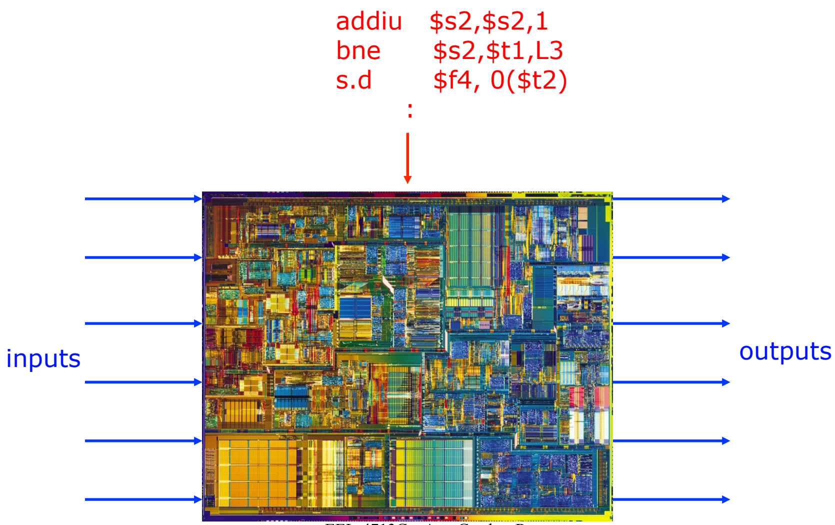
User

- High-level language (e.g. C++, Java)
- Low-level language (Assembly)

Software

Instruction set architecture defines the interface between the microprocessor hardware and software

The big picture (2)



Course Overview

Computer Architecture

Instruction Set
- Machine Language
- Compiler View
- Software interface

e.g. IA-32 vs. IA-64

Organization
- Datapath and control

e.g. Core Duo vs. Athlon

Hardware Design
- Machine Implementation
- Logic Design

e.g. 90nm vs. 65nm; low-power vs. fast clock

Higher → *Lower*

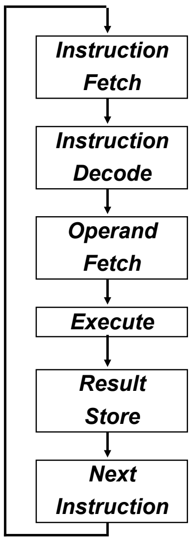
Level of abstraction

Topics addressed in this course

- How are programs written in a high-level language translated into the hardware language?
- What is the interface between the software and the hardware? What are the design criteria used in defining it?
- What determines the performance of a program? How can a programmer improve performance?
- What is the design process starting from the definition of a microprocessor's behavior and finishing with a functional implementation?
- What are techniques that a microprocessor designer can employ to improve performance while maintaining software compatibility?

- Focus on the architecture and organization aspects

Execution cycle (control)



Obtain instruction from program storage

Determine required actions and instruction size

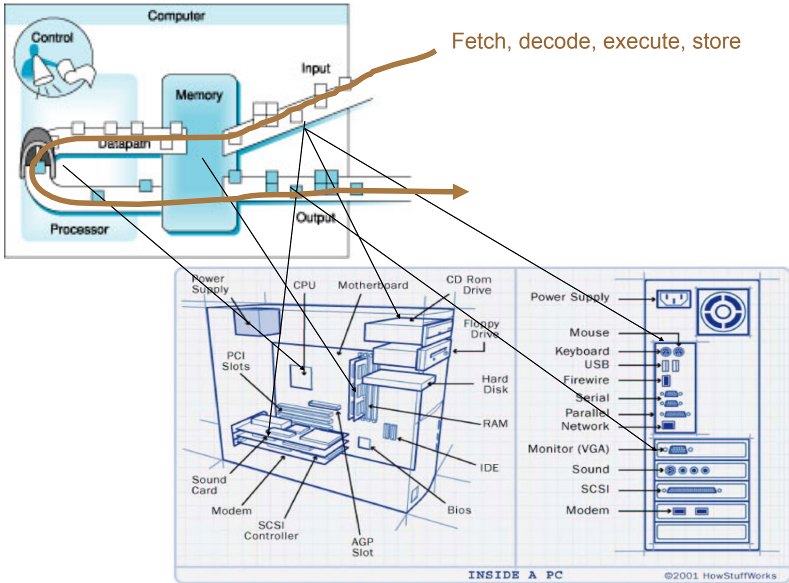
Locate and obtain operand data

Compute result value or status

Deposit results in storage for later use

Determine successor instruction

Five “classic” components of a computer organization



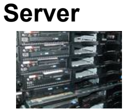
Understanding program performance

- Algorithms and data structures
  - Time/space complexity – e.g. naïve/bubble sort O(n^2) vs. quick sort O(n*logn) determines number of source-level statements executed
  - Not covered in this class
- Programming language, compiler, architecture
  - Determines number of machine-level instructions for each source-level statement
- Processor and memory system
  - Determines how fast instructions go through a fetch/execute/store cycle
- I/O subsystem (hardware and software)
  - How fast instructions which read from/write to I/O devices are executed

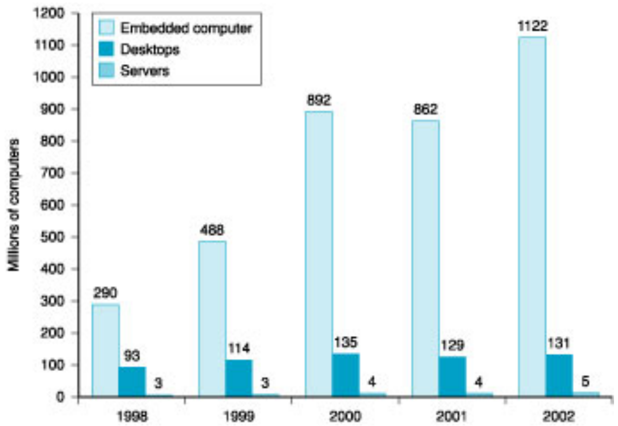
Before and during a program execution

- Before - Applications written in high-level language (e.g. C++) need to be translated to the machine language microprocessors recognize before they execute
  - Compilers
- During - At runtime, applications use services from an operating system to facilitate interaction with the hardware and sharing by multiple entities
  - E.g. Linux, Mac OS, Windows
  - Basic I/O operations on files, network sockets, …
  - Memory allocation
  - Scheduling of CPU cycles across multiple processes

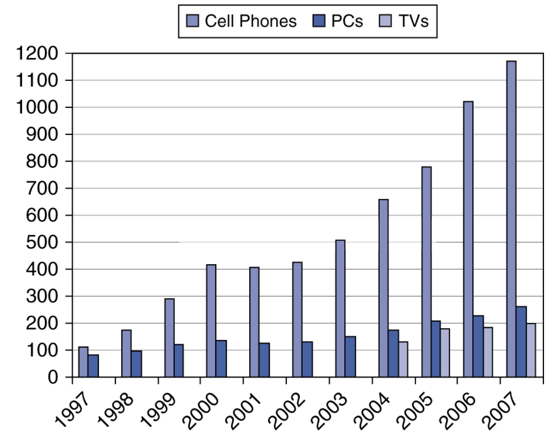
Application classes and characteristics

	Price of system	Price of micro-processor module	Critical system design issues
Desktop 	$500-$5,000	$50-$500	• Tradeoff price/performance • High graphics performance
Server 	$5,000-$5,000,000	$200-$10,000	• High throughput • High availability/dependability • High scalability
Embedded  	Free-$100,000	$0.01-$100	• Low price • Low power consumption • Application-specific performance

Microprocessor markets



Microprocessor market



* No TV data available prior to 2004

Course Overview

Computer Architecture

Instruction Set
- Machine Language
- Compiler View
- Software interface

IA-32 vs. IA-64

Organization
- Datapath and control

Core Duo vs. Athlon

Hardware Design
- Machine Implementation
- Logic Design

90nm vs. 65nm; low-power vs. fast clock

Instruction Set Architecture

. . . the attributes of a [computing] system as seen by the programmer, i.e. the conceptual structure and functional behavior, as distinct from the organization of the data flows and controls of the logic design, and the physical implementation.

Amdahl, Blaaw, and Brooks, 1964

-- Organization of programmable storage

-- Data types & data structures: encodings & representations

-- Instruction formats

-- Instruction (or operation code) set

-- Modes of addressing and accessing data items and instructions

-- Exceptional conditions

Levels of Representation

High Level Language Program

temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;

Compiler

Assembly Language Program

ISA

```
lw  $15,   0($2)
lw  $16,   4($2)
sw  $16,   0($2)
sw  $15,   4($2)
```

Assembler

Machine Language Program

001010…01111….0101
001010…10000….0101

Machine Interpretation

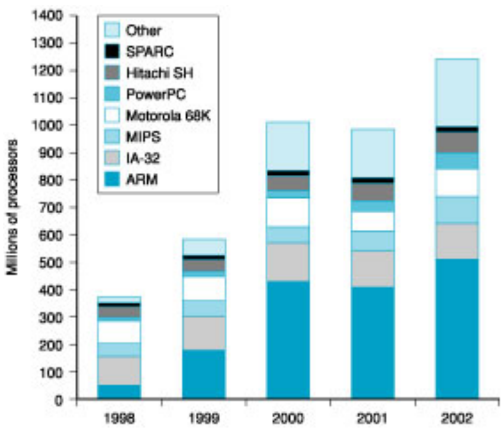
Control Signal Spec

assert address 0($2) on bus
assert memory read signal
select register $15; latch

Example Desktop/server Instruction Set Architectures

Same ISA	*Different Hardware Implementations*
° Digital Alpha	(v1, v3)
° HP PA-RISC	(v1.1, v2.0)
° Sun Sparc	(v8, v9)
° SGI MIPS	(MIPS I, II, III, IV, V)
° “x86” (IA-32)	(Intel 8086,80286,80386, 80486,Pentium, MMX, AMD Athlon,…)
° HP/Intel EPIC/IA-64	(Itanium)

Microprocessor sales by ISA



32- and 64-bit
ARM: 80% sales for cell phones
Other: application-specific or customized architectures

Example Instruction Set Architecture (ISA): MIPS R3000

° **Instruction Categories**
- Load/Store
- Integer computation
- Jump and Branch
- Floating Point
- Memory Management
- System

R0 - R31
Special range designations

PC
HI
LO

Instruction Format

OP	rs	rt	rd	shamt	funct

OP	rs	rt	immediate

OP	target

Course Overview

Computer Architecture

Instruction Set
- Machine Language
- Compiler View
- Software interface

IA-32 vs. IA-64

Organization
- Datapath and control

Core Duo vs. Athlon

Hardware Design
- Machine Implementation
- Logic Design

90nm vs. 65nm; low-power vs. fast clock

Organization

Logic Designer's View

-- capabilities & performance characteristics of principal functional units

(e.g., registers, ALU, shifters, etc.)

-- ways in which these components are interconnected

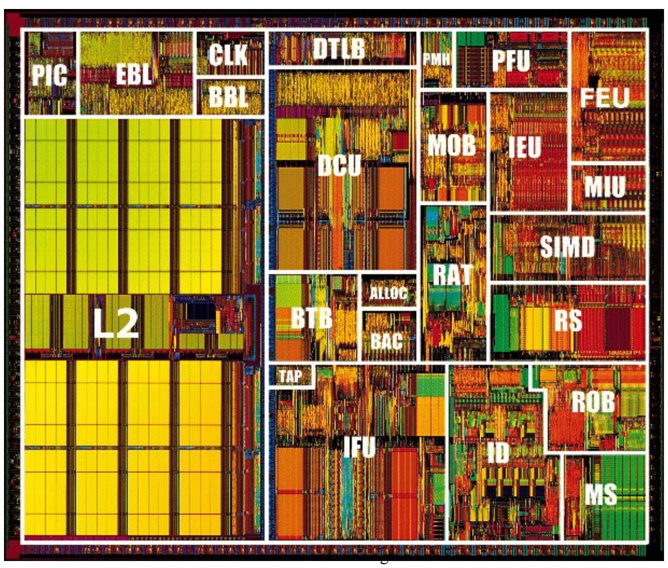
-- nature of information flows between components

-- logic and means by which such information flow is controlled.

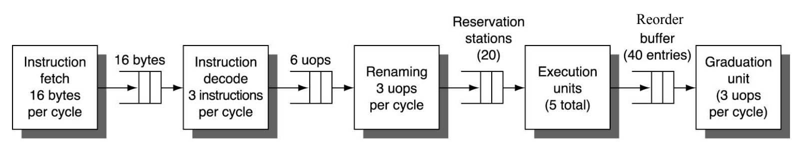
Choreography of units to realize the ISA

Register Transfer Level description

Example: Pentium III die



Example: Pentium III pipeline overview



Course Overview

Computer Architecture

Instruction Set

- Machine Language
- Compiler View
- Software interface

IA-32 vs. IA-64

Organization

- Datapath and control

Core Duo vs. Athlon

Hardware Design

- Machine Implementation
- Logic Design

90nm vs. 65nm; low-power vs. fast clock

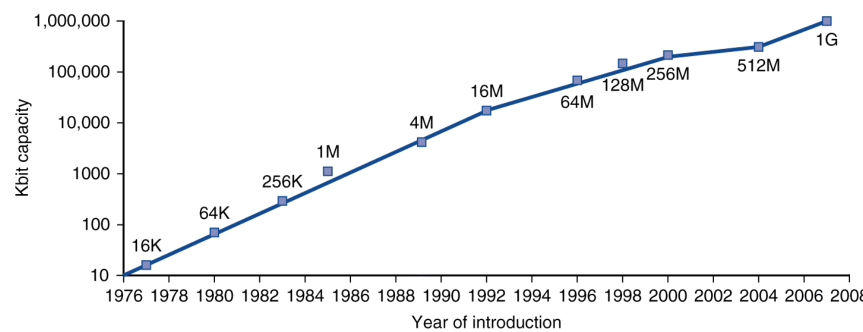
Hardware design and implementation

- Impact performance, cost, and power consumption of architectures
- So far we have enjoyed exponential improvements over time in:
  - Microprocessor performance
  - Main memory capacity
  - Secondary storage capacity
- "Moore's Law"
  - Not an actual physical law; observation of a technology trend
  - Microprocessor capacity doubles roughly every 18-24 months

Technology => dramatic change

- Processor
  - logic capacity: about 30% per year
  - clock rate: about 20% per year
- Memory
  - DRAM capacity: about 60% per year (4x every 3 years)
  - Memory speed: about 10% per year
  - Cost per bit: reduced by about 25% per year
- Disk
  - capacity: about 60% per year

DRAM capacity

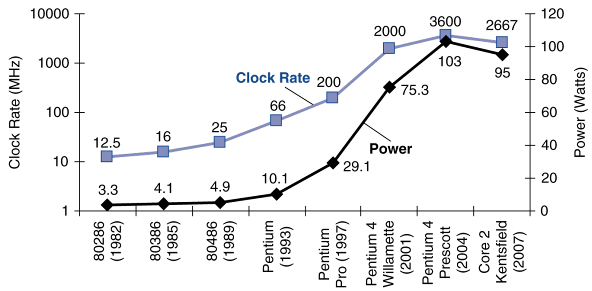


Microprocessor performance

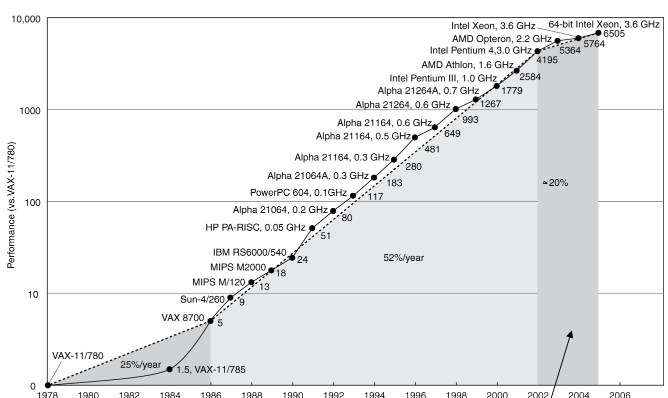
- Improvements also exponential
- Key technology driver: *device scaling*
- As transistors get smaller (e.g. 180nm to 90nm to 65nm feature sizes)
  - They tend to also get faster and consume less power
    - Faster clock rates
  - More transistors can be packed in the same area
    - Superscalar pipelines; multiple cores; larger caches
- Problems faced by scaling at current (nanoscale) technologies:
  - Fast transistors, but slow interconnect
  - Transient errors
  - Low power per device, but billions of them packed together

The power wall

- Dynamic power = capacitive load * Voltage^2 * Frequency
  - Load: function of transistor, wire technologies, fan-in/out
  - As frequency increases, voltage had to be dropped to maintain power at check => 5V down to 1V
  - At very low voltages, leakage and static power consumption become problems, approximately 40%
  - A "wall" blocking frequency scaling



Uniprocessor Performance



Constrained by power, instruction-level parallelism, memory latency

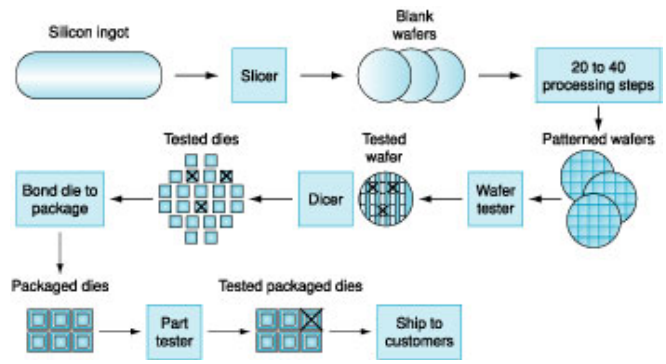
From uniprocessors to multiprocessors

- Clock frequency scaling limited
- Can get better performance by exploiting parallelism – multiple operations per cycle
- Instruction-level (superscalars): diminishing returns circa 2004
- Process/thread-level parallelism: multi-core processors

Multiprocessors

- Multicore microprocessors
  - More than one processor per chip
- 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

CMOS technology process



Next lectures

- Sign up for the Google group, check for assignment #1
- Reading for the next few lectures: chapters 1 and 2
- Computer Abstractions and Technology
  - Textbook, chapter 1
- Instruction set architectures
  - Textbook, Chapter 2
  - Sections 2.1-2.8, 2.10, 2.12-2.13, 2.18-2.20