CS 33:

Computer Organization

Glenn Reinman
4731G Boelter Hall
reinman@cs.ucla.edu

TAs:

Sharath Gopal

Olivera Grujic

Long Nguyen

Garett Ridge

Marco Vitanza

Jason Zheng

Some notes adopted from Bryant and O'Hallaron

Course Components

- Lectures
  - Higher level concepts
- Discussions
  - Applied concepts, important tools and skills for labs, clarification of lectures, exam coverage
- Labs
  - The heart of the course
  - Provide in-depth understanding of an aspect of systems
  - Programming and measurement

More Info

- Web
  - Class web page hosted by CourseWeb
  - Copies of lectures, assignments, exams, solutions
  - Forum
- Office Hours
- Textbook
  - Randal E. Bryant and David R. O'Hallaron. "Computer Systems: A Programmer's Perspective", **2nd Edition**, Prentice Hall 2010.

Grading

- Exams (45%)
  - Two in class exams (15% each)
  - Final (15%)
  - All exams are open book/open notes.
- Labs (50%)
  - 5 labs (10% each)
  - You must work alone on all labs
- Homework (5%)
  - 5 assignments (1% each)
  - Electronic submission only

Tentative Calendar

Week	M	T	W	R	F
1		Intro + Labs (1)		Bits and Bytes (2)	
2		Integers (2)		Floating Point (2)	Data Lab Due
3	HW #1 Due	Machine-Level Rep (3)		Machine-Level Rep (3)	
4	HW #2 Due	Machine-Level Rep (3)		Exam #1	Bomb Lab Due
5	HW #3 Due	Code Optimization (5)		Code Optimization (5)	
6		Memory (6)		Concurrency (12+handouts)	Buffer Lab Due
7	HW #4 Due	Concurrency (12+handouts)		Exam #2	
8		Concurrency (12+handouts)		I/O (10)	OpenMP Lab Due
9	HW #5 Due	Virtual Memory (9)		Linking + Exceptions (7,8)	
10		MIPS (handouts)		Review	CUDA Lab Due

- Homework and Labs Due via CourseWeb by Midnight

Cheating

- What is cheating?
  - Sharing code: either by copying, retyping, looking at, or supplying a copy of a file.
- What is NOT cheating?
  - Helping others use systems or tools.
  - Helping others with high-level design issues.
  - Helping others debug their code.
- Penalty for cheating:
  - At the discretion of the Associate Dean

Lab Facilities

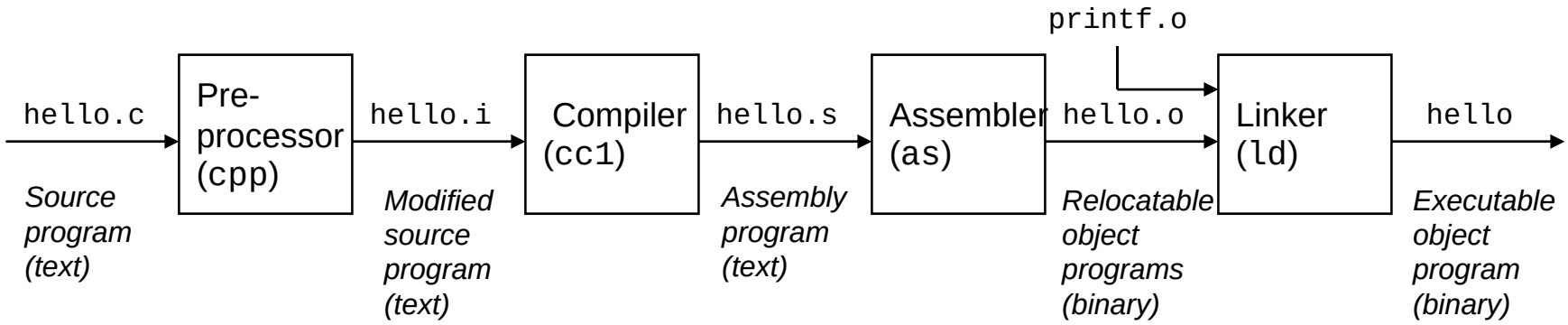
- SEAS Administered Linux Machine
  - `lnxsrv02.seas.ucla.edu`
  - Remote access only
    - Use ssh to log in with your SEAS account
  - Please direct any account issues to the SEAS help desk as they are the only ones with root access on this machine
- Alternatives (Not Recommended)
  - You may use other alternatives to develop your code
  - **BUT: We will test on the SEAS machines**
    - **Your code must work correctly on these machines for credit**

Course Theme

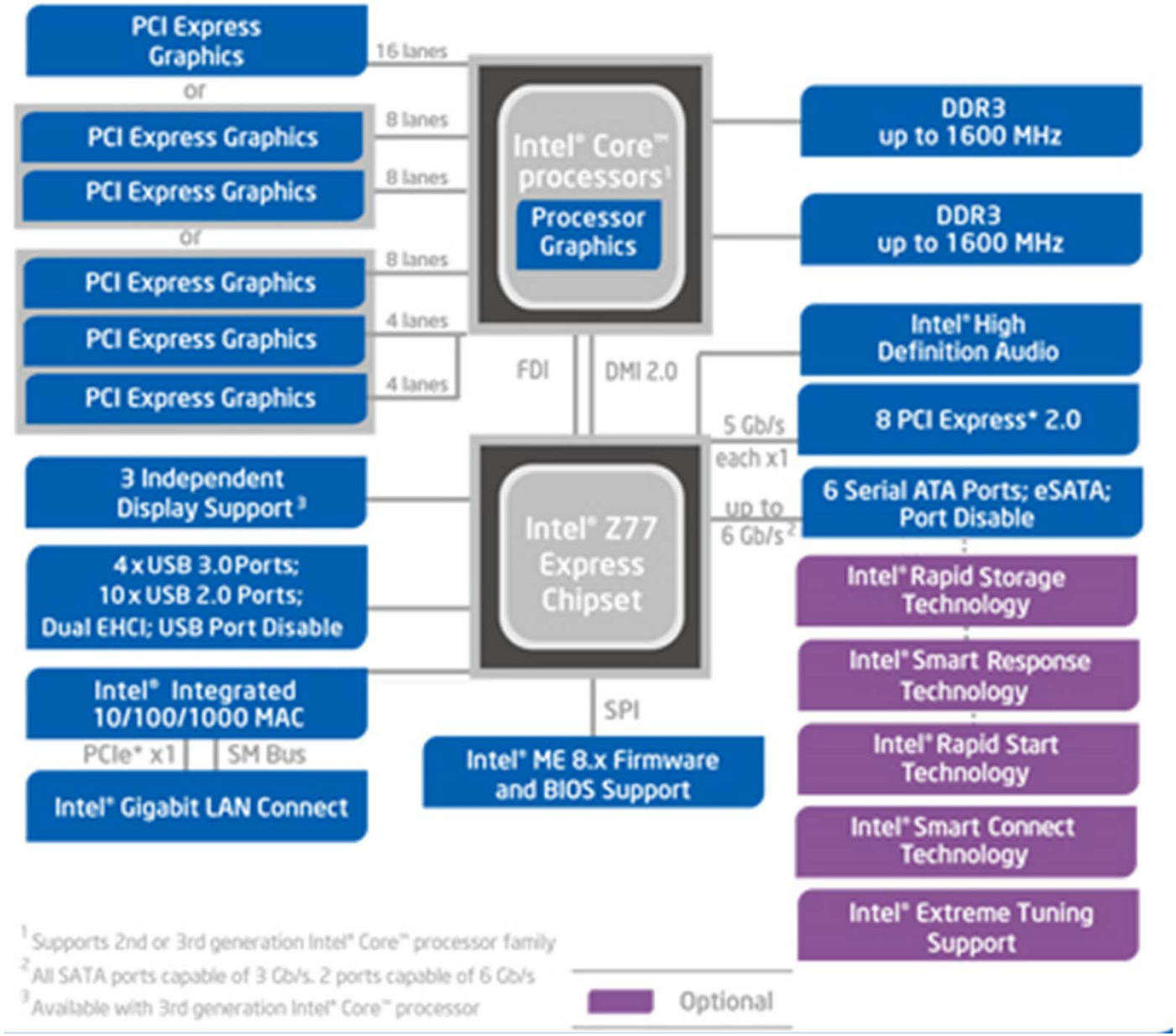
- Abstraction is good, but don't forget reality!
- Abstractions have limits
  - Things are more complex in hardware than they look in C/Java!!
  - Bugs are hard to track/understand if looking only from a high-level point of view
- Useful outcomes
  - Become more effective programmers
    - Able to find and eliminate bugs efficiently
    - Able to tune program performance
  - Prepare for later "systems" classes in CS
    - Compilers, Operating Systems, Networks, Computer Architecture, Parallel Programming

The Compilation System

```
#include <stdio.h>
int main()
{
   printf("hello, world\n");
}
```



Sandy Bridge Architecture



Great Reality #1

- *Computers do more than execute programs*
- They need to get data in and out
  - I/O system critical to program reliability and performance
- They communicate with each other over networks
  - Many system-level issues arise in presence of network
    - Concurrent operations by autonomous processes
    - Coping with unreliable media
    - Cross platform compatibility
    - Complex performance issues

Great Reality #2

- *Parallelism is key to future performance*
- Power-Efficient Performance comes from CMPs
  - Chip Multiprocessors (CMPs)
  - Multiple processor cores integrated onto a single silicon die or multichip module
  - Examples?
    - Intel's Core iX
    - Sun's Niagara 2
    - IBM/Sony's Cell
    - nVidia's Tesla, Fermi, Keppler
- Programmers need to learn how to exploit parallelism in their applications

Great Reality #3

- *You've got to know assembly & the machine!*
- Although you may never **program** in assembly
  - Compilers are much better & more patient than you are
- Assembly key to machine-level execution model
  - Behavior of programs in presence of bugs
    - High-level language model breaks down
  - Tuning program performance
    - Understanding sources of program inefficiency
  - Implementing system software
    - Compiler has machine code as target
    - Operating systems must manage process state

Great Reality #4

- *Memory Matters*
- Memory is not unbounded
  - It must be allocated and managed
  - Many applications are memory dominated
- Memory performance is not uniform
  - Cache and virtual memory effects can greatly affect program performance
  - Adapting program to characteristics of memory system can lead to major speed improvements
- Memory referencing bugs especially pernicious
  - Effects are distant in both time and space

Great Reality #5

- *There's more to performance than asymptotic complexity*
- Constant factors matter too!
  - 10:1 performance range depending on how code written
  - Must optimize at multiple levels: algorithm, data representations, procedures, and loops
- Must understand system to optimize performance
  - How programs are compiled and executed
  - Measure program performance and identify bottlenecks
  - How to improve performance without destroying code modularity and generality

Great Reality #6

- *Int's are not Integers, Float's are not Reals*
- Examples
  - Is $x^2 \geq 0$?
    - IEEE float: Yes!
    - 32 bit int:
      - 40000 * 40000 --> 1600000000
      - 50000 * 50000 --> ??
  - Is (x + y) + z = x + (y + z)?
    - Unsigned & Signed Int's: Yes!
    - Float's:
      - (1e20 + -1e20) + 3.14 --> 3.14
      - 1e20 + (-1e20 + 3.14) --> ??

Computer Arithmetic

- Does not generate random values
  - Arithmetic operations have important mathematical properties
- Cannot assume “usual” properties
  - Due to finiteness of representations
  - Integer operations satisfy “ring” properties
    - Commutativity, associativity, distributivity
  - Floating point operations satisfy “ordering” properties
    - Monotonicity, values of signs
- Observation
  - Need to understand which abstractions apply in which contexts
  - Important issues for compiler writers and numerical application programmers

Tools and Skills

- C Programming
- x86 Assembly
- Basic CUDA
- Basic MIPS
- Debugging
  - gdb – GNU Debugger
- Rudimentary Understanding of
  - An Editor (i.e. emacs, vi)
  - Linux