CSSE132
Introduction to Computer Systems

1 : Introduction

March 4, 2013

Overview

- Introduction
- Course details
- “Hello world”
  - Compiling
  - Hardware
- Abstractions
- Course Motivations

Introduction

- Micah Taylor
  - Call me ‘Micah’

- Graduated Rose in 2004
- Went to UNC for graduate work
  - Designed interactive acoustic simulations

- My schedule is here: http://www.rose-hulman.edu/~taylormt
  - My office is F216
  - Come by any time!

Introduction

- I drive an awesome 'sports Honda'

- I get worse at video games every year

- I tried to write a ray tracer on my watch

```
000000000000
 000000
9626000000000
.9709
0.2024826979630210000000
0.814309941805363
0.5773502691896260000000000
#f)
(trace
```

- I don't normally dress this formal

Introducing you

- Answer one of these:
  - What kind of awesome ride do you have?
  - Why do people wear watches these days?
  - What video game do you think you can beat me at?
  - Why are you not dressed formally?

Course details

- CSSE132
  - Introduction to Computer Systems
  - Sections 3,4 (and sometimes 2)

- Course website
  - http://www.rose-hulman.edu/class/csse/csse132/
  - See website for syllabus and schedule

Course Components

- Lectures
  - Higher level concepts
- Readings
  - Important tools and skills for labs, clarification of lectures, exam coverage
- Labs (10)
  - 1 each week
  - Provide in-depth understanding of an aspect of systems
  - Programming and measurement
- Homework (6-8)
  - Practice with fundamental ideas
- Exams (2 + final)
  - Test your understanding of concepts & mathematical principles

Textbooks

- Randal E. Bryant and David R. O'Hallaron,
  - "Computer Systems: A Programmer's Perspective, Second Edition" (CS:APP2e), Prentice Hall, 2011
  - http://csapp.cs.cmu.edu
  - This book really matters for the course!
    - How to solve labs
    - Practice problems typical of exam problems

- Brian Kernighan and Dennis Ritchie,
  - "The C Programming Language, Second Edition", Prentice Hall, 1988

Policies: Assignments And Exams

- Work groups
  - Please collaborate on labs and homework!
  - See course website for definition of collaboration
- Handins
  - Homework due at beginning of class
  - Labs are due before midnight
  - Quizzes cannot be made-up
- Conflict exams, other irreducible conflicts
  - OK, but must make PRIOR arrangements with instructor!
  - Notifying us well ahead of time shows maturity and makes us like you more (and thus to work harder to help you out of your problem)

Policies: Grading

- Homework & Quizzes (30%)

- Labs (35%): weighted according to effort

- Exams (35%): weighted 10%, 10%, 15% (final)

- Generally:
  - > 90%: A
  - > 80%: B
  - etc.

“Hello world”

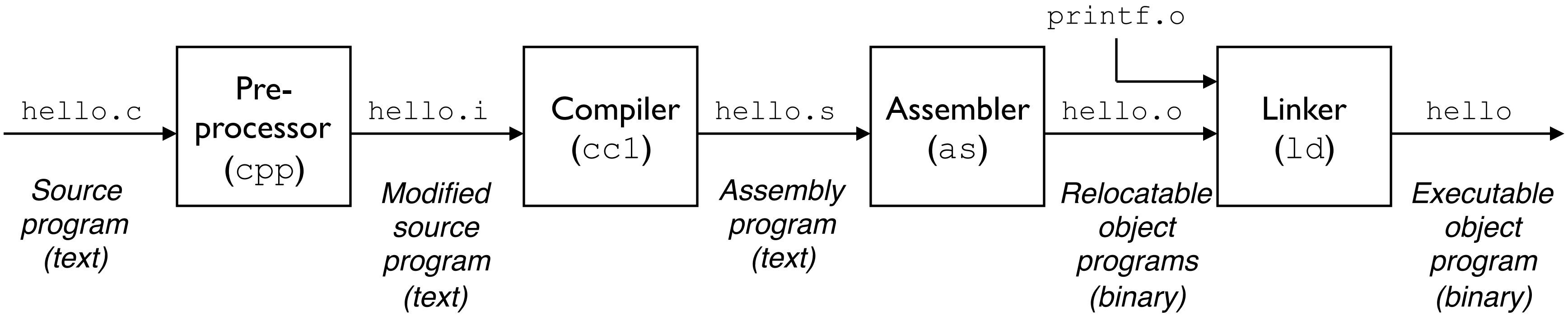
- A small program that prints
  ‘hello, world’

- Require an entire operating system to execute
- Moves data through many hardware components

…Just to print ‘hello, world’

“Hello world”

- First, create executable program
  - Resolve all text substitution needed (Pre-processor)
  - Translate to CPU’s assembly language (Compiler)
  - Translate to CPU’s machine language (Assembler)
  - Combine with other machine code to produce executable (Linker)



  - For demonstration, you can issue

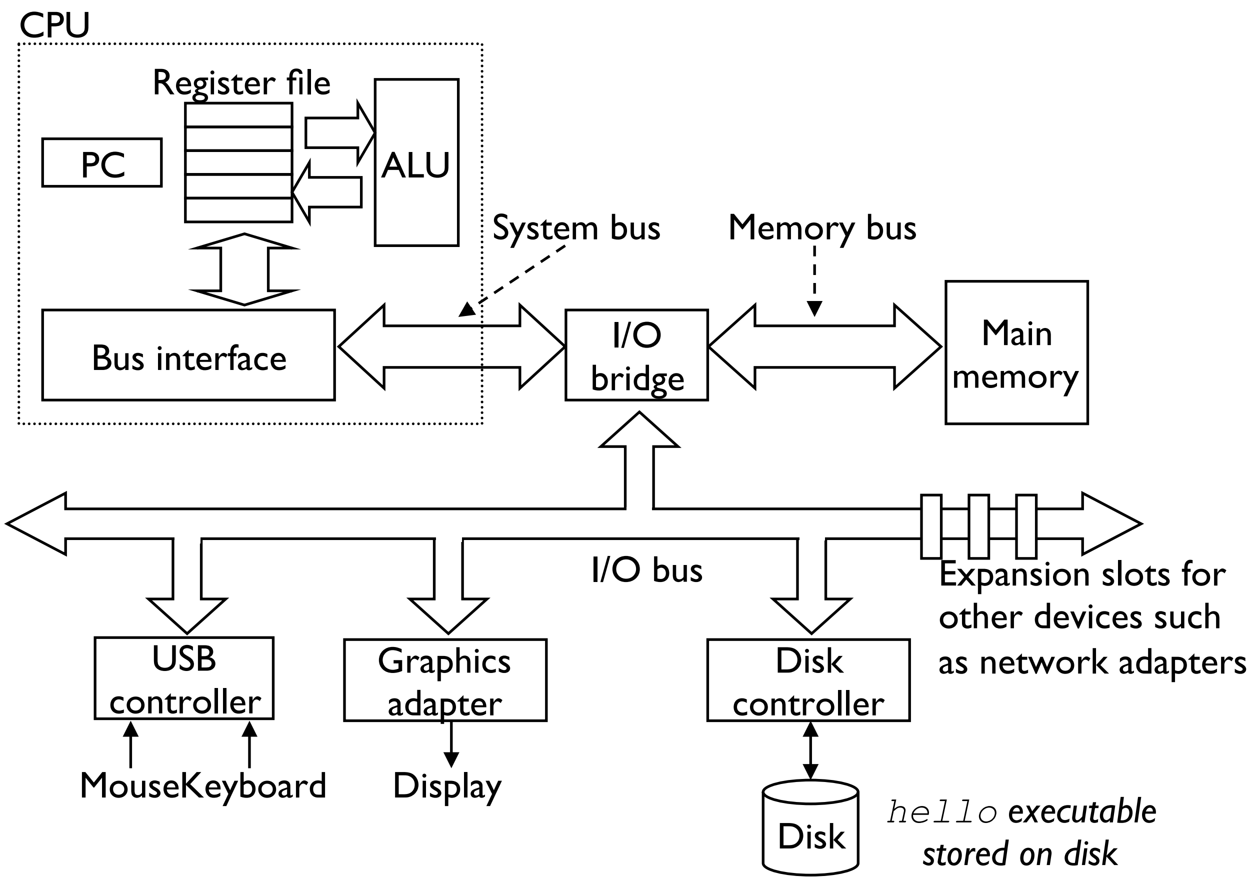
```
gcc –E hello.c > hello.i # preprocess to hello.i
gcc –S hello.i           # compile to hello.s
as hello.s –o hello.o    # assemble to hello.o
gcc hello.o –o hello     # link to hello
```

“Hello world”

- Run executable program on hardware
  - Read user input
    - Accept key stroke from keyboard input
    - Process in CPU, store results in memory
  - Load program
    - When enter is pressed, copy program from disk to memory
  - Execute
    - Load program’s instructions into CPU from memory
    - Execute each instruction on CPU
    - Store results back in memory
  - Output results
    - Copy results from memory to output graphics device

“Hello world”

- Run executable program on hardware
  - Read user input
  - Load program
  - Execute
  - Output results



Abstraction

- We will investigate many abstractions in computer systems
  - Abstraction hides complexity by providing an easy to understand interface

- CPUs appear to execute a list of instructions in sequence
  - Reality: CPUs will execute instructions simultaneously and out-of-order

- Memory allows data to be easily stored and accessed
  - Reality: Memory is incredibly slow and requires multiple levels of access

- Operating Systems run several programs at once
  - Reality: OSs rapidly switch which program processes can access resources
  - It looks like everything is happening at once to us!

Course motivation

- Describe common hardware & software abstractions
  - CPU abstraction
  - Memory hierarchy
  - OS resource management
  - Remove the ‘magic’ from computer systems!
- Become more effective programmers
  - Able to find and eliminate bugs efficiently
  - Able to understand and tune for program performance
- Prepare for later “systems” classes
  - Compilers, Operating Systems, Networks, Computer Architecture, Embedded Systems, etc.

Coming up…

- Prepare to install Linux (Lab1)
- Homework 1 due next Monday!