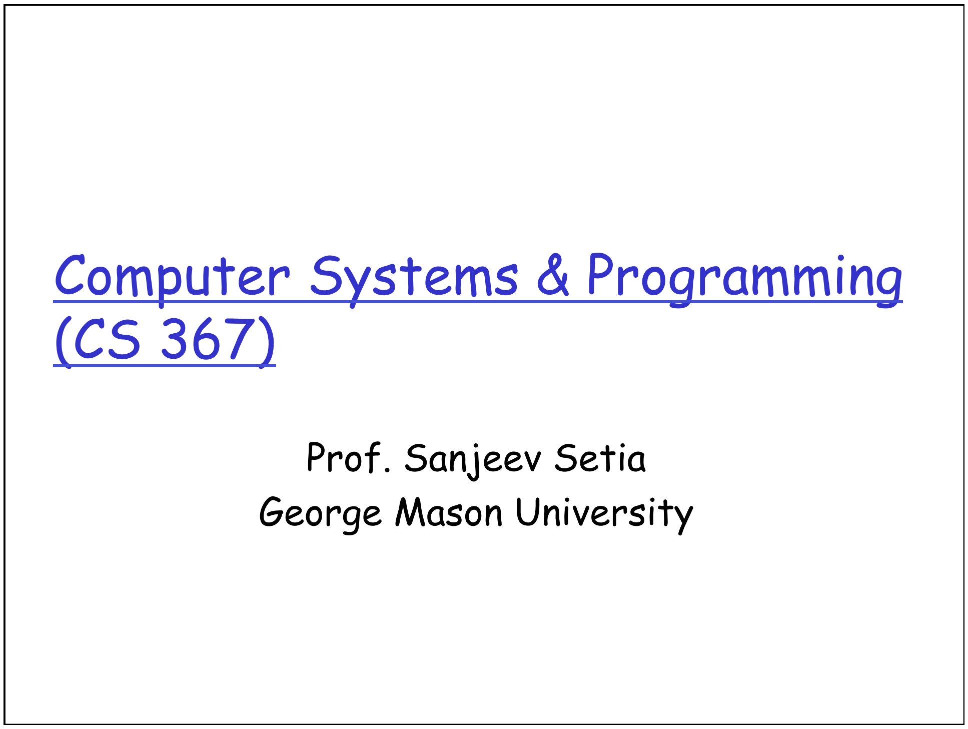
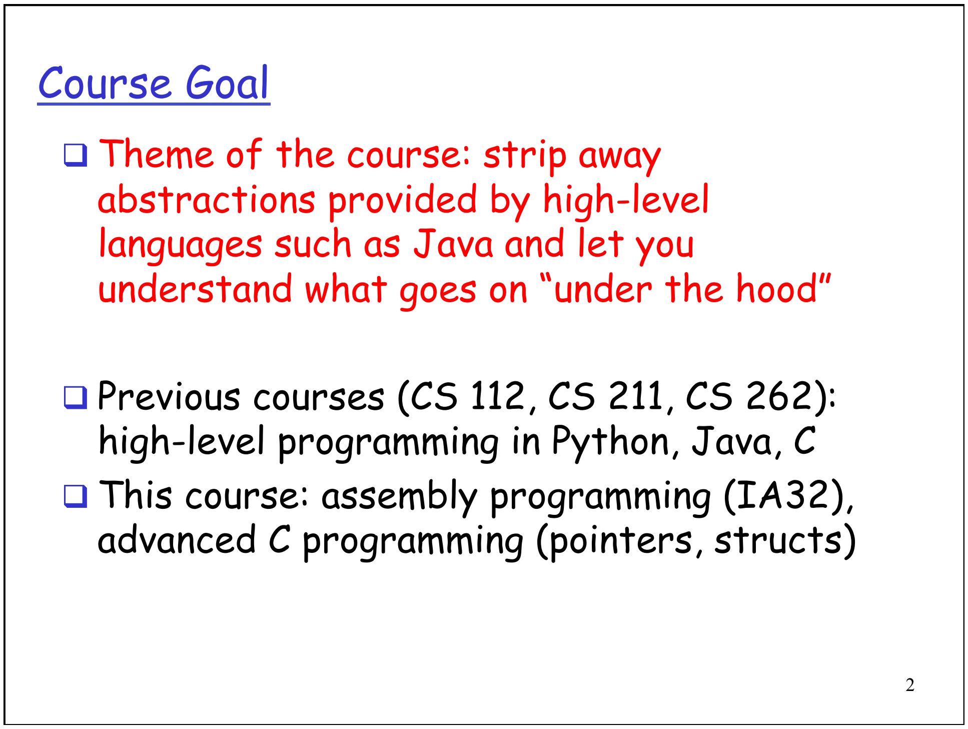


Computer Systems & Programming (CS 367)

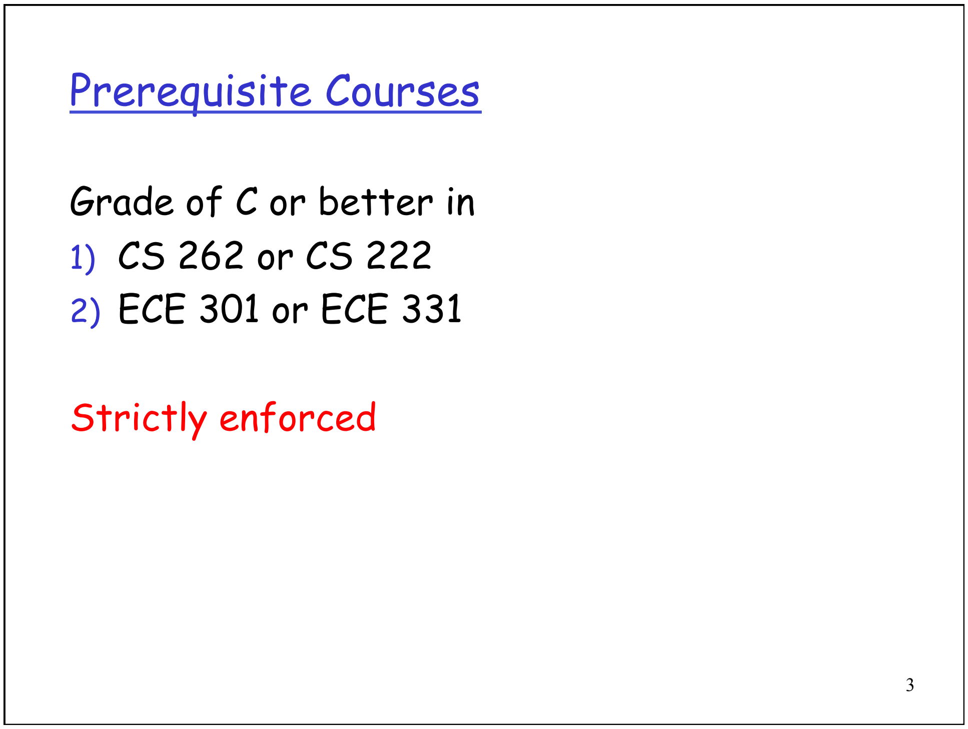
Prof. Sanjeev Setia
George Mason University



Course Goal

- Theme of the course: strip away abstractions provided by high-level languages such as Java and let you understand what goes on "under the hood"

- Previous courses (CS 112, CS 211, CS 262): high-level programming in Python, Java, C
- This course: assembly programming (IA32), advanced C programming (pointers, structs)

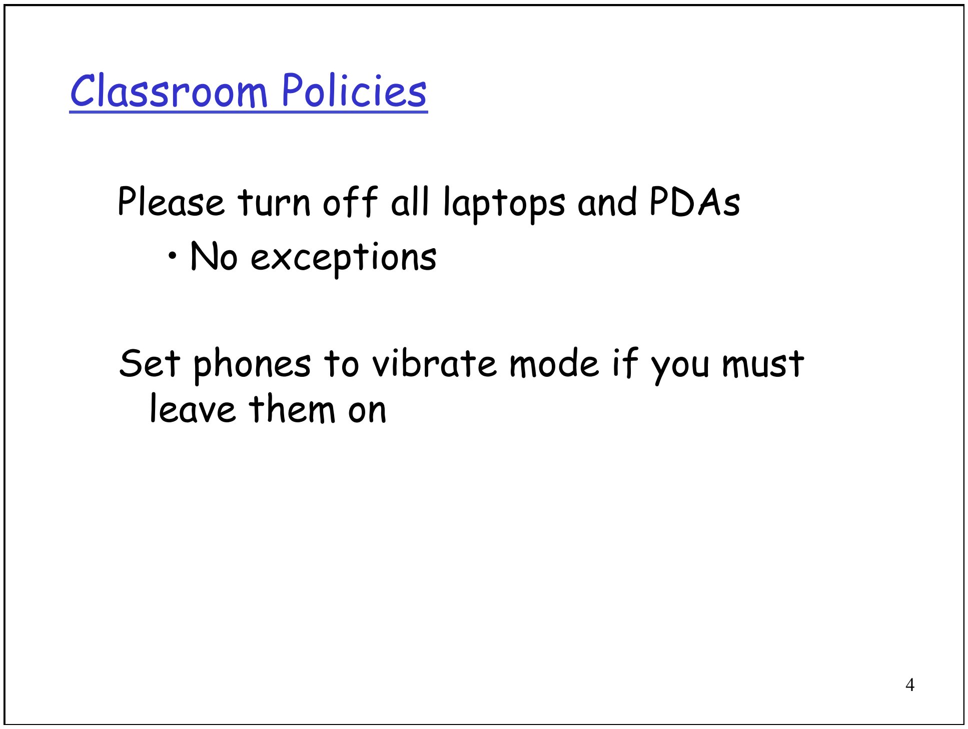


Prerequisite Courses

Grade of C or better in

1) CS 262 or CS 222
2) ECE 301 or ECE 331

Strictly enforced

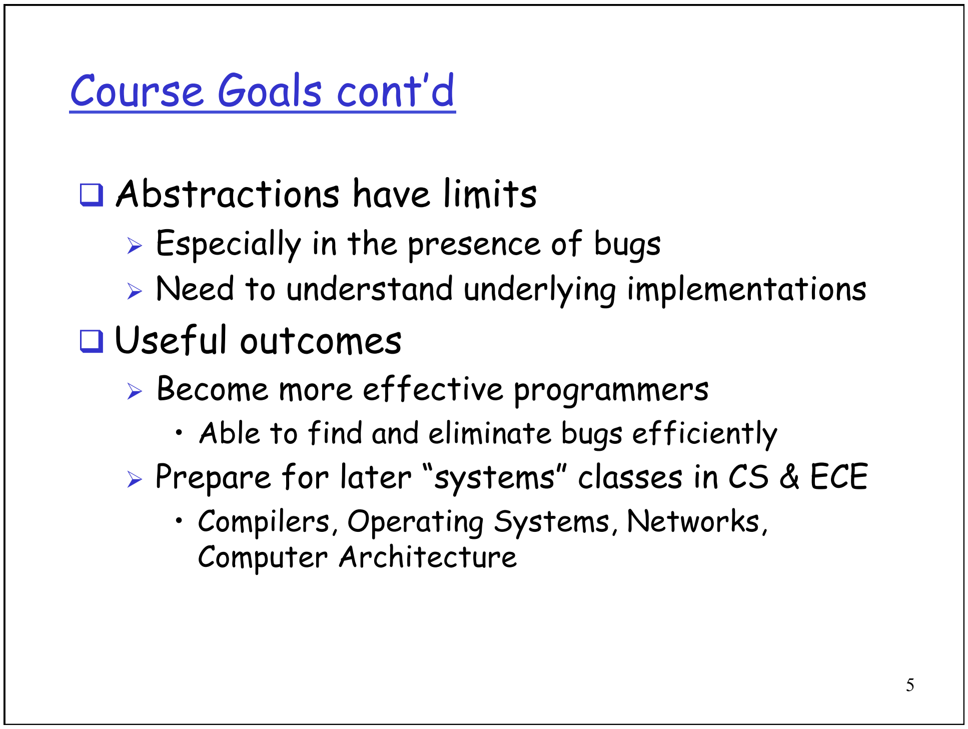


Classroom Policies

Please turn off all laptops and PDAs

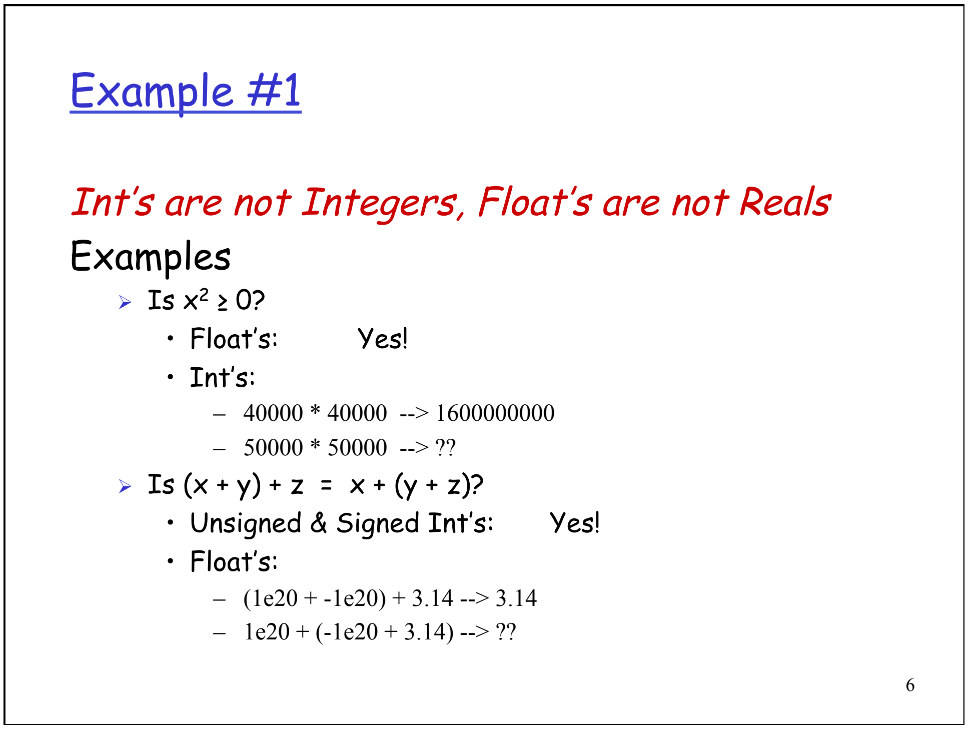
- No exceptions

Set phones to vibrate mode if you must leave them on



Course Goals cont'd

- Abstractions have limits
  - Especially in the presence of bugs
  - Need to understand underlying implementations
- Useful outcomes
  - Become more effective programmers
    - Able to find and eliminate bugs efficiently
  - Prepare for later "systems" classes in CS & ECE
    - Compilers, Operating Systems, Networks, Computer Architecture

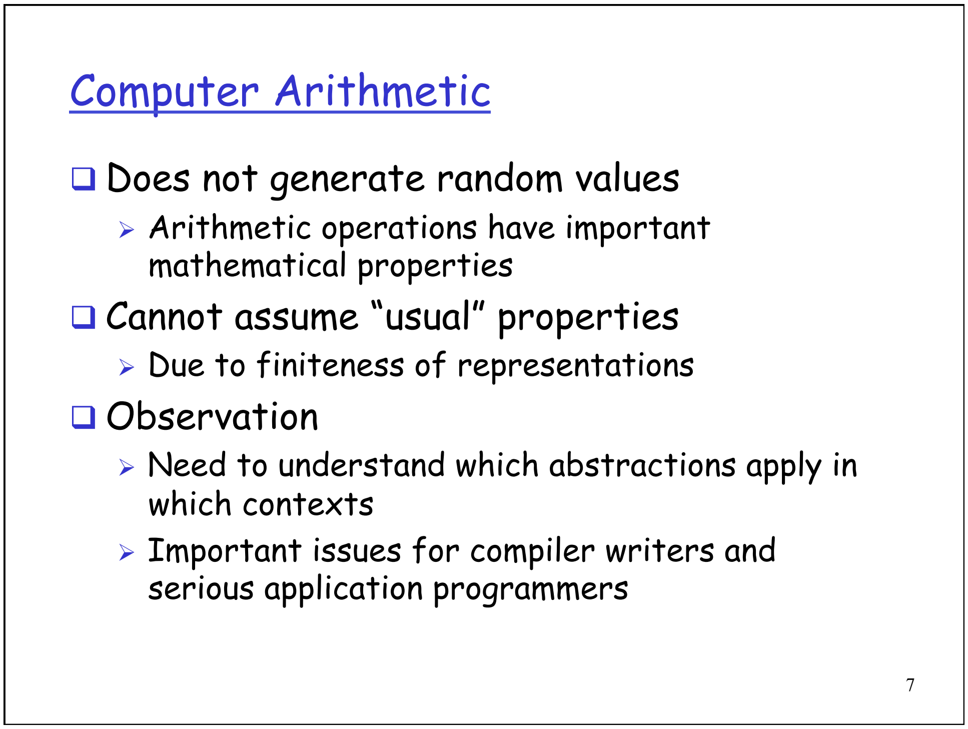


Example #1

Int's are not Integers, Float's are not Reals

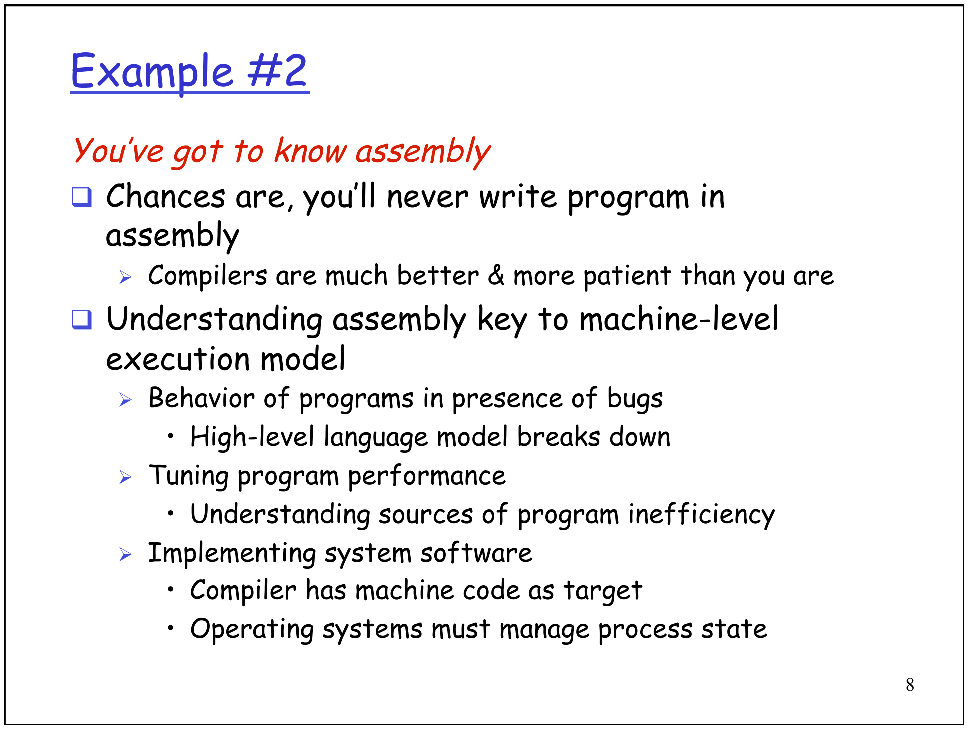
Examples

- Is $x^2 \geq 0$?
  - Float's: Yes!
  - Int's:
    - 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
- ❑ Observation
  - ➢ Need to understand which abstractions apply in which contexts
  - ➢ Important issues for compiler writers and serious application programmers



Example #2

You’ve got to know assembly

- ❑ Chances are, you’ll never write program in assembly
  - ➢ Compilers are much better & more patient than you are
- ❑ Understanding 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

Example #3

Memory Matters

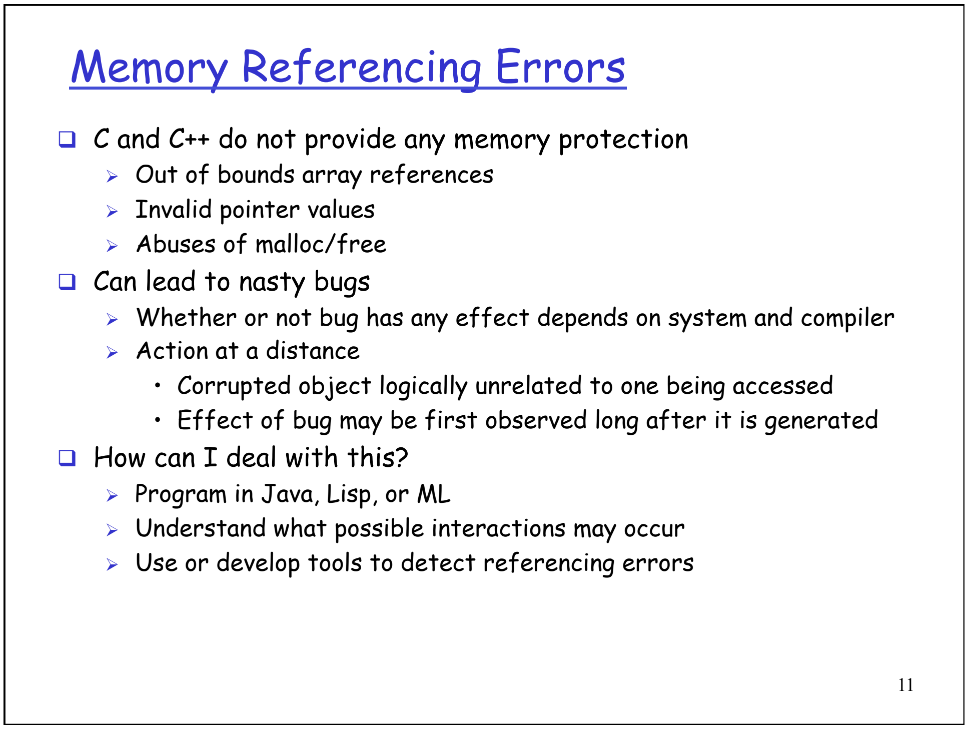
- Memory is not unbounded
  - It must be allocated and managed
  - Many applications are memory dominated
- Memory referencing bugs especially pernicious
  - Effects are distant in both time and space
- Memory performance is not uniform
  - Cache and virtual memory effects can greatly affect program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

Memory Referencing Bug Example

```
main ()
{
  long int a[2];
  double d = 3.14;
  a[2] = 1073741824; /* Out of bounds reference */
  printf("d = %.15g\n", d);
  exit(0);
}
```

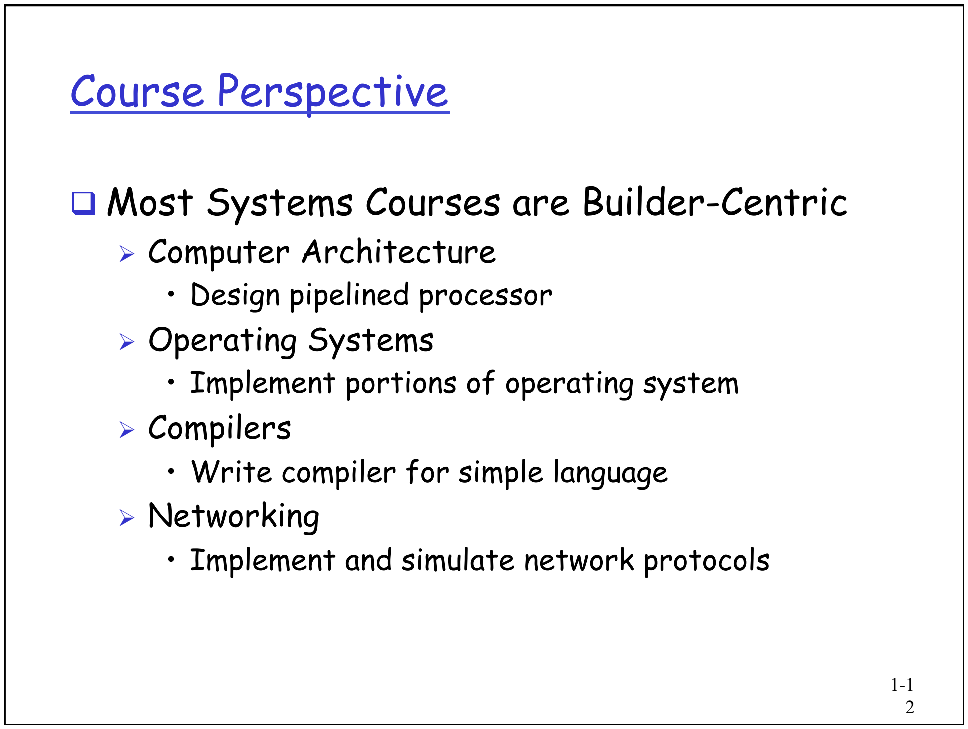
	Alpha	MIPS	Linux
-g	5.30498947741318e-315	3.1399998664856	3.14
-O	3.14	3.14	3.14

(Linux version gives correct result, but implementing as separate function gives segmentation fault.)



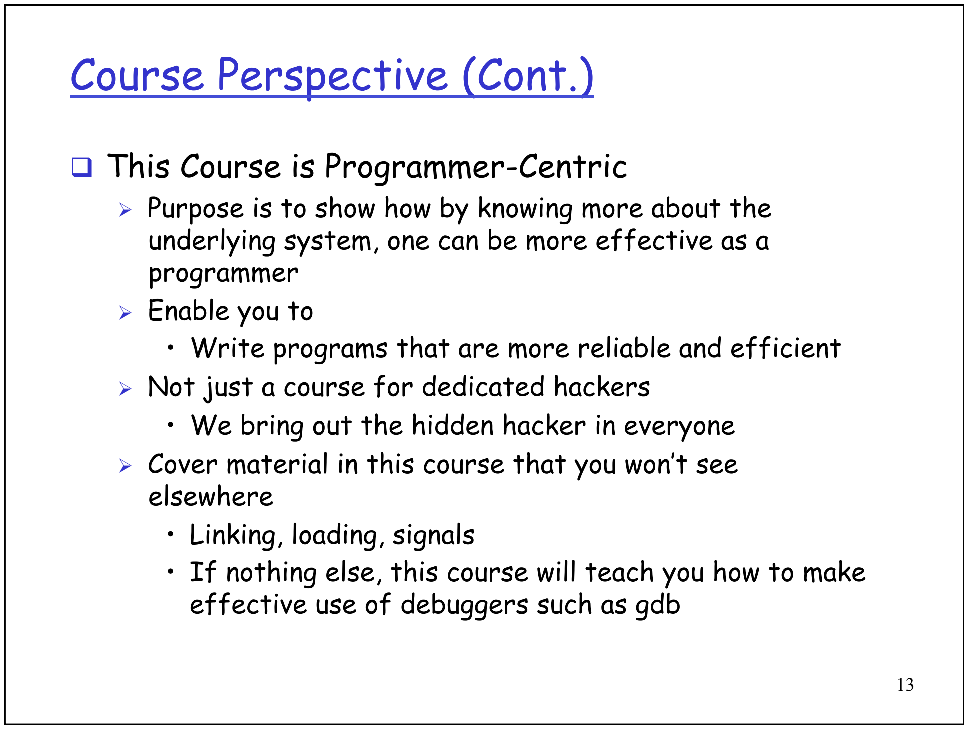
Memory Referencing Errors

- C and C++ do not provide any memory protection
  - Out of bounds array references
  - Invalid pointer values
  - Abuses of malloc/free
- Can lead to nasty bugs
  - Whether or not bug has any effect depends on system and compiler
  - Action at a distance
    - Corrupted object logically unrelated to one being accessed
    - Effect of bug may be first observed long after it is generated
- How can I deal with this?
  - Program in Java, Lisp, or ML
  - Understand what possible interactions may occur
  - Use or develop tools to detect referencing errors



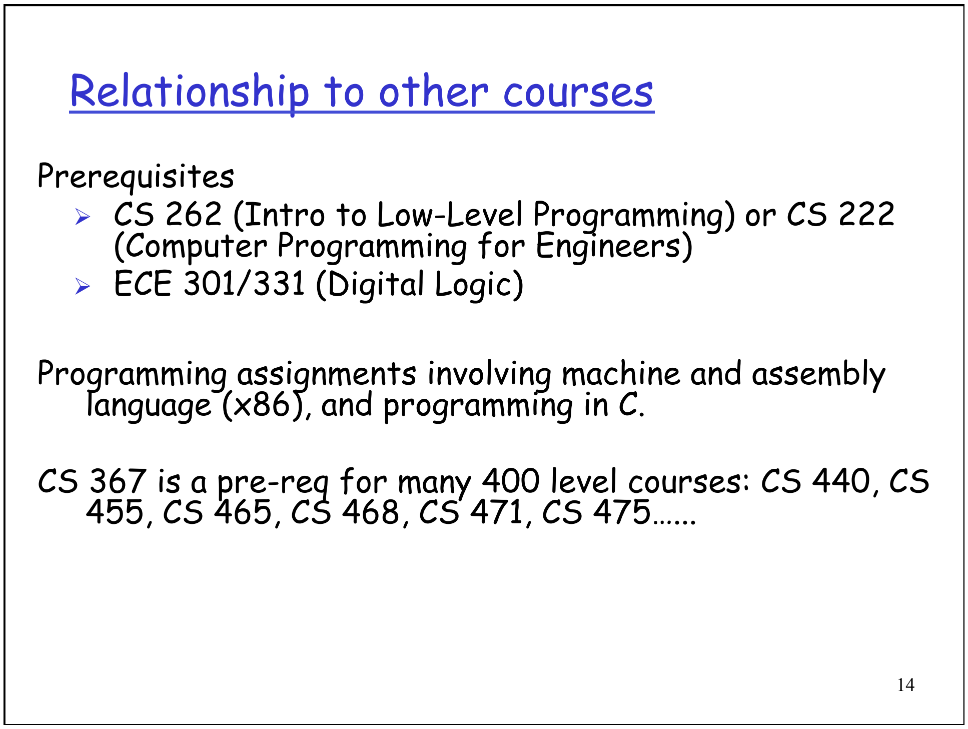
Course Perspective

- Most Systems Courses are Builder-Centric
  - Computer Architecture
    - Design pipelined processor
  - Operating Systems
    - Implement portions of operating system
  - Compilers
    - Write compiler for simple language
  - Networking
    - Implement and simulate network protocols



Course Perspective (Cont.)

- This Course is Programmer-Centric
  - Purpose is to show how by knowing more about the underlying system, one can be more effective as a programmer
  - Enable you to
    - Write programs that are more reliable and efficient
  - Not just a course for dedicated hackers
    - We bring out the hidden hacker in everyone
  - Cover material in this course that you won't see elsewhere
    - Linking, loading, signals
    - If nothing else, this course will teach you how to make effective use of debuggers such as gdb



Relationship to other courses

Prerequisites

- CS 262 (Intro to Low-Level Programming) or CS 222 (Computer Programming for Engineers)
- ECE 301/331 (Digital Logic)

Programming assignments involving machine and assembly language (x86), and programming in C.

CS 367 is a pre-req for many 400 level courses: CS 440, CS 455, CS 465, CS 468, CS 471, CS 475……

Textbooks

- ❑ Randal E. Bryant and David R. O'Hallaron,
  - "Computer Systems: A Programmer's Perspective", Prentice Hall 2nd edition, 2010.
  - csapp.cs.cmu.edu
- ❑ Brian Kernighan and Dennis Ritchie,
  - "The C Programming Language, Second Edition", Prentice Hall, 1988
  - You can use any book on C

Course Outline

- ❑ Programming in C (2 weeks)
- ❑ Data Representation (2 weeks)
- ❑ Program Representation (4 weeks)
- ❑ Linking (1 week)
- ❑ Dynamic Memory Allocation (1-2 weeks)
- ❑ Exceptional Control Flow, Memory Hierarchy, Virtual Memory (3 weeks)

Programming in C

Assumption: You are comfortable programming in Java and are familiar with basic C programming: control flow, procedures, bit-level operators, standard C library for input/output (scanf, printf)

- Topics covered in this class
  - Pointers & Structures
  - Memory allocation and deallocation
- Assignment
  - P1: Write a non-trivial program in C

Machine-level Representation of Data and Programs

Topics

- Bits operations, arithmetic, assembly language programs, representation of C control and data structures
- Includes aspects of of architecture and compilers

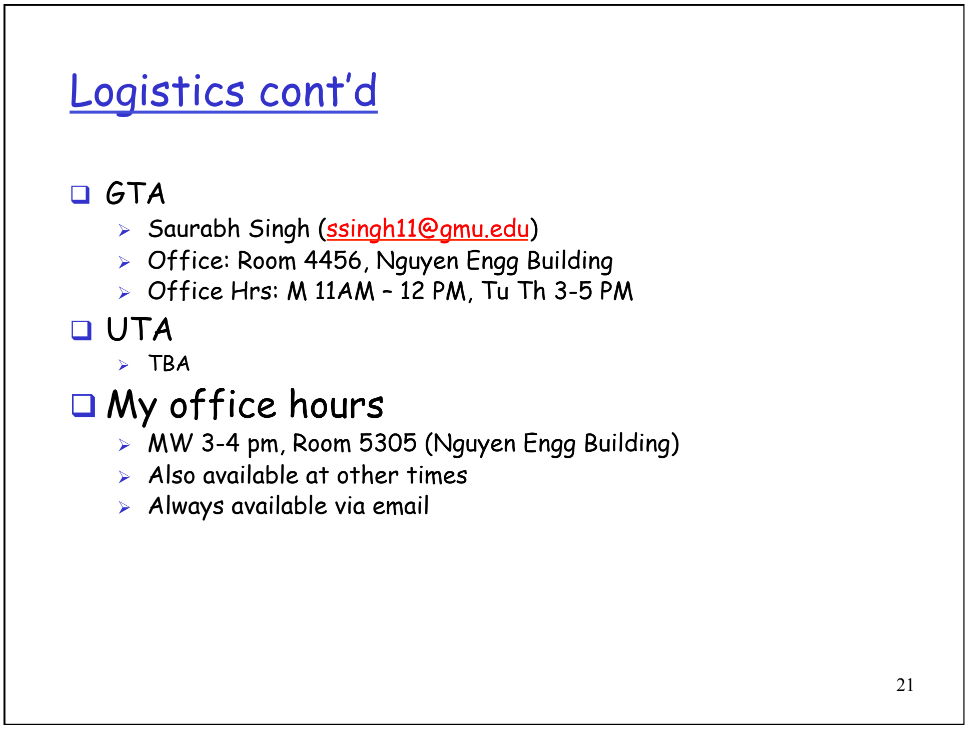
- Assignments
  - HW 1 (Chapter 2)
  - P2: Defusing a binary bomb
  - HW 2 (Chapter 3)

Linking, Exceptional Control Flow, Memory Hierarchy, Virtual Memory

- Topics
  - Object files, static and dynamic linking, libraries, loading
  - Hardware exceptions, processes, process control
  - Dynamic Memory Allocation, Garbage collection
  - Virtual Memory
  - Includes aspects of compilers, OS, and architecture
- Assignment
  - P3: Writing your own malloc() and free()
  - Optional HWs on topics listed above

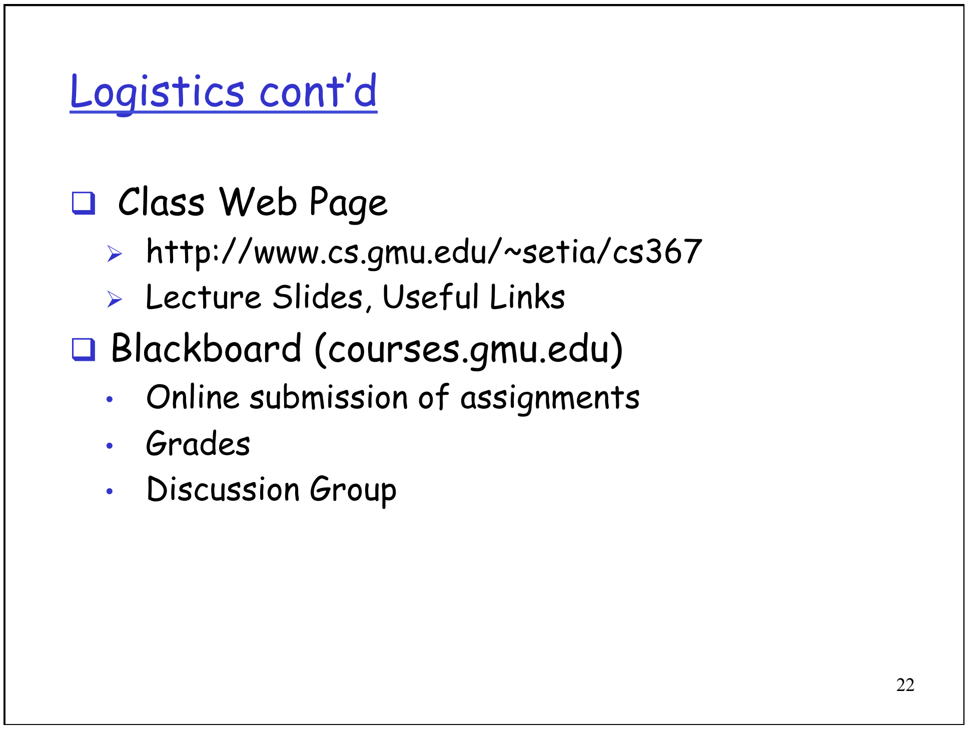
Logistics

- Grading
  - Two or more homework (written) assignments (10%)
    - To be completed individually
  - Three Programming Assignments (35%)
    - first assignment to be completed individually
    - can work in groups of two for remaining
    - All programs will be tested on a Linux platform for grading
      - You need to obtain an IT&E Unix Cluster (zeus) account (See syllabus)
      - zeus is not the same machine as mason2/osf1
  - Midterms (25%)
    - Midterm I - Chapter 2
    - Midterm II – Chapter 3
  - Final (20%)
  - Quizzes and Class Participation (10%)
    - At least four quizzes, not announced in advance



Logistics cont'd

- GTA
  - Saurabh Singh (ssingh11@gmu.edu)
  - Office: Room 4456, Nguyen Engg Building
  - Office Hrs: M 11AM – 12 PM, Tu Th 3-5 PM
- UTA
  - TBA
- My office hours
  - MW 3-4 pm, Room 5305 (Nguyen Engg Building)
  - Also available at other times
  - Always available via email



Logistics cont'd

- Class Web Page
  - http://www.cs.gmu.edu/~setia/cs367
  - Lecture Slides, Useful Links
- Blackboard (courses.gmu.edu)
  - Online submission of assignments
  - Grades
  - Discussion Group

Cheating

- What is cheating?
  - Sharing code: either by copying, retyping, looking at, or supplying a copy of a file
  - Obtaining a solution from the internet
- What is NOT cheating?
  - Helping others use systems or tools.
  - Helping others with high-level design issues.
- Penalty for cheating:
  - Referral to honor council with recommendation for F grade.

Next class

Review of C programming (material covered in CS 262)