Course Staff	e-mail	Location	Office Hours
Mark Morrissey	markem AT pdx.edu	zoom	Tu/Th 12:30 – 1:30
Pan Tan	ptan AT pdx.edu	zoom	M/W 12:00 – 1:00

Course Objectives

1. Describe the major components of computer architecture; explain their purposes and interactions and the instruction execution cycle.
2. Describe a basic instruction set architecture, including the arithmetic, logic, and control instructions; user and control registers; and addressing modes.
3. Do simple arithmetic in hexadecimal, decimal, and binary notation, and convert among these notations.
4. Explain how data types such as integers, characters, pointers, and floating point numbers are represented and used at the assembly level.
5. Write C language programs that use control structures, functions, IO, arrays, and dynamic memory.
6. Describe each step of the compilation process by which C language programs are transformed into machine code.
7. Explain how high-level programming constructs such as arrays, structures, loops, and stack-based function calls are implemented in machine code. Recognize and reverse engineer same.
8. Demonstrate and use a debugger to analyze program flow, inspect register and stack contents.
9. Identify and fix performance issues in C programs that are caused by machine level concepts.

Date	Topic	Readings	Slides	Due
3/30	Introduction to Systems	[1] Ch 1	pptx pdf	
4/1	Intro to C for Systems	[2] Ch 1	pptx pdf	
4/6	Weekly Quiz MetaCTF overview and demo The Compilation Process	[1] Ch 7.1-7.10	pptx pdf	
4/8	Debugging Concepts	No Reading		A1 Submission Instructions
4/13	Weekly Quiz Data Representation Base Types in C	[1] Ch 2.1-2.2 [2] Ch 2 handout	pptx pdf	CTFs Ch 1, 7
4/15	Arithmetic	[1] Ch 2.3	pptx pdf	
4/20	Weekly Quiz Floating Point	[1] Ch 2.4	pptx pdf	
4/22	C Functions	[2] Ch 4	pptx pdf	CTFs Ch 2
4/27	Weekly Quiz Program Encodings / Data Formats	[1] Ch 3.2-3.4	pptx pdf	A2 Submission Instructions
4/29	Arithmetic and Logical Operations	[1] Ch 3.5	pptx pdf	
5/4	Midterm Quiz Comprehensive Control	[1] Ch 3.6	pptx pdf	
5/6	Procedures	[1] Ch 3.7 [1] Ch 3.10	pptx pdf	
5/11	Weekly Quiz Stack-based buffer overflow attacks	No Reading	pptx pdf	A3 Submission Instructions
5/13	Pointers and Arrays	[1] Ch 3.8, [2] Ch 5	pptx pdf	
5/18	Weekly Quiz Structs, Unions, Alignment	[1] Ch 3.9, [2] Ch 6	pptx pdf	
5/20	Optimization in C	[1] Ch 5.1-5.6	pptx pdf	CTFs Ch 3
5/25	Weekly Quiz Embedded Assembly ARM Processor	Embedded Assembly GCC documentation	pptx pdf	
5/27	Dynamic Memory Allocation	[1] Ch 9.9	pptx pdf	
6/1	Weekly Quiz Interrupts, Traps, and Exceptions	[1] Ch 8.1-8.2	pptx pdf	A4 Submission Instructions
6/3	POSIX System Calls, Signals	[1] Ch 8.3-8.5		
6/8	10:15am Final Quiz Comprehensive. Final Exam Schedule			CTFs Ch 5, 8

Course Texts

1. Computer Systems: A Programmer's Perspective, 3rd ed., Bryant and O'Hallaron, Prentice Hall, 2015. ISBN-13: 978-0134092669..

2. The C Book, second edition by Mike Banahan, Declan Brady and Mark Doran, originally published by Addison Wesley in 1991. Available for free from GDB Direct.

Slack Channel

This section of CS201 has its own slack channel, cs201-2021-02. Just search on the channel name once you are logged in. You can create an account if you are a registered PSU CS student.

Access and Inclusion for Students with Disabilities

Accommodations are collaborative efforts between students, faculty, and the Disability Resource Center. Students with accommodations approved through the DRC are responsible for contacting the faculty member in charge of the course prior to or during the first week of the term to discuss accommodations. Students who believe they are eligible for accommodations but who have not yet obtained approval through the DRC should contact the DRC immediately. The DRC may be reached at 503-725-4150 or visit the DRC web page.

Extraordinary Circumstances

If an extraordinary circumstance (for example severe illness) prevents you from working for a period of time, contact us as soon as possible to discuss your situation and arrange a special schedule. Scheduled work commitments do not constitute an extraordinary circumstance. Makeup exams will not be given except in cases of severe and documented medical or family emergencies. Please note that travel is not considered an emergency. If an emergency arises and you miss an exam, contact the instructor before the exam to arrange for a special circumstance.

Testing Accommodation

Students who are eligible to use the SHAC Testing Center are required to take the exam at a time that overlaps with the same time period as the in-class exam. You are encouraged to note the exam dates on the syllabus and make your appointments as soon as you can. We recommend that you make your appointments as early in the term as possible. You can always cancel an appointment but later in the term, a suitable time may not be available. All students using the testing center are required to make their own appointments.

Grading and Exam Policy

No late work is accepted. All assigned work (programs and CTFs) are due at the start of the class period on the day listed on the schedule.

There is no extra credit for this class – keep up with the work and you will not be tempted to ask.

Grade Distribution		**Quizzes**	
A1 (C/libc)	5%	(8) Weekly Quizzes, each	5%
A2 (bit ops)	5%	Midterm Quiz	10%
A3 (floating point)	10%	Final Quiz	10%
A4	10%	Subtotal	60%
MetaCFT	15%	Drop Lowest Weekly Quiz	-5%
Quizzes	55%	Total Quizzes	55%

Online Resources

1. Coding Guidelines
2. Source code from slides. There are also additional example programs on the course handouts page.
3. Makefile example for this class. You can obtain a copy of the Makefile with tabs intact using this command from any PSU CS Linux system:
   wget http://web.cecs.pdx.edu/∼markem/CS201/homework/Makefile .
4. We have a website for practice with the techniques in this course. Use your MetaCTF credentials to log in.
5. C Programming Language Handbook, by Steve Summit.
   There is also an intermediate version of the book.
6. Gnu debugger, gdb
   (a) GDB cheat sheet
   (b) GDB Manual

Academic Integrity

- All assigned work is to be completed individually.
- For assignments, source code plagiarism tools will check that code has not been duplicated.
- Ensure your code is kept private. You are responsible for ensuring that others may not copy your work. Allowing others to copy your work is a form of academic misconduct.

Important University Resources

1. Student Conduct at PSU.
2. Policies & Codes of Conduct at PSU.
3. PSU Student Code of Conduct.

Course Policy on Academic Misconduct

- Academic misconduct includes but is not limited to:

  Allowing another student to copy your work.

  Representing the work of others as your own.

  Receiving and/or providing detailed guidance on any graded work.

- Results in a grade of 0 for the assignment or exam.
- Forwarded to the University for disciplinary action.

Accounts and the CS Tutors

You will need an account to log into the Linux systems provided by the college. If you don't already have an account, go to the CAT student page for instructions. The Linux systems are located in FAB 88-09 and 88-10 as well as remotely using ssh to babbage.cs.pdx.edu and ada.cs.pdx.edu.

The CS tutors are a helpful resource. They cannot help you with assignments but are very good at helping you to interpret compilation errors and use debuggers.

Programming Style

[1] An individual programmer can ease the maintenance burden greatly by writing programs that others can read and modify readily. Good programming style comes only with practice, and we recommend that you begin at once to try writing programs that are easy for others to understand. There is no magic formula that will guarantee readable programs, but there are several useful rules of thumb:

1. Modularize a program into coherent pieces.
2. Lay out a program so that its structure is clear.
3. Write intelligent comments to explain a program. Describe, clearly and precisely, the underlying data models, the data structures selected to represent them, and the operation performed by each procedure. When describing a procedure, state the assumptions made about its inputs, and tell how the output relates to the input[2].

[1]From *Foundations of Computer Science*, Aho and Ullman, Computer Science Press, 1994, ISBN:978-0-7167-8233-9. PDF version available for free at the FOCS website.

[2]However, there is a fine line between good internal documentation and annoyingly useless comments. For example, a variable named `NumberOfDwarfs` doesn't need a comment stating that fact – such a comment does nothing to improve the understanding of the program!

4. Use meaningful names for procedures and variables[3].

5. Avoid explicit constants (magic numbers) whenever possible. For example, do not use 7 for the constants number of dwarfs. Rather, use a defined constant such as `NumberOfDwarfs`, so that you can easily change all uses of this constant to 8, if you decide to add another dwarf.

6. Avoid the use of "global variables" – that is, variables defined for the program as a whole – unless the data represented by that variable really is used by most of the procedures of the program.

Test Suite

Another good programming practice is to maintain a test suite of inputs that will try to exercise every line of code while you are developing a program. Whenever new features are added to the program, the test suite can be run to make sure that the new program has the same behavior as the old on previously working inputs.

[3]Meaningful names can contribute to the understanding of the program's purpose, which provide good documentation on their own.