

EE 1520 Embedded Systems Engineering I

University of Colorado Denver
College of Engineering and Applied Science
Department of Electrical Engineering

Term: Fall 2009

Meeting: Tuesday/Thursday 1:00pm-2:15pm

Course location: NC 2609

Office: NC 2204A

Professor Dan Connors

Email: Dan.Connors@ucdenver.edu

Office Hrs: Tue/Thur 10am-12noon

Mon/Wed 2pm-4pm

Website: Web material will be posted at BlackBoard (EE1520): <http://blackboard.cuonline.edu>

NOTE: All email communication by students must use ucdenver.edu as the email domain, emails from gmail, hotmail, yahoo, etc are NOT considered valid methods of communication.

Course Design

Catalog Description:

Embedded Systems Engineering I. Programming concepts are introduced from a hardware design standpoint. Assembly language and the "C" language are used to interface and manipulate hardware. Microcontroller programming for electrical engineering applications is studied.

Instructor Description:

This is a first course in programming that targets microprocessors and personal computers used in embedded systems having from no operating system to Linux and Microsoft operating systems. The use of microprocessor and personal computer hardware, console and terminal compiler interfaces as well as integrated development environment (ide) compiler interfaces, the C programming language, serial communications to embedded systems, scientific computing and system/software design methods are presented.

Prerequisites:

The background for this course is EE 1510 Logic Design and an understanding of basic programming variables and functions.

Course Objectives:

To develop an understanding of modern programming and associated assembly language, gain familiarity with the use of computers as system components, and develop an ability to cast solutions to problems in terms of a mix of hardware and software

1. Learn problem-solving techniques to develop software for embedded systems.
2. Design, write, compile / cross-compile, download, and run C-programs on micro controllers and PCs.
3. Install and work with compilers on Linux and Microsoft operating systems.
4. Program the communications and control between a micro controller and a PC.

Laboratory Objectives:

To develop skills in: programming in assembly language and high-level languages, communicating with specific devices, and using development environments.

Requirements

Assessment Design/Grades:

- (15 %) Exam I
- (15 %) Exam II
- (25 %) Final
- (20 %) Assignments
- (25 %) Exercises

Grades are as follows:

- A – “Superior/Excellent”, 90 – 100%
- B – “Good/Better than Average”, 80 – 89%
- C – “Competent/Average”, 70 – 79%
- D – “Minimum Passing”, 60 – 69%
- F – “Failing”

Textbook:

C-Programming Language, 2nd edition by Brian W. Kernighan, and Dennis M. Ritchie. ISBN-10: 0131103628 and/or ISBN-13: 978-0131103627

There is an on-line edition of the textbook:

<http://www.docstoc.com/docs/293324/The-C-Programming-Language-Ritchie-Kernighan>

Assignments and Examinations:

Examinations are intended to measure your individual mastery of the material. Exams concentrate on your understanding of the important concepts, rather than your ability to memorize details. All major examinations will be held in class with exact dates determined in class. The exams will generally test your knowledge of assignment material, so you are responsible for mastering all lab, homework, and programming material submitted with other partners, as if you did all the work by yourself. All exams will be open book and open notes (unless otherwise stated). The nature of the course material is such that the final exam must be cumulative.

Reading: The student should budget time weekly for reading the text book and other course material. With 14 weeks of class (excluding exam days and finals week) and a text of 180 pages, the student should plan on 12 to 20 pages a week to stay abreast of the material covered in class.

Text Exercises: Assigned exercises from the textbook, typically requiring 20 to 100 lines of code to extend the programs discussed in the text, are intended to teach and reinforce the learning of the C language syntax and the fundamentals of many operating system programs. Students will code, test, print and submit these exercises weekly.

Assignments: Programming assignments are meant to develop program design and implementation skills. These assignments will be given every one to two weeks and due in one to two weeks (10 assignments for the semester).

Course Policies: Policies regarding class attendance, turning in late work, missing homework, tests or exams, make-ups, requesting extensions, reporting illnesses, cheating and plagiarism, changes to the syllabus. Academic policies will be consistent with the University's policies at the College of Engineering and Applied Science's website: <http://www.cudenver.edu/Academic>

Extensions/make-ups:

In general, late work will not be accepted. Turn in all work by the established deadline. In case you have difficulties finishing an assignment contact the instructor before the deadline. Late work can be accepted only under circumstances beyond student's control and after arrangement with the Instructor, prior to the deadline. Note: work turned-in on time is eligible for partial credit. It will always be better to turn work in by the deadline, as trying to "perfect" it and turn it in late will give you no points at all. You have to follow the submission and media policies and guidelines published on the web. Plagiarism is the passing of someone else's work as one's own, without giving the original author due credit. Scholastic dishonesty will be treated very strictly as per University of Colorado rules.

Students with disabilities requiring accommodations, please contact the Office of Disability Resources & Services located in NC #2514 phone 303.556.3450, TTY 303.556.4766. The staff will assist you in both determining reasonable accommodations as well as coordinating these accommodations.

Students called for military duty-If you are a student in the military with the potential of being called to military service and /or training during the course of the semester, you are encouraged to contact your school/college.

Lecture:

Lecture material (slides,notes) will be made available on the web prior to class. Lecture will also consist of chalk drawings, overhead drawings, and content not explicitly present in slides and notes.

Course Topics

- Processor Organization: We begin by looking at the central processing unit (CPU) using a bottom-up approach (transistor, digital logic, architecture). During that period we examine the CPU architecture, the structure of a CPU Program, and the data types that it understands. Examples are used to illustrate how programs are written in the native language of the machine, and how we control complexity by decomposing a program into modules.
- High-level Programming Language Principles: In computing, a high-level programming language is a programming language with strong abstraction from the details of the computer. In comparison to low-level programming languages, it may use natural language elements, be easier to use, or more portable across platforms. Such languages hide the details of CPU operations such as memory access models and management of scope.
- Assembly Programming Language: An assembly language is a low-level language for programming computers. It implements a symbolic representation of the numeric machine codes and other constants needed to program a particular CPU architecture. This representation is usually defined by the hardware manufacturer, and is based on abbreviations (called mnemonics) that help the programmer remember individual instructions, registers, etc. An assembly language is thus specific to a certain physical or virtual computer architecture (as opposed to most high-level languages, which are usually portable).
- Embedded Systems: An embedded system is a special-purpose computer system designed to perform one or a few dedicated functions,[1] often with real-time computing constraints. It is usually embedded as part of a complete device including hardware and mechanical parts. In contrast, a general-purpose computer, such as a personal computer, can do many different tasks depending on programming. Embedded systems control many of the common devices in use today.

Course Schedule

Class Schedule

Date	Topic	Required Reading	Assignments	Exercises/Tutorial
Week1	Generation Information-Introduction			
Week1	Computer Data Formats	Chapter 1 and 2		Shell Terminal
Week2	Datatypes and Control Flow in C	Chapter 3 to p.54	Assignment 1 Due	
Week2	Functions and Program Structure	Chapter 4 to p.72		Exercise
Week3	C Preprocessor, Compiler, Linker, Loader and Debugger		Assignment 2 Due	Shell pipes, unix commands
Week4	Arrays	Chapter 5 to p.90		Exercise
Week4	Pointers		Assignment 3 Due	
Week5	System Interface/API	Chapter 7		
Week6	Exam Review			
Week6	Exam 1			
Week7	Data Structures	Chapter 6 to p.126	Assignment 4 Due	
Week7	Dynamic Memory Allocation	Chapter 8		
Week8	Assembly Code I	Handouts	Assignment 5 Due	
Week8	Assembly Code II	Handouts		Exercise
Week9	Assembly Code III	Handouts	Assignment 6 Due	
Week9	Optimization			Exercise
Week10	Exam Review		Assignment 7 Due	
Week10	Exam 2			
Week11	Processor Design	Lecture Notes	Assignment 8 Due	
Week11	Memory-mapped I/O			Exercise
Week12	Compression Algorithm	Lecture Notes	Assignment 9 Due	
Week12	Graphics Software Interface	Lecture Notes		Exercise
Week13	Microcontrollers	Notes	Assignment 10 Due	
Week13	Microcontrollers	Notes		
Week14	Microcontrollers			Exercise
Week14	Embedded Design			
Week14	Embedded Design			
Week15	Embedded Design			
Week15	Course Review			Project Exercise
Week16	FINALS WEEK			