ELEC 1520 Embedded Systems Engineering I
University of Colorado Denver
College of Engineering and Applied Science
Department of Electrical Engineering

Term: Spring 2011
Meeting: TR 3:30-4:4 pm
Course location: NC 2609
Office: NC 2522A
Cell Phone: 303.594.3519

Instructor Bob Grabbe
Email: Robert.Grabbe@ucdenver.edu
Office Hrs:
TR 2-2:30, 5-5:30, 8-8:30, and additional times TBA

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 ELEC 1510 Logic Design and an understanding of basic programming variables and functions. The catalogue description for ELEC 1510 is:

- The design of combinational and sequential switching circuits. Topics include Boolean algebra, Boolean function minimization technique, combinational circuit analysis and synthesis, synchronous sequential circuit analysis and synthesis, algorithmic state machine design, asynchronous sequential circuit analysis and synthesis.

Important aspects of ELEC 1510 used in this course include Boolean algebra, combinational logic implementations of arithmetic circuits, twos complement numbers, state diagrams, and algorithmic state machines.

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.

Course Outcomes:
- Understanding of data types and numeric representations
- The principles associated with control flow in C programming
- Design of functions and program structure
- Basic understanding of C Preprocessor, Compiler, Linker, Loader/Debugger
- Design of arrays (single and multiple dimensions)
- Use of pointers to access data
- Working knowledge of system interface/API (printf, scanf, fopen, FILE) and math libraries
  - Ability to implement data structures for improving programming utilities
- Use dynamic memory allocation to provide code capabilities
- Identify, interpret, and implement the software and hardware architecture of a microcontrollers including assembly language programming, debugging, and hardware interfacing.
- Describe and have a working knowledge of the registers, memory maps, timing, decoding, memory addressing, and input/output porting of microcontroller-based systems.

Laboratory Objectives:

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

Requirements

Required textbook:
C Programming – A Modern Approach by K. N. King
Ghttp://knking.com/books/c/

Recommended: (Note it can be download for free, read on.)
There is an on-line edition of the textbook:

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”, 73 – 79%
(C-) – “Minimum Grade to Continue onto ELEC 2520, Embedded Systems II”, 70-72%
D – “Course Does not Apply in College of Engineering, Failing”, 60- 69%
F – “Course Does not Apply at the University, Failing” < 60%

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 500 pages, the student should plan on 25 to 50 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 (8 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 polices 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. 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 bottomup 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, 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**
Tentative Schedule
- **Week Topic**
  - Week 1: Generation Information - Introduction
  - Week 1: Computer Data Formats
  - Week 2: Datatypes and Control Flow in C
  - Week 2: Functions and Program Structure
  - Week 3: C Preprocessor, Compiler, Linker, Loader and Debugger
  - Week 4: Arrays
  - Week 4: Pointers
  - Week 5: System Interface/API, File manipulation
  - Week 6: Exam Review
  - Week 6: Exam 1
  - Week 7: Data Structures
  - Week 7: Dynamic Memory Allocation
  - Week 8: Assembly Code I
  - Week 8: Assembly Code II
Week9 Assembly Code III
Week9 Optimization
Week10 Exam Review
Week10 Exam 2
Week11 Processor Design
Week11 Memory-mapped I/O
Week12 Compression Algorithm
Week12 Graphics Software Interface
Week13 Microcontrollers
Week15 Embedded Design
Week15 Course Review