UNIVERSITY OF CALIFORNIA, SANTA CRUZ
BOARD OF STUDIES IN COMPUTER ENGINEERING

CMPE-012/L: COMPUTER SYSTEMS AND ASSEMBLY LANGUAGE PROGRAMMING

WINTER 2018 — COURSE INFO AND SYLLABUS

DETAILS:

Instructor	Gabriel Hugh Elkaim	elkaim+cmpe012@soe.ucsc.edu
Class Lecture	T-Th 9:50 – 11:25AM	Classroom Unit 2
Class Lab	Various Times	Ming Ong, SocSci. 1-135, JBE 109

OVERVIEW:

This course presents an introduction to computer systems and assembly language. That is, how computers work, how they achieve computation, what limitations this imposes, and to some extent how this changes the performance of the machine. We will also be exploring how to instruct the computer to do these computations at a very low level (assembly), and exploring both software and hardware interactions. At the end of the quarter, you should have a good machine model of the computers you use, and how they work at a very low level. Note that we will not dive into the semiconductor physics, but remain at a logic gate level.

The knowledge gained in this class will inform your career and increase your understanding of the limitations and methods of computation in general. This is useful background knowledge that will improve your understanding of almost all subsequent computer programming and architecture classes.

We will be using the MIPS assembly language throughout the class and using MARS as the simulator to code the labs. Note that this class has a prerequisite of prior programming experience (such as CMPS-5C/J/P). While we will be writing code in MIPS assembly, knowledge of programming is required to succeed. We will be using GIT for version control and also for lab submission; this is an industry standard tool for version control and learning it is an important part of the class.

The class is fast paced due to the breadth of material covered; keep up with the class and do the readings. Do NOT wait to start your programming assignments until the day before they are due; most students who have difficulty in this class wait until the last minute and don't have sufficient time to get help.

If you have any disability-related needs, be sure to contact the Disability Resource Center well in advance of any expected need.

TEXT BOOKS:

There is a limited time in lecture, and you are expected to supplement that knowledge with the reading for the class. To do well, you must read the books for this class. The required texts are:

[Patt and Patel]: Yale N. Patt and Sanjay J. Patel*, Introduction to Computing Systems: From Bits and Gates to C and Beyond (Reader)*, McGraw-Hill Education, First Edition, ISBN 978-1307117530, 2017.

This is an abridged version of the textbook previously used for a much cheaper price. If you have access to the full book the chapters taken are identical.

[Patternson and Hennessy]: David A. Patterson and John L. Hennessy, *Computer Organization and Design MIPS Edition: The Hardware/Software Interface*, Fifth Edition, ISBN 978-0124077263, 2013.

This is a very good textbook on Computer Organization which goes much deeper in detail than we will cover.

[Kann]: Charles W. Kann, *Introduction to MIPS Assembly Language Programming*, Gettysburg College Open Educational Resources, 2015. This is a free MIPS programming book that can be found at http://cupola.gettysburg.edu/oer/2/

You will need to read this book to understand MIPS assembly language.

In addition to the required texts, there are a few recommended texts that will help you with the background knowledge of GIT and other useful topics:

- Richard E. Silverman, *Git Pocket Guide*, O'Reilly, ISBN 978-1-449-32586-2, 2013
- Jon Loeliger and Matthew McCullough, *Version Control with Git: Powerful tools and techniques for collaborative software development*, O'Reilly, Second Edition, ISBN 978-1-449-31638-9, 2012.

GRADING:

The lab and the lecture part of the class are one cohesive unit; the grade you receive in one is the grade you receive in the other (consider it a single 7 unit class). The grading is comprised of the labs, the midterm, and the final.

- Labs – 50% of class grade
- Midterm – 25% of class grade
- Final – 25% of class grade

In order to pass the class, you will need to demonstrate basic mastery of the subject matter as demonstrated by your labs and exams; no one may pass the class with less than 50% average of your midterm and final scores. Likewise your average lab scores must be above 50% to pass the class. Note that these are not sufficient to pass the class, but you cannot pass the class without meeting these minimal criteria (necessary but not sufficient).

Re-grading of lab assignments will only be done if we have a clerical error (i.e.: we added points wrong) or we somehow missed your work. Note that this does not include you submitting the wrong commit ID in the Google Form for your lab. We will allow a single exception to resubmitting the correct commit ID per student once per quarter.

Lab Requirements:

The labs are to a large extent where you will master the class subject matter. Note that due to the size of the class, most of the assignments will be machine graded. As such, you will be required to comply with the auto-grading software in order to get any points in the lab. There is a hard requirement on each and every lab to have the correct files in the correct directories.

Each lab will specify a "**MINIMUM SUBMISSION REQUIREMENTS**" that will be checked automatically. You will receive an email after the due date and will be able to use your grace period to fix any issues. Failure to meet this requirement will result in failing the class regardless of any other scores. Typically this requirement will be to have the correctly named files in the correctly named directories but you should check each lab assignment carefully to make sure you have precisely (not approximately) followed its MINIMUM SUBMISSION REQUIREMENT.

We have set up a webpage that will check your repo for MINIMUM SUBMISSION REQUIREMENTS that you can use to verify that you have, in fact, turned in the correct files. See the CMPE012_GIT document for details.

Again, having these files in the correct place and with the correct names does not mean you pass the lab, but is a necessary requirement (necessary but not sufficient). These files need to be there for **each and every** lab to pass the class. Because of the size of the size of the class, there is no flexibility on this. If you cannot master these simple requirements, then you do not have the required prerequisite knowledge to succeed in this class.

All labs will require specific input and output formatting. Again, due to the nature of auto-grading, formats need to be adhered to. Each lab assignment will have the input/output specification called out, and will have at least three examples in the assignment. If there are ambiguities, post on Piazza and they will be resolved.

Assignments are designed to take more time than your lab section to complete; the lab section is to give you guidance and help, it is expected that you will go off and program on your own both before and after sections. The total time given to complete the labs is sufficient if you start early; starting just before it is due is a recipe for failure.

Lab extensions will rarely be given, and will only be given if we have sufficient evidence from the GIT commits/Google form to demonstrate that you have (collectively) been working on it the entire time. Learn to commit early and often. If we see evidence that too few started the lab early (in the form of commits and pushes), than no extension will be considered.

Grace Period:

We understand that sometimes things in life occur that are beyond your control, and you will need extra time not because you waited until the last minute to start. In order to mitigate this each student is granted 72 hours of "grace period" to be used throughout the quarter on your lab assignments. It is consumed in 6 hour increments, and you will be able to check the "late days" assignment on CANVAS to see how many hours you have left.

Specifically, we do NOT need to know why you are using your grace period. If there is some extenuating circumstance that is not covered by the grace period, we need to be informed of this ASAP (not after the fact).

Using GIT:

We have a supplementary document on using GIT, which explains the mechanics of using it specifically in this class. GIT is an industry standard version control tool, and will prevent you from having data loss in your coding assignments. It is also how we will be having you turn in your code for grading (both pushing it to the server and submitting the Commit ID on a google form). Pushing the code to the server safeguards it from loss, and the Commit ID tells us which version of your code you would like us to grade. Note that both are required to turn in your assignment.

You are encouraged to commit and push your repo early and often. There is no penalty for doing so, and it will make sure that you have met the minimum file requirements for the lab. Furthermore, we will be monitoring the activity on the server when deciding if a lab extension is merited. If you don't push your code, we have no idea you have been working on it. Learn to use GIT wisely, check your code in frequently (and start it early). A small portion of your lab grade will be based on how often you committed your code using GIT.

Getting Help:

This is a challenging class for many students; there are resources available to you for help. First off, ask questions in the lecture and lab sections. TAs and the instructor will hold office hours every week. There are lab sections with lab tutors (students who took the class and did well in the class) every week. Seating in lab sections is prioritized to students who are enrolled in that section, but otherwise you are welcome to any section.

Lastly, there is a Piazza forum for the class that is quite active and monitored by the TAs and the instructors. If you have a need to contact the teaching staff, do so with a private post on Piazza. Any email request will be bounced to Piazza, and answered there (you will be asked to post the question on Piazza). Likewise if a question has already been answered on Piazza, your question will be referred to the already posted answer or simply deleted (again, there are lots of you and we are trying to keep the information channel open).

Don't post code onto Piazza; if you need one of the teaching staff to look at your code, post it in a private message to the teaching staff.

If you do need to email the class instructor, use the elkaim+cmpe012@soe.ucsc.edu address. All emails from the class will be automatically filtered and all emails without the +cmpe012 in the address will be deleted before I ever see them. Emails will be checked at least once per business day; expect an answer by the end of the next business day. Piazza is a much faster form of communication.

Note that Piazza is a public, professional forum; please treat your communication with others with a modicum of dignity and respect. You are not anonymous. You ARE training to be a professional, treat each other with respect.

UC Santa Cruz is committed to creating an academic environment that supports its diverse student body. If you are a student with a disability who requires accommodations to achieve equal access in this course, please submit your Accommodation Authorization Letter from the Disability Resource Center (DRC) to me privately during my office hours or by appointment, preferably within the first two weeks of the quarter. At this time, I would also like us to discuss ways we can ensure your full participation in the course. I encourage all students who may benefit from learning more about DRC services to contact DRC by phone at 831-459-2089, or by email at drc@ucsc.edu.

LECTURE VIDEOS AND SLIDE COPIES:

The class lecture will be recorded and the slide decks recorded using the screen capture capability of the Surface computer. These will prove to be useful when reviewing the material for midterm and final, or merely to go back and go over material to refresh it in your mind. However, in order to keep attendance in the lectures, the videos and class slides will be uploaded with a one week delay (they will be on the class website, not on CANVAS).

ACADEMIC HONESTY:

Academic honesty is a requirement for the course. All material produced must be your own independent work; this includes homework/quizzes, exams, and lab assignments.

What is cheating? It is presenting work that is not yours as your own. You can and are encouraged to discuss and strategize with your colleagues on the material and labs, but your work should be your own. Copying is NEVER acceptable. On the labs, cheating is sharing code unless explicitly told that it is permitted. If a student is caught cheating in either the class or the lab this will result in an immediate failure in the class and the lab. It will be reported to your college and your department. DO NOT CHEAT; it is not worth it.

There will be a cheating quiz presented during the first lecture when we will go over specific scenarios, and if they are or are not cheating. We encourage you to ask questions for clarification during this lecture. Because we have had issues with academic integrity in the past (not unique to UCSC), all code turned in will be run through an automated code checker. Similarities will be flagged and academic misconduct charges will be filed.

Your Lab0 assignment will require you to check in this syllabus and the Personal Responsibility document into the repo and indicate that you have read it. No further work will be accepted from any student who has not completed this lab.

CLASS TOPIC SCHEDULE (APPROXIMATE):

The pace of the class will depend largely on how the students engage with the class, and how in depth we have time to cover the topics. Also, since the course has been redesigned and the underlying programming language changed, this is approximate at best. We will do our best to keep topics announced in CANVAS.

History and Introduction
Numbering Systems
Binary Numbers
Data Representation
Digital Logic
Digital Logic Gates in Transistors
Digital Logic Structures
ALU Computations
Computing Overview
MIPS ISA
Intro MIPS Programming
Arrays and Stacks
Function Calls / Macros
MIPS Instruction Decoding
MIPS Architecture
IO and Exceptions / Traps / Interrupts
Number systems revisited
Fractional Binary
Floating Point Representation
Floating Point Arithmetic
Sequential Logic / Boolean Algebra

ACKNOWLEDGEMENTS:

I would like to acknowledge the contributions of many previous instructors who helped design this class and the coverage therein. Specifically I would like to acknowledge Lecturer Max Dunne who has helped in the design of the class, and especially the back end that makes it possible to scale to this size. Professor Miller has been instrumental in getting GIT up and behaving for a class this size.