CE290I Assignments: The overall goal is to develop a minimalistic but fully functional, concurrent software system consisting of virtual machine and compiler that could in principle be used as foundation for cloud computing.

0. Familiarize yourself with the version control system git and create personal accounts on github.com for hosting your software and on drone.io for running your code in the cloud. Development is local on your machine. A terminal and text editor (e.g. Sublime or Emacs or Vim) plus local installations of git and a compiler for your choice of programming language are sufficient. Please check that the compiler is also available on drone.io. And, be brave! We recommend *not* to use any integrated development environments (IDEs) such as Eclipse. Form a team of 2-3 students, name the team, and send us that name as well as the names and github.com and drone.io credentials of its members.

Goal: Learn how to develop code professionally in a team and understand every single command, operation, action, error, and warning involved in coding.

Deadline: Friday, September 5, 2pm

1. Pick any programming language you like and implement a DLX emulator as discussed in class. The emulator must be able to load and execute an executable written in DLX binary or assembly form.

Goal: Understand the basic principles of machine architecture by implementing one. The emulator will serve as target platform for the C* compiler that you will implement later. C* is an extremely simple, close-to-minimal subset of C that will be defined in class.

Deadline: Friday, September 19, 2pm

2. Implement an iterative, linear-time, a recursive, exponential-time, and a recursive, logarithmic-time version of Fibonacci in DLX assembly. Argue why your implementations are indeed linear-, exponential-, and logarithmic-time and state and explain their space complexity as well. Demonstrate that your code runs on your DLX emulator and what difference their computational complexity makes.

Goal: Understand the basic principles of algorithms (not programming languages) and the temporal and spatial cost of (raw) computation.

Deadline: Friday, September 26, 2pm

3. Implement a self-compiling C* compiler that targets your DLX emulator as discussed in class. This may sound scary but is actually easier than you think.

Goal: Understand the full depth of programming language semantics (rather than just syntax) by implementing a minimal yet Turing-complete subset of C.

Deadline: Friday, October 24, 2pm

4. Implement process, thread, and virtual memory support in your DLX emulator effectively turning it into a virtual machine. Invent and implement a minimal communication mechanism for processes to overcome their spatial isolation. This may also sound scary but is in fact much easier than implementing a kernel that does that.

Goal: Understand the basic principles of operating system kernels without getting distracted by bootstrapping and self-reference issues of real kernels.

Deadline: Friday, November 7, 2pm

5. Implement process and thread support in your C* compiler based on your DLX virtual machine. Rewrite your three Fibonacci implementations in C* and parallelize them using processes and threads, argue about their temporal and spatial complexity, and demonstrate that your code runs on your DLX virtual machine.

Goal: Understand the fundamentals of concurrency and memory management as well as virtualization and how compilers and operating systems interact to enable concurrency and virtualization.

Deadline: Friday, November 28, 2pm