```
################################################################################
# CS:APP Architecture Lab
# Directions to Instructors
#
# Copyright (c) 2002, 2011, 2015 R. Bryant and D. O'Hallaron,
# All rights reserved. May not be used, modified, or copied without
permission.
#
################################################################################
#

This directory contains the files that you will need to run the
CS:APP
architecture lab, which develops a student's understanding of
processor design and the close relationship between software and
hardware.

************
1. Overview
************

In this lab, students learn about the design and implementation of a
pipelined Y86-64 processor, optimizing its performance on a benchmark
Y86-64
array copy program called ncopy.ys. Students are allowed to make any
semantics preserving transformations to the benchmark program, or to
make enhancements to the pipelined processor, or both. The goal is to
minimize the number of clock cycles per array element (CPE).

********
2. Files
********

Makefile                Makefile that builds the Lab
README                  This file
archlab-handout/        The files handed out to the students
grade/                  Autograding scripts
simguide/               CS:APP Guide to Simulators document
src/                    Master distribution of the Y86-64 tools
sim/                    Student distribution of the Y86-64 tools
(subset of src)
writeup/                The architecture lab writeup

*******************
3. Building the Lab
*******************

Step 1: Build the "master distribution" of the Y86-64 tools in
directory
./src on your system. The master distribution is the instructor's
version of the tools that contains the solution files for the lab.
See ./src/README for instructions on how to build the master
distribution. The process involves setting three variables in
./src/Makefile and then typing "make". The default values are for
Linux.

Step 2. Modify ./src/Makefile-sim with the same three variable
assignments that you used in ./src/Makefile.  The Makefile-sim file
is
the Makefile that the students will use their in their personal
student distributions of the Y86-64 tools. The student distribution
is a
subset of the master distribution, minus the solution files and a
master set of HCL files.

Step 3. Modify ./src/Makefile-handout with the default team name
(TEAM), default handin version number (VERSION), and the directories
where the three parts should be copied to when they are handed in
(HANDINDIR-PART{A,B,C}). The Makefile-handout file is the
the Makefile that the students receive in the archlab-handout/.
They use it to hand in their solutions.

Step 4: Modify the Latex writeup in ./writeup/archlab.tex to reflect
the handout and handin directions for your site. If you don't use
```

Latex, use your favorite document preparation system to prepare
Postcript and PDF versions of the writeup in archlab.pdf.

Step 5: Modify ./Makefile with the name of the lab (LABNAME) and the
name of the directory where the handout tarfile will be copied to and
where the students can pick it from (DEST). LABNAME is typically
archlab.

Step 6: In the same directory as this README file, type

        make clean; make

to the shell. This will do the following things:

(a) Compile the master distribution of the Y86-64 tools in ./src
(b) Build the student distribution in ./sim
(c) Build a tarfile of the student distribution in sim.tar
(d) Build a handout directory ./$(LABNAME)-handout
(e) Build the $(LABNAME)-handout.tar file that you will be
    handing out to students.

Step 7: Type "make dist" to copy the $(LABNAME)-handout.tar file
and the writeup to the distribution directory where the students
will retrieve the lab.

****
4. Grading the Lab
****

There are Perl autograding scripts for each part of this lab.
See ./grade/README for details.

****
5. Notes
****

* If you are running in GUI mode, you'll need to install Tcl/Tk along
  with the Tcl and Tk developer's packages. On an Ubuntu machine:

  linux> sudo apt-get tcl tcl-dev tk tk-dev

* If you are running in GUI mode on a system with X windows, make
sure
  that your DISPLAY environment variable is set:

  linux> setenv DISPLAY mymachine.myschool.edu:0

  If you're using ssh, you can set the DISPLAY variable automatically
by
  using the "ssh -X hostname" command.

* The lab compiles with no warnings and passes all regression tests
on
  the following systems:

  Linux/2.2.20, Pentium III, gcc 2.95.3, Gnu make, Perl 5
  Solaris/5.8, Sparc Ultra 80, gcc 2.95.3, Sun make, Perl 5
  Linux/2.6.18, Intel x86_64, gcc 4.3.5, Perl 5.8.8

* The autograders for parts B and C run the regression tests in
ptest/
  as part of the evaluation of the student solutions. On older
  systems, these tests can take a long time, 5 minutes or so, with
  most of the time being spent by the htest.pl script. If this is too
  long, you can sacrifice completeness for speed by commenting out
the
  call to htest.pl in ptest/Makefile.