ECE 353 Lab 2 Summary

Fall 2015

Fill out this form and upload it to Moodle.

Student Names:\_\_Sarah Mangels, Matteo Puzella, Aaron Lucia

1. Check off all the following functions which you believe work correctly in your code. These should each have passed some basic test cases. Each function should have one person principally responsible for writing the code and another who is responsible for checking its correctness and running test cases on it. Write their last names below. Authoring and testing responsibilities should be divided fairly evenly among group members. The principal author of a function should NOT be the tester for that function.

* 1. X☐ main() Author\_\_\_\_\_Aaron\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Tester:\_\_All\_\_
  2. ☐X progScanner() Author:\_Aaron\_Tester:\_\_Matteo\_\_\_
  3. ☐ X parser()Author:\_Sarah\_\_\_\_ Tester:\_\_\_\_Aaron
  4. ☐X IF()Author:\_Matteo\_\_ Tester:\_\_\_Aaron\_
  5. ☐X ID()Author:\_Sarah\_\_\_ Tester:\_\_Matteo\_\_\_
  6. ☐X EX()Author:\_\_\_Aaron\_\_ Tester:\_\_\_\_Sarah
  7. ☐X MEM()Author:\_\_\_Aaron\_\_\_ Tester:\_\_\_Matteo\_\_\_
  8. ☐X WB()Author:\_Matteo\_\_ Tester:Sarah\_

1. In the space below, draw a call graph of your code. This is a diagram which specifies which functions are called by which other functions. If A calls B, for example, there would be an arrow from node A to node B.

main

verifyInstruction

parser

instructionFetch, instructionDecode, execute, memory, writeBack

regNumberConverter

1. As a basic check of the correctness of your code, you should apply one or more test programs for which you know the execution time by construction. Indicate briefly what kind of test program(s) you used. Did your program pass all your test cases?

We created multiple test assembly instruction files which we calculated the results by hand for each timestep. We then stepped through the code to compare the expected results with the actual. After debugging and fixing errors, all our tests passed. We also ran these files using batch mode and they were successful. Additionally we wrote unit tests for individual function in our code.

1. In one of the lectures, we covered assertions as a way to identify errors in the program. Indicate all assertions you used in your code.

Rather than using assertions for most errors we used if statements so that our code could print out error messages rather than just exiting. We also have functions which checked for errors and printed out error messages if they catch any (for example verifyInstruction).

assert(pgm\_c > 0 && pgm\_c < MEM\_SIZE); // check for branch to invalid address

assert(state->result % 4 == 0); // check reading valid mem address during mem