ECE 353 Lab 2 Summary
Fall 2015

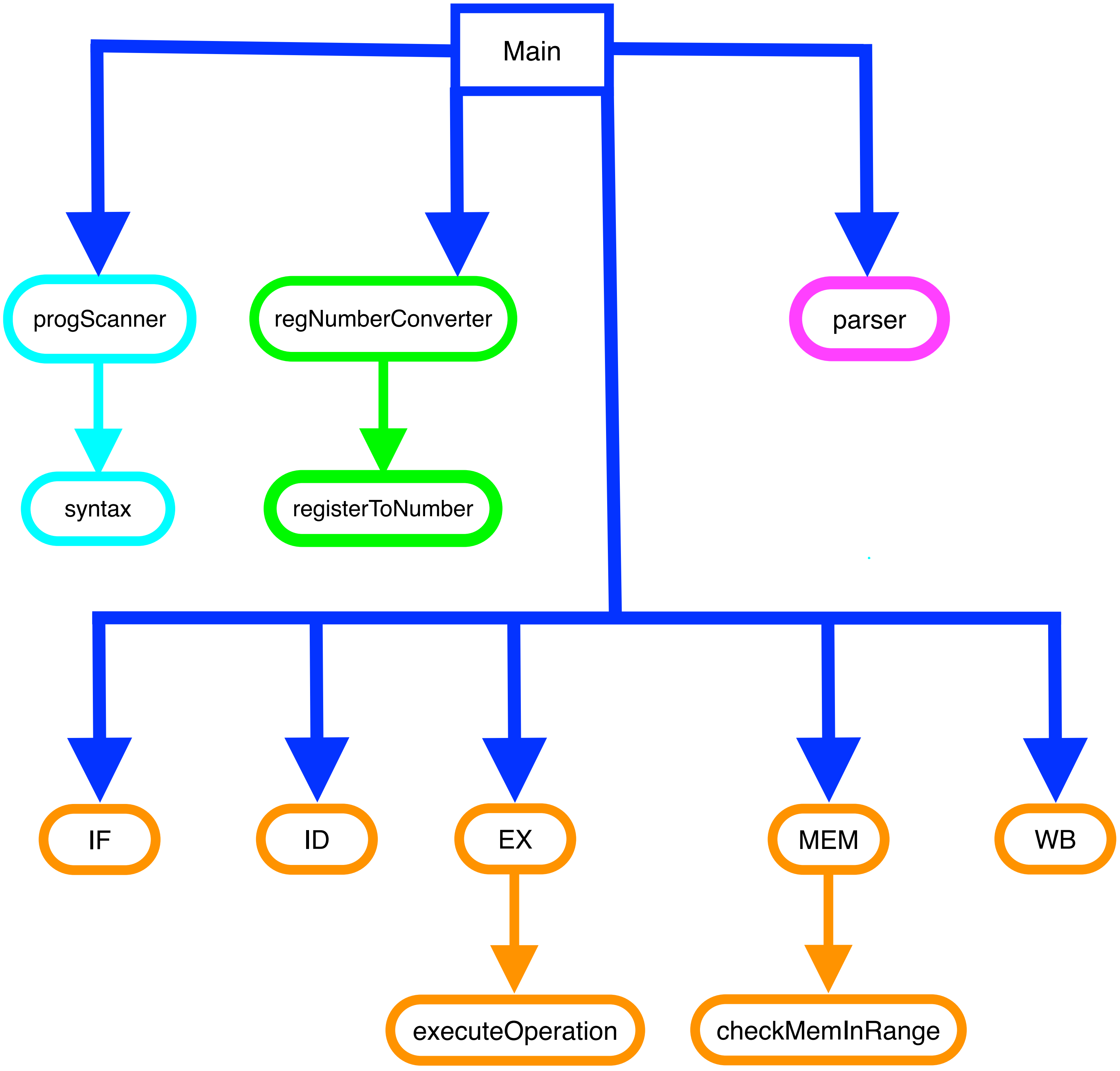
Fill out this form and upload it to Moodle.

Student Names: Evan Tyra & Patrick Sullivan

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. ☐ `main()` Author Evan Tyra Tester: Patrick Sullivan
2. ☐ `progScanner()` Author: Evan Tyra Tester: Patrick Sullivan
3. ☐ `parser()` Author: Evan Tyra Tester: Patrick Sullivan
4. ☐ `IF()` Author: Patrick Sullivan Tester: Evan Tyra
5. ☐ `ID()` Author: Patrick Sullivan Tester: Evan Tyra
6. ☐ `EX()` Author: Evan Tyra Tester: Patrick Sullivan
7. ☐ `MEM()` Author: Evan Tyra Tester: Patrick Sullivan
8. ☐ `WB()` Author: Evan Tyra Tester: Patrick Sullivan

2) 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.



3) 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?

Instruction Set 1

```
addi $t1, $zero, 7
addi $t2, $t2, 3
add $t3, $t2, $t1
sub $t4, $t1, $t2
mult $t5, $t3, $t4
sub $t6, $t5, $t4
mult $t7, $t5, $t6
add $s0, $t3, $t4
add $s1, $t4, $t5
sub $s2, $s1, $s0
haltSimulation
```

Cycle count = 25 total cycles

We used the instruction set above to test the correctness of the program. The benefit of using this instruction set for the official test was the various RAW hazards between the instruction as well testing a variety of functions. We also used another instruction set for verifying LW, SW, and BEQ functionality.

The test we used verified that the cycle count, utilization count, and final register value were at expected values that we derived from hand. The program passed all tests. We also used a variation of the instruction set above to test for edge cases in regards to memory accesses (LW, SW, and BEQ). The program behaves as expected and meets all specified requirements.

4) 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.

```
assert (multiplyTime > 0)
assert (executeTime > 0)
assert (memoryAccessTime >0)
assert (instruction != NULL)
assert (instruction[0] != ‘,’)
assert (opcode != NULL)
assert (commaCount == 1)
assert (paranLeftCount == 1)
assert (paranRightCount == 1)
assert (rightParanPlace > leftParanPlace)
assert ( (secondCommaPlace - firstCommaPlace) > 2)
assert (paranLeftCount == 0)
assert (paranRightCount == 0)
assert (commaCount == 2)
```

We had many other exits but we instead did a print(“Description”), then an exit(1); call within our code.