ECE 353 Lab 3 Summary
Fall 2018

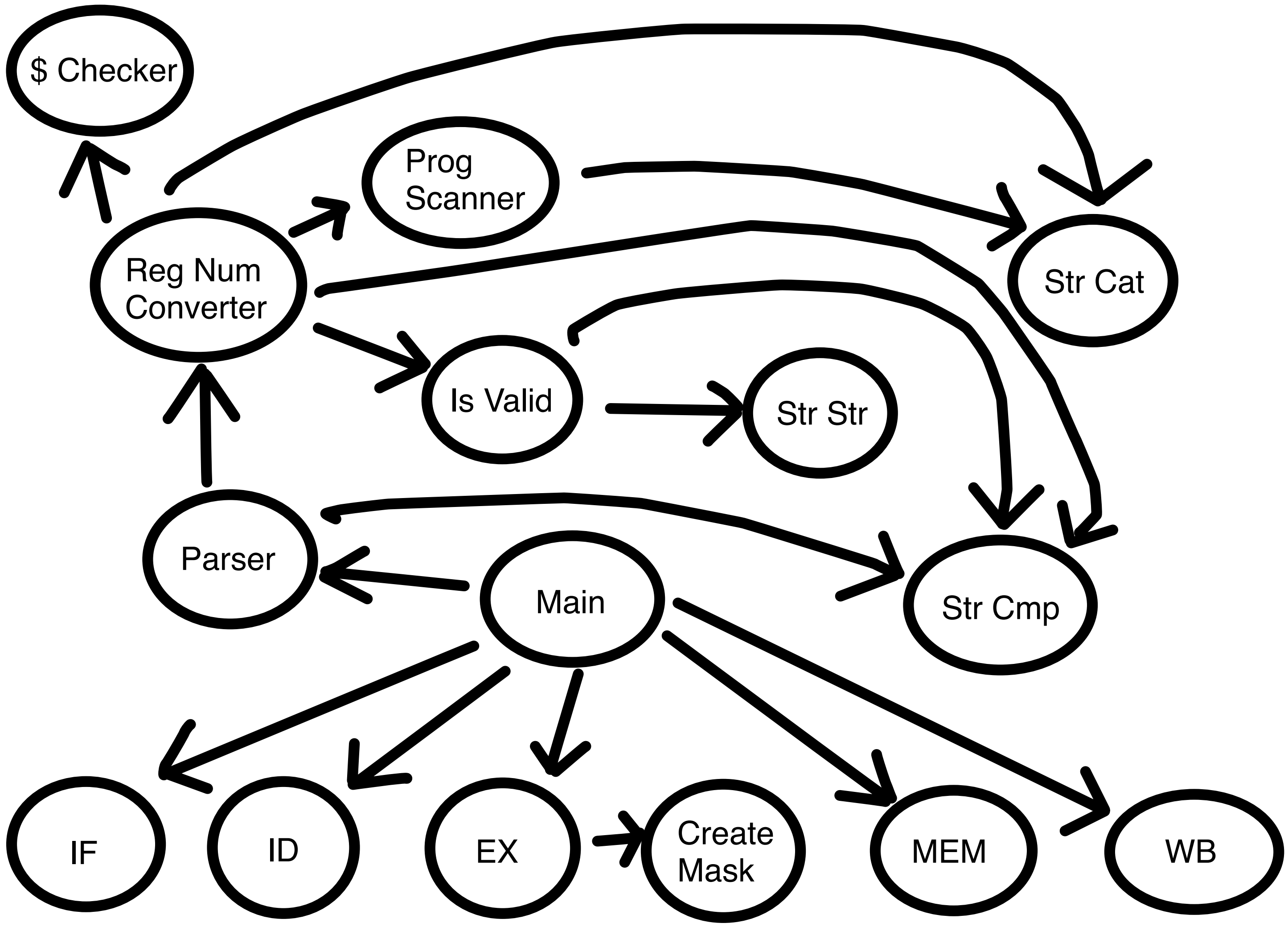
Fill out this form and upload it to Moodle.

Student Names: Dhimiter Shosho, Andrew Jewers, Alex Donadio, Sam Fick

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 Everyone Tester: Everyone
2. ☑ **`progScanner()`** Author: Everyone Tester: Everyone
3. ☑ **`parser()`** Author: Everyone Tester: Everyone
4. ☑ **`IF()`** Author: Everyone Tester: Everyone
5. ☑ **`ID()`** Author: Everyone Tester: Everyone
6. ☑ **`EX()`** Author: Everyone Tester: Everyone
7. ☑ **`MEM()`** Author: Everyone Tester: Everyone
8. ☑ **`WB()`** Author: Everyone Tester: Everyone

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?

To test our program we constructed each function while feeding it our own mips instructions from a text file to test what we implemented along the way. For example, we made progScanner( ) by testing to see if it could take instructions without formatting errors. Once it could, we would then implement and test for one of the required errors we had to account for until every formatting error was covered. We used this approach for all other functions. Furthermore, we would test prior functions with new functions since some functions build off the former. We would not move on from a function until its performance was satisfactory, and if a new function was causing issues we would debug with print statements to see if some code was reached or if a piece of code was implemented correctly.

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.

We opted to use print lines instead of assertions to identify errors. For example, to see if our program was reaching or going into a certain portion/piece of code we would put a print line within the code. If we saw our print statement appear or the expected output when we ran our program then we knew our code was working properly.