CS 445 – Spring 2023

Homework 7

Due on or before 5 May at 11:59 PM Pacific Time.

1. The problem

In this assignment we will be generating three-address TM code corresponding to any legal C- input program. The three-address TM code generated by your compiler must maintain semantic closure with the C- input program provided by the user.

1.1 Required Changes

Your compiler must generate three-address TM code for any legal C- program. The compile statement will generally be of the form:

```
c- testfile.c-
```

with no options. In this case, your compiler should produce a file named `testfile.tm` containing the three-address TM code corresponding to the program contained in `testfile.c-`. The three-address TM code produced by your compiler must be able to be interpreted by the TM virtual machine (version 4.6a). The TM is described in a document located on the course website, and the source code for the TM that will be used to interpret your generated code is also located on the course website. The document that describes the TM also contains three-address code idioms and examples of specific use cases.

1.2 Expectations

Now that you have reached the code generation phase, assessment of your compiler will be conducted differently than in previous assignments. The three-address TM code generated by your compiler will be interpreted by the TM virtual machine and the results of this interpretation will be compared with expected results. The data archive provided with this assignment contains many examples that you can use to test your compiler. In this archive is a script named runExamples that is very similar to the script that will be run on the test machine. This script contains the logic used to determine whether the three-address TM code generated by your compiler behaves correctly for a given input file. If the behavior of your generated code diverges from that of the expected result, any differences will be printed and displayed side-by-side as in previous assignments.

You have many examples in the data archive for this assignment. You have the source code for the TM that will be used to interpret your generated code. You have the runExamples script that will be used to evaluate the behavior of the code that your compiler generates. Since the TM code generated by your compiler will not be compared with the TM code generated by a working compiler, you are free to generate code that contains instrumentation and facilitates debugging while still maintaining semantic closure with the input program. You should use comments in generated code to increase your understanding of the source locus for a particular instruction or set of instructions that your compiler

generates. You might also consider including comments that dump state information related to your compiler associated with a particular block of generated code. The debugging features that are incorporated into the TM virtual machine will be extremely helpful if used wisely. You are free to use the emitCode.cpp and emitCode.h contained on the course website to help in generating TM code.

2. Deliverables and Submission

Your homework will be submitted as an *uncompressed* .tar file that contains no subdirectories. Your .tar file must contain all files and code necessary to build and test your compiler. If you use ourgetopt, *be sure to include the ourgetopt files in your .tar archive***.**

The .tar file is submitted to the class submission page. You can submit as many times as you like. **The last file you submit before the deadline will be the only one graded.** No late submissions are accepted for this assignment. For all submissions you will receive email at your uidaho address showing how your file performed on the pre-grade tests. The grading program will use more extensive tests, so thoroughly test your program with inputs of your own.

Your program must run with no runtime errors such as segmentation violations (SIGSEGVs). Any attempts to obfuscate the results of an unsuccessful compilation will be met with the harshest possible remedy.