

In this assignment you will extend the MiniJava language and compiler to enable the float data type. The report is a crucial part of this project and all other projects in this course. You should proof read your report and rewrite it a couple of times. **ANY spelling errors will result in 50% off the grade for this project.** If the same grammar errors appear in more than two reports, the resulting penalty will increase. This assignment should be done in groups of two. Please speak to me at least a week before the assignment is due if you cannot find a partner.

1 Motivation

The MiniJava compiler will be used throughout this course for hands-on projects such as implementing a garbage collector and implementing data-flow based code optimizations. By extending the MiniJava language and compiler to include the float data type, you will review all of the phases of a typical compiler and become comfortable with the MiniJava compiler.

2 The Assignment

Your assignment is to extend the MiniJava language (see <http://www.cs.tufts.edu/~sguyer/classes/comp181-2006/minijava.html>) and compiler to include the float data type. FloatTest1.java is a simple example of a MiniJava program that uses a float data type.

```
/*
  FloatTest1.java

  A program that prints a constant float.

MMS Aug 07
*/

class FloatTest1 {
    public static void main(String[] a){
        System.out.println(1.1);
    }
}
```

Your group will need to create MiniJava programs that use floats and compare the output of those test programs when compiled with `javac` and run with `java` versus when compiled with the MiniJava compiler and run with `spim`. For example,

```

% java -jar mjc.jar FloatTest1.java
% spim FloatTest1.java.s > spim.output
% javac FloatTest1.java
% java FloatTest1 > sun.output
% diff sun.output spim.output
0a1,5
> SPIM Version 7.2.1 of August 28, 2005
> Copyright 1990-2004 by James R. Larus (larus@cs.wisc.edu).
> All Rights Reserved.
> See the file README for a full copyright notice.
> Loaded: /usr/local/share/xspim/CPU/exceptions.s

```

The MiniJava-float compiler should be able to handle the following:

- float constants of the form digit* dot digit+
- declarations of a float scalar
- usage of a float in an expression
- System.out.println() should accept an int or a float.

For type rules, assume that floats can only be in arithmetic and comparison operations with other floats. Feel free to share such test cases with other groups in the class, but each group must submit their own test cases.

3 Getting Started

The course mailing list will be sent a starting MiniJavaCompiler, a new version of Mips/SpillAllRegAlloc.java, and a file called float.s, which contains example float MIPS instructions. For further MIPS information, you should use the Appendix A from Hennessy and Patterson, for which a link can be found at <http://pages.cs.wisc.edu/~larus/spim.html>.

1. Untar the the MiniJavaCompiler-start.tar file. It will expand into MiniJavaCompiler-float/.
2. Run the CodeGenAssem main on a test input.

```

> cd src
> javac CodeGenAssem.java
> java CodeGenAssem ../TestCases/And.java

```

3. Look at the output from various phases of the compiler. The output is in the ../TestCases/ subdirectory and files And.java.ast.dot, And.java.ST.dot, And.java.IRT, And.java.IRT.dot, And.java.MIPSnoreg, and And.java.s. Use the graphviz dot program to create a graphic from the dot files (see Project 0).

4. To add more tokens and modify the concrete syntax and abstract syntax grammars, edit the `ast.scc` file.
5. Run SableCC to regenerate the lexer, parser, and AST data structures. (see Project 0)

```
> sablecc ast.scc
```

6. Recompile `CodeGenAssem.java`.

Step through the compiler by starting in `CodeGenAssem.java` and using the Eclipse debugger. You are responsible for understanding how the provided MiniJava compiler works and you will be using it in all the projects in this course. Learning someone else's code is not easy, but it is a task that you will find yourself doing quite often in industry and/or academia. I am open to any ideas you might have about modifying the compiler. You can modify your own copy as much as you want as long as the intermediate outputs remain the same.

Modifications needed to lex, parse, and generate the AST:

- Modify `ast.scc` to include new tokens like float literals and the float keyword.
- Modify `ast.scc` to include grammar rules to parse declarations of float variables and parameters.
- Modify `ast.scc` to add new AST nodes for float literals and the float type.
- Modify the rules in `ast.scc` that convert the concrete syntax into abstract syntax to handle the new grammar rules and new AST nodes.

Building the symbol table and performing semantic type analysis:

- `ast_visitors/BuildSymTable.java`, have `getType()` handle the float type.
- `ast_visitors/CheckTypes.java`, floats should only be in arithmetic and comparison operations with other floats and/or as an input to `println()`.
- `SymTable/Type.java` needs a `FLOAT` type.

Translating the AST to the IR Tree representation:

- You will need to generate new classes called `Tree/ExpFLOATCONST`, `Tree/ExpFLOATMEM`, and `Temp/FloatTemp` (recommend that `FloatTemp` extends `Temp`).
- `Tree/ExpCALL` needs a flag to indicate if the call returns a float or not.
- `Tree/TreeVisitor.java` needs `visit(ExpFLOATCONST e)` and `visit(ExpFLOATMEM)`.
- `Tree/Print.java` and `Tree/PrintDot.java` need to implement `visit(ExpFLOATCONST e)` and `visit(ExpFLOATMEM)`.

- Temp/Temp.java, make num protected instead of private.
- ast_visitors/Translate.java
 - should generate an ExpFLOATCONST when it visits a float expression in the AST and an ExpFLOATMEM when it visits float id expressions.
 - outAPrintStatement should be modified to differentiate between int and float arguments (HINT: map relevant expression nodes in the AST to their Type in out methods).
 - outACallExp should pass a flag into ExpCALL to indicate whether the call returns a float or not.
 - outAMethodDecl should put the return value into the floating point return address if the method returns a float.

Generating MIPS assembly from the IR Tree representation

- Mips/CodeGen.java
 - Add munchExpFLOATCONST and munchExpFLOATMEM.
 - Generate floating point MIPS operations for arithmetic and comparison operators involving floats.
 - Modify munchExpCALL to return the float return register if the call returns a float.
 - Modify munchStmMOVE methods to handle ExpFLOATMEM targets and float expressions as sources.
 - Modify munchStmCJUMP to handle less than comparisons between floats.
- Frame/Frame.java, include a floatRegSet() method and a FRV() method.
- CodeGenAssem.java, call the spillAll that is capable of spilling floating point registers. Put the floating point return register (FRV()) in the noSpillSet.
- Mips/MipsFrame.java
 - Add "static final Temp"s for each of the floating point registers (\$f0 thru \$f31).
 - Put the corresponding string for each floating point register in the tempMap.
 - Create a method FRV() that returns the Temp that should hold the floating point return value (\$f0).
 - Create a floatRegSet() method that returns a set of at least three FloatTemps, but not \$f0.
 - Add printfloat MIPS code to programTail()
 - SymTable/MemberSTE.java and Mips/InFrame.java, edit exp method so that it generates an ExpFLOATMEM if the id is a float variable.

I recommend a test-driven approach. Write a test for each feature the extended compiler must handle and then make each test work. Also, make sure you understand how integers and integer operations are implemented in the current MiniJava compiler. Floats will be implemented very much like integers.

4 Your Report

The report is an essential part of your completed assignment. Use it to describe the phases of the MiniJava compilers and how you implemented the float data type. Organize and present your document as if it were the only basis for your assignment's grade. The format of your writeup is up to you, but it should minimally include the following:

- Introduce the main goals of the project and in a couple of sentences summarize what you have accomplished.
- Briefly describe how you augmented each phase of the MiniJava compiler. Indicate any new tokens and grammar rules added to the MiniJava language. What new semantic type checking rules were needed? What new MIPS instructions are generated?
- How did you test your MiniJava extensions?
- What problems did you encounter while implementing the float data type. If you knew someone who was just about to start work on this assignment, what advice would you give them?
- What, if any, outside sources did you use (e.g., articles, books, other students)? This is particularly important. It's OK to look at books and articles and speak with your professor and fellow students (although sharing code and working with another group is strictly forbidden), but as with any scientific document, you should always cite your references and collaborations. You can either cite collaborations in footnotes or in a separate Acknowledgment section.

There's no exact number of pages you should write, but if you've got between two and three then you're in the right ballpark.

5 Hints for doing the assignment

You should start this assignment right now!! You only have 11 days to complete this assignment. I recommend spending a couple hours a day on the project starting now. Start writing your report as you are planning the implementation. A well-written report with some missing implementation guarantees a much higher grade than a poorly written report with all the implementation.

I can look at your code and help point you in the right direction, but the amount of help I can give may be inversely proportional to the amount of time until the due date. By no means should you spend several hours trying to figure out a weird bug; consult me for help. When e-mailing me about the project, send a copy of the relevant section of your code (not as an attachment; send it as text pasted into your message). Give a good description of the problem including information about the stack in a debugger. Also, indicate possible solutions you are considering.

6 What to turn in

Turn in a hard copy of the report and email a copy in pdf format to mstrout@cs.colostate.edu.

Create a jar file that includes all of the bytecode files and the source files. Submit a README file, the jar file, and two test cases via email to mstrout@cs.colostate.edu. The README should include specific command-lines for running the jar file on your provided test cases.

7 Due date

This assignment is due Friday August 31st, **at 2:10pm** and is worth 10% of your final grade. Late assignments will be penalized 10% per day.