

Code Generator

For this milestone you will extend your program from the previous milestone to generate code for MiniJava programs.

Your compiler will take the name of a text file on disk as an argument. Your compiler will parse the file according to the syntax given in the Term Project Description. Your program will generate code for some target machine that will execute the input program. You have some latitude in choosing the target machine, subject to the following restrictions:

- You must be able to “print” integer values on the target machine in some fashion. This is necessary so that you can implement the semantics of MiniJava’s `System.out.println` statement.
- You must be able to demonstrate your compiler and the execution of its output for me in my office or in a CSSE lab. (If you need to run a remote shell to some other machine, that’s fine.)

As noted below, I want to discuss your choice of target machine before you do much code generation work.

Your compiler should implement the equivalent Java semantics for a MiniJava program. That is, for any legal MiniJava program, executing the code generated by your compiler should yield the same results as executing the code generated by `javac` for the same input file.

You are *not* required to implement garbage collection for your generated code, though you certainly may choose to do so. Note that targeting a virtual machine such as Sun’s JVM or Microsoft’s CLR may give you garbage collection “for free”.

You may use an existing assembler to convert the output of your compiler into target machine code.

DELIVERABLES

Target Machine Agreement

By class time on Monday of 9th week you must reach agreement with me on the architecture or virtual machine that your compiler will target.

Sample Input

You must submit 4 sample input files, **by midnight on Friday of 9th week**.

- All sample inputs must be valid MiniJava programs that produce some output. Because garbage collection is not required, your sample programs should require reasonably sized heaps and stacks.
- One of the samples must exclude both object instantiation and method calls. That is, it must include just a main class and a main method and use just `boolean` and `int` datatypes.

- One of the sample input files must include both object instantiation and method calls.
- The remaining two sample programs are left to your discretion.

Your sample input and output files should be wrapped in a zip or rar archive and uploaded to the drop box on Angel. Alternatively, you may commit the sample input files to your Subversion repository and let me know where to look for them. Please limit your sample inputs to no more than 250 lines and no more than 80 characters per line. Note that you do not have to submit sample output at this time.

I will select two sample inputs from each team and give the set of these to every team at least two days before the milestone deadline. I will give the remaining sample inputs to every team at class time on the milestone deadline.

Sample Output and Code

By midnight on the milestone deadline you must upload to the drop box on Angel a zip or rar archive containing:

- the output of your program for each sample input, one output file per input file, with the same name as the input file but the extension changed to “.out”
- the source code for your program

Alternatively, you may commit the sample output and source code to your Subversion repository and let me know where to look for it.

Grading

Your grade will be based on whether you submit the required sample inputs and the percentage of the sample input files that your program processes correctly.

I'll award bonus points for sample inputs that expose bugs in other teams' programs.