

Project 6: Instruction Selection

CSC 4351, Spring 2008

Due: 2 May 2008

Implement instruction selection for the MIPS architecture using Maximal Munch as described on pp. 197–200. You only need to modify class `Mips.Codegen`.

Environment and Support Files

For working on this project, change the environment variable `PROG` in your `.bashrc` file to `chap9`. There are two sets of support files. The code for canonicalizing the intermediate code trees is in `#{TIGER}/chap8`. The support files for instruction selection are in `#{TIGER}/chap9`. Copy the files for `chap8` first, as some of the files will be overwritten by the files for `chap9`.

There are some updates to the files in the `Frame` and `Mips` packages. The `Canon` package contains all the code for canonicalizing the intermediate code. Package `Assem` contains the abstract assembly instructions.

I made copies of all the class files and the modified source files available for Windowse users as for the previous lab to resolve the name clash between `Tree.Exp` and `Tree.EXP`.

Compilation

Since we don't use any non-Java source files anymore, it's easiest to use `'javac -g *.java'` manually for compiling.

For running the compiler, you will now use:

```
java Main.Main test.tig
```

This will produce a file `test.s` containing the various forms of intermediate code: before canonicalization, after canonicalization, after locating basic blocks, after scheduling these blocks, and finally the instructions selected.

For allocating registers and generating runnable assembly code, set the environment variable `PROG` to `chap11`. The generated file `test.s` will then only contain the finished assembly code (provided you don't use your own copy of `Main.Main`).

You can run your code on the MIPS simulator `SPIM`. Make sure you have the file `exceptions.s` in your working directory, start `xspim` (or `spim`), load the generated assembly file, and run it. `SPIM` is installed in `/soft/spim-7.1/` on `byte`. `SPIM` is also available for PCs from the University of Wisconsin.

Submission

Since there are only two files to modify, you can either submit them individually, or you could submit the entire working directory structure:

```
cd prog6; rm */*.class; ~cs435100/bin/r_copy 6
```

In the README file, provide any information that will help the grader to give you partial credit. Explain what's implemented and what's not. Explain important design decisions in your code.