

Project 3

CMSC 22620/32620, Spring 2007

Assigned: April 26, 2007

Due: May 3, 2007

1 Introduction

You are to write the *instruction selection* phase of the MiniML compiler. As discussed in class, you will use the Maximum Munch algorithm, taking advantage of ML pattern matching.

Instruction selection is explained in detail in Appel's textbook. Of course, you have to adapt it to the subset of the PowerPC instruction set that we are using, to our flavor of the Tree language (module `TraceTree`), and to our way of representing assembly code instructions.

The tarball for the project contains a template for the instruction selection phase in file `cg.sml`. You have worked with module `TraceTree` in project 2, so you should be very familiar with that.

The types for assembly instructions are in file `asm.sml`. This module is very similar to the one described in Appel's book, but there are some important differences:

- Unlike the compiler in the book, our compiler generates the entire code for a function (-cluster) during instruction selection—including prologue(s) and epilogue(s). (Our function clusters can have multiple entry points and multiple exits, including exits in form of tail-calls.)
- Since the stack frame size is not known until after register allocation, a symbolic constant is used in its place.
- Since we don't know which registers need to be saved and restored until register allocation is done, placeholder instructions are used to represent these (sequences of) operations.
- Jump information is divided into three cases: no jump, return, and jump to one of the labels in a list.

A list of PowerPC instructions has been posted to the course web page. There are also links to more extensive documentation. In case of doubt, it is useful to experiment with the C compiler on a MacOSX computer.

Your task is to fill out the template in file `cg.sml`. Numerous comments within that file should provide you with guidance. Also, be sure to read the comments in

asm.sml. File frame.sml contains information about physical registers and stack frames.

2 Instructions

2.1 Files

Download the file project3.tgz from the course web page. This is a compressed tarball containing the source code for the relevant portion of the MiniML compiler. File cg.sml is the only file you should need to edit.

2.2 Testing

During development, you can compile the code using

```
CM.make "proj3.cm";
```

When you think your code is ready to be tested, write a driver routine that takes a trace-tree and produces the corresponding instruction sequence. You can hand-craft small examples and inspect the output. You can take advantage of the `Asm.format` and pretty-print your generated output. (Use `Frame.showTemp` `Frame.precoloring` for the `saytemp` argument to `Asm.format`.)

3 Handing it in

You should only have to make changes to file cg.sml. To hand in your solution, you only need to send that file as an e-mail attachment to the instructor using the following e-mail addresses:

instructor	blume (at) tti (hyphen) c (dot) org
------------	-------------------------------------