15-411: Compiler Design — Fall 2017

Recitation 6: Miscellaneous — 13 October

About Code Review Week

First: if you haven't already, you should sign up for a code review timeslot using the link on Piazza.

After Lab 3 is due, you'll have one extra week before the Lab 4 handout is released. **You should use this time wisely** to improve your codebase and make up for technical debt. If you used any terrible hacks to get register allocation or calling conventions to work, fix them. If you think any part of your code is a jumbled mess, refactor it. You'll want to have a solid base upon which to implement Lab 4.

After you do this, push your code to the `code_review` branch. We will read your code and ask you questions about it during the code reviews. However, our focus during the code reviews **will not** be on your code itself! We don't really care about your code style, for example. We want to assess how well you **understand** the code you and your partner have written. We will be using `git blame` for this.

If there's any significant section of your compiler that your partner implemented and you did not read, you should read it. And if you don't understand how part of your compiler works, you should ask your partner to explain it to you. That said, we don't expect you to remember every detail of your implementation – we just want to make sure that both team members are participating roughly equally.

Announcements

To recap, here are some important dates and deadlines in the coming weeks:

- Lab 3 is due on Tuesday (10/17).
- There is no recitation next Friday (10/20) due to mid-semester break.
- Push your code to the `code_review` branch by the following Tuesday (10/24).
- There is no recitation the following Friday (10/27) due to code reviews.

Register Allocation in L3

As we mentioned last week, your register allocator for L3 will need to distinguish between caller- and callee-saved registers. To recap what was said in lecture, here are a few tips:

- When pre-coloring temps, you should ensure that the live ranges of the pre-colored temps are as short as possible. This is specifically relevant to the arguments and results of function calls – the best strategy is to have the arguments remain in temps until immediately before the function call, when they should be moved into argument registers.
- In order to account for caller-saved registers' values changing across function calls, you can just add the following rule to your liveness analysis:

$$\frac{l : \texttt{call}\ f \quad \text{caller-save}(r)}{\text{def}(\text{l}, \text{r})}\ J_8'$$

- When assigning registers to temps, choose caller-saved registers first. If you need to use callee-saved registers, be sure to save and restore them at the beginning and end of the function.