15-411 Compiler Design, Fall 2020
Lab 5 Checkpoint 0

15-411/611 staff

Due: 11:59pm, Tuesday, November 17, 2020
Estimated time to complete: ∼1-2 hours

1 Introduction

In this assignment, you will submit to gradescope a hand-annotated copy of your lab 4 compiler's assembly output on the `ckpt0.l4` benchmark. `ckpt0.l4` is a very simple benchmark designed to give you (and the course staff) an idea of the baseline code quality emitted by your compiler for an unexciting, no-nonsense code snippet with relatively few opportunities for advanced optimizations. We hope that this will give you a chance to prioritize which optimizations you start with for lab 5. This is important because fancy optimizations like PRE, software pipelining, or SCCP (which require a lot of work) may not yield much of a performance improvement if your emitted assembly is mostly bottlenecked by some basic inefficiencies. So, before you go deep down the rabbithole of implementing the fancy stuff, we'd like you to take stock of your backend for some simple code and tackle the low-hanging fruit, so that your effort on high-powered optimizations is worth the time.

2 Deliverable

You will submit to gradescope:

1. A copy of your lab 4 compiler's assembly output on the `ckpt0.l4` benchmark. The assembly should be thoroughly highlighted, circled, annotated, footnoted, emoji'd, etc. in the margins so that we get an idea of how you feel about each part of your compilers output. You can annotate your code for submission using any method you like (handwritten annotations are fine) so long as your ideas make it on the page.
2. Some subjective comments (a sentence or two) on how you could improve your compiler in each of the following categories:
   (a) Register Allocation and Coalescing
   (b) Spilling and Caller/Callee Save Registers
   (c) Instruction Selection
   (d) Control Flow, Branching, and Code Layout

Depending on the quality of your compiler, the given test program may not fully exercise your compiler in these categories. If you are aware of any other improvements that can be made to your compiler in these categories, you should make note them so that we can help you and give advice on how to prioritize for lab 5.

3 Grading Criteria

The goal of this assignment is mostly to prevent you from an having an existential crisis caused by spending several days implementing PRE only to find it yielded no performance improvement due to basic bottlenecks in your compiler. Our other goal is to provide you with personalized feedback on what to tackle first in lab 5. Therefore, we will mostly grade this assignment for completeness. So long as most of your interesting assembly is annotated and you put some thought into your category comments, you will get full points. **You do not need to write any code or improve your Lab 4 Compiler in any way for this project**. That being said, if you already started improving your compiler, feel free to show us your latest and greatest assembly output :)

4 The benchmark: `ckpt0.14`

For reference, here is the benchmark code. It is also distributed in `dist`.

```
//test return 1

int gcd1(int a, int b) {
    while (a != b) {
        if (a > b) {
            a -= b;
        } else {
            b -= a;
        }
    }
    return a;
}

int gcd2(int a, int b) {
    if (b == 0) {
        return a;
    } else {
        return gcd2(b, a % b);
    }
}

int main() {
    int N = 104;
    int M = 65;

    return gcd1(N, M) == gcd2(N, M) ? 1 : 0;
}
```