Project 2 Update and Clarifications

Hi students,

Due to that project 2 allows a lot of flexibility and the large volume of inquiries, we are creating the following FAQs for project 2.

Overarching goal

The goal of the project is to help you get some experience with the back end, and get some idea of instruction selection and different register allocation strategies. But on the other hand, we don't want the project to be too burdensome to you too. So the scope of register allocation is limited to local register allocation (per block), and you only need to perform a relatively simple greedy approach.

Should instruction selection come first before register allocation?

Yes, after instruction selection, you will have an IR that uses machine instructions with unbounded number of virtual registers. The goal of register allocation is to map the virtual registers to either physical register or (stack) memory. The naive approach can be viewed as such a mapping that all virtual registers are mapped to memory.

Should instruction selection assign a cost to each type of instruction?

You can just assume all instructions have the same cost to simplify things. As long as it can be executed by the SPIM simulator you are good.

How should the greedy approach be implemented?

We apologize for the confusion in the spec file (now updated). To make things clearer, we identified the following 3 reasonable routes to implement the greedy approach.

1. Build intra-block interference graph. Also compute the live in and out set for each block (global liveness info). With the inter-block live in and out set info, within each block, you will know what values will dead after what point in the block (it will not be used in later blocks), that makes it possible to reuse a physical register. In this approach, at the end of the block, you will only save the registers that correspond to the values still alive, i.e. no need to save a dead value back to memory.

2. Build intra-block interference graph, but not compute the live in and out set for each block. In this approach, you will assume that all variables are alive at the end of a block (because you don't have global liveness info). However, you can still reuse a physical register by storing a temporarily dead value (in the current block) to memory immediately before the physical register is reused. This will save the old register value, which may or may not be used in later blocks.

3. Not compute either global liveness info or local interference graph, just assume all variables are always alive, and assign physical registers one by one (sorted by use frequency count) without graph coloring.

Note that if you correctly implement route 1 or 2, and achieve reasonable memory loads reduction (30~60%) over the naive register allocation, you will receive full credits for the greedy register allocation part. If you implement route 3, you will still get 60% partial credits for the greedy part. You can also implement some other route that you believe is correct, just make sure to explain your approach and reasoning in the report. So however you implement it, please be suer to explain in your report.

Here are some general grading criteria:

- The naive / greedy register allocation both produce correct program output for test cases and input files.
- Correct implementation of the naive and the greedy approach. See above for more details.
- Both naive / greedy register allocation strategies produce a reasonable amount of memory loads. The # of memory loads doesn't have to be an exact number (since it depends on various factors), but the greedy allocation should show less number of memory reads compared to the naive allocation scheme. Depending on the input program, but if you are reducing like 30~60% of memory loads over the naive approach, you should be in good shape.
- Good document of your implementation design, considerations, observations etc.

#pin

project2

Edit good note | 0 Updated 3 years ago by Tong Zhou

followup discussions *for lingering questions and comments*

Start a new followup discussion

Compose a new followup discussion