Sample final questions and study suggestions

Please make sure you study the sample midterm questions and suggestions as well.

Suppose that you have defined a minimum set of instruction selection rules that are sufficient to tile any IR tree targeting a processor like the x86_64. Included in that set would be rules like this:

```
em.add(new MunchRule<IRExp, Temp>( CONST(_i_) ) {
        @Override
        protected Temp trigger(Muncher m, Matched c) {
                Temp t = new Temp();
                m.emit( A_MOV(t, c.get(_i_)) );
                return t;
        }
});
em.add(new MunchRule<IRExp, Temp>( PLUS(_l_, _r_) ) {
        @Override
        protected Temp trigger(Muncher m, Matched c) {
                Temp sum = new Temp();
                m.emit(A_MOV(sum, m.munch(c.get(_r_))));
                m.emit(A_ADD(sum, m.munch(c.get(_l_))));
                return sum;
        }
});
```

Now suppose that you add a new rule:

```
em.add(new MunchRule<IRExp, Temp>(PLUS(_e_, CONST(_i_))) {
        @Override
        protected Temp trigger(Muncher m, Matched c) {
                Temp sum = new Temp();
                m.emit( A_MOV(sum, m.munch(c.get(_e_))) );
                m.emit( A_ADD(sum, c.get(_i_)) );
                return sum;
        }
});
```

Explain how this rule might be considered an optimization. Explain what sort of IR trees would generate better code in this situation? Describe how the generated code would differ.

Encouraged by your success with this rule, you try again and add this rule:

```
em.add(new MunchRule<IRExp, Temp>(PLUS(TEMP(_t_), CONST(_i_))) {
        @Override
        protected Temp trigger(Muncher m, Matched c) {
                Temp res = c.get(_t_);
                m.emit( A_ADD(res, c.get(_i_)) );
                return res;
        }
```

```
    });
```
Unfortunately, you now see that your code generator is failing its correctness tests. Explain why this rule is incorrect.

Give an understandable informal English definition for the liveness of a temporary. Write down the data flow equations that define liveness of temps in terms of the sets use[n] (the temps read by an instruction n) and def[n] (the temps defined by the instruction n). Explain how the solution to these dataflow equations can be computed. Why is the solution that we want called the "least fixedpoint" of the equations?

Define an interference graph (What are the nodes? What does an edge in the graph represent?) Explain how liveness information is used to construct an interference graph. Explain how an interference graph is used after it is constructed.

Define a control flow graph (What are the nodes? What does an edge in the graph represent?). Explain how a control flow graph is used. Describe an alternative definition of a control flow graph that could be used to make the algorithms on the graph (which are often polynomial in the number of nodes in the graph) more efficient without making them significantly more complicated.

Describe the process of register allocation. What data structure is used? What is the difference between a legal allocation and an illegal allocation? What happens if we assign registers to temps based on an illegal allocation? Does the resulting code run? Does it compute the desired computation?

What is simplify in the context of register allocation?

What is coalescing in the context of register allocation?

What is a spilled node in the context of register allocation?

What is garbage collection? What information about the execution of a program is necessary for a garbage collector to do its job? What restrictions on the source language are necessary to make garbage collection legal? What is a mark-and-sweep garbage collector? What is a copying garbage collector? What is generational collection? Why does it work to reduce the cost of garbage collection?

Describe how an object is represented in an object-oriented language with single inheritance and virtual methods. What is a method dispatch table? When is it used? How is the information in a dispatch table accessed? Is a dispatch table ever searched linearly? What is the relationship between the "shape" of a super class method dispatch table and the "shape" of the method dispatch tables of all of its sub-classes? How is "instanceof" implemented in a language like Java? How do interfaces in Java complicate the simple structure described above?

What is the relationship between dataflow analysis and optimization? Name and describe 3 optimizations (or as many as you can) and their corresponding data flow analyses? What is the difference between a “may” and a “must” analysis? What does “global intraprocedural” optimization mean?

Describe the following loop optimizations and explain how they can improve the performance of a program:

- Hoisting loop-invariant computations
- Induction variable elimination
- Strength reduction
- Array bounds checking
- Unrolling loops

For each of these optimizations, try to construct an example where the optimization actually makes the code have worse performance.

Why is just-in-time compilation different from compilation that occurs completely before the program is executed? What additional complexity does JIT compilation have as a result? What additional opportunities does JIT compilation have as a result?

We may not talk about these, so it would be unfair to include them in the final exam. However, they are still interesting things to know about and you can learn about them in chapters 2 and 3

In the context of generating scanners, describe:

- Regular expressions
- Non-deterministic finite automata
- Deterministic finite automata

Give the sketch of a proof that any two of these formalisms are equivalent.

What does LL parsing mean? What does LR parsing mean? What is the intuition behind the claim that LR parsing is more powerful than LL parsing? Is an LL parser or an LR parser more efficient (in big-O terms)?