CS 6810 – Iloc Register Allocator
Due Date: *Friday, April 17, 2020 @ 5pm*

1 Required for an A

The project in this section is required in order to get an A in the course. Please see the following sections for what needs to be done for you to get either a BA or B.

Project Summary: Your task is to write a program that reads in optimized lloc, performs Chaitin-Briggs global register allocation and generates code for a restricted Iloc target.

The Environment: You will add a new pass to your SSA-based optimizer that will perform register allocation and generate code for the restricted iloc architecture. The restrictions for this architecture are on the number and function of the virtual register set. First, you will be generating code for machines with 12 and 16 *integer* registers (`%vr0` through `%vr15`). As in full iloc, there are four reserved virtual registers: `%vr0` is the frame pointer, `%vr1` is the stack pointer and `%vr2` and `%vr3` are used for function call linkage. Thus, you only have 8 and 12 registers, respectively, that you may use for global register allocation.

Report: You are to write a report on your optimizer consisting of the following:

1. a brief (1 paragraph) description of your global registration allocation implementation
2. a mention of any major problems that you encountered in the implementation
3. two tables summarizing the following on the set of benchmarks provided (1 for 12 registers, and 1 for 16 registers:
   - the running time of your allocator
   - the number of operations executed for the optimized code
   - the number of operations executed after register allocation.

   These tables should have the following format:

Benchmark	RA Time	Opt. # Instr.	RA # Instructions
...	...	...	...

What to Turn In: You should turn in your project to GitHub Classroom for the Register Allocation assignment. No matter the implementation language, create a `bash` script named `iloc` that invokes your optimizer/register allocator and emits the filan `iloc` to a file with the same prefix as the input file and the suffix `.ra.il`. Finally, include a PDF copy of your report in your submission.

The Intermediate Code: The iloc intermediate code is the same as the one provided in the *Engineering a Compiler* book with a few changes. The changes can be found in the documentation in `Iloc.pdf` on eLearning. You will have to deal with function calls and stores to memory in your optimizer. You may assume there will be no aliasing in the code provided.

The Prize: A prize for the optimizer and register allocator that produces the best code will be given. The team whose projects produce the overall best code will be given a free lunch at a local restaurant provided by the instructor.

2 Required for a BA

The register allocator is a large project. If you believe you will struggle finishing it, you have the option of implementing the following project.

Project Summary: Your task is to write a program that reads in optimized lloc, performs global common subexpression elimination by computing available expressions. This optimization *does not* use SSA. You will implement the algorithm using data flow analysis.

The Environment: You will add a new pass to your optimizer that will perform global common subexpression elimination. You will not be eligible for the prize nor an A should you choose to do this project.

Report: You are to write a report on your optimizer consisting of the following:

1. a brief (1 paragraph) description of your global value numbering implementation
2. a mention of any major problems that you encountered in the implementation
3. a table summarizing the following on the set of benchmarks provided.
   - the number of operations executed when doing local value numbering and SSA-based redundancy elimination.
   - the number of operations executed when doing local value numbering and global common subexpression elimination.

   These tables should have the following format:

Benchmark	LVN + SSA	LVN + GCSE
...	...	...

What to Turn In: You should turn in your project to GitHub Classroom for this assignment. No matter the implementation language, create a `bash` script named `iloc` that invokes your optimizerr and emits the final `iloc` to a file with the same prefix as the input file and the suffix `.gvn.il`. Finally, include a PDF copy of your report in your submission.

The Intermediate Code: The iloc intermediate code is the same as the one provided in the *Engineering a Compiler* book with a few changes. The changes can be found in the documentation in `Iloc.pdf` on eLearning. You will have to deal with function calls and stores to memory in your optimizer. You may assume there will be no aliasing in the code provided.

3 Required for a B

If you could not complete the previous project, your assignment is to finish that one before the end of the semester. To get a B, you must have a completely working optimizer.