**Lab 2 Report**

*Peisheng guo*

*Pg23*

**To Submit Your Code:**

Create a tar file with your source code, a makefile, and a README file that explains how to build your lab and how to run it. Send that tarfile, via email, to keith@rice.edu.

**To Submit Your Report:**

Answer the questions below. You may edit this document directly and submit via email — rename the file to your netid so that your filename does not clash with the filenames used by other students.

You may submit this document on paper by answering the questions and turning it in to my office, DH 2065. You can slide it under the door if I am not there.

Be sure to write down your name and your netid.

**Quantitative Results**

For each of the benchmark codes in ~comp506/students/lab2/TestCodes on the **CLEAR** systems, complete the following chart.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **Simulated Cycle Counts** | | | | |
| **Bench-mark** | **Input** | | **Original** | **Value Numbering** | **Loop Unrolling** | **Both** |
| bsort | bsort2.d | | 42,359 | 42359 | 42197 | 42197 |
| qsort | qsort2.d | | 15,505 | fail | fail | fail |
| fib | 20 | | 1,753 | 1,753 | 1487 | 1487 |
| mmult | 50 | | 8,425,269 | 8425269 | 8474469 | 8474469 |
| sumred | sumred1.d | | 4,272 | 4272 | 4272 | 4272 |
| algred | 50 | | 2,520,408 | 2270408 | 2315368 | 1998426 |
| oneloop | 50 | | 3,510,218 | 3260218 | 3510218 | 3260218 |

The code produced by your lab should produce the same answer as the original code— that is, both the original and the optimized code should print the same numbers in the output. (Of course, the simulator report on instructions executed and cycles executed should be different.)

**Experience**

1. Briefly discuss your experience building the optimizer. What things were easy to do? What things were hard?

I implemented two optimization methods: loop unrolling and local value numbering.

My optimizer is consist of input\_parse (parse the document), get\_leader\_and\_last (get leaders and corresponding lasts for each block), lvn (local value numbering), get\_largest\_reg\_and\_label (get the largest register and label has been used), lu (loop unrolling), update\_label\_to\_line\_num (update label\_to\_line\_num vector), output\_reduction (output the instructions) and a main function.

Easy part? None.

Hard thing would be to design how to store instruction in a good way. The design of data structure matters a lot in the program. A good way of storing sources, definitions, labels and opcode benefits the optimization process a lot.

**Experience** *(continued)*

1. Did you implement any extensions to value numbering?

I implemented a standard local value numbering.

**Experience** *(continued)*

1. How did you decide which loops to unroll.

Only accept loop in a single block.

**Experience** *(continued)*

1. What, if anything, would you do differently if you could start over on this project?

I will think carefully when I designed the data structure to store instructions. I have redesigned my data structure several times until finding the one that works.

**Experience** *(continued)*

1. If you could change the assignment, what would you change?

The assignment is comprehensive and taught us a lot about ILOC optimization methods. I cannot think of anything to improve.