

# **Project Proposal: CMPE202**

## **The Relationship between Reuse Buffer Size and Instruction Reuse**

Mark J. Boyd

### **Abstract**

Dynamic instruction reuse uses a buffer to retain candidate instruction sequences. As the size of the reuse buffer increases, the instruction reuse increases as well. certain limit. This paper describes this relationship based on simulations using various buffer sizes.

### **Importance of the Problem**

Available data from simulations implementing instruction reuse demonstrate significant increases in reuse as the size of the reuse buffer increases. This is shown clearly in the results from all three schemes in which A. Sodani and G. S. Sohi implement dynamic instruction reuse [1]. This data does not, however, show a clear limit to the increase in reused instructions. Since additional reuse may significantly increase execution time, finding an upper bound for reuse is critical for us to determine the possible contribution of this technique to improving execution time.

### **Planned Approach**

I plan to analyze program traces for local matching instruction sequences. By changing the distance of the search for matching sequences, I can determine the change in frequency of candidate instructions through a given length of the trace. This reuse buffer simulation will implement a simple FIFO replacement policy, and will show the expected reuse under optimal conditions as the size of the reuse cache increases.

### **Other Proposed Solutions**

Other proposed solutions implement replacement policies that remove sequences when registers are overwritten, or do not simulate larger cache sizes. Removing a sequence from the reuse buffer based on overwritten registers is inferior since a register may be rewritten to its original state or may have the same value written to it by sheer coincidence. The use of the upper limit of 1024 entries is also a poor approach, as this limit gives us little information about the theoretical useful upper limit for the cache size.

## **Metrics to Evaluate the Approach**

The simulation will track the percentage of instructions reused, and will also generate a histogram of the size of the matching sequences found each time a reuse is implemented. These statistics will be kept for each reuse buffer size.

### **Performance Metrics:**

The results of the simulation are measured using the following list of parameters calculated during the run.

- (1) **Total Instructions Executed:**
- (2) **Total Instructions Reused:**
- (3) **Histogram of Reused Sequences:**
- (4) **Percent of Reuse:**
- (5) **Size of Reuse Buffer:**

## **Evaluation Methodology**

The only tools required for the base analysis are output program traces from BACH or the traces directory, a PERL script to strip out unneeded statistics, and a string matching program. The base string matching program must implement a variable length 'reuse buffer', and be able to invalidate a sequence following a load instruction from a location that had a store since the load instruction entered the reuse buffer.

## **Planned Simulation Experiments**

With the proper trace formats, I'll run analysis of the traces in the string-matching program, doubling the size of the reuse cache each iteration (until the program takes more than a few hours to run). The data should show a clear relationship between cache size and potential reusable instructions.

## **Schedule**

Week of Quarter	Product or Result	Expected Hours
3	Proposal	8
4	Perl Scrpt	8
5	C++ Prog	16
7	Update	8
9	Final Rpt	16

Table 1: Planned timeline and breakdown of hours spent to run simulation.

## References

- [1] Sodani, Avinash and Sohi, Gurindar S. Dynamic instruction reuse. In *ISCA Proceedings on Computer Architecture*, pages 194–205, 1997. [soda97]