

Aaron Kobayashi

10/1/2002

Value Prediction and Instruction Reuse

Assumptions:

For the experimental results to be valid, we need to assume that the implementations for VP and IR are realistic and comparable. If we cannot make this assumption, then we cannot accurately evaluate the relative performance of these techniques. We also should assume that the respective memory table sizes are large enough to yield a good overall view of the performance of the technique. If these sizes are chosen to be too small, we will not be able to accurately portray the performance of the techniques in question.

Summary:

As we discovered in earlier studies [paper summary 2], the majority of a processor's resources are typically spent in approximately 10% of the code. This would suggest that there is a great deal of redundancy in applications. Further studies have shown that more than 75% of dynamic instructions produce a result that has already been calculated. This paper presents two techniques that attempt to exploit this apparent redundancy—value prediction and instruction reuse.

Value prediction (VP) predicts the results of instructions based on previous results and speculatively performs computation. The processor then confirms the validity of the speculative computations at a later point in time. Instruction Reuse (IR) takes a very different approach. Instead of speculatively performing computation, IR recognizes instruction traces that have been previously calculated and removes the trace from the instruction stream since the results have previously been calculated.

Since VP is speculative, the processor must still execute the instructions to verify the correctness of the predictions. If a misprediction is detected, the processor must re-execute any instructions that depend on the mispredicted value. This re-execution has a penalty; however, speculative execution can increase the CPI by eliminating the need to wait for values to be resolved. The repeated instructions increase the workload of a processor without increasing the amount of useful work done, lowering the overall efficiency.

IR is a nonspeculative technique. Consequently, reused instructions do not need to execute, saving processor resources for necessary calculations. Since IR is nonspeculative, it must either wait until all inputs have been resolved or simply not reuse instructions. Both waiting for resolved inputs and not reusing instructions decrease the amount of useful work the processor can accomplish. This limitation is offset by the fact that there is no reason to re-execute instructions for validity testing. Consequently, IR tends to reduce the workload of a processor while increasing the efficiency. IR also offers advantages when working with branch prediction. IR can reduce the misprediction penalty in two ways. When a mispredicted branch is reused, the misprediction can be detected at the decode stage well before it could have been detected if the branch had to execute. IR further reduces penalties by recovering useful work performed on the wrong path. This work can be saved and used at a later point. A final advantage of IR is that it reduces the execution latency of individual operations. This is accomplished by reducing multiple cycles to a single cycle.

For this research, experiments were conducted with the SimpleScalar toolset. This simulator models a 4-way dynamically scheduled processor and is highly configurable. Two models were created for the test—A VP model and an IR model. The

VP model was given four times the table entries (16K) to compensate for the fact that one IR entry takes four times the space as one VP entry, balancing the hardware requirements of the two schemes. VP Simulations were further decomposed to explore its effects on branch prediction. Branches were handled both speculatively and non-speculatively. As previously discussed, VP may cause instructions to re-execute several times. Consequently, simulations were also decomposed in two ways—multiple executions allowed and no multiple executions allowed—where we only re-execute instructions once after their correct operands are known.

These simulations were run on seven programs from the SPEC95 integer benchmarks suite. Each benchmark was simulated for 200 million cycles or until completion, whichever comes first.

Limitations:

The purpose of this paper was to examine the differences between VP and IR and evaluate the different microarchitectural interaction of these techniques. Even so a comparison of how the schemes scale as more memory space is added would be a welcomed addition. In all tests, the VP and IR schemes were limited to 16k and 4k respectively. To add more depth to the paper, an experiment that analyzes how performance is affected by an increase in available memory would be helpful.

Experimental Results:

The experimental results tended to fall in line with predictions made prior to quantification. Early validation (The scheme IR employees) was shown to cause a more than 100% increase in performance over late validation. This is largely due to the fact that early validation resolves branches early reducing the overall demand for execution resources. It was shown that over 30% of squashed work due to mispredictions could be recovered using the IR scheme greatly reducing the penalty IR pays for a branch misprediction. As predicted, in most cases IR reduced the amount of resource contention created by execution while VP increased contention in all cases.

With microprocessors the most important results are the net performance results. Overall the IR technique outperformed the VP technique. The main advantage VP has over IR is that its speculative prediction does not require all inputs to be present to make a prediction. Nevertheless, this advantage is overshadowed by the benefits of IR—no misprediction penalties, early validation, and the reduction of execution latency. Another interesting discovery was that less than 1% of instructions executed with VP had to be executed three or more times. This indicates that overall, the penalties associated with multiple re-executions due to value mispredictions are negligible.

References:

- A. Sodani, and G.S. Sohi, "Understanding the Differences Between Value Prediction and Instruction Reuse," 31st Annual International Symposium on Microarchitecture, December 1998.