Final Project Proposal

Dan Blandford and Takayuki Osogami
({dkb1, osogami}@cs.cmu.edu)

November 12, 2001

1 Project Web Page

http://www.cs.cmu.edu/~dkb1/740

2 Project Description

We propose to investigate the utility of the GUARD instruction for adding predication to an assembly language.

The GUARD instruction is a "meta-instruction" which does not modify the state of the CPU directly but instead may modify the subsequent instructions read by the system. GUARD takes two parameters: a register and a bitmask. Depending on the value of the register, GUARD may convert some of the subsequent instructions (determined by a bitmask) into NOPs. This allows us to convert a language which does not support predication into one which does, without the necessity of adding additional bits to each instruction to indicate a predicate. This, in turn, allows us to perform if-conversion on a language, replacing many of its branch instructions with GUARD operations. If the cost of a mispredicted branch is high relative to the cost of executing a GUARD operation and a few NOPs (as is the case with many deeply pipelined or out-of-order processors), this can improve performance significantly.

We propose to use the SimpleScalar simulator to implement and test the GUARD instruction. A significant portion of our project will consist of understanding the simulator to a sufficient degree that we can add our GUARD instruction to the set of instructions it supports. Once we have accomplished this, we will try to find the most efficient way to use the GUARD instruction to improve performance. Our primary performance metric will be the dynamic window size of a program - that is, the number of instructions the CPU executes between mispredicted branches.

3 Logistics

3.1 Plan of Attack and Schedule

We will devote the first week of our project to installing and understanding the SimpleScalar simulator. We will read the documentation in detail. SimpleScalar provides several processor models to choose from; we will need to select one of these for modification and implementation.

During the second and third week we will have modified the chosen simulator to implement the GUARD instruction. We will need to find an instruction which is unused in target language and modify the simulator to accept that instruction.

We will also need to find a way to add our instruction to programs that the simulator can run. We will probably try to do this using an ASM directive in the C source code, with possibly some postprocessing of the assembly code as well. We can accomplish this in parallel with the modification of the compiler; we will try to finish both of these tasks by the end of the third week.

We will allocate the fourth week for testing and debugging. As our milestone, we would like to have a simulator that implements the GUARD instruction by November 20th.

In the remaining two weeks we will develop test cases for our simulator and try to choose which branches to use if-conversion on so as to maximize performance. If possible we will use a professional suite of benchmarks for testing. We will also need to allocate some time to write the final paper and prepare our presentation.

We need to provide for dividing the work among group members. In the first few weeks the work will be fairly easy to divide up: Takayuki will implement the GUARD instruction on the simulator, and Dan will modify the compiler and/or source code to produce assembly code containing the GUARD instruction. After that, the work will be more difficult to divide up. Dan will probably handle running the simulator on the test cases, and we will collaborate on writing the paper and creating the proposal.

3.2 Literature Search

This project was inspired by the discussion of the GUARD instruction on page 6 of the paper "Demystifying EPIC and IA-64", by Peter Song. The first paper describing guarded execution is "Highly Concurrent Scalar Processing", by Peter Y. T. Hsu and Edward S. Davidson. A more current reference is "Guarded Execution and Branch Prediction in Dynamic ILP Processors", by Dionisios N. Pnevmatikos and Gurindar S. Sohi. We feel that these papers contain enough information to allow us to implement guarded execution.

Another paper which will be of great help to us is "The SimpleScalar Tool Set, Version 2.0", by Doug Berger and Todd Austin. This paper is the technical report for the simulator we will use.

3.3 Resources Needed

The most important resource we will need is the SimpleScalar processor simulator. We have acquired this simulator and are in the process of installing it. We will also need benchmark programs on which to test our simulator; we will need to further investigate this.

3.4 Getting Started

Thus far we have downloaded the SimpleScalar simulator distribution and are preparing to install it.