Chapter 8

A CHARACTERIZATION OF CONTROL INDEPENDENCE IN PROGRAMS

Anasua Bhowmik

Computer Sciences Department
University of Maryland, College Park
anasua@cs.umd.edu

Manoj Franklin

Electrical and Computer Engineering Department, and
University of Maryland Institute for Advanced Computer Studies (UMIACS)
University of Maryland, College Park
manoj@eng.umd.edu

Abstract Many studies have shown that significant amounts of parallelism can be extracted from ordinary programs if a processor can accurately look ahead arbitrarily far into the dynamic instruction stream. Control flow changes caused by conditional branches are a major impediment to determining which of the distant instructions belong to the dynamic instruction stream. This chapter deals with the use of control independence information for extracting this “distant parallelism”. The two major contributions of the chapter are: (i) demonstration of control independence as a viable means to extract distant parallelism, and (ii) detailed characterization of control independence, in terms of granularity and the available parallelism at each granularity level. The frequency data (on control independence granularity) presented is a first step in determining what situations require special attention. This knowledge can be applied to the development of specialized hardware architectures that can utilize control independences at different granularity levels.

Keywords: Branch prediction, control dependence, instruction-level parallelism (ILP), multiple threads, single flow of control, speculative execution.
1. INTRODUCTION

Many studies have shown that significant amounts of parallelism can be extracted from ordinary programs if a processor can accurately look ahead arbitrarily far into the dynamic instruction stream—by means of perfect branch prediction and an infinitely large scheduling buffer [8] [13]. Although both of these assumptions are somewhat idealistic, the high values of available parallelism do provide an impetus to explore techniques to extract it in future processors. A compelling question that requires attention is: what can be done to extract this parallelism at a distance? Current processors extract parallelism only within a narrow window of the dynamic instruction stream; they have no ability to identify distant instructions that belong to the dynamic instruction stream, let alone determine which ones among them are independent and can be executed earlier. The main reason for this is the presence of conditional branches, which cause control flow changes in the program.

The currently prevalent technique for dealing with the "conditional branch problem" is to predict branch outcomes, followed by speculative execution along the predicted path. Although highly accurate branch prediction schemes have been developed, the average number of instructions between adjacent mispredictions is still rather low—of the order of 100 or fewer. Therefore, control speculation by itself cannot be a means to get to the distant code earlier, because each branch misprediction causes the rest of the speculated path to be abandoned and a different path to be pursued, even when the new path has many instructions in common with the previously abandoned incorrect path. Identifying and exploiting control independences seem to be the natural solution to get to the distant code earlier\(^1\) [3] [4] [8] [12] [14] [16]. Effective use of control independence information helps to reach distant code, despite the presence of mispredicted branches in between.

The primary goal of this work is to present a characterization of control independences in ordinary programs. It presents many of the data that we collected when trying to understand and investigate techniques to exploit control independence. A good understanding of this subject is necessary before devising good mechanisms to exploit control independence. Unless key performance issues are understood, it is difficult to obtain performance benefits from new designs.

\(^1\)It is interesting to note that a time-honored method of dealing with data dependences is to exploit data independence by code re-scheduling, i.e., by executing the instructions in an order different from the original program order.