afs/cs.washington.edu/projects/dyncomp/tracker. The executable hello.issue tracks the dual-issue behavior of the infamous hello world program just to give a simple example of the output expected from the Issue Tracker.

Future Work

The current version of the Issue Tracking tool, is only half of the puzzle. Another version of the tool is needed to analyze the issue behavior of the program once the run-time compilation and stitching has been performed.

It has been proposed that an Issue Tracking tool for dynamically compiled code could be built using an instruction-level debugger, such as gdb, to monitor the running program and print the binary of each instruction executed. This list of instruction binaries, then, could be fed to the issue and disassembler code contained in the current version of the Issue Tracker and an output format identical to the current tool could be generated. The outputs from the two version of the Tracker tool could then be compared to determine the effects of the stitched code on the issue behavior of the executable.

References

1. *ATOM: User Manual.* Digital Equipment Corporation, Maynard, Mass. 1994

The analysis file, *issue.anal.c*, contains the dual issue check routine, *DualIssue()*, as well as the disassembly routine, *print_insn_alpha()*. The dual issue checking routine checks a pair of instructions to determine if they can be issued at the same time without interfering with each other's resources. An example of a dual issued pair would be a floating point arithmetic instruction and an integer arithmetic instruction, since each of these instructions executes on a different portion of the processor hardware. Conversely, it would not be possible to issue two load instructions together since both instructions require the same processor hardware, e.g., the memory bus. The code for the dual issue routine was borrowed from a sample tool provided with the ATOM development kit.

The disassembly routine translates a binary instruction into an Alpha instruction mnemonic to make reading the Issue Tracker output easier This code was adapted from a similar routine found in gdb for the Alpha.

The instrumentation routine, found in *issue.inst.c*, analyzes each basic block in each object file in the tracked program. It inserts a call to an instruction processing procedure, *ProcessInstruction(),* before each instruction in each basic block. Then, as the binary is executing, these procedure calls are executed and the analysis routines run for every executed instruction. On each call to *ProcessInstruction()*, the analysis routines gather the information for the instruction about to be executed. When the processing procedure has seen two instructions, it performs the dual-issue check and disassembles each instruction into an alpha pnumonic. These pnumonics are then passed to a print routine, *PrintIssue()*, which prints dual-issued instructions as pairs and single-issued instructions individually.

An example of the output file generate by the Issue Tracker is shown in below, where dual issued instructions are grouped by angled slashes.

```
ENTRING PROCEDURE: _strcmp
ENTRING BLOCK:  PC = 0x804380
/ 0x804380 lda  sp, 0x0(sp)
\ 0x804384 addl t2, #1, t9
/ 0x804388 addl t1, #1, t10
\ 0x80438c addl t2, #3, a3
/ 0x804390 addl t1, #3, a4
\ 0x804394 addl t2, #2, a5
/ 0x804398 addl t1, #2, t8
\ 0x80439c or   t1, zero, gp
/ 0x8043a0 or   t2, zero, t11
\ 0x8043a4 fnop
EXITING BLOCK:  PC = 0x804380
```

User's Guide

The two ATOM instrumentation and analysis files, as well as a makefile that demonstrates how they are linked in with the static program to be analyzed, can be found in /

There are two factors that determine if a pair of instructions can be executed on the same clock cycle:

1. the two instructions do not compete for the same resources, and
2. the two instructions are quad-word aligned.

The first factor, resource competition, arises when both instructions wish to perform the same type of modification to the state of the program in execution. An example would be two load instructions attempting to execute in parallel; they would each attempt to assert the address lines of the processor and the resulting bus conflict would prevent both instructions from executing.

The second factor, quad-word alignement, refers to the alignment of the instruction pair, meaning the alignment of the first instruction in the pair being considered. The Alpha dictates that for two instructions to be dual-issued, they must be aligned on a quad-word boundary, which simply means the address of the instruction pair must be evenly divisible by 8 or 16 decimal (8 or 0 hex).

The problem with dynamic compilation, then, is that since the dynamic portion of the compilation process stitches instructions into the instruction stream at run-time, the possibility exists that the stitched instructions will interfere with the alignment of the instructions scheduled during the static part of the compilation process.

The Issue Tracker, then, is a tool that tracks which instructions are dual issued in an effort to determine if the stitched code severely undermines the scheduling optimizations performed by the static compiler.

Implementation

The Issue Tracker is implemented as an ATOM tool[1]. ATOM is an instrumentation tool that allows the programmer to analyze various characteristics of an executing program. ATOM provides a set of functions that insert procedure calls into the program binary with different degrees of granularity, i.e., before and after procedures, basic blocks, and instructions. These inserted procedure calls, then, break the flow of the original program by jumping to the analysis routines written by the programmer to analyze and gather data on the program's exectution.

ATOM gives the programmer a way to monitor the original program as it is executing. ATOM provides a flexible code instrumentation interface that allows the programmer to build a wide variety of interesting and useful profiling- and simulation-type tools.

The Issue Tracker consists of two ATOM files, *issue.inst.c* and *issue.anal.c*, and one header file, *alpha_opc.h*. *inst* and *anal* are the standard Instrument and Analysis files used to direct ATOM in annotating the object files and performing the analysis desired. The header file contains the type definitions and tables used by the alpha disassembly procedure, *print_insn_alpha()* to convert an instruction binary into an alpha instruction string.

Issue Tracker

Scott van Woudenberg

This document describes an ATOM tool designed to track dual issued instructions in DEC Alpha executables.

Background

The Issue Tracker is a tool that was developed as part of an undergraduate research project advised by Professor Susan Eggers. The tool falls under the scope of the Dynamic Compilation research project led by Professor Eggers and Professor Craig Chambers. The Dynamic Compilation Project is investigating the performance gains of run-time compilation in which compiler optimizations are applied to programs at run-time, taking advantage of the extra information known during the execution of a program.

More information regarding the Dynamic Compilation Project can be found at *http://www.cs.washington.edu/* by following the *Research* link.

Impetus for Issue Tracker

The Issue Tracker tool is part of an investigation into the effects of run-time generated code on the scheduling behavior of a program in execution. Instruction scheduling plays an important role in achieving good performance on superscalar machine architectures like the DEC Alpha. A good compiler will rearrange instructions in the emitted instruction stream in an attempt to exploit the highest degree of parallelism offered by the superscalar architecture.

The dynamic compiler under development is based on the Multiflow compiler with a backend targeted for the Alpha 21064 processor which provides two execution pipelines. The original Multiflow compiler performs instruction scheduling optimizations in an effort to increase the number of instruction pairs that are executed simultaneously.