Project Proposal: Simulation and Analysis of Dynamic Instruction Scheduling Algorithms

Yankai Jiang
Dept. of Automation Engineering
Xi'an Jiaotong University
Xi'an, China
xjtujyk@gmail.com

I. Introduction

Data hazards in a program cause a processor to stall. With static scheduling, the compiler tries to reorder these instructions during compile time to reduce pipeline stalls. As for another choice, dynamic scheduling the hardware tries to rearrange the instructions during run-time to reduce pipeline stalls. Dynamic scheduling is an important technic utilizing in many modern processors. Three advantages listed:

- Enables handling some cases when dependencies are unknown at compile time;
- Simplifies the compiler;
- Allows code that was compiled with one pipeline in mind to run efficiently on a different pipeline.

Computer scientists develop novel algorithms for dynamic instruction scheduling to improve computer efficiency. The scoreboard algorithm and Tomasulo algorithm are two typical dynamic scheduling algorithms. The Tomasulo algorithm is much better than the scoreboard algorithm and is a stronger algorithm. Many modern processors that develop instruction-level parallelism use the Tomasulo algorithm or its variants. We will discuss these two algorithms in the next part.

II. Related Work

A. Scoreboarding Algorithm

Scoreboarding is a centralized method, first used in the CDC 6600 computer, for dynamically scheduling a pipeline so that the instructions can execute out of order when there are no conflicts and the hardware is available.

In a scoreboard, the data dependencies of every instruction are logged. Instructions are released only when the scoreboard determines that there are no conflicts with previously issued and incomplete instructions. If an instruction is stalled because it is unsafe to continue, the scoreboard monitors the flow of executing instructions until all dependencies have been resolved before the stalled instruction is issued.

Instructions are decoded in order and go through the following four stages:

- Issue: The system checks which registers will be read and written by this instruction. This information is remembered as it will be needed in the following stages. In order to avoid output dependencies (WAW – Write after Write) the instruction is stalled until instructions intending to write to the same register are completed. The instruction is also stalled when required functional units are currently busy.
- Read operands: After an instruction has been issued and correctly allocated to the required hardware module, the instruction waits until all operands become available. This procedure resolves read dependencies (RAW – Read after Write) because registers which are intended to be written by another instruction are not considered available until they are actually written.
- Execution: When all operands have been fetched, the functional unit starts its execution. After the result is ready, the scoreboard is notified.
- Write Result: In this stage the result is about to be written to its destination register. However, this operation is delayed until earlier instructions—which intend to read registers this instruction wants to write to—have completed their read operands stage. This way, so called data dependencies (WAR – Write after Read) can be addressed.

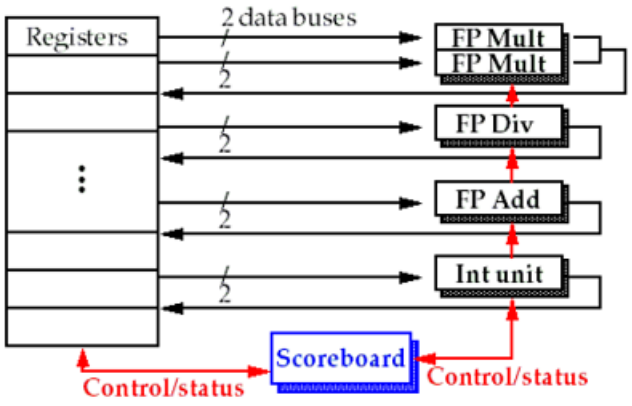

Fig. 1. the basic structure of a DLX machine with a scoreboard.

B. Tomasulo Algorithm

Tomasulo's algorithm is a computer architecture hardware algorithm for dynamic scheduling of instructions that allows

out-of-order execution and enables more efficient use of multiple execution units. It was developed by Robert Tomasulo at IBM in 1967 and was first implemented in the IBM System/360 Model 91's floating point unit.

The major innovations of Tomasulo's algorithm include register renaming in hardware, reservation stations for all execution units, and a common data bus (CDB) on which computed values broadcast to all reservation stations that may need them. These developments allow for improved parallel execution of instructions that would otherwise stall under the use of scoreboarding or other earlier algorithms.

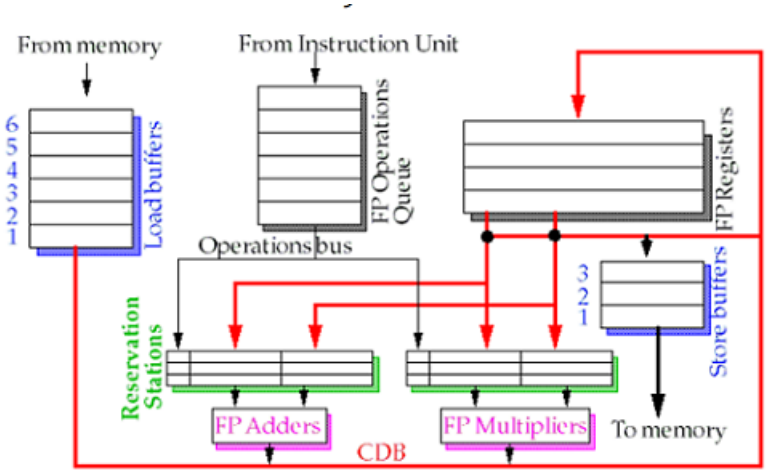

Fig. 2. The basic structure of a DLX FP unit using Tomasulo's algorithm.

III. Proposal Description

After studying from class about dynamic instruction scheduling, we have put our interest on simulating the algorithms. It is important to analyse these algorithms and their performances. In this project, The substance of my work can be presented by the following three steps:

- Research in detail on scoreboarding algorithm and Tomasulo algorithm, develop a solid understanding on the footstone approach
- Impelement these algorithms with GUI, get a better understanding.
- Evaluate and analyse the algorithms.

In order to do so, we have carried out a plan including two parts. First, two algorithms should be tested on PC. Second, a GUI platform will be built, in which simulations for dynamic instruction scheduling are displayed to demonstrate the precise mechanism of the pipeline and eventually show the effectiveness of the algorithms. After the implementation, comparing two algorithms will be evaluated and other relevant problems to be encountered will be discussed.