# 1 Abstract

This is a report of the project of MS108 homework implementing a simplified CPU supporting some of the RISC-V instruction sets. My solution is based on Tomasulo algorithm.

# 2 Feature

## 2.1 Tomasulo

The Tomasulo algorithm aims to execute Out-of-Order(abbreviated OoO), and solve problems triggered by OoO, such as WAR, WAW, and precise execution. The key is to use renaming, reservation station(abbreviated RS) and ROB.

In my design, each FU has its own RS, and the size of each RS is different. Renaming is based on where the instruction is in ROB.

## 2.2 Cache

I implemented a 512B I-cache in my project, then I made it a 2-way associative one, but there is little improvement. I guess this is because the testbench only contains short codes.

My 2-way associative cache is in /backup/Units.

## 2.3 Branch Policy

I first designed a one without branch prediction and its policy is stall. Since branch instructions hold about sixth of all instructions, it stalls frequently and is not so OoO. This one can run with a frequency of 100MHz.

Then I made one with branch prediction. Like what BOOM(Berkeley Out-of-Order Machine) does, it uses shadow registers and branch bits. Each instruction is given a branch mask to evidence if it follows in-flight branch instructions. If a branch instruction is executed, it boardcasts to update others' branch masks. Only Execute load/store instructions with cleared branch mask. This one can run with a frequency of 70MHz.

This sounds cool, but when I implemented it, I found the delay high, since to deal with the boardcast of branch instructions, a large amount of logic units are in need. (let alone the shadow registers use considerable sources too)

But it runs faster and is really COOOOOOOOL.

# 3 Summary

1. The main problem I face is that I do not understand how to make improvement in organization level. Once I noticed that I had repeated calculating one same thing in each RS, so I picked it out: calculate it outside and send the result to each RS.

The result was crazy: the number of LUT increased considerably.

2. The Tomasulo is efficient in handling a set of load/store instructions. Even the naive version (stall when branch) makes sense in such cases.

3. The biggest drawbck is unsupporting precise exceptions, as it does not handle branch misprediction in ROB. Other can-be improvements include superscalar and superpipeline.

*. At first, I wanted a CPU able to run an OS written by myself, later I just wanted to write something new instead of copying seniors' designs. Hope I achieve it. May it be.

# 4 Reference

1. https://github.com/riscv-boom/riscv-boom, an out of order RISC-V CPU using chisel, with branch bits(named branch mask there)

2. http://www.kroening.com/diplom/diplom/main003.html, details about the hardware of Tomasulo Architecture.