CS246 – Homework #2

Matheus S. Farias

School of Engineering and Applied Sciences, Harvard University

Spring, 2023

Abstract

This document presents my solutions for homework 2 of CS246 taught in the Spring of 2023.

Contents

§1 WAR Hazards and Scoreboarding

(a) The pipeline implemented in CDC6600 has Read Flag. Basically, these flags are barriers to further writing instructions. For WAR hazards, the writing instruction will only be performed when the read flag sets the registers to all clear.

(b) Let's say we have the two following instructions

- ADDI R1, R2, R3
- ADDI R2, R4, R5

For each instruction, we store the **index** of the registers we read in a reservation station, instead of the value of the register as Tomasulo does. In this example, we have two "columns" – the first one stores R2 and R3, and the second stores R4 and R5. Each of these registers will have a read flag the same way as in CDC6600. If the register is clear, we remove it from the column (that's how we come up with the read flag idea). At each cycle, we compare the index of the register we want to write to all the registers indices we stored in each column. If the register doesn't match with any other, it means either that register was never used, or it was read

Cycle 71

Instructions		Instruction Status Iter 1			Instruction Status Iter 2		
		Issue	Exec Comp	Write Result	Issue	Exec Comp	Write Result
LD	F0, 0(R1)	1	2-3	4	10	11-12	13
DIVD	F2, F0, F6	2	5-19	20	11	37-51	52
LD	F6, 8(R1)	3	4-5	6	12	13-14	15
DIVD	F6, F6, F2	4	21-35	36	13	53-67	68
SD	F6, 16(R1)	5	37-38	39	14	69-70	71
DADDI	R1, R1, #-32	6	7	8	15	16	17
BNEQZ	R1, LOOP	7	9	10	16	18	19

Figure 1. Instruction status at the last cycle (cycle 71)

before and thus is ready to be written (**WAR hazard detection!**). This solution is better than Tomasulo in terms of hardware resource utilization since it stores the index instead of the value. However, it needs a lot of comparisons for each cycle.

§2 Tomasulo without Speculation

You can see the solution cycle-per-cycle in the file `q2_answer.pdf`. Figure 1 presents the instruction status at the last cycle (cycle 71).

In this problem, we have a lot of race conditions so it doesn't make sense to add more FP dividers or FP adders (increase the probability of race conditions) or pipeline FP dividers (will have stalls due to race conditions). Then, the correct option is to **C) improve the performance of the FP divider**, this is the longest operation in our instruction set.