CS M151B Homework 4

Charles Zhang

January 27, 2022

Problem 4.10

When processor designers consider a possible improvement to the processor datapath, the decision usually depends on the cost/performance trade-off. In the following three problems, assume that we are beginning with the datapath from `Figure 4.21`, the latencies from `Exercise 4.7`, and the following costs:

I-Mem	RF	Mux	ALU	Adder	D-Mem.	Single Reg.	SE	SG	Ctrl
1000	200	10	100	30	2000	5	100	1	500

Suppose doubling the number of general purpose registers from 32 to 64 would reduce the number of `lw` and `sw` instructions executed by 12%, but increase the latency of the register file from 150 ps to 160 ps and double the cost from 200 to 400. (Use the instruction mix from `Exercise 4.8` and ignore the other effects on the ISA discussed in `Exercise 2.18`.)

a) What is the speedup achieved by adding this improvement?

$$\text{Speed Up} = \frac{\text{CT}_i}{\text{CT}_f}$$

$$\text{CT}_i = (0.52 \times \text{Latency(R/I-type)}) + (0.25 \times \text{Latency(lw)}) + (0.11 \times \text{Latency(sw)}) + (0.12 \times \text{Latency(beq)})$$

$$\text{Latency(R/I-type)} = \text{Latency(I-Mem + Reg. File + MUX + ALU + MUX)}$$

$$\text{Latency(R/I-type)} = 250\text{ps} + 150\text{ps} + 25\text{ps} + 200\text{ps} + 25\text{ps}$$

$$\text{Latency(R/I-type)} = 650\text{ps}$$

$$\text{Latency(lw)} = \text{Latency(I-Mem + MUX + Reg. File + MUX + ALU + D-Mem + MUX)}$$

$$\text{Latency(lw)} = 250\text{ps} + 25\text{ps} + 150\text{ps} + 25\text{ps} + 200\text{ps} + 250\text{ps} + 25\text{ps}$$

$$\text{Latency(lw)} = 925\text{ps}$$

$$\text{Latency(sw)} = \text{Latency(I-Mem + MUX + Reg. File + MUX + ALU + D-Mem)}$$
$$\text{Latency(sw)} = 250\text{ps} + 25\text{ps} + 150\text{ps} + 25\text{ps} + 200\text{ps} + 250\text{ps}$$
$$\text{Latency(sw)} = 900\text{ps}$$

$$\text{Latency(beq)} = \text{Latency(I-Mem + Reg. File + MUX + ALU + Gate + MUX)}$$
$$\text{Latency(beq)} = 250\text{ps} + 150\text{ps} + 25\text{ps} + 200\text{ps} + 5\text{ps} + 25\text{ps}$$
$$\text{Latency(beq)} = 655\text{ps}$$

$$\text{CT}_i = (0.52 \times 650\text{ps}) + (0.25 \times 925\text{ps}) + (0.11 \times 900\text{ps}) + (0.12 \times 655\text{ps}) = 746.85\text{ps}$$

$$\text{CT}_f = (\frac{52}{95.68} \times \text{Latency(R/I-type)}) + (\frac{22}{95.68} \times \text{Latency(lw)}) + (\frac{9.68}{95.68} \times \text{Latency(sw)}) +$$
$$(\frac{12}{95.68} \times \text{Latency(beq)}) + 10\text{ps}$$
$$\text{CT}_f = (353.26\text{ps} + 212.69\text{ps} + 91.05\text{ps} + 82.15\text{ps}) + 10\text{ps}$$
$$\text{CT}_f = 649.14\text{ps}$$

$$\frac{\text{CT}_i}{\text{CT}_f} = \frac{746.85\text{ps}}{649.14\text{ps}} = \boxed{1.15}$$

b) Compare the change in performance to the change in cost.

$$\Delta\text{Performance} = 649.14\text{ps} - 746.85\text{ps} = -97.71\text{ps}$$

$$\Delta\text{Cost} = 200$$

$$\frac{\Delta\text{Performance}}{\Delta\text{Cost}} = \frac{-97.71\text{ps}}{200} = \boxed{-0.49\text{ps/cost unit}}$$

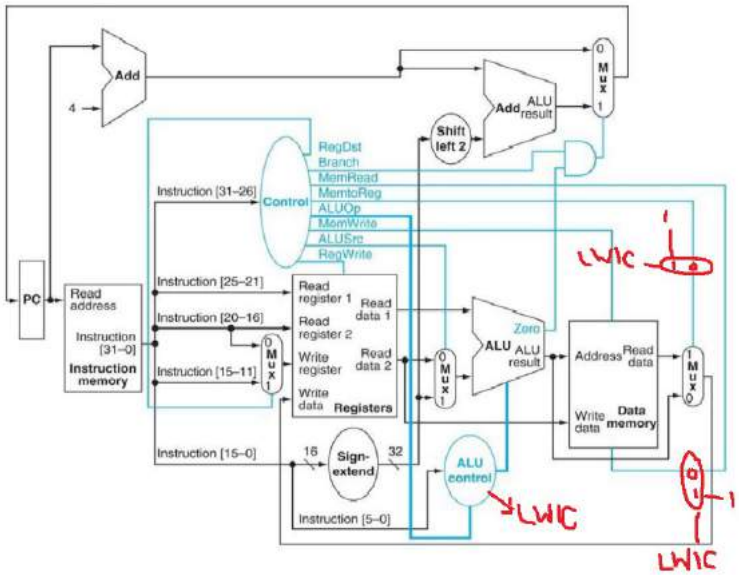
c) Given the cost/performance ratios you just calculated, describe a situation where it makes sense to add more registers and describe a situation where it doesn't make sense to add more registers.

Given this cost/performance ratio, it would make sense to add more registers if the performance of the datapath was greatly hindering the performance of the overall machine. The ratio is relatively low in terms of benefits, therefore it would only really be reasonable to make this tradeoff if the designer was trying to fully optimize the machine and had little cost concerns.

Problem 4.11

Examine the difficulty of adding a proposed `lwi.drd, rsl, rs2` ("Load With Increment") instruction to MIPS. Interpretation: `Reg[rd] = Mem[Reg[rs1] + Reg[rs2]]`.

The new datapath would look as follows:



Control would look as follows:

Signal Name	R-Format	`lw`	`sw`	`beq`	`lwi`
Op5	0	1	1	0	0
Op4	0	0	0	0	0
Op3	0	0	1	0	0
Op2	0	0	0	1	0
Op1	0	1	1	0	0
Op0	0	1	1	0	0
RegDst	1	0	X	X	1
ALUSrc	0	1	1	0	0
MemtoReg	0	1	X	X	0
RegWrite	1	1	0	0	1
MemRead	0	1	0	0	0
MemWrite	0	0	1	0	0
Branch	0	0	0	1	0
ALUOp1	1	0	0	0	1
ALUOp2	0	0	0	1	0

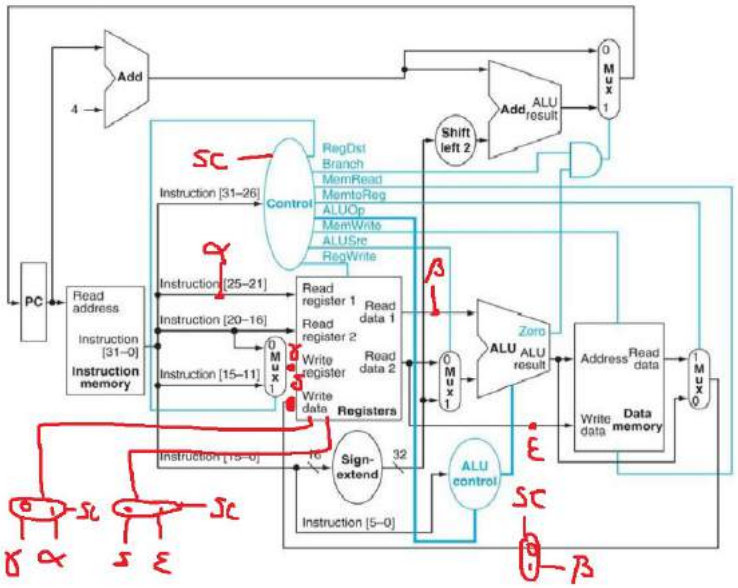
ALU control would look as follows:

Opcode	ALUOp	Operation	funct	ALU function	ALU control	LWIC
lw	00	load word	XXXXXX	add	0010	0
sw	00	store word	XXXXXX	add	0010	0
beq	01	branch equal	XXXXXX	subtract	0110	0
R-type	10	add	100000	add	0010	0
		subtract	100010	subtract	0110	0
		AND	100100	AND	0000	0
		OR	100101	OR	0001	0
		SLT	101010	SLT	0111	0
		load with increment	111111	add	0010	1

Problem 4.12

Examine the difficulty of adding a proposed `swap rs, rt` instruction to MIPS. Interpretation: `Reg[rt] = Reg[rs]; Reg[rs] = Reg[rt]`.

The new datapath would look as follows:



Control would look as follows:

Signal Name	R-Format	`lw`	`sw`	`beq`	`swap`
Op5	0	1	1	0	1
Op4	0	0	0	0	1
Op3	0	0	1	0	1
Op2	0	0	0	1	1
Op1	0	1	1	0	1
Op0	0	1	1	0	1
RegDst	1	0	X	X	0
ALUSrc	0	1	1	0	X
MemtoReg	0	1	X	X	X
RegWrite	1	1	0	0	1
MemRead	0	1	0	0	0
MemWrite	0	0	1	0	0
Branch	0	0	0	1	0
ALUOp1	1	0	0	0	1
ALUOp2	0	0	0	1	1

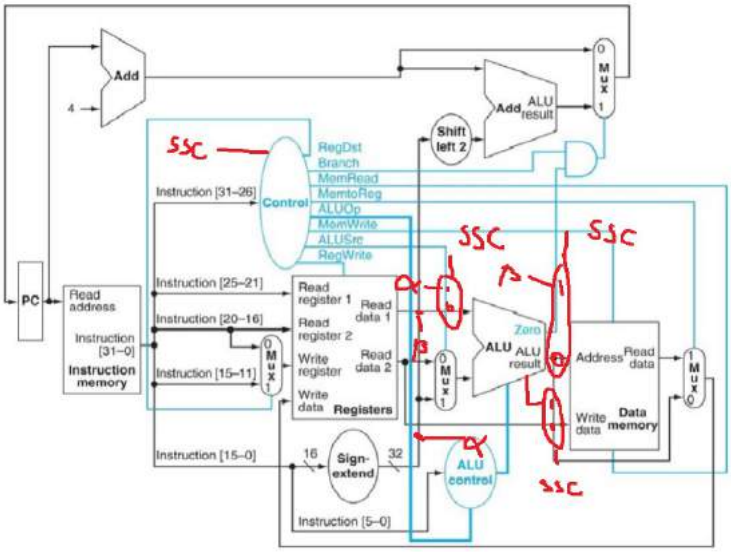
ALU control would look as follows:

Opcode	ALUOp	Operation	funct	ALU function	ALU control
lw	00	load word	XXXXXX	add	0010
sw	00	store word	XXXXXX	add	0010
beq	01	branch equal	XXXXXX	subtract	0110
R-type	10	add	100000	add	0010
		subtract	100010	subtract	0110
		AND	100100	AND	0000
		OR	100101	OR	0001
		SLT	101010	SLT	0111
swap	11	swap	XXXXXX	AND	0000

Problem 4.13

Examine the difficulty of adding a proposed `ss rt, rs, imm` (Store Sum) instruction to MIPS. Interpretation: `Mem[Reg[rt]]=Reg[rs]+SignExtend(immediate)`.

The new datapath would look as follows:



Control would look as follows:

Signal Name	R-Format	`lw`	`sw`	`beq`	`ss`
Op5	0	1	1	0	1
Op4	0	0	0	0	1
Op3	0	0	1	0	1
Op2	0	0	0	1	1
Op1	0	1	1	0	1
Op0	0	1	1	0	1
RegDst	1	0	X	X	X
ALUSrc	0	1	1	0	0
MemtoReg	0	1	X	X	X
RegWrite	1	1	0	0	0
MemRead	0	1	0	0	0
MemWrite	0	0	1	0	1
Branch	0	0	0	1	0
ALUOp1	1	0	0	0	1
ALUOp2	0	0	0	1	1

ALU control would look as follows:

Opcode	ALUOp	Operation	funct	ALU function	ALU control
lw	00	load word	XXXXXX	add	0010
sw	00	store word	XXXXXX	add	0010
beq	01	branch equal	XXXXXX	subtract	0110
R-type	10	add	100000	add	0010
		subtract	100010	subtract	0110
		AND	100100	AND	0000
		OR	100101	OR	0001
		SLT	101010	SLT	0111
ss	11	store sum	XXXXXX	add	0010