CS 61C Fall 2019

RISC-V Single Cycle Datapath

Discussion 7: October 14, 2019

1 Single-Cycle CPU

R = add

I = addi / lw

S = sw

SB = beq

UJ = jal

U = lui / auipc

1.1 For this worksheet, we will be working with the single-cycle CPU datapath on the last page.

(a) On the datapath, fill in each **round** box with the name of the datapath component, and each **square** box with the name of the control signal.

(b) Explain what happens in each datapath stage.

IF Instruction Fetch

Send address to the instruction memory, and read IMEM at that address.

ID Instruction Decode

Generate control signals from the instruction bits, generate the immediate, and read registers from the RegFile.

EX Execute

Perform ALU operations, and do branch comparison.

MEM Memory

Read from or write to the data memory.

WB Writeback

Write back either PC + 4, the result of the ALU operation, or data from memory to the RegFile.

Note: RegWEn should default to 0 so you do not accidentally write garbage.

1.2 Fill out the following table with the control signals for each instruction based on the datapath on the previous page. Wherever possible, use * to indicate that what this signal is does not matter.

	BrEq	BrLT	PCSel	ImmSel	BrUn	ASel	BSel	ALUSel	MemRW	RegWEn	WBSel
add	*	*	0 (PC + 4)	*	*	0 (Reg)	0 (Reg)	add	0	1	1 (ALU)
ori	*	*	0	I	*	0 (Reg)	1 (Imm)	or	0	1	1 (ALU)
lw	*	*	0	I	*	0 (Reg)	1 (Imm)	add	0	1	2 (MEM)
sw	*	*	0	S	*	0 (Reg)	1 (Imm)	add	1	0	*
beq	1/0	*	1/0	SB	*	1 (PC)	1 (Imm)	add	0	0	*
jal	*	*	1 (ALU)	UJ	*	1 (PC)	1 (Imm)	add	0	1	0 (PC + 4)
bltu	*	1/0	1/0	SB	1	1 (PC)	1 (Imm)	add	0	0	*

1.3 Clocking Methodology

- A **state element** is an element connected to the clock (denoted by a triangle at the bottom). The **input signal** to each state element must stabilize before each **rising edge**.
- The **critical path** is the longest delay path between state elements in the circuit. The circuit cannot be clocked faster than this, since anything faster would mean that the correct value is not guaranteed to reach the state element in the alloted time. If we place registers in the critical path, we can shorten the period by **reducing the amount of logic between registers**.

For this exercise, assume the delay for each stage in the datapath is as follows:

IF: 200 ps ID: 100 ps EX: 200 ps MEM: 200 ps WB: 100 ps

(a) Mark the stages of the datapath that the following instructions use and calculate the total time needed to execute the instruction.

	IF	ID	EX	MEM	WB	Total Time
add	X	X	X		X	600 ps
ori	X	X	X		X	600 ps
lw	X	X	X	X	X	800 ps
sw	X	X	X	X		700 ps
beq	X	X	X			500 ps
jal	X	X	X		X	600 ps
bltu	X	X	X			500 ps

200 + 100 + 200 + 100 = 600
IF + ID + EX + WB

load + store are only ones to use memory

branches does not writeback!

(b) Which instruction(s) exercise the critical path? ← longest delay path

Load word (lw), which uses all 5 stages.

(c) What is the fastest you could clock this single cycle datapath?

$$\frac{1}{\text{critical path}} = \frac{1}{800}\text{ picoseconds} = \frac{1}{800 * 10^{-12}}\text{ seconds} = 1,250,000,000s^{-1} = 1.25GHz$$

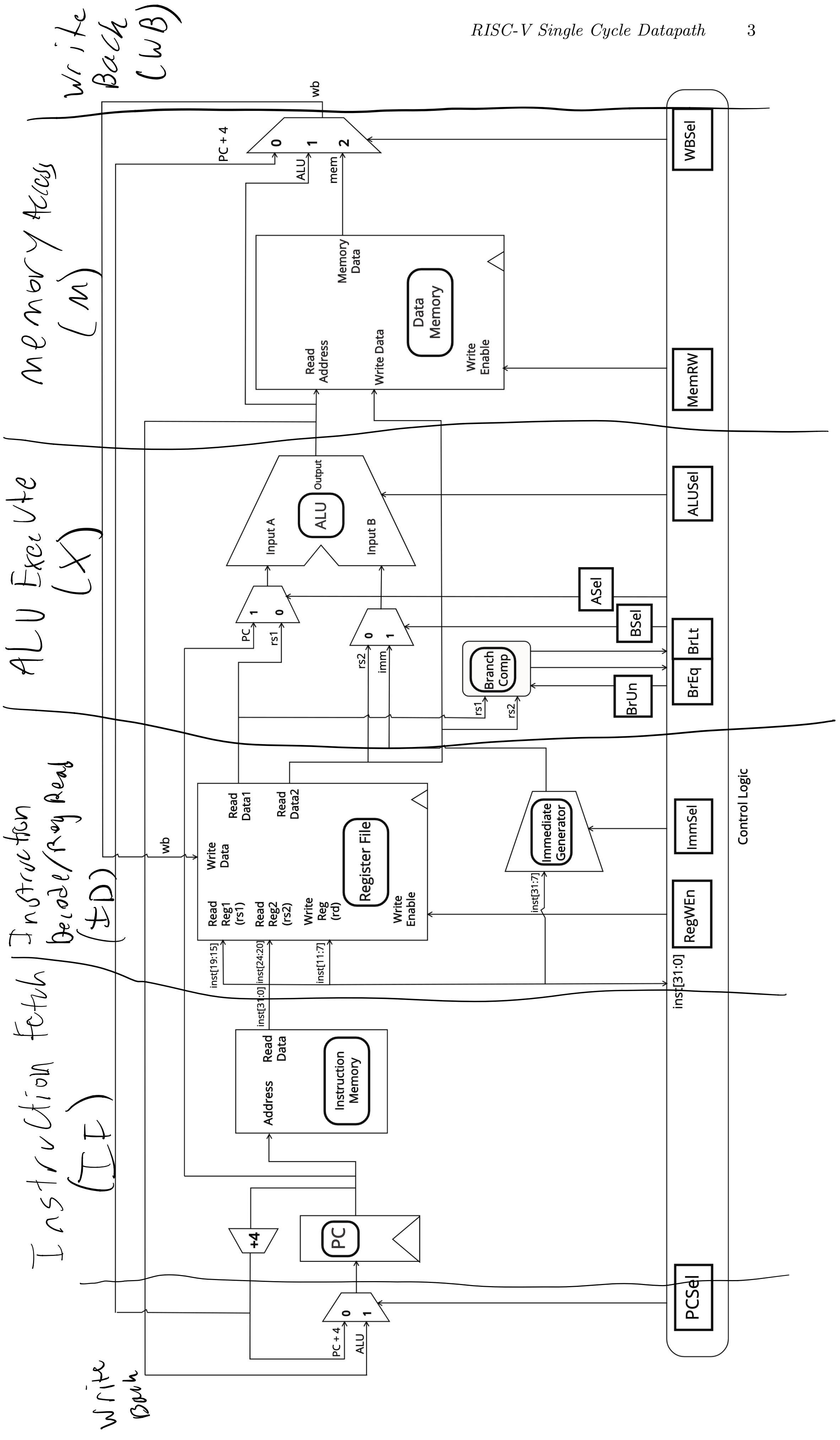
(d) Why is the single cycle datapath inefficient?

At any given time, most of the parts of the single cycle datapath are sitting unused. Also, even though not every instruction exercises the critical path, the datapath can only be clocked as fast as the slowest instruction.

(e) How can you improve its performance? What is the purpose of pipelining?

Performance can be improved with pipelining, or putting registers between stages so that the amount of combinational logic between registers is reduced, allowing for a faster clock time.

A caviotis you have to make all sections take the same time thus you have to set it to the longest time regardless of the stage. In our example, the clock is at least 200 ps.


Instruction Fetch (IF)
Instruction Decode/Reg Read (ID)
ALU Execute (X)
Memory Access (M)
Write Back (WB)
Write Back
PC
+4
PC + 4
ALU
0
1
PCSel
Address
Read Data
Instruction Memory
inst[31:0]
inst[19:15]
inst[24:20]
inst[11:7]
inst[31:7]
Read Reg1 (rs1)
Read Reg2 (rs2)
Write Reg (rd)
Write Data
Read Data1
Read Data2
Register File
Write Enable
wb
Immediate Generator
ImmSel
RegWEn
PC
rs1
rs2
imm
ASel
BSel
Branch Comp
BrUn
BrEq
BrLt
Input A
Input B
ALU
Output
ALUSel
Read Address
Write Data
Memory Data
Data Memory
Write Enable
MemRW
mem
2
WBSel
Control Logic