CS 61C Fall 2019 — RISC-V Single Cycle Datapath

Discussion 7: October 14, 2019

1 Single-Cycle CPU

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

ID Instruction Decode

EX Execute

MEM Memory

WB Writeback

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											
ori											
lw											
sw											
beq											
jal											
bltu											

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.

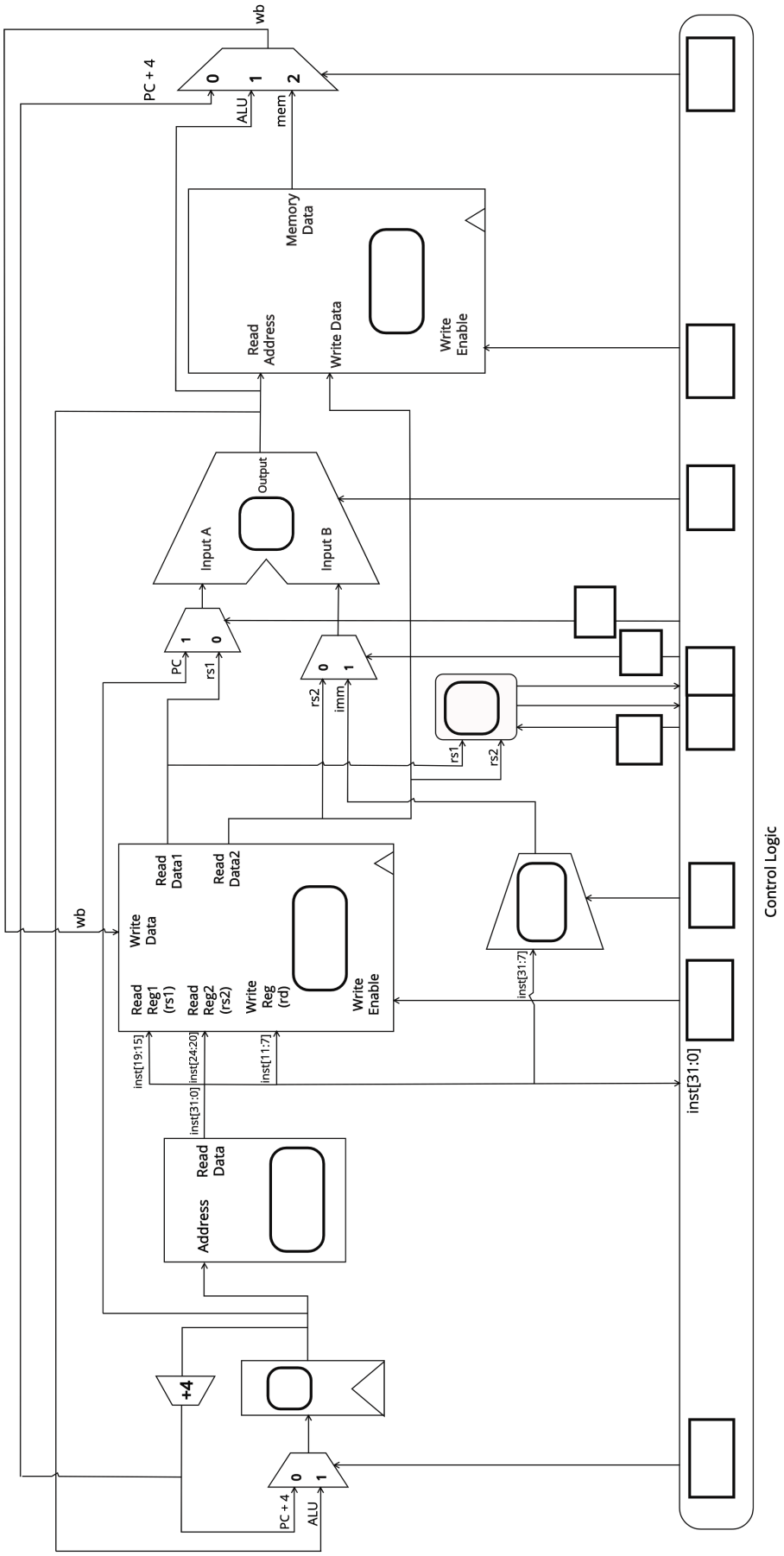
	IF	ID	EX	MEM	WB	Total Time
add						
ori						
lw						
sw						
beq						
jal						
bltu						

(b) Which instruction(s) exercise the critical path?

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

(d) Why is the single cycle datapath inefficient?

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


wb
PC + 4
ALU
mem
0
1
2
Memory Data
Read Address
Write Data
Write Enable
Output
Input A
Input B
PC
rs1
rs2
imm
Read Data1
Read Data2
Write Data
Read Reg1 (rs1)
Read Reg2 (rs2)
Write Reg (rd)
Write Enable
inst[31:7]
inst[19:15]
inst[24:20]
inst[11:7]
inst[31:0]
Read Data
Address
+4
Control Logic