CS 61C Spring 2023 — RISC-V Single Cycle Datapath

Discussion 7

1 Pre-Check

This section is designed as a conceptual check for you to determine if you conceptually understand and have any misconceptions about this topic. Please answer true/false to the following questions, and include an explanation:

1.1 The single cycle datapath makes use of all hardware units for each instruction.

1.2 It is possible to execute the stages of the single cycle datapath in parallel to speed up execution of a single instruction.

1.3 If the delay of reading from IMEM is reduced, then any (non-empty) program using the single cycle datapath will speed up.

1.4 The control signals used throughout all datapath stages to guide a correct 'execution line' all come from decoding an instruction's unique binary encoding only.

1.5 Storing instructions and loading instructions are the only instructions that require input/output from DMEM.

1.6 It is possible to use both the output of the immediate generator and the value in register rs2.

2 Single-Cycle CPU

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

(a) Explain what happens in each datapath stage, and which hardware units in the datapath are used.

IF Instruction Fetch

ID Instruction Decode

EX Execute

MEM Memory

WB Writeback

(b) 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.

(c) List all possible values that each control signal may take on for the single cycle datapath, then briefly describe what each value means for each signal.

Signal Name	Values	Signal Name	Values
PCSel		RegWEn	
ImmSel		BrEq	
BrLt		ALUSel	
MemRW		WBSel	

2.2 Fill out the following table with the control signals for each instruction based on the datapath on the last page. If the value of the signal does not affect the execution of an instruction, use the `*` (don't care) symbol to indicate this. If the value of the signal **does** affect the execution, but can be different depending on the program, list all possible values (for example, for a signal with possible values of 0 and 1, write 0/1).

	BrEq	BrLT	PCSel	ImmSel	BrUn	ASel	BSel	ALUSel	MemRW	RegWEn	WBSel
add											
ori											
lw											
sw											
beq											
jal											
blt											

3 Timing the Datapath

3.1 **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, the delay for each circuit element is given as follows:

Clk-to-Q	RegFile Read	RegFile Setup	Mux
5ns	35ns	20ns	15ns

ALU	Branch Comp	Imm Gen	MEM Read	MEM Write
100ns	50ns	45ns	300ns	200ns

(a) Mark the stages of the datapath that the following instructions use:

	IF	ID	EX	MEM	WB	
add						
ori						
lw						
sw						
beq						
jal						

(b) How long does it take to execute each instruction? Ignore the length of a clock cycle based off of the critical path, and assume that the setup times to the RegFile and the PC are the same.

1. jal

2. lw

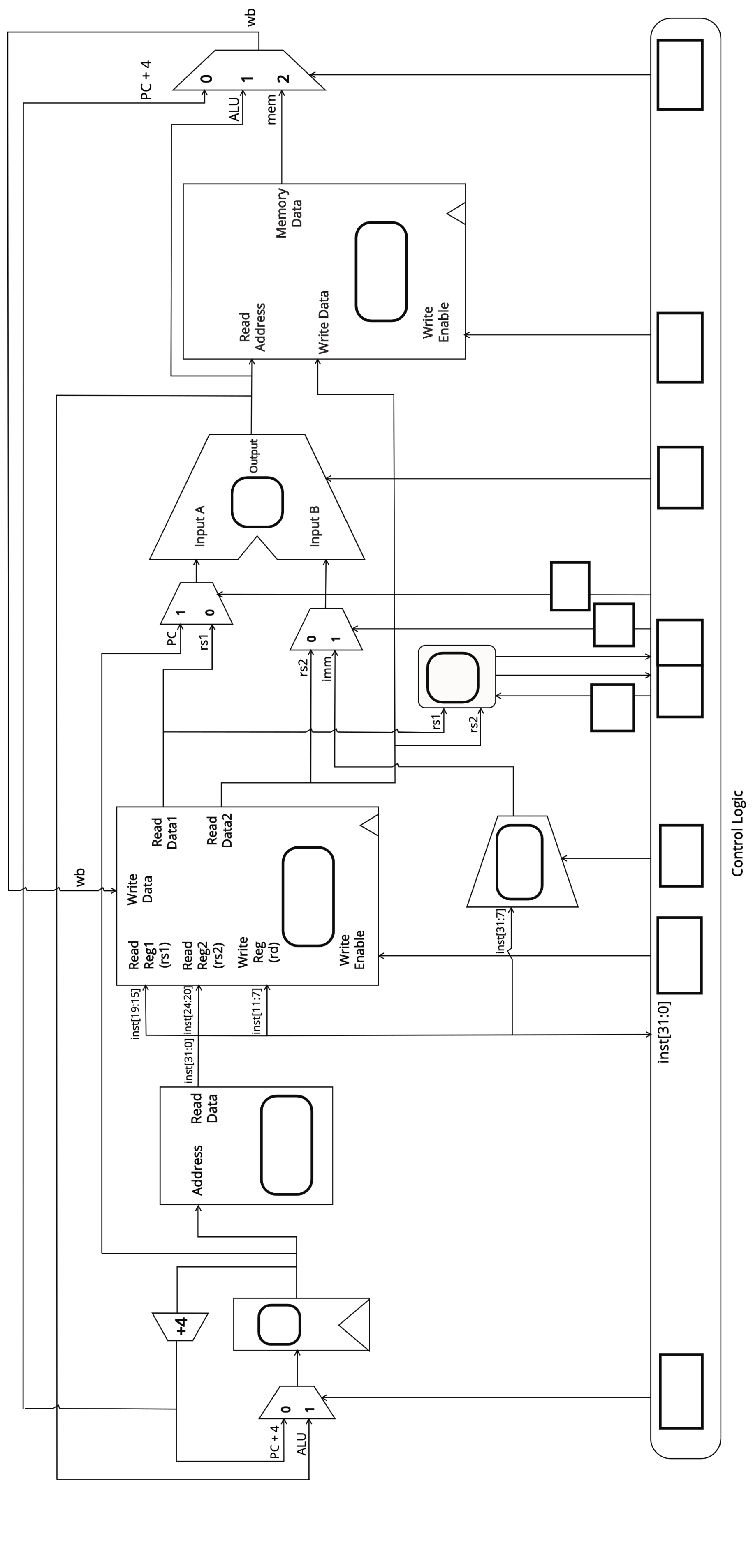
3. sw

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

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

(e) Why is the single cycle datapath inefficient?

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


wb
PC + 4
0
1
2
ALU
mem
Memory Data
Read Address
Write Data
Write Enable
Output
Input A
Input B
1
0
PC
rs1
0
1
rs2
imm
rs1
rs2
Read Data1
Read Data2
Write Data
wb
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]
inst[31:0]
Control Logic
Read Data
Address
+4
0
1
PC + 4
ALU