CS 61C Summer 2020 — RISC-V Single Cycle Datapath

Discussion 7: July 13, 2020

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.

false

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

false.

1.3 Combinational logic is only used in the instruction decode stage.

false

2 Single-Cycle CPU

2.1 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 inst
read inst

ID Instruction Decode

generate control sig from inst.
generate imm. read register

EX Execute

ALU operation & branch execution

MEM Memory

read/write to DMem

WB Writeback

write back to reg file
select source to
↓
mem / PC+4 / ALU.

2.2 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 (as in, letting the value be whatever it wants won't affect the execution of the instruction). If the value of the signal does matter for correct execution, but can vary, list all of the values (for example, for a signal that matters with possible values of 0 and 1, write 0/1).

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

2.3 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	✓	600
ori	✓	✓	✓	X	✓	600
lw	✓	✓	✓	✓	✓	800
sw	✓	✓	✓	✓	X	700
beq	✓	✓	✓	X	X	500
jal	✓	✓	✓	X	✓	600
bltu	✓	✓	✓	X	X	500

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

load.

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

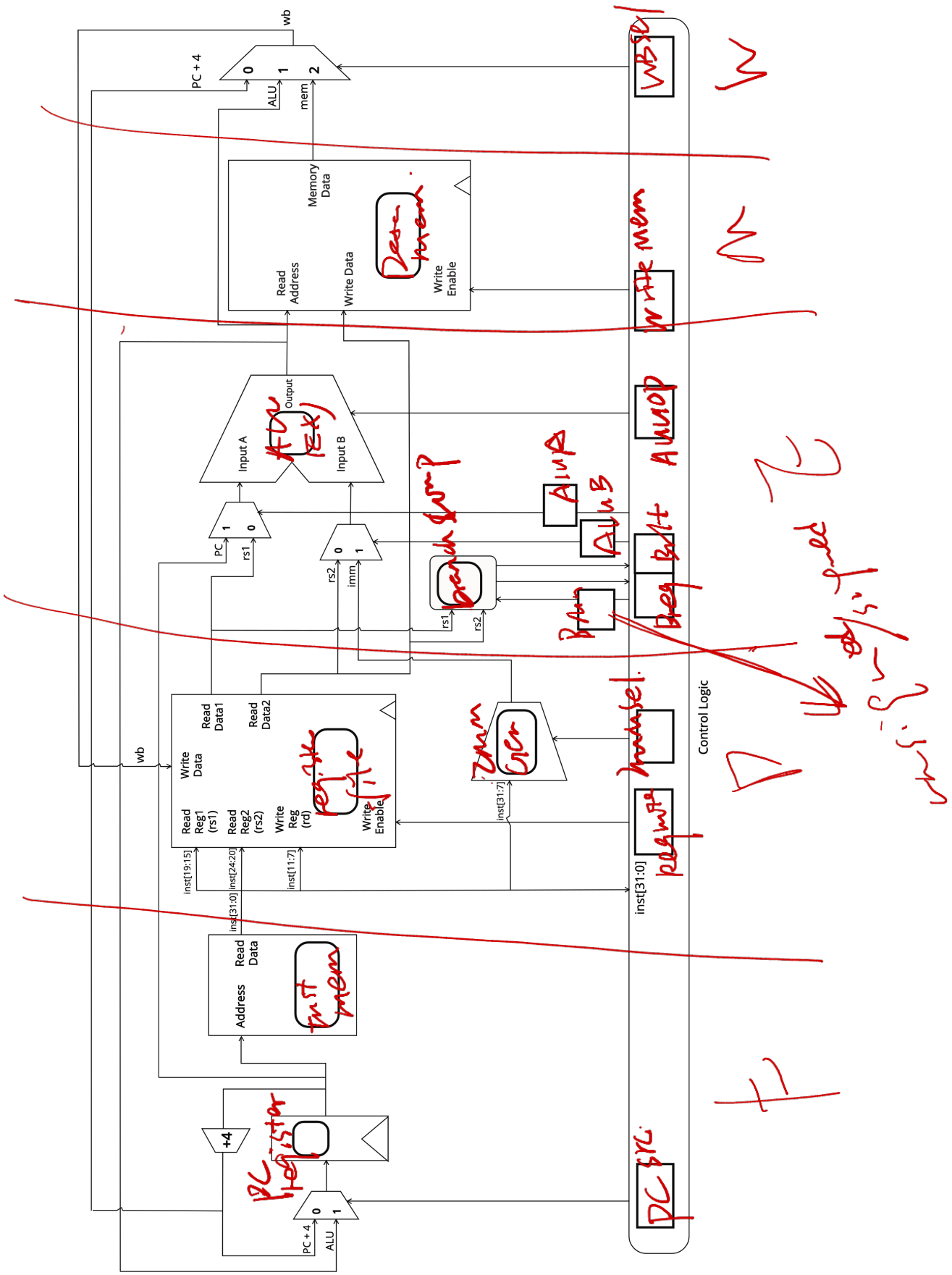
$\frac{1}{800ps} = \frac{1}{\text{how fast}}$: $1.25 \times 10^{9}\ s^{-1}$ 1.25 GHz

(d) Why is the single cycle datapath inefficient?

most components unused
critical path

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

pipelining
execute multiple inst. in one cycle on different stages
faster clock


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