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. Although most units will be lit, they won't be used

All units are active in each cycle, but their output may be ignored (gated) by control signals

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 We can execute in parallel to shorten the whole process but can't shorten a single.

Each stage depends on the value produced by the stage before it (e.g., instruction decode depends on the instruction fetched).

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

False It's used everywhere

Other stage executes combinational logic too (muxes for instruction fetch, memory write, register updates, ALU operations during execute)

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 address to the Instruction Memory (IMEM), and read IMEM Instruction Memory receive instruction signal from PC at that address.

ID Instruction Decode Generate control signals from the instruction bits, generate the immediate, Instruction is decoded in IMEM into several parts, then passed to next parts, and read registers setting them into the right pattern from the Register File

EX Execute ALU receives data from Read Data 1 and two numbers Perform ALU operations, and do branch comparison.

MEM Memory The output of ALU is written to Data Memory

WB Writeback Read from or write to the data memory (DMEM)

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

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	*	*	+4	*	*	Reg	Reg	add	Read	1	ALU
ori	*	*	+4	I	*	Reg	Imm	or	0	1	ALU
lw	*	*	+4	I	*	Reg	Imm	Add	Read	1	Mem
sw	*	*	+4	S	*	Reg	Imm	Add	Write	0	*
beq	0/1	*	+4/ALU	B	*	PC	Imm	Add	Read	0	*
jal	*	*	ALU	J	*	PC	Imm	Add	Read	1	PC+4
bltu	*	1	ALU	B	1	PC	Imm	Add	Read	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.

	200 IF	100 ID	200 EX	200 MEM	100 WB	Total Time
add	✓	✓	✓		✓	600 ps
ori	✓	✓	✓		✓	~~600ps~~
lw	✓	✓	✓	✓	✓	~~800ps~~
sw	✓	✓	✓	✓		~~700ps~~
beq	✓	✓	●			500ps
jal	●	●	●	✗	●	~~500ps~~ 600ps
bltu	●	●	●			500ps

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

lw ✓, which uses all 5 stages

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

1.25 GHz

$1 ps = 10^{-12} s$

(d) Why is the single cycle datapath inefficient?

While one instruction is using the hardwares, most of them are not being used

At any given time, most of the parts of the single cycle datapath are sitting unused

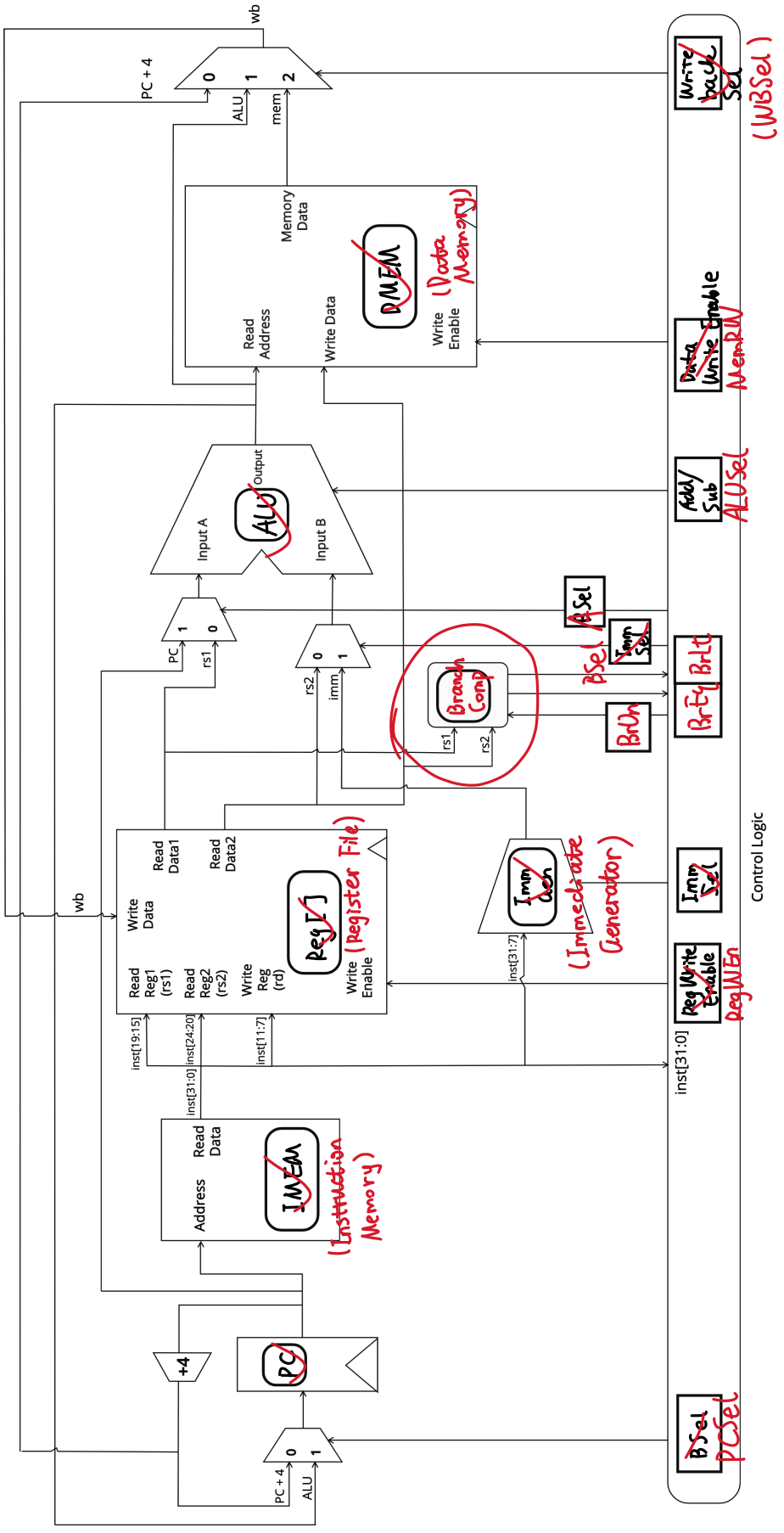
Also, even though not every instruction exercises the critical path, the datapath can only be clocked as fast as the slowest instruction

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

Use pipeline by adding registers between different parts

At any given time, ~~almost~~ all the hardwares are bing used

Perform 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.


wb
PC + 4
ALU
mem
0
1
2
Memory Data
Read Address
Write Data
DMEM
Write Enable
(Data Memory)
Input A
Output
ALU
Input B
PC
rs1
rs2
imm
Branch Comp
BrUn
BrEq
BrLt
ASel
ImmSel
Write back Sel
(WBSel)
Data Write Enable
MemRW
Add/Sub
ALUSel
Imm Gen
(Immediate Generator)
Imm Sel
Reg Write Enable
RegWEn
Read Data1
Read Data2
Write Data
Read Reg1 (rs1)
Read Reg2 (rs2)
Write Reg (rd)
Write Enable
Reg[]
(Register File)
inst[19:15]
inst[24:20]
inst[11:7]
inst[31:0]
inst[31:7]
inst[31:0]
Address
Read Data
IMEM
(Instruction Memory)
+4
PC
BSel
PCSel
Control Logic