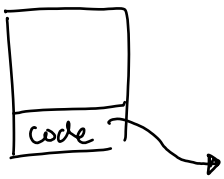
CS 61C Fall 2018

Single-Cycle Datapath

Discussion 11: November 5, 2018



Explained

PC: program counter
- current inst. addr.

IMEM: Instruction Memory
- inst. address —read→ instruction
  - ↑ addr
  - ↑ machine code

RegFile: Register File
- our 32 RISC-V registers
- read & write to them

Imm Gen: Immediate Generator
- create 32-bit immediate from machine code

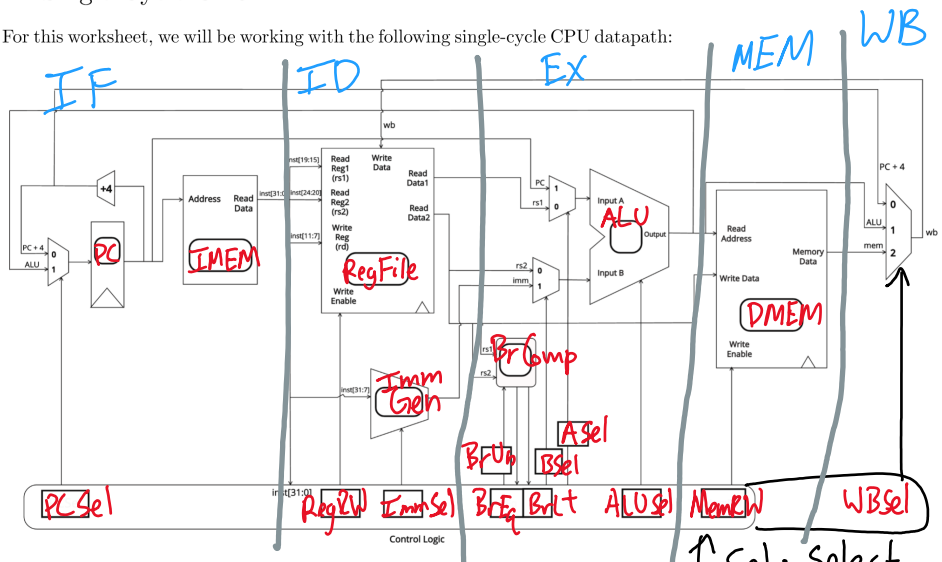
Br Comp: Branch Comparator
- evaluate branching conditions

ALU: Arithmetic Logic Unit
- arith. operations +, −, ×, ÷, etc.

DMEM: Data Memory
- where Data (not machine code) is stored (e.g. arrays [ ])

1 Single-Cycle CPU

1.1 For this worksheet, we will be working with the following single-cycle CPU datapath:



(a) On the datapath above, 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 addr to IMEM, read at that addr (get instruction to execute)

ID Instruction Decode — Generate control signals & read RegFile based on instruction (figure out what to do w/ inst.)

EX Execute — perform ALU op & branch comp. (do computations)

MEM Memory — Read/Write from memory (same ↗)

WB Writeback — Write Back PC+4, ALU computation, or read data to RegFile (store necessary values back)

[Sel: Select
- choose what input to pass through

✡ = easier way to remember

look @ diagram at end

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	*	*	0	0	add	0	1	1
ori											
lw											
sw	*	*	0	S	*	0	1	add	1	0	*
beq	1/0	*	1/0	SB	*	1	1	add	0	0	*
jal	*	*	1	UJ	*	1	1	add	0	1	0
bltu											

1.3 **Clocking Methodology**

↑ now switch next inst based on computation

notice write always matters (can't just write to mem/reg wherever we want)

- 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. 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
ori						
lw	X	X	X	X	X	800
sw	X	X	X	X		700
beq	X	X	X			500
jal	X	X	X		X	600
bltu						

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

- lw (not shown, but can do same thing as others)

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

$1/800ps = 1.25\ GHz$

(d) Why is the single cycle datapath inefficient?

idle components

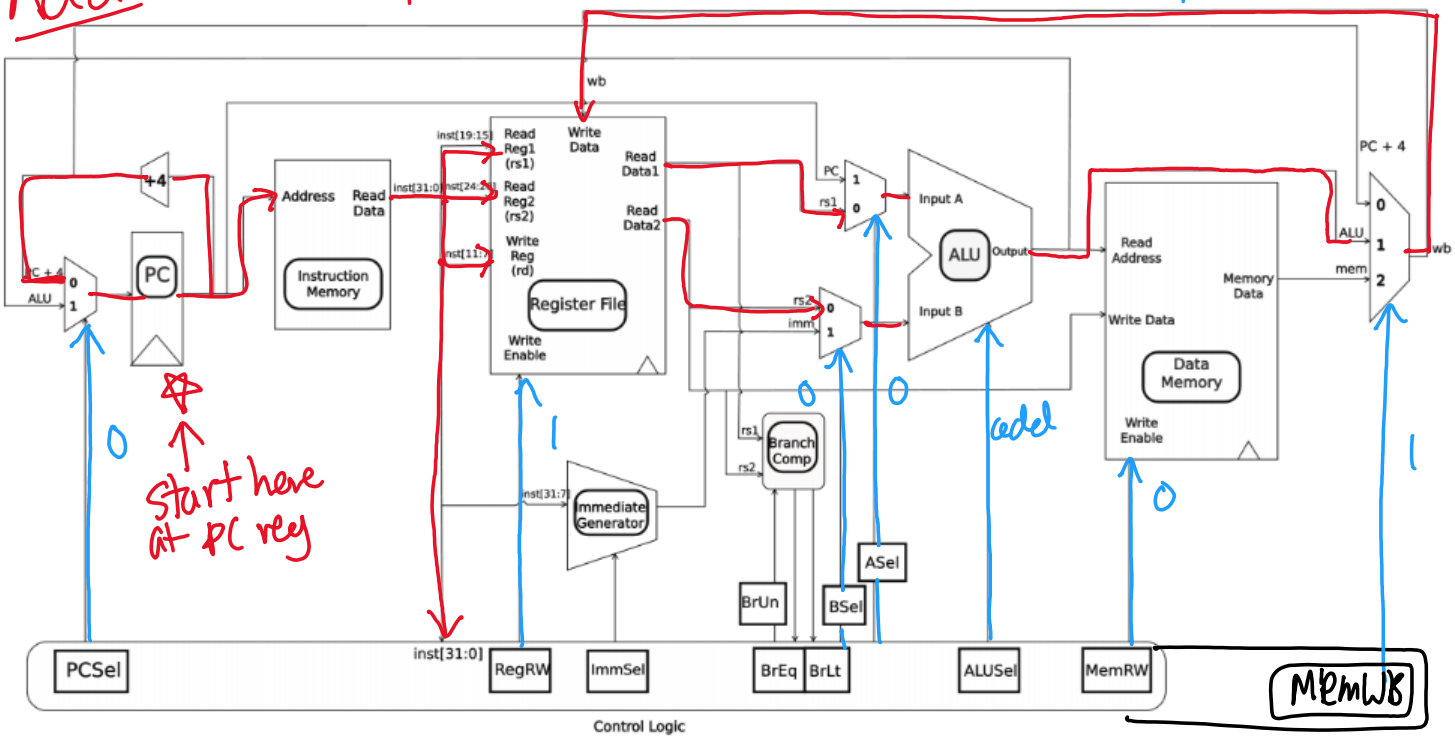
(e) How can you improve its performance? What is the purpose of pipelining?

pipeline, make use of idle components → do mult inst. at same time.

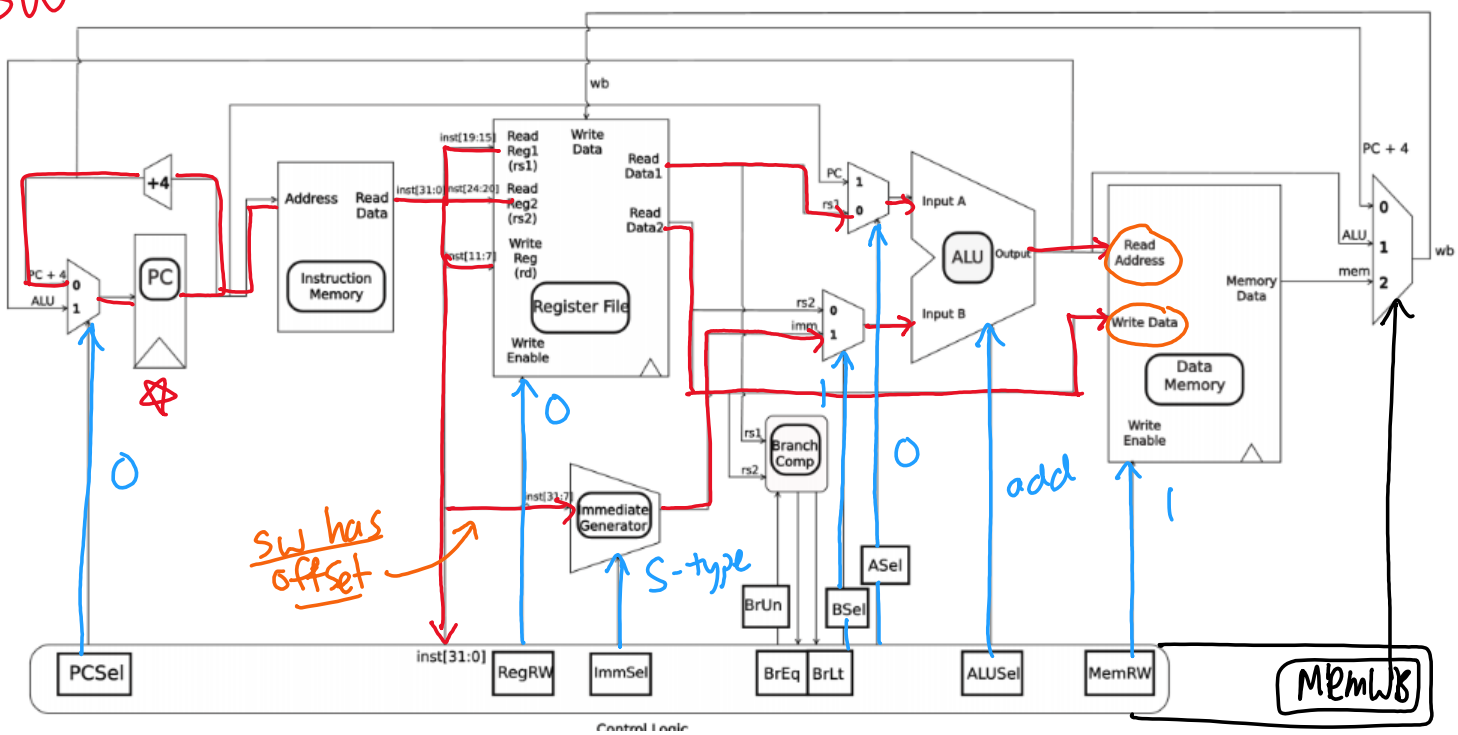
pipeline, make use of idle components
→ do mult inst. at same time.

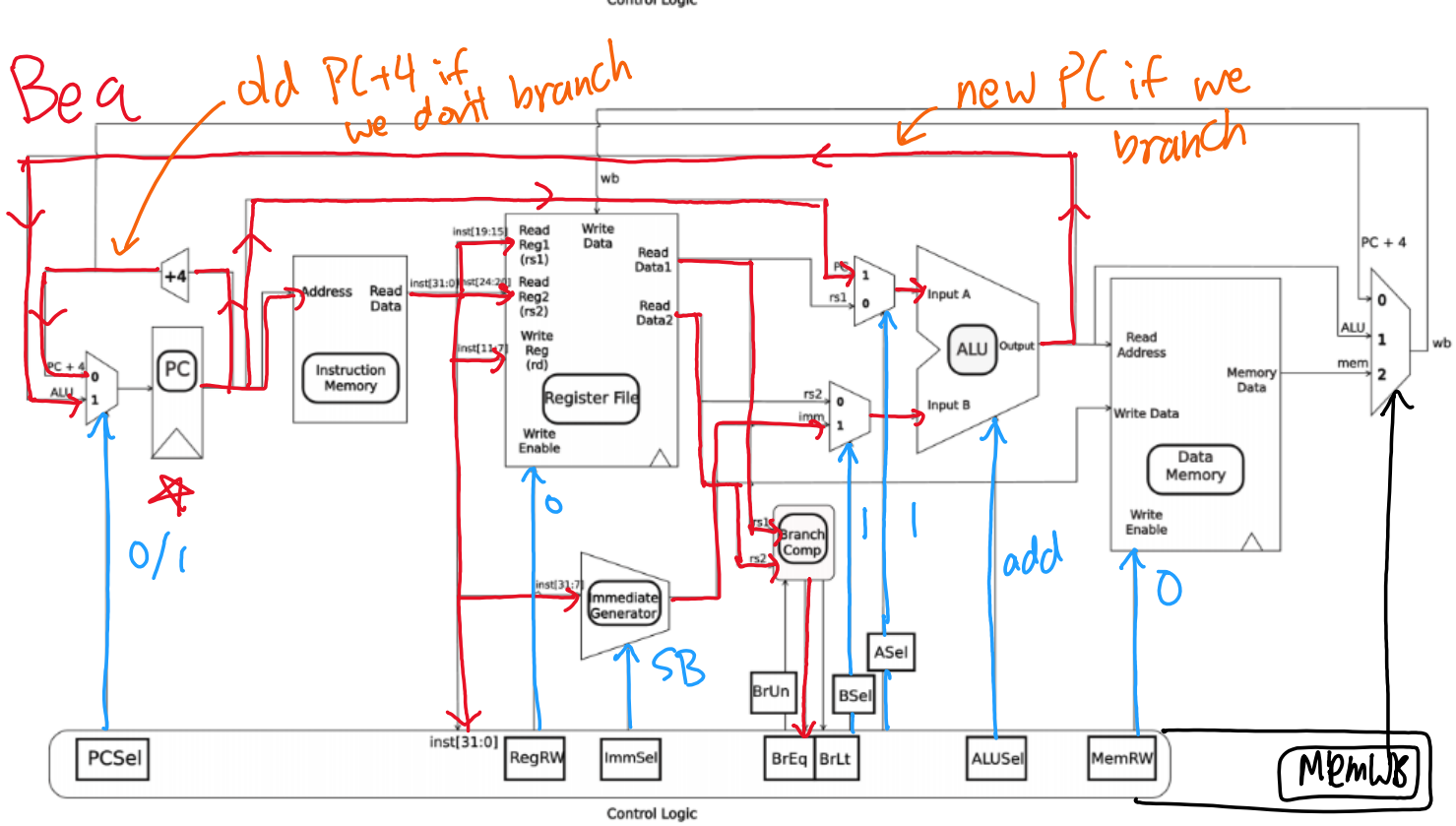
Add

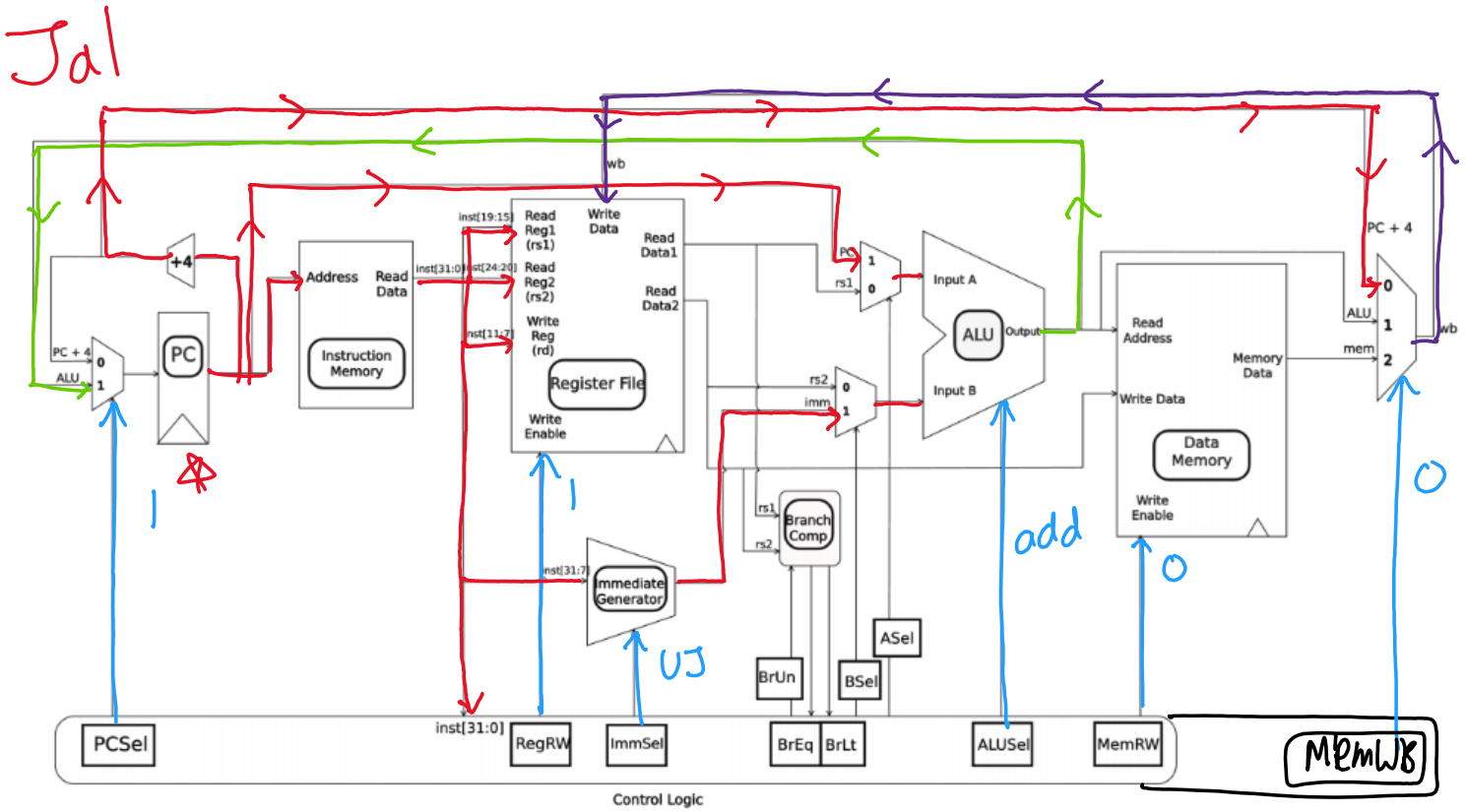
✡ = "operation" signal ✡ = used control signal



SW




Beq
old PC+4 if we don't branch
new PC if we branch
wb
PC + 4
+4
PC
ALU
Address
Read Data
Instruction Memory
inst[19:15]
inst[31:0]
inst[24:20]
inst[11:7]
Read Reg1 (rs1)
Read Reg2 (rs2)
Write Reg (rd)
Write Data
Read Data1
Read Data2
Register File
Write Enable
rs1
rs2
imm
Input A
Input B
ALU
Output
Read Address
Memory Data
Write Data
Data Memory
Write Enable
mem
0/1
0
1
1
add
0
Branch Comp
Immediate Generator
SB
BrUn
BSel
ASel
PCSel
RegRW
ImmSel
BrEq
BrLt
ALUSel
MemRW
MRMWB
Control Logic


Jal
wb
PC + 4
+4
PC
ALU
Address
Read Data
Instruction Memory
inst[19:15]
inst[31:0]
inst[24:20]
inst[11:7]
Read Reg1 (rs1)
Read Reg2 (rs2)
Write Reg (rd)
Write Data
Read Data1
Read Data2
Register File
Write Enable
rs1
rs2
imm
Input A
Input B
ALU
Output
Read Address
Memory Data
Write Data
Data Memory
Write Enable
mem
1
1
add
0
Branch Comp
Immediate Generator
UJ
BrUn
BSel
ASel
PCSel
RegRW
ImmSel
BrEq
BrLt
ALUSel
MemRW
MRMWB
Control Logic