CSED311: Lab 4
Multi-cycle CPU

Jiwoong Shin

jwshin0610@postech.ac.kr

Contact the TAs at csed311-ta@postech.ac.kr

Objectives

- **To understand why a multi-cycle CPU is better** than the single-cycle implementation

- **To design and implement a multi-cycle CPU**, which has its own datapath and control unit

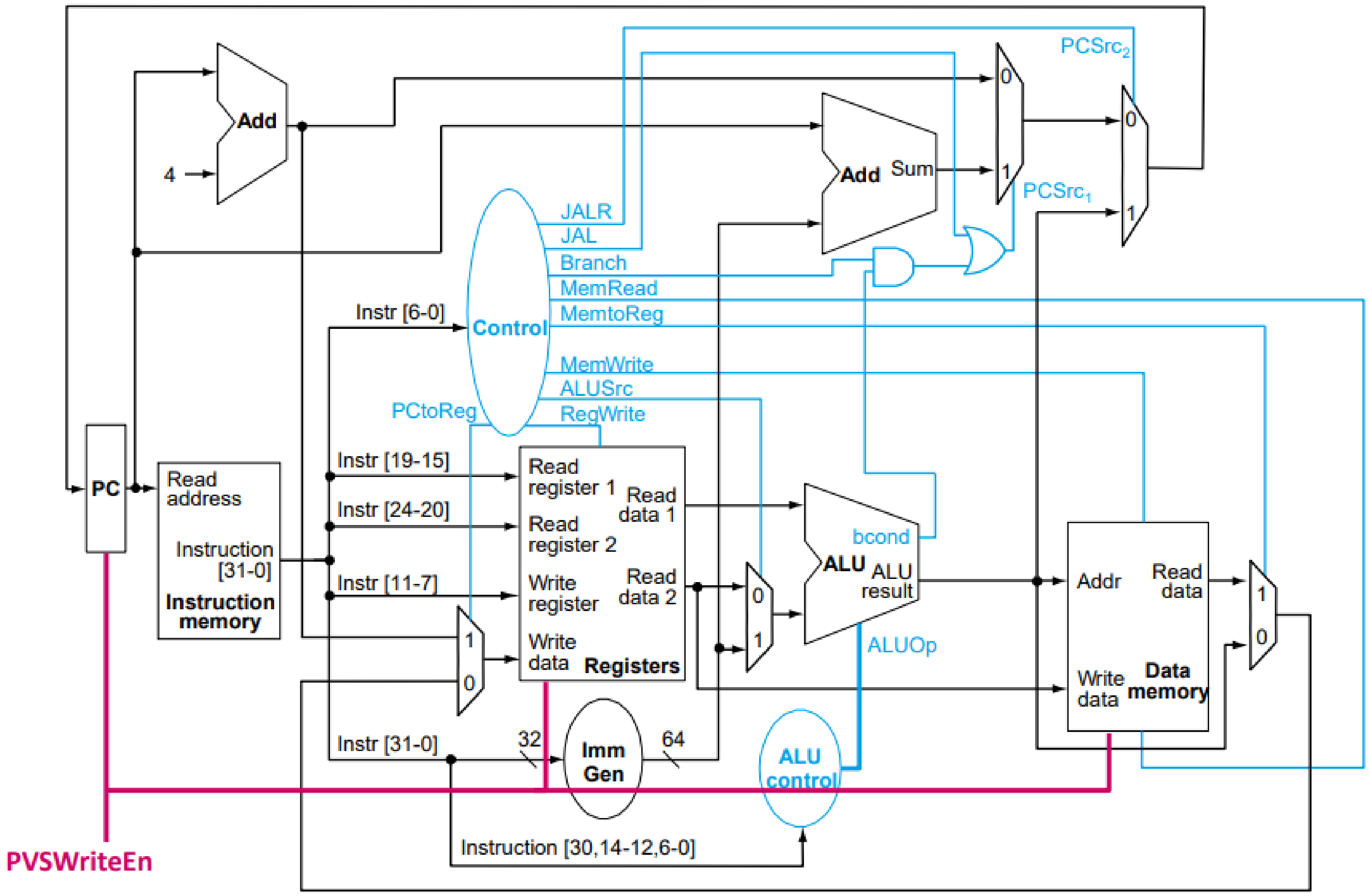
Why do we need multi-cycle CPU?

- **Problem on single-cycle CPU : underutilization of resources (ALU, memory, register file, etc.)**
- **Solution:** use higher clock frequency and allocate different number of cycles for each instruction type

Memory units (read or write):	200 ps
ALU (add op) :	100 ps
Register file (read or write):	50 ps
Other combinational logic:	0 ps

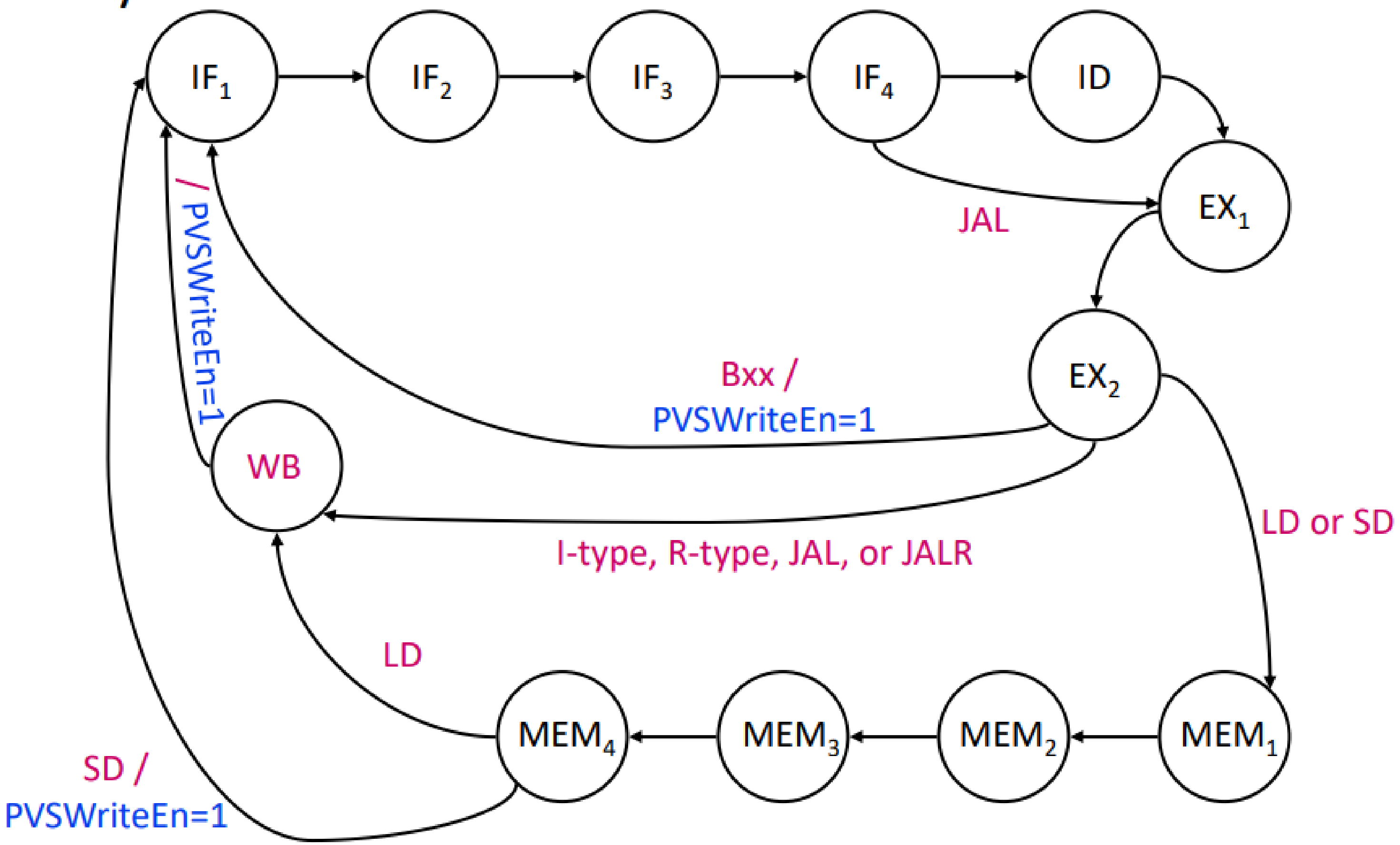
Steps	IF	ID	EX	MEM	WB	Delay
Resources	mem	RF	ALU	mem	RF	
R-type	200	50	100		50	400
I-type	200	50	100		50	400
LD	200	50	100	200	50	600
SD	200	50	100	200		550
Bxx	200	50	100			350
JAL	200		100		50	350
JALR	200	50	100		50	400

Multi-cycle CPU (Datapath)

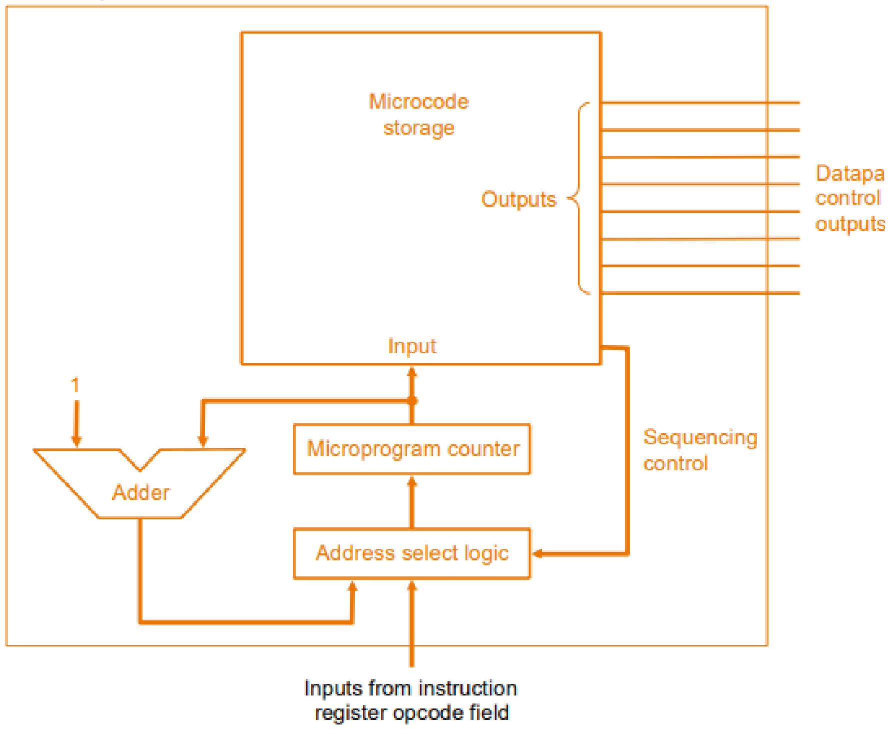


Multi-cycle CPU (Finite state machine)

Mealy FSM



Multi-cycle CPU (Micro-code controller)



State label	Control flow	Conditional targets					
		R/I-type	LD	SD	Bxx	JALR	JAL
IF_1	next	-	-	-	-	-	-
IF_2	next	-	-	-	-	-	-
IF_3	next	-	-	-	-	-	-
IF_4	go to	ID	ID	ID	ID	ID	EX_1
ID	next	-	-	-	-	-	
EX_1	next	-	-	-	-	-	-
EX_2	go to	WB	MEM_1	MEM_1	IF_1	WB	WB
MEM_1	next		-	-			
MEM_2	next		-	-			
MEM_3	next		-	-			
MEM_4	go to		WB	IF_1			
WB	go to	IF_1	IF_1			IF_1	IF_1
CPI		8	12	11	7	8	7

Multi-cycle CPU

- Details for multi-cycle CPU are given in the **lecture note and textbook**
- Please read those materials yourself to work on it

Let’s discuss the design and implementation

The testbench file

cpu UUT (clk, reset_n, readM, writeM, address, data, num_inst, output_port, is_halted);

Memory NUUT(!clk, reset_n, readM, writeM, address, data);

- Note that the memory is not using the clock signal properly.
  - It is done to allow the CPU to use access the memory in a single cycle and make it easier to implement.
  - Such use of clk is **forbidden** within your code.

Assignment (1)

- Implement a multi-cycle CPU
  - Datapath for a 16-bit CPU
  - Full support for TSC instruction set except for **RWD, ENI, DSI**
  - **num_inst** should be increased when the CPU finishes executing an instruction
  - The WWD instruction outputs the register value at the **output_port**
  - The HLT instruction sets the **is_halted** signal
  - Other instructions follow the TSC instruction set manual
- The datapath and the control unit should be separated
  - The control unit controls the finite state machine
  - The control unit gives appropriate signals to the datapath

Assignment (2)

- Your implementation should pass all tests in the testbench
  - But the message of "All pass!" does not mean you will get a full score
- Your control unit should have well-designed states
- Your control unit should be a well-implemented state machine
- Each state should generate its control signals
- All your circuits (Datapath + Control unit) should be **clock-synchronized**
  - You should not use "delay(#)" nor "wait"
  - All storage units (registers, PC, etc.) must be updated only at the clock's positive edges
- Your code should have **"resource reuse"**, which affects your control unit design
  - E.g. Using only one RAM port, combining a "PC + 1" logic with the ALU

Submission

- Please submit your **report** and **codes** to LMS
  - **Due date for the codes: 2020/5/11 (Mon) 9:00am**
  - **Due date for the report: 2020/5/11 (Mon) 20:00pm**
- The file names for report and codes follow **these formats**
  - **"Lab4_TeamID_StudentID1[_StudentID2].pdf"**: PDF file for your report
    - ex) Lab1_20_20180001_20180002.pdf, Lab1_21_20180003.pdf (for a single-person team)
  - **"Lab4_TeamID_StudentID1[_StudentID2].zip"**: Zip file for your own codes (*.v) only
- **If you do not follow the given format, you will get penalty on your score**