CSED311 Lab4: Multi-cycle CPU

Byeong-Hoon So
sbh4728@postech.ac.kr

Objectives

- To understand the reason of transferring from a single-cycle implementation to a multi-cycle implementation
- To design & implement a multi-cycle CPU, which has its own datapath and control unit

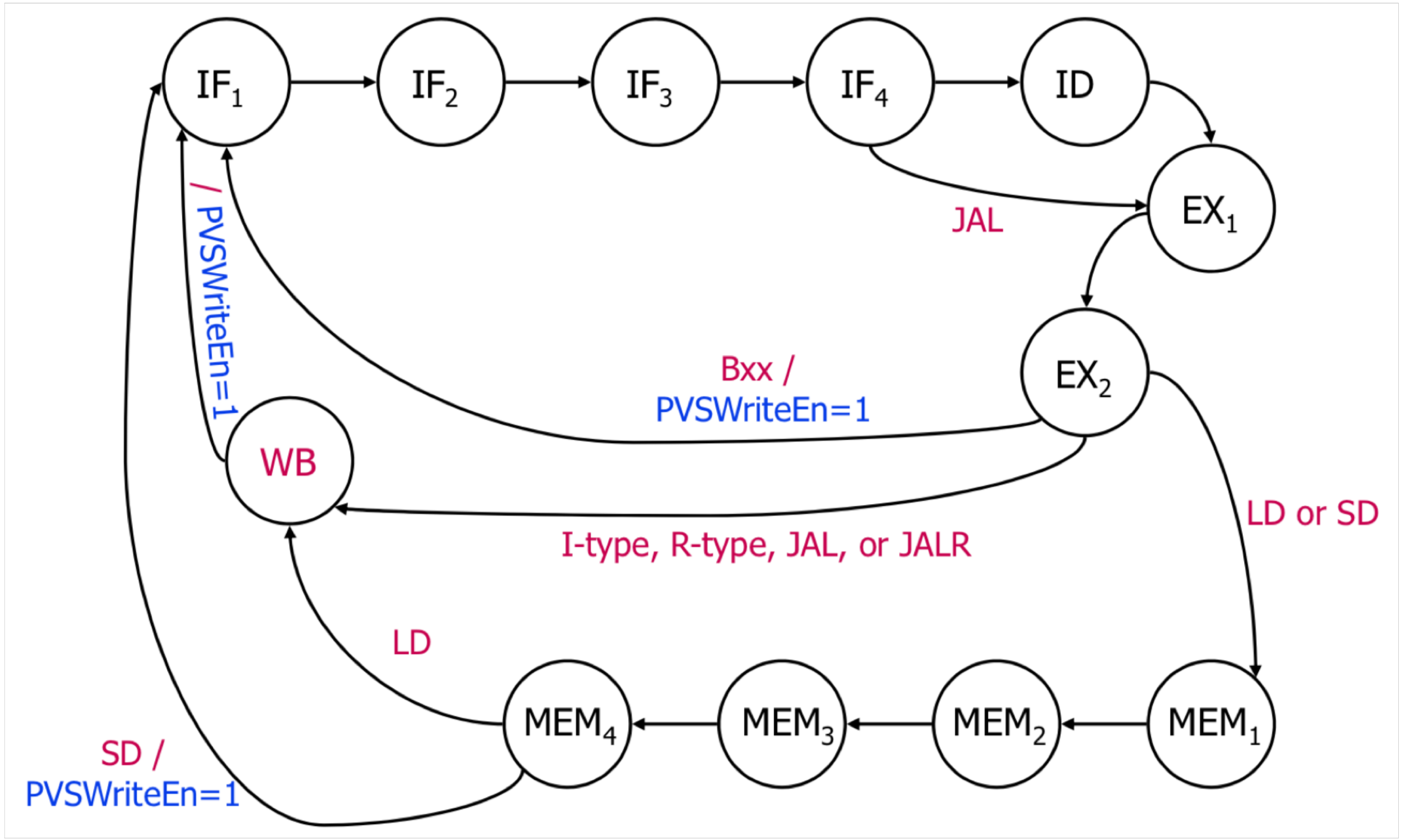
Why multi-cycle?

- Memory units (read or write): 200 ps
- ALU (add op) : 100 ps

- Register file (read or write): 50 ps
- Other combinational logic: 0 ps

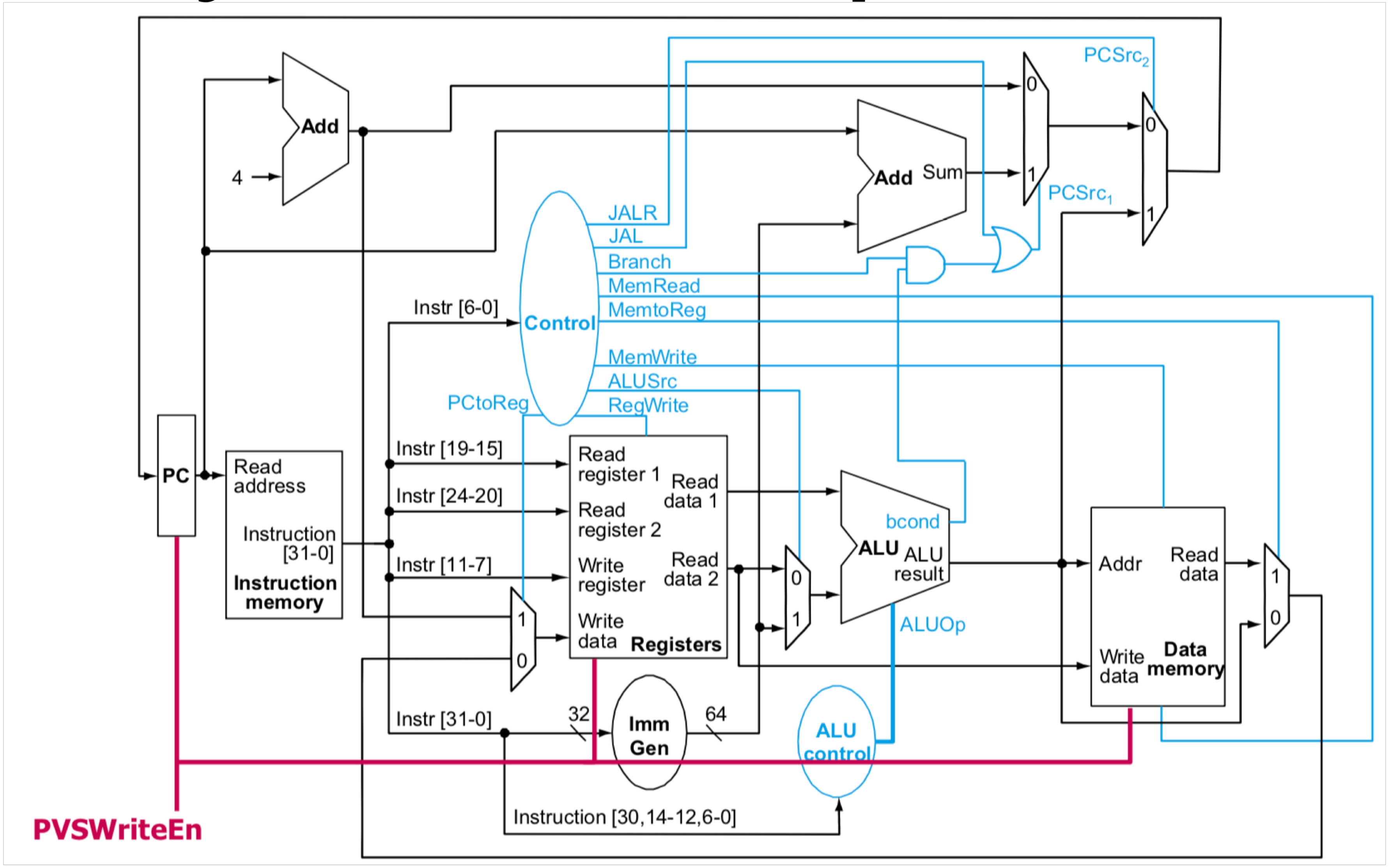
Steps	IF	ID	EX	MEM	WB	Delay
Resources	mem	RF	ALU	mem	RF	
R-type	200	50	100		50	400
I-type	200	50	100		50	400
LW	200	50	100	200	50	600
SW	200	50	100	200		550
Branch	200	50	100			350
Jump	200					200

Multi-cycle CPU (FSM)



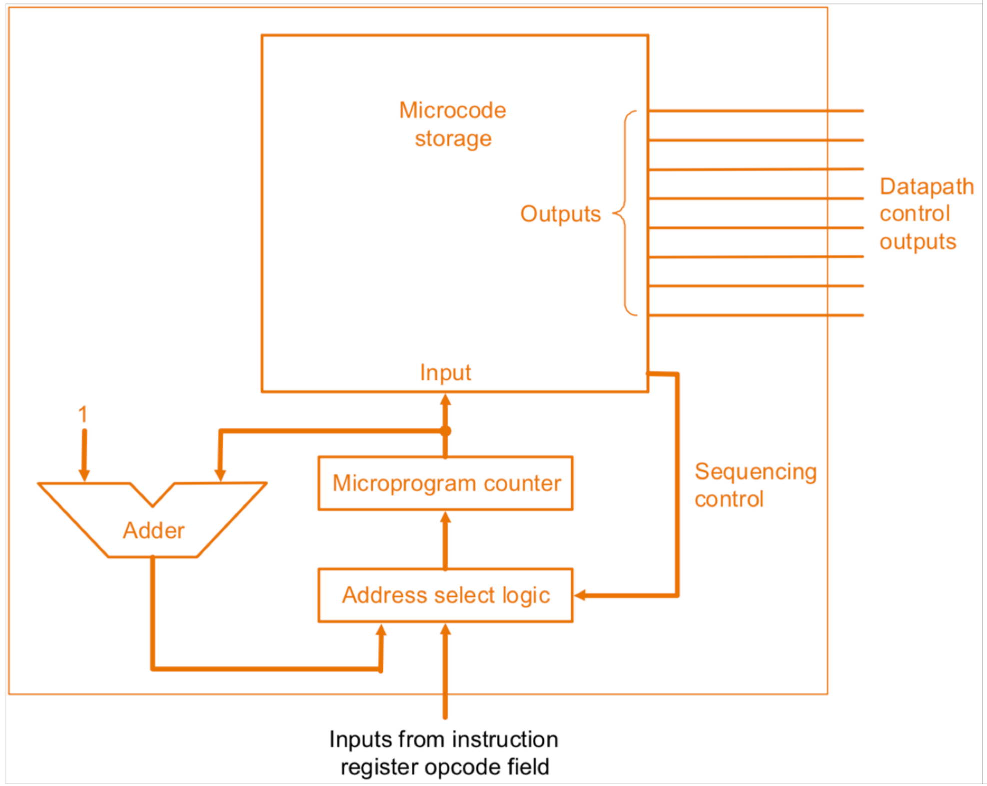
POSTECH

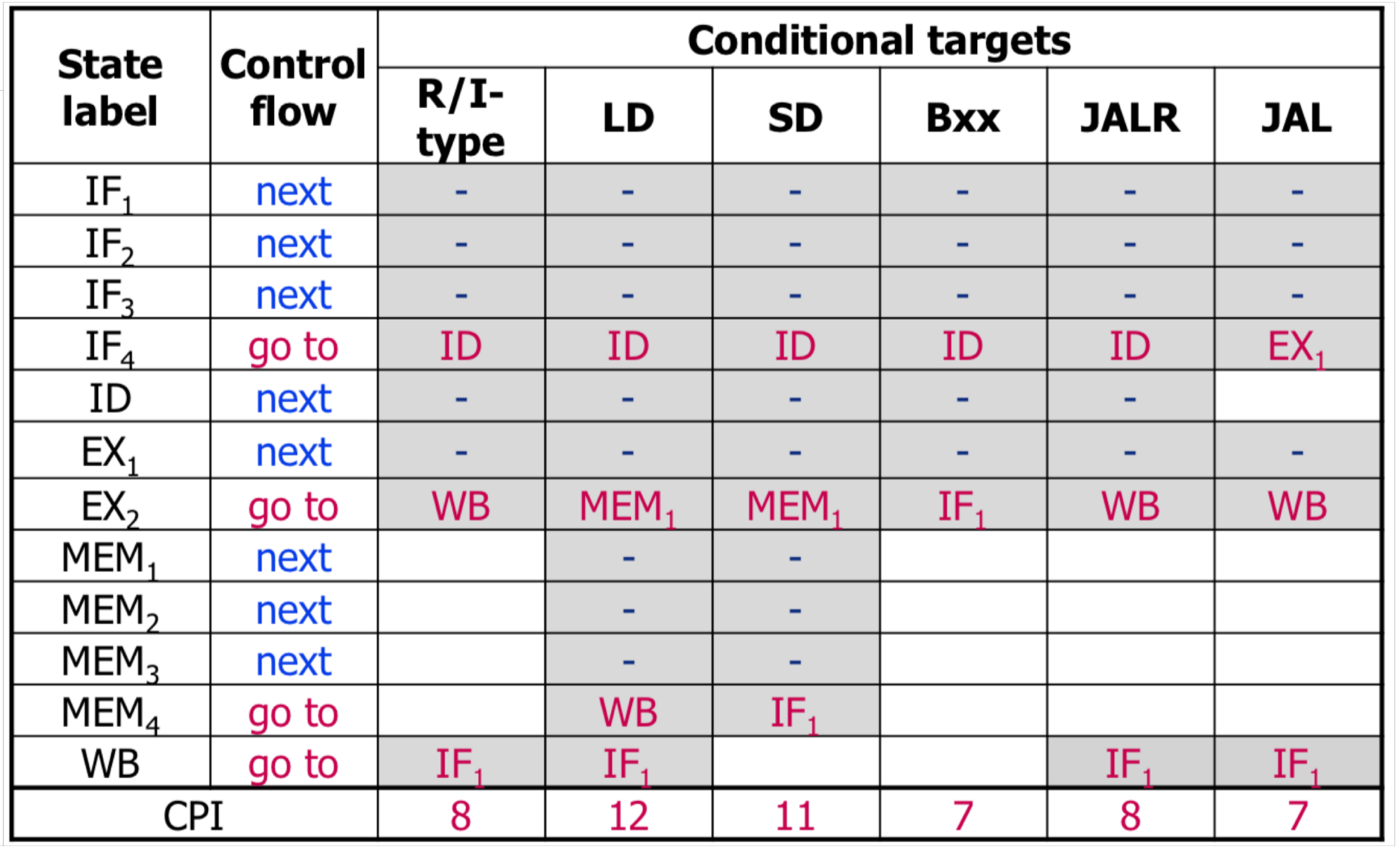
Multi-cycle CPU (Datapath)



POSTECH

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

POSTECH

Multi-cycle CPU

- I omitted the theoretical details about the multi-cycle CPU
- Please refer the lecture note (Multicycle CPU) and the textbook

- Let’s discuss the design & 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);
```

- Actually, it has a "clock abusing", which is usually forbidden
  - The purpose is to make a clock-synchronous memory, which responses in a single clock cycle
- As a result, you're not disturbed by the *inputReady* & *ackOutput* signal

Assignment

- Implement a multi-cycle CPU
  - Datapath unit for a 16-bit CPU
  - Full support for TSC Instruction Set, without RWD, ENI, DSI
  - *num_inst* should be increased when the CPU finish executing an instruction
  - The WWD instruction puts the register value to 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 FSM
  - The control unit gives right signals to the datapath unit

Assignment (cnt'd)

- Your implementation should pass all tests in the testbench
  - The message of "All Pass!" does not guarantee that 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**
  - There must be neither "delay(#)" nor "wait"
  - All storage units (registers, PC, etc.) must be updated only on clock 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 an ALU