浙江大学
ZHEJIANG UNIVERSITY

Computer Architecture Experiment

Topic 3. Pipelined CPU with stall

浙江大学计算机学院
陈文智、王总辉
{chenwz, zhwang}@zju.edu.cn

Outline

- Experiment Purpose
- Experiment Task
- Basic Principle
- Operating Procedures
- Precaution
- Checkpoints

Experiment Purpose

- Understand the principles of Pipelined CPU Stall
- Understand the principles of Data hazard
- Master the method of Pipelined CPU Stalls Detection and Stall the Pipeline.
- master methods of program verification of Pipelined CPU with Stall

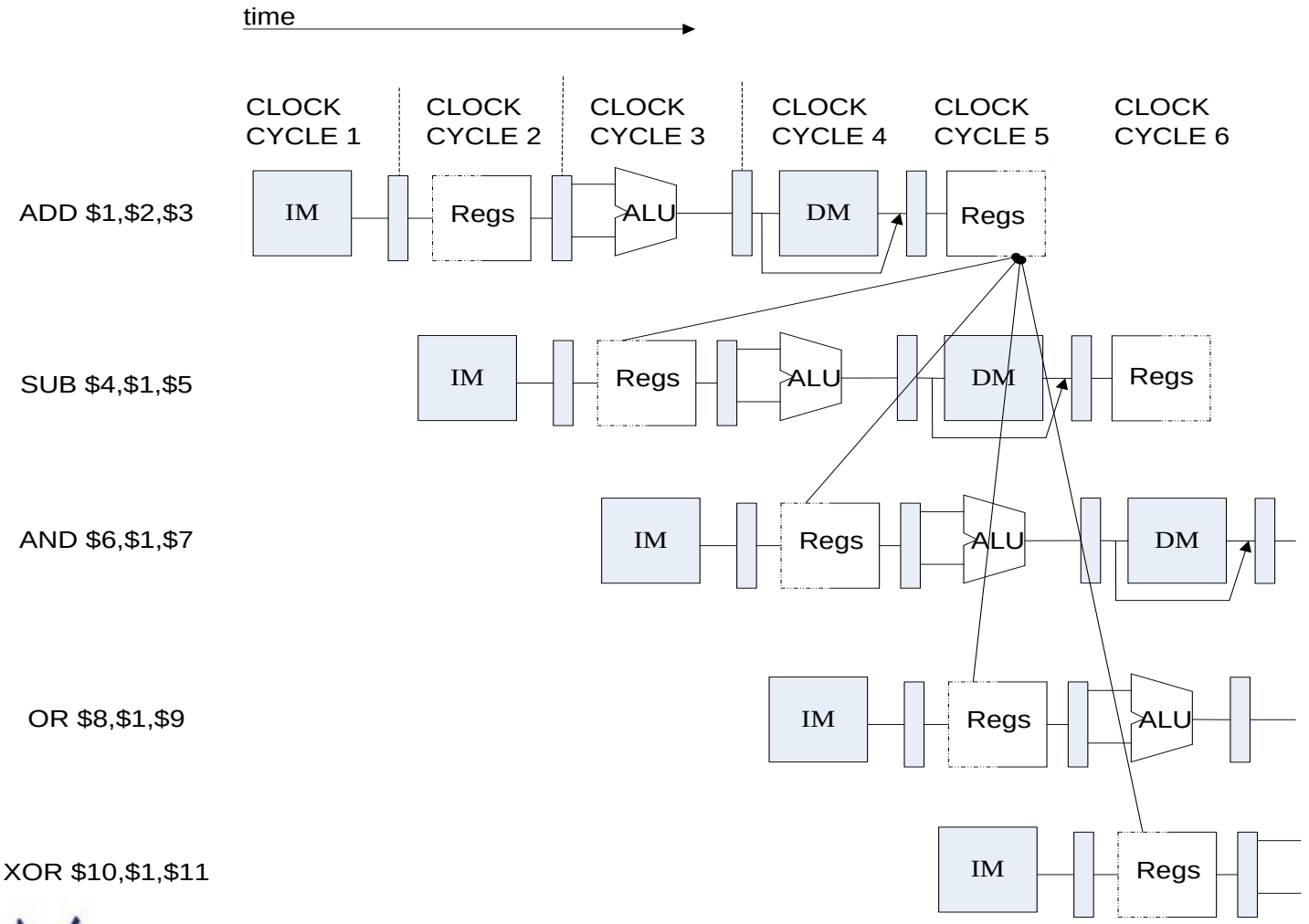
Experiment Task

- Design the Stall Part of Datapath of 5-stages Pipelined CPU
- Modify the CPU Controller, adding Condition Detection of Stall.
- Verify the Pp. CPU with program and observe the execution of program

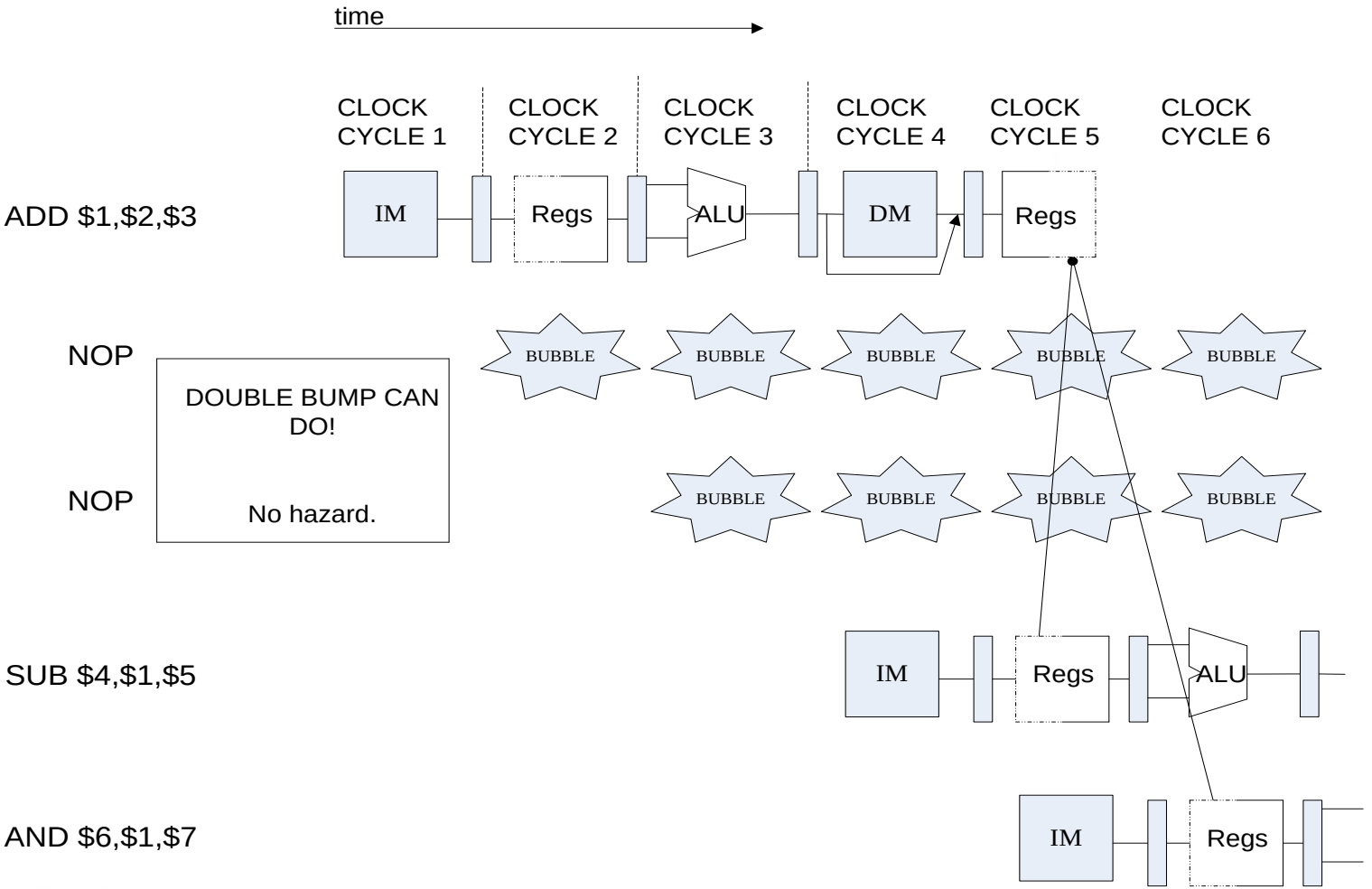
Data Hazard Definition

- **Data Hazards arise when an instruction depends on the results of a previous instruction in a way that is exposed by the overlapping of instructions in the pipeline.**
- **When inst. i executes before instr. j, there are data hazards:**
  - RAW, i.e. instr. j read a source before instr. i writes it.
  - WAW, i.e. instr. j write an operand before instr. i writes it.
  - WAR, i.e. instr. j write a destination before instr. i read it.

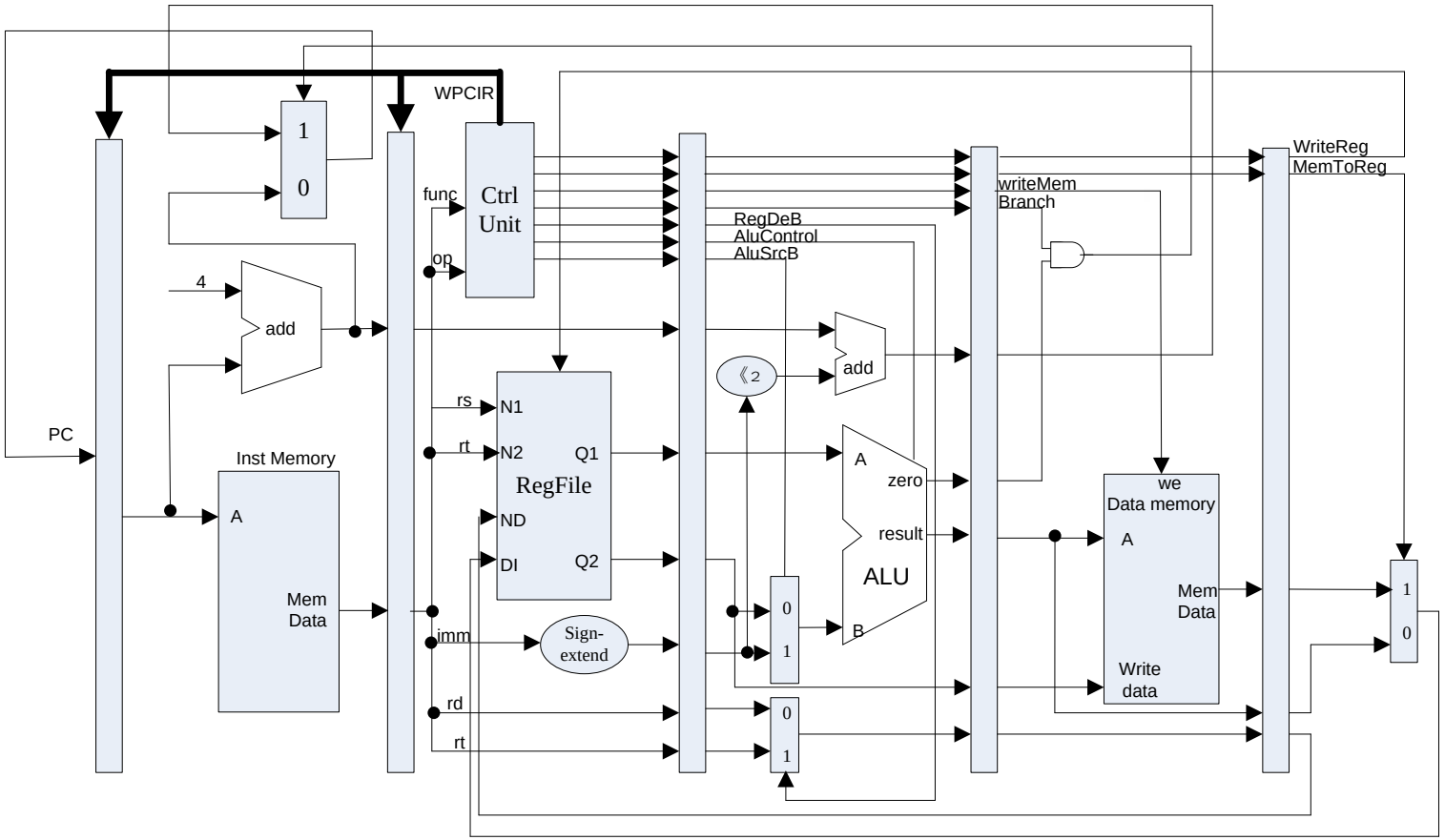
Instruction Demo



Data Hazard Causes Stalls



How to Stall the Pipeline



IF	ID	EXE	MEM	WB

In Which Conditions it Causes Stall

- assign AfromEx = (if_rs==rd) & (if_rs != 0) & (opcode ==`OP_ALUOp);
  //if instr.rs=id inst.rd and id instr = ALUOp

- Other expressions：
- BfromEx： if instr.rt=id instr.rd and id instr = ALUOp
- AfromMem： if instr.rs=ex instr.rd and ex instr = ALUOp
- BfromMem ： if instr.rt=ex instr.rd and ex instr = ALUOp
- AfromExLW ： if instr.rs=id instr.rt and id instr = LW
- BfromExLW ： if instr.rt=id instr.rt and id instr = LW
- AfromMemLW： if instr.rs=ex instr.rt and ex = LW
- BfromMemLW ： if instr.rt=ex instr.rt and ex = LW

- assign stall = AfromEx || …

Controller Module

- **if_instr:** if stage instruction
- **instr:** id stage instruction
- **ex_instr:** ex stage instruction
- **mem_instr:** mem stage instruction
- **wb_instr:** wb stage instruction
- **assign cu_wpcir = stall;**

Output signal cu_wpcir to ID Module，then ID Module Output id_wpcir to Top Module.

Pipelined CPU Top Module

- **module top (input wire CCLK, BTN3, BTN2, input wire [3:0]SW, output wire LED, LCDE, LCDRS, LCDRW, output wire [3:0]LCDDAT);**

```
assign pc [31:0] = if_npc[31:0];
if_stage x_if_stage(BTN3, rst, pc, mem_pc, mem_branch, id_wpcir, …
                    IF_ins_type, IF_ins_number,ID_ins_type,ID_ins_number);
id_stage x_id_stage(BTN3, rst, if_inst, if_pc4, wb_destR,…, id_wpcir,
                    ID_ins_type, ID_ins_number, EX_ins_type, EX_ins_number..);
ex_stage x_ex_stage(BTN3, id_imm, id_inA, id_inB, id_wreg, ..
                    EX_ins_type, EX_ins_number, MEM_ins_type, MEM_ins_number);
mem_stage x_mem_stage(BTN3, ex_destR, ex_inB, ex_aluR, …
                    MEM_ins_type, MEM_ins_number, WB_ins_type, WB_ins_number);
wb_stage x_wb_stage(BTN3, mem_destR, mem_aluR, …
                    WB_ins_type, WB_ins_number,OUT_ins_type, OUT_ins_number);
```

IF Module and ID Module

- **In IF Module， When it Causes Stalls**
  - PC will not be changed.
  - ID_ins_number will not be changed.
  - ID_ins_type will not be changed.
- **In ID Module， When it Causes Stalls**
  - reg_inst should be 0
  - pc4 will not be changed.
  - ID_ins_number will not be changed.
  - ID_ins_type = INST_TYPE_NONE

Observation Info

- **Input**
  - West Button: Step execute
  - South Button: Reset
  - 4 Slide Button: Register Index
- **Output**
  - 0-7 Character of First line: Instruction Code
  - 8 of First line : Space
  - 9-10 of First line : Clock Count
  - 11 of First line : Space
  - 12-15 of First line : Register Content
  - Second line : “stage name”/number/type
    - stage name:
    - 1-”f”, 2-”d”, 3-”e”, 4-”m”, 5-”w”

Program for verification

	Instruction	Bin Code	Address	Inst. Type
	lw r1, $20(r0)	0x8c01_0014	0	6
	lw r2, $21(r0)	0x8c02_0015	1	6
	add r3, r1, r2	0x0022_1820	2	1
	add r2,r0,r0	0x0000_1020	3	1
	sub r4, r1, r3	0x0023_2022	4	2
	and r5, r3, r4	0x0064_2824	5	3
	nor r6, r4, r5	0x0085_3027	6	5
	sw r6, $22(r0)	0xac06_0016	7	7
	beq r6,r7,-8	0x10c7_fff8	8	8

Checkpoints

- **CP 1:**

  Waveform Simulation of this Pipelined CPU CU

- **CP 2 (Optional):**

  Waveform Simulation of this Pipelined CPU with the verification program

- **CP 3:**

  FPGA Implementation of this Pipelined CPU with the verification program

Thanks!

浙江大学
ZheJiang University