浙江大学
ZHEJIANG UNIVERSITY

Computer Architecture Experiment

Topic 2. 5-stage CPU executing in pipeline

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

- ftp://10.214.50.74:1021
- 2014_arc_course/2014_arc_course，只读，用来下载课件
- 2014_arc_ug_up/2014_arc_ug_up，可读写，上传体系结构作业及报告，每个人都已建好目录

Outline

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

Experiment Purpose

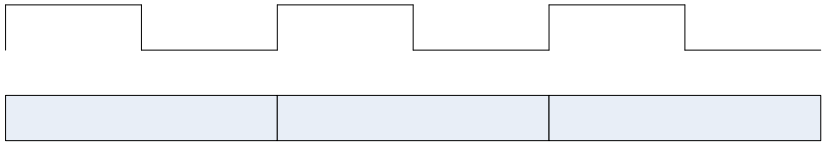
- Understand the principles of Pipelined CPU
- Understand the basic units of Pipelined CPU
- Understand the working flow of 5-stages
- Master the method of simple Pipelined CPU
- master methods of program verification of simple Pipelined CPU

Experiment Task

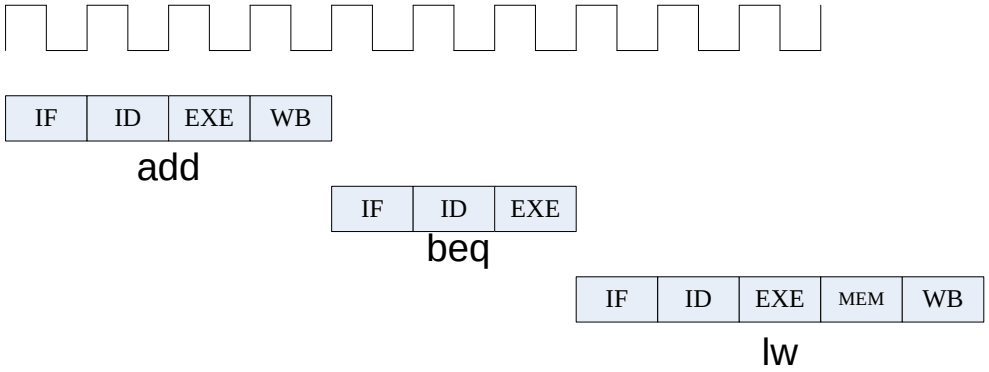
- **Design the CPU ALU**
- **Design the CPU Controller**
- **Based on the CPU Controller, Design the Datapath of 5-stages Pipelined CPU**
  - 5 Stages
  - Register File
  - Memory (Instruction and Data)
  - other basic units
- **Verify the Pp. CPU with program and observe the execution of program**

Comparison of three CPUs' work

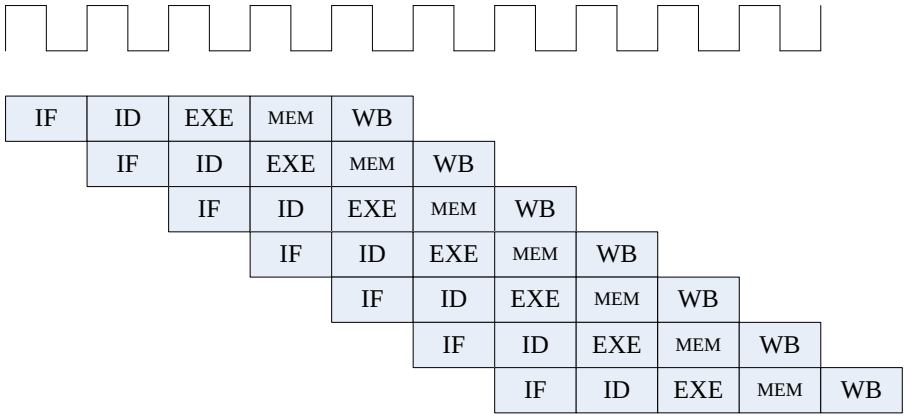
Simple-Cycle CPU



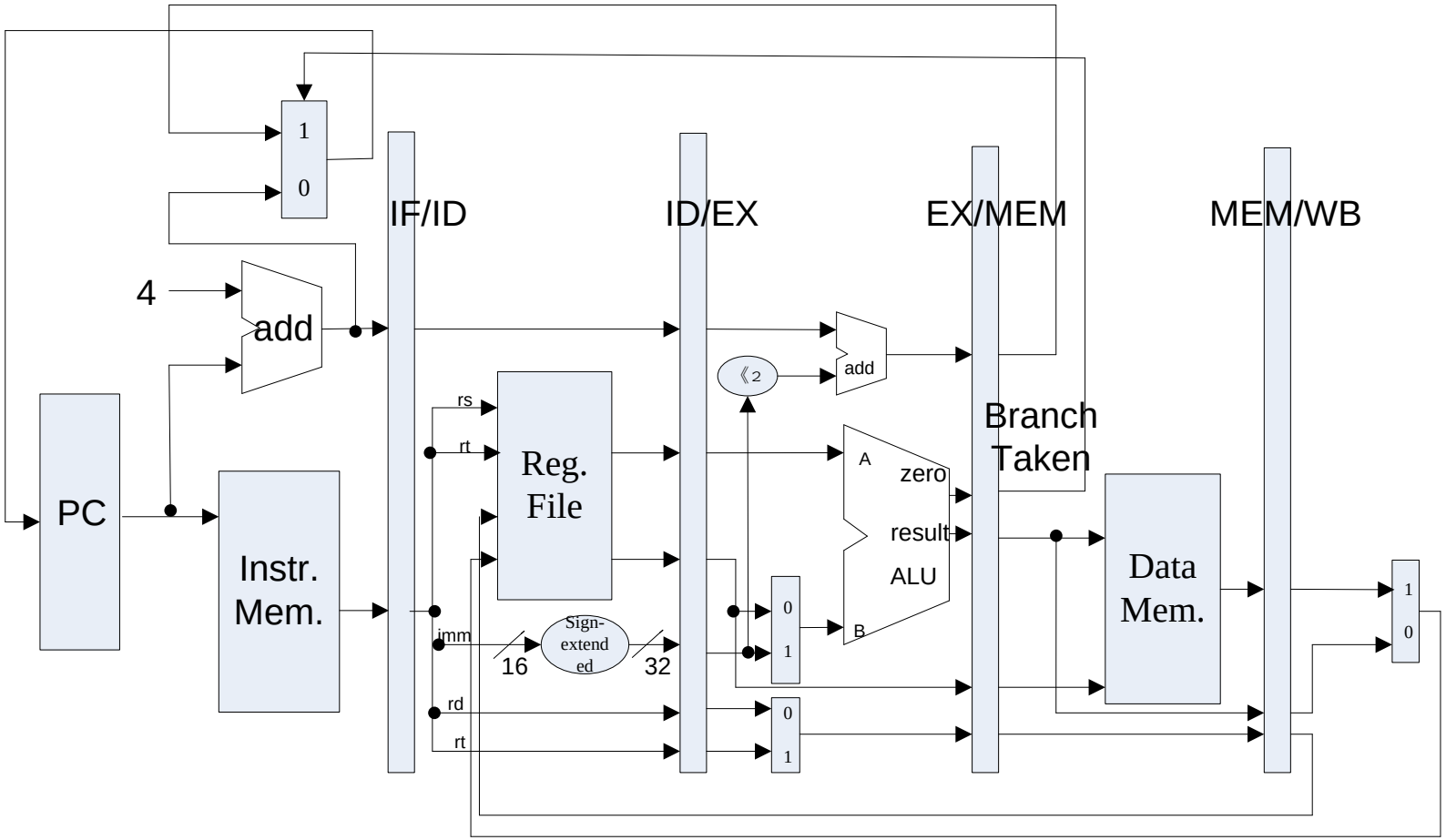
Multiple-Cycle CPU



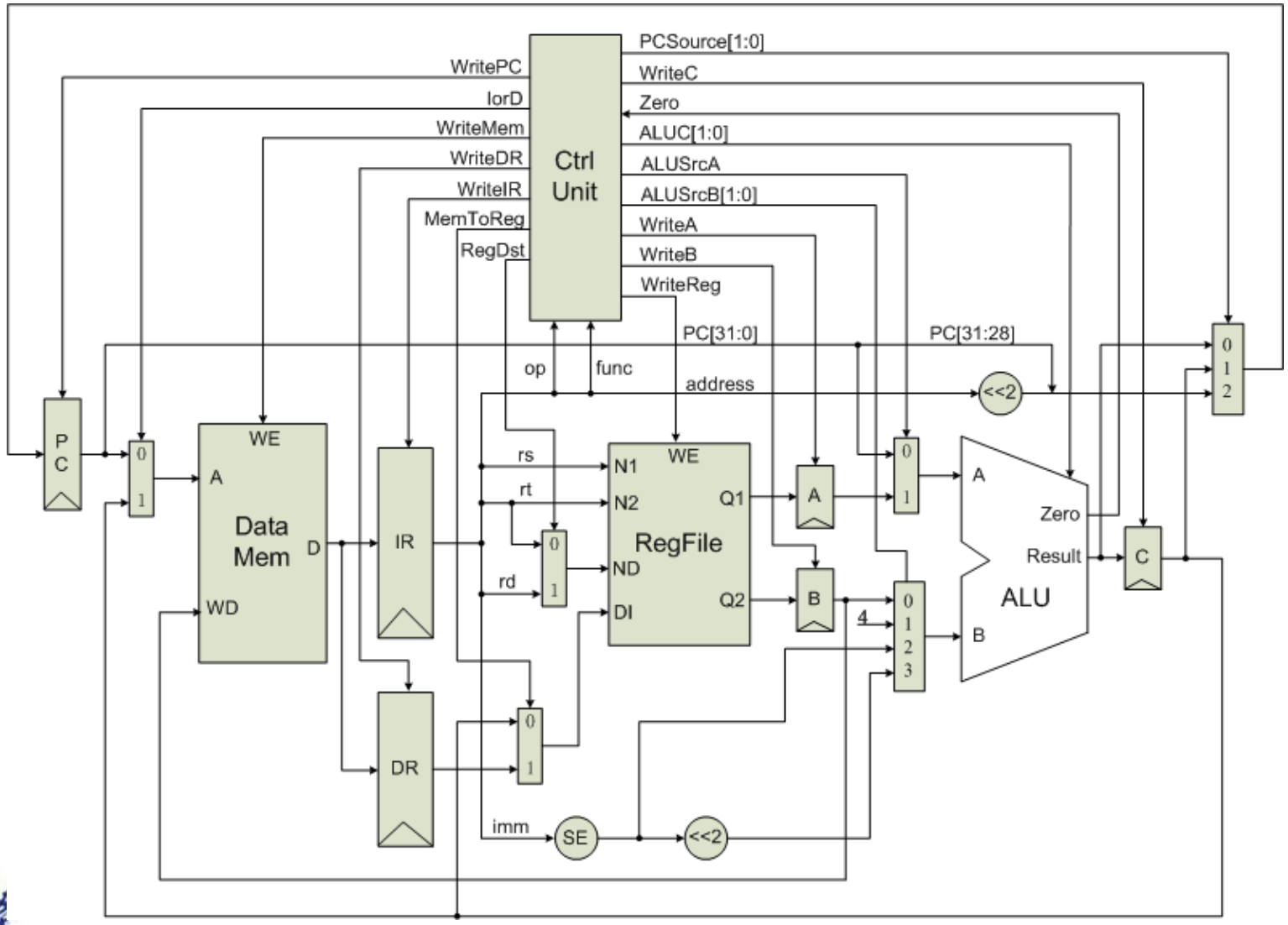
Pipelined CPU



Datapath of 5-stages Pipelined CPU



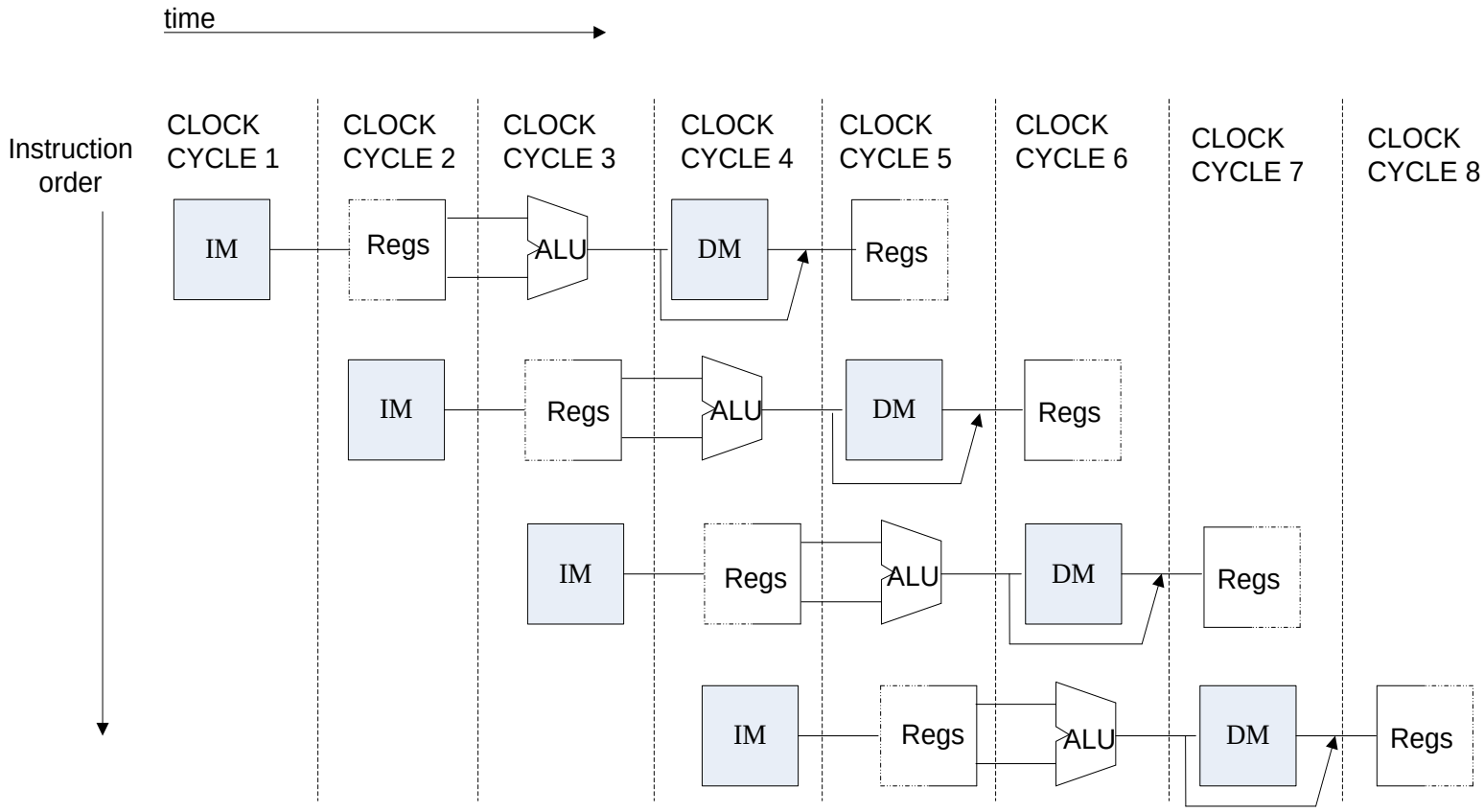
The principle of Multiple-cycle CPU



Structural hazards —resource conflicts

- **Structural hazards arise from resource conflicts when the hardware cannot support all possible combinations of instructions in simultaneous overlapped execution.**
  - Memory conflicts
  - Register File conflicts
  - Other units conflicts

How to resolve Structural hazards



Design of ALU and CU of Pipelined CPU

- **Design the ALU**
- **Design the Control Unit (CPU Controller)**
- **Note:**
  - Use all MIPS instructions defined in macro.vh file !

- **Our Ultimate Task in the Course:**
  - ✓ Design a Pipelined CPU support execution 31 MIPS Instructions (show in next table)

Output of CPU Controller

	Output Signal	Meaning When 1	Meaning When 0
1	Cu_branch	Branch Instr.	Non-Branch Instr.
2	Cu_shift	sa	Register data1
3	Cu_wmem	Write Mem.	Not Write Mem.
4	Cu_Mem2Reg	From Mem. To Reg	From ALUOut To Reg
5	Cu_sext	Sign-extend the imm.	No sign extended the imm.
6	Cu_aluc	ALU Operation	
7	Cu_aluimm	Imm.	Register data2
8	Cu_wreg	Write Reg.	Not Write Reg.
9	Cu_regrt	rt	rd

Pipelined CPU supporting execution of 31 MIPS instructions

MIPS Instructions								
Bit #	**31..26**	**25..21**	**20..16**	**15..11**	**10..6**	**5..0**	**Operations**	
R-type	op	rs	rt	rd	sa	func		
add		rs	rt	rd	00000	100000	rd = rs + rt; with overflow	PC+=4
addu		rs	rt	rd	00000	100001	rd = rs + rt; without overflow	PC+=4
sub		rs	rt	rd	00000	100010	rd = rs - rt; with overflow	PC+=4
subu		rs	rt	rd	00000	100011	rd = rs - rt; without overflow	PC+=4
and		rs	rt	rd	00000	100100	rd = rs & rt;	PC+=4
or		rs	rt	rd	00000	100101	rd = rs | rt;	PC+=4
xor		rs	rt	rd	00000	100110	rd = rs ^ rt;	PC+=4
nor		rs	rt	rd	00000	100111	rd = ~(rs | rt);	PC+=4
slt	000000	rs	rt	rd	00000	101010	if(rs < rt)rd = 1; else rd = 0; <(signed)	PC+=4
sltu		rs	rt	rd	00000	101011	if(rs < rt)rd = 1; else rd = 0; <(unsigned)	PC+=4
sll		00000	rt	rd	sa	000000	rd = rt << sa;	PC+=4
srl		00000	rt	rd	sa	000010	rd = rt >> sa (logical);	PC+=4
sra		00000	rt	rd	sa	000011	rd = rt >> sa (arithmetic);	PC+=4
sllv		rs	rt	rd	00000	000100	rd = rt << rs;	PC+=4
srlv		rs	rt	rd	00000	000110	rd = rt >> rs (logical);	PC+=4
srav		rs	rt	rd	00000	000111	rd = rt >> rs(arithmetic);	PC+=4
jr		rs	00000	00000	00000	001000		PC=rs

Pipelined CPU supporting execution of 31 MIPS instructions

MIPS Instructions								
Bit #	**31..26**	**25..21**	**20..16**	**15..11**	**10..6**	**5..0**	**Operations**	
I-type	op	rs	rt	immediate				
addi	001000	rs	rt	imm			rt = rs + (sign_extend)imm; with overflow	PC+=4
addiu	001001	rs	rt	imm			rt = rs + (sign_extend)imm;without overflow	PC+=4
andi	001100	rs	rt	imm			rt = rs & (zero_extend)imm;	PC+=4
ori	001101	rs	rt	imm			rt = rs | (zero_extend)imm;	PC+=4
xori	001110	rs	rt	imm			rt = rs ^ (zero_extend)imm;	PC+=4
lui	001111	00000	rt	imm			rt = imm * 65536;	PC+=4
lw	100011	rs	rt	imm			rt = memory[rs + (sign_extend)imm];	PC+=4
sw	101011	rs	rt	imm			memory[rs + (sign_extend)imm] <-- rt;	PC+=4
beq	000100	rs	rt	imm			if (rs == rt) PC+=4 + (sign_extend)imm <<2;	PC+=4
bne	000101	rs	rt	imm			if (rs != rt) PC+=4 + (sign_extend)imm <<2;	PC+=4
slti	001010	rs	rt	imm			if (rs < (sign_extend)imm) rt =1 else rt = 0; less than signed	PC+=4
sltiu	001011	rs	rt	imm			if (rs < (zero_extend)imm) rt =1 else rt = 0; less than unsigned	PC+=4
J-type	op	address						
j	000010	address					PC = (PC+4)[31..28],address<<2	
jal	000011	address					PC = (PC+4)[31..28],address<<2 ; $31 = PC+4	

Register File

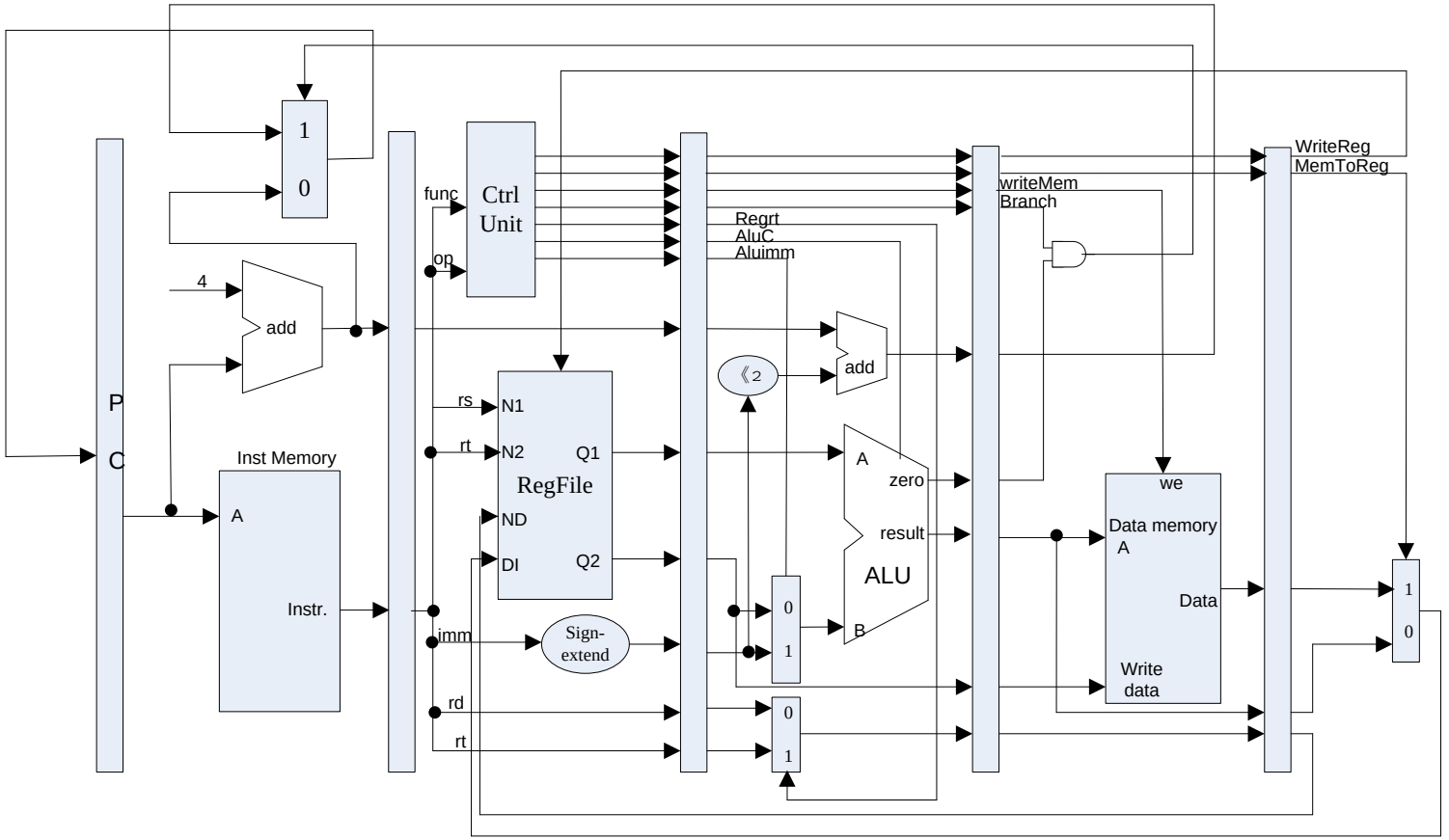
- **Register File**
  - Positive edge for transfer data for stages
  - Negative edge for write operation
  - Low level for read operation

Memory

- **Instruction Memory**
  - Single Port Block Memory
  - Read only, Width:32, Depth: 256
  - Falling Edge Triggered
- **Data Memory**
  - Single Port Block Memory
  - Read and write, Width:32, Depth: 256
  - Falling Edge Triggered

The principle of Pipelined CPU

—with CPU controller



IF	ID	EXE	MEM	WB

浙江大学 ZheJiang University

Units of Pipelined-cycle CPU

- **IF Stage (Instr. Mem.)**
- **ID Stage (CPU Ctl. And R.F.)**
- **EX Stage (ALU)**
- **Mem Stage (Data Mem.)**
- **WB Stage**

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, …
          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_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);
endmodule
```

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 r6, $21(r0)	0x8c06_0015	1	6
add r3,r0,r0	0x0000_1820	2	1
add r4,r0,r0	0x0000_2020	3	1
add r5,r0,r0	0x0000_2820	4	1
add r2,r2,r1	0x0041_1020	5	1
sub r3, r3, r1	0x0061_1822	6	2
and r4, r4, r1	0x0081_2024	7	3
nor r5, r5, r1	0x00a1_2827	8	5
beq r2, r1, -8	0x1041_fff8	9	8

Precaution

- **1. Add Anti-Jitter and display for “A-F”.**
- **2. Finish the blank.**
- **3. Debug method: Output whatever signal to LCD Display.**
- **4. Understand the principle of pipelined CPU and check the logic of circuit carefully, understand the sample code, then write code and synthesize the project, because it takes you a few minutes…**

Something Important ! ! !

- **The number and type are mean for the instruction which is to be executed in the stage.**
- **How to verify the result? Pls. check the result of WB stage for R-type and LW instructions, while check the result of EXEC stage for BEQ instruction.**
- **Why there are some NONE instructions following BEQ? How many NONE instructions? 3, because the condition of BEQ is generated in MEM stage.**
- **Why use the Reset button to initialize?**
- **Why we should pull the slide button after step execution to refresh the result? And instruction refresh is delayed by 1 clock-cycle? How to refresh automatically?**

Checkpoints

- **CP 1: Waveform Simulation of 5-stage Pipelined CPU ALU**
- **CP 2: Waveform Simulation of 5-stage Pipelined CPU CU**
- **CP 3 (Optional): Waveform Simulation of 5-stage Pipelined CPU with the verification program**
- **CP 4: FPGA Implementation of 5-stage Pipelined CPU with the verification program**

Thanks!

浙江大学
ZheJiang University