

# Computer Architecture Experiment

## Lab2



# Topics

- 0、Lab introduction
- Lab 1). Warmup Run you single-cycle CPU on 3E board. Try to implement 15 instructions.
- Lab 2). 5-stage pipelined CPU with 15 MIPS instructions (only required to execute in pipeline).
- Lab 3). Implementing "stall" when have hazards
- Lab 4). Implementing "forwarding paths"
- Lab 5). The whole CPU with 31 instructions.

# Outline

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

# Experiment Purpose 1

- 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 Controller, and the Datapath of 5-stages Pipelined CPU
  - 5 Stages
  - Register File
  - Memory (Instruction and Data)
  - other basic units
- Verify the Pipelined CPU with program and observe the execution of program

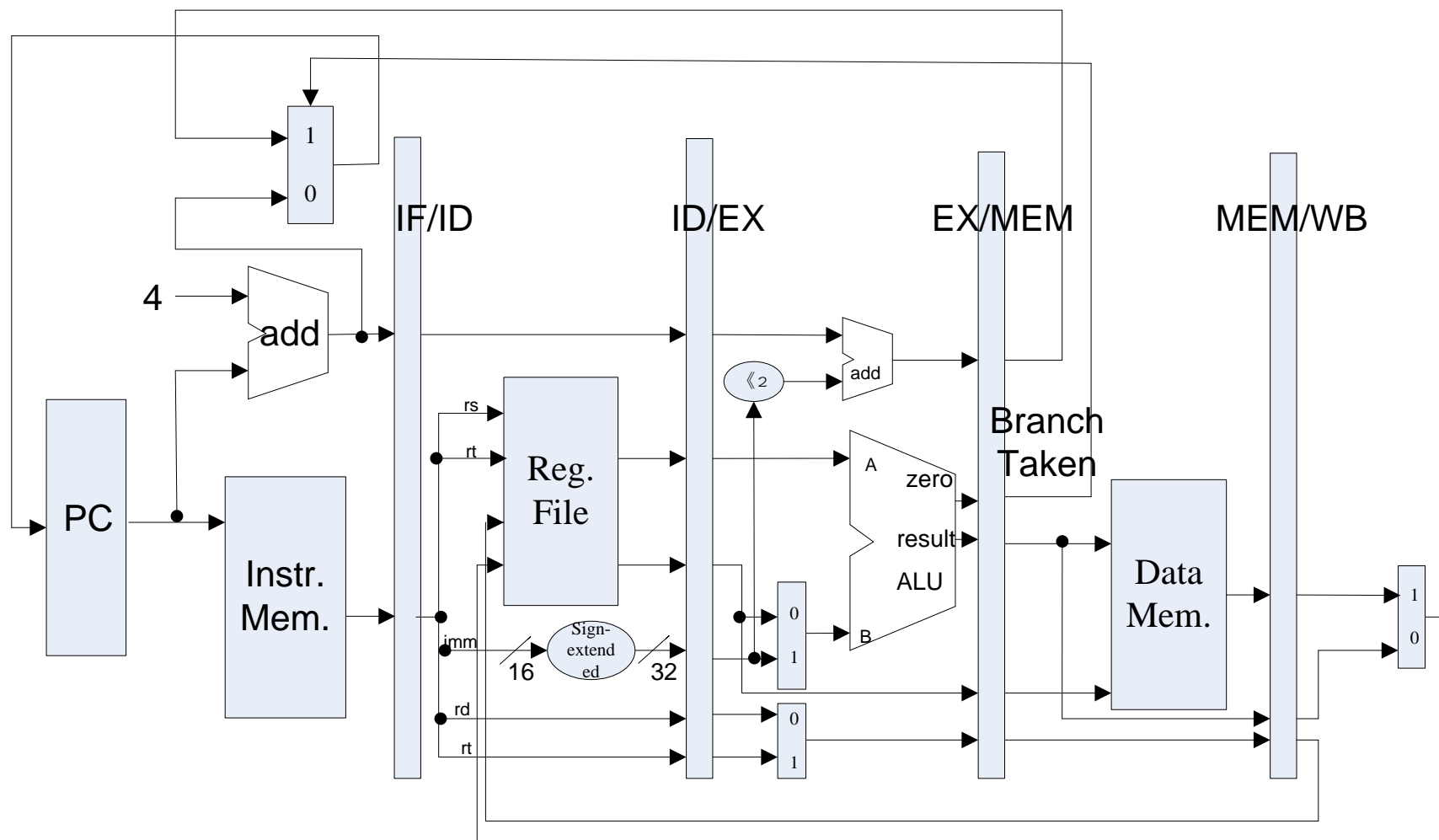
# 15 common used MIPS instructions

Instr No	MIPS Instructions								
	Bit #	[31..26]	[25..21]	[20..16]	[15..11]	[10..06]	[05..00]	Operations	
	R-type	op	rs	rt	rd	sa	func		
01	add	000000	rs	rt	rd	00000	100000	rd $\leftarrow$ rs + rt;	PC $\leftarrow$ PC + 4
02	sub	000000	rs	rt	rd	00000	100010	rd $\leftarrow$ rs - rt;	PC $\leftarrow$ PC + 4
03	and	000000	rs	rt	rd	00000	100100	rd $\leftarrow$ rs & rt;	PC $\leftarrow$ PC + 4
04	or	000000	rs	rt	rd	00000	100101	rd $\leftarrow$ rs   rt;	PC $\leftarrow$ PC + 4
05	sll	000000	00000	rt	rd	sa	000000	rd $\leftarrow$ rt << sa;	PC $\leftarrow$ PC + 4
06	srl	000000	00000	rt	rd	sa	000010	rd $\leftarrow$ rt >> sa (logical);	PC $\leftarrow$ PC + 4
07	sra	000000	00000	rt	rd	sa	000011	rd $\leftarrow$ rt >> sa (arithmetic);	PC $\leftarrow$ PC + 4
08	I-type	op	rs	rt	immediate				
	addi	001000	rs	rt	immediate			rt $\leftarrow$ rs + (sign_extend)immediate;	PC $\leftarrow$ PC + 4
	andi	001100	rs	rt	immediate			rt $\leftarrow$ rs & (zero_extend)immediate;	PC $\leftarrow$ PC + 4
09	ori	001101	rs	rt	immediate			rt $\leftarrow$ rs   (zero_extend)immediate;	PC $\leftarrow$ PC + 4
0A	lw	100011	rs	rt	immediate			rt $\leftarrow$ memory[rs + (sign_extend)immediate];	PC $\leftarrow$ PC + 4
0B	sw	101011	rs	rt	immediate			memory[rs + (sign_extend)immediate] $\leftarrow$ rt;	PC $\leftarrow$ PC + 4
0C	beq	000100	rs	rt	immediate			if (rs == rt) PC $\leftarrow$ PC + 4 + (sign_extend)immediate<<2; else	PC $\leftarrow$ PC + 4
0D	bne	000101	rs	rt	immediate			if (rs != rt) PC $\leftarrow$ PC + 4 + (sign_extend)immediate<<2; else	PC $\leftarrow$ PC + 4
0E	J-type	op	address						
	j	000010	address					PC $\leftarrow$ (PC+4)[31..28],address<<2	

# Precaution

- 1. Add Anti-Jitter
- 2. Finish the State Machine
- 3. Add Stage Status

# Datapath of 5-stages Pipelined CPU

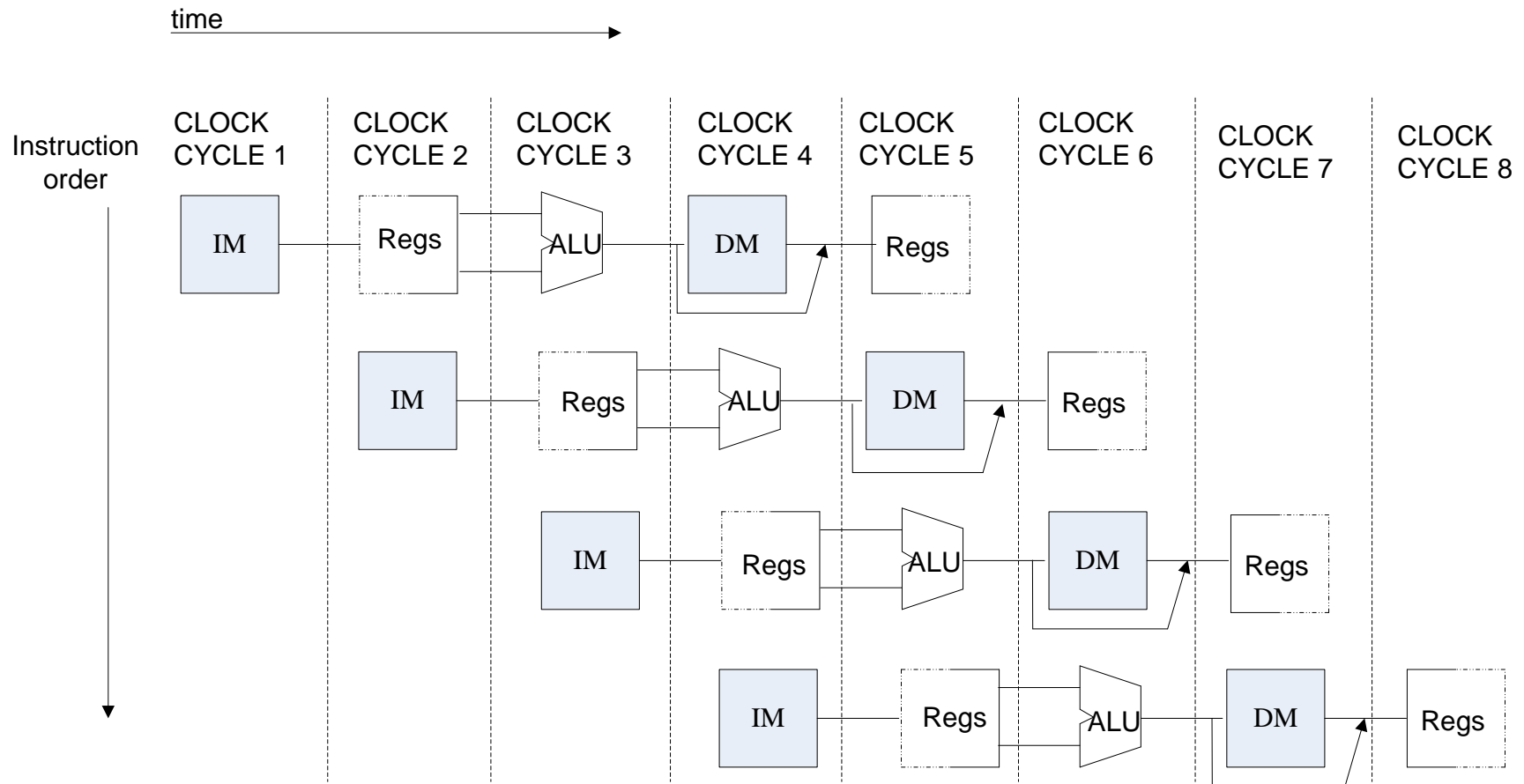




# Structural hazards —resource conflicts

- 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



# Register File

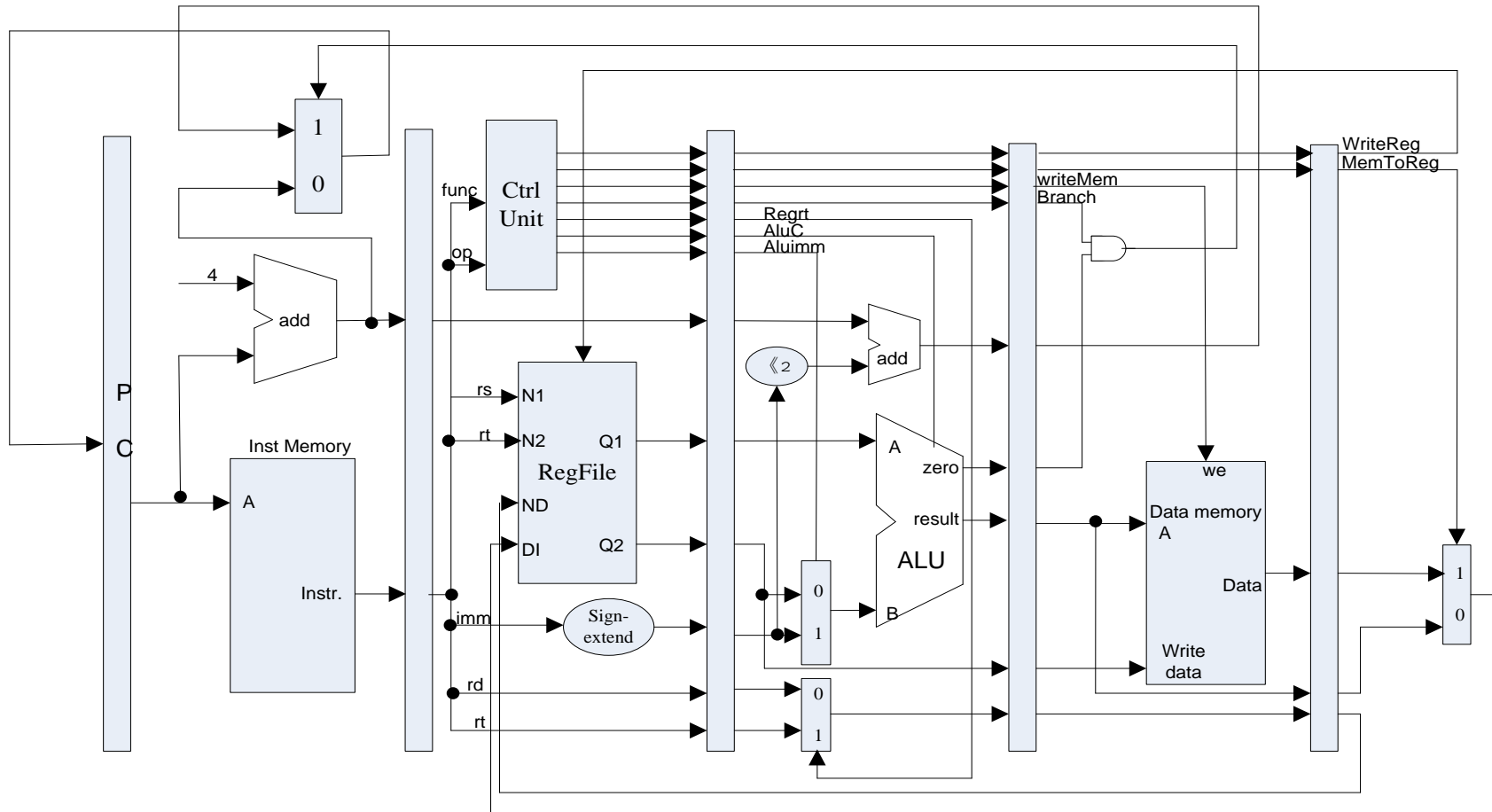
## ■ Register File

- 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
  - Falling Edge Triggered
- Data Memory
  - Single Port Block Memory
  - Read and write, Width:32
  - Falling Edge Triggered

# The principle of Pipelined CPU—with CPU controller



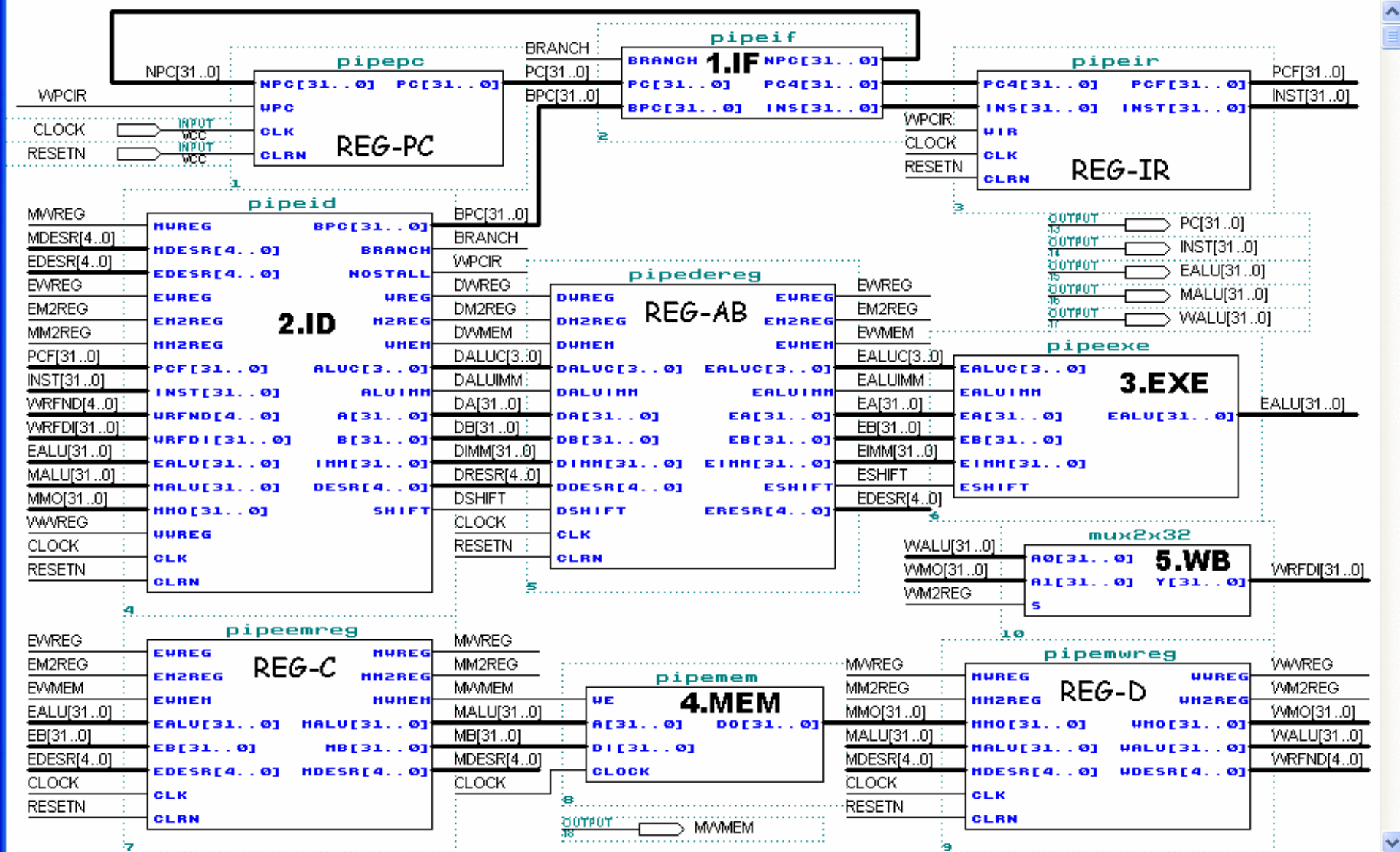
IF	ID	EXE	MEM	WB
----	----	-----	-----	----

# Output of CPU Controller(not limited)

	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

# 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);

# Observation Info

## ■ Input

- One Button: Step execute
- One Button: Reset
- One Button + 4 Switches : Register Index

## ■ Output (Led screen)

- 0-7 Character of First line: Instruction Code at ID-stage
- 8-15 of Character of First line : Register Value
- Second line : F D06 | E03 | M07 | W0A (06/03/07/0A: inst. No.)

# Prepare for checking

- Test code for /Lab2 can be downloaded from the material directory on course website.
- Do understand what you have implemented.
  - Prepare to answer questions on your results (Verilog code, logical graph, UCF...).

# Precaution

- 1. Add Anti-Jitter for buttons.
- 2. Modified your code based on your lab1 code.
- 3. Debug method: Output whatever signal to Led lamp or 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 ! ! !

- 1、 The number and type tells the information of the instruction that is to be executed in the stage.
- 2、 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.
- 3、 If you want to execute the test code and get the right result on your pre-pipelined CPU, how should you modify the test code ?
- 4、 Why the initial value of PC is FFFFFFFF, not 0?
- 5、 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?

■ Thanks!