Computer Organization & Design实验与课程设计

Lab04-3
CPU设计—指令集扩展

Ma De（马德）

made@zju.edu.cn

2022

College of Computer Science, Zhejiang University

Course Outline

- 一、实验目的
- 二、实验环境
- 三、实验目标及任务

实验目的

1. 运用寄存器传输控制技术
2. 掌握CPU的核心：指令执行过程与控制流关系
3. 设计数据通路和控制器
4. 设计测试程序

实验环境

- 实验设备

  1. 计算机（Intel Core i5以上，4GB内存以上）系统

  2. Sword 2.0/Sword4.0开发板

  3. Xilinx Vivado14.7及以上开发工具

- 材料

  无

实验目标及任务

- 目标：熟悉RISC-V RV32I的指令特点，了解控制器和数据通路的原理，扩展实验lab4-2 CPU指令集，设计并测试CPU
- 任务一：重新设计数据通路和控制器，在lab4-2的基础上完成
  - 兼容lab4-1、 lab4-2的数据通路和控制器
  - 替换lab4-1、 lab4-2的数据通路控制器核
  - 扩展不少于下列指令

  R-Type： add, sub, and, or, xor, slt, sltu,srl, sra, sll;

  I-Type： addi, andi, ori, xori, slti, sltiu,srli,srai,slli,lw,jalr;

  S-Type： sw;

  B-Type： beq,bne;

  J-Type： Jal;

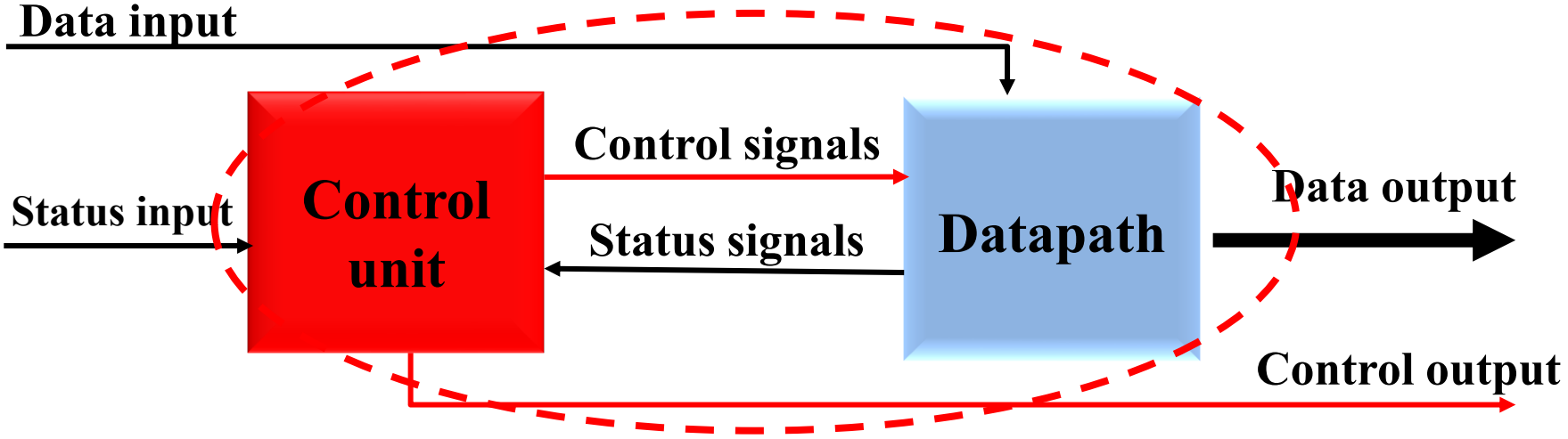
  U-Type： lui;
- 任务二：设计指令集测试方案并完成测试

RISC-V RV32I指令集扩展的原理介绍

CPU organization

Digital circuit

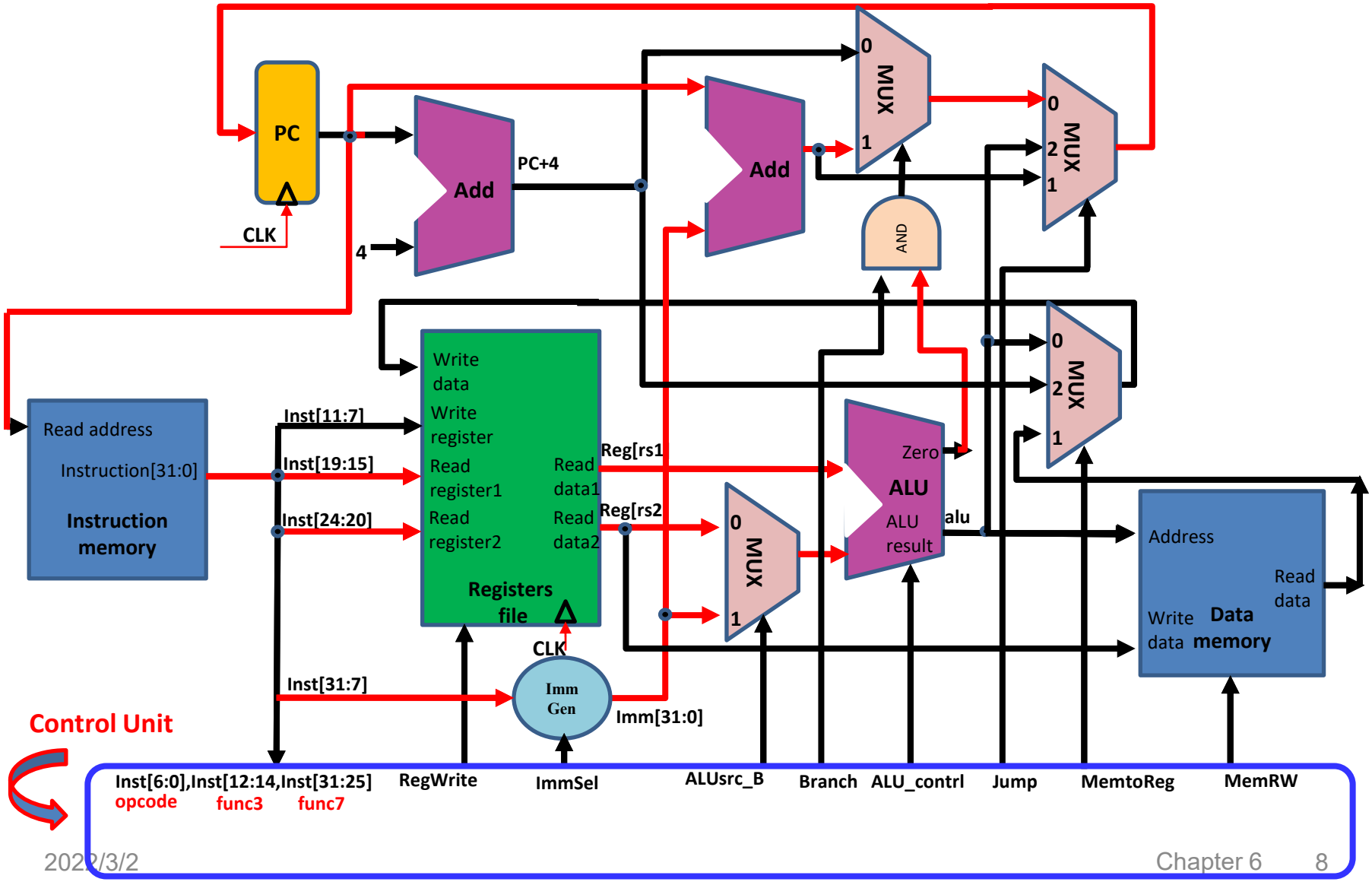
- General circuits that controls logical event with logical gates --Hardware



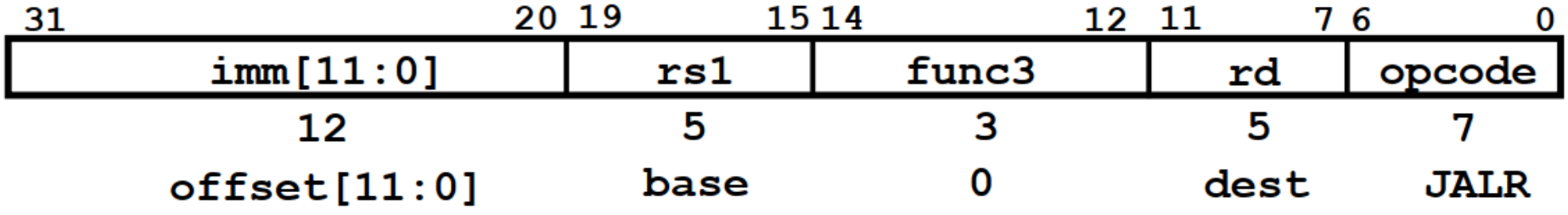
Computer organization

- Special circuits that processes logical action with instructions -Software

数据通路□ 兼容lab4-2需要增加哪些通路与操作控制



Adding JALR (I-Format)

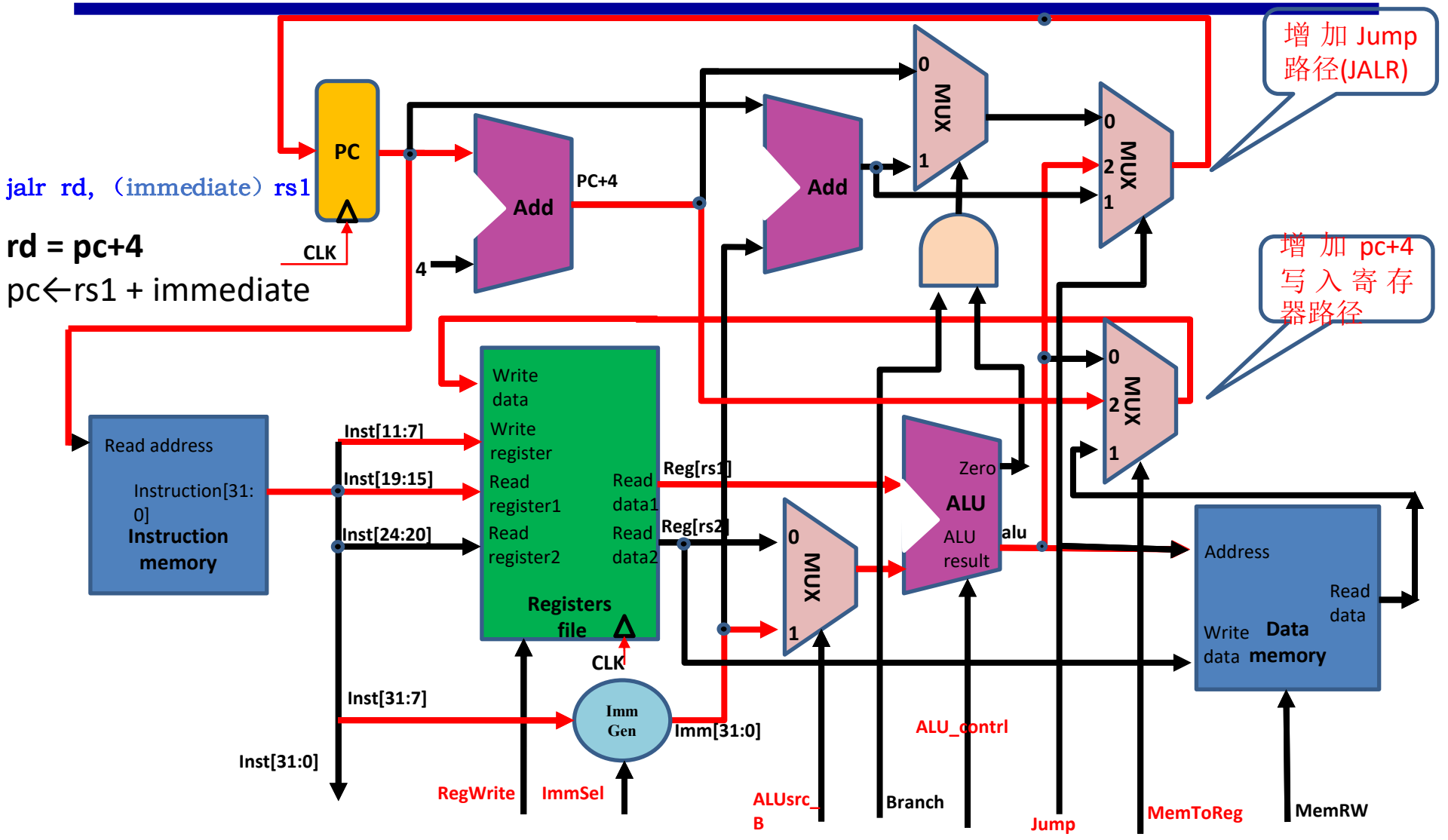


Imm[11:0]	Rs1(5位)	Func3(000)	rd(5位)	1100111（op）

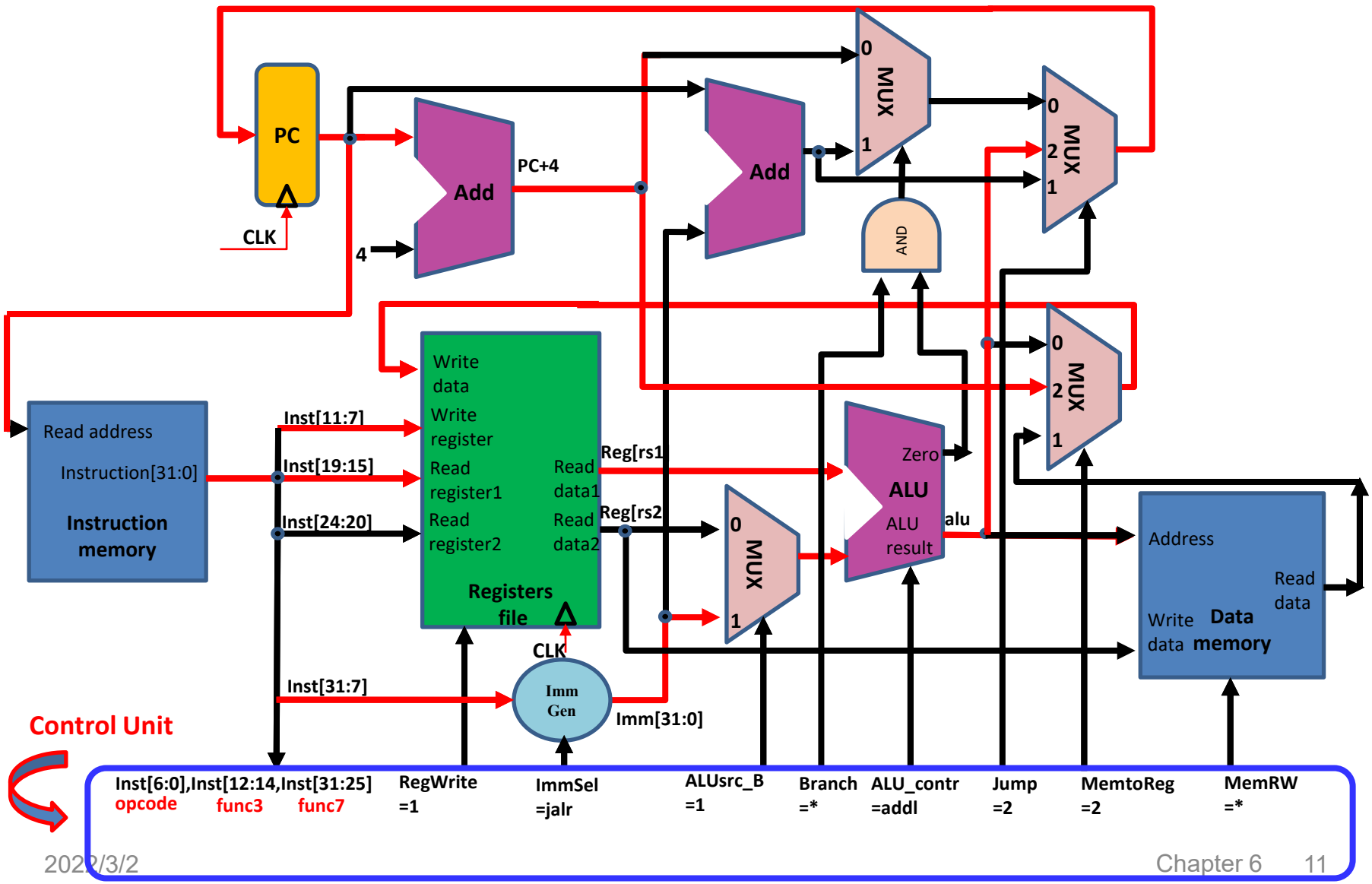
jalr rd,（immediate）rs1　　无条件跳转-链接（寄存器地址）

功能：rd = pc+4，pc←rs1 + immediate

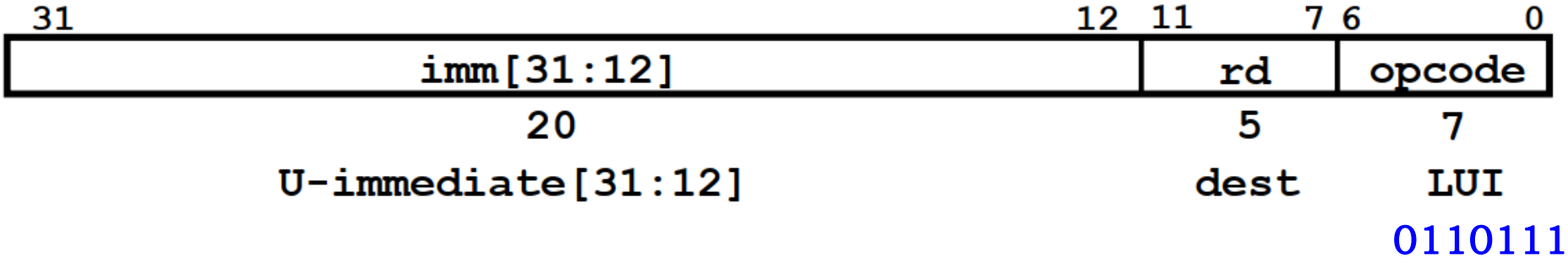
Adding JALR to Datapath



Control unit---jalr



Adding U-Format



lui rd, immediate

功能：rd = immediate <<12

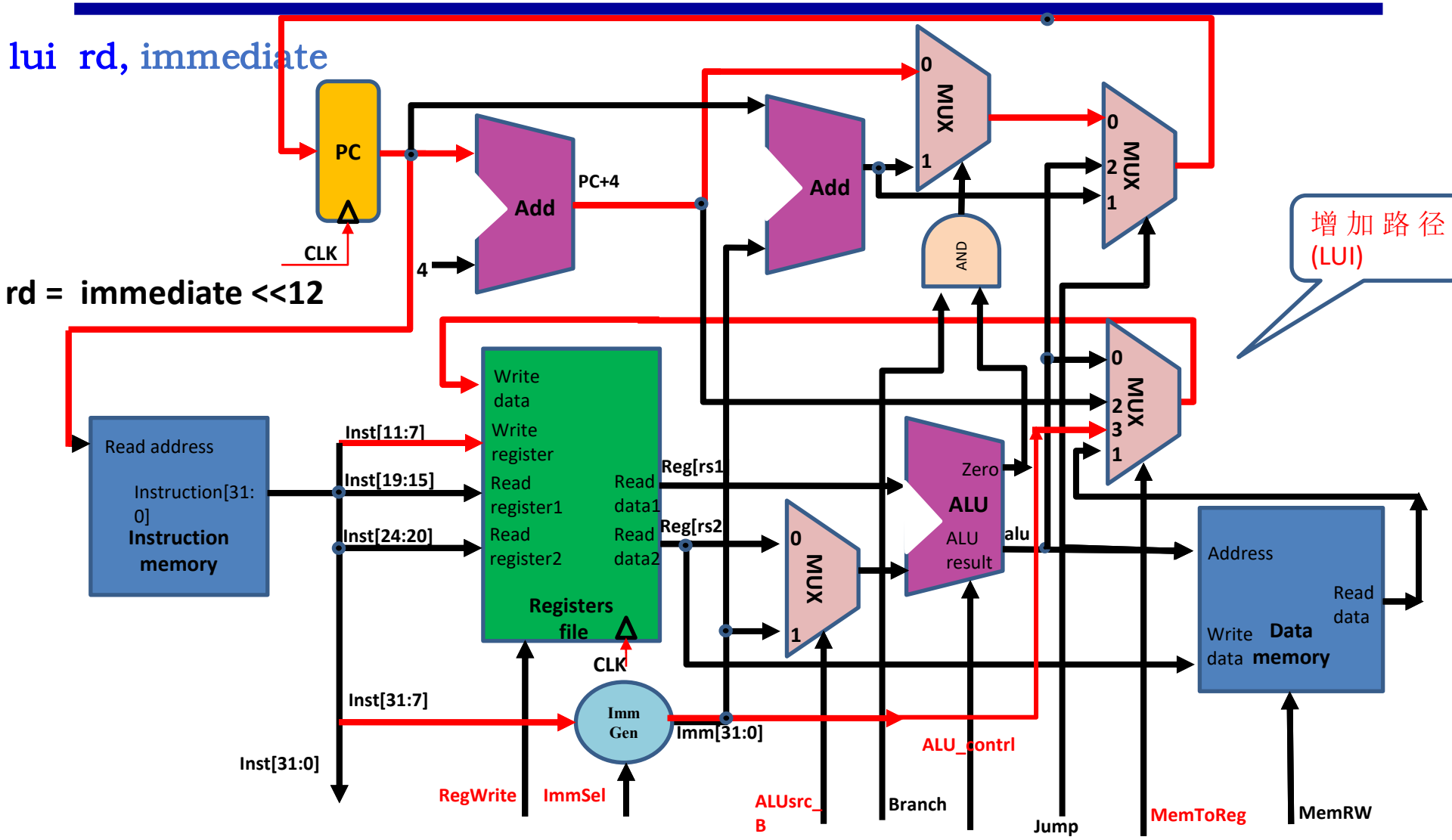
Eg: lui x5,0x12345

X5 = 0x12345000　　取左移12位的20位立即数

Adding LUI to Datapath

lui rd, immediate

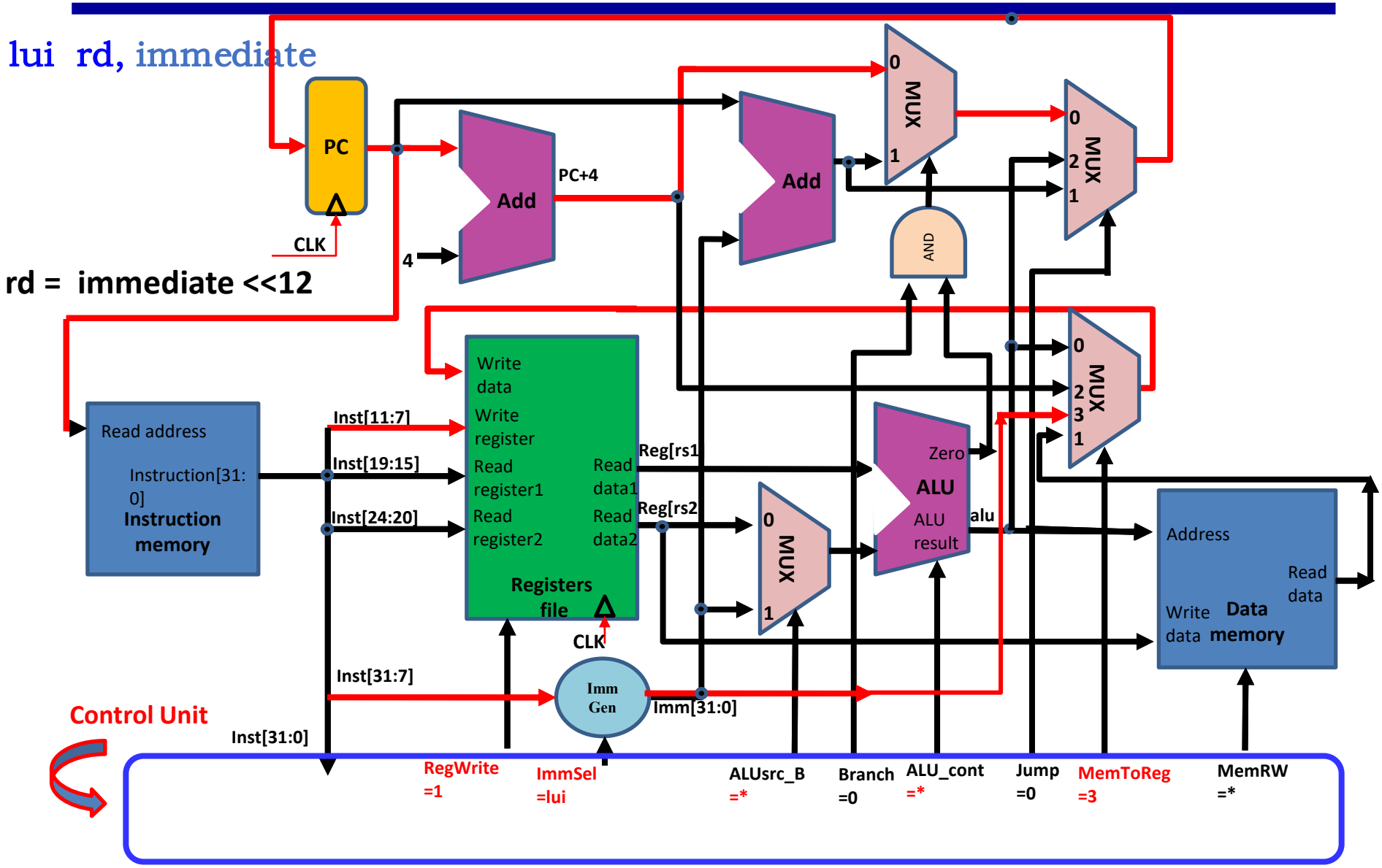
rd = immediate <<12



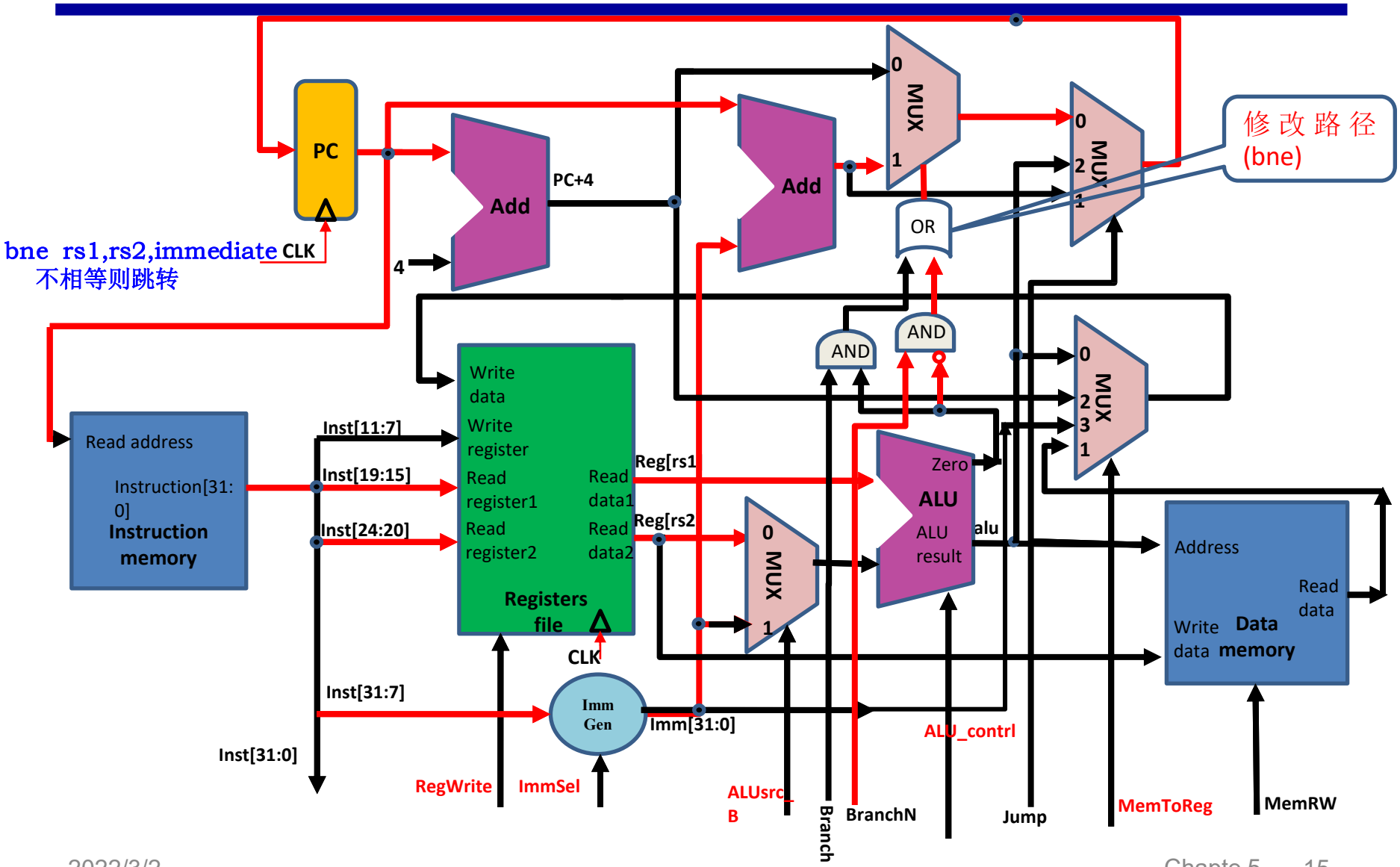
Control unit-- LUI

lui rd, immediate

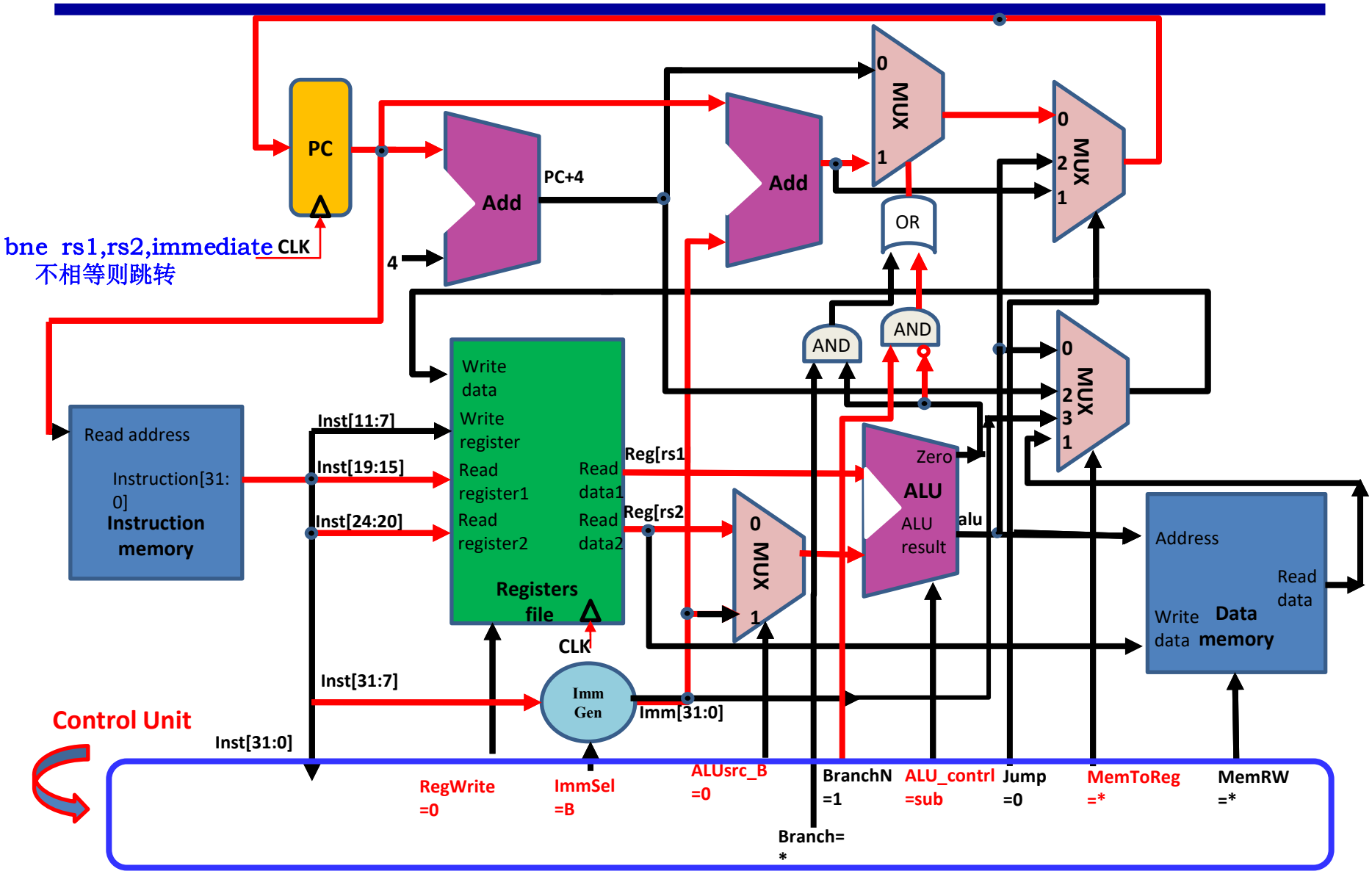
rd = immediate <<12



Adding bne to Datapath



Control unit-- bne



控制信号定义□ 兼容lab4-2需要增加哪些操作控制

信号	源数目	功能定义	赋值0时动作	赋值1时动作	赋值2时动作	赋值3时动作
ALUSrc_B	2	ALU端口B输入选择	选择源操作数寄存器2数据	选择32位立即数（符号扩展后）	-	-
MemToReg	4	寄存器写入数据选择	选择ALU输出	选择存储器数据	选择PC+4	选择imm
Branch	2	Beq指令目标地址选择	选择PC+4地址	选择转移目的地址PC+imm（zero=1）	-	-
BranchN	2	Bne指令目标地址选择	选择PC+4地址	选择转移目的地址PC+imm（zero=0）	-	-
Jump	3	J指令目标地址选择	由Branch决定输出	选择跳转目标地址PC+imm（JAL）	选择跳转目标地址ALU输出（JALR:rs1+imm）	-
RegWrite	-	寄存器写控制	禁止寄存器写	使能寄存器写	-	-
MemRW	-	存储器读写控制	存储器**读使能**，存储器写禁止	存储器**写使能**，存储器读禁止	-	-
ALU_Control	0000 - 1111	4位ALU操作控制	参考表ALU_Control			
ImmSel	000-111	3位立即数组合控制	参考表ImmSel			

控制信号真值表

- 根据数据通路重新设计控制器输出信号真值表

	ALU Src_B	Mem toReg	Reg Write	Mem RW	ImmSel	Branch	BranchN	Jump	ALUControl
R-格式									
I-格式									
S-格式									
B-格式									
J-格式									
U-格式									

重新设计真值表

需要增加控制信号吗？

控制信号真值表

Inst[31:0]	Banch	Branch N	Jump	ImmSel	ALUSrc_B	ALU_Control	MemRW	RegWrite	MemtoReg
add	0	0	0	*	Reg (0)	Add	Read	1	ALU(0)
sub	0	0	0	*	Reg (0)	Sub	Read	1	ALU(0)
(R-R Op)	0	0	0	*	Reg (0)	*(Op)*	Read	1	ALU(0)
addi	0	0	0	I	Imm (1)	Add	Read	1	ALU(0)
lw	0	0	0	I	Imm (1)	Add	Read	1	Mem(1)
sw	0	0	0	S	Imm (1)	Add	Write	0	*
beq	0	0	0	B	Reg (0)	sub	*	0	*
beq	1	*	0	B	Reg (0)	sub	*	0	*
bne	0	0	0	B	Reg (0)	sub	*	0	*
bne	*	1	0	B	Reg (0)	sub	*	0	*
jalr	*	*	2	I	Imm (1)	Add	*	1	PC+4(2)
jal	*	*	1	J	Imm (1)	*	*	1	PC+4(2)
lui	0	0	0	U	*	*	*	1	Imm(3)

ImmSel

Instruction type	Instruction opcode[6:0]	Instruction operation	(sign-extend)immediate	Imm Sel
I-type	0000011	Lw;lbu;lh;lb;lhu	(sign-extend) instr[31:20]	001
	0010011	Addi;slti;sltiu;xori;ori;andi;slli;srai		
	1100111	jalr		
S-type	0100011	Sw;sb;sh	(sign-extend) instr[31:25],[11:7]	010
B-type	1100011	Beq;bne;blt;bge;bltu;bgeu	(sign-extend) instr[31],[7],[30:25],[11:8], 1'b0	011
J-type	1101111	jal	(sign-extend) instr[31],[19:12],[20],[30:21],1'b0	100
U-type	0010111	auipc	instr[31:12],12'h000	000
	0110111	lui		

RV32I---decode

Instruction opcode	op	Instruction operation	Funct 3	Funct7	ALUop	Desired ALU action	ALUControl
R-type	0110011	add	000	0000000	10	add	0010
		sub	000	0100000		sub	0110
		sll	001	0000000		sll	1110
		slt	**010**	**0000000**		slt	0111
		sltu	**011**	**0000000**		sltu	1001
		xor	**100**	**0000000**		xor	1100
		srl	**101**	**0000000**		srl	1101
		sra	**101**	**0100000**		sra	1111
		or	**110**	**0000000**		or	0001
		and	**111**	**0000000**		and	0000

RV32I---decode

Instruction opcode	op	Instruction operation	Funct 3	Funct7	ALUop	Desired ALU action	ALUControl
S-Type	0100011	sb	000	-	00	add	0010
		sh	001	-		add	0010
		sw	010	-		add	0010

Instruction opcode	op	Instruction operation	Funct 3	Funct7	ALUop	Desired ALU action	ALUControl
B-Type	1100011	Beq	000	-	01	sub	0110
		Bne	001	-		sub	0110
		Blt	100	-		slt	0111
		Bge	101	-		slt	0111
		Bltu	110	-		sltu	1001
		bgeu	111	-		sltu	1001

RV32I---decode

Instruction opcode	op	Instruction operation	Funct 3	Funct7	ALUop	Desired ALU action	ALUControl
U-Type	0110111	lui	-	-	-	-	-
	0010111	auipc	-	-		-	-
Instruction opcode	op	Instruction operation	Funct 3	Funct7	ALUop	Desired ALU action	ALUControl
J-Type	1101111	jal	-	-	-	-	-

RV32I---decode

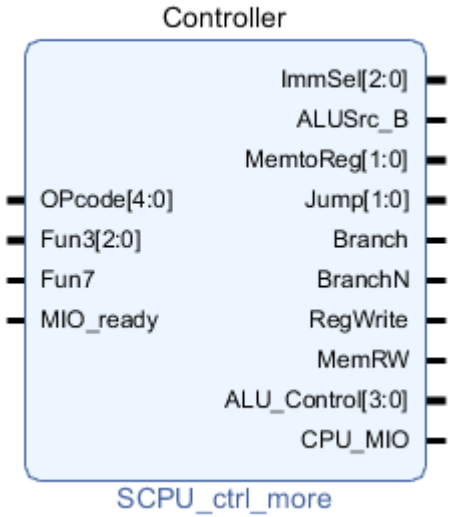
Instruction opcode	op	Instruction operation	Funct 3	Funct7	ALUop	Desired ALU action	ALUControl
I-Type	0000011	Lb	-	-	00	add	0010
		Lh	-	-		add	0010
		Lw	-	-		add	0010
		Lbu	-	-		add	0010
		lhu	-	-		add	0010
	1100111	jalr			00	add	0010

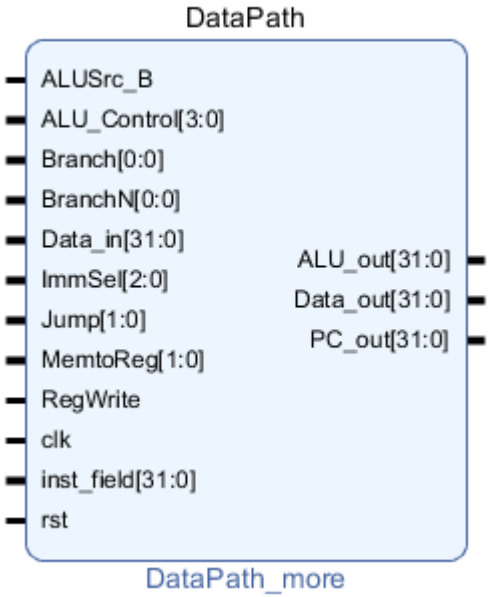
RV32I---decode

Instruction opcode	op	Instruction operation	Funct 3	Funct7	ALUop	Desired ALU action	ALUControl
I-Type	0010011	addi	000	-	11	add	0010
		slti	010	-		slt	0111
		sltiu	011	-		sltu	1001
		xori	100	-		xor	1100
		ori	110	-		or	0001
		andi	111	-		and	0000
		slli	001	0000000		sll	1110
		srli	101	0000000		srl	1101
		srai	101	0100000		sra	1111

重新设计数据通路与控制器接口：

- 重新设计接口
  - 扩展后增加了控制信号
  - 数据通路参考接口如右图
  - 控制器参考接口信号如下图





数据通路功能控制器接口信号标准

```
module    SCPU_ctrl_more( input[6:0]OPcode,        //OPcode
                          input[2:0]Fun3,          //Function
                          input     Fun7,          //Function
                          input MIO_ready,         //CPU Wait
                          output reg [2:0]ImmSel,
                          output reg ALUSrc_B,
                          output reg [1:0]MemtoReg,
                          output reg [1:0]Jump,
                          output reg Branch,
                          output reg BranchN,
                          output reg RegWrite,
                          output reg MemRW,
                          output reg [3:0]ALU_Control,
                          output reg CPU_MIO
                          );
endmodule
```

数据通路功能控制器接口信号标准

```
module      Data_path_more( input clk,              //寄存器时钟
                            input rst,              //寄存器复位
                            input[31:0]inst_field,  //指令数据域[31:7]
                            input ALUSrc_B,         //ALU端口B输入选择
                            input [1:0]MemtoReg,    //Regs写入数据源控制
                            input [1:0]Jump,        //J指令
                            input Branch,           //Beq指令
                            input BranchN,          //Bne指令
                            input RegWrite,         //寄存器写信号
                            input[31:0]Data_in,     //存储器输入
                            input[3:0]ALU_Control,  //ALU操作控制
                            input[2:0]ImmSel,       //ImmGen操作控制

                            output[31:0]ALU_out,    //ALU运算输出
                            output[31:0]Data_out,   //CPU数据输出
                            output[31:0]PC_out      //PC指针输出
                            );
endmodule
```

- 任务一：重新设计数据通路和控制器，在Exp4-2的基础上完成
  - 兼容Exp4-1、 Exp4-2的数据通路和控制器
  - 替换Exp4-1、 Exp4-2的数据通路控制器核

设计工程：OExp04-ExtSCPU

◎扩展不少于下列指令

R-Type： add, sub, and, or, xor, slt, sltu,srl, sra, sll；

I-Type： addi, andi, ori, xori, slti, sltiu,srli,srai,slli,lw,jalr;

S-Type： sw；

B-Type： beq,bne；

J-Type： Jal；

U-Type： lui；

◎集成替换验证通过的新CPU

- 替换 (Exp4-2)中的SCPU模块
- 替换 (Exp4-2)中的SCPU_ctrl模块
- 替换 (Exp4-2)中的Data_Path模块
- 顶层模块延用Exp04
  - 模块名：ExtSCPU.v

◎测试扩展后的CPU功能

- 设计测试程序(RISCV汇编)测试

设计要点

- **设计指令扩展后DataPath结构**
  - 在实验4-1的基础上扩展
- **根据新DataPath结构设计控制器**
  - 建议用HDL结构化描述
- **设计CPU调用模块**
  - 根据新的控制器和数据通路接口信号设计CPU模块
- **仿真新设计的模块**
  - 独立仿真DataPath和控制器
- **集成替换CPU及子模块**
  - 仿真正确后
    - 集成替换CPU、数据通路和控制器模块

设计要点

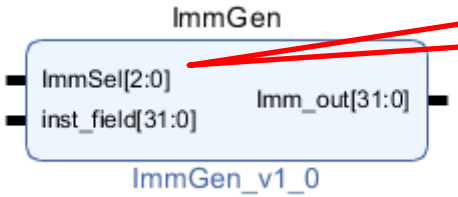
◎ 设计指令扩展后的ALU

- 在实验四的原理图上扩展

控制信号增加，内部实现了sra,sll等R型扩展指令的算数单元

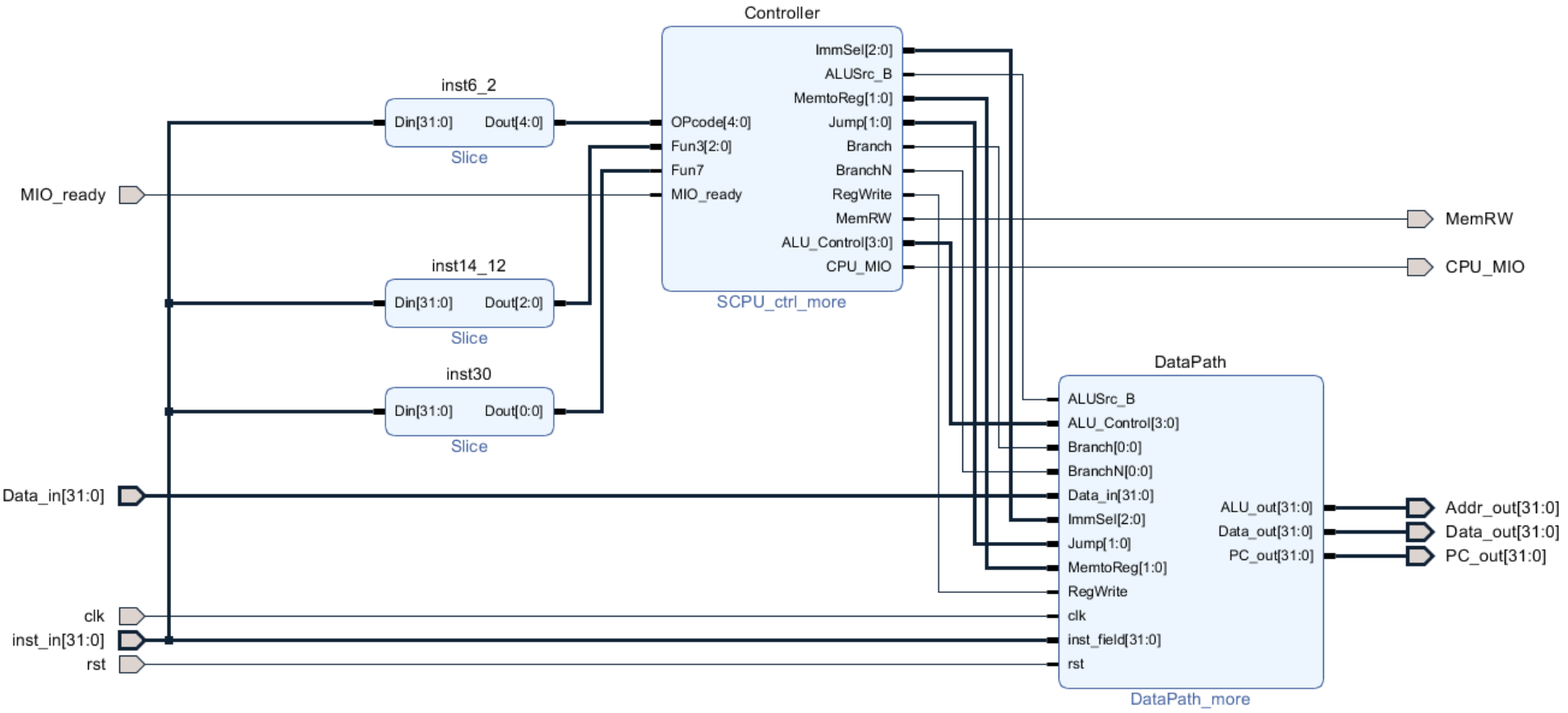
◎ 设计指令扩展后的ImmGen

- 在实验五的结构描述上扩展

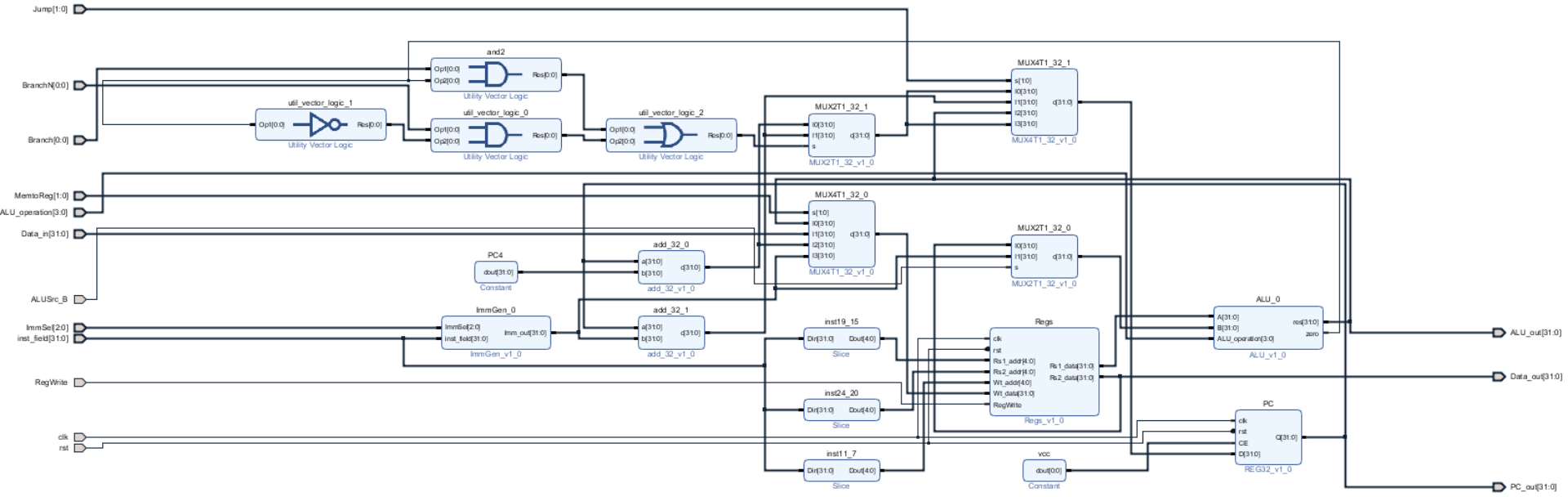


控制信号增加，内部实现了U型扩展指令的立即数生成

扩展指令后的CPU参考模块



Datapath参考设计



控制器描述参考结构

```
`define CPU_ctrl_signals {ALUSrc_B,MemtoReg,RegWrite,MemRW,Branch,Jump, ALU_Control
,  ……}
        always @* begin
                case(OPcode)
                5’b01100:                                                    //ALU
                  case(Fun)
                    4'b0000: begin CPU_ctrl_signals = ?;   end     //add
                    4’b0001: begin CPU_ctrl_signals = ?;   end     //sub
                                                ……
                    default:      begin CPU_ctrl_signals = ?;   end;
                  endcase
                5’b00000: begin CPU_ctrl_signals = ?;   end                  //load
                5’b01000: begin CPU_ctrl_signals = ?;   end                  //store
                                                                ……
                default:      begin  CPU_ctrl_signals = ?;  end
                endcase
        end
```

CPU调试与测试

- 调试
  - SCPU_ctrl_more模块仿真
    - 设计测试激励代码仿真测试*
  - Data_path_more模块仿真
    - 设计测试激励代码仿真测试*
  - CPU功能仿真（仿真测试平台参见lab04-2）

若含有提供的EDF格式IP则无法仿真

直接调用.v形式的子模块

- 集成替换
  - 仿真正确后逐个替换Exp04-2的相应模块
  - 使用DEMO程序（或另外编写）目测控制器正常运行

```
memory_initialization_radix=16;
memory_initialization_vector=
00007293,00007313,88888137,00832183,0032A223,00402083,01C02383,00338863,555550B7,0070A0B3,
FE0098E3,007282B3,00230333,00531463,40000033,40530433,405304B3,0080006F,00007033,0072F533,
00157593,00B51463,00006033,00A5E5B3,0015E513,00558463,00004033,00A5C633,00164613,00B61463,
00000013,0012D293,00060463,40000033,00129293,00B28463,00000013,001026B3,00503733,F65FF06F;
```

设计测试记录表格

- **CPU指令**测试结果记录
  - 自行设计记录表格

思考题

- 指令扩展时控制器用二级译码设计存在什么问题？
- 设计bne指令需要增加控制信号吗？
- 设计srai时需要增加新的数据通道吗？


END