实验四　单周期CPU设计

2022春季

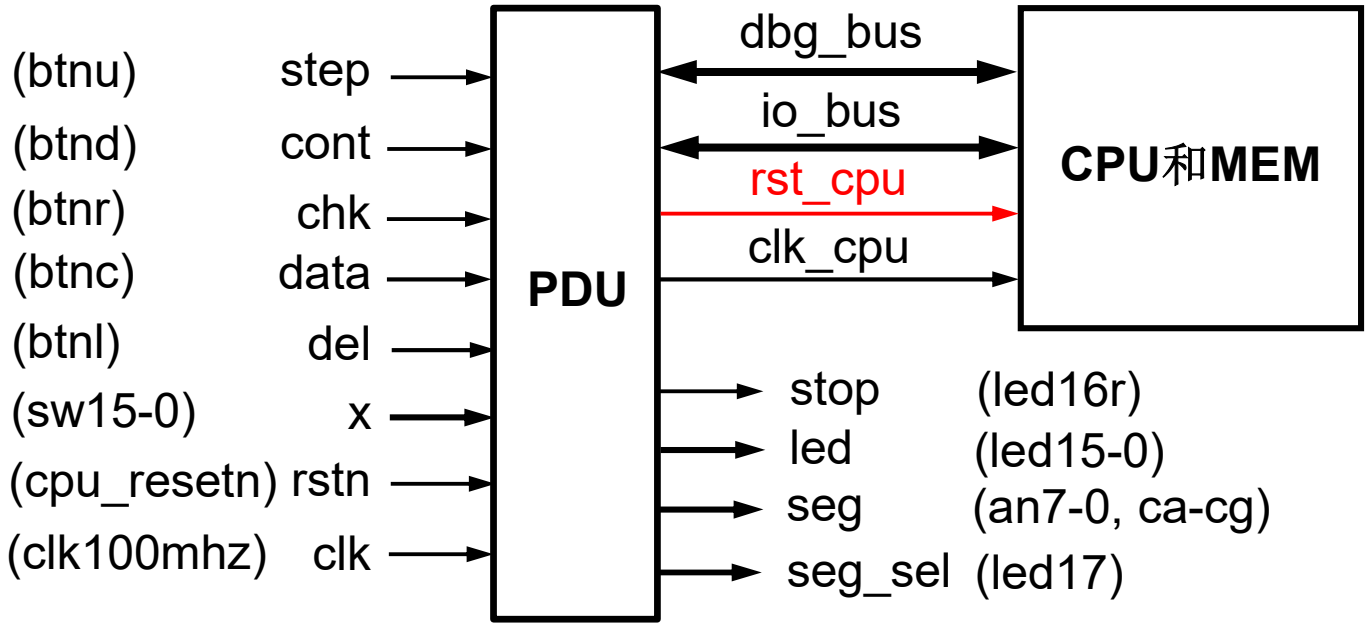
zjx@ustc.edu.cn

实验目标

- 理解单周期**CPU**的结构和工作原理
- 掌握单周期**CPU**的设计和调试方法
- 熟练掌握数据通路和控制器的设计和描述方法

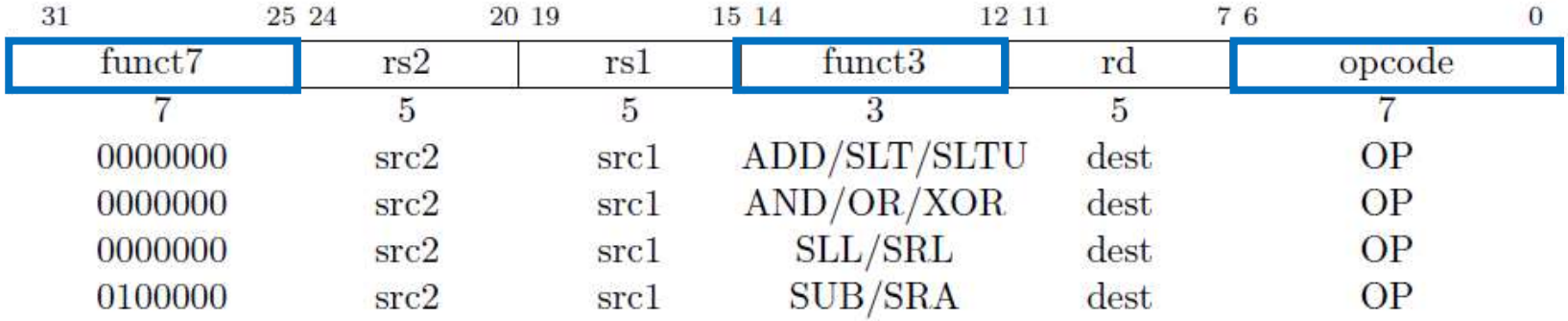
实验内容

- 设计单周期**RISC-V CPU**，可执行以下**10**条指令
  - add, addi, sub, auipc, lw, sw, beq, blt, jal, jalr
- 配合外设和调试单元**PDU**，实现对**CPU**的下载测试



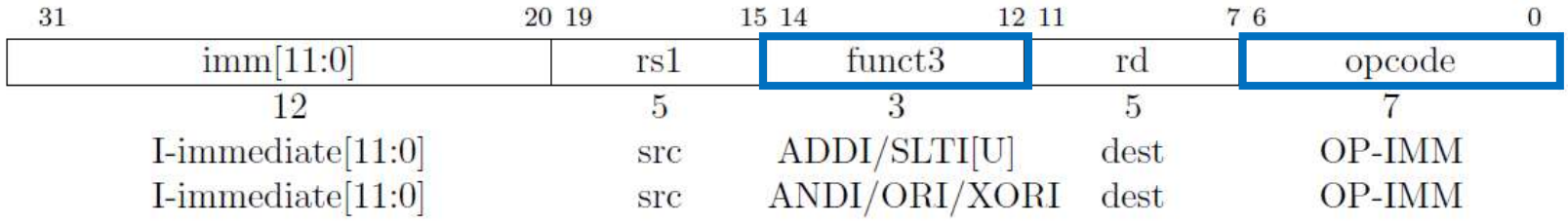
运算指令

- add rd, rs1, rs2 # x[rd] = x[rs1] + x[rs2]



31 25	24 20	19 15	14 12	11 7	6 0
funct7	rs2	rs1	funct3	rd	opcode
7	5	5	3	5	7
0000000	src2	src1	ADD/SLT/SLTU	dest	OP
0000000	src2	src1	AND/OR/XOR	dest	OP
0000000	src2	src1	SLL/SRL	dest	OP
0100000	src2	src1	SUB/SRA	dest	OP

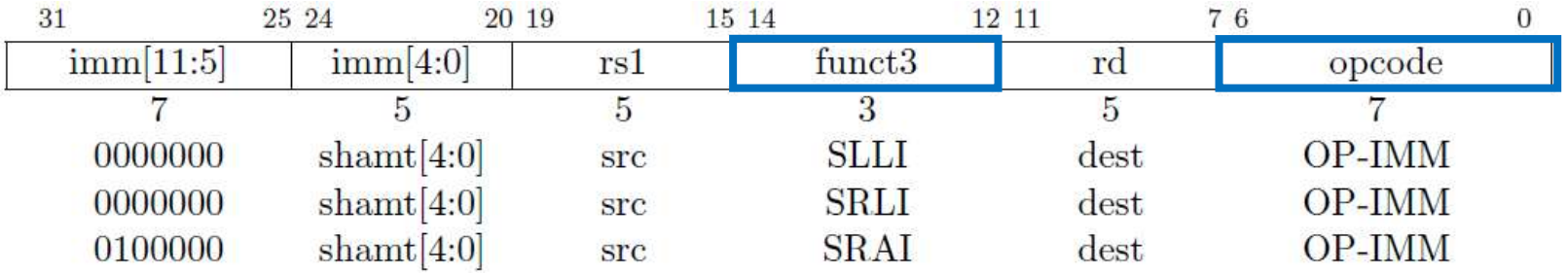
- addi rd, rs1, imm # x[rd] = x[rs1] + sext(imm)



31 20	19 15	14 12	11 7	6 0
imm[11:0]	rs1	funct3	rd	opcode
12	5	3	5	7
I-immediate[11:0]	src	ADDI/SLTI[U]	dest	OP-IMM
I-immediate[11:0]	src	ANDI/ORI/XORI	dest	OP-IMM

运算指令 (续)

- srai rd, rs1, shamt # x[rd] = (x[rs1] $\gg_s$ shamt)



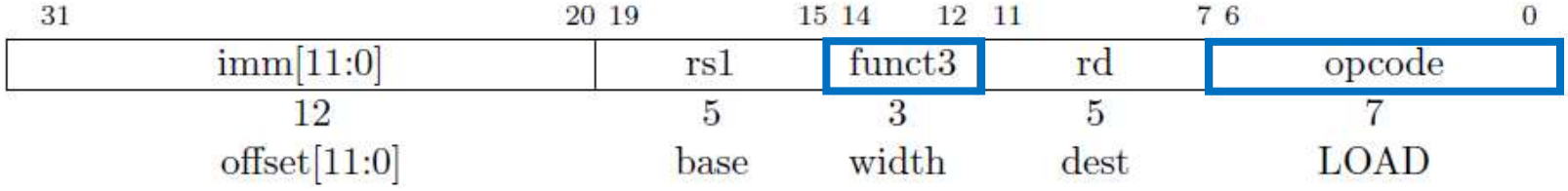
31 25	24 20	19 15	14 12	11 7	6 0
imm[11:5]	imm[4:0]	rs1	funct3	rd	opcode
7	5	5	3	5	7
0000000	shamt[4:0]	src	SLLI	dest	OP-IMM
0000000	shamt[4:0]	src	SRLI	dest	OP-IMM
0100000	shamt[4:0]	src	SRAI	dest	OP-IMM

- lui rd, imm # x[rd] = sext(imm[31:12] << 12)
- auipc rd, imm # x[rd] = pc + sext(imm[31:12] << 12)

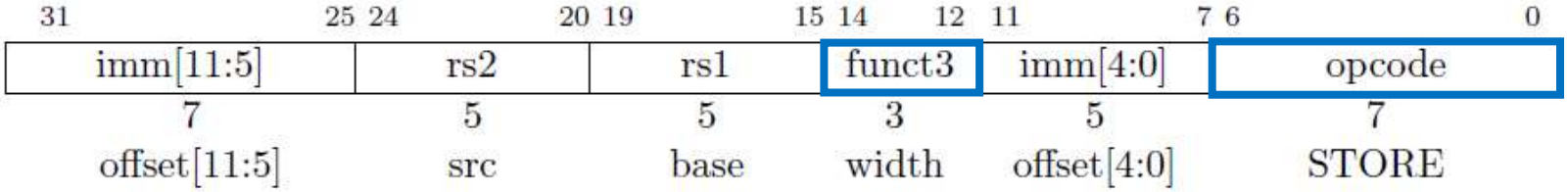
31 12	11 7	6 0
imm[31:12]	rd	opcode
20	5	7
U-immediate[31:12]	dest	LUI
U-immediate[31:12]	dest	AUIPC

访存指令

- lw rd, offset(rs1)　　# x[rd] = M[x[rs1] + sext(offset)]

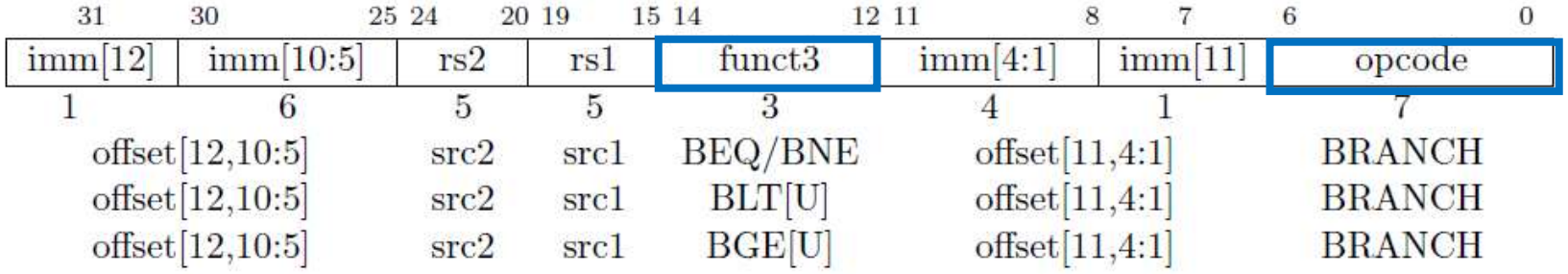


- sw rs2, offset(rs1)　　# M[x[rs1]+sext(offset)=x[rs2]



分支指令

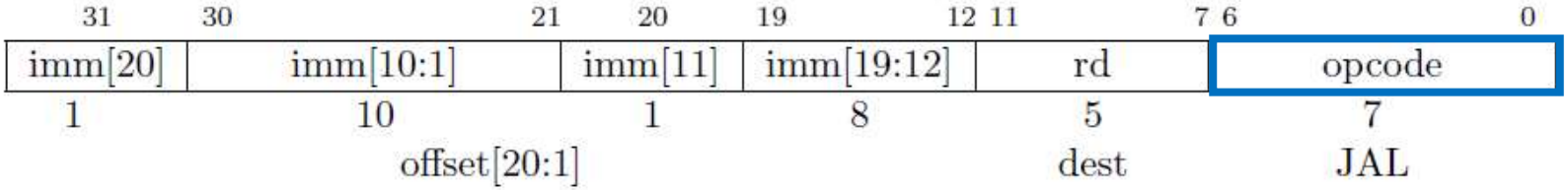
- beq rs1, rs2, offset # if (rs1 == rs2) pc += sext(offset)
- blt rs1, rs2, offset # if (rs1 $<_s$ rs2) pc += sext(offset)
- bltu rs1, rs2, offset # if (rs1 $<_u$ rs2) pc += sext(offset)



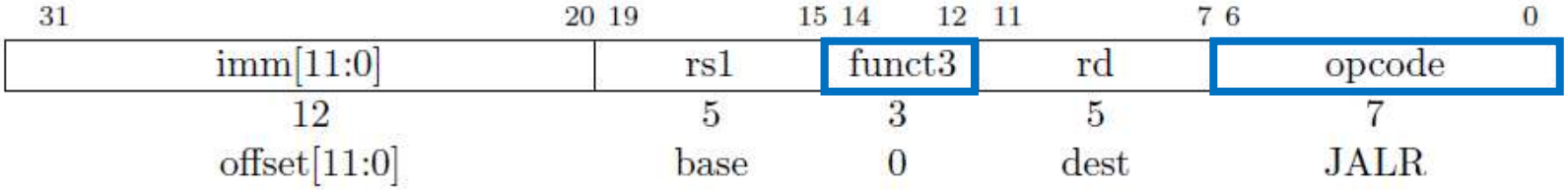
31	30 25	24 20	19 15	14 12	11 8	7	6 0
imm[12]	imm[10:5]	rs2	rs1	funct3	imm[4:1]	imm[11]	opcode
1	6	5	5	3	4	1	7
offset[12,10:5]		src2	src1	BEQ/BNE	offset[11,4:1]		BRANCH
offset[12,10:5]		src2	src1	BLT[U]	offset[11,4:1]		BRANCH
offset[12,10:5]		src2	src1	BGE[U]	offset[11,4:1]		BRANCH

跳转指令

- jal rd, offset　　# x[rd] = pc+4; pc += sext(offset)



- jalr rd, offset(rs1)　# t =pc+4; pc=(x[rs1]+sext(offset))&~1; x[rd]=t



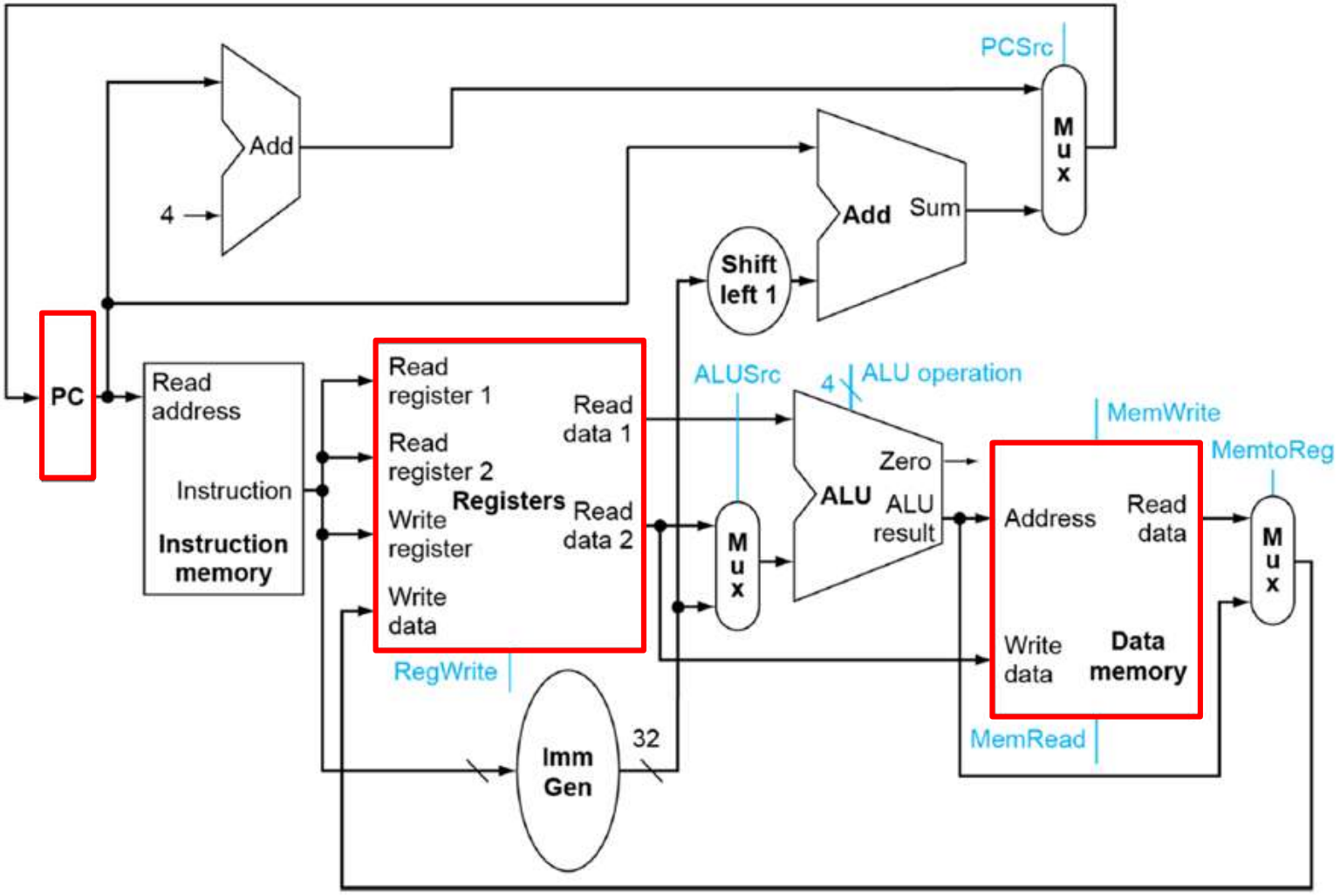
RV32I 指令编码

imm[11:0]		rs1	000	rd	0010011	ADDI
imm[11:0]		rs1	010	rd	0010011	SLTI
imm[11:0]		rs1	011	rd	0010011	SLTIU
imm[11:0]		rs1	100	rd	0010011	XORI
imm[11:0]		rs1	110	rd	0010011	ORI
imm[11:0]		rs1	111	rd	0010011	ANDI
0000000	shamt	rs1	001	rd	0010011	SLLI
0000000	shamt	rs1	101	rd	0010011	SRLI
0100000	shamt	rs1	101	rd	0010011	SRAI
0000000	rs2	rs1	000	rd	0110011	ADD
0100000	rs2	rs1	000	rd	0110011	SUB
0000000	rs2	rs1	001	rd	0110011	SLL
0000000	rs2	rs1	010	rd	0110011	SLT
0000000	rs2	rs1	011	rd	0110011	SLTU
0000000	rs2	rs1	100	rd	0110011	XOR
0000000	rs2	rs1	101	rd	0110011	SRL
0100000	rs2	rs1	101	rd	0110011	SRA
0000000	rs2	rs1	110	rd	0110011	OR
0000000	rs2	rs1	111	rd	0110011	AND

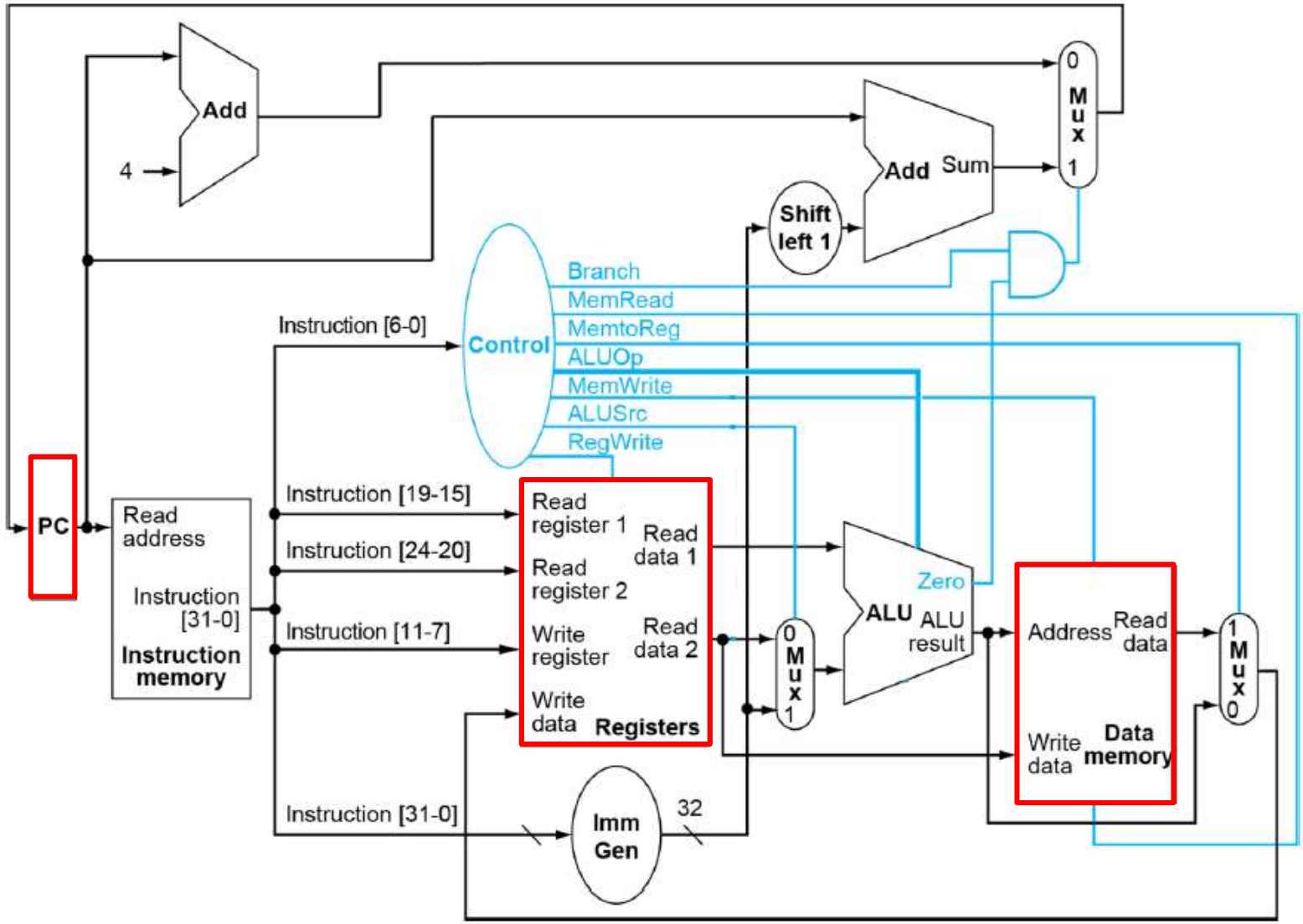
RV32I 指令编码 (续)

imm[31:12]				rd	0110111	LUI
imm[31:12]				rd	0010111	AUIPC
imm[20|10:1|11|19:12]				rd	1101111	JAL
imm[11:0]		rs1	000	rd	1100111	JALR
imm[12|10:5]	rs2	rs1	000	imm[4:1|11]	1100011	BEQ
imm[12|10:5]	rs2	rs1	001	imm[4:1|11]	1100011	BNE
imm[12|10:5]	rs2	rs1	100	imm[4:1|11]	1100011	BLT
imm[12|10:5]	rs2	rs1	101	imm[4:1|11]	1100011	BGE
imm[12|10:5]	rs2	rs1	110	imm[4:1|11]	1100011	BLTU
imm[12|10:5]	rs2	rs1	111	imm[4:1|11]	1100011	BGEU
imm[11:0]		rs1	000	rd	0000011	LB
imm[11:0]		rs1	001	rd	0000011	LH
imm[11:0]		rs1	010	rd	0000011	LW
imm[11:0]		rs1	100	rd	0000011	LBU
imm[11:0]		rs1	101	rd	0000011	LHU
imm[11:5]	rs2	rs1	000	imm[4:0]	0100011	SB
imm[11:5]	rs2	rs1	001	imm[4:0]	0100011	SH
imm[11:5]	rs2	rs1	010	imm[4:0]	0100011	SW

单周期CPU数据通路

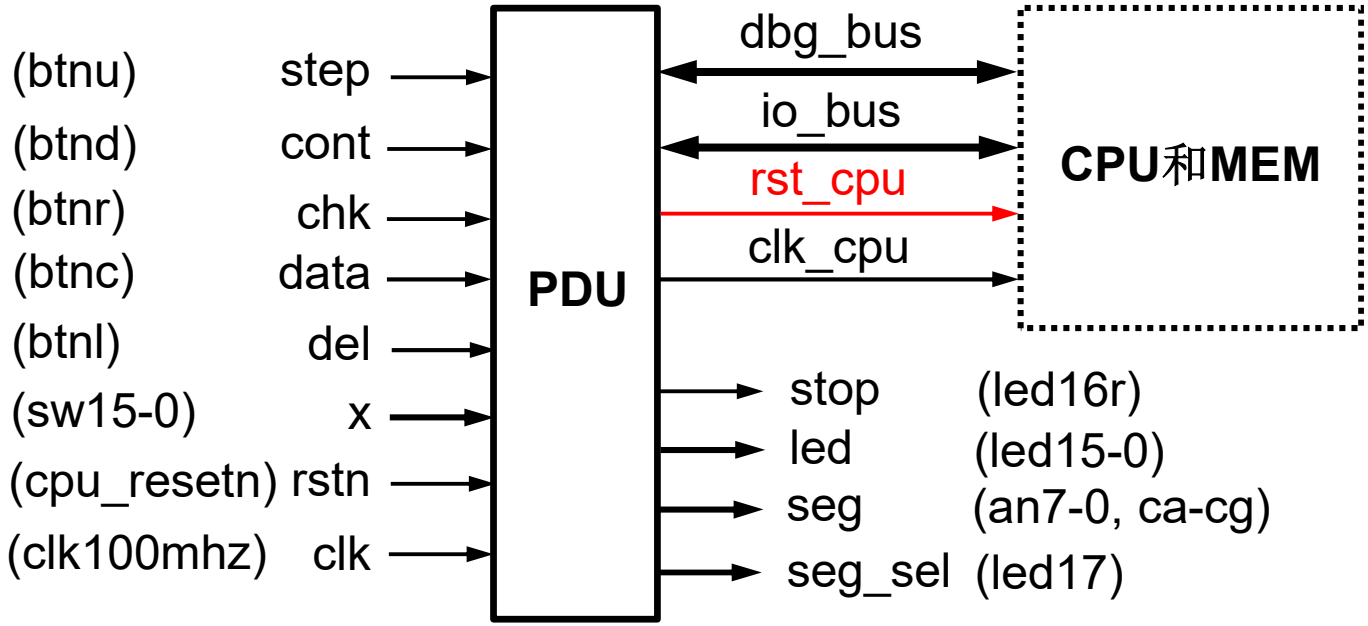

PCSrc
Add
4
M
u
x
Add
Sum
Shift
left 1
PC
Read
address
Instruction
Instruction
memory
Read
register 1
Read
register 2
Registers
Write
register
Write
data
Read
data 1
Read
data 2
RegWrite
ALUSrc
4
ALU operation
M
u
x
ALU
Zero
ALU
result
MemWrite
MemtoReg
Address
Read
data
Write
data
Data
memory
MemRead
M
u
x
Imm
Gen
32

单周期CPU数据通路+控制器



外设和调试单元

- **PDU：Peripherals and Debug Unit**
  - 控制CPU运行方式，查看数据通路状态
  - 管理外设 (开关sw、指示灯led、数码管seg、计数器cnt等)，实现基本输入/输出



CPU运行调试

- 控制CPU运行方式
  - step: 单步运行，按动step，CPU执行一条指令后停止(stop = 1)
  - cont: 连续运行，利用x和del编辑断点地址 (brk_addr)，按动cont，CPU连续运行 (stop = 0)，直至chk_pc = brk_addr后停止 (stop = 1)
- 查看数据通路状态
  - 当CPU停止 (stop = 1) 时，利用x和del编辑查看地址 (chk_addr)，按动chk，chk_addr和数据通路状态(chk_data)分别显示在指示灯led和数码管seg上，再次单独按动chk，将顺序显示后续信息
- 调试信号DBG_BUS
  - chk_pc：输入，32位，监测执行指令地址 = npc
  - chk_addr：输出，16位，数据通路状态的编码地址
  - chk_data：输入，32位，数据通路状态的数据

查看数据通路状态

- 数据通路地址编码
  - chk_addr：4位16进制数，最高位区分查看数据类型（寄存器堆RF、数据存储器DM、PC及其他)，余下位表示具体地址

chk_addr	chk_data
0 00x	pcs
1 0yy	RF
2 zzz	DM

x：流水段寄存器编号

yy：寄存器堆地址

zzz：数据存储器地址

x, y, z：十六进制数字

x	chk_data	说明
0	npc	下一条执行指令的地址
1	pc	当前执行指令的地址
2	instr	当前执行指令的代码
3	ctrl	当前执行指令的控制信号
4	a	RF读出的第1个源操作数
5	b	RF读出的第2个源操作数
6	imm	生成的立即数
7	y	ALU运算结果
8	mdr	数据存储器读出的数据

CPU输入/输出

- **CPU通过存储器映射的输入/输出 (**Memory-Mapped Input/Output, **MMIO)方式访问开关、指示灯、数码管、计数器等I/O设备(外设)**
- **输入/输出信号IO_BUS**
  - io_addr：输入，8位，外设地址
  - io_dout：输入，32位，外设输出数据
  - io_din：输出，32位，外设输入数据
  - io_we：输入，1位，外设写控制信号
  - io_rd：输入，1位，外设读控制信号

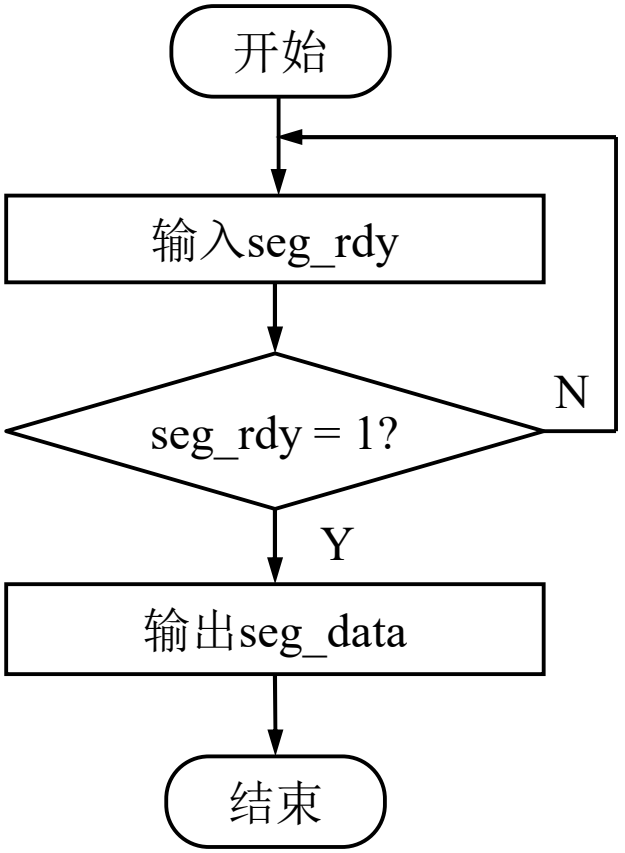
I/O端口

- 直接I/O端口：led_data、swt_data、cnt_data
- 查询式I/O端口：seg_data、seg_rdy、swx_data、swx_vld

序号	偏移地址	I/O端口名称	I/O端口类型	外设说明
0	0x00	led_data	Output	led15-0
1	0x04	swt_data	Input	btn, sw15-0
2	0x08	seg_rdy	Input	数码管准备好
3	0x0C	seg_data	Output	数码管输出数据
4	0x10	swx_vld	Input	开关输入有效
5	0x14	swx_data	Input	开关输入数据
6	0x18	cnt_data	Input	计数器数据

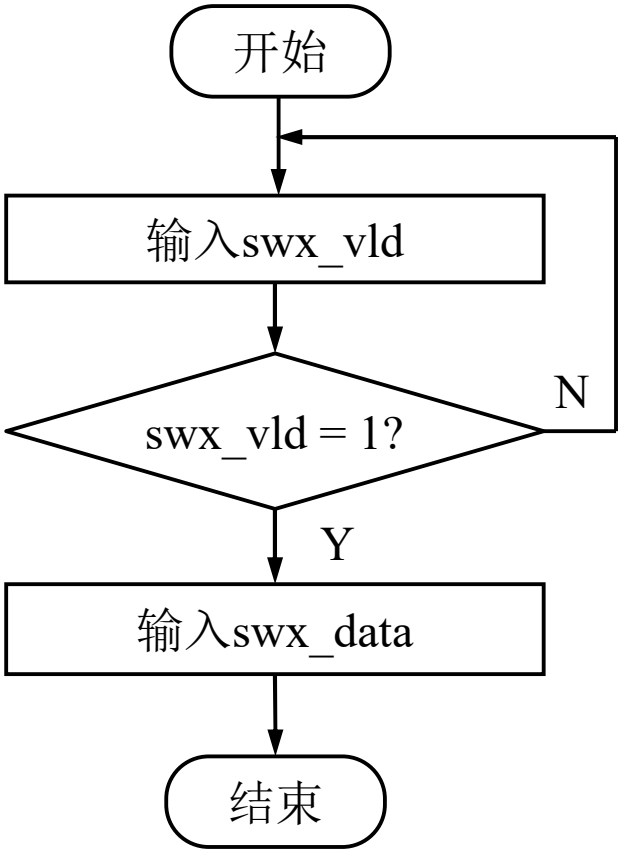
查询式输出过程

- 复位时，**PDU**将数码管准备好标志置**1 (seg_rdy = 1)**
- **CPU**输出**seg_data**时，**PDU**保存该数据，同时自动将**seg_rdy**清零
- 按任意键**(x、del)**，**PDU**自动将**seg_rdy**置 **1**（表示数码管输出的数据已被查看，可以接收下一个数据输出）



查询式输入过程

- 复位时，**PDU**将输入数据有效标志清零 **(swx_vld = 0)**
- **x、del**：编辑输入数据**(tmp)**
- **data**：如果**swx_vld = 0**，则将**tmp**保存至**swx_data**，并将**tmp**清零和**swx_vld**置 **1**
- **CPU**读取**swx_data**时，**PDU**自动将**swx_vld**清零



开关多用途输入

- 利用开关 **(sw15-0)** 一次输入**1**位十六进制数字，按钮 **(btnl)** 一次删除 **(del) 1**位十六进制数字，可以编辑多位十六进制数据，同时显示在数码管上
- 依据随后按动的功能按钮不同，该数据有不同的含义
  - 断点地址(brk_addr)：在CPU停止时，按动连续运行cont按钮(btnd)
  - 查看地址(chk_addr)：在CPU停止时，按动查看chk按钮(btnr)
  - 开关输入数据(swx_data)：在CPU运行或停止时，按动数据data按钮(btnc)

操作次序	数码管显示
x (sw1)	00000001
x (sw2)	00000012
x (sw0)	00000120
del (btnl)	00000012
x (sw10)	0000012A
x (sw11)	000012AB

数码管多用途显示

- **利用8个7段数码管可以显示8位十六进制数据，即32位二进制数据**
- **在不同的上下文情况下，该数据有不同的含义，利用三色指示灯(led17)指示当前数码管显示数据类型(seg_sel)**
  - 数码管输出数据(seg_data)：CPU运行时输出到数码管的数据，led17=蓝色
  - 开关多用途编辑数据：断点地址(brk_addr)、查看地址(chk_addr)、开关输入数据(swx_data)，led17=绿色
  - 数据通路状态数据：CPU停止时，按动查看(chk)按钮(btnr)显示寄存器堆RF、数据存储器DM、PC等内容，led17=红色

CPU模块接口

```
module  cpu (
  input clk,
  input rst,

  //IO_BUS
  output [7:0]  io_addr,      //外设地址
  ouput [31:0]  io_dout,      //向外设输出的数据
  output  io_we,              //向外设输出数据时的写使能信号
  output  io_rd,              //从外设输入数据时的读使能信号
  input [31:0]  io_din,       //来自外设输入的数据

  //Debug_BUS
  output [31:0] chk_pc,       //监测执行指令地址 = npc
  input [15:0] chk_addr,      //数据通路状态的编码地址
  output [31:0] chk_data     //数据通路状态的数据
);
```

PDU模块接口

```
module  pdu (
  input clk,          //clk100mhz
  input rstn,         //cpu_resetn

  input step,         //btnu
  input cont,         //btnd
  input chk,          //btnr
  input data,         //btnc
  input del,          //btnl
  input [15:0]  x,    //sw15-0

  output stop,                  //led16r
  output [15:0] led,            //led15-0
  output [7:0] an,              //an7-0
  output [6:0] seg,             //ca-cg
  output [2:0] seg_sel,         //led17

  output clk_cpu,       //cpu's clk
  output rst_cpu,       //cpu's rst

  //IO_BUS
  input [7:0] io_addr,
  input [31:0] io_dout,
  input io_we,
  input io_rd,
  output [31:0] io_din,

  //Debug_BUS
  input [31:0] chk_pc,
  output [15:0] chk_addr,
  input [31:0] chk_data
);
```

实验步骤

1. **设计单周期CPU，将CPU和PDU整合后下载至FPGA，进行逐条指令功能测试**
   - 指令存储器和数据存储器采用IP例化的分布式存储器，容量均为256x32位，使用LabH3实验步骤1生成的COE文件初始化
   - 寄存器堆和数据存储器各增加一个用于调试的读端口
   - MMIO的起始地址为0
2. **将CPU和PDU整合后下载至FPGA，进行排序程序测试**
   - 使用LabH3实验步骤2生成的COE文件初始化
   - MMIO的起始地址为0xff00
   - 查看电路资源使用情况和电路性能
3. **选项：扩展单周期CPU设计，实现更多条指令功能，并对扩展指令进行下载测试**

The End