# 模块实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 姓名 | 叶福伦 | 学号 | 21051331 |
| 专业 | 计算机科学与技术 | 班级 | 21052313 |
| 指导教师 | 髙志刚 | 课程名称 | 计算机组成原理课程设计（甲） |
| 实验序号 | 07 | 实验名称 | 取码译码实验 |
| 实验时间 | 05.10 | 实验地点 | 1教223 |

# 成绩表

|  |  |  |
| --- | --- | --- |
| **考核项目** | **考查分项** | **得分** |
| **一、实验方案设计（30分）** | **实验目的与要求（5分）** |  |
| **模块构成与连接（15分）** |  |
| **板级验证方案（10分）** |  |
| **二、FPGA程序设计（20分）** | **程序源代码（15分）** |  |
| **管脚约束程序源代码（5分）** |  |
| **三、仿真实验（15分）** | **仿真代码（5分）** |  |
| **仿真波形及结果分析（10分）** |  |
| **四、板级实验（15分）** | **实验操作（5分）** |  |
| **结果记录（5分）** |  |
| **结果分析（5分）** |  |
| **五、思考与探索（10分）** | **拓展提升（10分）** |  |
| **格式（10分）** | **书写规范、图表清晰、表达清楚** |  |
| **合计** | |  |

# 一、实验方案设计

1. 实验目的与要求（5分）

掌握指令存储器、PC与IR的设计方法；

掌握CPU取指令操作与指令译码的方法和过程，掌握指令译码器的设计方法；

理解RISC-V立即数的生成与扩展方法，掌握立即数拼接与扩展器的设计。

编写一个包含所有指令格式的测试程序，或者使用实验6的汇编程序，用其构造一个COE文件。

使用Vivado或者ISE的Memory IP核，新建一个64×32位只读存储器作为指令存储器IM，并关联上述COE文件，初始化其内容。

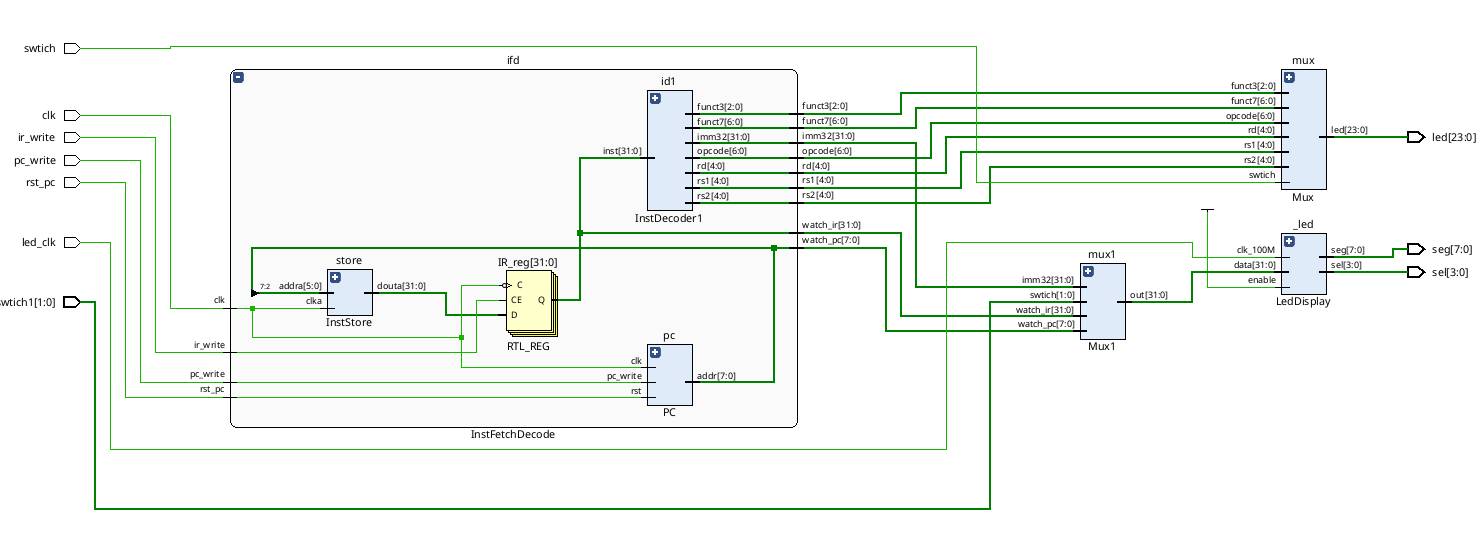
构造PC和IR寄存器，并与上述指令存储器IM连接，构成取指令模块IF；对取指令模块进行仿真，确保能正确取出指令，并且PC自增。

编写指令译码模块ID1，包括立即数拼接与扩展器ImmU；对ID1模块进行仿真，确保指令译码和产生的立即数正确。

连接取指令模块IF和指令译码模块ID1。

1. 模块构成与连接（15分）

（请附图，并说明各模块的功能，以及模块间的连接关系。也可附开发工具中显示的电路模块连接图，加以说明。）



主要是取码译码模块。mux模块用于funct3, funct7, opcode, rs1, rs2, rd的复用输出。mux1使用swtich1[1:0]形成imm32，watch\_pc, watch\_ir 的复用输出信号，输出out。连接到\_led上从而在数码管上显示。rst\_pc用于pc寄存器的复位。ir\_write和pc\_write控制指令寄存器和pc寄存器的写入。InstStore在每一个时钟的上升沿根据addr取出对应指令。在每一个时钟的下降沿尝试写入ir寄存器.PC寄存器在每一个下降沿尝试自增指向下一条指令的地址

1. 板级验证方案（10分）

（说明顶层测试模块的输入/输出信号，及对应使用的I/O设备；可附图说明）

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 信号名 | 使用的IO设备 | 说明 |
| input | led\_clk | 全局时钟 | 驱动数码管显示 |
| clk | 按键 | 模拟取码译码时钟 |
| rst\_pc | 按键 | pc复位信号 |
| pc\_write | 逻辑开关 | pc写允许 |
| ir\_write | 逻辑开关 | ir写允许 |
| switch | 逻辑开关 | 复用器切换开关 |
| Switch1[1:0] | 逻辑开关 | 复用器切换开关 |
| output | led[23:0] | led灯 | 显示rs1/2,rd,  opcode,func3,func7 |
| sel[3:0] | 数码管 | 显示imm32, watch\_pc, watch\_ir的复用输出信号 |
| seg[7:0] |

# 二、FPGA程序设计

1. 实验程序源代码（25分）

（实验各个模块的代码，包含功能注释）

//编译反编译自 acc.s

//InstStore.coe

memory\_initialization\_radix=16;

memory\_initialization\_vector=000002b3 00000333 00a00393 04032e03 01c282b3 00430313 fff38313 00038463 fedff06f 08502023;

//opcode.v

//opcode编码

`define INST\_R 7'b0110011 //R-type

`define INST\_I 7'b0010011 //I-type

`define INST\_L 7'b0000011 //L-type

`define INST\_S 7'b0100011 //S-type

`define INST\_B 7'b1100011 //B-type(beq)

`define INST\_LUI 7'b0110111 //lui

`define INST\_AUIPC 7'b0010111 //auipc 不使用

`define INST\_JAL 7'b1101111 //jal

`define INST\_JALR 7'b1100111 //jalr

//InstDecoder1.v 初级译码器

`include "../define/opcode.v"

module InstDecoder1(

input [31:0] inst,

output reg [4:0] rs1, rs2, rd,

output reg [31:0] imm32,

output reg [6:0] opcode,

output reg [2:0] funct3,

output reg [6:0] funct7

);

always @(\*) begin

opcode = inst[6:0];

rs1 = inst[19:15];

rs2 = inst[24:20];

rd = inst[11:7];

funct3 = inst[14:12];

funct7 = inst[31:25];

case(opcode)

`INST\_R: begin imm32 = 0; end

`INST\_I: begin

imm32 = (inst[14:12]==3'b101 && inst[30]==1'b1) ?

{27'b0, inst[24:20]}: //对应srai指令

{{20{inst[31]}}, inst[31:20]};

end

`INST\_L: begin imm32 = {{20{inst[31]}}, inst[31:20]} ;end

`INST\_S: begin imm32 = {{20{inst[31]}}, inst[31:25], inst[11:7]} ;end

`INST\_B: begin imm32 = {{20{inst[31]}}, inst[7], inst[30:25], inst[11:8], 1'b0} ;end

`INST\_LUI: begin imm32 = {inst[31:12], 12'b0} ;end

`INST\_AUIPC: begin imm32 = {inst[31:12], 12'b0} ;end

`INST\_JAL: begin imm32 ={{12{inst[31]}}, inst[19:12], inst[20], inst[30:21], 1'b0};end

`INST\_JALR: begin imm32 = {{20{inst[31]}}, inst[31:20]} ;end

default: begin imm32 = 0;end

endcase

end

endmodule

//PC.v

module PC(

input clk,

input rst,

input pc\_write,

output reg [7:0] addr

);

always @(negedge clk or posedge rst) begin

if(rst)begin

addr <= 0;

end else if(pc\_write)begin

addr <= addr + 4;

end

end

endmodule

`include "./PC.v"

`include "./InstDecoder1.v"

//InstFetchDecode.v

`include "./PC.v"

`include "./InstDecoder1.v"

module InstFetchDecode(

input clk,

input rst\_pc,

input pc\_write,

input ir\_write,

output [4:0] rs1, rs2, rd,

output [31:0] imm32,

output [6:0] opcode,

output [2:0] funct3,

output [6:0] funct7,

output [7:0] watch\_pc,

output [31:0] watch\_ir

);

wire [7:0] ins\_addr;

assign watch\_pc = ins\_addr;

PC pc(

.clk(clk),

.rst(rst\_pc),

.pc\_write(pc\_write),

.addr(ins\_addr)

);

wire[31:0] inst;

InstStore store(

.clka(clk),

.addra(ins\_addr[7:2]),

.douta(inst)

);

reg [31:0] IR;

assign watch\_ir = IR;

InstDecoder1 id1(

.inst(IR),

.rs1(rs1),

.rs2(rs2),

.rd(rd),

.imm32(imm32),

.opcode(opcode),

.funct3(funct3),

.funct7(funct7)

);

always @(negedge clk) begin

if(ir\_write)IR <= inst;

end

endmodule

//InstFetchDecode.top.v 顶级模块

`include "../include/LedDisplay.v"

module Mux ( //led灯不够用

input [4:0] rs1, rs2, rd, //led

input [6:0] opcode, //led

input [2:0] funct3, //led

input [6:0] funct7, //led

input swtich,

output reg [23:0] led

);

always @(\*) begin

led <= 0;

if(swtich)begin

led[23:19] <= rs1;

led[15:11] <= rs2;

led[7:3] <= rd;

end else begin

led[23:17] <= opcode;

led[15:13] <= funct3;

led[7:1] <= funct7;

end

end

endmodule

module Mux1 (

input [1:0] swtich,

input [31:0] imm32,

input [7:0] watch\_pc,

input [31:0] watch\_ir,

output reg [31:0] out

);

always@(\*) begin

case(swtich)

2'b00: out <= imm32;

2'b01: out <= {24'b0,watch\_pc};

2'b10: out <= watch\_ir;

2'b11: out <= out;

endcase

end

endmodule

module InstFetchDecodeTopper (

input led\_clk,

input clk, //按钮模拟，一个周期

input rst\_pc, //pc寄存器指向0地址

input pc\_write, //enable 1

input ir\_write, //enable 1

input swtich,

input [1:0] swtich1,

output [3:0] sel, //数码管

output [7:0] seg, //数码管

output [23:0] led

);

wire [31:0]mux1\_out;

wire [4:0] rs1, rs2, rd;

wire [6:0] opcode;

wire [2:0] funct3;

wire [6:0] funct7;

wire [31:0] imm32;

wire [7:0] watch\_pc;

wire [31:0] watch\_ir;

Mux1 mux1(

swtich1,

imm32,

watch\_pc,

watch\_ir,

mux1\_out

);

InstFetchDecode ifd(

clk,

rst\_pc,

pc\_write,

ir\_write,

rs1,

rs2,

rd,

imm32,

opcode,

funct3,

funct7,

watch\_pc,

watch\_ir

);

LedDisplay \_led(

led\_clk,

mux1\_out,

1,

sel,

seg

);

Mux mux(

rs1,

rs2,

rd,

opcode,

funct3,

funct7,

swtich,

led

);

endmodule

1. FPGA管脚约束代码（5分）

（引脚约束文件的内容，备注使用的具体设备）

led\_clk使用全局时钟。clk使用按键模拟。rst\_pc使用按键。pc\_write和ir\_write使用逻辑开关。switch使用逻辑开关。switch1[1:0]使用一组逻辑开关。sel[3:0]和seg[7:0]共同驱动数码管。led[23:0]使用led灯

set\_property BITSTREAM.GENERAL.COMPRESS TRUE [current\_design]

set\_property IOSTANDARD LVCMOS18 [get\_ports {seg[7]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {seg[6]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {seg[5]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {seg[4]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {seg[3]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {seg[2]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {seg[1]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {seg[0]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {sel[3]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {sel[2]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {sel[1]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {sel[0]}]

set\_property PACKAGE\_PIN H19 [get\_ports {seg[7]}]

set\_property PACKAGE\_PIN G20 [get\_ports {seg[6]}]

set\_property PACKAGE\_PIN J22 [get\_ports {seg[5]}]

set\_property PACKAGE\_PIN K22 [get\_ports {seg[4]}]

set\_property PACKAGE\_PIN K21 [get\_ports {seg[3]}]

set\_property PACKAGE\_PIN H20 [get\_ports {seg[2]}]

set\_property PACKAGE\_PIN H22 [get\_ports {seg[1]}]

set\_property PACKAGE\_PIN J21 [get\_ports {seg[0]}]

set\_property PACKAGE\_PIN L21 [get\_ports {sel[3]}]

set\_property PACKAGE\_PIN M22 [get\_ports {sel[2]}]

set\_property PACKAGE\_PIN M21 [get\_ports {sel[1]}]

set\_property PACKAGE\_PIN N22 [get\_ports {sel[0]}]

set\_property -dict {IOSTANDARD LVCMOS18 PACKAGE\_PIN H4} [get\_ports led\_clk]

set\_property IOSTANDARD LVCMOS18 [get\_ports {led[0]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {led[1]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {led[2]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {led[3]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {led[4]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {led[5]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {led[6]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {led[7]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {led[8]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {led[9]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports clk]

set\_property IOSTANDARD LVCMOS18 [get\_ports ir\_write]

set\_property IOSTANDARD LVCMOS18 [get\_ports pc\_write]

set\_property IOSTANDARD LVCMOS18 [get\_ports rst\_pc]

set\_property IOSTANDARD LVCMOS18 [get\_ports swtich]

set\_property IOSTANDARD LVCMOS18 [get\_ports {led[23]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {led[22]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {led[21]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {led[20]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {led[19]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {led[18]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {led[17]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {led[16]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {led[15]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {led[14]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {led[13]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {led[12]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {led[11]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {led[10]}]

set\_property PACKAGE\_PIN R4 [get\_ports clk]

set\_property PACKAGE\_PIN T3 [get\_ports ir\_write]

set\_property PACKAGE\_PIN U3 [get\_ports pc\_write]

set\_property PACKAGE\_PIN T4 [get\_ports swtich]

set\_property PACKAGE\_PIN AA4 [get\_ports rst\_pc]

set\_property PULLDOWN true [get\_ports ir\_write]

set\_property PULLDOWN true [get\_ports pc\_write]

set\_property PULLDOWN true [get\_ports swtich]

set\_property PACKAGE\_PIN R1 [get\_ports {led[23]}]

set\_property PACKAGE\_PIN P2 [get\_ports {led[22]}]

set\_property PACKAGE\_PIN P1 [get\_ports {led[21]}]

set\_property PACKAGE\_PIN N2 [get\_ports {led[20]}]

set\_property PACKAGE\_PIN M1 [get\_ports {led[19]}]

set\_property PACKAGE\_PIN M2 [get\_ports {led[18]}]

set\_property PACKAGE\_PIN L1 [get\_ports {led[17]}]

set\_property PACKAGE\_PIN J2 [get\_ports {led[16]}]

set\_property PACKAGE\_PIN G1 [get\_ports {led[15]}]

set\_property PACKAGE\_PIN E1 [get\_ports {led[14]}]

set\_property PACKAGE\_PIN D2 [get\_ports {led[13]}]

set\_property PACKAGE\_PIN A1 [get\_ports {led[12]}]

set\_property PACKAGE\_PIN L3 [get\_ports {led[11]}]

set\_property PACKAGE\_PIN G3 [get\_ports {led[10]}]

set\_property PACKAGE\_PIN K4 [get\_ports {led[9]}]

set\_property PACKAGE\_PIN G4 [get\_ports {led[8]}]

set\_property PACKAGE\_PIN K1 [get\_ports {led[7]}]

set\_property PACKAGE\_PIN J1 [get\_ports {led[6]}]

set\_property PACKAGE\_PIN H2 [get\_ports {led[5]}]

set\_property PACKAGE\_PIN G2 [get\_ports {led[4]}]

set\_property PACKAGE\_PIN F1 [get\_ports {led[3]}]

set\_property PACKAGE\_PIN E2 [get\_ports {led[2]}]

set\_property PACKAGE\_PIN D1 [get\_ports {led[1]}]

set\_property PACKAGE\_PIN B1 [get\_ports {led[0]}]

set\_property PACKAGE\_PIN AB7 [get\_ports {swtich1[1]}]

set\_property PACKAGE\_PIN AB8 [get\_ports {swtich1[0]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {swtich1[1]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {swtich1[0]}]

set\_property PULLDOWN true [get\_ports {swtich1[1]}]

set\_property PULLDOWN true [get\_ports {swtich1[0]}]

# 三、仿真实验

（只对CPU模块仿真，无需对板级验证的顶层测试模块仿真）

1. 仿真代码（5分）

（含仿真源代码、仿真验证方案）

//InstFetchDecode.test.v

module InstFetchDecodeTester();

reg clk = 0;

reg rst\_pc = 0;

wire [4:0] rs1, rs2, rd;

wire [31:0] imm32;

wire [6:0] opcode;

wire [2:0] funct3;

wire [6:0] funct7;

wire [7:0] watch\_pc;

wire [31:0] watch\_ir;

parameter PERIOD = 10;

initial begin

forever #(PERIOD/2) clk=~clk;

end

InstFetchDecode ins\_fetch\_decode(

.clk(clk),

.rst\_pc(rst\_pc),

.pc\_write(1),

.ir\_write(1),

.rs1(rs1),

.rs2(rs2),

.rd(rd),

.imm32(imm32),

.opcode(opcode),

.funct3(funct3),

.funct7(funct7),

.watch\_pc(watch\_pc),

.watch\_ir(watch\_ir)

);

initial begin

rst\_pc = 1;

#20

rst\_pc = 0;

#200

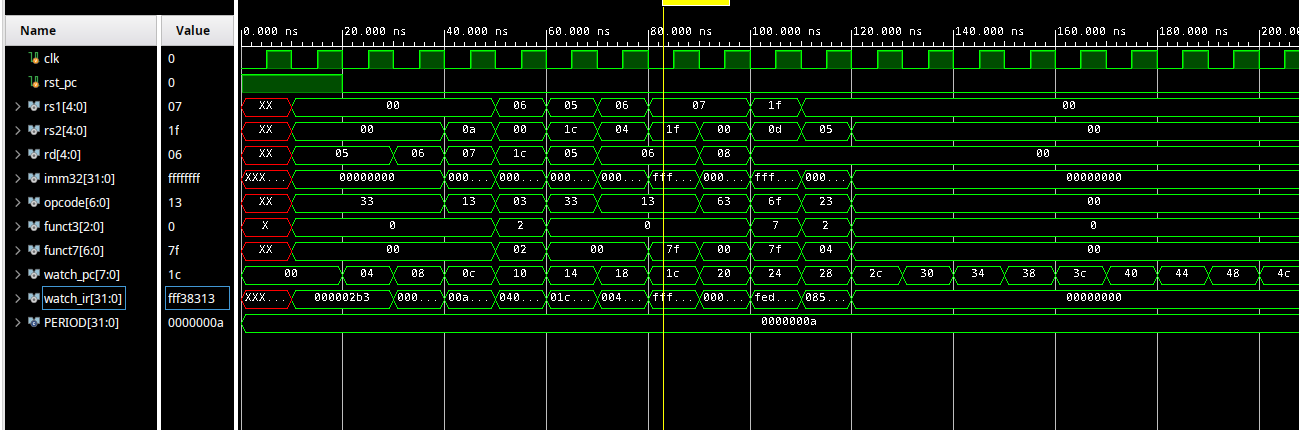
$finish;

end

endmodule

1. 仿真波形及结果分析（15分）

（仿真波形截图，并对仿真波形进行分析）



|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 指令码 | 对应汇编 | rs1 | rs2 | rd | imm32 | opcode | funct3 | funct7 | watch\_pc | watch\_ir | 是否正确 |
| 0000\_02b3 | add t0,zero,zero | 00 | 00 | 05 | 0000\_0000 | 0110011 | 0 | 00 | 04 | 0000\_02b3 | 对 |
| 0000\_0333 | add t1,zero,zero | 00 | 00 | 06 | 0000\_0000 | 0110011 | 0 | 00 | 08 | 0000\_0333 | 对 |
| 00a0\_0393 | li t2,10 | 00 | 0a | 07 | 0000\_000a | 0010011 | 0 | 00 | 0c | 00a0\_0393 | 对 |
| 0403\_2e03 | lw t3,64(t1) | 06 | 00 | 1c | 0000\_0040 | 0000011 | 2 | 02 | 10 | 0403\_2e03 | 对 |
| 01c2\_82b3 | add t0,t0,t3 | 05 | 1c | 05 | 0000\_0000 | 0110011 | 0 | 00 | 14 | 01c2\_82b3 | 对 |
| 0043\_0313 | add t1,t1,4 | 06 | 04 | 06 | 0000\_0004 | 0010011 | 0 | 00 | 18 | 0043\_0313 | 对 |
| fff3\_8393 | add t2,t2,-1 | 07 | 1f | 07 | ffff\_ffff | 0010011 | 0 | 7f | 1c | ffff3\_8313 | 对 |
| 0003\_8463 | beqz t2,24 <L2> | 07 | 00 | 08 | 0000\_0008 | 1100011 | 0 | 00 | 20 | 0003\_8463 | 对 |
| fedf\_f06f | j c <L1> | 1f | 0d | 00 | ffff\_ffec | 1101111 | 7 | 7f | 24 | fedf\_f06f | 对 |
| 0850\_2023 | sw t0,128(zero) | 00 | 05 | 00 | 0000\_0080 | 0100011 | 2 | 04 | 28 | 0850\_2023 | 对 |

# 说明取码译码能力大致正常

# 四、板级实验

1. 板级实验操作说明（5分）

板级实验：按照你所设计的实验方案，操作输入设备、观察输出设备，预期的验证操作如下：

按rst\_n按键，将PC和IR清零，接下来从0号单元开始取指令；

拨开关PC\_Write =1、IR\_Write=1，按动时钟键clk\_im，每按一下，就取出一条指令；

观察输出设备上读出的指令是否按序读出，代码是否与初始化的coe文件中的指令码一致，产生的立即数是否正确，并记录到表中。

1. 板级实验结果记录（5分）

（以表格形式展现，具体见教材上实验结果记录表）

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 指令码 | 对应汇编 | rs1 | rs2 | rd | imm32 | opcode | funct3 | funct7 | watch\_pc | watch\_ir | 是否正确 |
| 0000\_02b3 | add t0,zero,zero | 00 | 00 | 05 | 0000\_0000 | 0110011 | 0 | 00 | 04 | 0000\_02b3 | 对 |
| 0000\_0333 | add t1,zero,zero | 00 | 00 | 06 | 0000\_0000 | 0110011 | 0 | 00 | 08 | 0000\_0333 | 对 |
| 00a0\_0393 | li t2,10 | 00 | 0a | 07 | 0000\_000a | 0010011 | 0 | 00 | 0c | 00a0\_0393 | 对 |
| 0403\_2e03 | lw t3,64(t1) | 06 | 00 | 1c | 0000\_0040 | 0000011 | 2 | 02 | 10 | 0403\_2e03 | 对 |
| 01c2\_82b3 | add t0,t0,t3 | 05 | 1c | 05 | 0000\_0000 | 0110011 | 0 | 00 | 14 | 01c2\_82b3 | 对 |
| 0043\_0313 | add t1,t1,4 | 06 | 04 | 06 | 0000\_0004 | 0010011 | 0 | 00 | 18 | 0043\_0313 | 对 |
| fff3\_8393 | add t2,t2,-1 | 07 | 1f | 07 | ffff\_ffff | 0010011 | 0 | 7f | 1c | ffff3\_8313 | 对 |
| 0003\_8463 | beqz t2,24 <L2> | 07 | 00 | 08 | 0000\_0008 | 1100011 | 0 | 00 | 20 | 0003\_8463 | 对 |
| fedf\_f06f | j c <L1> | 1f | 0d | 00 | ffff\_ffec | 1101111 | 7 | 7f | 24 | fedf\_f06f | 对 |
| 0850\_2023 | sw t0,128(zero) | 00 | 05 | 00 | 0000\_0080 | 0100011 | 2 | 04 | 28 | 0850\_2023 | 对 |

1. 结果分析与结论（5分）

（分析实验结果，给出实验结论）

取码译码器输出的rs1， rs2， rd， imm32， opcode, funct3, funct7字段均正确，取码译码实验成功

# 五、思考与探索

（10分）

1. 问题与解决方案

（整个实验过程中发生了什么问题？你是如何解决的。）

立即码拼接错误，B型和J型指令跳转偏移量出错: 根据ppt修改拼接imm32的代码，

使偏移量修正为正确值

1. 思考题

（力所能及，尝试实践或回答教材上的思考与探索题目，至少完成1道）

1. 0号地址，上升沿根据pc寄存器指向的指令地址读出指令，下降沿pc寄存器才自增

2. 能，根据watch\_pc, watch\_ir和.coe文件可以确定。watch\_pc指向下一条指令地址。当watch\_pc = 4 ,watch\_ir = 000002b3, watch\_pc = 8 ,watch\_ir = ….

所以能够确定取出的指令正确

4. 一个，funct3 slli funct3为001，srai和srli为101

# 六、实验心得体会、意见建议

（如果是小组合作，请给出具体分工、分别给出心得体会）

作为一名计算机组成原理实验课的学生，我已经学习了取指令和译码设计实验。这个实验让我更深入地了解了计算机的指令系统、指令的编码方式和译码过程。以下是我对这次实验的心得体会:

首先，这次实验让我更深入地了解了计算机的指令系统。通过实验，我学习了不同类型的指令，如取指指令、译码指令和操作码等，并了解了这些指令的作用和功能。还学习了如何设计指令，包括指令的编码方式和译码过程。这些知识对于理解计算机的指令系统非常重要，也有助于在未来的研究中更好地掌握计算机的指令系统。

其次，这次实验让我更加熟悉了计算机的译码过程。我学习了不同类型的指令需要使用的译码器，并了解了译码器的工作原理。通过实验，我还学习了如何设计译码器，以及如何根据译码器的设计结果确定指令的操作码。这些知识对于理解计算机的指令系统非常重要，也有助于我在未来的研究中更好地掌握计算机的指令系统。

最后，这次实验让我掌握了计算机组成原理中的重要概念，如指令系统、译码器和操作码等。这些概念对于理解计算机的工作原理和功能非常重要。通过实验，我还能够设计和实现计算机的指令系统，提高了我对计算机组成原理的理解和应用能力。

总之，这次实验让我更深入地了解了计算机的指令系统、译码过程和指令的设计方法。掌握了计算机组成原理中的重要概念，并提高了对计算机组成原理的理解和应用能力。我相信这些经验和知识将对未来的学习和研究产生积极的影响。