浙江大学

本科实验报告

课程名称：	计算机体系结构
姓　　名：	孟展豪
学　　号：	3150104510
指导教师：	陈文智

2018 年　10 月 9 日

浙江大学实验报告

课程名称：计算机体系结构 实验类型：普通实验

实验项目名称：Single Cycle CPU Design

学生姓名：孟展豪 专业：计算机科学与技术 学号：3150104510

同组学生姓名：--- 指导老师：陈文智

实验地点：曹西 301 实验日期：2018 年 10 月 9 日开始

Single Cycle CPU Design

一、 实验介绍：

这个实验将指导理解单周期处理器的各个功能模块组成和内部实现方式。

二、 实验目标：

补全补全各个功能模块源代码中的空缺部分。

对处理器进行仿真，检验处理器的仿真结果是否符合要求。

综合工程并下载至开发板，在单步执行的过程中检查调试屏幕的输出，检验处理器的执行过程是否正确。

三、 实验步骤

1. 建立新工程
2. 设计代码与输入
3. 综合与仿真
4. 下载验证

四、 主要仪器设备

PC 机一台，安装有 ISE 软件

开发板一套

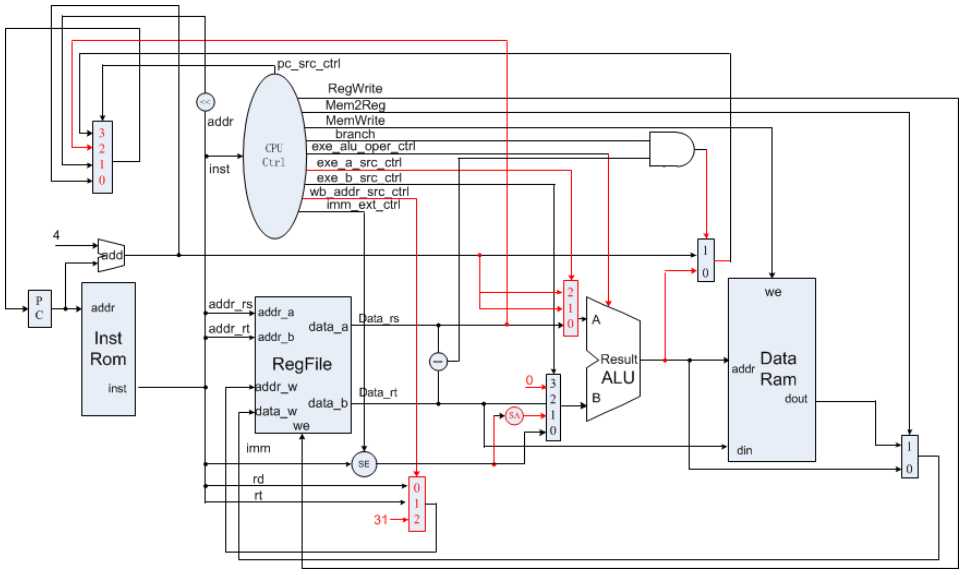
五、 实验原理

Mips 指令集原理

16 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
sub		rs	rt	rd	00000	100010	rd = rs - rt; with overflow	PC+=4
and		rs	rt	rd	00000	100100	rd = rs & rt;	PC+=4
or	000000	rs	rt	rd	00000	100101	rd = rs | rt;	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
slt		rs	rt	rd	00000	101010	if(rs < rt)rd = 1; else rd = 0; <(signed)	PC+=4
jr		rs	00000	00000	00000	001000		PC=rs
I-type	op	rs	rt		immediate			
addi	001000	rs	rt		imm		rt = rs + (sign_extend)imm; with 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
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
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	

CPU Controller 原理图及 datapath 原理



六、 实验具体操作步骤

通过 verliog 语言设计 CPU controller，本次实验中主要为补充 control 缺省部分，并依此设计 bne 指令信号：

```
INST_J: begin
                    pc_src = EXE_B_IMM;
                end
                INST_JAL: begin
                    pc_src = PC_JUMP;
                    exe_a_src = EXE_A_LINK;
                    exe_b_src = EXE_B_LINK;
                    exe_alu_oper = EXE_ALU_ADD;
                    wb_addr_src = WB_ADDR_LINK;
                    wb_data_src = WB_DATA_ALU;
                    wb_wen = 1;
                end
                INST_BNE: begin
                    pc_src = PC_BNE;
                    exe_a_src = EXE_A_BRANCH;
                    exe_b_src = EXE_B_BRANCH;
                    exe_alu_oper = EXE_ALU_ADD;
                    imm_ext = 1;
                end
                INST_ADDI: begin
                    imm_ext = 1;
                    exe_b_src = EXE_B_IMM;
                    exe_alu_oper = EXE_ALU_ADD;
                    wb_addr_src = WB_ADDR_RT;
                    wb_data_src = WB_DATA_ALU;
                    wb_wen = 1;
                end
                INST_ANDI: begin
                    imm_ext = 0;
                    exe_b_src = EXE_B_IMM;
                    exe_alu_oper = EXE_ALU_AND;
                    wb_addr_src = WB_ADDR_RT;
                    wb_data_src = WB_DATA_ALU;
                    wb_wen = 1;
                end
                INST_ORI: begin
                    imm_ext = 0;
                    exe_b_src = EXE_B_IMM;
                    exe_alu_oper = EXE_ALU_OR;
```

```
                wb_addr_src = WB_ADDR_RT;
                wb_data_src = WB_DATA_ALU;
                wb_wen = 1;
            end
            INST_LW: begin
                imm_ext = 1;
                exe_b_src = EXE_B_IMM;
                exe_alu_oper = EXE_ALU_ADD;
                mem_ren = 1;
                wb_addr_src = WB_ADDR_RT;
                wb_data_src = WB_DATA_MEM;
                wb_wen = 1;
            end
            INST_SW: begin
                imm_ext = 1;
                exe_b_src = EXE_B_IMM;
                exe_alu_oper = EXE_ALU_ADD;
                mem_wen = 1;
            end
```

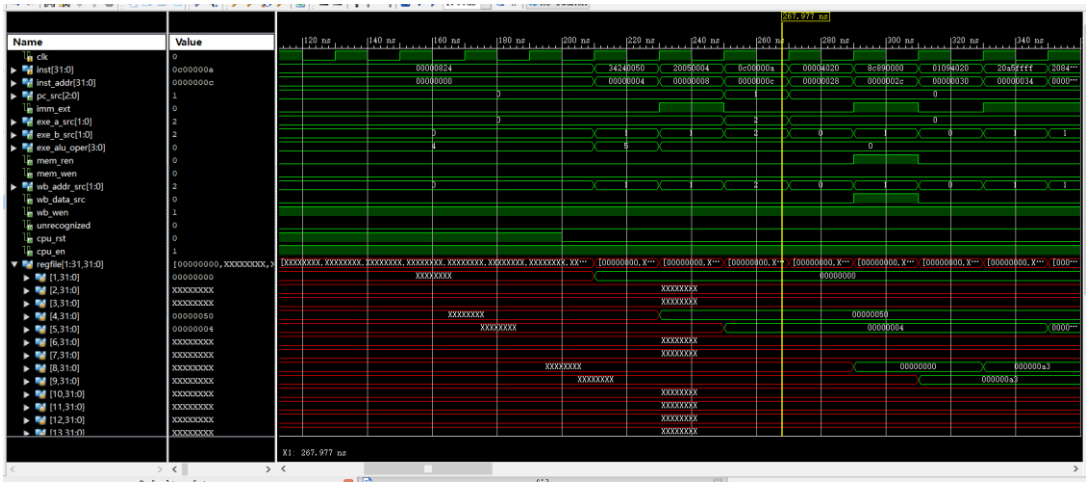
完成 controller 设计后，安装单周期 CPU 流程图，以及 ppt 描述，完成 datapath 设计，尤其注意 j，jl，bne 指令是设计中的重点：

```
always @(posedge clk) begin
        if (cpu_rst) begin
            inst_addr <= 0;
        end
        else if (cpu_en) begin
            case (pc_src_ctrl)
                PC_JUMP: inst_addr <= {inst_addr[31:28],inst_data_ctrl[25:0]
,2'b0};
                PC_JR: inst_addr <= data_rs;
                PC_BNE: if (rs_rt_equal) begin
                            inst_addr <= inst_addr + 4;
                end
                else begin
                    inst_addr <= alu_out;
                end
                default: inst_addr <= inst_addr + 4;
            endcase
        end
    end
```

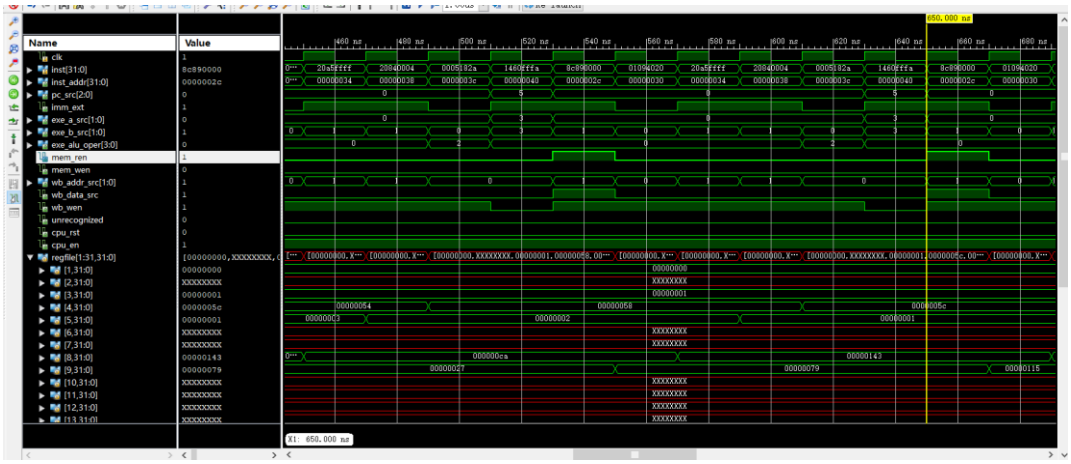
根据 ppt 完成设计：

```
always @(*) begin
        opa = data_rs;
        opb = data_rt;
        case (exe_a_src_ctrl)
            EXE_A_RS: opa = data_rs;
            EXE_A_LINK: opa = inst_addr_next;
            EXE_A_BRANCH: opa = inst_addr_next;
        endcase
        case (exe_b_src_ctrl)
            EXE_B_RT: opb = data_rt;
            EXE_B_IMM: opb = data_imm;
            EXE_B_LINK: opb = 32'h0;
            EXE_B_BRANCH: opb = {data_imm[29:0], 2'b0};
        endcase
    end

    alu ALU (
        .a(opa),
        .b(opb),
        .oper(exe_alu_oper_ctrl),
        .result(alu_out)
        );

    assign
        mem_ren = mem_ren_ctrl & cpu_en & ~cpu_rst,
        mem_wen = mem_wen_ctrl & cpu_en & ~cpu_rst,
        mem_addr = alu_out,
        mem_dout = data_rt;

    always @(*) begin
        regw_data = alu_out;
        case (wb_data_src_ctrl)
            WB_DATA_ALU: regw_data = alu_out;
            WB_DATA_MEM: regw_data = mem_din;
        endcase
    end
```

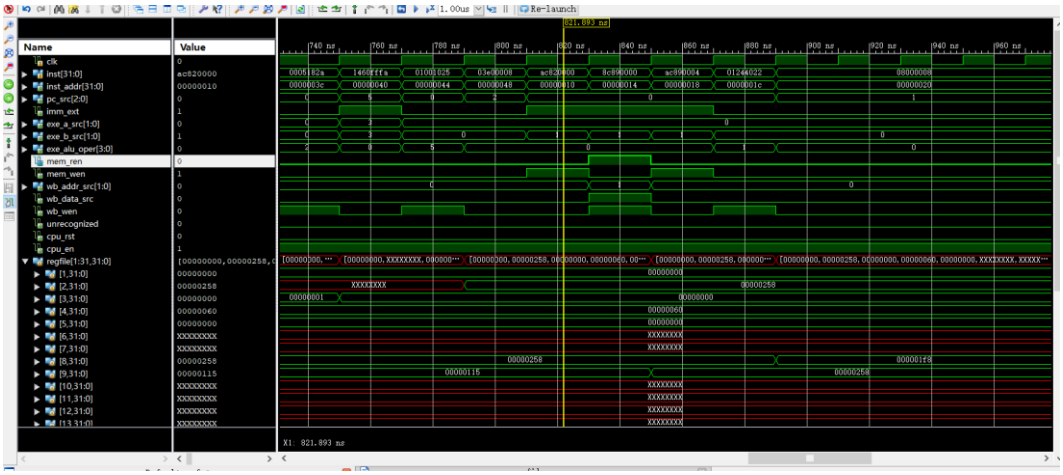
编写测试代码进行仿真测试：



如图所示，测试开始，当前指令地址为 0，执行 00000824 指令，之后指令地址+4 依次按地址执行指令，同时寄存器被赋值；当指令地址为 0000000c 时执行第 4 条 jal 指令，可见下一条指令地址跳转为 00000028，执行 00004020 指令，证明 jal 指令正确执行。

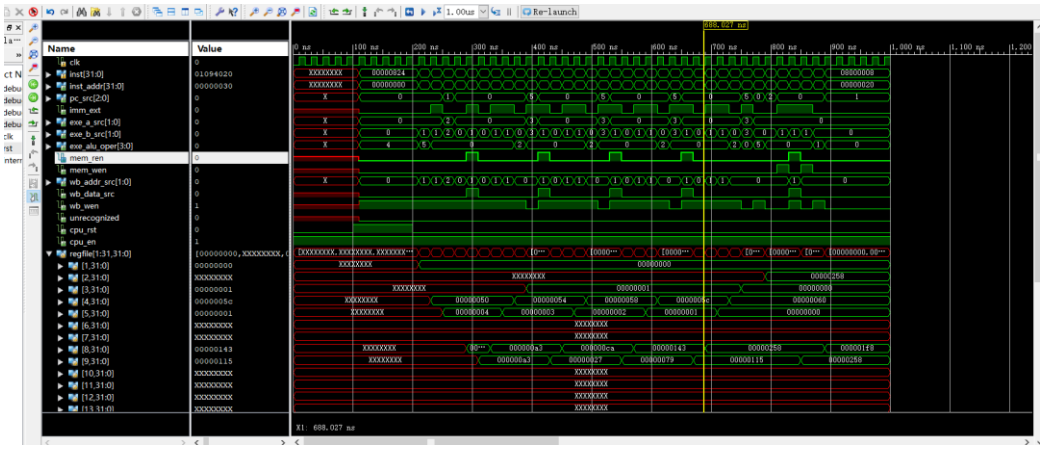


进入 loop 中，可见仿真中执行地址为 00000040 都 1460fffa 指令后跳转回地址为 0000002c 的 8c890000 指令，同时$5 寄存器的值在-1，说明 loop 执行中。



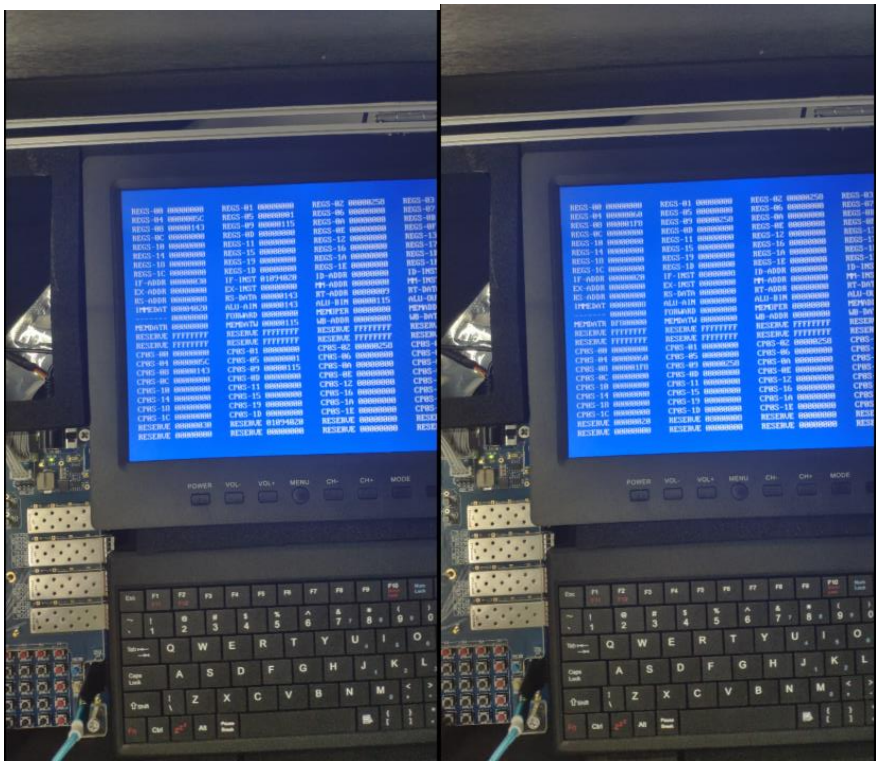
如图，当$5 寄存器值为 0 后，执行 1460fffa 后不再跳转回 loop 地址，而是转到 00000044

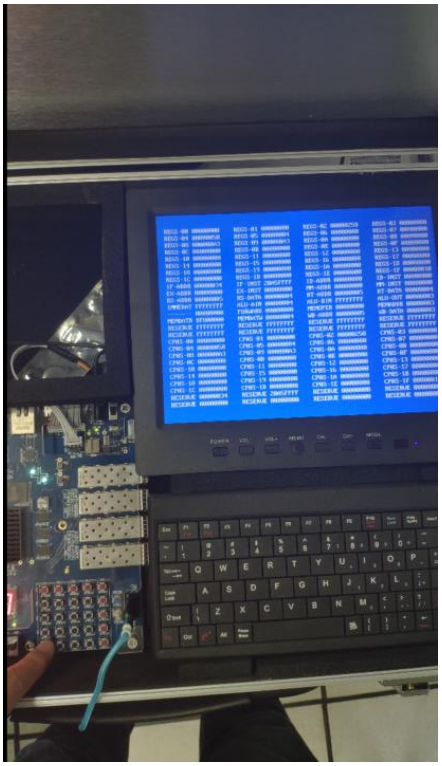
地址指令，证明 bne 判断逻辑正确，程序已挑出循环，随后执行 r 类型指令，观察$2，$8，$9 寄存器值可见 lw，sw 指令执行正确，最后执行 03e00008，jr 指令，停止程序。



宏观观察仿真如图，符合指令集计算结果。

下载验证





如图所示，验证最后执行结果，$9 寄存器值为 258，证明指令集计算正确，单周期 cpu 功能正确实现。

总结心得

本次实验作为计算机体系结构的第一次实验，旨在复习单周期 CPU，其中只需要设计 CPU controller 和 datapath 即可，其中 bne 指令的编写为本次的难点，在设计时我们也遇到了一定的麻烦，然后通过不断的仿真分析，最终成功修改错误代码，完成验证。这说明在设计，编写过程中不断的仿真的重要性。