浙 江 大 學

本科生实验报告

课程名称 计算机体系结构

姓　　名 吴同

学　　院 计算机科学与技术学院

专　　业 计算机科学与技术

学　　号 **3170104848**

指导教师 陈文智

浙江大学实验报告

专业：计算机科学与技术
姓名：吴同
学号：3170104848
日期：2019 年 12 月 11 日
地点：曹西 301

课程名称：计算机体系结构　指导老师：陈文智　电子邮件：wutongcs@zju.edu.cn
实验名称：支持中断的流水线 CPU　实验类型：综合型　同组同学：徐欣苑

一、 实验目的和要求

1. 实验目的

- 理解 CPU 中断的原理和执行过程
- 理解 CP0 协处理器的功能
- 掌握支持中断的流水线 CPU 的设计方法
- 掌握支持中断的流水线 CPU 的程序验证方法

2. 实验要求

- 设计支持中断的流水线 CPU 的数据通路、控制器和 CP0
- 使用程序验证 CPU 并观察程序的执行

二、 实验内容和原理

本实验中实现的 CP0 协处理器负责响应和处理中断。CP0 中有两个寄存器，一个作为基寄存器，一个作为程序计数器。当中断信号到达时，协处理器向控制器发出信号，停掉正在进行的译码过程，将 CP0 的基寄存器的地址送入流水线的程序计数器。

本实验实现三条与 CP0 有关的指令：

- mtc0：将普通寄存器的值写入 CP0 的寄存器

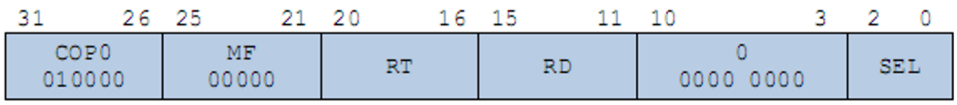

图 1: mfc0 指令格式

- mfc0：从 CP0 的寄存器读数据，写入普通寄存器中

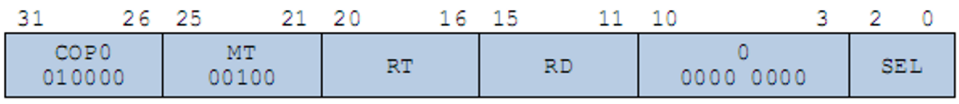

图 2: mtc0 指令格式

- eret：跳转到 CP0 的程序计数器存储的地址

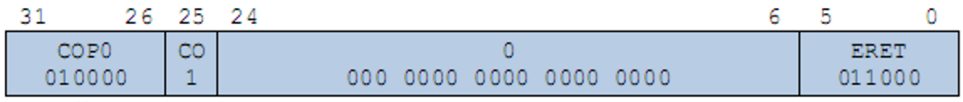

图 3: eret 指令格式

增加 CP0 后的数据通路图如下：

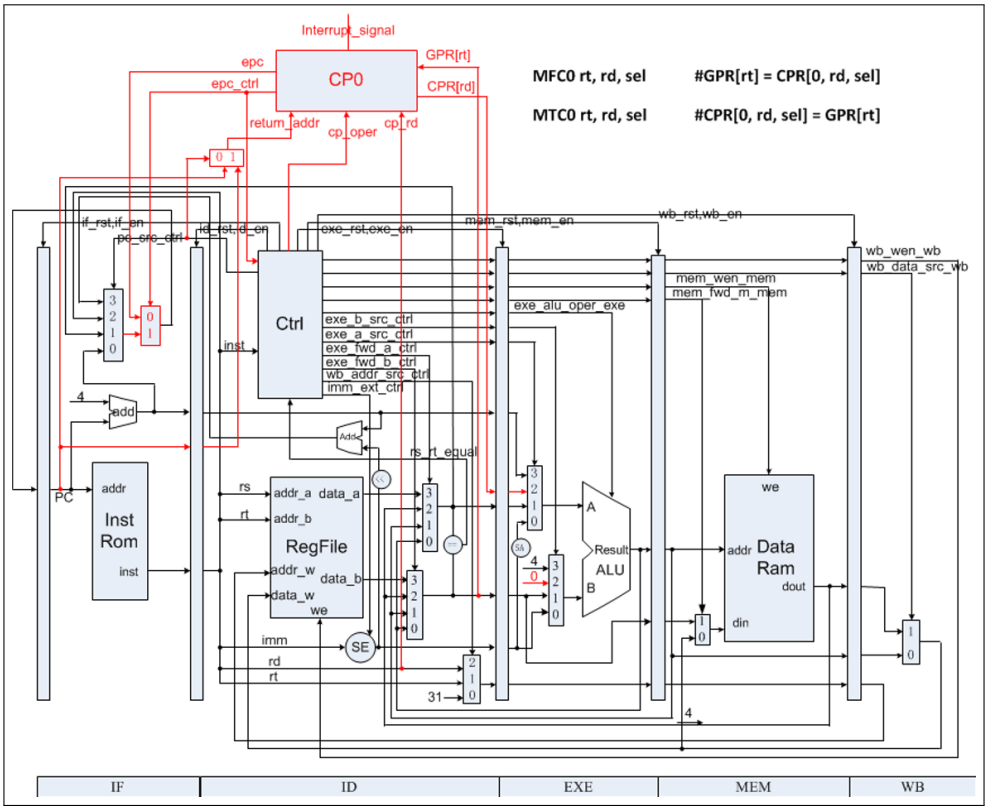

图 4: 增加 CP0 后的数据通路图

三、 实验过程记录

1. 设计 CP0

补全课件所给的框架内的代码：

```
always @(posedge clk) begin
   if (rst)
      ehbr <= 0;
   else if (oper == EXE_CP_STORE && addr_w == CP0_EHBR)
      ehbr <= data_w;
end

always @(posedge clk) begin
   if (rst || ir || oper == EXE_CP0_ERET)
      epcr <= 0;
   else if (oper == EXE_CP_STORE && addr_w == CP0_EPCR)
```

```
        epcr <= data_w;
    end

    always @(*) begin
        jump_en = 0;
        jump_addr = 0;
        eret = 0;
        if (rst) begin
            jump_en = 1;
        end
        else if (ir) begin
            jump_en = 1;
            jump_addr = {ehbr[31:2], 2'b00};
        end
        else if (oper == EXE_CP0_ERET) begin
            jump_en = 1;
            eret = 1;
            jump_addr = {epcr[31:2], 2'b00};
        end
    end

    always @(*) begin
        case (addr_r)
            CP0_EPCR: data_r = epcr;
            CP0_EHBR: data_r = ehbr;
            default: data_r = 0;
        endcase
    end
```

2. 修改控制器

三条与 CP0 有关的指令的译码：

```
    INST_CP0: begin
        if (inst[25]) begin
            case (inst[5:0])
                CP0_CO_ERET: begin
                    cp_oper = EXE_CP0_ERET;
                end
                default: begin
                    unrecognized = 1;
                end
            endcase
        end
        else begin
            case (inst[24:21])
                CP_FUNC_MF: begin
                    exe_alu_oper = EXE_ALU_ADD;
                    exe_a_src = EXE_A_CP0;
                    exe_b_src = EXE_B_CP0;
                    wb_addr_src = WB_ADDR_RT;
                    wb_data_src = WB_DATA_ALU;
                    wb_wen = 1;
```

```
            end
            CP_FUNC_MT: begin
               cp_oper = EXE_CP_STORE;
               rt_used = 1;
            end
            default: begin
               unrecognized = 1;
            end
         endcase
      end
   end
```

3. 修改数据通路

IF 阶段修改指令的跳转逻辑：

```
   always @(posedge clk) begin
      if (if_rst) begin
         inst_addr <= 0;
      end
      else if (jump_en) begin
         inst_addr <= jump_addr;
      end
      else if (if_en) begin
         case (pc_src_ctrl)
            PC_NEXT: inst_addr <= inst_addr_next;
            PC_JUMP: inst_addr <= {inst_addr_id[31:28],
                inst_data_id[25:0], 2'b0};
            PC_JR: inst_addr <= data_rs_modified;
            PC_BEQ: inst_addr <= inst_addr_next_id + {data_imm[29:0],
                2'b0};
         endcase
      end
   end
```

ID 阶段增加 CP0 的寄存器的通路：

```
   assign
      addr2cp0 = addr_rd,
      data2cp0 = data_rt_modified;
   assign
      ret_addr = (pc_src_ctrl == PC_NEXT) ? inst_addr : inst_addr_id;
```

EXE 阶段添加多路选择器的输入：

```
   always @(*) begin
      opa_exe = data_rs_exe;
      opb_exe = data_rt_exe;
      case (exe_a_src_exe)
         EXE_A_RS: opa_exe = data_rs_exe;
         EXE_A_SA: opa_exe = {27'b0, data_imm_exe[10:6]};
         EXE_A_LINK: opa_exe = inst_addr_next_exe;
         EXE_A_CP0: opa_exe = cp02data_exe;
```

```
    endcase
    case (exe_b_src_exe)
       EXE_B_RT: opb_exe = data_rt_exe;
       EXE_B_IMM: opb_exe = data_imm_exe;
       EXE_B_LINK: opb_exe = 32'h4; // linked address is the next one
           of the delay slot
       EXE_B_CP0: opb_exe = 32'h0;
    endcase
  end
```

四、 实验结果分析

本次实验使用的测试程序如下：

```
    lui $1, 0
    addiu $1, $1, 0x20
    mtc0 $1, $3
    add $2, $0, $0
    add $3, $0, $0
    addi $2, $2, 1
    j 0x14
    nop
    mfc0 $4, $2
    addi $3, $3, 1
    eret
    nop
```

完成 CPU 的设计后，修改仿真代码，在适当时候给出中断信号。收到中断前，2 号寄存器的值持续加 1. 收到中断信号后，CPU 开始执行中断代码，3 号寄存器的值加 1。

在本实验，如果 CP0 正在处理中断，则不会接收新到达的中断。仿真中对这种情况进行了模拟，实现了预期的效果。

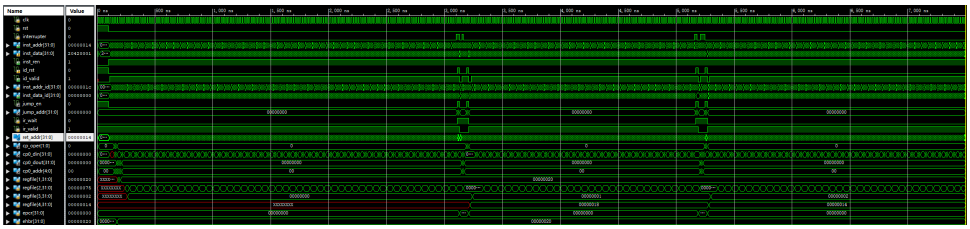

图 5: 仿真结果图（7500ns）

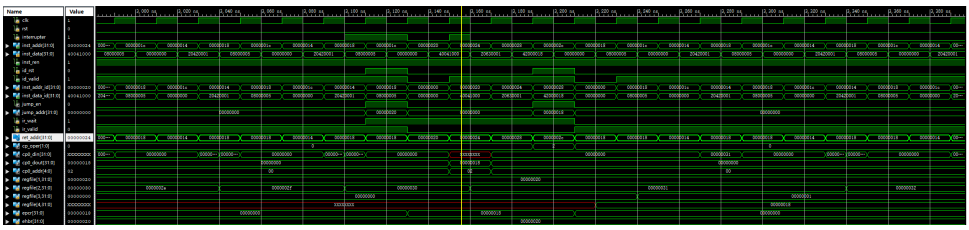

图 6: 第一次响应中断

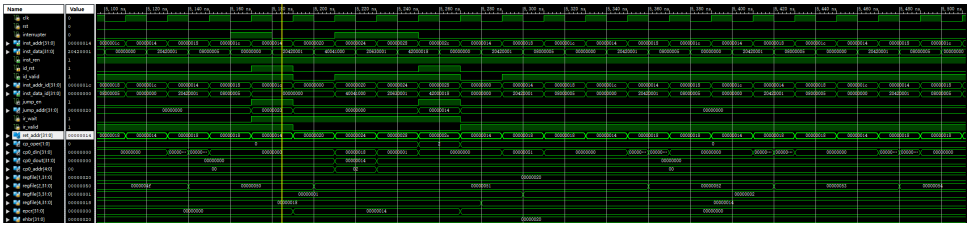

图 7: 第二次响应中断

将 CPU 与外设综合，下载到 FPGA 开发板中，实验结果符合预期。

五、 讨论与心得

本实验实现了简易的 CP0 协处理器。通过这次实验，我对中断的执行有了更加清晰的认识，同时对于操作系统的理解也更加深入。

在开始实验时，我不是很了解中断的实现过程，通过观察课件上的仿真图，我明确了 CP0 的功能，并根据所给的代码框架，完成了实验。