成绩评定日期：

实验报告成绩：

2024～2025学年秋季学期

**《计算机系统》必修课**

课程实验报告



班级：人工智能2201

组长：张一航

组员：王梓豪 雷鹏

报告日期：2024.12.11

目录

[1 实验概述 3](#_Toc186914457)

[1.1 小组成员及任务分工 3](#_Toc186914458)

[1.2 任务概述 3](#_Toc186914459)

[1.3 运行环境 3](#_Toc186914460)

[2 代码说明 4](#_Toc186914461)

[2.1 IF段 4](#_Toc186914462)

[2.2 ID段 5](#_Toc186914463)

[2.3 EX段 10](#_Toc186914464)

[2.4 MEM段 11](#_Toc186914465)

[2.5 WB段 12](#_Toc186914466)

[2.6 STALL CTRL段 13](#_Toc186914467)

# 1 实验概述

## 小组成员及任务分工

|  |  |  |
| --- | --- | --- |
| 姓名 | 任务分工 | 任务量占比 |
| 张一航 | 负责IF和ID模块，新增了数据前递和数据冒险的处理，实现了暂停逻辑，支持多种指令类型，实现了支持多种分支跳转指令、支持多种访存指令 | 40% |
| 王梓豪 | 主要负责EX和MEM模块，实现高效准确的ALU操作和地址计算。集成并优化乘法和除法模块，确保复杂运算的正确性和效率。 | 40% |
| 雷鹏 | 主要负责WB和CRTL模块 | 20% |

## 任务概述

本次实验要求完成一个五级流水的CPU设计， 需在框架代码的基础上补充逻辑指令、算数指令、跳转指令及数据相关的处理。

图示

描述已自动生成IF.v，ID.v，EX.v，MEM.v，WB.v文件对应五级流水线中的相关部分，lib文件夹下为提供的库文件，CTRL文件控制流水线暂停。

## 运行环境

CG服务器相关环境：vivado2019版本

# 代码说明

## IF段

该段的输入输出总线定义如下方图片及表格所示：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 接口名 | 宽度 | 输入/输出 | 作用 |
| 1 | clk | 1 | 输入 | 时钟信号 |
| 2 | rst | 1 | 输入 | 复位信号 |
| 3 | stall | 6 | 输入 | 暂停信号，控制指令是否暂停 |
| 4 | br\_bus | 33 | 输入 | 分支跳转信号，控制延迟槽是否跳转 |
| 5 | if\_to\_id\_bus | 33 | 输出 | IF段到ID段的数据总线 |
| 6 | inst\_sram\_en  inst\_sram\_en  \_en | 1 | 输出 | 读写使能信号 |
| 7 | inst\_sram\_wen | 4 | 输出 | 写使能信号 |
| 8 | inst\_sram\_addr | 32 | 输出 | 存放指令寄存器的地址 |
| 9 | inst\_sram\_wdata | 32 | 输出 | 存放指令寄存器的数据 |

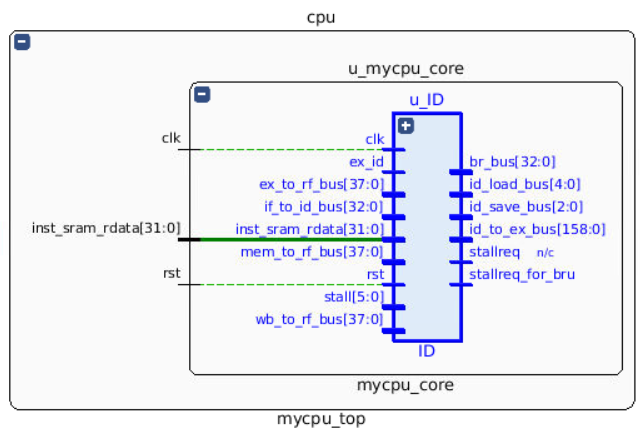
图示

描述已自动生成

这一阶段相对简单，没有太多需要改动的地方，大部分时候负责的工作是从PC+4的地址处获取下一条指令。只需要注意几个跳转指令需要跳到对应的地址读取指令。

## ID段

该段的输入输出总线定义如下方图片及表格所示：



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 接口名 | 宽度 | 输入/输出 | 作用 |
| 1 | clk | 1 | 输入 | 时钟信号 |
| 2 | rst | 1 | 输入 | 复位信号 |
| 3 | stall | 6 | 输入 | 暂停信号，控制指令是否暂停 |
| 4 | stallreq | 1 | 输出 | 暂停请求信号 |
| 5 | if\_to\_id\_bus | 33 | 输入 | IF段到ID段的数据总线 |
| 6 | inst\_sram\_rdata | 1 | 输入 | 读写使能信号 |
| 7 | ex\_id | 1 | 输入 | 写使能信号 |
| 8 | wb\_to\_rf\_bus | 38 | 输入 | WB段存放进寄存器的数据 |
| 9 | ex\_to\_rf\_bus | 38 | 输入 | EX段存放进寄存器的数据 |
| 10 | mem\_to\_rf\_bus | 38 | 输入 | MEM段存放进寄存器的数据 |
| 11 | ex\_hi\_lo\_bus | 66 | 输入 | EX段存放进hilo寄存器的数据的总线 |
| 12 | id\_hi\_lo\_bus | 72 | 输出 | ID段存放进hilo寄存器的数据的总线 |
| 13 | id\_load\_bus | 5 | 输出 | ID段执行load命令的数据总线 |
| 14 | id\_save\_bus | 3 | 输出 | ID段执行save命令的数据总线 |
| 15 | stallreq\_for\_bru | 1 | 输出 | 执行load命令时的暂停请求 |
| 16 | id\_to\_ex\_bus | 159 | 输出 | ID段到EX段的数据总线 |
| 17 | br\_bus | 33 | 输出 | 分支跳转信号，控制延迟槽是否跳转 |

ID段的主要功能是指令解码。

日程表

描述已自动生成

根据32位MIPS指令集类型可获取译码结果。根据I、J、R型指令对应部分读取寄存器文件中地址为 rs（inst[25;21]）以及地址为rt(inst[20:16]的通用寄存器,得到rdata1以及rdata2，并且通过判断是否发生数据相关，从而更改rdata1以及 rdata2的值。（由于数据相关，在这里，传给下一段的寄存器数据是ndata而非rdata）

    assign ndata1 = ((ex\_rf\_we && rs == ex\_rf\_waddr) ? ex\_rf\_wdata : 32'b0) |

                   ((!(ex\_rf\_we && rs == ex\_rf\_waddr) && (mem\_rf\_we && rs == mem\_rf\_waddr)) ? mem\_rf\_wdata : 32'b0) |

                   ((!(ex\_rf\_we && rs == ex\_rf\_waddr) && !(mem\_rf\_we && rs == mem\_rf\_waddr) && (wb\_rf\_we && rs == wb\_rf\_waddr)) ? wb\_rf\_wdata : 32'b0) |

                   (((ex\_rf\_we && rs == ex\_rf\_waddr) || (mem\_rf\_we && rs == mem\_rf\_waddr) || (wb\_rf\_we && rs == wb\_rf\_waddr)) ? 32'b0 : rdata1);

    assign ndata2 = ((ex\_rf\_we && rt == ex\_rf\_waddr) ? ex\_rf\_wdata : 32'b0) |

                   ((!(ex\_rf\_we && rt == ex\_rf\_waddr) && (mem\_rf\_we && rt == mem\_rf\_waddr)) ? mem\_rf\_wdata : 32'b0) |

                   ((!(ex\_rf\_we && rt == ex\_rf\_waddr) && !(mem\_rf\_we && rt == mem\_rf\_waddr) && (wb\_rf\_we && rt == wb\_rf\_waddr)) ? wb\_rf\_wdata : 32'b0) |

                   (((ex\_rf\_we && rt == ex\_rf\_waddr) || (mem\_rf\_we && rt == mem\_rf\_waddr) || (wb\_rf\_we && rt == wb\_rf\_waddr)) ? 32'b0 : rdata2);

同时根据opcode和func部分分析要执行的运算，给对应的ALU的相关控制逻辑赋值，其中，0表示该条指令不采用该 ALU，1 表示采用该ALU，同时将所有的 ALU 标识符组合起来成为 alu\_op，alu\_op为十二位宽，代表 16 种不同的 ALU，并且作为传入 EX 段的一部分。要写入的目的寄存器。rf\_we 代表写使能信号，表示该条指令是否用写入通用寄存器，sel\_rf\_dst[0]表示该指令要将计算结果写入 rd 通用寄存器，sel\_rf\_dst[1]表示该指令要将计算结果写入 rt 通用寄存器,sel\_rf\_dst[2]表示该指令要将计算结果写入 31 号通用寄存器。rf\_waddr 表示要该条指令的计算结果要写入的通用寄存器的地址，data\_ram\_en表示该条指令是否要与内存中取值或者写入值，如果该条指令要从内存中取值或者写入值，那么它将被赋值为 1’b1,data\_ram\_wen 为四位宽，表示该条指令是否要写入寄存器，如果该条指令要将计算结果的第几个字节写入寄存器，那么对应位置的值设为 1。

用sel\_alu\_src1和sel\_alu\_src2来判断操作数来源，第一个操作数有三种来源，第二个操作数有四种来源，通过区分不同的指令进而分辨不同的操作数的来源。

传值用ID段得到的数据，分别给id\_to\_ex\_bus和br\_bus赋值，其中br\_bus是传给IF段的用于传输跳转指令的判断信号和跳转的地址。

    // rs to reg1

    assign sel\_alu\_src1[0] =    inst\_lw | inst\_sw |inst\_lb| inst\_lbu  | inst\_lh | inst\_lhu| inst\_sb | inst\_sh |

                                inst\_ori | inst\_addiu | inst\_or | inst\_xor | inst\_and  | inst\_andi| inst\_nor | inst\_xori |

                                inst\_sub | inst\_subu | inst\_add | inst\_addi | inst\_addu |

                                inst\_jr | inst\_bgezal | inst\_bltzal |

                                inst\_slti | inst\_or | inst\_srav | inst\_sltu | inst\_slt | inst\_sltiu | inst\_sllv| inst\_srlv

                                ;

    // pc to reg1

    assign sel\_alu\_src1[1] = inst\_jal | inst\_jalr | inst\_bltzal | inst\_bgezal;

    // sa\_zero\_extend to reg1

    assign sel\_alu\_src1[2] = inst\_sll | inst\_sra | inst\_srl;

    // rt to reg2

    assign sel\_alu\_src2[0] =    inst\_sub | inst\_subu | inst\_addu | inst\_sll | inst\_or | inst\_xor | inst\_sra | inst\_srl |

                                inst\_srlv | inst\_sllv| inst\_sra | inst\_srav| inst\_sltu | inst\_slt  | inst\_add | inst\_and| inst\_nor ;

                                // inst\_div | inst\_divu |inst\_mult | inst\_multu;

    // imm\_sign\_extend to reg2

    assign sel\_alu\_src2[1] =    inst\_lui | inst\_addiu | inst\_lw | inst\_sw | inst\_slti| inst\_sltiu | inst\_addi |

                                inst\_lb  | inst\_lbu   | inst\_lh  | inst\_lhu | inst\_sh | inst\_sb;

    // 32'b8 to reg2

    assign sel\_alu\_src2[2] = inst\_jal | inst\_jalr | inst\_bgezal | inst\_bltzal;

    // imm\_zero\_extend to reg2

    assign sel\_alu\_src2[3] = inst\_ori | inst\_andi | inst\_xori;

跳转指令：先用br\_e表示这条指令是否为跳转指令，用rs\_ge\_z表示是否满足 rdata1的值大于等于 0，用rs\_le\_z表示是否满足rdata1的值小于等于0，用rs\_lt\_z表示是否满足 rdata1的值小于 0，rs\_eq\_rt表示是否满足rdata1是否等于radta2的值。br\_addr 表示跳转后的地址，根据不同的指令对地址做不同的计算，并将结果赋给br\_addr。

    assign br\_e = inst\_beq & rs\_eq\_rt | inst\_j | inst\_jalr | | inst\_jr | inst\_jal | inst\_bne & ~rs\_eq\_rt |

                  inst\_bgez & rs\_ge\_z | inst\_bgtz & rs\_gt\_z |inst\_blez & rs\_le\_z | inst\_bltz & rs\_lt\_z |

                  inst\_bgezal & rs\_ge\_z | inst\_bltzal & rs\_lt\_z ;

    assign br\_addr =

                        (inst\_beq       ? (pc\_plus\_4 + {{14{inst[15]}},inst[15:0],2'b0}) : 32'b0)   |

                        (inst\_jr        ? ndata1 : 32'b0)                                           |

                        (inst\_jal       ? {pc\_plus\_4[31:28],instr\_index,2'b0} : 32'b0)              |

                        (inst\_j         ? ({ pc\_plus\_4[31:28],inst[25:0],2'b0}) : 32'b0)            |

                        (inst\_jalr      ? ndata1:32'b0)                                             |

                        (inst\_bne       ? (pc\_plus\_4 + {{14{inst[15]}},inst[15:0],2'b0}) : 32'b0)   |

                        (inst\_bgez      ? (pc\_plus\_4 + {{14{inst[15]}},inst[15:0],2'b0}) : 32'b0)   |

                        (inst\_bgtz      ? (pc\_plus\_4 + {{14{inst[15]}},inst[15:0],2'b0}) : 32'b0)   |

                        (inst\_blez      ? (pc\_plus\_4 + {{14{inst[15]}},inst[15:0],2'b0}) : 32'b0)   |

                        (inst\_bltz      ? (pc\_plus\_4 + {{14{inst[15]}},inst[15:0],2'b0}) : 32'b0)   |

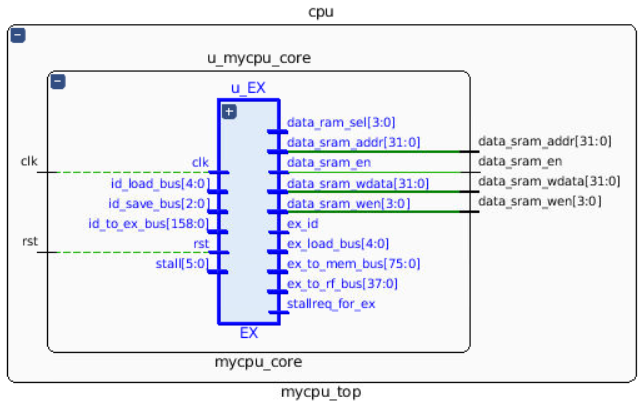
                        (inst\_bgezal    ? (pc\_plus\_4 + {{14{inst[15]}},inst[15:0],2'b0}) : 32'b0)   |

                        (inst\_bltzal    ? (pc\_plus\_4 + {{14{inst[15]}},inst[15:0],2'b0}) : 32'b0)

                        ;

## EX段

该段的输入输出总线定义如下方图片及表格所示：

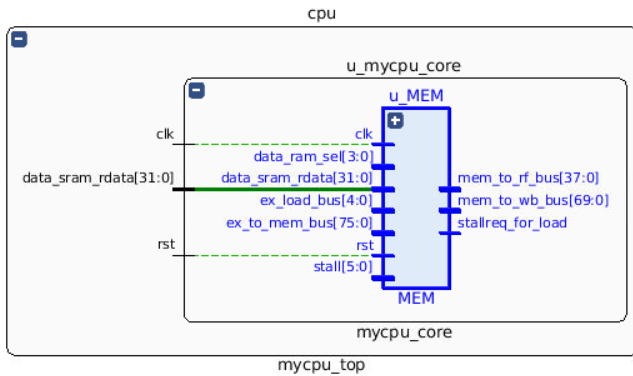


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 接口名 | 宽度 | 输入/输出 | 作用 |
| 1 | clk | 1 | 输入 | 时钟信号 |
| 2 | rst | 1 | 输入 | 复位信号 |
| 3 | stall | 6 | 输入 | 控制暂停信号 |
| 4 | id\_to\_ex\_bus | 169 | 输入 | ID 段传给 EX 段的数据 |
| 5 | id\_load\_bus | 5 | 输入 | ID段传递读的数据 |
| 6 | id\_save\_bus | 3 | 输入 | ID段传递写的数据 |
| 7 | ex\_to\_mem\_bus | 80 | 输出 | EX 段传给 MEM 段的数据 |
| 8 | ex\_to\_rf\_bus | 38 | 输出 | EX 段传给regfile段的数据 |
| 9 | ex\_hi\_lo\_bus | 66 | 输出 | EX 段传给hilo段的数据 |
| 10 | stallreq\_for\_ex | 1 | 输出 | 对EX段的stall请求 |
| 11 | data\_sram\_en | 1 | 输出 | 内存数据的读写使能信号 |
| 12 | data\_sram\_wen | 4 | 输出 | 内存数据的写使能信号 |
| 13 | data\_sram\_addr | 32 | 输出 | 内存数据存放的地址 |
| 14 | data\_sram\_wdata | 32 | 输出 | 要写入内存的数据 |
| 15 | ex\_id | 38 | 输出 | EX 段传给 ID 段的数据 |
| 16 | data\_ram\_sel | 4 | 输出 | 内存数据的选择信号 |
| 17 | ex\_load\_bus | 5 | 输出 | EX 段读取的数据 |

EX段的任务也比较简单，主要负责计算从ID段传来的各类数据（存储于寄存器中，包括地址、数据等）。ALU已在库文件中提供，只需设置相关控制信号即可。

## MEM段

该段的输入输出总线定义如下方图片及表格所示：



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 接口名 | 宽度 | 输入/输出 | 作用 |
| 1 | clk | 1 | 输入 | 时钟信号 |
| 2 | rst | 1 | 输入 | 复位信号 |
| 3 | stall | 6 | 输入 | 控制暂停信号 |
| 4 | ex\_to\_mem\_bus | 80 | 输入 | EX 传给 MEM 段的数据 |
| 5 | data\_sram\_rdata | 32 | 输入 | 从内存中读出来要写入寄存器的值 |
| 6 | data\_ram\_sel | 4 | 输入 | 内存数据的选择信号 |
| 7 | ex\_load\_bus | 5 | 输入 | EX 段读取的数据 |
| 8 | stallreq\_for\_load | 1 | 输出 | 对EX段的stall请求 |
| 9 | mem\_to\_wb\_bus | 70 | 输出 | MEM 传给 WB 段的数据 |
| 10 | mem\_to\_rf\_bus | 38 | 输出 | MEM段传给regfile段的数据 |

主要任务是访存取数，核心逻辑如下：

    assign b\_data = data\_ram\_sel[3] ? data\_sram\_rdata[31:24] :

                    data\_ram\_sel[2] ? data\_sram\_rdata[23:16] :

                    data\_ram\_sel[1] ? data\_sram\_rdata[15: 8] :

                    data\_ram\_sel[0] ? data\_sram\_rdata[ 7: 0] : 8'b0;

    assign h\_data = data\_ram\_sel[2] ? data\_sram\_rdata[31:16] :

                    data\_ram\_sel[0] ? data\_sram\_rdata[15: 0] : 16'b0;

    assign w\_data = data\_sram\_rdata;

    assign mem\_result = inst\_lb     ? {{24{b\_data[7]}},b\_data} :

                        inst\_lbu    ? {{24{1'b0}},b\_data} :

                        inst\_lh     ? {{16{h\_data[15]}},h\_data} :

                        inst\_lhu    ? {{16{1'b0}},h\_data} :

                        inst\_lw     ? w\_data : 32'b0;

    assign rf\_wdata = sel\_rf\_res & data\_ram\_en ? mem\_result : ex\_result;

## WB段

该段的输入输出总线定义如下方图片及表格所示：

图片包含 文本

描述已自动生成

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 接口名 | 宽度 | 输入/输出 | 作用 |
| 1 | clk | 1 | 输入 | 时钟信号 |
| 2 | rst | 1 | 输入 | 复位信号 |
| 3 | stall | 6 | 输入 | 控制暂停信号 |
| 4 | mem\_to\_wb\_bus | 70 | 输入 | MEM 传给 WB 的数据 |
| 5 | wb\_to\_rf\_bus | 38 | 输出 | WB 传给 rf 的数据 |
| 6 | debug\_wb\_pc | 32 | 输出 | 用来 debug 的 pc 值 |
| 7 | debug\_wb\_rf\_wen | 4 | 输出 | 用来 debug 的写使能信号 |
| 8 | debug\_wb\_rf\_wnum | 5 | 输出 | 用来 debug 的写寄存器地址 |
| 9 | debug\_wb\_rf\_wdata | 32 | 输出 | 用来 debug 的写寄存器数据 |

该段任务是将数据写回内存，没有什么需要改动的地方。

核心逻辑：提取从MEM传入的相关控制信号，然后将写回信号发送给寄存器堆完成内存写入操作：

    assign {

        wb\_pc,

        rf\_we,

        rf\_waddr,

        rf\_wdata

    } = mem\_to\_wb\_bus\_r;

    assign wb\_to\_rf\_bus = {

        rf\_we,

        rf\_waddr,

        rf\_wdata

    };

## STALL CTRL段

该段的输入输出总线定义如下方图片及表格所示：

图示

中度可信度描述已自动生成

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 接口名 | 宽度 | 输入/输出 | 作用 |
| 1 | clk | 1 | 输入 | 时钟信号 |
| 2 | stallreq\_for\_ex | 1 | 输入 | 执行阶段的指令是否请求流水线暂停 |
| 3 | stallreq\_for\_bru | 5 | 输入 | Load命令是否请求流水线暂停 |
| 4 | stall | 6 | 输出 | 暂停信号 |

Stall信号共6位，每一位分别代表流水线中某一段的暂停信号。stall[0]表示取指地址PC是否保持不变，为1表示保持不变；stall[1]表示流水线取指阶段是否暂停，为1表示暂停；stall[2]表示流水线译码阶段是否暂停，为1表示暂停；stall[3]表示流水线执行阶段是否暂停，为1表示暂停。stall[4]表示流水线访存阶段是否暂停，为1表示暂停；stall[5]表示流水线回写阶段是否暂停，为1表示暂停。

核心逻辑：

    always @ (\*) begin

        if (rst) begin

            stall = `StallBus'b0;

        end

        //stallreq\_for\_ex, stallreq\_for\_bru, stallreq\_for\_load

        else if (stallreq\_for\_ex) begin

            stall = `StallBus'b001111;

        end

        //

        else if (stallreq\_for\_bru) begin

            stall = `StallBus'b000111;

        end

        else if (stallreq\_for\_load) begin

            stall = `StallBus'b000011;

        end

        else begin

            stall = `StallBus'b0;

        end

    end