实验报告成绩：	成绩评定日期：

2022～2023 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 2001

组长：谭啸尘

组员：王建金 孙传真

报告日期：2023.1.2

目录

一、总体介绍

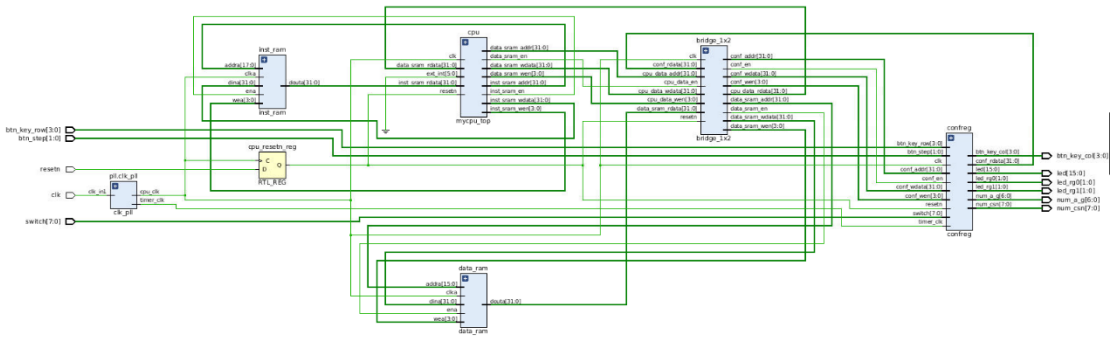
1.1 成员分工

谭啸尘：协助 43-51 号点，51—61 号点，报告整合

王建金：1-51 号点，数据相关的解决、load 相关问题的解决、hilo 寄存器的添加以及以上指令的添加。

孙传真：61-64 号点

1.2 总体设计



1.3 完成指令

subu、jal、jr、addu、bne、sll、or、lw、sw、xor、sltu、slt、slti、sltiu、j、add、addi、sub、and、andi、nor、xori、sllv、sra、srav、srl、srlv、begz、bgtz、blez、bltz、bltzal、bgezal、jalr、div、mflo、mfhi、mtlo、mthi、mul、mult、multu、divu、lb、lbu、lh、lhu

1.4 实验环境

VsCode 作为编辑器和 Vivado 作为仿真环境。

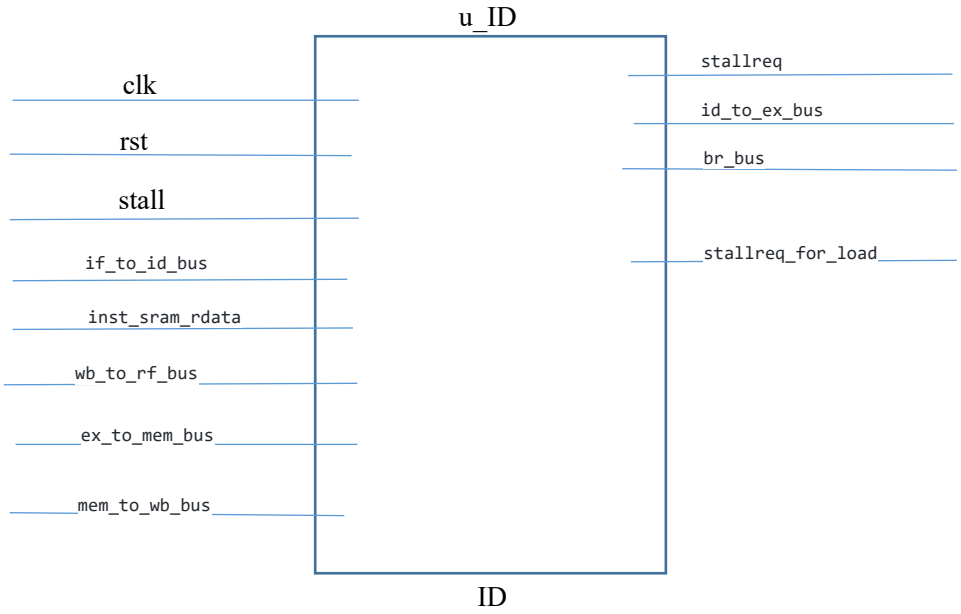
二.各流水段功能（整体）

2.1IF 段

未删增代码，模块功能为取指令，并控制指令延迟槽和跳转指令。

2.2ID 段

代码主要部分之一，模块功能为对指令进行译码，并将译码结果传给 EX 段，同时与寄存器进行交互，实现寄存器的读写，处理数据相关。模块接口如下：



接口	位宽	输入 or 输出	功能
clk	1	输入	时钟信号
rst	1	输入	复位信号
stall	6	输入	控制暂停信号
data_sram_rdata	32	输入	当前指令地址中储存的值
if_to_id_bus	33	输入	IF 段发给 ID 段的数据
wb_to_rf_bus	38	输入	WB 段传给 ID 段要写入寄存器的值
ex_to_mem_bus	147	输入	EX 段传给 MEM 段的数据，用来判断数据相关
mem_to_wb_bus	136	输入	MEM 段传给 WB 段的数据，用来判断数据相关
stallreq	1	输出	控制暂停
id_to_ex_bus	173	输出	ID 段传给 EX 段的数据
br_bus	33	输出	ID 段传给 IF 段的数据，用来判断下一条指令的地址
stallreq_for_load	1	输出	从 ID 段发出的暂停信号，用来判断 load 相关

功能模块说明：

（一）流水线暂停

```
always @ (posedge clk) begin
    if (rst) begin
        if_to_id_bus_r <= `IF_TO_ID_WD'b0;
        is_stop <= 1'b0;
    end
    // else if (flush) begin
    //     if_to_id_bus_r <= `IF_TO_ID_WD'b0;
    // end
    else if (stall[1]==`Stop && stall[2]==`NoStop) begin
        if_to_id_bus_r <= `IF_TO_ID_WD'b0;
        is_stop <= 1'b0;
    end
    else if (stall[1]==`NoStop) begin
        if_to_id_bus_r <= if_to_id_bus;
        is_stop <= 1'b0;
    end
    else if (stall[2]==`Stop) begin
        is_stop <= 1'b1;
    end
end

assign inst = is_stop ? inst : inst_sram_rdata;
```

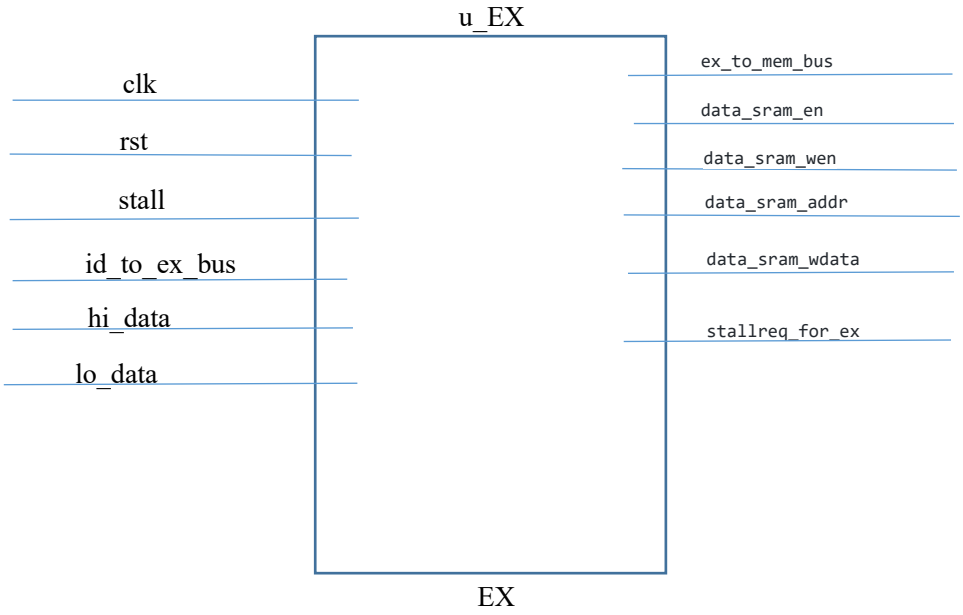
ID 段会收到来自 CTRL 模块的 stall 值，stall 值用来控制流水线的暂停。stall 值对应的部分为`NoStop 时，意味着没有流水线暂停，将 IF 段传给 ID 段的 if_to_id_bus 赋值给 if_to_id_bus_r，流水线正常运行。stall 值对应的部分是 `Stop 时，发生了访存冲突，需要读取的寄存器中的值还没有获得，需要在下一个周期才可以从内存中读取出来，无法通过 forwarding 技术解决，需要对流水线的 ID 段进行暂停一个周期，在下个周期获得到需要读取的值后再发给 ID 段。暂停时需要利用 is_stop 把当前时刻的 inst 值保存一个周期，下一个周期再使用当前周期的 inst 值，保证 ID 段和之后所有部分的指令的 pc 值和 inst 值是相互匹配的。

（二）译码

大多数译码工作初始代码已完成，需要添加的部分有定义新指令、激活对应指令变量、寄存器 1 与寄存器 2 的数据、分析执行的运算、是否写回以及写回地址、数据相关、id_to_ex_bus 打包、跳转指令。定义新指令格式为：wire inst_**。激活对应指令变量则需用对应 op 码（与扩展码）激活对应的已定义指令。reg1、reg2、op 运算、是否写回以及写回位置需要或上当前指令即可。数据相关需要来自 EX 段、MEM 段、WB 段的数据，判断使能信号与上寄存器是否相同判断是否发生了数据相关，进而改变 rdata1 与 rdata2。id_to_ex_bus 打包新增 hilo_op 与 mem_op，将已激活的指令变量向下传递。跳转指令有条件的指令需要与指令变量进行与运算的到跳转使能，根据指令要求计算出跳转地址，打包置 br_bus。

2.3EX 段

代码主要部分之一，模块功能为计算 ALU 的结果，根据当前指令，确定即将要写入内存的数据以及地址，或者下一步是否要从内存中读取值。模块接口如下：



接口	位宽	输入or输出	功能
clk	1	输入	时钟信号
rst	1	输入	复位信号
stall	6	输入	控制暂停信号
id_to_ex_bus	173	输入	ID 段传给 EX 段的数据
hi_data	32	输入	hi 寄存器的数据，高位计算结果。
lo_data	32	输入	lo 寄存器的数据，低位计算结果。
ex_to_mem_bus	147	输出	EX 段传给 MEM 段的数据
data_sram_en	1	输出	内存数据的读写使能信号
data_sram_wen	4	输出	内存数据的写使能信号
data_sram_addr	32	输出	内存数据存放的地址
data_sram_wdata	32	输出	要写入内存的数据
stallreq_for_ex	1	输出	EX 发出的是否暂停的信号

功能模块说明：

（一）alu 运算

此部分无代码删增，正常进行 rdata1 与 rdata2 的 op 运算，最后得到运算结果 alu_result。

（二）store 指令

根据 ID 段传输的 mem_op 内的 store 指令，根据指令变量判断是否改变内存数据的读写使能信号、写使能信号、内存数据的存放地址以及写入内存的数据。

（三）hilo 指令

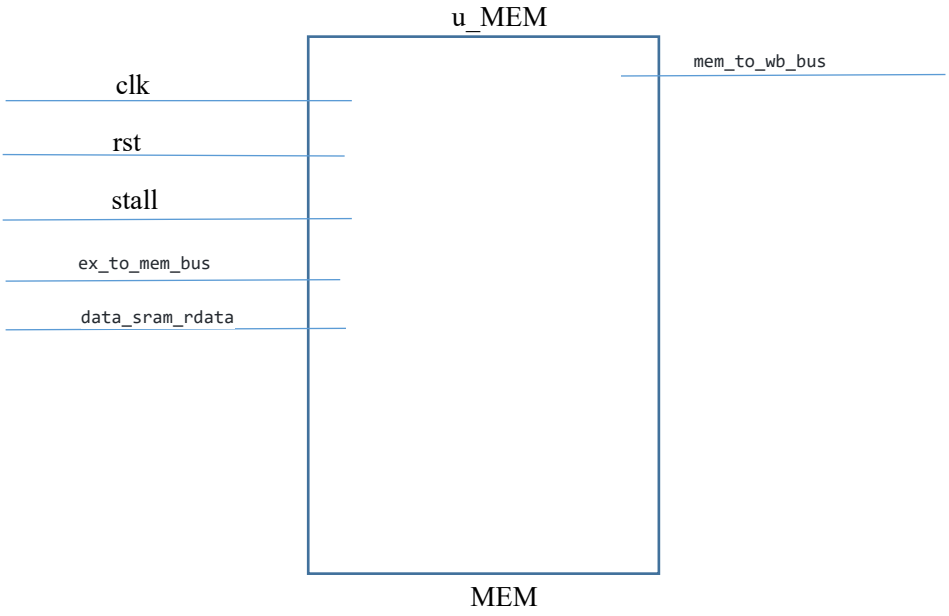
首先补全 u_mul 空缺部分，随后根据 ID 段传输的 hilo_op 内的 hilo 指令，判断乘除与写读，乘除计算结果高位写入 hi 寄存器，低位写入 lo 寄存器，根据指令对高低位的需求传入对应寄存器内的值。若读，将 result 改为当前部分的 result。最后将使能信号与数据一同打包，合并到 ex_to_mem_bus 中并向后传递。

（四）乘除部分

此部分只需补全 u_mul 空缺部分，mul_result 和 div_result 可以直接使用，计算结果为 64 位。

2.4MEM 段

代码主要部分之一，模块功能为读取内存中相应地址的值（其实是在 EX 到 MEM 的过程中读取的），根据当前指令，确定要写入寄存器的值。模块接口如下：



接口	位宽	输入 or 输出	功能
clk	1	输入	时钟信号
rst	1	输入	复位信号
stall	6	输入	控制暂停信号
ex_to_mem_bus	147	输入	EX 段传给 MEM 段的数据
data_sram_rdata	32	输入	从内存中读出来要写入寄存器的值
mem_to_wb_bus	136	输出	MEM 段传给 WB 段的数据

功能模块说明：

（一）load 指令

根据经过 EX 段传输的由 ID 段发出的 mem_op 内的 load 指令，根据指令变量判断是否改变 mem_result，并根据 sel_rf_res 判断 rf_wdata 赋值来自 mem_result 还是 ex_result。

（二）打包传递

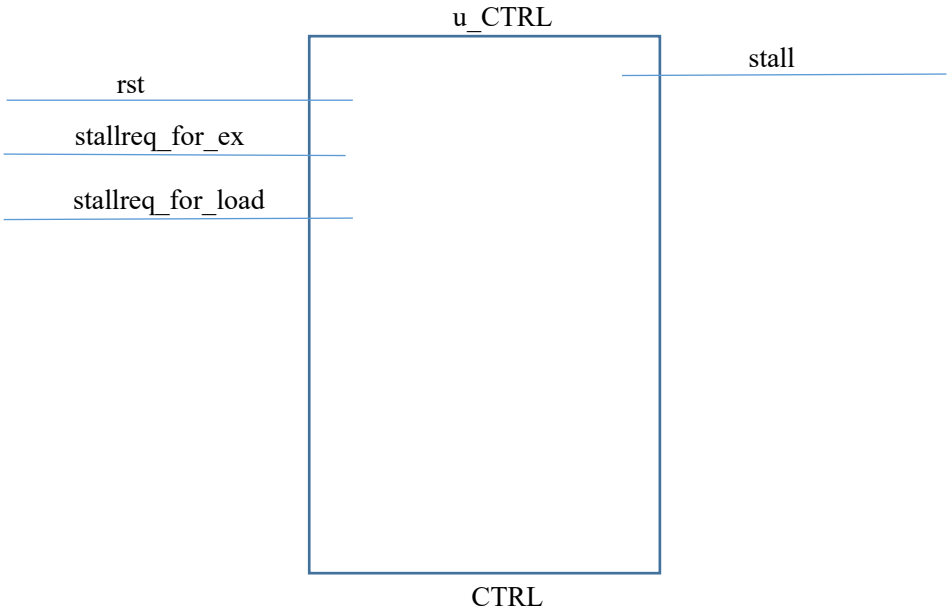
由 EX 传出进出的 hilo_bus 继续打包向下传递。

2.5WB 段

未删增代码，模块功能为将结果写入寄存器，这一段实际是在 ID 段调用 regfile 模块实现的。

2.6CTRL 模块

代码主要部分之一，模块功能为接收各段传递过来的流水线请求信号，从而控制流水线各阶段的运行。模块接口如下：



接口	位宽	输入 or 输出	功能
rst	1	输入	复位信号
stallreq_for_ex	1	输入	为了等待乘除而暂停
stallreq_for_load	1	输入	因 load 相关而暂停
stall	6	输出	暂停流水线控制信号

功能模块说明：

（一）输出暂停信号

stall[0]代表是否有暂停

stall[1]代表 if 段是否暂停

stall[2]代表 id 段是否暂停

stall[3]代表 ex 段是否暂停

stall[4]代表 mem 段是否暂停

stall[5]代表 wb 段是否暂停

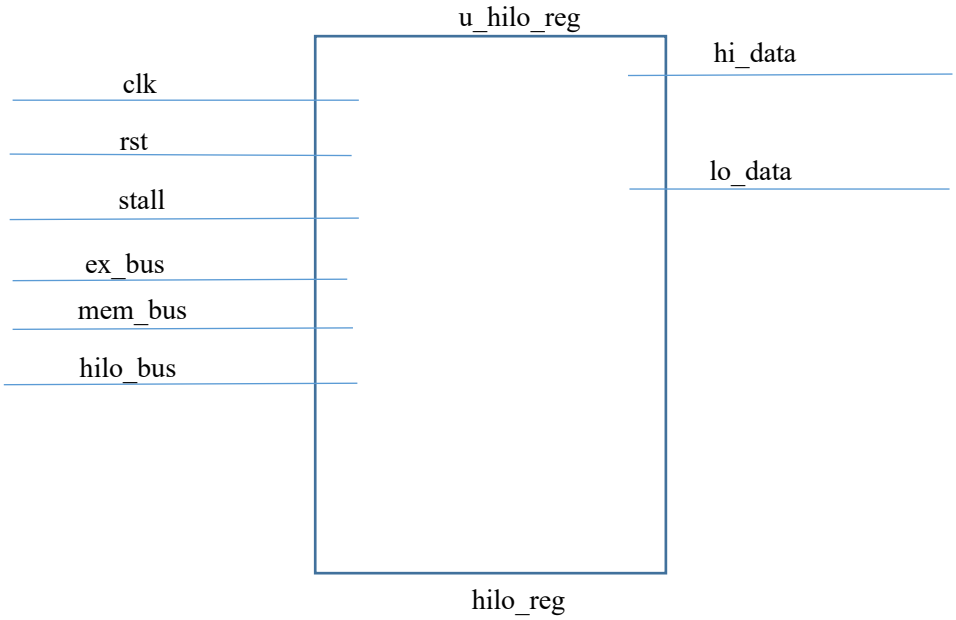
（二）等待乘除与 load 相关

```
else if(stallreq_for_load) begin
    stall = `StallBus'b00_0111;
end
else if(stallreq_for_ex) begin
    stall = `StallBus'b00_1111;
```

根据输入的信号来赋予指定的暂停信号。

2.7hilo_reg 模块（根据助教要求写在了 regfile.v 中）

代码主要部分之一，模块功能为按高 32 位与低 32 位分别保存 EX 段乘除的计算结果置 hi 与 lo 当中。模块接口如下：



接口	位宽	输入 or 输出	功能
clk	1	输入	时钟信号（无作用）
rst	1	输入	复位信号（无作用）
stall	1	输入	控制暂停信号（无作用）
ex_bus	66	输入	EX 段的 hilo_bus
mem_bus	66	输入	MEM 段的 hilo_bus
hilo_bus	66	输入	WB 段的 hilo_bus
hi_data	32	输出	hi 寄存器输出结果
lo_data	32	输出	lo 寄存器输出结果

功能模块说明：

（一）输出

```
assign hi_data = ex_hi_we ? ex_hi_in:
                 mem_hi_we? mem_hi_in:
                 hi_we    ? hi_in:
                 hi;
assign lo_data = ex_lo_we ? ex_lo_in:
                 mem_lo_we? mem_lo_in:
                 lo_we    ? lo_in:
                 lo;
```

为防止数据相关，EX 段、MEM 段与 WB 段的计算结果都要判断，hi 与 lo 为临时保存的 hilo 值的变量。

（二）输入

```
assign hi = hi_we ? hi_in : hi;
assign lo = lo_we ? lo_in : lo;
```

根据 hilo_bus 里的使能判断是否写入新的 hilo 值，否则不变。

三.感受与改进

3.1 谭啸尘部分

计算机系统的这次实验中，我通过 cg 实验平台和队友协作完成了 cpu 设计仿真，正如学长在 GitHub 中提到的那样，在人工智能相关知识之外了解到了 verilog 这样一门新的语言，并且在 Linux 系统下完成了该次实验，对 linux 操作系统也有了更深入的了解。相对于我们之前用 python、c、c++等语言完成的各项课程设计和实验，这次的实验无疑是更加抽象且复杂的。不过在同学和助教的耐心指导下，我成功理解了队友完成的部分，并且在队友完成的基础上继续跟进完成了后续的部分。不得不承认，纸上得来终觉浅，绝知此事要躬行，在课堂上完成了流水线相关知识的理论学习之后，完成实验依旧有不小的困难，这次试验让我更加深刻的了解到实践的重要性。而且我也从这次实验了解到另外一点，那就是不要急躁，有时候在对代码进行修改之后，跑通的点不仅没有增加，反而会有减少，有时候会怀疑自己改的是不是有问题从而陷入自我怀疑。而此时需要做的是相信自己，很有可能走在正确的道路上，不过道路是曲折的，继续前进就会迎来最终的胜利，跑通更多的测试点。

3.2 王建金部分

一开始实验时，面对陌生复杂的代码时不知所措。从 11 月 21 号开始到 26 号完成测试节点 1 花费了大量精力，首先是读懂已有代码，在课堂上学习的知识突然就活灵活现，流水线当中好多自己疏忽的地方、不解的地方就在眼前，逐渐拨云见日，理解了代码的整体框架，各部分联系与功能。根据助教发的内容、分享的文献我知道了自己的实验目标，助教学长的耐心指导令我感动，终于一点一点地完成了测试点 1。随后地测试节点便得心应手，都是前面测试点 1 的变式，只有 load 相关以及理解 hilo 寄存器是什么卡住了一段时间，最终过关斩将，完成了测试节点 51（其余测试节点由其他组员负责完成，我也参与了讨论与分享）。由于我开始着手实验的时间较早，有不少同学也会有问题与我讨论，在讨论当中教学相长，加深了自我对实验内容的进一步理解。

3.3 孙传真部分

刚开始实验时，对于完全陌生的代码感到无所适从，只感觉这类代码和以前所接触的编程思想有所差别。后来经过反复的阅读资料，以及和队友沟通交流，不懂就问，逐渐明确了实验目标。队友的进展相对更快，通过队友的指导，对代码有了一个整体上的理解。通过跟进队友的进度以及询问，逐渐理解 hilo 寄存器是什么，最后拿到分配到的任务时才能迅速着手于指令的添加，最终迅速接着队友的工作完成最后的工作。经过实验，理论与实践相结合，课堂上学的知识经过了实际检验，实验过程中的学习也弥补了课堂学习的部分欠缺疏忽之处，强化了对课堂知识的理解。

四、参考资料

1、雷思磊 著《自己动手写 CPU》 电子工业出版社
2、龙芯杯官方的参考文档