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

专业：人工智能

组长：李晨阳

组员：齐天泽

报告日期：2025.1.4

目录

小组工作量划分

本组共包含两名成员。

组长李晨阳负责 ID 模块，EX 模块，RF 模块与总体调试运行，掌握本组 github 账号和仓库

组员齐天泽负责 IF 模块，MEM 模块，WB 模块和实验报告的书写以及其余细节工作。

工作量分布表如下:

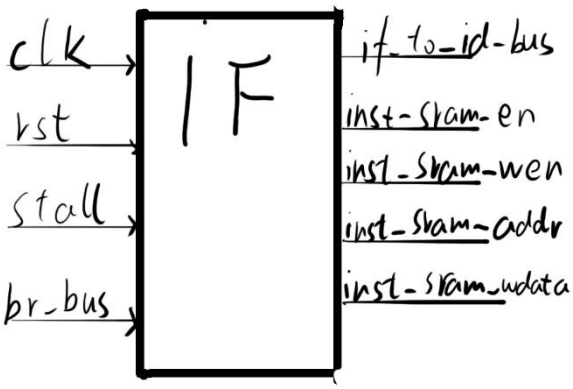
姓名	项目	占比
李晨阳	如上	50%
齐天泽	如上	50%

总体情况概述

实验结果

本组实验结果主要改进自课程提供的基础代码，通过学习实验课提供的教程，网络教程完成了最终代码的编写，并在 vivdao 平台上进行了仿真，最终结果成功通过了 64point 的仿真。

IF 模块



IF 模块的功能

该模块的主要功能是根据输入的 PC 取指令并发送给 ID 阶段, 并且根据条件更新 PC。

IF 模块的接口

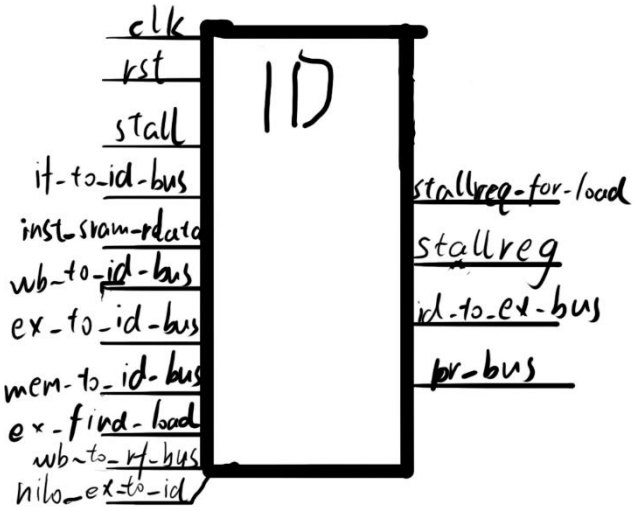
接口名	clk	rst	stall	br_bus	If_to_id_bus	Inst_sram_en	Inst_sram_wen	Inst_Sram_addr	Inst_Sram_wdata
宽度	1	1	1	33	33	1	4	32	32
I/O	输入	输入	输入	输入	输出	输出	输出	输出	输出

功能	时钟	复位	暂停控制	分支控制	向 ID 阶段的输出	指令储存器使能信号	指令储存器写使能信号 设置为 4‘b0，在此阶段不写数据’	指令储存器的地址输入	写入指令储存器的数据 设为 32‘b0，不向指令储存器写数据’

IF 模块的详细介绍

IF 模块从 clk，rst，stall 和 br_bus 分别获取时钟信息，复位信息，暂停信息和分支信息。在工作中，PC 寄存器根据时钟和复位信号进行 PC 值初始化，且若暂停信号有效，则不更新寄存器，反之当其代表不暂停时，按计划更新 PC 寄存器。对于取值的使能信号，根据时钟和复位信号在需要复位是会将其置为 0，其他情况若无暂停则置为 1。PC 寄存器的更新则根据分支信息进行，分支信息包含是否分支和分支地址两部分，若无分支，则按顺序更新 PC 值使其指向下一指令（+4），否则更新为分支地址。在完成这些操作后，使能信号和指令地址会被发给指令储存器进行取指令，这些信号也会被打包通过 if_to_id_bus 发给 ID 阶段进行进一步处理。

ID 模块



ID 模块的功能

解码指令，并根据解码结果操作寄存器，处理数据，同时向 EX 阶段传递信息。

ID 模块的接口

接口	宽度	I/O	功能
Clk	1	输入	时钟信号
Rst	1	输入	复位信号
Stall	6	输入	暂停控制
If_to_id_bus	33	输入	IF 向 ID 传递数据的总线
Inst_sram_rdata	32	输入	指令储存器回传的指令数据
Wb_to_rf_bus	38	输入	WB 阶段连来的数据总线，用于转发到寄存器文件
Ex_to_id_bus	38	输入	EX 阶段连来的数据总线
Mem_to_id_bus	38	输入	MEM 阶段连来的数据总线
Ex_find_load	1	输入	EX 回传的 load 指令信息
hilo_ex_to_id	66	输入	EX 传给 ID 段要写入乘 除法寄存器的值
wb_to_id_bus	38	输入	WB 段传给 ID 段的数 据，用来判断数据相关
Stallreq_for_load	1	输出	由于加载原因提出的暂停请求，发给 CTRL
Stallreq	1	输出	发给 CTRL 的暂停请

			求
Id_to_ex_bus	169	输出	ID阶段向EX阶段的输出总线，传递相关信息给 EX 阶段
Br_bus	33	输出	输出分支信息的总线

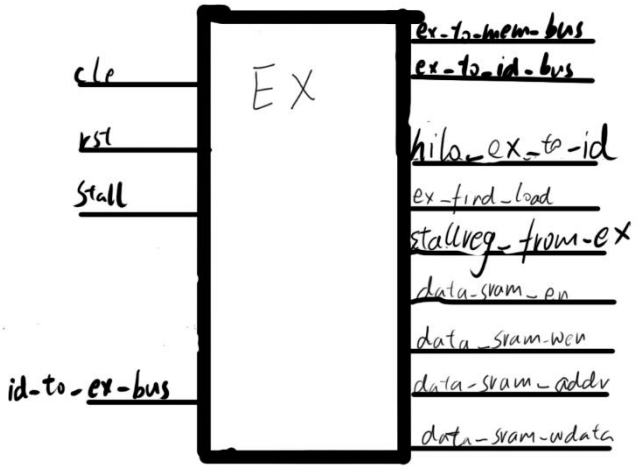
ID 模块的详细介绍

在从输入接口获得相关信息后，ID 阶段将结合这些信息解析指令。

首先，系统根据时钟，复位和暂停信息处理 slot 状态和 slot 寄存器，若流水线工作正常没有暂停，则 ID 阶段将接受 IF 阶段传来的信息供后续操作，若流水线后续状态出于暂停状态，则在 slot 寄存器中储存从指令储存器读来的指令且将 slot 状态设为是，若流水线工作正常但 ID 阶段暂停，则清空储存 IF 数据的寄存器。复位时，清空 slot 寄存器，重置 slot 状态。

之后开始解码指令，首先根据 slot 的状态判断指令来源，若 slot 内有指令则使用这条指令，否则使用 inst_sram_rdata 接口传入的指令，之后解析 IF 和 EX 阶段传来的信息。进行指令的解码，根据操作码确定于不同类型的指令，然后解析出所需的 rs,rt,rd 或者功能码，立即数和跳转地址。并根据解码结果设置后续 ALU 和寄存器等需要的控制信号。对于需要读取寄存器的情况，进行寄存器的读取。在完成分支判断后，解码的结果将被打包发送给 EX 阶段。

EX 模块



EX 模块的功能

处理 ALU 结果，并根据指令确定内存的读写情况，准备好对应的数据和地址

EX 模块的接口

接口	宽度	I/O	功能
Clk	1	输入	时钟信号
Rst	1	输入	复位信号
Stall	6	输入	暂停信号
id_to_ex_bus	169	输入	来自 ID 阶段传递给 EX 阶段的数据总线，包含了指令相关的各种信息
ex_to_mem_bus	80	输出	EX 阶段到 MEM 阶段的数据总线，传递计算结果、内存地址、控制信号等

ex_to_id_bus	38	输出	EX 阶段到 ID 阶段的数据总线，向 ID 阶段回传计算结果
hilo_ex_to_id	66	输出	EX 段将乘除法器的结 果发给 ID 段的 regfile 模块
stallreq_from_ex	1	输出	EX 发出的是否暂停的 信号
ex_find_load	1	输出	指示是否发现了 load 指令
data_sram_en	1	输出	存储器使能信号
data_sram_wen	4	输出	SRAM 的写使能信号
data_sram_addr	32	输出	存储器地址信号
data_sram_wdata	32	输出	写入存储器的数据

EX 模块的详细介绍

在 EX 模块中首先准备了两个寄存器分别储存 ID 阶段传来的 SRAM 读写信号。这两个寄存器的操作综合时钟信号，复位信号和暂停信号处理。当复位发生时，两个寄存器会被清空复位。在正常无暂停情况下，两个寄存器将会分别接受 ID 阶段的读写 SRAM 数据。当暂停发生时，寄存器会被清空防止储存错误数据。

之后来自 ID 总线的输入数据和其他打包的输入数据都会被解包为多个部分供后续操作使用。

根据输入信息, 立即数会被扩展。之后根据解包活动的控制信息为 ALU 选择操作数来源。在完成选择后，数据被输入调用的 ALU 模块里进行计算并输出结果。

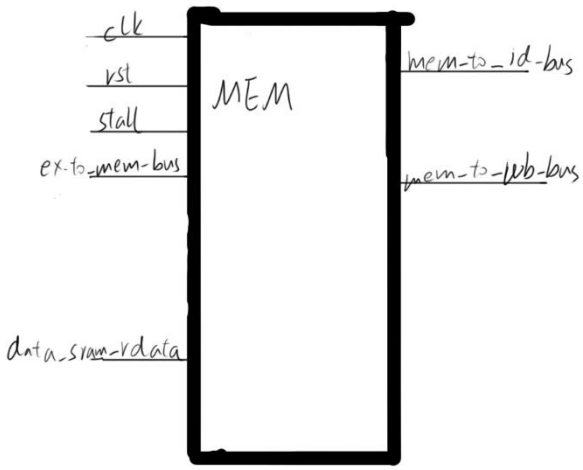
来自 SRAM 的读和写控制信号在解包后会被进一步处理。根据控制信号的不同，该模块会决定对内存的读写方法，选择不同字节进行内存操作。该决定会作为控制信号输出。此外，对 SRAM 的使能信号，写使能信号，数据地址和写入数据都会被处理并输出。另外，解包数据中如果遇到了加载指令，责输出需要暂停的信号

写回 ID 阶段的信息包含了是否写回寄存器，写回寄存器的地址和 ALU 的计算结果三部分。

传给 MEM 阶段的信息包括 PC 值，对 RAM 和 RF 的操作信号，和 ALU 的计算结果等部分。

该模块还包含乘法，除法和控制三个子模块。乘，除法器根据对于的操作指令和操作数完成计算。控制模块则控制除法模块的运作并控制是否请求进行暂停。这部分主要是控制除法器的操作数并根据除法器输出的是否准备好的状态信号判断是否需要申请停止，当除法器未准备好时，这部分即输出申请停止的信号。

MEM 模块



MEM 模块的功能

处理来着 EX 阶段的数据，进行内存的选择与访问存取，结果将被传向 WB 阶段或 ID 阶段

MEM 模块的接口

接口	宽度	I/O	功能
Clk	1	输入	时钟信号
Rst	1	输入	复位信号
Stall	6	输入	暂停信号

ex_to_mem_bus	80	输入	接受 EX 阶段数据的总线
data_sram_rdata	32	输入	从 sram 中读取数据
mem_to_id_bus	38	输出	MEM 阶段向 ID 阶段回传信息的总线
mem_to_wb_bus	70	输出	MEM 阶段向 WB 阶段传递信息的总线

MEM 模块的详细介绍

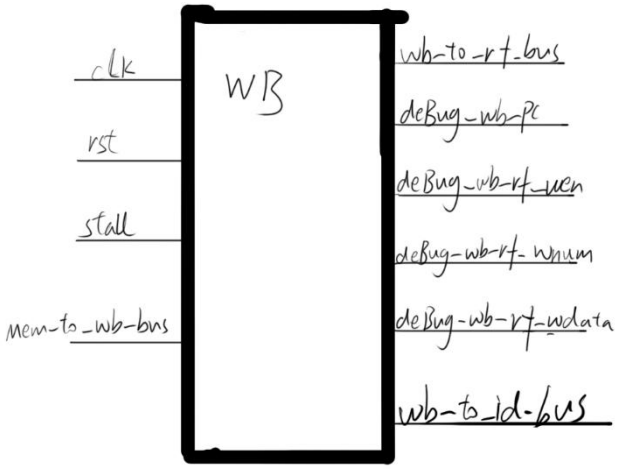
MEM 模块有三个寄存器分别储存从 EX 阶段传来的数据和 EX 阶段要读写的数据。这三个寄存器根据时钟和复位信号会进行复位操作，而结合不同的暂停信号，会进行不同的写入操作。当流水线正常时，寄存器会储存对于接口传来的信息，当 EX 阶段暂停时，寄存器会被清空防止误传错误数据。

之后，这些数据会被解析，其中部分数据将会被用于控制下一步数据读写。其中，EX 总线串联的信息可以控制选择不同大小的数据进行进一步操作。根据选择结构处理输入数据之后，会生成最终数据。

最终数据会与上一阶段传来的 ALU 结果进行对比，安装控制信号选择对应数据来写入寄存器。

最后，PC 地址，寄存器控制信号和对应的数据会在打包后分别通过对于总线传给 ID 阶段和 WB 阶段。

WB 模块



WB 模块的功能

将 MEM 阶段的处理结果传递给寄存器，并根据控制信号判断是否要进行写回操作。还有一些用于调试的输出接口。

WB 模块的接口

接口	宽度	I/O	功能
Clk	1	输入	时钟信号
Rst	1	输入	复位信号
Stall	6	输入	暂停信号
Mem_to_wb_bus	70	输入	来着 MEM 阶段的数据总线
Wb_to_id_bus	38	输出	WB 回传给 ID 的数据
Wb_to_rf_bus	38	输出	将写回数据传给寄存器文件的总线

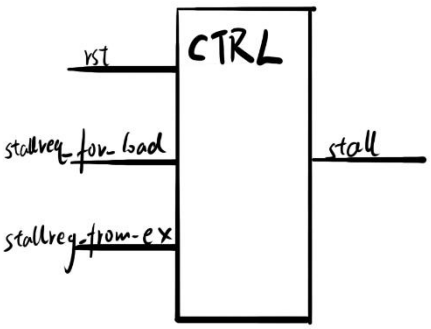
Debug_wb_pc	32	输出	输出本阶段的 PC 值
Debug_wb_rf_wen	4	输出	输出本阶段使能信号
Debug_wb_rf_wnum	5	输出	输出本阶段操作的寄存器编号
Debug_wb_rf_wdata	32	输出	输出本阶段要写入寄存器的值

WB 模块的详细介绍

WB 模块提供了一个用于储存 MEM 阶段传来的数据的寄存器，这个寄存器将根据时钟信号，复位信号和暂停信号来操作。当时钟信号和复位信号为高时，寄存器的数据将被清空。正常情况下，如果流水线没有被暂停，则来自 MEM 总线的数据将被写入寄存器。如果遇到暂停操作，寄存器会被清空防止储存错误数据。

在完成数据接收后，这部分数据会被解码出 PC 值，写使能信号，目标寄存器地址，写回数据四部分，后三部分会被再次打包并通过通向目标模块的总线传给寄存器文件等模块。此外这四部分数据还会被单独处理后作为调试输出。

CTRL 模块



CTRL 模块的功能

该模块主要用来处理暂停信号。

CTRL 模块的接口

接口	宽度	I/O	功能
Rst	1	输入	复位信号
Stallreq_for_load	1	输入	ID 发出的加载相关暂停请求信号
Stallreq_from_ex	1	输入	EX 阶段发出的暂停请求
stall		输出	输出暂停信号

CTRL 模块的详细介绍

本阶段主要结合复位信号和暂停请求生成暂停信号。当复位发生时，或没有收到暂停请求时，暂停信号会被清空代表流水线所以阶段都不暂停。如果收到了暂停请求，则译码阶段暂停，

执行阶段暂停，访存阶段不暂停，回写阶段不暂停。

最终输出的暂停信息为一个多位数据，结构如下：

stall[0]：取指地址 PC 是否保持不变，为 1 表示保持不变。

stall[1]：取指阶段是否暂停，为 1 表示暂停。

stall[2]：译码阶段是否暂停，为 1 表示暂停。

stall[3]：执行阶段是否暂停，为 1 表示暂停。

stall[4]：访存阶段是否暂停，为 1 表示暂停。

stall[5]：回写阶段是否暂停，为 1 表示暂停。

RegFile 模块

RegFile 模块的功能

本模块提供了 32 个普通的 32 位寄存器共处理器使用，且针对乘除法器提供了高位和低位寄存器。

RegFile 模块的接口

接口	宽度	I/O	功能
Clk	1	输入	时钟信号
Raddr1	5	输入	请求读取的第一个数的 地址
Rdata1	32	输出	读取出的第一个数的值
Raddr2	5	输入	请求读取的第二个数的 地址
Rdata2	32	输出	读取出的第二个数的值

We	1	输入	是否要写入寄存器的写 使能信号
Waddr	5	输入	要写入的地址
Wdata	32	输入	要写入寄存器的值
Hi_r	1	输入	是否要读取 hi 寄存器的 值的读使能信号
Hi_we	1	输入	是否要写入 hi 寄存器的 写使能信号
Hi_data	32	输入	要写入 hi 寄存器的值
Lo_r	1	输入	是否要读取 lo 寄存器的 值的读使能信号
Lo_we	1	输入	是否要写入 lo 寄存器的 写使能信号
Lo_data	32	输入	要写入 lo 寄存器的值
Hilo_data	32	输出	从 hilo 寄存器中读取出 来的值

RegFile 模块的详细介绍

regfile 模块定义了 32 个 32 位的通用寄存器，以及一个乘除法的高位 hi 寄存器，一个乘除法的低位 lo 寄存器。且因为这两个寄存器不会同时被调用，所以只设计了一个输出接口输出有结果的那个寄存器的值。在确定 rs 寄存器以及 rt 寄存器的值时，判断 raddr1 是否为零，如果为零，就赋值给 rdata1,如果不为零，就把 raddr1 对应的寄存器的值赋值给 rdata1; rdata2 与 raddr2 同理。

总结与思考

这次实验让我们把在课堂上学到的东西运用到了实践中，让我对流水线的整体运行有了更加深刻的理解。我认为本次实验是非常具有挑战性的，从一开始在仿真中根据输出结果定位问题就让我烦恼了很久，这是一种与传统编程 debug 非常不同的体验。 在编写流水线时，因为毫无经验且很多地方需要自行设计，在实验过程中经历了非常多的问题。也发现许多看起来简单的问题在实践时会有好多没注意到的细节问题导致失败。例如数据相关问题，流水线暂停问题，内存的读写问题等。在解决这些问题的过程中，我不断查阅资料书籍，搜索信息，与同学们交流讨论，最终逐一解决并获得了尚可的结果。 这次实验让我对流水线的工作原理有了更深入的理解，同时也让我更加认识到团队合作和系统架构的重要性。总体而言，这次课程设计不仅加深了我对相关理论的掌握，还提高了我的编程能力，并且让我积累了宝贵的团队协作经验，收获颇丰。