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2202

组长：张浩岚

组员：代诗博

报告日期：2025.1.4

目录

[1. 实验设计 3](#_Toc19256)

[1.1小组成员工作量划分 3](#_Toc7172)

[1.2 总体设计 3](#_Toc32491)

[1.3 运行环境及工具 3](#_Toc19652)

[2. 流水线各个阶段的说明 4](#_Toc23612)

[2.1 IF模块 5](#_Toc27838)

[2.2 ID模块 6](#_Toc16552)

[2.3 EX模块 7](#_Toc11788)

[2.4 MEM模块 9](#_Toc32299)

[2.5 WB模块 1](#_Toc2519)0

[2.6 CTRL模块 1](#_Toc25572)1

[2.7 HILO寄存器模块 1](#_Toc19197)2

[3. 实验感受及建议 1](#_Toc12453)3

[3.1 张浩岚部分 1](#_Toc32453)3

[3.2 代诗博部分 1](#_Toc26449)3

[4. 参考资料 1](#_Toc31488)3

1. **实验设计**

**1.1小组成员工作量划分**

|  |  |  |
| --- | --- | --- |
| 姓名 | 任务分工 | 任务量占比 |
| 张浩岚 | 添加算术运算，逻辑运算，位移指令 | 50% |
| 代诗博 | 添加分支跳转，数据移动，访存指令 | 50% |

**1.3 运行环境及工具**

运行环境：装有Vivado 的Linux服务器。 FPGA的Family为Artix 7，Package为fbg676，型号为xc7a200tfbg676-2。

编程工具：使用VSCode编写代码，使用Vivado 模拟仿真，使用git进行版本管理，使用GitHub搭建项目仓库。

**1.2 总体设计**

项目包括IF.v，ID.v，EX.v，MEM.v，WB.v，hi\_lo\_reg.v，mycpu\_core.v，mycpu\_top.v，这部分搭建了一条流水线的基本框架；及位于/lib目录下的alu.v，decoder\_2\_4.v，decoder\_5\_32.v，decoder\_6\_64.v，defines.vh，div.v，mmu.v，regfile.v，这部分构建了ALU和寄存器，定义了包含总线宽度信息在内的头文件；及位于/lib/mul目录下的add.v，fa.v，mul.v，这部分实现了乘法的运算。

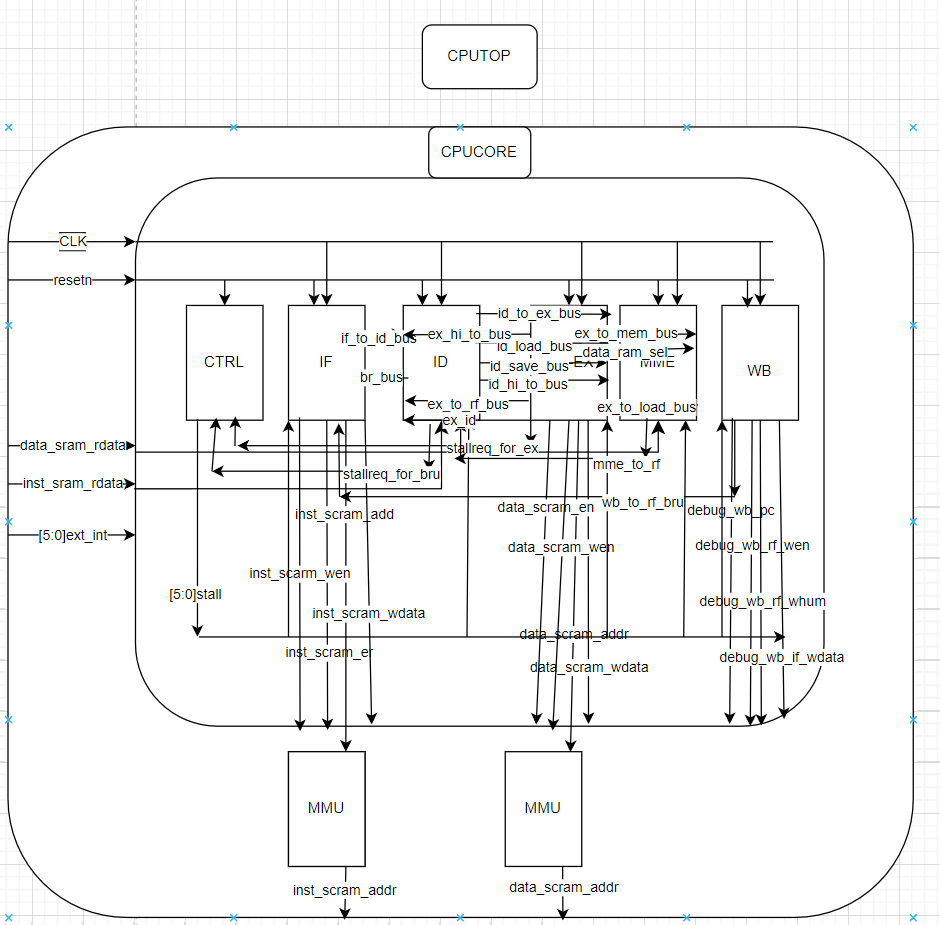
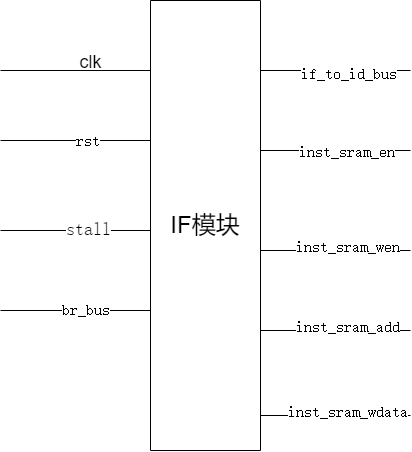
****

图 1 CPU流水线示意图

**2.流水线各个阶段的说明**

**2.1 IF模块详细说明**

功能：

指令取指模块（IF模块）是CPU流水线的第一个阶段，其主要任务是从指令存储器中取出指令，并为后续的指令译码和执行做准备。此外，它还负责处理指令延迟槽和跳转指令，以确保指令的正确顺序执行.

#### 主要操作：

输入时钟和复位信号：IF模块接收来自时钟源的时钟信号，以同步其操作。同时，它还接收复位信号，当复位信号为高时，程序计数器（PC）的值会被清零，以便从指令存储器的第一个地址开始取指.

检查暂停信号（stall）：在每个时钟周期，IF模块都会检查暂停信号的状态。如果暂停信号为1，表示流水线需要暂停，此时IF模块会保持PC值不变，不从指令存储器中取指，以等待后续阶段的数据相关性问题得到解决.

处理分支跳转信号（br\_bus）：当遇到分支跳转指令时，IF模块会根据分支跳转信号的状态来决定是否更新PC值。如果分支跳转信号指示需要跳转，则IF模块会将新的目标地址加载到PC中，从而实现指令流的跳转；否则，PC值会按照顺序递增，继续取指下一条指令.

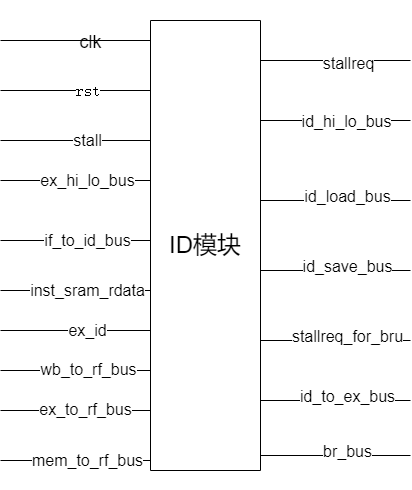
从指令内存中获取指令：在正常情况下，IF模块会根据当前PC值从指令存储器中读取指令，并将其发送至指令译码段（ID段），以便进行后续的指令译码和执行操作.

#### 接口：

IF模块的接口包括时钟信号、复位信号、暂停信号、分支跳转信号、数据总线等。通过这些接口，IF模块能够与CPU的其他模块进行通信和数据交换，以实现指令的正确取指和流水线的协调运行.

表 1 IF模块输入输出

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 |

**2.2 ID模块详细说明**

#### 功能：

指令译码模块（ID模块）是CPU流水线的第二个阶段，其主要任务是对从IF模块传来的指令进行译码，确定指令的类型、操作数以及执行所需的控制信号。此外，ID模块还负责处理寄存器的读写操作，并解决可能出现的数据相关性问题，以确保指令能够正确地执行.

#### 主要操作：

接收IF段的数据：ID模块接收来自IF模块的指令数据，包括指令的二进制编码以及相关的地址信息等。通过对这些数据进行译码，ID模块能够识别出指令的操作类型、操作数来源以及目标寄存器等信息.

处理数据相关性：在指令译码过程中，ID模块会检查是否存在数据相关性问题，即后续指令所需要的操作数是否依赖于前面指令的执行结果。如果检测到数据相关性，ID模块可能会发出暂停信号，要求流水线暂停，直到相关数据可用为止，以避免指令执行错误.

确定操作数来源：ID模块根据指令的类型和操作数的来源，从寄存器堆中读取相应的操作数。对于立即数指令，操作数直接从指令中提取；对于寄存器间操作的指令，则从寄存器堆中读取操作数的值.

设置ALU操作：根据指令的类型和操作数，ID模块会生成相应的控制信号，设置ALU（算术逻辑单元）的操作类型，如加法、减法、逻辑运算等，以便在后续的执行阶段中进行正确的运算操作.

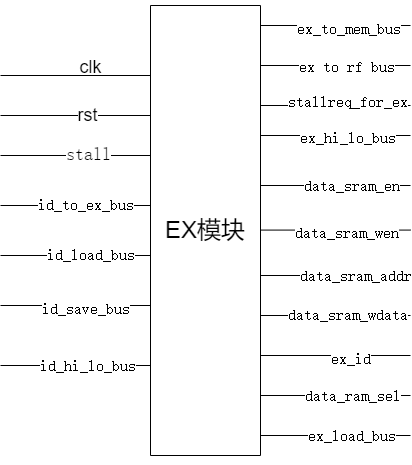
准备数据和控制信号：ID模块将译码后的指令信息、操作数以及相关的控制信号打包，传递给执行段（EX段），为后续的指令执行做好准备.

#### 接口：

ID模块的接口包括时钟信号、复位信号、暂停信号、数据总线、寄存器读写信号等。通过这些接口，ID模块能够与IF模块、EX模块以及寄存器堆等其他模块进行数据交换和通信，以实现指令的正确译码和数据的准确传递.

表 2 ID模块输入输出

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 |

**2.3 EX模块详细说明**

#### 功能：

执行模块（EX模块）是CPU流水线的第三个阶段，其主要任务是根据ID模块传来的指令信息和操作数，执行相应的算术或逻辑运算操作。此外，EX模块还负责计算内存访问指令的地址，并处理ALU运算的结果，为后续的内存访问和写回操作提供数据支持.

#### 主要操作：

从ID/EX寄存器读取操作数：EX模块会从ID/EX寄存器中读取来自ID模块的操作数和指令信息。ID/EX寄存器是一个缓冲寄存器，用于在ID模块和EX模块之间传递数据，确保数据在流水线中的正确流动.

执行ALU运算：根据指令的类型和操作数，EX模块会控制ALU进行相应的运算操作。ALU是CPU的核心部件之一，能够执行各种算术运算（如加法、减法、乘法等）和逻辑运算（如与、或、非等）。EX模块会将运算结果暂存在一个临时寄存器中，以供后续使用.

处理访存指令：对于内存访问指令（如load和store指令），EX模块会根据指令的类型和操作数，计算出内存访问的地址。对于load指令，EX模块会将计算出的地址传递给内存模块，以便从内存中读取数据；对于store指令，则会将数据和地址传递给内存模块，以便将数据写入内存.

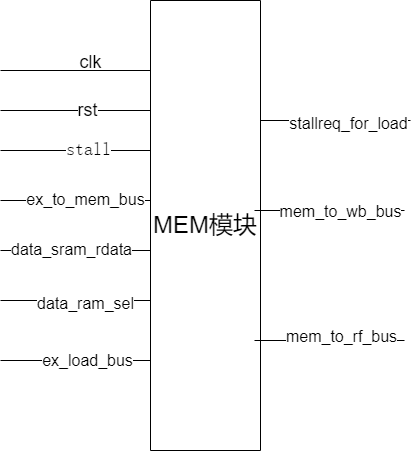
将结果传递给MEM段：EX模块将ALU运算的结果以及内存访问的地址等信息传递给内存访问段（MEM段），为后续的内存访问操作和写回操作提供数据支持.

#### 接口：

EX模块的接口包括时钟信号、复位信号、暂停信号、数据总线、内存控制信号等。通过这些接口，EX模块能够与ID模块、MEM模块以及ALU等其他模块进行数据交换和通信，以实现指令的正确执行和数据的准确传递.

表 4 EX模块输入输出

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 | id\_hi\_lo\_bus | 72 | 输入 | ID 段传给hilo段的数据 |
| 10 | ex\_hi\_lo\_bus | 66 | 输出 | EX 段传给hilo段的数据 |
| 11 | stallreq\_for\_ex | 1 | 输出 | 对EX段的stall请求 |
| 12 | data\_sram\_en | 1 | 输出 | 内存数据的读写使能信号 |
| 13 | data\_sram\_wen | 4 | 输出 | 内存数据的写使能信号 |
| 14 | data\_sram\_addr | 32 | 输出 | 内存数据存放的地址 |
| 15 | data\_sram\_wdata | 32 | 输出 | 要写入内存的数据 |
| 16 | ex\_id | 38 | 输出 | EX 段传给 ID 段的数据 |
| 17 | data\_ram\_sel | 4 | 输出 | 内存数据的选择信号 |
| 18 | ex\_load\_bus | 5 | 输出 | EX 段读取的数据 |

 **2.4 MEM模块详细说明**

#### 功能：

内存访问模块（MEM模块）是CPU流水线的第四个阶段，其主要任务是执行内存访问操作，包括读取内存中的数据和将数据写入内存。MEM模块根据EX模块传来的地址和控制信号，与内存进行交互，以完成指令的内存访问需求.

#### 主要操作：

从EX/MEM寄存器获取地址：MEM模块会从EX/MEM寄存器中获取来自EX模块的内存访问地址和指令信息。EX/MEM寄存器是一个缓冲寄存器，用于在EX模块和MEM模块之间传递数据，确保数据在流水线中的正确流动.

访问内存：根据指令的类型和地址，MEM模块会向内存发送相应的读写请求。对于load指令，MEM模块会从内存中读取数据，并将其暂存在一个临时寄存器中；对于store指令，则会将数据写入指定的内存地址.

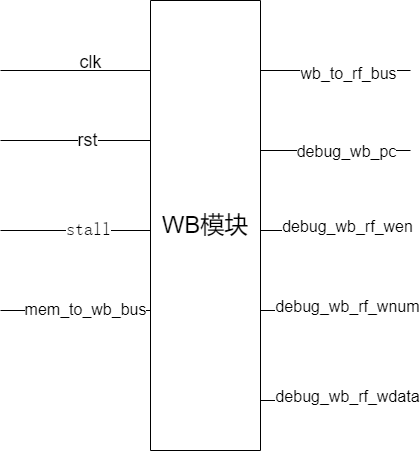
处理load和store指令：MEM模块会根据指令的类型，选择相应的数据进行写回。对于load指令，MEM模块会将从内存中读取的数据传递给写回段（WB段），以便将数据写回寄存器堆；对于store指令，则不需要进行写回操作.

#### 接口：

MEM模块的接口包括时钟信号、复位信号、暂停信号、数据总线、内存读写信号等。通过这些接口，MEM模块能够与EX模块、WB模块以及内存等其他模块进行数据交换和通信，以实现指令的正确内存访问和数据的准确传递.

表 5 MEM模块输入输出

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 |

 **2.5 WB模块详细说明**

#### 功能：

写回模块（WB模块）是CPU流水线的最后一个阶段，其主要任务是将执行结果写回到寄存器堆中。WB模块根据MEM模块传来的数据和控制信号，将数据写入指定的寄存器，从而完成指令的执行过程.

#### 主要操作：

从MEM/WB寄存器读取数据：WB模块会从MEM/WB寄存器中读取来自MEM模块的执行结果和指令信息。MEM/WB寄存器是一个缓冲寄存器，用于在MEM模块和WB模块之间传递数据，确保数据在流水线中的正确流动.

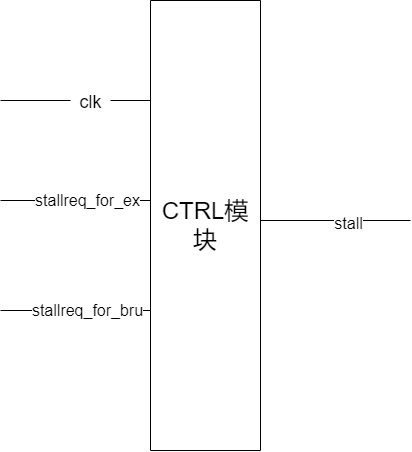
将数据写回寄存器堆：根据指令的目标寄存器地址，WB模块会将执行结果写入寄存器堆中的相应寄存器。寄存器堆是CPU中的一个重要部件，用于存储指令执行过程中产生的中间结果和最终结果.

#### 接口：

WB模块的接口包括时钟信号、复位信号、暂停信号、数据总线、调试信号等。通过这些接口，WB模块能够与MEM模块、寄存器堆等其他模块进行数据交换和通信，以实现指令的正确写回和数据的准确存储.

表 6 WB模块输入输出

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 |

 **2.6 CTRL模块详细说明**

#### 功能：

控制模块（CTRL模块）是CPU流水线的控制中心，其主要任务是控制流水线各阶段的运行，确保指令的正确取指、译码、执行、内存访问和写回。CTRL模块根据各阶段的状态和请求信号，生成相应的控制信号，以协调流水线的运行.

#### 主要操作：

接收各段的请求信号：CTRL模块会接收来自流水线各阶段的请求信号，如暂停请求信号、跳转请求信号等。这些请求信号反映了各阶段的运行状态和需求，CTRL模块需要根据这些信号来做出相应的控制决策.

生成暂停信号：当流水线出现数据相关性、资源冲突等问题时，CTRL模块会生成暂停信号，控制流水线暂停。暂停信号会传递给流水线的各个阶段，使它们暂时停止操作，等待相关问题得到解决后再继续执行.

#### 接口：

CTRL模块的接口包括时钟信号、暂停请求信号、暂停信号等。通过这些接口，CTRL模块能够与流水线的各个阶段进行通信和协调，以实现流水线的正确控制和高效运行.

表 7 CTRL模块输入输出

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 |

**2.7 HILO寄存器模块详细说明**

#### 功能：

HILO寄存器模块是CPU中用于处理乘法和除法运算的辅助寄存器模块。它们不存在通用寄存器内，其主要任务是提供hi和lo两个寄存器，用于存储乘法和除法运算的中间结果和最终结果，以便在后续的指令执行中使用.

#### 主要操作：

读写hi和lo寄存器：HILO寄存器模块能够根据指令的要求，对hi和lo寄存器进行读写操作。对于乘法运算，hi寄存器存储乘积的高位部分，lo寄存器存储乘积的低位部分；对于除法运算，hi寄存器存储商，lo寄存器存储余数.

支持独立读写和执行阶段的数值获取：HILO寄存器模块支持对hi和lo寄存器的独立读写操作，以便在不同的指令执行阶段中获取所需的数值。例如，在执行乘法指令时，可以在执行阶段读取hi和lo寄存器的值，以获取乘积的完整结果；在执行除法指令时，可以在写回阶段读取hi寄存器的值，以获取商的结果.

#### 接口：

HILO寄存器模块的接口包括时钟信号、暂停信号、写使能信号、读写数据等。通过这些接口，HILO寄存器模块能够与CPU的其他模块进行数据交换和通信，以实现乘法和除法运算的正确执行和结果的准确存储.

表 8 HILO寄存器输入输出

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 接口名 | 宽度 | 输入/输出 | 作用 |
| 1 | clk | 1 | 输入 | 时钟信号 |
| 2 | stall | 6 | 输入 | 控制暂停信号 |
| 3 | hi\_we | 1 | 输入 | hi寄存器的写使能信号 |
| 4 | lo\_we | 1 | 输入 | lo寄存器的写使能信号 |
| 5 | hi\_wdata | 32 | 输出 | Hi寄存器写的数据 |
| 6 | lo\_wdata | 32 | 输出 | Lo寄存器写的数据 |
| 7 | hi\_rdata | 32 | 输出 | Hi寄存器读的数据 |
| 8 | lo\_rdata | 32 | 输出 | Lo寄存器读的数据 |

**实验感受及建议**

**3.1 张浩岚部分**

在实验中我熟练的掌握了GitHub的使用，能够用它搭建仓库、审阅代码、管理版本，极大的提高了我们的工作效率。

除此之外根据前人经验和AI辅助我实现了算数运算指令，逻辑运算指令，位移指令的构建，在这个实验之中我明白了五级流水线工作的具体情况以及流程，体会到了团队合作的重要性。

**3.2 代诗博部分**

本次实验我深入了解了流水线的整体运行过程，把课堂中学习到的知识真正代入到了实践中。

本次实验我在借鉴学长的工作成果的条件下实现了分支跳转指令，访存指令，数据移动指令。

这次实验同时也让我明白了团队合作的重要性，要想使任务完成的更加成功，必须要分工明确并且和队友多多交流，体现出团队的价值。

总之，这次实验让我了解并学习了一门新的编程方法，更加深入的探究了流水线的运行逻辑和具体细节，领悟到了团队的强大力量。

**4. 参考资料**

1、张晨曦 著《计算机体系结构》（第二版） 高等教育出版社

2、雷思磊 著《自己动手写 CPU》 电子工业出版社