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》

课程实验报告

班级：人工智能 2002 班

组长：李文丽

组员：周萌萌

报告日期：2023.01.01

1. 总体设计

1.1 工作量

李文丽：60%，周萌萌：40%

1.2 连线图

由于连线图过大，见随文档提交的 pdf 文件

1.3 完成指令条数

一共完成了 52 条指令，分别为：

算数运算指令：ADD， ADDI,ADDU,ADDIU,SUB,SUBU,SLT,SLTI,SLTU,SLTIU, DIV,DIVU,MULT,MULU 共 14 条；

逻辑运算指令：AND,ANDI,LUI,NOR,OR,ORI,XOR,XORI 共 8 条；

位移指令：SLLV,SLL,SRAV,SRA,SRLV,SRL 共 6 条；

数据移动指令：MFHI,MFLO,MTHI,MTLO 共 4 条；

访存指令：LB,LBU,LH,LHU,LW,SB,SH,SW 共 8 条；

分支跳转指令： BEQ,BNE,BGEZ,BGTZ,BLEZ,BLTZ, BGEZAL, BLTZAL, J, JAL,JR,JALR 共 12 条。

1.4 程序运行环境

Windows 操作系统

1.5 使用工具

使用的工具为：Vivado，visual code

2. 单个流水段说明

2.1 IF 段

2.1.1 整体功能说明

本阶段主要是根据 PC 寄存器的值来从存储器相应地址中读取对应的指令信息的过程，负责取指，将得到的指令传递给 ID 段进行后续操作。

2.1.2 端口，信号介绍

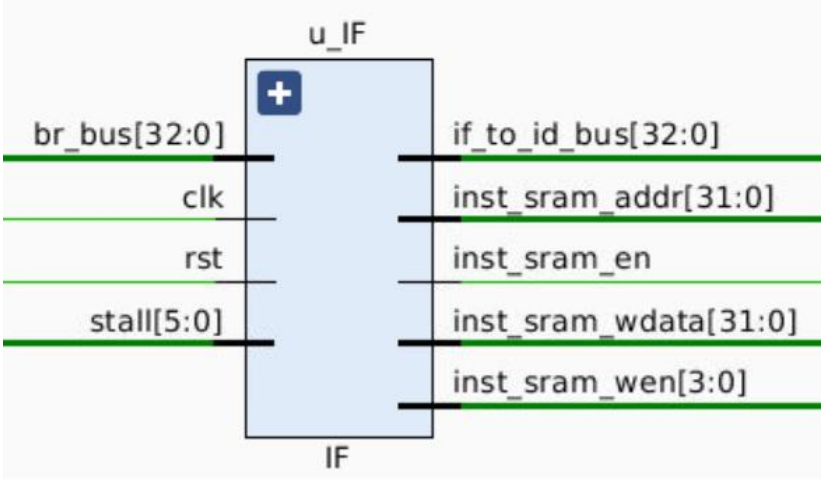
输入：clk：传入时钟周期信号；rst：复位信号，负责初始化各种数据；stall：暂停信号，用于暂停流水线；br_bus：来自 ID 段的输入，存放跳转指令的相关信号。

输出：if_id_to_bus：打包 IF 段信息传到 ID 段的总线；inst_sram_wen：存储器写入的使能信号；inst_sram_wen：存储器读取的使能信号；inst_sram_addr：存储器的地址；inst_sram_wdata：存储器写入内存的数据。

2.1.3 功能模块说明

br_bus：从 ID 段传来的跳转指令，判断是否需要跳转。若需要，就使用里面的跳转地址，否则就 PC+4；获得 PC 指令；获得使能信号 ce。

2.1.4 结构示意图



2.2 ID段

2.2.1 整体功能说明

指令译码是指将存储器中取出的指令进行翻译的过程，负责译码，将IF段传来的指令进行解析。

2.2.2 端口，信号介绍

输入：clk：传入时钟周期信号；rst：复位信号，负责初始化各种数据；stall：暂停信号，用于暂停流水线；if_to_id_bus：IF段传来的总线；inst_sram_rdata：要写入内存的数据；stallreq：向 CTRL 模块传递 ID 是否要暂停的信号；ex_to_id_bus：从EX段传回ID段的总线，用于解决数据相关；mem_to_id_bus：从MEM段传回ID段的总线，用于解决数据相关；wb_to_id_bus：从WB段传回ID段的总线，用于解决数据相关。

输出：id_to_ex_bus：打包ID段的信息传给EX段的总线；br_bus：传给IF段来进行跳转指令。

2.2.3 功能模块说明

从MEM传到ID段用来解决数据相关的总线定义为mem_to_id_bus；从EX段传到ID段用来解决数据相关的总线定义为ex_to_id_bus。

解决数据相关就是确保从寄存器读出的数据是上一步写入的，防止RAW相关的现象发生，具体的解决思路是如果从IF段传入的指令不生效则将从rs，rt寄存器读出的数据置为0，如果EX段需要写入且EX段写入的数据的地址是rs寄存器的地址，那么让rs寄存器中的数据直接等于EX段写入的数据；如果MEM段需要写入且MEM段要写入的地址与rs寄存器的地址相同，那么让rs寄存器中的数据直接等于MEM段要写入的数据，不用再等一段时间才能写入。我们使用相同的方法对rt寄存器进行处理。 always代码段部分进行赋值操作，当复位信号为1时，说明需要复位，那么给总线和停止指令赋初值0，当复位信号为0时，说明指令开始运行，如果需要停止则将总线用IF段传来的if_to_id_bus赋值，并且将停止使能id_stop置为0；如果不停止则将总线用从IF段传来的if_to_id_bus赋值，并且将停止使能id_stop置为1。

wb_to_rf_bus从WB段传到ID段用来写入寄存器的写入使能，要写入的寄存器的地址以及要写入的数据的总线。

sel_alu_src1[1]用来判断是否需要从寄存器中读取指令。

sel_alu_src1[2]用来判断是否需要进行 sa 的无符号扩展。

sel_alu_src2[0]用来判断该指令是否需要从 rt 读取数据。

sel_alu_src2[1]是否要对立即数进行有符号扩展

sel_alu_src2[2]是否要对该指令的 PC 进行+8 处理。

sel_alu_src2[3]是否要对立即数进行无符号扩展

alu_op 用来判断传入加法器的数据要进行哪种类型的操作，如果要进行某一操作，则将该操作所对应的为置为 1，将其它置为 0.

data_ram_en 用来判断该指令是否需要对内存进行处理，如果需要置为 1

data_ram_readen 表示是否要将从虚地址读出的数据写入 rt 寄存器，如果是 lw 指令那么置为 1111，代表取四个字节的数据存入到 rt 寄存器中；若是 lb 指令置为 0001，代表取一个字节并进行无符号扩展并存入 rt 寄存器中；若是 lbu 指令置为 0010，代表取一个字节并进行有符号扩展并存入 rt 寄存器中；若是 lh 指令置为 0011，代表取连续的两个字节并进行无符号扩展并存入 rt 寄存器中；若是 lhu 指令置为 0100，代表取连续的两个字节并进行有符号扩展并存入 rt 寄存器中；若是 sb 指令置为 0101，表取一个字节并进行有符号扩展并存入 rt 寄存器中；若是 sh 指令置为 0111，代表取三个字节的数据存入到 rt 寄存器中。

rf_we 写入使能，判断指令是否有写入寄存器的需要，若有置为 1

sel_rf_dst[0]判断指令是否有写入 rd 寄存器的需要，若有置为 1

sel_rf_dst[1]判断指令是否有写入 rt 寄存器的需要，若有置为 1

sel_rf_dst[2]判断指令是否有写入 31 号寄存器的需要，若有置为 1

rf_waddr 计算要写入的寄存器的地址。

sel_rf_res 判断写入寄存器的结果是在 EX 段还是 MEM 段得出的。

hi_read ，lo_read 判断该指令是否需要从 hilo 寄存器中读取数据（即从 hilo 寄存器中读取 hi 寄存器和 lo 寄存器的数据并将数据存到 rd 寄存器中）

hi_write，lo_write 是否要将乘法或除法得到的数据存入 hilo 寄存器中。

id_to_ex_bus 将从 id 段得到的需要在后面几段用到的中间结果进行打包，并将打包后的总线传给 EX 段。

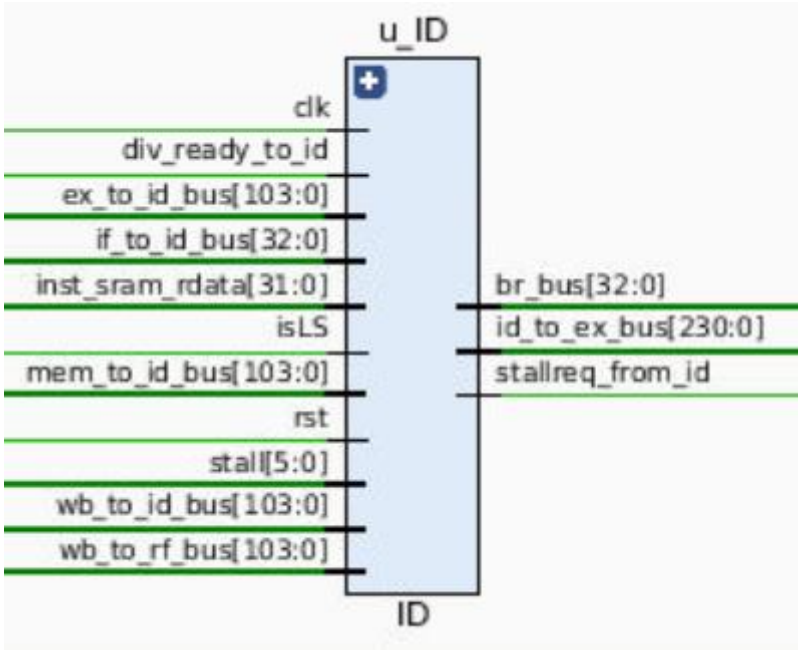
br_e 判断是要跳转指令还是顺序执行指令。

br_addr 下一个要执行的指令（是跳转之后的，不是顺序执行的）

br_bus 将上面得到的 br_e 和 br_addr 打包传给 IF 段，用来决定 IF 段下一个指令是按顺序取还是取跳转的指令。

stallreq 用来判断该指令是否需要停止，如果存在 rs 寄存器或 rt 寄存器的读后写等数据相关，则其赋值为 1，表示需要停止一个周期。

2.2.4 结构示意图



2.3 EX 段

2.3.1 整体功能说明

指令执行是指对指令进行真正运算的过程，对经 ID 段解析的指令进行运算，得到相关的结果。

2.3.2 端口，信号介绍

输入：clk：传入时钟周期；rst：复位信号，负责初始化各项数据；stall：停止信号，负责暂停流水线；id_to_ex_bus：从 ID 段传到 EX 段的总线。

输出：ex_to_mem_bus：EX 段到 MEM 段的总线；data_sram_en：是否对内存有操作； data_sram_wen：写入内存的使能；data_sram_addr：要写入的内存的地址；data_sram_wdata：写入内存的数据；ex_to_id_bus：EX 段到 ID 段的总线；stallreq_for_ex：判断是否需要暂停（乘除法）。

2.3.3 功能模块说明

alu 负责通过 alu_op 判断要进行的是哪个运算，然后对 alu_src1 和 alu_src2 两个操作数进行运算得出结果 alu_result。

乘法器 mul 负责乘法计算，基本原理是通过传入乘法的两个源操作数进行乘法运算，我们选用的乘法器具有多个周期，通过乘法的结束信号来判断来判断乘法是否已经运算完成，如果没有完成则要对整个流水线进行暂停，等运算结束后再进行其他指令的执行，其中乘法器接口的作用分别是：clk 用来传入时钟周期；resetn 是复位信号，负责初始化各项数据；mul_signed 用来判断当前乘法是否是有符号乘法；ina 表示第一个操作数；inb 表示第二个操作数。start_i 是乘法开始信号；ready_o 是乘法结束信号；计算结果存入 result。

除法器 div 负责除法计算，基本原理是通过传入的被除数和除数进行除法运算，我们选用的除法器具有多个周期，通过除法的结束信号来判断来判断乘法是否已经运算完成，如果没有完成则要对整个流水线进行暂停，等运算结束后再进行其他指令的执行，其中除法器接口的作用分别是：clk 用来传入时钟周期；rst 是复位信号，负责初始化各项数据；signed_div_i 用来判断当前乘法是否是有符号乘法；opdata1_i 表示被除数；opdata2_i 表示除数；annul_i 用来判断是否取消除法运算；start_i 是除法开始信号；ready_o 是乘法结束信号；计算结果存入 result_o。

always 中用来赋值，当复位信号为 1 时，给 ID 段到 EX 段的总线赋初值 0，当复位信号为 0 时，将总线用 ID 段传来的 id_to_ex_bus 赋值。

data_sram_addr 用来判断是否用到了虚拟内存，如果用到了，则将取数据的地址赋值为将 base 寄存器的值加上符号扩展后的立即数 offset，如果没有用到该地址赋值为 0。

data_sram_wen 用来判断写入内存的存放位置，如果为 sw 那么代表从 rt 寄存器中处取到的全部四个字节的数据存到之前计算出的虚地址所指的内存处；如果为 sh 指令且虚地址后两位为 00，那么代表将 rt 寄存器的低半字存到之前计算出的虚地址所指的内存处；如果为 sh 指令且虚地址后两位为 10，那么代表将 rt 寄存器的高半字存到之前计算出的虚地址所指的内存处；如果为 sb 指令且虚地址后两位为 00，么代表将 rt 寄存器的最低位的字节存到之前计算出的虚地址所指的内存处；如果为 sb 指令且虚地址后两位为 01，么代表将 rt 寄存器的倒数第二低位的字节存到之前计算出的虚地址所指的内存处；如果为 sb 指令且虚地址后两位为 10，么代表将 rt 寄存器的第二高位的字节存到之前计算出的虚地址所指的内存处；如果为 sb 指令且虚地址后两位为 11，么代表将 rt 寄存器的最高位的字节存到之前计算出的虚地址所指的内存处。

data_sram_wdata 表示 sw，sb，sh 三条指令要根据虚拟地址写入内存中的数据。如果为 sw 那么代表从 rt 寄存器中处取到的全部四个字节的数据存到之前计算出的虚地址所指的内存处；如果为 sh 指令且虚地址后两位为 00，那么代表将 rt 寄存器的低半字存到之前计算出的虚地址所指的内存处；如果为 sh 指令且虚地址后两位为 10，那么代表将 rt 寄存器的高半字存到之前计算出的虚地址所指的内存处；如果为 sb 指令且虚地址后两位为 00，么代表将 rt 寄存器的最低位的字节存到之前计算出的虚地址所指的内存处；如果为 sb 指令且虚地址后两位为 01，么代表将 rt 寄存器的倒数第二低位的字节存到之前计算出的虚地址所指的内存处；如果为 sb 指令且虚地址后两位为 10，么代表将 rt 寄存器的第二高位的字节存到之前计算出的虚地址所指的内存处；如果为 sb 指令且虚地址后两位为 11，么代表将 rt 寄存器的最高位的字节存到之前计算出的虚地址所指的内存处。

data_ram_readen 即从 ID 段传来的 data_ram_wen 用来判断传进来的指令是 sw，sb 还是 sw。

imm_sign_extend 对立即数进行有符号扩展。

imm_zero_extend 对立即数进行无符号扩展。

sa_zero_extend 对 sa 进行无符号扩展。

alu_src1 用来判断加法器的第一个操作数选择哪个（从 regfile 中读出的 rs 寄存器中的数，之前计算出的用于跳转指令的数以及立即数 sa 指定移位量对寄存器 rt 的值进行逻辑左移的数）。

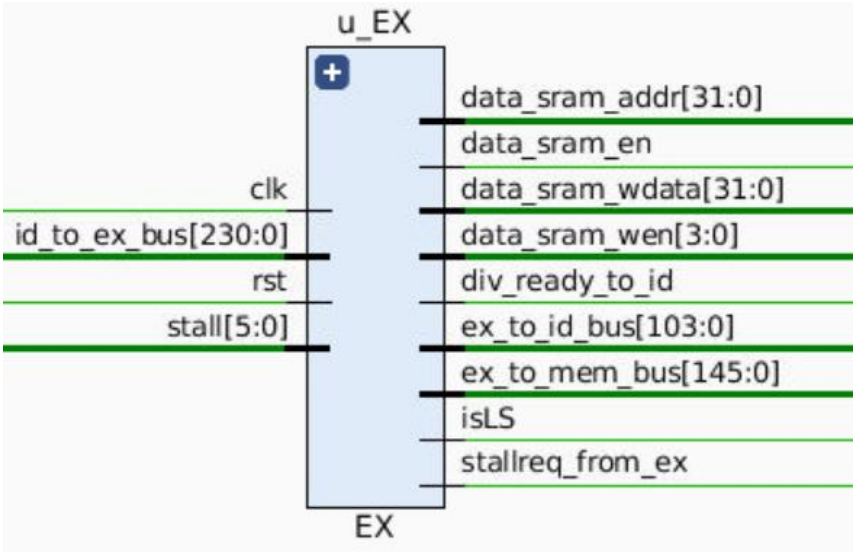
alu_src2 用来判断加法器的第一个操作数选择哪个（立即数有符号扩展之后的数，三十二位二进制的 8，立即数无符号扩展之后的数，以及从 regfile 中读出的 rt 寄存器中的数）。

stallreq_for_ex 用于判断当前阶段是否需要暂停（根据乘法器和除法器模块中的结束信号判断，即当结束信号为 1 时表示乘法和除法已经执行完成了，可以不用再暂停了；当结束为 0 时表示乘法和除法还没有完成，还需要继续暂停）。

ex_result 为再 EX 段计算出的结果，如果 mfhi 指令，那么将从 hilo 寄存器模块中读出的 hi 寄存器中的值赋给 ex_result；如果 mflo 指令，那么将从 hilo 寄存器模块中读出的 lo 寄存器中的值赋给 ex_result；如果都不是，就将从加法器 alu 模块中计算出的结果赋值给 ex_result。

ex_to_id_bus 负责将 ex 段的指令码，写入使能，写入地址和写入数据返回给 ID 段的数据通路，用来判断是否存在 RAW 等数据相关。

2.3.4 结构示意图



2.4 MEM 段

2.4.1 整体功能说明

访存是指存储器访问指令将数据从存储器中读出，或者写入存储器的过程，在 MEM 段进行对内存数据的读入，通过判断，决定使用的数据是从内存中传入的还是从 EX 段传入的，并将其传到 WB 段。

2.4.2 端口，信号介绍

输入：clk：传入时钟周期；rst：复位信号，负责初始化各项数据；stall：停止信号，负责暂停流水线；ex_to_mem_bus：EX 段到 MEM 段总线；data_sram_rdata：从内存中读入的数据。

输出：mem_to_wb_bus：MEM 段到 WB 段的总线；mem_to_id_bus：MEM 段到 ID 段的总线 。

2.4.3 功能模块说明

ex_to_mem_bus：负责获得从 EX 段传来的总线；

rf_wdata：存放数据，根据 sel_rf_res 进行判断是使用 MEM 段算出的数据还是从 EX 传来的数据；

mem_to_wb_bus：从 MEM 段传到 WB 段的总线；

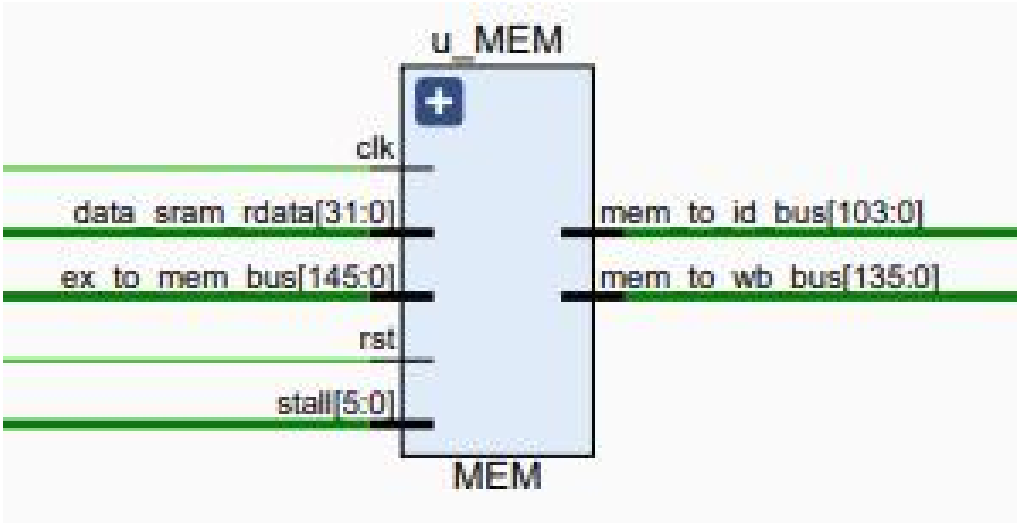
mem_to_id_bus：从 MEM 段传回 ID 段的总线；

mem_result：将从内存中取出来的数据存入到 rt 寄存器中；

always 模块：若复位信号为 1 或暂停信号为 1 时，将 ex_to_mem_bus_r 赋值为 0，否则将 ex_to_mem_bus_r 用从 EX 段传来的总线赋值;

其中，存入 rt 寄存器的结果进行分类讨论：

2.4.4 结构示意图



2.5 WB 段

2.5.1 整体功能说明

写回是指将指令执行的结果写回通用寄存器组的过程，负责将数据写回。

2.5.2 端口，信号介绍

输入：clk：传入时钟周期；rst：复位信号，负责初始化各项数据；stall：停止信号，负责暂停流水线；mem_to_wb_bus：MEM 段到 WB 段的总线。

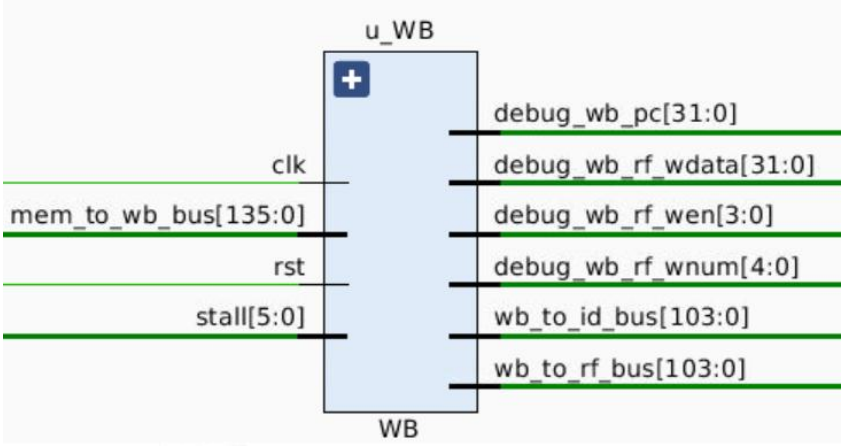
输出：wb_to_rf_bus：WB 段到 ID 段的总线；debug_wb_pc：当前阶段的指令；debug_wb_rf_wen：当前阶段的写入使能；debug_wb_rf_wnum：当前的写入地址；debug_wb_rf_wdata：当前的写入数据。

2.5.3 功能模块说明

mem_to_wb_bus_r：负责获得从 MEM 段传来的总线；
wb_to_rf_bus：从 WB 段传回 ID 段的总线；
debug_wb_pc：用来检测当前的 pc 值是否正确；
debug_wb_rf_wnum：用来检测当前的写入的地址值是否正确；
debug_wb_rf_wdata：用来检测当前的写入的数据是否正确；

always 模块：若复位信号为 1 或暂停信号为 1 时，将 mem_to_wb_bus_r 赋值为 0，否则将 mem_to_wb_bus_r 用从 MEM 段传来的总线赋值。

2.5.4 结构示意图



3. 实验感受，改进意见

3.1 李文丽感受及意见

通过本次实验，我充分理解了我们所学的流水线相关的理论知识，学习到了很多东西。在这次实验之前，我对于知识更多的是简单的概念加上简单的理解，而我做完该实验之后对于很多知识已经有了形象地理解。首先接触并学习到了 verilog 这门十分使用的语言，也学会了使用 vivado 仿真工具进行仿真，如果通过波形图去进行 debug 及查看错误。然后我对搭建 CPU 的基本流程有了更深刻的理解，了解了数据之间的具体流动过程，如何通过正确搭建数据通路解决数据相关问题。在做实验的过程中我们遍历了每一个指令，对于这些指令在流水线中每一阶段所进行的操作都有了比较详细的了解，这对于我更加深刻的理解 CPU 组成原理有很大的帮助。

3.2 周萌萌感受及意见

通过本次实验，我对 CPU 的基本原理与其相关的流水线有了进一步的了解与认识，摆脱了之前只在书本上了解到的片面认知。在进行实验的过程中，我进一步熟悉了多路选择器、译码器等组件的使用，培养了我对于汇编代码能够进行

简单阅读与分析从而得出每步相应结果的能力，也养成了我Vivado 运行出错时，能够尝试去寻找波形图中的错误，从而使问题得到解决的能力。本次实验使我巩固了课上所讲授的知识，让我不止停留于理论，而是更加关注实际实践操作，使我看到了自己的差距和经验的不足，以后需要勤奋的学习的同时更应多注重实际的运用。最后，感谢助教在实验过程的全程陪伴以及悉心教导。

4 参考资料

1. 《A03_“系统能力培养大赛”MIPS 指令系统规范_v1.01》
2. 《A07_vivado 使用说明_v1.00》
3. 《A09_CPU 仿真调试说明_v1.00》
4. 《A11_Trace 比对机制使用说明_v1.00》