实验一报告

学号 2016K8009915009
姓名　　钟 赟

一、实验任务（10%）

本次实验的任是将组成原理研讨课上完成的多周期 CPU 设计迁移到本学期实验平台上，修改 myCPU 顶层接口，使多周期 CPU 至少支持 19 条机器指令：LUI、ADDU、ADDIU、SUBU、SLT、SLTU、AND、 OR、XOR、NOR、SLL、SRL、SRA、LW、SW、BEQ、BNE、JAL、JR。

验证思路：在 windows 下运行 Vivado，进行仿真验证，观察是否有明显的错误。若没有，则观察 Vivado 控制台是否打印 PASS。若是，就上板运行 FPGA 验证，观察 LED 灯在开始、执行过程中和结束时的亮暗情况，如果符合预期结果，则验证成功。

二、实验设计（30%）

(一)　总体设计思路

本实验需要实现四个模块：顶层模块、寄存器堆模块、算术逻辑单元模块和控制单元模块。

(二)　顶层模块设计

1. 工作原理

顶层模块 mycpu_top.v 的功能为调用其它三个模块，并实现 pc 的更新、状态机、数据写回的功能。

2. 接口定义

表 1 顶层模块接口定义

名称	方向	位宽	功能描述
时钟信号和复位信号			
clk	IN	1	时钟信号，来自 clk_pll 的输出时钟
resetn	OUT	1	复位信号，低电平同步复位
取指端访存接口			
inst_sram_en	OUT	1	ram 读使能信号，高电平有效
inst_sram_wen	OUT	4	ram 字节写使能信号，高电平有效
inst_sram_addr	OUT	32	ram 读写地址，字节寻址
inst_sram_wdata	OUT	32	ram 写数据
inst_sram_rdata	IN	32	ram 读数据
数据端访存接口			
data_sram_en	OUT	1	ram 使能信号，高电平有效
data_sram_wen	OUT	4	ram 字节写使能信号，高电平有效
data_sram_addr	OUT	32	ram 读写地址，字节寻址
data_sram_wdata	OUT	32	ram 写数据
data_sram_rdata	IN	32	ram 读数据
debug 信号，供验证平台使用			

名称	方向	位宽	功能描述
debug_wb_pc	OUT	32	写回级（多周期最后一级）的 PC，需要 mycpu 里将 PC 一路带到写回级
debug_wb_rf_wen	OUT	4	写回级写寄存器堆(regfiles)的写使能，为字节写使能，如果 mycpu 写 regfiles 为单字节写使能，则将写使能扩展成 4 位即可。
debug_wb_rf_wnum	OUT	5	写回级写 regfiles 的目的寄存器号
debug_wb_rf_wdata	OUT	32	写回级写 regfiles 的写数据

3. 功能描述

指令有 7 个状态，分别为：IF(取指), IW(指令等待), ID_EX(译码和执行), LD(内存读), RDW(读数据等待), WB(写回), ST(内存写)。

采用“三段式”的结构书写。第一段为状态寄存器，用 always 时序逻辑，采用 D 触发器原理，作用为触发状态的改变，它的输入来自于 next_state，当时钟上升沿来临时进行状态切换，当 resetn 为 1 时，回到起始状态 IF。第二段为状态转移，用 always 组合逻辑，根据当前状态和输入信号，描述下一状态的计算逻辑。状态之间的转移参见图二（此处未详细描写状态转移的条件，待补充，因为这部分代码还有一些 bug，还未完全解决）。

第三段为输出逻辑，用 always 时序逻辑，根据状态机当前状态，描述出不同输出寄存器的同步变化。

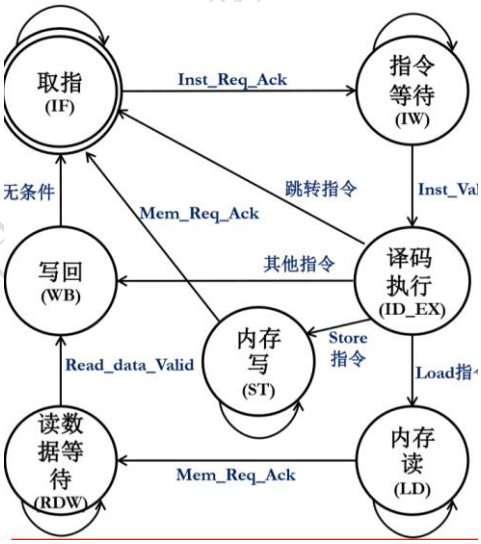

图 1　MIPS 处理器各阶段状态转移图（指令转移条件有所不同）

(三)　alu 模块设计

1. 工作原理

算术逻辑单元模块 alu.v，主要完成 CPU 的算术与逻辑运算。

2. 接口定义

表 2 alu 模块接口定义

名称	方向	位宽	功能描述
A	IN	32	运算操作数 A
B	IN	32	运算操作数 B
ALUop	IN	4	运算器的进行不同运算时的操作码
Zero	OUT	1	运算结果标志，结果为 0 时 Zero 为 1，否则为 0
ALU_Result	OUT	32	ALU 的运算结果

3. 功能描述

运用组合逻辑，将输入 A、输入 B 进行各种运算的结果赋值给各个对应的 wire 变量。此处需要注意，负数应使用补码表示。

最后，根据不同的 ALUop 操作码，将对应运算的结果赋值给最终的运算结果 ALU_Result，再将 ALU_Result 输出。

(四) 寄存器模块设计

1. 工作原理

寄存器组单元模块 reg_file.v，用于存放从内存读出的指令和数据，以及指令的运算结果。

2. 接口定义

表 3 寄存器模块接口定义

名称	方向	位宽	功能描述
clk	IN	1	时钟信号
rst	IN	1	复位信号
waddr	IN	5	写入数据的寄存器地址端口
raddr1	IN	5	rs 寄存器地址输入端口
raddr2	IN	5	rt 寄存器地址输入端口
wen	IN	1	寄存器组的写使能信号
wdata	IN	32	写入寄存器的数据输入端口
reg_rdata1	OUT	32	rs 寄存器数据输出端口
reg_rdata2	OUT	32	rt 寄存器数据输出端口

3. 功能描述

在时钟信号 clk 和复位信号 rst 控制的时序逻辑下，如果复位信号为 1，则将寄存器组数据清零复位；否则，当寄存器写使能 wen 为 1 时，将结果写入 waddr 对应的寄存器中。此处需要注意一点，不能向 0 号寄存器写入数据，因此需在 rd != 0 的条件下才能进行写寄存器的操作。

运用组合逻辑，根据指令中的 rs 段和 rt 段为寄存器编号，向对应的寄存器中获取数据，得到 reg_rdata1 和 reg_rdata2。最后将获取的数据 reg_rdata1 和 reg_rdata2 输出。

(五) 控制单元模块设计

1. 工作原理

控制单元单元模块 ctrl_unit.v，通过对不同的指令进行译码生成各种控制信号。

2. 接口定义

表 4 控制单元模块接口定义

名称	方向	位宽	功能描述
op	IN	6	指令的 31~26 位，控制操作码
func	IN	6	指令的 5~0 位，控制操作码
rt	IN	5	指令的 20~16 位，
Zero	IN	1	ALU 运算结果标志，结果为 0 时 Zero 为 1，否则为 0
ea	IN	32	即 ALU_Result，其 1~0 位控制移位运算
reg_rdata1	IN	32	rs 寄存器数据输出端口
Ctrl	OUT	13	输出的控制信号
ALUop	OUT	4	ALU(算逻运算单元)操作码
Write_strb	OUT	4	ram 字节写使能信号，高电平有效

3. 功能描述

利用 case 语句，根据操作码的变化确定指令，查阅 MIPS 手册中指令的定义，为各个信号赋值。各个控制信号的定义如下表，信号的传输路径可参考图二中黄线部分。最后将各个控制信号输出。

表 5 控制信号功能定义

Ctrl	信号	位宽	功能
Ctrl[12]	Jump	1	JUMP 指令操作标志，执行 JUMP 指令时为 1 ，否则为 0
Ctrl[11]	Jr	1	JR 指令操作标志，执行 JR 指令时为 1 ，否则为 0
Ctrl[10]	Jal	1	JAL 指令操作标志，执行 JAL 指令时为 1 ，否则为 0
Ctrl[9]	Jalr	1	JALR 指令操作标志，执行 JALR 指令时为 1 ，否则为 0
Ctrl[8]	Mem2Reg	1	控制写回寄存器组寄存器的数据，为 1 时来自数据 ram，为 0 时来自 ALU_Result
Ctrl[7]	Store	1	存数操作标志，相关指令有 SW,SWL,SWR,SB,SH
Ctrl[6]	RegDst	1	控制写寄存器组寄存器的地址
Ctrl[5]	RegWrite	1	寄存器组写使能信号，高电平有效
Ctrl[4]	PCsrc	1	控制 PC 的跳转，用于 BEQ 和 BNE 指令
Ctrl[3]	ExtSel	1	将 16 位立即操作数扩展为 32 位，为 1 时符号扩展，为 0 时五符号扩展
Ctrl[2]	ALUsrcB	1	控制 ALU 运算的操作数 B，为 0 时来自 rdata2，为 1 时来自立即数
Ctrl[1:0]	ALUsrcA	2	控制 ALU 运算的操作数 A，低位为 1 时，操作数 A 为 32’d0，高位为 1 时，操作数 A 为 sa，全为 0 时，操作数 A 为 reg_rdata1

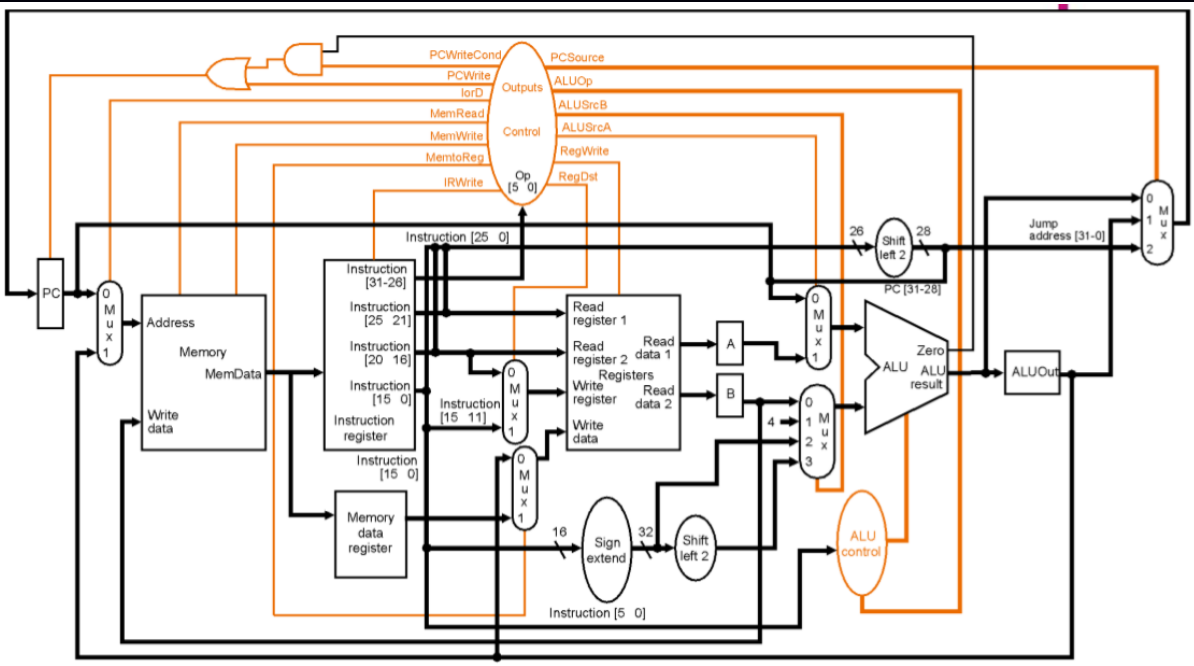

图 2 多周期数据通路（部分信号与具体实现稍有不同）

三、实验过程（60%）

（一）实验流水账

2018-09-08 19:00-21:00 阅读体系结果实验讲义，初步设计 CPU。

2018-09-09 21:00-10:00 按照讲义接口定义初步修改 CPU。

2018-09-11 21:00-12:30 按照实验课所讲代码风格修改代码，重新编写部分代码（主要是控制单元模块）。

2018-09-13 09:30-12:00 修改代码风格，开始 debug。找出部分 bug，但仿真还未通过。

2018-09-14 18:00-21:00 继续代码 debug。

2018-09-15 21:00-23:00 继续代码 debug。

2018-09-17 20:00-01:00 debug 终于结束，仿真通过，上板通过。

（二）错误记录

1、错误 1

（1）错误现象

第二条指令 debug_wb_rf_wnum 信号没有波形。

（2）分析定位过程

由波形知第二条指令无法正确执行，于是定位到 PC 无法正确更新，发现 debug_wb_pc 与 trace 不吻合。

（3）错误原因

debug_wb_pc 不应该只在 WB 状态更新，因为部分指令的执行过程不涉及 WB，因此需在每条指令执行的最后一个状态更新 debug_wb_pc。

（4）修正效果

指令执行过程的状态转移有四种类型：

IF→IW→ID_EX(更新 debug_wb_pc)→IF

IF→IW→ID_EX→WB(更新 debug_wb_pc)→IF

IF→IW→ID_EX→ST(更新 debug_wb_pc)→IF

IF→IW→ID_EX→LD→RDW→WB(更新 debug_wb_pc)→IF

应该分别在该类型指令执行的最后一个状态更新 debug_wb_pc。修正后 debug_wb_rf_wnum 信号有了波形。

（5）归纳总结（可选）

应在设计阶段将 pc 写回的时间考虑清楚。

2、错误 2

（1）错误现象

仿真时 CPU 持续执行指令 0xbfc0058c

（2）分析定位过程

根据反汇编得知该指令为 LW 指令。根据现象初步分析产生了电路环。

（3）错误原因

RDW 状态跳转的控制信号错误，控制信号不满足，所以一直停留在 RDW 状态。

（4）修正效果

RDW 状态直接跳转到 WB 状态，不设跳转条件。修正后程序正常执行。

（5）归纳总结（可选）

修改信号名称时手误。

3、错误 3

（1）错误现象

BNE 跳转错误，在不该跳转的地方产生跳转。

（2）分析定位过程

检查得知 BNE 指令没有错误，他的前一个指令 LW 写回寄存器的数据错误。

（3）错误原因

指令读使能信号提前拉高，导致 LW 执行时提前将下一条指令的内存数据写入寄存器。

（4）修正效果

在 inst_sram_en 拉高的下一拍将其拉低，避免指令提前被读取。修正后该处没有报错。

（5）归纳总结（可选）

后来得知，事实上此处的 BNE 指令检查没有很细致，后面因为这条指令再次报错（见错误 5）。

4、错误 4

（1）错误现象

SLL 指令运算结果错误

（2）分析定位过程

根据报错位置，找到对应的反汇编指令为 SLL，检查 ALU 模块中的 SLL 运算编写是否与 MIPS 指令手册的定义一致。

（3）错误原因

ALU 运算单元的操作数 A 的控制信号编写错误，SLL 左移指令的操作数 A 应该来自内存指令 10~6 位，即 sa，而不是 rt 寄存器，错将 SLL 和 SLLV 指令的操作数 A 弄反。

（4）修正效果

改写选择控制信号 ALUsrcA，在执行 SLL 指令时将操作数 A 赋值为 sa，同时修正了 SRL、SRA 指令(操作数 A 为 sa)，和 SLLV、SRLV、SRAV 指令(操作数 A 来自寄存器组寄存器 rt)。修正后 SLL 指令没有报错。

（5）归纳总结（可选）

低级错误。写控制信号时应仔细与 MIPS 手册比对。

5、错误 5

（1）错误现象

PC 持续跑到很大的值，超过反汇编的 PC 最大值。

```
[16702000 ns] Test is running, debug_wb_pc = 0xbfcffd68
[16712000 ns] Test is running, debug_wb_pc = 0xbfcfff5c
```

图 3 错误 5 现象截图

（2）分析定位过程

debug 是根据测试的功能点每隔一段时间打印一次 PASS，寻找最后一次打印 PASS 的位置，从该处比对 PC，定位异常的位置，发现在一处跳转指令 BNE 处 PC 跳转到异常值。分支跳转指令 BNE 执行之后，PC 跳转到范围外，一直执行下去，并且不能触发比对。

（3）错误原因

BNE 指令的控制信号编写错误，跳转的目标地址计算时所用的 16 位立即数未采用符号扩展。

（4）修正效果

将执行 BNE 指令时的符号扩展控制信号 ExtSel 置为 1，表示符号扩展。

（5）归纳总结（可选）

低级错误，写控制信号时应仔细与 MIPS 手册比对。

四、实验总结（可选）

本次实验主要在于修改，在修改接口和变量名的过程中由于手误给 debug 过程带来了很多不必要的麻烦。但是在仿真过程中由于可以比较精确地定位 bug，比起上学期组成原理实验 debug 会方便一些。另外，尝试优化了一下代码风格，但还不够完美。