

计算机组成原理

课程设计报告

学 号_____20020203_____

姓 名_____王思哲_____

指导教师_____魏坚华_____

提交日期_____2022 年 7 月 5 日_____

成绩评价表

报告内容	报告结构	报告最终成绩
<input type="checkbox"/> 丰富正确 <input type="checkbox"/> 基本正确 <input type="checkbox"/> 有一些问题 <input type="checkbox"/> 问题很大	<input type="checkbox"/> 完全符合要求 <input type="checkbox"/> 基本符合要求 <input type="checkbox"/> 有比较多的缺陷 <input type="checkbox"/> 完全不符合要求	
报告与大作业功能一致性	报告图表	总体评价
<input type="checkbox"/> 完全一致 <input type="checkbox"/> 基本一致 <input type="checkbox"/> 基本不一致	<input type="checkbox"/> 符合规范 <input type="checkbox"/> 基本符合规范 <input type="checkbox"/> 有一些错误 <input type="checkbox"/> 完全不正确	

教师签字:_____

目录

Project1 VerilogHDL 完成多周期处理器开发	6
一、总体数据通路结构设计图	6
二、模块定义	6
2.1 ALU 模块	6
2.1.1 基本描述	6
2.1.2 模块接口	6
2.1.3 功能定义	7
2.2 NPC 模块	7
2.2.1 基本描述	7
2.2.2 模块接口	7
2.2.3 功能定义	7
2.3 PC 模块	8
2.3.1 基本描述	8
2.3.2 模块接口	8
2.3.3 功能定义	8
2.4 IM 模块	8
2.4.1 基本描述	8
2.4.2 模块接口	9
2.4.3 功能定义	9
2.5 IR 模块	9
2.5.1 基本描述	9
2.5.2 模块接口	9
2.5.3 功能定义	9
2.6 GPR 模块	10
2.6.1 基本描述	10
2.6.2 模块接口	10
2.6.3 功能定义	10
2.7 EXT 模块	11
2.7.1 基本描述	11
2.7.2 模块接口	11
2.7.3 功能定义	11
2.8 aReg & bReg 模块	11
2.8.1 基本描述	11
2.8.2 模块接口(以 aReg 模块为例)	11
2.8.3 功能定义	12
2.9 aluReg 模块	12
2.9.1 基本描述	12
2.9.2 模块接口	12
2.9.3 功能定义	12
2.10 DM 模块	12
2.10.1 基本描述	12
2.10.2 模块接口	13
2.10.3 功能定义	13

2.11	DR 模块.....	13
2.11.1	基本描述.....	13
2.11.2	模块接口.....	13
2.11.3	功能定义.....	13
2.12	Controller 模块.....	14
2.12.1	基本描述.....	14
2.12.2	模块接口.....	14
2.12.3	功能定义.....	15
2.13	Muxdmin 模块.....	15
2.13.1	基本描述.....	15
2.13.2	模块接口.....	16
2.13.3	功能定义.....	16
三、指令描述		16
四、测试程序		17
4.1	MIPS-Lite2 指令集的测试程序.....	17
4.2	新增指令 BLTZAL 的测试程序.....	18
五、测试结果		18
5.1	MIPS-Lite2 指令集测试结果.....	18
5.1.1	MARS 中结果.....	18
5.1.2	Modelsim 中结果.....	19
5.2	BLTZAL 指令测试结果.....	20
5.2.1	MARS 中结果.....	20
5.2.2	Modelsim 中结果.....	21
六、总结与心得体会		21
Project2 VerilogHDL 完成 MIPS 微系统开发(支持设备与中断).....		22
一、	总体数据通路结构设计图.....	22
二、	模块定义.....	23
2.1	ALU 模块.....	23
2.1.1	基本描述.....	23
2.1.2	模块接口.....	23
2.1.3	功能定义.....	23
2.2	NPC 模块.....	23
2.2.1	基本描述.....	23
2.2.2	模块接口.....	24
2.2.3	功能定义.....	24
2.3	PC 模块.....	25
2.3.1	基本描述.....	25
2.3.2	模块接口.....	25
2.3.3	功能定义.....	25
2.4	IM 模块.....	25
2.4.1	基本描述.....	25
2.4.2	模块接口.....	25
2.4.3	功能定义.....	26
2.5	IR 模块.....	26

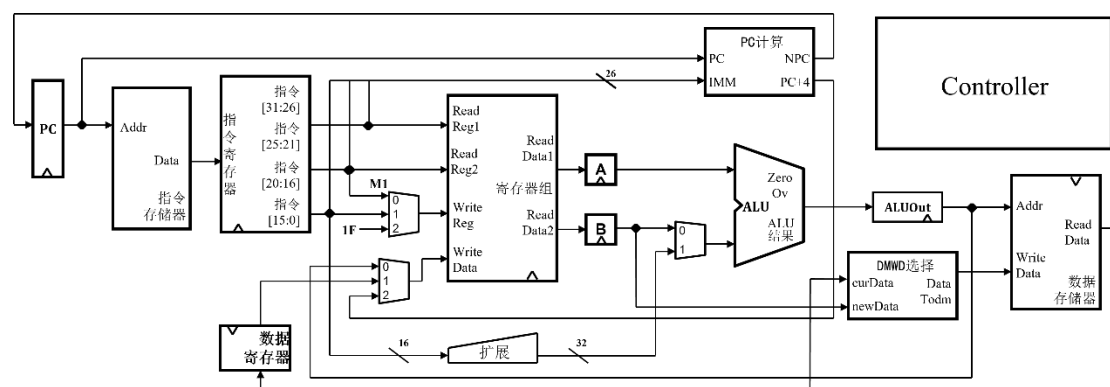
2.5.1	基本描述.....	26
2.5.2	模块接口.....	26
2.5.3	功能定义.....	26
2.6	GPR 模块.....	26
2.6.1	基本描述.....	26
2.6.2	模块接口.....	27
2.6.3	功能定义.....	27
2.7	EXT 模块.....	27
2.7.1	基本描述.....	27
2.7.2	模块接口.....	28
2.7.3	功能定义.....	28
2.8	aReg & bReg 模块.....	28
2.8.1	基本描述.....	28
2.8.2	模块接口 (以 aReg 模块为例)	28
2.8.3	功能定义.....	28
2.9	aluReg 模块.....	29
2.9.1	基本描述.....	29
2.9.2	模块接口.....	29
2.9.3	功能定义.....	29
2.10	DM 模块.....	29
2.10.1	基本描述.....	29
2.10.2	模块接口.....	29
2.10.3	功能定义.....	30
2.11	DR 模块.....	30
2.11.1	基本描述.....	30
2.11.2	模块接口.....	30
2.11.3	功能定义.....	30
2.12	Controller 模块.....	31
2.12.1	基本描述.....	31
2.12.2	模块接口.....	32
2.12.3	功能定义.....	33
2.13	Muxdmin 模块.....	33
2.13.1	基本描述.....	33
2.13.2	模块接口.....	33
2.13.3	功能定义.....	33
2.14	muxlw 模块.....	34
2.14.1	基本描述.....	34
2.14.2	模块接口.....	34
2.14.3	功能定义.....	34
2.15	bridge 模块.....	34
2.15.1	基本描述.....	34
2.15.2	模块接口.....	34
2.15.3	功能定义.....	35
2.16	cp0 模块.....	35

2.16.1	基本描述.....	35
2.16.2	模块接口.....	35
2.16.3	功能定义.....	36
2.17	timer 模块.....	36
2.17.1	基本描述.....	36
2.17.2	模块接口.....	36
2.17.3	功能定义.....	37
2.18	outputDev 模块.....	37
2.18.1	基本描述.....	37
2.18.2	模块接口.....	37
2.18.3	功能定义.....	37
2.19	输入模块.....	38
三、	指令描述.....	38
四、	测试程序.....	38
4.1	MIPS-Lite3 指令集的测试主程序.....	38
4.2	MIPS-Lite3 指令集的测试中断子程序.....	39
五、	测试结果.....	39
5.1	波形.....	39
5.2	寄存器组结果.....	39
5.2.1	gpr 寄存器.....	39
5.2.2	cp0 寄存器.....	40
六、	总结与心得.....	40

Project1 VerilogHDL 完成多周期处理器开发

一、总体数据通路结构设计图

下图为本次多周期处理器开发的数据通路设计图。本次的数据通路设计，是在 ppt 中给出的数据通路上进行了改进:为了适配 sb 指令，在 dm 模块之前新增 DMWD 选择模块。



二、模块定义

2.1 ALU 模块

2.1.1 基本描述

ALU 模块的功能有两个。一个是选择第二个操作数的来源，另一个是完成数据的运算并输出计算结果以及一些其他模块需要的标志位。

2.1.2 模块接口

信号名	方向	描述
[31:0]dataOut_1	I	GPR 输出数据 1 的值，来自 aReg 模块
[31:0]dataOut_2	I	GPR 输出数据 2 的值，来自 bReg 模块
[31:0]ext32	I	16 位立即数扩展结果
[1:0]ALUOp	I	运算方式选择信号 00: + 01: - 10: 11: slt
ALUSrc	I	参与运算的第二个操作数的选择信号 0: dataOut_2 1: ext32
write_30	I	是否为 addi 标志 0: 不是 1: 是
zero	O	运算结果是否为 0 标志 0: 不是 1: 是
overflow	O	addi 运算是否产生溢出标志 0: 不是 1: 是
[31:0]alu_res	O	运算结果
nCondition	O	第一个操作数是否小于 0 标志 (bltza1 指令)

		0: 否 1: 是
--	--	-----------

2.1.3 功能定义

序号	功能名称	功能描述
1	数据选择	选择正确的数据作为第二个操作数的值
2	运算并输出	根据指定运算类型完成运算，并输出结果和一些标志位

2.2 NPC 模块

2.2.1 基本描述

NPC 模块主要完成的功能是下一条 pc 地址的生成，并根据选择信号选择正确的地址输出给 pc 模块。

2.2.2 模块接口

信号名	方向	描述
rst	I	复位信号 0: 不复位 1: 复位
zero	I	两操作数相等信号，用于选择下一条地址
nCondition	I	指定寄存器操作数小于 0 信号，用于选择下一条寄存器
[1:0]npc_sel	I	选择输出的 pc 00: pc_ori+4 01: 按照 beq 跳转方式计算地址 10: 按照 j 跳转方式计算地址 11: 选择 jr 地址
[31:0]pcin	I	gpr 中 regFile[rs]值
[31:0]pc_ori	I	pc
[31:0]imm32	I	当前指令的 32 位值，只用到低 26 位
[31:0]pcout	O	npc 输出

2.2.3 功能定义

序号	功能名称	功能描述
1	复位	当复位信号有效时，pcout \leftarrow 32'h0000_3000
2	计算指令地	npc_sel:

	址，选择并输出 pc	00: $pcout \leftarrow pc_ori+4$ 01:如果 zero 或 nCondition 有效，则 pcout 按照 beq 跳转方式更新 10:按照 j 跳转方式更新 11:选择 jr 跳转地址并更新
--	------------	--

2.3 PC 模块

2.3.1 基本描述

PC 模块的主要功能是，将 NPC 模块的输出（下一条指令地址），在时钟上升沿且写使能有效时，将输出的值更改为输入的值。

2.3.2 模块接口

信号名	方向	描述
clk	I	时钟信号
pcwr	I	写使能信号 0: 不写入 1: 写入
[31:0]pcin	I	输入 npc 模块的输出，下一条指令的地址
[31:0]pcout	O	下一条指令的地址，PC 模块寄存器存储的值

2.3.3 功能定义

序号	功能名称	功能描述
1	存下一条指令地址	在时钟沿上升沿且写使能有效时，将当前输出寄存器的值更改为当前输入值
2	输出下一条指令地址	输出当前寄存器的值

2.4 IM 模块

2.4.1 基本描述

指令存储器，主要功能有两个。一个存储指令，另一个是根据输入地址的值，输出对应地址的指令。

2.4.2 模块接口

信号名	方向	描述
[9:0]addr	I	输入地址（指令地址）
[31:0]dout	O	地址对应的指令

2.4.3 功能定义

序号	功能名称	功能描述
1	存储指令	读取 txt 文件，存储指令
2	根据地址输出指令	根据模块的地址输入，输出对应的指令

2.5 IR 模块

2.5.1 基本描述

IR 模块的主要功能是切分数据通路。这个模块本质上是一个寄存器，用于存储 IM 输出的指令。IR 模块的写入代表取指阶段的完成。

2.5.2 模块接口

信号名	方向	描述
clk	I	时钟信号
irwr	I	写使能信号 0：不写入 1：写入
[31:0]imin	I	输入为 IR 模块的输出，是一条指令
[31:0]irout	O	输出寄存器中的指令

2.5.3 功能定义

序号	功能名称	功能描述
1	切分数据通路	在时钟沿上升沿且写使能有效时，将当前输出寄存器的值更改为当前输入值

2.6 GPR 模块

2.6.1 基本描述

GPR 模块的主要功能分三点。第一是数据选择器功能，选择正确的写入数据和正确的写入地址。第二个是数据的写入功能。第三个是数据的输出功能。

2.6.2 模块接口

信号名	方向	描述
clk	I	时钟信号
rst	I	复位信号 0: 不复位 1: 复位
gprwr	I	gpr 写使能信号 0: 不允许写入 1: 允许写入
[1:0]MemToReg	I	选择信号，选择写入数据 00: alu 运算结果 01: lw/lb 数据 10: 下一条指令地址
[1:0]RegDst	I	选择信号，选择写入地址 00:rt 01:rd 10:no.31
[4:0]rs	I	读出地址 1
[4:0]rt	I	读出地址 2
[4:0]rd	I	写入地址的一种选择
write_30	I	addi 指令标志 0: 不是 1: 是
[31:0]pc_p4	I	写入数据的一种选择，下一条指令地址
[31:0]aluReg_out	I	写入数据的一种选择，alu 的运算结果
[31:0]dmReg_out	I	写入数据的一种选择，lw/lb 数据
overflow	I	溢出写入数据 0: 不溢出写入 0 1: 溢出写入 1
[31:0]dataOut_1	O	rs 寄存器的值，数据输出
[31:0]dataOut_2	O	rt 寄存器的值，数据输出

2.6.3 功能定义

序号	功能名称	功能描述
1	数据选择	根据选择信号选择正确的写入数据和写入地址
2	寄存器存储数据	将数据写入对应寄存器，存储
3	输出数据	根据读出地址，读出对应的数据

2.7 EXT 模块

2.7.1 基本描述

EXT 模块的主要功能是，将 16 位立即数进行扩展。扩展的方式有三种，第一种是 0 扩展，第二种是符号扩展，第三种是将 16 位立即数加载到 32 位的高 16 位，低位补 0。

2.7.2 模块接口

信号名	方向	描述
[15:0]imm16	I	16 位立即数输入
[1:0]ExtOp	I	扩展方式选择信号 00: 0 扩展 01: 符号扩展 10: lui 指令方式扩展
[31:0]ext32	O	扩展结果输出

2.7.3 功能定义

序号	功能名称	功能描述
1	扩展立即数	根据扩展方式的选择信号，扩展 16 位立即数至 32 位。

2.8 aReg & bReg 模块

2.8.1 基本描述

aReg & bReg 模块实现的是同样的功能，即切分数据通路。他们存储的是 GPR 模块的两个输出数据。当两个输出数据写入两个模块的寄存器时，代表译码阶段结束。

2.8.2 模块接口(以 aReg 模块为例)

信号名	方向	描述
clk	I	时钟信号
[31:0]alu_dataOut_1	I	GPR 模块读出数据 1
[31:0]aReg_out	O	输出寄存器值

2.8.3 功能定义

序号	功能名称	功能描述
1	切分数据通路	在时钟沿上升沿时，将当前输出寄存器的值更改为当前输入值。

2.9 aluReg 模块

2.9.1 基本描述

aluReg 模块主要功能是切分数据通路。当时钟信号上升沿时，将输入数据写入模块中的寄存器。

2.9.2 模块接口

信号名	方向	描述
clk	I	时钟信号
[31:0]alu_res	I	ALU 模块运算结果
overflow	I	ALU 模块溢出标志
[31:0]aluReg_out	O	当前模块输出的 ALU 模块运算结果
overflowReg	O	当前模块输出的 ALU 模块溢出标志

2.9.3 功能定义

序号	功能名称	功能描述
1	切分数据通路	在时钟沿上升沿时，将当前输出寄存器的值更改为当前输入值。

2.10 DM 模块

2.10.1 基本描述

DM 模块的功能有两个。一是在写使能有效且时钟上升沿时，将输入数据写入到输入地址的存储空间中。二是在读出输入地址在存储空间中的数据。

2.10.2 模块接口

信号名	方向	描述
[9:0]addr	I	写入地址
[31:0]din	I	写入数据
we	I	写使能 0: 不写入 1: 允许写入
clk	I	时钟信号
[31:0]dout	O	读出数据

2.10.3 功能定义

序号	功能名称	功能描述
1	写入数据	在时钟沿上升沿且写使能有效时，写入数据
2	读出数据	读出指定地址的数据

2.11 DR 模块

2.11.1 基本描述

DR 模块的主要功能是切分数据通路，还有一个功能是选择输出数据（针对 1b 指令）。当时钟上升沿时，将输入数据写入到模块寄存器。

2.11.2 模块接口

信号名	方向	描述
clk	I	时钟信号
islb	I	1b 指令标志 0: 不是 1: 是
[31:0]dmout	O	DM 模块读出的数据
[31:0]drout	O	当前模块的输出值

2.11.3 功能定义

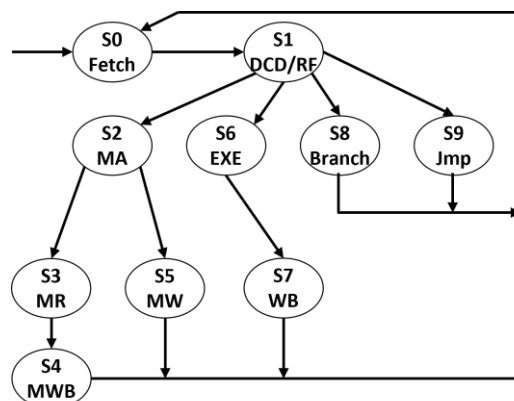
序号	功能名称	功能描述
1	切分数据通路	在时钟沿上升沿时，将当前输出寄存器的值更改为当前输入值。
2	选择输出数据	根据 isl b 信号，选择或生成正确的数据并输出。

2.12 Controller 模块

2.12.1 基本描述

Controller 模块是根据输入的 opcode 和 funct 值（还有 zero 信号），生成对应的指令。同时构建状态机，在对应的状态时，产生对应的控制信号。

状态机共有 10 个状态，为 S0 至 S9，状态机如下图所示：



S0 和 S1 是所有指令都要经历的取指和译码阶段。lw/sw/lb/sb 走 S2，beq/jr/bltzal 走 S8，j/jal 走 S9，其余指令走 S6。

lw/lb 指令由于访存后有回写阶段，故走 S3 读 dm，然后 S4 回写。sw/sb 指令则走 S5 访存（写入）阶段。

S6 阶段是执行阶段，进行 alu 的运算，为 R 型和 I 型指令。然后走到 S7，进行回写阶段。

j/jal 指令是直接跳转，故分到同一个状态 S9。需要注意的是，S9 阶段，如果是 jal 指令，是需要执行回写的，此时的形成跳转地址与回写是在同一个状态下完成的。

bltzal 指令与 beq 指令类似，先考察条件。满足判断条件形成跳转地址，故被分到同一个状态 S8。

2.12.2 模块接口

信号名	方向	描述
clk	I	时钟信号
rst	I	复位信号
[5:0]opcode	I	当前指令 opcode
[5:0]funct	I	当前指令 funct
zero	I	ALU 模块产生的 zero 标志
nCondition	I	ALU 模块产生的 nCondition 标志（bltzal 指令）
[1:0]RegDst	O	GPR 模块写入地址选择 00:rt 01:rd 10:no.31
[1:0]MemToReg	O	GPR 模块写入数据选择 00: alu 运算结果 01: lw/lb 数据

		10: 下一条指令地址
[1:0]npc_sel	0	NPC 模块输出选择 00: pc_ori+4 01: 按照 beq 跳转方式计算地址 10: 按照 j 跳转方式计算地址 11: 选择 jr 地址
[1:0]ALUOp	0	ALU 模块运算功能选择 00: + 01: - 10: 11: slt
[1:0]EXTOp	0	EXT 模块扩展类型选择 00: 0 扩展 01: 符号扩展 10: lui 指令方式扩展
RegWrite	0	GPR 模块写使能
ALUSrc	0	ALU 模块操作数 2 的选择 0: GPR 模块读出数据 2 1: 16 位立即数扩展结果
MemWrite	0	DM 模块写使能
write_30	0	addi 指令标志
pcwr	0	PC 模块写使能
irwr	0	IR 模块写使能
islb	0	LB 指令标志
issb	0	SB 指令标志

2.12.3 功能定义

序号	功能名称	功能描述
1	构建状态机， 确定当前状态	按设计好的状态机构建状态，并根据不同的指令确定下一个状态。
2	产生控制信号	在不同的状态下，产生对应的控制信号和标志。

2.13 Muxdmin 模块

2.13.1 基本描述

Muxdmin 模块设计的初衷是为了支持 sb 指令。主要功能是根据 issb 信号，选择或生成正确的写入数据，传送给 DM 模块的写入数据端口。

这里的逻辑是，由于 DM 模块在时钟上升沿来临时才进行写入，所以可以在来临时将需要写入的地址的值取出。如果是 sb 指令，则将需要写入的低 8 位拼上原来的 24 位输出即可。如果不是，则直接将取出的值输出。这一部分是组合逻辑电路，在时钟沿来临时，DM 模块将数据写入即可。

2.13.2 模块接口

信号名	方向	描述
[31:0]bReg_out	I	bReg 寄存器输出
[31:0]dmout_data	I	DM 模块读出数据
issb	I	sb 指令标志
[31:0]dm_datain	O	当前模块数据输出，输出给 DM 模块输入

2.13.3 功能定义

序号	功能名称	功能描述
1	选择或生成 DM 模块写入数据	根据 issb 控制信号，生成或选择 DM 模块的写入数据。

三、指令描述

序号	助记符	opcode	funct	指令功能
1	addu	000000	100001	无符号加法
2	subu	000000	100011	无符号减法
3	ori	001101		或立即数
4	lw	100011		加载字
5	sw	101011		存储字
6	beq	000100		等于时转移
7	lui	001111		立即数加载至高位
8	j	000010		无条件跳转
9	addi	001000		加立即数（有符号）
10	addiu	001001		加立即数（无符号）
11	slt	000000	101010	小于时置1
12	jal	000011		跳转并链接
13	jr	000000	001000	跳转寄存器
14	lb	100000		加载字节
15	sb	101000		存储字节
16	bltzal	000001	10000(rt位置)	小于0时跳转，并链接（存储下一条指令地址）

四、测试程序

4.1 MIPS-Lite2 指令集的测试程序

```
1  ori $16, $0, 1 # 把1与0号寄存器进行或运算, 结果存16号寄存器
2  ori $17, $0, 3 # 把3与0号寄存器进行或运算, 结果存17号寄存器
3  ori $8, $0, 1 # 把1与0号寄存器进行或运算, 结果存8号寄存器
4  ori $12, $0, 0xabab # 十六进制数abab与0号寄存器相或, 结果存12号寄存器
5  lui $13, 10 # 把10加载到32位数的高位, 低位补0, 存13号寄存器
6  start: addu $4, $0, $16 # 0号寄存器内容 和 16号寄存器内容 进行无符号加法, 结果存4号寄存器
7  addu $5, $0, $8 # 8号寄存器内容 和 0号寄存器内容 进行无符号加法, 结果存5号寄存器
8  jal newadd # 跳转到newadd处, 同时保存下一条指令地址到31号寄存器
9  addu $16, $0, $2 # 2号寄存器内容 和 0号寄存器内容 进行无符号加法, 结果存16号寄存器
10 subu $17, $17, $8 # 17号寄存器内容 和 8号寄存器内容 进行无符号减法, 结果存17号寄存器
11 beq $16, $17, start # 比较 16号寄存器内容 和 17号寄存器内容: 相等则跳转到start处
12 ori $8, $0, 4 # 把4与0号寄存器相或, 结果存8号寄存器
13 addiu $24, $0, 0x7fffffff # 0号寄存器内容加立即数7fffffff (十六进制) 的结果, 存入24号寄存器 (无符号加)
14 addiu $9, $24, 3 # 24号寄存器内容与3相加的结果 (无符号加法) 存9号寄存器
15 addiu $10, $24, 5 # 24号寄存器内容与5进行无符号加法的结果, 存10号寄存器
16 addu $0, $0, $0 # 0号寄存器与0号寄存器进行无符号加法的结果存0号寄存器。(理论上0号寄存器不能被写入)
17 #addi $22, $24, 6 # 6与24号寄存器内容进行有符号加法, 结果存入22寄存器, 若产生溢出, 30号寄存器存1 (实验要求)
18 start2: sw $9, -4($8) # 9号寄存器内容存入以8号寄存器内容为基地址, 并向左偏移4的地址的存储空间中
19 sw $1, 0($8) # 1号寄存器内容存入以8号寄存器内容为基地址, 偏移0的地址的存储单元中
20 lb $14, 3($8) # 以8号寄存器内容为基地址, 加上偏移量3后的地址的存储单元内容的低8位, 符号扩展后加载到14号寄存器
21 sb $12, 7($8) # 12号寄存器内容低8位存储到以8号寄存器内容为基地址, 偏移7后的地址的存储单元中
22 lw $15, 4($8) # 以8号寄存器为基地址, 偏移4后的地址的存储单元中的数据, 加载到15号寄存器中
23 sb $4, -3($8) # 4号寄存器内容低8位存入以8号寄存器为基地址, 加上偏移量-3后的地址的存储单元中
24 lb $18, -1($8) # 以8号寄存器为基地址, 加上偏移量-1后的地址的存储单元内容的低8位, 符号扩展后加载到18号寄存器
25 addu $4, $0, $8 # 8号寄存器内容 与 0号寄存器内容 进行无符号加法, 结果存入4号寄存器
26 addu $5, $0, $9 # 9号寄存器内容 与 0号寄存器内容 进行无符号加法, 结果存入5号寄存器

27 jal newadd # 将当前指令地址加4的内容 (下一条指令地址) 存入31号寄存器, 跳转到newadd
28 slt $25, $10, $8 # 如果10号寄存器内容 小于 8号寄存器内容, 25号寄存器内容置1, 否则置0
29 beq $25, $0, end2 # 比较25号寄存器 与 0号寄存器内容 (0), 相等则跳转end2处, 否则顺序执行
30 slt $20, $12, $4 # 如果12号寄存器内容 小于 4号寄存器内容, 20号寄存器内容置1, 否则置0
31 beq $20, $0, end1 # 比较20号寄存器 与 0号寄存器内容 (0), 相等则跳转end1处, 否则顺序执行
32 lui $12, 65535 # 将十进制65535加载到32位的高16位, 低16位补0, 扩展结果存入12号寄存器
33 end1: ori $0, $0, 1 # 0号寄存器内容 与 1 相或, 结果存0号寄存器 (实际上0号寄存器不允许写入)
34 lui $19, 0xefef # 十六进制数efef加载至32位中的高16位, 低位补0, 结果存入19号寄存器
35 addiu $3, $0, 0xababcdcd # 0号寄存器内容 无符号加立即数 ababcdcd (十六进制), 结果存入3号寄存器中
36 start3: addiu $4, $3, 2 # 3号寄存器内容 无符号加立即数2, 结果存入4号寄存器中
37 addi $23, $3, 5 # 3号寄存器内容 加 5, 有符号数的加法, 结果存23号寄存器。若结果溢出, 30号寄存器存1, 否则为0
38 jal newadd # 将下一条指令地址存入31号寄存器后, 跳转到newadd处
39 addu $8, $0, $2 # 0号寄存器内容 与 2号寄存器内容 进行无符号加法, 结果存8号寄存器
40 addu $4, $0, $8 # 0号寄存器内容 与 8号寄存器内容 进行无符号加法, 结果存4号寄存器
41 addu $5, $0, $9 # 0号寄存器内容 与 9号寄存器内容 进行无符号加法, 结果存5号寄存器
42 jal newadd # 将下一条指令地址存入31号寄存器后, 跳转到newadd处
43 addu $9, $0, $2 # 将2号寄存器内容与0号寄存器内容进行无符号加法, 结果存入9号寄存器
44 addu $9, $8, $0 # 将0号寄存器内容与8号寄存器内容进行无符号加法, 结果存入9号寄存器
45 lui $10, 0x69 # 将十六进制数69加载到32位数的高16位, 低位补0, 结果存入10号寄存器
46 beq $8, $9, start4 # 比较8号寄存器 和 9号寄存器的内容, 相等跳转start4处, 否则顺序执行
47 beq $0, $0, start3 # 比较0号寄存器 和 0号寄存器的内容, 相等跳转start3处, 否则顺序执行
48 start4: j end # 无条件跳转end处
49 newadd: addu $2, $4, $5 # 4号寄存器内容 无符号加 5号寄存器内容, 结果存入2号寄存器中
50 addi $0, $12, 0x1234 # 十六进制数1234 和12号寄存器的内容 有符号加法, 结果存0寄存器 (0寄存器理论上无法被写入)
51 jr $31 # 跳转到 31号寄存器存储的内容 所指示的地址处
52 end2: addi $26, $0, 0x5678 # 十六进制数5678 和 0号寄存器内容 有符号加法, 结果存26号寄存器
53 end: # end标志处
```

4.2 新增指令 BLTZAL 的测试程序

```

1  lui $1, 0xffff           # 加载十六进制数ffff到1号寄存器内容高位（负数）
2  bltzal $1, flag1         # 如果1号寄存器内容小于0，跳flag1，同时存下一条地址到31号寄存器；否则顺序执行
3  j jmp1                   # 无条件跳转 jmp1处
4  flag1:ori $7, $0, 6      # 6和0号寄存器进行或运算，结果存7号寄存器
5  j second                 # 无条件跳转second处
6  jmp1:ori $9, $0, 9       # 9和0号寄存器进行或运算，结果存9号寄存器
7  second:ori $2, $0, 3     # 3和0号寄存器进行或运算，结果存2号寄存器
8  bltzal $2, flag2        # 如果2号寄存器内容小于0，跳flag2；同时存下一条地址到31号寄存器；否则顺序执行
9  j jmp2                   # 无条件跳转 jmp2处
10 flag2:ori $10, $0, 9     # 9和0号寄存器进行或运算，结果存10号寄存器
11 j end                    # 无条件跳转end处
12 jmp2:ori $8, $0, 6       # 6和0号寄存器进行或运算，结果存8号寄存器
13 end:                     # end处

```

五、测试结果

5.1 MIPS-Lite2 指令集测试结果

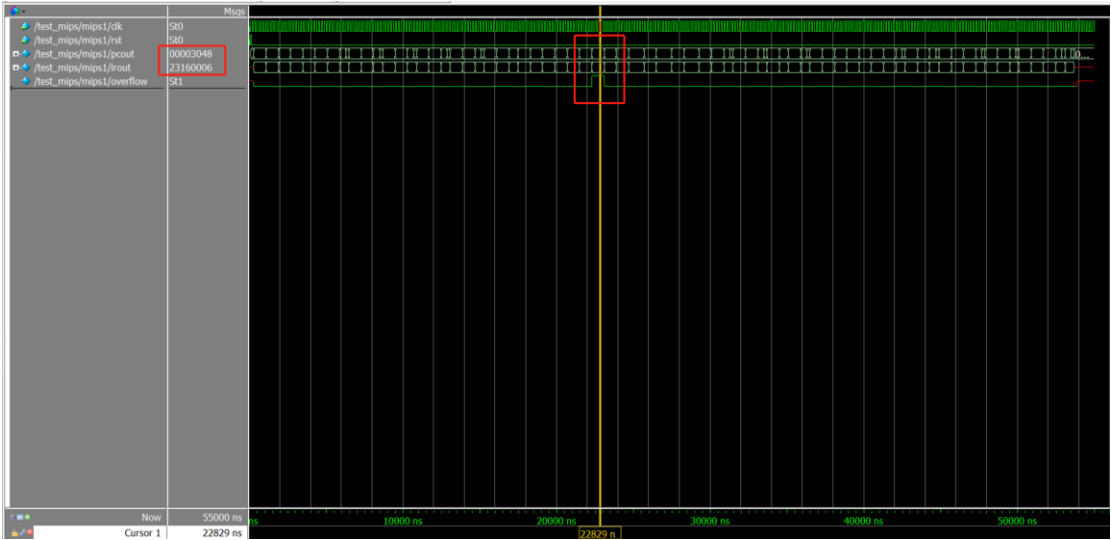
5.1.1 MARS 中结果

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0xabababab
\$v0	2	0xabababab
\$v1	3	0xabababab
\$a0	4	0x2bababab
\$a1	5	0x80000002
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x2bababab
\$t1	9	0x2bababab
\$t2	10	0x00690000
\$t3	11	0x00000000
\$t4	12	0x0000abab
\$t5	13	0x000a0000
\$t6	14	0x0000007f
\$t7	15	0xab000000
\$s0	16	0x00000003
\$s1	17	0x00000001
\$s2	18	0xffffffff
\$s3	19	0xefef0000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0xabababab
\$t8	24	0x7fffffff
\$t9	25	0x00000001
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x00001800
\$sp	29	0x00002ffc
\$fp	30	0x00000000
\$ra	31	0x000030b4
pc		0x000030de
hi		0x00000000
lo		0x00000000

Address	Value (+0)	Value (+4)	Value (+8)
0x00000000	0x80000202	0x7fffffff	0xab000000

5.1.2 Modelsim 中结果

(1) 溢出波形



(2) 寄存器 & 数据存储器内容

a. 寄存器

Memory Data - /test_mips/mips1/gpr_vsz/regFile											
0000001f	000030b4	00000000	00000000	00000000	00000000	00000000	00000001	7fffffff	ababdd2	80000005	00000000
00000014	00000000	efef0000	ffffff80	00000001	00000003	ab000000	0000007f	000a0000	0000abab	00000000	00690000
00000009	2babdd1	2babdd1	00000000	00000000	80000002	2babdd1	ababddcd	ababdd3	ababddcd	00000000	

b. 数据存储器

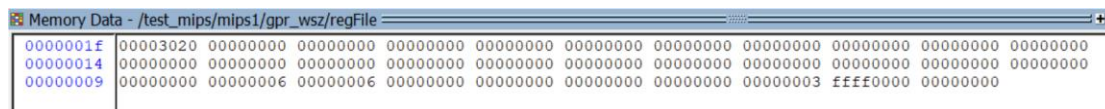
[illegible]

5.2 BLTZAL 指令测试结果

5.2.1 MARS 中结果

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0xfffff000
\$v0	2	0x00000003
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000006
\$t0	8	0x00000006
\$t1	9	0x00000000
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000000
\$s1	17	0x00000000
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$s8	24	0x00000000
\$s9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x00001800
\$sp	29	0x00002ffc
\$fp	30	0x00000000
\$ra	31	0x00003008
pc		0x00003030
hi		0x00000000
lo		0x00000000

5.2.2 Modelsim 中结果



The screenshot shows the 'Memory Data' window in Modelsim, displaying the MIPS register file. The title bar reads 'Memory Data - /test_mips/mips1/gpr_wsz/regFile'. The data is organized into three columns: the first column contains hexadecimal addresses (0000001f, 00000014, 00000009), and the subsequent two columns contain the corresponding 32-bit hexadecimal values for each register. The values for registers 0 through 8 are all 00000000, while register 9 contains the value ffffffff.

Address	Value 1	Value 2
0000001f	00003020	00000000
00000014	00000000	00000000
00000009	00000000	fffffff000

六、总结与心得体会

本次课设完成的是多周期处理器的开发，是在单周期处理器上的优化与改进，也是 MIPS 微系统的 cpu 内部部分，意义重大。

多周期处理器，相较于单周期来说，主要区别在于，将一条指令分为多个阶段来进行处理。原因是，每一条指令的指令周期不同，有的指令需要的时间短，有的指令需要的时间长，通过分为多个阶段进行处理，cpu 的主频将不再受限于最慢的那条指令的指令周期时间，可以大幅提高主频，加快 cpu 的处理速度。

多周期处理器的不同阶段，是通过构建状态机控制的。在不同的状态产生不同的控制信号，以达到合理控制的效果。

通过完成这次课设，我对于将指令分成多个阶段进行处理这一思想有了更为深刻的认识，也正是因为这一次课设，我把每一条指令需要的指令阶段以及控制信号的何时产生，都理的更加顺畅了，在自己脑子里有了更加清晰的构建。课本上的知识得到了运用，做到了理论与实践相结合。

Project2 VerilogHDL 完成 MIPS 微系统开发(支持设备与中断)

一、 总体数据通路结构设计图

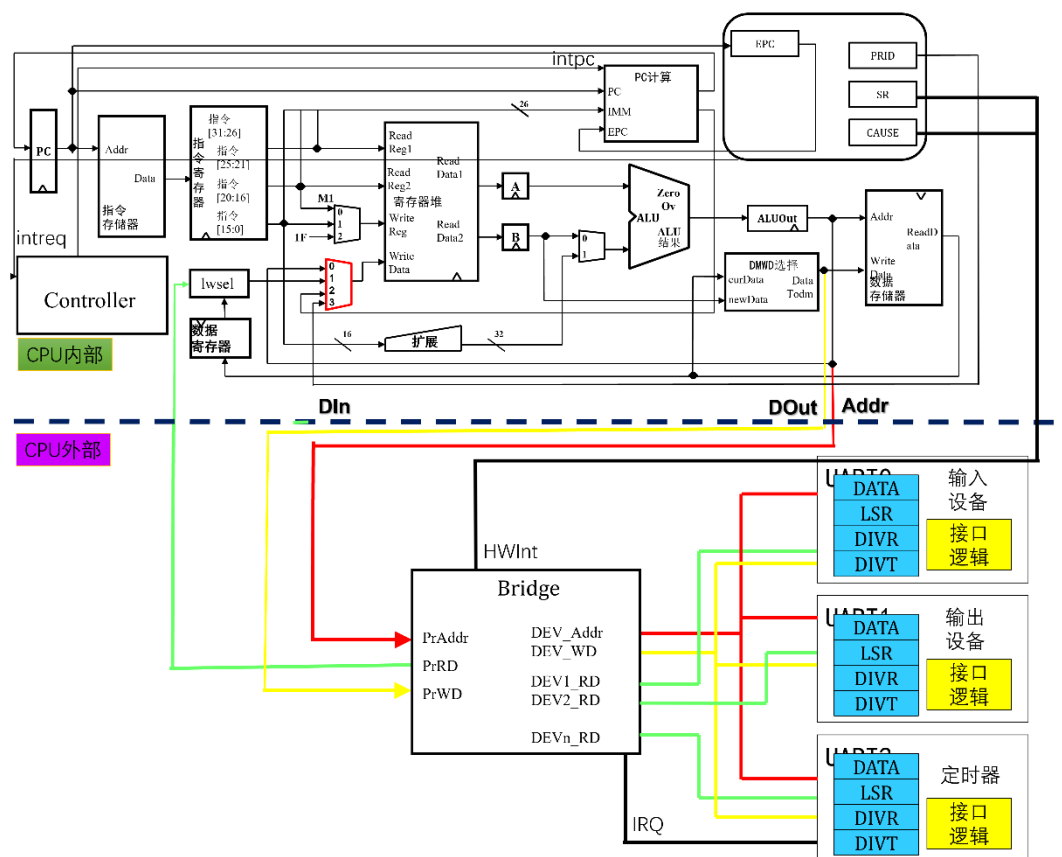
MIPS 微系统的数据通路主要分为两个部分，一个是多周期处理器的数据通路，另一个是 CPU 外部的 bridge 以及和外设的连接。

多周期处理器的数据通路，在 P1 的基础上增加了 CP0 协处理器及其相应数据通路，同时增加了一个 lwsel 模块，用于选择在 MemToWrite 信号为 01 时，写入 dm 的数据到寄存器，还是写入从外设读的数据到寄存器。

CPU 外部设备主要有三个：输入设备、输出设备以及定时器。

系统桥将外设与 CPU 内部连接起来，起到联系的作用。

通路中画出了数据通路以及中断信号的传输路径。



二、 模块定义

2.1 ALU 模块

2.1.1 基本描述

ALU 模块的功能有两个。一个是选择第二个操作数的来源，另一个是完成数据的运算并输出计算结果以及一些其他模块需要的标志位。模块设计与 p1 相同。

2.1.2 模块接口

信号名	方向	描述
[31:0]dataOut_1	I	GPR 输出数据 1 的值，来自 aReg 模块
[31:0]dataOut_2	I	GPR 输出数据 2 的值，来自 bReg 模块
[31:0]ext32	I	16 位立即数扩展结果
[1:0]ALUOp	I	运算方式选择信号 00: + 01: - 10: 11: slt
ALUSrc	I	参与运算的第二个操作数的选择信号 0: dataOut_2 1: ext32
write_30	I	是否为 addi 标志 0: 不是 1: 是
zero	O	运算结果是否为 0 标志 0: 不是 1: 是
overflow	O	addi 运算是否产生溢出标志 0: 不是 1: 是
[31:0]alu_res	O	运算结果
nCondition	O	第一个操作数是否小于 0 标志 (bltzal 指令) 0: 否 1: 是

2.1.3 功能定义

序号	功能名称	功能描述
1	数据选择	选择正确的数据作为第二个操作数的值
2	运算并输出	根据指定运算类型完成运算，并输出结果和一些标志位

2.2 NPC 模块

2.2.1 基本描述

NPC 模块主要完成的功能是下一条 pc 地址的生成，并根据选择信号选择正确

的地址输出给 pc 模块。

与 p1 的区别在于，增加了 ERET, EPC, intreq 等信号。ERET 和 intreq 是中断返回信号和中断跳转信号，EPC 是下一条指令地址输入。

2.2.2 模块接口

信号名	方向	描述
rst	I	复位信号 0: 不复位 1: 复位
zero	I	两操作数相等信号，用于选择下一条地址
nCondition	I	指定寄存器操作数小于 0 信号，用于选择下一条寄存器
[1:0]npc_sel	I	选择输出的 pc 00: pc_ori+4 01: 按照 beq 跳转方式计算地址 10: 按照 j 跳转方式计算地址 11: 选择 jr 地址
[31:0]pcin	I	gpr 中 regFile[rs]值
[31:0]pc_ori	I	pc
[31:0]imm32	I	当前指令的 32 位值，只用到低 26 位
ERET	I	ERET 中断返回指令标志
[31:0]EPC	I	下一条指令地址
intpc	I	中断信号
[31:0]pcout	O	npc 输出

2.2.3 功能定义

序号	功能名称	功能描述
1	复位	当复位信号有效时， $pc_new \leftarrow 32'h0000_3000$
2	计算指令地址，更新 pcnew	如果 intpc 信号有效，则 $pc_new \leftarrow 32'h0000_4180$ 否则根据 npc_sel 信号选择： 00: $pcnew \leftarrow pc_ori + 4$ 01: 如果 zero 或 nCondition 有效，则 pcnew 按照 beq 跳转方式更新 10: 按照 j 跳转方式更新 11: 选择 jr 跳转地址并更新
3	选择输出数据	根据 ERET 指令标志选择 pcout 为 pcnew 还是 epc 的值

2.3 PC 模块

2.3.1 基本描述

PC 模块的主要功能是，将 NPC 模块的输出（下一条指令地址），在时钟上升沿且写使能有效时，将输出的值更改为输入的值。模块设计与 p1 相同。

2.3.2 模块接口

信号名	方向	描述
clk	I	时钟信号
pcwr	I	写使能信号 0：不写入 1：写入
[31:0]pcin	I	输入 npc 模块的输出，下一条指令的地址
[31:0]pcout	O	下一条指令的地址，PC 模块寄存器存储的值

2.3.3 功能定义

序号	功能名称	功能描述
1	存下一条指令地址	在时钟沿上升沿且写使能有效时，将当前输出寄存器的值更改为当前输入值
2	输出下一条指令地址	输出当前寄存器的值

2.4 IM 模块

2.4.1 基本描述

指令存储器，主要功能有两个。一个存储指令，另一个是根据输入地址的值，输出对应地址的指令。

注意 p2 中 IM 寄存器大小变为了 8kb，则输入地址需要 13 位。

2.4.2 模块接口

信号名	方向	描述
[12:0]addr	I	输入地址（指令地址）
[31:0]dout	O	地址对应的指令

2.4.3 功能定义

序号	功能名称	功能描述
1	存储指令	读取 txt 文件，存储指令
2	根据地址输出指令	根据模块的地址输入，输出对应的指令

2.5 IR 模块

2.5.1 基本描述

IR 模块的主要功能是切分数据通路。这个模块本质上是一个寄存器，用于存储 IM 输出的指令。IR 模块的写入代表取指阶段的完成。IR 的写入在时钟上升沿且写使能有效时完成。

模块设计与 p1 相同。

2.5.2 模块接口

信号名	方向	描述
clk	I	时钟信号
irwr	I	写使能信号 0：不写入 1：写入
[31:0]imin	I	输入为 IR 模块的输出，是一条指令
[31:0]irout	O	输出寄存器中的指令

2.5.3 功能定义

序号	功能名称	功能描述
1	切分数据通路	在时钟沿上升沿且写使能有效时，将当前输出寄存器的值更改为当前输入值

2.6 GPR 模块

2.6.1 基本描述

GPR 模块的主要功能分三点。第一是数据选择器功能，选择正确的写入数据和正确的写入地址。第二个是数据的写入功能。第三个是数据的输出功能。

这里与 p1 的不同在于，写入数据的数据选择端加了一路输入，为 cp0in，即

将 cp0 内指定寄存器写入到 gpr 的寄存器中。

2.6.2 模块接口

信号名	方向	描述
clk	I	时钟信号
rst	I	复位信号 0: 不复位 1: 复位
gprwr	I	gpr 写使能信号 0: 不允许写入 1: 允许写入
[1:0]MemToReg	I	选择信号, 选择写入数据 00: alu 运算结果 01: lw/lb 数据 10: 下一条指令地址 11: cp0 中的指定寄存器信息
[1:0]RegDst	I	选择信号, 选择写入地址 00:rt 01:rd 10:no.31
[4:0]rs	I	读出地址 1
[4:0]rt	I	读出地址 2
[4:0]rd	I	写入地址的一种选择
write_30	I	addi 指令标志 0: 不是 1: 是
[31:0]pc_p4	I	写入数据的一种选择, 下一条指令地址
[31:0]aluReg_out	I	写入数据的一种选择, alu 的运算结果
[31:0]dmReg_out	I	写入数据的一种选择, lw/lb 数据
overflow	I	溢出写入数据 0: 不溢出写入 0 1: 溢出写入 1
cp0in	I	读取的 cp0 中寄存器的信息
[31:0]dataOut_1	O	rs 寄存器的值, 数据输出
[31:0]dataOut_2	O	rt 寄存器的值, 数据输出

2.6.3 功能定义

序号	功能名称	功能描述
1	数据选择	根据选择信号选择正确的写入数据和写入地址
2	寄存器存储数据	将数据写入对应寄存器, 存储
3	输出数据	根据读出地址, 读出对应的数据

2.7 EXT 模块

2.7.1 基本描述

EXT 模块的主要功能是, 将 16 位立即数进行扩展。扩展的方式有三种, 第一

种是 0 扩展，第二种是符号扩展，第三种是将 16 位立即数加载到 32 位的高 16 位，低位补 0。

模块设计与 p1 相同。

2.7.2 模块接口

信号名	方向	描述
[15:0]imm16	I	16 位立即数输入
[1:0]ExtOp	I	扩展方式选择信号 00: 0 扩展 01: 符号扩展 10: lui 指令方式扩展
[31:0]ext32	O	扩展结果输出

2.7.3 功能定义

序号	功能名称	功能描述
1	扩展立即数	根据扩展方式的选择信号，扩展 16 位立即数至 32 位。

2.8 aReg & bReg 模块

2.8.1 基本描述

aReg & bReg 模块实现的是同样的功能，即切分数据通路。他们存储的是 GPR 模块的两个输出数据。当两个输出数据写入两个模块的寄存器时，代表译码阶段结束。

模块设计与 p1 相同。

2.8.2 模块接口(以 aReg 模块为例)

信号名	方向	描述
clk	I	时钟信号
[31:0]alu_dataOut_1	I	GPR 模块读出数据 1
[31:0]aReg_out	O	输出寄存器值

2.8.3 功能定义

序号	功能名称	功能描述
1	切分数据通	在时钟沿上升沿时，将当前输出寄存器的值更改为

	路	当前输入值。
--	---	--------

2.9 aluReg 模块

2.9.1 基本描述

aluReg 模块主要功能是切分数据通路。当时钟信号上升沿时，将输入数据写入模块中的寄存器。

模块设计与 p1 相同。

2.9.2 模块接口

信号名	方向	描述
clk	I	时钟信号
[31:0]alu_res	I	ALU 模块运算结果
overflow	I	ALU 模块溢出标志
[31:0]aluReg_out	O	当前模块输出的 ALU 模块运算结果
overflowReg	O	当前模块输出的 ALU 模块溢出标志

2.9.3 功能定义

序号	功能名称	功能描述
1	切分数据通路	在时钟沿上升沿时，将当前输出寄存器的值更改为当前输入值。

2.10 DM 模块

2.10.1 基本描述

DM 模块的功能有两个。一是在写使能有效且时钟上升沿时，将输入数据写入到输入地址的存储空间中。二是在读出输入地址在存储空间中的数据。

与 p1 的区别在于，存储容量变成了 12KB，故写入地址 addr 需要 14 位。

2.10.2 模块接口

信号名	方向	描述
[13:0]addr	I	写入地址

[31:0]din	I	写入数据
we	I	写使能 0: 不写入 1: 允许写入
clk	I	时钟信号
[31:0]dout	O	读出数据

2.10.3 功能定义

序号	功能名称	功能描述
1	写入数据	在时钟沿上升沿且写使能有效时，写入数据
2	读出数据	读出指定地址的数据

2.11 DR 模块

2.11.1 基本描述

DR 模块的主要功能是切分数据通路，还有一个功能是选择输出数据（针对 1b 指令）。当时钟上升沿时，将输入数据写入到模块寄存器。

模块设计与 p1 相同。

2.11.2 模块接口

信号名	方向	描述
clk	I	时钟信号
islb	I	1b 指令标志 0: 不是 1: 是
[31:0]dmout	I	DM 模块读出的数据
[31:0]drout	O	当前模块的输出值

2.11.3 功能定义

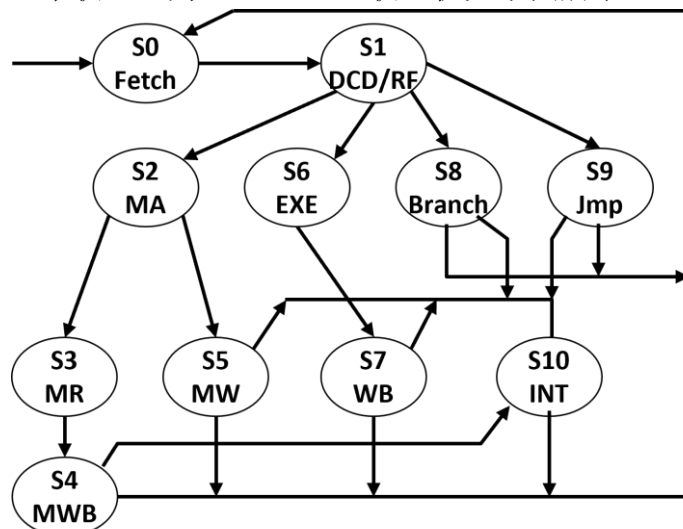
序号	功能名称	功能描述
1	切分数据通路	在时钟沿上升沿时，将当前输出寄存器的值更改为当前输入值。
2	选择输出数据	根据 islb 信号，选择或生成正确的数据并输出。

2.12 Controller 模块

2.12.1 基本描述

Controller 模块是根据输入的 opcode、funct、rt 值（用于 MTC0&MFC0 指令）、zero 信号、intreq 信号，生成对应的指令。同时构建状态机，在对应的状态时，产生对应的控制信号。

状态机共有 11 个状态，为 S0 至 S10，状态机如下图所示：



S0 和 S1 是所有指令都要经历的取指和译码阶段。lw/sw/lb/sb 走 S2，beq/jr/bltzal 走 S8，j/jal 走 S9，其余指令走 S6。

lw/lb 指令由于访存后有回写阶段，故走 S3 读 dm，然后 S4 回写。sw/sb 指令则走 S5 访存（写入）阶段。

S6 阶段是执行阶段，进行 alu 的运算，为 R 型和 I 型指令。然后走到 S7，进行回写阶段。

j/jal 指令是直接跳转，故分到同一个状态 S9。需要注意的是，S9 阶段，如果是 jal 指令，是需要执行回写的，此时的形成跳转地址与回写是在同一个状态下完成的。

上述阶段均与 p1 相同，下面阐述 p2 新增的状态 S10 以及新增的指令 MTC0，MFC0 和 ERET。

首先阐述状态机。这里新增加了一个 S10 状态，他是附于每一条指令的最后一个阶段之后的。如果在指令的最后一个阶段，并且 intreq 信号有效的话，就会进入 S10 状态，中断状态；否则就忽略掉 S10 状态，直接跳回 S0 即可。

对于 MTC0 和 MFC0 两条指令，他们的指令功能同 sw/sb/lw/lb 是相类似的，都是读/写功能，故将他们两个与上述四个指令分在同一个周期 S2。到了 S2 阶段后，MFC0 是读 cp0 寄存器并写入 gpr 寄存器中，与 lw/lb 指令类似，有回写阶段，故分到 S3 读阶段（MFC0 是读 cp0 的寄存器）然后 S4 回写阶段。MTC0 是写 cp0 寄存器，这与 sw/sb 指令功能类似，故分到 S5 阶段进行写入（这里写的是 cp0 的寄存器）。

对于 ERET 指令，其功能与 j/jal 类似，进行无条件跳转（形成地址），故分到 S9 阶段，在 S9 阶段完成跳转地址的更新。

2.12.2 模块接口

信号名	方向	描述
clk	I	时钟信号
rst	I	复位信号
[5:0]opcode	I	当前指令 opcode
[5:0]funct	I	当前指令 funct
zero	I	ALU 模块产生的 zero 标志
nCondition	I	ALU 模块产生的 nCondition 标志 (bltzal 指令)
[4:0]M	I	指令的 rt 部分的内容，用于译码 MTC0 和 MFC0 指令
intreq	I	中断请求信号
[1:0]RegDst	O	GPR 模块写入地址选择 00:rt 01:rd 10:no.31
[1:0]MemToReg	O	GPR 模块写入数据选择 00: alu 运算结果 01: lw/lb 数据 10: 下一条指令地址
[1:0]npc_sel	O	NPC 模块输出选择 00: pc_ori+4 01: 按照 beq 跳转方式计算地址 10: 按照 j 跳转方式计算地址 11:选择 jr 地址
[1:0]ALUOp	O	ALU 模块运算功能选择 00: + 01: - 10: 11: slt
[1:0]EXTOp	O	EXT 模块扩展类型选择 00: 0 扩展 01: 符号扩展 10: lui 指令方式扩展
RegWrite	O	GPR 模块写使能
ALUSrc	O	ALU 模块操作数 2 的选择 0: GPR 模块读出数据 2 1: 16 位立即数扩展结果
MemWrite	O	DM 模块写使能
write_30	O	addi 指令标志
pcwr	O	PC 模块写使能
irwr	O	IR 模块写使能
islb	O	LB 指令标志
issb	O	SB 指令标志
cp0_wen	O	cp0 写使能信号
bridge_wen	O	系统桥写使能信号，用于对外设的写使能，与选择信号配合使用
exlset	O	标志信号：存 epc，exl 中断防再入，
exlclr	O	标志信号：pc←epc，exl 恢复可中断
intpc	O	标志信号：pc 跳转中断地址

2.12.3 功能定义

序号	功能名称	功能描述
1	构建状态机， 确定当前状态	按设计好的状态机构建状态，并根据不同的指令确定下一个状态。
2	产生控制信号	在不同的状态下，产生对应的控制信号和标志。

2.13 Muxdmin 模块

2.13.1 基本描述

Muxdmin 模块设计的初衷是为了支持 sb 指令。主要功能是根据 issb 信号，选择或生成正确的写入数据，传送给 DM 模块的写入数据端口。

这里的逻辑是，由于 DM 模块在时钟上升沿来临时才进行写入，所以可以在来临时将需要写入的地址的值取出。如果是 sb 指令，则将需要写入的低 8 位拼上原来的 24 位输出即可。如果不是，则直接将取出的值输出。这一部分是组合逻辑电路，在时钟沿来临时，DM 模块将数据写入即可。

模块设计与 p1 相同。

2.13.2 模块接口

信号名	方向	描述
[31:0]bReg_out	I	bReg 寄存器输出
[31:0]dmout_data	I	DM 模块读出数据
issb	I	sb 指令标志
[31:0]dm_datain	O	当前模块数据输出，输出给 DM 模块输入

2.13.3 功能定义

序号	功能名称	功能描述
1	选择或生成 DM 模块写入数据	根据 issb 控制信号，生成或选择 DM 模块的写入数据。

2.14 muxlw 模块

2.14.1 基本描述

muxlw 模块主要功能是在 npc_sel 为 01 时，选择写入选择端的数据是 dm 的输出数据，还是从外设读的数据。这么做的原因是，gpr 的写入数据的数据选择器只有两位位宽，而 2' b11 要给 cp0 读出信息使用，故在数据选择器前另加模块，选择 npc_sel 为 2' b01 时的写入数据。

2.14.2 模块接口

信号名	方向	描述
[31:0]addr	I	bReg 寄存器输出
[31:0]drData	I	DM 模块读出数据
[31:0]bridgeData	I	sb 指令标志
[31:0]muxlwOut	O	当前模块数据输出，输出给 DM 模块输入

2.14.3 功能定义

序号	功能名称	功能描述
1	选择数据	根据地址，判断并选择数据

2.15 bridge 模块

2.15.1 基本描述

Bridge 模块的基本功能是连接 cpu 与外设的通讯桥梁，传递彼此之间的地址、数据以及自身生成的使能信号。三大功能为，地址转换、读数据、写数据。同时还是定时器产生的 IRQ 信号的传递桥梁。

外设 1 输入设备，不需要写使能，只读。外设 0 是 timer，外设 2 是输出设备。

2.15.2 模块接口

信号名	方向	描述
[31:0]praddr	I	外设地址
[31:0]prwd	I	外设写数据
[31:0]dev0_rd	I	外设 0 读数据

[31:0]dev1_rd	I	外设 1 读数据
[31:0]dev2_rd	I	外设 2 读数据
wecpu	I	外设写使能，需要搭配选择信号使用
IRQ	I	计时器产生的中断请求信号
[31:0]prrd	O	外设读的数据（选择后的）
[31:0]dev_wd	O	外设写数据
[31:0]dev_addr	O	外设地址
wedev0	O	外设 0 写使能
wedev2	O	外设 2 写使能
[5:0]HWInt	O	6 路中断信号

2.15.3 功能定义

序号	功能名称	功能描述
1	传递读出的外设数据	将从外设读的数据传给 gpr 进行写入
2	传递要写给外设的数据	将 dm 读出数据传给外设进行写入
3	外设选择，使能信号生成	根据地址和译码信号，产生使能信号
4	汇总中断信号并传递	计时器 IRQ 传给桥，桥汇总到 6 位中断信号，传给 cpu 内部

2.16 cp0 模块

2.16.1 基本描述

cp0 模块是协处理器，主要用来处理异常。本次 cp0 模块中包含 4 个寄存器：sr、cause、epc 和 prid。sr 寄存器的功能是存储屏蔽位 im，全局中断使能信号 ie 以及防止再中断的标志位 exl；cause 寄存器的内容是不断的锁存中断产生的原因；epc 寄存器就是存储下一条指令地址的；prid 寄存器是存储个性化签名等等。需要注意的是，cause 寄存器只读。

2.16.2 模块接口

信号名	方向	描述
clk	I	时钟信号
rst	I	复位信号
wen	I	寄存器写使能信号

exlset	I	中断开始信号，完成 epc 寄存器写入以及 exl 置位
exlclr	I	ERET 信号，开 exl 中断
[4:0]sel	I	读/写地址
[5:0]HWInt	I	6 路中断信号
[31:0]din	I	写入数据
[31:0]pc_p4	I	下一条指令地址，用于写入 npc 寄存器
[31:0]pcout	0	epc 的输出
[31:0]dout	0	读出数据的输出
intreq	0	根据屏蔽位 im 以及标志位 ie、exl 产生的中断信号

2.16.3 功能定义

序号	功能名称	功能描述
1	写数据	根据数据输入和写入地址，写入数据（使能有效时）
2	读数据	根据地址输入，读数据并输出
3	产生中断信号	根据 6 路 hwint 中断信号产生 1 位中断信号 intreq
4	记录异常原因	cause 寄存器不断锁存 HWInt 信号

2.17 timer 模块

2.17.1 基本描述

Timer 模块主要由三个寄存器组成：控制寄存器、初值寄存器以及计数器。主要的功能是，当计数器为 0 时，产生中断请求信号 IRQ。

接下来阐述每个寄存器的功能。控制寄存器主要负责的是管理是否允许中断、计数模式选择以及是否允许计数。初值寄存器用于存放初值，保持不变。计数器用于在时钟沿来临时进行自减，完成计数。

当计数器值为 0 时，根据不同的模式执行不同的操作。当为模式 0 时，则一直保持 0 不变，直到初值寄存器再次被外部写入后，初值寄存器值再次被加载至计数器，计数器重新启动倒计时；当为模式 1 时，则自动加载初值寄存器的值到计数器，开始新一轮的倒计时。

2.17.2 模块接口

信号名	方向	描述
CLK_I	I	时钟信号
RST_I	I	复位信号
[3:2]ADDR_I	I	读/写地址，三个寄存器，两位地址信号即可。且低两位永远为 0，故不用考虑。

WE_I	I	写使能信号
[31:0]DAT_I	I	写入数据
[31:0]DAT_O	O	读出数据
IRQ	O	中断请求信号，count 为 0 时产生

2.17.3 功能定义

序号	功能名称	功能描述
1	计数	根据控制寄存器内容，在不同的模式下完成计数（允许计数时）。
2	产生中断请求信号	当计数器值为 0 时，产生中断请求信号 IRQ。

2.18 outputDev 模块

2.18.1 基本描述

outputDev 为外设输出模块，包含两个寄存器：初值寄存器以及当前值寄存器。模块具有输入输出功能：输入时，在时钟上升沿且写使能有效时，将数据写入地址指示的寄存器中；输出时，根据地址，选择正确的寄存器的值进行输出。

两个寄存器的目的是，便于在后续秒计数时，判断当前输入与初值寄存器的值是否相同。如果相同，即输入没变化，则自加 1。否则将两个寄存器更新为新的输入值，然后在下一次中断时再进行自加 1。

2.18.2 模块接口

信号名	方向	描述
clk	I	时钟信号
en	I	写使能信号
[3:2]addr	I	读/写地址，也可以理解为寄存器选择信号。
[31:0]din	I	写入数据
[31:0]dout	O	读出数据

2.18.3 功能定义

序号	功能名称	功能描述
1	读数据	根据 addr 读出指定寄存器数据。
2	写数据	根据 addr，在写使能有效且时钟上升沿时，写入数

		据到寄存器中。
--	--	---------

2.19 输入模块

本次 MIPS 微系统的输入模块，是在顶层文件上直接定义了一个 32 位输入作为输入模块。因为输入模块只涉及数据的输入，不涉及其他功能，故只定义一个输入端口就够用了。

三、 指令描述

序号	助记符	opcode	funct	指令功能
1	addu	000000	100001	无符号加法
2	subu	000000	100011	无符号减法
3	ori	001101		或立即数
4	lw	100011		加载字
5	sw	101011		存储字
6	beq	000100		等于时转移
7	lui	001111		立即数加载至高位
8	j	000010		无条件跳转
9	addi	001000		加立即数（有符号）
10	addiu	001001		加立即数（无符号）
11	slt	000000	101010	小于时置1
12	jal	000011		跳转并链接
13	jr	000000	001000	跳转寄存器
14	lb	100000		加载字节
15	sb	101000		存储字节
16	ERET	010000	011000	中断返回
17	MFC0	010000	00000 (MF) rs位置	读cp0中寄存器到gpr[rt]
18	MTC0	010000	00100 (MT) rs位置	写cp0中的寄存器

四、 测试程序

4.1 MIPS-Lite3 指令集的测试主程序

```

1  ori $1,$0,0x7f00      # ctrl寄存器地址
2  ori $2,$0,0x7f04      # preset寄存器地址
3  ori $3,$0,0x7f08      # count寄存器地址
4  ori $4,$0,0x7f0c      # IPD输入设备地址
5  ori $5,$0,0x7f10      # OPD preData输出设备初值寄存器地址
6  ori $6,$0,0x7f14      # OPD curData输出设备当前值寄存器地址
7  lw $7,0($4)           # 从输入设备获取当前的输入值
8  sw $7,0($5)           # 把当前输入值保存到输出设备初值寄存器中
9  sw $7,0($6)           # 把当前输入值保存到输出设备当前值寄存器中
10 ori $12,$0,0x0401      # setSR 100_0000_0001设置sr寄存器的值（准备好值）
11 mtc0 $12,$12          # 把gpr12号寄存器内容写入到cp0的sr寄存器中（完成设置）
12 mfc0 $21,$15          # 获取cp0中prid寄存器的值，写入gpr21号寄存器
13 ori $13,$0,0x000a      # 10 倒计时（准备好初值）
14 ori $9,$0,0x0009      # ctrl控制位 1001 （准备好值）
15 sw $13,0($2)          # 把倒计时初值10存入preset寄存器（完成设置）
16 sw $13,0($3)          # 把倒计时初值10存入count寄存器（完成设置）
17 sw $9,0($1)           # 设置计时器的ctrl寄存器的标志位（完成设置）
18 loop:j loop          # 死循环

```

4.2 MIPS-Lite3 指令集的测试中断子程序

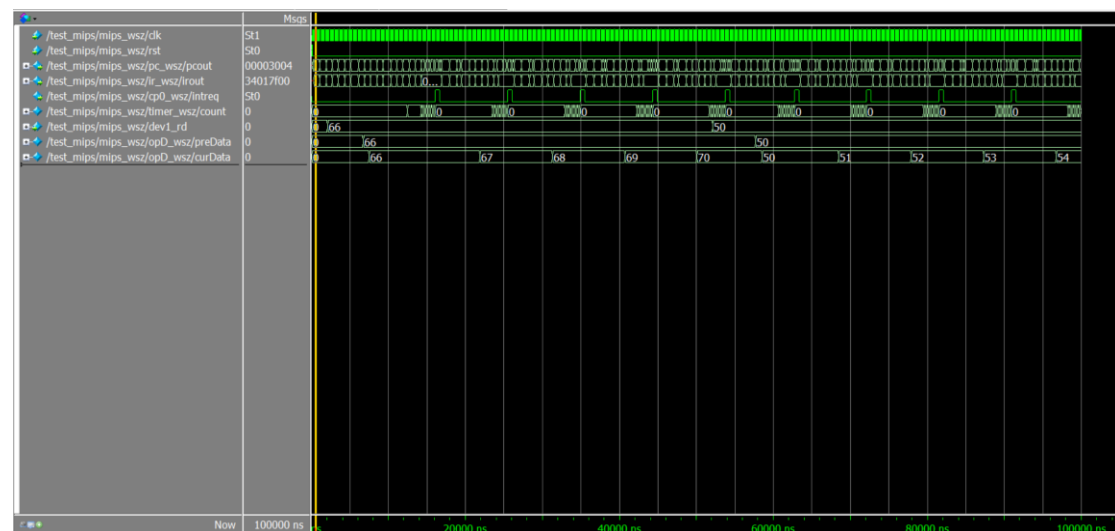
```

1 lw $15, 0($4)    # IPD 获取输入设备的值
2 lw $16, 0($5)    # OPD preData 获取输出设备的初值寄存器的值
3 beq $16, $15, eq  # 比较二者的值，相等跳eq处
4 sw $15, 0($5)    # 如果不相等，设置输出设备初值寄存器为当前输入的值
5 sw $15, 0($6)    # 如果不相等，设置输出设备当前值寄存器为当前输入的值
6 beq $0, $0, jmp  # 由于不清楚怎么在mars中设置起始地址为'h4180，故用beq指令代替j指令，类似j指令的无条件跳转
7 eq:lw $16, 0($6) # 如果二者相等，则获得输出设备的当前值寄存器的值
8 addiu $16, $16, 1 # 无符号加立即数，自加1
9 sw $16, 0($6)    # 将自加后的结果存回输出设备的当前值寄存器
10 jmp:ori $13, $0, 0x000a # 10 倒计时 重新设置倒计时初值（准备值）
11 sw $13, 0($2)   # 完成设置
12 eret           # 中断返回

```

五、 测试结果

5.1 波形



5.2 寄存器组结果

5.2.1 gpr 寄存器

mfc0 指令，加载 cp0 的 prid 寄存器的值到 gpr 寄存器中。

Memory Data - /test_mips/mips_wsz/gpr_wsz/regFile - Default									
0000001f	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
00000017	00000000	00000000	20020203	00000000	00000000	00000000	00000000	00000000	00000036
0000000f	00000032	00000000	0000000a	00000401	00000000	00000000	00000009	00000000	00000000
00000007	00000042	00007f14	00007f10	00007f0c	00007f08	00007f04	00007f00	00000000	00000000

5.2.2 cp0 寄存器

sr 寄存器由 mtc0 指令设置；epc 寄存器存返回地址；prid 的签名是在初始化时设置的。

0000001f	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
00000017	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
0000000f	20020203	00003044	00000000	00000401	00000000	00000000	00000000	00000000	00000000
00000007	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000

Cause 寄存器不断锁存 hwint 的值，下图是波形最后时刻为产生中断请求时，锁存的结果。

Memory Data - /test_mips/mips_wsz/cp0_wsz/regFile_cp0 - Default									
0000001f	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
00000017	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
0000000f	20020203	00003044	00000400	00000401	00000000	00000000	00000000	00000000	00000000
00000007	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000

六、 总结与心得

通过完成 MIPS 微系统的开发，我不仅仅对 cpu 内部的数据通路、信号控制有了更上一层次的了解，更关键的是，我对于 cpu 内部是如何与 cpu 外部进行通讯的有了很全面的了解。

在开发过程中，印象比较深刻的就是中断请求信号的传输过程：从外设定定时器发出，经历系统桥，到协处理器，再到控制器，最后再到 npc 模块。其中比较关键的是 intreq 信号与状态机的配合使用，以及在多个中断需要处理时，使用屏蔽位来管理他们的优先级这一思想，都使我受益匪浅。

这次的 MIPS 微系统开发，主要解决的问题是与外设的通讯、交互功能。解决问题的主要思想是，合理设计控制信号的通路、合理设计状态机、合理设计系统桥，并且需要给外设分配符合存储空间规定的外设地址，以此来达到设计要求。

至此，计算机组成原理的课设结束了，但是我对于硬件课程的学习还没有停止，并且相信在今后的课程、课设中，一定会加倍努力学习，将知识融会贯通。