计算机组成原理

期末大作业报告

学　　号_____20090121_____

姓　　名_____陈立标_____

指导教师_____魏坚华_____

提交日期_____2022 年 6 月 14 日_____

成绩评价表

报告内容	报告结构	报告最终成绩
□丰富正确 □基本正确 □有一些问题 □问题很大	□完全符合要求 □基本符合要求 □有比较多的缺陷 □完全不符合要求	
报告与 Project 功能一致性	报告图表	总体评价
□完全一致 □基本一致 □基本不一致	□符合规范 □基本符合规范 □有一些错误 □完全不正确	

教师签字:_____

目录

1 总体数据通路结构设计图.....1
2 模块定义.....1
2.1 IFU 模块定义.....1
2.1.1 基本描述.....1
2.1.2 模块接口.....1
2.1.3 功能定义.....2
2.2 GetCode 模块定义.....2
2.2.1 基本描述.....2
2.2.2 模块接口.....2
2.2.3 功能定义.....3
2.3 Controller 模块定义.....3
2.3.1 基本描述.....3
2.3.2 模块接口.....3
2.3.3 功能定义.....4
2.4 RegFile 模块定义.....4
2.4.1 基本描述.....4
2.4.2 模块接口.....4
2.4.3 功能定义.....5
2.5 Extender 模块定义.....5
2.5.1 基本描述.....5
2.5.2 模块接口.....5
2.5.3 功能定义.....5
2.6 ALU 模块定义.....5
2.6.1 基本描述.....5
2.6.2 模块接口.....5
2.6.3 功能定义.....6
2.7 DM 模块定义.....6
2.7.1 基本描述.....6
2.7.2 模块接口.....7
2.7.3 功能定义.....7
3 指令测试.....7
3.1 指令说明.....7
3.2 测试代码.....8
3.3 测试结果.....10
4 拓展指令设计.....12
4.1 设计思路.....12
4.2 指令设计.....12
4.3 测试结果.....12
5 总结.....13

1 总体数据通路结构设计图

总体数据通路以及控制器设置如图 1 所示，完成了 MIPS-LITE1 共 13 条指令以及拓展指令 BGTZ。

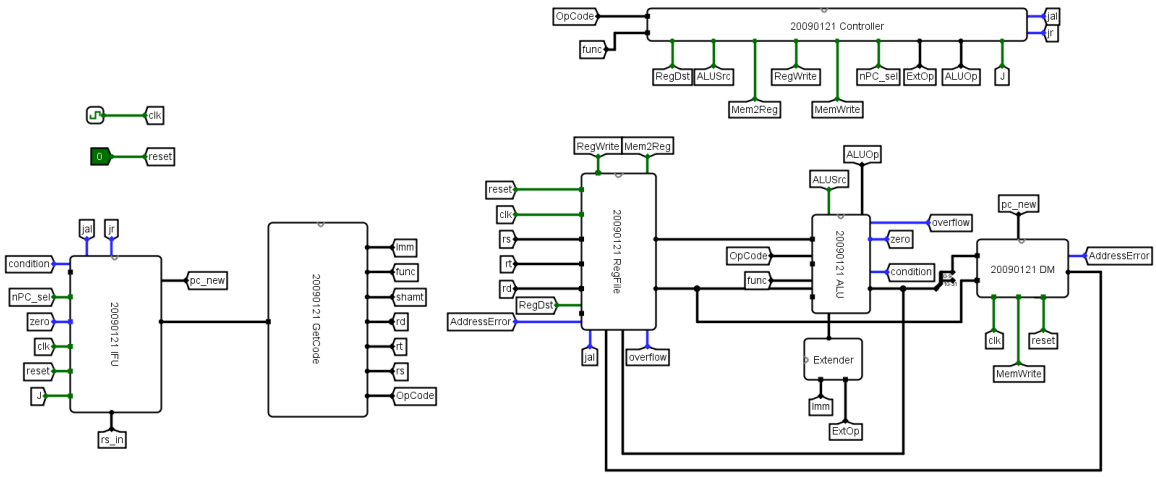


图 1 总体数据通路结构

2 模块定义

2.1 IFU 模块定义

2.1.1 基本描述

IFU 主要完成各种情况下的取指令功能：顺序取指令;BEQ、BGTZ 相对跳转取指令;J、JR、JAL 绝对跳转取指令。IFU 内还包含一个 1KB 指令存储器用来存储指令。

2.1.2 模块接口

信号名	方向	描述
nPC_sel	I	指示当前是否为 BEQ 指令。 0：当前指令非 BEQ 指令 1：当前指令为 BEQ 指令
$zero$	I	指示 ALU 的计算结果是否为 0。 0：计算结果非 0 1：计算结果为 0
$condition$	I	指示 RS 寄存器内容是否大于 0，用于 BGTZ 指令的跳转判断 0：寄存器内容不大于 0 1：寄存器内容大于 0
j	I	指示当前指令是否为 J 指令。 0：当前指令非 J 指令 1：当前指令为 J 指令
jal	I	指示当前是否为 JAL 指令。 0：当前指令非 JAL 指令 1：当前指令是 JAL 指令

jr	I	指示当前是否为 JRL 指令。 0：当前指令非 JR 指令 1：当前指令是 JR 指令
clk	I	时钟信号
$reset$	I	复位信号。 0：无效 1：复位
$rs\_in[31:0]$	I	传输 rs 寄存器内容，用于 jr 跳转
$Imm[15:0]$	I	MIPS 指令中的低 16 位立即数
$insout[31:0]$	O	32 位 MIPS 指令
$pc\_new[31:0]$	O	下一条应执行的指令， 1. 用于判断 lw、sw 地址是否正确。 2.用于实现 JAL 所规定的指令存储

2.1.3 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 被复位
2	取指令	根据 PC 从 IM 中取出指令
3	计算下一条指令地址	1. 若 nPC_sel 为 1，且 zero 为 1 或 condition 为 1，则 PC←PC+4+(sign-extend(Imm[15:0])<<2) 2. 若 J 为 1 或 JAL 为 1 或，则 PC←$PC_{GPRLEN-1..28}$ || instr_index || 0^2 3. 若 JR 为 1，则 PC←$GPR[rs]_{GPRLEN-1..1}$ || 0 4. 若为其他情况，则 PC←PC+4

2.2 GetCode 模块定义

2.2.1 基本描述

GetCode 模块用于将 IFU 取出的 32 位 MIPS 分割为多个信号方便后续器件使用，本身无特殊功能，为过渡器件。

2.2.2 模块接口

信号名	方向	描述
$Input[31:0]$	I	32 位 MIPS 指令
$Imm[15:0]$	O	I 指令中的低 16 位字段，
$func[5:0]$	O	功能码，与操作数码相结合，确定指令功能
$shamt[4:0]$	O	提供移位数目，非移位指令时总是 0
$rd[4:0]$	O	提供接收结果的目的寄存器的编号
$rs[4:0]$	O	提供第一源操作数所使用寄存器的编码
$rt[4:0]$	O	提供第二源操作数所使用寄存器的编码
$OpCode[5:0]$	O	操作码

2.2.3 功能定义

序号	功能名称	功能描述
1	取字段	将 32 位 MIPS 指令分解为 7 个字段

2.3 Controller 模块定义

2.3.1 基本描述

Controller 为系统的控制模块，通过分析 MIPS 指令的操作码与功能码，辨别该指令属于哪一条具体的指令，随后根据指令功能产生控制信号用于控制其他模块。

2.3.2 模块接口

信号名	方向	描述
$Opcode[5:0]$	I	操作码
$Func[5:0]$	I	功能码，与操作数码相结合，确定指令功能
$RegDst$	O	写寄存器的目标寄存器号来源。 0：来源于 rt 1：来源于 rd
$ALUSrc$	O	第二个 ALU 操作数来源。 0：来源于寄存器 1：来源于立即数拓展
$Mem2Reg$	O	写入寄存器的数据来源。 0：来源于 ALU 计算结果 1：来源于数据存储器
$RegWrite$	O	寄存器写使能。 0：不能写入 1：能够写入
$MemWrite$	O	数据存储器写使能。 0：不能写入 1：能够写入
nPC_sel	O	标志是否为 BEQ 指令 0：非 BEQ 指令 1：是 BEQ 指令
$ExtOp[1:0]$	O	控制立即数拓展方式。 00：0 拓展 01：符号拓展 10：低 16 位补零
$ALUOp[1:0]$	O	控制 ALU 运算种类。 00：加法运算 01：减法运算 10：按位或运算
J	O	标志是否为 J 指令。 0：非 J 指令 1：是 J 指令
jal	O	标志是否为 JAL 指令。

		0：非 JAL 指令 1：是 JAL 指令
jr	*0*	标志是否为 JR 指令。 0：非 JR 指令 1：是 JR 指令

2.3.3 功能定义

序号	功能名称	功能描述
1	译码操作	将 MIPS 指令译码为对应操作。
2	产生控制信号	针对不同指令产生相对应的控制信号

2.4 RegFile 模块定义

2.4.1 基本描述

RegFile 模块存放 32 个 32 位具有写使能的寄存器，使用 5 位译码器可以在使能端有效时向指定寄存器写入数据，另外具有两个多路选择器支持同时取出两个寄存器内的内容。当指令为 JAL 时，将当前指令顺序执行的下一条指令 PC 地址写入 32 号寄存器，当指令为 ADDI 时，若运算溢出则 30 号寄存器写入 1.

2.4.2 模块接口

信号名	方向	描述
clk	*I*	时钟信号
reset	*I*	重置所有寄存器的内容
RegWrite	*I*	寄存器写使能。 0：不能写入 1：能够写入
RegDst	*I*	指示写入 rt 寄存器还是 rd 寄存器 0：写入 rt 寄存器 1：写入 rd 寄存器
Mem_to_Reg	*I*	指示写入的数据来自 ALU 还是 DM 0：数据来自 ALU 1：数据来自 DM
overflow	*I*	指示立即数运算是否溢出 0：未溢出 1：溢出
AddressError	*I*	指示 lw、sw 地址是否正确（后两位是否为 00） 0：地址错误 1：地址正确
rd[4:0]	*I*	指示写入的寄存器编号
rs[5:0]	*I*	指示第一个取出数据的寄存器编号
rt[5:0]	*I*	指示第二个取出数据的寄存器编号 或指示写入的寄存器编号
pc_new[31:0]	*I*	若为 JAL 指令，pc_new 为 PC+4 对应地址，用于写入 32 号寄存器

$data\_alu[31:0]$	I	ALU 运算结果
$data\_dm[31:0]$	I	DM 存储器内容
$rs\_out[31:0]$	O	传输从 rs 对应寄存器取出的数据
$rt\_out[31:0]$	O	传输从 rt 对应寄存器取出的数据

2.4.3 功能定义

序号	功能名称	功能描述
1	写入数据	写使能有效时向指定寄存器中写入数据 1. 写入 ALU 运算结果 2. 写入 DM 指定存储器内容 3. 写入 PC+4 地址于 32 号寄存器 4. 写入溢出标志于 30 号寄存器
2	读取数据	同时从最多两个指定寄存器中取出数据

2.5 Extender 模块定义

2.5.1 基本描述

不同的指令要求不同形式的立即数拓展，Extender 设计了三种立即数拓展方式，支持本题要求的所有指令的拓展需求。

2.5.2 模块接口

信号名	方向	描述
$Imm[15:0]$	I	16 位待拓展立即数
$ExtOp[1:0]$	I	控制立即数拓展方式。 00：0 拓展 01：符号拓展 10：低 16 位补零
$Output[31:0]$	O	输出拓展后的数据

2.5.3 功能定义

序号	功能名称	功能描述
1	0 拓展	将 16 位立即数 0 拓展至 32 位
2	符号拓展	将 16 位立即数符号拓展至 32 位
3	低 16 位补零	将 16 位立即数置高 16 位同时低 16 位补 0

2.6 ALU 模块定义

2.6.1 基本描述

ALU 作为运算器，支持无符号数加法、与立即数的有符号支持溢出加法、与立即数的无符号加法、无符号数减法、与立即数按位取或五种运算。ALU 还能够为 BEQ 判断结果是否大于 0、为 BGTZ 判断 RS 寄存器内容是否大于 0、同时输出结果为 0 信号、溢出信号、RS 大于 0 信号.

2.6.2 模块接口

信号名	方向	描述

$rs\_in[31:0]$	I	rs 对应寄存器内容
$rt\_in[31:0]$	I	rt 对应寄存器内容
$imm\_in[31:0]$	I	拓展后的立即数
$ALUop[1:0]$	I	控制 ALU 运算种类。 00：加法运算 01：减法运算 10：按位或运算
$ALUSrc$	I	指示运算第二操作数来源 0：来源于 rt 寄存器内容 1：来源于拓展后立即数内容
$OpCode[5:0]$	I	用于判断具体指令，以完成对应运算
$func[5:0]$	I	用于判断具体指令，以完成对应运算
$result[32:0]$	O	运算结果，采用双符号位以判断溢出
$zero$	O	指示运算结果是否为 0。 0：结果非 0 1：结果为 0
$overflow$	O	指示运算是否溢出 0：不溢出 1：溢出
$condition$	O	指示 rs 寄存器内容是否大于 0 0：不大于 0 1：大于 0

2.6.3 功能定义

序号	功能名称	功能描述
1	无符号加法	完成 ADDU 关于两个寄存器的无符号加法
2	无符号减法	完成 SUBU 关于两个寄存器的无符号减法
3	按位或	完成 ORI 关于寄存器和 0 拓展后立即数按位或
4	有符号加法	完成 ADDI 关于寄存器和符号拓展立即数的加法，支持溢出
5	无符号立即数加法	完成 ADDIU 关于寄存器和符号拓展立即数的无符号加法。
6	指示结果 0	zero 指示结果是否为 0，可以与 nPC_Sel 一同确定是否为 BEQ 指令
7	指示溢出	overflow 指示 ADDI 运算结果溢出
8	指示 rs 寄存器内容大于 0	condition 指示 rs 寄存器内容大于 0，用于 BGTZ 指令的跳转判断

2.7 DM 模块定义

2.7.1 基本描述

此部分存储器容量为 1KB，用 RAM 实现，是与数据存储器的交互部分，能够实现存储器内数据向寄存器的传递，当使能有效时，支持数据向存储器的写入。

2.7.2 模块接口

信号名	方向	描述
$pc\_alu[31:0]$	I	确定写入的存储器地址
$data\_rt[31:0]$	I	rt 寄存器对应内容
$pc\_new[31:0]$	I	用于为 lw、sw 判断操作地址是否合法
$MemWrite$	I	数据存储器写使能。 0：不能写入 1：能够写入
clk	I	时钟信号
$reset$	I	复位信号 0：无效 1：复位
$data\_out[31:0]$	O	输出指定存储器的数据
$AddressError$	O	指示 lw、sw 操作地址错误 0：地址正确 1：地址错误

2.7.3 功能定义

序号	功能名称	功能描述
1	写入	当 MemWrite 为 1 时以小端序顺序向指定存储器地址处写入数据
2	读取	从指定存储器地址处读取数据

3 指令测试

3.1 指令说明

编号	操作码	助记符	指令功能
1	$opcode$：000000 $func$：100001	$addu$	两个寄存器的无符号数加
2	$opcode$：000000 $func$：100011	$subu$	两个寄存器的无符号减
3	$opcode$：00000 $func$：101010	slt	判断 rs 寄存器是否小于 rt 寄存器并将结果写入 rd 寄存器
4	$opcode$：000000 $func$：001000	jr	跳转至 rs 寄存器所存储的地址

5	$opcode$：100011	lw	从存储器中加载字
6	$opcode$：101011	sw	向存储器中存储字
7	$opcode$：000100	beq	若运算结果为 0 则相对跳转
8	$opcode$：001111	lui	低 16 位立即数左移至高位
9	$opcode$：000010	j	绝对跳转
10	$opcode$：001101	ori	寄存器内容与立即数按位或
11	$opcode$：001000	$addi$	寄存器内容与立即数有符号加
12	$opcode$：001001	$addiu$	寄存器内容与立即数无符号加
13	$opcode$：000011	jal	将 pc+4 地址写入 32 号寄存器后绝对跳转
14	$opcode$：000111	$bgtz$	若 rs 寄存器内容大于 0 则相对跳转

3.2 测试代码

编号	代码	注释
1	ori \$16,\$0,1	$GPR[16] = GPR[0] \mid zero\_ext(1)$
2	ori \$17,\$0,3	$GPR[17] = GPR[0] \mid zero\_ext(3)$
3	ori \$8,\$0,1	$GPR[8] = GPR[0] \mid zero\_ext(1)$
4	ori \$12,\$0,$0xabab$	$GPR[12] = GPR[0] \mid zero\_ext(0xabab)$
5	lui \$13,10	$GPR[13] = \{10, \{16\{0\}\}$
6	$start$: $addu$ \$4,\$0,\$16	$GPR[4] = GPR[0] + GPR[16]$
7	$addu$ \$5,\$0,\$8	$GPR[5] = GPR[0] + GPR[8]$
8	$jal\ newadd$	$GRG[31] = PC + 4$ $PC \leftarrow newadd$
9	$addu$ \$16,\$0,\$2	$GPR[16] = GPR[0] + GPR[2]$
10	$subu$ \$17,\$17,\$8	$GPR[17] = GPR[17] + GPR[8]$
11	beq \$16,\$17,$start$	$zero = (GPR[20] - GPR[0])?\ 0:1$ $if(zero = 1)\ pc \leftarrow start$
12	ori \$8,\$0,4	$GPR[8] = GPR[0] \mid zero\_ext(4)$
13	$addiu$ \$24,\$0,$0x7fffffff$	$GPR[24] = GPR[0] + sign\_ext(0x7fffffff)$
14	$addiu$ \$9,\$24,3	$GPR[9] = GPR[34] + sign\_ext(3)$
15	$addiu$ \$10,\$24,5	$GPR[10] = GPR[24] + sign\_ext(5)$
16	$addi$ \$22,\$24,6	$RESULT = \{(GPR[24] + sign\_ext(6))[31], GPR[24] + sign\_ext(6)\}$ $if(RESULT[32]! = RESULT[31])\ overflow = 1$ $if(overflow)\ REG[30] = 1$

		$else\ REG[30] = 0; REG[22] = RESULT[31:0]$
17	$start2: sw$ \$9,0(\$8)	$MEM[8] = GPR[9]$
18	lw \$14,0(\$8)	$GPR[14] = MEM[8]$
19	sw \$10,4(\$8)	$MEM[9] = GPR[10]$
20	lw \$15,4(\$8)	$GPR[15] = MEM[9]$
21	sw \$4, −4(\$8)	$MEM[7] = GPR[4]$
22	lw \$18, −4(\$8)	$GPR[18] = MEM[7]$
23	$addu$ \$4, \$0, \$8	$GPR[4] = GPR[0] + GPR[8]$
24	$addu$ \$5, \$0, \$9	$GPR[5] = GPR[0] + GPR[9]$
25	$jal\ newadd$	$GRG[31] = PC + 4$ $PC \leftarrow newadd$
26	slt \$25, \$10, \$8	$GPR[25] = (GPR[10] < GPR[8])?\ 1: 0$
27	beq \$25, \$0, $end2$	$zero = (GPR[25] - GPR[0])?\ 0: 1$ $if(zero)\ PC \leftarrow end2$
28	slt \$20, \$12, \$4	$GPR[20] = (GPR[12] < GPR[4])?\ 1: 0$
29	beq \$20, \$0, $end1$	$zero = (GPR[20] - GPR[0])?\ 0: 1$ $if(zero)\ PC \leftarrow end1$
30	lui \$12,65535	$GPR[12] = \{65535, \{16\{0\}\}\}$
31	$end1: ori$ \$0, \$0,1	$None$
32	lui \$19,0$xefef$	$GPR[19] = \{0xefef, \{16\{0\}\}\}$
33	$addiu$ \$3, \$0,0$xababcdcd$	$GPR[3] = GPR[0] + 0xababcdcd$
34	$start3: addiu$ \$4, \$3,2	$GPR[4] = GPR[3] + sign\_ext(2)$
35	$addi$ \$23, \$3,5	$RESULT = \{(GPR[24] + sign\_ext(5))[31], GPR[24] + sign\_ext(5)\}$ $if(RESULT[32]! = RESULT[31])\ overflow = 1$ $if(overflow)\ REG[30] = 1$ $else\ REG[30] = 0; REG[23] = RESULT[31:0]$
36	$jal\ newadd$	$GRG[31] = PC + 4$ $PC \leftarrow newadd$
37	$addu$ \$8, \$0, \$2	$GPR[8] = GPR[0] + GPR[2]$
38	$addu$ \$4, \$0, \$8	$GPR[4] = GPR[0] + GPR[8]$
39	$addu$ \$5, \$0, \$9	$GPR[5] = GPR[0] + GPR[9]$
40	$jal\ newadd$	$GRG[31] = PC + 4$ $PC \leftarrow newadd$
41	$addu$ \$9, \$0, \$2	$GPR[9] = GPR[0] + GPR[2]$
42	$addu$ \$9, \$8, \$0	$GPR[9] = GPR[8] + GPR[0]$
43	lui \$10,0$x69$	$GPR[10] = \{0x69, \{16\{0\}\}\}$
44	beq \$8, \$9, $start4$	$zero = (GPR[8] - GPR[9])?\ 0: 1$ $if(zero)\ PC \leftarrow start4$
45	beq \$0, \$0, $start3$	$zero = (GPR[8] - GPR[9])?\ 0: 1$ $if(zero)\ PC \leftarrow start3$
46	$start4: j\ end$	$PC \leftarrow end$
47	$newadd: addu$ \$2, \$4, \$5	$GPR[2] = GPR[4] + GPR[5]$

48	$addi\ \$0,\$12,0x1234$	$None$
49	$jr\ \$31$	$PC \leftarrow GPR[31]$
50	$end2: addi\ \$26,\$0,0x5678$	$RESULT = \{(GPR[0] + sign\_ext(0x5678))[31], GPR[24] + sign\_ext(0x5678)\}$ $if(RESULT[32]! = RESULT[31])\ overflow = 1$ $if(overflow)\ REG[30] = 1$ $else\ REG[30] = 0; REG[26] = RESULT[31:0]$
51	$end:$	$None$

3.3 测试结果

完整波形截图如图 3.3.1 所示，其中第一行为 clk 信号，第三行为 overflow 信号，共执行 69 行指令。最终寄存器结果如图 3.3.2 所示，存储器结果如图 3.3.3 所示，mars 寄存器、存储器结果如图 3.3.4、3.3.5 所示.

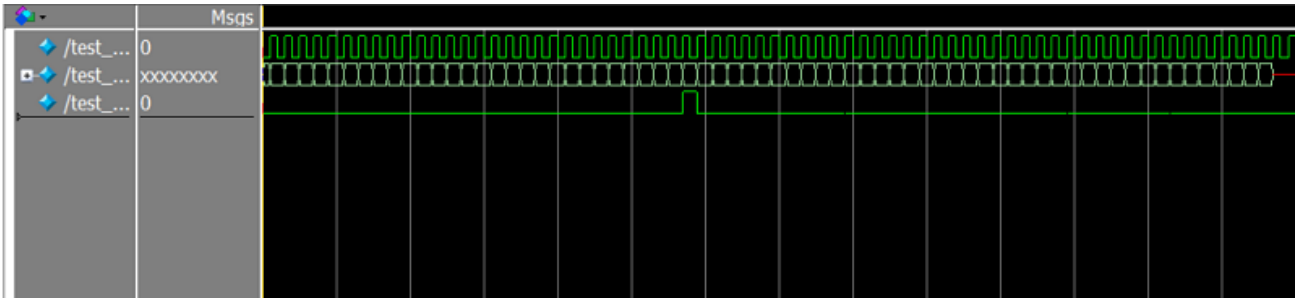


图 3.3.1 完整波形截图

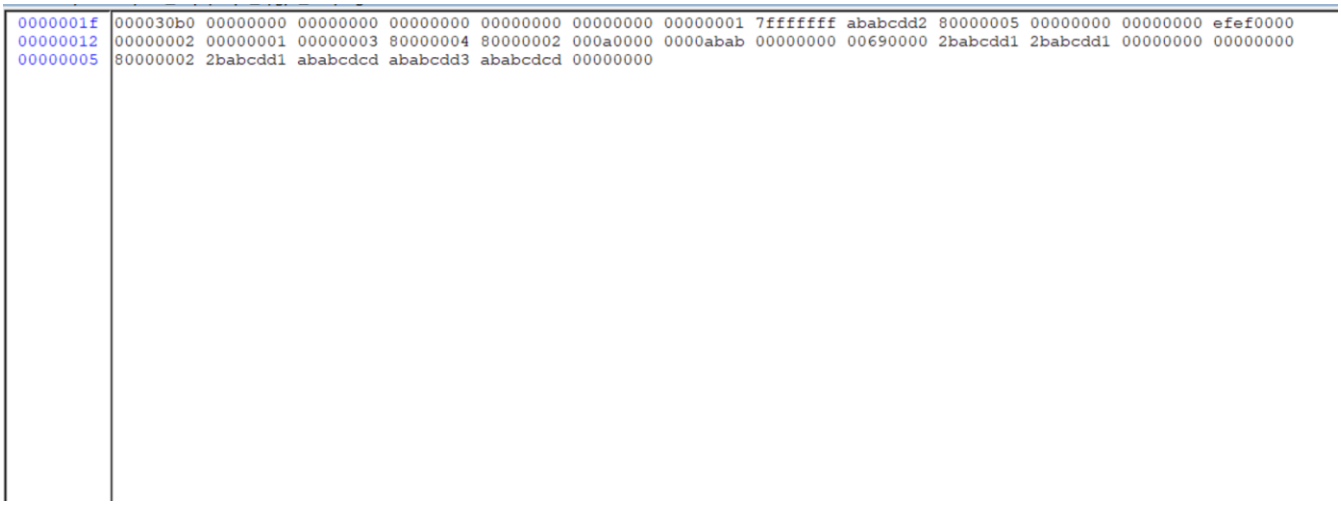


图 3.3.2 Modelsim 寄存器结果截图

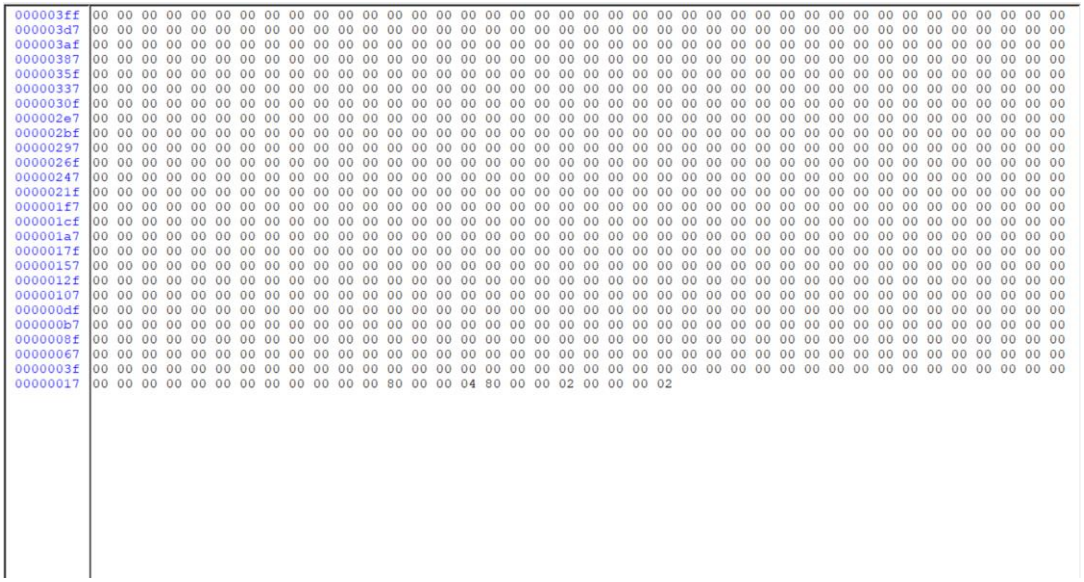


图 3.3.3Modelsim 存储器结果

Name	Number	Value
$zero	0	0x00000000
$at	1	0xababcdcd
$v0	2	0xababcdd3
$v1	3	0xababcdcd
$a0	4	0x2babcdd1
$a1	5	0x80000002
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x2babcdd1
$t1	9	0x2babcdd1
$t2	10	0x00690000
$t3	11	0x00000000
$t4	12	0x0000abab
$t5	13	0x000a0000
$t6	14	0x80000002
$t7	15	0x80000004
$s0	16	0x00000003
$s1	17	0x00000001
$s2	18	0x00000002
$s3	19	0xefef0000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0xababcdd2
$t8	24	0x7fffffff
$t9	25	0x00000001
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x00001800
$sp	29	0x00002ffc
$fp	30	0x00000000
$ra	31	0x000030b0
pc		0x000030d8
hi		0x00000000
lo		0x00000000

图 3.3.4 MARS 寄存器结果

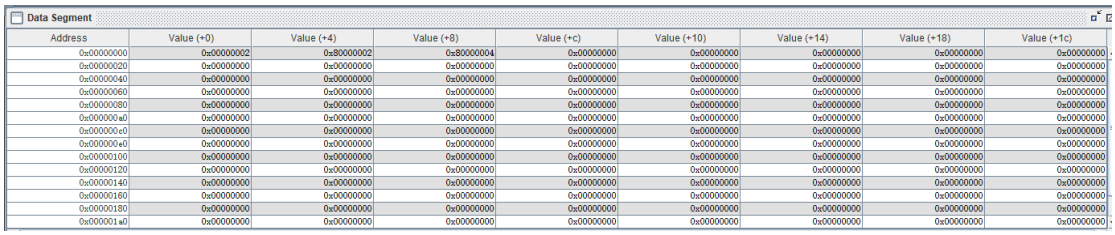


图 3.3.5 MARS 存储器结果

4 拓展指令设计

4.1 设计思路

拓展指令 BGTZ 功能为：若 rs 寄存器内容大于 0 则以相对跳转方式进行跳转，如果不大于 0 则无效果。

对于此功能的设计，首先要清楚如何判断数据是否大于 0。显然直接令 input>0 的方式是无法实现的，我采用的方法是 MIPS 手册上提供的判断方式：若符号位为 0 且数值位大于 0 则说明数据大于 0，此时置符号位 condition 为 1，传入 IFU 模块。在 IFU 模块中，BGTZ 的相对跳转方式和先前已经设计的 BEQ 指令相同，因此直接在判断 BEQ 指令的地方或上 condition 信号，使用 BEQ 跳转指令下的指令即可实现跳转。

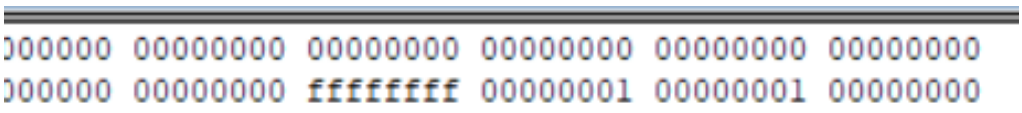
4.2 指令设计

思路如下：初始化 1 号寄存器为 1，此时 1 号寄存器大于 0，跳转至 CON1，将 2 号寄存器赋值为 0，此步若 BGTZ 不正常工作 2 号寄存器不会发生变化。随后 3 号寄存器赋值为 -1，此时小于 0，程序结束，此步若 BGTZ 不正常工作 4 号寄存器会写入 1。

```
1. ori $1,$0,1
2. bgtz $1,CON1
3. j CON2
4. CON1: ori $2,$0,1
5. CON2: subu $3,$0,$1
6. bgtz $3,CON3
7. j CON4
8. CON3: ori $4,$0,1
9. CON4:
```

4.3 测试结果

仿真后寄存器内容如图 3.4.3.1 所示，MARS 运行结果如图 3.4.3.2 所示，二者完全符合，说明 BGTZ 设计无错误。



3.4.3.1 仿真结果

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000001
$v0	2	0x00000001
$v1	3	0xffffffff
$a0	4	0x00000000
$a1	5	0x00000000

3.4.3.2 MARS 运行结果

5 总结

得益于先前大作业的基础和魏老师详细的课上讲解，本次期末作业对于单周期 cpu 的设计整体比较顺利。这次的实践工作让我对 cpu 有了更进一步的认识，感受到了计算机前辈的精妙设计，也为后续的继续深入探究打下了良好的基础。

实践作业总是麻烦与欣喜共存的。每次实践作业总会在很多意想不到的地方给我带来困扰，比如此次对于 vscode 的配置、对于 modelsim 的功能调试等，但每次解决后的成就感便是我长久以来学习的动力。本次设计令我感触最深的是在设计过程中应当给予设计手册十足的重视，所有的设计都应当在完全理解手册要求后进行，不可想当然地自行设计。这不仅是我个人遇到的问题，也是我在为他人解答问题时突出受到感触的地方。另外，我也愈发认为独立解决问题的能力是一个人的核心能力，遇到问题如何通过一步步地调试、通过准确查找相关资料是在学习过程中无论如何也避不开的问题，也是我们需要十分重视的问题。

最后，感谢在设计过程中帮助到我的同学，感谢魏老师辛勤的教学工作！