计算机组成原理

课内大作业报告

学　　号_____20074221_____

姓　　名_____游佳慧_____

指导教师_____魏坚华_____

提交日期_____2022.5.8_____

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

一、总体数据通路结构设计

1.1 总体数据通路结构图

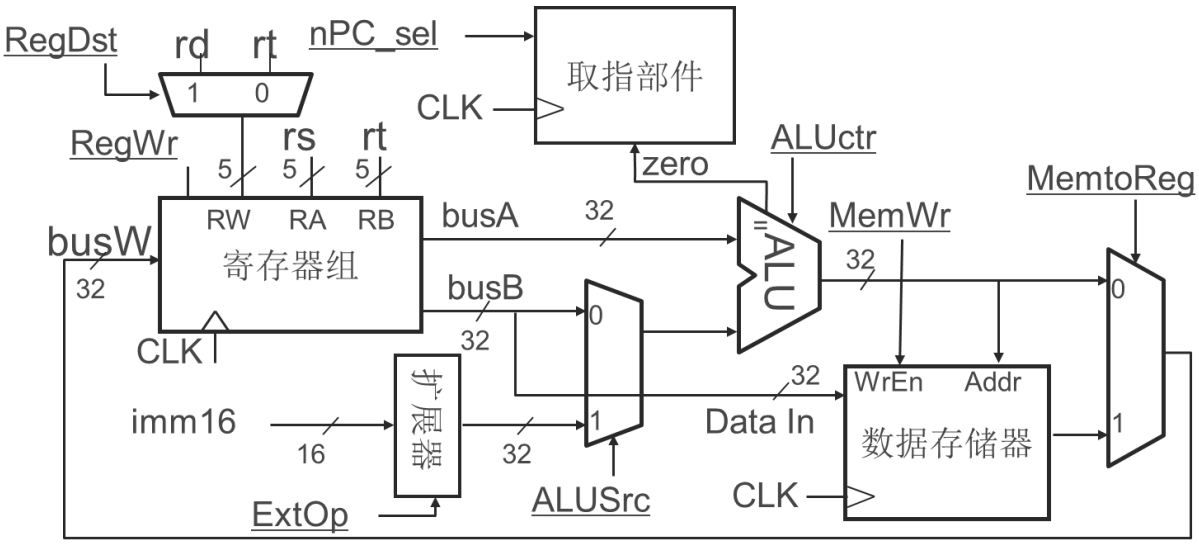


图 1：总体数据通路结构图

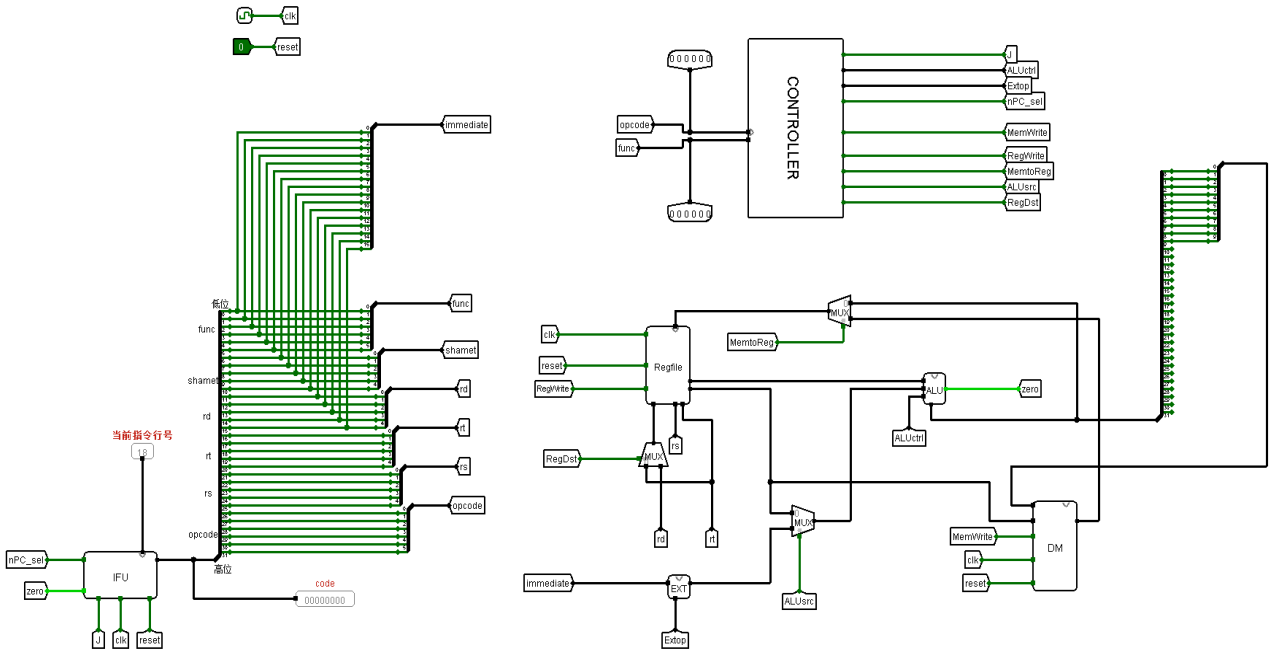


图 2：顶层设计

1.2 模块设计图

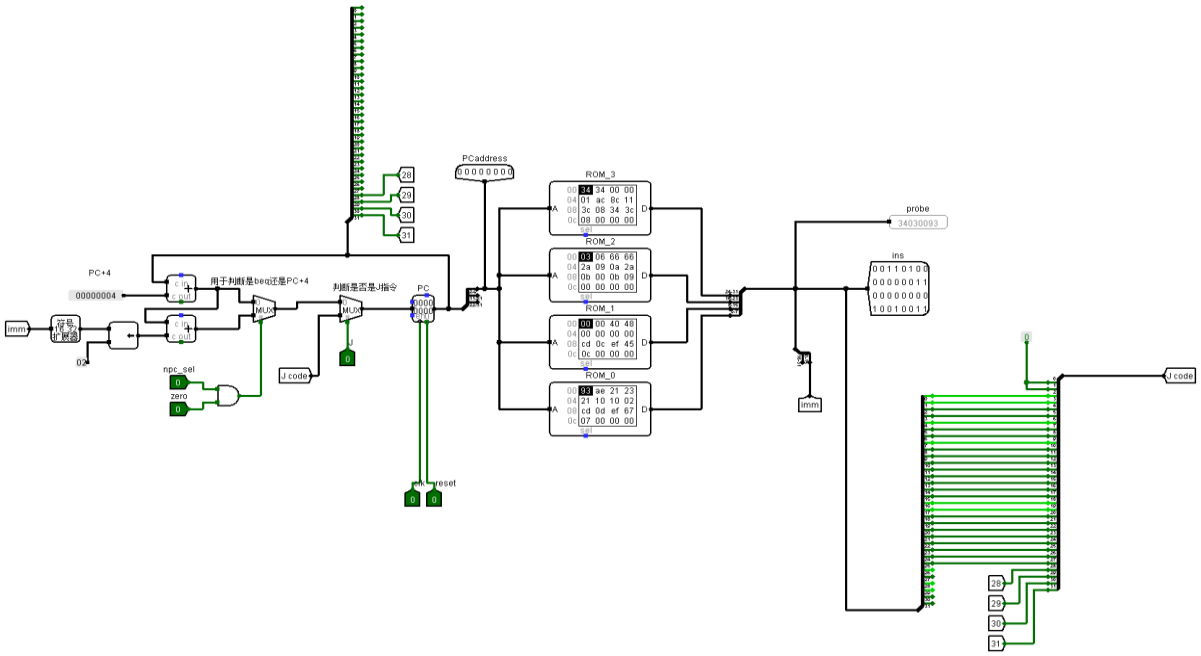


图 3：IFU 模块设计图

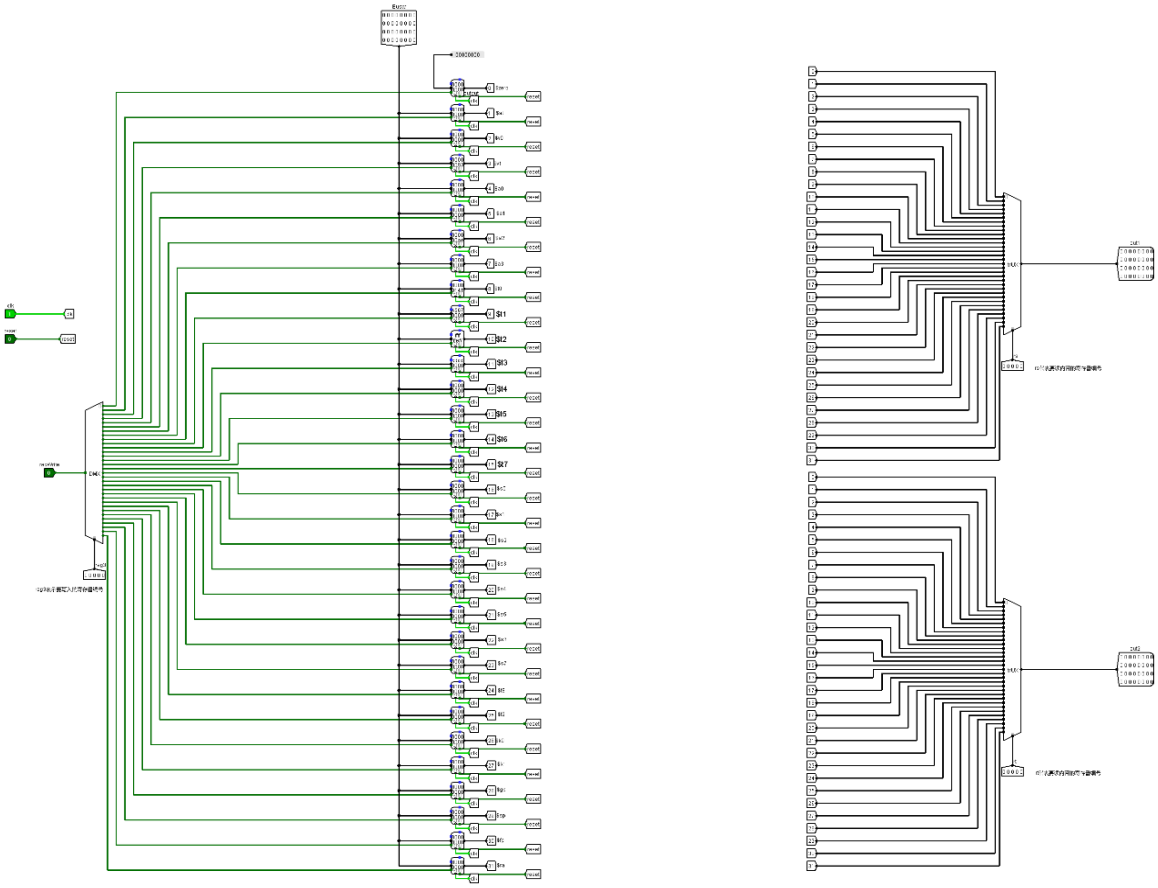


图 4：GPR 模块设计图

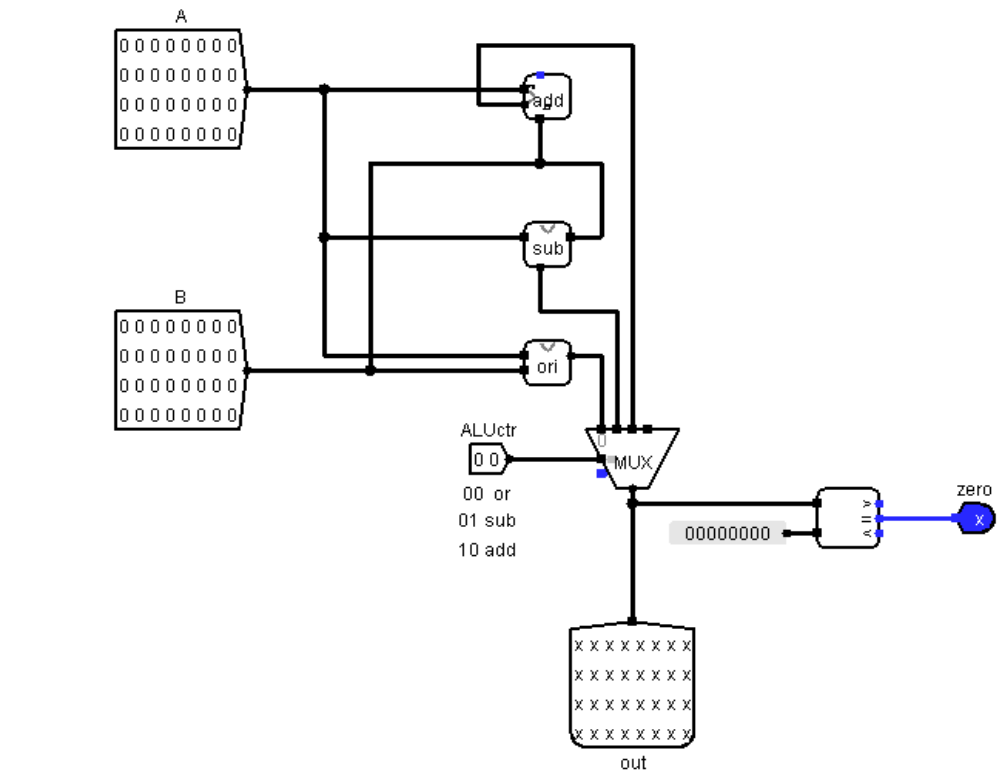
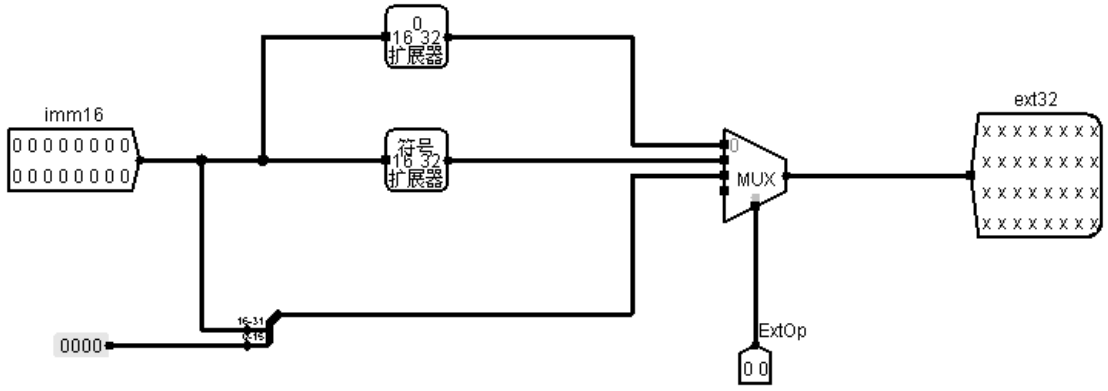


图 5：ALU 模块设计图



00　0拓展

01 符号拓展

10 lui高位复制

图 6：EXT 模块设计图

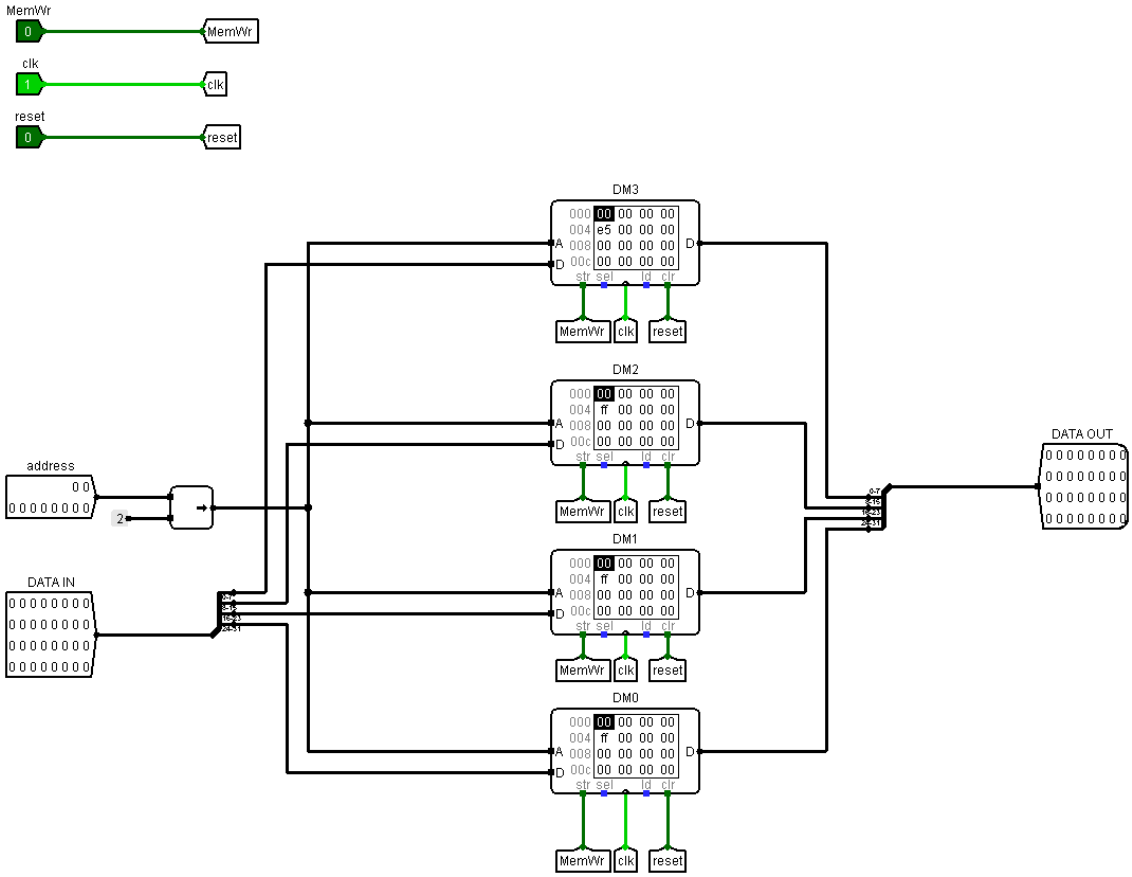


图 7：DM 模块设计图

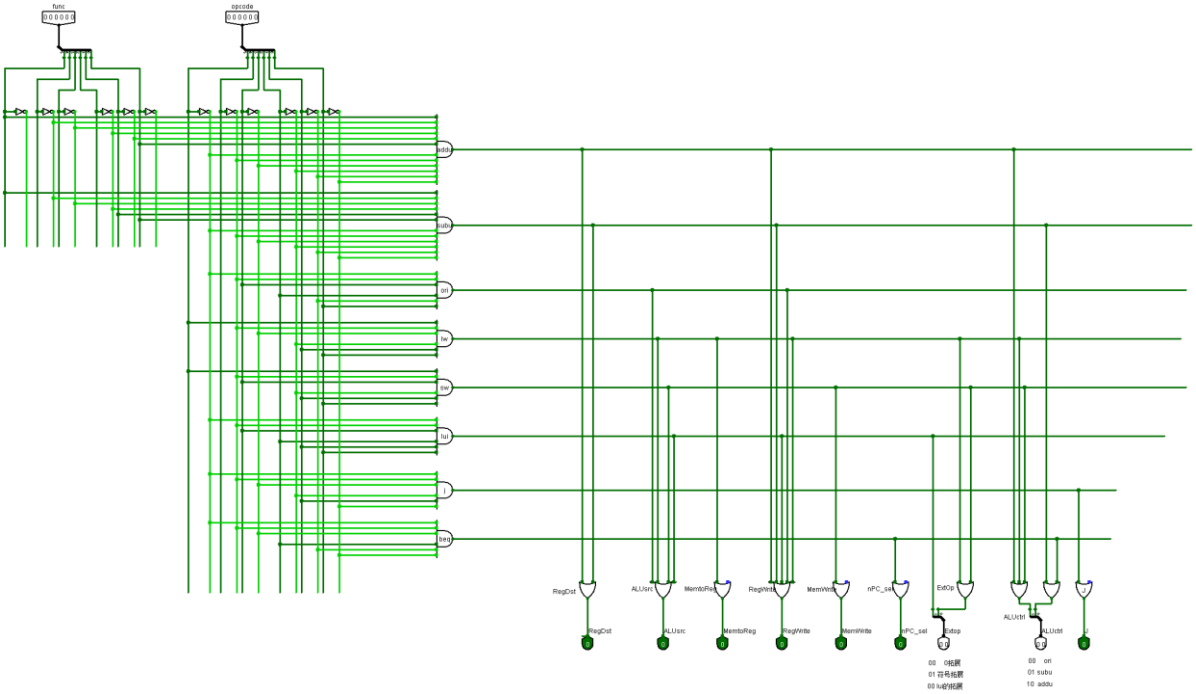


图 8：Control 模块设计图

二、模块定义

2.1 IFU 模块定义

2.1.1 基本描述

IFU 主要功能是完成取指令功能。IFU 内部包括 PC、IM(指令存储器)以及其他相关逻辑。IFU 除了能执行顺序取值令外，还能根据 BEQ 指令的执行情况决定顺序取值令还是转移取值令。

2.1.2 模块接口

信号名	方向	描述
npc_sel	I	当前指令是否为 beq 指令标志。 1：当前指令为 beq 0：当前指令非 beq
zero	I	ALU 计算结果为 0 标志。 1：计算结果为 0 0：计算结果非 0
clk	I	时钟信号
reset	I	复位信号。 1：复位 0：无效
ins[31:0]	O	32 位 MIPS 指令
PCaddress	O	查看 IFU 输出第几条指令

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 被设置为 0x00000000。
2	取指令	根据 PC 从 IM 中取出指令。
3	计算下一条指令地址	如果当前指令不是 beq 指令，则 PC←PC+4 如果当前指令是 beq 指令，且 zero 为 0，则 PC←PC+4 如果当前指令是 beq 指令，并且 zero 为 1，则 PC←PC+4+(sign_ext(ins[15:0])<<2)
4	计算当前输出指令行号	PCaddress 统计 IFU 中被取指令的条数

2.2 GPR 模块定义

2.2.1 基本描述

GPR 的主要功能是完成对 32 个 32 位寄存器的读写功能，有两个读端口和一个写端口，根据 RA 和 RB 的值分别选择要读取的寄存器，根据 RW 的值选择要写入的寄存器。

2.2.2 模块接口

信号名	方向	描述
regWrite	I	当前指令是否写入寄存器。 1：当前指令写入寄存器 0：当前指令不写入寄存器
reset	I	复位信号。 1：复位 0：无效
clk	I	时钟信号。
rs[4:0]	I	当前指令需要被读出数据的寄存器地址 1。
rt[4:0]	I	当前指令需要被读出数据的寄存器地址 2。
rw[4:0]	I	当前指令需要写入的寄存器地址。
Busw[31:0]	I	当前指令需要写入的数据。
out1[31:0]	O	当前指令读出的数据 1。
out2[31:0]	O	当前指令读出的数据 2。

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，所有寄存器数据清零。
2	取数据	根据 rs 和 rt 的地址从寄存器中取出数据。
3	写数据	如果 regWrite 有效且 clk 时钟信号触发边沿，则根据 rw 的地址将数据写入该地址所对应寄存器中。

2.3 ALU 模块定义

2.3.1 基本描述

ALU 的主要功能是完成算术运算和逻辑运算，本次设计的 ALU 可执行的算术运算包括加法和减法，逻辑运算包括或运算。多路选择器根据 ALU 控制信号判断 ALU 应进行的运算，选择出运算结果后对其做按位与运算判断是否为 0。

2.3.2 模块接口

信号名	方向	描述
A	I	参与运算的第一个输入数据。
B	I	参与运算的第二个输入数据
ALUctr	I	ALU 控制信号。 00：或运算 01：减法运算 10：加法运算
out[31:0]	O	ALU 运算结果。
zero	O	运算结果是否为零的标志位。 1：运算结果为 0 0：运算结果非 0

2.3.3 功能定义

序号	功能名称	功能描述
1	加法运算	ALUctr=10 时，out=A+B
2	减法运算	ALUctr=01 时，out=A-B
3	或运算	ALUctr=00 时，out=A|B

2.4 EXT 模块定义

2.4.1 基本描述

EXT 的主要功能是完成 16 位立即数扩展，根据 EXTop 信号的不同值分别进行 0 扩展、符号位扩展或 lui 指令高位复制扩展，扩展为 32 位立即数输出。

2.4.2 模块接口

信号名	方向	描述
imm16[15:0]	I	需要扩展的 16 位立即数。
Extop	I	符号扩展控制信号。 00：高位 0 扩展 01：符号位扩展 10：低 16 位补零扩展
Imm32[31:0]	O	完成扩展的 32 位立即数。

2.4.3 功能定义

序号	功能名称	功能描述
1	0 扩展	高 16 位补零
2	符号位扩展	最高有效位（符号位）复制填满高 16 位
3	低 16 位补零扩展	低 16 位补零

2.5 DM 模块定义

2.5.1 基本描述

DM 是数据存储器，主要功能是完成存储器读写。当写入使能有效时，根据输入的地址将输入的数据写入存储器的相应位置，并输出从该地址读取的数据。

2.5.2 模块接口

信号名	方向	描述
address	I	需要读或写的存储器地址。
DataIn[31:0]	I	需要写入的数据。
reset	I	复位信号。 1：复位 0：无效
clk	I	时钟信号。
MemWr	I	写入使能信号。 1：允许写入 0：不允许写入
DataOut[31:0]	O	从输入地址读出的数据。

2.5.3 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，所有寄存器数据清零。
2	读数据	根据输入的寄存器地址读出数据。
3	写数据	根据输入的地址将输入数据写入存储器的相应位置。

2.6 Control 模块定义

2.6.1 基本描述

Control 主要功能是完成对指令功能的判断和确定每条指令对应的控制信号。根据输入指令的操作码和功能码判断指令，并输出各单元的写使能信号、各多选器的选择信号和 ALU 的控制信号。

2.6.2 模块接口

信号名	方向	描述
func[5:0]	I	当前指令的功能码。
opcode[5:0]	I	当前指令的操作码。
ALUctr	O	ALU 控制信号。 00：或运算 01：减法运算 10：加法运算
ALUSrc	O	选择第二个 ALU 操作数。 0：操作数为寄存器取出的值 1：操作数为经 EXT 扩展后的 32 位立即数
Extop	O	控制 EXT 的扩展方式
nPC_sel	O	beq 指令标志。 1：是 beq 指令 0：非 beq 指令
MemWrite	O	DM 写使能信号。
RegWrite	O	GPR 写使能信号。
MemtoReg	O	选择写入寄存器的数据。 0：写入的数据是 ALU 计算输出结果 1：写入的数据是 DM 输出结果
RegDst	O	写入寄存器的目标寄存器号来源。
J	O	控制 PC 是否转移到 J 指令的地址。

2.6.3 功能定义

序号	功能名称	功能描述
1	产生控制信号	对输入指令的所有控制信号赋值。

三、设计的机器指令描述

指令操作码助记符	机器指令代码		指令功能
	opcode	funct	
addu	000000	100001	分别从 rs 和 rt 寄存器中取出两个数做无符号数加法，结果放入 rd 寄存器中。
subu	000000	100011	分别从 rs 和 rt 寄存器中取出两个数做无符号数减法，结果放入 rd 寄存器中。
ori	001101		从 rs 寄存器中取出一个数与高位零扩展后的 16 位立即数做或运算，结果放入 rt 寄存器。
lui	110000		从 rs 寄存器中取出一个数与低位补零扩展后的 16 位立即数做或运算，结果放入 rt 寄存器。
sw	101011		从 t 寄存器中取出一个数，根据基地址+偏移
lw	100011		根据基地址+偏移量算出地址，从该地址对应存储器中取出一个数存入 rt 寄存器中。
beq	000100		分别从 s 和 t 寄存器中取出两个数比较是否相等，若相等则进行分支跳转，PC<-PC+4+符号扩展 imm16, 若不相等则不跳转，PC<-PC+4。
j	000010		无条件跳转，PC<-{PC+4[31:28] imm26}。

四、测试程序

机器码	指令	注释说明
34030093	ori $3,$0,0x93	立即数 0x93 和$0 的内容按位或存入$3
340600ae	ori $6,$0,0xae	立即数 0xae 和$0 的内容按位或存入$6
00664021	addu $8,$3,$6	$3 和$6 的内容无符号相加存入$8
00664823	subu $9,$3,$6	$3 的内容减去$6 的内容存入$9
012a0021	addu $0,$9,$10	$9 和$10 的内容无符号相加存入$0
ac090010	sw $9,16($0)	$9 的内容写入以$0 内容为基地址偏移 16 个字节地址指向的存储器单元
8c0a0010	lw $10,16($0)	取出以$0 内容为基地址偏移 16 个字节地址指向的存储器单元内容存入$10
112a0002	l3:beq $9,$10,l1	比较$9 和$10 内容，若相等则跳转至 l1
3c0bcdcd	lui $11,0xcdcd	将 0xcdcd 存入$11 高 16 位，低 16 位补零
08000c0d	j end	无条件跳转至 end
340befef	l1:ori $11,$0,0xefef	立即数 0xefef 和$0 的内容按位或存入$11
3c094567	lui $9,0x4567	将 0x4567 存入$9 高 16 位，低 16 位补零
08000c07	j l3	无条件跳转至 l3
	end:	end

五、测试结果

5.1 GPR 运行结果

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000093
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x000000ae
$a3	7	0x00000000
$t0	8	0x00000141
$t1	9	0x45670000
$t2	10	0xfffffffe5
$t3	11	0xcdcd0000
$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
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x00001800
$sp	29	0x00002ffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00003034
hi		0x00000000
lo		0x00000000

图 9：Mars 中的 GPR

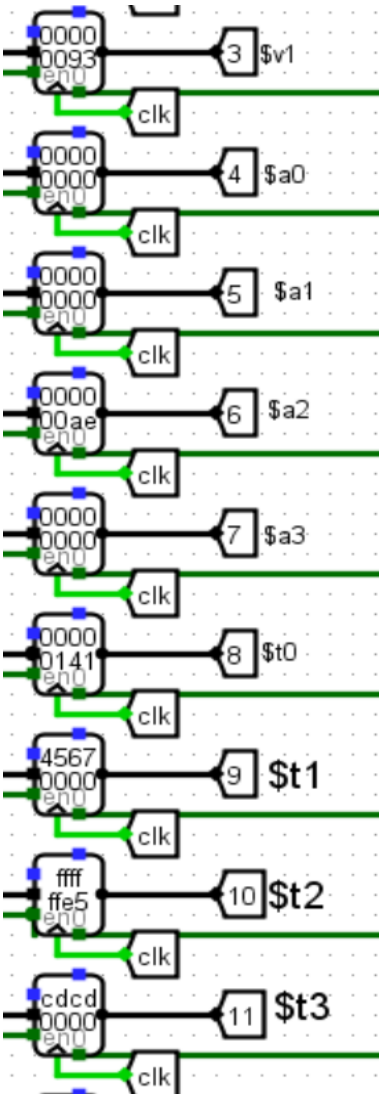


图 10：logisim 中的 GPR

说明：由图可知，在 Mars 中仿真后寄存器的结果与在 logisim 中的寄存器结果一致。

5.2 DM 运行结果

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)
0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0xffffffe5	0x00000000
0x00000020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000000a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000000c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000000e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

图 11：Mars 中的 DM

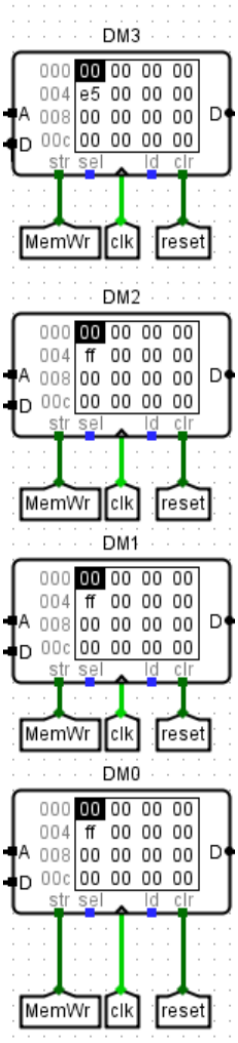


图 12：logisim 中的 DM

说明：在 Mars 中仿真后数据存储器的结果与在 logisim 中的数据存储器结果一致。

六、总结与收获

本次大作业的内容是利用 Logisim 平台和 Mars 仿真器构建 32 位单周期 CPU 处理器。从分析搭建总体架构，到分布设计各个模块，再到将各个模块进行连接，从而形成完整的数据通路，一步步自己搭建 CPU 的过程加深了我对每个指令的理解，对 MIPS 体系结构设计有了一定的认识，也加强了我对于模块化层次化设计的能力，构造数字系统时采用分层设计和模块设计可以将复杂的问题拆解成若干容易解决的小问题，降低了解决问题的难度，完成这次大作业的经历对于我之后的学习有很大帮助。