华中科技大学

2023

计算机组成原理 ·实验报告·

专　　业：	网络空间安全
班　　级：	网安 2104 班
学　　号：	U202112131
姓　　名：	邬雪菲
电　　话：	18172029686

评　分　表

评　分　项			总分	得分
实验	单周期MIPS CPU	设计思路	15	
		电路图	5	
		测试图	5	
		故障分析	10	
	多周期MIPS CPU	设计思路	15	
		电路图	5	
		测试图	5	
		故障分析	10	
设计总结	实验总结		10	
	实验心得		10	
格式	段落缩进是否规范		2	
	图表及编号是否规范		2	
	字号字体是否统一		2	
	是否签名		1	
	其他		3	
合　　计				
教　师　签　名				

目 录

1 实验概述

1.1 实验名称

CPU 设计---单周期和多周期 MIPS CPU(8 条指令)。

1.2 实验所需软件及设备

(1) Logisim2.7.1 软件 1 套
(2) 微型计算机 1 台；

1.3 实验目的

理解 MIPS 单周期和多周期处理器的基本原理，能分别利用硬布线控制器和微程序控制器的设计原理在 Logisim 平台中设计实现 MIPS 单周期和多周期处理器。

1.4 实验基础

硬布线和微程序控制器基本原理、MIPS 指令执行流程、寄存器相关知识、存储器访问相关基础知识等。

1.5 实验内容及要求

完成单周期硬布线控制器和多周期微程序控制器电路的设计。

了解单周期硬布线控制器和多周期微程序控制器电路的基本概念及使用原理，支持表 1 中的 8 条核心指令，实现 MIPS 处理器能运行实验包中 sort 排序程序，利用冒泡排序对数据进行升序排序。

表 1 MIPS 指令及功能描述

#	MIPS 指令	RTL 功能描述
1	add $rd,$rs,$rt	R[$rd]←R[$rs]+R[$rt],溢出时产生异常,且不修改 R[$rd]
2	slt $rd,$rs,$rt	R[$rd]←R[$rs]<R[$rt],小于置 1,有符号比较
3	addi$rt,$rs,imm	R[$rt]←R[$rs]+$SignExt_{16b}$(imm),溢出产生异常
4	lw$rt,imm($rs)	R[$rt]←$Mem_{4B}$(R[$rs]+$SignExt_{16b}$(imm))
5	sw$rt,imm($rs)	Mem_{4B}(R[$rs]+$SignExt_{16b}$(imm))←R[$rt]
6	beq $rs,$rt,imm	if(R[$rs]=R[$rt]) PC ← PC+$SignExt_{18b}$({imm,00})
7	bne $rs,$rt,imm	if(R[$rs]!=R[$rt]) PC ← PC+$SignExt_{18b}$({imm,00})
8	syscall	系统调用,这里用于停机

2 CPU 设计实验

2.1 单周期 MIPS CPU 设计

（1）设计思路

（1.1）总体设计思想

单周期 MIPS CPU 特点：

单周期 MIPS CPU 指令定长，且所有指令必须在一个时钟周期内完成，整个 CPU 的性能取决于最慢的指令。单周期 MIPS CPU 不能设 AR 、DR 和 IR 寄存器，采用程序和数据分开存放的哈佛结构，且运算器和 PC 累加器分离。

设计原理和思路：

CPU 的基本设计步骤：第一，分析设计指令集；第二，建立数据通路，连接各个功能组件；第三，分析出所有需要的控制信号，建立控制信号真值表，设计指令信号控制逻辑；第四，用逻辑电路控制形成信号真值表，构建产生控制信号的控制部件。

设计思路：

首先按照题目框架实现单周期硬布线控制器基本电路，然后再考虑要求实现的八个指令。查询手册得到各指令的指令字段信息，并分析各个指令执行时的数据通路和控制信号，列出控制信号真值表，由此完成指令译码逻辑部件从而完成硬布线控制器。再继续细化整体电路即可。

设计过程：

由于整体数据通路基本与实验所给通路一致，故不在此过多阐述，之后在单个指令数据通路中会体现设计原因。主要说明两处修改，第一是由于引入了停机操作 syscall，需要添加逻辑电路以实现停机，即增加一个计数器计算周期数，当周期达到最大周期数时停机。第二，由于 BEQ 指令和 BNE 指令的存在， 需要一个与门将 Beq 信号和 ALU 相等信号连接，需要一个与门将 Bne 信号和 ALU 相等信号取反连接，再连一个或门，连接到 PC 前的多路选择器上，由此可根据情况选择将 PC+4 或者 PC+(32[IMM]<<2)写入 PC。

（1.2） 各种指令数据通路

MIPS 有 R 型指令、I 型指令、J 型指令三种。R 型指令的 OP 字段全 0，根据 funct 字段区分不同的功能。而 I 型指令和 J 型指令都根据 OP 字段区分不同的功能。本实验中，查询 MIPS 指令手册可得到各个指令的指令字段，其中 add、slt 指令是 R 型指令，syscall 是特殊的 R 型指令，剩下的都是 I 型指令。

取指令数据通路如图 2.1 所示。为避免冲突，此处加法器必须与 ALU 区分开。从 PC 取出指令地址的同时，将 PC＋4，取出的地址送入指令存储器，接着执行指令。

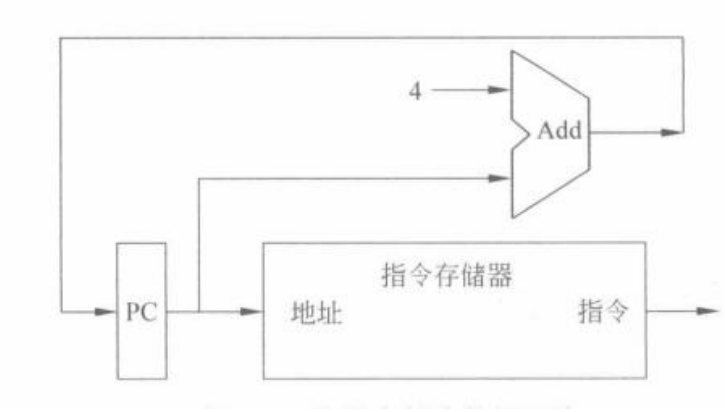


图 2-1-1 取指令数据通路图

R 型指令数据通路如图 2-1-2 所示。将指令字进行解析，得到源寄存器 Rs、Rt 和目的寄存器 Rd 的索引，输入寄存器组中，寄存器组输出 R1、R2 送入 ALU，ALU 运算后将结果送到写入端，时钟到来时，写入寄存器组中由 Rd 索引的寄存器。

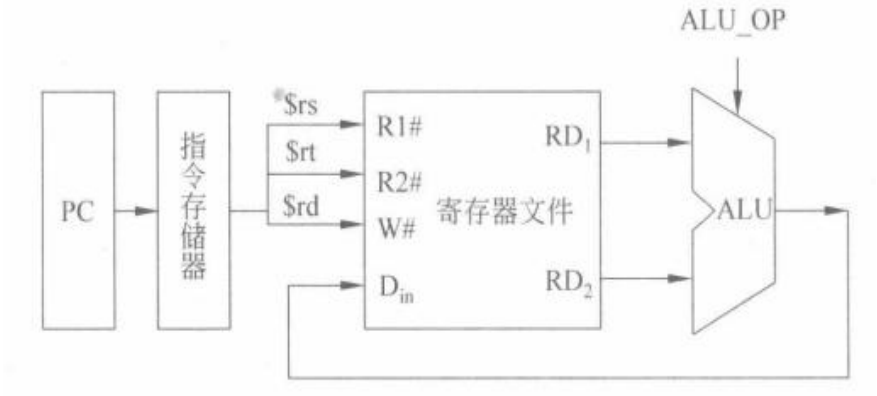


图 2-1-2 R 型指令数据通路图

访存指令数据通路如图 2-1-3 所示。只有 I 型指令中有访存指令，指令得到内存单元的地址=R[rs]+imm，imm 为 16 位立即数。因为寄存器的值为 32 位，所以需要

将 imm 符号扩展为 32 位，在和 R[rs]一同送入 ALU 中，将计算结果送入 RAM 的地址端。从 RAM 的输出端中得到数据，并送入寄存器的数据写入端中，最终根据 I 型指令的译码结果，写入对应的寄存器中。即 I 型指令中的 R[rt]。

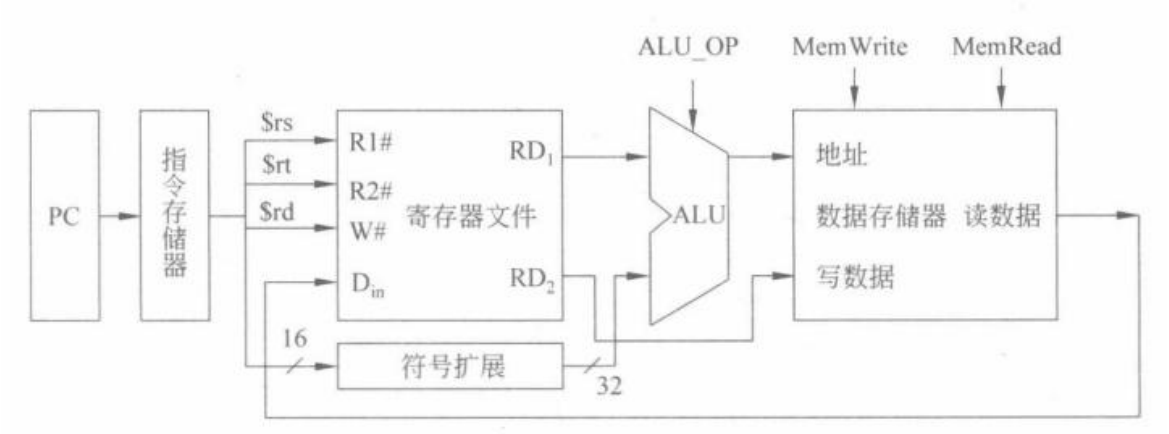


图 2-1-3 访存指令数据通路图

综合数据通路如图 2-1-4 所示。

由于 R 型指令有三个寄存器，I 型指令有两个寄存器，在综合的数据通路中，如果面对 R 型指令，就需要将寄存器 Rd 作为写入寄存器，如果面对 I 型指令，就需要将寄存器 Rt 作为写入寄存器,所以需要引入一个选择器和一个选择信号 RegDst 进行控制。同样，对于 ALU 而言，第二个操作数可能是 R 型指令中的 Rt，也可能是 I 型指令中的经过符号拓展的立即数，所以同样需要引入选择器和选择信号 AluSrc 进行控制。同样，在进行数据写回时，写回的数据可能是 ALU 的运算结果，也可能来自内存，同样需要引入选择器和选择信号 MemToReg 进行控制。

由于是单周期 CPU，需要将有相同功能器件分开，所以图中有三个加法器，分别用于 PC+4、计算数据、计算跳转地址。

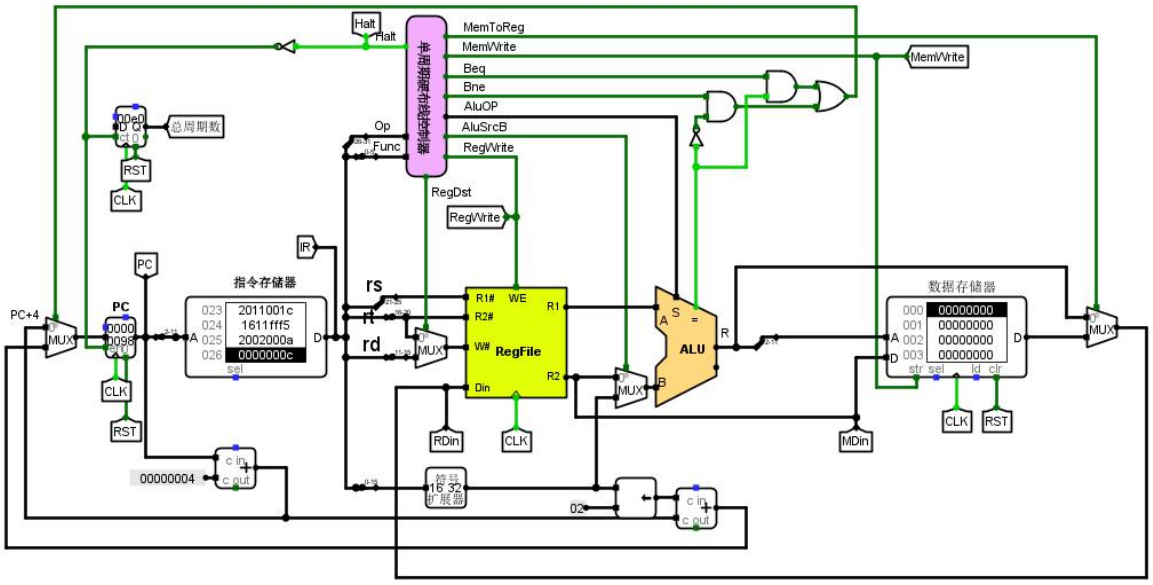


图 2-1-4 综合数据通路图

（1.3） 单周期 MIPS 控制器设计过程

具体设计思路如下：

首先对 OP 字段进行解码，利用比较器，如果 OP 不为 0，就是 I 型指令，根据具体的比较结果，控制相应的译码信号。如果 OP 为 0，就是 R 型指令，这时进一步看 FUNC 字段，同样利用比较器，当 FUNC=20、2a、0c 时，分别对应 ADD、SLT、SysCall 译码信号。

同时，在 SysCall=0 的前提下，ADD 或 SLT 为 1 时，当前指令都需要将结果写回到寄存器 Rd 中，所以利用组合逻辑使 R_TYPE=1，并将 R_TYPE 与控制信号 RegDst 的引脚相连。

对于 ALU，实验中涉及到的运算只有两种，利用选择器即可。当指令译码信号为 SLT 时，进行比较，其他时候 ALU 只进行加法运算。

根据指令译码信号，有如下逻辑：RegDst=R_TYPE；RegWrite = R_type & ADDI & LW 因为这三个指令译码信号出现意味着要写入寄存器； MemToReg = LW；MemWrite = SW；AluSrc = ADDI&LW&SW；Beq = BEQ；Bne = BNE；Halt = SysCall。

最后根据指令译码信号实现控制器输出控制信号逻辑，产生条件由图 2-1-5 给出：

#	控制信号	信号说明	产生条件
1	MemToReg	写入寄存器的数据来自存储器	lw指令
2	MemWrite	写内存控制信号	sw指令　未单独设置MemRead信号
3	Beq	Beq指令译码信号	Beq指令
4	Bne	Bne指令译码信号	Bne指令
5	AluOP	运算器操作控制符	加法，比较两种运算
6	AluSrcB	运算器第二输入选择	Lw指令，sw指令，addi
7	RegWrite	寄存器写使能控制信号	寄存器写回信号
8	RegDst	写入寄存器选择控制信号	R型指令
9	Halt	停机信号，取反后控制PC使能端	syscall指令

图 2-1-5　控制信号产生条件

（2）电路图

（2.1）硬布线控制器电路图

首先是指令译码逻辑的设计，该实验只涉及 8 条核心的 MIPS 指令。而这 8 条 MIPS 指令的指令字段已在附件中给出，因此，这部分只需根据相应的 OP 和 FUNC 字段进行简单地逻辑比较就可实现。

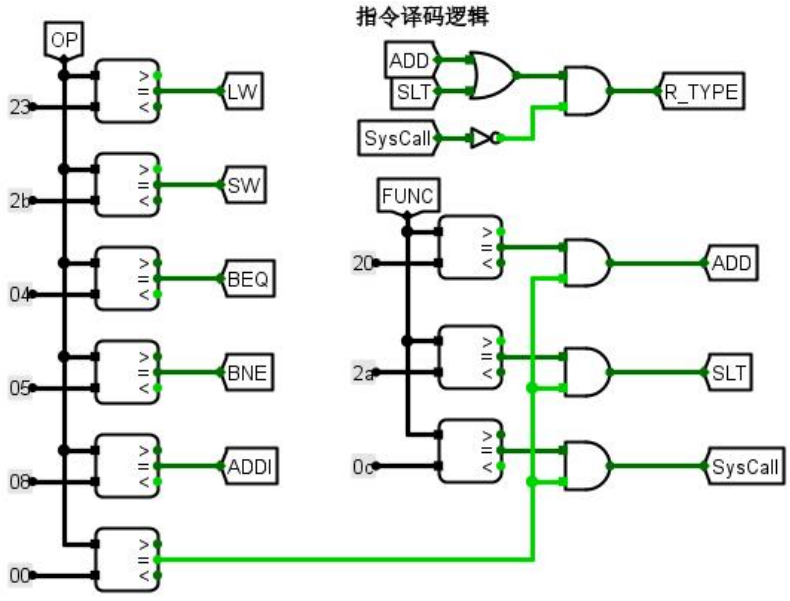


图 2-1-6 指令译码电路图

在 ALU 控制逻辑的设计中，8 条核心 MIPS 指令对于 ALU 运算逻辑单元只涉及到加法和比较，因此这一部分可以大大简化。只有运行 STL 指令时，需要选择比较运算，其余都是加法运算。

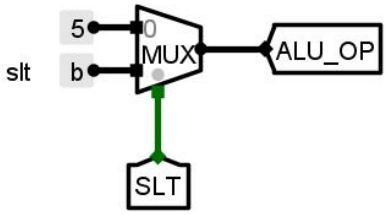


图 2-1-7 ALU 控制器电路图

对于控制器输出信号的设计，需要根据硬布线控制器中所包含的控制信号进行分析，如上文给出的图 2-1-5。主要考虑每种控制信号的产生条件。

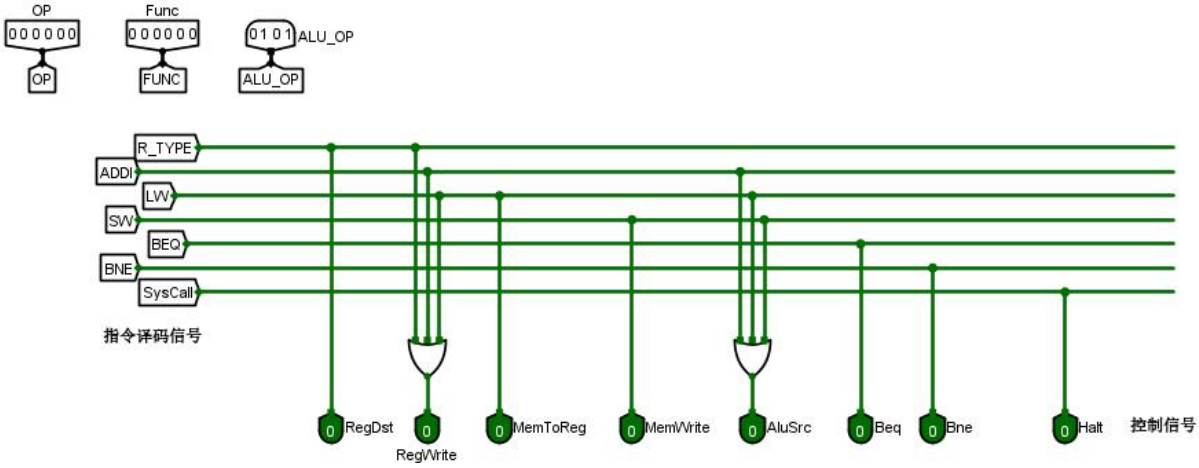


图 2-1-8 控制信号电路图

（2.2）CPU 设计图

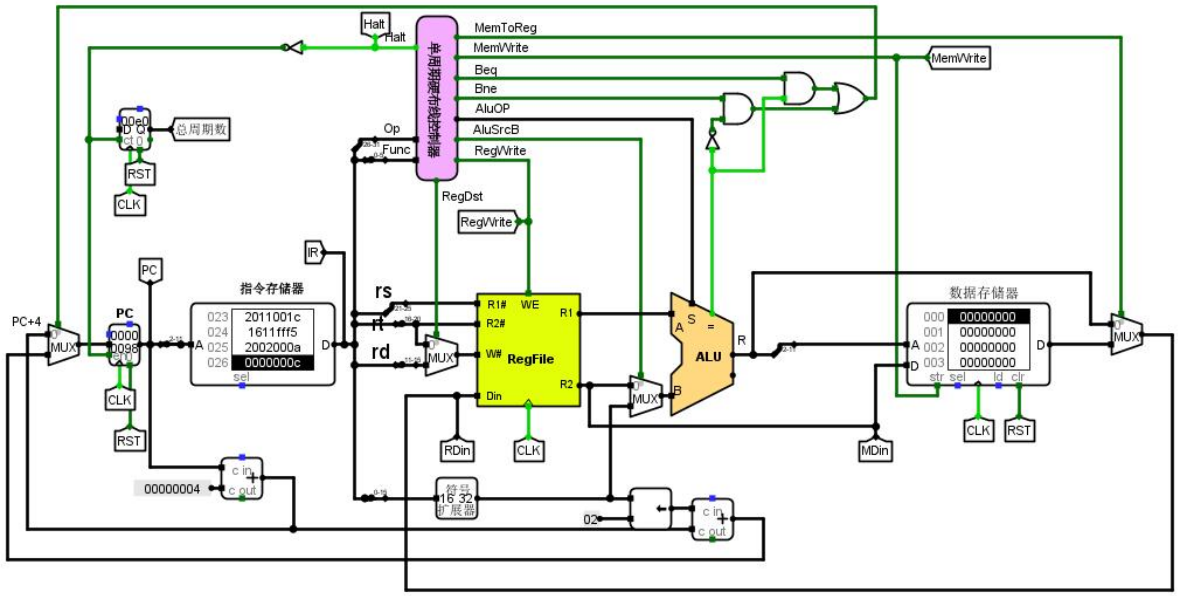


图 2-1-9 CPU 设计图

（3）测试图

将 sort.hex 文件载入指令寄存器，ctrl+k 启动时钟测试，在总周期数为 224 时停机，停机后状态和各引脚输出如图 2-1-10，数据存储器中数据排序情况如图 2-1-11，头歌平台测试通过结果如图 2-1-12，可见排序结果是有符号降序，符合设计要求。

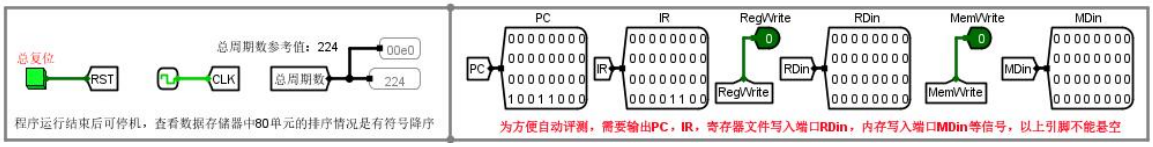


图 2-1-10 本地测试结果

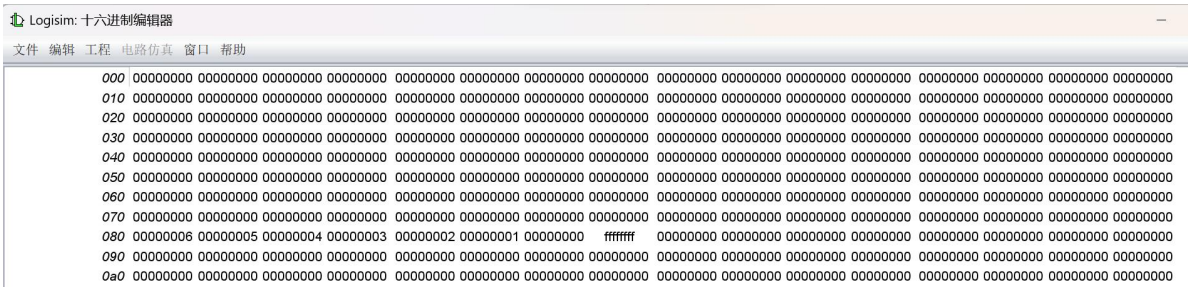
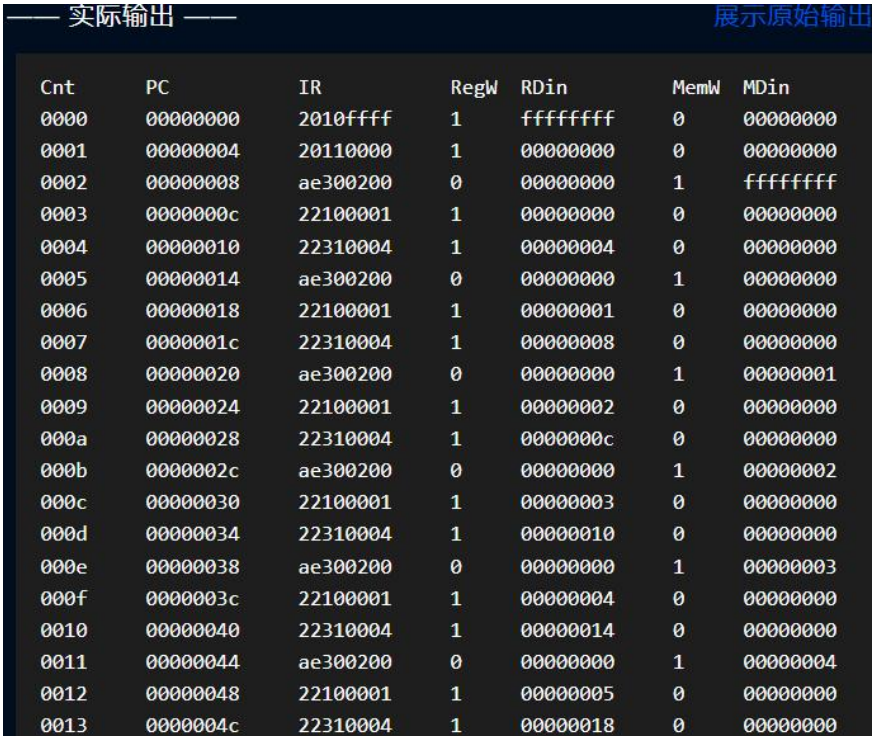


图 2-1-11 数据存储器测试结果



Cnt	PC	IR	RegW	RDin	MemW	MDin
0000	00000000	2010ffff	1	ffffffff	0	00000000
0001	00000004	20110000	1	00000000	0	00000000
0002	00000008	ae300200	0	00000000	1	ffffffff
0003	0000000c	22100001	1	00000000	0	00000000
0004	00000010	22310004	1	00000004	0	00000000
0005	00000014	ae300200	0	00000000	1	00000000
0006	00000018	22100001	1	00000001	0	00000000
0007	0000001c	22310004	1	00000008	0	00000000
0008	00000020	ae300200	0	00000000	1	00000001
0009	00000024	22100001	1	00000002	0	00000000
000a	00000028	22310004	1	0000000c	0	00000000
000b	0000002c	ae300200	0	00000000	1	00000002
000c	00000030	22100001	1	00000003	0	00000000
000d	00000034	22310004	1	00000010	0	00000000
000e	00000038	ae300200	0	00000000	1	00000003
000f	0000003c	22100001	1	00000004	0	00000000
0010	00000040	22310004	1	00000014	0	00000000
0011	00000044	ae300200	0	00000000	1	00000004
0012	00000048	22100001	1	00000005	0	00000000
0013	0000004c	22310004	1	00000018	0	00000000

图 2-1-12 头歌测试结果

（4）故障与测试分析

(4.1)

故障现象 1：educoder 上输出 xxxx

原因分析：无意中改变了引脚

解决方案：由于电路已经完成且电路本身比较复杂，无法估计是哪一步改动了引脚，直接回退不太现实，故手动纠正引脚位置

(4.2)

故障现象 2：出现红线

原因分析：连线错误发生短路

解决方案：检查错误电路，发现是连线被元器件覆盖时与其他部件错误连接导致的短路，故重新调整电路间距且修改短接线路。

2.2 多周期 MIPS 微程序 CPU 设计

（1） 设计思路

（1.1）总体设计思想

多周期 MIPS 微程序 CPU 特点：不再区分 IR 和 DR，而是在不同的时间用同一个器件，取指、译码、执行可以同时进行。使得整体数据通路的时钟周期变小、传输通路变短。同时需要在功能部件的输出端增加寄存器用来锁存数据，通过不同的锁存结构，将数据通路分为访存通路、运算通路、取操作数通路。

设计思路：整体的数据通路和单周期的类似，不同之处在于不再区分指令存储和数据存储，而是共用一个寄存器。同时新增寄存器 IR、DR、A、B、C。IR 用于锁存每个周期的指令，DR 用于锁存每个周期的数据，A、B 用于锁存寄存器的输出，C 用于锁存 ALU 的输出。

设计原理：同样通过控制器产生相应的控制信号。控制器需要时钟输入，因为多周期 CPU 的控制器需要涉及到时序逻辑。取指令需要两个时钟周期完成：第一个时钟周期负责将指令加载到 IR，同时 PC 更新为 PC+4，第二个时钟周期则完成对指令的译码、取操作数、和计算分支指令地址。对于不需要访存的 R 型指令，执行阶段通常需要两个时钟周期：在第一个时钟周期，根据对应的控制信号进行运算，将结果送入寄存器 C 中。在第二个时钟周期，将 C 的值写回到寄存器文件中。如果有访存操作，如 lw 指令，执行阶段通常需要三个时钟周期：第一个时钟周期计算得到操作数的地址，第二个时钟周期访问内存得到操作数并送到 DR 中，第三个时钟周期将 DR 中的操作数写入到寄存器。对于其他不同的指令，根据情况类推即可。

最终将数据通路与对应的控制逻辑连接即可完成设计。

（1.2）各种指令数据通路

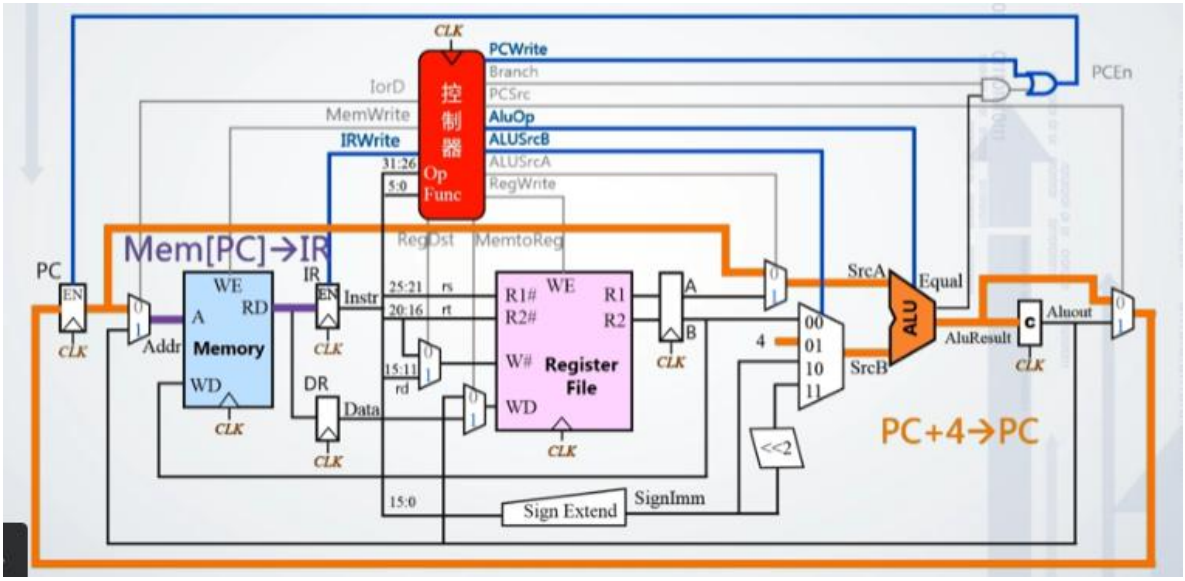


图 2-2-1 取指令数据通路图 1

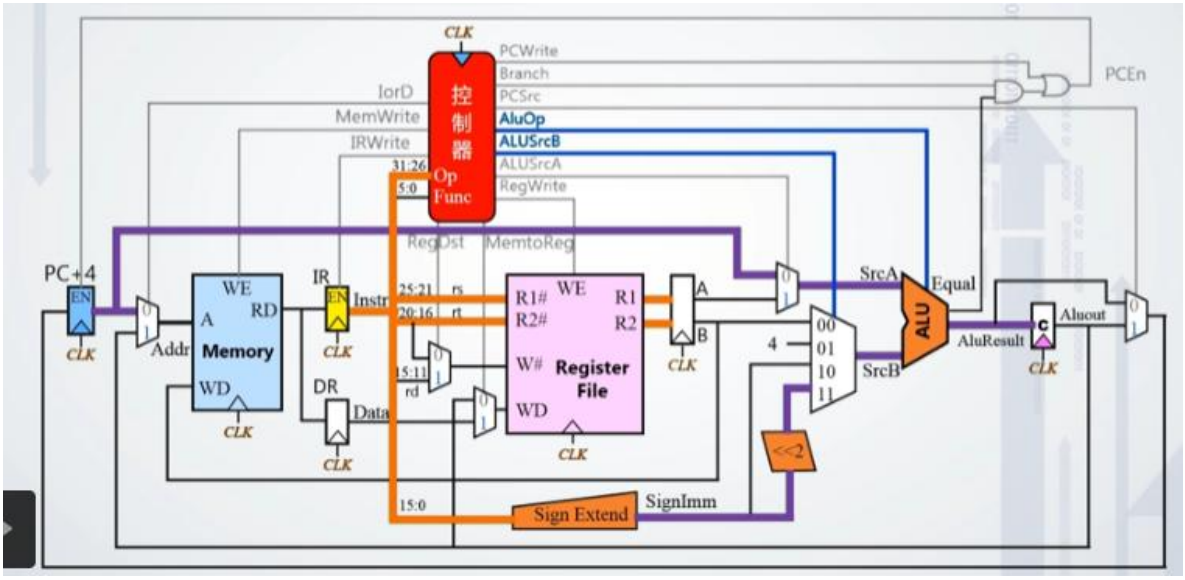


图 2-2-2 取指令数据通路图 2

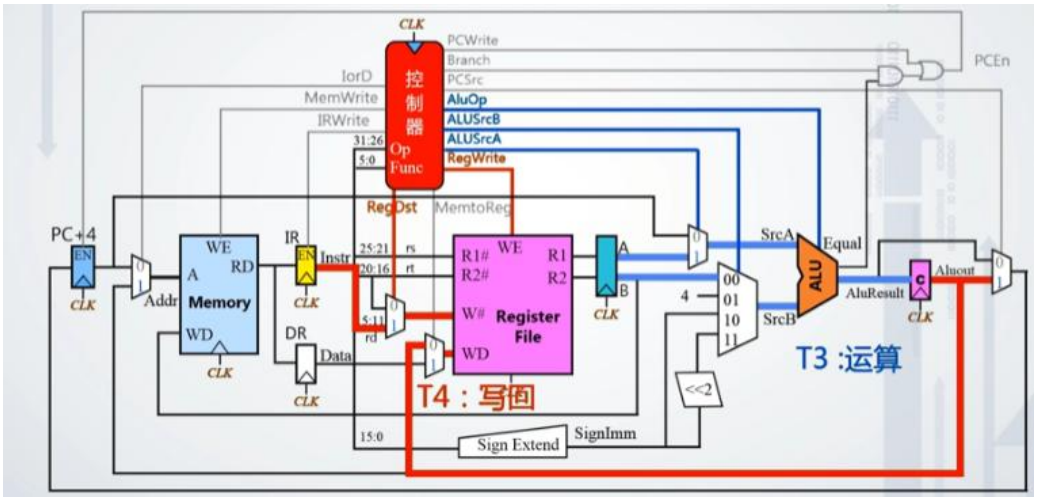


图 2-2-3 R 型指令数据通路图

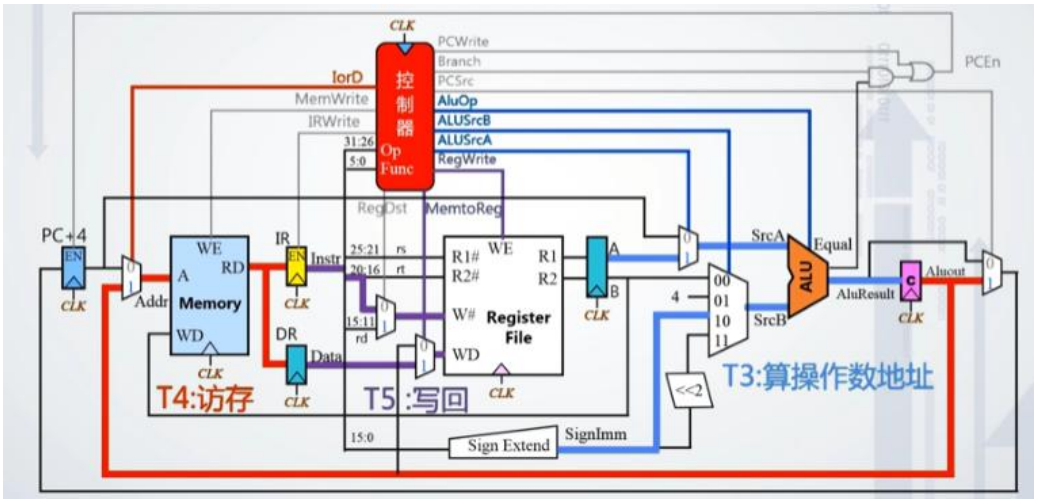


图 2-2-4 LW 指令数据通路图

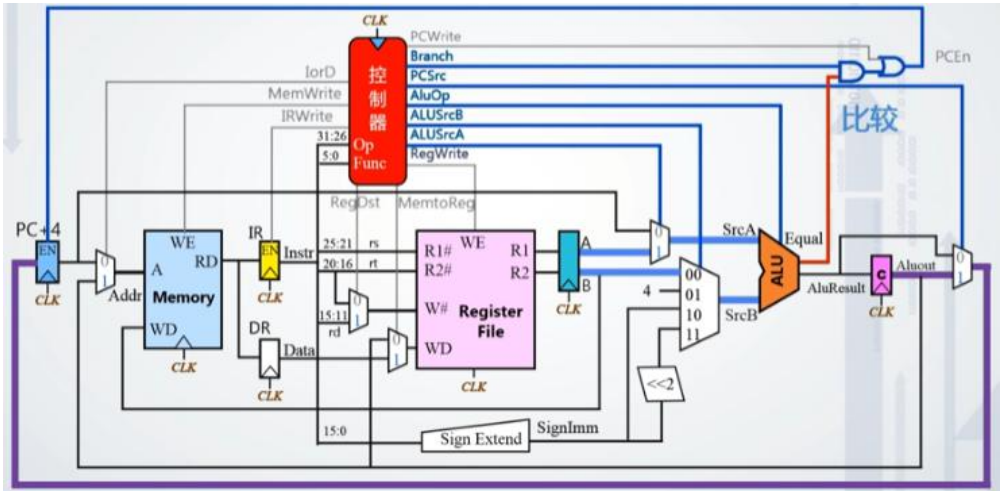


图 2-2-5 BEW 指令数据通路图

（1.3）多周期 MIPS 微程序控制器设计过程

整体设计思路为将指令译码信号输入到地址转移逻辑中，多路选择器根据微指令中的判断条件选择下址字段或微程序地址，输入到控制存储器中得到微指令，将微指令按位解析到对应的引脚，得到相应的微命令。其中指令译码部分与单周期完全相同，故不再赘述。

但 ALU 控制器逻辑与单周期不同，引入了 ALU_Control 进行选择，ALU_Control 为 00 时做加法，对应选择端输入加法的操作码，即 0x5，为 01 时做减法，对应选择端输入 0x6，为 10 时运算方式由 Func 决定对应选择端再连一个选择器，因为由 Func 决定时可能为加法或置位，将选择的结果送入 ALU_Control=10 端即可。

地址转移逻辑的实现首先需要确定各个微程序之间的状态转换关系。指令的状态转换图如图 2-2-6 所示：

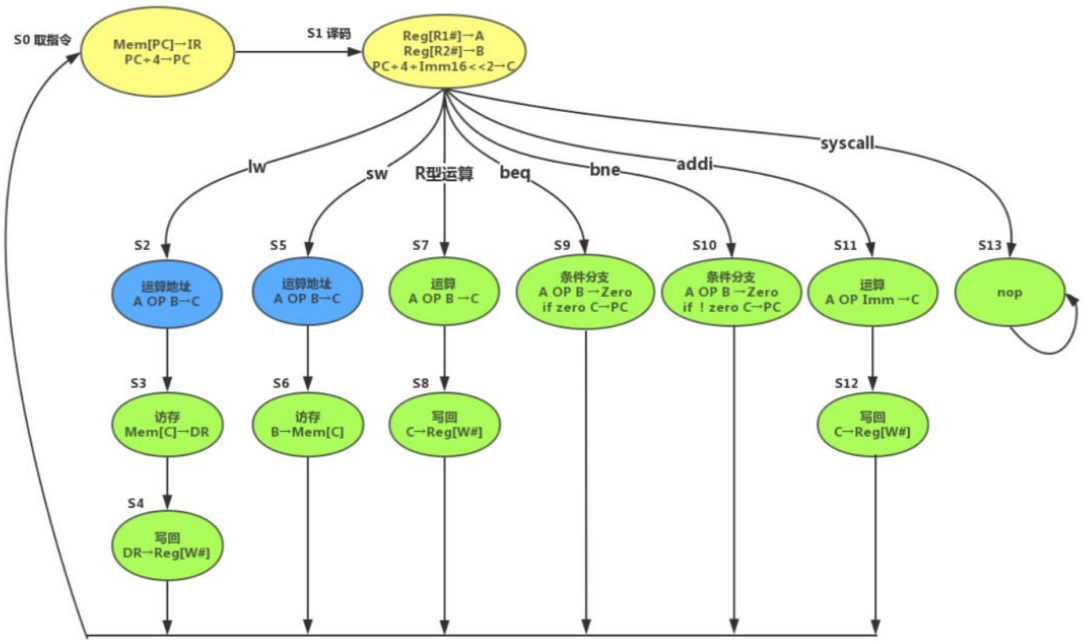


图 2-2-6 指令状态转换图

从图中可以看出 R_TYPE、ADDI、LW、SW、BEQ、BNE、SYSCALL 指令对应状态分别为 7、11、2、5、9、13，状态值即为微程序的入口地址在 Excel 中填入相关状态，如图 2-2-7 所示。由此得到逻辑表达式如图 2-2-8，然后使用 logisim 的分析组合逻辑电路自动生成电路。

机器指令译码信号							微程序入口地址				
R_Type	ADDI	LW	SW	BEQ	BNE	SYSCALL	入口地址10进制	S3	S2	S1	S0
1							7	0	1	1	1
	1						11	1	0	1	1
		1					2	0	0	1	0
			1				5	0	1	0	1
				1			9	1	0	0	1
					1		10	1	0	1	0
						1	13	1	1	0	1

图 2-2-7 微程序入口地址图

R_Type	ADDI	LW	SW	BEQ	BNE	SYSCALL	最小项表达式	S3	S2	S1	S0
R_Type&						~SYSCALL&	R_Type&~SYSCALL		R_Type&~SYSCALL+	R_Type&~SYSCALL+	R_Type&~SYSCALL+
	ADDI&						ADDI	ADDI+		ADDI+	ADDI+
		LW&					LW			LW+	
			SW&				SW		SW+		SW+
				BEQ&			BEQ	BEQ+			BEQ+
					BNE&		BNE	BNE+		BNE+	
R_Type&						SYSCALL&	R_Type&SYSCALL	R_Type&SYSCALL+	R_Type&SYSCALL+		R_Type&SYSCALL+

图 2-2-8 地址逻辑表达式生成图

之后根据所有微程序使用的微指令，设计控制存储器中的微指令.

各个控制信号与微指令的关系如图 2-2-9：

控制信号	功能	相关微指令或说明
PCWrite	控制PC写使能	取指令
IorD	选择访问内存是取指令还是取数据	LW2、SW2、SYSCALL
IRWrite	IR写使能，高电平有效	取指令
MemWrite	写入内存控制信号	SW2
MemRead	读内存控制信号	取指令、LW2
Beq	Beq指令译码信号	Beq
Bne	Bne指令译码信号	Bne
PcSrc	选择PC的输入，是顺序还是跳转	Beq、Bne
AluControl	控制ALU运算	R型运算1、Beq、Bne=10、SysCall=11
AluSrcA	选择ALU的第一个操作数，为0表示PC	LW1、SW1、R型运算1、Beq、Bne、ADDI1
AluSrcB	选择ALU的第二个操作数	00是寄存器、01是常量4、10是imm、11是imm<<2
RegWrite	寄存器写使能信号	LW3、R型运算2、ADDI2
RegDst	选择写入寄存器Rd还是Rt	R型写入指令为1、I型写入指令为0
MemToReg	选择写入寄存器的来自DR还是ALU的结	LW3

图 2-2-9 控制信号与微指令关系表

根据上表和微指令状态关系，可以填写如图 2-2-10 所示真值表，根据生成的逻辑表达式，最终实现了对应微指令控制逻辑。

微指令功能	状态	微指令地址	IorD	PcSrc	AluSrcA	AluSrcB	MemToReg	RegDst	IrWrite	PcWrite	RegWrite	MemWrite	MemRead	BEQ	BNE	AluControl	P	下址字段	微指令	十六进制
取指令	0	0000	0	0	0	01	0	0	1	1	0	0	1	0	0	00	0	0001	00001001100100000000	13201
译码	1	0001	0	0	0	11	0	0	0	0	0	0	0	0	0	00	1	0000	00011000000000010000	30010
LW1	2	0010	00	0	1	10	0	0	0	0	0	0	0	0	0	00	0	0011	00011000000000000011	60003
LW2	3	0011	1	0	0	00	0	0	0	0	0	0	1	0	0	00	0	0100	10000000001000000100	100204
LW3	4	0100	0	0	0	00	1	0	0	0	1	0	0	0	0	00	0	0000	00000100010000000000	8800
SW1	5	0101	0	0	1	10	0	0	0	0	0	0	0	0	0	00	0	0110	00110000000000000110	60006
SW2	6	0110	1	0	0	00	0	0	0	0	0	1	0	0	0	00	0	0000	10000000010000000000	100400
R型运算1	7	0111	0	0	1	00	0	0	0	0	0	0	0	0	0	10	0	1000	00100000000001001000	40048
R型运算2	8	1000	0	0	0	00	0	1	0	0	1	0	0	0	0	00	0	0000	00000010010000000000	4800
Beq	9	1001	0	1	1	00	0	0	0	0	0	0	0	1	0	10	0	0000	01100000000101000000	C0140
Bne	10	1010	0	1	1	00	0	0	0	0	0	0	0	0	1	10	0	0000	01100000000011000000	C00C0
ADDI1	11	1011	0	0	1	10	0	0	0	0	0	0	0	0	0	00	0	1100	00110000000000001100	6000C
ADDI2	12	1100	0	0	0	00	0	0	0	0	1	0	0	0	0	00	0	0000	00000000010000000000	800
SYSCALL	13	1101	1	0	0	00	0	0	0	0	0	0	0	0	0	11	0	1101	10000000000001101101	10006D

图 2-2-10 微指令生成图

（2） 电路图

（2.1） 微程序控制器电路图

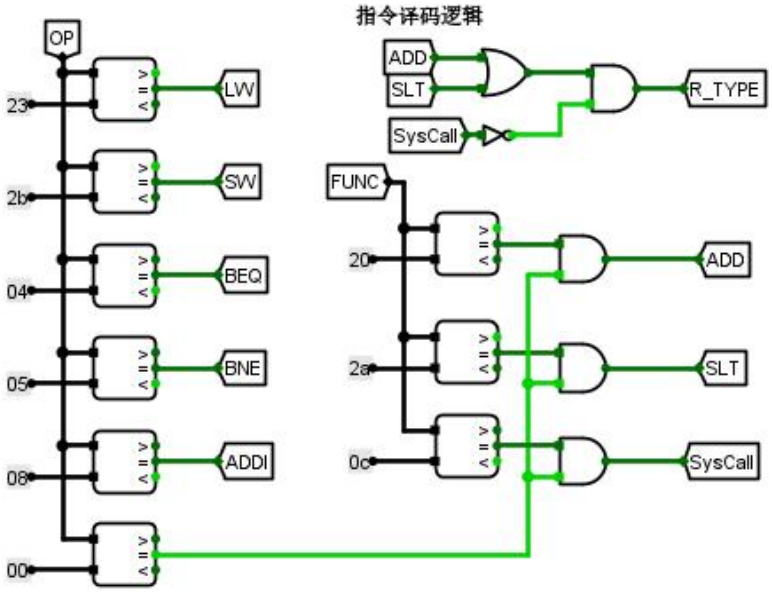


图 2-2-11 指令译码逻辑电路图

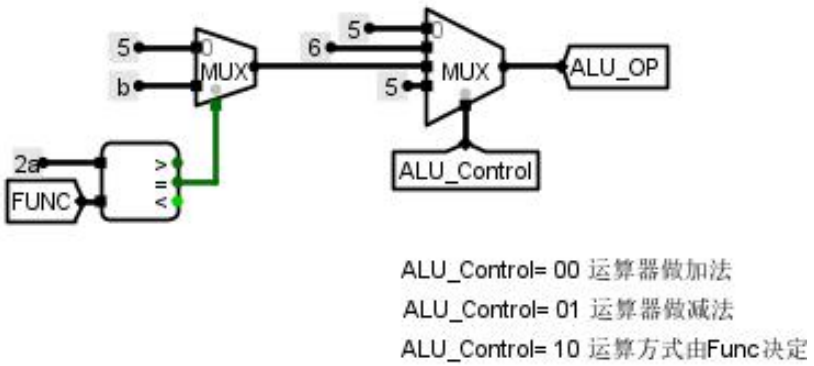


图 2-2-12 ALU 控制器电路图

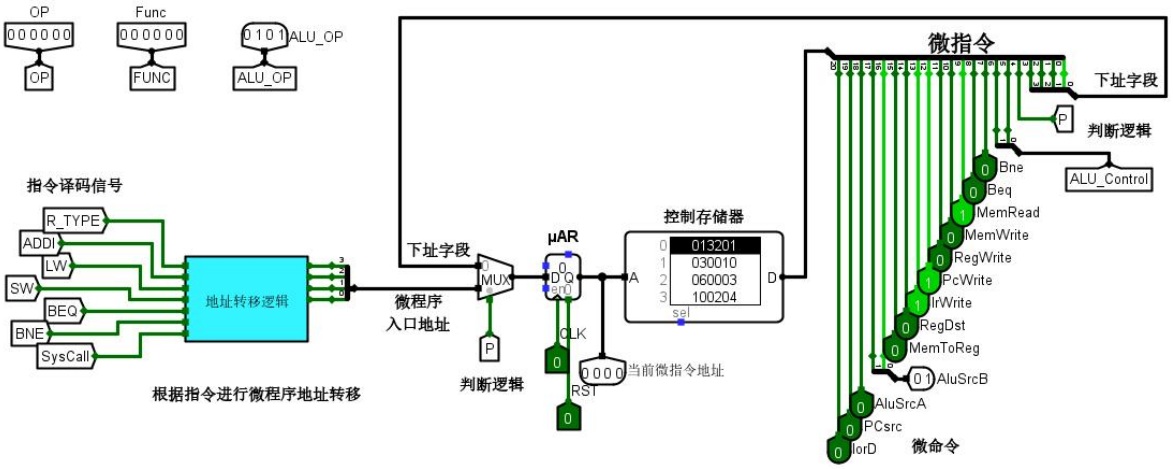


图 2-2-13 微指令生成电路图

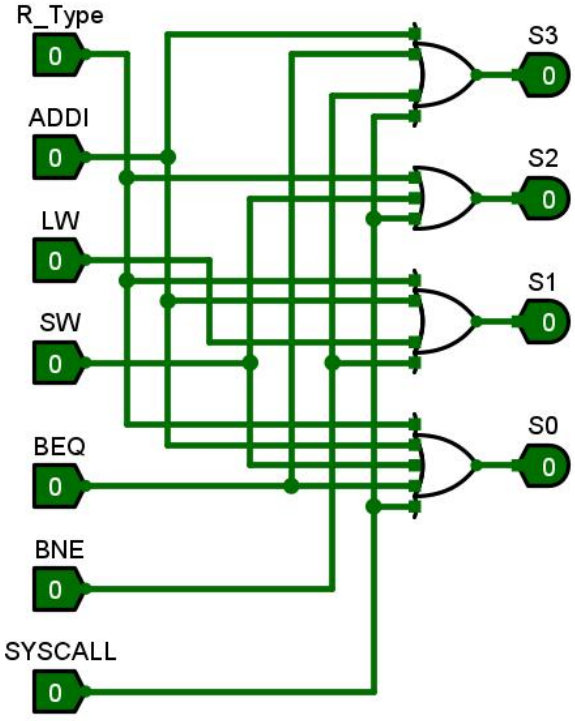


图 2-2-14 微程序地址转移逻辑自动生成图

（2.2） CPU 设计图

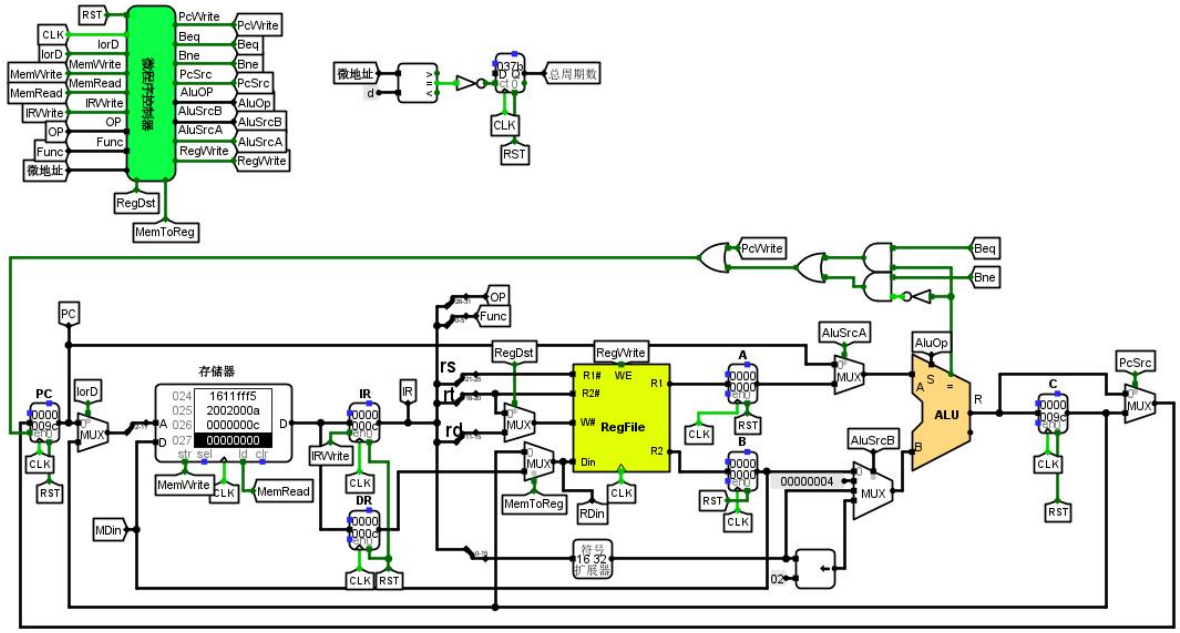


图 2-2-15 多周期 MIPS 微程序 CPU 设计图

（3）测试图

与单周期 MIPS CPU 一样，将 sort.hex 载入存储器，然后 ctrl+k 启动测试，在经过 891 个总周期后停止，如图 2-2-16。发现存储器中存储内容如图 2-2-17 所示，排序成功，本地电路测试通过。同时头歌平台测试通过如图 2-2-18 所示，说明设计成功。

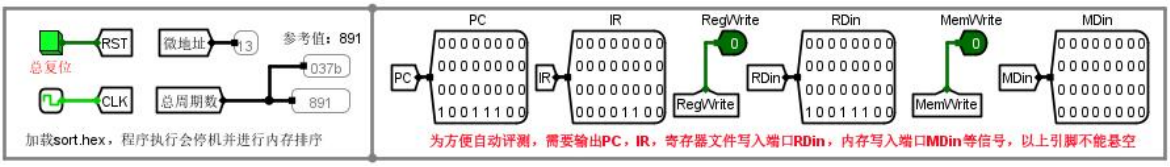


图 2-2-16 本地测试结果

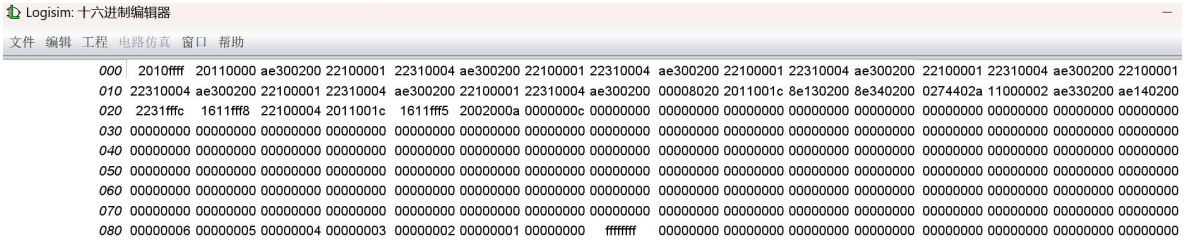


图 2-2-17 存储器结果

图 2-2-18 头歌测试结果

（4）故障与测试分析

(4.1)

故障现象 1：测试时生成的微指令不能得到正确的结果

原因分析：以为 SYSCALL 微指令的下址字段仍然为取指令指令的地址，但其实应该为 SYSCALL 本身，即下一状态仍为 SYSCALL，一直自旋。

解决方案：将 SYSCALL 的下址字段由 0000 修改为 1101。

(4.2)

故障现象 2：educoder 上输出 xxxx

原因分析：无意中改变了引脚

解决方案：由于电路已经完成且电路本身比较复杂，无法估计是哪一步改动了引脚，直接回退不太现实，故手动纠正引脚位置

3 总结与心得

3.1 实验总结

本门课实验主要完成了如下几点工作：

1） 完成方案总结

本次实验课由于时间比较紧张，我只完成了必做的八关内容而没有探索剩下的选做关卡。

在运算器实验中，可控加减法器和快速加法器主要利用了级联的方法完成了从较少的位数到较多位数器件的扩展。在 ALU 中，没有特殊要求的器件直接使用了软件中自带的器件。此外原码一位乘法电路需要自己引入计数器和停机信号等等，比较考验设计能力。

在存储系统实验中，主要利用分线器，将地址和数据分成特定的位数，完成了对内存按照字、半字、字节访问。对于每一块内存单元的使能信号的控制也比较复杂，可以先利用组合逻辑生成三种访问模式对应的信号，再用信号控制每一块内存区。

在 CPU 设计实验中，主要采取对指令译码，生成信号，利用信号控制数据通路的方案，多周期则是先译码成微指令，再从微指令中解析出相应的控制信号完成对电路的控制。虽然 CPU 的数据通路比较复杂，但在慕课的指导下整体框架不难得出，重点是对控制信号逻辑的理解与构建。

2） 实现的功能总结

实现了支持 13 种运算操作的 ALU，操作数位宽为 32 位，以及可以选择 32 位寄存器进行读写的寄存器文件，还有支持字访问、半字访问、字节访问三种模式的 32 位 RAM，最终实现可以完成简单冒泡排序的单周期、多周期 CPU。

3） 其他需要总结的内容

在排查错误方面，由于本学期有数字逻辑电路这门课作为本实验的前置内容，对 logisim 基本熟悉的情况下再开展计组实验难度减少了一些，我对 logisim 电路设计和排查错误也更加熟练了，很好地锻炼了自己的能力。

3.2 实验心得

本次实验让我利用 logisim，完成了运算器、存储系统与 CPU 设计三个实验，动手实现了理论课中学习到的很多硬件，而不只是停留在理论层面。

在理论课学习时，对于各种硬件理解的并不是很透彻，尤其是 CPU 的执行过程。实验帮助我深入了解硬件的工作原理，让我充分理解 CPU 是如何一步步装配起来并通过硬件逻辑执行指令的。

除此之外，本次实验让我对于计算机底层有了更加深入的认识，并进一步拉近了我与硬件的距离。实验中的电路设计与连线过程也是对我的极大考验，需要我对电路整体把握足够且拥有细心和耐心，往往一个微小的错误就会导致整个电路的失败，而这种错误的排查往往十分耗时且令人沮丧。但当我亲自完成一个个小型电路并测试成功时，看着时钟信号不断闪烁和明灭的电路顺利工作，不由得心生感叹硬件设计的精妙，也让我体会到计算机底层实现的一点乐趣。也许理论学习是枯燥乏味且困难的，但当自己动手实践完成了硬件逻辑电路，萌生出的不仅仅有成就感也有对计算机这门科学的肃然起敬。

本次实验的重点突出，指导性好，可以避免学生将精力花费在没有必要的地方。在构建微指令时，利用老师提供好的 Excel 就可以很方便地生成逻辑表达式和具体的微指令，节省时间的同时，又没有减弱实验的效果，因为理解状态转换关系才是重点。不过我认为本实验的资料包有些杂乱，各种指导信息分布在各个角落，很难让人一开始就对整个实验有很好的总体把握，建议整合实验资料更新实验指导书的内容，而不是简单地告知需要完成哪些关卡，即使可以通过慕课获取一些信息，但我觉得这样的吸收方式有点割裂，头歌平台的说明也不够详细。

总之这次实验让我获益匪浅，对我以后的学习有很大的帮助，感谢这门课程的用心设计和各位老师的辛苦付出，也感谢在实验过程中给予我帮助的同学们！

4 参考文献

[1] 谭志虎，秦磊华，胡迪青著. 计算机组成原理实践教程——从逻辑门到 CPU. 北京：清华大学出版社，2018.

[2] 谭志虎主编. 计算机组成原理（微课版）. 人民邮电出版社，2021

[3] 计算机硬件系统设计_华中科技大学_中国大学 MOOC(慕课)
https://www.icourse163.org/course/HUST-1205809816

原创性声明

本人郑重声明本报告内容，是由作者本人独立完成的。有关观点、方法、数据和文献等的引用已在文中指出。除文中已注明引用的内容外，本报告不包含任何其他个人或集体已经公开发表的作品成果，不存在剽窃、抄袭行为。

特此声明！

作者签字： 邬雪菲