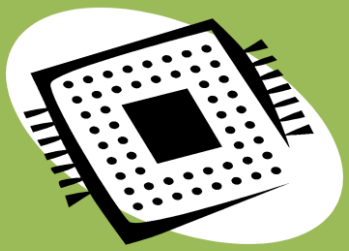
华中科技大学

2019

计算机组成原理 ·实验报告·

专　　业：	计算机科学与技术
班　　级：	ACM1701
学　　号：	U201714780
姓　　名：	刘晨彦
电　　话：	15927172332
邮　　件：	Chenyanliu712@qq.com
完成日期：	2019-12-12



计算机科学与技术学院

目　录

1 CPU 设计实验

1.1 设计要求

1.1.1 单周期 MIPS 硬布线 CPU 的设计要求

1. 单周期 MIPS 硬布线 CPU 通路的设计要求

利用 Logisim 平台中现有运算部件构建一个单周期硬布线的 CPU，支持八种基本的 MIPS 指令，能够实现内存区域的冒泡排序。单周期 MIPS 硬布线 CPU 通路的输入输出引脚见下表。

表 1.1 单周期 MIPS 硬布线电路引脚与功能描述

引脚	输入/输出	位宽	功能描述
RST	输入	1	复位信号
CLK	输入	1	时钟信号
PC	输出	32	地址
IR	输出	32	指令
RegWrite	输出	32	寄存器写使能控制信号
RDin	输出	32	写入寄存器的内容
MemWrite	输出	1	写内存控制信号
MDin	输出	32	写入随机存储器的内容

2. 单周期硬布线控制器的设计要求

为了实现电路的控制，需要利用 Logisim 平台设计一单周期硬布线控制器，能够根据输入的指令信号输出对应的控制信号。单周期硬布线控制器电路的输入输出引脚见下表。

表 1.2 单周期硬布线控制器电路引脚与功能描述

引脚	输入/输出	位宽	功能描述

OP	输入	6	操作码
Func	输入	6	功能码
ALU_OP	输出	4	运算器操作控制符，5H 为加法，BH 为比较
RegDst	输出	1	写入寄存器选择控制信号
RegWrite	输出	1	寄存器写使能控制信号
MenToReg	输出	1	写入寄存器的数据来自存储器
MemWrite	输出	1	写内存控制信号
AluSrc	输出	1	运算器第二输入选择
Beq	输出	1	Beq 指令译码信号
Bne	输出	1	Bne 指令译码信号
Halt	输出	1	停机信号

1.1.2 多周期 MIPS 微程序 CPU 的设计要求

1. 多周期 MIPS 微程序 CPU 通路的设计要求

利用 Logisim 平台中现有运算部件构建一个多周期微程序 CPU，支持八种基本的 MIPS 指令，能够实现简单的冒泡排序。多周期微程序 CPU 电路的输入输出引脚见下表。

表 1.3 多周期微程序 CPU 电路引脚与功能描述

引脚	输入/输出	位宽	功能描述
RST	输入	1	复位信号
CLK	输入	1	时钟信号
PC	输出	32	地址
IR	输出	32	指令
RegWrite	输出	32	寄存器写使能控制信号
RDin	输出	32	写入寄存器的内容
MemWrite	输出	1	写内存控制信号

MDin	输出	32	写入随机存储器的内容

2. 多周期微程序控制器的设计要求

为了实现电路的控制，需要设计多周期微程序控制器，用于对应指令下提供对应的控制指令。控制器的框架如下图所示：

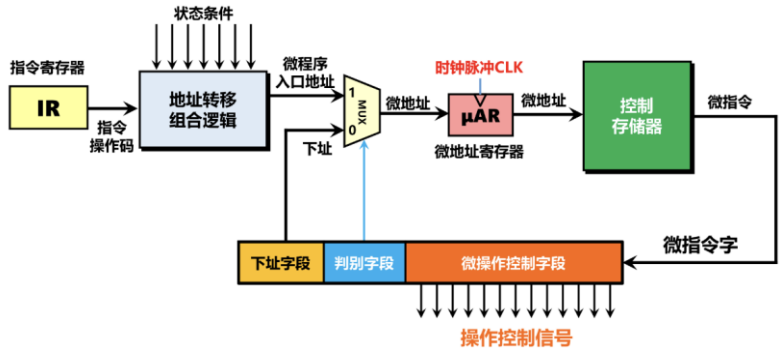


图 1.1 微程序控制器原理框架

当指令输入微程序控制器时，首先通过地址转移的组合逻辑电路给出该指令首个时钟周期内的控制指令所在的地址，利用该地址从微程序控制器中取得控制指令并输出控制信号，该指令若不是单周期的指令，则从控制指令中给出下一个时钟周期的控制指令所在地址，重复上述过程直至一条指令执行结束，回到微地址寄存器首行进行取指令和译码，处理下一条指令。

电路的输入输出引脚见下表：

表 1.4 多周期微程序 CPU 控制器电路引脚与功能描述

引脚	输入/输出	位宽	功能描述
OP	输入	6	操作码
Func	输入	6	功能码
PCWrite	输出	1	PC 写使能控制信号
LorD	输出	1	指令或数据信号：0 表示指令，1 表示数据
IRwrite	输出	1	指令寄存器写使能
MemWrite	输出	1	写内存控制信号

MemRead	输出	1	读内存控制信号
Beq	输出	1	Beq 指令译码信号
Bne	输出	1	Bne 指令译码信号
PcSrc	输出	1	PC 输入来源信号
AluOP	输出	4	运算器操作控制符
AluSrcA	输出	1	运算器第一输入选择信号
AluSrcB	输出	2	运算器第二输入选择信号
RegWrite	输出	1	寄存器写使能控制信号
RegDst	输出	1	写入寄存器选择控制信号
MenToReg	输出	1	写入寄存器的数据来自存储器

3. 微程序地址转移逻辑电路的设计要求

地址转移逻辑则需要根据输入的指令，通过组合逻辑电路产生对应的控制信号的首地址。微程序地址转移逻辑电路的输入输出引脚如下表所示：

表 1.5 多周期微程序 CPU 地址转移逻辑电路引脚与功能描述

引脚	输入/输出	位宽	功能描述
RTYPE	输入	1	1：指令为 RTYPE，0：指令不是 RTYPE
ADDI	输入	1	1：指令为 ADDI，0：指令不是 ADDI
LW	输入	1	1：指令为 LW，0：指令不是 LW
SW	输入	1	1：指令为 SW，0：指令不是 SW
BEQ	输入	1	1：指令为 BEQ，0：指令不是 BEQ
BNQ	输入	1	1：指令为 BNQ，0：指令不是 BNQ
SYSCALL	输入	1	1：指令为 SYSCALL，0：指令不是 SYSCALL
S1	输出	1	控制信号地址第 0 位
S2	输出	1	控制信号地址第 1 位
S3	输出	1	控制信号地址第 2 位
S4	输出	1	控制信号地址第 3 位

1.1.3 多周期 MIPS 硬布线 CPU 的设计要求

1. 多周期 MIPS 硬布线 CPU 通路的设计要求

利用 Logisim 平台中现有运算部件构建一个多周期硬布线 CPU，支持八种基本的 MIPS 指令，能够实现简单的冒泡排序。多周期硬布线 CPU 电路的输入输出引脚见下表。

表 1.6 多周期硬布线 CPU 电路引脚与功能描述

引脚	输入/输出	位宽	功能描述
RST	输入	1	复位信号
CLK	输入	1	时钟信号
PC	输出	32	地址
IR	输出	32	指令
RegWrite	输出	32	寄存器写使能控制信号
RDin	输出	32	写入寄存器的内容
MemWrite	输出	1	写内存控制信号
MDin	输出	32	写入随机存储器的内容

2. 多周期硬布线控制器的设计要求

为了实现电路的控制，需要设计多周期微程序控制器，根据上一条控制指令的地址和指令译码信号给出下一条控制指令的地址，根据下一条指令的地址在控制器中找到对应的控制指令并输出。电路的输入输出引脚与表 1.4 一致，此处不再复述。

3. 多周期硬布线有限状态机的设计要求

设计一个组合逻辑电路，以现态和指令译码信号为输入，下一条控制指令地址为次态和输出。电路的输入输出引脚如下表。

表 1.7 多周期硬布线有限状态机电路引脚与功能描述

引脚	输入/输出	位宽	功能描述
S0	输入	1	现态控制指令地址第 0 位

S1	输入	1	现态控制指令地址第 1 位
S2	输入	1	现态控制指令地址第 2 位
S3	输入	1	现态控制指令地址第 3 位
RTYPE	输入	1	1：指令为 RTYPE，0：指令不为 RTYPE
ADDI	输入	1	1：指令为 ADDI，0：指令不为 ADDI
LW	输入	1	1：指令为 LW，0：指令不为 LW
SW	输入	1	1：指令为 SW，0：指令不为 SW
BEQ	输入	1	1：指令为 BEQ，0：指令不为 BEQ
BNE	输入	1	1：指令为 BNE，0：指令不为 BNE
SYSCALL	输入	1	1：指令为 SYSCALL，0：指令不为 SYSCALL
N0	输出	1	次态控制指令地址第 0 位
N1	输出	1	次态控制指令地址第 1 位
N2	输出	1	次态控制指令地址第 2 位
N3	输出	1	次态控制指令地址第 3 位

1.2 方案设计

1.2.1 单周期 MIPS 硬布线 CPU 的方案设计

1. 单周期 MIPS 硬布线 CPU 通路设计

首先根据提供的单周期 CPU 的参考通路，绘制单周期 MIPS 硬布线 CPU 电路。如下图所示。

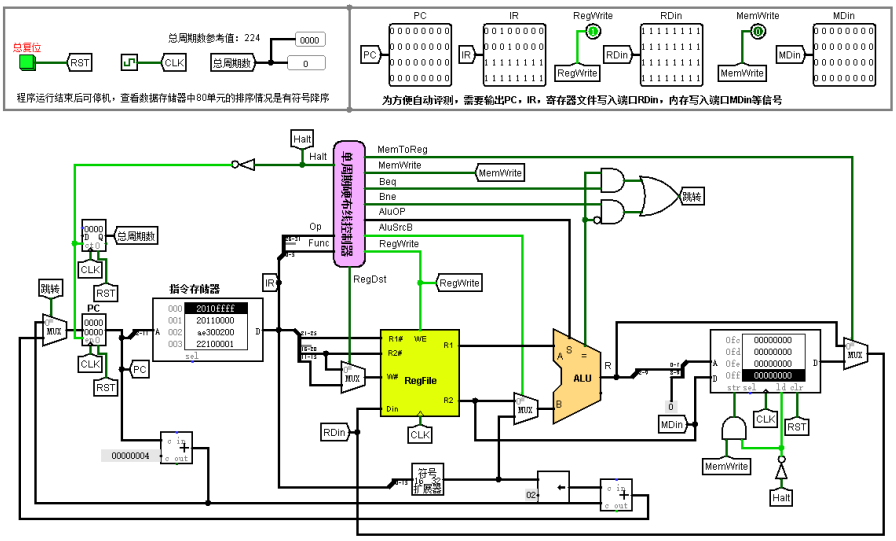


图 1.2 单周期 MIPS 硬布线 CPU 电路图

2. 单周期硬布线控制器设计

在指令译码逻辑电路中，可以查找 MIPS32 指令手册可知指令对应的操作码和功能码，如下表所示。其中指令 ADD 和 SLT 为 R 型指令。

表 1.8 指令与对应操作码和功能码表

指令	操作码	功能码
LW	23H	NULL
SW	2BH	NULL
BEQ	04H	NULL
BNE	05H	NULL
ADDI	08H	NULL
ADD	00H	20h
SLT	00H	2AH
SYSCALL	00H	0CH

在 ALU 控制逻辑中，当且仅当指令为 SLT 时，ALU 控制器输出比较信号 0BH，其余情况均输出加法信号 05H。

在控制信号电路中，根据各指令的功能，输出对应的控制信号。LW 指令需要写入寄存器、将存储器的数据写入寄存器、将立即数位扩展后和寄存器 R1 的输出相加。故需要 RegWrite、MenToReg、AluSrc 信号为 1。SW 指令需要写入存储器、将立即数位扩展后和寄存器 R1 的输出相加，故需要 MemWrite、AluSrc 信号为 1。BEQ 指令只需要 Beq 信号为 1，BNE 指令只需要 Bne 信号为 1。ADDI 指令需要写入寄存器、将立即数位扩展后和寄存器 R1 的输出相加，故需要 RegWrite、AluSrc 信号为 1。R 型指令则需要指令的第 11~15 位作为寄存器的写入地址、写入寄存器，故需要 RegDst、RegWrite 信号为 1。最后 SYSCALL 指令则只需要停机信号 Halt 为 1 即可。

根据以上控制器的设计思路，绘制电路图如下所示：

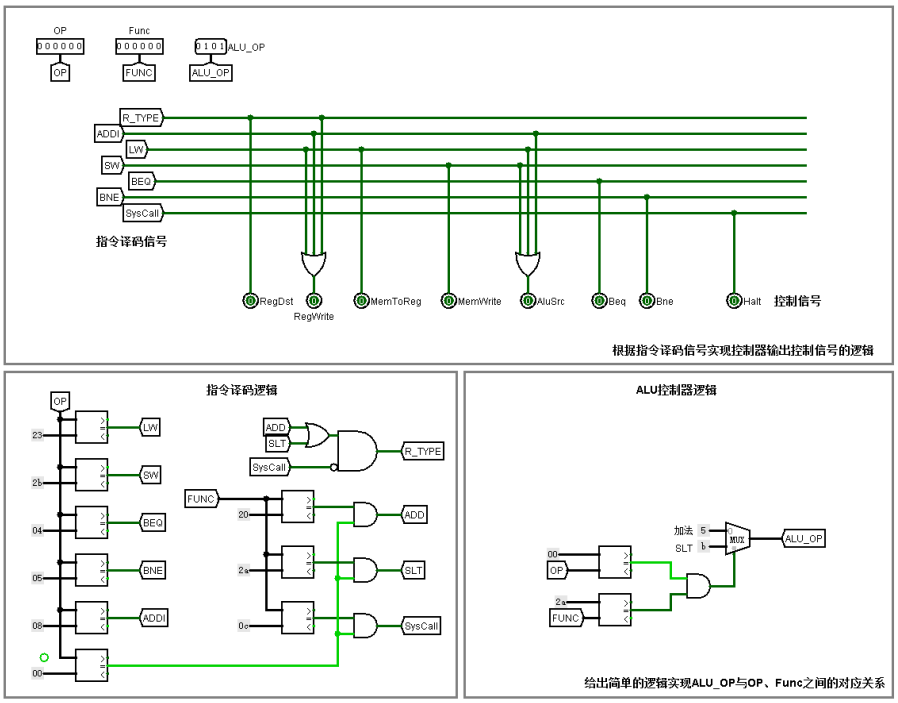


图 1.3 单周期硬布线控制器电路

1.2.2 多周期 MIPS 微程序 CPU 的方案设计

1. 多周期 MIPS 微程序 CPU 通路的设计

首先根据提供的多周期 CPU 的参考通路，绘制多周期微程序 CPU 电路。如下图所示。

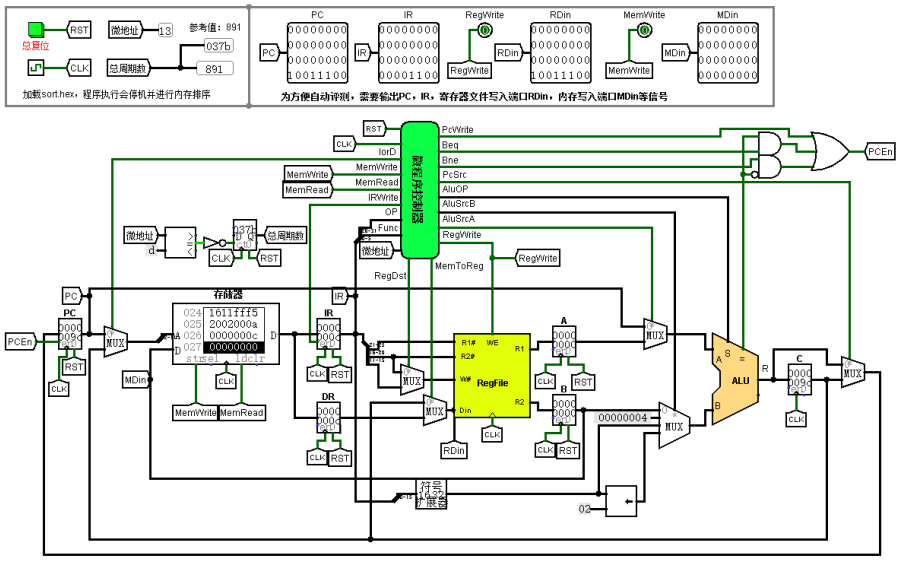


图 1.4 多周期微程序 CPU 电路

2. 微程序控制器电路的设计

指令译码逻辑电路和单周期硬布线控制器中的一致，此处不再赘述。

在 ALU 控制器逻辑中，当 ALU_Control 为 00 时运算器做加法，为 01 时做减法，为 10 时由 FUNC 决定，此时仅当 FUNC 为 2AH 时执行比较操作。

在控制存储器中需要填写对应的控制信号，控制信号与指令阶段的对应关系如下表：

表 1.9 指令阶段与对应信号量

指令阶段	操作流程
取指令	(MEM[PC])-> IR

	(PC)+4->PC
译码	(R[IR[25:21]])->A
	(R[IR[20:16]])->B
	(PC)+(S-EXT(IR[15:0])<<2) ->C
LW1	(A)+ S-EXT(IR[15:0]) ->C
LW2	(MEM[PC]) ->DR
LW3	(DR)-> R[IR[20:16]]
SW1	(A)+ S-EXT(IR[15:0]) ->C
SW2	(MEM[PC])->DR
RTYPE1	(A)+(B)| ((A)<(B))->C
RTYPE2	(C)-> R[IR[15:11]]
BEQ	If(A==B) (C) -> PC
BNE	If(A!=B) (C)-> PC
ADDI1	(A)+ S-EXT(IR[15:0]) ->C
ADDI2	(C) -> R[IR[20:16]]
SYSCALL	空操作

根据上表，可在提供的微指令自动生成表中填写相对应的控制信号，填写情况如图所示：

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	00001001100100000001	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	0	0	1	10	0	0	0	0	0	0	0	0	0	00	0	0011	00110000000000000011	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_Type1	7	0111	0	0	1	00	0	0	0	0	0	0	0	0	0	10	0	1000	00100000000001001000	40048
R_Type2	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	00	0	0000	01100000000010000000	C0100
BNE	10	1010	0	1	1	00	0	0	0	0	0	0	0	0	1	00	0	0000	01100000000010000000	C0080
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 1.5 微指令自动生成表

将十六进制结果复制黏贴至微程序控制器即可。根据以上设计，得到多周期微指令控制器如下图所示：

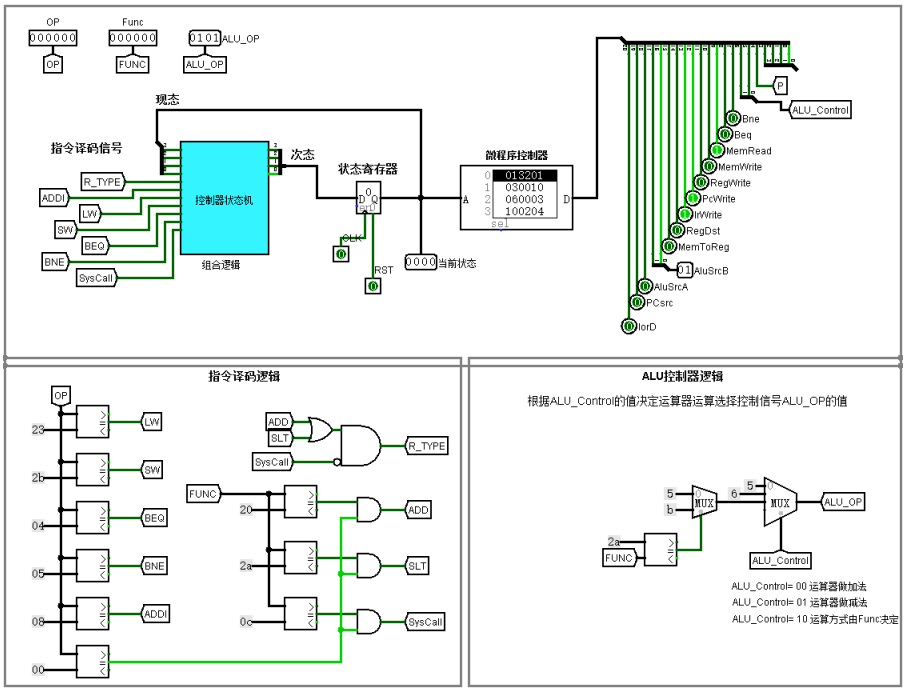


图 1.6 微程序控制器电路

3. 微程序地址转移逻辑电路设计

根据微指令自动生成表，可以得到每条指令的首地址。将每条指令首地址以十进制形式输入微地址转移逻辑自动生成表，可以得到转移逻辑电路的组合逻辑。如下图和下表所示：

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 1.7 微程序地址入口表截图

表 1.10 位号与对应地址组合逻辑表

位号	组合逻辑
S0	R_Type + ADDI + SW + BEQ + SYSCALL
S1	R_Type + ADDI + LW + BNE
S2	R_Type + SW + SYSCALL
S3	ADDI + BEQ + BNE + SYSCALL

将对应组合逻辑复制粘贴至微程序地址转移逻辑电路中自动生成电路，如下图所示：

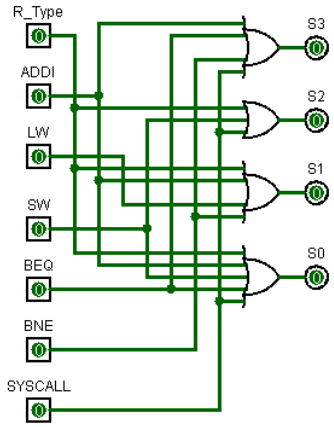


图 1.8 微程序地址转移逻辑电路

1.2.3 多周期 MIPS 硬布线 CPU 的方案设计

1. 多周期 MIPS 硬布线 CPU 通路的方案设计

首先根据提供的多周期 CPU 的参考通路，绘制多周期硬布线 CPU 电路。如下图所示。

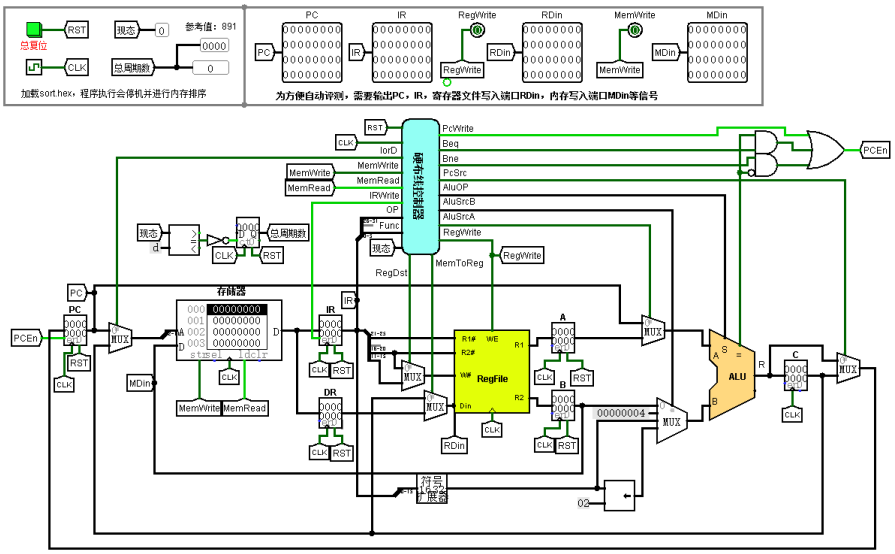


图 1.9 多周期硬布线 CPU 通路

2. 多周期硬布线控制器的方案设计

指令译码逻辑电路和单周期硬布线控制器中的一致，此处不再赘述。

ALU 控制器逻辑中和多周期微程序控制器中的一致，此处同样不再赘述。

故多周期硬布线控制器的设计如下图。

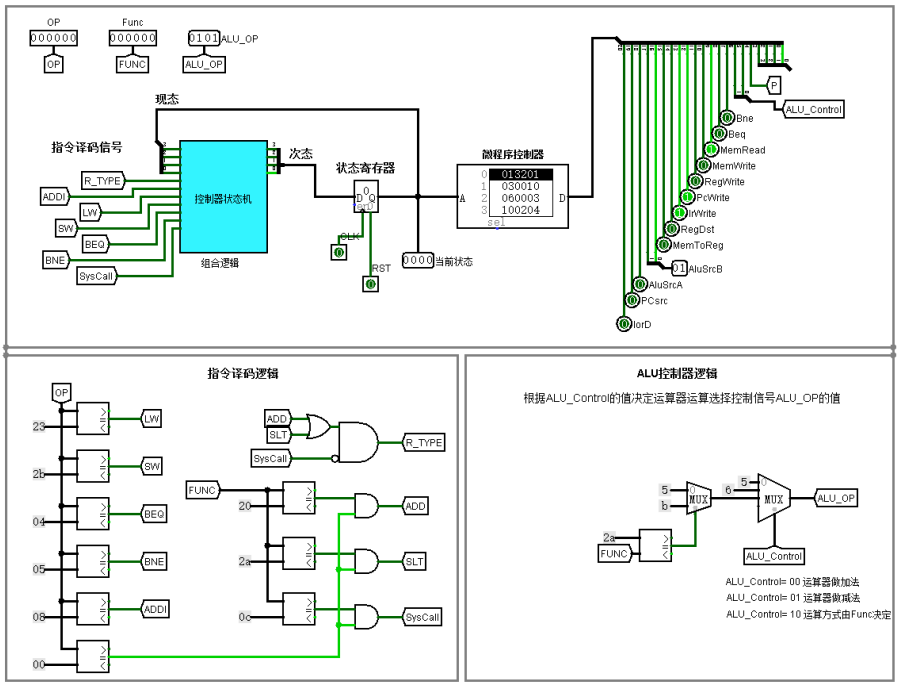


图 1. 10 多周期硬布线控制器

3. 多周期硬布线有限状态机的方案设计

根据图 1.5 的微指令自动生成表可知各控制指令所在的地址位置，将现态、指令译码信号和对应次态控制指令的地址，如下图所示。

当前状态(现态)					输入信号							下一状态（次态）				
S3	S2	S1	S0	现态10进制	R_Type	LW	SW	BEQ	BNE	SYSCALL	ADDI	次态10进制	N3	N2	N1	N0
0	0	0	0	0								1	0	0	0	1
0	0	0	1	1	1							7	0	1	1	1
0	0	0	1	1		1						2	0	0	1	0
0	0	0	1	1			1					5	0	1	0	1
0	0	0	1	1				1				9	1	0	0	1
0	0	0	1	1					1			10	1	0	1	0
0	0	0	1	1						1		13	1	1	0	1
0	0	0	1	1							1	11	1	0	1	1
0	0	1	0	2								3	0	0	1	1
0	0	1	1	3								4	0	1	0	0
0	1	0	0	4								0	0	0	0	0
0	1	0	1	5								6	0	1	1	0
0	1	1	0	6								0	0	0	0	0
0	1	1	1	7								8	1	0	0	0
1	0	0	0	8								0	0	0	0	0
1	0	0	1	9								0	0	0	0	0
1	0	1	0	10								0	0	0	0	0
1	0	1	1	11								12	1	1	0	0
1	1	0	0	12								0	0	0	0	0
1	1	0	1	13								13	1	1	0	1

图 1.1 硬布线状态转移表截图

则自动生成的地址位的逻辑表达式见下表。

表 1.11 地址位号与对应组合逻辑表

位号	组合逻辑
N0	~S3&~S2&~S1&~S0+~S3&~S2&~S1&S0&R_Type+~S3&~S2&~S1&S0&SW+~S3&~S2&~S1&S0&BEQ+~S3&~S2&~S1&S0&SYSCALL+~S3&~S2&~S1&S0&ADDI+~S3&~S2&S1&~S0+S3&S2&~S1&S0
N1	~S3&~S2&~S1&S0&R_Type+~S3&~S2&~S1&S0&LW+~S3&~S2&~S1&S0&BNE+~S3&~S2&~S1&S0&ADDI+~S3&~S2&S1&~S0+~S3&S2&~S1&S0
N2	~S3&~S2&~S1&S0&R_Type+~S3&~S2&~S1&S0&SW+~S3&~S2&~S1&S0&SYSCALL+~S3&~S2&S1&S0+~S3&S2&~S1&S0+S3&~S2&S1&S0+S3&S2&~S1&S0

N3	~S3&~S2&~S1&S0&BEQ+~S3&~S2&~S1&S0&BNE+~S3&~S2&~S1&S0&SYSCALL+~S3&~S2&~S1&S0&ADDI+~S3&S2&S1&S0+S3&~S2&S1&S0+S3&S2&~S1&S0

将对应组合逻辑复制粘贴至硬布线地址转移逻辑电路中自动生成电路，如下图所示：

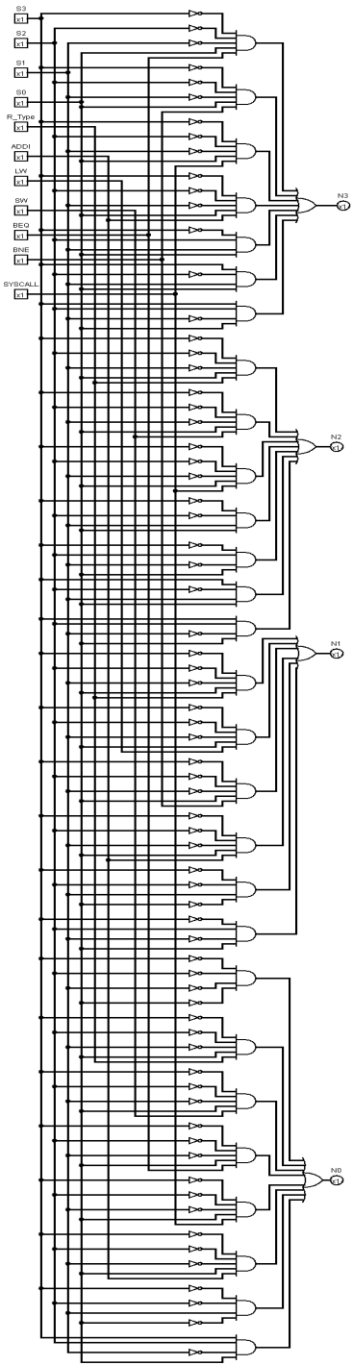

S3
S2
S1
S0
R_Type
ADDI
LW
SW
BEQ
BNE
SYSCALL
N3
N2
N1
N0

图 1.12 硬布线状态机电路

1.3 实验步骤

（1） 观看 MOOC，了解实验的操作步骤
（2） 按照给出的 CPU 通路绘制单周期硬布线 CPU 通路
（3） 根据对应指令译码信号设计单周期硬布线控制器
（4） 按照给出的 CPU 通路绘制多周期微程序 CPU 通路
（5） 在微指令自动生成表中填入指令对应的控制信号，将控制指令输入控制存储器，完成控制器的绘制
（6） 将对应控制指令入口地址填入微程序地址转移逻辑自动生成表，得到地址位的逻辑表达式，在 Logisim 中自动生成电路
（7） 按照给出的 CPU 通路绘制多周期硬布线 CPU 通路
（8） 完成控制器的绘制
（9） 将对应现态、指令译码信号和次态填入微程序地址转移逻辑自动生成表，得到地址位的逻辑表达式，在 Logisim 中自动生成电路
（10）测试三个 CPU 的结果，修改错误直至正确输出

1.4 故障与调试

1.4.1 控制器指令译码逻辑问题

故障现象： 单周期 MIPS 硬布线 CPU 运行无法结束，在地址 01AH~021H 之前循环但不结束。

原因分析： 由于 PC 一直在地址 01AH~021H 中循环而没有结束，问题可能存在于数据通路和指令译码逻辑电路中。

解决方案：检查 CPU 通路发现与参考通路一致，没有问题。检查指令译码逻辑电路，发现 RTYPE 类型的比较器的值为 04H，与 MIPS 手册上的不符，修改为 00H 后测试通过，故障产生的原因是在复制比较常量时忘记修改常数。

1.4.2 单周期数据存储器地址错误

故障现象：冒泡排序 sort.hex 程序执行完毕后，没有在指定的 80 号存储单元形成降序排列的数据，而是在 200 号存储单元形成降序排列数据，而且两个数据地址之差为 4。

```
200  00000006 00000000 00000000 00000000  00000005 00000000 00000000 00000000
208  00000004 00000000 00000000 00000000  00000003 00000000 00000000 00000000
210  00000002 00000000 00000000 00000000  00000001 00000000 00000000 00000000
218  00000000 00000000 00000000 00000000  ffffffff 00000000 00000000 00000000
```

图 1.13 存储地址错误

原因分析：Logisim 中 RAM 只支持一种访问模式，一次访问读出 32 位数据，直接给出字地址使得内存布局错误，如下图：

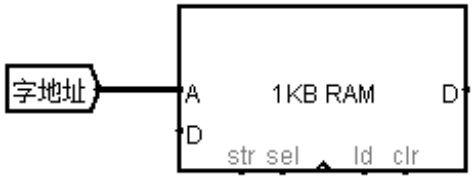


图 1.14 直接字地址访问

解决方案：字地址除以 4 即可得到正确的内存布局，即高两位补零作为字节地址送入存储器地址即可，如下图：

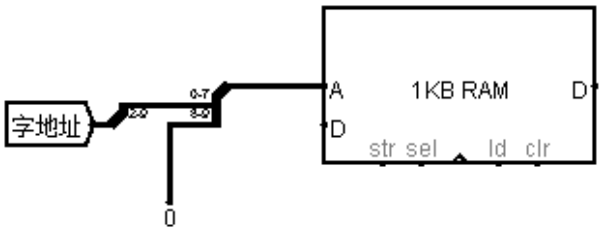


图 1.15 字节地址访问

1.5 测试与分析

1.5.1 单周期硬布线 CPU 执行

加载镜像 sort.hex，启动时钟模拟。运行周期数如下图所示：

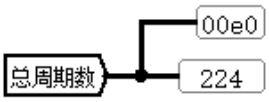


图 1.16 单周期硬布线 CPU 运行周期数

与参考周期数一致。查看内存，内存情况如下图所示：

```
080  00000006 00000005 00000004 00000003  00000002 00000001 00000000 ffffffff
```

图 1.17 内存第 80 号单元开始的存储情况

可知排序的结果正确。将文件上载至 Educoder 平台运行，运行结果如下图所示：

图 1.18 平台测试结果

综上所述，单周期硬布线 CPU 设计正确

1.5.2 多周期微程序 CPU 执行

加载镜像 sort.hex，启动时钟模拟。运行周期数如下图所示：

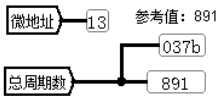


图 1.19 多周期微程序 CPU 运行周期数

与参考周期数一致。查看存储器，存储器情况如下图所示：

```
080  00000006 00000005 00000004 00000003  00000002 00000001 00000000 ffffffff
```

图 1.20 存储器第 80 号单元开始的存储情况

可知排序的结果正确。将文件上载至 Educoder 平台运行，运行结果如下图所示：

图 1.21 平台测试结果

*注：输出结果过长的问题出自第最后一个时钟周期，MDin 的输出 00000000 之后出现一连串“ • ”，如下图所示，由于 MDin 的最大输出仅为 32 位，不存在出现 • 的情况，应当认为时平台错误而非实验错误。

综上所述，多周期微程序 CPU 设计正确。

1.5.3 多周期硬布线 CPU 执行

加载镜像 sort.hex，启动时钟模拟。运行周期数如下图所示：

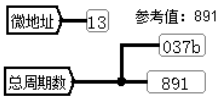


图 1.22 多周期硬布线 CPU 运行周期数

与参考周期数一致。查看存储器，存储器情况如下图所示：

```
080  00000006 00000005 00000004 00000003  00000002 00000001 00000000 ffffffff
```

图 1.23 存储器第 80 号单元开始的存储情况

可知排序的结果正确。将文件上载至 Educoder 平台运行，运行结果如下图所示：

图 1.24 平台测试结果

*注：输出结果过长的问题出自第最后一个时钟周期，MDin 的输出 00000000 之后出现一连串 •，如下图所示，由于 MDin 的最大输出仅为 32 位，不存在出现 • 的情况，应当认为时平台错误而非实验错误。

图 1. 25 平台测试警告截图

综上所述，多周期硬布线 CPU 设计正确。

2 总结与心得

2.1 实验总结

本次实验主要完成了如下几点工作：

1） 实现了对 MIPS 指令的解析

2） 实现了寄存器、存储器的读写

3） 实现了 MIPS 单周期、多周期硬布线和微程序 CPU 通路的绘制

4） 实现了 MIPS 单周期硬布线控制器的设计

5） 实现了 MIPS 多周期微程序控制器的设计

6） 实现了 MIPS 多周期微程序地址转移逻辑的的设计

7） 实现了 MIPS 多周期硬布线控制器的设计

8） 实现了 MIPS 多周期硬布线有限状态机的设计

2.2 实验心得

批注 [U1]: 主要讲课设体会，收获，以及对课设的建议

1） 巩固并熟练掌握了 Logisim 工具的使用方法。

2） 具体实践了 CPU 的工作原理，首次具体的了解到计算机底层的工作方式。

3） 对 MIPS 三类指令有了较为清晰的认识

4） 在 MOOC 的帮助下实验变得容易理解和上手。

参考文献

[1] DAVID A.PATTERSON(美).计算机组成与设计硬件/软件接口(原书第 5 版).北京:机械工业出版社.

[2] David Money Harris(美).数字设计和计算机体系结构（第二版）. 机械工业出版社

[3] 谭志虎,秦磊华,胡迪青.计算机组成原理实践教程.北京:清华大学出版社，2018 年.

[4] 秦磊华，吴非，莫正坤.计算机组成原理. 北京:清华大学出版社，2011 年.

[5] 袁春风编著. 计算机组成与系统结构. 北京:清华大学出版社，2011 年.

[6] 张晨曦，王志英. 计算机系统结构. 高等教育出版社，2008 年.

• 指导教师评定意见 •

一、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: 刘晨彦

二、对课程实验的学术评语（教师填写）

三、对课程实验的评分（教师填写）

评分项目 （分值）	报告撰写 （30 分）	课设过程 （70 分）	最终评定 （100 分）
得分			

指导教师签字: _____