華中科技大學

2020

计算机组成原理
课程实验报告

专　　业：信息安全

班　　级：

学　　号：

姓　　名：

电　　话：

邮　　件：

完成日期：2020 年 6 月 8 日

目　录

1　实验任务与要求

1.1 实验内容

1.掌握控制器设计的基本原理，利用已经构建的运算器、寄存器文件、存储系统等部件以及 logisim 中其他功能部件构建一个 32 位 MIPS CPU 单周期处理器。

2.掌握单周期 MIPS 处理器数据通路设计原理，采用微程序控制器的设计方法实现控制器，构造一个 32 位 MIPS CPU 多周期处理器。

1.2 具体要求

1.32 位单周期 MIPS 处理器：该处理器应支持如表 1-1 中的 8 条 MIPS 核心指令，且所有指令均能在一个时钟周期内完成。最终设计实现的单周期 MIPS 处理器能够运行冒泡排序测试程序 sort.hex，该程序自动在数据存储器 80~87 号字单元中写入 8 个数据，利用冒泡排序将数据有符号降序排列。实验电路能够自动统计指令执行过程中的时钟周期数，应为 224。

2.32 位多周期 MIPS 处理器：在单周期 MIPS 处理器基础上连接各功能部件，构建单周期 MIPS 处理器数据通路。该处理器应支持如表 1-1 中的 8 条 MIPS 核心指令，结合微程序控制器设计的基本原理，设计微指令格式并为每一条 MIPS 指令构建微程序。设计地址转移逻辑最终构建微程序控制器。最终实现的多周期 MIPS 处理器能够运行冒泡排序测试程序 sort.hex，使得存储器中 80~87 号字单元数据有符号降序排列。实验电路能够自动统计指令执行过程中的时钟周期数，应为 891。

表 1-1 MIPS 核心指令集

#	格式	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[$rs]+SignExt_16b(imm))←R[$r]

6	beq $rs, $rt, imm	if(R[$rs]=R[$rt]) PC←PC+SignExt_16b({imm,00})
7	bne $rs, $rt, imm	if(R[$rs]!=R[$rt]) PC←PC+SignExt_16b({imm,00})
8	syscall	系统调用,这里用于停机

2　设计方案

2.1　32 位 MIPS 单周期处理器

2.1.1 构建主要功能部件及其功能

1.程序计数器 PC：存放当前指令地址

2.指令存储器 IM：存储 CPU 要执行的指令

3.数据存储器 DM：存储 CPU 要处理的数据

4.立即数扩展器 S-EXT：

（1）将 MIPS 的 I 型指令中 16 位立即数符号扩展为 32 位

（2）条件转移指令中 16 位偏移地址符号扩展为 32 位

5.地址转移逻辑 NPC：确定下条执行指令的地址

（1）顺序执行：PC+4，即下条指令的地址

（2）条件转移指令：跳转到下条指令加上偏移地址的指令处

6.运算器 ALU：用于对操作数进行算术逻辑运算

7.寄存器堆 RegFile：提供 32 个 MIPS 通用寄存器

8.单周期硬布线控制器：根据 MIPS 指令的操作码 OP 字段和 Func 字段产生指令执行过程中的所有控制信号

2.1.2 构建功能部件输入来源表

逐一分析每条指令的功能，并根据指令功能记录各功能部件输入端数据来源，仅保留数据类信号，绘制成表 2-1 单周期功能部件输入来源表。

表 2-1 单周期功能部件输入来源表

指令	PC	IM	RegFile				S-EXT	ALU		DM	
			R1#	R2#	W#	Din		A	B	Addr	Din
add	PC+4	PC	rs	rt	rd	ALU		RF.D1	RF.D2		
slt	PC+4	PC	rs	rt	rd	ALU		RF.D1	RF.D2		
addi	PC+4	PC	rs		rt	ALU	IMM[15:0]	RF.D1	S-EXT		
lw	PC+4	PC	rs		rt	DM.Dout	IMM[15:0]	RF.D1	S-EXT	ALU	
sw	PC+4	PC	rs	rt			IMM[15:0]	RF.D1	S-EXT	RF.D2	ALU

beq	PC+4+offset/PC	PC	rs	rt			IMM[15:0]	RF.D1	RF.D2		
bne	PC+4+offset/PC	PC	rs	rt			IMM[15:0]	RF.D1	RF.D2		
syscall	PC	PC									

2.1.3 数据通路综合

将上述表 2-1 主要功能部件输入来源表中功能部件的输入按列进行合并，若某个输入有多个输入来源则引入多路选择器，同时新增多路选择器的选择控制信号。根据合并后的输入来源在 logisim 中连接各主要功能部件，构建单周期 MIPS 处理器的所有数据通路。

2.1.4 控制信号综合

1.分析所有功能部件、多路选择器控制器信号、运算操作选择的产生条件：

（1）RegDst 为 1 表示 RegiFile 写回地址由 R 型指令 rd 字段给出，否则由 I 型指令 rt 字段给出

（2）RegWrite 为 1 打开寄存器堆 RegiFile 写使能，将数据写入寄存器中；

（3）AluSrc 为 1 表示 ALU 的第二个操作数将由立即数扩展器 S-EXT 给出，否则由 RegiFile 的第二个输出 R2 给出；

（4）Bne 为 1 表示指令为 bne；

（5）Beq 为 1 表示指令为 beq，Beq 与 Bne 信号与 ALU 的 Equal 信号结合可控制 PC 是否发生跳转；

（6）MemWrite 为 1 打开数据存储器 DM 写使能，将 ALU 运算结果写入数据存储器 DM；

（7）MemToReg 为 1 表示从数据存储器 DM 中选数据送入 RegiFile 写入端；

（8）Halt 为停机信号，为 1 时系统停机；

（9）ALU_OP：ALU 操作信号，可能为加法信号或有符号比较信号

2.根据上述信号产生条件，确定每条指令执行时所需的控制信号，构建控制信号表，如表 2-2。

表 2-2 单周期 MIPS 处理器控制信号表

控制信号	1	2	3	4	5	6	7	8

	add	slt	addi	lw	sw	beq	bne	syscall
RegDst	1	1						
RegWrite	1	1	1	1				
AluSrc			1	1	1			
Bne							1	
Beq						1		
MemWrite					1			
MemToReg				1				
Halt				1				1
ALU_OP 加	1		1	1	1			
ALU_OP 符号比较		1						

3.对照控制信号表，根据指令译码信号实现控制器输出控制信号的逻辑，构建指令译码信号部分电路，如图 2-1.

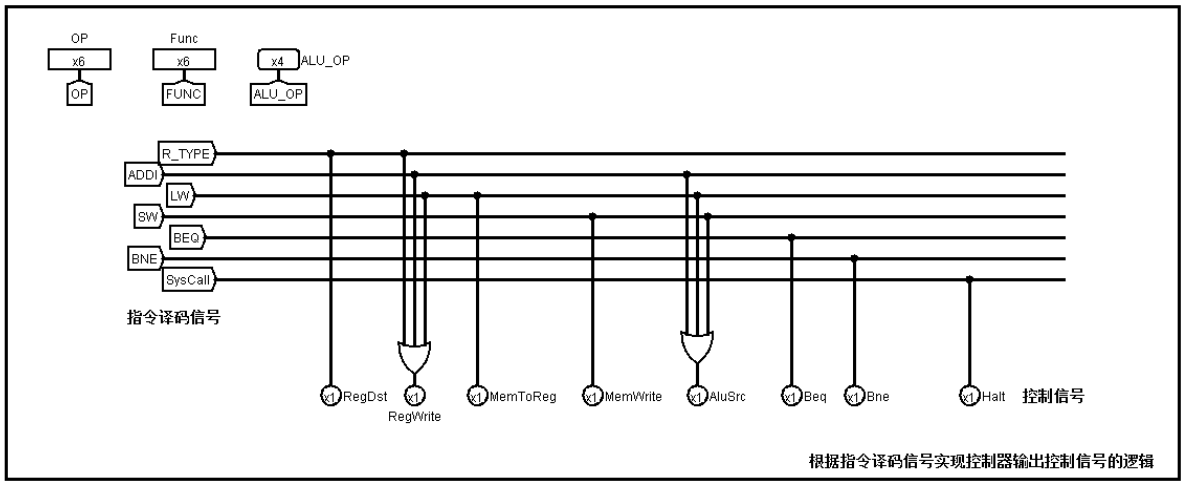


图 2-1 单周期 MIPS 控制器指令译码电路

表 2-3 MIPS 核心指令及其对应操作码

指令	指令	OP 字段	Func 字段
add	R 型	00h	20h
slt	R 型	00h	2ah
addi	I 型	08h	
lw	I 型	23h	

sw	I 型	2bh	
beq	I 型	04h	
bne	I 型	05h	
syscall	R 型	00h	0ch

4.根据根据 MIPS 指令及其 OP 字段和 Func 字段，确定指令的译码逻辑，如表 2-3，再结合 ALU_OP 信号确定相应的 ALU 控制器逻辑，最分别构建指令译码逻辑电路和 ALU 控制器逻辑电路，如图 2-2。

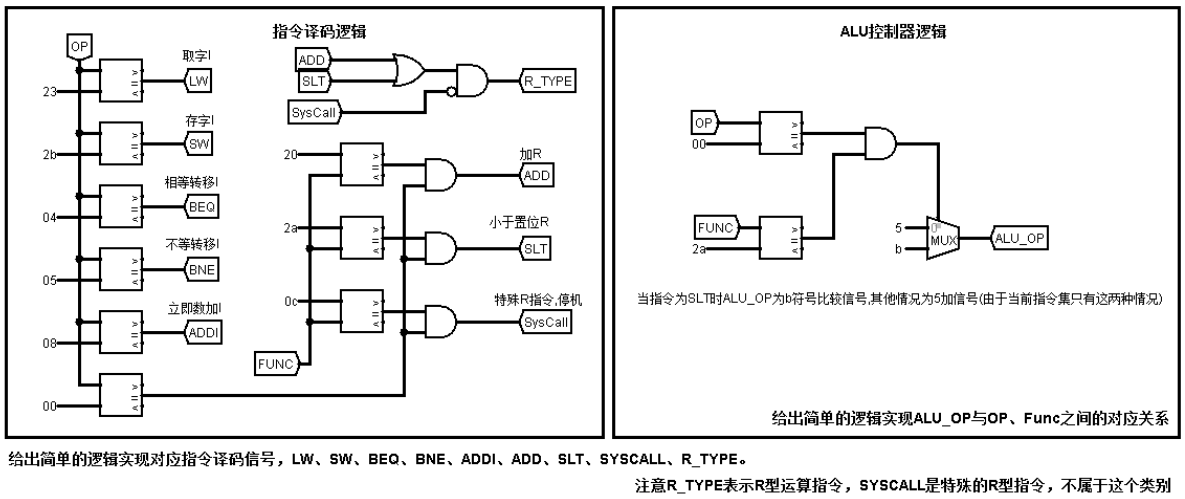


图 2-2 单周期 MIPS 控制器建指令译码逻辑和 ALU 控制器逻辑电路

2.1.5 单周期 MIPS 总体构建

1.跳转跳跃信号产生：

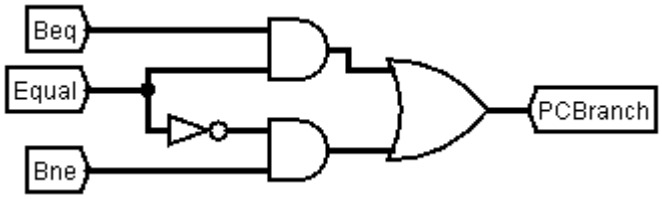


图 2-3 跳跃寻址信号 PCBranch 产生电路

2.单周期 MIPS 时钟周期计数模块：

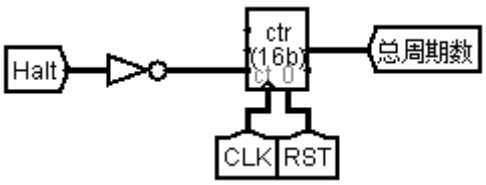


图 2-4 单周期 MIPS 时钟周期计数模块

结合单周期 MIPS 的数据通路、控制器和指令的控制信号以及以上辅助模块最终构

建单周期 MIPS 处理器的总体结构图，如图 2-5。

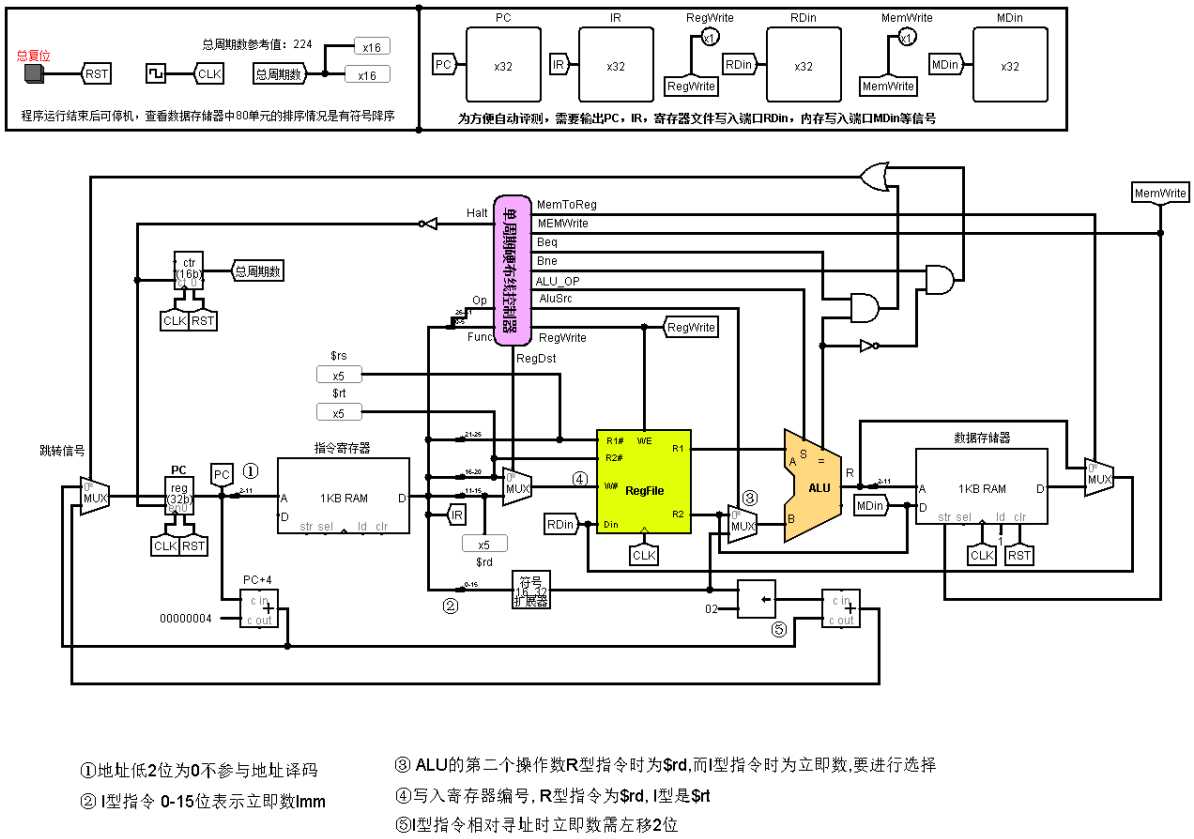


图 2-5 单周期 MIPS 处理器（硬布线）总体结构图

2.2 32 位 MIPS 多周期处理器(微程序)

2.2.1 多周期处理器与多周期处理器的区别

1.不再区分指令存储器与数据存储器，指令和数据保存在同一个存储器中

2.部分功能单元，如 ALU、RegiFile 可以在一条指令执行过程的不同周期中多次使用

3.主要功能单元输出端都增加了寄存器，在后续时钟周期中要用到的所有数据必须存储在相应的寄存器中：增加数据寄存器 DR，用于存放从存储器读取的数据；增加指令存储器 IR，用于存放从存储器读出的指令；增加了 3 个寄存器 A、B、C 用于存放从寄存器文件 RegiFile 的读出数据和 ALU 的运算结果，其中多周期处理器的 ALU 的输出结果可能是下列三者之一：

（1）分支目标地址，由 beq 或 bne 指令给出，该地址将被写到 PC 中；

（2）指令的运算结果，将被写入到寄存器文件中；

（3）为存储器访问指令 lw 和 sw 提供存储器地址；

4.增加了几个多路复用器，在共享部件前增加或修改了多路复用器

5.PC 作为指令计数器，由于不同指令时钟周期数不同，因此其写操作将不再仅由时钟周期控制，而且增加了专门的写操作控制信号；

2.2.2 构建主要功能部件和数据通路

1.多周期处理器功能部件及其功能：

（1）指令计数器 PC：存放当前指令地址

（2）存储器 Mem：用于存放指令和数据

（3）指令寄存器 IR：存放当前指令

（4）数据寄存器 DR：存放要操作的数据

（5）寄存器堆 RegiFile：提供 32 个 MIPS 通用寄存器

（6）三个寄存器 A、B、C：存放从寄存器文件 RegiFile 的读出数据和 ALU 的运算结果

（7）立即数扩展器 S-EXT：

①将 MIPS 的 I 型指令中 16 位立即数符号扩展为 32 位

②条件转移指令中 16 位偏移地址符号扩展为 32 位

（8）算术逻辑单元 ALU：用于对操作数进行算术逻辑运算

（9）微程序控制器：产生控制信号，控制指令执行的数据通路；

2.构建数据通路：根据单周期处理器总体结构以及功能部件输入来源表，结合单周期处理器与多周期处理器功能部件的区别，构建数据通路。

2.2.3 设计微指令

1.分析指令操作流程确定控制信号：

多周期处理器执行不同指令对应不同的时钟周期数，每条指令的执行可拆分为“取指->译码->执行”三个阶段，其中 “取指”和“译码”两阶段所有指令所对应的数据通路相同，结合指令功能分析每条指令执行阶段需要的时钟周期并确定每个时钟周期下所需控制信号，如表 2-4.

表 2-4 指令操作流程及控制信号

指令	指令阶段	操作流程	控制信号
	取指令	IR<-(MEM[PC])，PC<-(PC)+4	MemRead=IrWrite=PCWrite=1， AluSrcB=PCsrc=01
	译码	A<-(R[IR[25:21]]) B<-(R[IR[20:16]]) C<-(PC)+(S- EXT(IR[15:0])<<2)	AluSrcB=11
R_TYPE	加/比较运算	C<-(A)+(B)	AluSrcA=1， ALU_OP=10
	写回	R[IR[15:11]]<-(C)	RegDst=RegWrite=1
lw	计算地址	C<-(A)+S-EXT(IR[15:0])	AluSrcA=1，AluSrcB=10
	访存	DR<-(MEM[PC])	IorD=MemRead=1
	写回	R[IR[20:16]]<-(DR)	MemToReg=MemWrite=1
sw	计算地址	C<-(A)+ S-EXT(IR[15:0])	AluSrcA=1，AluSrcB=10
	访存	DR<-(MEM[PC])	IorD=MemWrite=1
beq	送目标地址	If(A==B) PC<-(C)	PCsrc=AluSrcA=Beq=1，ALU_OP=01
bne	送目标地址	If(A!=B) PC<-(C)	PCsrc=AluSrcA=Bne=1，ALU_OP=01
addi	加运算	C<-(A)+ S-EXT(IR[15:0])	AluSrcA=1，AluSrcB=10
	写回	R[IR[20:16]]<-(C)	RegWrite=1
syscall	空操作，停机	锁住 PC	

2.微指令格式设计：

微指令由微操作控制字段、判别测试字段 P 和下址字段组成，其中微操作控制字段用来表示该条微指令的控制信号；判别测试字段用于地址逻辑转移，仅译码微指令的该字段 P=1，下条微指令地址由地址转移逻辑给出，其他微指令为 0，下条微指令由下址字段给出；下址字段在 P 为 0 时用于指示即将访问的下一条微指令的地址。

根据表 2-4 确定微指令的控制信号，结合每条微指令在微指令控制存储器的存放地址，确定每条微指令的下址字段和判别测试字段。利用微指令自动生成.xlsx 表格，生成微指令，获得其 16 进制表示形式。

图 2-6 微指令表

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	0	0	0000	00000010010000000000	2400
BEQ	9	1001	0	1	1	00	0	0	0	0	0	0	0	1	0	01	0	0000	01100000000100100000	C0120
BNE	10	1010	0	1	1	00	0	0	0	0	0	0	0	0	1	01	0	0000	01100000000010100000	C00A0
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	00000000100000000000	800
SYSCALL	13	1101	0	0	0	00	0	0	0	0	0	0	0	0	0	00	0	1101	00000000000000001101	D

2.2.4 构建微程序控制器

1.设计地址转移逻辑电路：

结合图 2-6 每条微指令的地址，易得每条 MIPS 指令对应的微程序入口地址，如图 2-7.

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 2-7 微程序入口地址表

利用微程序地址转移逻辑自动生成.xlsx 表格，可自动生成对应电路的表达式：

S3 = ADDI + BEQ + BNE + SYSCALL

S2 = R_Type + SW + SYSCALL

S1 = R_Type + ADDI + LW + BNE

S0 = R_Type + ADDI + SW + BEQ + SYSCALL

利用 logisim 自动生成电路，如图 2-8.

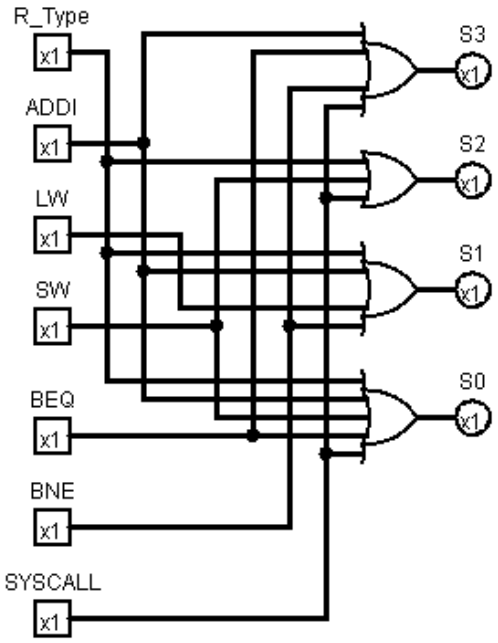


图 2-8 地址转移逻辑电路

2.构建微程序控制器：

多周期处理器的指令译码逻辑和 ALU 控制器逻辑与单周期处理器相近，如图 2-9。

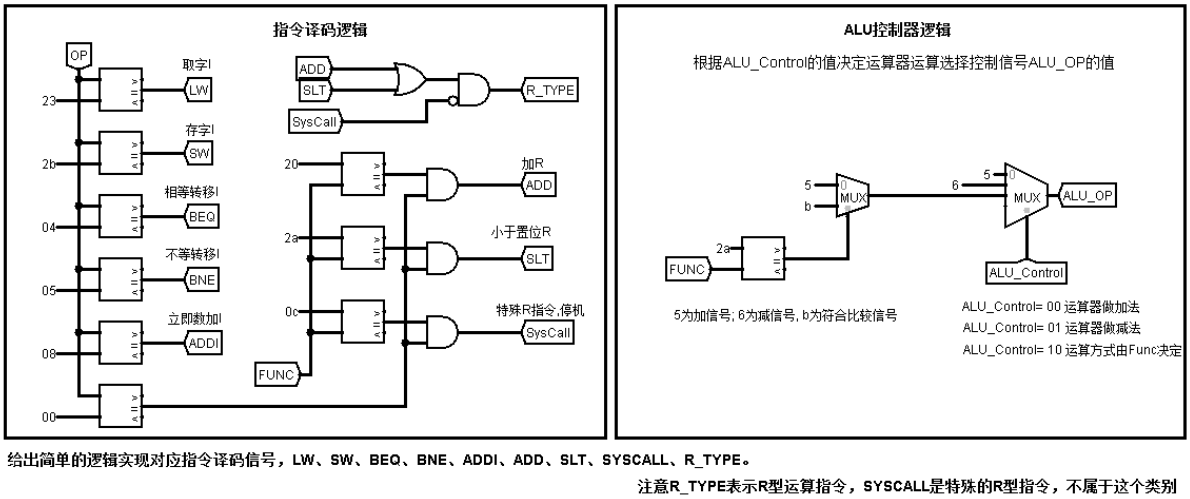


图 2-9 多周期处理器指令译码逻辑和 ALU 控制器逻辑电路

结合微指令格式以及地址转移逻辑，可构建微程序控制器。其中，由图 2-6 获得的微程序存入到控制存储器中。

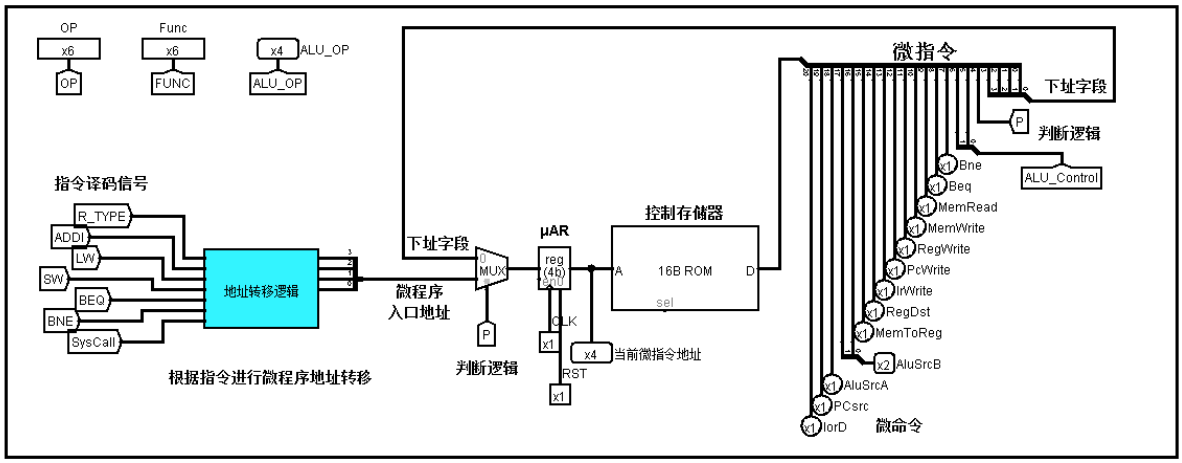


图 2-10 微程序控制

2.2.5 多周期 MIPS（微程序）总体构建

1.多周期处理器 PC 写使能信号 PCEn 产生：

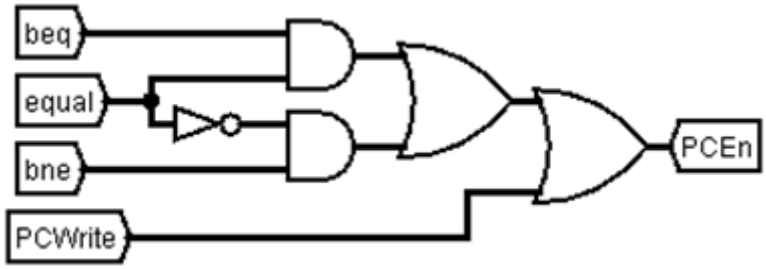


图 2-11 多周期处理器 PC 写使能信号产生电路

2. 多周期处理器（微程序）周期计数模块：微指令地址为 13（0xd）时，系统停机

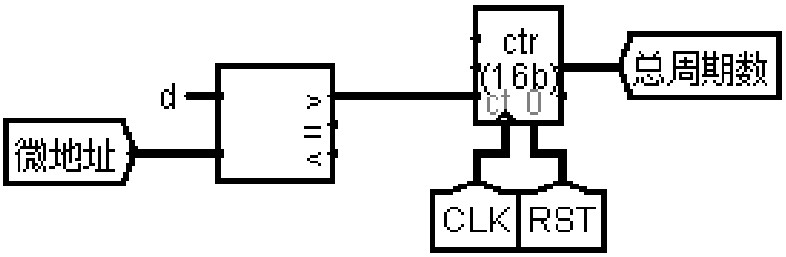
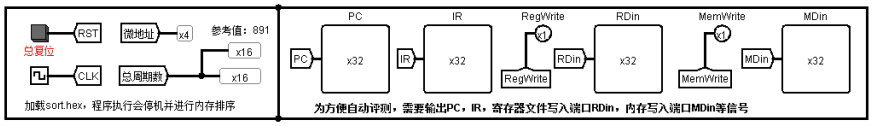
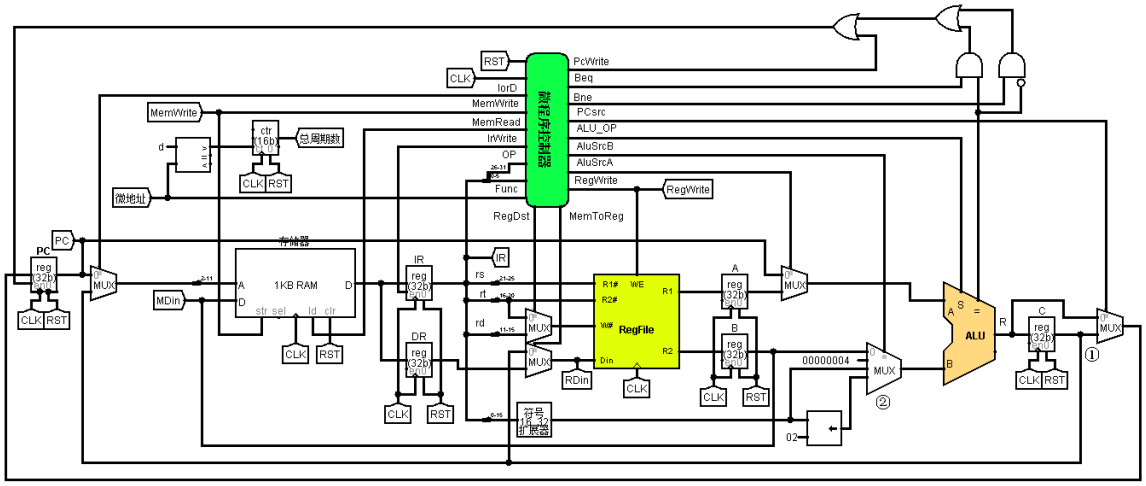


图 2-12 多周期处理器（微程序）周期计数模块

结合多周期 MIPS 的数据通路、微程序控制器和指令的控制信号以及以上辅助模块

最终构建多周期 MIPS 处理器（微程序）的总体结构图，如图 2-13。





① ALU输出结果可能为: 分支目标地址,将被写入PC; 运算结果, 将被写入寄存器; 为存储器访问指令lw sw提供有效地址

②ALU操作数B可能为: $rt/$rd; 常数4,表示PC+"1"; 指令中的立即数, 指令中立即数左移两位的值

图 2-13 多周期 MIPS 处理器（微程序）总体结构图

3　测试及故障调试

3.1 测试

3.1.1 MIPS 单周期处理器执行 sort.hex

1.时钟周期：执行完毕后，系统停机，sort.hex 的总时钟周期数为 224，如图：

图 3-1 单周期 MIPS 处理器总时钟周期数

2.内存布局：在数据存储器的 80~87 号单元为 6,5,4,3,2,1,ffffff(-1)的有符号降序数据，如图：

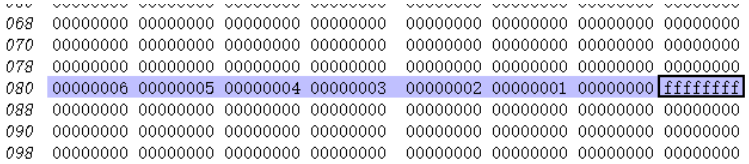


图 3-2 单周期 MIPS 处理器执行 sort.hex 后的内存布局

3.1.2 MIPS 多周期处理器（微程序）执行 sort.hex

1.时钟周期：执行完毕后，系统停机，sort.hex 的总时钟周期数为 981，如图：

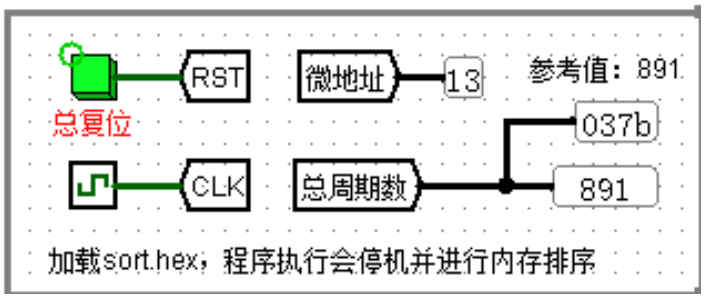


图 3-3 多周期 MIPS 处理器总时钟周期数

2.内存布局：在存储器的 80~87 号单元有 6,5,4,3,2,1, ffffff(-1)的有符号降序数据，如图：

000	2010ffff	20110000	ae300200	22100001	22310004	ae300200	22100001	22310004
008	ae300200	22100001	22310004	ae300200	22100001	22310004	ae300200	22100001
010	22310004	ae300200	22100001	22310004	ae300200	22100001	22310004	ae300200
018	00008020	2011001c	8e130200	8e340200	0274402a	11000002	ae330200	ae140200
020	2231fffc	1611fff8	22100004	2011001c	1611fff5	2002000a	0000000c	00000000
028	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
030	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
038	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
040	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
048	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
050	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
058	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
060	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
068	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
070	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
078	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
080	00000006	00000005	00000004	00000003	00000002	00000001	00000000	ffffffff
088	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
090	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
098	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000

图 3-4 多周期 MIPS 处理器（微程序）执行 sort.hex 后的内存布局

3.2 遇到的问题及处理

遇到的问题：多周期 MIPS 处理器运行 sort.hex 程序时，能得到正确的内存布局，但是总周期数为 212，并不是正确的 891。

问题处理：通过调试与分析发现，CPU 总体结构图和微程序控制器没有问题，但是存在指令执行过程中时钟周期数统计错误。经排查发现，是多周期处理器时钟周期统计模块的比较器属性未设置正确，应该是无符号比较器，而我设置成了有符号比较器，这样对于地址大于 7 的微指令的地址都当作负数，使得比较器无法得出正确结果，从而使计数器无法正确计数。因此，多周期处理器时钟周期计数模块的比较器应该为无符号比较器。

3.3 设计方案存在的不足

当前设计的 MIPS 多周期处理器（微程序）的微程序控制器的微指令长度较长，可以进一步优化：结合部分控制信号的互斥性和微指令的地址连续性，分别对互斥的控制信号进行编码、使用译码器译码，以及使用加法器控制微指令地址+1，从而缩短微指令的长度。

4　设计总结与心得

4.1 实验总结

本次实验通过对单周期和多周期 MIPS 处理器的设计，加深了对 MIPS 指令尤其是其核心指令集的认识，学会了如何使用电路对 MIPS 指令进行解析；对单周期和多周期 MIPS 处理器的整体架构有了更深刻的认识，认识到了单周期与多周期在指令执行过程以及电路设计上的不同，学会了如何构建 MIPS CPU。了解了 CPU 控制器的结构并实现了控制信号的生成。在多周期 MIPS 处理器的设计中，也加深了对微指令的格式的认识，掌握了微指令不同字段在指令执行过程中的不同作用，以及如何将 MIPS 指令转换为对应的微程序，并结合微程序入口地址转移电路构建起微程序控制器。最终成功构建了单周期 MIPS 处理器和多周期微程序 MIPS 处理器。

4.2 实验心得及建议

1.实验心得：

CPU 的设计实验整体难度比较大, 尤其是刚开始构建单周期 MIPS 处理器，在对 MIPS 指令了解还不够充分的情况下做开始也有些无从下手, 通过对实验的实践教程和 MOOC 上的视频的进一步学习, 逐渐将电路搭起。而多周期 MIPS 处理器的设计是在单周期处理器的基础上进行完善构建的，架构上有较多相似之处，在有了之前单周期 CPU 设计的经验后，构建多周期 CPU 相对就不那么困难。通过 MIPS CPU 的实验，也让我对 CPU 的结构及其设计有了进一步的认识，认识到对于相同的指令集，不同 CPU 的设计方案对处理器的性能，控制信号的需要是不同的。对于理论课上很多知识，在做了实验以后会更容易理解。此外，对于 CPU 的设计，电路相对就比较复杂了，在实验过程中需要认真仔细，对于一些控件属性的选择、逻辑门的选择都要准确，否则可能会因为细小的差别而使得处理器不能正确运行。

整个实验加深了我对计算机 CPU 结构的认识，巩固了我在理论课所学的相关知识，提高了我的电路设计和实验操作能力，很有收获。

2.建议：

由于理论课上讲述的知识点偏向理论，与实际设计过程中如何用逻辑器件实现功能

有一定的差别，希望实验前老师的讲解和指导能更加具体，或者多提供些参考书籍资料。

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 4 版).北京：机械工业出版社.

[2] 谭志虎,秦磊华,胡迪青.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.

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。

作者签名：