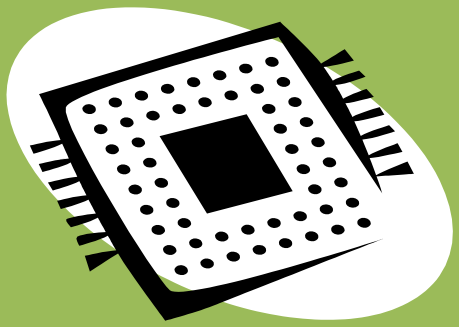
华中科技大学

2018

计算机组成原理 ·实验报告·

专　　业：	计算机科学与技术
班　　级：	CSIE1601
学　　号：	U201610504
姓　　名：	刘逸帆
电　　话：	13007159315
邮　　件：	lyf460315457@qq.com
完成日期：	2018-12-14



计算机科学与技术学院

目　录

1 单周期 CPU 设计实验

1.1 设计要求

利用实验提供的经过封装的寄存器文件 Regfile、ALU 以及 logisim 平台中现有运算部件构建一个单周期 MIPS CPU，最后在主电路中详细测试自己封装的 CPU。CPU 的 8 条核心指令如表 1.1 所示。

表 1.1 MIPS 指令集

#	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]+SignExt16b(imm) 溢出产生异常
4	lw $rt,imm($rs)	R[$rt]←Mem4B(R[$rs]+SignExt16b(imm))
5	sw $rt,imm($rs)	Mem4B(R[$rs]+SignExt16b(imm))←R[$rt]
6	beq $rs,$rt,imm	if(R[$rs] = R[$rt]) PC ← PC + SignExt18b({imm, 00})
7	bne $rs,$rt,imm	if(R[$rs] != R[$rt]) PC ← PC + SignExt18b({imm, 00})
8	syscall	系统调用，这里用于停机

在 CPU 中，控制信号需要通过控制器生成。根据 MIPS CPU 对控点的要求，要求设计一个单周期 MIPS CPU 硬布线控制器。单周期控制器无时序逻辑，为纯组合逻辑电路。控制器包含的输入信号有：指令字 Opcode、Func 字段（共 12 位）；控制器包含的输出信号有：多路选择器选择信号，内存访问控制信号，寄存器写使能信号，运算器控制信号、指令译码信号等。每种控制信号的具体功能说明见表 1.2 所示。

表 1.2 控制信号功能说明

#	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

1.2 方案设计

1.2.1 单周期硬布线 CPU 设计

设计原理：单周期 CPU 指的是一条指令在一个时钟周期内完成执行，然后开始下一条指令的执行，即一条指令用一个时钟周期完成。单周期 CPU 处理指令时在一个周期内可能执行如下步骤中的某几步：取指令、指令译码、指令执行、存储器访问、结果写回。指令执行的一般流程为取指令、执行指令反复循环，设计 CPU 时应为每条指令设计对应的数据通路。数据通路设计完成后可尝试从指令存储器中取得指令并执行，从而验证 CPU 数据通路的正确性。

设计思路：设计 CPU 首先应选定指令系统，用 RTL 表示指令功能；根据指令功能设计功能部件与通路，互连各部件；确定各个部件需要的控制信号；汇总控制信号分析指令信号序列；根据指令与控制信号的关系表设计控制器。将每个设计步骤中的结果汇总便可以绘制 CPU 的逻辑电路图。

设计过程：

(1) 选定指令系统：选用 MIPS 的指令系统，直接使用指令集中已设计好的 RTL；

(2) 设计部件与通路：以 R 型指令 ADD 和 LW 指令为例，设计八条核心指令对

应的数据通路:

a．ADD 指令数据通路的设计

ADD 指令为 R 型指令，其 RTL 功能描述为 R[$rd]←R[$rs]+R[$rt]。对于寄存器文件 RegFile，其 R1#、R2#、W#寄存器入口地址分别为指令中 Rs、Rt、Rd 部分。取得 R1#、R2#两个地址处的寄存器值后，通过 ALU 进行求和运算，运算结果从 RegFile 的 Din 输入数据端接入，为 RegFile 给写使能 WE，将结果保存在 Rd 指向的寄存器中。ADD 指令数据通路如图 1.1 所示。

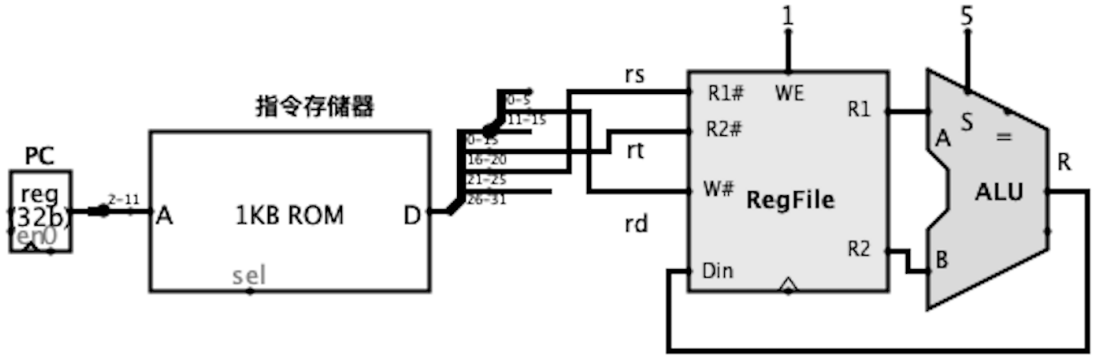


图 1.1 ADD 指令数据通路

b．LW 指令数据通路的设计

LW 指令为 I 型指令，RTL 功能描述为 R[$rt]←Mem4B(R[$rs]+SignExt16b(imm))。即将指令中的立即数进行有符号位扩展后，与 Rs 指向的寄存器内容通过 ALU 相加，并将 ALU 的运算结果写入内存。同时将存入内存的数据作为 RegFile 器件中的 Din 输入，写入 Rt 指向的寄存器中。LW 指令数据通路如图 1.2 所示。

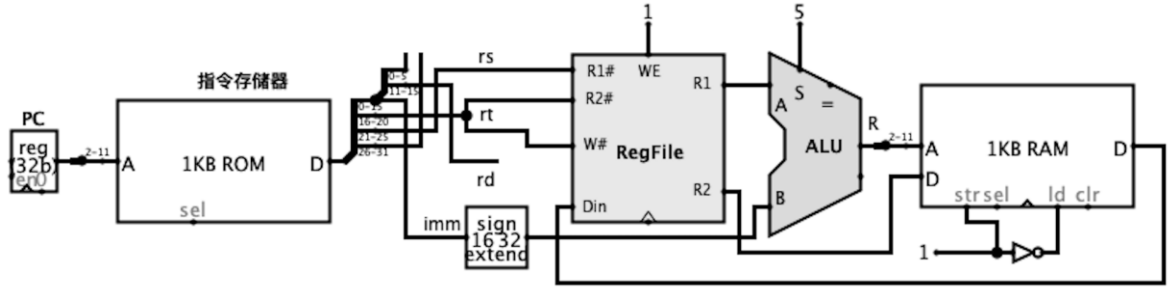


图 1.2 LW 指令数据通路

类似地，对八条核心指令分别进行数据通路的设计。

(3) 设计控制信号（控点）：为了合并不同指令的数据通路，需要设计相应的控制信号（控点）控制多路选择器。控制信号能够实现对数据通路的控制，保证在多输入的情形下能够取得正确的输入信号，使得不同的指令能够在 CPU 中共享相同的器件。以利用控制信号将 ADD 指令和 LW 指令的数据通路进行合并为例，设计控制信号合并八条核心指令的数据通路：

ADD 指令为 R 型指令，LW 指令为 I 型指令，对两条指令的数据通路而言，向 RegFile 寄存器器件输入的寄存器地址存在差异；对于 ALU 器件而言，虽然都是加运算，但输入的来源不同，ADD 数据通路中 ALU 的输入分别来自两个寄存器，而 LW 数据通路中 ALU 的输入分别来自寄存器与指令中的立即数；另外，对于 RegFile 的 Din 数据输入而言，两个数据通路中的数据来源也同。对于两条数据通路的上述区别，都需要使用多路选择器选择输入信号，从而保证两条数据通路的正确合并。在本例中，使用 RegDst 控制信号控制写入寄存器地址的来源、使用 ALUSrc 信号控制 ALU 的输入来源、使用 MemToReg 信号控制寄存器数据输入的来源，从而实现两条数据通路的合并，合并后的数据通路如图 1.3 所示。

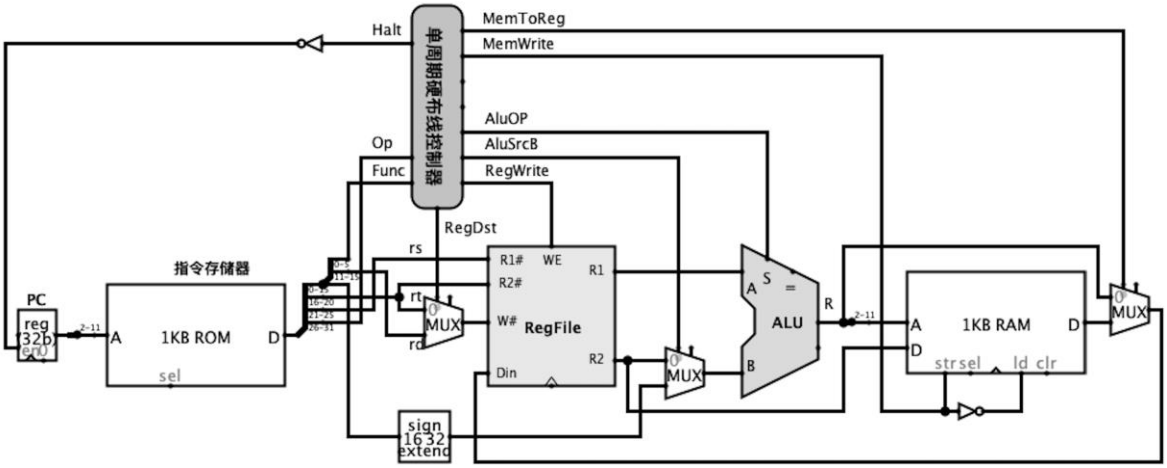


图 1.3 合并后 ADD 指令与 LW 指令的数据通路

(4) 分析指令信号序列：汇总八条核心指令的控制信号序列，生成指令与控制信号关系表，如表 1.3 所示。

表 1.3 指令与控制信号关系表

指令	MemToReg	MemWrite	Beq	Bne	AluOP	AluSrcB	RegWrite	RegDst	Halt
ADD	0	0	0	0	0101	0	1	1	0
SLT	0	0	0	0	1011	0	1	1	0
LW	1	0	0	0	0101	1	1	0	0
SW	0	1	0	0	0101	1	0	0	0
BEQ	0	0	1	0	0000	0	0	0	0
BNE	0	0	0	1	0000	0	0	0	0
ADDI	0	0	0	0	0101	1	1	0	0
Syscall	0	0	0	0	0000	0	0	0	1

(5) 设计控制器：根据指令与控制信号关系表，生成每个控制信号的逻辑表达式，从而设计硬布线控制器。控制器将在 CPU 中根据输入指令的不同，输出对应的控制信号，从而实现对 CPU 数据通路的控制。控制器的具体设计过程见 1.2.2 小节。

单周期 MIPS CPU 设计完成后结构如图 1.4 所示。

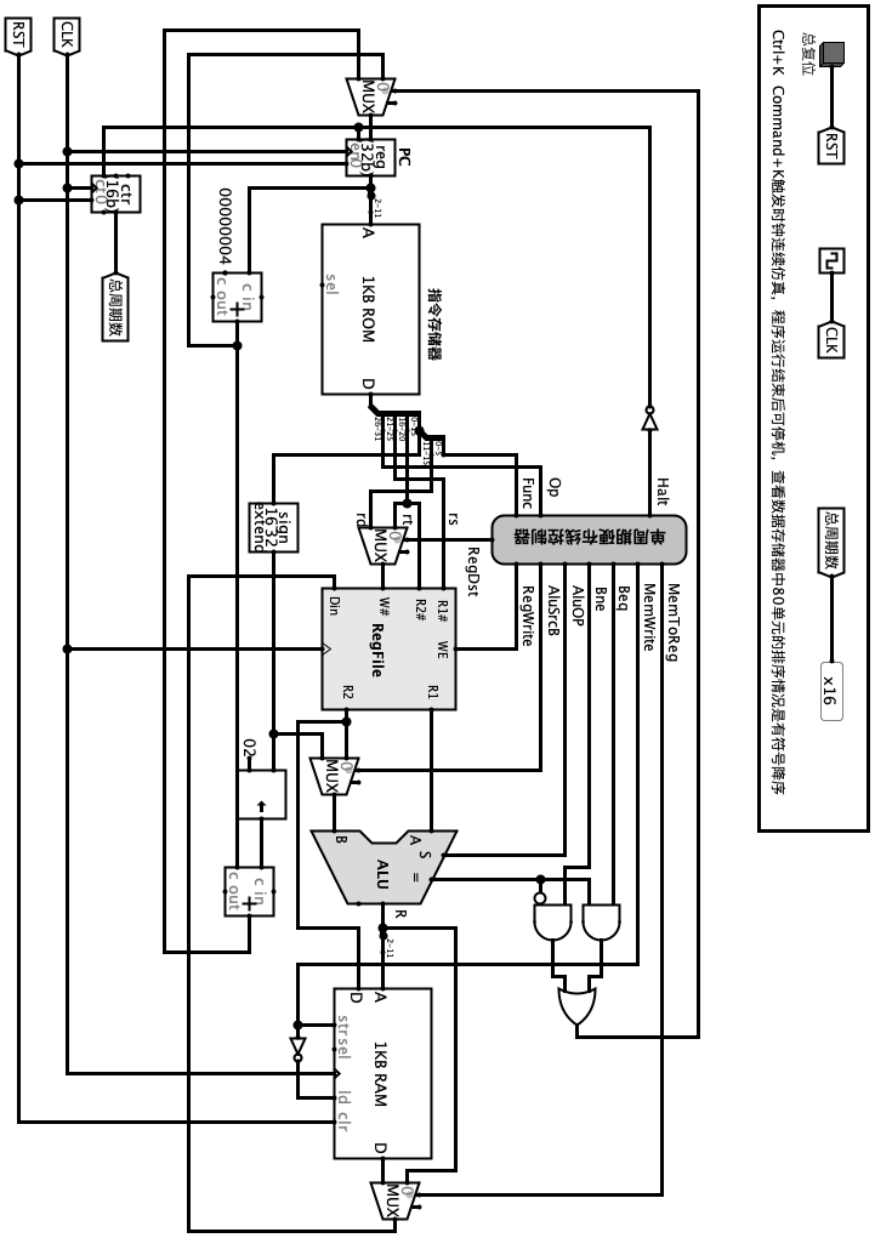


图 1.4 单周期 MIPS CPU 结构图

1.2.2 单周期硬布线控制器设计

设计原理：CPU 中的控制器器件用于根据输入信号译码出指令并产生对应的控制信号序列，即控制器是产生固定时序控制信号的逻辑电路，使用组合逻辑和多路选择器即可完成控制器设计。

设计思路：对输入信号 OP 和 FUNC 进行指令译码，得到具体指令；对每条指令（根据指令信号序列）分析其需要的控制信号并输出对应控制信号。

设计过程：

(1) 指令译码：通过比较器将输入信号 OP 与根据 MIPS 指令集预设的常数值进行比较，从而判断指令类型。若属于 R 型指令，继续与输入信号 FUNC 进行比较，从而判断具体的 R 型指令类型。

(2) 产生控制信号：指令译码得到具体指令类型后，根据 MIPS 指令集中每条指令的具体功能描述确定需要产生哪些控制信号，通过组合逻辑为每条指令产生对应地控制信号。对于 ALU 的控制信号 ALUOP，使用多路选择器实现信号的产生。

单周期硬布线控制器设计完成后结构图如图 1.5 所示。

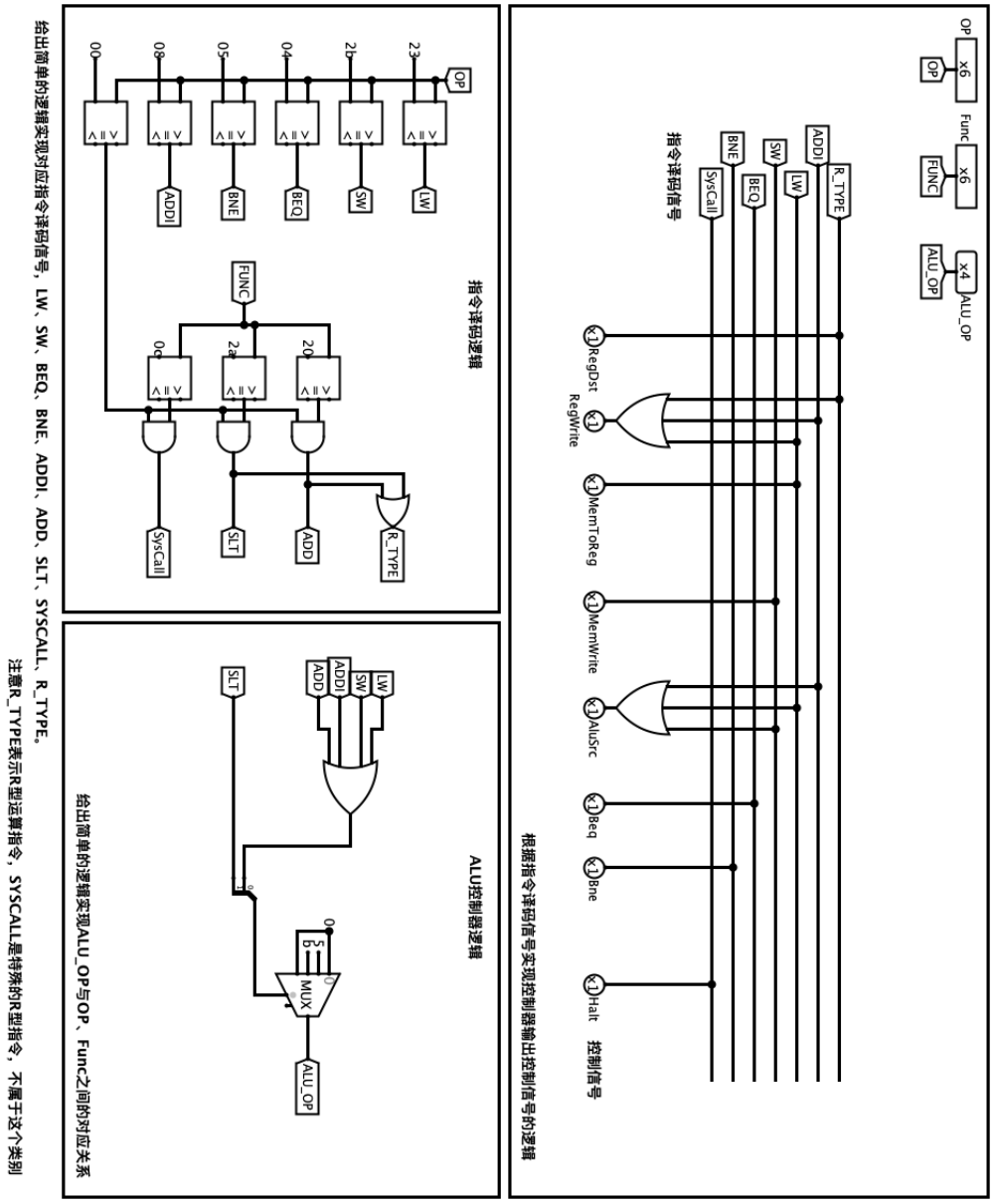


图 1.5 单周期硬布线控制器结构图

1.3 实验步骤

（1） 根据 1.2.1 中单周期 MIPS CPU 的设计过程，在 logisim 平台上绘制数据通路。

（2） 根据 1.2.2 中单周期硬布线控制器的设计过程，完成控制器内部电路的绘制。

（3） 令 CPU 读取单条指令，测试数据通路与控制器的设计实现正确与否。

（4） 用冒泡排序测试电路。

1.4 故障与调试

1.4.1 控制器控制信号产生有误

故障现象：CPU 在执行 SW 指令时寄存器组数据出错。

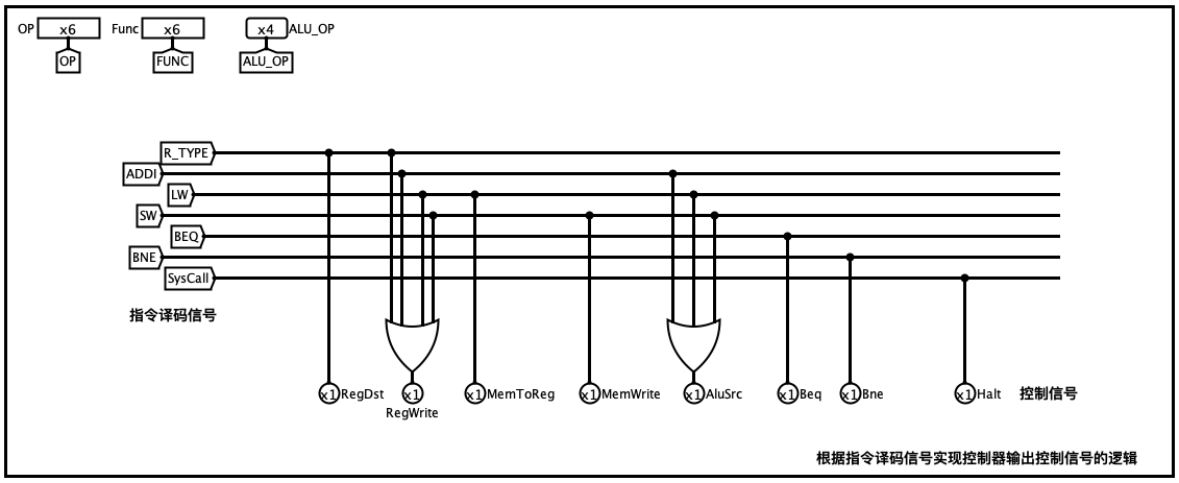


图 1.6 SW 指令控制信号产生错误图

原因分析：如图 1.所示，在设计单周期硬布线控制器的过程中，指令译码得到指令类型为 SW 时，产生了 RegWrite 写寄存器信号。但实际上根据 RTL 功能描述 Mem4B(R[$rs]+SignExt16b(imm))←R[$rt]，SW 指令只需要读内存信号 MemWrite。错误的控制信号导致寄存器组中的数据在 SW 指令执行时会由于误写而出错，从而 CPU 无法正常工作。

解决方案：检查了指令与控制信号关系表，关系表正确后修改单周期控制器中控制信号 RegWrite 的产生条件保证控制器的正常工作。

1.4.2 数据存储器中数据无法存储到预期位置

故障现象：执行冒泡排序后，数据无法保存到数据存储器中的正确位置，排序后的数据没有存储在预期的 80 地址处，而是存储在了 200 号地址处，错误现象如图 1.7 所示。

1f0	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
200	00000006	00000000	00000000	00000000	00000005	00000000	00000000	00000000
200	00000004	00000000	00000000	00000000	00000003	00000000	00000000	00000000
210	00000002	00000000	00000000	00000000	00000001	00000000	00000000	00000000
210	00000000	00000000	00000000	00000000	ffffffff	00000000	00000000	00000000
220	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000

图 1.7 访问数据存储器地址错误图

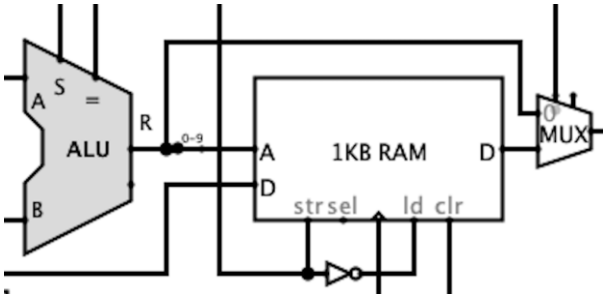


图 1.8 访问数据存储器数据通路错误图

原因分析：如图 1.8 所示，在设计数据通路的过程中，由 ALU 输出的地址信号没有舍弃低 2 位。在寻址时由于是字寻址，需要舍弃低 2 位，否则会导致地址*4，出现数据存储在与预期不符的错误地址上的情况。

解决方案：ALU 的输出在作为地址线连接数据存储器的输入端口前，通过分离器舍弃低 2 位，数据存储器中存储的数据恢复正常。

1.5 测试与分析

使用冒泡排序指令对单周期 CPU 进行测试，排序结果如图 1.所示 。

080	00000006	00000005	00000004	00000003	00000002	00000001	00000000	ffffffff

图 1.9 冒泡排序测试单周期 CPU 结果

2 多周期 CPU 设计实验

2.1 设计要求

利用实验提供的经过封装的寄存器文件 Regfile、ALU 以及 logisim 平台中现有运算部件构建一个多周期 MIPS CPU，最后在主电路中详细测试自己封装的 CPU。CPU 的 8 条核心指令如表 2.1 所示。

表 2.1 MIPS 指令集

#	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]+SignExt16b(imm) 溢出产生异常
4	lw $rt,imm($rs)	R[$rt]←Mem4B(R[$rs]+SignExt16b(imm))
5	sw $rt,imm($rs)	Mem4B(R[$rs]+SignExt16b(imm))←R[$rt]
6	beq $rs,$rt,imm	if(R[$rs] = R[$rt]) PC ← PC + SignExt18b({imm, 00})
7	bne $rs,$rt,imm	if(R[$rs] != R[$rt]) PC ← PC + SignExt18b({imm, 00})
8	syscall	系统调用，这里用于停机

在 CPU 中，控制信号需要通过控制器生成。要求根据 MIPS CPU 对控点的要求，分别设计一个多周期 MIPS CPU 硬布线控制器和一个多周期 MIPS CPU 微程序控制器。

对于多周期硬布线控制器，有传统三级时序和现代时序两种形式的控制器，由于传统三级时序的硬布线控制器指令周期等长、慢，本次实验采用指令周期可变且更加复杂的现代时序的方式设计多周期硬布线控制器，需要设计对应的控制器状态机与硬布线控制器的组合逻辑。

对于多周期微程序控制器，需要设计对应的微指令与地址转移逻辑。

硬布线与微程序控制器包含的输出信号相同，具体有：多路选择器选择信号，内存访问控制信号，寄存器写使能信号，运算器控制信号、指令译码信号等。每种控制信号的具体功能说明见表 2.2 所示。

表 2.2 控制信号功能说明

#	控制信号	信号说明	产生条件
1	PCWrite	PC 写使能控制	取指令周期，分支指令执行
2	IorD	指令还是数据	0 表示指令，1 表示数据
3	IRwrite	指令寄存器写使能	高电平有效
4	MemWrite	写内存控制信号	sw 指令
5	MemRead	读内存控制信号	Lw 指令 取指令
6	Beq	Beq 指令译码信号	Beq 指令
7	Bne	Bne 指令译码信号	Bne 指令
8	PcSrc	PC 输入来源	顺序寻址还是跳跃寻址
9	AluOP	运算器操作控制符 4 位	ALU_Control，00 加，01 减，10 由 Funct
10	AluSrcA	运算器第一输入选择	
11	AluSrcB	运算器第二输入选择	Lw 指令，sw 指令，addi
12	RegWrite	寄存器写使能控制信号	寄存器写回信号
13	RegDst	写入寄存器选择控制信号	R 型指令
14	MemToReg	写入寄存器的数据来自存储器	lw 指令

2.2 方案设计

2.2.1 多周期硬布线 CPU 设计

设计原理：多周期 CPU 指的是将整个 CPU 的执行过程分成几个阶段，每个阶段用一个时钟去完成，然后开始下一条指令的执行，而每种指令执行时所用的时钟数不尽相同。CPU 在处理指令时，一般需要经过以下无个阶段中的某几个阶段：取指令、指令译码、指令执行、存储器访问、结果写回。相比单周期 CPU 而言，多周期 CPU 的数据通路中，不再区分指令存储器和数据存储器，分时使用部分功能部件；主要功能单元输出端增加寄存器锁存数据；传输通路延迟变小，时钟周期变短。

设计多周期 CPU 时应也为每条指令设计对应的数据通路并合并出总的数据通路。数据通路设计完成后可尝试从指令存储器中取得指令并执行，从而验证多周期

CPU 数据通路的正确性。

设计思路：设计多周期 CPU 首先应选定指令系统，用 RTL 表示指令功能；根据指令功能设计功能部件与通路，互连各部件；确定各个部件需要的控制信号；汇总控制信号分析指令信号序列；根据指令与控制信号的关系表设计控制器。将控制器用于设计好的数据通路中便可以绘制完成多周期 CPU 的逻辑电路图。

设计过程：

(1) 选定指令系统：选用 MIPS 的指令系统，直接使用指令集中已设计好的 RTL；

(2) 设计部件与通路：以 R 型指令 ADD 和 LW 指令为例，设计八条核心指令对应的数据通路:

区别于单周期 CPU，多周期 CPU 中每条指令的不同状态都需要设计对应的数据通路，即相比单周期 CPU，多周期 CPU 的设计变得更加精细和复杂了。所有指令执行的前两个状态都分别为取指令和指令译码，先为这两个操作设计数据通路。

a．取指令和指令译码数据通路的设计

T1 为取指令阶段，其 RTL 功能描述为 Mem[PC]→IR，PC+4→PC。需要执行的操作分别是：将内存中的指令读到 IR 寄存器中；通过 ALU 将 PC+4 并保存在 PC 寄存器中。T1 取指令阶段的数据通路如图 2.1 所示。

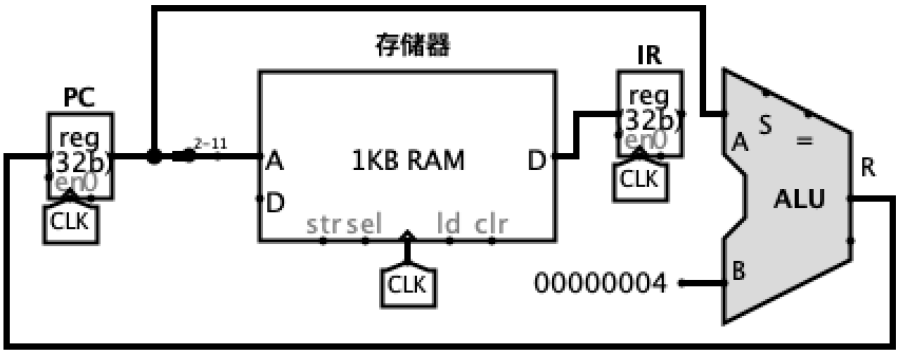


图 2.1 T1 取指令数据通路图

T2 为指令译码阶段，其 RTL 功能描述为 Reg→A、B，PC+4+Imm16<<2→C。需要执行的操作分别是：将 IR 寄存器中的指令作为 OP 输入控制器中,同时根据 IR 寄存器中的指令取出 Regfile 寄存器组中的值到锁存器 A、B 中；通过 ALU 将 PC 再于左移二位后的立即数进行加运算，结果保存在 C 锁存器中。T2 取指令阶段的数据通路如图 2.2 所示。

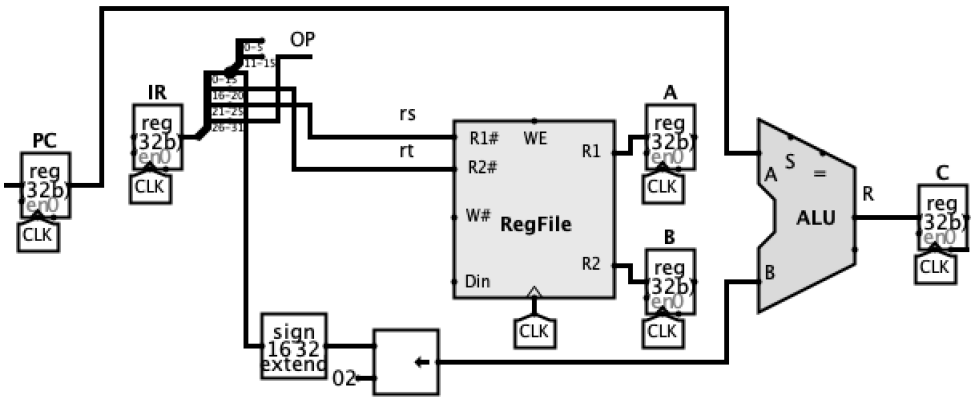


图 2.2 T2 指令译码数据通路图

经过取指令与指令译码两个时钟周期后，指令通过控制器译码得到了相应的控制信号，寄存器中的值被取到了 A、B 两个锁存器中，PC+4+Imm16<<2 的运算结果存储在了锁存器 C 中，从 T3 时钟周期开始能够执行一条指令的核心部分。

b．ADD 指令数据通路的设计

ADD 指令为 R 型指令，其 RTL 功能描述为 R[$rd]←R[$rs]+R[$rt]。在 T3 时，A、B 两个锁存器中的值通过 ALU 进行相加，运算结果存储在锁存器 C 中；在 T4 时，从 IR 寄存器中读取 Rd 寄存器的地址，并将锁存器 C 中的数据作为 Din 输入写回到 Regfile 寄存器组中，至此完成 ADD 指令。ADD 指令数据通路如图 2.3 所示。

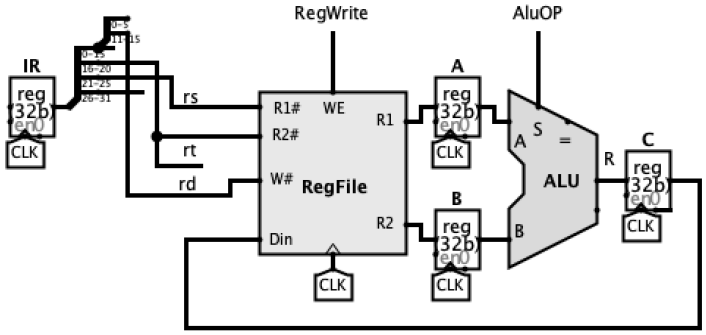


图 2.3 T3、T4 周期 R 型指令 ADD 数据通路图

c．LW 指令数据通路的设计

LW 指令为 I 型指令，RTL 功能描述为 R[$rt]←Mem4B(R[$rs]+SignExt16b(imm))。在 T3 时，锁存器 A 的值与位拓展后的立即数进行求和运算，结果存储在寄存器 C 中，此时算出了操作数地址；在 T4 时，锁存器 C 中的值作为地址用于访问内存，从内存中取得的数据被存储在 DR 数据寄存器中；在 T5 时，从 IR 寄存器中读取 Rd 寄存器的地址，并将 DR 数据寄存器中的数据作为 Din 输入写回到 Regfile 寄存器中，

至此完成 LW 指令。LW 指令数据通路如图 2.4 所示。

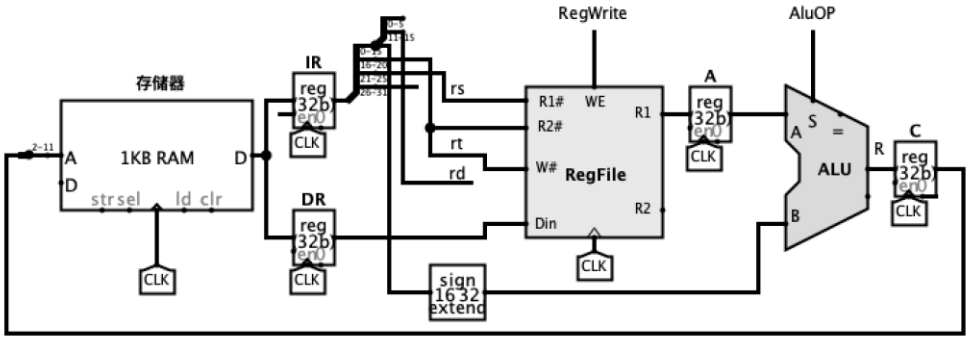


图 2.4 T3、T4、T5 周期 LW 数据通路图

类似地，对八条核心指令分别进行数据通路的设计。

(3) 设计控制信号（控点）：和单周期 CPU 相同，为了合并不同指令的数据通路，需要设计相应的控制信号（控点）控制多路选择器。控制信号能够实现对数据通路的控制，保证在多输入的情形下能够取得正确的输入信号，使得不同的指令能够在 CPU 中共享相同的器件。

(4) 构建指令状态转换图：汇总八条核心指令，根据一条指令在不同时钟周期内的行为不同，将每条指令拆分为多个状态，从而构建指令状态转换图。状态图中包含每条指令所对应的多个状态需要进行的操作，并描述了状态是如何跳转的，如图 2.5 所示。

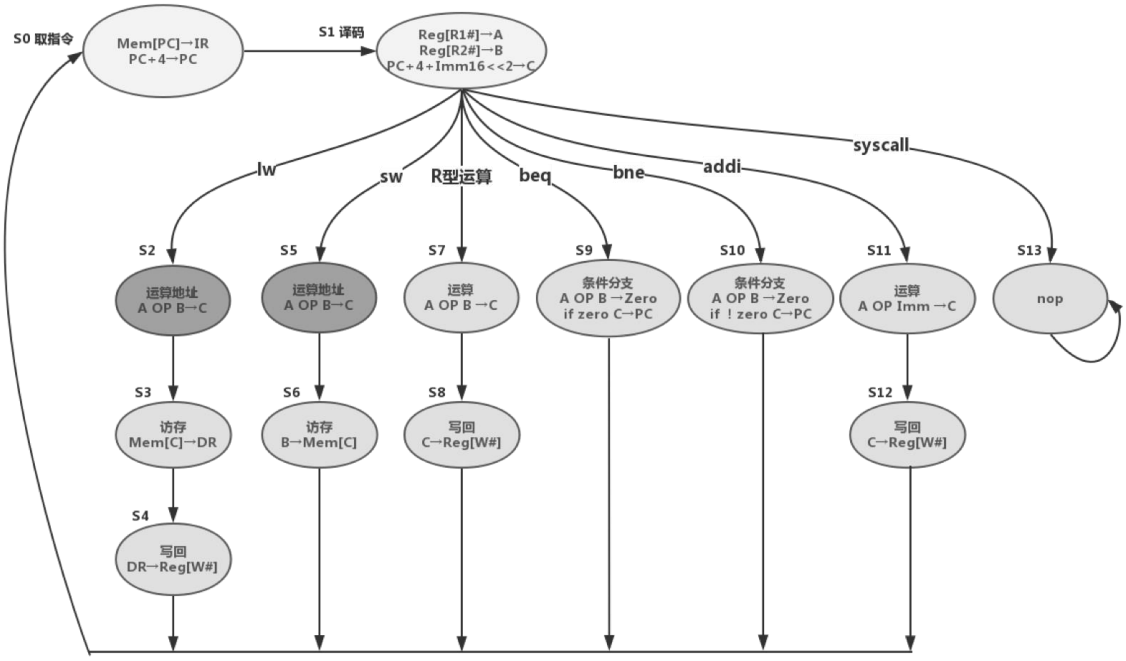


图 2.5 指令状态转换图

(5) 设计控制器：根据指令状态转换图中对每个指令状态需要进行的操作的描述，可以汇总出指令状态与控制信号关系表。根据关系表，能够生成控制信号的逻辑表达式从而设计硬布线控制器，或设计微指令从而设计微程序控制器。多周期硬布线控制器和多周期微程序控制器将在 CPU 中根据输入信号的不同输出对应的控制信号，实现对 CPU 数据通路的控制。多周期硬布线控制器的具体设计过程见 2.2.2 小节，多周期微程序控制器的具体设计过程见 2.2.3 小节。

多周期 MIPS CPU 设计完成后结构如图 2.6 所示。

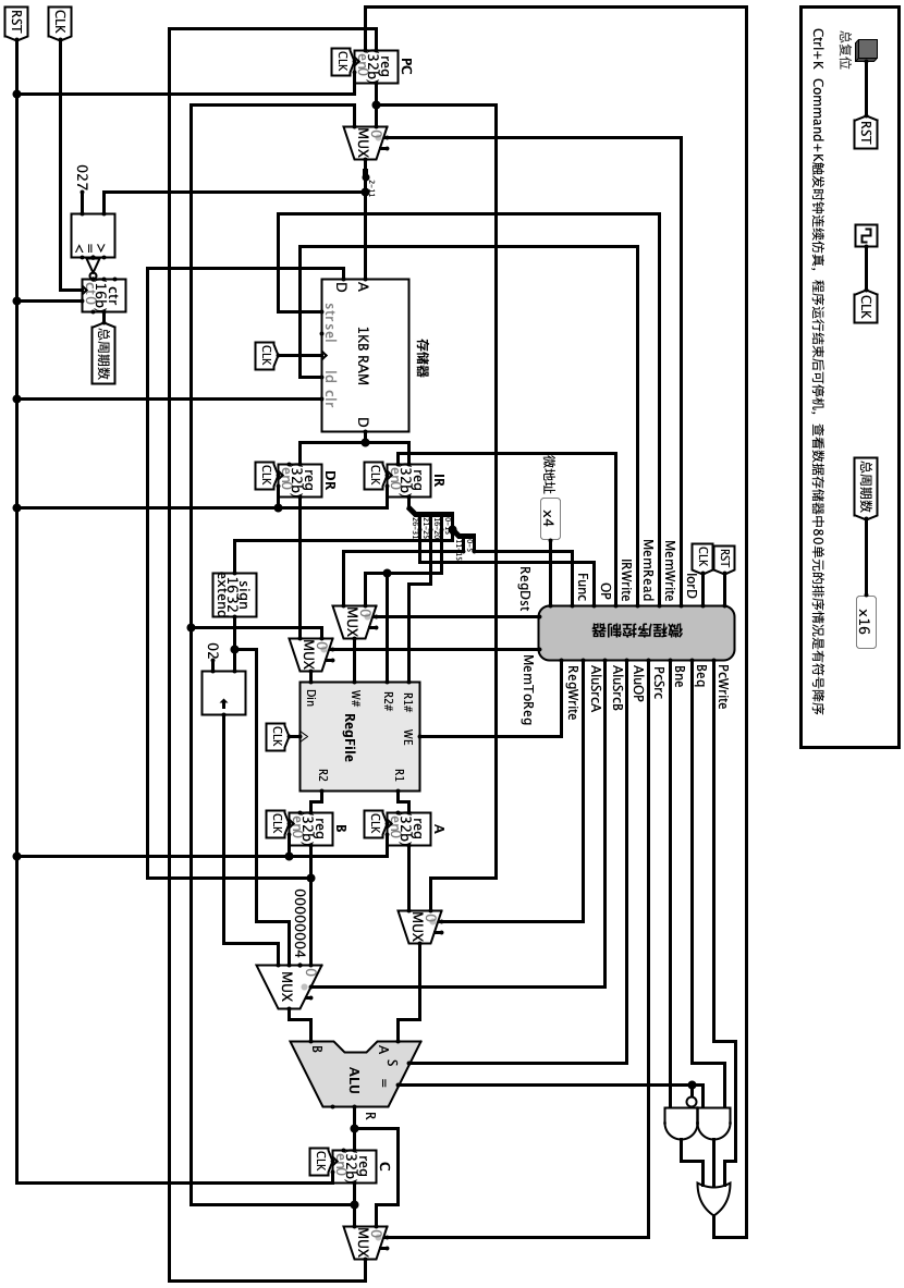


图 2.6 多周期 MIPS CPU 结构图

2.2.2 多周期硬布线控制器设计

设计原理：CPU 中的控制器器件用于根据输入信号译码出指令并产生对应的控制信号序列，即控制器是产生固定时序控制信号的逻辑电路。传统三级时序的硬布线控制器不够灵活，本次实验采用现代时序的硬布线控制器。对于硬布线控制器，一条指令中的每一个周期对应一个状态，由一个状态对应一组并发的控制信号，从而实现控制信号的输出。

多周期硬布线控制器原理图如图 2.7 所示

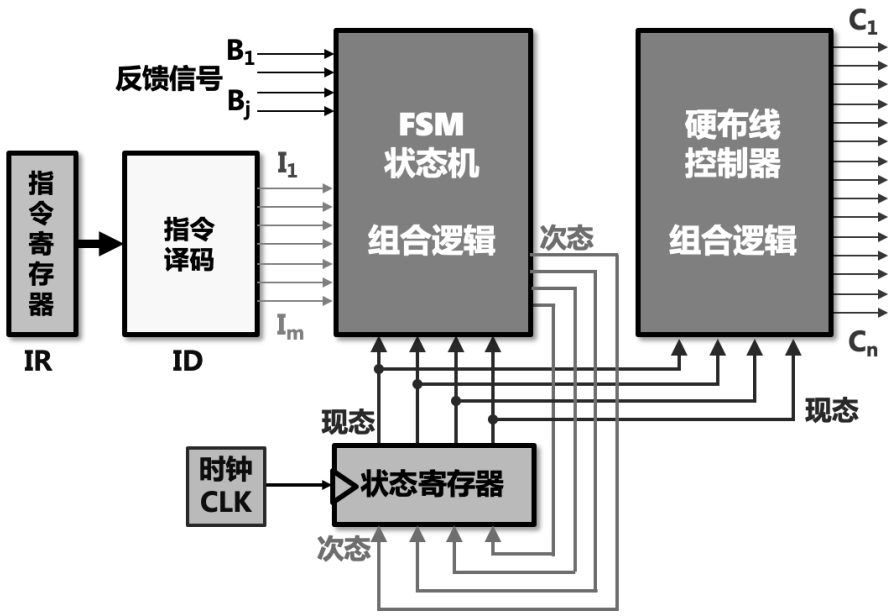


图 2.7 多周期硬布线控制器原理图

设计思路：对输入信号 OP 和 FUNC 进行指令译码，得到具体指令；为实现将指令中的每一个周期对应一个状态，需要根据状态转换图构建状态机真值表，从而实现有限状态机组合逻辑；为实现将每一个状态对应一组对应的并发信号，需设计对应的硬布线控制器组合逻辑电路。

设计过程：

(1) 指令译码：通过比较器将输入信号 OP 与根据 MIPS 指令集预设的常数值进行比较，从而判断指令类型。若属于 R 型指令，继续与输入信号 FUNC 进行比较，从而判断具体的 R 型指令类型。

(2) 设计 FSM 状态机组合逻辑：利用状态转换图，构建对应的状态机真值表，如表 2.3 所示。

表 2.3 状态机真值表

现态10进制	S3	S2	S1	S0	R_Type	LW	SW	BEQ	BNE	SYSCALL	ADDI	次态10进制	N3	N2	N1	N0
0	0	0	0	0	X	X	X	X	X	X	X	1	0	0	0	1
1	0	0	0	1	1	X	X	X	X	X	X	7	0	1	1	1
1	0	0	0	1	X	1	X	X	X	X	X	2	0	0	1	0
1	0	0	0	1	X	X	1	X	X	X	X	5	0	1	0	1
1	0	0	0	1	X	X	X	1	X	X	X	9	1	0	0	1
1	0	0	0	1	X	X	X	X	1	X	X	10	1	0	1	0
1	0	0	0	1	X	X	X	X	X	1	X	13	1	1	0	1
1	0	0	0	1	X	X	X	X	X	X	1	11	1	0	1	1
2	0	0	1	0	X	1	X	X	X	X	X	3	0	0	1	1
3	0	0	1	1	X	1	X	X	X	X	X	4	0	1	0	0
4	0	1	0	0	X	1	X	X	X	X	X	0	0	0	0	0
5	0	1	0	1	X	X	1	X	X	X	X	6	0	1	1	0
6	0	1	1	0	X	X	1	X	X	X	X	0	0	0	0	0
7	0	1	1	1	X	X	X	1	X	X	X	8	1	0	0	0
8	1	0	0	0	X	X	X	1	X	X	X	0	0	0	0	0
9	1	0	0	1	X	X	X	X	1	X	X	0	0	0	0	0
10	1	0	1	0	X	X	X	X	X	1	X	0	0	0	0	0
11	1	0	1	1	X	X	X	X	X	X	1	12	1	1	0	0
12	1	1	0	0	X	X	X	X	X	X	1	0	0	0	0	0
13	1	1	0	1	1	X	X	X	X	X	X	13	1	1	0	1

根据真值表得到对应的表达式分别为：

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 R_Type + S3 ~S2 S1 S0 ADDI + S3 S2 ~S1 S0 SYSCALL;

N2= ~S3 ~S2 ~S1 S0 R_Type + ~S3 ~S2 ~S1 S0 SW + ~S3 ~S2 ~S1 S0 SYSCALL + ~S3 ~S2 S1 S0 LW + ~S3 S2 ~S1 S0 SW + S3 ~S2 S1 S0 ADDI + S3 S2 ~S1 S0 SYSCALL;

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 LW + ~S3 S2 ~S1 S0 SW;

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 LW + S3 S2 ~S1 S0 SYSCALL.

利用 logisim 的构建电路功能，输入表达式，得到对应的 FSM 有限状态机组合逻辑，这样就实现了将指令中的每一个周期与一个状态相对应。

(3) 设计硬布线控制器组合逻辑：通过组合逻辑为每个状态设置一组对应的并发信号，作为控制信号输出。在本实验中为减少工作量，方便起见，使用微程序控制器中的 ROM 代替组合逻辑，实现一个状态到一组控制信号之间的映射关系，从而

控制器能够正确译码并产生相应的控制信号。

多周期硬布线控制器设计完成后结构图如图 2.8 所示。

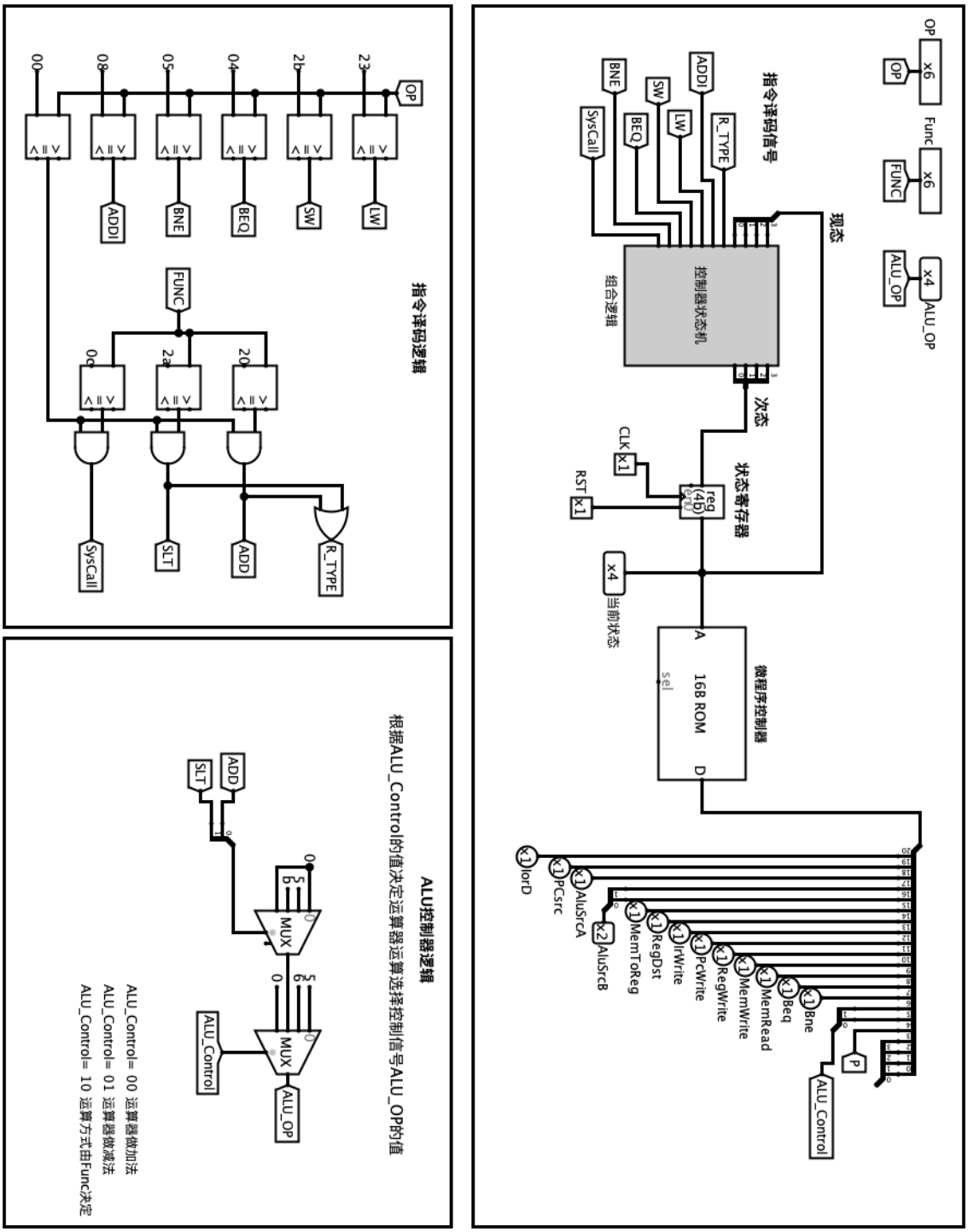


图 2.8 多周期硬布线控制器结构图

2.2.3 多周期微程序控制器设计

设计原理：微程序是利用软件方法来设计硬件的技术。在微程序控制器中，需要将完成指令所需的控制信号按格式编写成微指令，存放到控制存储器。其中，一条机器指令对应一段微程序（多条微指令）。微程序控制器将存储技术和程序设计相结合，回避复杂的同步时序逻辑设计。

相比于硬布线控制器，在微程序控制器中：一条指令对应多条微指令；状态等同与存储器地址。

多周期微程序控制器原理图如图 2.9 所示。

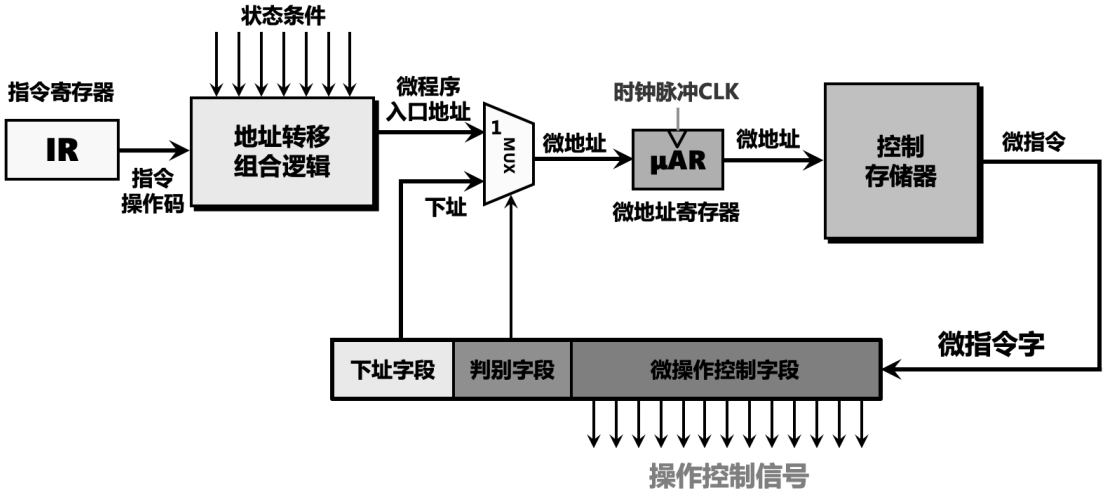


图 2.9 多周期微程序控制器原理图

设计思路：对输入信号 OP 和 FUNC 进行指令译码，得到具体指令；为实现将指令中的每一个周期对应一条微指令，需要根据状态转换图构建微程序，从而实现微程序控制器，微程序控制器中的每一条微程序直接对应一组控制信号；为实现将微程序地址的转移，需构建相应的真值表设计地址转移逻辑。

设计过程：

(1) 指令译码：通过比较器将输入信号 OP 与根据 MIPS 指令集预设的常数值进行比较，从而判断指令类型。若属于 R 型指令，继续与输入信号 FUNC 进行比较，从而判断具体的 R 型指令类型。

(2) 构建微程序控制器：利用状态转换图以及图中每个周期对应的操作所需信号，设计对应的微程序，如表 2.4 所示。

表 2.4 微指令设计表

微指令功能	状态	微指令地址	MemRead	BEQ	BNE	AluControl	P	下址字段	微指令	十六进制
取指令	0	0000	1	0	0	00	0	0001	00001001100100000001	13201
译码	1	0001	0	0	0	00	1	0000	00011000000000010000	30010
LW1	2	0010	0	0	0	00	0	0011	00110000000000000011	60003
LW2	3	0011	1	0	0	00	0	0100	10000000001000000100	100204
LW3	4	0100	0	0	0	00	0	0000	00000100010000000000	8800
SW1	5	0101	0	0	0	00	0	0110	00110000000000000110	60006
SW2	6	0110	0	0	0	00	0	0000	10000000010000000000	100400
R型运算1	7	0111	0	0	0	10	0	1000	00100000000001001000	40048
R型运算2	8	1000	0	0	0	00	0	0000	00000100100000000000	4800
Beq	9	1001	0	1	0	01	0	0000	01100000000100100000	C0120
Bne	10	1010	0	0	1	01	0	0000	01100000000010100000	C00A0
ADDI1	11	1011	0	0	0	00	0	1100	00110000000000001100	6000C
ADDI2	12	1100	0	0	0	00	0	0000	00000000100000000000	800
SYSCALL	13	1101	0	0	0	00	0	1101	00000000000000001101	D

将设计好的微程序存入 ROM 中，微程序控制器便构建完成。这样就实现了每一个状态与一条微指令相对应。

(3) 构建地址转移逻辑：需要将不同指令与对应的微程序路口地址通过组合逻辑的方式联系起来，使得对每条指令，控制器能够正确得寻找到第一条对应的微程序。地址转移逻辑真值表如表 2.5 所示。

表 2.5 地址转移逻辑真值表

R_Type	ADDI	LW	SW	BEQ	BNE	SYSCALL	微程序入口地址（10进制）	S3	S2	S1	S0
1	X	X	X	X	X	X	7	0	1	1	1
X	1	X	X	X	X	X	11	1	0	1	1
X	X	1	X	X	X	X	2	0	0	1	0
X	X	X	1	X	X	X	5	0	1	0	1
X	X	X	X	1	X	X	9	1	0	0	1
X	X	X	X	X	1	X	10	1	0	1	0
X	X	X	X	X	X	1	13	1	1	0	1

通过 logisim 中描述构建电路的功能，使用真值表构建微程序地址转移逻辑。

至此多周期微程序控制器设计完成，多周期微程序控制器设计完成后结构图如图 2.10 所示。

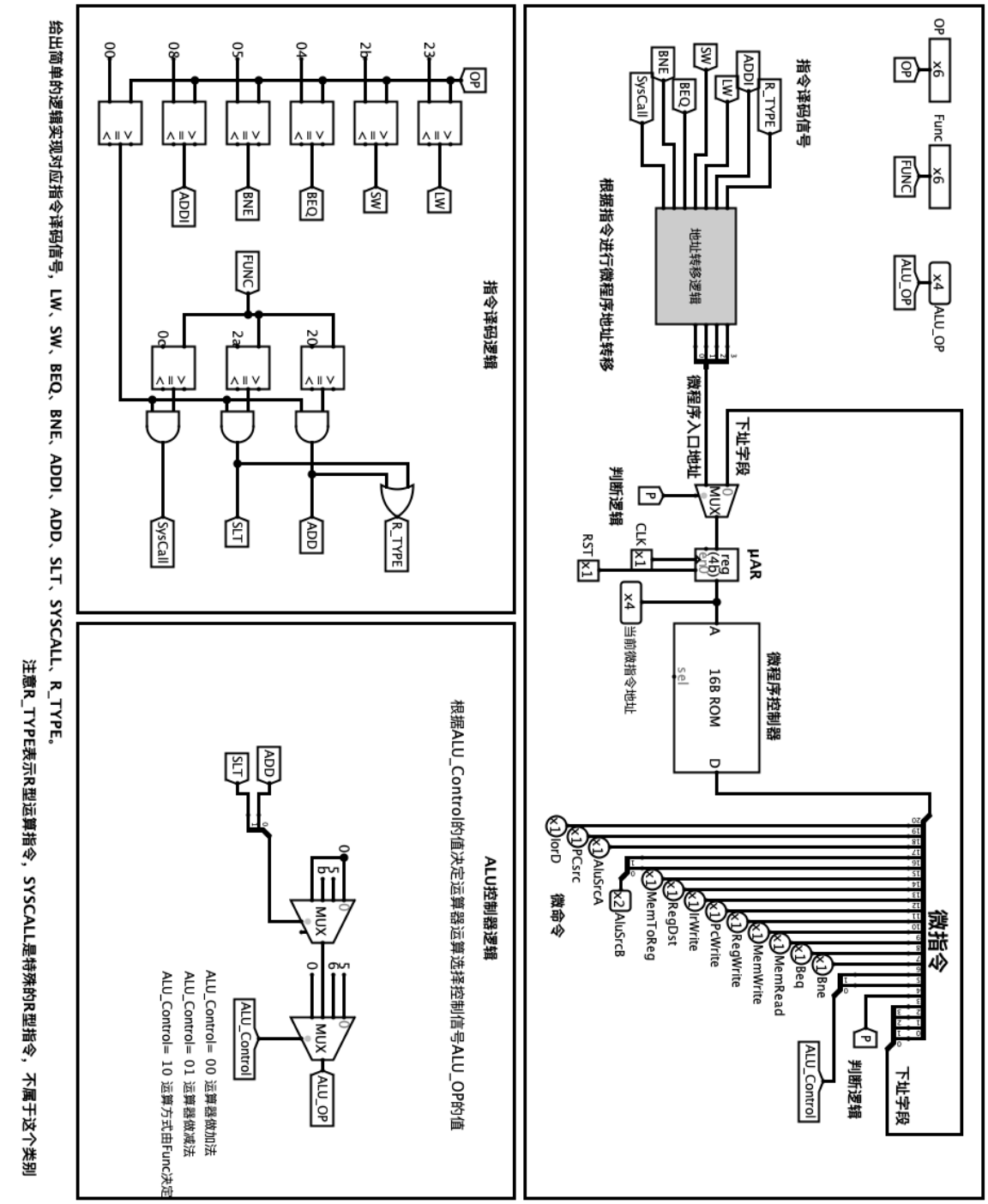


图 2.10 多周期微程序控制器结构图

2.3 实验步骤

（1） 根据 2.2.1 中多周期 MIPS CPU 的设计过程，在 logisim 平台上绘制多周期 CPU 的数据通路。

（2） 根据 2.2.2 中多周期硬布线控制器的设计过程，完成多周期硬布线控制

器内部电路的设计与绘制。

（3） 根据 2.2.3 中多周期微程序控制器的设计过程，完成多周期微程序控制器内部电路的设计与绘制。

（4） 用冒泡排序分别测试使用硬布线控制器和微程序控制器的电路。

2.4 故障与调试

2.4.1 ALU 输入有误

故障现象：执行冒泡排序测试指令后，排序结果出现在错误的地址处，且数据的值不符合预期。

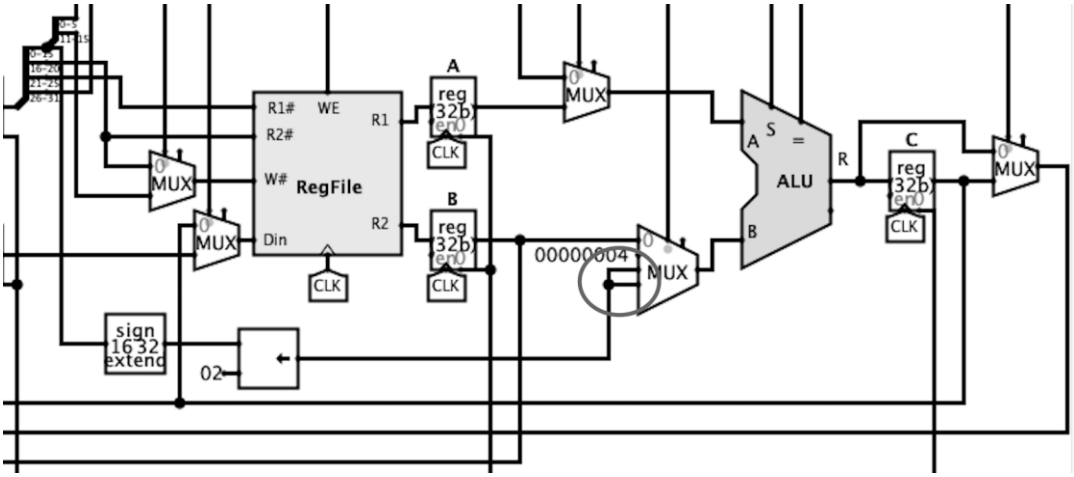


图 2.11 数据通路中 ALU 输入错误图

原因分析：如图 1.所示，在设计多周期 CPU 的数据通路时，ALU 的 B 输入端需要利用 ALUSrcB 控制多路选择器进行输入信号的选择，但可选输入信号的设置出现了错误，在 ALUSrcB 为 10 时，应当直接取得有符号拓展后的立即数，图中错接为了移位后的立即数，从而 CPU 无法正常工作。

解决方案：改正 ALU 的 B 端输入来源，在 ALUSrcB 为 10 时，直接从有符号位拓展后的立即数进行输入。

2.4.2 周期计数无法停止

故障现象：在对 CPU 运行周期进行计数时，无法停止计数。

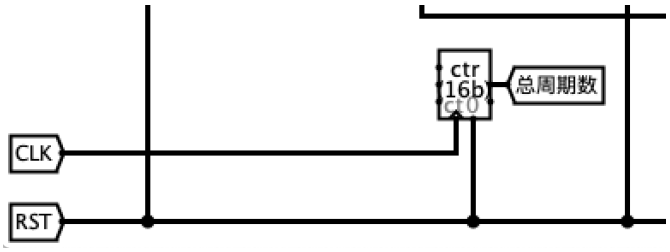


图 2.12 计数器设置错误图

原因分析：如图 2.12，在对 CPU 的时钟周期进行计数时，不同于单周期 CPU，多周期 CPU 的控制器中没有设置 Halt 信号，使得时钟周期计数时无法利用控制器的输出信号控制使能端实现停止计数。

解决方案：CPU 在执行到 SysCall 指令时将停在该处，此时添加一个比较器用于比较 CPU 是否执行到 SysCall 处即可判断 CPU 停止工作与否，将比较器的输出信号用于时钟周期计数器的使能端即可控制计数的停止。

2.4.3 使用微程序控制器时控制信号产生有误

故障现象：在使用单条指令测试多周期微程序 CPU 时，指令无法正确工作。

原因分析：在分别检查了地址转移逻辑与微程序控制器 ROM 后，发现通过表达式生成的地址转移逻辑有一定问题。同时微程序控制器中，LW 指令对应的微指令设计有细微的错误，这都导致了控制信号的异常。

解决方案：地址转移逻辑部分的组合逻辑电路改为使用真值表生成，修改了 LW 指令对应的微指令，多周期 CPU 正常工作。

2.5 测试与分析

使用冒泡排序指令对使用硬布线控制器的多周期 CPU 进行测试，排序结果见图 1.所示 。

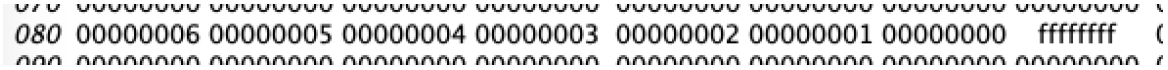


图 2.13 冒泡排序测试多周期硬布线 CPU 结果

使用冒泡排序指令对使用微程序控制器的多周期 CPU 进行测试，排序结果见图 1.所示 。

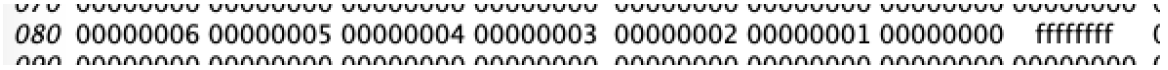


图 2.14 冒泡排序测试多周期微程序 CPU 结果

3 总结与心得

3.1 实验总结

在实验一数据表示实验中，主要完成了如下几点工作：

1） 设计了国标码转区位码的电路；根据输入的字符自动转换为 GB2312 编码并输出为文件的程序，且 logisim 中的 ROM 器件能够直接读入生成的文件。经过以上设计，实现了轮流将 ROM 中的国标码转换为区位码并显示在点阵上的电路。

2） 设计了偶校验编码电路与偶校验检错电路，其中偶校验检错电路只能检错奇数个错，偶数错无法被检测。经过以上设计，实现了偶校验编解码传输电路。

3） 设计了海明编码电路与海明解码电路，其中海明码解码电路能够纠一位错并检测出两位错。经过以上设计，实现了海明编解码传输电路。

4） 设计了 CRC 编码电路与 CRC 解码电路，其中 CRC 解码电路能够纠一位错并检测处两位错。经过以上设计，实现了 CRC 编解码传输电路。

在实验二运算器实验中，主要完成了如下几点工作：

1） 利用 1 位全加器设计了 8 位可控加减法器。

2） 利用门电路设计了 4 位先行进位电路 74182。

3） 利用设计的 74182 先行进位电路设计了 4 位快速加法器。

4） 利用设计的 74182 和 4 位快速加法器设计了 16 位快速加法器。

5） 利用设计的 74182 和 4 位快速加法器设计了 32 位快速加法器。

6） 设计了 5 位阵列乘法器。

7） 利用 5 位阵列乘法器设计了 6 位补码阵列乘法器。

8） 利用 8 位串行加法器设计了原码一位乘法器。

9） 利用 8 位串行加法器设计了补码一位乘法器。

10） 利用 32 位快速加法器设计了 32 位 ALU。ALU 中实现了逻辑左右移、算数右移、与、或、或非、异或、加减乘除、有符号比较、无符号比较等运算功能。

在实验三存储系统实验中，主要完成了如下几点工作：

1） 利用 ROM 器件和字库文件设计了字库电路。实现了 GB2312 编码数据通过字库输出为点阵文字的功能。

2） 利用 4 块 logisim 自带的 RAM 器件设计了 32 位的 MIPS RAM 器件。实现了能够分别按字节、字、双字读或写 RAM 的功能。

3） 利用 4 个 logisim 自带的 register 器件设计了 32 位的 MIPS Regfile 器件。实现了根据寄存器地址对 Regfile 器件中的某个寄存器进行读或写的功能。

4） 利用 3 位数据比较电路 Max3 设计了全相连 cache。实现了将内存中数据正确读入 cache、正确寻找到 cache 中指定数据、正确淘汰 cache 中指定数据的功能。

在实验四 CPU 实验中，主要完成了如下几点工作：

1） 设计了单周期CPU的数据通路和单周期硬布线控制器。实现了单周期MIPS CPU。完成了用单周期 CPU 实现冒泡排序的测试。

2） 设计了多周期 CPU 的数据通路和多周期硬布线控制器。实现了多周期硬布线 MIPS CPU。完成了用多周期硬布线 CPU 实现冒泡排序的测试。

3） 设计了多周期 CPU 的数据通路和多周期微程序控制器。实现了多周期微程序 MIPS CPU。完成了用多周期微程序 CPU 实现冒泡排序的测试。

3.2 实验心得

经过本门实验课程后，我有了这样一些体会：

1） 不同课程之间联系紧密，学习过程中无论是对哪一门课程的哪一个方面都不能马虎。以数字逻辑和组成原理两门课程为例，在数字逻辑中学习到的所有与数字电路有关的知识都会在组成原理中作为基础知识得以运用。而组成原理课程中学习到的知识在以后包括课程设计在内的很多课程中也会被需要，不同的课程之间实际上环环相扣，对待任何一环都不能马虎。另外对于操作系统中的存储部分，组成原理中有关 RAM 和 cache 的知识也有不少相同之处，保持认真对待每一门课程的态度，往往能事半功倍。

2） 将组成原理理论课程中的书面知识以实验的方式呈现出来，切实体会到组成原理这门课程中的知识在实际生产中的运用。通过实验的方式深刻地了解了数字电路这一技术是如何被用于包括信号编码解码、字符编码、计算机内各种器件（运算器、寄存器组、RAM、cache、CPU 等）在内的各种方面，体会到了计算机组成原理这一课程的魅力。

3） 动手实践才能更好地消化知识。在理论课程的学习过程中并没有感觉到太大的困难，但着手进行实验时常常无从下手，逐步发现各种细节上的困难无法轻易解决。在实验过程中发现了问题以后往往要反复研究课内知识，这样以后问题才能被解决。总的来说，通过组成原理实验课程，更加深入的领会了书本上的知识：为什么这样设计、如何设计等一系列问题都在实践中得到了解决，实验课程在很大程度上巩固了理论课的学习成果。

在本门实验课程的学习过程中，我有如下的收获：

1） 要善于使用合适的方式解决不同的问题。在字库实验中要求为 ROM 输入 100 个字符以上的内容测试编码转换电路的正确性，如果手动往 logisim 的 ROM 中一个个输入字节显得过于繁琐而低效。此时编写代码完成这项工作才是最合适的方法，当然编写代码时需要更加仔细地学习 ASCII 码与 GB2312 编码，从而能够实现他们之间的转换。

2） 在通过逻辑电路设计器件时，不能只以完成功能为目标，需要同时考虑电路的代价。如在 32 位 ALU 运算器的设计过程中，对于加减法功能只需要一个 32 位加法器便可完成目的，如果此时为加减两个运算分别提供一个加法器就显得过于浪费。总之在设计器件时应该尝试以最低成本进行设计。

3） 在设计器件的过程中，应该尽量使用规范的设计方法而不是使用拼凑的方式达到功能要求。如在 MIPS RAM 的设计过程中，起初我使用了大量的多路选择器实现对字节、字、双字的选择以及对他们进行的读写操作。但仔细思考后，其实通过真值表生成组合逻辑来控制每块 RAM 上的片选信号就能很轻松的选择不同长度的数据，修改设计方法后节省了大量多路选择器，同时设计电路也得以简化。总的来说，根据规范的方法（即分析需求并通过真值表、卡诺图得到表达式或直接借助工具生成电路）进行设计是一个优秀的设计所必不可少的。

4） 想要顺利完成实验，一定要首先对理论知识有深入理解。在进行设计 CPU 的实验过程中，PPT 与 MOOC 的解说十分细致，数据通路能够直接采用课上学习过的例子，对于比较繁琐的表达式计算也有相应的工具进行辅助，所以完成实验并不算困难。但如果仅仅是照搬了课件上的 CPU 数据通路并跟随 MOOC 的解说得到了控制器中对应的表达式，在攥写报告的时候便一定会感到力不从心，每个知识点稍微深究就会发现自己的理解存在问题。总之要做好一次实验，对理论知识的认真学

习一定是必不可少的，这种时候按部就班能够带来不错的学习效率。

5） 在设计器件时尽量采用模块化的思想。如在进行 cache 设计实验过程中，对于 cache 槽而言，每一行都是完全相同的。如果不进行封装直接进行复制，设计界面会显得比较臃肿而混乱，此时如果对 cache 行进行封装，电路将变得更加美观整洁。同时善于将一个需求拆分为多个模块也能很好的提高设计效率并方便在设计完成后进行检查。

对本门课程的建议：

1） 实验内容大多无法在课程学时内完成，需要耗费一定的课余时间。希望能对实验的分量做好控制，或者预先给同学们打好预防针。

2） 在对每次实验的内容进行检查核对的同时，希望能督促同学更高质量的完成一次对该实验总结，一次及时的总结能够有效的减少最后编写报告时回顾实验的工作量。现在根据提交审核的先后判分的机制并不利于我们在每次实验过后提交一份详细的总结。

3） 总的来说这门实验课程中教师的工作量远大于学生。实验中为了减少繁琐工作的工作量，文件里提供了诸如测试电路、电路框架、调试探针、表达式生成器等一系列工具用于提高学生的工作效率，学生实验体验极好。同时每个实验对应的微 MOOC 都适当的对实验进行了介绍，在不剧透答案的前提下也能辅助学生的思考，设置十分合理。希望这门实验课程能越办越好。

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] 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] 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.

[4] 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.

[5] 张晨曦，王志英. 计算机系统结构. 高等教育出版社，2008 年.

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: 刘逸帆

二、对课程实验的学术评语

三、对课程实验的评分

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: _____ **2018-12-24**