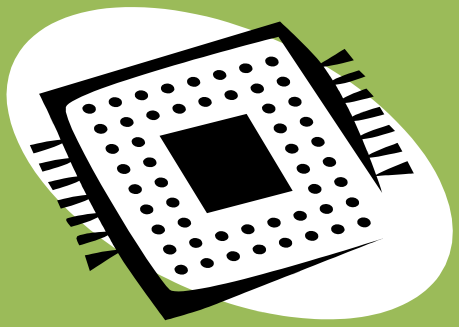
华中科技大学

2018

计算机组成原理 ·实验报告·

专　　业:	计算机科学与技术
班　　级:	IT1601
学　　号:	U201614898
姓　　名:	潘翔
电　　话:	18086010495
邮　　件:	hover@hust.edu.cn
完成日期:	2018-12-20



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 CPU（硬布线）

1) 实验目的：
   a) 能掌握硬布线控制器设计的基本原理
   b) 能利用相关原理在 Logisim 平台中实现 MIPS 单周期 CPU

2) 主要任务：
   a) 绘制 MIPS CPU 数据通路
   b) 实现单周期硬布线控制器
   c) 测试联调

1.1.2 多周期 MIPS CPU（微程序）

1) 实验目的：
   a) 掌握多周期 MIPS CPU 的设计原理
   b) 掌握微程序控制器设计的基本原理
   c) 利用微程序控制器的设计实现多周期 MIPS 处理器

2) 主要任务：
   a) 绘制多周期 MIPS CPU 数据通路
   b) 实现微程序控制器
   c) 测试联调

1.1.3 多周期 MIPS CPU（硬布线）

1) 实验目的：
   a) 掌握多周期 MIPS CPU 的设计原理
   b) 掌握硬布线控制器设计的基本原理
   c) 利用硬布线控制器的设计实现多周期 MIPS 处理器

2) 主要任务：
   a) 绘制多周期 MIPS CPU 数据通路
   b) 实现多周期硬布线控制器
   c) 测试联调

1.1.4 指令要求

利用 logisim 平台中现有运算部件构建一个 32 位运算器，可支持算数加、减、乘、除，逻辑与、或、非、异或运算、逻辑左移、逻辑右移，算术右移运算，支持常用程序状态标志（有符号溢出 OF、无符号溢出 CF，结果相等 Equal），运算器功能以及输入输出引脚见下表，在主电路中详细测试自己封装的运算器。

表 1.1 片引脚与功能描述

引脚	输入/输出	位宽	功能描述
X	输入	32	操作数 X
Y	输入	32	操作数 Y
ALU_OP	输入	4	运算器功能码，具体功能见下表
Result	输出	32	ALU 运算结果
Result2	输出	32	ALU 结果第二部分，用于乘法指令结果高位或除法指令的余数位，其他操作为零
OF	输出	1	有符号加减溢出标记，其他操作为零
CF	输出	1	无符号加减溢出标记，其他操作为零
Equal	输出	1	Equal=(x==y)?1:0, 对所有操作有效

表 1.2 运算符功能

ALU OP	十进制	运算功能
0000	0	Result = X << Y 逻辑左移 （Y 取低五位） Result2=0
0001	1	Result = X >>>Y 逻辑右移 （Y 取低五位） Result2=0
0010	2	Result = X >> Y 算术右移 （Y 取低五位） Result2=0
0011	3	Result = (X * Y)[31:0]; Result2 = (X * Y)[63:32] 有符号
0100	4	Result = X/Y; Result2 = X%Y 无符号
0101	5	Result = X + Y Result2=0 (Set OF/CF)
0110	6	Result = X - Y Result2=0 (Set OF/CF)
0111	7	Result = X & Y Result2=0
1000	8	Result = X | Y Result2=0
1001	9	Result = X ⊕ Y Result2=0
1010	10	Result = ~(X |Y) Result2=0
1011	11	Result = (X < Y) ? 1 : 0 Signed Result2=0
1100	12	Result = (X < Y) ? 1 : 0 Unsigned Result2=0
1101	13	Result = Result2=0
1110	14	Result = Result2=0
1111	15	Result = Result2=0

表 1.3 MIPS指令功能描述

MIPS指令	RTL功能描述
add $rd,$rs,$rt	R[$rd]←R[$rs]+R[$rt]　　溢出时产生异常，且不修改R[$rd]
slt $rd,$rs,$rt	R[$rd]←R[$rs]<R[$rt]　　小于置 1，有符号比较
addi $rt,$rs,imm	R[$rt]←R[$rs]+SignExt16b(imm)　　溢出产生异常
lw $rt,imm($rs)	R[$rt]←Mem4B(R[$rs]+SignExt16b(imm))
sw $rt,imm($rs)	Mem4B(R[$rs]+SignExt16b(imm))←R[$rt]
beq $rs,$rt,imm	if(R[$rs] = R[$rt])　PC ← PC + SignExt18b({imm, 00})
bne $rs,$rt,imm	if(R[$rs] != R[$rt]) PC ← PC + SignExt18b({imm, 00})
syscall	系统调用，这里用于停机

1.2 方案设计

1.2.1 单周期硬布线 CPU

1) 指令实现

系统整体架构图如 图 1.1 单周期硬布线总体设计图 所示

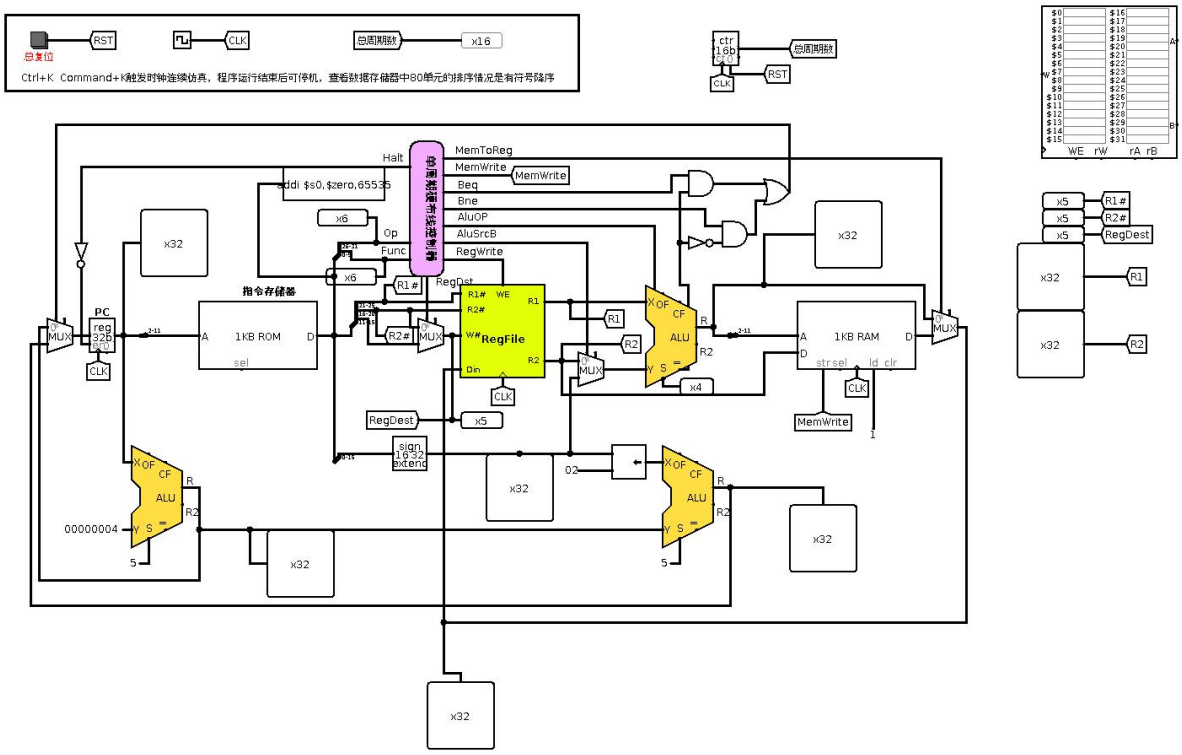


图 1.1 单周期硬布线总体设计图

需要实现:

R 型指令：ADD，SLT

I 型指令：ADDI，LW，SW，BEQ，BNE

程序中断指令：SYSCALL

译码及数据通路

1) 取指令数据通路

a. 跳转指令

若有跳转指令（BNE 和 BEQ），程序需要根据指令所指的位置跳转；若无跳转指令，程序执行需要所指位置继续加 4(32 位)。需要实现一个基本的跳转指令和非跳转指令的控制逻辑，来根据不同的指令选择执行。使用‘J’来表示指令是否是控制指令，然后根据‘J’来选择不同的指令地址。

实现如 图 1.2 跳转指令电路设计图 所示：

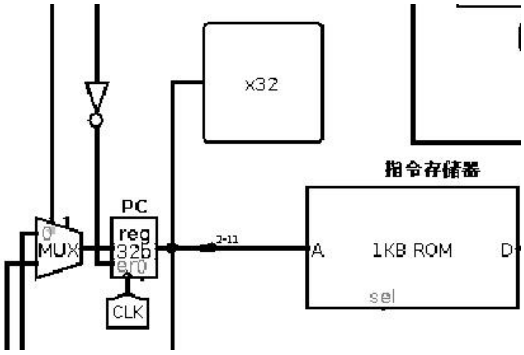


图 1.2 跳转指令电路设计图

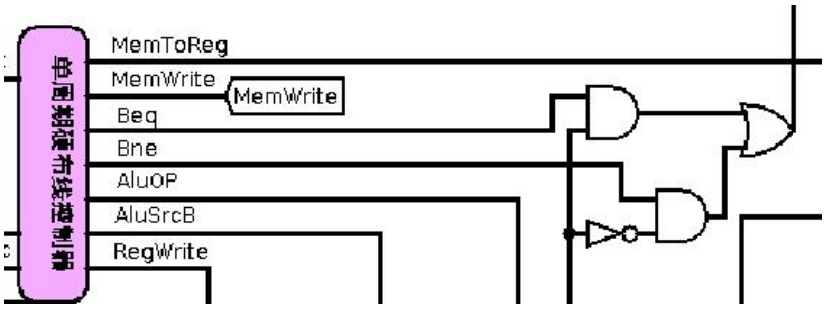


图 1.3 跳转信号生成图

b. R 型指令和 I 型指令译码电路

将取出的指令按照 R 型指令和 I 型指令分别进行译码。

实现逻辑如 图 1.4 R 型指令和 I 型指令译码图 所示：

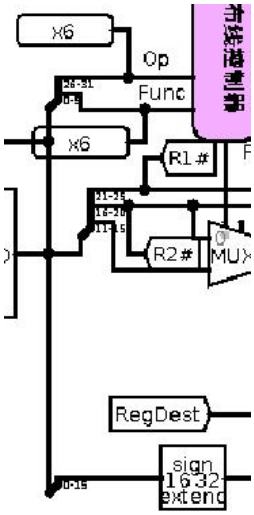


图 1.4 R 型指令和 I 型指令译码图

c. R 型指令和 I 型指令数据通路

R 型指令所有的操作数均是寄存器，执行过程中涉及的功能部件包括寄存器和 ALU，完成两个数的运算操作并将结果保存下来。

I 型指令需要计算访存地址，访存地址等于变址寄存器的值加上 16 位立即数，地址运算通过 ALU 完成，两者运算之后得到最终的访存地址然后进行后续操作。

设计的数据通路如 图 1.5 R 型指令和 I 型指令数据通路图 所示：

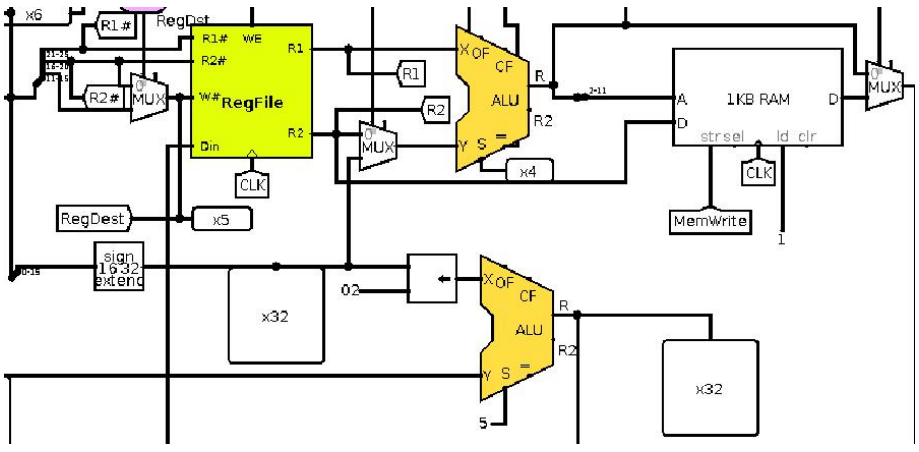


图 1.5 R 型指令和 I 型指令数据通路图

2) 单周期 MIPS 控制器

由分线器和控制器实现指令的译码及生成的状态信号传递。单周期控制器无时序逻辑，纯组合逻辑电路。

a) 输入信号

i. 指令字 Opcode，Func 字段（12 位）

b) 输出信号

i. 多路选择器选择信号

ii. 内存访问控制信号

iii. 寄存器写使能信号

iv. 运算器控制信号、指令译码信号

设计的封装如 图 1.6 单周期控制器封装图图 1.6 单周期控制器封装图 所示：

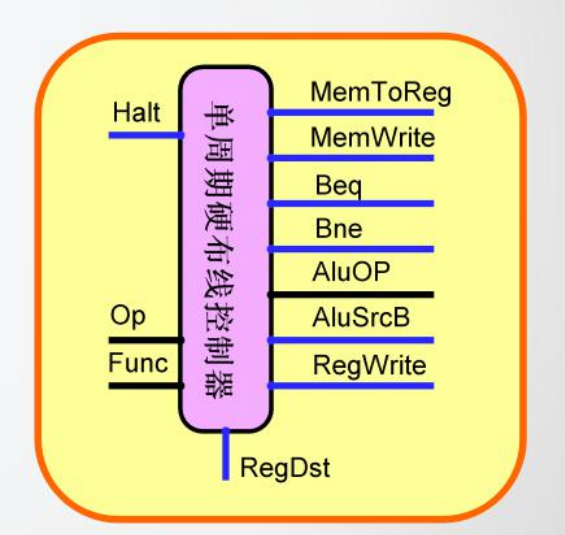


图 1.6 单周期控制器封装图

需要根据要求产生对应的控制信号。

如 表 1.4 单周期控制器指令对应控制信号表 所示:

表 1.4 单周期控制器指令对应控制信号表

#	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

R 型指令和 I 型指令对应的 OP 以及 func 如 表 1.5 R、I 型指令 OP，Func 表 所示：

表 1.5 R、I 型指令 OP，Func 表

指令名称	指令类型	OP	Func
ADD	R型	000000	20
SLT	R型	000000	2a
LW	I型	23	无
SW	I型	2b	无
BEQ	I型	04	无
BNE	I型	05	无
ADDI	I型	08	无

根据 表 1.4 单周期控制器指令对应控制信号表 和 表 1.5 R、I型指令OP，Func表 所示不同指令与不同控制信号之间的关系以及不同指令的OP，Func编码，设计单周期硬布线CPU控制器，如下 图 1.7 单周期硬布线控制器图 ，图 1.8 单周期硬布线指令译码逻辑图 ，图 1.9 单周期硬布线ALU控制器图 所示：

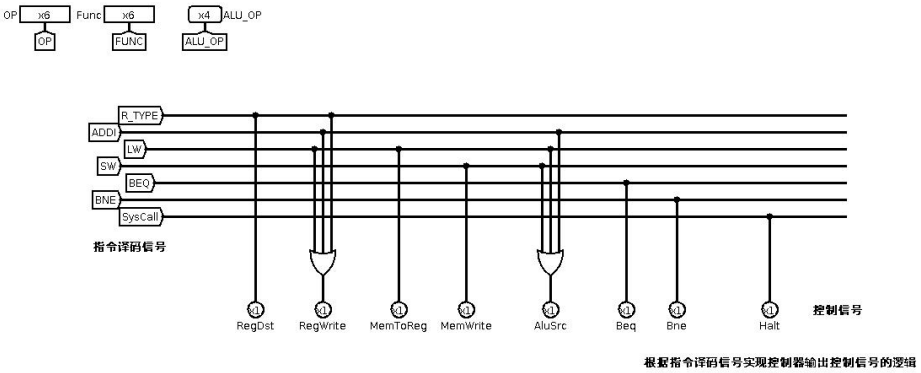


图 1.7 单周期硬布线控制器图

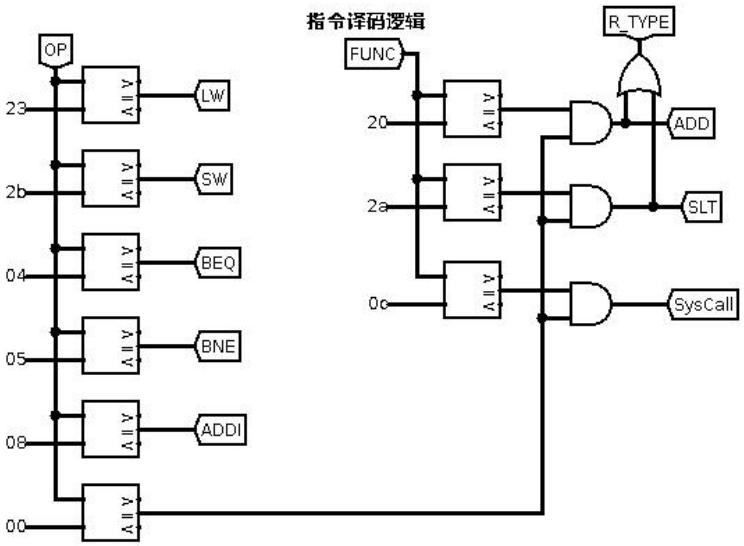


图 1.8 单周期硬布线指令译码逻辑图

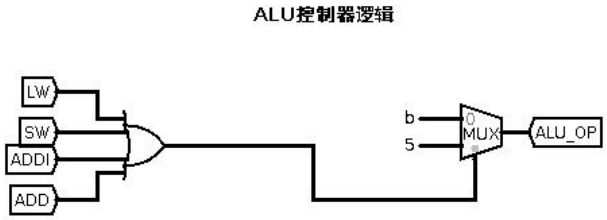


图 1.9 单周期硬布线 ALU 控制器图

3) 数据通路

核心单周期的 CPU 会在一个时钟周期内完成所有的工作，既从指令取出，到得到结果，全部在一个时钟之内完成，则无需处理指令间的关系。

整体数据通路图如 图 1.10 单周期数据通路图 所示

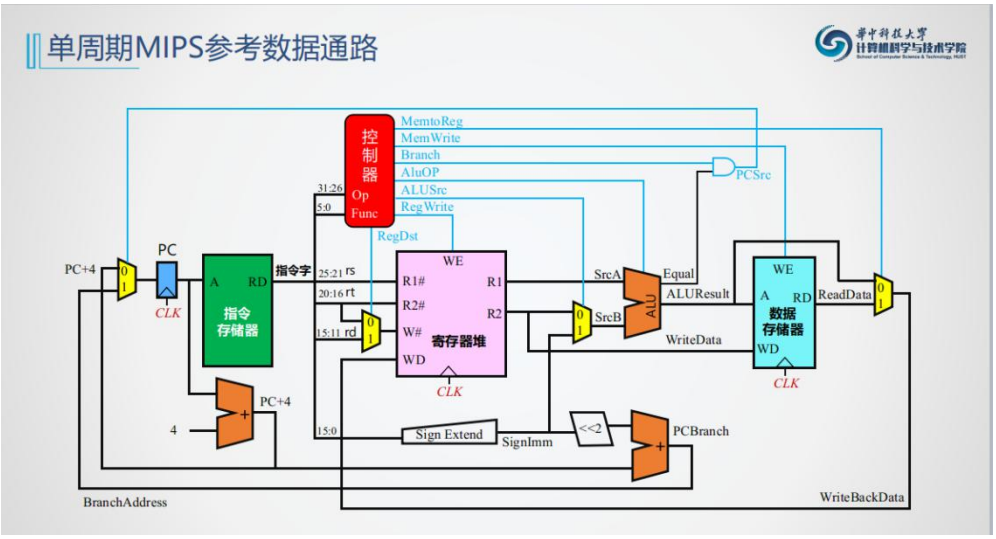


图 1.10 单周期数据通路图

1.2.2 多周期微程序 CPU

1) 总体架构

系统整体架构图如 图 1.11 多周期微程序 CPU 总体设计图 所示

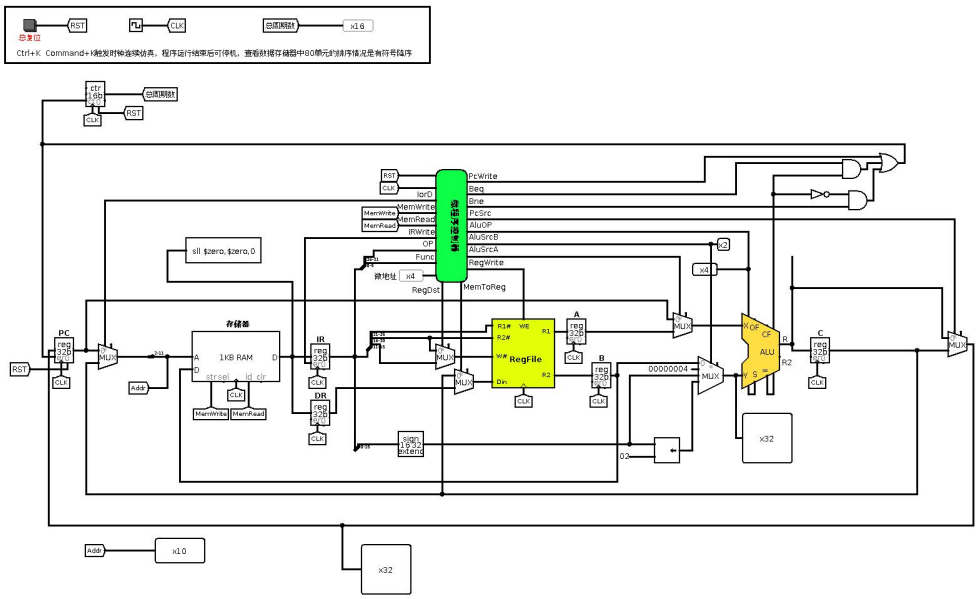


图 1.11 多周期微程序 CPU 总体设计图

需要实现:

R 型指令：ADD，SLT

I 型指令：ADDI，LW，SW，BEQ，BNE

程序中断指令：SYSCALL

译码及数据通路

多周期微程序 CPU 需要的的主要部件为 PC 寄存器、指令寄存器 IR、数据寄存器 DR、存储器、寄存器堆、用于存放寄存器文件的输出的寄存器和寄存器 B、ALU、用于存放 ALU 的输出的寄存器 C、控制器。

多周期微程序不再区分数据存储器和指令存储器，分阶段使用各功能部件。将一条指令分为多个周期执行，由于需要缓存数据，增加了 DR、IR 和部分寄存器来实现暂存数据和控制写入的效果。

所需要的组件如下：

- PC 寄存器、指令寄存器 IR、数据寄存器 DR、寄存器 A、B、C：logisim 寄存器；

- 存储器选择 RAM，寄存器堆：提供的寄存器封装；
- ALU：自己实现/CS3410 ALU；
- 微程序控制器：Logisim 器件构建；

2) 多周期微程序控制器

a) 整体封装

微程序控制器由指令寄存器 IR、程序计数器 PC、程序状态字寄存器 PSW、时序系统、控制存储器 CM、微指令寄存器以及微地址形成电路、微地址寄存器等部件组成。执行指令时，从控制存储器中找到相应的微程序段，逐次取出微指令，送入微指令寄存器，译码后产生所需微命令，控制各步操作完成。

如 图 1.12 多周期微程序控制器封装图 所示：

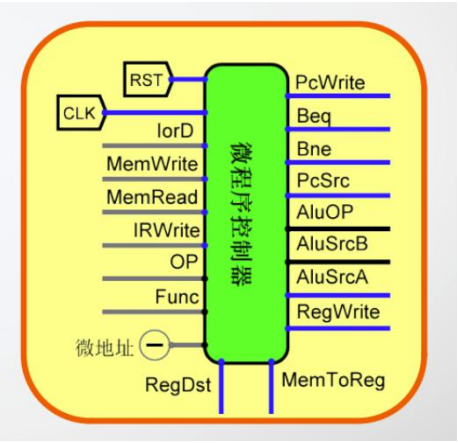


图 1.12 多周期微程序控制器封装图

与单周期相同，通过指令字 Op 和 Func 字段来对各个组件进行控制，控制选择一个通路来实现该指令的功能。

b) 控制器设计

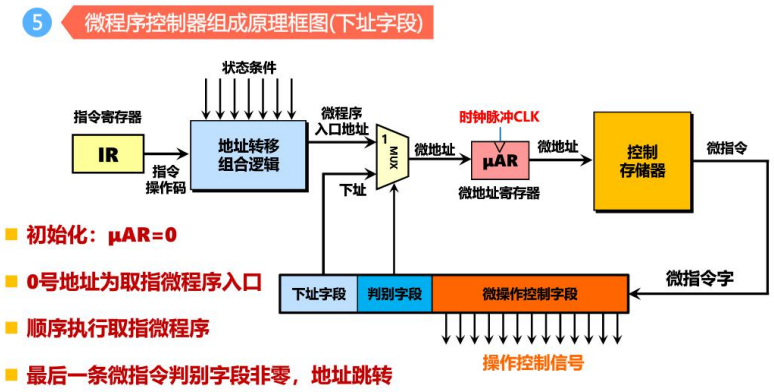


图 1.13 多周期微程序组成原理图

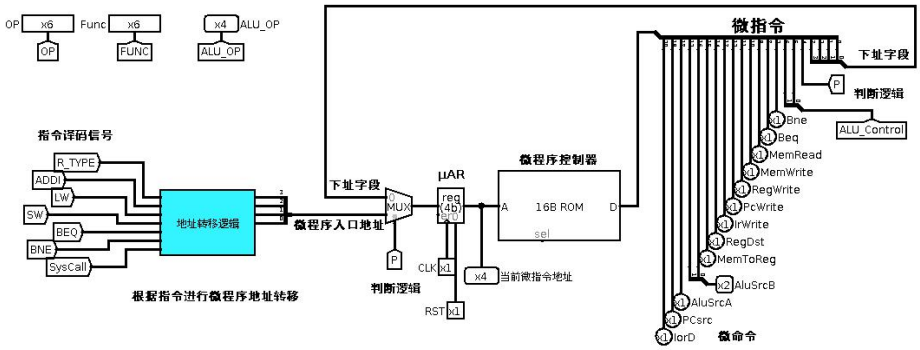


图 1.14 多周期微程序组成实现图

根据指令操作码生成入口地址，利用微程序的下址字段进行跳转，在最后一条微指令，进行下一个入口地址选通，完成一个周期内的相应功能。

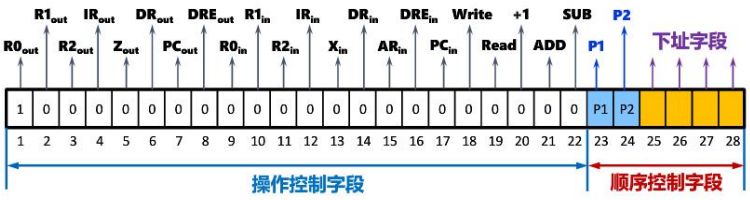


图 1.15 微指令格式图

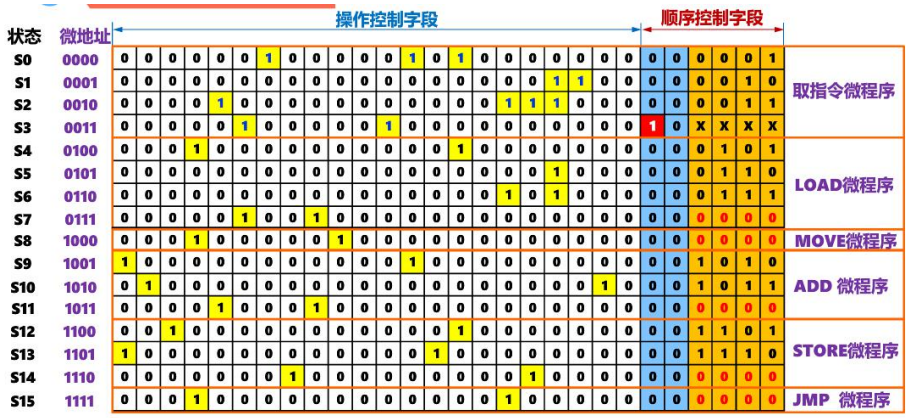


图 1.16 多周期微指令图

c) 地址转移逻辑设计

S3=ADDI + BEQ + BNE + SYSCALL

S2=R_Type + SW + SYSCALL

S1=R_Type + ADDI + LW + BNE

S0=R_Type + ADDI + SW + BEQ + SYSCALL

生成相应的电路

总体架构

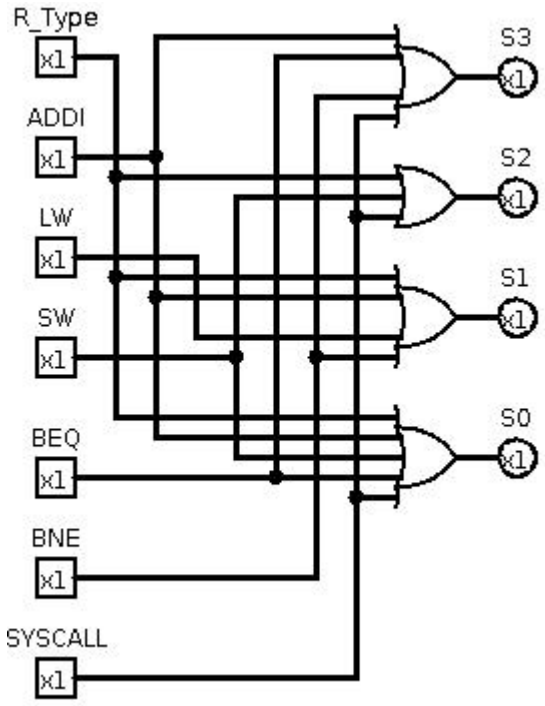


图 1.17 地址转移逻辑图

d) 指令译码逻辑和 ALU 控制器逻辑

表 1.6 多周期 ALU_OP 控制器逻辑表

Alu_Control	运算	ALU_OP
00	加法	5
01	减法	6
10	SLT 决定	
	0 加法	5
	1 三目运算	b

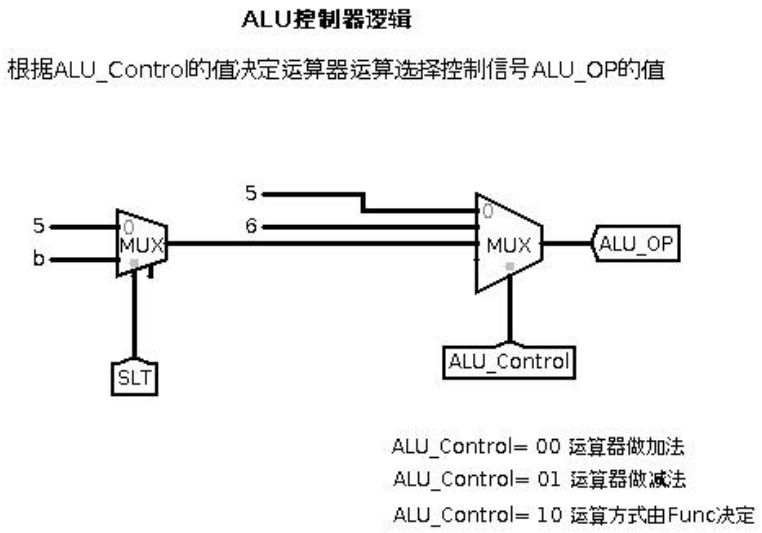


图 1.18 ALU 控制逻辑图

如 表 1.7 多周期控制器指令对应控制信号表 所示

表 1.7 多周期控制器指令对应控制信号表

	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 控制，00 加，01 减，10 由 Funct 定
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

3) 数据通路

分为取指，译码，执行，访存取数、结果写回。对于不同的指令类型，有不同的数据通路，因此需要根据不同的情形导通不同的数据通路。

多周期数据通路图如 图 1.19 多周期数据通路图 所示

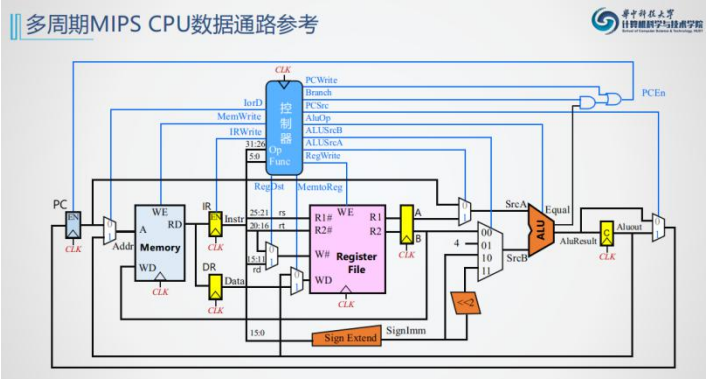


图 1.19 多周期数据通路图

a) 取指通路

取指令过程：

i. 取出 PC 值到 IR

(PC)->IR

此时需要：

IRWrite 进行 IR 写入

ii. 完成(PC)<-(PC)+4 计算

此时需要：

PCWrite 进行 PC 写入

ALUOP 进行运算器加法运算选择

ALUSrcB 进行运算器的运算数选择

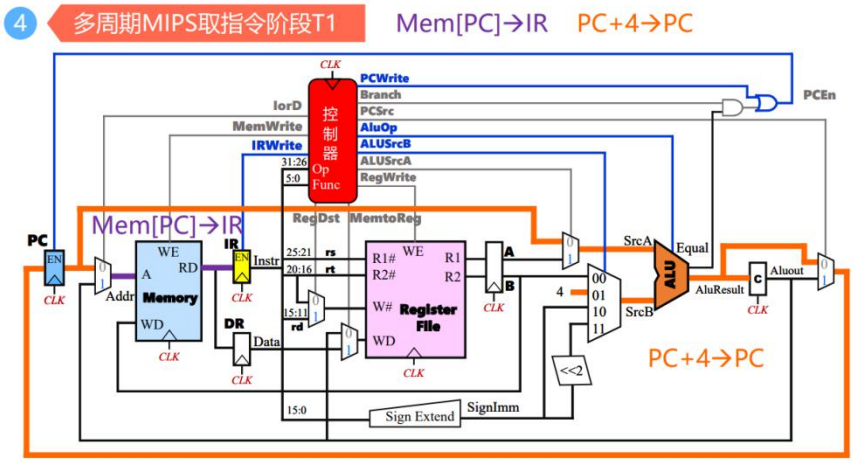


图 1.20 取指通路图

b) 译码通路

i. 将数据(运算数)送入 A,B 寄存器

ii. 进行跳转地址计算

PC+4+Imm16<<2->C

此时需要：

ALUOP 进行运算器加法运算选择

ALUSrcB 进行地址偏移计算选择

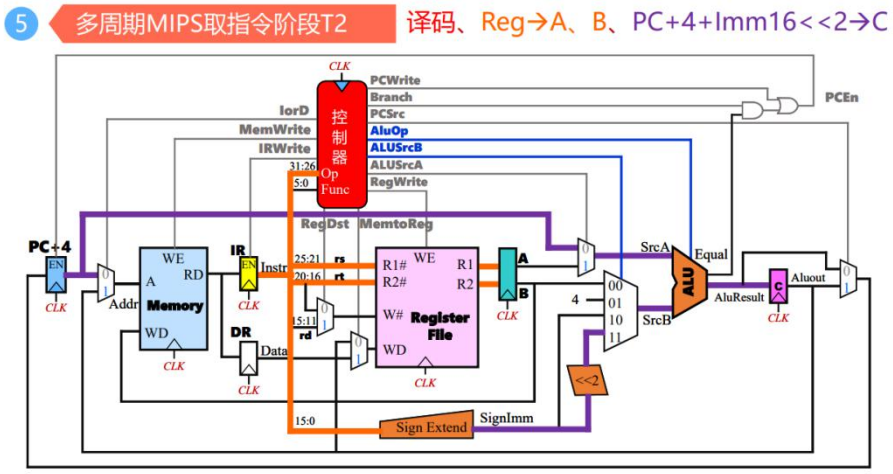


图 1.21 译码通路图

c) 执行通路

i. R 型指令执行通路

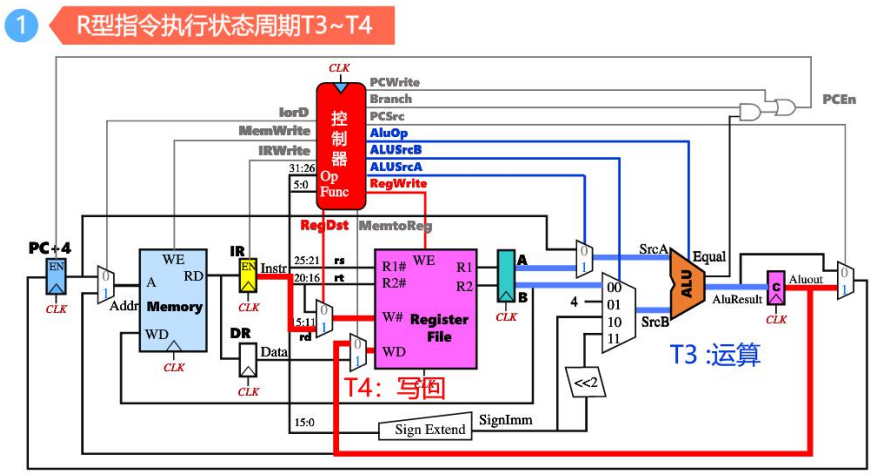


图 1.22 R 型指令执行通路图

① 进行指令运算

此时需要：

ALUOP 进行运算器加法运算选择

ALUSrcA 进行运算数 A 来源选择

ALUSrcB 进行运算数 B 来源选择

② 运算结果写回

将运算结果送到寄存器写地址

此时需要：

RegWrite 进行寄存器写使能

RegDst 寄存器写地址选择

ii. LW 指令执行通路

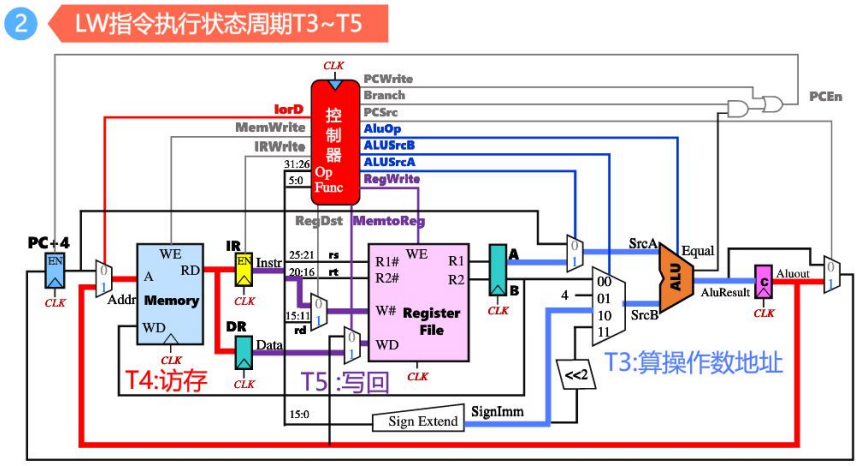


图 1.23 LW 指令执行通路图

① 操作数运算

此时需要：

ALUOP　　进行运算器加法运算选择

ALUSrcA　进行运算数 A 来源选择

ALUSrcB　进行运算数 B 来源选择

② 依据计算地址访存

此时需要：

LorD　　内存地址来源选择

③ 访存结果写回

将访存结果写入寄存器

此时需要：

RegWrite　进行寄存器写使能

MemToReg　写回数据来源选择

iii. Beq 指令执行通路

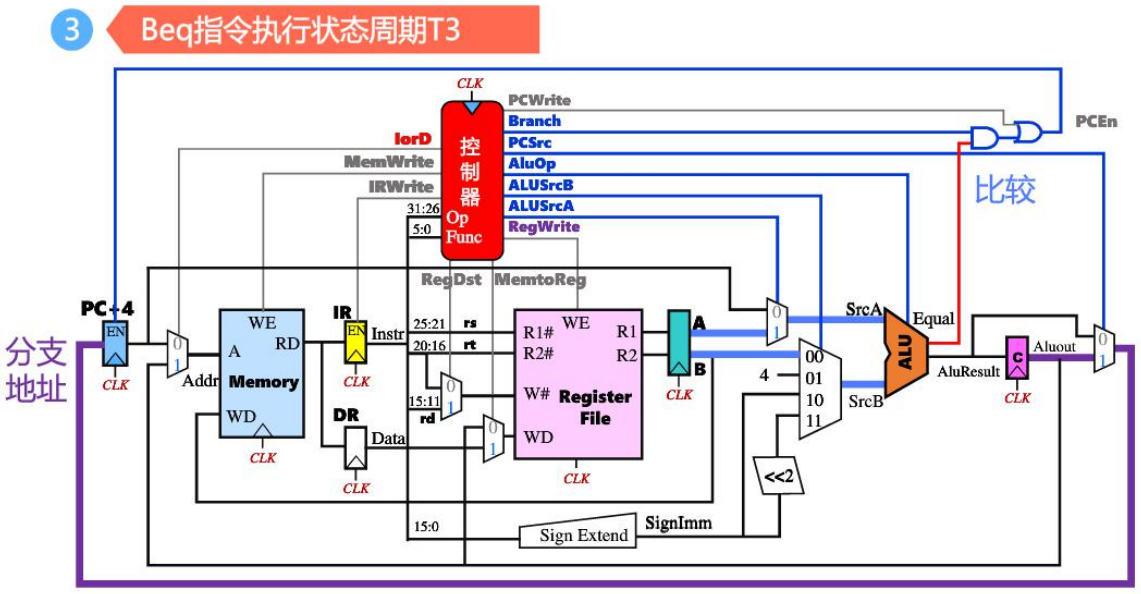


图 1.24 Beq 指令执行通路图

由于前面取值阶段已经完成了跳转地址的计算，此时只需要进行条件判断，如果满足，直接进行相应地址的跳转

① 运算结果(比较)

此时需要：

Branch　　进行分支的“使能”信号

ALUOP　　进行运算器加法运算选择

ALUSrcA　　进行运算数 A 来源选择

ALUSrcB　　进行运算数 B 来源选择

PCSrc　　PC 的数据来源选择，此时 PC 前向数据为跳转地址

4) 多周期状态机

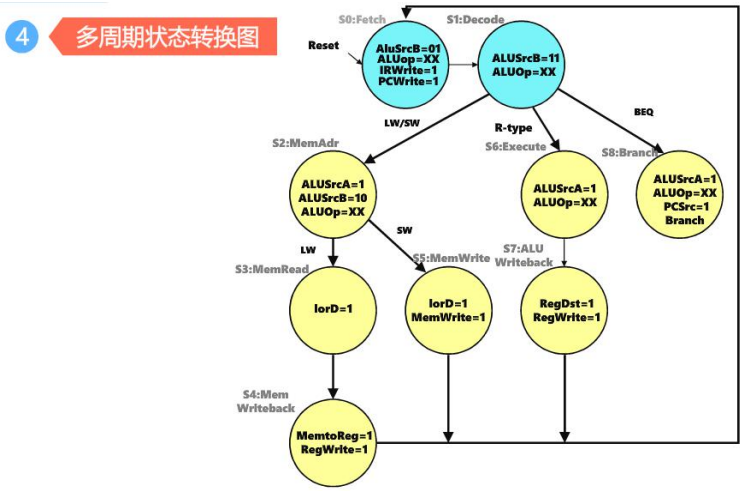


图 1.25 多周期状态转换图

在实现过程中，不进行状态的合并，方便对于不同指令进行不同通路的操作

1.2.3 多周期硬布线 CPU

1) 总体架构

系统整体架构图如 图 1.26 多周期硬布线 CPU 总体设计图 所示

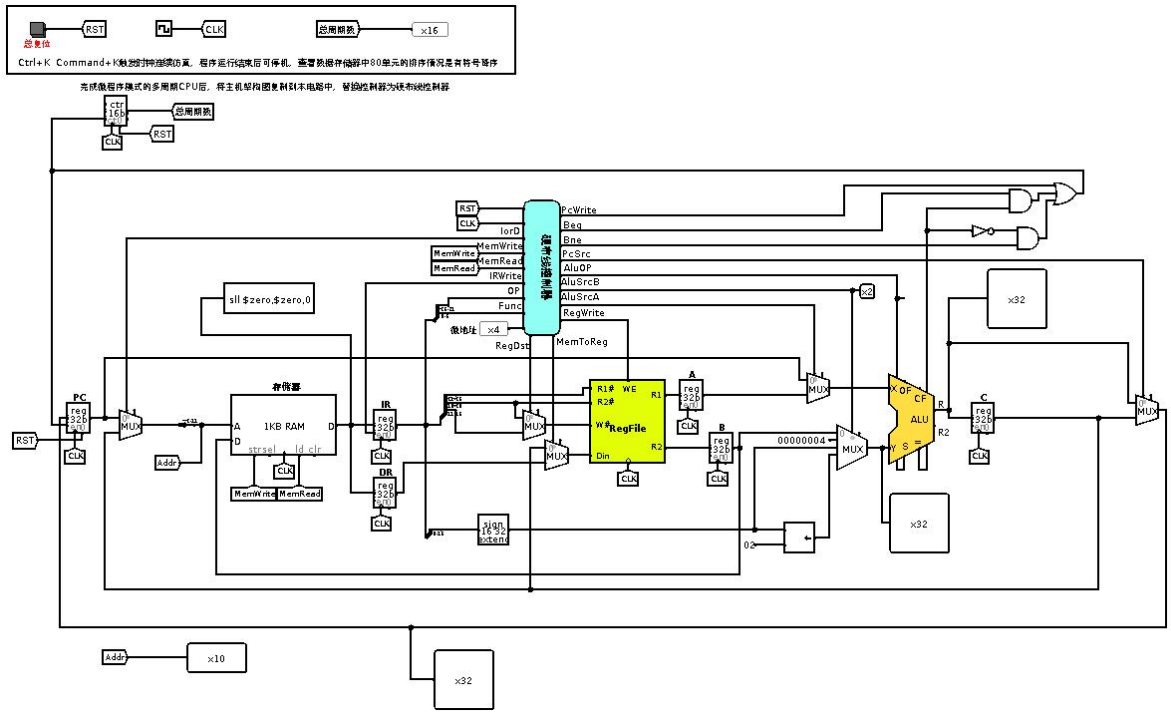


图 1.26 多周期硬布线 CPU 总体设计图

需要实现:

R 型指令：ADD，SLT

I 型指令：ADDI，LW，SW，BEQ，BNE

程序中断指令：SYSCALL

译码及数据通路

多周期硬布线 CPU 需要的的主要部件为 PC 寄存器、指令寄存器 IR、数据寄存器 DR、存储器、寄存器堆、用于存放寄存器文件的输出的寄存器和寄存器 B、ALU、用于存放 ALU 的输出的寄存器 C、控制器。

所需要的组件如下：

- PC 寄存器、指令寄存器 IR、数据寄存器 DR、寄存器 A、B、C：logisim 寄存器；
- 存储器选择 RAM，寄存器堆：提供的寄存器封装；
- ALU：自己实现/CS3410 ALU；

- 硬布线控制器：Logisim 器件构建；

2) 多周期硬布线控制器

a) 整体封装

如 图 1.27 多周期硬布线控制器封装图 所示：

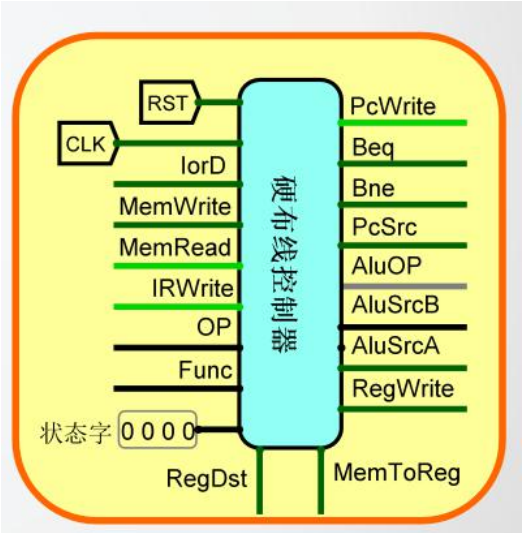


图 1.27 多周期硬布线控制器封装图

与单周期相同，通过指令字 Op 和 Func 字段来对各个组件进行控制，控制选择一个通路来实现该指令的功能。

b) 硬布线控制器设计

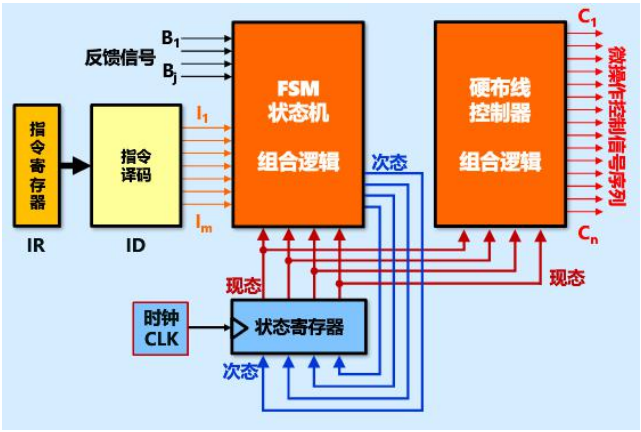


图 1.28 多周期硬布线控制器组成原理图

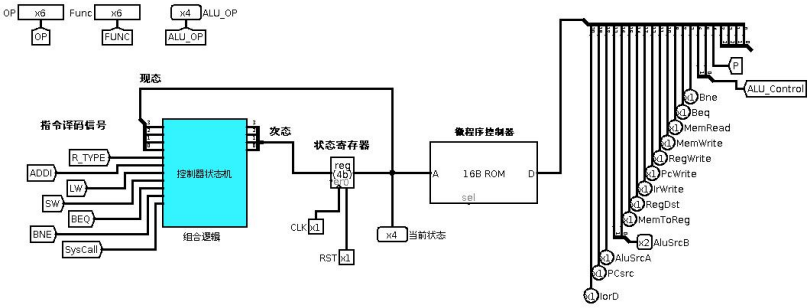


图 1.29 多周期硬布线控制器实现图

根据指令进行译码，然后根据当前状态生成控制信号。

c) 状态转移逻辑

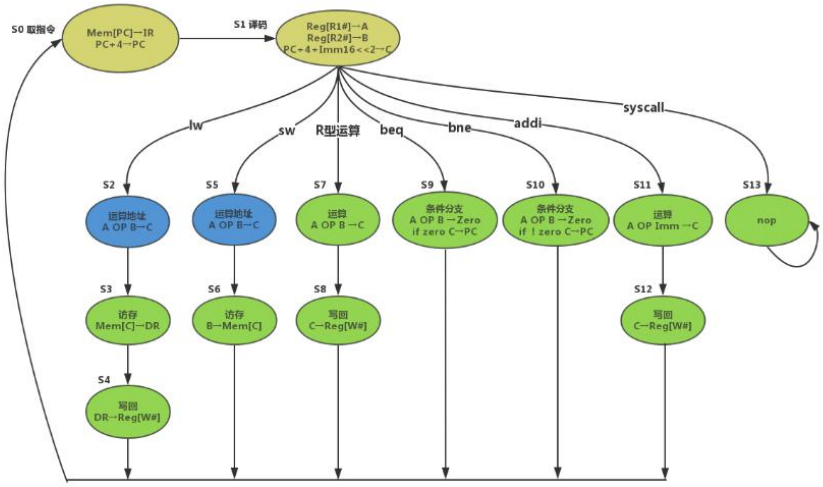


图 1.30 多周期硬布线控制器状态转移图

下列为多周期状态转移的表达式

S3 = ~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;

S2 = ~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;

S1 = ~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;

S0 = ~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;

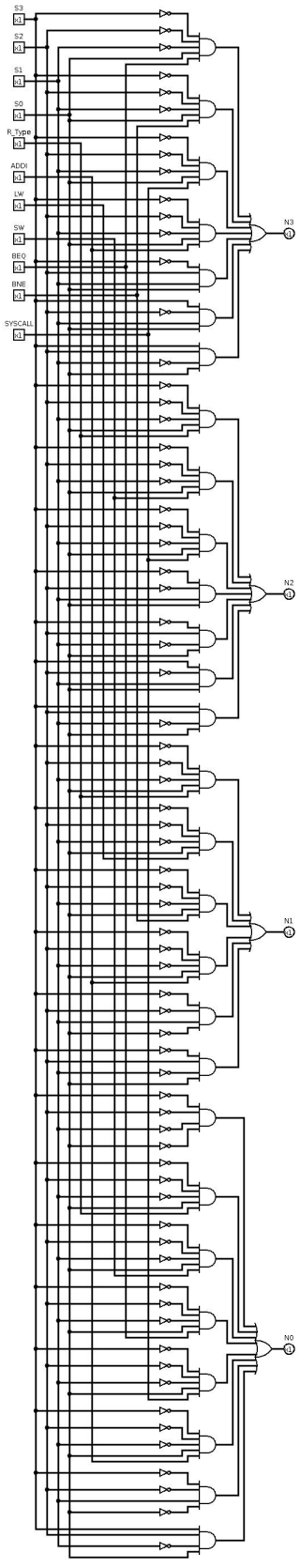


图 1.31 多周期硬布线控制器状态转移电路组合逻辑图

1.3 实验步骤

仔细阅读实验指导和任务书，明确实验任务，制定试验设计并按照设计进行连线，边连线边进行基本的测试。后续的实验可以类比上面实验的连线适当减轻工作量。

1) 单周期硬布线 CPU
   a) 设计步骤
      i. 单周期数据通路
      ii. 微程序控制器
         ① 设计时序产生器： 所有指令固定机器周期数，节拍数 ；
         ② 列出所有机器指令的指令周期流程图，明确每个节拍的控制信号；
         ③ 找出产生同一微操作控制信号的条件；
         ④ 写出各微操作控制信号的布尔表达式；
         ⑤ 化简各表达式；
         ⑥ 利用组合逻辑电路实现(logisim 生成)。
   b) 测试步骤
      i. 将冒泡程序的 MIPS 指令加载到指令存储器中。
      ii. 运行电路。
      iii. 待指令停止运行，
         ① 观测结果，是否有序
         ② 观测使用了多少个指令周期
2) 多周期微程序 CPU
   a) 设计步骤
      i. 多周期数据通路
      ii. 微程序控制器
         ① 根据状态转换图设计微程序
         ② 按微程序入口地址设计地址转移逻辑
         ③ 构造微程序控制器

b) 测试步骤

i. 将冒泡程序的 MIPS 指令加载到指令存储器中。

ii. 运行电路。

iii. 待指令停止运行，

① 观测结果，是否有序

② 观测使用了多少个指令周期

3) 多周期硬布线 CPU

a) 设计步骤

i. 多周期数据通路

ii. 硬布线控制器

① 设计时序产生器： 所有指令固定机器周期数，节拍数 ；

② 列出所有机器指令的指令周期流程图，明确每个节拍的控制信号；

③ 找出产生同一微操作控制信号的条件；

④ 写出各微操作控制信号的布尔表达式；

⑤ 化简各表达式；

⑥ 利用组合逻辑电路实现。

b) 测试步骤

i. 将冒泡程序的 MIPS 指令加载到指令存储器中。

ii. 运行电路。

iii. 待指令停止运行，

① 观测结果，是否有序

② 观测使用了多少个指令周期

1.4 故障与调试

因为报告和实验并未同步进行，此处尽可能对于当时遇到的错误进行复现。

1.4.1 ALU 选择问题

故障现象：CPU 周期较慢

原因分析：使用自己实现的 ALU，而性能方面存在问题，具有较高的门级时延，而 CS3410 ALU 使用 logisim 内部自带的运算器，能够较快的完成。

解决方案：使用 CS3410 ALU

1.4.2 布线误触

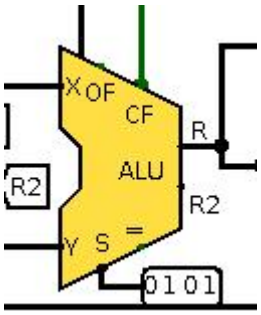


图 1.32 单周期布线误触图

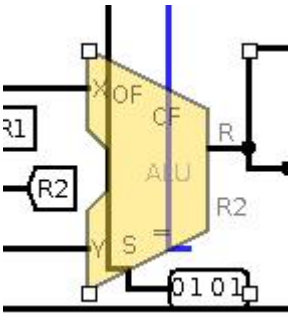


图 1.33 单周期布线透视图

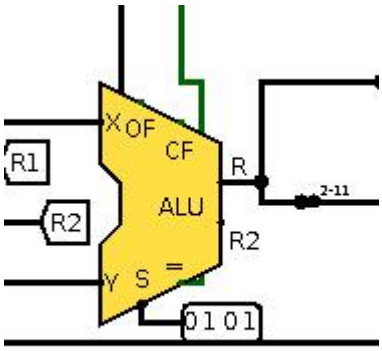


图 1.34 单周期布线修正图

故障现象：某一个过程不进行跳转，最终不停机

原因分析：布线误触

解决方案：更改布线

调试过程：在调试过程中，采用反汇编器一步步单步调试，发现某一步不进行跳转，从而对于某一步不进行跳转，于是观察线值，从而发现误触现象，此处提醒我们对于隐藏布线注意，对于玄学 bug 单步调试是好的解决方法。

1.4.3 中途红线

故障现象：程序运行过程中出现红线，但最终结果正确

原因分析：logisim 性能问题，也可能为实验过程中，未 ctrl+r 重置电路，造成某些部分的状态不对导致冲突

解决方案：重置程序

1.5 测试与分析

1.5.1 单周期硬布线 CPU 测试分析

```
v2.0 raw
2010ffff 20110000 ae300200 22100001 22310004 ae300200 22100001 22310004
ae300200 22100001 22310004 ae300200 22100001 22310004 ae300200 22100001
22310004 ae300200 22100001 22310004 ae300200 22100001 22310004 ae300200
8020 2011001c 8e130200 8e340200 274402a 11000002 ae330200 ae140200
2231fffc 1611fff8 22100004 2011001c 1611fff5 2002000a c
```

图 1.35 排序测试程序图

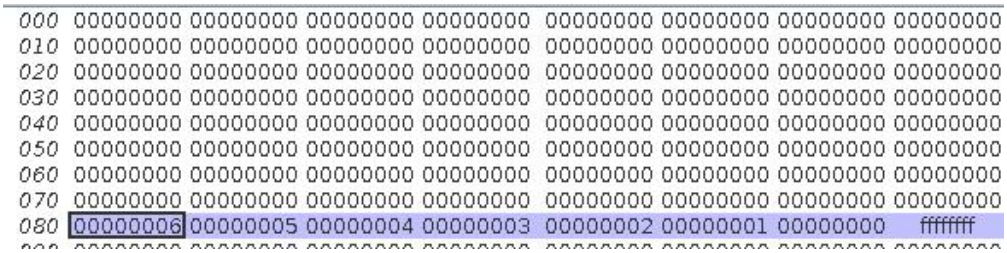


图 1.36 单周期硬布线 CPU 排序测试结果图

1.5.2 多周期微程序 CPU 测试分析

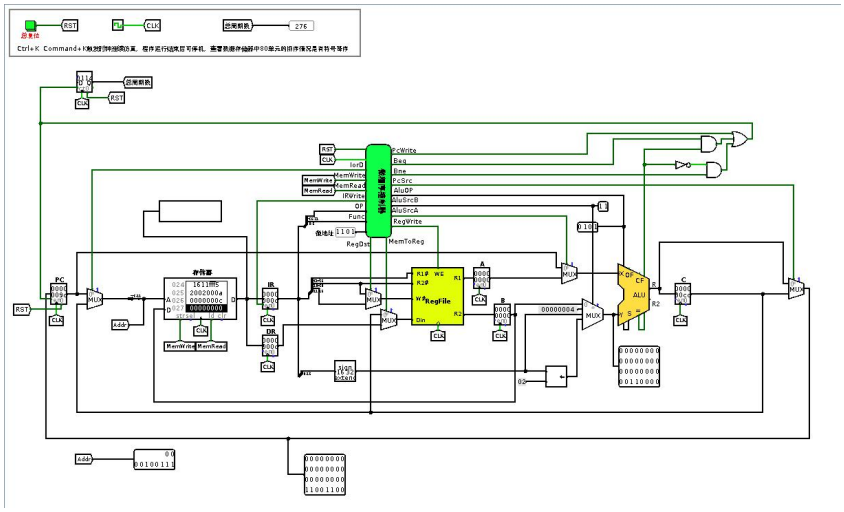


图 1.37 多周期微程序 CPU 排序测试结果图

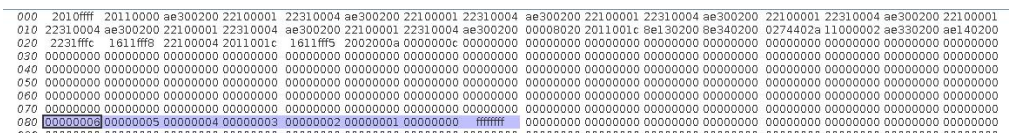


图 1.38 多周期微程序 CPU 测试周期结果图

1.5.3 多周期硬布线 CPU 测试分析

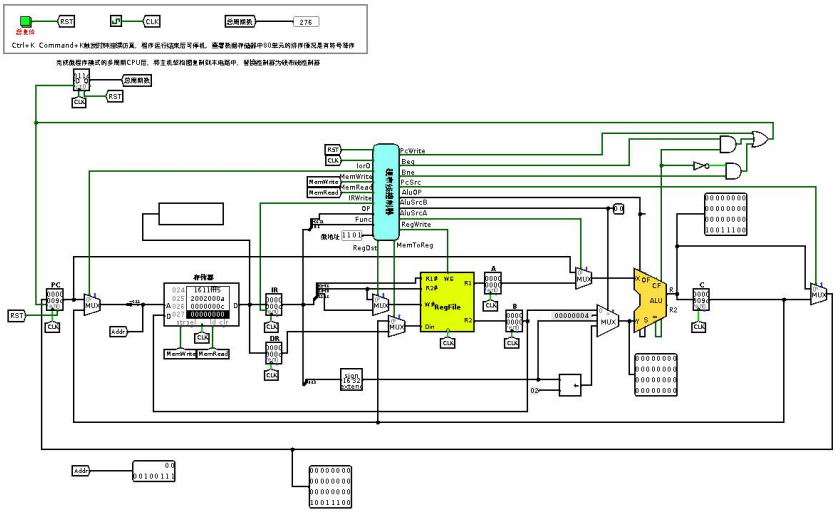
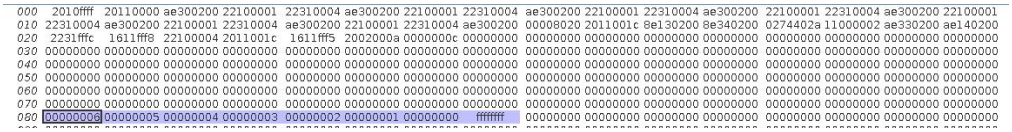


图 1.39 多周期硬布线 CPU 排序测试结果图



000	2010ffff	20110000	ae300200	22100001	22310004	ae300200	22100001	22310004	ae300200	22100001	22310004	ae300200	22100001	22310004	ae300200	22100001
010	22310004	ae300200	22100001	22310004	ae300200	22100001	22310004	ae300200	00008020	2011001c	8e130200	8e340200	0274402a	11000002	ae330200	ae140200
020	2231fffc	1611fff8	22100004	2011001c	1611fff5	2002000a	0000000c	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
030	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
040	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
050	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
060	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
070	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
080	00000006	00000005	00000004	00000003	00000002	00000001	00000000	ffffffff	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000

图 1.40 多周期硬布线 CPU 测试周期结果图

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） 学习补充 MIPS 汇编相关知识

2） 进行实验方案设计

3） 完成数据通路的连接

4） 完成控制器状态机，译码等的编码设计

5） 进行实验整体测试

2.2 实验心得

1） 实验过程中，复习了相关的知识之后进行学习会比较顺利，同时可以加深对于实验的理解，参阅了一些教材如《CPU 自制入门》

2） 整个实验过程数字电路较为紧密,学习了 logisim 的电路存储格式，为标签试样，描述当前标签的名称和起始点，此处对于程序的理解加深，并可以采用脚本语言进行大规模脚本复制，避免程序中复制的红线。

3） Logisim 对于自带运算器的实现直接采用 java 的运算而不是采用器件模拟，故自己实现的运算器和自带的运算器有一定的性能差异

4） 实验调试过程中，可以采用单步并输出线值，logisim 对于原件采用统一的门级时延，此时可以观察到在 cpu 单周期实验时，时钟周期的长短是不同的，原因是不同的指令有不同长度的数据通路，和不同的时延长短。

5） 整个实验对于组成原理的学习帮助较大，对于并未要求的实验也进行了一定的尝试，因为这样可以更好的理解整个过程，老师考虑到同学们的工作量，限定了实验的量和难度，可以考虑开放更多的选做实验，比如将 Cache 嵌入内存模组，在程序运行过程中观察 cache 实际访问情况，考虑书写 MIPS 汇编利用汇编器生成最终的机械码，能够更好的看到一步步过程，

同时加强实验的成就感。

6） 最后感谢组长原理课程组的支持和帮助以及在实验过程中参阅资料的提供者和提供帮助的老师同学。

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.

[6] 水头一寿(日)，米泽辽(日)，藤田裕士(日). CPU 自制入门.北京：人民邮电出版社，2014 年.

[7] 左冬红.计算机组成原理与接口技术：基于 MIPS 架构.北京：清华大学出版社，2014 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： 

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：_____ 2019-01-06

·指导教师评定意见·