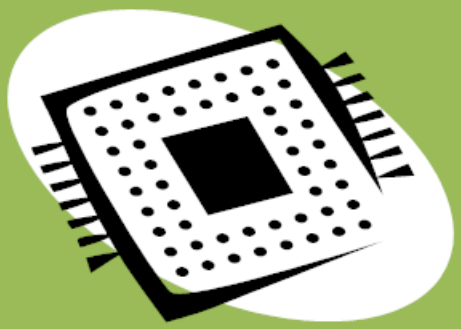
華中科技大學

2022

计算机组成原理 ·实验报告·

专　　业：	计算机科学与技术
班　　级：	CS2001
学　　号：	U202011641
姓　　名：	刘景宇
电　　话：	15671569229
邮　　件：	2537738252@qq.com
完成日期：	2022-06-30



计算机科学与技术学院

1 CPU 设计实验

1.1 设计要求

利用 logisim 平台中已有的组件构建一个支持中断的单总线 MIPS 现代时序 CPU，总线位宽为 32 位，控制器采用微程序、硬布线两种方式实现，支持 LW、SW、BEQ、SLT、ADDI 五条基础 MIPS 指令以及 ERET 中断返回指令，支持中断识别，能够根据中断的类型跳转到不同的中断处理程序的入口地址，并能够在设计的单总线 CPU 上利用以上指令运行简单的排序程序 sort-5-int.hex。

支持中断的单总线 MIPS CPU 计算机框图如下图所示，计算机框图展示了 CPU 的实现逻辑，本次实验中涉及到 28 个控制信号，控制信号及其功能描述如表 1-1 所示，本次实验中需要使用 5 条基础 MIPS 指令，指令及其 RTL 功能说明如表 1-2 所示。

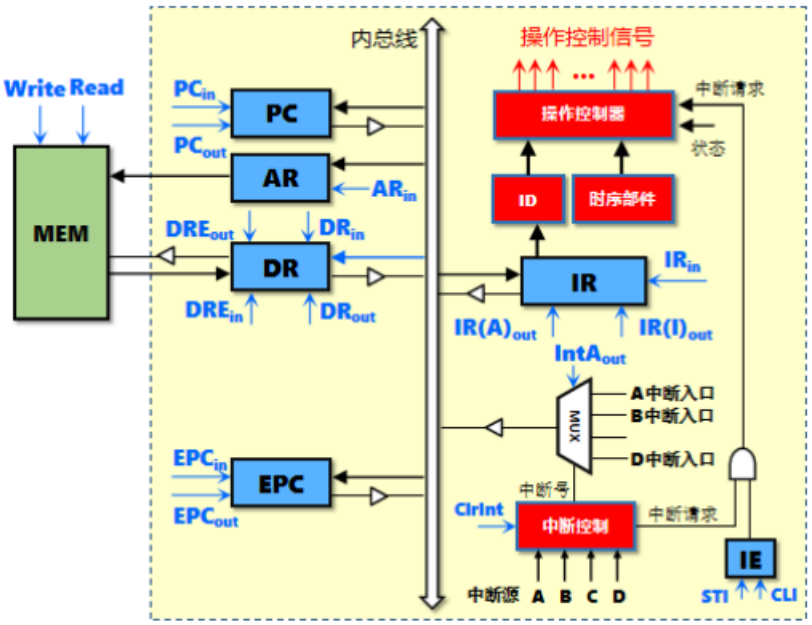


图 1-1 支持中断的单总线 MIPS CPU 结构的计算机框图

表 1-1 控制信号及其功能

控制信号	功能描述
PC_{in}	控制 PC 接收来自内总线的数据，需配合时钟控制
PC_{out}	控制 PC 向内总线输出数据
AR_{in}	控制 AR 接收来自内总线的数据，需配合时钟控制
DR_{in}	控制 DR 接收来自内总线的数据，需配合时钟控制
DR_{out}	控制 DR 向内总线输出数据
DRE_{in}	控制 DR 接收从主存读出的数据，需配合时钟控制
DRE_{out}	控制 DR 向主存输出数据，以便最后将该数据写入主存
X_{out}	控制暂存寄存器 X 接收来自内总线的数据，需配合时钟控制
+4	将 ALU A 端口的数据加 4 输出
ADD	控制 ALU 执行加法，实现 A 端口和 B 端口的两数相加
SUB	控制 ALU 执行减法运算
PSW_{in}	控制状态寄存器 PSW 接收 ALU 的运算状态，需配合时钟控制
Z_{out}	控制暂存寄存器 Z 向内总线输出数据
IR_{in}	控制 IR 接收来自内总线的指令，需配合时钟控制
$IR(A)_{out}$	控制 IR 中的分支目标地址输出到内总线
$IR(I)_{out}$	控制 IR 中的立即数输出到内部总线
Write	存储器写命令，需配合时钟控制
Read	存储器读命令
R_{in}	控制寄存器堆接收来自内总线的数据，写入 W#端口对应的寄存器中
R_{out}	控制寄存器堆输出指定编号 R#寄存器的数据
Rs/Rt	控制多路选择器选择送入 R#的寄存器编号
RegDst	控制多路选择器选择送入 W#的寄存器编号
EPC_{in}	控制 EPC 接收来自内总线的数据，需配合时钟控制
EPC_{out}	控制 EPC 向内总线输出数据
STI	开中断，将中断使能寄存器置 1，操作控制器可以接收中断请求
CLI	关中断，将中断使能寄存器置 0，操作控制器将接收不到任何中断请求
$IntA_{out}$	将中断服务程序地址输出到内总线
ClrInt	清除当前正在响应的中断请求

表 1-2 MIPS32 指令

指令	汇编代码	RTL 功能说明
lw	lw rt,imm(rs)	R[rt] ← M[R[rs] + SignExt(imm)]
sw	sw rt,imm(rs)	M[R[rs] + SignExt(imm)] ← R[rt]
beq	beq rs,rt,imm	if(R[rs]==R[rt]) PC ← PC + 4 + SignExt(imm)<<2
addi	addi rt,rs,imm	R[rd] ← R[rs] + R[rt]
slt	slt rd,rs,rt	R[rd]←(R[rs] < R[rt])?1:0

1.2 方案设计

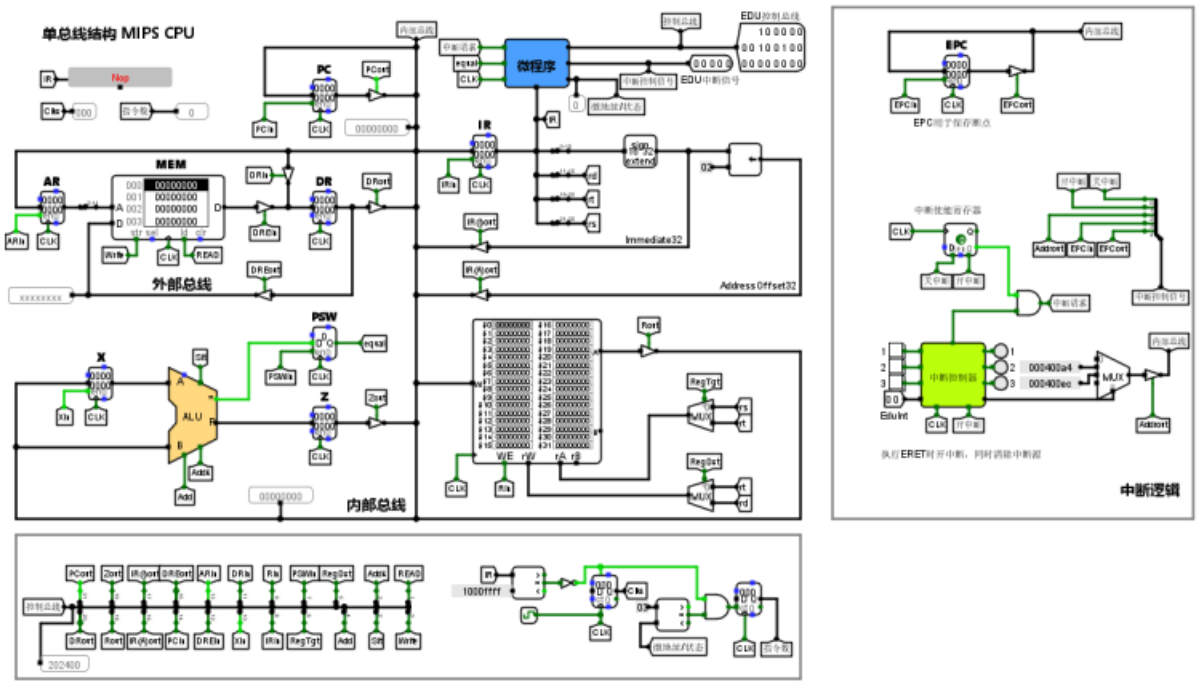


图 2-1 单总线 MIPS 总体图

1.2.1 MIPS 指令译码器

CPU 运行时，为了能识别 MIPS 指令，需要设计 MIPS 指令译码器。MIPS 指令分为三种类型 R、I、J 型指令，其中 R 型指令 OP 均为零，功能由 FUNCT 确定，I 和 J 型指令由 OP 确定。通过查阅 MIPS 指令手册，将 ADDI、LW、SW、BEQ、SLT 五条指令的机器码确定，通过比较 IR 和以上五条指令，相等时输出端口输出 1，否则为 0。当出现非以上五条指令的情况，将 OtherInstr 设置为 1，表示当前 IR 中存储的不是以

上五条指令，实现时可以采用组合逻辑电路。

1.2.2 支持中断的微程序入口地址查找逻辑

在设计微程序实现的 CPU 时，需要构建微指令的查找表，在控制存储器中存储控制信号。为了实现微程序入口查找，需要确定每条指令在控制存储器中的微地址。微地址需要设计，每个微指令对应一个状态，状态转换图如图 2-2 所示，根据状态转换图，可以确定各个指令的入口地址。当指令确定时，对应的入口地址也确定，即当输入指令信号时，能够输出该指令的入口微地址，对应的微程序入口地址表如表 2-1 所示，可以采用组合逻辑电路实现。

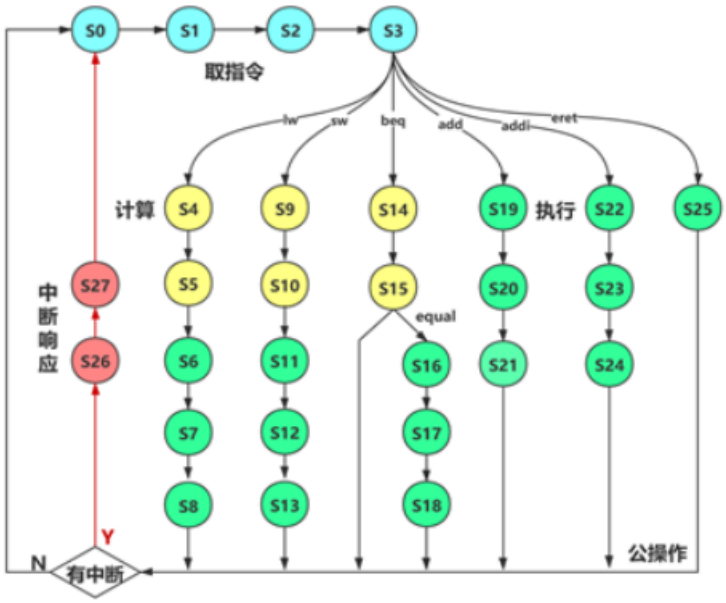


图 2-2 状态转换图

1.2.3 支持中断的微程序条件判别测试逻辑

BEQ 指令的功能是当两个寄存器内容相等的情况下实现跳转，不等的时候继续执行下一条指令。寄存器内容相等时，比不相等要多执行计算新的 PC 的并更改 PC 的过程，如图 2-1 状态转换图所示。

为了能够实现分支的过程，增加了条件判别逻辑。对于每条微指令设置 P0，P1，P2 判断字段，对于不同的情况，要取不同的微地址继续执行。当 P1=1 且 equal=1 时，微地址设置为 beq 的分支，当 P2=1，IntR=1 时，当前微地址设置为中断处理程序入口地址，当 P0=1 时，要根据当前指令设置微地址的值，其余情况通过下址字段法或者

计数器法确定微地址的值。不同的 PC 取值方式对应一个编号，判别字段、信号与 PC 取值方式对应关系如表 2-2 所示。注意 P1=1 且 equal=1 的优先级高于 P2=1 且 intR=1，因为当 P1=1 且 equal=1 时 BEQ 指令还未执行完，不能进入中断响应。

表 2-1 微程序入口地址表

机器指令译码信号						微程序入口地址					
LW	SW	BEQ	SLT	ADDI	ERET	入口地址10进制	S4	S3	S2	S1	S0
1						4	0	0	1	0	0
	1					9	0	1	0	0	1
		1				14	0	1	1	1	0
			1			19	1	0	0	1	1
				1		22	1	0	1	1	0
					1	25	1	1	0	0	1

表 2-2 条件判别逻辑表

P0	P1	P2	equal	IntR	S2	S1	S0
0	0	0	0	0	0	0	0
1					0	0	1
0	1		1		0	1	0
0	0	1	1	1	0	1	1
0		1	0	1	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0		1	0	0
0		1	0	0	1	0	0

1.2.4 支持中断的微程序控制器设计

微程序控制器中的重要部件是控制存储器，即微指令的存储器，存储着各个状态状态即微指令对应的微命令。由于是单总线 CPU，所以要在设计时，注意避免总线冲突。控制信号每个微指令对应的微命令都是固定的，微指令和控制信号的对应关系如表 2-3 所示。将微命令信号按照微地址的顺序存储到控制存储器 ROM，就完成了控制存储器的构造。

将以上完成的指令译码器、微程序入口查找逻辑电路、条件判别测试逻辑电路、存储控制器连接，完成微程序控制的主要数据通路，能够根据判别测试逻辑的输出选择 PC 取值方式，实现顺序取址、中断响应、beq 分支等的选择，输入指令字通过指令译码器产生响应的信号，结合所有的信号输入，能够在控制存储器 ROM 中找到对应的微命令对应的控制信号并输出，就构成了支持中断的微程序控制器。

1.2.5 支持中断的微程序单总线 CPU 设计

为了能够实现中断，需要在单总线数据通路中增加与中断相关的硬件模块，包括异常程序地址计数器 EPC、中断使能寄存器 IE、中断控制模块等，同时在添加模块的过程中注意避免总线冲突，连接好的单总线 CPU 要能够正确运行 sort5-int.hex。

异常程序地址计数器 EPC 用于保存断点，即保存中断相应前 PC 中的内容。在设计微指令时添加了 EPCin、EPCout 微命令，通过这两个微命令控制 EPC 与总线的数据交互，利用三态门控制输出避免数据总线冲突。中断使能寄存器 IE 用于存储中断使能信号，确定当前能否被产生的中断打断，当处于信号开中断，并且中断控制器检测识别到中断时，产生中断信号。

中断控制器产生中断类型的编号，不同类型的中断对应不同的中断处理程序入口地址。中断控制器在完成中断识别后，通过 Addrout 信号控制将中断程序入口地址输出到总线将程序入口地址送入到 PC 中。当产生中断时，就能够根据中断类型设置 PC 的值，从而实现跳转到中断处理程序执行。

表 2-3 微指令-控制信号对应表

微指令功能		PCout	DRout	Zout	Rout	IROout	IRAout	DRout	PCin	ARin	DRin	DRin	Xin	Rin	IRin	PSWin	Rs/Rt	Imdout	Add	Add4	Slt	READ	WRITE	EPCout	EPCin	Addrout	STI	CLI	P1	P2	P3
取指令	0	1								1			1																		
取指令	1																			1											
取指令	2			1					1		1											1									
取指令	3		1												1														1		
lw	4				1								1																		
lw	5					1													1												
lw	6			1						1																					
lw	7										1											1									
lw	8		1											1																	1
sw	9				1								1																		
sw	10					1													1												
sw	11			1						1																					
sw	12				1							1						1													
sw	13							1															1								1
beq	14				1								1																		
beq	15				1											1	1													1	1
beq	16	1											1																		
beq	17						1												1												
beq	18			1					1																						1
slt	19				1								1																		
slt	20				1												1				1										
slt	21			1										1				1													1
addi	22				1								1																		
addi	23					1													1												
addi	24			1										1																	1
中断	25								1															1			1				1
中断	26	1																							1			1			
中断	27								1																	1					1

1.2.6 支持中断的现代时序硬布线控制器状态机设计

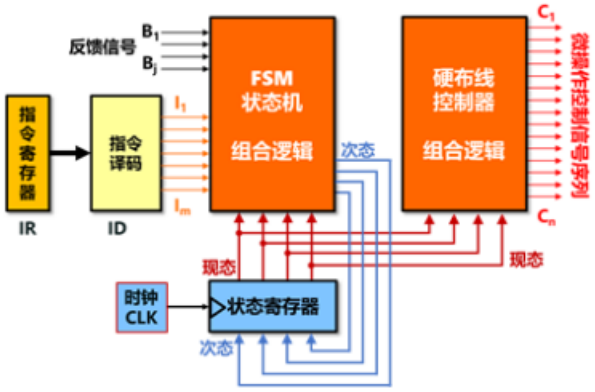


图 2-3 硬布线控制示意图

硬布线控制器不同于微程序控制器，微程序控制器将控制信号存储下来，而硬布线控制器根据指令以及反馈信号等确定控制信号。传统三级时序利用时序发生器产生不同的时钟节拍和当前的指令及反馈信号生成控制信号，而现代时序只根据当前的状态确定控制信号，现代时序对应的状态更多、更复杂，但是效率更高。现代时序硬布线控制示意如图 2-3 所示。

现代时序硬布线控制器状态机输入指令译码信号、CPU 中的反馈信号以及当前状态，生成次态，跟时钟无直接关系，是组合逻辑，所以采用组合逻辑电路。本实验中采用了五条机器指令以及中断响应指令，共有 27 个状态，取值结束后，根据指令译码确定执行指令的入口地址，部分状态转换如表 2-4 所示，在每条指令执行完成后，都要判断当前是否有中断信号，确定跳转到取指入口地址还是中断响应。

表 2-4 部分状态转换表

当前状态(现态)						输入信号								下一状态（次态）					
S4	S3	S2	S1	S0	现态10进制	LW	SW	BEQ	SLT	ADDI	ERET	IR	EQUAL	次态10进制	N4	N3	N2	N1	N0
0	1	1	0	1	13							0		0	0	0	0	0	0
0	1	1	0	1	13							1		26	1	1	0	1	0
0	1	1	1	0	14									15	0	1	1	1	1
0	1	1	1	1	15							0	0	0	0	0	0	0	0
0	1	1	1	1	15								1	16	1	0	0	0	0
0	1	1	1	1	15							1	0	26	1	1	0	1	0
1	0	0	0	0	16									17	1	0	0	0	1
1	0	0	0	1	17									18	1	0	0	1	0
1	0	0	1	0	18							0		0	0	0	0	0	0
1	0	0	1	0	18							1		26	1	1	0	1	0
1	0	0	1	1	19									20	1	0	1	0	0
1	0	1	0	0	20									21	1	0	1	0	1

1.2.7 支持中断的现代时序硬布线控制器设计

现代时序硬布线控制器只根据当前所处的状态确定控制信号序列，结合上面的示意图，将指令译码器、硬布线控制器状态机、状态寄存器、时钟 CLK 进行合理地连接，次态由现态以及部分指令信号有关，为了简化硬布线设计，利用控制存储器替代由状态生成控制信号的组合逻辑，连接以上部件即可以实现硬布线控制器。

1.3 实验步骤

（1） MIPS 指令译码器设计

R 型指令 OP 均为零，功能由 FUNCT 确定，I 和 J 型指令由 OP 确定。通过查阅 MIPS 指令手册，将 ADDI、LW、SW、BEQ、SLT 五条指令的机器码确定，并利用分线器取出 IR 中的 OP、FUNCT，R 型指令比较 OP 和 FUNCT 字段，I 型和 J 型指令比较 OP，相等时对应输出端口输出 1，否则为 0。当出现非以上五条指令的情况，将 OtherInstr 设置为 1，表示当前 IR 中存储的不是以上五条指令，可以通过异或实现，指令译码器的电路如图 3-1 所示。

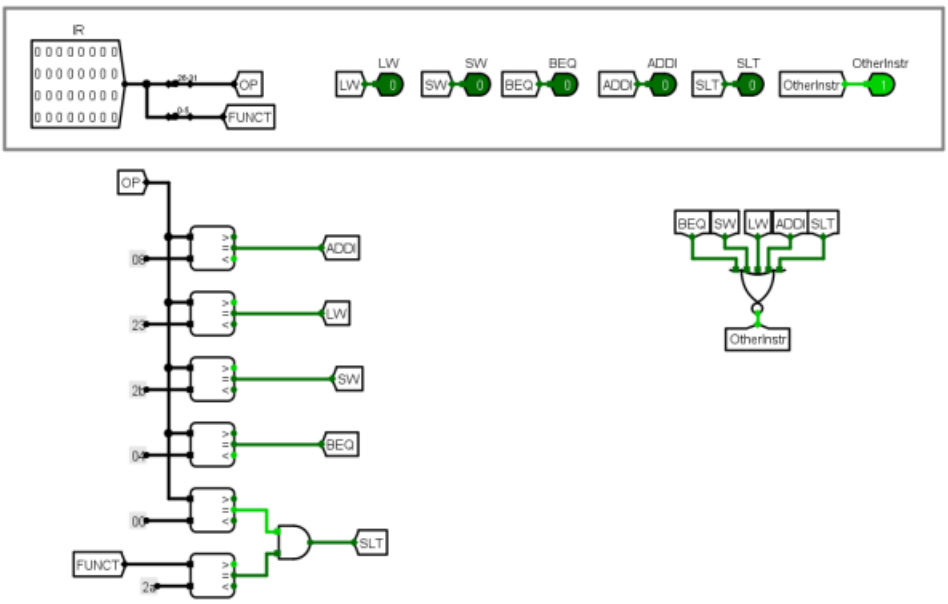


图 3-1 MIPS 指令译码器

（2） 支持中断的微程序入口查找逻辑设计

根据 1.2.2 中的微程序入口地址表，利用 Excel 得到逻辑表达式，利用 logisim 的分析电路的功能部件生成对应的电路，如图 3-2 所示。

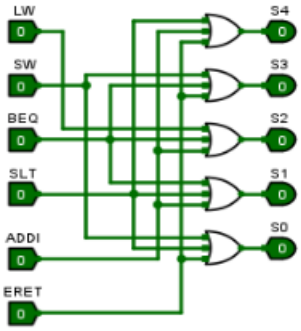


图 3-2 微程序入口地址查找逻辑电路

（3） 支持中断的微程序条件判别测试逻辑设计

在 1.2.3 中已经说明了条件判别测试的逻辑以及微程序条件判别逻辑表，利用 Excel 生成对应的逻辑表达式，利用 logisim 的电路分析功能组件，生成相应的逻辑电路，如图 3-3 所示。

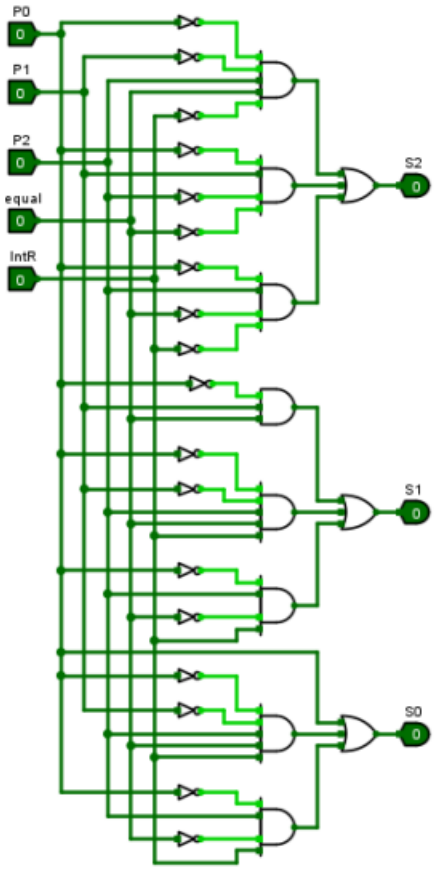


图 3-3 条件判别测试逻辑电路

（4） 支持中断的微程序控制器设计

根据 1.2.4 中的微指令-控制信号对应表，可以确定控制存储器中的内容，即每个

微指令对应的控制信号，将得到的所有控制信号导入控制存储器 ROM。

在 1.2.4 中描述了四中 PC 取值方式，分别为顺序地址、指令入口地址、beq 分支地址、中断响应入口、取指微指令程序入口，采用多路选择器实现不同信号下指令的跳转，beq 分支和中断响应入口地址为控制存储器中对应的地址，分别为 0x10,0x1a。这里采用计数器法得到一条指令中的微指令顺序执行的下一条微指令的微地址，微程序控制器电路如图 3-4 所示。

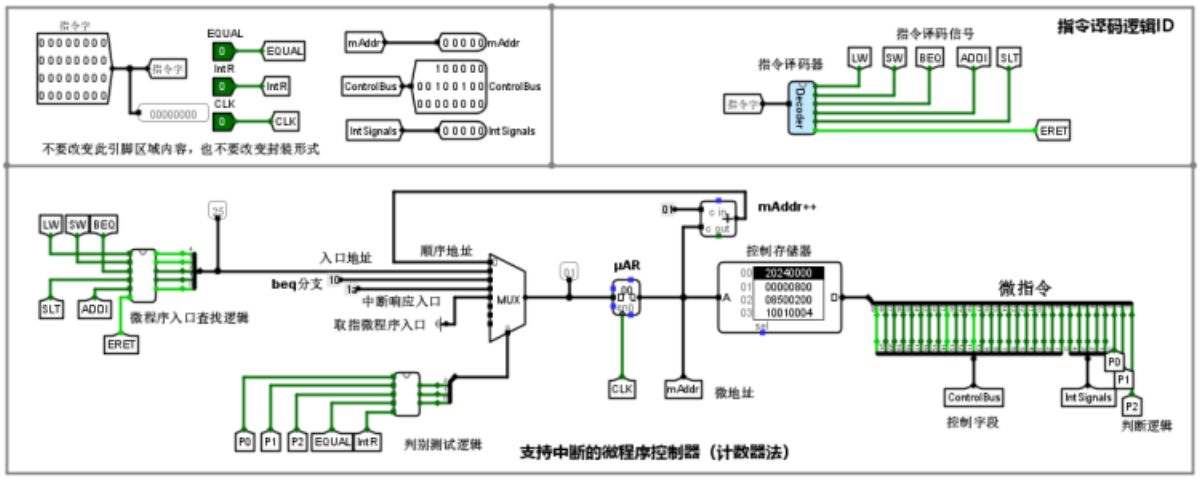


图 3-4 微程序控制器电路

（5） 支持中断的微程序单总线 CPU 设计

EPCin、EPCout 两个微命令控制 EPC 与总线的数据交互，用于保存断点。中断使能存储器 IE，用于存储使能信号，表示当前能否进行中断响应。中断控制器用于识别中断，根据识别到的中断类型，利用多路选择器实现 PC 值的更改，实现跳转到中断处理程序。当识别到中断并且处于开中断状态时产生中断请求信号。sort-5-int.hex 中包含两处中断，利用 Mars 工具获取这两个中断处理程序的入口地址，分别为 0x000400a4、0x000400ec。中断逻辑实现的电路如图 3-5 所示。

（6） 支持中断的现代时序硬布线控制器状态机设计

在 1.2.6 中已经说明了状态转换的逻辑以及硬布线状态转换表，利用 Excel 生成对应的逻辑表达式，利用 logisim 的电路分析功能组件，生成相应的逻辑电路。

（7） 支持中断的现代时序硬布线控制器设计

在 1.2.7 中简要说明了转换逻辑，在实现时将指令译码器、硬布线控制器状态机、状态寄存器、时钟 CLK 进行合理地连接，次态由现态以及部分指令信号有关。为了简化硬布线设计，利用控制存储器暂时替代由状态生成控制信号的组合逻辑电路，连接

以上部件即可以实现支持中断的硬布线控制器，如图 3-6 所示。

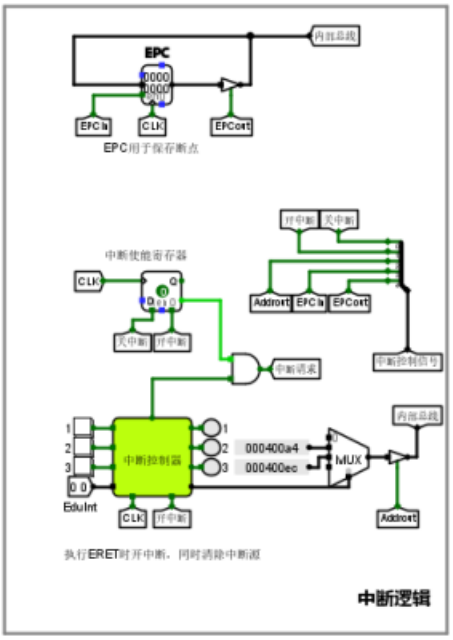


图 3-5 中断逻辑电路

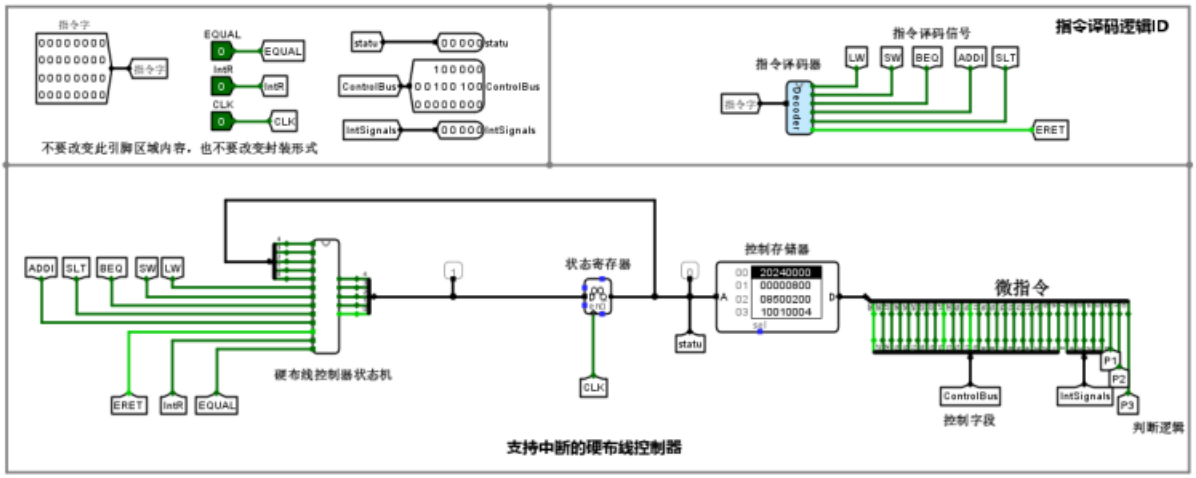


图 3-6 支持中断的硬布线控制器

1.4 故障与调试

1.4.1 条件判别测试逻辑产生错误的地址多路选择控制信号

故障现象：测试代码运行过程中跳转到错误的位置。

原因分析：填写条件判别测试逻辑表有问题，导致利用 logisim 电路分析生成的

逻辑电路结果出错。

解决方案：校正条件判别测试逻辑表，确定不同条件对应的选择信号，注意 P1=1 且 equal=1 的优先级高于 P2=1 且 intR=1，因为当 P1=1 且 equal=1 时 BEQ 指令还未执行完，不能进入中断响应，将校正好的表重新生成逻辑表达式，利用 logisim 电路分析功能生成组合逻辑电路，重新进行评测。

1.4.2 微程序控制器中产生错误的控制信号

故障现象：测试代码运行时，产生错误的控制信号，跳转到错误的位置。

原因分析：填写的微指令-控制信号对应表有问题，导致将错误的控制信息导入到控制存储器 ROM，运行过程中出现错误的跳转。

解决方案：校正填写的微指令-控制信号对应表，利用在线测试平台给出的错误提示信息，查找出现出现错误的微指令、出现错误的控制信号，将控制存储器中的内容更改后，重新进行测试。

1.4.3 在线测试中测试结果重复显示的问题

故障现象：在线测试平台中测试结果出现重复

原因分析：微地址寄存器采用了上升沿，输入来源为地址的转移逻辑的输出，开始种控制其地址更新，每一次时钟控制端的触发都会重新锁存新的微地址，从而取出下一条微指令。在现代时序中，这个时钟控制端应该在当前事中周期结束时触发。如果 CPU 中需要时序配合的控制信号是上升沿有效，那么这里的时钟控制端就应该是下降沿有效。

解决方案：状态寄存器时钟触发更换为下降沿触发，重新进行评测。

1.5 测试与分析

采用本地和在线两种方式测试 sort-5-int.hex，测试结果如图 5-1、5-2 所示，在本地测试中，按下了两次按钮 1，两次按钮 2，可以从测试结果中看出实现了数字的排序，并且在相应的位置完成了+1，-1 的操作，即 0x80 一行中数据加 1,0x90 一行中数据减 1，可以看出完成了 sort-5-int.hex 的功。结合在线测试结果，可以看出完成了所有功能。

```
000 23bd0400 2010ffff 20110000ae300200 22100001223100004ae30020022100001 22310004ae3002002210000122310004 ae3002002210000122310004ae300200
010 22100001223100004ae30020022100001 22310004ae3002002210000122310004 ae3002002010000002011001c8e130200 8e3402000274402a11000002ae330200
020 ae140200 2231fffc 12110001 1000fff7 22100004201100 1c12110001 1000fff3 1000ffff 23bd0008afb00000 afb10004 203102408e30000022100001ae300000
030 ae300004ae300008ae30000cae300010 ae300014ae300018ae30001c8fb10004 8fb00000 23bdfff8 4200001823bd0008 afb00000 afb10004 203102808e300000
040 2210ffff ae300000ae300004ae300008 ae30000cae300010ae300014ae300018 ae30001c8fb10004 8fb00000 23bdfff8 42000018000000000000000000000000
050 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
060 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
070 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
080 0000000600000005000000040000003 0000000200000001000000 ffffffff 00000000000000000000000000000000 00000000000000000000000000000000
090 0000000200000002000000020000002 0000000200000002000000020000002 00000000000000000000000000000000 00000000000000000000000000000000
0a0 fffffffe fffffffe fffffffe fffffffe fffffffe fffffffe fffffffe fffffffe 00000000000000000000000000000000 00000000000000000000000000000000
```

图 5-1 本地测试结果

图 5-2 在线评测结果

1.6 实验总结

本次实验主要完成了如下几点工作：

1） 本次实验设计了 MIPS 指令译码器、支持中断的微程序入口查找地址逻辑、条件判别测试逻辑、微程序控制器、硬布线状态机、硬布线控制器、单总线 CPU，利用 logisim 实现了各个组件的电路，完成了各个功能组件的电路的连接，完成了单总线 CPU 的数据通路，以及中断响应。

2） 本次实验中实现了各个组件的功能，实现了指令译码、入口地址查找、条件判别、中断处理等，完成了各个部件功能的评测，并利用完成的 CPU 通路实现了简单的冒泡排序程序 sort-5-int.hex。

3） 对于本次实验中出现的问题，进行了电路分析，根据分析结果，完成了对于功能的调试，解决了出现的生成错误信号等问题。

1.7 实验心得

1） 通过本次实验，首先从宏观的角度认识了 CPU 的工作原理，其次从微观的

角度实现一个个微小的逻辑电路单元，从一个个逻辑门连接成控制器，再连接成完整的 CPU，十分有成就感。对于理论课的学习到的理论知识起到了很好的巩固作用。

2） 计算机组成原理介绍了从机器指令到 CPU 硬件执行，完善了从高级语言到汇编语言，到机器指令，再到硬件中的控制信号的完整的结构，理解了汇编与底层如何联系起来的，理解了程序如何在硬件中执行起来的，以及程序是如何响应中断的。

3） 通过本次实验认识到了传统三级时序、现代时序的区别，硬布线控制器和微程序控制器的区别，体会到了微程序控制器的优缺点。

4） 通过实验中运行 sort-5-int.hex，进一步理解冯诺依曼架构“存储程序”思想。

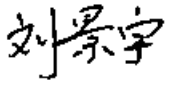
5） 通过本次实验更加熟悉 logisim，认识到 logisim 平台仿真的优势，直观性很强，处理速度也很快，仿真效果也很好。

6） 在理论课后，通过这样的闯关实验课能够巩固所学，通过 Educoder 平台实现自动评测。在线闯关实现 CPU，既有闯关带来的成就感，也有进一步理解 CPU 原理的收获感，使得实验课有趣又有料。希望实验课都可以像这样，那么做起实验来就会充满动力。

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：刘景宇 

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）	课程目标 1 工具应用（10 分）	课程目标 2 设计实现（70 分）	课程目标 3 验收与报告（20 分）	最终评定（100 分）
得分				

指导教师签字： _____