华中科技大学

2021

计算机组成原理 ·实验报告·

专　　业：计算机科学与技术

班　　级：

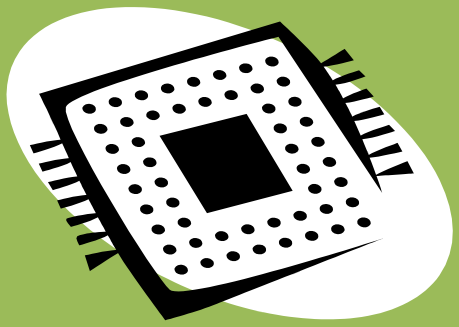
学　　号：

姓　　名：

电　　话：

邮　　件：

完成日期：



计算机科学与技术学院

目 录

1 CPU 设计实验——变长指令周期三级时序设计

1.1 设计要求

在 logisim 平台打开 RiscvOnBusCpu-3.circ 文件，在已有部件的基础上，完成单总线 CPU 中的三级时序硬布线控制器部件的设计。可支持 RISC-V 指令集中的 5 种指令：lw、sw、beq、slt、addi，指令具体功能见表 1.1。控制器实现完毕后，在主电路中的主存内载入排序文件 Sort-5-riscv.hex，测试控制器功能及 CUP 功能。

表 1.1 5 种 RISC-V 指令的功能介绍

序号	指令	汇编代码	指令类型	RTL 功能描述
1	lw	lw rd,imm(rs1)	I 型	R[rd] ← M[R[rs1] + SignExt(imm)]
2	sw	sw rs2,imm(rs1)	S 型	M[R[rs1] + SignExt(imm)] ← R[rs2]
3	beq	beq rs1,rs2,imm	B 型	if(R[rs1] == R[rs2]) PC ← PC + SignExt(imm) << 1
4	slt	slt rd,rs1,rs2	R 型	If (rs1 < rs2) R[rd] ← 1 else R[rd] ← 0
5	addi	addi rd,rs1,imm	I 型	R[rd] ← R[rs1] + SignExt(imm)

1.2 方案设计

1.2.1 三级时序硬布线控制器整体设计

传统三级时序硬布线控制器主要由指令译码器、时序发生器、硬布线控制器组合逻辑单元这几个部分组成，其逻辑组合模型如图 1.1 所示。

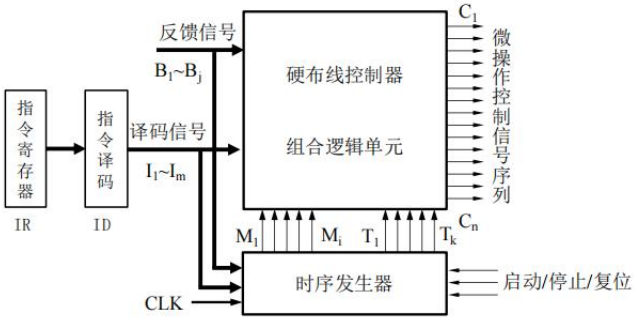


图 1.1 三级时序硬布线控制器整体设计模型

其中指令译码器部件用于产生译码信号，用来解析当前在指令寄存器 IR 中的指令操作码；时序发生器则根据译码信号进行对应状态周期和节拍电位的输出；硬布线控制器组合逻辑单元根据译码信息和状态周期及节拍电位，进行对应的微操作控制信号序列输出，用于完成一个时钟周期内的微操作。

1.2.2 指令译码器部件设计

根据接收到的指令寄存器 IR 中的数据，将指令中的操作码位和扩展位与 5 种指令的操作码位和扩展位比较，对应相等的指令即解析得到的指令。由于 RISC-V 指令中的这 5 类格式操作码和扩展位位置固定，操作码占低 7 位，扩展码占 12-14 位，可设置 5 个并行比较器进行比较，输出对应译码信号。

1.2.3 时序发生器部件设计

时序发生器则是根据译码信息和时钟信号，进行对应状态周期和节拍电位的输出。其中包含 3 个主要部件：FSM 状态机组合逻辑、输出函数组合逻辑以及状态寄存器，三者的组合逻辑模型见图 1.2。

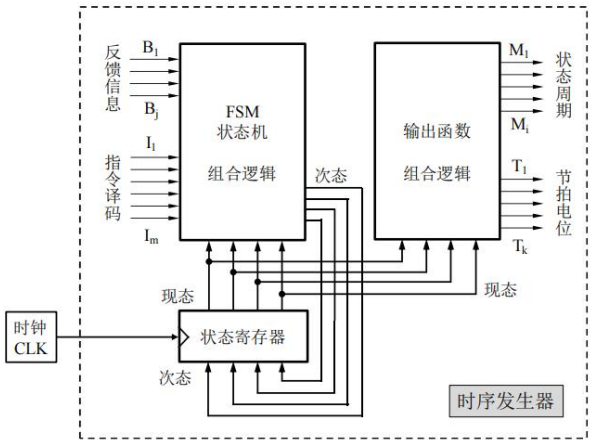


图 1.2 时序发生器的内部逻辑设计模型

FSM 状态机组合逻辑中，包含 8 个状态，其中 S_0-S_3 为取指周期状态，S_4-S_5 为计算周期状态，而 S_6-S_8 为执行周期状态，见图 1.3。这 8 个状态为公共状态，即每个指令共同享有这些状态，而根据指令信号的输入不同，状态跳转不一，实现变长指令周期，节约部分周期时间。在 S_3 状态时，根据译码信号进行不同状态的跳转。

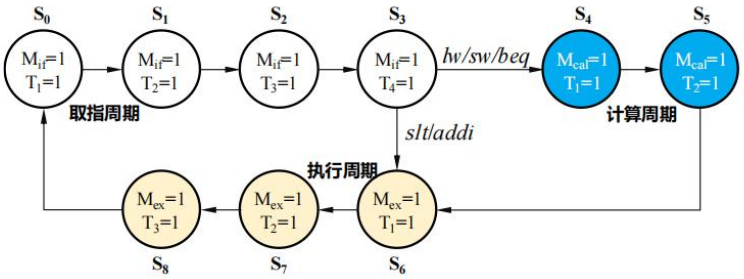


图 1.3 变长指令周期三级时序状态机设计

而输出函数组合逻辑则根据当前现态进行状态周期和节拍电位的输出，图 1.3 中状态内显示相应的时序输出。

状态寄存器则进行现态与次态的更替，当接收到一个时钟下降沿时，更替状态。此时使用下降沿而不用上升沿的原因在于，CPU 其他部件寄存器，如 AR，IR 等使用上升沿进行锁存，锁存需要一定的持续时间，而状态转变需要在状态稳定时进行切换，故使用下降沿进行状态改变。

1.2.4 硬布线控制组合逻辑单元部件设计

硬布线控制组合逻辑根据译码信息及时序信息进行对应的微操作控制信号序列的输出。取指周期为公共周期，不受输入译码信号的影响。取值结束后，根据不同的指令输入，进行对应的微操作控制信号序列的输出。一个节拍进行一次信号输出，由于单总线会有互斥性微操作，故执行一条指令需要多个节拍下的数条微操作协同完成。

1.3 实验步骤

1.3.1 实现指令译码器部件

首先需要对指令的操作码和扩展码进行提取，低 7 位为操作码部分，12-14 位为扩展位，值得注意的是 RISC-V 指令的低 2 位为‘11’，表明机器级特权，是 RISC-V 硬件平台唯一必须的特权级。在此约定下，只需判断 3-6 位即可。

其次，将提取的操作码与扩展位与指定的 5 种指令的操作码和扩展位进行并行比较。通过查表，易得到 lw、sw、beq、slt 和 addi 的具体指令操作码和扩展码。当与上述 5 种不一致时，输出 OtherInstr，表明该指令不属于这 5 种指令。

1.3.2 实现时序发生器部件

时序发生器的设计包括三个环节：FSM 状态机设计、输出函数组合逻辑电路设计以及各部件的组合设计。

（1） FSM 状态机设计

根据图 1.3 中的转换图可知，5 种指令在 S_3 后跳转存在差异，当收到译码信号 LW、SW 以及 BEQ 时，跳转到状态 S_4，收到译码信号 SLT 和 ADDI 时，跳转到状态 S_6。而其他状态跳转不受译码信号影响。填写 1.单总线 RISC-V 三级时序产生器逻辑自动生成(2021-11-11).xlsx 表格完成上述过程。

（2） 输出函数组合逻辑设计

由于时序发生器是 Moore 型电路，故输出只与现态有关，如图 1.3。根据现态可将时序周期分成取指周期 M_{if}、计算周期 M_{cal}、计算周期 M_{ex}。取指周期包含 4 个节拍，依次输出 T1、T2、T3、T4；计算周期包含 2 个节拍，依次输出 T1、T2；执行周期包含 3 个节拍，依次输出 T1、T2、T3。填写 1.单总线 RISC-V 三级时序产生器逻辑自动生成(2021-11-11).xlsx 表格完成上述过程。

（3） 时序发生器各部件的组合设计

根据 Moore 型电路，组装各个部件。Moore 型电路输出只与现态有关，故将输入端连接状态寄存器的现态输出端。而状态寄存器的状态转换有赖于 FSM 状态机的实现，将状态寄存器的输入端与 FSM 状态机的输出端相连，设置寄存器触发方式为下降沿，当收到下降沿信号时，实现状态切换。具体组合见图 1.4。

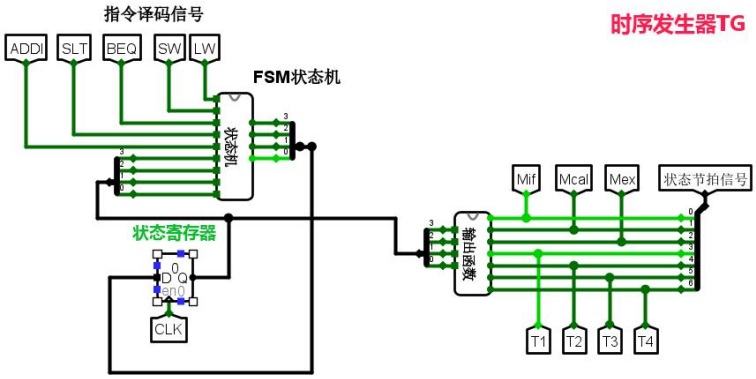


图 1.4 变长指令周期三级时序发生器设计

1.3.3 实现硬布线控制组合逻辑单元部件

由时序发生器输出的时序信号和译码信号为指示，输出对应的微操作控制信号序列。其中取指周期在各指令周期内相同，包含 4 个节拍，取指周期的流程及控制信号如表 1.2。值得注意的是，暂存器 Z 将新 PC 结果送入 PC 寄存器与读数据放入 DR 寄存器的操作可以同时进行，因为 Z_{out} 将数据送入内总线，而 DRE_{in} 则是读取外总线的数据，两者可以相容。

表 1.2 取指周期的流程及控制信号

节拍	操作	控制信号
1	PC→AR；PC→X	$PC_{out}=AR_{in}=X_{in}=1$
2	X+4→Z	Add4=1
3	Z→PC；M[AR]→DR	$Z_{out}=PC_{in}=1$ $Read=DRE_{in}=1$
4	DR→IR	$DR_{out}=IR_{in}=1$

对 beq 指令周期中的计算周期进行具体说明。第一个节拍将 R[rs1]送入暂存器 X；而第二个节拍将 R[rs2]送入总线，且将状态信息送入 PSW 中。此时并未进行加减操作，但仍有 equal 信号产生。因为在 ALU 部件中，equal 信号独立与其他运算操作，当输入两个操作数时即可生成对应的 equal 信息。

1.4 故障与调试

1.4.1 equal 信号生成问题

故障现象：执行 beq 指令时需将 equal 信号传入 PSW，此时出现状态生成条件的差异性，即是否进行运算操作。

原因分析：在书本资料中，需要进行运算操作如 sub，add 等产生 equal 信号。而在此次设计中，在 ALU 运算器部件内，equal 信号的产生与运算操作无关，见图 1.5。故无需进行运算操作，即可得到 equal 信号。

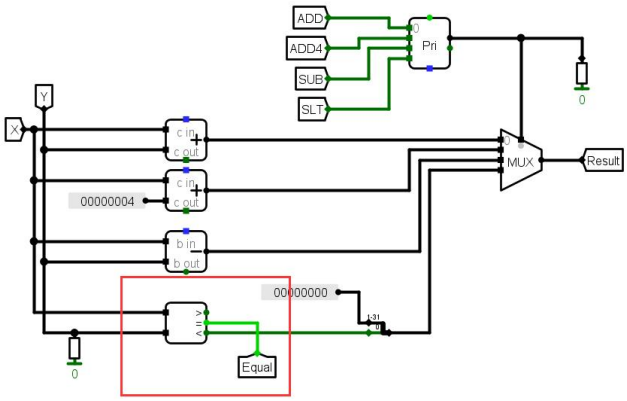


图 1.5 运算器 ALU 部件内部 equal 信号产生条件

解决方案：填写组合逻辑真值表时，在填写 beq 指令周期内的计算周期第 2 个节拍时，不需要进行 add 或 slt 运算，只需输出 R_{out}、rs1/rs2 及 PSW_{in} 控制信号即可。

1.4.2 状态寄存器触发方式的问题

故障现象：在设计时序发生器时，将状态寄存器设置为上升沿，出现如图 1.6 的错误。

预期输出

Cnt	Instr	equal	M123	T1234	cBus	ErrBit
00	2004aa03	0	100	1000	201200	xx
01	2004aa03	0	100	0100	000008	xx
02	2004aa03	0	100	0010	082802	xx
03	2004aa03	0	100	0001	100080	xx
04	2004aa03	0	010	1000	040200	xx
05	2004aa03	0	010	0100	020010	xx
06	2004aa03	0	001	1000	081000	xx

实际输出　展示原始输出

Cnt	Instr	equal	M123	T1234	cBus	ErrBit	
00	2004aa03	0	100	1000	201200	xx	
00	2004aa03	0	100	0100	000008	15	Error!
01	2004aa03	0	100	0100	000008	xx	
01	2004aa03	0	100	0010	082802	13	Error!
02	2004aa03	0	100	0010	082802	xx	
02	2004aa03	0	100	0001	100080	14	Error!
03	2004aa03	0	100	0001	100080	xx	

图 1.6 状态寄存器触发方式错误

原因分析：在 CPU 的其他寄存器如 AR、IR 进行锁存时，触发方式设置为上升沿，而锁存需要一定的时间，若此时进行状态跳转，会出现锁存错误，导致寄存器数据错乱，故要将锁存触发方式与状态跳转触发方式做出区分。

解决方案：将状态寄存器的触发条件设置为下降沿。

1.5 测试与分析

在 RAM 中加载 sort-5-riscv.hex 程序，ctrl+k 自动运行，程序应该运行至 0x81d 节拍停下，指令计数为 251，注意最后一条指令是一条 beq 分支指令，会跳回当前指令

继续执行，是死循环。

变长指令周期的状态周期电位和节拍电位信号时序如图 1.7 所示。其中第一个方框内为取指周期，第二个方框内为计算周期，第三个方框内为执行周期。满足周期数和节拍数。

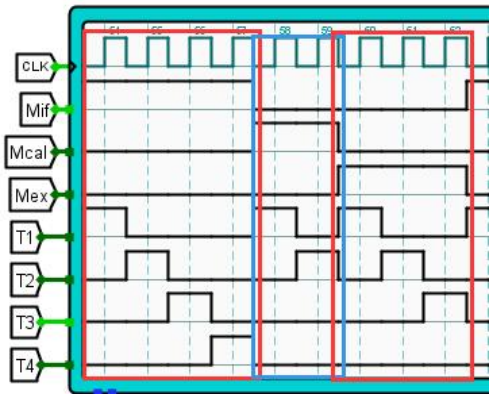


图 1.7 一个指令周期的时序信号

最终的节拍数、指令数和指令如图 1.8 所示，满足测试预估结果。

图 1.8 一个指令周期的时序信号

排序结果如图 1.9 所示，满足排序要求。

060	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
070	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
080	00000006	00000005	00000004	00000003	00000002	00000001	00000000	ffffffff
090	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
0a0	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
0b0	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000

图 1.9 最终排序结果

2 CPU 设计实验——现代时序中断机制设计

2.1 设计要求

在 logisim 平台打开 RiscvOnBusCpu-1.circ 文件，在现代时序微程序控制器的基础上，增加中断处理机制，可实现多个外部按键中断事件的随机处理。除支持 RISC-V 指令集中的 5 种指令：lw、sw、beq、slt、addi 外，还需支持中断返回指令 eret。需要设计支持中断的微程序控制器及中断控制器的中断逻辑。控制器和中断逻辑实现完毕后，在主电路中的主存内载入带中断处理的排序文件 Sort-5-int-riscv.hex，测试微程序控制器及中断逻辑功能。

2.2 方案设计

2.2.1 支持中断的微程序单总线 CPU 设计

为实现支持中断的微程序单总线 CPU，除了文件内已有的部件外，还需实现支持中断的微程序控制器以及中断控制器的中断逻辑。

其中支持中断的微程序控制器采用计数器法实现，见图 2.1，图中省略了指令译码器部分，相关部件的详细设计查看 1.2.2 指令译码器部件设计。

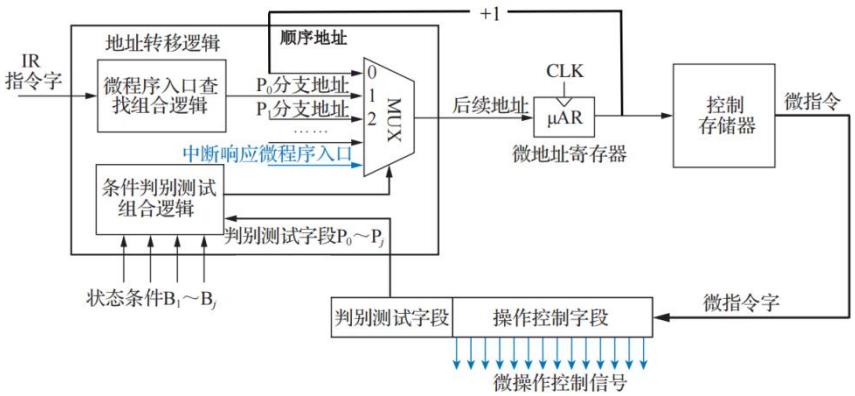


图 2.1 支持中断的计数器法微程序控制器

中断控制器的中断逻辑实现则以中断控制器为核心，根据中断信号产生对应的中断请求，传送给微程序控制器做出中断响应。除此之外，还需支持断点保存、断点返

回和中断程序入口地址的传送。

2.2.2 支持中断的微程序控制器设计

支持中断的微程序控制器采用计数器法寻址方式，包含 3 个主要部分：地址转移逻辑电路，控制存储器，微地址寄存器。见图 2.1。

地址转移电路则由微程序入口查找组合逻辑，条件判断测试组合逻辑以及选路器组成。微程序入口查找组合逻辑提供指令的执行入口地址。而条件判断测试组合逻辑提供选路控制信号，用于选择正确的下址。

2.2.3 中断控制器的中断逻辑设计

中断逻辑设计以中断控制器为核心，接收中断信号，发出相应的中断请求信号。提供 mEPC 寄存器保存断点地址。保存中断程序入口地址，在中断响应阶段提供对应的中断程序入口地址。

2.3 实验步骤

2.3.1 实现支持中断的微程序控制器

实现支持中断的微程序控制器主要包括三个环节：地址转移逻辑的实现，控制存储器的实现，逻辑部件的组装。

（1）地址转移逻辑的实现

地址转移逻辑包含微程序入口查找逻辑和条件判别测试逻辑。

微程序入口查找逻辑 可根据译码信号查找对应指令的入口地址。在取指令阶段后进行特定指令入口地址的载入，实现特定指令的后续阶段。如 lw 指令的入口地址为 4（十进制），sw 指令的入口地址为 9（十进制）等。

条件判别测试逻辑 则需要判别测试字段和状态条件的参与。本次 CPU 设计中包含 3 种判别测试字段，即 P0、P1、P2，状态条件包括中断请求信号 IntR 和 equal 信号。

P0 表示送入对应指令的入口地址，在取指令的阶段的最后一个状态时进行标志。而 P1 表示 beq 跳转分支判断，当接收到 equal 信号时，进行后续跳转操作，若无 equal 信号则进行请求判断，在 beq 指令的第二个节拍时标志，与此同时也要对 P3 进行标志。

P3 表示指令结束，需要与中断请求信号 IntR 协同进行判断。若此时有中断请求时，需要跳转到中断响应状态，即送入中断响应微程序入口地址；若此时无中断请求，则送入取指微程序入口地址。每次指令微程序完毕后都需进行中断请求信号判断，也即标志 P2。

多路选择器 0 号端口送入顺序地址，在微程序执行阶段过程内无需标志判别字段；1 号端口送入指令微程序入口地址，在取指阶段的最后一个节拍时标志，送入下一节拍的微地址；2 号端口送入 beq 分支地址，在 beq 指令微程序的第 2 个节拍时，进行判断，送入的逻辑为 P1&equal；3 号端口送入中断响应入口地址，在每条程序结束时进行判断，送入条件为 P2&IntR，但考虑到 beq 在第二节拍时有跳转判断，当 equal 不为 1 时，也需要进行中断判断，故更改送入条件为 P2&IntR&~(P1&equal)，来满足这一情况。4 号端口则送入取指微程序入口地址，判断条件为 P2&~IntR&~(P1&equal)。

（2）控制存储器的实现

控制存储器完成微程序的存储，根据微地址寄存器送入的微地址，输出对应的微指令字。微指令字包括操作控制字段和判别测试字段。

对于判别测试字段的构建在上述条件判别测试组合逻辑的构建中已做出具体的描述，在此不做赘述。

而操作控制字段中除 eret 和中断响应微程序外，其他的控制信号的输出与 1.3.3 节的输出相同。在此讨论 eret 和中断响应微程序的操作字段构建。

eret 微程序只含一条微指令，即将保存在 mEPC 寄存器中的断点地址送入 PC 寄存器，并开中断。相应的控制信号为 EPC_{out}=PC_{in}=STI=1=ClrInt，值得注意的是，此次 CPU 设计中将开中断信号与 ClrInt 信号合并，即达到开中断时清空中断源的效果。

中断相应微程序包含两个节拍，见表 2.1。

表 2.1 中断响应阶段的流程及控制信号

节拍	操作	控制信号
1	0→IE；PC→EPC	CLI=PC_{out}=EPC_{in}=1
2	中断程序入口→PC	$Addr_{out}$=PC_{in}=1

需要进行关中断和保存断点操作。且需要将中断程序入口送入 PC 中。保存断点和中断程序入口的送入需依赖中断逻辑实现。

（3）逻辑部件的组装

对地址转移逻辑进行组装。多路选择器的选择端为条件判别测试逻辑的输出，而输入端则对应判别逻辑的选择信号，具体对应关系在上述条件判别测试组合逻辑的构建中已做说明。对于顺序地址输入端口，需要由现微地址进行加 1 操作得到。控制存储器中存放各微程序的微指令字。微地址寄存器输入端为下一微地址，输出端为当前微地址，而寄存器触发条件与三级时序硬布线的状态寄存器相同，为下降沿，原因相同。原因已在 1.2.3 时序发生器部件设计中做出说明。

上述逻辑的具体实现见图 2.2。

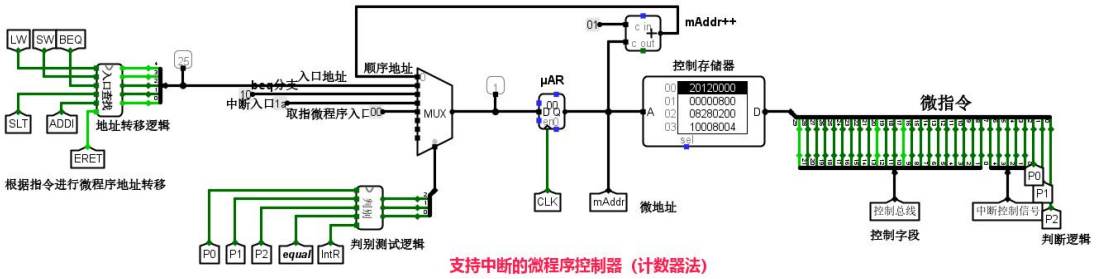


图 2.2 支持中断的计数器法微程序控制器

2.3.2 实现中断控制器的中断逻辑

中断逻辑实现同样包括 3 个部分：保存和返回断点部分、中断请求发生部分、中断程序入口的传送部分。

保存和返回断点部分以 mEPC 寄存器为核心，当接收到 $mEPC_{in}$ 信号时，内总线数据为断点地址，进行保存。而输出端由三态门控制，当接收到 $mEPC_{out}$ 信号时，需要返回断点地址，三态门打开，将断点地址传输至内总线中，由 PC 接收。具体保存和返回断点部分见图 2.3。

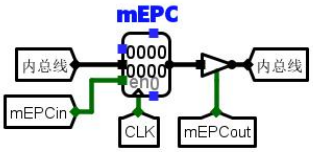


图 2.3 保存和返回断点逻辑

中断请求发生部分具体实现中断请求信号的输出。中断控制器部件根据按键的按下产生中断信号。在当前中断使能寄存器 IE 为开中断状态时，才进行中断请求信号输出；若为关中断状态，则屏蔽当前中断信号。具体由几个 D 触发器实现，关中断与置 1 端连接，开中断与置 0 端连接，而中断屏蔽信号与触发器非值状态连接。

中断程序入口的传送部分实现中断程序入口地址向内总线的传送，传送到 PC 寄存器。此 CPU 设计中包含 2 个中断服务程序，通过中断控制器的 IntNo 输出对应的序号，用来控制多路选择器的输出。多路选择器的 1 号端口对应中断程序 1 的入口地址，相应的 2 号端口对应中断程序 2 的入口地址。输出端由三态门控制，当 $Addr_{out}$ 控制信号到来时，将对应中断程序入口地址输入至内总线。

两部分逻辑部件见图 2.4。中断程序的入口地址使用 rars1_3_1.jar 打开 Sort-5-int-riscv.asm 文件显示，可知中断程序 1 入口地址为 0x004000a4，中断程序 2 入口地址为 0x004000ec。

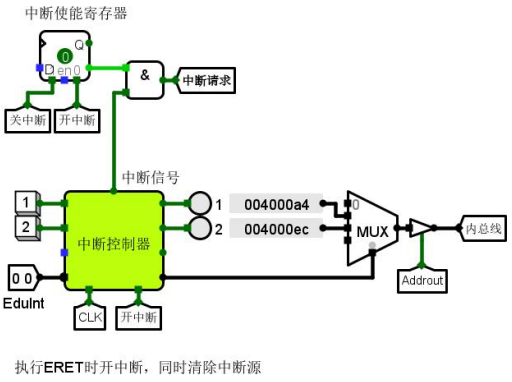


图 2.4 中断请求发生部分和中断程序入口的传送部分逻辑

2.4 故障与调试

2.4.1 中断请求信号生成问题

故障现象：实现中断请求信号生成部件时，将中断使能寄存器的输出端设置成触发器状态，中断信号始终屏蔽。

原因分析：如图 2.5，由于 D 触发器的初始状态为 0，处于开中断状态，中断屏蔽失效，应该将状态的非值作为输出值。若将输出设为状态值，中断信号则将持续被屏蔽，无法得到响应。此问题可归结为粗心问题。

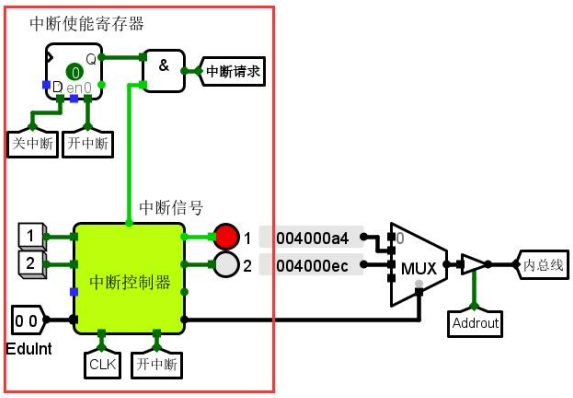


图 2.5 中断信号被持续屏蔽

解决方案：中断使能寄存器的输出端设置为 D 触发器的状态非值。

2.5 测试与分析

在 RAM 中加载 sort-5-int-riscv.hex 程序，ctrl+k 自动运行，在不产生中断的情况下，程序应该运行至 0x7c8 节拍停下，指令计数为 251，注意最后一条指令是一条 beq 分支指令，会跳回当前指令继续执行，是死循环。

在不产生中断情况下，进行测试，排序结果如图 2.6 所示，达到预期排序目的。

```
060 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
070 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
080 00000006 00000005 00000004 00000003 00000002 00000001 00000000 ffffffff
090 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0a0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

图 2.6 无中断情况下的排序结果

在产生中断 1 和中断 2 的中断情况下，进行测试。查看汇编程序可知，中断程序 1 将字节地址 0x240，字地址 0x90 下的数据进行加 1 操作，赋值给字地址 0x90 存储单元及其后 7 个存储单元，总共 8 个存储单元。中断程序 2 将字节地址 0x280，字地址 0xA0 下的数据进行加-1 操作，赋值给字地址 0xA0 存储单元及其后 7 个存储单元，总共 8 个存储单元。对两个中断程序各执行一次，执行结果和主程序排序结果如图 2.7 所示，达到预期分析结果。

```
080 00000006 00000005 00000004 00000003 00000002 00000001 00000000 ffffffff 00000000
090 00000001 00000001 00000001 00000001 00000001 00000001 00000001 00000001 00000000
0a0 ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff 00000000
```

图 2.7 中断情况下的中断程序执行结果和排序结果

3 总结与心得

3.1 实验总结

本次 CPU 设计实验包含变长指令周期三级时序设计和现代时序中断机制设计，主要完成了如下几点工作：

1） 实现了 RISC-V 指令译码器的设计。

2） 实现了变长指令周期三级时序发生器中的状态机设计。

3） 实现了变长指令周期三级时序发生器中的输出函数设计。

4） 实现了变长指令周期三级时序发生器的内部逻辑连接。

5） 实现了变长指令周期三级时序硬布线控制器组合逻辑单元的设计。

6） 实现了变长指令周期三级时序硬布线控制器的内部逻辑连接。

7） 完成了变长指令周期三级时序单总线 CPU 的设计。

8） 实现了支持中断的微程序入口查找逻辑。

9） 实现了支持中断的微程序条件判别测试逻辑。

10）实现了支持中断的控制存储器内容的填充。

11）实现了支持中断的计数器法微程序控制器的内部逻辑连接。

12）实现了对单总线数据通路中与中断相关的硬件模块的添加。

13）完成了支持中断的微程序单总线 CPU 的设计。

3.2 实验心得

1） 通过此次 CPU 设计实验，让我熟悉了 CPU 的两种控制器类型，硬部件控制器和微程序控制器，也让我体会到了两者的明显区别。变长指令周期硬布线控制器由特定的时序信号控制，由译码信号、状态周期电位信号和节拍电位信号共同指导微操作控制信号序列的输出。而微程序控制器则抛弃三级时序思路，仅按单节拍和微地址来产生对应的微指令字，微程序存储在控制存储器中，易扩展。

2） 通过此次实验，让我熟悉了 logisim 的许多便捷部件的使用，如隧道的使用，

探针的使用。以及许多逻辑部件的使用，如三态门的使用，优先编码器的使用和存储器的使用。受益匪浅，为今后的组原课设提供了软件操作知识储备，在设计中如鱼得水。

3） 深刻理解了 CPU 各部件的协同工作原理，由控制器出发，产生控制 CPU 全局的控制信号。以小见大，见微知著。通过控制器了解的单总线 CPU 的各部件运行逻辑，设计十分巧妙。不仅掌握了控制器的设计原理，还通过控制信号，对每个部件的输入输出、互斥相容进行了深刻的理解。

4） 收获最大的还是对中断的理解更加深刻了。在汇编的学习中，仅仅学到了汇编层次上中断的实现，而通过此次支持中断的单总线 CPU 设计理解了机器级的中断处理逻辑，并且最终的测试环节将汇编与硬件紧密联系。理解了两者之间的关联。

5） 另外，通过 educoder 的通关，体会到了设计的乐趣，满足感颇丰。另外，通过关卡的逐步引导，也让我理解了控制器内部组件的逻辑组成关系。这种学习方式对初步接触 CPU 设计的同学来说尤其舒适。

6） 最后，发现中断处理 Sort-5-int-riscv.asm 文件内的注释有一个小错误，即对中断程序的执行功能和地址位置的说明有错误，不过作为一个查错点也十分合适，还可以考察汇编代码与硬件实现的联系。

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,吴非，肖亮.计算机组成原理.北京:人民邮电出版社，2021 年.

[4] 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.

• 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:

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: _____