

《计算机组成原理实验》

(第六次)

厦门大学信息学院软件工程系 曾文华

2022年5月26日

目录

一、验证实验

1. 支持中断的单总线结构 MIPS 处理器（硬布线控制器）
2. 支持中断的单总线结构 MIPS 处理器（微程序控制器）

验证实验：有现成电路和部分程序

请同学们编写程序，在电路上验证，并对实验结果进行分析

二、设计实验

支持中断的单总线结构 MIPS 处理器（微程序控制器，增加了add指令）

设计实验：需要同学们自己设计电路和程序

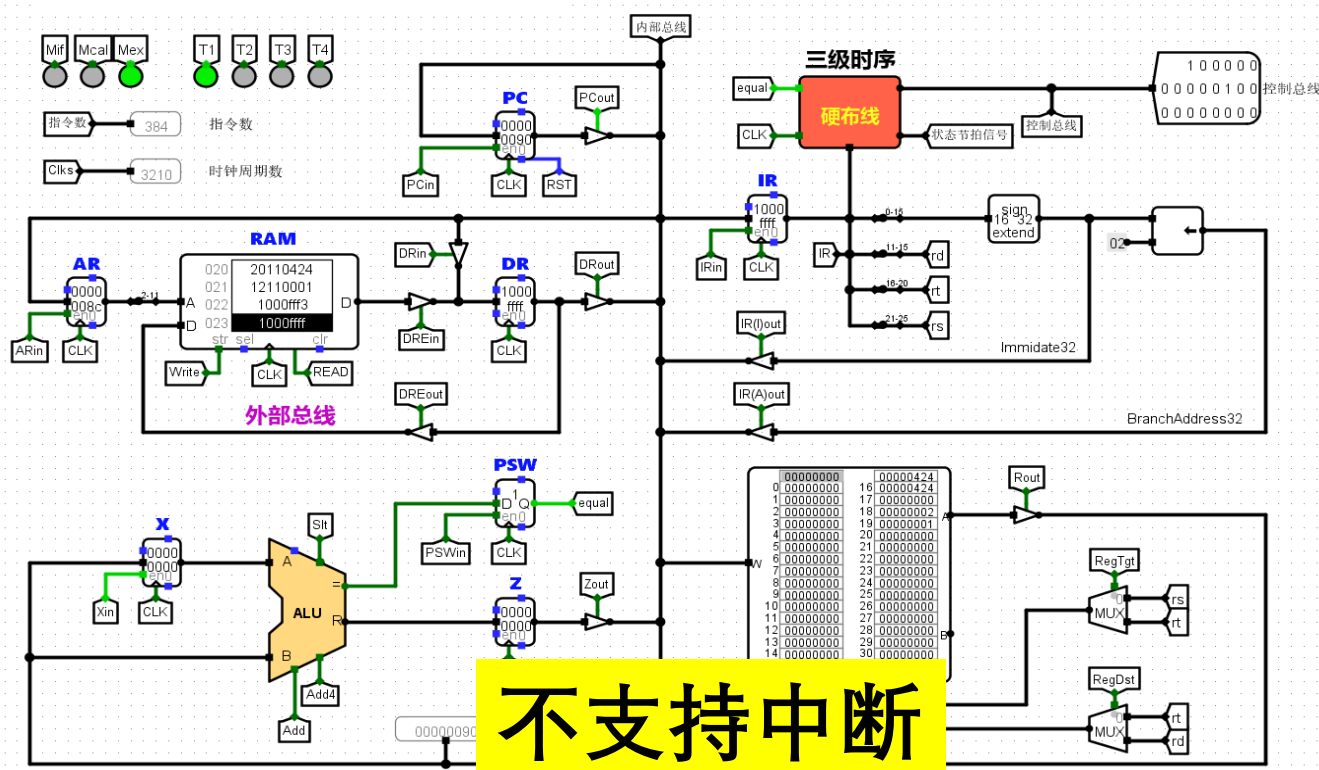
实验要求

- 实验结束后2周内（6月9日晚上24点前）提交实验报告（Word文档），同时提交相应的设计文件和程序（完成实验目录中“设计实验”的内容）。

一、验证实验

1、支持中断的单总线结构 MIPS 处理器 (硬布线控制器)

- (1) (实验五) 单总线结构 MIPS 处理器 (三级时序、变长指令周期、硬布线控制器、5条指令) 数据通路

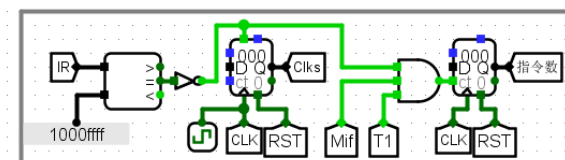
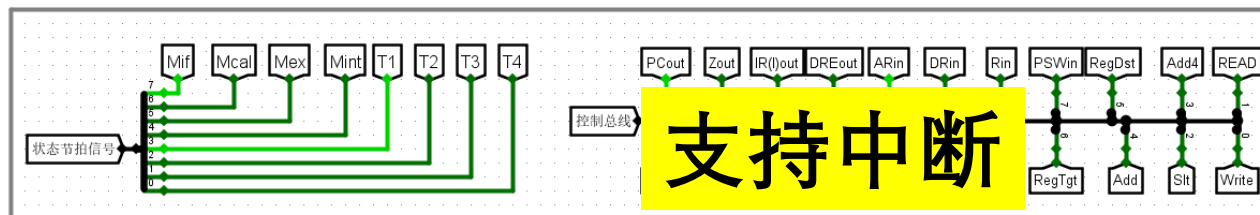
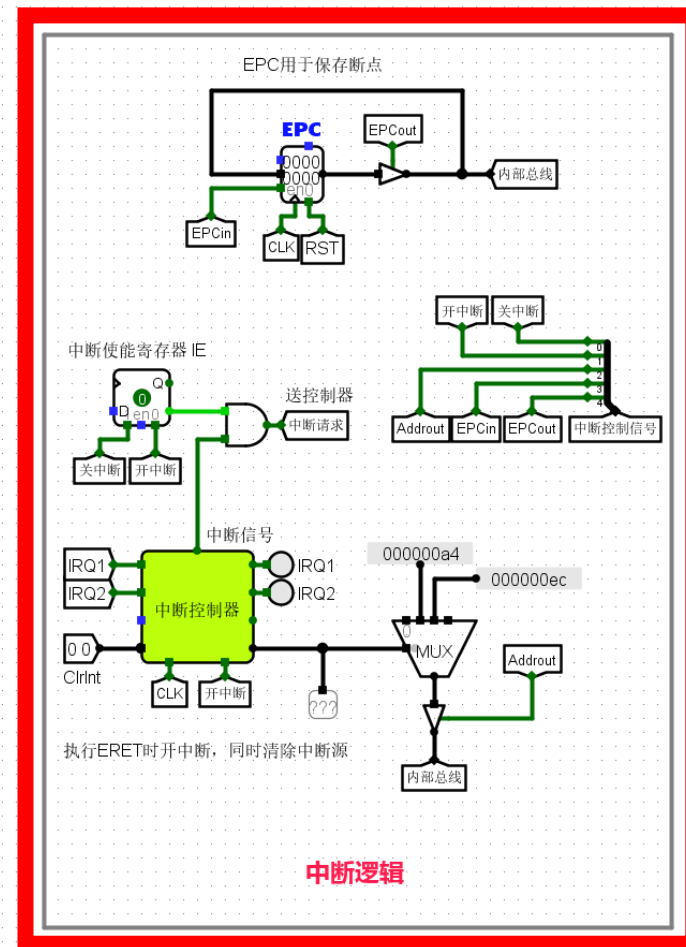
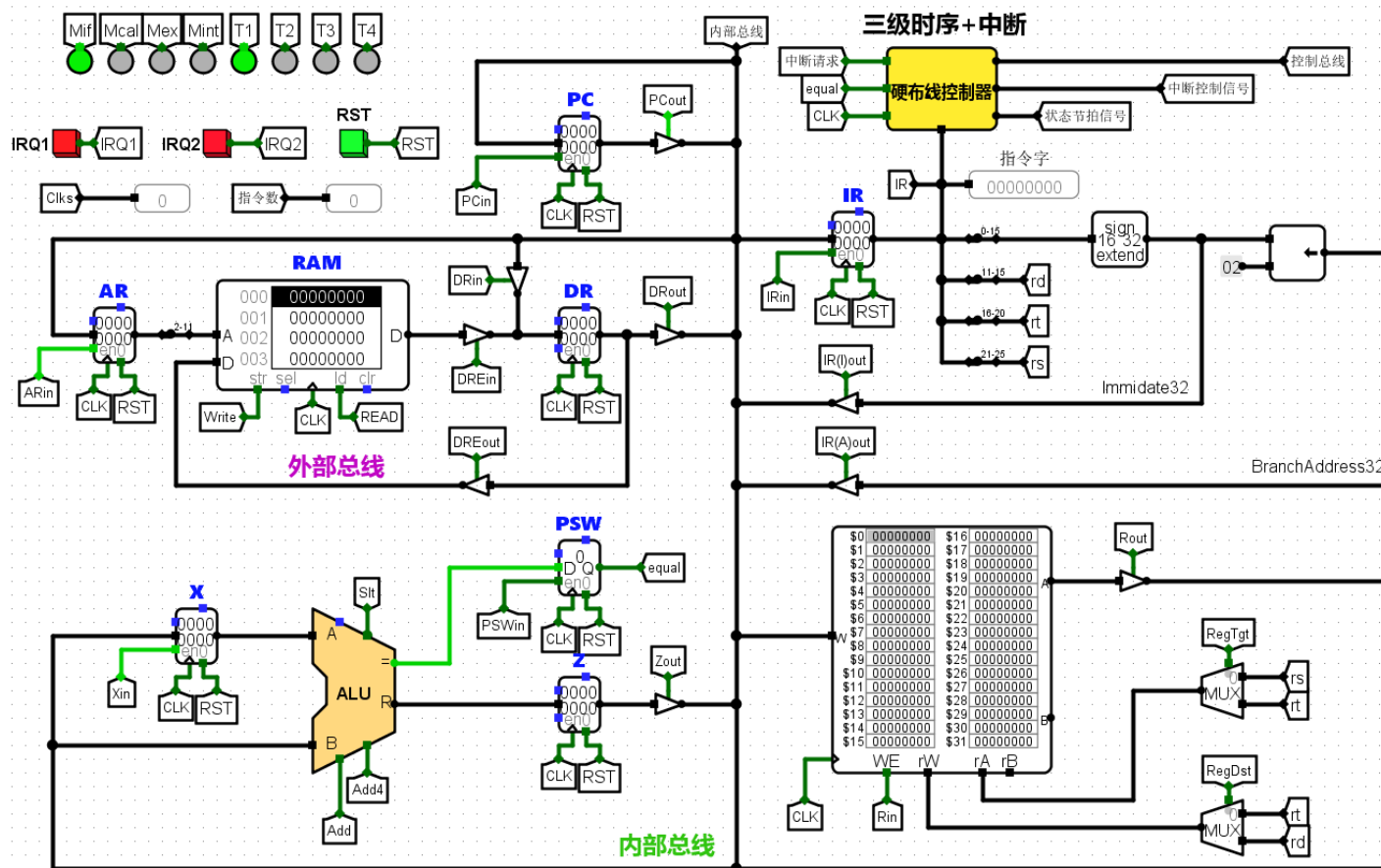


5条指令

序号	MIPS指令
1	<u>lw</u> rt,imm(rs)
2	<u>sw</u> rt,imm(rs)
3	<u>beq</u> rs,rt,imm
4	<u>addi</u> rt,rs,imm
5	<u>slt</u> rd,rs,rt

- (2) **支持中断控制**的单总线结构 MIPS 处理器（三级时序、变长指令周期、硬布线控制器、5条指令）数据通路

单总线结构 MIPS 处理器数据通路（三级时序+中断）



中断控制逻辑电路

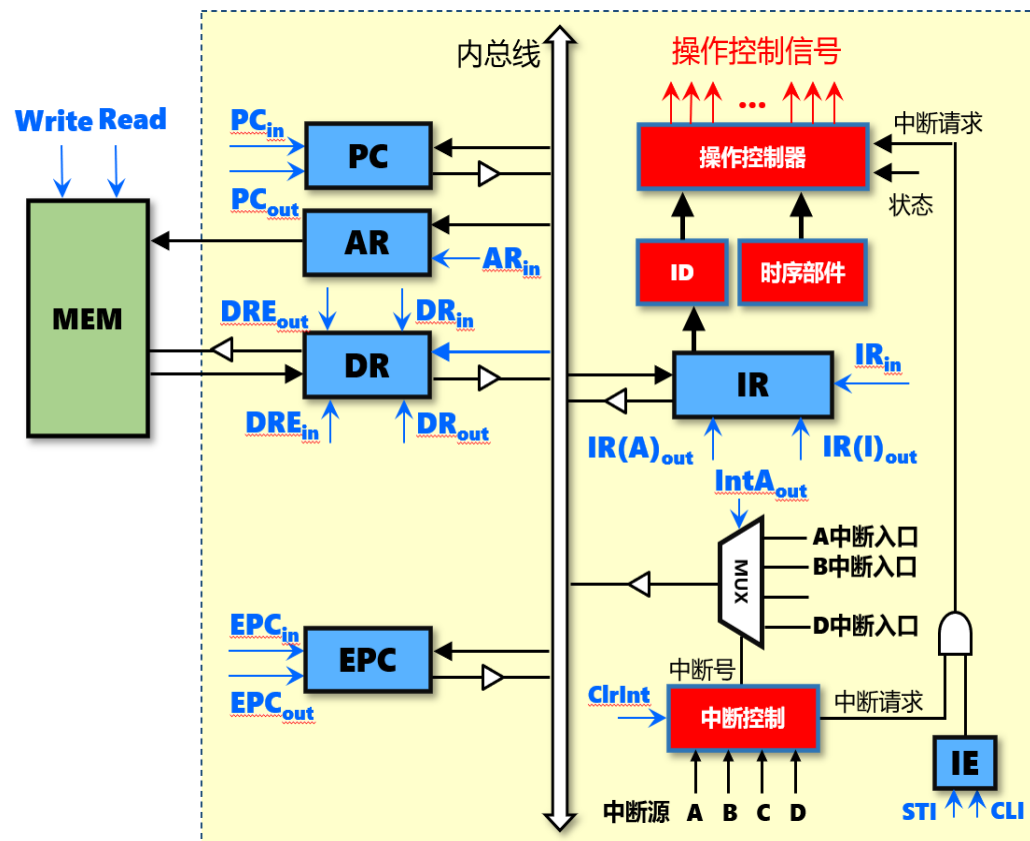
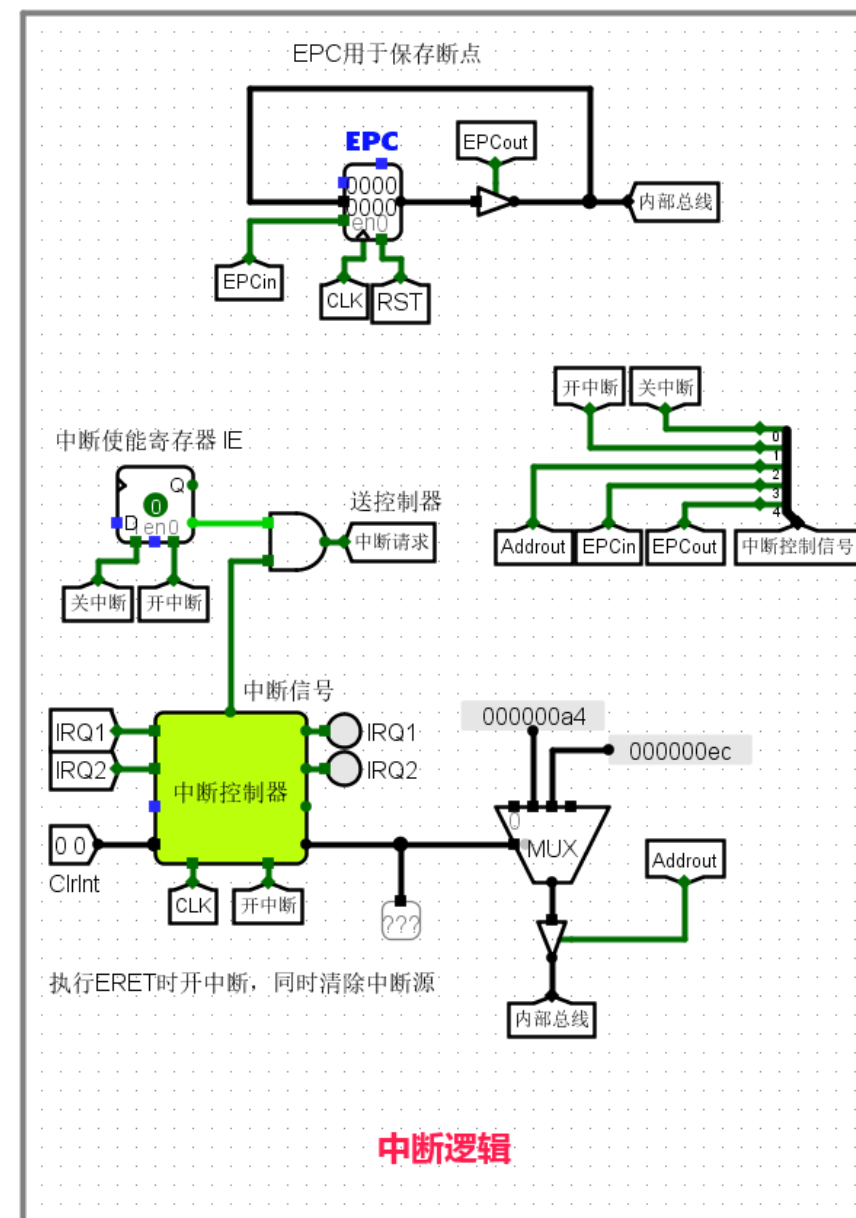
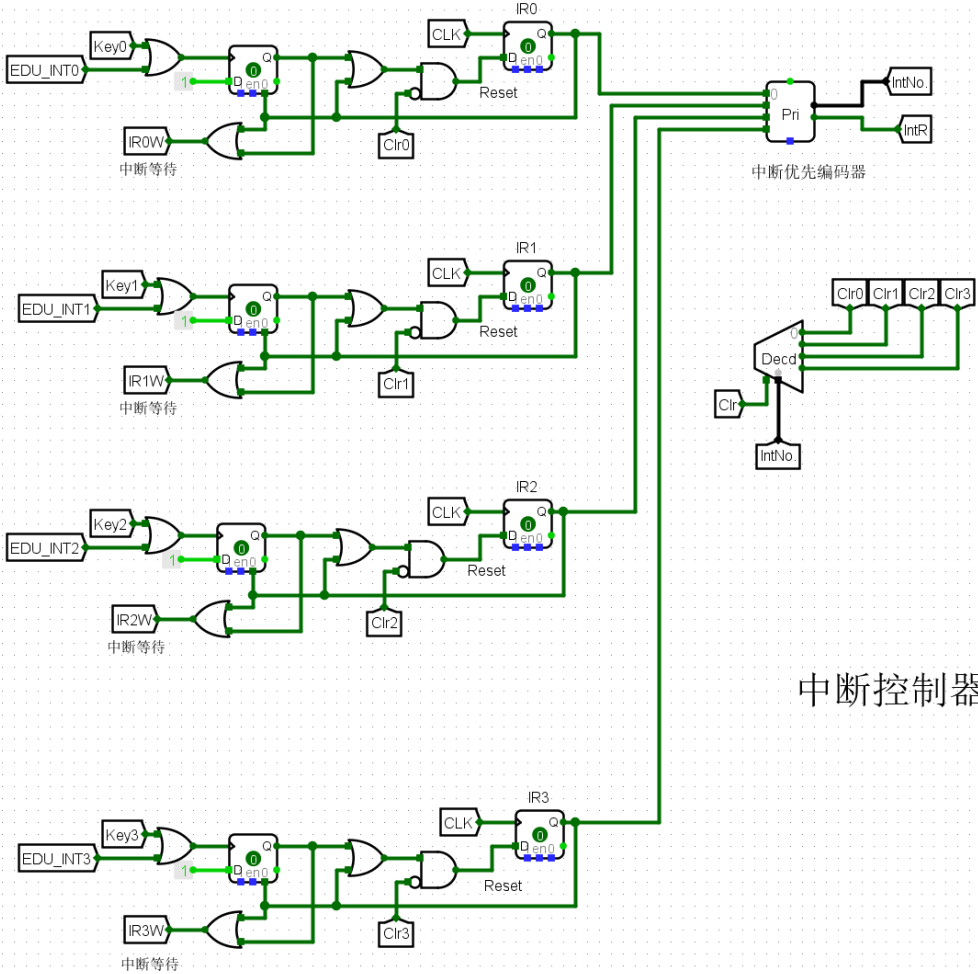
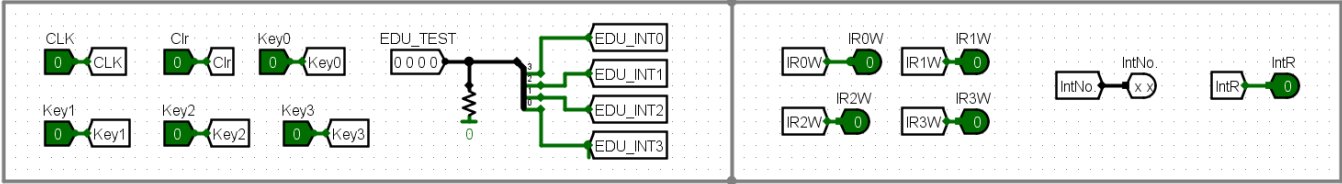


图6.64 支持中断的单总线结构计算机框图

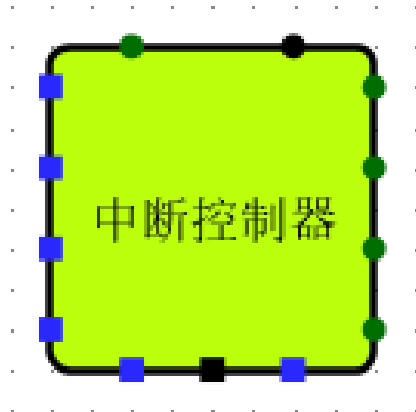


中断逻辑

中断控制器（4个中断源）

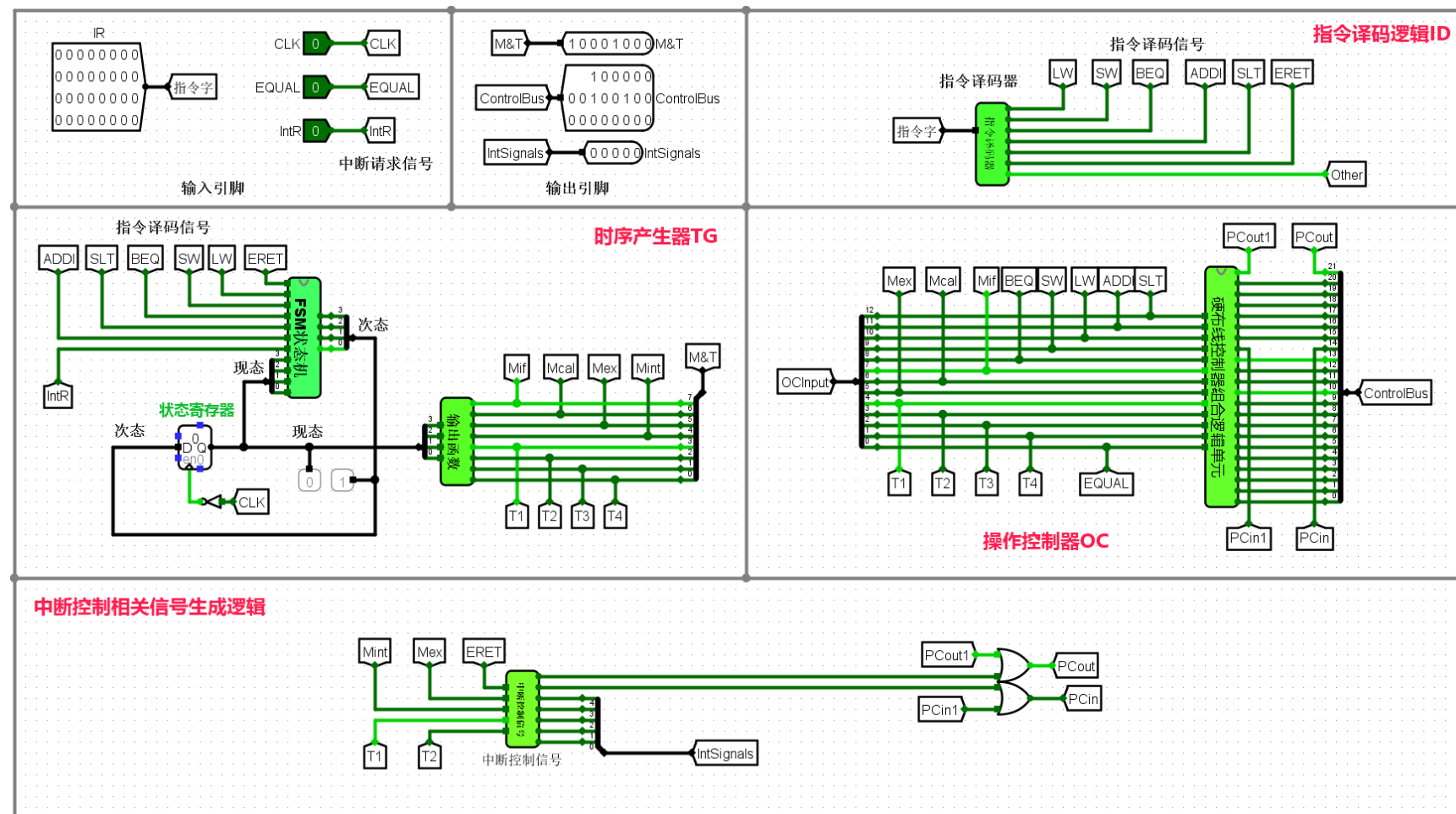
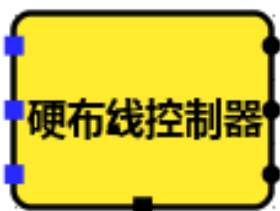


中断控制器

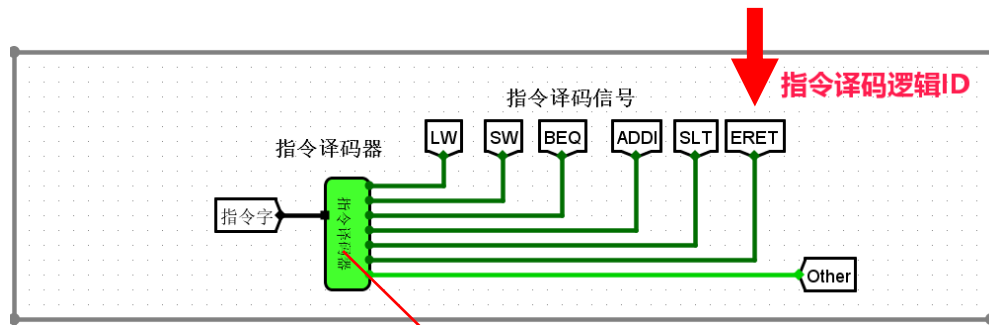


硬布线控制器（三级时序、变长指令周期、支持中断控制）

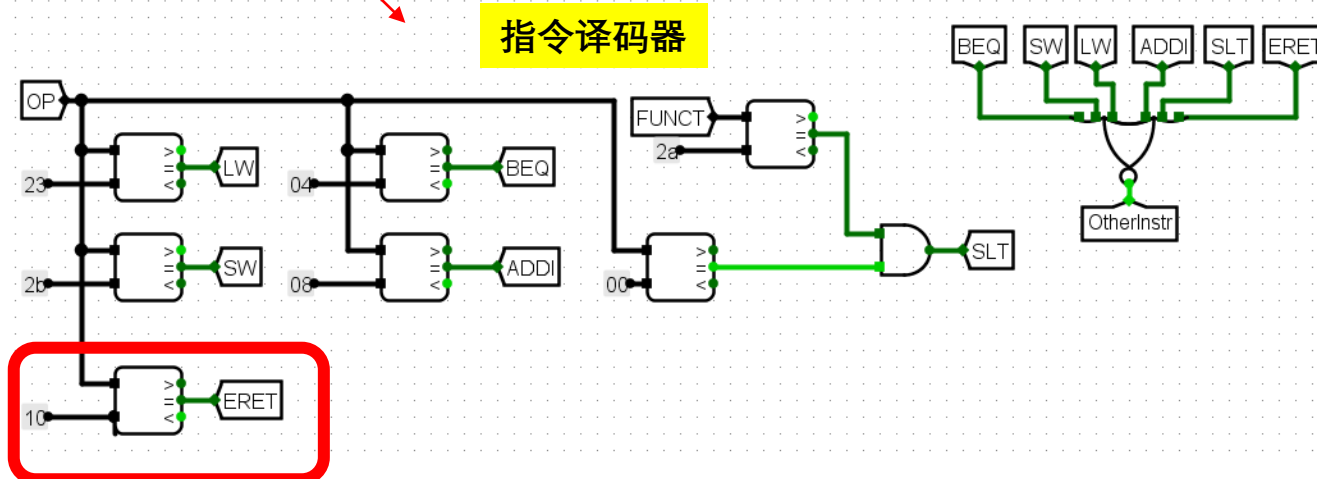
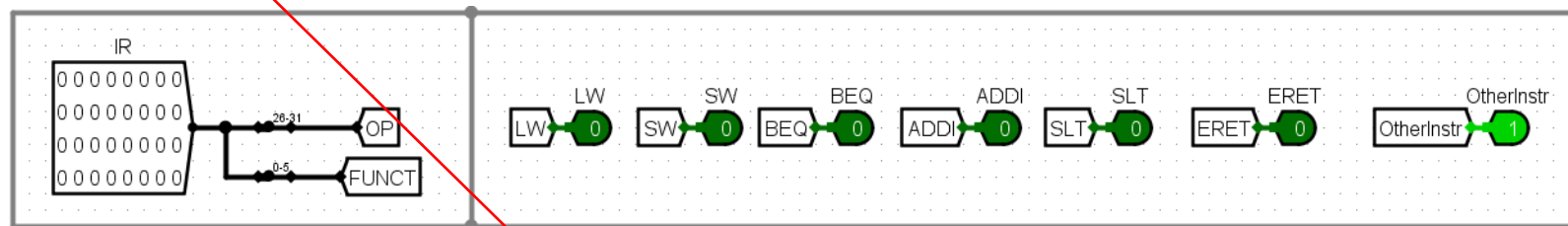
三级时序+中断



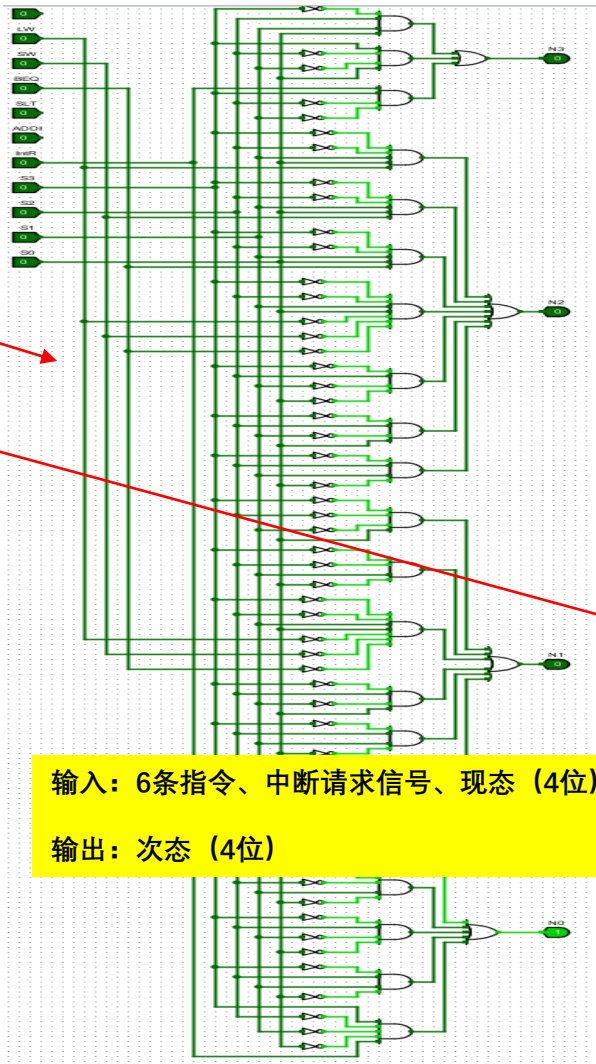
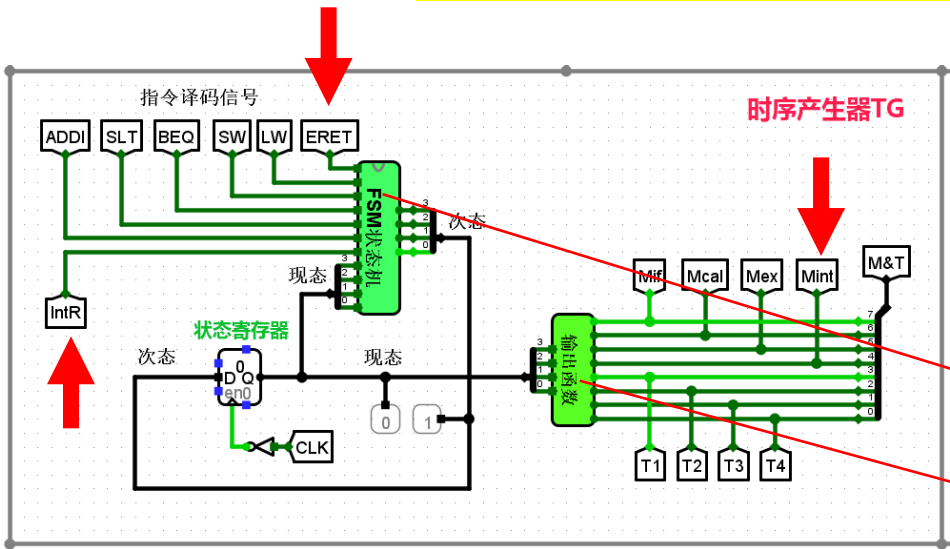
指令译码逻辑（增加ERET指令的译码）



Exception Return																								ERET					
31						26						25						24						6					
COP0						CO						0						0						ERET					
010000						1						000 0000 0000 0000 0000												011000					
6						1						19												6					

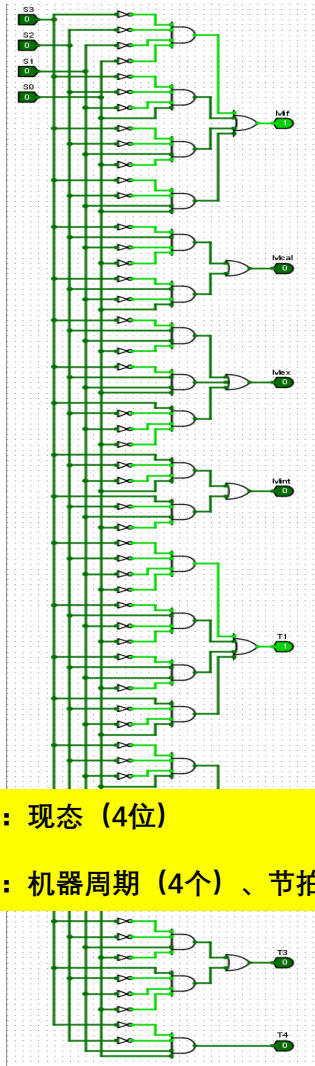


时序产生器（三级时序，变长指令周期，增加ERET指令、中断请求信号IntR和中断周期Mint）



输入：6条指令、中断请求信号、现态（4位）
输出：次态（4位）

FSM状态机



输入：现态（4位）
输出：机器周期（4个）、节拍（4个）

输出函数

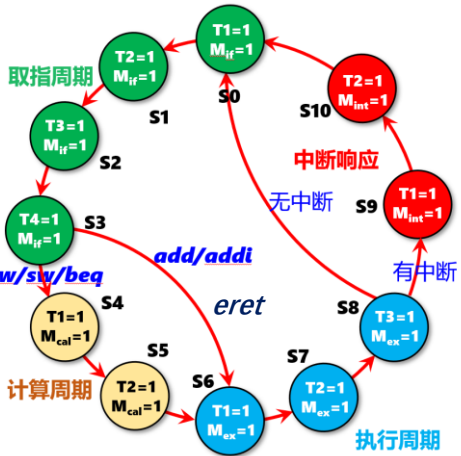
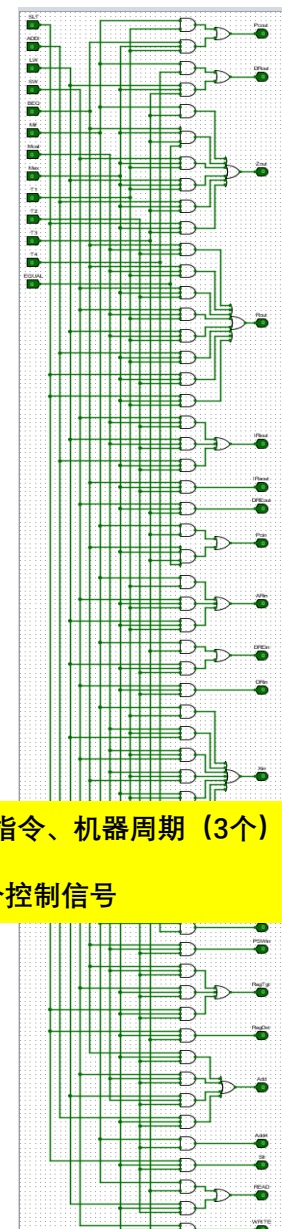
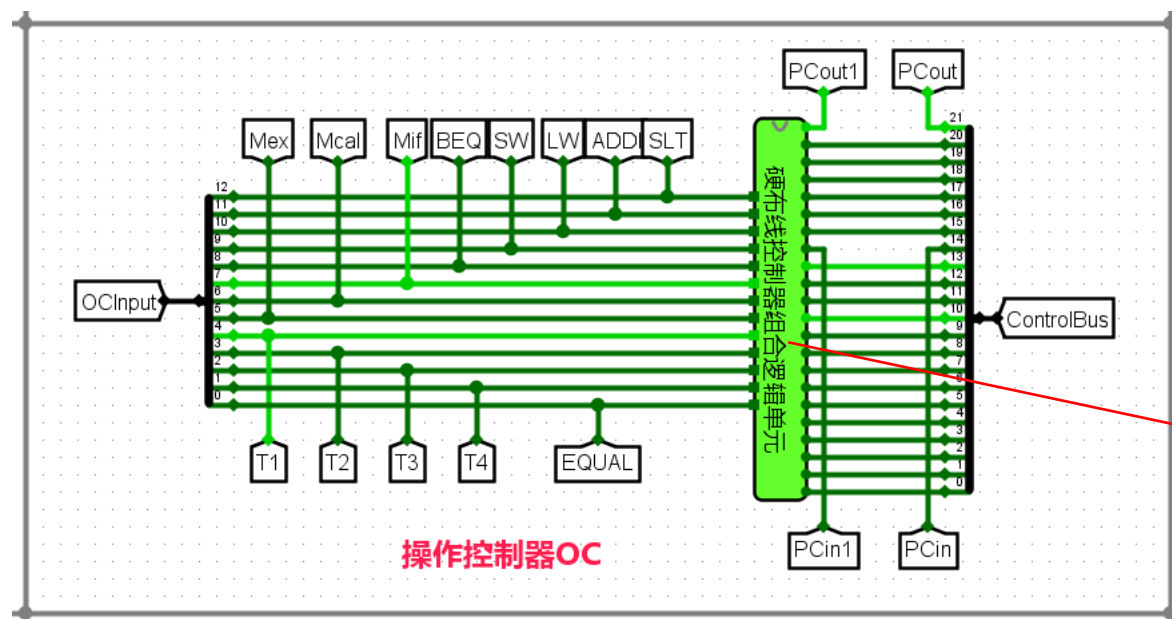


图6.65 支持中断的三级时序状态机（变长指令周期）

操作控制器（与不支持中断的操作控制器相同）

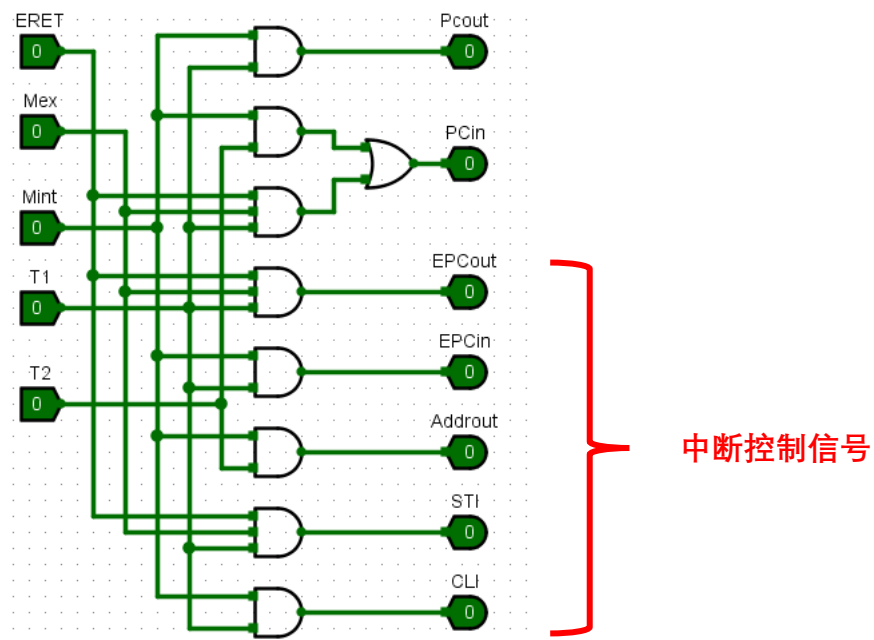
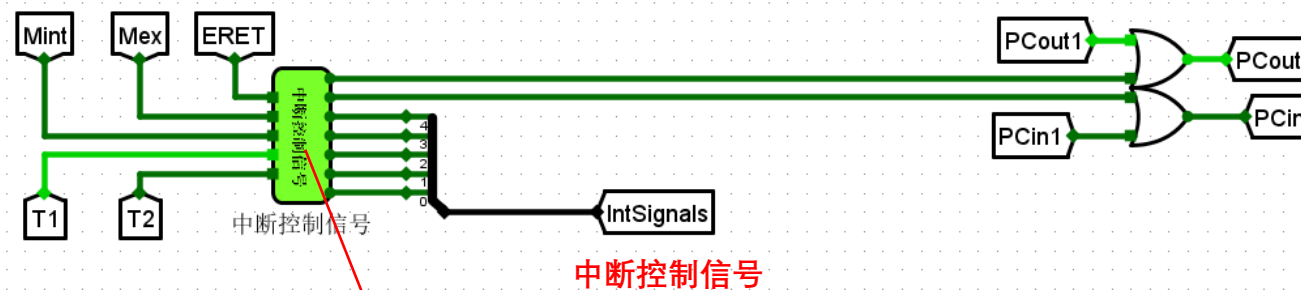


输入：5条指令、机器周期（3个）、节拍（4个）、Equal
输出：22个控制信号

硬布线控制器组合逻辑单元（与不支持中断的相同）

(新增加的) 中断控制信号 (5个) 以及PCout和PCin生成电路

中断控制相关信号生成逻辑



- (3) 编写含有中断服务程序的冒泡法**降序排序程序**（只能使用5+1条指令：lw、sw、beq、slt、addi、**eret**）

sort1_mips_bus_int.asm - 记事本

sort1_mips_bus_int.asm

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

#MIPS程序 sort1_mips_bus.asm 冒泡法排序（降序排列，从大到小）


#主程序起始地址： 0	RAM对应000
#IRQ0中断服务程序的入口地址： 1024	RAM对应100
#IRQ1中断服务程序的入口地址： 1536	RAM对应180
#IRQ2中断服务程序的入口地址： 2048	RAM对应200
#IRQ3中断服务程序的入口地址： 2560	RAM对应280
#10个数（8、1、5、2、7、9、6、4、3、10）存放的起始地址： 3072	RAM对应300
#IRQ0中断服务程序将地址为3328的存储单元的内容加1	RAM对应340
#IRQ0中断服务程序将地址为3392的存储单元的内容加1	RAM对应350
#IRQ0中断服务程序将地址为3456的存储单元的内容加1	RAM对应360
#IRQ0中断服务程序将地址为3520的存储单元的内容加1	RAM对应370
#堆栈的起始地址： 3840	RAM对应3c0

主程序

(只是改变10个数的起始地址, 其他与实验五的程序一样)

.text

main:

addi \$s0,\$zero,8 #第1个数=8 (可以修改) 保存到(3072+0)
sw \$s0,3072(\$zero) 

addi \$s0,\$zero,1 #第2个数=1 (可以修改) 保存到(3072+4)
sw \$s0,3076(\$zero)

addi \$s0,\$zero,5 #第3个数=5 (可以修改) 保存到(3072+8)
sw \$s0,3080(\$zero)

addi \$s0,\$zero,2 #第4个数=2 (可以修改) 保存到(3072+12)
sw \$s0,3084(\$zero)

addi \$s0,\$zero,7 #第5个数=7 (可以修改) 保存到(3072+16)
sw \$s0,3088(\$zero)

addi \$s0,\$zero,9 #第6个数=9 (可以修改) 保存到(3072+20)
sw \$s0,3092(\$zero)

addi \$s0,\$zero,6 #第7个数=6 (可以修改) 保存到(3072+24)
sw \$s0,3096(\$zero)

addi \$s0,\$zero,4 #第8个数=4 (可以修改) 保存到(3072+28)
sw \$s0,3100(\$zero)

addi \$s0,\$zero,3 #第9个数=3 (可以修改) 保存到(3072+32)
sw \$s0,3104(\$zero)

addi \$s0,\$zero,10 #第10个数=10 (可以修改) 保存到(3072+36)
sw \$s0,3108(\$zero)

addi \$s0,\$zero,3072 #\$s0=3072 排序区间开始地址
addi \$s1,\$zero,3108 #\$s1=3108=3072+10*4-4 排序区间结束地址

10个数 如果不是20个数, 这里要修改, 例如20个数, 这里修改为3148

sort_loop:	lw \$s3,0(\$s0)	#\$s3=(\$s0)			
	lw \$s4,0(\$s1)	#\$s4=(\$s1)			
	slt \$t0,\$s3,\$s4	#如果\$s3<\$s4, 则置\$t0=1; 否则, 置\$t0=0	降序排序	从大到小	
	beq \$t0,\$zero,sort_next	#如果\$t0=0, 则转sort_nent			
	sw \$s3,0(\$s1)	#交换(\$s0)和(\$s1)			
	sw \$s4,0(\$s0)	#交换(\$s0)和(\$s1)			
sort_next:	addi \$s1,\$s1,-4	#\$s1-4 -> \$s1			
	beq \$s0,\$s1,loop1	#如果\$s0=\$s1, 则转loop1			
	beq \$zero,\$zero,sort_loop	#转sort_loop			
loop1:	addi \$s0,\$s0,4	#\$s0+4 -> \$s0			
	addi \$s1,\$zero,3108	#\$s1=3108=3072+10*4-4	排序区间结束地址	10个数	如果不是10个数, 这里要修改, 例如20个数, 这里修改为3148
	beq \$s0,\$s1,loop2	#如果\$s0=\$s1, 则转loop2			
	beq \$zero,\$zero,sort_loop	#转sort_loop			
loop2:	beq \$zero,\$zero,loop2	#转loop2	死循环		

IRQ0:

```
addi $sp,$zero,3840|
sw $s0,0($sp)
sw $s1,4($sp)
```

#IRQ0中断服务程序的入口地址: 1024 = 400H

#push registers 需要保留中断程序用到的寄存器

\$s0 \$s1

RAM对应100

```
addi $s1,$zero,3328 ← #RAM对应340
lw $s0,0($s1)
addi $s0,$s0,1
sw $s0,0($s1)
```

```
lw $s1,4($sp)
lw $s0,0($sp)
eret
```

#pop registers

IRQ0的中断服务程序

取出地址为3328 (RAM对应340) 单元的内容, 加1, 再存回去

IRQ1:

```
addi $sp,$zero,3840
sw $s0,0($sp)
sw $s1,4($sp)
```

#IRQ0中断服务程序的入口地址: 1536 = 600H

#push registers 需要保留中断程序用到的寄存器

\$s0 \$s1

RAM对应180

```
addi $s1,$zero,3392 ← #RAM对应350
lw $s0,0($s1)
addi $s0,$s0,1
sw $s0,0($s1)
```

```
lw $s1,4($sp)
lw $s0,0($sp)
eret
```

#pop registers

IRQ1的中断服务程序

取出地址为3392 (RAM对应350) 单元的内容, 加1, 再存回去

IRQ2:

```
addi $sp,$zero,3840
sw $s0,0($sp)
sw $s1,4($sp)
```

#IRQ0中断服务程序的入口地址: 2048 = 800H
#push registers 需要保留中断程序用到的寄存器

RAM对应200
\$s0 \$s1

```
addi $s1,$zero,3456 ← #RAM对应360
lw $s0,0($s1)
addi $s0,$s0,1
sw $s0,0($s1)
```

```
lw $s1,4($sp)      #pop registers
lw $s0,0($sp)
eret
```

IRQ2的中断服务程序

取出地址为3456 (RAM对应360) 单元的内容, 加1, 再存回去

IRQ3:

```
addi $sp,$zero,3840
sw $s0,0($sp)
sw $s1,4($sp)
```

#IRQ0中断服务程序的入口地址: 2560 = A00H
#push registers 需要保留中断程序用到的寄存器

RAM对应280
\$s0 \$s1

```
addi $s1,$zero,3520 ← #RAM对应370
lw $s0,0($s1)
addi $s0,$s0,1
sw $s0,0($s1)
```

```
lw $s1,4($sp)      #pop registers
lw $s0,0($sp)
eret
```

IRQ3的中断服务程序

取出地址为3520 (RAM对应370) 单元的内容, 加1, 再存回去

- (4) 在MIPS汇编器上进行**汇编**，得到机器码，并添加：v2.0 raw, 220*0, 118*0, 118*0, 118*0；添加若干个0的目的是使4个中断服务程序的起始地址分别为：

打开“命令提示符”，执行：
cd \mips
java -jar Mars4_5.jar

sort1_mips_bus_int.hex - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

v2.0 raw
20100008
ac100c00
20100001
ac100c04
20100005
ac100c08
20100002
ac100c0c
20100007
ac100c10
20100009
ac100c14
20100006
ac100c18
20100004
ac100c1c
20100003
ac100c20
2010000a
ac100c24
20100c00
20110c24
8e130000
8e340000
0274402a
11000002
ae330000
ae140000
2231fffc
12110001
1000fff7
22100004
20110c24
12110001
1000fff3
1000ffff
220*0
201d0f00
afb00000
afb10004
20110d00
8e300000
22100001
ae300000
8fb10004
8fb00000
42000018

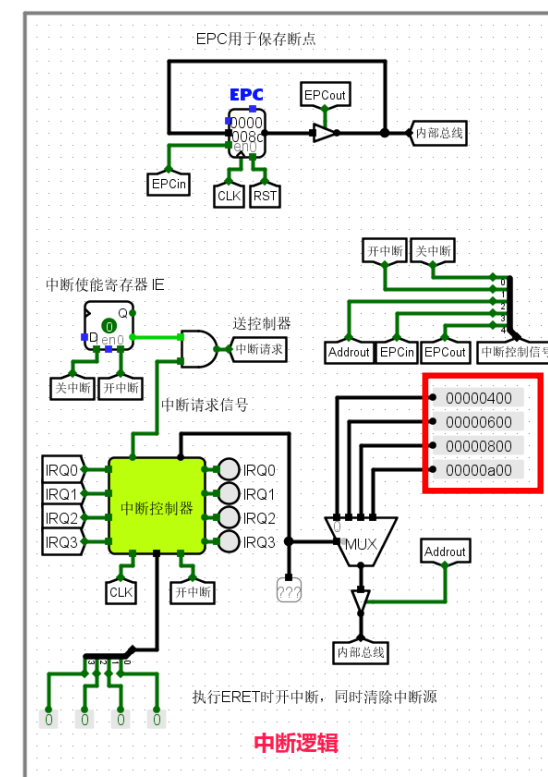
sort1_mips_bus_int.hex

sort1_mips_bus_int.hex - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

1000fff3
1000ffff
220*0
201d0f00
afb00000
afb10004
20110d00
8e300000
22100001
ae300000
8fb10004
8fb00000
42000018
118*0
201d0f00
afb00000
afb10004
20110d40
8e300000
22100001
ae300000
8fb10004
8fb00000
42000018
118*0
201d0f00
afb00000
afb10004
20110d80
8e300000
22100001
ae300000
8fb10004
8fb00000
42000018
118*0
201d0f00
afb00000
afb10004
20110dc0
8e300000
22100001
ae300000
8fb10004
8fb00000
42000018

#IRQ0中断服务程序的入口地址: 1024 = 400H
#IRQ1中断服务程序的入口地址: 1536 = 600H
#IRQ2中断服务程序的入口地址: 2048 = 800H
#IRQ3中断服务程序的入口地址: 2560 = A00H

RAM对应100
RAM对应180
RAM对应200
RAM对应280



```

000 20100008 ac100c00 20100001 ac100c04 20100005 ac100c08 20100002 ac100c0c 20100007 ac100c10 20100009 ac100c14 20100006 ac100c18 20100004 ac100c1c
010 20100003 ac100c20 2010000a ac100c24 20100000 ac100c28 2010000f ac100c34 2010000e ac100c38 2010000d ac100c3c 2010000c ac100c40 2010000b ac100c44
020 20110c24 12110001 1000ffff 1000ffff 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

```

主程序

```

030 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
050 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
070 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
080 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
090 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0a0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0b0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0c0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0d0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0e0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0f0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

```

```

100 201d0f00 afb00000 afb10004 20110d00 8e300000 22100001 ae300000 20110d00 8e300000 22100001 ae300000 20110d00 8e300000 22100001 ae300000
110 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
120 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
130 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
140 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
150 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
160 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
170 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

```

IRQ0中断服务程序

```

180 201d0f00 afb00000 afb10004 20110d40 8e300000 22100001 ae300000 20110d40 8e300000 22100001 ae300000 20110d40 8e300000 22100001 ae300000
190 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
1a0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
1b0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
1c0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
1d0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
1e0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
1f0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

```

IRQ1中断服务程序

```

200 201d0f00 afb00000 afb10004 20110d80 8e300000 22100001 ae300000 20110d80 8e300000 22100001 ae300000 20110d80 8e300000 22100001 ae300000
210 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
220 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
230 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
240 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
250 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
260 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
270 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

```

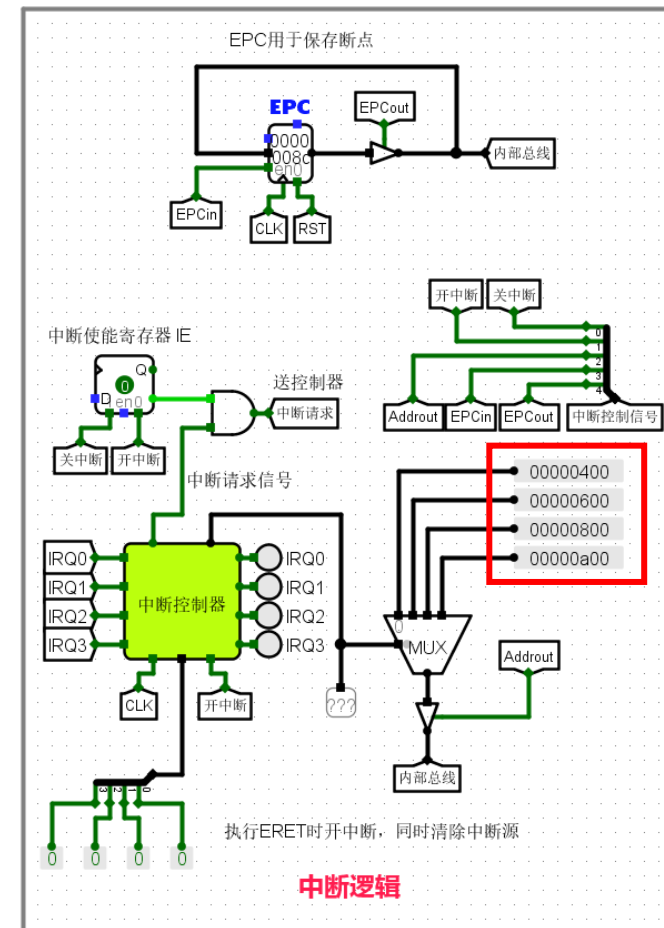
IRQ2中断服务程序

```

280 201d0f00 afb00000 afb10004 20110dc0 8e300000 22100001 ae300000 20110dc0 8e300000 22100001 ae300000 20110dc0 8e300000 22100001 ae300000

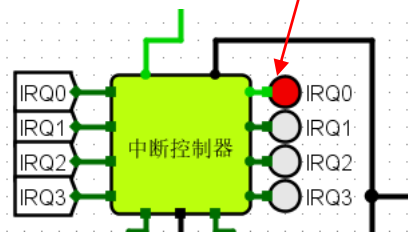
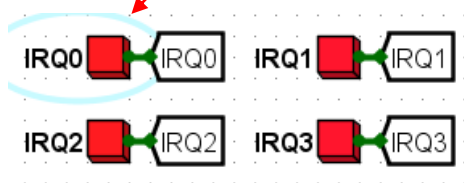
```

IRQ3中断服务程序

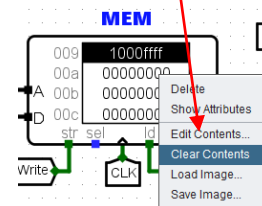


中断逻辑

- (5) 在支持中断控制的单总线结构 MIPS 处理器数据通路上**运行**含有中断服务程序的冒泡法降序排序**程序**
 - 第一步：在Logisim中打开电路“**单总线结构MIPS处理器（三级时序+变长指令周期+硬布线控制器+中断+5条指令）.circ**”
 - 第二步：按Ctrl+R，复位电路
 - 第三步：将“**sort1_mips_bus_int.hex**”载入存储器RAM；这一步可能要进行几次，确保机器码装入存储器RAM，也可以通过清除RAM中的内容后，再进行载入
 - 第四步：设置时钟频率=1KHz，按Ctrl+K 启动时钟，运行程序
 - 第五步：按“IRQ0”，发出中断0请求信号，此时IRQ0灯变红，表示执行中断服务程序0；过一会儿灯灭，表示中断服务程序0执行完毕
 - 第六步：同理可以按“IRQ1”、“IRQ2”、“IRQ3”按钮，分别发出中断1、中断2、中断3的请求信号
 - 第七步：等排序程序执行完毕，按Ctrl+K停止时钟，并查看存储器RAM中的内容



✓ Simulation Enabled	Ctrl+E
Reset Simulation	Ctrl+R
Step Simulation	Ctrl+I
Go Out To State	▶
Go In To State	▶
Tick Once	Ctrl+T
✓ Ticks Enabled	Ctrl+K
Tick Frequency	▶
Logging...	



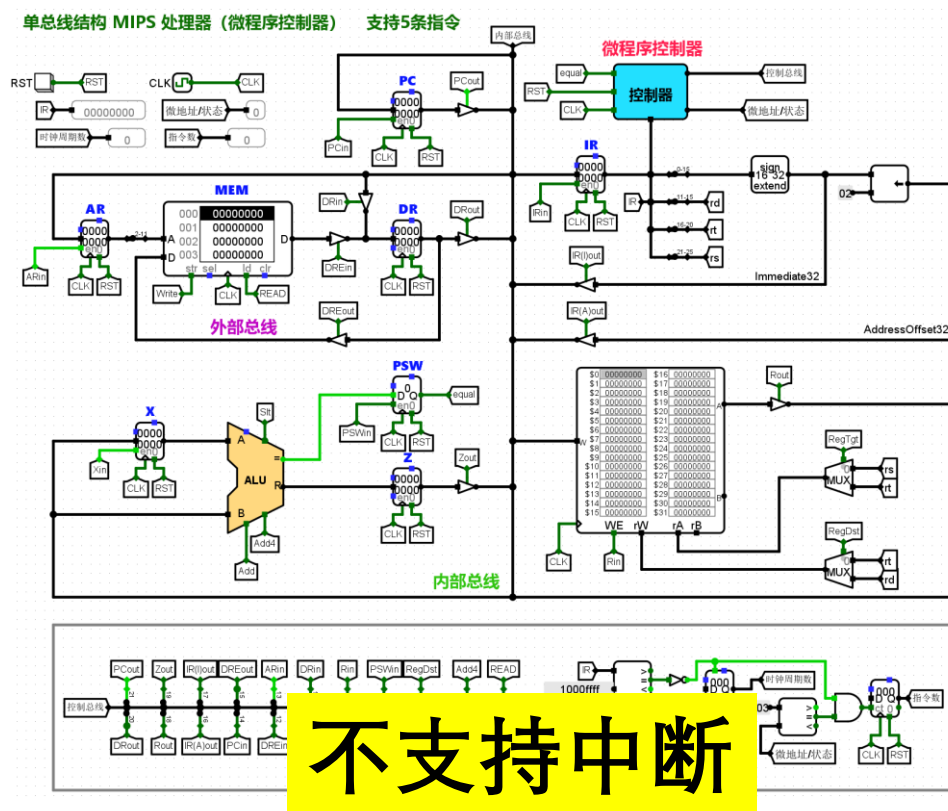
[illegible]

- (6) 其它要求

- ①运行“**升序排序程序**”（需要修改sort1_mips_bus_int.asm程序，并对汇编后的.hex文件进行修改），在程序的运行过程中，按IRQ0、IRQ1、IRQ2、IRQ3键，发出中断请求信号，并观察运行结果。
- ②请分析“支持中断控制的单总线结构 MIPS 处理器（硬布线控制器+中断）”的**电路原理**：
 - 包括：数据通路电路、中断逻辑电路、硬布线控制器电路、指令译码器电路、状态机电路、输出函数电路、中断控制信号产生电路、中断控制器电路等。

2、支持中断的单总线结构 MIPS 处理器 (微程序控制器)

- (1) (实验五) 单总线结构 MIPS 处理器 (微程序控制器、5条指令) 数据通路

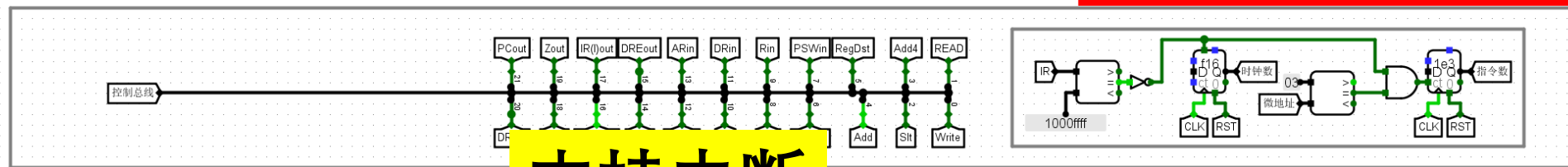
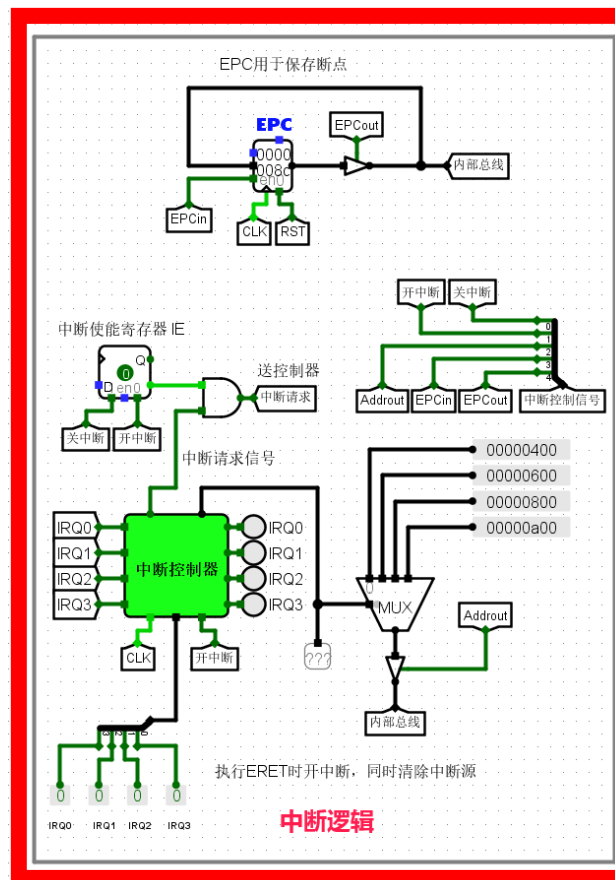
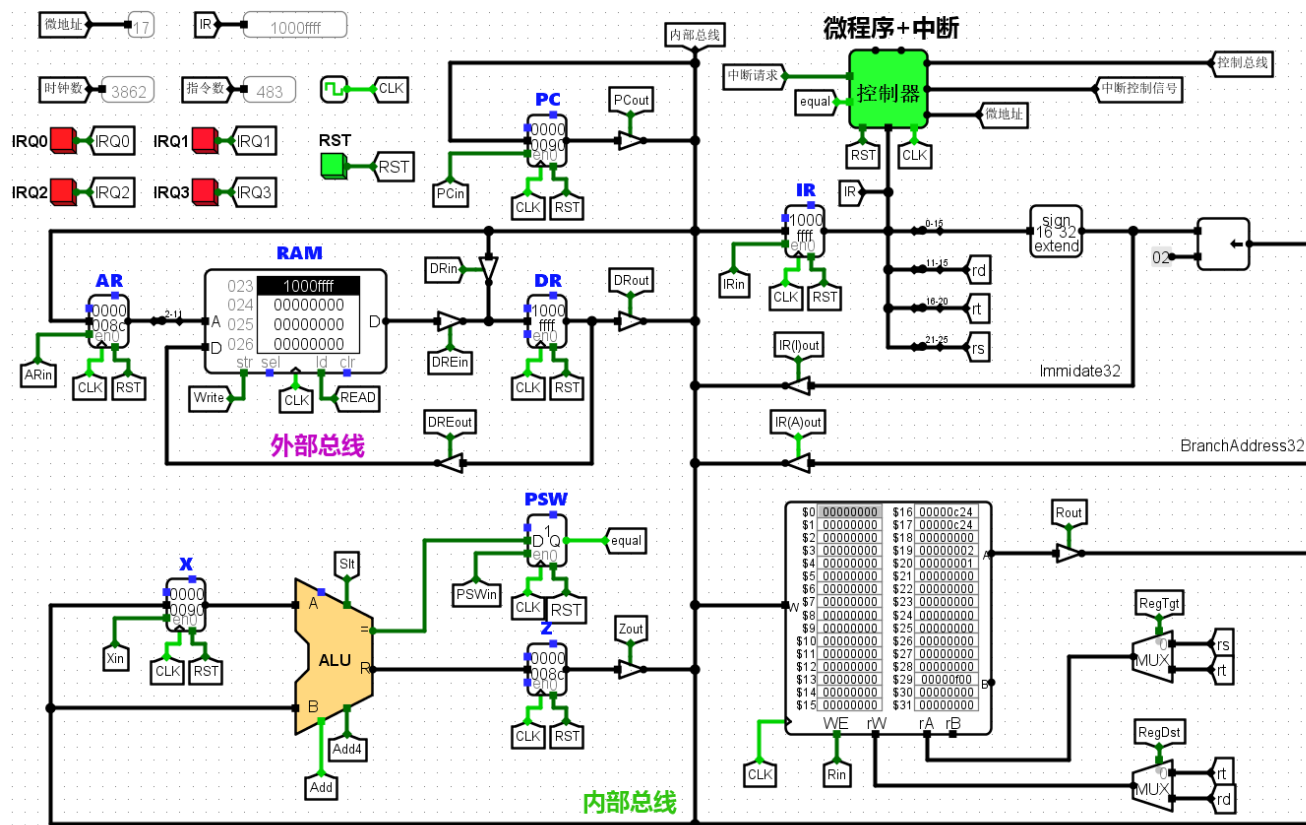


5条指令

序号	MIPS指令
1	<u>lw</u> rt,imm(rs)
2	<u>sw</u> rt,imm(rs)
3	<u>beq</u> rs,rt,imm
4	<u>addi</u> rt,rs,imm
5	<u>slt</u> rd,rs,rt

- (2) 支持中断控制的单总线结构 MIPS 处理器（微程序控制器、5条指令）数据通路

单总线结构 MIPS 处理器数据通路（微程序+中断）



支持中断

中断控制逻辑电路（与硬布线控制器相同）

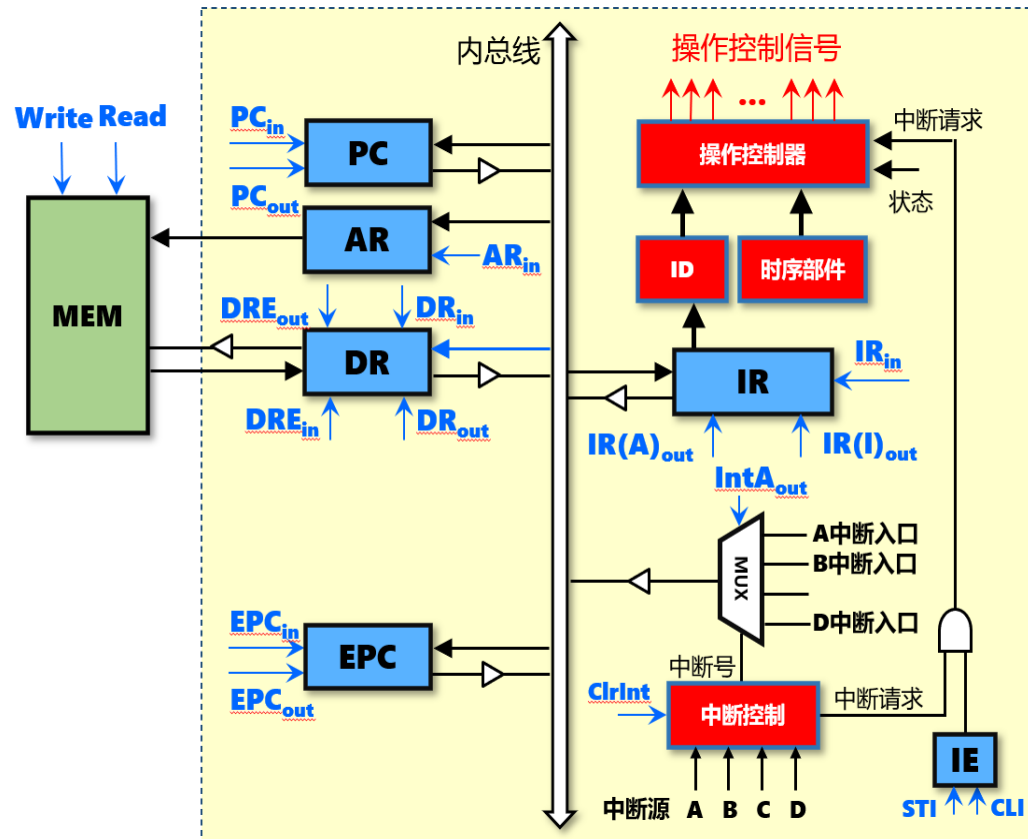
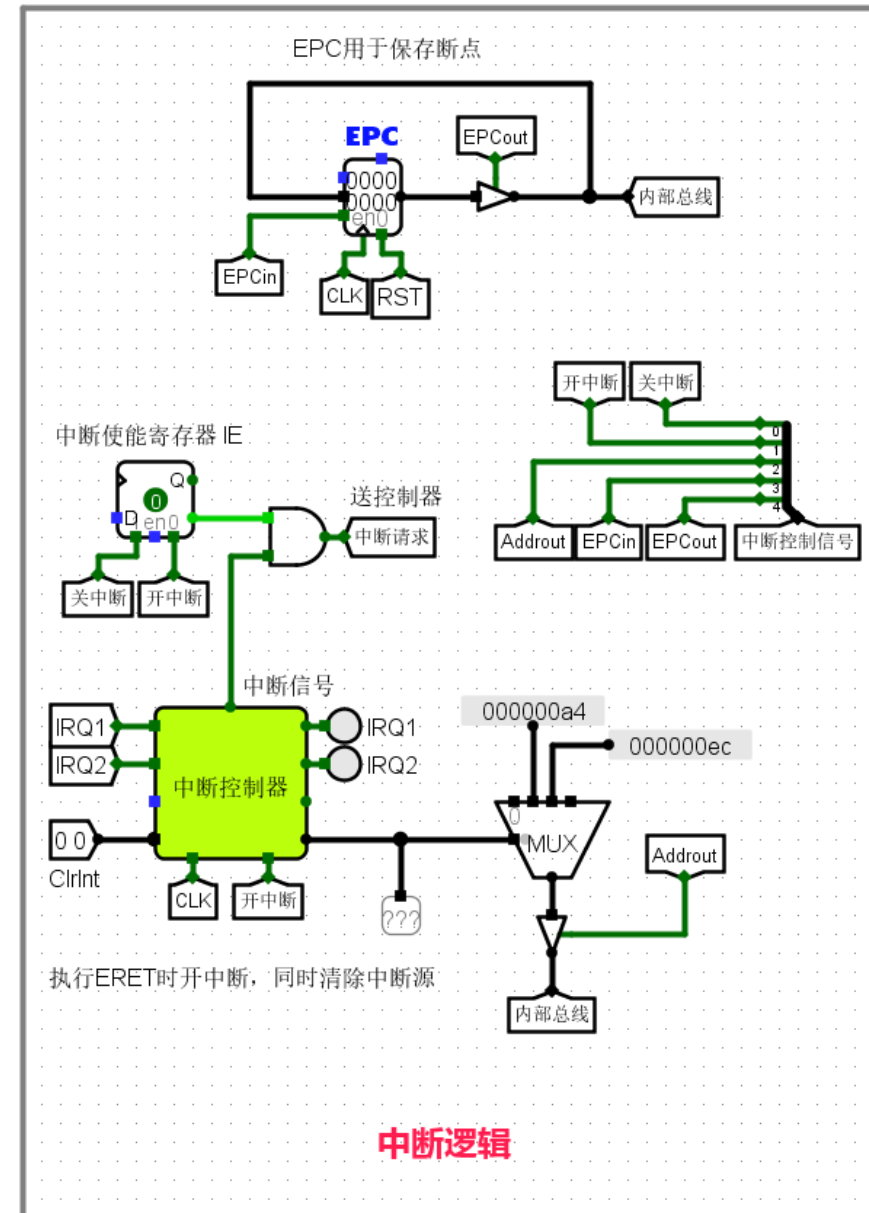
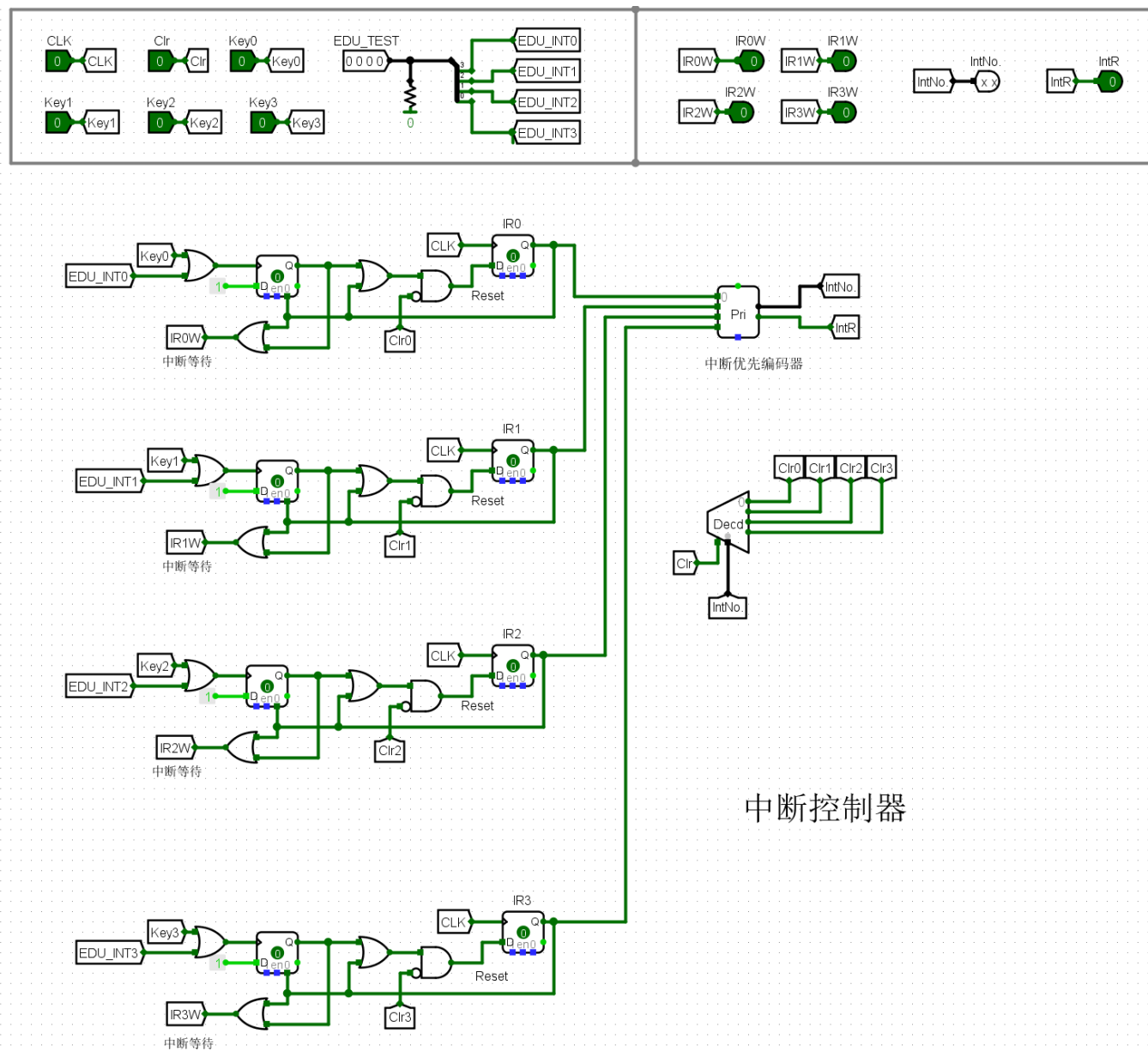
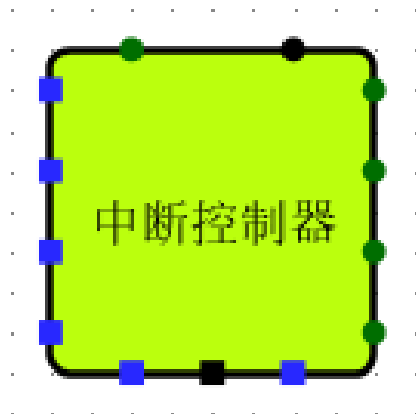


图6.64 支持中断的单总线结构计算机框图



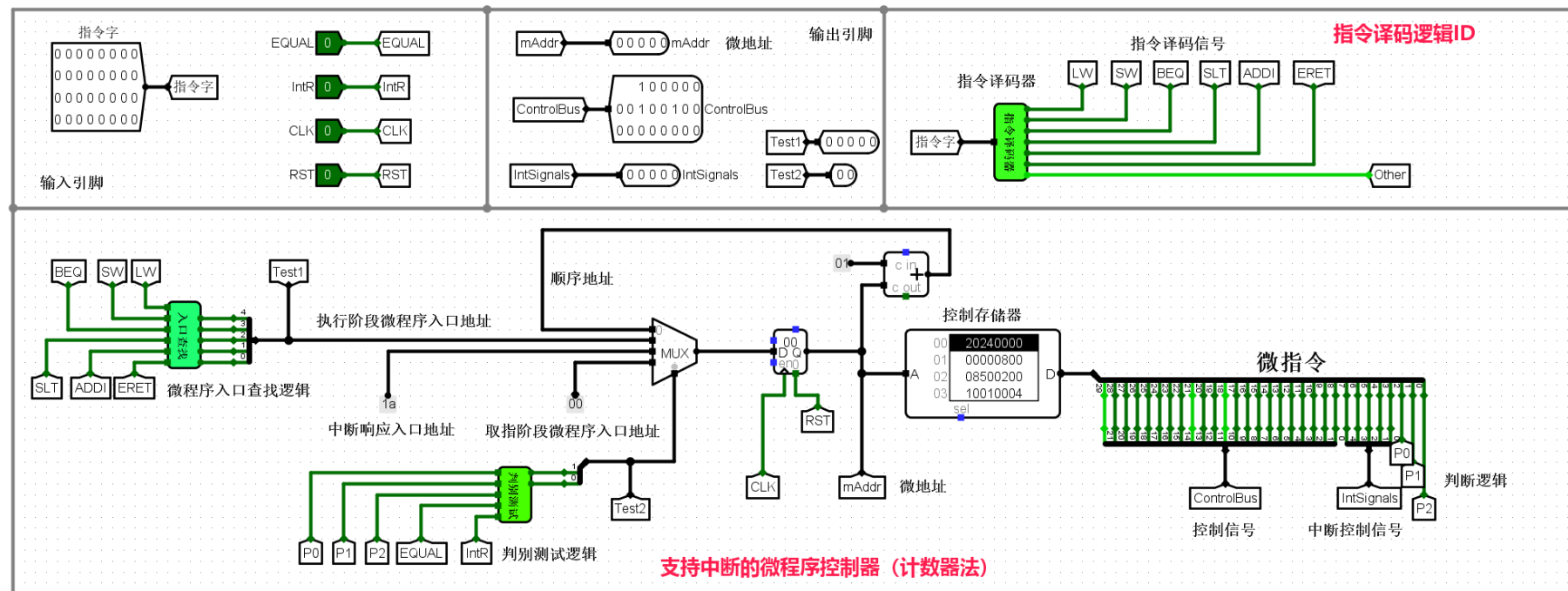
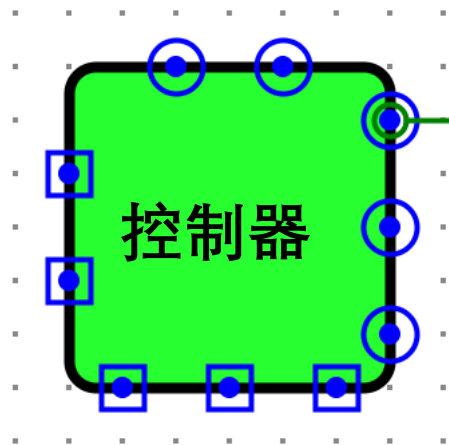
中断逻辑

中断控制器（4个中断源）（与硬布线控制器相同）

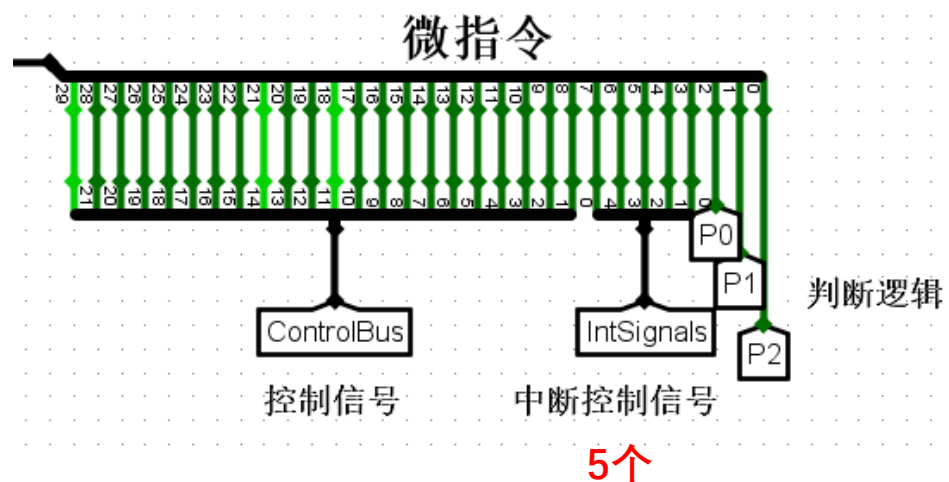


微程序控制器（支持中断）

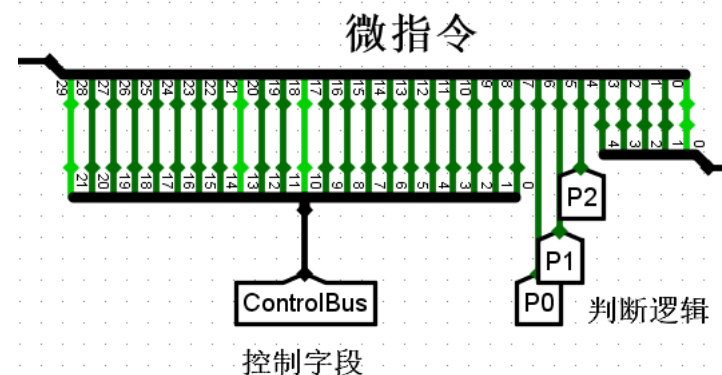
微程序+中断



(支持中断) 微指令格式 (计数器法)

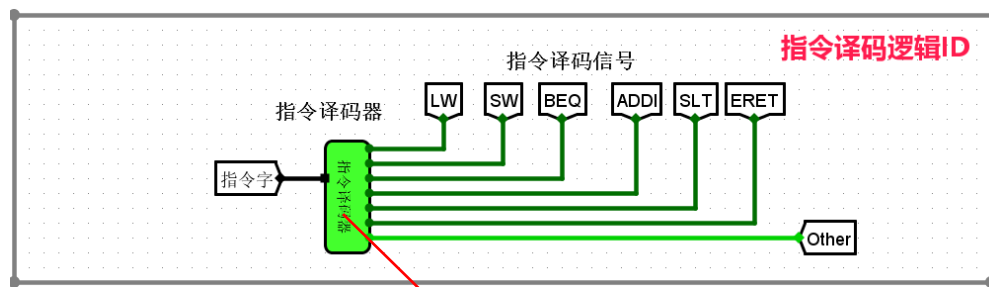


(不支持中断) 微指令格式 (下址字段法)

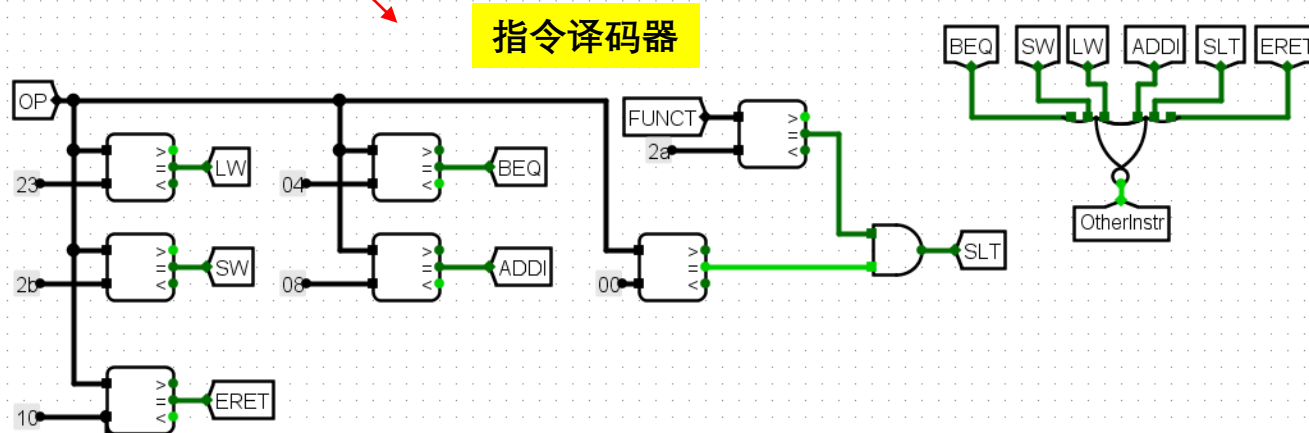
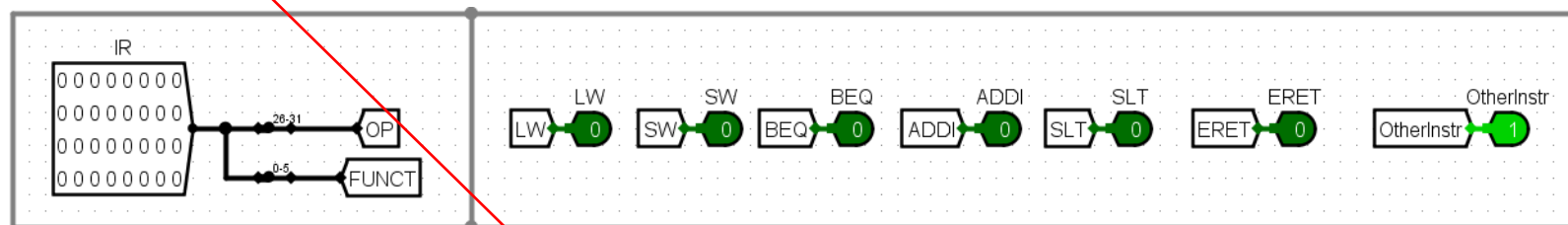


因为微指令的长度不能超过32位，增加5个中断控制信号后，如果采用下址字段法，微指令长度=22（单总线的控制信号）+5（中断控制信号）+3（判别测试）+5（下址字段）=35位；因此需要改为计数器法，微指令长度=22+5+3=30位

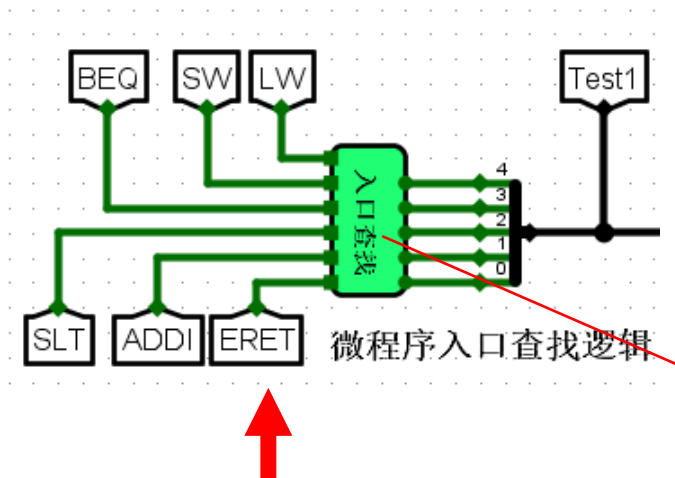
指令译码逻辑（与硬布线控制器相同）



Exception Return										ERET
31	26	25	24				6	5		0
COP0		CO	0					ERET		
010000		1	000 0000 0000 0000 0000					011000		
6		1	19					6		



微程序入口查找逻辑（增加ERET指令）



S4 = SLT+ADDI+ERET

$$S3 = SW + BEQ + ERET$$

S2 = LW+BEQ+ADDI

S1 = BEQ+SLT+ADDI

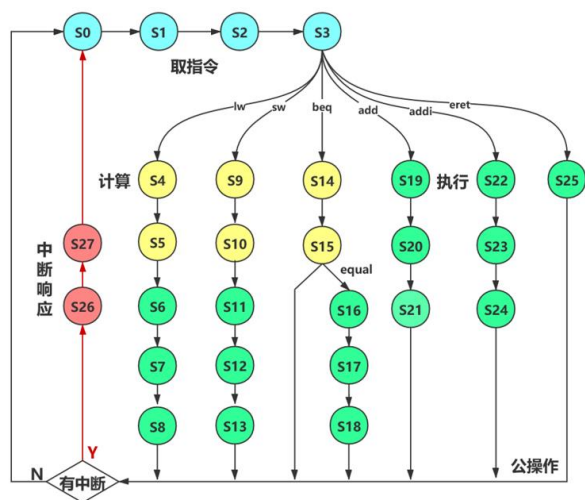
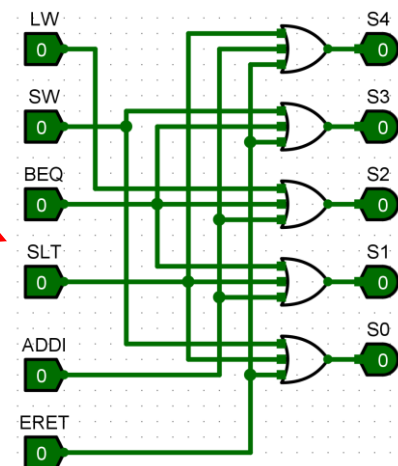
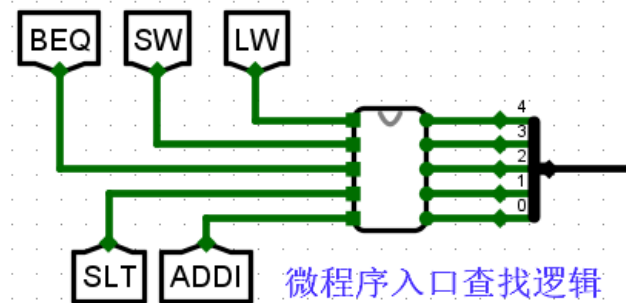
$$S0 = SW + SLT + ERET$$


图6.66 支持中断的现代时序状态机

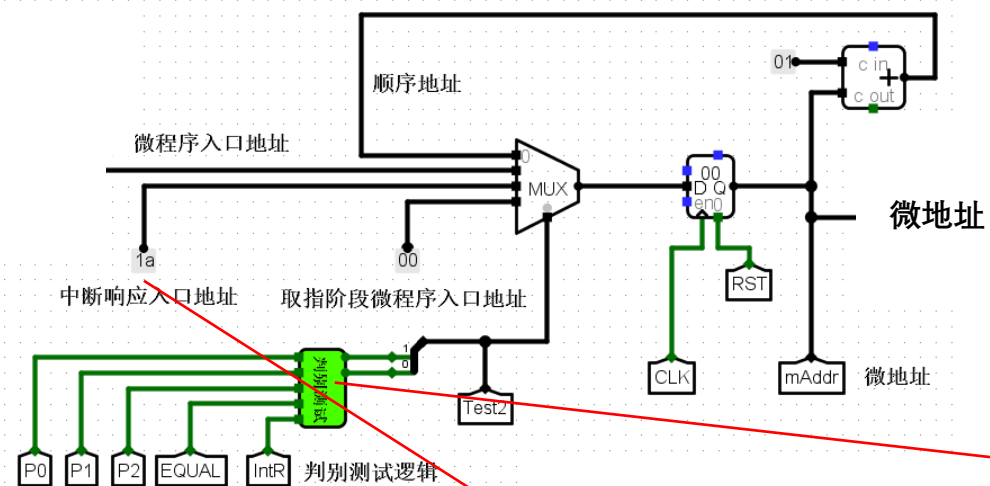
[illegible]

微程序入口查找逻辑自动生成——单总线结构MIPS处理器（微程序+中断）.xlsx

(不支持中断) 微程序入口查找逻辑



判别测试逻辑（增加P2、IntR、中断响应入口、取指阶段微程序入口）


$$M1 = \sim P0 \& \sim P1 \& P2 \& \text{IntR} + \sim P0 \& \sim P1 \& P2 \& \sim \text{IntR} + \sim P0 \& P1 \& P2 \& \sim \text{EQUAL} \& \text{IntR} + \sim P0 \& P1 \& P2 \& \sim \text{EQUAL} \& \sim \text{IntR}$$
$$M0 = P0 + \sim P0 \& \sim P1 \& P2 \& \sim \text{IntR} + \sim P0 \& P1 \& P2 \& \sim \text{EQUAL} \& \sim \text{IntR}$$

1a=26=S26

输入 (填1或0, 不填为无关项x)							
P0	P1	P2	EQUAL	IntR		M1	M0
1						0	1
0	0	0				0	0
0	0	1		1		1	0
0	0	1		0		1	1
0	1	0				0	0
0	1	1	1			0	0
0	1	1	0	1		1	0
0	1	1	0	0		1	1

判别测试逻辑自动生成——单总线结构MIPS处理器（微程序+中断）.xlsx

(不支持中断) 判别测试逻辑

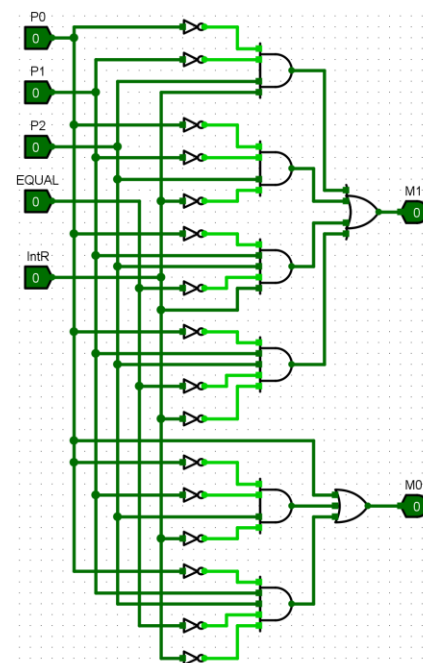
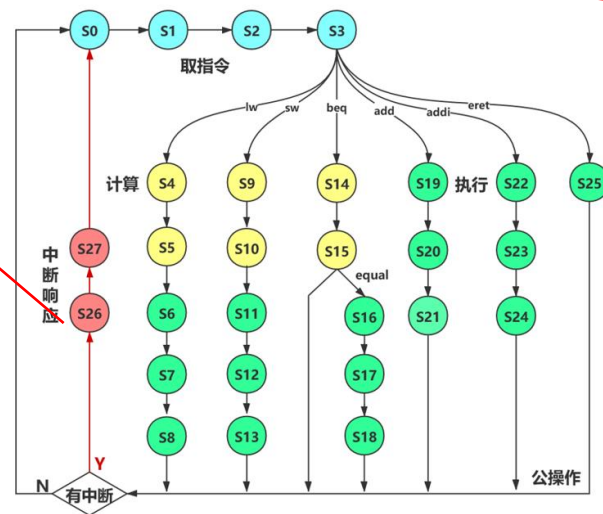
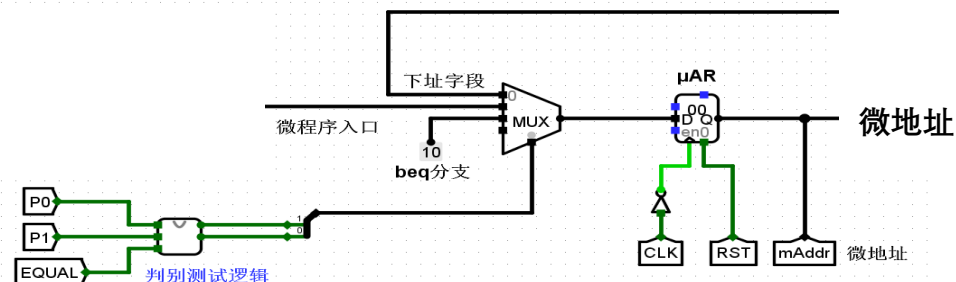


图6.66 支持中断的现代时序状态机

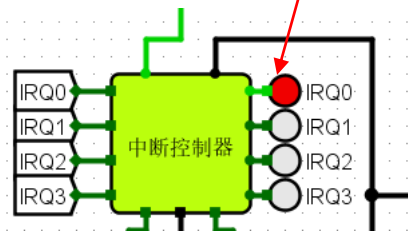
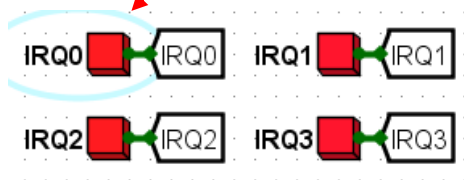
微程序 (采用计数器法, 并且增加3条: ERET指令执行周期1条、INT中断周期2条)

微指令功能	状态/微地址	PCout	D Rout	Zout	Rout	IR(Out)	IR(A)out	DREout	PCin	ARin	DREin	DRin	Xin	Rin	IRin	PSWin	Rs/Rt	RegDst	Add	Add4	Slt	READ	WRITE	EPCout	EPCin	Addrout	STI	CLI	P0	P1	P2	微指令	微指令十六进制
取指令	0	1							1			1																				10000000100100000000000000000000	20240000
取指令	1																			1												00000000000000000000000000000000	800
取指令	2			1					1		1											1										00100001010000000000000000000000	8500200
取指令	3		1												1														1			01000000000001000000000000000100	10010004
LW	4				1								1																			00010000000100000000000000000000	4040000
LW	5					1													1													00001000000000000100000000000000	2001000
LW	6			1						1																						00100000100000000000000000000000	8200000
LW	7										1											1										00000000010000000000000000000000	100200
LW	8		1											1																	1	01000000000100000000000000000001	10020001
SW	9				1								1																			00010000000100000000000000000000	4040000
SW	10					1													1													00001000000000000000000000000000	2001000
SW	11			1						1																						00100000100000000000000000000000	8200000
SW	12				1							1					1															00010000001000010000000000000000	4084000
SW	13						1															1								1		00000010000000000000000000000001	800101
BEQ	14				1								1																			00010000000100000000000000000000	4040000
BEQ	15				1												1	1												1	1	00010000000000011000000000000011	400C003
BEQ	16	1											1																			10000000000100000000000000000000	20040000
BEQ	17						1													1												00000100000000000001000000000000	1001000
BEQ	18			1						1																					1	00100001000000000000000000000001	8400001
SLT	19				1								1																			00010000000100000000000000000000	4040000
SLT	20				1												1				1											00010000000000010001000000000000	4004400
SLT	21			1										1					1											1		00100000000100010000000000000001	8022001
ADDI	22				1								1																			00010000000100000000000000000000	4040000
ADDI	23					1													1													00001000000000000000000000000000	2001000
ADDI	24			1										1																1		00100000000001000000000000000001	8020001
ERET	25								1															1			1				1	00000001000000000000000000000001	400091
INT	26	1																							1			1				10000000000000000000000000000000	20000048
INT	27								1																		1				1	00000001000000000000000000000001	400021

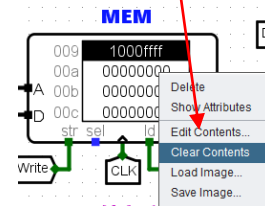
微程序自动生成——单总线结构MIPS处理器 (微程序控制器+中断+5条指令) .xlsx

• (3) 在支持中断控制的单总线结构 MIPS 处理器数据通路上运行含有中断服务程序的冒泡法降序排序程序

- 第一步：在Logisim中打开电路“单总线结构MIPS处理器（微程序控制器+中断+5条指令）.circ”
- 第二步：按Ctrl+R，复位电路
- 第三步：将“sort1_mips_bus_int.hex”载入存储器RAM；这一步可能要进行几次，确保机器码装入存储器RAM，也可以通过清除RAM中的内容后，再进行载入
- 第四步：设置时钟频率=1KHz，按Ctrl+K 启动时钟，运行程序
- 第五步：按“IRQ0”，发出中断0请求信号，此时IRQ0灯变红，表示执行中断服务程序0；过一会儿灯灭，表示中断服务程序0执行完毕
- 第六步：同理可以按“IRQ1”、“IRQ2”、“IRQ3”按钮，分别发出中断1、中断2、中断3的请求信号
- 第七步：等排序程序执行完毕，按Ctrl+K停止时钟，并查看存储器RAM中的内容



✓ Simulation Enabled	Ctrl+E
Reset Simulation	Ctrl+R
Step Simulation	Ctrl+I
Go Out To State	▶
Go In To State	▶
Tick Once	Ctrl+T
✓ Ticks Enabled	Ctrl+K
Tick Frequency	▶
Logging...	



[illegible]

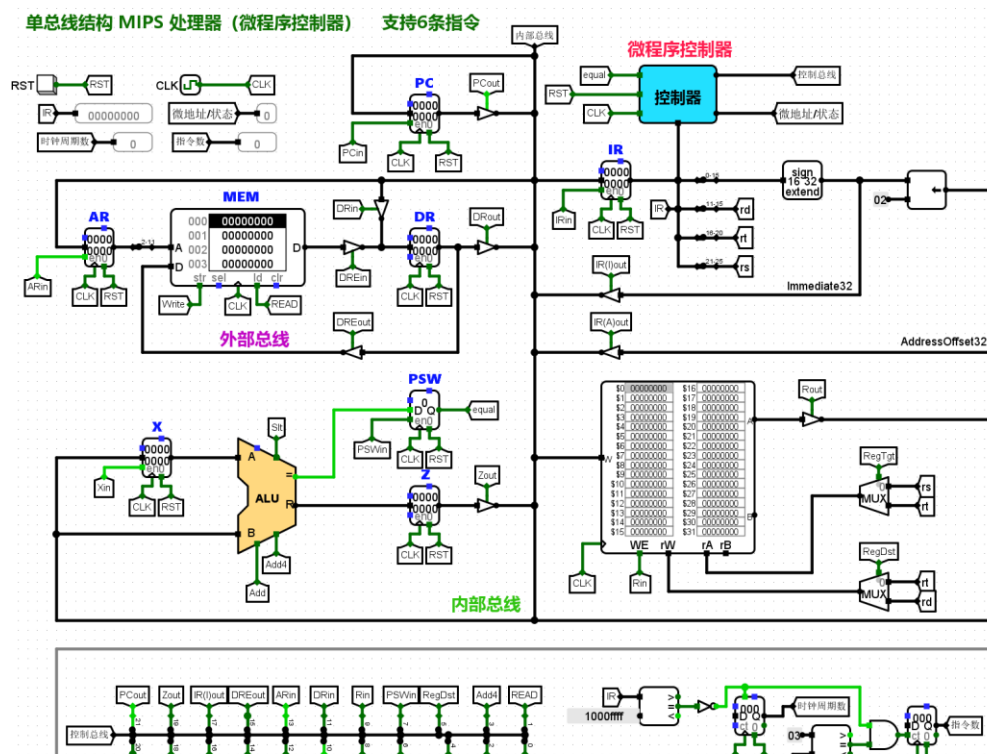
- (4) 其它要求

- ①运行“**升序排序程序**”，在程序的运行过程中，按IRQ0、IRQ1、IRQ2、IRQ3键，发出中断请求信号，并观察实验结果。
- ②请分析“支持中断控制的单总线结构 MIPS 处理器（微程序控制器+中断）”的**电路原理**：
 - 包括：数据通路电路、中断逻辑电路、微程序控制器电路、指令译码器电路、微程序入口查找逻辑电路、判别测试逻辑电路、控制存储器中的微程序等。

二、设计实验

支持中断的单总线结构 MIPS 处理器 (微程序控制器, 6条指令)

- 1、请在实验五完成的“单总线结构 MIPS 处理器（微程序控制器，6条指令，增加了add指令）数据通路”上增加支持中断控制的功能，使其能支持4个中断源



6条指令

序号	指令	汇编代码
1	<u>lw</u>	<u>lw rt,imm(rs)</u>
2	<u>sw</u>	<u>sw rt,imm(rs)</u>
3	<u>beq</u>	<u>beq rs,rt,imm</u>
4	<u>slt</u>	<u>slt rd,rs,rt</u>
5	<u>addi</u>	<u>addi rt,rs,imm</u>
6	<u>add</u>	<u>add rd,rs,rt</u>

不支持中断的单总线MIPS处理器数据通路（微程序控制器+6条指令）

- 2、设计思路:

- (1) **中断逻辑**部分与支持中断的单总线结构MIPS处理器（硬布线控制器/微程序控制器，5条指令）相同；请通过拷贝的方式，将这部分电路拷贝过去；包括**中断控制器**子电路，也要拷贝过去

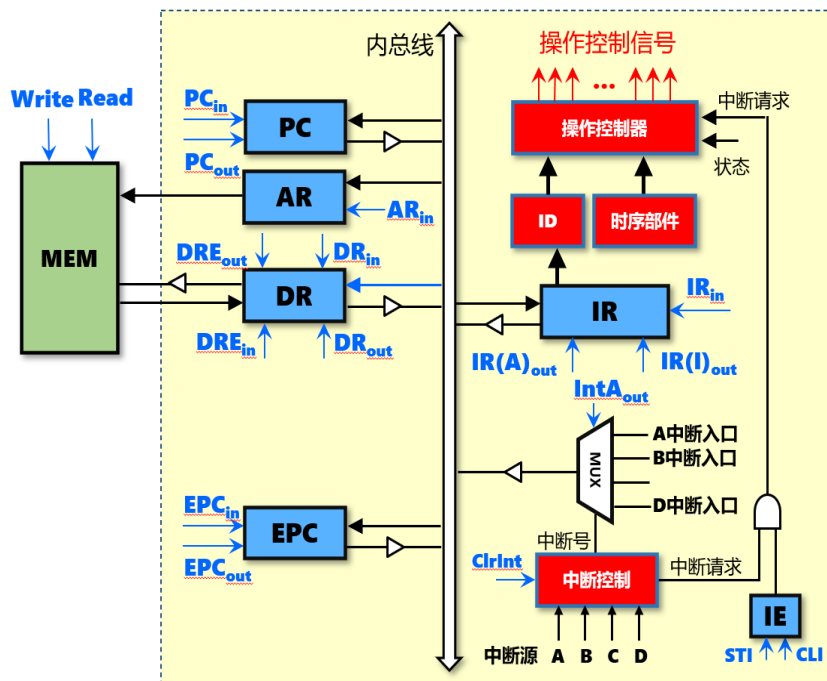
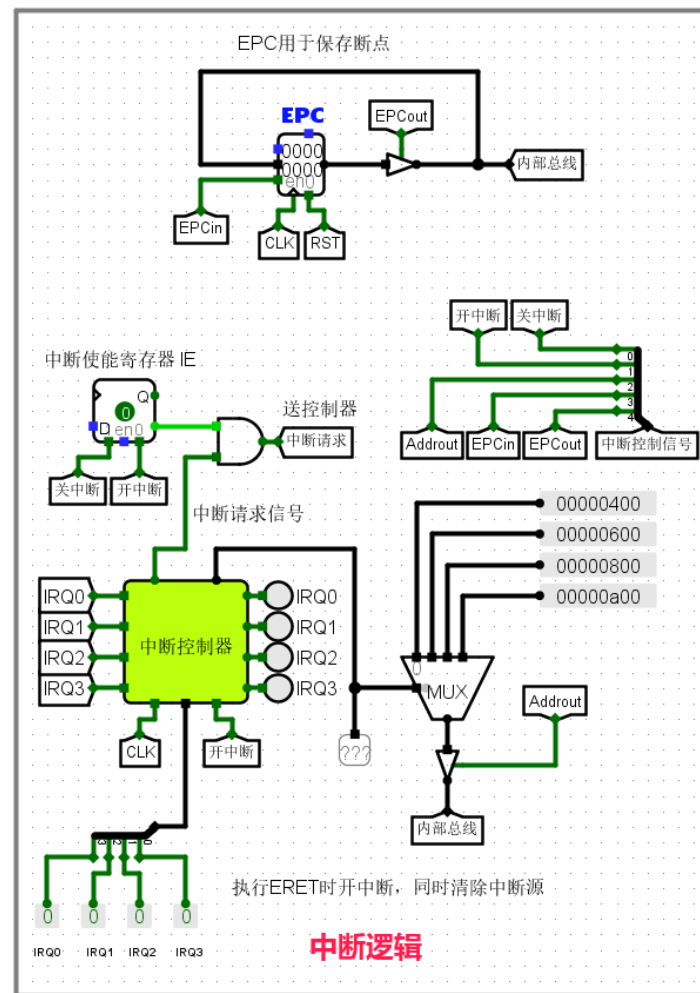
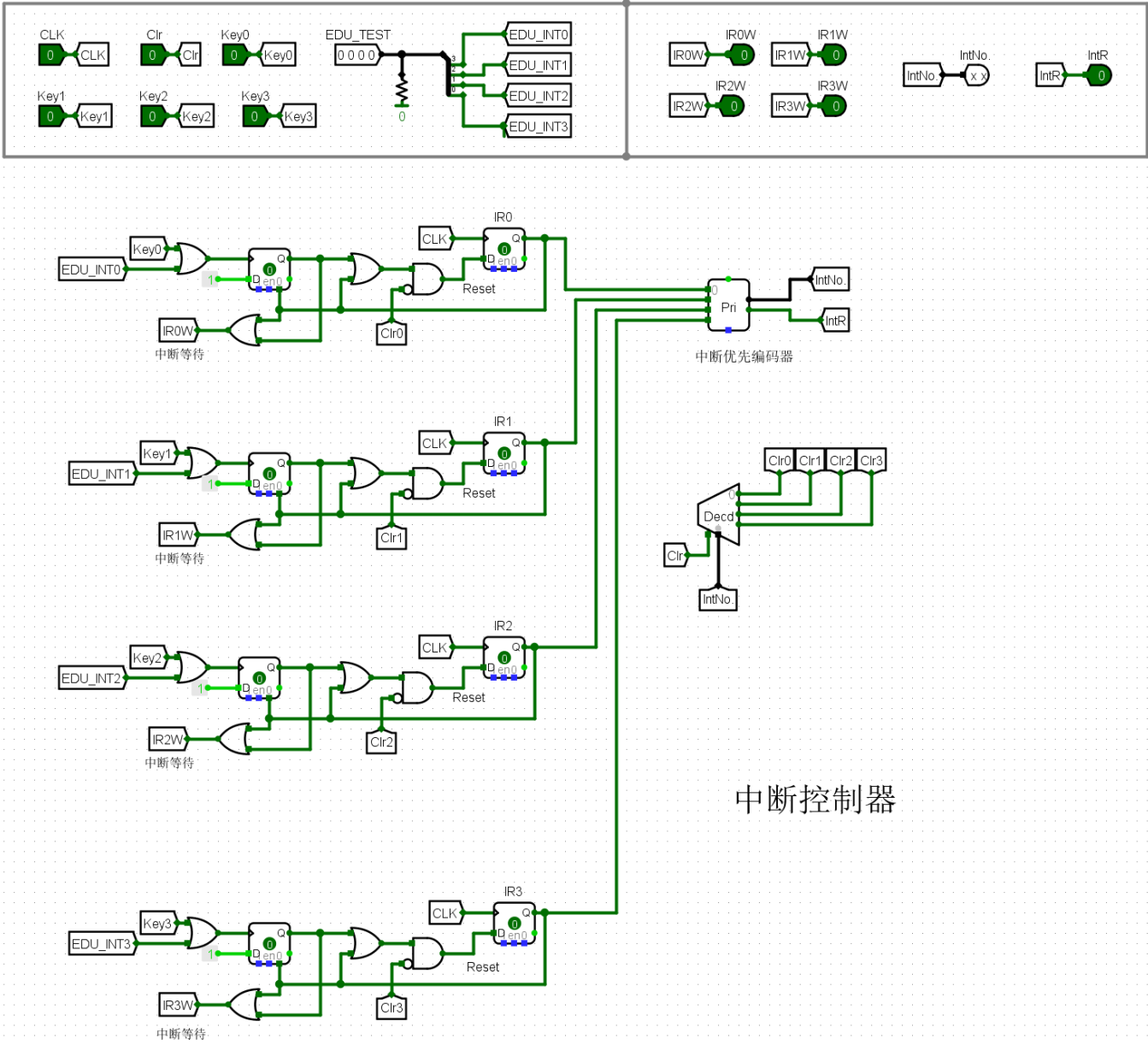
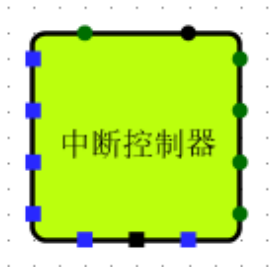


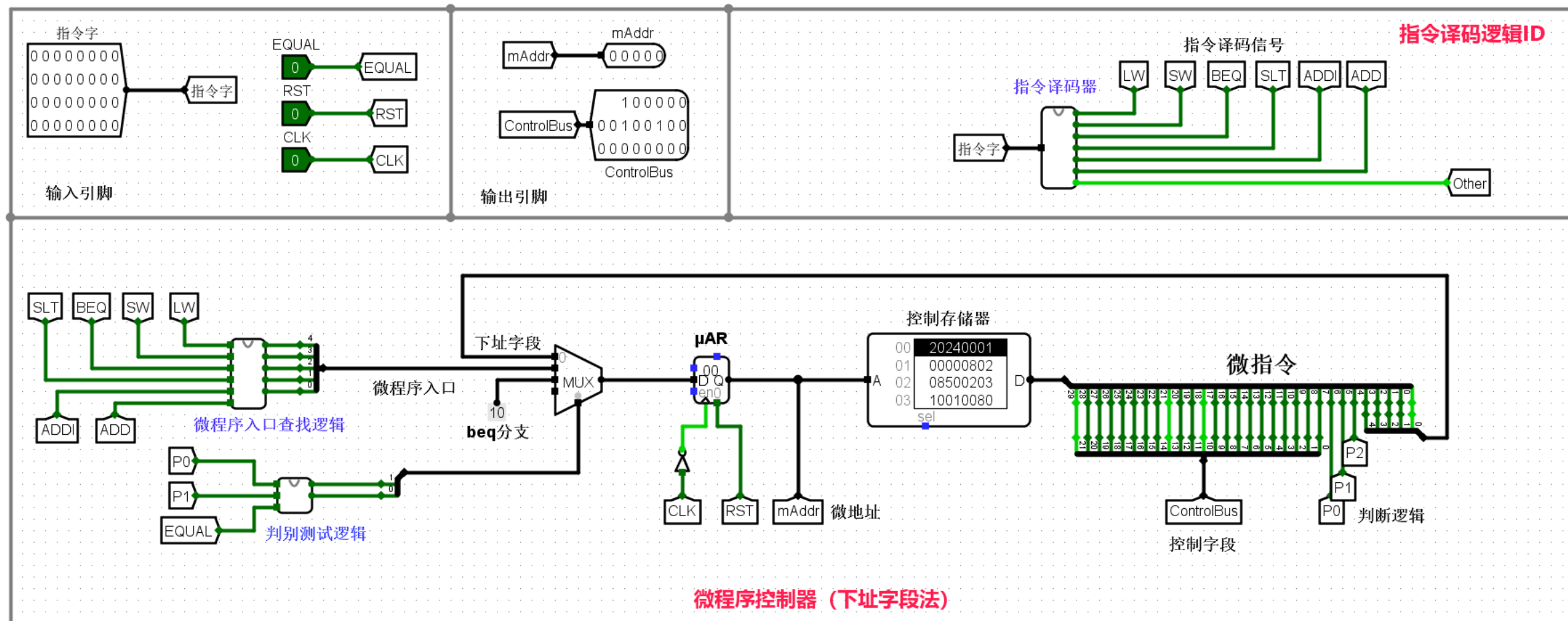
图6.64 支持中断的单总线结构计算机框图



中断控制器（4个中断源）

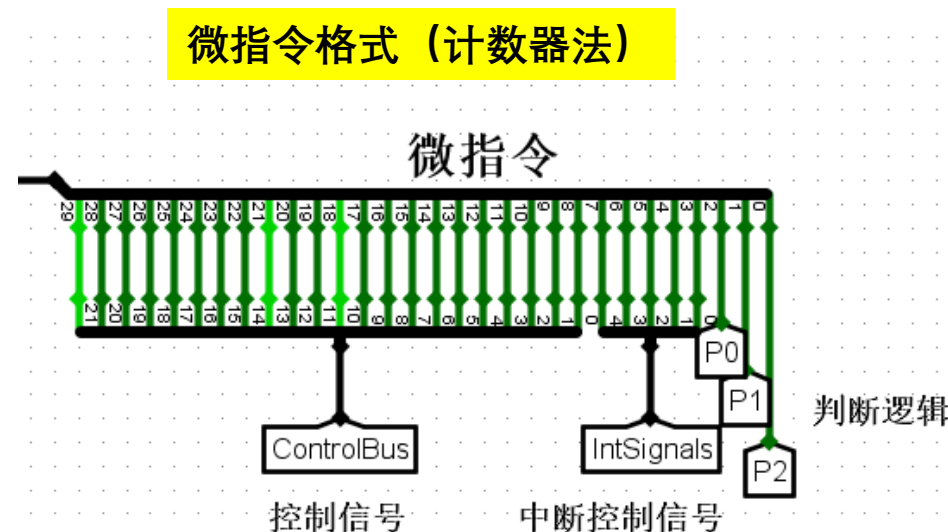
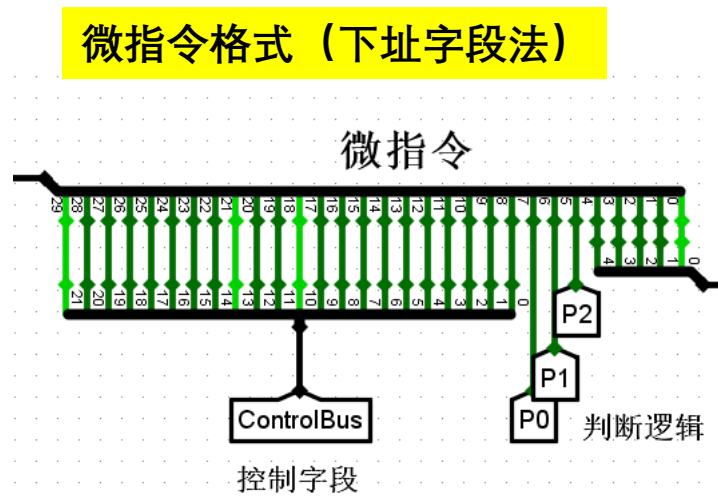


- 2、设计思路：
 - (2) 微程序控制器部分需要进行相应的修改



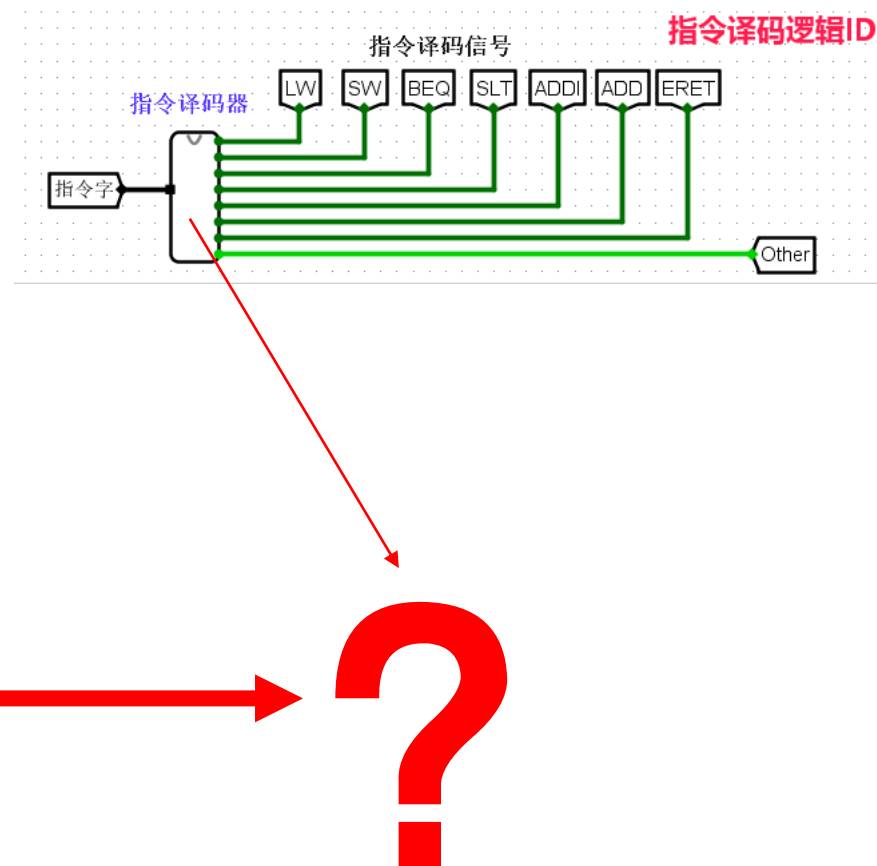
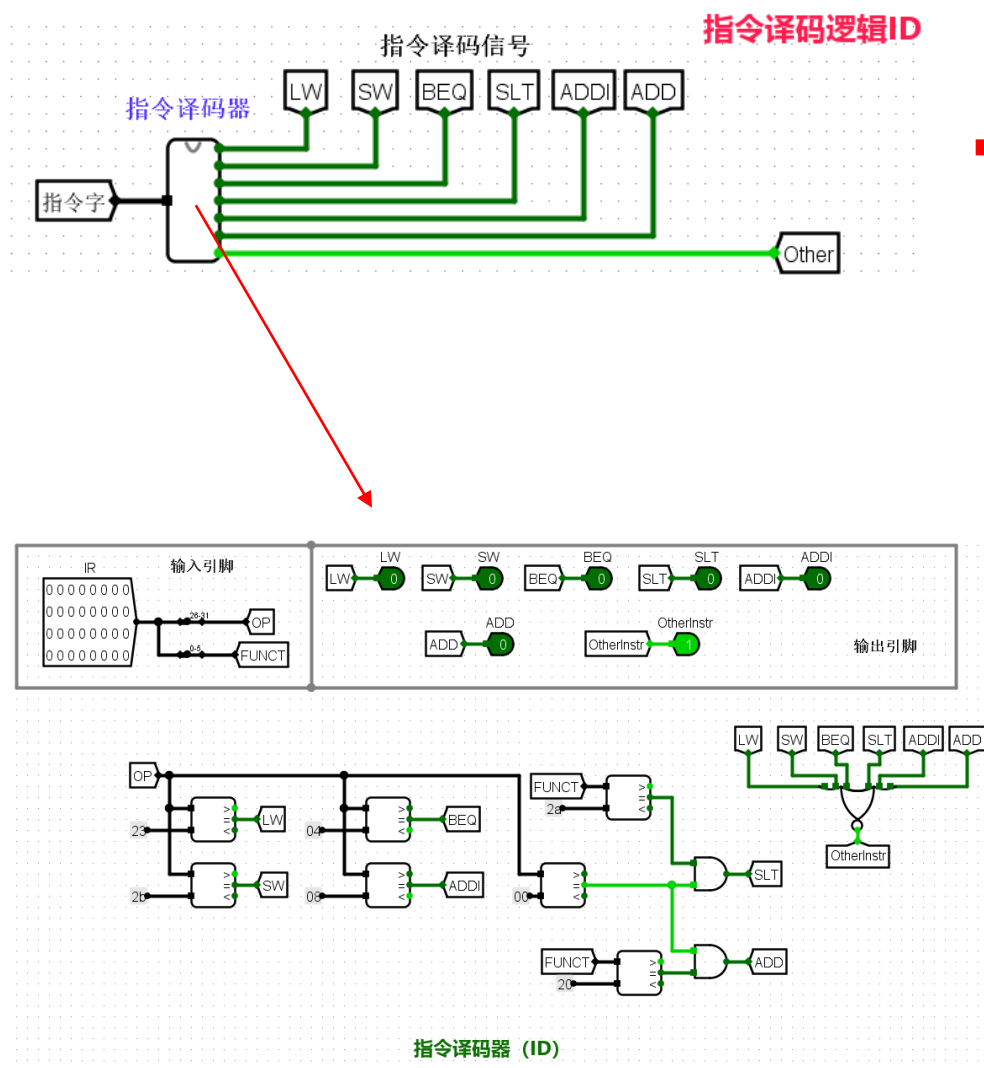
不支持中断的单总线MIPS处理器的微程序控制器（6条指令）

- 2、设计思路：
 - (3) 修改微指令格式：由下址字段法改为计数器法

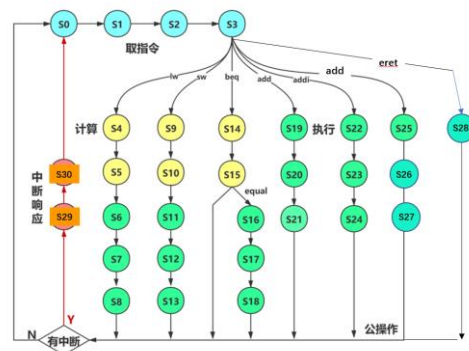
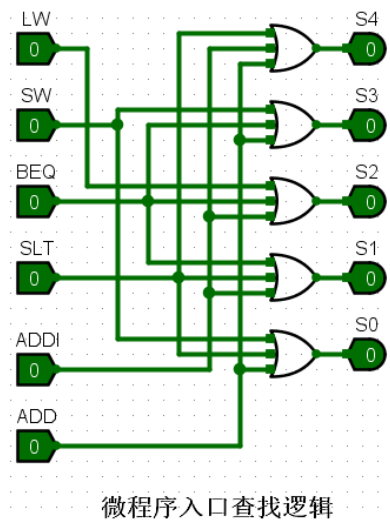
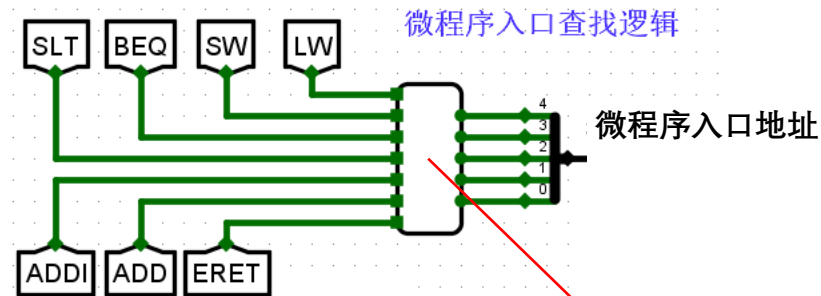
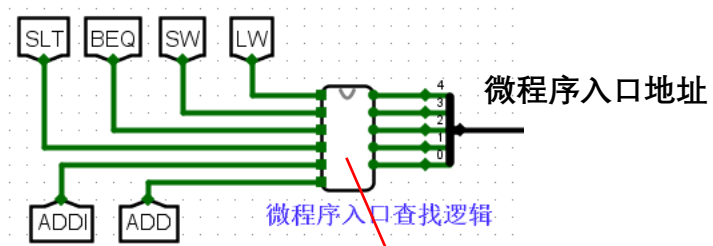


因为微指令的长度不能超过32位，增加5个中断控制信号后，如果采用下址字段法，微指令长度=22（单总线的控制信号）+5（中断控制信号）+3（判别测试）+5（下址字段）=35位；因此需要改为计数器法，微指令长度=22+5+3=30位

- 2、设计思路：
 - (4) 修改指令译码逻辑：增加ERET指令



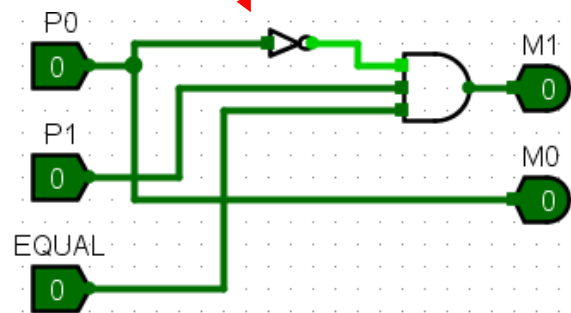
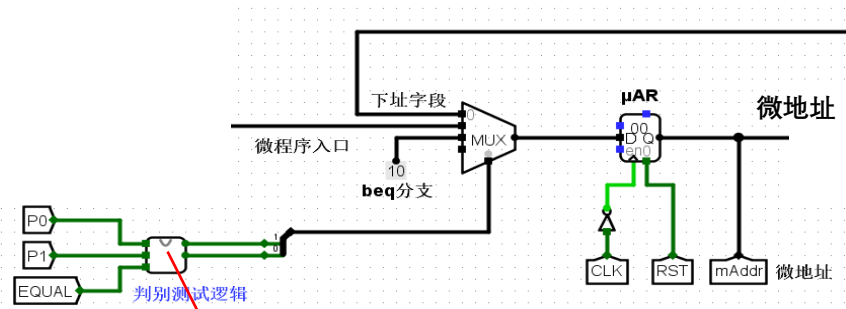
- 2、设计思路：
 - (5) 修改微程序入口查找逻辑：增加ERET指令



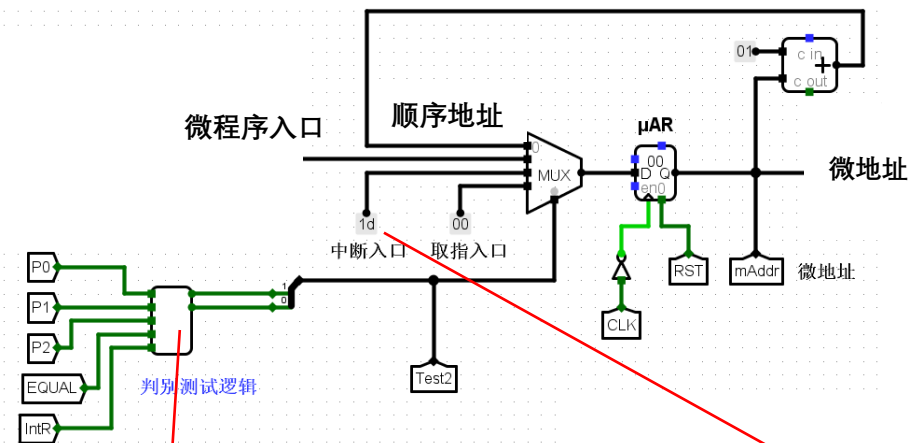
机器指令译码信号							微程序入口地址					
LW	SW	BEQ	SLT	ADDI	ADD	ERET	入口地址 10进制	S4	S3	S2	S1	S0
1							4	0	0	1	0	0
	1						9	0	1	0	0	1
		1					14	0	1	1	1	0
			1				19	1	0	0	1	1
				1			22	1	0	1	1	0
					1		25	1	1	0	0	1
						1	28	1	1	1	0	0

2、设计思路：

- (6) 修改**判别测试逻辑**：增加**P2**和**IntR**；多路选择器有4个输入（顺序地址、执行周期的微程序入口、中断周期微程序入口1d、取指周期入口00）



判别测试逻辑

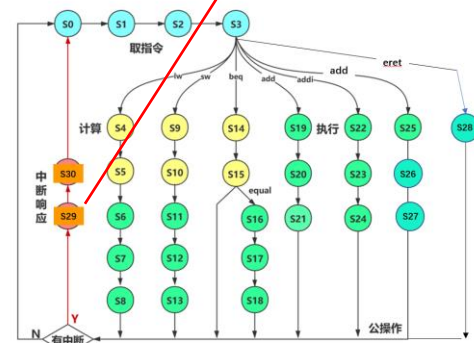


中断入口=1d=29=S29

?

输入 (填1或0, 不填为无关项x)						
P0	P1	P2	EQUAL	IntR	M1	M0
1					0	1
0	0	0			0	0
0	0	1		1	1	0
0	0	1		0	1	1
0	1	0			0	0
0	1	1	1		0	0
0	1	1	0	1	1	0
0	1	1	0	0	1	1

判别测试逻辑自动生成——单总线结构MIPS处理器（微程序+中断+6条指令）.xlsx



- 2、设计思路：

- (7) 修改控制存储器的内容（微程序）：采用计数器法，增加3条微指令

	微指令功能	PCOut	DRout	Zout	Rout	IR(In)	IR(Aout)	DR(In)	PCIn	ARIn	DRIn	XIn	RIn	IRIn	PSWin	Rs/Rt	RegDst	Add	Add4	SlT	READ	WRITE	EPCont	EPCLn	Address	STI	CLI	P0	P1	P2	微指令	微指令十六进制		
1	取指令	0	1						1			1								1											10000000100100000000000000000000	20240000		
2	取指令	1																		1											00000000000000000000000000000000	800		
3	取指令	2		1					1		1										1										00100001010000000000000000000000	8500200		
4	取指令	3		1										1													1				01000000000001000000000000000100	10010004		
5	LW	4			1								1					1													00010000000100000000000000000000	4040000		
6	LW	5				1													1												00001000000000000000000000000000	2001000		
7	LW	6		1						1																						00100000100000000000000000000000	8200000	
8	LW	7									1										1											00000000010000000000000000000000	100200	
9	LW	8		1									1																	1		01000000000010000000000000000001	10020001	
10	SW	9			1								1																			00010000000100000000000000000000	4040000	
11	SW	10				1													1													00001000000000000000000000000000	2001000	
12	SW	11		1						1																						00100000100000000000000000000000	8200000	
13	SW	12			1							1					1															00010000001000010000000000000000	4084000	
14	SW	13					1															1								1		00000010000000000000000000000001	800101	
15	BEQ	14			1								1																			00010000000100000000000000000000	4040000	
16	BEQ	15			1											1	1											1	1			0001000000000110000000000000011	400C003	
17	BEQ	16	1										1																			10000000000100000000000000000000	20040000	
18	BEQ	17					1												1													00000100000000000100000000000000	1001000	
19	BEQ	18		1						1																				1		00100001000000000000000000000001	8400001	
20	SLT	19			1								1																			00010000000100000000000000000000	4040000	
21	SLT	20				1											1				1											00010000000000000100010000000000	4004400	
22	SLT	21		1									1						1											1		00100000000010001000000000000001	8022001	
23	ADDI	22			1								1																			00010000000100000000000000000000	4040000	
24	ADDI	23				1													1													00001000000000000000000000000000	2001000	
25	ADDI	24		1									1																	1		00100000000010000000000000000001	8020001	
26	ADD	25			1								1																			00010000000100000000000000000000	4040000	
27	ADD	26			1												1		1													00010000000000010100000000000000	4005000	
28	ADD	27		1									1					1														00100000000010001000000000000001	8022001	
29	ERET	28						1															1			1				1		00000001000000000000000000000001	400091	
30	INT	29	1																					1				1				10000000000000000000000000000001	20000048	
31	INT	30						1																		1					1		00000001000000000000000000000001	400021

- 3、**编写**含有4个中断服务程序的**累加和程序**（使用6+1条指令：lw、sw、beq、slt、addi、add、eret）
- 4、在MIPS汇编器上进行**汇编**，得到机器码，并**添加相关的内容**，使4个中断服务程序的起始地址分别为：

#IRQ0中断服务程序的入口地址：1024 = 400H	RAM对应100
#IRQ1中断服务程序的入口地址：1536 = 600H	RAM对应180
#IRQ2中断服务程序的入口地址：2048 = 800H	RAM对应200
#IRQ3中断服务程序的入口地址：2560 = A00H	RAM对应280

- 5、在支持中断控制的单总线结构 MIPS 处理器数据通路（微程序控制器+6条指令）上**运行**含有中断服务程序的**累加和程序**，并测试**4个中断请求**
- 6、在支持中断控制的单总线结构 MIPS 处理器数据通路（微程序控制器+6条指令）上**运行**含有中断服务程序的程序的**冒泡法排序程序**、计算**费波那契数列程序**，并测试**4个中断请求**

Thanks