

年级	2018	班号	计科 1804	学号																																										
专业	计算机科学与技术			姓名																																										
实验名称	实验三：微程序控制器组成实验			实验类型	设计型	综合型	创新型																																							
					√	√																																								
实验目的	(1) 掌握并熟悉 MIPS 指令架构, MIPS 指令周期流程, 及其微程序编写。 (2) 理解微程序控制器基本工作的一般原理。 (3) 熟练理解单总线结构 MIPS CPU 结构及其工作原理。 (4) 熟悉并理解单总线结构 MIPS CPU 结构的微程序入口地址产生逻辑。 (5) 熟悉并理解单总线结构 MIPS CPU 结构的微程序条件判断测试产生逻辑。 (6) 熟练掌握 MIPS 指令连续和单步执行过程。																																													
实验原理及电路	(一) MIPS 指令架构, MIPS 指令周期流程, 及其微程序编写																																													
	1、MIPS 指令架构图																																													
	32 个寄存器; 按照字节编址。32 个寄存器集成一个小容量的高速存储器, 它们的命名标识、地址编码和基本用途如下表所示:																																													
	<table><tr><th>寄存器名字</th><th>约定命名</th><th>用途</th></tr><tr><td>\$0</td><td>zero</td><td>总是为 0</td></tr><tr><td>\$1</td><td>at</td><td>留作汇编器生成一些合成指令</td></tr><tr><td>\$2、\$3</td><td>v0、v1</td><td>用来存放子程序返回值</td></tr><tr><td>\$4~\$7</td><td>a0~a3</td><td>调用子程序时, 使用这 4 个寄存器传输前 4 个非浮点参数</td></tr><tr><td>\$8~\$15</td><td>t0~t7</td><td>临时寄存器, 子程序使用时可以不用存储和恢复</td></tr><tr><td>\$16~\$23</td><td>s0~s7</td><td>子程序寄存器变量, 改变这些寄存器值的子程序必须存储旧的值并在退出前恢复, 对调用程序来说值不变</td></tr><tr><td>\$24、\$25</td><td>t8、t9</td><td>临时寄存器, 子程序使用时可以不用存储和恢复</td></tr><tr><td>\$26、\$27</td><td>\$k0、\$k1</td><td>由异常处理程序使用</td></tr><tr><td>\$28 或 \$gp</td><td>gp</td><td>全局指针</td></tr><tr><td>\$29 或 \$sp</td><td>sp</td><td>堆栈指针</td></tr><tr><td>\$30 或 \$fp</td><td>s8/fp</td><td>子程序可以用来做堆栈帧指针</td></tr><tr><td>\$31</td><td>ra</td><td>存放子程序返回地址</td></tr></table>							寄存器名字	约定命名	用途	\$0	zero	总是为 0	\$1	at	留作汇编器生成一些合成指令	\$2、\$3	v0、v1	用来存放子程序返回值	\$4~\$7	a0~a3	调用子程序时, 使用这 4 个寄存器传输前 4 个非浮点参数	\$8~\$15	t0~t7	临时寄存器, 子程序使用时可以不用存储和恢复	\$16~\$23	s0~s7	子程序寄存器变量, 改变这些寄存器值的子程序必须存储旧的值并在退出前恢复, 对调用程序来说值不变	\$24、\$25	t8、t9	临时寄存器, 子程序使用时可以不用存储和恢复	\$26、\$27	\$k0、\$k1	由异常处理程序使用	\$28 或 \$gp	gp	全局指针	\$29 或 \$sp	sp	堆栈指针	\$30 或 \$fp	s8/fp	子程序可以用来做堆栈帧指针	\$31	ra	存放子程序返回地址
	寄存器名字	约定命名	用途																																											
	\$0	zero	总是为 0																																											
	\$1	at	留作汇编器生成一些合成指令																																											
	\$2、\$3	v0、v1	用来存放子程序返回值																																											
	\$4~\$7	a0~a3	调用子程序时, 使用这 4 个寄存器传输前 4 个非浮点参数																																											
	\$8~\$15	t0~t7	临时寄存器, 子程序使用时可以不用存储和恢复																																											
\$16~\$23	s0~s7	子程序寄存器变量, 改变这些寄存器值的子程序必须存储旧的值并在退出前恢复, 对调用程序来说值不变																																												
\$24、\$25	t8、t9	临时寄存器, 子程序使用时可以不用存储和恢复																																												
\$26、\$27	\$k0、\$k1	由异常处理程序使用																																												
\$28 或 \$gp	gp	全局指针																																												
\$29 或 \$sp	sp	堆栈指针																																												
\$30 或 \$fp	s8/fp	子程序可以用来做堆栈帧指针																																												
\$31	ra	存放子程序返回地址																																												
(图 1)																																														
<table><tr><th>#</th><th>MIPS指令</th><th>RTL功能描述</th></tr><tr><td>1</td><td>slt rd,rs,rt</td><td><math>R[rd] \leftarrow R[rs] &lt; R[rt]</math> 小于置1, 有符号比较</td></tr><tr><td>2</td><td>addi rt,rs,imm</td><td><math>R[rt] \leftarrow R[rs] + \text{SignExt}(imm)</math> 不考虑溢出</td></tr><tr><td>3</td><td>lw rt,imm(rs)</td><td><math>R[rt] \leftarrow M[R[rs] + \text{SignExt}(imm)]</math></td></tr><tr><td>4</td><td>sw rt,imm(rs)</td><td><math>M[R[rs] + \text{SignExt}(imm)] \leftarrow R[rt]</math></td></tr><tr><td>5</td><td>beq rs,rt,imm</td><td>if(<math>R[rs] = R[rt]</math>) <math>PC \leftarrow PC + \text{SignExt}(imm) \ll 2</math></td></tr></table>							#	MIPS指令	RTL功能描述	1	slt rd,rs,rt	$R[rd] \leftarrow R[rs] < R[rt]$ 小于置1, 有符号比较	2	addi rt,rs,imm	$R[rt] \leftarrow R[rs] + \text{SignExt}(imm)$ 不考虑溢出	3	lw rt,imm(rs)	$R[rt] \leftarrow M[R[rs] + \text{SignExt}(imm)]$	4	sw rt,imm(rs)	$M[R[rs] + \text{SignExt}(imm)] \leftarrow R[rt]$	5	beq rs,rt,imm	if( $R[rs] = R[rt]$ ) $PC \leftarrow PC + \text{SignExt}(imm) \ll 2$																						
#	MIPS指令	RTL功能描述																																												
1	slt rd,rs,rt	$R[rd] \leftarrow R[rs] < R[rt]$ 小于置1, 有符号比较																																												
2	addi rt,rs,imm	$R[rt] \leftarrow R[rs] + \text{SignExt}(imm)$ 不考虑溢出																																												
3	lw rt,imm(rs)	$R[rt] \leftarrow M[R[rs] + \text{SignExt}(imm)]$																																												
4	sw rt,imm(rs)	$M[R[rs] + \text{SignExt}(imm)] \leftarrow R[rt]$																																												
5	beq rs,rt,imm	if( $R[rs] = R[rt]$ ) $PC \leftarrow PC + \text{SignExt}(imm) \ll 2$																																												
(图 2)																																														
2、MIPS 指令周期流程图																																														



如果程序在执行过程中发生了中断，则要进行中断响应，否则直接返回取指令周期进行取指。

指令的周期分为取指令周期, 计算周期, 执行周期和中断周期.

[illegible]

(图 4)

将生成的十六进制微指令复制到控制存储器中。

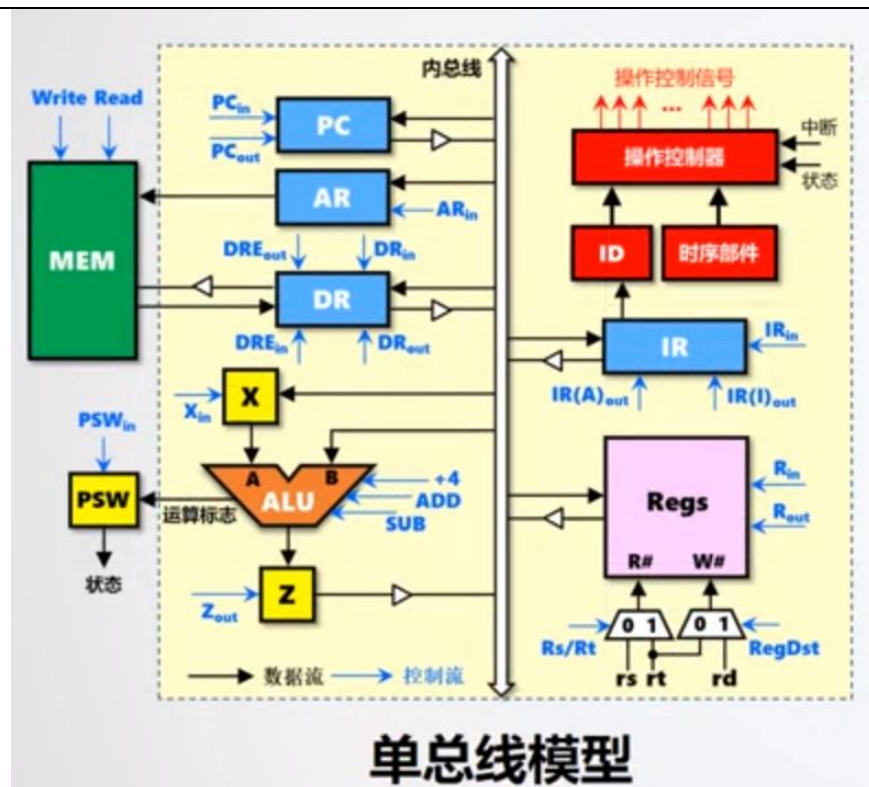
## （二）单总线结构 MIPS CPU 结构及其工作原理

### 1、阐述微程序控制器基本工作的一般原理

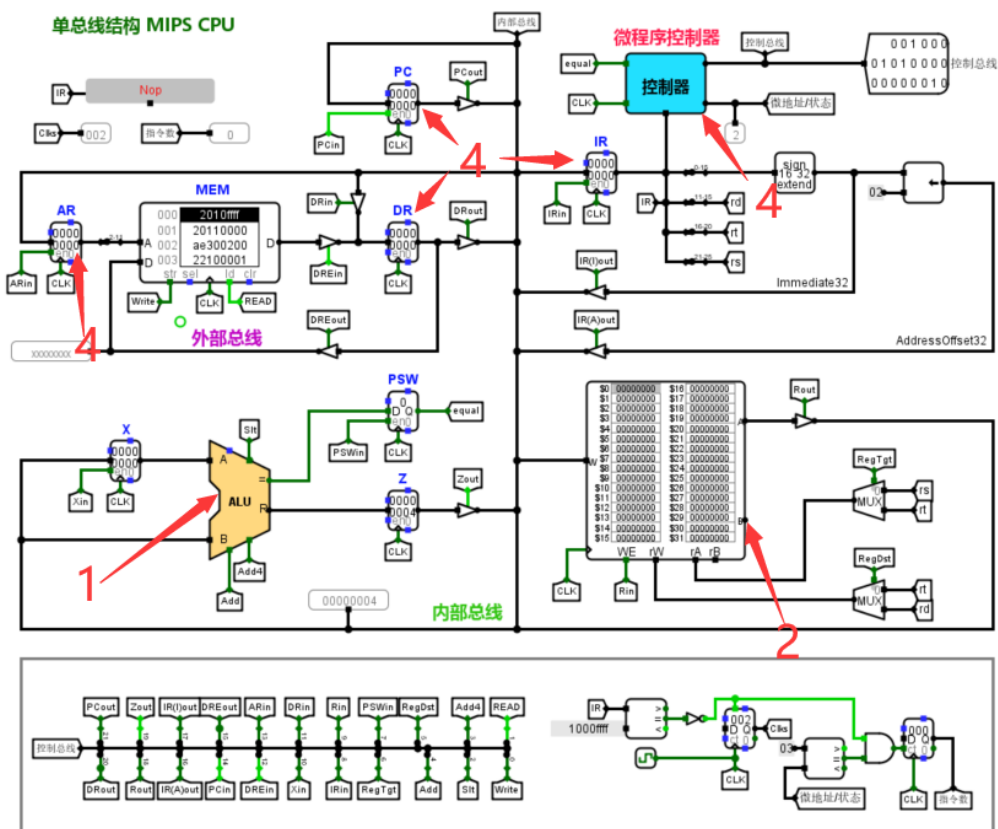
根据 IR（指令寄存器）中的操作码，找到与之对应的控存中的一段微程序的入口地址，并按指令功能所确定的次序，逐条从控制存储器中读出微指令，以驱动计算机各部件正确运行。

## 2、单总线结构 MIPS CPU 结构及其工作原理

单总线结构 MIPS CPU 结构图:



(图 5)



(图 6)

主要部件都连接在总线上，各部件间通过总线进行传输。  
黑色箭头为数据流，蓝色箭头为控制流。

各个部件：

1. ALU:

设置暂存器 X 和 Z，分别用于暂存内部总线输入的值和输出结果，其中 Z 仅受时钟信号的控制，ALU 上的控制信号+4、ADD、SUB 不能同时有效，只能给出一种运算。ALU 还包括一个状态寄存器，PSW 保存运算标志。

## 2. Regs:

通用寄存器堆，用于控制输入和输出，有一个读端口和一个写端口，读写控制信号分别为 Rin 和 Rout。

## 3. 其他寄存器:

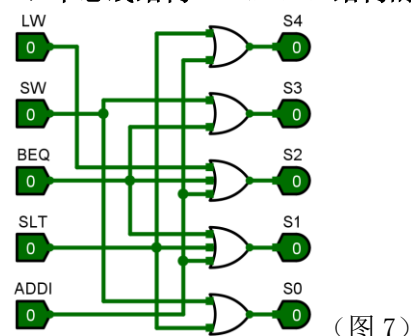
PC、AR（地址寄存器）、DR（缓冲寄存器），IR(指令寄存器)

## 4. 控制器:

产生操作控制信号，控制流的产生会引起数据的流动，从而形成相应的数据通路，完成指令的功能。

# (三) 单总线结构 MIPS CPU 结构的微程序入口地址产生逻辑

## 1、单总线结构 MIPS CPU 结构的微程序入口地址产生逻辑图



(图 7)

## 2、单总线结构 MIPS CPU 结构的微程序入口地址产生逻辑一般介绍

通过填写以下表格：

机器指令译码信号					微程序入口地址					
LW	SW	BEQ	SLT	ADDI	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

(图 8)

LW	SW	BEQ	SLT	ADDI	最小项表达式	S4	S3	S2	S1	S0
LW&					LW			LW+		
	SW&				SW		SW+			SW+
		BEQ&			BEQ		BEQ+	BEQ+	BEQ+	
			SLT&		SLT	SLT+			SLT+	SLT+
				ADDI&	ADDI	ADDI+		ADDI+	ADDI+	
						SLT+ADDI	SW+BEQ	LW+BEQ+ADDI	BEQ+SLT+ADDI	SW+SLT

(图 9)

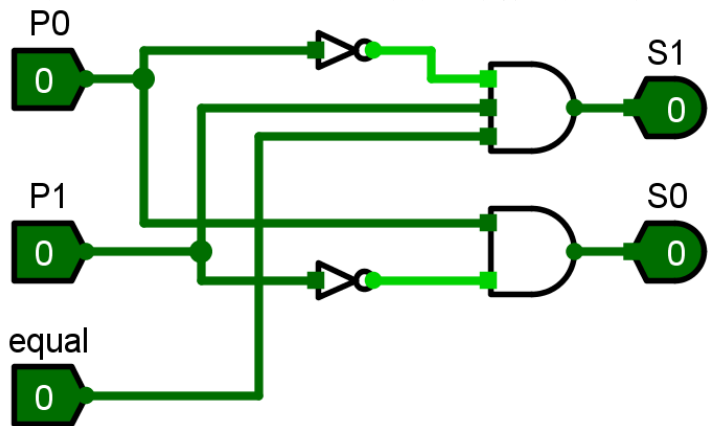
用 excel 的公式计算出微程序入口地址产生逻辑的公式为：

$$s0=SLT + SW$$

s1=ADDI + SLT + BEQ  
s2=ADDI + SLT + BEQ  
s3=BEQ + SW  
s4=ADDI + SLT  
将公式复制到 logsim 中，可自动生成电路。

（四）单总线结构 MIPS CPU 结构的微程序条件判断测试产生逻辑

1、单总线结构 MIPS CPU 结构的微程序条件判断测试产生逻辑图



（图 10）

2、单总线结构 MIPS CPU 结构的微程序条件判断测试产生逻辑一般介绍  
通过填写以下表格：

输入（填1或0，不填为无关项x）					输出（只填写为1的情况）							
P0	P1	P2	equal	IntR	S2	S1	S0	Out4	Out5	Out6	Out7	Out8
0	0				0	0	0					
1	0				0	0	1					
0	1		0			0	0					
0	1		1			1	0					

P0	P1	P2	equal	IntR	In6	In7	In8	In9	In10	In11	In12	最小项表达式	S2	S1	S0	Out4	Out5
$\sim P0 \& \sim P1$												$\sim P0 \& \sim P1$					
$P0 \& \sim P1$												$P0 \& \sim P1$					
$\sim P0 \& P1$			$\sim equal$									$\sim P0 \& P1 \& \sim equal$					
$\sim P0 \& P1$			$equal$									$\sim P0 \& P1 \& equal$					

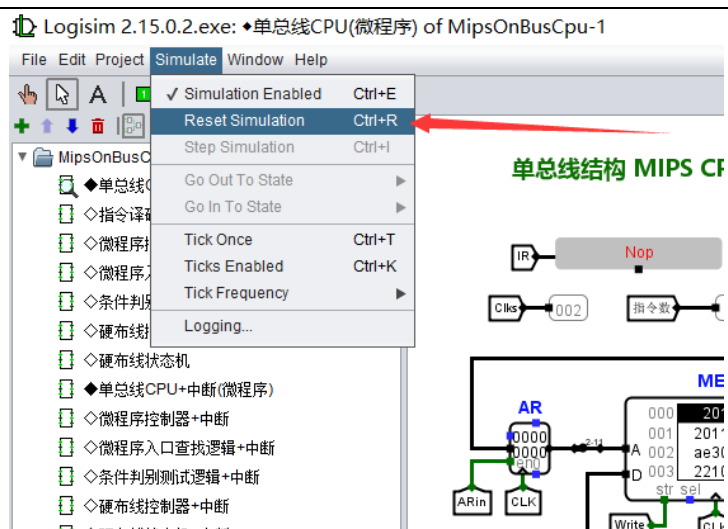
（图 11）

用 excel 的公式计算出微程序条件判断测试产生逻辑的公式为：  
s0= P0  $\sim$  P1  
s1= $\sim$  P0 P1 equal  
将公式复制到 logsim 中，可自动生成电路。

实验  
验证  
过程  
及  
结果

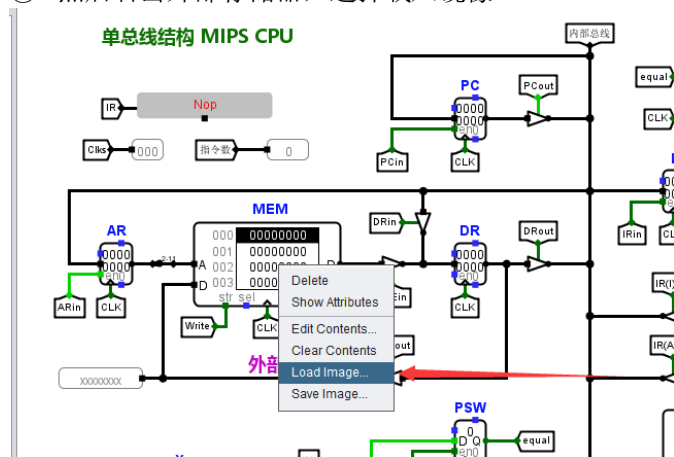
1、连续运行指令的操作：运行冒泡程序的实验操作过程及其数据记载与分析  
（1） 冒泡程序的导入过程及其注意事项  
① 先将电路复位

分析



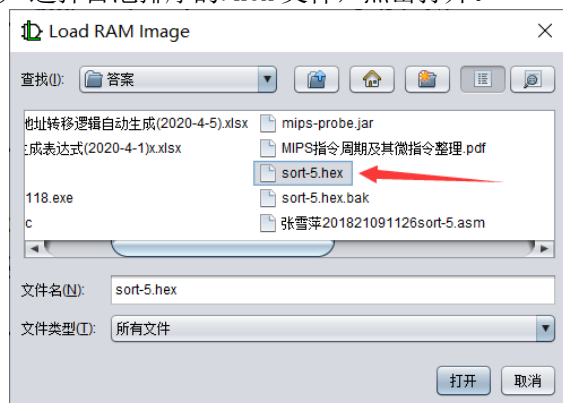
(图 12)

② 然后右击外部存储器，选择载入镜像



(图 13)

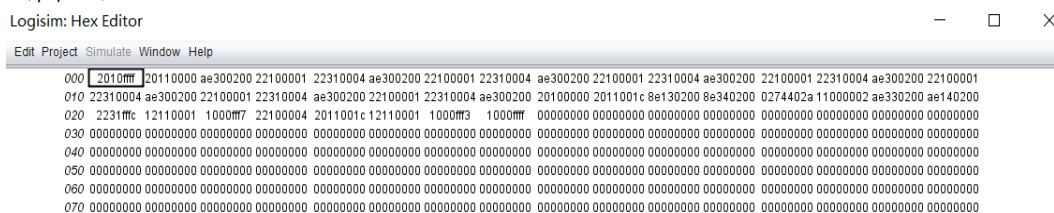
③ 选择冒泡排序的.hex 文件，点击打开。



(图 14)

④ 载入成功后右击外部存储器，编辑，效果如图：

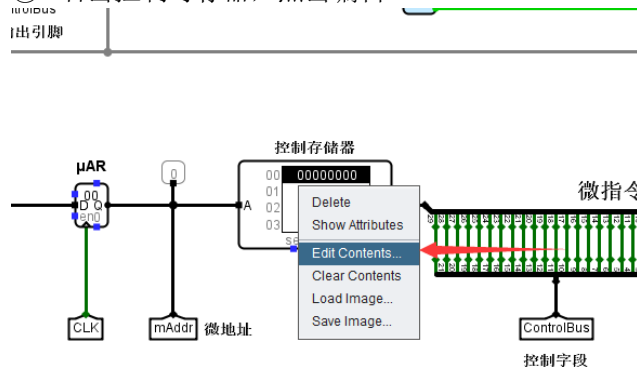
(图 15)



(图 16)

## (2) 微程序导入过程及其注意事项

① 右击控制寄存器，点击编辑

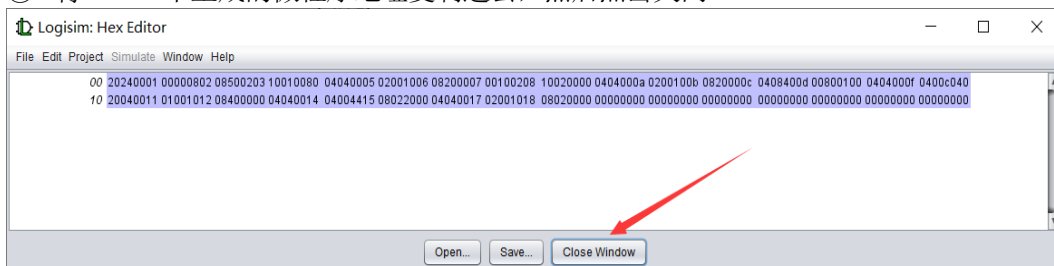


是上跳沿还是下跳沿有效?

### 微程序控制器（下址字段法）

(图 17)

② 将 excel 中生成的微程序地址复制进去, 然后点击关闭。



(图 18)

### (3) 执行冒泡排序程序过程、数据记载及其分析

运行:

将时钟频率调到最大，点击 ctrl+k 自动执行，因为冒泡排序结束的标志是进入死循环，此时时钟周期数和指令数不再增加，按 ctrl+k 暂停。

数据记载:



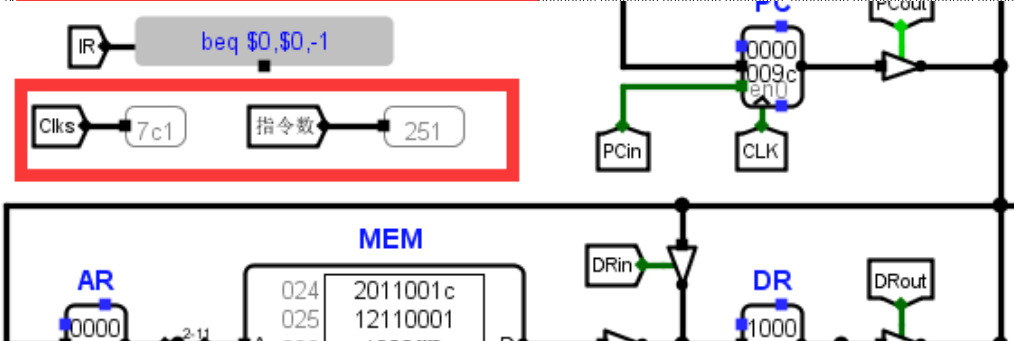
① 程序完成赋值时，内存中的数据：

```
000 2010fff 20110000 ae300200 22100001 22310004 ae300200 22100001 22310004 ae300200 22100001 22310004 ae300200 22100001 22310004 ae300200 22100001
010 22310004 ae300200 22100001 22310004 ae300200 22100001 22310004 ae300200 20100000 2011001c 8e130200 8e340200 0274402a 11000002 ae330200 ae140200
020 2231fff 12110001 1000fff7 22100004 2011001c 12110001 1000fff3 1000fff 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
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 ffffff 00000000 00000001 00000002 00000003 00000004 00000005 00000006 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
```

(图 19)

②程序运行结束时，打开外部存储器，可以看到降序排列的结果，另外可以观察到时钟周期是为 7c1，共执行了 251 条指令。

```
000 2010fff 20110000 ae300200 22100001 22310004 ae300200 22100001 22310004 ae300200 22100001 22310004 ae300200 22100001 22310004 ae300200 22100001
010 22310004 ae300200 22100001 22310004 ae300200 22100001 22310004 ae300200 20100000 2011001c 8e130200 8e340200 0274402a 11000002 ae330200 ae140200
020 2231fff 12110001 1000fff7 22100004 2011001c 12110001 1000fff3 1000fff 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
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 00000006 00000005 00000004 00000003 00000002 00000001 00000000 ffffff 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
```



(图 20)

结果分析：

已经写好的冒泡排序是从-1 到 6 递增添加到外部存储器中，所以开始是在外部存储器中升序排列的，而冒泡排序程序的任务是将外部存储器中的数据降序排列，因此实验结果和理论结果相同。

## 2、单步运行指令的操作，其过程数据记载与分析

运行：

点击 ctrl+t 单步执行，观察各个寄存器和暂寄存器中值的变化。

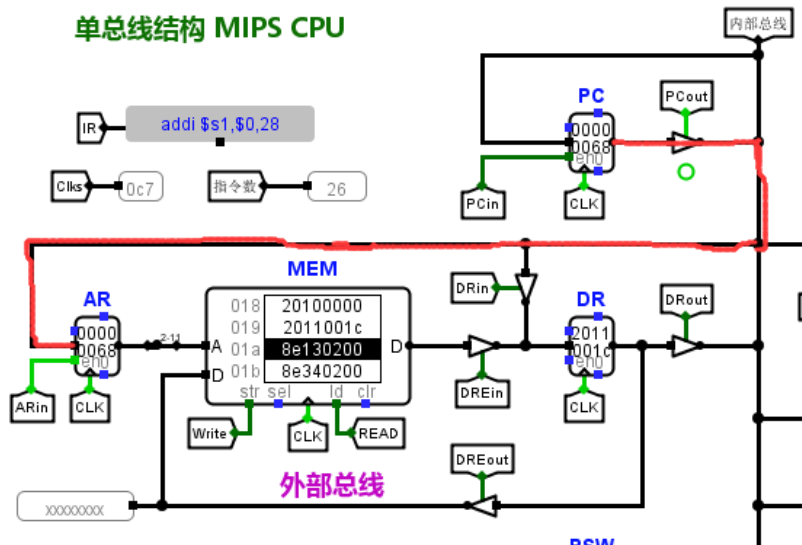
数据记载：

取指和译码：

① PCout 打开，PC 中的值进入地址寄存器 AR 中。



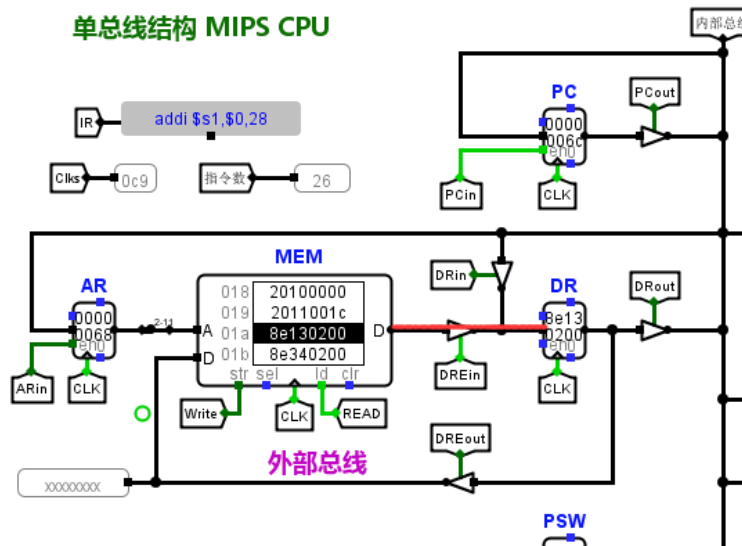
### 单总线结构 MIPS CPU



(图 21)

②根据 AR 的值，选中外部寄存器 MEM 中指定地址的值，DREin 打开，MEM 中选中的值进入缓冲寄存器 DR 中。

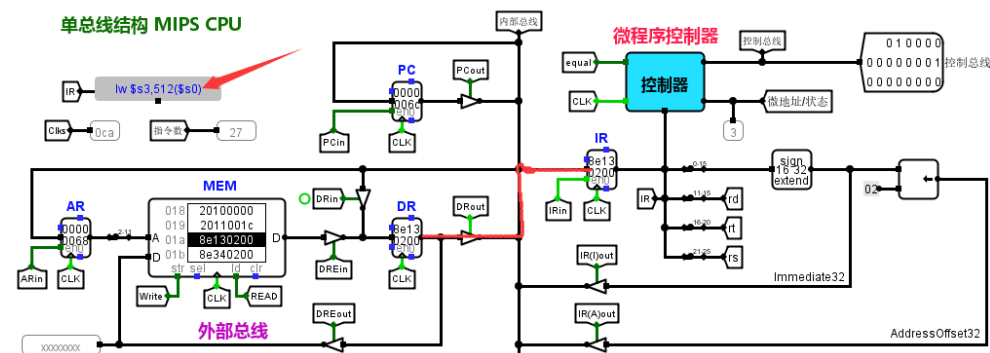
### 单总线结构 MIPS CPU



(图 22)

③DRout 打开，DR 中的值进入指令寄存器中，指令在控制器中完成译码，再由操作控制器产生操作控制信号，控制流的产生会引起数据的流动，从而形成相应的数据通路，完成指令的功能。

### 单总线结构 MIPS CPU

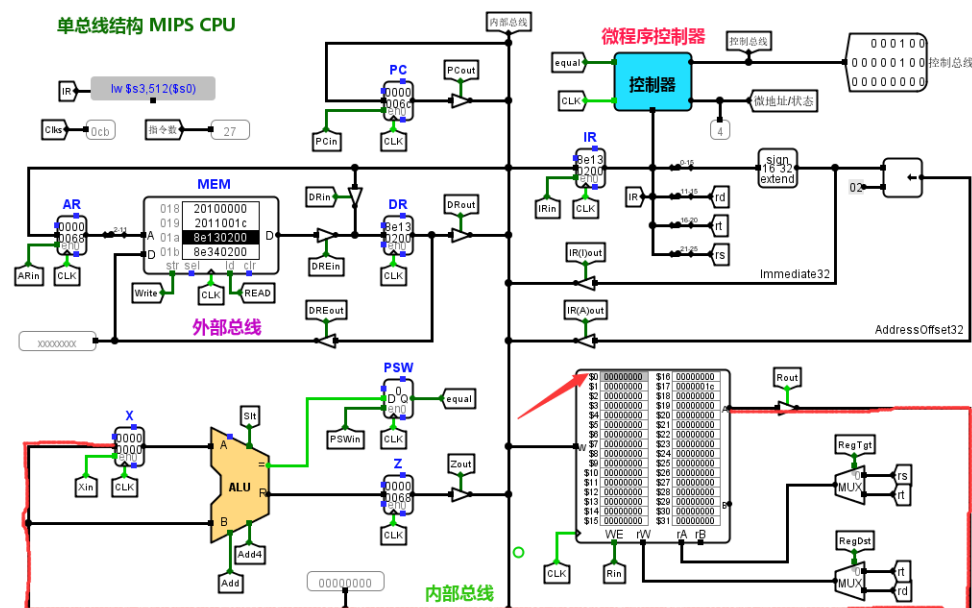


(图 23)

这里的译码结果是 `lw $s3, 512($s0)` ( $\$s3 = \text{MEM}[512 + \$s0]$ ，将内存  $[512 + \$s0]$  中的值取出

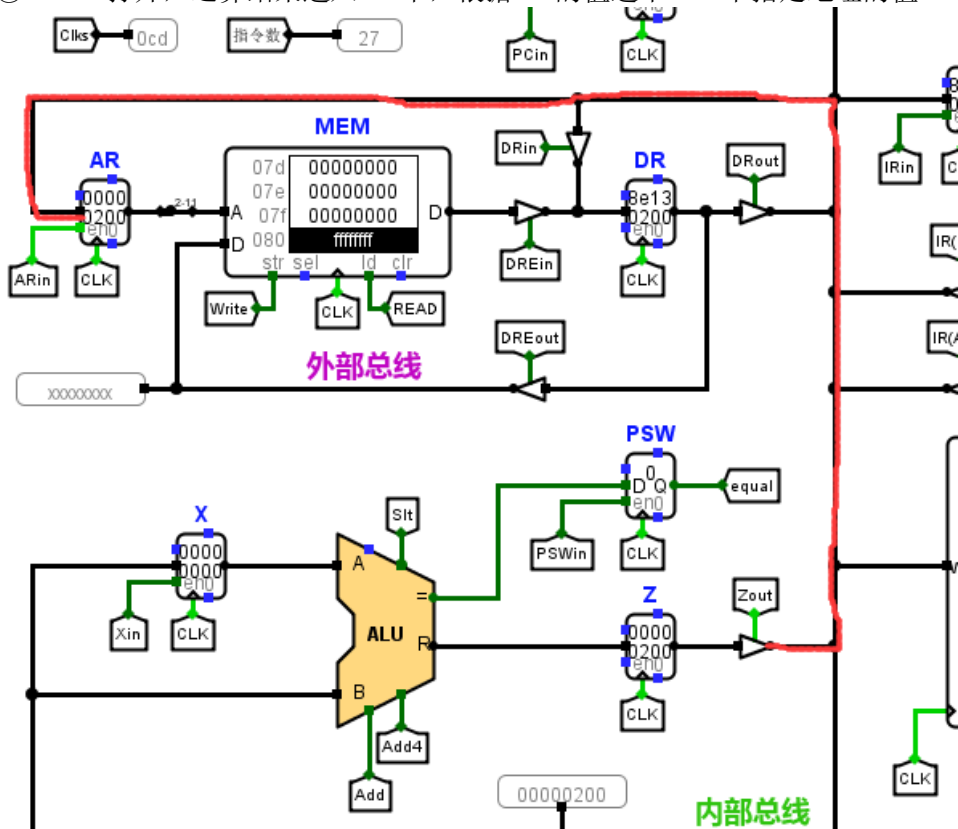
来，放到\$S3 中)，以这条指令为例描述指令执行的过程。

①先计算内存地址， $\$0+512$ ，Rout 打开， $\$0$  中的值进入暂存器 X 中，512 经过位扩展后到达 ALU 的 B 中，在 ALU 中完成加法运算，将结果放在暂存器 Z 中。



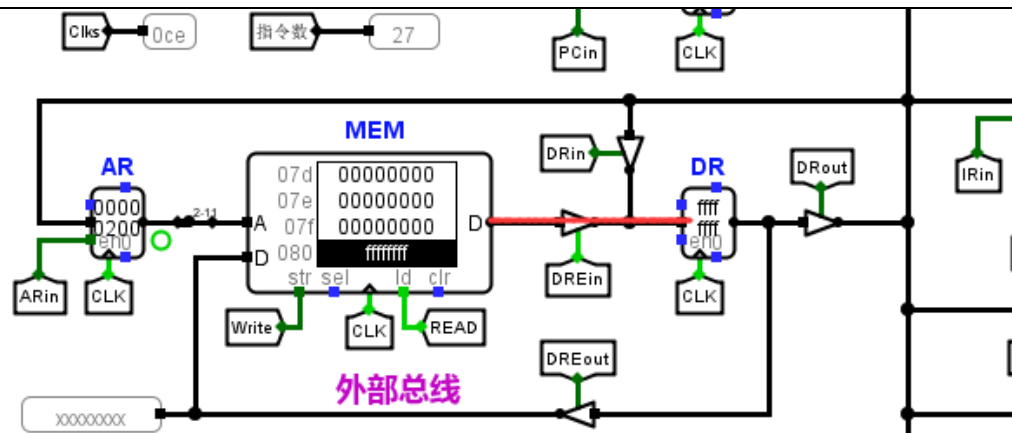
(图 24)

② Zout 打开，运算结果进入 AR 中，根据 AR 的值选中 MEM 中指定地址的值。



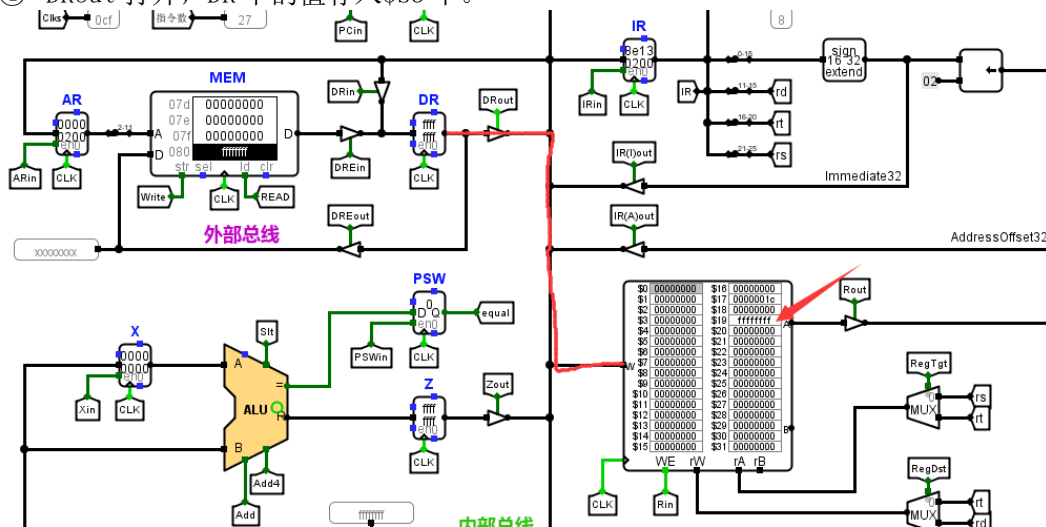
(图 25)

③ DREin 打开，MEM 中选中的值进入 DR 中。



(图 26)

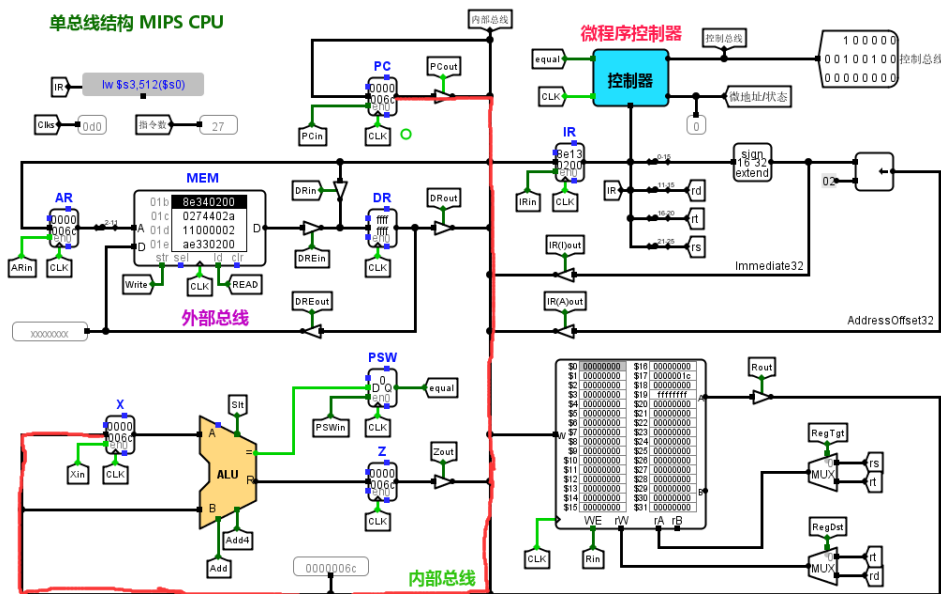
④ DRout 打开, DR 中的值存入 \$s3 中。



(图 27)

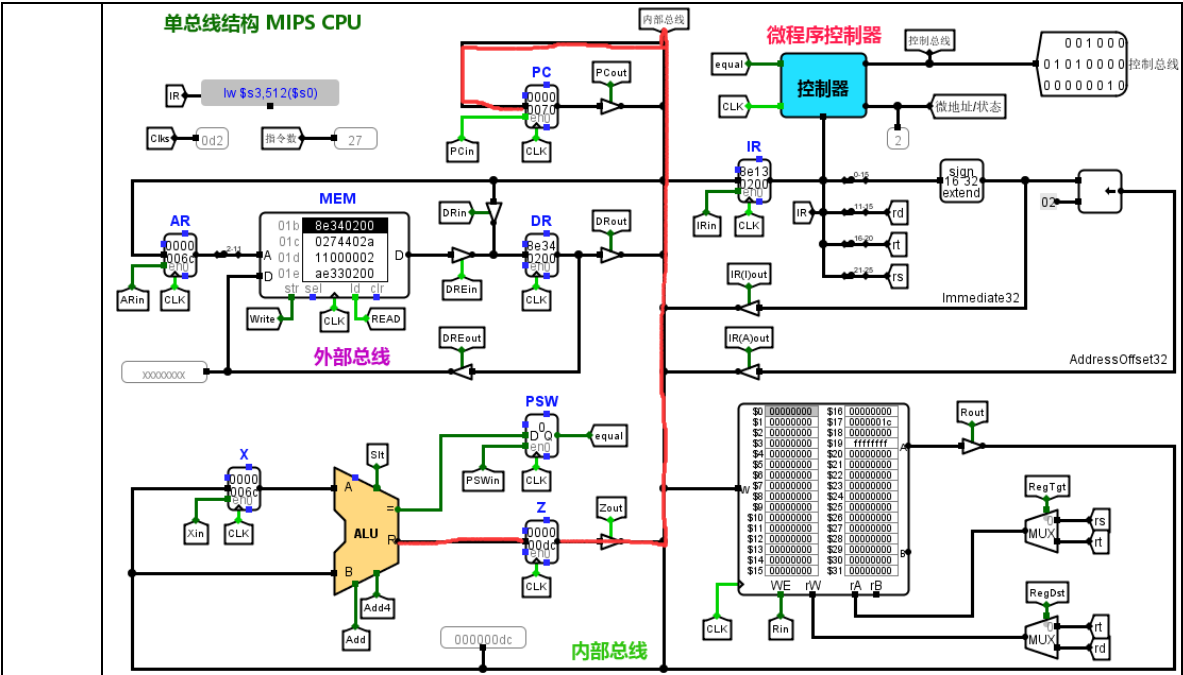
另外, PC++和取指是并行执行的, 通过下一条指令展示 PC++的过程。

①PCout 打开, PC 中的值进入暂存器 X, 在 ALU 中完成加一操作, 将结果放到暂存器 Z 中。



(图 28)

②Zout 打开, Z 中的结果存入 PC 中。



(图 29)

**结果分析:**

PC 加的是指令的字节长度，MIPS 指令是 32 位，4 个字节，所以加的是 4。综合上述，实验结果和理论一致，MIPS CPU 仿真基本实现。

心得  
体会

评语:

成绩: \_\_\_\_\_ 教师签名: \_\_\_\_\_ 日期: \_\_\_\_\_