题　　目: TEC-5H 模型计算机电路设计与仿真

学　　期: 2020-2021（1）

姓　　名: HandsomeHun

学　　院:

专　　业: 计算机科学与技术

班　　级:

学　　号:

指导教师:

202 年 月 25 日

计算机系制

目录

TEC-5H 模型计算机电路设计与仿真

计科 181　卢宁
指导教师　赵力

摘要：TEC-5H 是本人依据计算机组成原理所学知识，在 Proteus 仿真平台设计的八位模型计算机。TEC-5H 可以实现 TEC-5 的原有功能，在此基础上修改了数据通路和微指令，增加了立即寻址和移位指令。TEC-5H 由运算器、寄存器组、存储器、时钟信号产生器和微程序控制器等组成，可以在手动、半自动和全自动模式下自由切换。经测试，除了 74LS181 本身存在 BUG 外，TEC-5H 可以正常运行 ROM 中提前写入的程序。

关键词：TEC-5H；硬件仿真；计算机组成原理

1　系统简介

本次实验基于 Proteus 平台，结合了上学期计算机组成原理所学的相关知识，实现了 TEC-5 模型计算机的所有功能，并在此基础上修改了部分数据通路和微地址和二进制代码格式，加入了立即数寻址、移位等原先不具有的功能。我将其命名为 TEC-5H，作为计组课设的成果。

TEC-5H 是一个八位的模型计算机，支持八条 TEC-5 原生指令和五条额外添加的指令。由运算器模块、存储器模块、时钟信号产生模块和微程序控制器等模块组成，具体在第三部分 TEC-5H 模型计算机和第四部分微程序控制器描述。

遗憾的是，由于 Proteus 平台提供的关键器件 74LS181 存在一定的 BUG，在测试过程中，181 有概率在低三位产生异常的高电平，影响测试结果。除此之外，本系统所有功能都可以仿真实现。

2　指令系统

2.1 指令格式

Tec-5H 系统共有 13 条指令，具体指令格式如下图所示。

名称	助记符	功能	指令格式							
			IR7	IR6	IR5	IR4	IR3	IR2	IR1	IR0
加法	ADD Rd, Rs	Rd + Rs→Rd	0	0	0	0	Rs1	Rs0	Rd1	Rd0
减法	SUB Rd, Rs	Rd - Rs→Rd	0	0	0	1	Rs1	Rs0	Rd1	Rd0
逻辑与	AND Rd, Rs	Rd & Rs→Rd	0	0	1	0	Rs1	Rs0	Rd1	Rd0
存数	STA Rd, [Rs]	Rd→[Rs]	0	0	1	1	Rs1	Rs0	Rd1	Rd0
取数	LDA Rd, [Rs]	[Rs]→Rd	0	1	0	0	Rs1	Rs0	Rd1	Rd0
条件转移	JC R3	若 C=1 则 R3→PC	0	1	0	1	1	1	×	×
停机	STP	暂停执行	1	1	1	0	×	×	×	×
输出	OUT Rs	Rs→DBUS	0	1	1	1	Rs1	Rs0	×	×
立即数寻址	MOV Rd,X	x -> Rd	0	1	1	0	X	X	Rd1	Rd0
			八位二进制数							
循环右移	ROR Rd,X	Rd自身循环右移X位	1	0	0	0	X	X	X	X
			1	三位二进制数			X	X	Rd1	Rd0
循环左移	ROL Rd,X	Rd自身循环左移X位	1	0	0	1	X	X	X	X
			1	三位二进制数			X	X	Rd1	Rd0
逻辑右移	SHR Rd,X	Rd自身逻辑右移X位	1	0	0	0	X	X	X	X
			0	三位二进制数			X	X	Rd1	Rd0
逻辑左移	SHL Rd,X	Rd自身逻辑左移X位	1	0	0	1	X	X	X	X
			0	三位二进制数			X	X	Rd1	Rd0

2.2 指令分类

2.2.1 算术逻辑运算指令

包含加法(ADD)、减法(SUB)、逻辑与(AND)、循环右移(ROR)、循环左移(ROL)、逻辑右移(SHR)、逻辑左移(SHL)七条指令。其中移位指令长度为两字节，其余为一字节。

2.2.2 数据传输类指令

包含存数(STA)、取数(LDA)、立即寻址(MOV)三条指令。其中立即寻址指令长度为两字节，其余为一字节。

2.2.3 程序控制类指令

包含条件转移(JC)、停机(STP)两条指令。

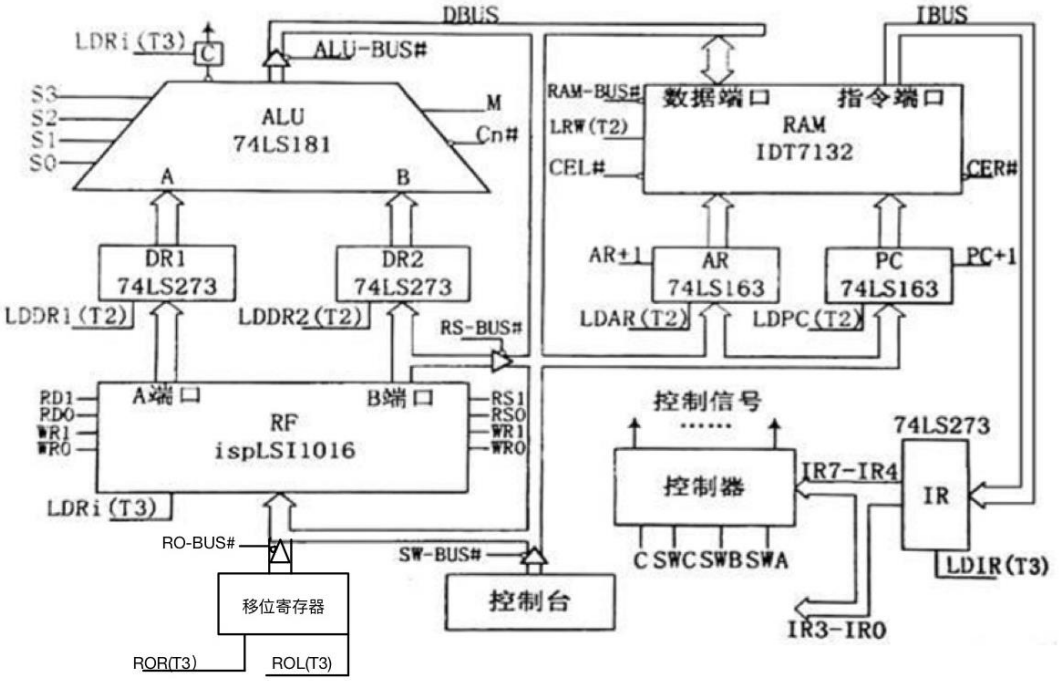
2.2.4 系统指令

包含输出(OUT)一条指令。

3 TEC-5H 模型计算机设计

3.1 总体设计

TEC-5H 的总体设计参照了 TEC-5，并在其基础上修改了双端口存储器的构造，增加了有关立即寻址和移位有关的控制器。总体上，TEC-5H 的数据通路设计电路可以划分成运算模块、通用寄存器模块、控制台模块、存储器模块、指令寄存器模块、时钟信号产生模块、手动控制模块和微程序控制器模块等。每个模块都执行相应的功能、通过线路和时钟信号传递信息。TEC-5H 的大致数据通路见下图。

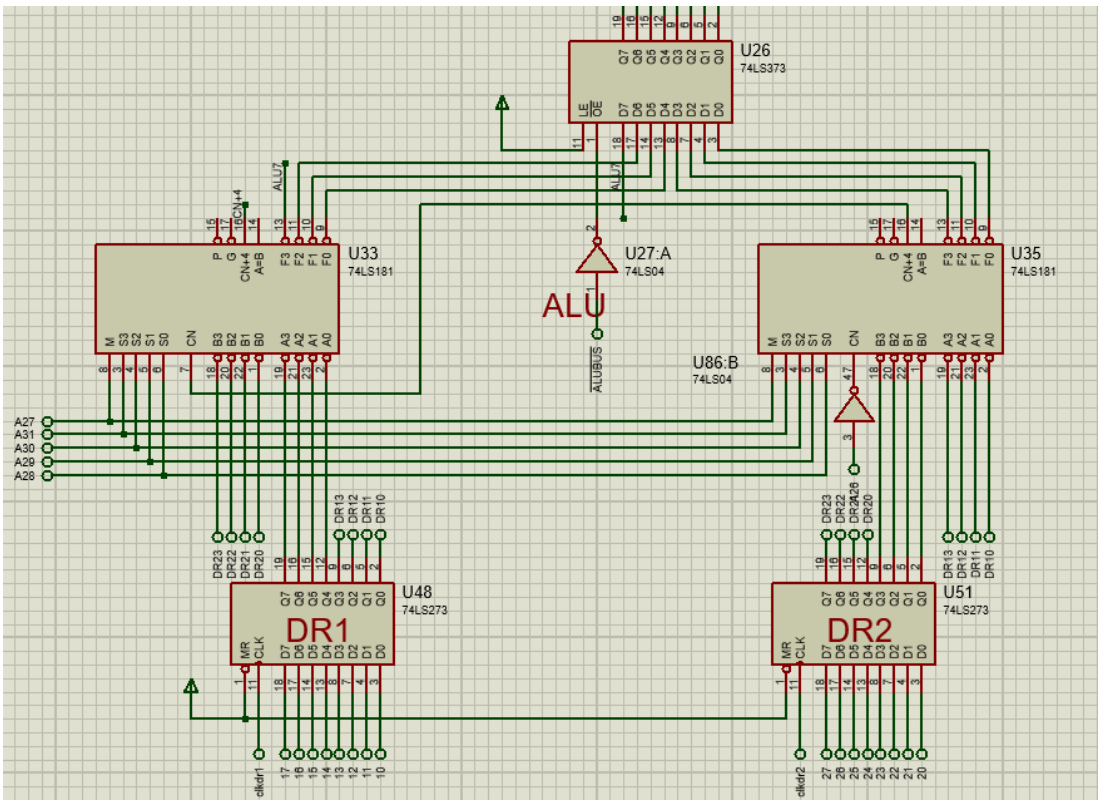


3.2 运算模块

运算模块主要由运算器和移位器组成。

3.2.1 运算器

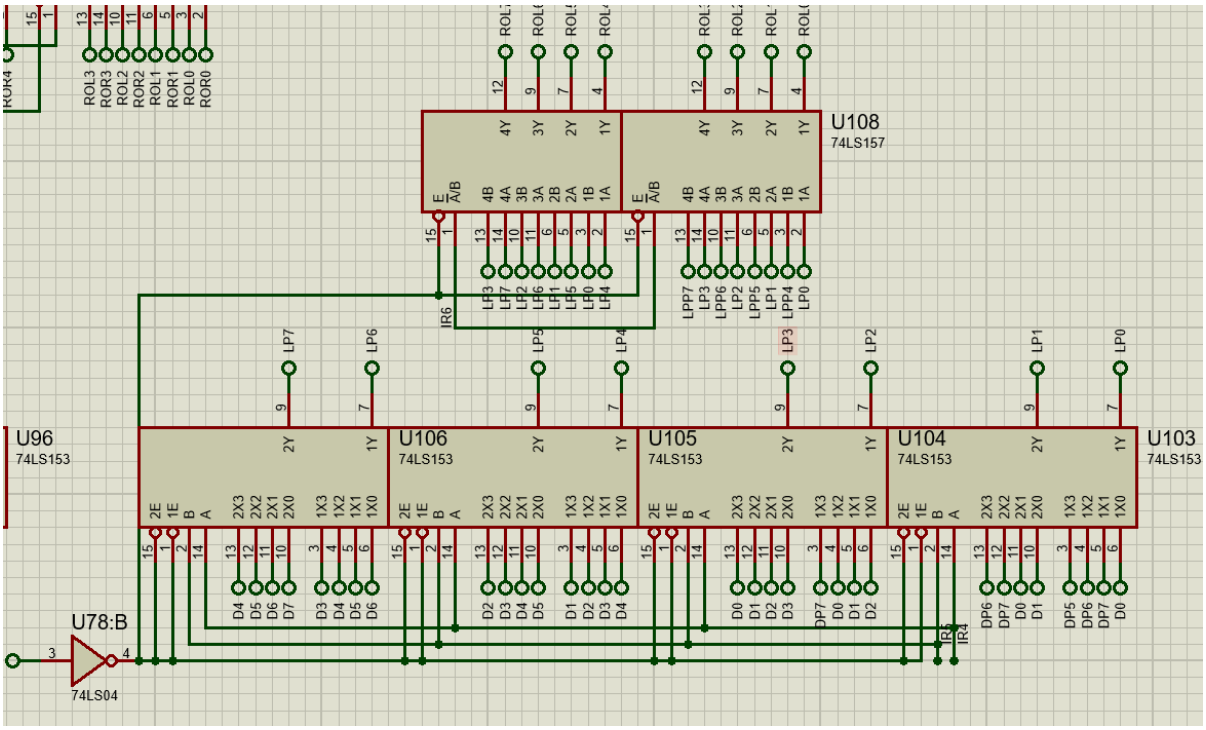
TEC-5H 为八位模型计算机，因此运算器的 ALU 由两片 74LS181 连接。其中 ALU 的数据输入端连接两片 74LS273，构成数据缓冲寄存器（DR），其功能是选择通用寄存器，在 LDDR 时钟信号的引导下适时进入 ALU 参与运算。ALU 的输出端连接一个 74LS373，构成八位的缓冲器，由 ALU-BUS 控制其是否将运算结果输出到数据总线上。



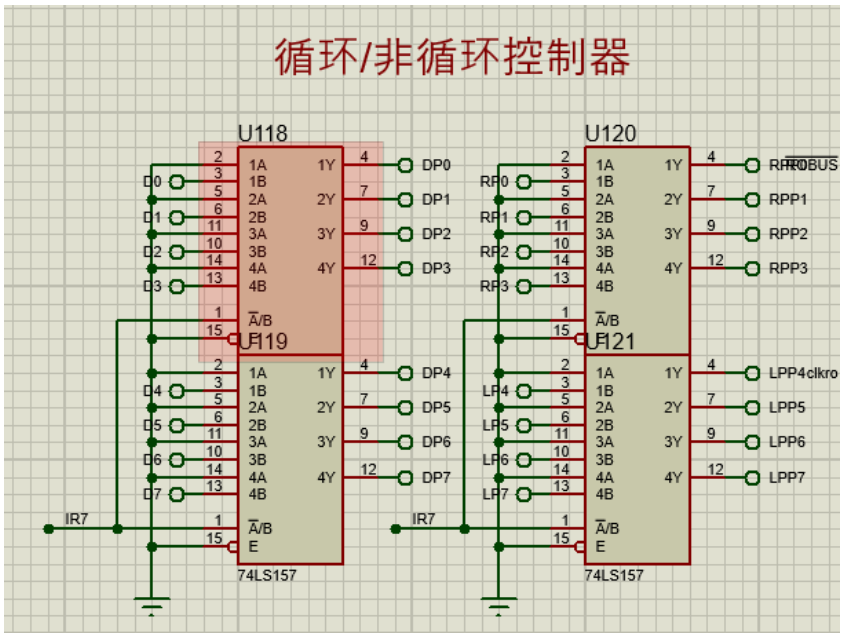
3.2.2 移位器

相较于运算器，移位器注重于实现移位功能，且没有现成的器件，因此实现电路较为复杂。TEC-5H 的移位器借鉴了桶型移位器的思路，但又有所不同，由大量的选择器和一片 74LS245,一片 74LS273 组成。为了实现左移和右移，TEC-5H 采用了两层的移位方式，并用两套类似的结构分别实现左移和右移。下以左移控制器为例介绍二层移位法

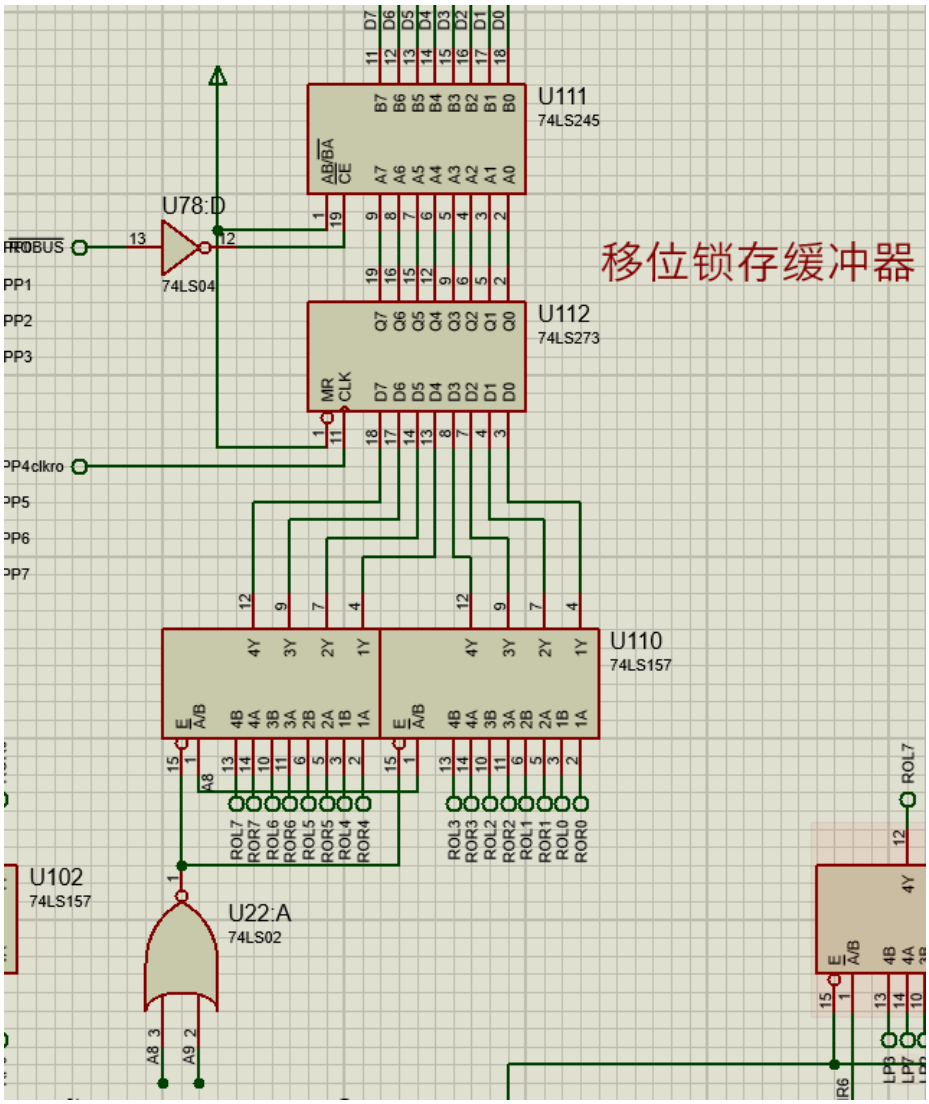
左移控制器中第一层由 4 片 74LS153 组成，74LS153 为四选一选择器。每个输出端对应四个输入端，分别为其本身、其低 1 位、低 2 位和低 3 位。具体输出哪一个输入端由 IR5,IR4（B,A）控制，以此实现左移 0 到 3 位。第二层由 2 个 74LS157 组成，75LS157 集成了 4 个二选一选择器。第一层输出端的结果可以直接连接第二层的输入端，与此同时，另一个输入端连接输出端的低 4 位，实现左移 0 或 4 位。两层结合起来，就可以实现 0 到 7 位的移位。



为了区别循环和非循环移位，TE-5H 额外采用了 4 片 74LS157 来控制是否循环。原始的输入端与低电平进行一个二选一，由 IR7 控制。非循环模式下输出端输出低电平，循环模式下输出端输出原信号。输出端有选择地接在左移和右移控制器的涉及移位后会循环的输入端，比如左移控制器第一层输入端最低位所对应的三个需要移位的输入端。在循环模式下，这三个输入端分别对应 D7，D6，D5,而在非循环模式下，对应三个低电平，即补零。同理，第二层的部分输入端也需要作类似的处理，具体参照左移控制器和循环控制器的电路图。

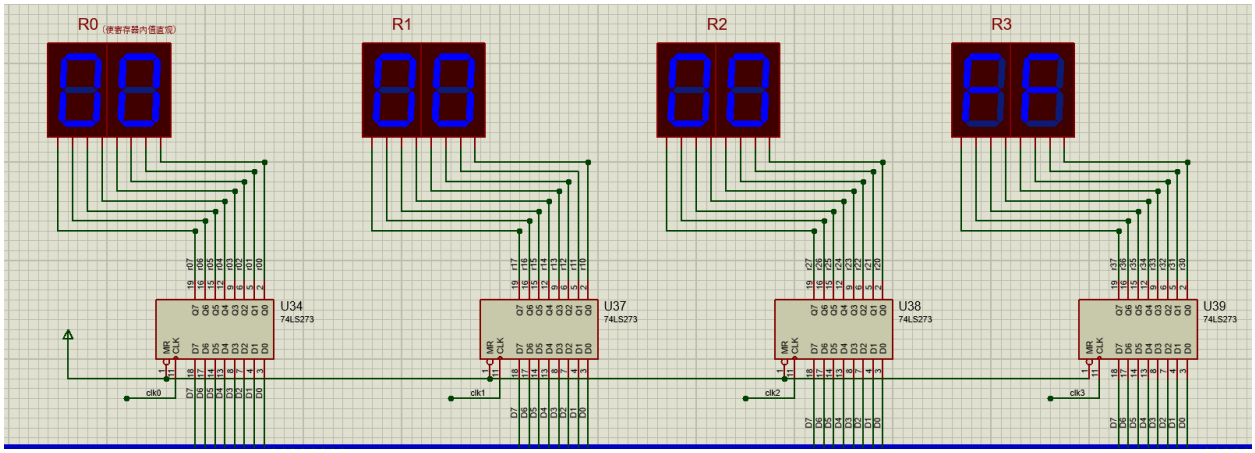


移位控制器的输出部分由 2 片 74LS157 和 74LS273、74LS245 组成。其中 74LS157 负责选择要输出的是左移控制器还是右移控制器。74LS273 在 clkro 的控制下锁存移位的结果，并在下一个时钟周期打开 RO-BUS，通过 74LS245 输出到数据总线。

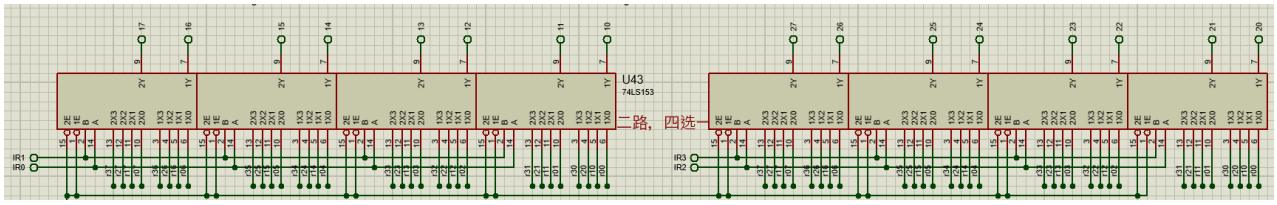


3.3 通用寄存器模块

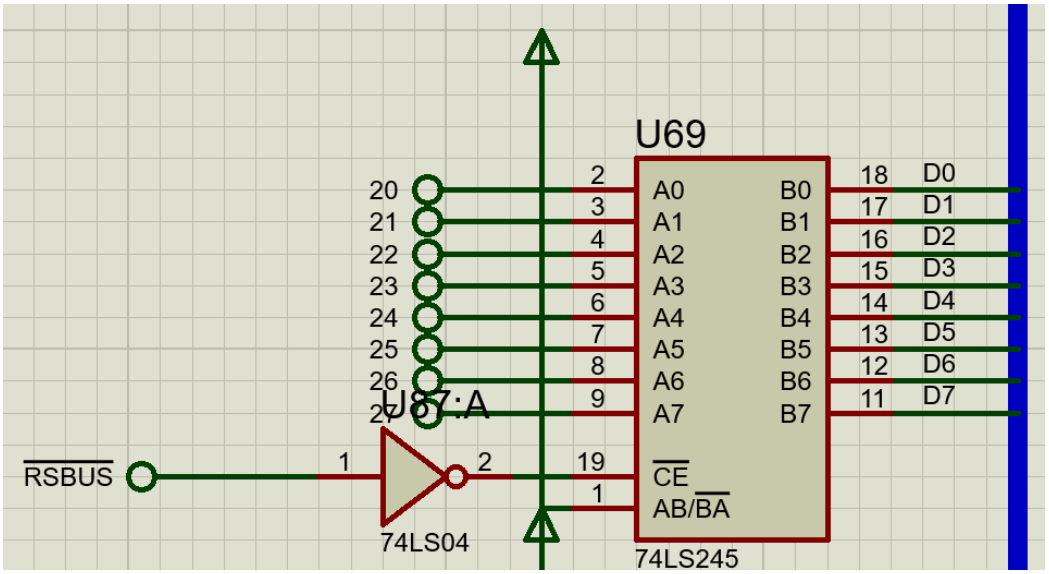
通用寄存器模块主要由 4 片 74LS273 和 8 片 74LS153 组成。其中 74LS273 在 LDRI 的控制下，根据 IR0 和 IR1 将 DBUS 上的数据锁存起来，放到 R0 至 R3 寄存器。



74LS153 为四选一选择器，负责根据 IR0 至 IR3 的信号从 R0 至 R3 选择数据。其中 Rd 由 IR0 和 IR1 控制，选择出来的数据直通运算器的 DR1。Rs 连接 IR2 和 IR3，选择出来的数据既可以通往运算器的 DR2 参与运算。

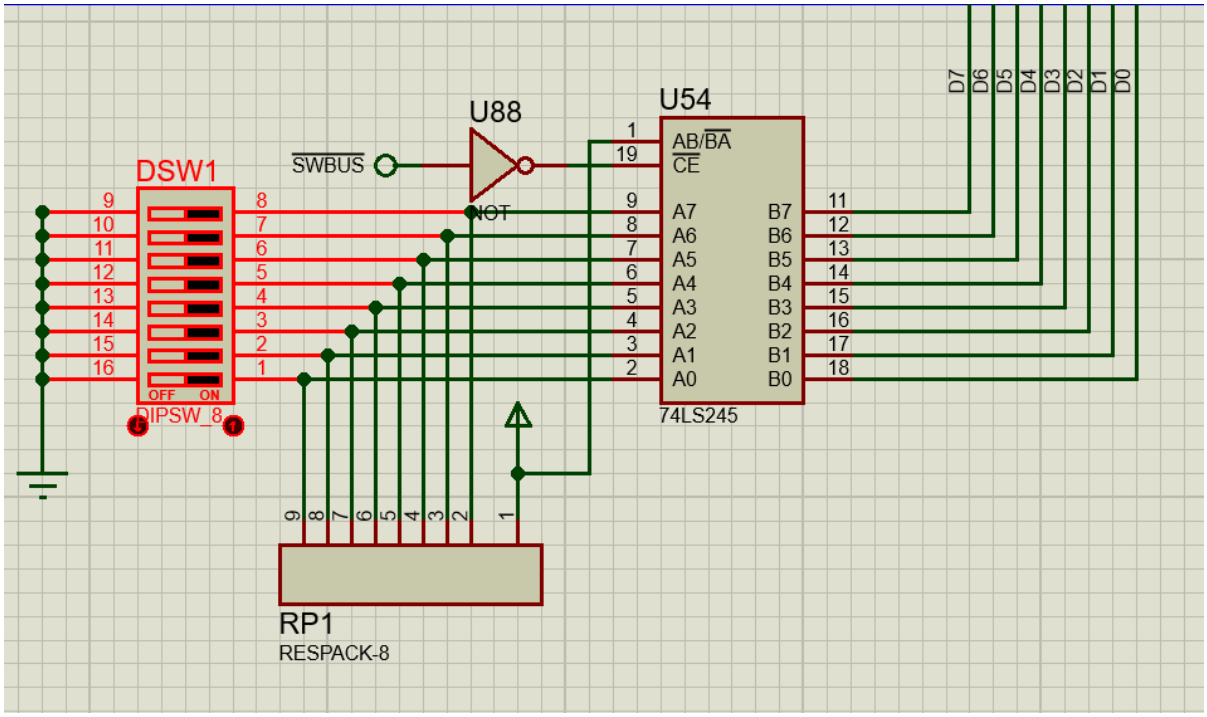


此外，在一片 74LS245 的控制下，寄存器的数据也可以在 RS-BUS 的控制下进入数据总线。



3.4 控制台模块

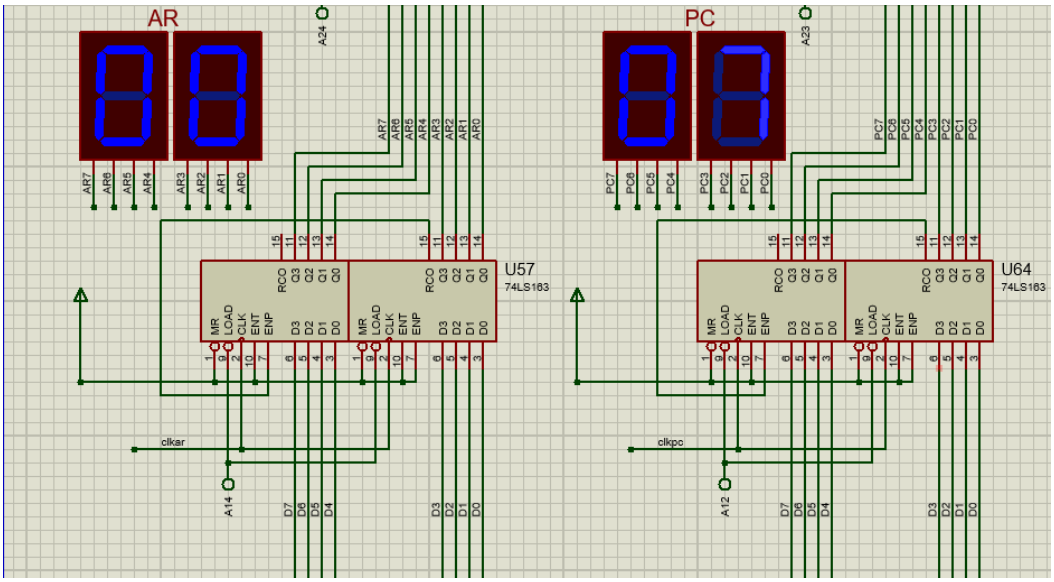
控制的设计比较简单。DIPSW_8 可以通过点击的方式产生一个 8 位数据，再通过一个 74LS245，在 SW-BUS 信号控制下将数据输入到数据总线。



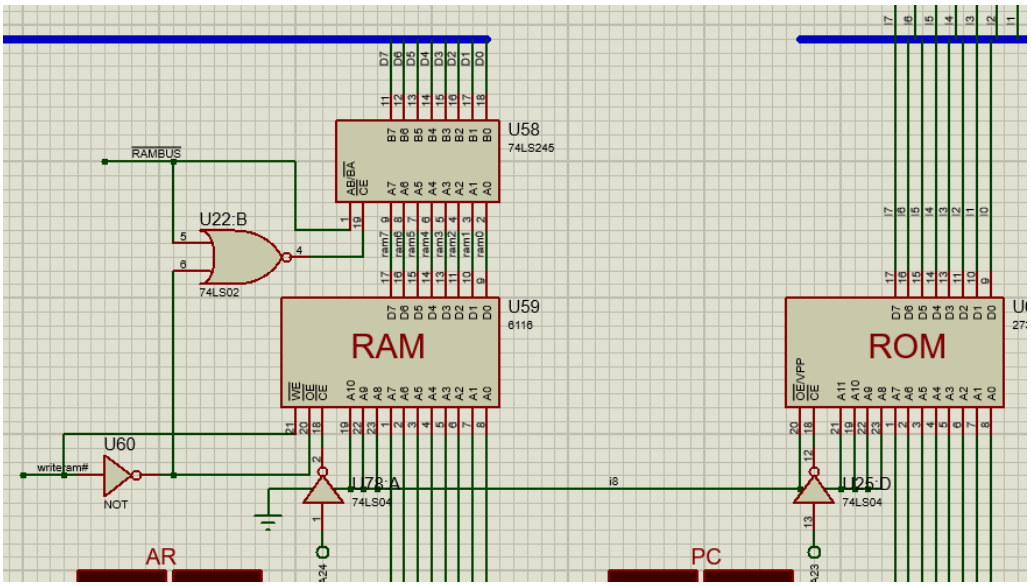
3.5 存储器模块

TEC-5 的存储器为双端口存储器，但是 Proteus 平台并未提供双端口存储器，因此 TEC-5H 使用 6116 来实现 RAM 的功能，2732 实现 ROM 的功能。除此之外，存储器模块还包括地址寄存器 AR 和 PC 以及为了实现立即寻址和写寄存器附加的部分通路。

在地址寄存器上，AR 和 PC 设计一致。以 AR 为例，其由两片 14LS163 组成。74LS163 是同步计数器，在开启状态下，每个 clkar 都会对 AR 寄存器产生影响。在 clkar 来临时如果 Load 是低电平，那么 DBUS 上的数据就会被读取，反之 AR 会在原数的基础上加 1。



AR 的输入端直接连接 6116 的低 8 位，如果 RAM 的总开关 CEL 打开，在默认的读状态下，读取使能端被打开，会将 RAM 中 AR 对应的地址上的数据读出，传入 74LS245。如果 RAM-BUS 有效，数据将被送往数据总线。在特定开启的写状态下，RAM-BUS 必须是关闭的，此时数据会从总线传入 74LS245，RAM 的写允许也会被打开，数据会写入 RAM 中 AR 对应的地址。



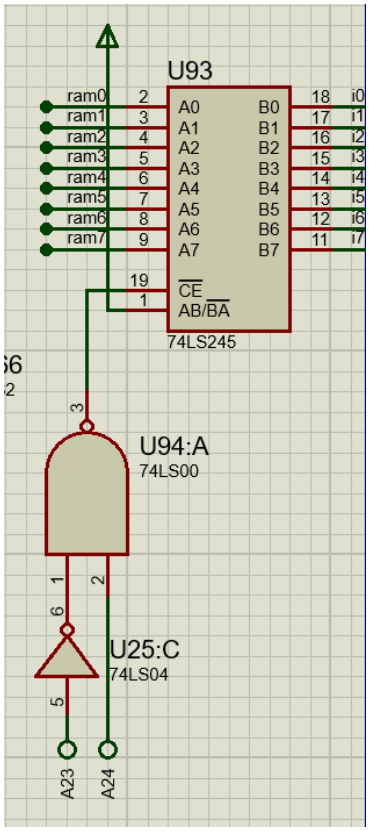
ROM 上的内容在 Proteus 运行时不可修改，需要提前烧写，烧写方式会在微程序控制器部分详细说明。PC 的输入端直连 2732 的低 8 位，如果 ROM 的总开关 CER 打开，

那么 ROM 的数据会被直接传入 IBUS。ROM 内部数据如图所示

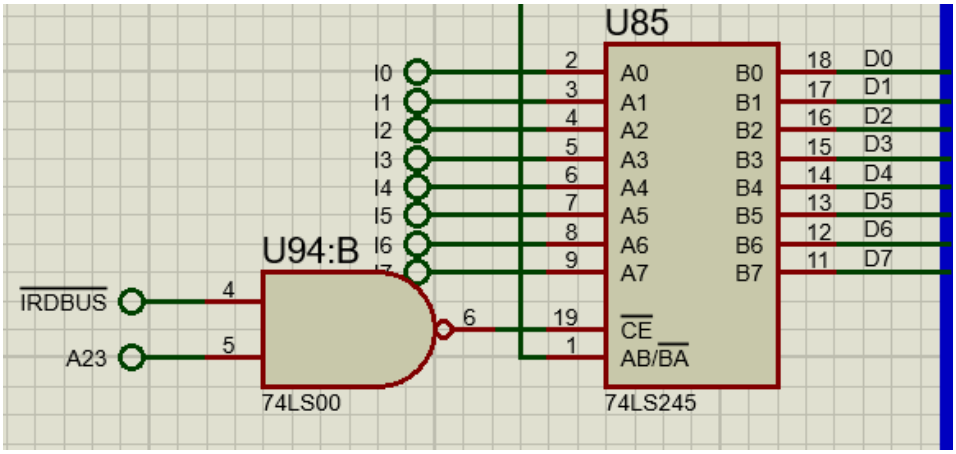
```
ORG 00H
    DB 60H ;MOV R0,13H
    DB 13H
    DB 61H
    DB 39H ;MOV R1,39H
    DB 04H ;ADD R0,R1
ORG 05H
```

```
0000 | 60 13 61 39 | 0
0038 | 00 00 00 00 | 0
```

为了满足 SWC， SWB， SWA 为 011 状态下写寄存器的要求，TEC-5H 添加了 RAM 输出至 IBUS 的命令。为了节省一个微命令，所控制的 74LS245 仅在 CER 无效且 CEL 有效的情况下将 RAM 上的输出端输出到 IBUS 上。

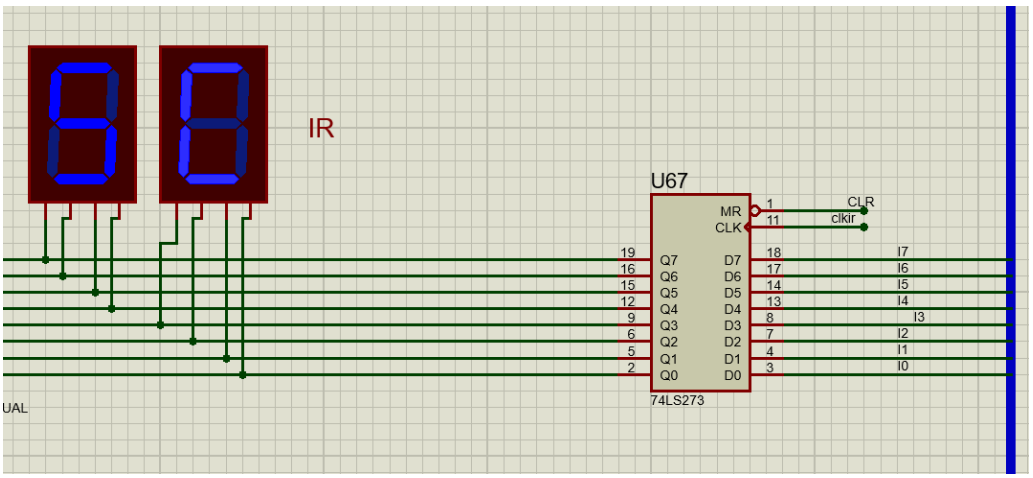


同时为了满足立即寻址的需要，添加了 IBUS 信号传输到 DBUS 的缓冲器。在 IRDBUS 和 CER 同时有效时， 会将 IBUS 上的内容输出到 DBUS。



3.6 指令寄存器模块

指令寄存器由一个 74LS273 组成，在 clkir 的控制下，将 IBUS 上的指令锁存到 IR 寄存器。

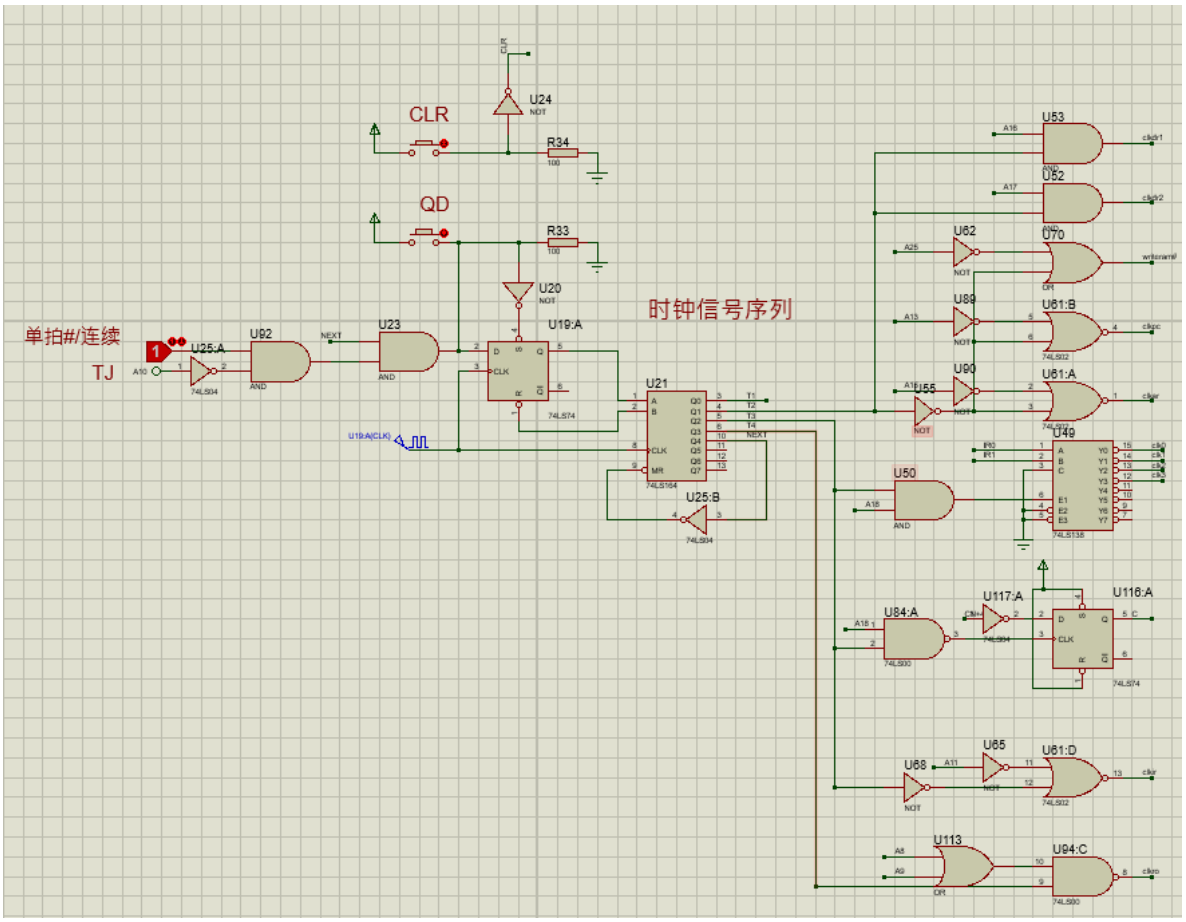


3.7 时钟信号序列模块

时钟信号序列模块主要由两部分组成。一部分为 CLR，按下按钮使得 CLR 生效，产生一个脉冲将微存地址设为 0，即重启程序。另一部分发挥了重要的作用，控制整个电路中所有需要时钟信号的部件，按照固定的顺序使他们依次工作。

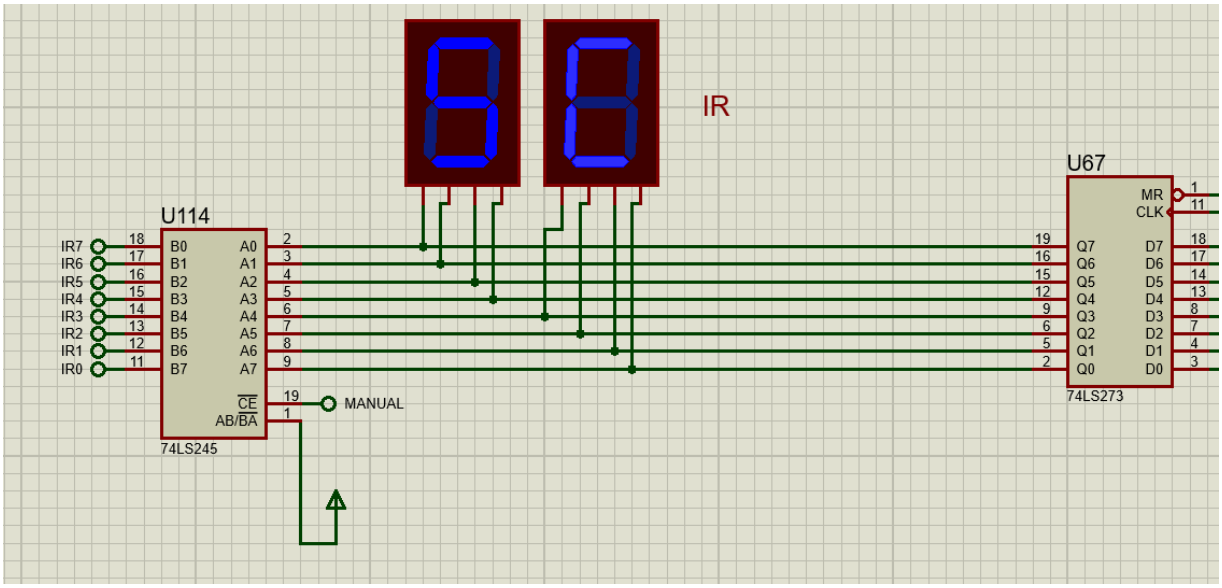
起到关键作用的是 74LS164。74LS164 是边沿触发的移位寄存器。当按下 QD 时，信号通过 74LS74（D 触发器）送到 74LS164 的 A 端口，使得输出端在每个 clk（频率为 3HZ）下都产生依次移位，起到了依次触发 T1，T2，T3 T4，NEXT 的效果。NEXT 信号连接在 164 的 MR 端，使得 164 的输出清零。在 TJ 指令无效且设置为连续情况下，会再触发 D 触发器，使得 164 进行新一循环的工作，其他情况下停止工作。

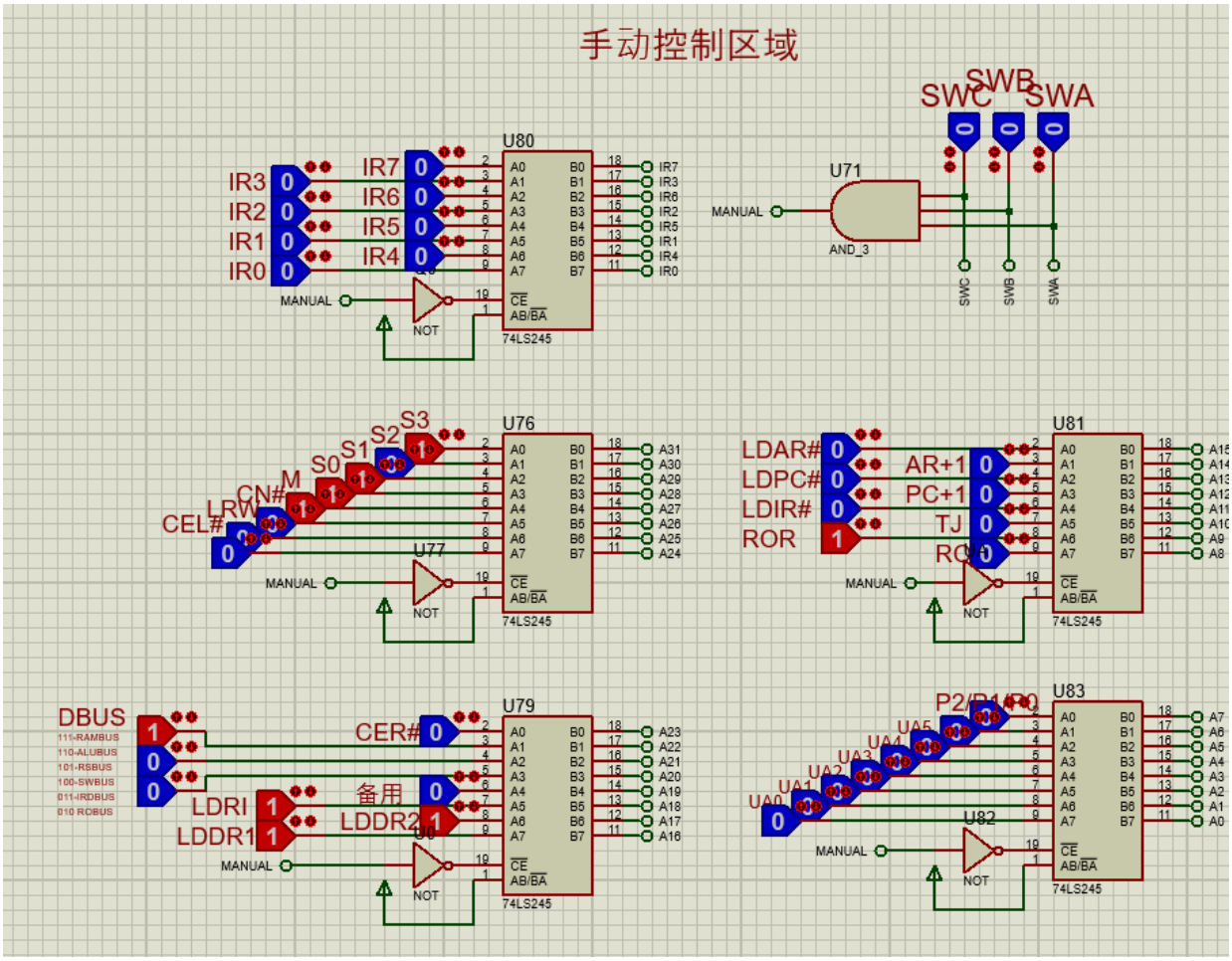
在 T1，T2，T3，T4 依次触发时，它们会与微命令通过与或非等逻辑门一起产生相关的时钟脉冲，控制指定部件的运行，具体信号见 3.1 总体设计。



3.8 手动控制模块

手动控制部分主要由 74LS245 和 LOGICSTATE 组成。LOGICSTATE 可以选择状态为 0 或者 1。当 SWC, SWB, SWA 为 111 时，MANUAL 信号生效，所有的微命令都会进入手动控制状态。此时所有的 74LS245 开始工作，微程序产生的微命令会被屏蔽，所有微命令和 IR 指令都来自于 LOGICSTATE。

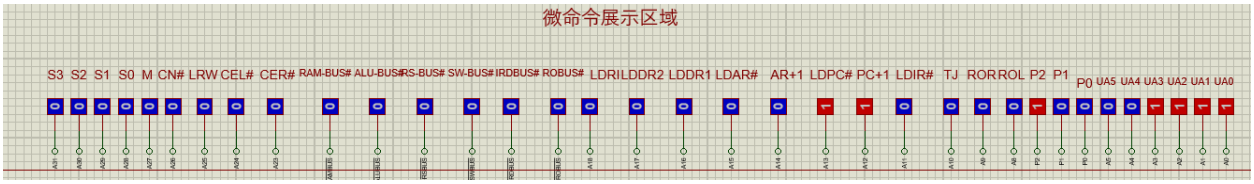




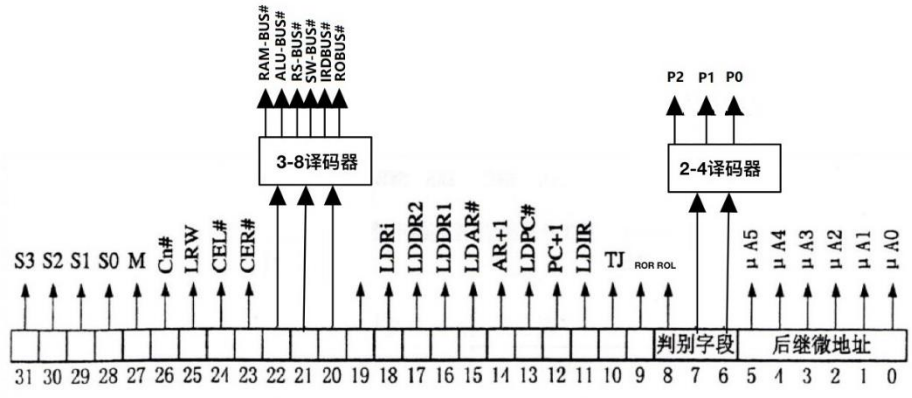
4 微程序控制器

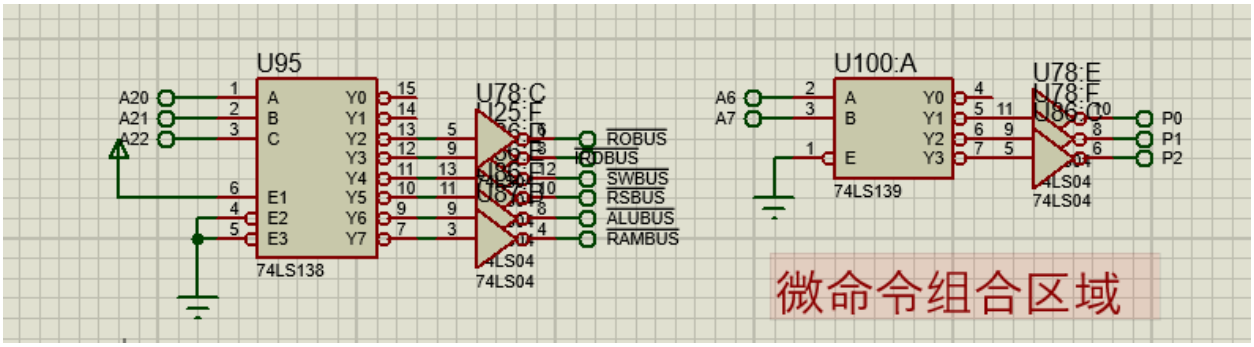
4.1 微指令和微命令

在非纯手动控制下，微命令由微程序控制器发出，与时钟控制序列一起控制其他器件的微操作。在 TEC-5H 中，所有微指令在无效时都为 0，在生效时都为 1，产生的控制信号均进过其他逻辑部件处理后作用到高电平有效或是低电平有效的器件上。



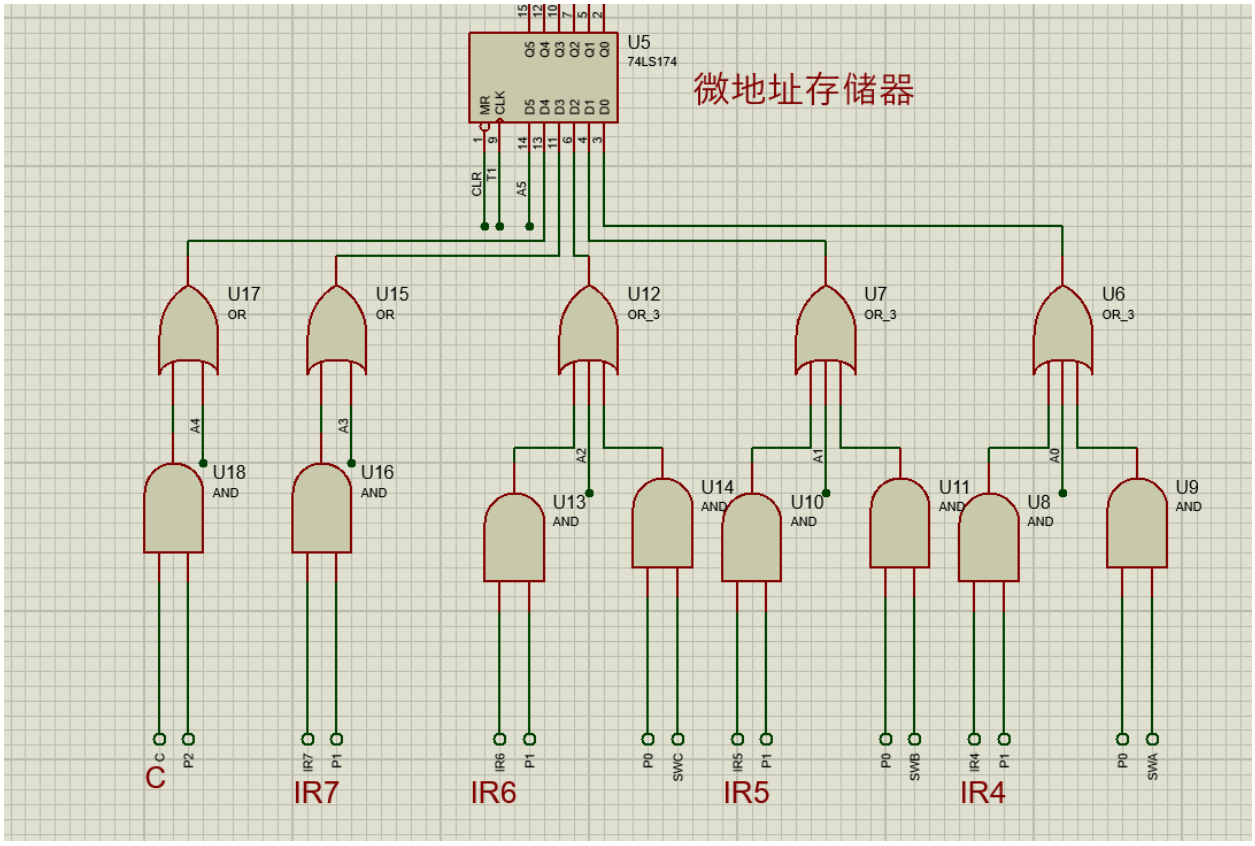
微指令是实现特定功能的微命令的组合，由操作控制和顺序控制两大部分组成。TEC-5H 的微指令为 32 位，其中操作控制 24 位，顺序控制 8 位。操作控制部分主要负责发出微命令控制其他器件的工作，顺序控制部分包含转移指令和后继微地址，负责确定下一条要运行的微指令的地址。微指令的编码采取了混合表示法，对于一些互斥的微命令，如涉及到输出至 DBUS 的微命令，借助 3-8 译码器或者 2-4 译码器进行编码，减少了所需空间。TEC-5H 一共有 36 条微命令，其格式见下二图。





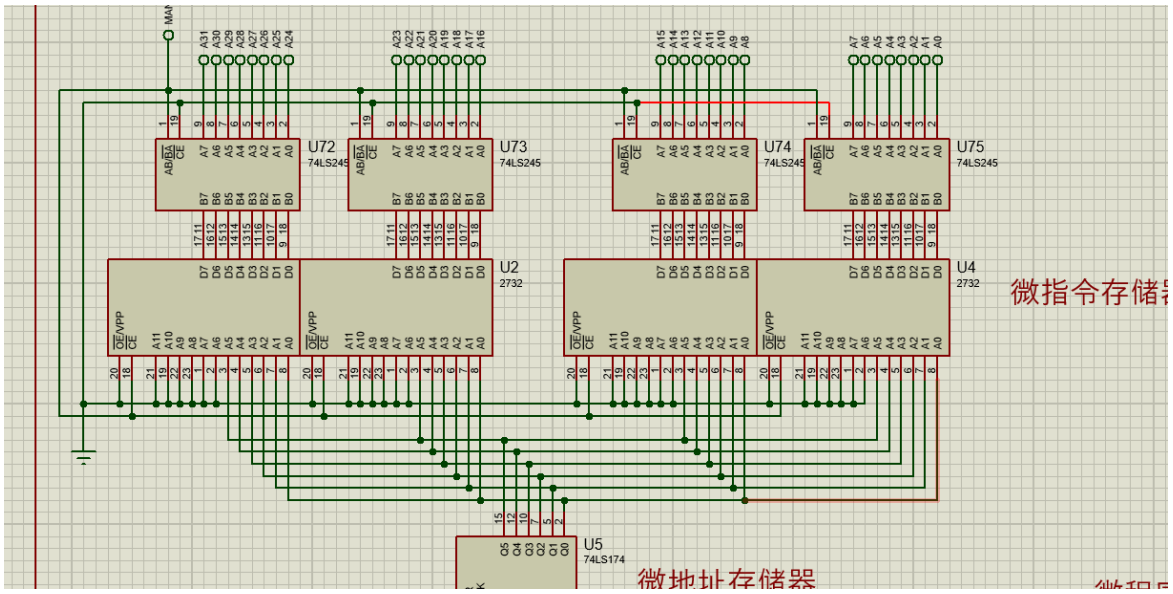
4.2 微地址存储器

微地址存储器由 74LS174 负责暂存正在执行指令的地址，并根据微指令的顺序控制字段和输入端部分的逻辑电路，给出下一条要执行微指令的地址。在每个 T1 来临时，将下一条的地址从输入端置入到输出端。同时手动产生的 CLR 信号会将微地址存储器的输出清零。



4.3 微指令存储器

微指令存储器主要由 4 个 2732ROM 组成，输入端为微地址存储器的输出端。在非纯手动控制地情况下工作，将输出端输出到对应的 74LS245 缓冲器保存。微指令存储器保存了所有的需要的微地址上的微指令，需要提前烧写。



4.4 ROM 烧写

指令烧写我采用 python 编程的方式。在 asm 文件上写好微指令内容后，我采用正则表达式提取地址信息和数据信息，建立一个地址和数据一一对应的字典。之后从字典中提取信息，将数据根据 hex 的规则，计算出每一行的校验码，按格式烧写成.hex 文件。这样的好处在于修改微指令内容后只需要运行程序就可以直接烧写并覆盖 4 个.hex 文件，极大地提升了我增添和修改微指令的速度。以下为部分程序代码截图。

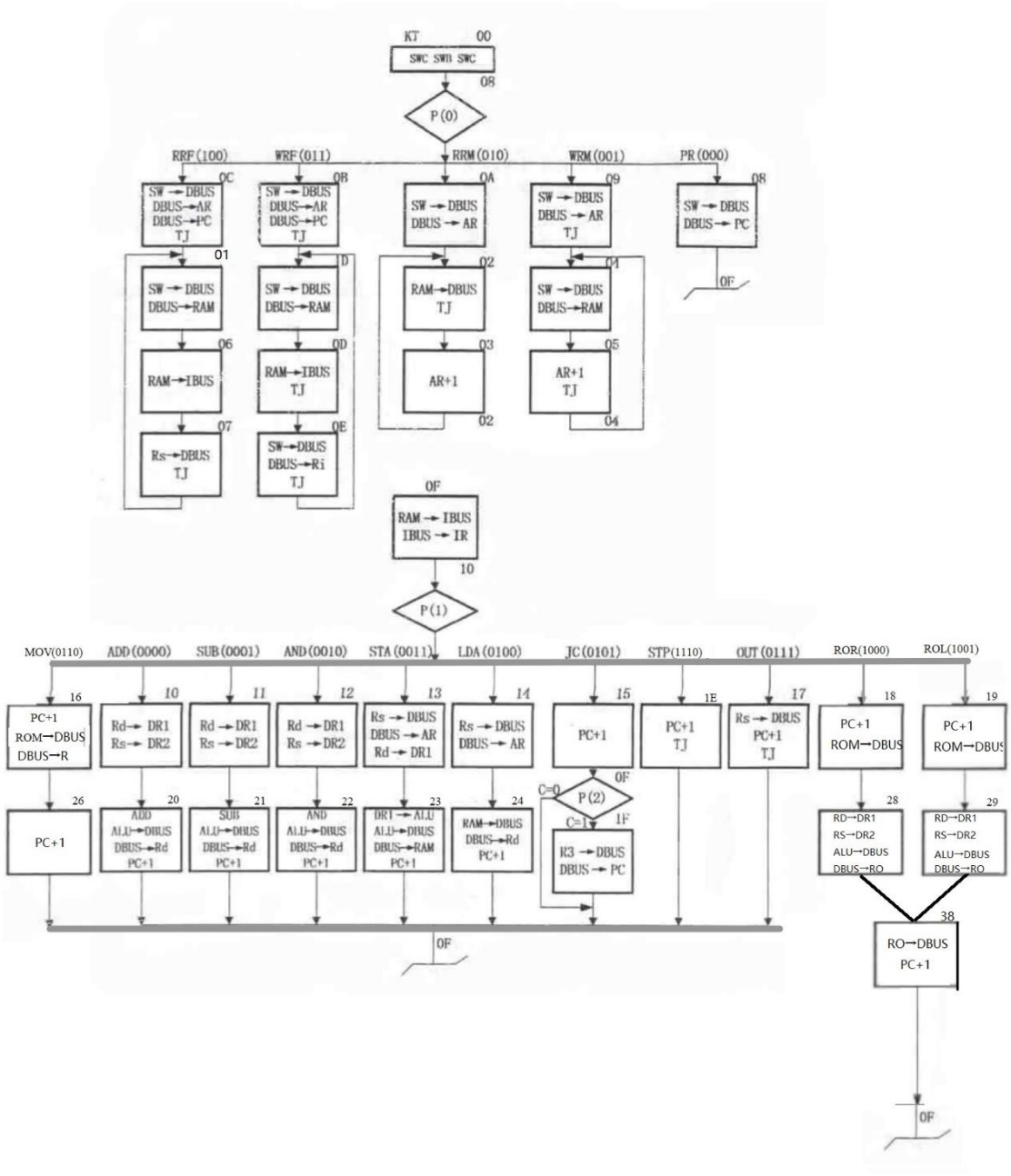
```
with open("CM1.asm") as f2:
    list2 = f2.read()
    list2_key=re.findall(r"(\S+)H",list2)
    list2_value=re.findall(r"(\d\d+)B",list2)
    for key,value in zip(list2_key,list2_value):
        if key in asmdict:
            asmdict[key]+=value
    f2.close()
```

```
with open("0.hex",'w') as t:
    for asm in asmhub:
        check = 1 + list(bytearray.fromhex(asm[0]))[0]+int('0b'+asm[1][0:8],2)
        dd = 0x100-check%0x100
        hexline = ":0100"+asm[0]+"00"+str('{:02X}'.format(int('0b'+asm[1][0:8],2)))+str('{:02X}'.format(dd))
        t.writelines(hexline)
        t.writelines('\n')
    t.writelines(":00000001FF\n")
    t.close()
```

4.5 微程序流程图

微程序流程图包含控制台流程图和机器指令微程序流程图。TEC-5H 在 TEC-5 的基

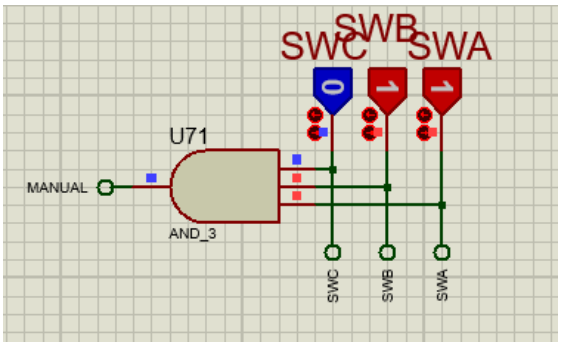
础上改变了 STP 的指令格式和所有指令的第二个微地址，具体详见下图。



5 测试

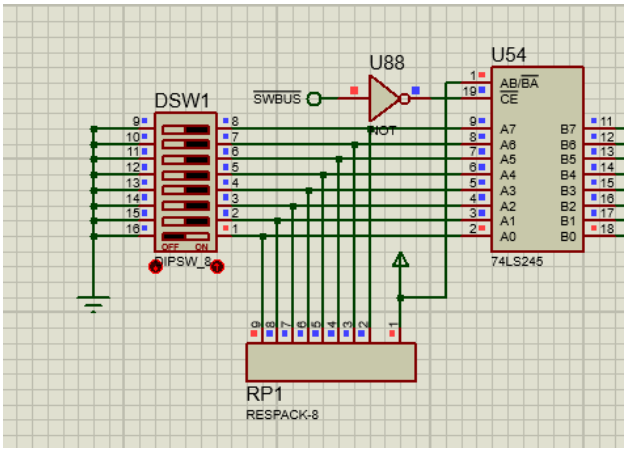
5.1 写寄存器

先将工作模式调为 011，按下 QD。

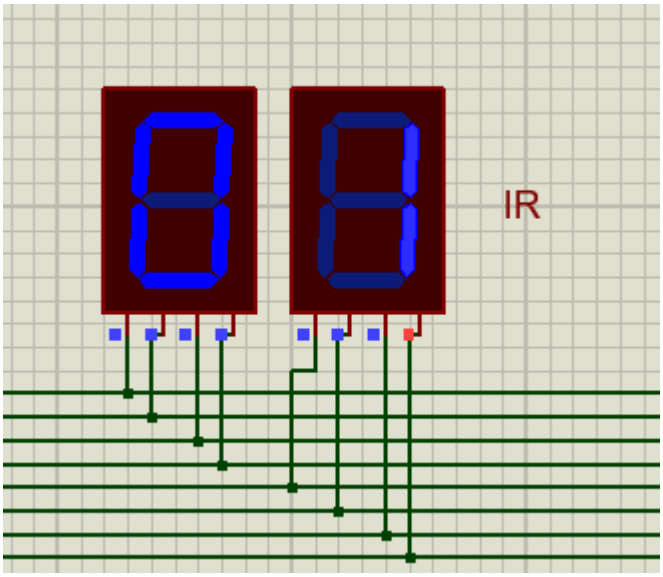


再向 SW 控制台输入任意数字指定途经 RAM 的地址，按下 QD。

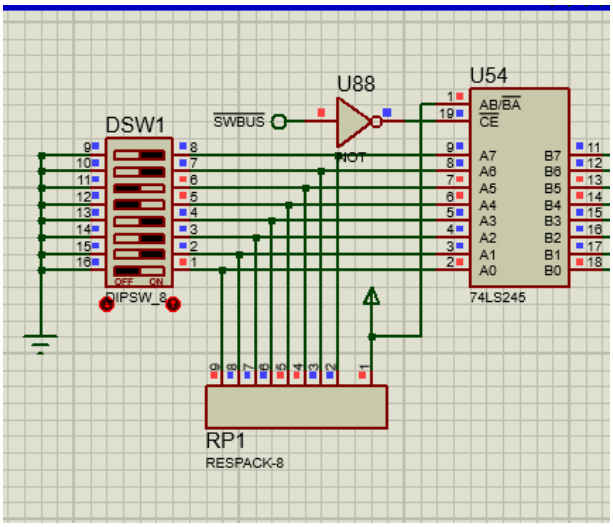
再向 SW 控制台输入数据，低两位控制要输入的寄存器，如 01，按下 QD。



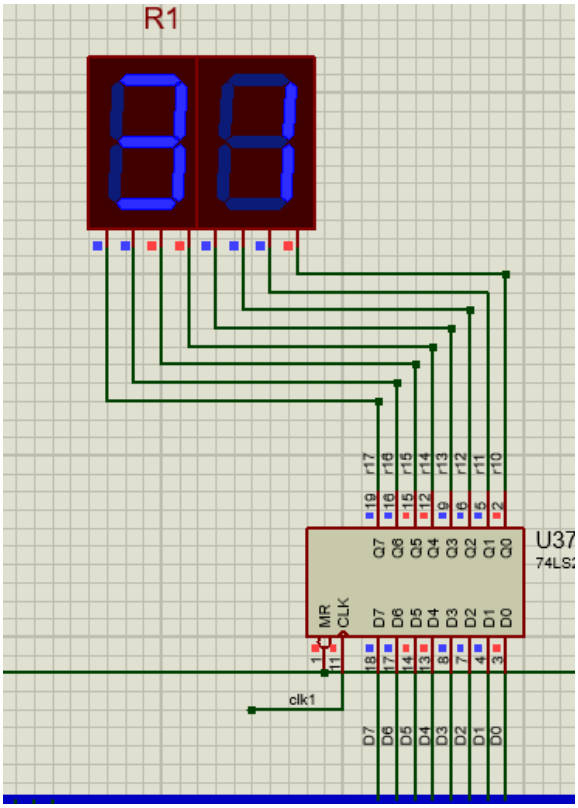
可以看到 IR 发生变化。



向 SW 控制台输入要存在寄存器的数据，按下 QD。

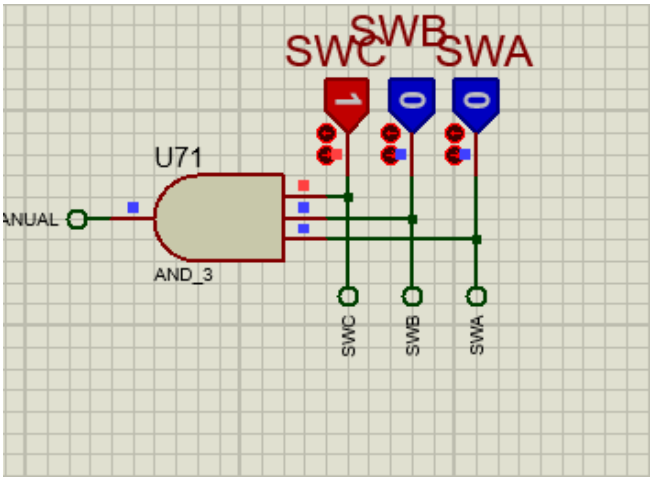


最后就可以看到数据 31H 成功写入寄存器。



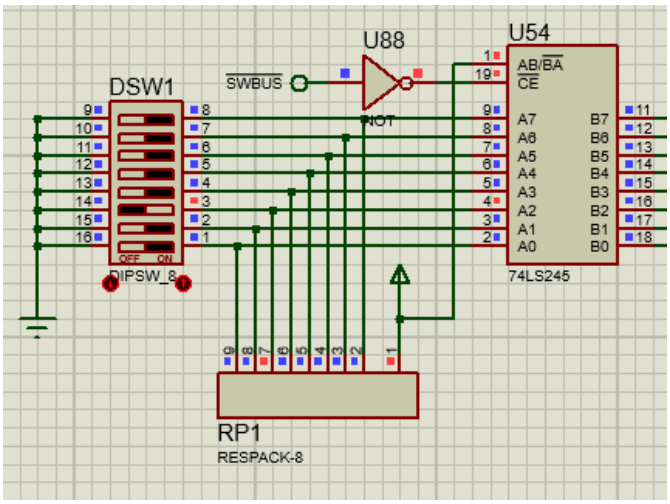
5.2　读寄存器

将工作模式设为 100。

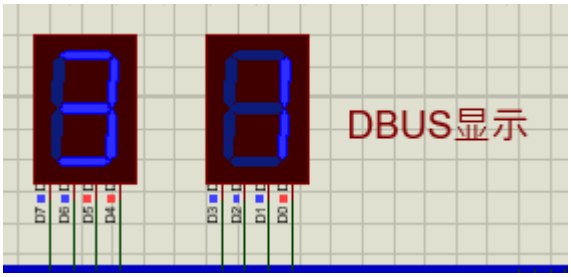


再向 SW 控制台输入任意数字指定途经 RAM 的地址，按下 QD。

再向 SW 控制台输入数据，其中 IR2 和 IR3 控制要输入的寄存器，如 04，按下 QD。

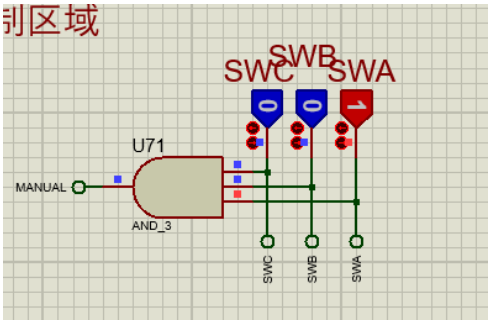


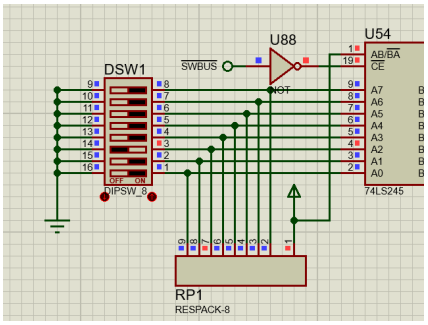
就可以看到在上一步写入的 31H 成功读至 DBUS。



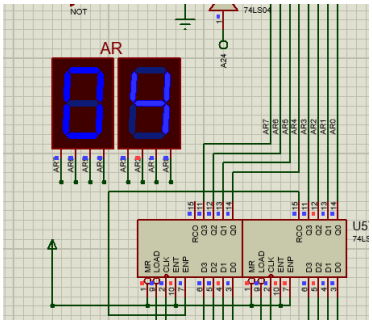
5.3 写存储器

将工作模式设为 001，向控制台输入要存储的地址，如 04H，按下 QD。

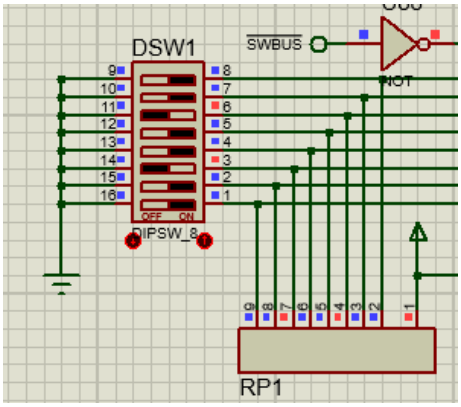




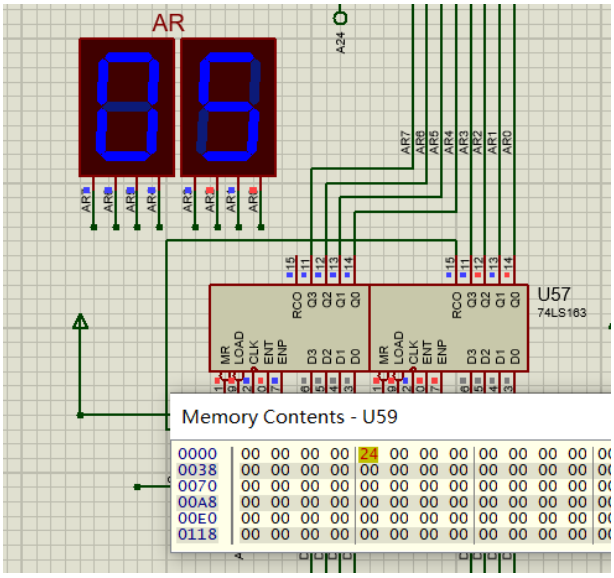
可以看到 04H 已经读入 AR。



继续向控制台输入要存储的数据，如 24H，按下 QD。

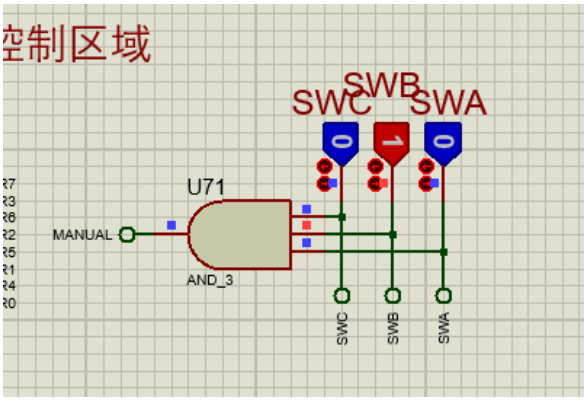


可以看到数据已存入存储器对应位置，且 AR 指向下一个地址。

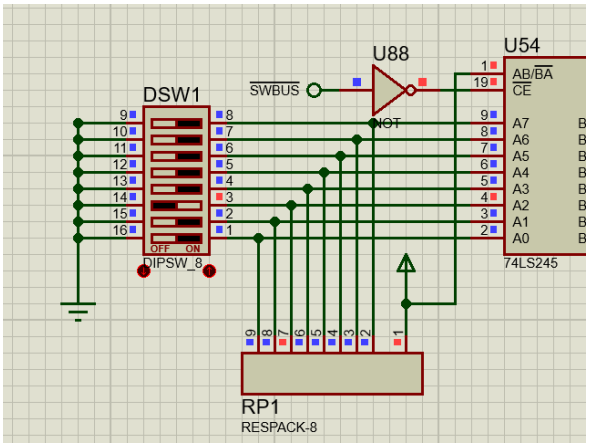


5.4 读存储器

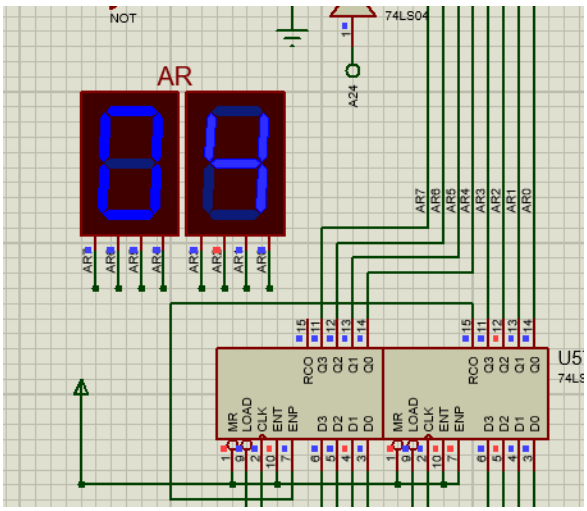
将工作模式设为 010。



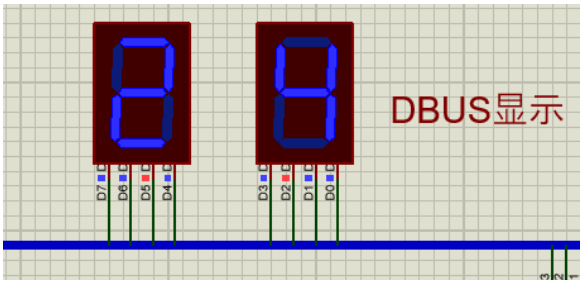
从控制台输入要读取存储器的地址，如 04H，按下 QD。



可以看到数据已输入 AR。

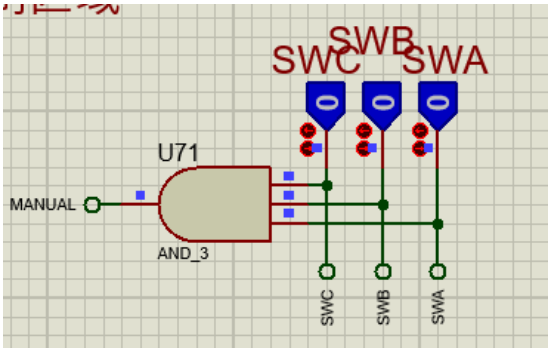


再按下 QD，可以看到数据从 RAM 读入 DBUS。



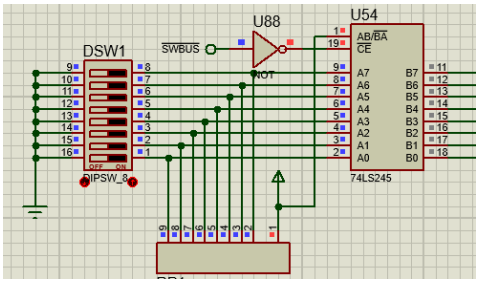
5.5 MOV 指令

将 SWC，SWB，SWA 改为 000。

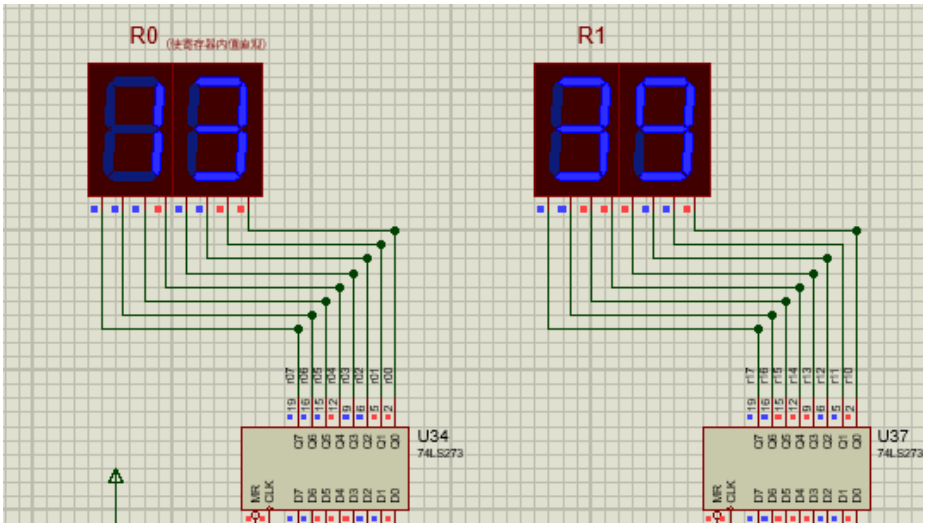


向 ROM 写入 MOV 指令，地址从 00H 开始，从控制台输入起始地址 00H。

```
DB 60H ;MOV R0,13H
DB 13H
DB 61H
DB 39H ;MOV R1,39H
```



按下 QD，两个循环后可以看到值已写入寄存器。

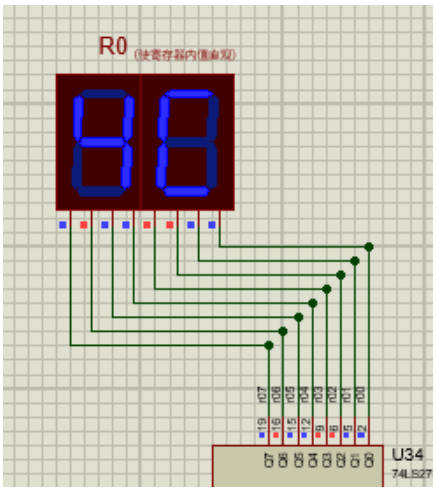


5.6 ADD 指令

提前向 ROM 输入测试指令，如下图。

```
DB 04H ;ADD R0,R1
```

接着上一步的 MOV 指令，按下 QD 可以看到相加成功。

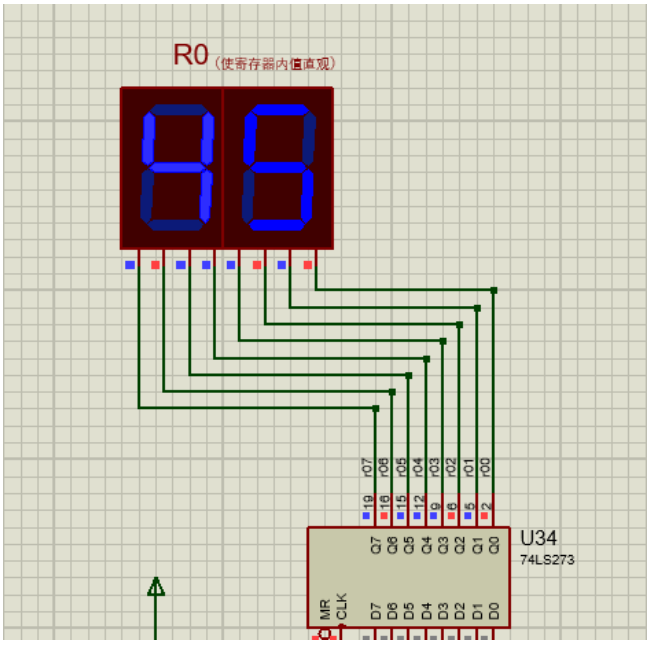


5.7 SUB 指令

提前向 ROM 输入测试指令，如下图。

```
DB 60H ;MOV R0,89H
DB 89H
DB 61H ;MOV R1,44H
DB 44H
DB 14H ;SUB R0,R1
```

按下 QD 即可看到相减成功并存入 R1。

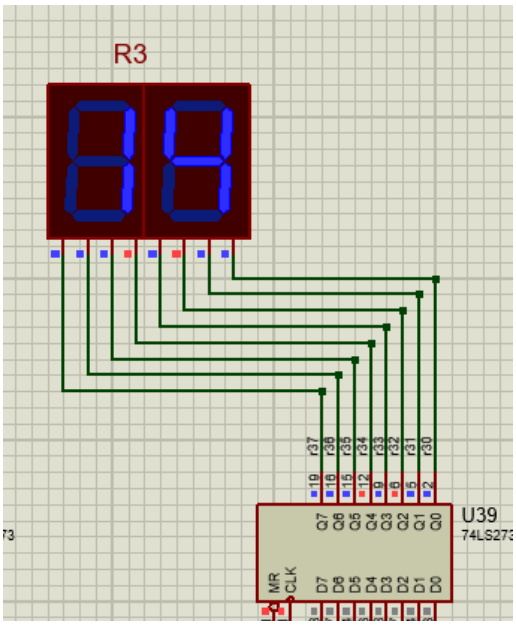


5.8 AND 指令

提前向 ROM 输入测试指令，如下图。

```
DB 60H ;MOV R0,56H
DB 56H
DB 63H ;MOV R3,94H
DB 94H
DB 23H ;AND R3,R0
```

按下 QD，连续几个节拍后可以看到与的结果存入 R3。

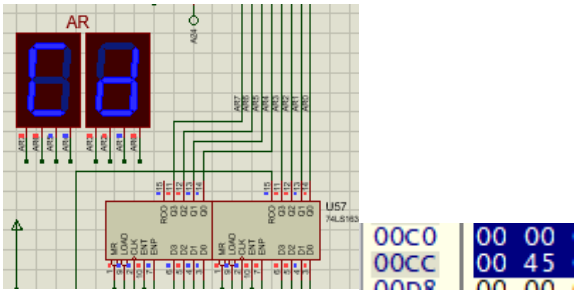


5.9 STA 指令

提前向 ROM 输入测试指令，如下图。

```
DB 60H ;MOV R0,45H
DB 45H
DB 61H ;MOV R1,CDH
DB 0CDH
DB 34H ;R0->[R1]
```

按下 QD 可看到 45H 已存入地址 0CDH。



5.10 LDA 指令

提前向 ROM 输入测试指令，如下图。

```
DB 60H ;MOV R0,45H
DB 45H
DB 61H ;MOV R1,CDH
DB 0CDH
DB 34H ;R0->[R1]
DB 47H ;[RO]->R3
```

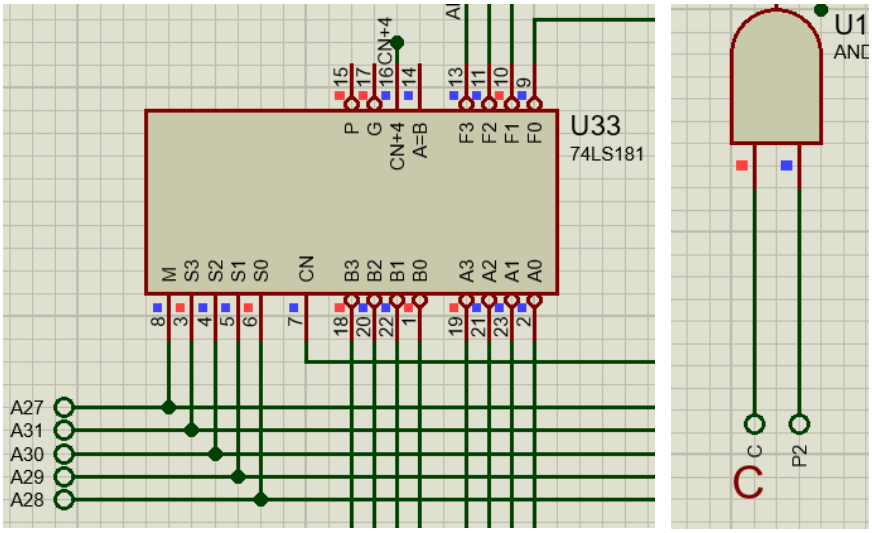
按下 QD，几个连拍后可看到 R3 显示 RAMOCDH 的地址

5.11 JC 指令

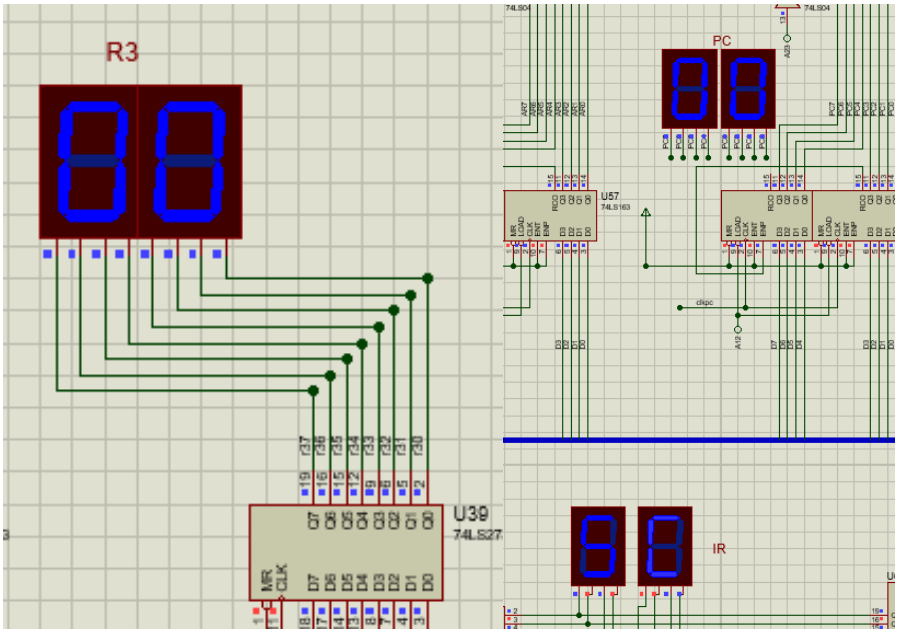
提前向 ROM 输入测试指令，如下图。

```
ORG 30H
    DB 60H ;MOV R0,99H
    DB 99H
    DB 61H ;MOV R1,8AH
    DB 8AH
    DB 63H ;MOV R3,00H
    DB 00H
    DB 01H ;ADD R1,R0
    DB 5CH ; JC R3
```

执行到 ADD 指令时，可以注意到 ALU 高位的 CN 处于有效状态，C 也被锁存起来。



执行 JC 后，PC 的地址被置为 0（R3）



5.12 STP 指令

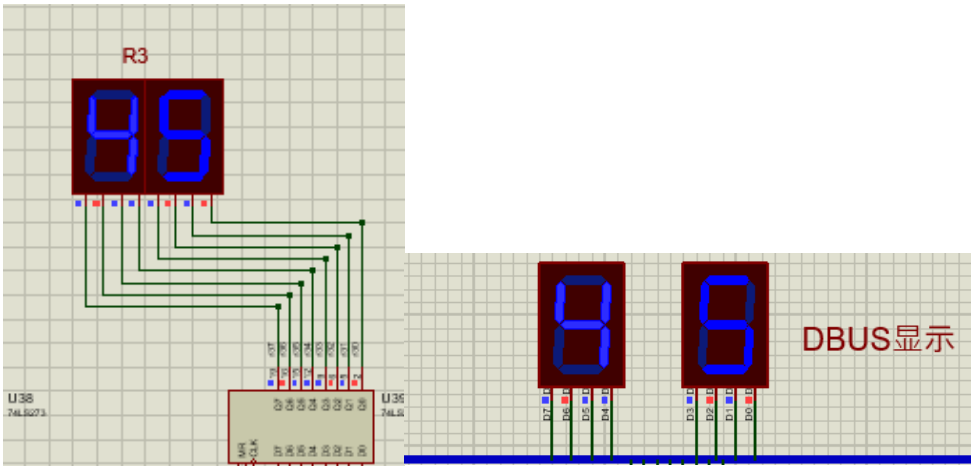
此指令无需单独测试，此前每个指令结束后都会以 STP 指令结尾。

5.13 OUT 指令

提前向 ROM 输入以下测试指令。

```
DB 60H ;MOV R0,45H
DB 45H
DB 61H ;MOV R1,CDH
DB 0CDH
DB 34H ;R0->[R1]
DB 47H ;[RO]->R3
DB 73H ;OUT R3
```

按下 QD，连续模式下可以看到 R3 被输出到数据总线。

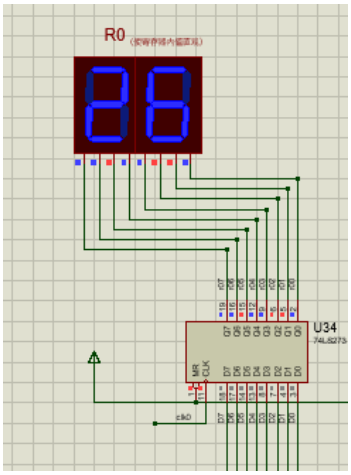


5.14 SHR 指令

提前向 ROM 输入以下测试指令。

```
DB 60H ;MOV R0,99H
DB 99H
DB 80H ;R0右移2位
```

按下 QD 后，连拍模式下即可看到 R0 右移了 2 位。

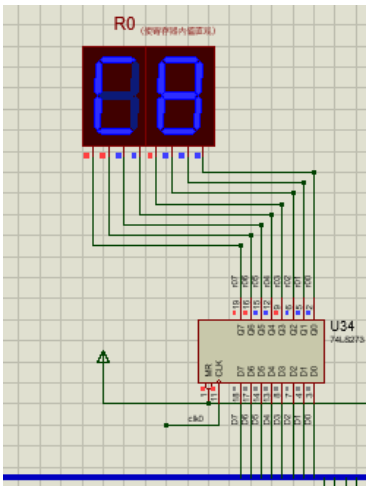


5.15 SHL 指令

提前向 ROM 输入以下测试指令。

```
DB 60H ;MOV R0,99H
DB 99H
DB 90H ;R0左移3位
DB 30H
```

按下 QD 后，连拍模式下即可看到 R0 左移了 3 位。

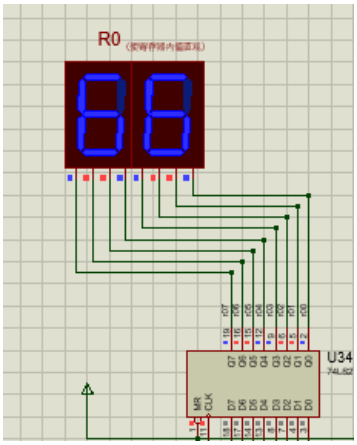


5.16 ROR 指令

提前向 ROM 输入以下测试指令。

```
ORG 40H
    DB 60H ;MOV R0,99H
    DB 99H
    DB 80H ;R0循环右2
    DB 0A0H
```

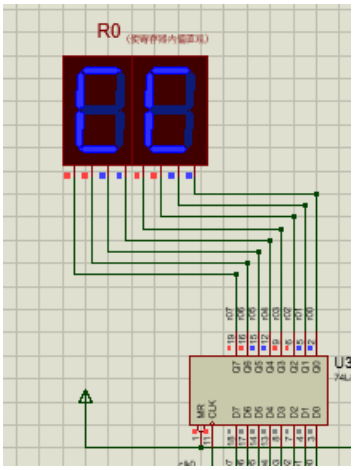
按下 QD 后，连拍模式下即可看到 R0 循环右移了 2 位。



5.17 ROL 指令

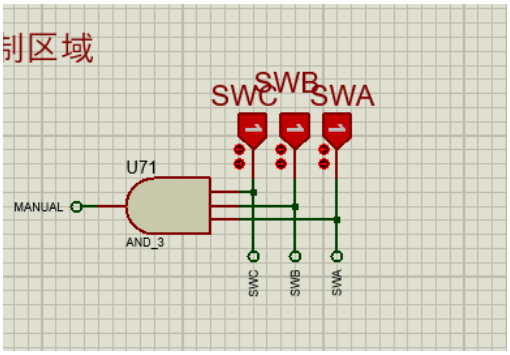
提前向 ROM 输入以下测试指令。

按下 QD 后，连拍模式下即可看到 R0 循环左移了 3 位。



5.18 手动控制

将 SWC, SWB, SWA 调至 111 状态。



即可借助旁边 40 个 LOGICSTATE 控制 IR 和所有微命令。值得注意的是，如果从手动状态切换到自动状态，需要重启 TEC-5H，否则有概率导致线路短路。

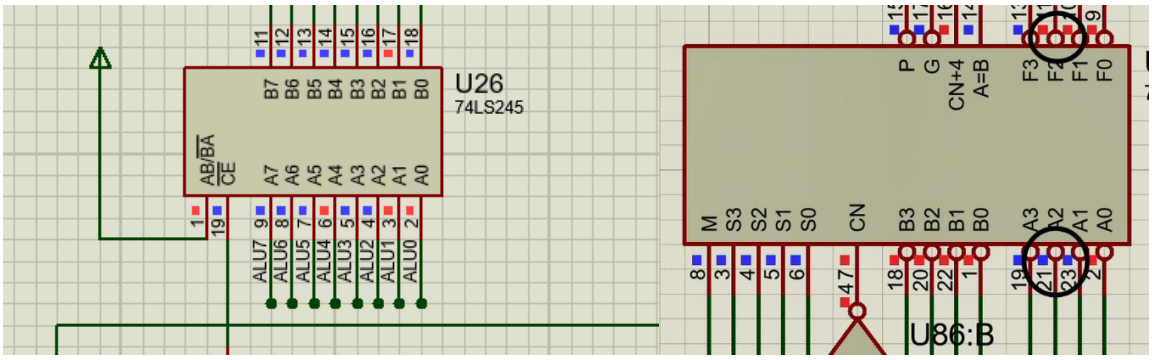
6 心得体会

本次课设是我学习计算机组成原理和使用 Proteus 仿真软件后遇到的最大的一次挑战，也是对我计组掌握知识最大的一次考验。这一周，我白天奋战在机房，晚上转移到寝室，遇到了无数个 BUG 和 ERROR，通过排查电路、查找资料和交流又逐步解决或是回避了它们。现在正在写报告的我，不仅实现了 TEC-5 原有的功能，还在此基础上增加了实用的指令，回顾这一周，成就感十足。

在课设开始的第一天，我就给自己定了一个计划，在第一天将上学期有关数据通路和微程序的实验整合起来，实现纯手动控制所有的微命令，并试着运行了最简单的微指令。在这一天，主要困扰我的还是我本人的失误，比如接线错误和逻辑电路本身的错误，在一次次地缩小错误范围后，都能比较容易地发现问题的根源。

课设第二天我开始在上学期最后一次实验的基础上补充了所有的微指令。在这一天，我意识到了烧写 4 份 HEX 文件给 4 个微指令存储器过程繁琐。由于我此前有 Python 的相关基础，在查阅了 HEX 的烧写规则后，我试着用 Python 读取我写的 asm 格式的微指令，并将其根据格式改写为 HEX。经过大约一小时的摸索，我成功地在 Proteus 上读取了用 Python 烧写的文件。这大幅提升了我后续修改和新增微指令的速度。成功地在第三天实现了 TEC-5 的所有功能，并新增了立即寻址指令。在第四天，我也顺利地设计出了移位控制器。

但是在后续的测试过程中，我遇到了 Proteus 固有的 BUG，如 74LS244 和 74LS245 在有效情况下传输错误和 74LS181 在多次 DR1 直通后 ALU 的低三位高电平错误（如下图）。一开始我以为是我自身原因导致的，但是在几小时的排查后，我确定了这是硬件本身的问题。在以前，我是不敢断言的，但是这一个星期几十小时的 DEBUG 经验在磨练了我耐心的同时，增强了我寻找问题的能力。



总而言之，虽然我在最后一天尝试了乘法但失败了，但我对我设计的 TEC-5H 非常满意。这次课设加深了我对数据通路、微程序等计组原理的理解，会对我之后相关专业知识的学习带来巨大的帮助。