**大连海事大学**

**实践课程报告**

计算机组成原理课程实践

**学 院 信息科学技术学院**

**专 业 网络工程**

**课程名称 计算机组成原理实践**

**组序号 15**

**成员姓名 李子毅、李毓琪、汪灵朴**

**指导教师 陈媛嫄、张少仲**

**起始日期 2021年7月12日——7月23日**

|  |  |
| --- | --- |
| **小组完成的主要工作** | 1. 复杂模型机设计实验（实验指导书上实验5.3） 2. 复杂模型机设计    1. **电路图的改装与设计**    2. **编写微指令，实现了立即寻址的操作**    3. **编写微指令，实现了离散数学中逻辑输出律的操作**    4. **改正了I/O相关指令，构建全新的指令系统。** |
| **成**  **员**  **分工** | **组长：学号2220191195、姓名李毓琪**  **主要电路图的搭建、电路图的设计与改装，微指令的编写与改错，查阅相关器件资料。**  **成员1：学号2220193626、姓名李子毅**  **部分电路图的搭建，微指令的编写，指令的编写，指令系统的构建，实验报告的撰写。**  **成员2：学号2220194464、姓名汪灵朴**  **部分电路图的搭建，微指令的编写与烧制，以及后续的查缺补漏。** |
| **综合评语** | **（设计方案、实践环节、问题解答、设计报告）** |

|  |  |
| --- | --- |
| 成  绩  评  定 | **成员1：学号、姓名**  2220191195 李毓琪  **成员2：学号、姓名**  2220193626 李子毅  **成员3：学号、姓名**  2220194464 汪灵朴 |

第一部分 复杂模型机设计实验

一、复杂模型机设计实验（实验指导书上实验5.3或3.1）

1.1实验原理

1.2 实验过程

1.3 实验结果（拷屏或拍照）

第二部分 复杂模型机设计

二、指令系统与寻址方式

* 1. **指令系统的设计**

**2.1.1指令设计的思路**

**2.1.2数据格式**

**2.1.3 指令格式**

**2.1.4 指令系统**

* 1. **寻址方式的设计**
     1. **寻址方式的设计思路**
     2. **立即寻址的流程**
     3. **立即寻址操作的演示**

1. **数据通路与微指令系统的设计**

**3.1 数据通路**

**3.2微指令设计思路**

**3.2.1 微指令格式**

**3.2.2微指令系统**

**3.2.3 微指令的设计->实现（（P->Q）->R）**

**四、各组成部分详细设计过程**

**4.1 组成部分设计逻辑**

**4.2组成部分设计过程**

**4.2.1微程序控制器**

**五、课程设计总结**

**六、参考文献**

**第一部分**

1. 复杂模型机设计实验

1.1实验原理

本实验的主要内容是掌握基于微程序控制器的 CPU 组成结构，了解 CPU 的中断工作机制，熟悉 CPU 微指令设计，掌握机器指令的微程序实现方法。本实验将设计一个微程序 CPU,其中包括微程序控制器、运算器、存储器、寄存器 堆及外部 I/O 接口。定义一套较完备的机器指令集，编写每条机器指令对应的 微程序，在 CPU 电路上运行基于上述机器指令集的机器语言程序，并且用汇编助记符(语言)加以注释。

1.2实验过程

(1)按照实验原理要求搭建复杂模型机电路。

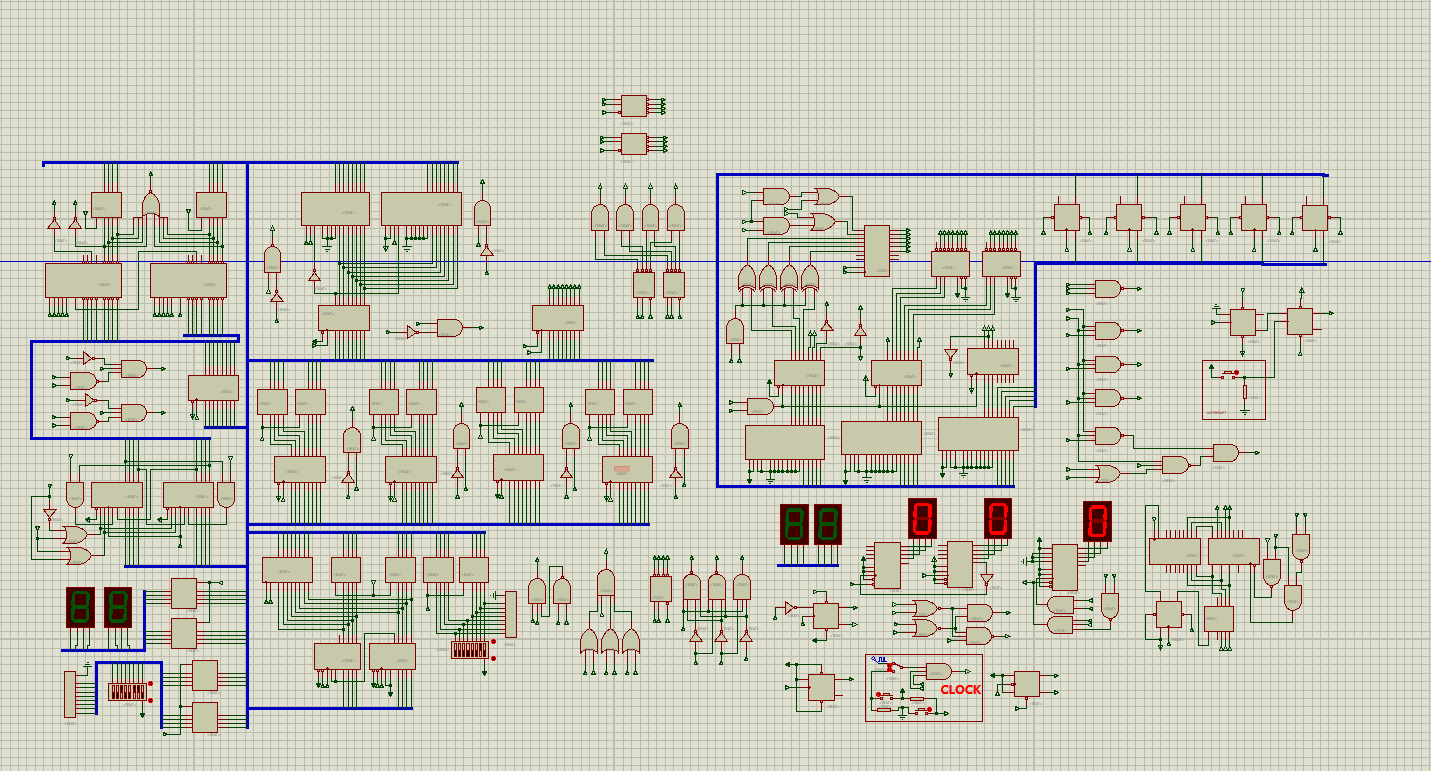
(2)烧制程序并存入ROM。

(3)按照不同程序的要求运行程序。

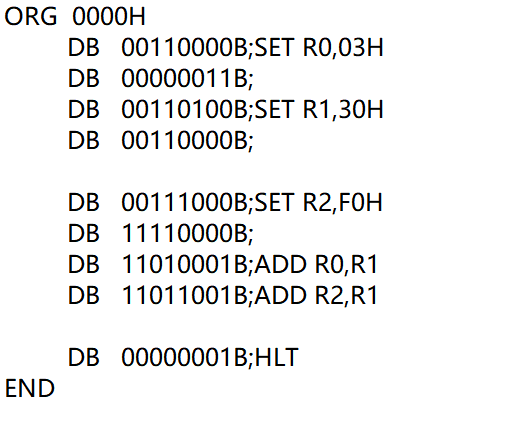
(4)更改数据为05H,0AH,0EH，测试ADD程序。

1.3实验结果

(1)首先按照要求，搭建好电路如下：



(2)烧制程序：由于比较简单，这里只展示ADD程序的编写。

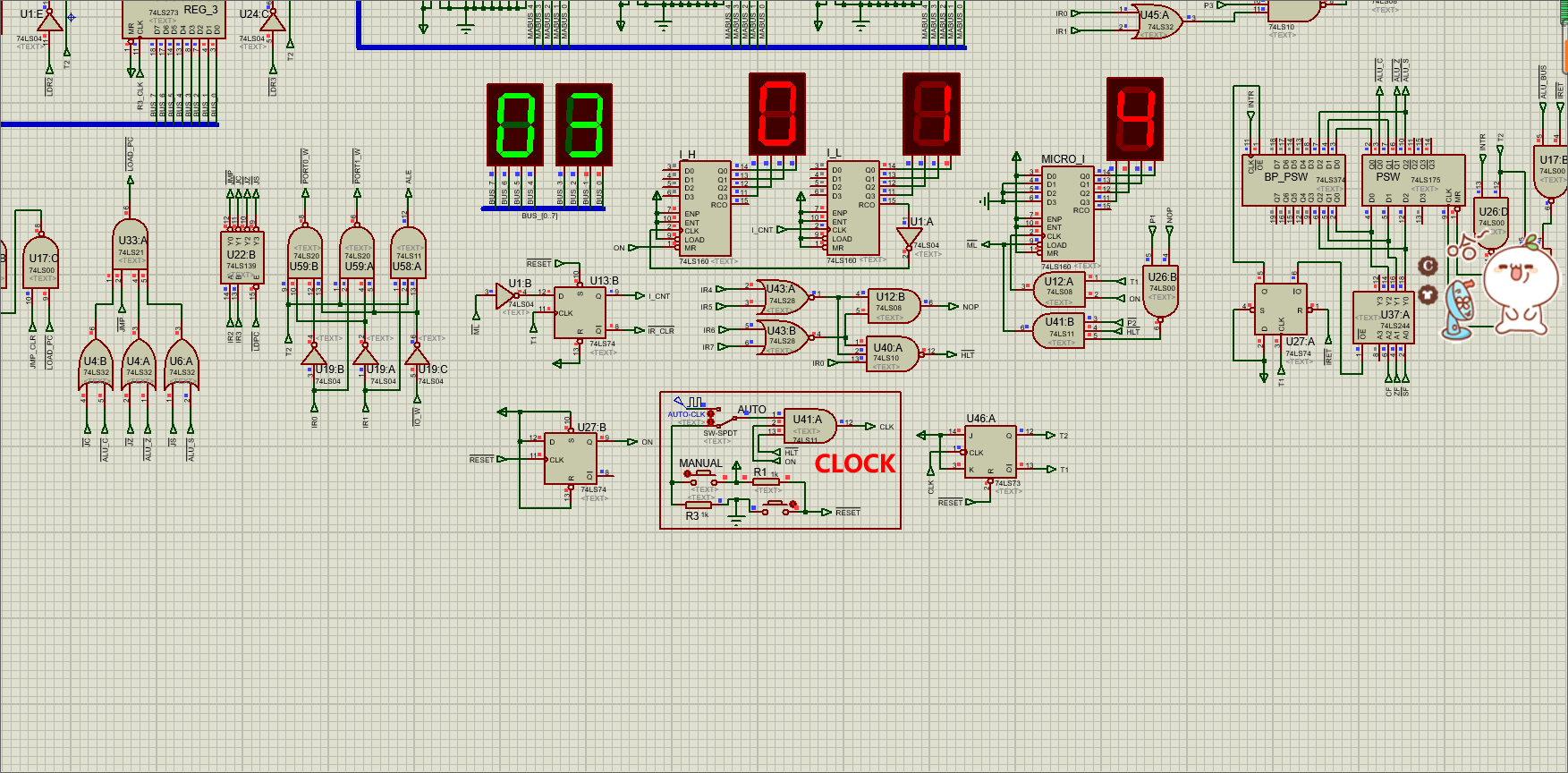


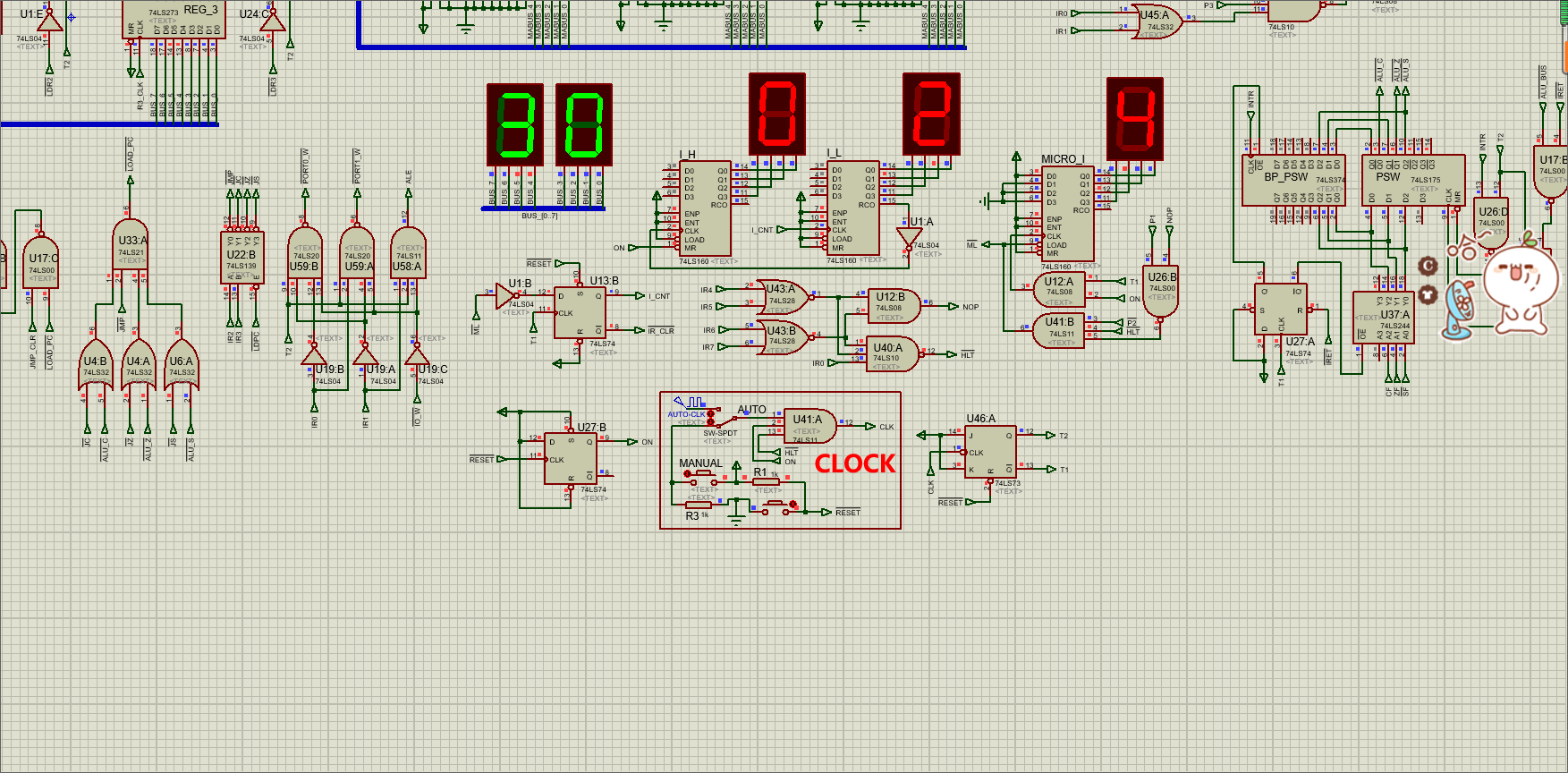
(3)ADD程序的运行过程及数据记录如下。

这里通过SET R0,SET R1以及ADD R1+R0的结果来验证电路图的搭建是否准确无误，通过上述烧制的程序可知，最终的结果应该是33H。

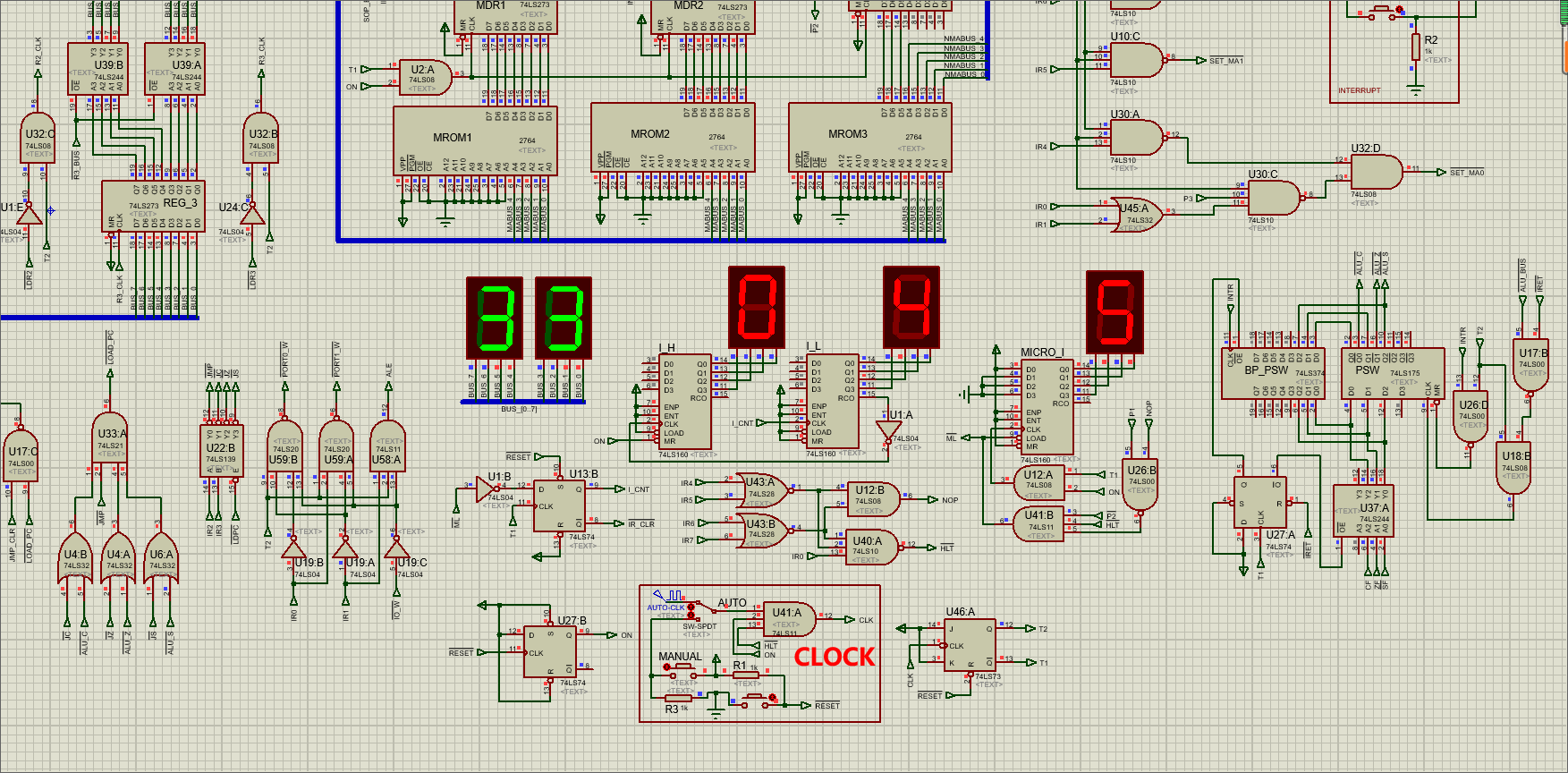
运行过程截图如下：

SET R0:



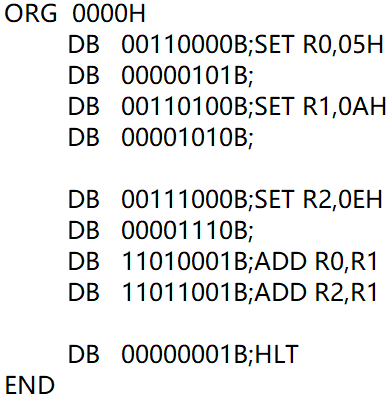
SET R1:

ADD R0 R1:



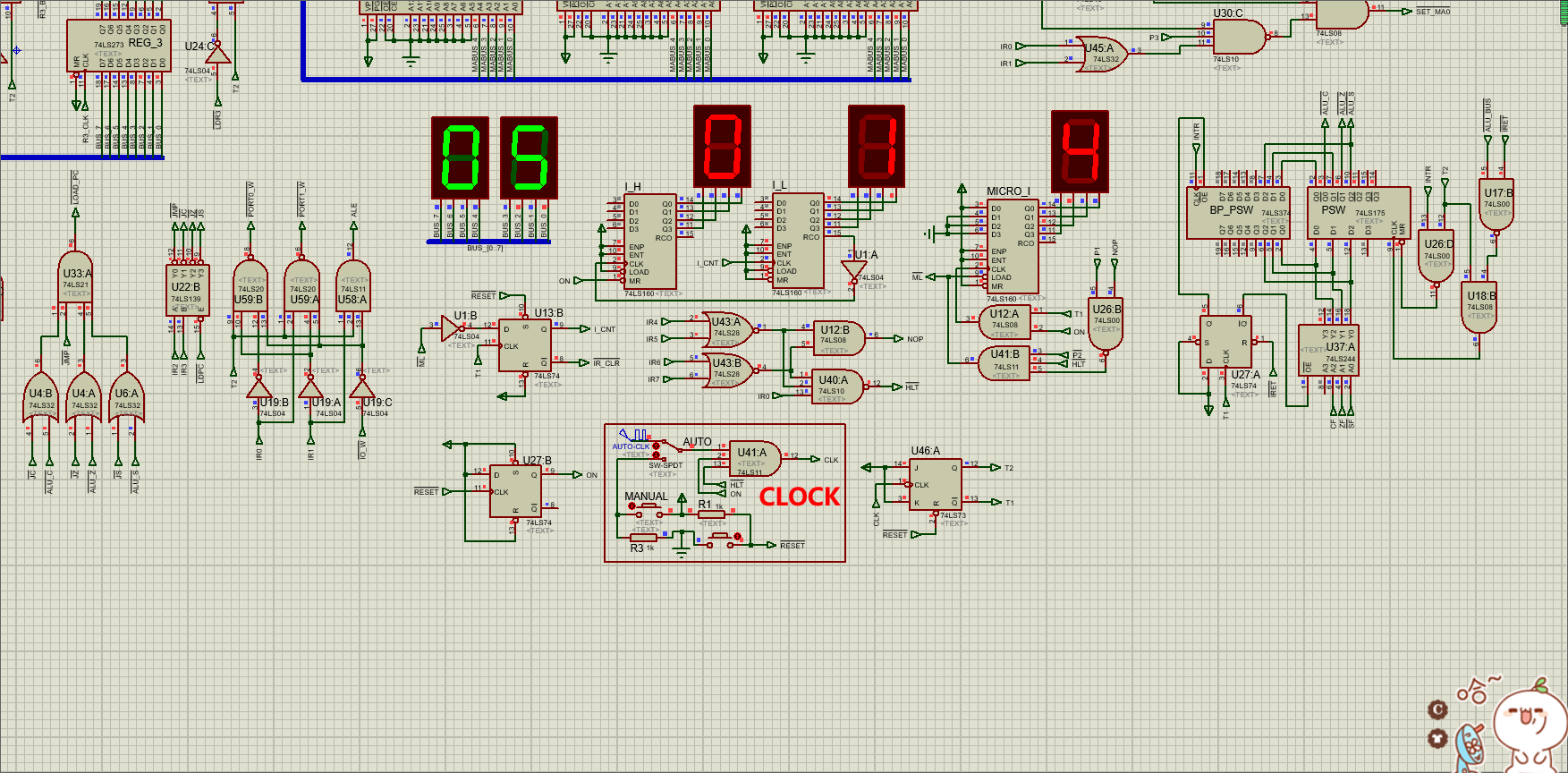
可以看到，是符合运算结果的，这说明电路图的搭建成功。

(4)为了进一步验证，重新烧制的程序如下

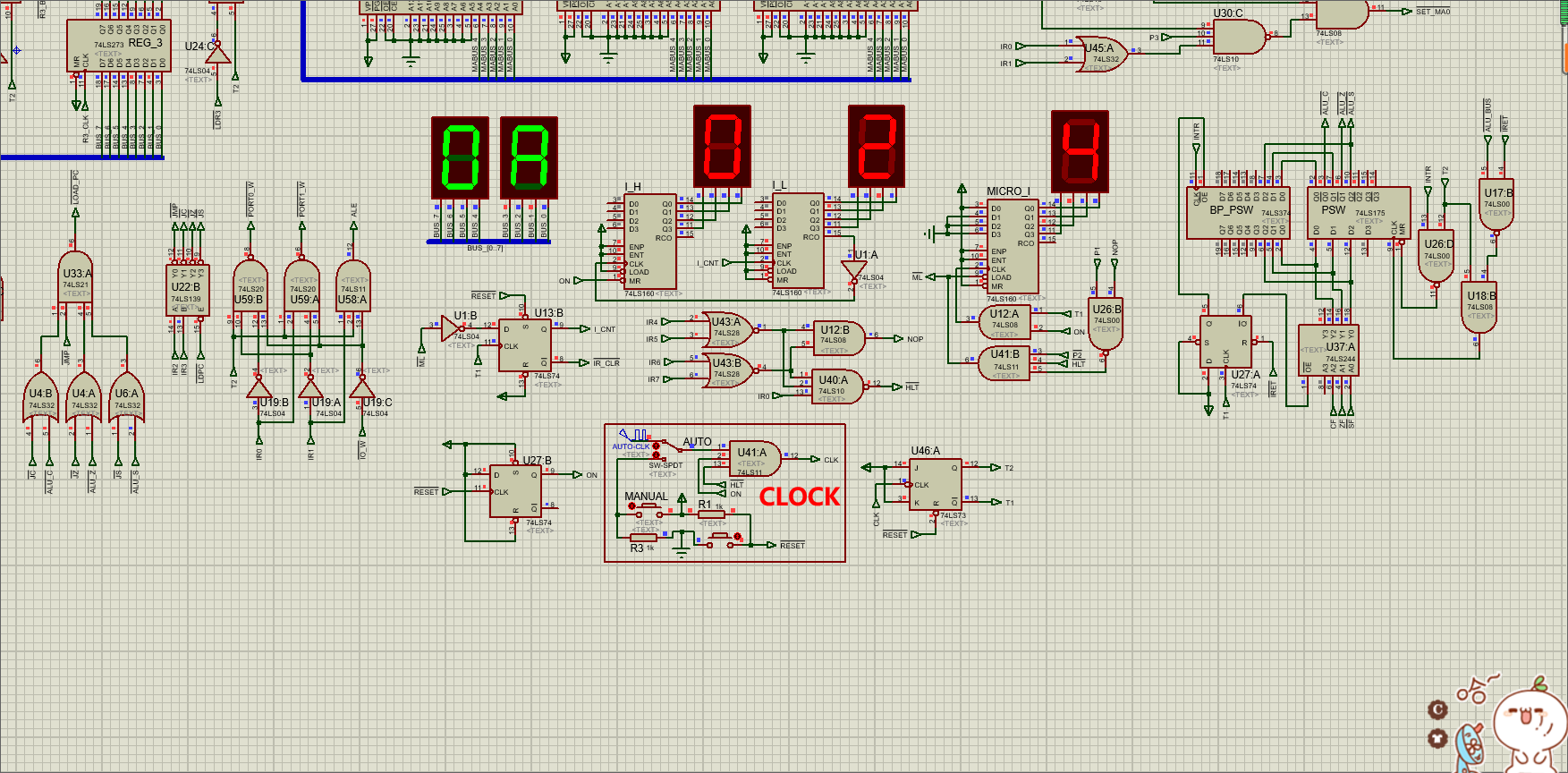


同样我们查看与上一步相同的三步的结果

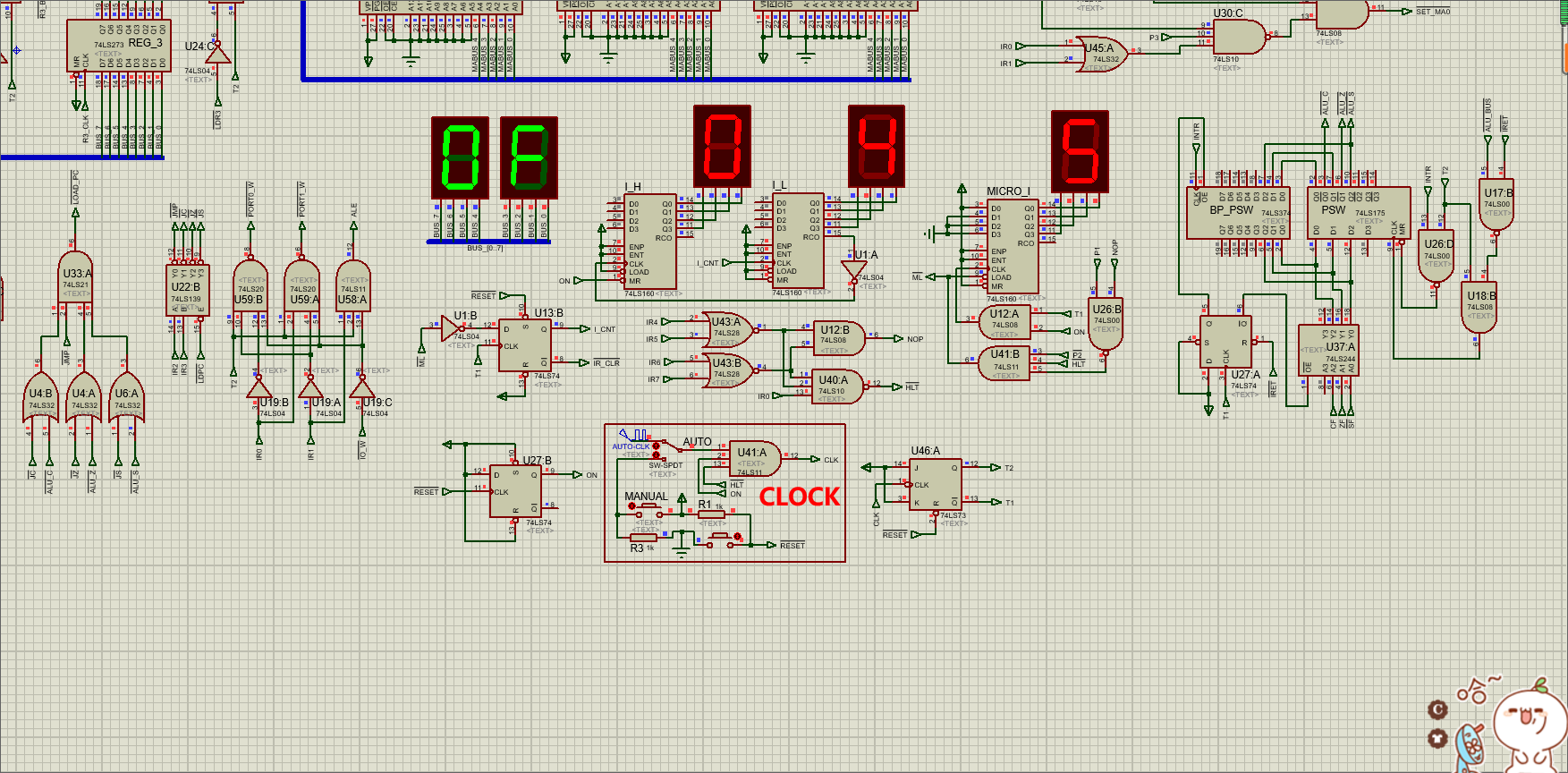
SET R0:



SET R1:



ADD R0 R1:

经计算发现也是符合要求的。

**第二部分** 复杂模型机设计

二、指令系统与寻址方式

2.1指令系统的设计

本章我们主要完成了一个指令系统的设计，用于实现一系列操作。

2.1.1 指令设计的思路

指令共分为三大类：运算类、控制转移类、数据传送类，其中运算类指令包含算术运算、逻辑运算和移位运算三种，设计有3条运算类指令，分别为：ADD、SUB、AP(实现/A||B操作的一条指令),控制转移类指令设置三条:HLT、MOV，JMP，数据传送类设置三条SET、LJXZ(立即寻址所用的指令),IN(改写后的IN指令)。

2.1.2数据格式

模型机规定采用定点补码表示法表示数据，字长为８位，8 位全用来表示数 据（最高位不表示符号），数值表示范围是： 0≤X≤28－1。

2.1.3.指令格式

所有单字节指令（ADD、SUB、OR、HLT 和 MOV）格式如下：

|  |  |  |
| --- | --- | --- |
| 7 6 5 4 | 3 2 | 1 0 |
| OP-CODE | RS | RD |

其中，OP-CODE 为操作码，RS 为源寄存器，RD 为目的寄存器，并且有：

|  |  |
| --- | --- |
| RS或RD | 寄存器 |
| 00  01  02  03 | R0  R1  R2  R3 |

IN指令和OUT指令的格式为：

|  |  |  |  |
| --- | --- | --- | --- |
| 7 6 5 4(1) | 3 2(1) | 1 0(1) | 7-0(2) |
| 操作码 | RS | RD | P |

其中括号中的 1 表示指令的第一字节，2 表示指令的第二字节，在本实验中，仅有一个 IN 单元 和一个 OUT 单元，所以不需要对地址进行译码，即第二个字节数据不产生影响。

寻址操作上，我们主要实现了立即寻址这一指令，格式如下：

|  |  |
| --- | --- |
| 7 6 5 4 | 3 2 1 0 |
| 操作码 | 立即数 |

通过修改电路，将立即寻址指令中隐含的立即数取出，并经过运算得到需要的数据。

2.1.4 指令系统  
本模型机我们共设置了9条指令，由于扩展了微指令的位数，所以理论上应该可以实现更多的指令，这里仅以9条指令来包含不同类别的指令，指令的格式如下。

|  |  |  |
| --- | --- | --- |
| 助记符号 | 指令格式 | 指令功能 |
| ADD | 1000 | 将两个寄存器中的数取出并相加再送到寄存器中 |
| SUB | 1100 | 将两个寄存器中的数取出并相减再送到寄存器中 |
| AP | 0110 | 实现了(P->Q)->R操作 |
| SET | 0011 | 将某一数据置入寄存器中 |
| MOV | 0101 | 将某一寄存器中的数据移动到另一寄存器中 |
| IN | 0100 | 从IN单元读入数据到寄存器中 |
| LJXZ | 1111 | 实现了立即寻址操作 |
| JMP | 0001 | 将主存单元的数据放入PC中 |
| HLT | 1110 | 停机 |

2.2 寻址方式的设计

本节我们主要**实现了课本上的立即寻址**操作，其中利用了微指令的保留字段，并**改了一部分电路，编写了立即寻址指令所需的微指令和指令。**

2.2.1寻址方式的设计思路

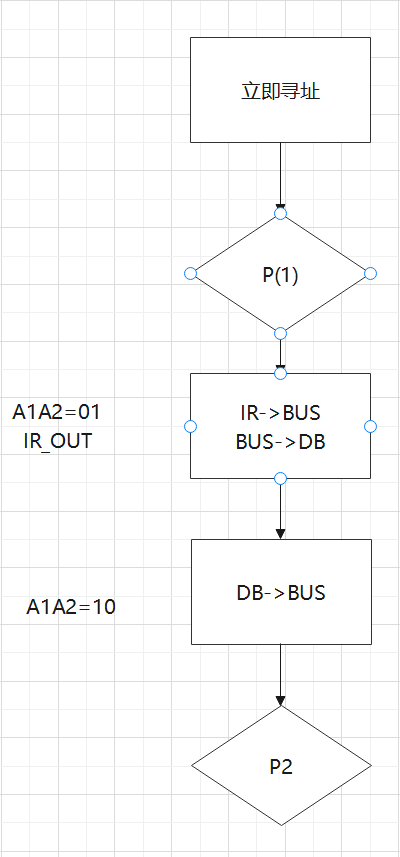
首先先查找课本上立即寻址这一寻址方式的定义如下：

文本

描述已自动生成

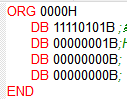
一开始时，我们分不清立即寻址与隐含寻址这两种寻址方式的差别，在决定设计的过程中，经常出现错误，经过认真思考，最终理解到，立即寻址这一寻址方式的要点在于所要查找的数据就存在于指令的部分位数中，这也被称之为立即数，这段数据由补码表示，所以设计的要点就是，如何确定这条指令是一条立即寻址指令，以及如何将立即数提取出来之后组成微指令来完成取址操作，有了这两个问题，就可以开始着手解决问题了。

2.2.2 立即寻址的流程：

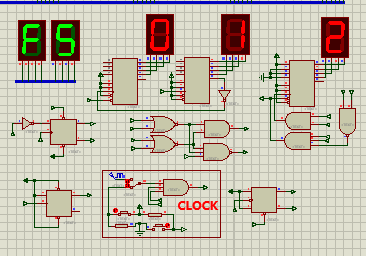


P2操作测试之后的情况以及取指周期的微指令这里不再展示，其中A1A2字段为01时，将操作码中的立即数读取进入新加的缓冲器中，其中A1A2字段为10时，将立即数的后四位填成完整的八位数据，也就是需要查找的数据，显示出来，至于A1A2字段的含义与相应的实现原理，会在第四章结合电路进行讲解。

2.2.3立即寻址操作的演示

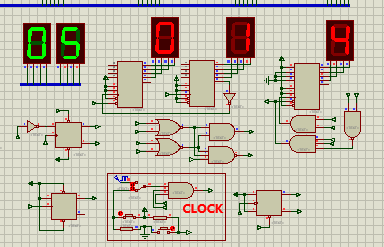


指令的编写如上，其中操作码为1111，立即数为0101。

运行立即寻址指令，查看BUS上的输出。

为指令的数据。

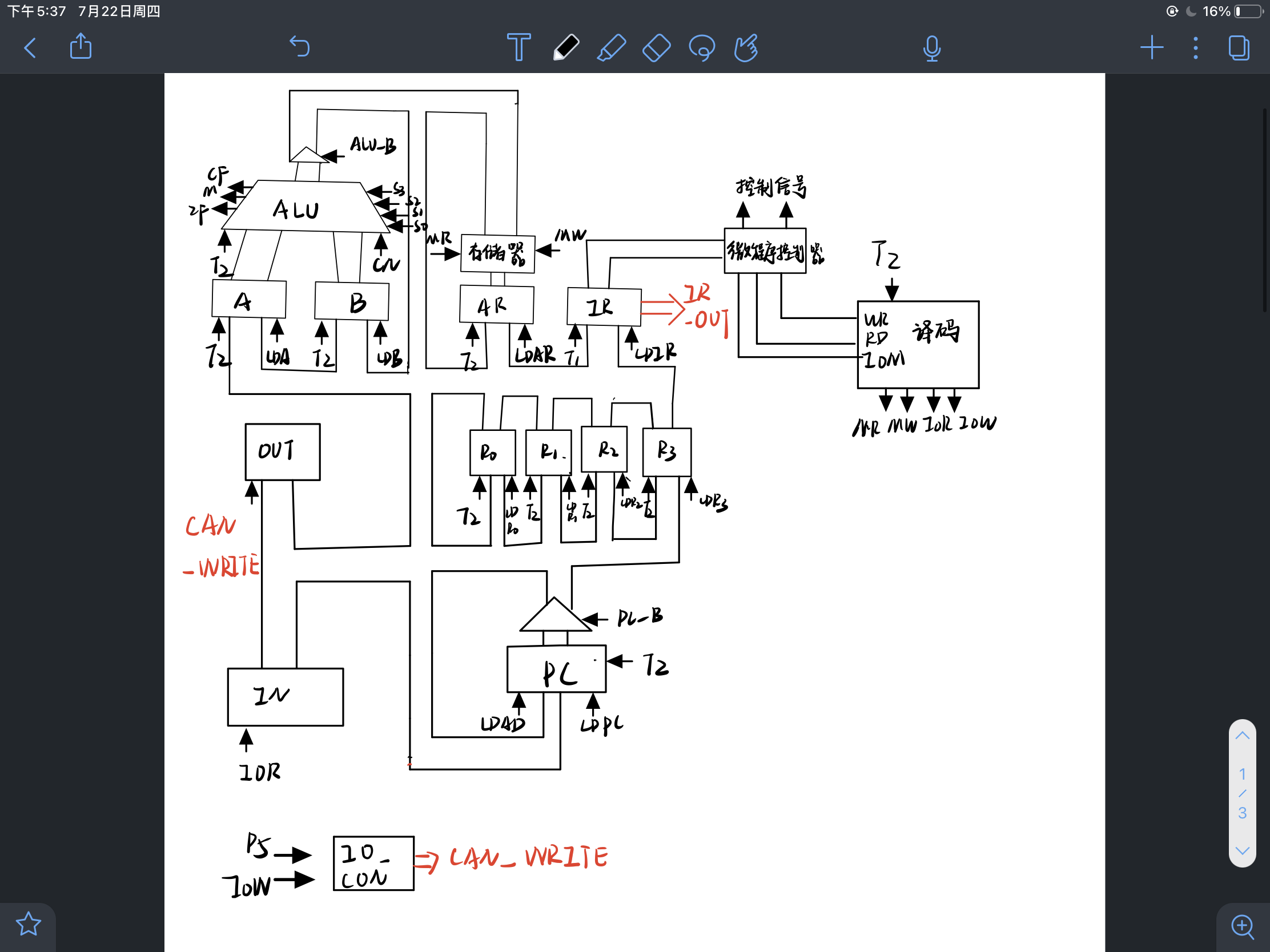
再运行两个时钟周期：



可以看到，总线上数据为05H，也就是0000 0101B，为立即数。

三、数据通路与微指令系统的设计

3.1 数据通路

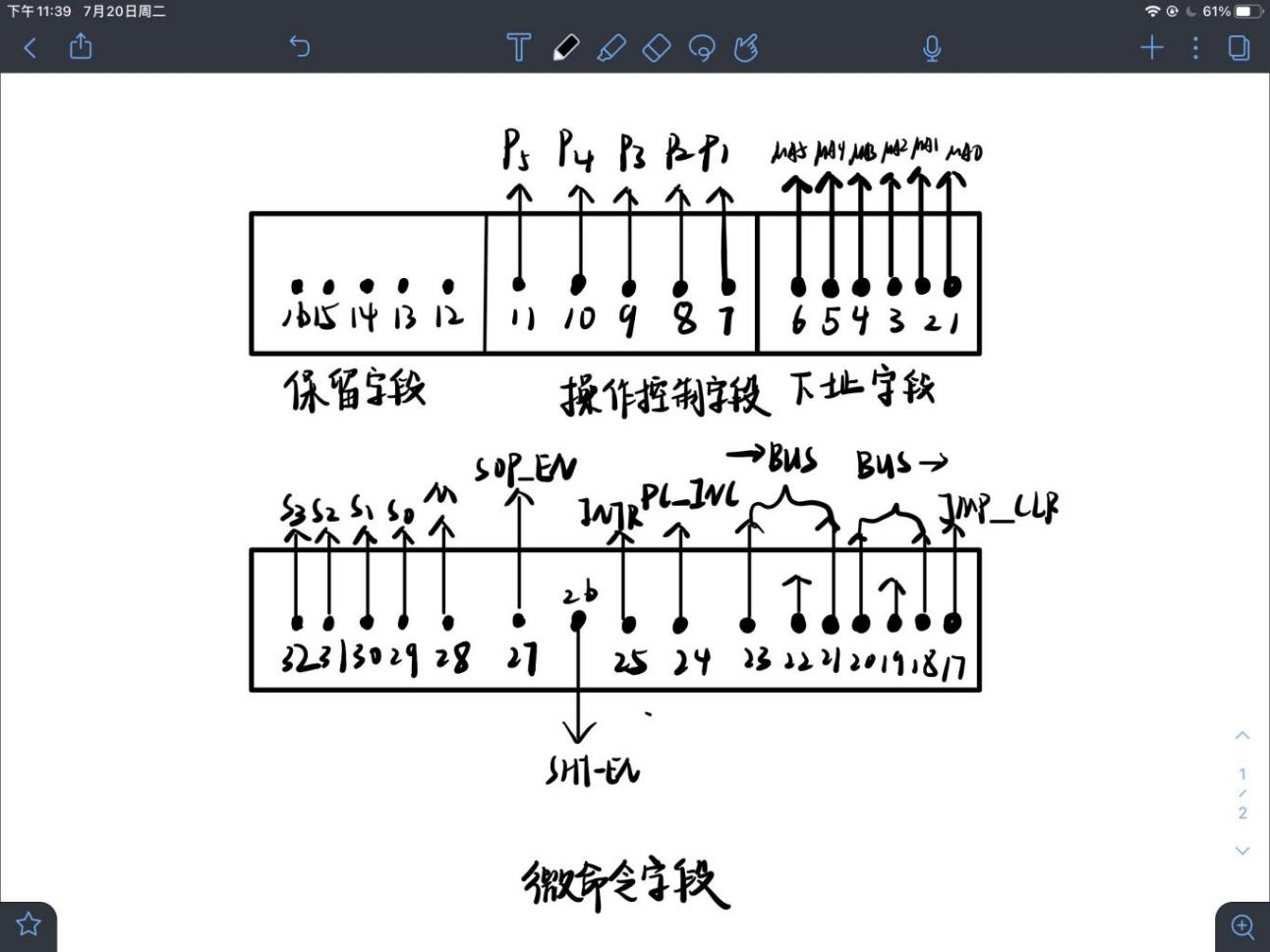


3.2 微指令系统的设计

本章我们主要实现了一个微指令系统，用于包含我们指令系统所需的微指令，另外，设计了一个逻辑运算的操作指令，在下文展示。

3.2.1微指令格式

在本次实验中，我们对微指令格式进行了修改，修改微指令的位数为32位，这样可以保证我们指令设计的灵活性和创造性，保证设计的指令有足够多的空间来编写微指令，具体的格式如下图所示：



其中：

1-6为下址字段uA0-uA5。

7-11为操作控制字段，P1,P2,P3的作用与之前相同，P4为I/O控制字段，I/O字段为1时允许打开I/O设备。P5留作保留，可用于中断控制。

12-16为保留字段，在立即寻址过程中，我们采用第12和第13位来实现立即寻址的功能，10和01分别代表立即寻址电路两个不同的状态。

17为JMP\_CLR,用于跳转指令。

18-20和21-23分别为总线的出操作和入 操作，采用138译码器实现，进行译码后的操作如下图所示：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ->BUS | | | | BUS-> | | | |
| 23 | 22 | 21 | 微命令 | 20 | 19 | 18 | 微命令 |
| 0 | 0 | 0 | \ | 0 | 0 | 0 | \ |
| 1 | 0 | 0 | PC\_BUS | 1 | 0 | 0 | LDAR |
| 0 | 1 | 0 | MEM\_OE | 0 | 1 | 0 | LDIR |
| 1 | 1 | 0 | IO\_R | 1 | 1 | 0 | LDD |
| 0 | 0 | 1 | RA\_BUS | 0 | 0 | 1 | LDR |
| 1 | 0 | 1 | RB\_BUS | 1 | 0 | 1 | RAM\_WE |
| 0 | 1 | 1 | ALU\_BUS | 0 | 1 | 1 | IO\_W |
| 1 | 1 | 1 | IRET | 1 | 1 | 1 | LDPC |

24为PC\_INC，代表PC+1操作

25为INTR,控制中断操作

26为SHT\_EN,用于移位操作的控制

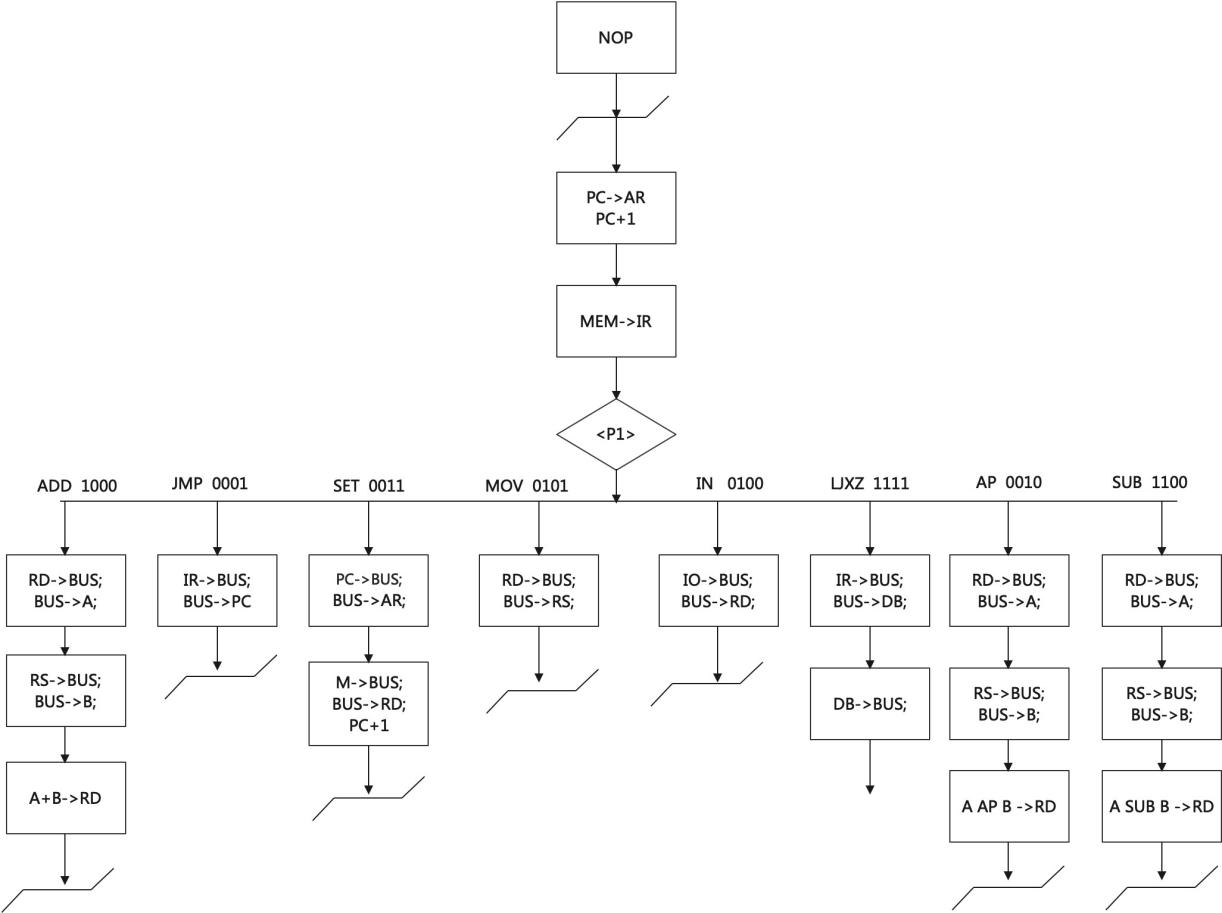
27为SOP\_EN，用于表示单操作数

28-32用于ALU的算术逻辑运算

3.2.2**微指令系统**

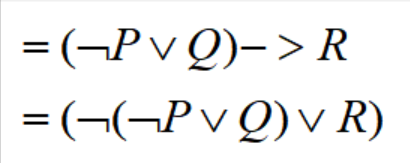
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Num | 微指令名 | 微指令地址 | 高五位 | SOP\_EN | SHT\_EN | INTR | PC\_INC | 六位BUS字段 | JMP\_CLR | 保留字段 | 操作控制 | 下址字段 |
| 1 | PC->BUS;BUS->AR(取值) | 000000 | 00000 | 0 | 0 | 0 | 0 | 100100 | 0 | 00000 | 00000 | 100001 |
| 2 | M->BUS;BUS->IR;PC+1 | 100001 | 00000 | 0 | 0 | 1 | 1 | 010010 | 0 | 00000 | 00001 | 000000 |
| 3 | PC->BUS;BUS->AR(SET) | 000011 | 00000 | 0 | 0 | 0 | 0 | 100100 | 0 | 00000 | 00000 | 100100 |
| 4 | M->BUS;BUS->RA;PC+1 | 100100 | 00000 | 0 | 0 | 0 | 1 | 010001 | 0 | 00000 | 00010 | 000000 |
| 5 | RB->BUS;BUS->RA(MOV) | 000101 | 00000 | 0 | 0 | 0 | 0 | 101001 | 0 | 00000 | 01000 | 000000 |
| 6 | PORTX->BUS;BUS->RA(IN) | 000100 | 00000 | 0 | 0 | 0 | 0 | 110001 | 0 | 00000 | 01010 | 000000 |
| 7 | RA->BUS;BUS->DA(ADD) | 001101 | 10010 | 0 | 0 | 0 | 0 | 001110 | 0 | 00000 | 00100 | 100111 |
| 8 | PC->BUS;BUS->AR(2) | 100111 | 00000 | 0 | 0 | 0 | 0 | 100100 | 0 | 00000 | 00000 | 101110 |
| 9 | M->BUS,PC->DB,PC+1 | 101110 | 00000 | 0 | 0 | 0 | 0 | 010110 | 0 | 00000 | 00000 | 101000 |
| 10 | ALU->BUS;BUS->RA; | 101000 | 00000 | 0 | 0 | 0 | 0 | 011001 | 0 | 00000 | 01000 | 000000 |
| 11 | IR->BUS;BUS->DB | 001111 | 00000 | 0 | 0 | 0 | 0 | 000000 | 0 | 00001 | 10000 | 100011 |
| 12 | DB->BUS | 100011 | 00000 | 0 | 0 | 0 | 0 | 000000 | 0 | 00001 | 10000 | 100011 |
| 13 | IR->BUS  BUS->PC | 000001 | 00000 | 0 | 0 | 0 | 0 | 000111 | 1 | 00000 | 00000 | 000000 |
| 14 | RD->BUS;  BUS->A | 001100 | 00000 | 0 | 0 | 0 | 0 | 001110 | 0 | 00000 | 00000 | 101100 |
| 15 | RS->BUS;  BUS->B | 101100 | 00000 | 0 | 0 | 0 | 0 | 101001 | 0 | 00000 | 00000 | 101101 |
| 16 | A SUB B->RD | 101101 | 01100 | 0 | 0 | 0 | 0 | 011110 | 0 | 00010 | 00000 | 000000 |
| 17 | RD->BUS,BUS->A | 000110 | 00000 | 0 | 0 | 0 | 0 | 001110 | 0 | 00000 | 00000 | 100110 |
| 18 | RS->BUS,BUS->B | 100110 | 00000 | 0 | 0 | 0 | 0 | 101001 | 0 | 00000 | 00000 | 100111 |
| 19 | A AP B->RD | 100111 | 00000 | 0 | 0 | 0 | 0 | 011110 | 0 | 00010 | 00000 | 000000 |

微指令的程序框图如下所示：



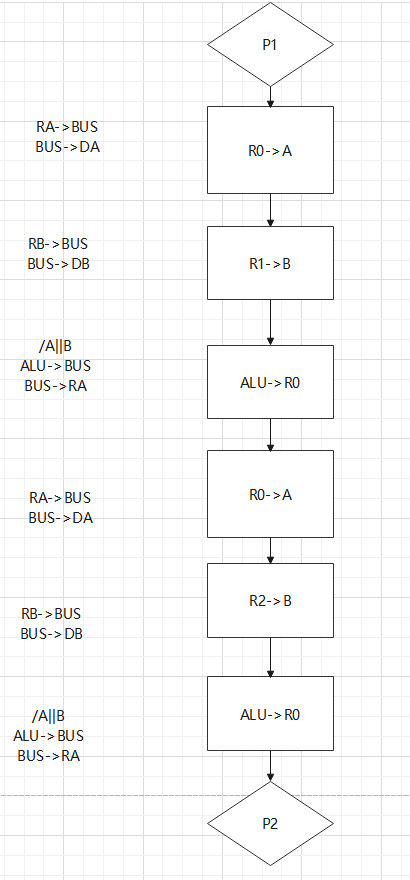
3.2.3微指令的设计——**实现((P->Q)->R)操作**

为了实现这一操作，首先我们利用离散数学的知识将其化简：



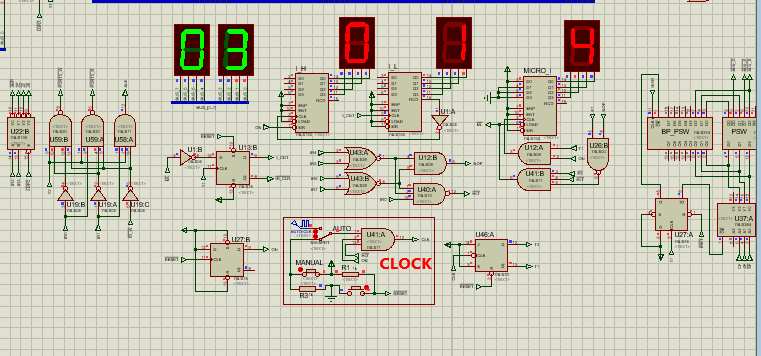
由于本指令是一个逻辑运算指令，所以需要用到ALU来实现，所以我们查阅了ALU功能所需的最后五位，也就是28-32位的操作表，如下：

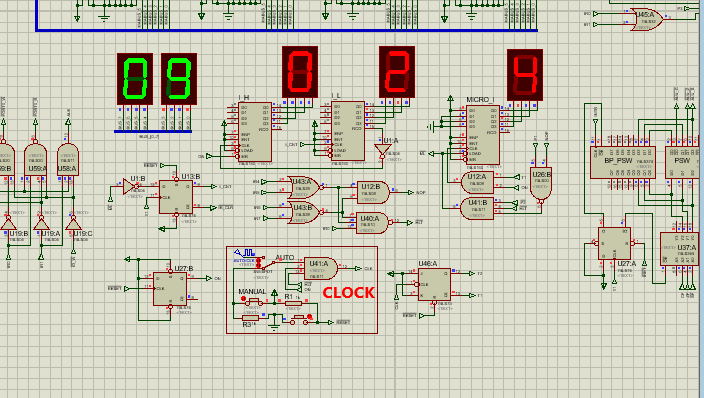
然后根据化简的结果，查找到所需的操作，编写微指令，流程图如下：



操作的具体实现如下：

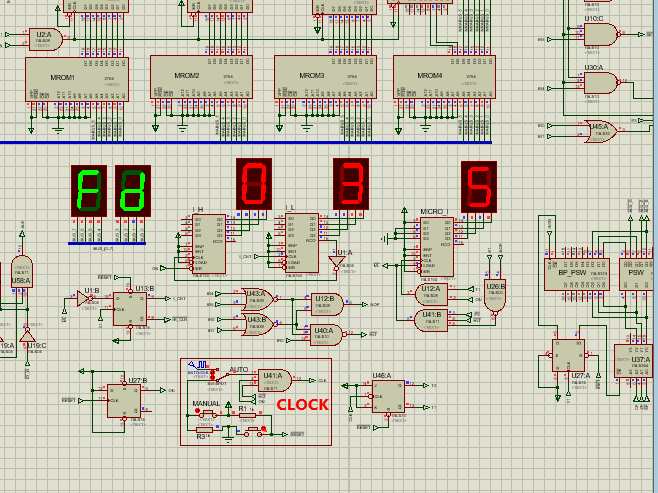
首先利用SET指令对R0、R1进行置数00000011和00001001。

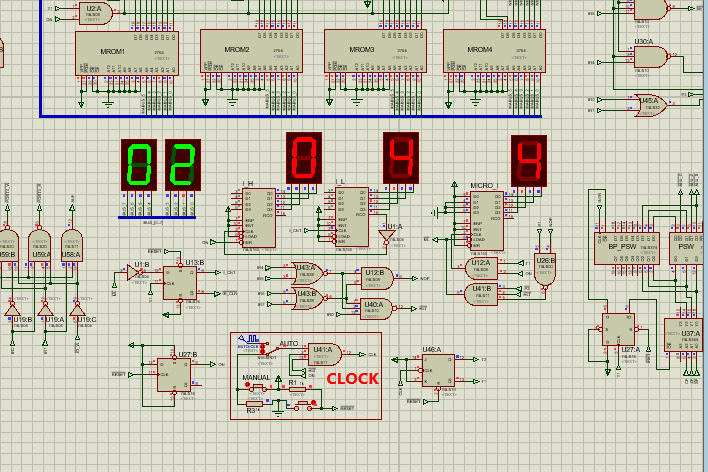




然后进行第一次运算，就是11111100和00001001进行并运算。

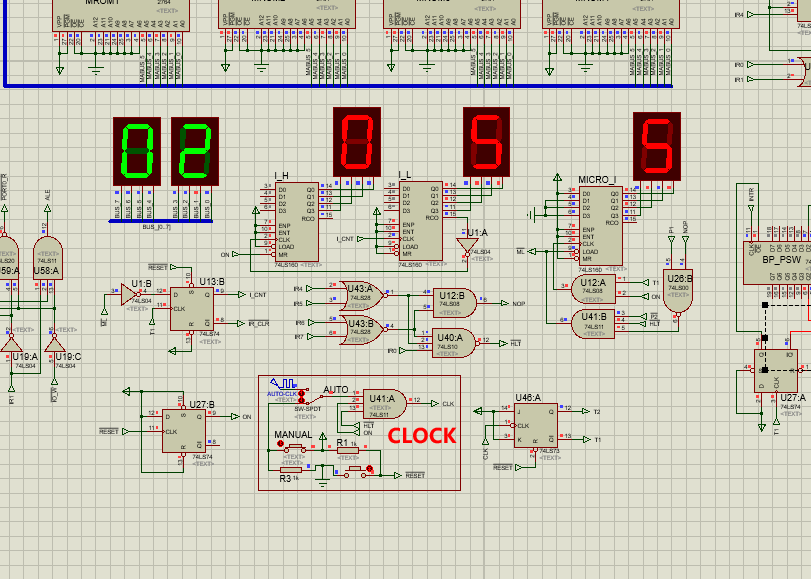
得到11111101，转化成十六进制为FD，如图所示。





然后进行第二次置数，R2为02H。

然后与第一次操作之后的结果进行上述运算，00000010(上述结果取反)和00000010进行并运算得到结果为00000010(02)。



四、各组成部分详细设计过程

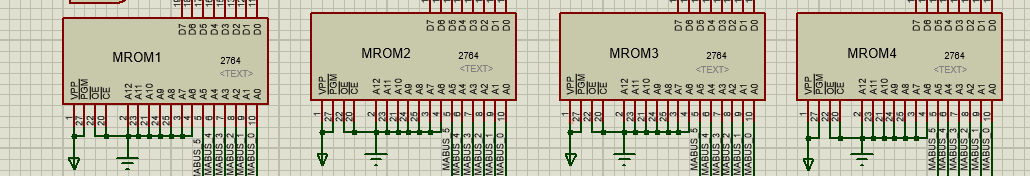
4.1组成部分设计逻辑

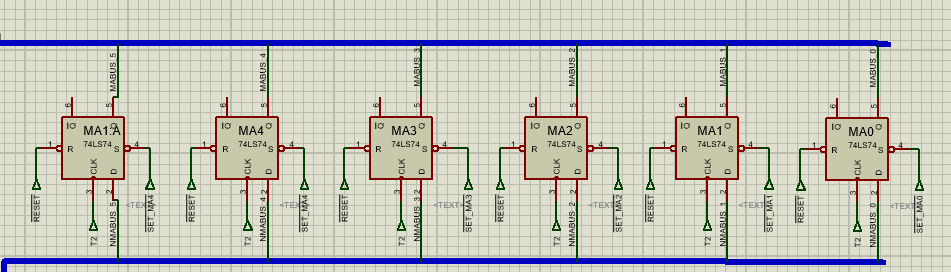
本章将叙述基于指令和微指令的实现而对复杂模型机电路进行的修改思路和修改操作。

4.2组成部分设计过程

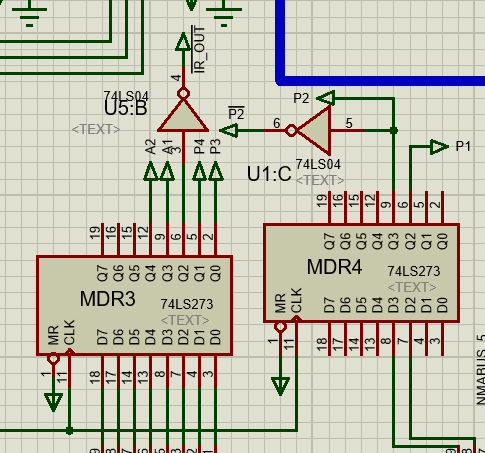
4.2.1 微程序控制器

1.**微指令存储器的扩充**：为了保证有我们的复杂模型机可以容纳更多的指令，我们对微指令的位数进行了扩充，这一操作我们采用增加了一个MROM实现，如图所示：

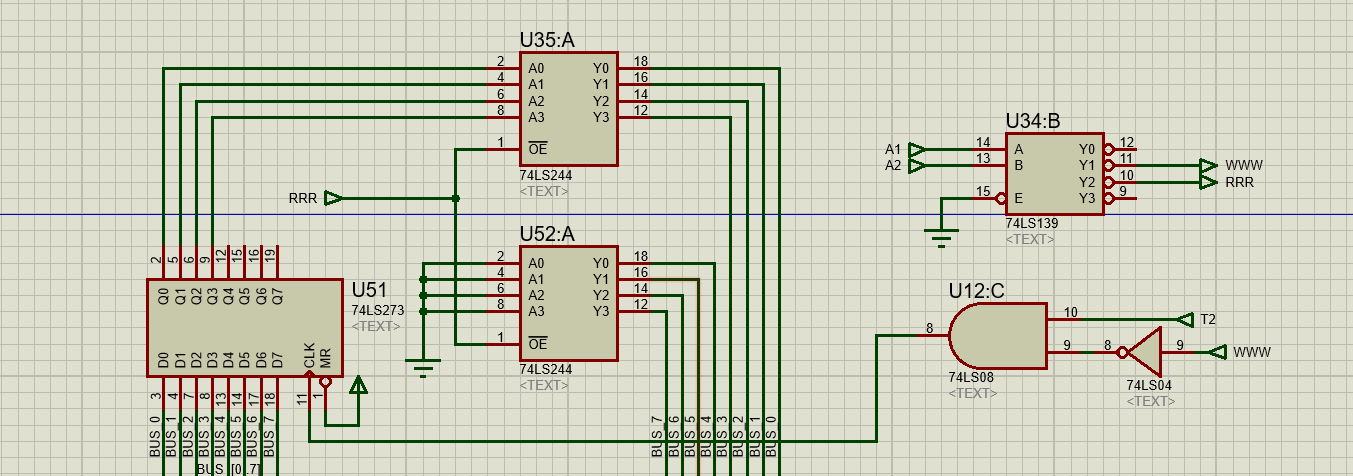
这样**微指令的位数就由24位变为了32位**，最多容纳的微指令条数也增加了。

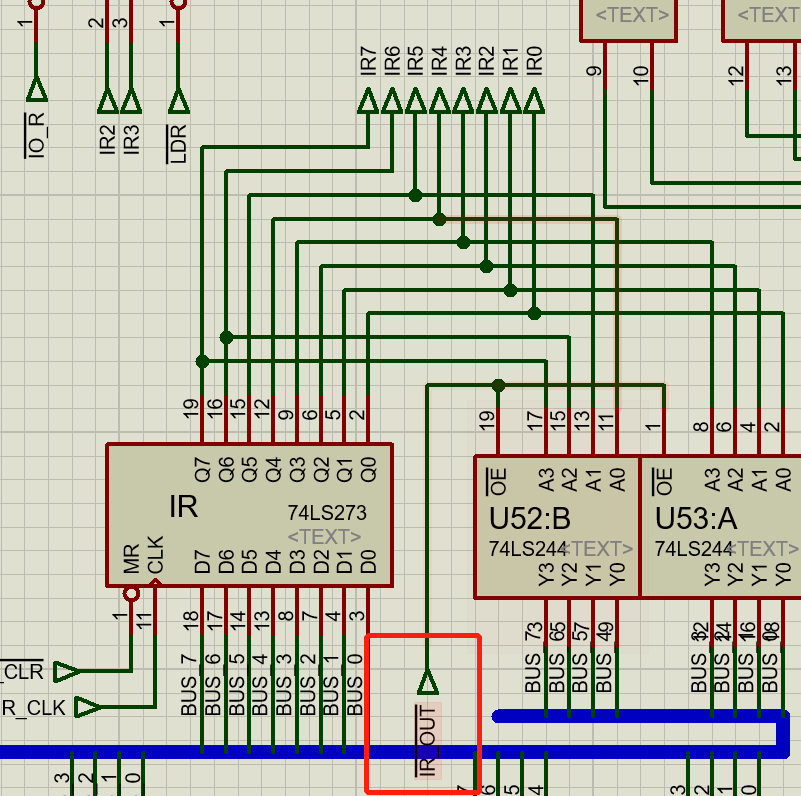
相应的，对于存放下址的74LS74也进行增加一个元件的修改方案以达到匹配的目的。

1. 微命令的改变：微指令通过74LS273进行译码，为了实现部分操作，我们对微命令进行了改变，即不同的微指令执行后对应的微操作发生了改变，其中前十二位的译码逻辑不变，这里不再叙述，仅对后十二位的译码逻辑进行讲解。



(1)**MDR3的19、16、15端口均对应保留字段，这一字段的空闲给予了复杂模型机实现更多操作的可能，体现了可扩充性。12、9号端口用于控制立即寻址暂存数据的暂存设施，暂存设施的设计如下**：



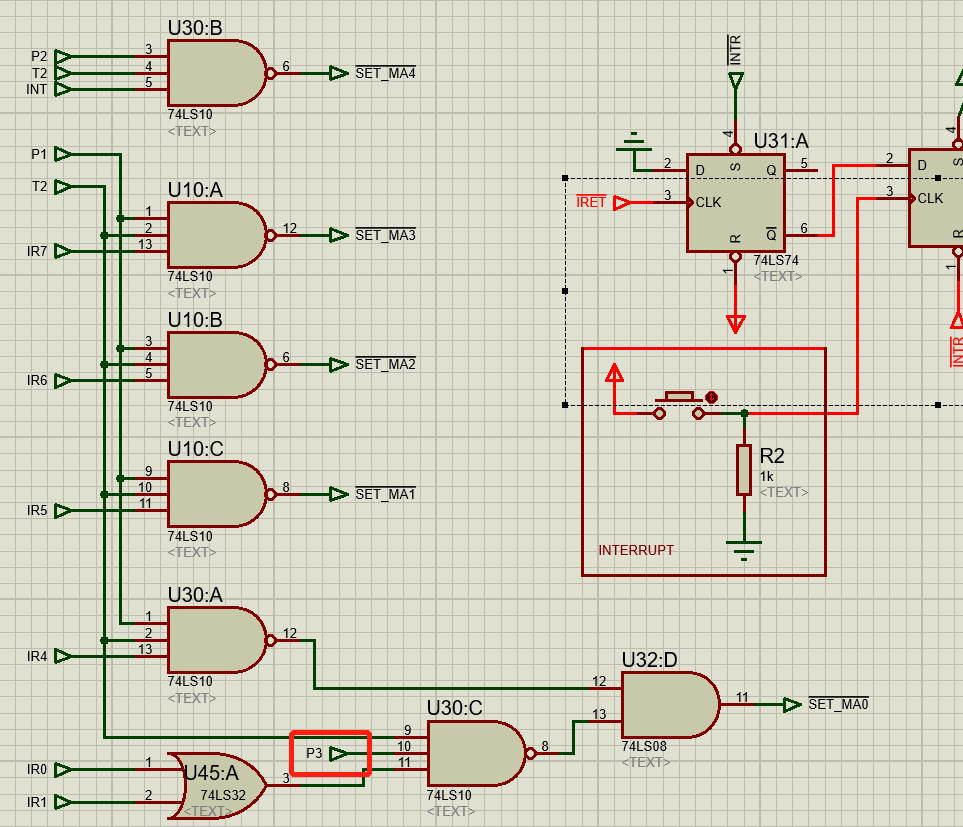


(2)**当A1、A2为01时，把指令的低四位数据读出，存入暂存器中。**

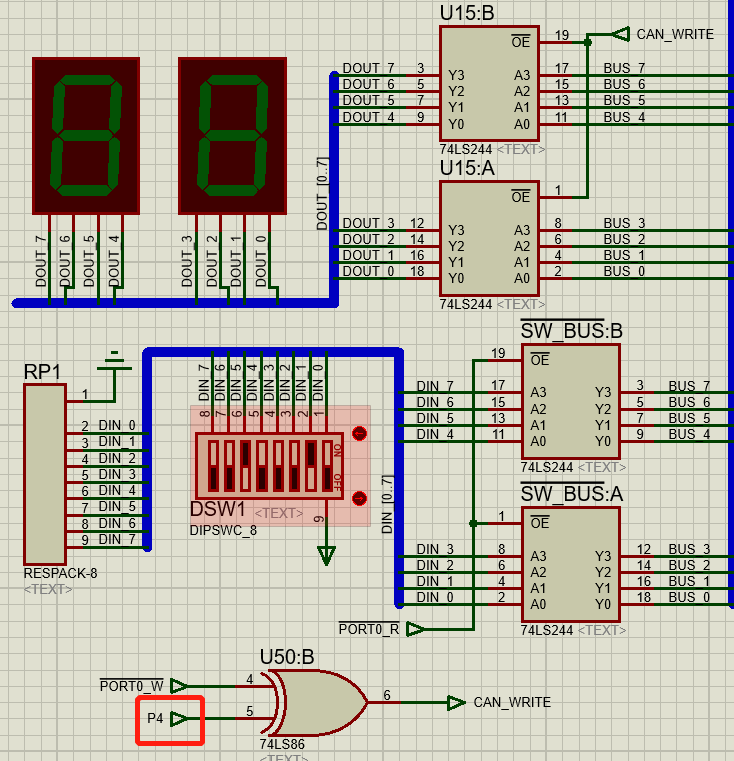
**A1、A2为10时，把暂存器中的数据读出、并将高四位置零**。

(3)**6号端口连接一个IR\_OUT,用于控制将IR中的数据输出到总线上，同样是由于我们发现没有这样的微指令所以设计而出，IR\_OUT的设置位置如下图所示。**

4)2、5对应着P3、P4字段，P3字段用于区分单字节指令和双字节指令，**P4则用于控制I/O指令的可行性。**其中P3字段的使用电路为：

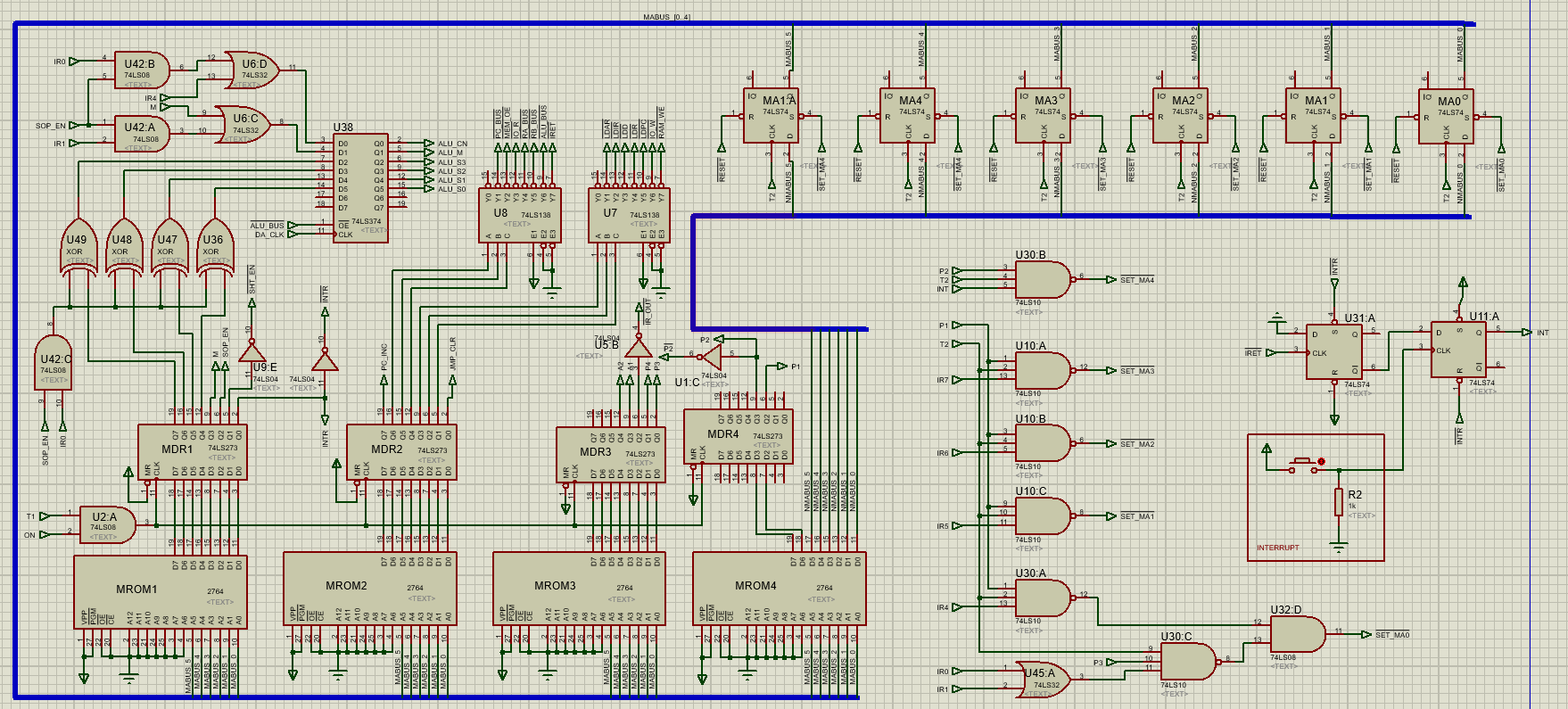


P4的使用电路为：



**当P4字段为1，且有W指令时，才允许IN指令，允许外来设备的数据写入到总线上，否则无法实现，这一操作增加了I/O设备与主设备之间的安全性。**

以上为全部对于电路的修改操作，controller部分的全部情况如下：



五、课程设计总结

本次实验我们首先实现了实验一的要求，搭建好了电路，实现了教材上的一系列操作，重新熟悉了各个元器件的功能以及指令的编写烧制过程。接着对复杂模型机进行了改造，首先通过增加一片EPROM的方式对微指令进行了扩展，可以同时进行更多的微操作，然后又进行了微地址的扩展，使得可以容纳更多条指令所需的微指令。

在搭建电路的过程中，我们遇到了许多问题，例如：无法从IR中取出数据，从而无法实现跳跃寻址以及其他的寻址方式，总线冲突、时钟不同步等问题，通过对电路的改造，加入缓冲器，调整时钟等方式逐一解决。

通过本次实验，我们掌握了基于微程序控制器的 CPU 组成结构，了解了CPU 的中断工作机制，熟悉了微指令的设计流程，掌握机器指令的微程序实现方法，修改电路、编写微指令的过程都是需要很大的耐心，付出很多时间才能完成，最终设计一套指令系统付出了很多心血，但电路的设计与搭建、对知识的掌握更加熟练，这一切都是值得的。

最后，感谢陈老师一个学期的耐心指导，在老师的指导下，顺利完成计算机组成原理实验和实践课程，对计组所学内容掌握得更加透彻，还学会了利用 Proteus 进行仿真测试，受益匪浅。

六、参考文献

1. CMA 组成原理与系统结构 ---说明

2. 基于 Proteus 的计算机系统实验教程