《计算机组成原理》实验报告

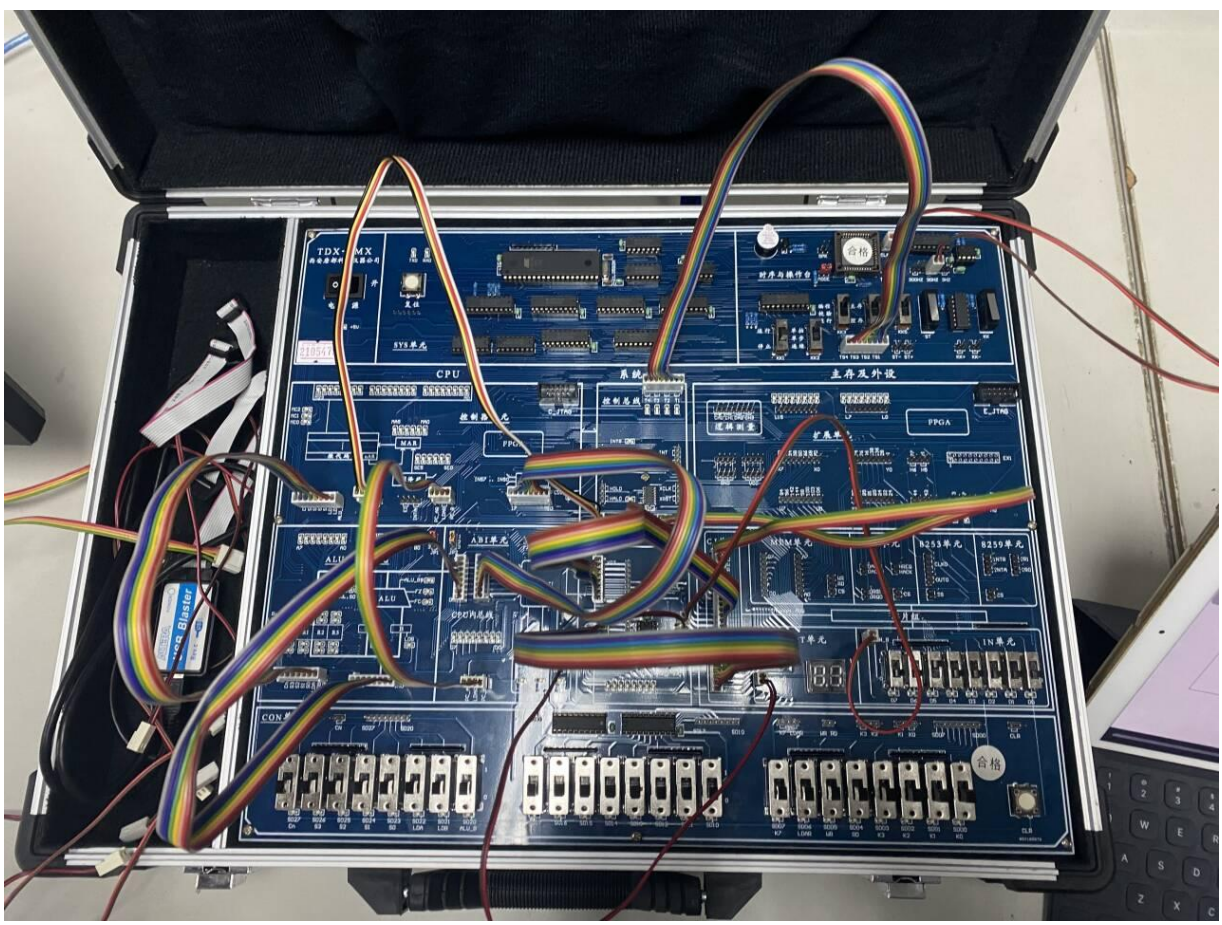
实验名称	5.3 复杂模型机设计实验	实验时间	2024 年 11 月 22 日

1 实验目的

综合运用所学计算机组成原理知识，设计并实现较为完整的计算机。

2 实验主要内容及过程

1. 把时序与操作台单元的“MODE”用短路块短接，使系统工作在四节拍模式，JP1、JP2 短路块均将 1、2 短接，按图 5-3-6 连接实验线路，仔细检查接线后打开实验箱电源。如图一所示：



图一　实验接线图

2. 写入实验程序，并进行校验，分两种方式，手动写入和联机写入。

1) 手动写入和校验

(1) 手动写入微程序

① 将时序与操作台单元的开关 KK1 置为‘停止’档, KK3 置为‘编程’档, KK4 置为‘控存’档，KK5 置为‘置数’档。

② 使用 CON 单元的 SD15——SD10 给出微地址，IN 单元给出低 8 位应写入的数据，连续两次按动时序与操作台的开关 ST，将 IN 单元的数据写到该单元的低 8 位。

③ 将时序与操作台单元的开关 KK5 置为‘加 1’档。

④ IN 单元给出中 8 位应写入的数据，连续两次按动时序与操作台的开关 ST，将 IN 单元的数据写到该单元的中 8 位。IN 单元给出高 8 位应写入的数据，连续两次按动时序与操作台的开关 ST，将 IN 单元的数据写到该单元的高 8 位。

⑤ 重复①、②、③、④四步，将表 5-3-5 的微代码写入 E2ROM 芯片中。

(2) 手动校验微程序

① 将时序与操作台单元的开关 KK1 置为‘停止’档，KK3 置为‘校验’档，KK4 置为‘控

存’档，KK5 置为‘置数’档。

② 使用 CON 单元的 SD15——SD10 给出微地址，连续两次按动时序与操作台的开关 ST，MC 单元的指数据指示灯 M7——M0 显示该单元的低 8 位。

③ 将时序与操作台单元的开关 KK5 置为‘加 1’档。

④ 连续两次按动时序与操作台的开关 ST，MC 单元的指数据指示灯 M15——M8 显示该单元的中 8 位，MC 单元的指数据指示灯 M23——M16 显示该单元的高 8 位。

⑤ 重复①、②、③、④四步，完成对微代码的校验。如果校验出微代码写入错误，重新写入、校验，直至确认微指令的输入无误为止。

(5) 手动写入机器程序

① 将时序与操作台单元的开关 KK1 置为‘停止’档，KK3 置为‘编程’档，KK4 置为‘主存’档，KK5 置为‘置数’档。

② 使用 CON 单元的 SD17——SD10 给出地址，IN 单元给出该单元应写入的数据，连续两次按动时序与操作台的开关 ST，将 IN 单元的数据写到该存储器单元。

③ 将时序与操作台单元的开关 KK5 置为‘加 1’档。

④ IN 单元给出下一地址（地址自动加 1）应写入的数据，连续两次按动时序与操作台的开关 ST，将 IN 单元的数据写到该单元中。然后地址会又自加 1，只需在 IN 单元输入后续地址的数据，连续两次按动时序与操作台的开关 ST，即可完成对该单元的写入。

⑤ 亦可重复①、②两步，将所有机器指令写入主存芯片中。

(6) 手动校验机器程序

① 将时序与操作台单元的开关 KK1 置为‘停止’档，KK3 置为‘校验’档，KK4 置为‘主存’档，KK5 置为‘置数’档。

② 使用 CON 单元的 SD17——SD10 给出地址，连续两次按动时序与操作台的开关 ST，CPU 内总线的指数据指示灯 D7——D0 显示该单元的数据。

③ 将时序与操作台单元的开关 KK5 置为‘加 1’档。

④ 连续两次按动时序与操作台的开关 ST，地址自动加 1，CPU 内总线的指数据指示灯 D7——D0 显示该单元的数据。此后每两次按动时序与操作台的开关 ST，地址自动加 1，CPU 内总线的指数据指示灯 D7——D0 显示该单元的数据，继续进行该操作，直至完成校验，如发现错误，则返回写入，然后校验，直至确认输入的所有指令准确无误。

⑤ 亦可重复①、②两步，完成对指令码的校验。如果校验出指令码写入错误，重新写入、校验，直至确认指令的输入无误为止。

2) 联机写入和校验

联机软件提供了微程序和机器程序下载功能，以代替手动读写微程序和机器程序，但是微程序和机器程序得以指定的格式写入到以 TXT 为后缀的文件中，本次实验程序如下，程序中分号‘；’为注释符，分号后面的内容在下载时将被忽略掉。

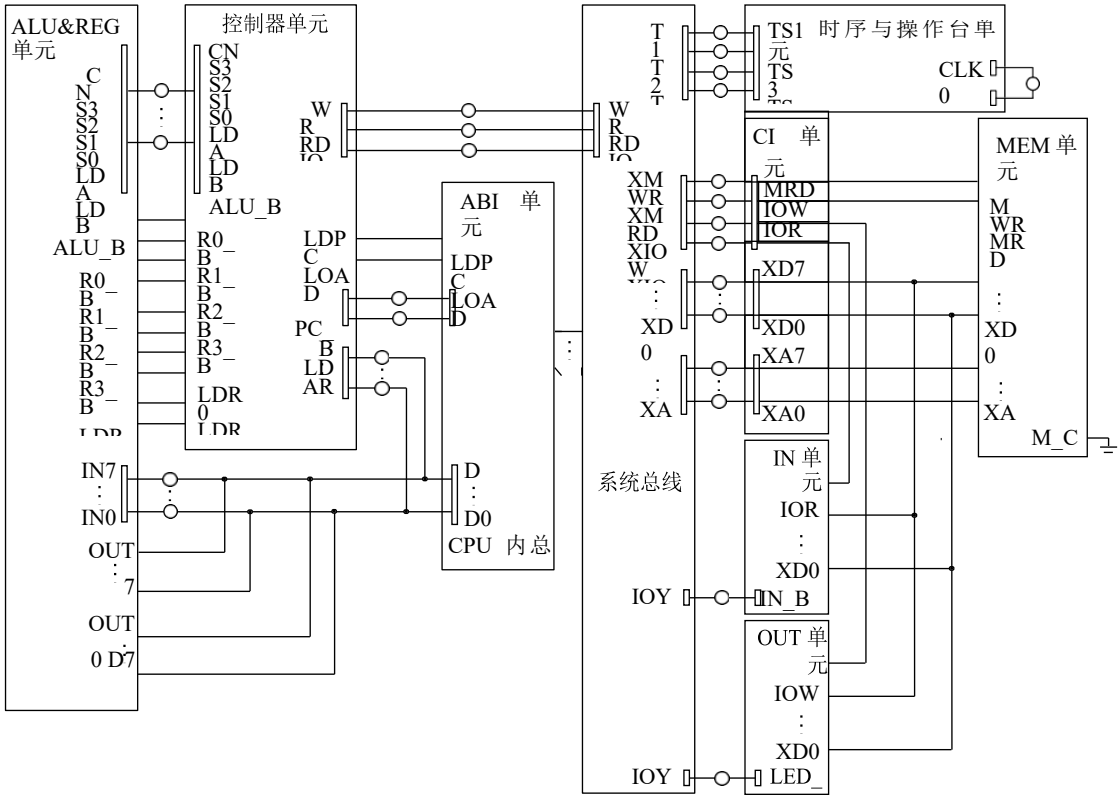

ALU&REG
单元
控制器单元
ALU_B
ABI 单元
CPU 内总
系统总线
TS1 时序与操作台单元
CI 单元
MEM 单元
IN 单元
OUT 单元
CLK
M_C
IOY
IN_B
LED_

图 5-3-6　实验接线图

```
; //*************************************//
; //                                     //
; //      复杂模型机实验指令文件         //
; //                                     //
; //      By TangDu CO.,LTD              //
; //                                     //
; //*************************************//

; //****** Start Of Main Memory Data ******//
  $P 00 20    ; START: IN R0,00H      从IN单元读入计数初值
  $P 01 00
  $P 02 61    ; LDI  R1,0FH           立即数 0FH送R1
  $P 03 0F
  $P 04 14    ; AND  R0,R1            得到R0低四位
  $P 05 61    ; LDI  R1,00H           装入和初值 00H
  $P 06 00
  $P 07 F0    ; BZC  RESULT           计数值为 0则跳转
  $P 08 16
  $P 09 62    ; LDI  R2,60H           读入数据始地址
  $P 0A 60
  $P 0B CB    ; LOOP: LAD R3,[RI],00H     从MEM读入数据送 R3，
                                      变址寻址，偏移量为 00H
  $P 0C 00
  $P 0D 0D    ; ADD  R1,R3            累加求和
  $P 0E 72    ; INC  RI               变址寄存加 1，指向下一数据
  $P 0F 63    ; LDI  R3,01H           装入比较值
  $P 10 01
  $P 11 8C    ; SUB  R0,R3
  $P 12 F0    ; BZC  RESULT           相减为 0，表示求和完毕
  $P 13 16
  $P 14 E0    ; JMP  LOOP             未完则继续
  $P 15 0B
  $P 16 D1    ; RESULT: STA  70H,R1   和存于 MEM的70H单元
  $P 17 70
  $P 18 34    ; OUT  40H,R1           和在 OUT单元显示
  $P 19 40
  $P 1A E0    ; JMP START             跳转至 START
  $P 1B 00
  $P 1C 50    ; HLT                   停机

  $P 60 01    ; 数据
  $P 61 02
  $P 62 03
  $P 63 04
  $P 64 05
  $P 65 06
  $P 66 07
  $P 67 08
  $P 68 09
  $P 69 0A
  $P 6A 0B
  $P 6B 0C
```

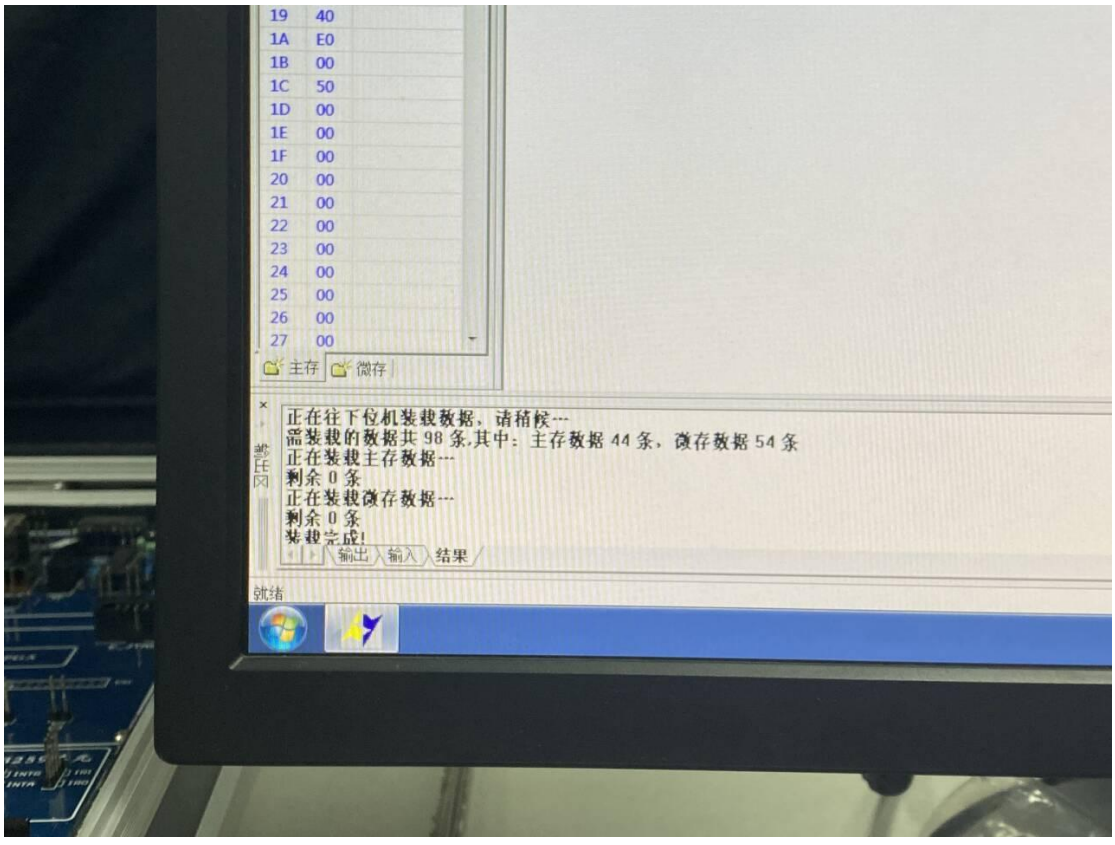
```
  $P 6C 0D
  $P 6D 0E
  $P 6E 0F
; //***** End Of Main Memory Data *****//

; //** Start Of MicroController Data **//
  $M 00 000001    ; NOP
  $M 01 006D43    ; PC->AR, PC加1
  $M 03 107070    ; MEM->IR, P<1>
  $M 04 002405    ; RS->B
  $M 05 04B201    ; A加 B->RD
  $M 06 002407    ; RS->B
  $M 07 013201    ; A与 B->RD
  $M 08 106009    ; MEM->AR
  $M 09 183001    ; IO->RD
  $M 0A 106010    ; MEM->AR
  $M 0B 000001    ; NOP
  $M 0C 103001    ; MEM->RD
  $M 0D 200601    ; RD->MEM
  $M 0E 005341    ; A->PC
  $M 0F 0000CB    ; NOP, P<3>
  $M 10 280401    ; RS->IO
  $M 11 103001    ; MEM->RD
  $M 12 06B201    ; A加1->RD
  $M 13 002414    ; RS->B
  $M 14 05B201    ; A减 B->RD
  $M 15 002416    ; RS->B
  $M 16 01B201    ; A或 B->RD
  $M 17 002418    ; RS->B
  $M 18 02B201    ; A右环移->RD
  $M 1B 005341    ; A->PC
  $M 1C 10101D    ; MEM->A
  $M 1D 10608C    ; MEM->AR, P<2>
  $M 1E 10601F    ; MEM->AR
  $M 1F 101020    ; MEM->A
  $M 20 10608C    ; MEM->AR, P<2>
  $M 28 101029    ; MEM->A
  $M 29 00282A    ; RI->B
  $M 2A 04E22B    ; A加 B->AR
  $M 2B 04928C    ; A加 B->A, P<2>
  $M 2C 10102D    ; MEM->A
  $M 2D 002C2E    ; PC->B
  $M 2E 04E22F    ; A加 B->AR
  $M 2F 04928C    ; A加 B->A, P<2>
  $M 30 001604    ; RD->A
  $M 31 001606    ; RD->A
  $M 32 006D48    ; PC->AR, PC加1
  $M 33 006D4A    ; PC->AR, PC加1
  $M 34 003401    ; RS->RD
  $M 35 000035    ; NOP
  $M 36 006D51    ; PC->AR, PC加1
  $M 37 001612    ; RD->A
  $M 38 001613    ; RD->A
  $M 39 001615    ; RD->A
  $M 3A 001617    ; RD->A
```

```
  $M 3B 000001    ; NOP
  $M 3C 006D5C    ; PC->AR, PC加1
  $M 3D 006D5E    ; PC->AR, PC加1
  $M 3E 006D68    ; PC->AR, PC加1
  $M 3F 006D6C    ; PC->AR, PC加1
; //** End Of MicroController Data **//
```

选择联机软件的“【转储】—【装载】”功能，在打开文件对话框中选择上面所保存的文件，软件自动将机器程序和微程序写入指定单元。

选择联机软件的“【转储】—【刷新指令区】”可以读出下位机所有的机器指令和微指令， 并在指令区显示，对照文件检查微程序和机器程序是否正确，如果不正确，则说明写入操作失败，应重新写入，可以通过联机软件单独修改某个单元的指令，以修改微指令为例，先用鼠标左键单击指令区的‘微存’TAB 按钮，然后再单击需修改单元的数据，此时该单元变为编辑框，输入6 位数据并回车，编辑框消失，并以红色显示写入的数据。如图二所示：



图二　指令装载图

3. 运行程序

方法一：本机运行

将时序与操作台单元的开关 KK1、KK3 置为‘运行’档，按动 CON 单元的总清按钮 CLR，将使程序计数器 PC、地址寄存器 AR 和微程序地址为 00H，程序可以从头开始运行，暂存器 A、B，指令寄存器 IR 和OUT 单元也会被清零。

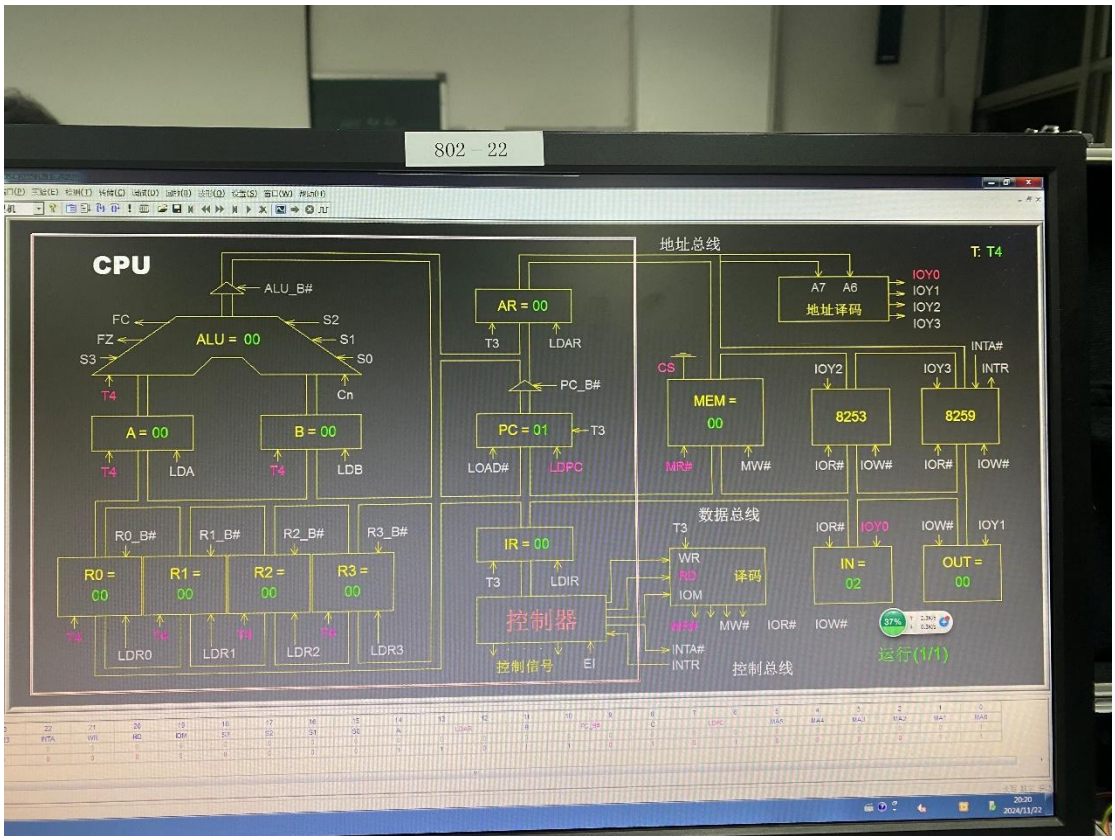
将时序与操作台单元的开关 KK2 置为‘单步’档，每按动一次 ST 按钮，即可单步运行一条微指令，对照微程序流程图，观察微地址显示灯是否和流程一致。每运行完一条微指令，观测一次数据总线和地址总线，对照数据通路图，分析总线上的数据是否正确。

当模型机执行完 OUT 指令后，检查 OUT 单元显示的数是否正确，按下 CON 单元的总清按钮CLR，改变 IN 单元的值，再次执行机器程序，从 OUT 单元显示的数判别程序执行是否正确。

方法二：联机运行（软件使用说明请看附录　　1）

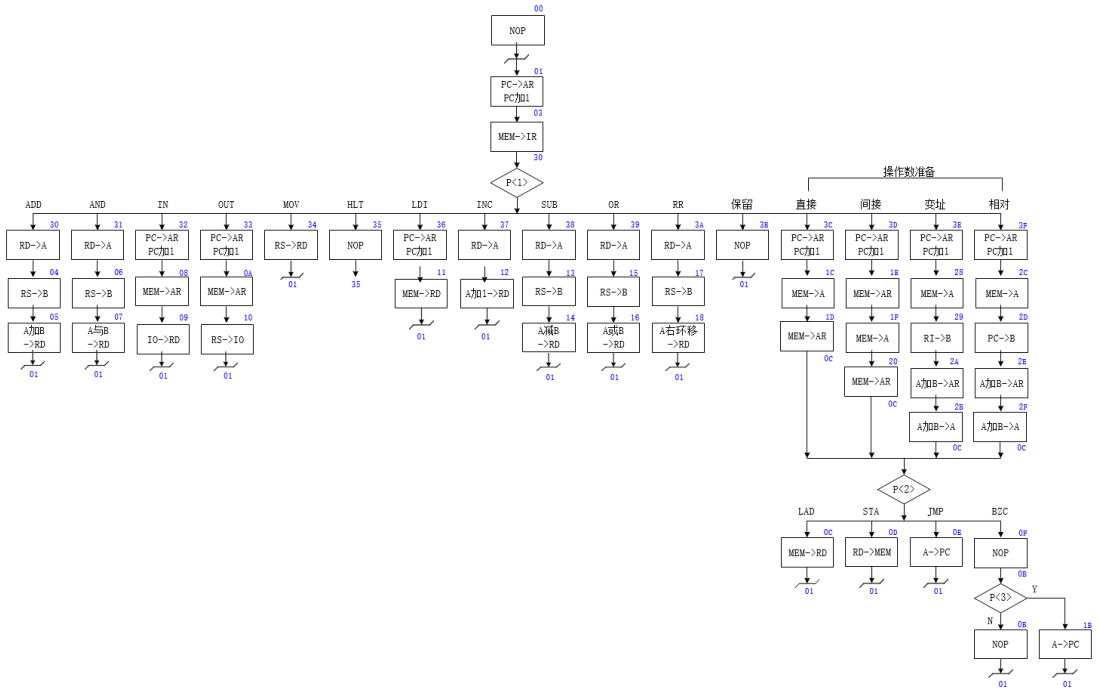
进入软件界面，选择菜单命令“【实验】—【CISC　实验】”，打开相应的数据通路图，选择相应的功能命令，即可联机运行、监控、调试程序。

按动CON 单元的总清按钮 CLR，然后通过软件运行程序，当模型机执行完 OUT 指令后，检查OUT 单元显示的数是否正确。在数据通路图和微程序流中观测指令的执行过程，并观测软件中地址总线、数据总线以及微指令显示和下位机是否一致。如图三所示：



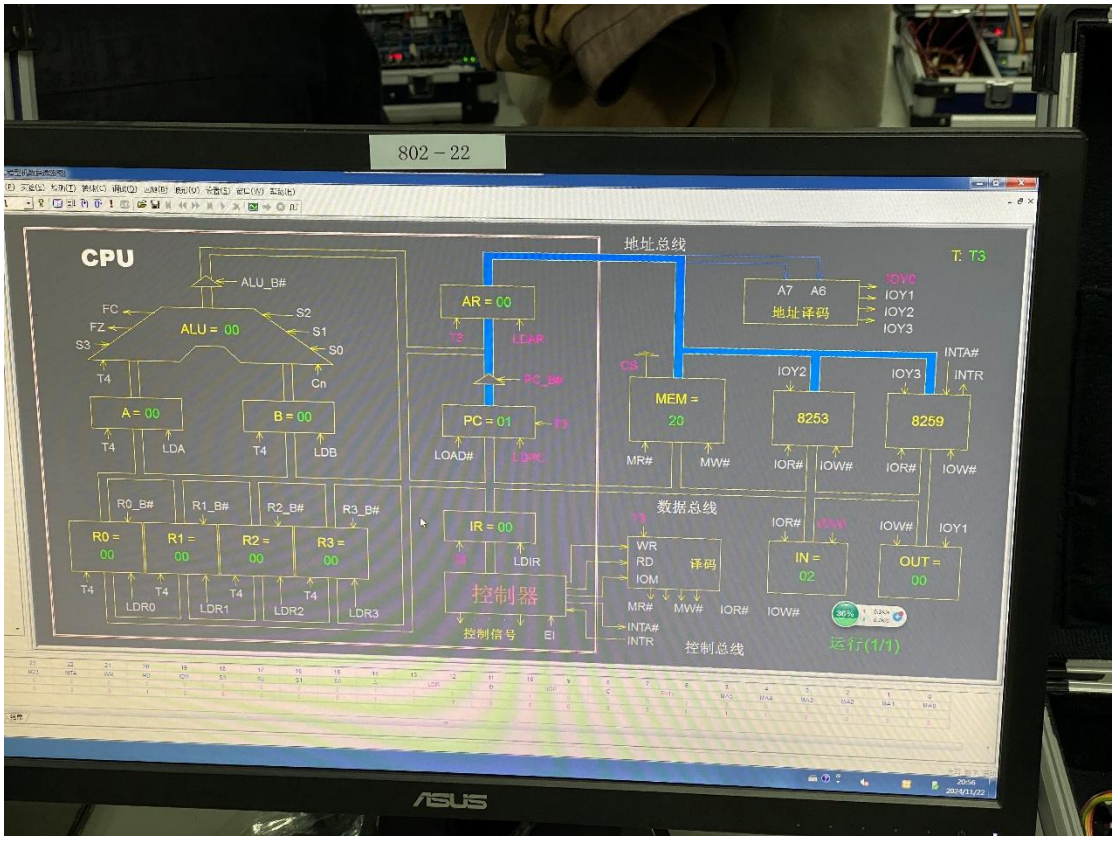
图三　数据通路图

根据图四所示，我们可以了解到CISC的基本工作流程：



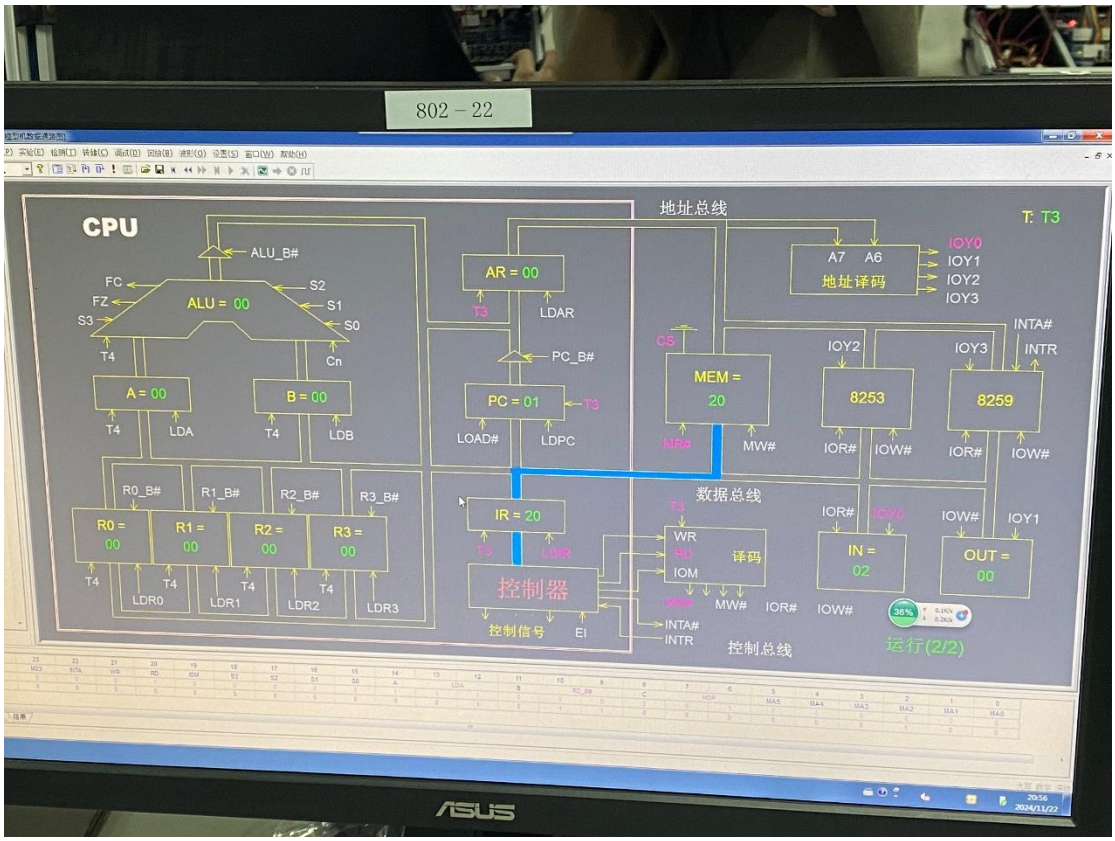
图四　CISC 工作流程图

简而言之，PC 计数器在每个周期赋值给 AR 寄存器，随后自增。正如之前图三所示。AR 寄存器从自身被写入的数据去寻找对应的指令操作码送入 MEM 寄存器，如图五所示：



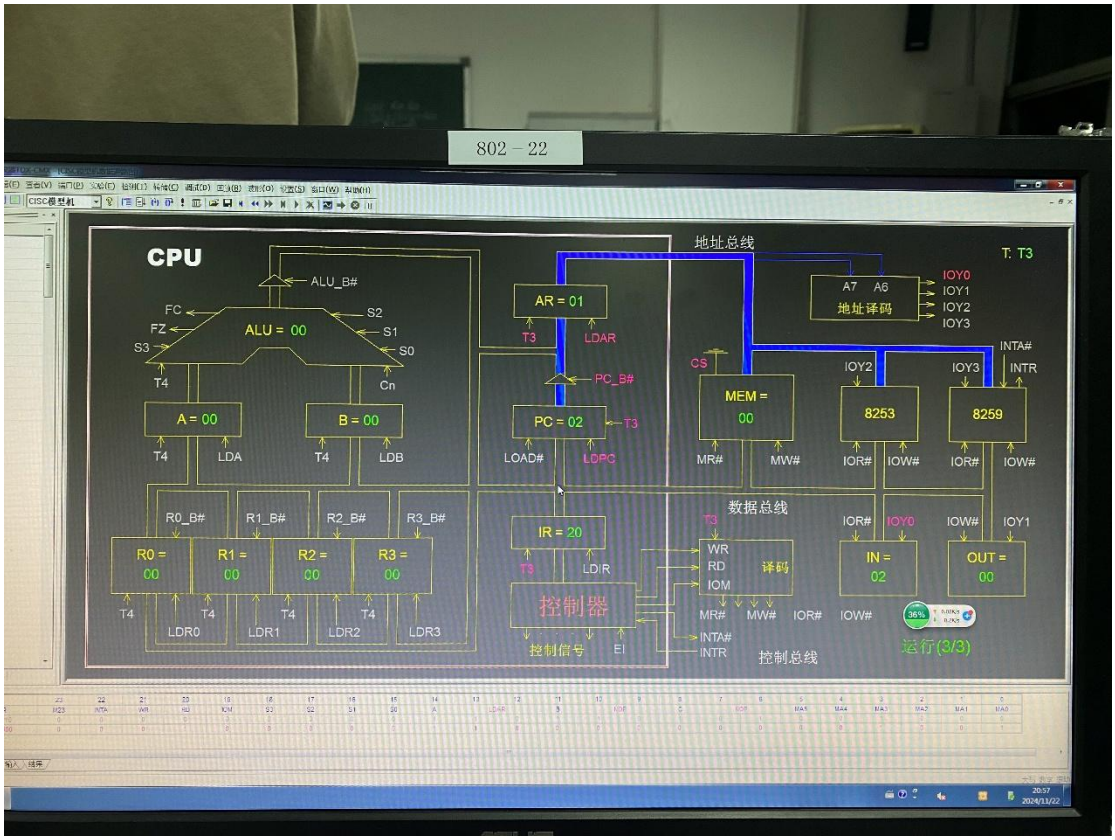
图五　数据通路图

MEM 寄存器随后将操作码送入 IR 寄存器。如图六所示：



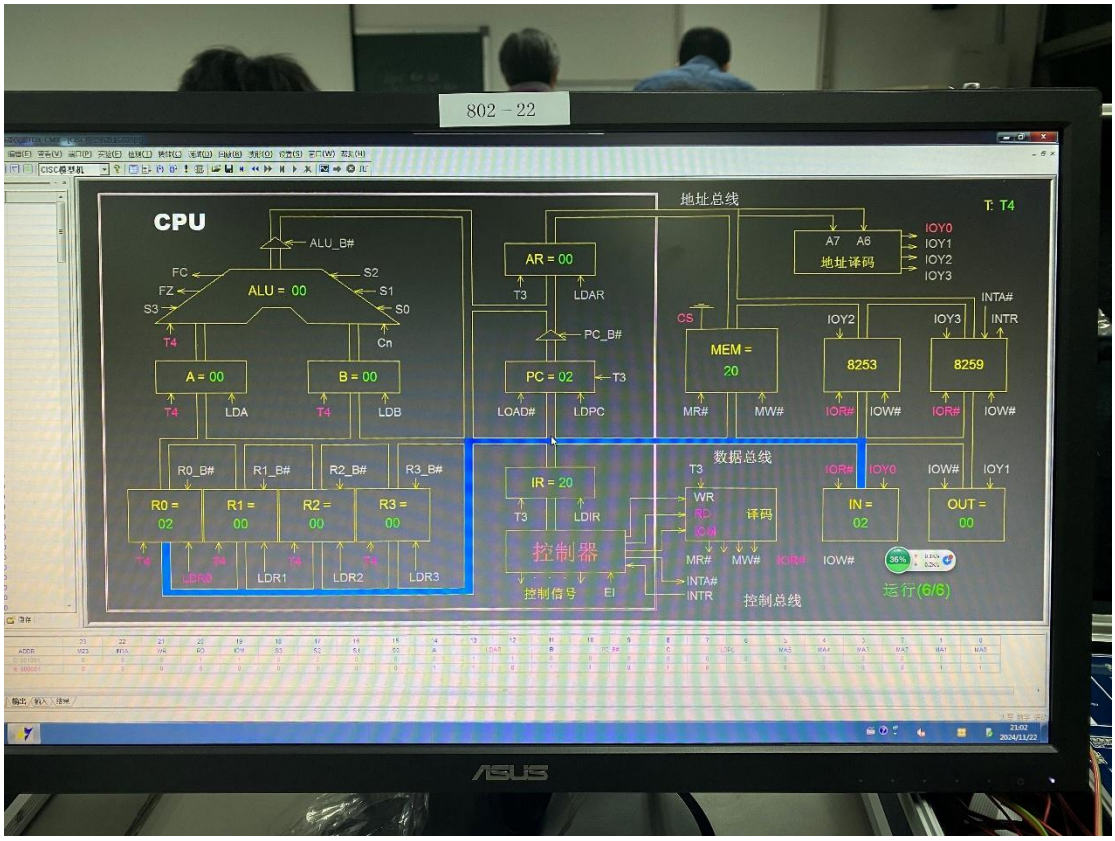
图六　数据通路图

随后，PC 赋值给 AR，继续下一时钟周期，如图七所示：



图七　数据通路图

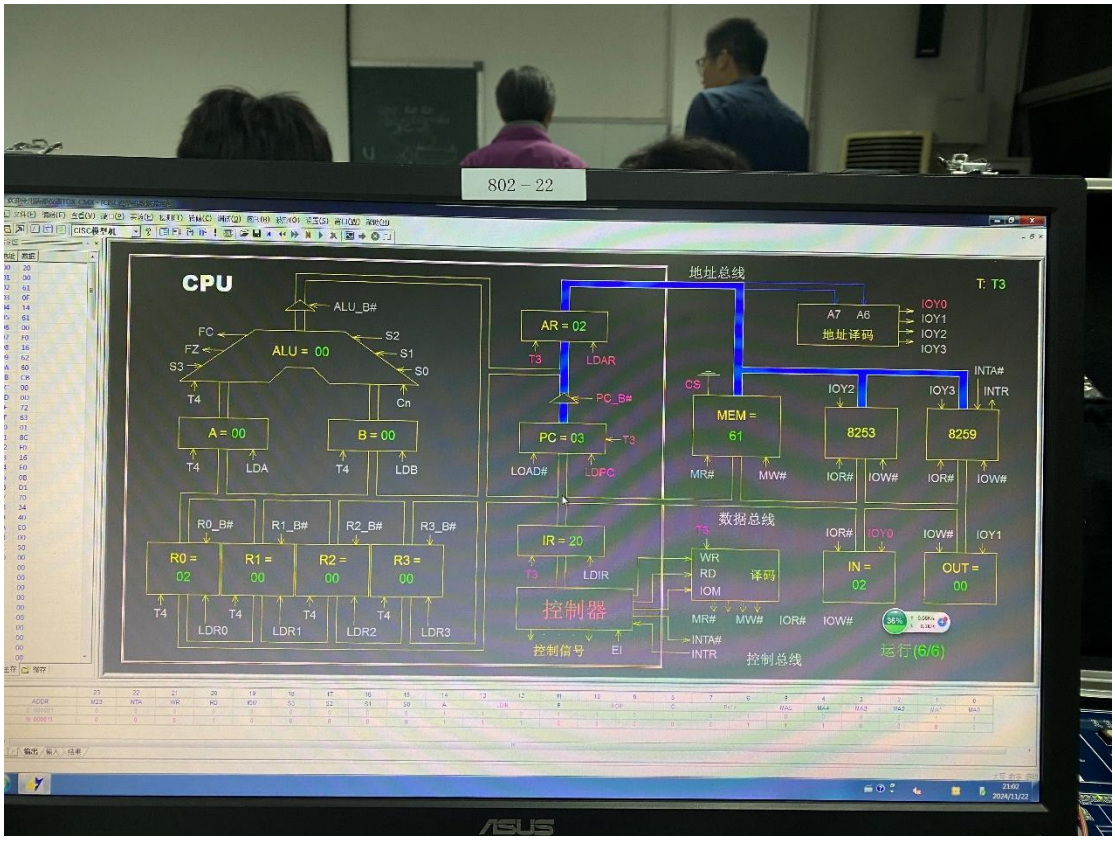
IR 内部指令为 20，意为将数据从 IN 读入计数初值 R0。如图八所示：



图八　数据通路图

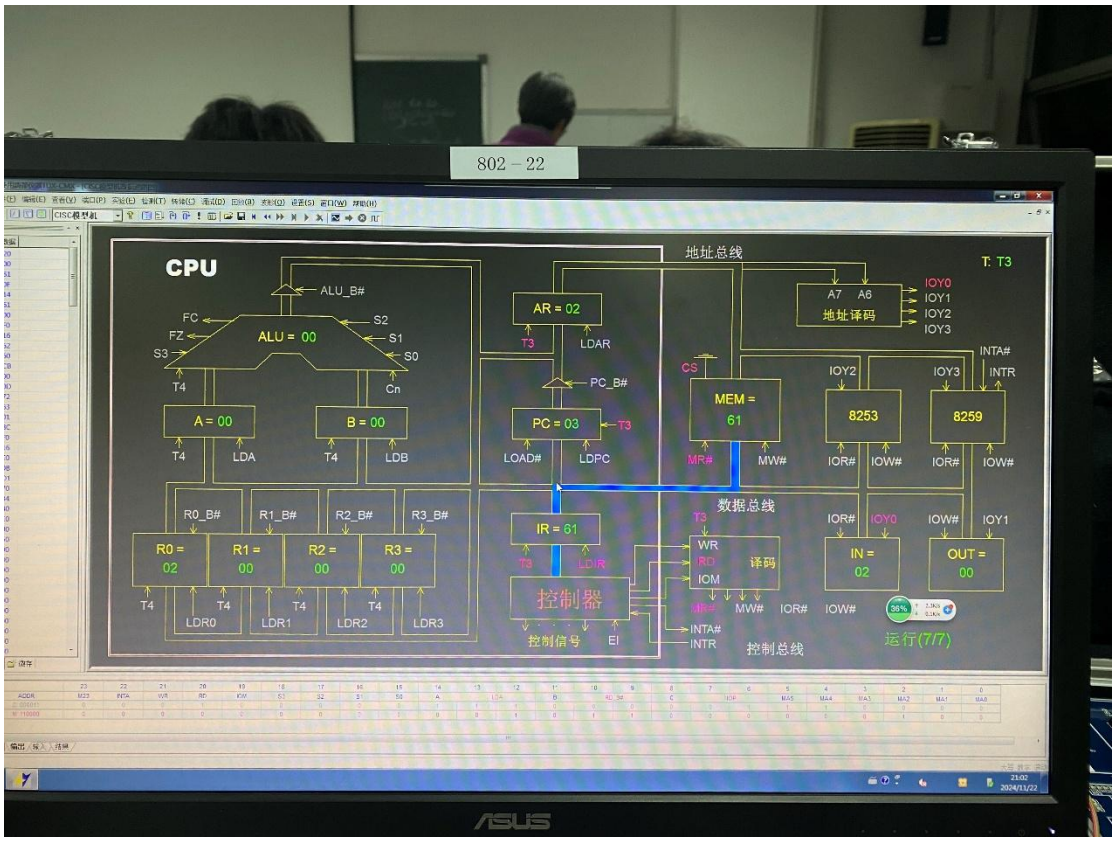
我们可以观察到数据确实从 IN 进入了寄存器 R0。

随后读出指令 61 如图九所示：



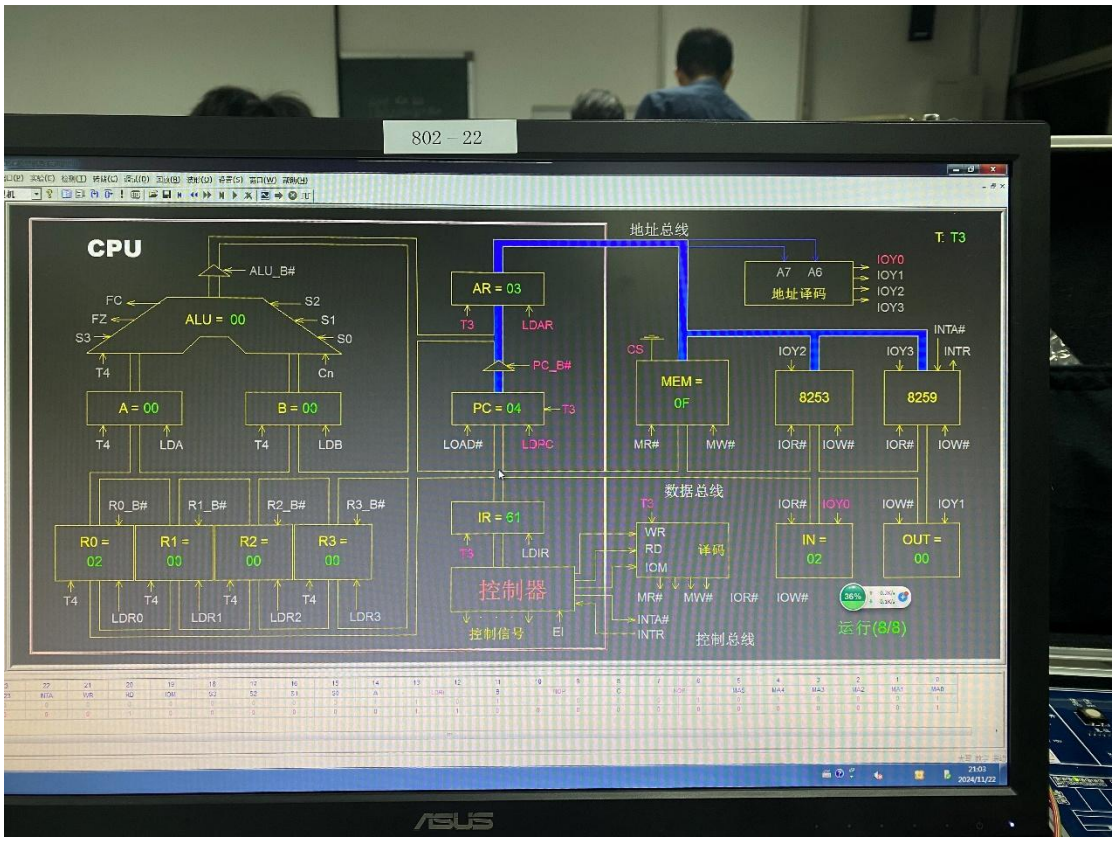
图九　数据通路图

指令被装载：



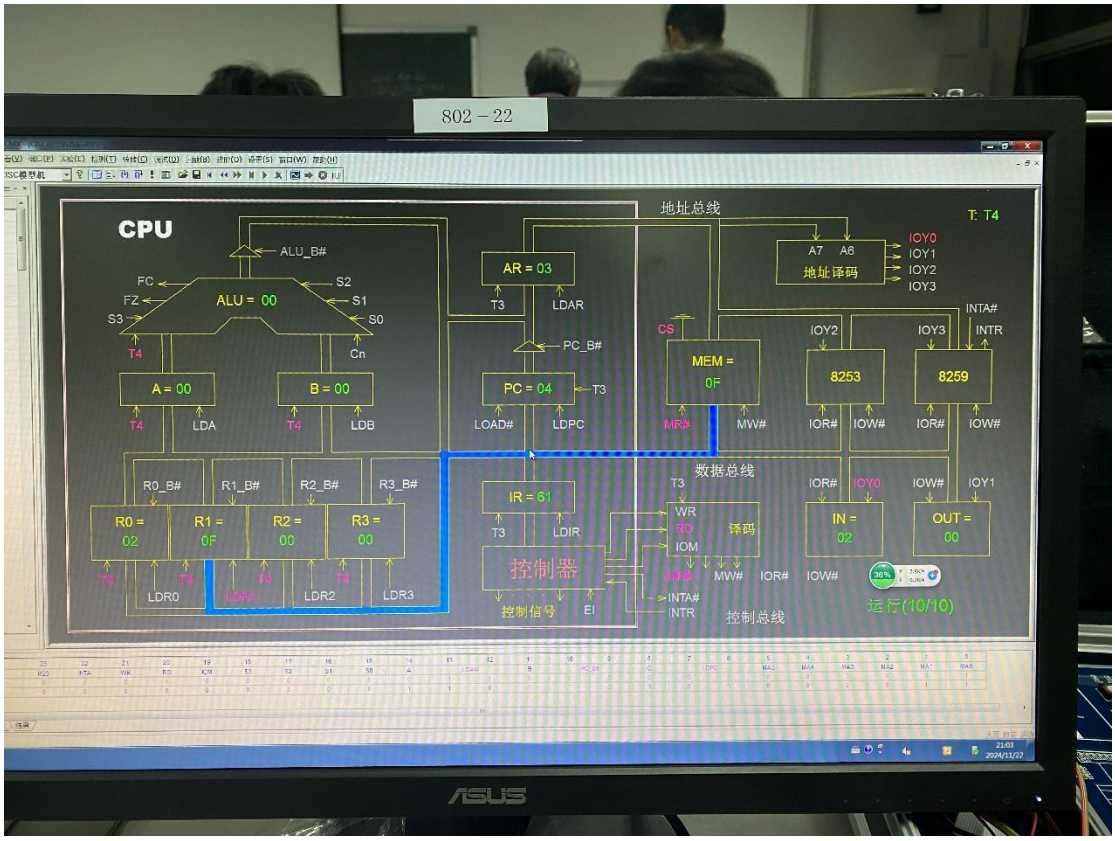
图十　数据通路图

这个指令需要一个立即数 0FH，他从下一条指令装载：



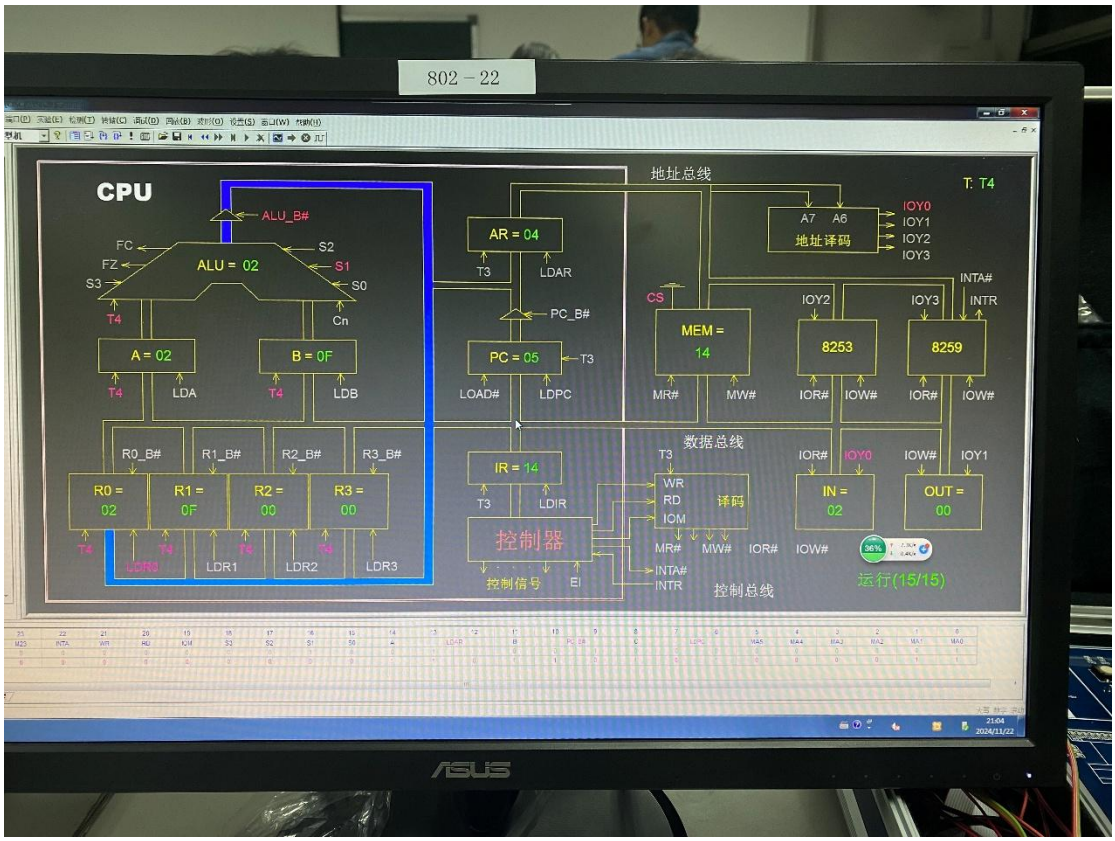
图十一　数据通路图

这个立即数被送入寄存器，如图十二所示：



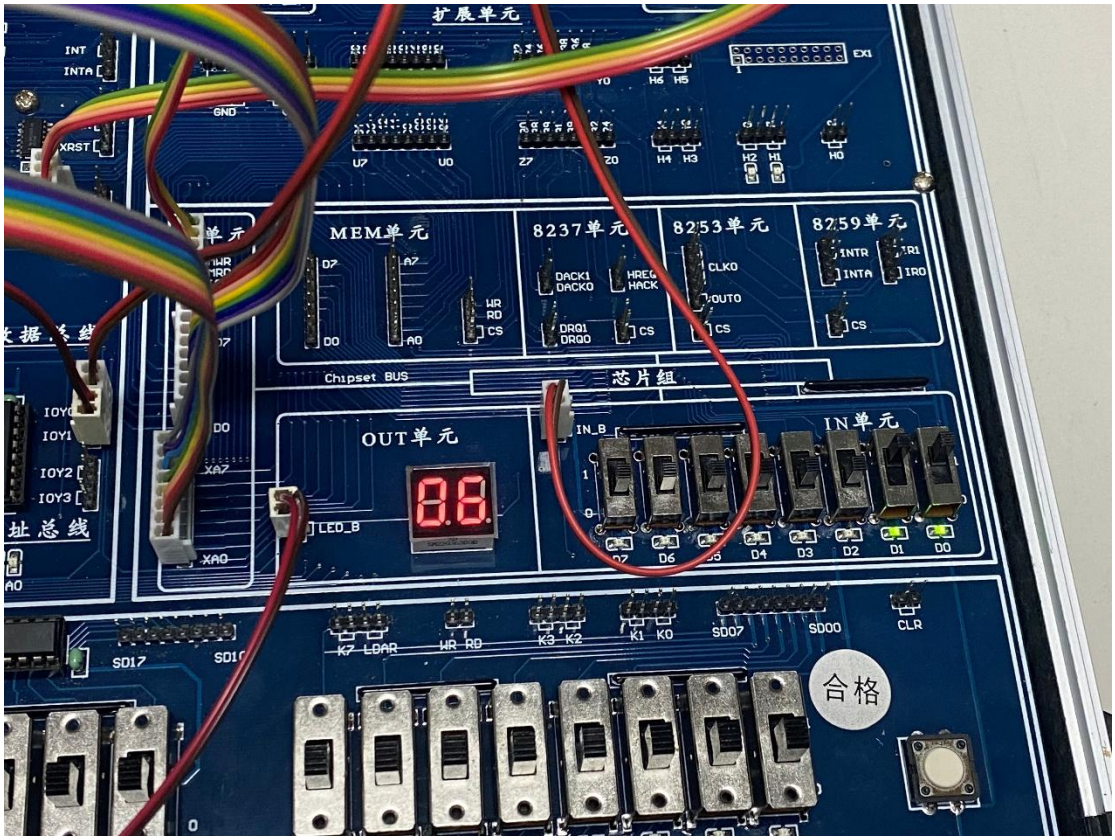
图十二　数据通路图

这个数是用于与运算，以取得数低四位的值。取指过程略过，如图十三所示：

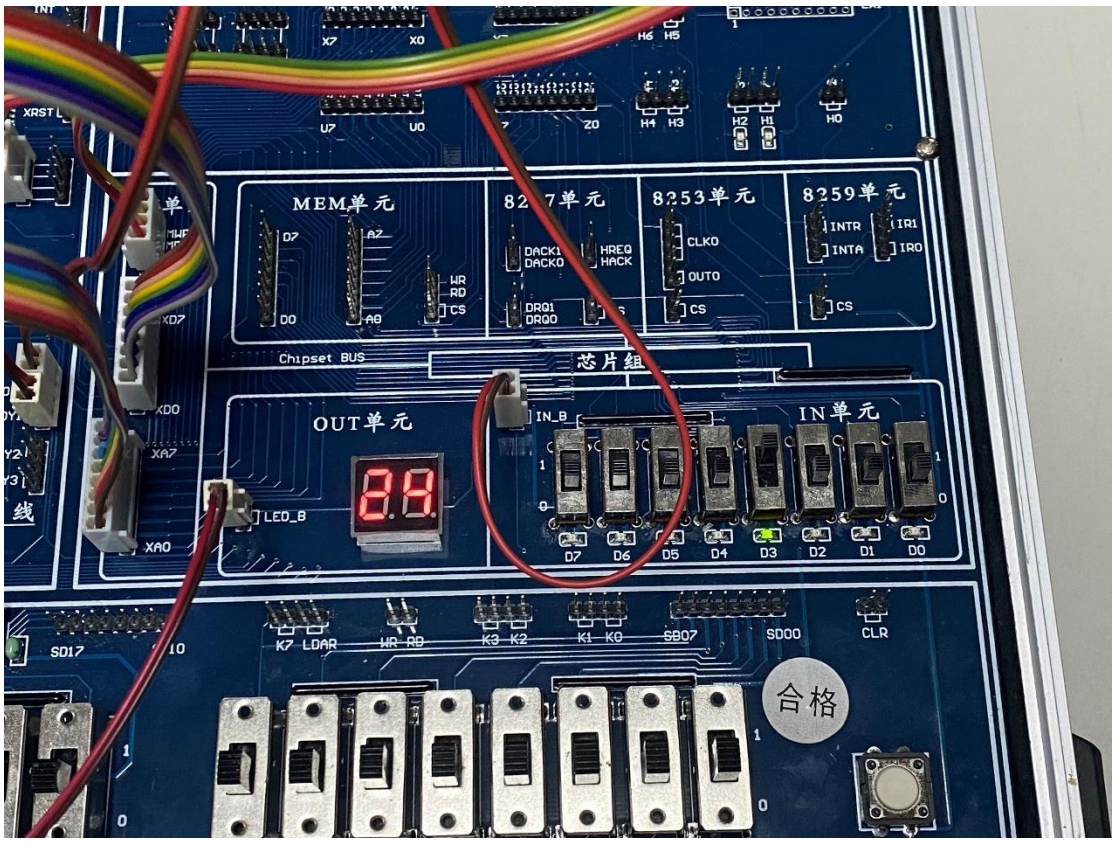


图十三　数据通路图

由于整个过程较长，故不再一一展示。接下来展示结果，如图十四十五所示：



图十四　结果图



图十五　结果图

整个程序接受一个数 n，取后四位，一直加比 n 小 1 的数，比 n 小 2 的数，一直到下一个数为 1 为止。

3 实验小结

在本次实验中，我们深入研究了指令系统的手动编程与执行方法，并了解了其在简单计算模型中的具体应用。通过实验操作，我掌握了指令系统的基本组成结构，包括如何通过寄存器单元、控制单元和内存单元的协同工作实现指令的加载、执行与结果校验。

实验的关键在于熟练掌握机器指令的编制和加载过程。在加载机器指令时，我们结合指令格式的操作码与操作数结构，逐条将指令输入到存储单元中，并利用寄存器的状态灯进行实时校验。通过观察操作数的传递路径与指令执行结果，确保指令的输入和逻辑操作准确无误。如果检测到异常，我们重新检查地址字段与操作数配置，及时更正并验证，直至所有指令正确完成加载。

我们还探索了数据加载与结果验证的过程。实验中，通过模型机的操作台开关与指令组合方式，我们实现了从输入单元读取数据、执行数据运算，并将结果存储到指定内存单元的操作。利用控制信号灯的状态变化，我们能够快速确认指令执行的状态，同时通过存储器数据校验方法验证运算结果的正确性。

在程序运行阶段，我们使用了单步执行与自动时钟发生器全速运行两种模式。通过逐步执行指令并对比程序流程图，我们观察到了寄存器、数据总线和地址总线之间的交互过程，进一步加深了对模型机中时序控制信号的理解。特别是在观察寄存器中数据变化的过程中，我体会到时序控制对于协调各个单元工作的重要性。同时，通过自动运行模式，我们验证了整个程序的完整性和执行效率。

总体而言，本次实验加深了我对简单计算模型的指令设计、加载和运行原理的理解，并让我熟悉了在硬件环境中通过手动方式编程、校验和调试的方法。这次实验不仅提升了我对模型机中指令系统各项原理的掌握，特别是在数据传输与信号控制方面的操作，还让我对其在基本计算任务中的实际应用有了更全面的认识。