《计算机组成原理》实验报告

实验名称	CPU 与简单模型机设计实验	实验时间	2024 年 11 月 08 日

1 实验目的

(1) 掌握一个简单 CPU 的组成原理。
(2) 在掌握部件单元电路的基础上，进一步将其构造一台基本模型计算机。
(3) 为其定义五条机器指令，编写相应的微程序，并上机调试掌握整机概念。

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 用短路块均将 1、2 短接，按图 5-1-5 连接实验线路。

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 5-1-2 的微代码写入 E2ROM 芯片中。

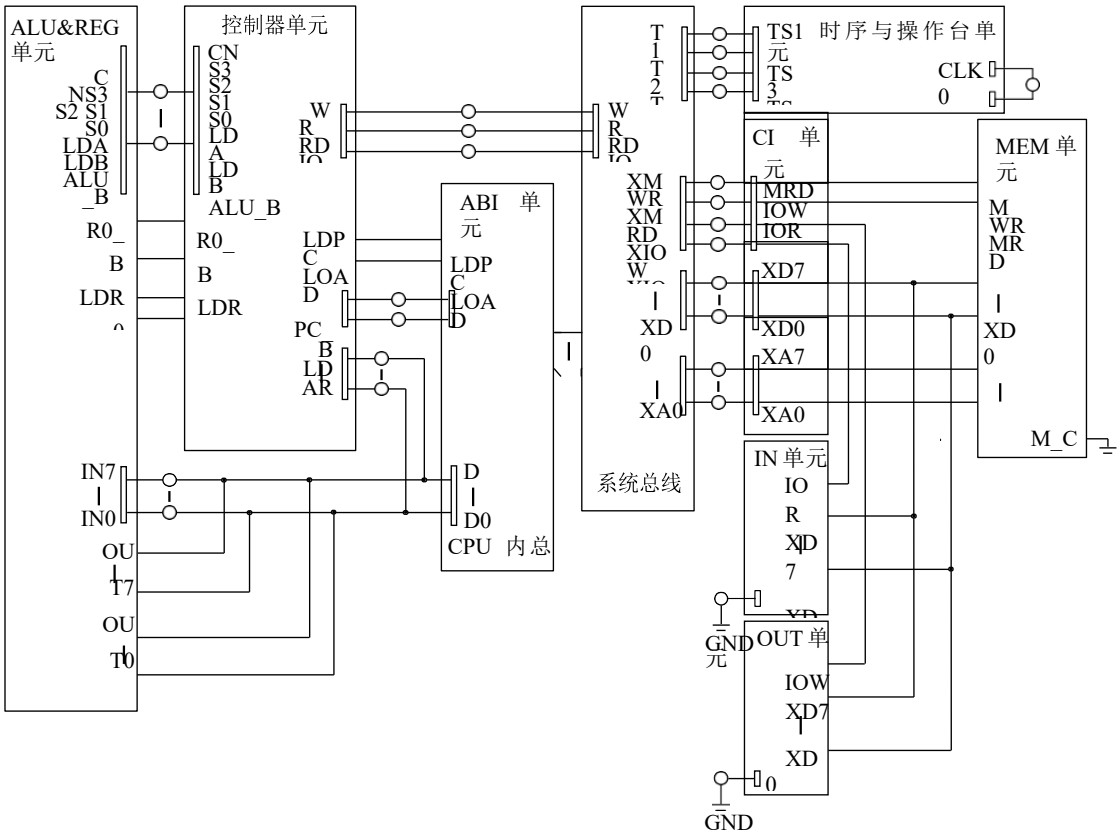


图 5-1-5　　实验接线图

实际接线如图5-1-5-1所示：

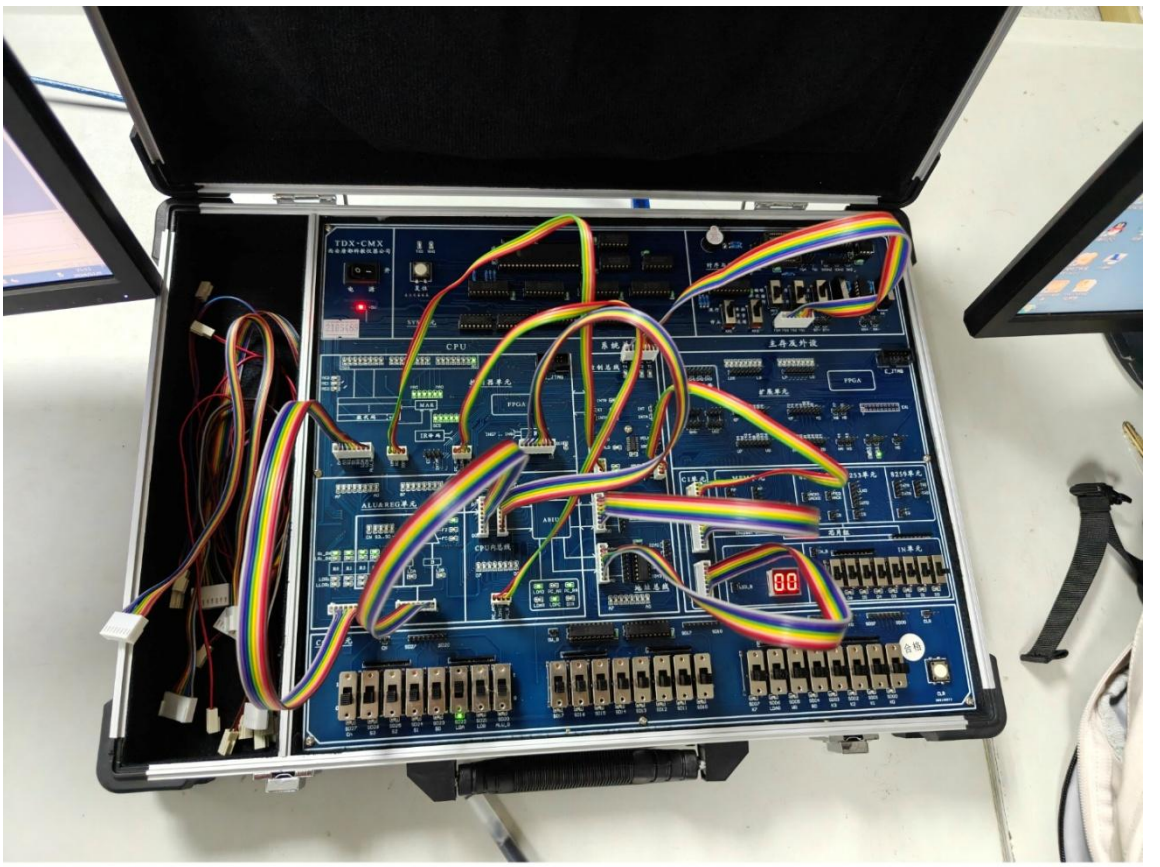

TDX-CMX
CPU
FPGA
OnePlus 13
HASSELBLAD
23mm f/1.6 1/100s ISO640

图 5-1-5-1　　实际接线图

(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。

(3) 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。

(4) 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。

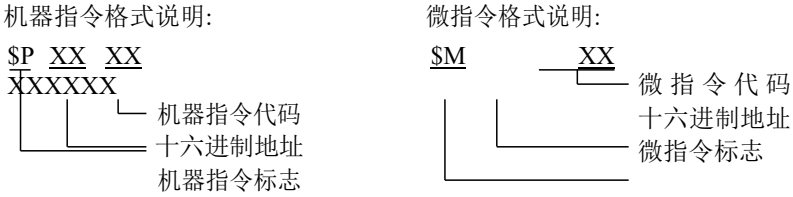
③ 将时序与操作台单元的开关 KK5 置为‘加 1’档。

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，直至确认指令码的输入无误为止。

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，微程序和机器程序的格式如下：



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：

```
; //************************************//
; //                                    //
; //     CPU与简单模型机实验指令文件    //
; //                                    //
; //     By TangDu CO.,LTD              //
; //                                    //
; //************************************//

; //****** Start Of Main Memory Data ******//
  $P 00 20    ; START: IN  R0     从IN单元读入数据送 R0
  $P 01 00    ; ADD R0,R0         R0和自身相加，结果送 R0
```

```
  $P 02 30    ; OUT R0            R0的值送 OUT单元显示
  $P 03 E0    ; JMP START         跳转至 00H地址
  $P 04 00    ;
  $P 05 50    ; HLT               停机
; //******* 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    ; R0->B
  $M 05 04B201    ; A加 B->R0
  $M 1D 105141    ; MEM->PC
  $M 30 001404    ; R0->A
  $M 32 183001    ; IN->R0
  $M 33 280401    ; R0->OUT
  $M 35 000035    ; NOP
  $M 3C 006D5D    ; 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。

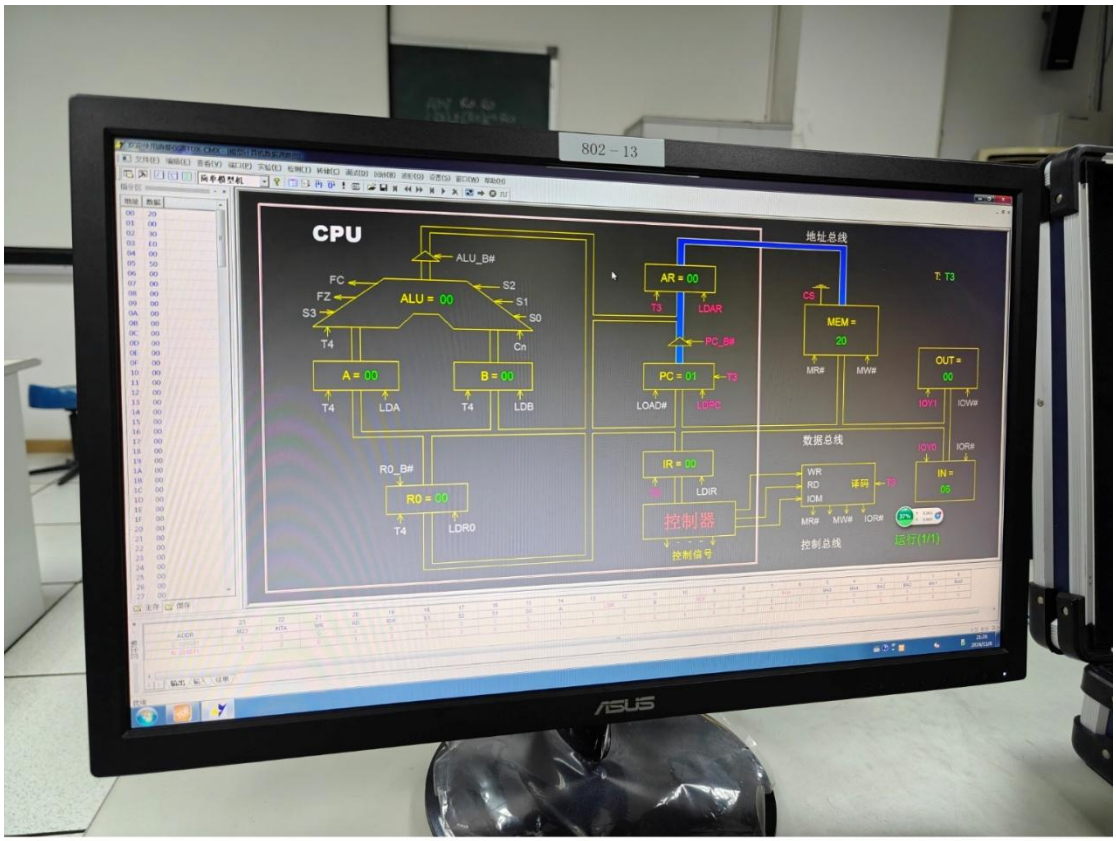
选择联机软件的“【转储】—【刷新指令区】”可以读出下位机所有的机器指令和微指令， 并在指令区显示，对照文件检查微程序和机器程序是否正确，如果不正确，则说明写入操作失败，应重新写入，可以通过联机软件单独修改某个单元的指令，以修改微指令为例，先用鼠标左键单击指令区的‘微存’TAB 按钮，然后再单击需修改单元的数据，此时该单元变为编辑框，输入 6 位数据并回车，编辑框消失，并以红色显示写入的数据。

3. 运行程序

方法一：本机运行

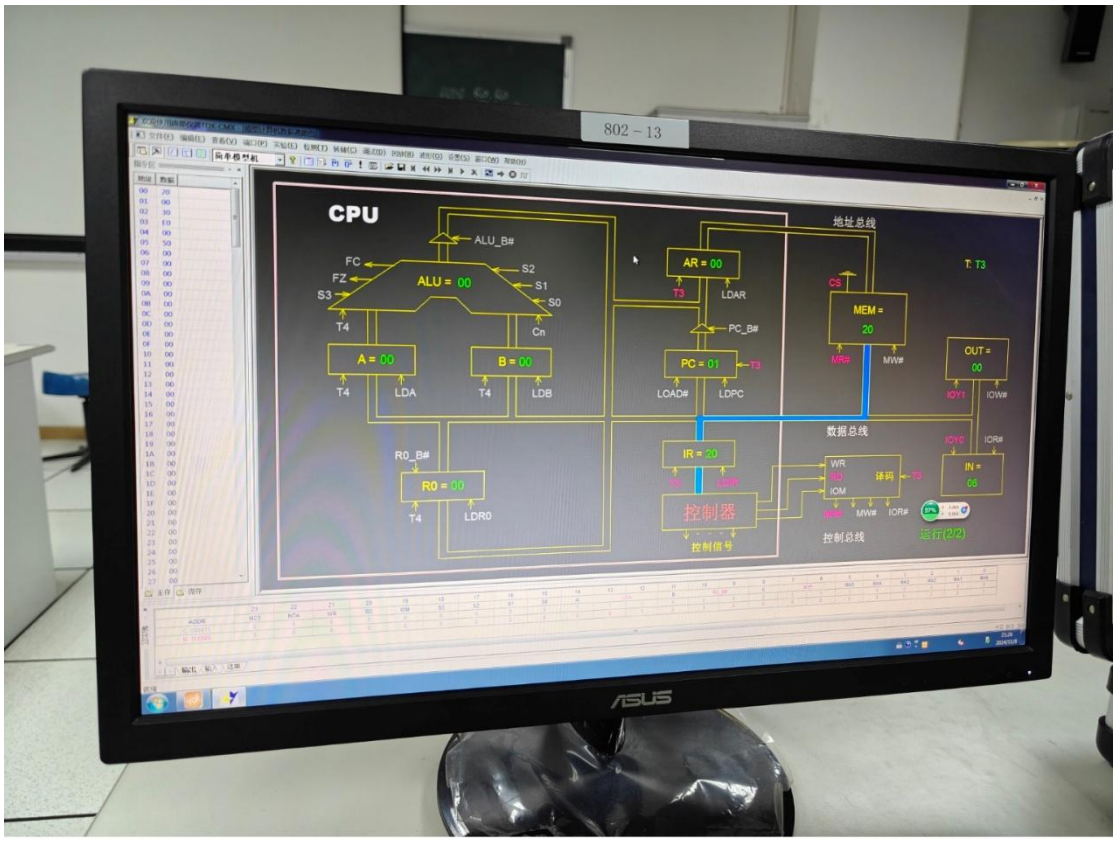
将时序与操作台单元的开关 KK1、KK3 置为‘运行’档，按动 CON 单元的总清按钮 CLR，将使程序计数器 PC、地址寄存器 AR 和微程序地址为 00H，程序可以从头开始运行，暂存器 A、B，指令寄存器 IR 和OUT 单元也会被清零。

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 CPU 内总线和地址总线，对照数据通路图，分析总线上的数据是否正确。如图一所示：



图一

当模型机执行完 JMP 指令后，检查 OUT 单元显示的数是否为 IN 单元值的 2 倍，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。

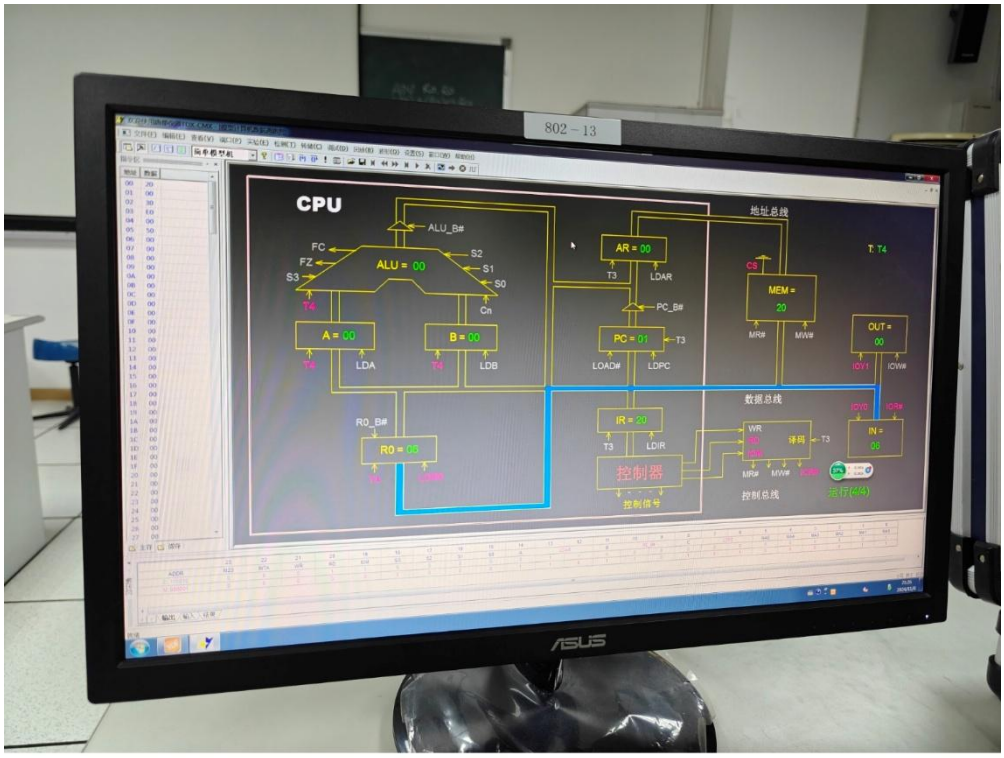


图二

方法二：联机运行

将时序与操作台单元的开关 KK1 和 KK3 置为‘运行’档，进入软件界面，选择菜单命令“【实验】—【简单模型机】”，打开简单模型机数据通路图。

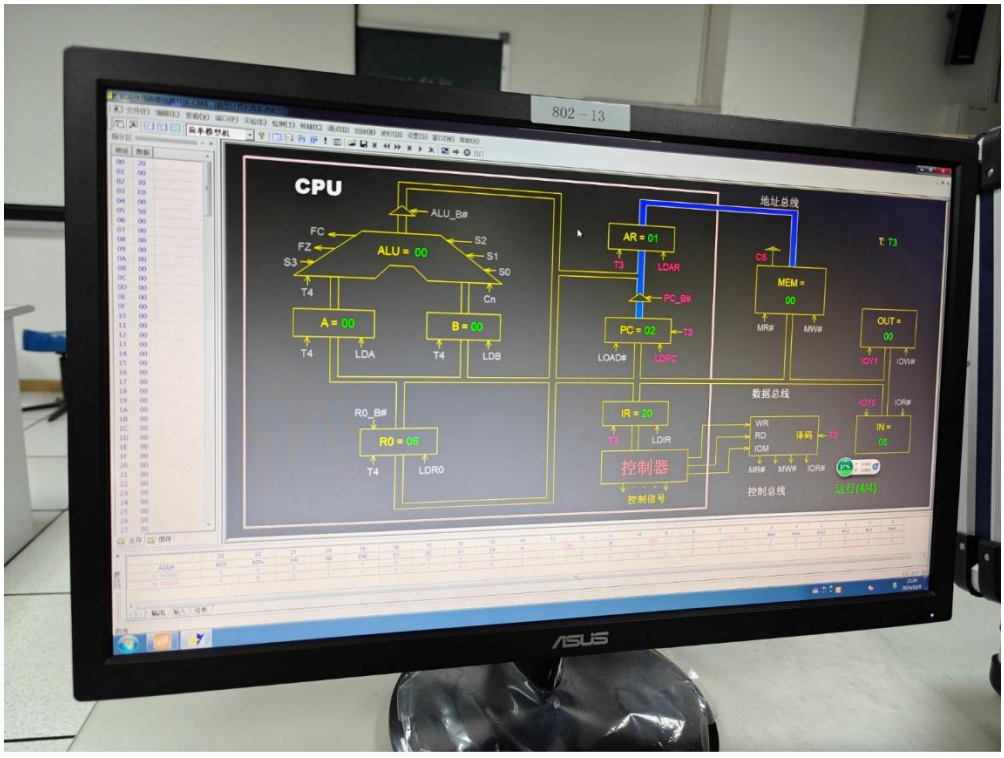
按动 CON 单元的总清按钮 CLR，然后通过软件运行程序，选择相应的功能命令，即可联机运行、监控、调试程序，当模型机执行完 JMP 指令后，检查 OUT 单元显示的数是否为 IN 单元值的 2 倍。在数据通路图和微程序流中观测指令的执行过程，并观测软件中地址总线、数据总线以及微指令显示和下位机是否一致。结果如图二到图十三所示：



OnePlus 13

HASSELBLAD
23mm f/1.6 1/100s ISO200

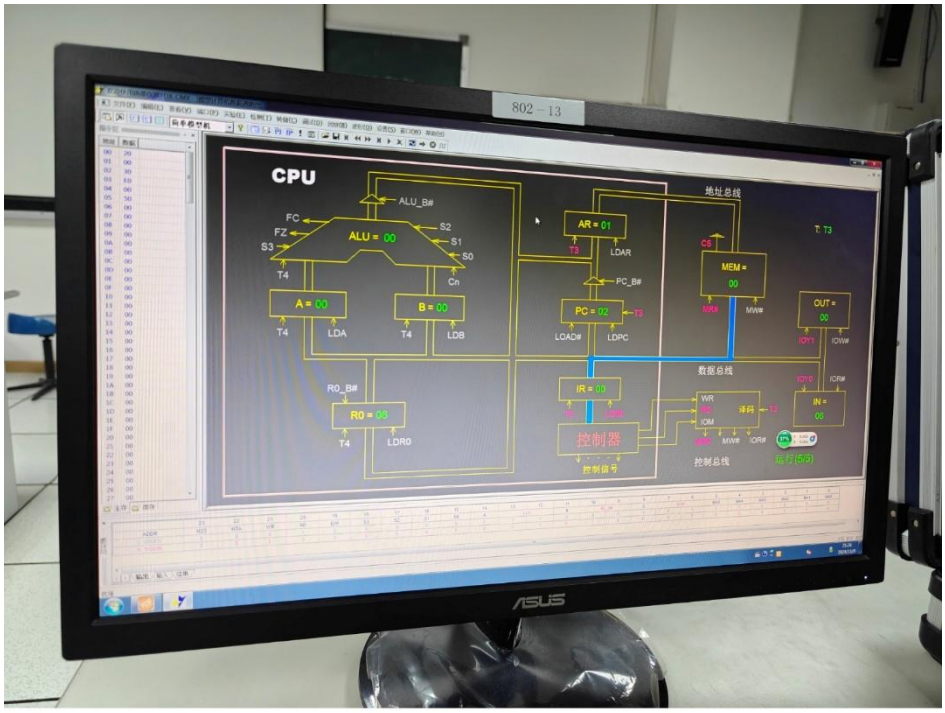
图三



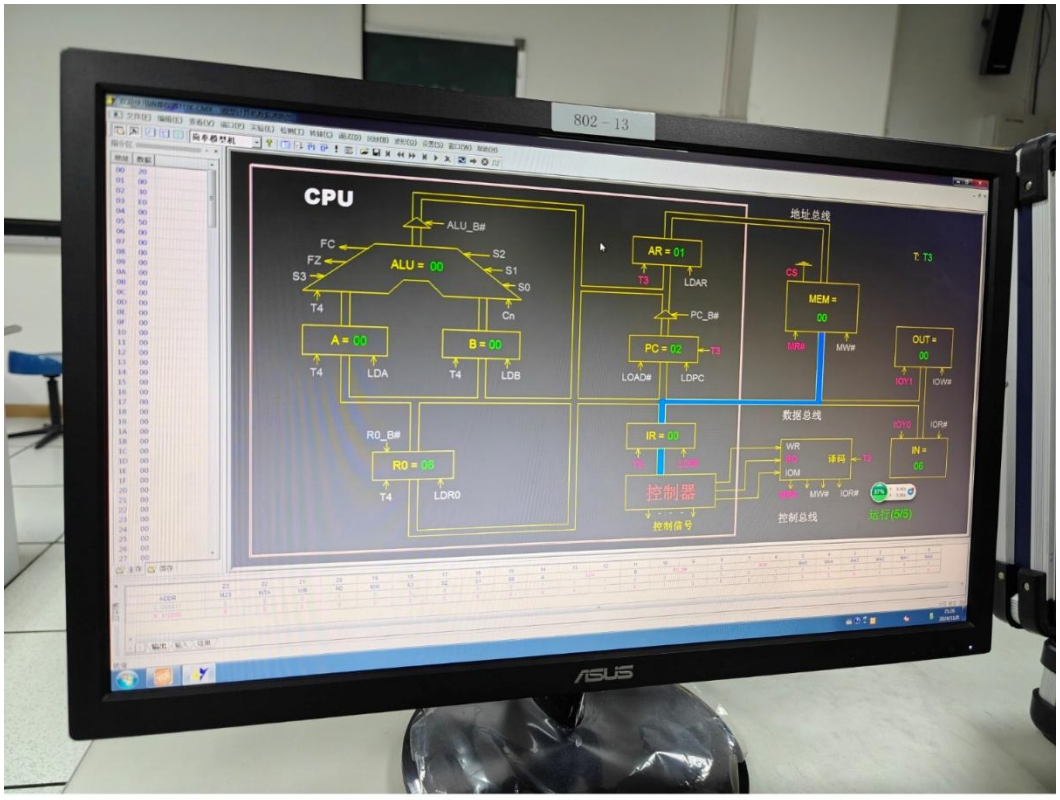
OnePlus 13

HASSELBLAD
23mm f/1.6 1/100s ISO200

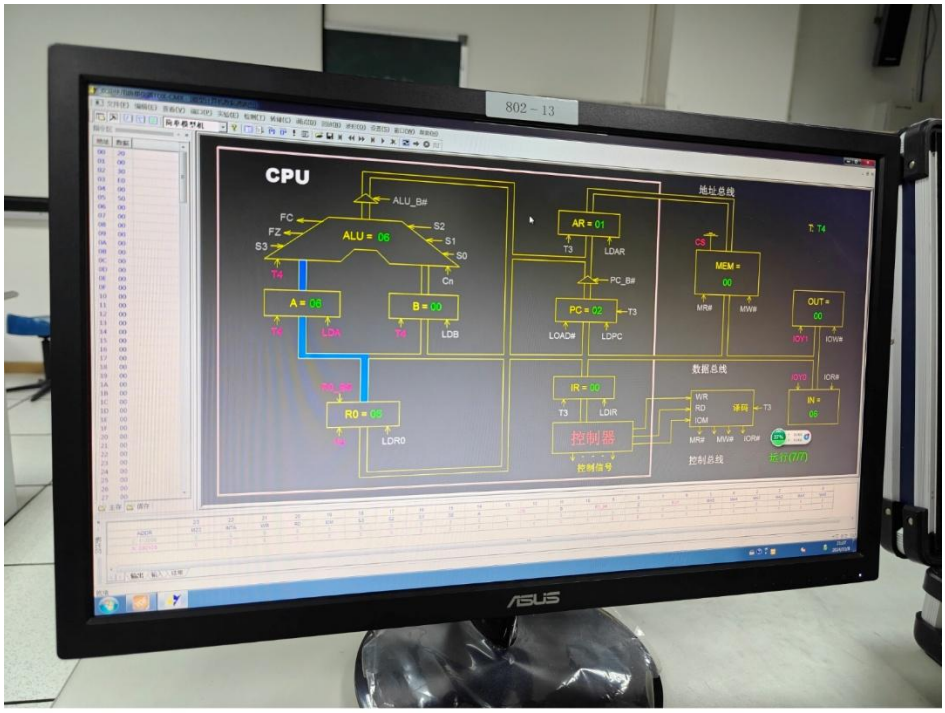
图四


802 – 13
CPU
ALU_B#
FC
FZ
S3
ALU = 00
S2
S1
S0
Cn
T4
A = 00
B = 00
LDA
LDB
AR = 01
LDAR
T3
地址总线
PC_B#
PC = 02
LOAD#
LDPC
MEM =
00
CS
MR#
MW#
OUT =
00
IOY1
IOW#
数据总线
IR = 00
WR
译码
IOM
IOY0
IOR#
IN =
00
R0_B#
R0 = 00
LDR0
控制器
控制信号
控制总线
运行(5/5)
OnePlus 13
HASSELBLAD
23mm f/1.6 1/100s ISO200

图五


802 – 13
CPU
ALU_B#
FC
FZ
S3
ALU = 00
S2
S1
S0
Cn
T4
A = 00
B = 00
LDA
LDB
AR = 01
LDAR
T3
地址总线
PC_B#
PC = 02
LOAD#
LDPC
MEM =
00
CS
MR#
MW#
OUT =
00
IOY1
IOW#
数据总线
IR = 00
WR
译码
IOM
IOY0
IOR#
IN =
00
R0_B#
R0 = 00
LDR0
控制器
控制信号
控制总线
运行(5/5)
OnePlus 13
HASSELBLAD
23mm f/1.6 1/100s ISO200

图六

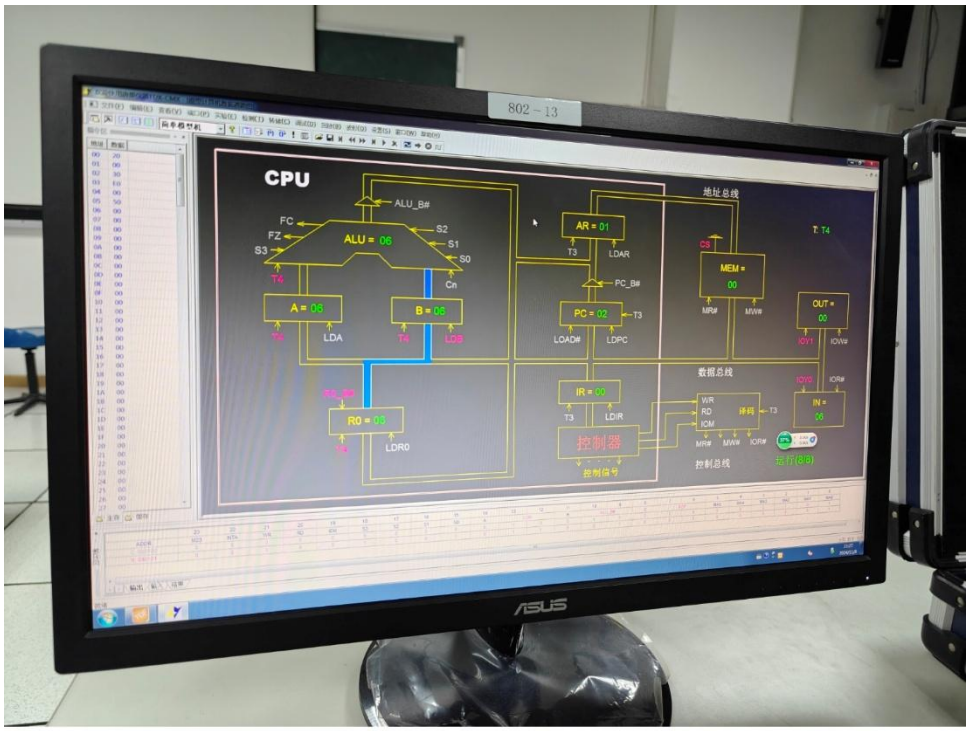


OnePlus 13

HASSELBLAD

23mm f/1.6 1/100s ISO200

图七

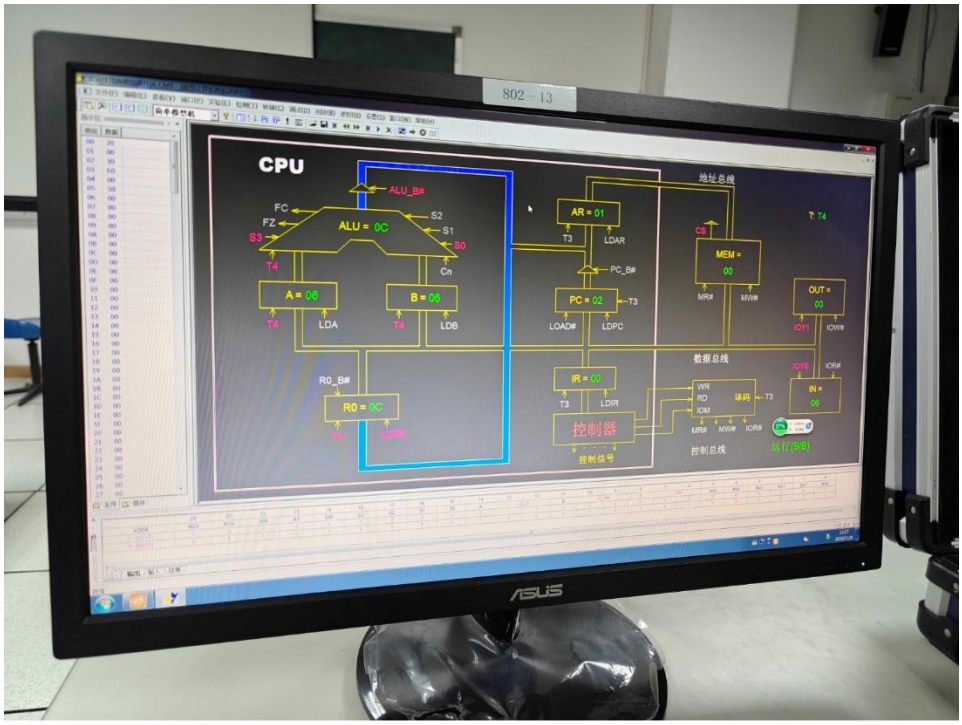


OnePlus 13

HASSELBLAD

23mm f/1.6 1/100s ISO200

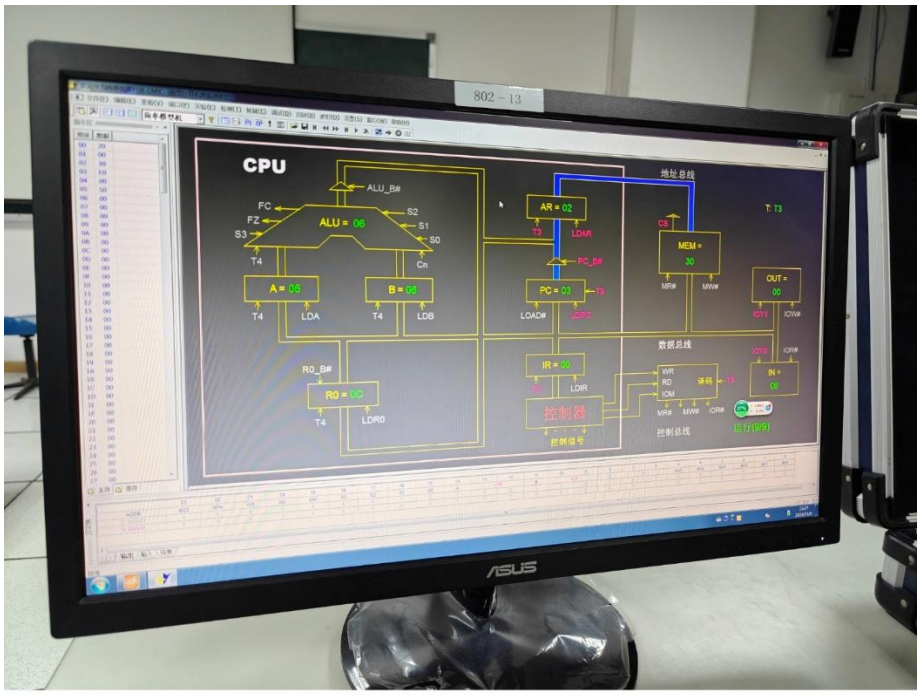
图八



OnePlus 13

HASSELBLAD
23mm f/1.6 1/100s ISO200

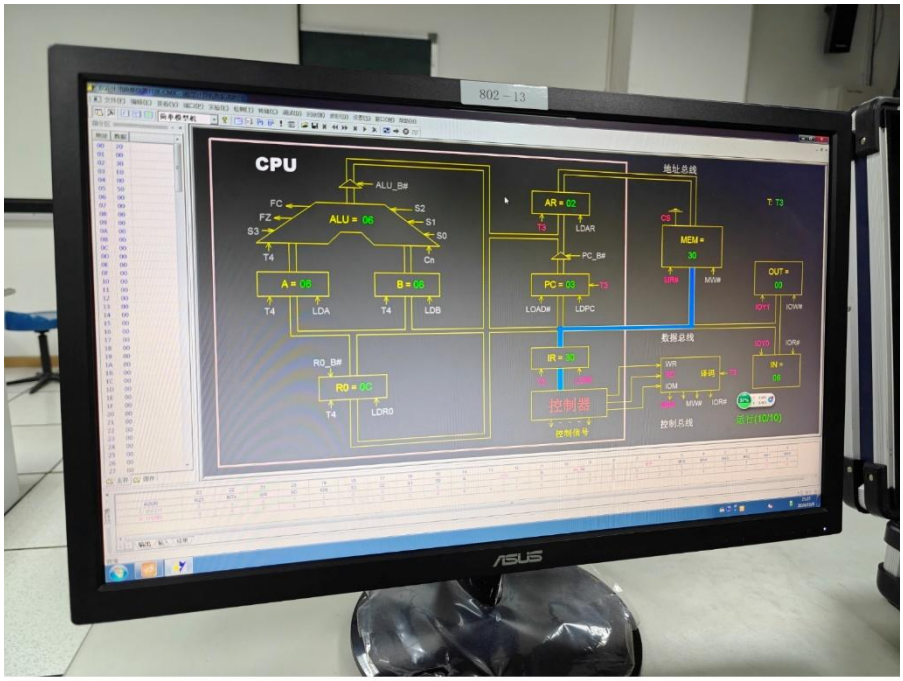
图九



OnePlus 13

HASSELBLAD
23mm f/1.6 1/100s ISO200

图十

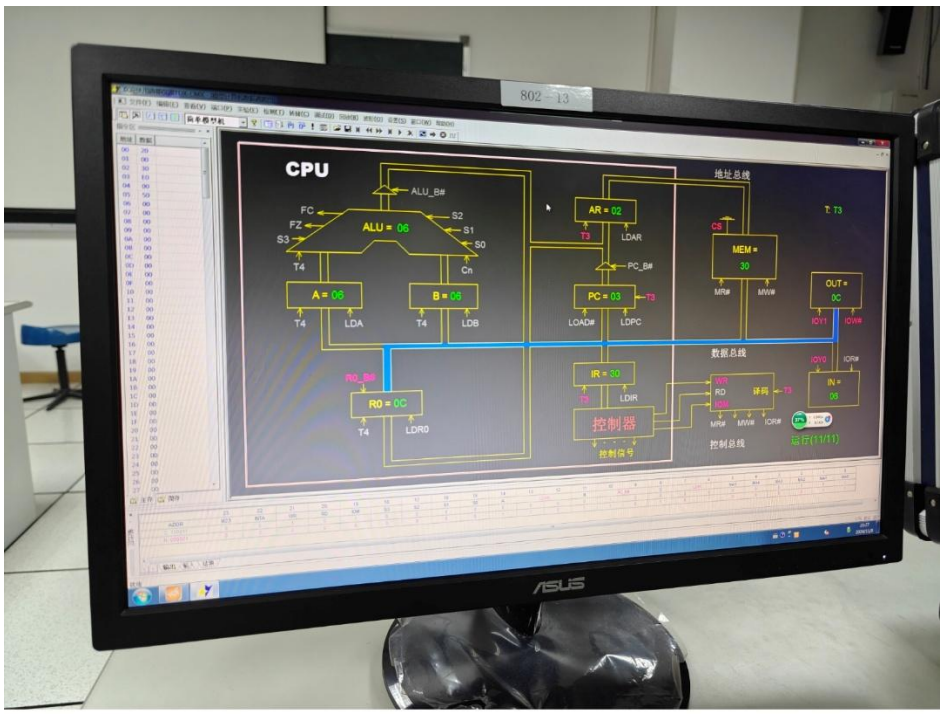


OnePlus 13

HASSELBLAD

23mm f/1.6 1/100s ISO200

图十一

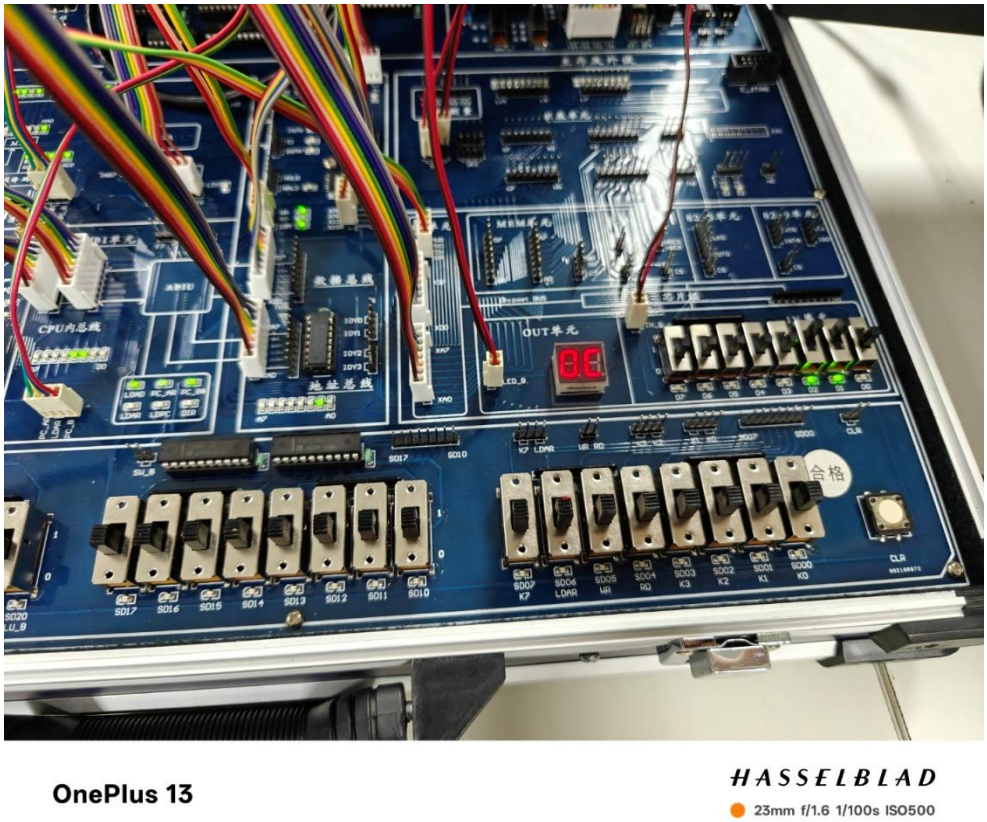


OnePlus 13

HASSELBLAD

23mm f/1.6 1/100s ISO200

图十二



图十三

3 实验小结

在本次实验中，我们深入探讨了微程序的手动校验与写入方法，并了解了其在机器程序设计中的具体应用。通过实验操作，我掌握了微程序控制器的基本组成结构，包括如何通过时序控制单元和操作台单元的设置实现微指令的正确加载和执行。

实验的关键在于熟练掌握微程序与机器程序的编制和写入过程。在手动校验微程序时，我们使用了时序与操作台单元的不同开关组合，通过对低位、中位和高位数据的观察，确保微指令的输入准确无误。在手动写入机器程序的过程中，我们按顺序将指令逐步写入存储器，并使用指数据指示灯进行校验。若出现错误，重新写入并验证，直至所有指令正确无误。

我们还探索了联机写入与校验的过程，利用软件简化了指令下载与验证。通过查看指令区显示的机器指令与微指令，我们能够快速确认写入内容的正确性，并通过编辑功能直接修改错误的指令数据，进一步提高了工作效率。

在程序运行阶段，我们使用了单步和联机运行两种方式。通过逐步执行微指令并对比微程序流程图，我们观察到时序控制信号在数据流动中的作用，了解了如何通过控制信号协调各单元的操作步骤，确保指令的顺利执行。此外，通过数据通路图观察到的地址总线和数据总线的变化，使我对微程序控制器的工作原理及其信号传输逻辑有了更清晰的认识。

总体而言，本次实验加深了我对微程序控制器设计、编程和运行原理的理解，并让我熟悉了在硬

件设计中通过手动和联机方式进行程序加载和校验的方法。这次实验不仅提升了我对微程序控制器各项原理的掌握，特别是在数据通路和信号控制方面的操作，还让我对其在电子系统中的实际应用有了更为全面的认识。