浙江工业大学

计算机组成原理课程设计

姓　　名：李飞飞

班　　级：2022 软件工程（移动应用开发 1）

学　　号：202105710309

提交日期：2024.6.3

目　　录

一、 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。

二、 实验内容

1. 设计并实现一套完整的指令系统；
2. 设计并实现完整的计算机（采用上述指令系统）；
3. 利用该计算机实现 更相减损数 求 GCD 的功能

三、 实验原理

3.1 高级语言代码

计组综合实验_LFF.txt　更相减损术.cpp

I: > 更相减损术.cpp > ...

```cpp
#include<iostream>
using namespace std;
void swap(int &a,int &b){
    a^=b;
    b^=a;
    a^=b;
}
bool checkBigger(int &a,int &b){
    return b-a<=0;
}
int gcd() {
    int a,b;
    cin>>a>>b;
    do{
        if(checkBigger(a,b))swap(a,b);
        b-=a;
    }while(b);
    cout<<a;
}
```

3.2 指令系统及分析

1. 指令介绍

   运算指令：XOR(异或) SUB

   控制转移指令：JMP BZC

   数据传送指令：TEST，OUT, IN (TEST 就是 MOV, 因为之前的需求想改功能，后面又不改了)，

   其中 BZC, JMP 为双字长，其它单字长

2. 指令格式

所有单字节指令(XOR、SUB、TEST)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，并规定：

RS 或 RD	选定的寄存器
00	R0
01	R1
10	R2
11	R3

IN 和 OUT 的指令格式为：

第一字节								第二字节
I7	I6	I5	I4	I3	I2	I1	I0	I7-I0
OP-CODE				RS		RD		P

其中 OP-CODE 为操作码，RS 为源寄存器，RD 为目的寄存器，P 为 I/O 端口号，使用的是地址总线的高两位进行译码，I/O 地址空间被分为四个区：

A7 A6	选定	地址空间
00	IOY0	00-3F
01	IOY1	40-7F
10	IOY2	80-BF
11	IOY3	C0-FF

系统设计五种数据寻址方式，即立即、直接、间接、变址和相对寻址，LDI 指令为立即寻址，LAD、STA、JMP 和 BZC 指令均具备直接、间接、变址和相对寻址能力。

JMP 和 BZC 指令格式如下。

第一字节								第二字节
I7	I6	I5	I4	I3	I2	I1	I0	I7-I0
OP-CODE				M		RD		D

其中 M 为寻址模式，具体见下表 ，以 R2 做为变址寄存器 RI。

寻址模式 M	有效地址 E	说明
00	E = D	直接寻址
01	E = （D）	间接寻址
10	E = （RI） + D	RI 变址寻址
11	E = （PC） + D	相对寻址

3. 微指令格式

23	22	21	20	19	18-15	14-12	11-9	8-6	5-0
M23	CN	WR	RD	IOM	S3-S0	A字段	B字段	C字段	UA5-UA0

A字段

14	13	12	选择
0	0	0	NOP
0	0	1	LDA
0	1	0	LDB
0	1	1	LDRi
1	0	0	保留
1	0	1	LOAD
1	1	0	LDAR
1	1	1	LDIR

B字段

11	10	9	选择
0	0	0	NOP
0	0	1	ALU_B
0	1	0	RS_B
0	1	1	RD_B
1	0	0	RI_B
1	0	1	保留
1	1	0	PC_B
1	1	1	保留

C字段

8	7	6	选择
0	0	0	NOP
0	0	1	P<1>
0	1	0	P<2>
0	1	1	P<3>
1	0	0	保留
1	0	1	LDPC
1	1	0	保留
1	1	1	保留

其中 UA5-UA0 为 6 位的后续微地址，A、B、C 字段为 3 个译码字段，分别由 3 个控制位译码得到多种指令。C 字段中 P<1>、P<2>、P<3>为测试字位，功能是根据机器指令及相应的微代码进行译码，使微程序转入相应的微地址入口，从而完成对指令的识别，并实现微程序的分支

4. 新指令介绍

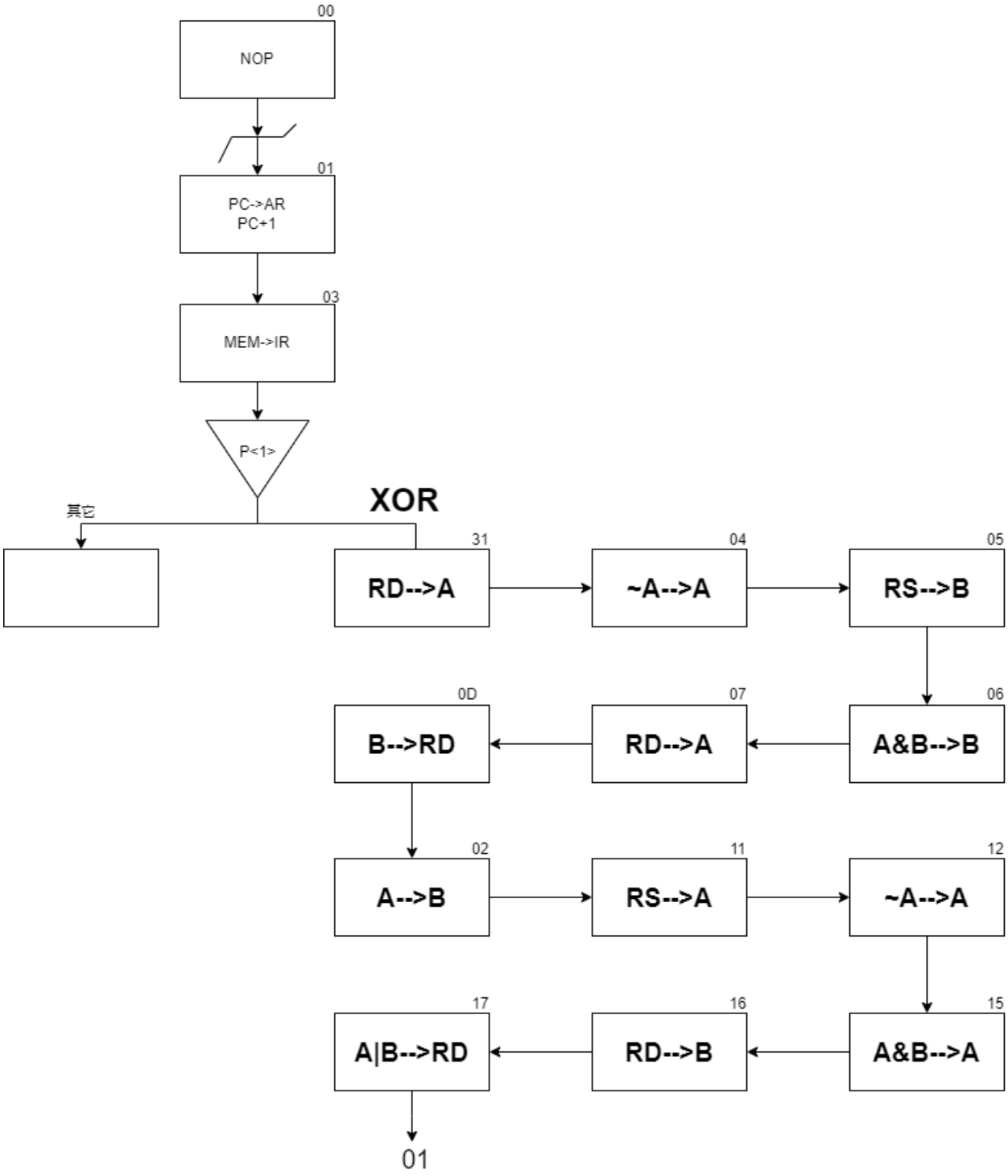
新增异或指令格式：

助记符	指令格式			指令功能
XOR RD RS	0000	RS	RD	RS^RD->RD

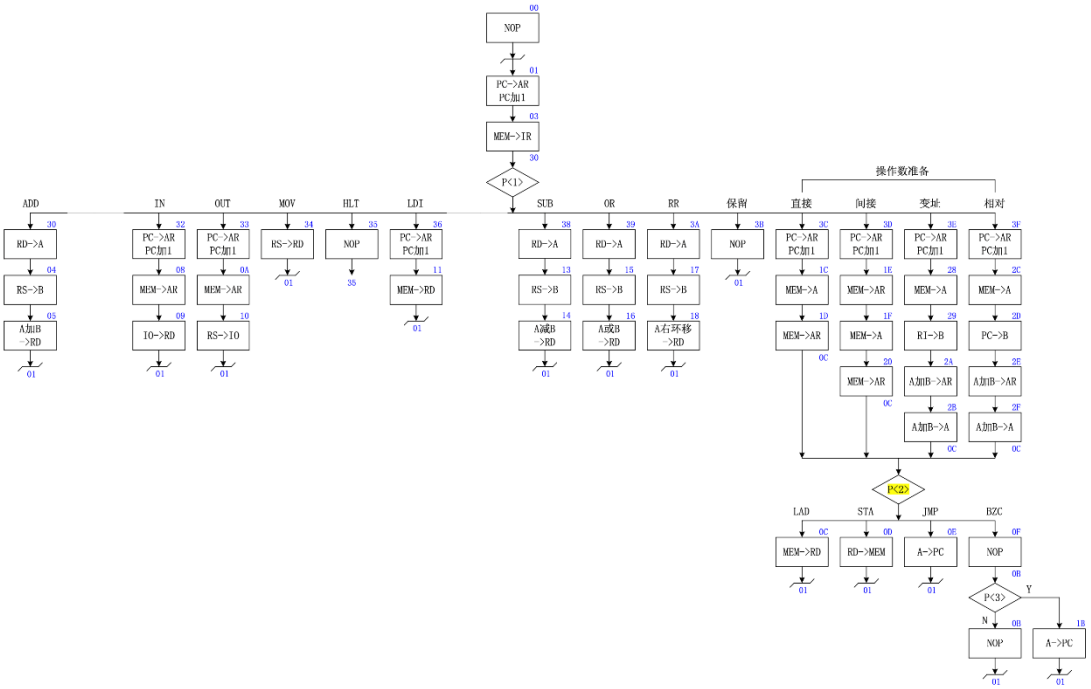
3.3 指令框图及分析

- 新增指令框图

  备注：此图省略了已有指令，仅给出异或指令，右上角是对应微指令编号。那个



- 其余指令框图



3.4 指令系统对应微程序二进制代码及分析

1. XOR 指令二进制表

备注：课本没有给出的其它指令都与课本的相同，这里不再赘述

功能	地址	高五位	S0~S3	A 字段	B 字段	C 字段	UA6-UA0	16 进制格式
RD->A	31	00000	0000	001	011	000	000100	001604
~A->A	04	00000	0100	001	001	000	000101	021205
RS->B	05	00000	0000	010	010	000	000110	002406
B&A->B	06	00000	0010	010	001	000	000111	012207
RD->A	07	00000	0000	001	011	000	001101	00160D
B->RD	0D	00000	0001	011	001	000	000010	00B202
A->B	02	00000	0000	010	001	000	010001	002211
RS->A	11	00000	0000	001	010	000	010010	001412
~A->A	12	00000	0100	001	001	000	010101	021215
A&B->A	15	00000	0010	001	001	000	010110	011216
RD->B	16	00000	0000	010	011	000	010111	002617
A|B->RD	17	00000	0011	011	001	000	000001	01B201

2. 分析

根据公式 **a ^ b = (a & ~ b) | (b & ~ a)**

注意两点：

- 取反操作只能对暂存器 A 进行
- 数据只能送到 RD,不能去 RS。

合理利用 RS、RD、暂存器 A、暂存器 B 来记录数据可以得到一个由 12 条微指令构成的只用一条单字节和且或非命令实现的异或命令

3.5 机器程序及分析

分析见每段程序前的备注，把每一部分干什么的说的很清楚了

```
    ;读入数据：
$P 00 21      ;IN R1,00H
$P 01 00      ;
$P 02 22      ;IN R2,00H
$P 03 00      ;

    ;GCD
$P 04 08      ;MOV R0,R2
$P 05 84      ;SUB R0,R1
$P 06 F0      ;BZC
$P 07 10      ; MYSWAP
    ;MYSUB
$P 08 86      ;TEST R2,R1
$P 09 F0      ;BZC-->跳 END
$P 0A 0D       ; END
$P 0B E0      ;JMP->跳 GCD
$P 0C 04      ; GCD

    ;END
$P 0D 34       ;OUT R1
$P 0E 40      ; 40H
$P 0F 50      ;HALT
```

```
      ;MYSWAP
$P 10 16       ;XOR R2,R1
$P 11 19       ;XOR R1,R2
$P 12 16       ;XOR R2,R1
$P 13 E0       ;JMP-->跳 MYSUB
$P 14 08       ; MYSUB

;//保留 IN,SUB,OUT,JMP,BZC  新增 TEST(双字长) XOR(六字长)//
;     TEST(占用 5 条,入口 30)
$M 30 003401    ; RS->RD
;       以下作废
$M 3B 006D74    ; PC->AR,PC+1
$M 34 107075    ; MEM->IR
$M 35 001636    ; RD->A
$M 36 002414    ; RS->B
;复用 M14
;本为:$M 37 05B201;A-B->RD

;     XOR(占用 12 条,入口 31)
$M 31 001604       ; RD->A
$M 04 021205       ; ~A->A
$M 05 002406       ; RS->B
$M 06 012207       ; A&B->B
$M 07 00160D       ; RD->A
$M 0D 00B202       ; B->RD //RD 当前为 R1&~R2
$m 02 002211       ; A->B
$M 11 001412       ; RS->A
$M 12 021215       ; ~A->A
$M 15 011216       ; A&B->A
$M 16 002617       ; RD->B
$M 17 01B201       ; A|B->RD
```

```
;    公用指令：
$M 00 000001      ; NOP
$M 01 006D43      ; PC->AR, PC 加 1
$M 03 107070      ; MEM->IR
;    IN
$M 32 006D48      ; PC->AR, PC 加 1
$M 08 106009      ; MEM->AR
$M 09 183001      ; IO->RD
;    SUB
$M 38 001613      ; RD->A
$M 13 002414      ; RS->B
$M 14 05B201      ; A 减 B->RD
;    OUT
$M 33 006D4A      ; PC->AR, PC 加 1
$M 0A 106010      ; MEM->AR
$M 10 280401      ; RS->IO
;    HLT
$M 35 000035      ; NOP
;直接寻址：
$M 3C 006D5C      ; PC->AR, PC 加 1
$M 1C 10101D      ; MEM->A
$M 1D 10608C      ; MEM->AR, P<2>
;    JMP & BZC
$M 0E 005341      ; A->PC
$M 0F 0000CB      ; NOP, P<3>
$M 0B 000001      ; NOP
$M 1B 005341      ; A->PC
```

四、 实验步骤

4.1 微程序写入及校验

1、写入微程序

用联机软件的“【转储】—【装载】”功能将该格式（*.TXT）文件装载入实验系统。装入过程中，在软件的输出区的‘结果’栏会显示装载信息，如当前正在装载的是机器指令还是微指令，还剩多少条指令等。

2、校验微程序

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。检查微控器相应地址单元的数据是否和表 3-2-2 中的十六进制数据相同，如果不同，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微指令，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。

4.2 机器程序写入及校验

1、写入微程序

用联机软件的“【转储】—【装载】”功能将该格式（*.TXT）文件装载入实验系统。装入过程中，在软件的输出区的‘结果’栏会显示装载信息，如当前正在装载的是机器指令还是微指令，还剩多少条指令等。

2、校验微程序

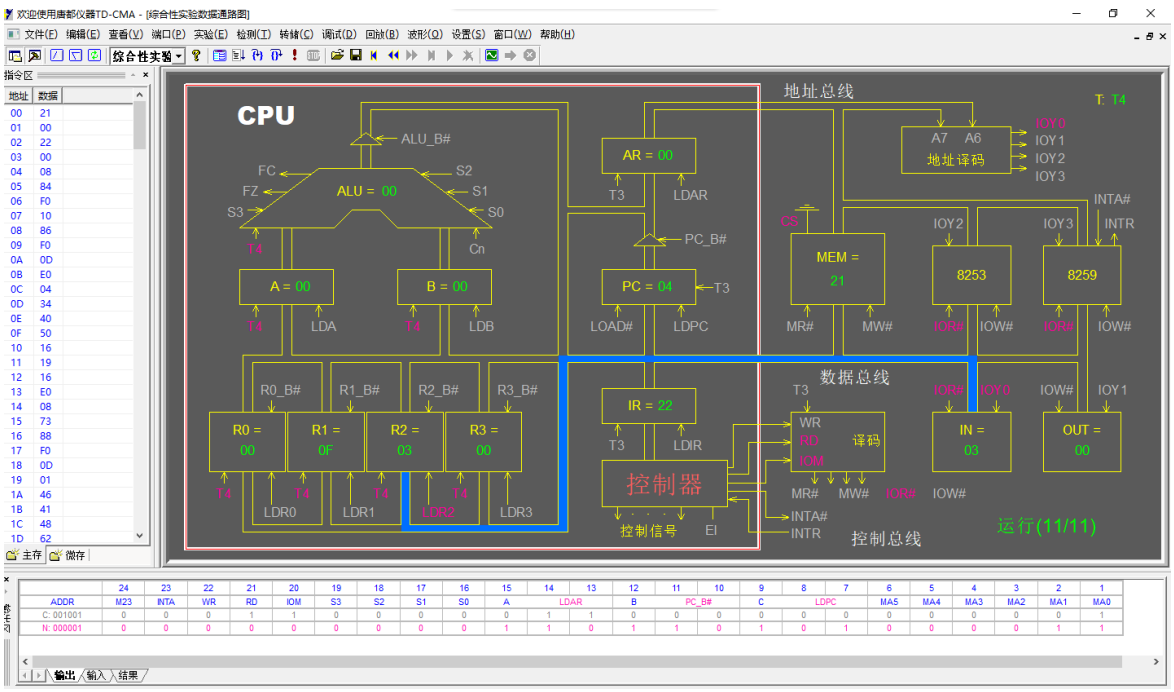
选择联机软件的“【转储】—【刷新指令区】”可以读出下位机所有的机器指令和微指令，并在指令区显示。检查微控器相应地址单元的数据是否和表 3-2-2 中的十六进制数据相同，如果不同，则说明写入操作失败，应重新写入，可以通过联机软件单独修改某个单元的微指令，先用鼠标左键单击指令区的‘微存’TAB 按钮，然后再单击需修改单元的数据，此时该单元变为编辑框，输入 6 位数据并回车，编辑框消失，并以红色显示写入的数据。

五、 实验结果及分析

5.1 演示程序一

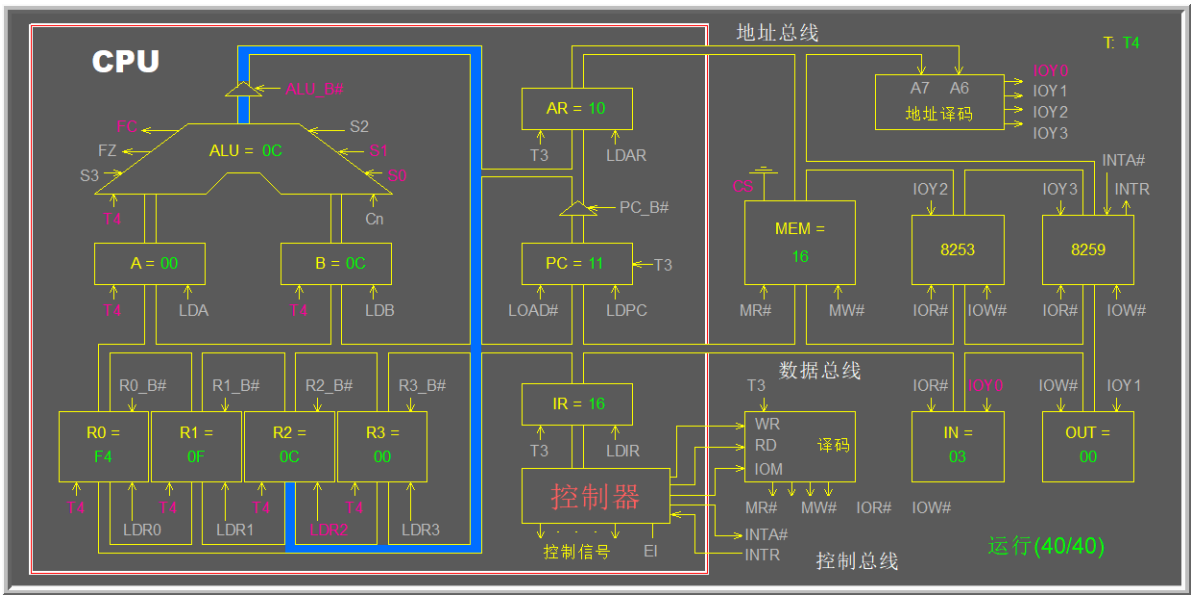
数据：15 3

输入：

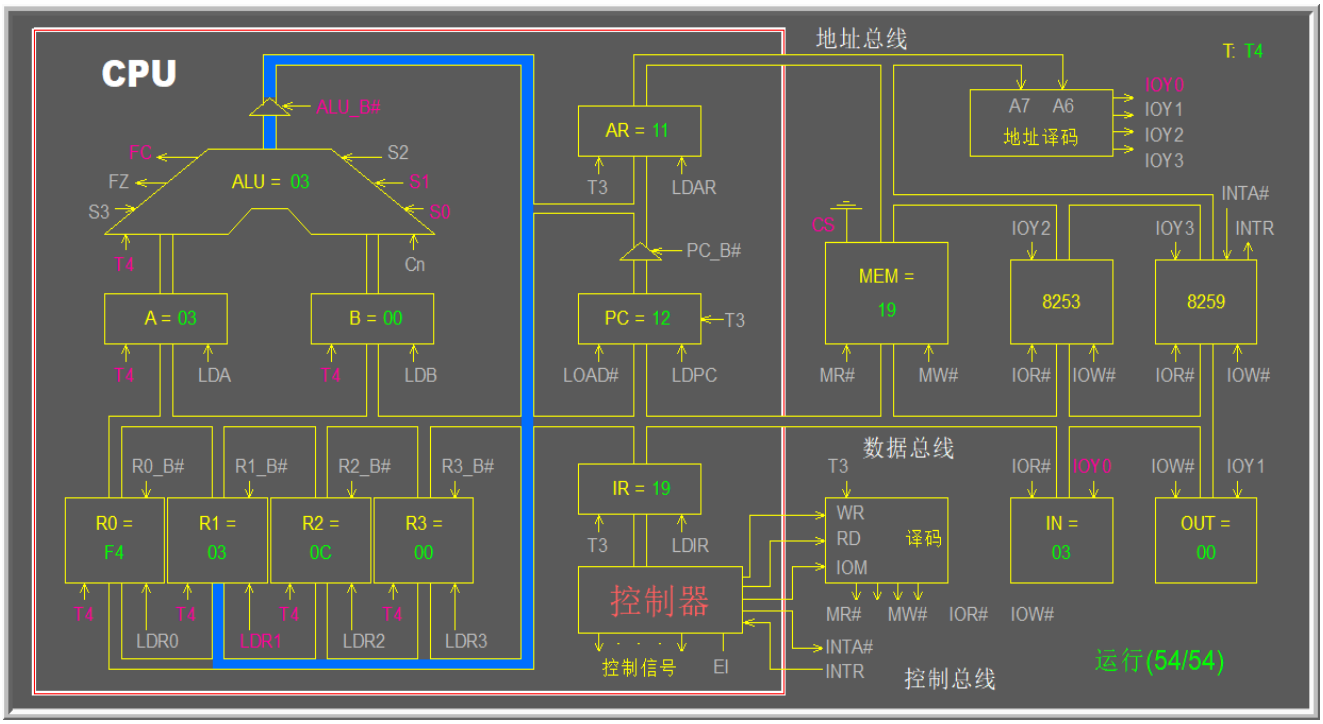


结果：

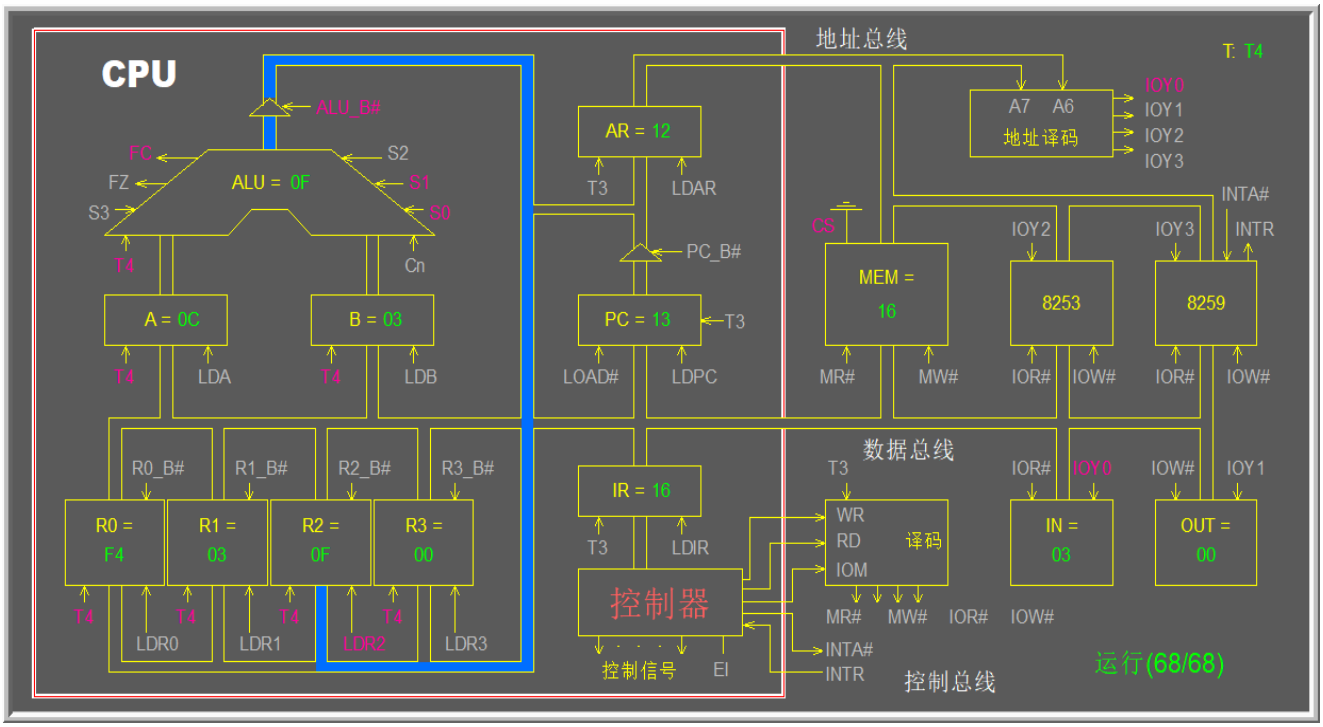
- R2^=R1



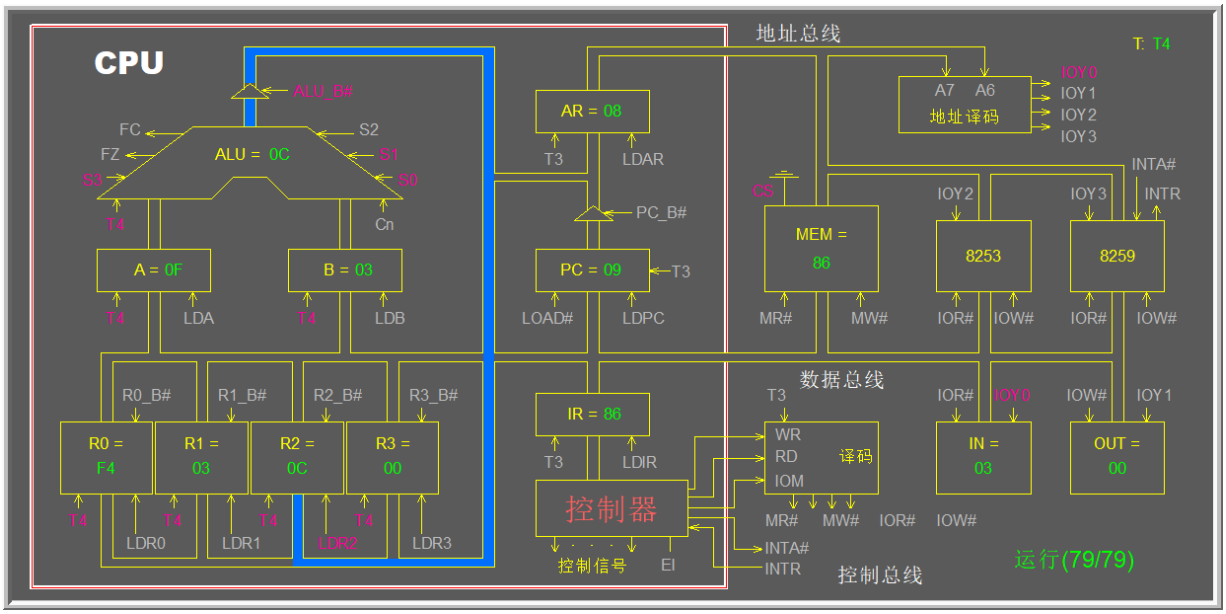
- R1^=R2



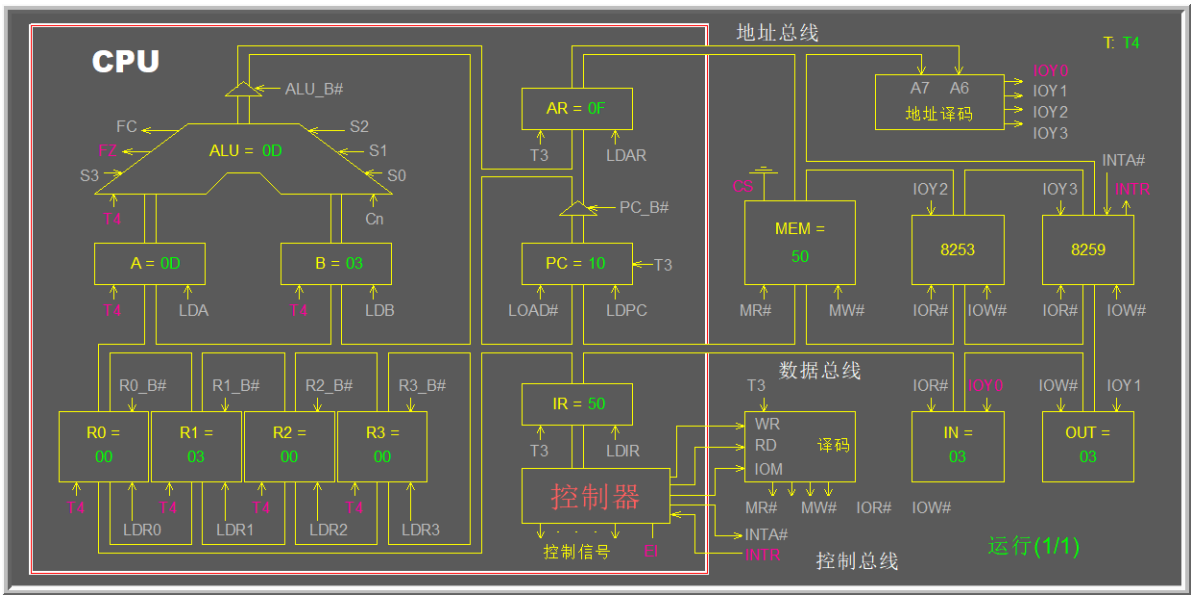
- R2^=R1（完成交换）



- R2-=R1（第一次更相减损）



- 结果



分析：

3^15=12 15^12=3 12^3=15

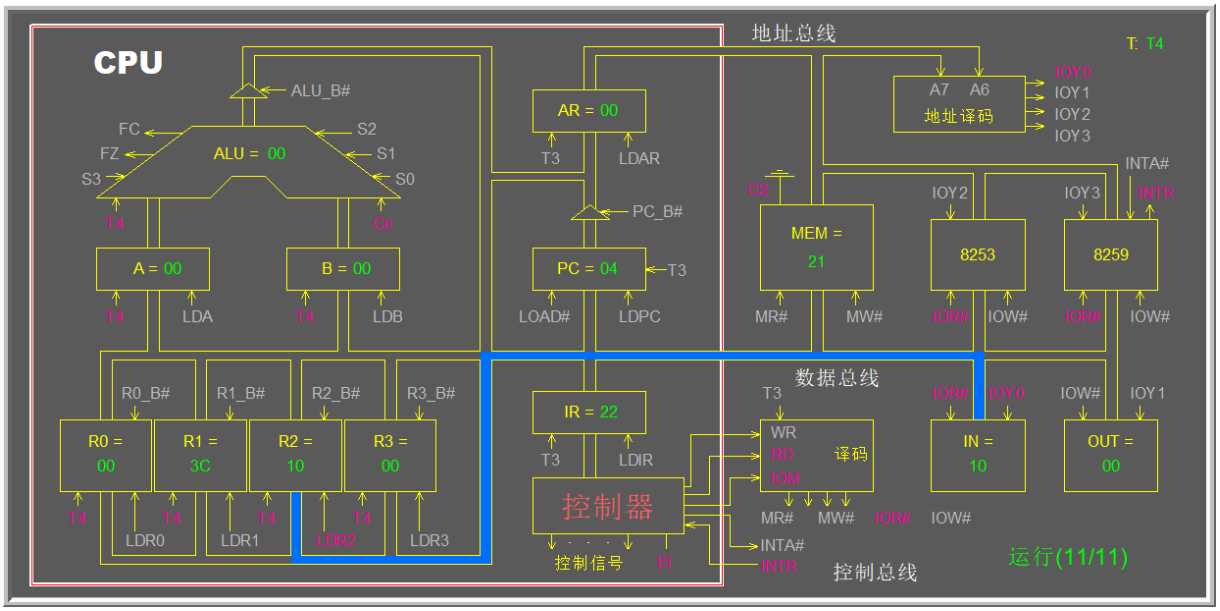
对应 16 进制：15=F, 3=3 12=C

之后正常求，15 与 3 的 GCD 是 3

5.2 演示程序二

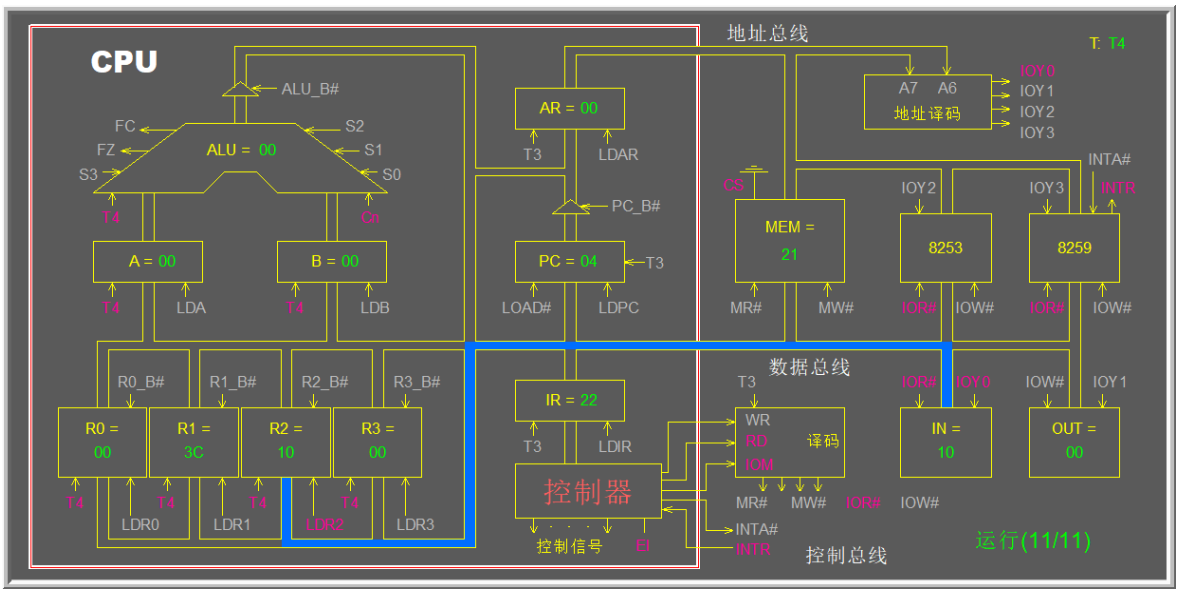
数据：60 16

输入 60 16

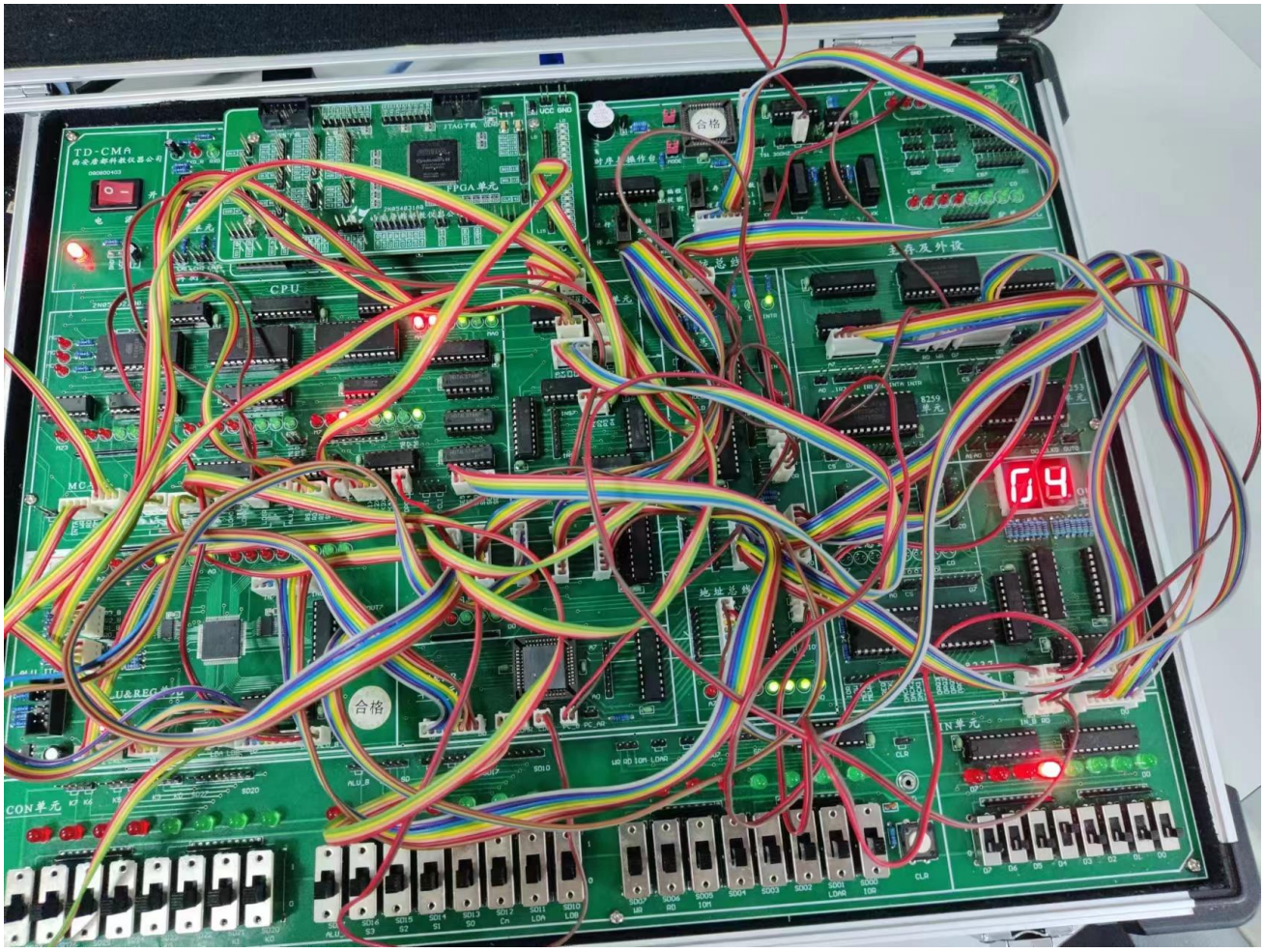


结果：

- 程序截图：



- 试验仪拍照（15, 3 的忘记拍了）



分析：GCD(60,16)=4

六、 实验问题及思考

6.1 当前所实现计算机是否完整？不完整是缺少哪些部件？

不完整，缺少中断功能，并且我移除了大量默认指令，所以不支持很多运算，不支持内存操作

6.2 当前所实现计算机，是否能实现除法运算？

同上，因为我删了很多默认指令所以当前不支持除法。加上默认指令后支持除法，可以通过右移和减法实现。此外本程序实现的是 GCD，拓展可得 LCM,。A*B 可以用右移和加法实现的龟速乘做，A*B/GCD 可以从 1~max(a,b)挨个试乘得到，因为 GCD 一定为 A*B 的一个约数，不会除出小数，所以试乘就能试出来

6.3 当前所实现计算机，还能实现哪些更复杂的计算？

加上默认指令后如上所诉说可以实现龟速乘、LCM、阶乘、快速排序等等

6.4 当前所实现计算机，指令系统的双字长指令是如何实现的？

用两个连续的地址实现，在指令中执行 PC->AR，PC+1->PC → MEM->?(AR、RS、RD 等)就取出来第二个字节的立即数

七、 实验验收答辩环节问题和解答

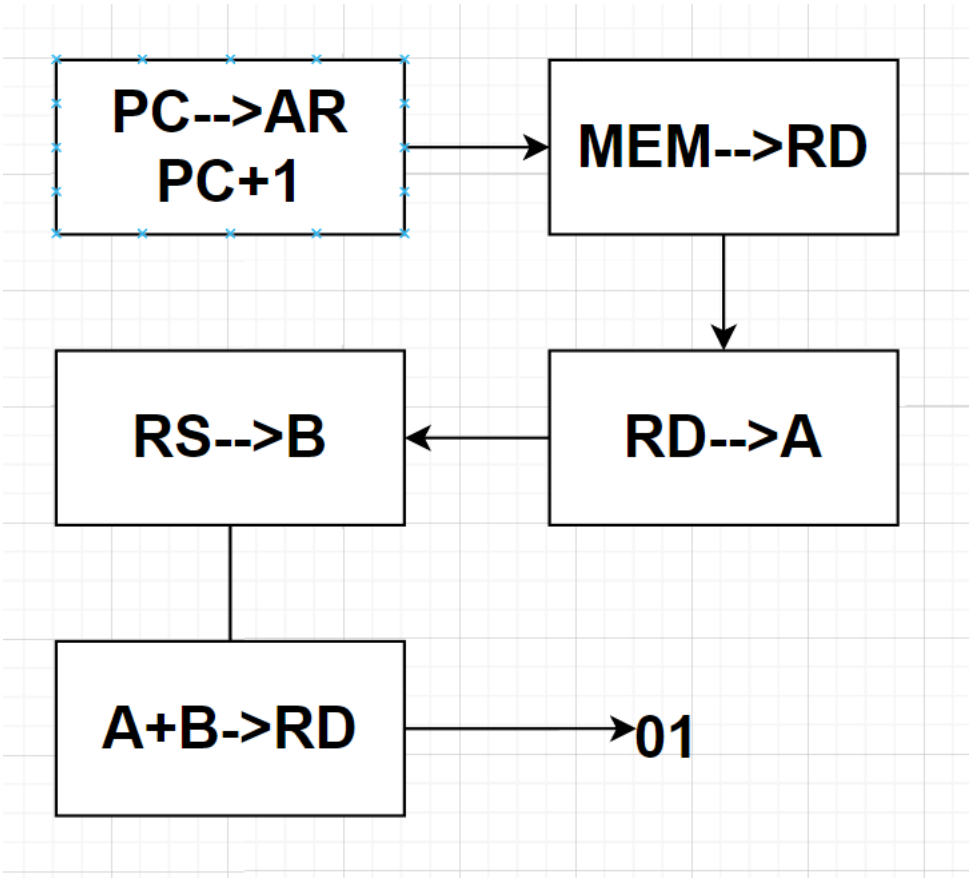
7.1 增加对输入数据判断 0 的指令

添加一条双字长机器指令 TEST：

TEST R0,R1/R2(RD)　00H

对应微程序：

这里是提问时我写的，实际上 MEM 可以直接送 A,不用多一步送 RD，也可以用&或其它更快的运算替代加法



7.2 如何使得 a,b 相同时不再执行 SWAP

SUB 后同时更新了 FC 和 FZ，可以通过增加一条指令置 FZ 为 0 来屏蔽 FZ 的影响，如取反指令会改变 FZ 而不改变 FC, 可以加一条双字长指令，对立即数 00H 取反以修正 FZ

7.3 C 字段作用

进行选择逻辑判断，使微程序产生分支

7.4 数据送去哪里是由哪个字段决定的

A 字段。三个字段分别决定数据去哪里、从哪里来、微程序选择控制

7.5 执行双字长指令后如何利用相对寻址跳转到上一条指令

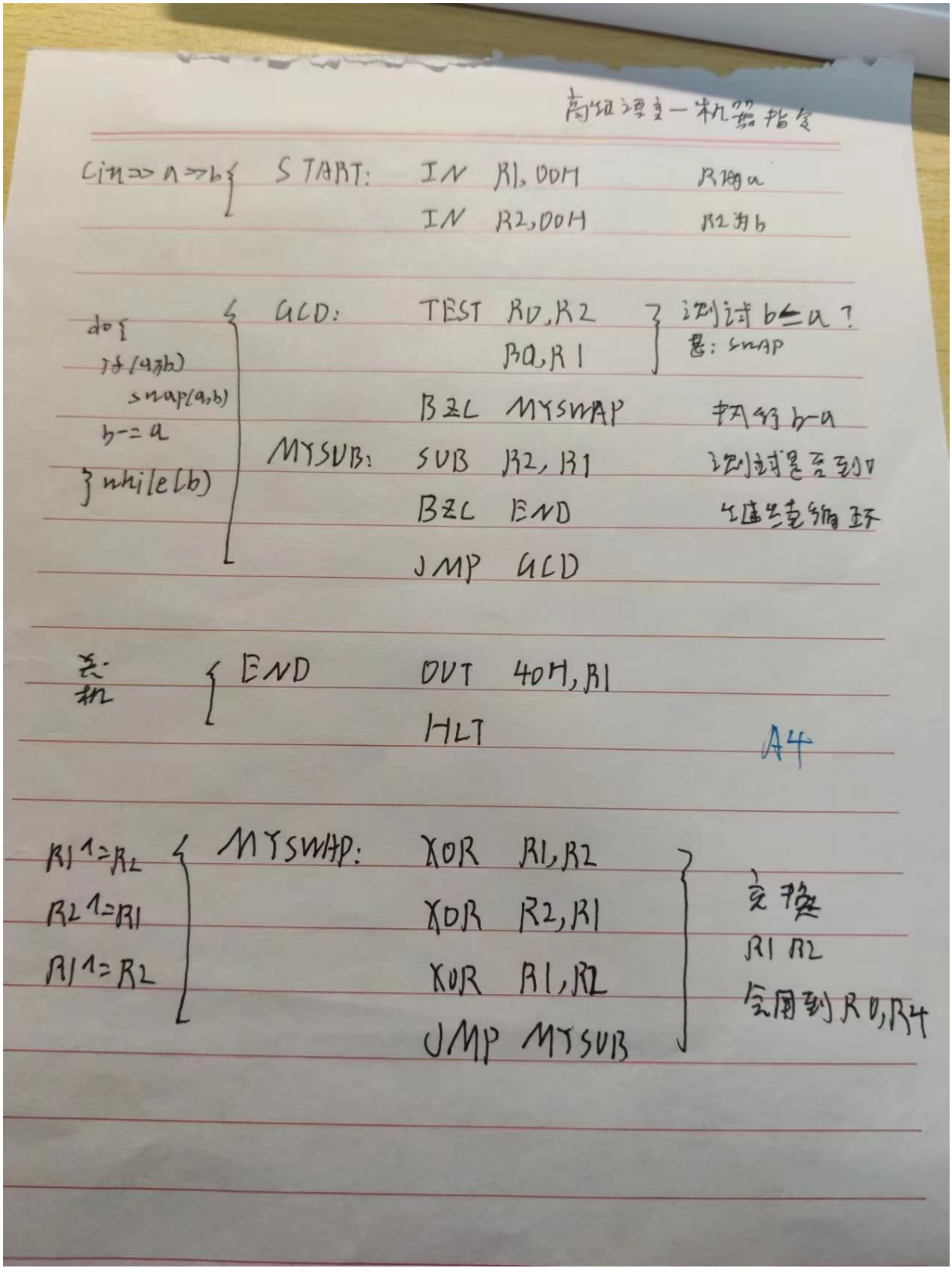
相对寻址寻址模式：11

执行双字长指令时首先加载第一字节控制指令时 PC+1，执行控制指令过程中又会执行 PC->AR,PC+1, 此时 PC 已经指向双字长指令的下一条指令入口地址。所有想要返回上一条指令需要另 PC-3（假设上一条指令是单字节，实验提问时举得例子是单字节的）。

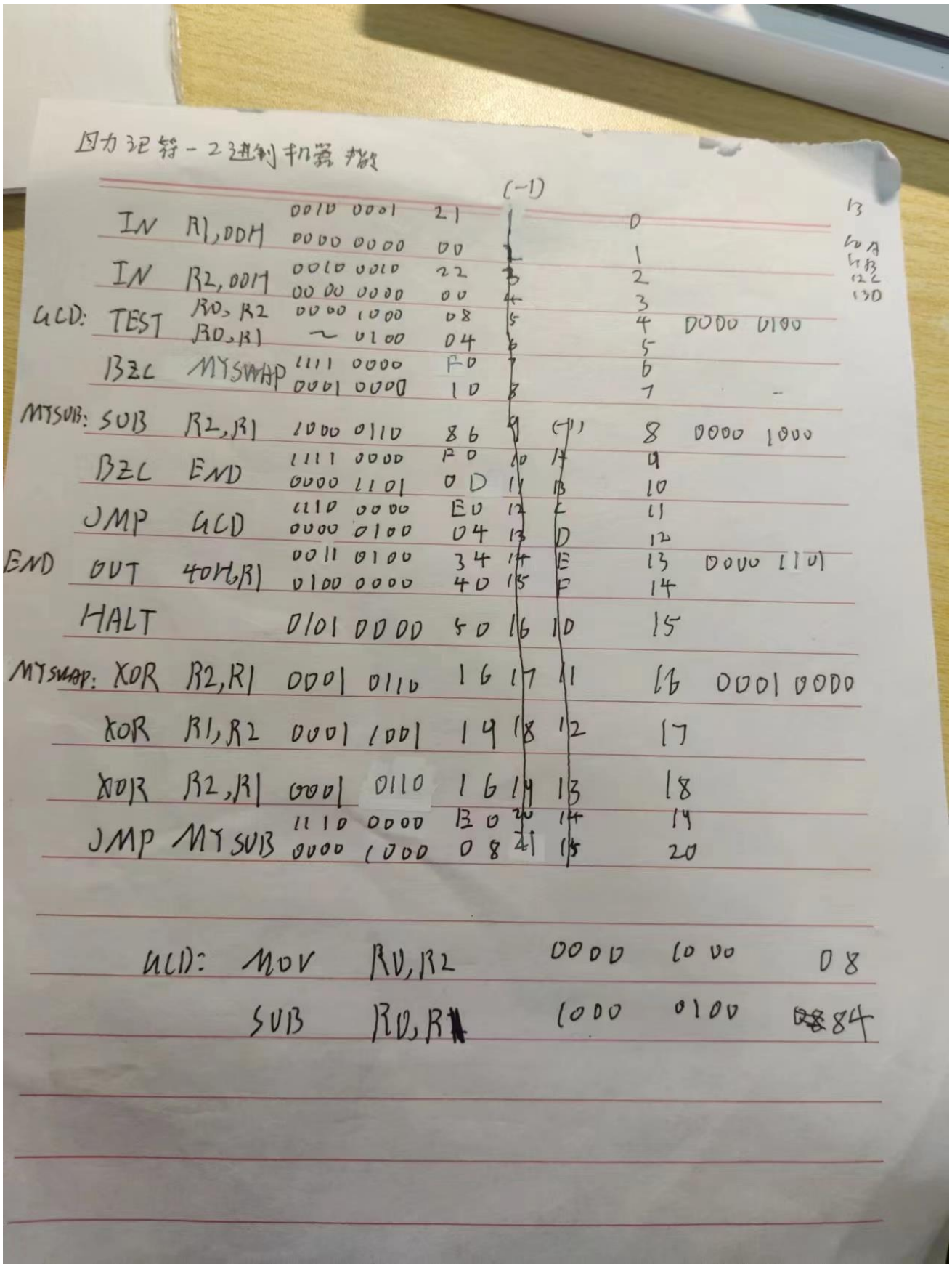
所以综上所述应该是一条寻址模式控制字段为 11，立即数为-3 的补码的双字长指令

八、 附：实验过程中间草稿

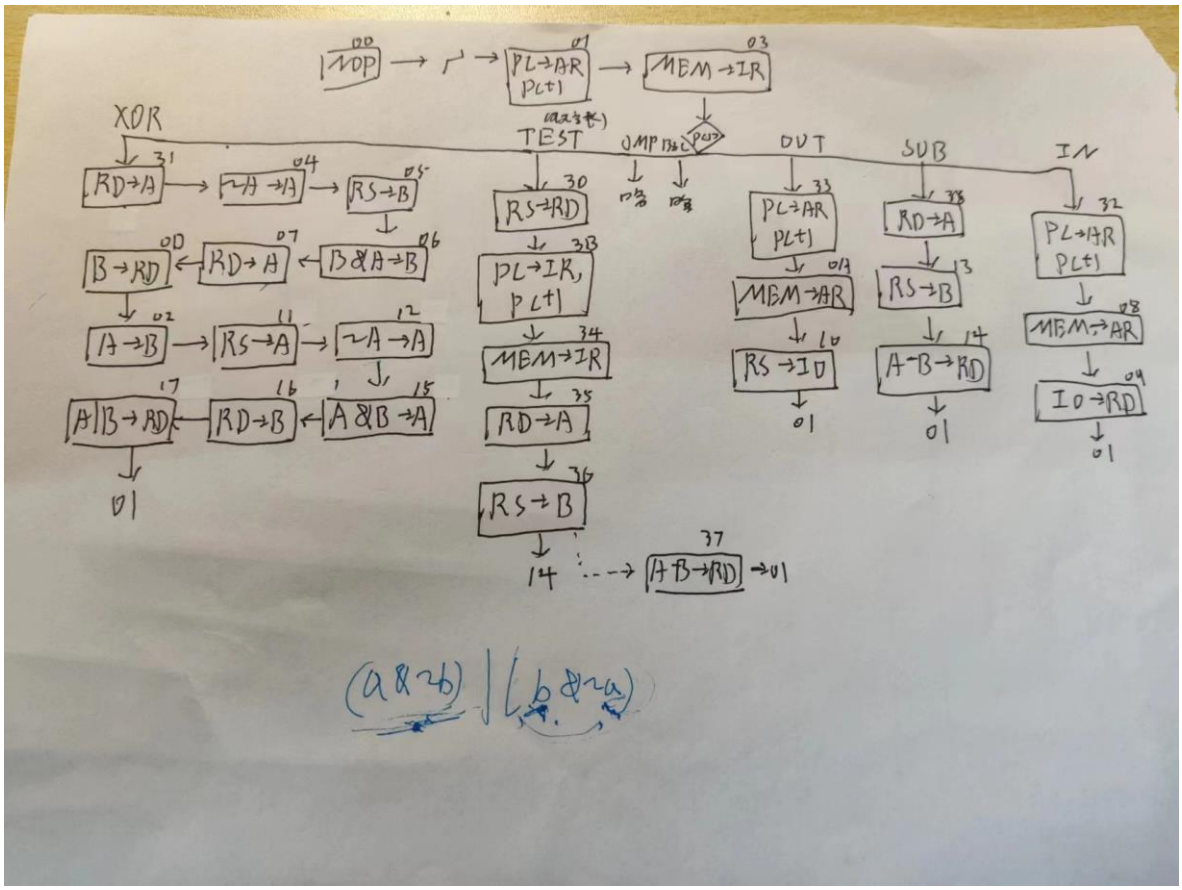
- 高级语言-机器指令



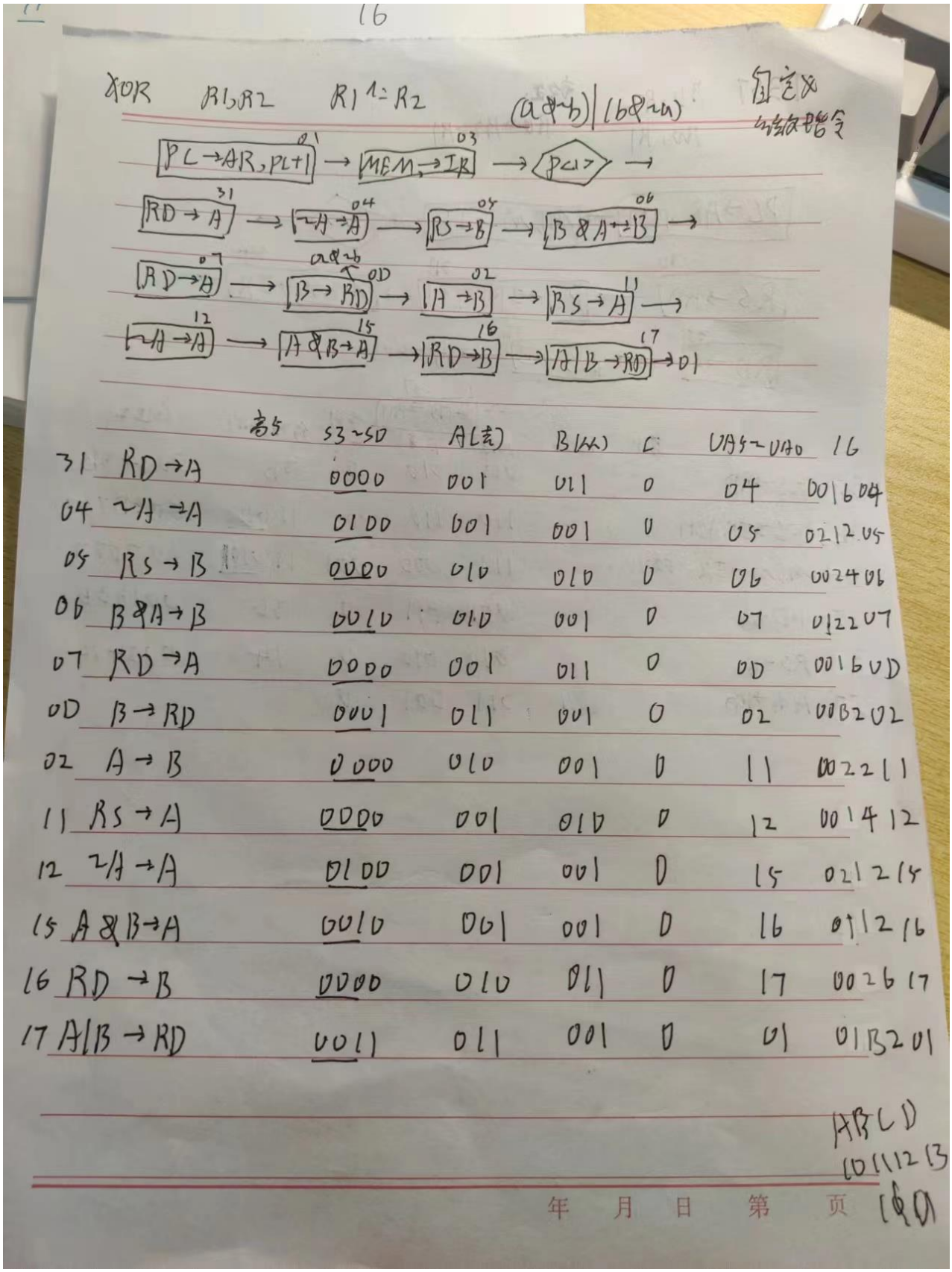
- 助记符-二进制机器指令


助记符-二进制机器指令
IN R1,00H
IN R2,001H
GCD: TEST R0,R2
R0,R1
BZC MYSWAP
MYSUB: SUB R2,R1
BZC END
JMP GCD
END OUT 40H,R1
HALT
MYSWAP: XOR R2,R1
XOR R1,R2
XOR R2,R1
JMP MYSUB
GCD: MOV R0,R2
SUB R0,R1

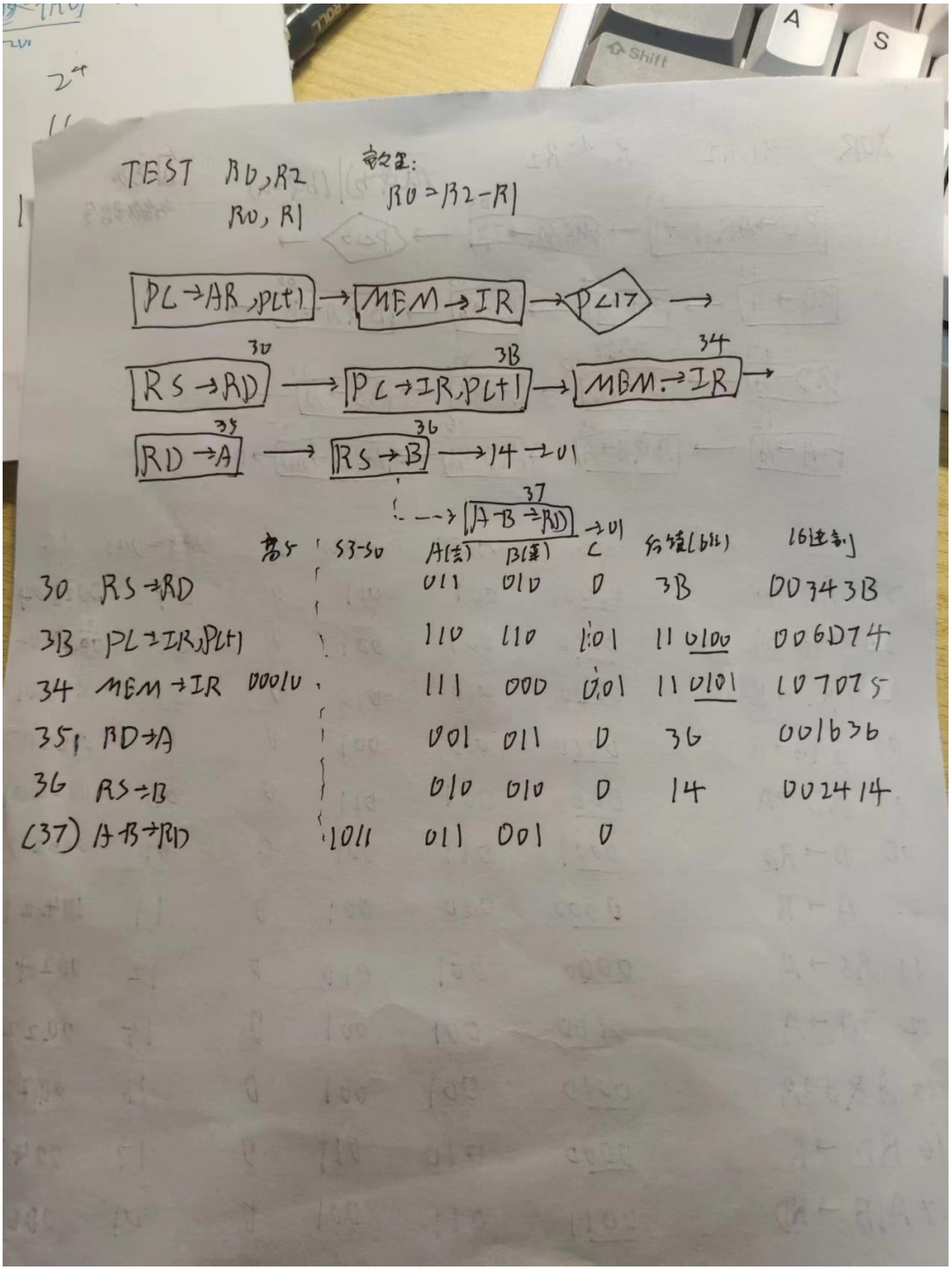
- 微程序框图（TEST 后被改为 MOV）



- 新增微指令框图及二进制代码
  - XOR



■ TEST（弃用）


TEST RD,RS
RD, RS
RD=RS-RD
PC→AR,PC+1
MEM→IR
PC+1
RS→RD
30
PC→IR,PC+1
3B
MEM→IR
34
RD→A
35
RS→B
36
14→01
A-B→RD
37
30 RS→RD 0011 010 0 3B 003A3B
3B PC→IR,PC+1 110 110 101 11 0100 006D74
34 MEM→IR 0001 111 000 101 11 0101 007075
35 RD→A 001 011 0 36 001636
36 RS→B 010 010 0 14 002414
(37) A-B→RD 1011 011 001 0