浙江工业大学

计算机组成原理课程设计

姓　　名 _____

班　　级 _____

学　　号 _____

提交日期 2025.6.8

目录

一、实验目的 1
二、实验内容 1
三、实验原理 1
3.1、指令系统及分析 1
3.2、指令框图及分析 2
3.3、指令系统对应微程序二进制代码及分析 3
3.4、机器程序及分析 5
四、实验步骤 10
4.1、微程序写入及校验 10
4.2、机器程序写入及校验 13
五、实验结果及分析 15
5.1、演示程序一 15
5.2、演示程序二 17
六、实验问题及思考 18
七、实验验收答辩环节问题和解答 19

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 组合数运算 。

三、实验原理

3.1、指令系统及分析

组合数的计算本质上依赖于乘法和除法运算，借助循环结构便可实现完整求解。由于实验所用的运算器并不支持直接的乘除法指令，因此采用移位与加法来模拟乘法，利用减法模拟除法过程。

为了在已有的模型机指令体系上实现乘法与除法的模拟，需使用到以下指令：CMP（比较）、TEST（测试）、SHR（逻辑右移）和 SHL（逻辑左移）。这四条均属于运算类指令，均为单字节，采用寄存器直接寻址方式。

其中，CMP 是算术类指令，通过对两个操作数进行减法运算（RD - RS）实现比较功能，仅修改 FC 和 FZ 标志位，不返回运算结果；TEST 是逻辑运算类指令，用于对两个操作数进行按位与操作（RD & RS），常用于判断某个位是否为 1，只影响 FZ 标志位，也不返回结果；SHR 与 SHL 属于移位类指令，分别完成对操作数逻辑右移一位和左移一位的操作，影响 FZ 标志位，且将结果写回 RD（即 RD >> 1 → RD 和 RD << 1 → RD）。

考虑到实验平台最多只支持 16 个微程序入口，最终通过替换未用到的 AND、OR、RR 三条指令以及利用保留的一个微程序入口，完成了新指令系统的设计，具体指令对照关系见下表：

指令助记符	指令格式				指令功能
MOV RD,RS	0100	RS	RD		RS -> RD
ADD RD,RS	0000	RS	RD		RD + RS -> RD
SUB RD,RS	1000	RS	RD		RD - RS -> RD
CMP RD,RS	1011	RS	RD		RD - RS
TEST RD,RS	0001	RS	RD		RD & RS
SHR RD,RS	1001	**	RD		RD >> 1 -> RD
SHL RD,RS	1010	**	RD		RD << 1 -> RD
INC RD	0111	**	RD		RD + 1 -> RD
LAD M D,RD	1100	M	RD	D	E -> RD
STA M D,RS	1101	M	RD	D	RD -> E
JMP M D	1110	M	**	D	E -> PC
BZC M D	1111	M	**	D	当 FC 或 FZ=1 时,E -> PC
IN RD,P	0010	**	RD	P	[P] -> RD
OUT P,RS	0011	RS	**	P	RS -> [P]
LDI RD,D	0110	**	RD	D	D -> RD
HLT	0101	**	**		停机

表 1 指令描述表

3.2、指令框图及分析

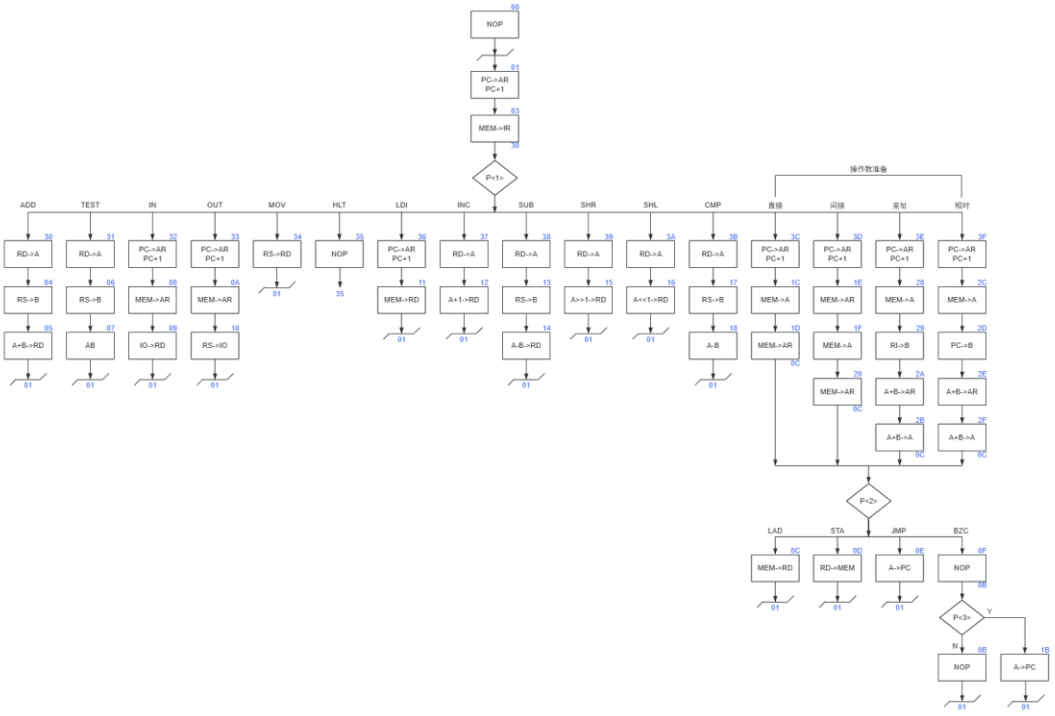

图 1 指令框图

如上图所示的指令框图，涵盖了该指令系统设计中的 16 条指令。相较于原先的复杂模型机，其中保留的微程序入口已被替换为 CMP 指令的入口地址，原先未被使用的 AND、OR、RR 指令则依次被替换为 TEST、SHR 和 SHL 指令。

CMP 指令的微指令起始地址为 3B，其执行过程占用 3 个 CPU 周期，具体步骤为：首先将 RD 寄存器的内容送入暂存器 A，其次将 RS 寄存器的内容送入暂存器 B，最后通过算术逻辑单元执行 A 减 B 的运算。

TEST 指令的微程序起点为 31，其操作过程与 CMP 类似，不同之处在于最后一个周期执行的是 A 与 B 的按位与运算（A&B）。

SHR 指令的入口地址为 39，执行过程需要 2 个 CPU 周期，分别为将 RD 的值送入暂存器 A，以及对 A 进行逻辑右移（A>>1）并将结果写回 RD。

SHL 指令从地址 3A 开始，执行流程与 SHR 一致，唯一区别在于最后一步改为逻辑左移（A<<1）后将结果存入 RD。

3.3、指令系统对应微程序二进制代码及分析

依据实验中提供的微指令格式、所设计的指令系统以及对应的指令流程图，可以对每一条微指令进行编码，最终生成如下表列出的微程序二进制指令序列：

地址	十六进制表示	高五位	S3-S0	A 字段	B 字段	C 字段	UA5-UA0
00	00 00 01	00000	0000	000	000	000	000001
01	00 6D 43	00000	0000	110	110	101	000011
03	10 70 70	00010	0000	111	000	001	110000
04	00 24 05	00000	0000	010	010	000	000101
05	04 B2 01	00000	1001	011	001	000	000001
06	00 24 07	00000	0000	010	010	000	000111
07	01 02 01	00000	0010	000	001	000	000001
08	10 60 09	00010	0000	110	000	000	001001
09	18 30 01	00011	0000	011	000	000	000001
0A	10 60 10	00010	0000	110	000	000	010000
0B	00 00 01	00000	0000	000	000	000	000001
0C	10 30 01	00010	0000	011	000	000	000001
0D	20 06 01	00100	0000	000	001	100	000001
0E	00 53 41	00000	0000	101	001	101	000001
0F	00 00 CB	00000	0000	000	000	011	001011
10	28 04 01	00101	0000	000	010	000	000001
11	10 30 01	00010	0000	011	000	000	000001

12	06 B2 01	00000	1101	011	001	000	000001
13	00 24 14	00000	0000	010	010	000	010100
14	05 B2 01	00000	1011	011	001	000	000001
15	03 32 01	00000	0110	011	001	000	000001
16	03 B2 01	00000	0111	011	001	000	000001
17	00 24 18	00000	0000	010	010	000	011000
18	05 82 01	00000	1011	000	001	000	000001
1B	00 53 41	00000	0000	101	001	101	000001
1C	10 10 1D	00010	0000	001	000	000	011101
1D	10 60 8C	00010	0000	110	000	010	001100
1E	10 60 1F	00010	0000	110	000	000	011111
1F	10 10 20	00010	0000	001	000	000	100000
20	10 60 8C	00010	0000	110	000	010	001100
28	10 10 29	00010	0000	001	000	000	101001
29	00 28 2A	00000	0000	010	100	000	101010
2A	04 E2 2B	00000	1001	110	001	000	101011
2B	04 92 8C	00000	1001	001	001	010	001100
2C	10 10 2D	00010	0000	001	000	000	101101
2D	00 2C 2E	00000	0000	010	110	000	101110
2E	04 E2 2F	00000	1001	110	001	000	101111
2F	04 92 8C	00000	1001	001	001	010	001100
30	00 16 04	00000	0000	001	011	000	000100
31	00 16 06	00000	0000	001	011	000	000110
32	00 6D 48	00000	0000	110	110	101	001000
33	00 6D 4A	00000	0000	110	110	101	001010
34	00 34 01	00000	0000	011	010	000	000001
35	00 00 35	00000	0000	000	000	000	110101
36	00 6D 51	00000	0000	110	110	101	010001
37	00 16 12	00000	0000	001	011	000	010010
38	00 16 13	00000	0000	001	011	000	010011
39	00 16 15	00000	0000	001	011	000	010101
3A	00 16 16	00000	0000	001	011	000	010110
3B	00 16 17	00000	0000	001	011	000	010111
3C	00 6D 5C	00000	0000	110	110	101	011100
3D	00 6D 5E	00000	0000	110	110	101	011110
3E	00 6D 68	00000	0000	110	110	101	101000
3F	00 6D 6C	00000	0000	110	110	101	101100

表 2 微程序二进制代码表

3.4、机器程序及分析

组合数的计算方法有多种形式，利用其特性可以推导出一种递推式，其对应的 C++ 实现如下：

```
// C(n, m) = n! / (m! * (n - m)!)
int C(int n, int m)
{
    int res = 1;

    for (int i = 1; i <= m; ++i)
    {
        res = DIV(MUL(res, (n - m + i)), i); // res = res * (n - m + i) / i
    }

    return res;
}
```

从程序逻辑可以看出，只需将循环结构与乘除操作合理结合，便能完成组合数的求解，因此关键在于如何实现乘法和除法。

其中，乘法可通过位移与加法的配合进行模拟，对应的 C++ 实现如下：

```
// x * y 采用移位加法实现
int MUL(int x, int y)
{
    int res = 0;

    while (y != 0)
    {
        if (y & 1) // 判断 y 是否是奇数
        {
            res += x;
        }
        x = x << 1; // x 左移一位
        y = y >> 1; // y 右移一位
    }

    return res;
}
```

除法运算可以借助重复减法来模拟。由于组合数的特性保证了运算结果为整数，因此在实现过程中可以进行简化，其对应的 C++ 代码如下所示：

```
// x / y 用减法实现
int DIV(int x, int y)
{
```

```
    int res = 1;

    while (x - y != 0)
    {
        x = x - y;
        res = res + 1;
    }

    return res;
}
```

在完成乘法与除法的模拟后，将程序转换为对应的机器指令即可形成完整的机器级程序。以下是通过 C++ 模拟实现的机器程序：

```
#include <iostream>
using namespace std;

// 寄存器
int R0, R1, R2, R3;
// 主存储器的三个地址单元
int P60H, P61H, P62H;

int main()
{
    cout << DIV(4, 1);
    cin >> R0; // 模拟 IN R0, 00H（输入 n）
    cin >> R1; // 模拟 IN R1, 00H（输入 m）
    P60H = R1; // 模拟 STA R1, 60H（将 m 保存到主存 60H）

    R0 = R0 - R1; // 模拟 SUB R0, R1（R0 = n - m）

    R1 = 0x01; // 模拟 LDI R1, 01H（R1 初始化为 1，存储结果 res）
    R3 = 0x01; // 模拟 LDI R3, 01H（R3 初始化为 1，存储计数 i）

LOOP:
    P61H = R3; // 模拟 STA R3, 61H（把当前 i 值存入主存地址 61H）
    R2 = P60H; // 模拟 LAD R2, 60H（取回 m 值，放入 R2）
    R2++;      // 模拟 INC R2（R2 = m + 1）
    // 模拟 CMP R2, R3（i > m）
    if (R2 - R3 == 0)
    {
        goto RESULT; // 模拟 BZC RESULT
    }

    R0++;      // 模拟 INC R0（R0 = n - m + i）
    P62H = R0; // 模拟 STA R0, 62H（将 R0 当前值备份至主存 62H）
```

```
    // 开始乘法计算
    R2 = 0x00; // 模拟 LDI R2, 00H（R2 = 0，用于存储乘法结果）

MULLOOP:
    R3 = 0x00; // 模拟 LDI R3, 00H（清零 R3，用于比较是否乘法结束）
    // 模拟 CMP R0, R3
    if ((R0 - R3) == 0)
    {
        goto MULEND; // 模拟 BZC MULEND
    }

    R3 = 0x01; // 模拟 LDI R3, 01H（R3 = 1，用于判断乘数奇偶性）
    // 模拟 TEST R0, R3
    if ((R0 & R3) == 0) // 若 R0 为偶数
    {
        goto EVEN; // 模拟 BZC EVEN（跳到偶数处理部分）
    }

    R2 = R2 + R1; // 模拟 ADD R2, R1（累加乘法）

EVEN:
    R1 = R1 << 1; // 模拟 SHL R1（被乘数左移 1 位）
    R0 = R0 >> 1; // 模拟 SHR R0（乘数右移 1 位）
    goto MULLOOP; // 模拟 JMP MULLOOP（重复乘法过程）

MULEND:
    R1 = R2; // 模拟 MOV R1 R2（把乘法结果放入 R1）

    R0 = P62H; // 模拟 LAD R0, 62H（还原 R0 原值）
    R3 = P61H; // 模拟 LAD R3, 61H（还原 R3 原值）
    // 乘法计算结束

    // 开始除法计算
    R2 = 0x01; // 模拟 LDI R2, 01H（R2 初始为 1，用作除法结果）

DIVLOOP:
    R1 = R1 - R3; // 模拟 SUB R1, R3（被除数减去除数）
    if (R1 == 0)
    {
        goto DIVEND; // 模拟 BZC DIVEND（除尽）
    }

    R2++;           // 模拟 INC R2（除法结果加 1）
    goto DIVLOOP; // 模拟 JMP DIVLOOP（重复除法过程）
```

```
DIVEND:
    R1 = R2; // 模拟 MOV R1 R2（除法结果保存在 R1）
    // 除法计算结束

    R3++; // 模拟 INC R3（i++）

    goto LOOP; // 模拟 JMP LOOP（进入下一轮循环）

RESULT:
    cout << R1; // 模拟 OUT R1, 40H（输出结果）

    system("pause");
    return 0;
}
```

以下是流程图：

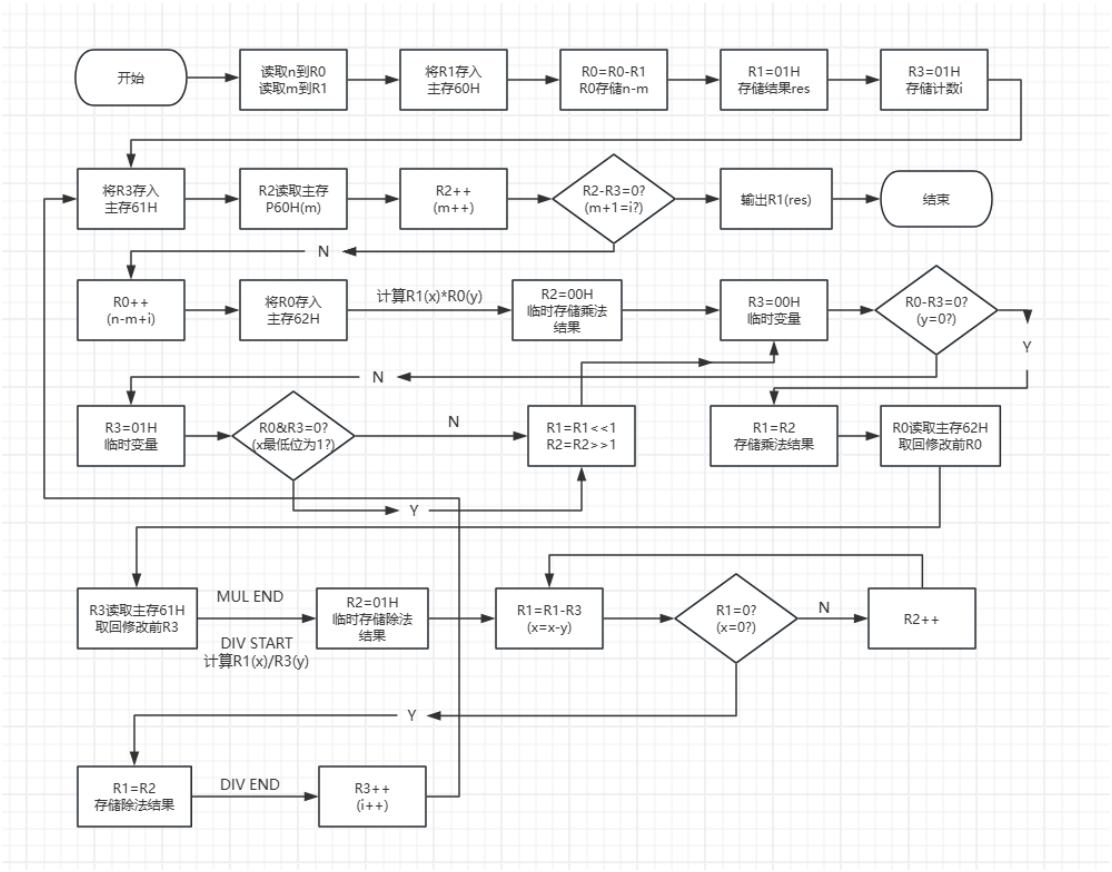

图 2 流程图

以下是机器程序：

地址	内容	二进制	助记符	说明
00	20	0010 0000	IN R0,00H	IN 读入 n
01	00	0000 0000		

02	21	0010 0001	IN R1,00H	IN 读入 m
03	00	0000 0000		
04	D1	1101 0001	STA R1,60H	将 R1 写入主存 60H
05	60	0110 0000		
06	84	1000 0100	SUB R0,R1	R0=R0-R1
07	61	0110 0001	LDI R1,01H	立即数 01H 送 R1
08	01	0000 0001		
09	63	0110 0011	LDI R3,01H	立即数 01H 送 R3
0A	01	0000 0001		
0B	D3	1101 0011	LOOP: STA R3,61H	开始循环: R3 写入主存 61H
0C	61	0110 0001		
0D	C2	1100 0010	LAD R2,60H	R2 读入主存 60H
0E	60	0110 0000		
0F	72	0111 0010	INC R2	R2++
10	BE	1011 1110	CMP R2,R3	比较 R2 和 R3
11	F0	1111 0000	BZC RESULT	R2=R3,表示循环结束
12	38	0011 1000		
13	70	0111 0000	INC R0	R0++
14	D0	1101 0000	STA R0,62H	R0 写入主存 62H
15	62	0110 0010		
16	62	0110 0010	LDI R2,00H	立即数 00H 送 R2
17	00	0000 0000		
18	63	0110 0011	MULLOOP: LDI R3,00H	开始乘法循环: 立即数 00H 送 R3 用于 CMP
19	00	0000 0000		
1A	BC	1011 1100	CMP R0 R3	比较 R0 和 R3
1B	F0	1111 0000	BZC MULEND	R0=R3,表示乘法循环结束
1C	27	0010 0111		
1D	63	0110 0011	LDI R3,01H	立即数 01H 送 R3 用于 TEST
1E	01	0000 0001		
1F	1C	0001 1100	TEST R0 R3	测试 R0 和 R3,判断 R0 奇偶
20	F0	1111 0000	BZC EVEN	R0 是偶,表示不需要+R1
21	23	0010 0011		
22	06	0000 0110	ADD R2,R1	R2=R2+R1
23	A1	1010 0001	EVEN: SHL R1	R1=R1<<1
24	90	1001 0000	SHR R0	R0=R0>>1
25	E0	1110 0000	JMP MULLOOP	跳转至 MULLOOP
26	18	0001 1000		

27	49	0100 1001	MULEND: MOV R1,R2	R1=R2
28	C0	1100 0000	LAD R0,62H	R0 读入主存 62H
29	62	0110 0010		
2A	C3	1100 0011	LAD R3,61H	R3 读入主存 61H
2B	61	0110 0001		
2C	62	0110 0010	LDI R2,01H	立即数 01H 送 R2
2D	01	0000 0001		
2E	8D	1000 1101	DIVLOOP: SUB R1,R3	开始除法循环: R1=R1-R3
2F	F0	1111 0000	BIZ DIVEND	R1=0,表示除法循环结束
30	34	0011 0100		
31	72	0111 0010	INC R2	R2++
32	E0	1110 0000	JMP DIVLOOP	跳转至 DIVLOOP
33	2E	0010 1110		
34	49	0100 1001	DIVEND: MOV R1,R2	R1=R2
35	73	0111 0011	INC R3	R3++
36	E0	1110 0000	JMP LOOP	跳转至 LOOP
37	0B	0000 1011		
38	34	0011 0100	RESULT: OUT 40H R1	OUT 写入 R1
39	40	0100 0000		
3A	50	0101 0000	HLT	停机
60	00	0000 0000		临时存储
61	00	0000 0000		临时存储
62	00	0000 0000		临时存储

表 3 机器程序表

四、实验步骤

4.1、微程序写入及校验

将微程序以指定格式写入 TXT 文件中，内容如下：

```
$M 00 000001 ; NOP
$M 01 006D43 ; PC->AR, PC+1
$M 03 107070 ; MEM->IR, P<1>
$M 04 002405 ; RS->B
$M 05 04B201 ; A+B->RD
$M 06 002407 ; RS->B
```

```
$M 07 010201 ; AB
$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+1->RD
$M 13 002414 ; RS->B
$M 14 05B201 ; A-B->RD
$M 15 033201 ; A>>1->RD
$M 16 03B201 ; A<<1->RD
$M 17 002418 ; RS->B
$M 18 058201 ; A-B
$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+B->AR
$M 2B 04928C ; A+B->A, P<2>
$M 2C 10102D ; MEM->A
$M 2D 002C2E ; PC->B
```

```
$M 2E 04E22F ; A+B->AR
$M 2F 04928C ; A+B->A, P<2>
$M 30 001604 ; RD->A
$M 31 001606 ; RD->A
$M 32 006D48 ; PC->AR, PC+1
$M 33 006D4A ; PC->AR, PC+1
$M 34 003401 ; RS->RD
$M 35 000035 ; NOP
$M 36 006D51 ; PC->AR, PC+1
$M 37 001612 ; RD->A
$M 38 001613 ; RD->A
$M 39 001615 ; RD->A
$M 3A 001616 ; RD->A
$M 3B 001617 ; RD->A
$M 3C 006D5C ; PC->AR, PC+1
$M 3D 006D5E ; PC->AR, PC+1
$M 3E 006D68 ; PC->AR, PC+1
$M 3F 006D6C ; PC->AR, PC+1
```

装载成功：

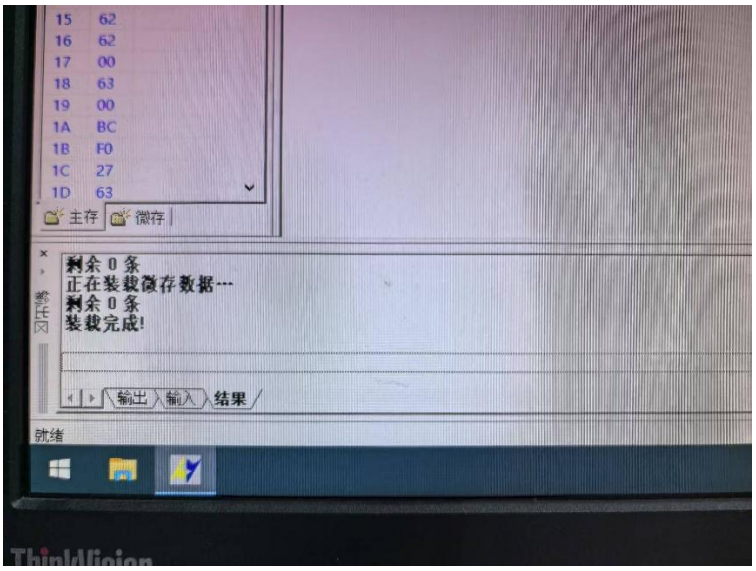

图 3 装载成功图

4.2、机器程序写入及校验

将机器程序以指定格式写入 TXT 文件中，内容如下：

```
$P 00 20 ; IN R0,00H IN 读入 n
$P 01 00
$P 02 21 ; IN R1,00H IN 读入 m
$P 03 00
$P 04 D1 ; STA R1,60H  将 R1 写入主存 60H
$P 05 60
$P 06 84 ; SUB R0,R1 R0=R0-R1
$P 07 61 ; LDI R1,01H  立即数 01H 送 R1
$P 08 01
$P 09 63 ; LDI R3,01H  立即数 01H 送 R3
$P 0A 01
$P 0B D3 ; LOOP: STA R3,61H  开始循环: R3 写入主存 61H
$P 0C 61
$P 0D C2 ; LAD R2,60H R2 读入主存 60H
$P 0E 60
$P 0F 72 ; INC R2 R2++
$P 10 BE ; CMP R2,R3  比较 R2 和 R3
$P 11 F0 ; BZC RESULT R2=R3,表示循环结束
$P 12 38
$P 13 70 ; INC R0 R0++
$P 14 D0 ; STA R0,62H R0 写入主存 62H
$P 15 62
$P 16 62 ; LDI R2,00H  立即数 00H 送 R2
$P 17 00
$P 18 63 ; MULLOOP: LDI R3,00H  开始乘法循环:  立即数 00H 送 R3 用于
CMP
$P 19 00
```

```
$P 1A BC ; CMP R0 R3  比较 R0 和 R3
$P 1B F0 ; BZC MULEND R0=R3,表示乘法循环结束
$P 1C 27
$P 1D 63 ; LDI R3,01H  立即数 01H 送 R3 用于 TEST
$P 1E 01
$P 1F 1C ; TEST R0 R3  测试 R0 和 R3,判断 R0 奇偶
$P 20 F0 ; BZC EVEN R0 是偶,表示不需要+R1
$P 21 23
$P 22 06 ; ADD R2,R1 R2=R2+R1
$P 23 A1 ; EVEN: SHL R1 R1=R1<<1
$P 24 90 ; SHR R0 R0=R0>>1
$P 25 E0 ; JMP MULLOOP  跳转至 MULLOOP
$P 26 18
$P 27 49 ; MULEND: MOV R1,R2 R1=R2
$P 28 C0 ; LAD R0,62H R0 读入主存 62H
$P 29 62
$P 2A C3 ; LAD R3,61H R3 读入主存 61H
$P 2B 61
$P 2C 62 ; LDI R2,01H  立即数 01H 送 R2
$P 2D 01
$P 2E 8D ; DIVLOOP: SUB R1,R3  开始除法循环: R1=R1-R3
$P 2F F0 ; BIZ DIVEND R1=0,表示除法循环结束
$P 30 34
$P 31 72 ; INC R2 R2++
$P 32 E0 ; JMP DIVLOOP  跳转至 DIVLOOP
$P 33 2E
$P 34 49 ; DIVEND: MOV R1,R2 R1=R2
$P 35 73 ; INC R3 R3++
$P 36 E0 ; JMP LOOP  跳转至 LOOP
$P 37 0B
```

```
$P 38 34 ; RESULT: OUT 40H R1 OUT 写入 R1
$P 39 40
$P 3A 50 ; HLT  停机
$P 60 00 ;  临时存储
$P 61 00 ;  临时存储
$P 62 00 ;  临时存储
```

装载成功：

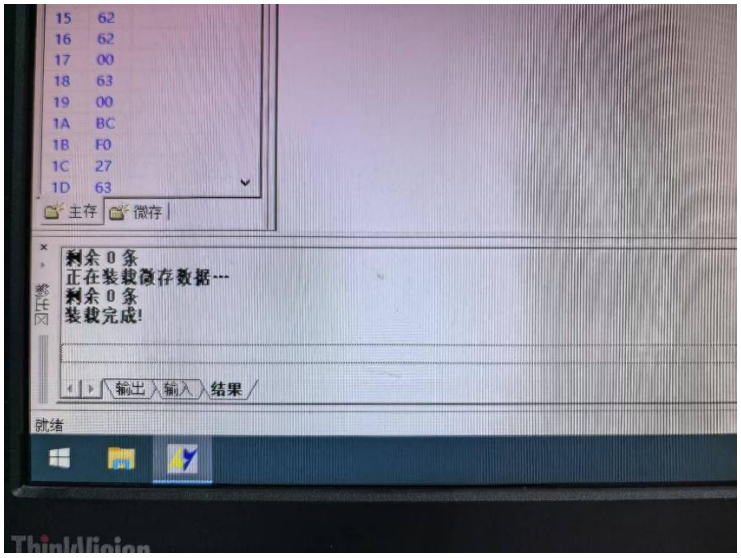

图 3 装载成功图

五、实验结果及分析

5.1、演示程序一

数据：

输入 1：0000 0100

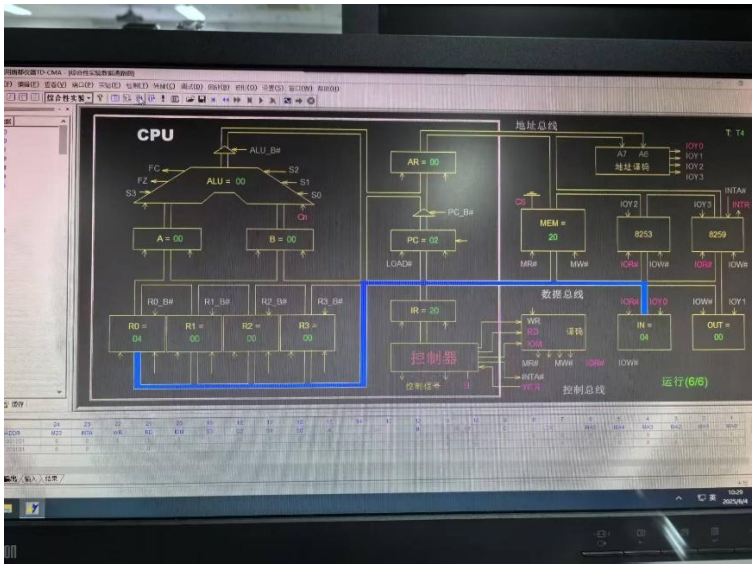

图 4 演示程序一输入 1

输入 2：0000 0011

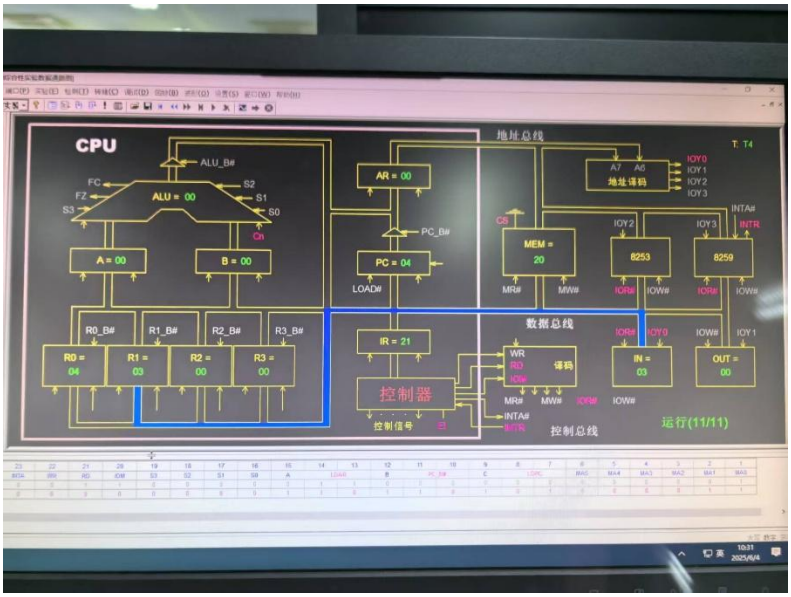

图 5 演示程序一输入 2

结果：

输出：4

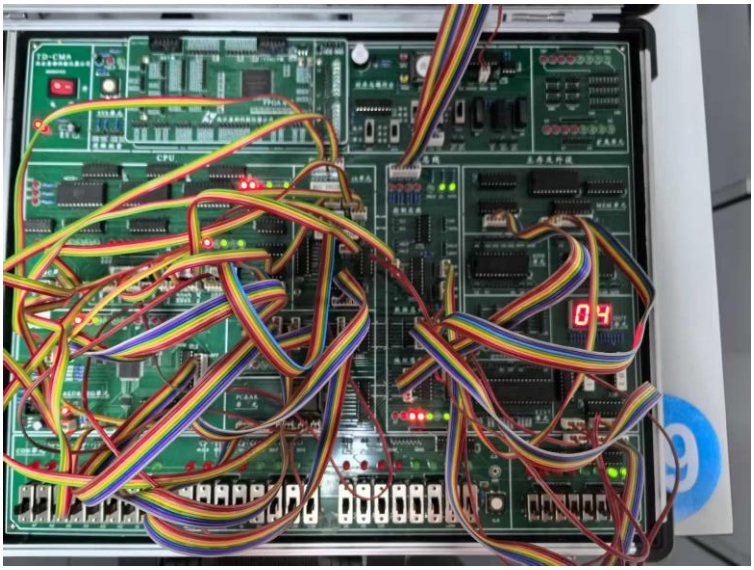

图 6 演示程序一输出

分析：$C_4^3 = 4$

5.2、演示程序二

数据：

输入 1：0000 0110

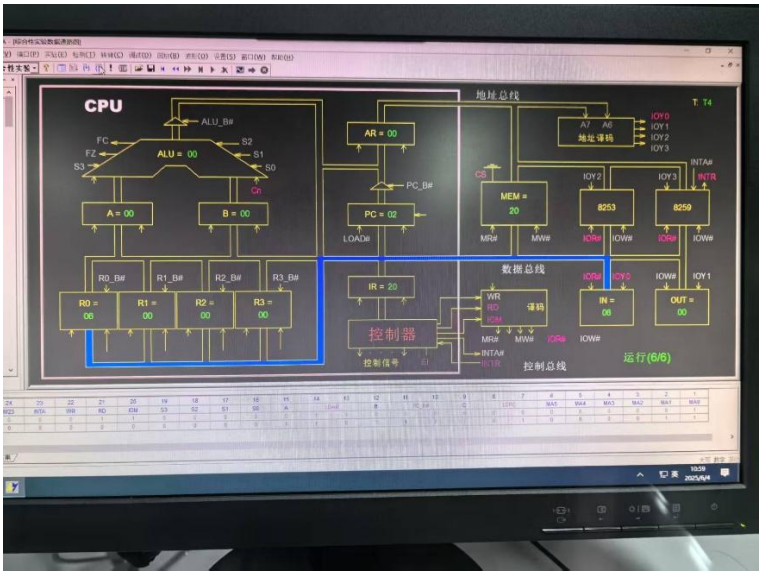

图 7 演示程序二输入 1

输入 2：0000 0010

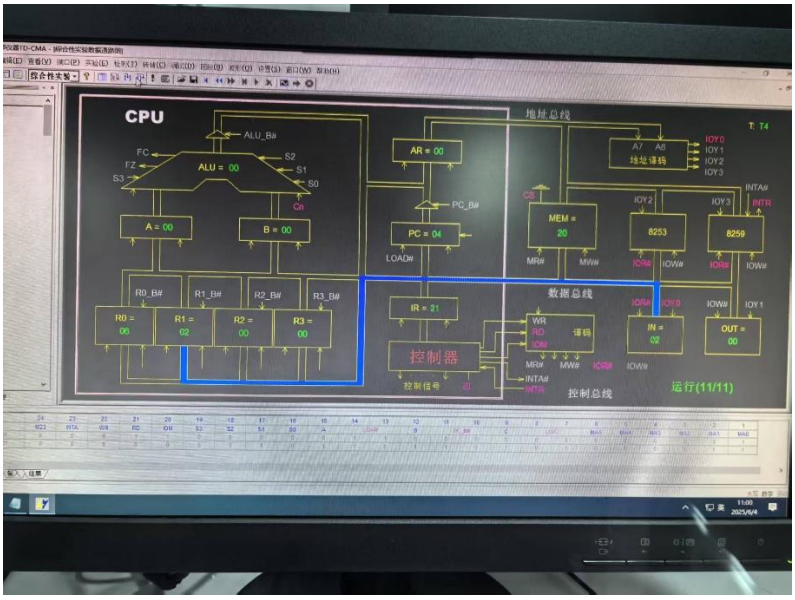

图 8 演示程序二输入 2

结果：

输出：0F（15）

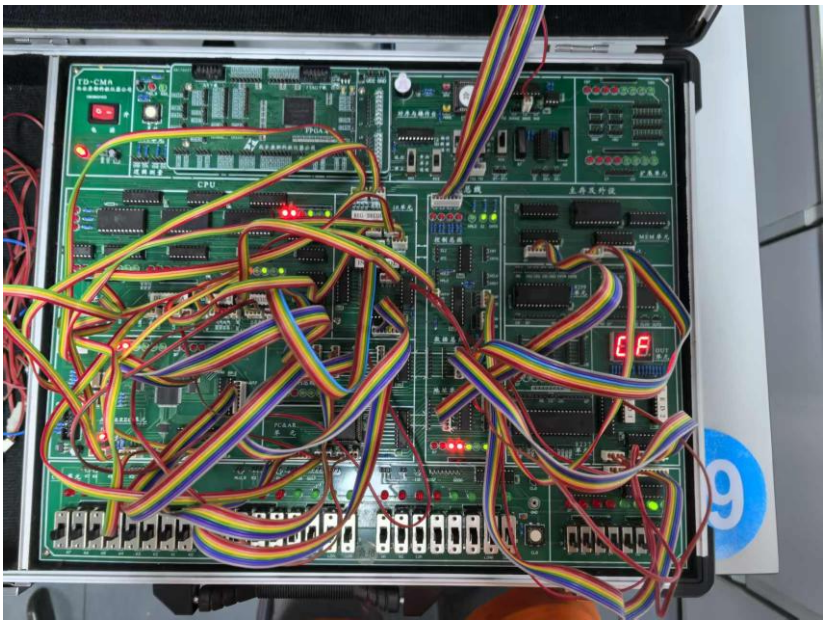

图 9 演示程序二输出

分析：$C_6^2 = 15$，相当于 16 进制中的 F

六、实验问题及思考

1、当前所实现计算机，是否完整？如果不完整，还缺少哪些部件？

答：从冯诺依曼体系结构来看，目前实现的计算机是完整的，具备五大基本

组成部分：存储器、运算器、控制器、输入设备和输出设备。但由于主存容量、控制存储空间以及寄存器数量的限制，在微程序层面上，某些复杂功能可能还无法完全实现。

2、当前所实现计算机，是否能实现除法运算？如果能，可通过哪些指令实现除法运算？

答：是可以进行除法运算的。尽管没有专门的除法指令，但可以通过组合使用 CMP、TEST、ADD、SUB、SHL、SHR 等运算指令，配合 JMP、BZC 等控制转移指令以及 LAD、STA、LDI 等数据传送指令，模拟实现恢复余数法的加减交替除法过程。

3、当前所实现计算机，还能实现哪些更复杂的计算？请举例说明。

答：除了基本运算，还能实现一些较复杂的算法，比如排列数、阶乘、斐波那契数列的生成、最小公倍数的求解，以及冒泡排序、选择排序等常见排序算法。

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？

答：当 CPU 在取指阶段获取一条双字长指令后，会在执行阶段通过将 PC 中的值送入地址寄存器 AR，再通过 PC+1 取出指令的第二个字节。这样就可以利用两个连续的存储单元完成一条双字长指令的执行。

七、实验验收答辩环节问题和解答

1. 实验仪器能够表示数的范围是多少？你设计的程序（组合数运算）能计算多大的数？

答：实验仪器是 8 位模型机，数据总线宽度为 8 位，因此数的表示范围为无符号整数：0 到 255。

计算结果 C(n, m) 必须 ≤ 255，否则会溢出（程序未设计溢出检测，因此当 C(n, m) > 255 时，结果会截断（只保留低 8 位），导致错误）。

2. 算术右移和逻辑右移的区别？你为什么要设计出逻辑右移/左移的指令？

答：逻辑右移是操作数向右移动指定位数，左侧空位补 0，右侧移出位丢弃；算术右移是操作数向右移动指定位数，左侧空位补符号位，右侧移出位丢弃。组合数运算需模拟乘法（移位加）。乘法算法依赖于逻辑移位处理无符号整数，SHL 用于左移被乘数（x << 1），SHR 用于右移乘数（y >> 1），测试最低位（奇偶性）。

3. 请你描述 BZC 指令的完整流程。

答：

- 取指阶段 00 000001 (NOP)
- PC 送地址寄存器 01 006D43 (PC->AR, PC+1)
- 内存读指令，将内存指令读入指令寄存器 IR 03 107070 (MEM->IR, P<1>判别)

P<1>关键作用：

识别操作码高 4 位(1111=BZC)

SE5=MA5, SE4=MA4, SE3=1, SE2=1, SE1=13, SE0=12 -> 微地址 3C (0011 1100)

- 取目标地址, 双字长指令第二字节 3C 006D5C (PC->AR, PC+1)
- 目标地址送暂存器 A 1C 10101D (MEM->A)
- 地址处理 1D 10608C (MEM->AR, P<2>判别)

P<2>关键作用：

SE5=MA5, SE4=MA4, SE3=MA3, SE2=MA2, SE1=15, SE0=14 -> 微地址 0C (0000 1100)

- 条件判别 0F 0000CB (NOP, P<3>判别)

P<3>关键作用：

SE5=MA5, SE4=FC∨FZ, SE3=MA3, SE2=MA2, SE1=MA1, SE0=MA0

当 FC 或 FZ=1 时：产生微地址 1B (0001 1011)

当 FC 且 FZ=0 时：产生微地址 0B (0000 1011)

- 条件成立时，将暂存器 A 中的目标地址写入 PC，实现跳转 1B 005341 (A->PC)
- 条件不成立时，继续顺序执行下一条指令 0B 000001 (NOP)

4. 请你写出 MEM->A 的微指令。并说明当 19 位为 0 时表示的含义。

答：00010 0000 001 000 000 011101

RD=1（读内存）、A 字段=001（LDA）。当执行时，数据总线内容加载到暂存器 A。

位 19 是 IOM 控制位，当 IOM=0 时访问内存（操作对象为主存储器）；当 IOM=1 时访问 I/O 设备（操作对象为输入/输出端口）。

在 MEM->A 中是指：IOM=0 确保读操作针对内存单元，而非 I/O 端口。