《计算机组成原理》课程设计

**复杂模型机的设计与实现**

2023年秋季学期

（起止时间：第17周~第17周）

指导教师：邹晓红、张秀杰、任大伟、王世明

2023年12月

**目 录**

[复杂模型机设计 1](#_Toc154127712)

[1. 项目任务 1](#_Toc154127713)

[2. 总体思路 1](#_Toc154127714)

[3. 技术路线 1](#_Toc154127715)

[4. 可行性分析 2](#_Toc154127716)

[4.1操作可行性 2](#_Toc154127717)

[4.1.1硬件可行性 2](#_Toc154127718)

[4.1.2软件可行性 2](#_Toc154127719)

[4.2经济可行性 2](#_Toc154127720)

[4.3技术可行性 3](#_Toc154127721)

[5. 复杂模型机 3](#_Toc154127722)

[5.1数据格式 3](#_Toc154127723)

[5.2指令设计 3](#_Toc154127724)

[5.2.1指令格式 3](#_Toc154127725)

[5.3数据寻址方式 4](#_Toc154127726)

[5.4 指令系统 5](#_Toc154127727)

[5.5 微指令设计 5](#_Toc154127728)

[6. 项目内容 9](#_Toc154127729)

[6.1实施方案 9](#_Toc154127730)

[6.2实施方法 10](#_Toc154127731)

[7. 项目实施 11](#_Toc154127732)

[7.1项目概述 11](#_Toc154127733)

[7.2项目目标 11](#_Toc154127734)

[7.3程序流程 11](#_Toc154127735)

[8. 项目成果 12](#_Toc154127736)

[8.1线路连接 12](#_Toc154127737)

[8.2机器码展示 13](#_Toc154127738)

[8.3运行结果 23](#_Toc154127739)

[9. 项目总结 25](#_Toc154127740)

[10. 参考文献 25](#_Toc154127741)

复杂模型机设计

**摘要**：在本实验中，我们设计和实现了一个复杂模型机来执行二进制一位乘法操作。通过该实验，我们深入研究了二进制乘法的原理，并将其应用于硬件电路设计中。我们进行了详细的设计过程和实验结果分析，展示了该复杂模型机的性能和准确性。

**关键词**：计算机组成原理；复杂模型机；二进制乘法

# 1. 项目任务

本课程研究项目是《计算机组成原理》课程学习的一个重要组成部分。通过课程研究项目的实施，使学生在掌握计算机组成课程理论的基础上，结合复杂模型机的设计与实现，深入了解并掌握组成计算机各主要功能部件的相关知识及工作原理，掌握研究（开发）技术报告的撰写方法，使学生具备独立设计计算机的基本能力，提高综合应用已有知识解决实际问题的能力，更好地培养计算机专业学生的专业技术能力和综合素质。

# 2. 总体思路

实现目标任务的分工如下

研究带符号位的二进制乘法原理：深入了解带符号位的二进制乘法规则和运算过程，包括正负数的表示方法、补码运算等。

实现指令集和控制逻辑：根据设计的复杂模型机结构，定义合适的指令集，并实现对应的控制逻辑，包括指令的解码、执行和存储器访问等。

性能评估与优化：对实现的复杂模型机进行性能评估，包括计算精度、运行速度等方面。根据评估结果，进行必要的优化和改进，以提高性能和效率。

撰写实验报告：详细记录和总结复杂模型机设计与实现的过程，包括设计思路、实验步骤、测试结果和分析等，并撰写一份完整的实验报告。

展示与演示：准备一个展示与演示的环节，向同学和老师展示实现的复杂模型机，并解释其工作原理和实现的功能。

# 3. 技术路线

项目可分为三个模块，确定研究目标、具体开发过程，最终结果及总结反馈。确定设计方向阶段主要包括查阅资料，确定设计题目，并进行可行性分析。具体开发过程是项目开发的重要部分，包括对代码的编写，分析，翻译和测试。最终结果及总结是对开发项目的过程和结果进行总结，总结调试过程中发现的问题和不足之处，思考在将来的开发工作中的注意事项，并编写体现整个开发过程的报告。

# 4. 可行性分析

系统可行性分析分为以下两个方面：

操作可行性分析评估：对项目的软件硬件等方面进行评估，硬件包括复杂模型机中的运算器存储器等硬件的功能是否可以实现等，软件包括机器码和微指令的设计是否可以实现该二进制乘法操作。

技术可行性评估：对项目所需的技术方案和技术资源进行评估，包括技术的成熟度、可用性和可靠性等。通过评估技术可行性，可以确定项目在技术上是否能够实现，并预先识别潜在的技术风险。

经济可行性评估：对项目的经济成本和收益进行评估，包括项目的投资成本、运营成本和预期的经济收益。通过评估经济可行性，可以判断项目是否具有可盈利性或产生其他经济效益，并为决策者提供经济依据。

## 4.1操作可行性

### 4.1.1硬件可行性

实现二进制乘法需要用到的硬件包括以下几个：

1、加法器：二进制乘法可以转化为多次位移和加法操作。实验箱内的加法器可以完成以上操作。

2、寄存器和存储器：需要使用寄存器和存储器来存储中间结果和最终结果，以便于在多次加法和位移操作之间进行数据传递和存储。

3、控制逻辑电路：设计合适的控制逻辑来协调各个部件的工作，确保它们按照正确的顺序执行乘法运算的各个步骤。

### 4.1.2软件可行性

本次设计主要使用《计算机组成原理》知识，针对复杂模型机的指令系统进行设计，硬件使用的是唐都实验仪器,软件使用CMA软件，CMA软件可以识别翻译后的二进制代码，并且可以显示每一个周期的指令操作及数据流向，满足实验要求。因此只需要有连接唐都仪器实验箱的台式机即可满足项目开发要求。

## 4.2经济可行性

无经济消费。

## 4.3技术可行性

二进制乘法是计算机科学中的基础运算之一，其原理已经被广泛研究和应用。同时，硬件电路的设计和实现也是一项成熟的技术，已经有很多成熟的方法和工具可供使用。因此，从技术上来说，该项目是可行的。

# 5. 复杂模型机

## 5.1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：

## 5.2指令设计

模型机设计三大类指令共十五条，其中包括运算类指令、控制转移类指令，数据传送类指令。运算类指令包含三种运算，算术运算、逻辑运算和移位运算，设计有6条运算类指令，分别为：ADD、AND、INC、SUB、OR、RR，所有运算类指令都为单字节，寻址方式采用寄存器直接寻址。控制转移类指令有三条HLT、JMP、BZC，用以控制程序的分支和转移，其中HLT为单字节指令，JMP和BZC为双字节指令。数据传送类指令有IN、OUT、MOV、LDI、LAD、STA共6条，用以完成寄存器和寄存器、寄存器和I/O、寄存器和存储器之间的数据交换，除MOV指令为单字节指令外，其余均为双字节指令。

### 5.2.1指令格式

所有单字节指令（ADD、AND、INC、SUB、OR、RR、HLT和MOV）格式如下：

表5-1 单字节指令格式

|  |  |  |
| --- | --- | --- |
| ７６５４ | ３２ | １０ |
| 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：

表5-2 寄存器编码

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

IN和OUT的指令格式为：

表5-3 IN和OUT指令

|  |  |  |  |
| --- | --- | --- | --- |
| ７６５４（1） | ３２（1） | １０（1） | 7—0（2） |
| OP-CODE | RS | RD | P |

## 5.3数据寻址方式

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。

LDI的指令格式如下，第一字节同前一样，第二字节为立即数。

表5-4 LDI指令格式

|  |  |  |  |
| --- | --- | --- | --- |
| ７６５４（1） | ３２（1） | １０（1） | 7—0（2） |
| OP-CODE | RS | RD | data |

LAD、STA、JMP和BZC指令格式如下。

表5-5 跳转指令

|  |  |  |  |
| --- | --- | --- | --- |
| ７６５４（1） | ３２（1） | １０（1） | 7—0（2） |
| OP-CODE | M | RD | D |

其中M为寻址模式，具体见表5-7，以R2做为变址寄存器RI。

表5-6寻址方式

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

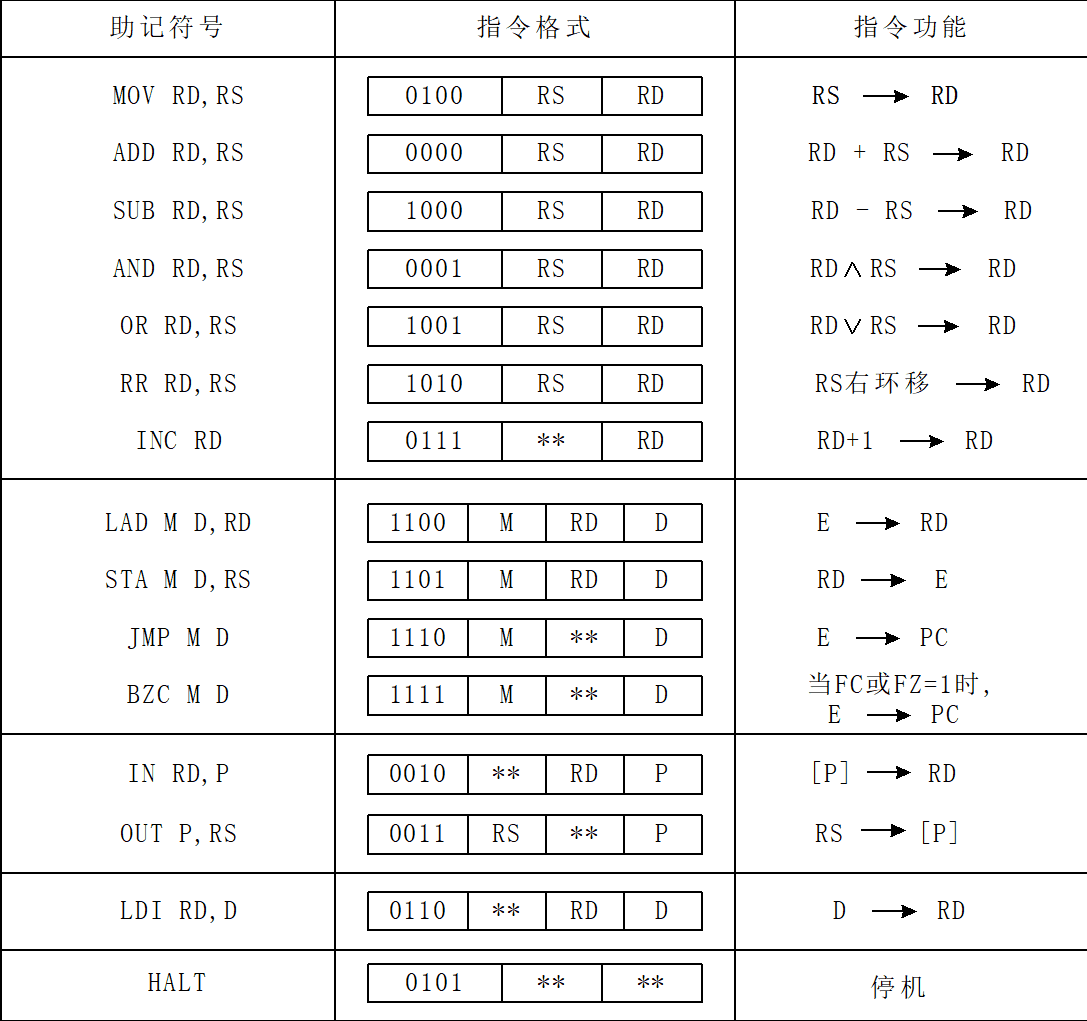
其中，OP-CODE为操作码，RS为源寄存器，RD为目的寄存器，D为I/O端口，占用一个字节。而对于LAD和JMP指令而言，引入了寻址方式选择。寻址方式就是寻找操作数或操作数地址的方式。寻址方式有立即、直接、间接、变址和相对寻址等，而本文设计的指令LAD和JMP均具备直接、间接、变址和相对寻址。因此，对于LAD和JMP指令其格式中第一字节的2、3位为寻址模式M选择，第二字节为位移量D。对于M寻址模式，规定：00、01、10、11分别代表直接寻址、间接寻址、RI变址寻址、相对寻址四种寻址方式，以R2作为变址寄存器RI。

## 5.4 指令系统

本模型机共有15条基本指令

其中包括单字节指令MOV、SUB等和双字节指令LDI、LAD、STA、等指令同样还包括IN和OUT输入输出指令和HLT停机指令。

如下表5-8列出了各条指令的格式、汇编符号、指令功能。

表5-7 指令描述

## 5.5 微指令设计

设计微指令结构应该追求的目标是：

（1）有利于缩短微指令字长度；

（2）有利于减小控制存储器的容量；

（3）有利于提高微程序的执行速度；

（4）有利于对微指令的修改；

（5）有利于提高微程序设计的灵活性。

基于上述考虑，我们对于微指令的设计应该尽量减少冗余

计算机微指令存在于微程序控制器中，每一条机器指令都是由若干条微指令序列来完成。因此，微指令是设计的核心，只有将每一条微指令理解透彻，才能完成指令系统的设计以及程序的正确执行。本文设计的微指令格式为24位。

本实验中要用到四个通用寄存器R3…R0，而对寄存器的选择是通过指令的低四位，为此还得设计一个寄存器译码电路，在IR单元的REG\_DEC（GAL16V8）中实现，如图5-3所示。

根据机器指令系统要求，设计微程序流程图及确定微地址，如图5-4所示。按照系统建议的微指令格式，见表5-9，参照微指令流程图，将每条微指令代码化，译成二进制代码表，见表5-10，并将二进制代码表转换为联机操作时的十六进制格式文件。

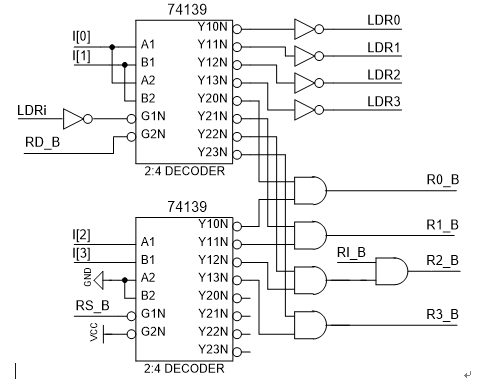


图5-1 寄存器译码原理图

表5-8 微指令格式



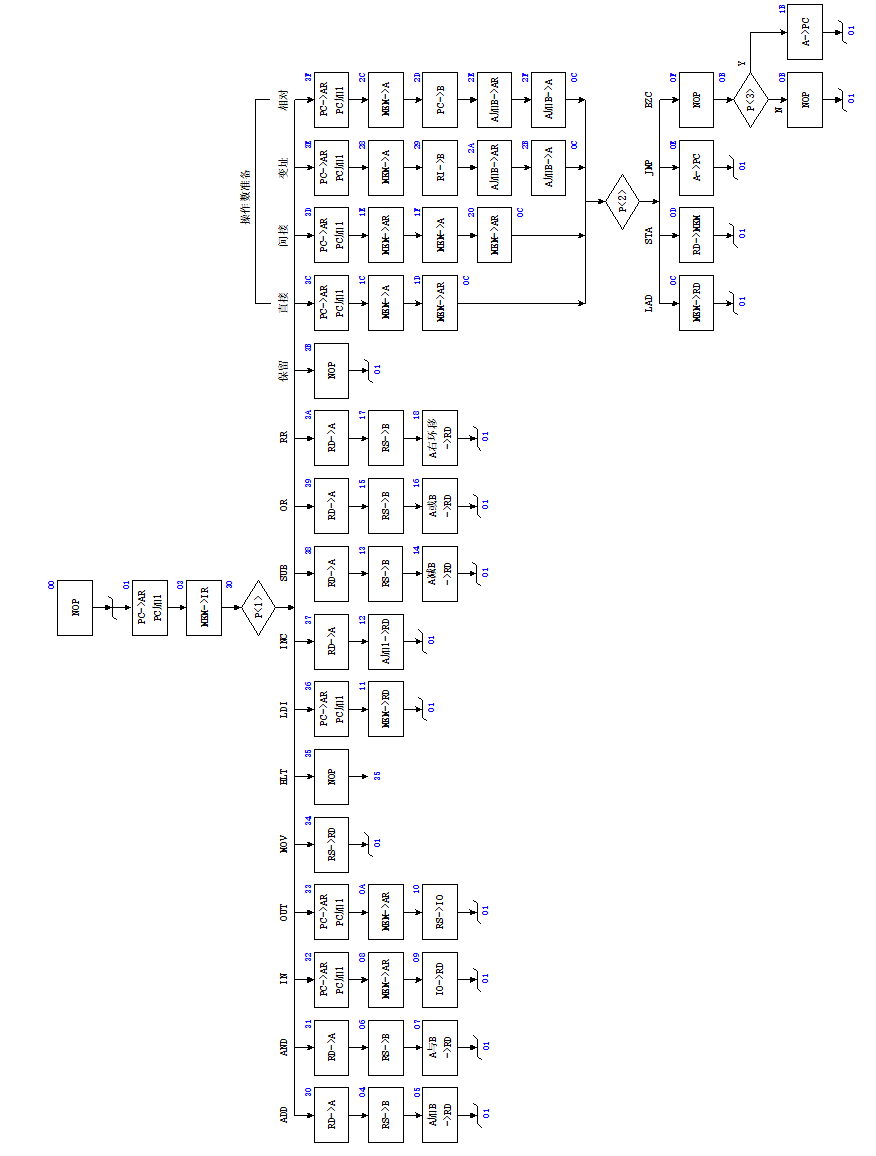


图5-2微程序流程图

表5-9二进制代码表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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 32 01 | 00000 | 0010 | 011 | 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 | 011 | 000 | 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 | 00 24 16 | 00000 | 0000 | 010 | 010 | 000 | 010110 |
| 16 | 01 B2 01 | 00000 | 0011 | 011 | 001 | 000 | 000001 |
| 17 | 00 24 18 | 00000 | 0000 | 010 | 010 | 000 | 011000 |
| 18 | 02 B2 01 | 00000 | 0101 | 011 | 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 17 | 00000 | 0000 | 001 | 011 | 000 | 010111 |
| 3B | 00 00 01 | 00000 | 0000 | 000 | 000 | 000 | 000001 |
| 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 |

# 6. 项目内容

## 6.1实施方案

项目分为确定选题、构思框架结构、完成程序设计、编码实现、运作调试5个阶段。

（1）确定选题：对复杂模型机以及给定的指令与寄存器进行分析，本组决定设计并实现带符号位的八位二进制乘法。

（2）构思框架结构：确定项目要实现的目标，确定程序框架结构。

（3）完成程序设计：完成程序的各个模块及跳转的设计。

（4）编码实现：通过编程在复杂模型机上完成上述设计的代码并翻译为机器码。

（5）运作调试：在设计的复杂模型机上运行程序，根据CMA上的模型运行情况判断代码的正确性，根据相应结果对程序进行调试。

## 6.2实施方法

（1）连接并测试线路：首先按图6-1所示连接图完成相应器材的连线，然后通过项目指导书所给测试样例测试线路。

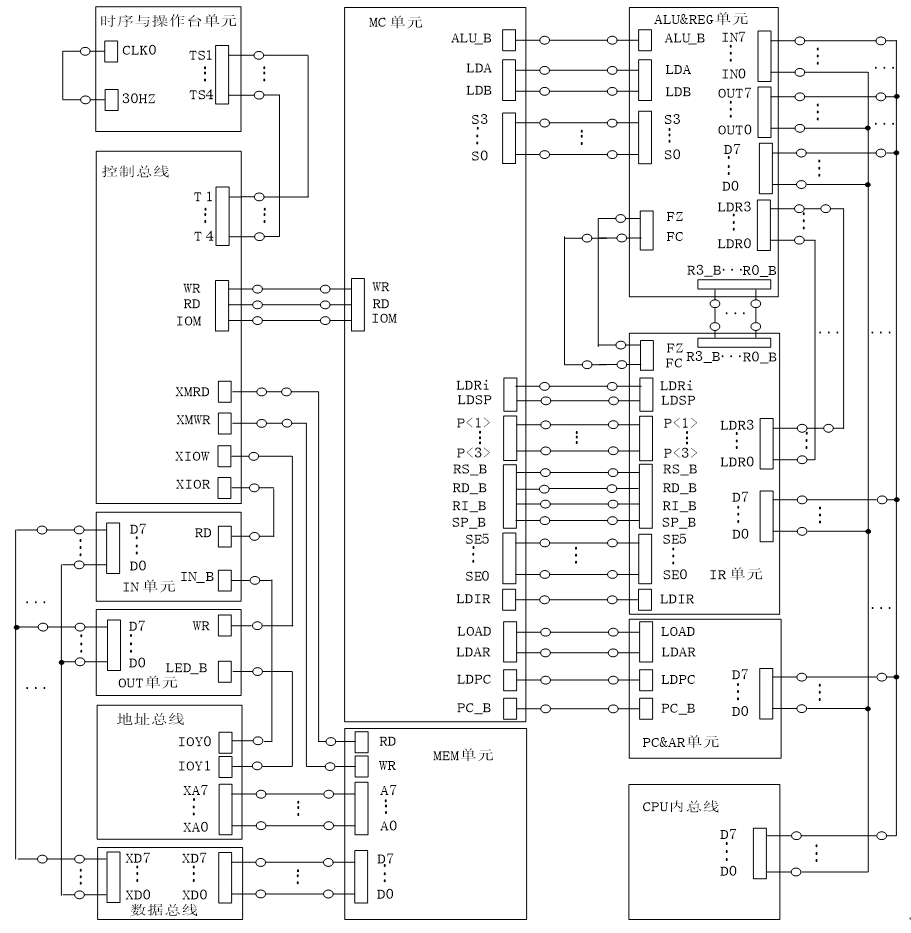


图6-1 线路连接图

1. 环境分析：分析所给复杂模型及特点。
2. 编码设计：在环境分析的基础上结合实施方案完成程序的设计编码。
3. 代码测试：将程序翻译完成的机器码、对应的测试数据及复杂模型机内的微指令机器码输入到一个txt文件中，然后将该文件加载到CMA中执行。
4. 调试与优化：根据模型机运行状况调试修改程序，继续进行代码测试直至程序完全正确。

# 7. 项目实施

## 7.1项目概述

项目实现复杂模型机用于计算二进制乘法。我们深入研究了二进制乘法的原理，并认真设计机器码使其实现所需要的操作和结果。我们进行了详细的设计过程和实验结果分析，展示了该复杂模型机的性能和准确性。

## 7.2项目目标

在给定寄存器数量及指令的限制下，完成复杂算法实现能通过完成算法在复杂模型机的实现，提升自己对计算机程序实现过程的理解，掌握计算机取指令，分析指令，执行指令等过程。对于八位二进制乘法，在输入单元，输入两个带符号数。通过一系列运算对两个数实现相乘运算，最后再输出结果，在输出结果时因为结果是16位，所以要显示高位再显示地位。

## 7.3程序流程

(1)、首先输入被乘数到R0，乘数到R1寄存器

(2)、再将R3和R4赋值为80H，再与R0和R1相与求得两个数的符号位。两两个数的符号位进行加法运算代替异或运算，最后存储在F5单元。

(3)、将R3和R4赋值7FH，与两个数相与获得被乘数与乘数的绝对值。

(4)、存储循环次数和进位标志。

(5)、取出乘数最后一位，检查最后一位是0或1。若是0，跳转noAdd代码段处处理。

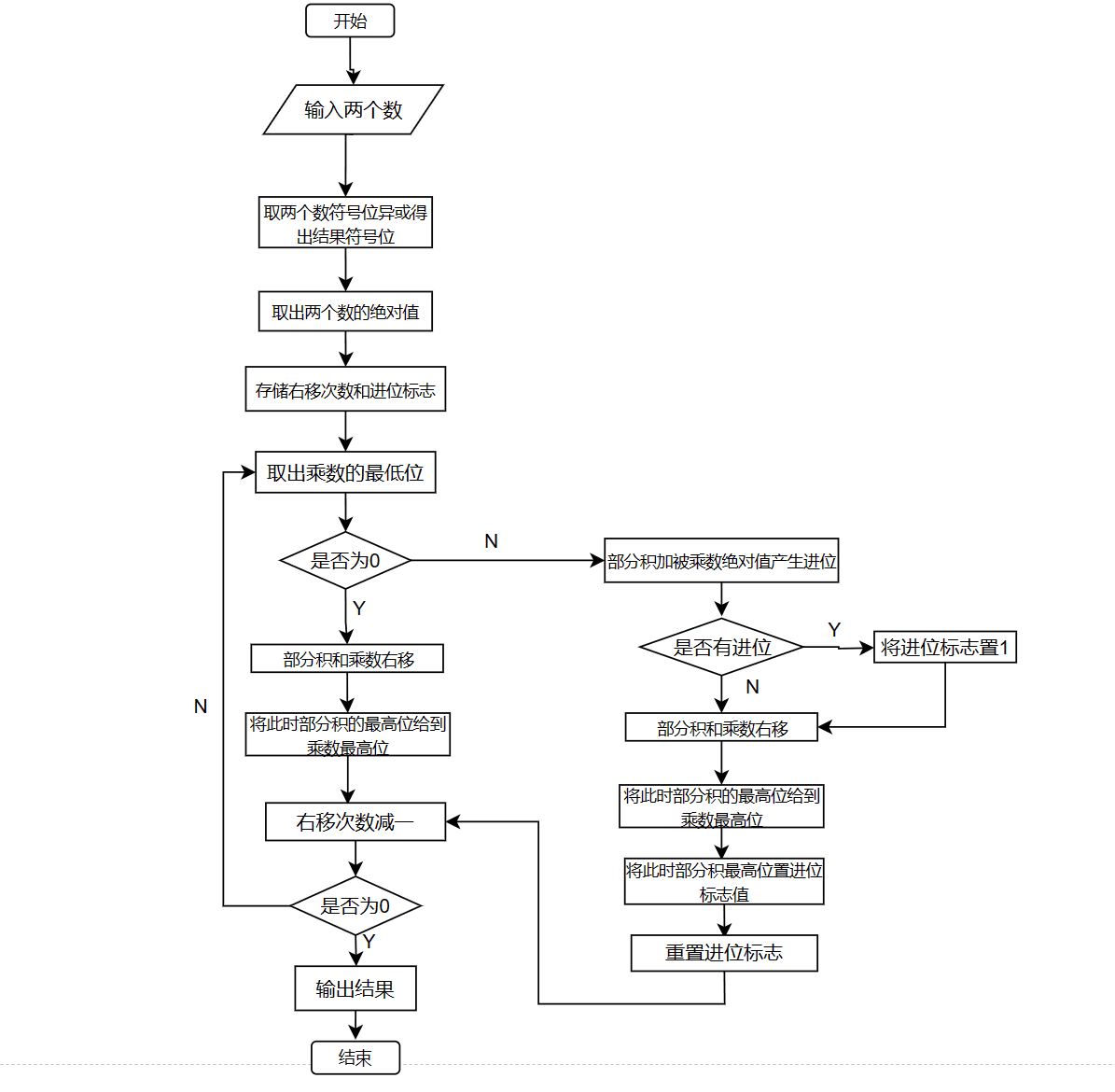
否则跳add代码段处理。

(6)、noAdd处理：将部分积和乘数右环移，取部分积最高位赋值给乘数最高位。跳转回BACK代码段

(7)、add处理：将部分积加被乘数，产生进位，将进位标志更改。将部分积个乘数右环移一位，将部分积最高位赋值给乘数最高位，然后将部分积最高位赋值为进位标志。跳转BACK代码段。

(8)、BACK处理：取出此时的循环次数，和总循环此数相比较，若为0则跳转OUT单元输出结果。否则跳回第5步骤。

(9)、OUT代码：取出部分积和乘数和存储在存储单元中的最终符号位，相与后在数码管单元输出。

图7-1 八位带符号的二进制乘法流程图

# 8. 项目成果

## 8.1线路连接

根据所给连接图对实验箱进行连线：

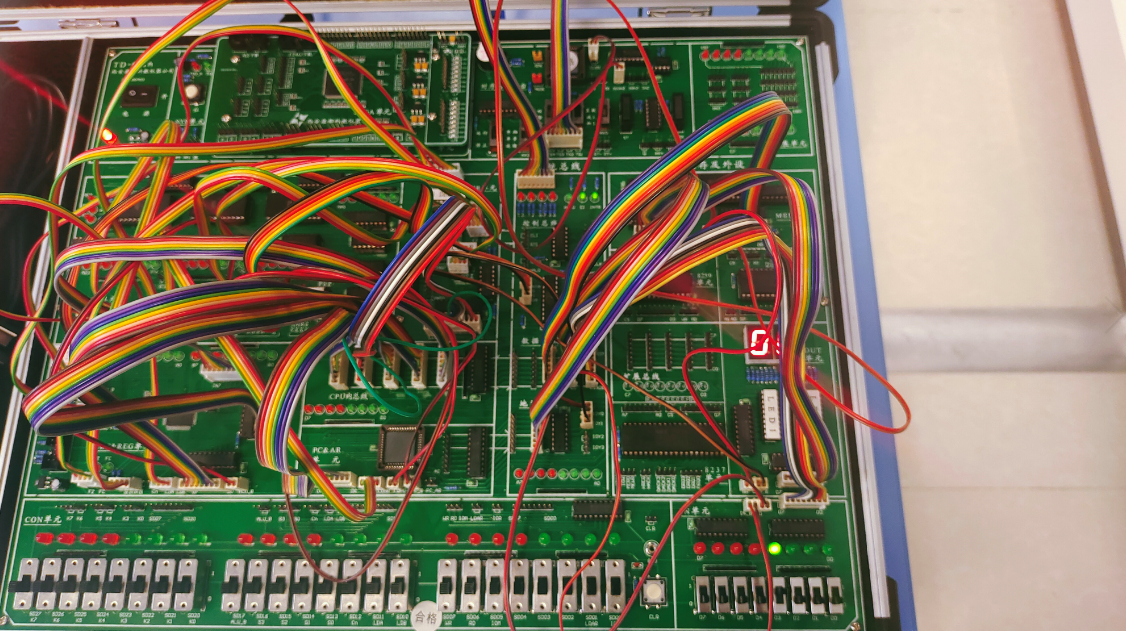


图8-1 实验连线图

## 8.2机器码展示

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* //

;/\* 该机器码和微指令用于实现八位（1 + 7）带符号位的二进制乘法 //

;\* 操作步骤： //

;\*1、 在IN单元输入第一个数送入R0单元 //

;\*2、在IN单元输入第二个数到R1单元 //

;\*3、点击CMA界面的连续运行按钮，就会在数码管单元分次显示高八位和低八位结果 //

;\*小组大概分工思路：刘星恺负责输入输出部分，刘广淼负责取出符号位和绝对值以及移位运算 //

;\* 梁子康负责乘数末尾为0，刘运硕负责乘数末尾为1 //

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

$P 00 20 ;IN R0, 00H ;输入被乘数第一个数到R0寄存器

$P 01 00 ;

$P 02 21 ;IN R1, 00H ;输入乘数第二个数到R1寄存器

$P 03 00 ;

$P 04 62;LDI R2 80H R2 = 10000000 用于取符号位

$P 05 80

$P 06 63;LDI R3 80H 与上同理

$P 07 80

$P 08 12;AND R2, R0 进行与运算求取第一个数的最高位，即符号位

$P 09 17;AND R3, R1 同上

$P 0A 0E;ADD R2, R3 相加代替异或

$P 0B D2;STA 00 F5H, R2 把符号位的结果存储到F5单元

$P 0C F5

$P 0D 62;LDI R2, 7FH R2 = 01111111 用于取绝对值

$P 0E 7F;

$P 0F 63;LDI R3, 7FH 同上

$P 10 7F;

$P 11 18;AND R0, R2 进行与运算取绝对值

$P 12 1D;AND R1, R3 同上

$P 13 62 ;LDI R2, 00H 用R2来存储循环次数

$P 14 00 ;

$P 15 63 ;LDI R3, 00H 用R3来存储部分积

$P 16 00 ;

$P 17 D0 ;STA 00, F0H, R0 寄存器有限，将被乘数绝对值存储在F0单元中

$P 18 F0 ;

$P 19 60 ;LDI R0, 00H 初始化进位标志位为0

$P 1A 00 ;

$P 1B D0 ;STA 00H, F4H, R0 将进位标志存储在内存F4单元

$P 1C F4 ;

; 主方法

$P 1D 72 ;INC R2 进行循环比较前就先进行计数加一操作，7位乘法进行8次

$P 1E 60 ;LDI R0, 09H R0赋值为需要循环的次数+1

$P 1F 09 ;

$P 20 88 ;SUB R0, R2 比较循环次数

$P 21 F0 ; BZC Output 计算完毕跳转输出

$P 22 5B ;

$P 23 60 ;LDI R0,01H

$P 24 01

$P 25 14 ;AND RO,R1 R1和01H AND，获得乘数的最低位

$P 26 F0 ;BZC noAdd 乘数最低位是0，部分积和乘数只需右移 跳转地址2A

$P 27 2A

$P 28 E0 ;JMP add 乘数最低位是1，部分积需要加被乘数，然后部分积和乘数右移 地址49

$P 29 49

; 乘数最低为此时为0 跳转至此

;noAdd

$P 2A 60 ;LDI R0,01H;

$P 2B 01

$P 2C D2 ;STA 00H, F1H, R2 将循环次数存入内存F1单元

$P 2D F1

$P 2E A1 ;RR R1,R0 对y进行一位右环移

$P 2F A3 ;RR R3,R0 对部分积进行右环移

$P 30 62 ;LDI R2, 80H

$P 31 80

$P 32 1E ;AND R2,R3 将部分积和80H AND操作，取出部分积的最高位 （部分积的最低位是要放入乘数的最高位）

$P 33 F0 ;BZC set0 部分积最高位为0，需要把乘数最高位设置为0 地址为37

$P 34 37

$P 35 E0 ;JMP set1 部分积最高位为1， 需要把乘数最高位设置为1 地址为40

$P 36 40

;set0 部分积最高位为0执行将乘数最高位置0

$P 37 66 ;LDI R2, 7FH

$P 38 7F

$P 39 19 ;AND R1,R2 R1与7FH AND 将乘数最高位置0

$P 3A E0 ;JMP FC 跳转到FC子程序 判断进位

$P 3B 6E ;

;BACK 部分积最高位置完 回到Main程序

$P 3C C2 ;LAD 00H, F1H, R2 ;取出计数值

$P 3D F1

$P 3E E0 ;JMP Main

$P 3F 1D

;set1 部分积最高位为1执行将乘数最高位置1

$P 40 62 ;LDI R2, 80H

$P 41 80

$P 42 99 ;OR R1,R2 R1与80H OR 将乘数最高位置1

$P 43 E0 ;JMP B0 判断进位标志子程序

$P 44 83 ;

;BACK1部分积最高位置完,跳回Main程序

$P 45 C2 ;LAD 00H, F1H, R2 ; 取出计数值

$P 46 F1

$P 47 E0 ;JMP Main

$P 48 1D

;add

$P 49 C0 ;LAD 00, F0H, R0 ;从存储器中取出被乘数

$P 4A F0

$P 4B 03 ;ADD R3,R0 ;加被乘数

$P 4C F0 ;BZC 相加有进位C0

$P 4D 67 ;

;相加无进位直接到此处，或相加有进位设置进位标志位1后跳转此处

;C1

$P 4E D2 ;STA 00H, F1H, R2 ;存储计数值

$P 4F F1

$P 50 62 ;LDI R2, 80H

$P 51 80

$P 52 60 ;LDI R0,01H

$P 53 01

$P 54 A1 ;RR R1,R0 乘数右移一位

$P 55 A3 ;RR R3,R0 部分积右移一位

$P 56 1E ;AND R2,R3 取部分积最高位

$P 57 F0 ;BZC set0 部分积最高位为0跳转

$P 58 37

$P 59 E0 ;JMP set1 部分积最高位为1跳转

$P 5A 40

;Output:

$P 5B C0 ;LAD 00, F5H, R0

$P 5C F5

$P 5D 03 ;ADD R1,R0

$P 5E E0 ;JMP 98 输出结果的最高位6遍，才会显示一下

$P 5F 98;

$P 60 34 ;OUT 40H, R1

$P 61 40

$P 62 D3 ;STA 00H, F2H, R3

$P 63 F2

$P 64 D1 ;STA 00H, F3H, R1

$P 65 F3

$P 66 50 ;HALT

;相加有进位，置标志位为1:

;C0

$P 67 60 ;LDI R0 01H

$P 68 01 ;

$P 69 D0 ;STA 00H, F4H, R0 ;存储进位标志

$P 6A F4

$P 6B 80 ;SUB R0,R0 ;清空R0

$P 6C E0 ;JMP C1 ;相加有进位返回此处

$P 6D 4E

;判断进位标志位，判断是否置部分积最高位0

;FC:

$P 6E C0 ;LAD 00H, F4H, R0 ;取出进位标志位

$P 6F F4 ;

$P 70 62 ;LDI R2,01H

$P 71 01 ;

$P 72 82 ;SUB R2,R0 看标志位是否为1，如果是，跳到CROSS子程序

$P 73 F0 ;BZC CROSS

$P 74 7A ;

$P 75 62 ;LDI R2, 7FH

$P 76 7F ;

$P 77 1B ;AND R3,R2 ;7FH与R3 AND 将部分积最高位置0

$P 78 E0 ;JMP BACK

$P 79 3C ;

;CROSS: 进位标志为1，将部分积最高位置1

$P 7A 62 ;LDI R2, 80H

$P 7B 80 ;

$P 7C 9B ;OR R3, R2 ;部分积最高位置1

$P 7D 60 ;LDI R0, 00H ;重置标志位为0/

$P 7E 00 ;

$P 7F D0 ;STA 00H, F4H, R0 存储标志位

$P 80 F4 ;

$P 81 E0 ;JMP BACK

$P 82 3C ;

;判断进位标志位，置部分积最高位0

;FC1

$P 83 C0 ;LAD 00H, F4H, R0 ;取出进位标志位

$P 84 F4 ;

$P 85 62 ;LDI R2,01H

$P 86 01 ;

$P 87 82 ;SUB R2,R0 判断标志位

$P 88 F0 ;BZC B2 标志位为1跳转

$P 89 8F ;

$P 8A 62 ;LDI R2, 7FH

$P 8B 7F ;

$P 8C 1B ;AND R3,R2 ;标志位为0 部分积最高位 置0

$P 8D E0 ;JMP B1

$P 8E 45 ;

;CROSS1: 进位标志为1 部分积最高位 置1

$P 8F 62 ;LDI R2, 80H

$P 90 80 ;

$P 91 9B ;OR R3, R2 ;部分积最高位 置1

$P 92 60 ;LDI R0, 00H ;重置标志位为0

$P 93 00 ;

$P 94 D0 ;STA 00H, F4H, R0

$P 95 F4 ;

$P 96 E1 ;JMP BACK1

$P 97 45 ;

$P 98 3C ;OUT 40H,R3

$P 99 40

$P 9A 3C

$P 9B 40

$P 9C 3C

$P 9D 40

$P 9E 3C

$P 9F 40

$P A0 3C

$P A1 40

$P A2 3C

$P A3 40

$P A4 3C

$P A5 40

$P A6 3C

$P A7 40

$P A8 3C

$P A9 40

$P AA E0

$P AB 60

$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

## 8.3运行结果

运行分析：通过两次输入，输入乘数FF（-127）与被乘数7F（127），在输出结果时因为结果是16位，所以要显示高位（BF）再显示低位（01）。即FF \* 7F = BF01 :127 \* -127 = -16129，经过运行发现实验结果与分析无误。

输入：7F和FF

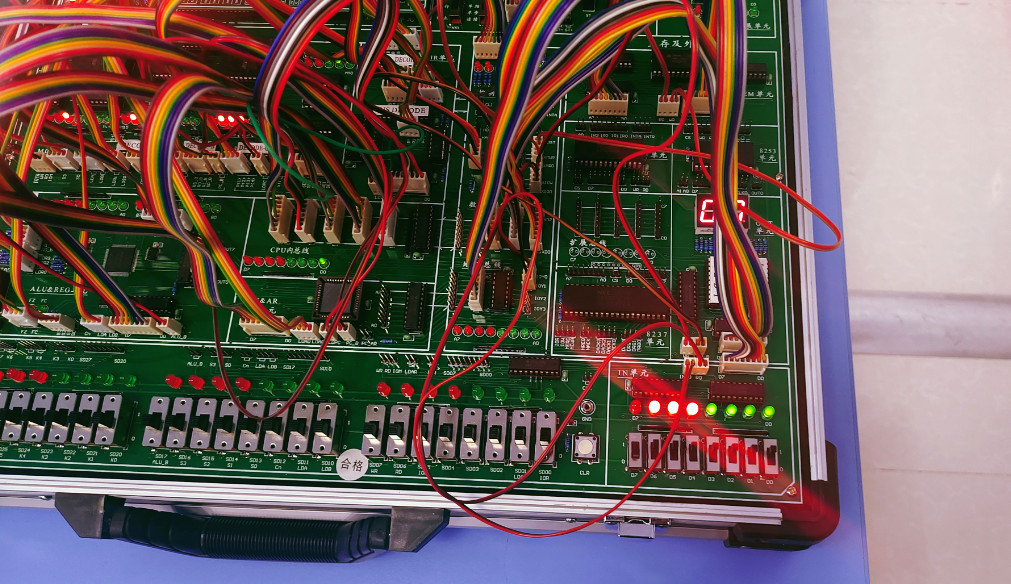


图8-2 输入（1）



图8-3 输入（2）

输出结果：

FF \* 7F = BF01 :127 \* -127 = -16129

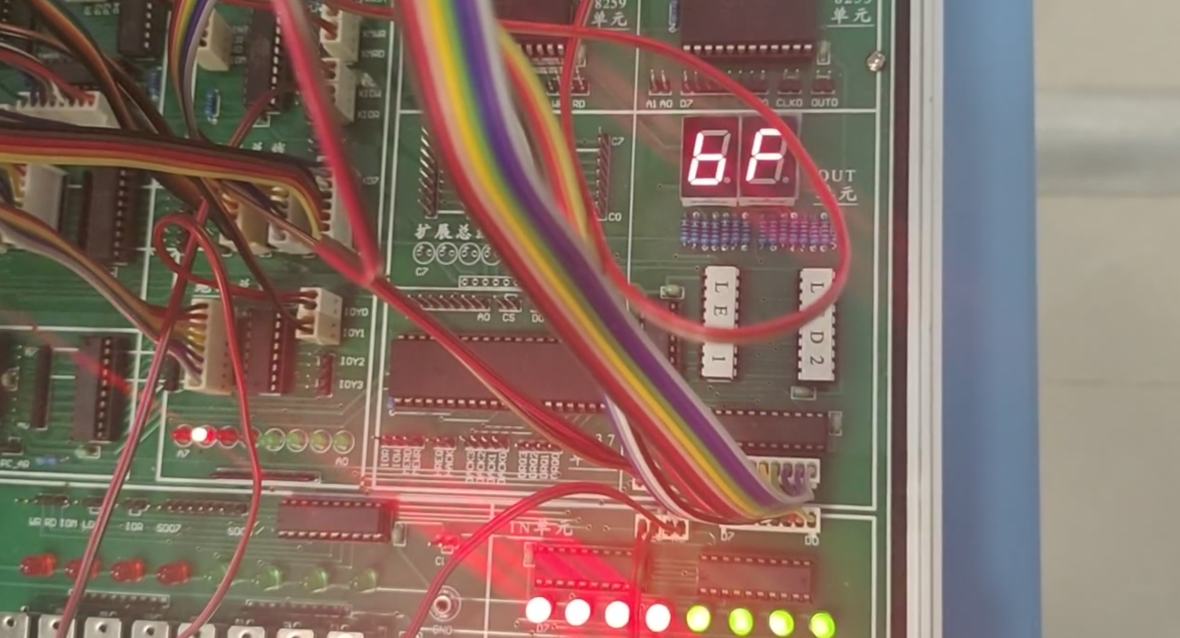


图8-4 输出（1）

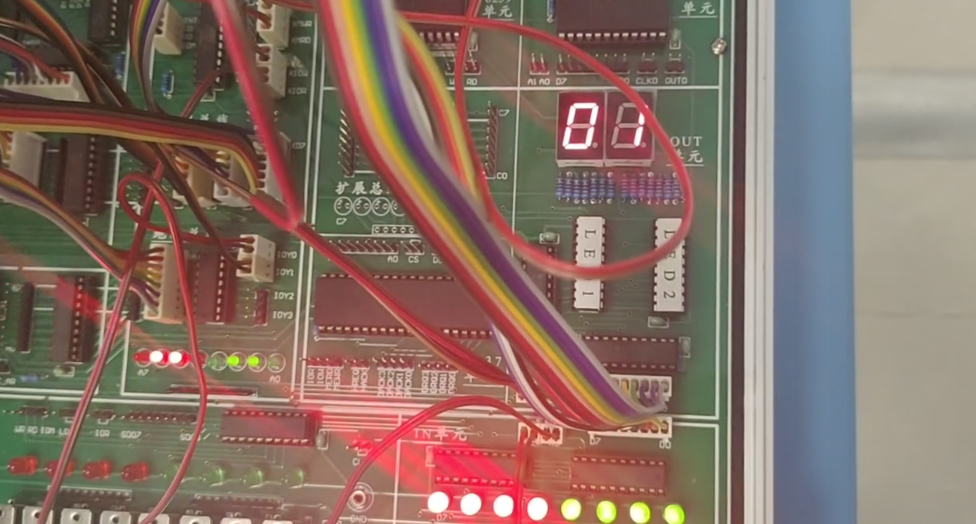


图8-5 输出（2）

# 9. 项目总结

通过本项目的设计与实现，我们成功地实现了一个复杂模型机，可以执行带符号位的二进制乘法操作。该模型机在功能上达到了预期要求，并具备一定的计算精度和运行效率。总体而言，项目取得了较好的成果。

# 10. 参考文献

[1]王爱英.计算机组成与结构[M].北京:清华大学出版社，2013

[2]丁红胜.CPU与简单模型机设计实验中的关键问题分析[J].计算机教育.2013（15）：13-17