长安大学

课程设计报告

课程名称：计算机组成原理

设计题目：复杂模型机设计与实现

专　　业：人工智能

姓　　名：郝志阳

学　　号：2022905226　32

同 组 人：程杪东、李晨阳、尹任博

指导教师：兰勇

二零 二四 年 十二 月

目录

1、课程设计任务书

1.1 设计任务

复杂模型机设计与实现

1.2 性能指标和设计要求

本设计实现了一个 8 位复杂指令系统模型机，具有以下主要特点：

1. 基本硬件规格
    - CPU 数据字长：8 位，采用定点补码表示
    - 指令字长：8 位的整数倍
    - 微指令字长：24 位
2. 指令系统设计
    - 基础指令集：包含输入(IN)、加法(ADD)、输出(OUT)、无条件转移(JMP)和停机(HLT)等基本指令
    - 扩展指令集：在基础上扩展至 16 条指令，涵盖算术逻辑指令(ADD/SUB/AND/OR/SHR)、访存指令(LAD/STA)、控制转移指令(JMP/BZC)、I/O 指令(IN/OUT)等多种类型
3. 寻址方式支持
    - 实现直接寻址、间接寻址、变址寻址和相对寻址等多种寻址方式
    - 采用统一编址的存储系统，支持灵活的数据访问
4. 功能验证要求
    - 设计完整的测试验证程序
    - 通过实际应用验证模型机的可行性与可靠性
    - 确保各类指令和寻址方式的正确实现

2、本设计的模型机体系结构及功能

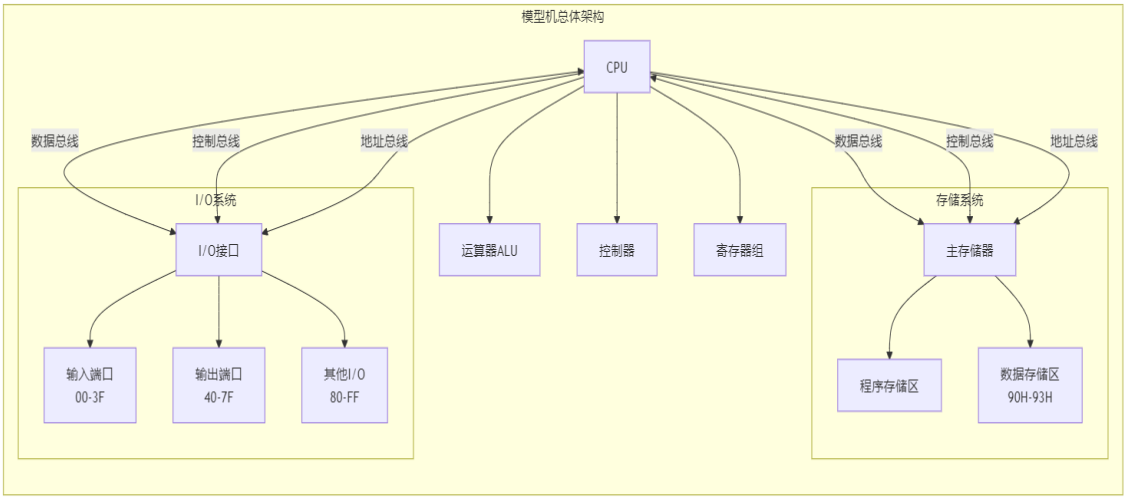

图 1：模型机总体结构框图

2.1 总体架构

本设计实现了一个基于冯·诺依曼结构的 8 位模型机，采用统一编址的存储系统，主要由以下部分组成：

表 1：总体架构功能表

功能部件	特性	说明
运算器(ALU)	数据处理能力	8 位算术逻辑运算，支持定点补码运算
	运算功能	基本运算(ADD/SUB)、逻辑运算(AND/OR)、逻辑右移(SHR)、自增自减(INC/DEC)
	状态标志	进位/借位标志(FC)、零标志(FZ)，支持条件判断
控制器	微程序控制	24 位微指令字长，水平型微指令格式
	分支控制	P<1>(指令分支)、P<2>(寻址分支)、P<3>(条件分支)
	控制功能	指令译码执行、时序控制、数据通路控制
存储系统	主存储器	统一编址，支持字节寻址
	专用存储	90H(输入 N)、91H(下界)、92H(上界)、93H(mid)
	地址空间	程序区、数据区、I/O 映射区
I/O 系统	地址划分	00-3F(输入)、40-7F(输出)、80-FF(扩展)
	接口功能	数据输入输出、设备选择、数据缓冲
	控制机制	地址译码、数据传送、状态检测

模型机数据通路图如下:

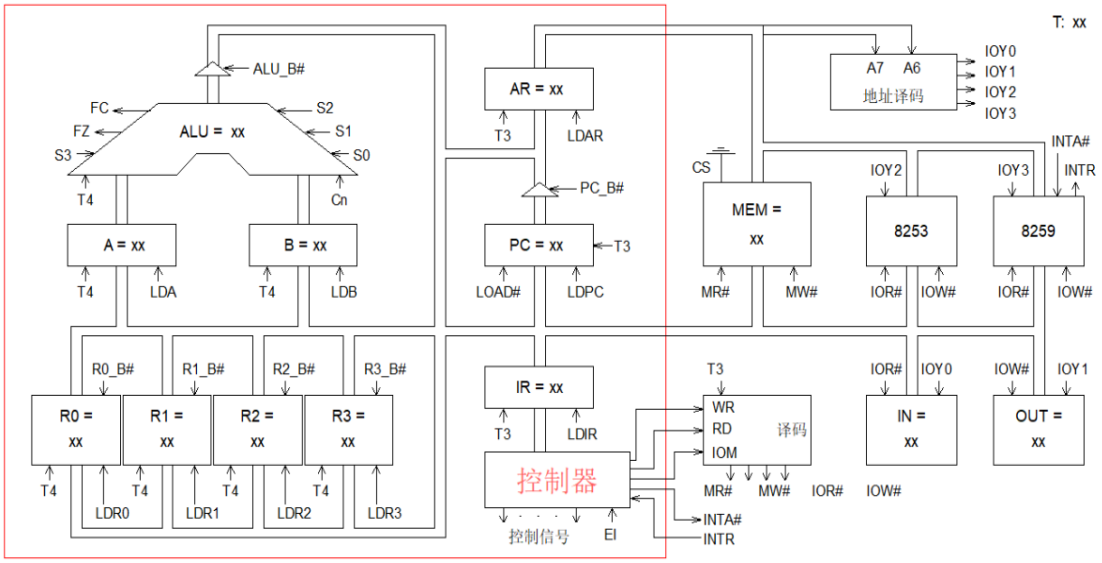

图 2：数据通路图

2.2 指令系统特点

指令字长设计	说明
单字节	运算和简单控制指令
双字节	访存和 I/O 指令
操作码	4 位，支持 16 条指令
寄存器编码	2 位，支持 4 个通用寄存器

寻址方式	说明
直接寻址	操作数地址直接给出
间接寻址	通过地址的地址访问
变址寻址	使用 R2 作变址寄存器
相对寻址	以 PC 为基准的转移

指令类型	包含指令
数据传送	MOV、LAD、STA
运算指令	ADD/SUB、AND/OR、SHR
控制指令	JMP、BZC
I/O 指令	IN、OUT

本系统在保留了原先的指令系统的基础上，新增了自减运算指令 DEC（RD-1->RD）和逻辑右移一位 SHR 指令（RS 逻辑右移一位->RD），并修改了原先的循环右移 RR 指令（RS 右环移->RD）为逻辑右移一位 SHR 指令（RS 逻辑右移一位->RD）。

3、模型机硬件设计

3.1 数据通路设计

3.1.1 主要功能部件

1. 寄存器组
    - 通用寄存器：
        - 4 个 8 位通用寄存器(R0-R3)
        - R0：主要用于存储运算结果和 I/O 操作
        - R1：常用于临时存储和比较操作
        - R2：兼作变址寄存器(RI)
        - R3：用于循环计数和临时存储

- 专用寄存器：
  - 程序计数器(PC)：存储下一条指令地址
  - 指令寄存器(IR)：存储当前执行的指令
  - 地址寄存器(AR)：存储访存地址
  - 变址寄存器(RI)：实现变址寻址功能

2. 算术逻辑单元(ALU)
   - 基本运算功能：
     - 加法运算(ADD)：实现 8 位二进制加法
     - 减法运算(SUB)：实现 8 位二进制减法
     - 逻辑与运算(AND)：按位与运算
     - 逻辑或运算(OR)：按位或运算
   - 特殊运算功能：
     - 逻辑右移(SHR)：数据右移一位，左侧补 0
     - 自增运算(INC)：操作数加 1
     - 自减运算(DEC)：操作数减 1
3. 存储器系统
   - 主存储器
   - 专用存储单元：
     - 90H：存储输入数 N，用于保存原始输入
     - 91H：存储下界值，用于二分查找
     - 92H：存储上界值，二分查找范围控制
     - 93H：存储中间值 mid，保存临时计算结果
4. I/O 接口系统
   - 输入接口：
     - 地址空间：00-3F
   - 输出接口：
     - 地址空间：40-7F

3.1.2 硬件接线

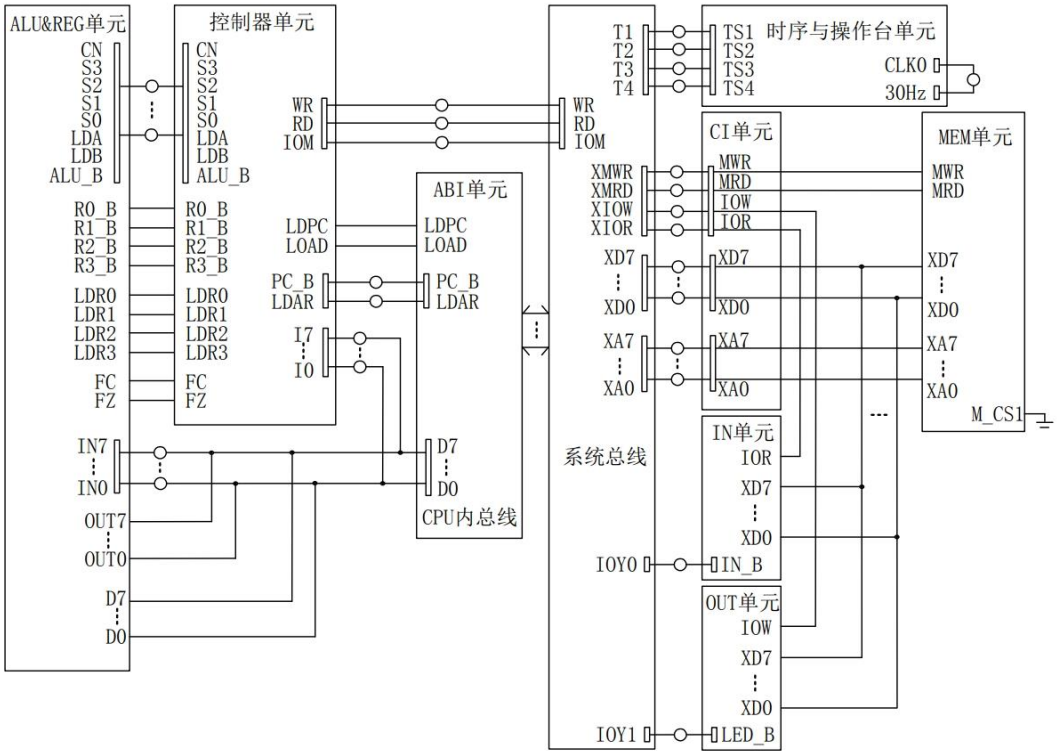

图 3：实验接线图

3.2 控制器设计

3.2.1微指令字段功能

微指令字长为 24 位，采用水平型微指令格式，各字段功能如下：

表 2：微指令格式位段含义

位段	功能描述	具体说明
23	M(微指令类型)	0：运算类型，1：控制类型
22	CN(进位控制)	0：无进位，1：进位
21	WR(写控制)	0：读操作，1：写操作
20	RD(读控制)	0：非读操作，1：读操作
19	IOM(存储器/IO 选择)	0：访问存储器，1：访问 I/O
18~15	S3~S0(ALU 操作选择)	0000：加法，0001：减法，等
14~12	A 字段(数据写入选择)	000：NOP，001：LDA，等
11~9	B 字段(数据读出选择)	000：NOP，001：ALU_B，等

位段	功能描述	具体说明
8~6	C 字段(分支控制)	000：NOP，001：P<1>，等
5~0	UA5~UA0(下地址)	下一条微指令地址

3.2.2 相关译码电路

对于每条微指令，需要微指令的 P 测试字段和机器指令操作码的各个位共同作用产生译码，译码电路如下，具体原理将在指令系统设计中详细探讨。

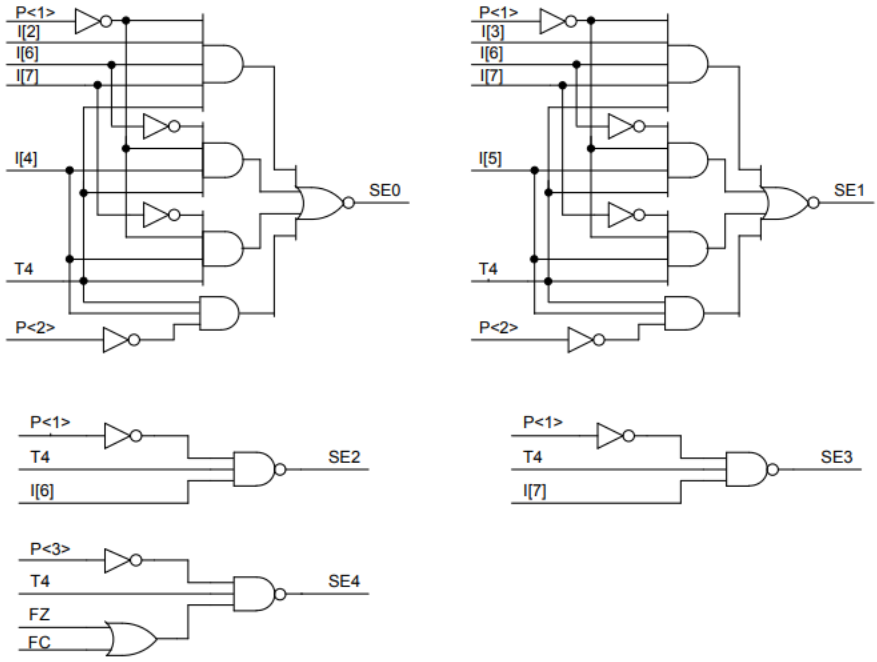

图 4：指令译码原理图

这种详细的硬件设计方案不仅完整地描述了模型机的各个硬件组成部分，还通过合理的控制方式和时序安排，确保了整个系统的可靠运行。系统的每个部分都经过设计，既保证了各个功能模块的独立性，又实现了它们之间的有效协同工作。

系统的 I/O 地址译码原理见下图（在地址总线单元）。

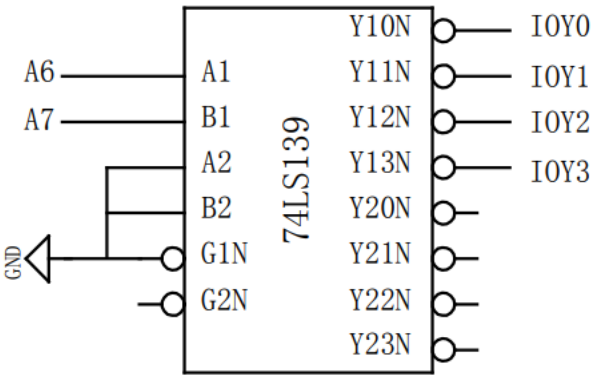

图 5：I/O 地址译码原理图

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，在控制器单元的 REG_DEC 中实现，译码电路如下图。

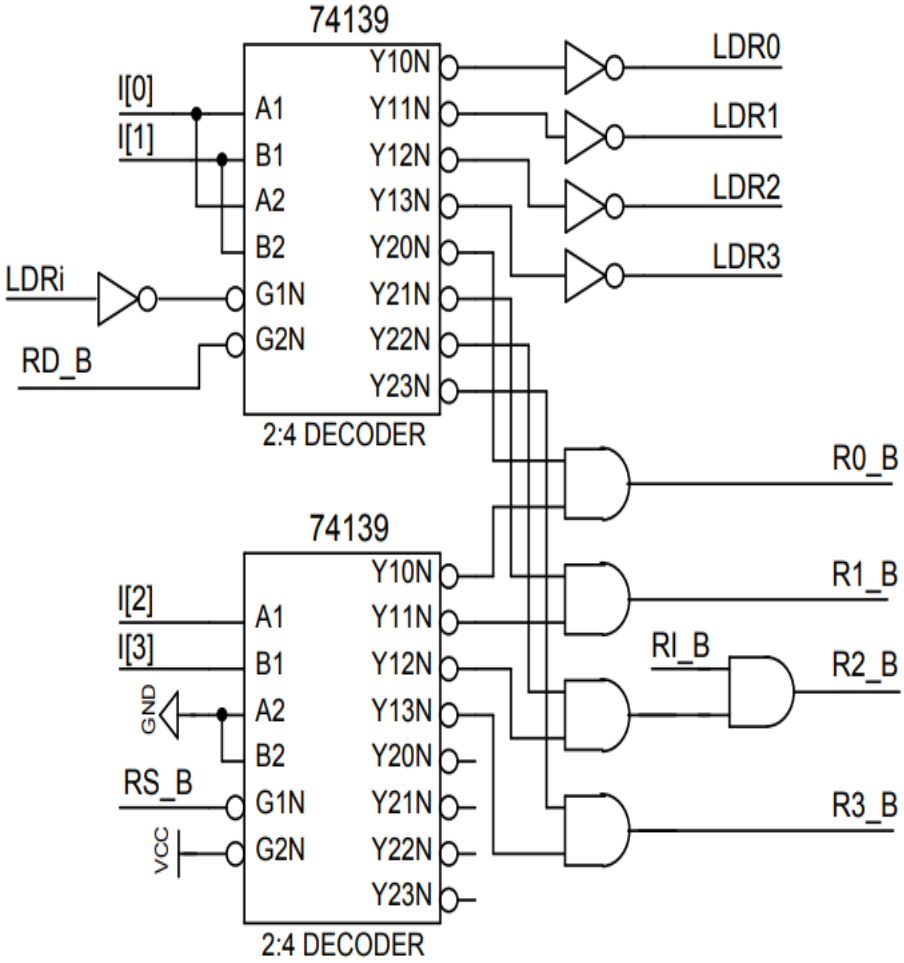

图 6：寄存器译码原理图

4、模型机机器指令系统设计

4.1 指令系统及分析

4.1.1指令系统

本模型机共有 16 条机器指令，表 3 列出了各条指令的格式、汇编符号、指令功能。

表 3 机器指令描述

汇编符号	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
AND RD,RS	0001	RS	RD		RD∧RS->RD
OR RD,RS	1001	RS	RD		$\overline{RD}$->RD
SHR RD,RS	1010 A	RS	RD		RS 逻辑右移一位->RD
DEC RD	1011 B	**	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
HALT	0101	**	**		停机

4.1.2 指令系统分析

本模型机在原先指令系统的基础上，对指令系统进行了一定的修改，并新增了一条指令。

1. 在保留字段**新增自减运算指令 DEC（RD-1->RD）**具体指令如下表所示。

表 4：DEC

汇编符号	指令格式			指令功能
DEC RD	1011	**	RD	RD-1->RD

2. 修改原先循环右移 RR 指令（RS 右环移->RD）
   改为**逻辑右移一位 SHR 指令（RS 逻辑右移一位->RD）**具体指令如下表所示。

表 5：SHR

汇编符号	指令格式			指令功能
SHR RD,RS	1010 A	RS	RD	RS 逻辑右移一位->RD

4.1.2.1 指令设计

模型机设计三大类指令共十六条，其中包括运算类指令、控制转移类指令，数据传送类指令。

1. **运算类指令**包含三种运算：算术运算、逻辑运算和移位运算。设计有的运算指令分别为：ADD、SUB、AND、OR、SHR、DEC、INC 共 7 条，所有运算类指令都为单字节。
2. **控制转移类指令** JMP、BZC、HLT 共 3 条，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。
3. **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，寻址方式采用直接寻址。

4.1.2.2 指令格式

1. 单字节指令格式

ADD、SUB、AND、OR、SHR、DEC、INC、HLT 和 MOV。

表 6：单字节指令格式

I7	I6	I5	I4	I3	I2	I1	I0
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。

用两位二进制对于 RS 和 RD 寄存器的选定，本模型机按照一定规则选定。

表 7：RS 或 RD 选定的寄存器

RS 或 RD	选定的寄存器
00	R0

01	R1
10	R2
11	R3

2. 双字节指令格式

IN、OUT、JMP、BZC、LDI、LAD、 STA

- IN 和 OUT 的指令

表 8：IN 和 OUT 指令格式表

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

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，占用一个字节。I/O 地址译码原理图如图 5 所示（在地址总线）。

由于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，如表 9 所示：

表 9：I/O 地址空间分配表

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

实验中遇到的问题：

在指令集的最后，在 OUT 单元输出结果时，原先我选取了随意的一个地址单元进行输出，发现在连续执行并停机后，未在 OUT 单元输出结果，结果仍然保存在寄存器。因此对地址单元进行检查与思考，在实验模型机的数据通路框图中发现只有选定 IOY1 时才能正确输出，因此我将 OUT 指令的第二字节改为 40-7F 的任意单元，结果正确显示在 OUT 单元。

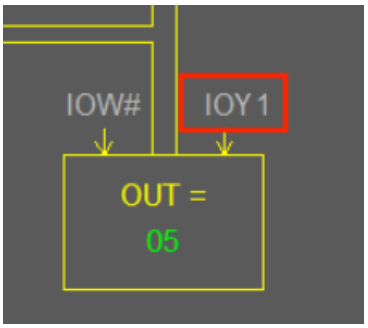

图 7：OUT 单元地址

- LDI 指令

LDI 的指令格式第一字节与 IN 和 OUT 指令相同，第二字节为立即数。

表 10：LDI 指令格式表

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

系统设计五种数据寻址方式，即立即寻址、直接寻址、间接寻址、变址寻址和相对寻址，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。

- LAD、STA、JMP 和 BZC 指令

表 11：LAD 等指令格式表

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。

表 12：寻址方式表

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

4.2 机器程序及分析

4.2.1机器程序流程图

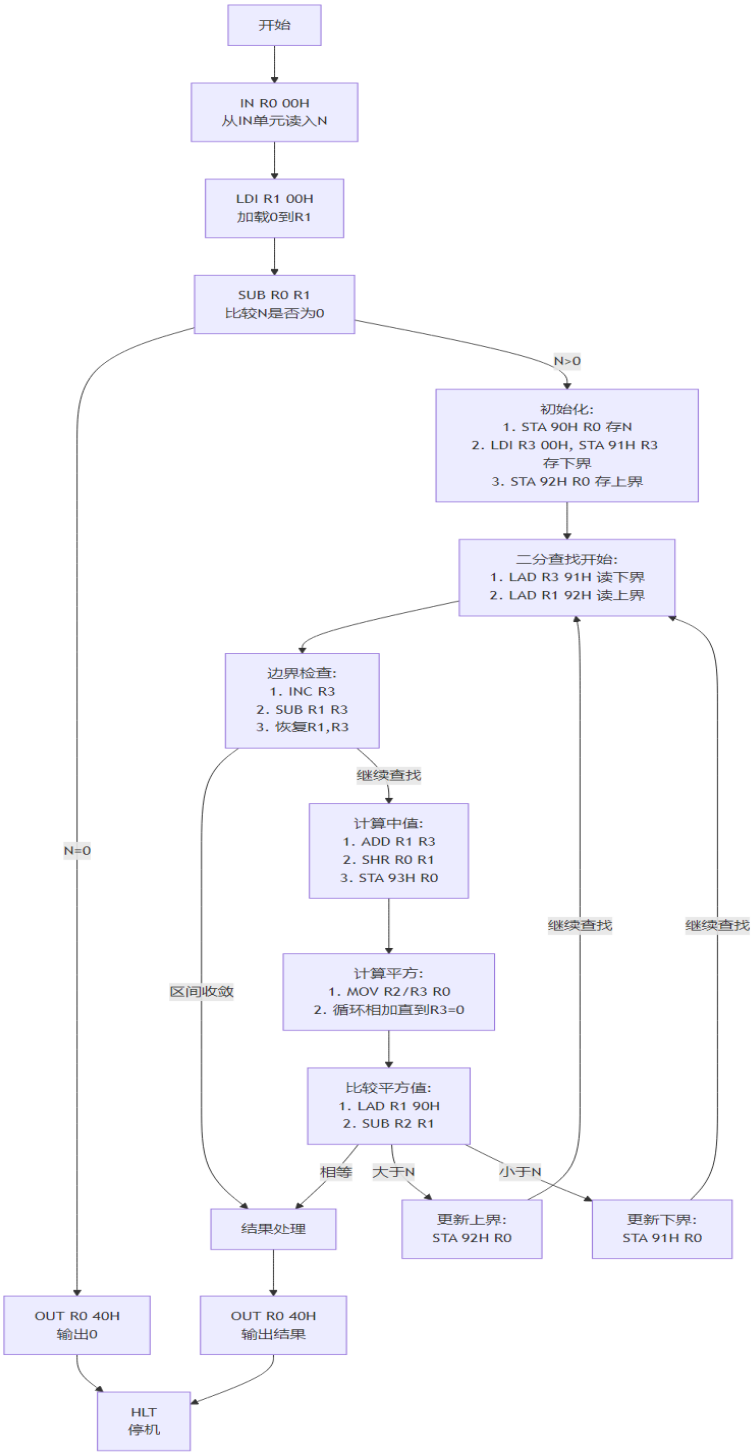

图 8：开平方根机器程序流程

4.2.2程序流程及说明

表 13：程序流程说明

地址	内容	助记符	说明
00	20	IN R0 00H	从 IN 单元读入数 N 到 R0
01	00		
02	61	LDI R1 00H	加载 0 到 R1 用于比较
03	00		
04	84	SUB R0 R1	R0 = R0 - R1，判断是否为 0
05	F0	BZC OUT	如果为 0 则跳转到 OUT
06	36		
07	D0	STA 90H R0	将 N 存入 90H 单元
08	90		
09	63	LDI 00H R3	R3=0 (下界)
0A	00		
0B	D3	STA 91H R3	存入 91H (下界)
0C	91		
0D	D0	STA 92H R0	将 N 存入 92H (上界)
0E	92		
0F	C3	LAD R3 91H	读取下界到 R3
10	91		
11	C1	LAD R1 92H	读取上界到 R1
12	92		
13	73	INC R3	R3 = R3 + 1
14	8D	SUB R1 R3	R1 = R1 - R3
15	F0	BZC Result	如果 R1=0 或有借位则完成
16	36		
17	0D	ADD R1 R3	恢复 R1 的值
18	B3	DEC R3	R3 = R3 - 1
19	0D	ADD R1 R3	R1 = R1 + R3
1A	A4	SHR R0 R1	R1 逻辑右移一位到 R0 (mid)
1B	D0	STA 93H R0	存储 mid 到 93H
1C	93		

地址	内容	助记符	说明
1D	42	MOV R2 R0	R2 = R0 (用于计算平方)
1E	43	MOV R3 R0	R3 = R0
1F	B3	LOOP: DEC R3	R3 = R3 - 1
20	F0	BZC Result	如果 R3=0 则完成乘法
21	25		
22	02	ADD R2 R0	R2 = R2 + R0
23	E0	JMP LOOP	继续循环
24	1F		
25	C1	LAD R1 90H	读取原始数 N 到 R1
26	90		
27	86	SUB R2 R1	R2 = R2 - R1
28	F0	BZC Result	如果 R2=0 或有借位则跳转
29	2E		
2A	D0	STA 92H R0	更新上界为 mid
2B	92		
2C	E0	JMP 0F	继续二分查找
2D	0F		
2E	06	ADD R2 R1	恢复 R2 的值
2F	89	SUB R1 R2	R1 = R1 - R2
30	F0	BZC OUT	如果 R1=0 则跳转到 OUT
31	36		
32	D0	STA 91H R0	更新下界为 mid
33	91		
34	E0	JMP 0F	继续二分查找
35	0F		
36	30	OUT R0 40H	输出结果
37	40		
38	50	END: HLT	停机

主要功能包括：

1. **输入处理：**
    - 从 IN 单元读入数 N
    - 判断输入是否为 0，如果为 0 直接输出 0 并停机

- 将输入值 N 保存到 90H 存储单元

2. **初始化：**
   - 设置下界为 0（存入 91H）
   - 设置上界为 N（存入 92H）
3. **二分查找过程：**
   - 读取当前上下界
   - 计算中间值 mid = (上界 + 下界) >> 1
   - 将 mid 存入 93H
   - 计算 mid 的平方（通过循环加法实现）
   - 比较 mid^2 与 N 的大小
   - 根据比较结果更新上界或下界
   - 继续二分查找直到找到合适的值
4. **结果输出：**
   - 将计算得到的平方根输出到 40H 端口
   - 程序停机

4.3 算法核心点说明

4.3.1二分查找算法实现

- 本模型机采用二分查找法实现整数平方根的计算
- 时间复杂度为 O(log n),相比线性查找大大提高了效率
- 具体实现:
  - 初始化下界为 0(存储在 91H),上界为输入值 N(存储在 92H)
  - 每次取中间值 mid = (上界 + 下界) >> 1
  - 计算 mid 的平方与 N 比较,据此更新上下界，直到找到合适平方根值

4.3.2平方根判断的精确实现

- 采用双向减法比较策略确保结果准确性:
  1. 首先计算(R1)-(R2)->R1
  2. 当 P<3>判断结果为 0 或有借位时,需要进一步判断
  3. 将 R1、R2 恢复原值，再计算(R2)-(R1)->R2
  4. 根据 P<3>判断结果确定最终关系:
     - 若为 0 或有借位,则 R1=R2，否则 R1<R2

4.3.3边界条件处理

- 特殊输入处理:
  - 输入为 0 时直接返回 0
  - 输入为 1 时直接返回 1

5、模型机控制器微程序设计

5.1 机器指令周期分析

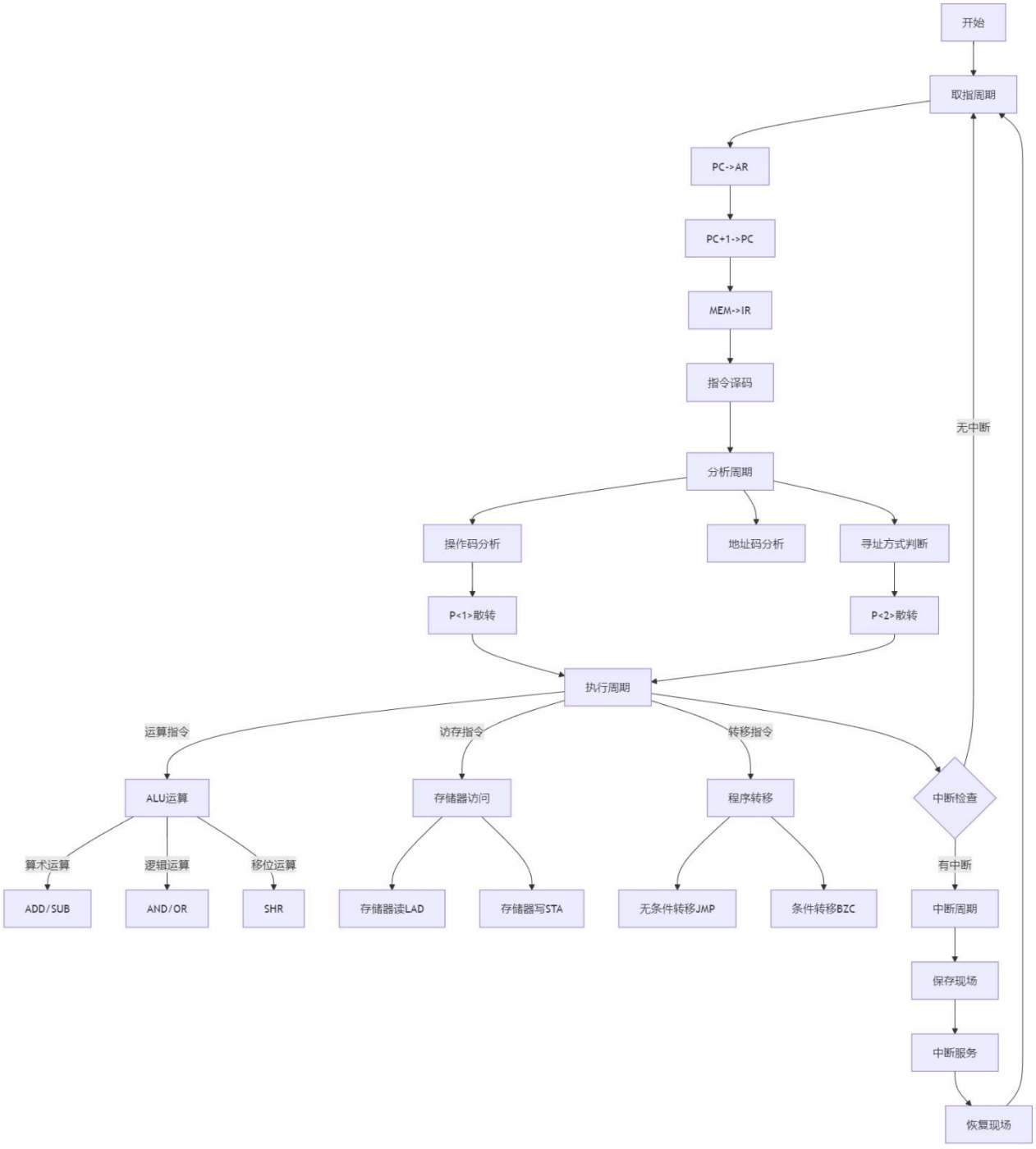

图 9：机器指令周期分析流程

每条机器指令的执行都需要经过以下基本周期,每个周期都有其特定的功能和微操作序列:

1. 取指周期(Fetch Cycle)
    - 基本功能:
        - 从程序计数器(PC)获取当前指令地址
        - 从主存储器读取指令到指令寄存器(IR)
        - PC 自动加 1,为执行下一条指令做准备
    - 微操作序列:

```
PC->AR            // 将 PC 内容送到地址寄存器
PC+1->PC          // PC 自增 1
MEM->IR           // 从存储器读取指令到 IR
IR->译码器,P<1>   // 指令译码并进行 P<1>判断
```

    - 时序控制:
        - T0: PC->AR,启动存储器读操作
        - T1: PC+1->PC,等待存储器读完成
        - T2: MEM->IR,开始指令译码
        - T3: 完成译码,形成入口地址
2. 分析周期(Analysis Cycle)
    - 指令分析:
        - 操作码分析:确定指令类型和功能
        - 地址码分析:确定操作数地址
        - 寻址方式判断:确定有效地址的计算方法
    - 地址计算:
        - 直接寻址: EA = D
        - 间接寻址: EA = (D)
        - 变址寻址: EA = (RI) + D
        - 相对寻址: EA = (PC) + D
    - 控制流程:
        - P<1>散转:根据操作码形成入口地址
        - P<2>散转:根据寻址方式选择地址计算方法
        - 准备执行阶段所需的操作数
3. 执行周期(Execute Cycle)
    - 运算类指令执行:
        - 算术运算(ADD,SUB):

```
RS->B          // 源操作数送 B
RD->A          // 目的操作数送 A
ALU 运算->RD    // 结果存入目的寄存器
```

- 逻辑运算(AND,OR):

```
RS->B          // 源操作数送 B
RD->A          // 目的操作数送 A
逻辑运算->RD   // 结果存入目的寄存器
```

- 移位运算(SHR):

```
RS->A          // 操作数送 A
A 右移->RD      // 逻辑右移结果送 RD
```

- 访存类指令执行:
  - 存储器读(LAD):

```
EA->AR         // 有效地址送 AR
MEM->RD        // 从存储器读数据到寄存器
```

  - 存储器写(STA):

```
EA->AR         // 有效地址送 AR
RS->MEM        // 寄存器内容写入存储器
```

- 转移类指令执行:
  - 无条件转移(JMP):

```
EA->PC         // 直接修改 PC 值
```

  - 条件转移(BZC):

```
检查条件标志
若满足:EA->PC  // 条件成立时修改 PC
若不满足:PC 不变 // 继续顺序执行
```

5.2 微程序流程图分析

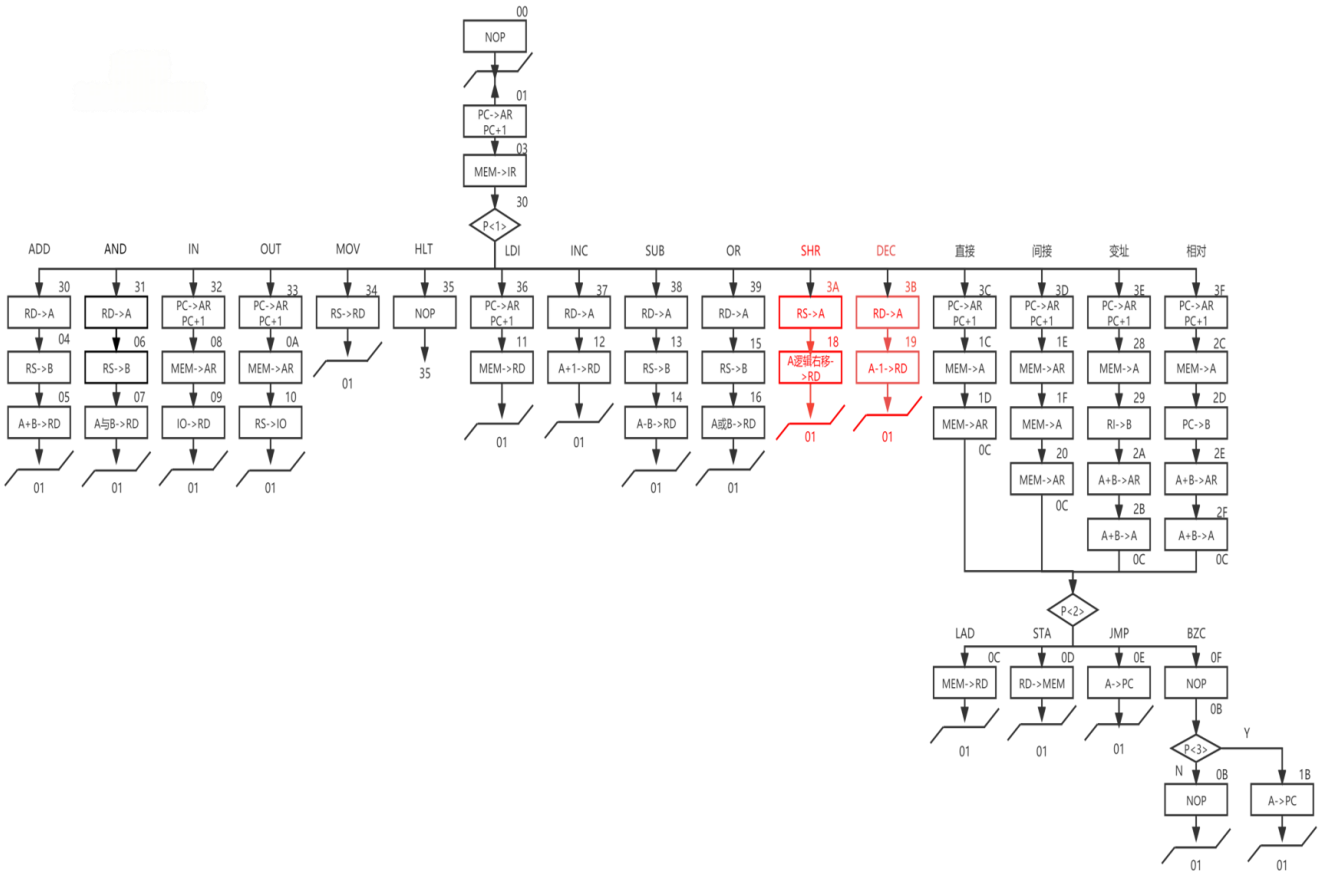

图 10：微程序流程图

5.2.1基本指令周期

- **取指周期：**

  开始 -> PC->AR -> PC+1 -> MEM->IR -> P<1>散转

  - 从 PC 获取指令地址送到 AR
  - PC 自动加 1 为下一条指令做准备
  - 从存储器读取指令到 IR
  - 通过 P<1>散转进入相应的指令执行流程

5.2.2指令执行流程

5.2.2.1 运算类指令

- **ADD/SUB/AND/OR 指令：**

  RS->B -> RD->A -> ALU 运算 -> 结果->RD -> 取指

  - 将源操作数和目的操作数送入 ALU

- 执行相应的算术或逻辑运算
- 结果存回目的寄存器

- **SHR 指令**（修改后）：

  RS->A -> A 逻辑右移一位->RD -> 取指

  - 直接将源操作数右移一位
  - 结果存入目的寄存器

- **DEC 指令**（新增）：

  RD->A -> A-1->RD -> 取指

  - 将目的寄存器内容减 1
  - 结果存回目的寄存器

5.2.2.2 数据传送指令

- **MOV/LDI 指令**：

  源操作数->目的寄存器 -> 取指

- **LAD/STA 指令**：

  地址计算 -> 存储器访问 -> 数据传送 -> 取指

  - 支持多种寻址方式（直接、间接、变址、相对）

5.2.2.3 转移指令

- **JMP 指令**：

  地址计算 -> E->PC -> 取指

  - 无条件转移到目标地址

- **BZC 指令**：

  条件判断(P<3>) -> 满足条件时 E->PC -> 取指

  - 根据零标志或进位标志决定是否转移

5.2.3关键控制点

- **P<1>散转**：
  - 根据操作码确定指令类型
  - 生成相应的入口地址
- **P<2>散转**：
  - 处理不同的寻址方式
  - 确定有效地址的计算方法

- P<3>散转：
  - 用于条件转移指令
  - 根据标志位决定转移方向

5.2.4特殊处理流程

- I/O 操作：

  IN/OUT 指令 -> 设备选择 -> 数据传送 -> 取指

  - 通过专门的 I/O 端口进行数据交换

5.3 微指令格式设计

本模型机微指令字长共 24 位，微指令格式如下表所示。

表 14：微指令格式

23	22	21	20	19	18~15	14~12	11~9	8~6	5~0
M	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

微指令根据 24 位对应的值进行翻译和执行，其中 A 字段代表数据打入寄存器或暂存器或缓存，B 字段代表数据从上述存储器中打出到数据总线，_B 代表 BUS，即数据总线。C 字段代表微指令进行一些分支以及判断。

5.4 微指令编码设计

下表为模型机指令系统对应微程序二进制代码。

表 15：微指令代码

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	011	000	000101
05	04 B2 01	00000	1001	011	001	000	000001
06	00 24 07	00000	0000	010	011	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	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	011	000	010100
14	05 B2 01	00000	1011	011	001	000	000001
15	00 24 16	00000	0000	010	011	000	010110
16	01 B2 01	00000	0011	011	001	000	000001
17	00 24 18	00000	0000	010	011	000	011000
18	03 32 01	00000	0110	011	001	000	000001
19	06 32 01	00000	1100	011	001	000	000001

地址	十六进制表示	高五位	S3-S0	A 字段	B 字段	C 字段	UA5-UA0
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 14 18	00000	0000	001	010	000	011000
3B	00 16 19	00000	0000	001	011	000	011001
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

其中修改的微程序指令已用红色字体标出，如下表所示。

表 16：修改的微指令

地址	十六进制表示	含义	下地址	修改部分
18	03 32 01	A 逻辑右移一位->RD	01	S3~S0 改为 0110（逻辑右移），CN 位为 0
19	06 32 01	A-1->RD	01	S3~S0 改为 1100（自减）
3A	00 14 18	RS->A	18	B 字段改为 010（RS_B）
3B	00 16 19	RD->A	19	A 字段改为 001（LDA），B 字段改为 011（RD_B）

修改 SHR 指令时，实验册中的逻辑循环右移 RR 实现了 RD->A,RS->B,A 右环移 B 位送 RD，而 SHR 指令中，由于 S3~S0 为 0110（逻辑右移）已经规定移动一位，因此不用再打入 1 至 B 暂存器，并且可以实现 RS 右移一位->RD。

5.5 微指令地址及控存存储设计

5.5.1微指令地址分配方案

1. 基本地址分配原则
    - 00H-0FH: 基本指令周期微程序
    - 10H-1FH: 运算类指令微程序
    - 20H-2FH: 寻址方式处理微程序
    - 30H-3FH: 转移类指令微程序
2. 具体地址分配
    - 取指周期: 00H-03H
        - 00H: NOP(空操作)
        - 01H: PC->AR,PC+1
        - 03H: MEM->IR,P<1>
    - 运算类指令: 10H-1FH
        - 18H: SHR 指令(A 右移->RD)
        - 19H: DEC 指令(A-1->RD)
        - 其他算术逻辑运算指令
    - 寻址方式处理: 20H-2FH
        - 28H-2BH: 变址寻址
        - 2CH-2FH: 相对寻址
    - 转移类指令: 30H-3FH
        - 30H-35H: 条件转移处理
        - 36H-3FH: 无条件转移处理

5.5.2控存组织结构

1. 控存容量设计
    - 地址空间：6 位(64 个地址)
    - 字长：24 位
    - 总容量：64×24 位
2. 控存字段划分

表 17：控存字段划分表

字段	位数	功能描述
M	1	微指令类型
CN	1	进位控制
WR	1	写控制
RD	1	读控制
IOM	1	存储器/IO 选择
S3~S0	4	ALU 操作选择
A 字段	3	数据写入选择
B 字段	3	数据读出选择
C 字段	3	分支控制
UA5~UA0	6	下地址

3. 控存访问机制
    - 读取方式：同步读取
    - 时序控制：由时钟信号控制
    - 地址形成：
        - 常规顺序执行
        - 分支转移地址生成

- 条件判断地址选择

5.5.3地址生成逻辑

1. 顺序地址生成
    - 直接使用 UA5~UA0 字段
    - 用于顺序执行的微指令
2. 分支地址生成
    - P<1>散转：根据操作码生成
    - P<2>散转：根据寻址方式生成
    - P<3>散转：根据条件标志生成

通过合理的微指令地址分配和控存组织设计，不仅保证了微程序的正确执行，也为系统的扩展和维护提供了便利。同时，优化的设计方案提高了系统的执行效率和可靠性。

5.6 微指令执行流程

每一次取指令后，机器指令经过译码，进入 P<1>判断，进而形成入口地址，根据入口地址进入相应的微指令执行流程。 每一条微指令执行结束后，根据后续微地址及微地址形成规则跳转到下一条微指令，最后回到取指令的微指令。整个流程循环完成对某个功能的实现。

- P<1>散转下址生成规则

表 18：P<1>散转下址生成规则

类型	SE5	SE4	SE3	SE2	SE1	SE0
I7I6≠11	MA5	MA4	I7	I6	I5	I4
I7I6=11	MA5	MA4	1	1	I3	I2

MA5~MA0 为当前微指令中的地址字段内容;SE5~SE0 为下一条实际执行的微指令的地址。

分析：

I7I6≠11，即操作码前两位不为 11 时，下地址的生成方式是用指令的操作码（即前四位）替换 MA5~MA0 的最后四位。

I7I6=11，即操作码前两位为 11 时，下地址的生成方式是保持 MA5~MA0 的前两位不变，后两位置为 1，最后两位置为寻址方式 M。

在本模型机的指令系统中，只有下面四条指令 I7I6=11，用 M 表示寻址方式，在指令译码时，采取第二种散转下地址生成规则。即后四位替换成 11M。

下表为 I7 I6=11 的指令。

表 19：I7I6=11 的指令

汇编符号	指令格式				指令功能
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

以下图的变址寻址方式为例，变址寻址方式 M 为 10，则 MA5~MA0 原先为 30（110000）将后四位替换为 11+M，即 3E（111110）跳转到变址寻址的微指令流程执行。

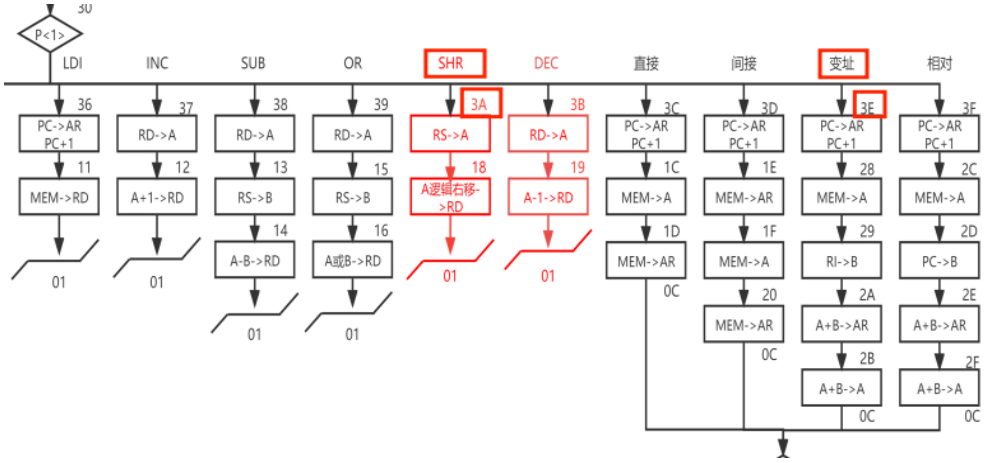

图 11：P<1>散转下址生成规则

在本模型机的指令系统中,除上述四条指令之外，其他指令的 I7I6≠11，则采取第一种散转下地址生成规则，即后四位替换成操作码。

以上图 SHR 指令为例，SHR 指令格式如下表所示

表 20：SHR 指令格式

汇编符号	指令格式			指令功能
SHR RD,RS	1010 A	RS	RD	RS 逻辑右移一位->RD

经 P<1>散转下址生成规则判别后，将后四位替换成操作码（1010）即 3A (111010)跳转到 SHR 的微指令流程执行。

- P<2>散转下址生成规则

表 21：P<2>散转下址生成规则

SE5	SE4	SE3	SE2	SE1	SE0

MA5	MA4	MA3	MA2	I5	I4

在指令框图中，P<2>散转下址生成规则只设计到四条指令，即在不同的寻址方式后经 P<2>判定需要继续执行什么操作。

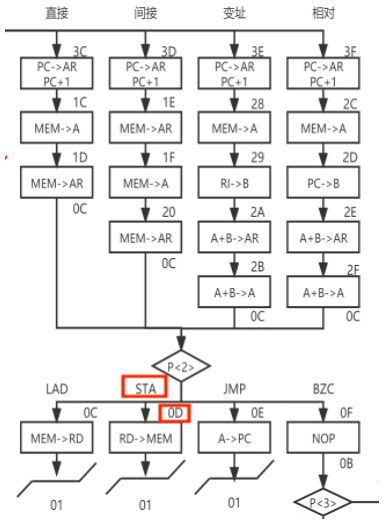

图 12：P<2>散转下址生成规则

当前 MA5~MA0 为 0C（001100）P<2>散转下址生成规则表明下一条微指令应该将 MA5~MA0 的最后两位替换成操作码的后两位（I5I4）即可正确跳转到下一条微指令。以 STA 指令为例，当前 MA5~MA0 为 0C（001100），将后两位替换成 STA 指令操作码的后两位（01）后变为 0D（001101），即跳转到 STA 微指令流程执行。

- P<3>散转下址生成规则

表 22：P<3>散转下址生成规则

SE5	SE4	SE3	SE2	SE1	SE0
MA5	FC（OR）FZ	MA3	MA2	MA1	MA0

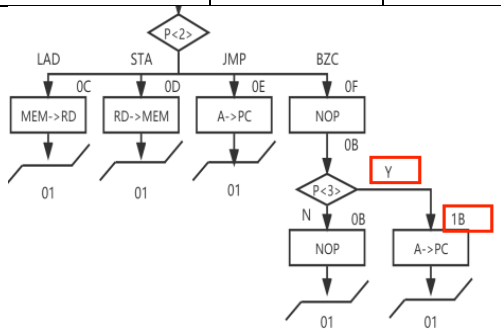

图 13：P<3>散转下址生成规则

BZC 与 JMP 指令同属于无条件转移指令，由 P<3>散转下址生成规则可以看出，若 FC（OR）FZ=1，即有进位（或借位）或结果为 0 时将 MA5~MA0 中的第二位改成 1，若没有上述情况则为 0。

以结果为 0 为例，当前 MA5~MA0 为 0B（001011）,结果为 0,FZ=1,第二位替换成 1，下地址变为 1B（011011），执行 A->PC，即跳转到 PC。

6、模型机功能测试

6.1 机器指令功能调试

6.1.1运算类指令测试程序

6.1.1.1 算术运算测试

```
; 测试 ADD、SUB、INC、DEC 指令
$P 00 61   ; LDI R1,03H    ; R1 = 3     (0110 0001)
$P 01 03
$P 02 62   ; LDI R2,02H    ; R2 = 2     (0110 0010)
$P 03 02
$P 04 09   ; ADD R1,R2     ; R1 = R1 + R2 = 5  (0000 1001)
$P 05 71   ; INC R1        ; R1 = R1 + 1 = 6   (0111 0001)
$P 06 89   ; SUB R1,R2     ; R1 = R1 - R2 = 4  (1000 1001)
$P 07 B1   ; DEC R1        ; R1 = R1 - 1 = 3   (1011 0001)
$P 08 34   ; OUT 40H,R1    ; 输出结果   (0011 0001)
$P 09 40
$P 0A 50   ; HALT          ; 停机       (0101 0000)
```

6.1.1.2 逻辑运算测试

```
; 测试 AND、OR、SHR 指令
$P 00 61   ; LDI R1,0FH    ; R1 = 1111B (0110 ** 01 0F)
$P 01 0F
$P 02 62   ; LDI R2,05H    ; R2 = 0101B (0110 ** 10 05)
$P 03 05
$P 04 19   ; AND R1,R2     ; R1 = R1 AND R2 = 0101B (0001 10 01)
$P 05 99   ; OR R1,R2      ; R1 = R1 OR R2 = 0101B  (1001 10 01)
$P 06 A7   ; SHR R1,R3      ; R3 = R1 >> 1 = 0010B   (1010 01 11)
$P 07 34   ; OUT 40H,R1    ; 输出结果   (0011 01 ** 40)
$P 08 40
$P 09 50   ; HALT          ; 停机       (0101 ** **)
```

6.1.2数据传送指令测试程序

```
; 测试 MOV、LAD、STA、IN、OUT 指令
$P 00 20   ; IN R0,00H     ; 从 00H 端口输入数据到 R0 (0010 ** 00 00)
$P 01 00
$P 02 41   ; MOV R1,R0     ; R0 -> R1   (0100 00 01)
$P 03 D1   ; STA 90H,R1    ; R1 -> [90H] (1101 00 01 90)
$P 04 90
$P 05 C2   ; LAD 90H,R2    ; [90H] -> R2 (1100 00 10 90)
$P 06 90
$P 07 38   ; OUT 40H,R2    ; 输出到 40H 端口 (0011 10 ** 40)
```

```
$P 08 40
$P 09 50   ; HALT          ; 停机       (0101 ** **)
```

6.1.3转移指令测试程序

```
; 测试 JMP 和 BZC 指令
$P 00 61   ; LDI R1,01H    ; R1 = 1     (0110 ** 01 01)
$P 01 01
$P 02 62   ; LDI R2,01H    ; R2 = 1     (0110 ** 10 01)
$P 03 01
$P 04 89   ; SUB R1,R2     ; R1 = R1 - R2 (1000 10 01)
$P 05 F0   ; BZC NEXT      ; 如果结果为 0 则跳转 (1111 00 ** 09)
$P 06 09
$P 07 E0   ; JMP END       ; 无条件跳转到结束 (1110 00 ** 0B)
$P 08 0B
$P 09 34   ; NEXT: OUT 40H,R1  ; 输出结果 (0011 01 ** 40)
$P 0A 40
$P 0B 50   ; END: HALT     ; 停机       (0101 ** **)
```

6.1.4测试结果分析

1. 运算类指令测试
    - ADD/SUB 指令正确执行加减运算
    - INC/DEC 指令正确执行自增自减
    - AND/OR 指令正确执行逻辑运算
    - SHR 指令正确执行逻辑右移
2. 数据传送指令测试
    - MOV 指令正确完成寄存器间数据传送
    - LAD/STA 指令正确完成内存访问
    - IN/OUT 指令正确完成 I/O 操作
3. 转移指令测试
    - JMP 指令正确执行无条件转移
    - BZC 指令正确根据条件进行转移

6.2 整机功能调试

6.2.1微程序写入

选择联机软件的“转储”→“装载”命令，将该格式的文件(*.txt)装载入 TD-CMA 实验系统。装载过程中，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。

模型机装载的微程序代码如下：

```
; //** 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 033201    ; A 右移->RD
$M 19 063201    ; A 减 1->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 001418    ; RS->A
$M 3B 001619    ; RD->A
$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 **//
```

6.2.2微程序校验

选择联机软件的“转储”→“刷新指令区”命令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。检查微程序控制器相应地址单元的数据是否和装载进去的十六进制数据相同，如果不同，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，先单击指令区的微存 TAB 按钮，然后单击需要修改的单元的数据，此时该单元变为编辑框，输入 6 位数据并按回车键，编辑框消失，写入的数据显示红色。

其中，我通过对修改的几条指令进行和核对，发现已经装载和更新完毕，下图以 18 这条微程序为例。

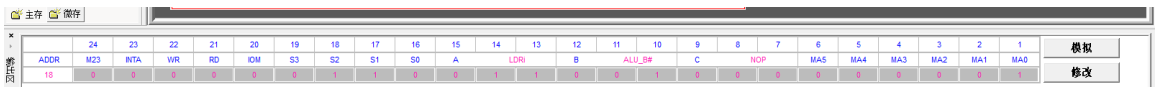

图 14：微程序校验

6.2.3机器程序写入

选择联机软件的“转储”→“装载”命令，将该格式的文件(*.txt)装载入 TD-CMA 实验系统。装载过程中，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。

模型机装载的代码如下：

```
; //************************************** //
; //                                      //
; //        复杂模型机实验指令文件     //
; //                                        //
; //                                        //
; //                                        //
; //************************************** //
; //****** Start Of Main Memory Data ****** //
; //************************************** //
; //        开平方根程序                    //
; //************************************** //
$P 00 20 ;IN R0 00H         ; 从 IN 单元读入数 N 到 R0
$P 01 00
$P 02 61 ;LDI R1 00H       ; 加载 0 到 R1 用于比较
```

```
$P 03 00
$P 04 84 ;SUB R0 R1       ; R0 = R0 - R1，判断是否为 0
$P 05 F0 ;BZC OUT        ; 如果为 0 则跳转到 OUT
$P 06 36
$P 07 D0 ;STA 90H R0      ; 将 N 存入 90H 单元
$P 08 90
$P 09 63 ;LDI 00H R3      ; R3=0 (下界)
$P 0A 00
$P 0B D3 ;STA 91H R3      ; 存入 91H (下界)
$P 0C 91
$P 0D D0 ;STA 92H R0      ; 将 N 存入 92H (上界)
$P 0E 92
$P 0F C3 ;LAD R3 91H      ; 读取下界到 R3
$P 10 91
$P 11 C1 ;LAD R1 92H      ; 读取上界到 R1
$P 12 92
$P 13 73 ;INC R3          ; R3 = R3 + 1
$P 14 8D ;SUB R1 R3       ; R1 = R1 - R3
$P 15 F0 ;BZC Result      ; 如果 R1=0 或有借位则完成
$P 16 36
$P 17 0D ;ADD R1 R3       ; 恢复 R1 的值
$P 18 B3 ;DEC R3          ; R3 = R3 - 1
$P 19 0D ;ADD R1 R3       ; R1 = R1 + R3
$P 1A A4 ;SHR R0 R1       ; R1 逻辑右移一位到 R0 (mid)
$P 1B D0 ;STA 93H R0      ; 存储 mid 到 93H
$P 1C 93
$P 1D 42 ;MOV R2 R0       ; R2 = R0 (用于计算平方)
$P 1E 43 ;MOV R3 R0       ; R3 = R0
$P 1F B3 ;LOOP: DEC R3    ; R3 = R3 - 1
$P 20 F0 ;BZC Result      ; 如果 R3=0 则完成乘法
$P 21 25
$P 22 02 ;ADD R2 R0       ; R2 = R2 + R0
$P 23 E0 ;JMP LOOP        ; 继续循环
$P 24 1F
$P 25 C1 ;LAD R1 90H      ; 读取原始数 N 到 R1
$P 26 90
$P 27 86 ;SUB R2 R1       ; R2 = R2 - R1
$P 28 F0 ;BZC Result      ; 如果 R2=0 或有借位则跳转
$P 29 2E
$P 2A D0 ;STA 92H R0      ; 更新上界为 mid
$P 2B 92
$P 2C E0 ;JMP 0F          ; 继续二分查找
$P 2D 0F
$P 2E 06 ;ADD R2 R1       ; 恢复 R2 的值
$P 2F 89 ;SUB R1 R2       ; R1 = R1 - R2
$P 30 F0 ;BZC OUT         ; 如果 R1=0 则跳转
$P 31 36
```

```
$P 32 D0 ;STA 91H R0        ; 更新下界为 mid
$P 33 91
$P 34 E0 ;JMP 0F            ; 继续二分查找
$P 35 0F
$P 36 30 ;OUT R0 40H        ; 输出结果
$P 37 40
$P 38 50 ;END: HLT          ; 停机
; //***** End Of Main Memory Data *****//
```

6.2.4机器指令校验

选择联机软件的“转储”→“刷新指令区”命令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。检查微程序控制器相应地址单元的数据是否和装载进去的十六进制数据相同，如果不同，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，先单击指令区的微存 TAB 按钮，然后单击需要修改的单元的数据，此时该单元变为编辑框，输入 6 位数据并按回车键，编辑框消失，写入的数据显示红色。

通过对左侧指令区的校验检查，发现机器指令已经正确装载完毕。

6.2.5整机测试结果及分析

本实验通过五组测试数据验证了开平方根程序的正确性和可靠性。

6.2.5.1 基础测试 计算 4、16、25 的平方根

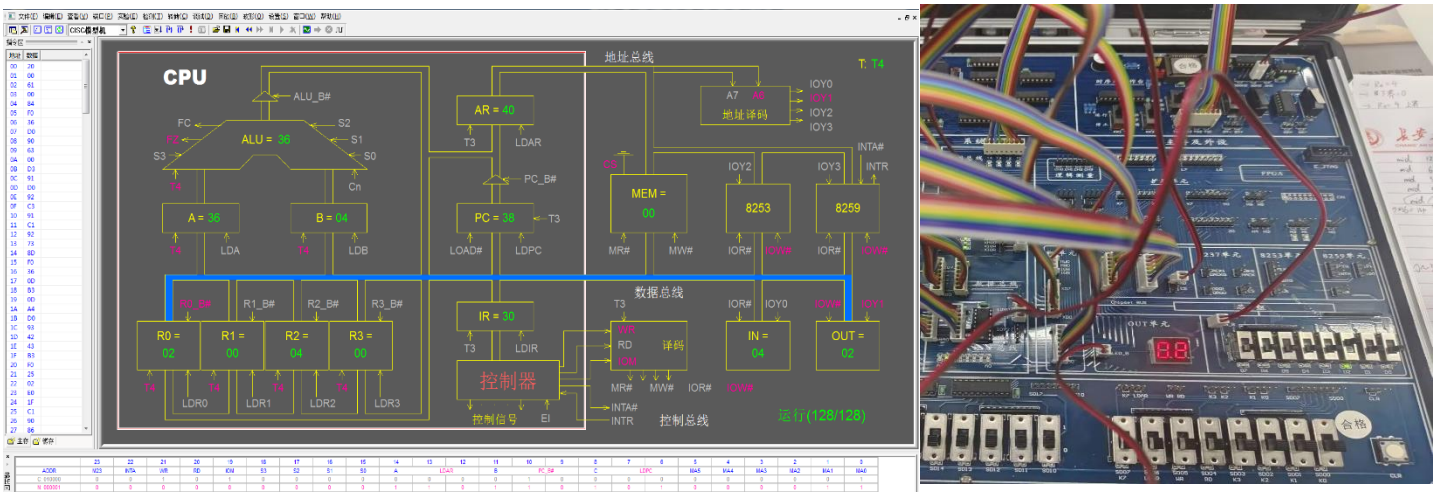

图 15：计算 4 的平方根

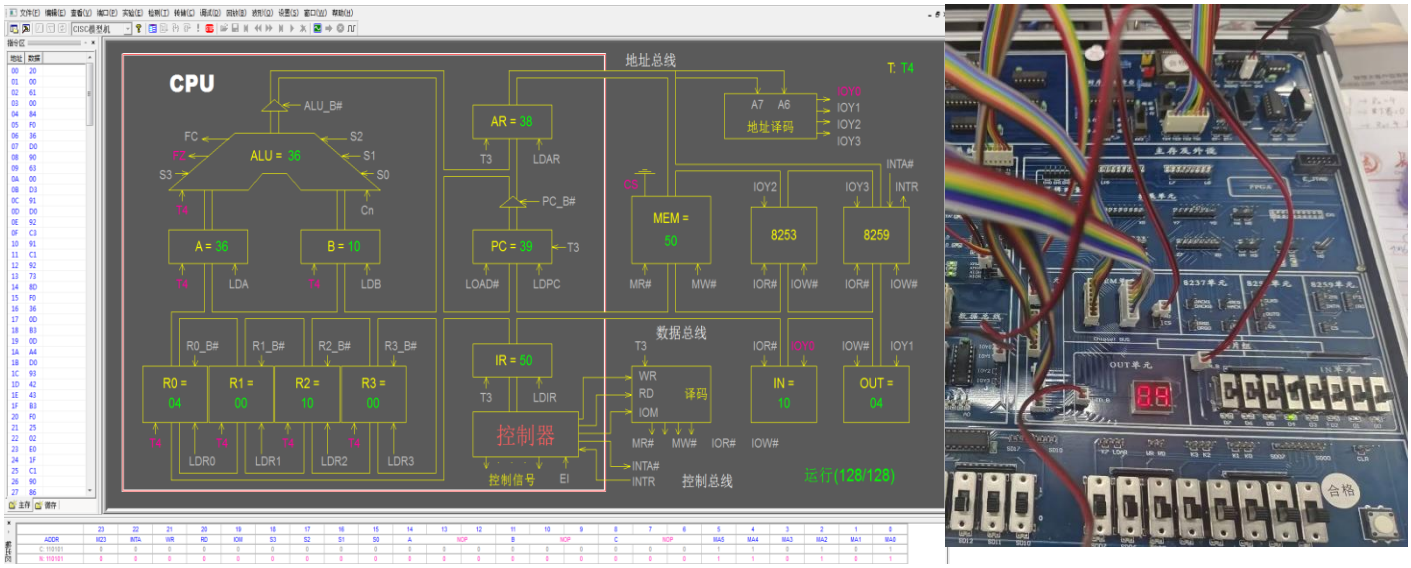

图 16：计算 16 的平方根

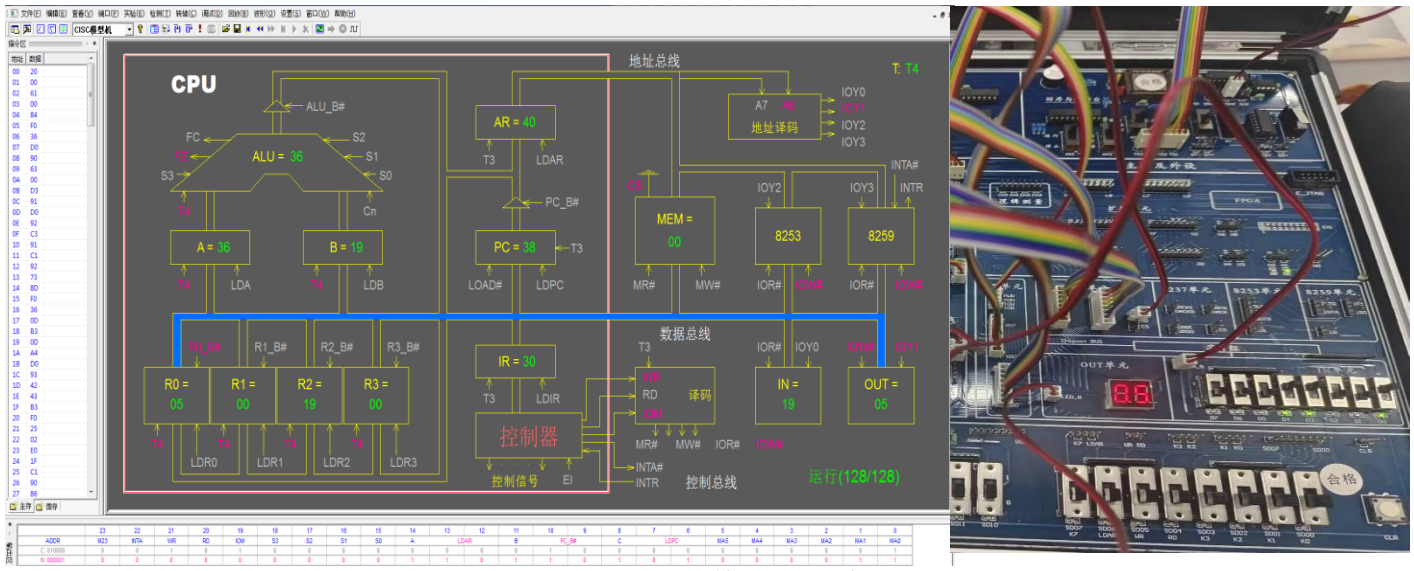

图 17：计算 25 的平方根

6.2.5.2　　特殊情况测试

1. 输入为 0 的处理

- 输入：N = 0，程序直接输出 0

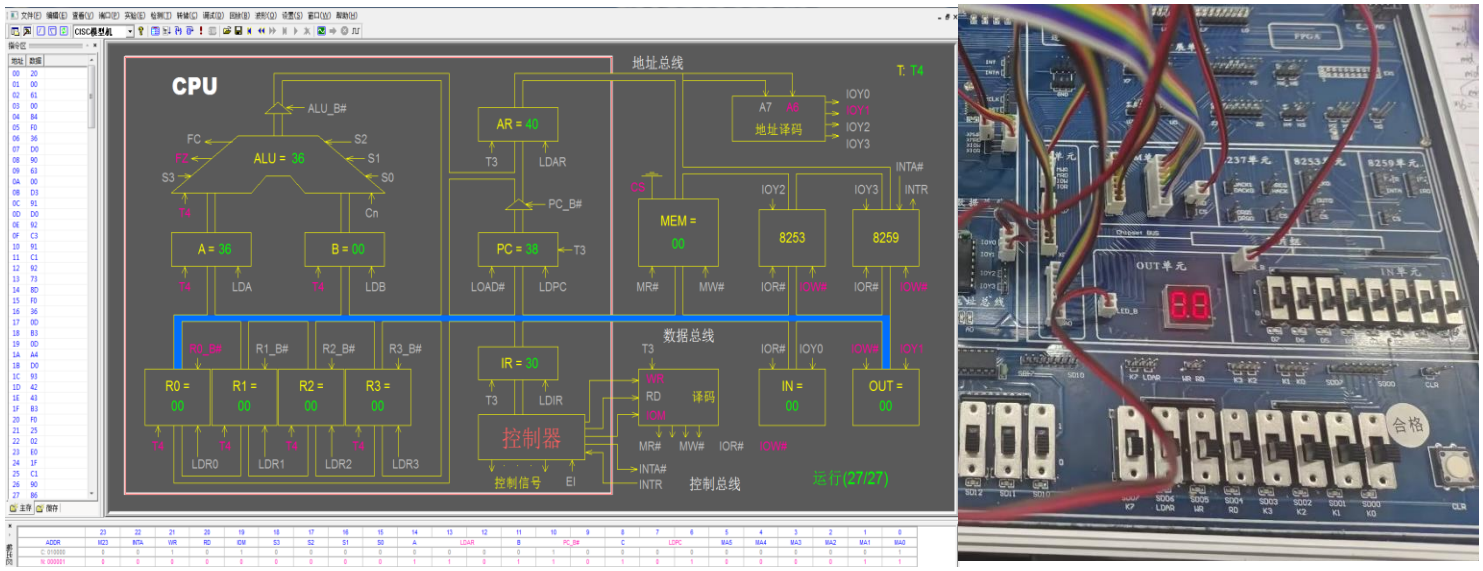

图 18：输入 0 的特殊处理

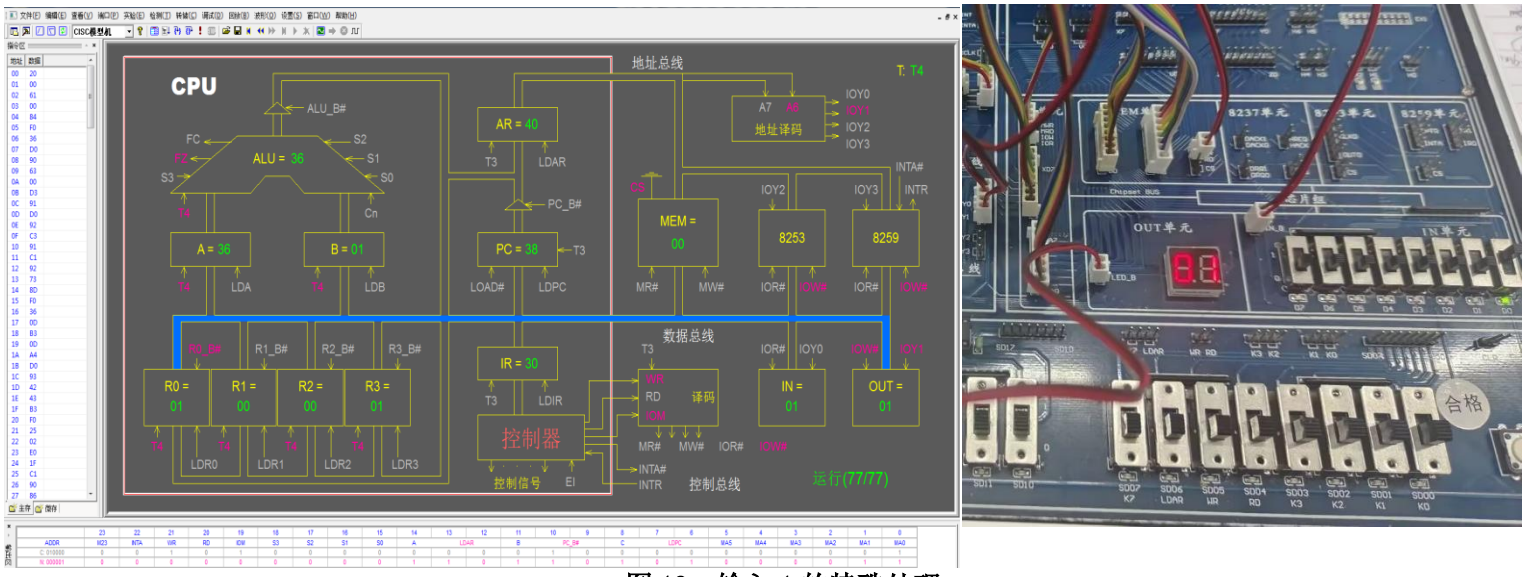

图 19：输入 1 的特殊处理

6.2.5.4　　结果分析

1. 功能完整性

- 成功实现了整数平方根的计算
- 对完全平方数和非完全平方数都能给出正确结果

2. 算法效率

- 采用二分查找算法
- 时间复杂度为 O(log n)

- 相比线性查找大大提高了效率

3. 程序稳定性

- 多次测试结果稳定，未出现死循环或异常情况

7、结论

通过本次复杂模型机的设计与实现，我们成功完成了以下目标：

7.1 硬件设计目标实现

- 成功设计并实现了一个 8 位复杂指令系统模型机
- 实现了 24 位微指令字长的控制器
- 建立了完整的数据通路和控制系统
- 合理划分了 I/O 地址空间，实现了有效的输入输出控制

7.2 指令系统设计成果

- 设计实现了 16 条功能完备的机器指令
- 支持多种寻址方式（直接、间接、变址和相对寻址）
- 指令类型覆盖了数据传送、算术逻辑运算、程序控制等
- 成功修改和优化了 SHR 和 DEC 等关键指令

7.3 微程序控制器实现

- 完成了微指令格式的合理设计
- 实现了 P<1>、P<2>、P<3>等多级分支控制
- 建立了有效的地址生成规则

7.4 应用程序开发成果

- 成功实现了整数平方根的计算功能
- 采用高效的二分查找算法，正确处理了各种边界情况和特殊输入

7.5 实践经验总结

1. 理论与实践结合
    - 深化了对计算机组成原理的理解，掌握了微程序控制器的设计方法，提高了硬件系统设计能力
2. 问题解决能力
    - 成功解决了指令执行过程中的各种问题
    - 优化了算法实现，提高了系统效率
3. 系统设计经验
    - 掌握了完整的计算机系统设计流程

- 学会了硬件与软件的协同设计
- 提高了系统级的设计和实现能力

本次设计不仅完成了课程要求的各项指标，还通过实际的开发过程加深了对计算机系统的理解。在今后的学习和工作中，这些经验将为进一步学习和实践打下坚实基础。

8、致谢

感谢老师和同学们的帮助和支持，让我们能够顺利完成本次实验。

9、参考文献

[1] 计算机组成原理. 唐朔飞. 高等教育出版社. 2014.

[2] 计算机组成原理. 王爱英. 清华大学出版社. 2018.

参考链接：

https://blog.csdn.net/weixin_48388330/article/details/122500174

https://blog.csdn.net/qq_63306482/article/details/128568976