

苏州大学实验报告

院、系	计算机学院	年级专业	21 计科	姓名	赵鹏	学号	2127405037
课程名称	计算机组成及系统结构					成绩	
指导教师	张春生	同组实验者	无	实验日期	2023.5.25		

实 验 名 称

实验四、基本模型机设计与实现

一、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实验的基础上，进一步将其构造成一台基本模型计算机。
3. 为该模型机定义五条机器指令，并编写相应的微程序，调试掌握整机概念。

二、实验内容

定义五条机器指令，编写相应的微程序，实现基本模型计算机功能。设计一段机器程序，读入一个数据，存于寄存器中，再将该数据自加，结果送输出单元显示。
修改程序，实现 2+3、7-5 功能。

三、实验原理

1. 基本模型计算机原理

本实验要实现一个简单的 CPU，并且在此 CPU 的基础上，继续构建一个简单的模型计算机。在 CPU 中写入相应的微指令，在主存中存放机器指令，再加上基本的输入输出部件，即可构成一个简单的模型计算机。

CPU 由运算器（ALU）、微程序控制器（MC）、通用寄存器（R0），指令寄存器（IR）、程序计数器（PC）和地址寄存器（AR）组成，如图 1 所示。

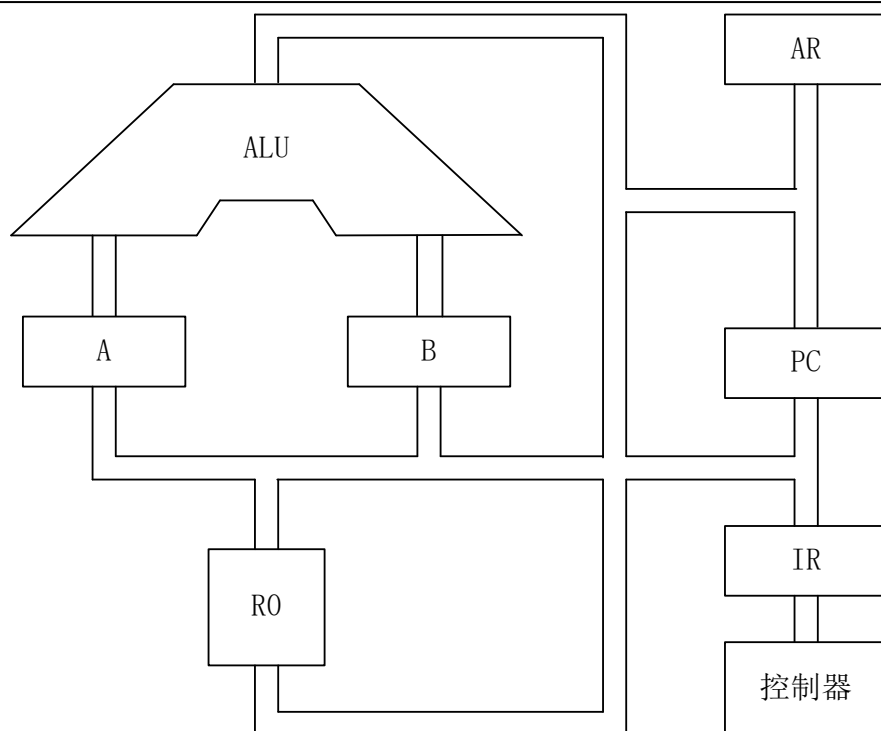


图 1 基本 CPU 构成原理图

系统的程序计数器 (PC) 和地址寄存器 (AR) 集成在一片 CPLD 芯片中, 其原理如图 2 所示。CLR 连接至 CON 单元的总清端 CLR, 按下 CLR 按钮, 将使 PC 清零。LDPC 和 T3 相与后作为计数器的计数时钟, 当 LOAD 为低电平时, 计数时钟到来后将 CPU 内总线上的数据打入 PC。

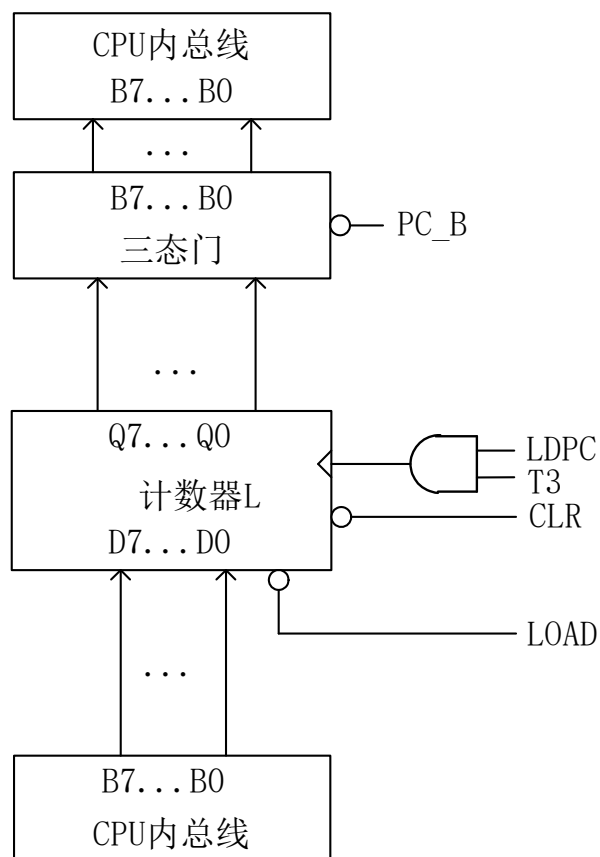


图 2 程序计数器(PC)原理图

除了程序计数器（PC），其余部件在前面的实验中都已用到，不再详细讨论。

2. 微程序设计

微指令字长共 24 位，控制位顺序如表 1 所示。

表 1 微指令格式

23	22	21	20	19	18~15	14~12	11~9	8~6	5~0
M23	M22	WR	RD	IOM	S3~S0	A	B	C	MA5~MA0

本实验中各控制位的解释如下：

WR, RD, IOM 为读写控制信号，其功能表如表 2 所示。IOM 用来选择是对 I/O 还是对 MEM 进行读写操作，IOM=0 时对 MEM 进行读写操作，IOM=1 时对 I/O 设备进行读写操作。RD=1 时为读，WR=1 时为写。

表 2 读写控制逻辑功能表

IOM	RD	WR	XM RD	XM WR	XI OR	XI OW
0	1	0	0	1	1	1
0	0	1	1	0	1	1
1	1	0	1	1	0	1
1	0	1	1	1	1	0

S3~S0 为运算器控制信号，其功能表如表 3 所示。

表 3 运算器控制信号功能表

运算类型	S3 S2 S1 S0	CN	功能
逻辑运算	0000	X	F=A（直通）
	0001	X	F=B（直通）
	0010	X	F=AB（FZ）
	0011	X	F=A+B（FZ）
	0100	X	F=/A（FZ）
移位运算	0101	X	F=A 不带进位循环右移 B（取低 3 位）位（FZ）
	0110	0	F=A 逻辑右移一位（FZ）
		1	F=A 带进位循环右移一位（FC, FZ）
	0111	0	F=A 逻辑左移一位（FZ）
		1	F=A 带进位循环左移一位（FC, FZ）
算术运算	1000	X	置 FC=CN（FC）
	1001	X	F=A 加 B（FC, FZ）
	1010	X	F=A 加 B 加 FC（FC, FZ）
	1011	X	F=A 减 B（FC, FZ）
	1100	X	F=A 减 1（FC, FZ）
	1101	X	F=A 加 1（FC, FZ）
	1110	X	（保留）
	1111	X	（保留）

微指令中 MA5~MA0 为 6 位的后续微地址；A、B、C 为三个译码字段，分别由三个控制位译码出多位。ABC 各字段解释如表 4 所示。C 字段中的 P<1>为测试字位。其功能是根据机器指令及相应微代码进行译码，使微程序转入相应的微地址入口，从而实现完成对指令的识别，并实现微程序的分支。

表 4 微指令 ABC 各字段解释

A 字段				B 字段				C 字段			
14	13	12	选 择	11	10	9	选 择	8	7	6	选 择
0	0	0	NOP	0	0	0	NOP	0	0	0	NOP
0	0	1	LDA	0	0	1	ALU-B	0	0	1	P<1>
0	1	0	LDB	0	1	0	R0-B	0	1	0	保留
0	1	1	LDR0	0	1	1	保留	0	1	1	保留
1	0	0	保留	1	0	0	保留	1	0	0	保留
1	0	1	LOAD	1	0	1	保留	1	0	1	LDPC
1	1	0	LDAR	1	1	0	PC-B	1	1	0	保留
1	1	1	LDIR	1	1	1	保留	1	1	1	保留

本模型机共有五条指令：IN（输入）、ADD（二进制加法）、OUT（输出）、JMP（无条件转移）、HLT（停机），指令格式如表 5 所示（高 4 位为操作码）。

表 5 机器指令格式

助记符	机器指令码	说明
IN	0000 0000	IN -> R0
ADD	0010 0000	R0 + R0 -> R0
OUT	0011 0000	R0 -> OUT
JMP addr	1110 0000 **** *	addr -> PC
HLT	0101 0000	停机

其中 JMP 为双字节指令，其余均为单字节指令，**** * 为 addr 对应的二进制地址码。本实验中，CPU 自动从存储器读取指令并执行。由此可设计数据通路图，如图 3 所示。

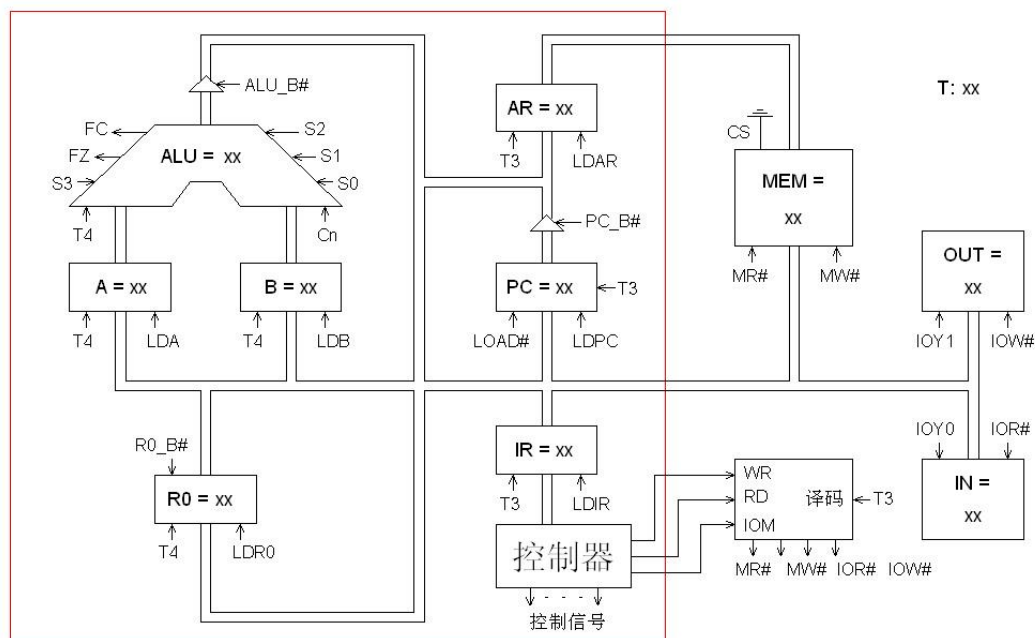


图 3 数据通路图

系统涉及到的微程序流程如图 4 所示（图中单元地址为十六进制）。当拟定“取指”微指令时，该微指令的判别测试字段为 P<1>测试。指令译码原理如图 5 所示，由于“取指”微指令是所有微程序都使用的公用微指令，因此 P<1> 的测试结果出现多路分支。本机用指令寄存器的高 6 位（IR7~IR2）作为测试条件，出现 5 路分支，占用 5 个固定微地址单元，剩下的其它地方就可以一条微指令占用控存一个微地址单元，随意填写。

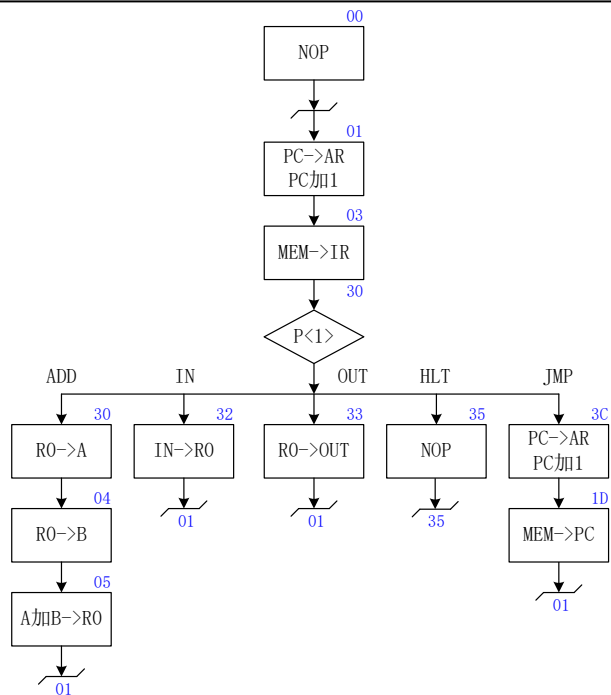


图 4 微程序流程图

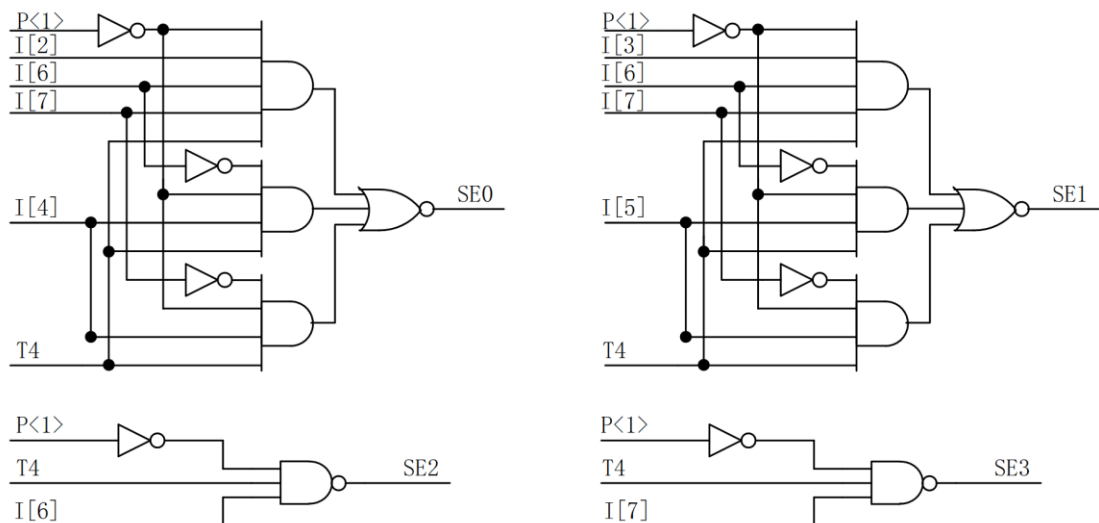


图 5 指令译码原理图

当全部微程序设计完毕后，应将每条微指令代码化，表 6 为将图 4 所示的微程序流程图按微指令格式转化而成的“二进制微代码表”。

表 6 二进制微代码表

地址	十六进制	高五位	S3~S0	A 字段	B 字段	C 字段	MA5~MA0
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
1D	10 51 41	00010	0000	101	000	101	000001
30	00 14 04	00000	0000	001	010	000	000100
32	18 30 01	00011	0000	011	000	000	000001

33	28 04 01	00101	0000	000	010	000	000001
35	00 00 35	00000	0000	000	000	000	110101
3C	00 6D 5D	00000	0000	110	110	101	011101

根据联机下载的微指令格式（图 6）将其转换成相应格式代码为：

微指令格式说明：

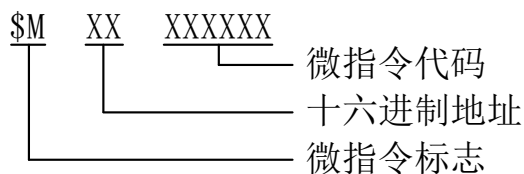


图 6 微指令格式

```
$M 00 000001;
$M 01 006D43;
$M 03 107070;
$M 04 002405;
$M 05 04B201;
$M 1D 105141;
$M 30 001404;
$M 32 183001;
$M 33 280401;
$M 35 000035;
$M 3C 006D5D;
```

3. 数据自加的程序设计

设计一段机器程序，要求从 IN 单元读入一个数据，存于 R0，将 R0 和自身相加，结果存于 R0，再将 R0 的值送 OUT 单元显示。

根据要求可以得到如下程序（表 7），地址和内容均为二进制数。

表 7 机器程序

地址	内容	助记符	说明
0000 0000	0010 0000	START: IN -> R0	从 IN 单元读入数据送 R0
0000 0001	0000 0000	ADD R0,R0	R0 和自身相加，结果送 R0
0000 0010	0011 0000	OUT R0	R0 的值送 OUT 单元显示
0000 0011	1110 0000	JMP START	跳转至 00H 地址
0000 0100	0000 0000		
0000 0101	0101 0000	HLT	停机

根据联机下载的机器指令格式（图 7）将其转换成相应格式代码为：

机器指令格式说明：

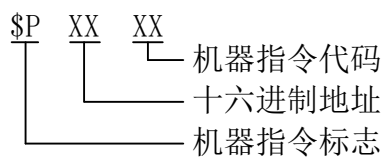


图 7 机器指令格式

```
$P 00 20;
$P 01 00;
$P 02 30;
$P 03 E0;
$P 04 00;
$P 05 50;
```

4. 实现 2+3、7-5 功能的程序设计

由于现有的微指令不能满足需求，需要对表 6 的二进制微代码进行修改。

首先是将“R0 和自身相加”改为两数相加，需要修改图 4 中 ADD 分支。该分支前两条微指令分别是 R0->A 和 R0->B，任意修改一个为从 IN 获取数据。这里修改 R0->B 为 IN->B，具体做法是：由表 1 的微指令格式可知，控制读写信号需做修改，WR=0，RD=1，IOM=1，即高五位改为 00011，以获取来自 I/O 设备的数据，还需根据表 4 中 B 字段解释，将 B 字段从 010（R0->B）改为 000（NOP）。由图 4 可知 R0->B 的地址为 04，修改表 6 中地址为 04 的微代码为：

地址	十六进制	高五位	S3~S0	A 字段	B 字段	C 字段	MA5~MA0
04	18 20 05	00011	0000	010	000	000	000101

其次是实现 7-5 中的减法，仍需要修改图 4 中 ADD 分支。该分支第三条指令为 A+B->R0，需修改为 A-B->R0。具体做法是：由表 1 的微指令格式可知，S3~S0 是运算器控制信号，可以根据表 3 将其改为 1011（A 减 B）。由图 4 可知 A+B->R0 的地址为 05，修改表 6 中地址为 05 的微代码为：

地址	十六进制	高五位	S3~S0	A 字段	B 字段	C 字段	MA5~MA0
05	05 B2 01	00000	1011	011	001	000	000001

最后使用表 7 的程序即可实现 2+3（仅做第一处修改）和 7-5（两处均修改）功能。

四、实验步骤

1. 按图 8 连接实验线路。

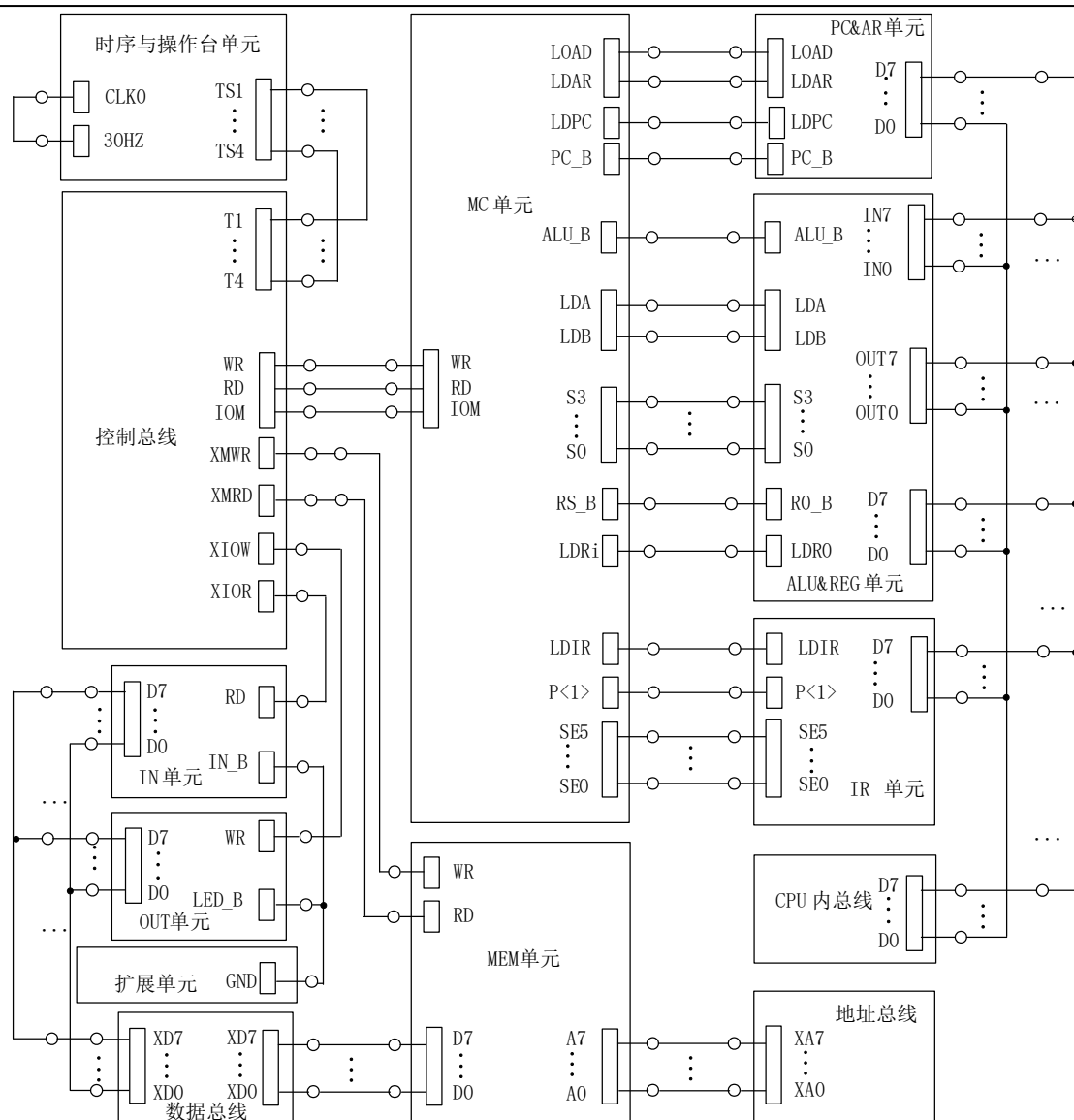


图 8 实验接线图

2. 联机写入实验程序，并进行校验。

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 //
; //***** //
; //***** 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 **/

```

具体方法为：

(1) 写入微程序

单击“【开始】/【程序】/TangDu/CMA/CMA”的程序。选择联机软件的“【转储】—【装载】”功能，在打开文件对话框中选择本次实验文件（TangDu/CMA/CMA/Sample/ CPU 与简单模型机设计实验.Txt），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。

(2) 修改微程序

可以通过联机软件单独修改某个单元的指令。本实验中 2+3、7-5 功能需修改微指令：先用鼠标左键单击指令区的“微存”，然后再单击需修改 04、05 单元的数据，此时该单元变为编辑框，输入 6 位修改数据并回车，编辑框消失，并以红色显示写入的数据。

3. 联机运行程序。

将时序与操作台单元的开关 KK1 和 KK3 置为“运行”档，用联机软件的“【实验】—【简单模型机】”打开简单模型机数据通路图。使用联机软件的“【转储】—【装载】”功能将该格式 (*.TXT) 文件（TangDu/CMA/CMA/Sample/ CPU 与简单模型机设计实验.Txt）装载入实验系统。

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预期结果（R0 自加为 IN 单元值的 2 倍，2+3 实验为 5，7-5 实验为 2）。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。

五、实验结果

实验中实际接线图 9 如所示，上位机（PC）联机运行“7-5”程序界面如图 10 所示。

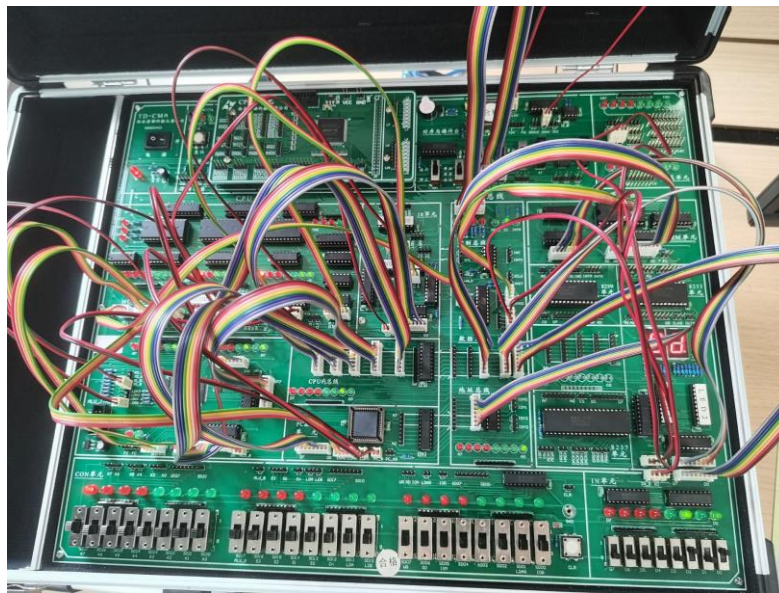


图 9 实验接线图

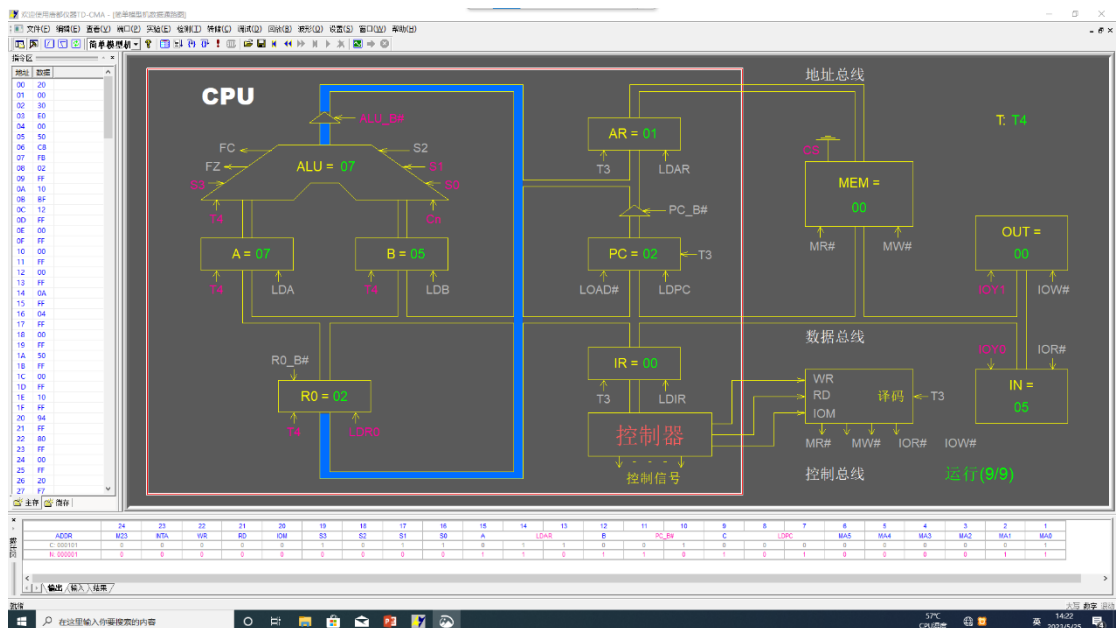
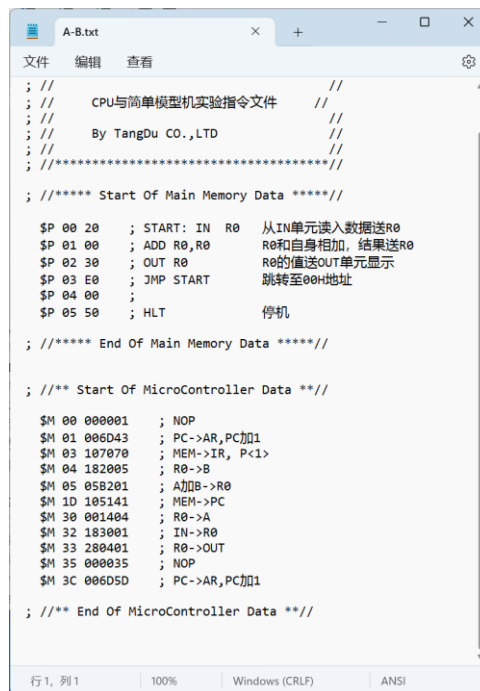
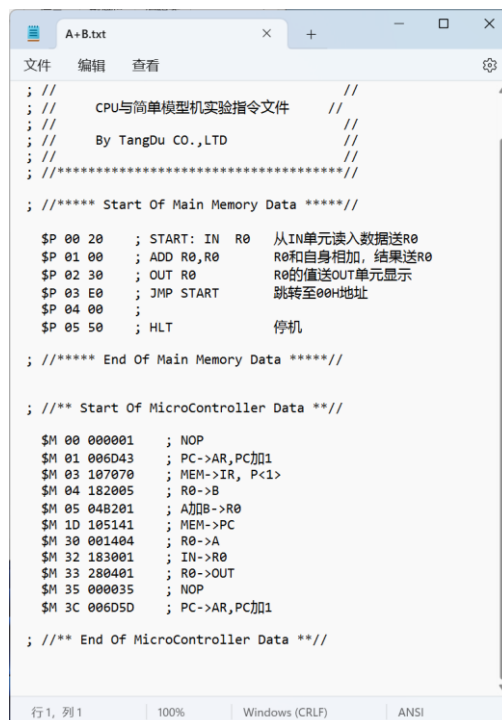


图 10 联机运行界面



```
//  
// 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 00001 ; NOP  
$M 01 006D43 ; PC->AR,PC加1  
$M 03 107070 ; MEM->IR, P<1>  
$M 04 182005 ; R0->B  
$M 05 058201 ; 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 **/  
  
行 1, 列 1 100% Windows (CRLF) ANSI
```

图 11 修改执行 A-B 的微程序



```
//  
// 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 00001 ; NOP  
$M 01 006D43 ; PC->AR,PC加1  
$M 03 107070 ; MEM->IR, P<1>  
$M 04 182005 ; R0->B  
$M 05 048201 ; 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 **/  
  
行 1, 列 1 100% Windows (CRLF) ANSI
```

图 12 修改执行 A+B 的微程序

六、实验体会

通过这次实验我, 掌握了一个简单 CPU 的组成原理, 在此基础上, 进一步将其构造成一台基本模型计算机, 并为该模型机定义五条机器指令, 并编写相应的微程序, 调试掌握整机概念。本次实验较为顺利, 接线一次正确, 完成较好。