复杂模型机设计实验

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、实验设备

PC 机一台，TDX-CMX 实验系统一套。

三、实验原理

下面讲述一下模型计算机的数据格式及指令系统。

1、数据格式

模型机规定采用定点无符号数表示数据，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： $0 \leqslant X \leqslant 2^8-1$。

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。

3、指令格式

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

7 6 5 4 （1）	3 2 （1）	1 0 （1）	7—0（2）
OP-CODE	RS	RD	P

其中括号中的 1 表示指令的第一字节，2 表示指令的第二字节，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，系统的 I/O 地址译码原理见图 1（在地址总线单元）。

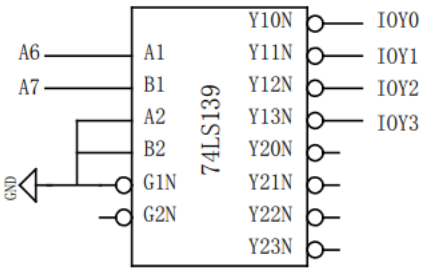

图 1 I/O 地址译码原理图

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 1 所示：

表 1 I/O 地址空间分配

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。

LDI 的指令格式如下，第一字节同前一样，第二字节为立即数。

7 6 5 4 （1）	3 2 （1）	1 0 （1）	7—0（2）
OP-CODE	RS	RD	data

LAD、STA、JMP 和 BZC 指令格式如下。

7 6 5 4 （1）	3 2 （1）	1 0 （1）	7—0（2）
OP-CODE	M	RD	D

其中 M 为寻址模式，具体见表 2，以 R2 做为变址寄存器 RI。

表 2 寻址方式

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、指令系统

本模型机共有 15 条基本指令，表 3 列出了各条指令的格式、汇编符号、指令功能。

表 3 指令描述

助记符号	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	RD∨RS → RD
RR RD, RS	1010 RS RD	RS右环移 → 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

四、总体设计

本模型机的数据通路框图如图 2 所示。

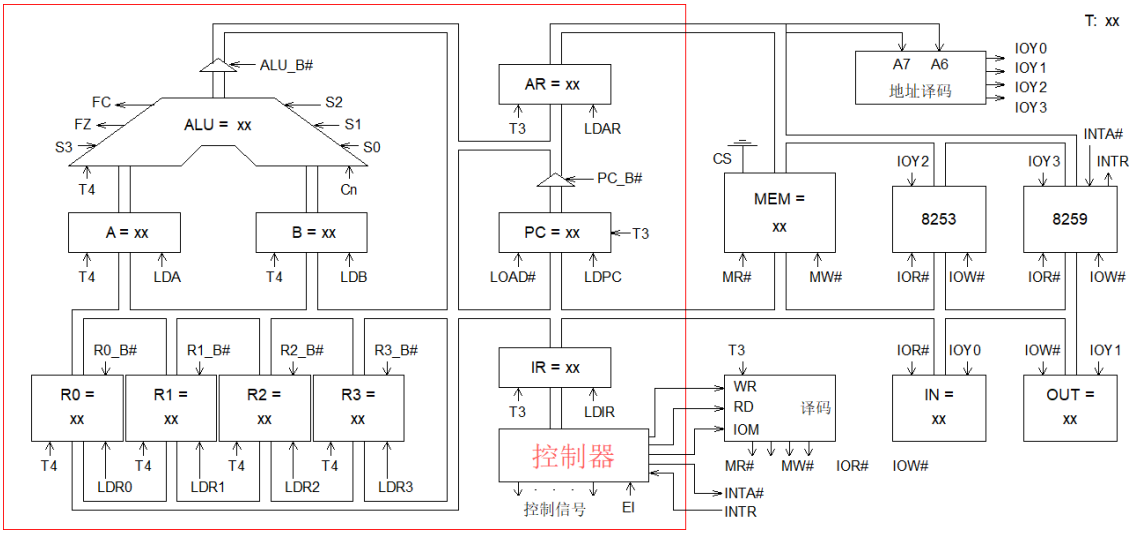

图 2　数据通路框图

和前面的实验相比，复杂模型机实验指令多，寻址方式多，只用一种测试已不能满足设计要求，为此指令译码电路需要重新设计。如图 3 所示在控制器单元的 INS_DEC 中实现。

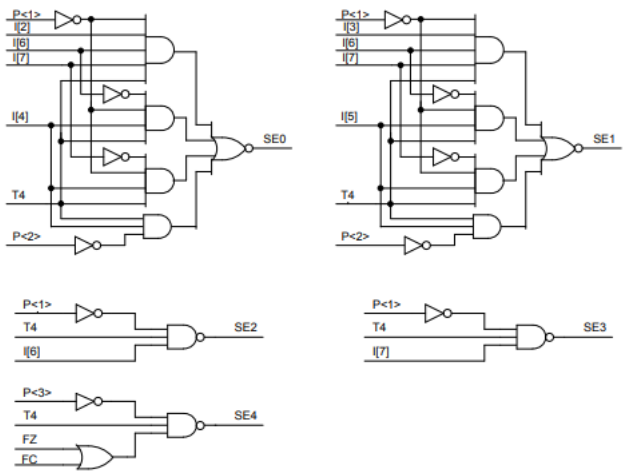

图 3　指令译码原理图

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，如图 4 所示。

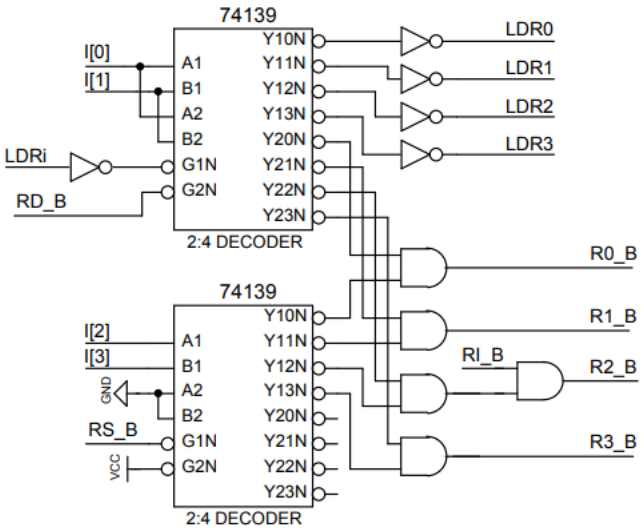

图 4　寄存器译码原理图

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 所示。

按照系统建议的微指令格式，见表 4，参照微指令流程图，将每条微指令代码化，译成二进制代码表，见表 5，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。

表 4　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

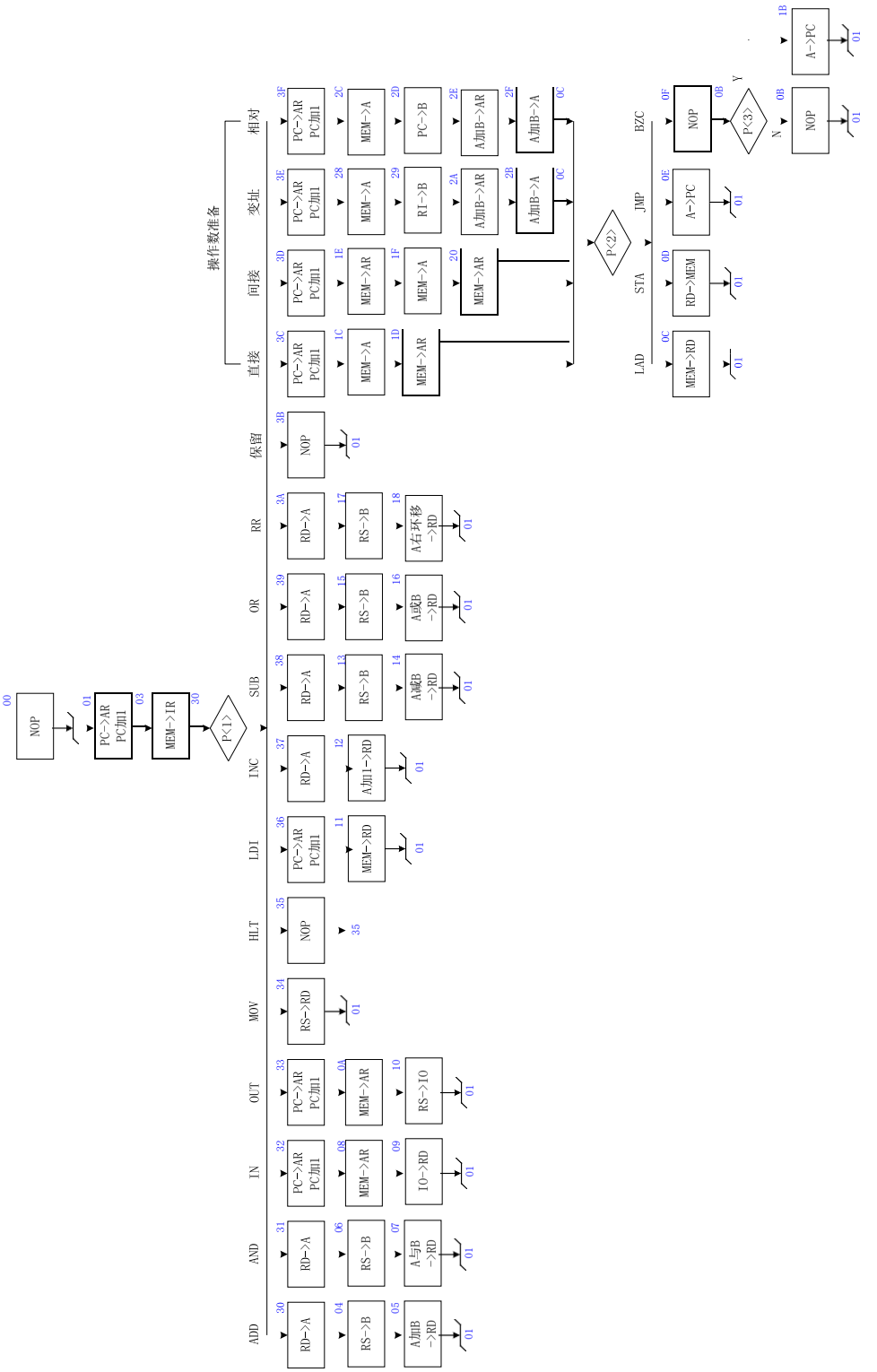
NOP
PC->AR PC加1
MEM->IR
P<1>
操作数准备
ADD
AND
IN
OUT
MOV
HLT
LDI
INC
SUB
OR
RR
保留
直接
间接
变址
相对
RD->A
RS->B
A加B ->RD
A与B ->RD
IO->RD
RS->IO
RS->RD
MEM->RD
A加1->RD
A减B ->RD
A或B ->RD
A右环移 ->RD
MEM->AR
MEM->A
RI->B
PC->B
A加B->AR
A加B->A
P<2>
LAD
STA
JMP
BZC
MEM->RD
RD->MEM
A->PC
NOP
P<3>
Y
N
A->PC

图 5 微程序流程图

表 5　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	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	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

根据现有指令，在模型机上实现以下运算：从 IN 单元读入一个数据，根据读入数据的低 4 位值 X，求 1+2+…+X 的累加和，01H 到 0FH 共 15 个数据存于 60H 到 6EH 单元。

根据要求可以得到如下程序，地址和内容均为二进制数。

```
  地 址       内 容          助记符                    说 明
00000000   00100000   ; START: IN R0,00H     从 IN 单元读入计数初值
00000001   00000000
00000010   01100001   ; LDI  R1,0FH          立即数 0FH 送 R1
00000011   00001111
00000100   00010100   ; AND  R0,R1           得到 R0 低四位
00000101   01100001   ; LDI  R1,00H          装入和初值 00H
00000110   00000000
00000111   11110000   ; BZC  RESULT          计数值为 0 则跳转
00001000   00010110
00001001   01100010   ; LDI  R2,60H          读入数据始地址
00001010   01100000
00001011   11001011   ; LOOP: LAD R3,[RI],00H   从 MEM 读入数据送 R3,
                                          变址寻址，偏移量为 00H
00001100   00000000
00001101   00001101   ; ADD  R1,R3           累加求和
00001110   01110010   ; INC  RI             变址寄存加 1，指向下一数据
00001111   01100011   ; LDI  R3,01H          装入比较值
00010000   00000001
```

```
00010001   10001100   ; SUB  R0,R3
00010010   11110000   ; BZC  RESULT          相减为 0，表示求和完毕
00010011   00010110
00010100   11100000   ; JMP  LOOP            未完则继续
00010101   00001011
00010110   11010001   ; RESULT: STA  70H,R1  和存于 MEM 的 70H 单元
00010111   01110000
00011000   00110100   ; OUT  40H,R1          和在 OUT 单元显示
00011001   01000000
00011010   11100000   ; JMP START           跳转至 START
00011011   00000000
00011100   01010000   ; HLT              停机

01100000   00000001   ; 数据
01100001   00000010
01100010   00000011
01100011   00000100
01100100   00000101
01100101   00000110
01100110   00000111
01100111   00001000
01101000   00001001
01101001   00001010
01101010   00001011
01101011   00001100
01101100   00001101
01101101   00001110
01101110   00001111
```

五、实验步骤

1 把时序与操作台单元的“MODE”用短路块短接，使系统工作在四节拍模式，JP1、JP2 短路块均将 1、2 短接，按图 6 连接实验线路，仔细检查接线后打开实验箱电源。

2 写入实验程序，并进行校验，分两种方式，手动写入和联机写入。

1) 手动写入和校验

(1) 手动写入微程序

① 将时序与操作台单元的开关 KK1 置为‘停止’档，KK3 置为‘编程’档，KK4 置为‘控存’档，KK5 置为‘置数’档。

② 使用 CON 单元的 SD15——SD10 给出微地址，IN 单元给出低 8 位应写入的数据，连续两次按动时序与操作台的开关 ST，将 IN 单元的数据写到该单元的低 8 位。

③ 将时序与操作台单元的开关 KK5 置为‘加 1’档。

④ IN 单元给出中 8 位应写入的数据，连续两次按动时序与操作台的开关 ST，将 IN 单元的数据写到该单元的中 8 位。IN 单元给出高 8 位应写入的数据，连续两次按动时序与操作台的开关 ST，将 IN 单元的数据写到该单元的高 8 位。

⑤ 重复①、②、③、④四步，将表 5 的微代码写入 E2ROM 芯片中。

(2) 手动校验微程序

① 将时序与操作台单元的开关 KK1 置为‘停止’档，KK3 置为‘校验’档，KK4 置为‘控

存’档，KK5 置为‘置数’档。

② 使用 CON 单元的 SD15——SD10 给出微地址，连续两次按动时序与操作台的开关 ST，MC 单元的指数据指示灯 M7——M0 显示该单元的低 8 位。

③ 将时序与操作台单元的开关 KK5 置为‘加 1’档。

④ 连续两次按动时序与操作台的开关 ST，MC 单元的指数据指示灯 M15——M8 显示该单元的中 8 位，MC 单元的指数据指示灯 M23——M16 显示该单元的高 8 位。

⑤ 重复①、②、③、④四步，完成对微代码的校验。如果校验出微代码写入错误，重新写入、校验，直至确认微指令的输入无误为止。

(3) 手动写入机器程序

① 将时序与操作台单元的开关 KK1 置为‘停止’档，KK3 置为‘编程’档，KK4 置为‘主存’档，KK5 置为‘置数’档。

② 使用 CON 单元的 SD17——SD10 给出地址，IN 单元给出该单元应写入的数据，连续两次按动时序与操作台的开关 ST，将 IN 单元的数据写到该存储器单元。

③ 将时序与操作台单元的开关 KK5 置为‘加 1’档。

④ IN 单元给出下一地址（地址自动加 1）应写入的数据，连续两次按动时序与操作台的开关 ST，将 IN 单元的数据写到该单元中。然后地址会又自加 1，只需在 IN 单元输入后续地址的数据，连续两次按动时序与操作台的开关 ST，即可完成对该单元的写入。

⑤ 亦可重复①、②两步，将所有机器指令写入主存芯片中。

(4) 手动校验机器程序

① 将时序与操作台单元的开关 KK1 置为‘停止’档，KK3 置为‘校验’档，KK4 置为‘主存’档，KK5 置为‘置数’档。

② 使用CON 单元的 SD17——SD10 给出地址，连续两次按动时序与操作台的开关 ST，CPU 内总线的指数据指示灯 D7——D0 显示该单元的数据。

③ 将时序与操作台单元的开关 KK5 置为‘加 1’档。

④ 连续两次按动时序与操作台的开关 ST，地址自动加 1，CPU 内总线的指数据指示灯 D7——D0 显示该单元的数据。此后每两次按动时序与操作台的开关 ST，地址自动加 1，CPU 内总线的指数据指示灯 D7——D0 显示该单元的数据，继续进行该操作，直至完成校验，如发现错误，则返回写入，然后校验，直至确认输入的所有指令准确无误。

⑤ 亦可重复①、②两步，完成对指令码的校验。如果校验出指令码写入错误，重新写入、校验，直至确认指令的输入无误为止。

2) 联机写入和校验

联机软件提供了微程序和机器程序下载功能，以代替手动读写微程序和机器程序，但是微程序和机器程序得以指定的格式写入到以 TXT 为后缀的文件中，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。

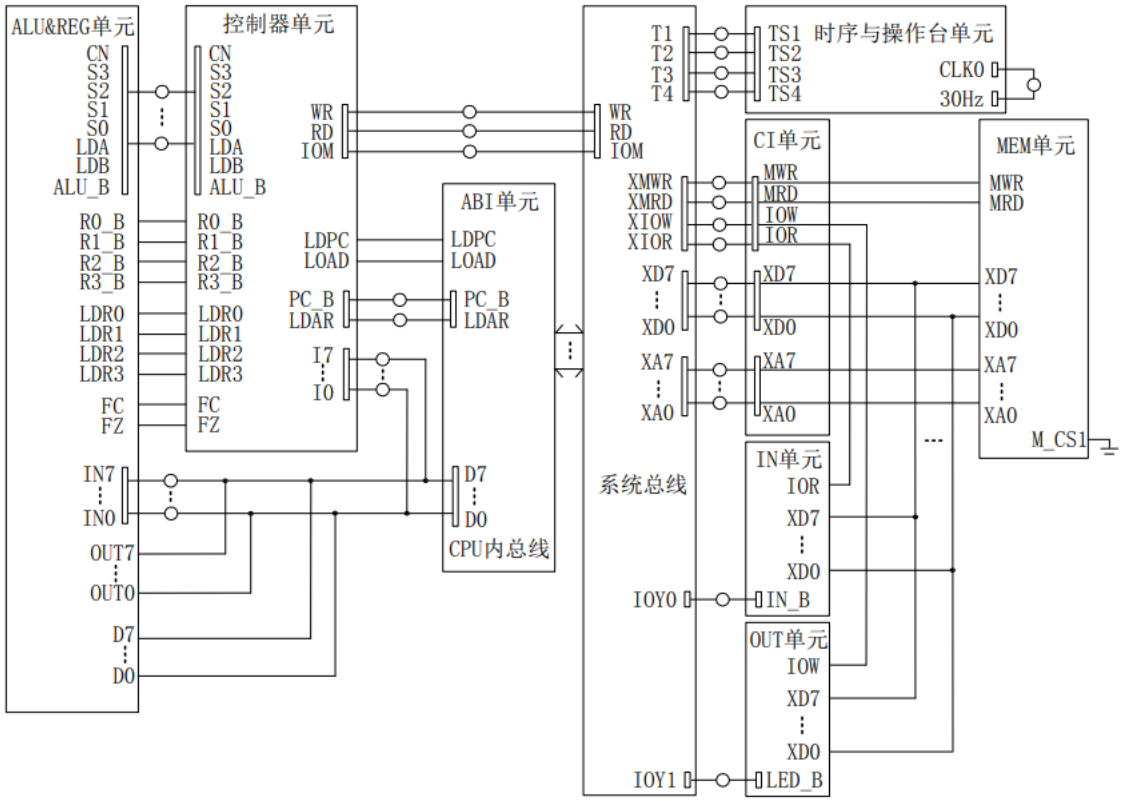
ALU&REG单元
CN
S3
S2
S1
S0
LDA
LDB
ALU_B
R0_B
R1_B
R2_B
R3_B
LDR0
LDR1
LDR2
LDR3
FC
FZ
IN7
IN0
OUT7
OUT0
D7
D0
控制器单元
CN
S3
S2
S1
S0
LDA
LDB
ALU_B
R0_B
R1_B
R2_B
R3_B
LDR0
LDR1
LDR2
LDR3
FC
FZ
WR
RD
IOM
LDPC
LOAD
PC_B
LDAR
I7
I0
ABI单元
LDPC
LOAD
PC_B
LDAR
D7
D0
CPU内总线
WR
RD
IOM
T1
T2
T3
T4
TS1 时序与操作台单元
TS2
TS3
TS4
CLK0
30Hz
CI单元
MWR
MRD
IOW
IOR
XD7
XD0
XA7
XA0
XMWR
XMRD
XIOW
XIOR
XD7
XD0
XA7
XA0
系统总线
IOY0
IOY1
IN单元
IOR
XD7
XD0
IN_B
OUT单元
IOW
XD7
XD0
LED_B
MEM单元
MWR
MRD
XD7
XD0
XA7
XA0
M_CS1

图 6　实验接线图

```
; //*************************************//
; //                                     //
; //      复杂模型机实验指令文件         //
; //                                     //
; //      By TangDu CO.,LTD              //
; //                                     //
; //*************************************//

; //****** Start Of Main Memory Data ******//
  $P 00 20   ; START: IN R0,00H      从 IN 单元读入计数初值
  $P 01 00
  $P 02 61   ; LDI  R1,0FH           立即数 0FH 送 R1
  $P 03 0F
  $P 04 14   ; AND  R0,R1            得到 R0 低四位
  $P 05 61   ; LDI  R1,00H           装入和初值 00H
  $P 06 00
  $P 07 F0   ; BZC  RESULT           计数值为 0 则跳转
  $P 08 16
  $P 09 62   ; LDI  R2,60H           读入数据始地址
  $P 0A 60
  $P 0B CB   ; LOOP: LAD R3,[RI],00H   从 MEM 读入数据送 R3，
                                     变址寻址，偏移量为 00H
  $P 0C 00
  $P 0D 0D   ; ADD  R1,R3            累加求和
  $P 0E 72   ; INC  RI               变址寄存加 1，指向下一数据
  $P 0F 63   ; LDI  R3,01H           装入比较值
  $P 10 01
  $P 11 8C   ; SUB  R0,R3
  $P 12 F0   ; BZC  RESULT           相减为 0，表示求和完毕
  $P 13 16
  $P 14 E0   ; JMP  LOOP             未完则继续
  $P 15 0B
  $P 16 D1   ; RESULT: STA  70H,R1   和存于 MEM 的 70H 单元
  $P 17 70
  $P 18 34   ; OUT  40H,R1           和在 OUT 单元显示
  $P 19 40
  $P 1A E0   ; JMP START             跳转至 START
  $P 1B 00
  $P 1C 50   ; HLT                   停机

  $P 60 01   ; 数据
  $P 61 02
  $P 62 03
  $P 63 04
  $P 64 05
  $P 65 06
  $P 66 07
  $P 67 08
  $P 68 09
  $P 69 0A
  $P 6A 0B
  $P 6B 0C
```

```
  $P 6C 0D
  $P 6D 0E
  $P 6E 0F
; //***** 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    ; 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
; //** End Of MicroController Data **//
```

选择联机软件的“【转储】—【装载】”功能，在打开文件对话框中选择上面所保存的文件， 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，可以通过联机软件单独修改某个单元的指令，以修改微指令为例，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。

3 运行程序

方法一：本机运行

将时序与操作台单元的开关 KK1、KK3 置为‘运行’档，按动 CON 单元的总清按钮 CLR，将使程序计数器 PC、地址寄存器 AR 和微程序地址为 00H，程序可以从头开始运行，暂存器 A、B，指令寄存器 IR 和 OUT 单元也会被清零。

将时序与操作台单元的开关 KK2 置为‘单步’档，每按动一次 ST 按钮，即可单步运行一条微指令，对照微程序流程图，观察微地址显示灯是否和流程一致。每运行完一条微指令，观测一次数据总线和地址总线，对照数据通路图，分析总线上的数据是否正确。

当模型机执行完 OUT 指令后，检查 OUT 单元显示的数是否正确，按下 CON 单元的总清按钮 CLR，改变 IN 单元的值，再次执行机器程序，从 OUT 单元显示的数判别程序执行是否正确。

方法二：联机运行（软件使用说明请看附录 1）

进入软件界面，选择菜单命令“【实验】—【CISC 实验】”，打开相应的数据通路图，选择相应的功能命令，即可联机运行、监控、调试程序。

按动 CON 单元的总清按钮 CLR，然后通过软件运行程序，当模型机执行完 OUT 指令后，检查 OUT 单元显示的数是否正确。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。