

# 计算机组成原理

## 实验指导书

# 目 录

实验一	基本运算器实验 .....	1
实验二	阵列乘法器设计实验 .....	7
实验三	静态随机存储器实验 .....	10
实验四	时序发生器设计实验 .....	16
实验五	微程序控制器实验 .....	19
实验六	简单模型机设计实验 .....	27
实验七	硬布线控制器模型机设计实验.....	33
实验八	复杂模型机设计实验 .....	37
实验九	带中断处理能力的模型机设计实验.....	49
附录 1	软件使用说明.....	61
附录 2	时序单元介绍.....	70
附录 3	实验用芯片介绍.....	72

# 实验一 基本运算器实验

计算机的一个最主要的功能就是处理各种算术和逻辑运算，这个功能要由 CPU 中的运算器来完成，运算器也称作算术逻辑部件 ALU。首先安排基本运算器实验，了解运算器的基本结构。

## 1.1 实验目的

- (1) 了解运算器的组成结构。
- (2) 掌握运算器的工作原理。

## 1.2 实验设备

PC 机一台，TD-CMX 实验系统一套。

## 1.3 实验原理

本实验的原理如图 1-1 所示。

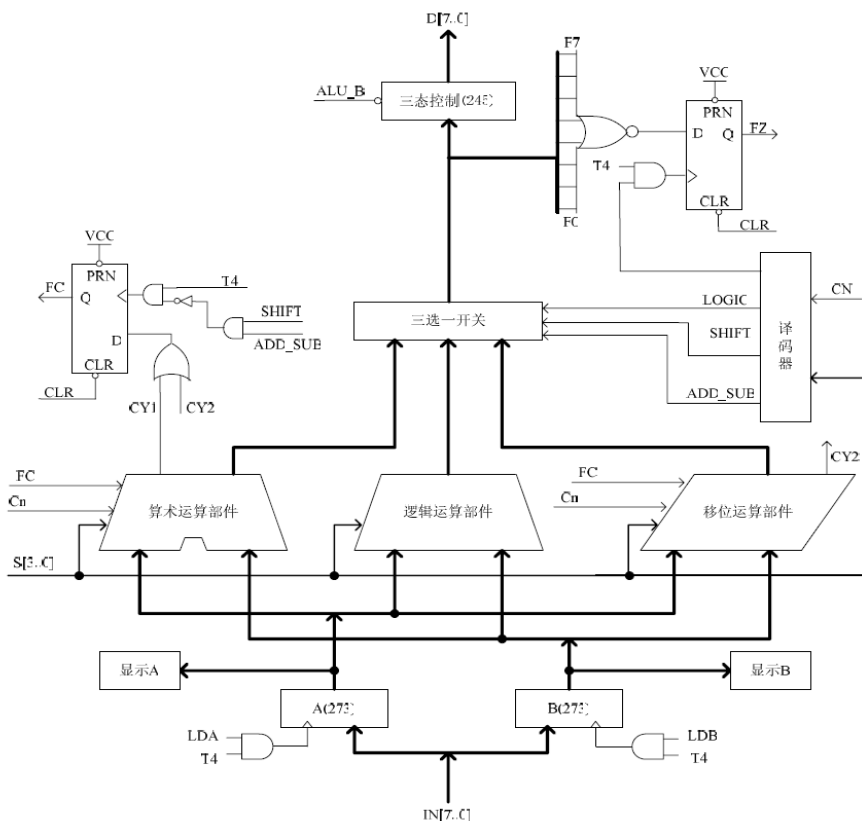


图 1-1 运算器原理图

运算器内部含有三个独立运算部件，分别为算术、逻辑和移位运算部件，参与运算的数据

存于暂存器 A 和暂存器 B 中，算术、逻辑运算部件以及移位部件同时接受来自 A 和 B 的数据（有些处理器体系结构把移位运算器放于算术和逻辑运算部件之前，如 ARM），各部件对操作数进行何种运算由控制信号 S3...S0 来决定，任何时候，多路选择开关只选择三部件中一个部件的结果作为 ALU 的输出。如果是算术运算，还将置进位标志 FC，在运算结果输出前，置 FZ 零标志。ALU 中所有模块集成在一片 CPLD 中。

逻辑运算部件由逻辑门构成，较为简单，而后面又有专门的算术运算部件设计实验，在此对这两个部件不再赘述。移位运算采用的是桶形移位器，一般采用交叉开关矩阵来实现，交叉开关的原理如图 1-2 所示。图中显示的是一个 4X4 的矩阵（系统中是一个 8X8 的矩阵）。每一个输入都通过开关与一个输出相连，把沿对角线的开关导通，就可实现移位功能，即：

(1) 对于逻辑左移或逻辑右移操作，将一条对角线的开关导通，这将所有的输入位与所使用的输出分别相连，而没有同任何输入相连的则输出连接 0。

(2) 对于循环右移操作，右移对角线同互补的左移对角线一起激活。例如，在 4 位矩阵中使用‘右 1’和‘左 3’对角线来实现右循环 1 位。

(3) 对于未连接的输出位，算术右移使用符号扩展而不是 0 填充。使用另外的逻辑进行移位总量译码和符号判别。

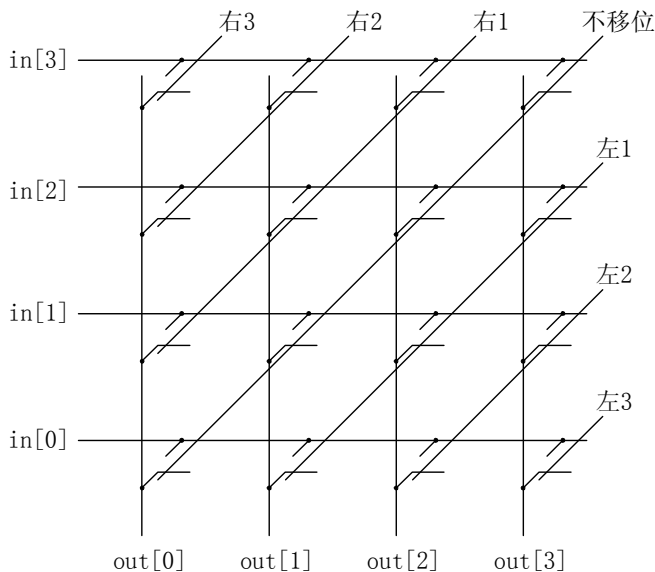



图 1-2 交叉开关桶形移位器原理图

运算器单元由以下部分构成：一片 CPLD 实现的 ALU，两片 74LS245 构成的保护电路。ALU 的输入和输出通过三态门 74LS245 连到 CPU 内总线上，另外还有进位标志 FC 和零标志 FZ 指示灯。图 1-1-1 中有三部分不在 CPLD 中实现，而是在外围电路中实现，这三部分为图中的‘显示 A’、‘显示 B’和 ALU 的输出控制‘三态控制 245’，请注意：**实验箱上凡丝印标注有马蹄形标记‘’，表示这两根排针之间是连通的。**图中除 T4 和 CLR，其余信号均来自于 ALU 单元的排线座，实验箱中所有单元的 T1、T2、T3、T4 都连接至 MC 单元的 T1、T2、T3、T4，CLR 都连接

至 CON 单元的 CLR 按钮。T4 由时序单元的 TS4 提供（时序单元的介绍见附录二），其余控制信号均由 CON 单元的二进制数据开关模拟给出。控制信号中除 T4 为脉冲信号外，其余均为电平信号，其中#ALU\_B 为低有效，其余均为高有效。ALU\_B 信息的标注可以理解为 ALU 输出送总线 BUS。

暂存器 A 和暂存器 B 的数据能在 LED 灯上实时显示，原理如图 1-3 所示（以 A0 为例，其它相同）。进位标志 FC、零标志 FZ 和数据总线 D7...D0 的显示原理也是如此。

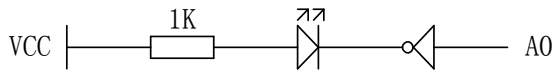


图 1-3 A0 显示原理图

ALU 和外围电路的连接如图 1-4 所示，由于 ALU 的工作电压为 3.3V，所以在所有用户操作的 I/O 脚都加上 74LS245 加以隔离保护，以防误操作烧坏 ALU 芯片，图中的小方框代表排针座。

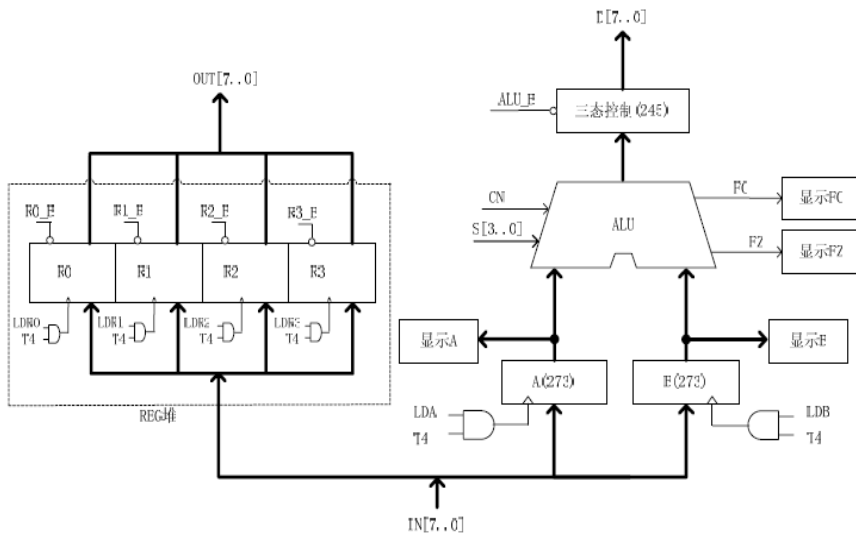


图 1-4 ALU 和外围电路连接原理图

运算器的逻辑功能表如表 1-1 所示，其中 S3 S2 S1 S0 为 4 位控制信号，Cn 为来自低位的进位信号，FC 为向高位的进位标志，FZ 为运算器零标志，表中功能栏内的 FC、FZ 表示当前运算会影响到该标志。

表 1-1 运算器逻辑功能表

运算类型	S3 S2 S1 S0	CN	功 能
逻辑运算	0000	X	F=A (直通) (FZ)
	0001	X	F=B (直通) (FZ)
	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 带进位逻辑右移一位 (FC, FZ)
	0111	0	F=A 逻辑左移一位 (FZ)
1		F=A 带进位逻辑左移一位 (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)

## 1.4 实验步骤

- (1) 按图 1-5 连接实验电路，并检查无误。图中将用户需要连接的信号用圆圈标明
- (2) 将时序与操作台单元的开关 KK2 置为‘单步’档，开关 KK1、KK3 置为‘运行’档。
- (3) 打开电源开关，如果听到有‘嘀’报警声，说明有总线竞争现象，应立即关闭电源，重新检查接线，直到错误排除。然后按动 CON 单元的 CLR 按钮，将运算器的 A、B 和 FC、FZ 清零。
- (4) 用输入开关向暂存器 A 置数。

① 拨动 CON 单元的 SD27…SD20 数据开关，形成二进制数 01100101 (或其它数值)，数据显示亮为‘1’，灭为‘0’。

② 置 LDA=1, LDB=0, 按动时序单元的 ST 按钮，产生一个 T4 上沿，则将二进制数 01100101 置入暂存器 A 中，暂存器 A 的值通过 ALU 单元的 A7…A0 八位 LED 灯显示。

(5) 用输入开关向暂存器 B 置数。

① 拨动 CON 单元的 SD27…SD20 数据开关，形成二进制数 10100111 (或其它数值)。

② 置 LDA=0, LDB=1, 按动时序单元的 ST 按钮，产生一个 T4 上沿，则将二进制数 10100111 置入暂存器 B 中，暂存器 B 的值通过 ALU 单元的 B7…B0 八位 LED 灯显示。

(6) 改变运算器的功能设置，观察运算器的输出。置 ALU\_B=0、LDA=0、LDB=0，然后按表 1-1 置 S3、S2、S1、S0 和 Cn 的数值，并观察数据总线 LED 显示灯显示的结果。如置 S3、S2、S1、S0 为 0010，运算器作逻辑与运算，置 S3、S2、S1、S0 为 1001，运算器作加法运算。

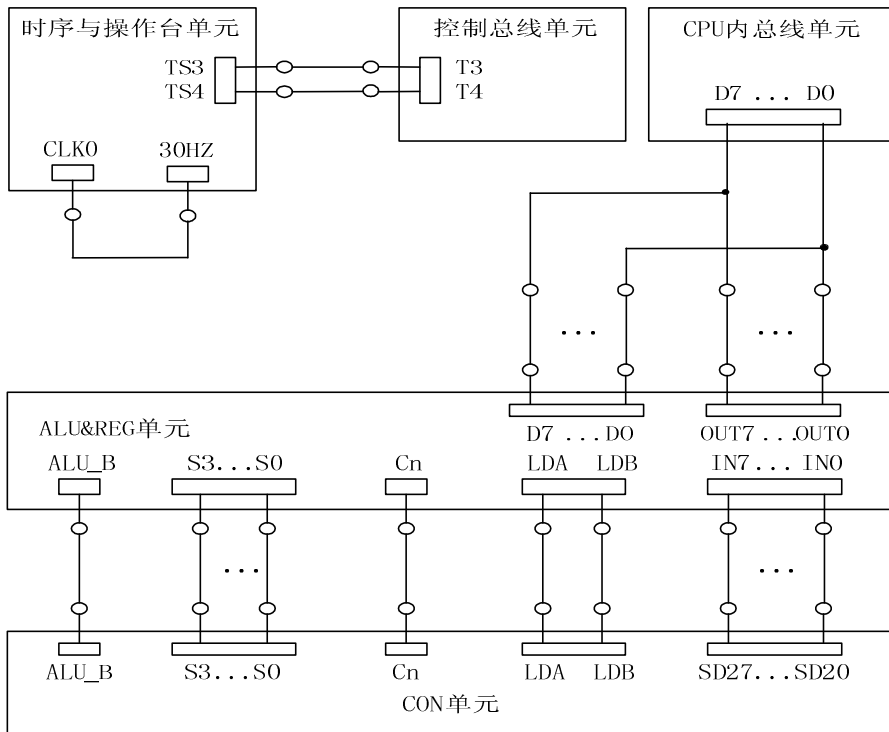


图 1-5 实验接线图

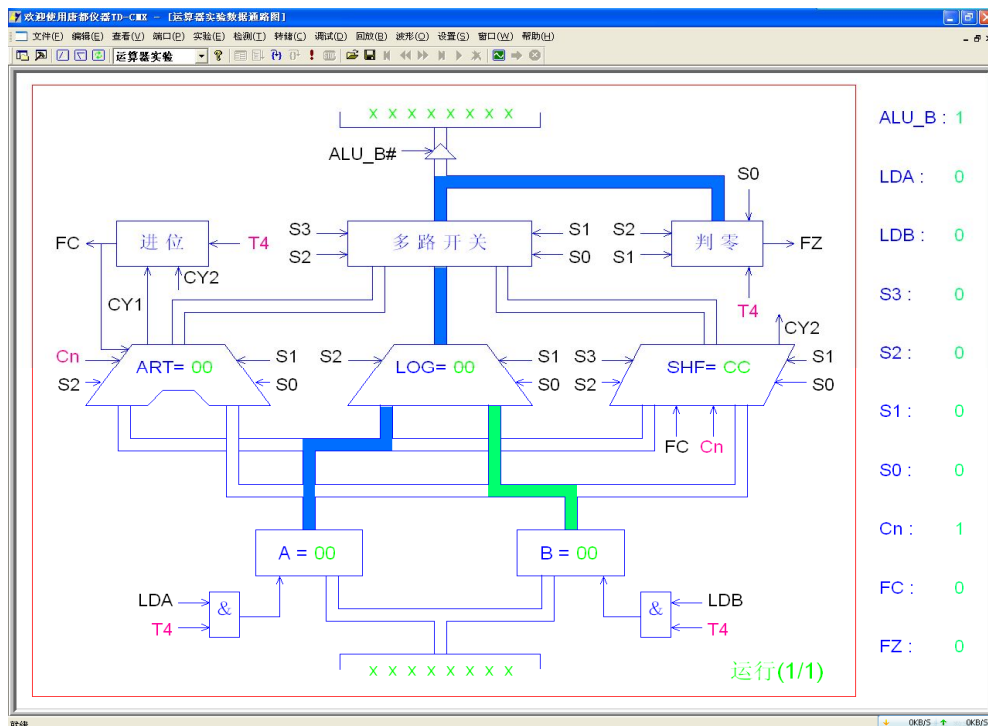


图 1-6 数据通路图

如果实验箱和 PC 联机操作，则可通过软件中的数据通路图来观测实验结果（软件使用说明请看附录），方法是：打开软件，选择联机软件的“【实验】—【运算器实验】”，打开运算器实验的数据通路图，如图 1-6 所示。进行上面的手动操作，每按动一次 ST 按钮，数据通路图会有数据的流动，反映当前运算器所做的操作，或在软件中选择“【调试】—【单周期】”，其作用相当于将时序单元的状态开关 KK2 置为‘单步’档后按动了一次 ST 按钮，数据通路图也会反映当前运算器所做的操作。

重复上述操作，并完成表 1-2。然后改变 A、B 的值，验证 FC、FZ 的锁存功能。

表 1-2 运算结果表

运算类型	A	B	S3 S2 S1 S0	结果
逻辑运算	65	A7	0 0 0 0	F=( 65 ) FC=( ) FZ=( )
	65	A7	0 0 0 1	F=( A7 ) FC=( ) FZ=( )
			0 0 1 0	F=( ) FC=( ) FZ=( )
			0 0 1 1	F=( ) FC=( ) FZ=( )
			0 1 0 0	F=( ) FC=( ) FZ=( )
移位运算			0 1 0 1	F=( ) FC=( ) FZ=( )
			0 1 1 0	F=( ) FC=( ) FZ=( )
			0 1 1 1	F=( ) FC=( ) FZ=( )
			1 0 0 0	F=( ) FC=( ) FZ=( )
算术运算			1 0 0 1	F=( ) FC=( ) FZ=( )
			1 0 1 0 (FC=0)	F=( ) FC=( ) FZ=( )
			1 0 1 0 (FC=1)	F=( ) FC=( ) FZ=( )
			1 0 1 1	F=( ) FC=( ) FZ=( )
			1 1 0 0	F=( ) FC=( ) FZ=( )
			1 1 0 1	F=( ) FC=( ) FZ=( )

运算器实验中由单步操作改为单拍操作，更详细的反映运算器在带进位运算时的运行状况。



## 实验二 阵列乘法器设计实验

### 2.1 实验目的

- (1) 掌握乘法器的原理及其设计方法。
- (2) 熟悉 CPLD 应用设计及 EDA 软件的使用。

### 2.2 实验设备

PC 机一台，TD-CMX 实验系统一套。

### 2.3 实验原理

硬件乘法器常规的设计是采用“串行移位”和“并行加法”相结合的方法，这种方法并不需要很多的器件，然而“加法-移位”的方法毕竟太慢。随着大规模集成电路的发展，采用高速的单元阵列乘法器，无论从计算机的计算速度，还是从提高计算效率，都是十分必要的。阵列乘法器分带符号和不带符号的阵列乘法器，本节只讨论不带符号阵列乘法。高速组合阵列乘法器，采用标准加法单元构成乘法器，即利用多个一位全加器（FA）实现乘法运算。

对于一个 4 位二进制数相乘，有如下算式：

×		A3	A2	A1	A0			
		B3	B2	B1	B0			
			A3B0	A2B0	A1B0	A0B0		
			A3B1	A2B1	A1B1	A0B1		
		A3B2	A2B2	A1B2	A0B2			
+	A3B3	A2B3	A1B3	A0B3				
	P7	P6	P5	P4	P3	P2	P1	P0

这个 4 × 4 阵列乘法器的原理如图 2-1 所示。

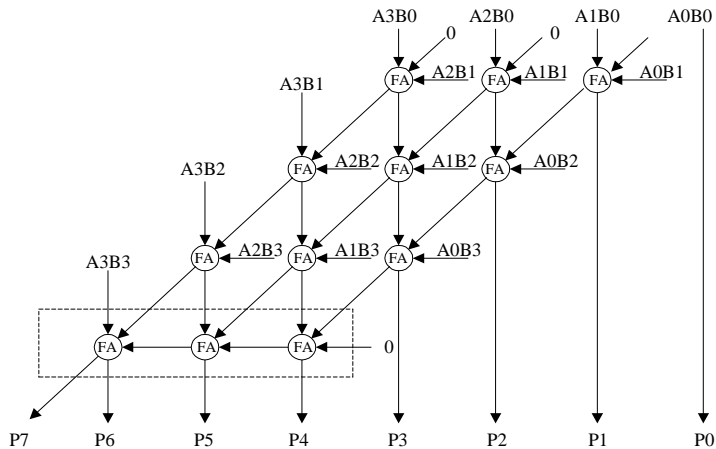


图 2-1 4×4 阵列乘法器原理图

FA（全加器）的斜线方向为进位输出，竖线方向为和输出。图中阵列的最后一行构成了一个串行进位加法器。由于 FA 一级是无需考虑进位的，它的进位被暂时保留下来不往前传递，因此同一极中任意一位 FA 加法器的进位输出与和输出几乎是同时形成的，与“串行移位”相比可

大大减少同级间的进位传递延迟，所以送往最后一行串行加法器的输入延迟仅与 FA 的级数（行数）有关，即与乘数位数有关。本实验用 CPLD 来设计一个 4×4 位加法器，且全部采用原理图方式实现。

本实验在 CPLD 单元上进行，CPLD 单元由由两大部分组成，一是 LED 显示灯，两组 16 只，供调试时观测数据，**LED 灯为正逻辑，1 时亮，0 时灭**。另外是一片 MAXII EPM1270T144 及其外围电路。

EPM1270T144 有 144 个引脚，分成四个块，即 BANK1...BANK4，将每个块的通用 I/O 脚加以编号，就形成 A01...A24、B01...B30 等 I/O 号，如图 2-2 所示。CPLD 单元排针的丝印分为两部分，一是 I/O 号，以 A、B、C、D 打头，如 A15，一是芯片引脚号，是纯数字，如 21，它们表示的是同一个引脚。在 Quartus II 软件中分配 I/O 时用的是引脚号，而在实验接线图中，都以 I/O 号来描述。



图 2-2 EMP1270 引脚分配图

EPM1270T144 共有 116 个 I/O 脚，本单元引出 110 个，其中 60 个以排针形式引出，供实验使用，其余 50 个以双列扩展插座形式给出，并标记为 JP，JP 座的 I/O 分配如图 2-3 所示。

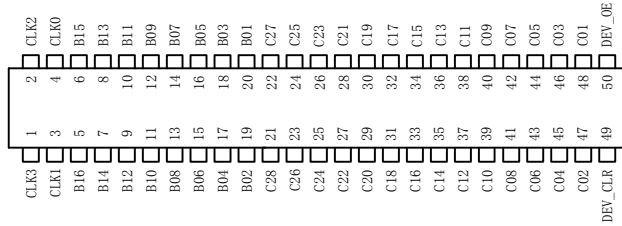


图 2-3 JP 座 I/O 分配图

## 2.4 实验步骤

(1) 根据上述阵列乘法器的原理,使用 Quartus II 软件编辑相应的电路原理图并进行编译,其在 EPM1270 芯片中对应的引脚如图 2-4 所示,框外文字表示 I/O 号,框内文字表示该引脚的含义(本实验例程见‘C:\TangDu\CMX\CPLD\Multiply.qpf’工程)。

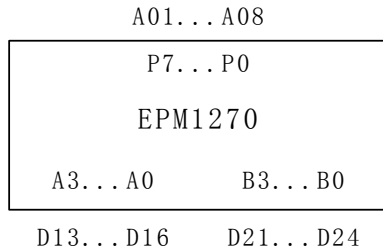


图 2-4 引脚分配图

- (2) 关闭实验系统电源,按图 2-5 连接实验电路,图中将用户需要连接的信号用圆圈标明。
- (3) 打开实验系统电源,将生成的 POF 文件下载到 EPM1270 中去。

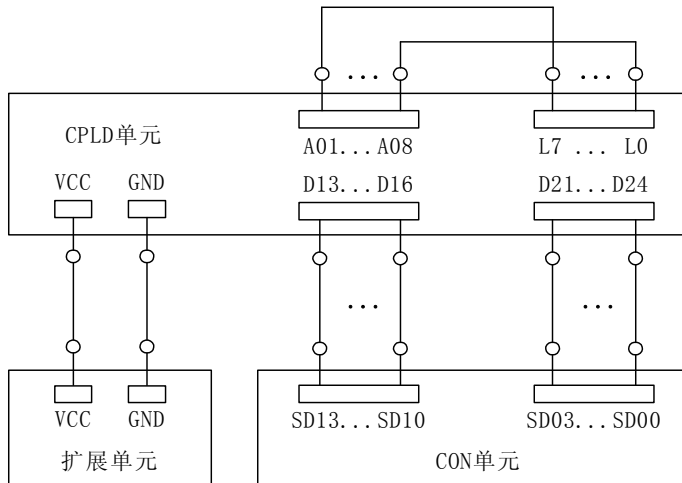


图 2-5 阵列乘法器实验接线图

(4) 以 CON 单元中的 SD13...SD10 四个二进制开关为乘数 A, SD03...SD00 四个二进制开关为被乘数 B, 而相乘的结果在 CPLD 单元的 L7...L0 八个 LED 灯显示。给 A 和 B 置不同的数, 观察相乘的结果。

## 实验三 静态随机存储器实验

存储器是计算机各种信息存储与交换的中心。在程序执行过程中，所要执行的指令是从存储器中获取，运算器所需要的操作数是通过程序中的访问存储器指令从存储器中得到，运算结果在程序执行完之前又必须全部写到存储器中，各种输入输出设备也直接与存储器交换数据。把程序和数据存储在存储器中，是冯·诺依曼型计算机的基本特征，也是计算机能够自动、连续快速工作的基础。

### 3.1 实验目的

掌握静态随机存储器 RAM 工作特性及数据的读写方法。

### 3.2 实验设备

PC 机一台，TD-CMX 实验系统一套。

### 3.3 实验原理

实验所用的静态存储器由一片 6116 (2K×8bit) 构成 (位于 MEM 单元)，如图 3-1 所示。6116 有三个控制线：CS# (片选线)、OE# (读线)、WE# (写线)，其功能如表 3-1 所示，当片选有效 (CS#=0) 时，OE#=0 时进行读操作，WE#=0 时进行写操作，本实验将 CS# 常接地。

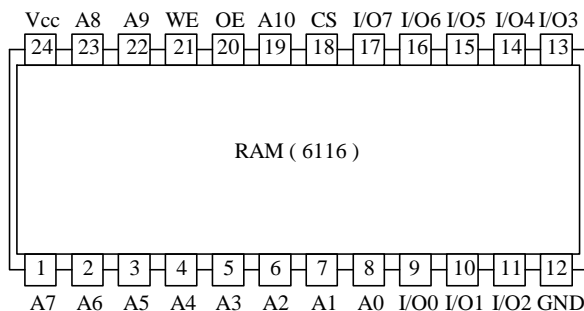


图 3-1 SRAM 6116 引脚图

表 3-1 SRAM 6116 功能表

$\overline{CS}$	$\overline{WE}$	$\overline{OE}$	功能
1	×	×	不选择
0	1	0	读
0	0	1	写
0	0	0	写

由于存储器 (MEM) 最终是要挂接到 CPU 上，所以其还需要一个读写控制逻辑，使得 CPU 能控制 MEM 的读写，实验中的读写控制逻辑如图 3-2 所示，由于 T3 的参与，可以保证 MEM 的写脉宽与 T3 一致，T3 由时序单元的 TS3 给出 (时序单元的介绍见附录)。IOM 用来选择是对 I/O 还是对 MEM 进行读写操作，**RD=1 时为读，WR=1 时为写**。

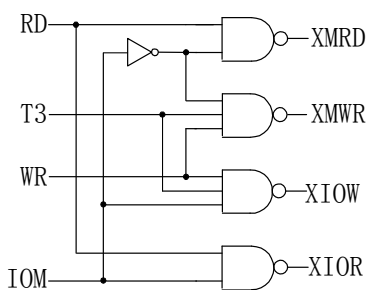


图 3-2 读写控制逻辑

表 3-2 读写控制逻辑功能表

输入				输出				
T3	WR	RD	IOM	XMRD	XMWR	XIOW	XIOR	说明
X	0	1	0	0	1	1	1	读 6116
1	1	0	0	1	0	1	1	写 6116
X	0	1	1	1	1	1	0	读 IO
1	1	0	1	1	1	0	1	写 IO

**从读写控制逻辑功能表可以看出：**

- 1) 写操作发生在 T3=1 期间，T3=0 是读操作；
- 2) 当 IOM=0 时，为存储器操作，当 IOM=1 时，为 IO 操作；
- 3) 输入是高电平有效，输出是低电平有效。

实验原理图如图 3-3 所示，存储器数据线接至数据总线，数据总线上接有 8 个 LED 灯显示 D7…D0 的内容。地址线接至地址总线，地址总线上接有 8 个 LED 灯显示 A7…A0 的内容，地址由地址锁存器(74LS273, 位于 PC&AR 单元)给出。数据开关(位于 IN 单元)经一个三态门(74LS245)连至数据总线，分时给出地址和数据。地址寄存器为 8 位，接入 6116 的地址 A7…A0，6116 的高三位地址 A10…A8 接地，所以其实际容量为 256 字节。

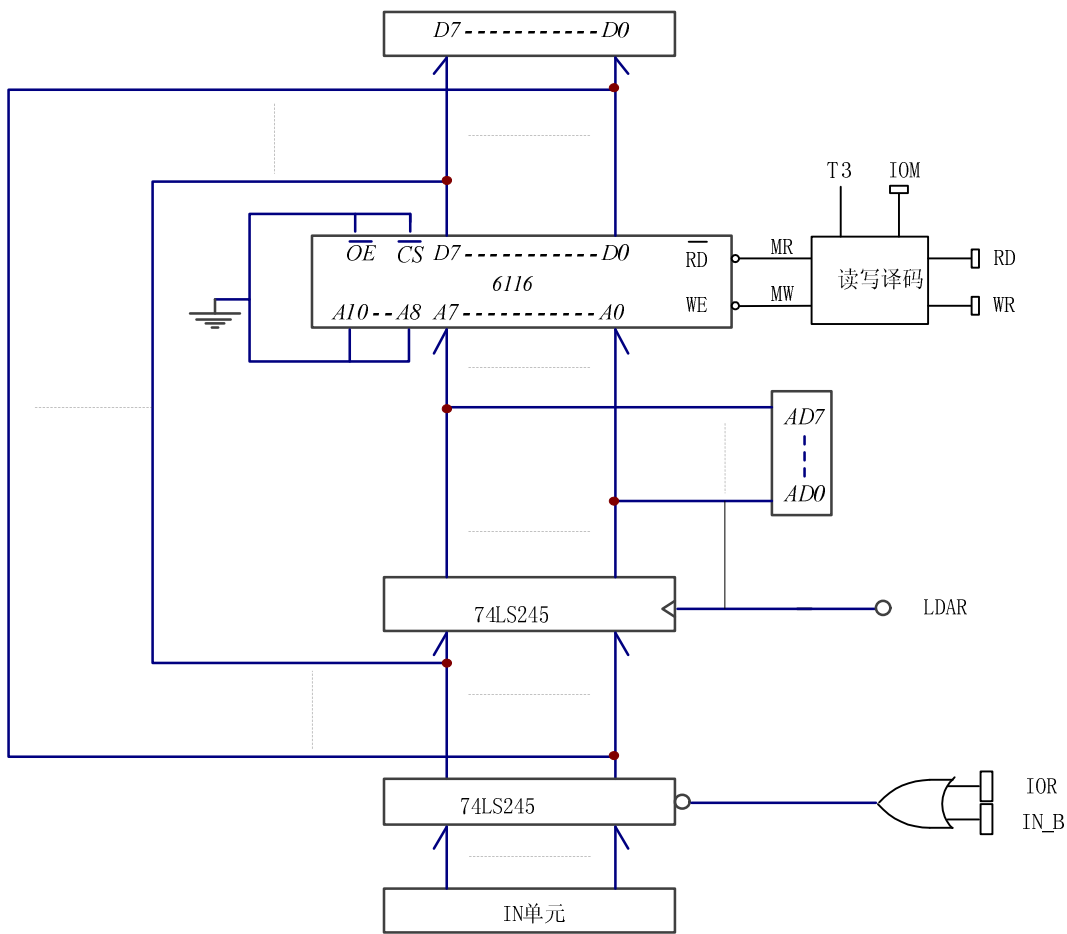


图 3-3 存储器实验原理图

实验箱中所有单元的时序都连接至时序与操作台单元，CLR 都连接至 CON 单元的 CLR 按钮。实验时 T3 由时序单元给出，其余信号由 CON 单元的二进制开关模拟给出，其中 IOM 应为低（即 MEM 操作），RD、WR 高有效，MR 和 MW 低有效，LDAR 高有效。

### 3.4 实验步骤

(1) 关闭实验系统电源，按图 3-4 连接实验电路，并检查无误，图中将用户需要连接的信号用圆圈标明。

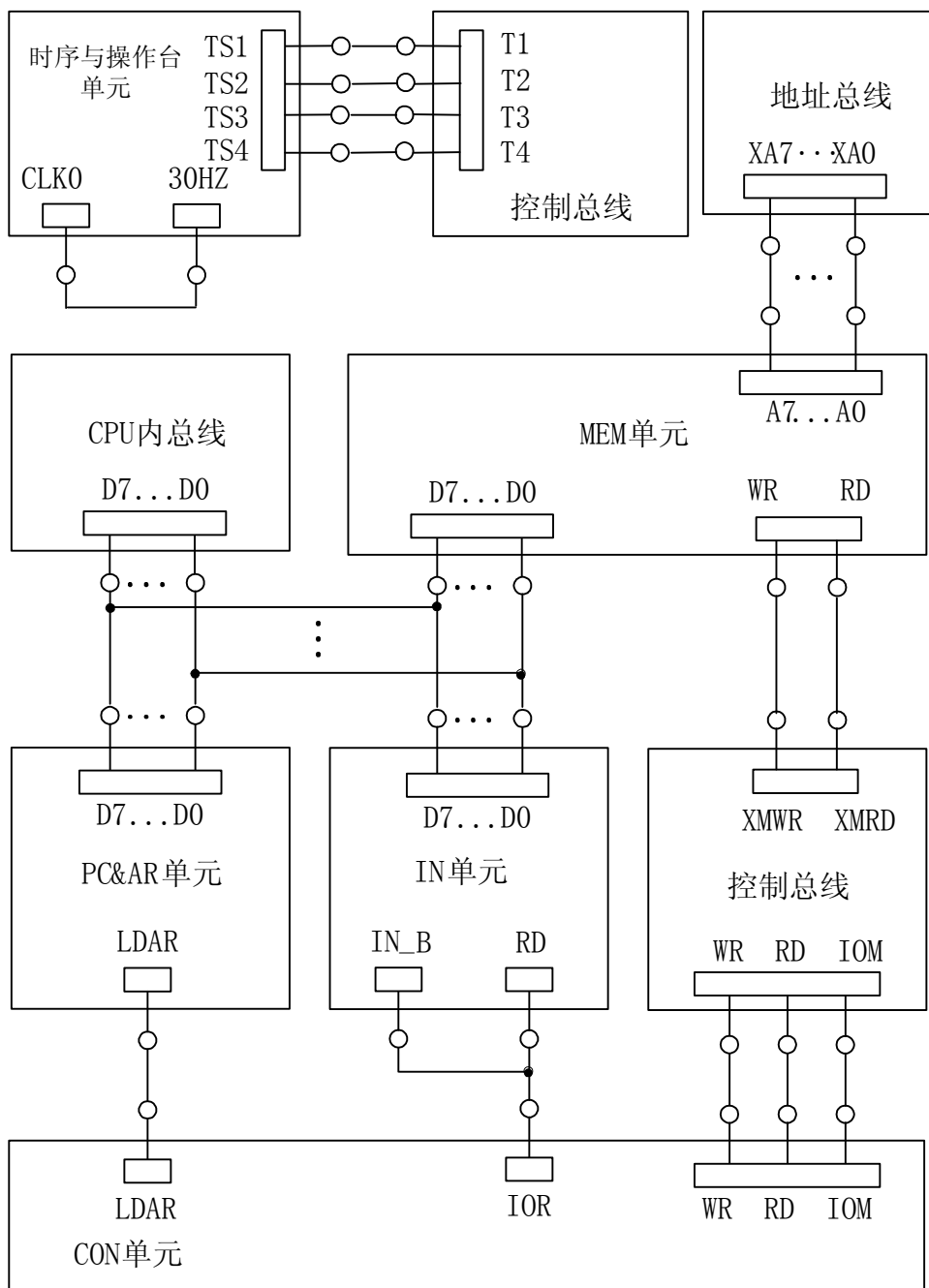


图 3-4 实验接线图

(2) 将时序与操作台单元的开关 **KK1、KK3 置为运行档、开关 KK2 置为‘单步’档**（时序单元的介绍见附录）。

(3) 将 CON 单元的 IOR 开关置为 1（使 IN 单元无输出），打开电源开关，如果听到有‘嘀’报警声，说明有总线竞争现象，应立即关闭电源，重新检查接线，直到错误排除。

(4) 给存储器的 00H、01H、02H、03H、04H 地址单元中分别写入数据 11H、12H、13H、14H、

15H。由前面的存储器实验原理图（图 3-3）可以看出，由于数据和地址由同一个数据开关给出，因此数据和地址要分时写入，先写地址，具体操作步骤为：先关掉存储器的读写（WR=0，RD=0），数据开关输出地址（IOR=0），然后打开地址寄存器门控信号（LDAR=1），**按动 ST 产生 T3 脉冲**，即将地址打入到 AR 中。再写数据，具体操作步骤为：先关掉存储器的读写（WR=0，RD=0）和地址寄存器门控信号（LDAR=0），数据开关输出要写入的数据，打开输入三态门（IOR=0），然后使存储器处于写状态（WR=1，RD=0，IOM=0），按动 ST 产生 T3 脉冲，即将数据打入到存储器中。写存储器的流程如图 3-5 所示（以向 00 地址单元写入 11H 为例）：

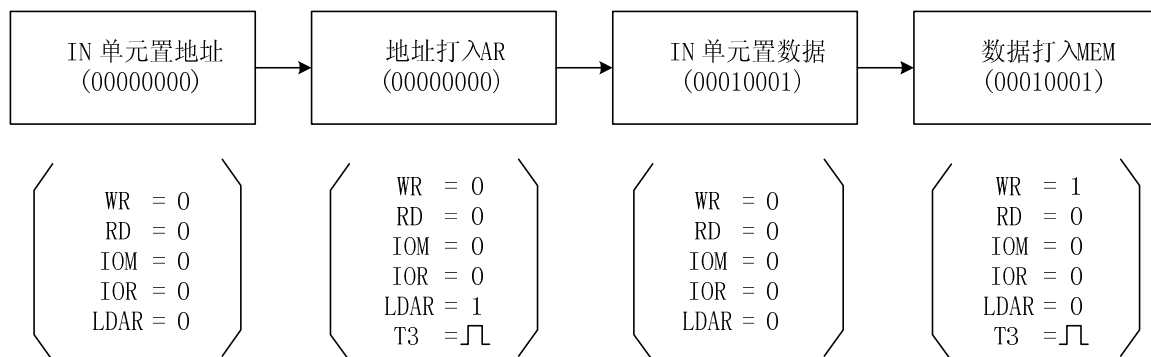


图 3-5 写存储器流程图

(5) 依次读出第 00、01、02、03、04 号单元中的内容，观察上述各单元中的内容是否与前面写入的一致。同写操作类似，也要先给出地址，然后进行读，地址的给出和前面一样，而在进行读操作时，应先关闭 IN 单元的输出（IOR=1），然后使存储器处于读状态（WR=0，RD=1，IOM=0），此时数据总线上的数即为从存储器当前地址中读出的数据内容。读存储器的流程如图 3-6 所示（以从 00 地址单元读出 11H 为例）：

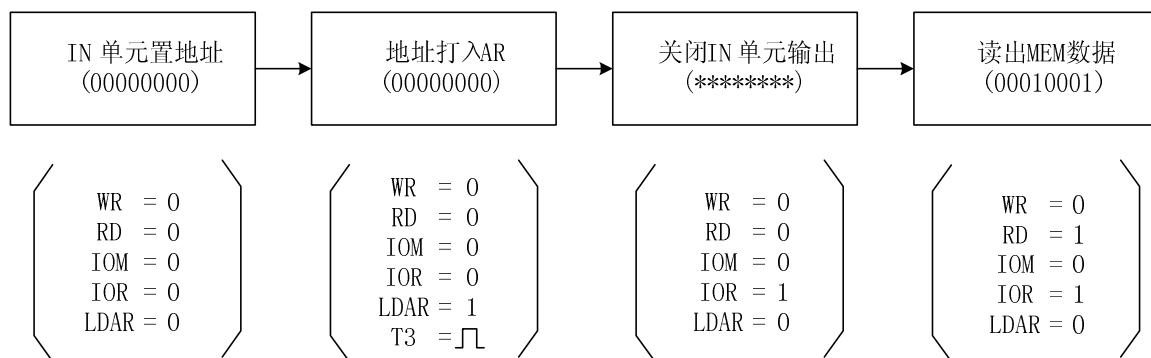


图 3-6 读存储器流程图

如果实验箱和 PC 联机操作，则可通过软件中的数据通路图来观测实验结果（软件使用说明请看附录），方法是：打开软件，选择联机软件的“【实验】—【存储器实验】”，打开存储器实验的数据通路图，如图 3-7 所示。



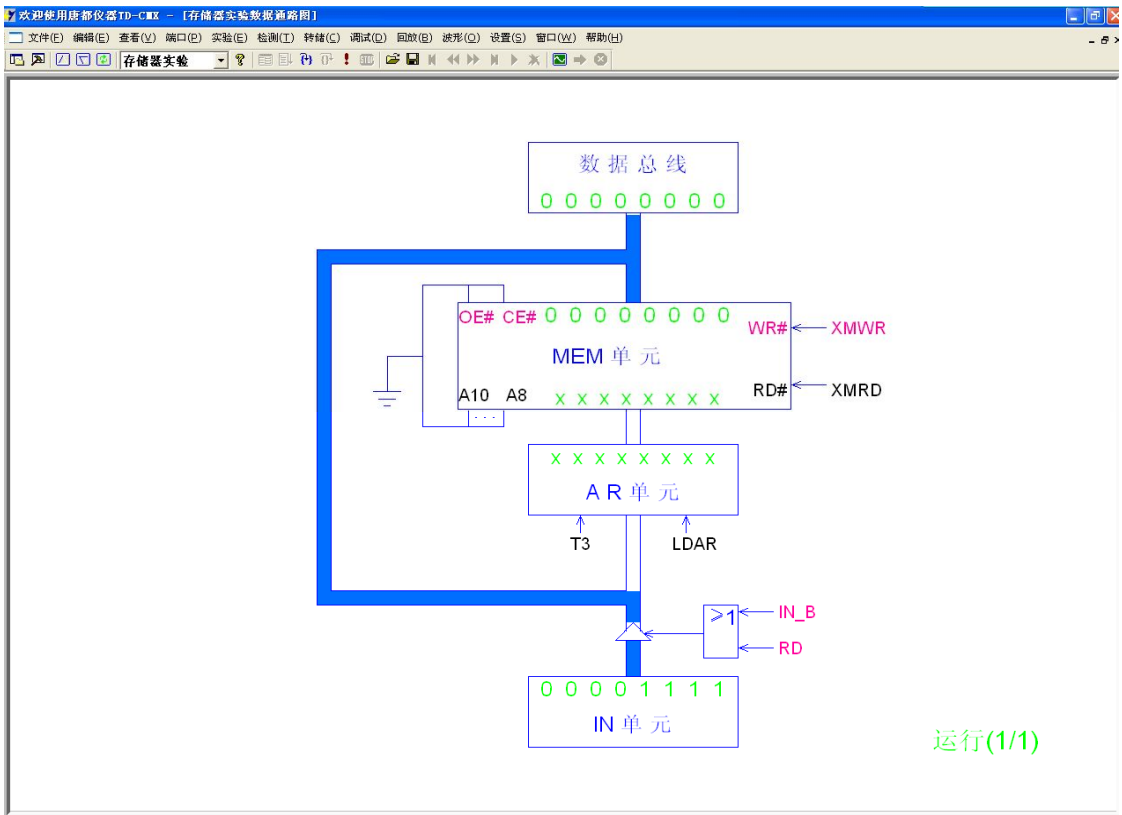


图 3-7 数据通路图

进行上面的手动操作，每按动一次 ST 按钮，数据通路图会有数据的流动，反映当前存储器所做的操作（即使是对存储器进行读，也应**按动一次 ST 按钮**，数据通路图才会有数据流动），或在软件中选择“【调试】—【单周期】”，其作用相当于将时序单元的状态开关置为‘单步’档后按动了一次 ST 按钮，数据通路图也会反映当前存储器所做的操作，借助于数据通路图，仔细分析 SRAM 的读写过程。

## 实验四 时序发生器设计实验

控制器是计算机的核心部件，计算机的所有硬件都是在控制器的控制下，完成程序规定的操作。控制器的基本功能就是把机器指令转换为按照一定时序控制机器各部件的工作信号，使各部件产生一系列动作，完成指令所规定的任务。

### 4.1 实验目的

- (1) 掌握时序发生器的原理及其设计方法。
- (2) 熟悉 CPLD 应用设计及 EDA 软件的使用。

### 4.2 实验设备

PC 机一台，TD-CMX 实验系统一套。

### 4.3 实验原理

计算机的工作是按照时序分步地执行。这就需要能产生周期节拍、脉冲等时序信号的部件，称为时序发生器。如图 4-1 所示。

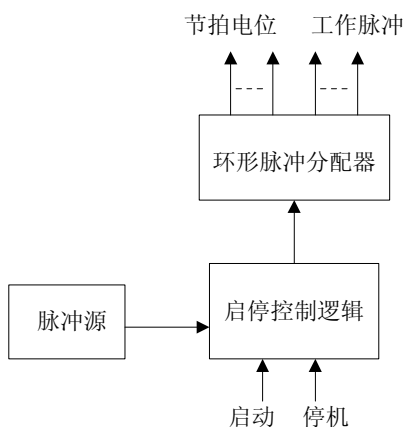


图 4-1 时序发生器

时序部件包括：

- (1) 脉冲源：又称主振荡器，为计算机提供基准时钟信号。
- (2) 脉冲分配器：对主频脉冲进行分频，产生节拍电位和脉冲信号。时钟脉冲经过脉冲发生器产生时标脉冲、节拍电位及周期状态电位。一个周期状态电位包含多个节拍电位，而一个节拍单位又包含多个时标脉冲。
- (3) 启停控制电路：用来控制主脉冲的启动和停止。

本实验是用 VHDL 语言来实现一个时序发生器，输出如图 4-2 所示 T1…T4 四个节拍信号。时序发生器需要一个脉冲源 ST，由时序单元的  $\phi$  提供（时序单元的介绍见附录），一个总清零 CLR，为低时，T1…T4 输出低。一个停机信号 STOP，当 T4 的下沿到来时，且 STOP 为低，T1…T4 输出低。一个启动信号 START，当 START、T1…T4 都为低，且 STOP 为高，T1…T4 输出环形脉冲。

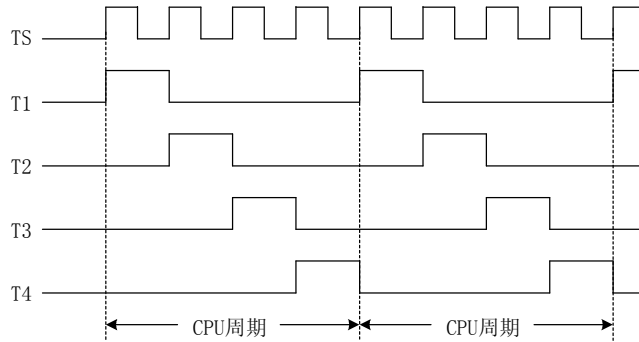


图 4-2 时序状态图

可通过 4 位循环移位寄存器来实现 T4…T1，CLR 为总清零信号，STOP 为低时在 T4 脉冲下沿清零时序，时序发生器启动后，移位寄存器在 ST 的上沿循环左移一位，移位寄存器的输出端即为 T4…T1。

#### 4.4 实验步骤

(1) 参照上面的实验原理，用 VHDL 语言来具体设计一个时序发生器。使用 Quartus II 软件编辑 VHDL 文件并进行编译，时序发生器在 EPM1270 芯片中对应的引脚如图 4-3 所示，**框外文字表示 I/O 号，框内文字表示该引脚的含义。**

本实验例程见 ‘[C:\TangDu\CMX\CPLD\Timer\Timer.qpf](#)’ 工程。

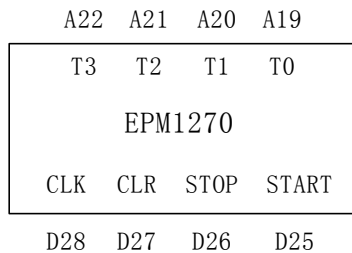


图 4-3 实验接线图

(2) 关闭实验系统电源，按图 4-4 连接实验电路，并检查无误。

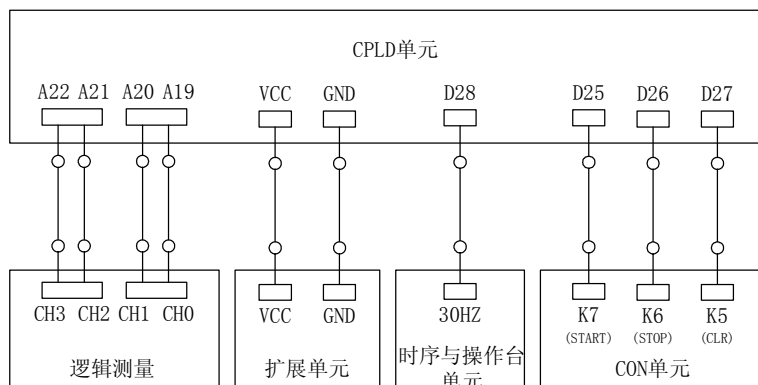


图 4-4 实验接线图

(3) 打开实验系统电源，将生成的 POF 文件下载到 EPM1270 中去。

(4) 将 CON 单元的 **K7 (START)**、**K6 (STOP)** 开关置 ‘1’，**K5 (CLR)** 开关置 ‘1-0-1’，使 T1…T4 输出低。运行联机软件，选择“【波形】—【打开】”打开逻辑示波器窗口，然后选择“【波形】—【运行】”启动逻辑示波器，逻辑示波器窗口显示 T1…T4 四路时序信号波形。

(5) 将 CON 单元的 K7 (START) 开关置 ‘1-0-1’，启动 T1…T4 时序，示波器窗口显示 T1…T4 波形，如图 4-5 所示。

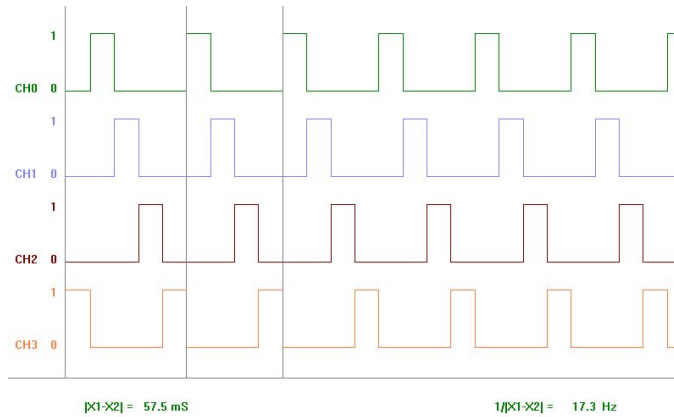


图 4-5 时序波形图

(6) 将 CON 单元的 K6 (STOP) 开关置 ‘0’，停止 T1…T4 时序，示波器窗口显示 T1…T4 波形均变为低。

## 实验五 微程序控制器实验

### 5.1 实验目的

- (1) 掌握微程序控制器的组成原理。
- (2) 掌握微程序的编制、写入，观察微程序的运行过程。

### 5.2 实验设备

PC 机一台，TD-CMX 实验系统一套。

### 5.3 实验原理

微程序控制器的基本任务是完成当前指令的翻译和执行，即将当前指令的功能转换成可以控制的硬件逻辑部件工作的微命令序列，完成数据传送和各种处理操作。它的执行方法就是将控制各部件动作的微命令的集合进行编码，即将微命令的集合仿照机器指令一样，用数字代码的形式表示，这种表示称为微指令。这样就可以用一个微指令序列表示一条机器指令，这种微指令序列称为微程序。微程序存储在一种专用的存储器中，称为控制存储器，微程序控制器原理框图如图 5-1 所示。

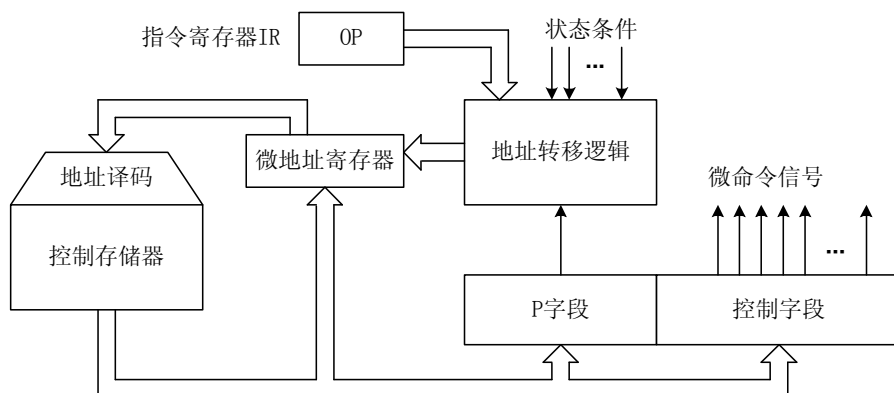


图 5-1 微程序控制器组成原理框图

控制器是严格按照系统时序来工作的，因而时序控制对于控制器的设计是非常重要的，从前面的实验可以很清楚地了解时序电路的工作原理，本实验所用的时序由时序单元来提供，分为四拍 TS1、TS2、TS3、TS4，时序单元的介绍见附录。

微程序控制器的组成见图 5-2，其中控制存储器采用 3 片 2816 的 E<sup>2</sup>PROM，具有掉电保护功能，微命令寄存器 18 位，用两片 8D 触发器（273）和一片 4D（175）触发器组成。微地址寄存器共 6 位，用三片正沿触发的双 D 触发器（74）组成，它们带有清“0”端和预置端。在不判别测试的情况下，T2 时刻打入微地址寄存器的内容即为下一条微指令地址。当 T4 时刻进行测试判别时，转移逻辑满足条件后输出的负脉冲通过强置端将某一触发器置为“1”状态，完成地址修改。

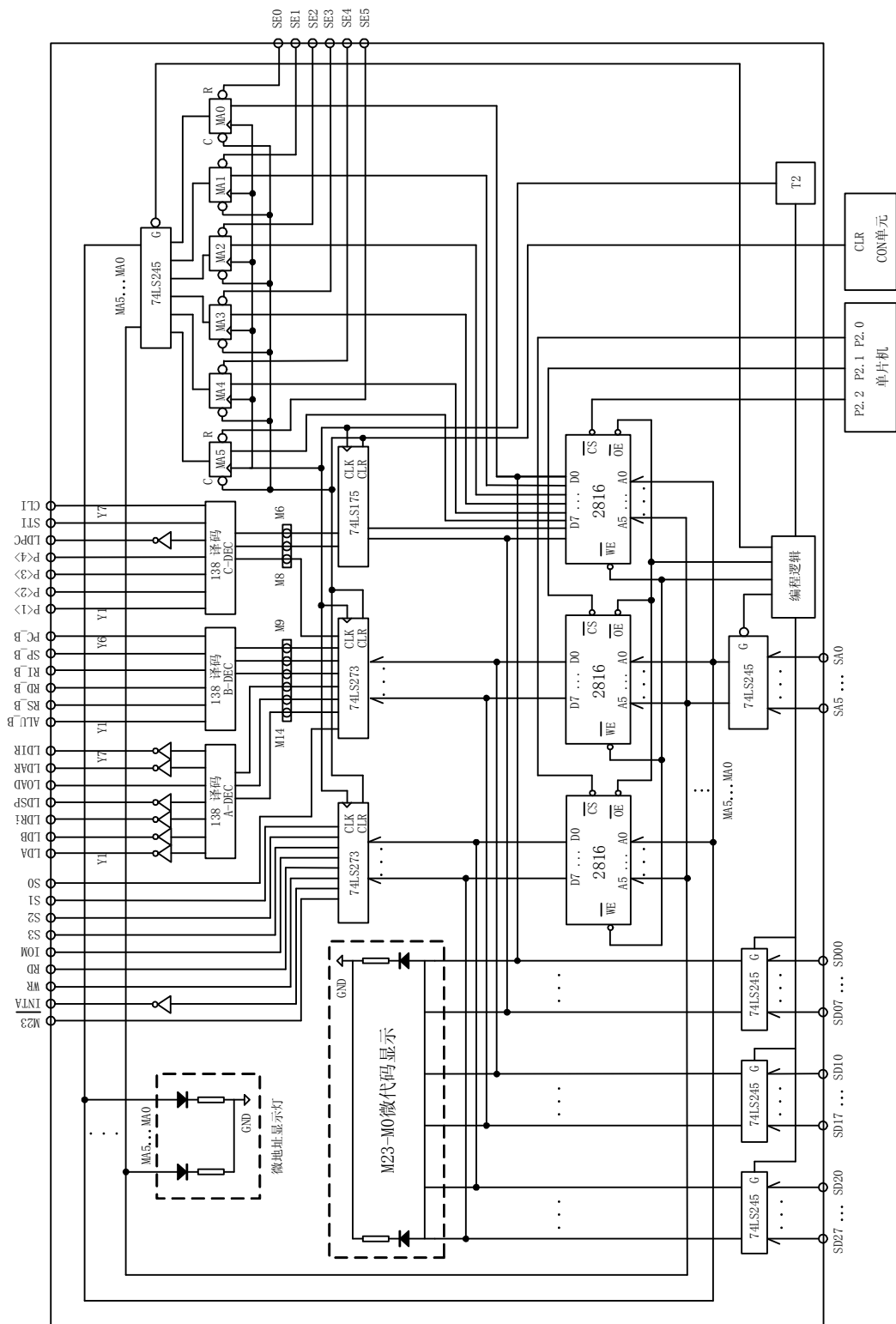


图 5-2 微程序控制器原理图

位于实验平台 MC 单元左上角一列三个指示灯 MC2、MC1、MC0 用来指示当前操作的微程序字  
段，分别对应 M23——M16、M15——M8、M7——M0。实验平台提供了比较灵活的手动操作方式，  
比如在上述操作中在对地址置数后将开关 KK4 拨至‘减 1’档，则每次操作的地址递减（不是地  
址每次减一），每次随着开关 ST 的两次拨动操作的字节数依次从高 8 位到低 8 位递减。减至低 8  
位后，再按动两次开关 ST，地址会自动减一，继续对下一个单元的操作。

微指令字长共 24 位，控制位顺序如表 5-1：

表 5-1 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 字段	B 字段	C 字段	MA5-MA0

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	RO_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	保留	1	0	1	保留	1	0	1	保留
1	1	0	保留	1	1	0	保留	1	1	0	保留
1	1	1	LDIR	1	1	1	保留	1	1	1	保留

其中 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。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，本系统上的**指令  
译码原理**如图 5-3 所示，图中 I7…I2 为指令寄存器的第 7…2 位输出，SE5…SE0 为微控器单元  
微地址锁存器的强置端输出，指令译码逻辑在 IR 单元的 INS\_DEC (GAL20V8) 中实现。

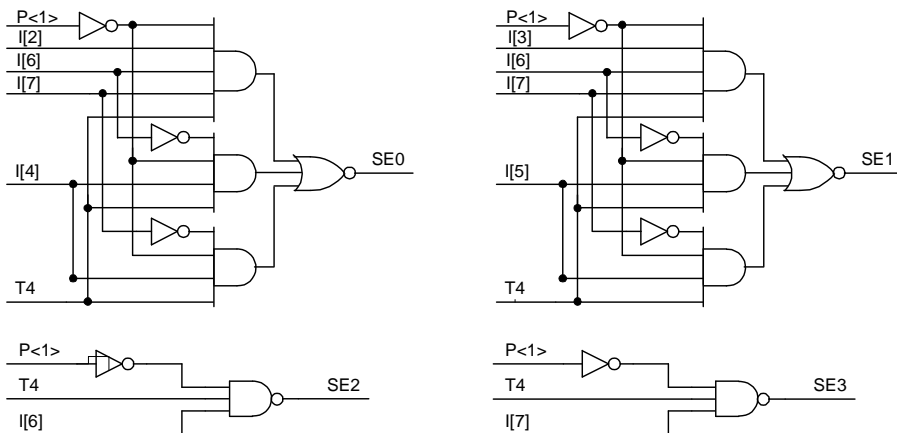


图 5-3 指令译码原理图

从图 5-2 中也可以看出，微控器产生的控制信号比表 5-1 中的要多，这是因为实验的不同，

所需的控制信号也不一样，本实验只用了部分的控制信号。

本实验除了用到指令寄存器（IR）和通用寄存器 R0 外，还要用到 IN 和 OUT 单元，从微控器出来的信号中只有 IOM、WR 和 RD 三个信号，所以对这两个单元的读写信号还应先经过译码，其译码原理如图 5-4 所示。IR 单元的原理图如图 5-5 所示，R0 单元原理如图 5-7 所示，IN 单元的原理图见图 3-3 所示，OUT 单元的原理图见图 5-6 所示。

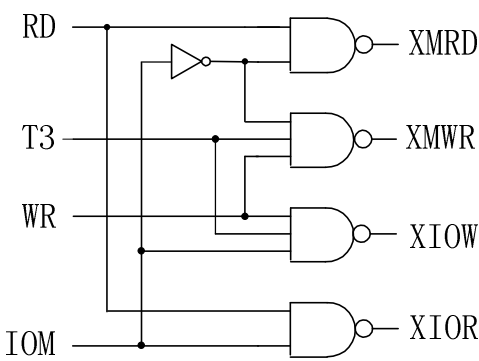


图 5-4 读写控制逻辑

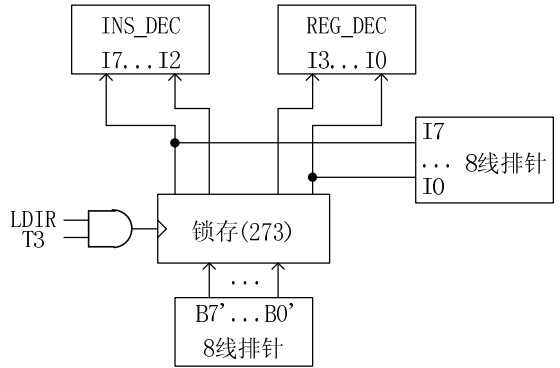


图 5-5 IR 单元原理图

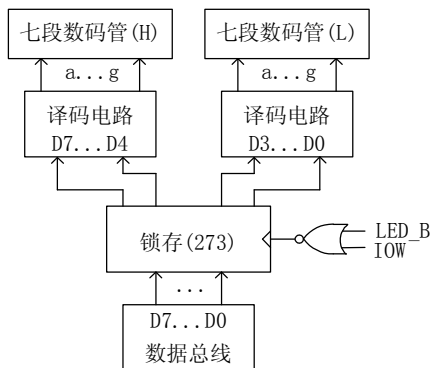


图 5-6 OUT 单元原理图

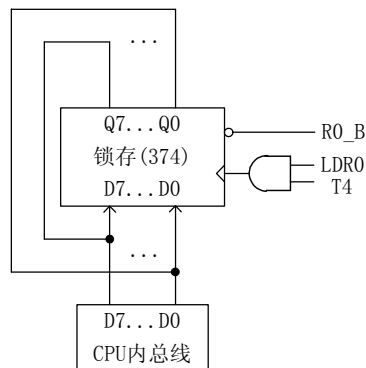


图 5-7 R0 原理图

本实验安排了四条机器指令，分别为 ADD（0000 0000）、IN（0010 0000）、OUT（0011 0000）和 HLT（0101 0000），括号中为各指令的二进制代码，指令格式如下：

助记符	机器指令码	说明
IN	0010 0000	IN → R0
ADD	0000 0000	R0 + R0 → R0
OUT	0011 0000	R0 → OUT
HLT	0101 0000	停机

实验中机器指令由 CON 单元的二进制开关手动给出，其余单元的控制信号均由微程序控制器自动产生，为此可以设计出相应的数据通路图，见图 5-8 所示。

几条机器指令对应的参考微程序流程图如图 5-9 所示。图中一个矩形方框表示一条微指令，**方框中的内容**为该指令执行的微操作，**左上角的数字**是该条指令的微地址，**右下角的数字**是该条指令的后续微地址，所有微地址均用 16 进制表示。向下的箭头指出了下一条要执行的指令。P<1>为测试字，根据条件使微程序产生分支。



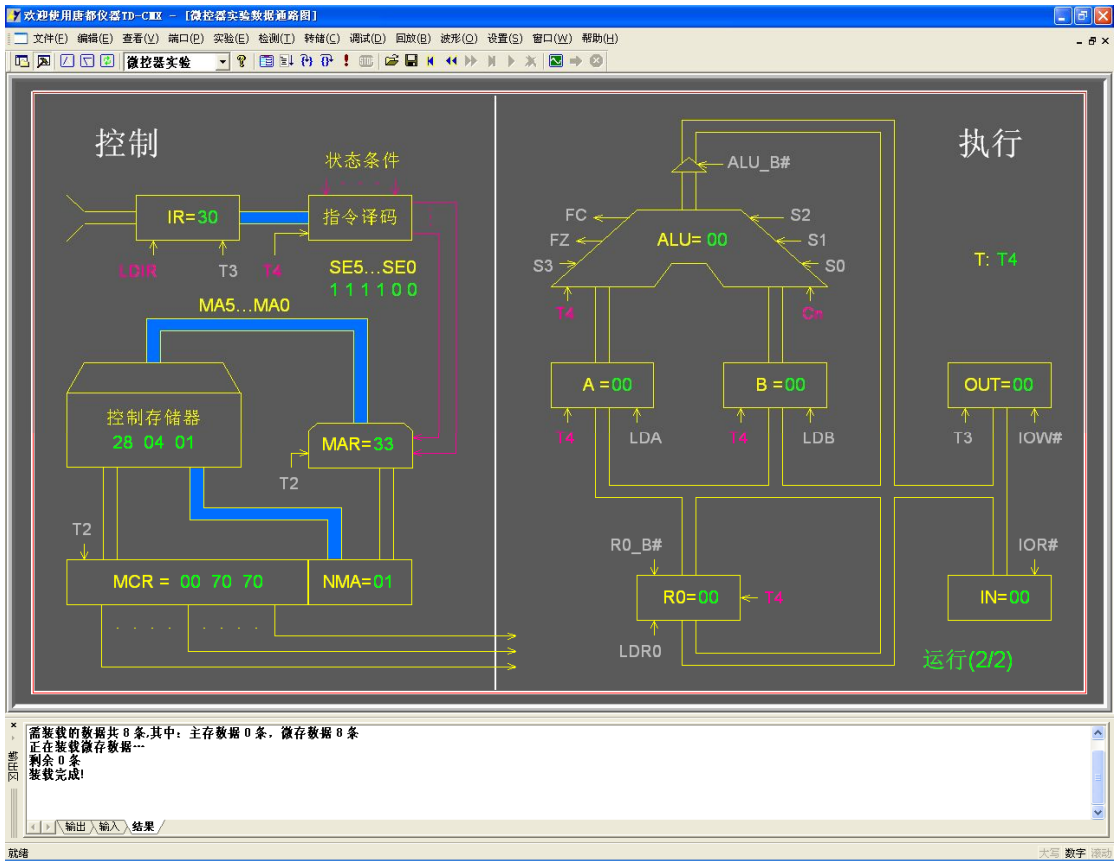


图 5-8 数据通路图

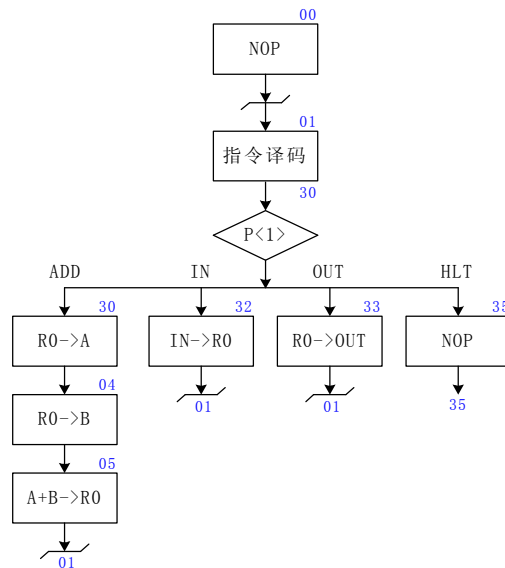


图 5-9 微程序流程图

将全部微程序按微指令格式变成二进制微代码，可得到表 5-2 的二进制代码表。

表 5-2 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 70 70	00000	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
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

#### 5.4 实验步骤

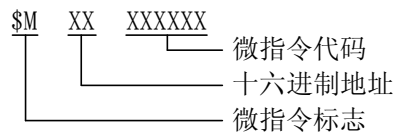
1. 按图 5-10 所示连接实验线路，仔细查线无误后接通电源。如果有‘滴’报警声，说明总线有竞争现象，应关闭电源，检查接线，直到错误排除。

2. 对微控器进行读写操作

(1) 将微程序写入文件

联机软件提供了微程序下载功能，以代替手动读写微控器，但微程序得以指定的格式写入到以 TXT 为后缀的文件中，微程序的格式如下：

微指令格式说明：



如\$M 1F 112233，表示微指令的地址为 1FH，微指令值为 11H（高）、22H（中）、33H（低），本次实验的微程序如下，其中分号‘；’为注释符，分号后面的内容在下载时将被忽略掉。

```

$M 00 000001 ; NOP
$M 01 007070 ; CON(INS)->IR, P<1>
$M 04 002405 ; R0->B
$M 05 04B201 ; A 加 B->R0
$M 30 001404 ; R0->A
$M 32 183001 ; IN->R0
$M 33 280401 ; R0->OUT
$M 35 000035 ; NOP
    
```

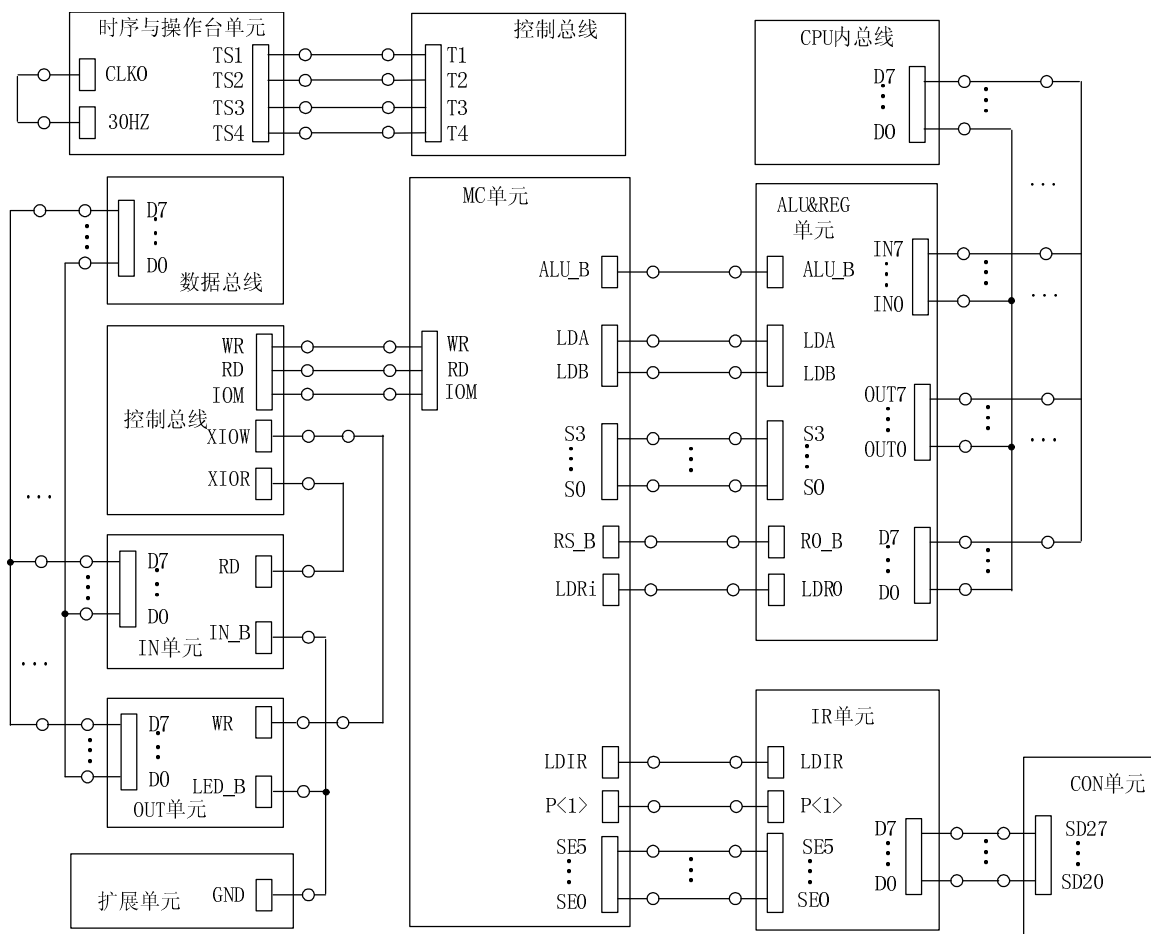


图 5-10 实验接线图

## (2) 写入微程序

用联机软件的“【转储】—【装载数据】”功能将该格式 (\*.txt) 文件装载入实验系统。安装路径 C:\TangDu\CMX\Sample\微程序控制器实验.txt。装入过程中，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。

## 3. 联机运行微程序

联机运行时，进入软件界面，在菜单上选择【实验】—【微控器实验】，打开本实验的数据通路图，也可以通过工具栏上的下拉框打开数据通路图，数据通路图如图 5-8 所示。

将时序与操作台单元的开关 KK1、KK3 置为‘运行’档，按动 CON 单元的总清开关后，按动软件中单节拍按钮，当后续微地址（通路图中的 MAR）为 000001 时，置 CON 单元 SD07…SD00，手动模拟产生相应的机器指令，该指令将会在下个 T3 打入指令寄存器（IR），在后面的节拍中将执行这条机器指令。仔细观察每条机器指令的执行过程，体会后续微地址被强制置转换的过程，这是计算机识别和执行指令的根基。也可以打开微程序流程图，跟踪显示每条机器指令的执行过程。

按本机运行的顺序给出数据和指令，观查最后的运算结果是否正确。

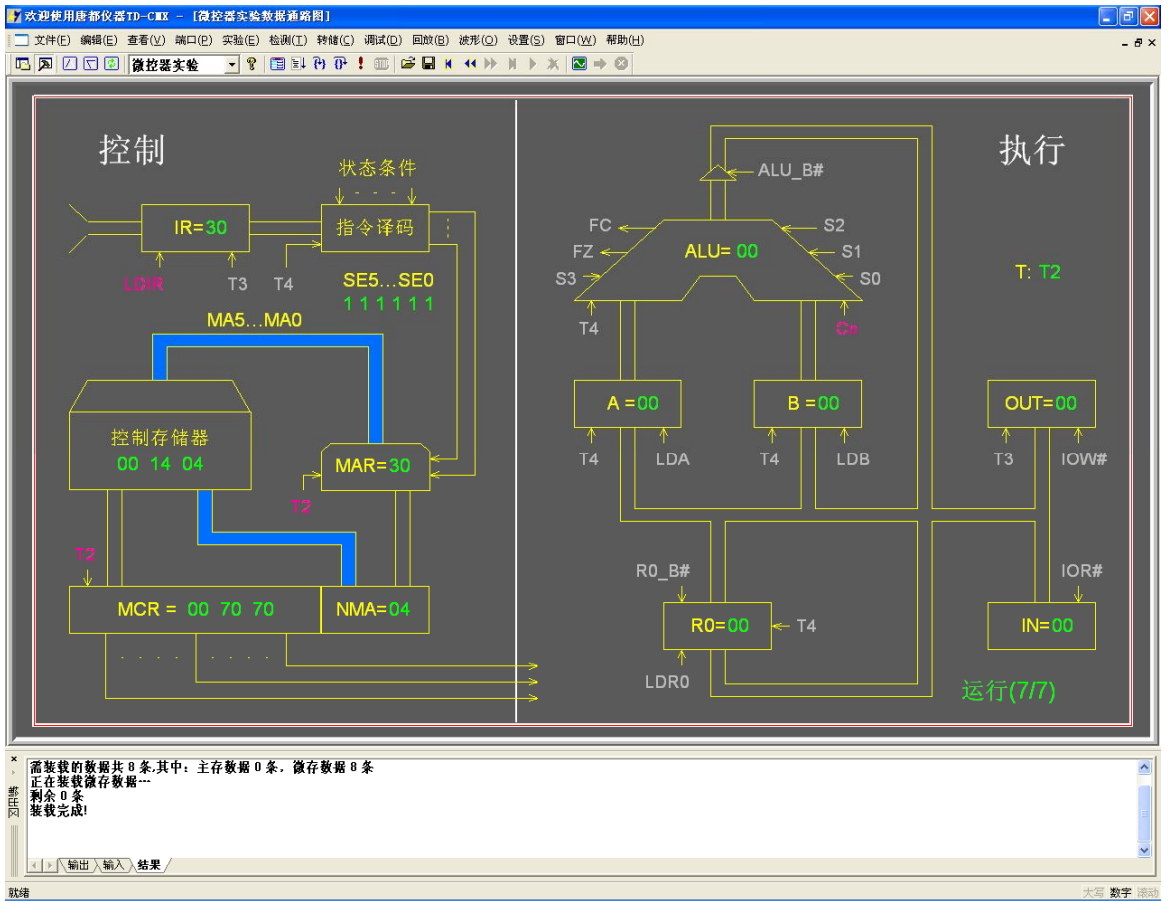


图 5-11 实验运行截图

图中 **MAR** 微地址寄存器; **NMA** 再下一条微地址寄存器; **MCR** 当前微指令寄存器。当前的状态为: MCR 当前微指令内容 007070H, 其低 6 位 30H 是下址字段, 存放在微地址寄存器 MAR 中, 而 30H 地址中的微指令内容在控制存储器方框中显示, 其内容为 001404H, 下址字段低 6 位在 NMA 中为 04H, 这是再下一条将执行的微指令地址。目前正在执行的微指令在 MCR 中。

## 实验六 简单模型机设计实验

在前面的章节中，我们重点讨论计算机中每个部件的组成及特性，下面我们将重点讨论如何完整设计一台模型计算机，进一步建立整机的概念。

### 6.1 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的基础上，进一步将其构造一台基本模型计算机。
- (3) 为其定义五条机器指令，编写相应的微程序，并上机调试掌握整机概念。

### 6.2 实验设备

PC 机一台，TD-CMX 实验系统一套。

### 6.3 实验原理

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。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 6-1 所示。这个 CPU 在写入相应的微指令后，就具备了执行机器指令的功能，但是机器指令一般存放在主存当中，CPU 必须和主存挂接后，才有实际的意义，所以还需要在该 CPU 的基础上增加一个主存和基本的输入输出部件，以构成一个简单的模型计算机。

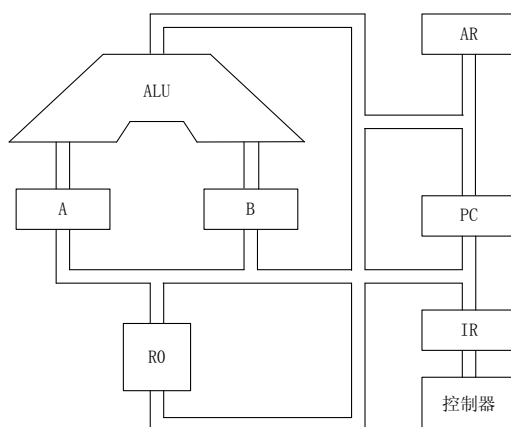


图 6-1 基本 CPU 构成原理图

除了程序计数器（PC），其余部件在前面的实验中都已用到，在此不再讨论。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。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 为低时，计数时钟到来后将 CPU 内总线上的数据打入 PC。

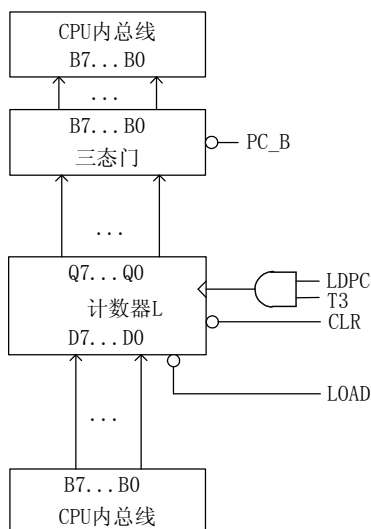


图 6-2 程序计数器(PC)原理图

本模型机和前面微程序控制器实验相比，新增加一条跳转指令 JMP，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），其指令格式如下（高 4 位为操作码）：

助记符	机器指令码	说明
IN	0010 0000	IN → R0
ADD	0000 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。微程序控制器实验的指令是通过手动给出的，现在要求 CPU 自动从存储器读取指令并执行。根据以上要求，设计数据通路图，如图 6-3 所示。

本实验在前一个实验的基础上**增加了三个部件**，一是 PC（程序计数器），另一个是 AR（地址寄存器），还有就是 MEM（主存）。因而在微指令中应增加相应的控制位，其微指令格式如表 6-1 所示。

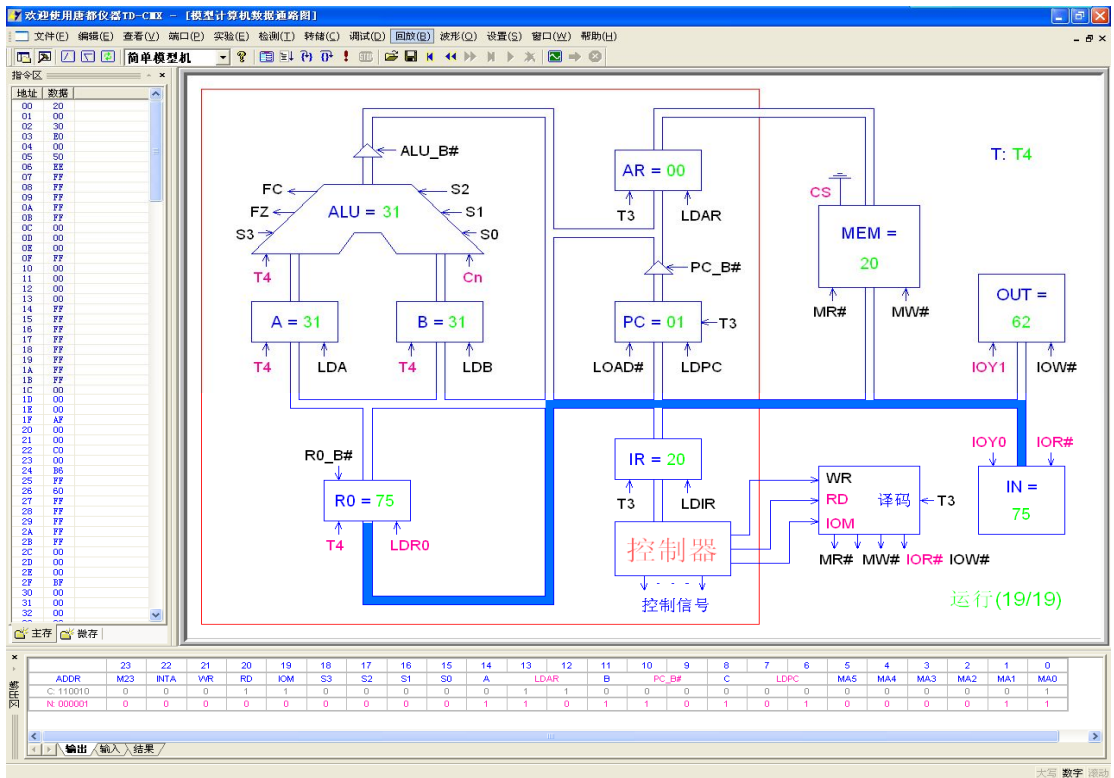


图 6-3 数据通路图

表 6-1 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 字段	B 字段	C 字段	MA5-MA0

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	LDRO	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

系统涉及到的微程序流程见图 6-4 所示，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。指令译码原理见图 5-3 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**，微程序流程图上的单元地址为 16 进制。

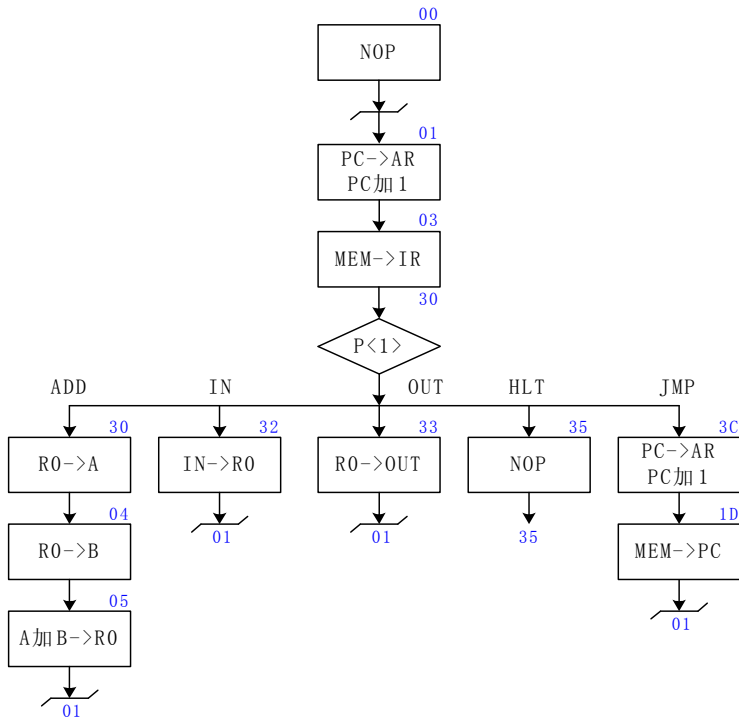


图 6-4 简单模型机微程序流程图

当全部微程序设计完毕后，应将每条微指令代码化，表 6-2 即为将图 6-4 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 6-2 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

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，地址和内容均为十六进制数。

地址	内容	助记符	说明
00	20	; START: IN R0	从 IN 单元读入数据送 R0
01	00	; ADD R0,R0	R0 和自身相加，结果送 R0
02	30	; OUT R0	R0 的值送 OUT 单元显示
03	E0	; JMP START	跳转至 00H 地址
04	00	;	
05	50	; HLT	停机

## 6.4 实验步骤

1. 按图 6-5 连接实验线路。

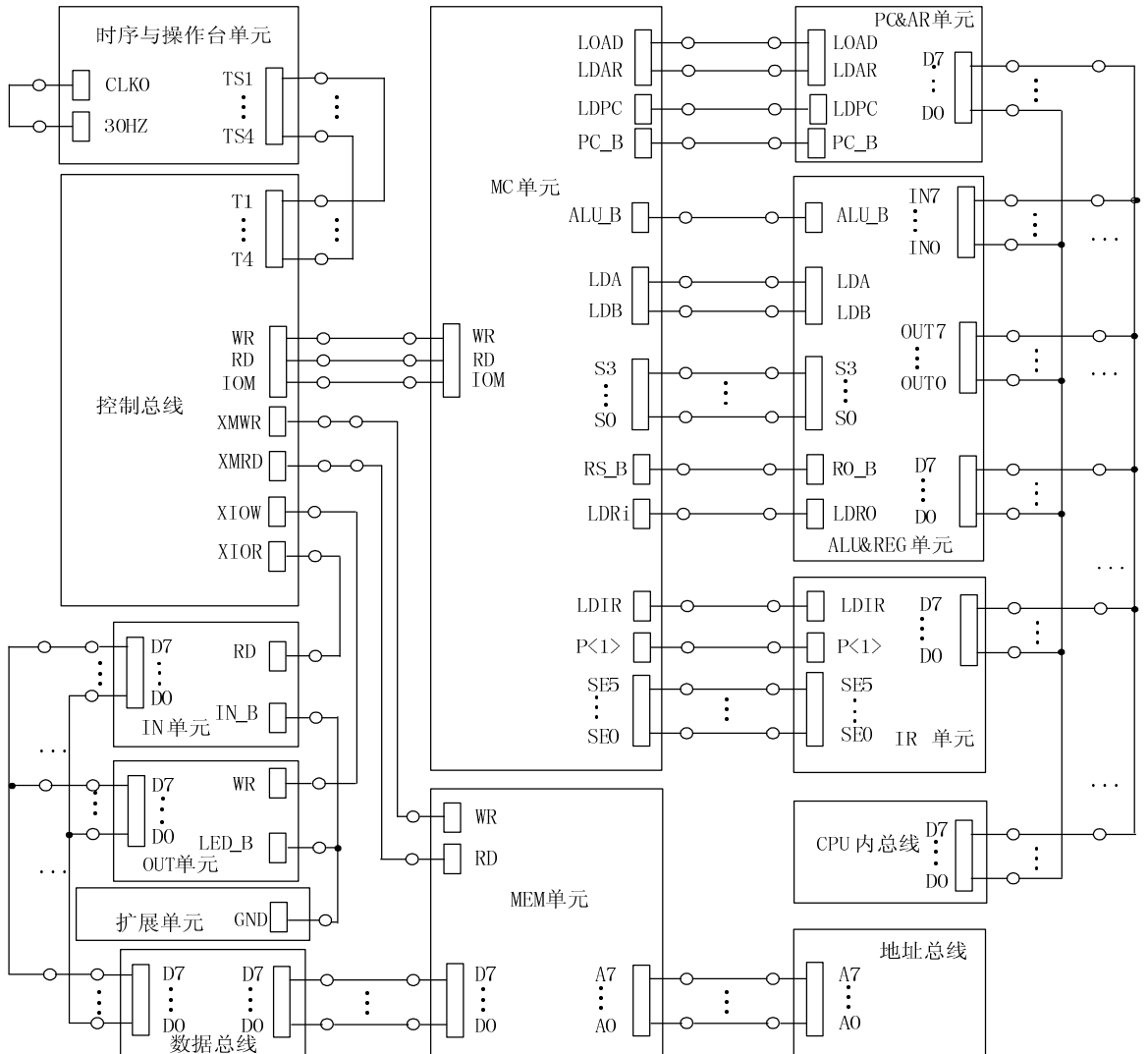
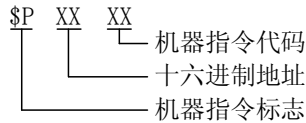


图 6-5 实验接线图

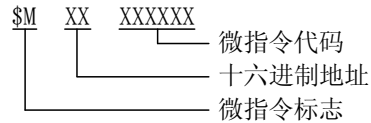
2. 联机写入实验程序，并进行校验，

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，微程序和机器程序的格式如下：

机器指令格式说明：



微指令格式说明：



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 与简单模型机设计实验机器指令和微程序代码如下：

； **机器指令**

```
$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
```

； **微程序**

```
$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
```

选择联机软件的“【转储】—【装载】”功能，在 C:\TangDu\CMX\Sample 路径下打开文件对话框中选择“CPU 与简单模型机设计实验.txt”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，如果机器程序有错误，重点检查与存储器有关的连线。模型机能正常运行的前提是微程序和机器指令必须正确。**转储下载正确，表明硬件连线没有问题**，否则必须重新连线。

### 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**，进入软件界面，选择菜单命令“【实验】—【简单模型机】”，打开简单模型机数据通路图。

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 IN 单元值的 2 倍。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。

## 实验七 硬布线控制器模型机设计实验

### 7.1 实验目的

- (1) 掌握硬布线控制器的组成原理、设计方法。
- (2) 了解硬布线控制器和微程序控制器的各自优缺点。

### 7.2 实验设备

PC 机一台，TD-CMX 实验系统一套。

### 7.3 实验原理

硬布线控制器本质上是一种由门电路和触发器构成的复杂树形网络，它将输入逻辑信号转换成一组输出逻辑信号，即控制信号。硬布线控制器的输入信号有：指令寄存器的输出、时序信号和运算结果标志状态信号等，输出的就是所有各部件需要的各种微操作信号。

硬布线控制器的设计思想是：在硬布线控制器中，操作控制器发出的各种控制信号是时间因素和空间因素的函数。各个操作定时的控制构成了操作控制信号的时间特征，而各种不同部件的操作所需要的不同操作信号则构成了操作控制信号的空间特征。硬布线控制器就是把时间信号和操作信号组合，产生具有定时特点的控制信号。

简单模型机的控制器是微程序控制器，本实验中的模型机将用硬布线取代微程序控制器，其余部件和简单模型机的一样，所以其数据通路图也和简单模型机的一样，见图 6-3，机器指令也和简单模型机的机器指令一样，如下所示。

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

根据指令要求，得出用时钟进行驱动的状态机描述，即得出其有限状态机，如图 7-1 所示。

下面分析每个状态中的基本操作：

S0：空操作，系统复位后的状态

S1：PC→AR, PC+1

S2：MEM→BUS, BUS→IR

S3：R0→BUS, BUS→A

S4：R0→BUS, BUS→B

S5：A 加 B→BUS, BUS→R0

S6：IN→BUS, BUS→R0

S7：R0→BUS, BUS→OUT

S8：空操作

S9：PC→AR, PC+1

S10：MEM→BUS, BUS→PC

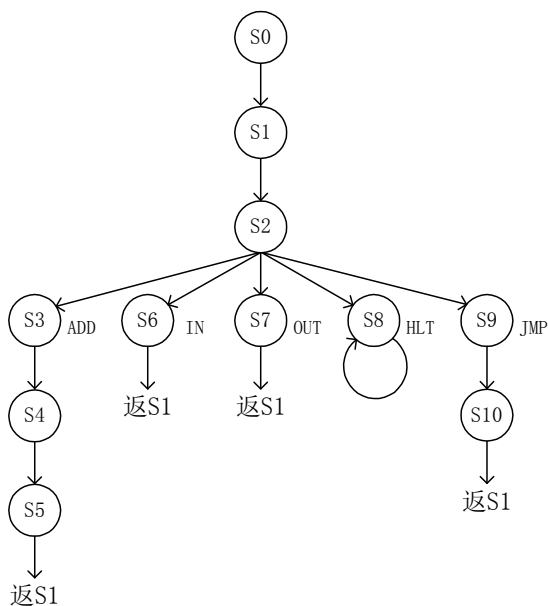


图 7-1 状态机描述

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。

### 7.4 实验步骤

(1) 分析每个状态所需的控制信号，并汇总成表，如表 7-1 所示。

表 7-1 控制信号表

状态号	控制信号
S0	0 0 0 0 0 0 0 0 0 1 0 0 1 1 0 1 0
S1	0 0 0 0 0 0 0 0 0 1 1 0 1 1 0 0 1
S2	0 1 0 0 0 0 0 0 0 1 0 1 1 1 0 1 0
S3	0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 1 0
S4	0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 1 0
S5	0 0 0 1 0 0 1 0 0 1 0 0 0 1 1 1 0
S6	0 1 1 0 0 0 0 0 0 1 0 0 1 1 1 1 0
S7	1 0 1 0 0 0 0 0 0 1 0 0 1 0 0 1 0
S8	0 0 0 0 0 0 0 0 0 1 0 0 1 1 0 1 0
S9	0 0 0 0 0 0 0 0 0 1 1 0 1 1 0 0 1
S10	0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1

**控制信号由左至右**，依次为：WR, RD, IOM, S3, S2, S1, S0, LDA, LDB, LOAD, LDAR, LDIR, ALU\_B, R0\_B, LDRO, PC\_B, LDPC。

(2) 用 VHDL 语言来设计本实验的状态机，使用 Quartus II 软件编辑 VHDL 文件并进行编译，

硬布线控制器在 EPM1270 芯片中对应的引脚如图 7-2 所示（本实验例程见‘[C:\TangDu\CMX\CPLD\Controller\Controller.qpf](#)’工程）。

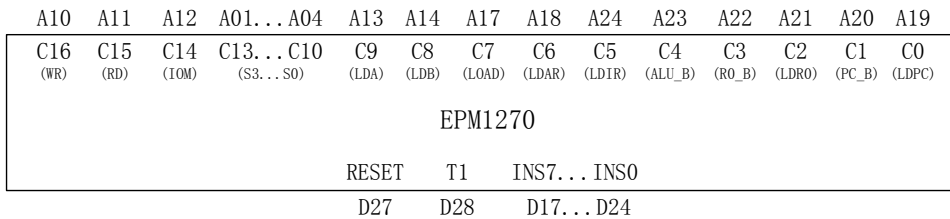


图 7-2 引脚分配图

(3) 关闭实验系统电源，按图 7-3 连接实验电路。

(4) 打开实验系统电源，将生成的 POF 文件下载到 CPLD 单元的 EPM1270 中去。

(5) 用本实验定义的机器指令系统，可具体编写多种应用程序，下面给出的是本次实验的例程，其程序的文件名以 .TXT 为后缀。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：

```

$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 START
$P 04 00 ;
$P 05 50 ; HLT           停机
    
```

(6) 进入软件界面，装载机器指令，选择菜单命令“【实验】—【简单模型机】”，打开简单模型机数据通路图，按动 CON 单元的总清按钮 CLR，使程序计数器 PC 地址清零，控制器状态机回到 S0，程序从头开始运行，选择相应的功能命令，即可联机运行、监控、调试程序。

(7) 当模型机执行完 JMP 指令后，检查 OUT 单元显示的数是否为 IN 单元值的 2 倍，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。

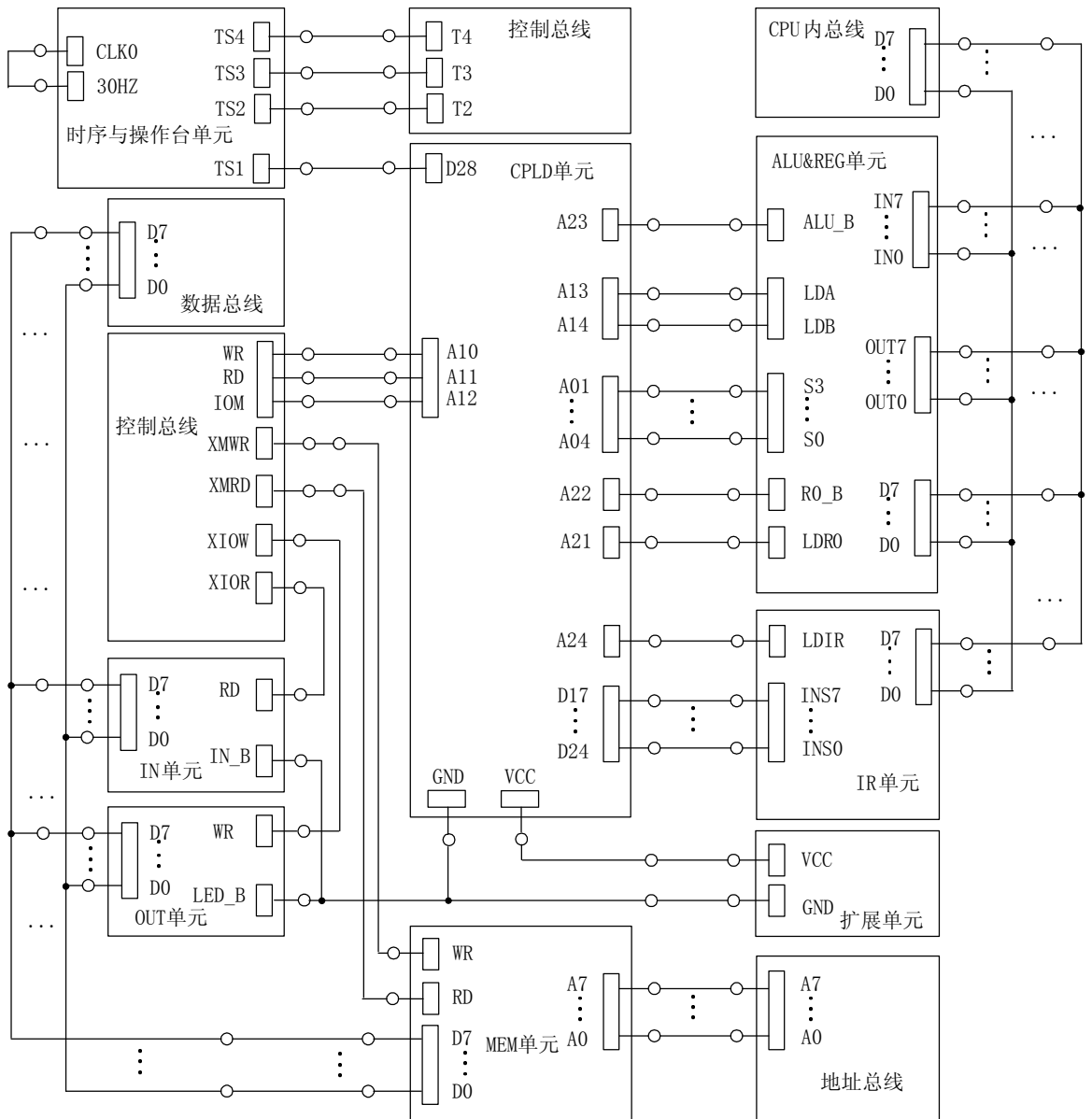


图 7-3 实验接线图

## 实验八 复杂模型机设计实验

### 8.1 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。

### 8.2 实验设备

PC 机一台，TD-CMX 实验系统一套。

### 8.3 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： $0 \leq X \leq 2^8 - 1$ 。

#### 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 8-1（在地址总线单元）。

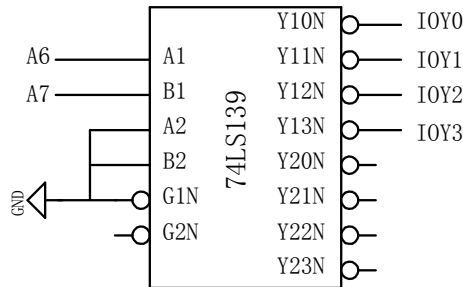


图 8-1 I/O 地址译码原理图

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 8-1 所示：

表 8-1 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 8-2，以 R2 做为变址寄存器 RI。

表 8-2 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 8-3 列出了各条指令的格式、汇编符号、指令功能。

表 8-3 指令描述

助记符号	指令格式	指令功能				
MOV RD, RS	<table border="1"><tr><td>0100</td><td>RS</td><td>RD</td></tr></table>	0100	RS	RD	RS → RD	
0100	RS	RD				
ADD RD, RS	<table border="1"><tr><td>0000</td><td>RS</td><td>RD</td></tr></table>	0000	RS	RD	RD + RS → RD	
0000	RS	RD				
SUB RD, RS	<table border="1"><tr><td>1000</td><td>RS</td><td>RD</td></tr></table>	1000	RS	RD	RD - RS → RD	
1000	RS	RD				
AND RD, RS	<table border="1"><tr><td>0001</td><td>RS</td><td>RD</td></tr></table>	0001	RS	RD	RD ∧ RS → RD	
0001	RS	RD				
OR RD, RS	<table border="1"><tr><td>1001</td><td>RS</td><td>RD</td></tr></table>	1001	RS	RD	RD ∨ RS → RD	
1001	RS	RD				
RR RD, RS	<table border="1"><tr><td>1010</td><td>RS</td><td>RD</td></tr></table>	1010	RS	RD	RS 右环移 → RD	
1010	RS	RD				
INC RD	<table border="1"><tr><td>0111</td><td>**</td><td>RD</td></tr></table>	0111	**	RD	RD+1 → RD	
0111	**	RD				
LAD M D, RD	<table border="1"><tr><td>1100</td><td>M</td><td>RD</td><td>D</td></tr></table>	1100	M	RD	D	E → RD
1100	M	RD	D			
STA M D, RS	<table border="1"><tr><td>1101</td><td>M</td><td>RD</td><td>D</td></tr></table>	1101	M	RD	D	RD → E
1101	M	RD	D			
JMP M D	<table border="1"><tr><td>1110</td><td>M</td><td>**</td><td>D</td></tr></table>	1110	M	**	D	E → PC
1110	M	**	D			
BZC M D	<table border="1"><tr><td>1111</td><td>M</td><td>**</td><td>D</td></tr></table>	1111	M	**	D	当 FC 或 FZ=1 时, E → PC
1111	M	**	D			
IN RD, P	<table border="1"><tr><td>0010</td><td>**</td><td>RD</td><td>P</td></tr></table>	0010	**	RD	P	[P] → RD
0010	**	RD	P			
OUT P, RS	<table border="1"><tr><td>0011</td><td>RS</td><td>**</td><td>P</td></tr></table>	0011	RS	**	P	RS → [P]
0011	RS	**	P			
LDI RD, D	<table border="1"><tr><td>0110</td><td>**</td><td>RD</td><td>D</td></tr></table>	0110	**	RD	D	D → RD
0110	**	RD	D			
HALT	<table border="1"><tr><td>0101</td><td>**</td><td>**</td></tr></table>	0101	**	**	停机	
0101	**	**				

#### 8.4 总体设计

本模型机的数据通路框图如图 8-2 所示。

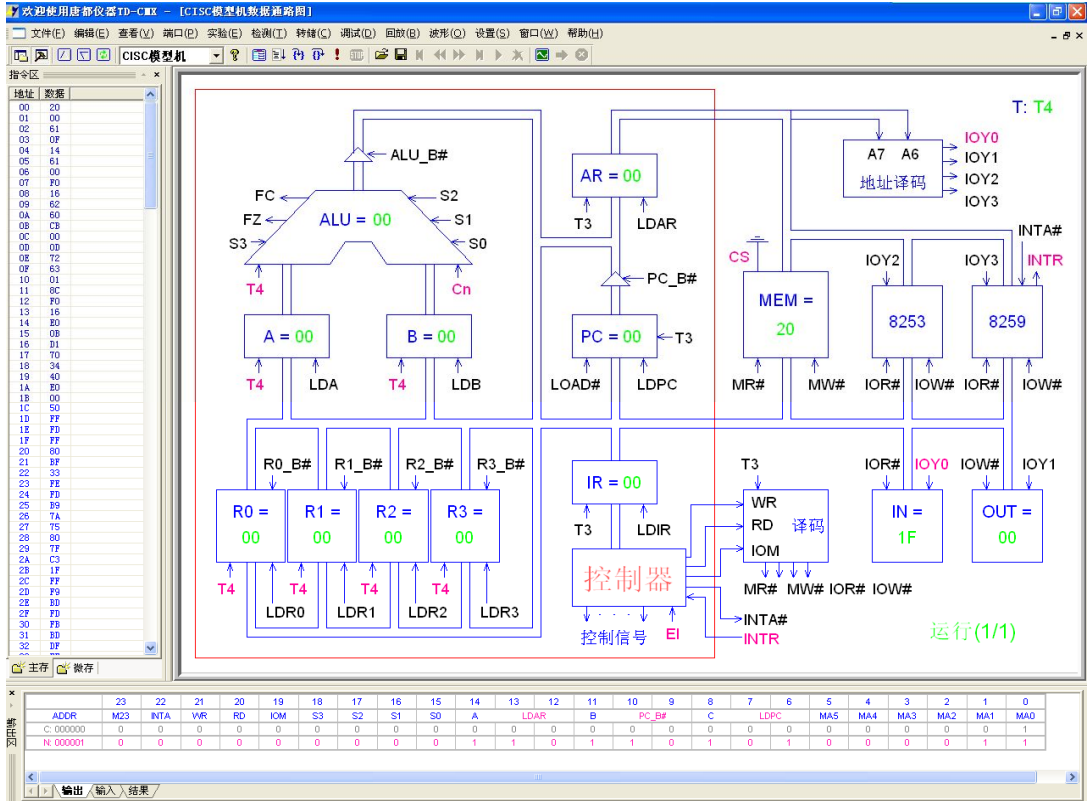


图 8-2 数据通路框图

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 8-3 所示在 IR 单元的 INS\_DEC 中实现。

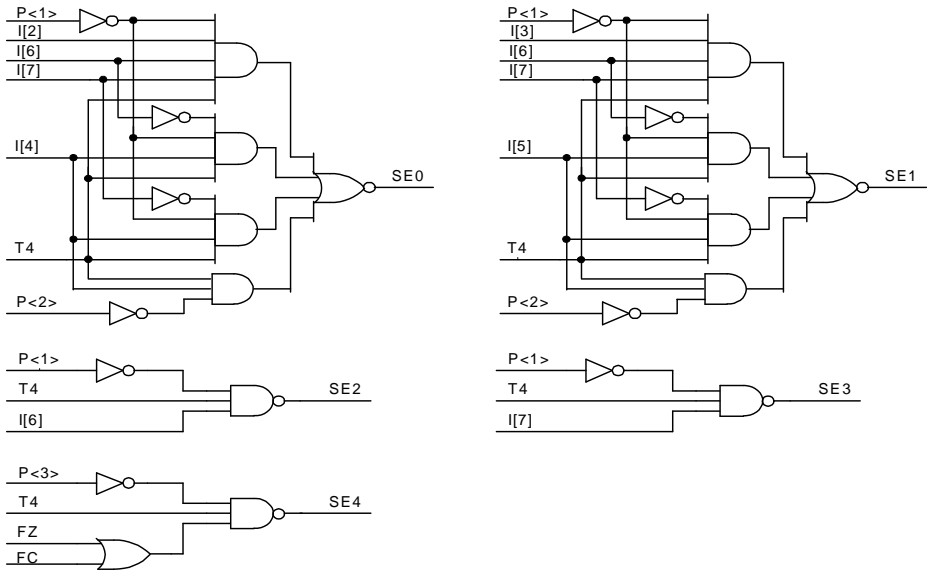


图 8-3 指令译码原理图

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，为此还设计一个寄存器译码电路，在 IR 单元的 REG\_DEC (GAL16V8) 中实现，如图 8-4 所示。

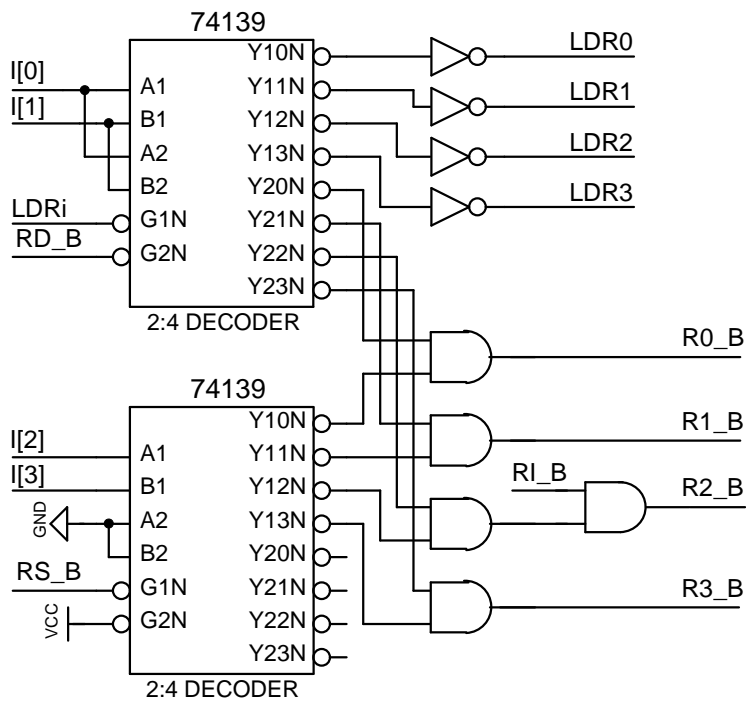


图 8-4 寄存器译码原理图

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 8-5 所示。

按照系统建议的微指令格式，见表 8-4，参照微指令流程图，将每条微指令代码化，译成二进制代码表，见表 8-5，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 8-4 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字段	B字段	C字段	MA5-MA0

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

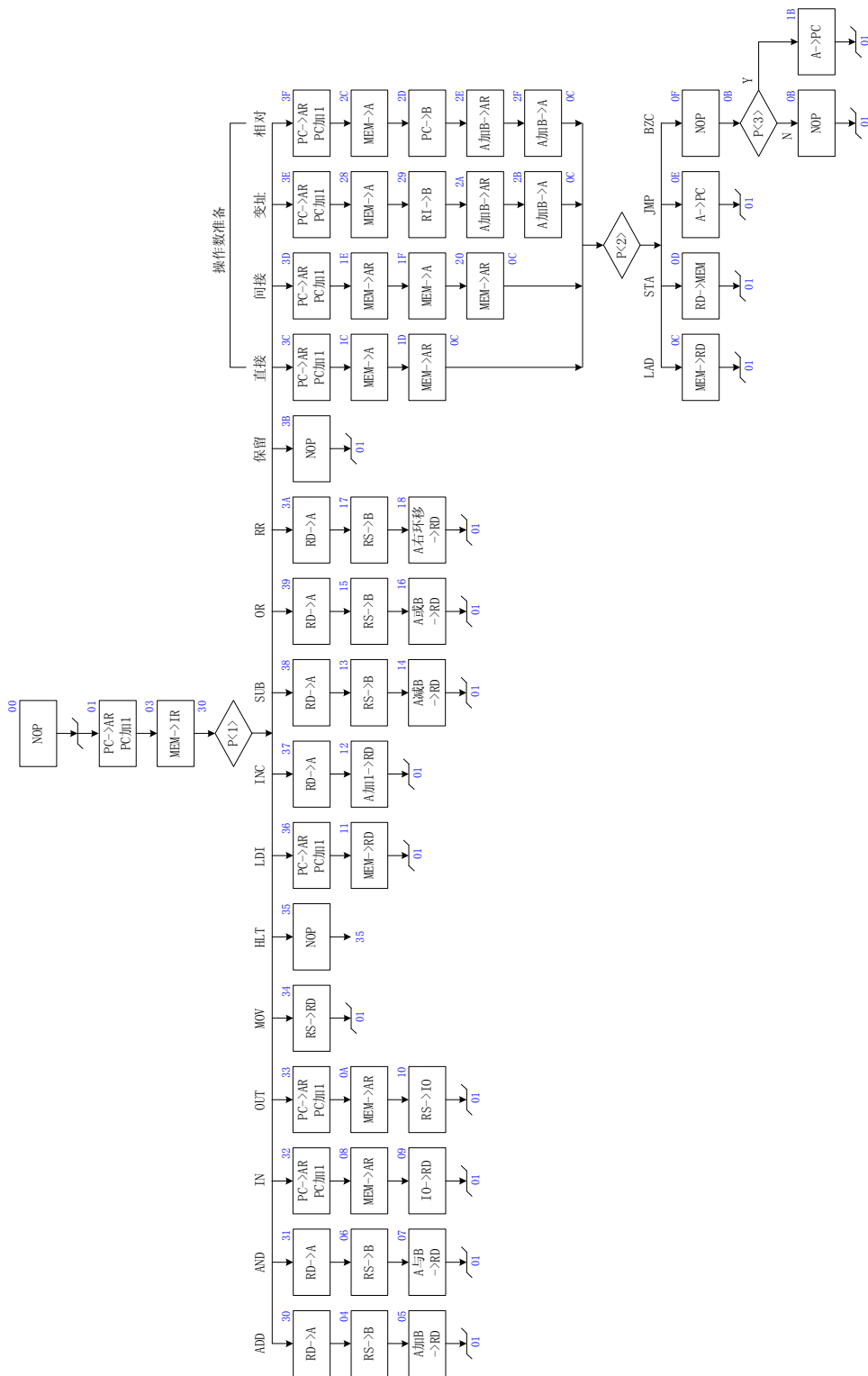


图 8-5 微程序流程图

表 8-5 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	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	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	04 32 01	00000	1000	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	110	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	010	000	000100
31	00 16 06	00000	0000	001	010	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	010	000	010011
39	00 16 15	00000	0000	001	010	000	010101
3A	00 16 17	00000	0000	001	010	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  $1+2+\dots+X$  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 (H)	内容 (H)	助记符	说明
00	20 00	<b>START:</b> IN R0, 00H	从 IN 单元读入计数初值
02	61 0F	LDI R1, 0FH	立即数 0FH 送 R1
04	14	AND R0, R1	得到 R0 低四位
05	61 00	LDI R1, 00H	装入和初值 00H
07	F0 16	BZC RESULT	计数值为 0 则跳转
09	62 60	LDI R2, 60H	读入数据始地址
0B	CB 00	<b>LOOP:</b> LAD R3, [RI], 00H	读入数据送 R3, 变址寻址, 偏移量为 00H
0D	0D	ADD R1, R3	累加求和
0E	72	INC RI	变址寄存器加 1, 指向下一数据
0F	63 01	LDI R3, 01H	装入比较值
11	8C	SUB R0, R3	
12	F0 16	BZC RESULT	相减为 0, 表示求和完毕
14	E0 0B	JMP LOOP	未完则继续
16	D1 70	<b>RESULT:</b> STA 70H, R1	和存于 MEM 的 70H 单元
18	34 40	OUT 40H, R1	和在 OUT 单元显示
1A	E0 00	JMP START	跳转至 START
1C	50	HLT	停机

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

## 8.5 实验步骤

1. 按图 8-6 连接实验线路，仔细检查接线后打开实验箱电源。

2. 联机写入实验程序，并进行校验。选择联机软件的“【转储】—【装载】”功能，在 C:\TangDu\CMX\Sample 路径下打开文件对话框中选择“CISC 模型机设计实验.txt”文件，软件自动将机器程序和微程序写入指定单元。机器程序见前面。

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，如果机器程序有错误，重点检查与存储器有关的连线。模型机能正常运行的前提是微程序和机器指令必须正确。转储下载正确，表明硬件连线没有问题，否则必须重新连线。

在调试过程中，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. 联机运行程序

进入软件界面，选择菜单命令“【实验】—【CISC 模型机】”，打开 CISC 模型机实验数据通路图，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。

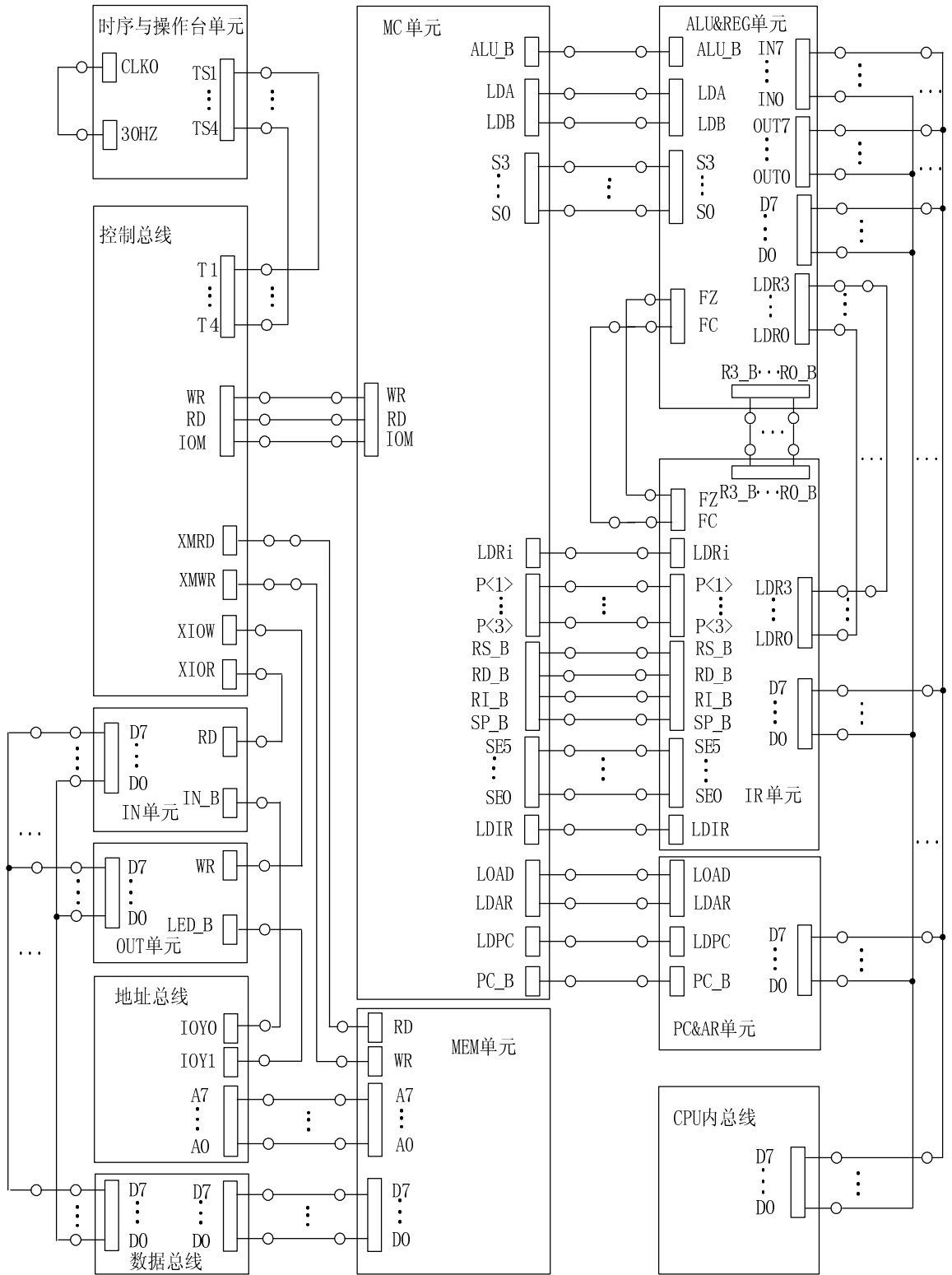


图 8-6 实验接线图



; 微程序

\$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 043201 ; 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

## 实验九 带中断处理能力的模型机设计实验

计算机的输入输出系统也称为 I/O 系统，包括外围设备、设备控制器、I/O 接口以及一些专门为输入输出操作而设计的软件和硬件。

### 9.1 实验目的

- (1) 掌握中断原理及其响应流程。
- (2) 掌握 8259 中断控制器原理及其应用编程。

### 9.2 实验设备

PC 机一台，TD-CMX 实验系统一套。

### 9.3 实验原理

8259 的引脚分配图如图 9-1 所示。

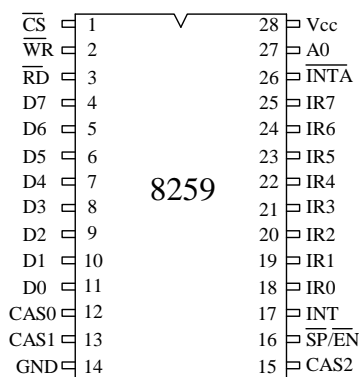


图 9-1 8259 芯片引脚说明

8259 芯片引脚说明如下。

- D7~D0 为双向三态数据线
- $\overline{CS}$  片选信号线
- A0 用来选择芯片内部不同的寄存器，通常接至地址总线的 A0。
- $\overline{RD}$  读信号线，低电平有效，其有效时控制信息从 8259 读至 CPU。
- $\overline{WR}$  写信号线，低电平有效，其有效时控制信息从 CPU 写入至 8259。
- $\overline{SP/EN}$  从程序/允许缓冲
- $\overline{INTA}$  中断响应输入
- INT 中断输出
- IR0~IR7 8 条外界中断请求输入线。
- CAS2~CAS0 级连信号线。

$\overline{CS}$ 、A0、 $\overline{RD}$ 、 $\overline{WR}$ 、D4、D3 位的电平与 8259 操作关系如表 9-1 所示。

CPU 必须有一个中断使能寄存器，并且可以通过指令对该寄存器进行操作，其原理如图 9-2 所示。CPU **开中断**指令 STI 对其置 1，而 CPU **关中断**指令 CLI 对其置 0。

表 9-1 8259A 的读/写操作

A0	D4	D3	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	操 作
0			0	1	0	输入操作(读) IRR, ISR或中断级别 → 数据总线
1			0	1	0	IMR 数据总线
0	0	0	1	0	0	输出操作(写) 数据总线 → OCW2
0	0	1	1	0	0	数据总线 → OCW3
0	1	X	1	0	0	数据总线 → OCW1
1	X	X	1	0	0	数据总线 → ICW1, ICW2, ICW3, ICW4
X	X	X	1	1	0	断开功能 数据总线 → 三态(无操作)
X	X	X	X	X	1	数据总线 → 三态(无操作)

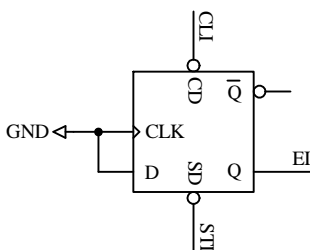


图 9-2 中断使能寄存器原理图

8259 的数据线 D7…D0 挂接到数据总线,地址线 A0 挂接到地址总线的 A0 上,片选信号  $\overline{CS}$  接控制总线的 IOY3, IOY3 由地址总线的高 2 位译码产生,其地址方配见表 9-2,  $\overline{RD}$ 、 $\overline{WR}$  (实验箱上丝印为 XIOR 和 XIOW) 接 CPU 给出的读写信号,8259 和系统的连接如图 9-3 所示。

表 9-2 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

本实验要求设计的模型计算机具备有类 X86 的中断功能,当外部中断请求有效、CPU 允许中断,且在一条指令执行完时,CPU 将响应中断。当 CPU 响应中断时,将会向 8259 发送两个

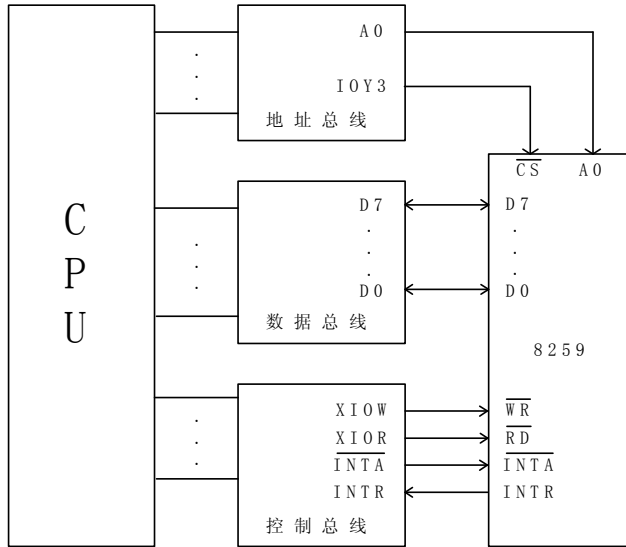


图 9-3 8259 和 CPU 连接图

连续的  $\overline{INTA}$  信号，请注意，8259 是在接收到第一个  $\overline{INTA}$  信号后锁住向 CPU 的中断请求信号 INTR（高电平有效），并且在第二个  $\overline{INTA}$  信号到达后将其变为低电平（自动 EOI 方式），所以，中断请求信号 IR0 应该维持一段时间，直到 CPU 发送出第一个  $\overline{INTA}$  信号，这才是一个有效的中断请求。8259 在收到第二个  $\overline{INTA}$  信号后，就会将中断向量号发送到数据总线，CPU 读取中断向量号，并转入相应的中断处理程序中。

本系统的指令译码电路是在 IR 单元的 INS\_DEC (GAL20V8) 中实现，如图 9-4 所示。和前面复杂模型机实验指令译码电路相比，主要增加了对中断的支持，当 INTR（有中断请求）和 EI（CPU 允许中断）均为 1，且  $P < 4$  测试有效，那么在 T4 节拍时，微程序就会产生中断响应分支，从而使得 CPU 能响应中断。

在中断过程中需要有现场保护，而且在编程的过程中也需要一些压栈或弹栈操作，所以还需设置一个堆栈，由 R3 做堆栈指针。系统的寄存器译码电路如图 9-5 所示，在 IR 单元的 REG\_DEC (GAL16V8) 中实现，和前面复杂模型机实验寄存器译码电路相比，增加一个或门和一个与门，用以支持堆栈操作。

本模型机共设计 16 条指令，表 9-3 列出了基本指令的格式、助记符及其功能。

其中，D 为立即数，P 为外设的端口地址，RS 为源寄存器，RD 为目的寄存器，并规定如下。

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

设定微指令格式，如表 9-4 所示。

根据指令系统要求，设计微程序流程及确定微地址，并得到微程序流程图，如图 9-6 所示。

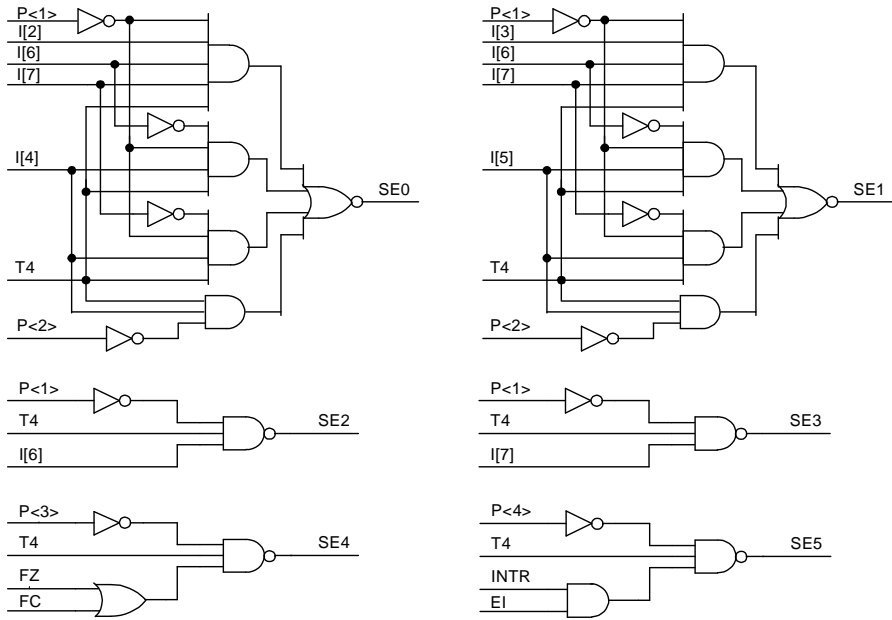


图 9-4 指令译码原理图

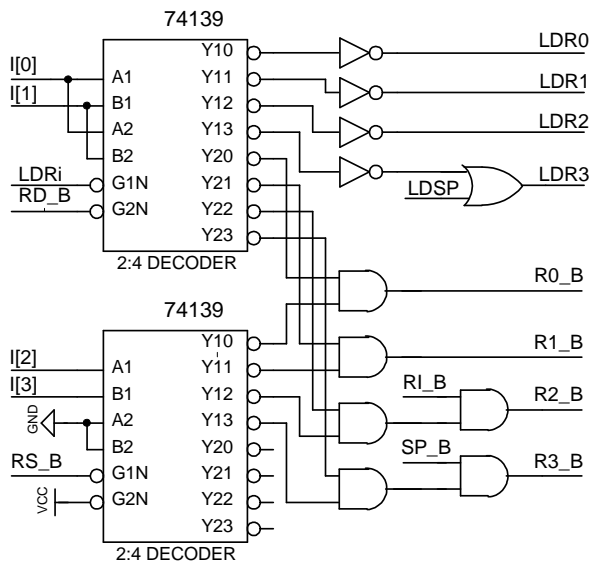


图 9-5 寄存器译码原理图

表 9-3 指令助记符、格式及功能

助记符号	指令格式	指令功能				
MOV RD, RS	<table border="1"><tr><td>0100</td><td>RS</td><td>RD</td></tr></table>	0100	RS	RD	RS → RD	
0100	RS	RD				
ADD RD, RS	<table border="1"><tr><td>0000</td><td>RS</td><td>RD</td></tr></table>	0000	RS	RD	RD + RS → RD	
0000	RS	RD				
AND RD, RS	<table border="1"><tr><td>0001</td><td>RS</td><td>RD</td></tr></table>	0001	RS	RD	RD ∧ RS → RD	
0001	RS	RD				
STI	<table border="1"><tr><td>0111</td><td>**</td><td>**</td></tr></table>	0111	**	**	CPU开中断	
0111	**	**				
CLI	<table border="1"><tr><td>1000</td><td>**</td><td>**</td></tr></table>	1000	**	**	CPU关中断	
1000	**	**				
PUSH RS	<table border="1"><tr><td>1001</td><td>RS</td><td>**</td></tr></table>	1001	RS	**	RS → 堆栈	
1001	RS	**				
POP RD	<table border="1"><tr><td>1010</td><td>**</td><td>RD</td></tr></table>	1010	**	RD	堆栈 → RD	
1010	**	RD				
IRET	<table border="1"><tr><td>1011</td><td>**</td><td>**</td></tr></table>	1011	**	**	中断返回	
1011	**	**				
LAD M D, RD	<table border="1"><tr><td>1100</td><td>M</td><td>RD</td><td>D</td></tr></table>	1100	M	RD	D	E → RD
1100	M	RD	D			
STA M D, RS	<table border="1"><tr><td>1101</td><td>M</td><td>RD</td><td>D</td></tr></table>	1101	M	RD	D	RD → E
1101	M	RD	D			
JMP M D	<table border="1"><tr><td>1110</td><td>M</td><td>**</td><td>D</td></tr></table>	1110	M	**	D	E → PC
1110	M	**	D			
BZC M D	<table border="1"><tr><td>1111</td><td>M</td><td>**</td><td>D</td></tr></table>	1111	M	**	D	当FC或FZ=1时, E → PC
1111	M	**	D			
IN RD, P	<table border="1"><tr><td>0010</td><td>**</td><td>RD</td><td>P</td></tr></table>	0010	**	RD	P	[P] → RD
0010	**	RD	P			
OUT P, RS	<table border="1"><tr><td>0011</td><td>RS</td><td>**</td><td>P</td></tr></table>	0011	RS	**	P	RS → [P]
0011	RS	**	P			
LDI RD, D	<table border="1"><tr><td>0110</td><td>**</td><td>RD</td><td>D</td></tr></table>	0110	**	RD	D	D → RD
0110	**	RD	D			
HALT	<table border="1"><tr><td>0101</td><td>**</td><td>**</td></tr></table>	0101	**	**	停机	
0101	**	**				

表 9-4 微指令格式

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

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	LDSP
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	SP_B
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	P<4>
1	0	1	LDPC
1	1	0	STI
1	1	1	CLI

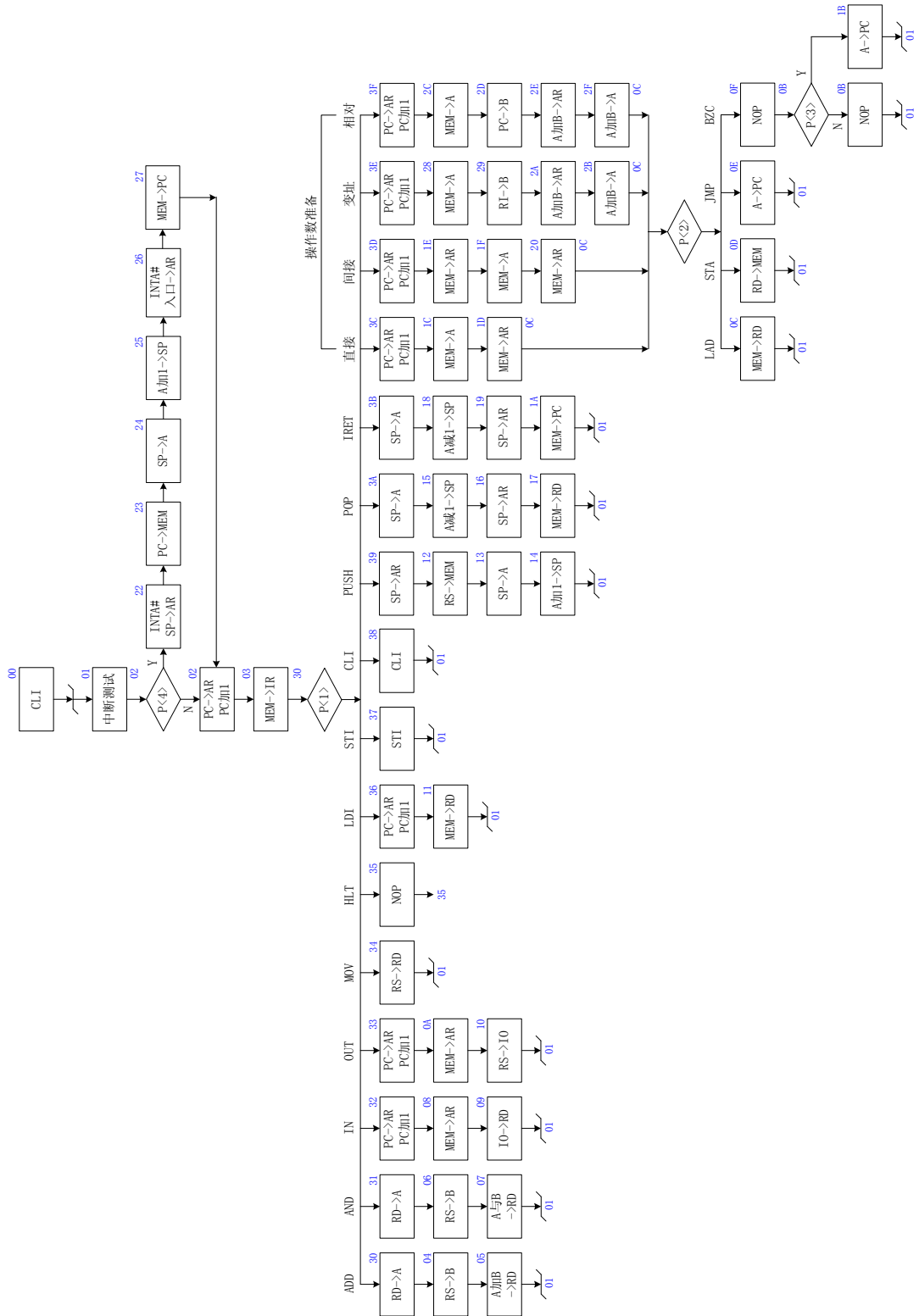


图 9-6 微程序流程图



参照微程序流程图，将每条微指令代码化，译成二进制微代码表，如表 9-5 所示。

表 9-5 二进制微代码表

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 01 C1	00000	0000	000	000	111	000001
01	00 01 02	00000	0000	000	000	100	000010
02	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	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	20 04 13	00100	0000	000	010	000	010011
13	00 1A 14	00000	0000	001	101	000	010100
14	06 C2 01	00000	1101	100	001	000	000001
15	06 42 16	00000	1100	100	001	000	010110
16	00 6A 17	00000	0000	110	101	000	010111
17	10 30 01	00010	0000	011	000	000	000001
18	06 42 19	00000	1100	100	001	000	011001
19	00 6A 1A	00000	0000	110	101	000	011010
1A	10 51 41	00010	0000	101	000	101	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
22	40 6A 23	01000	0000	110	101	000	100011

23	20 0C 24	01000	0000	000	110	000	100100
24	00 1A 25	00000	0000	001	101	000	100101
25	06 C2 26	00000	1101	100	001	000	100110
26	40 60 27	01000	0000	110	000	000	100111
27	10 51 42	00010	0000	101	000	101	000010
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	010	000	000100
31	00 16 06	00000	0000	001	010	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 01 81	00000	0000	000	000	110	000001
38	00 01 C1	00000	0000	000	000	111	000001
39	00 6A 12	00000	0000	110	101	000	010010
3A	00 1A 15	00000	0000	001	101	000	010101
3B	00 1A 18	00000	0000	001	101	000	011000
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，编写一段程序，在模型机上实现以下功能：从 IN 单元读入一个数据 X 存于寄存器 R0，CPU 每响应一次中断，对 R0 中的数据加 1，并输出到 OUT 单元。程序见后面。

## 9.4 实验步骤

1. 按图 9-6 所示连接实验接线，仔细检查接线后打开实验箱电源。
2. 联机写入和校验

选择联机软件的“【转储】—【装载】”功能，在 <C:\TangDu\CMX\Sample> 路径下打开文件对话框，选择“带中断处理能力的模型机设计实验.txt”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，如果机器程序有错误，重点检查与存储器有关的连线。模型机能正常运行的前提是微程序和机器指令必须正确。**转储下载正确，表明硬件连线没有问题**，否则必须重新连线。

在调试过程中，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‘连续’档**，按动一次 ST 按钮，即可连续运行指令，按动 KK 开关，每按动一次，检查 OUT 单元显示的数是否在原有基础加 1（第一次是在 IN 单元值的基础加 1）。

方法二：联机运行

进入软件界面，选择菜单命令“【实验】—【CISC 模型机】”，打开 CISC 模型机数据通路图，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，在数据通路图和微程序流中观测程序的执行过程。在微程序流程图中观测：选择‘单周期’运行程序，当模型机执行完 MOV 指令后，**按下 KK 开关，不要松开，可见控制总线 INTR 指示灯亮**，继续‘单周期’运行程序，直到模型机的 CPU 向 8259 发送完第一个  $\overline{INTA}$ ，然后松开 KK 开关，INTR 中断请求被 8259 锁存，CPU 响应中断。仔细分析中断响应时现场保护的过程，中断返回时现场恢复的过程。

每响应一次中断，检查 OUT 单元显示的数是否在原有基础加 1（第一次是在 IN 单元值的基础加 1）。

#### **机器指令程序**

\$P 00 60	;	LDI R0, 13H	将立即数 13 装入 R0
\$P 01 13			
\$P 02 30	;	OUT C0H, R0	将 R0 中的内容写入端口 C0 中，即写
\$P 03 C0	;		ICW1，边沿触发，单片模式，需要 ICW4
\$P 04 60	;	LDI R0, 30H	将立即数 30 装入 R0
\$P 05 30			
\$P 06 30	;	OUT C1H, R0	将 R0 中的内容写入端口 C1 中，即写
\$P 07 C1	;		ICW2，中断向量为 30-37
\$P 08 60	;	LDI R0, 03H	将立即数 03 装入 R0
\$P 09 03			
\$P 0A 30	;	OUT C1H, R0	将 R0 中的内容写入端口 C1 中，即写
\$P 0B C1	;		ICW4，非缓冲，86 模式，自动 EOI
\$P 0C 60	;	LDI R0, FEH	将立即数 FE 装入 R0
\$P 0D FE			
\$P 0E 30	;	OUT C1H, R0	将 R0 中的内容写入端口 C1 中，即写
\$P 0F C1	;		OCW1，只允许 IRO 请求
\$P 10 63	;	LDI SP, A0H	初始化堆栈指针为 A0

\$P 11 A0		
\$P 12 70	; STI	CPU 开中断
\$P 13 20	; IN R0, 00H	从端口 00 (IN 单元) 读入计数初值
\$P 14 00		
\$P 15 41	; LOOP: MOV R1, R0	移动数据, 并等待中断
\$P 16 E0	; JMP LOOP	跳转, 并等待中断
\$P 17 15		

### 中断服务程序:

\$P 20 80	; CLI	CPU 关中断
\$P 21 61	; LDI R1, 01H	将立即数 01 装入 R1
\$P 22 01		
\$P 23 04	; ADD R0, R1	将 R0 和 R1 相加, 即计数值加 1
\$P 24 30	; OUT 40H, R0	将计数值输出到端口 40 (OUT 单元)
\$P 25 40		
\$P 26 70	; STI	CPU 开中断
\$P 27 B0	; IRET	中断返回
\$P 30 20		IRO 的中断入口地址 20

### 微程序

\$M 00 0001C1	; NOP
\$M 01 000102	; 中断测试, P<4>
\$M 02 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 200413	; RS->MEM
\$M 13 001A14	; SP->A
\$M 14 06C201	; A 加 1->SP
\$M 15 064216	; A 减 1->SP
\$M 16 006A17	; SP->AR
\$M 17 103001	; MEM->RD
\$M 18 064219	; A 减 1->SP
\$M 19 006A1A	; SP->AR
\$M 1A 105141	; MEM->PC
\$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 22 406A23 ; INTA#, SP->AR
$M 23 200C24 ; PC->MEM
$M 24 001A25 ; SP->A
$M 25 06C226 ; A 加 1->SP
$M 26 406027 ; INTA#, 入口->AR
$M 27 105142 ; MEM->PC
$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 000181 ; STI
$M 38 0001C1 ; CLI
$M 39 006A12 ; SP->AR
$M 3A 001A15 ; SP->A
$M 3B 001A18 ; SP->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

```

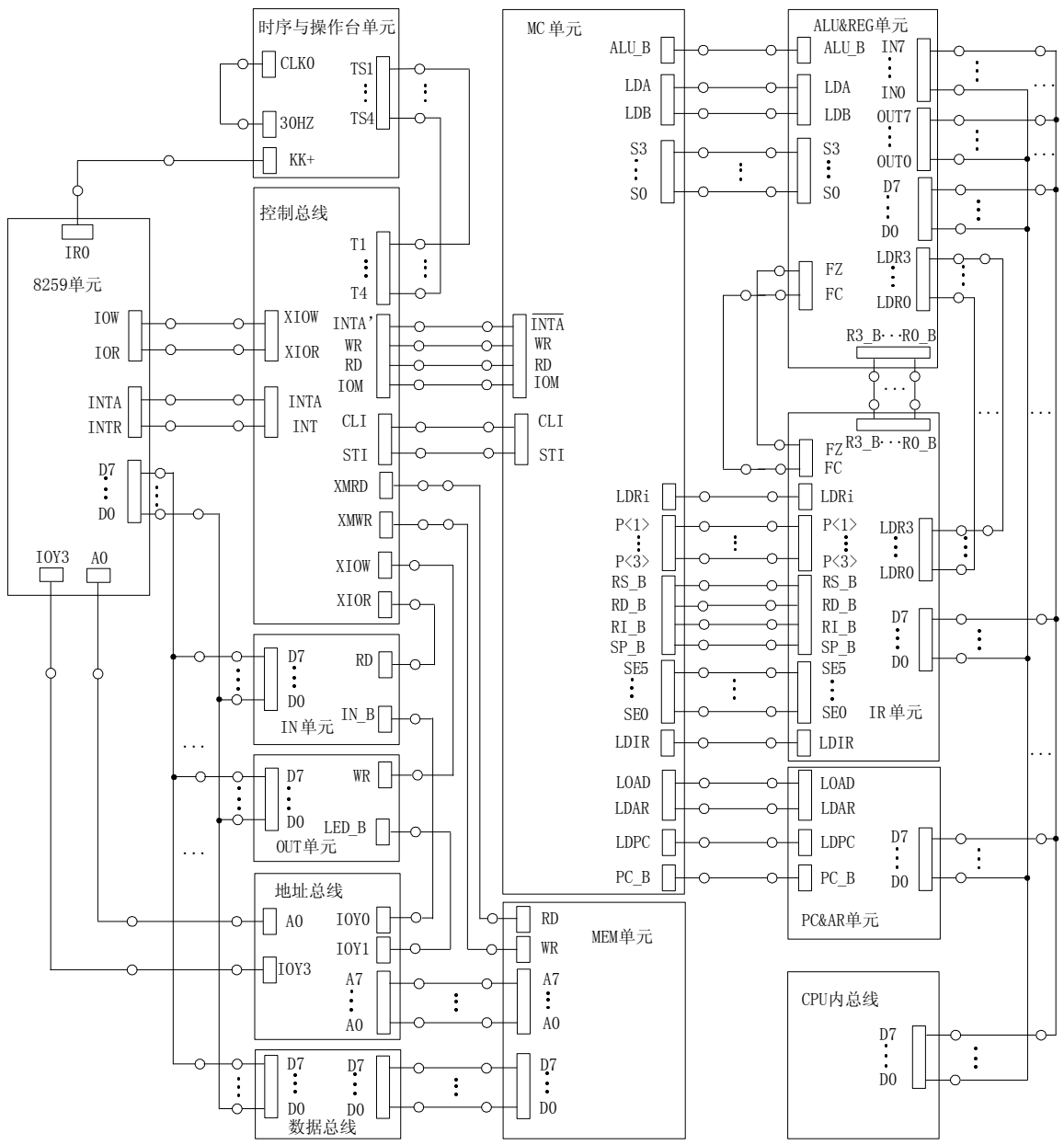


图 9-6 实验接线图

# 附录 1 软件使用说明

## 软件运行环境

操作系统：Windows 98/NT/2000/XP

最低配置：

CPU：奔腾 300MHz

内存：64MB

显示卡：标准 VGA，256 色显示模式以上

硬盘：20MB 以上

光驱：标准 CD-ROM

## 安装软件

安装操作如下：

可以通过“资源管理器”，找到光盘驱动器本软件安装目录下的‘安装 CMX. EXE’，双击执行它，按屏幕提示进行安装操作。

“TD-CMX”软件安装成功后，在“开始”的“程序”里将出现“CMX”程序组，点击“CMX”即可执行程序。

## 卸载软件

联机软件提供了自卸载功能，使您可以方便地删除“TD-CMX”的所有文件、程序组或快捷方式。单击【开始】/【程序】打开“CMX”的程序组，然后运行“卸载”项，就可执行卸载功能，按照屏幕提示操作即可以安全、快速地删除“TD-CMX”。

## （一）界面窗口介绍

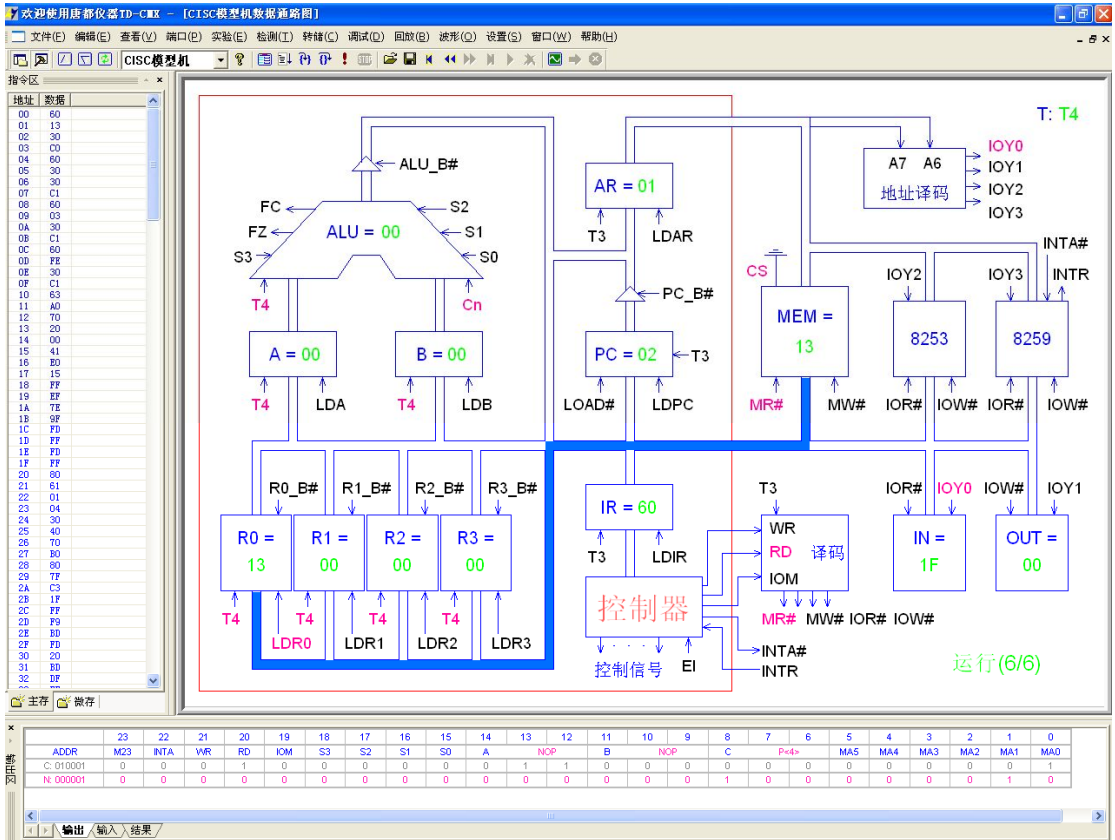
主界面如附图 1-1 所示，由指令区、输出区和图形区三部分组成。

### 指令区：

分为机器指令区和微指令区，指令区下方有两个 Tab 按钮，可通过按钮在两者之间切换。

机器指令区：分为两列，第一列为主存地址（00—FF，共 256 个单元），第二列为每个地址所对应的数值。串口通讯正常且串口无其它操作，可以直接修改指定单元的内容，用鼠标单击要修改单元的数据，此时单元格会变成一个编辑框，即可输入数据，编辑框只接收两位合法的 16 进制数，按回车键确认，或用鼠标点击别的区域，即可完成修改工作。按下 ESC 键可取消修改，编辑框会自动消失，恢复显示原来的值，也可以通过上下方向键移动编辑框。

微指令区：分为两列，第一列为微控器地址（00—3F，共 64 个单元），第二列为每个地址所对应的微指令，共 6 字节。**修改微指令操作和修改机器指令一样**，只不过微指令是 6 位，而机器指令是 2 位。



附图 1-1 软件主界面

### 输出区：

输出区由输出页、输入页和结果页组成。

输出页：在数据通路图打开，且该通路中用到微程序控制器，运行程序时，输出区用来实时显示 **当前正在执行 (C 010001)** 的微指令和 **下条将要执行 (N 000001)** 的微指令的 24 位微码及其微地址。当前正在执行微指令的显示可通过菜单命令“【设置】—【当前微指令】”进行开关。

输入页：可以对微指令进行按位输入及模拟，鼠标左键单击 ADDR 值，此时单元格会变成一个编辑框，即可输入微地址，输入完毕后回车，编辑框消失，后面的 24 位代表当前地址的 24 位微码，微码值用红色显示，鼠标左键单击微码值可使该值在 0 和 1 之间切换。在数据通路图打开时，按动‘模拟’按钮，可以在数据通路中模拟该微指令的功能，按动‘修改’按钮则可以将当前显示的微码值下载到下位机。

结果页：用来显示一些提示信息或错误信息，保存和装载程序时会在这区域显示一些提示信息。在系统检测时，也会在这区域显示检测状态和检测结果。

### 图形区：



可以在此区域编辑指令，显示各个实验的数据通路图、示波器界面等。

## (二) 菜单功能介绍

### 1. 文件菜单项:

文件菜单提供了以下命令:

#### ①. 新建 (N):

在 CMX 中建立一个新文档。在文件新建对话框中选择您所要建立的新文件的类型。

#### ②. 打开 (O)

在一个新的窗口中打开一个现存的文档。您可同时打开多个文档。您可用窗口菜单在多个打开的文档中切换。

#### ③. 关闭 (C)

关闭包含活动文档的所有窗口。CMX 会建议您在关闭文档之前保存对您的文档所做的改动。如果您没有保存而关闭了一个文档，您将会失去自从您最后一次保存以来所做的所有改动。在关闭一无标题的文档之前，CMX 会显示另存为对话框，建议您命名和保存文档。

#### ④. 保存 (S)

将活动文档保存到它的当前的文件名和目录下。当您第一次保存文档时，CMX 显示另存为对话框以便您命名您的文档。如果在保存之前，您想改变当前文档的文件名和目录，您可选用另存为命令。

#### ⑤. 另存为 (A) ...

保存并命名活动文档。CMX 会显示另存为对话框以便您命名您的文档。

#### ⑥. 打印 (P) ...

打印一个文档。在此命令提供的打印对话框中，您可以指明要打印的页数范围、副本数、目标打印机，以及其它打印机设置选项。

#### ⑦. 打印预览 (V)

按要打印的格式显示活动文档。当您选择此命令时，主窗口就会被一个打印预览窗口所取代。这个窗口可以按它们被打印时的格式显示一页或两页。打印预览工具栏提供选项使您可选择一次查看一页或两页，在文档中前后移动，放大和缩小页面，以及开始一个打印作业。

#### ⑧. 打印设置 (R) ...

选择一台打印机和一个打印机连接。在此命令提供的打印设置对话框中，您可以指定打印机及其连接。

#### ⑨. 最近使用文件

您可以通过此列表，直接打开最近打开过的文件，共四个。

#### ⑩. 退出 (X)

结束 CMX 的运行阶段。您也可使用在应用程序控制菜单上的关闭命令。

### 2. 编辑菜单项:

#### ①. 撤消 (U)

新建 (N)	Ctrl+N
打开 (O)...	Ctrl+O
关闭 (C)	
保存 (S)	Ctrl+S
另存为 (A)...	
打印 (P)...	Ctrl+P
打印预览 (V)	
打印设置 (R)...	
最近文件	
退出 (X)	

撤消 (U)	Ctrl+Z
剪切 (T)	Ctrl+X
复制 (C)	Ctrl+C
粘贴 (P)	Ctrl+V

撤消上一步编辑操作。

②. 剪切 (T)

将当前被选取的数据从文档中删除并放置于剪贴板上。如当前没有数据被选取时，此命令则不可用。

③. 复制 (C)

将被选取的数据复制到剪切板上。如当前无数据被选取时，此命令则不可用。

④. 粘贴 (P)

将剪贴板上内容的一个副本插入到插入点处。如剪贴板是空的，此命令则不可用。

3. 查看菜单项:

查看菜单提供了以下命令:



①. 工具栏 (T)

显示和隐藏工具栏，工具栏包括了 CMX 中一些最普通命令的按钮。当工具栏被显示时，在菜单项目的旁边会出现一个打勾记号。

②. 指令区 (W)

显示和隐藏指令区，当指令区被显示时，在菜单项目的旁边会出现一个打勾记号。

③. 输出区 (O)

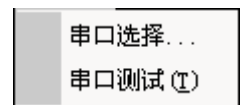
显示和隐藏输出区，当输出区被显示时，在菜单项目的旁边会出现一个打勾记号。

④. 状态栏 (S)

显示和隐藏状态栏。状态栏描述了被选取的菜单项目或被按下的工具栏按钮，以及键盘的锁定状态将要执行的操作。当状态栏被显示时，在菜单项目的旁边会出现一个打勾记号。

4. 端口菜单项:

端口菜单提供了以下命令:



①. 端口选择...

选择通讯端口，选择该命令时会弹出附图 1-2 所示对话框。该命令会自动检测当前系统可用的串口号，并列于组合框中，选择某一串口后，按确定键，对选定串口进行初始化操作，并进行联机测试，报告测试结果，如果联机成功，则会将指令区初始化。



附图 1-2 串口选择对话框

②. 端口测试 (T)

对当前选择的串口进行联机通讯测试，并报告测试结果，只测一次，如果联机成功，则会将指令区初始化。如串不能正常初始化，此命令则不可用。

## 5. 实验菜单项:

实验菜单提供了以下命令:

### ①. 运算器实验

打开运算器实验数据通路图, 如果该通路图已经打开, 则把通路激活并置于最前面显示。

### ②. 存储器实验

打开存储器实验数据通路图, 如果该通路图已经打开, 则把通路激活并置于最前面显示。

### ③. 微控器实验

打开微控器实验数据通路图, 如果该通路图已经打开, 则把通路激活并置于最前面显示。

### ④. 简单模型机

打开简单模型机数据通路图, 如果该通路图已经打开, 则把通路激活并置于最前面显示。

### ⑤. 综合性实验

打开综合性实验数据通路图, 如果该通路图已经打开, 则把通路激活并置于最前面显示。

## 6. 检测菜单项:

检测菜单提供了以下命令:

### ①. 连线检测 (C)

#### a. 简单模型机

对简单模型机的连线进行检测, 并在‘输出区’的‘结果页’显示相关信息。

#### b. 复杂模型机

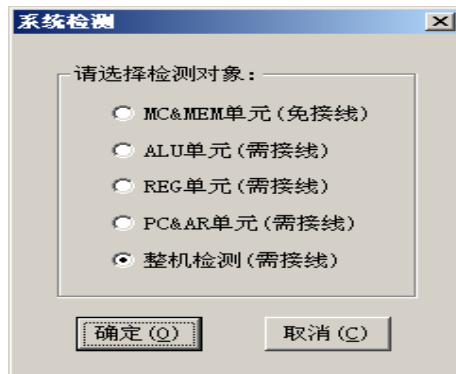
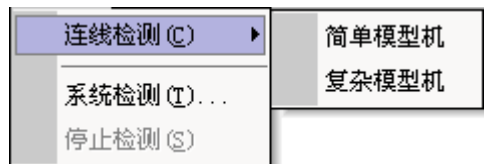
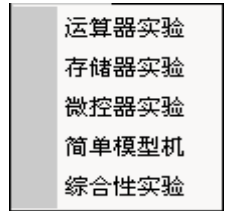
对复杂模型机的连线进行检测, 并在‘输出区’的‘结果页’显示相关信息。

### ②. 系统检测 (T) ...

启动系统检测, 可以进行部件或是整机检测, 进行微控器或存储器检测时不需要连线, 其余的都需要连线。选择该命令时会弹出如附图 1-3 所示对话框, 选择一个对象后按‘确定’键就可以启动系统检测。

### ③. 停止检测 (S)

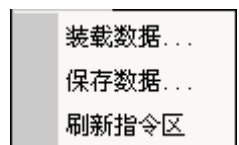
停止系统检测。



附图 1-3 系统检测对话框

## 7. 转储菜单项:

转储菜单提供了以下命令:



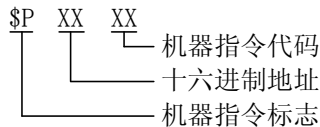
### ①. 装载数据...

将上位机指定文件中的数据装载到下位机中，您选择该命令会弹出打开文件对话框。

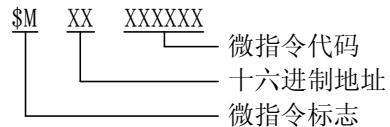
可以打开任意路径下的\*.TXT 文件,如果指令文件合法,系统将把这些指令装载到下位机中,装载指令时,系统提供了一定的检错功能,如果指令文件中有错误的指令,将会导致系统退出装载,并提示错误的指令行。

指令文件中指令书写格式如下:

机器指令格式说明:



微指令格式说明:



例如机器指令\$P00FF,“\$”为标记号,“P”代表机器指令,“00”为机器指令的地址,“FF”为该地址中的数据。微指令\$M00AA77FF,“\$”为标记号,“M”代表微指令,“00”为机器指令的地址,“AA77FF”为该地址中的数据。

### ②. 保存数据...

将下位机中(主存,微控器)的数据保存到上位机中,选择该命令会弹出一个保存对话框,如附图 1-4 所示。



附图 1-4 保存数据对话框

可以选择保存机器指令,此时首尾地址输入框将会变亮,否则首尾地址输入框将会变灰,在允许输入的情况下您可以指定需要保存的首尾地址,微指令也是如此,数据到保存指定路径的\*.TXT 格式文件中。

### ③. 刷新指令区

从下位读取所有机器指令和微指令,并在指令区显示。

## 8. 调试菜单项:

调试菜单提供了以下命令:

### ①. 微程序流程图...



当微控器实验、简单模型机和综合性实验中任一数据通路图打开时，可用此命令来打开指定的微程序流程图，选择该命令会弹出打开文件对话框。

②. 单节拍

向下位机发送单节拍命令，下位机完成一个节拍的工作。

③. 单周期

向下位机发送单周期命令，下位机完成一个机器周期的工作。

④. 单步机器指令

向下位机发送单步机器指令命令，下位机运行一条机器指令。

⑤. 连续运行

向下位机发送连续运行命令，下位机将会进入连续运行状态。

⑥. 停止运行

如果下位机处于连续运行状态，此命令可以使得下位机停止运行。

### 9. 回放菜单项：

回放菜单提供了以下命令。

①. 打开…

打开现存的数据文件。

②. 保存…

保存当前的数据到数据文件。

③. 首端

跳转到首页。

④. 向前

向前翻一页。

⑤. 向后

向后翻一页。

⑥. 末端

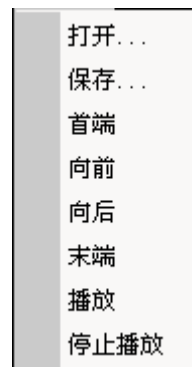
跳转到末页。

⑦. 播放

连续向后翻页。

⑧. 停止播放

停止连续向后翻页。



### 10. 波形菜单项：

波形菜单提供了以下命令：

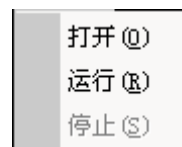
①. 打开 (O)

打开示波器窗口。

②. 运行 (R)

启动示波器，如果下位机正运行程序则不启动。

③. 停止 (S)



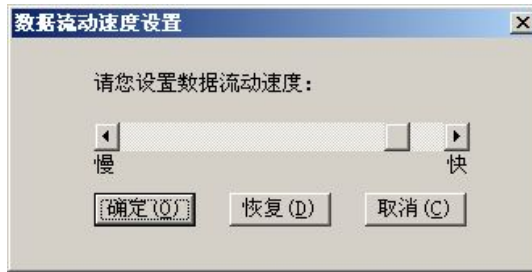
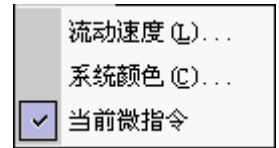
停止处于启动状态的示波器。

## 11. 设置菜单项:

设置菜单提供了以下命令:

### ①. 流动速度 (L) ...

设置数据通路图中数据的流动速度, 选择该命令会弹出一个流动速度设置对话框, 如附图 1-5 所示。拖动滑动块至适当位置, 点击‘确定’按钮即完成设置。



附图 1-5 流动速度设置对话框

### ②. 系统颜色 (C) ...

设置数据通路图、微程序流程图和示波器的显示颜色, 选择该命令会弹出一个设置对话框, 如附图 1-6 所示。



附图 1-6 系统颜色设置对话框

分为三页, 分别为通路图、微流图和示波器, 按动每页的 TAB 按钮, 可在三页之间切换。选择某项要设置的对象, 然后按下‘更改’按钮, 或直接用鼠标左键点击要设置对象的颜色框, 可弹出颜色选择对话框, 选定好颜色后, 点击‘应用’按钮相应对象的颜色就会被修改掉。

### ③. 当前微指令

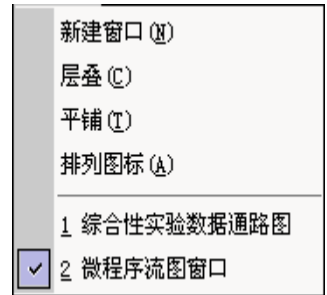
设置‘输出区’的‘输出页’是否显示当前微指令, 当前微指令用灰色显示, 并在地址栏标记为‘C’, 下条将要执行的微指令标记为‘N’。

## 12. 窗口菜单项:

窗口菜单提供了以下命令。这些命令使您能在应用程序窗口中安排多个文档的多个视图：

①. 新建窗口 (N)

打开一个具有与活动的窗口相同内容的新窗口。您可同时打开数个文档窗口以显示文档的不同部分或视图。如果您对一个窗口的内容做了改动，所有其它包含同一文档的窗口也会反映出这些改动。当您打开一个新的窗口，这个新窗口就成了活动的窗口并显示于所有其它打开窗口之上。



②. 层叠 (C)

按相互重叠形式来安排多个打开的窗口。

③. 平铺 (T)

按互不重叠形式来安排多个打开的窗口。

④. 排列图标 (A)

在主窗口的底部安排被最小化的窗口的图标。如果在主窗口的底部有一个打开的窗口，则有可能看不见某些或全部图标，因为它们在这个文档窗口的下面。

⑤. 窗口选择

CMX 在窗口菜单的底部显示出当前打开的文档窗口的清单。有一个打勾记号出现在活动的窗口的文档名前。从该清单中挑选一个文档可使其窗口成为活动窗口。

**13. 帮助菜单项：**

帮助菜单提供以下的命令，为您提供使用这个应用程序的帮助：

①. 关于 (A) CMX...

显示您的 CMX 版本的版权通告和版本号码。

②. 实验帮助 (E) ...

显示实验帮助的开场屏幕。从此开场屏幕，您可跳到关于 CMX 所提供实验的参考资料。

③. 软件帮助 (S) ...

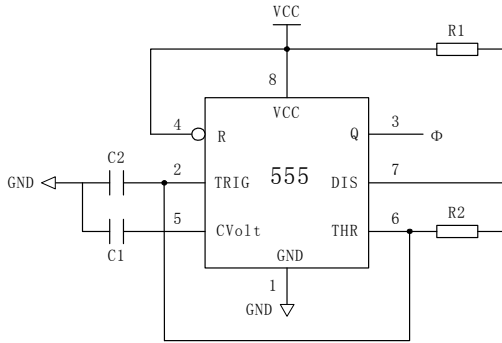
显示软件帮助的开场屏幕。从此开场屏幕，您可跳到关于使用 CMX 设备的参考资料。

**工具栏命令按钮介绍：**

	显示或隐藏指令区		显示或隐藏输出区		保存下位机数据
	向下位机装载数据		刷新指令区数据		打开实验帮助。
	打开微程序流程图		单节拍运行		单周期运行
	单机器指令运行		连续运行		停止运行
	打开实验数据文件		保存实验数据		跳转到首页
	向前翻页		向后翻页		跳转到末页
	连续向后翻页		停止向后翻页		打开示波器窗口
	启动示波器		停止示波器		

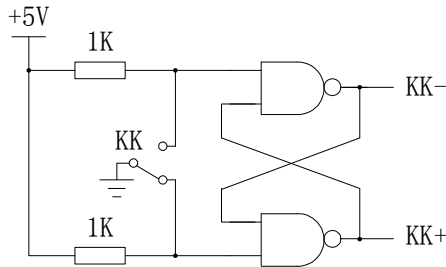
## 附录 2 时序单元介绍

时序单元可以提供两种时序：KK 和 ST。ST 为 555 构成的多谐振荡器的输出，其原理如附图 2-1 所示，频率大约为 3Hz、30Hz、300Hz，占空比约为 50%。



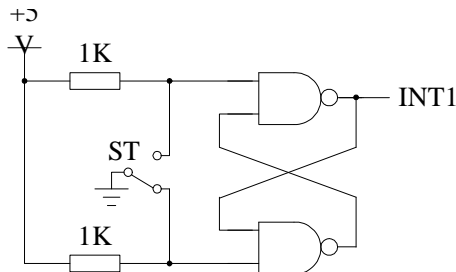
附图 2-1 555 多谐振荡器原理图

每按动一次 KK 按钮，在 KK+ 和 KK- 端将分别输出一个上升沿和下降沿单脉冲。其原理如附图 2-2 所示：



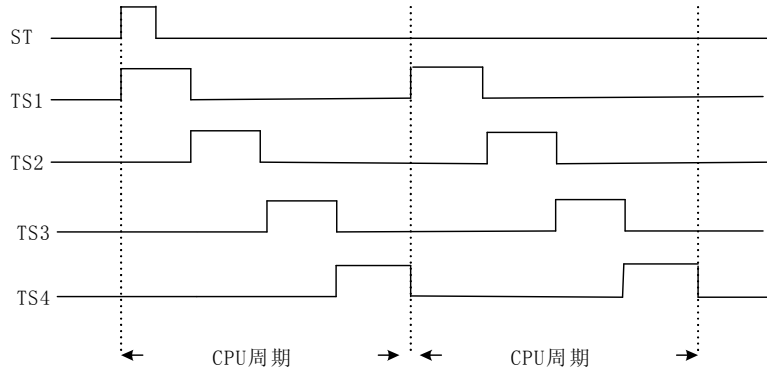
附图 2-2 KK 单脉冲电路原理图

每按动一次 ST 按钮，将对单片机 INT1 产生一次中断请求，其原理如附图 2-3 所示，单片机根据时序开关档位的不同，在 TS1、TS2、TS3、TS4 端输出不同的波形。当开关处于‘连续’档时，TS1、TS2、TS3、TS4 输出的是附图 2-4 所示的连续时序。开关处于‘单步’档时，TS1、TS2、TS3、TS4 只输出一个 CPU 周期的波形，如附图 2-5。开关处于‘单拍’档时，TS1、TS2、TS3、TS4 交替出现，如附图 2-6 所示。

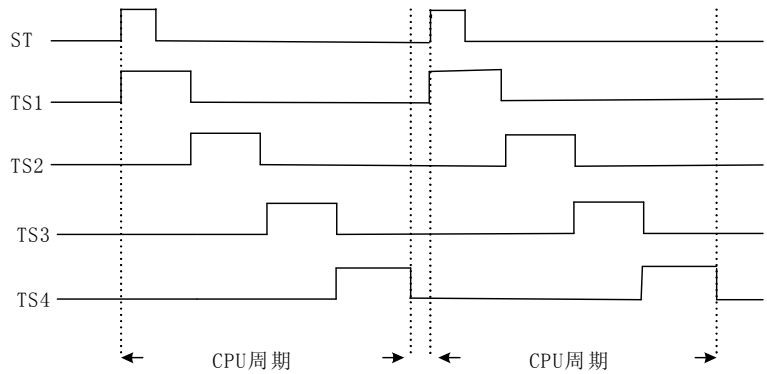


附图 2-3 ST 时序请求原理图

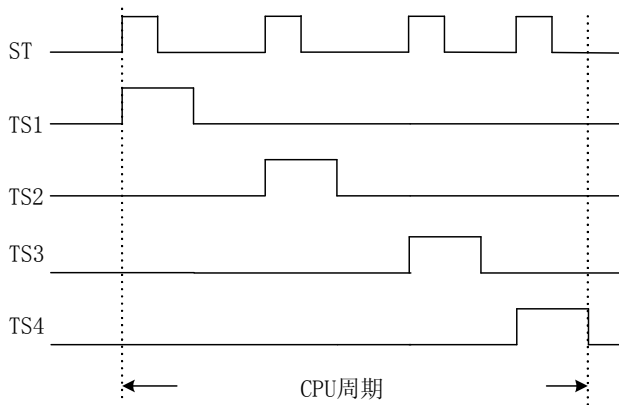




附图 2-4 连续时序



附图 2-5 单步时序



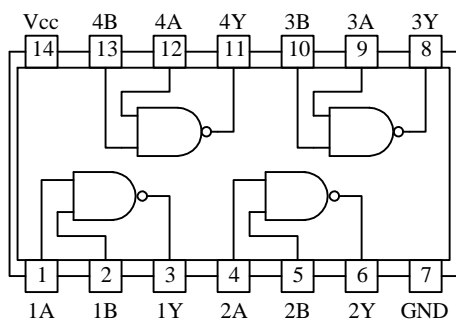
附图 2-6 单拍时序

当 TS1、TS2、TS3、TS4 输出连续波形时，有三种方法可以停止输出，一种是将时序状态开关拨至非连续档，按动 CON 单元的 CLR 按钮或是系统单元的复位按钮，CON 单元的 CLR 按钮和 SYS 单元的复位按钮的区别是，CLR 按钮完成对各单元清零，复位按钮完成对单片机复位。

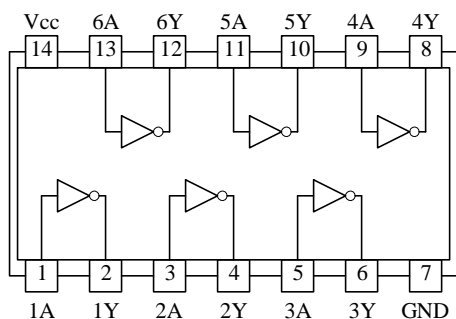
### 附录 3 实验用芯片介绍

本附录介绍一些实验电路中用到的中大规模数字功能器件，以供教学实验参考，其中  $Q_0$  为时钟脉冲的上升沿之前  $Q$  的输出，1 为高电平，0 为低电平，Z 为高阻态。

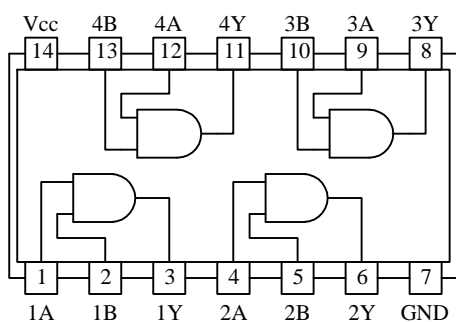
74LS00



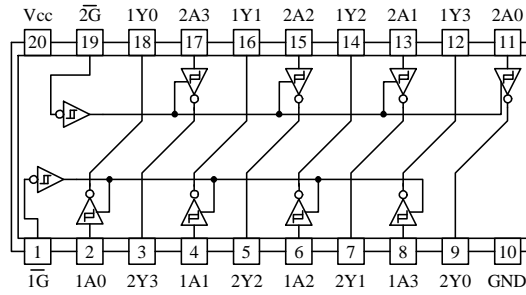
74LS04



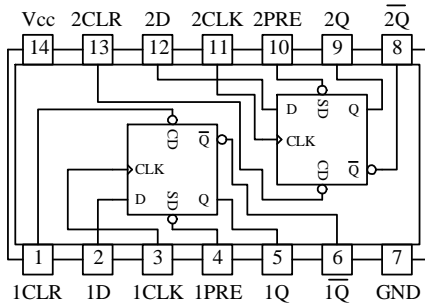
74LS08



74LS240

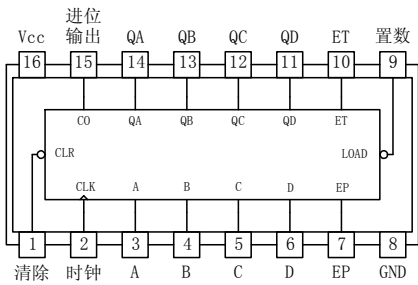


74LS74



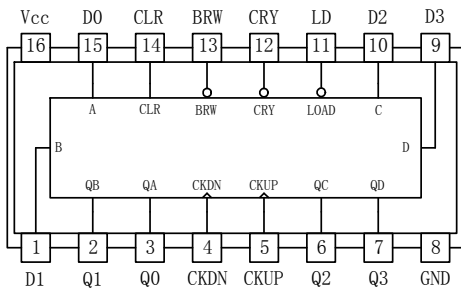
PRC	CLR	CLK	D	Q	$\bar{Q}$
0	1	X	X	1	0
1	0	X	X	0	1
0	0	X	X	1*	1*
1	1	↑	1	1	0
1	1	↑	0	0	1
1	1	0	X	Q <sub>0</sub>	$\bar{Q}_0$

74LS161



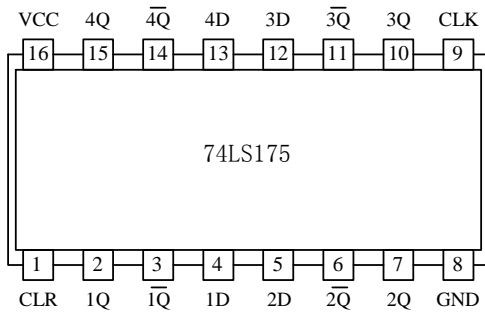
输入				输出			工作		
清除	置数	时钟	使能	QA	QB	QC		QD	进位输出
1	1	↑	1 1	—	—	—	—	—	计数
1	0	↑	X X	A	B	C	D	—	置数
0↓	X	X	X X	0	0	0	0	—	清除
1	X	X	X 1	1	1	1	1	1	—

74LS193



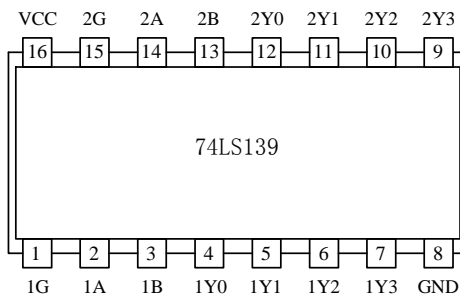
CKUP	CKDN	CLR	LD	功能
↑	1	0	1	上行计数
1	↑	0	1	下行计数
X	X	1	X	清除
X	X	0	0	置数

### 74LS175



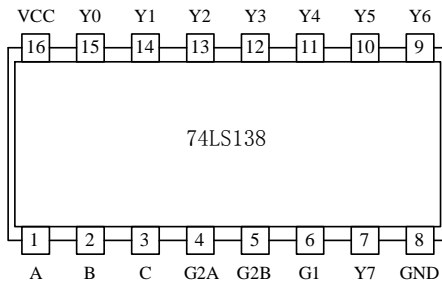
CLR	CLK	D	Q	$\bar{Q}$
0	X	X	0	1
0	↑	1	1	0
1	↑	0	0	1
1	0	X	$Q_0$	$\bar{Q}_0$

### 74LS139



G	B	A	Y0	Y1	Y2	Y3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

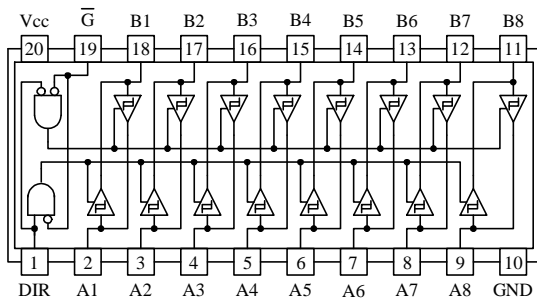
### 74LS138



G1	G*	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	1	X	X	X	1	1	1	1	1	1	1	1
0	X	X	X	X	1	1	1	1	1	1	1	1
1	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	1	1	0	1	1	1	1	1	1
1	0	0	1	0	1	1	0	1	1	1	1	1
1	0	0	1	1	1	1	0	1	1	1	1	1
1	0	1	0	0	1	1	1	1	0	1	1	1
1	0	1	0	1	1	1	1	1	1	0	1	1
1	0	1	1	0	1	1	1	1	1	1	0	1
1	0	1	1	1	1	1	1	1	1	1	1	0

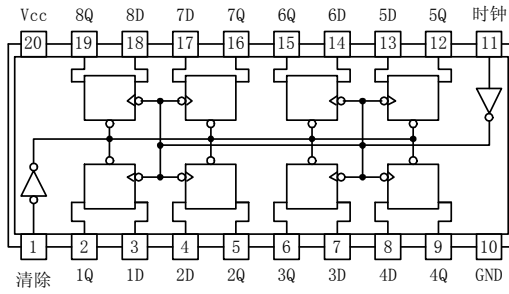
$G^* = G2A + GAB$

### 74LS245



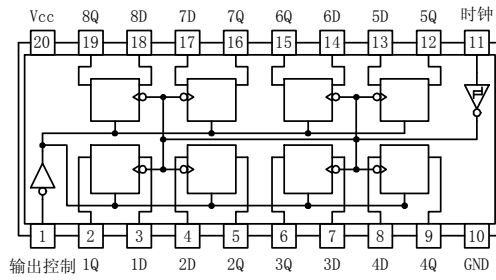
使能 G	方向控制 DIR	操作
0	0	B→A
0	1	A→B
1	X	隔开

### 74LS273



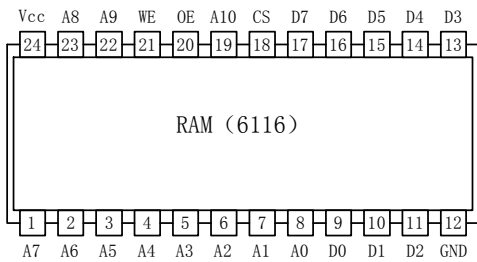
输入			输出 Q
清除	时钟	D	
0	X	X	0
1	↑	1	1
1	↑	0	0
1	0	X	Q <sub>0</sub>

### 74LS374



输出控制	G	D	输出
0	↑	1	1
0	↑	0	0
0	0	X	Q <sub>0</sub>
1	X	X	Z

### SRAM 6116



CS	WE	OE	功能
1	X	X	不选择
0	1	0	读
0	0	1	写
0	0	0	写