

计算机组成原理 实验教程

计算机组成原理课程组
硬件实验中心
哈尔滨工业大学计算学部

2024.03

目录

第 1 章	运算器	1
1.1	基本运算器实验	1
1.2	阵列乘法器设计实验	8
第 2 章	存储系统	10
2.1	静态随机存储器实验	10
2.2	Cache 映射机制模拟实验	16
第 3 章	控制器	21
3.1	微程序控制器实验	21
3.2	CPU 与简单模型机设计实验	32
第 4 章	系统总线与总线接口	39
4.1	系统总线和具有基本输入输出功能的总线接口实验	39
4.2	运算器与总线接口协作实验	44

第 1 章 运算器

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。本章首先安排一个基本的运算器实验，了解运算器的基本结构，然后再设计一个加法器和一个乘法器。

1.1 基本运算器实验

- (1) 了解运算器的组成结构。
- (2) 基于数据通路图，观测并分析运算器的工作原理。
- (3) 基于信号时序图，观测并分析运算器的工作原理。

1.1.1 实验目的

1.1.2 实验设备

PC 机一台，TDX-CMX 实验系统一套。

1.1.3 实验原理

本实验的原理如图 1-1-1 所示。

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，要处理的数据存于寄存器 A 和寄存器 B，三个部件同时接受来自 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 和 CN 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。如果是影响进位的运算，还将置进位标志 FC，在运算结果输出前，置 ALU 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 1-1-2 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功能，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）对于循环右移功能，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，移位时使用符号扩展或是 0 填充，具体由相应的指令控制。使用另外的逻辑进行移位总量译码和符号判别。

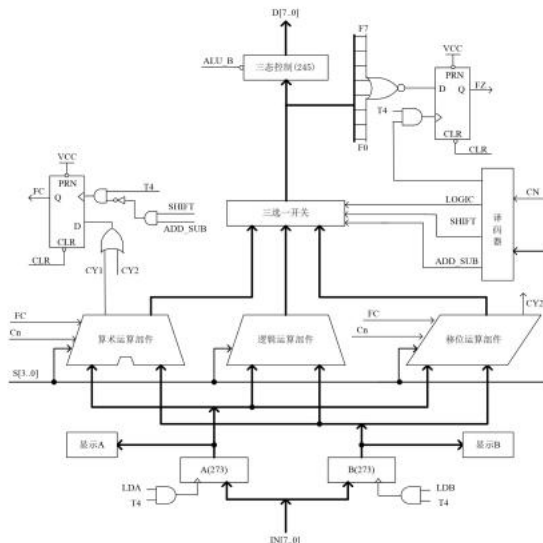


图 1-1-1 运算器原理图

ALU 的输入是通过 IN7~IN0 来引入的，而输出则是通过三态门 74LS245 已经连到 CPU 内总线上了，另外还有指示灯标明进位标志 FC 和零标志 FZ。请注意：实验箱上凡丝印标注有马蹄形标记 ‘ \sqcup ’，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 都连接至控制总线单元的 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，其余为高有效。

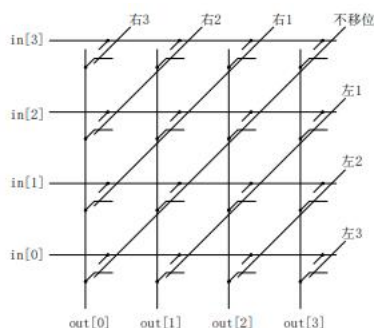


图 1-1-2 交叉开关桶形移位器原理图

寄存器 A 和寄存器 B 的数据能在 LED 灯上实时显示，原理如图 1-1-3 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。

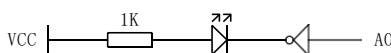


图 1-1-3 A0 显示原理图

ALU 和寄存器堆的连接如图 1-1-4 所示，这里的 OUT[7..0]也连接到了 CPU 内总线上。运算器的逻辑功能表如表 1-1-1 所示，其中 S3 S2 S1 S0 CN 为控制信号，FC 为进位标志，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。

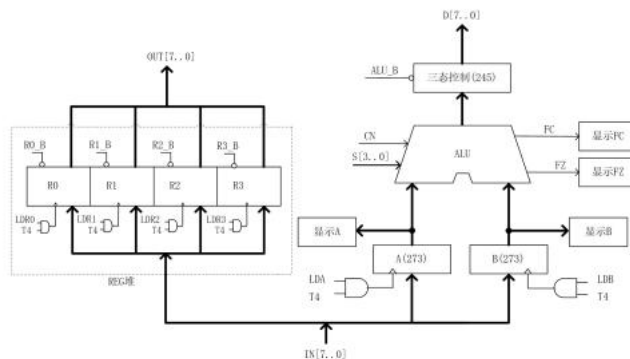


图 1-1-4 ALU 和外围电路连接原理图

表 1-1-1 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)
	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)

*表中“X”为任意态，下同

1.1.4 实验步骤

本实验支持两种方式运行：本机运行（不需电脑）和联机运行（需要电脑）。其中联机运行方式既支持数据通路图的观测，也支持信号时序图的观测。

一. 本机运行

（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 用短路块将 1、2 短接，按图 1-1-5 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（其它实验相同）。

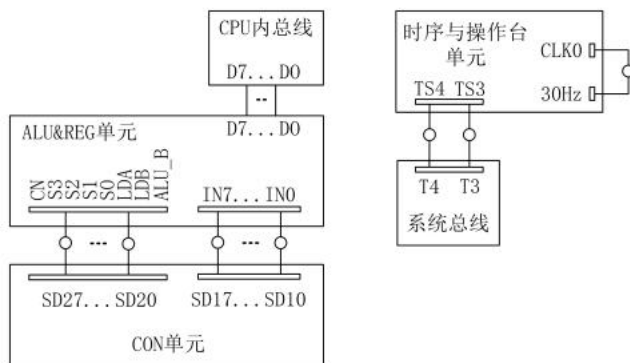


图 1-1-5 实验接线图

（2）将时序与操作台单元的开关 KK2 置为‘单拍’档，开关 KK1、KK3 置为‘运行’档。

（3）打开电源，如果听到有‘嘀’报警声，说明有总线竞争，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）用输入开关向寄存器 A 置数。

按动 2 次时序单元的 ST 按钮，产生 T1、T2 节拍后，拨动 CON 单元的 SD17...SD10 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’。置 LDA=1，LDB=0，按动 2 次 ST 按钮产生 T3、T4 节拍，则将二进制数 01100101 置入寄存器 A 中，寄存器 A 的值通过 ALU 单元的 A7...A0 八位 LED 灯显示。

（5）用输入开关向寄存器 B 置数。

按动 2 次时序单元的 ST 按钮，产生 T1、T2 节拍后，拨动 CON 单元的 SD17...SD10 数据开关，形成二进制数 10100111（或其它数值）。置 LDA=0，LDB=1，按动 2 次 ST 按钮产生 T3、T4 节拍，则将二进制数 10100111 置入寄存器 B 中，寄存器 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。

按动 2 次时序单元的 ST 按钮，产生 T1、T2 节拍后，置 ALU_B=0、LDA=0、LDB=0，然后按表 1-1-1 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 1001，运算器作加法运算，置 S3、S2、S1、S0 为 0010，运算器作逻辑与运算。按动 2 次 ST 按钮产生 T3、T4 节拍，观察 FC、FZ 标志位变化。

二. 联机运行

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 1），也可通过软件中的信号时序图来观测实验结果。

(1) 观测数据通路图

打开 TDX-CMX 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 1-1-6 所示。

操作方法同本机运行，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】—【单节拍】”，其作用相当于将时序单元的状态开关 KK2 置为‘单拍’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 1-1-2。然后改变 A、B 的值，验证 FC、FZ 的锁存功能。点击联机软件的“【回放】—【保存...】”按钮，可保存数据通路图的实验过程。

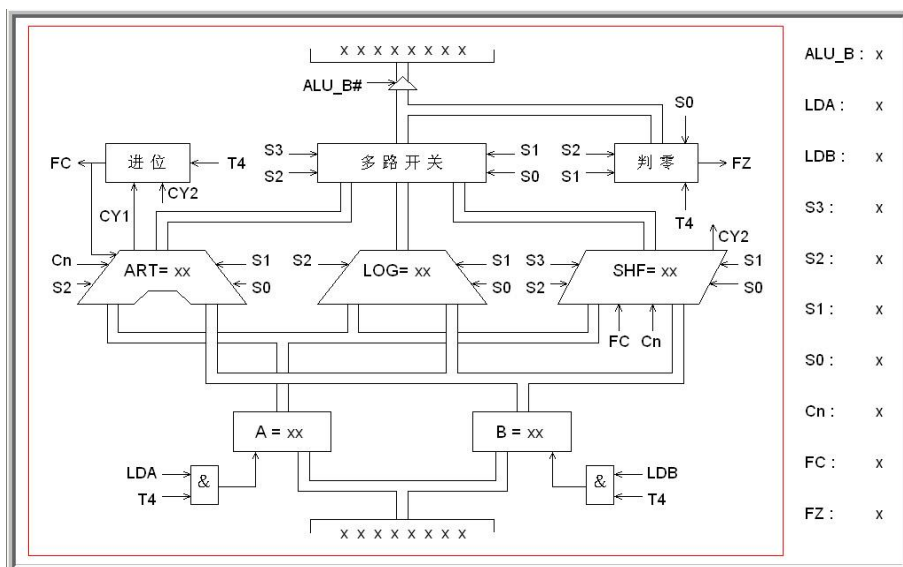



图 1-1-6 数据通路图

表 1-1-2 运算结果表

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

(2) 观测信号时序图

打开 TDX-CMX 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。再点击  打开选择观察信号窗口，或者选择联机软件的“【调试】—【时序观测窗】”，选择想要观察的信号，如图 1-1-7，点击确定。

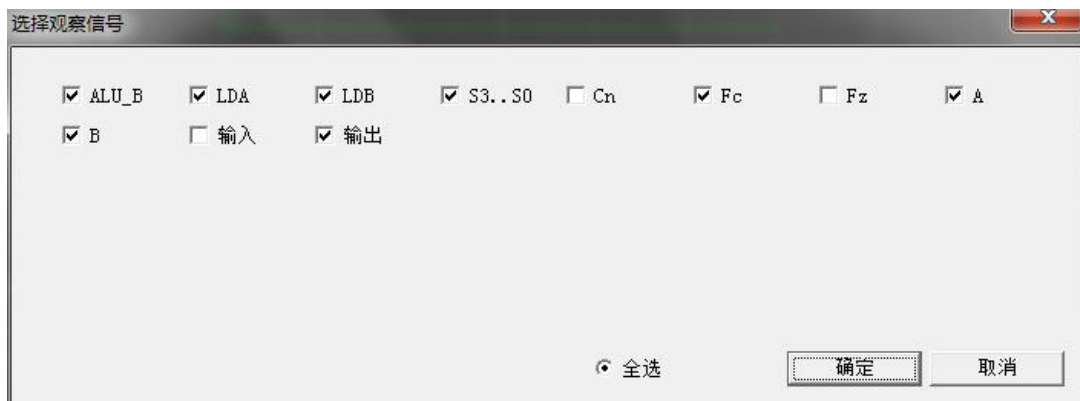


图 1-1-7 选择观察信号

弹出时序观测窗，操作方法同本机运行，可得到如下图 1-1-8 所示的时序图。

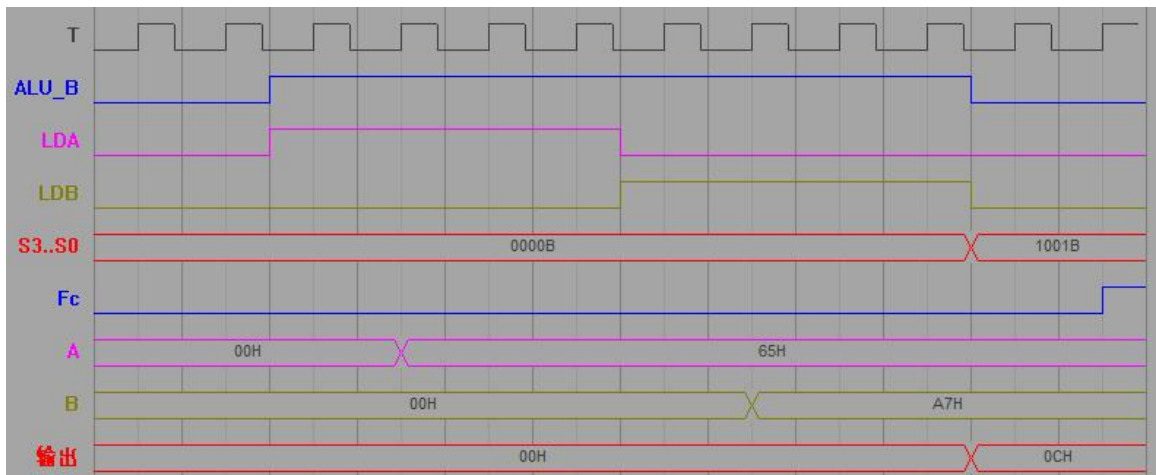


图 1-1-8 观察信号时序

观察上图，可知寄存器 A 的控制信号 LDA 在第一个机器周期的 T2 节拍后已经有效，但是寄存器 A 的数据在 T4 节拍上升沿才改变为 65H，说明寄存器 A 的输入是时序逻辑，受 T4 节拍控制。寄存器 B 同理。运算方式选择 S3..S0 在第三个机器周期 T2 节拍后被设置为算术加法运算，进位标志 FC 在 T4 节拍上升沿才改变，说明进位标志 FC 是时序逻辑，受 T4 节拍影响。运算器的输出在第三个机器周期 T2 节拍结束后 T3 节拍来之前 ALU_B 变有效后直接输出结果，可知运算器的输出是组合逻辑，只受 ALU_B 影响。右键点击保存按钮可将时序观测窗结果保存为图片格式。

思考题：将 A=01H 和 B=02H 进行逻辑与运算，观察运算器零标志 FZ 的时序，是否和进位标志 FC 一致？

1.2 阵列乘法器设计实验

1.2.1 实验目的

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

1.2.2 实验设备

PC 机一台，TDX-CMX 实验系统一套。

1.2.3 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：

$$\begin{array}{r}
 \times \qquad \qquad \qquad \begin{array}{cccc} A3 & A2 & A1 & A0 \\ B3 & B2 & B1 & B0 \end{array} \\
 \hline
 \qquad \qquad \qquad \begin{array}{cccc} A3B0 & A2B0 & A1B0 & A0B0 \\ A3B1 & A2B1 & A1B1 & A0B1 \\ A3B2 & A2B2 & A1B2 & A0B2 \\ + \quad A3B3 & A2B3 & A1B3 & A0B3 \end{array} \\
 \hline
 \begin{array}{cccccccc} & & & & P7 & P6 & P5 & P4 & P3 & P2 & P1 & P0 \end{array}
 \end{array}$$

这个 4×4 阵列乘法器的原理如图 1-3-1 所示。

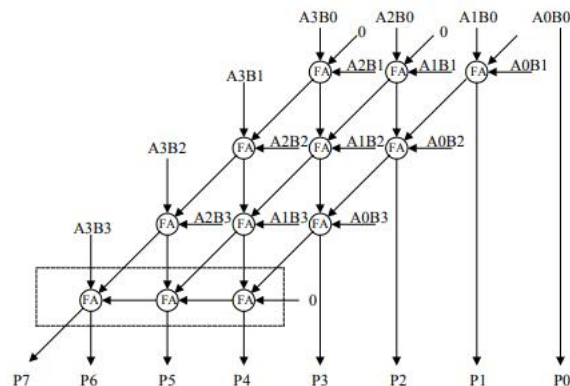


图 1-3-1 4×4 阵列乘法器原理图

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 FPGA 来设计一个 4×4 位加法器，且全部采用原理图方式实现。

1.2.4 实验步骤

- （1）根据上述阵列乘法器的原理，使用 Quartus 软件编辑相应的电路原理图并进行编译，其在 FPGA 芯片中对应的引脚如图 1-3-2 所示，框外文字表示连线标号，框内文字表示该引脚的含义（本实验例程见‘安装路径\FPGA\Multiply\Multiply.qpf’工程）。

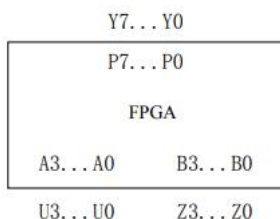


图 1-3-2 引脚分配图

- （2）关闭实验系统电源，按图 1-3-3 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，将下载电缆插入扩展单元的 E_JTAG 口，把生成的 SOF 文件下载到扩展单元中去，扩展单元介绍见实验 1.2。

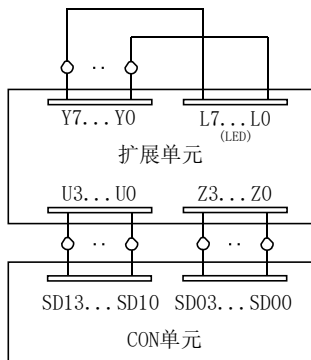


图 1-3-3 阵列乘法器实验接线图

- （4）以 CON 单元中的 SD10...SD13 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，而相乘的结果在扩展单元的 L7...L0 八个 LED 灯显示。给 A 和 B 置不同的数，观察相乘的结果。

第2章 存储系统

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。

本章安排了两个实验：静态随机存储器实验及 Cache 映射机制模拟实验。

2.1 静态随机存储器实验

2.1.1 实验目的

1. 掌握静态随机存储器 RAM 工作特性及数据的读写方法。
2. 基于信号时序图，了解读写静态随机存储器的原理。

2.1.2 实验设备

PC 机一台，TDX-CMX 实验系统一套。

2.1.3 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 2-1-1 所示。6116 有三个控制线：CS 片选线、OE 读线、WE 写线，其功能如表 2-1-1 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。

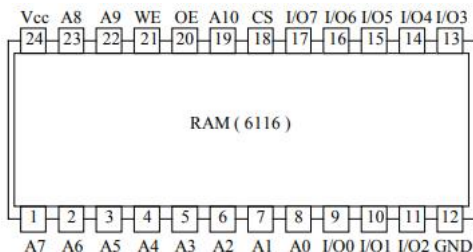


图 2-1-1 SRAM 6116 引脚图

表 2-1-1 SRAM 6116 功能表

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

实验原理图如图 2-1-2 所示，存储器数据线接至 CPU 内总线，内总线上接有 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，内嵌于 ABI 单元）给出。数据开关（位于 CON 单元的 SD17..SD10）经一个三态门（74LS245）连至 CPU 内总线，分时给出地址和数据。地址寄存器为 8 位，接入存储器的地址 A7...A0，高三位地址 A10...A8 接地，所以其实际容量为 256 字节。

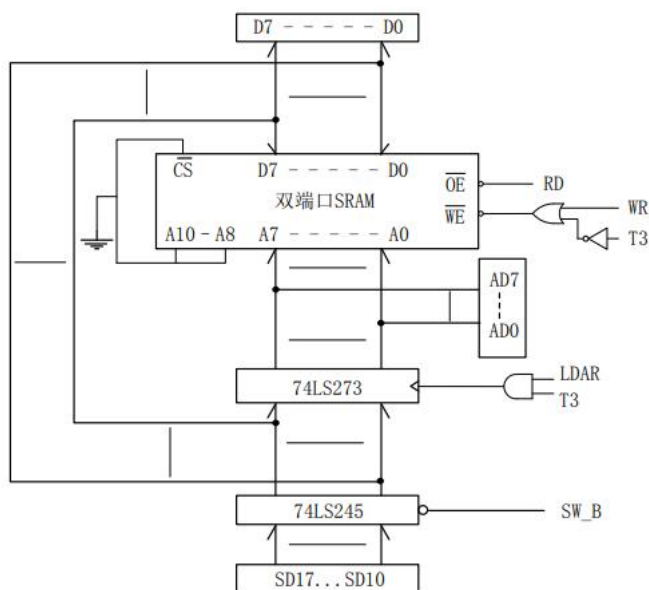


图 2-1-2 存储器实验原理图

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 单元的对应二进制开关模拟给出，其中 RD、WR 低有效，SW_B 低有效，LDAR 高有效。

2.1.4 实验步骤

本实验支持两种方式运行：本机运行（不需电脑）和联机运行（需要电脑）。

其中联机运行方式既支持数据通路图的观测，也支持信号时序图的观测。

一. 本机运行

(1) 关闭实验系统电源，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，JP2 用短路块将 1、2 短接，按图 2-1-3 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 KK1、KK3 置为运行档、开关 KK2 置为‘单拍’档（时序单元的介绍见附录二）

(3) 将 CON 单元的 K7 开关（SW_B）置为 1（使 SD17..SD10 开关组无输出）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。

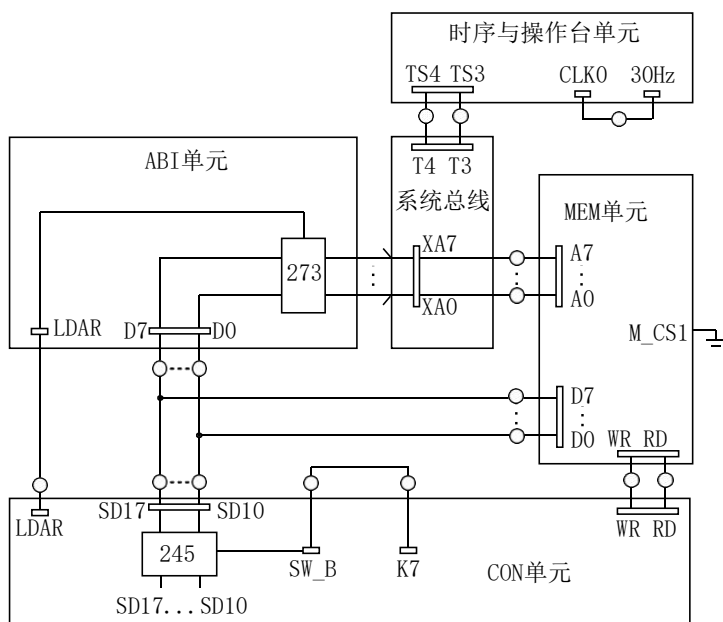


图 2-1-3 实验接线图

④ 给存储器的 00H、01H 地址单元中分别写入数据 11H、12H。由前面的存储器实验原理图（图 2-1-2）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：按动 2 次时序单元的 ST 按钮，产生 T1、T2 节拍后，先关掉存储器的读写（WR=1，RD=1），开关 SD17..SD10 输出地址 00H（SD17..SD10=0000 0000B，K7=0），然后打开地址寄存器门控信号（LDAR=1），按动 1 次 ST 产生 T3 脉冲，即将地址 00H 写入到 AR 中，按动 1 次 ST 产生 T4 脉冲，第 1 个机器周期结束。

再写数据：按动 2 次时序单元的 ST 按钮，产生 T1、T2 节拍后，先关掉地址寄存器门控信号（LDAR=0），数据开关输出要写入的数据 11H（SD17..SD10 = 0001 0001B），打开三态门（K7=0），然后使存储器处于写状态（WR=0，RD=1），按动 1 次 ST 产生 T3 脉冲，即将数据

11H 写入到存储器 00H 地址中，按动 1 次 ST 产生 T4 脉冲，第 2 个机器周期结束。

重复上述操作，继续向 01H 地址单元中写入数据 12H。

写存储器的流程如图 2-1-4 所示（以向 00 地址单元写入 11H 为例）：

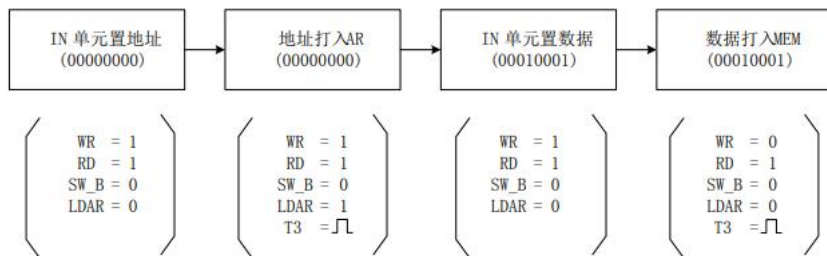


图 2-1-4 写存储器流程图

⑤ 读出 00H 地址单元中的内容，观察单元中的内容是否与前面写入的一致。

先写地址：按动 2 次时序单元的 ST 按钮，产生 T1、T2 节拍后，先关掉存储器的读写（WR=1，RD=1），开关 SD17..SD10 输出地址 00H（SD17..SD10 = 0000 0000B，K7=0），然后打开地址寄存器门控信号（LDAR=1），按动 1 次 ST 产生 T3 脉冲，即将地址 00H 写入到 AR 中，按动 1 次 ST 产生 T4 脉冲，一个机器周期结束。

再读数据：按动 2 次时序单元的 ST 按钮，产生 T1、T2 节拍后，先关掉地址寄存器门控信号（LDAR=0），关闭 IN 单元的输出（SW_B=1），然后使存储器处于读状态（WR=1，RD=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。按动 2 次 ST 产生 T3、T4 脉冲，一个机器周期结束。

读存储器的流程如图 2-1-5 所示（以从 00 地址单元读出 11H 为例）

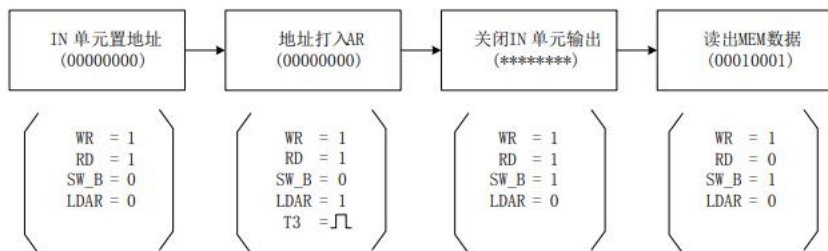


图 2-1-5 读存储器流程图

二. 联机运行

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 1），也可通过软件中的信号时序图来观测实验结果。

(1) 观测数据通路图

打开 TDX-CMX 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 2-1-6 所示。

操作方法同本机运行，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】—【单节拍】”，其作用相当于将时序单元的状态开关置为‘单拍’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。

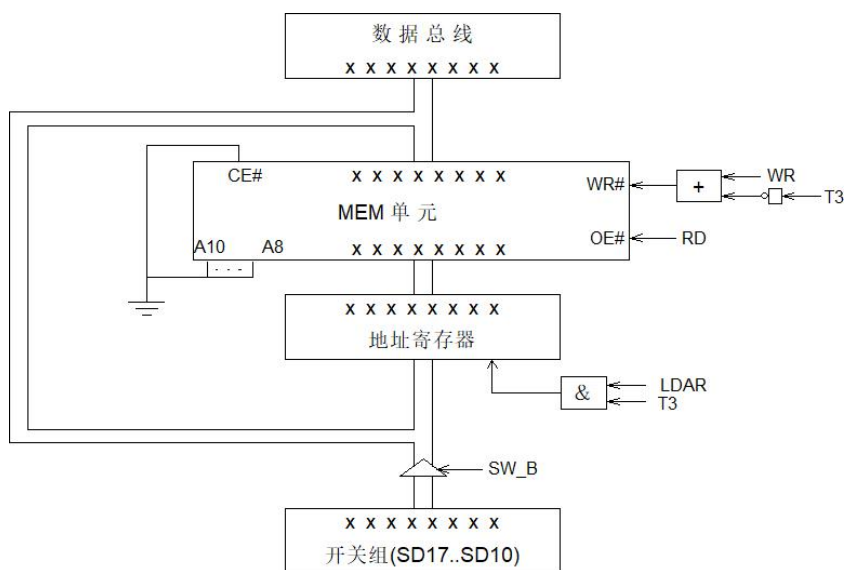


图 2-1-6 数据通路图

(2) 观测信号时序图


打开存储器实验的数据通路图。再点击  打开选择观察信号窗口，或者选择联机软件的“【调试】—【时序观测窗】”，选择想要观察的信号，如图 2-1-7，点击确定。



图 2-1-7 选择观察信号

弹出时序观测窗，操作方法同本机运行，可得到如下图 2-1-8 所示的时序图。

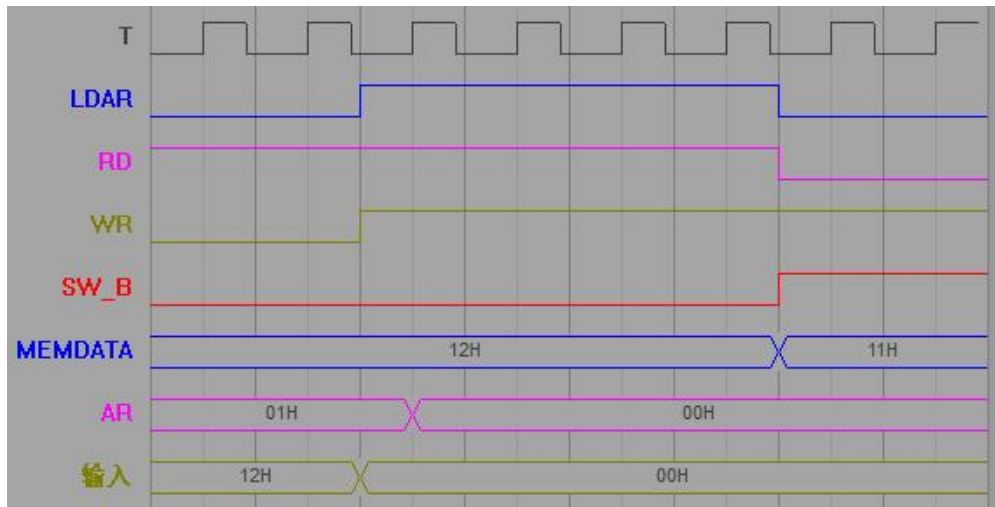


图 2-1-8 观察信号时序

观察上图，可知最后一个机器周期的 T2 节拍后，RD 有效的同时存储器输出 11H，说明读存储器受 RD 信号影响。观察倒数第二个机器周期的 T2 节拍后，地址寄存器门控信号 LDAR 有效，同时开关 SD17..SD10 已经改为 00H 地址，但是地址寄存器 AR 中的地址直到 T3 时刻上升沿才发生改变，说明地址寄存器 AR 的写入受 T3 上升沿影响。

2.2 Cache 映射机制模拟实验

2.2.1 实验目的

- (1) 掌握 Cache 的原理及其设计方法。
- (2) 熟悉 FPGA 应用设计及 EDA 软件的使用。

2.2.2 实验设备

PC 机一台，TDX-CMX 实验系统一套。

2.2.3 实验原理

本实验采用的地址变换是直接映象方式，这种变换方式简单而直接，硬件实现很简单，访问速度也比较快，但是块的冲突率比较高。其主要原则是：主存中一块只能映象到 Cache 的一个特定的块中。

假设主存的块号为 B ，Cache 的块号为 b ，则它们之间的映象关系可以表示为：

$$b = B \bmod C_b$$

其中， C_b 是 Cache 的块容量。

设主存的块容量为 M_b ，区容量为 M_c ，则直接映象方法的关系如图 2-2-1 所示。把主存按 Cache 的大小分成区，一般主存容量为 Cache 容量的整数倍，主存每一个分区内的块数与 Cache 的总块数相等。直接映象方式只能把主存各个区中相对块号相同的那些块映象到 Cache 中同一块号的那个特定块中。例如，主存的块 0 只能映象到 Cache 的块 0 中，主存的块 1 只能映象到 Cache 的块 1 中，同样主存区 1 中的块 C_b （在区 1 中的相对块号是 0）也只能映象到 Cache 的块 0 中。根据上面给出的地址映象规则，整个 Cache 地址与主存地址的低位部分是完全相同的。

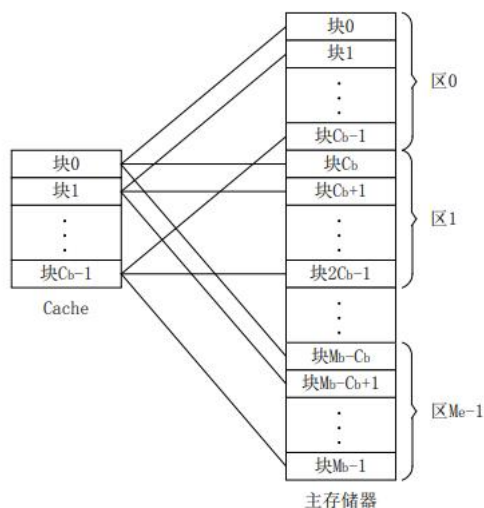


图 2-2-1 直接相联映象方式

直接映象方式的地址变换过程如图 2-2-2 所示，主存地址中的块号 B 与 Cache 地址中的块号 b 是完全相同的。同样，主存地址中的块内地址 W 与 Cache 地址中的块内地址 w 也是完全相同的，主存地址比 Cache 地址长出来的部分称为区号 E。

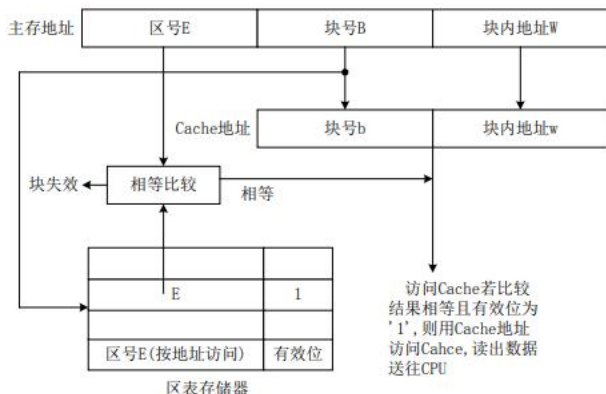


图 2-2-2 直接相联地址变换

在程序执行过程中，当要访问 Cache 时，为了实现主存块号到 Cache 块号的变换，需要有一个存放主存区号的小容量存储器，这个存储器的容量与 Cache 的块数相等，字长为主存地址中区号 E 的长度，另外再加一个有效位。

在主存地址到 Cache 地址的变换过程中，首先用主存地址中的块号去访问区号存储器（按地址访问）。把读出来的区号与主存地址中的区号 E 进行比较，根据比较结果以及与区号在同一存储字中的有效位情况作出处理。如果区号比较结果相等，有效位为‘1’，则 Cache 命中，表示要访问的那一块已经装入到 Cache 中了，这时 Cache 地址（与主存地址的低位部分完全相同）是正确的。用这个 Cache 地址去访问 Cache，把读出来的数据送往 CPU。其他情况均为 Cache 没有命中，或称为 Cache 失效，表示要访问的那个块还没有装入到 Cache 中，这时，要用主存地址去访问主存储器，先把该地址所在的块读到 Cache 中，然后 CPU 从 Cache 中读取该地址中的数据。

本实验要在 FPGA 中实现 Cache 及其地址变换逻辑（也叫 Cache 控制器），采用直接相联地址变换，只考虑 CPU 从 Cache 读数据，不考虑 CPU 从主存中读数据和写回数据的情况，Cache 和 CPU 以及存储器的关系如图 2-2-3 所示。

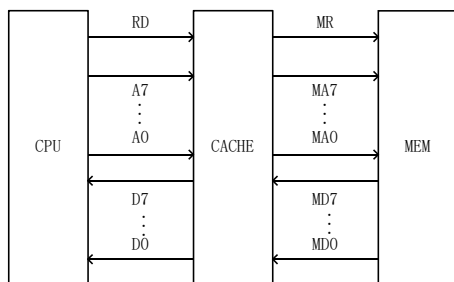


图 2-2-3 Cache 系统图

Cache 控制器顶层模块如图 2-2-4 所示，主存地址为 **A7...A0,共 8 位**，区号 **E 取 3 位**，这样Cache 地址还剩 5 位，所以 Cache容量为 32个单元，**块号B取 3位**，那么 Cache分为 8块，**块内地址 W 取 2位**，则每块为 4 个单元。图 2-2-4 中，WCT为写 Cache 块表信号，CLR 为系统总清零信号，A7...A0 为 CPU 访问内存的地址，M 为 Cache失效信号，CA4...CA0 为 Cache地址，MD7...MD0 为主存送 Cache的数据，D7...D0 为 Cache送 CPU数据，T2 为系统时钟，RD 为CPU 访问内存读信号，LA1和 LA0为块内地址。

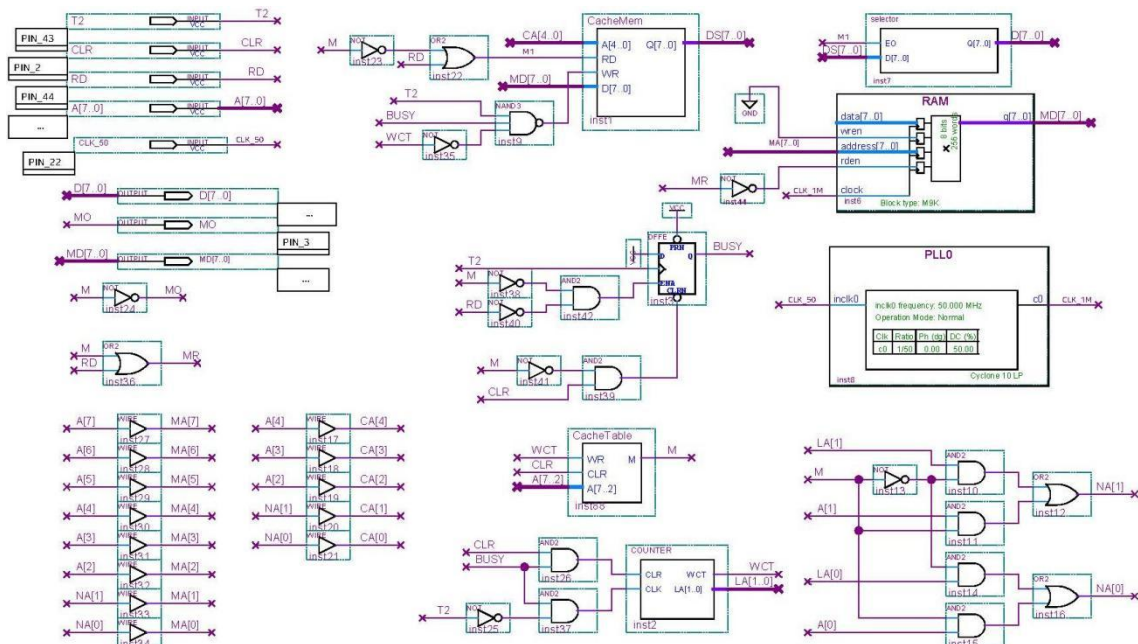


图 2-2-4 Cache 控制器顶层模块图

在 Quartus 软件中先调用一个 8 位的 SRAM 的 IP 核，编写一个 MIF 文件用来存储数据，然后实现一个 8 位的存储单元（见例程中的 MemCell.bdf），然后用这个 8 位的存储单元来构成一个 32×8 位的 Cache（见例程中的 CacheMem.bdf），这样就实现了 Cache 的存储体。

再实现一个 4 位的存储单元（见例程中的 TableCell.bdf），然后用这个 4 位的存储单元来构成一个 8×4 位的区表存储器，用来存放区号和有效位（见例程中的 CacheTable.bdf），在这个文件中，还实现了一个区号比较器，如果主存地址的区号 E 和区表中相应单元中的区号相等，且有效位为 1，则 Cache 命中，否则 Cache 失效，标志为 M，M=0 时表示 Cache 失效。

当 Cache 命中时，就将 Cache 存储体中相应单元的数据送往 CPU，这个过程比较简单。当 Cache 失效时，就将主存中相应块中的数据读出写入 Cache 中，这样 Cache 控制器就要产生访问主存储器的地址和主存储器的读信号，由于每块占四个单元，所以需要连续访问四次主存，这就需要有一个低地址发生器，即一个 2 位计数器（见例程中的 Counter.vhd），将低 2 位和 CPU 给出的高 6 位地址组合起来，形成访问主存储器的地址。M 就可以做为主存的读信号，这样在时钟的控制下，就可以将主存中相应的块写入到 Cache 的相应块中，最后再修改区表（见例程中的 CacheCtrl.bdf）。

2.2.4 实验步骤

(1) 用 Quartus 软件编辑实现相应的逻辑并进行编译，直到编译通过，Cache控制器在FPGA芯片中对应的引脚如图 2-2-5所示，框外文字表示连接标号，框内文字表示该引脚的含义。

(本实验例程见 ‘安装路径\FPGA\CacheCtrl\CacheCtrl.qpf’ 工程)。

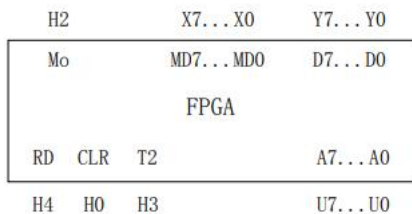


图 2-2-5 引脚分配图

- (2) 关闭实验系统电源，按图 2-2-6 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，将下载电缆插入扩展单元的 E_JTAG口，把生成的 SOF文件下载到扩展单元中的 FPGA中，扩展单元介绍见实验 1.2。
- (4) 将时序与操作台单元的开关 KK3 置为 ‘运行’ 档，CLR 信号由 CON 单元的 CLR 模拟给出，按动 CON 单元的 CLR 按钮，清空区表。
- (5) 预先往主存写入数据：存储器已经提前装载好了数据文件 (CacheCtrl.hex)，用户也可以自己改写内容。

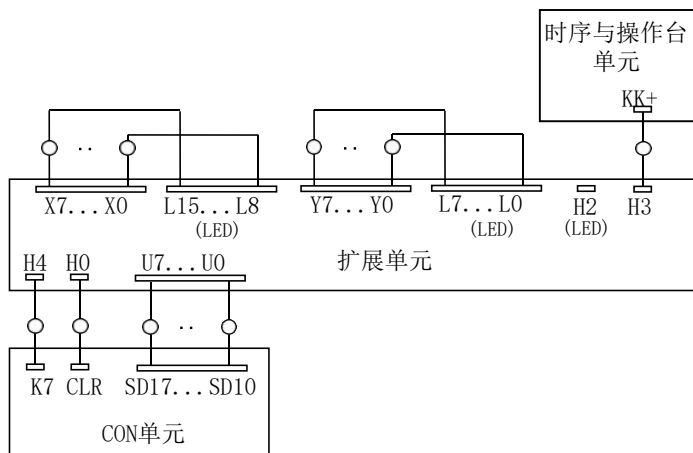


图 2-2-6 实验接线图

(6) CPU访问主存地址由 CON单元的 SD17...SD10 模拟给出，如 0000 0001。CPU访问主存的读信号由 CON单元的K7模拟给出，置 K7为低，可以观察到扩展单元上的H2指示灯亮，L7...L0指示灯灭，表示 Cache 失效。此时按动 KK 按钮四次，注意 L15...L8 指示灯的变化情况，地址会依次加一，L15...L8 指示灯上显示的是当前主存数据，按动四次 KK 按钮后，H2 指示灯变灭，L7...L0上显示的值即为 Cache 送往 CPU 的数据。

(7) 重新给出主存访问地址，如 00000011，H2 指示灯变灭，表示 Cache 命中，说明第 0 块数据已写入 Cache。

(8) 重新给出大于 03H 地址，体会 Cache 控制器的工作过程。

第 3 章 控制器

3.1 微程序控制器实验

3.1.1 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。
- (3) 基于数据通路图，掌握微程序控制器的工作原理。
- (4) 基于微程序流程图，掌握微程序控制器的工作原理。
- (5) 基于信号时序图，掌握微程序控制器的工作原理。

3.1.2 实验设备

PC 机一台，TDX-CMX 实验系统一套。

3.1.3 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 3-1-1 所示。

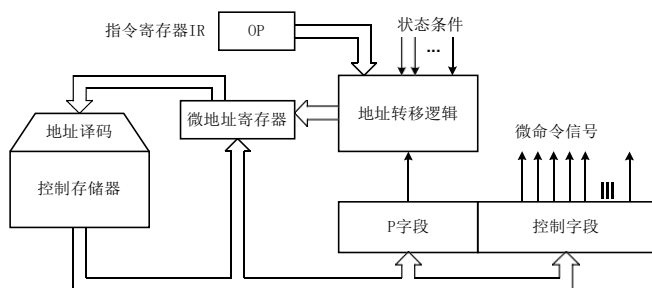
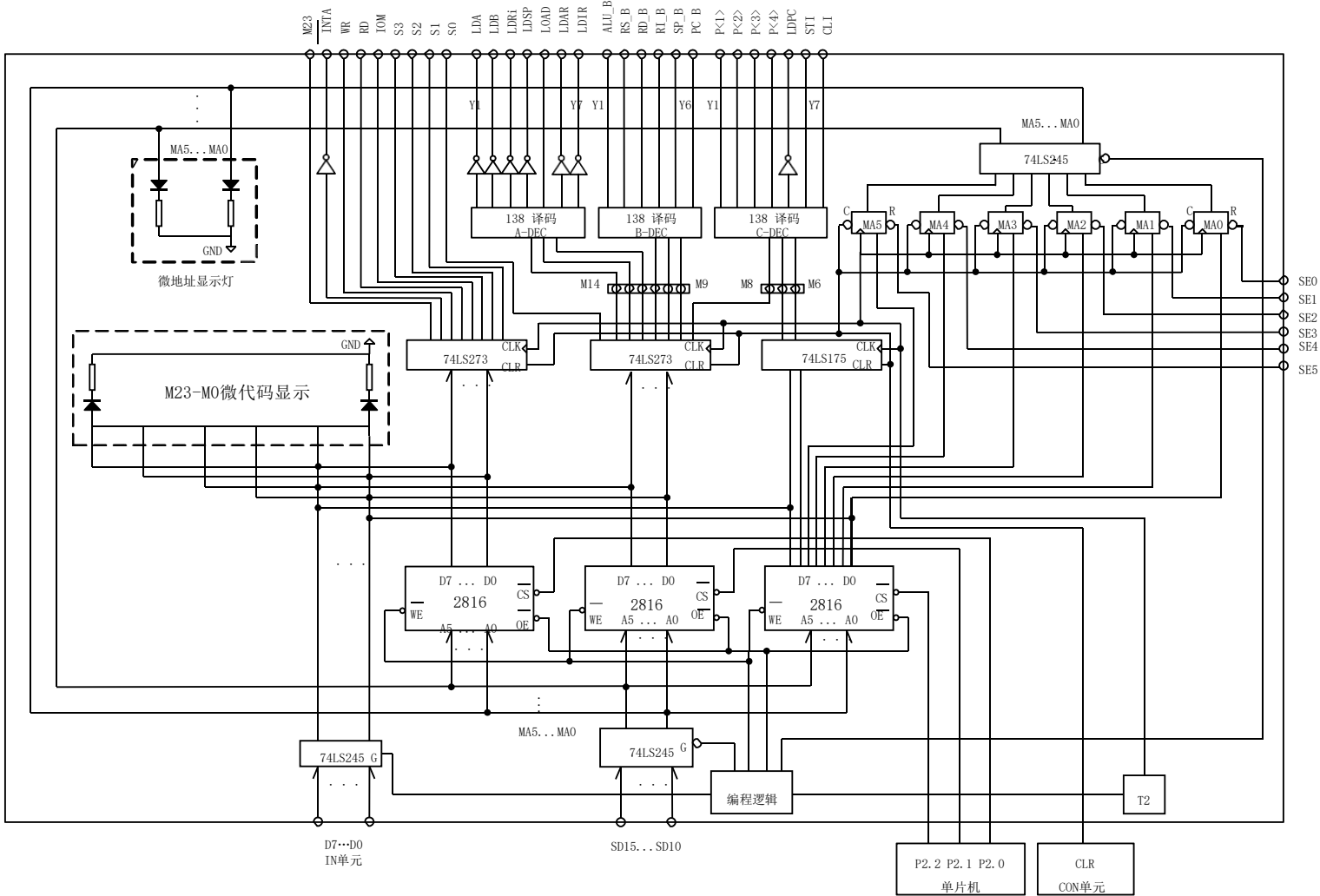


图 3-1-1 微程序控制器组成原理框图

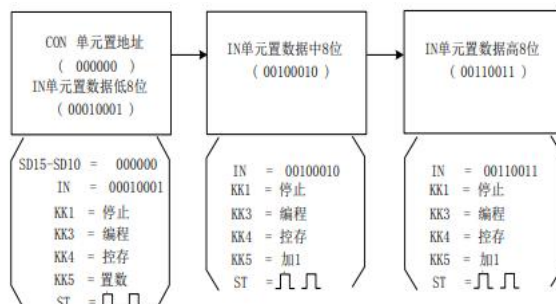
控制器是严格按照系统时序来工作的，因而时序控制对于控制器的设计是非常重要的，从前面的实验可以很清楚地了解时序电路的工作原理，本实验所用的时序由时序单元来提供，分为四拍 TS1、TS2、TS3、TS4，时序单元的介绍见附录 2。

微程序控制器的组成见图 3-1-2，其中控制存储器采用 3 片 E²PROM，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 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。

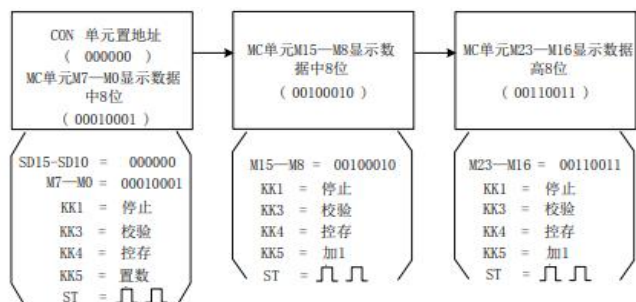
图 3-1-2 微程序控制器原理图



在实验平台中设有一组编程控制开关 KK3、KK4、KK5（位于时序与操作台单元），可实现对存储器（包括存储器和控制存储器）的三种操作：编程、校验、运行。考虑到对于存储器（包括存储器和控制存储器）的操作大多集中在一个地址连续的存储空间中，实验平台提供了便利的手动操作方式。以向 00H 单元中写入 332211 为例，对于控制存储器进行编辑的具体操作步骤如下：首先将 KK1 拨至‘停止’档、KK3 拨至‘编程’档、KK4 拨至‘控存’档、KK5 拨至‘置数’档，由 CON 单元的 SD15——SD10 开关给出需要编辑的控存单元首地址（000000），IN 单元开关给出该控存单元数据的低 8 位（00010001），连续两次按动时序与操作台单元的开关 ST（第一次按动后 MC 单元低 8 位显示该单元以前存储的数据，第二次按动后显示当前改动的数据），此时 MC 单元的指示灯 MA5——MA0 显示当前地址（000000），M7——M0 显示当前数据（00010001）。然后将 KK5 拨至‘加1’档 IN 单元开关给出该控存单元数据的中 8 位（00100010），连续两次按动开关 ST，完成对该控存单元中 8 位数据的修改，此时 MC 单元的指示灯 MA5——MA0 显示当前地址（000000），M15——M8 显示当前数据（00100010）；再由 IN 单元开关给出该控存单元数据的高 8 位（00110011），连续两次按动开关 ST，完成对该控存单元高 8 位数据的修改此时 MC 单元的指示灯 MA5——MA0 显示当前地址（000000），M23——M16 显示当前数据（00110011）。此时被编辑的控存单元地址会自动加 1（01H），由 IN 单元开关依次给出该控存单元数据的低 8 位、中 8 位和高 8 位配合每次开关 ST 的两次按动，即可完成对后续单元的编辑。



编辑完成后需进行校验，以确保编辑的正确。以校验 00H 单元为例，对于控制存储器进行校验的具体操作步骤如下：首先将 KK1 拨至‘停止’档、KK3 拨至‘校验’档、KK4 拨至‘控存’档、KK5 拨至‘置数’档。由 CON 单元的 SD15——SD10 开关给出需要校验的控存单元地址（000000），连续两次按动开关 ST，MC 单元指示灯 M7——M0 显示该单元低 8 位数据（00010001）；KK5 拨至‘加 1’档，再连续两次按动开关 ST，MC 单元指示灯 M15——M8 显示该单元中 8 位数据（00100010）；再连续两次按动开关 ST，MC 单元指示灯 M23——M16 显示该单元高 8 位数据（00110011）。再连续两次按动开关 ST，地址加 1，MC 单元指示灯 M7——M0 显示 01H 单元低 8 位数据。如校验的微指令出错，则返回输入操作，修改该单元的数据后再进行校验，直至确认输入的微代码全部准确无误为止，完成对微指令的输入。



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，则每次随着开关 ST 的两次拨动操作，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，微地址会自动减一，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 3-1-1：

表 3-1-1 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

14	13	12	选择
0	0	0	NOP
0	0	1	LDA
0	1	0	LDB
0	1	1	LDRO
1	0	0	保留
1	0	1	保留
1	1	0	保留
1	1	1	LDIR

B字段

11	10	9	选择
0	0	0	NOP
0	0	1	ALU_B
0	1	0	RO_B
0	1	1	保留
1	0	0	保留
1	0	1	保留
1	1	0	保留
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	保留
0	1	1	保留
1	0	0	保留
1	0	1	保留
1	1	0	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 3-1-3 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，指令译码逻辑在控制器单元的 INS_DEC 中实现。

从图 3-1-2 中也可以看出，微控器产生的控制信号比表 3-1-1 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 3-1-4 所示。IR 的原理图如图 4-1-5 所示，R0 单元原理如图 3-1-7 所示，OUT 单元的原理图见图 3-1-6 所示。

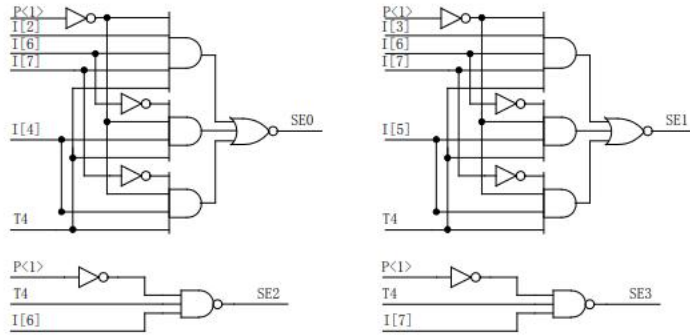


图 3-1-3 指令译码原理图

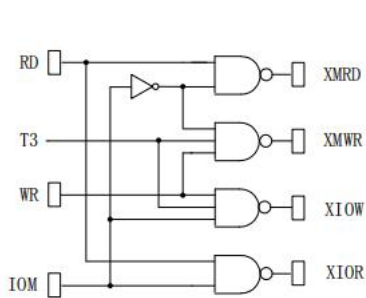


图 3-1-4 读写控制逻辑

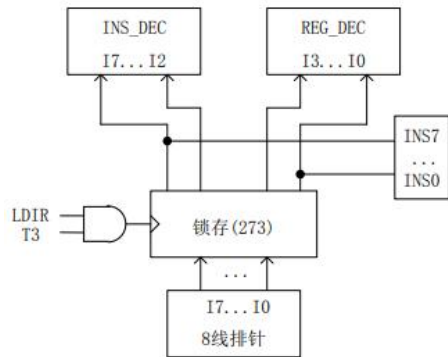


图 3-1-5 IR 原理图

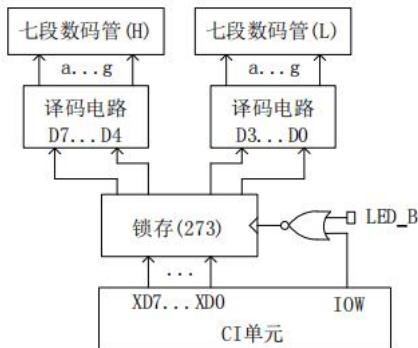


图 3-1-6 OUT 单元原理图

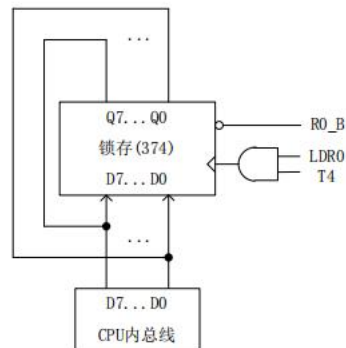


图 3-1-7 R0 原理图

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	$IN \rightarrow R0$
ADD	0000 0000	$R0 + R0 \rightarrow R0$
OUT	0011 0000	$R0 \rightarrow OUT$
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 3-1-8 所示。

几条机器指令对应的参考微程序流程图如图 3-1-9 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，右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。

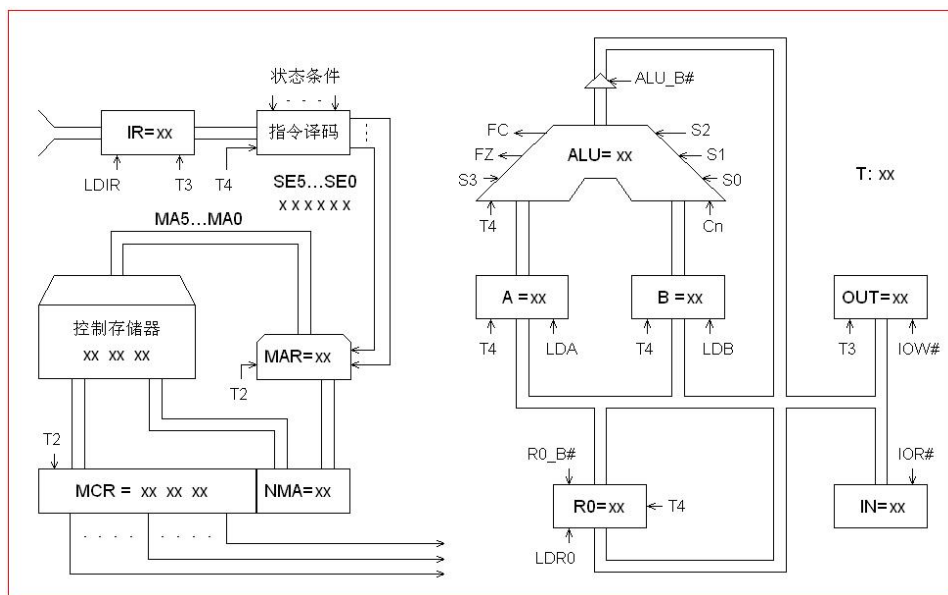


图 3-1-8 数据通路图

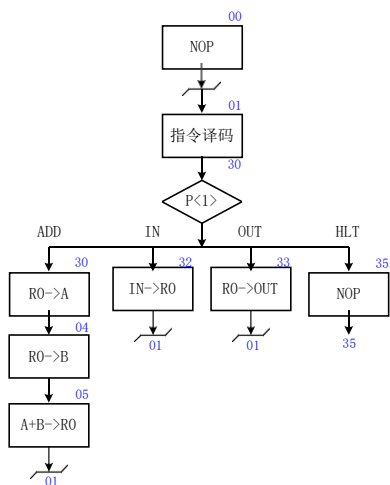


图 3-1-9 微程序流程图

将全部微程序按微指令格式变成二进制微代码，可得到表 3-1-2 的二进制代码表。

表 3-1-2 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

3.1.4 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 用短路块将 1、2 短接，按图 3-1-10 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。

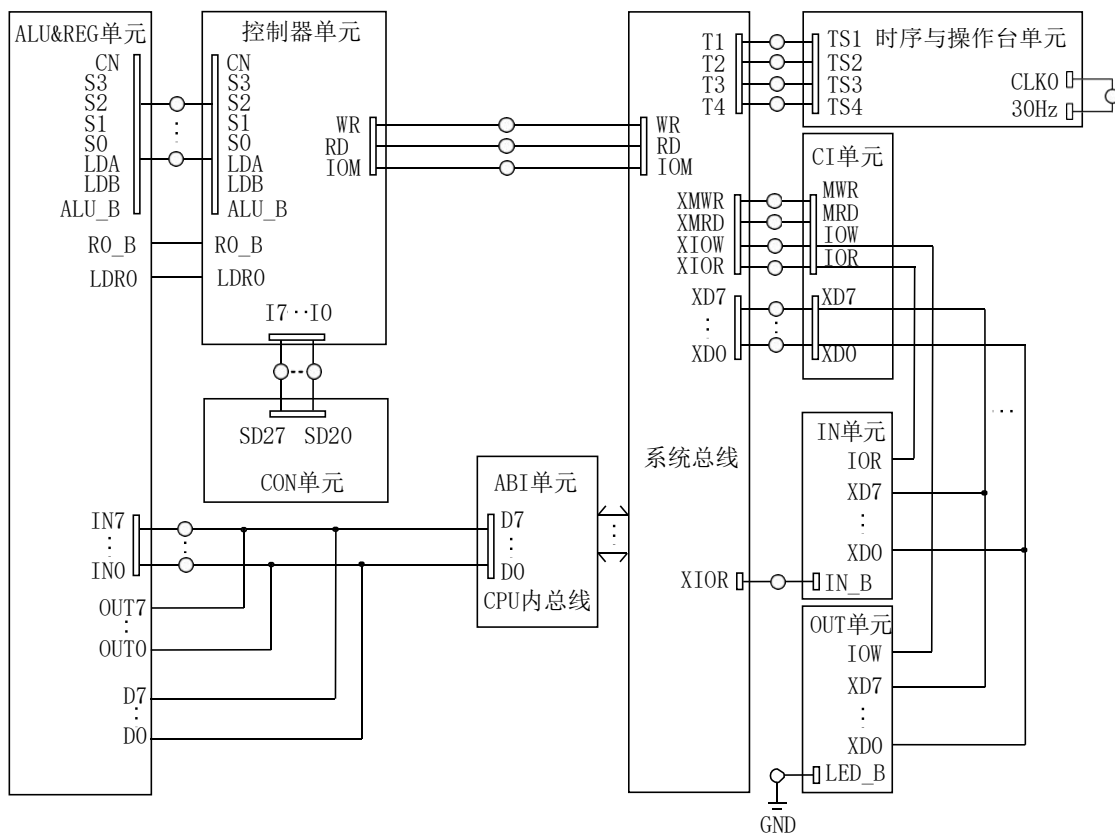


图 3-1-10 实验接线图

2. 对微控器进行读写操作，分两种情况：手动读写和联机读写。

1) 手动读写

(1) 手动对微控器进行编程（写）

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3-1-2的微代码写入 E2ROM 芯片中。

(2) 手动对微控器进行校验（读）

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 单元的指数数据指示灯 M15——M8 显示该单元的中 8 位，MC 单元的指数数据指示灯 M23——M16 显示该单元的高 8 位。

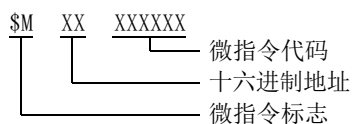
⑤ 重复①、②、③、④四步，完成对微代码的校验。如果校验出微代码写入错误，重新写入、校验，直至确认微指令的输入无误为止。

2) 联机读写

(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。

(2) 写入微程序

用联机软件的“【转储】—【装载】”功能将该格式 (*.TXT) 文件装载入实验系统。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) 校验微程序

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。检查微控器相应地址单元的数据是否和表 4-1-2 中的十六进制数据相同，如果不同，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微指令，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。

```
; //***** //
; // //
; //      微控器实验指令文件      //
; // //
; //      By TangDu CO.,LTD      //
; // //
; //***** //
; //**** Start Of MicroController Data **** //
$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
; //***** End Of MicroController Data *****//

```

3. 运行微程序

本实验支持两种方式运行：本机运行（不需电脑）和联机运行（需要电脑）。其中联机运行方式支持数据通路图的观测，支持微程序流图的观测，支持信号时序图的观测。

1) 本机运行

① 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 CLR 按钮，将微地址寄存器（MAR）清零，同时也将指令寄存器（IR）、ALU 单元的暂存器 A 和暂存器 B 清零。

② 将时序与操作台单元的开关 KK2 置为‘单拍’档，然后按动 ST 按钮，体会系统在 T1、T2、T3、T4 节拍中各做的工作。T2 节拍微控器将后续微地址（下条执行的微指令的地址）打入微地址寄存器，当前微指令打入微指令寄存器，并产生执行部件相应的控制信号；T3、T4 节拍根据 T2 节拍产生的控制信号做出相应的执行动作，如果测试位有效，还要根据机器指令及当前微地址寄存器中的内容进行译码，使微程序转入相应的微地址入口，实现微程序的分支。

③ 按动 CON 单元的 CLR 按钮，清微地址寄存器（MAR）等，并将时序与单元的开关 KK2 置为‘单拍’档。

④ 置 IN 单元数据为 00100011，按动 ST 按钮，当 MC 单元后续微地址显示为 000001 时，在 CON 单元的 SD27...SD20 模拟给出 IN 指令 00100000 并继续单步执行，当 MC 单元后续微地址显示为 000001 时，说明当前指令已执行完；在 CON 单元的 SD27...SD20 给出 ADD 指令 00000000，该指令将会在下个 T3 被打入指令寄存器（IR），它将 R0 中的数据和其自身相加后送 R0；接下来在 CON 单元的 SD27...SD20 给出 OUT 指令 00110000 并继续单步执行，在 MC 单元后续微地址显示为 000001 时，观察 OUT 单元的显示值是否为 01000110。

2) 联机运行

(1) 观测数据通路图

打开 TDX-CMX 软件，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 3-1-8 所示。


操作方法同本机运行，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转换的过程，这是计算机识别和执行指令的根基。

按本机运行的顺序给出数据和指令，观察最后的运算结果是否正确。

(2) 观测微程序流图

打开数据通路图后，点击“【调试】—【微程序流图】”，打开微程序流程图，如图 3-1-9 所示，操作方法同本机运行，跟踪显示每条机器指令的执行过程。

(3) 观测信号时序图

点击  打开选择观察信号窗口，或者选择联机软件的“【调试】—【时序观测窗】”，选

择想要观察的信号，如图 3-1-11，点击确定。

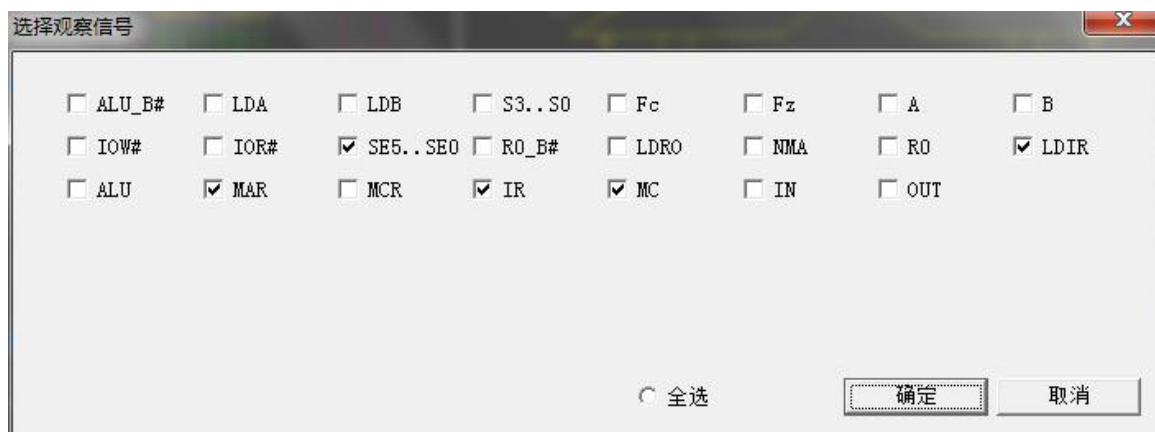


图 3-1-11 选择观察信号

弹出时序观测窗，如图 3-1-12 所示。

按动 CON 单元的 CLR 按钮，将时序与操作台单元的开关 KK2 置为‘单拍’档，然后按动4次 ST 按钮，此时序观测窗口上方 T 代表节拍，从左到右依次为 T1、T2、T3、T4，观察 T2 时刻上升沿时，NMA 中地址 01H 送入 MAR，MC 显示 MAR 地址对应的值。



图 3-1-12 时序观测图

再按动 4 次 ST 按钮，观察第二个机器周期 T2 时刻上升沿时，NMA 中地址 30H 送入 MAR，MC 显示 MAR 地址对应的值，MCR 为上一机器周期执行的微指令 007070H，该微指令使 LDIR 有效；T3 时刻上升沿 CON 单元 SD27~SD20 输入的机器指令 20H 被打入指令寄存器 IR；T4 时刻上升沿，机器指令译码 SE5...SE0 变更为 3DH，所以 MAR 变为地址 32H，将要执行的微指令 MC 变为 183001H。由此可知第二个机器周期执行了微指令 007070H，完成了机器指令 20H 的译码。再按动 4 次 ST 按钮，观察 IN 单元数据被打入寄存器 R0 中。

思考题：机器指令 30H(即 OUT 指令)执行时，观察寄存器 R0 中的数据何时送入 OUT 单元，是组合逻辑还是时序逻辑？受哪些信号影响？

3.2 CPU 与简单模型机设计实验

3.2.1 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。

3.2.2 实验设备

PC 机一台，TDX-CMX 实验系统一套。

3.2.3 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 3-2-1 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。

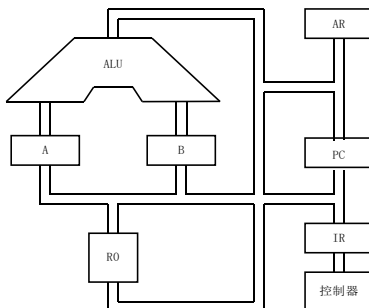


图 3-2-1 基本 CPU 构成原理图

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 ABI 单元的 FPGA 器件中。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。

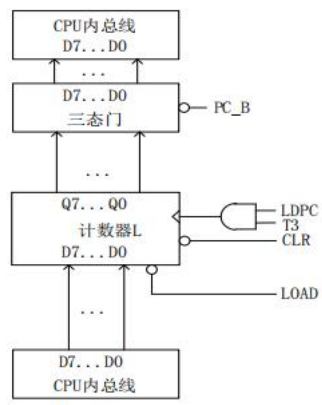


图 3-2-2 程序计数器(PC)原理图

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，可以设计数据通路图，如图 3-2-3 所示。

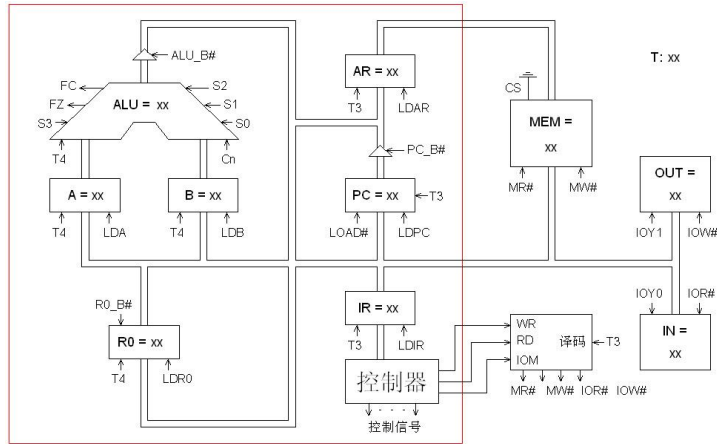


图 3-2-3 数据通路图

本实验在微程序控制器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 3-2-1 所示。

表 3-2-1 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

14	13	12	选择
0	0	0	NOP
0	0	1	LDA
0	1	0	LDB
0	1	1	LDRO
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	RO_B
0	1	1	保留
1	0	0	保留
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	保留
0	1	1	保留
1	0	0	保留
1	0	1	LDPC
1	1	0	保留
1	1	1	保留

系统涉及到的微程序流程见图 3-2-4 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 3-1-3 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。

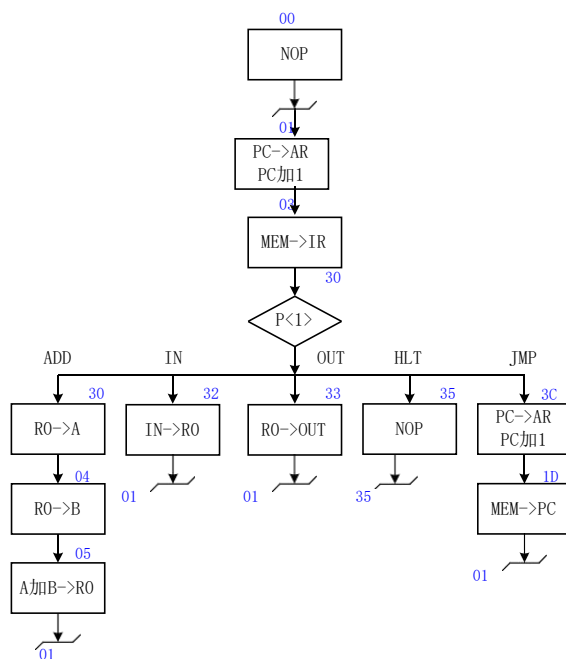


图 3-2-4 简单模型机微程序流程图

当全部微程序设计完毕后，应将每条微指令代码化，表 3-2-2 即为将图 3-2-4 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 3-2-2 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，地址和内容均为二进制数。

地 址	内 容	助记符	说 明
00000000	00100000	; START: IN R0	从 IN 单元读入数据送 R0
00000001	00000000	; ADD R0,R0	R0 和自身相加，结果送 R0
00000010	00110000	; OUT R0	R0 的值送 OUT 单元显示
00000011	11100000	; JMP START	跳转至 00H 地址
00000100	00000000	;	
00000101	01010000	; HLT	停机

3.2.4 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 用短路块均将 1、2 短接，按图 3-2-5 连接实验线路。

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 3-2-2 的微代码写入 E2ROM 芯片中。

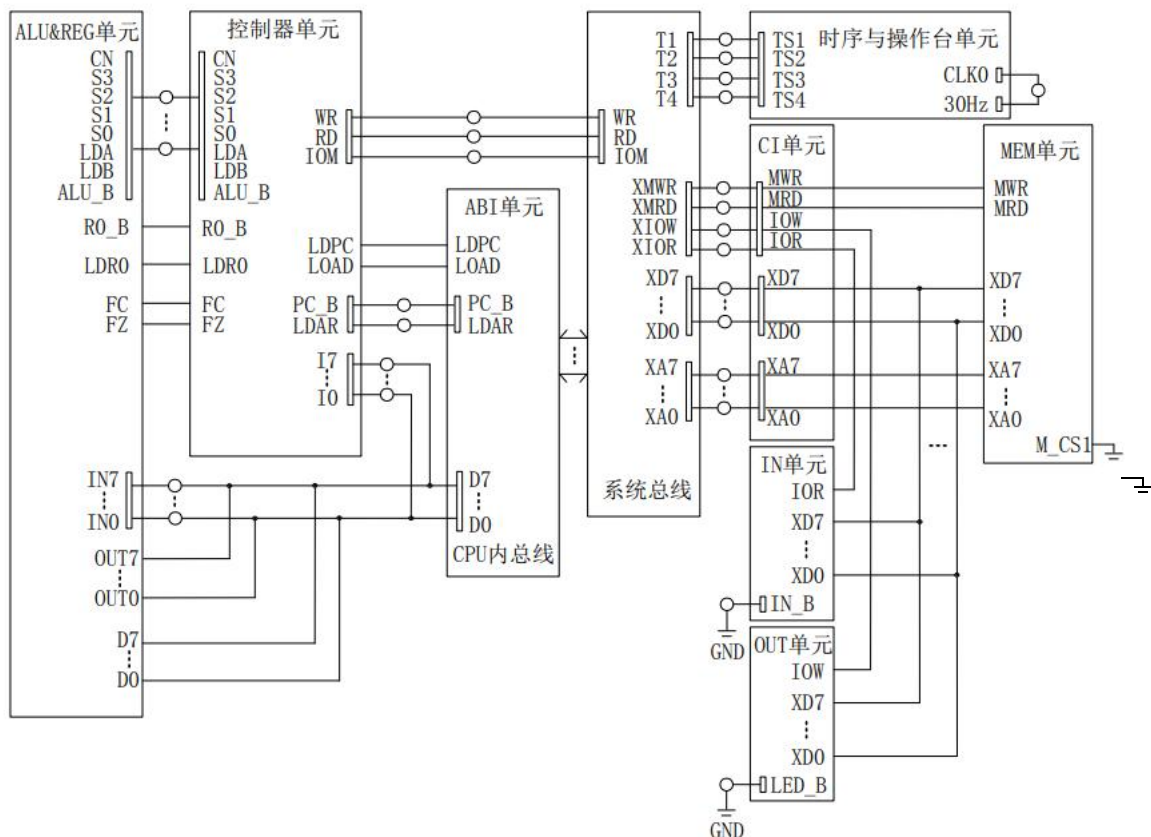


图 3-2-5 实验接线图

(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 单元的指数数据指示灯 M15——M8 显示该单元的中 8 位，MC 单元的指数数据指示灯 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。

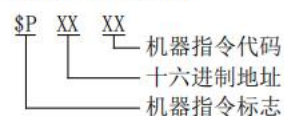
④ 连续两次按动时序与操作台的开关 ST，地址自动加 1，CPU 内总线的指数数据指示灯D7——D0 显示该单元的数据。此后每两次按动时序与操作台的开关 ST，地址自动加 1，CPU 内总线的指数数据指示灯 D7——D0 显示该单元的数据，继续进行该操作，直至完成校验，如发现错误，则返回写入，然后校验，直至确认输入的所有指令准确无误。

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，直至确认指令码的输入无误为止。

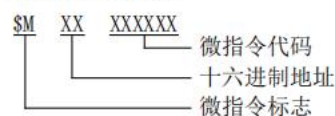
2) 联机写入和校验

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，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 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 **//

```

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 CPU 内总线和地址总线，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 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。

方法二：联机运行

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。

第 4 章 系统总线与总线接口

总线是计算机中连接各个功能部件的纽带，是计算机各部件之间进行信息传输的公共通路。总线不只是一组简单的信号传输线，它还是一组协议。分时与共享是总线的两大特征。所谓共享，在总线上可以挂接多个部件，它们都可以使用这一信息通路来和其他部件传送信息。所谓分时，同一总线在同一时刻，只能有一个部件占领总线发送信息，其他部件要发送信息得在该部件发送完释放总线后才能申请使用。总线结构是决定计算机性能、功能、可扩展性和标准化程度的重要因素。

本章安排了两个实验：系统总线和具有基本输入输出功能的总线接口实验，以及具有中断控制功能的总线接口实验。

4.1 系统总线和具有基本输入输出功能的总线接口实验

4.1.1 实验目的

1. 理解总线的概念及其特性。
2. 掌握控制总线的功能和应用。

4.1.2 实验设备

PC 机一台，TDX-CMX 实验系统一套。

4.1.3 实验原理

由于存储器和输入、输出设备要挂接到系统总线上，所以需要系统总线提供地址信号、控制信号以及双向数据通路。在该实验平台中，系统总线分为地址总线、控制总线和数据总线，分别为外设提供上述信号及通路。系统总线和 CPU 内总线之间通过三态门连接，同时实现了内外总线的分离和对于数据流向的控制。地址总线可以为外部设备提供地址信号和片选信号，片选信号由地址总线的高位通过半片 74LS139 双 2 线 - 4 线译码器进行译码，系统的 I/O 地址译码原理见图 4-1-1（在地址总线单元）。

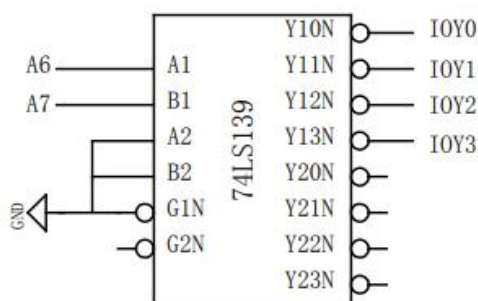


图 4-1-1 I/O 地址译码原理图

由于使用 A7、A6 进行译码，I/O 地址空间被分为四个区，如表 4-1-1 所示：

表 4-1-1 I/O 地址空间分配

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

为了实现对于 MEM 和外设的读写操作，还需要一个读写控制逻辑，使得 CPU 能控制 MEM 和 I/O 设备的读写，实验中的读写控制逻辑如图 4-1-2 所示，由于 T3 的参与，可以保证写脉宽与 T3 一致，T3 由时序单元的 TS3 给出（时序单元的介绍见附录 2）。IOM 用来选择是对 I/O 设备还是对 MEM 进行读写操作，IOM=1 时对 I/O 设备进行读写操作，IOM=0 时对 MEM 进行读写操作。RD=1 时为读，WR=1 时为写。

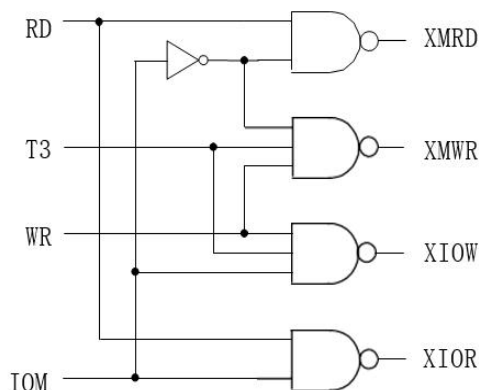


图 4-1-2 读写控制逻辑

在理解读写控制逻辑的基础上我们设计一个总线传输的实验。实验所用总线传输实验框图如图 4-1-3 所示，它将几种不同的设备挂至总线上，有存储器、输入设备、输出设备、寄存器。这些设备都需要有三态输出控制，按照传输要求恰当有序的控制它们，就可实现总线信息传输。

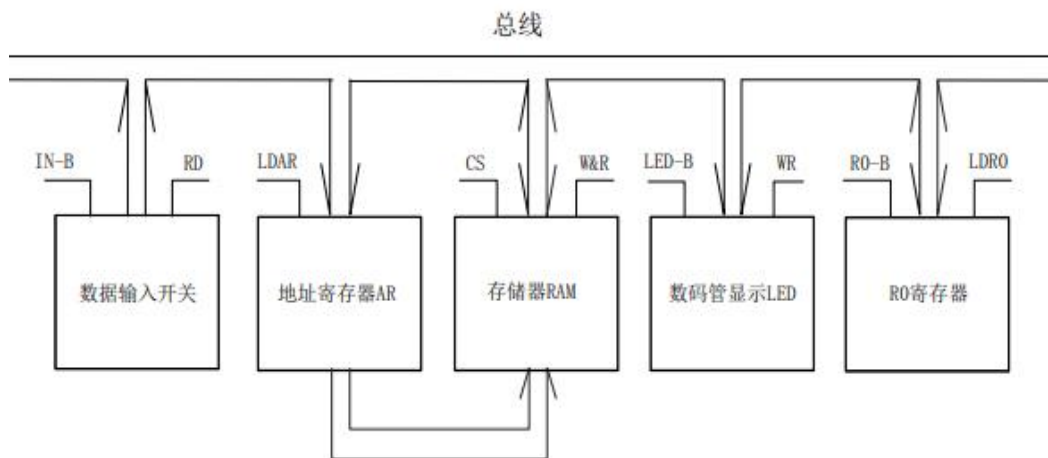


图 4-1-3 总线传输实验框图

4.1.4 实验步骤

3. 读写控制逻辑设计实验。

(1) 按照图 4-1-4 实验接线图进行连线。

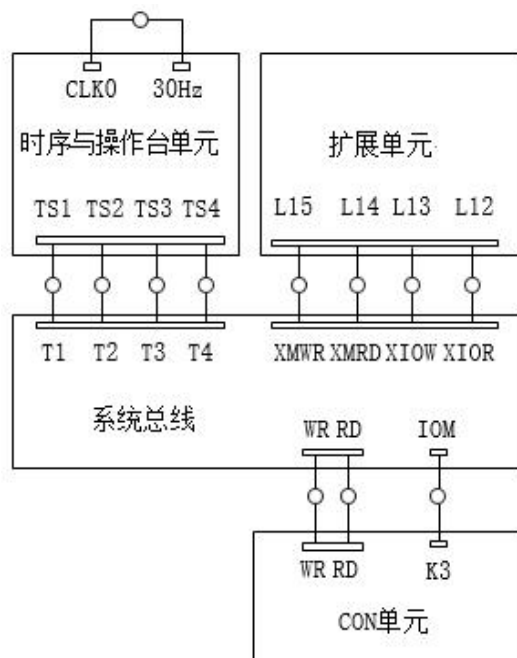


图 4-1-4 实验接线图

(2) 具体操作步骤如下：

首先将时序与操作台单元的开关 KK1、KK3 置为“运行”档，开关 KK2 置为“单拍”档，按动 CON 单元的总清按钮 CLR，并执行下述操作。

① 对 MEM 进行读操作 (WR=0, RD=1, IOM=0)，此时 L14 灭,表示存储器读功能信号有效。

② 对 MEM 进行写操作 (WR=1, RD=0, IOM=0)，连续按动开关 ST，观察扩展单元数据指示灯，指示灯显示为 T3 时刻时，L15 灭,表示存储器写功能信号有效。

③ 对 I/O 进行读操作 (WR=0, RD=1, IOM=1)，此时 L12 灭，表示 I/O 读功能信号有效。

④ 对 I/O 进行写操作 (WR=1, RD=0, IOM=1)，连续按动开关 ST，观察扩展单元数据指示灯，指示灯显示为 T3 时刻时，L13 灭，表示 I/O 写功能信号有效。

4. 基本输入输出功能的总线接口实验。

(1) 根据挂在总线上的几个基本部件，设计一个简单的流程：

① 输入设备将一个数写入 R0 寄存器。

② 输入设备将另一个数写入地址寄存器。

③ 将 R0 寄存器中的数写入到当前地址的存储器中。

④ 将当前地址的存储器中的数用 LED 数码管显示。

(2) 按照图 4-1-5 实验接线图进行连线。

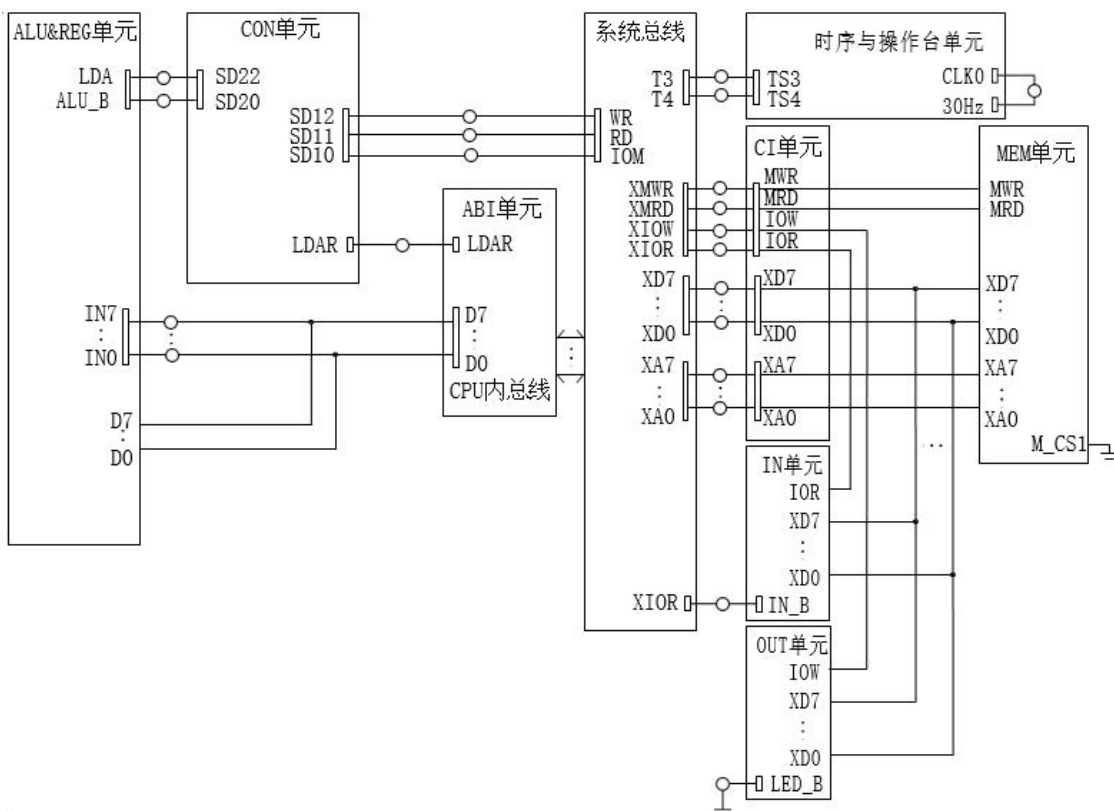


图 4-1-5 实验接线图

(3) 具体操作步骤如下：

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

将时序与操作台单元的开关 KK1、KK3 置为“运行”档，开关 KK2 置为“单拍”档，CON 单元所有开关置 0（由于总线有总线竞争报警功能，在操作中应当先关闭应关闭的输出开关，再打开应打开的输出开关，否则可能由于总线竞争导致实验出错），按动 CON 单元的总清按钮 CLR，然后按下面的顺序操作，在数据通路图中观测结果。

① 输入设备将 11H 写入 A 寄存器。

将 ALU_B 置为 1，关闭 A 寄存器的输出；WR、RD、IOM 分别置为 0、1、1，对 IN 单元进行读操作；IN 单元置 00010001，LDA 置为 1，打开 A 寄存器的输入；LDAR 置为 0，不将数据总线的数写入地址寄存器。连续四次点击图形界面上的“单节拍运行”按钮（运行一个机器周期），观察图形界面，在 T4 时刻完成对寄存器 A 的写入操作。

② 将 A 中的数据 11H 写入存储器 01H 单元。

将 ALU_B 置为 1，关闭 A 寄存器的输出；WR、RD、IOM 分别置为 0、1、1，对 IN 单元进行读操作；LDA 置为 0，关闭 A 寄存器的输入；IN 单元置 00000001（或其他数值）。LDAR 置为 1，将数据总线的数写入地址寄存器。连续四次点击图形界面上的“单节拍运行”按钮，观察图形界面，在 T3 时刻完成对地址寄存器的写入操作。

将 LDAR 置为 0，不将数据总线的数写入地址寄存器；LDA 置为 0，关闭 A 寄存器的输入；WR、RD、IOM 分别置为 1、0、0，对存储器进行写操作；ALU_B 置为 0，打开 A 寄存器的输出。连续四次点击图形界面上的“单节拍运行”按钮，观察图形界面，在 T3 时刻完成对存储器的写入操作。

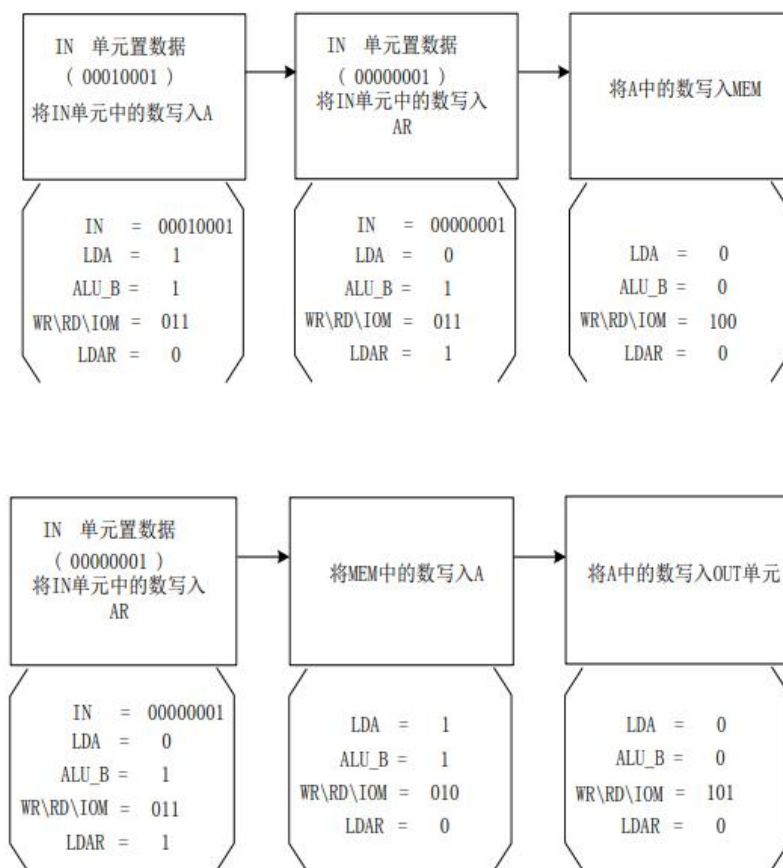
③ 将当前地址的存储器中的数写入到 A 寄存器中。

将 ALU_B 置为 1，关闭 A 寄存器的输出；WR、RD、IOM 分别置为 0、1、1，对 IN 单元进行读操作；LDA 置为 0，关闭 A 寄存器的输入；IN 单元置 00000001（或其他数值）。LDAR 置为 1，将数据总线的数写入地址寄存器。连续四次点击图形界面上的“单节拍运行”按钮，观察图形界面，在 T3 时刻完成对地址寄存器的写入操作。

ALU_B 置为 1，关闭 A 寄存器的输出；将 LDAR 置为 0，不将数据总线的数写入地址寄存器；WR、RD、IOM 分别置为 0、1、0，对存储器进行读操作；LDA 置为 1，打开 A 寄存器的输入。连续四次点击图形界面上的“单节拍运行”按钮，观察图形界面，在 T4 时刻完成对 A 寄存器的写入操作。

④ 将 A 寄存器中的数送往 LED 数码管进行显示。

先将 LDA 置为 0，关闭 A 寄存器的输入；LDAR 置为 0，不将数据总线的数写入地址寄存器；WR、RD、IOM 分别置为 1、0、1，对 OUT 单元进行写操作；再将 ALU_B 置为 0，打开 A 寄存器的输出。连续四次点击图形界面上的“单节拍运行”按钮，观察图形界面，在 T3 时刻完成对 OUT 单元的写入操作。



4.2 运算器与总线接口协作实验

4.2.1 实验目的

1. 理解总线的功能和和典型工作流程。
2. 掌握在总线上协调ALU和外设交换数据的方法。

4.2.2 实验设备

PC 机一台，TDX-CMX 实验系统一套。

4.2.3 实验原理

(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 用短路块将 1、2 短接，按图 4-1-6 连接实验电路，并检查无误。

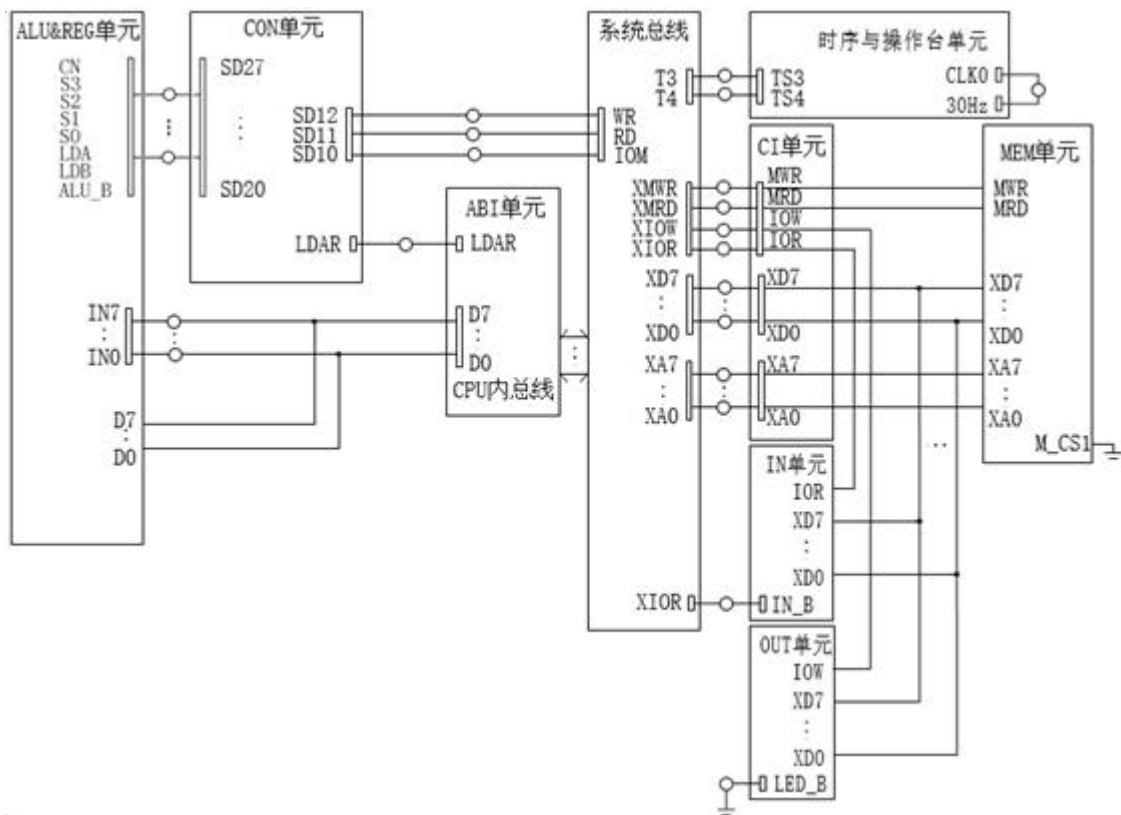


图 4-1-6 实验接线图

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

将时序与操作台单元的开关 KK1、KK3 置为“运行”档，开关 KK2 置为“单拍”档，CON 单元所有开关置 0（由于总线有总线竞争报警功能，在操作中应当先关闭应关闭的输出开关，再打开应打开的输出开关，否则可能由于总线竞争导致实验出错），按动 CON 单元的总清按钮 CLR，然后按下面的顺序操作，在数据通路图中观测结果。

(3) 打开电源，如果听到有“嘀”报警声，说明有总线竞争，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，将系统状态清零。

(2) 具体操作步骤如下:

首先将时序与操作台单元的开关 KK1、KK3 置为“运行”档, 开关 KK2 置为“单拍”档, 置 S3、S2、S1、S0 为 0000, 使 ALU 输出直接连接 A 寄存器, 置 LDA、LDB 为 0, 按动 CON 单元的总清按钮 CLR。然后依次执行下述操作, 并在数据通路图中观察结果。

① 输入设备将 22H 写入 A 寄存器。

将 ALU_B 置为 1, 关闭 ALU 的输出; WR、RD、IOM 分别置为 0、1、1, 对 IN 单元进行读操作; IN 单元置 00010001, LDA 置为 1, 打开 A 寄存器的输入; LDAR 置为 0, 不将数据总线的数写入地址寄存器。连续四次点击图形界面中的“单节拍运行”按钮(运行一个机器周期), 观察图形界面, 在 T4 时刻完成对寄存器 A 的写入操作。

② 将 A 中的数据写入存储器 01H 单元。

设置存储器地址: 将 ALU_B 置为 1, 关闭 ALU 的输出; WR、RD、IOM 分别置为 0、1、1, 对 IN 单元进行读操作; LDA、LDB 置为 0, 关闭 A、B 寄存器的输入; IN 单元置 00000001 (或其他数值)。LDAR 置为 1, 将数据总线的数写入地址寄存器。连续四次点击图形界面中的“单节拍运行”按钮, 观察图形界面, 在 T3 时刻完成对地址寄存器的写入操作。

写入数据到存储器: 将 LDAR 置为 0, 不将数据总线的数写入地址寄存器; LDA 置为 0, 关闭 A 寄存器的输入; WR、RD、IOM 分别置为 1、0、0, 对存储器进行写操作; ALU_B 置为 0, 打开 A 寄存器的输出。连续四次点击图形界面中的“单节拍运行”按钮, 观察图形界面, 在 T3 时刻完成对存储器的写入操作。

③ 改变地址和数据, 重复过程①、②, 将数据 44H 写入存储器 02H 单元。

④ 将 01H 存储器中的数送入 A 寄存器中。

设置存储器地址: 将 ALU_B 置为 1, 关闭 ALU 的输出; WR、RD、IOM 分别置为 0、1、1, 对 IN 单元进行读操作; LDA、LDB 置为 0, 关闭 A、B 寄存器的输入; IN 单元置 00000001 (或其他数值)。LDAR 置为 1, 将数据总线的数写入地址寄存器。连续四次点击图形界面中的“单节拍运行”按钮, 观察图形界面, 在 T3 时刻完成对地址寄存器的写入操作。

读出数据写入寄存器: ALU_B 置为 1, 关闭 ALU 的输出; 将 LDAR 置为 0, 不将数据总线的数写入地址寄存器; WR、RD、IOM 分别置为 0、1、0, 对存储器进行读操作; LDA 置为 1, 打开 A 寄存器的输入。连续四次点击图形界面中的“单节拍运行”按钮, 观察图形界面, 在 T4 时刻完成对 A 寄存器的写入操作。

⑤ 将 02H 存储器中的数送入 B 寄存器中。

设置存储器地址: 将 ALU_B 置为 1, 关闭 ALU 的输出; WR、RD、IOM 分别置为 0、1、1, 对 IN 单元进行读操作; LDA、LDB 置为 0, 关闭 A、B 寄存器的输入; IN 单元置 00000010 (或其他数值)。LDAR 置为 1, 将数据总线的数写入地址寄存器。连续四次点击图形界面中的“单节拍运行”按钮, 观察图形界面, 在 T3 时刻完成对地址寄存器的写入操作。

读出数据写入寄存器: ALU_B 置为 1, 关闭 ALU 的输出; 将 LDAR 置为 0, 不将数据总线的数写入地址寄存器; WR、RD、IOM 分别置为 0、1、0, 对存储器进行读操作; LDB 置为 1, 打开 B 寄存器的输入。连续四次点击图形界面中的“单节拍运行”按钮, 观察图形界面, 在 T4 时刻完成对 B 寄存器的写入操作。

⑥ 将 ALU 的运算结果写入存储器 00H 单元。

设置存储器地址: 将 ALU_B 置为 1, 关闭 ALU 的输出; WR、RD、IOM 分别置为 0、1、

1, 对 IN 单元进行读操作; LDA、LDB 置为 0, 关闭 A、B 寄存器的输入; IN 单元置 00000000 (或其他数值)。LDAR 置为 1, 将数据总线的数写入地址寄存器。连续四次点击图形界面上的“单节拍运行”按钮, 观察图形界面, 在 T3 时刻完成对地址寄存器的写入操作。

写入数据到存储器: 将 LDAR 置为 0, 不将数据总线的数写入地址寄存器; LDA、LDB 置为 0, 关闭 A、B 寄存器的输入; WR、RD、IOM 分别置为 1、0、0, 对存储器进行写操作; 置 S3、S2、S1、S0 为 1001, 使 ALU 做加法运算; ALU_B 置为 0, 打开 ALU 的输出。连续四次点击图形界面上的“单节拍运行”按钮, 观察图形界面, 在 T3 时刻完成对存储器的写入操作。

⑦ 将 00H 存储器中的数送入 A 寄存器中。

设置存储器地址: 将 ALU_B 置为 1, 关闭 ALU 的输出; WR、RD、IOM 分别置为 0、1、1, 对 IN 单元进行读操作; LDA、LDB 置为 0, 关闭 A、B 寄存器的输入; IN 单元置 00000001 (或其他数值)。LDAR 置为 1, 将数据总线的数写入地址寄存器。连续四次点击图形界面上的“单节拍运行”按钮, 观察图形界面, 在 T3 时刻完成对地址寄存器的写入操作。

读出数据写入寄存器: ALU_B 置为 1, 关闭 ALU 的输出; 将 LDAR 置为 0, 不将数据总线的数写入地址寄存器; WR、RD、IOM 分别置为 0、1、0, 对存储器进行读操作; 置 S3、S2、S1、S0 为 0000, 使 ALU 输出直接连接 A 寄存器; LDA 置为 1, 打开 A 寄存器的输入。连续四次点击图形界面上的“单节拍运行”按钮, 观察图形界面, 在 T4 时刻完成对 A 寄存器的写入操作。

⑧ 将 A 寄存器中的数送往 LED 数码管进行显示。

先将 LDA 置为 0, 关闭 A 寄存器的输入; LDAR 置为 0, 不将数据总线的数写入地址寄存器; WR、RD、IOM 分别置为 1、0、1, 对 OUT 单元进行写操作; 再将 ALU_B 置为 0, 打开 A 寄存器的输出。连续四次点击图形界面上的“单节拍运行”按钮, 观察图形界面, 在 T3 时刻完成对 OUT 单元的写入操作。