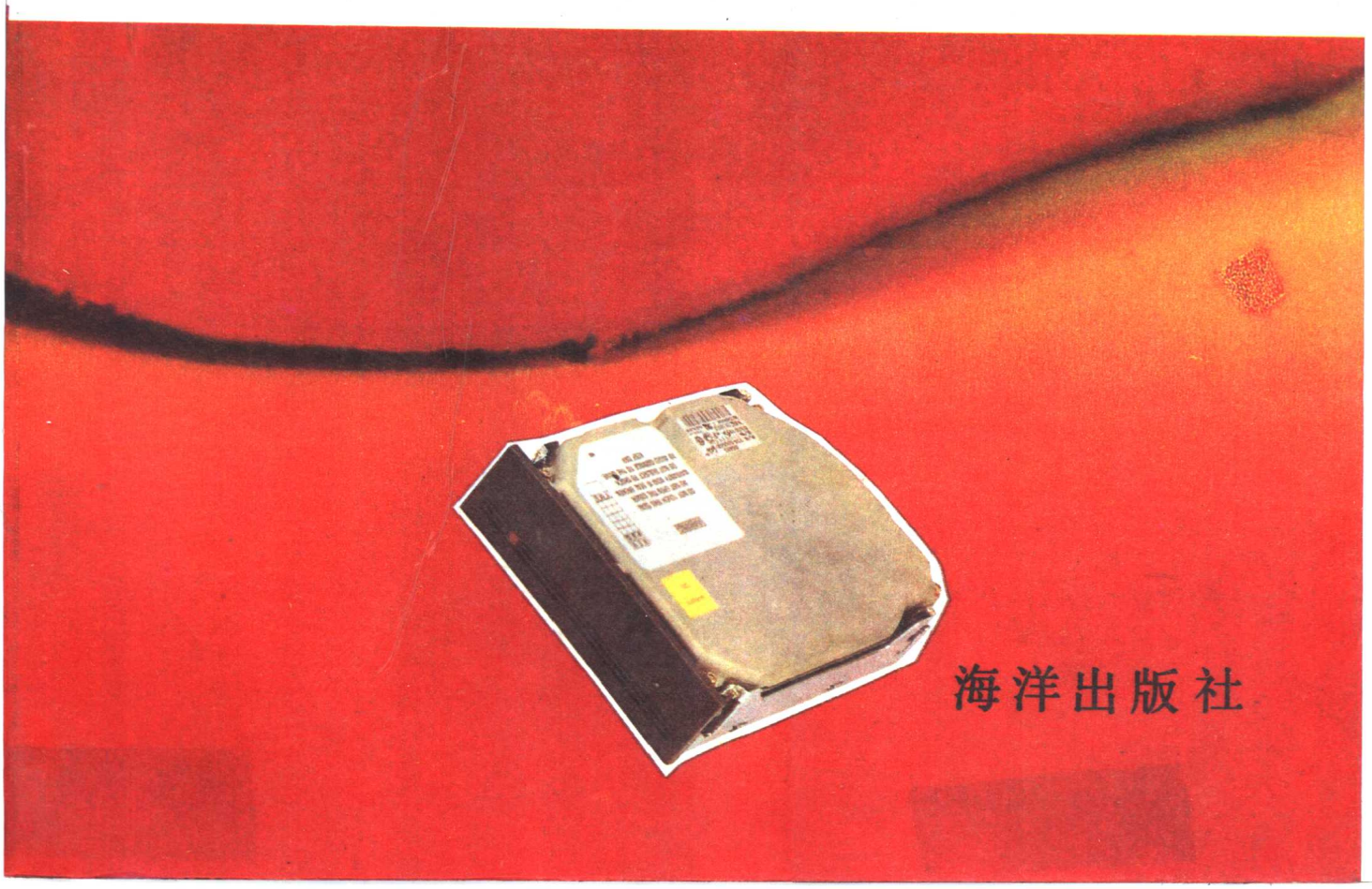
北京希望电脑公司微机技术丛书

微型计算机故障查找排除和程序设计

孙德和 编著
王 岩 审校



海洋出版社

北京希望电脑公司微机技术丛书

微型计算机故障查找排除和程序设计

孙德和　　编　著
王　岩　　审　校

海　洋　出　版　社

内 容 提 要

本书介绍微型计算机故障查找排除和程序设计的原理、方法和步骤。全书共分十三章，前一部分介绍微型计算机结构、无反应计算机的测试、板测试、诊断软件、外设测试、信号代码分析，以及逻辑分析仪和内电路仿真器；后一部分介绍典型的 VDU 设计、打印机设计、EPROM 编程器设计、DVM（数字电压表）设计和自举加载器设计。

本书原理与应用相结合，工具与方法并述，适用于广大微机工作者、科研人员和从事微机维修服务的工作者，也可作为大专院校有关专业的教学参考用书。

需要本书的用户，请直接与北京 8721 信箱联系，邮政编码 100080，电话 2562329。

(京) 新登字 087 号

特约编辑　辛　勤
责任编辑　刘荔馨

★

微型计算机故障查找排除和程序设计

孙德和　编　著
王　岩　审　校

★

海洋出版社出版（北京市复兴门外大街 1 号）
海洋出版社发行　　双青印刷厂印刷

开本：787×1092　1/16　　印张：14.25　字数：337 千字
1993 年 2 月第一版　　1993 年 2 月第一次印刷
印数：4000 册
ISBN7-5027-2938-0/TP·144　　定价：13.00 元

前 言

微型计算机是应用最为广泛的一种计算机，品种繁多。既有如袖珍计算器的单片系统，又有工业控制器和多用户办公用计算机。随着微型计算机应用的迅猛发展，要求服务人员必须掌握这些新电子工艺以及新技术。本书主要阐述基于微处理器设备的故障查找过程，测试设备及应用设计，并介绍了一些新颖的微型计算机故障检测手段。

第一章介绍微型计算机的操作原理，第二章到第八章介绍不同的测试设备、测试方法和故障查找过程。剩余章节介绍典型微型计算机系统的一系列应用设计研究。附录提供了所有微处理器及其支持器件的功能。

本书要求读者懂得二进制数，基本的数字电路，以及标准电子测试设备如 CRO（示波器）的使用。该书通俗易懂，图文并茂，具有一定的实用性和通俗性。既使是计算机业余爱好者也可以使用书中提供的设计过程定位并消除系统故障。

限于水平，书中错误之处在所难免，恳请读者批评指正。

编者

目录

此为试读，需要完整PDF请访问：www.ertongbook.com

第一章　微型计算机结构

1.1　微型计算机结构

微型计算机的通用结构如图 1.1 所示．中央处理单元 CPU（Central Processor Unit）通常是一单个集成电路 IC（integrated circuit），称作"微处理器"．它生成三条总线（或信号导体组），如下：

(a) 地址总线——在内存或输入／输出IC中选择一个位置单元；
(b) 数据总线——载送将要传进或传出CPU的程序指令或数据项；
(c) 控制总线——运载激活数据传送或指示特定进／出CPU的事件的控制信号．

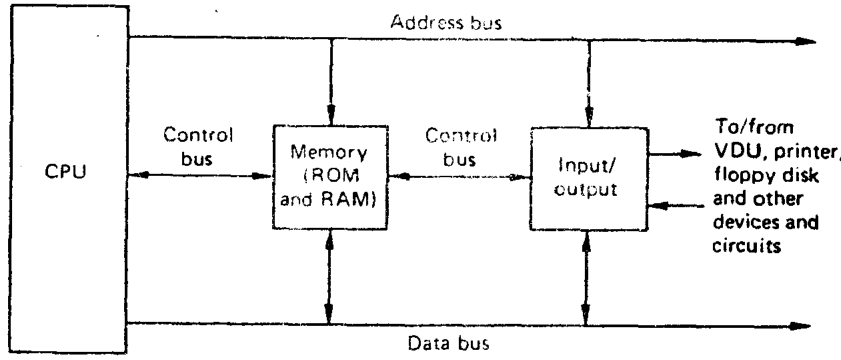


图 1.1　微型计算机结构

存储器模块包含将要由 CPU 执行的程序指令，以及在 CPU 执行程序时由它使用或生成的数据值．存储器通常由 ROM 和 RAM 家族的一个或多个 IC 组成，详见 1.4 节．

输入／输出模块把数据值传进和传出微型计算机，并且连接下述典型的设备项：

(a) 操作员终端，或 VDU（Visual Display Unit）；
(b) 打印机；
(c) 后备存储器（硬盘，或软盘，或二者兼有）——存储额外的程序以便传送到存储器由 CPU 执行；
(d) 各种类型的显示，例如数值段显示，LED（Light Emitting Diode）指示器；
(e) 到其他计算机的数据链路；
(f) 电控设备（马达，螺线管，继电器，加热器，开关）；
(g) 测试设备（连续或模拟信号）；

输入／输出模块可以由几个 IC 构成。

有时，整个电路可以包含到一单个 IC 中，例如袖珍计算器，洗衣机控制器，以及电话应答机。

1.2 CPU 执行的程序

中央处理器单元（CPU）按 8，16 或 32 位来处理数据值，其中：

1 位（二进制数字）＝0 或 1

也就是说，CPU 有一个 8，16 或 32 位的‘字长’．早期的微型计算机以及现代许多单功能的微型计算机，例如工业序列控制器，都使用 8 位微处理器．办公用计算机，例如‘PC’(Personal Computer)，具有很高的计算能力并使用 16 位或 32 位微处理器．

图 1.2 给出了典型微处理器的交互连接管脚功能．8 位微处理器通常固定在 40 个管脚的 DIL（Dual-in-Line）封装 IC 上，而 16 位和 32 位微处理器既可以固定在 40 到 60 管脚的 DIL 封装上，又可以固定在最多 120 个管脚的‘芯片载体’（管脚位于 4 个边上）上。

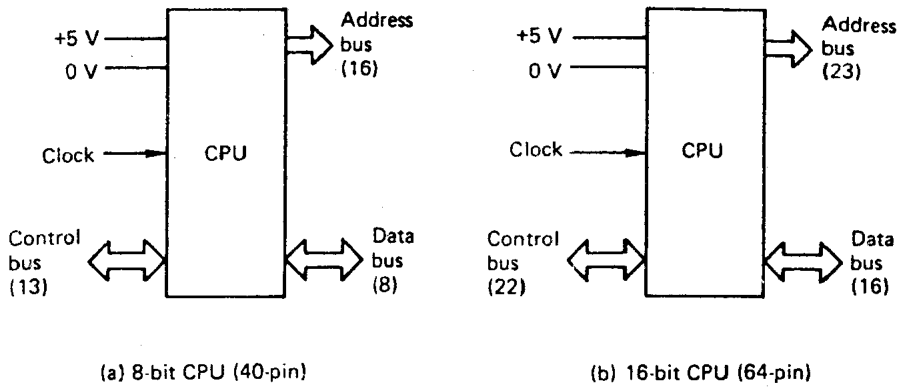


图 1.2　CPU 管脚功能

图 1.3 给出了 CPU 中的主要模块以及同包含 CPU 将要执行程序的存储器 IC 相连接的方法．存储器单元宽 8 位（字节），用于该 8 位微处理器的程序按机器码显示，即位格式存储在每个存储器单元中．注意，这种位格式按十六进制（hexadecimal）形式表示，例如：

十六进制 3E＝0011 1110 二进制

二进制与十六进制字符之间的关系见表 1.1。

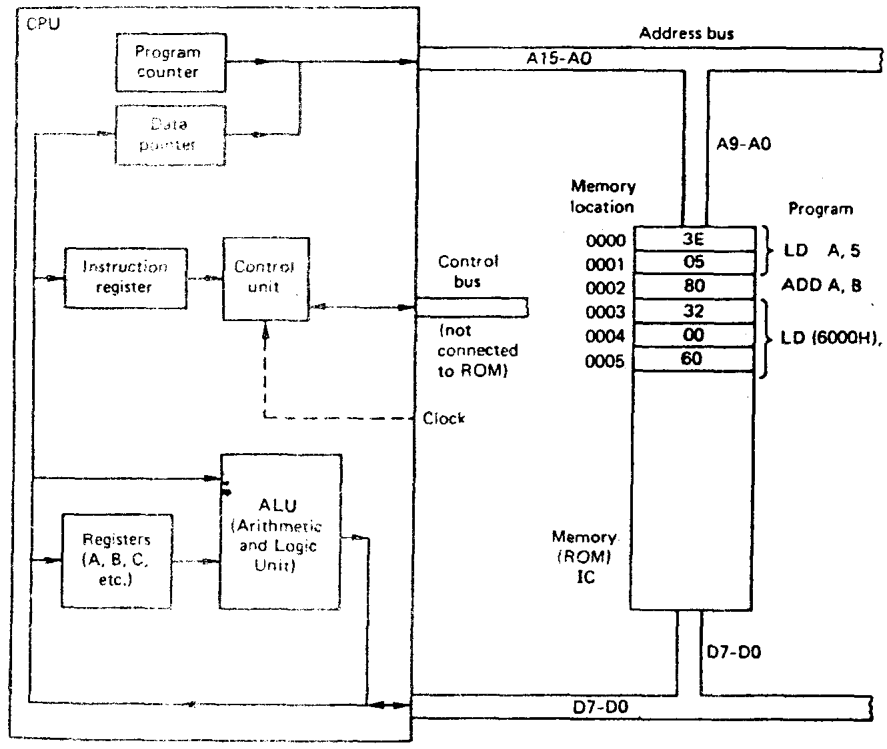


图 1.3　存储器中程序的 CPU 执行

表 1.1　二进制到十六进制转换

Binary	*Hexadecimal*
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

程序中的前三个指令在图 1.3 中给出，具体分析如下：

内存地址 (hex)	机器码 (hex)	汇编语言	注释
0000	3E,05	LD A,5	5→A
0002	80	ADD A,B	A+B→A
0003	32,00,60	LD(6000H),A	A→(6000H)

注意，对于这种 8 位微处理器，例如 Zilog Z80，指令具有可变长度（1,2 或 3 字节）。每一条指令都由 CPU 用下面的'取–执行'循环执行：

(a) 取——从存储器中取出指令字节（称作'操作码'）；
(b) 执行——CPU执行位于其指令寄存器中的操作码；这通常包括从内存读一个或多个字节（称作'操作数'）送给 CPU。

CPU 的各个模块规则可以概括如下：

(1) 指令寄存器——当 CPU 执行每条指令时存放机器码。
(2) 控制单元——检查操作码，向CPU以及CPU之外的元器件（在控制总线上）发送信号，以便完成指令；它完成的每个动作都由时钟信号上的一个脉冲触发。
(3) 寄存器——存入由程序处理的数据项。
(4) ALU——实现数据项上的算术（加、减等）和逻辑('与'、'或'等）运算。
(5) 程序计数器——存放将要从存储器读出的下一个程序指令字的存储器地址。
(6) 数据指针——存放将在从存储器读出或者写入存储器的数据项的存储器地址。

在本例中，CPU 执行三个程序指令时发生的事件序列在图 1.4 中给出。在没有遇到跳转指令时，CPU 逐步遍历存储器单元，并将 CPU 定向到一个不同的内存单元——用这一新存储器地址装入程序计数器。每个程序都用一个跳转指令来结束。只有在极少数情况下才使用'输入／输出'指令结束程序。

尽管本程序示例是基于 8 位 CPU 的，但是其操作原理同样可应用于 16 位 CPU，正常情况下，指令操作码长 2 个字节（占两个存储器单元），类似地，数据项也是两个字节长。很显然，这可以扩展到 32 位 CPU。

在上面简单的例子中并没有用到控制总线，但是控制总线却应用于所有的电路配置。最重要、最常见的控制总线信号如下：

(a) 中断，通常称做
RESET——设置该信号将使CPU的程序计数器为一定值（典型的是0000）；
NMI（非屏蔽中断）——迫使程序计数器为一个不同的存储器地址，这一地址包含由中断信号的设置所激活的中断服务例程的起始地址；

此为试读，需要完整PDF请访问：www.ertongbook.com

INT——用于促使CPU开始执行服务中断程序（中断服务程序）的各种技术。

(b) 读（例如RD）——选择数据传进或传出存储器以及沿双向数据总线输入／输出的方向。

(c) 写（例如WR）——(b)的反相。

(d) 存储器请求（例如MREQ)——选择存储器IC。

(e) 输入／输出请求（例如IORQ）——选择输入／输出IC。

(f) 直接存储器存取（DMA）请求和识别（例如BUSRQ和BUSAK，或HOLD和HOLDA——输入／输出IC使用这些同步交换信号来要求CPU解除与总线的连接，以便它自己用CPU总线寻址存储器IC。

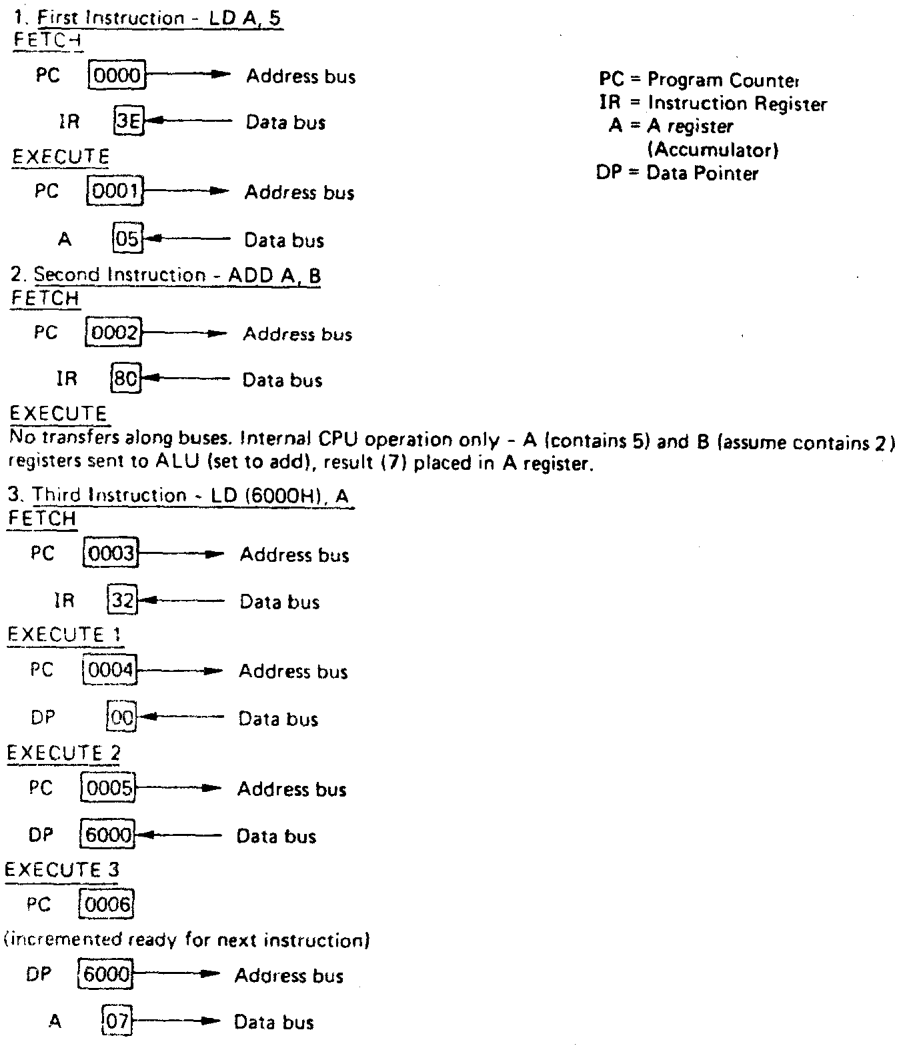


图 1.4　图 1.3 中程序指令的取–执行循环

1.3 地址译码

带有 16 个地址总线的微处理器可寻址:

2^{16} = 64K (65 536)存储器单元　　1K = 1024

同样，带有 23 个地址总线的 CPU 可寻址:

2^{23} = 8M 存储器单元　　1M = 1K × 1K

存储器 IC 必须适应于这一'存储器映象'的各个部分，电路管理必须能够保证一次只有一个 IC 被选择. 图 1.5(a)显示了如何用地址译码器 IC 来完成这一功能. 从图中可以看出两个存储器 IC 按相同方式连接到地址和数据总线上，但是它们却接收来自地址译码器电路的不同的 CE (Chip Enable) 信号——有时也称作 CS (Chip Select). 该电路可保证一次只能设置一个 CE 信号，这样在同一时刻也只有一个存储器 IC 可被选择. 地址译码器 IC 本身应该由 G (Enable) 信号的设置来激活，这一信号通常与 CPU 的控制总线信号 MREQ (Memory Request) 相连接。注意，在该特定布局中，有二个备用的未用 CE 信号.

ROM1 占据 2048 个存储器单元，如下:

11 条地址线→2^{11} = 2048 可寻址单元

并且每个单元占 8 位 (8 数据总线连接)，因此其"存储器组织"达到

2048 × 8

类似的，ROM2 为

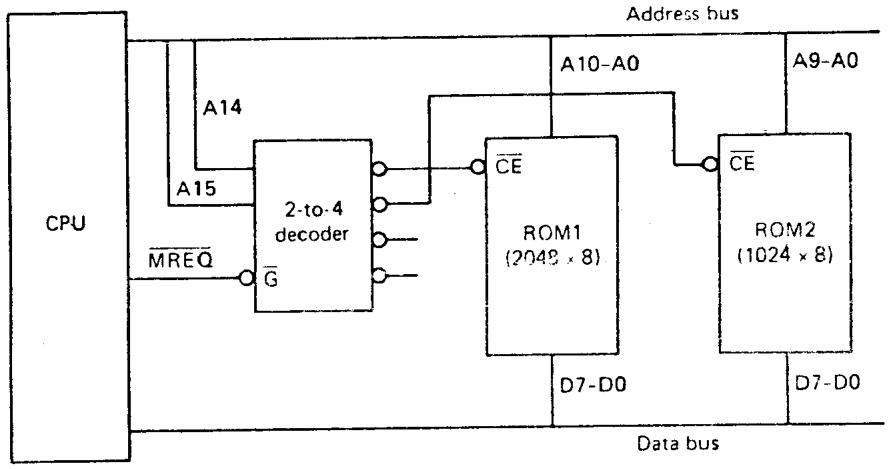
1024 × 8

地址译码器 IC 的操作可以概括到表 1.2 所示的真值表中.

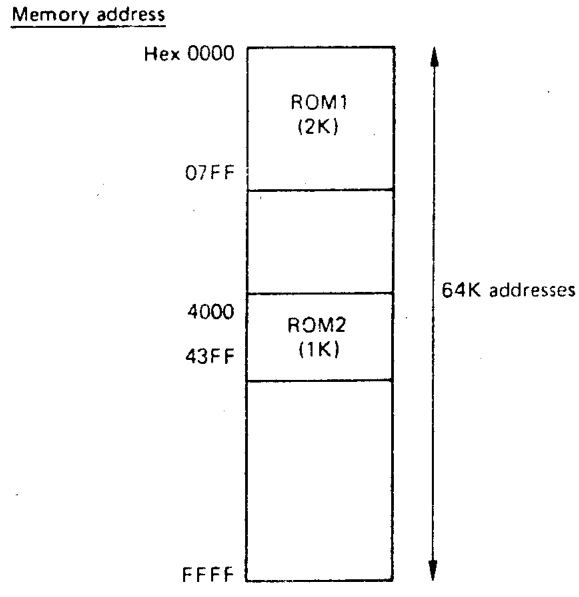
表 1.2　2-4 译码器 (SN74LS139 IC) 真值表

A13	*A14*	$\overline{CE4}$	$\overline{CE3}$	$\overline{CE2}$	$\overline{CE1}$
0	0	1	1	1	0
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	1	1	1

利用这个真值表，可以计算图 1.5(a)中两个 ROM 的存储器地址，见表 1.3. 注意，未用输入标为'X'，并假定为 0. 这样，每个 IC 的存储器范围就是图 1.5(b)存储器映象中的所给值.



(a) 地址译码布局



(b) 存储器映象

图 1.5 微型计算机存储器电路

表 1.5　用于图 1.5 的地址计算

Hex.	*Address line (An)*															
	15	*14*	*13*	*12*	*11*	*10*	*9*	*8*	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
0000	0	0	X	X	X	0	0	0	0	0	0	0	0	0	0	0
07FF	0	0	X	X	X	1	1	1	1	1	1	1	1	1	1	1
4000	0	1	X	X	X	X	0	0	0	0	0	0	0	0	0	0
43FF	0	1	X	X	X	X	1	1	1	1	1	1	1	1	1	1

ROM1 start	
ROM1 end	
ROM2 start	
ROM2 end	

Address decoder

用于输入／输出 IC 地址译码电路布局同用于存储器设备的电路布局是一样的．读者如果有兴趣可以对图 1.6(a)中的输入／输出电路重复上述分析．3–8 译码器的操作原理与 2–4 译码器的相同——输入中有一个增加二进制计数来设置每个连续的输出信号，以便选择不同的输入／输出 IC．注意，在这种情况下只有半数地址总线（A7–A0）用于输入／输出寻址，这是因为在输入／输出 IC 中，可寻址单元的数量少（本例中为 4）．输入／输出映象见图 1.6(b)．

虽然前面对图 1.5、图 1.6 的存储器和输入／输出电路作了单独的分析，但它们的典型组合可以产生一个全微型计算机电路．有几个微处理器不支持输入／输出程序指令，因此不能区分存储器和输入／输出 IC．在这种布局中，两种类型的设备都如前一样连接到地址和数据总线上，但要从同一个地址译码电路（例如 3–8 译码器）接收各自的 CE 信号，同时输入／输出被看成是‘存储器映象’．

1.4　存储器 IC

存储器 IC 是:

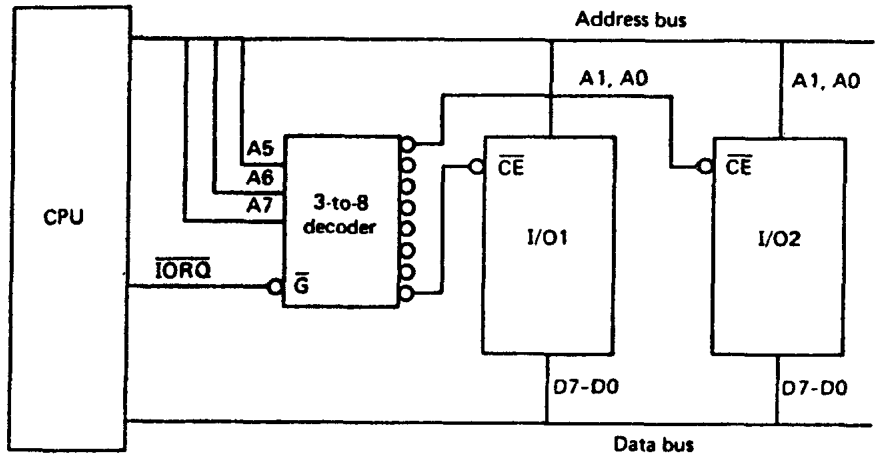
(a) ROM——只读存储器（Read–Only Memroy），或
(b) RAM——读／写存储器（Read／write Memory）
（RAM = Random Access Memory 随机存储器）

ROM 可细分为:

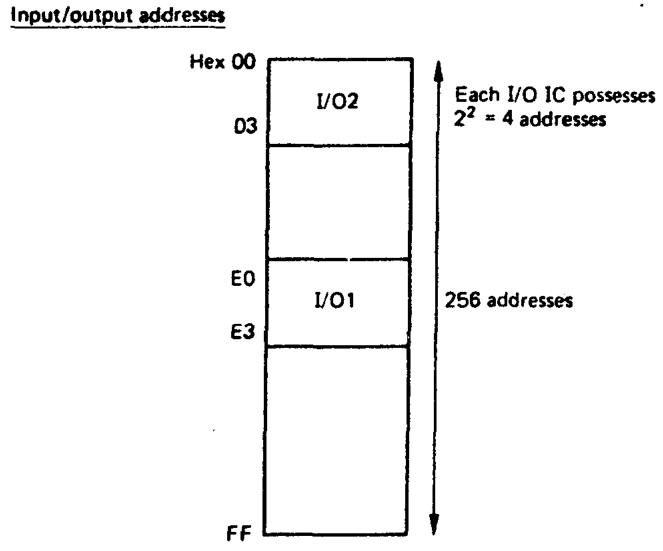
(1) ROM ——在设计和制造 IC 时编程;
(2) PROM——可编程 ROM，由用户编程;
(3) EPROM——可擦除 PROM，可重新编程多次.

通常，ROM 家族的成员都是管脚兼容的，因此考虑到 EPROM 器件的可重编程

性，可用 EPROM 设计和测试电路板，然后用廉价的 ROM 器件大量制造．擦抹 EPROM 时需要 UV (ultra-violet)（紫外线）光源．



(a) 地址译码布局



(b) 输入／输出映象

图 1.6　微型计算机输入／输出电路

每台计算机都将其所有的或部分程序存放在 ROM 中，而不是 RAM 中，以便在启动后输入该程序——参见 1.2 节中的 RESET．这是因为 RAM 明显的读／写优点具有"易

失性"，也就是说当关闭直流电源（例如关机）时，存储在其中的位格式将丢失。然而大多数微型计算机都具有一些 RAM，并且基于磁盘的机器中的多数程序都存放在磁盘上，在将被执行时再传送进 RAM。另外，数据项只能暂时存储在 RAM 中。

类似于 ROM，RAM 也可以细分如下：

(1) 静态 RAM；
(2) 动态RAM——具有与静态RAM相同的易失性。为防止存放的位格式丢失，需要多于典型的 2ms 一次的刷新操作。

图 1.7 给出一个典型的微型计算机存储器电路。ROM 和 RAM（静态）器件具有相同的管脚功能，在 RAM 中，用读／写（本例中为 WR）信号选择数据传送的方向。读者如有兴趣可以计算一下这些存储器设备的地址范围，以测试自己对地址译码的理解程度。答案是：

ROM 起始 : 0000（十六进制）
ROM 结束 : 0FFF
EPROM 起始 : 2000
EPROM 结束 : 23FF
RAM 起始 : 4000
RAM 结束 : 5FFF

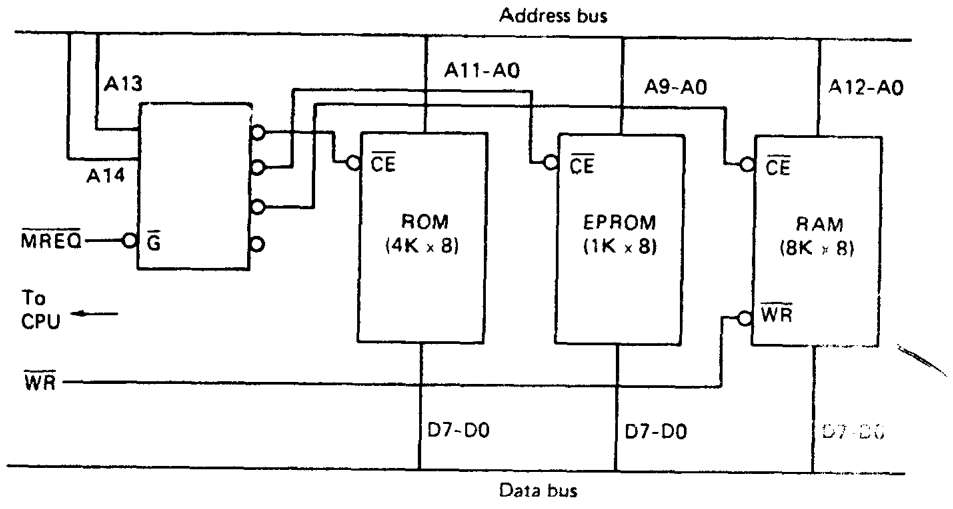


图 1.7 存储器电路

对于大的 RAM 系统，比如 8K 字节到 1M 或更多，考虑到价格因素，一般用动态 RAM 代替静态 RAM。动态 RAM IC 比静态 RAM 减少的价格超过使用动态 RAM 阵

此为试读，需要完整PDF请访问：www.ertongbook.com