

IBM PC

硬件系统线路分析

中国科学院 希望高级电脑技术公司

一九八八年一月

IBM PC

硬件系统线路分析

一九八八年一月

序 言

IBM PC 是目前 16 位个人电脑中最流行的一种,它对于个人电脑市场所造成的震撼,就如同当年的 *Apple II* 一样。在 *IBM PC* 尚未推出以前,一般人总是把个人电脑与电动玩具联想在一起,认为它最大的功能只是在娱乐方面。自从 *IBM PC* 上市后,人们才发现个人电脑不只是小孩手中的玩具,它是一个完整的电脑系统,能够帮助我们在工作上或日常生活上处理很多事情。

随着 *IBM PC* 的流行,市面上出现了很多介绍 *IBP PC* 的书籍,但是在这些书籍中专门谈 *IBM PC* 硬件的并不多。这本书的目的是帮助对 *IBM PC* 硬件电路有兴趣者,让读者了解 *IBM PC* 硬件是如何工作的。本书共分为两个部分,前面的第一至五章讲的是主机板的硬件电路,后面的附录 A 与附录 B 讨论的是视频显示电路。此外,附录 C 列出了 *IBM PC* 主机板、*IBM* 单色显示打印机接口卡的电路图。读者可以在阅读本书的过程中参考这些电路图,以助于了解每一个电路无件的细微动作。在分析主机板或 *IBM* 单色显示与打印机接口的电路以前,本书都会对整个电路功能做一概述,希望能给读者一个整体的概念。虽然本书讨论的是 *IBM PC* 而非 *PC-XT*,然而,由于 *PC-XT* 电路大部分皆与 *IBM PC* 电路机同,因此读者在阅读本书之后,自然也可以了解 *PC-XT* 的硬件电路。

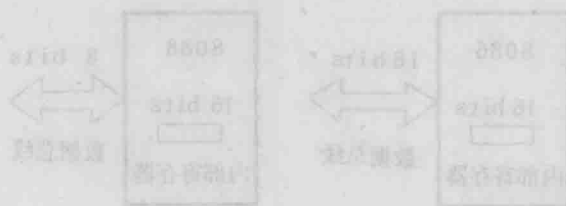
目 录

第一章 导论	(1)
1-1 IBM 个人电脑的核心——Intel8088	(1)
简介	(1)
功能说明	(2)
1-2 主机板上零件的布置	(11)
1-3 扩充槽工作原理	(13)
1-4 数值处理机 8087	(14)
概述	(14)
8087 和 8088 原系统构成	(15)
总线操作	(15)
8087 内部的结构	(15)
控制单元	(16)
数值运算单元	(17)
第二章 主机板电路功能概述	(18)
2-1 系统框图	(18)
2-2 CPU 电路	(21)
8088	(21)
地址锁闭和数据缓冲器	(21)
8084 时钟产生器	(21)
8288 总线控制器	(21)
8087	(23)
2-3 总线信号	(23)
2-4 READY 信号产生电路	(24)
8284 的 RDY 与 $\overline{\text{AEN}}$	(24)
WAIT STATE 的产生	(24)
2-5 中断	(25)
不可屏蔽中断	(25)
可屏蔽中断	(26)
2-6 DMA	(26)
DMA 控制器 8237A-5	(27)
DMA 地址	(27)
DMA 控制时序	(27)
DMA 请求信号	(28)

2-7	存储器.....	(28)
	ROM	(28)
	RAM	(28)
2-8	输入 / 输出.....	(29)
	主机板上的输入 / 输出设备.....	(29)
	扩充槽.....	(30)
(1)	系统设置开关.....	(31)
第三章	CPU 和相关电路.....	(32)
(1) 3-1	8088 和外围电路.....	(32)
(5)	时钟产生器.....	(32)
(11)	总线控制器.....	(33)
(21)	地址锁门和数据缓冲器.....	(33)
(41) 3-2	系统时钟.....	(34)
(41) 3-3	READY 信号产生电路.....	(36)
(61)	如何延长读写周期.....	(36)
(61)	使用 8284 产生 READY 信号.....	(37)
(61)	READY 信号产生电路分析.....	(38)
(81) 3-4	CPU 基本时序.....	(40)
(71)	CPU 自 RAM 读取数据.....	(40)
(81)	CPU 到 RAM 写入数据.....	(41)
(81)	CPU 自 ROM 读取数据.....	(42)
(13)	CPU 自输入 / 输出设备读取数据.....	(43)
(12)	CPU 到输入 / 输出设备写入数据.....	(44)
(13) 3-5	中断.....	(45)
(13)	8088 如何处理中断.....	(46)
(13)	NMI 电路.....	(48)
(13)	INTR 电路.....	(50)
(13) 3-6	存储器直接存取 (DMA).....	(51)
(13)	DMA 基本时序.....	(51)
(43)	DMA 控制信号产生电路.....	(54)
(13)	在 DMA 周期中 local bus 控制权的转移.....	(55)
(23)	DMA 周期的 READY 信号.....	(56)
(23)	DMA 控制器的时钟 DCLK	(57)
(33)	DMA 地址产生.....	(59)
第四章	存储器存取和地址译码.....	(62)
(33) 4-1	只读存储器 ROM	(62)
(33)	典型的 ROM	(62)
(33)	基本时序.....	(63)
(33) 4-2	ROM 的读取电路.....	(63)

ROM 的地址	(64)
读周期的时序	(64)
不同规格的 TMS4764	(65)
4-3 随机存取存储器 RAM	(65)
典型的 RAM	(66)
读取周期	(67)
写入周期	(68)
刷新周期	(68)
4-4 RAM 的控制信号产生电路	(70)
RAM 的地址	(70)
刷新周期的 $\overline{\text{RAS}}$	(71)
RAM 地址控制信号	(71)
row address 和 column address 的切换	(73)
刷新电路	(74)
刷新周期的 READY 信号	(77)
使用延迟线(delay line)的好处	(79)
Parity 的产生和检验	(80)
4-5 I/O 地址译码	(82)
第五章 主机板的界入/输出	(84)
5-1 主机板的输入/输出 LSI 芯片	(84)
8253-5	(84)
8255A-5	(86)
5-2 喇叭接口	(88)
5-3 录音机接口	(90)
5-4 键盘(keyboard)接口	(93)
键盘工作原理	(93)
键盘接口电路分析	(95)
5-5 系统设置开关	(96)
5-6 I/O 扩充槽	(97)
附录 A 视频原理	(102)
A-1 电视显像原理	(102)
阴极射线管	(102)
扫描	(103)
同步信号	(105)
合成视频信号	(107)
A-2 单色显示器(IBM Monochrome Display)	(110)
图像和同步信号分离式接收	(110)
设备特性	(111)
附录 B IBM 单色显示和打印机接口	(113)

B-1	视频扫描.....	(113)
B-2	CRT 控制器——MC6845	(114)
	概述.....	(114)
	信号引脚说明.....	(116)
	功能描述.....	(117)
	内部寄存器.....	(120)
B-3	IBM 单色显示和打印机接口卡的功能.....	(123)
	概述.....	(123)
	方框图.....	(130)
B-4	电路分析.....	(135)
	控制时序.....	(135)
	选迟电路.....	(137)
	显存.....	(138)
	字符发生器和移位寄存器.....	(140)
	属性码.....	(142)
	光标和字符闪烁, 亮度信号.....	(144)
	视频输出.....	(146)
	地址译码和数据输入 / 输出.....	(147)
	打印机接口.....	(148)
附录 C	电路图.....	(151)
C-1	主机板.....	(151)
C-2	IBM 单色显示和打印机接口卡.....	(163)



第一章 导 论

在当前的个人电脑市场上,IBM 个人电脑(简称 IBM PC)算是 16 位个人电脑中最流行的了。由于 IBM 公司利用它巨大而响亮的招牌和成功的行销策略,在推出 IBM PC 后迅速席卷了大部分的 PC 市场。

本书讲的是 IBM 个人电脑的硬件线路,目的在探讨 IBM 个人电脑硬件设计上的技巧。由于 IBM PC 的视频显示电路是设计在接口卡上,因此除了主机线路外,我们在附录中也叙述了单色显示和打印机接口卡的电路原理。

本书假定你已经具有基本组合和时序电路的基础,而且熟悉微处理机的功能。在讨论的过程中,我们常常会提到 IC 的编号和功能。如果书中没有详细分析过些 IC 的时序和规格,而你又想了解的话,请参阅 Intel、MOS 和 TTL 有关的技术资料。IBM PC 的主机板有两个版本:16/64K 主机板和 64/256K 主机板。因为这两个版本除了 RAM 电路部分有些差异外(地址译码、RAM chip 的容量和 RAM 时序控制电路)其他部分完全相同,所以本书选择了 64/256K 主机板来讨论。在附录 C 中,这两个版本的电路都详细的列出,读者可以根据对 64/256K 主机板电路了解的基础来分析 16/64K。

1-1 IBM 个人电脑的核心——Intel 8088

简介

一个电脑的核心是处理机(processor)的部分,一般称为中央处理机 CPU (central processing unit)。处理机的功能是把构成程序的每一条指令由存储器取出予以译码,再根据指令所代表的意义,执行算术或逻辑运算。个人电脑所用的处理机称为微处理机(microprocessor),CPU 的功能都浓缩在一个小小的芯片上,这样不但节省空间,还可以减少功耗。

IBM 个人电脑使用的微处理机是 Intel 公司所设计的 8088,它和 Intel 公司设计的另一种微处理机 8086 是同一系列。由于二者所有指令完全相同,所以在程序设计师的观点上,两个微处理机都是一样的。可是,在设计电路的观点上,两个微处理机就有点不同了。因为虽然 8088 内部的结构是 16 位,但是和外界连接的数据总线—data bus 却是 8 位;相反,8086 无论是内部结构或是对外连接的数据总线都是 16 位。我们可以由图 1-1 看出这两种差别。

虽然 8088 和外部连接的数据总线是 8 位,和一般 8 位的微外理机,例如:Z-80、6502 并

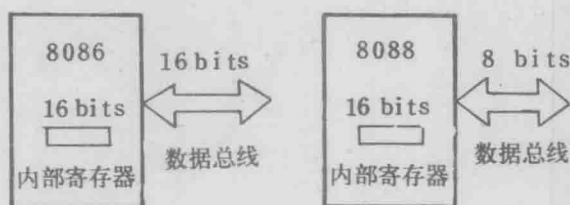


图 1-1 8086 和 8088 的区别

没有什么不同,但由于它内部所有的运算还是以 16 位为主,因此我们的它归纳为 16 位的微处理器。

刚刚提到 8088 对外界数据传输只有 8bits,而 8086 却有 16bits,也就是 8088 传送一个 byte 的时间,8086 可以传送两个 byte,这不是代表 8086 的速度是 8088 的两倍吗?事实并不然,因为在整个微处理机的执行时间里有部分时间用于等待数据由外界传入或输出到外界,甚至在某些时间里数据只需要用 8bits 来传送,这一点在下一节我们再详细讨论。

为什么 IBM 公司会选用 Intel 8088 呢?它的原因有两个:第一,是目前 8bits 的器件比 16bits 的器件多,设计 8bits 的电路也比 16bits 的电路来的容易。第二,市面上有许多可靠而便宜的 8bits 器件可以供电路设计者来选择。因此,IBM 根据这两点简化了它个人电脑电路的设计,并降低了它的价格,但也不可避免地降低了速度。

功能说明

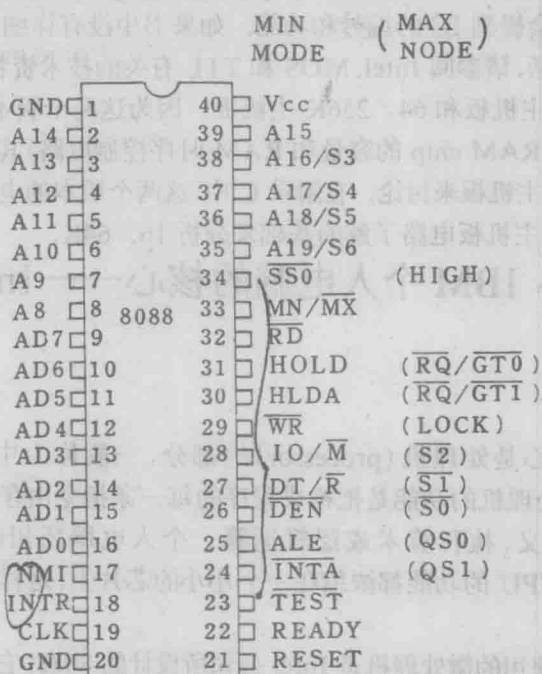


图 1-2 8088 微处理器

8088 是一种高性能的微处理器,它用 HMOS 技术制造,管脚(pin)数目为 40 的 VLSI 芯片(如图 1-2 所示)。8088 内部的运算完全以 16 位为主,虽然和外界数据传送的宽度只有 8 位,但性能并不是 8086 的一半。因为在 CPU 执行运算的过程中,等待输入数据的时间占全部时间的比例并不大,例如:CPU 正在执行一条乘法指令时,一旦操作数(operand)取入内部后,就开始运算直到求出结果为止,这时,它才可能与外部传送数据。何况 8088 内部分为两个部分:BUS INTERFACE UNIT (BIU)和 EXECUTION UNIT

(EU)。这两个部分可以同时工作。当 EXECUTION UNIT 执行内部运算时, BUS INTERFACE UNIT 还可以继续从存储器读取数据到内部 INSTRUCTION QUEUE (4 个 bytes), 这种功能叫做 prefetch。由于 8088 和 8086 都有 prefetch 的功能, 所以虽然二者的数据总线宽度比是 1 比 2, 它们之间执行速度不会受到数据总线宽度的影响而差别太大。图 1-3 框图可以说明 8088 的 prefetch 功能。

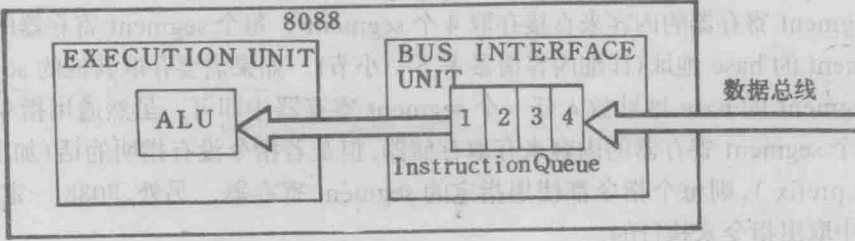


图 1-3 8088 的 prefetch

寄存器

8088 内部有 8 个 16 位的通用寄存器 (AX, BX, CX, DX, SP, BP, SI, DI) 和 4 个 16 位的 segment 寄存器 (CS, DS, SS, ES,) (如图 1-4 所示)。此外它还有一个 16 位的 Instruction pointer (IP) 和一个 16 位的标志寄存器 (FLAGS)。8 个通用寄存器中 4 个寄存器 (AX, BX, CX, DX) 的每一个又可以分为两个 8 位寄存器。8088 可以分别处理这些 8 位寄

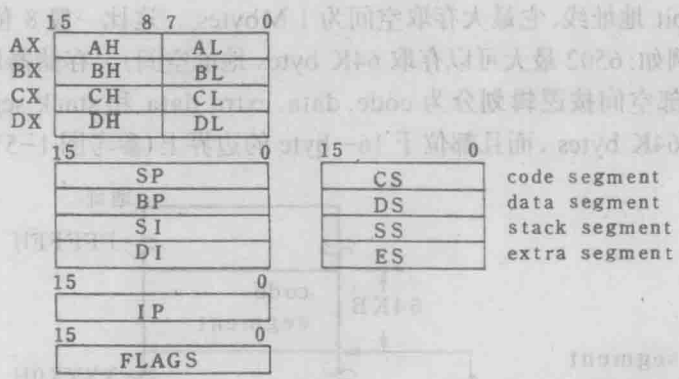


图 1-4 8088 内部寄存器

寄存器	运 算
AX	WORD MULTIPLY, WORD DIVIDE, WORD I/O
AL	BYTE MULTIPLY, BYTE DIVIDE, BYTE I/O, TRANSLATE, DECIMAL ARITHMETIC
AH	BYTE MULTIPLY, BYTE DIVIDE
BX	TRANSLATE
CX	String Operations, Loops
CL	Variable Shift and Rotate
DX	WORD Multiply, word divide, Indirect I/O
SP	Stack Operation
DI	String Operation
SI	String Operation

表 1-1 各寄存器的特殊用法

寄存器,而一般的算术逻辑运算都可以使用这些寄存器。另外4个通用寄存器(SP, BP, SI, DI)也可用来做一般的运算,但是这些寄存器只能以16位为单位来处理,而不像前面4个寄存器可以分为两个8位的寄存器来运算。虽然8个通用寄存器都能用来做一般的算术和逻辑运算,某些8088指令却将某个寄存器当成特殊寄存器来使用,表1-1列出了8个通用寄存器及其特殊的运算。

8088的存储地址被划分为64K byte的逻辑segment。在某一时刻,CPU只能根据内部4个segment寄存器的内容来直接存取4个segment。每个segment寄存器的内容都是一个segment的base地址(详细内容请参考下一小节)。如果需要存取其他的segment,只要将此segment的base地址放入任一个segment寄存器中即可。虽然通用指令都可以根据任何一个segment寄存器的内容来存取存储器,但是若指令没有指明的话(加上segment overwrite prefix),则每个指令都使用指定的segment寄存器。另外,8088一定是由code segment中取出指令来执行的。

16位的Instruction pointer (IP)和8085 CPU中的程序计数器(PC)类似。它是用来指定下一个将要执行指令的地址和目前code segment起始地址的位移(称为offset),即IP的内容指向下一条指令。16位的标志寄存器(FLAGS)有一些由内部运算而产生的1位状态信息,例如:zero, carry...等。某些指令可以根据FLAGS的内容来改变程序执行的次序,另外也有一些指令可以影响FLAGS。

存储器的组成

8088有20-bit地址线,它最大存取空间为1 Mbytes。这比一般8位微处理机的地址空间要大得多(例如:6502最大可以存取64K bytes地址空间)。存储器地址由00000H到FFFFFH。全部空间按逻辑划分为code, data, extra data和stack segments。每个segment的大小是64K bytes,而且都位于16-byte的边界上(参考图1-5)。

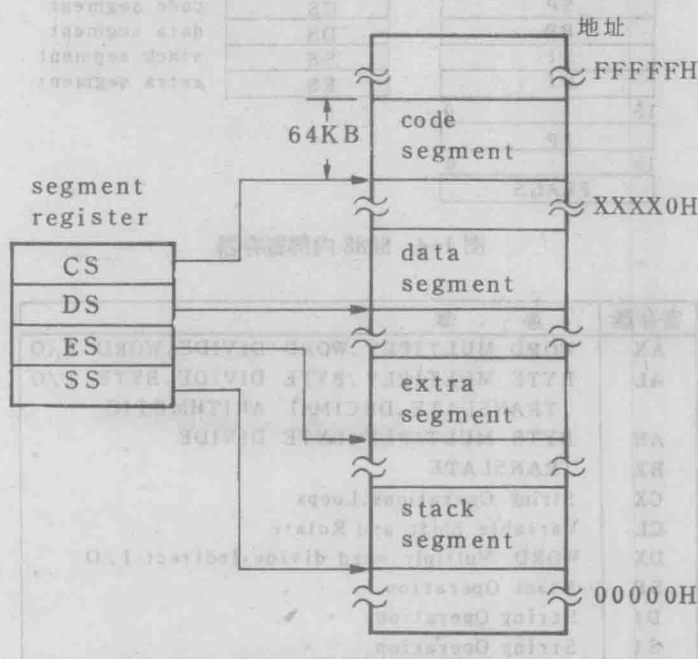


图1-5 存储器的指定

所有存储器地址都由两个部分组成: segment base 和 offset。segment base 由 8088 内部 4 个 16 位的 segment registers : code (CS)、data (DS)、extra (ES) 和 stack (SS) 提供, 因此任何一条存取存储器的指令都要指定 segment register 并提供 16 位的 offset, 8088 是将 segment base 左移(shift) 4 位后, 和 offset 相加而得到的一个 20 位数就是实际地址, 例如: 如果 segment base 是 1234 H, offset 是 55H, 则实际地址为 12395H, 如图 1-6 所示。由于存储器由 segment 组成, 因此程序浮动(relocate) 比较方便。只要改变 code segment register 的内容, 就可使一个程序在不同时间装在不同段内仍可正常执行, 而不需要做软件上的修改。此外, 这种方法也使程序结构性更强。

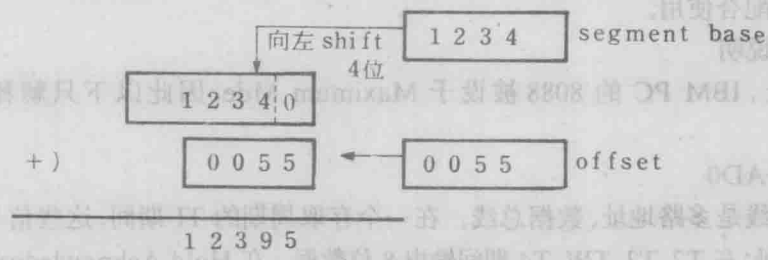


图 1-6 实际地址计算

必须保留某些单元作为 8088 系统的特殊用途。其中, FFFF0H 到 FFFFFH 必须存放一条转到起始地址的 JUMP 指令。当 8088 RESET 后, 它就从 FFFF0H 单元开始执行, 因此 FFFF0H 都存放一条 JUMP 指令, 这条指令转移到系统起始入口。00000H 到 003FFH 单元则保留给中断(interrupt)用。每个 4 byte 位置包含两部份: 16 位 segment base 和 16 位 offset, 称为 interrupt pointer。每一个 interrupt pointer 都是一个 interrupt Service Routine 的入口, 因此系统允许有 256 个 interrupt Service Routine。以上特定的单元可以用图 1-7 来说明之。

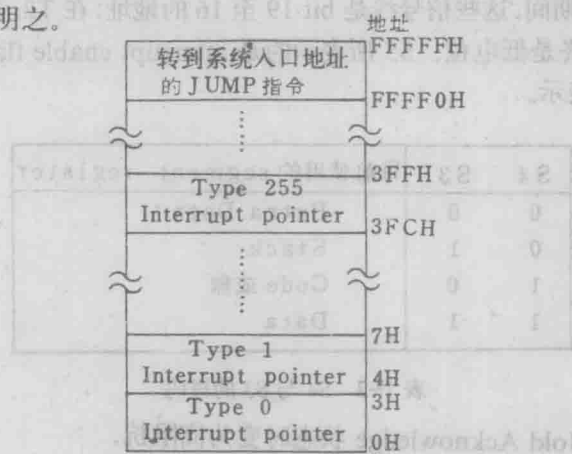


图 1-7 8088 保留的地址

Minimum 和 Maximum Mode

由图 1-2 可以看到, 有些信号引脚标出了两种不同的名称(例如: 第 28 脚的 IO / M 和 S2) 和。因为 8088 有两种不同的工作模式(mode): Minimum Mode 和 Maximum Mode。第 33 脚接高电位时, 8088 处于 Minimum Mode; 接低电位时, 8088 被设为 Maximum Mode。

在 Minimum Mode 时, 8088 第 24 脚到第 31 脚及第 34 脚, 产生总线控制信号, 8088 可

以在这种模式中以多路总线(multiplexed bus)和非多路总线(demultiplexed bus)的方式来使用。如果使用非多路总线,8088 要加锁闭(latch)和缓冲器。8088 用 $\overline{DEN}$ 和 $\overline{DT/R}$ 来控制缓冲器的开关和方向,用 ALE 来锁住地址。多路总线方式时可与 8155、8355、8755、.....等 IC 配合使用。这种方式使 8088 能够用最少的 IC 组成一个非常紧凑的系统。

在 Maximum Mode 中,8088 必须使用 8288 总线控制器。8088 在这种模式下提供了三个状态信号 $\overline{S_0}$ 、 $\overline{S_1}$ 和 $\overline{S_2}$,而 8288 将这些状态信息译码后产生了总线控制信号。由于 8288 是 bi-polar 器件,因此它的驱动能力很强,适用于复杂的系统,除此之外,8088 还提供了硬件锁定(lock)信号、Queue 状态和 request/grant 接口。有了这些特性 8088 才能和 co-processor 配合使用。

引脚信号说明

在设计上,IBM PC 的 8088 被设于 Maximum Mode,因此以下只解释 Maximum Mode 信号。

(1)AD7-AD0

这些信号线是多路地址、数据总线。在一个存取周期的 T1 期间,这些信号线输出 bit 7 至 bit 0 的地址;在 T2、T3、TW、T4 期间输出 8 位数据。在 Hold Acknowledge 时这些信号线变为高阻抗状态。

(2)A15-A8

这些信号线在整个存取周期内(T1 到 T4)提供了 bit 15 至 bit 8 的地址。由于在一个周期内一直有效,这些信号线不需要用 ALE 来锁住。在 Hold Acknowledge 状态时它们变为高阻抗。

(3)A19/S6, A18/S5, A17/S4, A16/S3

在存取周期 T1 期间,这些信号线是 bit 19 至 16 的地址;在 T2、T3、TW 与 T4 时,它们是状态信号。S6 始终是低电位。S5 所表示的是 interrupt enable flag bit。S4 和 S3 的译码可以用表 1-2 来表示。

S4	S3	目前使用的 segment register
0	0	Extra Data
0	1	Stack
1	0	Code 或無
1	1	Data

表 1-2 S4 与 S3 的译码

这些信号线在 Hold Acknowledge 状态时变为高阻抗。

(4) $\overline{RD}$

这个信号表示目前 8088 是否执行一个 memory 或 I/O read 周期。当 $\overline{RD}$ 为低电位,8088 处于 read 周期。此信号在 read 周期的 T2 到 T4 才变为低电位。

此信号在 Hold Acknowledge 时是高阻抗状态。

(5)READY

这个信号决定了当前请存取的 memory 或 I/O 设备发出 read 或 write 信号后,假如这个 memory 或 I/O 设备无法在 4 个 state 内完成

read 或 write,它必须在 T3 state 前将 READY 拉到低电位。直到 READY 变为高电位后,8088 才结束此周期,这个信号一般由 8284 的 READY 输出端提供。READY 须满足 setup 和 hold 时间,否则不能正常工作。

(6)INTR

这是一个 level - triggerred 的中断请求线。8088 在每一条指令的最后一个 clock 对此信号采样以决定是否进入 interrupt Acknowledge 周期。这个信号以高电位为触发条件,它是一个可屏蔽(maskable)中断信号线,因此可以用软件将 interrupt enable bit 置为 0,使 INTR 不起作用。

(7)TEST

当 8088 执行到“wait for test”指令时它曾检查该输入端。如果信号为高电位,则 8088 暂停执行而等待此信号变为低电位;如果为低电位,则 8088 继续执行下一个周期。

(8)NMI

这是一个 edge - triggerred 的不断请求线。由于它是不可屏蔽(non-maskable)中断信号,因此不能由软件使 8088 停止处理这类中断,NMI 中断类型为 type 2,因此 8088 不需要从外界读入 type vector。这个信号由低电位到高电位的变化作为它的触发条件,8088 会自动将这个变化锁住。

(9)RESET

此信号变为高电位后 8088 停止所有运算进入复位(reset)状态。当它变为低电位后 8088 才重新开始运行。

(10)CLK

这个时钟输入端提供了 8088 基本工作时钟。通常此输入端都由 8284 时钟发生器提供。为使 8088 正确运行,应输入一个高电平(duty cycle)为 33 % 的时钟。

(11)MN / MX

此信号线接高电位则 8088 处于 minimum mode;若接低电位,则 8088 进入 maximum mode。

(12)S2, S1, S0

这些状态信号在 T4 时钟的高电位、T1 及 T2 时是有效的;在 T3 或 TW (如果 READY 是高电位)时,这些信号则回到它们的被动状态(1,1,1)。8288 总线控制器利用这些状态信号来产生 memory 和 I / O 存取控制信号,如果在 T4 时,S2, S1, S0 发生变化,此时代表总线周期的开始。到了 T3 或 TW, S2, S1, S0 回到被动状态,代表总线周期结束。这些

S2	S1	S0	解 释
0	0	0	Interrupt Acknowledge
0	0	1	I/O Read
0	1	0	I/O Write
0	1	1	停止
1	0	0	Code Access
1	0	1	Memory Read
1	1	0	Memory Write
1	1	1	被动状态

表 1-3 不同 S2, S1, S0 所代表的意义

状态信号的各种组合及其代表的含义可由表 1-3 来说明。

这些信号在 Hold Acknowledge 状态时变为高阻抗。

(13) $\overline{RQ}/\overline{GT0}, \overline{RQ}/\overline{GT1}$

其他 local bus master (processor 或 DMA 控制器) 可以利用这两个 request / grant 输入端来迫使 8088 在结束当前总线周期后, 放弃本地总线(local bus)。这两根信号线都是双向的, 而 $\overline{RQ}/\overline{GT0}$ 比 $\overline{RQ}/\overline{GT1}$ 优先权高。此外, 它们的内部都连接了一个上拉(pull up)电阻, 因此, 如果不使用这个输入端时, 可以让他们不连接任何信号。一个 request / grant 的应答过程如下(参考图 1-8):

1. 另一个 local bus master 发出一个宽度为 CLK 周期的脉冲(pulse)给 8088, 表示它想使用本地总线(pulse 1)。

2. 在 T4 或 T1 (idle) 时钟周期中, 8088 由 $\overline{RQ}/\overline{GT}$ 发出一个宽度为 CLK 周期的脉冲给 master (pulse 2), 表示 8088 允许此 master 使用本地总线并放弃本地总线。在下一个 CLK, 8088 则进入 Hold Acknowledge 状态。此时, 8088 的 Bus Interface Unit 和本地总线切断。

3. 当 master 使用完局部总线后, 它发出一个宽度为 CLK 周期的脉冲(pulse 3)给 8088, 表示 8088 可以结束 Hold 状态。下一个 CLK, 8088 重新控制本地总线, 这时 8088 进入 T4。

本地总线控制权的切换需要三个脉冲, 而当总线控制权交换时, 中间必须间隔一个 idle CLK MFK 周期。以上的脉冲是以低电位为其触发状态。

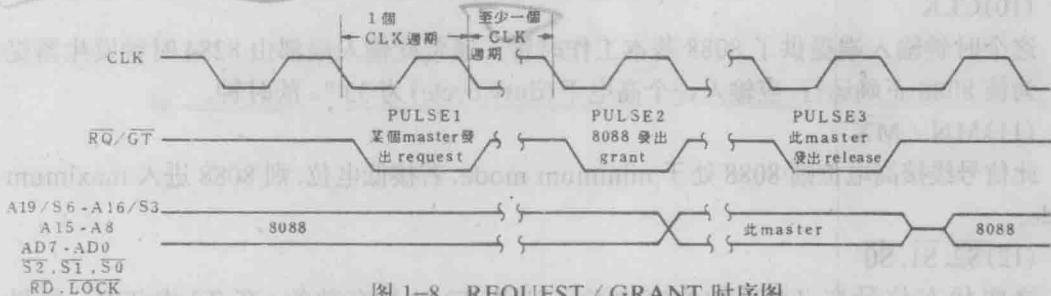


图 1-8 REQUEST/GRANT 时序图

(14) LOCK

在 LOCK 是低电位时, 系统中其他 bus masters 无法取得总线控制权。当 8088 执行到“LOCK”prefix 指令后, LOCK 变为低电位, 并持续到下一条指令结束。这个信号对多处理机系统的设计提供了一种方便的手段, 不需要再设计一个“test and set”电路。

这个信号在 Hold Acknowledge 状态下呈高阻抗。

(15) QS1, QS0

这两个状态信号使外部可以知道 8088 内部 Queue 的状态。在处理过 Queue 后的 CLK 周期, Queue 状态才是有效的。表 1-4 所示是 QS1, QS0 所代表的状态。

QS1	QS0	意 义
0	0	No operation
0	1	First byte of opcode from queue
1	0	Empty the queue
1	1	Subsequent byte from queue

表 1-4 QS1, QS0 代表的状态

(16)第 34 脚

此引脚在 Maximum mode 时,永远维持在高电位。

I/O 地址

8088 的 I/O 可以寻址到最大 64K bytes 的空间。由于 I/O 地址最大只有 64K bytes, 因此有效 I/O 地址线只有 A15—A0。在执行 I/O 指令时, A19—A16 全部为 0。I/O 指令可以分为: (1) 以 DX 暂存器为指针的指令。这些指令能够寻址到全部 64K bytes; (2) 直接寻址命令。这类指令直接指明 I/O 地址, 不是间接地用暂存器作指针。它们只能寻址到 256 bytes。第二类指令提供了一种和 8085 I/O 地址相匹配的方式。

总线操作

8088 的地址 / 数据总线可以分为三个部分: (1) 低 8 位数据 / 地址 (AD0—AD7), (2) 中间 8 位地址 (A8—A15), (3) 高 4 位地址 (A16—A19)。第一部分和第三部分的总线是以分时多路 (time multiplexed) 方式操作的。这项技术最有效地利用了处理器的引脚, 使 8088 可以放在 40 脚的封装内。中间 8 位地址不是多路的, 它们在整个总线周期内, 都有效。如果需要一个非多路总线的系统, 我们只要用一个地址门将 8088 的地址在适当时候锁住就可以了。

每个处理器总线周期至少包含 4 个 CLK 周期 (参考图 1-9)。我们用 T1、T2、T3 和 T4 来表示这 4 个 CLK 周期。在 T1, 地址由处理器内发出而数据的传送则发生在 T3 和 T4 之间。T2 主要是在 read 周期中用来改变总线方向的。如果在周期运行时, 被 CPU 寻址到的设备不能在 4 个 CLK 时钟内完成 (“NOT READY”), 它可以控制 READY 信号, 使 “wait” states (TW) 能够插在 T3 和 T4 中间。每个插入的 “wait” state 时间都和 CLK 周期时间相同。如果 8088 在相邻两个总线周期内不使用总线, 则这些 CLK 周期称为 “idle” states (TI)。在这些周期里, 8088 完全执行内部的运算。

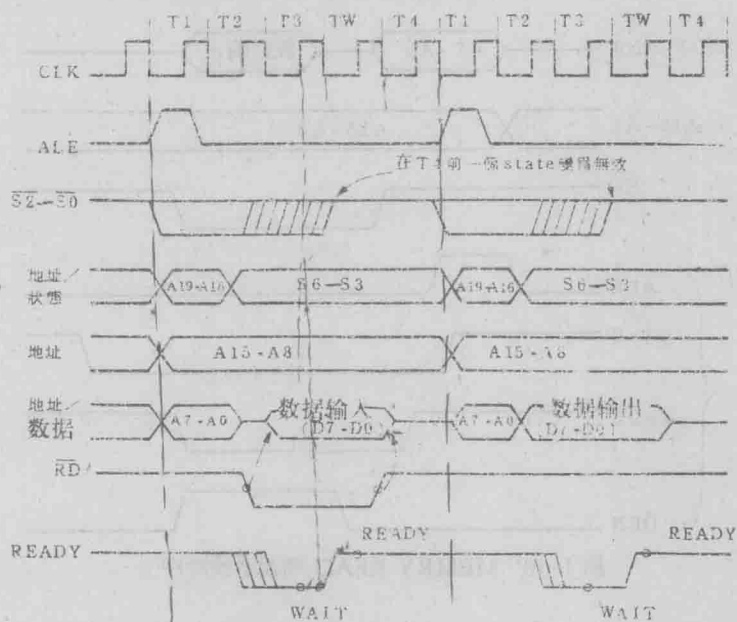


图 1-9 基本系统时序

在每个总线周期的 T1 内, ALE (Address Latch Enable) 由处理器 (Minimum mode) 或 8288 总线控制器 (Maximum mode) 发出。在它的下降沿 (falling edge) 可锁住有效地址和某些周期状态信号。

在 Maximum mode 时, 总线控制器利用状态位 $\overline{S0}$ 、 $\overline{S1}$ 、和 $\overline{S2}$ 来区别当前 8088 总线周期类型。状态位 S3 到 S6 则和高 4 位地址以多路的方式发出。前者在 T2 至 T4 的中间是有效的。它们指明了在目前总线周期内使用哪个 Segment 寄存器来形成地址。S5 和 Interrupt enable bit 是相同的, 而 S6 则永远为低电位。

总线时序

以下就 Maximum mode 的总线周期来说明 (如图 1-10 所示)。对于一个较复杂的系统, $\overline{MX}/\overline{MN}$ 端应该接到低电位, 而 8088 则必须和 8288 配合使用。同时使用地址锁来锁住系统地址。数据缓冲 (data transceiver) 则用来提供比 8088 更大的电流推动能力, 在这种结构里 ALE、DEN (data enable) 和 $\overline{DT}/\overline{R}$ (Data Transmit/ Receiver) 并非由 8088 提供, 而是由 8288 发出。8088 还提供状态信号 ($\overline{S2}$ 、 $\overline{S1}$ 和 $\overline{S0}$) 使 8288 能够辨认目前是哪一种周期。这些状态信号指明了目前周期是 read (code、data 或 I/O)、write (data 或 I/O)、interrupt acknowledge 或 software halt。根据不同的周期, 8288 发出不同的控制信号 ($\overline{MRDC}$ 、 $\overline{IORC}$ 、 $\overline{MWTC}$ 、 $\overline{IOWC}$ 、 $\overline{INTA}$ 等)。系统的数据传送可以利用 $\overline{DT}/\overline{R}$ 来决定数据方向, 而 DEN 则控制了数据缓冲的开或关。后面几章, 我们再针对 IBM PC 的电路来分析 8088 几种总线周期的详细时序。

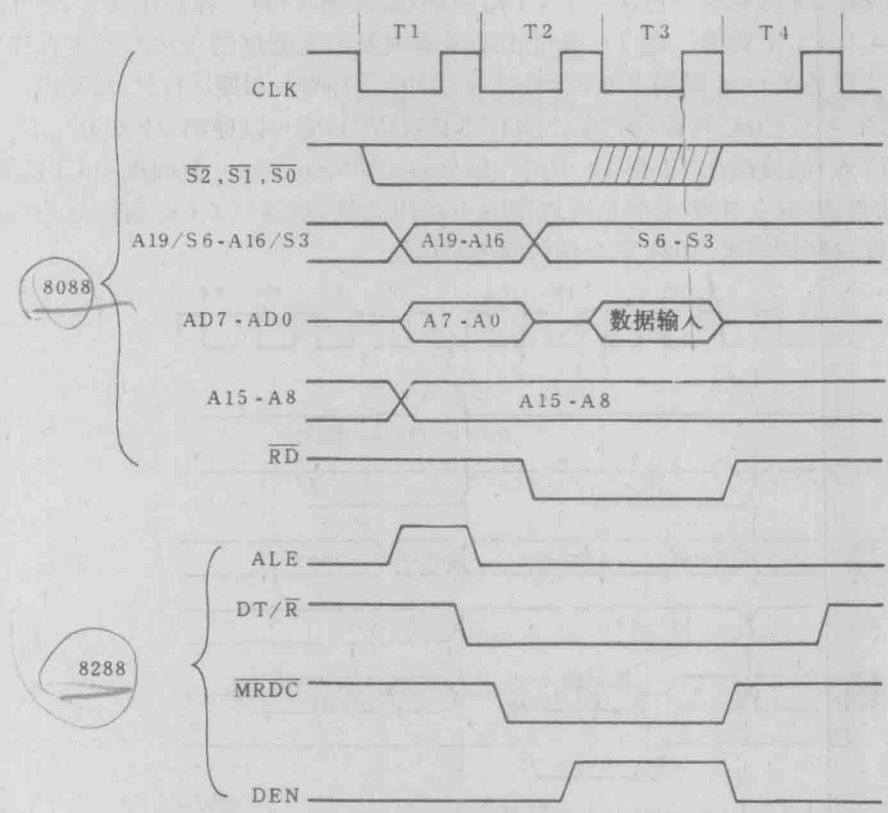


图 1-10 MEMORY READ 周期系统时序