硬件系统线路分析

一九八八年一月

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 位的微处理机，例如：Z-80、6502 并

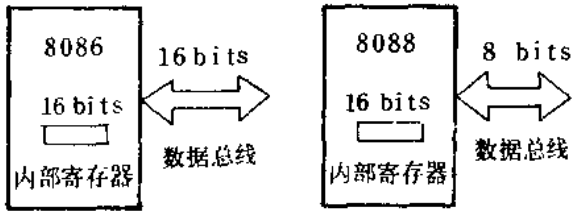


图 1-1 8086 和 8088 的区别

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 位的微处理机。

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

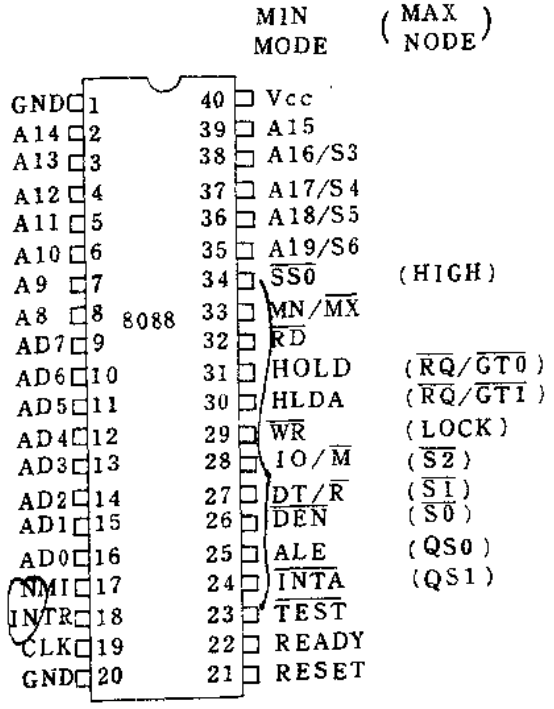
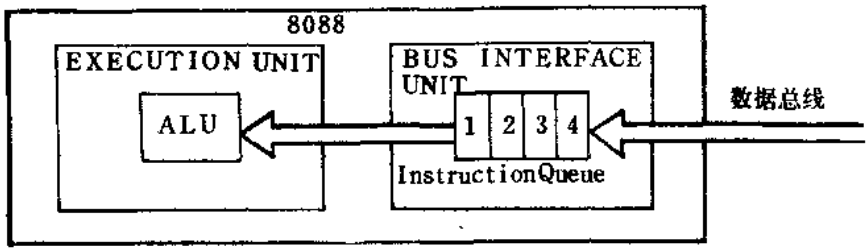


图 1-2 8088 微处理机

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。



■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

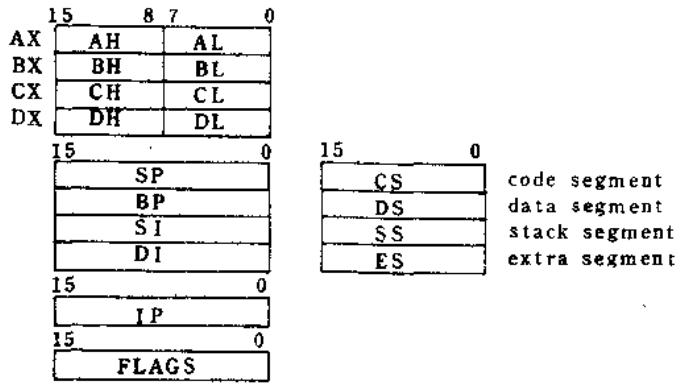


图 1-4　8088 内部寄存器

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

在各种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)，ewr 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)。

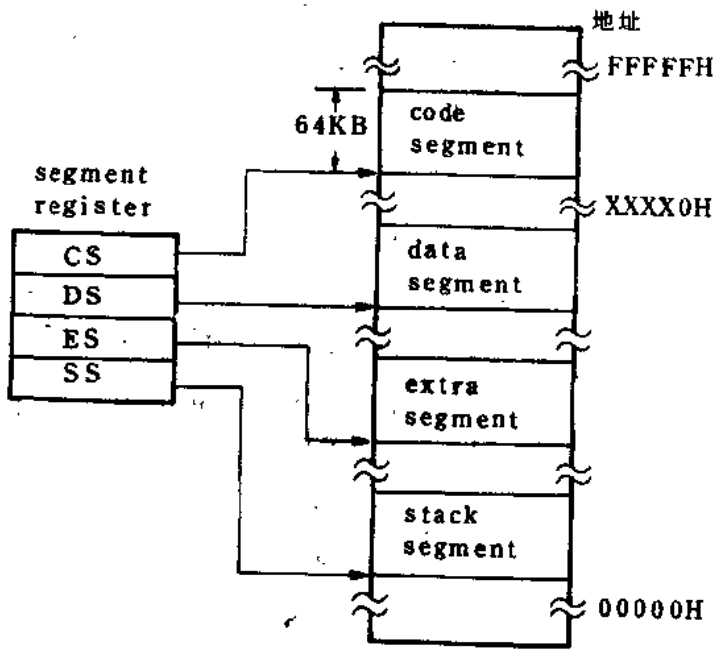


图1-5 存储器的指定

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 (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，offest 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。

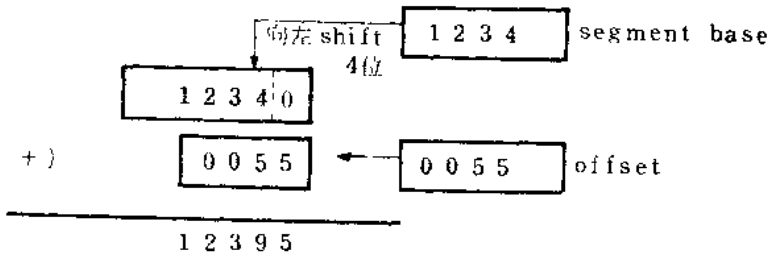


图 1-6 实际地址计算

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。

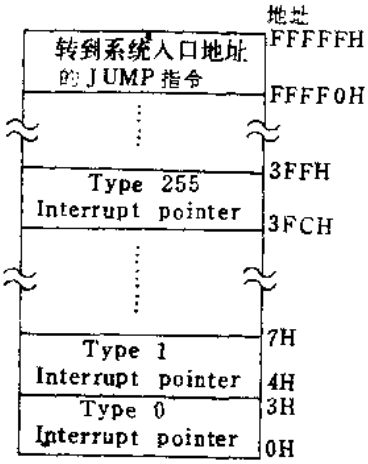


图 1-7 8088 保留的地址

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 / $\overline{M}$ 和 $\overline{S2}$) 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要加锁门(latch)和缓冲器。8088用$\overline{DEN}$和DT/$\overline{R}$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$\overline{S0}$、$\overline{S1}$和$\overline{S2}$,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 Mde,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bit0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／$\overline{MX}$

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)$\overline{S2}$，$\overline{S1}$，$\overline{S0}$

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，$\overline{S2}$、$\overline{S1}$、$\overline{S0}$ 发生变化，此时代表总线周期的开始。到了 T3 或 TW，$\overline{S2}$、$\overline{S1}$、$\overline{S0}$ 回到被动状态，代表总线周期结束。这些

$\overline{S2}$	$\overline{S1}$	$\overline{S0}$	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	被動狀態

表 1－3　不同 $\overline{S2}$，$\overline{S1}$，$\overline{S0}$ 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 (processer 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。

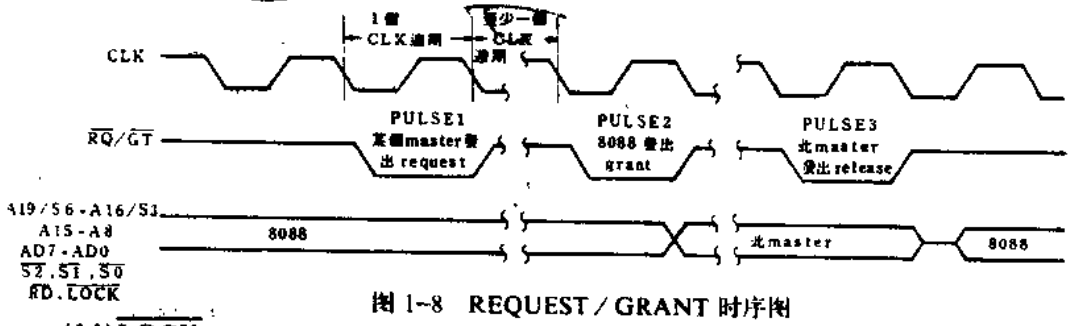


图 1-8 REQUEST / GRANT 时序图

(14) $\overline{LOCK}$

在 $\overline{LOCK}$ 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，$\overline{LOCK}$ 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。

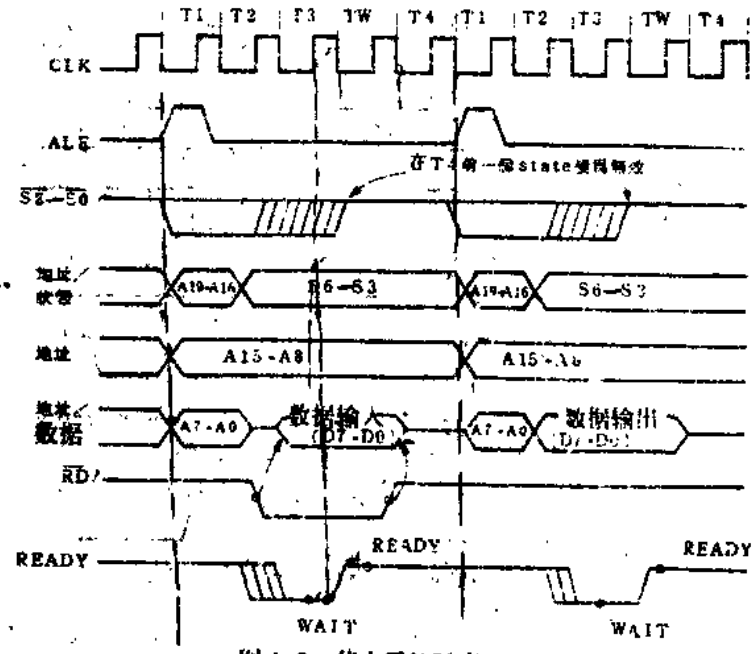


图 1—9 基本系统时序

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。同时使用地址闩来锁住系统地址。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 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 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。

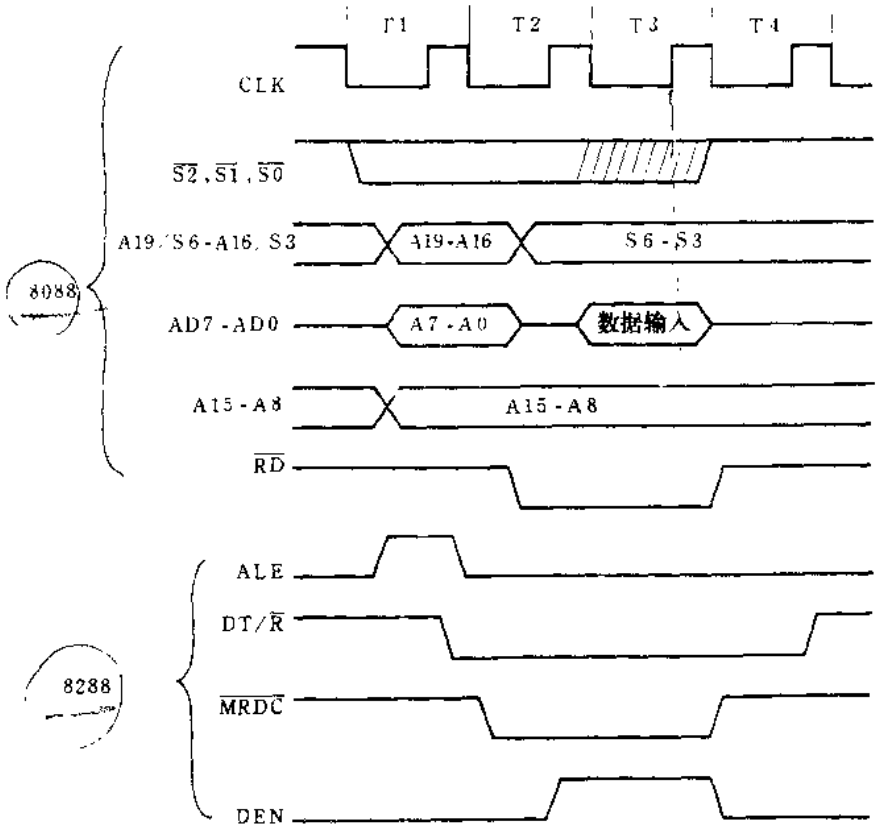


图 1-10 MEMRY READ 周期系统时序

中断

8088 的中断可以分为可屏蔽中断(maskable)和不可屏蔽中断(non-maskable)两种。可屏蔽中断由 INTR 输入,它是一个 level triggerred 的信号(高电位是它的触发状态)。我们可以将 8088 的 interrupt enable bit 设为 0 而使中断无效。INTR 通常连接了一个 8259 中断控制器,此中断控制器可以控制 8 级中断。并在 Interrupt Acknowlege 周期内提供必要的 type vector 给 8088(详细操作,请参考第三章)。不可屏蔽中断由 NMI 端输入,我们不能用软件屏蔽。它是一个 edge-triggerred 的信号(由低电位到高电位的变化是它的触发状态),而且比 INTR 有较高的优先权。在 INTR 端和 NMI 端同时触发时,8088 先响应 NMI。

LOCK

当处理器不希望相邻两个总线周期被其他 bus master 打断时,它就发出状态信号 LOCK。这使处理机可以对存储器进行 read/modify/write 操作,不会被系统中其它 bus master 中断这些存取周期。这项功能在多处理器系统下是非常有用的,它能够实现所谓的"test and set"操作。

在处理机将 LOCK prefix 指令译码后,在下一个 CLK 周期将 $\overline{LOCK}$ 降为低电位。等到了 LOCK prefix 的下一指令最后一个总线周期结束后,$\overline{LOCK}$ 才升到高电位。当 $\overline{LOCK}$ 为低电位时,任何在 $\overline{RQ}/GT$ 端的请求信号(其他 bus master 想要控制总线)会先被记录起来。直到 LOCK 结束后,这个信号才被处理。

8088 和 8086 的比较

8088 是一个 8 位处理机,但是它却和 8086 有类似的内部 16 位结构。大部分 8088 内部的功能和 8086 的功能都没有区别。8088 对外部总线处理的方法和 8086 相同。二者的区别在于 8088 每次只处理 8 位。它们内部寄存器和指令都没有不一样的地方,因此软件是完全兼容。

在内部,8088 和 8086 有三点区别,而这些区别都和 8088 的 8 位总线接口有关。以下列出这三种区别:

(1)8088 内部 Queue 长度只有 4 个 bytes,但是 8086 却有 6 个 bytes(或 3 个 words)缩短 Queue 长度的目的是为了避免 BIU 因为 prefetch 指令而造成滥用总线的情况。因为用 8 位 prefetch 需要更多的时间,所以这种顾虑是必要的。

(2)为了更进一步使 Queue 更有效率,8088 和 8086 的 prefetch 方式有些区别。每当内部 Queue 有一个 byte 空时,8088 的 BIU 则 prefetch 一个 byte 到 Queue 中。8086 却需要等到 Queue 有两个 byte 的空间才做 prefetch。

(3)有些指令的执行时间受 8 位总线接口影响。所有到存储器存取 16 位数据的指令需要额外 4 个 CLK 周期(无 wait state)来完成。此外,8088 也可能受 prefetch 的速度所影响。当然,这是指 CPU 执行一连串简单指令的时候。当 8088 执行比较复杂的指令,BIU 则能够在 EU 需要从 Queue 读入数据前将数据填好。

1-2 主机板布局

除了微处理器外,个人电脑的构成还需要其他外围电路,在这一节里,我们来看看 IBM PC 每一部分元件的位置。如图 1-11 所示。

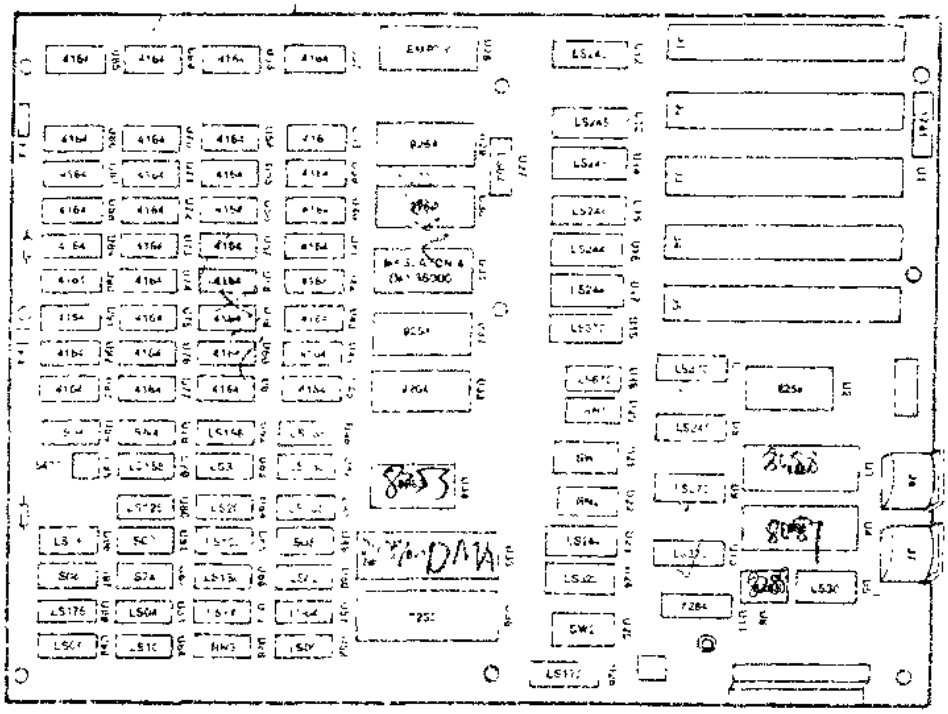


图 1-11 IBM PC 主机板电路元件位置图

这是一张 IBM PC 主机元件配置图(你可以对照实际的电路)。在图 1-11 的右下角有两个长方形的方块,左边那一个代表 40 脚的 IC-8088,右边也是 40 脚的 IC 叫做 8087,通常在购买 IBM PC 时这个位置并没有插入 8087,你可以视需要购买。8087 是一种浮点运算处理机,我们刚刚曾说过 IBM PC 所有的运算都在 8088 上执行,但是 8088 所做的运算只限于整数运算,如果要做实数运算(或称为浮点运算),一种方法是用软件模拟,就是把整数部分和小数部分分开处理,再把结果组合起来,这种软件的浮点运算需要的指令多而且速度慢。另一种方法是使用 8087 浮点处理机,利用 8087 指令,可以进行很精确的浮点计算,这些运算都由 8087 内部电路来执行。

在主机板中央有一排 ROM (Read Only Memory),在这几片 ROM 内储存的是 cassette BASIC 和 ROM BIOS (基本输入 / 输出系统)。ROM BIOS 直接控制硬件,我们可以不管控制硬件的细微动作,只要调用 ROM BIOS,由 BIOS 来完成,这一点帮我们省去了写输入 / 输出驱动程序(I / O Driver)的时间。此外,ROM BIOS 也完成开机初始化和自诊断(self-test)的工作。

在 ROM 的下方有 4 排整齐排列的 RAM,每一个 IC 插座都可以插入一个 4164(64K × 1),因此全部插满后可以有 256Kbyte 的存储器。在每排的最后一个 RAM 是用来存 parity,利用 parity 我们可以知道所存的数据是否有错,这部分功能以后再详述。

其他的电路元件主要位于 RAM 的右方,部分散置于 8080 和 8087 的四周,这些元件大部分都是 SSI,也有几片是 LSI。这些 LSI 包括 8253 可编程计时 / 计数器,8237 可编程 DAM 控制器,8235 可编程外部接口和 8259 可编程中断控制器。另外有两个辅助 8088 的芯片,一个是 8288 总线控制器,另一个是 8284 时序产生器,有了这些芯片,才使 IBM PC 成为一个完整的系统。

最后,我们看到主机板的左上方有 5 个 62 脚的扩充槽插座,每个扩充槽插座上提供了

足够的信号使接口卡能和IBM PC连在一起，因此由这些扩充槽插座，我们可以设计适合我们需要的硬件接口，使得IBM PC变成一种特殊的控制器。例如：马达转速控制、机械手臂控制……等等。IBM公司本身也推出了一些标准的接口卡让使用者选购，其中单色显示或彩色图形接口卡是基本必备的(IBM PC的视频电路放在接口卡上，而Apple II则把视频电路放在主机板上，这是二者不同的地方)。

1-3 扩充槽工作原理

Apple II成功的因素之一是有扩充能力，IBM PC也继承了这一特点。因此，当用户购买了IBM PC后，等于拥有了一个多功能的个人电脑，只要把某种接口卡插入任一个扩充槽内，IBM PC就具备了不同的功能。

所有扩充槽都是相同的，因此任何一个扩充槽上某个引脚的信号有了变化，在其他的扩充槽所对应的引脚上也同时可以看到这个变化。所以扩充的接口卡能插在5个扩充槽(PC-XT有8个扩充槽)上的任何一个。

每个扩充槽上的62根信号线可以分为四类。第一类有8根信号线，提供了系统的电源，包括几种不同的电压标准。

第二类的8根信号线用来传送数据总线上的8位数据，不论从主机传到接口卡或者由接口卡传入主机，所有数据都经过这个总线。这样主机才能和接口卡上的存储器或输入/输出设备传递数据。

另外有20根信号线专门用来识别地址，也称为地址线(address line)。因为当数据在数据总线上传送时，处理机必须指明这些数据传送至何处，或由何处传入。经这20根地址线，主机才能和接口卡上的某个特定的存储器单元或某个特定的输入输出设备交换数据。当这20根信号线用来指定存储器的地址时，寻址空间为1024K (2^{20}) bytes；用于指定输入/输出设备地址时，只有低位16根地址线是有效的，也就是最大寻址空间为64K (2^{16}) bytes。

其余的信号线用来传送基本控制信号，如：存储器read信号、存储器write信号、输入/输出read信号、输入/输出write信号、系统时序、Ready信号……等等。

在扩充接口卡上需要和主机交换数据的设备有两种，一种是存储器，另一种是输入/输出设备。如果在某一个时间输入/输出read信号发生变化，代表处理机想从接口卡上的输入/输出设备read数据，由地址线指定的设备侧到输入/输出read信号后，就把数据放在数据总线上等处理机读取，在这段时间内存储器不响应这些信号，因此存储器保持原来状态。同理，当存储器read信号有变化时，由地址线指定的存储器单元会把数据放在数据总线上，同样，这时候输入/输出设备也不响应这些信号。按同样的道理，我们也可以得到输入/输出write和存储器write时的情形。

一个接口卡设计者在设计他的电路前，必须详细地分析扩充槽上每一根信号线的规格，也必须了解某个信号和另一个信号的时序关系，例如：在存储器read时，必须了解在地址有效后，存储器read信号何时有效，这样才能决定接口卡上的存储器何时将数据送到数据总线上，也才能决定何时结束存储器读取周期。如果不按规格设计，可能会造成数据无法正确读取的情况。

1-4 数值处理机 8087

概述

8087 是一个和 8088 共同操作的协处理机(coproccessor)。它能够对多种数值数据类型做算术和比较运算。它还能执行多超越函数(transcendental functions),例如: tangent 和 log 函数。8087 和 8088 配合工作时,8088 必须被设定为 MaximumMode。使用 8087 就相当于给 CPU 增加了额外的寄存器(参考图 1-12)、指令组和新的数据类型。程序设计师通常并不把 8087 看作另一个独立设备;相反,在他看来,CPU 的计算能力似乎增加了许多。图 1-13 是 8087 的 IC 引脚图。

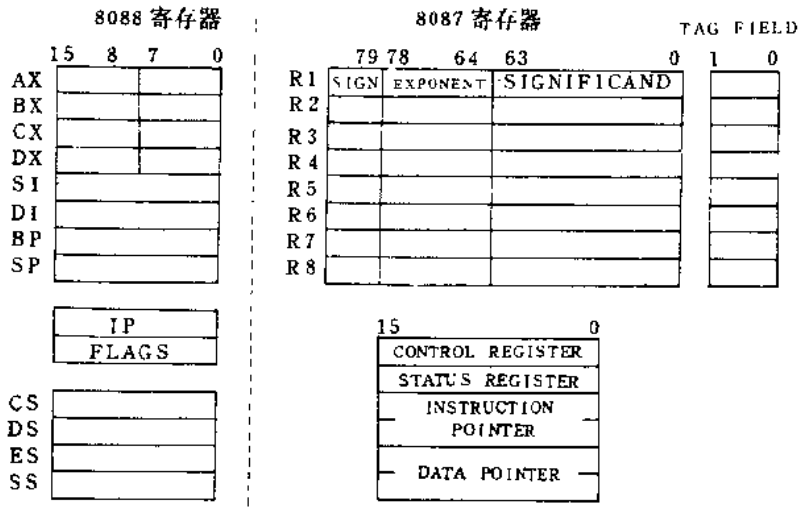


图 1-12 8088 与 8087 寄存器

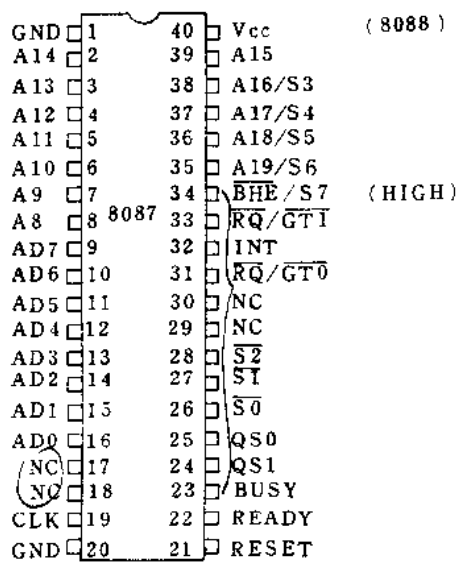


图 1-13 8087 引脚图

8087 和 8088 的系统构成

8087 可以直接和 8088 连接在一起工作。我们从二者 IC 引脚图(图 1-2,图 1-13)中的信号名称可看出这一点。8088 的状态(S0-S2)和 Queue 状态信号(QS0-QS1)使 8087 可以同步地监视 8088 并译码 8088 的指令。由于 8087 的操作不影响 8088 的工作,因此不会降低 8088 执行效率。一旦 8087 启动以后,它能够和 8088 同时运行。

二者的同步由 BUSY 信号和 WAIT 指令实现指。8087 可以用 BUSY 信号告诉 8088 当前它正执行某个指令,8088 则可以执行 WAIT 指令,等待 8087 执行完当前指令而继续下步工作。如果 8087 产生错误或 exception 时,它会对 CPU 发出中断信号。这个中断信号先被送到 8259 中断控制器,再由 8259 对 8088 发出中断请求。

当 8087 要进行数据传送时,它可以用 8088 的 request / grant 信号线(通常是 $\overline{RQ}$ / $\overline{GT1}$)请求总线控制权。另外,系统内其他 bus master 也可以连接到 8087 的 $\overline{RQ}$ / $\overline{GT1}$ 上来请求总线控制权。如 8087 目前并没有控制本地总线,则它会将此 bus master 的 request / grant 信号送给 8088 来处理;如果 8087 目前控制本地总线,则它直接把本地总线控制权交给此 master。在这种方式中,此 bus master 比 8087 有较高的优先权来使用本地总线。

总线操作

8087 总线结构、操作、时序和 8088(Maximum Mode)是相同的。在地址 / 数据总线 AD0-AD7 上,地址和数据是以分时多路方式送出。地址总线 A8-A15 不是以多路方式出现,因此不需要用地址闩将其锁住。A16-A19 和 4 个状态信号 S3-S6 是以分时多路方式送出。其中 S3、S4 和 S6 在 8087 控制总线时都保持在高电位,而 S5 则保持在低电位。8087 在监视 CPU 总线周期时,就是靠 S6 来分辨 8088 和其他 local bus master 的操作(只有 8088 能够使 S6 变为低电位)。

8288 用状态信号或 $\overline{S0}$-$\overline{S2}$ 来决定目前应执行的总线周期。各种状态如下所示:

$\overline{S2}$	$\overline{S1}$	$\overline{S0}$	
0	X	X	不使用
1	0	0	
1	0	1	memory data read
1	1	0	memory data write
1	1	1	passive(无总线周期)

8087 内部的结构

如图 1-14 所示,8087 内部被划分为两个单位:控制部件(control unit,简称 CU)和数值处理部件(numeric execution unit,简称 NEU)。当 CU 接受并解释指令、存取存储器操作数及执行控制指令的时候,NEU 则执行所有的数值运算指令。两个部件可以互相独立地操作。因此,在 CU 和 CPU 保持同步的时候,NEU 则执行数值运算指令。

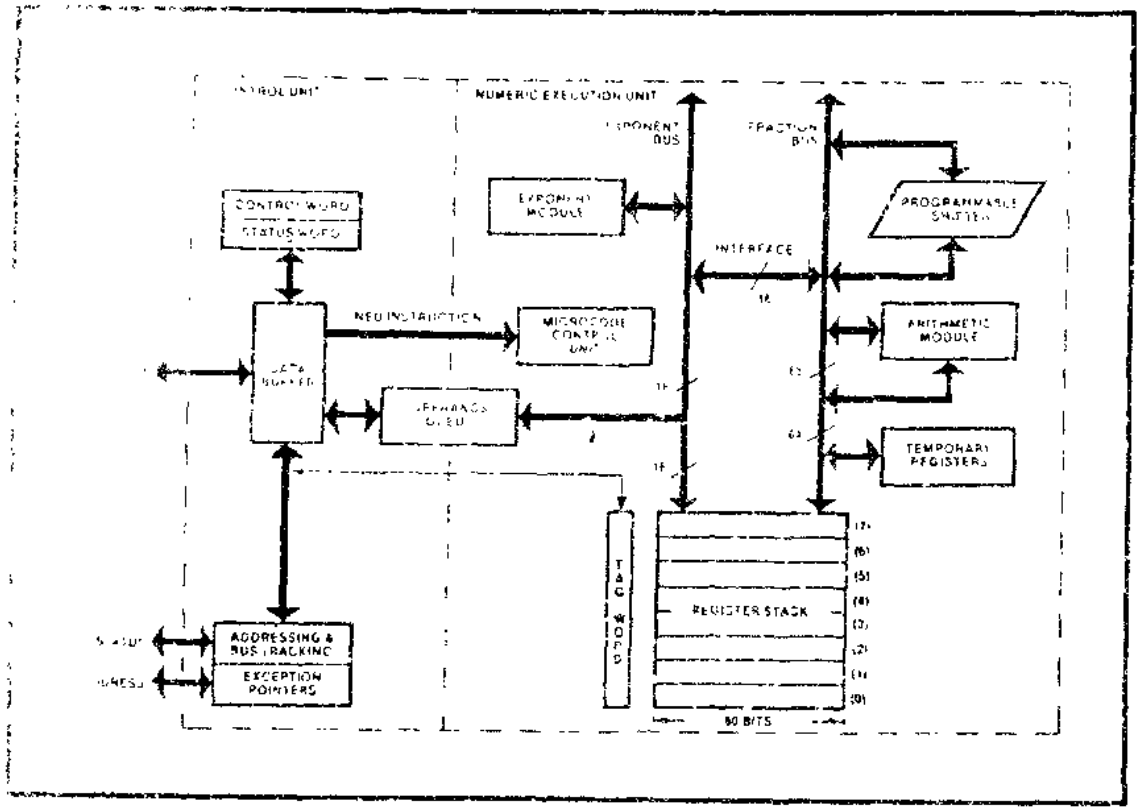


图 1-14 8087 内部结构图

控制部件

CU 能够使 8087 和 CPU 保持同步。8087 的指令和 CPU 的指令是混合在同一个指令流(instruction stream)内的。当 CPU 从存储器取指令的时候，8087 靠监视 CPU 发出的状态信号($\overline{S0}$–$\overline{S2}$, S6)，决定何时 8087 的指令才由存储器中读出。CU 同时监视数据总线以便读取属于 8087 的指令。

就 8088 来说，一个 8087 的数值运算指令就是它的一个 ESCAPE 指令。8088 和 8087 可以同时译码并执行 ESCAPE 指令。8087 只认识数值运算指令。当 CPU 执行 ESCAPE 指令时就代表一个数值运算的开始。这个数值运算指令可能有存储器操作数，也可能没有。

CPU 负责辨别有存储器操作数和无存储器操作数的 ESC (ESCAPE)指令。如果某指令欲存取存储器操作数，则 CPU 利用它所提供的寻址方法，把操作数的实际地址计算出来。接着，它对此地址发出一个"假 read (dummy read)"。这个"假 read"和一般 read 周期并没有什么两样的，但是，CPU 忽略它所读到的数据。如果 ESC 指令没有存储器操作数，CPU 就继续执行下一条指令。

一个 8087 指令有下列三种存取存储器的方式：(1)不访问存储器。(2)从存储器中读操作数到 8087 内。(3)从 8087 中取操作数存入存储器。如果 8087 不访问存储器的话，它只执行这条指令。但是，假如它必须访问存储器，CU 则利用 CPU 的"假 read"周期记住 CPU 放在总线上的地址。假如这条指令是 load 指令，CU 读入在数据总线上的数据(在"假 read"周期内)。如果所需的数据比读入的数据还要长的话，CU 立刻占用总线控制权(用 request / grant)，而在下面几个总线周期读取剩余的数据。假如这条指令是 store 指