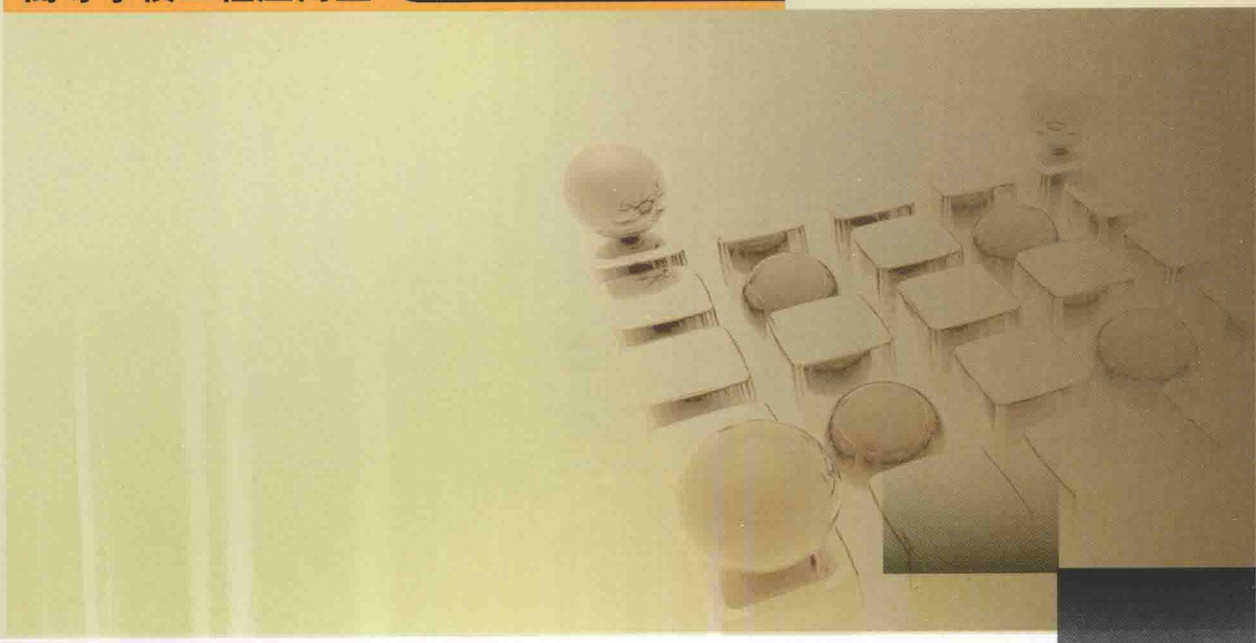
高等学校工程应用型“十二五”系列规划教材



微机原理与接口技术

饶志强　钮文良　编著

科学出版社

高等学校工程应用型“十二五”系列规划教材

微机原理与接口技术

饶志强　钮文良　编著

科学出版社

北京

内 容 简 介

本书系统地介绍了80X86 PC机的原理、汇编语言程序设计及接口技术。主要内容包括8086微处理器、8086指令系统、汇编语言基本语法、汇编语言程序设计、半导体存储器、中断系统、常用可编程接口芯片、嵌入式系统等内容。

本书内容精炼、实例丰富，其中大量的接口电路和程序是作者多年来在科研和教学中反复提炼得来的，因而本书应用性强、突出工程实际应用，可作为大专院校和高职高专的高等教育“汇编语言程序设计”“微机原理及应用”“接口技术”“微机原理与嵌入式系统”等课程的教学用书。

本书可作为电气、电子、通信、机电一体化、生物医学工程、物联网工程等工程应用型本科教材，也可作为应用工程师的参考书。

图书在版编目(CIP)数据

微机原理与接口技术 / 饶志强，钮文良编著. —北京：科学出版社，2015.9
高等学校工程应用型“十二五”系列规划教材
ISBN 978-7-03-045603-8

Ⅰ. ①微… Ⅱ. ①饶… ②钮… Ⅲ. ①微型计算机－理论－高等学校－教材 ②微型计算机－接口技术－高等学校－教材 Ⅳ. ①TP36

中国版本图书馆CIP数据核字(2015)第212028号

责任编辑：潘斯斯 张丽花 / 责任校对：桂伟利
责任印制：霍 兵 / 封面设计：迷底书装

科学出版社 出版
北京东黄城根北街16号
邮政编码：100717
http://www.sciencep.com

新科印刷有限公司 印刷

科学出版社发行 各地新华书店经销

*

2015年9月第 一 版 开本：787×1092 1/16
2015年9月第一次印刷 印张：19 1/2
字数：496 000

定价：46.00元

（如有印装质量问题，我社负责调换）

前　言

“微机原理与接口技术”是高等院校电子信息科学与技术、计算机科学与技术专业的一门专业基础课，也是电子信息工程、通信工程、自动化等专业的必修课程。本书的任务是使学生从系统的角度出发，掌握微机系统的基本组成、工作原理、接口电路及应用方法，以提高其计算机硬件和软件知识，并将硬件和软件有机结合起来，培养其分析和设计微机应用系统的能力，并最终掌握微机系统的开发技巧。

本书以国家教育部计算机专业和电气、电子信息专业微机原理类课程教学大纲为基础，立足于本书教学内容和课程体系的改革，面向电子信息专业人才市场，以培养信息类专业高水平、高质量的工程技术人才为目标，参考了国内外大量的文献资料和相关教材，吸取各位专家之长，内容深入浅出、重点突出、条理清晰、通俗易懂、实用性强，每章后均附有习题。

全书分 8 章，内容包括：8086/8088 微处理器、8086 指令系统、汇编语言基本语法、汇编语言程序设计、半导体存储器、中断系统、常用可编程接口芯片、嵌入式系统等。书中内容注重理论和实践相结合，力求做到既有一定的理论基础，又能运用理论解决实际问题；既掌握一定的先进技术，又着眼于当前的应用服务。

本书可作为高等院校电类专业“微型计算机原理”和“嵌入式设计基础”等课程的教材，也可作为广大微机系统设计爱好者的入门读物。

本书由饶志强、钮文良编著，负责总体设计和统稿；梁家海、路铭、陈景霞、肖琳、申海伟参编，采用集体讨论、分工编写、交叉修改的方式进行。

本书的编写大纲及内容由李哲英教授审阅，朱定华教授对本书的出版给予了极大的关注和支持，提出了宝贵的建设性意见，在此表示衷心的感谢！本书编写得到了北京联合大学应用科技学院的教师和有关领导的大力支持。本书编写时参考了诸多书籍，在此对参考文献的作者表示感谢！最后，感谢科学出版社各位编辑为本书的出版倾注的大量的心血和热情，也正是他们前瞻性的眼光，才让读者有机会看到本书。

由于本书的编写风格和内容结构是一种新的尝试，加之作者水平有限，若书中存在疏漏之处，欢迎读者批评指正。读者可通过电子邮箱 yykjtzhiqaing@buu.edu.cn 与作者进行交流。

作　者

2015 年 7 月

目　录

第 1 章　8086/8088 微处理器

教学提示：尽管微处理器已进入了 Pentium 时代，其内部结构和性能也发生了巨大的变化，但其基本结构仍然和早期的 8086/8088 相似，可以说 8086/8088 是 80X86 系列芯片的基础。本章就以 8086/8088 为例介绍微处理器的总体结构。

教学要求：通过本章的学习，读者可以了解 8086/8088 微处理器的内部结构、引脚和工作方式、存储器组织和工作时序。

注意：本章的学习是比较枯燥的一章，但论述的内容是为后续的学习做好基础性工作。内容不难，但是内容较多，基础知识较多，读者在学习中是不容易记住的。本章学习的最大难点就是记不住。所以在学习本章的内容时，希望课前预习，课后一定要多看几遍内容，同行之间要多多讨论，书后思考题和习题希望读者尽可能都做。

1.1　8086/8088 微处理器的内部结构

8086/8088 是 Intel 系列的 16 位微处理器，它是采用 HMOS 工艺制造的，内部包含约 29000 个晶体管，用单一的+5V 电源，时钟频率为 5～10MHz。

8086 有 16 根数据线和 20 根地址线，其寻址空间达 1MB；8088 是一种准 16 位微处理器，它的内部寄存器、内部运算部件以及内部操作都是按 16 位设计的，但对外的数据总线只有 8 条。8086/8088 芯片内设有硬件乘除指令部件和串处理指令部件，可对位、字节、字、串、BCD 码等多种数据类型进行处理。

1.1.1　总线接口单元和执行单元

8086/8088 CPU 采用了全新的指令流水线结构(Instruction Pipeline)。从功能上看，它由两个独立的逻辑单元组成，即总线接口单元(Bus Interface Unit，BIU)和执行单元(Execute Unit，EU)。内部结构如图 1-1 所示。

1. 总线接口单元

BIU 的功能是 8086 CPU 与存储器或 I/O 设备之间的接口部件，负责全部引脚的操作。具体来说，BIU 负责产生指令地址，根据指令地址从存储器取出指令，送到指令队列中排队或直接送给 EU 去执行；BIU 也负责从存储器的指定单元或外设端口中取出指令规定的操作数传送给 EU，或者把 EU 的操作结果传送到指定的存储单元或外设端口中。

BIU 内部设有 4 个 16 位的段寄存器：代码段寄存器(Code Segment，CS)、数据段寄存器(Data Segment，DS)、堆栈段寄存器(Stake Segment，SS)和附加段寄存器(Extra Segment，ES)，还有一个 16 位的指令指针寄存器(Instruction Pointer，IP)，一个 6 字节指令队列缓冲器，20 位地址加法器和总线控制电路。

1)指令队列缓冲器

该缓冲器是用来暂存指令的一组暂存器，它由 6 个 8 位寄存器组成(8088 为 4 个)，最多

可同时存放 6 个字节的指令。采用“先进先出”(First in First out，FIFO)原则，按顺序存放，再按顺序被取到 EU 中去执行。它遵循以下原则。

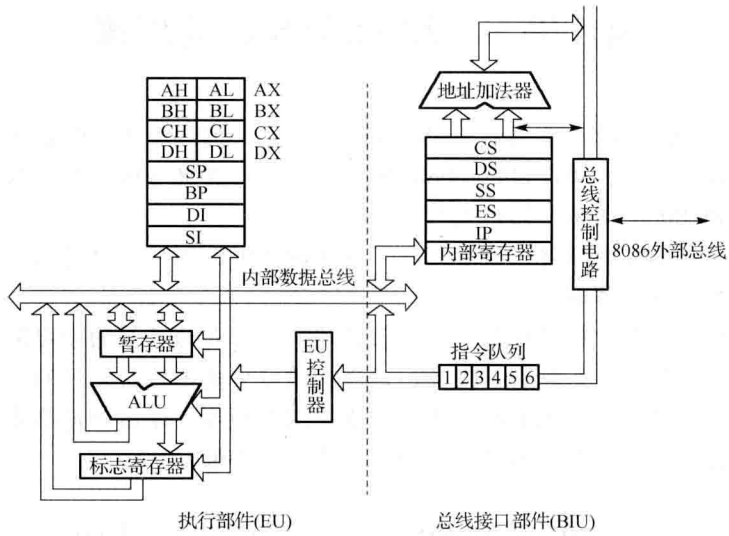


图 1-1　8086 CPU 内部结构

(1)取指令时，将指令存入队列缓冲器，缓冲器中只要有一条指令，EU 就开始执行。

(2)缓冲器中只要有一个字节没装指令，BIU 便自动执行取指操作，直到填满为止。

(3)在 EU 执行指令的过程中，当指令需要对存储器或 I/O 设备进行数据存取时，BIU 将在执行完当前取指的存储器周期后的下一个存储器周期，对指定的存储器单元或 I/O 设备进行存取操作，交换的数据通过 BIU 送 EU 进行处理。

(4)当 EU 执行完转移、调用和返回指令时，要清除队列缓冲器，同时，BIU 需从新的地址重新开始取指，新取的第一条指令将直接送 EU 执行，随后取的指令将填入指令队列。

2)20 位的地址加法器

地址加法器用来产生 20 位地址。8086 有 20 根地址线，可寻址 1MB，但内部所有的寄存器都是 16 位的，所以需要一个部件来根据 16 位寄存器提供的信息计算出 20 位物理地址，这个部件就是 20 位的地址加法器。地址计算公式是

物理地址=段基址值×16+偏移地址值

3)指令指针寄存器

指令指针寄存器的功能相当于程序计数器 PC，用于存放 EU 要执行的下一条指令的偏移地址。程序不能直接对 IP 进行存取，但可在程序运行中自动修正，使之指向要执行的下一条指令。每取一条指令字节，IP 自动加 1。

4)总线控制电路

8086 分配 20 条总线，用来传送 16 位数据信号、20 位地址信号和 4 位状态信号。这就需要分时进行传送。总线控制电路的功能就是以逻辑控制方法实现上述信号的分时传送。

2. 执行单元

执行单元 EU 包括一个 16 位的算术逻辑单元 ALU、一个 16 位的状态标志寄存器、一组数据暂存寄存器、一组指令指针和变址寄存器，一个数据暂存器和 EU 控制电路。

执行单元 EU 的功能是从 BIU 的指令队列中取出指令代码，然后执行指令所规定的全部功能。在执行指令的过程中，如果需要向存储器或 I/O 传送数据，则 EU 向 BIU 发出访问存储器或 I/O 的命令，并提供访问的地址和数据。

1) 算术逻辑单元

算术逻辑单元(Arithmetic and Logic Unit，ALU)可以用来进行算术、逻辑运算，也可以按指令的寻址方式计算出寻址单元的 16 位偏移地址，并将其偏移地址送到 BIU 中形成一个 20 位的物理地址。ALU 只能进行运算，不能存放数据。在运算时数据先传送到暂存器中，再经 ALU 运算处理。运算后，运算结果经内部总线送回累加器或其他寄存器，或者存储单元中。

2) 状态标志寄存器

状态标志寄存器(Flags，F)用来反映 CPU 运算后的状态特征或存放控制标志。

3) 数据暂存寄存器

数据暂存寄存器协助 ALU 完成运算，对参加运算的数据进行暂存。

4) 通用寄存器组

通用寄存器包括 8 个 16 位寄存器。其中 AX、BX、CX、DX 为数据寄存器，它们既可以存放 16 位数据，也可分为两半，分别寄存 8 位数据；SP(Stack Pointer)为堆栈指针，用于堆栈操作；BP(Base Pointer)为基址指针寄存器，用来存放位于堆栈段中的一个数据区基址的偏移量；SI(Source Index)源变址寄存器和 DI(Destination Index)目的变址寄存器，用来存放被寻址地址的偏移量。

5) 控制单元

控制单元接收从 BIU 指令流队列中来的指令，经过解释、翻译形成各种控制信号，对 EU 的各个部件实现在规定时间内完成规定操作。

1.1.2　8086 CPU 内部寄存器

Intel 8086/8088 CPU 共有 14 个 16 位的寄存器，如图 1-2 所示。分别为 8 个通用寄存器(AX、BX、CX、DX、SP、BP、SI、DI)、2 个控制寄存器(IP、FLAGS)和 4 个段寄存器(CS、DS、SS、ES)。

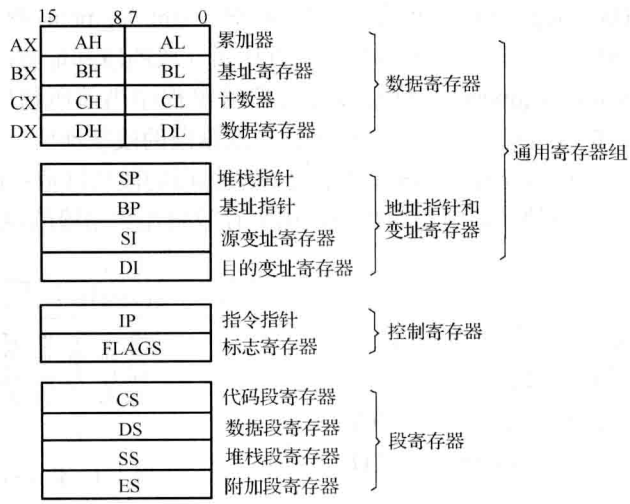


图 1-2　8086/8088 CPU 内部寄存器

1. 通用寄存器

通用寄存器共有 8 个，按照使用情况分为数据寄存器、指针寄存器和变址寄存器。

1) 数据寄存器

8086 的通用数据寄存器既可用作 16 位寄存器 AX、BX、CX、DX，也可用作 8 位寄存器 AL、AH、BL、BH、CL、CH、DL、DH。它们均可独立寻址，独立使用。数据寄存器(Data Register)主要用来存放操作数或中间结果，以减少访问存储器的次数。

通常选 AX(AH、AL)作为累加器(Accumulator)，但实际上，每个数据寄存器都具有累加器的功能，只不过对于同样的运算，使用 AX(AH、AL)比使用其他数据寄存器能得到更短的目标代码和更快的速度；BX(BH、BL)也叫基址寄存器(Base Register)，可用作间接寻址的地址寄存器；CX(CH、CL)也叫计数器(Count Register)，用来控制循环重复的次数；DX(DH、DL)是数据寄存器(Data Register)，用来扩展累加器。在进行乘法和除法运算时，DX 和 AX 联合存放 32 位的二进制数(AX 存放低 16 位，DX 存放高 16 位)。

2) 指针寄存器(Pointer Register)

SP(Stack Pointer)是堆栈指针寄存器，BP(Base Pointer)是基址指针寄存器，它们常用来指示相对于段起始地址的偏移量。BP 一般用于访问堆栈段的任意单元，SP 用于访问堆栈段的栈顶单元。

3) 变址寄存器(Index Register)

SI(Source Index)源变址寄存器和 DI(Destination Index)目的变址寄存器用来存放当前数据段的偏移地址。SI 存放源操作数的偏移地址，DI 存放目的操作数的偏移地址。

2. 段寄存器

由于在 8086 系统中，需要用 20 位物理地址访问 1MB 的存储空间，但 8086 的内部结构以及内部数据的直接处理能力和寄存器都只有 16 位，故只能直接提供 16 位地址寻址 64KB 存储空间。为了能够寻址 1MB 空间，8086 CPU 中引入了存储器地址空间分段的概念。

8086 CPU 的 BIU 中有 4 个 16 位的段寄存器，是用来存放段起始地址值(又叫段基址值)的，8086 的指令能直接访问这 4 个段，即 SL Code Segment 代码段寄存器、SS(Stack Segment)堆栈段寄存器、DS(Data Segment)数据段寄存器和 ES(Extra Segment)附加段寄存器。

CS(Code Segment)：代码段寄存器存放当前程序所在段的段基址值，CPU 执行的指令将从代码段获得。SS(Stack Segment)：堆栈段寄存器给出程序当前所使用的堆栈段基址值。DS(Data Segment)：数据段寄存器存放当前程序的主数据段的段基址值，一般来说，程序所用的数据是放在数据段中，而段寄存器 DS 中保存了该数据段的段基址值；ES(Extra Segment)：附加段寄存器指出程序当前使用的附加数据段，用来存放附加数据段的段基址值数据。

3. 状态寄存器

Intel 8086/8088 CPU 的处理器设置了一个 16 位的状态标志寄存器(Flag)。其中安排了 9 位作用标志位，包括 6 个状态标志位(CF、PF、AF、ZF、SF、OF)和 3 个控制标志位(TF、IF、DF)，如图 1-3 所示。

图 1-3　状态标志寄存器

由于该寄存器的标志位与指令队列中的指令代码一起参与对现行指令执行过程的控制，因此，我们可以把标志寄存器看成与指令队列缓冲器、指令指针寄存器一起实现计算机程序控制工作方式的一个辅助控制寄存器。

1) 状态标志

状态标志位用来反映 EU 执行算术或逻辑运算后的结果特征。

(1) CF (Carry Flag) 进位标志：在进行算术运算时，如果在最高位产生了进位或借位，则 CF= 1，否则 CF= 0。

(2) PF (Parity Flag) 奇偶标志：表明运算结果中“1”的个数是奇数还是偶数。若结果中有偶数个“1”，则 PF= 1；若结果中有奇数个“1”，则 PF= 0。

(3) AF (Auxiliary Carry Flag) 辅助进位标志：在运算时，如果在第 4 位产生了进位或借位，则 AF= 1，否则 AF= 0。

(4) ZF (Zero Flag) 零标志：如果运算结果为全“0”，则 ZF= 1，否则 ZF= 0。

(5) SF (Sign Flag) 符号标志：该位与运算结果的最高有效位相同，当运算结果为负时，SF= 1；结果为正时，SF= 0。

(6) OF (Overflow Flag) 溢出标志：当运算结果超出计算机用补码所能表示数的范围时，OF= 1，否则 OF= 0。

2) 控制标志

控制标志位用来控制 CPU 的操作，它由程序设置或由程序清除。

(1) TF (Trap Flag) 单步标志：TF 是为了使程序调试方便而设置的。若 TF= 1，8086 CPU 处于单步工作方式，即每执行完一条指令就自动地产生一个内部中断，转去执行一个中断服务程序；当 TF= 0 时，8086 CPU 正常执行程序。

(2) IF (Interrupt-Enable Flag) 中断允许标志：用指令 STI 可使 IF= 1，8086 CPU 开中断，允许接受外部从 INTR 引脚发来的可屏蔽中断请求；用指令 CLI 使 IF= 0，8086 CPU 关中断，不能接受外部从 INTR 引脚发来的中断请求。

(3) DF (Direction Flag) 方向标志：DF 用来控制数据串操作指令的方向。用指令 STD 使 DF= 1，控制串操作指令将串操作数的地址偏移量以递减的方式改变，从高地址到低地址的方向对串操作数据进行处理；用指令 CLD 使 DF= 0，则控制串操作指令将以递增的顺序从低地址到高地址的方向对串操作数进行处理。

4. 指令指针寄存器

指令指针寄存器 (Instruction Pointer，IP) 相当于程序计数器 PC，用于控制程序中指令的执行顺序。IP 中的内容是下一条待取指令的偏移地址，每取一次指令，IP 内容就自动加 1，从而保证指令按顺序执行。IP 实际上是指令机器码存放单元的地址指针，IP 的内容可以被转移指令强制改写。但程序不能直接访问 IP，即不能用指令去取出 IP 的值或给 IP 赋值。

1.2　8086/8088 的引脚和工作方式

CPU 的功能越强，需要的引脚就越多，但由于受当时集成电路制造工艺的限制，芯片的引脚不可能做得很多。为了解决功能强和引脚少的矛盾，8086/8088 CPU 内采用了引脚复用技术，使部分引脚具有双重功能。

1.2.1　8086/8088 CPU 引脚特性

8086/8088 CPU 采用双列直插式的封装形式，有 40 条引脚，如图 1-4 所示。由于受集成电路制造工艺的限制，芯片的引脚不能做得很多。为了解决功能与引脚的矛盾，8086/8088 CPU 采用了分时复用的地址/数据总线，所以有一部分引脚具有双重功能。为了适应不同的应用环境，8086/8088 CPU 有两种工作方式：最大方式($\overline{MX}$)和最小方式(MN)，这由引脚 33(MN / $\overline{MX}$)加以控制。最小方式适用于单微处理器组成的小系统，在这种系统中，所有的总线控制信号都直接由 8086/8088 产生。最大方式适用于多微处理器组成的大系统，它包含两个或多个微处理器，其中一个就是 8086/8088，称为主处理器，其他处理器则称为协处理器。

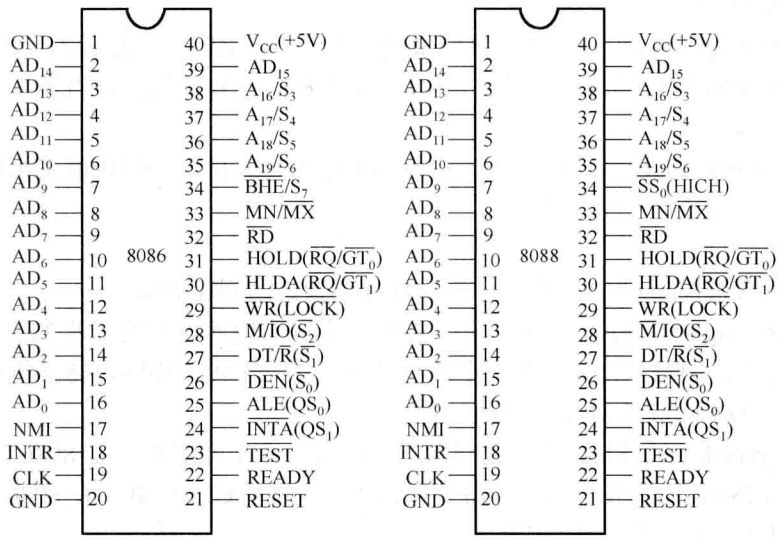


图 1-4　8086/8088 的引脚信号(括号中为最大方式引脚名)

CPU 的许多引脚在设计时都采用了三态门逻辑输出电路。即输出信号除了逻辑状态 1 和逻辑状态 0 以外，还有第三种状态——高阻状态(也叫悬空状态)。当输出为高阻状态时，表示芯片已放弃了对引脚的控制，该引脚所连接的设备就可以接管对它的控制了。

1. 地址/数据总线 AD_{15} ~ AD_0

地址数据总线(Address Data Bus)是分时复用的存储器或 I/O 端口地址和数据总线，双向工作，三态。它在总线周期的 T_1 状态作为地址线，输出要访问的存储器或 I/O 端口的地址；在总线周期的 T_2~T_3 状态作为数据线传输数据。

2. 地址/状态线 A_{19}/S_6 ~ A_{16}/S_3(Address Status)

地址/状态复用引脚，输出，三态。这些引脚在总线周期的 T_1 状态，用来输出地址的最高 4 位，在总线周期的 T_2、T_3、T_W 和 T_4 状态时，用来输出状态信息。

当 S_6 为 0 时，表示 8086/8088 CPU 当前与总线连接。S_5 表明中断允许标志的当前设置，为 1 时，表示当前允许可屏蔽中断请求，为 0 则禁止一切可屏蔽中断请求。状态信号中的 S_4 和 S_3 用来指示当前使用哪一段寄存器，具体规定如表 1-1 所示。

表 1-1　当前使用的段寄存器的指示

S_4	S_3	当前正在使用的段寄存器	S_4	S_3	当前正在使用的段寄存器
0	0	ES	1	0	$\overline{CS}$ /不需要用段寄存器
0	1	SS	1	1	DS

3. 控制总线

(1) $\overline{BHE}$ / S_7 (Bus High Enable/Status)。高 8 位数据总线允许/状态复用引脚，输出，三态。在总线周期的 T_1 状态，8086 在 $\overline{BHE}$ / S_7 引脚输出 $\overline{BHE}$ 信号，$\overline{BHE}=0$ 表示高 8 位数据线 D15～D8 上的数据有效；在 T_2、T_3、T_W 和 T_4 状态，$\overline{BHE}$ / S_7 引脚输出状态信号 S_7，在 8086 设计中，S_7 为备用信号，其内容不固定。

通过 $\overline{BHE}$ 信号和 A_0 的组合就可以告诉连接在总线上的存储器，当前的数据在总线上将以何种形式出现。表 1-2 已归纳出 4 种读/写格式。

表 1-2　$\overline{BHE}$ 和 A_0 信号的意义

$\overline{BHE}$	A_0	操　　作	所使用的数据引脚
0	0	从偶地址开始读/写一个字	AD_{15}～AD_0
1	0	从偶地址开始读/写一个字节	AD_7～AD_0
0	1	从奇地址开始读/写一个字节	AD_{15}～AD_8
1	0	从奇地址开始读/写一个字(需要两个周期完成，第一个周期将低 8 位数据送 AD_{15}～AD_8，第二个周期将高 8 位数据送 AD_7～AD_0)	AD_{15}～AD_8 AD_7～AD_0
1	1	无存取操作	

从表 1-2 中可以看出，在 8086 系统中，如果要读/写从奇地址单元开始的一个字，需要用两个总线周期。

(2) $\overline{RD}$ (Read)。读信号输出，三态。$\overline{RD}$ 信号指出将要执行一个对存储器或 I/O 端口的读操作。低电平有效，在一个执行读操作的总线周期中，$\overline{RD}$ 信号在 T_2、T_3 和 T_W 状态均为低电平。

(3) READY。等待状态控制(输入，高电平有效)，表示数据传送已结束的信号。8086 CPU 与存储器或 I/O 相配时，当 CPU 发出读/写操作，而后者速度慢，来不及响应时，CPU 通常在 T_3 之后，检测 READY 引脚上的信号。如果 READY 为 0，则自动插入一个等待时钟周期，然后检测 READY 的状态，如果还是 0，则再插入等待周期，直到 READY=1 时为止；当 READY = 1 时，即通知 CPU 数据传输完成，结束等待而进入 T_4 状态。

(4) $\overline{TEST}$。等待测试(输入，低电平有效)，当 CPU 在执行 WAIT 指令时，每隔 5 个时钟周期对该线的输入进行一次测试。$\overline{TEST}=1$ 时，CPU 进入等待，重复执行 WAIT 指令，直到 $\overline{TEST}=0$，再继续执行 WAIT 后的下一条指令，等待期间允许外部中断。

(5) INTR (Interrupt Request)。屏蔽中断请求信号输入端，高电平有效。CPU 在执行每条指令的最后一个时钟周期时，会对 INTR 引脚的信号进行采样。若 CPU 的中断允许标志为 1，且又接收到 INTR 信号，则 CPU 会在执行完当前指令后，响应中断请求，执行一个中断处理子程序。

(6) NMI (Non-Maskable Interrupt)。非屏蔽中断请求信号输入，NMI 不受中断允许标志 IF 的影响，也不能用软件进行屏蔽。每当 NMI 端输入一个正沿触发信号时，CPU 会在执行完当前指令后，执行对应的不可屏蔽中断处理程序。

(7) RESET。复位信号输入，高电平有效。RESET 将使 8086 CPU 立即结束当前操作。CPU 内部进入复位工作。CPU 要求复位信号至少要保持 4 个时钟周期的高电平，才能结束正

在进行的操作。当 RESET 信号变为低电平时，CPU 就开始执行再启动过程。复位后，CPU 内部各寄存器的状态如表 1-3 所示。

表 1-3　复位后 CPU 中寄存器状态

寄存器	内容	寄存器	内容
状态标志寄存器 F	清零	堆栈段寄存器 SS	0000H
指令指针 IP	0000H	附加段寄存器 ES	0000H
代码段寄存器 CS	FFFFH	指令流队列	清空
数据段寄存器 DS	0000H		

(8) CLK (Clock)。系统时钟输入。8086/8088 要求时钟信号的占空比为33%，即 1/3 周期为高电平，2/3 周期为低电平，即时钟信号的低、高之比采用 2:1 时为最佳状态。

(9) MN / $\overline{\text{MX}}$。最小/最大方式信号输入。当 MN / $\overline{\text{MX}}$ 接 +5V 电压时，CPU 工作于最小方式；接地时，CPU 工作于最大方式。

4. 电源线 V_{CC} 和地线 GND

电源线 V_{CC} (第 40 脚)接入的电压为 +5V，第 1 脚、第 20 脚为地线 GND，均应接地。

1.2.2　最小/最大工作方式

1. 8086 最小工作方式时的引脚功能

8086/8088 CPU 最小工作方式用于单片微处理器组成的小系统。在这种方式中，由 8086/8088 CPU 直接产生小系统所需要的全部控制信号。当 MN / $\overline{\text{MX}}$ 接+5V 电压时，CPU 工作于最小方式，引脚 24～31 的控制功能如下。

(1) $\overline{\text{INTA}}$ (Interrupt Ackonwledge)。中断响应信号，输出。该引脚用来对外设的中断请求 INTR 做出响应。此后 CPU 进入中断响应周期，该周期由两个连续的典型总线周期构成，$\overline{\text{INTA}}$ 信号是位于连续总线周期中的两个负脉冲，在 T_2、T_3 和 T_W 状态为低电平，第 1 个负脉冲通知外部设备的接口，它发出的中断请求已获允许；外设接口收到第 2 个负脉冲后，往数据总线上放中断类型码，中断类型码代表中断源，从而使 CPU 得到了有关此中断请求的详尽信息。

(2) ALE (Address Latch Enable)。地址锁存允许信号，输出，三态。在任何一个总线周期的 T_1 状态，ALE 输出有效电平(高电平)，表示当前在地址/数据复用总线上输出的是地址信息，地址锁存器 8282/8283 将 ALE 作为锁存信号，对地址进行锁存。

(3) $\overline{\text{DEN}}$ (Data Enable)。数据允许信号，输出，三态，低电平有效。在用 8286/8287 作为数据总线收发器时，$\overline{\text{DEN}}$ 为收发器提供一个控制信号，表示 CPU 当前准备发送或接收数据，总线收发器将 $\overline{\text{DEN}}$ 作为输出允许信号。

(4) DT/ $\overline{\text{R}}$ (Data Transmit Receive)。数据发送/接收信号，输出，三态。用来控制 8286/8287 的数据传送方向，当为高电平时，进行数据发送，低电平则进行数据接收。

(5) M/ $\overline{\text{IO}}$ (Memory Input and Output)。存储器/输入输出控制信号，输出，三态。用来区分 CPU 进行存储器访问还是输入/输出访问的控制信号。高电平时，CPU 和存储器之间进行数据传输；低电平时，表示 CPU 和输入/输出端口进行数据传输。8088 的对应引脚是 $\overline{\text{M}}$ / IO 。

(6) $\overline{\text{WR}}$ (Write)。写控制信号，输出，三态，低电平有效。表示 CPU 当前正在进行对存储器或 I/O 端口的写操作。它只在 T_2、T_3 和 T_W 状态有效。

(7) HOLD (Hold Request)。总线保持请求信号，输入，高电平有效，是系统中的其他总线主控部件向 CPU 发出的请求占用总线的控制信号。当 CPU 从 HOLD 处收到一个高电平请求信号时，如果 CPU 允许让出总线，并且就在当前总线周期完成时，它会在 T_4 状态从 HLDA 线上发出一个应答信号，同时使地址/数据总线和控制总线处于浮空。总线请求部件收到 HLDA 后，获得总线控制权，这时，HOLD 和 HLDA 都为高电平。当请求部件完成对总线的占用后，HOLD 变为低电平，CPU 收到无效信号后，将 HLDA 也变为低电平，即 CPU 又恢复了对地址/数据总线和控制总线的占有权。

(8) HLDA (Hold Acknowledge)。总线保持响应信号，输出，高电平有效，是与 HOLD 配合使用的联络信号。当 HLDA 为有效电平时，所有与三态门相接的 CPU 的引脚都应处于浮空，从而让出总线。

2. 8086 最小工作方式时的系统总线结构

所谓最小工作方式，就是系统中只有一个微处理器(如8086)。在这种系统中，所有的总线控制信号都直接由 8086 产生，系统中总线控制逻辑电路减到最少。最小工作方式系统适合于较小规模的应用，其典型系统结构图如图 1-5 所示。

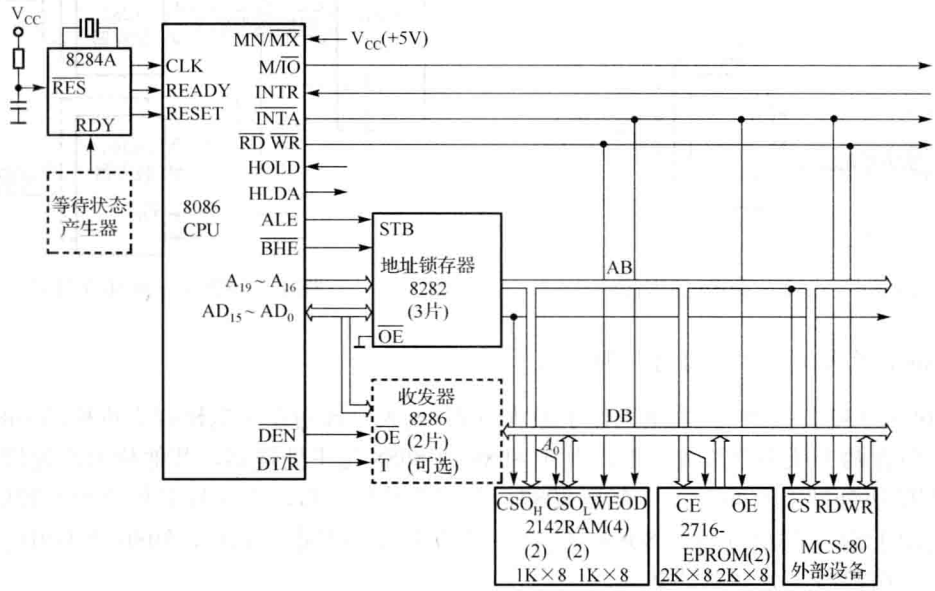


图 1-5　8086 最小方式典型系统结构

1) 时钟发生器

一片 8284A 时钟发生器产生系统所需要的时钟信号 CLK，同时对外部 READY 信号和 RESET 信号进行同步。由 8284A 内部产生自激振荡，在 CLK 端输出占空比为 33%的时钟信号。上电复位及按钮复位信号经 $\overline{RES}$ 端送入 8086，从存储器和外设来的等待请求信号送 RDY 端，经 8284 同步后再送 8086 的 READY 端。

2) 地址锁存

由于受外部引脚数量的限制，8086 CPU 采用了地址/数据线分时复用的总线结构，使 CPU 不能同时发送地址和数据，在 T_1 状态，输出地址信号，而在 T_2~T_4 状态，总线用于数据传送。为了有一个稳定的地址信号，保证数据的有向传送，所以在系统中地址信号消失之前，

必须用锁存器将地址锁存。地址锁存器 8282(3 片)与 8086 连接的原理图如图 1-6 所示。由于地址信号要一直有效，所以 8282 的输出端 $\overline{OE}$ 要接地。在 T_1 状态，CPU 输出地址锁存允许信号 ALE，将 ALE 接到 8282 的选通输入端 STB，当 ALE = 1 时，8282 输出跟随输入变化，用 ALE 的下降沿将总线上已经稳定的信号锁入 8282。

3) 数据线驱动

为了避免 T_1 期间出现的地址信息送入数据总线，CPU 用 $\overline{DEN}$ 信号表示复用的引脚已转换到数据状态的时刻，同时由于 CPU 的数据引脚驱动能力有限，为避免出现逻辑电平异常现象，需要给它加上一个总线驱动器，用 $\overline{DEN}$ 控制驱动器是否接通，以保证只在数据传输时驱动器接通。总线驱动器 8286 与 8086 的数据总线连接如图 1-7 所示。两片 8286 的 A_7～A_0 分别与 CPU 的 AD_7～AD_0 和 AD_{15}～AD_8 相连，$\overline{OE}$ 端接 8086 数据输出允许信号端 $\overline{DEN}$，发送数据控制端 T(Transmit)接到数据发送/输出端 $DT/\overline{R}$。8286 是双向总线驱动器，$\overline{OE}$ = 1 时，8286 两端均为高阻状态。T = 1 时，数据从 A 传向 B；当 T = 0 时，数据从 B 传向 A。

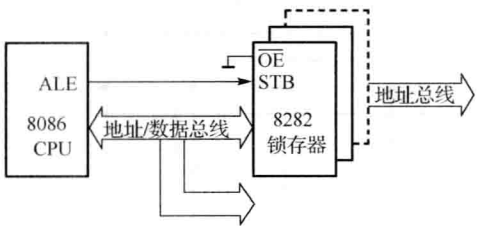


图 1-6　8282 与 8086 的连接

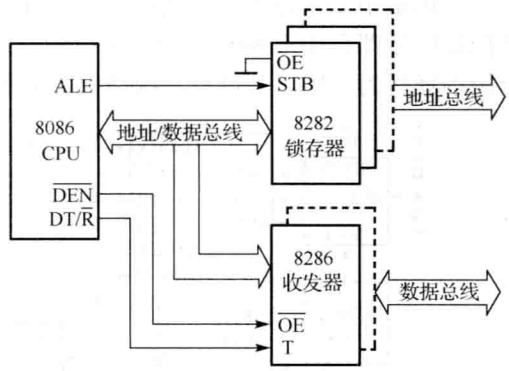


图 1-7　8286 与 8086 的连接

3. 8086 最大工作方式时的引脚功能

当 8086 的 33 脚接地时，系统工作于最大方式。最大方式用在中规模或大规模的 8086/8088 系统中，包含两个或多个微处理器，其中 8086 或 8088 是主处理器，其他称为协处理器。和 8086 匹配的协处理器有两个：一个是 8087，专用于数值运算，能实现多种类型的数值操作；另一个是用于输入/输出处理的 8089，它有专用的输入/输出指令系统，8086 就不用再处理这类工作，从而提高主处理器的效率。

(1) $\overline{S}_2$、$\overline{S}_1$、$\overline{S}_0$ (Bus Cycle Status)。总线周期状态信号，输出，三态。在最大方式下，这些信号组合起来指出当前总线周期中所进行的数据传输过程的类型。在最大方式系统中使用总线控制器 8288 后，就可以从 $\overline{S}_2$、$\overline{S}_1$、$\overline{S}_0$ 状态信息的编码中产生对存储器和 I/O 接口的控制信号。$\overline{S}_2$、$\overline{S}_1$、$\overline{S}_0$ 和具体的总线操作之间的对应关系如表 1-4 所示。

表 1-4　$\overline{S}_2$、$\overline{S}_1$、$\overline{S}_0$ 的编码与总线操作对应关系

$\overline{S}_0$	$\overline{S}_1$	$\overline{S}_2$	总线操作	$\overline{S}_0$	$\overline{S}_1$	$\overline{S}_2$	总线操作
0	0	0	中断响应	1	0	0	取指令码
0	0	1	读 I/O 端口	1	0	1	读存储器
0	1	0	写 I/O 端口	1	1	0	写存储器
0	1	1	暂停	1	1	1	无效