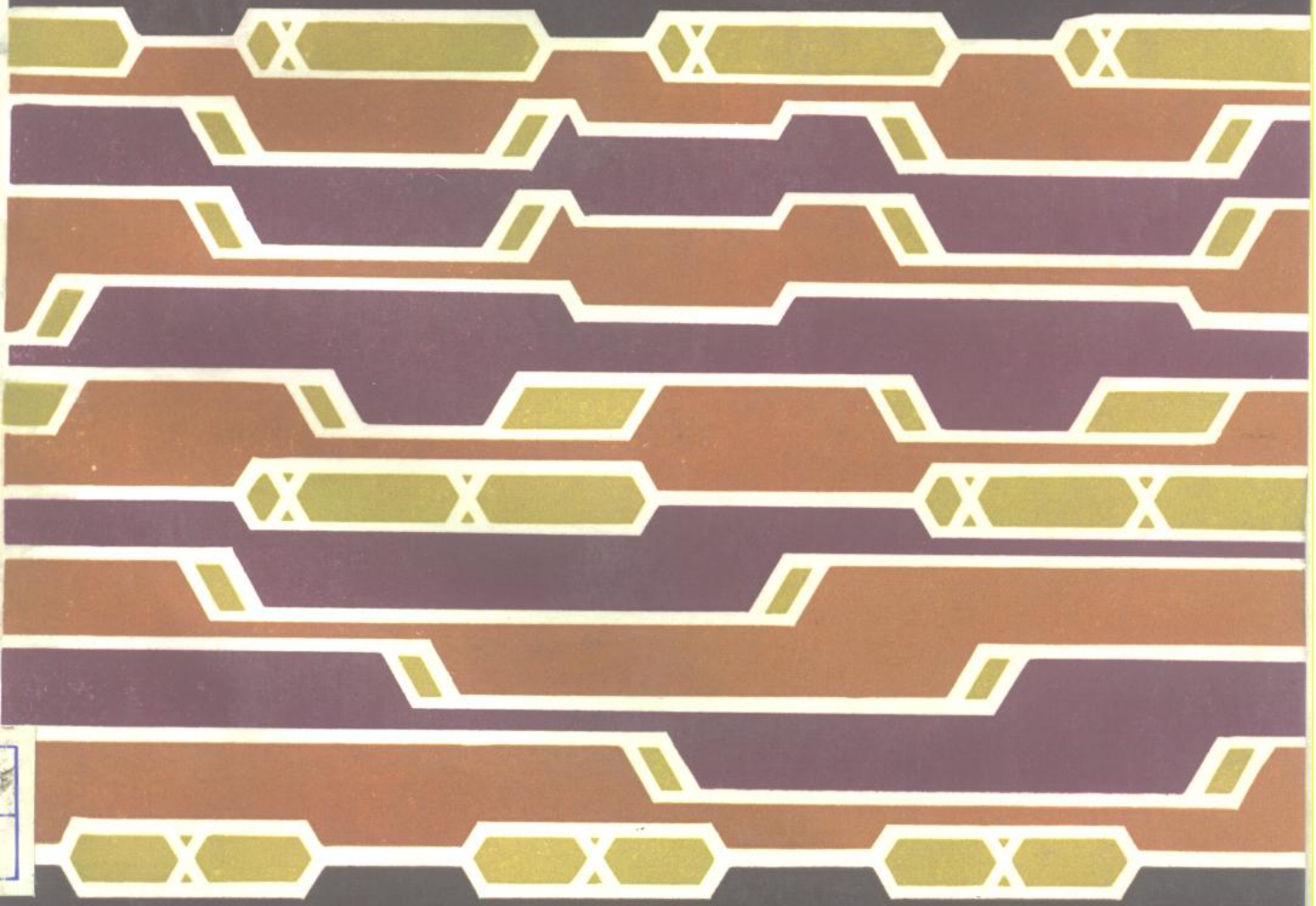
16位与32位微计算机手册

I6WEI YU 32WEI WEIJISUANJI SHOUCE

周继武 黄秦 等编译



国防工业出版社

16位与32位微计算机手册

周继武　黄　秦　等编译

国防工业出版社

内 容 简 介

这是迄今国内、外比较完整的16位与32位微计算机技术手册。本《手册》分上、下两篇，共十二章。上篇为十六位机部分，下篇为32位机部分。上篇的第一至第七章详细介绍了世界上较有影响，在我国较流行的各种16位微处理器及其专用支持芯片的结构、功能、指令系统和用法，每种芯片均有详尽的数据图表及技术指标。下篇的第八至第十二章，除综述了1985年之前世界上流行的十多种32位微型机之外，重点介绍了 NS16000，Z80000，iAPX432 和 MC68020。本书既可作为所有从事计算机，特别是微型机的科研、设计、生产、使用和维护的工程技术人员的工具书，也可作为大专院校师生的参考手册。

16位与32位微计算机手册

周继武　黄　秦　等编译

*

国防工业出版社出版

新华书店北京发行所发行　　各地新华书店经售

国防工业出版社印刷厂印装

*

787×1092　1/16　印张48 3/4　1120千字

1987年5月第一版　1987年5月第一次印刷　印数：0,001—6,400册

统一书号：15034·3151　　定价：10.15元

前　言

自 1971 年第一台微处理器 Intel 4004问世以来，随着集成电路技术的迅速发展，微处理器的性能和集成度也以成倍的速度迅速发展，其产品和应用的发展更是势不可挡。据统计，在西方世界，1980年生产了微处理器 1.2 亿个，估计 1985 年将超过 4.5 亿个，1989年将超过 13 亿个。其中，16位微处理器产量自1985年超过 8 位微处理器之后，将以极快的速度增长，1989年，16位微处理器将超过 11 亿个，8 位微处理器仅为 1 亿多个，32位微处理器 7000 万个左右。为了给读者提供有关 16 位与 32 位微处理器及其支持芯片的比较完整的资料，为了有助于我国用户选择和使用 16 位与 32 位微型计算机，我们编译了这本《手册》。本手册的编译原则如下。就16位机而言，尽量收录在国际上影响大，在我国流行广的机种。为了有利于我国使用小型计算机的经验继续在微型机的应用中发挥作用，我们还收录了我国广泛应用的仿 NoVa 系列和 PDP 11系列两类小型机的微型化处理机。鉴于 IBM PC机潮水般涌入我国市场，所以在介绍 8086 系列之后，附录了《IBM PC概要》。就 32 位机而言，国外有人称1984年是 32 位微型机年，但在我国却不多见。虽然有关 32 位微型机的资料较为缺乏，但从长远考虑，我们尽可能广采博收。除了就已公布的十多种 32 位机作了一般介绍和综述之外，还特别重点介绍了NS16000，Z 80000，iAPX422和MC68020等比较有影响的 32 位机。

本书原始资料主要是美国各有关厂家的产品手册，美国Osborne/Mc Graw Hill出版公司1981年第五次修订再版的《Osborne 16 Bit Microprocessor Handbook》，Micro Text/Mc Graw Hill 出版公司 1983 年出版的《Guide to Microcomputer》等书籍，以及美国和日本的微计算机杂志（如Byte，マイコン等）。对于我们所取的资料，编译中尽可能多方查核，据实修正。但是，鉴于有些器件找不到更为详尽或更可靠的资料，无法对有关内容考证核对，故只能仍按其原貌译出。限于编译者所识不广，书中谬误之处定然不少，尚祈读者指正。

参加本书编译、校核的有周继武，黄秦，张雄杰，史柯，杨新之，陈宗穆，朱圣瑜等同志。

说　明

1．信号规则

信号可以是高有效、低有效或高、低都有效。高有效信号就是在高电平状态时导致事件发生，而在低电平状态时则无意义。低有效信号就是在低电平时导致事件发生，而在高电平时则不起作用。高、低都有效的信号是指高电平状态和低电平状态分别导致不同类型的事件发生，这种信号没有无效状态。本书中在信号名称上加一横线表示低有效信号。例如，$\overline{WR}$就是低有效的“写选通”信号。信号名称上没有横线的信号是高有效或高、低都有效信号。

2．时序图规则

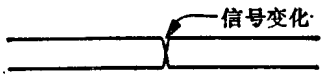
时序图在说明微处理器或支持器件时很重要。本书中时序图用得特别多。所有时序图都遵循下列规则：

1）低信号电平相当于零电压，高信号电平相当于有电压存在：

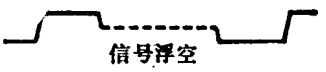
2）构成从低向高电平转换的单个信号表示为：

3）构成从高向低电平转换的单个信号表示为：

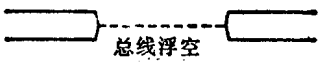
4）当存在两个或两个以上并行信号时，用下图表示有一个或一个以上的并行信号改变电平，但不指明高向低或低向高转换：



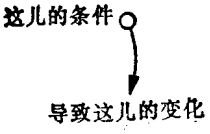
5）三态单信号浮空的表示：



6）含两个或两个以上信号的三态总线浮空的表示：

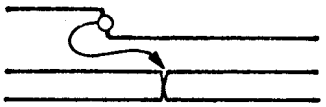


7）若一个信号条件触发另一个信号发生变化，则用箭头指示这种关系：



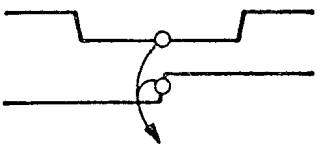
例如，若一个信号从低向高电平的转换触发另一信号从高向低电平的转换，则用下图表示：

又例如：若一个信号从高向低电平的转换触发总线状态变化，则用下图表示：

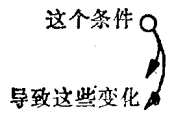


8）若触发一个逻辑事件必须有两个或两个以上条件存在，则用下图表示：

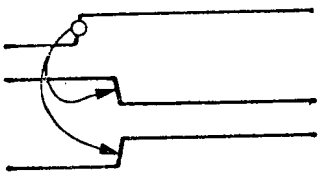
例如，若一个信号从低向高电平转换，另一信号处于低电平，这两个条件触发第三个事件，则用下图表示：



9）若一个触发条件导致两个或两个以上的事件发生，则用下图表示：



例如，若一个信号从低向高电平的转换触发另外两个信号电平变化，则用下图表示：



10）所有信号电平的变化都用方波表示，而将其上升和下降时间忽略不计。这些时间都附在每章后面的数据表里。

3．指令系统规则

每种微计算机指令系统都用两类表格说明：一类表格说明指令执行时的操作；另一类表格说明目标代码和指令时间。

由于指令系统之间的差异很大，我们没有采取用同一组代码和符号来描述所有指令系统中全部指令的每一种操作的办法。我们认为，任何类型的通用规则都可能导致混乱而不是导致明确。所以每个指令系统表前面都列出了表中具体使用的符号。

短的基准程序是说明指令系统用的。关于基准程序的解释一般按顺序进行。我们不打算评论各种不同器件的长处或短处，也不打算在本书中进行比较分析，因为一种微计算机比另一种微计算机强的标准在很大程度上只是取决于应用。

目　录

上篇　16位微计算机

下篇　32位微计算机

上　篇

16位微计算机

第一章 INTEL 8086系列

8086是Intel公司的第一种16位微处理器，它的功能要比该公司以前的各种微处理器强得多。

8086汇编语言指令系统仅在源程序这一级与8080A向上兼容。这就是说，每条8080A汇编语言指令可以转换成一条或几条8086汇编语言指令。无疑，谁也不会去把8086汇编语言指令逐条转换成一条或几条8080A汇编语言指令。如果真的这样做，他马上就会被存储单元分配和特殊的翻译规则弄得晕头转向。所以说，8086与8080A汇编语言指令系统是向上兼容的。

8086和8080A汇编语言指令系统在目标代码这一级是不兼容的。这就是说，ROM中的8080A程序在8086系统中是无效的。

8085和8080A汇编语言指令系统是一致的，区别仅在于8085A的RIM和SIM指令。8085的RIM和SIM指令使用8085的串行I/O逻辑，而8086无此逻辑。除了RIM和SIM指令之外，8085和8080A汇编语言指令系统是完全相同的。所以，8086汇编语言指令系统当然也与8085汇编语言指令系统（除RIM和SIM指令外）向上兼容。

8085和8080A汇编语言指令系统（除8085的RIM和SIM指令外）的目标代码也是兼容的。这就是说，8085和8080A的ROM中的程序可以互相通用。

8080A汇编语言指令系统是Z80汇编语言指令系统的一个子系统。这就是说，Z80可以执行8080A的目标程序，但反之则不可。当Z80的全部指令都在程序中，8080A就不能执行这种程序。8086与Z80的汇编语言指令系统不向上兼容。

回顾一下，8080A的前驱8008微处理器与8080A也只在源程序这一级上兼容。即是说，每条8008汇编语言指令在8080A上都有相应的汇编语言指令，但这两种微处理器的目标代码系统却不相同。

上述各种指令系统的兼容性如图1.1所示。

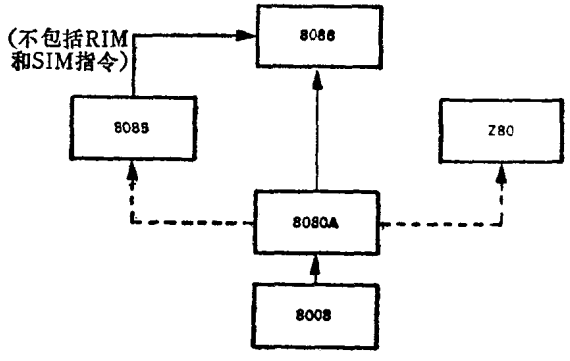


图1.1 几种指令系统的兼容性

虚线表示在目标程序级上，下层微处理器的指令系统是上层微处理器的子系统；实线表示下层微处理器的源程序可汇编产生上层微处理器的目标程序。

下面是8086硬件设计中最引人注目的改进：

（1）8086中央处理器逻辑分成两半：执行部件（Execution Unit，EU）和总线接口部件（Bus Interface Unit，BIU），这两部分异步运行。BIU用来管理与外部总线的所有接口，产生外存●及I/O地址，并且具有一个6字节的指令目标代码队列。在

● 此处“外存”系指CPU外部的存储器，并非指外部设备中的磁盘、磁带等，下文类此不再另注。 ——译者

EU需要访问存储器或I/O装置时，它就向BIU发出总线访问请求。如果BIU当时处于空闲状态，它就受理EU的总线访问请求。当BIU没有正待执行的来自EU的总线访问请求时，它就执行取指令机器周期，以填入6字节的指令目标代码队列。CPU从队列的前面取指令目标代码，这样就大大减少了取指令的时间。

（2）8086在设计中就考虑了要适用于从简单的单CPU系统到多CPU网络这样范围广阔的微计算机系统配置。为了提供这种灵活的多用性，8086的一些引脚能输出两组不同的信号，如图1.2所示。

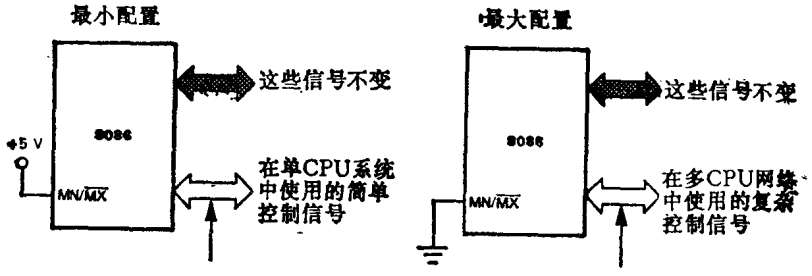


图1.2 8086所输出的两组不同的控制信号

根据MN/$\overline{MX}$的电平，同样的一些引脚输出这两组不同的信号。这种整批的重新分配信号的能力是微处理器工业的一大创新。

（3）8086具有处理多CPU配置中的总线访问优先级别的内部逻辑。(但这不算是创新，National Semiconductor公司的SC/MP多年前就是如此。)

（4）在多CPU配置中，每个8086CPU具有自己的局部存储器，同时又共享公用存储器。公用存储器可由所有的CPU共享，也可以只供某几个CPU共享。

（5）8086在设计中考虑了它在复杂程序的应用中有效的竞争能力，这些应用曾是小型计算机的应用范围。它直接访问的外存空间达1兆字节。所有的存储器寻址都是基址相对寻址，这种寻址法自然产生浮动目标程序。(浮动目标程序能从一个存储器地址空间移到另一地址空间，并且不用更改就可重新执行。）另外，由于8086采用堆栈相对寻址，故很容易编写重入程序。(重入程序能在执行中途被中断，之后又可重新执行。例如，调用自身的子程序就是一种重入程序；在执行中能被外部中断信号所中断，然后又能在中断服务程序中重新开始执行的程序，这也是一种重入程序。)

（6）8086采用前缀指令，它能修改对下一指令目标代码的解释。

8086与其前驱8080A一样，实际上是多微处理器结构中的一个部件。

除了8086微处理器本身，还必须有个8284时钟发生器/驱动器。你可利用替代逻辑来产生所需的时钟信号，不过，这样做实际上既不经济，也不实用。

某些8086微处理器结构中所需的第三种部件是8288总线控制器。

在8086与其系统总线（或各条总线）之间一般要用8288总线控制器，正如8080A与系统总线之间要用8228总线控制器一样。但就8086来说，在单总线结构中可省去8288总线控制器而不受影响。

8086拥有配套齐全的支援器件系列。本章中要介绍的支援器件是：

（1）8284时钟发生器/驱动器；

（2）8288总线控制器；

（3）8282/8283 8 位输入/输出锁存器；

（4）8286/8287 8 位并行双向总线驱动器。

此外，还介绍8088，它是8086的 8 位改型。

8086的主要生产厂家是：

INTEL CORPORATION
3065 Bowers Avenue
Santa Clara, California 95051

作为第二来源的厂家是：

MOSTEK CORPORATION
1215 West Crosby Road
Carrollton, TX 75006

NEC MICROCOMPUTERS INC
Five Militia Drive
Lexington, MA 02173

SIEMENS AG
Components Group
Balanstrasse 73, D8000
Munich-80, West Germany

8086是用N沟耗尽型负载、硅栅工艺制造的。它采用 40 引脚双列直插封装，单＋5 V电源。除了时钟输入，其他全部信号都是与TTL电平兼容的。其时钟输入必须是MOS电平信号，它由8284时钟发生器/驱动器产生（这将在本章最后面介绍）。

指令执行时间不定，这取决于指令排队效果如何。一般每执行一条指令需 2～30个时钟周期。乘法和除法指令所需的执行时间更长。时钟周期可短到125ns。

已公布了一种主频为 8 MHz 的8086改型，称为8086-2。主频为 4 MHz 的8086改型称为8086-4。标准的8086主频为 5 MHz。这三者除了允许的最高时钟速度外，并无其他区别。

1.1 8086CPU

8086微处理器芯片上具备的功能块如图 1.3 所示。

图中对中断优先级仲裁逻辑只给出了一半：外部逻辑（如8259A），必须提供识别中断的设备代码，但其后所有仲裁和向量逻辑都是由CPU内的逻辑处理。

值得指出的是8086的总线接口逻辑（图 1.3 中已绘），它要比其他微处理器所提供的总线接口逻辑的功能强得多。

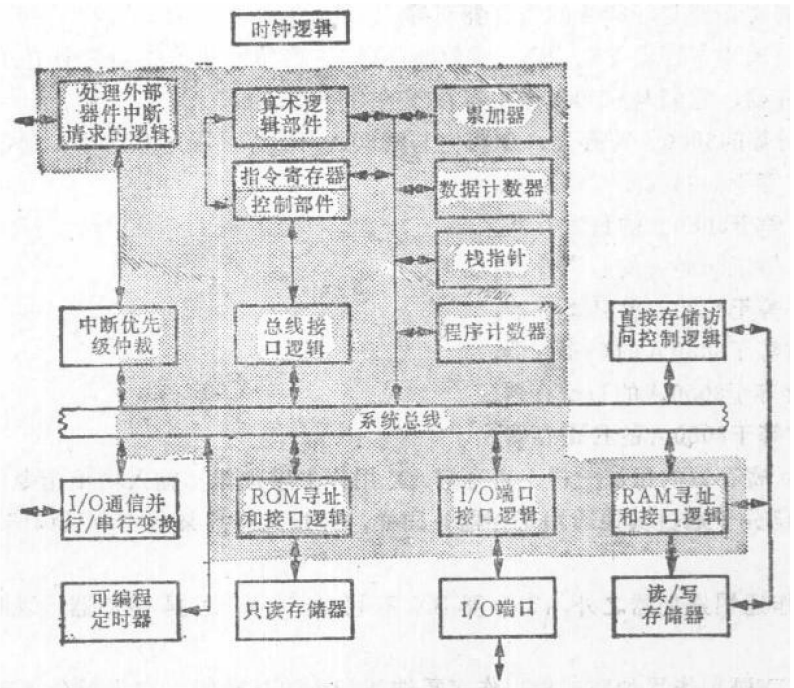


图1.3 Intel 8086CPU 的逻辑框图

1.1.1 8086可编程序寄存器与寻址方式

这里将8086可编程序寄存器与8086寻址方式一并讨论，这是因为8086的许多可编程序寄存器在这里只用来支援存储器寻址逻辑。8086可编程序寄存器如图 1.4 所示。

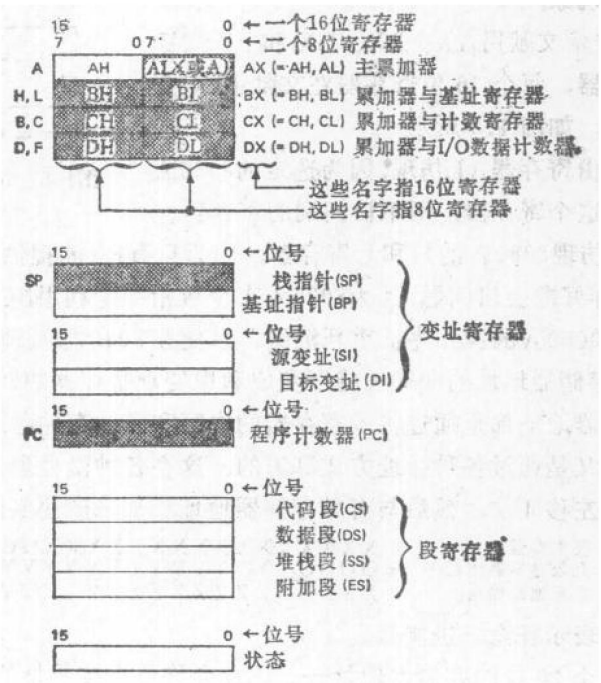


图1.4 8086可编程序寄存器

有阴影的寄存器是8080A寄存器在8086里的对应寄存器。8080A寄存器名称标于左侧。

有阴影的寄存器与8080A的寄存器对等。

首先看看通用寄存器AX、BX、CX及DX。这些单元被当成是四个16位寄存器或八个8位寄存器，它们与8080A通用寄存器的对应关系如下：

AH无对等的8080A寄存器，勿将它与8080A PSW（程序状态字）混淆。

AL 对等于8080A的A寄存器。

BH 对等于8080A的H寄存器。

BL 对等于8080A的L寄存器。

CH 对等于8080A的B寄存器。

CL 对等于8080A的C寄存器。

DH 对等于8080A的D寄存器。

DL 对等于8080A的E寄存器。

与8080A寄存器的用法一样，寄存器AX用作主累加器。输入输出指令使数据通过AX或AL（优先于通过其他通用寄存器）；同样，经过选择的某些指令只访问AX或AL的内容。

除了用作通用累加器之外，寄存器BX还可在计算数据存储器地址时用作基址寄存器。

寄存器CX既用作累加器，也用作多重迭代指令的计数器，这些指令在寄存器CX的内容增或减到0时就终止执行。

有的I/O指令在指定的I/O端口和寄存器DX所定址的存储单元之间传送数据。寄存器DX也用作累加器。

在了解通用寄存器AX、BX、CX和DX时，常常会遇到术语上的混乱现象。

Intel公司的技术文献用AX、BX、CX和DX来标记四个16位寄存器，每个16位寄存器在文献中又分成两个8位寄存器，如图1.5所示。

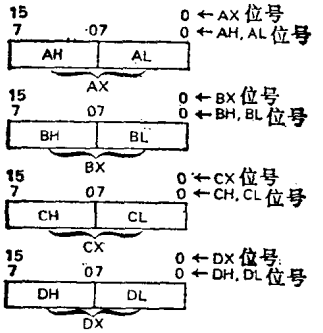


图1.5 每个16位寄存器分成两个8位寄存器

8080A累加器由寄存器AL仿现，因为选定的8080A和8086指令只访问这个寄存器，而不访问别的寄存器。

BH和BL必须仿现8080A的H和L寄存器，因为只有BX能提供8086数据存储器地址。表面看来，这样好象会出问题，因为8080A有少数指令能利用BC和DE寄存器提供16位存储器地址。在8080A的源程序被重新汇编，以便在8086微处理器上执行时，从BC和DE寄存器寻求存储器地址的8080A指令变成利用变址寄存器的8086指令。

所有8086存储器地址都是通过将段寄存器的内容和有效存储器地址相加而算出的。而有效存储器地址又是通过各种寻址方式算出的，这在各种微处理器上都是如此。选定的段寄存器的内容左移4位，然后与有效存储器地址相加来产生实际地址输出：

```
段寄存器内容：      XXXXXXXXXXXXXXXX0000
有效存储器地址：  +0000YYYYYYYYYYYYYYYY
实际地址输出：      XXXZZZZZZZZZZZZZZZZ
```

这里，X、Y和Z表示任意二进制数。

这样算出的是个20位的存储器地址——因此允许可直接寻址的外存容量是1048576字节。

8086的段寄存器与本书所介绍的其他微处理器的寄存器都不相同。它们用来作基址寄存器，可指向位于16字节的偶数倍地址边界上的任何存储单元。下面对随意选定的存储器地址作出图解（图1.6）。

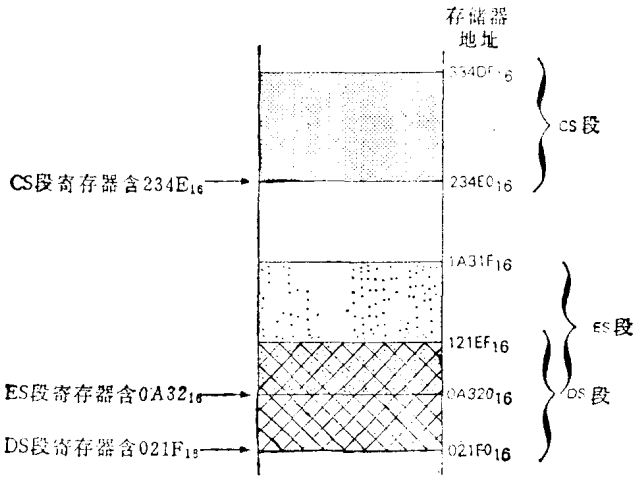


图1.6 8086 各存储段示例

如图 1.6 所示，每个段寄存器都标示一个 65536 字节存储段的起始地址。由于8086有四个段寄存器，故任何时候都有选用的四个 65536 字节的存储段。实际地址输出总是选择这四段中某段的一个存储单元。例如，若实际地址输出是 DS 段寄存器内容与有效存储器地址之和，则实际地址输出必须选择 DS 段内的一个存储单元，即是说，在图1.6中地址范围必然在$021FO_{16}$至$121EF_{16}$之间。同样，作为 CS 段寄存器内容与有效存储器地址之和的实际地址输出必须在 CS 段内选择一个存储单元，在图 1.6 中就是在地址范围 $234E0_{16}$ 至 $234DF_{16}$之间。

段寄存器的内容不受限制。所以，8086存储器既不分成 65536 字节的页面，也不必使四个段寄存器指定互不重叠的存储空间。每个段寄存器标示 65536 字节存储段的起始地址，这个存储段可位于可寻址存储器中的任何位置，与其他某一个或几个段可以重叠，也可不重叠。

尽管段寄存器可以形成重叠的或不重叠的存储段，它们却有特定的寻址功能。即是说，不同类型的存储器访问可在特定段内计算存储地址。

在取指令过程中，要使程序计数器的内容与代码段寄存器（CS）的内容相加，以便计算出应取指令的存储地址，如图 1.7 所示。

堆栈指令〔如 Push（压入），Pop（弹出），Call（调用），或 Return（返回）〕将栈指的内容与栈段寄存器（SS）的内容相加，以算出要访问的栈单元的地址，如图 1.8 所示。

处理数据串的指令利用 SI 和 DI 变址寄存器以及数据段寄存器（DS）和附加段寄存器（ES）来指明数据串的原地址和目的地址，如图 1.9 所示。

由图 1.9 可见，处理数据串的指令要求源串和目标串各居于一个 65536 字节的地址范围内，而不必在同一 65536 字节范围内。

访问数据存储器的指令将有效存储地址与数据段寄存器（DS）或栈段寄存器（SS）