UNIX 环境与 80x86 丛书

80x86 80x87 的结构与汇编语言程序设计

周明德 张淑玲 编著

UNIX 80X86

清华大学出版社

《UNIX 环境与 80x86》丛书

80x86
80x87的结构与汇编语言程序设计

周明德　张淑玲　编著

清 华 大 学 出 版 社

内 容 简 介

本书对 80x86 系列的芯片(8086、80286、80386 和 80486)及 80x87 系列的芯片(8087、80287 和 80387)的结构及指令系统,作了纵向的归纳与比较,使读者更容易理解和掌握。详细介绍了 80x86 的宏汇编程序的使用方法,并从实地址方式和保护虚地址方式两个方面分别深入地介绍了汇编语言的程序设计。在实地址方式方面着重介绍了 MS-DOS 系统调用功能的使用,也介绍了 80x87 的程序设计。重点放在保护方式下的汇编语言程序设计,有实例有分析有探讨。书中还提供了不少十分实用的实例。本书可作为高校的本科生、研究生的教材和广大软件工作者的重要参考书。

(京)新登字 158 号

80x86
80x87 的结构与汇编语言程序设计

周明德 张淑玲 编著

清华大学出版社出版
北京 清华园
通县向阳印刷厂印刷
新华书店总店科技发行所发行

☆

开本:787×1092 1/16 印张:18.5 字数:462 千字
1993 年 12 月第 1 版 1993 年 12 月第 1 次印刷
印数:00001—10000
ISBN7-302-01272-5/TP·480
定价:12.50 元

《UNIX 环境与 80x86》丛书

出 版 说 明

微电子技术、计算机技术是当今世界发展最为迅速的科学技术。1976 年 Intel 公司推出它的第一个 16 位微处理器 8086 时，芯片上集成的晶体管数大约 2 万个；1985 年 Intel 公司第一次宣布 32 位的微处理器 80386 时，芯片上集成的晶体管数已达 20 万个；目前正在大量推广的 80486，集成的晶体管数 达 100 万个；Intel 公司计划于 1993 年初推出的 P5(也即 80586)，估计集成的晶体管数可达 400 万个。芯片的工作速度则是成 10 倍地提高：8086 的主振频率为 5MHz，80486 的主振频率为 50MHz，不久，微处理器的主振频率将可达 100MHz 或 200MHz。相应地，微处理器的功能和性能也有了极大的提高。

8086 有 20 条地址线，寻址能力达到 1M 字节，比之 8 位微处理器的 64K 字节的寻址能力，提高了 16 倍，这在当时已是一种可观的发展。但是，应用证明，1MB 的内存是远远不够用的。80386 等 32 位微处理器的地址线扩展为 32 条，直接寻址能力达到了 4000MB (4Gb)，这显然是一个十分庞大的内存空间。随着半导体存储器容量的迅速提高和价格的急剧下降，386、486 微机的常用内存容量已达到 4MB、8MB 或 16MB。

总之，自 1981 年 IBM PC 问世以来的短短 10 余年，微型计算机有了极大的飞跃，系统的配置有了很大的扩展，性能有了很大的提高，年产量已突破了 2500 万台。目前，80386、80486 已成为微型计算机的主流 CPU。

微型计算机性能的极大的提高，使得以单用户、单任务为主要特征的、得到了广泛应用的操作系统 MS-DOS(PC-DOS)已经远远跟不上硬件系统的发展，也不能满足广大用户应用的需要。

多用户、多任务分时式操作系统——UNIX 因其具有适用面宽、可移植性好等开放系统的基本特征，在 80 年代得到了广泛的应用，已成为高档微型机、工程工作站和超级小型计算机的主流操作系统，在 90 年代必将得到极大的发展。

为发展我国的软件产业，1989 年，机电部决定集中资金，集中技术力量，以操作系统为突破口，大力进行国产系统软件的开发。此项目并被国家列为“八五”科技攻关的重点软件开发项目。

国产操作系统要求遵循国标标准“可移植操作系统界面 (POSIX)”，与国际上的主流 UNIX 系统相兼容，它的第一目标机是以 386、486 为 CPU 的高档微机。

为适应计算机的发展，为满足广大读者的要求，我们决定组织编写一套《UNIX 环境与 80x86》丛书。这套丛书详细地、深入地分析在 UNIX 环境下工作的 80x86 的原理、特点和使用。Intel 公司在设计 80386 时考虑了多用户多任务操作系统的需要，在芯片上提供了 4 种特权级，提供了任务和任务切换的机制，提供了片内的存储管理单元等硬件支持。UNIX 操作系统则充分利用了这些机制，充分挖掘了 80386 的潜力。只有透彻地了解 80386 的全部机制和功能，才能设计出在 80386 上运行的多用户、多任务操作系统。

这套丛书还全面地、分层次地介绍在 80x86 上运行的 UNIX 操作系统。

这套丛书的作者都是国产操作系统项目的主要技术骨干和技术带头人，是国产操作系统的主要设计者。他们对各种 UNIX 系统和在 UNIX 环境下的 80386 都作了深入的分析。

这套丛书具有理论性、先进性、系统性和实用性，适合各种不同层次的计算机工作者使用。

前　言

自 1981 年 IBM 公司推出 PC(Personal Computer)机以来，微型计算机有了迅猛的发展。全世界微型计算机的年产量已超过了 2500 万台，累计装机台数已达 1.5 亿台。Intel 公司的 86 家族的芯片 8086(8088)、80286、80386、80486 成为微型计算机中的主导 CPU，功能十分强大，并在继续发展(Intel 的 80586-P5 不久将推出)。

80x86 是一个向上兼容的系列芯片，它们在硬件结构上、指令系统上都是兼容的，在机器码级和汇编语言级上也是向上兼容的。从而在 MS-DOS(PC-DOS)支持下的大量应用软件在升级产品上仍可直接使用，这是十分重要的。预计在未来的 5 年内，Intel 80x86 系列的芯片仍是微型计算机的主流 CPU。

80x86 是向上兼容的，但自 80286 开始，特别自 80386 以后，芯片的功能有了质的飞跃，例如：能寻址 4G 字节的物理内存，具有 4 个特权级，任务切换机制，片内的内存管理单元等等。这些功能只有在保护方式下才能实现。但目前大部分各种档次的 PC 机的操作系统，采用的是 MS-DOS(PC-DOS)，而 DOS 只工作在 80386 的实地址方式，远不能充分发挥 80386 的强大功能。

由于 80386、80486 的出现及硬件成本的急剧下降，广大用户迫切需要使用在 80x86 的保护方式下的多用户、多任务操作系统，例如 Microsoft 公司的 Windows 和即将推出的 Windows NT(New Technology)以及 UNIX。要深入地学习和掌握这些操作系统，就需要全面地、深入地掌握 80386、80486 的功能，特别是保护方式下的功能。

汇编语言具有内存开销小，运行速度快的特点，虽然学习和使用比较困难，但在实时控制和实时处理领域仍是不可替代的。对于计算机软件工作者来说，汇编语言是十分重要的基本工具。例如，MS-DOS 向用户提供了丰富的命令，更重要的是提供了功能十分强大的系统调用函数，而这些只有在汇编语言级才能使用。若要对程序进行调试和排错，也只有熟悉了汇编语言才能深入地掌握系统所提供的调试工具，才能设置断点和进行单步跟踪。至于软件工程中的逆向工程更是以汇编语言作为基础的。就是在 MS-DOS 下，利用汇编语言的程序也能进入保护方式，挖掘保护方式的潜力来完成一些 MS-DOS 不支持的任务。总之，汇编语言的程序设计是十分重要的。

8086 的程序设计，或 80286、80386 在实地址方式下的程序设计(实质上仍相当于 8086 的程序设计)已有相当的著作论述，但对于 80x86 的保护方式，特别是保护方式下的程序设计仍是空白。

本书对80x86、80x87的结构和指令系统，从纵向上做了归纳和总结，对80x86、80x87的汇编语言程序设计作了深入的分析和开拓，重点是在保护方式下的程序设计，在大量实践的基础上作了较深入的探讨，所提供的实例也是很实用的。

本书是由周明德和张淑玲合作编写的，第2、4、7章是由张淑玲编写的，全书由周明德主编和审定。

本书是作者主编的80x86系列图书中的一篇，希望读者与其他姐妹篇结合起来使用。

限于水平和条件，书中会存在不少缺点和问题，恳请广大读者指正。

周明德

1992、8、29.

目　录

1　80x86 的结构

1.1　80x86 的功能结构

8086、80286、80386 和 80486 的功能结构分别如图 1-1、1-2、1-3 和 1-4 所示。

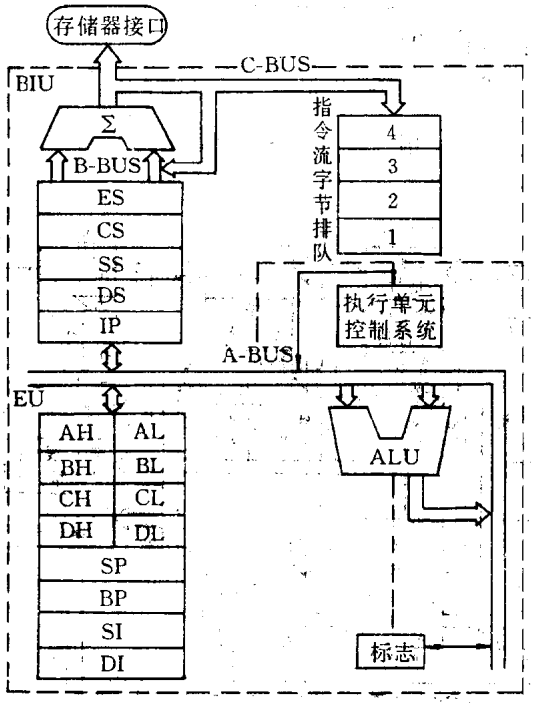

图 1-1　8086 的功能结构

8086 已经在 8 位微处理器的基础上前进了一步，把执行单元 EU 与总线接口单元 BIU 分开，从而在执行指令的同时(若不取操作数的话)可以取下一条指令，具有了指令流水线的功能，如图 1-5 所示。

80286 比 8086 前进了一大步，除了保留执行单元 EU 和总线接口单元 BIU 外，由于要支持保护虚地址方式下的 16M 内存寻址功能，支持描述符表示的寻址机制，就扩展了一个地址单元 AU；同时在 8086 的指令流水线的基础上，增加了指令预取队列(3 条指令)，扩展为指令单元 IU。

80386 除了把指令预取队列扩展为 16 个字节，把地址单元的地址扩展为 32 位，可直接寻址 4G 字节外，最重要的扩展是增加了片内的存储管理单元 MMU，它能实现分页机制，把逻辑地址和物理地址分为大小相等(每页 4K 字节)的若干页，通过两级页表(页目录表和

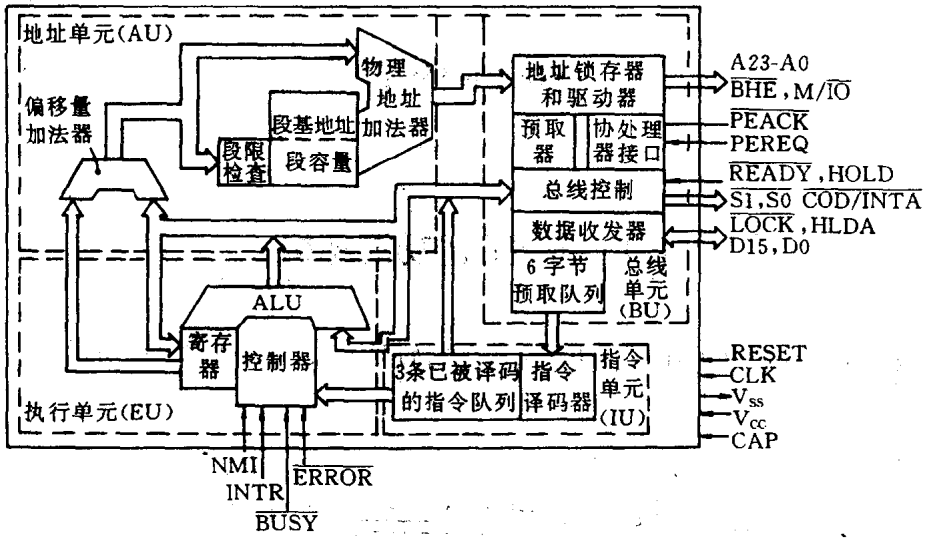

图 1-2 80286 的功能结构

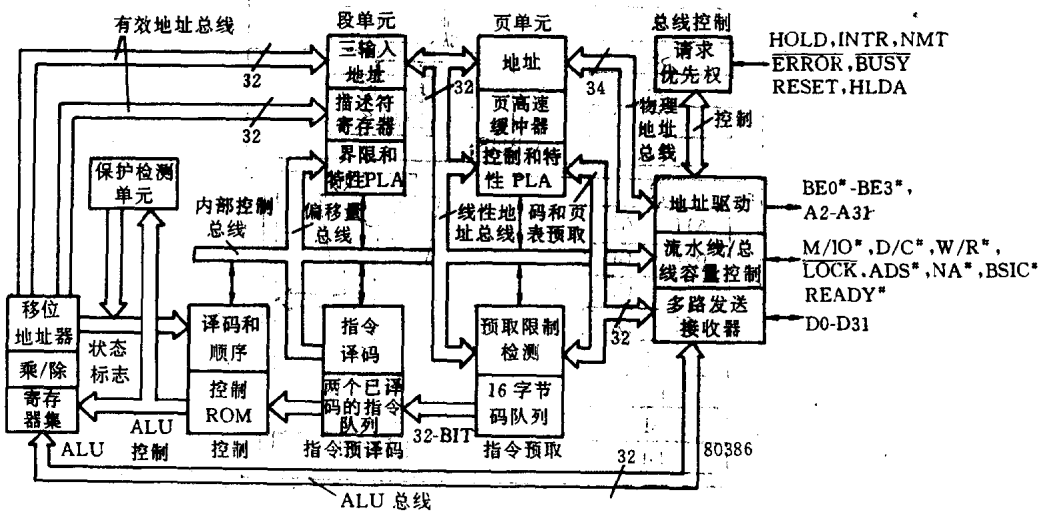

图 1-3 80386 的功能结构

页表)实现逻辑地址与物理地址的映射,能实现缺页中断,从而实现了虚拟存储器管理。

80486 的结构在多个方面有很大的发展,但最突出的是,它是 80386 和 80387(浮点运算协处理器)和 8K 字节的 Cache(指令与数据 cache)的结合。从而在功能上有了极大的提高。80486 的主 CPU 功能与 80386 一致,有关协处理器的内容我们在第二章中介绍。

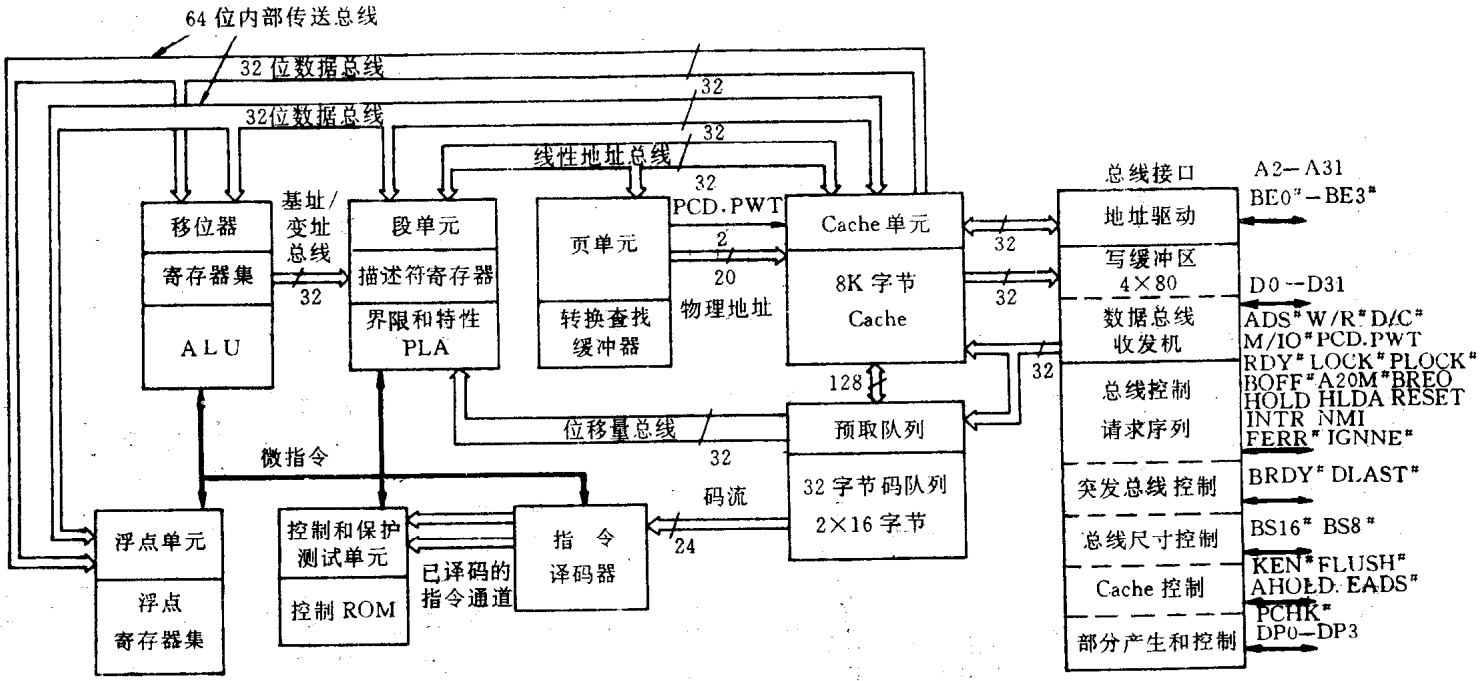

图 1-4　80486 的功能结构

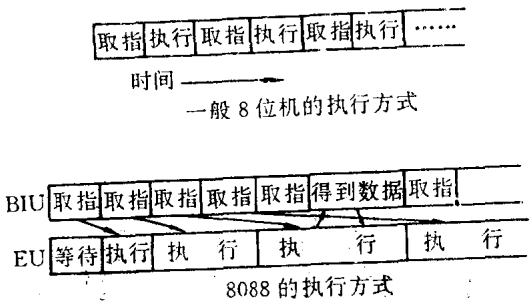

图 1-5　8086 中的指令流水线

1.2　80x86 的寄存器结构

8086、80286、80386 和 80486 的寄存器结构分别如图 1-6、1-7、1-8 所示。

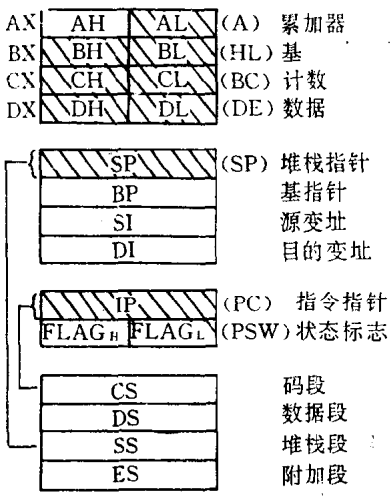

图 1-6　8086 的寄存器结构

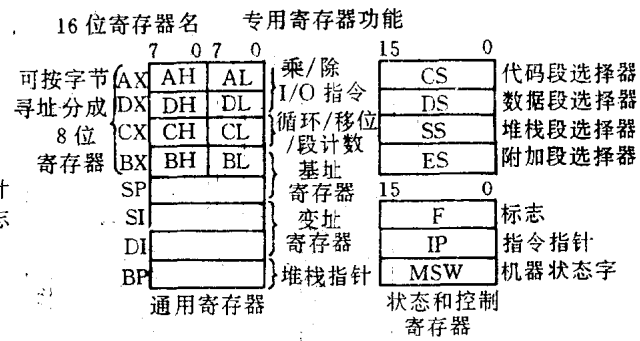

图 1-7　80286 的寄存器结构

8086 和 80286 的通用寄存器 AX、DX、CX、BX、SI、DI、BP 和 SP 是 16 位的，其中 AX、DX、CX 和 BX 都可以分为两个 8 位寄存器 AH、AL，DH、DL，CH、CL，BH、BL 使用。

8086 和 80286 中，指令指针 IP 和标志寄存器 FLAG 也都是 16 位的。

8086 和 80286 中，都有 4 个 16 位的段寄存器 CS、DS、SS 和 ES。

80286 比 8086 多了一个 16 位的机器状态字 MSW。

在 80386 和 80486 中，通用寄存器都扩展为 32 位，成为 EAX、EDX、ECX、EBX、ESI、EDI、EBP 和 ESP。它们的低 16 位分别为 8086 和 80286 的通用寄存器 AX、DX、CX、BX、SI、DI、BP 和 SP，可以单独使用。而且 AX、DX、CX 和 BX 都可以分为两个 8 位的寄存器使用。

在 80386 和 80486 中，指令指针和标志寄存器也都扩展为 32 位的 EIP 和 EFLAG，但它们的低 16 位仍可单独使用。

80386 和 80486 的段寄存器仍是 16 位的，但增加了两个数据段寄存器 FS 和 GS。

在 80386 和 80486 中，已经把 80286 中的机器状态字 MSW，扩展为三个 32 位的控制寄存器 CR0、CR2 和 CR3。此外，在 80386 和 80486 中，还有 6 个排错寄存器 DR0、DR1、DR2、

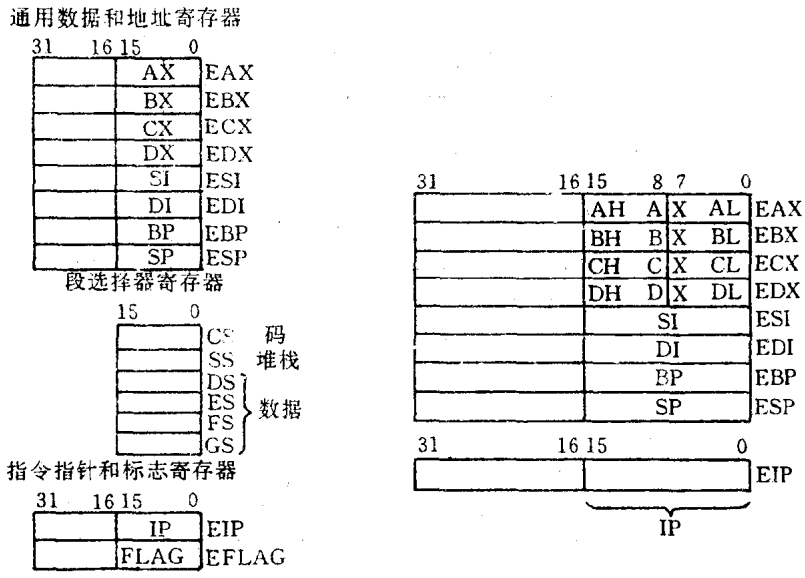

图 1-8　80386 的寄存器结构

DR3、DR6 和 DR7；80386 中有两个测试寄存器 TR6、TR7；在 80486 中有 5 个测试寄存器 TR3、TR4、TR5、TR6 和 TR7。

一、标志寄存器

8086、80286、80386 和 80486 的标志寄存器如图 1-9、1-10、1-11 和 1-12 所示。

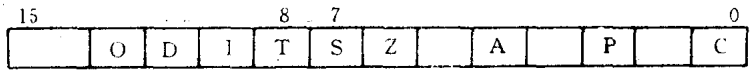

图 1-9　8086 的标志寄存器

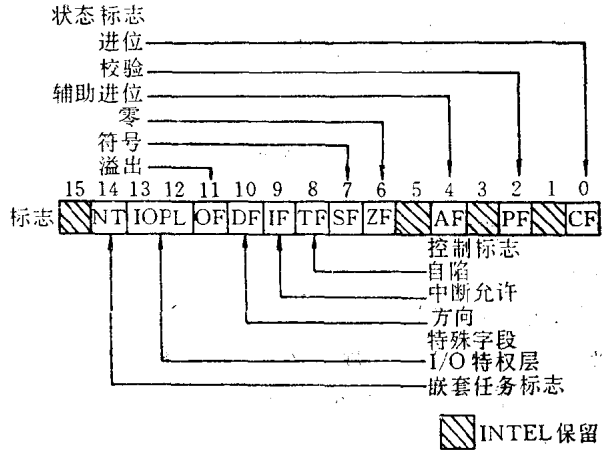

图 1-10　80286 的标志寄存器

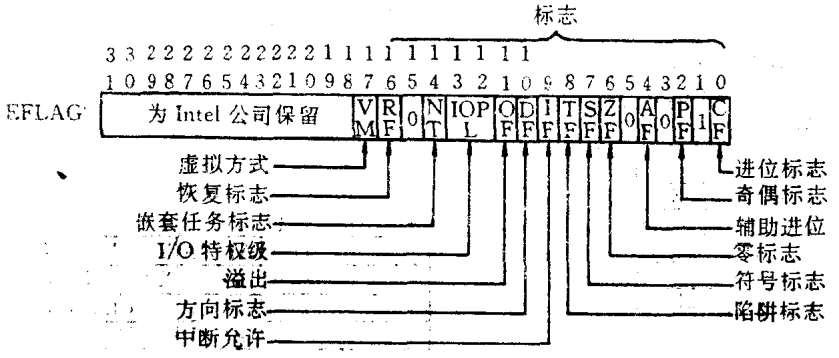

图 1-11　80386 的标志寄存器

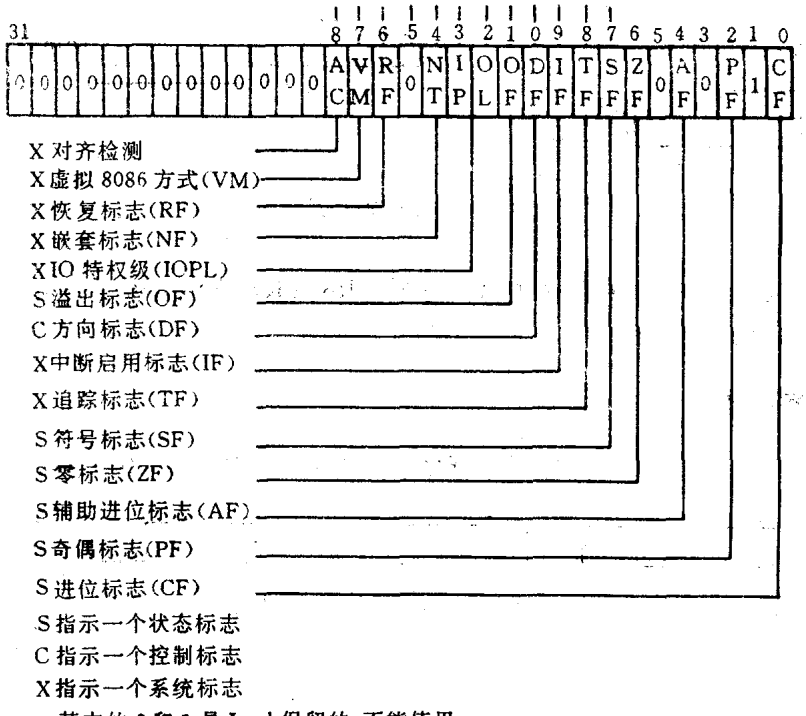

图 1-12　80486 的标志寄存器

在 8086 中共有 9 个标志，其中

1. 进位标志 C(Carry Flag)

当结果的最高位(字节操作时的 D_7 或字操作时的 D_{15})产生一个进位或借位，则 C=1；否则为 0。这个标志主要用于多字节的加、减法运算。移位和循环指令也能够把操作数(在寄存器或某一存储单元中)的最高位(左移时)或最低位(右移时)放入标志 C 中。

2. 奇偶标志 P(Parity Flag)

若操作结果中"1"的个数为偶数，则 P=1；否则 P=0。这个标志可用于检查在数据传送过程中是否出现错误。特别是在串行通信中，为了提高传送的可靠性，常采用奇偶校验。利用 P 标志可进行奇偶校验，或产生奇偶校验位。

3. 辅助进位标志 A(Auxiliary Carry Flag)

在字节操作时，由低半字节(一个字节的低 4 位)向高半字节进位或借位，或在字操作时，低位字节向高位字节进位或借位，则 A=1；否则为 0。这个标志用于十进制算术运算指令中。

4. 零标志 Z(Zero Flag)

若运算的结果为 0，则 Z=1；否则 Z=0。

5. 符号标志 S(Sign Flag)

S 的值与运算结果的最高位相同。即结果的最高位(字节操作时为 D_7，字操作时为 D_{15})为 1，则 S=1；否则 S=0。

由于在 8086 中，符号数是用补码表示的，所以 S 标志反映了结果的符号：0=正，1=负。

6. 溢出标志 O(Overflow Flag)

在算术运算中，带符号数的运算结果，超出了 8 位或 16 位带符号数能表达的范围，即在字节运算时大于+127 或小于−128；在字运算时大于+32767 或小于−32768，则此标志置位；否则复位。一个任选的溢出中断指令，在溢出情况下能产生中断。

以上 6 个标志为状态标志，反映了操作结果的状态。

以下三个标志为控制标志。

7. 方向标志 D(Direction Flag)

若用指令置 D=1，则串操作指令就成为自动减量指令，也就是从高地址到低地址或是从“右到左”来处理串；若使 D=0，则串指令就成为自动增量指令。

8. 中断允许标志 I(Interrupt-enable Flag)

若用指令使 I=1，则允许 CPU 响应外部的可屏蔽中断请求；若使 I=0，则屏蔽上述的中断请求。但此标志的状态，对于外部的非屏蔽中断请求，或内部产生的中断不起作用。

9. 追踪标志 T(Trap Flag)

若置 T=1，则使 CPU 进入单步方式，以便于调试。在这种方式中，CPU 在每条指令执行以后，产生一个内部的中断。允许程序在每条指令执行完以后进行检查。若置 T=0，则 CPU 执行指令后不产生内部中断。

80286 中标志寄存器仍是 16 位，它保留了 8086 的所有 9 个标志位，还增加了两种(3 位)系统标志。

I/O 特权标志 IOPL(两位)

它规定了能使用 I/O 敏感指令的特权级，在 80286 以上的处理器中，有一部分指令如 CLI(关中断指令)、STI(开中断指令)、IN(输入)、INS(输入串)、OUT(输出)和 OUTS(输出串)为 I/O 敏感指令。IOPL 的值(可为 0−3)规定了能执行这些指令的特权级(在 80286 以上的处理器中有 0−3 共 4 个特权等级，0 级特权最高，3 级特权最低)。在 IOPL 的值确定以后，特权高于 IOPL 的程序能执行 I/O 敏感指令；而特权低于 IOPL 的程序，若企图执行 I/O 敏感指令，则会引起异常。

嵌套标志 NT

在 80286 以上的处理器上，允许执行多任务，故任务可以切换也可以嵌套。NT=1，表示当前执行的任务嵌套于另一个任务中，执行完该任务后，可以用 IRET 指令返回到原来的任务；在 NT=0 时，则不能。详见后面任务切换部分的说明。

在 80386 中，除了保留 80286 的所有标志位外又增加了两个标志位，而且标志寄存器扩展为 32 位。扩展的两个系统标志为：

虚拟 8086 方式标志 VM

在 80386 的保护虚地址方式下，为了能在多任务系统中执行 8086 的任务，设置了虚拟 8086 方式。在 80386 处于保护虚地址方式时，若使 VM 位置位，则 80386 就进入了虚拟 8086 方式。VM 位只能在保护方式下由 IRET 指令（若当前的特权级＝0）或在任何特权级下由任务切换设置。VM 位不受 POPF 指令的影响，PUSHF 指令总是使此位清零。

恢复标志 RF

此标志是与调试寄存器的断点或单步操作一起使用的。在断点处理之前，在两条指令之间对该位进行检查。若 RF 位置位，则在下一条指令执行期间不处理任何调试故障。

在 80486 中，包括了 80386 的所有标志位，还增加了一个地址对齐检测标志。

地址对齐检测标志 AC。

当置 AC 位为 1 时，若发现地址不对齐就会产生异常。地址不对齐是指：若访问一个字时，地址为奇地址；若访问双字时，地址不处在双字边界上；若访问 8 字节操作数时，地址不对齐在 64 位的边界上。但不对齐故障只是在特权级 3 的程序运行时才会产生。在特权级 0、1、2 运行的程序，忽略 AC 标志的设置。

在 80486 中，各种数据类型地址对齐的要求如表 1-1 所示。

表 1-1　数据类型对齐要求

存储器访问	对齐（字节边界）
字	2
双字	4
单精度实数	4
双精度实数	8
扩展精度实数	8
选择子	2
48-位段指针	4
32-位段指针	2
32-位浮点指针	4
48-位“伪描述符”	4
FSTENV/FLDENV 保存区	4/2（操作数大小）
FSAVE/FRSTOR 保存区	4/2（操作数大小）
位串	4

二、控制寄存器

80286 比 8086 多了一个控制寄存器——16 位的机器状态字寄存器，如图 1-13 所示。

其中各位功能如下：

PE（第 0 位）：保护方式允许位。当 PE＝0 时，CPU 处在实地址方式；当 PE＝1 时，CPU 处在保护虚地址方式。在 CPU 复位时，PE＝0。

MP（第 1 位）：监督协处理器位，它用于与 TS 位一起决定 WAIT 指令是否引起异常。若