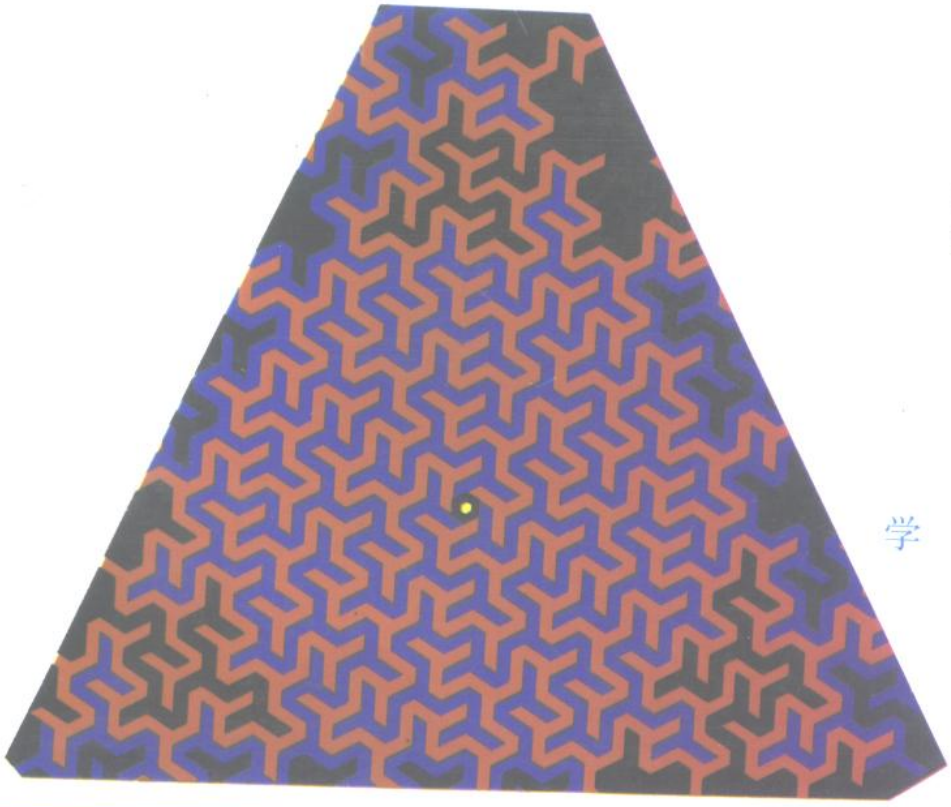
# 自学ASM386汇编语言
# 程 序 设 计



任千生 等编
熊可宜 审校

学 苑 出 版 社

北京希望电脑公司计算机技术丛书

# 自学 ASM386 汇编语言程序设计

任千生　等编
熊可宜　审校

学苑出版社
1993

(京)新登字 151 号

## 内 容 摘 要

汇编语言是一种低级语言，紧密地关联着 80386CPU．它的突出优点是程序代码短，运行速度快，与操作系统接口方便．许多属于低层的软件均由汇编语言写成．目前许多部门已广泛地使用着 32 位微型机，大批读者有了一定的水平，迫切地希望有一本关于 386 汇编语言方面能协助自学的参考书，以帮助消化、分析和应用 ASM386 汇编语言所写的软件．为此，本书本着力求深入浅出，资料较全，比较详细，例证较多，便于自学的出发点，编写了此书，是计算机应用人员的必备参考资料．





**自学 ASM386 汇编语言程序设计**

**编　　者**：任干生 等
**审　　校**：熊可宜
**责任编辑**：徐建军
**出版发行**：学苑出版社　　**邮政编码**：100032
**社　　址**：北京市西城区成方街 33 号
**印　　刷**：兰空印刷厂印刷
**开　　本**：787×1092　1/16
**印　　张**：42.9　　**字　数**：992 千字
**印　　数**：1—5000 册
**版　　次**：1993 年 10 月北京第 1 版第 1 次
ISBN 7—5077—0779—2/TP・11
**定　　价**：49.00 元

学苑版图书印、装错误可随时退换

# 前　言

自从 1971 年 Intel 公司的 4004 微处理器上市后，随之就出现了八位微处理器 8008。到 1974 年，第二代微处理器 8080 问世。过了四年之后，1978 年终于开发出功能完善的 16 位微处理器 8086 芯片，及其变种的 8088 准 16 位微处理器。又过了七年，Intel 公司推出 80286 微处理器，并随后于 1985 年宣布了 32 位功能强劲的 80386 微处理器芯片。这样高速度地发展向上兼容的微处理器芯片，已使微型计算机具有了小型机或中型机的若干功能。目前许多部门已广泛地使用着 32 位微型机。涌入计算机应用队伍的大批同志在具备了一定的应用水平后，迫切地希望有一本关于 386 汇编语言方面能协助自学的参考书，以帮助消化、分析和应用 ASM386 汇编语言所写的软件。为此，我们本着力求深入浅出，资料较全，比较详细，例证较多，便于自学的出发点，编写了这本资料，使具备高中文化程度的计算机应用人员自学之用。

以上由于汇编语言是一种低级语言，紧密地关联着 80386 CPU。它的突出优点是程序代码短，运行速度快，与操作系统接口方便。许多属于低层的软件均由汇编语言写成。对于一个较好的软件人员，必须撑握用汇编语言编程的技巧。

为了能循序渐进地自学 386 汇编语言，需要对 80386 微处理器的结构、特性及其内部若干约定有较细致地了解，所以本书第一章简介了 80386 微处理器的结构，以及它的实地址方式，保护地址方式和虚拟 8086 方式。为了便于查找，书中对 80386 CPU 的指令，协处理器的指令和宏汇编的伪指令，以及操作系统的功能调用等，均按 ASCII 码字符字母为序进行排列。

参加编写的还有：任一凡，刘策，任一民等工程师，任舒炜，任楠，刘若菁，刘星，刘明涛，吴进，刘童，吴园，潘兰等。

希望本书对读者能有所帮助，其中错误和不妥之处，恳请读者指正。

编　者

1993 年 · 9 月

# 目　录

# 第一章　Intel 80386 微处理器结构简介

80386 是一种与 8088、8086、80186、80286 相兼容的 32 位微处理器芯片。它新增了内部存储器管理部件和保护机构。其数据总线、地址总线、寄存器结构和操作均为 32 位，所以，它的寻址能力可达 4 千兆字节，虚拟存储器空间可达 64 兆兆字节。它具有下列性能:

(1) 芯片具有 132 条引脚的网格式阵列封装。

(2) 具有高速的 CHMOSⅢ技术。

(3) 有八个 32 位的通用寄存器，适用于 8 位、16 位、32 位的数据类型。

(4) 寻址空间很大。有4千兆字节的物理空间或64兆兆字节的虚拟存储空间。存储器有分段、分页的结构。每个段可达 4 千兆字节的空间。

(5) 有存储器部分管理部件。能支持虚拟存储器，可选择内部分页机构，具有四级保护，并且与 80286 等兼容。

(6) 指令目标码向下兼容。在受保护的系统或分页系统中运行 8086 的虚拟方式。

(7) 有指令的流水线结构，有内部地址转换的高速缓冲存储器，有多种挡次的时钟频率: 12.5MHZ; 16MHZ; 25MHZ，33MHZ，50MHZ 等。其总线的带宽很大。

(8) 配用 80287／80387 协处理器后可支持高速数据处理。

(9) 有完整的系统开发工具。

## 1.1　80386 的宏结构及寄存器

### 1.1.1　80386 结构简介

80386 由三部分组成: 中央处理器 CPU; 存储器管理部件 MMU; 总线接口部件 BIU。

一、中央处理器CPU(Central Processing Unit): 由指令部件IU(Instruction Unit)和执行部件 EU(Execution Unit)组成。

1. IU的功能是: 预取指令，对指令操作数进行译码，并将其放在指令队列中供EU使用。这样就使 EU 节省了取指令的译码所需要的时间。

2. EU 的功能是: 负责执行指令。EU 包括有:

(1) 八个 32 位通用寄存器。用于数据操作和地址计算。

(2) 一个64位的桶形移位器(Barrel Shifter)，用于加速位移，循环和乘、除法操作。

二、存储器管理部件MMU(Menory Menagement Unit): 由分段部件(Segment Unit)和分页部件(Page Unit)组成。

1. 分段部件: 由一个附加的寻址器件来管理逻辑地址空间。可将任务隔离开。可实现指令区与数据区的再定位。

(1) 每个段的大小可达 4 千兆字节。

(2) 给定范围的线性地址空间，或者是给定一个段后，它们均有相应的属性。属性是指该段或空间的位置、大小、保护方式和类型。类型用来表征该段是否是堆栈段、指令代码段、数据段等。

(3) 80386的每项任务最多可占用16381个段。所以可为每项任务提供64兆兆字节的虚拟存储器。

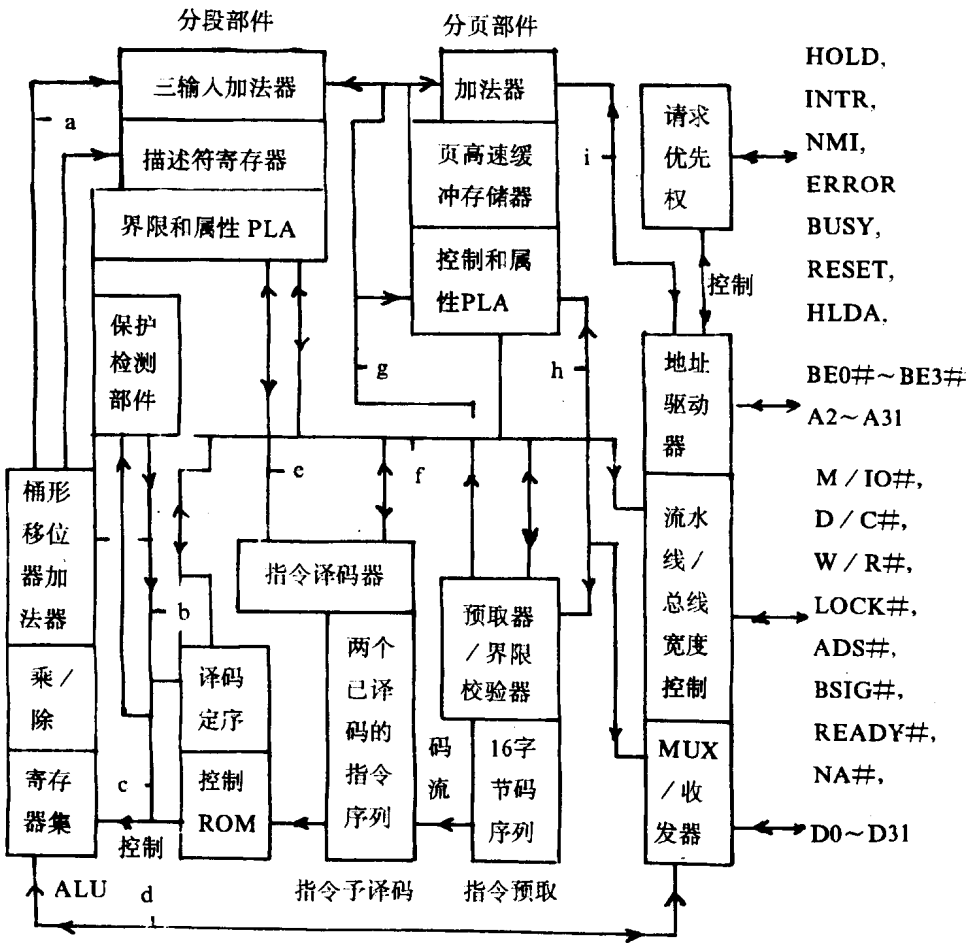
(4) 分段部件具有四级保护。

2. 分页部件：用来管理物理地址空间。

(1) 每页的大小为 4KB。

(2) 一段内可以是一页，也可以是若干页。

(3) 80386 对全部的页和故障均可再启动，以实现虚拟存储器。




注释：a——32位的有效地址总线，b——状态标志，i——34位物理地址总线，
d——专用的ALU总线，c——位移量总线，h——32位码获取与页表获取，
g——32位线性地址总线，f——内部控制总线，c——ALU控制，

三、总线接口部件BIU(Base Interface Unit): 负责与存储器之间传递信息。即从内存的指定区域取指令，送到指令队列中排队。操作数也是由 BIU 从内存指定区域取数据送到 EU 部件去执行。

四、操作方式: 80386 有两种操作方式。

1. 实地址方式(Real Address Mode)。是一个可扩展到 32 位操作数的快速 8086。
2. 受保护的虚拟地址方式(Protected Virtual Address Unit)。对复杂的MMU 访问，以及分页和 CPU 的各种特殊性能。

(1) 在保护方式下，执行软件可进入8086的虚拟方式。这时的任务均用8086的语句来执行，从而向下兼容。

(2) 通过分页指令或模拟I／O指令，虚拟的8086软件可与80386主操作系统隔离开，并受到保护。

五、80386 宏结构的示意图如上页所示:

### 1.1.2 80386 的寄存器结构

80386共有七类32个寄存器。它们包括了全部8086、80186、80286的寄存器。80386中有些磕存器的位是没有定义的，要切记避免任何软件使用这些未定义的位，因为升级后的 CPU 有可能占用这些由 Intel 公司保留的位。为此，要遵循下述规则:

| 寄存器 | 实地址 | | 保护地址 | | 虚拟地址 | |
|---|---|---|---|---|---|---|
| | 取 | 存 | 取 | 存 | 取 | 存 |
| 通用寄存器 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 段寄存器 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 标志寄存器 | ✓ | ✓ | ✓ | ✓ | IOPL | IOPL * * |
| 控制寄存器 | ✓ | ✓ | PL＝0 | PL＝0 | × | ✓ |
| 全局描述符表寄存器 | ✓ | ✓ | PL＝0 | ✓ | × | ✓ |
| 中断描述符表寄存器 | ✓ | ✓ | PL＝0 | ✓ | × | ✓ |
| 局部描述符表寄存器 | × | × | PL＝0 | ✓ | × | × |
| 任务寄存器 | × | × | PL＝0 | ✓ | × | × |
| 调试寄存器 | ✓ | ✓ | PL＝0 | PL＝0 | × | × |
| 测试寄存器 | ✓ | PL＝0 | PL＝0 | PL＝0 | × | × |

注: ✓表示可以; ×表示不可以.

PL＝0表示只有当现行特权级别为0时，才可以对该寄存器进行存取。

IOPL * * 表示在PUSHF和POPF指令造成虚拟8086方式的I／O特权级敏感。

① 在测试各寄存器中已定义的位时，不能依据未定义的位的状态。测试要屏蔽未定义的位.

② 把这些磕存器的内容送入存储器，或其它的寄存器时，不要依据任何未定义的位

的状态。

③ 不要向未定义的位写入任何信息做为依据。

④ 在装入寄存器时，未定义的位总是写入 0。

⑤ 预先已存储的寄存器，可能不带屏蔽而被重新装入，这一点要特别注意。这些寄存器在实地址方式、保护地址方式和虚拟地址方式下的存取性和兼容性如上页表中所示：

以下对七类寄存器进行说明。

*1.1.2.1* 通用寄存器*(General Register)*

共有八个，用于存放数据或地址。

| 31 16 | 15 8 | 7 0 |
|---|---|---|
| EAX | AH | AX AL |
| EBX | BH | BX BL |
| ECX | CH | CX CL |
| EDX | DH | DX DL |
| ESI | SI | |
| EDI | DI | |
| EBP | BP | |
| ESP | SP | |

1. 数据寄存器(Data Resgister):共有四个。

(1) 累加器(Accumulate)。它与算术逻辑单元 ALU(Arithmetic Logic Unit)对数据进行算术或逻辑运算。

① EAX: 32 位累加器。主要用于双字乘法，双字除法，和输入／输出双字 I／O。

② AX: 16 位累加器。主要用于字乘法，字除法，和输入／输出字 I／O。

③ AH: 8 位累加器。主要用于字节乘法，字节除法，和输入／输出字节 I／O。

④ AL: 8 位累加器。主要用于字节乘法，字节除法，和字节 I／O，转移或十进制运算。

(2) 基数寄存器(Base Register)。主要用于寻址时存放基址。

① EBX: 32 位基数寄存器。主要用于存放 32 位基址或转移。

② BX: 16 位基数寄存器。主要用于存放 16 位基址或转移。

③ BH、BL: 8 位基数寄存器。主要用于存放 8 位基址或转移。

(3) 计数寄存器(Count Register)。主要用于循环和移位的计数。

① ECX: 32 位计数寄存器，用于串操作，循环次数计数。

② CX: 16 位计数寄存器，用于串操作，循环次数计数。

③ CL: 8 位计数寄存器，用于变量移位或循环次数计数。

(4) 数据寄存器(Data Register)。主要用于存放指令操作数做算术或逻辑运算，以及间接 I／O 端口寻址。

① EDX: 32 位数据寄存器，用于双字的乘、除和 I／O 端口寻址。

② DX: 16 位数据寄存器，用于字的乘、除和 I／O 端口寻址。

③ DH、DL: 8 位数据寄存器，用于字节的乘、除和 I／O 端口寻址。

2. 指针寄存器(Pointer Register)。共有两个。

(1) 堆栈指针寄存器(Stack Pointer Register)。与堆栈段选择器一起确定堆栈在内存中之实际位置。

① ESP: 32 位堆栈指针寄存器。

② SP: 16 位堆栈指针寄存器。

(2) 基数指针寄存器(Base Pointer Register)。在寻址中存放基址。

① EBP: 32 位基数指针寄存器，存放双字基地址。

② BP: 16 位基数指针寄存器，存放字基地址。

3. 变址寄存器(Index Register)。在带有变址的寻址中存放变址。

(1) 源变址寄存器(Source Index Register)。

① ESI: 32 位源变址寄存器，在数组操作时存放双字源操作数地址。

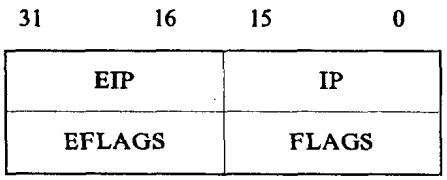
② SI: 16 位源变址寄存器，在数组操作时存放字源操作数地址。

(2) 目的变址寄存器(Destination Index Register)。

① EDI: 32 位目的变址寄存器，在数组操作时存放双字目的操作数地址。

② DI: 16 位目的变址寄存器，在数组操作时存放字目的操作数地址。

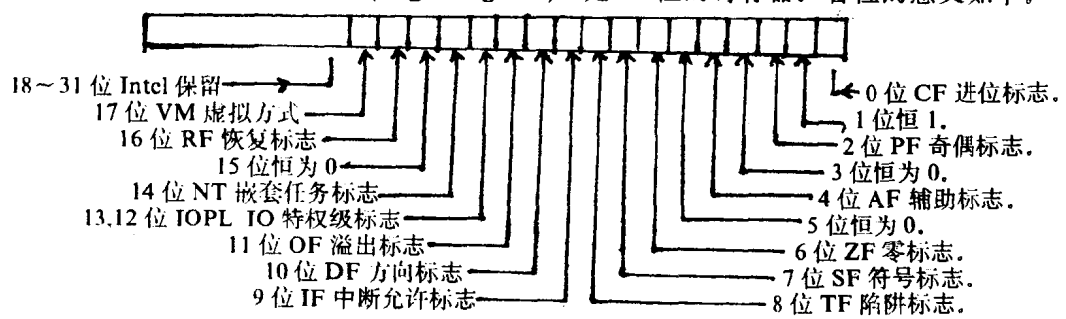
*1.1.2.2* 指令指针寄存器和标志寄存器。示意图如下




1. 指令指针寄存器:
EIP(Instruction Pointer Register).
存放下一条要取出的指令的偏移量.
它是相对于码段描述符寄存器中段
基地址的偏移量. EIP 的低 16 位称为
IP. 它由 16 位的地址操作数使用.

由于 80386 的地址线有 32 条，故指令指针是 32 位的寄存器。

2. 标志寄存器 EFLACS(flags Register)。是 32 位的寄存器。各位的意义如下。




第0位: CF —— 进位标志(Carry Flag)。用于多字节的加减运算，在移位指令或循环指令中，可将操作数的最高位(左移时)或最低位(右移时)移入标志位，它标志着:

① 若CF＝1，说明结果的最高位，即字节操作时的第7位; 字操作时的第15位; 双字

操作时的第 31 位，产生了进位或借位。

② 若 CF＝0，说明结果无进位或借位。

第2位: PF ——— 奇偶标志(Parity Flag)。用于检查数据在传输过程中有无出错，尤其是在串行通信时。

① 若 PF＝1，说明操作结果中"1"的个数为偶数。

② 若 PF＝0，说明操作结果中"1"的个数为奇数。

第4位: AF ——— 辅助进位标志(Auxitiary Carry Flag)。用于BCD码十进制算术运算指令中。

① 若AF＝1，说明在字节操作时，占四位的低半字节向高四位的高半字节有进位或借位；在字操作时，占八位的低半字向高八位的高半字有进位或借位；在双字操作时，占十六位的低字向高十六位的高字有进位或借位。

② 若 AF＝0，说明各低半部分向高半部分均无进位或借位。

第 6 位: ZF ——— 零标志(Zero Flag)。用于判断运算结果是否为 0。

① 若 ZF＝1，说明运算结果为 0。

② 若 ZF＝0，说明运算结果为非 0。

第7位: SF ——— 符号标志(Sign Flag)。由于带符号的数要由补码来表示，SF 用于对符号的测试，0 表示正，1 表示负。

① 若 SF＝1，说明结果的最高位为 1，结果是负数。

② 若 SF＝0，说明结果的最高位为 0，结果是正数。

第8位: TF ——— 陷阱标志(Trap Flag)。用于调试程序时使CPU单步操作，使每条指令执行后产生一个内部中断，以便检查程序是否有错。

① 若 TF＝1，说明 CPU 进入单步，可检查程序。

② 若 TF＝0，说明 CPU 不进入单步，程序正常执行。

第9位: IF ——— 中断允许标志(Interrupt enable Flag)。用于允许或屏蔽中断请求。对外部非屏蔽中断或内部中断，此标志不起作用。

① 若 IF＝1，允许 CPU 接收外部的可屏蔽中断请求。

② 若 IF＝0，屏蔽掉 CPU 接收外部的可屏蔽中断请求。

第10位: DF ——— 方向标志(Direction Flag)。用于控制串操作时的地址增 减量。

① 若DF＝1，串操作指令自动地减量执行，即从高地址向低地址来处理串，或者说从左向右地来处理串。

② 若DF＝0，串操作指令自动地增量执行，即从低地址向高地址来处理串，或者说从右向左地来处理串。

第11位: OF ——— 溢出标志(Overflow Flag)。判断结果是否超过了带符号的数所能表示的范围。

① 若 OF＝1，说明结果超出了范围，发生了溢出，就产生溢出中断。

② 若 OF＝0，说明结果没有超出了范围，不发生溢出，就不产生溢出中断。

第13、12位: IOPL ——— 输入／输出特权级标志(I／O Privilege Level)。用以指定 I／O 操作处于 0～3 特权级中的哪一级。

① 若任务的当前特权级CPL大于IOPL时，试图执行IN，INS，OUT，OUTS，

STI，CLI，LOCK 等指令时，将引起异常 13。

② 若任务的当前特权级CPL小于IOPL时，试图执行IN，INS，OUT，OUTS，STI，CLI，LOCK 等指令时，将能正常地执行。

第 14 位：NT ——— 嵌套任务标志(Nest Flag)，用以指明有无任务嵌套。

① 若NT＝1，说明当前执行的任务嵌套在另一个任务中。执行完被嵌套的任务后，返回外层到原来的任务中去。

② 若 NT＝0，说明无任务嵌套。

第16位：RF ——— 恢复标志(Resume Flag)。与测试寄存器TR的断点或单步操作一起使用。在断点处理之前于两条指令之间，对 RF 进行检查。

① 若RF＝1，下条指令中的所有调试故障被忽略，但在执行IRET，POP，JMP，CALL，INT 等指令时例外。这些指令要根据存储器映象所确定的值来设置 RF。如在断点服务子程序的结尾处，IRET 可弹出一个带有 RF＝1 的标志映象，并在断点地址处恢复程序的执行，而不致在同一个位置上产生另一次断点故障。

② 在成功地完成了各条指令时，RF＝0。

第17位：VM ——— 虚拟的8086方式(Virtual 8086 Mode)。允许使用虚拟的8086 方式进行操作。

① 若VM＝1，且80386在保护方式下，则80386转入虚拟的8086方式。VM只能在保护方式下。若当前特权级 CPL＝0 时，由 IRET 或在其它特权级下由任务切换来设置。VM 置位后，使所有的段在执行操作时，与在 8086 上运行时的情况一样。在仿真 8086 期间，80386 对特权操作码将产生异常 13。

② POPF 不影响 VM，但 PUSHF 将使 VM＝0。

③ 中断处理过程被压入的VM，或在任务切换期间被保存在标志位中的VM，均置成 1。其条件是被中断的程序代码正作为虚拟的 8086 任务而在执行。

第 1 位：恒为 1。

第 3，5，15 位：恒为 0。

第 18～31 位：由 Intel 公司所保留。

*1.1.2.3 段寄存器(Segment Register)*

80386存储单元的地址，是由32位段基地址和32位偏移量组成。段基址不能由段寄存器直接确定，而是保存在一个表中。段寄存器的值只是该表的一个索引。所以把段寄存器叫做段选择器，而该表叫做段描述符寄存器。

1. 段选择器共有六个 16 位的寄存器。如下所示：

(1) CS：码段选择器，表示当前的程序代码所寻址的存储器单元的选择器的值。

15 0

| |
|---|
| CS 代码段寄存器 |
| SS 堆栈段寄存器 |
| DS 数据段寄存器 |
| ES 附加段寄存器 |
| FS 附加段寄存器 |
| GS 附加段寄存器 |

(2) SS: 堆栈段选择器，表示当前的堆栈所寻址的存储器单元的选择器的值。
(3) DS: 数据段选择器，表示当前的数据所寻址的存储器单元的选择器的值。
(4) ES: 附加段选择器，表示当前的附加数据所寻址的存储器单元的选择器的值。
(5) FS: 附加段选择器，表示当前另一个数据所寻址的存储器单元的选择器的值。
(6) GS: 附加段选择器，表示当前另一个数据所寻址的存储器单元的选择器的值。

2. 段选择器对应的段描述符寄存器如下图示意:
由段选择器的值指出 32 位物理基地址、32 位段长度和描述符的各种属性。一个描述符占有八个字节。

| | | | |
|---|---|---|---|
| CS 选择器 | 32 位物理基址 | 32 位段长 | 描述符其它属性 |
| SS 选择器 | 32 位物理基址 | 32 位段长 | 描述符其它属性 |
| DS 选择器 | 32 位物理基址 | 32 位段长 | 描述符其它属性 |
| ES 选择器 | 32 位物理基址 | 32 位段长 | 描述符其它属性 |
| FS 选择器 | 32 位物理基址 | 32 位段长 | 描述符其它属性 |
| GS 选择器 | 32 位物理基址 | 32 位段长 | 描述符其它属性 |

31 16 15 0

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 段基地址 15...0 (第二个字) | | | | | 段长度 15...0 (第一个字) | | | | | |
| 基址 31～24 | G | D | 00 | 段长 19～16 | P | DPL | S | 类型 | A | 基址 23～16 |

其中: 段长度—— 0～7位占第一字节，8～15占第二字节，16～19占第七字节的 0～3 位。共 20 位长。
段基址—— 0～7位占第三字节，8～15占第四字节，16～23占第五字节，24～31 位占第八字节。共 32 位长。
A —— 标志着已存取位，该位占第六字节的第 0 位。
类型 —— 共三位，表示段的类型，它占第六字节的第 1～3 位。
S —— 占第6字节的第4位。S＝1是系统描述符，S＝0是代码或数据描述符。
DPL —— 表示描述符的特权级 0～3，它占第六字节的第 5、6 位。
P —— P＝1 表示已存在，P＝0 表示不存在。它占第六字节的第 8 位。
00 —— 为与将来的处理器兼容必须设置为 0。占第七字节的第 4、5 位。
D —— 在码段描述符中识别默认操作数的大小，D＝1是32位段，D＝0中16位段。该位占第七字节的第 6 位。
G —— 粒度标志，G＝1表示是代码段描述符或和数据段描述符，G＝0表示是系统描述符。该位占第七字节的第 7 位。

3. 每个段描述符寄存器保存有32位的段基址，32位的段长度，以及其它一些属性。在段选择器的值确定之后，会自动地按值索引。从表中取出四个字长的描述符，自动地装入到描述符寄存器中。当存储器访问时，从描述符寄存器中取段基址作为线性地址计算中的一个元素，而不必在访问时查表，这就是 80386 寻址方式的硬件支持。
4. 在80286／80386中，由段选择器和段描述符相对应的数据结构来确定存储器单元的段地址，这样，就可以用 16 位的段选择器来确定 80286 的 24 位和 80386 的 32 位段基地址，还可以确定段的一些属性，如特权级，粒度位，段类型或默认操作数的大小等。段基址是由一个 8 字节的描述符确定的。

*1.1.2.4 系统编址寄存器*

80386 有四个特殊的段寄存器。它支持访问保护方式的描述符表。

1. 由相关的描述符，可以组成一个表。在 80386 中共有四种表，如下所述:

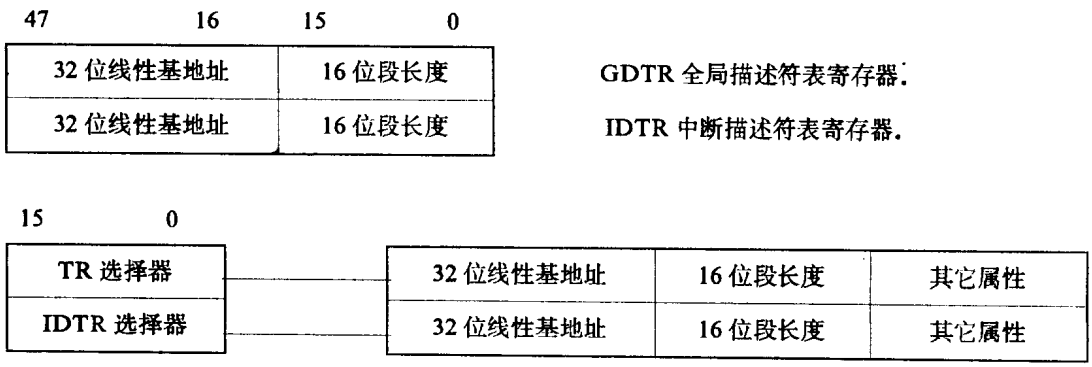
(1) 全局描述符表 GDT(Global Descriptor Table)。

(2) 局部描述符表 LDT(Local Descriptor Table)。

(3) 中断描述符表 IDT(Interrupt Descriptor Table)。

(4) 任务状态段表 TSS(Task State segment Table)。

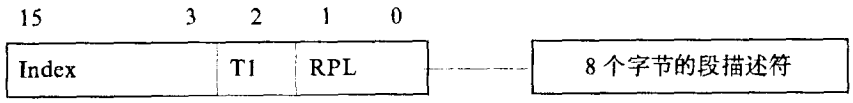
2. 这些表的基地址和它们的大小均存放在相应的寄存器中，於是就对应了四个寄存器示意如下:




(1) GDTR: 全局描述符表寄存器。它保存GDT的32位线性基地址和16位的段长度。每个表最大为 64K，每个描述符为 8 个字节，所以每个表最多有 8K 个描述符。由于 GDT 对系统中的所有任务都是全局性的，所以 GDT 表所在之段，由 32 位的线性地址(若分页时，就转向页转换)和 16 位段长来确定。

(2) IDTR: 中断描述符表寄存器，它保存IDT的32位线性基地址和16位的段长度。每个表最大为 64K，每个描述符为 8 个字节，所以每个表理论上最多有 8K 个描述符。而实际上只有 256 个中断或异常向量，所以，IDT 表中实际上最多只有 256 个中断描述符。由于 IDT 对系统中的所有任务都是全局性的，所以 IDT 表所在之段，由 32 位的线性地址(若分页时，就转向页转换)和 16 位段长来确定。

(3) LDTR: 局部描述符表寄存器。由于LDT是面向任务的，故其所在段不能由表本身确定，而应该由任务来确定，即由任务的系统段寄存器中之选择器值来确定。所以，LDTR 只是一个 16 位的选择器，如下图所示。




其中: RPL --- 申请优先级 0～3，占第 1～0 位。

TI --- 表标志，占第2位。 TI＝0表示是全局描述符表GDT，

TI＝1表示是局部描述符表LDT。

Index --- 表中之索引，占第 15～3 位。其值指向表中所对应的描述符。

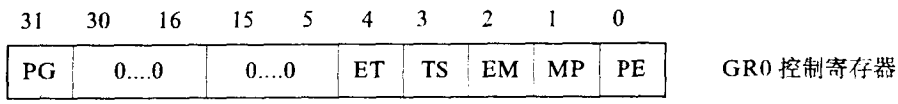
在80286／80386中是自动装入的，描述符寄存器是不可见的，它作为对存储器单元访问的硬件支持，以加速存储器单元的访问。

(4) TR: 任务状态段寄存器。其结构与 LDTR 局部描述符表寄存器一样。

*1.1.2.5 控制寄存器(Control Register)*

为了保存全局性的机器状态字MSW，80386设置了三个32位的控制寄存器，它们与系统编址寄存器一起，保存着系统中所有任务的机器状态。

1. CR0: 机器控制寄存器。主要用于扩展，控制协处理器。它定义了六个控制位和状态位。对于 32 位的 CR0 的存取，应使用以下语句: MOV CR0,Reg;通用寄存器内容送入 CR0 控制寄存器。而不能使用 LMSW 和 SMSW 指令。这两条指令虽然列在 80386 的指令表中，只是为了与 80286 兼容，这两条指令只能对 CR0 的低 16 位进行操作，而不能包括 PG 位。CR0 寄存器示意如下:




(1) PG --- 允许分页标志(Paging Enable)，占第 31 位。

① 若 PG＝1，允许片内分页部件工作。

② 若 PG＝0，禁止片内分页部件工作。

(2) ET --- 扩展的协处理器的类型(Processor Extension Type)。占第4位。 80386 复位后，由 ERROR#输入信号的电平来确定，也可以由程序控制向 CR0 送数。

① 若 ET＝1，使用与 80387 协处理器兼容的 32 位规程。

② 若ET＝0，使用与80287协处理器兼容的16位规程。为了严格与80286兼容， ET不受 LMSW(装入机器状态字)指令的影响。

(3) TS --- 任务切换(Task Switched)，占第3位。用来标志协处理器的任务是否完成?

① 若TS＝1，当任务切换完成后，TS自动置成1．若一条协处理器操作指令使TS＝1，且MP＝1时，则将会引起协处理器无效陷阱(异常7)．此异常处理程序保存着属于上一个任务的80287／80387内容，装入属于当前任务的80287／80387的状态。

② 若TS＝0，当前任务切换没有完成．在返回失败的协处理器操作码之前，TS也被清零。

(4) EM ——— 模拟协处理器(Emulate Coprocessor)．占第2位。

① 若EM＝1，使协处理器的所有操作码均产生异常7(无效陷阱)．且WAIT指令不受EM置位的影响。

② 若EM＝0，允许协处理器的所有操作码在实际的80287／80387协处理器上执行。复位后的默认状态是80387。

(5) MP ——— 监控协处理器(Monitor Coprocessor)．占第1位。

① 若MP＝1，且TS＝1时，WAIT指令产生无效陷阱异常7．在任务切换操作完成时，TS自动置位。

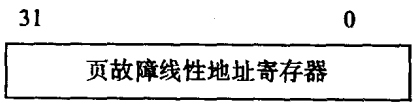
② 若MP＝0，且TS＝1时，WAIT指令不产生无效陷阱异常7。

(6) PE ——— 保护允许(Protection Eable)．占第0位。

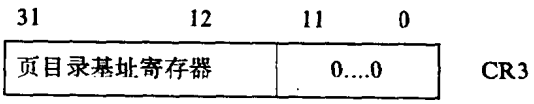
① 若PE＝1，CPU进入保护状态。PE可由加载机器状态字MSW，或者加载CR0指令来置位。

② 若PE＝0，CPU再次处于实地址方式，不能由LMSW指令来复位。

2. CR2: 32位的页故障线性地址寄存器。当检测到发生页故障时，将该故障的最后地址送入CR2寄存器。在引用CR2时，把它送到页失败处理程序的堆栈上的错误码处。页故障线性地址寄存器以提供页失败的附加信息。该寄存器的示意图如下:




3. CR3: 页目录基址寄存器，示意图如下:— 该寄存器包含着页目录表的物理地址。由于80386的页目录表是按页对齐的，即按4K字节对齐。故CR3的低12位被忽略。不能由任务切换状态TSS来改变CR3的内容，也不能用任何值装入CR3，这会使分页单元高速缓冲器中的所有缓存的页表输入变成无效。




*1.1.2.6 调试寄存器(Dubeg Register)*

有八个32位的调试寄存器DR7～DR0。其中六个可供程序人员使用，如左图所示。