第2章 微处理器及其体系结构

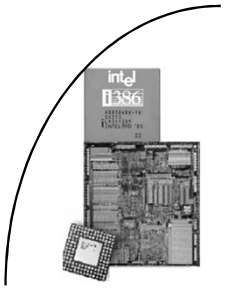
罗文坚
中国科大 计算机学院

http://staff.ustc.edu.cn/~wjluo/mcps/

1

本章内容

- 微处理器的内部体系结构
- 实模式存储器寻址
- 保护模式存储器寻址简介
- 内存分页
- 平展模式内存



2

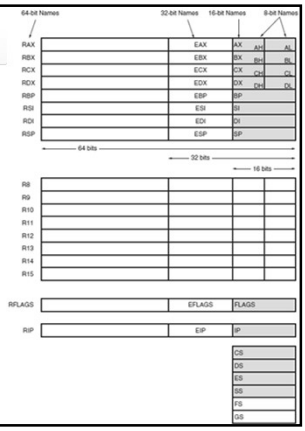
程序设计模型

- 程序可见的寄存器：程序设计期间必须使用的。
  - 程序设计模型
- 程序不可见的寄存器：在应用程序设计期间不能直接寻址，但系统程序设计期间可能间接使用到。
  - 只有80286及更高档的微处理器才包含程序不可见寄存器，用于控制和操作保护模式存储系统和其他特征。

3

程序设计模型

- 8086~Core2的程序设计模型（含64位扩展）
- 8位寄存器
- 16位寄存器
- 32位寄存器
- 64位寄存器
- 多功能寄存器
  - 又称为通用寄存器
- 专用寄存器



4

程序设计模型

- 8086，8088和80286包含16位内部结构。
- 80386~Core2包括全部的32位内部结构。
- Pentium 4和Core2在使用64位模式操作时，也包括64位寄存器。

5

程序设计模型

- 8位寄存器
  AH，AL，BH，BL，CH，CL，DH，DL
  - 例如，ADD AL，AH
- 16位寄存器
  AX，BX，CX，DX，BP，SI，DI，SP
  IP，FLAGS
  CS，DS，ES，SS，FS，GS
  - 例如，ADD DX，CX
  - 注意1：AX，BX，CX，DX个包含2个8位寄存器。
  - 注意2：FS和GS值用于80386以上微处理器。

6

程序设计模型

- 32位寄存器
  EAX，EBX，ECX，EDX，ESI，EDI，EBP，ESP
  EIP，EFLAGS
  - 例如，**ADD ECX，EBX**
  - 注意：只用于**80386**以上微处理器。
- 64位寄存器
  RAX，RBX，RCX，RDX，SRI，RDI，RBP，RSP
  RIP，RFLAGS
  R8，R9，R10，R11，R12，R13，R14，R15
  - 例如，**ADD RCX, RBX**

7

64位模式下的程序设计模型

❖64位模式下，需注意：

1. 64位寄存器R8~R15是可以按照字节、字、双字或者四字的方式寻址的。按字节寻址部分是最低8位。
2. 不支持把其中的第8~15位作为1字节来直接寻址。
3. 高字节寄存器（AH、BH、CH、DH）不能与由R8~R15所表示的字节在同一指令中寻址。

寄存器大小	控制字	访问位数	示例
8位	B	7~0	MOV R9B, R10B
16位	W	15~0	MOV R10W, AX
32位	D	31~0	MOV R14D, R15D
64位	——	63~0	MOV R13, R12

8

多功能寄存器

- **RAX（累加器，Accumulator）**
  - **RAX，EAX，AX，AH，AL**
  - 在乘法、除法及一些调整指令中有专门用途。
  - 在**386**以上微处理器中，**EAX**及**RAX**可用于保存访问存储单元时的偏移地址。
- **RBX（基址，Base）**
  - **RBX，EBX，BX，BH，BL**
  - 可用于保存访问存储单元时的偏移地址。

9

多功能寄存器

- **RCX（计数，Counter）**
  - **RCX，ECX，CX，CH，CL**
  - 可保存许多指令的计数值，如移位指令、循环指令、串指令等。
  - 在386以上微处理器中，ECX及RCX可用于保存访问存储单元时的偏移地址。
- **RDX（数据，DATA）**
  - **RDX，EDX，DX，DH，DL**
  - 可用于保存乘法形成的部分结果，或除法指令的部分被除数。
  - 在386以上微处理器中，EDX及RDX可用于保存访问存储单元时的偏移地址。

10

多功能寄存器

- **RBP（基指针， Base Pointer）**
  - **RBP，EBP，BP**
  - 可用于保存访问存储单元时的偏移地址。
- **RDI（Destination Index）**
  - **RDI，EDI，DI**
  - 可用于保存访问存储单元时的偏移地址。
  - 常用于寻址串指令的目的操作数。
- **RSI（Source Index）**
  - **RSI，ESI，SI**
  - 可用于保存访问存储单元时的偏移地址。
  - 常用于寻址串指令的源操作数。

11

多功能寄存器

- **R8~R15**
  - 只存在于**Pentium 4**和**Core 2**中**64**位扩展允许的情况下。
  - 这些寄存器中的数据是用于通用目的的。
  - 可以按照**64**、**32**、**16**、**8**位大小寻址。
  - **8**位部分是寄存器最右边的**8**位。
  - 第**8**位~第**15**位不能按照一个字节直接寻址。

12

专用寄存器

- 专用寄存器包括：
  - RIP（指令指针）
  - RSP（堆栈指针）
    - 有参考书把RSP/ESP/SP作为一个通用寄存器
  - RFLAGS（标志寄存器）
  - CS，DS，ES，SS，FS，GS（段寄存器）

13

RIP（指令指针）

- RIP：在64位模式中，目前包括40位地址，指向要执行的下一条指令的偏移值，该偏移值相对于指令所在代码段的基地址（段基址）。
- EIP：32位指令指针，用于32位微处理器中。
- IP：在8086和80286中，指令指针为16位寄存器。
- 程序员不能对RIP/EIP/IP进行直接存取操作。程序中的转移指令、返回指令以及中断处理能对RIP/EIP/IP进行操作。

14

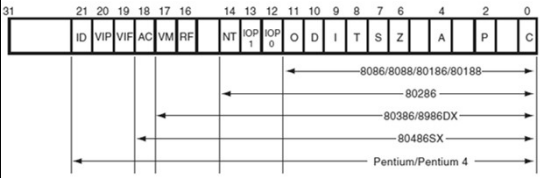
RSP（堆栈指针）

- RSP
  - RSP，ESP，SP
  - 用于寻址一个称为堆栈的存储区。

15

RFLAGS（标志寄存器）

- RFLAGS：用于指示微处理器的状态并控制它的操作。
  - FLAGS：8086~80286
  - EFLAGS：32位微处理器，80386及其以上
  - RFLAGS：64位微处理器（高32位保留）



16

RFLAGS（标志寄存器）

- CF：进位标志（Carry Flag）
  - CF=1，表示有进位或借位。
- PF：奇偶标志（Parity Flag）
  - 奇校验
  - 早期Intel微处理器在数据通信环境中校验数据的一种手段。奇偶校验标志在现代程序设计中很少使用。目前，奇偶校验通常由数据通信设备完成，而不是微处理器完成。
- AF：辅助进位标志（Auxiliary Carry）
  - 加法结果中第3位与第4位之间的进位，或减法结果中第3位与第4位之间的借位。
  - BCD码加减法指令使用这一标志。

17

RFLAGS（标志寄存器）

- ZF：零标志（Zero Flag）
  - ZF=1表示结果为0；ZF=0表示结果不为0。
- SF：符号标志（Sign Flag）
  - 保存执行算术或逻辑运算指令后所得结果的算术符号。
  - SF=1，表示符号位（结果的最左1位）为1；SF=0，表示符号位（结果的最左1位）为0。
- TF：陷阱标志（Trap）
  - 使能微处理器芯片上的调试功能。TF=1，根据调试寄存器和控制寄存器的指示中断程序流；TF=0，禁止陷阱（调试）功能。
  - 例，VC++利用陷阱特性和调试寄存器调试有缺陷的软件。

18

RFLAGS（标志寄存器）

- IF：中断允许标志（INTR Enable）
  - 控制INTR（中断请求）输入引脚的操作。
  - IF=1，允许中断；IF=0，禁止中断。
- DF：方向标志 （Direction Flag）
  - 用于串操作指令中的地址增量修改（DF＝0）还是减量修改（DF＝1）。
- OF：溢出标志（Overflow Flag）
  - 用于有符号数运算。若运算过程中发生了溢出，则OF＝1。对于无符号数操作，不考虑该标志。

19

RFLAGS（标志寄存器）

- IOPL（Input/Output Privilege Level）：I/O特权级标志
  - 用于保护方式，指示I/O设备的特权级，IOPL的2位代码决定4级特权级。
  - 如果当前特权级（Current Privilege Level，CPL）高于IOPL，则I/O指令能顺利执行；否则产生中断（异常13故障），使任务挂起。
- NT（Nested Task）：嵌套任务标志
  - 用于保护方式。
  - 指示当前执行的任务是否嵌套在另一任务中，该位的置位和复位通过向其它任务的控制转移来实现。
  - IRET指令会检测NT的值。若NT=0，则执行中断的正常返回；若NT＝1，则执行任务切换操作。

20

RFLAGS（标志寄存器）

- RF（Resume）：恢复标志
  - 该标志和调试寄存器配合使用，用于控制调试失败后强制程序恢复，返回断点继续执行。
  - 断点处理前在指令边界上检查该位。
    1) 若RF=1，则下条指令的任何调试故障都被忽略，不产生异常中断。
    2) 若RF=0，调试故障被接受，指令断点可以产生调试异常。
  - 每条指令成功执行完毕，如无故障出现，则RF自动被清0。但IRET、POPF以及引起任务切换的JMP、CALL和INT指令除外，这些指令将RF置成存储器映象指定的值。

21

RFLAGS（标志寄存器）

- VM（Virtual 8086 Mode）：V86方式
  - VM＝1，指示处理器在V86模式下工作；
  - VM用于在现代Windows环境下仿真DOS。
  - VM用于在保护模式系统中选择虚拟操作模式。虚拟模式系统允许多个1MB长的DOS存储器分区共存于存储系统中。这样，可以允许系统执行多个DOS程序。
- AC（Alignment Check）：对界检查
  - 仅用于80486 CPU，供80487sx协处理器使用，用作同步。
  - 进行字或双字寻址时，如果地址不处于字或双字的边界上，则AC=1。
  - 只有80486SX包含AC标志，用于同步配套的80487SX。

22

RFLAGS（标志寄存器）

- VIF（Virtual Interrupt）：虚拟中断标志
  - Pentium开始引入。
  - 在虚拟方式下提供中断允许标志位IF的副本(copy)。
- VIP（Virtual Interrupt Pending）：虚拟中断挂起标志
  - Pentium开始引入。
  - 在多任务环境下，为操作系统提供虚拟中断挂起信息。
  - 此标志置位为1指示有一个中断被挂起。
- ID：标识标志。
  - 用来指示Pentium~Pentium 4对CPUID指令的支持状态。

23

RFLAGS（标志寄存器）

- 关于六个标志位CF，PF，AF，ZF，SF，OF：
  - 在执行算术和逻辑指令后会改变；而对于任何数据传送指令或程序控制操作，这些标志都不改变。
  - 注意：任务切换。

例，-2147483648（0x8000 0000）<2147483647

结果为FALSE?

int i = -2147483648;
i < 2147483647

做减法进行比较 →

A: 100 • • • 0
B: 011 • • • 1
Sub: 1

结果为true

- 无符号整数，大于，CF∪ZF=0。 (注：CF=0, ZF=0)
- 有符号整数，小于，OF⊕SF=1。 (注：OF=1，SF=0)

24

段寄存器

- 6个段寄存器：CS, DS, SS, ES, FS, GS
  - 用来保存标志现行可寻址存储段的段基址或段选择子（Selector，又称选择符）。
- 16位微处理器： CS, DS, SS, ES，保存段基址。
- 32位和64位微处理器：CS, DS, SS, ES, FS, GS，保存段选择子。

25

段寄存器

- CS：Code Segment，代码段寄存器
  - 定义代码段存储区的起始地址（及有关属性）。
  - 代码段是微处理器用来存放代码（程序，包括过程）的一段存储区。
  - 在实地址方式下，定义了一段64KB存储区的起始地址。
  - 在保护方式下，用来选择一个描述符（又称为描述子），该描述符用来描述一个代码段的若干特性——起始地址、段限以及访问权等。
  - 最大段限在8086及80286中为64KB，而在80386及其以上的微处理器中则为4GB。64位模式下，代码段寄存器仍用于平展模式，但用法不同。

26

段寄存器

- DS：Data Segment，数据段寄存器
  - 定义了数据段存储区的起始地址（及有关属性）。
  - 数据段是存放供程序使用的数据的一段存储区。
  - 数据段中的数据按其给定的偏移地址值offset（或称有效地址EA，Effective Address）来访问。
  - 数据段的长度规定同代码段。

27

段寄存器

- SS：Stack Segment，堆栈段寄存器
  - 定义了堆栈段存储区的起始地址（及有关属性）。
  - 堆栈段是一段用作堆栈的存储区。
  - 堆栈段现行的入口地址由堆栈指针RSP（或ESP、SP）决定。
  - RBP（或EBP、BP）也可寻址堆栈段中的数据。

28

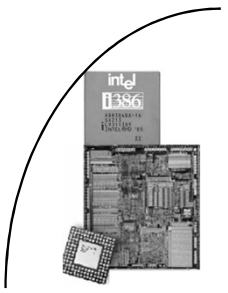
段寄存器

- ES：Extra Segment，附加段寄存器
  - 定义了附加段存储区的起始地址（及有关属性）。
  - 附加段是一段附加的数据段，通常供串操作类指令用于存放目的串数据。
- FS、GS
  - 80386以上的微处理器（含80386）有两个新增的附加的段存储区。
  - FS和GS用来定义这两个附加的数据段存储区的起始地址（及有关属性）。
- DS、ES、FS和GS的选择子用来指出现行的数据段。

29

本章内容

- 微处理器的内部体系结构
- 实模式存储器寻址
- 保护模式存储器寻址简介
- 内存分页
- 平展模式内存



30

实模式存储器寻址

- 8086：只能工作于实模式。
- 80286及其以上：可以工作在实模式或者保护模式。
- 实模式的特点：
  - 只允许微处理器寻址起始的1MB存储器空间，即使是Pentium 4和Core 2微处理器。
  - 微处理器每次加电或复位后，以实模式开始工作。
  - Windows操作系统不使用实模式。
  - 如果Pentium 4或Core 2处于64位模式中，则不能执行实模式应用程序，从而DOS应用程序不存在于64位模式。
    - 除非为64位模式编写虚拟DOS。

31

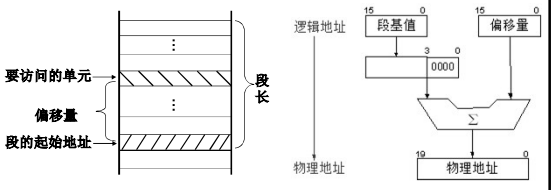
段和偏移

- 实模式下，用段地址和偏移地址的组合来访问存储单元。所有实模式存储单元的地址都有段地址加偏移地址组成。
- 实模式段的长度总是64KB。
- 段地址（Segment Address）装在段寄存器中，确定64KB存储段的起始地址。
- 偏移地址（Offset Address）用于在64KB存储段内选择任一单元。

32

实模式下的段式地址管理

- 逻辑地址：程序设计时，使用的是逻辑地址。逻辑地址由“段基址”和“偏移量”构成（均为16位）。
  - 表示为“段基址:偏移量”
- 物理地址： CPU访问存储器时，在地址总线上实际送出的地址。物理地址（20位）＝段基值×16＋偏移量。



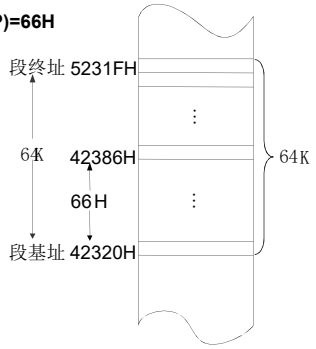
33

地址计算示例

- 例1. 设(CS)=4232H, (IP)=66H

则物理地址计算如下：

4 2 3 2 0 H
＋） 6 6 H
4 2 3 8 6 H



34

实模式下的段式地址管理

- 有时，寻址方式将多个寄存器内容与一个位移量结合，形成偏移地址。
  - 此时，如果这些值超过FFFFH，则将进位舍去。
- 例如， MOV AX, [SI+3000H]
  - 设DS=4000H，SI=F000H。
  - 因为SI+3000H=12000H，则偏移地址2000H。
  - 由此，物理地址为4000：2000H或42000H。

35

实模式下的段式地址管理

- 在80286（有专门外部电路）及80386~Pentium 4中，当段地址是FFFFH，而且系统中安装了用于DOS的驱动程序HIMEM.SYS时，可以寻址64KB减16字节的附加存储器区域。
  - 这个可寻址的存储器区域（0FFFF0H~10FFEFH）称为高端存储器。
  - 用段地址FFFFH生成地址时，地址A20引脚被置位（如果支持）。
- 例如，段地址为FFFFH，偏移地址为4000H。
  - 如果支持A20，则寻址FFFF0H+4000H，即103FF0H.
  - 如果不支持A20，则寻址03FF0H，即A20为0。

36

段的划分

- 段的划分：定长，可连续、可离散、可覆盖、可重叠。
  - “碎片”
  - 每个存储单元有唯一的物理地址，但它却可由不同的“段基址”和“偏移量”组成。
  - 例如：
    1200H:0345H → 12345H
    1100H:1345H → 12345H

37

默认段和偏移寄存器

- 关于存储段的访问，微处理器有一套规则，既适用于实模式，也适用于保护模式。
  - 定义了各种寻址方式中段寄存器和偏移地址寄存器的组合方式。
- 默认的“16位段+偏移”寻址组合：

Segment	*Offset*	*Special Purpose*
CS	IP	Instruction address
SS	SP or BP	Stack address
DS	BX, DI, SI, an 8- or 16-bit number	Data address
ES	DI for string instructions	String destination address

38

默认段和偏移寄存器

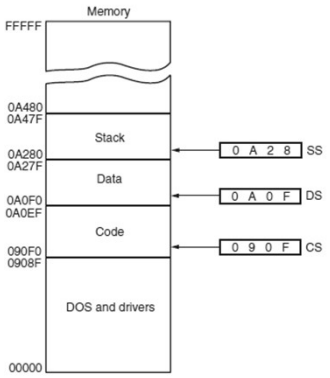
- 默认的“32位段+偏移”寻址组合：

Segment	*Offset*	*Special Purpose*
CS	EIP	Instruction address
SS	ESP or EBP	Stack address
DS	EAX, EBX, ECX, EDX, ESI, EDI, an 8- or 32-bit number	Data address
ES	EDI for string instructions	String destination address
FS	No default	General address
GS	No default	General address

39

段的加载

- 含有代码段、数据段和堆栈段的应用程序装入DOS系统存储器中。



40

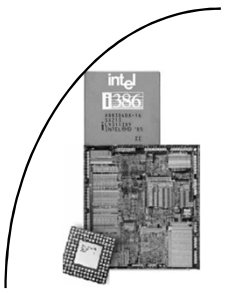
段和偏移寻址机制允许重定位

- 可重定位程序（relocatable program）：可以放入存储器的任何区域，且不需修改仍能执行的程序。
- 可重定位数据（relocatable data）：可以放入存储器的任何区域，且不需修改就可以被程序引用的数据。
- “段+偏移”寻址机制允许程序和数据的重定位。

41

本章内容

- 微处理器的内部体系结构
- 实模式存储器寻址
- 保护模式存储器寻址简介
- 内存分页
- 平展模式内存



42

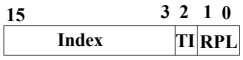
保护模式存储器寻址

- 形式上，依然是“段基址+偏移量”
  - 因此，很多保护模式指令和实模式指令是相同的，但访问存储器时的解释方法不同。
- 不同：
  - 段寄存器中不是直接存放“段基址”，而是存放着一个“选择子”，用于选择描述符表中的一个描述符。描述符（Descriptor）包含了存储段的位置、长度和访问权限。
  - 在32位微处理器中，偏移地址是32位。
- 为32位保护模式编写的程序也可以在Pentium 4的64位模式下运行。

43

选择子

- 段选择子寄存器
  - 保护模式下CS、DS、SS、ES、FS、GS寄存器称为段选择子寄存器，其值不再是段基址而是选择子，它从描述符表中选择一个定义存储器段大小和属性的描述符。



- RPL: 请求特权级
  0~3特权级
- TI: 表指示符
  0——使用全局描述符表
  1——使用局部描述符表
- Index（描述符索引）：选择描述符表中的表项。

44

描述符和描述符表

- 段寄存器可以访问两个描述符表：全局描述符表和局部描述符表。
- 全局描述符表（Global Descriptor Table，GDT）：定义了能被系统中所有任务公用的存储分段，可以避免对同一系统服务程序的不必要的重复定义与存储。
- 通常GDT中包含了操作系统使用的代码段、数据段、任务状态段以及系统中各个LDT所在段的描述符。
  - 注意：中断服务程序所在段的段描述符在IDT中。

45

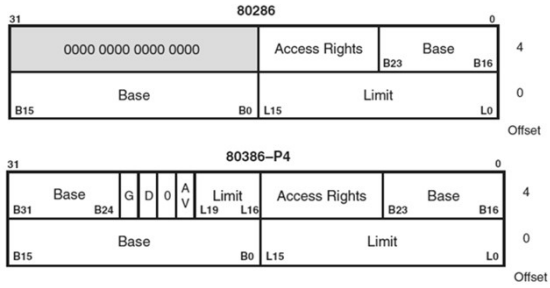
描述符和描述符表

- 局部描述符表（Local Descriptor Table，LDT）包含了与某个任务相关联的段描述符。
  - 在设计操作系统时，通常一个任务有一个独立的LDT。
  - LDT提供了将一任务的代码段、数据段与操作系统的其余部分相隔离的机制。
- 每个描述符表包含8192个描述符，所以任何时刻应用程序最多可以有16384（2^{14}）个描述符。
- 一个描述符对应一个存储段，一个存储段最大可达4GB，则应用程序能够访问4G*16384B=64TB。

46

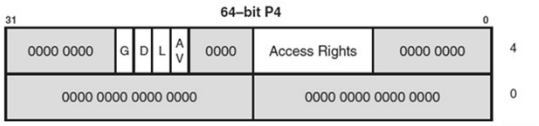
描述符

- 每个描述符占8个字节。



47

描述符



- 描述符的基地址（Base address）
- 段界限（Segment limit）
- 粒度位（Granularity bit，G）
  - G=1，界限值以4KB为单位；
  - G=0，以字节为单位。

48

描述符

- 64位描述符中的L位：
  - L=0，选择32位兼容模式。
  - L=1，选择带64位扩展的64位地址（Pentium 4 ~ Core 2）。
- 注意：在64位保护模式下，描述符中没有段基址和界限，只包含一个访问权限字节和若干控制位。
  - 为了实现64位操作，所有段的基地址都是00 0000 0000H。
  - 64位段没有边界检查。

49

描述符

- 例、设段基址为1000 0000H，段界限为001FFH，G位为0，求段的起始地址和段的结束地址。
  - 段的起始地址=段基址=1000 0000H
  - 段的结束地址=段基址+段限
    = 1000 0000H+ 0 01FFH
    = 1000 01FFH
- 例、若G位为1，其余数据与上例相同，则
  段的结束地址=段基址+段限
  = 1000 0000H+ (0 01FFH+1)*4K-1
  = 101F FFFFH

50

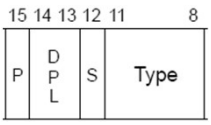
描述符

- AV位：Available for use by system software，提供给系统软件使用。
  - 可用于指示段是否有效。AV=0，段有效；AV=1，段有效。
- D位：指示保护模式下或实模式下80386~Core2指令如何访问寄存器和存储器数据。
  - 如果D=0，则指令与8086~80286兼容，是16位指令。在默认情况下，用16位偏移地址和16位寄存器。这种模式通常称为16位指令模式或DOS模式。
  - 如果D=1，则指示32位指令模式、32位段。

51

描述符

- 访问权限字节



- P：P=0，段不在内存中；P=1，段在内存中。
- DPL：描述符特权级，取值0~3，确定段的特权级，为任务允许访问该段的最低特权级。
- S：S=0，表示该描述符为系统段描述符；S=1表示该描述符为代码或数据段描述符。
- Type：段的类型（与段描述符类型相关）。

52

代码或数据段描述符的类型

Type Field					Descriptor Type	Description
Decimal	11	10 E	9 W	8 A		
0	0	0	0	0	Data	Read-Only
1	0	0	0	1	Data	Read-Only, accessed
2	0	0	1	0	Data	Read/Write
3	0	0	1	1	Data	Read/Write, accessed
4	0	1	0	0	Data	Read-Only, expand-down
5	0	1	0	1	Data	Read-Only, expand-down, accessed
6	0	1	1	0	Data	Read/Write, expand-down
7	0	1	1	1	Data	Read/Write, expand-down, accessed
		C	R	A		
8	1	0	0	0	Code	Execute-Only
9	1	0	0	1	Code	Execute-Only, accessed
10	1	0	1	0	Code	Execute/Read
11	1	0	1	1	Code	Execute/Read, accessed
12	1	1	0	0	Code	Execute-Only, conforming
13	1	1	0	1	Code	Execute-Only, conforming, accessed
14	1	1	1	0	Code	Execute/Read-Only, conforming
15	1	1	1	1	Code	Execute/Read-Only, conforming, accessed

53

系统段描述符的类型

Type Field					Description
Decimal	11	10	9	8	
0	0	0	0	0	Reserved
1	0	0	0	1	16-Bit TSS (Available)
2	0	0	1	0	LDT
3	0	0	1	1	16-Bit TSS (Busy)
4	0	1	0	0	16-Bit Call Gate
5	0	1	0	1	Task Gate
6	0	1	1	0	16-Bit Interrupt Gate
7	0	1	1	1	16-Bit Trap Gate
8	1	0	0	0	Reserved
9	1	0	0	1	32-Bit TSS (Available)
10	1	0	1	0	Reserved
11	1	0	1	1	32-Bit TSS (Busy)
12	1	1	0	0	32-Bit Call Gate
13	1	1	0	1	Reserved
14	1	1	1	0	32-Bit Interrupt Gate
15	1	1	1	1	32-Bit Trap Gate

54

系统段描述符的类型

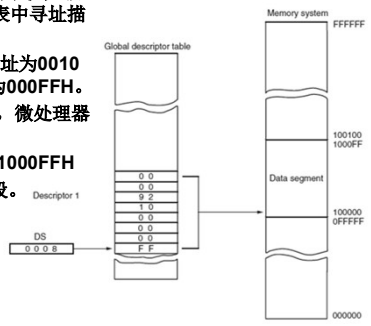
TYPE为4个字节，共有16种类型。其中：

2，LDT	5，任务门
9，TSS，非忙	C，调用门
B，TSS，忙	E，中断门
	F，陷阱门

55

分段寻址示意

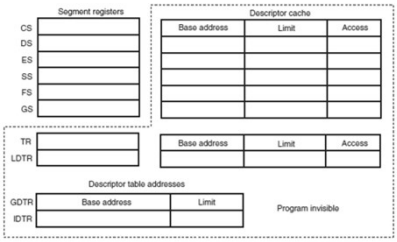
- DS=0008H，用请求优先级00从全局描述符表中寻址描述符1。
- 描述符1中的基地址为00100000H，段界限为000FFH。
- 0008H装入DS后，微处理器将使用位于 00100000H~001000FFH 的区域作为数据段。



56

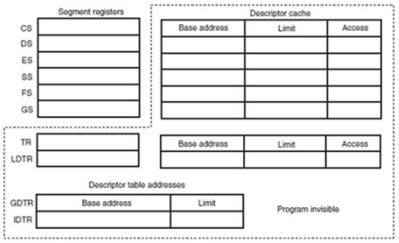
程序不可见寄存器

- 存储系统中有全局描述符表和局部描述符表。为了访问和指定这些表的地址，微处理器中包含一些程序不可见寄存器。
  - 程序不可见寄存器不直接被软件访问，但其中一些寄存器可以被系统软件访问。



57

程序不可见寄存器

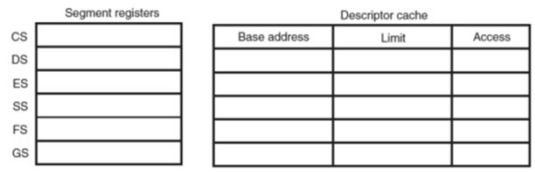


- 80286没有FS和GS，以及相应的Cache。
- 80286的基地址为24位，界限为16位，访问权限8位。
- 80386~Pentium Pro的基地址32位、界限20位、访问权限12位。

58

程序不可见寄存器

- 段寄存器及其高速缓冲存储器（Cache）
  - 段寄存器：程序员可见。
  - 段描述符高速缓冲存储器：对程序员透明，又称段描述符寄存器。



59

程序不可见寄存器

- 段寄存器及其高速缓冲存储器（Cache）
  - 每当一个新的“段号”放入段寄存器时，微处理器就访问一个描述符表，并把描述符装入该段寄存器对应的高速缓冲存储器区域内。
  - 该描述符一直保存在高速缓冲存储器区域内，并在访问内存段时使用，直到“段号”发生变化。这就允许微处理器重复访问一个内存段时，不必每次去查询描述符表。

60

程序不可见寄存器

- GDTR（Global Descriptor Table Register），全局描述符表寄存器，共有48位。
  - 高32位保存全局描述符表的线性基地址。
  - 低16位是表限字段，即表的最大长度仅64KB。
- IDTR（Interrupt Descriptor Table Register），中断描述符表寄存器，共有48位。
  - 高32位用于保存中断描述符表IDT的32位线性基地址。
  - 低16位是表限字段，表的最大长度也是64KB。

	Base address	Limit
GDTR		
IDTR		

61

全局描述符表

- 一个系统只能有一个全局描述符表（GDT）。
- 全局描述符表：定义了能被系统中所有任务公用的存储分段，可以避免对同一系统服务程序的不必要的重复定义与存储。
- 通常GDT中包含了操作系统使用的代码段、数据段、任务状态段以及系统中各个LDT所在段的描述符。
  - 注意：中断服务程序所在段的段描述符在IDT中。
- GDT本身不是一个段；它是一个在线性地址空间的数据结构。

62

全局描述符表寄存器GDTR

例：(GDTR)=0010 0000 0FFFH，求GDT在存储器中的起始地址，结束地址，表的大小，表中可以存放多少个描述符？

解：GDT的起始地址为0010 0000H
GDT的结束地址为
0010 0000H+0FFFH=0010 0FFFH
表的大小为
0FFFH+1=4096字节
表中可以存放
4096/8=512个描述符

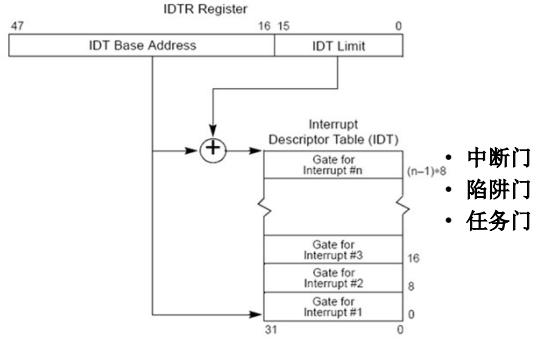
63

中断描述符表IDT

- 中断描述符表（Interrupt Descriptor Table，IDT）最多包含256个中断服务程序的位置的描述符。
- 系统所使用的每种类型的中断在IDT中必须有一个描述符表项。
- IDT的表项通过中断指令、外部中断和异常事件来访问。
- 为容纳Intel保留的32个中断描述符，IDT的长度至少应有256Byte（32×8byte）。
- 与GDT相似，IDT也不是一个段。

64

IDTR和IDT的关系



- 中断门
- 陷阱门
- 任务门

65

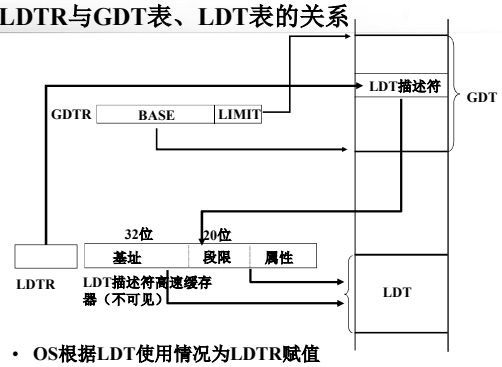
程序不可见寄存器

- LDTR（Local Descriptor Table Register），局部描述符表寄存器。
  - 包括16位段选择子，不可编程的64位段描述符寄存器（又称为段描述符Cache）。
  - 在64位段描述符寄存器中，有32位LDT的线性基地址、20位段限及12位的段属性。
  - 一个任务一个LDT。因此，系统中有多个LDT。为了访问局部描述符表，需在LDTR中装入一个选择子，用该选择子访问全局描述符表，并把局部描述符表的段基址、段限和段属性装入LDTR的Cache区。

		Base address	Limit	Access
TR				
LDTR				

66

LDTR与GDT表、LDT表的关系

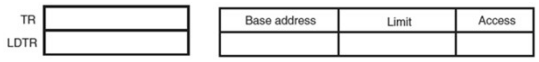


- OS根据LDT使用情况为LDTR赋值

67

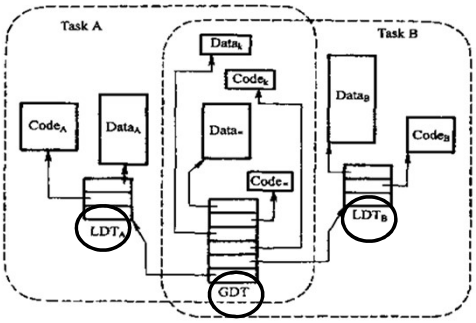
程序不可见寄存器

- TR（Task State Segment Register），任务状态段寄存器。
  - 包括16位段选择子，64位描述符寄存器（其中包括32位任务状态段的线性基地址，20位的段限及12位段属性）。
  - 每一个任务都有一个任务状态段TSS。
  - 现行的TSS由TR来标识。
  - 根据TR中的选择子，从GDT中索引TSS描述符，微处理器自动将TSS描述符装入TSS Cache中。



68

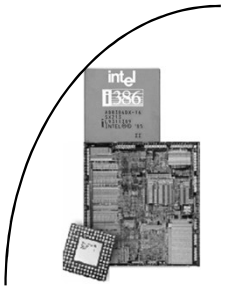
全局和局部地址空间



69

本章内容

- 微处理器的内部体系结构
- 实模式存储器寻址
- 保护模式存储器寻址简介
- 内存分页
- 平展模式内存



70

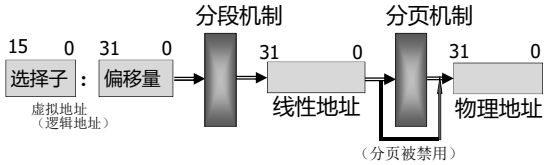
内存分页机制

- 内存分页机制
  - 80386及更高档微处理器。
  - 为任何线性地址分配任何物理存储器地址。
- 逻辑地址，即虚拟地址，这是应用程序设计人员进行编程时要用到的地址。
  - “段选择子：偏移量”：32位微机中，由一个16位的段选择子和32位偏移量两部分组成。
  - 段选择子放在段寄存器中；偏移量也称为偏移地址、有效地址。
- 线性地址：沟通逻辑地址与物理地址的桥梁。
  - 32位微处理器芯片内的分段部件将逻辑地址空间转换成32位的线性地址。
- 物理地址：指内存芯片阵列中每个阵列对应的唯64一的地址。32位地址线可直接寻址4GB内存单元。

71

二级虚拟－物理地址转换

- 以32位微机为例。



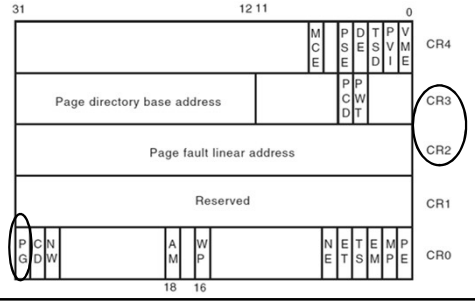
？什么情况下，线性地址和物理地址一致？

？什么情况下，偏移地址和线性地址一致？

72

分页有关的寄存器

- 微处理器中的控制寄存器的内容控制着分页部件。CR4是Pentium及其后才增加的。



73

分页有关的寄存器

- CR0最左边的一位PG：
  - PG=1，允许分页，通过分页部件将线性地址转换成物理地址；
  - PG=0，线性地址就是物理地址。
- CR2：页故障线性地址
  - 用于保存页故障线性地址（this linear address that caused a page fault），32位。
  - 操作系统中的页异常处理程序可以通过检查CR2的内容，得知32位的线性地址。

74

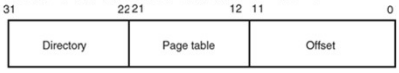
分页有关的寄存器

- CR3页目录基址寄存器。
  - 高20位存放页目录表的物理基地址。由于页目录表是按页对齐的（4K），因此只需保存高20位。
  - 低12位，有PCD和PWT等7位已定义。
  - PWT：Page write-through,指示是页面通写还是回写。
    - PWT=1,外部Cache对页目录进行通写；
    - PWT=0,进行回写。
  - PCD：Page Cache disable，页面Cache工作情况。
    - PCD=1，禁止片内的页面Cache；
    - PCD=0，允许片内的页面Cache。

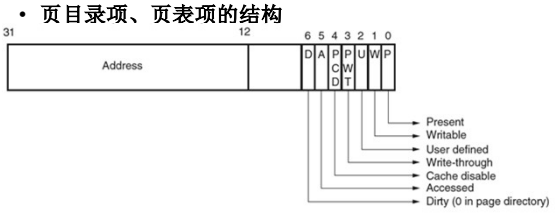
75

线性地址

- 线性地址的结构
  - 页目录索引、页表索引、位移量



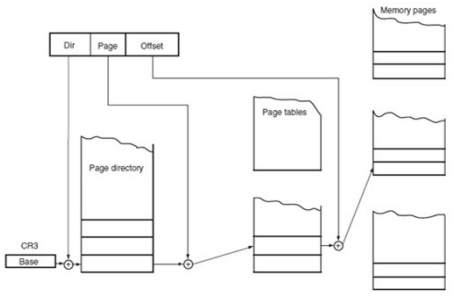
- 页目录项、页表项的结构



76

分页机制

- 80386~Core 2的分页机制



77

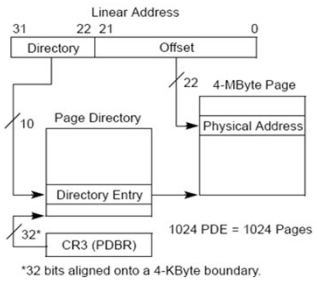
分页机制

- 页目录表的长度为4KB，最多可寻址1024个页表。
- 页表的长度为4KB，最多可寻址1024个页。
- 对于Pentium~Core2，可以按4KB、2MB、4MB长度分页。
  - 按2MB、4MB长度分页时，只有页目录表和内存分页，没有页表。

78

分页机制

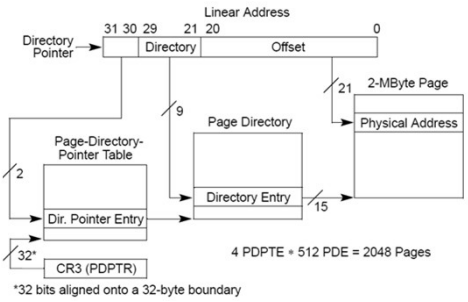
- **Linear Address Translation (4-MByte Pages)**



79

分页机制

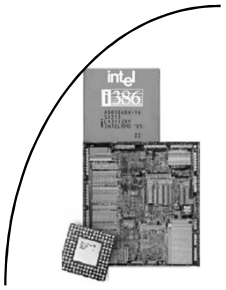
- **Linear Address Translation (2-MByte Pages)**



80

本章内容

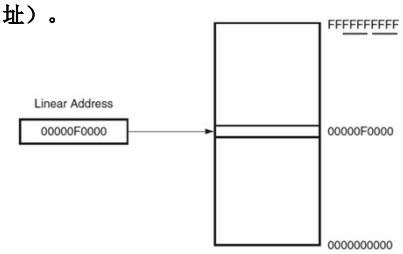
- 微处理器的内部体系结构
- 实模式存储器寻址
- 保护模式存储器寻址简介
- 内存分页
- 平展模式内存



81

平展模式内存模型

- 采用64位扩展的Pentium 4或Core 2的内存系统为平展模式内存系统。
  - 平展模式内存系统是不存在分段的系统（40位地址）。



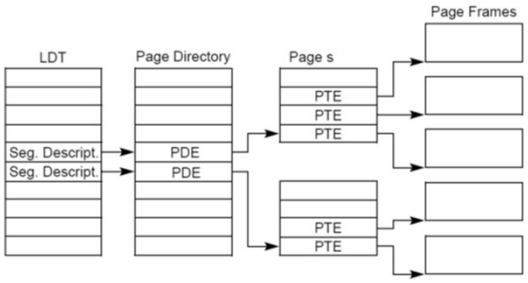
82

平展模式内存模型

- 平展模式不使用段寄存器进行寻址。
  - 平展模式不使用描述符中的基址和界限来选择段的内存地址。
  - 段是重叠的（Overlapping），都从地址0开始。
  - 64位模式下的偏移地址即为有效地址Effective address。
- CS段寄存器用来从只定义代码段访问权限的描述符表中选择描述符。
- 段寄存器仍然负责选择软件的优先级别。

83

分页和分段的组合（一种方式）



84

平展模式内存模型

- 在64位模式下，如果把地址设置为IA32兼容的（描述符中L位为0），那么地址是64位的，但是由于地址中只有40位被引出到地址线，任何超过40位的地址都会截断。
  - 使用偏移地址的指令只能使用32位偏移，即允许从当前指令开始的±2GB的地址范围。这种寻址方式被称为RIP相对寻址（在第3章解释）。
  - 直接传送指令（move immediate instruction）允许完全64位寻址和对任意平展模式内存地址的访问。
  - 其他指令不允许对4GB以上的地址空间进行访问，因为其偏移地址仍为32位。
- 如果Pentium工作在完全64位模式下（描述符中L位为1），其地址可以为64位或32位（在第4章解释）。

85

本章小结

- 微处理器的内部结构
  - 程序设计模型、多功能寄存器
- 实模式寻址
  - 段基址和偏移地址
- 保护模式寻址
  - 选择子、描述符、程序不可见的寄存器
- 内存分页机制

86

作业

- 习题13
- 习题19
- 习题21
- 习题27
- 习题37
- 习题43

87