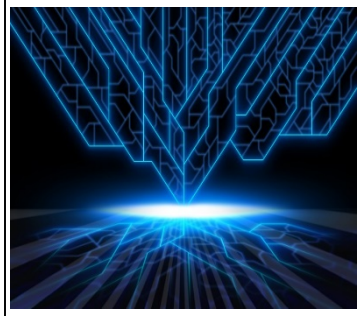
第2讲
Intel IA-32架构



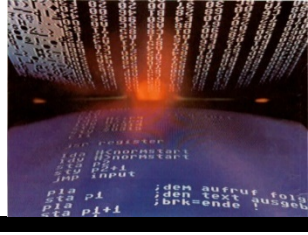
Intel Architecture, 32-bit

*www.**intel**.com/.../us/.../64-ia-**32**-**architecture**s-software-developer-vol-1-manual.pdf*

吴海军

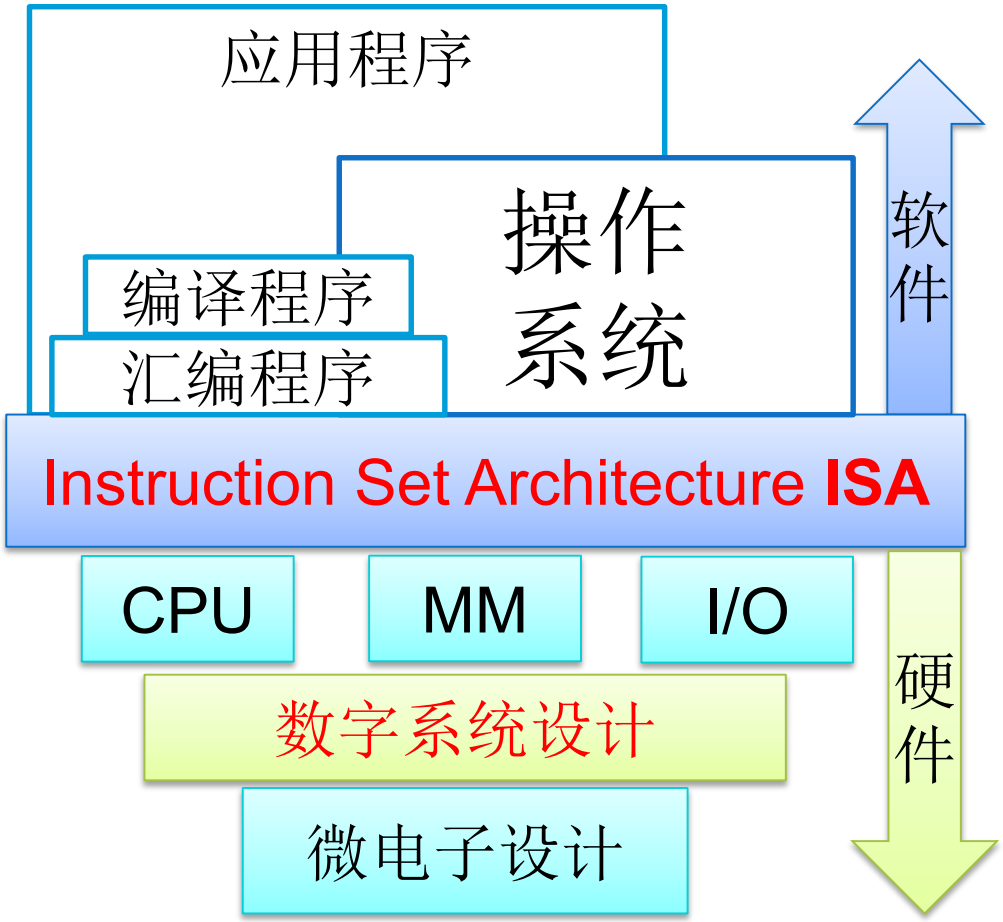
南京大学计算机科学与技术系

主要内容



- ISA
- Intel CPU
- 处理器工作模式
- 寄存器
- 存储器
- 指令格式

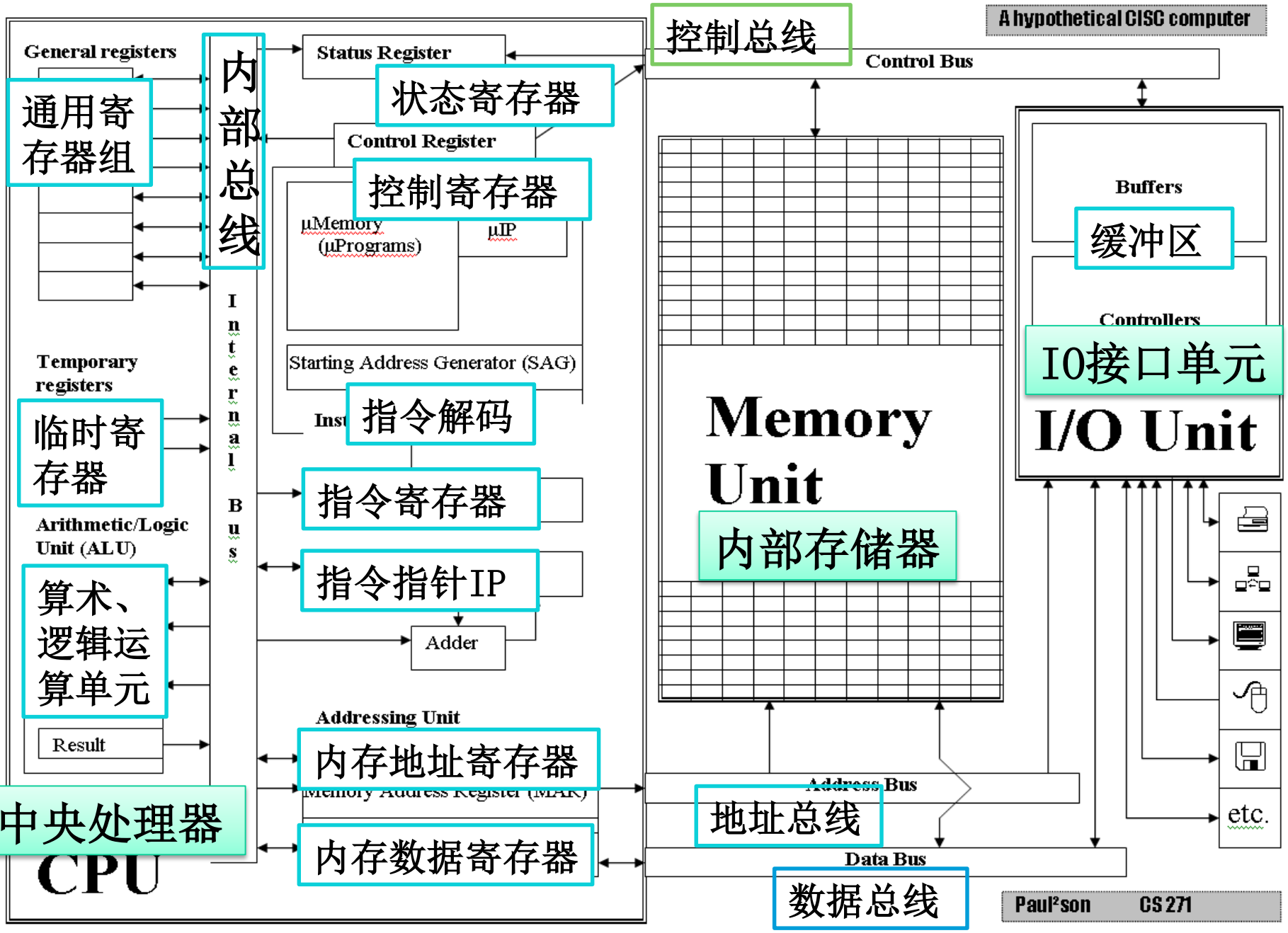
计算机系统抽象层次



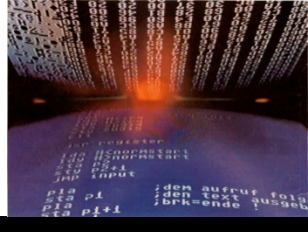
计算机系统由硬件和软件组成，
它们协同完成程序的执行。

ISA是最重要的层次！

ISA是对硬件的抽象，所有软件功能都建立在ISA之上！

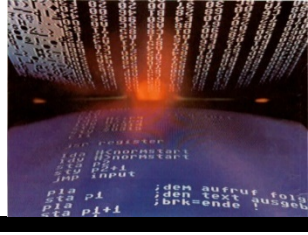

A hypothetical CISC computer
General registers
通用寄存器组
内部总线
Status Register
状态寄存器
控制总线
Control Bus
Control Register
控制寄存器
µMemory (µPrograms)
µIP
Buffers
缓冲区
Controllers
IO接口单元
Temporary registers
临时寄存器
Starting Address Generator (SAG)
Inst
指令解码
Memory Unit
I/O Unit
内部存储器
Internal Bus
指令寄存器
Arithmetic/Logic Unit (ALU)
指令指针IP
算术、逻辑运算单元
Adder
Result
Addressing Unit
内存地址寄存器
Memory Address Register (MAR)
Address Bus
地址总线
中央处理器
CPU
内存数据寄存器
Data Bus
数据总线
etc.
Paul²son CS 271

指令集体系架构ISA



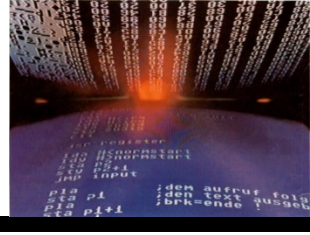
- 指令集体系架构（Instruction Set Architecture, ISA）是计算机体系结构中与程序设计有关的部分，包含了指令集、寄存器、寻址模式、基本数据类型、存储体系、中断、异常处理以及外部I/O等。
- 没有ISA，软件无法使用计算机硬件！
- 不同ISA规定的指令集不同，如：IA-32、IA-64、x86-64、 MIPS、ARM等。
- 计算机组成必须能够实现ISA规定的功能，如提供GPR、标志、运算电路等。

指令集体系架构ISA



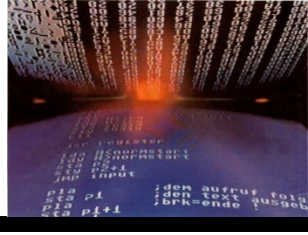
- ISA是一种规约，规定的主要功能有：
  - 可执行的**指令集合**，包括指令格式、操作种类以及每种操作对应的操作数的相应规定；
  - 指令可以接受的操作数的类型；
  - 操作数所能存放的寄存器组的结构，包括每个寄存器的名称、编号、长度和用途；
  - 操作数所能存放的存储空间的大小和编址方式；
  - 操作数在存储空间存放时按照大端还是小端方式存放；
  - 指令获取操作数的方式，即寻址方式；
  - 指令执行过程的控制方式，包括程序计数器、条件码定义等。

指令集体系架构ISA



- ISA**指令集**包含一系列的**指令码**以及由处理器执行的**基本命令**。
  - 指令码：处理器制造厂商在芯片内部**预先定义**的一串二进制代码，它规定了**处理器**的功能和任务，也称为**机器语言**。
  - 不同的制造厂商的指令码可能不同，不同系列的处理器的指令码也有可能不同。
  - 同一种ISA可以有不同的计算机组成，如乘法指令可用ALU或乘法器实现。
  - 指令集的分类：复杂指令集、精简指令集

复杂指令集计算CISC

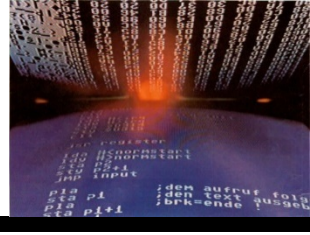


- Complex Instruction Set Computing是一种微处理器指令集架构，每个指令可执行若干低阶操作，如从内存读取、储存和计算操作，由单一指令完成。
- 特点：
  - 指令数目多而复杂；
  - 每条指令字长并不相等，电脑必须加以判读，并为此付出了性能的代价；
  - 对常用的简单指令会以硬件线路控制尽全力加速；
  - 不常用的复杂指令则由微码循序器“慢慢解码、慢慢跑”

当时的认识是硬件比编译器更容易设计。

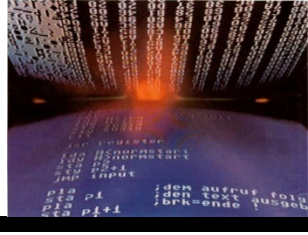
另一因素是缺乏大容量的内存，高消息密度的程序更实用。

精简指令集计算RISC



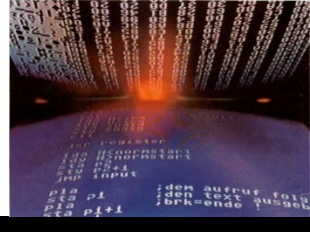
- Reduced Instruction Set Computing RISC：对指令数目和寻址方式都做了精简，使其实现更容易，指令并行执行程度更好，编译器的效率更高。
- 目前常见的精简指令集微处理器包括DEC Alpha、ARM、MIPS、PA-RISC、Power Architecture和SPARC等。
- 特点：指令数目少，每条指令都采用标准字长、执行时间短、中央处理器的实现细节对于机器级程序是可见的等等。
- RISC与CISC在竞争的过程中相互学习，现在的RISC指令集也达到数百条，运行周期也不再固定。

Intel X86 处理器



- Intel公司的CPU占领主流地位
- 第一款微处理器:是1971年11月Intel公司推出的4004微处理器。
- 循序渐进的设计
  - 开始于1978年的8086单芯片
  - 随着时间推移增加很多新的特性
  - 尽管有些特性已经过时，但仍然被兼容着
- 采用**复杂指令集系统**
  - 指令集由具有不同格式的多种类型指令构成
  - 只有小部分子集被用到Linux程序中

Intel X86处理器

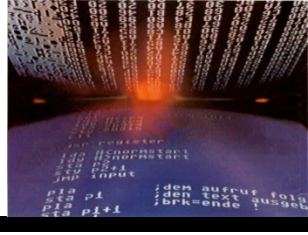


Name	*Date*	*Transistors*	*MHz*
8086	1978	29K	5-10
386	1985	275K	16-33
Pentium4F	2004	125M	2800-3800
Core i7	2008	731M	2667-3333

- 第1个16位处理器. 构成了IBM PC&DOS系统的基础
- 1MB地址寻址空间. DOS系统只留给用户640K
- 第1个32位处理器，被称为IA-32, x86-32，i386
- 加了平面寻址方式
- 可运行Unix，32位的Linux/gcc 没有使用后续推出的新指令
- 64位处理器, 被称为x86-64

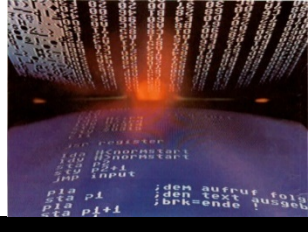
通常按照处理器芯片支持的指令码的数量和类型进行分类。

64位架构



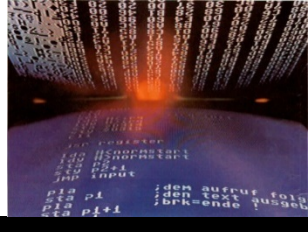
- AMD64 ：1999年由AMD设计，是64位版本的x86指令集，兼容于16位及32位的x86架构。AMD64架构在IA-32上新增了64位寄存器，并兼容早期的16位和32位软件，可使现有以x86为对象的编译器容易转为AMD64版本。AMD对这种CPU架构的原始称呼——“x86-64”
- IA-64：Intel最早推出的64位架构是基于超长指令字VLIW技术的IA-64体系结构，**安腾**和安腾2分别在2000年和2002年问世。是一款独立的架构，可通过模拟来运行IA-32的指令，但指令在运行前需经转换，才能在IA-64上运行，导致其速度变慢。
- Intel64：Intel在2004年推出IA32-EM64T，它支持x86-64指令集。英特尔称之为“Intel 64” 、“IA-32e”及“EM64T”。一般称为：x86-64、x86_64、x64等。

计算机发展的持续动力



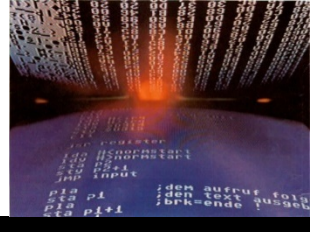
- 一方面希望计算机做得更多
- 一方面希望计算机运行得更快
- 并发：同时具有多个活动的系统
- 并行：用并发使一个系统运行得更快
- 线程级并发
  - 单处理器系统
  - 多处理器系统
- 指令级并发：流水线
- 单指令多数据并行SIMD

IA-32 处理器基本工作模式



- 实地址模式Real Mode
  - 与8086/8088兼容，可以处理32位数据
  - 1MB内存空间，分段管理，20位地址：(CS)<<4+(IP)
  - MS-DOS运行在此模式下，PC机开机或复位时进入该模式
  - 对内存和程序甚至操作系统没有任何保护能力
- 保护模式Protected Mode
  - 支持多任务操作，并保护每个任务的数据和程序
  - 存储器采用虚拟地址空间、线性地址空间和物理地址空间三种方式来描述，具有存储保护功能
  - 虚拟地址空间64TB（2^{46}B），段基址（14位）+段内偏移量（32位）
  - 内存4级管理，可以使用分页或分段技术管理内存
  - Windows、Linux操作系统均运行在该模式下

IA-32处理器基本工作模式

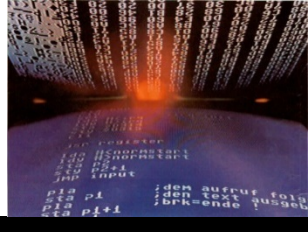


- 虚拟8086模式（Virtual 86 mode）
  - 在保护模式下可以同时模拟多个8086处理器的工作

项目	实地址模式	虚拟8086模式
内存管理	分段管理	既分段又分页
存储空间	1MB	每个8086程序任务寻址1MB，总寻址空间4GB
多任务	不支持	支持，虚拟8086模式是IA-32保护模式中多任务的一个任务

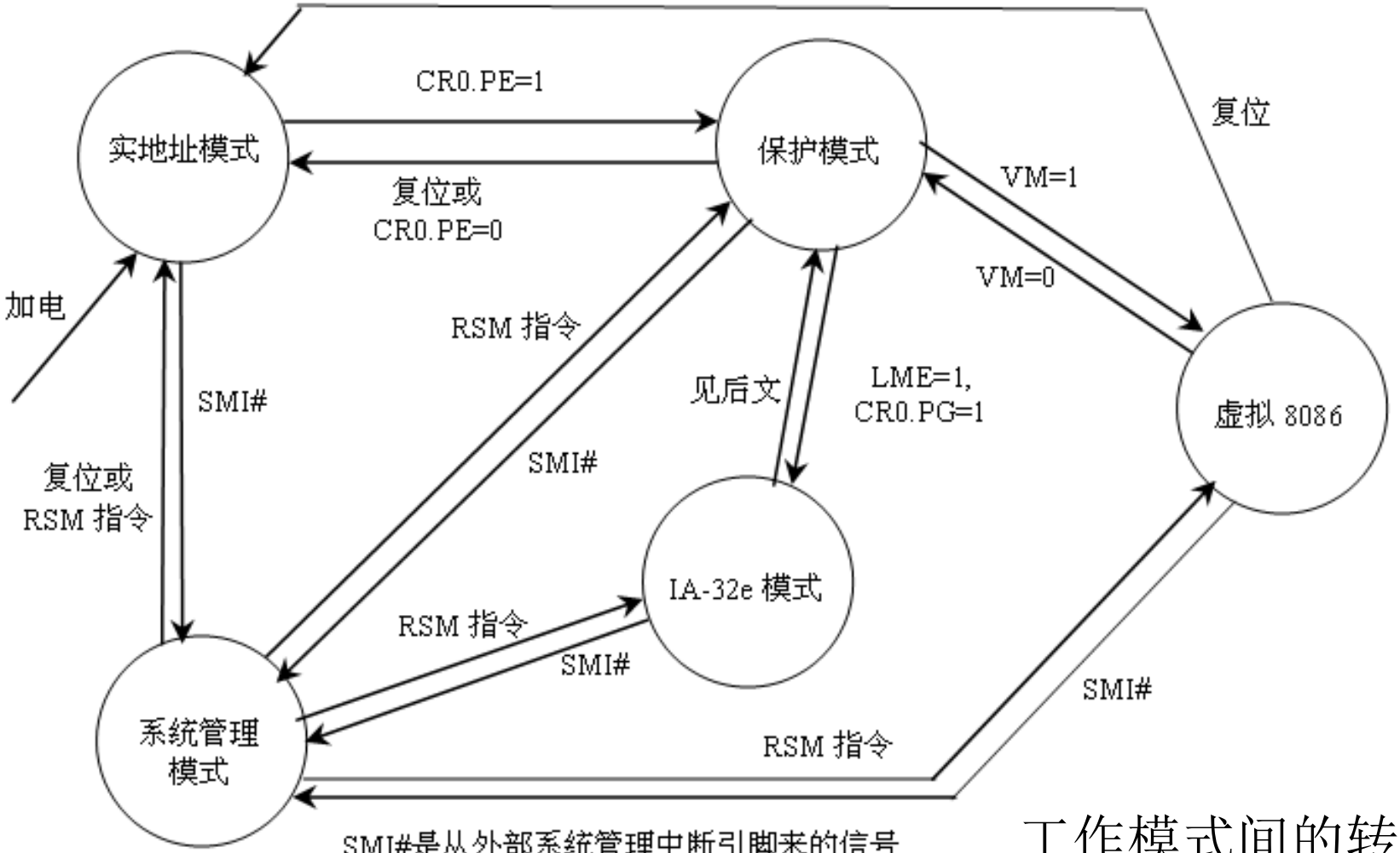
- 系统管理模式System Management mode SMM：
  - 只能通过系统管理中断指令SMI进入，并只能通过执行RSM指令退出。

IA-32处理器基本工作模式



- IA-32e模式（支持Intel EM64T的处理器）
  - 兼容模式
    类似于32位保护模式，传统的16位或32位程序无需重新编译就可以运行在64位操作系统中。在64位模式和保护模式下支持的所有特权级别在兼容模式同样支持。
  - 64位模式
    允许64位操作系统运行存取64位线性地址空间的应用程序，程序可访问的线性地址空间达到2^{64}字节，物理地址空间增加到2^{40}字节。通用寄存器的宽度增加到64位，并新增了8个通用寄存器和8个SIMD寄存器。

IA-32 CPU基本工作模式

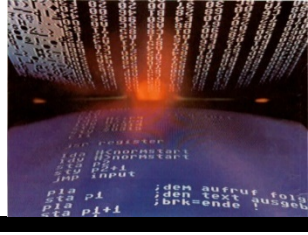


SMI#是从外部系统管理中断引脚来的信号

RSM 是从系统管理中断服务程序中返回的指令

工作模式间的转换

IA-32处理器中的寄存器

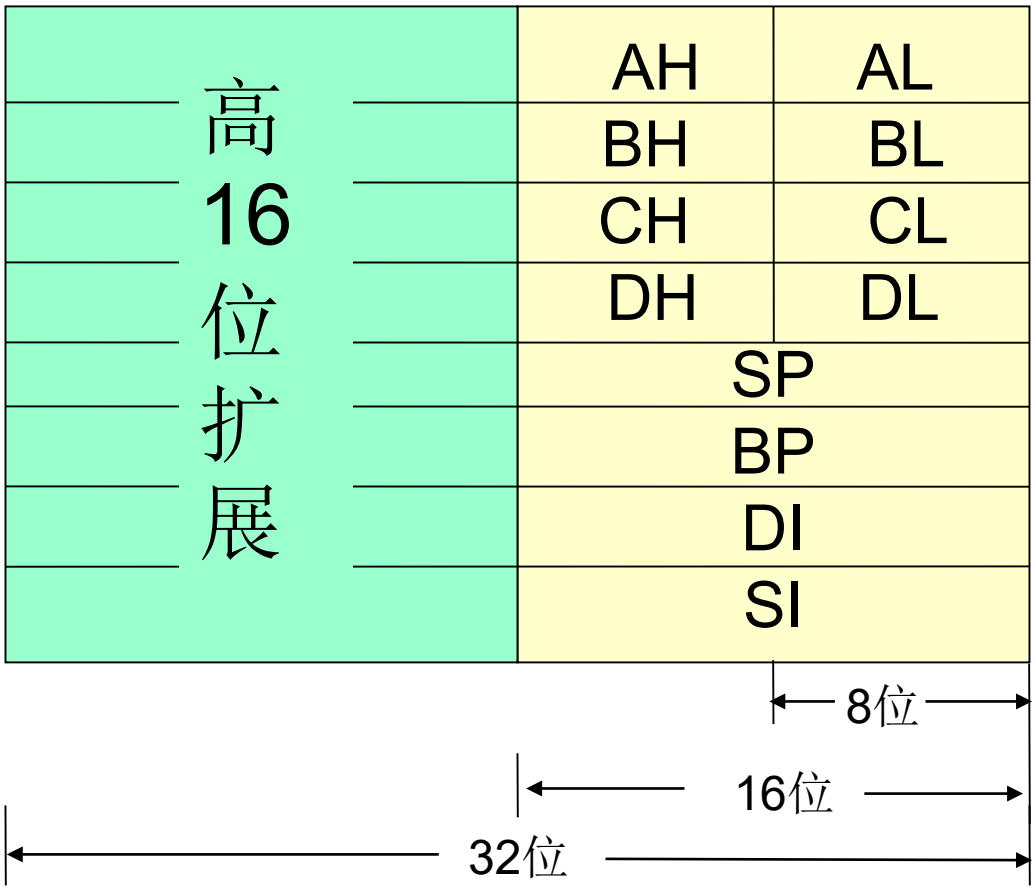


存储指令、数据、地址、状态等信息。

- 基本寄存器
  - 通用寄存器
    - 数据寄存器
    - 地址指针寄存器
    - 变址寄存器
  - 段寄存器
  - 指令指针寄存器
  - 标志寄存器
- 系统级寄存器
  - 控制寄存器
  - 系统地址寄存器
- 调试与测试寄存器
- 浮点寄存器

只有少量寄存器：
1、价格高
2、可以减少指令字中寄存器编码长度。

通用寄存器

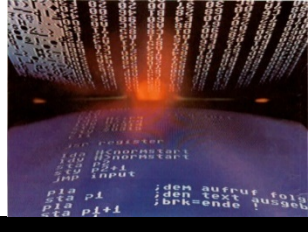


16位名称	32位名称	
AX	EAX	累加器
BX	EBX	基址变址
CX	ECX	计数
DX	EDX	数据
SP	ESP	堆栈指针
BP	EBP	基址指针
DI	EDI	目的变址
SI	ESI	源变址

可以32位、16位或8位形式访问，例如EAX可使用16位的AX，也可以使用8位的AH、AL来访问。

体现了微处理器的兼容性。

通用寄存器



- 8个32位通用寄存器，用于保存逻辑和算术运算的操作数、地址运算和内存指针：
  - EAX：累加器 存放操作数和结果，乘除运算、I/O指令中特指
  - EBX：基址寄存器 查表转换和间接寻址时存放基址
  - ECX：计数寄存器 串操作和循环中做计数
  - EDX：数据寄存器 乘除运算、I/O指令中特指
  - ESI：源变址寄存器
  - EDI：目的变址寄存器，目标地址指针
  - EBP：基址指针寄存器，存放栈段基地址
  - ESP：堆栈指针寄存器，存放栈顶地址

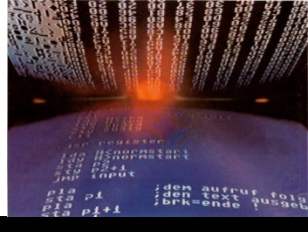
存放地址的偏移量，也可存放操作数；只能以32位或16位为单位访问。

段地址寄存器

- 有6个16位段地址寄存器。 用于存储器寻址，存放段的开始地址。
  - CS 代码段寄存器
  - SS 堆栈段寄存器
  - DS 数据段寄存器
  - ES 数据附加段寄存器
  - FS、GS 数据附加段寄存器

- 现代的操作系统和应用程序使用（不分段）内存模型，所有段地址的值相同。
- 在64位模式下，FS, GS无效，CS, DS, ES, SS均指向基地址为0的“段”。

指令指针寄存器EIP



- EIP寄存器也被称为“程序计数器PC”。 它存放的是在当前代码段中要执行的下一条指令地址的**偏移量**。
- 它可以是指令地址顺序产生，也可能是由跳转、调用、返回等指令产生。
- EIP的不能由软件直接访问；它是由控制转移指令、中断产生和异常隐含控制。
- EIP寄存器可以间接通过修改程序堆栈上的返回指令指针的值，并执行返回指令（RET或IRET）。
- 读取EIP的唯一方法是执行CALL指令，然后读取从程序堆栈中的返回指令指针的值。

标志寄存器EFLAGS

- EFLAGS 标志寄存器（ 程序状态字寄存器PSW ）：记录系统运行中的各种状态 和信息。由各种标志位构成，反映运算后的结果特征，将影响某些指令（如条件转移指令）的执行。
- 8086/8088程序状态寄存器（标志寄存器）

				OF	DF	IF	TF	SF	ZF		AF		PF		CF

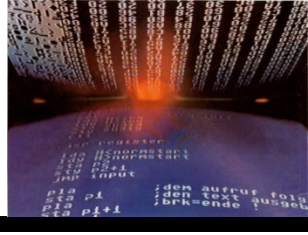
符号	名称	值为“1”的条件
CF	进位标志	加/减法时产生进位/借位
OF	溢出标志	运算结果超出有符号整数能表示的范围
ZF	零标志	运算结果为0时
SF	符号标志	运算结果的最高位为“1”时
AF	辅助进位标志	运算时半字节（b3）产生进位/借位
PF	奇偶标志	操作结果低8位为“1”的位数为偶数时
DF	方向标志	串操作中地址指针向低地址方向移动
IF	中断允许标志	允许CPU响应可屏蔽中断请求时
TF	跟踪标志	CPU处于单步执行的工作方式

标志寄存器EFLAGS

- 以下的几个4位十六进制数相加，会使得8088状态寄存器的以下几位为什么值？

	8000H + 8000H = 0000H	C000H + C000H = 8000H	4008H + 4008H = 8010H	0808H + C000H = C808H
CF	1	1	0	0
PF	1	1	0	0
AF	0	0	1	0
ZF	1	0	0	0
SF	0	1	1	1
OF	1	0	1	0

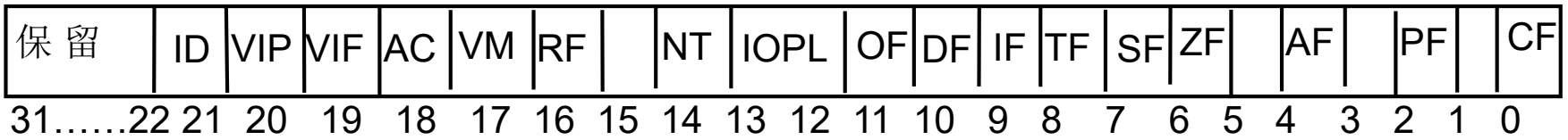
标志寄存器EFLAGS



- 注意:

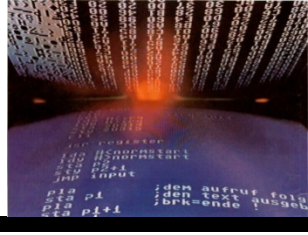
1. 进位标志CF是表示无符号数是否超出范围，但运算结果仍然正确；
2. 溢出标志表示的是有符号数运算结果是否超出范围，超出范围则运算结果已经不正确；
3. 处理器对两个操作数进行运算的时候是按照无符号数求得结果，并相应设置CF，根据是否超出有符号数的范围设置OF；
4. 对于程序员，如果做无符号运算，应该关心CF，做有符号运算应该关心OF。

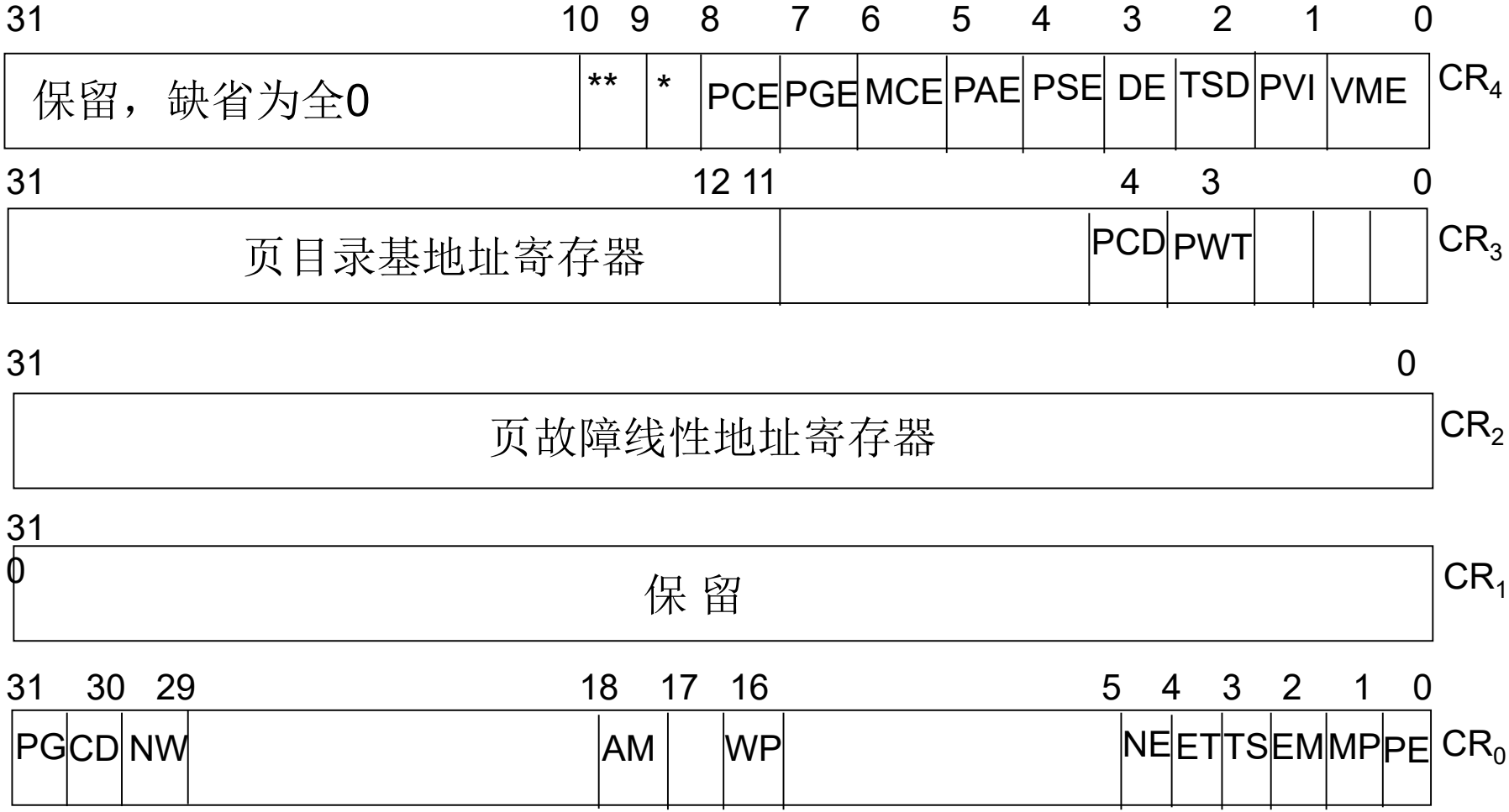
标志寄存器EFLAGS



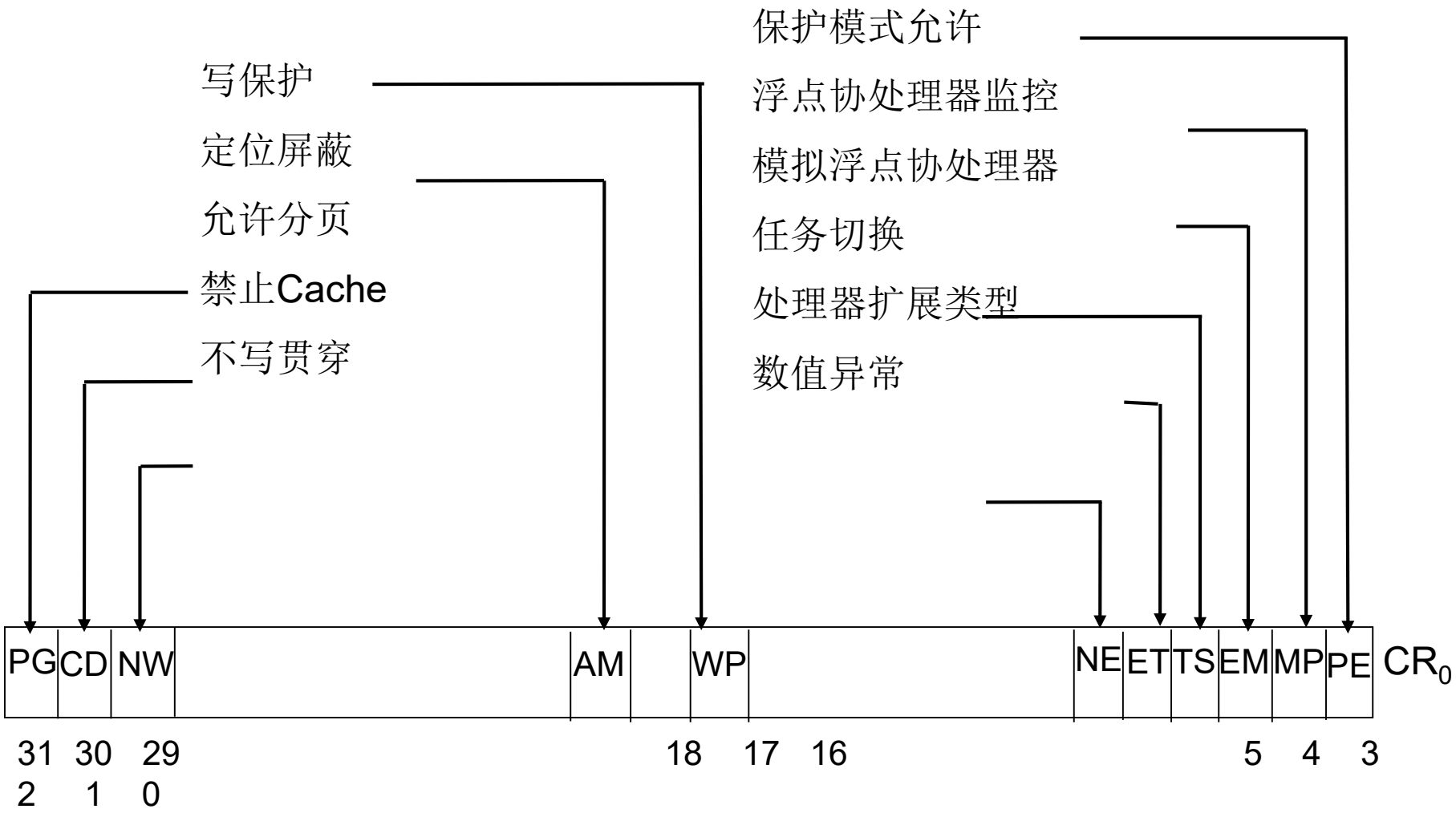
符号	名称	值为“1”的条件
IOPL	IO特权位	其值表示该任务使用的I/O操作的特权级 00为最高的核心级，11是最低的用户级
NT	嵌套标志	当前任务嵌套在另一个任务中时
RF	恢复标志	DBUG调试时忽略下一条指令遇到的断点
VM	虚拟8086模式	从保护模式进入到虚拟8086模式
AC	对齐检查	当有数据访问出现对齐故障的时候
VIF	虚拟中断位	允许V86扩展或允许保护模式虚拟中断
VIP	虚拟中断挂起位	虚拟中断被挂起
ID	标识位	对它的读写表明处理器支持CPUID

控制寄存器

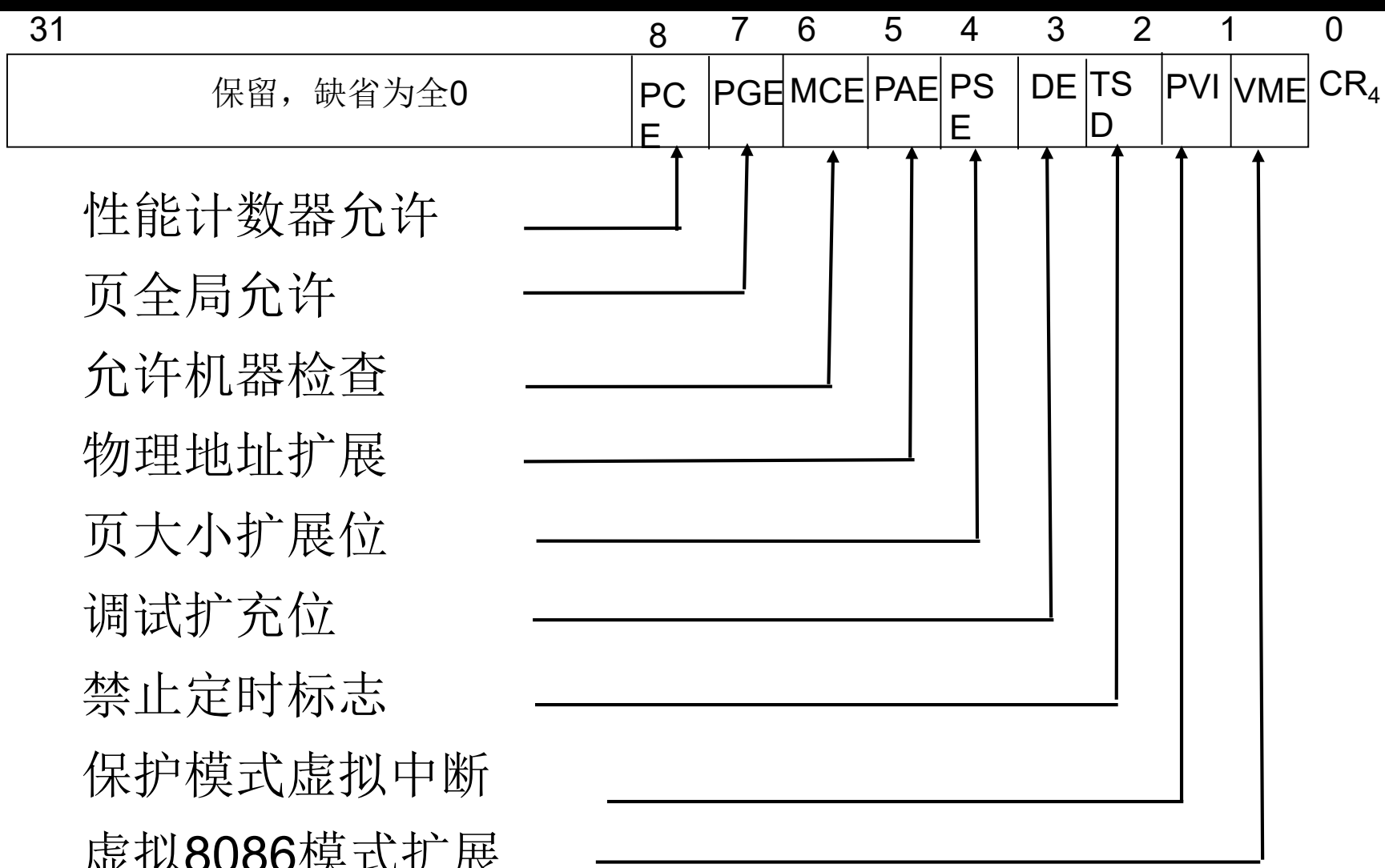



31
10 9 8 7 6 5 4 3 2 1 0
保留，缺省为全0
**
*
PCE
PGE
MCE
PAE
PSE
DE
TSD
PVI
VME
CR4
31
12 11
4 3 0
页目录基地址寄存器
PCD
PWT
CR3
31
0
页故障线性地址寄存器
CR2
31
0
保 留
CR1
31 30 29
18 17 16
5 4 3 2 1 0
PG
CD
NW
AM
WP
NE
ET
TS
EM
MP
PE
CR0

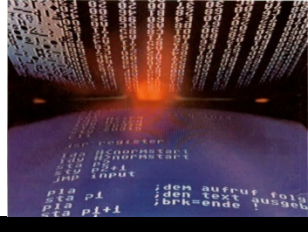
控制寄存器


保护模式允许
写保护
浮点协处理器监控
定位屏蔽
模拟浮点协处理器
允许分页
任务切换
禁止Cache
处理器扩展类型
不写贯穿
数值异常
PG
CD
NW
AM
WP
NE
ET
TS
EM
MP
PE
CR0
31
30
29
18
17
16
5
4
3
2
1
0

控制寄存器


31
8
7
6
5
4
3
2
1
0
保留，缺省为全0
PCE
PGE
MCE
PAE
PSE
DE
TSD
PVI
VME
CR4
性能计数器允许
页全局允许
允许机器检查
物理地址扩展
页大小扩展位
调试扩充位
禁止定时标志
保护模式虚拟中断
虚拟8086模式扩展

系统地址寄存器



- GDTR — 48位的全局描述符表寄存器

全局描述符表32位线性地址	16位界限值

- IDTR — 48位的中断描述符表寄存器

中断描述符表32位线性地址	16位界限值

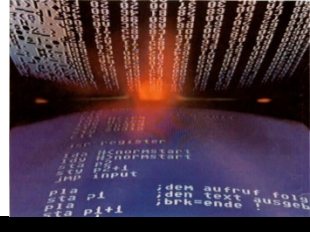
- TR — 16位的任务状态段寄存器

TSS的16位选择字

- LDTR — 16位的局部描述符选择字寄存器

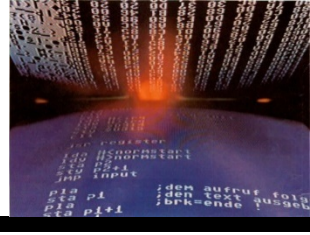
LDT的16位选择字

IA-32存储器



- IA-32 的存储器**以字节为单位**编址，即一个字节数据占一个存储单元。
- 提供四种数据类型：
  - 字节Byte（8位）、字word（16位）、双字double-word（32位）、四倍长字quad-word（64位）
- 以字、双字、四字为单位存储数据时，分别占相邻2个、4个、8个连续字节单元，由其最低字节的地址来表示。
- 字、双字、四字的地址一般采用边界对齐，即它们地址分别是偶数地址，4的倍数和8的倍数。
  - 如果边界不对齐，CPU会采用两个总线周期来完成操作，或者直接报错。如果边界对齐，则只用一个总线周期完成操作。

IA-32存储器

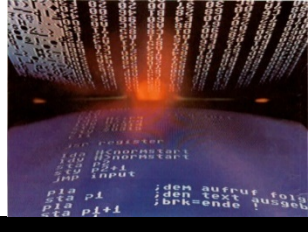


- IA-32处理器使用“小端模式Little Endian”的字节顺序
  - 高位字存放在高地址区，低位字存放在低地址区
  - 数据的最低有效字节存放在数值最小的地址中。

0000: 0H	1H	2H	3H	4H	5H	6H	7H
12H	34H	56H	78H	9AH	BCH	DEH	FFH

0000H地址上，
表示字节数据：12H
表示字数据：3412H
表示双字数据：78563412H
表示四字数据：FFDEBC9A78563412H

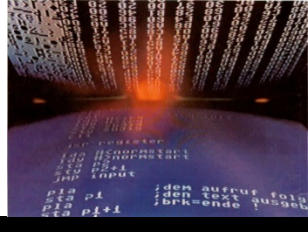
IA-32存储器寻址方式



- 寻址方式：根据指令给定信息得到操作数或操作数地址。
- 实模式存储器寻址
  - 解决16位寄存器表示20位地址的问题
  - 存储器地址的分段，段是最大长度为64KB的连续的内存块
  - 物理地址 每个存储单元的20位实际地址，有唯一性，访问主存时必须用物理地址
  - 逻辑地址 每个存储单元的地址用两部分表示：
    - 段基址（段首址的高16位），左移四位
    - 偏移量 （段内某单元相对段首址的地址差，也称为有效地址EA）

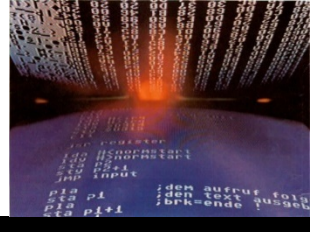
物理地址=段基址×16+偏移量

IA-32存储器寻址方式



- 保护模式存储器寻址
  - 应用程序使用的是**逻辑地址**
    - 16位段描述符索引：32位偏移地址
    - 16位的段描述符索引是由段寄存器提供，在保护模式下它不再提供段基址
  - 在全局描述符表寄存器GDTP或局部描述符表寄存器LDTR协助下，可以根据段描述符索引确定**段基址**
  - 确定段基址之后，可以将逻辑地址转换成**线性地址**
    - 线性地址=32位段基址+32位偏移地址
  - 在处理器页表机制下，再将线性地址转变成**物理地址**
    - 物理地址=页表转换(线性地址)
  - 逻辑地址==》线性地址==》物理地址

IA-32存储器寻址方式



- 保护模式存储器寻址
  - 实模式能使用的方式，保护模式都能使用
  - 所有的通用寄存器都能够用来寻址，不再限于bx、bp、si、di
  - 还支持有新的寻址方式，例如：
    - mov ebx，[ecx + eax * 2 + 78H]
  - 完整的形式是：
    - base + index * 2^{scale} + displacement
    - base和index为32位通用寄存器，scale的值是0至3，displacement是偏移量

IA-32存储器寻址方式

寻址方式	说明
立即寻址	指令直接给出操作数
寄存器寻址	指定的寄存器R的内容为操作数
位移	LA=(SR)+A
基址寻址	LA=(SR)+(B)
基址加位移	LA=(SR)+(B)+A
比例变址加位移	LA=(SR)+(I)×S + A
基址加变址加位移	LA=(SR)+(B)+(I)+A
基址加比例变址加位移	LA=(SR)+(B)+(I)×S + A
相对寻址	LA=(PC)+A

存储器操作数

跳转目标指令地址

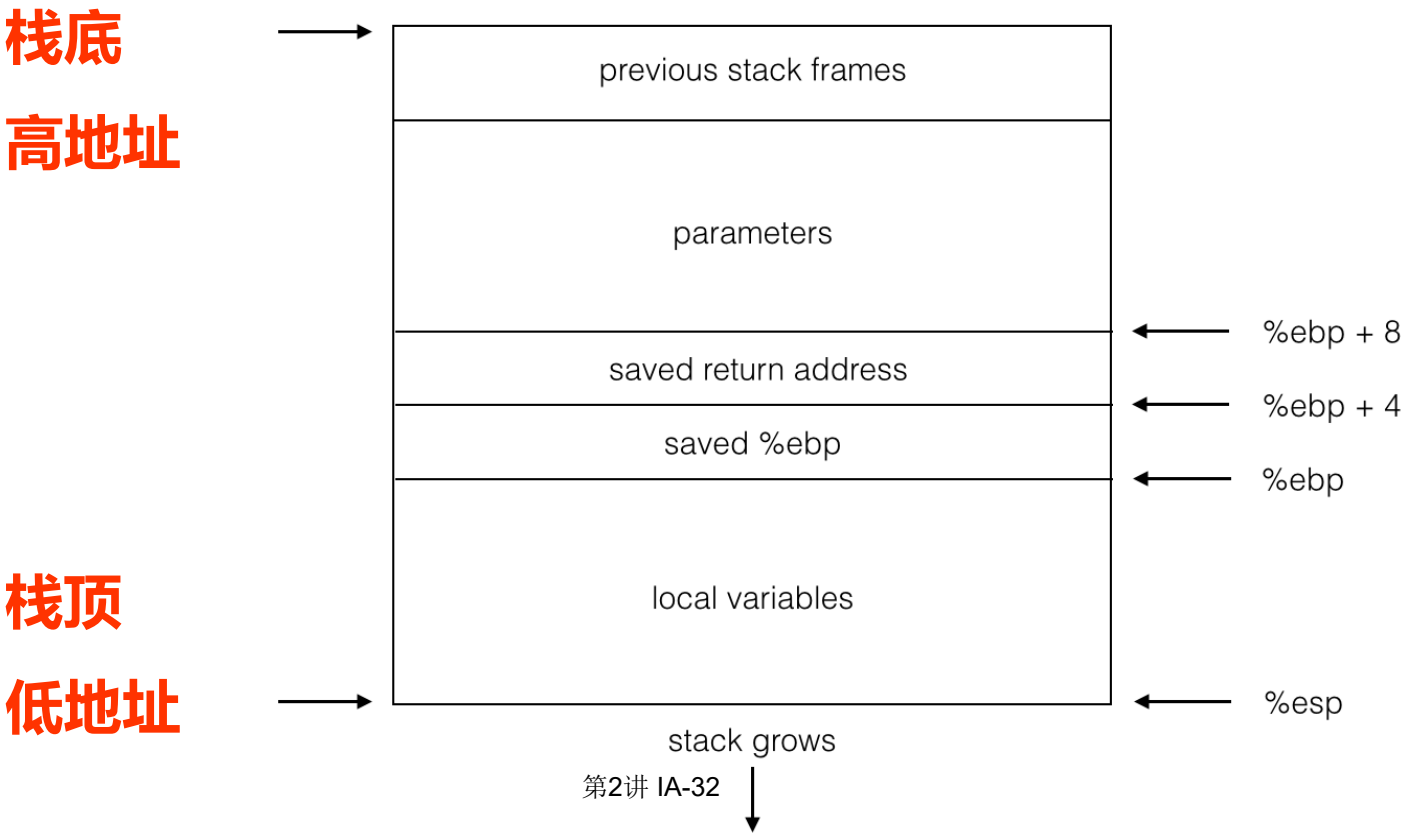
注：LA:线性地址　(X):X的内容　SR:段寄存器　PC:程序计数器　R:寄存器
A:指令中给定地址段的位移量　B:基址寄存器　I:变址寄存器　S:比例系数

保护模式下的寻址方式

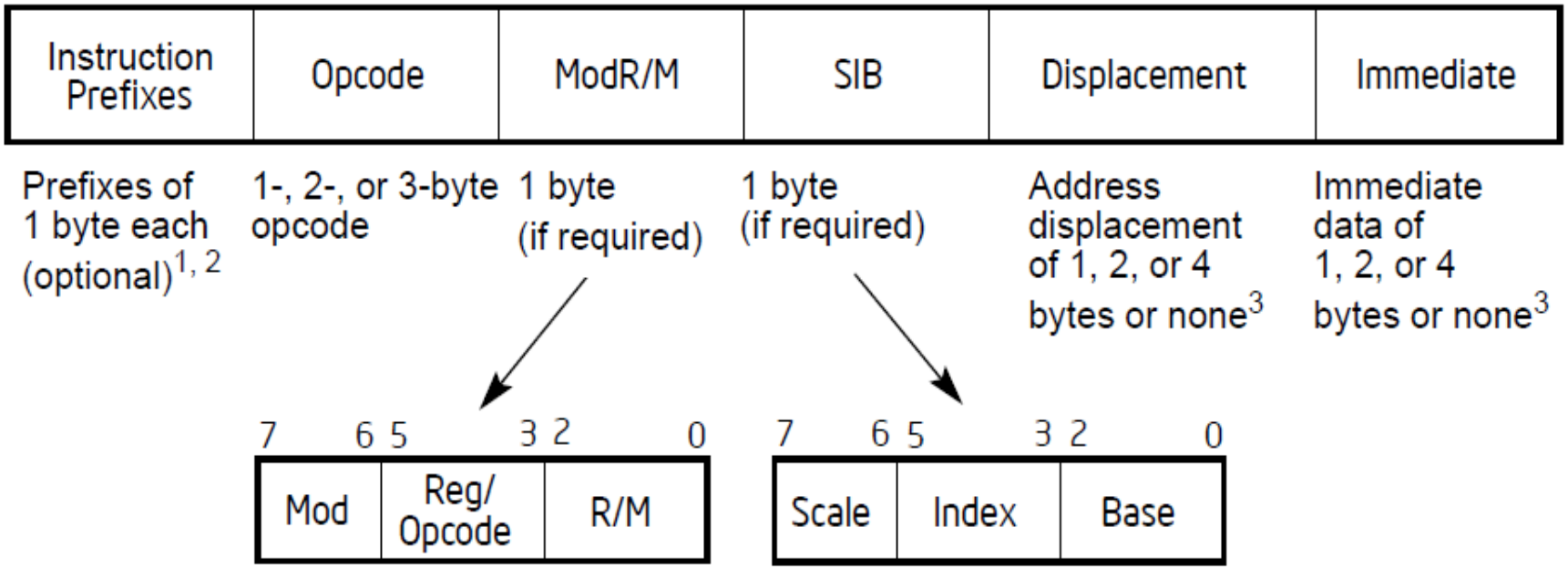
线性地址=段地址+偏移量

IA-32堆栈结构

- 堆栈是内存中以字长为单元的“先入后出”、最大空间为64KB的存储区域，**栈底地址大于栈顶地址。**
- 栈用于嵌套、过程调用等。用了保存参数、返回地址、局部变量、寄存器等。

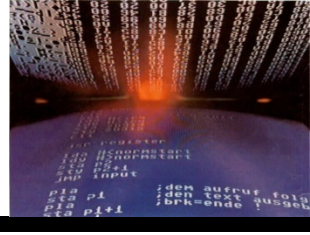


IA-32指令码格式



1. The REX prefix is optional, but if used must be immediately before the opcode; see Section 2.2.1, “REX Prefixes” for additional information.
2. For VEX encoding information, see Section 2.3, “Intel® Advanced Vector Extensions (Intel® AVX)”.
3. Some rare instructions can take an 8B immediate or 8B displacement.

IA-32指令码格式



- 处理器芯片按照指令码完成相应的操作。
- 指令码包含1个或多个处理器需要处理的信息。
- IA-32指令码格式，分四个部分：

名称	指令前缀	操作码	修饰符			数据元素
			ModR/M	SIB	移位	
字节数	0~4	1~3	0~1	0~1	0~4	0~4

- 指令前缀：包含0到4个修改操作码行为的1字节前缀
  - 锁定前缀和重复前缀：锁定前缀表示指令将独占共享内存区域
  - 段覆盖前缀和分支提示前缀：段覆盖前缀覆盖定义了的段寄存器的值
  - 操作数长度覆盖前缀：通知处理器，程序将切换16位和32位操作数
  - 地址长度覆盖前缀：通知处理器，程序将切换16位和32位内存地址

IA-32指令码格式

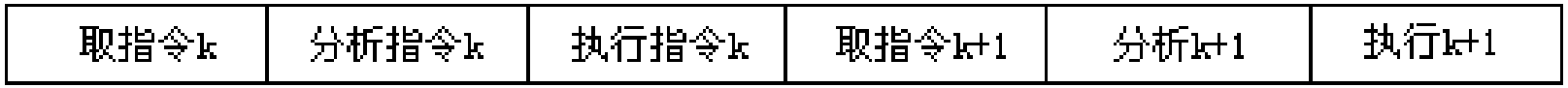
- 操作码：定义了处理器执行的基本功能和任务，是IA-32指令码格式中唯一必须存在的部分，1到3个字节长。比如0F A2定义了IA-32 CPUID指令
- 修饰符：定义了执行的功能中涉及的寄存器和内存位置，包含3个独立的部分
  - ModR/M字节：寻址方式说明符，1个字节
    - 3个字段的信息构成：Mod(2bit) reg/opcode(3bit) r/m(3bit)
    - Mod字段和r/m字段一起使用，定义指令用使用的寄存器或者寻址模式
    - Reg/opcode字段用于允许使用更多的3位进一步定义操作码功能或者用于定义寄存器。

IA-32指令码格式

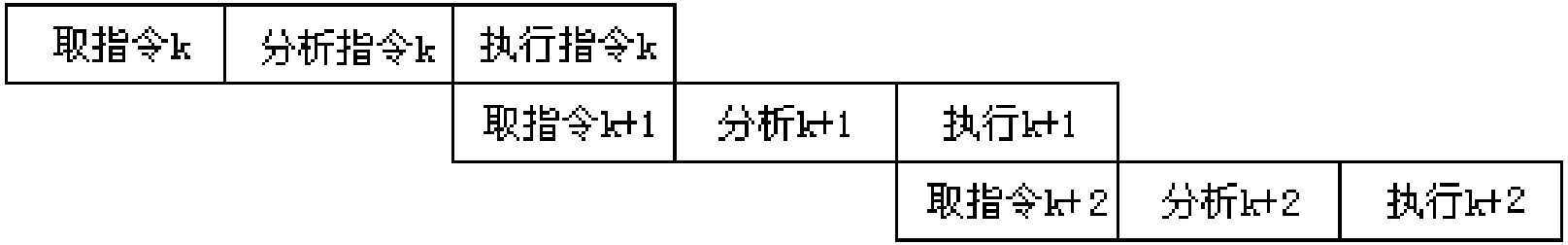
- SIB字节：比例-索引-基地址，1个字节
  - 由3个字段组成
  - SS比例字段：指定操作的比例因子
  - Index索引字段：指定内存访问中用作索引寄存器的寄存器
  - Base基址字段：指定用作内存访问的基址寄存器的寄存器

- 移位字节：1、2或4个的地址移位字节，4个字节，用来指定对于ModR/M和SIB字节中定义的内存位置的偏移量。
- 数据元素：指令中包含的数据，称为立即数。可以包含1、2或4字节的信息。

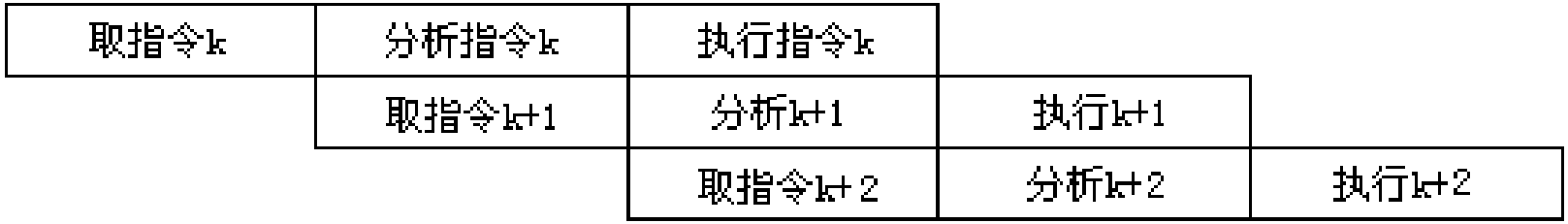
IA-32指令的执行方式



(a)顺序执行方式

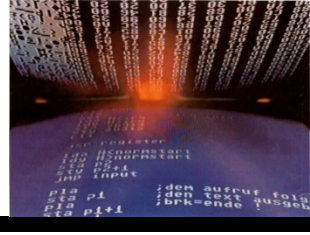


(b) 一次重叠执行方式



(c)二次重叠执行方式

IA-32指令码格式



- 指令码示例：C7　45　FC　01　00 00 00
  - C7：操作码，定义把值传送到内存指定位置（MOV）
  - 45 FC：修饰符，定义内存位置是从EBP寄存器（45）中值指向的内存位置开始的4个字节（FC）
  - 01 00 00 00：定义传送的数值，立即数1。
- 指令码由0、1序列组成，难以记忆
- 使用助记符表示—汇编语言
  - 操作码和汇编指令一一对应，都是机器级指令
  - 汇编指令是机器指令的符号表示