



# Intel 指令集体系结构简介



上海交通大学  
SHANGHAI JIAO TONG UNIVERSITY

# Intel X86 处理器

- 桌面机/服务器市场中的主流处理器 
- ISA的演化设计
  - 向后兼容：从第一代8086处理器开始，全部兼容
  - 每一代都增加新的特性 
- 复杂指令集计算机（Complex instruction set computer：CISC）
  - 指令条数多
  - 不同指令，格式可能不同，长度也不相同

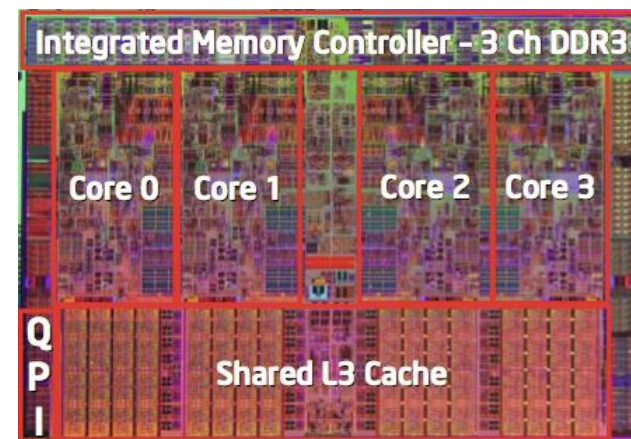
# 指令系统的两个发展方向

- CISC(复杂指令集计算机)---指令数量多，指令功能，复杂的计算机。
- RISC(精简指令集计算机)---指令数量少，指令功能单一的计算机。

特 征	CISC			RISC	
	IBM 370	VAX 11	Intel 80486	SPARC	MIPS R4000
开发年份	1973	1978	1989	1987	1991
指令数量/条	208	303	235	69	94
指令长度/B	2 ~ 6	2 ~ 5	1 ~ 11	4	4
寻址方式	4	22	11	2	2
通用寄存器数/个	16	16	8	40~ 520	32
控制存储器大小/Kb	420	480	246	—	—
Cache大小/KB	64	64	8	32	128

# Intel x86 体系结构 (16->32-64位) 演变过程:

名字	时间	晶体管数	主频MHz
<ul style="list-style-type: none"> <li>8086</li> <li> <ul style="list-style-type: none"> <li>第一个16位Intel 处理器, 对外有16根数据线和20根地址线, 1MB (<math>2^{20}</math>) 寻址空间</li> </ul> </li> </ul>	1978	29K	5-10
<ul style="list-style-type: none"> <li>386</li> <li> <ul style="list-style-type: none"> <li>第一个32 位Intel 处理器, 指令集体系结构简称为 IA32, 可以访问4GB的存储器</li> </ul> </li> </ul>	1985	275K	16-33
<ul style="list-style-type: none"> <li>Pentium 4E</li> <li> <ul style="list-style-type: none"> <li>第一个64 位Intel x86 处理器, (x86-64) 可以访问高于4GB的存储器</li> </ul> </li> </ul>	2004	125M	2800-3800
<ul style="list-style-type: none"> <li>Core 2</li> <li> <ul style="list-style-type: none"> <li>第一个 多核 Intel 处理器</li> </ul> </li> </ul>	2006	291M	1060-3500
<ul style="list-style-type: none"> <li>Core i7</li> <li> <ul style="list-style-type: none"> <li>四核Intel 处理器</li> </ul> </li> </ul>	2008	731M	1700-3900





# 从16位到64位：x86体系结构的演变



寄存器模型

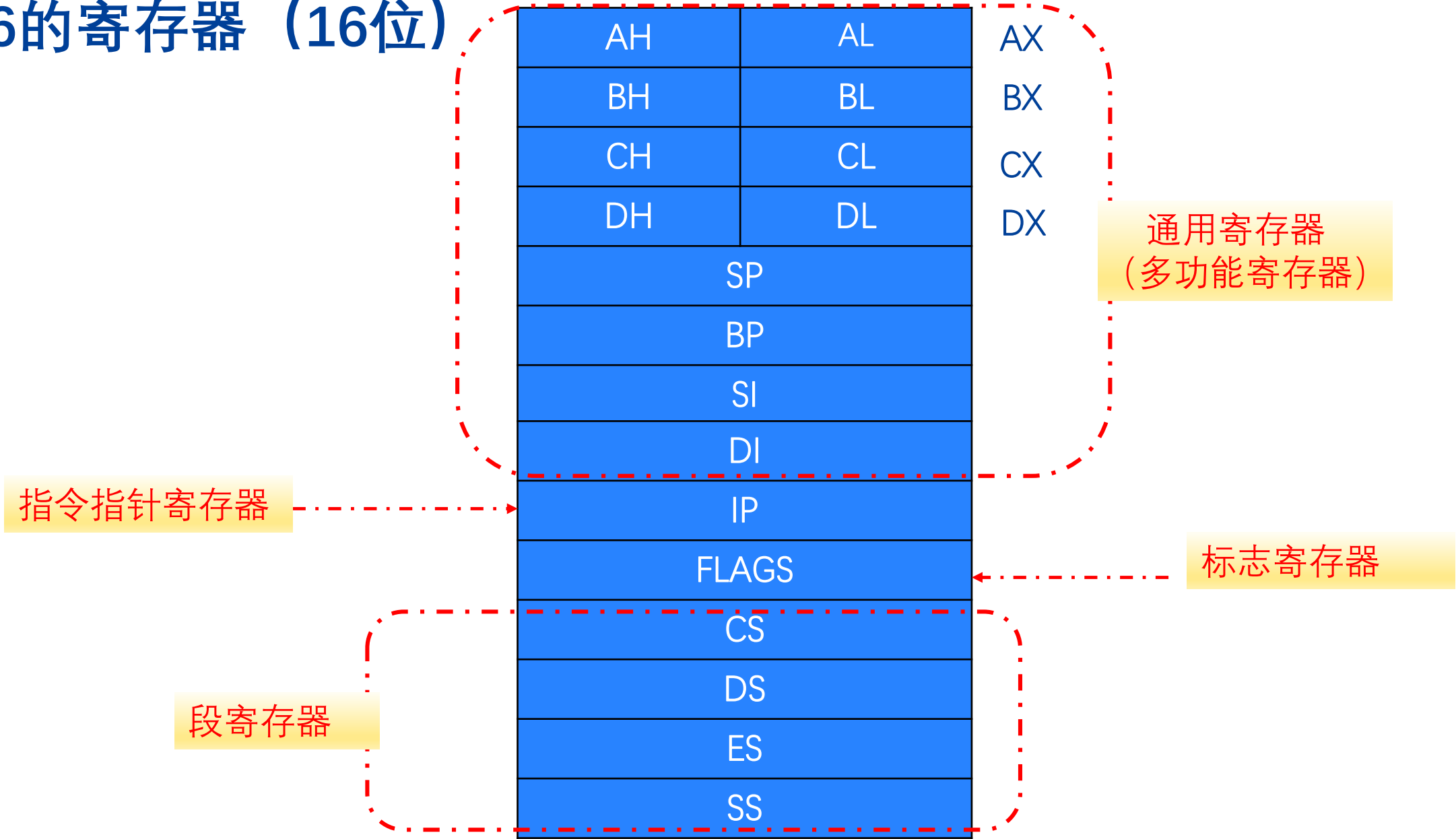
---



存储器寻址

---

# 8086的寄存器 (16位)



## 8086的数据寄存器

- 数据寄存器，共有4个16位寄存器
  - 适用大多数算术运算和逻辑运算指令
  - 每个16位寄存器都可分为两个8位寄存器使用，
    - 例如char c; 数据放置在AL中
  - 除存放通用数据外，各有一些专门的用途：

AX	Accumulator	存放乘除等指令的操作数
BX	Base	存放存储单元的偏移地址
CX	Count	存放计数值
DX	Data	乘法运算产生的部分积 除法运算的部分被除数

AH	AL	AX
BH	BL	BX
CH	CL	CX
DH	DL	DX
SP		
BP		
SI		
DI		
IP		
FLAGS		
CS		
DS		
ES		
SS		

# 指针和变址寄存器

- 16位
- 可以作为算术和逻辑运算的数据寄存器使用
- 也有约定用途
  - SP和BP用于堆栈操作
  - SI和DI用于串操作

SP	stack pointer	堆栈指针寄存器
BP	(stack)base pointer	(堆栈)基址指针寄存器
SI	source index	源变址寄存器
DI	destination index	目的变址寄存器

AH	AL	AX
BH	BL	BX
CH	CL	CX
DH	DL	DX
SP		
BP		
SI		
DI		
IP		
FLAGS		
CS		
DS		
ES		
SS		



# 标志寄存器

## ▪ 标志位

- FLAGS寄存器中包含若干标志位
- 标志位分为两大类：状态标志和控制标志

## ▪ 状态标志反映CPU的工作状态

### ▪ 例如：

- 执行加法运算时是否产生进位
- 运算结果是否为零

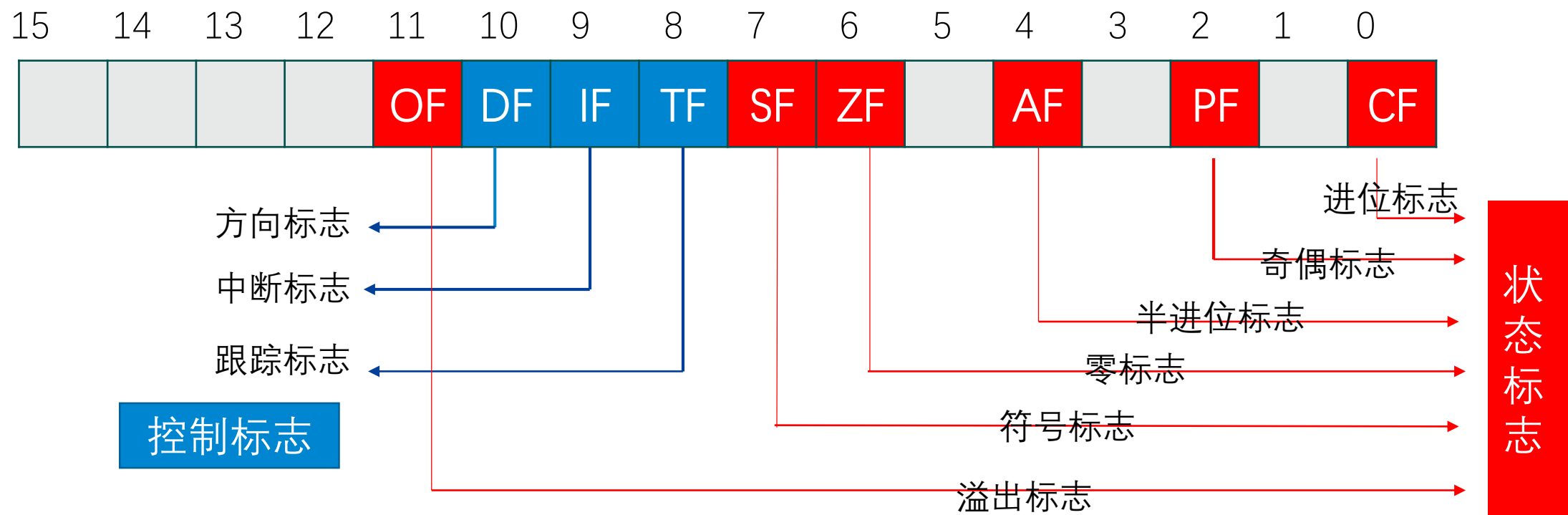
## ▪ 控制标志对CPU的运行起特定控制作用

### ▪ 例如：

- 以单步方式还是连续方式运行
- 是否允许响应外部中断请求

AH	AL	AX
BH	BL	BX
CH	CL	CX
DH	DL	DX
SP		
BP		
SI		
DI		
IP		
FLAGS		
CS		
DS		
ES		
SS		

# 8086的标志位



# IA32 寄存器模型 (16位到32位)

通用寄存器  
(多功能寄存器)

指令指针寄存器

标志寄存器

EAX  
EBX  
ECX  
EDX  
ESP  
EBP  
ESI  
EDI  
EIP  
EFLAGS

	AH	AL
	BH	BL
	CH	CL
	DH	DL
	SP	
	BP	
	SI	
	DI	
	IP	
	FLAGS	
	CS	
	DS	
	ES	
	SS	
	FS	
	GS	

模型

段寄存器



# X86-64 寄存器 (32位到64位)

X86-64 寄存器新增部分

IA32 寄存器新增部分

8086寄存器模型

	64	32	16	8	
RAX			AH	AL	AX
RBX			BH	BL	BX
RCX			CH	CL	CX
RDX			DH	DL	DX
RSP			SP		
RBP			BP		
RSI			SI		
RDI			DI		
RIP			IP		
RFLAGS			FLAGS		
			CS		
			DS		
			ES		
			SS		
			FS		
			GS		

	64	32	16
R8			
R9			
.....			
R15			

# x86-64 通用寄存器

- 所有通用寄存器都从32位扩充到64位
  - 32位通用寄存器EAX、EBX、ECX、EDX、EBP、ESP、ESI和EDI
  - 对应扩展寄存器:RAX、RBX、RCX、RDX、RBP、RSP、RSI和RDI
- 原来IA-32中的每个通用寄存器都可以是8位、16位、32位、64位
  - 如: SIL (8位)、SI (16位)、ESI (32位)、RSI (64位)
  - 可兼容使用原AH、BH、CH和DH寄存器
  - EBP、ESP、ESI和EDI的低8位寄存器分别是BPL、SPL、SIL和DIL
- 新增8个64位通用寄存器 (整数寄存器)
  - R8、R9、R10、R11、R12、R13、R14和R15
  - 可作为8位 (R8B~R15B)、16位 (R8W~R15W) 或 32位寄存器 (R8D~R15D) 使用



# 从16位到64位：x86指令体系结构的演变



1 寄存器模型

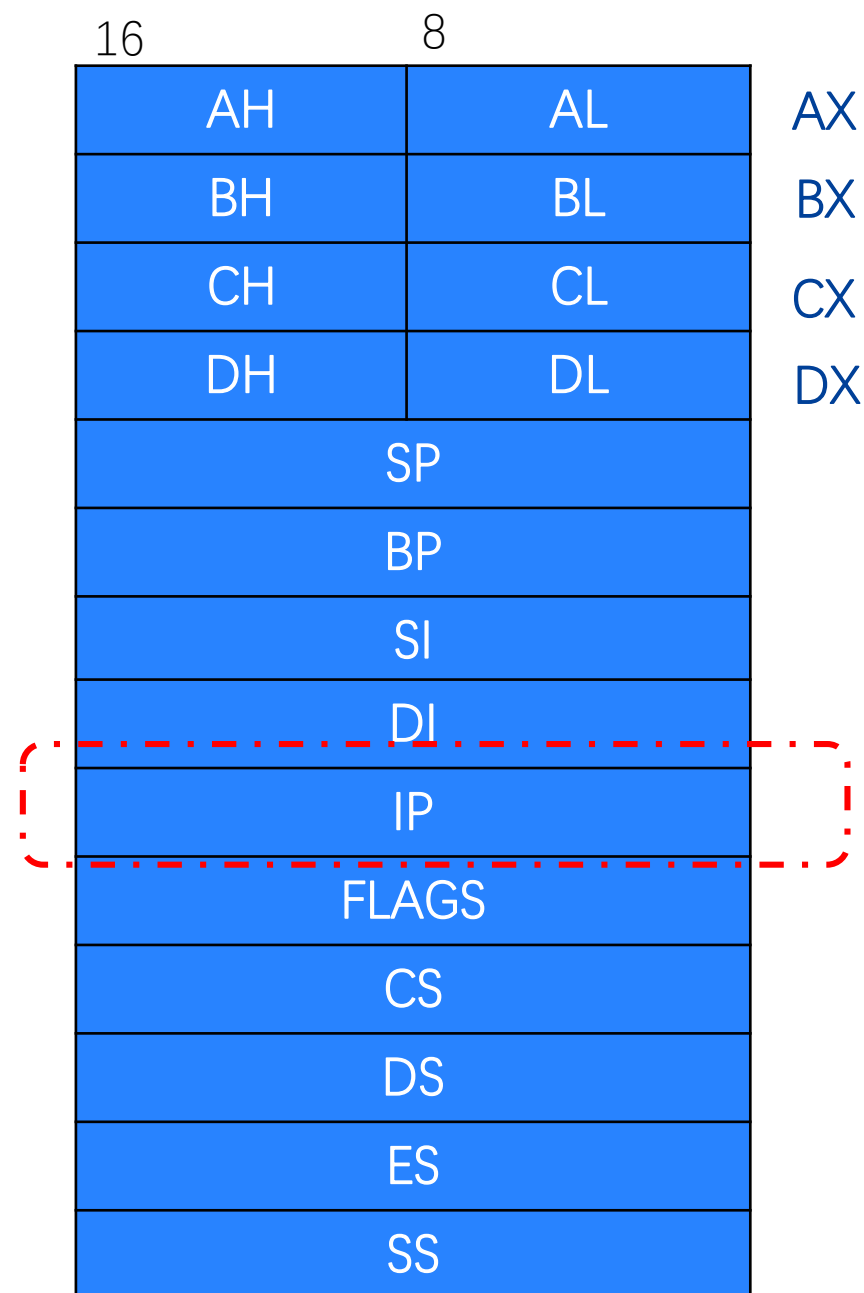
---

★ 2 存储器寻址

---

# 8086的指令寻址

- 指令指针寄存器IP (Instruction Pointer)
  - 相当于其他体系结构中的PC (program counter) 寄存器
  - 内容是一个内存地址, 指向当前需要取出并执行的指令
  - 当CPU从内存中取出一条指令后, IP自动增加, 指向下一指令的地址
  - 转移指令、过程调用/返回指令等会改变IP的内容
  - 程序员不能直接对IP进行存取操作 
- IP寄存器的寻址能力:  $2^{16}=65536(64K)$ 字节单元
- 8086对外有20位地址线 (如何实现20位寻址?) 
  - 寻址范围:  $2^{20}=1M$ 字节单元



# 8086的段寄存器

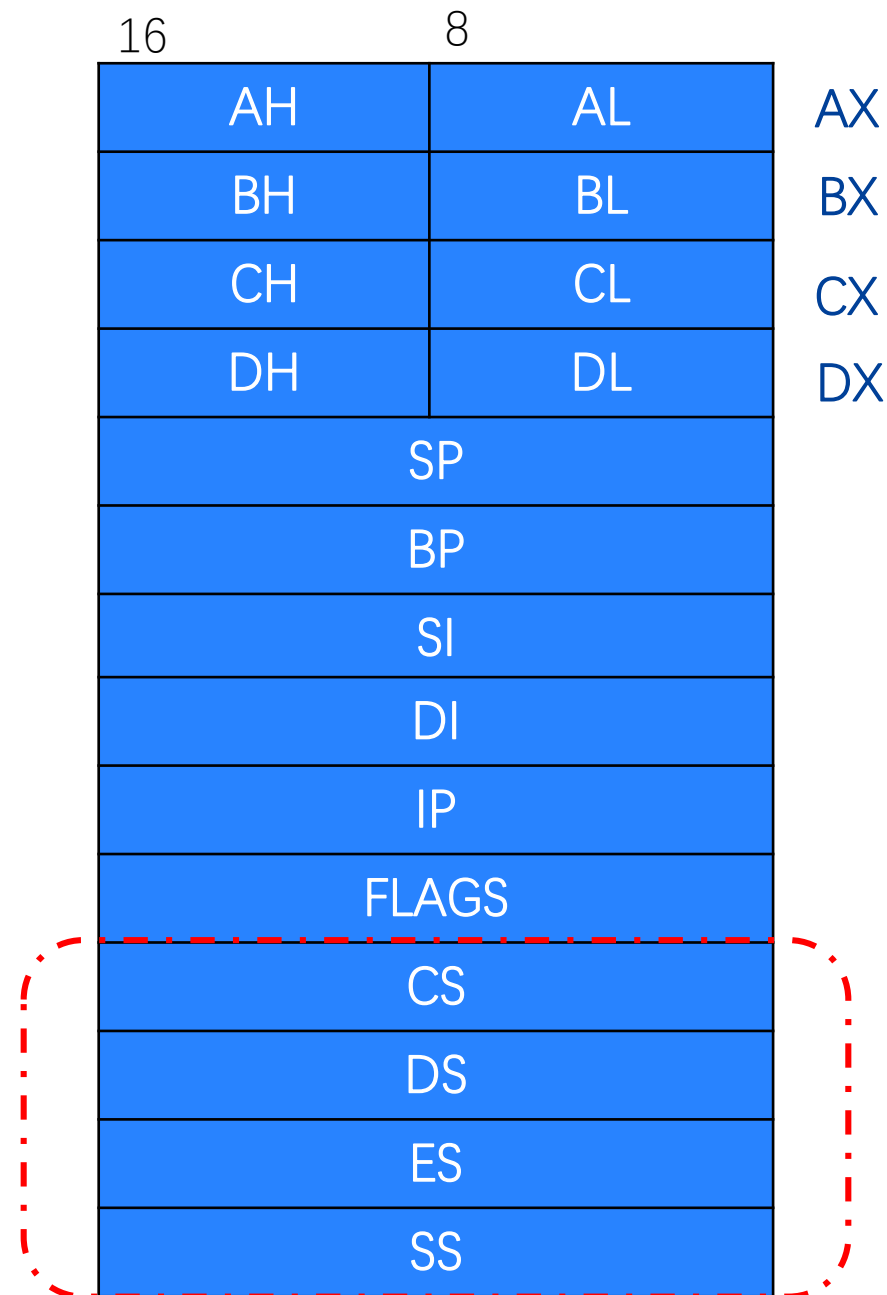
- 段寄存器 (Segment Register)
  - 存储段基址: 该段的起始地址
  - 与其它寄存器联合生成存储器地址

**DS** 数据段寄存器 (Data Segment)

**CS** 代码段寄存器 (Code Segment)

**ES** 附加段寄存器 (Extra Segment)

**SS** 堆栈段寄存器 (Stack Segment)







# 8086的物理地址生成

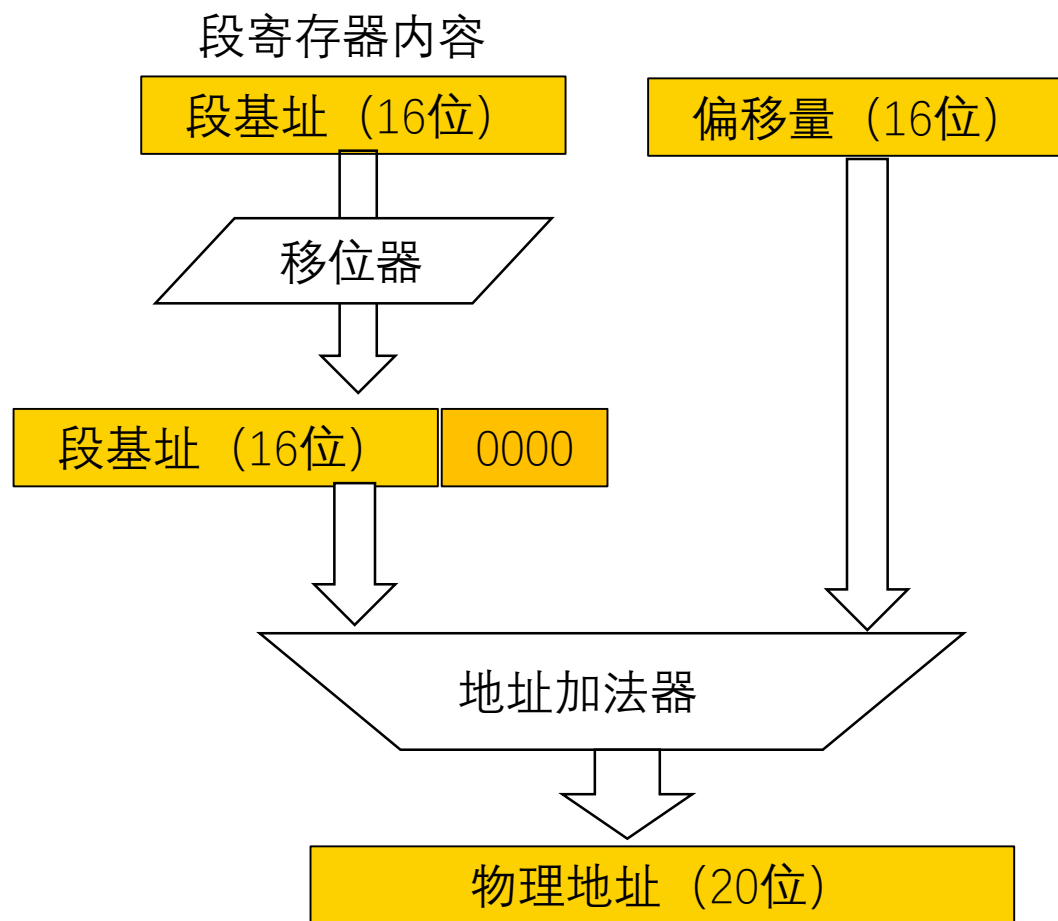
物理地址

$$= \text{段基值} \times 16 + \text{偏移量}$$

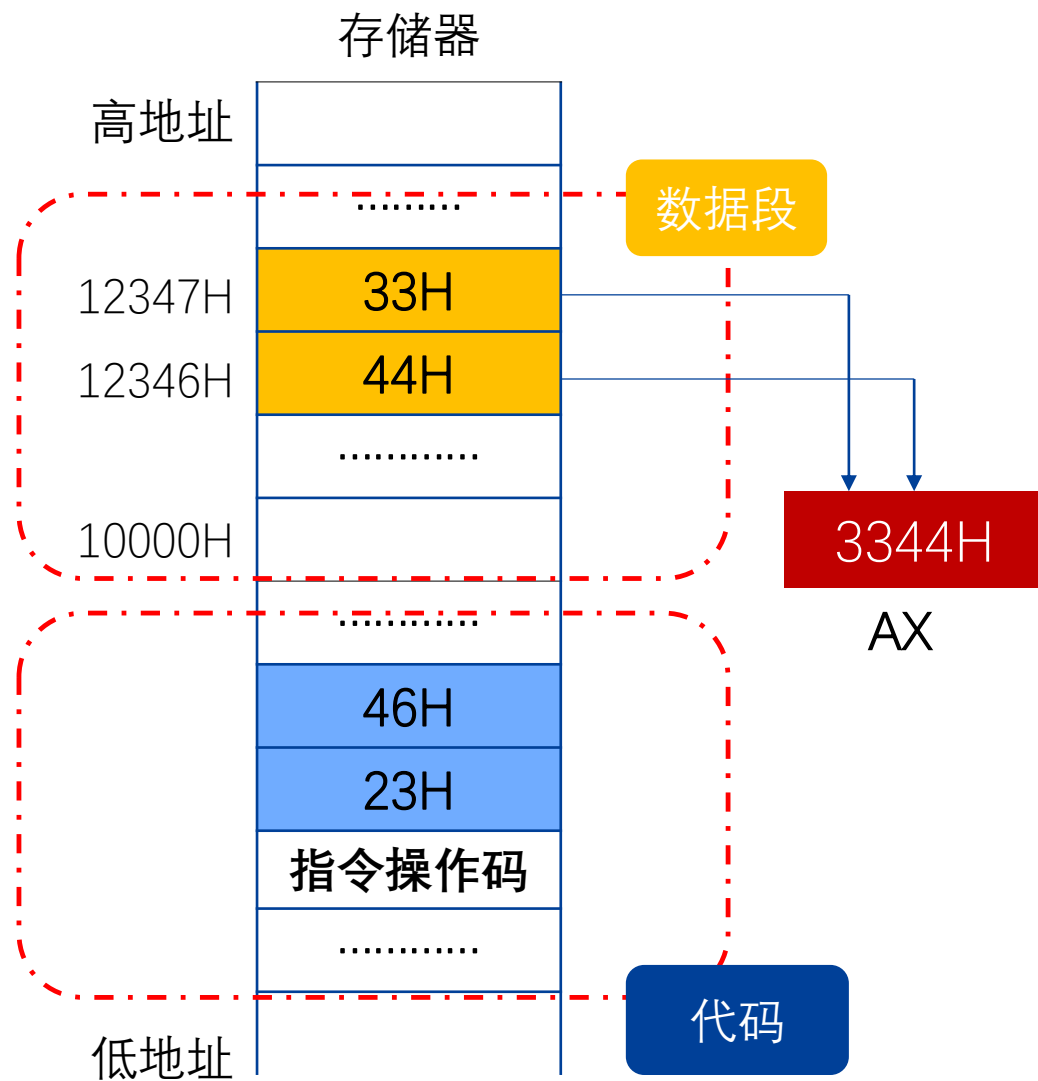
$$= \text{段基值} \ll 4 + \text{偏移量}$$

指令寻址

$$= \text{CS} \ll 4 + \text{IP}$$



# 数据寻址：“段加偏移”的编程实例



汇编指令：

**MOV AX, 2346H**

操作数：

默认存放在DS指向的数据段中

即： [2346H]

假设： DS= 1000H

物理地址 =  $1000H \ll 4 + 2346H$   
= 12346H

# 16位与32位的访存模式对比：“实模式”与“保护模式”

- 8086的访存模式： 实模式 CS: IP
  - CPU通电后，进入实模式
  - 指令的物理地址=CS<<4 + IP
  - 8086时，无操作系统，无任何权限控制信息
  - 程序通过驱动接口可以任意更改段寄存器的值
  - 实模式下访问内存是不安全的
- 80386及以上的微处理器的主要工作模式： 保护模式 CS: EIP
  - 操作系统引入了虚拟存储器的概念
  - 加强了对应用软件的控制，使得系统运行更安全
  - 指令物理地址由CS、EIP和其他寄存器，组合计算

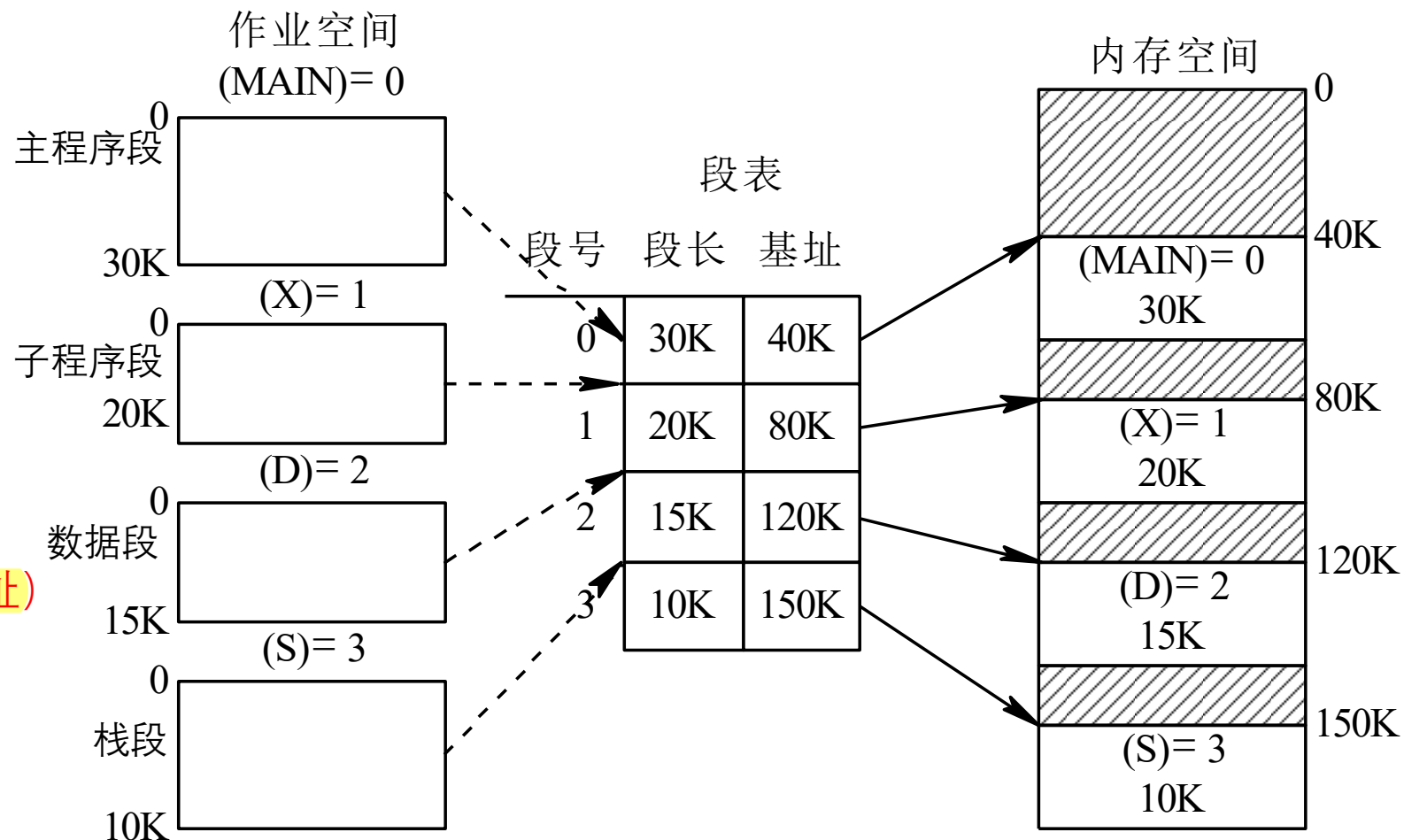
# 分段存储管理方式

- 保护模式：
- 安全性、高效性
  - 信息保护、信息共享
  - 动态增长、动态链接

访问内存时，需要知道：

- 物理内存的起始地址（段基址）
- 内存段的大小（段长）
- 内存段的访问权限  
(可读、可写或可执行)

这些信息太多，不能都存入CS段寄存器



利用段表实现地址映射

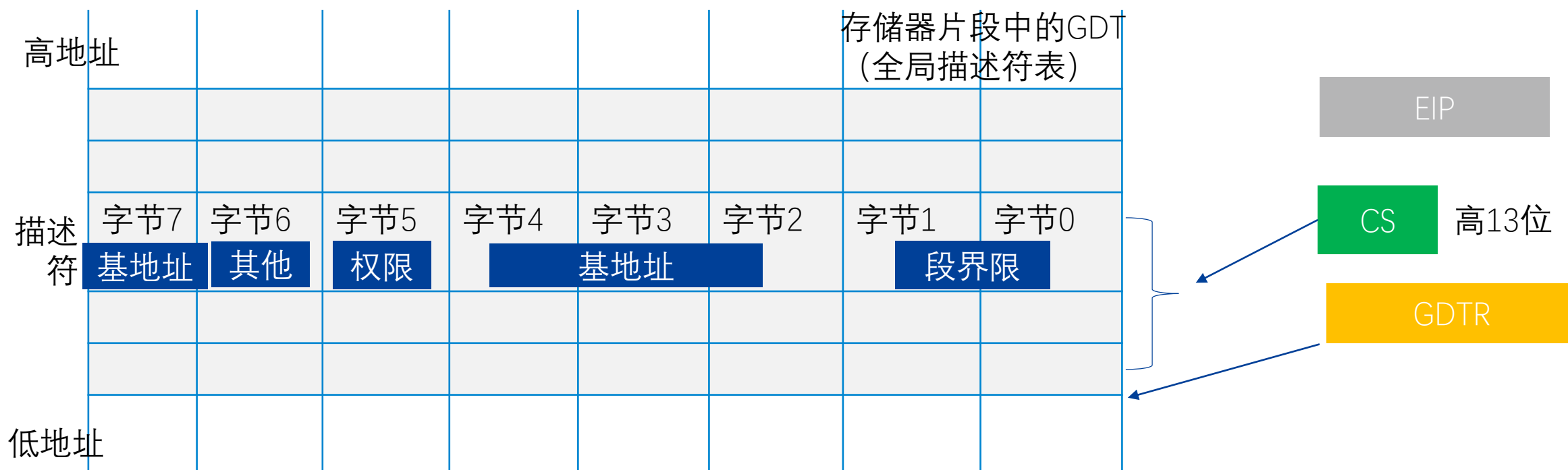
# IA-32的存储器寻址：段式存储

- 段基址不在CS中，而是在段表（内存）中

每一段的信息至少包括：物理内存的起始地址（段基址）、内存段的长度、内存段的访问权限

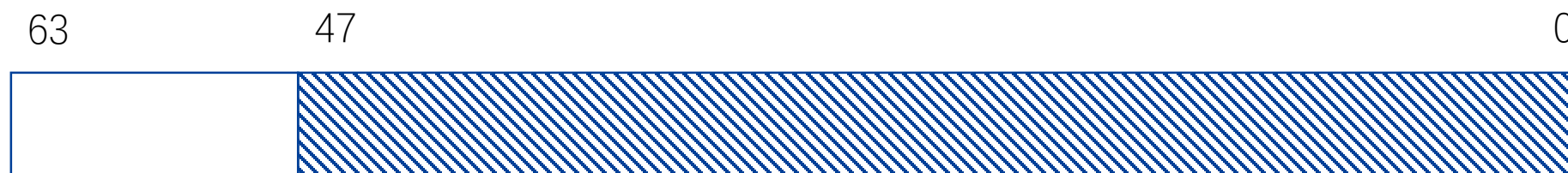
- IA-32增加了两个寄存器：
  - GDTR：全局描述符表的地址寄存器
  - LDTR：局部描述符表的地址寄存器

$$\text{GDTR}[\text{CS} \gg 3].\text{BaseAddr} + \text{EIP} = \text{线性地址} \\ = \text{物理地址（如果不分页存储）}$$



## x86-64的地址和寻址空间

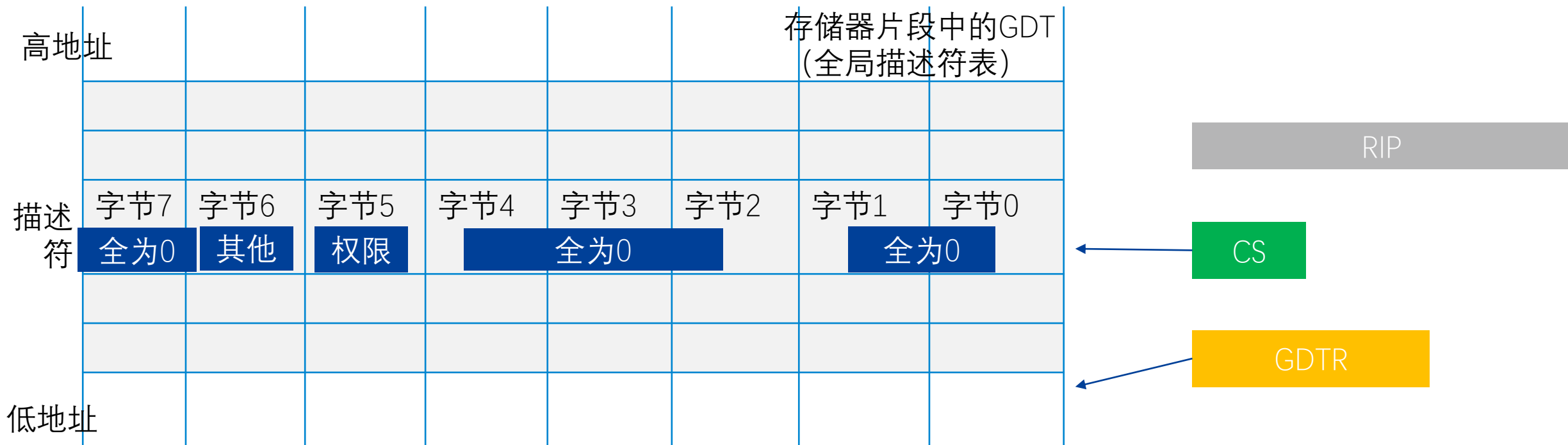
- 字长从32位变为64位，也称为四字（qw: quadword）
- 逻辑地址理论上可访问 $2^{64}$ 字节或16EB（ExaByte）的存储空间
- 但实际上，AMD和Intel的x86-64仅支持48位虚拟地址，
- 程序的虚拟地址空间大小为 $2^{48}=256\text{TB}$
- 不再采用段式存储，而是采用页式存储



# X86-64 的存储器寻址

- 取消了段式存储和段页式存储，只有页式存储
- 段基地器全为0(段映射根本没用)，CS: RIP 仍然是逻辑地址（虚拟地址）
- 通过多级页表映射转换为物理地址

$$\text{GDTR}[\text{CS}].\text{BaseAddr} + \text{RIP} = \text{地址} \\ = \text{逻辑地址}$$





# 小结

- X86 体系结构
- 寄存器模型
- 存储器寻址



谢谢！

