计算机组织与体系结构

Computer Architectures

陆俊林

北京大学 Peking University

2024年春季学期

第三讲 CISC和x86指令

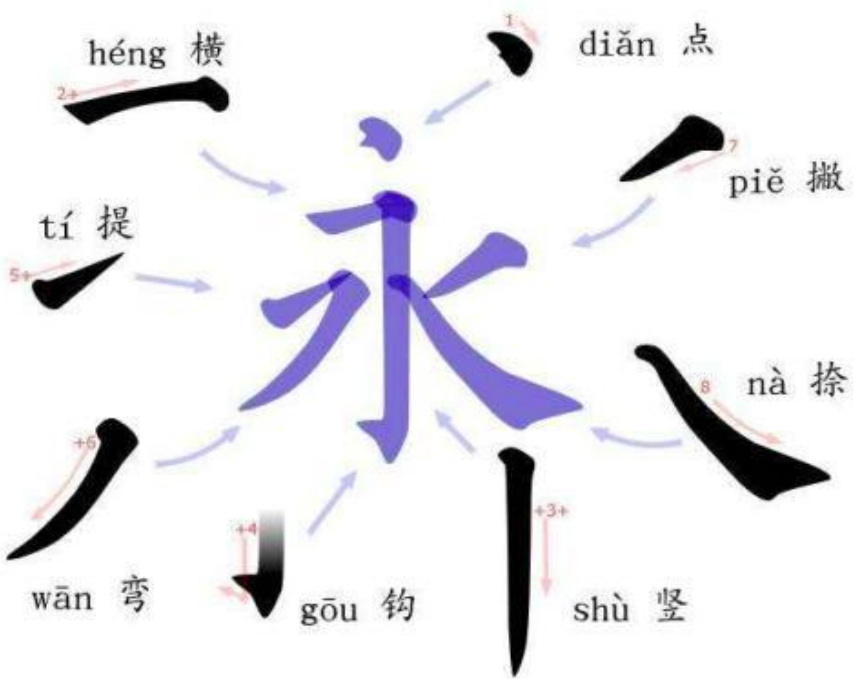
本讲要点

分类介绍了x86的若干基本指令，重点说明其CISC的特点，最后简要介绍了x86指令系统逐步新增的指令。

回顾：CISC与RISC

- CISC: Complex Instruction Set Computer
  - 复杂指令系统计算机（复杂指令集计算机）
- RISC: Reduced Instruction Set Computer
  - 精简指令系统计算机（精简指令集计算机）

类比：笔画和字

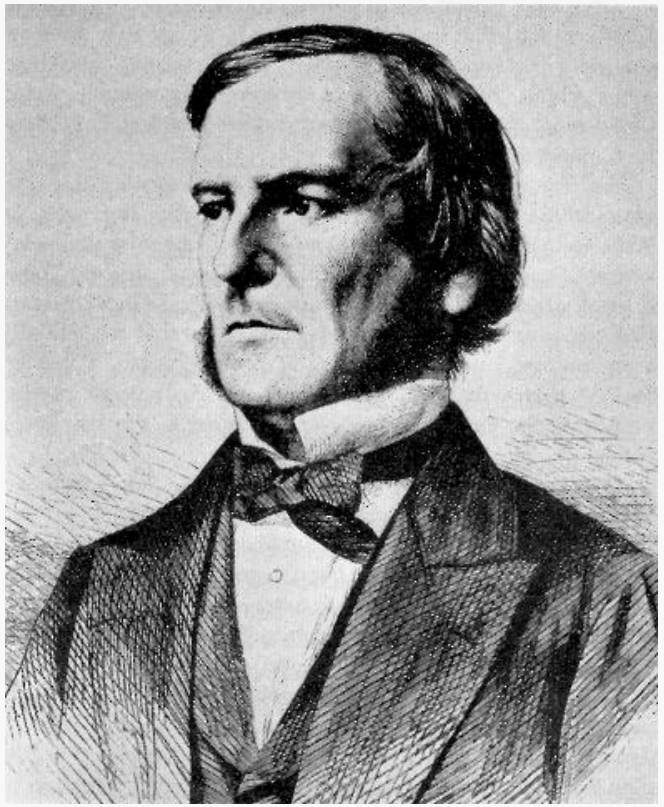


序号	笔画	名称	例字	序号	笔画	名称	例字
1	丶	点	主	17	乙	横折弯钩	九
2	一	横	三	18	𠃌	横撇弯钩	郑
3	丨	竖	十	19	𠃍	横折折折钩	奶
4	丿	撇	八	20	𠃎	竖折折钩	与
5	乀	捺	人	21	乚	竖弯	西
6	𠂉	提	打	22	乙	横折弯	没
7	㇀	撇点	女	23	𠃌	横折	口
8	㇁	竖提	长	24	𠄌	竖折	山
9	𠄎	横折提	语	25	𠂆	撇折	云
10	㇂	弯钩	了	26	𠃌	横撇	水
11	亅	竖钩	小	27	𠃍	横折折撇	及
12	𠃌	竖弯钩	孔	28	𠃎	竖折撇	专
13	乀	斜钩	我	29	乙	横斜钩	风
14	㇄	卧钩	心	30	𠃎	竖折折	鼎
15	𠃌	横钩	写	31	乙	横折折	凹
16	𠃌	横折钩	力	32	𠃍	横折折折	凸

精简到极致：一条指令？

- 布尔和布尔代数
  - 1854年，代表作《The Laws of Thought》
  - 全名：《An Investigation of the Laws of Thought on Which are Founded the Mathematical Theories of Logic and Probabilities》
- 布尔代数的基本思想
  - 两个元素：真，假
  - 三种运算：与，或，非

这三种运算又都可以转换成
“与非”或者“或非”运算



乔治·**布尔**
George Boole
1815年~1864年

类比：理发师的装备

- 哪个水平高？哪个效果好？

CISC与RISC

- CISC: Complex Instruction Set Computer
  - 复杂指令系统计算机（复杂指令集计算机）
- RISC: Reduced Instruction Set Computer
  - 精简指令系统计算机（精简指令集计算机）

“一快遮百丑”

回顾：体系结构与微体系结构

- 体系结构：architecture
- 微体系结构：microarchitecture

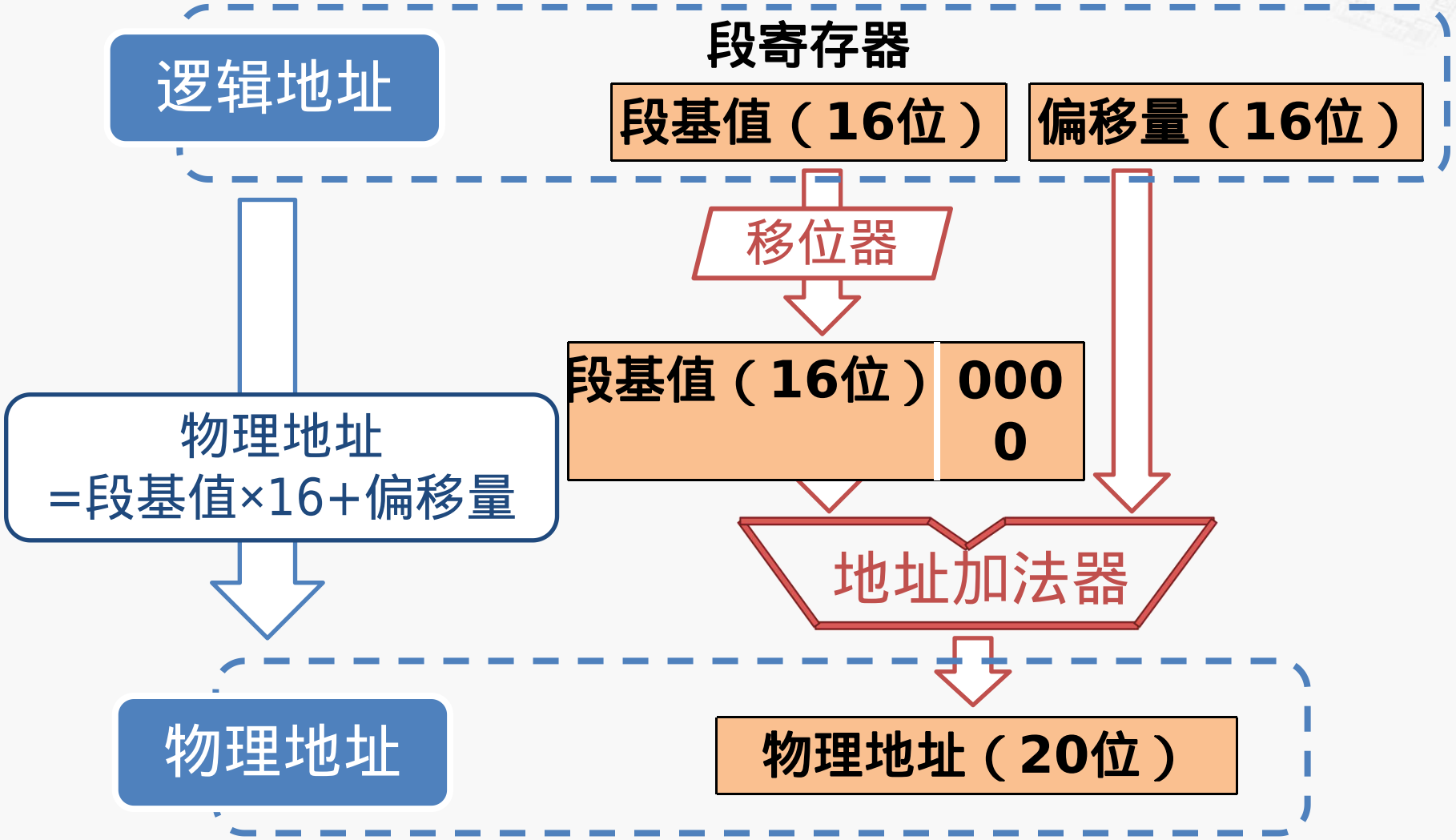
不同的体系结构

不同的微体系结构

增加指令或特性

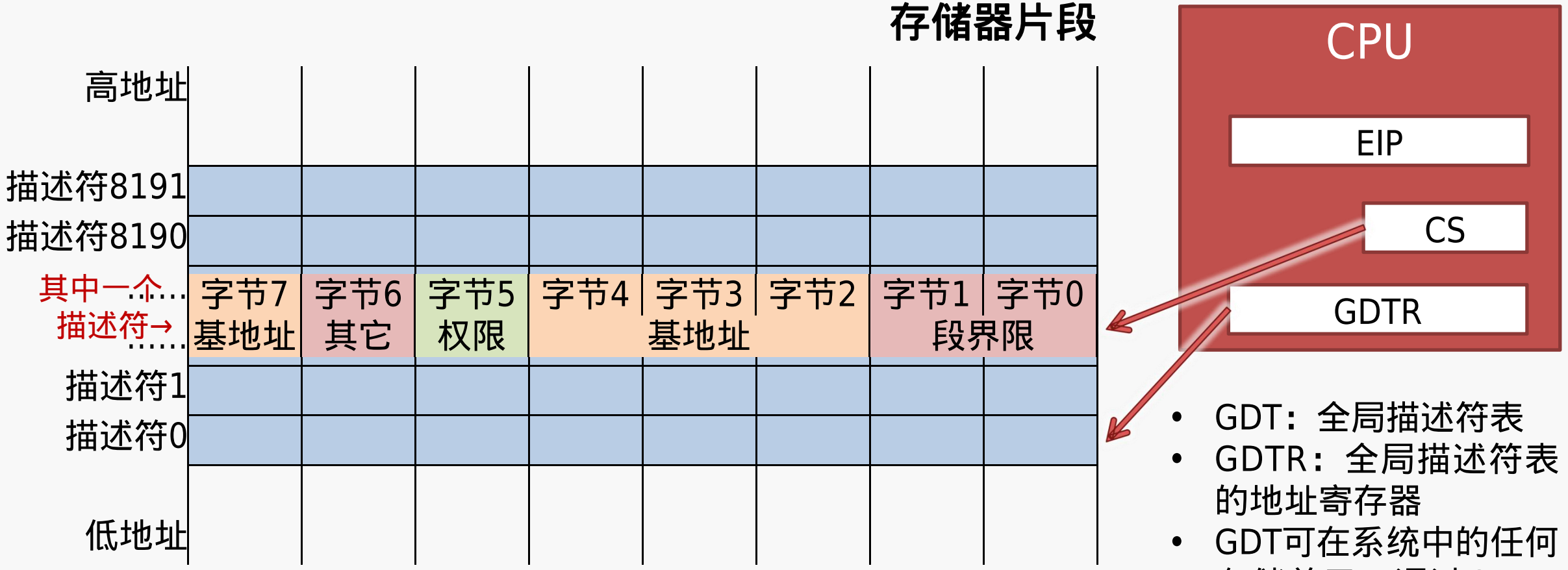
- 增加指令或特性
  - 增加车筐
  - 增加后排座
  - ……
- 兼容性
  - 在新的体系结构上运行旧的软件
  - 在旧的体系结构上运行新的软件

回顾：8086的物理地址生成



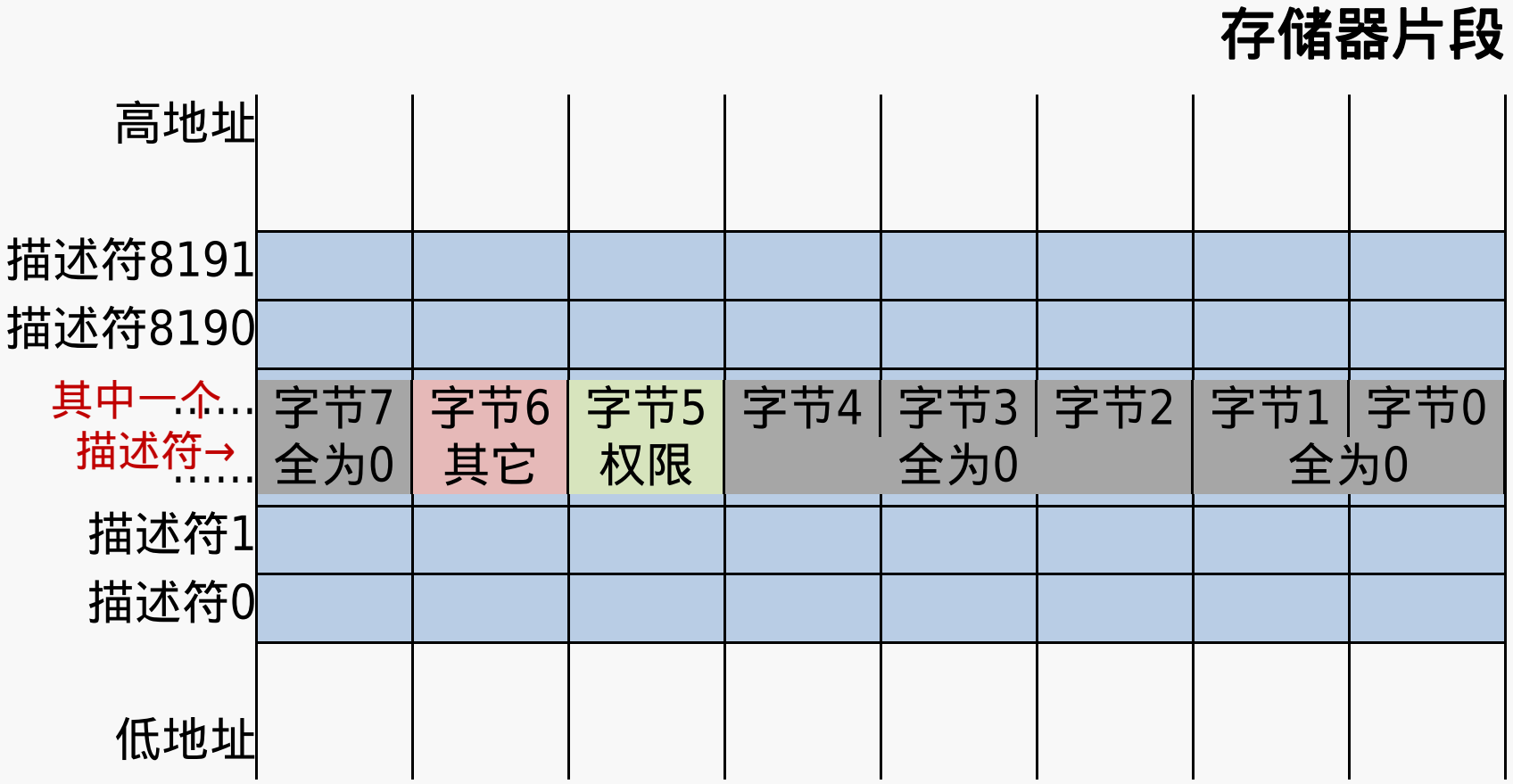
回顾： IA-32的存储器寻址

- 保护模式下，段基址不在CS中，而是在内存中



- GDT：全局描述符表
- GDTR：全局描述符表的地址寄存器
- GDT可在系统中的任何存储单元，通过GDTR定位

回顾： x86-64的描述符



注：描述符中没有了段基址和段界限，只有访问权限字节和若干控制位。所有的代码段都从地址0开始。

主要内容

通过学习本课程
了解计算机的发展历程，理解计算机的组成原理，掌握计算机的设计方法

指令分类举例

1.传送类指令
2.运算类指令
3.转移类指令
4.控制类指令

传送指令

- 作用
  - 把数据或地址传送到寄存器或存储器单元中
- 分类
  - 分四大类
  - 共14条指令

传送指令的列表

分组	助记符	功能	操作数类型
通用数据传送指令	MOV	传送	字节/字
	PUSH	压栈	字
	POP	弹栈	字
	XCHG	交换	字节/字
累加器专用传送指令	XLAT	换码	字节
	IN	输入	字节/字
	OUT	输出	字节/字
地址传送指令	LEA	装入有效地址	字
	LDS	把指针装入寄存器和DS	4个字节
	LES	把指针装入寄存器和ES	4个字节
标志传送指令	LAHF	把标志装入AH	字节
	SAHF	把AH送标志寄存器	字节
	PUSHF	标志压栈	字
	POPF	标志弹栈	字

MOV指令说明

MOV指令（传送）

- 格式：MOV　DST, SRC
- 操作：DST←SRC

- 说明：
  - DST表示目的操作数，SRC表示源操作数
  - MOV指令把一个字节或字操作数从源传送至目的，源操作数保持不变

MOV指令示例

```
MOV   AL, BL
MOV   [DI], AX
MOV   CX, DS:[1000H]
MOV   BL, 40
MOV   WORD PTR[SI] , 01H
```

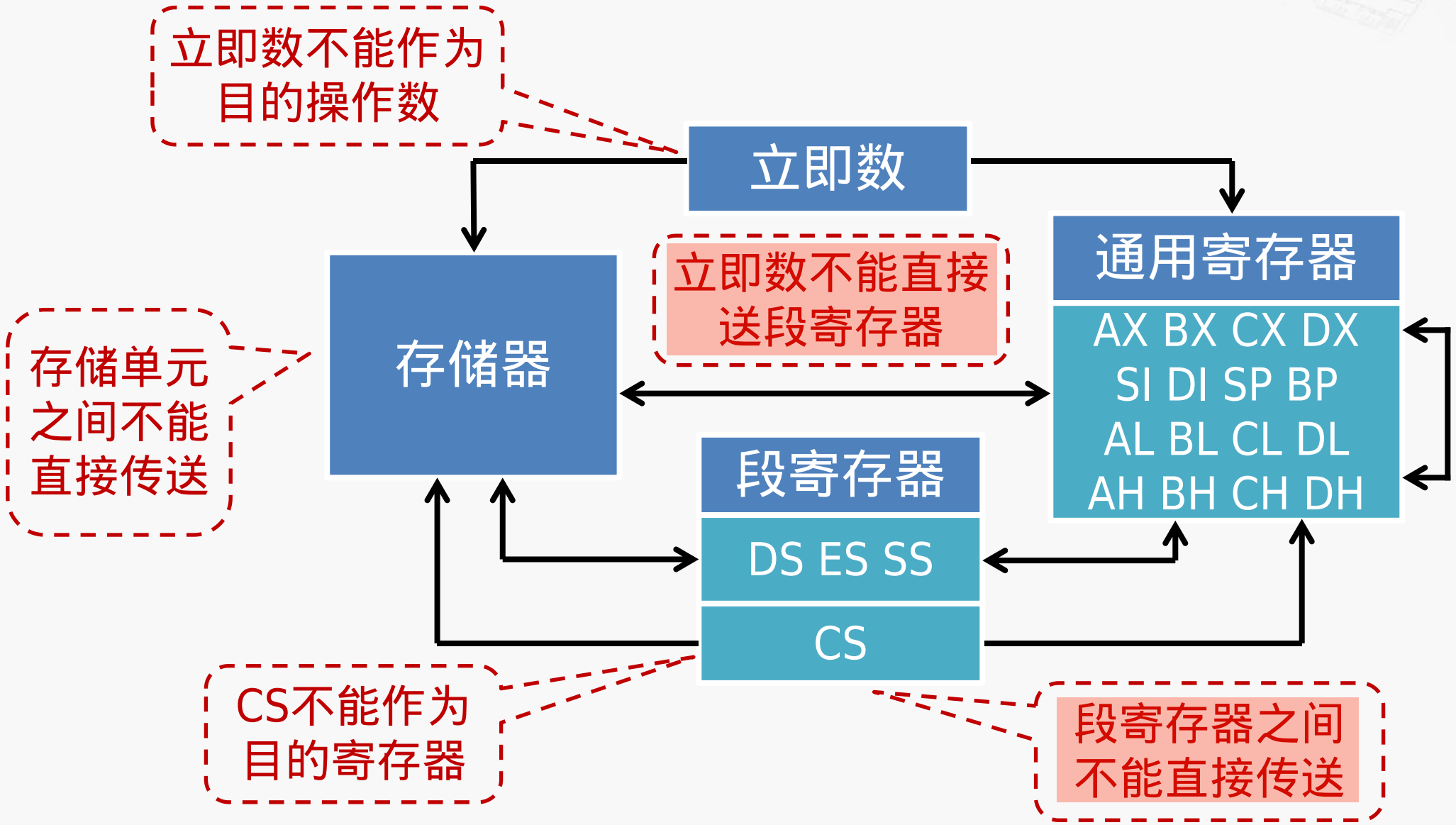
注:

BYTE PTR：字节长度标记

WORD PTR：字长度标记

DWORD PTR：双字长度标记

MOV指令的传送方向和限制



MOV指令编码（七种类型）

76543210	76543210	76543210	76543210	76543210	76543210
1100011w	mod 000 r/m	DISP-LO	DISP_HI	data	data if w=1
100010dw	mod reg r/m	DISP-LO	DISP_HI	Displacement	
10001110	mod 0 SR r/m	DISP-LO	DISP_HI		
10001110	mod 0 SR r/m	DISP-LO	DISP_HI		
1010000w	addr-lo	addr-hi			
1010001w	addr-lo	addr-hi			
1011wreg	data	data if w=1			

MOV指令编码示例

76543210	76543210	76543210	76543210	76543210	76543210
1011 w reg	data	data if w=1			
1011 1 000	11101110	00010000			
B8	EE	10			

立即数到寄存器

MOV AX,10EEH

76543210	76543210	76543210	76543210	76543210	76543210
100010dw	mod reg r/m	DISP-LO	DISP_HI		
10001011	000 011 11				
8B	0F				

存储器到寄存器

MOV CX,[BX]

现在，让我们根据您给出的二进制代码 `10001011 00001111` 来解析这个指令：

- `10001011` 是操作码，表示 `MOV` 指令，用于将数据从内存移动到寄存器。
- `00001111` 是 ModR/M 字节。在这里，`mod` 部分为 `00`，`reg` 部分为 `001`，`r/m` 部分为 `111`。

根据上面的 `mod reg r/m`：

- `mod` 为 `00`，表示没有位移，直接从寄存器寻址。
- `reg` 为 `001`，表示目标寄存器是 `CX`（在基于 `w` 标志的寄存器编码中）。
- `r/m` 为 `111`，表示使用 `BX` 寄存器作为内存寻址的基址。

因此，根据 ModR/M 字节的解析，没有 `DISP-LO` 和 `DISP-HI` 部分，因为 `mod` 为 `00` 表示没有位移。

所以，整个指令 `10001011 00001111` 对应的汇编指令是：

```
MOV CX, [BX]
```

这表示将 `BX` 寄存器指向的内存地址中的数据移动到 `CX` 寄存器中。

XCHG指令说明

XCHG指令（交换）

- 格式：XCHG　OPR1, OPR2
- 操作：OPTR1 ↔ OPTR2
- 说明：
  - 两个操作数的位宽要相同
  - 两个操作数的类型包括：
    - 寄存器/存储器
    - 存储器/寄存器
    - 寄存器/寄存器
  - 不允许使用段寄存器

XCHG指令示例

- 用XCHG指令完成“存储器中两个字节单元内容的交换”

```
MOV   BL, [2035H]
MOV   CL, [2045H]
MOV   [2045H], BL
MOV   [2035H], CL
```

```
MOV   BL, [2035H]
XCHG  BL, [2045H]
MOV   [2035H], BL
```

XCHG指令的两种编码：
1、寄存器/存储器与寄存器交换
2、寄存器和累加器（AX）交换

76543210	76543210	76543210	76543210
1000011w	mod reg r/m	DISP-LO	DISP_HI
10010reg			

XLAT指令说明

XLAT指令（换码，查表）

- 格式：XLAT
- 操作：

（事先在数据段中定义了一个字节型数据表）

① 从BX中取得数据表起始地址的偏移量

② 从AL中取得数据表项索引值

③ 在数据表中查得表项内容

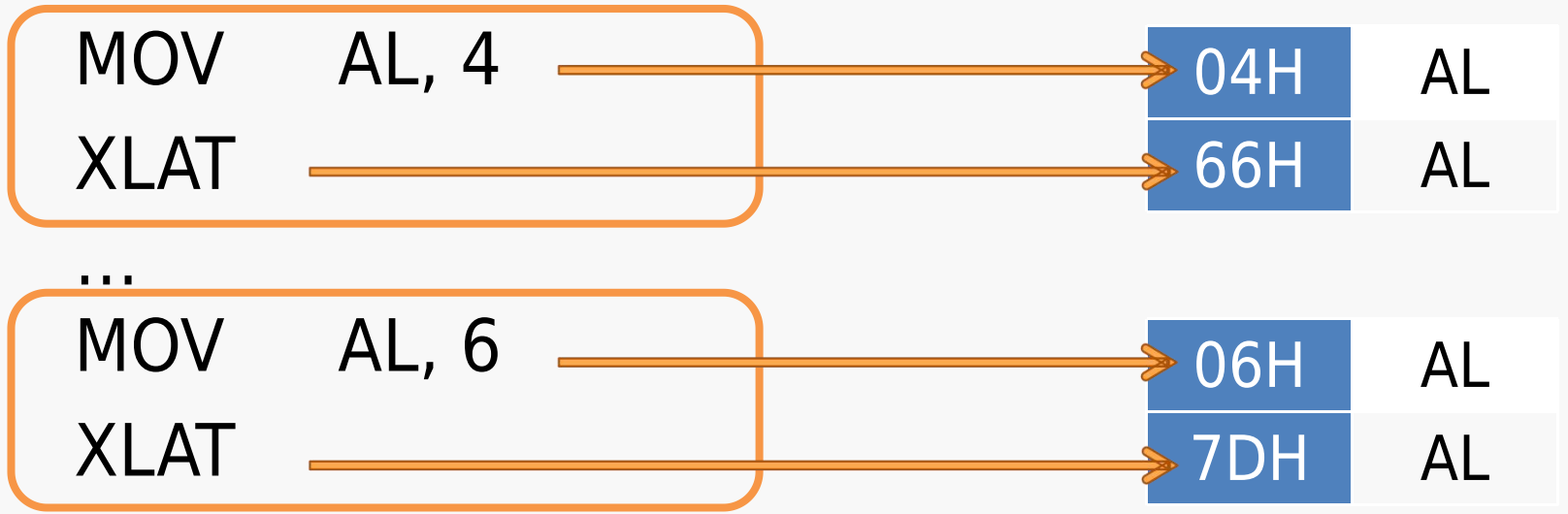
④ 将查得的表项内容存入AL

XLAT指令示例

```
TAB     DB    3FH, 06H, 5BH, 4FH, 66H
        DB    6DH, 7DH, 07H, 7FH, 6FH

...
MOV     BX, OFFSET  TAB

...
MOV     AL, 4
XLAT
...
MOV     AL, 6
XLAT
...
```



主要内容

通过学习本课程
了解计算机的发展历程，理解计算机的组成原理，掌握计算机的设计方法

指令分类举例

1.传送类指令

2.运算类指令

3.转移类指令

4.控制类指令

算术运算指令

- 作用
  - 完成加、减、乘、除等算术运算
  - 提供运算结果调整、符号扩展等功能
- 分类
  - 分五大类，共20条指令

- 操作数的限制
  - 对于双操作数的指令，限制与MOV指令相同
    - 目的操作数不能是立即数或CS寄存器
    - 两个操作数不能同时为存储器操作数

算术运算指令的列表

分组	助记符	功能	操作数类型
加法	ADD	加	字节/字
	ADC	加（带进位）	字节/字
	INC	加1	字节/字
减法	SUB	减	字节/字
	SBB	减（带借位）	字节/字
	DEC	减1	字节/字
	NEG	取补	字节/字
	CMP	比较	字节/字
乘法	MUL	乘（不带符号）	字节/字
	IMUL	乘（带符号）	字节/字
除法	DIV	除（不带符号）	字节/字
	IDIV	除（带符号）	字节/字

算术运算指令的列表

分组	助记符	功能	操作数类型
符号扩展	CBW	将字节扩展为字	/
	CWD	将字扩展为双字	/
十进制调整	AAA	加法的ASCII调整	/
	DAA	加法的十进制调整	/
	AAS	减法的ASCII调整	/
	DAS	减法的十进制调整	/
	AAM	乘法的ASCII调整	/
	AAD	除法的ASCII调整	/

加法类指令说明

ADD指令（加）

- 格式：ADD　DST, SRC
- 操作：DST←DST+SRC

ADC指令（带进位的加）

- 格式：ADC　DST, SRC
- 操作：DST←DST+SRC+CF

INC指令（加1）

- 格式：INC　OPR
- 操作：OPR←OPR+1

```
ADD  BL, 8
ADD  WORD PTR[BX], 01H

ADD  AX, CX
ADC  AX, DX

INC  CL
```

示例

加法类指令编码

76543210	76543210	76543210	76543210	76543210	76543210
100000sw	mod 000 r/m	DISP-LO	DISP_HI	data	data if w=1
000000dw	mod reg r/m	DISP-LO	DISP_HI		
0000010w	data	data if w=1			

立即数到内存

寄存器和内存之间加

立即数加到AX

ADD指令编码：
1、寄存器/存储器与寄存器相加
2、立即数加至寄存器/存储器
3、立即数加至累加器

76543210	76543210	76543210	76543210
1111111w	mod 000 r/m	DISP-LO	DISP_HI
01000reg			

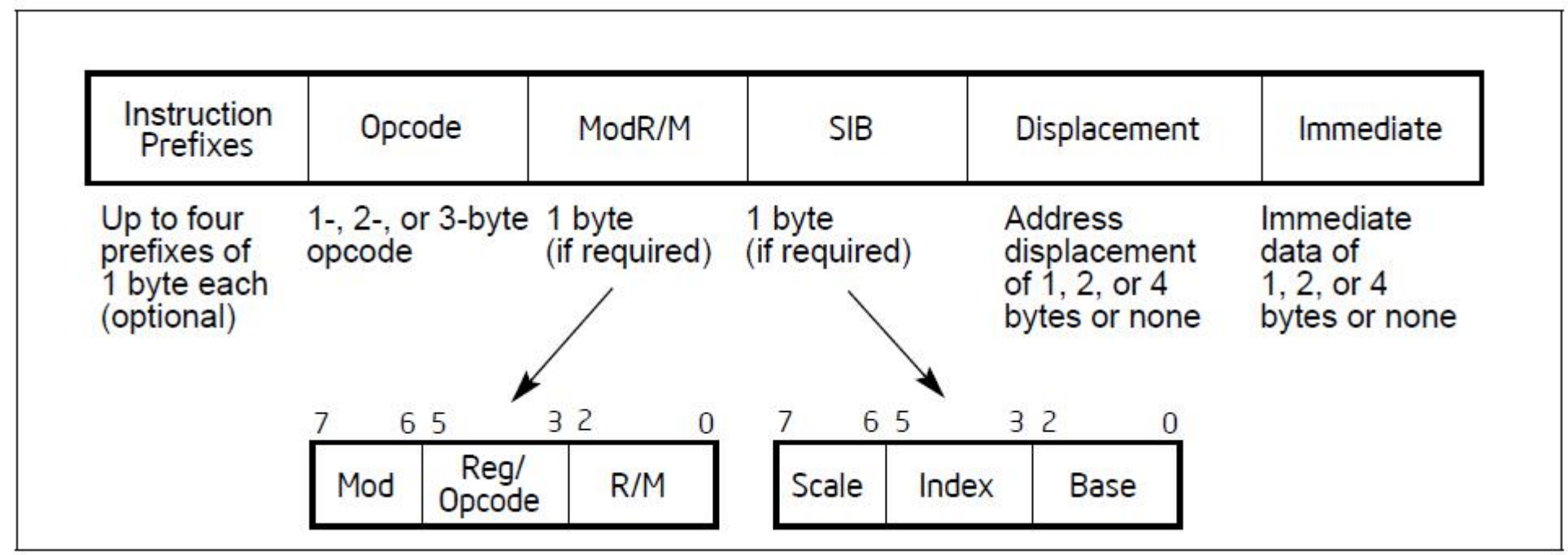
最短的指令（之一）

INC指令编码：
1、寄存器/存储器
2、寄存器

“最长的指令”

LOCK ADD DWORD PTR ES:[EAX+ECX*8+11223344H],12345678H

指令编码（15个字节）：26 66 67 F0 81 84 C8 44 33 22 11 78 56 34 12



十进制调整指令说明

DAA指令（加法十进制调整指令）

- 格式：DAA
- 操作：
  - 跟在二进制加法指令之后
  - 将AL中的“和”数调整为压缩BCD数格式
  - 调整结果送回AL

当执行二进制加法后，如果在低四位（也就是一个BCD数字）的结果大于9或者设置了半进位标志（Auxiliary Carry Flag），DAA指令就会被用来添加6到低四位。如果高四位的结果大于9或者设置了进位标志（Carry Flag），那么DAA指令还会添加60（即0110 0000二进制）到高四位。

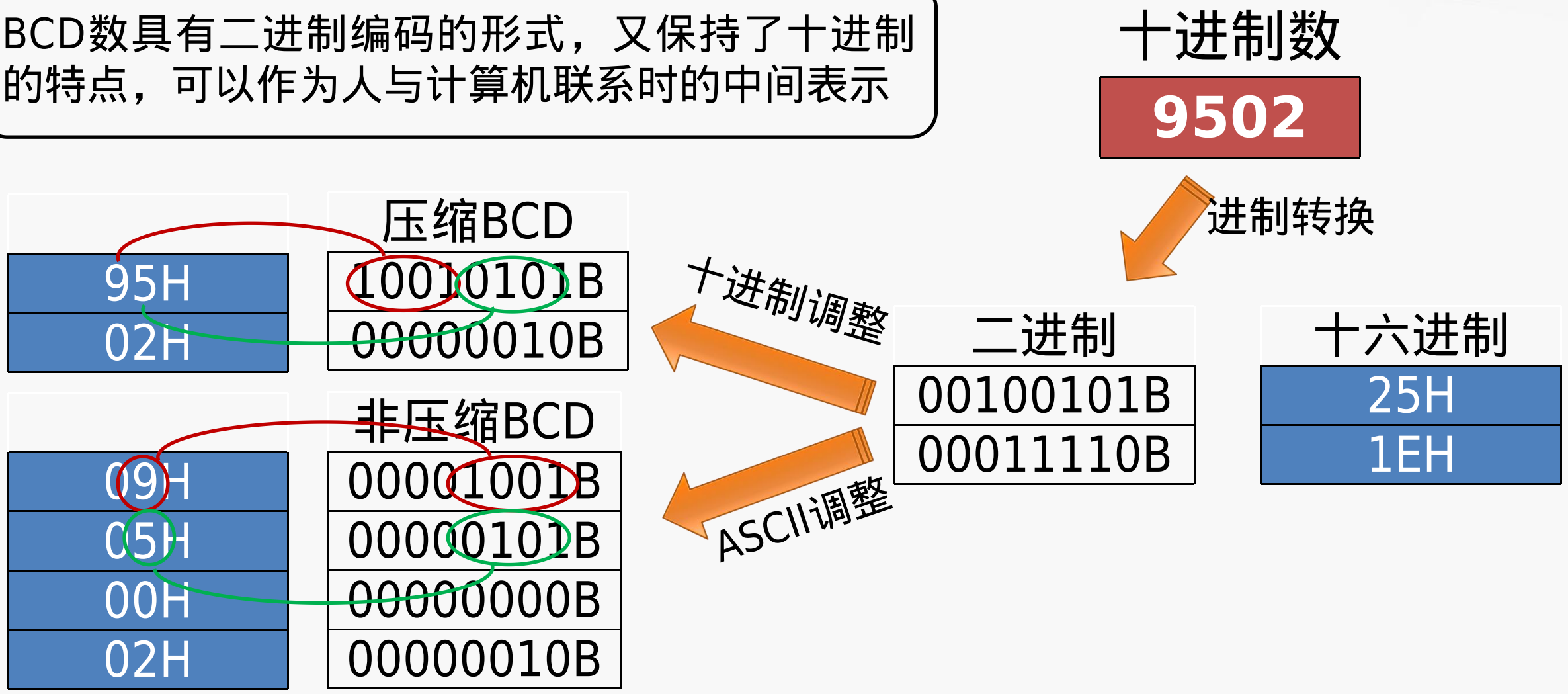
如果大于9，那么对每4位二进制数 +6， 让它在十进制表示下进位

示例

```
MOV  AL, 27H     ; AL=27H
ADD  AL, 15H     ; AL=3CH
DAA              ; AL=42H
```

BCD (Binary-Coded Decimal)

BCD数具有二进制编码的形式，又保持了十进制的特点，可以作为人与计算机联系时的中间表示



主要内容

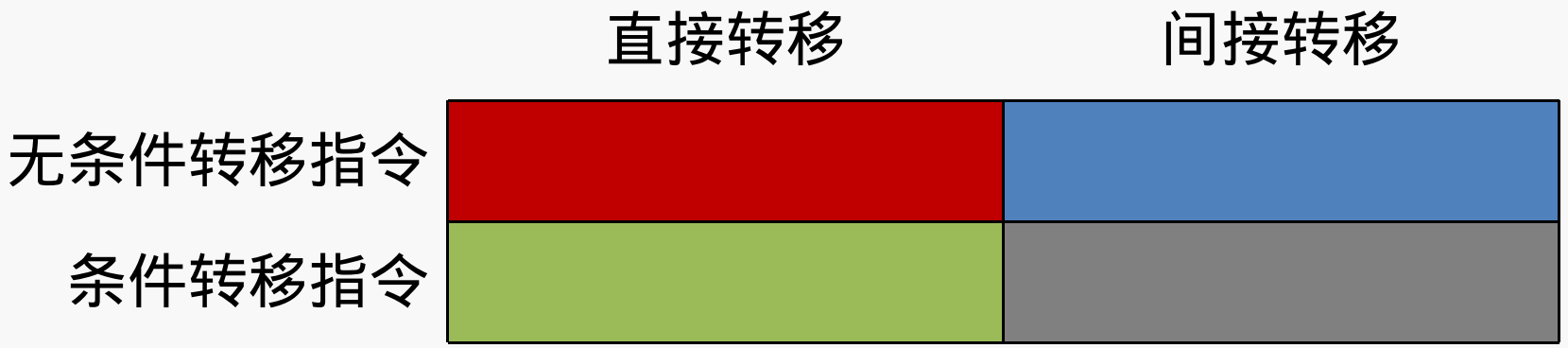
通过学习本课程
了解计算机的发展历程，理解计算机的组成原理，掌握计算机的设计方法

指令分类举例

1.传送类指令
2.运算类指令
3.转移类指令
4.控制类指令

转移指令

- 作用
  - 改变指令执行顺序
- 说明
  - 根据是否有判断条件，分为无条件转移指令和条件转移指令
  - 根据转移目标地址的提供方式，可分为直接转移和间接转移



转移指令的列表（1）

分组	格式	功能	测试条件
无条件转移指令	JMP LABEL	无条件转移	
	CALL LABEL	过程调用	
	RET	过程返回	

无条件转移指令 - 直接转移

- 短转移：JMP SHORT LABEL
  - 操作：IP←IP+8位的位移量（-128~127Byte）
- 近转移：JMP NEAR PTR LABEL
  - 操作：IP←IP+16位的位移量（±32KByte）
- 远转移：JMP FAR PTR LABEL
  - 操作：IP←LABEL的偏移地址；CS ←LABEL的段基值

1. 位移量是一个带符号数，为LABEL的偏移地址与当前EIP/IP值之差
2. 从80386开始，近转移可以使用32位的位移量

说明

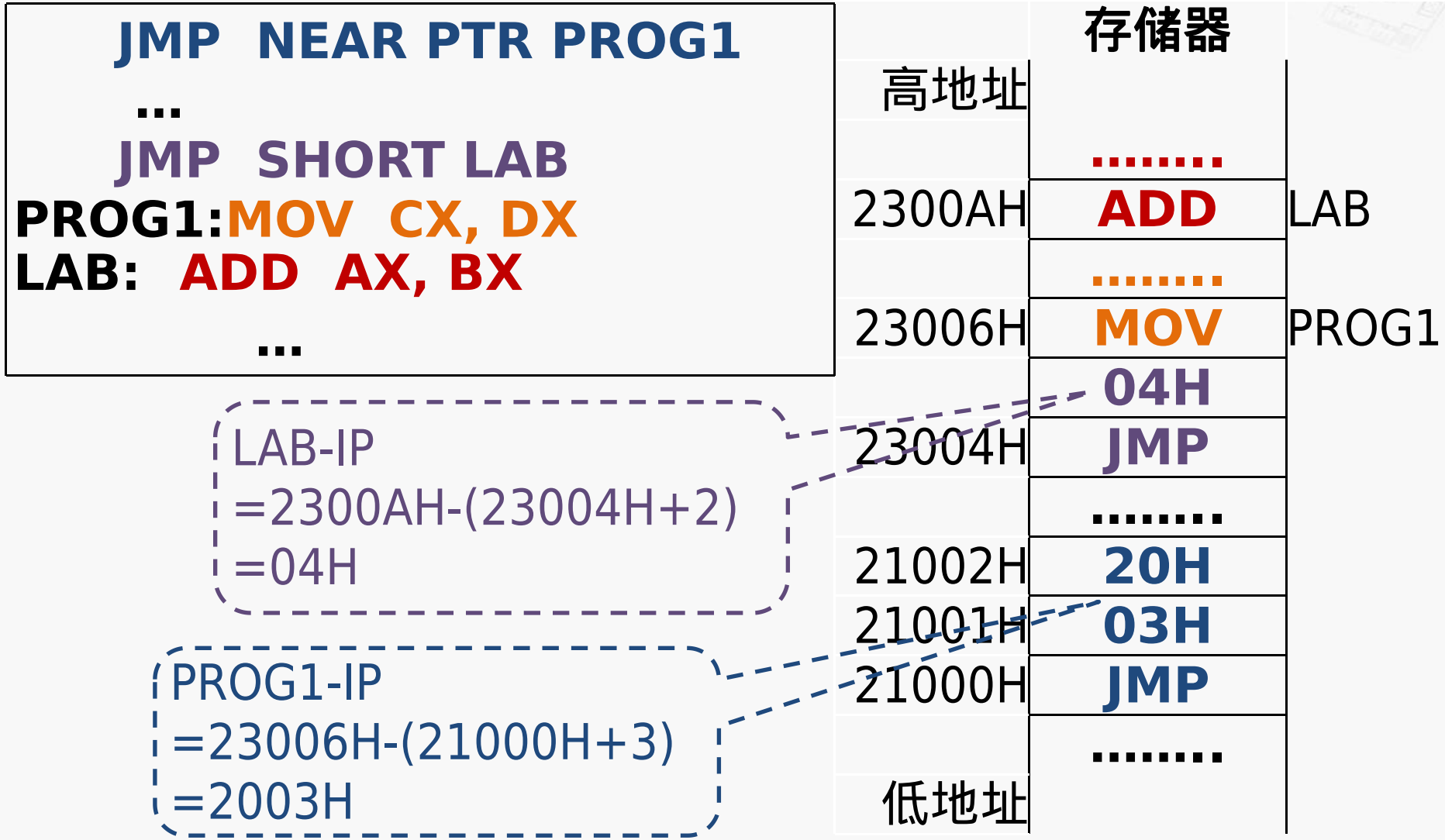
操作码

短转移	EB	8-bit位移量			
近转移	E9	16-bit位移量			
远转移	EA	IP	IP	CS	CS

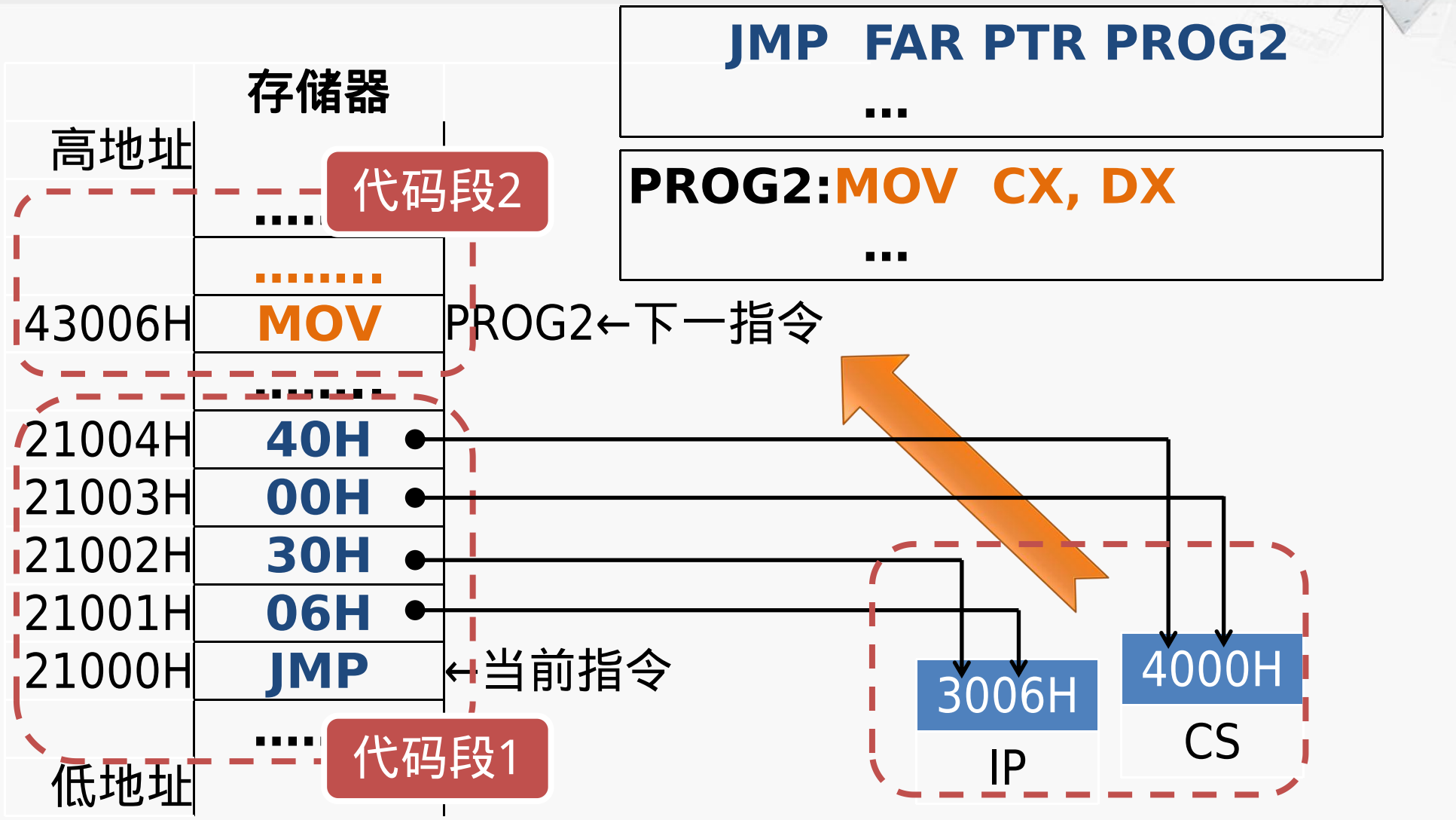
不会考背诵

段内直接转移的执行过程

```
    JMP  NEAR PTR PROG1
      …
    JMP  SHORT LAB
PROG1:MOV  CX, DX
LAB:  ADD  AX, BX
      …
```



段间直接转移的执行过程



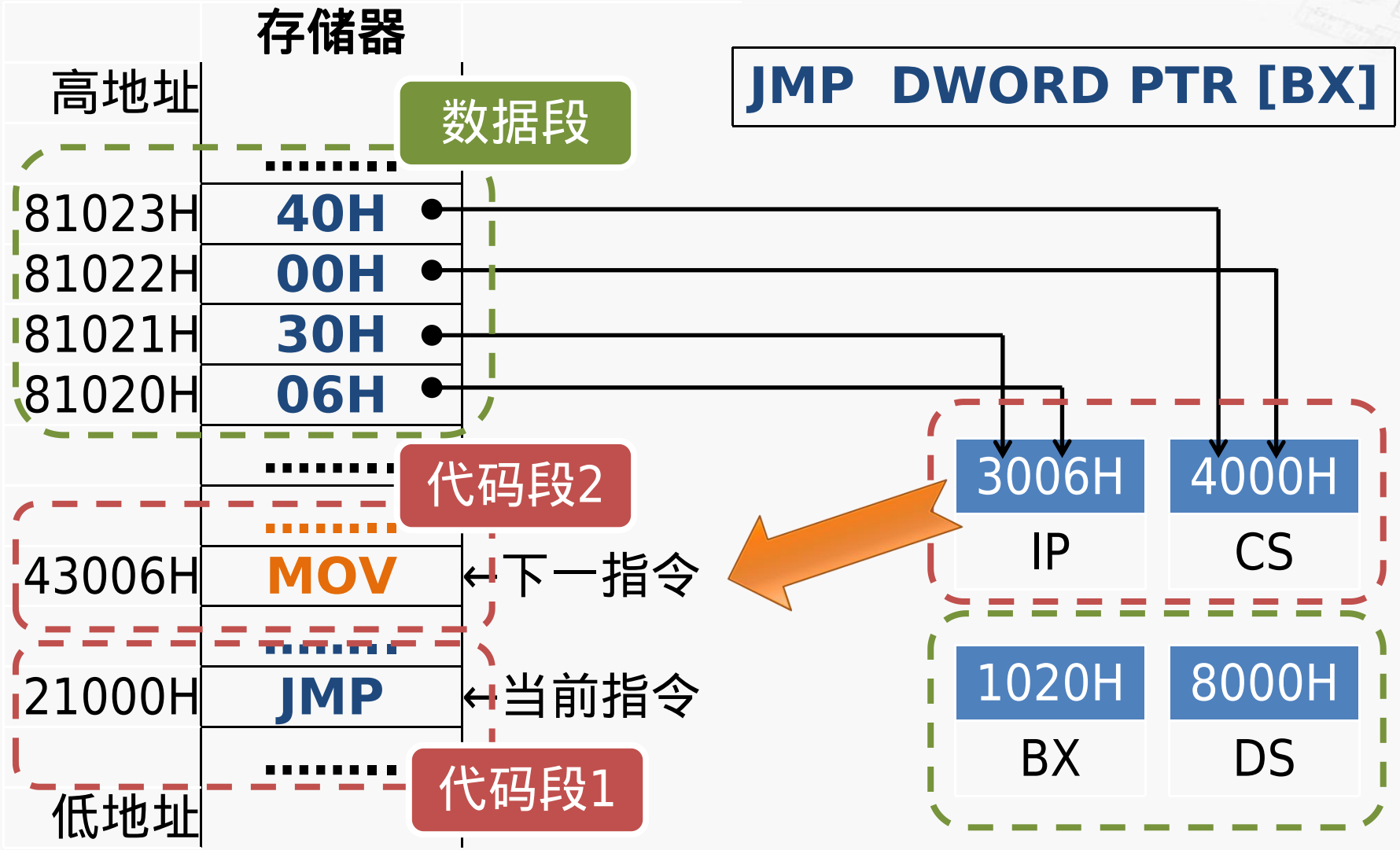
不是保护模式的寻址模式，是实模式的寻址模式

段间间接转移

- 格式：JMP　DWORD PTR　OPR

- 操作
  ① 寻址到OPR指定的存储器单元双字
  ② 将该双字中的低字送到IP寄存器中
  ③ 将该双字中的高字送到CS寄存器中

段间间接转移的执行过程



转移指令的列表（2）

分组		格式		功能	测试条件
条件转移指令	根据某一状态标志转移	JC	LABEL	有进位时转移	CF=1
		JNC	LABEL	无进位时转移	CF=0
		JP/JPE	LABEL	奇偶位为1时转移	PF=1
		JNP/JPO	LABEL	奇偶位为0时转移	PF=0
		JZ/JE	LABEL	为零/相等时转移	ZF=1
		JNZ/JNE	LABEL	不为零/不相等时转移	ZF=0
		JS	LABEL	负数时转移	SF=1
		JNS	LABEL	正数时转移	SF=0
		JO	LABEL	溢出时转移	OF=1
		JNO	LABEL	无溢出时转移	OF=0

转移指令的列表（3）

分组		格式		功能	测试条件
条件转移指令	对无符号数	JB/JNAE	LABEL	低于/不高于等于时转移	CF=1
		JNB/JAE	LABEL	不低于/高于等于时转移	CF=0
		JA/JNBE	LABEL	高于/不低于等于时转移	CF=0且ZF=0
		JNA/JBE	LABEL	不高于/低于等于时转移	CF=1或ZF=1
	对有符号数	JL/JNGE	LABEL	小于/不大于等于时转移	SF≠OF
		JNL/JGE	LABEL	不小于/大于等于时转移	SF=OF
		JG/JNLE	LABEL	大于/不小于等于时转移	ZF=0且SF=OF
		JNG/JLE	LABEL	不大于/小于等于时转移	ZF=1或SF≠OF

条件转移指令的说明

操作

- 根据当前的状态标志位决定是否发生转移

说明

- 一般在影响标志位的算术或逻辑运算指令之后
- 8086中，所有的条件转移都是短转移
  - 同一代码段内，-128~127字节范围内
- 从80386起，条件转移指令可以使用32位的长位移量

转移指令的列表（4）

分组	格式		功能	测试条件
循环控制指令	LOOP	LABEL	循环	CX≠0
	LOOPZ/LOOPE	LABEL	为零/相等时循环	CX≠0且ZF=1
	LOOPNZ/LOOPNE	LABEL	不为零/不相等时循环	CX≠0且ZF=0
	JCXZ	LABEL	CX值为零时循环	CX=0

LOOPNE/LOOPNZ指令说明

LOOPNE/LOOPNZ指令（不为零/不相等时循环）

- 格式：LOOPNE　LABEL
  　　或　LOOPNZ　LABEL

- 操作
  ① CX←CX-1
  ② 若CX≠0且ZF=0，转移到LABEL处继续执行
  否则，结束循环，顺序执行下一条指令

循环控制指令示例

- 在100个字符的字符串中寻找第一个$字符

```
    MOV CX, 100
    MOV SI, 0FFFH
NEXT:INC SI
    CMP BYTE PTR [SI], '$'
    LOOPNZ NEXT
```

在循环出口
分析查找情况

ZF=0 CX=0	查找完毕，在串中没有$字符
ZF=1 CX≠0	已找到$字符，通过CX的内容可确定位置
ZF=1 CX=0	已找到$字符，在串的最后一个字符处

指令分类举例

1.传送类指令
2.运算类指令
3.转移类指令
4.控制类指令

处理器控制指令

- 作用
  - 控制CPU的功能
  - 对标志位进行操作

分组	格式	功能
标志操作指令	STC	把进位标志CF置1
	CLC	把进位标志CF清0
	CMC	把进位标志CF取反
	STD	把方向标志DF置1
	CLD	把方向标志DF清0
	STI	把中断标志IF置1
	CLI	把中断标志IF清0
外同步指令	HLT	暂停
	WAIT	等待
	ESC	交权
	LOCK	封锁总线（指令前缀）
空操作	NOP	空操作

主要内容

通过学习本课程
了解计算机的发展历程，理解计算机的组成原理，掌握计算机的设计方法

Ⅰ x86指令-传送类

Ⅱ x86指令-运算类

Ⅲ x86指令-转移类及其它

Ⅳ x86指令的发展

x86指令系统的发展

- 兼容性
  - 每款处理器包含该系列早期处理器的全部指令
  - 每款处理器包含该系列早期处理器的寄存器和操作方式

- 指令系统的增强和扩充
  - 对已有指令进行功能上的扩展和改进
  - 增加新指令

x86指令增强和扩充举例

处理器	新增指令举例	指令功能扩充举例
80286	8个通用寄存器压栈 PUSHA	立即数的移位次数 SHL AX, 31
80386	符号扩展传送 MOVSX AX, CL	条件转移的位移量 可以是32位
80486	比较并交换 CMPXCHG [DX], CX	
Pentium	处理器特征识别 CPUID	MOV指令的源操作数可以使用控制寄存器CR4

x86指令系统的发展历程（1）

1978年	Intel8086、8088
要点	16位的x86指令
1985年	Intel80386，AMD Am386
要点	扩展为32位的x86指令
1989年	Intel486，AMD Am486
要点	增加x87指令（浮点指令）
1996年	Pentium MMX
要点	增加了MMX指令：一般认为MMX是指Multi Media eXtension，即多媒体扩展指令；AMD称为Matrix Math eXtension。拥有57条多媒体指令（SIMD），不能与浮点数操作同时进行

x86指令系统的发展历程（2）

1999年	Pentium III
要点	SSE：Streaming SIMD Extension，即SIMD扩展指令集，共70条指令。包括50条浮点SIMD运算指令、12条定点MMX指令和8条优化内存数据块传输指令
2001年	Pentium 4（Willamette核心） AMD Opteron（SledgeHammer核心）
要点	SSE2：共144条指令，扩展了MMX（定点）和SSE（浮点）技术
2004年	Pentium 4（Prescott核心） AMD Opteron（Troy核心）（皓龙）
要点	SSE3：在SSE2的基础上增加了13条SIMD指令，目的是改进线程同步和特定应用程序领域，例如媒体和游戏

本讲到此结束，谢谢

欢迎继续学习本课程

计算机组织与体系结构 Computer Architectures

主讲：陆俊林

北京大学 Peking University