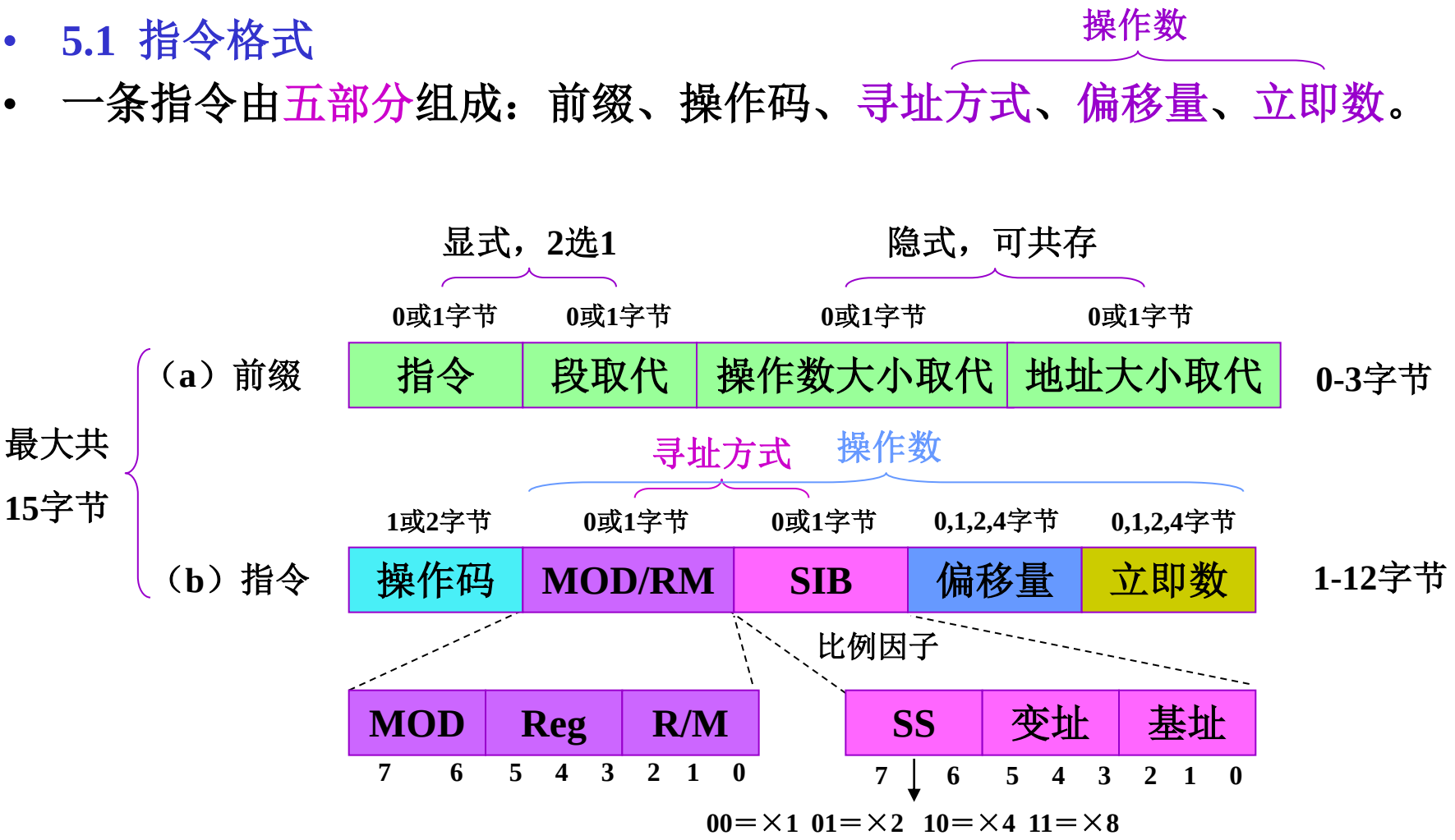
第5章 Pentium微处理器的指令系统

- 5.1 指令格式
- 一条指令由五部分组成：前缀、操作码、寻址方式、偏移量、立即数。

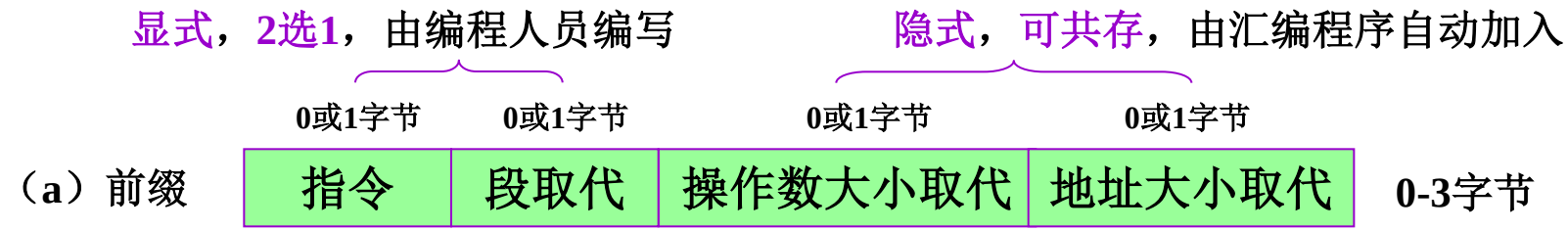


1. 前缀

- 前缀：是对指令的某种限定或说明。

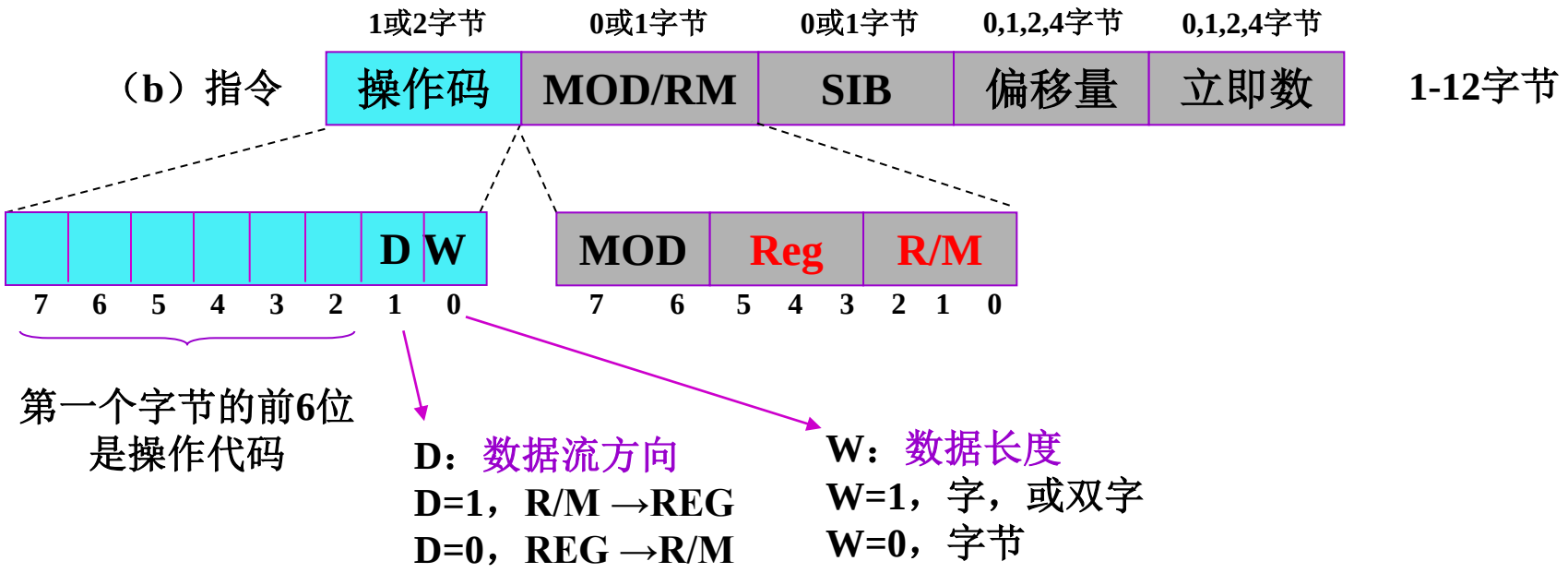
4种
- 指令前缀：说明指令的相关特性。如LOCK、REP、REPE/REPZ、REPNE/REPNZ
- 段超越前缀：用超越段取代默认段。三种情况禁止段超越：串操作（ES）、堆栈指令（SS）、代码段（CS）。
- 操作数大小前缀：改变默认的数据长度。16→32， 32→16。
- 地址大小前缀：改变默认的地址长度。

- 在实地址、虚拟8086方式下，隐含寻址16位。
- 在保护方式下，隐含寻址由段描述符中的D位确定，D=0默认是16位，D=1默认是32位，操作数大小前缀和地址大小前缀用于改变隐含特性。



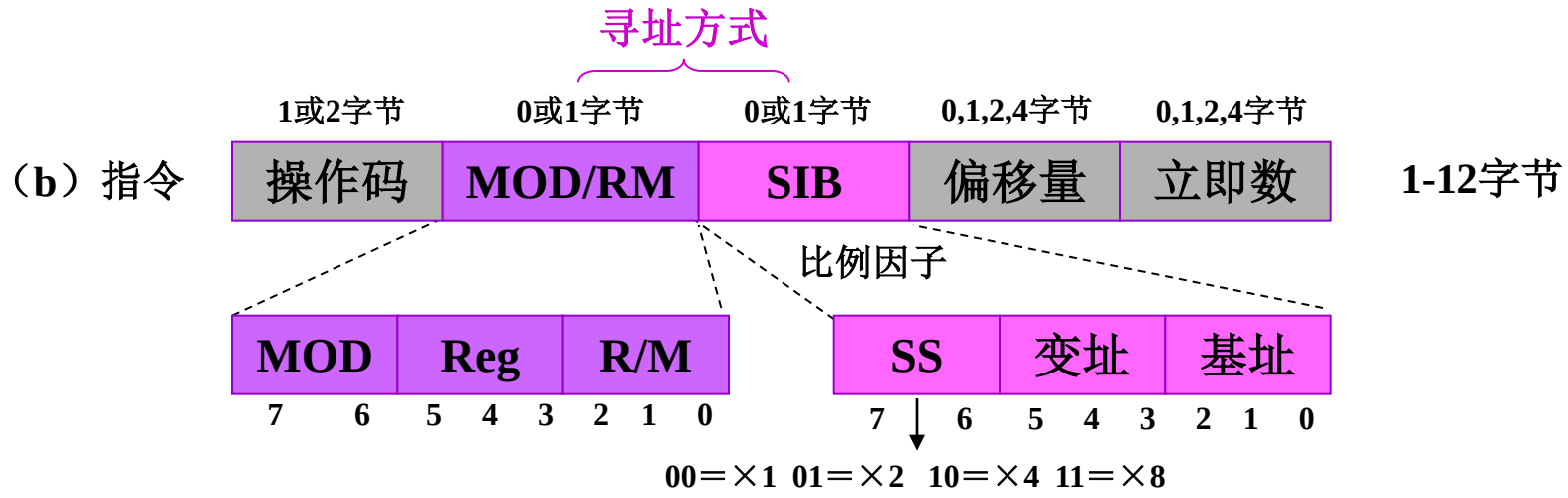
2．操作码

- 操作码：规定了微处理器执行的操作，如加、减、传送等。
- **Pentium微处理器指令的操作码长为1或2个字节。第一个字节的前6位是操作码，其余两位中的D位用来指示数据流的方向、W位用来指示数据的长度是字节还是字或双字。**



3．寻址方式

- 寻址方式：用来指出CPU获取操作数的方式，如寄存器寻址、直接寻址、寄存器间接寻址等。



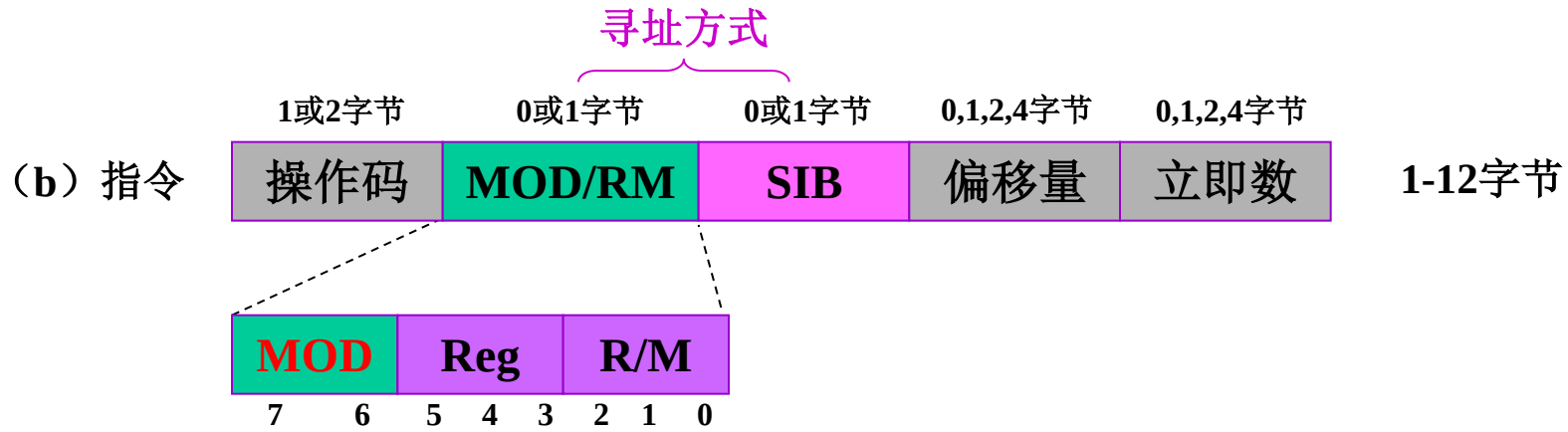
（1）MOD域

- **MOD域：**规定所选指令的寻址方式，选择寻址类型及所选的类型是否有位移量。

表5.1　指令模式中的MOD值		
模式	16位指令功能	32位指令功能
00	没有位移量	没有位移量
01	8位符号扩展的位移量	8位符号扩展的位移量
10	16位的位移量	32位的位移量
11	R/M是寄存器	R/M是寄存器

数据存储器寻址，R/M中选M

寄存器寻址， R/M中选R



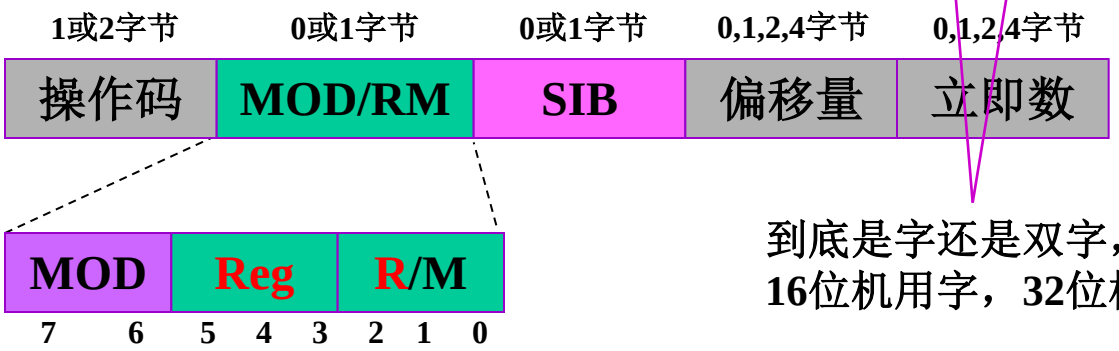
（2）REG域

REG或R/M的代码

MOD=11时，R/M中选R

表5.2 REG域和R／M域的分配（当MOD=11时）

代码	W=0（字节）	W=1（字）	W=1（双字）
000	AL	AX	EAX
001	CL	CX	ECX
010	DL	DX	EDX
011	BL	BX	EBX
100	AH	SP	ESP
101	CH	BP	EBP
110	DH	SI	ESI
111	BH	DI	EDI



（b）指令

1-12字节

到底是字还是双字，依CPU而定，
16位机用字，32位机用双字。

（3）R/M域 -16位指令模式的存储器寻址 8088/86、286

表5.3 16位的R/M存储器寻址方式（MOD≠11时，用于存储器寻址）

R/M \ MOD	00（无位移量）	01（8位位移量）	10（16位位移量）
000	(BX)+(SI) DS	(BX)+(SI)+D8 DS	(BX)+(SI)+D16 DS
001	(BX)+(DI) DS	(BX)+(DI) +D8 DS	(BX)+(DI) +D16 DS
010	(BP)+(SI) SS	(BP)+(SI) +D8 SS	(BP)+(SI) +D16 SS
011	(BP)+(DI) SS	(BP)+(DI) +D8 SS	(BP)+(DI) +D16 SS
100	(SI) DS	(SI) +D8 DS	(SI) +D16 DS
101	(DI) DS	(DI) +D8 DS	(DI) +D16 DS
110	D16（直接寻址） DS	(BP) +D8 SS	(BP) +D16 SS
111	(BX) DS	(BX) +D8 DS	(BX) +D16 DS

（3）R/M域 -32位指令模式的存储器寻址 386以上

表5.4 由R/M选择的32位寻址方式（MOD≠11时，用于存储器寻址）

MOD	R/M	基址	寻址方式	说明
00	000		DS:[EAX] ← 16位寻址没有类似功能	无位移量，或单独使用位移量（单独使用位移量时为直接寻址）。
00	001		DS:[ECX]	
00	010		DS:[EDX]	
00	011		DS:[EBX]	
00	100	000	DS:[EAX+比例变址]	
00	100	001	DS:[ECX+比例变址]	
00	100	010	DS:[EDX+比例变址]	
00	100	011	DS:[EBX+比例变址]	
00	100	100	SS:[EBP+比例变址]	
00	100	101	DS:[32位位移量+比例变址]	
00	100	110	DS:[ESI+比例变址]	
00	100	111	DS:[EDI+比例变址]	
00	101		DS:[32位位移量]	
00	110		DS:[ESI]	
00	111		DS:[EDI]	

（3）R/M域 -32位指令模式的存储器寻址 386以上

表5.4 由R/M选择的32位寻址方式（MOD≠11时，用于存储器寻址）

MOD	R/M	基址	寻址方式	说明
01	000		DS:[EAX+8位位移量]	带8位（字节）位移量
01	001		DS:[ECX+8位位移量]	
01	010		DS:[EDX+8位位移量]	
01	011		DS:[EBX+8位位移量]	
01	100	000	DS:[EAX+比例变址+8位位移量]	
01	100	001	DS:[ECX+比例变址+8位位移量]	
01	100	010	DS:[EDX+比例变址+8位位移量]	
01	100	011	DS:[EBX+比例变址+8位位移量]	
01	100	100	SS:[ESP+比例变址+8位位移量]	
01	100	101	SS:[EBP+比例变址+8位位移量]	
01	100	110	DS:[ESI+比例变址+8位位移量]	
01	100	111	DS:[EDI+比例变址+8位位移量]	
01	101		SS:[EBP+8位位移量]	
01	110		DS:[ESI+8位位移量]	
01	111		DS:[EDI+8位位移量]	

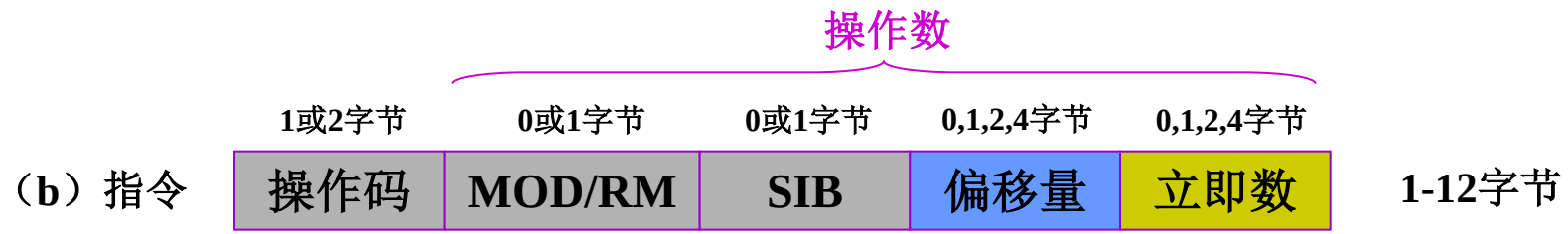
（3）R/M域 -32位指令模式的存储器寻址 386以上

表5.4 由R/M选择的32位寻址方式（MOD≠11时，用于存储器寻址）

MOD	R/M	基址	寻址方式	说明
10	000		DS:[EAX+32位位移量]	带32位（双字）位移量
10	001		DS:[ECX+32位位移量]	
10	010		DS:[EDX+32位位移量]	
10	011		DS:[EBX+32位位移量]	
10	100	000	DS:[EAX+比例变址+32位位移量]	
10	100	001	DS:[ECX+比例变址+32位位移量]	
10	100	010	DS:[EDX+比例变址+32位位移量]	
10	100	011	DS:[EBX+比例变址+32位位移量]	
10	100	100	SS:[ESP+比例变址+32位位移量]	
10	100	101	SS:[EBP+比例变址+32位位移量]	
10	100	110	DS:[ESI+比例变址+32位位移量]	
10	100	111	DS:[EDI+比例变址+32位位移量]	
10	101		SS:[EBP+32位位移量]	
10	110		DS:[ESI+32位位移量]	
10	111		DS:[EDI+32位位移量]	

4. 位/偏移量及立即数

- 位/偏移量：允许指令中直接给出寻址方式所需的位/偏移量。
- 偏移量：直接寻址的地址，无符号数，如 MOV AX，[2000H]
- 位移量：相对寻址中的地址数据，有符号数，如 MOV AX，[SI+6]
- 立即数：允许指令中直接给出操作数。



5.2 寻址方式

- 一条汇编指令要解决2个问题：
- （1）做什么
- （2）对象，或对象来源，即寻址方式

- 寻址方式掌握要点：侧重理解操作数的寻址过程，如何找到一个操作数。

- 有2类寻址：
- （1）对操作数寻址
- （2）对转移地址、调用地址寻址

5.2.1 数据的寻址方式

- **Pentium系统中使用字节（8位）、字（16位）、双字（32位）和4倍字（64位）。采用低端存储方式。**

- **与数据有关的寻址方式有11种：**

- 立即寻址
- 寄存器寻址
- 存储器寻址
  - 直接寻址
  - 寄存器间接寻址
  - 寄存器相对寻址
  - 基址变址寻址
  - 相对基址变址寻址
  - 比例变址寻址
  - 相对比例变址寻址
  - 基址比例变址寻址
  - 相对基址比例变址寻址

1．立即寻址

- 立即寻址：操作数是指令的一部分。完整地取出该条指令之后也就获得了操作数。这种操作数称为立即数。
- 主要用于对寄存器赋值，速度快。
- 汇编语言规定：立即数必须以数字开头，以字母开头的十六进制数前必须以数字0做前缀；数制用后缀表示，B表示二进制数，H表示十六进制数，D或者缺省表示十进制数，Q表示八进制数，程序员可以用自己习惯的计数制书写立即数。汇编程序在汇编时，对于不同进制的立即数一律汇编成等值的二进制数，有符号数以补码表示。
- 此外，立即数还可以是用+、—、×、／表示的算术表达式。汇编程序按照先乘除后加减的规则自动计算，也可以用圆括号改变运算顺序。

- 立即寻址的例子：

  MOV　AL，0101010lB

  MOV　BX，1234H

  MOV　EAX，12315678H

2. 寄存器寻址

- 寄存器寻址：操作数在CPU的某个寄存器中，指令指定寄存器号。
- 寄存器寻址是最通用的数据寻址方式。对于8位操作数，寄存器有AH、AL、BH、BL、CH、CL、DH和DL；对于16位操作数，寄存器可以是AX、BX、CX、DX、SP、BP、SI和DI；在80386以上的微处理器中，还可以是32位操作数，寄存器有EAX、EBX、ECX、EDX、ESP、EBP、EDI和ESI。
- 有些用寄存器寻址的MOV、PUSH和POP指令可寻址16位的段寄存器(CS、ES、DS、SS、FS和GS)。

- 寄存器寻址的例子：

```
MOV   DS, AX
MOV   EAX, ECX
MOV   DL, BH
```

3．存储器寻址

- 存储器寻址：操作数在存储器中。

- 存储器地址：
  - 物理地址
  - 逻辑地址：　　段基址:偏移量

- CPU要访问存储器操作数，必须先计算存储器的物理地址。
- 在实地址和保护方式下，段基址的获取方式不同，但偏移量的获取思想基本一致。
- 在存储器寻址方式中，要解决的问题是如何取得操作数的偏移地址。

- 有效地址EA：Effective Address，在8086～Pentium微处理器里，把操作数的偏移地址称为有效地址EA。

有效地址的计算方法

- 有效地址的计算公式：

EA＝基址 ＋（变址×比例因子）＋ 位移量

- 基址：基址寄存器中的内容。基址寄存器通常用于编译程序指向局部变量区、数据段中数组或字符串的首地址。
- 变址：变址寄存器中的内容。变址寄存器用于访问数组或字符串的元素。
- 比例因子：32位机的寻址方式。寻址中，用变址寄存器的内容乘以比例因子得到变址值。该方式对访问数组特别有用。
- 位移量：指令中的一个数，但不是立即数，而是一个地址。
- **8086/286是16位寻址，80386及以后机型是32位寻址，也可用16位寻址。**

四种成分	16位寻址	32位寻址
位移量	0、8、16位	0、8、32位
基址寄存器	BX、BP	任何32位通用寄存器
变址寄存器	SI、DI	除ESP以外的任何32位通用寄存器
比例因子	无	1、2、4、8

9种存储器寻址方式

寻址方式	基址	变址	比例因子	位移量
1．直接寻址				√
2．寄存器间接寻址	√*	√*		
3．寄存器相对寻址	√*	√*		√
4．基址变址寻址	√	√		
5．相对基址变址寻址	√	√		√
6．比例变址寻址		√	√	
7．相对比例变址寻址		√	√	√
8．基址比例变址寻址	√	√	√	
9．相对基址比例变址寻址	√	√	√	√

一对，寻址及相对寻址

四种成分	16位寻址	32位寻址
位移量	0、8、16位	0、8、32位
基址寄存器	BX、BP	任何32位通用寄存器
变址寄存器	SI、DI	除ESP以外的任何32位通用寄存器
比例因子	无	1、2、4、8

任取一个

存储器寻址方式举例

寻址方式	有效地址EA如何计算？	基址	变址	比例因子	位移量
1．直接寻址	MOV BX，[1234H]				√
2．寄存器间接寻址	MOV AL，[BX]	√*	√*		
3．寄存器相对寻址	MOV BX，[SI+40H]	√*	√*		√
4．基址变址寻址	MOV DX，[BX+SI]	√	√		
5．相对基址变址寻址	MOV AX，[BP+DI+2]	√	√		√
6．比例变址寻址	MOV EAX，[4×ECX]		√	√	
7．相对比例变址寻址	MOV EDX，[4×ECX＋5]		√	√	√
8．基址比例变址寻址	MOV AL，[EBX+2×EDI]	√	√	√	
9．相对基址比例变址寻址	MOV AL，[EBX+2×EDI-2]	√	√	√	√

四种成分	16位寻址	32位寻址
位移量	0、8、16位	0、8、32位
基址寄存器	BX、BP	任何32位通用寄存器
变址寄存器	SI、DI	除ESP以外的任何32位通用寄存器
比例因子	无	1、2、4、8

寻址方式的段约定和段超越

- MOV　AL，DS：[BP]　　；用BP间址访问数据段
- MOV　AL，ES：[BP]　　；用BP间址访问ES附加段
- MOV　AL，FS：[EBP]　　；用EBP间址访问FS附加段
- MOV　AL，CS：[BX]　　；用BX间址访问代码段
- MOV　AL，ES：[SI+5]　　；用SI变址寻址，访问ES附加段
- MOV　AL，GS：[EAX+10]；用EAX基址寻址，访问GS附加段
- 为提高程序的可读性和效率，要尽量少使用段超越。

存储器操作类型	隐含段	超越段	段内偏移量
取指令	CS	无	IP
堆栈操作	SS	无	SP
数据变量	DS	CS、SS、ES	有效地址EA
源串变量	DS	CS、SS、ES	SI
目的串变量	ES	无	DI
基址BP指针	SS	CS、SS、ES	有效地址EA

5.2.2 转移地址的寻址方式

- 转移地址寻址：用于确定转移指令和CALL指令的转向地址。
- 转移指令使程序不再顺序执行，而是按指令中给出的操作数转移到相应的目的地址。
- 转移地址：转移指令中的操作数是转移的目的地址，称为转移地址。

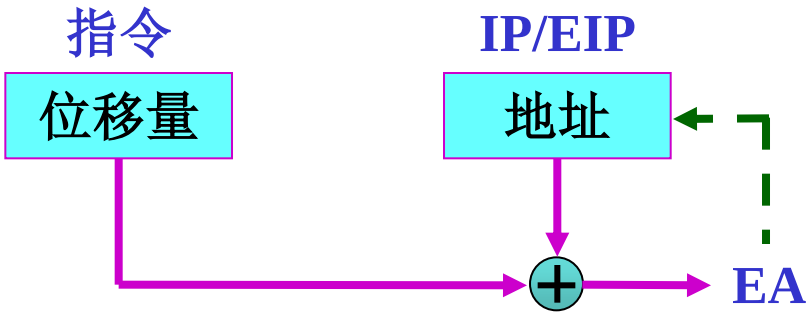
转移指令
调用指令
- 段内（IP/EIP）
  - 直接（相对）
  - 间接
- 段间（CS:IP/EIP）
  - 直接
  - 间接

1. 段内相对寻址

- 位移量+IP=EA→IP/EIP
- 位移量是相对值，所以称为段内相对寻址。
- 因为是相对值，所以便于程序再定位。
- 用于条件转移、无条件转移，条件转移只能用此方式。

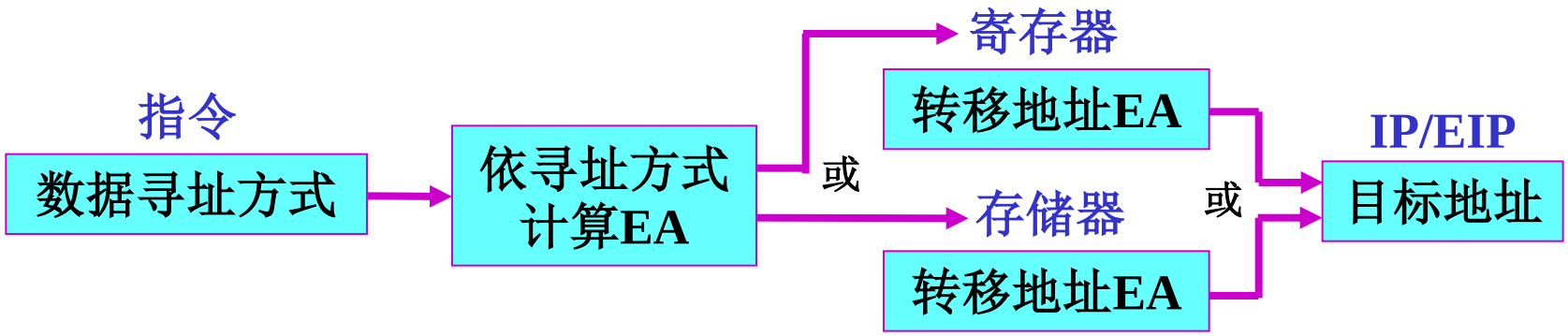
短跳转：JMP SHORT AA9 ；位移量 8位, -128～+127

近跳转：JMP NEAR PTR AA8 ；位移量 16位±32K，32位±2G



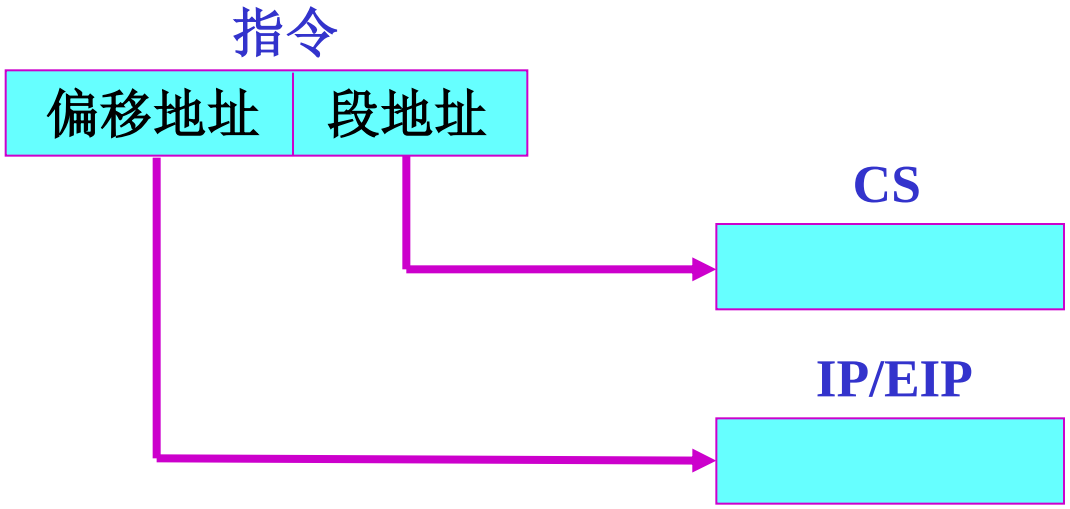
2. 段内间接寻址

- 寄存器（存储器）内容=EA→IP/EIP
- 这种寻址方式不能用于条件转移指令。
- **JMP BX**
- **JMP WORD PTR[BX+AA7]**
- **JMP ECX**



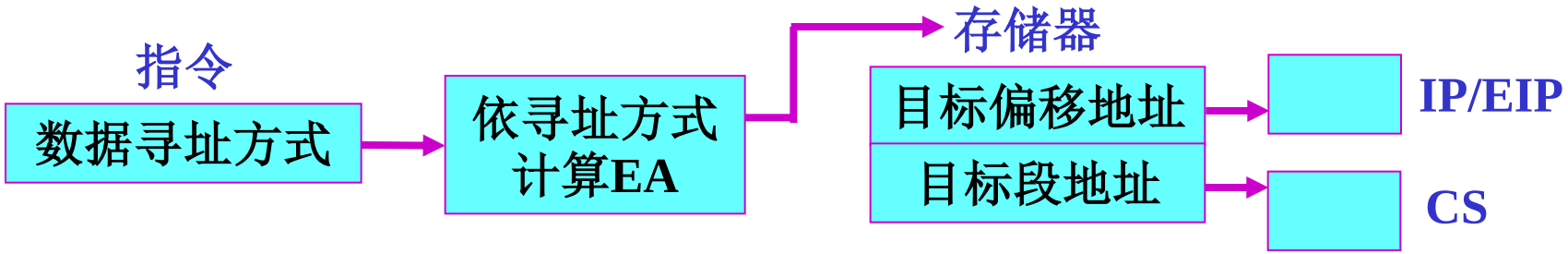
3. 段间直接寻址

- 目标段地址:偏移量=CS:IP/EIP
- **JMP FAR PTR AA6**



4. 段间间接寻址

- 连续的存储器单元内容= CS:IP/EIP
- **JMP DWORD PTR [SI] ； [SI]指向的字送入IP，[SI+2] 指向的字送入CS。**



5.2.3 堆栈地址寻址

- 堆栈：以“先进后出”方式工作的一个特定的存储区。
- 一端固定，称为栈底；另一端浮动，称为栈顶，只有一个出入口。
- 堆栈作用：保存传递参数、现场参数、寄存器内容、返回地址。
- 80x86规定：
- （1）堆栈向小地址方向增长。
- （2）必须使用堆栈段SS。
- （3） PUSH指令压入数据时，先修改指针，再按照指针指示的单元存入数据。
- （4） POP指令弹出数据时，先按照指针指示的单元取出数据，再修改指针。
- （5）压入和弹出的数据类型（数据长度）不同，堆栈指针修改的数值也不同。

- 16位堆栈地址操作，还是32位堆栈地址操作，由堆栈段SS寄存器内的数据段描述符的属性说明，确定用SP还是ESP，是一种隐含属性。

5.3 指令系统

- 指令系统：指令的集合。软硬件交界面。
- 指令系统掌握要点：侧重掌握指令的功能和使用方法，为后续程序设计打基础。

- 8086-110条指令
- 80286-143条指令
- 80386-154条指令
- 80486-160条指令
- Pentium -165条指令

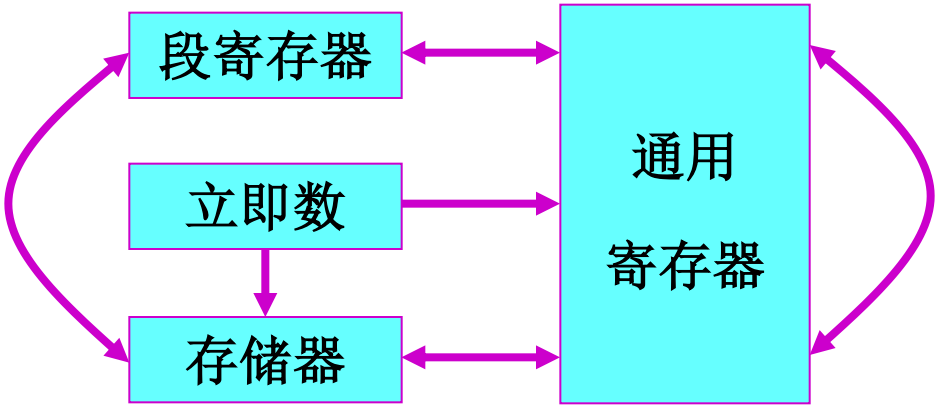
- Pentium指令系统按功能分为10类：

  数据传送指令、算术运算指令、BCD码调整指令、逻辑运算指令、位处理指令、控制转移指令、条件设置指令、串操作指令、处理器控制指令、保护模式系统控制指令。

5.3.1 数据传送指令

- 数据传送指令功能：源 → 目。
- 指令执行后，源操作数不变，不影响状态标志。

- 1. MOV 传送（move）
- 格式：MOV DST，SRC ； (DST) ← (SRC)
- 式中DST（Destination）为目的操作数，SRC（Source）为源操作数。
- 源、目长度相等。 CS可出不可入。
- MOV WORD PTR [BX]，10H；立即数10H送入BX指向的字存储单元
- 例：……



数据传送指令-1

- **4．PUSH 进栈 （push onto stack）**
- **格式：PUSH SRC ；堆栈 ← (SRC)**
- **操作：**

16位指令： (SP) ← (SP) - 2

((SP)+1, (SP)) ← (SRC)

32位指令： (ESP) ← (ESP) - 4

((ESP)+3, (ESP)+2, (ESP)+1, (ESP)) ← (SRC)

- **类型：**

数据传送指令-2

- **5．POP 出栈（pop from the stack）**
- **格式：POP DST ；(DST) ← 堆栈栈顶内容**
- **操作：**

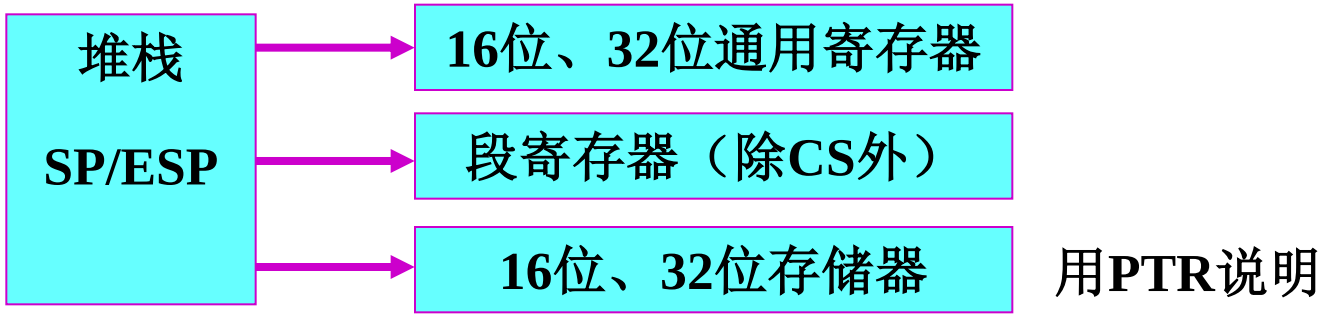
16位指令： **(DST) ← ((SP)+1，(SP))**

(SP) ← (SP)+2

32位指令： **(DST) ← ((ESP)+3，(ESP)+2，(ESP)+1，(ESP))**

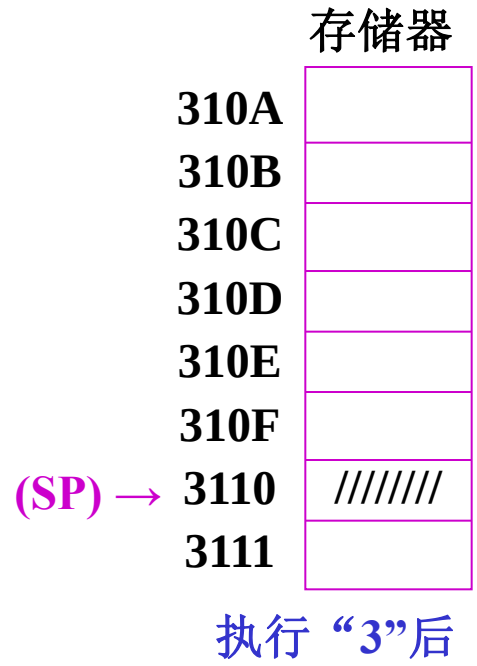
(ESP) ← (ESP)+4

- **类型：**

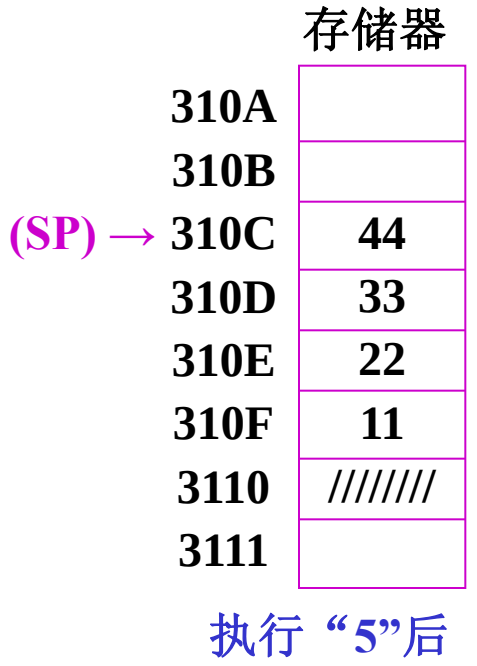


堆栈指令举例

- 1 MOV AX, 1122H
- 2 MOV BX, 3344H
- 3 MOV SP, 3110H
- 4 PUSH AX
- 5 PUSH BX
- 6 MOV AX, 0
- 7 MOV BX, 0
- 8 POP BX
- 9 POP AX



执行“3”后



执行“5”后

- 问题：1）运行结果　2）进、出栈平衡　3）交换

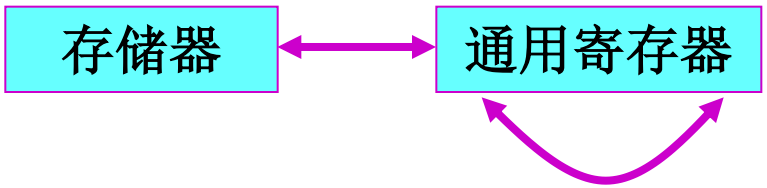
数据传送指令-3

- **6． PUSHF/ POPF 16位标志寄存器进栈/出栈**
  （push the flags/pop the flags）
- **格式：**

  PUSHF ；标志寄存器低16位压入堆栈

  POPF ；从栈顶弹出2个字节送标志寄存器低16位

- **10．XCHG 交换（exchange）**
- **格式：XCHG OPR1, OPR2 ；(OPR1) ←→ (OPR2)**
- **类型：**



数据传送指令-4

- **12．CBW 字节转换为字（convert byte to word）**
- **格式：CBW　　；　AX← AL符号扩展**

- **13．CWD 字转换为双字（convert word to double word）**
- **格式：CWD　　；　DX:AX ← AX符号扩展**

数据传送指令-5

- **16．LEA 有效地址送寄存器（load effective address）**
- **格式：LEA REG，SRC ； (REG) ← (SRC)的有效地址**
- **类型：**

16位、32位通用寄存器 ← **16位、32位存储器有效地址**

- **操作规则：**

操作数长度（寄存器）	地址长度（EA）	操作规则
16	16	16位EA→16位REG
16	32	32位EA的低16位→16位REG
32	16	16位EA，零扩展→32位REG
32	32	32位EA→32位REG

```
1  MOV AX, MEM    ;AX=3412H
2  LEA  AX, MEM   ;AX=3100H
3  MOV  BX, 2000H
4  MOV  DS, BX
5  MOV  SI, 1000H
6  MOV  AX, [BX+SI+102H]    ; AX=7856H
7  LEA   BX, [BX+SI+102H]   ; BX=3102H
```

MEM	2000:3100	12H
	3101	34H
	3102	56H
	3103	78H
	3104	

数据传送指令-6

- **17．LDS、LES、LFS、LGS和LSS 指针送寄存器和段寄存器**

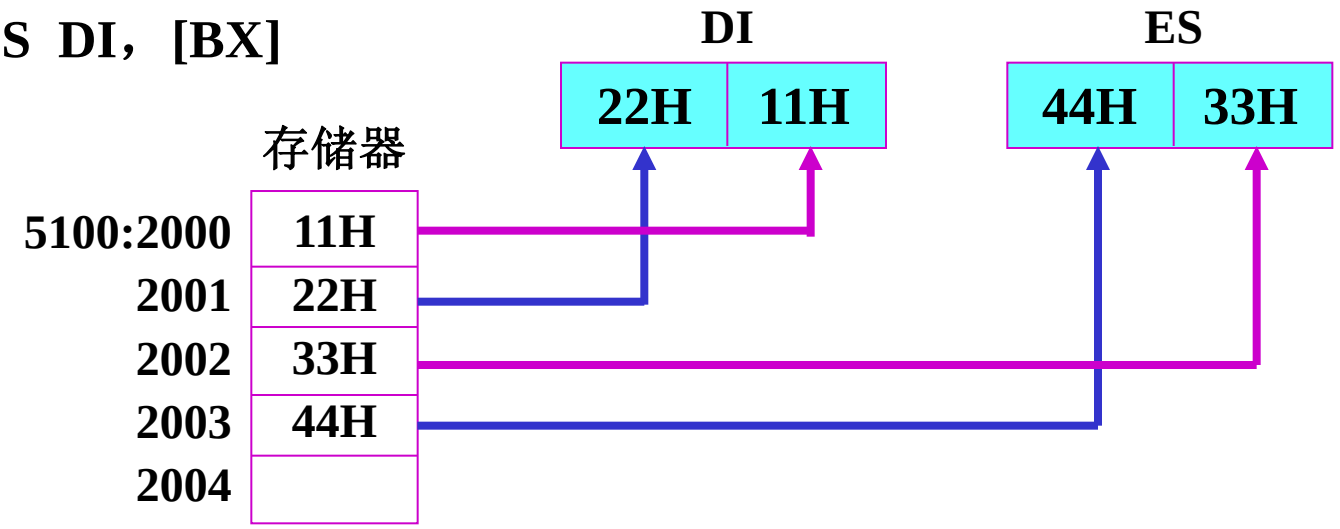
  （load ds with pointer）

- **格式：LDS　REG，SRC**
- **功能：(REG) ← [SRC]**

  (DS) ← [SRC+2]　（16位EA）

  (DS) ← [SRC+4]　（32位EA）
- **例：已知 (DS)＝5100H，(BX)＝2000H**

LES　DI，[BX]

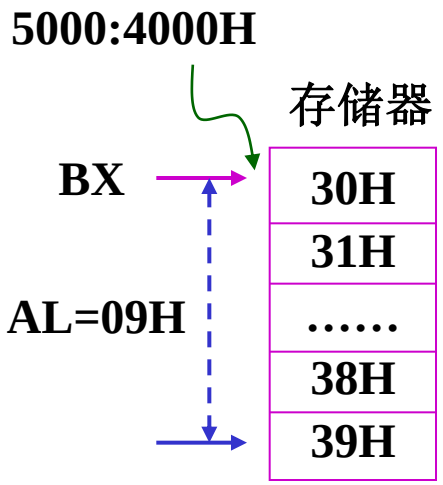


数据传送指令-7

- **18. LAHF 标志送AH （load AH with flags）**
- **格式：LAHF ； AH ← flags低8位**

- **19．SAHF AH送标志寄存器（store AH into flags）**
- **格式：SAHF ； flags低8位 ← AH**

- **20．XLAT 换码（translate）**
- **格式：XLAT**
- **功能：完成代码转换，偏移量8位，表长≤256**
  16位指令 （AL）←[(BX)+(AL)]
  32位指令 （AL）←[(EBX)+(AL)]



- **例：非压缩BCD码转换成ASCII码，09H→39H**
  已知(DS)＝5000H，(BX)＝4000H，(AL)＝09H
  执行 XLAT ；AL＝39H

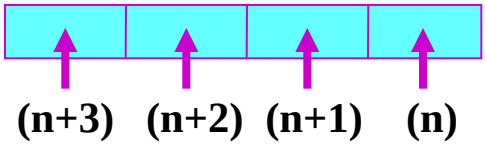
数据传送指令-8

- 21．IN/OUT 输入/输出 （不需要段地址）
- （1）IN 输入指令（input）
- 1）直接寻址（端口地址≤255）

  IN AL, n ； 字节，(AL) ←(n)

  IN AX, n ； 字， (AH) ←(n+1)，(AL) ←(n)

  IN EAX, n ； 双字， EAX

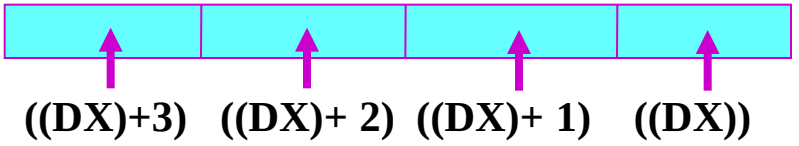


- 2）间接寻址（全部端口地址可用）

  IN AL, DX ； 字节，(AL) ←((DX))

  IN AX, DX ； 字， (AH) ← ((DX+1))，(AL) ← ((DX))

  IN EAX, DX ； 双字， EAX



数据传送指令-9

- （2）OUT 输出指令（output）
- 1）直接寻址（端口地址≤255）

  OUT n, AL ；字节， (n)← (AL)

  OUT n, AX ；字

  OUT n, EAX ；双字

- 2）间接寻址（全部端口地址可用）

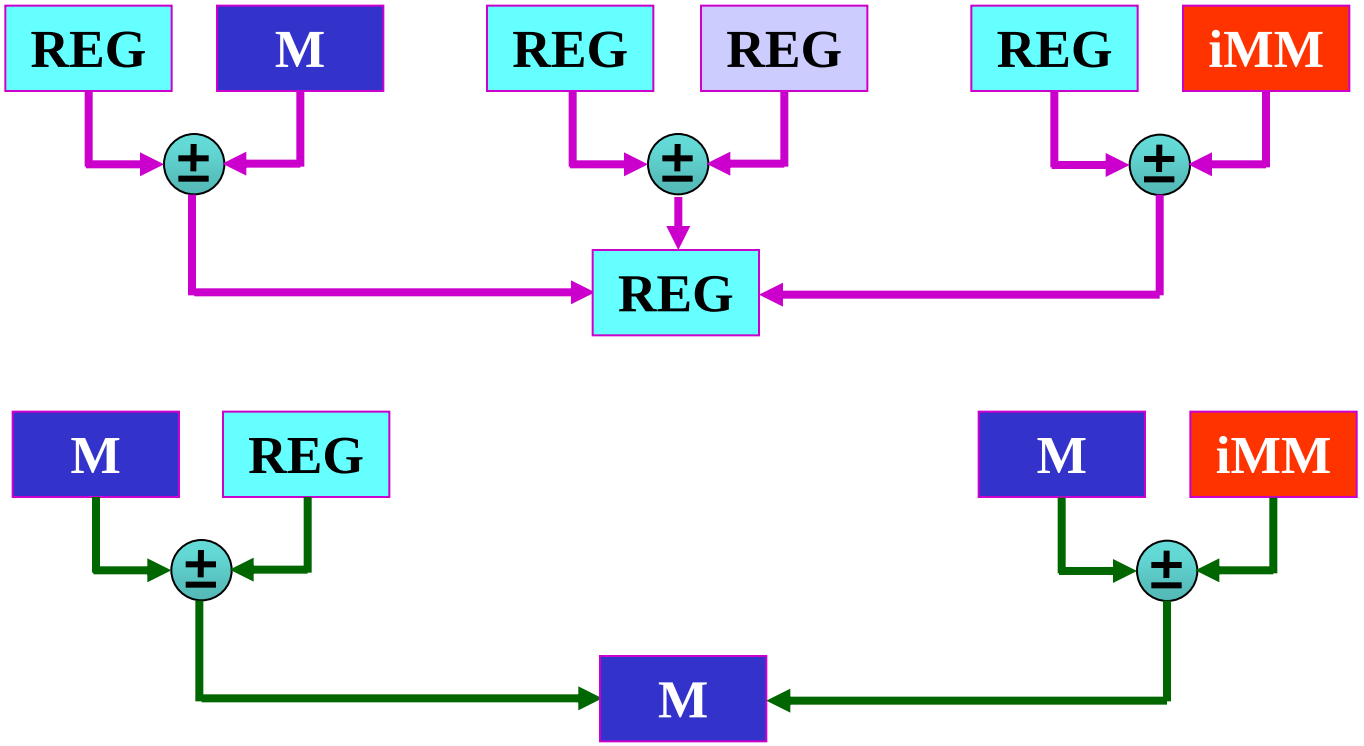
  OUT DX, AL ；字节， ((DX)) ← (AL)

  OUT DX, AX ；字

  OUT DX, EAX ；双字

5.3.2 算术运算指令

- (1) ADD 加法（add）
- 格式：ADD DST, SRC ；（DST）← (SRC)+(DST)
- 类型：影响CF、PF、AF、ZF、SF、OF。可进行8、16、32位操作。



算术运算指令-1

- **(2) ADC 带进位加法（add with carry）**
- **格式：ADC DST, SRC ；（DST）← (SRC)+(DST)＋CF**

- **(3) SUB 减法（subtract）**
- **格式：SUB DST, SRC ；（DST）← (DST) - (SRC)**

- **(4) SBB 带借位减法（subtract with borrow）**
- **格式：SBB DST, SRC ；（DST）← (DST) - (SRC) – CF**

- **例：**

```
MOV  AX,  7626H
MOV  BX,  6615H
ADD  AX,  BX        ; AX=0DC3BH
```

CF=? PF=? AF=? ZF=? SF=? OF=?

算术运算指令-2

- (5) INC 加1（increment）
- 格式：INC OPR ；(OPR) ← (OPR)+1
- (6) DEC 减1（decrement）
- 格式：DEC OPR ；（OPR）← (OPR)-1
- (7) NEG 求补（negate）
- 格式：NEG OPR ；(OPR) ←+1
- 3条指令共性说明：
- 1）目的操作数：通用寄存器、存储器单元，可以是8位、16位或32位。
- 2）执行INC、DEC指令后，影响AF、OF、PF、SF、ZF，但对CF没有影响。
- 3）执行NEG指令后，影响AF、CF、OF、PF、SF、ZF这6个标志位。
- 4）如果原来的操作数为80H(即-128)，执行NEG指令后，仍为80H，此时OF=1，溢出。
- 5）操作数=0时，在执行NEG指令后，CF=0；其他操作数，执行NEG指令后，CF总为1。

算术运算指令-3

- **(8) CMP 比较（compare）**
- **格式：CMP OPR1, OPR2 ；(OPR1) - (OPR2)**

- **(12) MUL 无符号数乘法（unsigned multiple）**
- **格式：MUL SRC**
- **功能：**

  字节乘 (AX) ← (AL) × (SRC)

  字乘 (DX:AX) ← (AX) × (SRC)

  双字乘 (EDX:EAX) ← (EAX) × (SRC)
- **说明：**

  1）乘数和被乘数必须等长。

  2）SRC：通用寄存器、存储器单元。

  3）被乘数：默认AL、AX、EAX之一。

  4）若乘积的高半部分结果为0，则CF=0、OF=0；否则，CF=1、OF=1。而AF、PF、ZF、SF无定义，即状态不定。

算术运算指令-4

- (13) IMUL 带符号数乘法（ signed multiple ）
- 格式1：IMUL　SRC
- 功能：　　字节乘　(AX) ← (AL)×(SRC)
  　　字乘　　(DX:AX) ← (AX)×(SRC)
  　　双字乘　(EDX:EAX) ← (EAX)×(SRC)
- (14) DIV 无符号数除法（unsigned divide）
- 格式：DIV　SRC
- 功能：字节除 (AL) ←(AX)/(SRC)的商，(AH) ←(AX)/(SRC)的余数
  　字除　(AX) ←(DX:AX)/(SRC)的商，(DX) ←(DX:AX)/(SRC)的余数
  　双字除(EAX) ←(EDX:EAX)/(SRC)的商，(EDX) ←(EDX:EAX)/(SRC)的余数
- 说明：被除数默认在AX、DX:AX、EDX:EAX中。 SRC是通用寄存器、存储器单元。所有标志位无定义。
- (15) IDIV有符号数除法（signed divide）
- 格式：IDIV　SRC
- 功能: 与DIV指令相同

5.3.3 BCD码调整指令

- (1) DAA 压缩BCD码加法调整（decimal adjust for addition）
- 格式：DAA
- 功能：

  1）如果AL的低4位大于9或AF＝1，则(AL)+6 → (AL) 和 1→AF。

  2）如果AL的高4位大于9或CF＝1，则(AL)+60H → (AL) 和 1→CF。
- 说明：

  1）只对AL调整。

  2） OF无定义，其他5个状态受影响。
- 例 5.3.12

```
MOV  AL，54H ；54H代表十进制数54
MOV  BL，37H ；37H代表十进制数37
ADD  AL，BL  ；AL中的和为十六进制数8BH
DAA          ；AL=91H，AF=1，CF=0
```

BCD码调整指令-1

- (2) DAS 压缩BCD码减法调整（decimal adjust for subtraction）
- 格式：DAS
- 功能：

  1）如果AL的低4位大于9或AF=1，则(AL)-6 → (AL) 和 1→AF。

  2）如果AL的高4位大于9或CF=1，则(AL)-60H → (AL) 和 1→CF。

- (3) AAA 非压缩BCD码加法调整（ ASCII adjust for addition）
- 格式：AAA
- 功能：

  ① 如果AL的低4位小于等于9，并且AF=0，则转步骤③；

  ② (AL)+6→AL，1→AF，(AH)+1→AH；

  ③ AL高4位清0；

  ④ AF→CF。
- 说明：影响AF和CF，而PF、SF、ZF、OF无定义。

BCD码调整指令-2

- (4) AAS 非压缩BCD码减法调整（ ASCII adjust for subtraction ）
- 格式：AAS
- 功能：

  ① 如果AL的低4位小于等于9，并且AF=0，则转步骤③；

  ② (AL)-6 →AL，1→AF，(AH)-1→AH；

  ③ AL高4位清0；

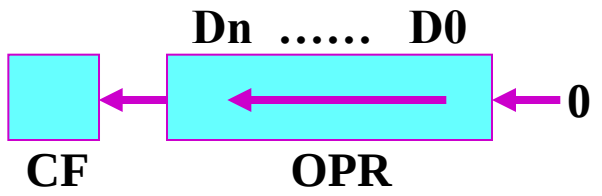
  ④ AF→CF。
- (5) AAM 非压缩BCD码乘法调整（ ASCII adjust for multiplication）
- 格式：AAM
- 功能：(AL)/0AH → (AH)，余数 → (AL)
- 说明：影响SF、ZF、PF，而OF、AF、CF无定义。
- (6) AAD 非压缩BCD码除法调整（ASCII adjust for division）
- 格式：AAD
- 功能：(AH)×10＋(AL) → (AL)，0 → (AH)
- 说明：先调整后运算。影响SF、ZF、PF，而OF、CF和AF无定义。

5.3.4 逻辑运算指令

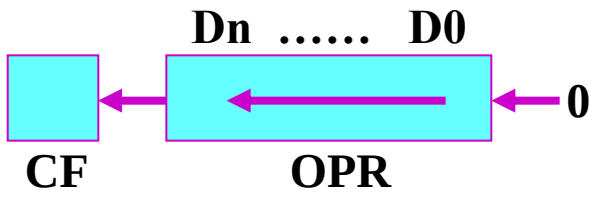
- (1) AND 按位逻辑与运算
- 格式：AND DST，SRC ；(DST)←(DST)∧(SRC)
- (2) OR 按位逻辑或运算
- 格式：OR DST，SRC ；(DST)←(DST)∨(SRC)
- (3) XOR 按位逻辑异或运算（exclusive or）
- 格式：XOR DST，SRC ；(DST)←(DST)⊕(SRC)
- (4) TEST 按位逻辑比较运算
- 格式：TEST OPRl，OPR2 ；(OPRl)∧(OPR2)
- 4条指令类型：同ADD。
- 4条指令说明：CF、OF置0，影响SF、ZF、PF，AF无定义。
- (5) NOT按位取反
- 格式：NOT OPR ；(OPR)← $\overline{\text{OPR}}$
- 类型：通用寄存器、存储器单元，8位、16位、32位。
- 说明：不影响标志位。

5.3.5 位处理指令

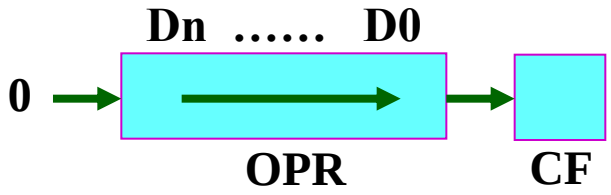
- (7) SHL 逻辑左移（shift logical left）
- 格式： SHL OPR，CNT



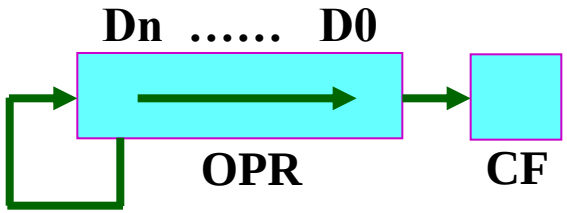
- (8) SAL 算术左移（shift arithmetic left）
- 格式：SAL OPR，CNT



- (9) SHR 逻辑右移（shift logical right）
- 格式：SHR OPR，CNT



- (10) SAR 算术右移（shift arithmetic right）
- 格式：SAR OPR，CNT



- 移位指令说明：

1）OPR：通用寄存器、存储器单元。 8位、16位、32位。

2）CNT：8位立即数（ 1-31，8086中只能为1）、CL。

3）CF依操作设置。OF当CNT=1时有效。

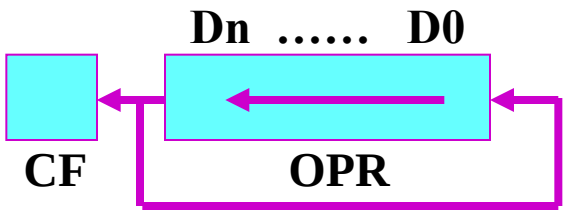
移位指令举例

- 1）AL×10
- MOV AH, 0 ；AL扩展为字，无符号
- SAL AX, 1 ；2X
- MOV BX, AX
- SAL AX, 1 ；4X
- SAL AX, 1 ；8X
- ADD AX, BX ；8X+2X

- 2）ASCII码←AL组合BCD
- MOV AH, AL ；AL组合BCD，保存
- AND AL, 0FH
- OR AL, 30H ；低位ASCII码在AL中
- MOV CL, 4
- SHR AH, CL
- OR AH, 30H ；高位ASCII码在AH中。AX为2个ASCII码

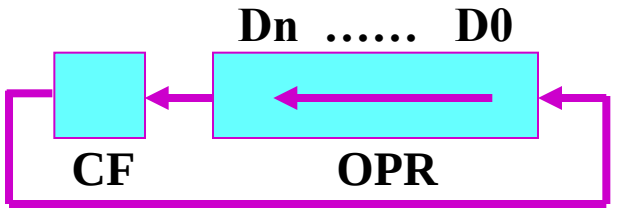
循环指令

- (11) ROL 循环左移（rotate left ）
- 格式：ROL OPR, CNT

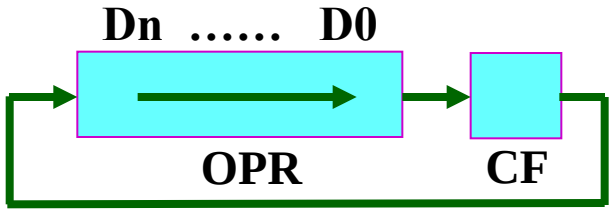


- (12) ROR 循环右移（rotate right）
- 格式：ROR OPR, CNT

- (13) RCL 带进位循环左移（rotate left through carry）
- 格式：RCL OPR, CNT



- (14) RCR 带进位循环右移（rotate right through carry ）
- 格式：RCR OPR, CNT



5.3.6 控制转移指令

- 1. 无条件段内相对短转移（jump）
- 格式：JMP　SHORT OPR
- 功能：　(IP) ←(IP)+8位位移量
  或　(EIP) ←(EIP)+8位位移量
- 说明：SHORT是短转移属性描述符，8位位移量是一个带符号数，范围是-128到+127字节。

- 2. 无条件段内相对近转移
- 格式：JMP　NEAR PTR OPR
- 功能：　(1P) ←(1P)+16位位移量
  或　(EIP) ←(EIP)+32位位移量
- 说明：NEAR是近转移属性描述符，16位、32位位移量是一个带符号数。

控制转移指令-1

- 3. 无条件段内间接转移
- 格式：JMP　OPR　；　OPR可以是寄存器、存储器
- 功能：　(IP)←(EA)　　或　　(EIP) ←(EA)

- 4. 无条件段间直接转移
- 格式：JMP　FAR PTR OPR　；　OPR为指令的标号（即指令地址）
- 功能：(IP/EIP) ←OPR的段内偏移地址
  (CS)←OPR所在段的段地址

- 5. 无条件段间间接转移
- 格式：JMP　DWORD　PTR　OPR　；　OPR为存储器单元
- 功能：(IP/EIP) ←[EA]
  (CS) ←[EA+2]/[EA+4]

控制转移指令-2

- **6．条件相对转移 （8位偏移量）**
- **(1) 单条件相对转移**

① JZ／JE OPR ； ZF=1 转移(结果为零或相等转移)
② JNZ／JNE OPR ； ZF=0 转移(结果不为零或不相等转移)
③ JS OPR ； SF=1 转移(结果为负转移)
④ JNS OPR ； SF=0 转移(结果为正转移)
⑤ JO OPR ； OF=1 转移(结果溢出转移)
⑥ JNO OPR ； OF=0 转移(结果不溢出转移)
⑦ JP／JPE OPR ； PF=1 转移(结果为偶转移)
⑧ JNP／JPO OPR ； PF=0 转移(结果为奇转移)
⑨ JC OPR ； CF=1 转移(有借位或有进位转移)
⑩ JNC OPR ； CF=0 转移(无借位或无进位转移)

- J:jump Z:zero E:equal N:not
- S:sign P:parity C:carry O:overflow
- PE:parity even PO:parity odd

控制转移指令-3

- (2) 无符号数比较条件相对转移(A-B) （ A:above高于 B:below低于）
- ① JB／JNAE／JC OPR ；CF=1 (A<B转移)
- ② JAE／JNB／JNC OPR ；CF=0 (A≥B转移)
- ③ JBE／JNA OPR ；(CF∨ZF)=1 (A≤B转移)
- ④ JA／JNBE OPR ；(CF∨ZF)=0 (A>B转移)

- (3) 有符号数比较条件相对转移(A-B) （ G:greater大于 L:less小于）
- ① JL／JNGE OPR ；(SF⊕OF)=1 (A<B转移)
- ② JGE／JNL OPR ；(SF⊕OF)=0 (A≥B转移)
- ③ JLE／JNG OPR ；((SF⊕OF) ∨ZF)=1 (A≤B转移)
- ④ JG／JNLE OPR ；((SF⊕OF) ∨ZF)=0 (A>B转移)

- (4) 测试CX条件相对转移
- JCXZ OPR ；(CX)=0转移

控制转移指令-4

- 7. LOOP 循环控制相对转移
- 格式：LOOP　OPR
- 功能：（CX/ECX）←(CX/ECX)-1；(CX/ECX)≠0转移

- 8. LOOPZ／LOOPE循环控制相对转移
- 格式：LOOPZ/LOOPE　OPR
- 功能：（CX/ECX）←(CX/ECX)-1；(CX/ECX)≠0且ZF=1转移

- 9. LOOPNZ/LOOPNE循环控制相对转移
- 格式：LOOPNZ(或LOOPNE)　OPR
- 功能：（CX/ECX）←(CX/ECX)-1；(CX/ECX)≠0且ZF=0转移

- 说明：上述三条循环控制相对转移指令中的转移地址为（IP/EIP）←(IP/EIP)+ 8位带符号数；8位位移量是由目标地址OPR确定的。

控制转移指令-5

- **10．子程序调用与返回**
- 子程序：具有独立功能的程序模块。
- 程序中可由调用程序（或称主程序）调用这些子程序，而在子程序执行完后又返回调用程序继续执行。为实现这一功能，微机提供了CALL调用指令和RET返回指令。

- **(1) 段内相对调用**
- 格式：CALL　DST　；　DST为直接入口地址，指令中给出
  CALL　NEAR PTR DST
- 功能：
  操作数16位：
  SP← (SP)-2，[SP]←(IP)
  IP←(IP)+16位位移量
  操作数32位：
  ESP← (ESP)-4，[ESP]←(EIP)
  EIP←(EIP)+32位位移量
- 说明：16位、32位位移量是一个有符号数。

控制转移指令-6

- (2) 段内间接调用
- 格式：CALL　DST　；DST为R、M
- 功能：

  操作数16位：

  SP← (SP)-2，[SP]←(IP)

  (IP) ←(EA)

  操作数32位：

  ESP← (ESP)-4，[ESP]←(EIP)

  (EIP) ←(EA)

- 类型：寄存器、存储器单元。

控制转移指令-7

- (3) 段间直接调用
- 格式：CALL DST ； DST为直接入口地址，指令中给出
- 功能：

操作数16位：

SP← (SP)-2， [SP] ← (CS)

SP← (SP)-2， [SP] ← (IP)

IP ←DST的偏移地址

CS←DST所在段的段地址

操作数32位：

ESP← (ESP)-2， [ESP] ← (CS)

ESP← (ESP)-4， [ESP] ← (EIP)

EIP ←DST的偏移地址

CS←DST所在段的段地址

- 说明：该指令先将CS 、IP或EIP压栈保护返回地址。然后转移到由DST（DST为汇编语言中的过程名）指定的转移地址。

控制转移指令-8

- (4) 段间间接调用
- 格式：CALL　DST；DST为M
- 功能：

操作数16位：

SP← (SP)-2，　[SP] ← (CS)

SP← (SP)-2，　[SP] ← (IP)

IP ←(EA)

CS←(EA+2)

操作数32位：

ESP← (ESP)-2，　[ESP] ← (CS)

ESP← (ESP)-4，　[ESP] ← (EIP)

EIP ←(EA)

CS←(EA+4)

- 说明：先保护返回地址，然后转移到由DST指定的转移地址。EA由DST确定的任何内存寻址方式。

控制转移指令-9

- (5) 段内返回
- 格式：RET
- 功能：

  操作数16位：　IP←栈弹出2字节，SP ← (SP)＋2

  操作数32位：　EIP←栈弹出4字节，ESP ← (ESP)＋4

- (6) 段内带参数返回
- 格式：RET　N
- 功能：

  操作数16位：　IP←栈弹出2字节，SP ← (SP)＋2

  SP ←(SP)+N

  操作数32位：　EIP←栈弹出4字节，ESP ← (ESP)＋4

  ESP ←(ESP)+N

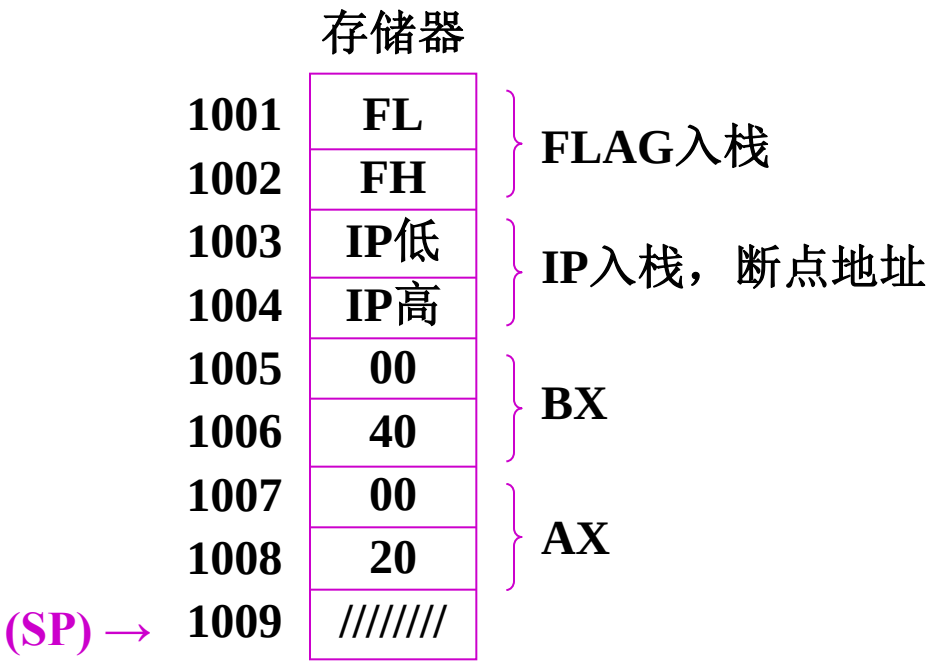
- 说明：N是一个16位的常数（偶数）。

RET N 举例

```
；主程序
MOV   SP,  1009H
MOV   AX,  2000H
MOV   BX,  4000H
PUSH  AX
PUSH  BX
CALL  NEAR PTR ADDPRG
MOV   AX,  MEM1
HLT

；子程序
ADDPRG:
   PUSHF
   MOV   BP,  SP
   MOV   AX,  [BP+4]
   ADD   AX,  [BP+6]
   MOV   MEM1,  AX
   POPF
   RET 4
```

执行后；AX=6000H　SP=1009H



控制转移指令-10

- （7）段间返回
- 格式：RET
- 功能：

  操作数16位：　IP←栈弹出2字节，SP ← (SP)＋2
  CS←栈弹出2字节，SP ← (SP)＋2

  操作数32位：　EIP←栈弹出4字节，ESP ← (ESP)＋4
  CS←栈弹出2字节，ESP ← (ESP)＋2
- (8) 段间带参数返回
- 格式：RET　N
- 功能：

  操作数16位：　IP←栈弹出2字节，SP ← (SP)＋2
  CS←栈弹出2字节，SP ← (SP)＋2
  SP ←(SP)+N

  操作数32位：　EIP←栈弹出4字节，ESP ← (ESP)＋4
  CS←栈弹出2字节，ESP ← (ESP)＋2
  ESP ←(ESP)+N

控制转移指令-11

- 11．中断指令
- 中断指令作用：调用中断服务程序。中断向量地址=中断类型码×4。

- （1）INT 中断指令
- 格式：INT n　　；不影响标志位
- 功能：

① SP←（SP）-2
② PUSH (FR)　　；标志寄存器FR进栈 1
③ SP←（SP）-2
④ PUSH （CS）　　；断点段地址CS进栈 2
⑤ SP←（SP）-2
⑥ PUSH （IP）　　；断点地址指针IP进栈 3
⑦ TF←0　　；禁止单步
⑧ IF←0　　；禁止中断 4
⑨ IP←[n×4]　　；转向中断服务程序 5
⑩ CS←[n×4+2]

控制转移指令-12

- (2) INT3中断断点
- 格式：INT3 ；同INT n。
- 说明：产生类型为 3 的中断，不影响标志位。
- (3) INTO溢出中断
- 格式：INTO ；OF=1产生类型为4的中断，OF=0顺序执行下条指令。
- 说明：产生中断时，TF←0，IF←0，其他标志位不受影响。
- (4) IRET中断返回
- 格式：IRET ；标志位随标志寄存器出栈操作而变
- 功能：

① IP←栈弹出2字节 ；断点偏移量出栈

② SP←（SP）+2

③ CS←栈弹出2字节 ；断点段地址出栈

④ SP←(SP)+2

⑤ FR←栈弹出2字节 ；标志寄存器出栈

⑥ SP←(SP)+2

控制转移指令-13

- **(6) INT 21H 系统功能调用**
- **系统功能调用：DOS为系统程序员及用户提供的一组中断服务程序。**
- **DOS规定用中断指令INT 21H作为进入各功能调用中断服务程序的总入口，再为每个功能调用规定一个功能号，以便进入相应各个中断服务程序的入口。**
- **程序员使用系统功能调用的过程：**

  1）AH ←功能调用编号

  2）设置入口参数

  3）CPU执行 INT 21H

  4）给出出口参数

- **DOS共提供了约80个功能调用（见附录B）**

控制转移指令-14

- ① 功能号：1 ；等待键盘输入，并回送显示器

  MOV AH，1 ；（AH）←功能号01H

  INT 21H ；调用21H号软中断
- 说明：出口参数（AL）= 键入字符的ASCII码。

- ② 功能号：2 ；输出字符送显示器

  MOV DL，41H ；入口参数：（DL）←字符A的ASCII码

  MOV AH，2 ；（AH）←功能号02H

  INT 21H ；在屏幕上显示输出字符A
- 说明：无出口参数。

- ③ 功能号：4CH ；终止程序，返回

  MOV AH，4CH ；（AH）←功能号4CH

  INT 21H ；调用21H号软中断

5.3.8 串操作指令

- 串操作指令处理连续存放在存储器中的数据。
- 7个，其中与存储器相关的5个，与I/O相关的2个。
- 存储器相关的5个：传送MOVS、比较CMPS、扫描SCAS、装入LODS、存储STOS
- 与I/O相关的2个： 输入INS、输出OUTS

- 共同特性：

1）数据类型：字节、字、双字

2）源串： DS:SI/ESI，允许段超越

3）目的串： ES:DI/EDI

4）每执行一次串操作指令，SI/ESI、DI/EDI自动修改，指向下一位置。

5）地址修改方向：由DF控制，DF=0增址（CLD），DF=1减址（STD）

6）带B、W、D为隐式，否则为显式，如MOVS BYTE PTR [DI], [SI]

7）指令中只给出一个操作数时，另一个隐含为AL、AX、EAX之一。

8）可加相应的重复前缀，赋值CX/ECX为计数值。

串操作指令-1

- (1) MOVS串传送（move string）
- 格式：MOVS　DST, SRC

  MOVSB　(字节)

  MOVSW　(字)

  MOVSD　(双字) (自386起有)

- 功能：ES:[DI/EDI]←DS:[SI/ESI]　　；不影响标志位

  字节传送：（DI/EDI）←(DI/EDI)±1

  （SI/ESI）←(SI/ESI) ±1

  字传送：（DI/EDI）←(DI/EDI) ± 2

  （SI/ESI）←(SI/ESI) ± 2

  双字传送：（DI/EDI）←(DI/EDI) ± 4

  （SI/ESI）←(SI/ESI) ± 4

串操作指令-2

- (2) LODS串装入（load from string ）
- 格式：LODS SRC ；字节装入 AL←DS:[SI/ESI]
- (3) STOS 串存储（store into string）
- 格式：STOS DST ；字节存储 ES:[DI/EDI] ←AL
- (4) CMPS 串比较（compare string）
- 格式：CMPS DST, SRC ； DS:[SI/ESI]-ES:[DI/EDI]，影响标志位
- (5) SCAS 串扫描（scan string）
- 格式：SCAS DST ；字节扫描 （AL）- ES:[DI/EDI]，影响标志位

- (6) INS串输入（input from port to string）
- 格式：INS DST, DX ； ES:[DI/EDI] ←[DX]
- (7) OUTS串输出（output string to port）
- 格式：OUTS DX, SRC ； [DX]←DS:[SI/ESI]

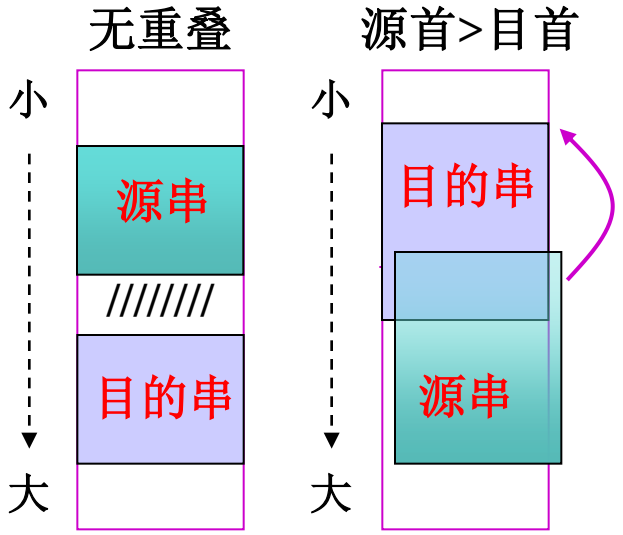
串操作指令-3

- **(8) REP 计数重复串操作（repeat）**
- **格式：REP OPR ； OPR是MOVS、STOS、LODS、INS、OUTS**
- **功能：① 如果(CX)=0，则退出REP，否则往下执行。**
  ②（CX）←（CX）-1
  ③ 执行其后的串指令
  ④ 重复①～③
- **(9) REPZ 计数相等重复串操作（repeat while zero/equal）**
- **格式：REPZ/REPE OPR ； OPR是CMPS、SCAS**
- **功能：① 如果(CX)=0或ZF=0(比较的两数不等)，则退出REPZ，否则往下执行。**
  ②（CX）←（CX）-1
  ③ 执行其后的串指令
  ④ 重复①～③
- **(10) REPNZ 计数不相等重复串操作（repeat while not zero/not equal）**
- **格式：REPNZ/REPNE OPR ； OPR是CMPS、SCAS**
- **功能：与REPZ相同，但条件不同。**

MOVS举例

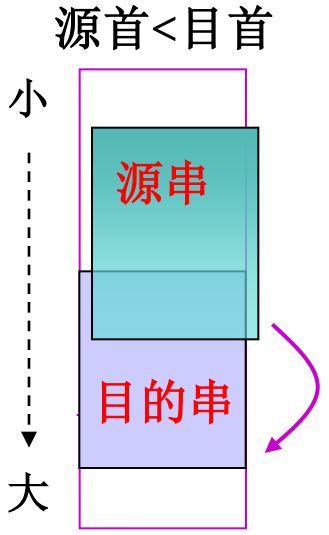
（1）MEM1→MEM2，无重叠，或源首>目首

```
        MOV  SI,  OFFSET MEM1
        MOV  DI,  OFFSET MEM2
        MOV  CX, 100
        CLD                  ；增址
LOOP1:  MOVSB                可改成REP MOVSB
        LOOP  LOOP1
        HLT
```



（2）MEM1→MEM2，有重叠，源首<目首

```
        MOV  SI,  OFFSET MEM1
        MOV  DI,  OFFSET MEM2
        MOV  CX, 100
        ADD  SI,  99
        ADD  DI,  99
        STD                  ；减址
        REP  MOVSB
        HLT
```



STOS举例

例：将MEM1开始的100个字单元清0。

```
LEA   DI, MEM1
MOV  AX,  0
MOV  CX, 100
CLD                ;  增址
REP STOSW
HLT
```

5.3.9 处理器控制指令

- **1. 标志处理指令**
- 标志处理指令只设置或清除本指令的标志位，而不影响其他标志位。

- **CLC ；进位位置0， CF←0 （clear carry）**
- **CMC ；进位位取反，CF← $\overline{CF}$ （complement carry）**
- **STC ；进位位置1， CF←1 （set carry）**
- **CLD ；方向标志位置0，DF←0 （clear direction）**
- **STD ；方向标志位置1，DF←1 （set direction）**
- **CLI ；中断标志置0，IF←0，禁止中断（clear interrupt）**
- **STI ；中断标志置1，IF←1，允许中断 （set interrupt）**

处理器控制指令

- 2. 处理器控制指令
- 处理器控制指令可以控制处理机状态。它们都不影响条件码。
- 1）HLT ；暂停

  暂停指令停止软件的执行。有三种方式退出暂停：中断、硬件复位、DMA操作。
- 2）NOP ；空操作

  不执行任何操作。
- 3）ESC ；换码

  指定由协处理器执行的指令。486后浮点处理部件已装入CPU，系统可直接支持协处理器指令，因此ESC指令已成为未定义指令。
- 4）WAIT ；等待

  使处理机处于空转状态，它也可以用来等待外部中断发生，但中断结束后仍返回WAIT指令继续等待。
- 5）LOCK ；封锁前缀

  有此前缀的指令执行后，CPU封锁总线，禁止其他总线主设备访问总线。如LOCK MOV AL，[SI]。

第5章 结 束