第3章
指令系统（Ⅳ）

3.6 字符串操作指令

串操作

- 字符串：一系列存放在存储器中的字或字节数据，不管它们是不是ASCII码。字符串长度可达64K字节。
- 字符串元素：组成字符串的字节或字。每种字符串指令对字符串元素只进行同一种操作。
- 字符串操作指令：对字符串进行的传送、比较、搜索、装入及填充等5种操作。

串操作指令就是用一条指令实现对一串字符数据的操作；是唯一的一组源操作数和目的操作数都在存储单元的指令。

字符串操作指令的类型和格式

指令名称		字节操作	字操作
字符串传送 MOVS	目的串，源串	MOVSB	MOVSW
字符串比较 CMPS	目的串，源串	CMPSB	CMPSW
字符串搜索 SCAS	目的串	SCASB	SCASW
字符串装入 LODS	源串	LODSB	LODSW
字符串填充 STOS	目的串	STOSB	STOSW

每种指令都有3种格式。有两种方法用以说明是字节操作还是字操作。

- 方法一：用指令中的源串和目的串名(即操作数)来表明是字节还是字；
- 方法二：在指令助记符后加B说明是字节，加W说明是字操作。

❖ 串操作指令的隐含约定：

①源串：用寄存器SI指向源串偏移首地址，DS作为源串的段地址，即DS：SI 。源串允许使用段超越前缀来修改段地址。

②目的串：用寄存器DI指向目的串偏移首地址，ES作为目的串段地址，即ES：DI。目的串不允许使用段超越前缀修改ES。如果要在同一段内进行串运算，必须使DS和ES指向同一段。

③指针：每执行一次字符串指令，指针SI和DI会自动修改，以指向下一个待操作单元。

④用方向标志位DF的值决定串处理方向；

DF＝0为递增方向。DS：SI指向源串首地址；

DF＝1为递减方向。DS：SI指向源串末地址。

⑤字节串操作时，地址指针SI/DI分别加/减1

字串操作时，地址指针SI/DI分别加/减2；

STD使DF置1，CLD将DF清0。

⑥串长度：在串操作指令前面可以加重复前缀指令前缀REP、REPZ/REPE、REPNZ/REPNE，能重复执行该指令。固定用**CX指定字符串个数**。

关于重复前缀:

在基本指令前加重复前缀，可加快串运算指令的执行速度。每重复执行一次，地址指针SI和DI按方向标志自动修改，CX的值自动减1。

类型与格式:

- REP：无条件重复(Repeat)，常与(MOVS)连用，连续传送字符串。直到传送完毕，即CX＝0为止。

- REPE/REPZ：相等/结果为零则重复(Repeat while Equal/Zero)，常与(CMPS)连用，连续比较字符串。当两个字符串对应字符相等(ZF＝1)和CX≠0时，则重复进行比较，直到ZF＝0或CX＝0为止。
- REPNE/REPNZ：不相等/结果非零则重复(Repeat while Not Equal/Not Zero)，常与(SCAS)连用，当结果非0(ZF＝0)和CX≠0时，重复进行扫描，直到ZF＝1或CX＝0为止。
- 带有重复前缀的串指令执行过程中允许有中断进入。

1．MOVS字符串传送指令(Move String)

- 指令格式：MOVS　目的串，源串
- 指令功能：将DS段SI寻址的一个字节或字传送到ES段DI寻址的单元中，并修改SI和DI使指向下一个元素。
- 应用：
  - 解决MOV指令不能直接在存储单元间进行数据传送的问题。
  - 若使用重复前缀，还可以利用一条指令传送一批数据。

例　要求把数据段中以SRC_MESS为偏移地址的一串字符“HELLO！”，传送到附加段中以NEW_LOC开始的单元中。

```
DATA        SEGMENT                ；数据段
SRC_MESS    DB    'HELLO！'        ；源串
DATA        ENDS
;
EXTRA       SEGMENT                ；附加段
NEW_LOC     DB  6  DUP(？)         ；存放目的串
EXTRA       ENDS
;
CODE        SEGMENT
            ASSUME  CS：CD，  DS：DATA，  ES：EXTRA
START：     MOV   AX，DATA
            MOV   DS，AX                ；DS＝数据段段址
            MOV   AX，EXTRA
            MOV   EX，AX                    ；ES＝附加段段址
            LEA   SI，SRC_MESS              ；SI指向源串偏移地址
            LEA   DI，NEW_LOC               ；DI指向目的串偏移地址
            MOV   CX，6                     ；CX作串长度计数器
            CLD                             ；清方向标志，地址增量
            REP   MOVSB                 ；重复传送串中各字节，直到CX=0
```

其中“REP MOVSB”指令可用以下几条指令代替：

```
  AGAIN：   MOVS    NEW_LOC，  SRC_MESS
            DEC     CX
            JNZ     AGAIN
```

课堂练习：课后习题11

若把1K字节的数据块从偏移地址为1000H开始的单元传送到偏移地

冲区。试用串传送操作

串操作指令编程	
	MOV SI，1000H
	MOV DI，1400H
	MOV CX，400H
	CLD
REP	MOVSB
	MOV AH，4CH
	INT 21H

一般操作指令编程	
	MOV SI，1000H
	MOV DI，1400H
	MOV CX，400H
NEXT:	MOV AL，[SI]
	MOV [DI]，AL
	INC SI
	INC DI
	LOOP NEXT
	MOV AH，4CH
	INT 21H

2、CMPS字符串比较指令(Compare Sting)

- 指令格式：CMPS　目的串，源串
- 指令功能：把DS段由SI作指针的源串减去ES段由DI作指针的目的串数据，结果反映在标志位上；两个数据串的原始值不变；源串和目的串指针自动修改，指向下一对待比较的串。
- 加重复前缀：

▲REPE/REPZ　CMPS；直至CX＝0(比完了)或ZF＝0(两串不相等)时停止操作。

▲REPNE/REPNZ CMP；直至CX＝0(比完了)或ZF＝1(两串相等)时停止比较。

例 比较两个字符串，一个是程序中设定的口令串PASSWORD，另一个是从键盘输入的字符串IN_WORD，若输入串与口令串相同，程序将开始执行。否则，程序驱动PC机的扬声器发声，警告用户口令不符，拒绝往下执行。

```
DATA        SEGMENT                 ;数据段
PASSWORD    DB      '750424LI'      ;口令串
IN_WORD     DB      '750424LE'      ;从键盘输入的串
COUNT       EQU     8               ;串长度
DATA        ENDS
            ……
CODE        SEGMENT                 ;代码段
            ……
            LEA     SI, PASSWORD    ;源串指针
            LEA     DI, IN_WORD     ;目的串指针
            MOV     CX, COUNT       ;串长度
            CLD                     ;地址增量
            REPZ    CMPSB           ;CX≠0且串相等时重复比较
            JNE     SOUND           ;若不相等，转发声程序
OK:         ……                      ;比完且相等，往下执行
            ……
SOUND:      ……                      ;使PC机扬声器发声
            ……                      ;并退出
CODE        ENDS
```

3、SCAS字符串扫描指令(Scan String)

- 指令格式：SCAS　目的串
- 指令功能：AL/AX(字节/字操作)内容减去ES:DI中的串元素，结果反映在标志位上，源操作数不变，操作后目的串指针会自动修改，指向下一个待搜索的串元素。
- 应用：
  - 利用SCAS指令，可在内存中搜索关键字。指令执行前，必须事先将关键字存在AL(字节)或AX(字)中，才能用SCAS指令进行搜索。
  - SCAS指令可以加重复前缀。

例 在某一字符串中搜寻是否有字符A，若有，则把搜索次数记下来，送到BX寄存器中，若没有查到，则将BX寄存器清0。设字符串起始地址STRING的偏移地址为0，字符串长度为CX。

```
解:     MOV   DI，OFFSET STRING ；DI＝字符串偏移地址
        MOV   CX，COUNT         ；CX＝字符串长度
        MOV   AL，‘A’           ；AL＝关键字A的ASCII码
        CLD                     ；清标志方向
        REPNE SCASB     ；CX≠0（没查完）和ZF=0（不相等）时重复
        JZ    FIND      ；若ZF=1，表示已搜到，转出
        MOV   DI，0     ；若ZF=0，表示没搜到，DI ←0
FIND:   MOV   BX，DI            ；BX←搜索次数
        HLT
```

说明：

- DI初值存起始地址**偏移量0**，搜索一次后，DI自动加1，使DI的值等于1；
- 每执行一次搜索操作，**DI自动增1**；
- 因此可用DI的值表示**搜索次数**。

4、LODS数据串装入指令(Load String)

- 指令格式：LOAD 源串
- 指令功能：DS:SI中的串元素加载到AL/AX(字节/字操作)中，修改SI，以指向串中的下一个元素，修改量遵守隐含约定(4)。
- 注：

该指令加重复前缀意义不大，因为重复传送只能保留最后写入的数据。

5、STOS数据串填充指令(Store string)

- 指令格式：STOS　目的串
- 指令功能：AL/AX(字节/字操作)的一个字节或字传送到ES:DI所指的目的串；修改DI，以指向串中的下一个单元。
- 关于重复前缀：

“REP STOS”，可用累加器中的常数，对数据串初始化。如初始化为全0串

3.7 程序控制类指令

无条件转移和过程调用指令	
JMP	无条件转移
CALL	过程调用
RET	过程返回
条件转移	
JZ / JE (10条指令)	直接标志转移
JA / JNBE (8条指令)	间接标志转移
条件循环控制	
LOOP	CX≠0 则循环
LOOPE / LOOPZ	CX≠0和ZF=1 则循环
LOOPNE / LOOPNZ	CX≠0和ZF=0 则循环
JCXZ	CX=0 则转移
中 断	
INT	中断
INTO	溢出中断
IRET	中断返回

通过改变IP和CS值，实现程序执行顺序的改变。

3.7.1 调用、转移与返回指令

1、JMP无条件转移指令(Jump)

- 指令格式：JMP　目的
- 指令功能：程序无条件地转移到指定的目的地址去执行。
- 两种转移类型：段内转移或近(NEAR)转移

转移指令的目的地址和JMP指令在同一代码段中，转移时，仅改变IP寄存器的内容，段地址CS的值不变。

段间转移，又称为远(FAR)转移

转移指令的目的地址和JMP指令不在同一段中，转移时，CS和IP的值都要改变，程序转到另一代码段去执行。

❖ 与转移地址有关的4种寻址方式

类型	方式	寻址目标		指令举例
段内转移	直接	立即短转移（8位）	①	JMP SHORT PROG_S
	直接	立即近转移（16位）		JMP NEAR PTR ROG_N
	间接	寄存器（16位）	②	JMP BX
	间接	存储器（16位）		JMP WORD PTR 5[BX]
段间转移	直接	立即转移（32位）	③	JMP FAR PTR PROG_F
	间接	存储器（32位）	④	JMP DWORD PTR [DI]

①段内直接寻址：转移的偏移地址是当前IP值和指令中指定的8位或16位位移量之和；

②段内间接寻址：转移的偏移地址是一个寄存器或一个存储单元的内容；

③段间直接寻址：指令中直接提供了转向的段地址和偏移地址；

④段间间接寻址：用存储器中的两个相继字内容取代IP和CS中的内容；

【例】 JMP BX

解： 若该指令执行前BX＝4500H;

则指令执行时，将当前IP修改成4500H;

程序转向段内偏移地址为4500H处执行。

【例】 JMP WORD PTR 5[BX]

解：设指令执行前，DS＝2000H，BX＝100H，(20105H)＝04F0H;

则指令执行后，IP＝(20000H+100H+5H)＝(20105H)＝04F0H;

转到代码段内偏移地址为04F0H处执行。

例 JMP FAR PTR PROG_F

解：设标号PROG_F所在段的基地址＝3500H，偏移地址＝080AH;

则指令执行后，IP＝080AH，CS＝3500H;

程序转到3500：080AH处执行。

【例】 **JMP** DWORD PTR[SI十0125H]

解：设指令执行前：

CS＝1200H，IP＝05H，DS＝2500H，SI＝1300H；

内存单元(26425H)＝4500H，(26427H)＝32F0H。

指令中的位移量D1SP＝0125H，即：DISP_H＝01H，DISP_L＝25H。

由附录B可知，该指令占4个字节。

第1个字节是操作码：FFH；

第2个字节的编码：MOD 101 R/M，查表3-2可得到MOD＝10，R/M＝100；

第3和第4字节：位移量低字节和高字节；

于是，机器码为FF AC 25 01H，该指令的编码格式为：

OP	MOD 101 R/M	DISP_L	DISP_H
1111 1111	10 101 100	0010 0101	0000 0001

2、调用指令(CALL)

CALL指令用来调用一个过程或子程序。由于过程或子程序有段间(即远程**FAR**)和段内调用(即近程**NEAR**)之分。

近过程调用：调用指令CALL和被调用的过程在同一代码段中。

远过程调用：两者在不同代码段中。

近调用的操作：SP←SP-2，IP入栈；

远调用的操作：SP←SP-2，CS入栈；SP←SP-2，IP入栈

【课后习题14】若DS=1100H，SI=1000H，CS=1200H，IP=100H，[12000H]=2000H，[12002H]=1500H，求执行指令

（1）CALL FAR[SI]后CS与IP的内容；

（2）CALL [SI]后CS与IP的内容。

解：

（1）CS=1500H，IP=2000H

（2）CS=1200H，IP=2000H

3、RET指令

从栈中弹出返回地址，使程序返回主程序继续执行。

这里分为两种情况：

如果从**近过程**返回：

从栈中弹出一个字→IP，并且使SP←SP+2

如果从**远过程**返回：

先从栈中弹出一个字→IP，并且使SP←SP+2；

再从个字栈中弹出一个字→CS，并使SP←SP+2。

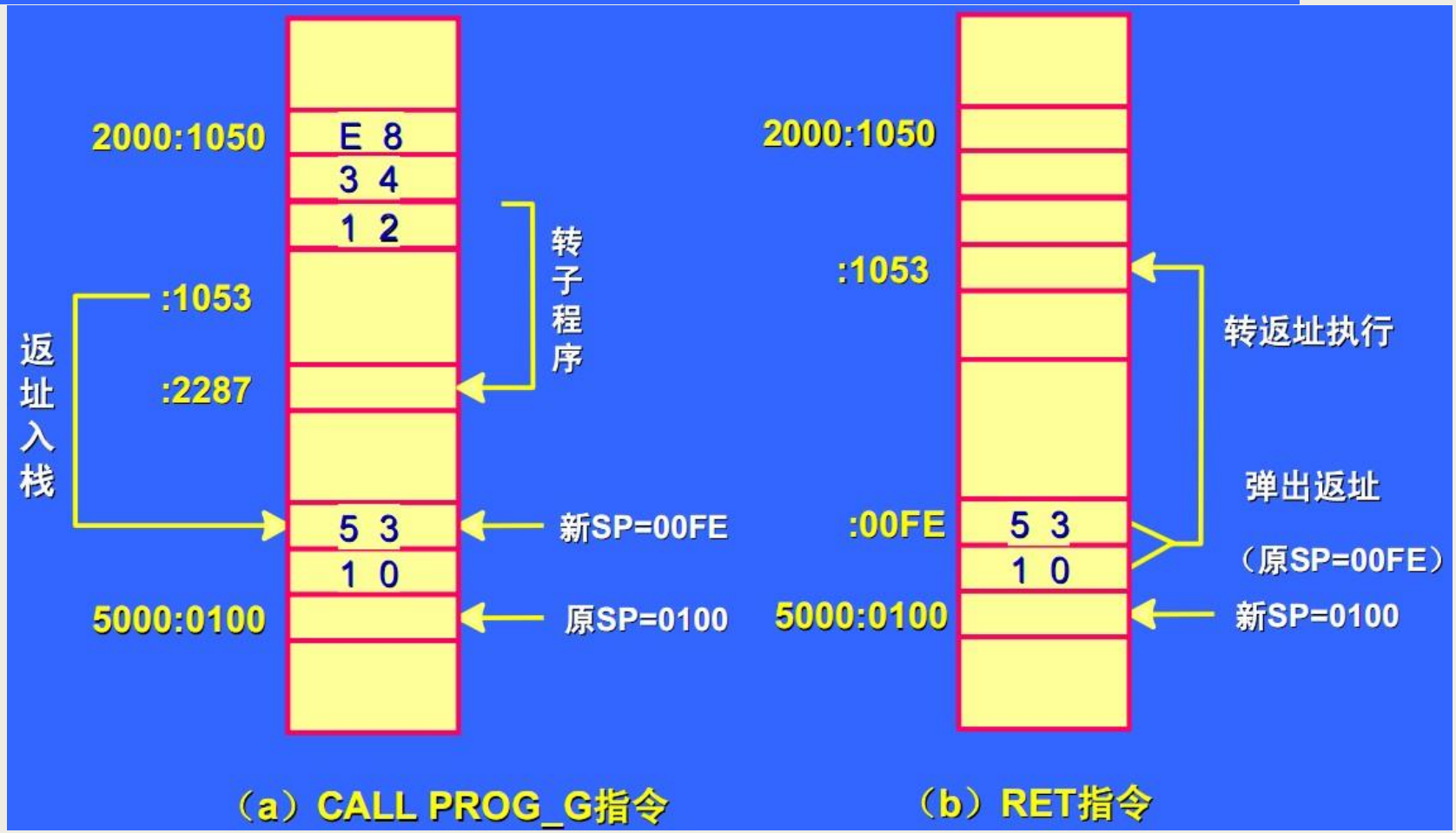
例 CALL PROG_N ；PROG_N是一个近标号

解：该指令含3字节，编码格式为：

调用之前：CS=2000H，IP=1050H，SS=5000H，SP=0100H，PROG_N与CALL指令之间的距离等于1234H(即DISP＝1234H)，则

E8	DISP_L	DISP_H

近过程调用



例　　CALL FAR PTR PROG_F；PROG_F是一个远标号

解：指令为5字节，编码格式为：

9A	DISP_L	DISP_H	SEG_L	SEG_H

调用之前：

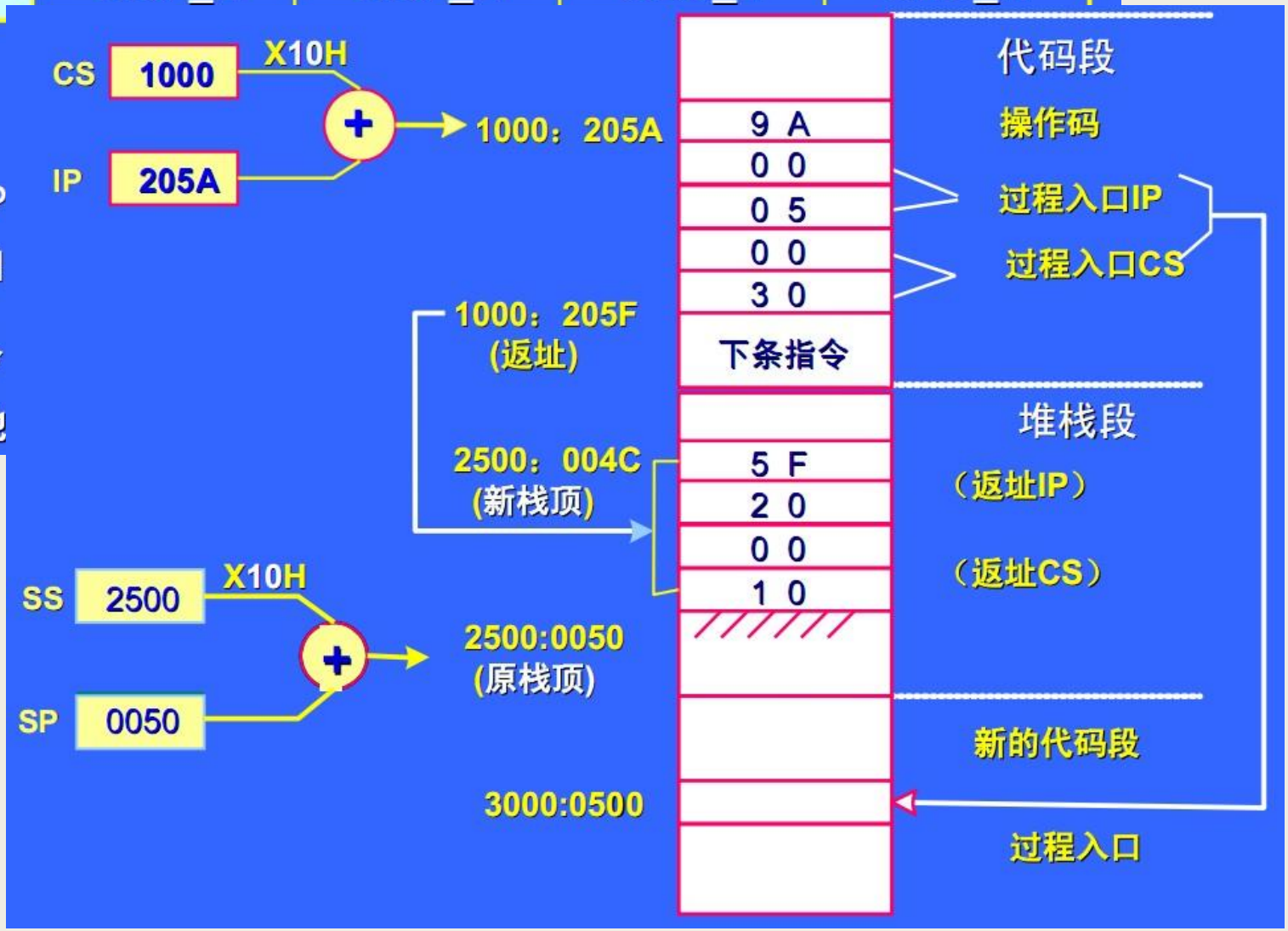
CS=1000H，IP

指针CS=3000H

存放CALL指令

节，所以返回地

远过程调用



3.7.2 条件转移指令

根据上一条指令执行后的状态标志作为判别测试条件来决定是否转移。

注意:

条件转移均为段内短转移。

目的地址＝当前IP值＋8位相对位移量。

它与转移指令后那条指令的距离，允许为-128~+127 字节;

8位偏移量用符号扩展法扩展到16位与IP相加。

指令格式: 条件操作符 标号

条件转移指令共有18条，可以为两类:

直接标志转移指令

间接标志转移指令

1、直接标志转移指令(10条)

以CF，ZF，SF，OF和PF等5个标志的10种状态为判断的条件。

指令助记符	测试条件	指令功能
JC	CF=1	有进位 转移
JNC	CF=0	无进位 转移
JZ/JE	ZF=1	结果为零/相等 转移
JNZ/JNE	ZF=0	不为零/相等 转移
JS	SF=1	符号为负 转移
JNS	SF=0	符号为正 转移
JO	OF=1	溢出 转移
JNO	OF=0	无溢出 转移
JP/JPE	PF=1	奇偶位为1/为偶 转移
JNP/JPO	PF=0	奇偶位为0/为奇 转移

例： 求AL和BL寄存器中的两数之和，若有进位，则AH置1，否则AH清0。

【解】：可用如下程序段来实现该操作：

```
        ADD     AL，BL          ；两数相加
        JC      NEXT            ；若有进位，转NEXT
        MOV     AH，0           ；无进位，AH=0
        JMP     EXIT            ；往下执行
NEXT:   MOV     AH，1           ；有进位，AH置1
EXIT:   ……                      ；程序继续进行
```

2、间接标志转移

指令的助记符中不直接给出标志状态位的测试条件，而是标志的状态组合作为测试的条件。

指令应用：通常放在比较指令CMP之后，以比较两个数的大小。

- 无符号数比较测试指令中，指令助记符中的“A”是英文Above的缩写，表示“高于”之意，“B”是英文Below的缩写，表示“低于”之意；
- 带符号数比较测试指令中，指令助记符中的“G”(Great than)表示大于，“L”(Less than)表示小于。

2、间接标志转移

类别	指令助记符	测试条件	指令功能
无符号数比较测试	**JA/JNBE**	CF ∨ ZF =0	高于/不低于等于　转移
	JAE/JNB	CF=0	高于等于/不低于　转移
	JB/JNAE	CF=1	低于/不高于等于　转移
	JBE/JNA	CF ∨ ZF=1	低于等于/不高于　转移
带符号数比较测试	**JG/JNLE**	(SF ⊕ OF) ∨ZF=0	大于/不小于等于　转移
	JGE/JNL	SF ⊕ OF =0	大于等于/不小于　转移
	JL/JNGE	SF ⊕ OF =1	小于/不大于等于　转移
	JLE/JNG	(SF ⊕ OF) ∨ ZF=1	小于等于/不大于　转移

【例】 设某个学生的英语成绩已存放在AL寄存器中，若低于60分，则打印F(FAIL)；若高于或等于85分，则打印G(GOOD)；当在60分和84分之间时，打印P(PASS)。

程序：

```
        CMP     AL，60          ；与60分比较
        JB      FAIL            ；＜60，转FAIL
        CMP     AL，85          ；≥60，与85分比较
        JAE     GOOD            ；≥85，转GOOD
        MOV     AL，'P'         ；其它，将AL←'P'
        JMP     PRINT           ；转打印程序
FAIL：  MOV     AL，'F'         ；AL←'F'
        JMP     PRINT           ；转打印程序
GOOD：  MOV     AL，'G'         ；AL←'G'
PRINT： …                       ；打印存在AL中的字符
```

3.7.3 循环指令

特点：

- 是一组增强型的条件转移指令，控制程序段的重复执行，重复次数存于CX寄存器中。
- 指令的字节数均为2，第一字节是操作码，第二字节是8位偏移量，转移的目标都是短标号。
- 循环指令中的偏移量都是负值。
- 循环控制指令均不影响任何标志。

这类指令共有4条。

1、LOOP循环指令(Loop)

- 指令格式：LOOP 短标号
- 指令功能：控制重复执行一系列指令，重复次数放在CX寄存器中，每执行一次LOOP指令，CX 自动减1。如果减1后CX≠0，则转移到指令中所给定的标号处继续循环；若自动减1后CX＝0，则结束循环。
- 一条LOOP指令相当于执行以下两条指令的功能：

```
DEC  CX
JNZ  标号
```

2、 LOOPE/LOOPZ相等或结果为零时循环 (Loop If Equal / Zero)

- 指令格式：LOOPE 标号 或 LOOPZ 标号
- 指令功能：

LOOPE是相等时循环；

LOOPZ是结果为零时循环；

指令执行前，先将重复次数送到CX中，每执行一次指令，CX 自动减1，若**减1后CX≠0和ZF＝1**，则转到指令所指定的标号处**重复执行**；若**CX＝0或ZF＝0**，便**退出循环**。

3、LOOPNE/LOOPNZ不相等或结果不为零循环(Loop If Not Equal/Not Zero)

- 指令格式：LOOPNE 标号 或 LOOPNZ 标号
- 指令功能：

LOOPNE是不相等时循环

LOOPNZ是结果不为零循环

指令执行前，应将重复次数送入CX，每执行一次，CX自动减1，若**减1后CX≠0和ZF＝0**，则转移到标号所指定的地方**重复执行**；若**CX＝0或ZF＝1**，则退出循环，顺序**执行下一条指令**。

4、JCXZ若CX为0跳转(Jump If CX Zero)

- 指令格式：JCXZ　标号
- 指令功能：

若CX寄存器为零，则转移到指令中标号所指定的地址处，它不对CX寄存器进行自动减1的操作。

主要用在循环程序开始处，为跳过循环，把CX寄存器清零。

中断概念

- 定义：计算机暂时中止当前程序的运行，执行中断服务程序去为临时发生的事件服务，执行完毕后，返回正常程序继续运行，这个过程称为中断。

3.7.4 中断控制指令

中断过程:

- 断点(CS，IP值)、标志寄存器的值入栈保护
- 取入口地址(中断类型×4得到中断向量的地址)
- 执行中断服务程序
- 执行中断返回指令IRET，从堆栈中恢复中断前CPU的状态和断点。

中断向量表:

- 低2字节存放中断服务程序入口地址的偏移量IP;
- 高2字节存放中断服务程序入口地址的段地址CS; 中断服务程序入口地址＝中断类型×4;

1、软件中断指令(Interrupt)

INT n

也称陷阱中断。n为中断类型号，范围0～255。

CPU执行INT n指令的过程：

- 标志寄存器的内容入栈;
- 当前断点的段基地址CS、偏移地址IP入栈;
- 清除中断标志IF、单步标志TF ；
- 将中断类型号n乘以4，找到中断服务程序的入口地址表的表头地址，获得中断服务程序的入口地址，置入CS和IP寄存器转到相应中断服务程序。

2、溢出中断指令(Interrupt On overflow)

INTO

- 带符号数进行算术运算时，若OF＝1，则溢出中断指令INTO产生类型为4的中断。
- 可在带符号加减法运算之后安排一条INTO指令。

3、中断返回指令(Interrupt Return)

IRET

实现中断返回。

安排在中断服务程序的出口处。

执行后：

- 首先从堆栈中依次弹出程序断点，送到IP和CS寄存器中；
- 接着弹出标志寄存器的内容，送回标志寄存器；
- CPU按CS：IP的值返回断点，继续执行原来被中断的程序。

【课后习题17】下面程序段，在什么情况下执行结果是AL = 0?

```
BEGIN:      IN AL，5FH
            TEST AL，80H
            JZ BRCH1
            XOR AX，AX
            JMP STOP
BRCHI:      MOV AL，0FFH
STOP:       MOV AH，4CH
            INT 21H
```

解：如果端口5FH读出的数据最高位(D7)为1时，则AL=0，否则AL=FFH

3.8 CPU控制指令

1、标志操作指令(CF、DF、IF位操作)

指令助记符	操作	指令名称
CLC	CF←0	进位标志清0 （Clear Crray）
CMC	CF←/CF	进位标志求反 (Complement Carry)
STC	CF←1	进位标志置1 （Set Carry）
CLD	DF←0	方向标志清0 （Clear Direction）
STD	DF←1	方向标志置1 （Set Direction）
CLI	IF←0	中断标志清0 （Clear Interrupt）
STI	IF←1	中断标志置1 （Set Interrupt）

2、NOP空操作或无操作指令(No Operation)

单字节指令，执行时需耗费3个时钟周期的时间，但不完成任何操作。

常在循环等操作中增加延时，或在调试程序时使用空操作指令。

3、HLT处理器暂停(Processor Halt)

CPU进入暂停状态，不进行任何操作。

下列情况发生时，CPU才脱离暂停状态：

- 在RESET线上加复位信号；
- 在NMI引脚上出现中断请求信号；
- 在允许中断的情况下，在INTR引脚上出现中断请求信号；

4．WAIT等待指令(Wait)

通常用在CPU执行ESC指令后，表示8086CPU正处于等待状态，它不断检测8086的测试引脚TEST，每隔5个时钟周期检测一次，若此脚为高电平，则重复执行WAIT指令，处理器处于等待状态。一旦TEST引脚的信号变为低电平，便退出等待状态，执行下条指令。

5、ESC换码指令(Escape)

- 指令格式：ESC 外部操作码，源操作数
- 指令功能：换码指令用来实现8086对8087协处理器的控制。

6、LOCK封锁总线指令(Lock Bus)

是一种前缀，加在指令的前端，用来维持8086的总线封锁信号LOCK有效，带有LOCK前缀的指令在执行过程中，禁止其它协处理器使用总线。

以上三个是使CPU与其它协处理器同步工作的指令，用于多处理机；指令执行后均不影响标志位。