计算机系统基础

许 向 阳

xuxy@hust. edu. cn

计算机科学与技术学院

第9章 串处理程序设计

一、学习内容

串操作指令的使用格式及功能；
对应的 C 程序中的库函数

二、学习重点

MOVS、CMPS、SCAS的使用格式及功能

三、学习的难点

灵活应用，编写程序

提高编程效率，简化程序设计工作。
使用串操作指令，提高程序运行速度。

9.1 串操作指令简介

串传送指令　MOVSB、MOVSW、MOVSD

串比较指令　CMPSB、CMPSW、CMPSD

串搜索指令　SCASB、SCASW、SCASD

从源串中取数　LODSB、LODSW、LODSD

向目的串中存数　STOSB、STOSW、STOSD

9.1 串操作指令简介

串操作的共同特征

■串传送（串拷贝）

将以BUF1为首址的100个字节的数据拷贝到以BUF2为首址的区域。

源串指示器
目的串指示器
拷贝字符数计数
拷贝方向（增量：从缓冲区头部开始）
（减量：从缓冲区尾部开始）

9.1 串操作指令简介

串操作的共同特征

■串比较

比较以BUF1和BUF2为首址的字符串是否相同

源串指示器
目的串指示器
比较字符数计数
方向　　（增量：从缓冲区头部开始比较）

9.1 串操作指令简介

串操作的共同特征

■串搜索

在以BUF1 首址的字符串中是否含有空格？

待搜索对象
目的串指示器
字符串长度
方向　　（增量：从缓冲区头部开始找）

9.1 串操作指令简介

串操作的共同特征

■串存储（往目的串中存数据）

将以BUF1 首址的100个字节缓冲区中的内容均置为0。

待存储对象
目的缓冲区指示器
数据个数
方向　　（增量：从缓冲区头部开始找）

9.1 串操作指令简介

源串指针：　DS：ESI　源串在当前数据段
目的串指针：　ES：EDI　目的串在附加数据段
重复计数器：　ECX
中间寄存器：　AL / AX / EAX

传送/比较方向：(decrease)

DF=0，　ESI、EDI自动增量（加1,2,4）
DF=1，　ESI、EDI自动减量（减1,2,4 ）

重复前缀：

REP　（ECX）<>0 时重复执行；
REPE　（ECX）<>0 且 ZF=1时重复执行；
REPNE　（ECX）<>0 且 ZF=0时重复执行；

9.2 串传送指令

语句格式：

MOVSB　　MOVSW

MOVSD　　MOVS　OPD, OPS

功能：　(1) (DS:[ESI]) → ES:[EDI]

(2) 若 DF=0, 则 (ESI)和(EDI)增1 (字节)

或增2（字操作）

或增4（双字操作）

若DF=1,则(ESI)和(EDI)减1、或2、或4。

9.2 串传送指令

串传送程序示例：

观察执行一次 MOVSB指令后，ESI,EDI的变化

要传送多个字符时，怎么办？

使用前缀　REP

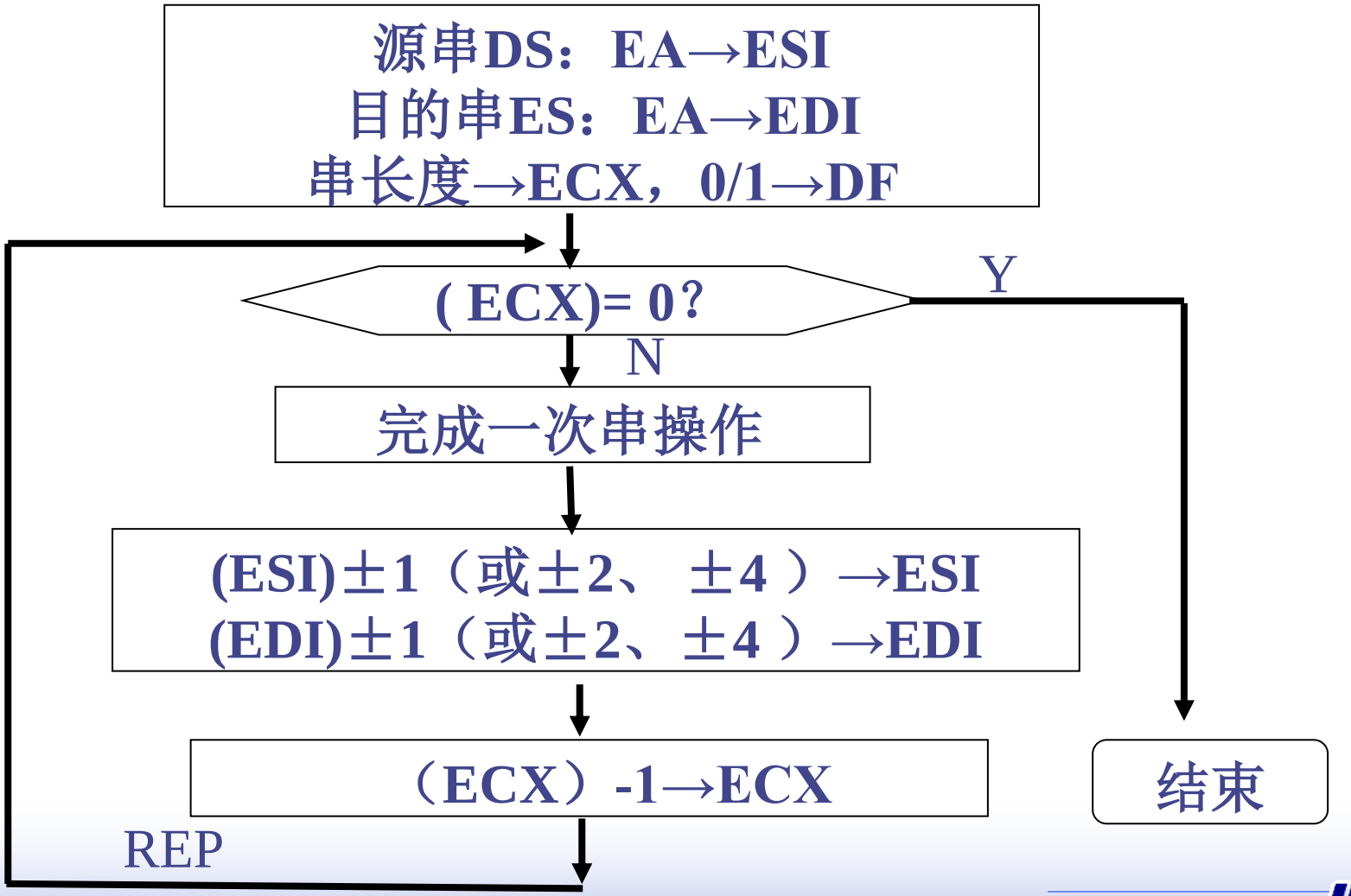
指令的运行过程

注意ECX的设置，ESI，EDI的变化

有前缀 REP时，先判断(ECX) 是否为 0，

若(ECX)== 0，则一次也不会传送。

9.2 串传送指令


源串DS：EA→ESI
目的串ES：EA→EDI
串长度→ECX，0/1→DF
(ECX)= 0?
Y
N
完成一次串操作
(ESI)±1（或±2、 ±4 ）→ESI
(EDI)±1（或±2、±4 ）→EDI
（ECX）-1→ECX
结束
REP

用与不用串传送指令的程序效率比较

Q: 如何计算一段程序的运行时间？

```
invoke clock
mov    begin_time, eax
……
invoke clock
mov    end_time, eax
sub    eax, begin_time
```

9.3 串比较指令

语句格式：

CMPSB　　CMPSW

CMPSD　　CMPS　OPD, OPS

功能：(1) ([ESI]) – ([EDI]) , 设置标志位

(2) 若 DF=0, 则 (ESI)和(EDI)

增1、2、4（字节、字、双字操作）

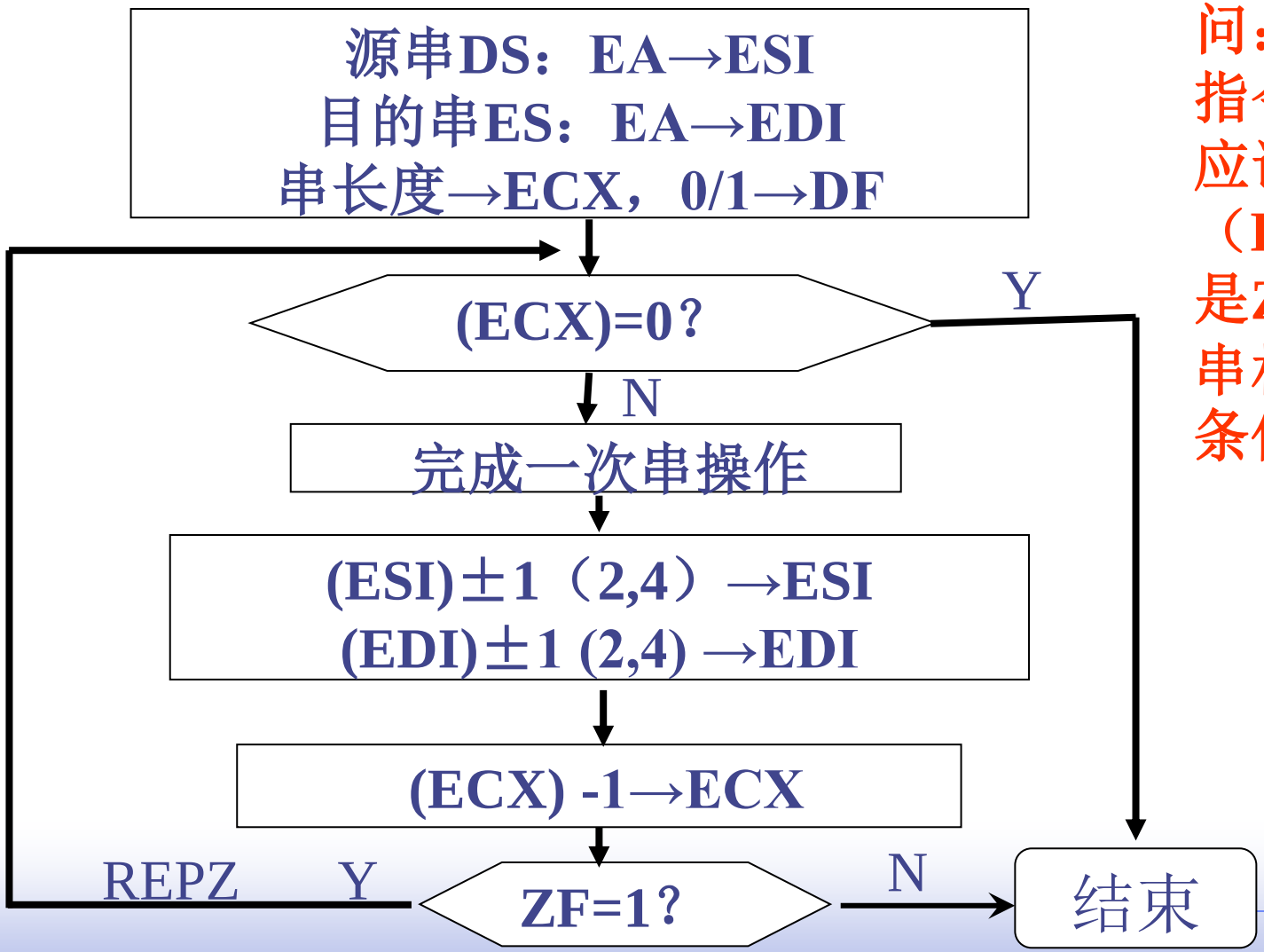
若DF=1,则(ESI)和(EDI)减1或2、4

9.3 串比较指令

例：输入一个串，判断该串是否为 ‘masm’.
若是，则输出 equal, 否则 not equal.

- 定义一个变量，存放串 ‘masm’
- 定义一个输入缓冲区，存放输入串。
- 先比较两串长度是否相等，不等，则显示 not equal;
- CMPSB
- 若 ([ESI])==([EDI]), 即ZF=1,要继续比较。
- 使用前缀 REPZ / REPE
  ZF=1时重复执行,直到（ECX）=0

9.3 串比较指令



问：循环比较指令结束时，应该用（ECX)=0 还是ZF=0作为串相等的判断条件？

9.3 串比较指令

注意：

（1）ZF 是根据串比较指令设置的，而不是最后的 (ECX)-1 ->ECX。

（2）先执行比较，后修改 ESI、EDI

（3）两个串是否相等，要用ZF来判断

（4）若串不等，ESI 指向第一个不相等的字符的下一个字符。

9.4 串搜索指令

语句格式：

SCASB　　SCASW

SCASD　　SCAS　OPD

功能：

(1) (AL/AX/EAX) – ([EDI])，设置标志位

(2) 若 DF=0, 则 (EDI) 增1,2,4 (字节，字,双字)

若 DF=1, 则 (EDI) 减1,或2,或4

9.4 串搜索指令

例：在一个字符串中，找第一个非空格字符。

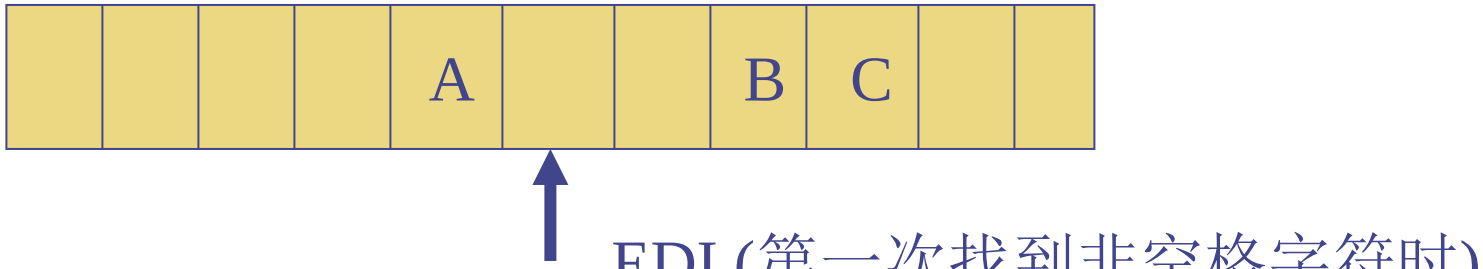
从首字符开始，判断当前字符是否为空格，若是空格，则继续判断，直到 第一个非空格字符出现，或者字符串扫描完。

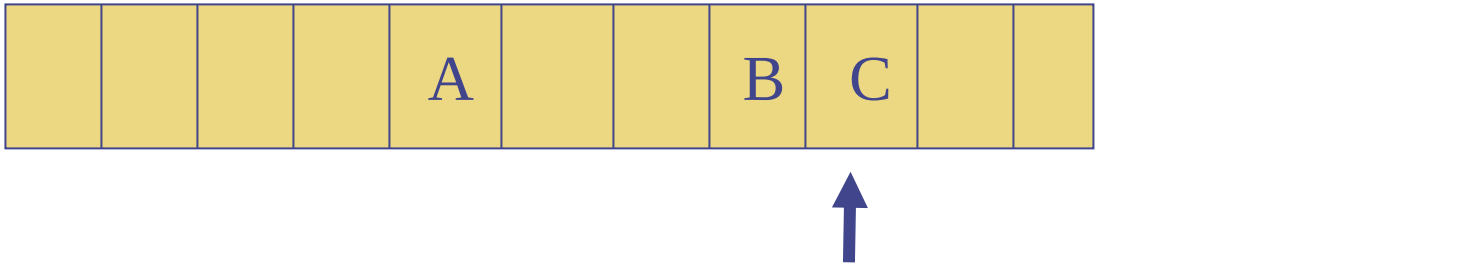
REPZ SCASB

循环条件 (ECX)<>0 且 ZF=1

9.4 串搜索指令

例： 统计显示一个串中非空格字符的个数。





（讨论程序结束的条件 (ECX) =0？ ZF=0？ ）

9.4 串搜索指令

例：在串中找第一个空格字符。

REPNZ　SCASB

9.5 从源串中取数指令

语句格式：

LODSB

LODSW

LODSD

LODS　OPS

功能：　(1) (DS:[ESI]) → AL / AX / EAX

(2) 若 DF=0, (ESI) 如何变化?

若DF=1,则(ESI)如何变化?

9.6 往目的串中存数指令

语句格式：

STOSB

STOSW

STOSD

STOS　OPD

功能：　(1) (AL / AX /EAX) → ES:[EDI]

(2) 若 DF=0，(EDI)如何变化？

若DF=1，(EDI) 如何变化？

9.6 往目的串中存数指令

对于一个C语言程序中的函数（Debug 版），反汇编后，在函数开头可看到的代码：

```
lea         edi,[ebp-0E4h]
mov         ecx,39h
mov         eax, 0CCCCCCCCh
rep stos    dword ptr es:[edi]
```

串操作指令的综合示例

输入一个串，判断是否为一个命令列表中的命令，若是，则调用相应的处理程序。否则，显示出错提示。

(要求能够连续处理命令，直到输入空串)

命令列表：

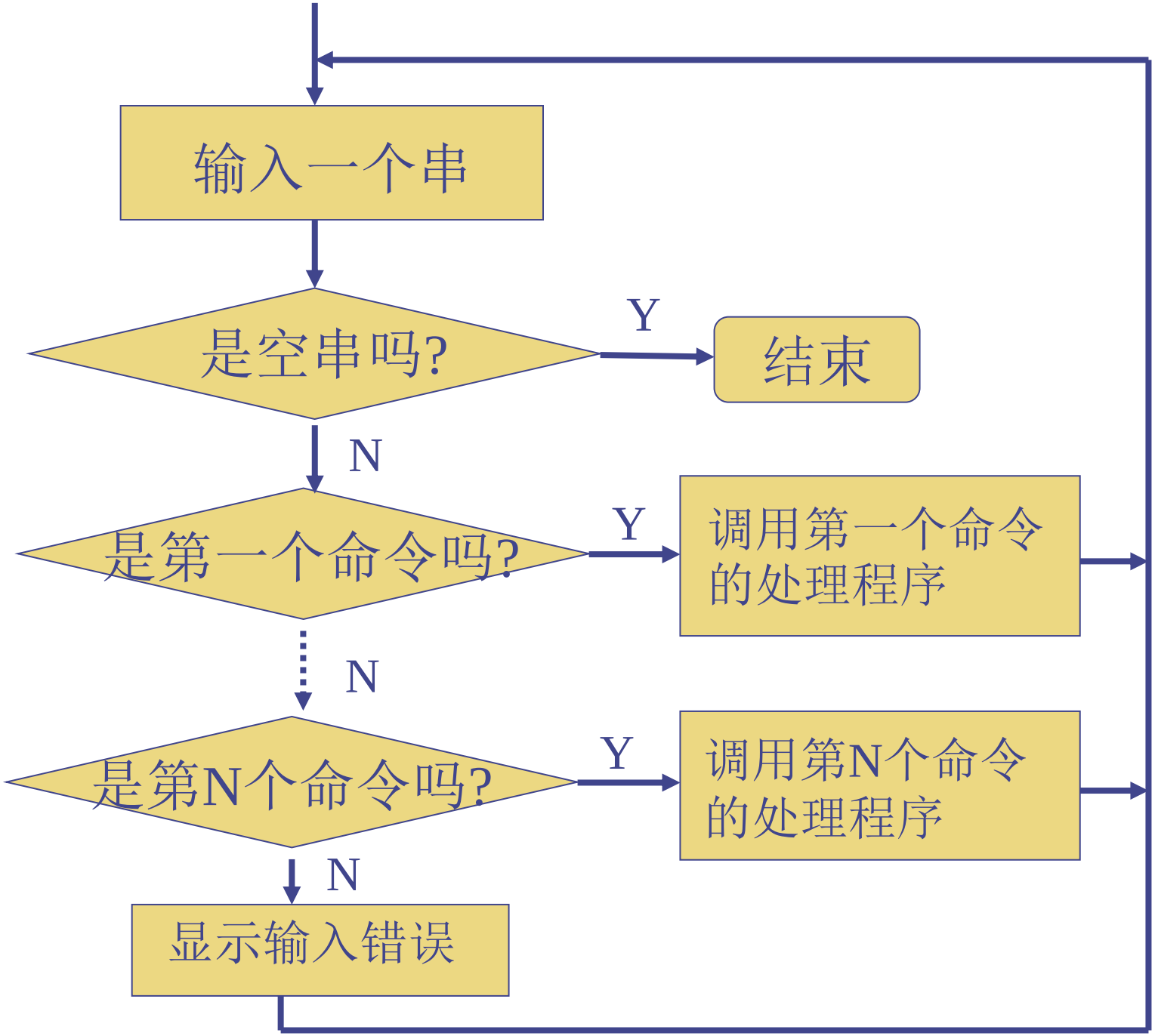
DIR

COPY

RENAME

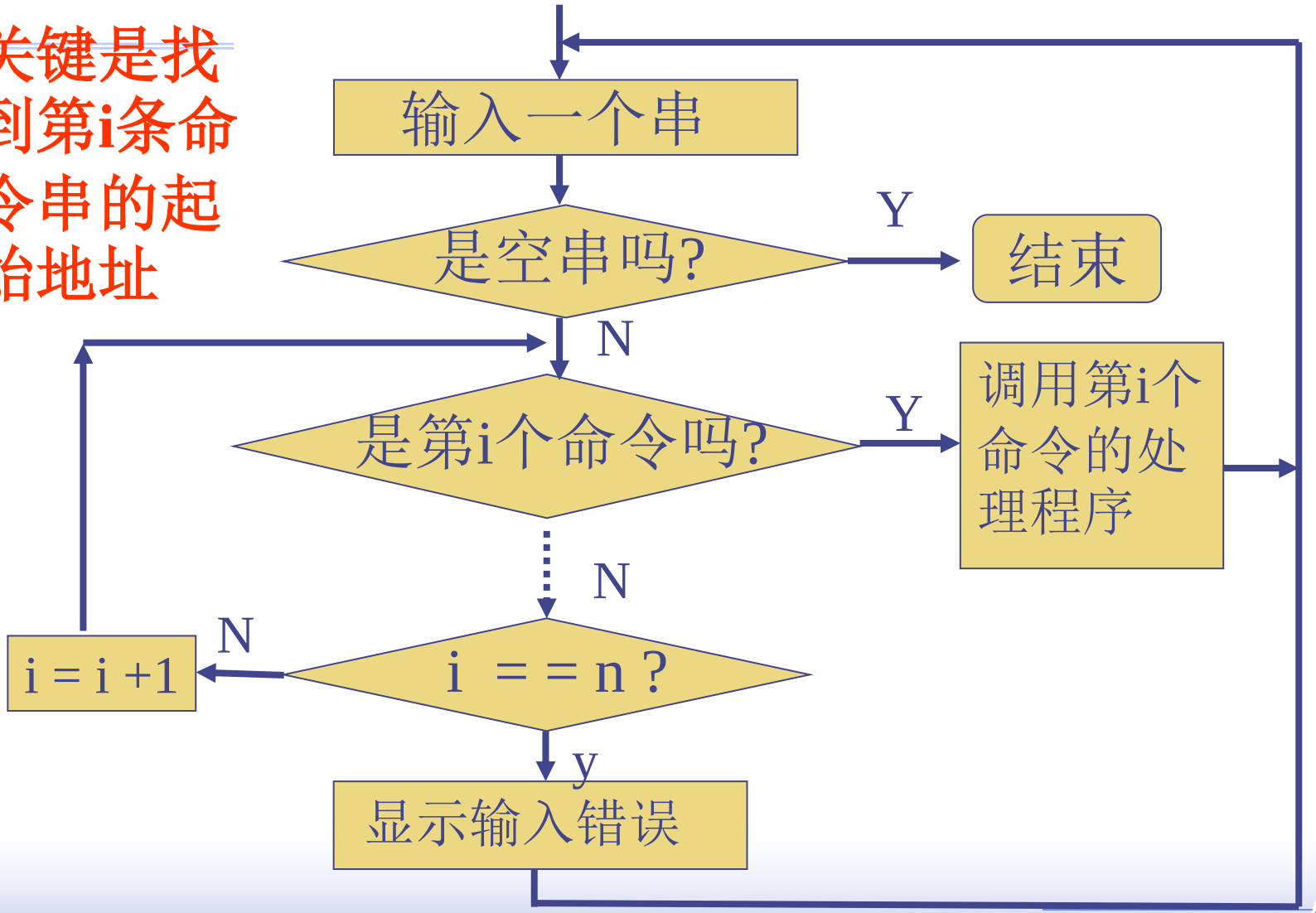
DATE

DEL ……


输入一个串
是空串吗?
Y
结束
N
是第一个命令吗?
Y
调用第一个命令
的处理程序
N
是第N个命令吗?
Y
调用第N个命令
的处理程序
N
显示输入错误

串操作指令的综合示例

关键是找到第i条命令串的起始地址



串操作指令的综合示例

方法一： 地址表法

```
MSG1    DB   4,’DIR’
MSG2    DB   7,’RENAME’
MSG3    DB   4,’DEL’
MSG4    DB   5,’COPY’
ADDR_MSG    DD   MSG1,MSG2,MSG3,MSG4
ADDR_PRO    DD   DIR,  RENAME,DEL,COPY
```

命令比较时：MOV EDI, ADDR_MSG[EBX*4]

子程序调用：CALL ADDR_PRO[EBX*4]

串操作指令的综合示例

方法 二:

```
TAB   DB    4,’DIR’
      DD    DIR
      DB    7,’RENAME’
      DD    RENAME
      DB    4,’DEL’
      DD    DEL
      DB    5,’COPY’
      DD    COPY
```

串操作指令的综合示例

设(ECX) 是第 i个串的长度,

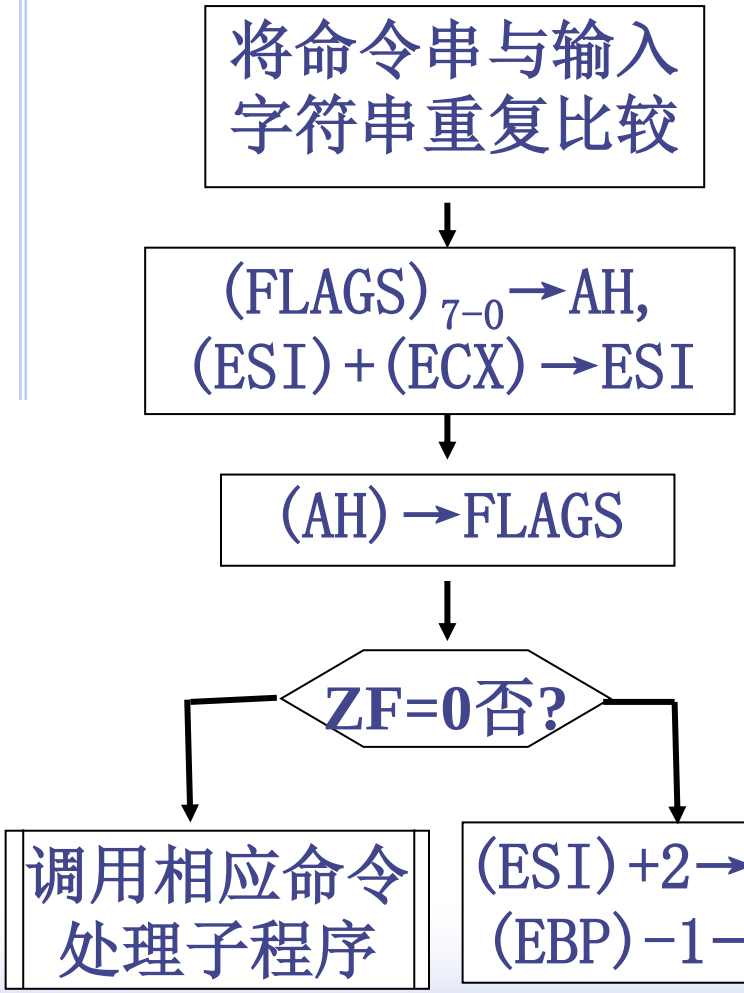
ESI 指向该串的首地址.

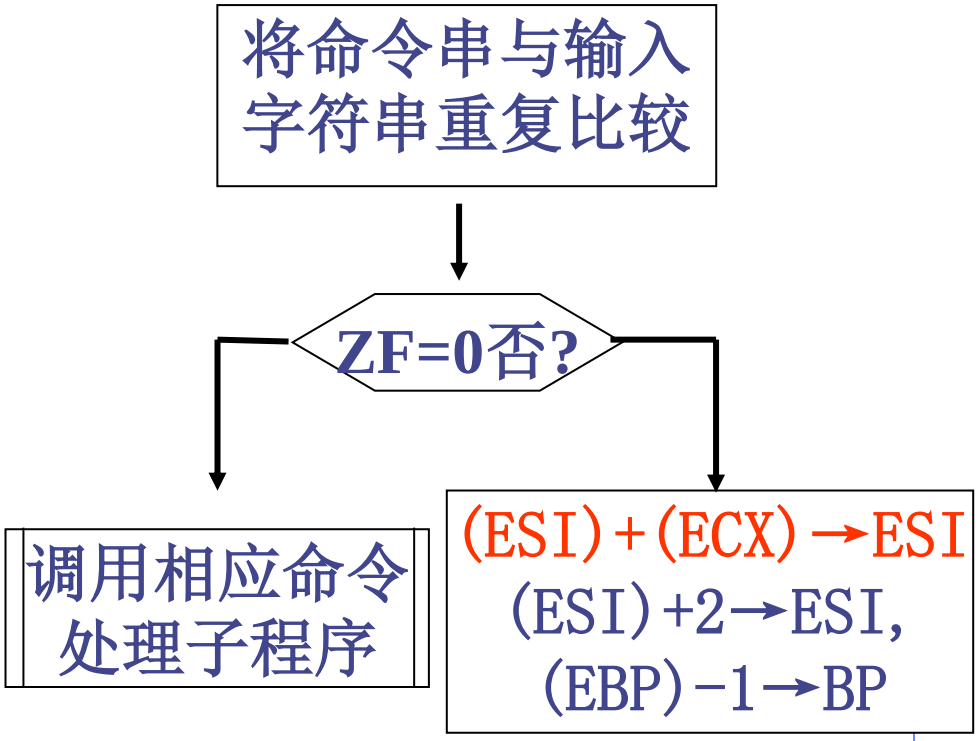
用带前缀REPZ串比较指令,和输入串比较.

若不相等,则:

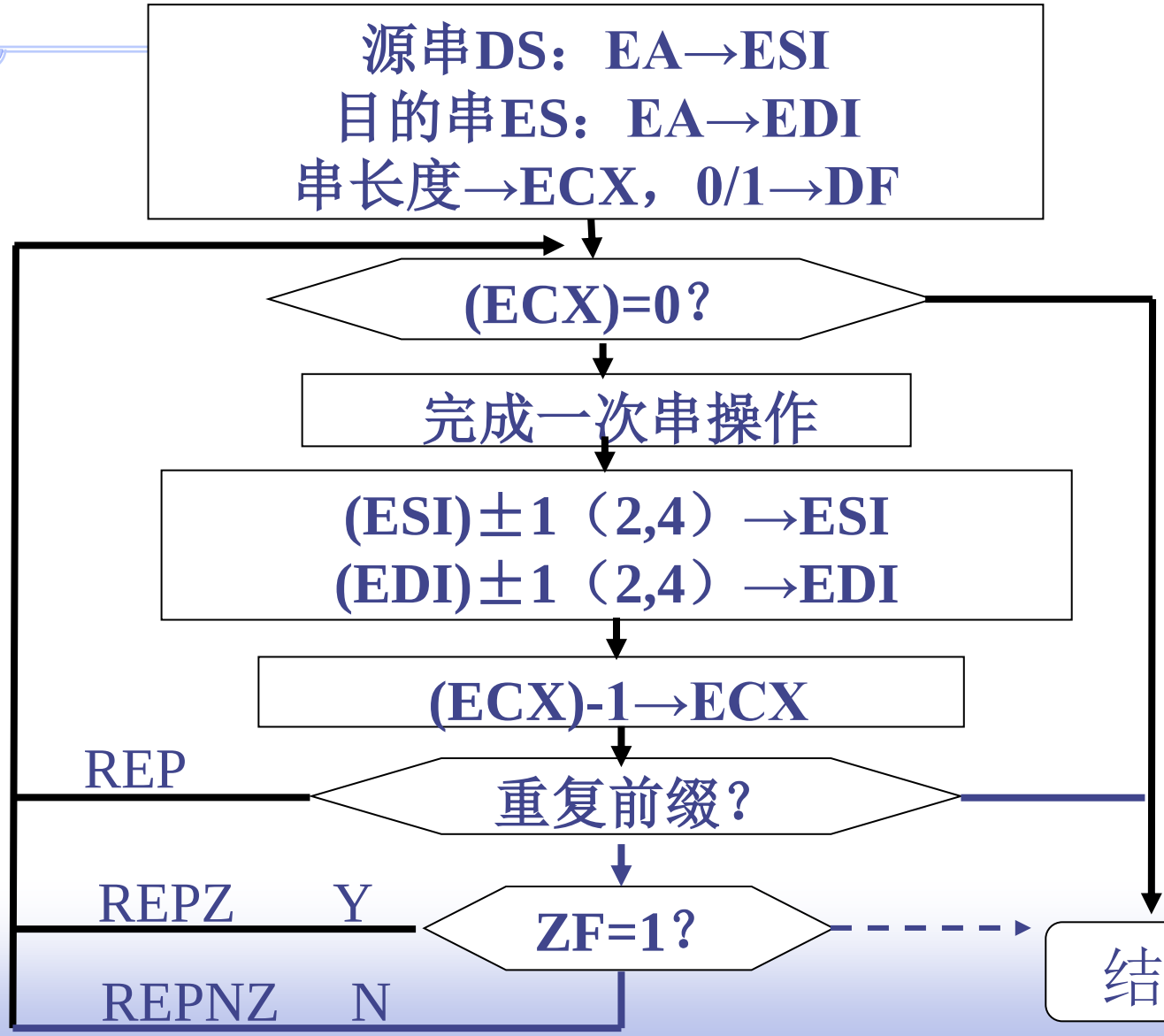
(ECX) + (ESI) 为串尾的下一个单元的地址

串操作指令的综合示例


将命令串与输入字符串重复比较
(FLAGS)7-0→AH,
(ESI)+(ECX)→ESI
(AH)→FLAGS
ZF=0否?
调用相应命令处理子程序
(ESI)+2→ESI,
(EBP)-1→EBP


将命令串与输入字符串重复比较
ZF=0否?
调用相应命令处理子程序
(ESI)+(ECX)→ESI
(ESI)+2→ESI,
(EBP)-1→BP

第9章 串处理程序设计



有前缀时，
执行的顺序

ZF 的改变

第9章 串处理程序设计

串传送指令　MOVSB、MOVSW、MOVSD

串比较指令　CMPSB、CMPSW、CMPSD

串搜索指令　SCASB、SCASW、SCASD

从源串中取数　LODSB、LODSW、LODSD

向目的串中存数　STOSB、STOSW、STOSD

封装了串操作指令的 C 库函数

	memcpy	memcmp	memset
VS	strcpy	strcmp	strset

第9章 串处理程序设计

void *memchr(const void *buf, int ch, size_t count);
功能：从buf所指内存区域的前count个字节查找字符ch。

int memcmp(const void *str1, const void *str2, size_t n));
功能：把存储区 str1 和 str2 的前 n 个字节进行比较。

void *memcpy(void *str1, const void *str2, size_t n);
功能：从存储区 str2 复制 n 个字节到存储区 str1。

void *memset(void *str, int c, size_t n);
功能：str 所指向的存储区的前 n 个字符复制为 c 。

void *memmove(void *str1, const void *str2, size_t n);
功能：与 memcpy()类似，但memmove能够保证源串在被覆盖之前将重叠区域的字节拷贝到目标区域中，复制后源区域的内容会被更改。

第9章 串处理程序设计

源串指针：　　DS：ESI
目的串指针：　ES：EDI
重复计数器：　ECX
中间寄存器：　AL / AX / EAX

重复前缀：

REP　　**（ECX）<>0 时重复执行**
REPE　　**（ECX）<>0 且 ZF=1时重复执行**
REPNE　**（ECX）<>0 且 ZF=0时重复执行**

测验

```
int a[100];
int b[100];
int i;
for (i = 0;i < 100;i++)
        b[i] = a[i];
```

Q：有无其他方式实现上述功能？

```
memcpy(b, a, sizeof(int) * 100);
```

memcpy 的实现代码片段

```
503538D0   push      edi
503538D1   push      esi
503538D2   mov       esi,dword ptr [esp+10h]   //源串 首地址
503538D6   mov       ecx,dword ptr [esp+14h]   //串长
503538DA   mov       edi,dword ptr [esp+0Ch]   //目的地首地址
503538DE   mov       eax,ecx
503538E0   mov       edx,ecx
503538E2   add       eax,esi     //  源串结束地址
503538E4   cmp       edi,esi
503538E6   jbe       503538F0
503538E8   cmp       edi,eax
503538EA   jb        50353B84
503538F0   cmp       ecx,20h
503538F3   jb        50353DCB
503538F9   cmp       ecx,80h
503538FF   jae       50353914
```

memcpy 的实现代码片段

```
50353901   bt          dword ptr ds:[5036A024h],1
50353909   jb          50353D9D
5035390F   jmp         50353AF7
50353914   bt          dword ptr ds:[5036A2E0h],1
5035391C   jae         50353927
5035391E   rep movs    byte ptr es:[edi],byte ptr [esi]
50353920   mov         eax,dword ptr [esp+0Ch]
50353924   pop         esi
50353925   pop         edi
50353926   ret
```

bt : Bit test ; CF ← selected bit