高等学校教材

《微机原理与接口技术》——学习指导与实验

雷丽文　蔡征宇　缪均达　编

Publishing House of Electronics Industry

北京 · BEIJING

内 容 简 介

本书以 16 位微机为对象，全书分为三部分：第一部分习题与解答，列入了《微机原理与接口技术》教材各章全部习题与思考题，并提供了解答；第二部分实验教程，包括软件和硬件共 25 个实验，涵概了课程的主要内容；第三部分提供了实验中全部程序的清单。所有程序均在 PC 系列微机的 DOS 操作系统环境下运行通过，具有一定的实用价值。

本书是《微机原理与接口技术》教材的辅助用书，可供非计算机类机、电专业本科、专科学生作为实验课教材或参考书，也适合从事微机应用的工程技术人员阅读参考。

丛 书 名：高等学校教材
书　　名：《微机原理与接口技术》
——学习指导与实验
编　　者：雷丽文　蔡征宇　缪均达
责任编辑：赵家鹏
特约编辑：天　马
排版制作：电子工业出版社计算机排版室
印 刷 者：
出版发行：电子工业出版社　URL：http://www.phei.com.cn
北京市海淀区万寿路 173 信箱　邮编 100036
经　　销：各地新华书店
开　　本：787×1092　1/16　印张：18.5　插页：1　字数：473.6 千字
版　　次：1998 年 8 月第 1 版　1999 年　月第 1 次印刷
书　　号：ISBN 7-5053-5220-2 / G·424
定　　价：　元
凡购买电子工业出版社的图书，如有缺页、倒页、脱页、所附磁盘或光盘有问题者，请向购买书店调换。
若书店售缺，请与本社发行部联系调换。电话：68279077

前　言

《微机原理与接口技术》一书(雷丽文等编著,电子工业出版社,1997 年 2 月)面世一年多了。为了便于组织教学活动和读者自学,我们编写了这本辅助教材。

全书由三部分组成:

第一部分习题解答,给出了《微机原理与接口技术》教材中的全部习题与思考题的解答。凡需上机操作或检验的习题解答,已全部在机上运行通过。

第二部分实验教程,包括 25 个实验,其中软件实验和硬件实验各占一半左右。这些实验,基本涵概了微机原理与接口技术课程的主要内容。软件实验侧重于培养 16 位微机的汇编语言程序设计与调试能力,硬件实验则以 intel8086 系列微处理器常用外围器件和存储器芯片与系统总线的接口技术为重点。对每个实验,都提出了明确而具体的目的,较详细地叙述了实验内容、方法与步骤,给出了参考程序的流程图和硬件实验的电路图。其中不少内容是教材中有关章节的补充和扩展。

第三部分列出了 25 个实验中全部程序的清单。这些程序具有一定的实用参考价值。

全部实验均在 PC 系列微机(具有 XT 总线或者 ISA 总线)的 DOS 操作系统环境下运行通过。

本书可作为非计算机类工科专业微机原理与接口技术类实验课教材,也可作为学生的辅导读物。对从事微机应用的工程技术人员也有一定的参考价值。

本书由雷丽文主持编写,参加编写的有蔡征宇、缪均达。朱晓华、马玲、谭文三位老师提供了支持和帮助,谨向他们致谢。在编写过程中,参考借鉴了若干有关的出版物,已列入参考文献中,在此向有关作者表示感谢。

限于水平,难免有疏漏之处,尚祈读者不吝指教。

编　　者

一九九八年八月

目　　录

第一部分　习题解答参考

第二部分　实 验 教 程

软件实验部分

硬件实验部分

第三部分　实验参考程序

第一部分　习题解答参考

第一章　微 机 基 础

1-1　微处理器、微机和微机系统三者之间有什么不同？

答：微处理器是构成微机的一个核心部件，通常是包含有运算器和控制器的一块集成电路。它具有解释指令、执行指令和与外界交换数据的能力。微处理器也称为中央处理单元 CPU。

微机是通过总线把 CPU、I/O 接口电路和半导体存储器(ROM 和 RAM)组合在一起构成的一台计算机的物理装置。

微机配上外部设备、系统电源和系统软件就构成一个微机系统，简称系统机。其中，所有的物理装置的集合称为硬件系统，也称为裸机或硬核，它是计算机存储和执行程序、实现各种功能的物质基础。硬件系统必须在由系统软件和应用软件构成的软件系统的配合下，构成一个微机系统，才能完成各种工作。

人们通常所说的微机都是指的系统机。

1-2　8 位 CPU 在内部结构上由哪几部分组成？

答：8 位 CPU 在内部结构上由寄存器阵列、算术逻辑运算单元 ALU、控制器和内部总线及缓冲器等部分组成。

1-3　8088/8086 的总线接口部件有什么功能？其执行部件又有什么功能？

答：总线接口部件 BIU 负责与存储器和 I/O 端口传送数据，包括从存储器中取指令以及配合执行单元对存储单元或外设接口进行读或写操作。执行部件 EU 负责指令的执行。

1-4　8088/8086 状态标志和控制标志分别有哪些？

答：8088/8086 的状态标志有：进位标志 CF、零标志 ZF、符号标志 SF、溢出标志 OF、奇偶标志 PF 和辅助进位标志 AF；控制标志有：方向标志 DF、中断允许标志 IF 和跟踪标志 TF。

1-5　8088/8086 微处理器和以往的 8 位微处理器相比在执行指令方面有什么不同？这样的设计思想有什么优点？

答：8 位 CPU 的指令执行方式是取指、执行指令两个阶段循环进行的方式，而 8088/8086 的指令执行方式是流水线方式。因为 8088/8086CPU 由 BIU 和 EU 两部分组成，取指和执指是分开而且可以重叠进行的。这样使在执行指令的同时可以取下一条或下几条指令送至指令队列中排队，于是执行完一条指令后便可立即执行下一条指令，从而提高了 CPU 的效率和整机运行速度。

1-6　8088/8086 微处理器有哪些寄存器？通用寄存器中哪些可以作地址指针用？

答：8088/8086 微处理器的通用寄存器有 AX、BX、CX、DX、SP、BP、SI、DI；控制寄存器有 IP、

F;段寄存器有 CS、DS、ES 和 SS。通用寄存器中,BX、SI 和 DI 可作地址指针,BP 只能在堆栈段当地址指针,SP 专作堆栈指针用。AX、CX 和 DX 不能作地址指针用。

1-7 8088/8086 20 位物理地址是怎样形成的?当 CS=2000H,IP=0100H,其指向的物理地址等于多少?

答:8088/8086 20 位物理地址=段地址 * 10H+偏移地址。

当 CS=2000H, IP=0100H 时,物理地址是 20100H。

1-8 将下列十进制数分别转换为二进制数、八进制数和十六进制数:

128,511,1024,65535,1048575,512、0.625,27/32,0.4375。

解:$128=2^7=10000000B=200Q=80H$

$511=2^9-1=111111111B=777Q=1FFH$

$1024=2^{10}=10000000000B=2000Q=400H$

$65535=2^{16}-1=1111111111111111B=177777Q=FFFFH$

$1048575=2^{20}-1=11111111111111111111B=3777777Q=FFFFFH$

$512=2^9=1000000000B=1000Q=200H$

$0.625=0.101B=0.5Q=0.AH$

$27/32=108/128=0.1101100B=0.660Q=0.D8H$

$0.4375=0.0111B=0.34Q=0.7H$

1-9 将下列二进制数转换成十进制及十六进制数:

1100110101B,101101.1011B

解:1100110101B=335H=821

101101.1011B=2D.BH=45.6875

1-10 $(4578)_{10}=(\underline{\qquad\qquad})_{BCD}=(\underline{\qquad\qquad})B$

解:$(4578)_{10}=(\underline{100010101111000})_{BCD}=(\underline{1000111100010})B$

1-11 写出下列数的原码、反码、补码和移码(该字长为 8 位):15,-20,-27/32

解:

真值	原码	反码	补码	移码
15	00001111	00001111	00001111	10001111
-20	10010100	11101011	11101100	01101100
-27/32	1.1101100	1.0010011	1.0010100	0.0010100

1-12 16 位无符号定点整数的数值表示范围为多少?8 位补码表示的数值范围是多少?16 位补码呢?(整数情况)

答:16 位无符号定点整数范围是 0~65535。8 位补码表示的整数范围是-128~+127。16 位补码表示的整数范围是-32768~+32767。

1-13 当两个正数相加时,补码溢出意味着什么?两个负数相加能产生溢出吗?试举例说明。

答:两个正数相加,补码溢出意味着其和正方向超出数域范围而产生错误结果。例如,当 $n=8$ 时,用补码求 109 加 84 之和,所得结果(也为补码)为 11000001,其真值为-63,显然出错了。原因是 193>127。两个负数相加也可能产生溢出。当和小于-128 时产生负向溢出。例如用补码求-109 加-84 之和,得结果□ 00111111(补码),其真值为+63。出错原因是-193<-128。

1-14 试写出 3,A,ESC,CR,SP 的 ASCII 码。

答：

字符或命令	3	A	ESC	CR	SP(空格)
相应的 ASCII 码	33H	41H	1BH	0DH	20H

1-15　每个汉字的编码由几个字节组成？计算机中如何区别 ASCII 码和汉字内码？

答：在计算机中，每个汉字的编码由两个字节组成。 ASCII **码的最高位** $b_7=0$，**而汉字内码的最高位** $b_7=1$。**计算机根据字节的最高位来区分** ASCII **码和汉字内码。**

1-16　设有两个正的浮点数 $N_1=2^{P_1}*S_1$，$N_2=2^{P_2}*S_2$，

（1）**若** $P_1>P_2$，**是否有** $N_1>N_2$？

（2）**若** S_1，S_2 **是规格化的，上述结论是否正确？**

答：（1）$P_1>P_2$，**不一定** $N_1>N_2$。

（2）**若** S_1，S_2 **是规格化的，则当** $P_1>P_2$ **时有** $N_1>N_2$。

第二章　8088/80286 的指令系统

2-1　试述 8088 微处理器的各种寻址方式，并写出各种寻址方式的传送指令 1～2 条（源操作数与目的操作数寻址）。

答：寻址方式是指计算机在执行指令时寻找操作数的方式。8088CPU 有以下几种寻址方式：

1. **立即寻址。操作数（仅限于源操作数）直接放在指令中，紧跟在操作码后面，与操作码一起放在码段区域中。例如** MOV AX,50H；MOV DX,30。

2. **寄存器寻址。操作数在 CPU 的内部寄存器中。例如** MOV AX,BX；MOV DS,AX。

3. **直接寻址。指令中直接给出操作数存放地址的 16 位偏移量，它与操作码一起存放在码段中。实际的操作数一般在数据段中，实际地址为** DS:16 **位偏移量。直接寻址可以在 64K 字节范围内寻找操作数。例如** MOV AX,[2000H]；MOV BUFFER,AX。

4. **寄存器间接寻址。操作数的 16 位偏移地址存放在** BP、BX、DI、SI **这 4 个内部寄存器当中的任意一个寄存器里。若无特殊说明，用** BX、DI、SI **间址时，对应数据段寄存器** DS；**若用** BP **间址，则对应堆栈段寄存器** SS。**例如** MOV [DI],AX；MOV DX,[SI]。

5. **变址寻址。它是在寄存器间址基础上再加上一个 16 位地址偏移量** COUNT。**操作数的有效地址为上述 4 个内部寄存器之一的内容与** COUNT **之和。例如** MOV AX,COUNT[DI]；MOV COUNT[ST],AX。

6. **基址加变址寻址。在这种寻址方式下，操作数的有效地址由基址寄存器，变址寄存器和 16 位偏移量之部分（相加）组成。通常把** BX **和** BP **作为基址寄存器，并用来决定段指针：** BX **对应于** DS,BP **对应于** SS；SI **和** DI **作为变址寄存器。例如** MOV AX,[BX＋DI＋100]；MOV 100H[BX][DI],DX。

在后 4 种寻址方式中，操作数均存放在存储器中。

2-2　对 8088/8086 CPU **指出下列指令中哪些是错误的，并说明原因。**

(1) MOV BL,AX　　(2) MOV 100,CX

(3) MOV[SI],AX　　(4) MOV CS,AX

(5) MOV[SI],BUFFER　　(6) OUT 541H,AL

(7) IN BL,DX　　(8) LEA BX,AX

(9) MOV BX,2[DI]　　(10) XCHG AL,100

(11) MOV BYTE PTR[BX],1000　　(12) MOV AX,[BP＋4]

(13) MOV AX,CS　　(14) MOV SS,2400H

答：错误指令有：

(1) **源操作数与目的地操作数位数不等。**

(2) **立即数不能作目的地操作数。**

(4) CS **不能作目的地操作数。**

(5) **内存单元之间不能用** MOV **指令传送。**

(6) **大于** 0FFH **的端口地址应放** DX **中。**

(7) 输入指令不能用 BL 寄存器。

(8) LEA 指令的源操作数应是内存操作数,不能是寄存器操作数。

(10) 交换指令不能对立即数操作。

(11) 同(1)。

(14) 立即数不能写入段寄存器。

2-3 连续执行以下指令,填写指令执行结果,并上机验证结果。假设 M 代表存储单元物理地址,[R]代表寄存器间接寻址的存储单元内容,FLAGL 代表标志寄存器低字节,SRC 代表源操作数,DST 代表目的操作数,MOD 代表寻址方式。填空时用 IM 代表立即寻址方式,DRT 代表直接寻址方式,R 代表寄存器寻址方式,RIN 代表寄存器间接寻址方式,IDX 代表变址寻址方式,B&IDX 代表基址加变址寻址方式。

```
;EXSE 2_3
MOV AX,2000H                ;AH=20H     MOD=IM
MOV DS,AX                   ;AL=00H     DS=2000H    MOD=R
MOV SS,AX                   ;SS=2000H   AX=2000H
MOV BX,2050H                ;BH=20H     BL=50H
MOV SI,BX                   ;SI=2050H
MOV DI,3050H                ;DI=3050H
MOV SI,DI                   ;SI=3050H
MOV SP,5FFFH                ;SP=5FFFH
MOV CL,25                   ;CL=19H
MOV BL,CL                   ;CL=19H     BL=19H
MOV AH,0F0H                 ;AH=0F0H
MOV CH,AH                   ;CH=0F0H
MOV BYTE PTR[DI],64         ;[DI]=40H   M=23050H   MOD=DST   RIN
MOV WORD PTR[SI],256        ;DST   MOD=RIN   [ST]=00H   [SI+1]=01H
MOV DL,[SI+1]               ;DL=01H     M=23051H   SRC   MOD=IDX
MOV DH,1[SI]                ;DH=01H     M=23051H   SRC   MOD=IDX
MOV AL,1+[SI]               ;AL=01H     M=23051H
MOV WORD PTR[BX][SI],34     ;[BX+SI]=22H    DST    MOD=B&IDX
                            ;[BX+SI+1]=00H
MOV [BX+SI+4],BX            ;[BX+SI+4]=19H,20H   M=2506DH,2506EH
MOV 2[BX+SI],BX             ;DST   MOD=B&IDX
MOV BP,2[BX+DI]             ;BP=2019H   M=2506BH,2506CH   SRC MOD=B&IDX
MOV [BP][DI],BX             ;[BP][DI]=19H,20H   M=25069H,2506AH
MOV AX,[BP][DI]             ;AX=2019H   M=25069H,2506AH
MOV BL,AL                   ;BL=19H
MOV ES,BX                   ;ES=2019H
PUSH BX                     ;SP=5FFDH   [SP]=19H   [SP+1]=20H
PUSH DI                     ;SP=5FFBH   [SP]=50H   [SP+1]=30H
POP CX                      ;SP=5FFDH   CX=3050H
POP DX                      ;SP=5FFFH   DX=2019H
XCHG AX,BP                  ;AX=2019H   BP=2019H
XCHG DH,BL                  ;DH=19H     BL=20H
```

```
SAHF                    ;AH=20H
CMC                     ;CF=1
LAHF                    ;AH=03H      (ONLY 5 FLAG BITS)
STD                     ;DF=1
CLI                     ;IF=0
INT 20H
```

2-4　连续执行以下指令，填写执行指令的结果，并上机核对结果。

```
;EXSE2-4
MOV AX,3502H            ;AL=02H      AH=35H      CF=0
MOV DS,AX               ;DS=3502H    AH=35H      CF=0
ADD AL,AH               ;AL=37H      AH=35H      CF=0
MOV DX,258              ;DH=01H      DL=02H      CF=0
SUB AX,DX               ;AX=3435H    DX=0102H    CF=0
MOV CX,0E0BAH           ;CX=E0BAH    CF=0
MOV AX,2400H
ADD AX,CX               ;AX=04BAH    CX=E0BAH    CF=1
ADC CX,AX               ;CX=E575H    AX=04BAH    CF=0
MOV SI,4000H            ;SI=4000H
MOV [SI],CX             ;[SI]=75H    [SI+1]=E5H  M=39020
ADC [SI],AL             ;[SI]=2FH    CF=1
DEC BYTE PTR[SI]        ;[SI]=2EH    CF=1
MOV AX,09H              ;AX=0009H
ADC AX,09H              ;AX=0013H
AAA                     ;AX=0109H    AF=1        CF=1
ADD AL,09H              ;AL=12H
DAA                     ;AL=18H      AF=1
ADD AL,98H              ;AL=B0H      CF=0        AF=1
DAA                     ;AL=16H      CF=1
MOV AL,5                ;AL=05H
NEG AL                  ;AL=FBH
MOV BX,-15              ;BX=FFF1H
NEG BX                  ;BX=000FH
CMP BH,BL               ;BH=00H      BL=0FH
MOV DL,20               ;DL=14H
MOV AL,5                ;AL=05H
MUL DL                  ;AX=0064H    DL=14H
MOV CH,4
DIV CH                  ;AX=0019H    CH=04H
MOV DX,0F000H           ;DX=F000H
MUL DX                  ;AX=7000H    DX=0017H
MOV AL,5
NEG AL                  ;AL=FBH
CBW                     ;AH=FFH      AL=FBH
MOV DX,5                ;DX=0005H
```

```
IMUL DX                    ;AX=FFE7H    DX=FFFFH
MOV AX,5                   ;AX=0005H
CWD                        ;AX=0005H    DX=0000
MOV CX,5                   ;CX=0005
NEG CX                     ;CX=FFFBH
IDIV CX                    ;AX=FFFFH    DX=0000
MOV AX,05H                 ;AX=0005
NOT AX                     ;AX=FFFAH
MOV BL,16H                 ;BL=16H
AND AH,BL                  ;AH=16H      BL=16H
MOV CX,0F54BH              ;CX=F54BH
OR AX,CX                   ;AX=F7FBH
XOR CX,AX                  ;AX=F7FBH    CX=02B0H
XOR CX,AX                  ;AX=F7FBH    CX=F54BH
MOV AX,0FFFFH
XOR CX,AX                  ;CX=0AB4H
TEST CX,1234H              ;CX=0AB4H    PF=0    ZF=0    CF=0    SF=0
MOV AL,9
SAR AL,1                   ;AL=04H
MOV CL,4                   ;CX=0A04H
PUSH CX
SHL AL,CL                  ;AL=40H      CX=0A04H      CF=0
POP CX                     ;CX=0A04H
MOV BX,850H
RCL BX,CL                  ;BX=8500H    CF=0
RCR BX,CL                  ;BX=850H     CF=0
MOV DI,4050H
MOV [DI],BX
SAR BYTE PTR [DI],1        ;[DI]=28H    M=39070H
CLC                        ;CF=0
CMC                        ;CF=1
STC                        ;CF=1
CLD                        ;DF=0
STD                        ;DF=1
CLI                        ;IF=0
STI                        ;IF=1
INT 20H
```

2-5　将共阳 LED 显示器显示的 0～9 数字的七段码列成一张表存在数据段。欲从 10H 号外设端口读入 0～9 中间的一个数字(ASCII 码),将它转换为七段码后输出到 20H 号端口去,写出完成上述任务的指令序列。

解:完成以上任务指令如下:

```
MOV BX,OFFSET TABLE        ;BX 指向表首址
IN AL,10H
```

```
    SUB AL,30H                ;AL 中 ASCII→BCD
    XLAT TABLE                ;查表 A 中得七段码
    OUT 20H,AL
TABLE DB 0C0H,0F9H            ;此表在数据段中
    DB 0A4H,0B0H
    DB 99H,92H
    DB 82H,0F8H
    DB 80H 90H
```

2-6　写出根据 BX 寄存器中的 $b_5=0$ 转到 L1 的指令序列。若 $b_5=1$ 转移，指令应作何修改？

解：根据 BX 中的 $b_5=0$ 转：

```
    TEST BX,0020H
    JZ L1
```

根据 BX 中的 $b_5=1$ 转 L1：

```
    TEST BX,0020H
    JNZ L1
```

2-7　写出将 BX 和 SI 寄存器内容进行交换的堆栈操作指令序列，并画出堆栈区和 SP 的内容变化过程示意图。

解：堆栈操作指令序列

```
    PUSH BX
    PUSH SI
    POP BX
    POP SI
```

堆栈内容示意图如图 1-1 所示。

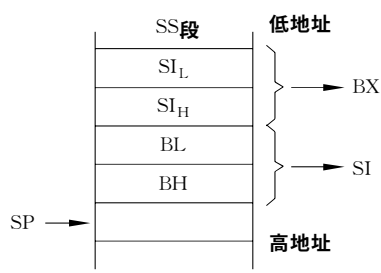

图 1-1　堆栈操作示意图(题 2-7 解)

2-8　设 a、b、c、d 是互不相等的 8 位符号数(补码)，并假设加减法运算均不产生溢出。试写出完成下列运算的程序段。结果放在 DX 和 AX 中。

(1)$(a+b)/(c-d)$

(2)$(a+b)*(c-d)$

解：(1)$(a+b)/(c-d)$的程序段如下：

```
    MOV AL, a
    ADD AL, b
    MOV BL, c
    SUB BL, d
```

```
CBW              ;AL 中的符号位扩展到 AX 中
IDIV BL          ;AX/BL 商在 AL 中,余数在 AH 中
```

(2)$(a+b)*(c-d)$的程序段如下：

```
MOV AL, a
ADD AL, b
MOV BL, c
SUB BL, d
IMUL BL          ;积在 AX 中,仍为补码
```

本题主要要求掌握 CBW 和 CWD 指令的用法。8 位数除 8 位数应先将被除数扩展为 16 位后再进行除法。符号数的扩展用 CBW 或 CWD 指令,无符号数的扩展是高字节(或字)送入 0。

2-9　利用串操作指令将以 AREA1 起始的区域Ⅰ中的 100 个字节数据传送到以 AREA2 为起始地址的区域Ⅱ(两个区域可能重叠)。

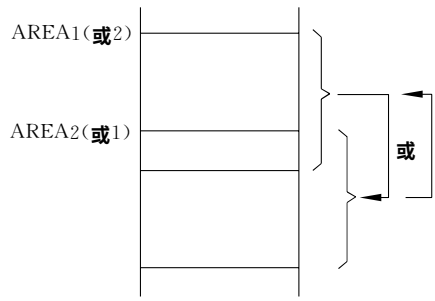

图 1-2　数据传送示意图(题 2-9 解)

解:根据题意可画出数据传送示意图如图 1-2。AREA1 和 AREA2 的取值可能有三种情况：AREA1=AREA2,无需传送;AREA1<AREA2 时,应从数据区Ⅰ的最大地址单元开始传送;AREA1>AREA2 时,应从数据区Ⅰ的最小地址单元开始传送。后两种情况下的 DF 取值不同。另外注意用串操作指令传送时,必须对 ES 赋值。程序段如下：

```
;EXSE2-9
        MOV AX,2000H
        MOV DS,AX
        MOV ES, AX
        LEA SI, AREA1
        LEA DI, AREA2
        MOV CX,100
        CMP SI,DI
        JE DONE
        JA DF0
        STD              ;置 DF=1
        ADD SI,99        ;数据传送从高地址开始
        ADD DI,99
        JMP TRAN
DF0:CLD
```

```
TRAN:REP MOVSB
DONE:INT 20H
```

2-10　在指令 CMP AX,BX 后面紧跟一条格式为 J＊　L1 的条件转移指令,其中＊可以是 B、NB、BE、NBE、L、NL、LE、NLE 中任一个,如果 AX 和 BX 的内容如下:

	AX	BX
(1)	3500H	3500H
(2)	0ABCDH	7500H
(3)	0FCD0H	0FFE0H
(4)	5678H	4500H
(5)	4567H	0BA00H

对每一组 AX 和 BX 数据,使用哪几种格式的转移指令将引起程序转移到 L1?

答:在＊可取的 8 种形式中,前四种是将参与比较的 AX 和 BX 中的数看成是无符号数,B 含义是低于(即 AX 低于 BX)、NB 是不低于、BE 是低于或等于、NBE 是不低于且不等于;后四种是将 AX 和 BX 中的数看成是符号数(补码),L 含义是小于、NL 是不小于、LE 是小于或等于、NLE 是不小于且不等于。

(1) **只要包含有"等于"的＊号形式均可使程序转移到 L1,所以可取 NB、BE、NL 和 LE。**

(2) **若将 0ABCDH 和 7500H 看成无符号数,则前者高于后者,所以可取 NB、NBE。若将它们看成符号数,前者为负,后者为正,所以可取 L 和 LE。**

(3) **与(2)同理,＊可取 B、BE、L 和 LE 四种。**

(4) **＊号可取 NB、NBE、NL 和 NLE 四种。**

(5) **＊号可取 B、BE、NL 和 NLE 四种。**

2-11　试将 BUFFER 起始的 50 个字节的组合 BCD 码数字转换成 ASCII 码存放于 ASC 为起始地址的单元中。高位 BCD 码数字位于较高地址。

解:组合 BCD 码数字的存放形式是一个字节存放两位 BCD 码数字。用 SI 作为源数据区指针,DI 作为目的地数据区指针,根据题意可画出内存图,如图 1-3 所示。BUFFER 和 ASC 分别代表两个数据区的首地址,通常称之为符号地址。程序如下:

```
;EXSE2_11
        LEA SI,BUFFER          ;SI 指向 BCD 码缓冲区
        LEA DI,ASC             ;DI 指向 ASCII 码缓冲区
        MOV CX,50              ;CX 作计数器
L1:     MOV AL,[SI]
        MOV BL,AL
        AND AL,0FH             ;个位 BCD→ASCII
        OR AL,30H
        MOV [DI],AL            ;存放个位
        INC DI
        MOV AL,BL              ;十位 BCD→ASCII
        PUSH CX                ;保护 CX
        MOV CL,4
        SHR AL,CL
        POP CX                 ;恢复 CX
```

```
        AND AL,0FH
        OR AL,30H
        MOV [DI],AL              ;存放十位
        INC DI
        INC SI
        LOOP LI                  ;50 个未做完,则继续
        INT 20H
```

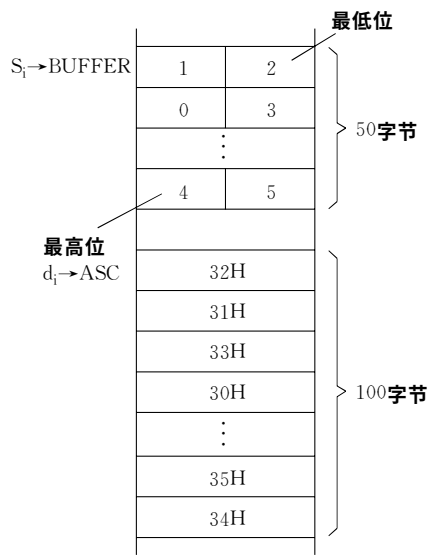

图 1-3　组合 BCD 码至 ASCII 码的转换(题 2-11 解)

程序中用了两条堆栈操作指令,目的是保护用作计数器的 CX 寄存器的值不受 SHR 指令的破坏。

2-12　给以 TAB 为首地址的 100 个 ASCII 码字符添加奇偶校验位(bit7),使每个字节中的"1"的个数为偶数,再顺序输出到 10H 号端口。

解:

```
;EXSE2_12
        MOV SI,OFFSET TAB
        MOV CX,100
L2:     MOV AL,[SI]
        AND AL,0FFH              ;BUILD FLAG
        JP L1
        OR AL,80H
L1:     OUT 10H,AL
        INC SI
        LOOP L2
        INT 20H
```

2-13　已知四字节数存放在 NUM 开始的连续四字节中,高字节位于高地址。试编写将它左移一位的程序段(假设移位后字节数不变)。

解:

```
LEA BX,NUM
```