第 1 章 汇编语言基础知识

〔习题 1.1〕简述计算机系统的硬件组成及各部分作用。

〔解答〕

CPU：包括运算器、控制器和寄存器组。运算器执行所有的算术和逻辑运算；控制器负责把指指令逐条从存储器中取出，经译码分析后向机器发出各种控制命令，并正确完成程序所要求的功能；寄存器组为处理单元提供所需要的数据。

存储器：是计算机的记忆部件，它用来存放程序以及程序中所涉及的数据。

外部设备：实现人机交换和机间的通信。

〔习题 1.2〕明确下列概念或符号：

主存和辅存，RAM 和 ROM，存储器地址和 I/O 端口，KB、MB、GB 和 TB

〔解答〕

主存又称内存是主存储器的简称，主存储器存放当前正在执行的程序和使用的数据，CPU 可以直接存取，它由半导体存储器芯片构成其成本高、容量小、但速度快。辅存是辅助存储器的简称，辅存可用来长期保存大量程序和数据，CPU 需要通过 I/O 接口访问，它由磁盘或光盘构成，其成本低、容量大，但速度慢。

RAM 是随机存取存储器的英语简写，由于 CPU 可以从 RAM 读信息，也可以向 RAM 写入信息，所以 RAM 也被称为读写存储器，RAM 型半导体存储器可以按地址随机读写，但这类存储器在断电后不能保存信息；而 ROM 中的信息只能被读出，不能被修改，ROM 型半导体通常只能被读出，但这类存储器断电后能保存信息。

存储器由大量存储单元组成。为了区别每个单元，我们将它们编号，于是，每个存储单元就有了一个存储地址，I/O 接口是由一组寄存器组成，为了区别它们，各个寄存器进行了编号，形成 I/O 地址，通常称做 I/O 端口。

KB 是千字节、MB 是兆字节、GB 是吉字节和 TB 是太字节，它们都是表示存储器存储单元的单位。

〔习题 1.3〕什么是汇编语言源程序、汇编程序、目标程序？

〔解答〕

用汇编语言书写的程序就称为汇编语言源程序；完成汇编工作的程序就是汇编程序；由汇编程序编译通过的程序就是目标程序。

〔习题 1.4〕汇编语言与高级语言相比有什么优缺点？

〔解答〕

汇编语言与高级语言相比的优点：由于汇编语言本质就是机器语言，它可以直接地、有效地控制计算机硬件，因而容易产生运行速度快，指令序列短小的高效目标程序，可以直接控制计算机硬件部件，可以编写在“时间”和“空间”两方面最有效的程序。

汇编语言与高级语言相比的缺点：由于与处理器密切相关导致通用性差、可移植性差，汇编语言功能有限，又涉及寄存器、主存单元等硬件细节，编写汇编语言比较繁琐，调试起来也比较困难，编译程序产生的目标程序往往比较庞大、程序难以优化，运行速度慢。

〔习题 1.5〕将下列十六进制数转换为二进制和十进制表示

（1）FFH　　（2）0H　　（3）5EH　　（4）EFH

（5）2EH　　（6）10H　　（7）1FH　　（8）ABH

〔解答〕

（1）FFH	11111111B	255D
（2）0H	0B	0D
（3）5EH	1011110B	94D
（4）EFH	11101111B	239D
（5）2EH	101110B	46D
（6）10H	10000B	16D
（7）1FH	11111B	31D
（8）ABH	10101011B	171D

〔习题 1.6〕

将下列十进制数转换为 BCD 码表示

（1）12　　（2）24　　（3）68　　（4）127

（5）128　　（6）255　　（7）1234　　（8）2458

〔解答〕

（1）12	00010010
（2）24	00100100
（3）68	01101000
（4）127	000100100111
（5）128	000100101000
（6）255	001001010101
（7）1234	0001001000110100
（8）2458	0010010001011000

〔习题 1.7〕

将下列 BCD 码转换为十进制数

（1）10010001　（2）10001001　（3）00110110　（4）10010000

（5）00001000　（6）10010111　（7）10000001　（8）00000010

〔解答〕

（1）91

（2）89

（3）36

（4）90

（5）08

（6）97

（7）81

（8）02

〔习题 1.8〕将下列十进制数分别用 8 位二进制数的原码、反码和补码表示

（1）0　（2）-127　（3）127　（4）-57

（5）126　（6）-126　（7）-128　（8）68

〔解答〕

（1）0 +0 00000000 00000000 00000000

-0 10000000 11111111 00000000

（2）-127 11111111 10000000 10000001

（3）127 01111111 01111111 01111111

（4）-57 10101111 11010000 11010001

（5）126 01111110 01111110 01111110

（6）-126 11111110 10000001 10000010

（7）-128 10000000

（8）68 01000100 01000100 01000100

〔习题 1.9〕完成下列二进制数的运算

（1）1011＋1001　（2）1011－1001　（3）1011×1001　（4）10111000÷1001

（5）1011 ∧1001　（6）1011 ∨1001　（7）~1011　（8）1011？1001

〔解答〕

（1）1011＋1001=10100

（2）1011－1001=0010

（3）1011×1001=1100011

（4）10111000÷1001=10100，余数 1000

（5）1011 ∧ 1001=1001

（6）1011 ∨1001=1011

（7）~1011=0100

(8) 1011?1001=0010(?代表异或)

〔习题 1.10〕数码 0～9、大写字母 A～Z、小写字母 a～z 对应的 ASCII 码分别是多少？ASCII 码为 0dh、0ah 对应的是什么字符？

〔解答〕

数码 0～9：30H～39H

大写字母 A～Z：41H～5AH

小写字母 a～z：61H～7AH

ASCII 码为 0dh、0ah 分别对应回车和换行控制字符。

〔习题 1.11〕计算机中有一个“01100001”编码，如果把它认为是无符号数，它是十进制什么数？如果认为它是 BCD 码，则表示什么数？又如果它是某个 ASCII 码，则代表哪个字符？

〔解答〕

十进制无符号数：01100001B＝61H＝97

BCD 码：61

ASCII 码：a

〔习题 1.12〕简述 Intel 80x86 系列微处理器在指令集方面的发展。

〔解答〕

1978 年 Intel，正式推出了 16 位 8086CPU，1979 年 Intel 推出了准 16 位微处理器 8088，随后，Intel 推出了 80186/80188，80186/80188 指令系统比 8086 指令系统新增了若干条实用的指令，涉及堆栈操作、移位指令、过程指令和边界检测及乘法指令，1982 年 Intel 推出 80286 CPU， 80286 指令系统包括全部 80186 指令及新增的保护指令 15 条，其中有些保护方式在实方式下也可以使用，1985 年，Intel80x86 推出微处理器地进入第三代 80386 CPU， 80386 指令系统在兼容原来 16 位指令系统的基础上，全面升级为 32 位，还新增了有关位操作、条件设置指令以及控制、调试和测试寄存器的传送指令等，1989 年，Intel 推出了 80486CPU,80486 将浮点处理单元 FPU 集成进来，还采用了精简指令集计算机技术 RISC 和指令流水线方式，还新增了用于多处理器和内部 Cache 操作的 6 条指令，1993 年 Intel 制成了俗称 586 的微处理器，取名 Pentium。Pentium 仍为 32 位结构，地址总线为 32 位，对常用的简单指令用硬件实现，重新设计指令的微代码等，Pentium 新增了一条 8 字节比较交换指令和一条处理器识别指令，以及 4 条系统专用指令，1996 年推出了 MMX Pentium,新增了 57 条多媒休指令，1995 年 Intel 推出 Pentium Pro 新增了 3

条指令，1999 年推出了 PentiumIII新增了 70 条 SSE 指令，2000 年推出的 Pentium4 新增了 76 条 SSE2 指令

〔习题 1.13〕什么是 DOS 和 ROM-BIOS?

〔解答〕

DOS 是 Diskette Operating system 的缩写，意思是磁盘操作系统,DOS 主要是面向磁盘的系统软件，说得简单些，就是人与机器的一座桥梁，是罩在机器硬件外面的一层“外壳”，是 1981~1995 年的个人电脑上使用的一种主要的操作系统。BIOS（Basic Input / Output System）即基本输入输出系统，通常是固化在只读存储器（ROM）中，所以又称为 ROM－BIOS。它直接对计算机系统中的输入、输出设备进行设备级、硬件级的控制，是连接软件程序和硬件设备之间的枢纽。ROM－BIOS 是计算机系统中用来提供最低级、最直接的硬件控制的程序。

〔习题 1.14〕简述 PC 机最低 1MB 主存空间的使用情况。

〔解答〕

（1）基本 RAM 区(00000H—9FFFFH)该区共 640KB，由 DOS 进行管理。在这个区域中操作系统要占用掉一部分低地址空间,其它则向用户程序开放。

（2）保留区 RAM（A0000H--BFFFFFH）该区为系统安排的“显示缓冲存储区”，共 126KB，是显卡上的芯片提供支持，用于存放屏幕显示信息。但这部分地址空间实际上并没有全部使用。

（3）扩展区 ROM（C0000H--DFFFFH）该区 128KB，由接口卡上的芯片提供支持，用于为系统不直接支持的外设安排设备驱动程序。用户固化的程序就可[安排在这一段，系统的会对它进行确认和连接。

（4）系统区 ROM（E0000H--FFFFFH）该区共 128KB，由系统占用，它主要提供 ROM--BIOS 程序，基本输入输出程序 BIOS，是操作系统的重要组成部分，主要用来驱动输入输出设备，也负责系统的上电检测，磁盘引导等初始化操作，在 ROM--BIOS 中还有 CMOS 微机设置程序以及使用的字符图符信息等内容。

〔习题 1.15〕罗列 8086CPU 的 8 个 8 位和 16 位通用寄存器，并说明各自的作用。

〔解答〕

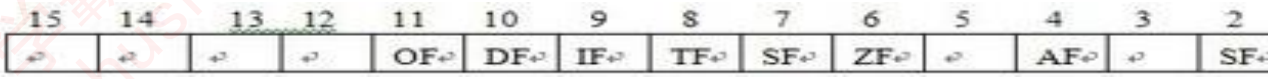
（1）数据寄存器：AX 称为累加器，使用频度最高，用于算术、逻辑运算以及与外设传送信息等；BX 称为基址寄存器，常用做存放存储器地址；CX 称为计数器，作为循环和串操作等指令中的隐含计数器；DX 称为数据寄存器，常用来存放双字长数据的高 16 位，或存放外设端口地址。

（2）指针及变址寄存器包括 SI,DI,BP,S P,四个寄存器，常用于存储器寻址时提供地址。SI 是源变址寄存器，DI 是目的变址寄存器，一般与 DS 联用确定数据段和附加段中某一存储单元地址，在串指令中，SI 与 DS 联用、DI 和 ES 联用，分别寻址数据段和附加段；同时，在串指令中，SI 和 DI 还都具有自动增量或减量的功能。S P,为堆栈指针寄存器，指示栈顶的偏移地址；BP 为基地址指针寄存器，表示堆栈段中的基地址。S P 与 BP 寄存器均可与 SS 段寄存器联合使用以确定堆栈段中的存储单元地址。

〔习题 1.16〕什么是标志，它有什么用途？状态标志和控制标志有什么区别？画出标志寄存器 FLAGS，说明各个标志的位置和含义。

〔解答〕

标志用于反映指令执行结果或控制指令执行形式。它是汇编语言程序设计中必须特别注意的一个方面，状态用来记录运行的结果的状态信息，许多指令的执行都将相应地设置它，控制标志位可由程序根据需要用指令设置，用来控制处理器执行指令的方式。



CF 是进位标志；ZF 是零标志；SF 是符号标志；PF 奇偶标志；OF 溢出标志；AF 辅助进位标志；DF 方向标志；IF 中断允许标志；TF 陷阱标志。

〔习题 1.17〕举例说明 CF 和 OF 标志的差异。

〔解答〕

溢出标志 OF 和进位标志 CF 是两个意义不同的标志。

进位标志表示无符号数运算结果是否超出范围，运算结果仍然正确；溢出标志表示有符号数运算结果是否超出范围，运算结果已经不正确。

例 1：3AH + 7CH＝B6H

无符号数运算：58＋124＝182，范围内，无进位

有符号数运算： 58＋124＝182 ，范围外，有溢出

例 2：AAH + 7CH＝（1）26H

无符号数运算：170＋124＝294，范围外，有进位

有符号数运算：－86＋124＝28 ，范围内，无溢出

〔习题 1.18〕字和双字在存储器中如何存放，什么是“小端方式”？对字和双字存储单元，什么是它们的对齐地址？为什么要对齐地址？

〔解答〕

字或双字在存储器中占相邻的 2 个或 4 个存储单元；存放时，低字节存入低地址，高字节存入高地址；字或双字单元的地址用它的低地址来表示。80x86 处理器采用的这种“低对低，高对高”的存储形式，被称为“小端方式”；将字单元安排在偶地址，双字节单元安排在模 4 地址，被称为“地址对齐方式”因为对于不对齐地址的数据，处理器访问时，需要额外的访问时间，所以通常应该将数据的地址对齐，以取得较高的存取速度。

〔习题 1.19〕什么是 8086 中的逻辑地址和物理地址？逻辑地址如何转换成物理地址？请将如下逻辑地址用物理地址表达：

（1）FFFFh:0 （2）40h:17h （3）2000h:4500h （4）B821h:4567h

〔解答〕

在 8086 处理器中，对应每个物理存储单元都有一个唯一的 20 位编号，就是物理地址，从 00000H～FFFFFH。

在 8086 内部和用户编程时，采用的段基地址 ：段内偏移地址形式称为逻辑地址。

将逻辑地址中的段地址左移二进制 4 位（对应 16 进制是一位，即乘以 16），加上偏移地址就得到 20 位物理地址

如下逻辑地址用物理地址表达：

（1）FFFFh:0＝FFFF0H

（2）40h:17h＝00417H

（3）2000h:4500h＝24500H

（4）B821h:4567h＝BC777H　　（不要算错）

〔习题 1.20〕8086 有哪 4 种逻辑段，各种逻辑段分别是什么用途？

〔解答〕

代码段（Code Segment）用来存放程序的指令序列。处理器利用 CS : IP 取得下一条要执行的指令。

堆栈段（Stack Segment）确定堆栈所在的主存区域。处理器利用 SS : SP 操作堆栈中的数据。

数据段（Data Segment）存放当前运行程序所用的数据。处理器利用 DS : EA 存取数据段中的数据。

附加段（Extra Segment）是附加的数据段，也用于数据的保存。处理器利用 ES : EA 存取数据段中的数据

〔习题 1.21〕数据的默认段是哪个，是否允许其他段存放数据？如果允许，如何实现，有什么要求？

〔解答〕

数据的默认段是安排在数据段，也经常安排在附加段，尤其是串操作的目的区必须是附加段，允许其它段存放数据，数据的存放比较灵活的，实际上可以存放在任何一种逻辑段中，这时，只要明确指明是哪个逻辑段就可以了。

〔习题 1.22〕什么是操作码、操作数和寻址方式？有哪三种给出操作数的方法？

〔解答〕

操作码说明计算机要执行哪种操作，它是指令中不可缺少的组成部分，操作数是指令执行的参与者，也是各种操作的对象，我们把寻找数的方式叫做操作数的寻址方式。给出操作数的三种方法是直接给出，间接给出，隐藏操作数方式给出。

〔习题 1.23〕什么是有效地址 EA？8086 的操作数如果在主存中，有哪些寻址方式可以存取它？

〔解答〕

DS 存放数据段的段地址，存储器中操作数的偏移地址则由各种主存方式得到，称之为有效地址 EA。8086 的操作数如果在主存中，可以存取它的寻址方式有直接寻址方式、寄存器间接寻址方式、寄存器相对寻址方式、基址变址寻址方式、相对基址变址寻址方式。

〔习题 1.24〕说明下列指令中源操作数的寻址方式？如果 BX＝2000H，DI＝40H，给出 DX 的值或有效地址 EA 的值。

（1） mov dx,[1234h]

（2） mov dx,1234h

（3） mov dx,bx

（4） mov dx,[bx]

（5） mov dx,[bx+1234h]

（6） mov dx,[bx+di]

（7） mov dx,[bx+di+1234h]

〔解答〕

（1）直接寻址，EA＝1234H

（2）立即数寻址，DX＝1234H

（3）寄存器寻址，DX＝2000H

（4）间接寻址，EA＝2000H

（5）相对寻址，EA＝3234H

（6）基址变址寻址，EA＝2040H

（7）相对基址变址寻址，EA＝3274H

第 2 章 8086 的指令系统

〔习题 2.1〕已知 DS＝2000H、BX＝0100H、SI＝0002H，存储单元[20100H]～[20103H]依次存放 12 34 56 78H，[21200H]～[21203H]依次存放 2A 4C B7 65H，说明下列每条指令执行完后 AX 寄存器的内容。

（1） mov ax,1200h

（2） mov ax,bx

（3） mov ax,[1200h]

（4） mov ax,[bx]

（5） mov ax,[bx+1100h]

（6） mov ax,[bx+si]

（7） mov ax,[bx][si+1100h]

〔解答〕

（1）AX＝1200H

（2）AX＝0100H

（3）AX＝4C2AH　;偏移地址=bx=0100h

（4）AX＝3412H　;偏移地址=bx=0100h

（5）AX＝4C2AH　;偏移地址=bx+1100h=1200h

（6）AX＝7856H　;偏移地址=bx+si=0100h+0002h=0102h

（7）AX＝65B7H　;偏移地址=bx+si+1100h=0100h+0002h+1100h=1202h

〔习题 2.2〕指出下列指令的错误

（1） mov cx,dl

（2） mov ip,ax

（3） mov es,1234h

（4） mov es,ds

（5） mov al,300

（6） mov [sp],ax

（7） mov ax,bx+di

（8） mov 20h,ah

〔解答〕

（1）两操作数类型不匹配

（2）IP 指令指针禁止用户访问

（3）立即数不允许传给段寄存器

（4）段寄存器之间不允许传送

（5）两操作数类型不匹配

（6）目的操作数应为[SI]

（7）源操作数应为 [BX+DI]

（8）立即数不能作目的操作数

〔习题 2.3〕已知数字 0～9 对应的格雷码依次为：18H、34H、05H、06H、09H、0AH、0CH、11H、12H、14H，它存在于以 table 为首地址（设为 200H）的连续区域中。请为如下程序段的每条指令加上注释，说明每条指令的功能和执行结果。

```
lea bx,table
mov al,8
xlat
```

〔解答〕

```
lea bx,table      ；获取 table 的首地址，BX＝200H
mov al,8          ；传送欲转换的数字，AL＝8
xlat              ；转换为格雷码，AL＝12H   P35
```

〔习题 2.4〕什么是堆栈，它的工作原则是什么，它的基本操作有哪两个，对应哪两种指令？

〔解答〕

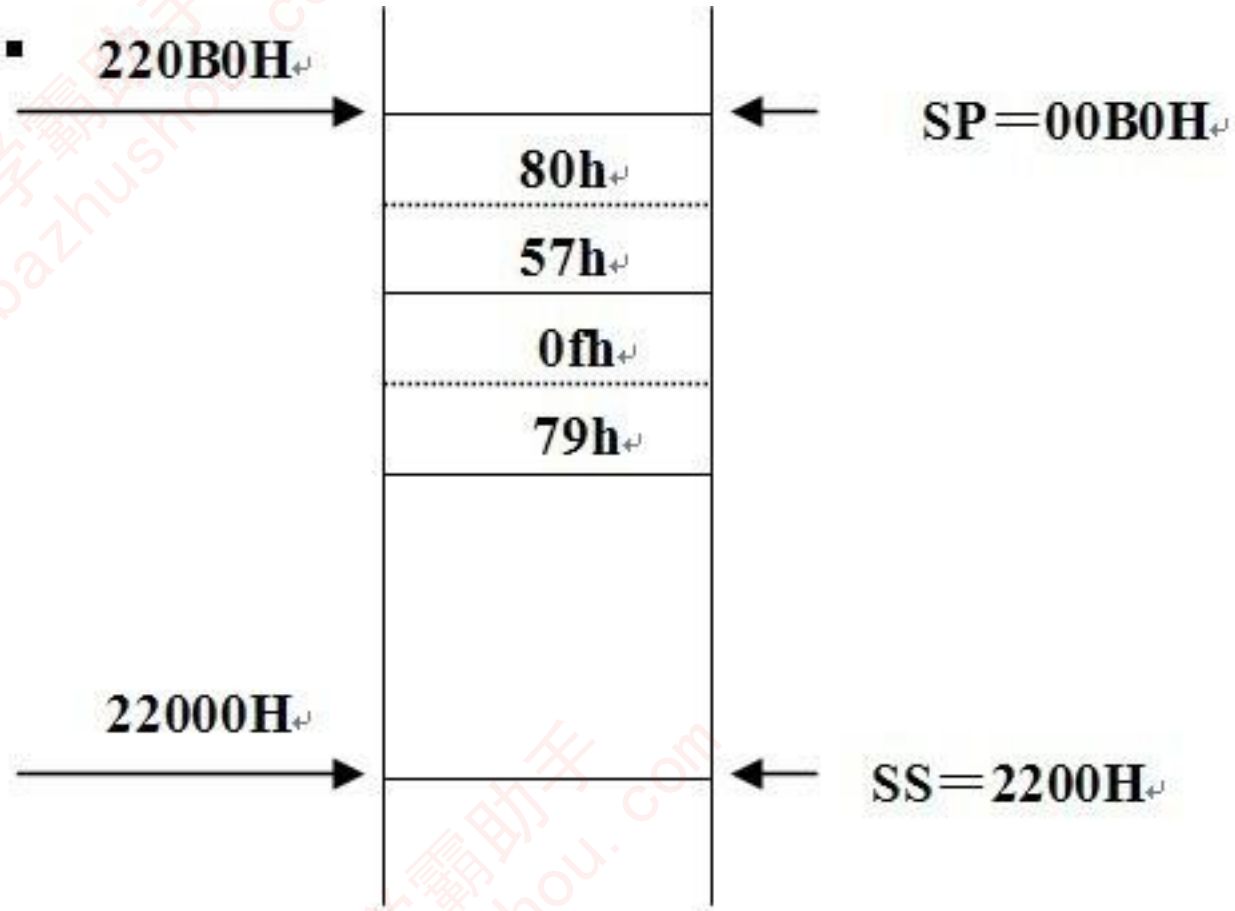
堆栈是一种按“先进后出”原则存取数据的存储区域，位于堆栈段中，使用 SS 段寄存器记录其段地址；它的工作原则是先进后出；堆栈的两种基本操作是压栈和出栈，对应的指令是 PUSH 和 POP。

〔习题 2.5〕已知 SS = 2200H、SP = 00B0H，画图说明执行下面指令序列时，堆栈区和 SP 的内容如何变化？

```
mov ax,8057h
push ax
mov ax,0f79h
push ax
pop bx
pop [bx]
```

〔解答〕

```
mov ax,8057h
push ax
mov ax,0f79h
push ax
pop bx          ;bx=0f79h
pop [bx]        ;DS:[0f79h]=8057h
```



〔习题 2.6〕给出下列各条指令执行后 AL 值，以及 CF、ZF、SF、OF 和 PF 的状态：

```
mov al,89h
add al,al
add al,9dh
cmp al,0bch
sub al,al
dec al
inc al
```

〔解答〕

```
mov al,89h          ; AL=89h    CF   ZF   SF   OF   PF
add al,al           ; AL=12h    1    0    0    1    1
; 1000 1001
+1000 1001
```

10001 0010

add al,9dh ; AL=0afh 0 0 1 0 1

; 0001 0010

+ 1001 1101

1010 1111

cmp al,0bch ; AL=0afh 1 0 1 0 1

; 1010 1111

-1011 1100

* 0100 0011

sub al,al ; AL=00h 0 1 0 0 1

dec al ; AL=0ffh 0 0 1 0 1

; 0000 0000

- 0000 0001

*1111 1111

inc al ; AL=00h 0 1 0 0 1

;1111 1111

+0000 0001

*1111 1111

〔习题 2.7〕设 X、Y、Z 均为双字数据，分别存放在地址为 X、X+2；Y、Y+2；Z、Z+2 的存储单元中，它们的运算结果存入 W 单元。阅读如下程序段，给出运算公式。

```
mov ax,X
mov dx,X+2

add ax,Y
```

```
adc dx,Y+2

add ax,24
adc dx,0

sub ax,Z
sbb dx,Z+2

mov W,ax
mov W+2,dx
```

〔解答〕

W＝X+Y+24-Z

〔习题 2.8〕请分别用一条汇编语言指令完成如下功能：

（1）把 BX 寄存器和 DX 寄存器的内容相加，结果存入 DX 寄存器。

（2）用寄存器 BX 和 SI 的基址变址寻址方式把存储器的一个字节与 AL 寄存器的内容相加，并把结果送到 AL 中。

（3）用 BX 和位移量 0B2H 的寄存器相对寻址方式把存储器中的一个字和 CX 寄存器的内容相加，并把结果送回存储器中。

（4）用位移量为 0520H 的直接寻址方式把存储器中的一个字与数 3412H 相加，并把结果送回该存储单元中。

（5）把数 0A0H 与 AL 寄存器的内容相加，并把结果送回 AL 中。

〔解答〕

（1）ADD DX,BX

（2）ADD AL,[BX+SI]

（3）ADD [BX+0B2H],CX

（4）ADD WORD PTR [0520H],3412H

（5）ADD AL,0A0H

〔习题 2.9〕设 X、Y、Z、V 均为 16 位带符号数，分别装在 X、Y、Z、V 存储单元中，阅读如下程序段，得出它的运算公式，并说明运算结果存于何处。

```
mov ax,X      ;ax=X
imul Y        ;DX.AX=X*Y
```

```
mov cx,ax    ;cx=X*Y 的低 16 位
mox bx,dx    ;bx=X*Y 的高 16 位

mov ax,Z     ;ax=Z
cwd
add cx,ax    ;cx=Z 的低 16 位+X*Y 的低 16 位
adc bx,dx    ;bx=Z 的高 16 位+X*Y 的高 16 位+低位进位

sub cx,540   ;cx=Z 的低 16 位+X*Y 的低 16 位-540
sbb bx,0     ;bx=Z 的高 16 位+X*Y 的高 16 位+低位进位-低位借位

mov ax,V     ;ax=V
cwd
sub ax,cx    ;ax=V 的低 16 位-（Z 的低 16 位+X*Y 的低 16 位-540）
sbb dx,bx    ;dx=V 的高 16 位-（Z 的高 16 位+X*Y 的高 16 位+低位进位-低位借位）-低位借位

idiv X ;/X
```

〔解答〕

[V-(X*Y+Z-540)]/X

AX 存商，DX 存余数

〔习题 2.10〕指出下列指令的错误：

（1） xchg [si],30h

（2） pop cs

（3） sub [si],[di]

（4） push ah

（5） adc ax,ds

（6） add [si],80h

（7） in al,3fch

（8） out dx,ah

〔解答〕

（1）xchg 的操作数不能是立即数

（2）不应对 CS 直接赋值

（3）两个操作数不能都是存储单元

（4）堆栈的操作数不能是字节量

（5）adc 的操作数不能是段寄存器

（6）没有确定是字节还是字操作

（7）in 不支持超过 FFH 的直接寻址

（8）out 只能以 AL/AX 为源操作数

〔习题 2.11〕给出下列各条指令执行后的结果，以及状态标志 CF、OF、SF、ZF、PF 的状态。

```
mov ax,1470h
and ax,ax
or ax,ax
xor ax,ax
not ax
test ax,0f0f0h
```

〔解答〕

```
mov ax,1470h        ；AX＝1470H CF   ZF   SF   OF   PF

and ax,ax           ；AX＝1470H 0    0    0    0    0
 ;0001 0100 0111 0000

or ax,ax            ；AX＝1470H 0    0    0    0    0
xor ax,ax           ；AX＝0000H 0    1    0    0    1
not ax              ；AX＝FFFFH 0    1    0    0    1
test ax,0f0f0h      ；AX＝FFFFH 0    0    1    0    1
```

注意：MOV 和 NOT 指令不影响标志位；其他逻辑指令使 CF＝OF＝0，根据结果影响其他标志位。

〔习题 2.12〕假设例题 2.32 的程序段中，AX = 08H，BX = 10H，请说明每条指令执行后的结果和各个标志位的状态。

〔解答〕

指令	；执行结果	CF	OF	SF	ZF	PF
mov si,ax	; SI=AX=0008H	-	-	-	-	-
shl si,1	; SI=2*AX=0010H	0	0	0	0	0
add si,ax	; SI=3*AX=0018H	0	0	0	0	1
mov dx,bx	; DX=BX=0010H	0	0	0	0	1
mov cl,03h	; CL=03H	0	0	0	0	1
shl dx,cl	; DX=8*BX=0080H	0	u	0	0	0
sub dx,bx	; DX=7*BX=0070H	0	0	0	0	0
add dx,si	; DX=7*BX+3*AX=0088H	0	0	0	0	1

注意：逻辑左移 N 次相当于无符号整数乘以 2 的 N 次方，逻辑右移 N 次相当于无符号整数除以 2 的 N 次方。移位指令根据移位的数据设置 CF，根据移位后的结果影响 SF，ZF，PF。在进行一位移位时，根据最高符号位是否改变设置 OF，如改变则 OF＝1。另外，程序注释用“u”表示标志无定义（不确定），“-”表示无影响。

〔习题 2.13〕编写程序段完成如下要求：

（1）用位操作指令实现 AL（无符号数）乘以 10

（2）用逻辑运算指令实现数字 0 ~ 9 的 ASCII 码与非压缩 BCD 码的互相转换

（3）把 DX.AX 中的双字右移 4 位

〔解答〕

（1）;不考虑进位

```
mov bl,al
mov cl,3
shl al,cl          ;*8
add al,bl          ;shl bl,1
add al,bl
；考虑进位
xor ah,ah
mov bx,ax
mov cl,3
```

```
shl ax,cl
add ax,bx          ;shl bx,1
add ax,bx
```

（2）数字 0～9 的 ASCII 码是：30h～39h

非压缩 BCD 码的 0～9 是：00h～09h

方法一：

```
and al,0fh         ;实现 ASCII 到非压缩 BCD 码的转换
or al,30h          ;实现非压缩 BCD 码到 ASCII 的转换
```

方法二：

```
xor al,30h         ；求反 D5D4 位，其他不变
；即高 4 位为 3，则变为 0；高 4 位为 0，则变为 3
```

（3）

```
    mov cl,4
again:  shr dx,1           ;实现逻辑右移
                           ；采用“sar dx,1”，则实现算术右移
        rcr ax,1
        dec cl
        jnz again
```

〔习题 2.14〕已知 AL = F7H（表示有符号数-9），分别编写用 SAR 和 IDIV 指令实现的除以 2 的程序段，并说明各自执行后，所得的商是什么？

〔解答〕

（1）用 sar 编写

```
mov al,0f7h        ; -9 送 AL 1111 1001
sar al,1           ; 结果：AL=1111 1100B=0FBH  即-5
```

（2）用 idiv 编写

```
mov al,0f7h        ; -9 送 al
cbw                ; 字节符号扩展位字
mov bl,2           ; 注意除数不可为立即数
idiv bl            ; 结果：商为 al=fch  (-4)
                   ; 余数：ah=ffh (-1)
```

结论：符号数的除法用 idiv 准确。

〔习题 2.15〕已知数据段 500h ~600h 处存放了一个字符串，说明下列程序段执行后的结果：

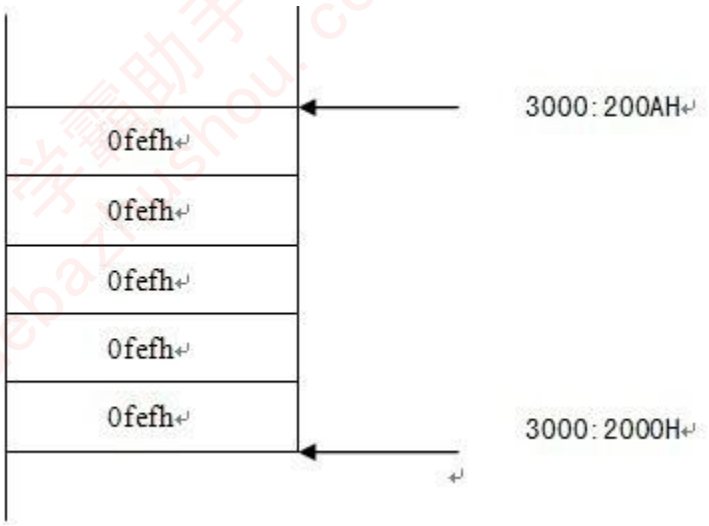
```
mov si,600h
mov di,601h
mov ax,ds
mov es,ax
mov cx,256
std
rep movsb
```

〔解答〕

〔习题 2.16〕说明下列程序段的功能

```
cld
mov ax,0fefh
mov cx,5
mov bx,3000h
mov es,bx
mov di,2000h
rep stosw
```

〔解答〕



将 es:di (即 3000：2000h 或 32000h)开始的 5 个单元的内容置为 0fefh。

〔习题 2.17〕指令指针 IP 是通用寄存器还是专用寄存器？有指令能够直接它赋值吗？哪类指令的执行会改变它的值？

〔解答〕

指令指针 IP 不是通用寄存器，不可直接赋值，属于专用寄存器。有且仅有循环、转移、子程序调用和返回、中断类等指令可以改变它的值。

〔习题 2.18〕控制转移类指令中有哪三种寻址方式？

〔解答〕

控制转移类指令的寻址方式：相对寻址、直接寻址方式和间接寻址方式（又可以分成寄存器和存储器间接寻址）。

〔习题 2.19〕什么是短转移 short jump、近转移 near jump 和远转移 far jump？什么是段内转移和段间转移？8086 有哪些指令可以实现段间转移？

〔解答〕

短转移：指段内－128～127 之间的转移，位移量用一个字节表示

近转移：指段内±32K 之间的转移，位移量用一个字表示

远转移：指段间 1MB 范围的转移

段内转移：指在同一个代码段内的转移，可以是短转移或者近转移

段间转移：指转移到另外一个代码段，就是远转移

8086/8088CPU 的 JMP、CALL 和 INT n 指令可以实现段间转移

〔习题 2.20〕8086 的条件转移指令的转移范围有多大？实际编程时，你如何处理超出范围的条件转移？

〔解答〕

8086 的条件转移的转移范围：在当前指令地址的 +127～-128 之内。

如条件转移的转移范围超出此范围，可在此范围内安排一条无条件转移，再转移到范围外的目标地址。

〔习题 2.21〕假设 DS=2000H，BX=1256H，SI=528FH，位移量 TABLE=20A1H，[232F7H]=3280H，[264E5H]=2450H，试问执行下列段内间接寻址的转移指令后，转移的有效地址是什么？

（1） JMP BX

（2） JMP TABLE[BX]

（3） JMP [BX][SI]

〔解答〕

（1）转移的有效地址 EA= BX=1256H

（2）转移的有效地址 EA= [DS:20A1H+1256H]=[232F7]=3280H

（3）转移的有效地址 EA= [DS:1256H+528FH]=264E5H=2450H

〔习题 2.22〕判断下列程序段跳转的条件

（1） xor ax,1e1eh

je equal

（2）test al,10000001b

jnz there

（3）cmp cx,64h

jb there

〔解答〕

（1）AX＝1e1eh（异或后为 0）

（2）AL 的 D0 或 D7 至少有一位为 1

（3）CX（无符号数）＜ 64h

〔习题 2.23〕设置 CX = 0，则 LOOP 指令将循环多少次？例如：

mov cx,0

delay: loop delay

〔解答〕

216 次。

〔习题 2.24〕假设 AX 和 SI 存放的是有符号数，DX 和 DI 存放的是无符号数，请用比较指令和条件转移指令实现以下判断：

（1）若 DX > DI，转到 above 执行；

（2）若 AX > SI，转到 greater 执行；

（3）若 CX = 0，转到 zero 执行；

（4）若 AX－SI 产生溢出，转到 overflow 执行；

（5）若 SI≤AX，转到 less_eq 执行；

（6）若 DI≤DX，转到 below_eq 执行。

〔解答〕

（1）若 DX > DI，转到 above 执行

```
cmp dx,di
ja above          ；＝jnbe above
```

（2）若 AX > SI，转到 greater 执行

```
cmp ax,si
jg greater        ；＝jnle greater
```

（3）若 CX = 0，转到 zero 执行

```
cmp cx,0
jz zero           ;＝ jcxz zero
```

（4）若 AX－SI 产生溢出，转到 overflow 执行；

```
cmp ax,si
jo overflow
```

（5）若 SI≤AX，转到 less_eq 执行；

```
cmp si,ax         ; cmp ax,si
jle less_eq       ; jge less_eq
```

（6）若 DI≤DX，转到 below_eq 执行。

```
cmp di,dx         ; cmp dx,di
jbe below_eq      ; jae below_eq
```

〔习题 2.25〕有一个首地址为 array 的 20 个字的数组，说明下列程序段的功能。

```
mov cx,20
mov ax,0
```

```
        mov si,ax
sum_loop:  add ax,array[si]
        add si,2
        loop sum_loop
        mov total,ax
```

〔解答〕

将首地址为 array 得 20 个字的数组求和，并将结果存入 total 单元中。

〔习题 2.26〕按照下列要求，编写相应的程序段：

（1）起始地址为 string 的主存单元中存放有一个字符串（长度大于 6），把该字符串中的第 1 个和第 6 个字符（字节量）传送给 DX 寄存器。

（2）从主存 buffer 开始的 4 个字节中保存了 4 个非压缩 BCD 码，现按低（高）地址对低（高）位的原则，将它们合并到 DX 中。

（3）编写一个程序段，在 DX 高 4 位全为 0 时，使 AX = 0；否则使 AX = -1。

（4）有两个 64 位数值，按“小端方式”存放在两个缓冲区 buffer1 和 buffer2 中，编写程序段完成 buffer1－buffer2 功能。

（5）假设从 B800h : 0 开始存放有 100 个 16 位无符号数，编程求它们的和，并把 32 位的和保存在 DX.AX 中。

（6）已知字符串 string 包含有 32KB 内容，将其中的’$’符号替换成空格。

（7）有一个 100 个字节元素的数组，其首地址为 array，将每个元素减 1（不考虑溢出）存于原处。

（8）统计以 ’$’ 结尾的字符串 srting 的字符个数。

〔解答〕

（1）解答：

```
        mov si,0
        mov dl,string[si]   ；第 1 个字符送 dl 寄存器：mov dl,stirng[0]
        mov si,5
        mov dh,string[si]   ；第 6 个字符送 dh 寄存器：mov dl,stirng[5]
```

（2）解答：

```
        xor si,si           ；si 清零
        mov al,buffer[si]   ；第一字节
        inc si
        mov ah,buffer[si]   ；第二字节
```

```
        mov cl,4
        shl ah,cl          ；BCD 码移到高半字节
        or al,ah           ；组合成压缩 BCD 码
        mov dl,al              ；存入 dl 寄..
        inc si
        mov al,buffer[si]  ；第三字节
        inc si
        mov ah,buffer[si]  ；第四字节
        mov cl,4
        shl ah,cl          ；BCD 码移到高半字节
        or al,ah           ；组合成压缩 BCD 码
        mov dh,al          ；存入 dh 寄..
```

（3）解答：

```
        test dx,0f000h      ; test dh,0f0h
        jz next            ; jnz next
        mov ax,-1          ; mov ax,0
        jmp again
next:  mov ax,0            ; mov ax,0ffffh
again:    ...
```

（4）解答：

```
        mov ax, word ptr buffer1
        sub ax, word ptr buffer2    ；先减低 16 位
        mov dx, word ptr buffer1+2
        sbb dx, word ptr buffer2+2  ；后减高 16 位，需减低 16 位的借位
```

（5）解答：

```
        mov ax,0b800h
         mov ds,ax      ；段地址
         xor si,si      ；地址偏移量 si=0
         xor dx,dx      ；和的高字 dx=0
         mov cx,99      ；加的次数
```

```
        mov ax,[si]     ；第一个数
again:  inc si          ；指向下一个字单元
        inc si
        add ax,[si]     ；加下一个数
        jnc  noc        ；无进位转
        inc dx          ；有进位 dx=dx+1
   noc: dec cx          ；次数-1
        jnz cx,again    ；非 0 继续加
```

（6）解答 1：不使用串操作指令（更好）

```
        mov si,offset string
        mov cx,8000h            ; 32k=2^15=8000h

again:    cmp byte ptr [si], '$'   ;'$' =24h
        jnz next                ；不要采用 jz 进行分支

        mov byte ptr [si], ' '  ;' '=20h
next:     inc si
        loop again              ; dec cx
                                ; jnz again
```

（6）解答 2：使用串操作指令

```
        mov di,offset string
        mov al,'$'
        mov cx,8000h
        cld
again:    scasb
        jnz next
        mov byte ptr es : [di-1], ' '

next:     loop again
```

（7）解答 1：

```
        mov si,offset array
        mov cx,100
again:  dec byte ptr [si]
        inc si
        loop again
```

（7）解答 2：

```
        xor si,si               ; si<--0
        mov cx,100              ; 循环次数
again:  dec array[si]
        inc si
        loop again
```

（7）解答 3：

```
        mov si,offset array
        mov di,si
        mov ax,ds
        mov es,ax
        mov cx,100
        cld
again:  lodsb
        dec al
        stosb
        loop again
```

（8）解答：

```
        xor si,si               ;si<--0
coun:   cmp string[si],'$'
        je  done
        inc si
        jmp coun
done:   ...
```

〔习题 2.27〕对下面要求，分别给出 3 种方法，每种方法只用一条指令。

（1）使 CF=0　（2）使 AX=0　（3）同时使 AX=0 和 CF=0

〔解答〕

（1）解答：

```
clc
and ax,ax
or ax,ax
```

（2）解答：

```
xor ax,ax
and ax,0
mov ax,0
```

（3）解答：

```
and ax,0
xor ax,ax
sub ax,ax
```

〔习题 2.28〕参照本习题的示意图，分析调用序列，画出每次调用及返回时的堆栈状态。其中 CALL 前是该指令所在的逻辑地址；另外，段内直接调用指令的机器代码的字节数为 3，段间直接调用指令则为 5 个字节。

〔解答〕

主程序转子 suba 时段内调用：断点 1 为 2000h：0400h+3，

转子是只将 IP 压栈。

suba 转子 subb 时段间调用：断点 2 为 2000h：0840h+5，转子时须将 cs 段地址和 IP 压栈

suba 转子 subc 时段内调用：断点 3 为 2000h：0c021h+3，转子是只将 IP 压栈。

注：压栈时先修改 sp 再压入断点，弹栈时先弹出断点再修改 sp。

〔习题 2.29〕已知 AX 、BX 存放的是 4 位压缩 BCD 表示的十进制数，请说明如下子程序的功能和出口参数。

```
add al,bl
daa
xchg al,ah
adc al,bh
```

```
daa
xchg al,ah
ret
```

〔解答〕

压缩 BCD 码加法：AX←AX＋BX

出口参数：AX＝BCD 码和

〔习题 2.30〕AAD 指令是用于除法指令之前，进行非压缩 BCD 码调整的。实际上，处理器的调整过程是：AL←AH×10＋AL，AH←0。如果指令系统没有 AAD 指令，请用一个子程序完成这个调整工作。

〔解答〕

```
shl ah,1        ;ah=2*a   (设原 ah=a)
mov dl,ah       ;dl=2*a
mov cl,2        ;设定移位次数
shl ah,cl       ;ah=8*a
add ah,dl       ;ah=10*a
add al,ah       ;al=10*a+al
xor ah,ah       ;清零 ah
int 3           ;返回 DOS
```

注意：入口：AX 中存放有“和”（两非压缩 BCD 码）

出口：AL 中 已为调整后的二进制数

〔习题 2.31〕解释如下有关中断的概念：

（1） 内部中断和外部中断

（2） 单步中断和断点中断

（3） 除法错中断和溢出中断

（4） 中断向量号和中断向量表

〔解答〕

（1）内部中断是由于 8086CPU 内部执行程序引起的程序中断；外部中断是来自 8086CPU 之外的原因引起的程序中断；

（2）单步中断是若单步标志 TF 为 1，则在每条指令执行结束后产生的中断；断点中断是供调试程序使用的，它的中断类型号为 3 通常调试程序时，把程序按程序的任务分成几段，然后，每段设一个段点；

（3）除法错中断是在执行除法指令时，若除数为 0 或商超过了寄存器所能表达的范围产生的中断；溢出中断是在执行溢出中断指令 INTO 时，若溢出标志 OF 为 1 时产生的中断；

（4）中断向量号是 中断类型号；中断向量表是中断向量号与它所对应的中断服务程序起始地址的转换表。

〔习题 2.32〕试比较 INT n 和段间 CALL 指令、IRET 和段间 RET 指令的功能。

〔解答〕

INT n 响应中断时，除象 CALL 保护断点外，还应保护 FR；段间 CALL 指令用在主程序中实现子程序的调用。IRET 返回时，除象 RET 恢复断点外，还应恢复 FR；子程序执行完成后，应返回主程序中继续执行，这一功能由 RET 指令完成。

〔习题 2.33〕什么是系统功能调用？汇编语言中，它的一般格式是怎样的？

〔解答〕

系统功能调用是用户在程序一级请示操作系统服务的一种手段，它不是一条简单的硬指令，而是带有一定功能号的“访指令”，它的功能并非由硬件直接提供，而是由操作系统操作系统中的一段程序完成的，即由软件方法实现的

汇编语言中，它的一般格式是分如下四步进行：

（1）在 AH 寄存器置系统功能调用号；

（2）在指定的寄存器中设置 入口参数；

（3）用 INT21H(或 ROM—BIOS 的中断向量号)指令执行功能调用；

（4）据出口参数分析功能调用运行情况。

〔习题 2.34〕补充例 2.40，当有溢出时显示“Error! Overflow!”，无溢出时显示“OK”。

〔解答〕

```
okmsg      db ‘OK’, ‘$’
errmsg     db ‘Error ! Overflow !’, ‘$’
             …
             mov ax,X
            sub ax,Y
            jo overflow
            mov dx,offset okmsg
            jmp next
overflow:  mov dx,errmsg
   next:     mov ah,9
             int 21h
```

错误解答：

```
        mov ax,X
        sub ax,Y
        jo overflow
        mov dx,offset okmsg
okmsg  db 'OK', '$'
        mov dx,errmsg                    ;错误 1：数据定义在代码中
        mov ah,9
        int 21h
overflow:   errmsg db 'Error ! Overflow !', '$'
        mov dx,errmsg                    ; 错误 2：缺少 JMP 指令
        mov ah,9
        int 21h
```

〔习题 2.35〕补充例 2.42，显示“1”的个数；注意首先将个数转换为 ASCII 码。

〔解答〕

```
        and al,7fh      ；使 d7=0
        mov dl,al       ；转存于 dl
        jnp next        ；奇数个 1 转
        or al,80h       ；偶数个 1  使 d7=1
next:   xor bl,bl       ；纪录 1 的个数  dl=0
        mov cx,8        ；移位次数
again:  shl dl,1        ；逻辑左移 1 次
        jnc desp        ；如 cf=0  转
        inc bl          ；否则 cf=1  个数加 1
desp:   loop again      ；cx=cx-1  如 cx 不等于 0，循环
        or bl,30h       ；个数变为 ASCII 码
        mov ah,02h      ；在 CRT 上显示个数
        mov dl,bl
        int 21h
        ret
```

〔习题 2.36〕先提示输入数字“Input Number：0 ~ 9”，然后在下一行显示输入的数字，结束；如果不是键入了 0 ~ 9 数字，就提示错误“Error!”，继续等待输入数字。

〔解答〕

```
        ; 数据段
str1    db 'Input Number:0~9 : ',0dh,0ah,'$'
str2    db 'Error!',0dh,0ah,'$'
        ; 代码段
        mov ah,09h          ; 显示 str1 字符串
        mov dx,offset str1
        int 21h
getkey: mov ah,1            ; 调用 DOS 功能
        int 21h
        cmp  al,'0'
        jb error            ; 小于 0，出错处理
        cmp al,'9'
        ja error            ; 大于 9，出错处理
        mov ah,02h          ; 调用 DOS 显示字符功能，显示该数字
        mov dl,al
        int 21h
        ...                 ; 终止程序执行，返回 DOS
error:  mov ah,09h          ; 出错，调用 DOS  功能显示 str2 字符串
        mov dx,offset str2
        int 21h
        jmp getkey          ; 返回按键
```

〔习题 2.37〕从键盘输入一个字符串（不超过 255 个），将其中的小写字母转换成大写字母，然后按原来的顺序在屏幕上显示。

〔解答〕

思路：参考 P67 例 2.52 用 0ah 号 DOS 系统功能调用，从键盘输入一个字符串，然后从键盘缓冲区逐个取字符，在“a”与“z”之间的字符为小写字母，需要转换为大写字母（减去 20h），其他不变。

```
        ;xt237.asm
```

```
        .model small
        .stack
        .data
array    db 255
        db 0
array1    db 255 dup('$')
array2    db 0dh,0ah,'$'
        .code
        .startup
        mov ah,0ah              ; 键盘输入字符串
        mov dx,offset array
        int 21h
        mov dx,offset array2    ; 回车换行
        mov ah,09h
        int 21h
        mov bx,offset array1
again:    mov al,[bx]
        cmp al,'$'
        jz done
        cmp al,'a'              ; 小于 a 和大于 z 的字符不是小写字母
        jb next
        cmp al,'z'
        ja next
        sub al,20h              ; 在 a 和 z 之间的字符才是小写字母，转换为大写
        mov [bx],al             ; 保存到原位置
next:    inc bx
        jmp again
done:    mov dx,offset array1
        mov ah,09h
        int 21h
```

```
    .exit 0
    end
```

〔习题 2.38〕指令对状态标志的作用可以分成多种情况，例如无影响、无定义、按结果影响、特别说明的影响等，你能区别这些情况吗？分别用具体的指令来说明。

〔解答〕

指令对状态标志的影响

指令	OF	SF	ZF	AF	PF	CF
ADD/ADC/SUB/SBB/CMP/NEC/CMPS/SCAS	x	x	x	x	x	x
INC/DEC	x	x	x	x	x	-
MUL/IMUL	#	u	u	u	u	#
DIV/IDIV	u	u	u	u	u	u
AND/OR/XOR/TEST	0	x	x	u	x	0
SAL/SAR/SHL/SHR	#	x	x	u	x	#
ROL/ROR/RCL/RCR	#	-	-	-	-	#
SHLD/SHRD	#	x	x	u	x	#
XADD/CMPXCHG	x	x	x	x	x	x

状态符号说明

符号	说明
-	标志位不受影响（没有改变）
0	标志位复位（置 0）
1	标志位置位（置 1）
x	标志位按定义功能改变
#	标志位指令的特定说明改变
u	标志位不确定（可能为 0，也可能为 1）

〔习题 2.39〕8086 指令系统分成哪 6 个功能组？各组主要包含什么指令，举例说明。

〔解答〕

8086 指令系统分成的 6 个功能组是

（1）数据传送类令：

①通用数据传送指令如传送指令 MOV、交换指令 XCHG、换码指令 XLAT

②堆栈操作指令如进栈指令 PUSH 出栈指令 POP

③标志传送指令如标志寄存器传送;标志位操作

④地址传送指令如有效地址传送指令 LEA、指针传送指令 LDS 指针传送指令 LES

⑤输入输出指令如输入指令 IN、输出指令 OUT

（2）算术运算指令

①加法指令如加法指令 ADD、带进位加法指令 ADC、增量指令 INC

②减法指令如减法指令 SUB、带借位指令 SBB、减量指令 DEC、求补指令 NEG、比较指令 CMP

③乘法指令如无符号乘法指令 MUL、有符号乘法指令 IMUL

④除法指令如无符号数除法指令 DIV、有符号数除法指令 IDIV

⑤符号扩展指令如字节转换字指令 CBW、字转换字节指令 CWD

⑥十进制调整指令如压缩 BCD 码调整指令、非压缩 BCD 码调整指令

（3）位操作类指令

①逻辑运算指令如逻辑指令 AND、逻辑或指令 OR、逻辑异或指令 XOR、逻辑非指令 NOT、测试指令 TEST

②移位指令如逻辑移位指令 SHL 和 SHR、算术移位指令 SAL 和 SAR

③循环移位指令如不带移位指令 XOL 和 ROR、带移位指令 RCL 和 RCR

（4）控制转移类指令

①无条件转移指令 JMP

②条件转移指令如判断单个标志状态 JZ/JE、JNZ/JNE、JS/JNS、JP/JPE、JNP/JPO、JC/JB、JNC/JNB/JAE，用于比较无符号数高低 JB(JNAE)、JNB(JAE)、JBE(JNA)、JNBE(JA)，用于比较有符号数大小 JL(JNGE)、JNL(JGE)、JLE(JNG)、JNLE(JG)

③循环指令 LOOP

④子程序指令如子程序调用指令 CALL、子程序返回指令 RET

⑤中断指令如 INT N

（5）串操作类指令

①串传送指令 MOVS

②串存储指令 STOS

③串读取指令 LODS

④串比较指令 CMPS

⑤串扫描指令 SCAS

⑥重复前缀指令 REP;

（6）处理机控制类指令

①空操作指令 NOP

②段超越前缀指令 SEG

③封锁前缀指令 LOCK

④暂停指令 HLT

⑤交权指令 ESC

⑥等待指令 WAIT。

〔习题 2.40〕总结 8086 指令系统所采用的各种寻址方式，包括一般的数据寻址、外设数据寻址、堆栈数据寻址、串操作数据寻址、转移指令目的地址的寻址等，并举例说明。

〔解答〕

1. 一般的数据寻址

立即数寻址方式如：mov al,05h

寄存器寻址方式如：mov bx,ax

存储器寻址方式

（1）直接寻址方式如：mov ax,[2000h]

（2）寄存器间接寻址方式如：mov ax,es:[2000h]

（3）寄存器相对寻址方式如：mov ax,[di+06h]

（4）基址变址寻址方式如：mov ax,[bx+si]

（5）相对基址变址寻址方式如：mov ax,[bx+si+06h]

2. 外设数据寻址

输入指令 IN，如：

in al,21h

in ax,dx

输出指令 OUT，如：

out dx,al

3. 堆栈数据寻址

进栈指令 PUSH，如：PUSH [2000H]

出栈指令 POP，如：POP [2000H]

4. 串操作数据寻址

5. 转移指令目的地址的寻址

直接寻址、间接寻址、相对寻址

第 3 章 汇编语言程序格式

〔习题 3.1〕伪指令语句与硬指令语句的本质区别是什么？伪指令有什么主要作用？

〔解答〕

伪指令语句与硬指令语句的本质区别是能不能产生 CPU 动作；

伪指令的作用是完成对如存储模式、主存变量、子程序、宏及段定义等很多不产生 CPU 动作的说明，并在程序执行前由汇编程序完成处理。

〔习题 3.2〕什么是标识符，汇编程序中标识符怎样组成？

〔解答〕

为了某种需要，每种程序语言都规定了在程序里如何描述名字，程序语言的名字通常被称为标识符；

汇编语言中的标识符一般最多由 31 个字母、数字及规定的特殊符号（如-，$，？，@）组成，不能以数字开头。

〔习题 3.3〕 什么是保留字，汇编语言的保留字有哪些类型，并举例说明。

〔解答

保留字是在每种语言中规定了有特殊意义和功能的不允许再做其它用处的字符串；汇编语言的保留字主要有硬指令助记、伪指令助记符、运算符、寄存器名以及预定义符号等。汇编语言对大小写不敏感。如定义字节数和字符串的 DB 就是伪指令助记符。

〔习题 3.4〕汇编语句有哪两种，每个语句由哪 4 个部分组成？

〔解答〕

汇编语句有执行性语句和说明性语句；

执行性语句由标号、硬指令助记符、操作数和注释四部分组成；

说明性语句由名字、伪指令助记符、参数和注释四部分组成

〔习题 3.5〕汇编语言程序的开发有哪 4 个步骤，分别利用什么程序完成、产生什么输出文件。

〔解答〕

1. 编辑　　文本编辑程序　　汇编语言源程序.asm
2. 汇编　　汇编程序　　目标模块文件.obj
3. 连接　　连接程序　　可执行文件.exe 或.com
4. 调试　　调试程序　　应用程序

〔习题 3.6〕将第 2 章习题 2.36 采用简化段定义格式编写成一个完整的源程序。

〔解答〕

```
;简化段定义格式
.model small        ；定义程序的存储模式（小模式）
.stack              ；定义堆栈段（默认 1024 个字节）
```

```
        .data           ；定义数据段
str1    db 'Input Number:0~9 : ',0dh,0ah,'$'
str2    db 'Error!',0dh,0ah,'$'
        .cade           ；定义代码段
        .startup        ；说明程序的起始点，建立 ds,ss 的内容。
        mov ah,09h      ；显示 str1 字符串
        mov dx,offset str1
        int 21h
getkey: mov ah,1        ；调用 DOS 功能
        int 21h
        cmp  al,'0'
        jb error        ；小于 0，出错处理
        cmp al,'9'
        ja error        ；大于 9，出错处理
        mov ah,02h      ；调用 DOS 显示字符功能，显示该数字
        mov dl,al
        int 21h
        .exit 0         ；终止程序执行，返回 DOS
error:  mov ah,09h      ；出错，调用 DOS  功能显示 str2 字符串
        mov dx,offset str2
        int 21h
        jmp getkey      ；返回按键
        end             ；汇编结束
```

〔习题 3.7〕将第 2 章习题 2.37 采用完整段定义格式编写成一个完整的源程序。

〔解答〕

```
;xt307.asm
stack   segment
        dw 512 dup(?)
stack   ends
data    segment
```

```
array      db 255
           db 0
array1     db 255 dup('$')
array2     db 0dh,0ah,'$'
data       ends
code       segment 'code'
assume cs:code, ds:data, ss:stack
start:     mov ax,data
           mov ds,ax
           mov ah,0ah              ; 键盘输入字符串
           mov dx,offset array
           int 21h
           mov dx,offset array2    ; 回车换行
           mov ah,09h
           int 21h
           mov bx,offset array1
again:     mov al,[bx]
           cmp al,'$'
           jz done
           cmp al,'a'              ; 小于 a 和大于 z 的字符不是小写字母
           jb next
           cmp al,'z'
           ja next
           sub al,20h              ; 在 a 和 z 之间的字符才是小写字母，转换为大写
           mov [bx],al             ; 保存到原位置
next:      inc bx
           jmp again
done:      mov dx,offset array1
           mov ah,09h
           int 21h
```

```
        mov ax,4c00h
        int 21h
code    ends
        end start
```

〔习题 3.8〕区分下列概念：

（1）变量和标号

（2）数值表达式和地址表达式

（3）符号常量和字符串常量

〔解答〕

（1）变量是在程序运行过程中，其值可以被改变的量；标号是由用户自定义的标识符，指向存储单元，表示其存储内容的逻辑地址。

（2）数值表达式一般是由运算符连接的各种常数所构成的表达式，地址表达式是由名字、标号以及利用各种的操作符形成的表达式。

（3）在程序中，为了使常量更便于使用和阅读,经常将一些常量用常量定义语句定义为符号常量，被一对双引号括起来的若干个字符组成的字符序列被称为字符串常量。

〔习题 3.9〕假设 myword 是一个字变量，mybyte1 和 mybyte2 是两个字节变量，指出下列语句中的错误原因。

（1） mov byte ptr [bx],1000

（2） mov bx,offset myword[si]

（3） cmp mybyte1,mybyte2

（4） mov al,mybyte1+mybyte2

（5） sub al,myword

（6） jnz myword

〔解答〕

（1）1000 超出了一个字节范围

（2）寄存器的值只有程序执行时才能确定，而 offset 是汇编过程计算的偏移地址，故无法确定，改为 lea bx,myword[si]

（3）两个都是存储单元，指令不允许

（4）变量值只有执行时才确定，汇编过程不能计算

（5）字节量 AL 与字量 myword，类型不匹配

（6）Jcc 指令只有相对寻址方式，不支持间接寻址方式

〔习题 3.10〕OPR1 是一个常量，问下列语句中两个 AND 操作有什么区别？

AND AL,OPR1 AND 0feh

〔解答〕

前者为“与”操作硬指令助记符，可汇编成机器代码。

后者为逻辑运算符，在汇编时进行“与”运算，产生具体数值。

〔习题 3.11〕给出下列语句中，指令立即数（数值表达式）的值：

（1） mov al,23h AND 45h OR 67h

（2） mov ax,1234h/16+10h

（3） mov ax,NOT(65535 XOR 1234h)

（4） mov al,LOW 1234h OR HIGH 5678h

（5） mov ax,23h SHL 4

（6） mov ax,1234h SHR 6

（7） mov al,’a’ AND (NOT(’a’-’A’))

（8） mov al,’H’ OR 00100000b

（9） mov ax,(76543 LT 32768) XOR 7654h

〔解答〕

注：对于逻辑运算，有关操作数可化为二进制数。

（1）67h

（2）133h

（3）1234h

（4）76h

（5）0234h

（6）0048h

（7）41h

（8）68h

（9）7654h

〔习题 3.12〕为第 2 章例题 2.54 定义变量 count、block、dplus 和 dminus。

〔解答〕

假设 block 开始的数据块有 32 个字节数据：16 个正数+100 (64h)、16 个负数 -48 （0d0h）

分别连续分布：

```
block    db 16 dup（100），16 dup（-48）；也可以是任意字节数据，随意分布。
dplus    db 32 dup(?)              ；为正数预留存储空间
dminus   db 32 dup(?)              ；为负数预留存储空间
count    equ 32                    ；字节数
```

〔习题 3.13〕为第 2 章例题 2.55 定义相应变量，并形成一个完整的汇编语言程序。

〔解答〕

```
；lt239b.asm
        .model   small
        .stack
        .data
string1    db 'good morning !' ；两字符串可相同或不同，但字符数要求相同。
string2    db 'Good morning !'
result     db ?              ；预留结果字节
    count = 14               ；字符数
        .code
        .startup
        mov ax,ds            ；所有数据在同一个段，所以使 es=ds
        mov es,ax
        mov si,offset string1
        mov di,offset string2
        mov cx,count
 again:   cmpsb
        jnz unmat
        dec cx
        jnz again
        mov al,0
        jmp output
unmat:    mov al,0ffh
output：   mov result, al
```

```
    .exit0
    end
```

〔习题 3.14〕画图说明下列语句分配的存储空间及初始化的数据值：

（1） byte_var DB 'ABC',10,10h,'EF',3 DUP(-1,?,3 DUP(4))

（2） word_var DW 10h,-5,'EF',3 DUP(?)

〔解答〕

（1）从低地址开始，依次是（十六进制表达）：

41 42 43 0a 10 45 46 ff — 04 04 04 ff — 04 04 04 ff — 04 04 04

（2）从低地址开始，依次是（十六进制表达）：

10 00 FB FF 46 45 — — — — — —

〔习题 3.15〕请设置一个数据段 mydataseg，按照如下要求定义变量：

（1） my1b 为字符串变量：Personal Computer

（2） my2b 为用十进制数表示的字节变量：20

（3） my3b 为用十六进制数表示的字节变量：20

（4） my4b 为用二进制数表示的字节变量：20

（5） my5w 为 20 个未赋值的字变量

（6） my6c 为 100 的常量

（7） my7c 表示字符串：Personal Computer

〔解答〕

```
mydataseg segment
my1b    db 'Personal Computer'
my2b    db 20
my3b    db 14h          ;20h
my4b    db 00010100b
my5w    dw 20 dup(?)
my6c    equ 100         ;my6c = 100
my7c    equ <Personal Computer>
mydataseg ends
```

〔习题 3.16〕分析例题 3.2 的数据段，并上机观察数据的存储形式。

〔解答〕

以字节为单位从低地址向高地址依次是：

16

00 12

FFH FFH FFH FFH

00 00 00 00 00 00 00 00

1 2 3 4 5

45H 23H 00 00 00 00 00 00 00 00

‘a’ ‘b’ ‘c’

‘H’ ‘e’ ‘l’ ‘l’ ‘o’ 13 10 ‘$’

12 个字符串’month’，每个字符串从低地址到高地址依次是：’m’ ‘o’ ‘n’ ‘t’ ‘h’

25×4 个字节未定义初值的存储单元，操作系统设置为 0

〔习题 3.17〕修改例题 3.3，现在用字定义伪指令 dw、字串传送指令 movsw 和字符串显示 9 号功能调用实现。

〔解答〕

```
        .model small
        .stack
        .data
source    dw 3433h,3635h
target    dw 40 dup(?),'$'
        .code
        .startup
        mov ax,ds
        mov es,ax
        cld
        mov si,offset source
        mov di,offset target
        mov cx,40
        rep movsw
        mov si,0
```

```
mov dx,offset target
mov ah,9
int 21h
.exit 0
end
```

〔习题 3.18〕变量和标号有什么属性？

〔解答〕

段地址：表示变量和标号所在代码段的段地址；

偏移地址：表示变量和标号所在代码段的段内偏移地址；

类型：引用变量时，表示是字节、字、双字等数据量。引用该标号时，表示它所在同一个段——near 类型，还是另外一个段——far 类型。

〔习题 3.19〕设在某个程序中有如下片段，请写出每条传送指令执行后寄存器 AX 的内容：

```
mydata    segment
          ORG 100H
VARW      DW 1234H,5678H
VARB      DB 3,4
          ALIGN 4
VARD      DD 12345678H
          EVEN
BUFF      DB 10 DUP(?)
MESS      DB 'HELLO'
BEGIN:    MOV AX,OFFSET MESS
          MOV AX,TYPE BUFF+TYPE MESS+TYPE VARD
          MOV AX,SIZEOF VARW+SIZEOF BUFF+SIZEOF MESS
          MOV AX,LENGTHOF VARW+LENGTHOF VARD
          MOV AX,LENGTHOF BUFF+SIZEOF VARW
          MOV AX,TYPE BEGIN
          MOV AX, OFFSET BEGIN
```

〔解答〕

```
MOV AX, OFFSET MESS                    ；AX＝116H
```

MOV AX, TYPE BUFF+TYPE MESS+TYPE VARD ；AX = 1+1+4 = 06H

MOV AX，SIZEOF VARW+SIZEOF BUFF+SIZEOF MESS ；AX = 4+10+5 = 19 = 13H

MOV AX,LENGTHOF VARW + LENGTHOF VARD ；AX = 2+1 = 03H

MOV AX,LENGTHOF BUFF + SIZEOF VARW ；AX = 10+4 =14 = 0EH

MOV AX,TYPE BIGIN ；AX = FF02H (近)

MOV AX,OFFSET BEGIN ；AX = 1BH

〔习题 3.20〕利用简化段定义格式，必须具有.MODEL 语句。MASM 定义了哪 7 种存储模式，TINY 和 SMALL 模式创建什么类型（EXE 或 COM）程序？设计 32 位程序应该采用什么模式？

〔解答〕

MASM 定义的 7 种存储模式是 TINY（微型模式、SMALL（小型模式）、COMPACT（紧凑模式）、MEDIUM（中型模式）、LARGE（大型模式）、HUGE（巨大模式）、FLAT（平展模式）；TINY 用于创建 COM 类型程序、一般程序都可以选用 SMALL 模式；设计 32 位的程序应该采用 FLAT 模式。

〔习题 3.21〕源程序中如何指明执行的起始点？源程序应该采用哪个 DOS 功能调用，实现程序返回 DOS？

〔解答〕

源程序中运用 STARTUP 伪指令指明执行的起始点；源程序应该采用 DOS 功能调用的 4CH 子功能实现程序返回 DOS 的。

〔习题 3.22〕在 SMALL 存储模式下，简化段定义格式的代码段、数据段和堆栈段的缺省段名、定位、组合以及类别属性分别是什么？

〔解答〕

段定义伪指令	段名	定位	组合	类别	组名
.CODE	_TEXT	WORD	PUBLIC	'CODE'	
.DATA	_DATA	WORD	PUBLIC	'DATA'	DGROUP
.DATA?	_BSS	WORD	PUBLIC	'BSS'	DGROUP
.STACK	STACK	PARA	STACK	'STACK'	DGROUP

〔习题 3.23〕如何用指令代码代替.startup 和.exit 指令，使得例题 3.1a 能够在 MASM 5.x 下汇编通过？

〔解答〕

```
;lt301a.asm(文件名)
.model small
    .stack
    .data
```

```
string      db ‘Hello,Everybody !’,0dh,0ah.’$’
            .code
start:      mov ax ,@data
            mov ds,ax
            mov dx,offset string
            mov ah,9
            int 21h
            mov ax ,4c00h
            int 21h
            end start
```

〔习题 3.24〕创建一个 COM 程序完成例题 3.1 的功能。

〔解答〕

```
；lt301a,asm
            .model tiny
            .code
            .startup
            mov dx,offset string
            mov ah,9
            int 21h
            .exit 0
string   db 'Hello,Everybody!'0dh,0ah,'$'   ；
            end
```

〔习题 3.25〕按下面要求写一个简化段定义格式的源程序

（1） 定义常量 num，其值为 5；数据段中定义字数组变量 datalist，它的头 5 个字单元中依次存放 -1、0、2、5 和 4，最后 1 个单元初值不定；

（2） 代码段中的程序将 datalist 中头 num 个数的累加和存入 datalist 的最后 1 个字单元中。

〔解答〕

```
            .model small
            .stack
            .data
```

```
num         equ 5
datalist    dw -1,0,2,5,4,?
            .code
            .startup
            mov bx,offset datalist
            mov cx,num
            xor ax,ax

again:      add ax,[bx]
            inc bx
            inc bx
            loop again
            mov [bx],ax
            .exit 0
            end
```

〔习题 3.26〕按下面要求写一个完整段定义格式的源程序

（1）数据段从双字边界开始，其中定义一个 100 字节的数组，同时该段还作为附加段；

（2）堆栈段从节边界开始，组合类型为 stack；

（3）代码段的类别是'code'，指定段寄存器对应的逻辑段；主程序指定从 100h 开始，给有关段寄存器赋初值；将数组元素全部设置为 64h。

〔解答〕

```
stack       segment para 'stack'
dw   512 dup(?)
stack       ends
data        segment
array       db 100 dup(?)
data        ends
code        segment 'code'
assume cs:code,ds:data,es:data,ss:stack
org 100h
```

```
start:    mov ax,data
          mov ds,ax
          mov es,ax
          mov di,offset array
          mov al,64h
          mov cx,100
          cld
          rep stosb
          mov ax,4c00h
          int 21h
code      ends
          end start
```

〔习题 3.27〕编制程序完成两个已知双精度数（4 字节）A 和 B 相加并将结果存入双精度变量单元 SUM 中（不考虑溢出）。

〔解答〕

```
；xt327.asm
          .model   small
          .stack   256        ；定义堆栈段大小为 256 个字节
          .data
A         dd   11223344h      ；定义两个双字的数（随意）
B         dd   77553311h
sum       dd   ?              ；定义结果，执行后为：88776655h
          .code
          .startup
          xor   si, si        ；相对于变量的位移量清零
          mov   cx, 2         ；分高低字分别相加，共两次
          clc                 ；清零 cf
again:    mov ax, word ptr A[si]    ；取第一个数的一个字（先低字后高字）
          adc ax, word ptr B[si]    ；取第二个数的一个字（先低字后高字）
          mov word ptr sum[si], ax  ；存和的一个字（先低字后高字）
```

```
        inc si              ；修改位移量指向下一个字（加 2）
        inc si
        loop   again        ；cx=cx-1 ,if cx<>0 ,jump again
        .exit 0
        end
```

〔习题 3.28〕编制程序完成 12H、45H、0F3H、6AH、20H、0FEH、90H、0C8H、57H 和 34H 等 10 个字节数据之和，并将结果存入字节变量 SUM 中（不考虑溢出）。

〔解答〕

```
        .startup
            xor    si, si          ；位移量清零
            mov al, bdata[si]      ；取第一个数
            mov cx, num-1          ；累加次数
        again:    inc   si         ；指向下一个数
            adc al,   bdata[si]    ；累加
            loop   again           ；如未完，继续累加
            mov   sum, al          ；完了，存结果
            .exit 0
            end
```

〔习题 3.29〕结构数据类型如何说明、结构变量如何定义、结构字段如何引用？

〔解答〕

结构类型的说明使用一对伪指令 STRUCT（MASM5.x 是 STRUC，功能相同）和 ENDS。它们的格式为：

结构名 STRUCT

…　　;数据定义语句

结构名 ENDS

结构变量定义的格式为：

变量名，结构名　〈字段初值表〉

引用结构字段，采用圆点“.”操作符，其格式是：

结构变量名.结构字段名。

〔习题 3.30〕记录数据类型如何说明，记录变量如何定义，width 和 mask 操作符是什么作用？

〔解答〕

记录类型的说明采用伪指令 RECORD，它的格式为：

记录名 RECORD 位段[，位段…]

定义记录变量的格式：

记录变量名 记录名 〈段初值表〉

Width 记录名/记录位段名操作符返回记录或记录位段所占用的位数。

mask 记录位段名操作符返回一个 8 位或 16 位数值，其中对应该位段的个位为 1，其余位为 0。

第 4 章 基本汇编语言程序设计

〔习题 4.1〕例题 4.2 如果要求算术右移 8 位，如何修改程序。

〔解答〕

思路： 首先由最高位字节向次低位字节传送……次低位字节向最低位字节传送（共 7 次）；再判最高位字节符号位，如为 0，送 00h 到最高位字节；如为 1，送 ffh 到最高位字节。传送可参考例题 4.2，不过应从第一号字节送第零号字节，……最高位字节向次低位字节传送；也可以用循环来完成：

```
        .model small
        .stack 256
        .data
qvar    dq  1234567887654321h
        .code
        .startup
        mov cx,7
        mov si,1
again:  mov al, byte ptr qvar[si]
        mov byte ptr qvar[si-1],al
        inc si
        loop again
        test al,80h
        jz ezz
        mov bl,0ffh
        jmp  done
ezz:    mov bl,0
done:   mov byte ptr qvar[7],bl
        .exit 0
        end
```

〔习题 4.2〕例题 4.2 如果要求算术左移 7 位，如何用移位指令实现。

〔解答〕

思路：可设计外循环体为 8 个字节左移一次，方法是：最低位字节算术左移一次，　　次低位字节至最高位字节依次带 CF 位循环左移一次（内循环共 8 次），外循环体控制执行 7 次即可。

```
        .model small
        .stack 256
        .data
qvar    dq   1234567887654321h
        .code
        .startup
        mov dx, 7                    ；外循环次数
        mov ax, byte ptr qvar[0]     ；最低位字节送 ax
lpp:    shl ax, 1                    ；最低位字节左移一次，其 d7 移入 CF 位
        mov   si, 1
        mov cx, 7                    ；内循环次数
again:  rcl byte ptr qvar[si], 1     ；高位字节依次左移 P50
        inc   si
        loop   again
        dec dx
        jnz   lpp
        .exit 0
        .end
```

〔习题 4.3〕将 AX 寄存器中的 16 位数连续 4 位分成一组，共 4 组，然后把这 4 组数分别放在 AL、BL、CL 和 DL 寄存器中。

〔解答〕

思路：设这四组从低位到高位分别放在 AL、BL、CL 和 DL 寄存器中。仅列出代码段：

```
        mov bl, al      ；将 al 中的两组分开
        and al, 0fh     ；屏蔽高四位后送 al
        mov cl, 4       ；原 al 中的数据逻辑右移 4 次送 bl
        shr bl, cl
        mov dl, ah      ；将 ah 中的两组分开
        and dl, 0f0h    ；屏蔽低高四位后送 dl
        mov cl, 4       ；原 ah 中的数据逻辑右移 4 次送 dl
        shr dl, cl
        mov cl, ah      ；屏蔽高四位后送 cl
        and cl, 0fh
```

〔习题 4.4〕编写一个程序，把从键盘输入的一个小写字母用大写字母显示出来。

〔解答〕

```
getkey:   mov ah, 1          ；从键盘输入，出口:al 存键值
          int 21h
          cmp   al, 'a'      ；判键值是小写字母？
           jb   getkay
```

```
        cmp   al, 'z'
        ja   getkay
        sub al,20h          ；是小写字母转换为大写字母
        mov dl,al
        mov ah, 02h         ；显示
        int 21h
```

〔习题 4.5〕已知用于 LED 数码管显示的代码表为：

```
LEDtable    DB 0c0h,0f9h,0a4h,0b0h,99h,92h,82h,0f8h
            DB 80h,90h,88h,83h,0c6h,0c1h,86h,8eh
```

它依次表示 0～9、A～F 这 16 个数码的显示代码。现编写一个程序实现将 lednum 中的一个数字（0～9、A～F）转换成对应的 LED 显示代码。

〔解答〕

```
        .model small
        .stack 256
        .data
LEDtable    DB 0c0h，0f9h，0a4h,0b0h,99h,92h，82h，0f8h
            DB 80h，90h，88h，83h，0c6h，0clh、86h,8eh
 lednum     DB ?
        .code
        .startup
        mov bx, offset LEDtable
        mov al, lednum
        xlat                ；al 中存有对应的 LED 显示代码
        .exit 0
        end
```

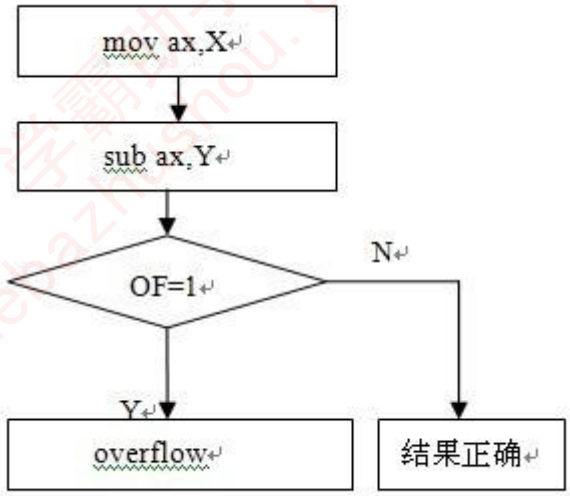
〔习题 4.6〕请问第 2 章例题 2.39 / 2.41 / 2.43 / 2.46 的分支是单分支、双分支或多分支结构？

〔解答〕

例题 2.43/2.45 / 2.47 为单分支结构；2.50 为多分支结构。

〔习题 4.7〕分析第 2 章例题 2.39 的分支结构，画出流程图。

〔解答〕



〔习题 4.8〕如果在例题 4.4 的 table 中依次填入 msg1～msg8，程序应该怎样修改？
〔解答〕
（1） 将 jmp table[bx]指令改为：mov dx, table[bx]
（2） 去掉源程序中：.exit 0---end 之间的语句

〔习题 4.9〕编制一个程序，把变量 bufX 和 bufY 中较大者存入 bufZ；若两者相等，则把其中之一存入 bufZ 中。假设变量存放的是 8 位无符号数。
〔解答〕

```
        .model small
        .stack 256
        .data
bufx      db ?
bufY      db ?
bufz      db ?
        .code
        .startup
        mov al, bufX
        mov bl, bufY
        cmp al,bl
        ja next
        mov bufZ, bl
        jmp done
    next: mov bufZ, al
done:     .exit 0
        end
```

〔习题 4.10〕设变量 bufX 为有符号 16 位数，请将它的符号状态保存在 signX，即：如果 X 大于等于 0，保存 0；如果 X 小于 0，保存-1（ffh）。编写该程序。
〔解答〕

```
        .model small
```

```
        .stack
        .data
bufX    dw -7
signX   db ?
        .code
        .startup
        cmp bufX,0          ;test bufX,80h
        jl next             ;jnz next
        mov signX,0
        jmp done
next:   mov signX,-1
done:   .exit 0
        end
```

〔习题 4.11〕bufX、bufY 和 bufZ 是 3 个有符号 16 进制数，编写一个比较相等关系的程序：
（1）如果这 3 个数都不相等，则显示 0；
（2）如果这 3 个数中有两个数相等，则显示 1；
（3）如果这 3 个数都相等，则显示 2。
〔解答〕

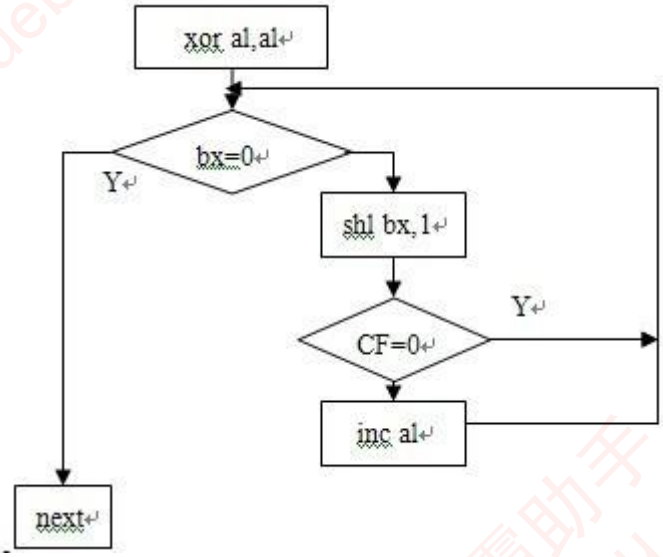
```
 .model small
        .stack 256
        .data
bufx    dw ?
bufY    dw ?
bufz    dw ?
        .code
        .startup
        mov ax,  bufX
        mov bx,  bufY
        mov cx,  bufZ
        mov dl, '0'
        cmp ax,bx
        jnz next1
        inc dl
next1:  cmp ax,cx
        jnz next2
        inc dl
next2:  cmp bx,cx
        jnz next3
        inc dl
next3:  cmp dl,'3'
        jb next4
        mov dl,'2'
next4:  mov ah,02h      ；显示
```

```
int 21h
.exit 0
end
```

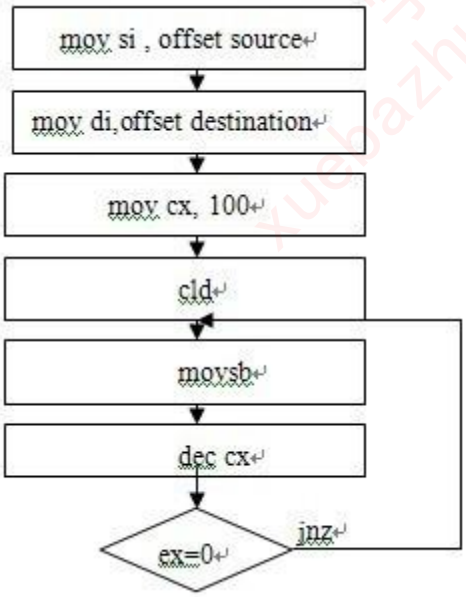
〔习题 4.12〕分析第 2 章例题 2.42、2.52、2.53 的结构，分别画出它们的流程图。

〔解答〕

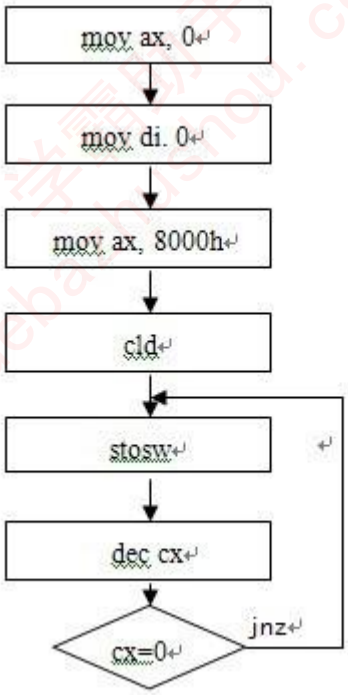
2.42



2.52



2.53



〔习题 4.13〕例题 4.8 内外循环次数共是多少？如果要求按从大到小排序，程序如何修改？

〔解答〕

外循环次数是： count-1 次 （19 次）

内循环次数是： （count-1)!次 (19! 次)

内外循环次数共是 count-1 + （count-1）!次 ，即 19+19！次

〔习题 4.14〕串操作指令常要利用循环结构，现在不用串操作指令实现字符串 string1 内容传送到字符串 string2，字符长度为 count。

〔解答〕

```
 .model   small
          .stack 256
          .data
  string1    db   'good morning!'
  len     =   $-string1
  string2    db   len dup(?)
          .code
          .startup
          mov cx, len            ；字符数
          mov si, offset string1    ；源指针
          mov di, offset string2    ；目标指针
again:      mov al, [si]             ；送一个字符
          mov [di], al
          inc si                ；修改指针
          inc di                ；修改指针
         loop again               ；cx=cx-1,cx=0 时退出循环
          .exit 0
```

```
        end
```

〔习题 4.15〕不用串操作指令求主存 0040h : 0 开始的一个 64KB 物理段中共有多少个空格？

〔解答〕

这里仅列出主程序段：

```
        mov ax,0040h     ；送段地址
        mov ds, ax
        xor si, si       ；偏移量地址
        xor cx, cx       ；计数（循环次数）
        xor dx, dx       ；空格计数器清零
again:    cmp [si], 20h   ；与空格的 ASCII 码比较
        jne   next       ；不是空格，转
        inc dx           ；是空格，空格数加 1
next:     inc si         ；修改地址指针
       loop again        ；cx=cx-1,如 cx=0  退出循环
        .exit 0
        end
```

〔习题 4.16〕编程实现把键入的一个字符，用二进制形式（0 / 1）显示出它的 ASCII 代码值。

〔解答〕

```
.model small
        .stack 256
        .data
   stri1   db 'please input',0dh,0ah,'$'
        .code
        .startup
        mov   dx,offset stri1
        mov ah,09h
        int 21h
        mov ah, 01h
        int 21h
        mov cx, 8
  again:   xor dl,dl
          shl al, 1
           adc dl, '0'
        mov ah,02h
        int 21h
       loop again
      .exit 0
       end
```

〔习题 4.17〕编写程序，要求从键盘接收一个数 bellN（0～9），然后响铃 bellN 次。

〔解答〕

```
.model small
```

```
        .stack
        .data
  stri1   db 'please input number:1--9',0dh,0ah,'$'
        .code
        .startup
 again:   mov  dx,offset stri1   ；显示 stri1，提示输入
        mov ah,09h
        int 21h
        mov ah,01h          ；调用输入一个字符
        int  21h            ；输入一个字符存在 al 中
        cmp al, '1'         ；判该字符，如不在‘1’--‘9’
        jb  again           ；重新输入
        cmp al, '9'
        ja  again
        and al,0fh          ；在‘1’--‘9’，屏蔽高 4 位
        mov cl, al          ；振铃次数送 cx
       xor ch,ch
 abc:     mov dl, 07h         ；调用一次振铃
        mov ah, 02h
        int 21h
       loop abc
        .exit 0
        end
```

〔习题 4.18〕编写程序，将一个包含有 20 个有符号数据的数组 arrayM 分成两个数组：正数数组 arrayP 和负数数组 arrayN，并分别把这两个数组中的数据个数显示出来。

〔解答〕

```
        .model small
       include io.inc
      .stack
      .data
arrayM   db  1,2,3,4,5,6,0,-3,-5,-6,0,7,6,90,-18,-23,34,7,9,8   ；源数组
arrayP   db  20 dup(?)        ；正数数组
arrayN   db  20 dup(?)        ；负数数组
dispP    db 'Plus Number: ','$'
dispN    db 0dh,0ah,'Negs Number: ','$'
      .code
      .startup
      mov cx, 20            ；源数组元素数
      xor bx, bx            ；设 bh 为正数个数，bl 为负数个数，均清零
      xor si, si            ；源数组地址位移量
    again:                  ；循环 20 次
      mov ax, arrayM[si]    ；取一个元素
      cmp ax,0              ；判正数
```

```
        jl Neg
          inc bh          ；是，正数个数加 1
        jmp next
  Neg:    inc bl          ；否，负数个数加 1
  next:   inc si          ；修改位移量
        loop again        ；循环次数减 1
        mov ah，09h       ；调用显示字符串功能
        mov dx, offset dispP   ；显示个数
        int  21h
         mov al,bh
         call dispuib       ; 调用 I/O 子程序库中的子程序
        mov ah，09h       ；调用显示字符串功能
        mov dx, offset dispN   ；显示个数
        int  21h
         mov al,bl
         call dispuib       ; 调用 I/O 子程序库中的子程序
        .exit 0
        end
```

〔习题 4.19〕编写计算 100 个正整数之和的程序。如果和不超过 16 位字的范围（65535），则保存其和到 wordsum，如超过则显示‘overflow’。

〔解答〕

```
        .model small
        .stack
        .data
num       equ 100
wlist     dw num dup(?)
wordsum   dw ?
error     db 'overflow. $'
        .code
        .startup
        mov bx,offset wlist
        mov cx,num
        xor ax,ax
again:    add ax,[bx]
        jc next
        inc bx
        inc bx
        loop again
        mov [bx],ax
        jmp done
next:     mov dx,offset error
        mov ah,9
        int 21h
```

```
done:     .exit 0
          end
```

〔习题 4.20〕编程判断主存 0070h：0 开始的 1KB 中有无字符串‘DEBUG’。这是一个字符串包含的问题，可以采用逐个向后比较的简单算法。

〔解答〕

```
          .model small
           .stack
           .data
  disp1     db 'There is   DEBUG in the aera!' ,0dh,0ah,‘$'
  disp2     db 'There is no   DEBUG in the aera!' ,0dh,0ah,‘$'
          .code
          .startup
          mov ax, 0070h       ；送段地址
          mov ds, ax
          xor si, si          ；地址指针清零
        mov cx,1024
           cmp [si], 'D'     ；与‘D'比较
           jne next          ；不是，转
           inc si            ；是，地址增 1
           cmp [si], 'E'     ；同上
           jne next
           inc si
           cmp [si], 'B'
           jne next
           inc si
           cmp [si], 'U'
           jne next
           inc si
          cmp [si], 'G'
          je yes             ；是‘DEBUG',转
    next:     inc si          ；不是，地址增 1
        loop again            ；循环
  no:     mov dx, offset disp2   ；没找到，显示 disp2
        jmp   dsp
  yes:    mov dx, offset disp1   ；找到，显示 disp1
  dsp:    mov ah, 09h
        int 21h
        .exit 0
        end
```

〔习题 4.21〕编程把一个 16 位无符号二进制数转换成为用 8421BCD 码表示的 5 位十进制数。转换算法可以是：用二进制数除以 10000，商为“万位”，再用余数除以 1000，得到“千位”；依次用余数除以 100、10 和 1，得到“百位”、“十位”和“个位”。

〔解答〕

```
        .model small
       .stack 256
       .data
 var      dw   3546
 dbcd     db    5 dup(?)
      .code
     .startup
        mov ax, var
        mov bx, 10000
        mov cl, 10
        xor si, si
        xor dx, dx
 again:          div   bx
             mov dbcd[si], al
             inc   si
              xchg ax, bx
              div   cl
              xchg ax, bx
              cmp si,5
              jnz again
         .exit 0
          end
```

〔习题 4.22〕过程定义的一般格式是怎样的？子程序入口为什么常有 PUSH 指令、出口为什么有 POP 指令？下面的程序段有什么不妥吗？若有，请改正：

```
crazy      PROC
        push ax
        xor ax,ax
        xor dx,dx
again:     add ax,[bx]
        adc dx,0
        inc bx
        inc bx
        loop again
        ret
        ENDP crazy
```

〔解答〕

```
crazy      PROC              ；crazy PROC
        push ax           ；
        xor ax,ax         ；    xor ax,ax
        xor dx,dx         ；    xor dx,dx
again:     add ax,[bx]       ；again:    add ax,[bx]
        adc dx,0          ；    adc dx,0
```

```
        inc bx          ;    inc bx
        inc bx          ;    inc bx
        loop again      ;    loop again
        ret             ;    ret
        ENDP crazy      ;    crazy    ENDP
```

〔习题 4.23〕子程序的参数传递有哪些方法，请简单比较。
〔解答〕
寄存器、共享变量（公共存储单元）、堆栈
用寄存器传递参数是把参数存于约定的寄存器中，这种方法简单易行，经常采用；
用变量传递参数是主程序与被调用过程直接用同一个变量名访问传递的参数，就是利用变量传递参数。如果调用程序与被调用程序在同一个源程序文件中，只要设置好数据段寄存器 DS，则子程序与主程序访问变量的形式相同，也就是它们共享数据段的变量，调用程序与被调用程序不在同一个源文件中，必须利用 public/extern 进行声明，才能用变量传递参数，利用变量传递参数，过程的通用性比较差，然而，在多个程序段间，尤其在不同程序的模块间，利用全局变量共享数据也是一种常见的参数传递方法；
用堆栈传递参数是主程序将子程序的入口参数压入堆栈，子程序从堆栈中取出参数；子程序将出口压入堆栈，主程序弹出堆栈取得它们。

〔习题 4.24〕采用堆栈传递参数的一般方法是什么，为什么应该特别注意堆栈平衡问题。
〔解答〕
采用堆栈传递参数的一般方法是主程序将子程序的入口参数压入堆栈，子程序从堆栈中取出参数子程序将出口参数压入堆栈，主程序弹出堆栈取得它们。因为堆栈采用“先进后出”原则存取，而且返回地址和保护的寄存器等也要存于堆栈，所以要特别注意堆栈平衡问题。

〔习题 4.25〕什么是子程序的嵌套、递归和重入？
〔解答〕
子程序中又调用子程序就形成子程序嵌套。
子程序中直接或间接调用该子程序本身就形成子程序递归。
子程序的重入是指子程序被中断后又被中断服务程序所调用，能够重入的子程序称为可重入子程序。

〔习题 4.26〕将例题 4.7 的大写转换为小写字母写成过程，利用 AL 作为入口、出口参数完成。
〔解答〕

```
.model small
        .stack 256
        .data
  stdng   db 'HeLLO eveRyboDy ! ' , 0
        .code
        .startup
        mov bx, offset atring
```

```
again: mov al, [bx]
       call chan          ；调用过程
       mov [bx] , al
next: inc bx
       jmp again
done: .exit 0
chan   proc               ；大写转换为小写字母的过程
       or al, al
       jz done
       cmp al,'A'
       jb next
       cmp al, 'Z'
       ja next
       or al, 20h
       ret
chan   endp
   end
```

〔习题 4.27〕请按如下子程序说明编写过程：

```
;子程序功能：把用 ASCII 码表示的两位十进制数转换为对应二进制数
;入口参数：DH=十位数的 ASCII 码，DL=个位数的 ASCII 码
;出口参数：AL=对应的二进制数
```

〔解答〕

```
astob     proc
          and dh, 0fh    ；十位数的 ASCII 码转为二进制数
          mov al, dh
          mul 10         ；al= 10*dh
          and dl, 0fh    ；个位数的 ASCII 码转为二进制数
          add al, dl     ；al= 10*dh + dl
          ret
astob     endp
```

〔习题 4.28〕写一个子程序，根据入口参数 AL=0/1/2，分别实现对大写字母转换成小写、小写转换成大写或大小写字母互换。欲转换的字符串在 string 中，用 0 表示结束。

〔解答〕

```
Change    proc
          Push   bx                ；保护 bx
          xor    bx, bx            ；位移量清零
          cmp al,0                 ；根据入口参数 AL=0/1/2，分别处理
          jz   chan_0
           dec al
          jz   chan_1
           dec al
          jz   chan_2
```

```
        jmp done
chan_0:    mov al,string[bx]                ；实现对大写字母转换成小写
        cmp al,0
        jz done
         cmp al,'A'        ；是大写字母
        jb next0
         cmp al,'Z'        ；是大写字母
        ja next0
          add   al, 20h         ；转换
          mov   string[bx], al
next0:      inc   bx              ；位移量加 1，指向下一字母
          jmp   chan_0
chan_1:    mov al,string[bx]                ；实现对小写字母转换成大写
        cmp al,0
        jz done
          cmp al,'a'        ；是大写字母
        jb next1
          cmp al,'z'        ；是大写字母
        ja next1
          sub   al, 20h         ；转换
          mov   string[bx], al
next0:      inc   bx              ；位移量加 1，指向下一字母
          jmp   chan_1
chan_2:    mov al,string[bx]                ；实现对大写字母转换成小写
        cmp al,0
        jz done
          cmp al,'A'        ；是大写字母
        jb next2
          cmp al,'Z'        ；是大写字母
        ja next20
          add   al, 20h         ；转换
        jmp next2
next20:    cmp al,'a'        ；是大写字母
        jb next2
          cmp al,'z'        ；是大写字母
        ja next2
          sub   al, 20h         ；转换
          mov   string[bx], al
next2:      inc   bx              ；位移量加 1，指向下一字母
          jmp   chan_2
done:      pop bx               ；恢复 bx
        ret
change   endp
```

〔习题 4.29〕编制一个子程序把一个 16 位二进制数用十六进制形式在屏幕上显示出来，分别运用如下 3 种参数传递方法，并用一个主程序验证它。

（1）采用 AX 寄存器传递这个 16 位二进制数

（2）采用 wordTEMP 变量传递这个 16 位二进制数

（3）采用堆栈方法传递这个 16 位二进制数

〔解答〕

（1）采用 AX 寄存器传递这个 16 位二进制数

```
        .model small
        .stack
        .data
wdata     dw 34abh
        .code
        .startup
        mov ax,wdata
        call dispa
        .exit 0
        ;
dispa     proc
        push cx
        push dx
        mov cl,4
        mov dl,ah
        shr dl,cl
        call dldisp
        mov dl,ah
        and dl,0fh
        call dldisp
        mov dl,al
        shr dl,cl
        call dldisp
        mov dl,al
        and dl,0fh
        call dldisp
        pop dx
        pop cx
        ret
dispa     endp
        ;
dldisp    proc
        push ax
        or dl,30h
        cmp dl,39h
        jbe dldisp1
        add dl,7
```

```
dldisp1:    mov ah,2
            int 21h
            pop ax
            ret
dldisp      endp
            end
```

（2）采用 wordTEMP 变量传递这个 16 位二进制数

```
            .model small
            .stack
            .data
wdata       dw 34abh
wordtemp    dw ?
            .code
            .startup
            mov ax,wdata
            mov wordtemp,ax
            call dispa
            .exit 0
            ;
dispa       proc
            push cx
            push dx
            mov cl,4
            mov dl,byte ptr wordtemp+1
            shr dl,cl
            call dldisp
            mov dl,byte ptr wordtemp+1
            and dl,0fh
            call dldisp
            mov dl,byte ptr wordtemp
            shr dl,cl
            call dldisp
            mov dl,byte ptr wordtemp
            and dl,0fh
            call dldisp
            pop dx
            pop cx
            ret
dispa       endp
            ;
dldisp      proc
            push ax
            or dl,30h
            cmp dl,39h
```

```
        jbe dldisp1
        add dl,7
dldisp1:    mov ah,2
        int 21h
        pop ax
        ret
dldisp    endp
        end
```

（3）采用堆栈方法传递这个 16 位二进制数

```
        .model small
        .stack
        .data
wdata       dw 34abh
        .code
        .startup
        push wdata
        call dispa
        pop ax              ;add sp,2
        .exit 0
        ;
dispa       proc
        push bp
        mov bp,sp
        push ax
        push cx
        push dx
        mov ax,[bp+4]
        mov cl,4
        mov dl,ah
        shr dl,cl
        call dldisp
        mov dl,ah
        and dl,0fh
        call dldisp
        mov dl,al
        shr dl,cl
        call dldisp
        mov dl,al
        and dl,0fh
        call dldisp
        pop dx
        pop cx
        pop ax
        pop bp
```

```
        ret
dispa    endp
        ;
dldisp   proc
        push ax
        or dl,30h
        cmp dl,39h
        jbe dldisp1
        add dl,7
dldisp1:  mov ah,2
        int 21h
        pop ax
        ret
dldisp   endp
        end
```

〔习题 4.30〕设有一个数组存放学生的成绩（0～100），编制一个子程序统计 0～59 分、60～69 分、70～79 分、80～89 分、90～100 分的人数，并分别存放到 scoreE、scoreD、score C、score B 及 score A 单元中。编写一个主程序与之配合使用。

〔解答〕

```
        .model small
        .stack
        .data
score    db 70,86,90,45,60,96,100,0,...   ；全班成绩数组
count    equ  $-score        ；总人数
scoreE   db ?              ；0--59 分人数
scoreD   db ?              ；60--69 分人数
scoreC   db ?              ；70--79 分人数
scoreB   db ?              ；80--89 分人数
scoreA   db ?              ；90--99 分人数
        .code
        .startup
        lea bx, score
        mov cx, count
again:
        mov al,[bx]        ；取一个成绩
        call tjrs          ；调用统计分段人数
        inc bx             ；调整指针
        loog again         ；cx-cx-1,cx=0 退出循环
        .exit 0
tjrs   proc ；统计分段人数
       cmp al, 60
       jae  next0          ；al>= 60 转
       inc  scoreE         ；al<60,0--59 分的人数加 1
```

```
        jmp next4
next0:  cmp al, 70
        jae  next1
        inc  scoreD
        jmp next4
next1:  cmp al, 80
        jae  next2
        inc  scoreC
        jmp next4
next2:  cmp al, 90
        jae  next3
        inc  scoreB
        jmp  next4
next3:   inc  scoreA
next4:   ret
tjrs     endp
        end
```

〔习题 4.31〕编写一递归子程序，计算指数函数 X n 的值。

〔解答〕

```
model small
        .stack
        .data
x       dw  5
n       dw  6
zish    dw ?
        .code
        .startup
        mov bx,x
        push bx
        mov bx,n
        push bx
        mov bx,x
        call zshhsu
        pop zish
        add sp,2
        .exit 0
zshhsu  proc
        push bx
        push ax
        push bp
        mov bp,sp
        mov ax,[bp+8]
        cmp ax,0
```

```
        jne zshhsu1
        inc ax
        jmp zshhsu2
zshhsu1:  dec ax
        push bx
        push ax
        call  zshhsu
        pop ax
        pop bx
        mul bx
zshhsu2:  mov [bp+8],ax
        pop bp
        pop ax
        pop bx
        ret
zshhsu  endp
        end
```

5.27pc 扬声器的声音频率是可调的

```
            .model tiny
            .code
           .startup
            mov dx,offset msgl
            mov ah,09h
            int 21h
again :    mov bx, Hz1-8
            mov ah,01h
            int 21h
            cmp al,1bh
            jz done
            cmp al,31h
            jb off
            cmp al,38h
            ja off
            and al,0cfh
            dec al
off:        call speaker-off
            jmp again
done:    .exit 0
speaker-on proc
            push ax
            in al,61h
            and al,0fch
            out 61h,al
            pop ax
```

```
        ret
speak-off endp
speaker proc
        push ax
        mov al,0b6h
        out 43h,al
        pop ax
        out 42h,al
        mov al,ah
        out 42h,al
        ret
speaker endp
asgl db’Please intput(1-8),or press ASCII!’,0bh,0ah,”$”
Hzl_8 dw 524,588,660,698,784,880,988,1048
    end
```