

# 汇编语言习题答案

## 第一章

### 1.4 汇编语言与高级语言相比有什么优缺点？

答：

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。

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。

### 1.5 将下列十六进制数转换为二进制和十进制表示。

(1)FFH (3)5EH (5)2EH (7)1FH

答：

(1) FFH=11111111B=255D

(3) 5EH=1011110B=94D

(5) 2EH=101110B=46D

(7) 1FH=11111B=31D

### 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。

(2)-127 (4)-57 (6)-126 (8)68

答：

十进制数	原码	反码	补码
-127	11111111	10000000	10000001
-57	10111001	11000110	11000111
-126	11111110	10000001	10000010
68	01000100	01000100	01000100

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？

答：

十进制无符号数：97    BCD 码：61    ASCII 码：a

**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。**

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。



状态标志：

0：进位标志 CF    2：奇偶标志 PF    4：调整标志 AF    6：零标志 ZF

7：符号标志 SF    11：溢出标志 OF

控制标志：

8：陷阱标志 TF    9：中断允许标志 IF    10：方向标志 DF

**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**

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 ~ FFFFFFFH。

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。

(1) FFFFh:0=FFFF0H

(2) 40h:17h=00417H

(3) 2000h:4500h=24500H

(4) B821h:4567h=BC777H

**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]**

**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

## **第二章**

**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]**

**答：**

- (1) AX=1200H
- (2) AX=0100H
- (3) AX=4C2AH ; 偏移地址=1200h
- (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  $=bx+si+1100h=0100h+0002h+1100h=1202h$

## 2.4 什么是堆栈，它的工作原理是什么，它的基本操作有哪两个，对应哪两种指令？

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；它的工作原理是先进后出；堆栈的两种基本操作是压栈和出栈，对应的指令是 PUSH 和 POP。

## 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
```

答：

mov al, 89h	al=89h	
add al, al	al=12h	CF=1, ZF=0, SF=0, OF=1, PF=1
add al, 9dh	al=0afh	CF=0, ZF=0, SF=1, OF=0, PF=1
cmp al, 0bch	al=0afh	CF=1, ZF=0, SF=1, OF=0, PF=1
sub al, al	al=00h	CF=0, ZF=1, SF=0, OF=0, PF=1
dec al	al=0ffh	CF=0, ZF=0, SF=1, OF=0, PF=1
inc al	al=00h	CF=0, ZF=1, SF=0, OF=0, PF=1

## 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。**

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

**2.19 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]**

答:

- (1)转移的有效地址 EA= BX=1256H
- (2)转移的有效地址 EA= [DS:20A1H+1256H]=232F7H=3280H
- (3)转移的有效地址 EA= [DS:1256H+528FH]=264E5H=2450H

**2.29 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**

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。

### 第三章

**3.7 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`

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，无法确定。
- (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

**3.10 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(?)`

答：

11) 存储单元	偏移地址	12) 存储单元	偏移地址
04h	0015H	—	
04h		—	000CH
04h		—	
—		—	0002H
ffh		—	
04h		—	0008H
04h		45h	
04h		46h	0004H
—	----	ffh	
ffh		fbh	0002H
04h		00h	
04h		10h	0000H
04h			
—			
ffh			
46h			
45h			
10h			
0ah	0003H		
43h	0002H		
42h	0001H		
41h	0000H		

### 3.18 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？

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

### 3.21 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。

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
```

## 第四章

### 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。

答:

```
.model small
.stack
.data
.code
.startup
-getkey: mov ah, 01h
          int 21h      ; 等待按键
          cmp al, 'a'
          jb -getkey    ; 小于'a'
          cmp al, 'z'
          ja -getkey    ; 大于'z'
          sub al, 20h    ; 转换为大写
          mov dl, al
          mov ah, 02h    ; 显示
          int 21h
          .exit 0
end
```



**4.6 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。**

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
```

**4.23 子程序的参数传递有哪些方法，请简单比较。**

答：

(1) 寄存器：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；

(2) 共享变量（公共存储单元）：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；

(3) 堆栈：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。

#### 4.25 什么是子程序的嵌套、递归和重入？

答：

(1) 子程序中又调用子程序就形成子程序嵌套。

(2) 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。

(3) 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。

#### 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。

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'          ; 是小写字母
        jb next1
        cmp al, 'z'          ; 是小写字母
        ja next1
        sub al, 20h          ; 转换
        mov string[bx], al
next1:
        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 next20
        cmp al, 'Z'          ; 是大写字母
        ja next20
        add al, 20h          ; 转换
        jmp next2
next20:
        cmp al, 'a'          ; 是小写字母
        jb next2
        cmp al, 'z'          ; 是小写字母
        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
```

## 第五章

### 5.5 宏是如何定义、调用和展开的？

答：

宏定义采用一对伪指令实现，格式如下（其中方括号表示可选）：

宏名 macro 【形参表】

宏定义体

endm

宏调用时，利用宏名带上实参即可，格式如下：

宏名【实参表】

### 5.6 宏定义中的形式参数有什么特点，它是如何进行形参和实参结合的？

答：

宏的参数功能非常强大，即可以没有参数，也可以带有一个或多个参数；而且参数的形式非常灵活，可以是常量、变量、存储单元、指令或它们的一部分，还可以是表达式等。

宏展开时的形参与实参结合是用位置匹配的实参对形参进行取代。实参个数与形参个数不同时，多余的实参没有作用，缺少的实参则用空格替代。取代时，不进行有效性等检查。

### 5.7 宏结构和子程序在应用中有什么不同，如何选择采用何种结构？

答：

宏汇编的特点是在汇编阶段进行语句展开，无需返回，不减少目标程序代码，执行速度没有改变；通过形参、实参结合传递参数。宏汇编适合于程序段较短或要求执行速度快的场合。

子程序在执行阶段需要由主程序执行调用指令 CALL 进行调用，进而转入子程序

执行，子程序执行结束执行返回指令 RET 返回主程序。子程序如果被多次使用，则会减少目标程序代码；主程序与子程序之间主要采用寄存器、存储单元、堆栈等传递参数。子程序适合于程序段较长或为减少目标代码的场合。

### 5.16 利用宏结构完成以下功能：如果名为 byteX 的数据大于 5 时，指令“ADD AX, AX”将汇编 10 次，否则什么也不汇编。

答：

```
compare_5 macro byteX
    cmp byteX, 05h
    jbe done
    xor cx, cx
next:
    add ax, ax
    loop next
done:
    endm
```

### 5.19 将例 4.7 的大写字母转换为小写字母写成宏完成。

#### 例 4.7

例 4.7 把一个字符串中的所有大写字母改为小写字母，该字符串以 '0' 结尾。

分析：这是一个循环次数不定的循环程序结构，宜用转移指令决定是否循环结束，并应该先判断后循环。循环体判断每个字符，如果是大写字母则转换为小写，否则不予处理。循环体中具有分支结构。大小写字母的 ASCII 码不同之处是：大写字母  $D_5 = 0$ ，而小写字母  $D_5 = 1$ 。

```
. model small
. stack
. data
string db 'HeLLo,eveRyboDy ! ',0 ;可以任意给定一个字符串
. code
. startup
mov bx, offset string
again: mov al, [bx] ;取一个字符
      or al, al ;是否为结尾符 0
      jz done ;是,退出循环
      cmp al, 'A' ;是否为大写字母 A~Z
      jb next
      cmp al, 'Z'
      ja next
      or al, 20h ;是,转换为小写字母(使  $D_5 = 1$ )

      mov [bx], al ;仍保存在原位置
next: inc bx
      jmp again ;继续循环
done: . exit 0
end
```

答:

change macro string

    mov bx , offset string

again:

    mov al , [bx]

    or al , al

    jz done

    cmp al , 'A'

    jb next

    cmp al , 'Z'

    ja next

    or al , 20h

    mov [bx] , al

next:

    inc bx

    jmp again

done:

    endm