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。