

《汇编语言》作业（三）

参考答案

简答题

(1) 概念辨析（进行概念上的辨析，并举例说明）

a. 变量和标号

变量：变量定义在数据段、堆栈段以及附加段中，表示逻辑段中的存储单元。变量具有三个属性，分别是段属性、偏移属性以及类型属性。其中段属性表示存储单元所在逻辑段的段基址，偏移属性表示存储单元相对于段首单元的偏移地址，类型属性表示变量的数据类型，如字节型、字型、双字型等等。

举例：

```
DATAS SEGMENT
    DA DB 12,34,56,78
DATAS ENDS
```

该段代码描述数据段中的变量 DA 的定义。DA 的段属性是 DS 寄存器中的值，假定其偏移地址为 0000H，则偏移属性为 0000H，类型属性为字节型。

标号：标号定义在代码段中，通常作为指令的转移目标。直接定义的标号后面必须加上冒号。也可以使用伪指令 LABEL 以及 EQU 定义标号。标号具有与变量相同的三个属性，分别为段属性、偏移属性、类型属性。段属性表示标号所在的代码段的段基址，该段基址存放在代码段寄存器 CS 中。偏移属性表示在代码段中，标号所在位置相对于段首的偏移地址。类型属性指出是段内转移的目标地址还是段间转移的目标地址，前者类型属性用 NEAR 表示，后者类型属性用 FAR 表示。类型属性不写出时默认为 NEAR。

举例：

```
CMP AX,0
JZ NEXT
...
```

NEXT:

....

其中 NEXT 为标号，其段属性即为 CS 寄存器内容，偏移属性则为 NEXT 相对于段首的偏移地址，类型属性为 NEAR。

b. 数值表达式和地址表达式

数值表达式：数值表达式一般是指由运算符连接的各种常数所构成的表达式，其在汇编过程中进行计算，最终得到一个数值。

举例：MOV AH, (12*3+4)/10 ;此条指令源操作数即为一个数值表达式

地址表达式：地址表达式在汇编后的结果是变量或者标号所在逻辑段的偏移地址值。

举例：MOV AH, [BX+1] ;此条指令中的[BX+1]就是一个地址表达式

c. 符号常量和字符串常量

符号常量：符号常量是使用符号定义伪指令 EQU 或 = 定义的常数。将程序中经常使用到的数值定义为符号常数可以让源程序简洁，符号常数在汇编后被替换为原有的常数数值。

举例：NUM EQU 15 ;NUM 定义为数值为 15 的符号常数

MOV AH, NUM ;该指令在汇编后 NUM 将被替换为 15，等同于指令 MOV AH,15

字符串常量：字符串常量是用单引号括起来的一个或多个字符，在汇编后，字符串常量中的

字符被转换为对应的 ASCII 码。

举例：DA DB 'ABCD' ;此处 DA 定义在数据段，DA 即为一个字符串常量。

(2) 假设 myword 是一个字变量，mybyte1 和 mybyte2 是 2 个字节变量，指出下列语句中的错误原因。

● MOV BX, OFFSET myword[SI]

错误原因：OFFSET 为属性操作符，表示将跟着的符号地址的值（不是内容）作为操作数。而该语句中的 myword[SI]相当于[myword+SI]代具体的内存单元，不是符号地址。

b. MOV AL, mybyte1+mybyte2

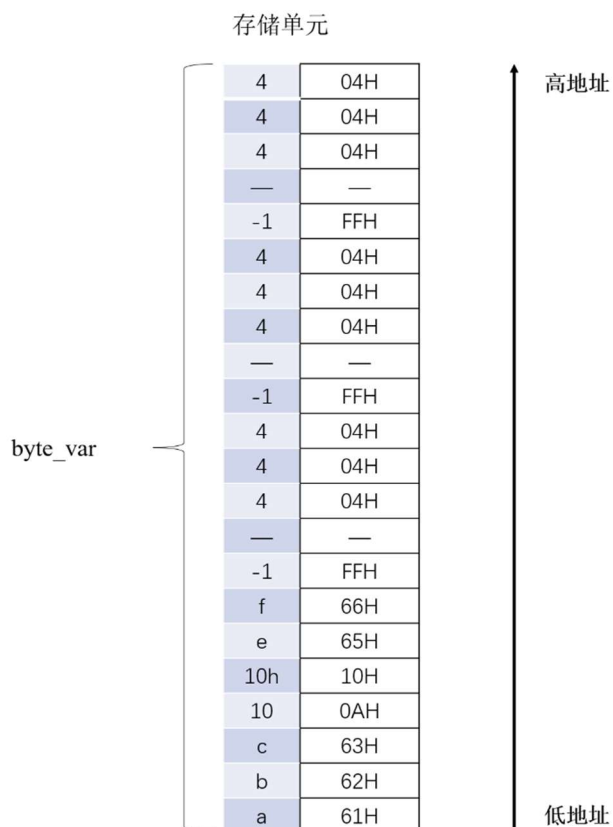
错误原因：数值表达式一般是指由运算符连接的各种常数所构成的表达式，而此处 mybyte1 和 mybyte2 为变量，不能进行运算。

c. JNZ myword

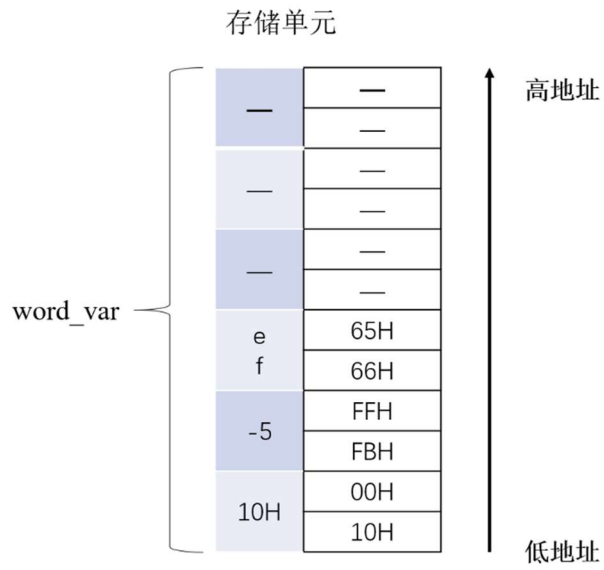
错误原因：条件转移指令的格式统一为 条件转移指令助记符 目标地址标号。而 myword 是变量不是标号。

(3) 画图说明下列语句分配的存储空间以及初始化的数据值：

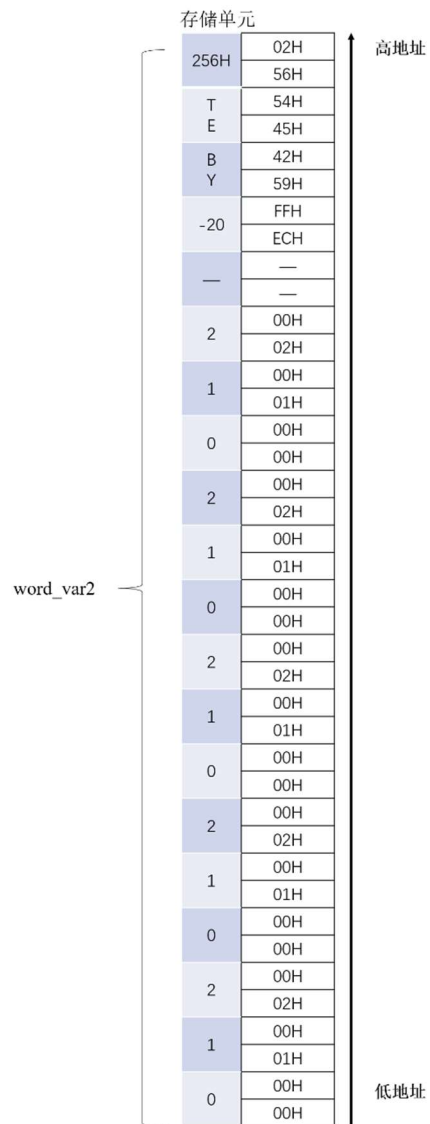
a. byte_var db 'abc',10,10h,'ef',3 dup(-1,?,3 dup(4))



b. word_var dw 10h,-5,'ef',3 dup(?)



c. word_var2 dw 5 dup(0,1,2),?,-20,'BY','TE',256H



(4) 请设置一个数据段 DATASG, 定义如下字符变量或数据变量
数据段 DATASG 定义如下:

DATASG SEGMENT

FLD1B DB 'deep learning is popular',

FLD2B DB 20H

FLD3B DB 33H, 32H, 36H, 35H, 34H ;此处有两种表示, 可以用 ASCII 码, 或者直接
;用字符表示, 如 FLD3B DB '3','2','6','5','4'

FLD4B DB 'PART1', 20

DB 'PART2', 50

DB 'PART3', 14

FLD3W DW OFFSET FLD4B

FLD4W DW \$- OFFSET FLD1B ; MASM 中, 符号"\$"表示当前偏移地址值。

DATASG ENDS

(5) 设程序中的数据定义如下:

LNAME DB 30 DUP(?)

ADDRESS DB 30 DUP(?)

CITY DB 15 DUP(?)

CODELIST DB 1,7,8,8,2

a. 用一条 MOV 指令将 LNAME 的偏移地址放入 AX

指令: MOV AX, OFFSET LNAME

b. 用一条指令将 CODELIST 两个字节的内容放入 SI

指令: MOV SI, WORD PTR CODELIST

c. 写一条伪操作使 CODE_LENGTH 的值等于 CODELIST 域的实际长度

指令: CODE_LENGTH EQU LENGTHOF CODELIST 或 CODE_LENGTH = LENGTHOF
CODELIST

编程题

(1) 编写一个程序, 要求能从键盘接收一个个位数 N, 然后响铃 N 次(响铃的 ASCII 码为 07)
代码如下:

```

01 DATASG SEGMENT
02 ;此处输入堆栈段代码
03 DATASG ENDS
04 |
05 STACKS SEGMENT
06 ;此处输入堆栈段代码
07 STACKS ENDS
08
09 CODES SEGMENT
10 ASSUME CS:CODES,DS:DATASG,SS:STACKS
11 START:
12     MOV AX,DATASG
13     MOV DS,AX
14
15     MOV AH,01H
16     INT 21H
17     SUB AL,30H ;修改成输入对应数值
18     MOV BL,AL
19     MOV AH,02H ;换行
20     MOV DL,0AH
21     INT 21H
22 AGAIN:
23     CMP BL,0 ;判断循环次数是否为0
24     JZ DONE
25     MOV DL,07H ;响铃
26     MOV AH,02H
27     INT 21H
28     MOV DL,'1' ;输出1
29     INT 21H
30     DEC BL
31     JMP AGAIN
32 DONE:
33     MOV AH,4CH
34     INT 21H
35 CODES ENDS
36 END START

```

结果：其中输出 1 表示响铃

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Pr
6
111111
Press any key to continue_

(2) 编程序求出首地址为 DATA 的 10 个数组中的最小偶数，把它存放在 AX 中，并显示出来。

代码如下：

```

01 DATASG SEGMENT
02     DATA DW 12,50,33,45,89,72,6,5,22,18
03 DATASG ENDS
04
05 STACKS SEGMENT
06 ;此处输入堆栈段代码
07 STACKS ENDS
08

```

```

08
09 CODES SEGMENT
10     ASSUME CS:CODES,DS:DATASG,SS:STACKS
11 START:
12     MOV AX,DATASG
13     MOV DS,AX
14
15     MOV BX,OFFSET DATA
16     MOV CX,20 ;按字节设置数组长度,控制循环
17     MOV SI,0
18     MOV AX,7FFFH
19 AGAIN:
20     CMP SI,CX ;若数组长度为0,结束
21     JZ EXIT1
22     TEST WORD PTR [BX+SI],0001H
23     JNZ NEXT ;2标为1,表示当前元素为奇数
24     CMP AX,WORD PTR [BX+SI]
25     JLE NEXT
26     MOV AX,WORD PTR [BX+SI]
27 NEXT:
28     ADD SI,2
29     JMP AGAIN

```

```

30
31 EXIT1:
32     CMP AX,7FFFH
33     JZ EXIT2
34     ;输出AX内容
35     MOV BH,10
36     DIV BH
37     ADD AX,3030H
38     MOV DX,AX
39     MOV AH,02H
40     INT 21H
41     MOV DL,DH
42     INT 21H
43 EXIT2:
44     MOV AH,4CH
45     INT 21H
46 CODES ENDS
47 END START

```

运行结果:

```

DOS
BOX
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip
36
Press any key to continue_

```

(3) 已知数组 A 包含 15 个互不相等的整数, 数组 B 包含 20 个互不相等的整数, 试编程把既在 A 中又在 B 中出现的整数存放于数组 C 中。

代码如下:

```

01 DATASG SEGMENT
02     DATA1 DW 1,2,3,4,5
03             DW 6,7,8,9,10
04             DW 11,12,13,14,15
05     DATA2 DW 5,4,3,2,1
06             DW 16,17,18,19,20
07             DW 21,22,23,24,25
08             DW 11,13,15,17,19
09     RESULT DW 15 DUP(?)
10 DATASG ENDS
11
12 STACKS SEGMENT
13     ;此处输入堆栈段代码
14 STACKS ENDS


```

```

15
16 CODES SEGMENT
17     ASSUME CS:CODES,DS:DATASG,SS:STACKS
18 START:
19     MOV AX,DATASG
20     MOV DS,AX
21
22     MOV BX,OFFSET DATA1
23     DIS1 EQU OFFSET DATA2 - OFFSET DATA1 ;用于计算DATA2的偏移地址
24     DIS2 EQU OFFSET RESULT - OFFSET DATA1 ;用于计算RESULT的偏移地址
25     MOV SI,0
26     MOV CX,0
27
28 OUT_LOOP:
29     CMP SI,15*2 ;控制对DATA1的遍历
30     JZ EXIT ;遍历完成跳至EXIT, 结束程序
31     MOV AX,WORD PTR [BX+SI] ;将当前DATA1中待判断元素存入AX
32     MOV DI,0 ;控制对DATA2的遍历
33 INNER_LOOP:
34     CMP DI,20*2
35     JZ END_INNER_LOOP ;DATA2遍历完成, 跳至END_INNER_LOOP
36     CMP AX,WORD PTR [BX+DI+DIS1] ;比较DATA1与DATA2中字符
37     JZ EQULE ;相等跳至EQULE进行结果存储操作
38     ADD DI,2 ;因为定义为DW, 所以加2
39     JMP INNER_LOOP
40
41 EQULE: ;存储相同结果
42     MOV DX,WORD PTR [BX+DI+DIS1]
43     MOV DI,CX ;CX存储RESULT中元素个数, 根据CX值将新值存入RESULT中
44     ADD DI,DI
45     MOV WORD PTR [BX+DI+DIS2],DX
46     INC CX
47
48 END_INNER_LOOP:
49     ADD SI,2
50     JMP OUT_LOOP
51
52
53 EXIT:
54     MOV AX,CX
55     ;输出AX内容
56     MOV BH,10
57     DIV BH
58     ADD AX,3030H
59     MOV DX,AX
60     MOV AH,02H
61     INT 21H
62     MOV DL,DH
63     INT 21H
64
65     MOV AH,4CH
66     INT 21H
67 CODES ENDS
68     END START

```

运行结果：输出相同数字个数

 DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip

08

Press any key to continue