# 

32位汇编语言程序设计

郑州大学 软件工程

2011级 汇编语言程序设计 课程试题（A卷）

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 题号 | 一 | 二 | 三 | 四 | 五 | 六 | 七 | 总分 |
| 分数 |  |  |  |  |  |  |  |  |

合分人： 复查人：

|  |  |  |
| --- | --- | --- |
| 一、填空题（每空1分，共15分） | 分数 | 评卷人 |
|  |  |

1．用汇编语言书写的程序叫做 ，MASM要求该文件的扩展名是 ，编写32位Windows应用程序应选择 存储模型。

2．汇编语言语句有两种，分别是 和 。3．除外设数据外的数据寻址方式有3类，分别称为 、

和 寻址。

4．进行Windows应用程序开发时，需要 库文件，执行该应用程序时，则需要对应的 库文件。

5．汇编语言与C和C++语言有两种混合编程方法，分别是 和 。

6．IA-32处理器有 个32位通用寄存器，有 个段寄存器，段寄存器都是 位的。

|  |  |  |
| --- | --- | --- |
| 二、判断题（每题1分，共10分） | 分数 | 评卷人 |
|  |  |

1．Windows的模拟DOS环境与控制台环境是一样的。 （ ）

2．“TYPE DX”的结果是一个常量，等于2。 （ ）

3．数据扩展的基本原则是位数加长，数值大小不变。 （ ）

4．指令的相对寻址都是近转移。 （ ）

5．JA和JG指令的条件都是“大于”，所以是同一个指令的两个助记符。 （ ）

6．CALL指令的执行必然影响堆栈指针ESP。 （ ）

7．在宏定义中，LOCAL伪指令声明标识符；而在过程定义中，LOCAL伪指令用于分配局部变量。 （ ）

8．指令“OUT DX，AX”的两个操作数均采用寄存器寻址方式。 （ ）

9．MASM32软件包既支持32位图形界面应用程序的开发，也支持控制台应用程序开发。 （ ）

10．程序终止执行也就意味着汇编结束，所以两者含义相同。 （ ）

|  |  |  |
| --- | --- | --- |
| 三、按下面要求写出相应的数据定义语句或汇编指令（变量名可任意指定）（每小题2分，共12分） | 分数 | 评卷人 |
|  |  |

1. 定义字符串变量：assembly language
2. 定义20个未赋值的字变量
3. 定义符号常量str，表示字符串assembly language
4. 定义用十六进制数表示的双字变量：20
5. 用寄存器EBX和ESI的基址变址寻址方式把存储器的一个双字与EAX寄存器的内容相加，并把结果送回存储器中。用一条汇编指令完成该功能。
6. 用一条汇编指令实现将寄存器EAX和标志位CF、OF同时清零。

|  |  |  |
| --- | --- | --- |
| 四、程序分析与阅读题（共 28分） | 分数 | 评卷人 |
|  |  |

1．根据数据段定义的变量，分析下列指令的运行结果并填在空格里（每空1分，共8分）

.data

array qword 1234567887654321h

.code

mov eax，dword ptr array ；EAX= \_\_\_\_\_\_\_

mov ebx，dword ptr array+4 ；EBX= \_\_\_\_\_\_\_

sub al，bl ；EAX= \_\_\_\_\_\_\_，CF= \_\_\_\_\_\_\_，OF= \_\_\_\_\_\_\_

；ZF= \_\_\_\_\_\_\_，SF= \_\_\_\_\_\_\_ PF= \_\_\_\_\_\_\_

2．指出如下语句的错误，并改正（其中var1和var2是两个双字变量）(每小题2分，共8分)

① add eax，bx

② mov var1， var2

③ push al

④ mov [ebx]，100

3．有如下程序片断，请使用一条指令实现该程序段所完成的功能(每小题2分，共4分)

① mov eax，ebx

mov ebx，[edi]

mov [edi]，eax

\_\_\_\_\_\_\_\_\_\_

② mov al，[esi]

mov [edi]，al

inc esi

inc edi

\_\_\_\_\_\_\_\_\_\_

4．如下程序段实现在一个已知长度（长度为count）的串string中查找是否存在子串bug的功能，如果找到则显示“Y”，否则显示“N”。请在空白处填上恰当的语句。（每空2分，共8分）

mov ecx，count

mov edi，offset string

L1: mov esi，offset bug

push edi

mov edx，sizeof bug

LN: mov al，[esi]

\_\_\_\_\_\_\_\_\_\_

jne L2

inc esi

inc edi

\_\_\_\_\_\_\_\_\_\_

jne LN

pop edi

mov al，'Y'

\_\_\_\_\_\_\_\_\_\_

L2: pop edi

\_\_\_\_\_\_\_\_\_\_

loop L1

mov al，'N'

L3: call dispc

|  |  |  |
| --- | --- | --- |
| 五、编程题：（共35分） | 分数 | 评卷人 |
|  |  |

1．（5分）将调用字符串输入子程序readmsg编写成一个宏readstring，其中宏的参数是输入缓冲区的名称msg，实际输入字符的个数存入ecx寄存器中。

|  |  |
| --- | --- |
| 子程序名 | 参数及功能说明 |
| READMSG | 入口参数：EAX=缓冲区地址 功能说明：输入一个字符串（回车结束）  出口参数：EAX=实际输入的字符个数（不含结尾字符0） |

2．（8分）编程直接调用Windows控制台API函数显示“Hello，Assembly! ”。控制台API函数位于动态连接库KERNEL32.DLL中，标准输出句柄值：-11，常用函数说明如下：

VOID ExitProcess(

DWORD uExitCode // exit code for all threads

)；

HANDLE GetStdHandle(

DWORD nStdHandle //input,output,or error device

)；

WriteConsole参数

第一个：控制台输出句柄实例

第二个：指向字符串的指针、即缓冲区地址

第三个：指明字符串长度

第四个：指向一个整数变量，返回实际输出的字符数

第五个：保留，设置为0

3．（10分）使用DOS系统功能编写DOS应用程序，要求如下：提示“Input number:0~9”，等待用户输入字符。如果用户按下数字字符则退出，如果用户按下非数字字符则继续等待输入字符。要求写出完整的源程序代码，不要使用输入输出子程序。

DOS基本功能调用表如下所示：

|  |  |  |  |
| --- | --- | --- | --- |
| 子功能号 | 功能 | 入口参数 | 出口参数 |
| AH=01H | 从标准输入设备输入一个字符 |  | AL=输入字符的ASCII码 |
| AH=09H | 向标准输出设备输出一个字符串 | DX=字符串地址 |  |
| AH=4CH | 程序执行终止 | AL=返回代码 |  |

4．（12分）请用子程序实现将字符串“Are you sure you can pass in the Assembly exam?”中的英文字母进行大小写互换，即如果是大写则转换成小写，如果是小写则转换成大写。对字符串中的非英文字母不做任何处理。在主程序中显示出转换后的结果，要求用主程序对子程序调用完成，利用堆栈传递入口参数，写出完整的主程序、子程序代码，并加上必要的注释。

2011级 汇编语言程序设计 课程试题（A卷答案）

一、填空题（毎空1分，共15分）

1．汇编语言程序/汇编语言源程序，ASM，平展（flat）

2．执行性语句（硬指令），说明性语句（伪指令）

3．立即数，寄存器，存储器

4．导入，动态连接

5．PUSH EBX

6．嵌入汇编，模块连接

7．8，6，16

二、判断题（每题1分，共10分）

1．错 2．对 3．对 4．对 5．错 6．对 7．对 8．错 9．对 10．错

三、按下面要求写出相应的数据定义语句或汇编指令（变量名可任意指定）（每小题2分，共12分）

1. string byte ‘assembly language’
2. var word 20 dup (?)
3. str equ <assembly language>
4. dvar dword 14h
5. add [ebx+esi],eax
6. xor eax,eax 或sub eax,eax

四、程序分析与阅读题

1．（每空1分，共8分）

EAX=87654321h

EBX=12345678h

EAX=876543a9h CF=1 OF=0 ZF=0 SF=1 PF=1

2．(每小题2分，共8分)每一小题指出错误得1分，写出正确指令得1分

① 操作数类型不一致 add eax，ebx 或add ax，bx

② 两个操作数同时是内存单元的数 mov eax，var2 mov var1，eax

③ push指令不支持字节操作 push ax 或 push eax

④ 两个操作数类型不明确 mov byte ptr [ebx]，100

或mov word ptr [ebx]，100

或mov dword ptr [ebx]，100

3．(每小题2分，共4分)

① xchg ebx，[edi] 或 xchg [edi]，ebx

② movsb

4．（每空2分，共8分）

cmp al，[edi]

dec edx

jmp L3

inc edi

五、编程题（共35分）

1．（5分）

readstring macro，：msg

lea eax，msg

call readmsg

mov ecx，eax

endm

2．（8分）

.686

.model flat,stdcall

option casemap:none

includelib bin\kernel32.lib

ExitProcess proto,:dword

GetStdHandle proto,:dword

WriteConsoleA proto,:dword,:

dword,:dword,:dword,:dword

WriteConsole equ <WriteConsoleA>

STD\_OUTPUT\_HANDLE = -11

.data

msg byte 'Hello, Assembly!'，13，10

outsize dword ?

.code

start:

invoke GetStdHandle，

STD\_OUTPUT\_HANDLE ;获得输出句柄

invoke WriteConsole，eax，

addr msg，sizeof msg，addr outsize，0 ;

invoke ExitProcess，0 ;退出

end start

3．（10分）

.model small

.686

.stack

.data

msg byte ‘Input number:0~9’，’$’

.code

start:

mov ax，@data

mov ds，ax

mov ah，9

mov dx，offset msg

int 21h

again:mov ah，1

int 21h

cmp al，30h

jb next

cmp al，39h

jbe done

next:jmp again

done:mov ax，4c00h

int 21h

end start

4． （12分）

include io32.inc

.data

msg byte ‘Are you sure you can pass in

the Assembly exam?’，13，10，0

.code

start:

mov eax，sizeof msg

push eax

mov eax，offset msg

push eax

call zhuanhuan

add esp，8

mov eax，offset msg

call dispmsg

exit 0

zhuanhuan proc

push ebp

mov ebp，esp

push ebx

push ecx

mov ebx，[ebp+8]

mov ecx，[ebp+12]

again:mov al，[ebx]

cmp al，’A’

jb next

cmp al，’Z’

jbe next1

cmp al，’a’

jb next

cmp al，’z’

ja next

next1:xor al，20h

mov [ebx]，al

next:inc ebx

loop again

pop ecx

pop ebx

mov esp，ebp

pop ebp

ret

zhuanhuan endp

end start

# 32位汇编语言程序设计1.简答

**第一章**

**1.1 简答题-1**

**（1）哪个处理器的指令系统成为Intel 80x86系列处理器的基本指令集？**

**• 8086**

**（3）什么是通用寄存器？**

**• 一般指处理器最常使用的整数寄存器，可用于保存整数数据、地址等**

**（6）最高有效位MSB是指哪一位？**

**• 数据的最高位，例如对8、16、32位数据，MSB依次指D7、D15和D31位**

**1.1 简答题-2**

**（7）汇编语言中的标识符与高级语言的变量和常量名的组成原则有本质的区别吗？**

**• 没有**

**（8）汇编语言的标识符大小写不敏感意味着什么？**

**• 表示字母大小写不同、但表示同一个符号**

**1.2 判断题-1**

**1）EAX也被称为累加器，因为它使用最频繁。**

**• 对，EAX中的A来自英文累加器（Accumulator）**

**2）指令指针EIP寄存器属于通用寄存器。**

**• 错，EIP只用于指示指令地址，不做它用，属于专用寄存器。**

**3）IA-32处理器在实地址方式下，不能使用32位寄存器。**

**• 错，实地址方式可以使用32位的寄存器、操作数和寻址方式。**

**4）保护方式下，段基地址加偏移地址就是线性地址或物理地址。**

**• 对。**

**1.2 判断题-2**

**5）Windows的模拟DOS环境与控制台环境是一样的。**

**• 错，外观和基本操作基本相同，但实质完全不同。**

**6）处理器的传送指令MOV属于汇编语言的执行性语句**

**• 对**

**7）汇编语言的语句由明显的4部分组成，不需要分隔符区别。**

**• 错，使用了分隔符才有了明显的4部分**

**8）MASM汇编语言的注释用分号开始，不能用中文分号**

**• 对，源程序中的分隔符以及各种标识符都是英文**

**1.2 判断题-3**

**9）程序终止执行也就意味着汇编结束，所以两者含义相同。**

**• 错，两者完全是两个概念**

**10）源程序文件和列表文件都是文本性质的文件。**

**• 对，文本文件一般是只包含字符内容的文件，可以使用Windows的记事本打开。**

**1.3 填空题-1**

**（1）Intel 8086支持\_\_\_容量主存空间，IA-32处理器支持\_\_\_容量主存空间。**

**• 1MB，4GB**

**（2）英特尔\_\_\_处理器将80x86指令系统升级为32位指令系统，\_\_\_处理器内部集成浮点处理单元、开始支持浮点操作指令。**

**• 80386,80486**

**（3）IA-32处理器有8个32位通用寄存器，其中EAX，\_\_\_，\_\_\_和EDX，可以分成16位和8位操作；还有另外4个是\_\_\_，\_\_\_，\_\_\_，和\_\_\_。**

**• EBX，ECX，ESI，EDI，EBP，ESP**

**1.3 填空题-2**

**（4）寄存器EDX是\_\_\_位的，其中低16位的名称是\_\_\_\_，还可以分成两个8位的寄存器，其中D0～D7和D8～D15部分可以分别用名称\_\_\_\_和\_\_\_\_表示。**

**• 32，DX，DL，DH**

**（5）IA-32处理器有\_\_\_个段寄存器，它们都是\_\_\_\_\_位的。**

**• 6，16**

**（6）IA-32处理器复位后，首先进入是\_\_\_\_工作方式。该工作方式分段最大不超过\_\_\_\_。**

**• 实地址，64KB**

**1.3 填空题-3**

**（7）逻辑地址由\_\_\_和\_\_\_两部分组成。代码段中下一条要执行的指令由CS和\_\_\_寄存器指示，后者在实地址模型中起作用的仅有\_\_\_寄存器部分。**

**• 段基地址，偏移地址，EIP，IP**

**（8）Windows的文件夹对应的专业术语是\_\_\_\_\_。**

**• 目录**

**（9）指令由表示指令功能的\_\_\_\_和表示操作对象的\_\_\_部分组成，IA-32处理器的指令前缀属于\_\_\_\_部分。**

**• 操作码，操作数（地址码），操作码**

**（10）MASM要求汇编语言源程序文件的扩展名是\_\_\_，汇编产生扩展名为OBJ的文件被称为\_\_\_文件，编写32位Windows应用程序应选择\_\_\_存储模型。**

**• ASM，目标模块，FLAT**

**习题1.9**

**• IA-32处理器有哪三类基本段，各是什么用途？**

**• 解答：**

**代码段：存放程序的指令代码**

**数据段：存放当前运行程序所用的数据**

**堆栈段：主存中堆栈所在的区域**

**习题1.13**

**• 汇编语言语句有哪两种，每种语句由哪4个部分组成**

**• 解答：**

**汇编语句有两种**

**执行性语句（处理器指令）**

**说明性语句（伪指令）**

**每个语句由4部分组成**

**标号**

**指令助记符**

**操作数或参数**

**注释**

**习题1.15**

**• MASM汇编语言中，下面哪些是程序员可以使用的自定义标识符。**

**FFH，DS，Again，next，@data，h\_ascii，6364b，flat**

**• 解答：**

**FFH，Again，next，h\_ascii**

**其他符号错误的原因：**

** DS是数据段寄存器名**

** 6364b是数字开头**

** @data是汇编语言使用预定义符号，表示数据段地址**

** flat是表示平展存储模型的关键字**

**习题1.16**

**• 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：**

**汇编语言程序的开发有4个步骤：**

** 编辑：用文本编辑器形成一个以ASM为扩展名的源程序文件。**

** 汇编：用汇编程序将ASM文件转换为OBJ模块文件。**

** 连接：用连接程序将一个或多个目标文件链接成一个EXE或COM可执行文件。**

** 调试：用调试程序排除错误，生成正确的可执行文件。**

**第2章 数据表示和寻址**

**2.1 简答题**

**（2）字符“'F'”和数值46H作为MOV指令的源操作数有区别吗？**

**• 没有，因为字符“'F'”的ASCII码就是数值46H**

**（3）为什么可以把指令“MOV EAX, (34＋67H)\*3”中的数值表达式看成是常量？**

**• 汇编程序在汇编过程中对数值表达式计算，得到一个确定的数值，故称数值表达式为常量**

**（4）汇编语言为什么规定十六进制数若以A～F开头，需要在前面加个0？**

**• 以便与标识符区别，因为标识符要求以字母（或特殊符号）开头**

**2.1 简答题-2**

**（7）为什么将查找操作数的方法称为数据寻“址”方式？**

**• 指令的操作数需要通过存储器地址或I/O地址，才能查找到数据本身，故称数据寻址方式**

**（8）为什么变量VAR在指令“MOV EAX, VAR”中表达直接寻址？**

**• 因为这里的变量名VAR实际上表达了变量所存放的主存单元，即[地址]**

**2.2 判断题-1**

**（1）对一个正整数，它的原码、反码和补码都一样，也都与无符号数的编码一样。**

**• 对**

**（2）常用的BCD码为8421 BCD码，其中的8表示D3位的权重。**

**• 对**

**（3）排序一般按照ASCII码值大小，从小到大升序排列时，小写字母排在大写字母之前。**

**• 错。小写字母的ASCII值大于大写字母，升序排列应该排在后。**

**2.2 判断题-2**

**（4）用“BYTE”和“DWORD”定义变量，如果初值相同，则占用的存储空间也一样多。**

**• 错。用BYTE只占一个存储单元，而DWORD要占4个存储单元**

**（5）“TYPE DX”的结果是一个常量，等于2。**

**• 对。TYPE后跟寄存器名得到寄存器数据的字节数。**

**（6）IA-32处理器采用小端方式存储多字节数据。**

**• 对。**

**2.2 判断题-3**

**（7）某个双字变量存放于存储器地址0403H～0406H中，对齐了地址边界。**

**• 错，没有对齐，因为双字变量对齐地址边界是起始于模4地址，即4×N地址。**

**（8）立即数寻址只会出现在源操作数中。**

**• 对**

**（9）存储器寻址方式的操作数当然在主存了。**

**• 对**

**（10）指令“MOV EAX, VAR+2”与 “MOV EAX, VAR[2]”功能相同。**

**• 对，仅是不同的形式罢了**

**2.3 填空题-1**

**（1）计算机中有一个“01100001”编码。如果把它认为是无符号数，它是十进制数\_\_\_\_；如果认为它是BCD码，则表示真值\_\_\_\_\_；又如果它是某个ASCII码，则代表字符\_\_\_\_。**

**• 97，61，小写字母a**

**（2）C语言用“\n”表示让光标回到下一行首位，在汇编语言中需要输出两个控制字符：一个是回车，其ASCII码是\_\_\_\_\_，它将光标移动到当前所在行的首位；另一个是换行，其ASCII码是\_\_\_\_，它将光标移到下一行。**

**• 0DH（13），0AH（10）**

**2.3 填空题-2**

**（3）定义字节变量的伪指令助记符是\_\_\_\_\_，获取变量名所具有的偏移地址的操作符是\_\_\_\_。**

**• BYTE，OFFSET**

**（4）数据段有语句“H8843 DWORD 99008843H”，代码段指令“MOV CX, WORD PTR H8843”执行后，CX＝\_\_\_\_\_\_\_。**

**• 8843H**

**（5）用DWORD定义的一个变量XYZ，它的类型是\_\_\_\_，用“TYPE XYZ”会得到数值为\_\_\_。如果将其以字量使用，应该用\_\_\_\_\_说明。**

**• DWORD，4，WORD PTR XYZ**

**（6）数据段有语句“ABC BYTE 1,2,3”，代码段指令“MOV CL, ABC+2”执行后，CL＝\_\_\_\_\_。**

**• 3**

**（7）除外设数据外的数据寻址方式有3类，分别被称为\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_和\_\_\_\_\_\_\_。**

**• 立即数寻址，寄存器寻址，存储器寻址**

**（8）指令“MOV EAX, OFFSET MSG”的目的操作数和源操作数分别采用\_\_\_\_\_\_和\_\_\_\_\_\_\_寻址方式。**

**• 寄存器寻址，立即数寻址**

**（9）已知ESI＝04000H，EBX＝20H，指令“MOV EAX, [ESI+EBX\*2+8]”中访问的有效地址是\_\_\_\_\_\_\_。**

**• 04000H＋20H×2＋8＝04048H**

**（10）用EBX做基地址指令，默认采用\_\_\_\_段寄存器指向的数据段；如果采用BP，EBP或SP，ESP作为基地址指针，默认使用\_\_\_\_\_段寄存器指向堆栈段。**

**• DS，SS**

**习题2.10-1**

**• 按照如下要求定义变量或符号常量：**

**（1） my1b为字符串变量：Personal Computer**

**• my1b byte 'Personal Computer'**

**（2） my2b为用十进制数表示的字节变量：20**

**• my2b byte 20**

**（3） my3b为用十六进制数表示的字节变量：20**

**• my3b byte 14h**

**（4） my4b为用二进制数表示的字节变量：20**

**• my4b byte 00010100b**

**习题2.10-2**

**• 按照如下要求定义变量或符号常量：**

**（5） my5w为20个未赋值的字变量**

**• my5w word 20 dup(?)**

**（6） my6c为100的常量**

**• my6c = 100**

**（7） my7c表示字符串：Personal Computer**

**• my7c equ <Personal Computer>**

**习题2.14**

**• 按照如下输出格式，在屏幕上显示ASCII表**

**• 编程在数据段直接构造这样的表格、填写相应ASCII代码值（不是字符本身），然后使用字符串显示子程序DISPMSG实现显示**

**习题2.14解答1**

**.data**

**table byte ' |0 1 2 3 4 5 6 7 8 9 A B C D E F', 13,10**

**byte'---+-------------------------------', 13,10**

**byte '20 |',20h,20h,21h,20h,22h,20h,23h,20h, 24h,20h,25h,20h,26h,20h,27h,20h,28h,20h,29h,20h**

**byte2ah,20h,2bh,20h,2ch,20h,2dh,20h,2eh,20h, 2fh,20h,13,10**

**byte '30 |',30h,20h,31h,20h,32h,20h,33h,20h, 34h,20h,35h,20h,36h,20h,37h,20h,38h,20h,39h,20h**

**byte 3ah,20h,3bh,20h,3ch,20h,3dh,20h,3eh,20h, 3fh,20h,13,10**

**……**

**byte 0**

**习题2.14解答2**

**……**

**ca ='2'**

**cas =20h**

**byte ca,'0 |',cas,20h,cas+1,20h,cas+2,20h, cas+3,20h,cas+4,20h,cas+5,20h,cas+6,20h,cas+7,20hbyte cas+8,20h,cas+9,20h,cas+0ah,20h,cas+0bh,20h, cas+0ch,20h,cas+0dh,20h,cas+0eh,20h,cas+0fh,20h,13,10**

**cas =cas+10h**

**ca =ca+1**

**byte ca,'0 |',cas,20h,cas+1,20h,cas+2,20h, cas+3,20h,cas+4,20h,cas+5,20h,cas+6,20h,cas+7,20h**

**byte cas+8,20h,cas+9,20h,cas+0ah,20h,cas+0bh,20h, cas+0ch,20h,cas+0dh,20h,cas+0eh,20h,cas+0fh,20h,13,10**

**……**

**习题2.15**

**• 数据段有如下定义：**

**var dword 12345678h**

**• 现以字节为单位按地址从低到高的顺序，写出这个变量内容。**

**var变量的内容：78H、56H、34H、12H**

**• 并说明如下指令的执行结果：**

**mov eax,var ;EAX＝12345678H**

**mov bx,word ptr var ;BX＝5678H**

**mov cx,word ptr var+2 ;CX＝1234H**

**mov dl,byte ptr var ;DL＝78H**

**mov dh,byte ptr var+3 ;DH＝12H**

**习题2.17，说明源操作数的寻址方式**

**（1）mov edx,1234h 立即数**

**（2）mov edx,vard 直接**

**（3）mov edx,ebx 寄存器**

**（4）mov edx,[ebx] 寄存器间接**

**（5）mov edx,[ebx+1234h] 寄存器相对**

**（6）mov edx,vard[ebx] 寄存器相对**

**（7）mov edx,[ebx+edi] 基址变址**

**（8）mov edx,[ebx+edi+1234h] 相对基址变址**

**（9）mov edx,vard[esi+edi] 相对基址变址**

**（10）mov edx,[ebp\*4] 带比例寻址**

**第3章 通用数据处理指令**

**3.1 简答题-1**

**（1）如何修改“MOV ESI, WORD PTR 250”语句使其正确？**

**• 删除“WORD PTR”即可**

**（2）为什么说“XCHG EDX,CX”是一条错误的指令？**

**• 源、目标寄存器位数不同，不能用该指令进行数据交换**

**（5）执行了一条加法指令后，发现ZF＝1，说明结果是什么？**

**• 表明运算结果为0**

**3.1 简答题-2**

**（7）大小写字母转换使用了什么规律？**

**• 大小写字母转换利用它们的ASCII码相差20H**

**（9）除法指令“DIV ESI”的被除数是什么？**

**• 被除数64位，在EDX和EAX中，EDX保存高32位、EAX保存低32位**

**3.2 判断题-1**

**（1）指令“MOV EAX,0”使EAX结果为0，所以标志ZF＝1。**

**• 错，MOV指令不影响标志**

**（2）空操作NOP指令其实根本没有指令。**

**• 错，存在该指令，实际上是“XCHG EAX，EAX”指令。**

**（3）堆栈的操作原则是“先进后出”，所以堆栈段的数据除PUSH和POP指令外，不允许其他方法读写。**

**• 错，可以像主存操作数那样随机读写。**

**3.2 判断题-2**

**（4）虽然ADD指令和SUB指令执行后会影响标志状态，但执行前的标志并不影响它们的执行结果。**

**• 对**

**（5）80减90（80－90）需要借位，所以执行结束后，进位标志CF＝1。**

**• 对，减法时借位也用CF标志反映**

**（6）指令“INC ECX”和“ADD ECX,1”的实现功能完全一样，可以互相替换。**

**• 错，不是完全一样，不同之处是对CF标志，INC不影响、ADD影响。**

**3.2 判断题-3**

**（7）无符号数在前面加零扩展，数值不变；有符号数前面进行符号扩展，位数加长一位、数值增加一倍。**

**• 错，符号扩展也不改变数值大小**

**（8）CMP指令是目的操作数减去源操作数，与SUB指令功能相同。**

**• 错，虽然都是做减法，但CMP不改变目的操作数，而SUB要改变。**

**3.2 判断题-4**

**（9）逻辑运算没有进位或溢出问题，此时CF和OF没有作用，所以逻辑运算指令如AND、OR等将CF和OF设置为0。**

**• 对。**

**（10）SHL指令左移一位，就是乘10。**

**• 错，左移一位，相当于乘2。这里的位是二进制比特位**

**3.3 填空题**

**（1）指令“PUSH DS”执行后，ESP会\_\_\_\_。**

**• 减2。**

**（2）指令“POP EDX”的功能也可以用MOV和ADD指令实现，依次应该是\_\_\_\_\_\_\_\_\_\_和\_\_\_\_\_\_\_\_\_\_指令。**

**• MOV EDX,[ESP]，**

**• ADD ESP,4**

**（3）例3-3的TAB定义如果是“1234567890”，则显示结果是\_\_\_\_\_\_\_\_\_\_。**

**• 78894111**

**（4）进行8位二进制数加法：BAH＋6CH，8位结果是\_\_\_\_\_，标志PF＝\_\_\_\_。如果进行16位二进制数加法：45BAH＋786CH，16位结果是\_\_\_\_\_，标志PF＝\_\_\_\_\_。**

**• 26H，0，BE26H，0**

**（5）已知AX＝98H，执行“NEG AX”指令后，AX＝\_\_\_\_，标志SF＝\_\_\_。**

**• FF68H，1**

**（6）假设CL＝98H，执行“MOVZX DX,CL”后，DX＝\_\_\_\_\_\_，这称为\_\_\_\_\_扩展。**

**• 0098H，零位**

**（7）假设CL＝98H，执行“MOVSX DX,CL”后，DX＝\_\_\_\_\_\_\_\_\_\_，这称为\_\_\_\_\_\_\_\_\_\_扩展。**

**• FF98H，符号**

**（8）指令“XOR EAX, EAX”和“SUB EAX, EAX”执行后，EAX＝\_\_\_，CF＝OF＝\_\_\_。而指令“MOV EAX, 0”执行后，EAX＝\_\_\_，CF和OF没有变化。**

**• 0，0，0**

**（9）例3-9程序执行结束，变量QVAR内容是\_\_\_\_\_\_\_\_\_。**

**• 0123456788765432H**

**（10）欲将EDX内的无符号数除以16，使用指令“SHR EDX, \_\_\_\_”，其中后一个操作数是一个立即数。**

**• 4**

**习题3.4**

**• 请给每种组合各举一个实例**

**（1）mov reg, imm**

**（2）mov mem, imm**

**（3）mov reg, reg**

**（4）mov mem, reg**

**（5）mov seg, reg**

**（6）mov reg, mem**

**（7）mov seg, mem**

**（8）mov reg, seg**

**（9）mov mem, seg**

**习题3.5**

**• 操作数的组合通常符合逻辑，但不是任意的，指出如下指令的错误原因。**

**（1） mov ecx,dl**

**；两操作数类型不匹配**

**（2） mov eip,ax**

**；IP指令指针禁止用户访问**

**（3） mov es,1234h**

**；立即数不允许传给段寄存器**

**（4） mov es,ds**

**；段寄存器之间不允许传送**

**习题3.5解答**

**（5） mov al,300**

**；两操作数类型不匹配**

**（6） mov [esi],45h**

**；无操作数类型**

**（7） mov eax,ebx+edi**

**；源操作数应为 [BX+DI]**

**（8） mov 20h,ah**

**；立即数不能作目的操作数**

**习题3.6**

**• 使用MOV指令实现交换指令“XCHG EBX,[EDI]”功能。**

**• 解答：**

**mov eax,ebx**

**mov ebx,[edi]**

**mov [edi],eax**

**习题3.11-1**

**• 执行如下程序片断，说明每条指令的执行结果**

**（1）**

**mov eax,80h ;EAX＝80H**

**add eax,3 ;EAX＝83H，CF＝0，SF＝0**

**add eax,80h ;EAX＝103H，CF＝0，OF＝0**

**adc eax,3 ;EAX＝106H，CF＝0，ZF＝0**

**习题3.11-2**

**• 执行如下程序片断，说明每条指令的执行结果**

**（2）**

**mov eax,100 ;EAX＝100（64H）**

**add ax,200 ;EAX＝300（12CH），CF＝0**

**（3）**

**mov eax,100 ;EAX＝100（64H）**

**add al,200 ;EAX＝44（2CH），CF＝1**

**（包含256的进位含义：256＋44＝300）**

**习题3.11-3**

**• 执行如下程序片断，说明每条指令的执行结果**

**（4）**

**mov al,7fh ;AL＝7FH**

**sub al,8 ;AL＝77H，CF＝0，SF＝0**

**sub al,80h ;AL＝F7H，CF＝1，OF＝1**

**sbb al,3 ;AL＝F3H，CF＝0，ZF＝0**

**习题3.14-1**

**• 请分别用一条汇编语言指令完成如下功能：**

**（1）把EBX寄存器和EDX寄存器的内容相加，结果存入EDX寄存器。**

**• add edx,ebx**

**（2）用寄存器EBX和ESI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。**

**• add al,[ebx+esi]**

**习题3.14-2**

**• 请分别用一条汇编语言指令完成如下功能：**

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

**• add [ebx+0b2h],ecx**

**（4）将32位变量VARD与数3412H相加，并把结果送回该存储单元中。**

**• add vard,3412h**

**（5）把数0A0H与EAX寄存器的内容相加，并把结果送回EAX中。**

**• add eax,0a0h**

**习题3.16-1**

**• 执行如下程序片断，说明每条指令的执行结果**

**（1）**

**mov esi,10011100b ;ESI＝ 9CH**

**and esi,80h ;ESI＝ 80H**

**or esi,7fh ;ESI＝ FFH**

**xor esi,0feh ;ESI＝ 01H**

**（2）**

**mov eax,1010b ;EAX＝ 1010B（可有前导0）**

**shr eax,2 ;EAX＝ 0010B，CF＝1**

**shl eax,1 ;EAX＝ 0100B，CF＝0**

**and eax,3 ;EAX＝ 0000B，CF＝0**

**习题3.16-2**

**• 执行如下程序片断，说明每条指令的执行结果**

**（3）**

**mov eax,1011b ;EAX＝ 1011B（可有前导0）**

**rol eax,2 ;EAX＝ 101100B，CF＝0**

**rcr eax,1 ;EAX＝ 10110B，CF＝0**

**or eax,3 ;EAX＝ 10111B，CF＝0**

**（4）**

**xor eax,eax ;EAX＝0，CF＝0，OF＝0**

**;ZF＝1，SF＝0，PF＝1**

**习题3.21**

**• 如下程序片段实现EAX乘以某个数X的功能，请判断X等于多少。**

**• 请使用一条乘法指令实现上述功能。**

**mov ecx, eax**

**shl eax, 3**

**lea eax, [eax+eax\*8]**

**sub eax, ecx**

**• 解答：**

**8\*8+8-1＝71**

**imul eax, 71**

**习题3.22**

**• 请使用移位和加减法指令编写一个程序片段计算：EAX×21，假设乘积不超过32位。提示：21＝24＋22＋20。**

**• 解答：**

**mov ebx,eax ;EBX＝EAX×20**

**shl ebx,2 ;EBX＝EAX×22**

**add eax,ebx ;EAX＝EAX×(22＋20)**

**shl ebx,2 ;EBX＝EAX×24**

**add eax,ebx ;EAX＝EAX×(24＋22＋20)**

**第4章 程序结构**

**4.1 简答题**

**（2）数据的直接寻址和指令的直接寻址有什么区别？**

**• 数据的直接寻址给出数据所在的存储器地址，指令的直接寻址给出指令所在的存储器地址。**

**（5）什么是奇偶校验？**

**• 数据通信时，数据的某一位用做传输数据的奇偶校验位，数据中包括校验位在内的“1”的个数恒为奇数，就是奇校验；恒为偶数，就是偶校验**

**（6）助记符JZ和JE为什么表达同一条指令？**

**• 两个数相减，差值为0（JZ）与两个数相等（JE）是一个含义**

**（9）如果循环体的代码量远超过128个字节，还能用LOOP指令实现计数控制循环吗？**

**• 不能，因为LOOP指令的目标地址采用相对短转移，只能在－128～＋127字节之间循环**

**（10）什么是“先循环、后判断”循环结构？**

**• 指先执行循环体，然后判断是否继续循环的结构，这种结构通常至少执行一次循环体**

**4.2 判断题**

**（1）指令指针或者还包括代码段寄存器值的改变将引起程序流程的改变。**

**• 对。**

**（2）指令的相对寻址都是近转移。**

**• 对**

**（3）采用指令的寄存器间接寻址，目标地址来自存储单元。**

**• 错，来自寄存器，它与数据的寄存器寻址类似。**

**（4）JMP指令对应高级语言的GOTO语句，所以不能使用**

**• 错，需要使用。**

**（5）因为条件转移指令Jcc要利用标志作为条件，所以也影响标志。**

**• 错，Jcc指令本身的执行不影响标志。**

**（6）JA和JG指令的条件都是“大于”，所以是同一个指令的两个助记符。**

**• 错，JA针对无符号数，JG针对有符号数。**

**（7）JC和JB的条件都是CF＝1，所以是同一条指令。**

**• 对。无符号小于（JB）必然借位（JC）。**

**（8）控制循环是否结束只能在一次循环结束之后进行。**

**• 错，可以先判断、后循环。**

**（9）介绍LOOP指令时，常说它相当于DEC ECX和JNZ两条指令。但考虑对状态标志的影响，它们有差别。LOOP指令不影响标志，而DEC指令却会影响除CF之外的其他状态标志。**

**• 对。但很多时候往往并不关心此时的标志。**

**（10）若ECX＝0，则LOOP指令和JECX指令都发生转移。**

**• 对。**

**4.3 填空题**

**（1）JMP指令根据目标地址的转移范围和寻址方式，可以分成4种类型：段内转移、\_\_\_\_\_\_\_\_\_\_，段内转移、\_\_\_\_\_\_\_\_\_\_和段间转移、\_\_\_\_\_\_\_\_\_\_，段间转移、\_\_\_\_\_\_\_\_\_\_。**

**• 相对寻址，间接寻址，直接寻址，间接寻址**

**（2）MASM给短转移、近转移和远转移定义的类型名依次是\_\_\_、\_\_\_和\_\_\_。**

**• SHORT，NEAR，FAR**

**（3）假设在平展存储模型下，EBX＝1256H，双字变量TABLE的偏移地址是20A1H，线性地址32F7H处存放3280H，执行指令“JMP EBX” 后EIP＝\_\_\_\_\_\_\_，执行指令“JMP TABLE[EBX]”后EIP＝\_\_\_\_\_\_\_。**

**• 1256H，3280H**

**（4）“CMP EAX,3721H”指令之后是JZ指令，发生转移的条件是EAX＝\_\_\_\_\_\_\_\_，此时ZF＝\_\_\_\_\_\_\_。**

**• 3721H，1**

**（5）执行“SHR EBX,4”指令后，JNC发生转移，说明EBX的D3＝\_\_\_\_\_。**

**• 0**

**（6）在EDX等于0时转移，可以使用指令“CMP EDX,\_\_\_\_”、也可以使用“TEST EDX,\_\_\_\_\_”构成条件，然后使用JE指令实现转移。**

**• 0，EDX**

**（7）循环结构程序一般有三个部分组成，它们是\_\_\_\_\_\_\_\_\_\_，循环体和\_\_\_\_\_\_\_\_\_\_部分。**

**• 循环初始，循环控制**

**（8）JECXZ指令发生转移的条件是\_\_\_\_\_\_\_\_\_\_，LOOP指令不发生转移的条件是\_\_\_\_\_\_\_\_\_\_。**

**• ECX＝0，ECX＝1**

**（9）LOOP指令进行减1计数，实际应用中也常进行加1计数。针对例4-14程序，如果删除其中的LOOP指令，则可以使用指令“CMP \_\_\_, ECX”和“JB \_\_\_\_”替代。**

**• EBX，AGAIN**

**（10）小写字母“e”是英文当中出现频率最高的字母。如果某个英文文档利用例题4-11的异或方法进行简单加密，统计发现密文中字节数据“8FH”最多，你判断该程序采用的字节密码可能是\_\_\_\_\_\_\_\_\_\_。**

**• EAH**

**习题4.4**

**• 已知var1、var2、var3和var4是32位无符号整数，用汇编语言程序片段实现如下C++语句：**

**var4=(var1\*6)/( var2-7)+var3**

**mov eax,var1**

**mov ebx,6**

**mul ebx ;var1\*6**

**mov ebx,var2**

**sub ebx,7 ;var2-7**

**div ebx ;(var1\*6)/( var2-7)**

**add eax,var3 ;(var1\*6)/( var2-7)+var3**

**mov var4,eax**

**习题4.7**

**• 定义COUNT（假设为10）个元素的32位数组，输入元素编号（0～COUNT-1），利用DISPHD子程序输出其地址、利用DISPSID子程序输出其值。**

**;数据段**

**count = 10**

**array dword 0,1,2,3,4,5,6,7,8,9**

**;代码段**

**call readuid ;输入N，小于10**

**push eax**

**lea eax,array[eax\*4] ;EAX＝地址**

**call disphd ;显示地址**

**pop eax**

**mov eax,array[eax\*4] ;EAX＝变量值**

**call dispsid ;显示数值**

**习题4.15-1**

**• IA-32处理器的指令CDQ将EAX符号扩展到EDX。假若没有该指令，编程实现该指令功能。**

**（1）按照符号扩展的含义编程，即：EAX最高位为0，则EDX＝0；EAX最高位为1，则EDX＝FFFFFFFFH。**

**test eax,80000000h ;测试最高位**

**jz next1 ;最高位为0，转移**

**mov edx,0ffffffffh**

**;最高位为1，设置EDX＝FFFFFFFFH**

**jmp done ;跳过另一个分支**

**next1: mov dx,0 ;设置EDX＝0**

**done:**

**习题4.15-2**

**• 编程实现指令CDQ,将EAX符号扩展到EDX**

**（2）使用移位等指令进行优化编程。**

**• 方法1：**

**mov edx,eax**

**sar edx,31**

**• 方法2：**

**rol eax,1**

**rcr edx,1**

**sar edx,31**

**ror eax,1**

**习题4.17-1**

**• 编写一个程序，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。**

**• 方法1：使用字符输入输出子程序**

**;数据段**

**inmsg byte 'Input number(0~9): ',0**

**ermsg byte 0dh,0ah,'Error! Input again: ',0**

**;代码段**

**mov eax,offset inmsg ;提示输入数字**

**call dispmsg**

**习题4.17-2**

**• 方法1：使用字符输入输出子程序**

**again: call readc ;等待按键**

**cmp al,'0' ;数字 < 0？**

**jb erdisp**

**cmp al,'9' ;数字 > 9？**

**ja erdisp**

**call dispcrlf**

**call dispc**

**jmp done**

**习题4.17-3**

**• 方法2：使用数字输入输出子程序**

**again: call readuid ;call readsid**

**cmp eax,0 ;数字 < 0？**

**jb erdisp ;jl erdisp**

**cmp eax,9 ;数字 > 9？**

**ja erdisp ;jg erdisp**

**call dispuid ;call dispsid**

**jmp done**

**erdisp: mov eax,offset ermsg**

**call dispmsg**

**jmp again**

**习题4.27-1**

**• 习题2.14在屏幕上显示ASCII表，现仅在数据段设置表格缓冲区，编程将ASCII代码值填入留出位置的表格，然后调用显示功能实现（需要利用双重循环）。**

**table byte ' |0 1 2 3 4 5 6 7 8 9 A B C D E F',13,10**

**byte '---+-------------------------------',13,10**

**tab1 byte 6 dup(36 dup(?),13,10)**

**byte 0**

**习题4.27-2**

**mov ebx,offset tab1**

**mov edx,'| 02'**

**mov ax,2020h**

**mov esi,6**

**again0: mov [ebx],edx**

**add ebx,4**

**mov ecx,16**

**again1: mov word ptr [ebx],ax**

**add ebx,2**

**inc al**

**loop again1**

**习题4.27-3**

**add ebx,2**

**add edx,1**

**dec esi**

**jnz again0**

**mov eax,offset table**

**call dispmsg**

**习题4.29 素数判断程序-1**

**• （1）采用直接简单的算法：假设输入N，将其逐个除以2～N-1，只要能整除（余数为0）说明不是素数，只有都不能整除才是素数。**

**• 核心程序片段如下：**

**cmp eax,3**

**jbe doney ;小于等于3的都是素数**

**cdq**

**mov esi,eax ;EDI.ESI=N>3**

**mov edi,edx**

**mov ebx,2 ;EBX从2开始**

**习题4.29 素数判断程序-2**

**again: div ebx**

**test edx,edx**

**jz donen**

**;余数为0，能整除，不是素数**

**inc ebx**

**cmp ebx,esi ;EBX到N-1结束**

**mov eax,esi**

**mov edx,edi**

**jb again**

**习题4.29 素数判断程序-3**

**（2）采用只对奇数整除的算法：1、2和3是素数，所有大于3的偶数不是素数，从5开始的数字只要除以从3开始的奇数，只有都不能整除才是素数。**

**• 提示：**

**• 排除4的情况，从3开始除，每次循环加2**

**mov ebx,3**

**……**

**add ebx,2**

**习题4.24**

**• 在一个已知长度的字符串中查找是否包含“BUG”子字符串。如果存在，显示“Y”，否则显示“N”。**

**• 解答：**

**;数据段**

**string byte 'If you find any error in the program, you can DEBUG it.'**

**count = sizeof string**

**bug byte 'BUG'**

**习题4.24解答-1**

**;代码段**

**mov ecx,count**

**mov edi,offset string**

**L1:mov esi,offset bug**

**push edi**

**mov edx,sizeof bug**

**LN:mov al,[esi]**

**cmp [edi],al**

**jne L2**

**inc esi**

**inc edi**

**dec edx**

**jne LN**

**习题4.24解答-2**

**pop edi**

**mov al,'Y'**

**jmp L3**

**L2:pop edi**

**inc edi**

**loop L1**

**mov al,'N'**

**L3:call dispc**

**习题4.26**

**• 已知一个字符串的长度，剔除其中所有的空格字符。请从字符串最后一个字符开始逐个向前判断并进行处理。**

**• 解答：**

**;数据段**

**string byte 'Let us have a try !',0dh,0ah,0**

**;代码段**

**mov ecx,sizeof string**

**cmp ecx,2**

**jb done**

**lea eax,string ;显示处理前的字符串**

**call dispmsg**

**习题4.26解答**

**mov esi,ecx**

**dec esi**

**outlp:cmp string[esi],' ' ;检测是否是空格**

**jnz next ;不是空格继续循环**

**mov edi,esi ;是空格，进入剔除空格分支**

**dec ecx**

**inlp:inc edi**

**mov al,string[edi] ;前移一个位置**

**mov string[edi-1],al**

**cmp edi,ecx**

**jb inlp**

**next:dec esi ;继续进行**

**cmp esi,0**

**jnz outlp ;为0结束**

**lea eax,string ;显示处理后的字符串**

**call dispmsg**

**done:**

**第5章 模块化程序设计**

**5.1 简答题**

**（1）指令“CALL EBX”采用了指令的什么寻址方式？**

**• 寄存器间接寻址**

**（4）参数传递的“传值”和“传址”有什么区别？**

**• 传值是传递参数的拷贝，传址是传递参数的地址**

**（5）子程序采用堆栈传递参数，为什么要特别注意堆栈平衡问题？**

**• 子程序保持堆栈平衡才能保证正确返回**

**• 主程序保持堆栈平衡，才能释放传递参数占用的堆栈空间，否则多次调用该子程序可能致使堆栈溢出**

**（6）INCLUDE语句和INCLUDELIB有什么区别？**

**• INCLUDE语句包含的是文本文件、是源程序文件的一部分；INCLUDELIB语句包含的是子程序库文件**

**（7）什么是子程序库？**

**• 子程序库就是子程序模块的集合，其中存放着各子程序的名称、目标代码以及有关定位信息，便于子程序的管理和调用**

**5.2 判断题**

**（1）过程定义PROC是一条处理器指令。**

**• 错，是伪指令。**

**（2）CALL指令的执行并不影响堆栈指针ESP。**

**• 错，要改变，因为返回地址要压入堆栈**

**（3）CALL指令本身不能包含子程序的参数。**

**• 对。**

**（4）CALL指令用在调用程序中，如果被调用程序中也有CALL指令，说明出现了嵌套。**

**• 对。**

**（5）子程序需要保护寄存器，包括保护传递入口参数和出口参数的通用寄存器。**

**• 错，不能保护传递出口参数的寄存器**

**（6）利用INCLUDE包含的源文件实际上只是源程序的一部分。**

**• 对**

**（7）宏调用与子程序调用一样都要使用CALL指令实现。**

**• 错，宏调用是通过宏展开实现的调用，不用CALL指令**

**（8）宏定义与子程序一样一般书写与主程序之后。**

**• 错，宏需要先定义后调用，一般在源程序开头部分。**

**（9）重复汇编类似宏汇编，需要先定义后调用。**

**• 错，直接在需要位置书写重复汇编的语句。**

**（10）条件汇编并不像条件转移指令那样使用标志作为条件。**

**• 对。**

**5.3 填空题**

**（1）指令“RET i16”的功能相当于“RET”指令和“ADD ESP, \_\_\_\_\_\_\_\_\_\_”组合。**

**• i16**

**（2）例5-1程序中的RET指令，如果用POP EBP指令和JMP EBP指令替换，此时EBP内容是\_\_\_\_\_\_\_\_\_\_。**

**• REPT1标号的地址**

**（3）子程序的参数传递主要有3种，它们是\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_和\_\_\_\_\_\_\_\_\_。**

**• 寄存器，共享变量（公共存储单元），堆栈**

**（4）数值10在计算机内部用二进制“1010”编码表示，用十六进制表达是：\_\_\_\_。如果将该编码加37H，则为\_\_\_\_\_，它是字符\_\_\_\_\_\_的ASCII码值。**

**• A，41H，A**

**（5）利用堆栈传递子程序参数的方法是固定的，例如寻址堆栈段数据的寄存器是\_\_\_\_\_。**

**• EBP**

**（6）MASM汇编语言中，声明一个共用的变量应使用\_\_\_伪指令；而使用外部变量要使用\_\_\_伪指令声明。**

**• PUBLIC，EXTERN**

**（7）过程定义开始是“TEST PROC”语句，则过程定义结束的语句是\_\_\_\_\_\_\_\_。宏定义开始是“DISP MACRO”语句，则宏定义结束的语句是\_\_\_\_\_\_\_\_。**

**• TEST ENDP，ENDM**

**（8）一个宏定义开始语句“WriteChar MACRO CHAR:REQ”，则宏名是\_\_\_\_\_\_\_，参数有\_\_\_\_\_\_\_\_个，并且使用“:REQ”说明该参数\_\_\_\_\_\_\_\_\_\_。**

**• WriteChar，1，在调用时不能缺少**

**（9）实现“BYTE 20 DUP(20H)”语句的功能也可以使用重复汇编，第1个语句是\_\_\_\_\_\_\_\_\_\_，第2个语句是“BYTE 20H”，第3个语句是\_\_\_\_\_\_\_\_\_\_。**

**• REPEAT 20，ENDM**

**（10）条件汇编语句“IF NUM LT 100”中的LT表示\_\_\_\_\_\_\_\_\_\_；该语句需要配合\_\_\_\_\_\_\_\_\_\_语句结束条件汇编。**

**• 小于，ENDIF**

**习题5.7-1**

**• 编写一个源程序，在键盘上按一个键，将其返回的ASCII码值显示出来，如果按下退格键（对应ASCII码是08H）则程序退出。请调用书中的HTOASC子程序**

**;代码段，主程序**

**again: call readc**

**cmp al,08h ;退格键BS=08H**

**jz done**

**mov bl,al**

**mov al,':'**

**call dispc**

**习题5.7-2**

**mov al,bl**

**rol al,4**

**call htoasc ;调用子程序**

**call dispc ;显示一个字符**

**mov al,bl**

**call htoasc ;调用子程序**

**call dispc ;显示一个字符**

**call dispcrlf**

**jmp again**

**done:**

**习题5.8-1**

**• 编写一个子程序，它以二进制形式显示EAX中32位数据，并设计一个主程序验证。**

**;代码段，主程序**

**mov eax,8F98FF00H**

**call dispbd ;调用子程序**

**;代码段，子程序**

**dispbd proc ;32位二进制数的输出**

**push ecx**

**mov ecx,32 ;要输出的字符个数**

**dbd: rol eax,1 ;AL循环左移一位**

**习题5.8-2**

**push eax**

**and al,01h ;取AL最低位**

**add al,30h ;转化成相应的ASCLL码值**

**call dispc ;以二进制的形式显示**

**pop eax**

**loop dbd**

**pop ecx**

**ret**

**dispbd endp**

**习题5.11**

**;计算字节校验和的通用过程**

**;入口参数：DS:EBX＝数组的段地址:偏移地址**

**ECX＝元素个数**

**;出口参数：AL＝校验和**

**;说明：除EAX/EBX/ECX外，不影响其他寄存器**

**checksum proc**

**xor al,al ;累加器清0**

**sum: add al,[ebx] ;求和**

**inc ebx ;指向下一个字节**

**loop sum**

**ret**

**checksum endp**

**习题5.13-1**

**• 利用十六进制字节显示子程序DISPHB设计一个从低地址到高地址逐个字节显示某个主存区域内容的子程序DISPMEM。其入口参数：EAX＝主存偏移地址，ECX＝字节个数（主存区域的长度）。同时编写一个主程序进行验证。**

**;数据段**

**var byte 'This is a test!'**

**;主程序**

**mov eax,offset var**

**mov ecx,sizeof var**

**call dispmem**

**习题5.13-2**

**;子程序**

**dispmem proc**

**push ebx**

**mov ebx,eax**

**dispm1: cmp ecx,0**

**jz dispm2**

**mov al,[ebx]**

**call disphb**

**mov al,' '**

**call dispc**

**习题5.24**

**• 用宏结构实现宏指令FINSUM，它比较两个数varx和vary，若varx ≥ vary，则执行sum＝varx＋8×vary，否则执行sum＝4×varx＋vary。**

**习题5.24解答**

**finsum macro varx,vary,sum**

**push eax**

**if varx ge vary**

**mov eax,vary**

**shl eax,3**

**add eax,varx**

**else**

**mov eax,varx**

**shl eax,2**

**add eax,vary**

**endif**

**mov sum,eax**

**pop eax**

**endm**

**习题5.18**

**• 编写一个宏SWAP，参数是两个32位寄存器或存储器操作数，宏定义体实现两个操作数交换位置，包括两个都是存储器操作数的情况。**

**swap MACRO dword1, dword2**

**push eax**

**mov eax, dword1**

**xchg eax, dword2**

**mov dword1, eax**

**pop eax**

**ENDM**

**习题5.20-1**

**有一个宏定义：**

**defstr macro name,num,string**

**name&num byte ‘&string&’,0**

**endm**

**给出如下宏调用的宏展开：**

**（1）defstr msg,4,< Chapter 4: Program**

**Structure >**

**（2）defstr msg,5,< Chapter 5: Procedure**

**Progamming >**

**习题5.20-2**

**• 解答：**

**（1）**

**msg4 byte‘Chapter 4: Program Structure’,0**

**（2）**

**msg5 byte ‘Chapter 5: Procedure Progamming’,0**

**第6章 Windows编程**

**6.1 简答题**

**（3）运行Windows应用程序，有时为什么会提示某个DLL文件不存在？**

**• Windows程序在运行时需要加载其配套的动态连接库DLL文件，当其没有被搜索到时就会提示不存在。**

**（5）ExitProcess函数可以按汇编语言习惯全部使用小写字母表示吗？**

**• 不能，因为Windows的API函数按照C语言习惯区别大小写字母，是不同的**

**（6）Win32 API中可以使用哪两种字符集？**

**• 8位的ASCII字符集和16位的Unicode字符集**

**（7）为什么调用API函数之后，ECX等寄存器改变了？**

**• 因为API函数并不是按照汇编语言的规则编写的，它的规则是不保护它们**

**6.2 判断题**

**（1）Windows可执行文件中包含动态连接库中的代码。**

**• 错，不含，运行时才加载**

**（2）导入库文件和静态子程序库文件的扩展名都是LIB，所以两者性质相同。**

**• 错。静态子程序库中是子程序代码，导入库文件中是代码所在动态连接库的有关信息。**

**（3）INVOKE语句只能传递主存操作数，不能传递寄存器值。**

**• 错，可以使用寄存器参数**

**（4）Windows控制台是命令行窗口，也就是MS-DOS窗口。**

**• 错，Windows控制台与DOS窗口本质不同**

**（5）类似高级语言，汇编语言的使用结构变量也需要先说明结构类型。**

**• 对。**

**（6）PROC伪指令可使用USES操作符，但PROTO伪指令不可以使用。**

**• 对。**

**（7）在宏定义中，LOCAL伪指令声明标识符；而在过程定义中，LOCAL伪指令用于分配局部变量。**

**• 对。**

**（8）条件汇编IF和条件控制.IF伪指令都包括条件表达式，它们的表达形式一样。**

**• 错，两者不相同。例如，“小于”关系在条件汇编中用“LT”表示，而在条件控制中使用符号“<”。**

**（9）条件控制.IF伪指令和循环控制.WHILE伪指令中的条件表达式具有相同的表达形式。**

**• 对。**

**（10）MASM32软件包只支持32位图形界面应用程序的开发，不支持控制台应用程序开发。**

**• 错，都支持，但汇编和连接命令不同。**

**6.3 填空题**

**（1）Windows系统有3个最重要的系统动态连接库文件，它们是\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_和\_\_\_\_\_\_\_\_。**

**• KERNEL32.DLL，USER32.DLL，GDI32.DLL**

**（2）进行Windows应用程序开发时，需要\_\_\_\_\_\_\_库文件；执行该应用程序时则需要对应的\_\_\_\_\_\_\_\_\_\_库文件。**

**• 导入库，动态连接库**

**（3）获得句柄函数GetStdHandle执行结束，使用\_\_\_\_\_提供返回结果。**

**• EAX**

**（4）函数GetStdHandle需要一个参数，对标准输入设备应该填入\_\_\_\_\_\_\_数值，对标准输出设备应该填入\_\_\_\_\_数值，对标准错误设备应该填入\_\_\_\_\_\_数值。**

**• -10，-11，-12**

**（5）调用ReadConsole函数时，用户在键盘上按下数字8，然后回车，则键盘缓冲区的内容依次是\_\_\_\_。**

**• 38H 0DH 0AH**

**（6）WriteConsole和ReadConsole函数的参数类似，都有5个，第1个参数是\_\_\_\_\_\_，第2个参数是输出或输入缓冲区的\_\_\_\_\_\_，第3个参数是输出或输入的字符\_\_\_\_\_\_\_，第4个参数指向实际输出或输入字符个数的变量，最后1个参数一般要求代入\_\_\_\_\_。**

**• 句柄实例，地址，个数，0**

**（7）消息窗口MessageBox函数有4个参数，第一个是0，第2个是要显示字符串的\_\_\_\_\_\_\_\_\_\_，第3个是\_\_\_\_\_\_\_\_\_的地址指针，第4个参数指明窗口形式。注意字符串要使用\_\_\_\_\_\_\_\_\_\_作为结尾标志。**

**• 地址指针，标题字符串，0（NULL）**

**（8）使用获取系统时间函数GetLocalTime需要定义一个\_\_\_\_\_\_\_\_\_\_结构变量，其中返回系统时间数值，这些数值采用二进制编码，例如日期返回的编码是0019H，它表示日期是\_\_\_\_\_\_\_\_\_\_日。**

**• 系统时间（SYSTEMTIME），25**

**（9）使用扩展的PROC伪指令编写子程序比较方便，例如子程序中需要保护和恢复ESI和EDI，就只需要使用\_\_\_\_\_\_\_\_\_\_就可以了。**

**• USES ESI EDI**

**（10）MASM进行汇编时生成最大化源代码列表，其中语句前使用字母\_\_\_\_\_\_\_\_\_\_表示是通过包含文件插入的语句，使用“\*”符号的语句常是\_\_\_\_\_\_\_\_\_\_的代码，而语句前的数字则说明是\_\_\_\_\_\_\_\_\_\_语句。**

**• C，汇编程序生成，宏展开**

**习题6.4**

**• 执行CPUID指令，直接使用控制台输出函数将处理器识别字符串显示出来。**

**.686**

**.model flat,stdcall**

**option casemap:none**

**includelib bin\kernel32.lib**

**ExitProcess proto,:dword**

**GetStdHandle proto,:dword**

**WriteConsoleA\**

**proto,:dword,:dword,:dword,:dword,:dword**

**WriteConsole equ <WriteConsoleA>**

**STD\_OUTPUT\_HANDLE = -11**

**.data**

**outhandle dword ?**

**outbuffer byte 'The processor ……', 12 dup(0)**

**outbufsize = sizeof outbuffer**

**outsize dword ?**

**mov eax,0**

**cpuid ;执行处理器识别指令**

**mov dword ptr outbuffer+outbufsize-12,ebx**

**mov dword ptr outbuffer+outbufsize-8,edx**

**mov dword ptr outbuffer+outbufsize-4,ecx**

**invoke GetStdHandle,STD\_OUTPUT\_HANDLE**

**mov outhandle,eax**

**invoke WriteConsole,outhandle,\**

**addr outbuffer,outbufsize,addr outsize,0**

**invoke ExitProcess,0**

**习题6.6**

**• 直接使用控制台输出函数实现主存区域内容的显示**

**• 显示形式**

** 每行显示16个字节（128位）**

** 每行开始先显示首个主存单元的偏移地址**

** 然后用冒号分隔主存内容**

**.data**

**var byte \**

**'This is a test!','ABCDEFG','0123456789'**

**\_outsize dword ?**

**\_outhandle dword ?**

**\_membuffer byte 57 dup(20h),13,10**

**.code**

**start: mov eax,offset var**

**mov ecx,sizeof var**

**call dispmem**

**invoke ExitProcess,0**

**dispmem proc**

**test ecx,ecx ;个数为0，不显示**

**jz dispm11 ;退出**

**pushad**

**mov esi,ecx ;ESI=要显示内容的字节数**

**mov edi,eax ;EDI=要显示内容的地址**

**invoke GetStdHandle,STD\_OUTPUT\_HANDLE**

**mov \_outhandle,eax ;获得输出句柄**

**;1.显示缓冲区全部填充为空格**

**dispm1: xor ebx,ebx ;指示显示缓冲区**

**dispm2: mov \_membuffer[ebx],' '**

**inc ebx**

**cmp ebx,(sizeof \_membuffer)-2**

**jb dispm2**

**;2.显示内容所在的存储器地址**

**xor ebx,ebx**

**mov ecx,8 ;地址是十六进制8位**

**mov eax,edi**

**dispm3: rol eax,4**

**mov dl,al**

**and dl,0fh**

**or dl,30h**

**cmp dl,39h**

**jbe dispm4**

**add dl,7**

**dispm4: mov \_membuffer[ebx],dl**

**inc ebx**

**loop dispm3**

**mov \_membuffer[ebx],':' ;显示冒号**

**add ebx,2**

**习题6.6-6**

**mov ecx,16 ;一行最多显示16个字节**

**;3.显示一个字节内容**

**dispm5: mov al,[edi]**

**mov dl,al**

**shr dl,4**

**or dl,30h**

**cmp dl,39h**

**jbe dispm6**

**add dl,7**

**dispm6: mov \_membuffer[ebx],dl**

**inc ebx**

**and al,0fh**

**or al,30h**

**cmp al,39h**

**jbe dispm7**

**add al,7**

**dispm7: mov \_membuffer[ebx],al**

**add ebx,2**

**inc edi ;指向下一个要显示的字节**

**dec esi**

**jz dispm10 ;没有要显示的内容，退出**

**loop dispm5**

**invoke WriteConsole,\_outhandle,\**

**addr \_membuffer,sizeof \_membuffer,\**

**addr \_outsize,0**

**jmp dispm1**

**dispm10: invoke WriteConsole,\_outhandle,\**

**addr \_membuffer,sizeof \_membuffer,\**

**addr \_outsize,0**

**popad**

**dispm11: ret**

**dispmem endp**

**习题6.7**

**• 执行CPUID指令，在消息窗口显示处理器识别字符串，要求该消息窗有OK和Cancel两个按钮。**

**MessageBoxA proto :dword,:dword,:dword,:dword**

**MessageBox equ <MessageBoxA>**

**NULL equ 0**

**MB\_OK equ 1**

**.data**

**szCaption byte '消息窗口',0**

**outbuffer byte '本机的处理器是', 12 dup(0),0**

**outbufsize = sizeof outbuffer-1**

**.code**

**start: mov eax,0**

**cpuid ;执行处理器识别指令**

**mov dword ptr outbuffer+outbufsize-12,ebx**

**mov dword ptr outbuffer+outbufsize-8,edx**

**mov dword ptr outbuffer+outbufsize-4,ecx**

**invoke MessageBox,NULL,addr outbuffer,\**

**addr szCaption,MB\_OK**

**invoke ExitProcess,NULL**

**end start**

**第7章 与Visual C++的混合编程**

**7.1 简答题**

**（1）什么是混合编程？**

**• 组合多种程序设计语言，通过相互调用、参数传递、共享数据结构和数据信息而形成程序的过程**

**（3）汇编语言与C++语言的混合编程有哪两种方法？**

**• 嵌入汇编和模块连接**

**（5）C++语言函数通过什么方式传递入口参数？**

**• 堆栈**

**（6）堆栈帧是一个什么作用的堆栈空间？**

**• 用于传递参数，保存返回地址、局部变量，保护寄存器**

**7.2 判断题**

**（1）C++中可以嵌入汇编指令，但嵌入汇编中不能使用汇编语言的注释形式。**

**• 错，可以使用C++的注释形式，也支持汇编语言的注释形式。**

**（2）汇编语言的宏很有特色，所以仍然可以应用于嵌入汇编中。**

**• 错，不可以。**

**（3）嵌入汇编中可以直接使用C++语言定义的变量。**

**• 对。**

**（4）嵌入汇编语句仍然可以利用OFFSET获得全部变量的地址。**

**• 错**

**（5）MASM汇编语言的C语言类型对应C++语言的\_cdecl。**

**• 对**

**（6）局部变量是通过堆栈创建的。**

**• 对**

**（7）使用寄存器替代频繁访问的变量，可以提升程序性能。**

**• 对，因为减少了相对耗时的存储器访问操作**

**（8）Visual C++的发布版本相对于调试版本来说，只是去掉了调试信息，其他一样。**

**• 错，发布版本通常还要对程序代码进行优化，以获得性能更高的指令代码。**

**（9）汇编语言可以调用C库函数，且不需要导入库文件。**

**• 错。运行时采用动态链接库，开发时仍需要导入库文件。**

**（10）没有调试信息，调试程序无法进行汇编语言级的调试。**

**• 错。可以直接反汇编可执行文件，进行汇编语言级的调试。**

**7.3 填空题**

**（1）有一个数据100，要在嵌入汇编指令中作为立即数，且用十六进制形式表达，可以像汇编语言中一样表达为\_\_\_\_，也可以像C++语言一样表达为\_\_\_\_\_。**

**• 64H，0x64**

**（3）有一个采用C语言类型的汇编语言子程序，如果C++中要调用，声明函数时要增加\_\_\_\_\_\_\_\_\_\_修饰符。**

**• extern "C"**

**（4）函数调用中，通常通过EBP指向堆栈帧，其值减\_\_\_\_\_访问第一个局部变量，其值加\_\_\_\_\_\_访问第一个入口参数，返回地址则由其值加\_\_\_\_\_指向。**

**• 4，8，4**

**（5）C++函数返回一个32位整数，返回值使用\_\_\_保存。**

**• EAX**

**（6）C++语言的char、short和long变量类型对应汇编语言的类型依次是\_\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_\_和\_\_\_\_\_\_\_\_\_\_。**

**• BYTE，WORD，DWORD**

**习题7.4**

**• 阅读如下嵌入汇编的C++程序，说明显示结果。**

**#include <iostream.h>**

**int power2(int,int);**

**void main(void)**

**{ cout<<power2(5,6)<<endl; }**

**int power2(int num,int power)**

**{ \_ \_asm {**

**mov eax,num**

**mov ecx,power**

**shl eax,cl }**

**}**

**习题7.6-1**

**• 如下C++程序中调用汇编语言子程序对这两个数求积**

**• 编写汇编语言子程序模块**

**#include <iostream.h>**

**extern "C" { int multi(int x,int y);}**

**void main(void)**

**{**

**int x,y;**

**cin>>x;**

**cin>>y;**

**cout<< multi(x,y)<<endl;**

**}**

**习题7.6-2**

**• 解答：**

**.686**

**.model flat,c**

**.code**

**multi proc ,var1:dword,var2:dword**

**mov eax,var1**

**imul var2**

**ret**

**multi endp**

**end**

**第8章 DOS环境的程序设计**

**8.1 简答题**

**（1）实地址方式的段基地址为什么常只给出高16位？**

**• 在20位物理地址中，段起始地址的低4位规定必须是0**

**（2）实地址存储模型下，逻辑地址如何转换为物理地址？**

**• 高16位段地址左移4位加偏移地址**

**（3）编写DOS应用程序为什么通常要设置DS值？**

**• DOS程序默认情况下使用不同的数据段和代码段，要求程序员设置DS指向用户的数据区。**

**（4）80x86处理器的MOV指令支持外设数据传送吗？**

**• 不支持，外设数据传送使用IN和OUT指令**

**（5）什么是I/O敏感指令？**

**• 指直接针对I/O端口操作的指令，例如IN、OUT和INS、OUTS，CLI和STI等。**

**（6）PC机中CMOS RAM属于主存空间吗？**

**• PC机中CMOS RAM不属于主存空间，CMOS RAM有64个字节容量，以8位I/O接口形式与处理器连接，通过两个I/O地址访问。**

**8.2 判断题**

**（1）在运行于实地址方式的DOS系统下，允许读写任何主存单元。**

**• 对，不像Windows那样有保护、特权的限制。**

**（2）8086中，“MOV [AX],BX”是非法指令。**

**• 对，8086不支持使用AX进行寄存器间接寻址**

**（3）尽管参数不同，但将模块文件进行连接的连接程序却是同一个文件。**

**• 错，因为OBJ文件格式不同，需要使用不同的连接程序。**

**（4）DOS程序和Windows程序一样使用同一个退出函数（功能）实现退出。**

**• 错**

**（5）重复前缀指令REP可以用于绝大多数MOV指令之前。**

**• 错，只能用于串操作指令，例如MOVS等。**

**（6）指令“OUT DX,AX”的两个操作数均采用寄存器寻址方式，一个来自处理器、一个来自外设。**

**• 错，OUT指令的DX表示采用寄存器间接寻址I/O地址**

**（7）指令“IN BX,20H”正确，表示从20H端口输入一个数据、存放到BX寄存器。**

**• 错。指令“IN AX,20H”才是正确的。**

**（8）IA-32处理器的64K个I/O地址也像存储器地址一样分段管理。**

**• 错，不进行分段管理。**

**8.3 填空题**

**（1）DOS平台下使用实地址存储模型，只能访问\_\_\_\_\_存储空间，仍进行分段管理，但每段不大于\_\_\_\_\_\_容量，且起始物理地址的低4位必须是\_\_\_\_\_\_。**

**• 1MB，64KB，0000**

**（2）在实地址工作方式下，逻辑地址“7380H∶400H”表示的物理地址是\_\_\_\_，并且该段起始于\_\_\_\_\_\_物理地址。**

**• 73C00H，73800H**

**（3）8086支持的16位存储器寻址方式中用作基址寄存器的只能是\_\_\_和\_\_\_，用作变址寄存器的只能是\_\_\_\_和\_\_\_\_。**

**• BX，BP，SI，DI**

**（4）CMPS指令执行后，自动增量或减量的寄存器是\_\_和\_\_\_。希望它们增量可以使用\_\_\_\_指令设置，否则使用\_\_\_\_\_指令设置。**

**• ESI，EDI，CLD，STD**

**（5）指令“IN AL, 21H”的目的操作数是\_\_\_\_\_寻址方式，源操作数是\_\_\_\_\_寻址方式。**

**• 寄存器，I/O地址的直接寻址**

**（6）指令“OUT DX, EAX”的目的操作数是\_\_\_\_\_寻址方式，源操作数是\_\_\_\_\_寻址方式。**

**• I/O地址的间接寻址，寄存器**

**（7）IA-32处理器支持4GB主存，具有\_\_\_\_\_个8位外设端口，在IN和OUT指令中使用\_\_\_\_\_寄存器可以访问到全部端口。**

**• 64K，DX**

**习题8.6**

**• 使用DOS系统功能编写DOS应用程序，要求如下：**

**• 提示“Press ESC to Exit”，等待用户输入字符。如果用户按下ESC（其ASCII码是1BH）则退出，否则继续等待输入字符。**

**• 解答：**

**.model small**

**.686**

**.stack**

**.data**

**msg byte ‘Press ESC to Exit',13,10,'$‘**

**.code**

**start: mov ax,@data**

**mov ds,ax**

**mov ah,9**

**mov dx,offset msg**

**int 21h**

**again:mov ah,1**

**int 21h**

**cmp al,1bh**

**jnz again**

**mov ax,4c00h**

**int 21h**

**end start**

**习题8.10**

**不使用数据串指令改写例8-4程序，实现同样功能。**

**• 解答1：**

**again:mov es:[edi],ax**

**add edi,2**

**loop again**

**不使用数据串指令改写例8-4程序，实现同样功能。**

**• 解答2：**

**mov dx,0b800h**

**mov ds,dx**

**mov ebx,0**

**mov ecx,25\*80**

**again:mov word ptr [ebx],0720h**

**inc ebx**

**inc ebx**

**loop again**

**习题8.14**

**• 利用CMOS RAM的系统时间，将年月日时分秒星期等时间完整的显示出来。**

**• 解答：参考例8-7程序**

**• 难点是星期的显示，可以利用地址表方法如下实现**

**.data**

**week1 byte 'Sunday',0**

**week2 byte 'Monday',0**

**……**

**week7 byte 'Saturday',0**

**week word week1,week2,week3,……,week7**

**mov al,6 ;AL＝6（从6号单元获取星期数据）**

**out 70h,al**

**in al,71h**

**dec al**

**shl al,1**

**xor ah,ah ;查表获得星期字符串**

**mov bx,offset week**

**add bx,ax**

**mov ax,[bx]**

**call dispmsg ;显示星期**