汇编语言基础

为了不影响您的浏览体验，建议您使用电脑或平板类设备进行阅读，以免出现代码格式混乱。好了，下面进入正题。对于汇编这门语言来说，了解是很必要的。无论外面世界的高级语言如何变幻，谁强又谁弱，哪个被淘汰哪个又兴起，哪个才是天下第一，但作为汇编这门语言来说，超然物外。因为偏向底层，直接与计算机硬件打交道，你能了解到计算机原理及编译原理的相关知识，在破解软件、游戏外挂、计算机病毒、加密、脱壳、逆向工程等方面汇编语言也能发挥它极强的作用。所以下面简单介绍与学习一下这门语言。

目录

一、基础介绍

1. 汇编语言简介
2. 80×86 计算机组织
3. 80×86 的指令系统和寻址方式
4. 汇编语言程序格式
5. 汇编语言程序的运行
6. 子程序结构

二、汇编实验

1. 打印输出"Hello World!"
2. 双精度数加减法
3. 四则运算
4. 串操作
5. 数组求和
6. 冒泡排序
7. n 的阶乘
8. 大小写转换

三、例题讲解

1. 单项选择题
2. 填空专题
3. 程序格式专题

汇编语言简介

汇编语言作为大学许多基础课程（如：C语言、计算机组成原理、操作系统等）的先行课，不仅能为我们打下牢固的计算机学习地基，也能让我们建立起一个完整的计算机学习构架。

学来只是为了给其他课程过渡？of course not！它也可以用作程序开发。虽然语法很不友好，但由于偏向底层，所以执行效率极高，作为高效率程序开发仍是个不二之选（但其实如果选它还是蛮二的...）。由于偏向底层硬件，所以在嵌入式开发中仍还是有一席之地。并且，如果你对黑客这一行感兴趣，汇编语言是一定得去了解的，这是一切高级语言的基础，用高级语言能做到的，用汇编一定也能做到。总之，汇编语言很重要！

在学习中，我采用的是网上 Masm for Windows 汇编编译器（点击下载安装包），语言学习嘛，工具不重要，学习到其中的东西才是最主要的。当然后续的实际开发中，工具合不合适，上不上手才是程序员最应该考虑的，当前学生阶段还是不要太挑了。

后面不啰嗦了，大家一起加油吧！

◀返回目录

80×86 计算机组织

计算机主要由 CPU（运算器和控制器集成在的一个芯片）、存储器、输入输出设备 构成，80×86就是这样一组微处理器系列。

80×86寄存器组

通用寄存器——AX、BX、CX、DX、SP、BP、DI、SI，8个通用寄存器（只限于8086/8088）。

其中AX、BX、CX、DX可称为数据寄存器，用来暂时存放计算过程中所用到的操作数、结果或其他信息。它们都可以以字（16位）的形式访问，或者也可以以字节（8位）的形式访问。例如，对AX可以分别访问高位字节AH或者低位字节AL。

- AX（accumulator）作为累加器用，所以它是算术运算的主要寄存器。
- BX（base）可以作为通用寄存器使用。此外在计算存储器地址时，它经常用作基址寄存器。
- CX（count）可以作为通用寄存器使用。此外常用来保存计数值。
- DX（data）可以作为通用寄存器使用。一般在作双字长运算时把DX和AX组合在一起存放一个双字长数，DX用来存放高位字。

其中SP、BP、SI、DI这四个16位寄存器可以像数据寄存器一样在运算过程中存放操作数，但它们只能以字（16位）位单位使用。因为它们更经常的用途是在存储器寻址时，提供偏移地址，所以它们亦称为

指针或变址寄存器。

- SP（stack pointer）称为堆栈指针寄存器，用来指示段顶的偏移地址。
- BP（base pointer）称为基址指针寄存器。
- SI（source index）称为源变址寄存器，用来确定段中某一存储单元的地址，有自动增量和自动减量的功能。
- DI（destination index）称为目的变址寄存器，用来确定段中某一存储单元的地址，有自动增量和自动减量的功能。

专用寄存器——IP、SP、FLAGS、（后续为标志寄存器）OF、SF、ZF、CF、AF、PF、等...

- IP（instruction pointer）为指令指针寄存器，它用来存放代码段中的偏移地址。在程序运行的过程中，它始终指向下一条指令的首地址，与段寄存器CS联用确定下一条指令的物理地址。IP寄存器是计算机中很重要的一个控制寄存器。
- SP为堆栈指针寄存器。它与堆栈段寄存器联用来确定堆栈段中栈顶的地址。
- FLAGS为标志寄存器，又称程序状态寄存器（program status word，PSW）。这是一个存放条件码标志、控制标志和系统标志的寄存器。

下面6个条件码标志用来记录程序中运行结果的状态信息，它们是根据有关指令的运行结果由CPU自动设置的。

- OF（overflow flag）为溢出标志。在运算过程中，操作数超出了机器能表示的范围称为溢出，此时OF位置为1，否则置为0.
- SF（sign flag）为符号标志。记录运算结果的符号，结果为负时置为1，否则置为0。
- ZF（zero flag）为零标志。运算结果为0时置为1，否则置为0。
- CF（carry flag）为进位标志。记录运算时从最高有效位产生的进位值。例如，执行加法指令时，最高有效位有进位时置为1，否则置为0。
- AF（auxiliary carry flag）为辅助进位标志。记录运算时第3位（半个字节）产生的进位值。进位时置为1，否则置为0。
- PF（parity flag）为奇偶标志。当结果操作数中1的个数为偶数时置为1，否则置为0。

段寄存器——CS、DS、SS、ES，4个，后续还增加了两个FS和GS就不加以说明了。

段寄存器也是一种专用寄存器，它们专用于存储器寻址，用来直接或间接地存放段地址。段寄存器长度为16位。

- CS（code segment）为代码段。
- DS（data segment）为数据段。
- SS（stack segment）为堆栈段。
- ES（extra segment）为附加段。

◀返回目录

80×86 的指令系统和寻址方式

何为指令系统？指令嘛，执行来使计算机解决实际问题的。每种计算机都有一组指令集供给用户使用，这组指令集就称为计算机的指令系统。计算机中的指令由操作码字段和操作数字段两部分组成，其中操作码字段指示计算机所要执行的操作。用一句来体现80×86指令的格式：`[标号:]指令的助记符[操作数1[,操作数2]][;注释]`

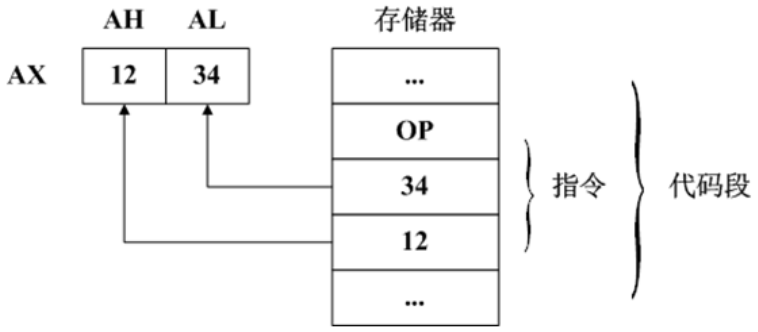
何为寻址方式？寻址嘛，找到要执行的数的地址。准确来说，确定指令中用于说明操作数所在地址的方法称为寻址方法。8086/8088有七种基本的寻址方式。

1. 立即寻址方式

操作数就包含在指令中，它作为指令的一部分，跟在操作后存放在代码段，这种操作数称为立即数（可以是8位，也可以是16位）。

例如：

```
MOV AX,1234H
```



2. 寄存器寻址方式

操作数在CPU内部的寄存器中，指令指定寄存器号。对于16位操作数，寄存器可以是：AX、BX、CX、DX、SI、DI、SP、BP。对于8位操作数，寄存器可以是：AL、AH、BL、BH、CL、CH、DL、DH。这种寻址方式由于操作数就是寄存器中，不需要访问外部存储器来取得操作数，因而可以取得较高的运算速度。例如：

```
MOV AX,BX
```

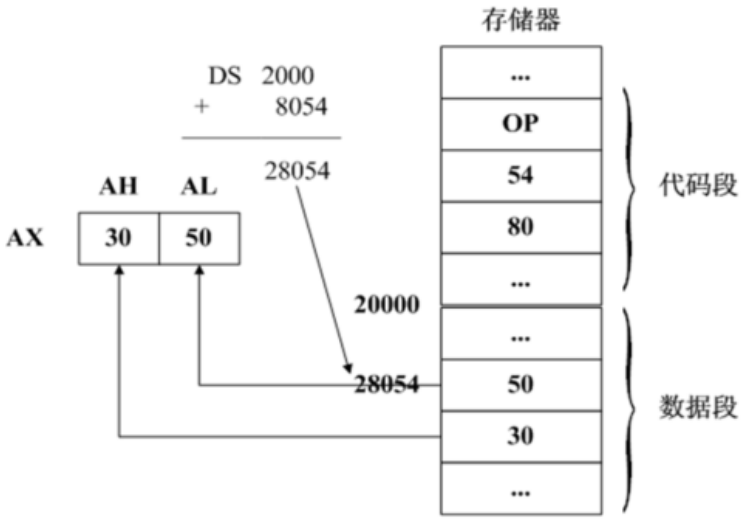
假设指令执行前：(AX)=3064H，(BX)=1234H。指令执行后：(AX)=1234H，(BX)保持不变。

3. 直接寻址方式

操作数在寄存器中，指令直接包含有操作数的有效地址（偏移地址）。操作数一般存放在数据段，其低地址由DS加上指令中直接给出的16位偏移得到。

例如：假设(DS)=2000H

```
MOV AX,[8054H]
```

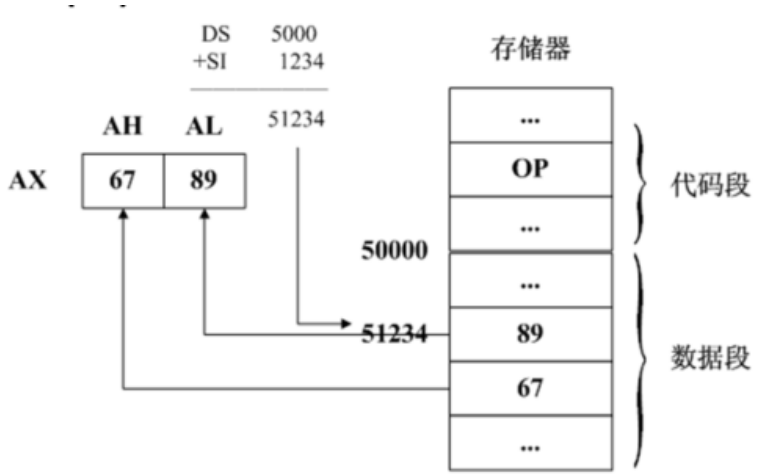


4. 寄存器间接寻址方式

操作数在存储器中，操作数有效地址在SI、DI、BX、BP这四个寄存器之一中。在一般情况下，如果有效地址在SI、DI和BX中，则默认段位DS段。如果有效地址在BP中，则默认段为SS段。

例如：假设(DS)=5000H，(SI)=1234H

```
MOV AX,[SI]
```



5. 寄存器相对寻址方式

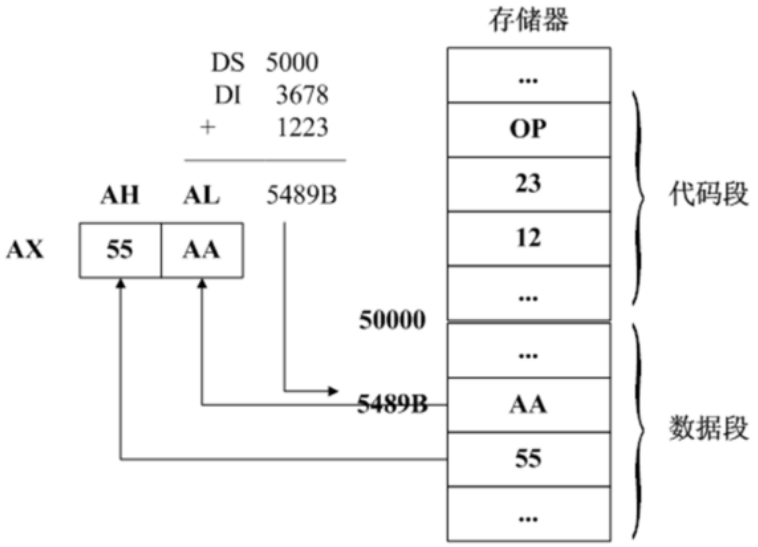
操作数在存储器中，操作数的有效地址是一个基址寄存器（BX、BP）或变址寄存器（SI、DI）内容加上指令中给定的8位或16位位移量之和。在一般情况下，如果SI、DI或BX之内容作为有效地址的一部分，那么引用的段寄存器是DS。如果BP的内容作为有效地址的一部分，那么引用的段寄存器是SS。

$$EA（有效地址）= \begin{matrix}（BX）\\（BP）\\（SI）\\（DI）\end{matrix} + \begin{matrix}8位位移量\\16位位移量\end{matrix}$$

例如：假设(DS)=5000H，(DI)=3678H

```
MOV AX,[DI+1234H]
```

则物理地址=50000+3678+1223=5489BH，如果该字存储单元的内容如下，则(AX)=55AAH



6. 基址加变址寻址方式

操作数在存储器中，操作数的有效地址是由：基址寄存器之一的内容与变址寄存器之一的内容相加。即：

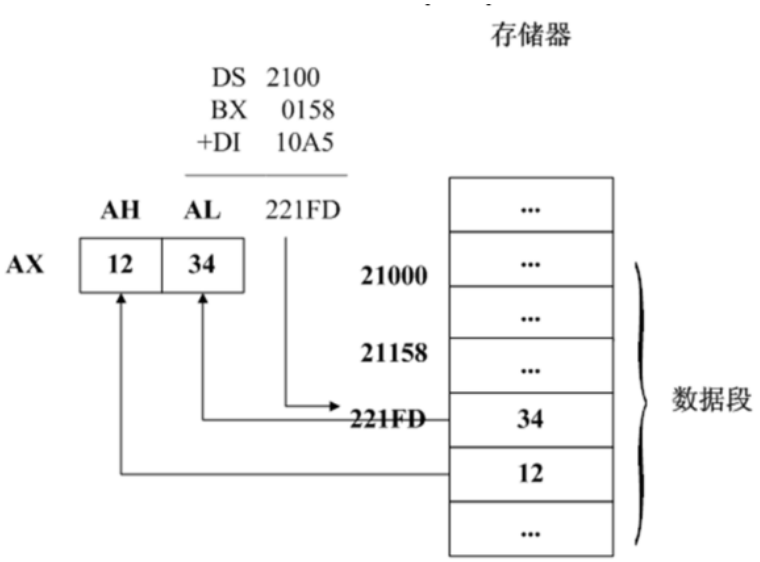
$$EA（有效地址）= \begin{matrix}（BX）\\（BP）\end{matrix} + \begin{matrix}（SI）\\（DI）\end{matrix}$$

在一般情况下，如果BP的内容作为有效地址的一部分，则引用的段寄存器是SS，否则是DS。

例如：假设(DS)=2100H，(BX)=0158H，(DI)=10A5H

```
MOV AX,[BX][DI]
```

假设该字存储单元的内容如下，则(AX)=1234H



7. 相对基址加变址寻址方式

操作数在存储器中，其有效地址是由：基址寄存器之一的内容与变址寄存器之一的内容及指令中给定的8位或16位位移量相加得到。即：

$$\text{EA（有效地址）} = \begin{matrix}(BX)\\(BP)\end{matrix} + \begin{matrix}(SI)\\(DI)\end{matrix} + \begin{matrix}8位\\16位\end{matrix}\text{位移量}$$

在一般情况下，如果BP的内容作为有效地址的一部分，则引用的段寄存器是SS，否则是DS。

例如：假设(DS)=5000H，(BX)=1223H，(DI)=54H，(51275)=54H，(51276)=76H

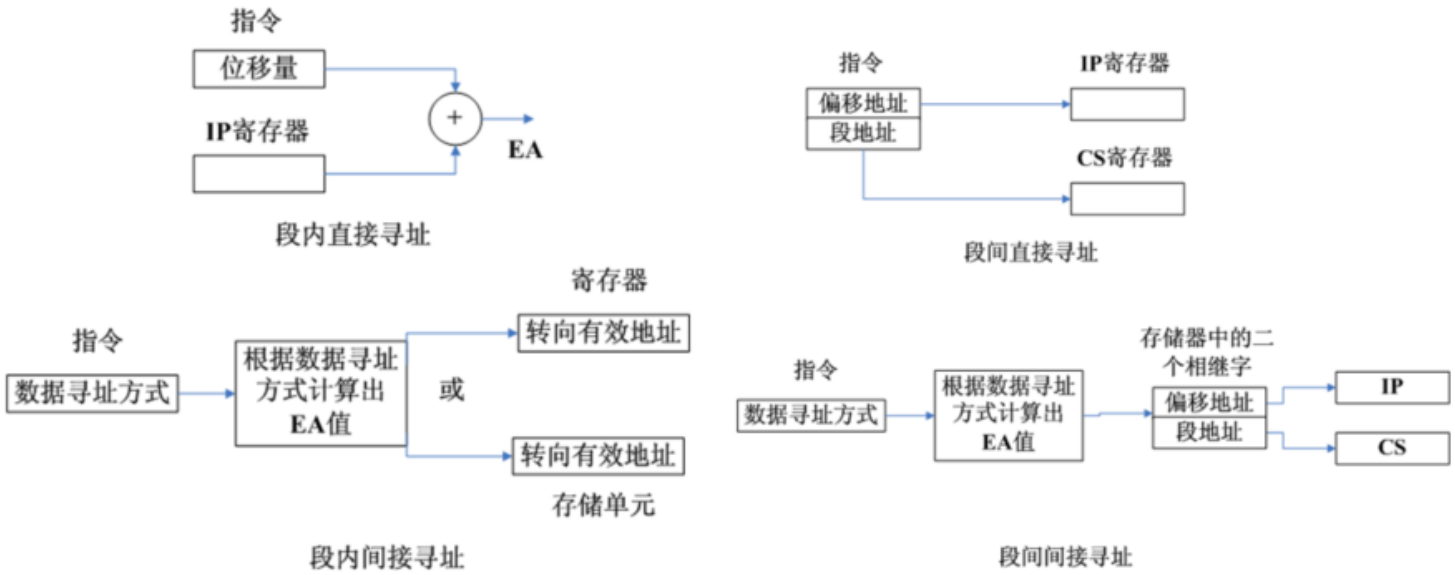
```
MOV AX,[BX+DI-2]
```

则物理地址=50000+1223+0054+FFFFE=51275H，指令执行后，(AX)=7654H

相对基址加变址表示方法多种多样，下面这四种表示方法均是等价的：

```
MOV AX,[BX+DI+1234H]
MOV AX,1234H[BX][DI]
MOV AX,1234H[BX+DI]
MOV AX,1234H[DI][BX]
```

除了这7种基本的寻址方式外，8086/8088还提供了4种基于转移地址的寻址方式（左边为段内，右边为段间）：



◀返回目录

汇编语言程序格式

汇编语言有多种类型的语句——指令语句，伪指令语句、宏指令语句。汇编语言在对源程序进行汇编时，把指令语句翻译成机器指令，而伪指令语句没有与之相对应的机器指令，换句话说，伪指令语句实际属于一种说明语句，给编译器执行，不由CPU执行。

指令语句格式：`(标号)指令助记符(操作数(,操作数))(;注释)`

伪指令语句格式：`(名字)伪指令定义符(参数,...,参数)(;注释)`

汇编语言中标号和名字不能是汇编语言的保留字，如不能是“MOV”。汇编语言不区分保留字中字母的大小写。

下面介绍一下常见的伪指令/伪操作。

（1） 段定义语句

segment和ends是一对成对使用的伪指令，这是在写可被编译器编译的汇编程序时，必须要用到的一对伪指令。segment说明一个段开始，ends说明一个段结束。一个汇编程序是由多个段组成的，这些段被用来存放代码、数据或被当作栈空间来使用。使用格式为：

```
段名 SEGMENT
段名 ENDS
```

一个简单的段示例：

```
DSEG SEGMENT
    MESS DB 'HELLO' , 0DH , 0AH , '$'
DSEG ENDS
```

汇编程序根据段开始语句和段结束语句判断出源程序的段划分，为了有效地产生目标代码，汇编程序还要了解各程序段与段寄存器间的对应关系。这种对应关系由段使用设定语句说明。使用格式为：

```
ASSUME 段寄存器名:段名[,段寄存器名:段名…]
```

段寄存器名可以是CS、DS、SS、ES。例如：CS寄存器对应CSEG段，DS寄存器对应DSEG段。

```
ASSUME CS:CSEG,DS:DSEG
```

下面感受一段较为完整的段定义：

```
DSEG1 SEGMENT
    VARW DW 12
DSEG1 ENDS
DSEG2 SEGMENT
    XXX DW 0
DSEG2 ENDS
CSEG SEGMENT
    ASSUME CS:CSEG , DS: DSG1 , ES : DSG2
    MOV AX , DSEG1
    MOV DS , AX
    MOV AX , DSEG2
    MOV ES , AX
    ……
    ASSUME DS: DSG2 , ES :NOTHING    ;NOTHING表示该ES寄存器不与任何段有对应关系
    MOV AX , DSEG2
    MOV DS , AX
    ……
DSEG ENDS
```

(2) 数据定义及存储器分配伪操作

数据定义语句是最常用的伪指令语句，一般格式如下：

```
[变量名] 数据定义符 表达式[,表达式,...,表达式][;注释]
```

例如：

```
VARB DB 3
VARW DW -1234
BUFF DB 100, 3+4, 5*6
```

DB——定义字节数据项（Define Byte），DW——定义字数据项（Define Word），DD（Define Double）——定义双字数据项。如何定义没有初值的数据项呢？如果数据定义语句中的表达式只是一个问号（？），则表示不预置对应的变量初值，而仅仅是给变量分配存储单元。例如：

```
INBUFF DB 5, ?, ?, ?, 8, ?
VARW DW ?
OLDV DD ?
```

上面定义的都是数值型变量，而如果要定义字符串该怎么表示？定义字节数据的伪指令DB也可以用于定义字符串，字符串要用引号括起来（不区分单引号或者双引号，只要求配对），例如：

```
MESS DB 'HELLO!'
```

上述语句等价于如下：

```
MESS DB 'H','E','L','L','O','!'
```

有时需要定义数组，有时还需要定于数据缓冲区。例如：

```
BUFFER DB 0, 0, 0, 0, 0, 0, 0, 0
```

以上操作太不方便了有没有！为此，汇编语言提供了在数据定义语句中使用的重复操作符DUP，下面这条语句等价于上面的语句：

```
BUFFER DB 8 DUP(0)
```

来看看关于重复操作符DUP的一些其他用法案例：

```
BUFFER1  DB 5 , 0 , 5 DUP(?)
BUFFER2  DB 100 DUP(0, 2 DUP(1, 2), 0, 3)
BUFFER3  DB 256 DUP('ABCDE')
```

（3） 程序开始与结束伪操作

一段汇编程序代码从哪儿开始被执行，又从哪儿结束，都由我们END伪操作来执行，其格式为： `END [标号]` 。其中标号表示程序开始执行的起始地址，即程序是从END所指的“标号”开始执行，遇到END指令后结束。如果END没有指定标号，则程序从头开始运行。下面由两个对比的汇编程序段来说明END伪操作的用处。

```
CODES SEGMENT
START:
    MOV DL , '1'
    MOV AH , 2
    INT 21H
    MOV DL , '2'
    MOV AH , 2
    INT 21H

    MOV AH , 4CH
    INT 21H
CODES ENDS
    END START
```

```
CODES SEGMENT
    MOV DL , '1'
    MOV AH , 2
    INT 21H
START:
    MOV DL , '2'
    MOV AH , 2
    INT 21H

    MOV AH , 4CH
    INT 21H
CODES ENDS
    END START
```

上面的程序格式，包括数据定义我们都讲过了，现在不用去理解这段代码到底要干嘛。观察区别，我们发现了这两个程序都有END这个伪操作符，说明是这个程序结束的地方，而同时END后面指明了标号START，说明这段程序被执行的第一行应该是从START处开始的，所以左边的代码比右边的代码多执行了三句操作。

◀返回目录

汇编语言程序的运行

有了前面的知识作铺垫，那么一段汇编程序是如何在计算机中运行的呢？在DOS中，可执行文件中的程序P1若要运行，必须有一个正在运行的程序P2将P1从可执行文件中加载入内存，将CPU的控制权交给它，此时P2处于挂起状态，P1才能得以运行。当P1运行完毕后，应该将CPU的控制权交还给使它得以运行的程序P2。如果要清楚的了解执行电脑中某个exe可执行文件时，是哪个正在运行的程序将它载入内存的，必须先了解一个操作系统的外壳的概念。

> **操作系统的外壳：** 操作系统是由多个功能模块组成的庞大、复杂的软件系统。任何通用的操作系统，都要提供一个称为shell（外壳）的程序。用户（操作人员）使用这个程序来操作计算机系统工作。DOS中有一个程序command.exe（cmd.exe），这个程序在DOS中称为命令解释器，也就是DOS系统的shell。

我们在DOS中直接执行exe可执行文件时，是正在运行的command程序将exe中的程序加载入内存。command设置CPU的CS:IP指向程序的第一条指令（即程序的入口），从而使程序得以运行。程序运行结束后，返回到command中，CPU继续运行command。

汇编程序从写出到执行的整个过程：

```
 编程  -> .asm -> 编译  -> .obj -> 链接  -> .exe -> 加载  -> 内存中的程序 -> 运行
(edit)          (masm)           (link)         (command)                 (CPU)
```

由于不是针对零基础编程的同学，所以编写程序、编译链接这些作用不再赘述。

本来想再细致说明一些汇编常用的 `INT 21H` 系统调用、寄存器用法、以及一些常用的汇编指令。但我们还是只针对考试以及了解为主，程序中重要的指令我将在第二节汇编实验中一一说明。

◀返回目录

子程序结构

学过C语言的同学都知道，为了程序的可读性以及代码的复用，我们不可能把所有的代码都写在main函数里，所以有了函数的出现。汇编程序也是一样的道理，所以在汇编语言中，我们给出了子程序的概念。

过程定义语句（process）： 利用过程定义伪指令语句，可把程序片段说明为具有近类型或远类型的过程，并且能给过程取一个名字。

过程定义语句的格式如下：

```
过程名 PROC [NEAR | FAR]
    ...
    RET
过程名 ENDP
```

> **注意：** 1. 过程的类型在过程定义开始语句PROC中指定；2. 过程可以被指定位近(NEAR)类型，也可以被指定为远类型。如果不指定，则通常默认为近类型；3. 定义一个过程的开始语句PROC和结束语句ENDP前使用的过程名称必须一致，从而保持配对。

◀返回目录

打印输出"Hello World!"

```
DATAS   SEGMENT                              ;定义一个DATAS段
    STR1   DB   'Hello World!',13,10,'$'     ;具体看下面解答部分↓
DATAS   ENDS                                 ;DATAS段结束

CODES   SEGMENT                              ;定义一个CODES段
    ASSUME     CS:CODES,DS:DATAS             ;关联代码段寄存器CODES和数据段寄存器CS、DATAS和DS
START:                                       ;程序开始标号处
    MOV   AX,DATAS                           ;先将段DATAS中立即数存到通用寄存器AX中作为中转
    MOV   DS,AX                              ;将立即数送到段寄存器DS中
    LEA   DX,STR1                            ;调用字符串开始地址
    MOV   AH,9                               ;调用DOS系统9号功能：显示字符串
    INT   21H                                ;调用DOS功能中断

    MOV   AH,4CH                             ;调用DOS系统4C号功能：结束程序
    INT   21H                                ;调用DOS功能中断
CODES   ENDS                                 ;CODES段结束
    END   START                              ;汇编程序运行结束

;--------------------------------------------------------------------------------
;1. 如何理解第二段中“STR1   DB   'Hello World!',13,10,'$'”这句指令?

;答：由于STR是汇编一个关键字指令，则命名应该避免否则会报错;
;     伪指令DB用于字符串中定义字节数据，字符串用引号括起来;
;     13，10分别是回车符与换行符的ASCII值，执行的结果是回车换行，$是字符串结束的标志。
;--------------------------------------------------------------------------------
;2. 如何理解第八、九段中MOV的操作?

;答：MOV指令不允许将立即数直接送给段寄存器，通常是借助通用寄存器中转。
;--------------------------------------------------------------------------------
;3. 什么是立即数?

;答：汇编语言中中操作数有三种：寄存器操作数、存储器操作数和立即数;
;     其中立即数相当于高级语言中的常量（常数），它是直接出现在指令中的数，
;     不用存储在寄存器或存储器中。如指令ADD AL,06H中的06H即为立即数。
;--------------------------------------------------------------------------------
```

◀返回目录

双精度数加减法

双精度数加法

```
DATAS SEGMENT                          ;定义一个DATAS段
    X DW 1,2                           ;给字变量X赋值，具体看下面解答部分↓0001H，0002H
    Y DW 3,4                           ;给字变量Y赋值，其中X和Y为加数
    Z DW 0,0                           ;给字变量Z赋值，其中Z为X和Y的和，输出0406
DATAS ENDS                             ;DATAS段结束

CODES SEGMENT                          ;定义一个CODES段
    ASSUME CS:CODES,DS:DATAS           ;关联代码段寄存器CODES和数据段寄存器CS、DATAS和DS
START:                                 ;程序开始标号处
    MOV AX,DATAS                       ;先将段DATAS中立即数存到通用寄存器AX中作为中转
    MOV DS,AX                          ;将立即数送到段寄存器DS中

    MOV AX,X                           ;将X的低位部分存放在AX寄存器中
    MOV DX,X+2                         ;将X的高位部分存放在DX寄存器中

    ADD AX,Y                           ;将X的低位部分和Y的低位部分相加，结果存放在AX寄存器中
    ADD DX,Y+2                         ;将X的高位部分和Y的高位高位相加，结果存放在DX寄存器中

    MOV Z,AX                           ;将AX中X和Y低位部分相加得到的结果存放在Z的低位部分中
    MOV Z+2,DX                         ;将DX中X和Y高位部分相加得到的结果存放在Z的高位部分中
    ;输出Z低位的0
    MOV DX,Z                           ;将Z的低位16位放在DX寄存器（16位）中，此时DH中八位为00H，DL中八位
    MOV DL,DH                          ;将DH中的值0 赋给DL用于输出
    ADD DL,'0'                         ;把数字变成字符输出，因为汇编中只能输出字符；0的ASCII值是30H，数字
    MOV AH,02H                         ;调用DOS系统的02号功能：显示一个字符
    INT 21H                            ;调用DOS功能中断
    ;输出Z低位的4
    MOV DX,Z                           ;由于DL中的值被改变，所以重新将Z的低位存入DX中
    ADD DL,'0'                         ;把数字变成字符输出，因为汇编中只能输出字符；0的ASCII值是30H，数字
    MOV AH,02H                         ;调用DOS系统的02号功能：显示一个字符
    INT 21H                            ;调用DOS功能中断
    ;输出Z高位的0
    MOV DX,Z+2                         ;将Z的高位16位放在DX寄存器（16位）中，此时DH中八位为00H，DL中八位
    MOV DL,DH                          ;将DH中的值0 赋给DL用于输出
    ADD DL,'0'                         ;把数字变成字符输出，因为汇编中只能输出字符；0的ASCII值是30H，数字
    MOV AH,02H                         ;调用DOS系统的02号功能：显示一个字符
    INT 21H                            ;调用DOS功能中断
    ;输出Z高位的6
    MOV DX,Z+2                         ;由于DL中的值被改变，所以重新将Z的高位存入DX中
    ADD DL,'0'                         ;把数字变成字符输出，因为汇编中只能输出字符；0的ASCII值是30H，数字
    MOV AH,02H                         ;调用DOS系统的02号功能：显示一个字符
    INT 21H                            ;调用DOS功能中断
```

```
    MOV AH,4CH                      ;调用DOS系统4C号功能：结束程序
    INT 21H                         ;调用DOS功能中断
CODES ENDS                          ;CODES段结束
    END START                       ;汇编程序运行结束

;--------------------------------------------------------------------------
;1. 在DATAS段中，如何理解“X DW 1,2”这个赋值过程?

;答：DW全称为：define word，其中一个word字占四个字节。一个字节又占八位，
;    所以，一个DW子类型变量占32位，即32个长度的二进制数。1，2是十进制表
;    示法，而如果要用我们常用的16进制，则应该表示为：0001H,0002H，其中
;    0001H在低半位，0002H在高半位。因为4位二进制等价于1位十六进制，所以
;    4个长度的十六进制表示16位二进制。
;--------------------------------------------------------------------------
;2. DW的高低位是分别存放什么寄存器之中?

;答：双字长运算时用来存放高位字的寄存器为DX，存放低位字的寄存器一般是AX。
;--------------------------------------------------------------------------
;3. 在程序第14段，为什么X+2就能代表X的高位（同理Y和Z）?

;答：X是变量名，本身也是地址。X+2表示基于X的地址多加了两个字节长度（16位），
;    而第1问中讲了DW一共由32位二进制来表示，所以第16位上下分别其高低部分。
;--------------------------------------------------------------------------
;4. 如何输出一个数字?

;答：如果要输出，我们得用到DX寄存器，其中DL用于输出显示。由于DL只能存放
;    8位，所以如果一个数大于8位，应该每一部分分别输出。而汇编中只能输出
;    字符，利用DOS系统的02号功能：显示一个字符，所以我们得先把数字变为字
;    符，在数字后面加上字符'0'即可。
;--------------------------------------------------------------------------
```

双精度数减法

```
DATAS SEGMENT                          ;定义一个DATAS段
    X DW 4,6                           ;给字变量X赋值，0004H，0006H
    Y DW 2,3                           ;给字变量Y赋值，其中X和Y分别为被减数和减数
    Z DW 0,0                           ;给字变量Z赋值，其中Z为X和Y的差，输出0203
DATAS ENDS                             ;DATAS段结束

CODES SEGMENT                          ;定义一个CODES段
    ASSUME CS:CODES,DS:DATAS           ;关联代码段寄存器CODES和数据段寄存器CS、DATAS和DS
START:                                 ;程序开始标号处
    MOV AX,DATAS                       ;先将段DATAS中立即数存到通用寄存器AX中作为中转
    MOV DS,AX                          ;将立即数送到段寄存器DS中

    MOV AX,X                           ;将X的低位部分存放在AX寄存器中
    MOV DX,X+2                         ;将X的高位部分存放在DX寄存器中

    SUB AX,Y                           ;将X的低位部分和Y的低位部分相减，结果存放在AX寄存器中
    SUB DX,Y+2                         ;将X的高位部分和Y的高位高位相减，结果存放在DX寄存器中

    MOV Z,AX                           ;将AX中X和Y低位部分相减得到的结果存放在Z的低位部分中
    MOV Z+2,DX                         ;将DX中X和Y高位部分相减得到的结果存放在Z的高位部分中
    ;输出Z低位的0
    MOV DX,Z                           ;将Z的低位16位放在DX寄存器（16位）中，此时DH中八位为00H，DL中八位
    MOV DL,DH                          ;将DH中的值0 赋给DL用于输出
    ADD DL,'0'                         ;把数字变成字符输出，因为汇编中只能输出字符；0的ASCII值是30H，数字
    MOV AH,02H                         ;调用DOS系统的02号功能：显示一个字符
    INT 21H                            ;调用DOS功能中断
    ;输出Z低位的2
    MOV DX,Z                           ;由于DL中的值被改变，所以重新将Z的低位存入DX中
    ADD DL,'0'                         ;把数字变成字符输出，因为汇编中只能输出字符；0的ASCII值是30H，数字
    MOV AH,02H                         ;调用DOS系统的02号功能：显示一个字符
    INT 21H                            ;调用DOS功能中断
    ;输出Z高位的0
    MOV DX,Z+2                         ;将Z的高位16位放在DX寄存器（16位）中，此时DH中八位为00H，DL中八位
    MOV DL,DH                          ;将DH中的值0 赋给DL用于输出
    ADD DL,'0'                         ;把数字变成字符输出，因为汇编中只能输出字符；0的ASCII值是30H，数字
    MOV AH,02H                         ;调用DOS系统的02号功能：显示一个字符
    INT 21H                            ;调用DOS功能中断
    ;输出Z高位的3
    MOV DX,Z+2                         ;由于DL中的值被改变，所以重新将Z的高位存入DX中
    ADD DL,'0'                         ;把数字变成字符输出，因为汇编中只能输出字符；0的ASCII值是30H，数字
    MOV AH,02H                         ;调用DOS系统的02号功能：显示一个字符
    INT 21H                            ;调用DOS功能中断
```

```
    MOV AH,4CH                  ;调用DOS系统4C号功能：结束程序
    INT 21H                     ;调用DOS功能中断
CODES ENDS                      ;CODES段结束
    END START                   ;汇编程序运行结束
```

◀返回目录

四则运算

```
DATAS SEGMENT                   ;定义一个DATAS段
    X DW 3                      ;给字变量X赋值，X占16位
    Y DW 2                      ;给字变量Y赋值，Y占16位
    STR1 DB 'X = $'             ;用于输出的表达式字符串，下同理
    STR2 DB 'Y = $'
    STR3 DB 'X + Y = $'
    STR4 DB 'X - Y = $'
    STR5 DB 'X * Y = $'
    STR6 DB 'X / Y = $'
    STR7 DB '...$'              ;余数的表达形式，如：5/2=2...1
DATAS ENDS                      ;DATAS段结束

CODES SEGMENT                   ;定义一个CODES段
    ASSUME CS:CODES,DS:DATAS    ;关联代码段寄存器CODES和数据段寄存器CS、DATAS和DS
START:                          ;程序开始标号处
    MOV AX,DATAS                ;先将段DATAS中立即数存到通用寄存器AX中作为中转
    MOV DS,AX                   ;将立即数送到段寄存器DS中

    ;输出"X = "
    LEA DX,STR1                 ;调用字符串STR1开始有效地址（偏移地址），存放在寄存器DX中
    MOV AH,09H                  ;调用DOS系统9号功能：显示字符串
    INT 21H                     ;调用DOS功能中断
    ;输出X的值
    MOV DX,X                    ;将X的值存放在DX寄存器中
    ADD DL,'0'                  ;把数字变成字符输出，因为汇编中只能输出字符；0的ASCII值是30H，数字加上
    MOV AH,02H                  ;调用DOS系统的02号功能：显示一个字符
    INT 21H                     ;调用DOS功能中断
    ;输出回车换行
    MOV DL,10                   ;输出回车换行，回车键ACSII值为10
    MOV AH,02H                  ;调用DOS系统的02号功能：显示一个字符
    INT 21H                     ;调用DOS功能中断

    ;输出"Y = "
    LEA DX,STR2                 ;调用字符串STR2开始有效地址（偏移地址），存放在寄存器DX中
    MOV AH,09H                  ;调用DOS系统9号功能：显示字符串
    INT 21H                     ;调用DOS功能中断
    ;输出Y的值
    MOV DX,Y                    ;将Y的值存放在DX寄存器中
    ADD DL,'0'                  ;把数字变成字符输出，因为汇编中只能输出字符；0的ASCII值是30H，数字加上
    MOV AH,02H                  ;调用DOS系统的02号功能：显示一个字符
```

```
INT 21H                    ;调用DOS功能中断
;输出回车换行
MOV DL,10                  ;输出回车换行，回车键ACSII值为10
MOV AH,02H                 ;调用DOS系统的02号功能：显示一个字符
INT 21H                    ;调用DOS功能中断

;输出"X + Y = "
LEA DX,STR3                ;调用字符串STR3开始有效地址（偏移地址），存放在寄存器DX中
MOV AH,09H                 ;调用DOS系统9号功能：显示字符串
INT 21H                    ;调用DOS功能中断
;输出X+Y的值
MOV DX,X                   ;将X的值存放在DX中
ADD DX,Y                   ;将X和Y相加，结果存放在DX中
ADD DL,'0'                 ;把数字变成字符输出，因为汇编中只能输出字符；0的ASCII值是30H，数字加上
MOV AH,02H                 ;调用DOS系统的02号功能：显示一个字符
INT 21H                    ;调用DOS功能中断
;输出回车换行
MOV DL,10                  ;输出回车换行，回车键ACSII值为10
MOV AH,02H                 ;调用DOS系统的02号功能：显示一个字符
INT 21H                    ;调用DOS功能中断

;输出"X - Y = "
LEA DX,STR4                ;调用字符串STR4开始有效地址（偏移地址），存放在寄存器DX中
MOV AH,09H                 ;调用DOS系统9号功能：显示字符串
INT 21H                    ;调用DOS功能中断
;输出X-Y的值
MOV DX,X                   ;将X的值存放在DX中
SUB DX,Y                   ;将X和Y相减，结果存放在DX中
ADD DL,'0'                 ;把数字变成字符输出，因为汇编中只能输出字符；0的ASCII值是30H，数字加上
MOV AH,02H                 ;调用DOS系统的02号功能：显示一个字符
INT 21H                    ;调用DOS功能中断
;输出回车换行
MOV DL,10                  ;输出回车换行，回车键ACSII值为10
MOV AH,02H                 ;调用DOS系统的02号功能：显示一个字符
INT 21H                    ;调用DOS功能中断

;输出"X * Y = "
LEA DX,STR5                ;调用字符串STR5开始有效地址（偏移地址），存放在寄存器DX中
MOV AH,09H                 ;调用DOS系统9号功能：显示字符串
INT 21H                    ;调用DOS功能中断
;输出X*Y的值
MOV AX,X                   ;MUL乘法指令中一个乘数在AL寄存器中
MUL Y                      ;Y为另一个乘数，X*Y的结果存放在了AX寄存器中
```

```
    MOV DX,AX                   ;将AX中的乘积内容送到DX中用于输出
    ADD DL,'0'                  ;把数字变成字符输出，因为汇编中只能输出字符；0的ASCII值是30H，数字加上
    MOV AH,02H                  ;调用DOS系统的02号功能：显示一个字符
    INT 21H                     ;调用DOS功能中断
    ;输出回车换行
    MOV DL,10                   ;输出回车换行，回车键ACSII值为10
    MOV AH,02H                  ;调用DOS系统的02号功能：显示一个字符
    INT 21H                     ;调用DOS功能中断

    ;输出"X / Y = "
    LEA DX,STR6                 ;调用字符串STR6开始有效地址（偏移地址），存放在寄存器DX中
    MOV AH,09H                  ;调用DOS系统9号功能：显示字符串
    INT 21H                     ;调用DOS功能中断
    ;输出X/Y的商值
    XOR DX,DX                   ;做16位除法前需要将DX清零
    MOV AX,X                    ;DIV除法指令中16位被除数在AX寄存器中
    DIV Y                       ;Y为除数，X/Y的结果16位商存放在AX中，余数存放在DX中，(如果是8位，商存
    MOV DX,AX                   ;将AX中的商值内容送到DX中用于输出
    ADD DL,'0'                  ;把数字变成字符输出，因为汇编中只能输出字符；0的ASCII值是30H，数字加上
    MOV AH,02H                  ;调用DOS系统的02号功能：显示一个字符
    INT 21H                     ;调用DOS功能中断
    ;输出"..."
    LEA DX,STR7                 ;调用字符串STR7开始有效地址（偏移地址），存放在寄存器DX中
    MOV AH,09H                  ;调用DOS系统9号功能：显示字符串
    INT 21H                     ;调用DOS功能中断
    ;输出X/Y的余数值
    XOR DX,DX                   ;由于DL中值已被覆盖，重新进行一次除法运算
    MOV AX,X                    ;DIV除法指令中16位被除数在AX寄存器中
    DIV Y                       ;Y为除数，X/Y的结果16位商存放在AX中，余数存放在DX中，(如果是8位，商存
    ADD DL,'0'                  ;把数字变成字符输出，因为汇编中只能输出字符；0的ASCII值是30H，数字加上
    MOV AH,02H                  ;调用DOS系统的02号功能：显示一个字符
    INT 21H                     ;调用DOS功能中断

    MOV AH,4CH                  ;调用DOS系统4C号功能：结束程序
    INT 21H                     ;调用DOS功能中断
CODES ENDS                      ;CODES段结束
    END START                   ;汇编程序运行结束
```

◀返回目录

串操作

串拷贝

```
DATAS SEGMENT                           ;定义一个DATAS段
    STR1 DB 'fmw2017110110 $'           ;定义一串源字符串STR1，$是字符串结束的标志
    STR2 DB 20 DUP(?)                   ;定义一串目的字符串STR2，具体看下面解答部分↓
DATAS ENDS                              ;DATAS段结束

CODES SEGMENT                           ;定义一个CODES段
    ASSUME CS:CODES,ES:DATAS            ;关联代码段寄存器CODES和数据段寄存器CS、DATAS和ES
START:                                  ;程序开始标号处
    MOV AX,DATAS                        ;先将段DATAS中立即数存到通用寄存器AX中作为中转
    MOV ES,AX                           ;将立即数送到段寄存器ES中

    LEA SI,STR1                         ;调用字符串STR1开始有效地址（偏移地址），存放在源变址寄存器SI
    LEA DI,STR2                         ;调用字符串STR2开始有效地址（偏移地址），存放在目的变址寄存器D
    MOV CX,20                           ;将STR2的存储单元20存放入保存计数值的CX寄存器中
    CLD                                 ;clear direction flag，具体看下面解答部分↓
    REP MOVS ES:BYTE PTR[DI],ES:[SI]    ;表明是字节BYTE操作，具体看下面解答部分↓

    LEA DX,STR2                         ;调用字符串STR2开始有效地址（偏移地址），存放在寄存器DX中
    MOV AX,ES                           ;将段寄存器ES的值放入AX中
    MOV DS,AX                           ;将AX的值放入段寄存器DS中（AX作为中转）,DS作为DX的偏移地址
    MOV AH,09H                          ;调用DOS系统9号功能：显示字符串
    INT 21H                             ;调用DOS功能中断

    MOV AH,4CH                          ;调用DOS系统4C号功能：结束程序
    INT 21H                             ;调用DOS功能中断
CODES ENDS                              ;CODES段结束
    END START                           ;汇编程序运行结束

;--------------------------------------------------------------------------------
;1. 如何理解第三段中“STR2 DB 20 DUP(?)”这句指令的意思?

;答：定义字符串STR2，由于STR2在此程序中用作拷贝的对象，所以STR2在定义时
;    预先给它分配20个存储单元，且不预置对应的变量初值。使用重复操作符DUP，
;    来设置20个存储单元的初值。使用'?'来表示未设值的变量。所以在定义时被
;    表示成“20 DUP(?)”。
;--------------------------------------------------------------------------------
;2. 第十二、十三段中，为什么STR1和STR2的有效地址要分别存入SI和DI寄存器中?

;答：通用寄存器SI（source index）称为源变址寄存器，
;    用来确定段中某一存储单元的地址，有自动增量和自动减量的功能。
;    通用寄存器DI（destination index）称为目的变址寄存器，
;    用来确定段中某一存储单元的地址，有自动增量和自动减量的功能。
;--------------------------------------------------------------------------------
```

```
;3．第十五段中CLD指令的作用是什么?

;答：该指令使DF=0，在执行串处理指令时可使变址寄存器SI和DI的地址指针自动
;     增加。(DF=0)表示方向向前。与之相反的是STD（set direction flag）指令，
;     该指令使DF=1，在执行串处理指令时可使地址自动减少。（DF=1）表示向后。
;-------------------------------------------------------------------------
;4．如何理解第十六段中“REP MOVS ES:BYTE PTR[DI],ES:[SI]”指令操作?

;答：MOVS指令可以把由源变址寄存器指向的数据段中的一个字（或双字，或字节）
;     传送到由目的变址寄存器指向的附加段中的一个字（或双字，或字节）中去，
;     同时根据方向标志DF及数据格式（字、双字或字节）对源变址寄存器和目的变
;     址寄存器进行修改。当该指令与前缀REP联用时，则可将数据段中的整串数据
;     传送到附加段中去。这里，源串必须在数据段DS中，目的串必须在附加段ES中
;     但源串允许使用段跨越前缀来修改。在于REP联用时还必须先把数据串的长度
;     送到计数寄存器值CX中，以便控制指令结束。
;-------------------------------------------------------------------------
```

串扫描

```
DATAS SEGMENT                          ;定义一个DATAS段
    STR1 DB 'fmw2017110110'            ;定义字符串，由于DOS系统功能只能输出末位带$的字符串，所以此串不
DATAS ENDS                             ;DATAS段结束

CODES SEGMENT                          ;定义一个CODES段
    ASSUME CS:CODES,ES:DATAS           ;关联代码段寄存器CODES和数据段寄存器CS、DATAS和DS
START:                                 ;程序开始标号处
    MOV AX,DATAS                       ;先将段DATAS中立即数存到通用寄存器AX中作为中转
    MOV ES,AX                          ;将立即数送到段寄存器ES中码段代码

    LEA DI,STR1                        ;把STR1的有效地址传送到DI中
    MOV AL,'m'                         ;将'm'的ASCII码送到AL中
    MOV CX,20                          ;将STR1的存储单元20存放入保存计数值的CX寄存器中
    CLD                                ;clear direction flag
    REPNE SCASB                        ;串扫描指令，具体看下面解答部分↓

    MOV DL,ES:[DI]
    MOV AH,02H                         ;调用DOS系统的02号功能：显示一个字符
    INT 21H                            ;调用DOS功能中断

    MOV AH,4CH                         ;调用DOS系统4C号功能：结束程序
    INT 21H                            ;调用DOS功能中断
CODES ENDS                             ;CODES段结束
    END START                          ;汇编程序运行结束

;--------------------------------------------------------------------------
;1. 如何理解第15段中“REPNE SCASB”这句指令的意思?

;答: REPNE是一个串操作前缀，它重复串操作指令，每重复一次ECX的值就减一，
;     一直到CX不为0或ZF为0时停止。SCASB是字符串操作指令，SCASB指令常与循
;     环指令REPZ/REPNZ合用。例如，REPNZ SCASB 语句表示当寄存器ECX>0 且标
;     志寄存器ZF=0，则再执行一次SCASB指令。用于比较寄存器AL的值不相等则重
;     复查找的字。
;--------------------------------------------------------------------------
```

串比较

```
DATAS1 SEGMENT                          ;定义一个DATAS1段
    STR1 DB 'Hello!'                    ;定义一个STR1字符串
DATAS1 ENDS                             ;DATAS1段结束

DATAS2 SEGMENT                          ;定义一个DATAS2段
    STR2 DB 'Helle.'                    ;定义一个STR2字符串
DATAS2 ENDS                             ;DATAS2段结束

CODES SEGMENT                           ;定义一个CODES段
    ASSUME CS:CODES,DS:DATAS1,ES:DATAS2 ;关联代码段寄存器和数据段寄存器
START:                                  ;程序开始标号处
    MOV AX,DATAS1                       ;先将段DATAS1中立即数存到通用寄存器AX中作为中转
    MOV DS,AX                           ;将立即数送到段寄存器DS中
    MOV AX,DATAS2                       ;先将段DATAS2中立即数存到通
    MOV ES,AX                           ;将立即数送到段寄存器ES中

    LEA SI,STR1                         ;调用字符串STR1开始有效地址（偏移地址），存放在源变址寄存器S
    LEA DI,STR2                         ;调用字符串STR2开始有效地址（偏移地址），存放在目的变址寄存器
    MOV CX,6                            ;将字符串的存储单元6存放入保存计数值的CX寄存器中
    CLD                                 ;clear direction，使变址寄存器SI和DI的地址指针自动增加
    REPE CMPSB                          ;串操作，字符串比较指令，具体看下面解答部分↓

    MOV DL,ES:[DI-1]                    ;将ES寄存器中STR2不相等的位置的下一位置的上一位置字符输出
    MOV AH,02H                          ;调用DOS系统的02号功能：显示一个字符
    INT 21H                             ;调用DOS功能中断

    MOV AH,4CH                          ;调用DOS系统4C号功能：结束程序
    INT 21H                             ;调用DOS功能中断
CODES ENDS                              ;CODES段结束
    END START                           ;汇编程序运行结束

;--------------------------------------------------------------------------
;1. 如何理解第21段中“REPE CMPSB”这句指令的意思?

;答：REPE是一个串操作前缀，它重复串操作指令，每重复一次ECX的值就减一，
;     一直到CX为0或ZF为0时停止。CMPSB是字符串比较指令，把SI指向的数据
;     与DI指向的数一个一个的进行比较。当相同时继续比较，不同时不比较。
;--------------------------------------------------------------------------
```

从串中取元素

```
DATAS SEGMENT                          ;定义一个DATAS段
    STR1 DB 'fanmaowei'                ;定义一串源字符串STR1
DATAS ENDS                             ;DATAS段结束

CODES SEGMENT                          ;定义一个CODES段
    ASSUME CS:CODES,DS:DATAS           ;关联代码段寄存器CODES和数据段寄存器CS、DATAS和DS
START:                                 ;程序开始标号处
    MOV AX,DATAS                       ;先将段DATAS中立即数存到通用寄存器AX中作为中转
    MOV DS,AX                          ;将立即数送到段寄存器DS中

    LEA SI,STR1                        ;调用字符串STR1开始有效地址（偏移地址），存放在源变址寄存器SI
    CLD                                ;clear direction，使变址寄存器SI的地址指针自动增加

    MOV CX,3                           ;循环执行三次。为什么是三次？具体看下面解答部分↓
PRINT:                                 ;PRINT循环开始标号
    LODSB                              ;串操作，块读出指令，具体看下面解答部分↓
    MOV DL,AL                          ;将STR1串中读出的数从AL放到DL中用于输出
    MOV AH,02H                         ;调用DOS系统的02号功能：显示一个字符
    INT 21H                            ;调用DOS功能中断

LOOP PRINT                             ;当CX不等于0时，跳入上面进行循环。CX等于0时，不进入循环，程序

    MOV AH,4CH                         ;调用DOS系统4C号功能：结束程序
    INT 21H                            ;调用DOS功能中断
CODES ENDS                             ;CODES段结束
    END START                          ;汇编程序运行结束

;--------------------------------------------------------------------------
;1. 如何理解第十四段中“LODSB”这句指令的意思?

;答：汇编语言中，串操作指令LODSB/LODSW是块读出指令，其具体操作是把SI指向的
;     存储单元读入累加器,其中LODSB是读入AL,LODSW是读入AX中,然后SI自动增加或
;     减小1或2位。当方向标志位DF=0时，则SI自动增加；DF=1时，SI自动减小。
;--------------------------------------------------------------------------
;2. 第十九段中，为什么CX寄存器里放入3是使循环执行三次?

;答：CX=3时，往下执行遇到标号PRINT，程序运行到LOOP处，判断CX=3>0，所以
;     程序跳到PRINT处开始执行，此时动用了LOOP指令，所以CX要自动-1（=2）
;     执行LODSB指令，然后将读出的'f'输出。执行到LOOP指令处，CX=2>0，所以
;     继续跳到PRINT处开始执行，此时CX自动-1（=1）。PRINT里继续输出'a',执行
;     到LOOP指令处，CX=1>0，继续执行PRINT，CX的值-1=0，输出'n'。当在执行到
```

```
;     LOOP处判断，由于CX=0，不再进行循环，所以最后循环一共执行了3次。
;--------------------------------------------------------------------------------
```

将元素存入串

```
DATAS SEGMENT                          ;定义一个DATAS段
    STR1 DB 6 DUP(?),'$'               ;预定义6个Byte大小字符串变量STR1
DATAS ENDS                             ;DATAS段结束

CODES SEGMENT                          ;定义一个CODES段
    ASSUME CS:CODES,ES:DATAS           ;关联代码段寄存器CODES和数据段寄存器CS、DATAS和DS
START:                                 ;程序开始标号处
    MOV AX,DATAS                       ;先将段DATAS中立即数存到通用寄存器AX中作为中转
    MOV ES,AX                          ;将立即数送到段寄存器ES中

    LEA DI,STR1                        ;调用字符串STR1开始有效地址（偏移地址），存放在目的变址寄存器D

    MOV AL,'f'                         ;将'f'的ASCII码送到AL中，同理后面几句。
    STOSB                              ;串操作，单字符输出指令，同理后面几句。具体看下面解答部分↓
    MOV AL,'m'                         ;见第13段
    STOSB                              ;见第14段
    MOV AL,'w'                         ;见第13段
    STOSB                              ;见第14段

    MOV AL,'6'                         ;将'6'的ASCII码送到AL中
    MOV CX,3                           ;设定循环次数3次
    CLD                                ;clear direction，使变址寄存器DI的地址指针自动增加
    REP STOSB                          ;当CX不等于0时，重复执行第20段操作

    LEA DX,STR1                        ;调用字符串STR1开始地址
    MOV AX,ES                          ;将段寄存器ES的值放入AX中
    MOV DS,AX                          ;将AX的值放入段寄存器DS中（AX作为中转），DS作为DX的偏移地址
    MOV AH,09H                         ;调用DOS系统9号功能：显示字符串
    INT 21H                            ;调用DOS功能中断

    MOV AH,4CH                         ;调用DOS系统4C号功能：结束程序
    INT 21H                            ;调用DOS功能中断
CODES ENDS                             ;CODES段结束
    END START                          ;汇编程序运行结束

;--------------------------------------------------------------------------------
;1. 如何理解第十四段中“LODSB”这句指令的意思?

;答：该指令为单字符输出指令，调用该指令后，可以将累加器AL中的值传递到
;    当前ES段的DI地址处，并且根据DF的值来影响DI的值，如果DF为0，则调用
;    该指令后，DI自增1，如果DF为1,DI自减1.相当于：MOV ES:[DI],AL INC DI
```

```
;    或者 MOV ES:[DI],AL DEC DI
;-------------------------------------------------------------------------------
```

◀返回目录

冒泡排序

```
DATAS SEGMENT                          ;定义一个DATAS段
    BUF DB 3,4,5,0,6,2,1               ;定义一个长度为7的BUF数组
DATAS ENDS                             ;DATAS段结束

CODES SEGMENT                          ;定义一个CODES段
    ASSUME CS:CODES,DS:DATAS           ;关联代码段寄存器CODES和数据段寄存器CS、DATAS和DS
START:                                 ;程序开始标号处
    MOV AX,DATAS                       ;先将段DATAS中立即数存到通用寄存器AX中作为中转
    MOV DS,AX                          ;将立即数送到段寄存器DS中

    ;输出排序前数组的值，与后面输出实现方法不同
    MOV BX,OFFSET BUF                  ;将BUF偏移地址送到BX中
    MOV CX,7                           ;将数组BUF的长度7赋给CX计数寄存器，用于控制循环次数
 S1:MOV DL,BUF[BX]                     ;将BX的值，即BUF初始地址上是值（BUFF首地址元素）送到DL中
    ADD DL,'0'                         ;把数字变成字符
    MOV AH,02H                         ;调用DOS系统的02号功能：显示一个字符
    INT 21H                            ;调用DOS功能中断
    INC BX                             ;BX+1，使BUF数组遍历输出
    LOOP S1                            ;当CX>0时，循环S1操作，执行一次LOOP，CX-1
    ;输出回车换行
    MOV DL,10                          ;输出回车换行，回车键ACSII值为10
    MOV AH,02H                         ;调用DOS系统的02号功能：显示一个字符
    INT 21H                            ;调用DOS功能中断

    CALL BUBBLE                        ;执行冒泡排序子程序
    ;输出排序后数组的值
    MOV BX,0                           ;同OFFSET BUF，将BX位置为0，用于BUF数组下标遍历
    MOV CX,7                           ;将数组BUF的长度7赋给CX计数寄存器，用于控制循环次数
 S2:MOV DL,[BX]                        ;将BX地址上的值，即BUF[BX]的值送给DL
    ADD DL,'0'                         ;把数字变成字符
    MOV AH,02H                         ;调用DOS系统的02号功能：显示一个字符
    INT 21H                            ;调用DOS功能中断
    INC BX                             ;BX+1，使BUF数组遍历输出
    LOOP S2                            ;当CX>0时，循环S2操作，执行一次LOOP，CX-1

    MOV AH,4CH                         ;调用DOS系统4C号功能：结束程序
    INT 21H                            ;调用DOS功能中断

BUBBLE PROC                            ;定义BUBBLE子程序段，默认NEAR
    MOV CX,7                           ;将数组BUF的长度7赋给CX计数寄存器，用于控制循环次数
```

```
    DEC CX                          ;CX-1，冒泡排序中，循环次数为数组长度-1
 L1:MOV DI,CX                       ;将CX的值存入DI寄存器中
    MOV BX,0                        ;将BX位置为0，用于BUF数组下标遍历
 L2:MOV AL,BUF[BX]                  ;将BUF[BX]的值存在AL中
    CMP AL,BUF[BX+1]                ;比较BUF[BX]和BUF[BX+1]的大小
    JLE L3                          ;如果BUF[BX]<BUF[BX+1] 则跳转，不交换值。否则执行下面交换值这
    XCHG AL,BUF[BX+1]               ;将BUF[BX+1]的值赋为BUF[BX]的值，AL的值赋为BUF[BX+1]
    MOV BUF[BX],AL                  ;将AL中BUF[BX+1]的值送到BUF[BX]中，完成值交换。XCHG指令不能用

 L3:ADD BX,1                        ;BX+1，使BUF数组遍历输出
LOOP L2                             ;当CX>0时，循环L2操作，执行一次LOOP，CX-1
    MOV CX,DI                       ;将之前存入DI中CX的值还给CX寄存器
LOOP L1                             ;当CX>0时，循环L1操作，执行一次LOOP，CX-1

    RET                             ;子程序调用结束，IP指针回到主程序段
BUBBLE ENDP                         ;BUBBLE子程序结束

CODES ENDS                          ;CODES段结束
    END START                       ;汇编程序运行结束
```

◀返回目录

数组求和

```
DATAS SEGMENT                          ;定义一个DATAS段
    ARRAY1 DB 1,2,3,4,5,6,7,8,9,1      ;定义数组ARRAY1
    ARRAY2 DB 8,7,6,5,4,3,2,1,0,8      ;定义数组ARRAY1
    SUM DW 10 DUP(?)                   ;定义数组SUM用来保存数组和的结果
DATAS ENDS                             ;DATAS段结束

CODES SEGMENT                          ;定义一个CODES段
    ASSUME CS:CODES,DS:DATAS           ;关联代码段寄存器CODES和数据段寄存器CS、DATAS和DS
START:                                 ;程序开始标号处
    MOV AX,DATAS                       ;先将段DATAS中立即数存到通用寄存器AX中作为中转
    MOV DS,AX                          ;将立即数送到段寄存器DS中

    MOV CX,10                          ;将数组的存储单元10存放入保存计数值的CX寄存器中
    MOV SI,0                           ;设定16位通用寄存器SI值为0，用于作为数组索引值

NEXT:                                  ;NEXT循环开始标号
    ;输出ARAAY1
    MOV BX,OFFSET ARRAY1               ;将ARRAY1的偏移地址（起始地址）放入BX寄存器中
    MOV DL,[BX+SI]                     ;间接寻址方式，[BX+SI]操作数是以BX中存放的数+SI为地址的单元中
    ADD DL,'0'                         ;把ARAAY1[BX+SI]中的数字变成字符输出
    MOV AH,02H                         ;调用DOS系统的02号功能：显示一个字符
    INT 21H                            ;调用DOS功能中断
    ;输出'+'
    MOV DL,'+'                         ;将'+'放入DL中用于输出
    MOV AH,02H                         ;调用DOS系统的02号功能：显示一个字符
    INT 21H                            ;调用DOS功能中断
    ;输出ARAAY2
    MOV BX,OFFSET ARRAY2               ;将ARRAY2的偏移地址（起始地址）放入BX寄存器中
    MOV DL,[BX+SI]                     ;间接寻址方式，[BX+SI]操作数是以BX中存放的数+SI为地址的单元中
    ADD DL,'0'                         ;把ARAAY2[BX+SI]中的数字变成字符输出
    MOV AH,02H                         ;调用DOS系统的02号功能：显示一个字符
    INT 21H                            ;调用DOS功能中断
    ;输出'='
    MOV DL,'='                         ;将'='放入DL中用于输出
    MOV AH,02H                         ;调用DOS系统的02号功能：显示一个字符
    INT 21H                            ;调用DOS功能中断
    ;输出加和
    MOV BX,OFFSET ARRAY1               ;将ARRAY1的偏移地址（起始地址）放入BX寄存器中
    MOV DL,[BX+SI]                     ;间接寻址方式，[BX+SI]操作数是以BX中存放的数+SI为地址的单元中
    MOV BX,OFFSET ARRAY2               ;将ARRAY2的偏移地址（起始地址）放入BX寄存器中
```

```
    ADD DL,[BX+SI]                  ;将ARAAY2的值和ARAAY1的值相加，结果保存在DL中
;后面直接把结果输出，这里就不将结果另外保存在SUM数组中了。如果有此需要则执行后面两句指令即可。
    ;MOV BX,OFFSET SUM
    ;MOV [SUM+SI],BX
    ADD DL,'0'                      ;把数字变成字符输出，如果要将字符变成数字保存在数组中，则执行'
    MOV AH,02H                      ;调用DOS系统的02号功能：显示一个字符
    INT 21H                         ;调用DOS功能中断

    INC SI                          ;SI加1，索引值加1

    ;输出回车
    MOV DL,10                       ;输出回车换行，回车键ACSII值为10
    MOV AH,02H                      ;调用DOS系统的02号功能：显示一个字符
    INT 21H                         ;调用DOS功能中断

LOOP NEXT                           ;当CX不等于0时，跳入上面进行循环。CX等于0时，不进入循环，程序

    MOV AH,4CH                      ;调用DOS系统4C号功能：结束程序
    INT 21H                         ;调用DOS功能中断
CODES ENDS                          ;CODES段结束
    END START                       ;汇编程序运行结束
```

◀返回目录

n 的阶乘

```
DATAS SEGMENT                          ;定义一个DATAS段
    NUM DB 3                           ;定义要阶乘的数NUM=3
    RES DB ?                           ;预先定义结果变量RES
DATAS ENDS                             ;DATAS段结束

CODES SEGMENT                          ;定义一个CODES段
    ASSUME CS:CODES,DS:DATAS           ;关联代码段寄存器CODES和数据段寄存器CS、DATAS和DS
START:                                 ;程序开始标号处
    MOV AX,DATAS                       ;先将段DATAS中立即数存到通用寄存器AX中作为中转
    MOV DS,AX                          ;将立即数送到段寄存器DS中

    CALL MAIN                          ;转移指令，同RET指令，修改IP地址。

    MOV DL,RES                         ;将RES中的值赋给DL用于输出
    ADD DL,'0'                         ;把数字变成字符
    MOV AH,02H                         ;调用DOS系统的02号功能：显示一个字符
    INT 21H                            ;调用DOS功能中断

    MOV AH,4CH                         ;调用DOS系统4C号功能：结束程序
    INT 21H                            ;调用DOS功能中断

MAIN PROC                              ;定义MAIN子程序段
    MOV AL,NUM                         ;将NUM放入AL寄存器中用于计算
    MOV CL,NUM                         ;将NUM放入CL寄存器中用于计数
    DEC CL                             ;CL-1
    DEC CL                             ;CL-1，阶乘次数为n-2
FACT:                                  ;FACT循环开始标号
    DEC NUM                            ;NUM-1
    MUL NUM                            ;将上面NUM减一后=2，乘AL中的NUM=3，结果=6，同时也存入AL寄存器
LOOP FACT                              ;当CX不等于0时，跳入上面进行循环。CX等于0时，不进入循环，程序往

    MOV RES,AL                         ;将AL中的阶乘结果放入RES中
    RET                                ;转移指令，同CALL指令，修改IP地址。
MAIN ENDP                              ;MAIN子程序结束
CODES ENDS                             ;CODES段结束
    END START                          ;汇编程序运行结束
```

◀返回目录

大小写转换

```
DATAS SEGMENT                          ;定义一个DATAS段
    STR1 DB 'FanMaoWei',13,10,'$'      ;定义字符串STR1，"13,10,'$'"仅方便输出显示
DATAS ENDS                             ;DATAS段结束

CODES SEGMENT                          ;定义一个CODES段
    ASSUME CS:CODES,DS:DATAS           ;关联代码段寄存器CODES和数据段寄存器CS、DATAS和DS
START:                                 ;程序开始标号处
    MOV AX,DATAS                       ;先将段DATAS中立即数存到通用寄存器AX中作为中转
    MOV DS,AX                          ;将立即数送到段寄存器DS中

    LEA DX,STR1                        ;调用字符串STR1开始有效地址（偏移地址），存放在DX中
    MOV AH,09H                         ;调用DOS系统9号功能：显示字符串
    INT 21H                            ;调用DOS功能中断

    MOV BX,0                           ;设置(BX)=0，DS:BX指向'FanMaoWei'的第一个字母
    MOV CX,9                           ;设置循环次数5，因为'FanMaoWei'有9个字母
S:                                     ;S循环开始标号
    MOV AL,[BX]                        ;将ASCII码从DS:BX所指向的单元中取出放在AL中
    AND AL,11011111B                   ;将AL中的字符变为大写，AND是逻辑运算符。（变小写：OR AL,00100
    MOV [BX],AL                        ;将AL中已经变为大写的字符还给[BX]地址，即DS第BX个位置，BX初值
    INC BX                             ;BX+1，将DS中字符串位置向后移一位

    LOOP S                             ;当CX不等于0时，跳入上面进行循环。CX等于0时，不进入循环，程序

    LEA DX,STR1                        ;调用字符串STR1开始有效地址（偏移地址），存放在DX中
    MOV AH,09H                         ;调用DOS系统9号功能：显示字符串
    INT 21H                            ;调用DOS功能中断

    MOV AH,4CH                         ;调用DOS系统4C号功能：结束程序
    INT 21H                            ;调用DOS功能中断
CODES ENDS                             ;CODES段结束
    END START                          ;汇编程序运行结束
```

◀返回目录

单项选择题

1. CPU发出的访问存储器的地址是（）

A. 物理地址　　B. 偏移地址　　C. 逻辑地址　　D. 段地址

2. 将高级语言的程序翻译成机器码程序的实用程序是（）

A. 编译程序　　B. 汇编程序　　C. 解释程序　　D. 目标程序

3. DEC BYTE PTR [BX] 指令中的操作数的数据类型是（）

A. 字　　B. 双字　　C. 字节　　D. 四字

4. BUFFER DB 01H,0AH 指令中BUFFER称为（）

A. 符号　　B. 变量　　C. 助记符　　D. 标号

5. 串操作指令中，源串操作数的段地址一定在（）寄存器中。

A. CS　　B. SS　　C. DS　　D. ES

6. 使计算机执行某种操作的命令是（）

A. 伪指令　　B. 指令　　C. 标号　　D. 助记符

7. 将数据 5618H 存放在存储单元中的伪指令是（）

A. DATA1 DW 1856H　　B. DATA1 DB 18H,56H　　C. DATA1 EQU 5618H　　D. DATA1 DB 18H,00H,56H,00H

8. 若 AX=3500H，CX=56B8H，当 AND AX,CX 指令执行后，AX=（）

A. 1400H　　B. 77F8H　　C. 0000H　　D. 0FFFFH

9. 计算机处理问题中会碰到大量的字符、符号，对此必须采用统一的二进制编码。目前，微机中普遍采用的是（）码。

A. BCD码　　B. 二进制码　　C. ASCII码　　D. 十六进制码

10. 用指令的助记符、符号地址、标号和伪指令、宏指令以及规定的格式书写程序的语言称为（）

A. 汇编语言　　B. 高级语言　　C. 机器语言　　D. 低级语言

11. 指令 JMP FAR PTR DONE 属于（）

A. 段内转移直接寻址　　B. 段内转移间接寻址　　C. 段间转移直接寻址　　D. 段间转移间接寻址

12. 下列叙述正确的是（）

A. 对两个无符号数进行比较采用 CMP 指令，对两个有符号数比较用 CMPS 指令

B. 对两个无符号数进行比较采用 CMPS 指令，对两个有符号数比较用 CMP 指令

C. 对无符号数条件转移采用 JAE/JNB 指令，对有符号数条件转移用 JGE/JNL 指令

D. 对两个无符号数进行比较采用 CMP 指令，对两个有符号数比较用 CMPS 指令

13. 在下列指令的表示中，不正确的是（）

A. MOV AL,[BX+SI]　　B. JMP SHORT DONI　　C. DEC [BX]　　D. MUL CL

14. 条件转移指令 JNE 的测试条件为（）

A. ZF=0　　B. CF=0　　C. ZF=1　　D. CF=1

15. 8086CPU 在基址加变址的寻址方式中，变址寄存器可以为（）

A. BX 或 CX　　B. CX 或 SI　　C. DX 或 SI　　D. SI 或 DI

16. 已知 BX=2000H，SI=1234H，则指令 MOV AX,[BX+SI+2] 的源操作数在（）中。

A. 数据段中偏移量为3236H的字节中

B. 附加段中偏移量为3234H的字节中

C. 数据段中偏移量为3234H的字节中

D. 附加段中偏移量为3236H的字节中

17. 执行如下程序：

```
MOV AX,0
MOB BX,1
MOV CX,100
A: ADD AX,BX
INC BX
LOOP A
HLT
```

执行后（BX）=()

A. 99　　B. 100　　C. 101　　D. 102

18. 对于下列程序段：

```
AGAIN: MOV AL,[SI]
MOV ES:[DI],AL
INC SI
INC DI
LOOP AGAIN
```

也可用（）指令完成同样的功能。

A. REP MOVSB　　B. REP LODSB　　C. REP STOSB　　D. REPE SCASB

19. 下面指令序列执行后完成的运算，正确的算术表达式应是（）

```
MOV AL,BYTE PTR X
SHL AL,1
DEC AL
MOV BYTE PTR Y AL
```

A. y=2x+1　　B. x=2y+1　　C. x=2y-1　　D. y=2x-1

20. 在一段汇编程序中多次调用另一段程序，用宏指令比用子程序实现起来（）

A. 占内存空间小，但速度慢　　B. 占内存空间大，但速度快　　C. 占内存空间相同，速度快

D. 占内存空间相同，速度慢

21. 在程序执行过程中，IP寄存器中始终保存的是（）

A. 上一条指令的首地址　　B. 下一条指令的首地址　　C. 正在执行指令的首地址　　D. 需计算有效地址后才能确定地址

22. PSW寄存器中共有___位条件状态位，有___位控制状态位（）
    A. 6、3　　B. 3、6　　C. 8、4　　D. 4、8
23. 下列指令执行时出错的是（）
    A. ADD BUF1,BUF2　　B. JMP DWORD PTR DAT [BX]　　C. MOV AX,[BX+DI] NUM　　D. TEST AL,08H
24. 串指令中的目的操作数地址是由（）提供。
    A. SS:[BP]　　B. DS:[SI]　　C. ES:[DI]　　D. CS:[IP]
25. 将 DX 的内容除以2，正确的指令是（）
    A. DIV 2　　B. DIV DX,2　　C. SAR DX,1　　D. SHL DX,1

◀返回目录

填空专题

1. 通常所说的计算机系统包括 _____ 和 _____ 两大部分。（软件、硬件）
2. 8086/8088存储器分为四个段，这四个段的段名所对应的段寄存器分别是 _____ 、 _____ 、 _____ 、 _____ 。（CS、DS、ES、SS）
3. 现有AX=2000H，BX=1200H，DS=3000H，DI=0002H，(31200H)=50H，(31201H)=02H，(31202H)=40H，请写出下列各条指令独立执行完后有关寄存器及存储单元的内容，并指出标志位ZF、CF的值。
   A. ADD AX,1200H ;问 AX= _____ H，ZF= _____ （3200H、0）

   B. SUB AX,BX ;问 AX= _____ H，ZF= _____ （0E00H、0）

   C. MOV AX,[BX] ;问 AX= _____ H，CF= _____ （0250H、0）

   D. NEG WORD PTR [1200H];问 (31200H)= _____ H，CF= _____ （0B0H、1）
4. 设DS=2200H，BX=1000H，SI=0100H，偏移量D=0A2B1H，试计算出下列各种寻址方式下的有效地址。
   A. 使用D的直接寻址（）0A2B1H

   B. 使用BX的寄存器间接寻址（）1000H

   C. 使用BX和D的寄存器相对寻址（）0B2B1H

   D. 使用BX、SI和D的相对基址变址寻址（）0B3B1H

E. 使用BX、SI的基址变址寻址（ ） 1100H

◀返回目录

程序格式专题

1. 指出下列指令的错误：

```
(1) MOV AH,BX                    ;寄存器类型不匹配
(2) MOV [BX],[SI]                ;不能都是存储器操作数
(3) MOV AX,[SI][DI]              ;[SI]和[DI]不能一起使用
(4) MOV MYDAT[BX][SI],ES:AX      ;AX寄存器不能使用段超越
(5) MOV BYTE PTR[BX],1000        ;1000超过了一个字节的范围
(6) MOV BX,OFFSET MYDAT[SI]      ;MYDAT[SI]已经是偏移地址,不能再使用OFFSET
(7) MOV CS,AX                    ;CS不能用作目的寄存器
(8) MOV ECX,AX                   ;两个操作数的数据类型不同
```

2. 下面哪些指令是非法的？（假设OP1，OP2是以及用DB定义的变量）

```
(1) CMP 15,BX                    ;错，立即数不能作为目的操作数
(2) CMP OP1,25
(3) CMP OP1,OP2                  ;错，不能都是存储器操作数
(4) CMP AX,OP1                   ;错，类型不匹配，应为CMP AX,WORD PTR OP1
```

◀返回目录