**实验报告**

**实验序号：2 实验项目名称： 编写子程序**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 学　　号 |  | 姓 名 |  | 班 级 |  |
| 实验地点 |  | 指导教师 |  | 实验成绩 |  |
| **一、实验目的**  **（一）子程序：显示字符串**  1）熟练掌握在dos屏幕上输出字符的基本操作。掌握显示缓冲区范围；  2）熟练掌握从内存中读取字节单元内容和字单元内容；并将该内容写入我们期望的内存中；  3）掌握8位乘法和16位乘法的操作。  **（二）子程序：解决除法溢出的问题**  1）考察对于一个较大的数值的存储,无论是在寄存器中还是在内存中。  2）熟悉div除法指令的内涵，它的操作数存储的寄存器是哪些、运算的结果又存储在那些寄存器中。  3）学会合理利用栈空间保存一些临时的数值。  **（三）子程序：数值显示**  1）学会将二进制转化为十进制。  2）理解概念世界的抽象数据与现实世界的多种表示形式之间的区别与联系、以及它们之间的相互转化。  **二、实验环境(设备**)  编辑器：Visual Studio 2022  汇编程序：MASM.EXE  调试程序：DEBUG.EXE  连接装配程序：LINK.EXE  **三、实验内容**  **（一）子程序：显示字符串**    **（二）子程序：解决除法溢出的问题**    **（三）子程序：数值显示**    **四、实验结果及其分析**  **【说明】完整的实验代码附在文档最后。**  **（一）子程序：显示字符串**  **1.程序分析**  **（1）关于显示缓冲区：**  DOS窗口可以显示80X25个字符(每行80个字符x 25行，行偏移是160个字节)。它们在内存中是在一个内存段中存储的，这个内存区域叫做**显示缓冲区**，即从物理地址B8000H~BFFFH这个32K的内存区域。  在显示缓冲区中，每2个字节负责屏幕上一个字符的显示（包括显示的属性），偶数字节单元表示的是字符，奇数字节单元表示的是字符的属性（颜色、闪烁等）。在显示缓冲区内写入的字符，立即就显示在屏幕上。  **（2）参数的传递：**  此程序有3个参数，它们是：dh=行号（0-24取值范围）；dl=列号（0-79取值范围）；cl=颜色。需要通过分析屏幕上的行列位置和显存地址的对应关系，在子程序内部将它们转化为显存中的地址。  **（3）字符的表示方式：**  在写入显示缓冲区中时，使用[bx+di+idata]的表示方式。其中，（bx）为特定行和特定列（由主程序中的dl和dh参数传入）基于显示缓冲区b800：0000的偏移地址，（di）代表了从这个偏移地址开始每个字符的偏移地址；（idata）代表了每个字符的二个字节（一个是字符本身，一个是字符的颜色属性）。  **2.代码与结果测试**  **（1）代码**  **① data数据段程序代码：**    **② code代码段程序代码：**    **（2）结果测试**  如下图，在DOS界面执行该程序成功显示指定颜色（绿色）    **（二）子程序：解决除法溢出的问题**  **1.程序分析**  **（1）除法溢出：**  除法操作时，由于运算结果商的值过大，超出ax寄存器的存储范围，导致ax寄存器不能存储该值，因此会引发CPU的一个内部错误，即除法溢出。  **（2）分析子程序参数在程序实现中的作用：**    将被除数（4个字节）的高16位存储在dx中，低16位存储在ax中；除数存储在cx中。   1. **解读**   X是被除数：（范围[0~FFFFFFFFH]），也就是0F4240H  N是除数：（范围[0~FFFFH]）,也就是0AH  H：高16位（范围[0~FFFFH]）,对于被除数来说就是000FH  L：低16位（范围[0~FFFFH]），对于被除数来说就是4240H  int（）：取商，int（H/N）也就是求H/N结果的商。  rem（）：取余，rem（H/N）也就是求H/N结果的余数。  **其中，十进制数65536相对于十六进制数就是10000H，等价于左移16位，也就是对应于子程序中存储高位寄存器的步骤。因此，公式解读如下：**  ① int（H/N）\*65536代表了X/N结果高16位  ② [rem（H/N）\*65536+L]/N代表了X/N结果的低16位  ③二者用加号连接代表了将它们的组合一起，代表了一个dw类型的双字单元**。**  **求公式前半部分：int（H/N）\*65536，所得结果即为最终结果的高16位；公式后半部分：[rem（H/N）\*65536+L]/N，运算结果商代表最终结果的低16位，运算结果的余数，代表最终结果的余数。**  **2.代码与结果测试**  **（1）代码**    **（2）结果调试**  ① 使用Debug的**R命令**查看CPU寄存器的内容时，可以看到，CS=078D，IP=0000，即内存076A:0000处的指令为CPU当前要读取、执行的指令；且跳转到的地址即为子程序的起始地址。    ② 求出int（H/N）\*65536，得到最终结果的高16位。【同时使用计算器验证结果，发现计算无误】    ③ 求出[rem（H/N）\*65536+L]/N，运算结果商代表最终结果的低16位，运算结果的余数，代表最终结果的余数。【同时使用计算器验证结果，发现计算无误】    **（3）结果检验**  使用计算器计算出正确的结果后与程序执行结果对比，看程序执行是否正确。如下图，结果准确无误，为186A0H。    **（三）子程序：数值显示**  **1.程序分析**  **（1）子程序的功能**  将ax中的存储十进制数值（传入参数（ax））转换成对应的ASCII码，并将这些字符按顺序写入到内存data段中.  **（2）数字的显示进制问题（将二进制信息存储的数据变为十进制形式的字符串）**  123, 12366, 1, 8 , 3, 38这些数在**内存中**是以二进制形式存储的（二进制补码），2个字节存储一个数字。在计算机屏幕上显示的数字、字符、其他符号，一律按照字符方式显示，都是按照ASCII码来处理的。也就是说在屏幕上显示的1它不代表数值1，而是字符1。因此，该程序需要解决的第一个问题是如何把二进制代码（补码方式存储的）变成ASCII码。  **（3）十进制的数值转变为表示十进制数的字符串，并且字符串以0为结尾符号**  根据书上的提示例子：将数值12666转变成一个字符串“12666”。  **（4）显示字符串**  在底层显卡显示方面，由第一个子程序可知，CPU只要在显卡的显存中写入期望的数据，它就显示在屏幕上。因此，只需要调用第一个子程序show\_str即可实现。也就是说要分别得到1，2，6，6, 6的ASCII码，分别为31H、32H、36H、36H，36H。  如何得到十进制的各个数字——将12666除以10，然后取余数，将余数倒序后，得到12666各个位的数字。之后，再通过“30H（字符0对应的ASCII码）+余数”转化为ASCII码。  **【子问题1——对于一个位置十进制数字，如何判断各位的值并求出来？】**只要保证结果的商是0，那么这个数除以10肯定结束了。因此可以使用jcxz指令判断（CX）是否为0（将结果的商每次送入到cx中），作为结束循环条件。  **【子问题2——（ax）/10的求商（求各个位的数字）的顺序是倒序的，怎样把它的顺序给倒过来？】**可以采用栈的结构，利用栈的先进后出的原理，弹栈时将栈顶的值（也就是数字的最高位的值）先写入内存data段中，这样就解决了字符顺序的问题。  **2.代码与结果测试**  **（1）代码**  **① data数据段程序代码：**    **② code代码段程序代码：**      **（2）结果测试**  如下图，在DOS界面执行该程序成功显示指定颜色（绿色）的十进制表示的数12666：    **五、实验心得与讨论**  在这次实验中，我一共实现了**显示字符串、解决除法溢出的问题、数值显示**三个子程序，期间也遇到了各种各样的问题：比如，在实现“显示字符串”时，我一开始对于显示缓冲区还有字符串的表示方式理解的都不是很深入，导致不知道如何使用题目所给参数编写子程序，但是通过反复看书、查阅资料、理解知识点，我逐渐能够上手，并最终实现了该功能。有了第一个子程序的编写成功经验，剩下两个子程序的编写上手也就更加顺利了。 | | | | | |
| **评语：**  1．按时完成实验；  2．实验内容和过程记录完整；  3．有实验的心得或讨论；  **老师：xx** | | | | | |

附完整代码：

**（一）子程序：显示字符串**

assume cs:code, ds:data, es:table

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*data数据段·开始\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

data segment

;定义年份,占21\*4=84个字节

db '1975', '1976', '1977', '1978', '1979', '1980', '1981', '1982','1983'

db '1984', '1985', '1986', '1987', '1988', '1989', '1990', '1991', '1992'

db '1993', '1994', '1995'

;以上是表示21年的21个字符串

;定义总收入, 占21\*4=84个字节

dd 16, 22, 382, 1356, 2390, 8000, 16000, 24486,50065, 97479, 140417, 197514

dd 345980, 590827, 803530, 1183000,1843000, 2759000, 3753000, 4649000, 5937000

;以上是表示21年公司总收入的21个dword型数据

;定义雇员人数，占21\*2=42个字节

dw 3, 7, 9, 13, 28, 38, 130, 220,476, 778, 1001, 1442, 2258, 2793, 4037,5635,8226

dw 11542, 14430, 15257, 17800

;以上是表示21年公司雇佣人数的21个word型数据

data ends

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*data数据段·结束\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*table数据段·开始\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

table segment

;定义21行的一个表格，每行16个字节，共占246个字节

db 21 dup ('year summ ne ?? ')

table ends

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*table数据段·结束\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*代码段·开始\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

code segment

;初始化2个数据段，将ds指向data，es指向table

start: mov ax,data

mov ds,ax

mov ax,table

mov es,ax

;初始化偏址寄存器变量

mov bx,0 ;用于data 数据段中ds:[bx+idata]，用来定位年份与收入

mov di,0 ;用于data 数据段中ds:[di+idata]，用来定位雇员数

mov si,0 ;用于table数据段中es:[si+idata]

;共21行，循环21次，初始化计数器

mov cx,21

;++++++写入年份++++++

s: mov ax,[bx+0] ;复制年份前两个字节

mov es:[si+0],ax ;粘贴年份前两个字节

mov ax,[bx+2] ;复制年份后两个字节

mov es:[si+2],ax ;粘贴年份后两个字节

;++++++写入空格++++++

mov al,20H ;空格的ASCII码对应十六进制数为20

mov es:[si+4],al

;++++++写入收入++++++

mov ax,[bx+84] ;复制收入前两个字节

mov es:[si+5],ax ;粘贴收入前两个字节

mov ax,[bx+86] ;复制收入后两个字节

mov es:[si+7],ax ;粘贴收入后两个字节

;++++++写入空格++++++

mov al,20H ;空格的ASCII码对应十六进制数为20

mov es:[si+9],al

;++++++雇员数++++++

mov ax,[di+84+84] ;复制雇员数量

mov es:[si+10],ax ;粘贴雇员数量

;++++++写入空格++++++

mov al,20H ;空格的ASCII码对应十六进制数为20

mov es:[si+12],al

;++++++人均收入，高16位送入dx，低16位送入ax++++++

mov ax,[bx+84]

mov dx,[bx+86]

;++++++用bp变量存储除数，为以后实验考虑++++++

mov bp,[di+84+84]

div bp ;16位除法指令

mov es:[si+13],ax ;将商的结果（ax）写入table段中

;++++++写入空格++++++

mov al,20H ;空格的ASCII码对应十六进制数为20

mov es:[si+15],al

;++++++bx、si、di变量的递增++++++

add bx,4 ;年份和总收入都是双字单元，故bx的递增量是4

add si,16 ;table中每行是16个字节，偏移量为16

add di,2 ;人数是字单元，故di的递增量是2

loop s

mov ax,4c00H

int 21H

code ends

end start

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*代码段·结束\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**（二）子程序：解决除法溢出的问题**

assume cs:code

code segment

start: mov ax, 4240H ;被除数，低16位

mov dx, 000FH ;被除数，高16位

mov cx, 0AH ;除数

call divdw ;调用divdw子程序，做不溢出的除法运算。

mov ax, 4c00H

int 21H

; ----------------------------------------------

; 子程序开始

;-----------------------------------------------

divdw: push ax ;将被除数低16位先压栈保存。t

;++++公式前半部分：int（H/N）\*65536++++

mov ax, dx ;（ax）=（dx）

mov dx, 0000H

div cx ;此时（dx）=0000H，（ax）=000FH，组合成0000000FH

mov bx, ax ;将H/N结果的商先保存在bx中，（bx）=0001H

;++++公式后半部分：[rem（H/N）\*65536+L]/N++++

pop ax ;将L值弹栈到ax

div cx ;此时（dx）=0005H，（ax）=4240H,组合成54240H

mov cx, dx ;返回值（cx）等于最终结果的余数

mov dx, bx ;最终结果高16位值=（bx）

ret

code ends

end start

**（三）子程序：数值显示**

assume cs:code

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*data数据段·开始\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

data segment

db 10 dup (0) ;初始化10个字节，置零

data ends

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*data数据段·结束\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*代码段·开始\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

code segment

start: mov ax, 12666 ;将显示的数字赋值给ax

;将ds：si指向data内存段

mov bx, data

mov ds, bx

mov si, 0

call dtoc ;调用dtoc子程序

;为调用show\_str做准备

mov dh, 8 ;屏幕的行数

mov dl, 3 ;所在行的列数

mov ch, 0 ;ch清零，防止高8位不为零。

mov cl, 2 ;颜色属性（此处应是二进制数0000 0010）

call show\_str ;调用show\_str子程序将字符串显示?

mov ax, 4c00H

int 21H

; ----------------------------------------------

; 子程序dtoc功能：将一个数字转换成字符串，并写入data段中

; 入口参数：ax, ds

; 返回值：无

;-----------------------------------------------

;++++++保护寄存器变量值，因为下面的变量子程序都用到++++++

dtoc: push ax

push cx

push bx

push si

mov si, 0 ;偏移地址置零

mov bx, 10 ;除数=10

change: mov dx, 0 ;涉及到16位除法，先将存储余数的变量置零

div bx ;将（ax）/（bx）

mov cx, ax ;将除法运算结果的商赋值给cx，用于条件判断

jcxz last ;判断cx是否为0？或商为零？

add dx, 30H ;将每个位的数字转换成ASCII码值

push dx ;将ASCII码值压栈保存

inc si

jmp short change

;++++++最后一次除法，商为0++++++

last: add dx, 30H ;将数字转换成ASCII码

push dx ;将字符值压栈

inc si ;最后一次也要转换并压栈

;将栈中数据倒序写入内存data段中

mov cx, si ;（si）=字符串共几个字符，设置循环计数器cx

mov si, 0

s: pop ds:[si] ;弹栈，并写入data内存段

inc si

loop s

;++++++恢复寄存器，并返回主调程序++++++

exit: pop si

pop bx

pop cx

pop ax

ret

; ----------------------------------------------

; show\_str功能 :按行和列及字符属性显示字符串

; 入口参数：dh-行数、dl-列数、cl-字符属性

; 返回值：无

;-----------------------------------------------

show\_str: push dx

push cx

push si

; ----------------------------------------------

; 计算行偏移

;-----------------------------------------------

mov ax,0 ;（ax）= 0，防止高位不为零

mov al,160 ;每行160字节

dec dh ;dh-1才是真正的行，因为行号从0开始计数

mul dh ;用 行数\*每行偏移量 得到目标行的偏移量

mov bx,ax ;将第（dh）行的偏移地址送入bx，bx代表行偏移

; ----------------------------------------------

; 计算列偏移，并与行偏移量相加得到最终偏移量

;-----------------------------------------------

mov ax,0 ;（ax）= 0，防止高位不为零

mov al,2 ;列的偏移量为2，两个字节代表一列

mul dl ;与行偏移量同理

add bx,ax ;将列偏移量与行偏移量相加，得到指定位置的偏移量

; ----------------------------------------------

; 进入显示缓存区

;-----------------------------------------------

mov di,0 ;将di作为每个字符的偏移量

mov ax,0b800h ;显示缓冲区为80\*25彩色字符模式

mov es,ax ;指定显示缓存区的内存位置

mov al,cl ;由于后面jcxz语句的判断要用到cx，所以这里将cl(颜色)先存下来。

; ----------------------------------------------

; 显示字符串内容

;-----------------------------------------------

show: mov ch,0 ;将cx高8位设置为0

mov cl,ds:[si] ;将字符串单个字符读入cl中

jcxz ok ;如果cx为0，则转移到ok标号执行相应代码

mov es:[bx+di+0],cl ;将字符传入低地址

mov es:[bx+di+1],al ;将颜色传入高地址

add di,2 ;列偏移量为2

inc si ;字符串的偏移量为1

loop show ;不为0,继续复制

; ----------------------------------------------

; 若字符串已全部显示完毕，则子程序调用结束

;-----------------------------------------------

ok: pop dx

pop cx

pop si ; 还原寄存器变量

ret ; 结束子程序调用

code ends

end start

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*代码段·结束\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*