

实验一 Debug 和 Emu8086 软件的使用

一、实验目的:

- (1) 熟悉 Debug 软件和汇编语言开发环境。
- (2) 掌握 Emu8086 软件使用方法。

二、实验内容:

1、Debug 的使用

Debug 是 DOS、Windows 都提供的实模式 (8086 方式) 程序的调试工具。使用它可以查看 CPU 各种寄存器的内容、内存的情况和在机器码级跟踪程序的运行。

- (1) 32 位 win 7 以及 windows XP 系统: 直接使用 C 盘中的 debug 程序
- (2) 64 位 win 7 使用 debug 的方法:
 - i) 安装 dos 模拟器 DOSBOX
 - ii) 下载一个 win7 32 位的 debug 程序, 并将它放在某个盘中 (例如 d 盘根目录)
 - iii) 双击运行 DOSBOX, 并运行以下命令:

Z:\>mount c d:/

Z:\>c:

C:\>debug

Debug 功能:

- 用 Debug 的 R 命令查看、改变 CPU 寄存器的内容;

➤ -r

```
C:\>debug
-r
AX=0000 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0100  NU UP EI PL NZ NA PO NC
073F:0100 0000          ADD     [BX+SI],AL          DS:0000=CD
```

➤ -r 寄存器名

```
C:\>debug
-r
AX=0000 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0100  NU UP EI PL NZ NA PO NC
073F:0100 0000          ADD     [BX+SI],AL          DS:0000=CD
-r AX
AX 0000
:1111
-r
AX=1111 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0100  NU UP EI PL NZ NA PO NC
073F:0100 0000          ADD     [BX+SI],AL          DS:0000=CD
```

- 用 Debug 的 D 命令查看内存中的内容;

➤ -d 查看 Debug 预设地址处的内容

```
-d
073F:0100  00 00 00 00 00 00 00 00-00 00 00 00 00 AE FE .....
073F:0110  00 F0 46 74 00 00 B2 00-B2 16 99 00 2E 07 2E 07 ..Ft.....
073F:0120  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
073F:0130  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
073F:0140  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
073F:0150  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
073F:0160  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
073F:0170  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
```

➤ -d 段地址: 偏移地址

```
-d 1000:9
1000:0000 00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
1000:0010 00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
1000:0020 00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
1000:0030 00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
1000:0040 00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
1000:0050 00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
1000:0060 00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
1000:0070 00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
1000:0080 00 00 00 00 00 00 00 00 00-00 .....

```

➤ -d 段地址: 起始偏移地址 结尾偏移地址

```
-d 1000:0 9
1000:0000 00 00 00 00 00 00 00 00 00-00 00 .....

```

● 用 Debug 的 E 命令改写内存中的内容;

➤ -e 起始地址 数据 数据 数据

```
-e 1000:0 0 1 2 3 4 5 6 7 8 9
-d 1000:0 9
1000:0000 00 01 02 03 04 05 06 07-08 09 .....

```

➤ -e 起始地址 逐个输入数据

```
-e 1000:0
1000:0000 00.1 01.2 02.3 03.4 04.5 05.6 06.7 07.8
1000:0008 08.9 09.A
-d 1000:0 9
1000:0000 01 02 03 04 05 06 07 08-09 0A .....

```

➤ -e 起始地址 字符/字符串

```
-e 1000:0 1 'a' 2 'b' 3 'c'
-d 1000:0 5
1000:0000 01 61 02 62 03 63 .a.b.c
-e 1000:0 1 2 3 'abc'
-d 1000:0 5
1000:0000 01 02 03 61 62 63 ...abc

```

● 用 Debug 的 U 命令将内存中的机器指令翻译成汇编指令;

➤ -u 起始地址

```
-e 1000:0 b8 01 00 b9 02 00 01 c8
b80100 mov AX, 0001
b90200 mov CX,0002
01c8 add AX,CX

```

```

-u 1000:0
1000:0000 B80100      MOV     AX,0001
1000:0003 B90200      MOV     CX,0002
1000:0006 01C8        ADD     AX,CX
1000:0008 090A        OR      [BP+SI],CX
1000:000A 0000        ADD     [BX+SI],AL
1000:000C 0000        ADD     [BX+SI],AL
1000:000E 0000        ADD     [BX+SI],AL
1000:0010 0000        ADD     [BX+SI],AL
1000:0012 0000        ADD     [BX+SI],AL
1000:0014 0000        ADD     [BX+SI],AL
1000:0016 0000        ADD     [BX+SI],AL
1000:0018 0000        ADD     [BX+SI],AL
1000:001A 0000        ADD     [BX+SI],AL
1000:001C 0000        ADD     [BX+SI],AL
1000:001E 0000        ADD     [BX+SI],AL

```

- 用 Debug 的 T 命令执行一条机器指令；

➤ -t

```

-r
AX=1111 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0100  NV UP EI PL NZ NA PO NC
073F:0100 0000      ADD     [BX+SI],AL          DS:0000=CD
-r CS
CS 073F
:1000
-r IP
IP 0100
:0
-r
AX=1111 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=1000 IP=0000  NV UP EI PL NZ NA PO NC
1000:0000 B80100      MOV     AX,0001
-t
AX=0001 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=1000 IP=0003  NV UP EI PL NZ NA PO NC
1000:0003 B90200      MOV     CX,0002

```

- 用 Debug 的 A 命令以汇编指令的格式在内存中写入机器指令。

➤ -a 起始地址

```

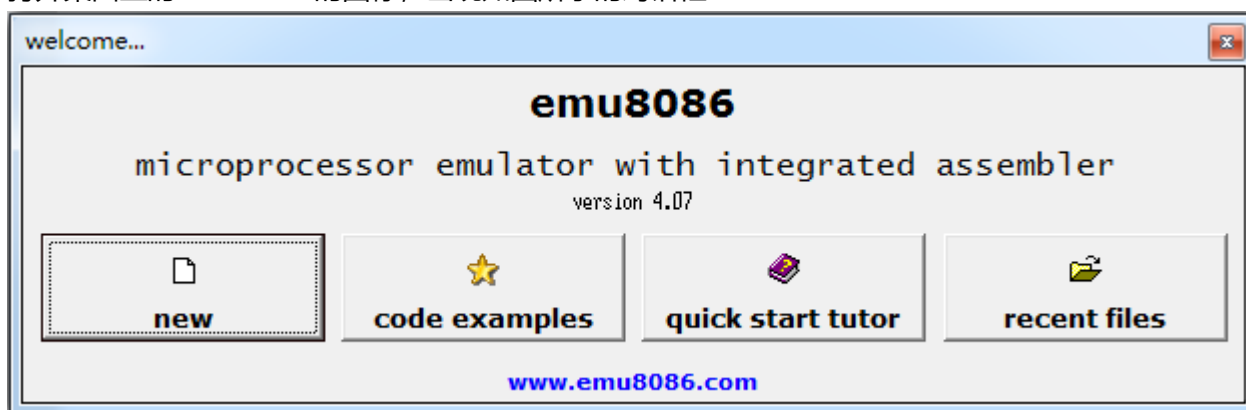
-a 1000:0
1000:0000 mov AX,1
1000:0003 mov BX,2
1000:0006 mov CX,3
1000:0009
-u 1000:0
1000:0000 B80100      MOV     AX,0001
1000:0003 B80200      MOV     BX,0002
1000:0006 B90300      MOV     CX,0003
1000:0009 0A00      OR      AL,[BX+SI]
1000:000B 0000      ADD     [BX+SI],AL
1000:000D 0000      ADD     [BX+SI],AL
1000:000F 0000      ADD     [BX+SI],AL
1000:0011 0000      ADD     [BX+SI],AL
1000:0013 0000      ADD     [BX+SI],AL
1000:0015 0000      ADD     [BX+SI],AL
1000:0017 0000      ADD     [BX+SI],AL
1000:0019 0000      ADD     [BX+SI],AL
1000:001B 0000      ADD     [BX+SI],AL
1000:001D 0000      ADD     [BX+SI],AL
1000:001F 0000      ADD     [BX+SI],AL

```

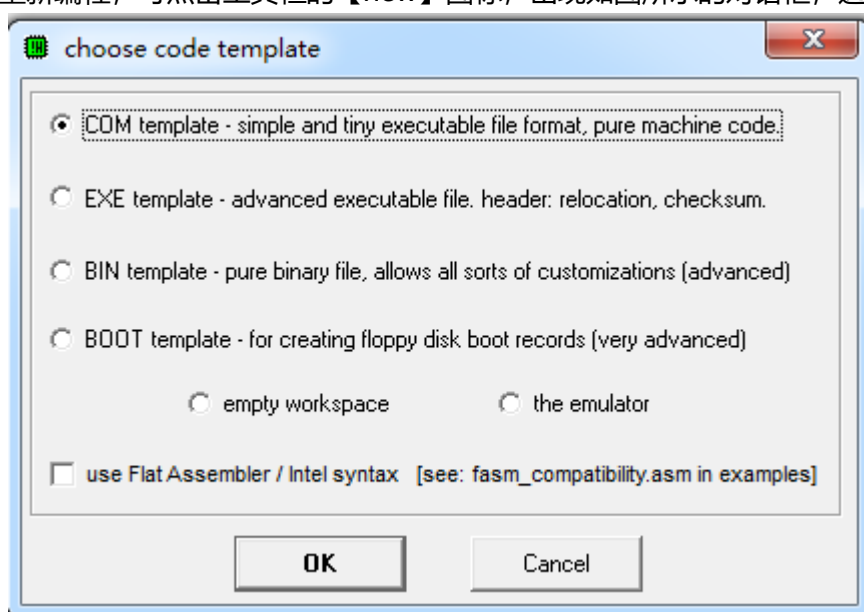
2、Emu8086 的使用

Emu8086 是集源代码编辑器，汇编/反汇编工具以及可以运行 debug 的模拟器于一身的汇编语言集成编译环境。

(1) 打开桌面上的 Emu8086 的图标，出现如图所示的对话框



若用户需要自己重新编程，可点击工具栏的【new】图标，出现如图所示的对话框，选择编程所采用的模板。



选择不同的模板：COM 模板、BIN 模板、EXE 模板、BOOT 模板

➤ COM 模板

最古老的一个最简单的可执行文件格式。采用此格式，源代码应该在 100H 后加载（即：源代码之前应有 ORG 100H）。从文件的第一个字节开始执行。支持 DOS 和 Windows 命令提示符。

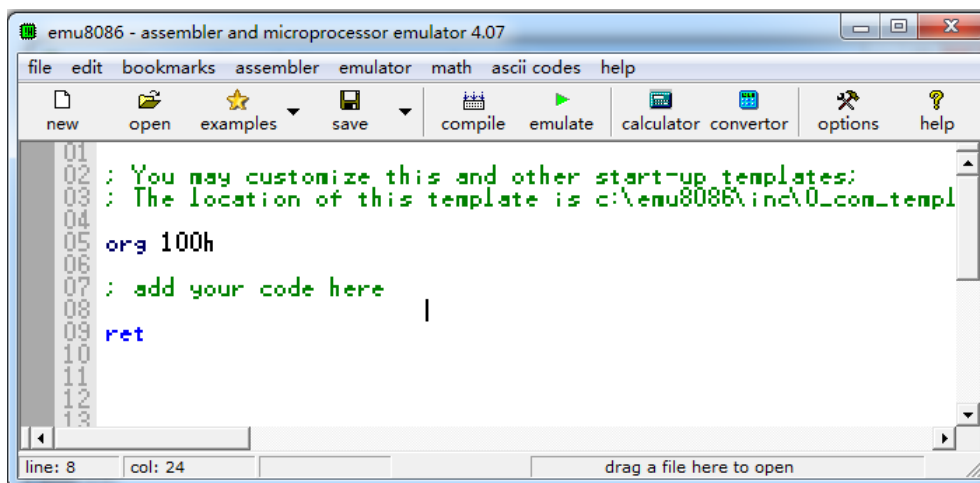
➤ ExE 模板

一种更先进的可执行文件格式。源程序代码的规模不限，源代码的分段也不限，但程序中必须包含堆栈段的定义。您可以选择从新建菜单中的 EXE 模板创建一个简单的 EXE 程序，有明确的数据段，堆栈段和代码段的定义。

程序员在源代码中定义程序的入口点（即开始执行的位置），该格式支持 DOS 和 Windows 命令提示符。

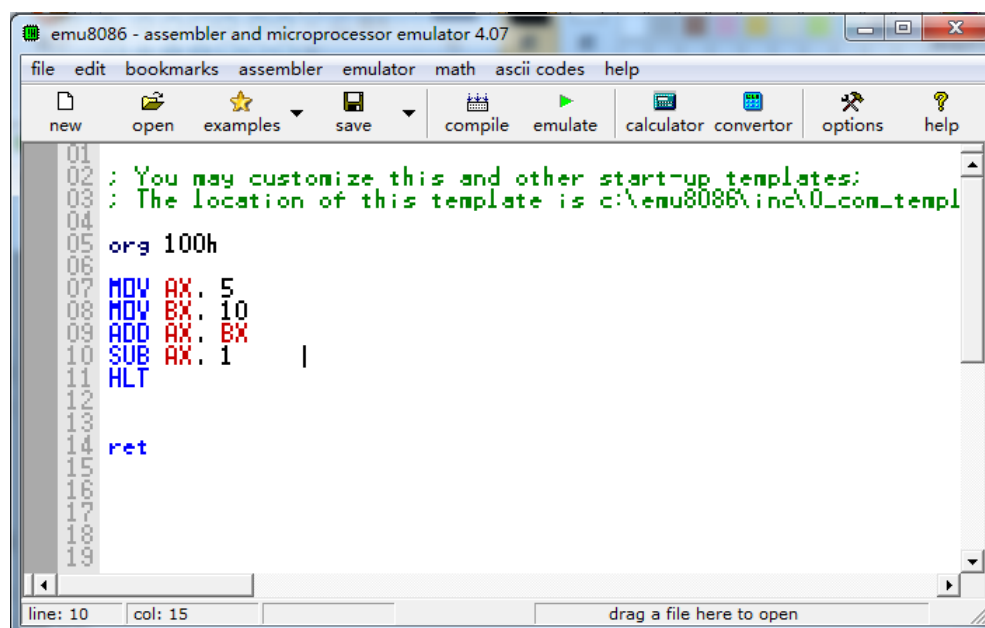
以上两种模板是最常用的模板

(2) 选择 COM 模板，点击【OK】，软件出现源代码编辑器的界面，如图所示：

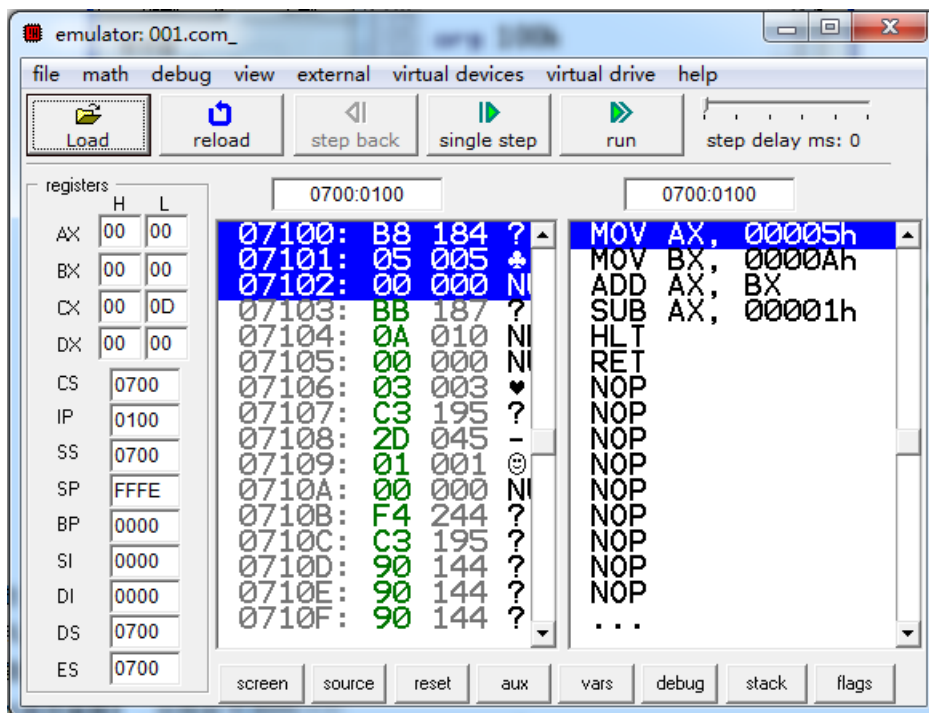


在源代码编辑器的空白区域，编写如下一段小程序：

```
MOV AX, 5
MOV BX, 10
ADD AX, BX
SUB AX, 1
HLT
```

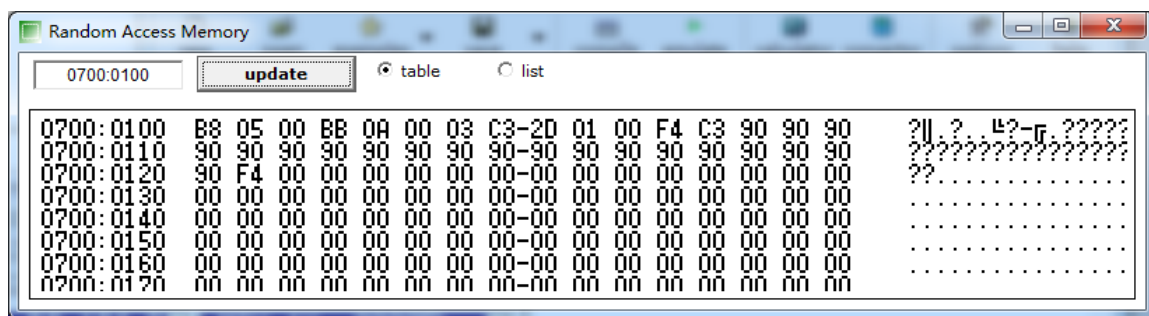


代码编写结束，点击菜单【file】【save as】，将源代码换名保存。本例将源代码保存为 001.asm。点击工具栏的【emulate】按钮,如果源程序无错误，则编译通过，出现如图所示的界面：



点击【single step】，程序将每执行一条指令便产生一次中断。点击【run】，程序将从第一句直接运行到最后一句。

界面的左侧可以观察程序运行过程中，各个寄存器的值的变化。若是查看内存区域的值，可以选择菜单【view】【memory】，出现如下界面：



(3) 单步运行该程序段，观察各寄存器的变化。

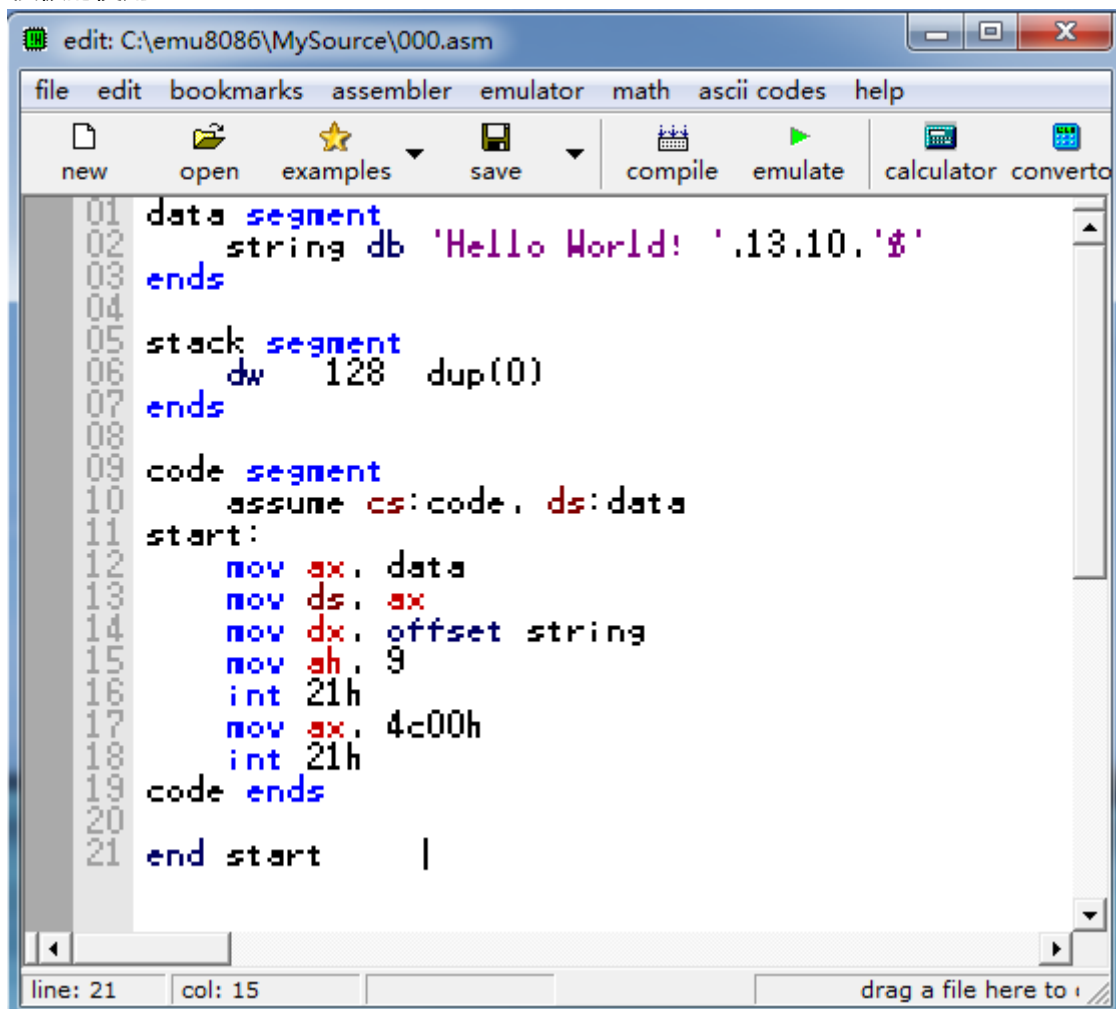
三、实验要求

1、使用 Debug，将下面的程序段写入内存（提示：可用 E 命令或 A 命令两种方式将指令写入内存），逐条执行，观察每条指令执行后，CPU 中相关寄存器中内容的变化。使用表格记录每一条指令执行后 AX 及 BX 寄存器的内容。

机器码	汇编指令
b8 20 4e	mov ax,4E20H
05 16 14	add ax, 1416H
bb 00 20	mov bx,2000H
01 d8	add ax,bx
89 c3	mov bx,ax
01 d8	add ax,bx
b8 1a 00	mov ax,001AH
bb 26 00	mov bx, 0026H
00 d8	add al,bl

00 c7	add bh, al
b4 00	mov ah,0
00 d8	add al,bl
04 9c	add al,9CH

2、EXE 模板的使用。



```

01 data segment
02     string db 'Hello World! '.13.10.'$'
03 ends
04
05 stack segment
06     dw 128 dup(0)
07 ends
08
09 code segment
10     assume cs:code, ds:data
11 start:
12     mov ax, data
13     mov ds, ax
14     mov dx, offset string
15     mov ah, 9
16     int 21h
17     mov ax, 4c00h
18     int 21h
19 code ends
20
21 end start

```

(1) 在 emu8086 中调试运行该程序。请单步执行该程序，使用表格记录下每执行一句话后寄存器 AX、DX、DS、CS、IP、SS、SP 内容的变化情况，体会各个寄存器的作用。

(2) 该程序运行结果是什么？（截图，图片大小不能超过 100K）

3、实验完成后将以上两个实验的结果记录在实验报告（实验报告文件名：汇编语言设计实践 1_学号_姓名.doc）中，并提交到教学云平台，截止时间请见教学云平台。