



山东大学

信息科学与工程学院

2020—2021 学年第二学期

实 验 报 告

课程名称: 微处理器原理与应用

实验名称: 实验 1.1 Hello World

专 业 班 级 2019 级崇新学堂

学 生 学 号 201900121023

学 生 姓 名 李禹申

实 验 时 间 2021 年 3 月 5 日

实验报告

【实验目的】

1. 掌握 window 的基本 Masm for windows 集成实验环境的使用
2. 掌握 win10 系统通过 DOSBox 实现虚拟 DOS 环境下 masm5 的编译和运行

【实验要求】

1. 通过对某部分代码的修改从而加深对于 debug 所涉及命令的理解
2. 不懂的知识及时百度

【实验具体内容】

【实验准备部分】

两个实验所涉及的源代码如下，予以相应的注释

； 是一个伪指令，用来定义一个堆栈段

STACKS SEGMENT STACK

； ① DW 是定义字类型变量的意思（问题①） DUP 是连续复制 128 个变量，(?)代表所赋值数值为随机

DW 128 DUP(?)

； 结束堆栈段的定义

STACKS ENDS

； 开始 DATA 段的定义，SEGMENT 是定义段

DATAS SEGMENT

； string 是一个标识符，标记给这个空间的首地址，13 是 ASCII 中的回车 10 是 ASCII 中的换行 '\$'是汇编中字符串结束的标志

STRING DB 13,10,'Hello World!',13,10,'\$'

； DATA 段结束

DATAS ENDS

； CODE 段开始定义

CODES SEGMENT

； ASSUME 建立段寄存器寻址，将 CODE 段和 CS 段寄存器相关联，DATAS 段和 DS 段寄存器相关联

ASSUME CS:CODES,DS:DATAS

； 入口地址

START:

；下面两行将 DATAS 地址放到 DS 段寄存器中，之所以要 AX 过渡，是因为不能直接对段寄存器给数

```
MOV AX,DATAS
```

```
MOV DS,AX
```

；将 STRING 所在的内存地址的偏移地址赋给 DX

```
LEA DX,STRING
```

；下面两条同时作用可以让其显示字符串

```
MOV AH,9
```

```
INT 21H
```

；下面两条指令是返回 DOS

```
MOV AH,4CH
```

```
INT 21H
```

；CODES 段结束，出口

```
CODES ENDS
```

```
END START
```

；是一个伪指令，用来定义一个堆栈段

```
STACKS SEGMENT STACK
```

；④ DW是定义字类型变量的意思(问题④) DUP是连续复制128个变量，(?)代表所赋值数值为随机

```
DW 128 DUP(?)
```

；结束堆栈段的定义

```
STACKS ENDS
```

；开始DATA段的定义，SEGMENT是定义段

```
DATAS SEGMENT
```

；string是一个标识符，标记给这个空间的首地址，13是ASCII中的回车 10是ASCII中的换行 '\$'是汇编中字符串结束的标志

```
STRING DB 13,10,'Hello World! ',13,10,'$'
```

；DATA段结束

```
DATAS ENDS
```

；CODE段开始定义

```
CODES SEGMENT
```

；ASSUME建立段寄存器寻址，将CODE段和CS段寄存器相关联，DATAS段和DS段寄存器相关联

```
ASSUME CS:CODES,DS:DATAS
```

；入口地址

```
START:
```

；下面两行将DATAS地址放到DS段寄存器中，之所以要AX过渡，是因为不能直接对段寄存器给数

```
MOV AX,DATAS
```

```
MOV DS,AX
```

；将STRING所在的内存地址赋给DX

```
LEA DX,STRING
```

；下面两条同时作用可以让其显示字符串

```
MOV AH,9
```

```
INT 21H
```

；下面两条指令是返回DOS

```
MOV AH,4CH
```

```
INT 21H
```

；CODES段结束，出口

```
CODES ENDS
```

```
END START
```

【第一个实验】使用 DOSBox 虚拟 DOS 环境运行所编写程序。

(1) 实验流程图（从实验 2.2 分支程序实验和循环程序实验开始必须画流程图）：

(2) 实验源代码（粘贴源代码）：

例如：（第一次实验的源代码）

(3) 实验代码、过程、相应结果（截图）并对实验进行说明和分析：

```
Source listing [NUL.LST]:
Cross-reference [NUL.CRF]:
hw.ASM(2): error A2009: Symbol not defined: UP

51614 + 464930 Bytes symbol space free

0 Warning Errors
1 Severe Errors

C:\>masm hw.asm
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987. All rights reserved.

Object filename [hw.OBJ]:
Source listing [NUL.LST]:
Cross-reference [NUL.CRF]:
hw.asm(2): error A2009: Symbol not defined: UP

51616 + 464928 Bytes symbol space free

0 Warning Errors
1 Severe Errors

C:\>_
```

在使用 `masm hw.asm` 命令后出现一个 Severe Errors，查看 DOS 给出的错误信息：

`hw.asm(2): error A2019: Symbol not defined: UP`，查看代码第 2 行发现

```
STACKS SEGMENT STACK
    DW 128DUP(?)
STACKS ENDS

DATAS SEGMENT
    STRING DB 13,10,'Hello World!',13,10,'$'
DATAS ENDS

CODES SEGMENT
    ASSUME CS:CODES,DS:DATAS
START:
    MOV AX,DATAS
    MOV DS,AX
    LEA DX,STRING
    MOV AH,9
    INT 21H

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

原来是 128 和 dup 中间没有写入 space 而产生的错误，更正之后继续运行该程序

输入 `masm hw.asm`

```
C:\>masm hw.asm
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987. All rights reserved.

Object filename [hw.OBJ]:
Source listing [NUL.LST]:
Cross-reference [NUL.CRF]:

51766 + 464778 Bytes symbol space free

0 Warning Errors
0 Severe Errors

C:\>_
```

没有错误和警告

输入 link hw.obj, link 成功后输入 hw 运行该程序

```
C:\>link hw.obj

Microsoft (R) Overlay Linker  Version 3.60
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.

Run File [HW.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:

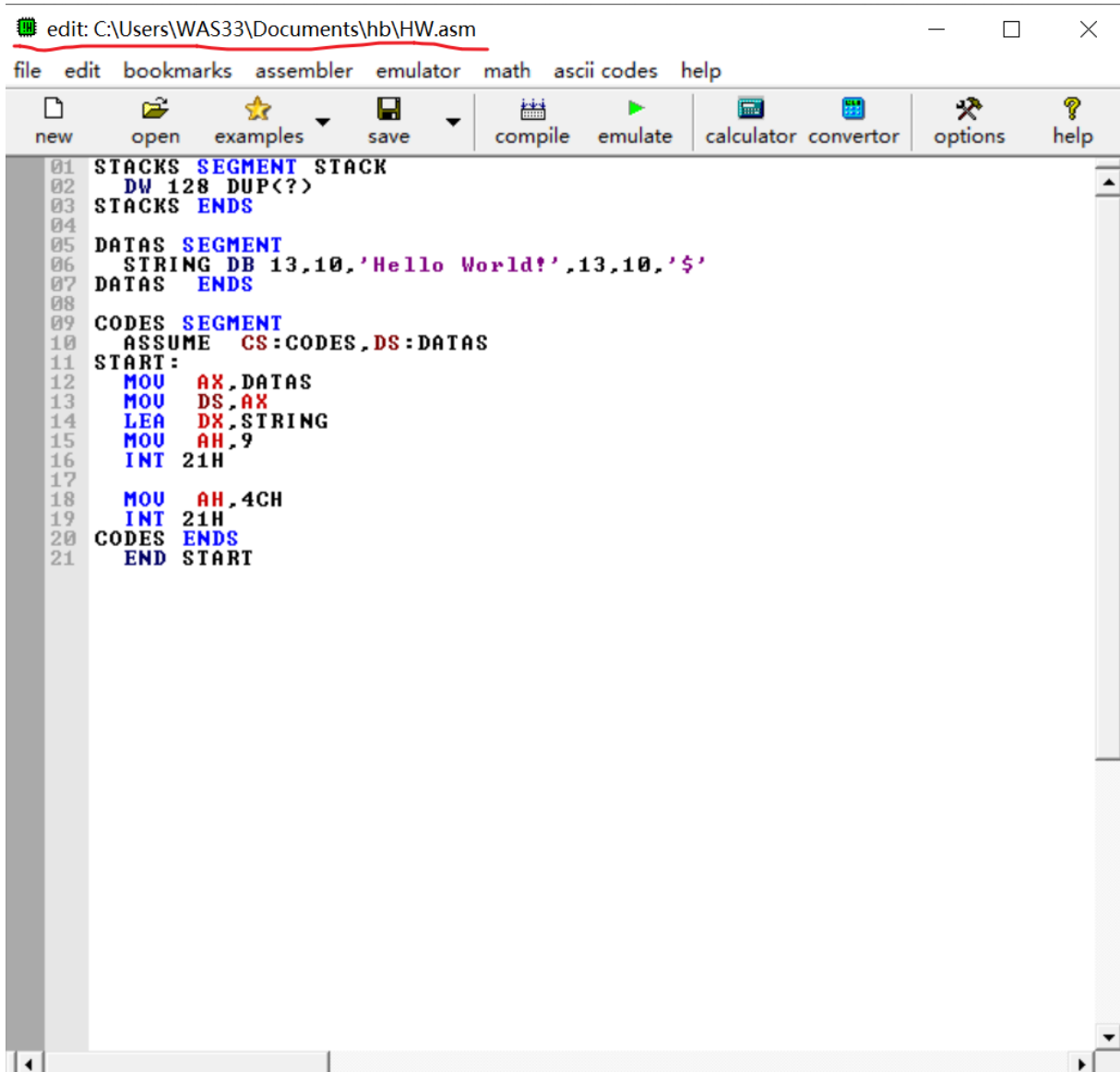
C:\>hw

Hello World!
```

【第二个实验】在 windows 环境下调试 Hello World 程序

- (1) 实验流程图:
- (2) 实验源代码（粘贴源代码）:
- (3) 实验代码、过程、相应结果（截图）并对实验进行说明和分析:

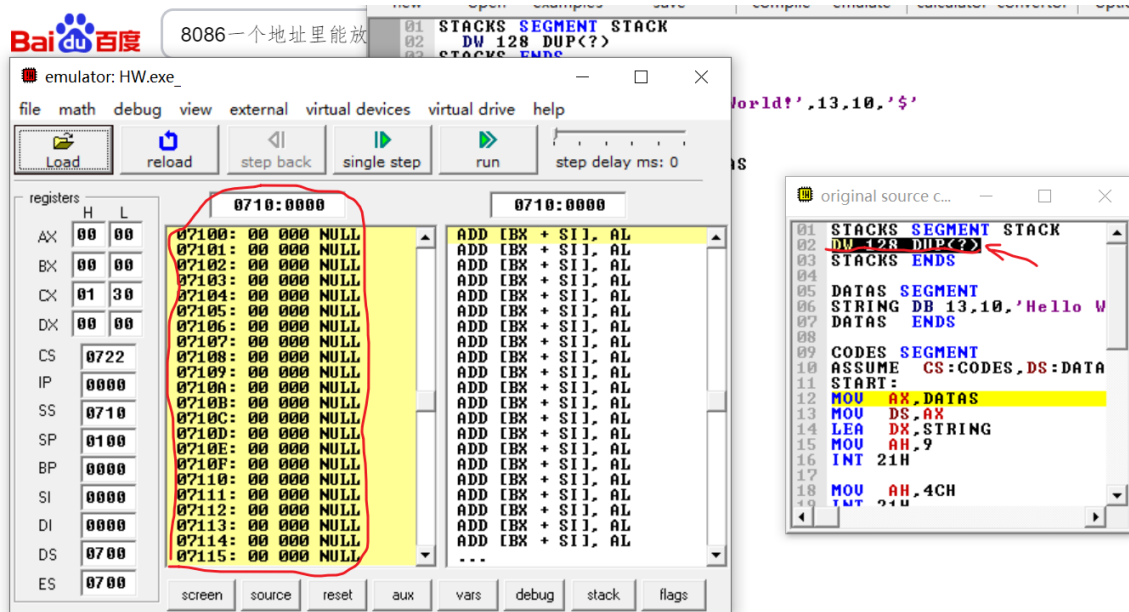
经过安装钟民老师软件 2 小时之后，我选择了放弃，在学长的推荐下选择了 EMU8086 作为调试软件，如下：



```

edit: C:\Users\WAS33\Documents\hb\HW.asm
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help
01 STACKS SEGMENT STACK
02 DW 128 DUP(?)
03 STACKS ENDS
04
05 DATAS SEGMENT
06 STRING DB 13,10,'Hello World!',13,10,'$'
07 DATAS ENDS
08
09 CODES SEGMENT
10 ASSUME CS:CODES,DS:DATAS
11 START:
12 MOV AX,DATAS
13 MOV DS,AX
14 LEA DX,STRING
15 MOV AH,9
16 INT 21H
17
18 MOV AH,4CH
19 INT 21H
20 CODES ENDS
21 END START
    
```

运行之后得到下图,点击 DW 128 DUP(?)的指令得到所开辟的内存空间确实是 128 个字, 256 个字节, 佐证了我问题①的提出

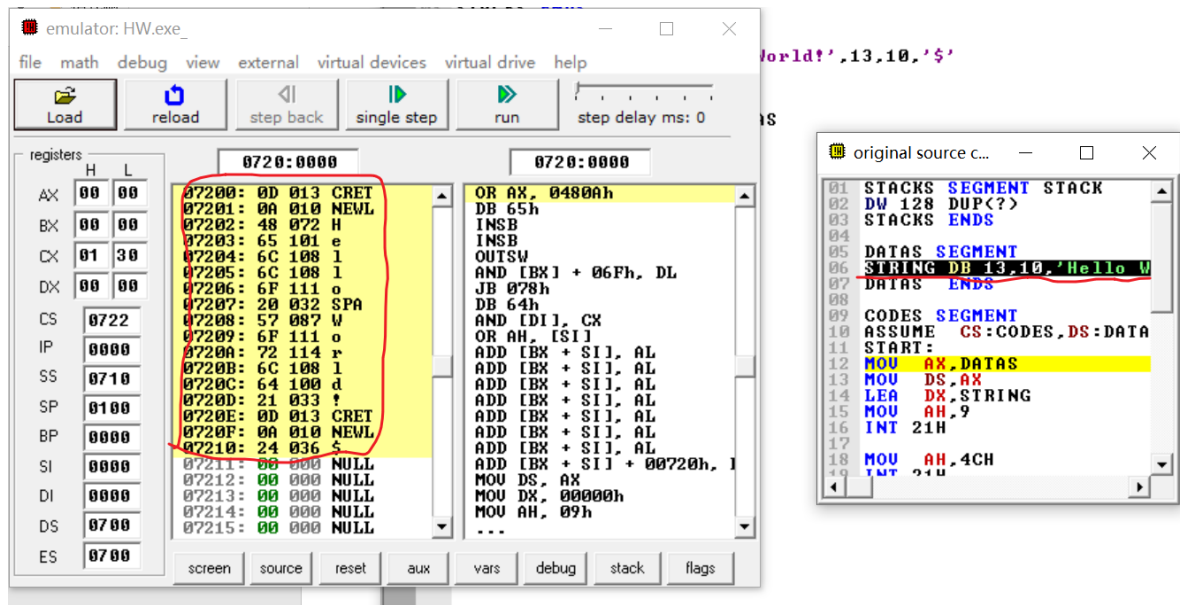


1个回答 2019年01月06日

最佳答案: 错。低字节总

8086处理一个地址

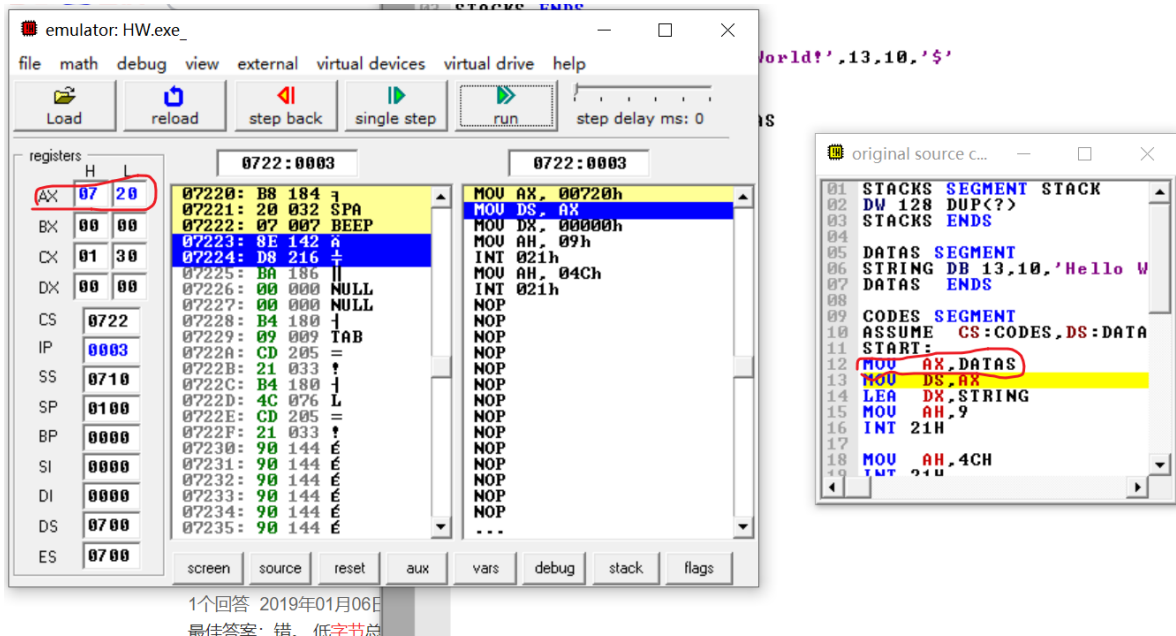
再点击 String 语句得到如下图



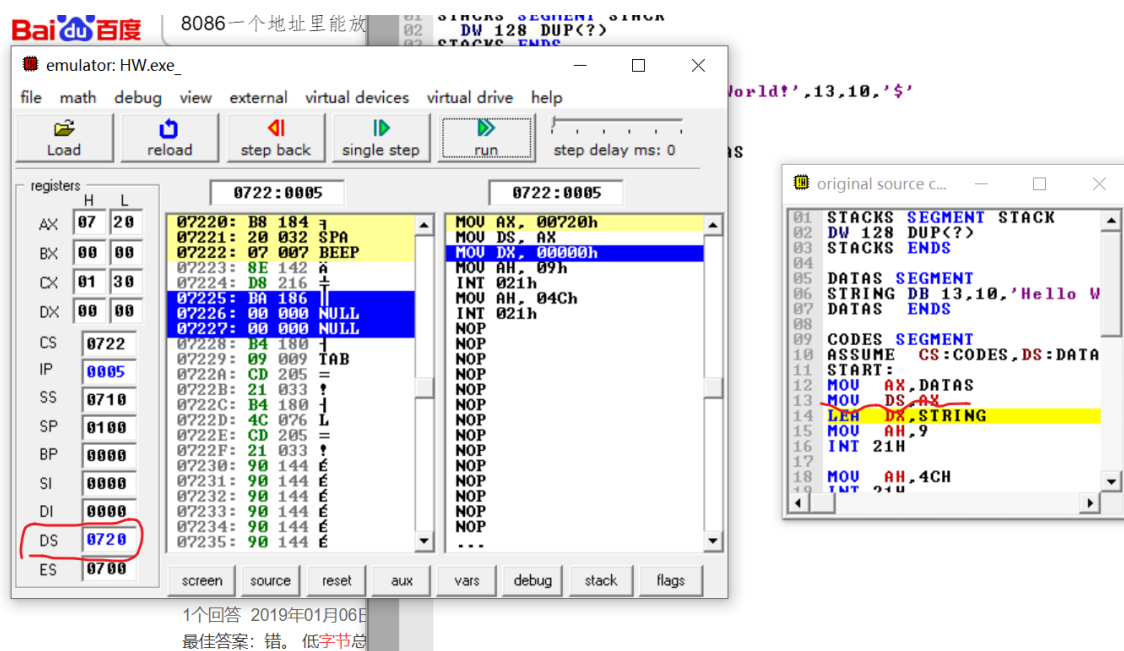
可以清晰地看到 Hello world 每个字母都是按照字节存储到地址空间中，且 13 和 10 是转换成 ASCII 码的

Start!

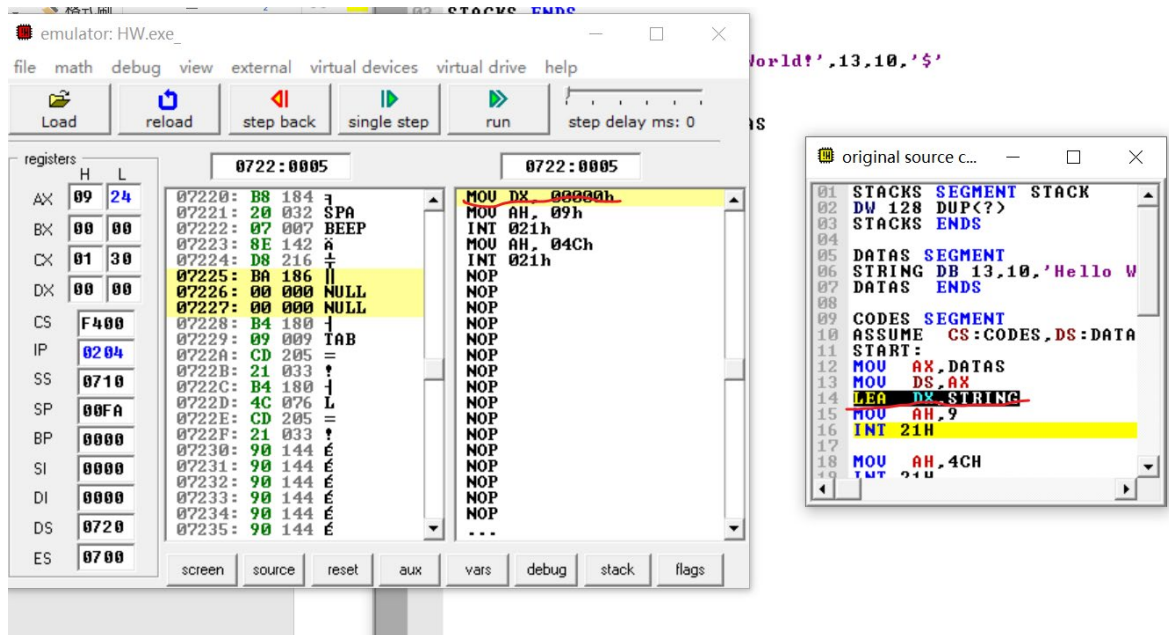
执行第一条指令：可以发现 AX 寄存器的值改变，将字符串的首地址放入了 AX 中



执行第二条指令：DS 的值发生了改变



将偏移地址通过 LEA dx string 指令给到 DX 寄存器，其中偏移地址为 0



再继续向下执行得到 Hello World



【实验心得】

【所遇到的问题】

1. 有关 DW 128 DUP(?)的问题

#是一个伪指令，用来定义一个堆栈段

STACKS SEGMENT STACK

#① DW是定义字类型变量的意思(问题?)|

DW 128 DUP(?)

STACKS ENDS

① 红笔所画的那一行代码中 DW 的意思是定义字类型变量，但是这里并没有涉及到变量为何要定义。是否是通过 DUP 的重复定义，定义了 128 个字类型变量将其放入 stack

中

② 红笔所画的那一行代码 128 所占数字部分，百度资料显示都是开辟了 128 个 **字** 的空间，为什么老师特意强调仅有 128 个 **字节** 的？

10 dup (1) 重复定义了10个字元素，初始值为1，占用10*2=20个字节。

5 dup (?) 重复定义了5个字元素（其初始值实际为0），占用5*2=10个字节

【百度所得】

1. 段

有数据段，代码段，堆栈段以及附加段

格式：

段名 SEGMENT [定位类型] [组合类型] [类别名]

段名 ENDS

功能：把程序分段，实现存储器的分段管理。

有关格式 SEGMENT 后面的部分暂时还未需要，要用用到时再去理解

2. DW 和 DB

；① DW是定义字类型变量的意思(问题①) DUP是连续复制

DW 128 DUP(?)

；结束堆栈段的定义

STACKS ENDS

；开始DATA段的定义，SEGMENT是定义段

DATAS SEGMENT

STRING **DB** 13,10,'Hello World! ',13,10,'\$'

DATAS ENDS

为什么这个上面是 DW 下面却用了 DB

百度结果如下所示：

通常情况下，汇编语言中的字符串，都会被赋予 **DB**（字节）数据类型，那么，为什么这样做？

不能使用DW数据类型吗？

当然可以，**但是不建议使用**。

3. INT 21H 的使用指南

；下面两条同时作用可以让其显示字符串

MOV AH,9

INT 21H

开始的时候我以为这条指令就是简单的 AH 赋值为 9，但是没想到 INT 21 是个中断指令并且可以和下面协调作用产生功能是属奇妙

调用步骤大致

- (1) 系统功能号送到寄存器AH中；
 - (2) 入口参数送到指定的寄存器中；
 - (3) 用INT 21H指令执行功能调用；
 - (4) 根据出口参数分析功能调用执行情况。
- 下面归纳5个在汇编中常用的INT 21H系统功能调用。

AH	功能	入口参数	出口参数
4CH	返回DOS	无	无
1	键盘输入一个字符到AL中	无	AL=字符
2	输出DL寄存器的字符到显示器	DL (存放一个字符)	无
9	输出一个以"\$"结尾的字符串到显示器	DS:字符串所在的段地址 DX:字符串首地址	无
0AH	从键盘输入一个字符串到指定缓冲区	DS:缓冲区所在的段地址 DX:缓冲区首地址	缓冲区相应位置

【上课所提问题的回答】

1. 堆和栈的区别？

答

栈内存:栈内存首先是一片内存区域，存储的都是局部变量，凡是定义在方法中的都是局部变量（方法外的是全局变量），for循环内部定义的也是局部变量，是先加载函数才能进行局部变量的定义，所以方法先进栈，然后再定义变量，变量有自己的作用域，一旦离开作用域，变量就会被释放。栈内存的更新速度很快，因为局部变量的生命周期都很短。

堆内存:存储的是数组和对象（其实数组就是对象），凡是new建立的都是堆中，堆中存放的都是实体（对象），实体用于封装数据，而且是封装多个（实体的多个属性），如果一个数据消失，这个实体也没有消失，还可以用，所以堆是不会随时释放的，但是栈不一样，栈里存放的都是单个变量，变量被释放了，那就没有了。堆里的实体虽然不会被释放，但是会被当成垃圾，Java有垃圾回收机制不定时的收取。

联系：堆可以给栈中的一个变量地址，使得改变量指向堆中的一个数组，类似于 C 语言中的指针

2. 下图的 String 可否换成任意的单词或者字符串？

```

01 STACKS SEGMENT STACK
02 DW 128 DUP(?)
03 STACKS ENDS
04
05 DATAS SEGMENT
06 STRING DB 13,10,'Hello W
07 DATAS ENDS
08
09 CODES SEGMENT
10 ASSUME CS:CODES,DS:DATA
11 START:
12 MOV AX,DATAS
13 MOV DS,AX
14 LEA DX,STRING
15 MOV AH,9
16 INT 21H
17
18 MOV AH,4CH
19 INT 21H
    
```

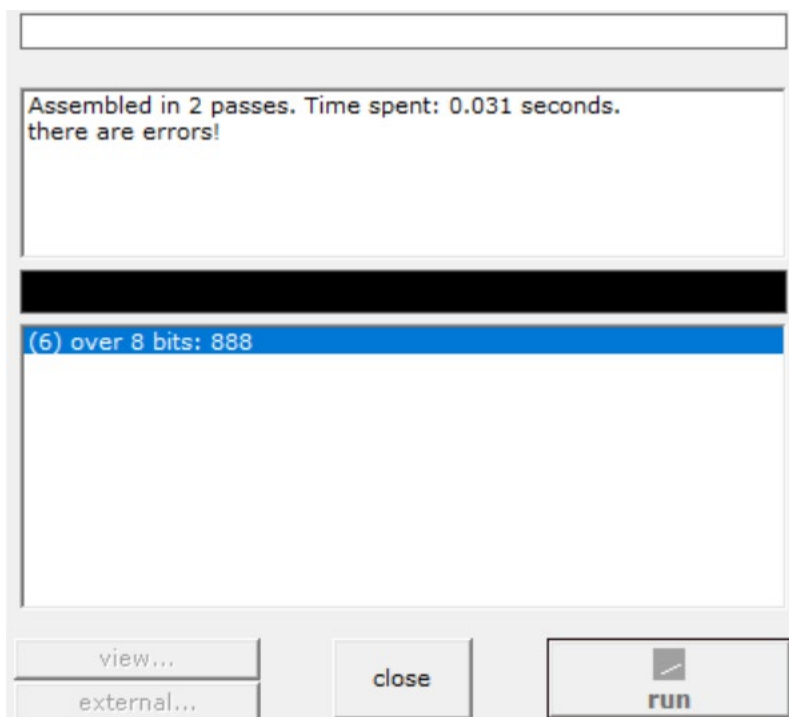
答：完全可以，它仅仅就是一个标识符而已

3. 下图中数据能否换成酷酷的 8888?

```

01 STACKS SEGMENT STACK
02 DW 128 DUP(?)
03 STACKS ENDS
04
05 DATAS SEGMENT
06 STRING DB 13,10,'Hello W
07 DATAS ENDS
08
09 CODES SEGMENT
10 ASSUME CS:CODES,DS:DATA
11 START:
12 MOV AX,DATAS
13 MOV DS,AX
14 LEA DX,STRING
15 MOV AH,9
16 INT 21H
17
18 MOV AH,4CH
19 INT 21H
    
```

理论上是可以的，但是 8888,8848 没有相应的 ASCII 码与之对应，就会显示超过了 8bits



4. 南桥和北桥分别指的是什么？芯片组

南北桥是一种芯片结构，南北桥结构是历史悠久而且相当流行的主板芯片组架构。采用南北桥结构的主板上都有两个面积比较大的芯片，靠近CPU的为北桥芯片，主要负责控制AGP显卡、内存与CPU之间的数据交换；靠近PCI槽的为南桥芯片，主要负责软驱、硬盘、键盘以及附加卡的数据交换。传统的南北桥架构是通过PCI总线来连接的，常用的PCI总线是33.3MHz工作频率，32bit传输位宽，所以理论最高数据传输率仅为133MB/s。由于PCI总线的共享性，当子系统及其它周边设备传输速率不断提高以后，主板南北桥之间偏低的数据传输率就逐渐成为影响系统整体性能发挥的瓶颈。因此，从英特尔i810开始，芯片组厂商都开始寻求一种能够提高南北桥连接带宽的解决方案。

北桥一般都有散热片的，离CPU不远

南桥一般都没有散热片，很好找

南北桥的位置是不固定的，看主板厂家

【心得体会】

我现在感觉跟电脑有关的职业和专业真的是痛并快乐着，老师您说的可真对，这次实验我装了2个多小时的软件都没装上，心情十分复杂，但是在装机的过程中，我却很享受，因为我百度到了很多关于软件的资料，实际上软件不也就是一些代码组成的么，在百度的过程中我了解到了什么是注册表，怎么将软件加入到防火墙的可信程序中，怎么通过错误信息到网上去查找解决方案，虽然最终还是没有将老师推荐的软件装上，也因为这个郁闷的晚上多吃了好几碗饭，但是最终的结果还是很快乐的。我换了一个软件emu8086去完成相应的实验。

学习汇编给我最大的感受就是爽！之前学习C和python那种高级语言的时候我都感

觉仅仅学到了一些皮毛，对很多东西的底层架构完全不了解，而汇编以及那么多优秀的 IDE 却能够让我从内存的角度去看很多东西，看计算机是怎样运行的，每一条指令去向什么地方。之前我其实对编程这种东西不是很感冒，感觉很虚没有意思，但是学习汇编这仅仅一周就让我感受到了它无尽的创造力，实际上我不仅爱上了编程，更爱上了计算机硬件，发明这个东西的人究竟是有多么聪明的大脑。就像 `MOV AH,9 INT 21H` 这样的指令是多么的奇妙，这样两行代码就能够让计算机把字符串打印到屏幕上，无敌！