# 计算机组成原理实验报告

班级：计科2班

姓名：薄劲阳

学号：2020115025

# 实验一 汇编环境安装与基础实验

**实验内容：**

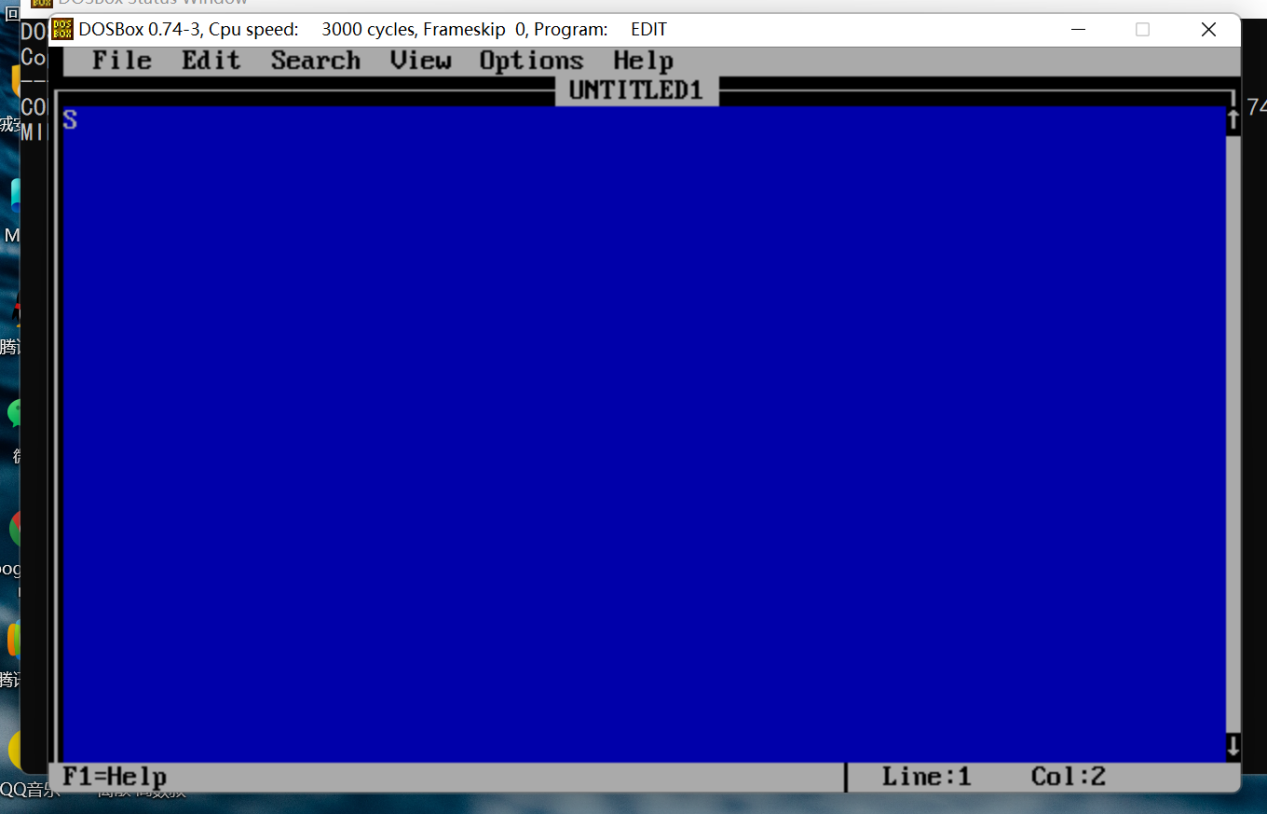
（1）下载安装MASM5.0集成开发软件，或者其他x86汇编语言开发环境；熟悉开发工具的使用方法；

（2）利用串操作指令在内存中存储字符串“Hello world”,通过观察窗口查看内存内容，检查每个字符的ASCII编码是否正确。

**要求：**

1. 编写汇编程序
2. 保存内存检测窗口截图

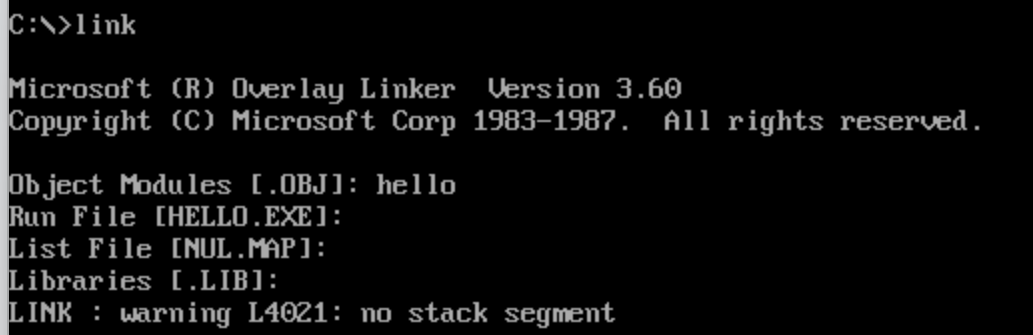
* 查看cs：ip中的内容
* 进入edit模式，编写程序

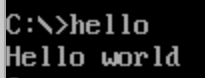


1. data segment ;这里定义一个数据段
2. tab db 'Hello world$' ;这里用内存存放字节数据 'hellow world!',$用来判断字符串是否输出完毕
3. data ends ;数据段的结束标志
5. code segment ;这里定义了一个代码段
6. assume cs:code, ds:data ;这里把程序中定义的段与对应的段寄存器关联起来
8. start: ;这里是一个标号，根据end后面的标号判断这里是程序的开始位置
9. mov ax,data
10. mov ds,ax ;这里把数据段的地址放到数据段寄存器ds中
11. lea dx,tab ;dx中放将要显示数据的偏移地址
12. mov ah,9h
13. int 21h ;调用21号中断的9号功能来显示字符串
14. mov ah,4ch
15. int 21h
16. code ends ;代码段的结束语
17. end start ;定义程序从哪个标号处开始执行

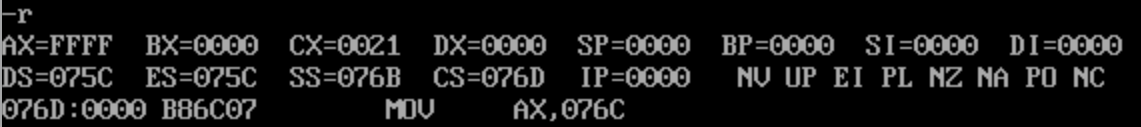
* 编译，链接，运行程序
  + 编译

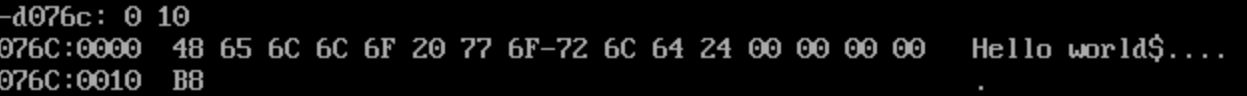


* + 链接
  + 运行程序



* 查看ds段中内存中内容，判断是否进行了修改





# 实验二 内存读写

**实验内容：**

（1）在内存数据区连续存储10个随机整数，依次读出这10个数，并存储到内存的另一个区域。

（2）不要使用循环程序

**要求：**

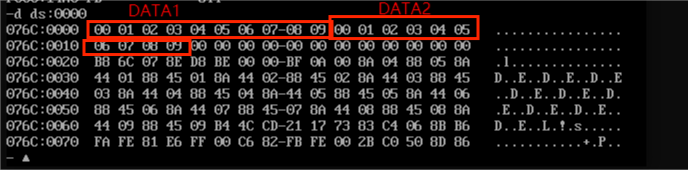
（1）编写汇编程序

（2）保存内存检测窗口截图

**1.汇编程序如下**

1. ；数据段
2. DATA SEGMENT
3. AREA1 DB 00H,01H,02H,03H,04H,05H,06H,07H,08H,09H ；AREA1存放十个未知数
4. AREA2 DB 10 DUP(0)  ；AREA2为十个字节大小的的内存区，初始化为0
5. DATA ENDS
7. CODE SEGMENT
8. ASSUME CS:CODE,DS:DATA  ;伪指令
9. START:
10. MOV AX,DATA
11. MOV DS,AX  ；DS指向DATA段
12. ；获取偏移地址
13. MOV SI,OFFSET AREA1
14. MOV DI,OFFSET AREA2
15. ;将AREA1中的内容搬运到AREA2中
16. MOV AL,[SI]
17. MOV [DI],AL
18. ;搬运下一个内存单元（字节），偏移地址加一即可
19. MOV AL,[SI+1]
20. MOV [DI+1],AL
22. MOV AL,[SI+2]
23. MOV [DI+2],AL
25. MOV AL,[SI+3]
26. MOV [DI+3],AL
28. MOV AL,[SI+4]
29. MOV [DI+4],AL
31. MOV AL,[SI+5]
32. MOV [DI+5],AL
34. MOV AL,[SI+6]
35. MOV [DI+6],AL
37. MOV AL,[SI+7]
38. MOV [DI+7],AL
40. MOV AL,[SI+8]
41. MOV [DI+8],AL
43. MOV AL,[SI+9]
44. MOV [DI+9],AL
46. MOV AH,4CH
47. **INT** 21H
48. ；，这个中断调用指令就是告诉程序当程序里的指令（除了放在它最低行的“mov
49. ah,4ch int 21h”）执行完毕后要做什么——返回dos，此时程序就会结束，电脑界面上dos窗口（就是windowsxp运行cmd后出现的那个窗口）就会出现一行英文，其意思是“请按任意键继续”
50. CODE ENDS
51. END START
52. **内存检测窗口如下：**

可以看到DATA1与DATA2内容一样，数据转移成功。



# 实验三 定点整数加减法

**实验内容：**

（1）在内存数据区定义10个小于20的定点整数，利用加法指令累加，结果存入内存指定单元；

（2）可以使用循环程序；

**要求：**

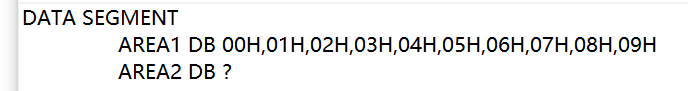
（1）编写汇编程序

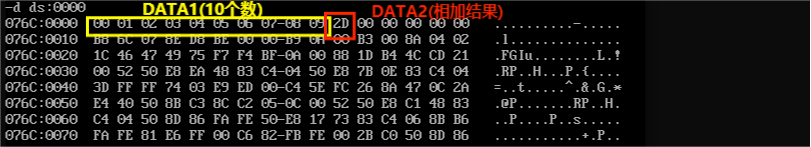
（2）保存内存检测窗口截图

**1.汇编程序如下：**

1. DATA SEGMENT
2. AREA1 DB 00H,01H,02H,03H,04H,05H,06H,07H,08H,09H  ；连续的十个内存单元
3. AREA2 DB ?  ；在AREA2中随机存放数据
4. DATA ENDS
6. CODE SEGMENT
7. ASSUME CS:CODE,DS:DATA
8. START:
9. MOV AX,DATA
10. MOV DS,AX  ；DS指向DATA（获取段地址）
12. MOV SI,OFFSET AREA1  ；获取AREA1的偏移地址
13. MOV CX,10  ；循环变量
14. MOV BL,0   ；累加器
15. AGAIN:
16. MOV AL,[SI]
17. ADD BL,[SI]
18. INC SI
19. INC DI
20. DEC CX
21. JNZ AGAIN  ；如果不为零 跳转到AGAIN
22. HLT
23. ；将求和结果放入AREA2中
24. MOV DI,OFFSET AREA2
25. MOV [DI],BL
26. MOV AH,4CH
27. **INT** 21H
29. CODE ENDS
30. END START
31. **内存检测窗口如下：**

定义的10个数为0~9，相加结果为2D





**实验四** **寻址方式测试**

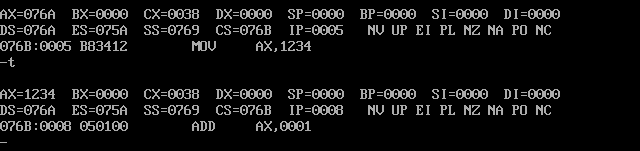
**实验内容：**

使用不同的汇编指令，分别测试隐含寻址、立即寻址、直接寻址、间接寻址、寄存器寻址、寄存器间接寻址、偏移寻址和段寻址；

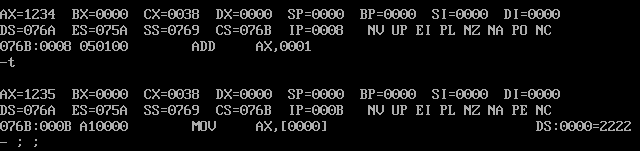
**要求：**

1. 编写汇编程序
2. ;数据段
3. DATA    SEGMENT
4. DATA1   DW 2222H  ；定义一个字单元 高八位和低八位均为22H
5. DATA2   DB 0 ;定义一个字节单元 存零
6. DATA    ENDS
7. ;代码段
8. CODE    SEGMENT
9. ASSUME CS:CODE,DS:DATA
10. START:  MOV AX,DATA
11. MOV DS,AX  ；DS指向DATA段（获取段地址）
12. MOV AX,1234H ;立即寻址 - 源操作数为立即数
14. ADD AX,0001H;
16. MOV AX,DS:[0000H] ;直接寻址-立即数存放在内存单元中
18. MOV AX,1234H
19. MOV DX,AX ;寄存器寻址-源操作数为寄存器编号，立即数在寄存器中
21. MOV SI,1234H
22. MOV AX,[SI]
23. MOV DX,AX ;寄存器间接寻址-源操作数为寄存器编号，寄存器中存放立即数的内存地址
25. MOV AX,[SI+100H];寄存器相对寻址
27. MOV BX,0001H
28. MOV SI,0500H
29. MOV AX,[BX][SI] ;基址变址寻址
31. MOV AH,4CH
32. **INT** 21H
33. CODE    ENDS
34. END START
35. 保存内存检测窗口截图

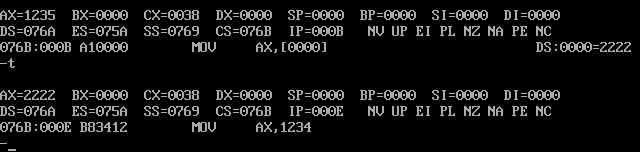
MOV AX,1234H;立即寻址



ADD AX,0001H;隐含寻址

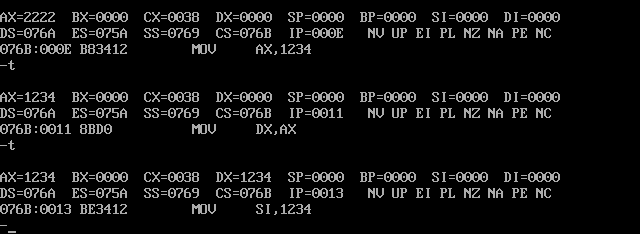


MOV AX,DS:[0000H];直接寻址



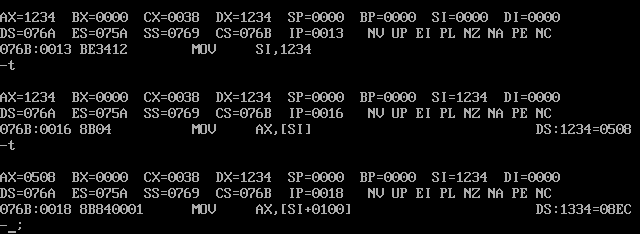
MOV AX,1234H

MOV DX,AX;寄存器寻址

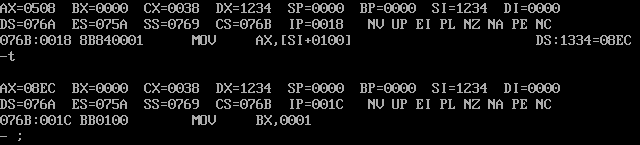


MOV SI,1234H

MOV AX,[SI];寄存器间接寻址



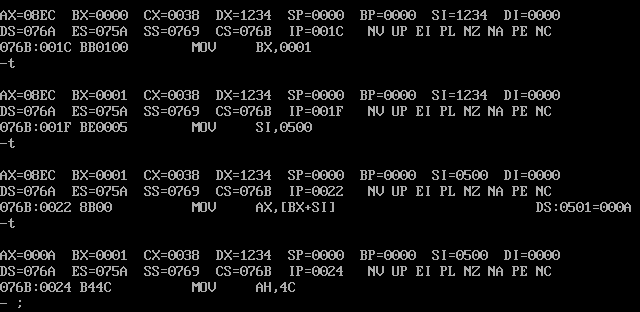
MOV AX,[SI+100H];寄存器相对寻址



MOV BX,0001H

MOV SI,0500H

MOV AX,[BX][SI];偏移寻址（基址变址寻址）



**实验五 堆栈实验**

**实验内容：**

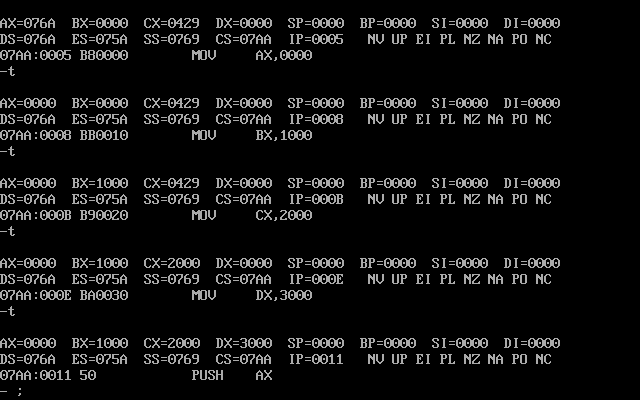
在内存开辟一个堆栈区，利用堆栈操作指令实现4个已赋值寄存器内容入栈，然后出栈，重新保存到对应寄存器。

**要求：**

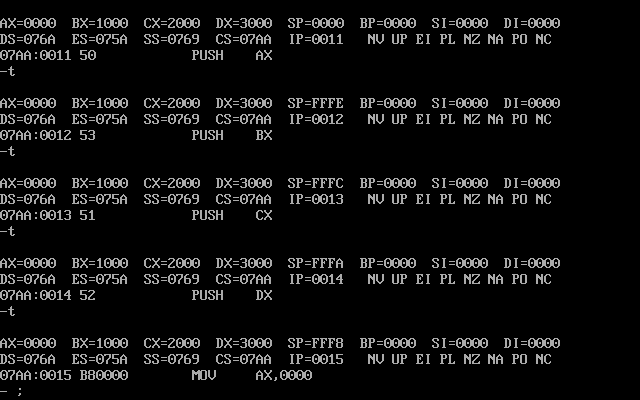
1. 编写汇编程序
2. DATA    SEGMENT
3. DATA1   DW 2222H
4. DATA2   DB ?
5. DATA    ENDS
6. ；堆栈段
7. STACK   SEGMENT
8. DW 500 DUP(0)  ；500个内存单元（字节） 存放500个0
9. STACK   ENDS
11. CODE    SEGMENT
12. ASSUME CS:CODE,DS:DATA,SS:STACK
13. START:  MOV AX,DATA
14. MOV DS,AX
15. ；堆栈段特点——先进后出
16. MOV AX,0000H
17. MOV BX,1000H
18. MOV CX,2000H
19. MOV DX,3000H
20. ；压栈
21. PUSH AX
22. PUSH BX
23. PUSH CX
24. PUSH DX
26. MOV AX,0H
27. MOV BX,0H
28. MOV CX,0H
29. MOV DX,0H
30. ；出栈
31. POP DX
32. POP CX
33. POP BX
34. POP AX
36. MOV AH,4CH
37. **INT** 21H
38. CODE    ENDS
39. END START

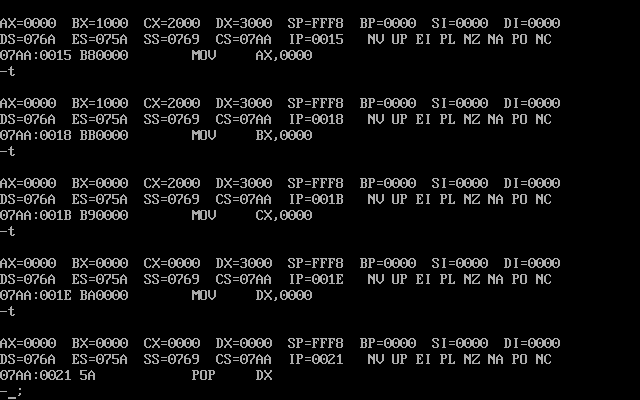
（2）保存内存检测窗口截图

首先给AX,BX,CX,DX分别赋值为0000，1000,2000,3000



然后将其按照AX,BX,CX,DX的顺序入栈，之后改变四个寄存器的值全为0





最后按照DX,CX,BX,AX的顺序出栈，可以看到结果为0000,1000,2000,3000，与初始结果相同。