# 《微机系统与汇编语言》

# 第一次实验报告

报告撰写人：2152118 史君宝

作业的问题解答在报告的最后面。

## 一、实验目的和要求

### 1、掌握和了解程序的运行方式

解答：汇编程序的内容格式，一个基本的如下：
总结来说就是将代码分为数据段和代码段，利用汇编指令和伪指令将其组织起来，并最终进行编译。

```
DATA    SEGMENT                          ;数据段定义
A       DB      12H
B       DB      78H
C       DB      ?
DATA    ENDS
CODE    SEGMENT                          ;代码段定义
        ASSUME  CS:CODE,DS:DATA          ;ASSUME语句
START:  MOV     AX,DATA                  ;装填DS
        MOV     DS,AX
        MOV     AL,A
        ADD     AL,B                     ;程序主体
        MOV     C,AL
        MOV     AH,4CH                   ;返回DOS
        INT     21H
CODE    ENDS
        END     START                    ;结束语言
```

六要素

### 2、根据模板文件，创建自己的程序

解答：
问题原题：

1、写出下面运算对应的程序
a=15,b=16,c=17,d=19
a=b+c-d
要求计算过程的每一步都进行寄存器的查看，并在文档中叙述每一步的工作机理

编写的程序显示：

```
data segment
   A db 15
   B db 16
   Ca db 17
   D db 19
data ends

code segment
   assume cs:code, ds:data
start: mov ax, data
       mov ds, ax
       mov al, B
       add al, Ca
       sub al, D
       mov A, al

       mov ah, 4CH
       int 21H
code ends
end start
```

## 二、实验环境及配置

### 1、软件环境

Windows10 64 位+MasmPlus+DosBox +Debug32

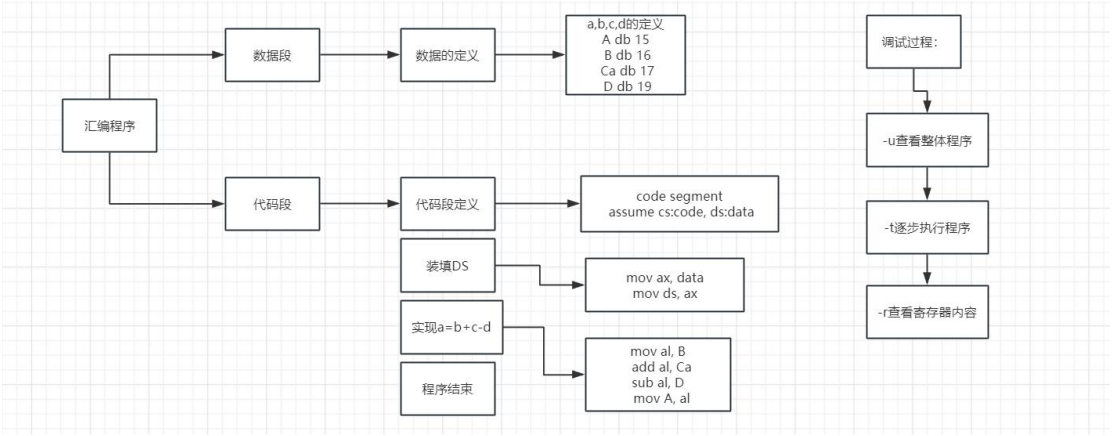
## 三、实验设计及调试

### （一）实验内容

编写对应的程序代码，实现对 a,b,c,d 四个的计算，即实现 a=b+c-d 的计算，并通过调试程序 debug32 对所编写的 first.exe 进行调试，通过 debug 程序逐步观察汇编指令和各步骤之间的寄存器情况，进行逐步调试。

### （二）实验设计及调试步骤

#### （1）对实验内容进行分析，设计程序流程图




## （2）实验程序

编写的程序如下

```
data segment
   A db 15
   B db 16
   Ca db 17
   D db 19
data ends

code segment
   assume cs:code, ds:data
start: mov ax, data
       mov ds, ax
       mov al, B
       add al, Ca
       sub al, D
       mov A, al

       mov ah, 4CH
       int 21H
code ends
end start
```

## （3）调试计划

### a、调试时所需观察的内容结果

首先是第一步：-u 查看整体的程序：

```
C:\>debug32 first.exe
Debug32 - Version 1.0 - Copyright (C) Larson Computing 1994

CPU = 486, Real Mode, Id/Step = 0402, A20 disabled
-u
1CA6:0000 B8A51C          MOV     AX,1CA5h
1CA6:0003 8ED8            MOV     DS,AX
1CA6:0005 A00100          MOV     AL,[0001]
1CA6:0008 02060200        ADD     AL,[0002]
1CA6:000C 2A060300        SUB     AL,[0003]
1CA6:0010 A20000          MOV     [0000],AL
1CA6:0013 B44C            MOV     AH,4Ch
1CA6:0015 CD21            INT     21h
1CA6:0017 0000            ADD     [BX+SI],AL
1CA6:0019 0000            ADD     [BX+SI],AL
1CA6:001B 0000            ADD     [BX+SI],AL
1CA6:001D 0000            ADD     [BX+SI],AL
```

总共有 8 步：到最后的一步 int 21H，剩下的程序是自动执行的无关代码，不断循环往复，与本题没有关系。

我们逐步对 8 步的汇编程序逐步进行调试，然后依次利用-t 和-r 来进行查看。未执行的时候：

```
-r
AX=0000  BX=0000  CX=0027  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=1C95  ES=1C95  SS=1CA4  CS=1CA6  IP=0000  NV UP DI PL NZ NA PO NC
1CA6:0000 B8A51C          MOV     AX,1CA5h
```

执行下一步：

```
-t
AX=1CA5  BX=0000  CX=0027  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=1C95  ES=1C95  SS=1CA4  CS=1CA6  IP=0003  NV UP DI PL NZ NA PO NC
1CA6:0003 8ED8            MOV     DS,AX
Trace Interrupt
-r
AX=1CA5  BX=0000  CX=0027  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=1C95  ES=1C95  SS=1CA4  CS=1CA6  IP=0003  NV UP DI PL NZ NA PO NC
1CA6:0003 8ED8            MOV     DS,AX
```

```
-t
AX=1CA5  BX=0000  CX=0027  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=1CA5  ES=1C95  SS=1CA4  CS=1CA6  IP=0005  NV UP DI PL NZ NA PO NC
1CA6:0005 A00100          MOV     AL,[0001]
Trace Interrupt
-r
AX=1CA5  BX=0000  CX=0027  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=1CA5  ES=1C95  SS=1CA4  CS=1CA6  IP=0005  NV UP DI PL NZ NA PO NC
1CA6:0005 A00100          MOV     AL,[0001]
```

```
-t
AX=1C10  BX=0000  CX=0027  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=1CA5  ES=1C95  SS=1CA4  CS=1CA6  IP=0008  NV UP DI PL NZ NA PO NC
1CA6:0008 02060200        ADD     AL,[0002]
Trace Interrupt
-r
AX=1C10  BX=0000  CX=0027  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=1CA5  ES=1C95  SS=1CA4  CS=1CA6  IP=0008  NV UP DI PL NZ NA PO NC
1CA6:0008 02060200        ADD     AL,[0002]
```

```
-t
AX=1C21  BX=0000  CX=0027  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=1CA5  ES=1C95  SS=1CA4  CS=1CA6  IP=000C  NV UP DI PL NZ NA PE NC
1CA6:000C 2A060300        SUB     AL,[0003]
Trace Interrupt
-r
AX=1C21  BX=0000  CX=0027  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=1CA5  ES=1C95  SS=1CA4  CS=1CA6  IP=000C  NV UP DI PL NZ NA PE NC
1CA6:000C 2A060300        SUB     AL,[0003]
```

```
-t
AX=1C0E  BX=0000  CX=0027  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=1CA5  ES=1C95  SS=1CA4  CS=1CA6  IP=0010  NV UP DI PL NZ AC PO NC
1CA6:0010 A20000          MOV     [0000],AL
Trace Interrupt
-r
AX=1C0E  BX=0000  CX=0027  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=1CA5  ES=1C95  SS=1CA4  CS=1CA6  IP=0010  NV UP DI PL NZ AC PO NC
1CA6:0010 A20000          MOV     [0000],AL
```

```
-t
AX=1C0E  BX=0000  CX=0027  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=1CA5  ES=1C95  SS=1CA4  CS=1CA6  IP=0013  NV UP DI PL NZ AC PO NC
1CA6:0013 B44C            MOV     AH,4Ch
Trace Interrupt
-r
AX=1C0E  BX=0000  CX=0027  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=1CA5  ES=1C95  SS=1CA4  CS=1CA6  IP=0013  NV UP DI PL NZ AC PO NC
1CA6:0013 B44C            MOV     AH,4Ch
```

```
-t
AX=4C0E  BX=0000  CX=0027  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=1CA5  ES=1C95  SS=1CA4  CS=1CA6  IP=0015  NV UP DI PL NZ AC PO NC
1CA6:0015 CD21            INT     21h  ;End Program
Trace Interrupt
-r
AX=4C0E  BX=0000  CX=0027  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=1CA5  ES=1C95  SS=1CA4  CS=1CA6  IP=0015  NV UP DI PL NZ AC PO NC
1CA6:0015 CD21            INT     21h  ;End Program
```

b、根据各调试目的分别选择调试所需的方法，如单步、断点等命令采用单步执行，通过 r 命令观察寄存器，d 命令观察内存 c、调试过程记录

```
-u
1CA6:0000 B8A51C          MOV     AX,1CA5h
1CA6:0003 8ED8            MOV     DS,AX
1CA6:0005 A00100          MOV     AL,[0001]
1CA6:0008 02060200        ADD     AL,[0002]
1CA6:000C 2A060300        SUB     AL,[0003]
1CA6:0010 A20000          MOV     [0000],AL
1CA6:0013 B44C            MOV     AH,4Ch
1CA6:0015 CD21            INT     21h
1CA6:0017 0000            ADD     [BX+SI],AL
1CA6:0019 0000            ADD     [BX+SI],AL
1CA6:001B 0000            ADD     [BX+SI],AL
1CA6:001D 0000            ADD     [BX+SI],AL
-d cs:ip
1CA6:0000 B8 A5 1C 8E D8 A0 01 00-02 06 02 00 2A 06 03 00  8?..X ......*...
1CA6:0010 A2 00 00 B4 4C CD 21 00-00 00 00 00 00 00 00 00  "..4LM!.........
1CA6:0020 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
***Duplicate Line(s)***
1CA6:0070 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
-r
AX=0000  BX=0000  CX=0027  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=1C95  ES=1C95  SS=1CA4  CS=1CA6  IP=0000  NV UP DI PL NZ NA PO NC
1CA6:0000 B8A51C          MOV     AX,1CA5h
```

四、实验小结（实验调试过程中所遇到的问题、解决问题的思路和解决的方法）

（自己的东西自己写，后面的问题的回答也放在这里）

遇到的问题：

（1）程序编写的问题 1：在编写源程序的时候我们将第三个变量仍然写作 c，但是在编译的过程中会出现编译错误的问题，我上网查找原因的时候，查到了这样的官方手册：

```
D:\MASMPlus\Project\second.asm(11) : error A2008: syntax error : C
D:\MASMPlus\Project\second.asm(19) : error A2008: syntax error : C
```

语法错误:↵
当前位置的标记导致语法错误。↵
可能发生下列情况之一:↵
在指令中添加或省略点前缀。↵
保留字(如 C 或 SIZE)被用作标识符。↵
所使用的指令在当前处理器或协处理器选择中不可用。↵
在条件汇编语句中使用比较运行时操作符(如==)而不是关系操作符(如 EQ)。↵
一条指令或一条指令的操作数太少。↵
使用了一个过时的指令。↵



看来是 c 是一种保留字，我在这里将其用做了标识符，然后就报错了。

（2）在最开始的调试的时候，由于方法的不正确，导致了很多相关的问题：
比如，我在运行 debug first.exe 的时候由于没有加上最后的.exe 出现了很多的报错：

```
C:\>debug32 first
Debug32 - Version 1.0 - Copyright (C) Larson Computing 1994

File not found
CPU = 486, Real Mode, Id/Step = 0402, A20 disabled
-u
1C8B:0100 0000            ADD     [BX+SI],AL
1C8B:0102 0000            ADD     [BX+SI],AL
1C8B:0104 0000            ADD     [BX+SI],AL
1C8B:0106 0000            ADD     [BX+SI],AL
1C8B:0108 0000            ADD     [BX+SI],AL
1C8B:010A 0000            ADD     [BX+SI],AL
1C8B:010C 0000            ADD     [BX+SI],AL
1C8B:010E 0000            ADD     [BX+SI],AL
1C8B:0110 0000            ADD     [BX+SI],AL
1C8B:0112 0000            ADD     [BX+SI],AL
1C8B:0114 0000            ADD     [BX+SI],AL
1C8B:0116 0000            ADD     [BX+SI],AL
```

比如这里我们执行 debug32 first 的时候由于未找到文件出现了 File not found 的问题，因此开始循环执行默认的程序。

我还试过其他的：

```
C:\>debug first
Extended Error 2

-u
073F:0100 2E            CS:
073F:0101 FF16075F      CALL    [5F07]
073F:0105 C706B1DEDBDD  MOV     WORD PTR [DEB1],DDDB
073F:010B B030          MOV     AL,30
073F:010D E93F03        JMP     044F
073F:0110 2E            CS:
073F:0111 FF16095F      CALL    [5F09]
073F:0115 C706B1DEEBDD  MOV     WORD PTR [DEB1],DDEB
073F:011B E93400        JMP     0152
073F:011E 2E            CS:
073F:011F 07            POP     ES
```

出现了很多我看不懂的东西，暂时不太明白其内容。

（3）调试的过程，在原来的 debug.ppt 中我还是不太能够看懂很多的调试方法，最终我上网查了一下具体的方法。

# 问题解答：

特别需要关注，cs 和 ds 的值

问题一、按照上面屏幕给出的信息，代码中涉及到的内存和寄存器的值是？

问题二、单步执行 2 步后截屏如下，a 的物理地址是多少？

问题三、继续单步执行每一步，解释每一步对应的指令影响的寄存器的变化情况

（0）问题前述，cs 是代码段的段地址，而 ds 是数据段的段地址，两者分别对应数据和对应的代码段地址，我们在之后的查看中可以明了。

（1）解释具体的内存和寄存器内容：

```
-r
AX=0000  BX=0000  CX=0027  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=1C95  ES=1C95  SS=1CA4  CS=1CA6  IP=0000  NV UP DI PL NZ NA PO NC
1CA6:0000 B8A51C          MOV     AX,1CA5h
```

AX BX CX DX 分别是 4 个通用的寄存器，然后 DS 是数据段的段地址，CS 是代码段的段地址，我们在之后可以看到，具体内容。
例如本句之后，执行 MOV AX，1CA5h
执行后的寄存器内容如下：

```
-t
AX=1CA5  BX=0000  CX=0027  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=1C95  ES=1C95  SS=1CA4  CS=1CA6  IP=0003  NV UP DI PL NZ NA PO NC
1CA6:0003 8ED8            MOV     DS,AX
Trace Interrupt
-r
AX=1CA5  BX=0000  CX=0027  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=1C95  ES=1C95  SS=1CA4  CS=1CA6  IP=0003  NV UP DI PL NZ NA PO NC
1CA6:0003 8ED8            MOV     DS,AX
```

我们可以看到，对应的 AX 改为 1CA5，然后执行本条代码之后，继续跳到下一条指令，我们可以看到对应的 IP 寄存器也发生了改变，也就是代码的偏移地址发生变化，指向下一条指令。

我们可以采用-d 指令去查看对应的内存单元：
代码内存单元的内容：

```
-d cs:ip
1CA6:0000 B8 A5 1C 8E D8 A0 01 00-02 06 02 00 2A 06 03 00  8%..X ......*...
1CA6:0010 A2 00 00 B4 4C CD 21 00-00 00 00 00 00 00 00 00  "..4LM!.........
1CA6:0020 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
***Duplicate Line(s)***
1CA6:0070 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
```

数据的内存单元是：

```
-d 1CA5:0000
1CA5:0000 0F 10 11 13 00 00 00 00-00 00 00 00 00 00 00 00  ................
1CA5:0010 B8 A5 1C 8E D8 A0 01 00-02 06 02 00 2A 06 03 00  8%..X ......*...
1CA5:0020 A2 00 00 B4 4C CD 21 00-00 00 00 00 00 00 00 00  "..4LM!.........
1CA5:0030 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
***Duplicate Line(s)***
1CA5:0070 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
```

我们可以看到从 1CA5 开始的单元分别对应 a，b，c，d 它们的十六进制分别是 0F，10，11，13 对应的数字时 15 16 17 19
对应题目：

a=15,b=16,c=17,d=19

a=b+c-d

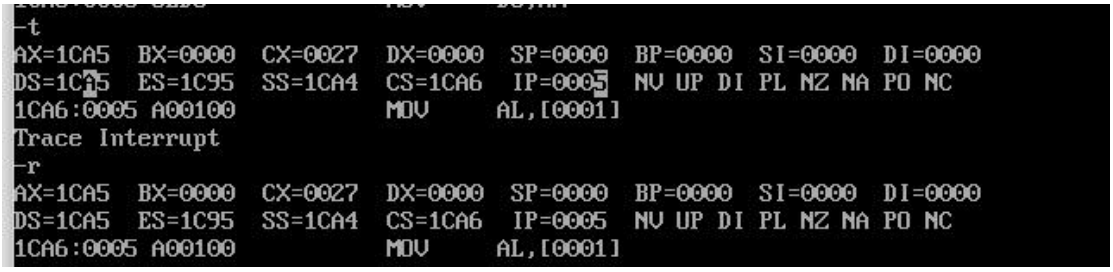
（2）问题二，对于原来的汇编程序我们可以知道，a 的段地址和偏移地址是 1CA5:0000 算出来的物理地址为 1CA50H

（3）接下来我们逐步解释内容：

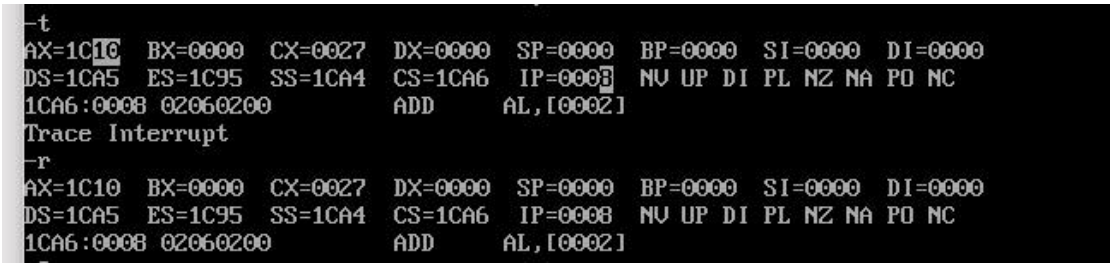
```
-r
AX=0000  BX=0000  CX=0027  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=1C95  ES=1C95  SS=1CA4  CS=1CA6  IP=0000  NV UP DI PL NZ NA PO NC
1CA6:0000 B8A51C          MOV     AX,1CA5h
```

执行 MOV AX，1CA5h 能够看到 AX 发生了变化，为 1CA5h

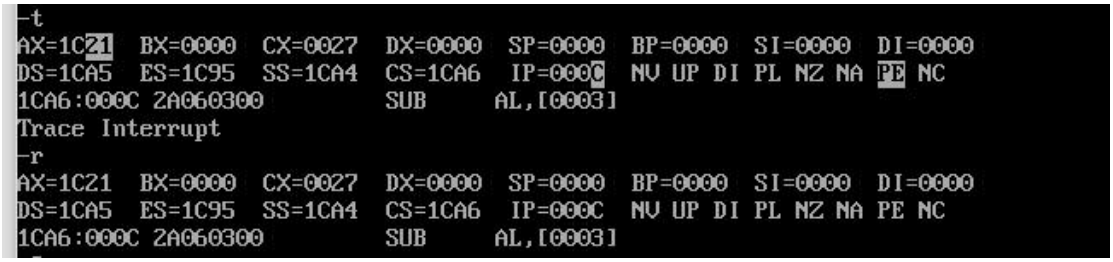

执行 MOV DS,AX 能够看到 DS 发生了变化，为 AX 的值



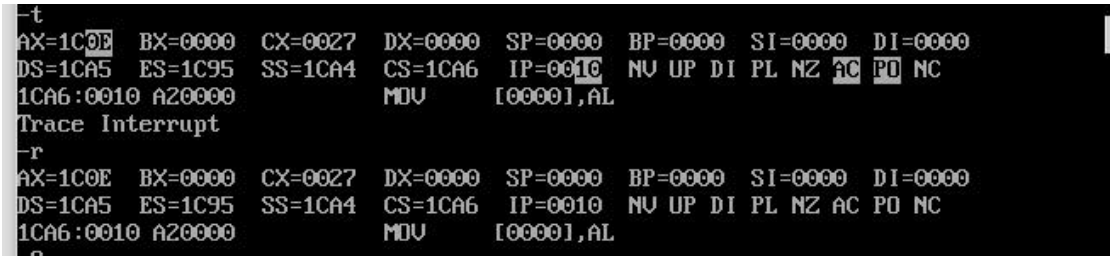

执行 MOV AL,[0001] 能够看到 AX 的低位部分发生了变化，为 B 的值



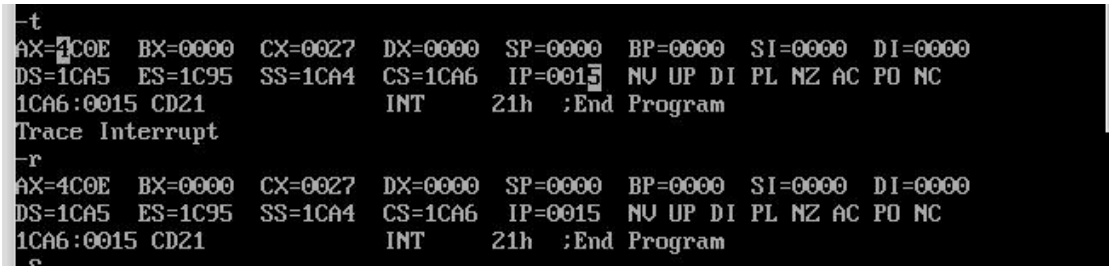

执行 ADD AL,[0002] 能够看到 AX 的低位部分发生了变化，加上了 C 的值




执行 SUB AL,[0003] 能够看到 AX 的低位部分发生了变化，减了 D 的值




执行 MOV [0000],AL 能够看到将 AX 低位部分存储到内存单元中


问题四、根据本次实验所学到知识，解答下面问题？

1. 设 A=186，B=273Q，C=0BBH，它们之间的关系是（ D ）

A.A>B>C B.A<B<C

C.A=B=C D.A<B=C

2. 8086／8088 的存贮器组织是将存储器划分为段，可作为段的起始地址是（ BC ）

A.185A2H B.00020H

C.01004H D.0AB568H

3、某存储单元的段地址是 0AB90H，偏移地址是 1234H，该存储单元的物理地址是（ B ）

A.0BDC4H B.0ACB34H

C.0AD134H D.1CED0H

4、某存储单元的物理地址是 12345H，可以作为它的段地址有（ D ）

A．2345H B．12345H

C．12340H D．1234H

5、已知某操作数的物理地址是 2117AH，则它的段地址和偏移地址可能是( A )。

A.2025：0F2A B.2108：00EA

C.2000：017A D.2100：117A

6、已知 SP=2110H,执行 POP AX 后，SP 寄存器的值是( B )。

A.2111H B.2112H

C.210FH D.210EH

7、设物理地址(21000H)=20H,(21001H)=30H，(21002H)=40H。如从地址 21001H 中取出一个字的内容是( D )

A.2030H B.3040H

C.3020H D.4030H

8、 设 SP 的初值为 1000H，执行指令 PUSH AX 后 SP 的值是( C )

A.0FFFH B.1001H

C.0FFEH　　D.1002H

9、假设存储器中从 7462H 单元开始的四个相邻字节单元中的内容依次是 32H，46H，52H，OFEH，则存放字数据 OFE52H 的字地址是（　C　）

A.7462H　　B.7463H

C.7464H　　D.7465H

10、若 AX=3500H,CX=56B8H，当 AND AX，CX 指令执行后，AX=(　A　)

A. 1400H　　B. 77F8H

C. 0000H　　D. 0FFFFH

扩展题目

下面是数据段的定义，用程序的方式解答下面的问题

```
.data
No1 dw 12
No2 db 20 dup(30)
No3 dd 34
No4 equ $-No1
```

No4 的值是　　C

A.56　　B.78

C.1AH　　D.27

我构建了以下程序：通过 BX 可以知道 No4 的值

```
8  DATA     SEGMENT            ;数据段定义
9  No1 dw 12
10 No2 db 20 dup(30)
11 No3 dd 34
12 No4 equ $-No1
13 DATA     ENDS
14 CODE     SEGMENT            ;代码段定义
15 ASSUME  CS:CODE,DS:DATA     ;ASSUME语句
16 START:  MOV AX,DATA         ;装填DS
17      MOV    DS,AX
18      MOV    BX,No4
19
20      MOV    AH,4CH          ;返回DOS
21      INT      21H
22    CODE     ENDS
23      END    START           ;结束语言
```

```
Trace Interrupt
-t
AX=1CA5  BX=0000  CX=002C  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=1CA5  ES=1C95  SS=1CA4  CS=1CA7  IP=0005  NV UP DI PL NZ NA PO NC
1CA7:0005 BB1A00          MOV     BX,001Ah
Trace Interrupt
-t
AX=1CA5  BX=001A  CX=002C  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=1CA5  ES=1C95  SS=1CA4  CS=1CA7  IP=0008  NV UP DI PL NZ NA PO NC
1CA7:0008 B44C            MOV     AH,4Ch
Trace Interrupt
-
```