实验五　子程序结构仿真操作

一、实验目的

熟悉 Keil 环境，通过在 Keil 环境下调试汇编子程序，掌握汇编语言程序的调试方法及设计方法，加深对汇编语言子程序结构、循环、寄存器、数据指针、汇编语言指令、机器码、片外数据操作等基本概念的理解，为后续程序编制和调试打下基础。同时掌握汇编语言相关算法设计的技巧。

二、 实验内容

以 4050H 为起始地址的外存储区中，存放有 16 个双字节无符号十六进制数（高位在后），试编写一个程序，

1、用循环结构给这 16 个数赋值。从大到小顺序排列。

2、用子程序实现这 16 个数的平均值，整数存入 4080H、4081H 单元(低位在前)，余数存在 4082H 单元。

3、用子程序实现这 16 个数的排序按从小到大顺序排列。

4、用主程序调用以上功能子程序实现整个功能。

三、实验步骤和结果

1、首先设计主程序调用结构，再根据主程序结构编写相应子程序。

结合上述实验要求，本次仿真实验需要完成的内容及相应子程序名称如下：

- 初始化 16 位双字节无符号数并存入 4050H 处的外存储区→INIT
- 求取 16 个数的平均数，将结果存入相应存储区→AVERAGE
- 将 16 个双字节无符号数按从小到大顺序排序并存入内存→SORT

由此可以设计三个子程序:INIT、AVERAGE、SORT。分别用于初始化、求平均数、。排序。因此设计相应主程序如下所示：

```
MAIN: MOV A,#20H   ;初始值低位
      MOV DPTR,#4050H
      MOV R2,#10H

      ACALL INIT
      ACALL AVERAGE
      ACALL SORT
      LJMP DONE
```

其中#20H 是设置数据的低位初始值，#10H 是数值个数，4050H 是片外起始地址，后面是相应的三个子程序，最后完成结果。

2、编写初始化赋值子程序 INIT

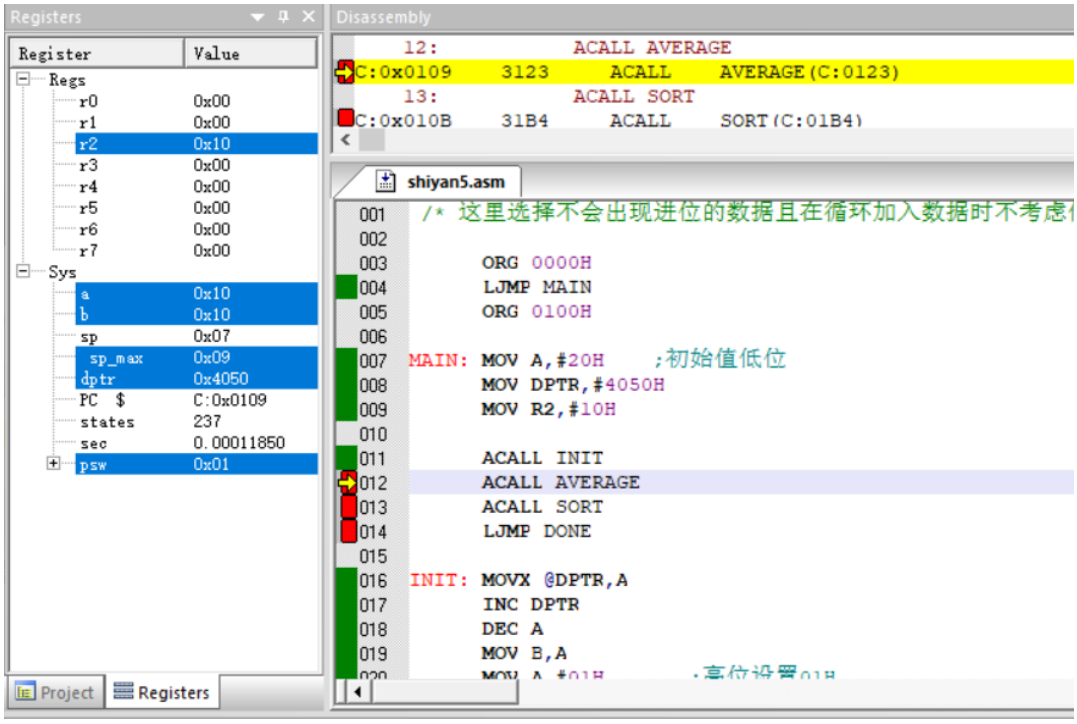
结合题目要求，通过循环向 4050H 为起始地址的外存储区加入 16 个双字节无符号数，其中高位在后，低八位在前，并从大到小排列。该循环中需要如下操作：

- 设置一个计时器为 16 用于中断循环，即初始化 16 个双字节数
- 循环中不断改变片外地址，同时数据不断递减以实现按从大到小顺序排列。

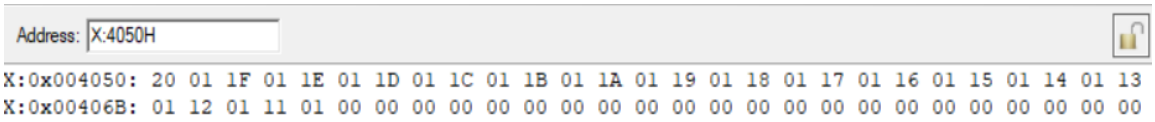
结合上述思路设计相应程序如下所示：

```
INIT: MOVX @DPTR,A
      INC DPTR
      DEC A
      MOV B,A
      MOV A,#01H        ;高位设置 01H
      MOVX @DPTR,A
      MOV A,B
```

结合主程序中的初始值设置，该子程序实现将 0120H 值依次递减存入片外 4050H 起始地址处，本次设置数据高位为 01，低位递减，低八位在前，高八位在后，keil 中设置断点调试结果如下所示：



图一：断点设置调试图



图二：数据初始化完成

由图可知，在 4050H 起始处，0120H、011F、…、0111H 等 16 个双字节 16 进制数按由大到小顺序依次存入相应内存，结果符号条件。

3、编写平均值子程序 AVERAGE

实现求取上述 16 个双字节的平均数首先需要求和，接着除以数量，相应操作如下：

- 双字节求和，通过循环求取上述值的和，先低位相加，再高位相加，结果存到 20H，21H 处，22H 存储进位
- 进行多字节除数操作，获取平均值，商存入 4080H、4081H 单元，余数存在 4082H 单元。

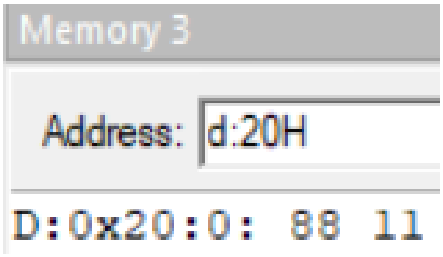
3.1 双字节求和

设置循环从 4050H 处开始读取数值，不断相加求和，先低位相加，保存进位信息，再高位相加，加上低位的进位信息，相应程序如下：

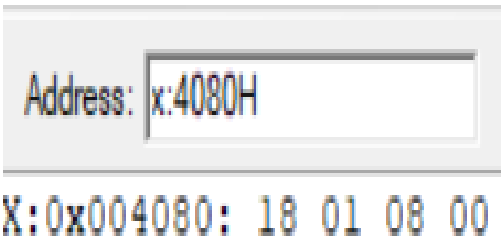
```
//求得所有的和，和存在 20H,21H 处，进位信息存在 22H 处。

AVERAGE: MOVX A,@DPTR
         MOV B,A
         MOV A,20H
         ADD A,B
         MOV 20H,A
         MOV A,21H
         ADDC A,#00H
         MOV 21H,A
         INC DPTR
         INC DPTR
         DJNZ R2,AVERAGE
         MOV DPTR,#4051H
         MOV R2,#10H
         SJMP SUMHIGH
```

```
SUMHIGH: MOVX A,@DPTR
         MOV B,A
         MOV A,21H

         ADD A,B
         MOV 21H,A
         MOV A,22H
         ADDC A,#00H
         MOV 22H,A
         INC DPTR
         INC DPTR
         DJNZ R2,SUMHIGH

    MOV R5,#10H//传递除数
    MOV R6,21H //被除数
    MOV R7,20H
    SJMP DIV_H2
         RET
```

上述程序中 AVERAGE 中将低位相加保存进位信息，SUMHIGH 中将高位相加保存进位信息，因为双字节，所以依次循环 DPTR 值加 2，由此可得所有数值的和如下图：



图三：数值和



图四：平均数

由图三可知，上述数值和在内存中为 1188H，这和上述 16 个双字节数相加结果一样，说明加法结果正确。

3.2 双字节除法

将上述总和除以数量便是平均值，在除法中，R6、R7 存被除数，R5 存除数，经过运算后，R6、R7 存商，R1 存余数。其中被除数除数的传递在求和中已经实现，上面代码已给出。因为本次除数只有八位，所以双字节除法运算算法如下：

1、 被除数高八位除以除数，所得结果商存入 R6,作为商高八位，R7 存储商的低八位；余数存入 R5，用于后续运算。

2、 余数左移一位，被除数低八位左移一位，判断被除数左移是否存在进位，若进位说明被除数此时最高位为 1，此时余数需要加一变为新的余数，被除数则是原被除数左移之后的结果。

3、 判断余数和除数的大小，若余数大于等于除数，则余数=余数-除数，此时商左移一位并加一(相当于除法中商上 1);若余数小于除数，不够除，此时商上 0，即商仅左移一位。

4、 判断被除数低八位是否左移八次，若是则退出循环，否则继续 2、3 操作直到左移八次。

5、 最后将 R6、R7 中的商，R1 中的余数存入 4080H 开始的地址处，退出平均数子程序

结合上述除数算法思路，得到程序如下所示：

```
DIV_H2:;除数高 8 位为 00H，被除数高 8 为非 00H，则余数肯定是单字节
    MOV     A,R5
    MOV     R0,A      ;R0 暂存除数
    MOV     B,A
    MOV     A,R6
    DIV     AB              ;被除数高 8 位除以余数
    JB      OV,UIDIV_END;检查到除数低 8 位也为 00H，直接结束程序
    MOV     R6,A
    MOV     R5,B  ;商存在 R6，余数存在 R5
    MOV     B,#08H    ;移位相减共 08H 次
```

```
UIDIV_LOOP2:          ;低 8 位运算
      MOV          A,R7
      RLC          A     //看看是否进位
      MOV R7,A
      JC     NEXT   //进位说明余数后面加 1,此时玉树左移并加上 1
      SJMP JIA0     //不进位余数后面加 0

JIA0: MOV A,R5
        RLC A         //得到余数，要判断余数和除数的大小
        MOV R5,A   //余数存到 R5
        MOV 10H,R0                //用 10H 临时储存除数的那个值。
        CJNE A,10H,CHANGE          //有进位跳转,有进位余数小
        SJMP NEXT2

NEXT: MOV A,R5
        ADD A,R5
        ADD A,#01H     //得到余数，要判断余数和除数的大小
        MOV R5,A      ///余数存到 R5
        MOV 10H,R0
        CJNE A,10H,CHANGE
        SJMP NEXT2

CHANGE:     JNC NEXT2
                SJMP NEXT1

NEXT1:
//余数小，不够减
      MOV A,R1
      ADD A,R1
      MOV R1,A
      DJNZ    B,UIDIV_LOOP2
      SJMP DONE1
NEXT2: //余数够减
      SUBB A,R0
      MOV R5,A
      MOV A,R1
      ADD A,R1
      ADD A,#01H
      MOV R1,A
      DJNZ      B,UIDIV_LOOP2
      SJMP DONE1
UIDIV_END:
      SJMP DONE1
```

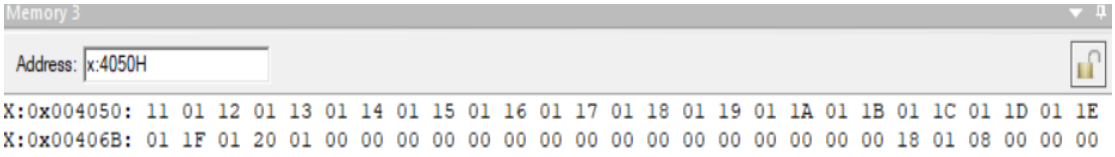
```
DONE1:   //结果存入外存储区
      MOV 22H,R1
      MOV 23H,R6
      MOV 24H,R5
      MOV A,22H
      MOV DPTR,#4080H
      MOVX @DPTR,A
      INC DPTR
      MOV A,23H
      MOVX @DPTR,A
      INC DPTR
      MOV A,24H
      MOVX @DPTR,A
      RET
```

具体实现见上述程序即程序注释，由此得到除法，算出平均数，观察内存地址，如图四平均数所示，可以发现：商存入 4080H、4081H，低八位在前，余数存在 4082H，所得结果是商 0118H，余数 08H，与 1188H 除以 10H 计算结果相同，因此平均数计算结果正确完成要求。

4、编写排序子程序 SORT

考虑到最初存入数据按照从大到小顺序排列，现在要求按照从小到大顺序排列只需要第一个和最后一个交换位置、第二个和倒数第二个交换位置、…依次类推，但考虑到片外 DPTR 不好实现地址跳转，因此考虑先将数据转移到片内，在进行相应排序操作，之后再转移到片外相应地址处。结合上述思路得到程序如下：

```
SORT: MOV DPTR,#4050H
      MOV R0,#30H
      MOV R2,#20H
      SJMP MOVE

MOVE: MOVX A,@DPTR
      MOV @R0,A
      INC R0
      INC DPTR
      DJNZ R2,MOVE
      MOV R0,#4EH
      MOV R2,#10H
      MOV DPTR,#4050H
      SJMP PAIXU

PAIXU: MOV A,@R0
       MOVX @DPTR,A
       INC DPTR
       INC R0
       MOV A,@R0
       MOVX @DPTR,A
       INC DPTR
       DEC R0
       DEC R0
       DEC R0
       DJNZ R2,PAIXU
       RET
```



上述程序按照前述思路实现数据按照从小到大排序，其结果如上图所示。

可以看到，片外 4050H 起始地址之后的 16 个双字节无符号数按照从小到大顺序排列，因此满足实验仿真要求。

四、实验总结

本次仿真实验相对来说较为复杂，但是通过子程序调用，将问题转化为一个个子程序的设计问题，最终完成了实验仿真任务。通过本次实验，进一步掌握了片内、片外数据转递、分级调试，断点设置、循环设计等汇编语言相关问题，对 Keil 的使用进一步掌握，同时更加熟悉了单片机指令系统(本次实验自己总结了相应 51 单片机指令系统)。相应解决问题设计算法的能力也得到了一定的提升，总之本次实验虽然相对复杂，但也收获很多。

在子程序调节中，比较复杂的便是排序问题和平均数中的除法问题。排序问题开始主要是片外操作方式有限，后来通过牺牲内存的方式先转到片内在进行相关操作，成功解决问题。双字节除法问题更加复杂，由于指令系统只有单字节除法，因此需要自己进行设计，结合上课内容，将除法问题转化为加法问题，综合考虑各种情况，最终成功的写出了双字节除法程序，且该程序有一定适应性，只要是双字节除以单字节都可以进行计算。在这两个较难问题中，虽然解决问题过程较为复杂，但也收获很多，尤其是对片内、片外数据操作得到进一步的掌握！