**冒泡排序算法简介：**

（1）基本思想

冒泡排序的基本思想就是：从无序序列头部开始，进行两两比较，根据大小交换位置，直到最后将最大（小）的数据元素交换到了无序队列的队尾，从而成为有序序列的一部分；下一次继续这个过程，直到所有数据元素都排好序。

算法的核心在于每次通过两两比较交换位置，选出剩余无序序列里最大（小）的数据元素放到队尾。

（2）运行过程

冒泡排序算法的运作如下：

1、比较相邻的元素。如果第一个比第二个大（小），就交换他们两个。

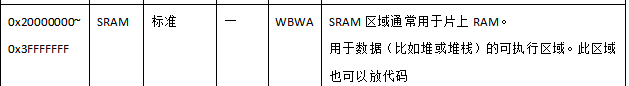
2、对每一对相邻元素作同样的工作，从开始第一对到结尾的最后一对。这步做完后，最后的元素会是最大（小）的数。

3、针对所有的元素重复以上的步骤，除了最后已经选出的元素（有序）。

4、持续每次对越来越少的元素（无序元素）重复上面的步骤，直到没有任何一对数字需要比较，则序列最终有序。

实验目的：实现输入乱序，输出从小到大排列。

基本设计思路：先将输入的数据（乱序）放在R1到R7寄存器，用存储器保存指令，将R1到R7的数据依次导入R0指向的存储器中。再用寄存器两两读取交换位置，遍历排序。最后用存储器加载指令，将存储器中存放的数据（已排序）依次导入R0到R7寄存器。用户进入调试界面即可看到输出数据在R0到R7从小到大依次排列。

下图所示存储器存储数据的地址如下：基址为0x2000 0000

排序部分的具体设计思路

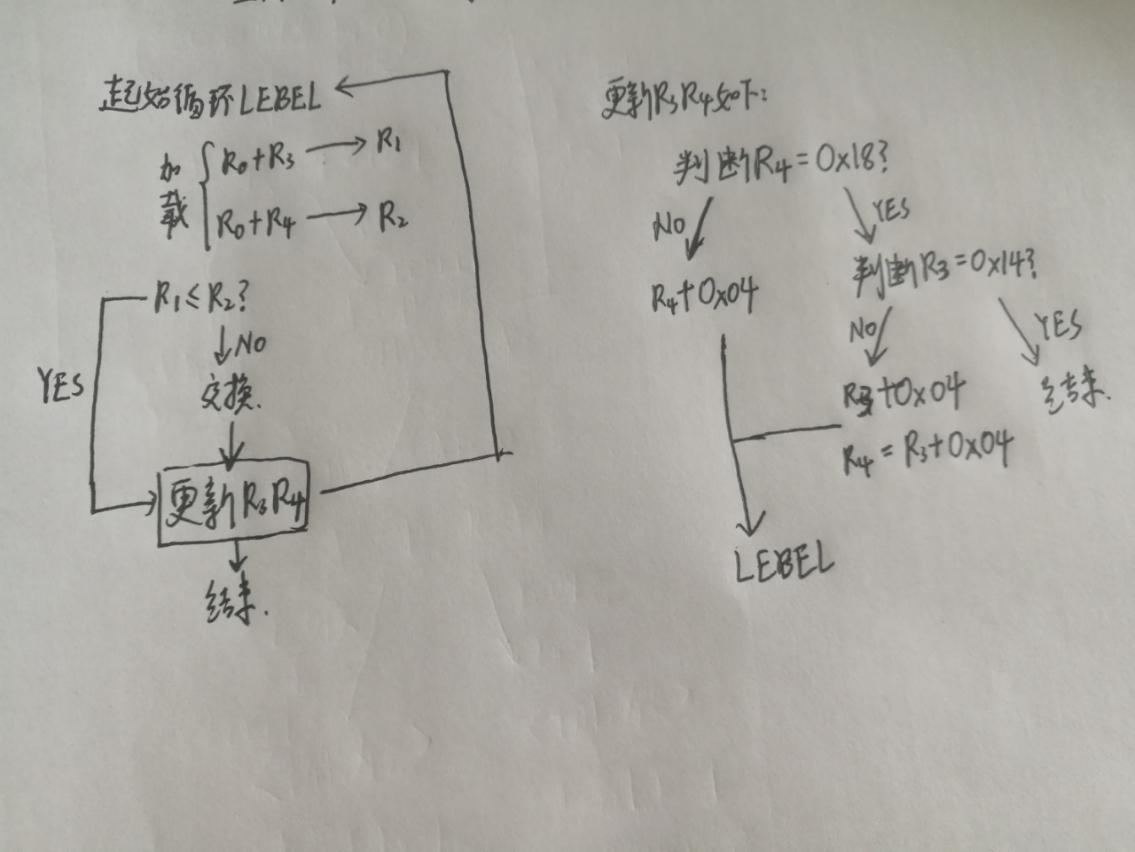
初始化：R0寄存器存储基址0x2000 0000

R1\R2加载被比较的两个数

R3\R4寄存器存储两个比较值的偏移地址

同时用R3和R4控制冒泡排序的循环

循环排序部分如下图：



设计代码如下：

MOVS R0, #0x20

MOVS R1, #24

LSLS R0, R1 ;R0指向存存储器基址0x2000 0000

MOVS R1, #0x85

MOVS R2, #0x6F

MOVS R3, #0xC2

MOVS R4, #0x1E

MOVS R5, #0x34

MOVS R6, #0x14

MOVS R7, #0x8E ;R1到R7存储需要排序的数据

STM R0!, {R1,R2-R7} ;将需要排序的数据导入R0指向的存储器中

MOVS R0, #0x20 ;R0 back

MOVS R1, #24

LSLS R0, R1 ;R0指针归位继续指向0x2000 0000

MOVS R3, #0x00 ; 初始R3指向第一个需要排序数据的偏移地址

MOVS R4, #0x04 ; 初始R4指向第二个需要排序数据的偏移地址

LEBEL ; 循环开始标志

LDR R1, [R0,R3] ;将第一个需要排序的数据从存储器加载到R1中

LDR R2, [R0,R4] ;将第二个需要排序的数据从存储器加载到R2中

CMP R1, R2 ;比较R1和R2的值

BLS LEBEL1 ;R1<=R2不需要交换跳转到LEBEL1

STR R1, [R0,R4] ;否则R1和R2需要交换

STR R2, [R0,R3] ;将R1/R2交叉保存在存储器中，完成交换

LEBEL1

CMP R4, #0x18 ;比较判断第二个被比较数是否遍历到最后一个

BEQ LEBEL2 ;如果是则跳转到LEBEL2

ADDS R4, #0x04 ;如果不是则指向下一个被比较数据的偏移地址

B LEBEL ;开始新的比较循环

LEBEL2

CMP R3, #0x14 ;比较判断第一个被比较数是否为倒数第二个数

BEQ LEBEL3 ;是则全部遍历完成，跳转到LEBEL3结束排序

ADDS R3, #0x04 ;如果不是，让R3指向下一个被比较的数

MOVS R4, R3

ADDS R4, #0x04 ;让R4指向R3后面的一个数

B LEBEL ;开始新的比较循环

LEBEL3

LDM R0!, {R1, R2-R7} ;将存储器中排列好的数字，依次导出到R1到R7

实验结果如下：左图为乱序的输入，右图为从存储器排序结果的导出

