

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

0x20000000~ 0x3FFFFFFF	SRAM	标准	—	WBWA	SRAM 区域通常用于片上 RAM。 用于数据（比如堆或堆栈）的可执行区域。此区域也可以放代码
---------------------------	------	----	---	------	--

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：

	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

```
MOVSR1, #0x85
MOVSR2, #0x6F
MOVSR3, #0xC2
MOVSR4, #0x1E
MOVSR5, #0x34
MOVSR6, #0x14
MOVSR7, #0x8E
```

Register	Value
Core	
R0	0x2000001C
R1	0x00000014
R2	0x0000001E
R3	0x00000034
R4	0x0000006F
R5	0x00000085
R6	0x0000008E
R7	0x000000C2