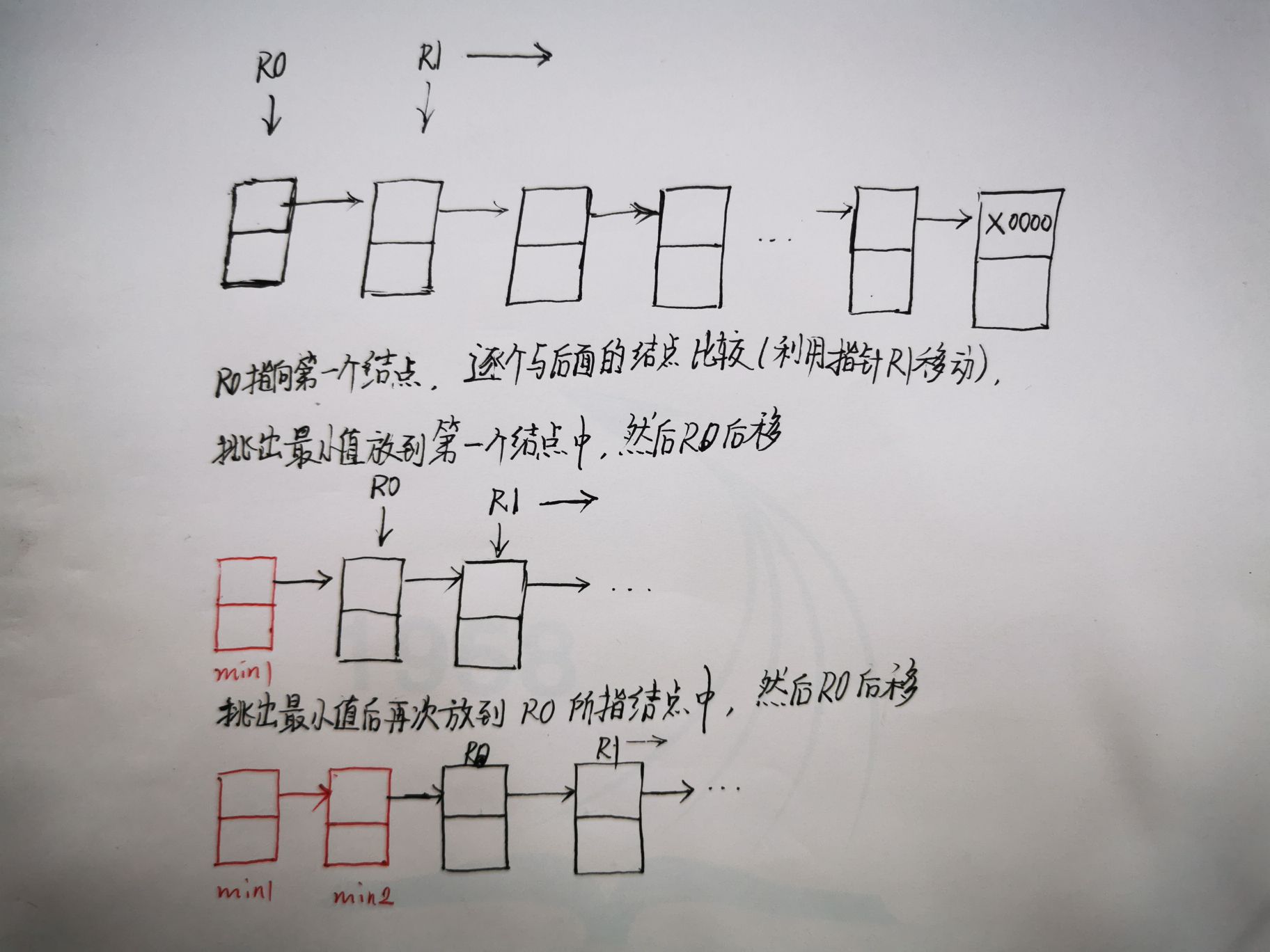
**LAB 03**

**梁峻滔 PB19051175**

**1. 算法思想**

目前学过的排序方法主要有选择排序和冒泡排序，由于冒泡排序是两个相邻的数依次比较、满足条件时交换，在排序过程中需要使用两个指针来跟踪所比较的操作数，而且要更新比较停止的位置，操作相对复杂；而选择排序是每一轮选出未排序序列中最小的，然后放到已排好序的序列的后面，只需要一个指针跟踪操作数，而且更新排序位置时相对容易(使该指针后移一个结点即可)，示意图如下图所示：



故采用选择排序方法。排序过程中使用R7记录最小值，待R1访问完所有未排序的结点后把R7存到R0所指结点的数据域中，然后R0指向下一个结点，再利用R1和R7来筛选剩余未排序的结点中的最小值，重复该过程直到R0指向最后一个结点。

**2. 代码实现**

****

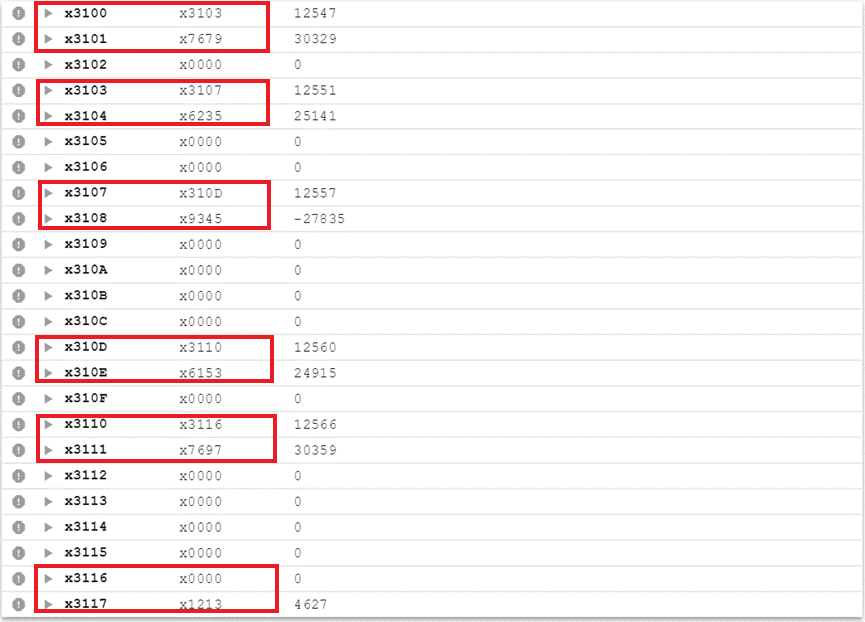
除了R0、R1和R7，程序中还使用了其他寄存器来存放执行过程中需要暂时保存的数据：R2存放R1所指结点的值，R3用于R7和R2相减比较，R4用于记录最小值结点的地址，R5用于两个结点交换数值过程中数值的转移，由于要给出最后一个结点的地址，在更新R0时使用了R6来判断R0的下一个结点地址是否为x0000，R6不为0时才更新R0，否则R0存的就是最后一个结点的地址。

**3. 测试**

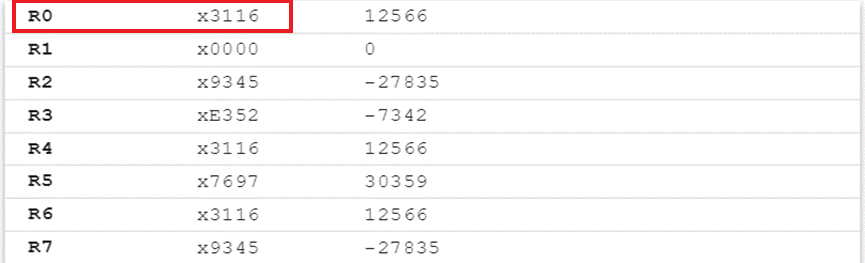
Case1：

初始状态：x3100为第一个结点地址，链表长度为6

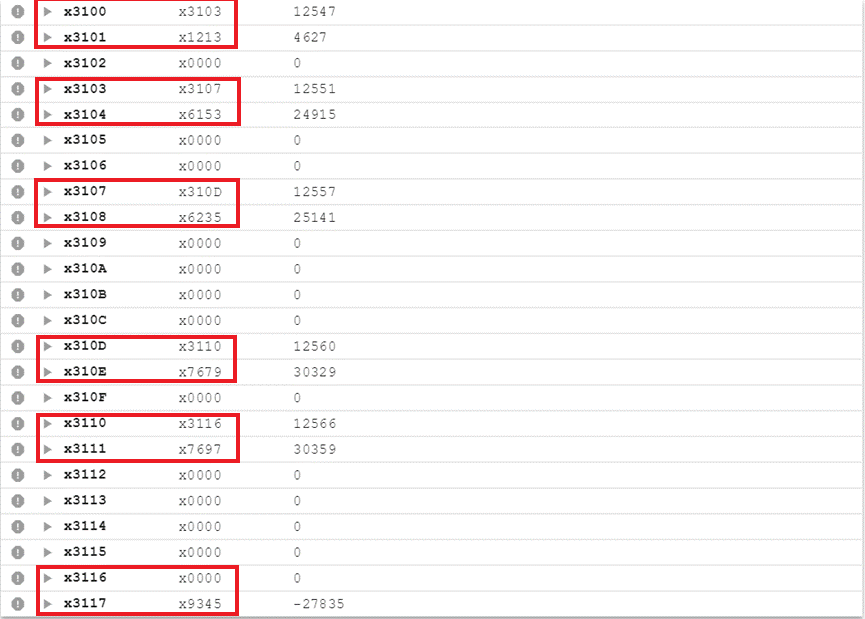




程序运行完成后：



R0存放的是x3116，是最后一个结点的地址。



链表正确地按升序排好了序。

Case2：

初始状态：第一个结点地址为x9000，链表长度为11

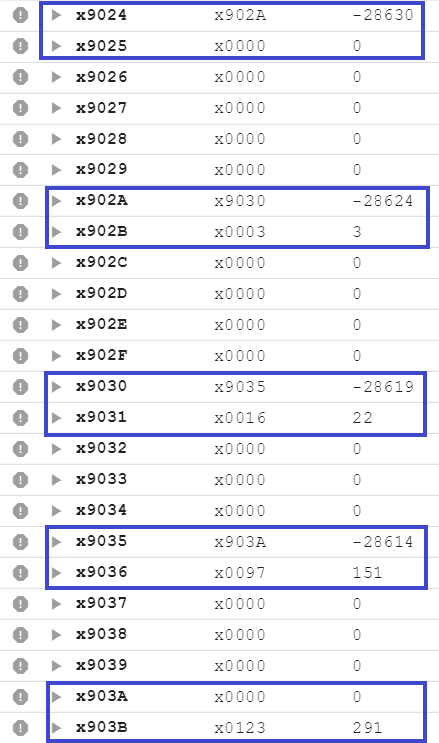




程序运行完成后：



R0存放的是x903A，是最后一个结点的地址。



case2加入了负整数，程序也正确地完成了升序排序，但测试过程中如果在相减时(正数减负数或负数减正数)出现了溢出，就不能正常排序。