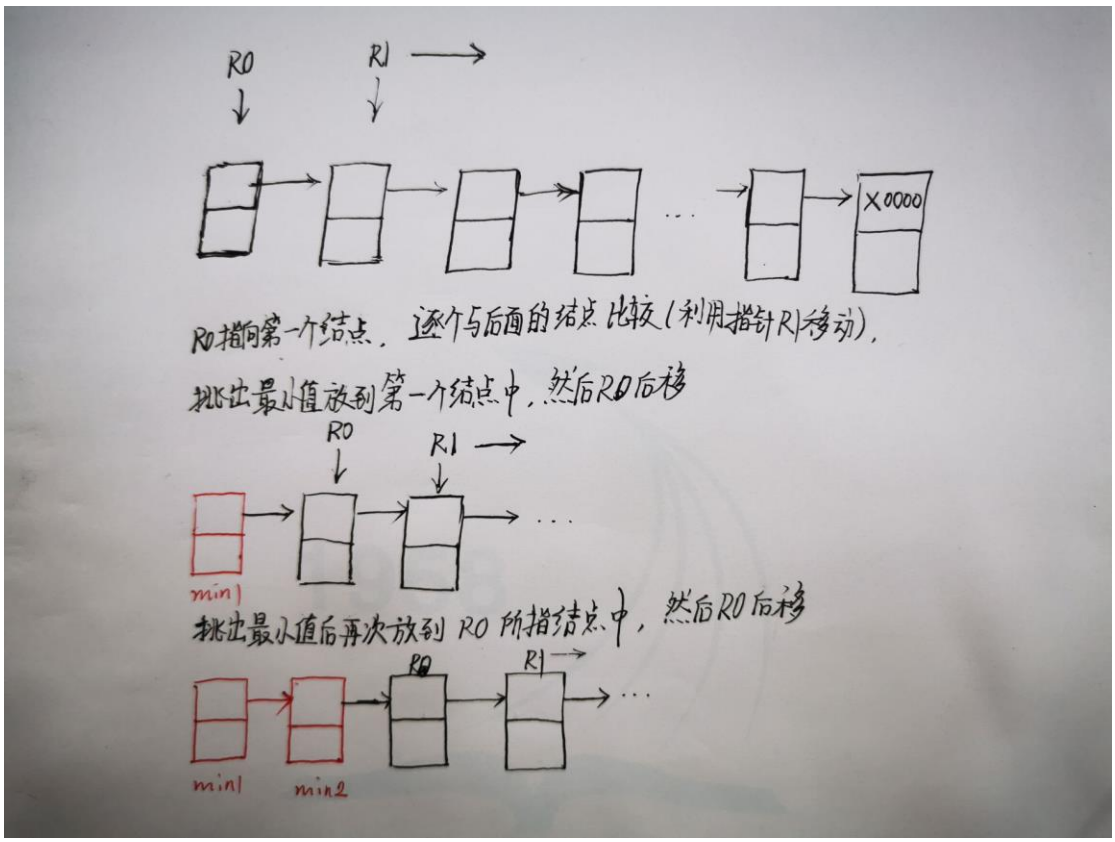
LAB 03

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

```
;对二进制补码表示的整数链表进行升序排序
;采用选择排序，每次都从未排序的数值中选择最小值min，并放到前面
;R0指向当前要存放min的位置
;R7用于比较和存放min值
;R1指向当前访问结点的地址
;R2存放当前结点的数值
        .ORIG x3000
;R0存放第一个结点的地址，这里为x3100,需要手动输入
UPDATER LDR R7,R0,#1
UPDATEA LDR R1,R0,#0      ;R1存放R0下一个结点的地址
        BRz FINISH        ;R0已经是最后一个结点
UPDATED LDR R2,R1,#1      ;R2存放R1所指结点的数值
;比较R7与当前访问结点数值的大小
        NOT R3,R2
        ADD R3,R3,#1
        ADD R3,R7,R3      ;R7-R2
        BRnz NEXT
        ADD R7,R2,#0      ;R2<R7时,更新R7
        ADD R4,R1,#0      ;更新R7时也把更新值的结点地址记录下来
;访问下一个结点
NEXT    LDR R1,R1,#0
        BRnp UPDATED
;如果不跳转，说明R1指向了x0000，这时已经把未排序数值中的最小值求出在R7
        LDR R5,R0,#1      ;把原先存放在R0结点的值取出
        STR R7,R0,#1      ;把求得的min值放进R0结点的值域
        STR R5,R4,#1      ;把取出的R0结点的值放进存有min值的结点中，即交换值域
        LDR R6,R0,#0      ;R6指向下一个要存放min值的结点
        BRz FINISH        ;当R6为x0000时，说明R0已经指向最后一个结点，排序完成
        ADD R0,R6,#0
        ADD R4,R0,#0      ;避免残留的R4对后面的排序造成影响
        BRnzp UPDATER
FINISH  HALT
        .END
```

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

R0	x3100	12544
R1	x0000	0
R2	x0000	0
R3	x0000	0
R4	x0000	0
R5	x0000	0
R6	x0000	0
R7	x0000	0

x3100	x3103	12547
x3101	x7679	30329
x3102	x0000	0
x3103	x3107	12551
x3104	x6235	25141
x3105	x0000	0
x3106	x0000	0
x3107	x310D	12557
x3108	x9345	-27835
x3109	x0000	0
x310A	x0000	0
x310B	x0000	0
x310C	x0000	0
x310D	x3110	12560
x310E	x6153	24915
x310F	x0000	0
x3110	x3116	12566
x3111	x7697	30359
x3112	x0000	0
x3113	x0000	0
x3114	x0000	0
x3115	x0000	0
x3116	x0000	0
x3117	x1213	4627

程序运行完成后：

R0	x3116	12566
R1	x0000	0
R2	x9345	-27835
R3	xE352	-7342
R4	x3116	12566
R5	x7697	30359
R6	x3116	12566
R7	x9345	-27835

R0 存放的是 x3116，是最后一个结点的地址。

x3100	x3103	12547
x3101	x1213	4627
x3102	x0000	0
x3103	x3107	12551
x3104	x6153	24915
x3105	x0000	0
x3106	x0000	0
x3107	x310D	12557
x3108	x6235	25141
x3109	x0000	0
x310A	x0000	0
x310B	x0000	0
x310C	x0000	0
x310D	x3110	12560
x310E	x7679	30329
x310F	x0000	0
x3110	x3116	12566
x3111	x7697	30359
x3112	x0000	0
x3113	x0000	0
x3114	x0000	0
x3115	x0000	0
x3116	x0000	0
x3117	x9345	-27835

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

R0	x9000	-28672

x9000	x9005	-28667	x9024	x902A	-28630
x9001	xFFFF	-1	x9025	x0123	291
x9002	x0000	0	x9026	x0000	0
x9003	x0000	0	x9027	x0000	0
x9004	x0000	0	x9028	x0000	0
x9005	x9008	-28664	x9029	x0000	0
x9006	xEEEE	-4370	x902A	x9030	-28624
x9007	x0000	0	x902B	x0097	151
x9008	x900E	-28658	x902C	x0000	0
x9009	xFEFA	-262	x902D	x0000	0
x900A	x0000	0	x902E	x0000	0
x900B	x0000	0	x902F	x0000	0
x900C	x0000	0	x9030	x9035	-28619
x900D	x0000	0	x9031	x0016	22
x900E	x9013	-28653	x9032	x0000	0
x900F	x0000	0	x9033	x0000	0
x9010	x0000	0	x9034	x0000	0
x9011	x0000	0	x9035	x903A	-28614
x9012	x0000	0	x9036	xFABC	-1348
x9013	x9016	-28650	x9037	x0000	0
x9014	x0003	3	x9038	x0000	0
x9015	x0000	0	x9039	x0000	0
x9016	x9024	-28636	x903A	x0000	0
x9017	xFCDE	-802	x903B	xFFFE	-2

程序运行完成后：

R0	x903A	-28614

R0 存放的是 x903A，是最后一个结点的地址。

x9000	x9005	-28667
x9001	xEEEE	-4370
x9002	x0000	0
x9003	x0000	0
x9004	x0000	0
x9005	x9008	-28664
x9006	xFABC	-1348
x9007	x0000	0
x9008	x900E	-28658
x9009	xFCDE	-802
x900A	x0000	0
x900B	x0000	0
x900C	x0000	0
x900D	x0000	0
x900E	x9013	-28653
x900F	xFEFA	-262
x9010	x0000	0
x9011	x0000	0
x9012	x0000	0
x9013	x9016	-28650
x9014	xFFFE	-2
x9015	x0000	0
x9016	x9024	-28636
x9017	xFFFF	-1

x9024	x902A	-28630
x9025	x0000	0
x9026	x0000	0
x9027	x0000	0
x9028	x0000	0
x9029	x0000	0
x902A	x9030	-28624
x902B	x0003	3
x902C	x0000	0
x902D	x0000	0
x902E	x0000	0
x902F	x0000	0
x9030	x9035	-28619
x9031	x0016	22
x9032	x0000	0
x9033	x0000	0
x9034	x0000	0
x9035	x903A	-28614
x9036	x0097	151
x9037	x0000	0
x9038	x0000	0
x9039	x0000	0
x903A	x0000	0
x903B	x0123	291

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。