

# 中国科学技术大学计算机学院 《计算机系统概论》实验报告



RISC-V LAB 03: The Linked-List Sort

姓名:王晨      学号:PE20060014

完成日期:2020 年 12 月 21 日

## 一、实验要求：

用RISC-V指令集重新实现对链表的排序。

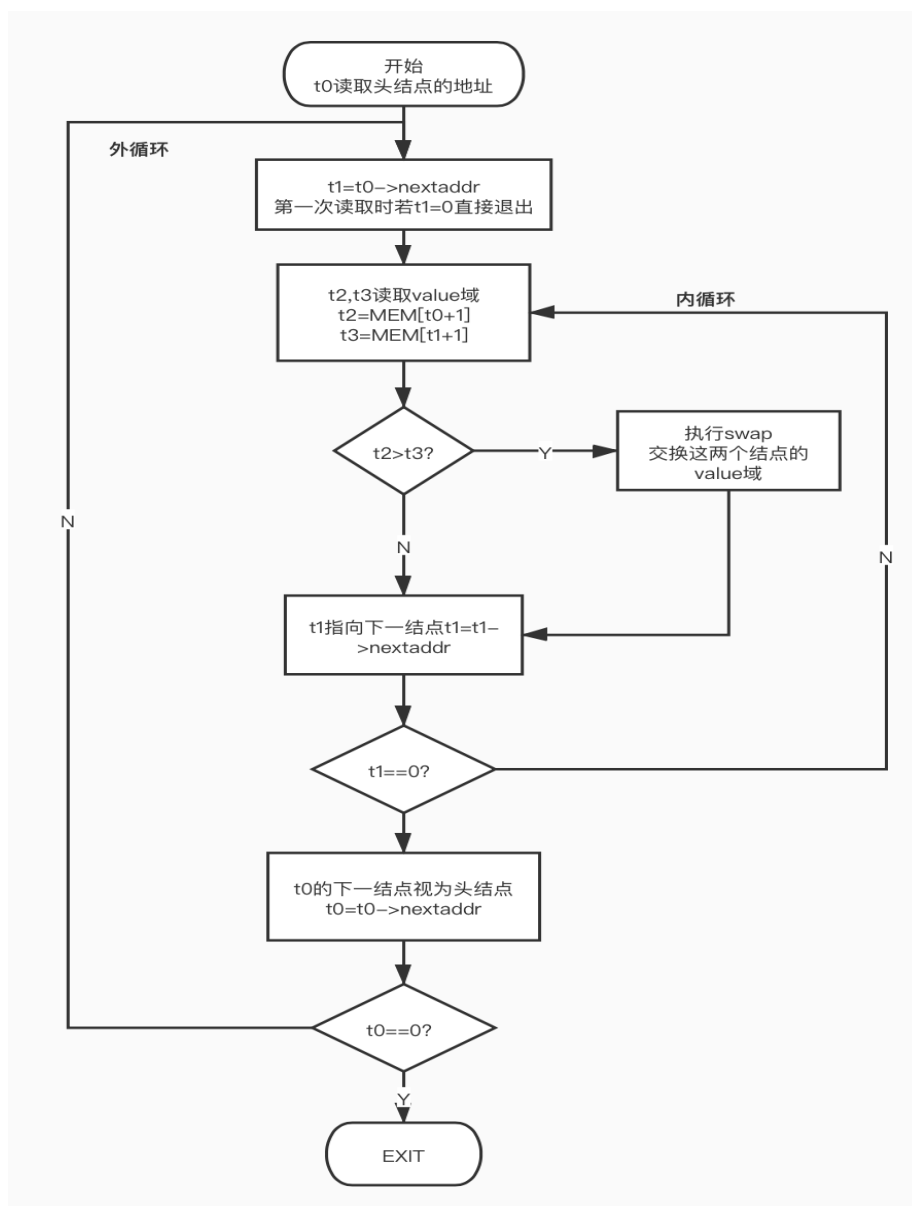
## 二、实验环境：

Mac OS Big Sur 11.01

LC-3 Simulator Mac Version

## 三、算法思路：

同LC-3的实现方式相同，采用冒泡排序算法进行排序，两层循环，内循环执行比较和交换功能，外循环执行找到第  $i$  小的元素功能，外循环每执行一次可以使得第  $i$  个元素比它后面的元素都小，从而完成升序排序。本算法仅交换value域，首结点地址为  $x10010020$ ，尾结点的address域为  $x00000000$ 。下图是算法流程图：



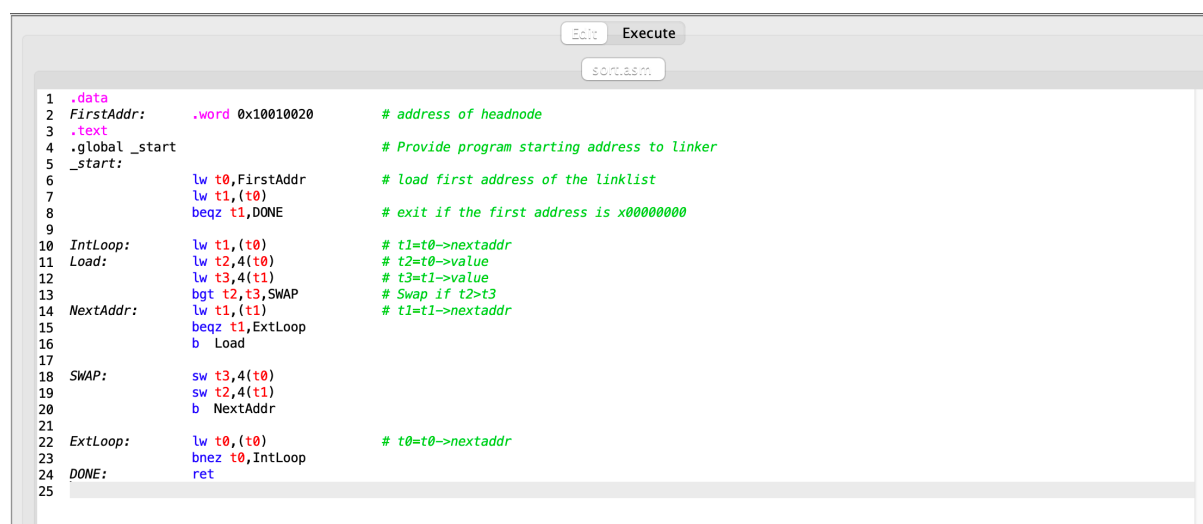
寄存器说明：t0指向第i次要确认的最小元素的位置，也就是第i小元素的地址。

t1指向第j个（j>i）元素的地址。

t2, t3分别读取t0, t1的value域。

每一次比较和交换结束后，t1=t1->nextaddr，即t1指向下一个元素位置，再同t0位置的元素进行比较。当t1=x0时，说明一轮循环结束，t0位置的元素已经是最小的值了，然后将t0=t0->nextaddr继续依次比较后面的元素，直到最终R0=x0完成排序。

## 四、程序代码及注释：



```

1  .data
2  FirstAddr: .word 0x10010020      # address of headnode
3  .text
4  .global _start
5  _start:
6      lw t0,FirstAddr             # load first address of the linklist
7      lw t1,(t0)
8      beqz t1,DONE                # exit if the first address is x00000000
9
10 IntLoop:
11     lw t1,(t0)                  # t1=t0->nextaddr
12     lw t2,4(t0)                 # t2=t0->value
13     lw t3,4(t1)                 # t3=t1->value
14     bgt t2,t3,SWAP              # Swap if t2>t3
15     lw t1,(t1)                  # t1=t1->nextaddr
16     beqz t1,ExtLoop
17     b Load
18
19 SWAP:
20     sw t3,4(t0)
21     sw t2,4(t1)
22     b NextAddr
23
24 ExtLoop:
25     lw t0,(t0)                  # t0=t0->nextaddr
26     bnez t0,IntLoop
27
28 DONE:
29     ret

```

1. #冒泡排序，两层循环，第i次外循环结束，使得链表的第i个元素比它之后的元素都要小（只交换value域）。
2. .data
3. FirstAddr .word 0x10010020 # address of headnode
4. .text
5. .global \_start # Provide program starting address to linker
6. \_start:
7. lw t0,FirstAddr # load first address of the linklist
8. lw t1,(t0)
9. beqz t1,DONE # exit if the first address is x00000000
- 10.
11. IntLoop: lw t1,(t0) # t1=t0->nextaddr

```

12. Load:    lw t2,4(t0)           # t2=t0->value
13.          lw t3,4(t1)           # t3=t1->value
14.          bgt t2,t3,SWAP         # Swap if t2>t3
15. NextAddr: lw t1,(t1)           # t1=t1->nextaddr
16.          beqz t1,ExtLoop
17.          b Load
18.
19. SWAP:     sw t3,4(t0)
20.          sw t2,4(t1)
21.          b NextAddr
22.
23. ExtLoop: lw t0,(t0)             # t0=t0->nextaddr
24.          bnez t0,IntLoop
25. DONE:    ret

```

## 五、调试和测试：

测试任意长度且包含不同大小的正数、负数、零的链表。为了便于检查，链表结点都放在一页Memory中，起始地址为x10010020，末尾地址为x00000000。

执行前的初始链表如下所示，链表长度为6，头结点的point域为x10010040，value域为x10：

Data Segment								
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x10010040	0x00000010	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x10010060	0x00000007	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x10010080	0x00000002	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x100100a0	0x00000001	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x100100c0	0x80000001	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x8000000e	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

程序执行的结果如下图所示，可以看到完成了升序排列：

Data Segment								
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x10010040	0x80000001	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x10010060	0x8000000e	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x10010080	0x00000001	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x100100a0	0x00000002	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x100100c0	0x00000007	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000010	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

## 六、小结：

采用了冒泡排序，对于N个结点的链表来说，时间复杂度为 $O(N)$ ，空间复杂度为 $O(1)$ 。这种方法的优点是程序思路非常简单，对于不太长的链表（指数级以下）可以很好的完成排序，且基本不需要额外空间暂存排序过程中的变量。

主循环体指令条数为12条，整个程序使用寄存器t0~t4，可以说程序是非常简洁的。由于用RISC-V指令集重新实现对链表的排序，方法和LC-3指令集是相同的，注释也已写上，因此就不再赘述了。