RISCV 链表排序

实验要求相关同LC-3实验。

实验思路

首先构建链表，之后采用冒泡排序算法对其进行排序。

冒泡链表排序法的主要思想是，设置一个尾指针（假设为R1），从头结点开始扫描，如果当前结点（R2) 的value大于下一个value，则交换二者的value值，直到当前节点等于尾节点为止，当尾结点等于头结点时结束。

代码解释分析

①通过指令向内存中存入希望排序的初始值：(这里举其中一个例子)

```
.equ CONSTANT1,0x00000056
.equ CONSTANT2,0x00000034
.equ LOCA1,0x3100
.equ LOCA2,0x3500
.equ LOCA3,0x3200
.equ LOCAFINAL,0x5000
lui a0,LOCA1
lui a1,LOCA2
sw a1,0(a0)
lui   a0,LOCA2
lui a1,LOCA3
sw a1,0(a0)
lui   a0,LOCA3
lui a1,LOCAFINAL
sw a1,0(a0)


lui   a0,LOCA2
addi a0,a0,4
lui a1,CONSTANT1
sw a1,0(a0)
lui   a0,LOCA3
addi a0,a0,4
lui a1,CONSTANT2
sw a1,0(a0)



lui   a0,LOCA1
lw t0,0(a0)
lui   a0,LOCA2
lw t1,0(a0)
lui   a0,LOCA3
lw t2,0(a0)
```

其中数值CONSTANT1,2分别存放在LOCA2和LOCA3中。最后查看最终结果使用命令：

```
lui   s0,LOCA2
addi s0,s0,4
lw s2,0(s0)
lui   s1,LOCA3
addi s1,s1,4
lw s1,0(s1)
```

将LOCA2,3储存的内容load到s1和s2寄存器中。

②初始化一些寄存器的值

```
andi a0,a0,0
andi a1,a1,0
andi a2,a2,0
andi a3,a3,0
andi a4,a4,0
andi a5,a5,0
andi a6,a6,0
andi a7,a7,0
lui  a0, LOCA1 #a0储存起始点
lui  a1,LOCAFINAL #a1储存LOCA2的值
lw a0,0(a0)
```

在初始时刻开始部分位置是x3100，结束部分的位置在x5000。（这里由于代码初始位置在0x0处，所以这里选取x5000作为链表结束的标志。

```
JUDGE1: andi a5,a5,0
lw a5,0(a0)
not a2,a5
addi a2,a2,1
add a2,a2,a1
beqz a2,THEEND1
```

在这个JUDGE1块中，程序判断R0的下一个结点是否等于R1，如果相等，则跳出循环，进入THEEND1块，否则进入BUBBLE循环。这里同时考虑了负数的情况，如果链表中的数据存在负数的情况，如果R3的值对应为负数，继续查看R4的值，如果也是负数，则将其都翻转为正数后比较，如果R3的相反数小于R4的相反数，那么执行SWAP交换。如果R3的值为正数，对R4进行判断，如果R4是负数，那么执行SWAP交换，否则正常操作。

```
BUBBLE: addi a6,a0,4
lw a3,0(a6)
bltz a3,NEGA3
NEGABACK1:
lw a6,0(a0)
addi a6,a6,4
lw a4,0(a6)
bltz a4,SWAP
NEGABACK2:
not a5,a3
addi a5,a5,1
```

```
add a5,a5,a4
bltz a5,SWAP
jal NEXT

NEGA3:
lw a6,0(a0)
addi a6,a6,4
lw a4,0(a6)
bltz a4,NEGA4
jal NEXT

NEGA4:
not a4,a4
addi a4,a4,1
not a3,a3
addi a3,a3,1
not a5,a3
addi a5,a5,1
add a5,a5,a4
bgtz a5,SWAPNEGA
jal NEXT

SWAPNEGA:
not a4,a4
addi a4,a4,1
not a3,a3
addi a3,a3,1
jal SWAP


SWAP:
addi a7,a0,4
sw a4,0(a7)
lw a7,0(a0)
addi a7,a7,4
sw a3,0(a7)
jal NEXT

NEXT:
lw a0,0(a0)
jal JUDGE1
```

在BUBBLE循环中，取出R0对应的value值放入R3中，并取出R0所存的地址对应的value放入R4中，判断R3和R4的大小，如果R3>R4，则交换这两个值的位置（跳入SWAP块），将R4的值存入R0的下一个地址位置，将R3存入R0中存的地址对应的value的位置。进入NEXT模块之后，将R0的值改变为R0所存的地址的值，即进入下一个结点，再次进入JUDHE1模块，开始循环。

```
THEEND1: addi a1,a0,0
lui  a0, LOCA1
lw a0,0(a0)
jal JUDGE2
```

如果R0的下一个结点是R1,则这层循环结束，将R1的值换成它的前一个结点，即R0的值，进入下一个循环。

```
JUDGE2: andi a5,a5,0
andi a2,a2,0
lui      a5, LOCA1
lw a5,0(a5)
not a2,a1
addi a2,a2,1
add a2,a5,a2
beqz a2,THEEND2
jal JUDGE1
```

如果R1等于初始结点的下一个结点，循环结束，否则，继续JUDGE1开始的循环。

测试用例

【测试用例1】

初始化的代码如下所示：

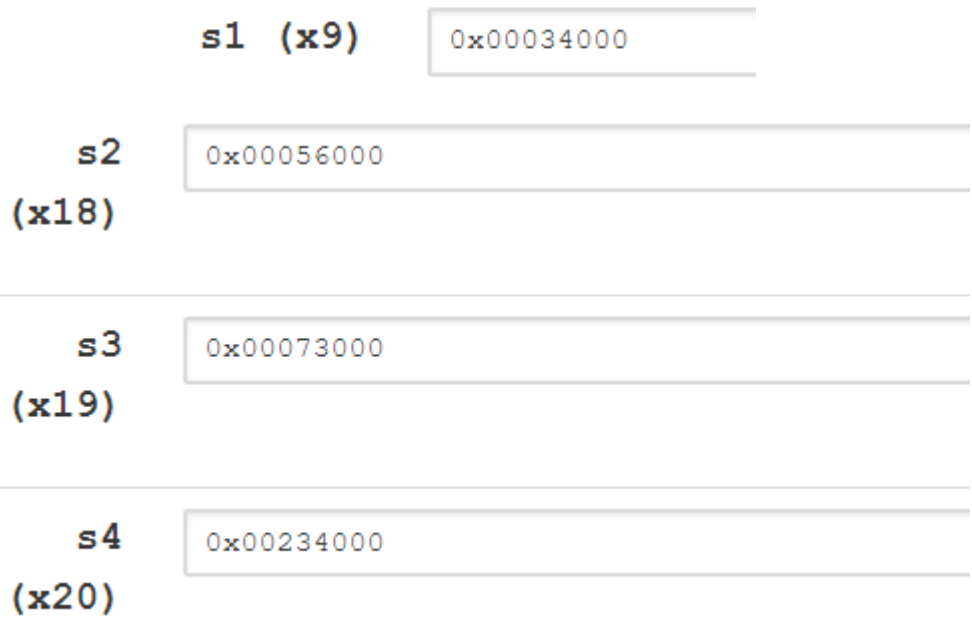
```
.equ CONSTANT1,0x00000056
.equ CONSTANT2,0x00000034
.equ CONSTANT3 0x00000234
.equ CONSTANT4 0x00000073
.equ LOCA1,0x3100
.equ LOCA2,0x3500
.equ LOCA3,0x3200
.equ LOCA4,0x3126
.equ LOCA5,0x3108
.equ LOCAFINAL,0x5000
lui a0,LOCA1
lui a1,LOCA2
sw a1,0(a0)
lui   a0,LOCA2
lui a1,LOCA3
sw a1,0(a0)
lui   a0,LOCA3
lui a1,LOCA4
sw a1,0(a0)
lui   a0,LOCA4
lui a1,LOCA5
sw a1,0(a0)
lui   a0,LOCA5
lui a1,LOCAFINAL
sw a1,0(a0)

lui   a0,LOCA2
addi a0,a0,4
lui a1,CONSTANT1
sw a1,0(a0)
lui   a0,LOCA3
addi a0,a0,4
```

```
lui a1,CONSTANT2
sw a1,0(a0)
lui   a0,LOCA4
addi a0,a0,4
lui a1,CONSTANT3
sw a1,0(a0)
lui   a0,LOCA5
addi a0,a0,4
lui a1,CONSTANT4
sw a1,0(a0)
```

在最后将内存的值分别存入s1,s2,s3,s4,代码如下：

```
lui   s0,LOCA2
addi s0,s0,4
lw s1,0(s0)
lui   s2,LOCA3
addi s2,s2,4
lw s2,0(s2)
lui   s3,LOCA4
addi s3,s3,4
lw s3,0(s3)
lui   s4,LOCA5
addi s4,s4,4
lw s4,0(s4)
```

结果：



可以看到，按顺序排列，与预期相同。

【测试用例2】有负数存在的情况

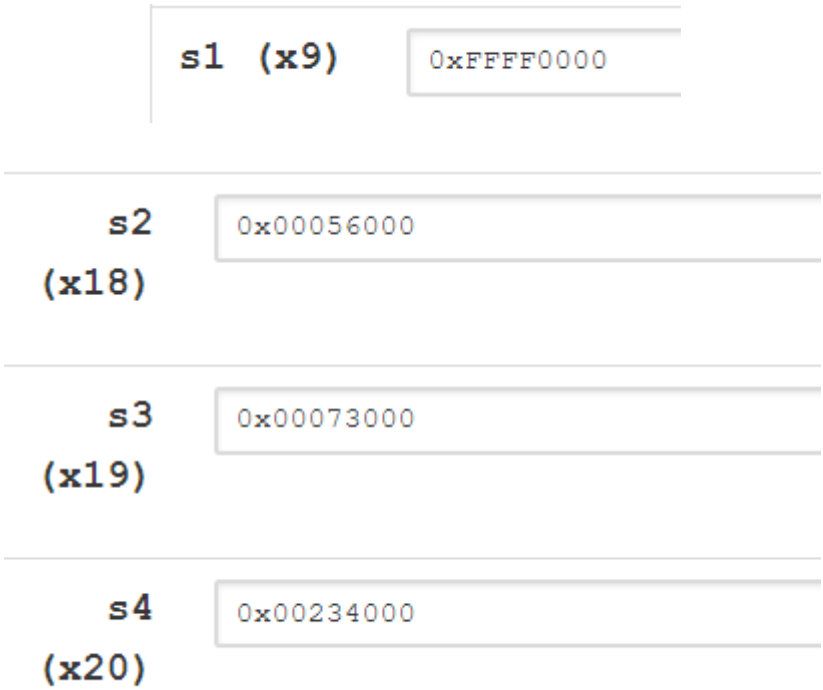
初始化数字及代码如下：

```
.equ CONSTANT1,0x00000056
.equ CONSTANT2,0xffff0
.equ CONSTANT3 0x00000234
.equ CONSTANT4 0x00000073
.equ LOCA1,0x3100
.equ LOCA2,0x3500
.equ LOCA3,0x3200
.equ LOCA4,0x3126
.equ LOCA5,0x3108
.equ LOCAFINAL,0x5000
lui a0,LOCA1
lui a1,LOCA2
sw a1,0(a0)
lui   a0,LOCA2
lui a1,LOCA3
sw a1,0(a0)
lui   a0,LOCA3
lui a1,LOCA4
sw a1,0(a0)
lui   a0,LOCA4
lui a1,LOCA5
sw a1,0(a0)
lui   a0,LOCA5
lui a1,LOCAFINAL
sw a1,0(a0)

lui   a0,LOCA2
addi a0,a0,4
lui a1,CONSTANT1
sw a1,0(a0)
lui   a0,LOCA3
addi a0,a0,4
lui a1,CONSTANT2
sw a1,0(a0)
lui   a0,LOCA4
addi a0,a0,4
lui a1,CONSTANT3
sw a1,0(a0)
lui   a0,LOCA5
addi a0,a0,4
lui a1,CONSTANT4
sw a1,0(a0)
```

最后结果储存代码：

```
lui   s0,LOCA2
addi s0,s0,4
lw s1,0(s0)
lui   s2,LOCA3
addi s2,s2,4
lw s2,0(s2)
lui   s3,LOCA4
addi s3,s3,4
lw s3,0(s3)
lui   s4,LOCA5
addi s4,s4,4
lw s4,0(s4)
```

测试结果：



与预期结果相同。

源代码

```
.equ CONSTANT1,0x00000056
.equ CONSTANT2,0xffff0
.equ CONSTANT3 0x00000234
.equ CONSTANT4 0x00000073
.equ LOCA1,0x3100
.equ LOCA2,0x3500
.equ LOCA3,0x3200
.equ LOCA4,0x3126
.equ LOCA5,0x3108
.equ LOCAFINAL,0x5000
lui a0,LOCA1
lui a1,LOCA2
```

```
sw a1,0(a0)
lui   a0,LOCA2
lui a1,LOCA3
sw a1,0(a0)
lui   a0,LOCA3
lui a1,LOCA4
sw a1,0(a0)
lui   a0,LOCA4
lui a1,LOCA5
sw a1,0(a0)
lui   a0,LOCA5
lui a1,LOCAFINAL
sw a1,0(a0)

lui   a0,LOCA2
addi a0,a0,4
lui a1,CONSTANT1
sw a1,0(a0)
lui   a0,LOCA3
addi a0,a0,4
lui a1,CONSTANT2
sw a1,0(a0)
lui   a0,LOCA4
addi a0,a0,4
lui a1,CONSTANT3
sw a1,0(a0)
lui   a0,LOCA5
addi a0,a0,4
lui a1,CONSTANT4
sw a1,0(a0)


lui   a0,LOCA1
lw t0,0(a0)
lui   a0,LOCA2
lw t1,0(a0)
lui   a0,LOCA3
lw t2,0(a0)

andi a0,a0,0
andi a1,a1,0
andi a2,a2,0
andi a3,a3,0
andi a4,a4,0
andi a5,a5,0
andi a6,a6,0
andi a7,a7,0
lui  a0, LOCA1 #a0储存起始点
lui  a1,LOCAFINAL #a1储存LOCA2的值

lw a0,0(a0)
```

```
JUDGE1: andi a5,a5,0
lw a5,0(a0)
not a2,a5
addi a2,a2,1
add a2,a2,a1
beqz a2,THEEND1

BUBBLE: addi a6,a0,4
lw a3,0(a6)
bltz a3,NEGA3
NEGABACK1:
lw a6,0(a0)
addi a6,a6,4
lw a4,0(a6)
bltz a4,SWAP
NEGABACK2:
not a5,a3
addi a5,a5,1
add a5,a5,a4
bltz a5,SWAP
jal NEXT

NEGA3:
lw a6,0(a0)
addi a6,a6,4
lw a4,0(a6)
bltz a4,NEGA4
jal NEXT

NEGA4:
not a4,a4
addi a4,a4,1
not a3,a3
addi a3,a3,1
not a5,a3
addi a5,a5,1
add a5,a5,a4
bgtz a5,SWAPNEGA
jal NEXT

SWAPNEGA:
not a4,a4
addi a4,a4,1
not a3,a3
addi a3,a3,1
jal SWAP


SWAP:
addi a7,a0,4
sw a4,0(a7)
lw a7,0(a0)
addi a7,a7,4
```

```
sw a3,0(a7)
jal NEXT

NEXT:
lw a0,0(a0)
jal JUDGE1

THEEND1: addi a1,a0,0
lui  a0, LOCA1
lw a0,0(a0)
jal JUDGE2

JUDGE2: andi a5,a5,0
andi a2,a2,0
lui      a5, LOCA1
lw a5,0(a5)
not a2,a1
addi a2,a2,1
add a2,a5,a2
beqz a2,THEEND2
jal JUDGE1
#查看结果
THEEND2:
lui   s0,LOCA2
addi s0,s0,4
lw s1,0(s0)
lui   s2,LOCA3
addi s2,s2,4
lw s2,0(s2)
lui   s3,LOCA4
addi s3,s3,4
lw s3,0(s3)
lui   s4,LOCA5
addi s4,s4,4
lw s4,0(s4)
andi ra,ra,0
addi ra,ra,-100
ret
```