PB18000028 邱子悦 lab01

一、实验题目：Grade Sorting with Arrays

二、实验思路：

考虑到总共只有 101 种成绩，却有 60 人，桶排序必然比冒泡排序节约指令。

1.利用.STRINGZ 初始化标识符 BZ 后连续的 101 个地址，为#48，初始化具体为多少不重要，**只需是正数**。PS：**.STRINGZ 初始化尾端为 0**，很方便检验。

2.CHECK：遍历 x3200 开始存的 60 个数，若值为 x，在（BZ+x）处**存入-1**。

3.SAVE：遍历 BZ 后 101 个地址，检验到负数（-1）的，将相对 BZ 的距离**倒着存入 x403B**；检验到正数就跳过；检验到 0 结束。

4.COUNT（A/B/BB/C/CC/D）：从 x4000 开始遍历，利用.FILL，**事先存好**了

```
N85     .FILL #-85
N75     .FILL #-75
N60     .FILL #-60
```

，由于**本题特殊性**，30%、50%等条件可以不需要考虑（85 及以上的成绩只能有 16 个，必然都在前 30%，即前 18 名；75 及以上的成绩只能有 26 个，必然都在前 50%，即前 30 名）。

将结果保存在 x4100-x4103。

三、运行结果

【测试数据 1：随机数】

用 python 程序生成的：

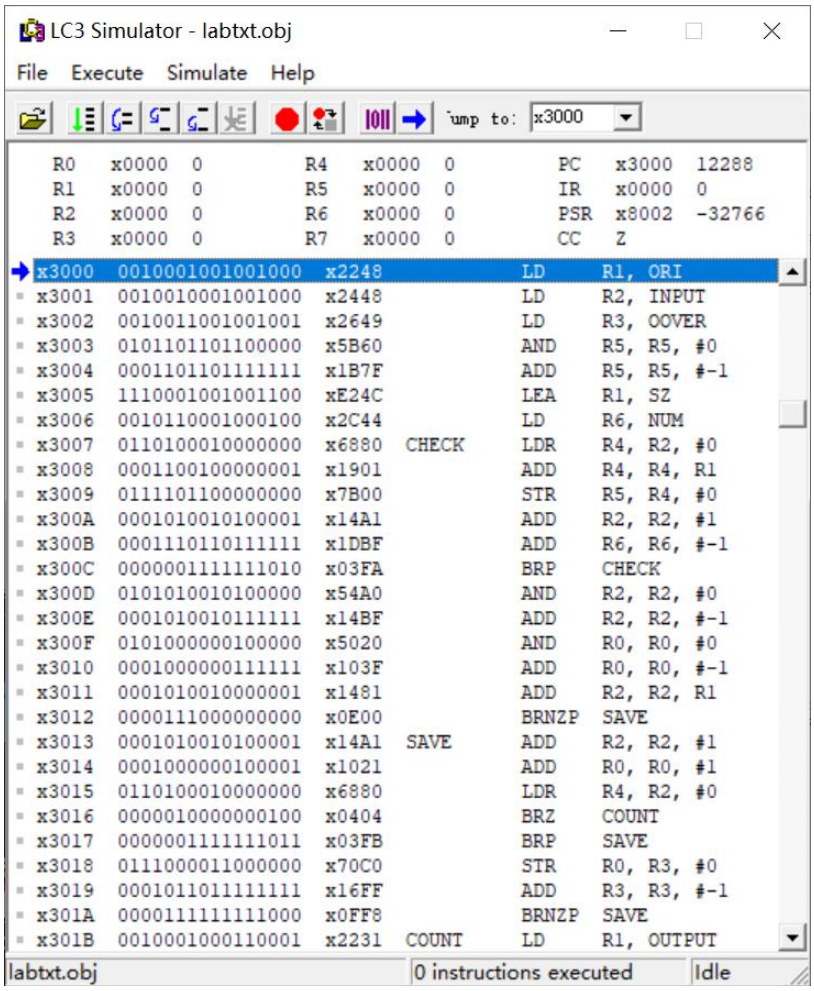
```
import random
l=[]
while(len(l)<60):
    x=int(100*random.random())
    for k in l:
        if(k==x):
            break
    else:
        l.append(x)

for x in l:
    y=hex(x)
    y=y[2:]
    y=y.zfill(4)
    print(y)

s=set(l)
for z in s:
    z=hex(z)
    z=z[2:]
    z=z.zfill(4)
    print(z)
```

写成的 num 文件内容为：

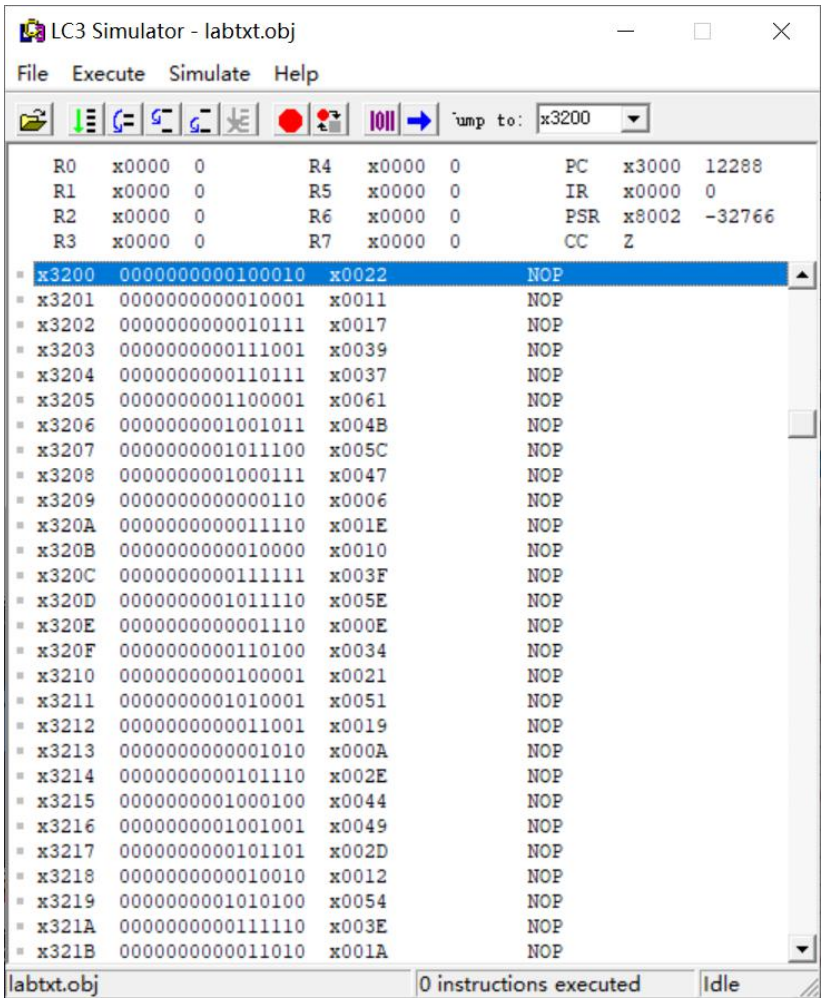
3200
0022
0011
0017
0039
0037
0061
004b
005c
0047
0006
001e
0010
003f
005e
000e
0034
0021
0051
0019
000a
002e
0044
0049
002d
0012
0054
003e

001a
0003
004c
0000
0030
002b
001f
0040
0020
0005
0018
0027
0033
0041
0062
004d
0029
0036
002f
0002
005b
0014
0007
001d
0032
0026
000d
0038
0043
0015
0025
0045
0028
与源代码一起执行的情况如下：
执行前

x3000 处存了指令：

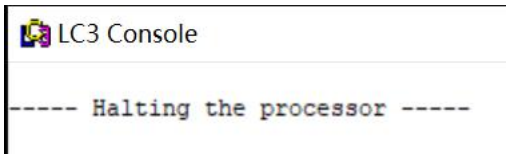

LC3 Simulator - labtxt.obj
File Execute Simulate Help
Jump to: x3000
R0 x0000 0 R4 x0000 0 PC x3000 12288
R1 x0000 0 R5 x0000 0 IR x0000 0
R2 x0000 0 R6 x0000 0 PSR x8002 -32766
R3 x0000 0 R7 x0000 0 CC Z
x3000 0010001001001000 x2248 LD R1, ORI
x3001 0010010001001000 x2448 LD R2, INPUT
x3002 0010011001001001 x2649 LD R3, OOVER
x3003 0101101101100000 x5B60 AND R5, R5, #0
x3004 0001101101111111 x1B7F ADD R5, R5, #-1
x3005 1110001001001100 xE24C LEA R1, SZ
x3006 0010110001000100 x2C44 LD R6, NUM
x3007 0110100010000000 x6880 CHECK LDR R4, R2, #0
x3008 0001100100000001 x1901 ADD R4, R4, R1
x3009 0111101100000000 x7B00 STR R5, R4, #0
x300A 0001010010100001 x14A1 ADD R2, R2, #1
x300B 0001110110111111 x1DBF ADD R6, R6, #-1
x300C 0000001111111010 x03FA BRP CHECK
x300D 0101010010100000 x54A0 AND R2, R2, #0
x300E 0001010010111111 x14BF ADD R2, R2, #-1
x300F 0101000000100000 x5020 AND R0, R0, #0
x3010 0001000000111111 x103F ADD R0, R0, #-1
x3011 0001010010000001 x1481 ADD R2, R2, R1
x3012 0000111000000000 x0E00 BRNZP SAVE
x3013 0001010010100001 x14A1 SAVE ADD R2, R2, #1
x3014 0001000000100001 x1021 ADD R0, R0, #1
x3015 0110100010000000 x6880 LDR R4, R2, #0
x3016 0000010000000100 x0404 BRZ COUNT
x3017 0000001111111011 x03FB BRP SAVE
x3018 0111000011000000 x70C0 STR R0, R3, #0
x3019 0001011011111111 x16FF ADD R3, R3, #-1
x301A 0000111111111000 x0FF8 BRNZP SAVE
x301B 0010001000110001 x2231 COUNT LD R1, OUTPUT
labtxt.obj
0 instructions executed
Idle

X3200 处存了数据

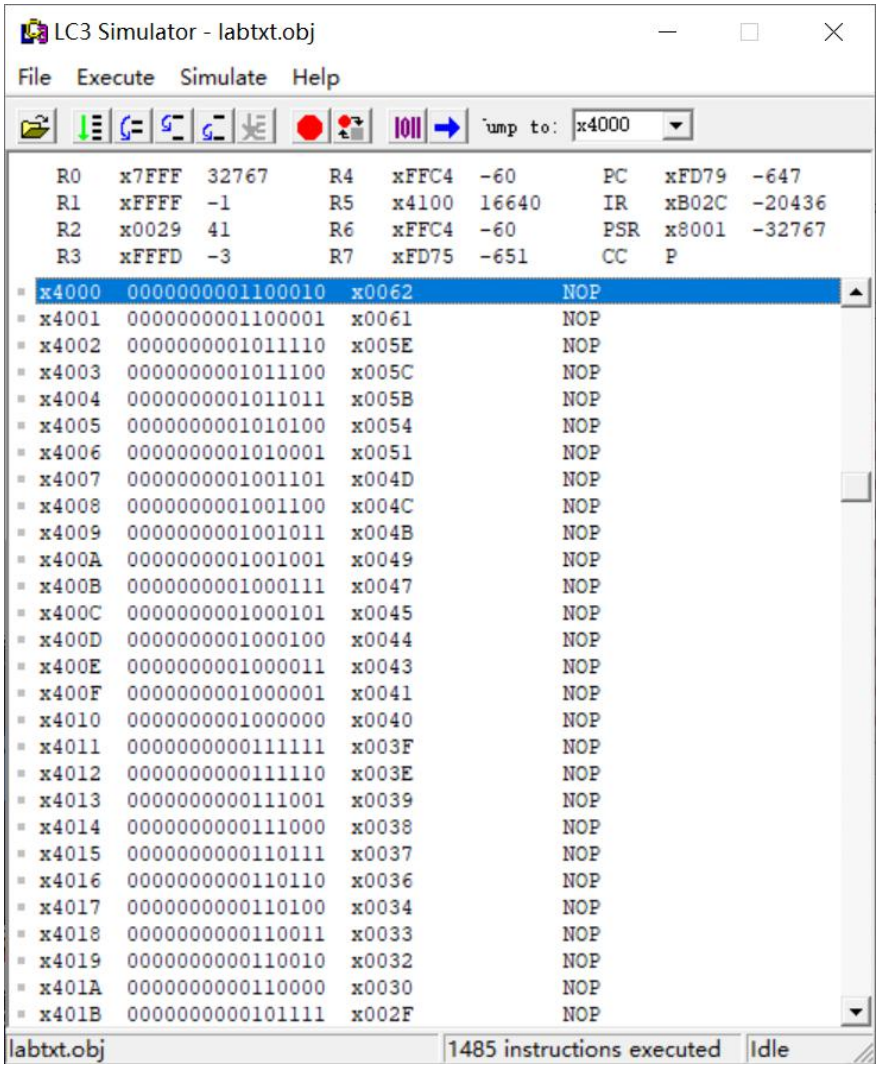


运行后：1485instructions executed

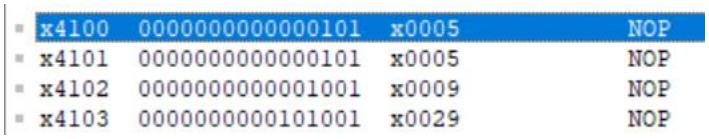
HALT 正常



X4000 开始有降序排列的数字，与 python 排序后的结果完全一致。



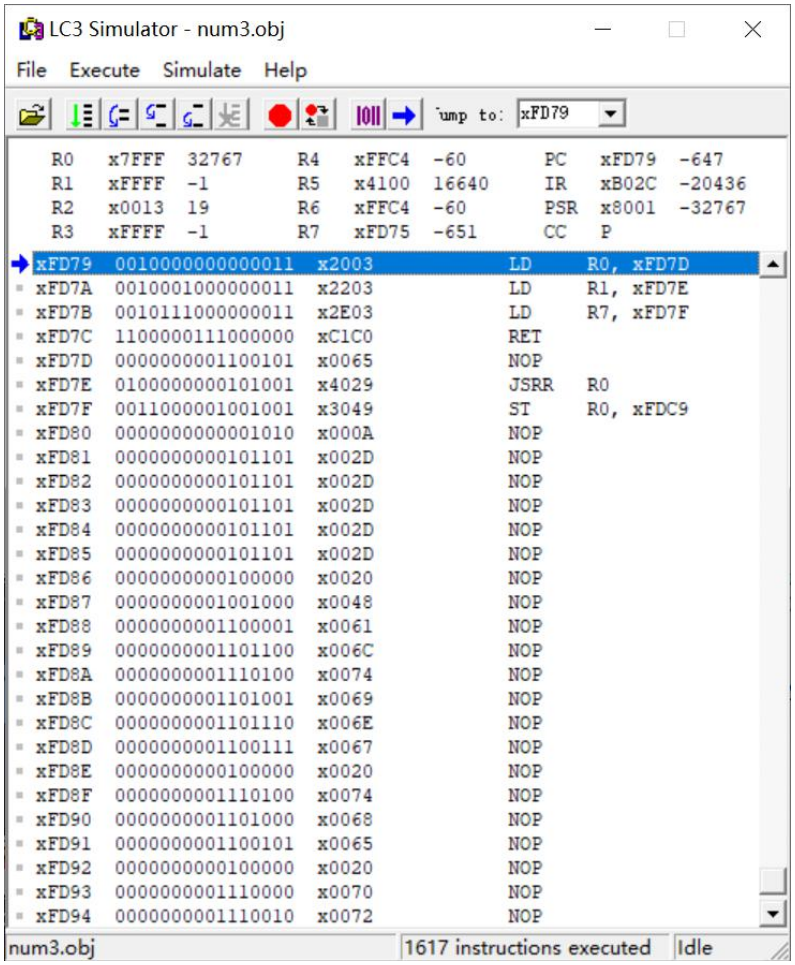
x4100-x4103 处



即 A 有 5 个，B 有 5 个，C 有 9 个，D 有 41 个，符合数据情况。

【测试数据 2：严格升序的数列 41~100】

执行后指令数为1617 instructions executed，说明本程序采用桶排序相比冒泡排序的综合效率较高，基本不受极端测试数据影响。



四、源代码

```
.ORIG x3000
LD R2,INPUT     ;x3200 as a pointer
LD R3,OOVER
AND R5,R5,#0    ;
ADD R5,R5,#-1 ;R5<-#-1 as a set
LEA R1,SZ       ;R1<-address of SZ
LD R6,NUM

CHECK     LDR R4,R2,#0
      ADD R4,R4,R1
            STR R5,R4,#0    ;set -1
            ADD R2,R2,#1    ;pointer++
            ADD R6,R6,#-1 ;counter--
            BRp CHECK
            AND R2,R2,#0    ;?
            ADD R2,R2,#-1
      AND R0,R0,#0
      ADD R0,R0,#-1
      ADD R2,R2,R1 ;point to SZ-1
```

```
            BR SAVE

SAVE       ADD R2,R2,#1 ;R2 as a pointer
      ADD R0,R0,#1 ;R0~number
            LDR R4,R2,#0
            BRz COUNT
            BRp SAVE
            ;BRn
            STR R0,R3,#0 ;number is stored in output backwards
            ADD R3,R3,#-1
            BR SAVE
;85-100 A
;84-75   B
;60-74   C
;0-59    D

COUNT     LD R1,OUTPUT ;pointer
      AND R2,R2,#0 ;R2 as a counter
            LD R4,N85
            BR A

A      LDR R3,R1,#0
       ADD R3,R3,R4
       BRn B
       ADD R2,R2,#1
       ADD R1,R1,#1
       BR A

B      LD R5,SCORES
       STR R2,R5,#0
       AND R2,R2,#0
       ADD R5,R5,#1
       LD R4,N75
       BR BB

BB    LDR R3,R1,#0
       ADD R3,R3,R4
       BRn C
       ADD R2,R2,#1
       ADD R1,R1,#1
       BR BB

C      STR R2,R5,#0
       AND R2,R2,#0
```

```
        ADD R5,R5,#1
        LD R4,N60
        BR CC

CC    LDR R3,R1,#0
        ADD R3,R3,R4
        BRn D
        ADD R2,R2,#1
        ADD R1,R1,#1
        BR CC

D      STR R2,R5,#0
        ADD R5,R5,#-1
        LDR R6,R5,#0
        ADD R2,R2,R6
        ADD R5,R5,#-1
        LDR R6,R5,#0
        ADD R2,R2,R6
        LD R6,N60
        ADD R2,R2,R6
        NOT R2,R2
        ADD R2,R2,#1
        STR R2,R5,#3
        HALT

ORI         .FILL x3000
INPUT      .FILL x3200
NUM          .FILL #60
OOVER      .FILL x403B
OUTPUT    .FILL x4000
N85          .FILL #-85
N75          .FILL #-75
N60          .FILL #-60
SCORES    .FILL x4100
SZ                                                                    .STRINGZ
      "0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000"

.END
```