ICS-lab4 实验报告

孔浩宇 PB20000113

2022 年 12 月 27 日

目录

1 实验目的

对于存储于 x4000 至 x400F 的 16 名学生的成绩 $(0-100)$，将其升序排列并依次存储于 x5000 至 x500F，并统计获得 A，B 成绩的学生数量，将结果依次存储在 x5100 与 x5101 中。

2 实验原理

2.1 实现 if 语句

(1) if A==B.

```
        ADD     R, A, #0
        NOT     R, R
        ADD     R, R, #1    ; R=-A
        ADD     R, R, B     ; R=B-A
        BRnp     AFTER      ;
        ...                 ; if(B-A==0) do this
    AFTER
```

(2) if A>B.

```
        ADD     R, A, #0
        NOT     R, R
        ADD     R, R, #1    ; R=-A
        ADD     R, R, B     ; R=B-A
        BRzp     AFTER      ;
        ...                 ; if(B-A<0) do this
AFTER
```

(3) if A≥B.

```
        ADD     R, A, #0
        NOT     R, R
        ADD     R, R, #1    ; R=-A
        ADD     R, R, B     ; R=B-A
        BRp     AFTER      ;
        ...                 ; if(B-A<=0) do this
AFTER
```

(4) 其余 A<B, A≤B 类似，不再赘述.

2.2 数据交换

将 R0 (起始) 与 R0 + 1 内存中的数据交换，利用 R3 存储读取前一个数，R4 读取后一个数，再利用 STR 指令将 R3 的值存储到后一个地址单元，将 R4 的值存储到前一个地址单元。

```
LDR R3, R0, #0
ADD R0, R0, #1
LDR R4, R0, #0
STR R3, R0, #0
ADD R0, R0, #-1
STR R4, R0, #0
ADD R0, R0, #1
```

3 实验步骤

3.1 初始化

(0) 标号

```
DATA    .FILL X4000
COPY    .FILL X5000
COUNTER .FILL X0010
MAX     .FILL x500F
ASubs   .FILL X5100
BSubs   .FILL X5101
ALAST   .FILL X500C
BLAST   .FILL X5008
A           .FILL #85
B           .FILL #75
NUM         .FILL #15
```

(1) 初始化其他变量

```
LD R0, DATA
LD R1, COPY
LD R2, COUNTER
```

3.2 数据搬迁

用 R_0 来存储原数据首地址 x4000 (DATA)，用 R_1 来存储排序后数据存储的首地址 x5000 (COPY)，R_2 的值为循环次数 16，采用基址加偏移的寻址方式读取内存，R_3 作为中间搬运工，利用 LDR 和 STR 指令先将数据搬移。

```
LOOP1   BRZ NEXT1
LDR R3, R0, #0
ADD R0, R0, #1
STR R3, R1, #0
ADD R1, R1, #1
ADD R2, R2, #-1
BRNZP LOOP1
```

3.3 冒泡排序

现在数据已经搬移到 x4000 ∼ x400F，开始排序。

普通的冒泡排序需要内外两层循环，外循环 n-1 次，内循环 n-1 次，因此，我们让 R_0 和 R_1 的值都为 15，读取内存时依旧采用基址加偏移的寻址方式，R_2 存储数据存储地址 x5000，R_3 读取前一个数，R_4 读取后一个数，之后需要进行比较大小，将 R_3 取反加 1 存储在 R_5，然后将 R_5 和 R_4 相加的结果存储在 R_5，通过判断 R_5 的正负来判断 R_3 和 R_4 的大小，如果 R3 大于 R4，即 R5 是正数，那么就交换这两个数，利用 STR 指令将 R3 的值存储到后一个地址单元，将 R4 的值存储到前一个地址单元。

```
NEXT1       LD R1, NUM
LOOP2       BRZ NEXT2
        LD R0, COPY
        LD R2, NUM
LOOP3       BRZ AGAIN
        LDR R3, R0, #0
        ADD R0, R0, #1
        LDR R4, R0, #0
        NOT R5, R3
        ADD R5, R5, #1
        ADD R5, R5, R4
        BRp RIGHT
        STR R3, R0, #0
        ADD R0, R0, #-1
        STR R4, R0, #0
        ADD R0, R0, #1
RIGHT       ADD R2, R2, #-1
        BRNZP LOOP3
AGAIN       ADD R1, R1, #-1
        BRNZP LOOP2
```

3.4 成绩分类

读取数据采取基址加偏移的寻址方式，R0 存储最大成绩的地址 x500F，R1 和 R2 清 0 用来存储 A 和 B 的人数，R3 存储循环次数 16，R4 存储每一个数据，R5 作为中间载体，存储数据取反加一的值，R6 和 R7 存储着 A 和 B 的临界成绩 85 和 75，同时承担存储和 R5 相加后的数据，用于比较成绩。最后将 R1 和 R2 的值分别存进地址为 x5100 和 x5101 的内存单元。

```
NEXT2       LD  R0, MAX
    AND R1, R1, #0
    AND R2, R2, #0
    LD  R3, COUNTER
LOOP4       BRZ FINISH
        LDR R4, R0, #0
        LD  R6, A
        LD  R7, B
        NOT R4, R4
        ADD R4, R4, #1
        ADD R6, R6, R4
        BRP BB
        NOT R5, R0
        ADD R5, R5, #1
        LD  R6, ALAST
        ADD R6, R6, R5
        BRP BB
        ADD R1, R1, #1
        BRNZP TAIL
BB      ADD R7, R7, R4
        BRP TAIL
        LD  R7, BLAST
        ADD R7, R7,R5
        BRP TAIL
        ADD R2, R2, #1
TAIL        ADD R0, R0, #-1
        ADD R3, R3, #-1
        BRNZP LOOP4
```

3.5 结束

```
FINISH      STI R1, ASubs
        STI R2, BSubs
        HALT
```

3.6 代码

```
        .ORIG X3000
        LD R0, DATA
        LD R1, COPY
        LD R2, COUNTER
LOOP1   BRZ NEXT1
        LDR R3, R0, #0
        ADD R0, R0, #1
        STR R3, R1, #0
        ADD R1, R1, #1
        ADD R2, R2, #-1
        BRNZP LOOP1
NEXT1   LD R1, NUM
LOOP2   BRZ NEXT2
        LD R0, COPY
        LD R2, NUM
LOOP3   BRZ AGAIN
        LDR R3, R0, #0
        ADD R0, R0, #1
        LDR R4, R0, #0
        NOT R5, R3
        ADD R5, R5, #1
        ADD R5, R5, R4
        BRp RIGHT
        STR R3, R0, #0
        ADD R0, R0, #-1
        STR R4, R0, #0
        ADD R0, R0, #1
RIGHT   ADD R2, R2, #-1
            BRNZP LOOP3
AGAIN   ADD R1, R1, #-1
            BRNZP LOOP2
NEXT2   LD  R0, MAX
        AND R1, R1, #0
        AND R2, R2, #0
        LD  R3, COUNTER
LOOP4   BRZ FINISH
        LDR R4, R0, #0
        LD  R6, A
        LD  R7, B
        NOT R4, R4
        ADD R4, R4, #1
```

```
        ADD R6, R6, R4
        BRP BB
        NOT R5, R0
        ADD R5, R5, #1
        LD  R6, ALAST
        ADD R6, R6, R5
        BRP BB
        ADD R1, R1, #1
        BRNZP TAIL
BB          ADD R7, R7, R4
        BRP TAIL
        LD  R7, BLAST
        ADD R7, R7,R5
        BRP TAIL
        ADD R2, R2, #1
TAIL    ADD R0, R0, #-1
        ADD R3, R3, #-1
        BRNZP LOOP4

FINISH  STI R1, ASubs
        STI R2, BSubs
        HALT
DATA    .FILL X4000
COPY    .FILL X5000
COUNTER .FILL X0010
MAX     .FILL x500F
ASubs   .FILL X5100
BSubs   .FILL X5101
ALAST   .FILL X500C
BLAST   .FILL X5008
A           .FILL #85
B           .FILL #75
NUM         .FILL #15
        .END
```

4 实验结果

选择评测实验

○ lab1 ○ lab2 ○ lab3 ◉ lab4 ○ 自定义

测试样例，样例之间以逗号分割

100:95:90:85:80:60:55:50:45:40:35:30:25:20:10:0,95:100:0:50:45:40:80:65:70:75:35:20:25:15:10:90,88:77

代码文本

```
COUNTER .FILL X0010
```

调试模式

评测

汇编评测

3 / 3 个通过测试用例

- 平均指令数: 3064.6666666666665
- 通过 100:95:90:85:80:60:55:50:45:40:35:30:25:20:10:0, 指令数: 3177, 输出: 0,10,20,25,30,35,40,45,50,55,60,80,85,90,95,100,4,1
- 通过 95:100:0:50:45:40:80:65:70:75:35:20:25:15:10:90, 指令数: 3008, 输出: 0,10,15,20,25,35,40,45,50,65,70,75,80,90,95,100,3,2
- 通过 88:77:66:55:99:33:44:22:11:10:9:98:97:53:57:21, 指令数: 3009, 输出: 9,10,11,21,22,33,44,53,55,57,66,77,88,97,98,99,4,1