Lab04 Sort and Count

Category

```
Lab04 Sort and Count
Category
Task & Purpose
Example
Input:
Output:
Principle
Procedure
Code in C++
Code
Result
```

Task & Purpose

Now that there are 16 students' scores.

If a student scores 85 or above and is in top 25%, he/she will receive an A.

If he/she does not get an A but scores 75 or above and is in top 50%, he/she will get a B.

Note:

- Each score is stored in successive memory locations starting with address x4000. Each score is an integer between 0 and 100. ($0 \le score \le 100$).
- Everyone gets a different score.

The job:

- The program should sort scores in ascending order (smallest-to-largest) and store them in successive memory locations starting with address x5000.
- The program should count how many students get an A and store the number in x5010.
- The program should count how many students fet a B and store the number in x5101.

The program should start at x3000

Example

Input:

Memory address	example1	example2	example3
x4000	100	95	88
x4001	95	100	77
x4002	90	0	66
x4003	85	50	55
x4004	80	45	99
x4005	60	40	33
x4006	55	80	44
x4007	50	65	22
x4008	45	70	11
x4009	40	75	10
x400A	35	35	9
x400B	30	20	98
X400C	25	25	97
X400D	20	15	53
X400E	10	10	57
X400F	0	90	21

Output:

Memory address	example1	example2	example3
x5000	0	0	9
x5001	10	10	10
x5002	20	15	11
x5003	25	20	21
x5004	30	25	22
x5005	35	35	33
x5006	40	40	44
x5007	45	45	53
x5008	50	50	55
x5009	55	65	57
x500A	60	70	66
x500B	80	75	77
X500C	85	80	88
X500D	90	90	97
X500E	95	95	98
X500F	100	100	99

Memory address	example1	example2	example3
x5100	4	3	4
x5101	1	2	1

Principle

The program can be divided into 3 parts:

- 1. Copy the array from its original address to the targeted address;
- 2. Sort the array by bubble sorting;
- 3. Count the number of students who should be given a A and ones with B.

Procedure

Firstly, I chose to use select sorting instead of bubble sorting. However, I find it really hard considering that select sorting need to determine the index of the minimal or maximal number, which is not handy to achieve owing to the extremely limited registers and instructions. On the other hand, with bubble sorting, in each iteration, it only requires to locate two adjoining numbers.

Apart from that, it relatively easy to complete.

Code in C++

In this specific problem, I found it massively helpful to right the program first with my most skilled programming language - C++, and I evenly found it much more useful than draw a flowchart which requires me to use another software and to spend much more time.

```
1 | void SortAndCount(int * in, int * & out, int &countA, int
   &countB)
 2
   {
 3
        int i, j, temp;
 4
        countA = 0;
 5
        countB = 0;
        i = 15;
 6
 7
        while(i \ge 0)
8
        {
            out[i] = in[i];
9
10
            i--;
11
        for(i = 16; i > 0; i--)
12
13
            for(j = 0; j < i; j++)
14
            {
15
                 countA++;
16
                 if(out[j] > out[j+1])
17
                 {
                     temp = out[j];
18
                     out[j] = out[j+1];
19
20
                     out[j+1] = temp;
21
                     countB++;
22
                 }
            }
23
```

```
24
        for(i = 12; i < 16; i++)
25
            if(out[i] \ge 85)
26
                countA++;
27
            else if(out[i] \geq 75)
28
                countB++;
        for(i = 8; i < 12; i++)
29
30
            if(out[i] \ge 75)
31
                countB++;
32 }
```

Code

```
.ORIG X3000
 2
            AND
                    RO, RO, #0
 3
                    R1, R1, #0
            AND
 4
            AND
                    R2, R2, #0
 5
                    R3, R3, #0
           AND
                    R4, R4, #0
 6
           AND
 7
                    R5, R5, #0
           AND
                    R6, R6, #0
                                    ;STACKPOINTER
 8
           AND
                    R7, R7, #0
 9
           AND
                                   ;RETURN LINKAGE
                    R6, STACKX6000
10
            LD
                                       ;STACKPOINTER
11
                    RO, STOREX4000
                                        ;WHERE THE 16 SCORES ARE
            LD
   STORED
12
            LD
                    R1, STOREX5000
                                       ; WHERE THE SORTED SCORES
   ARE STORED
13
                    COPY
            JSR
14
            JSR
                    SORT
15
            JSR
                    COUNT
16
           HALT
17
18
            ;LABEL: COPY
19
            ;FUNCTION: COPY THE INPUT ARRAY TO THE OUTPUT ARRAY
            ;PARAMETER: I(R2), (I - 16)(R3), (-16)(R4), VALUE OF
20
   R0(R5)
            ;INPUT: R0 = INPUT ARRAY
21
22
            ;OUTPUT: R1 = OUTPUT ARRAY
23
            COPY
                    ADD
                            R6, R6, #-1
                            R7, R6, #0 ;SAVE RETURN LINKAGE
24
                    STR
```

```
25
                            R6, R6, #-1
                    ADD
26
                    STR
                            R2, R6, #0 ;SAVE R2, WHICH WILL
   BE USEED AS I
27
                            R6, R6, #-1
                    ADD
                                           ;SAVE R3, WHICH WILL
28
                            R3, R6, #0
                    STR
   BE USED AS (I - 16)
29
                            R6, R6, #-1
                    ADD
                            R4, R6, #0
30
                                            ;SAVE R4, WHICH WILL
                    STR
   BE USED AS 16
31
                            R6, R6, #-1
                    ADD
32
                    STR
                            R5, R6, #0
                                           ;SAVE R5, WHICH WILL
   BE USED AS VALUE OF RO
33
                    AND
                            R2, R2, #0
                                           :I = 0
34
                            R4, STORE16
                                            ; R4 = 16
                    LD
35
                    NOT
                            R4, R4
                            R4, R4, #1
36
                    ADD
                                           ;R4 = -16
37
           L00P1
                    ADD
                            R3, R2, R4
                                            ;R3 = (I - 16)
                                           ;IF R3 < 0, END LOOP
38
                    BRZP
                            ENDCOPY
                            R5, R0, #0
39
                    LDR
                                            ;R5 = VALUE OF R0
40
                            R5, R1, #0
                                            ;STORE R5 IN R1
                    STR
                            R2, R2, #1
41
                    ADD
                                            ; I = I + 1
42
                            R0, R0, #1
                                            ;R0 = R0 + 1
                    ADD
43
                    ADD
                            R1, R1, #1
                                           ;R1 = R1 + 1
44
                    BRNZP
                            L00P1
45
           ENDCOPY LDR
                            R5, R6, #0
46
                                           ;RESTORE R5
47
                            R6, R6, #1
                    ADD
48
                    LDR
                            R4, R6, #0
                                            ; RESTORE R4
49
                            R6, R6, #1
                    ADD
50
                    LDR
                            R3, R6, #0
                                            ; RESTORE R3
                            R6, R6, #1
51
                    ADD
52
                            R2, R6, #0
                    LDR
                                            ; RESTORE R2
53
                            R6, R6, #1
                    ADD
54
                            R7, R6, #0
                    LDR
                                         ; RESTORE RETURN
   LINKAGE
                            R6, R6, #1
55
                    ADD
56
                    RET
57
58
            ;LABEL: SORT
            ; FUNCTION: SORT THE ARRAY
59
            ; PARAMETER: (R1), (R2), (R3), R4, R5
60
```

```
61
           ; INPUT:
62
           ;OUTPUT:Y
63
           SORT
                   ADD
                          R6, R6, #-1
                          R7, R6, #0
64
                   STR
                                          ;SAVE RETURN LINKAGE
                          R6, R6, #-1
65
                   ADD
66
                   STR
                          R1, R6, #0
                                          ;SAVE R1
                          R6, R6, #-1
67
                   ADD
68
                   STR
                          R2, R6, #0
                                          ;SAVE R2
69
                   ADD
                          R6, R6, #-1
                                          ;SAVE R3, WHICH WILL
70
                          R3, R6, #0
                   STR
   BE USED AS POINTER TO THE ARRAY
71
                   ADD
                          R6, R6, #-1
72
                   STR
                          R4, R6, #0
                                          ;SAVE R4, WHICH WILL
   BE USEED AS I
73
                          R6, R6, #-1
                   ADD
74
                   STR
                          R5, R6, #0
                                         ;SAVE R5, WHICH WILL
   BE USED AS J
75
                   ;-----
76
                   LD R4, STORE16
77
                   OUTERLOOP
                                         R4, R4, #-1; loop n
                                  ADD
   - 1 times
78
                                          SORTED
                                  BRNZ
   Looping complete, exit
79
                                          R5, R4, #0;
                                  ADD
   Initialize inner loop counter to outer
80
                                  LD
                                          R3, STOREX5000
   Set file pointer to beginning of ARRAY
                                          R0, R3, #0 ; Get
81
                   INNERLOOP
                                  LDR
   item at ARRAY pointer
82
                                          R1, R3, #1 ; Get
                                  LDR
   next item
83
                                          R2, R1 ; Negate
                                  NOT
84
                                  ADD
                                          R2, R2, #1;
   ... next item
85
                                  ADD
                                          R2, R0, R2; swap =
   item - next item
                                          SWAP
86
                                  BRNZ
                                                     ; Don't
   swap if in order (item ≤ next item)
87
                                          R1, R3, #0;
                                  STR
   Perform ...
```

```
88
                                      STR R0, R3, #1;
       ... swap
                                              R3, R3, #1;
 89
                     SWAP
                                      ADD
    Increment file pointer
                                              R5, R5, #-1;
 90
                                      ADD
    Decrement inner loop counter
                                              INNERLOOP ; End of
 91
                                      BRP
    inner loop
 92
                                      BRNZP
                                              OUTERLOOP ; End of
    outer loop
 93
 94
                              R5, R6, #0
             SORTED LDR
                                             ;RESTORE R5
 95
                              R6, R6, #1
                     ADD
                              R4, R6, #0
96
                     LDR
                                              ; RESTORE R4
97
                              R6, R6, #1
                     ADD
                              R3, R6, #0
98
                     LDR
                                              ; RESTORE R3
99
                              R6, R6, #1
                     ADD
100
                              R2, R6, #0
                                              ; RESTORE R2
                     LDR
                              R6, R6, #1
101
                     ADD
                              R1, R6, #0
102
                     LDR
                                              ; RESTORE R1
103
                     ADD
                              R6, R6, #1
                              R7, R6, #0
104
                                             ; RESTORE RETURN
                     LDR
    LINKAGE
                             R6, R6, #1
105
                     ADD
106
                     RET
107
108
             ;LABEL: COUNTA
             COUNT
                              R6, R6, #-1
109
                     ADD
110
                     STR
                              R7, R6, #0
                                              ;SAVE RETURN LINKAGE
                              R6, R6, #-1
111
                     ADD
                              R0, R6, #0
112
                     STR
                                               ;SAVE RO
113
                     ADD
                              R6, R6, #-1
114
                     STR
                              R1, R6, #0
                                               ;SAVE R1
115
                     ADD
                              R6, R6, #-1
                              R2, R6, #0
116
                     STR
                                               ;SAVE R2
117
                              R6, R6, #-1
                     ADD
                              R3, R6, #0
118
                     STR
                                               ;SAVE R3
119
                              R6, R6, #-1
                     ADD
120
                              R4, R6, #0
                                               ;SAVE R4
                     STR
121
                              R6, R6, #-1
                     ADD
```

```
R5, R6, #0 ;SAVE R5
122
                   STR
                           RO, STOREX500F ; ARRAY POINTER
123
                   LD
124
                           R1, R0, #0
                   LDR
                                         ;R1 = A[15]
125
                   AND
                           R2, R2, #0
                                          ;R2 = 0, USED AS
    LOOP COUNTER
                           R2, R2, #4 ;COUNTER = 4
126
                   ADD
                           R4, R4, #0
127
                                         ;R4 IS COUNTA
                   AND
                           R5, R5, #0 ; R5 IS COUNTB
128
                   AND
                                  R2, R2, #-1;
129
                   LOOPA
                           ADD
130
                           BRN
                                  OUTLOOPA
                                         R6, R6, #-1
131
                           IFA
                                  ADD
                                         R2, R6, #0
132
                                  STR
133
                                  LD
                                         R3, STOREN85
                                         R2, R1, R3
134
                                  ADD
                                                         ; R2
    = R1 - 85
135
                                  BRN
                                         ELIFA
                                         R4, R4, #1
136
                                  ADD
    ;COUNTA++
137
                                  BRNZP ENDIFA
                                         R3, STOREN75
138
                           ELIFA
                                  LD
139
                                  ADD
                                         R2, R1, R3
                                                         ;R2
    = R1 - 75
                                  BRN ENDIFA
140
                                         R5, R5, #1
141
                                  ADD
                                        R2, R6, #0
142
                           ENDIFA LDR
                                        R6, R6, #1
143
                                  STR
144
                                  R0, R0, #-1
                           ADD
                                  R1, R0, #0
145
                           LDR
                           BRNZP LOOPA
146
147
                   OUTLOOPA
                   AND R2, R2, \#0; R2 = 0, USED AS
148
    LOOP COUNTER
                           R2, R2, #4 ; COUNTER = 4
149
                   ADD
150
                   L00PB
                           ADD
                                  R2, R2, #-1;
151
                           BRN
                                  OUTLOOPB
                                         R6, R6, #-1
152
                           IFB
                                  ADD
153
                                  STR
                                         R2, R6, #0
                                         R3, STOREN75
154
                                  LD
155
                                         R2, R1, R3
                                  ADD
                                                         ;R2
    = R1 - 75
156
                                  BRN
                                         ENDIFB
157
                                  ADD R5, R5, #1
    ; COUNTA++
```

```
158
                            ENDIFB LDR R2, R6, #0
                                   STR R6, R6, #1
159
                                   R0, R0, #-1
160
                           ADD
                                   R1, R0, #0
161
                           LDR
162
                                   L00PB
                           BRNZP
163
                    OUTLOOPB
                           R4, STOREX5100
164
                    STI
165
                    STI
                           R5, STOREX5101
                           R5, R6, #0
166
                    LDR
                                      ;RESTORE R5
167
                           R6, R6, #1
                    ADD
                           R4, R6, #0
168
                    LDR
                                          ; RESTORE R4
169
                           R6, R6, #1
                    ADD
170
                    LDR
                           R3, R6, #0
                                           ; RESTORE R3
                           R6, R6, #1
171
                    ADD
172
                           R2, R6, #0
                                          ; RESTORE R2
                    LDR
                           R6, R6, #1
173
                    ADD
174
                           R1, R6, #0
                                           ; RESTORE R1
                    LDR
175
                           R6, R6, #1
                    ADD
                           RO, R6, #0 ; RESTORE RO
176
                    LDR
177
                    ADD
                           R6, R6, #1
178
                    RET
179
180
            STOREX4000 .FILL X4000
                       .FILL X5000
181
            STOREX5000
182
                       .FILL
            STOREX5100
                               X5100
            STOREX5101 .FILL X5101
183
            STACKX6000
                       .FILL X6000
184
                       .FILL X500F
185
            STOREX500F
186
            STORE16
                        .FILL X0010
                       .FILL XFFB5
187
            STOREN75
188
            STOREN85
                        .FILL XFFAB
189 .END
```

Result

The result is shown below:

汇编评测

3/3个通过测试用例

- 平均指令数: 1682.666666666667
- 通过 100:95:90:85:80:60:55:50:45:40:35:30:25:20:10:0, 指令数: 1738, 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: 1656, 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: 1654, 输出: 9,10,11,21,22,33,44,53,55,57,66,77,88,97,98,99,4,1