Lab04 Sort and Count

Category

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 ≤ score ≤ 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.

```cpp
void SortAndCount(int * in, int * & out, int &countA, int &countB)
{
    int i, j, temp;
    countA = 0;
    countB = 0;
    i = 15;
    while(i >= 0)
    {
        out[i] = in[i];
        i--;
    }
    for(i = 16; i > 0; i--)
        for(j = 0; j < i; j++)
        {
            countA++;
            if(out[j] > out[j+1])
            {
                temp = out[j];
                out[j] = out[j+1];
                out[j+1] = temp;
                countB++;
            }
        }
    for(i = 12; i < 16; i++)
        if(out[i] >= 85)
            countA++;
        else if(out[i] >= 75)
            countB++;
    for(i = 8; i < 12; i++)
        if(out[i] >= 75)
            countB++;
}
```

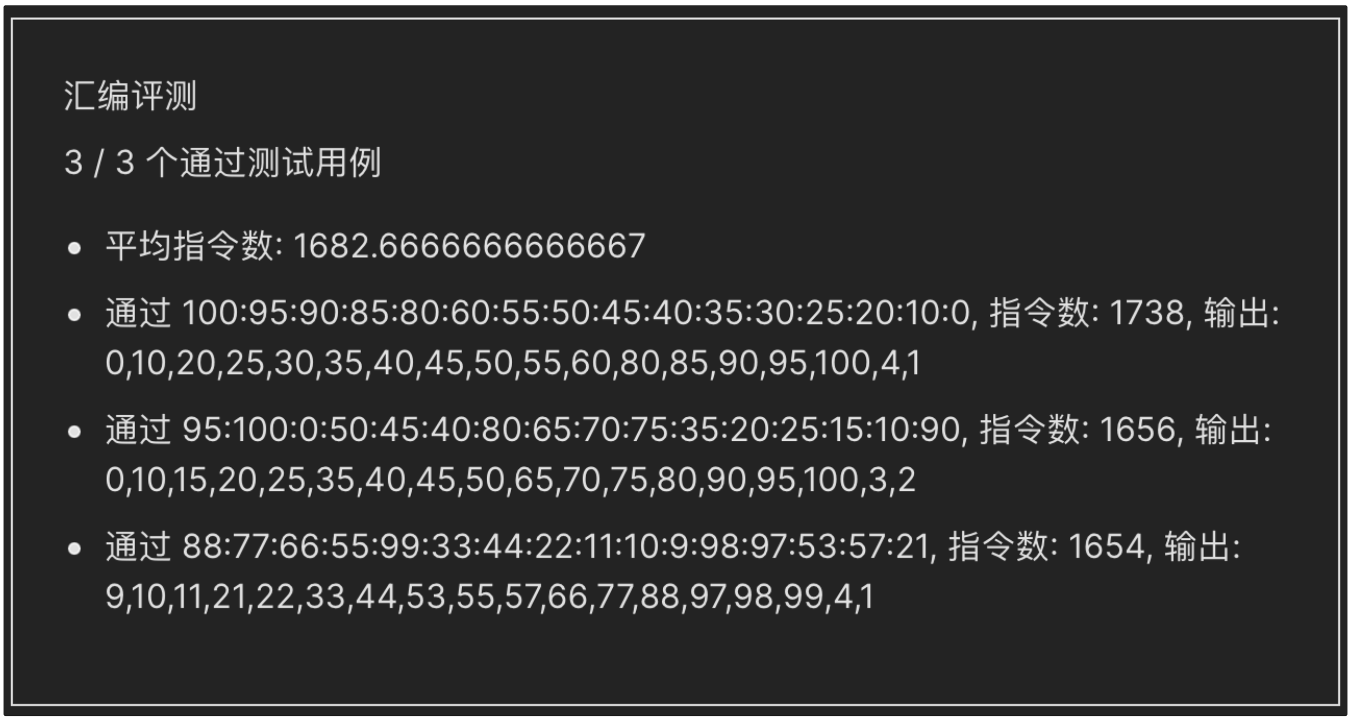
Code

```
.ORIG X3000
        AND     R0, R0, #0
        AND     R1, R1, #0
        AND     R2, R2, #0
        AND     R3, R3, #0
        AND     R4, R4, #0
        AND     R5, R5, #0
        AND     R6, R6, #0      ;STACKPOINTER
        AND     R7, R7, #0      ;RETURN LINKAGE
        LD      R6, STACKX6000      ;STACKPOINTER
        LD      R0, STOREX4000      ;WHERE THE 16 SCORES ARE STORED
        LD      R1, STOREX5000      ;WHERE THE SORTED SCORES ARE STORED
        JSR     COPY
        JSR     SORT
        JSR     COUNT
        HALT
        ;-----------------------------------------------------------------------------------------------------------------
        ;LABEL: COPY
        ;FUNCTION: COPY THE INPUT ARRAY TO THE OUTPUT ARRAY
        ;PARAMETER: I(R2), (I - 16)(R3), (-16)(R4), VALUE OF R0(R5)
        ;INPUT: R0 = INPUT ARRAY
        ;OUTPUT: R1 = OUTPUT ARRAY
        COPY    ADD     R6, R6, #-1
                STR     R7, R6, #0      ;SAVE RETURN LINKAGE
                ADD     R6, R6, #-1
                STR     R2, R6, #0      ;SAVE R2, WHICH WILL BE USEED AS I
                ADD     R6, R6, #-1
                STR     R3, R6, #0      ;SAVE R3, WHICH WILL BE USED AS (I - 16)
                ADD     R6, R6, #-1
                STR     R4, R6, #0      ;SAVE R4, WHICH WILL BE USED AS 16
                ADD     R6, R6, #-1
                STR     R5, R6, #0      ;SAVE R5, WHICH WILL BE USED AS VALUE OF R0
                AND     R2, R2, #0      ;I = 0
                LD      R4, STORE16      ;R4 = 16
                NOT     R4, R4
                ADD     R4, R4, #1      ;R4 = -16
        LOOP1   ADD     R3, R2, R4      ;R3 = (I - 16)
                BRZP    ENDCOPY         ;IF R3 < 0, END LOOP
                LDR     R5, R0, #0      ;R5 = VALUE OF R0
                STR     R5, R1, #0      ;STORE R5 IN R1
                ADD     R2, R2, #1      ;I = I + 1
                ADD     R0, R0, #1      ;R0 = R0 + 1
                ADD     R1, R1, #1      ;R1 = R1 + 1
                BRNZP   LOOP1

        ENDCOPY LDR     R5, R6, #0      ;RESTORE R5
                ADD     R6, R6, #1
                LDR     R4, R6, #0      ;RESTORE R4
                ADD     R6, R6, #1
                LDR     R3, R6, #0      ;RESTORE R3
                ADD     R6, R6, #1
                LDR     R2, R6, #0      ;RESTORE R2
                ADD     R6, R6, #1
                LDR     R7, R6, #0      ;RESTORE RETURN LINKAGE
                ADD     R6, R6, #1
                RET
        ;----------------------------------------------------------------------------------------------------------------
        ;LABEL: SORT
        ;FUNCTION: SORT THE ARRAY
        ;PARAMETER: (R1), (R2), (R3), R4, R5
        ;INPUT:
        ;OUTPUT:Y
        SORT    ADD     R6, R6, #-1
                STR     R7, R6, #0      ;SAVE RETURN LINKAGE
                ADD     R6, R6, #-1
                STR     R1, R6, #0      ;SAVE R1
                ADD     R6, R6, #-1
                STR     R2, R6, #0      ;SAVE R2
                ADD     R6, R6, #-1
                STR     R3, R6, #0      ;SAVE R3, WHICH WILL BE USED AS POINTER TO THE ARRAY
                ADD     R6, R6, #-1
                STR     R4, R6, #0      ;SAVE R4, WHICH WILL BE USEED AS I
                ADD     R6, R6, #-1
                STR     R5, R6, #0      ;SAVE R5, WHICH WILL BE USED AS J
                ;----------------------------------------------------------------------------------------------------------------
                LD      R4, STORE16
                OUTERLOOP       ADD     R4, R4, #-1 ; loop n - 1 times
                                BRNZ    SORTED      ; Looping complete, exit
                                ADD     R5, R4, #0  ; Initialize inner loop counter to outer
                                LD      R3, STOREX5000    ; Set file pointer to beginning of ARRAY
                INNERLOOP       LDR     R0, R3, #0  ; Get item at ARRAY pointer
                                LDR     R1, R3, #1  ; Get next item
                                NOT     R2, R1      ; Negate ...
                                ADD     R2, R2, #1  ; ... next item
                                ADD     R2, R0, R2  ; swap = item - next item
                                BRNZ    SWAP        ; Don't swap if in order (item ≤ next item)
                                STR     R1, R3, #0  ; Perform ...
                                STR     R0, R3, #1  ; ... swap
                SWAP            ADD     R3, R3, #1  ; Increment file pointer
                                ADD     R5, R5, #-1 ; Decrement inner loop counter
                                BRP     INNERLOOP   ; End of inner loop
                                BRNZP   OUTERLOOP   ; End of outer loop
                        ;----------------------------------------------------------------------------------------------------------------
        SORTED  LDR     R5, R6, #0      ;RESTORE R5
                ADD     R6, R6, #1
                LDR     R4, R6, #0      ;RESTORE R4
                ADD     R6, R6, #1
                LDR     R3, R6, #0      ;RESTORE R3
                ADD     R6, R6, #1
                LDR     R2, R6, #0      ;RESTORE R2
                ADD     R6, R6, #1
                LDR     R1, R6, #0      ;RESTORE R1
                ADD     R6, R6, #1
                LDR     R7, R6, #0      ;RESTORE RETURN LINKAGE
                ADD     R6, R6, #1
                RET
        ;---------------------------------------------------------------------------------------------------
        ;LABEL: COUNTA
        COUNT   ADD     R6, R6, #-1
                STR     R7, R6, #0      ;SAVE RETURN LINKAGE
                ADD     R6, R6, #-1
                STR     R0, R6, #0      ;SAVE R0
                ADD     R6, R6, #-1
                STR     R1, R6, #0      ;SAVE R1
                ADD     R6, R6, #-1
                STR     R2, R6, #0      ;SAVE R2
                ADD     R6, R6, #-1
                STR     R3, R6, #0      ;SAVE R3
                ADD     R6, R6, #-1
                STR     R4, R6, #0      ;SAVE R4
                ADD     R6, R6, #-1
                STR     R5, R6, #0      ;SAVE R5
                LD      R0, STOREX500F  ;ARRAY POINTER
                LDR     R1, R0, #0      ;R1 = A[15]
                AND     R2, R2, #0      ;R2 = 0, USED AS LOOP COUNTER
                ADD     R2, R2, #4      ;COUNTER = 4
                AND     R4, R4, #0      ;R4 IS COUNTA
                AND     R5, R5, #0      ;R5 IS COUNTB
                LOOPA   ADD     R2, R2, #-1     ;
                        BRN     OUTLOOPA
                        IFA     ADD     R6, R6, #-1
                                STR     R2, R6, #0
                                LD      R3, STOREN85
                                ADD     R2, R1, R3      ;R2 = R1 - 85
                                BRN     ELIFA
                                ADD     R4, R4, #1      ;COUNTA++
                                BRNZP   ENDIFA
                        ELIFA   LD      R3, STOREN75
                                ADD     R2, R1, R3      ;R2 = R1 - 75
                                BRN     ENDIFA
                                ADD     R5, R5, #1
                        ENDIFA  LDR     R2, R6, #0
                                STR     R6, R6, #1
                        ADD     R0, R0, #-1
                        LDR     R1, R0, #0
                        BRNZP   LOOPA
                OUTLOOPA
                AND     R2, R2, #0      ;R2 = 0, USED AS LOOP COUNTER
                ADD     R2, R2, #4      ;COUNTER = 4
                LOOPB   ADD     R2, R2, #-1     ;
                        BRN     OUTLOOPB
                        IFB     ADD     R6, R6, #-1
                                STR     R2, R6, #0
                                LD      R3, STOREN75
                                ADD     R2, R1, R3      ;R2 = R1 - 75
                                BRN     ENDIFB
                                ADD     R5, R5, #1      ;COUNTA++
                        ENDIFB  LDR     R2, R6, #0
                                STR     R6, R6, #1
                        ADD     R0, R0, #-1
                        LDR     R1, R0, #0
                        BRNZP   LOOPB
                OUTLOOPB
                STI    R4, STOREX5100
                STI    R5, STOREX5101
                LDR     R5, R6, #0      ;RESTORE R5
                ADD     R6, R6, #1
                LDR     R4, R6, #0      ;RESTORE R4
                ADD     R6, R6, #1
                LDR     R3, R6, #0      ;RESTORE R3
                ADD     R6, R6, #1
                LDR     R2, R6, #0      ;RESTORE R2
                ADD     R6, R6, #1
                LDR     R1, R6, #0      ;RESTORE R1
                ADD     R6, R6, #1
                LDR     R0, R6, #0      ;RESTORE R0
                ADD     R6, R6, #1
                RET
        ;---------------------------------------------------------------
        STOREX4000  .FILL   X4000
        STOREX5000  .FILL   X5000
        STOREX5100  .FILL   X5100
        STOREX5101  .FILL   X5101
        STACKX6000  .FILL   X6000
        STOREX500F  .FILL   X500F
        STORE16     .FILL   X0010
        STOREN75    .FILL   XFFB5
        STOREN85    .FILL   XFFAB
.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: 1682.6666666666667
- 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