Systems & Networks 2017 Assessed Exercise

Paul McHard - 2085227M

Tuesday 21st November, 2017

Status Report

Program State: The program has been fully completed as per the specification and returns expected output for **"Acceptance Test"** data. Additional test data was also used, as shown in the Evidence Table below, and works accurately.

Limitations and Bugs: Three limitations exist for this code. Firstly, if data is added or removed from array X by the user without altering the value of n accordingly, the program will run as if there is no problem and not observe an error, but will return incorrect results. The program is incapable of ensuring that the true size of array X and the integer value n, which is taken as array size, are the same. The other limitations both arise from Sigma16 itself. As it uses a 16-bit architecture, it is incapable of handling values outwith the range of 16 bit two's complement: -32,768 to 32,767. This counts as much for input values as for output, as even if all input values are within the range it could cause overflow in **possum**. Again due to architecture limitations, there is a maximum of 64kB of addressable memory, which limits possible array length. The program has no known bugs that have been found.

Evidence: The following data was used to test the system.

Data	**possum**	**oddcount**	**negcount**
X = [3, -6, 27, 101, 50, 0, -20, -21, 19, 6, 4, -10]	210	4	4
X= [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	0	0	0
X =[1, 3, 5, 7, 9, -8. -6, -4, -2, 0]	25	5	4
X= [15879, 3443, 5817, 117, 809, -844, -652, -32768]	26065	5	3

Feedback

Code Listing

```
;Systems and Networks Assessed Exercise

;The program takes in:
;       - An integer value n, assume n > 0.
;       - An array X of length n.
;It will calculate:
;       - Possum: Sum of positive values in array X.
;       - Negcount: The count of negative numbers in array.
;       - oddcount: The count of positive odd numbers in the array.

;Psuedo-HLL program:
;       n=12;
;       X = {3, -6, 27, 101, 50, 0, -20, -21, 19, 6, 4, -10};
;       for( i=0 ; i<n ; i++ )
;               if (X[i] >= 0)
;                       possum+=X[i]
;                       if(X[i] % 2 != 0)
;                               oddcount++
;                       else
;                               negcount++

; Register usage
;       R1 = constant 1
;       R2 = n
;       R3 = i
;       R4 = X[i]
;       R5 = possum
;       R6 = oddcount
;       R7 = negcount
;       R8 = check ( i < n )
;       R9 = check ( X[i] < 0 ) Check negative
;       R10= check ( X[i] == ODD ) := Bitwise AND
;       R11= check ( X[i] == ODD ) := Boolean

; Initialise

        LEA     R1,1[R0]        ; R1 = constant 1
        LOAD    R2,n[R0]        ; R2 = n
        LEA     R3,0[R0]        ; R3 = i = 0
        LOAD    R4,X[R0]        ; R4 = X[0]
        LOAD    R5,possum[R0]   ; R5 = possum = 0
        LOAD    R6,ocount[R0]   ; R6 = oddcount = 0
        LOAD    R7,ncount[R0]   ; R7 = negcount = 0

;Top of FOR loop needs to check to remain in loop

floop   CMPLT   R8,R3,R2        ;R8 = ( i < n )
        JUMPF   R8,end[R0]      ;if i>n goto end

;IF (X[i] >= 0)

```

```
52  ifpos   LOAD    R4,X[R3]        ;R4 = X[i]
53          CMPLT   R9,R4,R0        ;R9 = ( X[i] < 0 )
54          JUMPT   R9,else[R0]     ;jump to else if negative
55
56  ;then possum += X[i]
57
58          ADD     R5,R5,R4        ;R5 = possum+X[i]
59
60  ;IF ( X[i] == ODD )
61
62
63          AND     R10,R4,R1       ;R10 = bitwise AND on X[i] and constant 1
64          CMPEQ   R11,R10,R1      ;R11 = (R10 == 1)
65          JUMPF   R11,incr[R0]    ;if X[i] even goto bottom of loop
66
67  ;then increment oddcount
68
69          ADD     R6,R6,R1        ;oddcount++
70          JUMP    incr[R0]        ;goto bottom of loop, always skip else from here
71
72  ;ELSE (on first IF) incremenent negcount
73
74  else    ADD     R7,R7,R1        ;negcount++
75
76  ;no goto bottom needed after negcount++ as else is last operation before bottom
77
78  ;Bottom of loop, increment i and goto top of loop
79
80  incr    ADD     R3,R3,R1        ;i++
81          JUMP    floop[R0]       ;goto top of for loop
82
83  ;Exit from the for loop
84  end     STORE   R5,possum[R0]   ;possum = R5
85          STORE   R6,ocount[R0]   ;oddcount = R6
86          STORE   R7,ncount[R0]   ;negcount = R7
87          TRAP    R0,R0,R0        ;terminate program
88
89  ;Set Data values
90  n       DATA    12      ;n = 12
91  X       DATA    3;      ;X[0]  = 3
92          DATA    -6      ;X[1]  = -6
93          DATA    27      ;X[2]  = 27
94          DATA    101     ;X[3]  = 101
95          DATA    50      ;X[4]  = 50
96          DATA    0       ;X[5]  = 0
97          DATA    -20     ;X[6]  = -20
98          DATA    -21     ;X[7]  = -21
99          DATA    19      ;X[8]  = 19
100         DATA    6       ;X[9]  = 6
101         DATA    4       ;X[10] =4
102         DATA    -10     ;X[11] =-10
103 possum  DATA    0       ;initialise sum of positives to zero
104 ocount  DATA    0       ;initialise oddcount to zero
105 ncount  DATA    0       ;initialise negcount to zero
```