Tutorial #4
Hilary Term Weeks 10 and 11
General Question

Work in groups of 6 students using one of the whiteboards.
Complete the exercises in your own time if necessary.
Revise the exercises to prepare for the next lab.

Design and write an ARM Assembly Language subroutine that will determine whether a square two-dimensional array is a sub-array of a larger square two-dimensional array. For example, in the figure below, B is a sub-array of A.

48	37	15	44	3	17	26
2	9	12	18	14	33	16
13	20	1	22	7	48	21
27	19	44	49	44	18	10
29	17	22	4	46	43	41
37	35	38	34	16	25	0
17	0	48	15	27	35	11

A

49	44	18
4	46	43
34	16	25

B

Your answer must include:

(i) a detailed explanation of your approach, including pseudo-code,

(ii) an ARM Assembly Language listing for your subroutine, including a description of its interface.

Complete the exercises in your own time if necessary!

Tutorial #4 SOLUTION
Hilary Term Weeks 10 and 11
General Question

Sketch solution! Not tested!

The key idea is to iterate first over every element LARGE[i][j] in the large array. For each of these elements, take the element as the starting element for a comparison with the smaller array, iterating over each element SMALL[x][y]. The comparison is between LARGE[i+x][j+y] and SMALL[x][y].

Iterating over the outer array, we assume we don't have a match and keep iterating until we find a match. Iterating over the inner array, we assume we do have a match and try to disprove that assumption.

The iteration over the larger array can be truncated in its rows and columns by the size of the smaller array.

```
1         AREA    RESET, CODE, READONLY
2         ENTRY
3
4         LDR     R0, =LGE
5         LDR     R1, =LGE_SZ
6         LDR     R2, =SML
7         LDR     R3, =SML_SZ
8         BL      subarray
9
10 stop   B       stop
11
12 ; subarray
13 ; paramaters:
14 ;       R0: large array start address
15 ;       R1: large array size
16 ;       R2: small array start address
17 ;       R3: small array size
18 subarray
19        STMFD   SP!, {R4-R12, LR}
20        SUB     R12, R1, R3       ; limit = largeSize - smallSize
21        MOV     R11, #0           ; result = FALSE
22        MOV     R4, #0            ; for (i = 0; i <= limit && !result; i++) {
23 fori                            ;
24        CMP     R11, #1           ;
25        BEQ     efori             ;
26        CMP     R4, R12           ;
27        BHI     efori             ;
28        MOV     R5, #0            ;   for (j = 0; j <= limit && !result; j++) {
29 forj                            ;
30        CMP     R11, #1           ;
31        BEQ     eforj             ;
32        CMP     R5, R12           ;
33        BHI     eforj             ;
34        MOV     R11, #1           ;     resut = TRUE
35        MOV     R6, #0            ;     for (x = 0; x < smallSize && result; x++) {
36 forx                            ;
37        CMP     R11, #1           ;
```

```
38          BNE     eforx               ;
39          CMP     R6, R3              ;
40          BHS     eforx               ;
41          MOV     R7, #0              ;       for (y = 0; y < smallSize && result; y++) {
42  fory                                ;
43          CMP     R11, #1             ;
44          BNE     efory               ;
45          CMP     R7, R3              ;
46          BHS     efory               ;
47          ADD     R8, R4, R6          ;         lge = LARGE[i+x][j+y]
48          MUL     R8, R1, R8          ;
49          ADD     R8, R8, R5          ;
50          ADD     R8, R8, R7          ;
51          LDR     R9, [R0, R8, LSL #2];
52          MUL     R8, R6, R3          ;         sml = SMALL[x][y]
53          ADD     R8, R8, R7          ;
54          LDR     R10, [R2, R8, LSL #2];
55          CMP     R9, R10             ;         if (lge != sml) {
56          BEQ     stillEqual          ;
57          MOV     R11, #0             ;           result = 0
58  stillEqual                          ;         }
59          ADD     R7, R7, #1          ;
60          B       fory                ;       }
61  efory                               ;
62          ADD     R6, R6, #1          ;
63          B       forx                ;     }
64  eforx                               ;
65          ADD     R5, R5, #1          ;
66          B       forj                ;   }
67  eforj                               ;
68          ADD     R4, R4, #1          ;
69          B       fori                ; }
70  efori                               ;
71          MOV     R0, R11             ;
72          LDMFD   SP!, {R4-R12, PC}   ; return result
73
74
75  LGE_SZ  EQU     7
76  SML_SZ  EQU     3
77
78  LGE     DCD     48,37,15,44, 3,17,26
79          DCD     2, 9, 12,18,14,33,16
80          DCD     13,20, 1,22, 7,48,21
81          DCD     27,19,44,49,44,18,10
82          DCD     29,17,22, 4,46,43,41
83          DCD     37,35,38,34,16,25, 0
84          DCD     17, 0,48,15,27,35,11
85
86  SML     DCD     49,44,18
87          DCD      4,46,43
88          DCD     34,16,25
89
90          END
```