CMSC 411 – Homework #3 – Due 9/26/2017

Remember:

If $\mathbf{A} = \begin{pmatrix} a & b & c \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} x \\ y \\ z \end{pmatrix},$ their matrix products are:

$$\mathbf{AB} = \begin{pmatrix} a & b & c \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = ax + by + cz, \quad \text{and} \quad \mathbf{BA} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \begin{pmatrix} a & b & c \end{pmatrix} = \begin{pmatrix} xa & xb & xc \\ ya & yb & yc \\ za & zb & zc \end{pmatrix}.$$

If $A = [1\ 2\ 3\ 4]$ and $B = [9\ 8\ 7\ 6]$

MatLab is an array based language so we can just say *C = AB and D = BA* to multiply the matrixes

[20 points] Write the C++ code to do *C = AB* and *D = BA* with these arrays:

C++ Code:

```
#include <iostream>
using namespace std;

int main()
{
   const int len = 4;
   int A[len] = {1, 2, 3, 4};
   int B[len] = {9, 8 , 7, 6};

   // multiply A * B
   int res[1] = {0};                    // result array
   for (int i=0; i<len; i++) {
      res[0] += A[i] * B[i];
   }
   cout << "A * B result: [" << res[0] << "]" << endl;

   // multiply B * A
   int arr[len][len];                   // result array
   cout << "\nB * A result: " << endl;

   for (int i=0; i<len; i++) {          // iterate over each row of B
      for (int j=0; j<len; j++) {       // now iterate over each col of A
         arr[i][j] = B[i] * A[j];
         cout << arr[i][j] << " ";
      }
      cout << endl;
   }
return 0;
}
```

CMSC 411 – Homework #3 – Due 9/26/2017

[80 points] Write the ARM code to do this with arrays. You can have the input and output the arrays are in either memory or registers. Provide your code and a screen shot of the ARMSim results that show the final state (below) as PDF to Blackboard. Be sure to identify, label, and highlight the input array values and output array values and write your name on your work.

Free Simulators:
ARMSIM: http://webhome.cs.uvic.ca/~nigelh/ARMSim-V2.1/index.html for Windows
Emulates ARM7TDMI-S: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.set.arm7/index.html
Visual: http://salmanarif.bitbucket.org/visual/ for Linux
Keil: http://www.keil.com/support/man_arm.htm

General ARM information:
http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.dui0058d/Chdcdbib.html

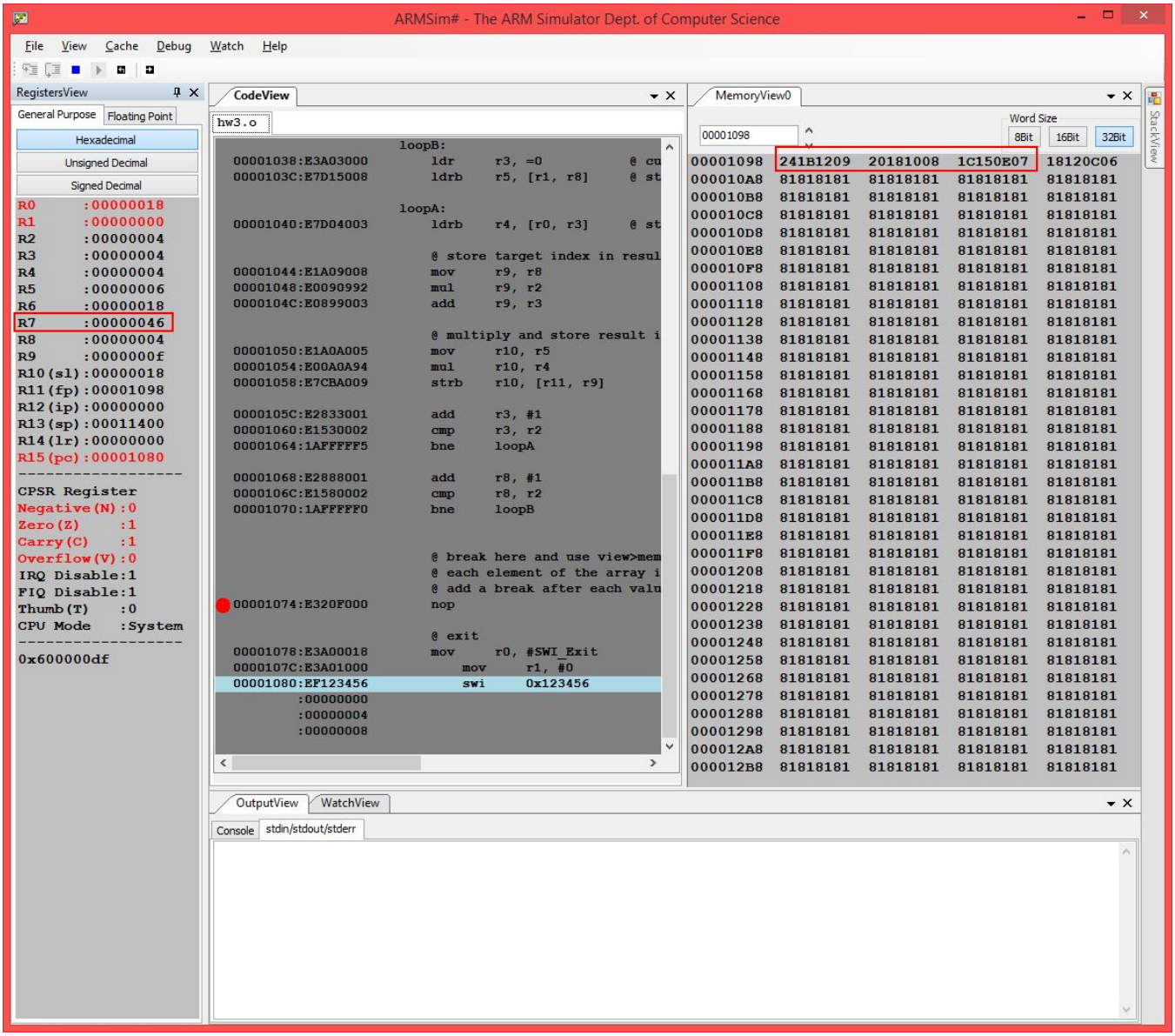
ARM Code:

```
@ Dan Engbert - CMSC 411, Fall 17
@ HW3: multiply arrays
@ run this program in ARMSim

@ *************** Data Segment ***************
        .data
arrA:   .byte   1, 2, 3, 4    @ input array values
arrB:   .byte   9, 8, 7, 6    @ input array values
@ TODO: could also make res a huge array and then the size to use could be chosen later
res:    .byte   0,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,0,0


@ *************** Code Segment ***************
        .equ    SWI_Write, 0x05
        .equ    SWI_Exit,  0x18

        .text
_start:
   @ store address of arrays in registers r0 and r1
   ldr    r0, =arrA
   ldr    r1, =arrB
   ldr    r2, =4          @ length of arrays
   ldr    r11, =res       @ result array

   @ multiply A * B (results is 1x1 matrix)
   ldr    r3, =0          @ current position in array
   ldr    r7, =0          @ result value

loop:
   ldrb   r4, [r0, r3]    @ store element of arrA at position #r3 in r4
   ldrb   r5, [r1, r3]    @ store element of arrB at position #r3 in r5

   mul    r6, r4, r5      @ multiply the two elements
   add    r7, r6          @ add the result to the running total
```

```
   add     r3, #1          @ increment index
   cmp     r3, r2
   bne     loop            @ loop until we reach the end of the array
   @ resulting value is now stored in r7


   @ multiply B * A (results is 4x4 matrix, stored in res)
   ldr     r8, =0          @ current position in arrB
loopB:
   ldr     r3, =0          @ current position in arrA
   ldrb    r5, [r1, r8]    @ store element of arrB at position #r8 in r5

loopA:
   ldrb    r4, [r0, r3]    @ store element of arrA at position #r3 in r4

   @ store target index in result array in r9
   mov     r9, r8
   mul     r9, r2
   add     r9, r3

   @ multiply and store result in res array
   mov     r10, r5
   mul     r10, r4
   strb    r10, [r11, r9]

   add     r3, #1
   cmp     r3, r2
   bne     loopA

   add     r8, #1
   cmp     r8, r2
   bne     loopB


   @ break here and use view>memory in ARMSim to view the hex values stored in the res arary (address in r11)
   @ each element of the array is 1 byte (2 hex characters)
   @ add a break after each value is set above to see how the array is populated
   nop

   @ exit
   mov     r0, #SWI_Exit
         mov      r1, #0
         swi      0x123456
```

CMSC 411 – Homework #3 – Due 9/26/2017

My output:



Explanation:

The arrays being multiplied are highlighted in the code.

The output values are shown above with red boxes around them.

The result of A * B is a 1x1 array and so the result is stored in r7, and 0x46 = 70 (decimal) which is correct

The result of B * A is a 4x4 array and the result is stored in memory at address 0x1098.

r11 contains the address of the result array (called res in the code).

The values in memory corresponding to the result array are highlighted in the memory view in the screenshot above. There are 16 elements in the result array and each element is 1 byte (2 hex characters).

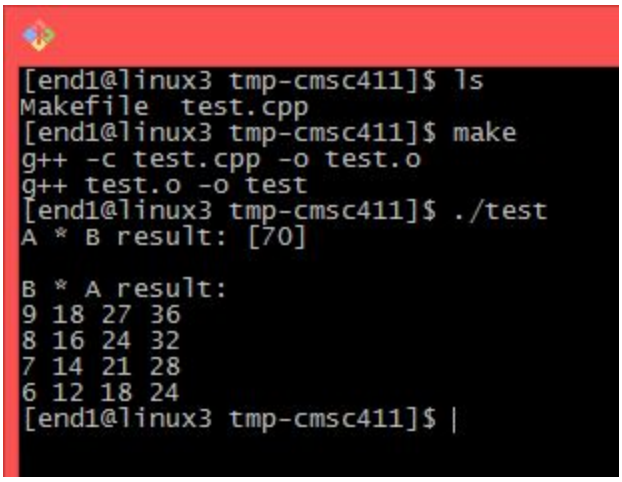
For example the first 4 bytes are 0x241B1209.

based off my observations of the way ARMSim stores values in memory, each group of 4 bytes is filled from right

Due 9/26/2017

to left. So the first element of our array is 0x09 = 9 (decimal)
The second element is 0x12 = 18 (decimal)
The third element is 0x1B = 27 (decimal)
The fourth element is 0x36 = 36 (decimal)
etc.
When you convert all these hexadecimal values to decimal, you can see that the result is
9 18 27 36 8 16 24 32 7 14 21 28 6 12 18 24
(every group of 4 number represents a row of the result matrix)
This is the correct result for B * A.

C++ Code Output:



Extra credit: 25 points if you make work with variable length matrixes