Lab #10 – Floating Point Arithmetic

Goals:

Write an assembly code function on the Discovery Board that performs floating point arithmetic

Experiment with floating point debugging windows

For each step enter the necessary responses into the Canvas assignment text entry tool to let the TA follow your work.

Remember the '`SVC #0`' debugging instruction available with your Discover Board library. That can be inserted in your assembly code any time you need to clarify exactly what is in the core registers R0 - R12. In addition, in this lab we will learn to use the '`SVC #1`' floating point debugging breakpoint as well to inspect floating point registers S0-S31.

Refer to the Lecture slides for the following:

Part 1:

If you are not already familiar with the determinant measure of square matrices from linear algebra, first visit https://en.wikipedia.org/wiki/Determinant for an introduction. Consider a system of two simultaneous linear equations in two unknown variables x_1 and x_2. This can be written in the form

$$a_{11}x_1 + a_{12}x_2 = b_1$$
$$a_{21}x_1 + a_{22}x_2 = b_2$$

where the a_{ij} are constant nonzero coefficients and the b_j are assumed to be nonzero. The system may have a single consistent solution for the variables x_1 and x_2 , or have no unique solution. To determine whether or not the system has a solution we can form a square matrix of the equation coefficients

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

and compute the determinant, as in the Wikipedia article, as $D = a_{11}a_{22} - a_{12}a_{21}$ If the determinant D is nonzero, the corresponding linear system has a unique solution for x_1 and x_2 . But if D = 0, no unique solution exists, and the matrix is said to be "singular". In this lab we will compute some determinants of matrices in an assembly language function and test for singular matrices in the calling C program. The coefficients of the equations will be represented as floating point variables.

a) Download Lab10-DeterminantMain.c and Lab10-determinant.s from the Canvas files area, and in the LabCode folder.

First find the location in the C code where the array A is declared as an array of four floats, and initialized with four matrix coefficients. We will follow the usual C standard of storing a two-dimensional matrix "row-wise" in an array, that is, the sequence of array elements starts with the elements of the top row of the matrix in order, followed by the second row, etc. (This is the same ordering as the column and row pixel elements of the Discovery Board display you have been working with.) For example, to initialize the array A to a matrix $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ you would include the line in the C code

```
float A[4] = {1.0, 2.0, 3.0, 4.0};
```

Now complete the assembly code for the function `determinant` in the assembly file. Note the prototype for this function in the C main program declared the single function argument to be of type pointer to float, and the return value to be of type float. Write the assembly code to expect an address to the array of float coefficients to be passed as an argument in R0. This is similar to your code in Lab #9 when the address of a character array was passed to your function. However, as you write the memory addressing code bear in mind that elements in a float array are 32-bits in size, similar to 32-bit integers, and therefore require four byte memory address locations for each element. Write the assembly code to use the floating point instructions `VLDR`, `VMUL`, `VSUB` with the floating point registers S0, S1, and S2 to calculate the matrix determinant and return the value in S0.

Start with the matrix $A = \begin{bmatrix} 4 & 6 \\ 3 & 8 \end{bmatrix}$. First calculate by hand the expected determinant of this matrix. Then write the C initialization statement to create this matrix in the A array and test your assembly function. Enter into the Canvas text entry your expected determinant and the result returned and displayed on the screen. Upload your code and a cell phone photo.

b) Insert the breakpoint instruction '`SVC #1`' between the statements in your function code. This is similar to the '`SVC #0`' instruction that pauses execution and displays the core registers, but '`SVC #1`' instead displays the floating point registers for debugging. Trace the execution of your function code.

c) Now change the C initialization statement to enter the matrix $A = \begin{bmatrix} 1.75 & 3.5 \\ -0.5 & -1.0 \end{bmatrix}$. What determinant do you expect? What determinant value is returned? A matrix is singular if multiplying all the elements in one matrix row by some constant will give all the elements in another row. Find a way that this test confirms the determinant value for this matrix. Enter your answers in Canvas.

d) Now change the C initialization statement to enter the matrix $A = \begin{bmatrix} 0.1 & 0.6 \\ -0.7 & -4.2 \end{bmatrix}$. Using the same principle for a singular matrix as in part c), is there a common factor between the row elements of this matrix? Is this consistent with the numerical results of your code? Why or why not? Enter your answers in Canvas.

Part 2: Time to work on the current programming project assignment.