1. Write a function in ARM Cortex-M4 assembly language to calculate the area of a circle. Write a C program to test your function. The function prototype is:

```
float  CircleArea(float radius) ;
```

```
CircleArea: // S0 = radius
            VMUL.F32    S0,S0,S0    // S0 = radius * radius
            VLDR        S1,pi       // S1 = 3.14159
            VMUL.F32    S0,S0,S1    // S0 = 3.14159*radius*radius
            BX          LR

pi:         .float 3.14159
```

2. Write a function in ARM Cortex-M4 assembly language to compute the dot product of two vectors. Write a C program to test your function. The function prototype is:

```
float DotProduct(float vec1[], float vec2[], int32_t len) ;
```

```
DotProduct: // R0 = &vec1[0], R1 = &vec2[0], R2 = len
            VSUB.F32    S0,S0,S0    // sum = 0.0
L1:         CBZ         R2,L2       // Done if len == 0
            VLDR        S1,[R0]     // S1 = *vec1
            ADD         R0,R0,4     // vec1++
            VLDR        S2,[R1]     // S2 = *vec2
            ADD         R1,R1,4     // vec2++
            VMLA.F32    S0,S1,S2    // sum += S1*S2
            SUB         R2,R2,1     // items--
            B           L1          // repeat
L2:         BX          LR          // return
```

Faster if replaced by:

```
VLDMIA   R0!,{S1}
VLDMIA   R1!,{S2}
```

6. Write a function in assembly language to compute the arithmetic mean (average) of an array of floating-point values, given by:

$$\text{Mean } (\sigma) = \frac{1}{n}\sum_{i=0}^{n-1} x_i$$

Write a C program to test your function. The function prototype is:

```
float Mean(float x[], uint32_t n) ;
```

```
Mean:       // R0 = &x[0], R1 = n (assume n > 0)
            VSUB        S0,S0,S0    // sum = 0.0
            VCVT.F32.U32 S2,R1      // S2 ← (float) n
loop:       CBZ         R1,done
            VLDMIA      R0!,{S1}    // S1 ← x[i]
            VADD.F32    S0,S0,S1    // sum += x[i]
            SUB         R1,R1,1     // n ← n - 1
            B           loop        // repeat
done:       VDIV.F32    S0,S0,S2    // average = sum/n
            BX          LR          // return
```

8. The standard deviation of a set of values is the square root of their variance. Consider the following function that computes the standard deviation. Draw a clock cycle timing diagram similar to **Error! Reference source not found.** and determine the worst-case number of clock cycles required to execute the three instructions of the function:

```
// void StdDev(float var, float *result) ;

StdDev:   VSQRT    S0,S0       // S0 = Std. Dev.
          VSTR     S0,[R0]     // Store result
          BX       LR
```

clock cycle	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VSQRT	F	D	E	E	E	E	E	E	E	E	E	E	E	E	
VSTR		F	D	S	S	S	S	S	S	S	S	S	S	S	E
BX			F	D	E	E									

F = "Fetch phase", D = "Decode phase", E = "Execute phase", S = "Stall phase"

Notes:

(1) The VSTR cannot complete it's execute phase until the result is available from the VSQRT. However, the BX does not depend on either of the previous two instructions and can thus proceed without waiting. Ultimately, the result stored by the VSTR will be used wherever the function was called when it tries to reference the result, and then that's where the stall will occur again.

(2) I can't find any documentation about this, but it's quite possible that the BX instruction will also stall, simply because it would become a complex hardware design for the hardware to know what the instruction at the destination of the branch will be and thus whether or not it will need the result of the VSTR. If the BX does stall, it's execute phase would begin at clock cycle 16 in the above figure.