1.

```
float CircleArea(float radius)
CircleArea:     // R0 = radius
                // Area = pi * r^2
                VLDR          R1, pi
                VMUL.F32      R0, R0, R0     // R0 = r^2
                VMUL.F32      R0, R1, R0
                BX            LR
pi:             .float    3.14159
```

2.

```
float DotProduct(float vec1[], float vec2[], int32_t len)
DotProduct:     // S0 = vec1[], S1 = vec2[], R0 = len
                LDR           R1, =0         // i for the loop
                VMOV          S2, = 0.0
next:           CMP           R1, R0
                BGE           done
                VLDR          S0, [S0, R1]   // S0 = vec1[i]
                VLDR          S1, [S1, R1]   // S1 = vec2[i]
                VMLA.F32      S2, S0, S1, S2         // S2 = S2 + (S0 * S1)
                ADD           R1, R1, 4      // i++
                B             next
done:           VMOV          S0, S2
                BX            LR
```

6.

```
float Mean(float x[], uint32_t n)
Mean:   // S0 = x[], R0 = n
        LDR           R1, =0
        VLDR          S1, =0.0
next:   CMP           R1, R0
        BHS           done    // i <= n - 1 → i < n
        VLDR          S0, [S0, R1]    // S0 = x[i]
        VADD.F32      S1, S1, S0
        ADD           R1, R1, 1       // R1 = R1 + 1
        BL            next
done:
        // Method #1:
        VMOV          S2, R0
        VDIV          S0, S1, S2      // S0 = sum/n
```

// I'm thinking about using LSR so we can have fewer clock cycle, but not sure if we can directly use LSR on float number, or we need to MOV S1 to R2, LSR R2, then MOV R2 back to S1?

```
        // Method #2
        VMOV        R2, S1
        LSR         R2, R2, R0      // sum >> n
        VMOV        S1, R2
        VMOV        S0, S1
        BX          LR
```

8.

```
void StdDev(float var, float *result)
StdDev:     // S0 = var, S1 = *result
            VSQRT       S0, S0
            VSTR        S0, [R0]
            BX          LR
```

Fetch	Decode	Execute	Execute		
	Fetch	Decode	Stall	Execute	Execute