1.

```
CircleArea:
        push    {r4, lr}
        vmov    s0, r0
        vmul.f32    s0, s0, s0
        vldr    s1, =3.14159
        vmul.f32    s0, s0, s1
        pop     {r4, pc}
```

2.

```
DotProduct:
        push    {r4, r5, r6, lr}
        mov     r4, r0
        mov     r5, r1
        mov     r6, r2
        mov     r1, 0
dot_loop:
        ldr     s0, [r4], 4
        ldr     s1, [r5], 4
        vmul.f32    s2, s0, s1
        vadd.f32    s1, s1, s2
        subs    r6, r6, #1
        bne     dot_loop
        pop     {r4, r5, r6, pc}
```

5.

a.

```
Inverse:
        push    {r4, r5, r6, r7, lr}
        mov     r4, r0
        sub     r4, r4, 1
        mov     r5, r4
        mov     r7, 0
        mov     r6, r1
        ldr     r3, [r6], 4
        mov     r1, 0
inverse_loop:
        cmp     r7, r2
        bge     inverse_end
        mov     r2, r7
        bl      Polynomial
```

```
        subs    r7, r7, 1
        cmp     r7, 0
        bne     inverse_add
        add     r3, r3, 1
inverse_add:
        cmp     r7, 0
        beq     inverse_subtract
        cmp     r7, 1
        beq     inverse_subtract
        cmp     r7, 2
        beq     inverse_subtract
        neg     r3, r3
inverse_subtract:
        sub     r1, r1, r3
        b       inverse_loop
inverse_end:
        pop     {r4, r5, r6, r7, pc}
```

b. Sine:

```
        push    {r4, r5, r6, r7, lr}
        mov     r4, r0
        mov     r5, r4
        mov     r7, 1
        mov     r6, r1
        ldr     r3, [r6], 4
        mov     r1, r4
sine_loop:
        cmp     r7, r2
        bge     sine_end
        mov     r2, r7
        bl      Polynomial
        subs    r7, r7, 2
        cmp     r7, 1
        bne     sine_subtract
        add     r3, r3, 1
sine_subtract:
        cmp     r7, 1
        beq     sine_add
        cmp     r7, 0
        beq     sine_add
```

```
        neg         r3, r3
sine_add:
        div         r3, r3, r7
        sub         r1, r1, r3
        b           sine_loop
sine_end:
        pop         {r4, r5, r6, r7, pc}
```

c. Exponential:

```
        push    {r4, r5, r6, r7, lr}
        mov     r4, r0
        mov     r7, #0
        mov     r6, r1
        ldr     r3, [r6], #4
        mov     r1, #0

exponential_loop:
        cmp     r7, r2
        bge     exponential_end
        mov     r2, r7
        bl      Polynomial
        cmp     r7, #0
        beq     exponential_add
        div     r3, r3, r7

exponential_add:
        add     r1, r1, r3
        add     r7, r7, #1
        b       exponential_loop

exponential_end:
        pop     {r4, r5, r6, r7, pc}
```

6.

Mean:

```
push        {r4, r5, r6, r7, lr}
mov         r4, r0
ldr         r5, [r4, 0]
mov         r6, 1
mov         r7, 0
vldr        s16, [r5]
```

```
mean_loop:
        cmp         r6, r1
        bge         mean_end
        add         r5, r5, 4
        vldr        s17, [r5]
        vadd.f32    s16, s16, s17
        add         r6, r6, 1
        b           mean_loop
mean_end:
        vdiv.f32    s16, s16, r1
        vmov        r0, s16
        pop         {r4, r5, r6, r7, pc}
```

7.

```
Variance:
        push        {r4, r5, r6, r7, lr}
        mov         r4, r0
        ldr         r5, [r4, 0]
        mov         r6, 1
        vmov        s17, r2
        mov         r7, 0
        vldr        s18, [r5]
variance_loop:
        cmp         r6, r1
        bge         variance_end
        add         r5, r5, 4
        vldr        s16, [r5]
        vsub.f32    s16, s16, s17
        vmul.f32    s16, s16, s16
        vadd.f32    s18, s18, s16
        add         r6, r6, 1
        b           variance_loop
variance_end:
        vdiv.f32    s18, s18, r1
        vmov        r0, s18
        pop         {r4, r5, r6, r7, pc}
```