Chapter11

1.

```
int32_t SIMD_Find4Mins(int16_t a[], int32_t n)
{
	int min = a[0];
	int temp;
	int i;
	for(i = 1; i < n; i ++)
	{
		temp = a[i];
		if(temp < min)
			min = temp;
	}
	return min;
}
```

```
SIMD_Find4Mins: // R0 = &a[0], R1 = n
	LDR		R2, [R0], 2	// a 32-bit register can fit in 2 16-bit integers
loop:	SUBS		R1, R1, 2
	BEQ		done
	LDR		R3, [R0], 2
	SSUB16		R12, R3, R2
	SEL		R2, R2, R3
	B		loop
done:	MOV		R0, R2
	BX		LR
```

3.

```
void Slow_USatAdd(uint8_t bytes[], uint32_t count, uint8_t amount)
Slow_USatAdd: // R0 = &bytes[0], R1 = count, R2 = amount
		BFI	R2, R2, 8, 8
		BFI	R2, R2, 16, 16
loop:		CBZ	R1, done
		LDR	R3, [R0]
		ADD	R3, R3, R2
		MSR	R12, APSR	// Copy flags to R12
		TST	R12, 1<<27	// Check value of Q
		BNE	QFlagSet	// Branch if Q = 1
		STR	R3, [R0], 4
		SUB	R1, R1, 4
		B	loop
QFlagSet:	RSB	R3, R2, 255	// R3 = 255 - R2
		B	loop
done:		BX	LR
```

Chapter12
4.

```
/*
typedef double COMPLEX;
COMPLEX a;
COMPLEX b;
D0 = a, S1.S0 = a, S0 = REAL(a), S1 = IMAG(a)
D1 = b, S3.S2 = b, S2 = REAL(b), S3 = IMAG(b)
*/
CAdd:  VADD.F32     S0, S0, S2
       VADD.F32     S1, S1, S3
       BX      LR

CSub:  VSUB.F32     S0, S0, S2
       VSUB.F32     S1, S1, S3
       BX      LR

CMul:  VPUSH        {S4}
       // real = REAL(a)*REAL(b) - IMAG(a)*IMAG(b)
       VMUL.F32     S12, S0, S2            // S12 = REAL(a)*REAL(b)
       VMLS.F32     S12, S1, S3, S12       // S12 = S12 - IMAG(a)*IMAG(b) = real

       // imag = REAL(a)*IMAG(b) + REAL(b)*IMAG(a)
       VMUL.F32     S4, S0, S3             // S4 = REAL(a)*IMAG(b)
       VMLA.F32     S4, S1, S2, S4         // S4 = S4 + REAL(b)*IMAG(a) = imag

       VMOV         S0, S12
       VMOV         S1, S4
       VPOP         {S4}
       BX           LR

CDiv:  VPUSH        {S4}
       // real = REAL(a)*REAL(b) + IMAG(a)*IMAG(b)
       VMUL.F32     S12, S0, S2
       VMLA.F32     S12, S1, S3, S12

       // imag = REAL(a)*IMAG(b) - REAL(b)*IMAG(a)
       VMUL.F32     S4, S0, S3
       VMLS.F32     S4, S1, S2, S4

       // dvsr = REAL(b)*REAL(b) + IMAG(b)*IMAG(b)
       VMUL.F32     S2, S2, S2
       VMLA.F32     S2, S3, S3, S2
```

```
VDIV.F32    S0, S12, S2         // real /= dvsr
VDIV.F32    S1, S4, S2          // imag /= dvsr

VPOP        {S4}
BX          LR
```