(1) Calculate the delay produced by the subroutine in terms of number of execution states.

Ans: Tallying up the total number of cycles will give (1 + 1×10 + 1×10 +1) = 22 cycles.

(2) Given that the processor clock is 10MHz, what is the duration of the time delay produced by the routine?

Ans: At 10MHz, each cycle is 0.1µs, therefore total delay = 22*0.1 = **2.2µs**

(3) How can a time delay of 1ms be achieved?

Ans: we change `MOV R0,#10` to `MOV R0,#0x87, then ADD R0,R0,#0x1300`.

3.4 Multi-precision Arithmetic

(1) With reference to Fig. 3.4, for each "?", give the ARM mnemonic that will implement the corresponding functionality described by each of the comments shown.

Suggested solutions:

```
START  MOV          SP, #0xFFFFFFFC
                    LDR          R0,=N1
                    LDR          R1,[R0]
                    ADD          R0,R0,#4
                    STMFD        SP!,{R1,R0}
                    BL           SubA
                    END

SubA                STMFD  SP!,{R4-R7}
                    ADD          SP,SP,#-12
                    LDR          R4,[SP,#32]
                    STR          R4,[SP]
                    LDR          R4,[SP,#28]
                    MOV          R5,#0
                    STR          R5,[SP,#4]
                    STR          R5,[SP,#8]
Loop                LDR          R5,[SP,#4]
                    LDR          R6,[R4],#4
                    LDR          R7,[R4],#4
                    ADDS         R6,R5,R6
                    LDR          R5,[SP,#8]
                    ADC          R7,R7,R5
                    STR          R6,[SP,#4]
                    STR          R7,[SP,#8]
                    LDR          R7,[SP]
                    SUBS         R7,R7,#1
                    STR          R7,[SP]
                    BNE          Loop
                    LDR          R5,[SP,#4]
                    LDR          R6,[SP,#8]
                    STR          R5,[R4],#4
                    STR          R6,[R4]
                    ADD          SP,SP,#12
                    LDMFD        SP!,{R4-R7}
                    MOV          PC,LR
```

(2) Describe what changes to the program in Fig. 3.4 you would make if the multi-precision integers are stored using the Big Endian format instead.

This can be done by changing lines 22-29 as follows:

```
LDR  R5,[SP,#8]
LDR  R6,[R4,#4]
LDR  R7,[R4]
ADDS R6,R6,R5
LDR  R5,[SP,#4]
ADC  R7,R7,R5
STR  R6,[SP,#8]
STR  R7,[SP,#4]
```

(3) Give the two 32-bit hexadecimal values in memory addresses `0x114` and `0x118` at the end of the execution of the code segment shown in Fig 3.4.

Ans: `0x114=0x223` (lower word) and

`0x118=0x812` (upper word)