(5) Question 1. Assume an 8-bit unsigned binary fixed-point format with a resolution $\Delta = 2^{-4}$ (1/16). If the integer part of a number is $B4, what is the corresponding value of this fixed-point number?

First, we convert $B4 to integer $B4=11*16+4 (but multiplying out is unnecessary)

Next, we use the formula value=integer*Δ = (11*16+4)/16 = 11.25

(5) Question 2. What will be the value of the carry (C) bit after executing the following?

```
ldab #210
subb #60
```

210 - 60 = 150, which does fit into an 8-bit unsigned number, so `C=0`

(5) Question 3. What will be the value of the overflow (V) bit after executing the following?

```
ldaa #-70
adda #-60
```

-70 + -60 = -130, which does not fit into an 8-bit signed number, so `V=1`

(5) Question 4. A software variable can take on the following specific values 0.0, 0.1, 0.2 ..., 99.8, 99.9, 100.0. Which fixed-point format should be used for this variable? If more than one format could be used to solve the problem, choose the most space-efficient format. Enter the correct letter A-H.

A) 8-bit signed fixed-point, Δ = 0.01
B) 8-bit signed fixed-point, Δ = 0.1
C) 16-bit signed fixed-point, Δ = 0.01
D) 16-bit unsigned fixed-point, Δ = 0.1
E) 24-bit unsigned fixed-point, Δ = 0.01
F) 8-bit unsigned fixed-point, Δ = 0.1
G) 32-bit floating point
H) binary fixed-point must be used

The resolution is clearly 0.1, and the range of integers is 0 to 1000, so D

(5) Question 5. Consider the result of executing the following three 9S12 assembly instructions.

```
ldx #12
ldd #20
idiv
```

Part a) What is the decimal value in Register X after these three instructions are executed?

Reg X is the quotient of 20/12 = 1

Part b) What is the decimal value in Register D after these three instructions are executed?

Reg D is the remainder of 20%12 = 8

(5) Question 6. What is the instruction corresponding to the following machine code?

```
$9371
```

This is `subd $0071` direct mode addressing

(5) Question 7. You are asked to measure a parameter to 4 decimal digits. What is the minimum number of ADC binary bits that will be needed?

10^4 = 10000 and 2^{14}=16384, so we will need at least 14 bits for the ADC.

(5) Question 8. Consider the following piece of code that calls the subroutine, `SCI_OutString`

```
$4029 CE40C4        ldx  #Err
$402C 16417E        jsr  SCI_OutString
$402F 20EA          bra  loop
```

During the execution of the `jsr` instruction, what number is pushed on the stack?

The `jsr` instruction pushes the return address on the stack, which is `$402F`

(10) Question 9. Assume RegX is $3800, RegD is $4647, the PC is $4123, and Ram locations $3800 to $38FF are initially $00, $01,…$FF respectively. E.g., location $3856 contains $56. Show the simplified bus cycles occurring when the **`subd 2,x`** instruction is executed. In the "**changes**" column, specify which registers get modified during that cycle, and the corresponding new values. Do not worry about changes to the CCR. *Just show the one instruction.*

```
$4123 A302            subd 2,x
```

R/W	Addr	Data	Changes to A,B,X,Y,S,PC,IR,EAR
R	$4123	$A3	PC=$4124, IR=$A3
R	$4124	$02	PC=$4125, EAR=$3802
R	$3802	$02	
R	$3803	$03	D=$4647-$0203=$4444

(25) Question 10. There is 25-element 8-bit unsigned array, called **`Buffer`**. Write a subroutine **`Add1`** that adds one to each element in the array. To get full credit, use indexed addressing mode and a loop. Don't worry about initializing the array. Don't worry about establishing the reset vector, creating a main program, calling the subroutine or initializing the stack. Comments are not required.

```
       org $3800
Buffer rmb 25    ;unsigned 8-bit integers
       org $4000
```

```
Add1
     ldx #Buffer
loop inc 1,x+
     cpx #Buffer+25
     bne loop
     rts
```

```
Add1
     ldaa #25
     ldx #Buffer
loop inc 1,x+
     dbne A,loop
     rts
```

```
Add1
     ldaa #25
     ldx #Buffer
loop ldab 0,x
     incb
     stab 1,x+
     deca
     bne  loop
     rts
```

(25) Question 11. There are two 16-bit unsigned variables, called **`Input`** and **`Output`**. Write assembly code that checks the **`Input`**, and if **`Input`** is less than 100, then the code sets the **`Output`** to 40. Conversely if **`Input`** is greater than or equal to 100, then the code does not modify **`Output`**. Don't worry about initializing the variables. Don't worry about establishing the reset vector, creating a main program, or initializing the stack. Comments are not required.

```
       org $3800
Input  rmb 2    ;unsigned 16-bit integer
Output rmb 2    ;unsigned 16-bit integer
       org $4000
      ldd  Input
      cpd  #100
      bhs  skip
      movw #40,Output
skip
```