Final Exam Part 2

Submission: Work on paper/CCS submit answers to Carmen

**Due: Wednesday December 13 – at the End of the Day (11:59 PM)
No Late Submissions**

**This exam is individual work.
You are not allowed to collaborate with others including AI.**

Question 1 – A simple subroutine (6 + 5 pts)

You are asked to write a subroutine (let's call it cfg) with following contract:

```
;------------------------------------------------------------------
; Subroutine: cfg
; input:  R10 = strictly positive integer n > 0
; output: R12 = sum of all positive integers less than and equal to n
;
; Free to modify R10 and R12 but no other core register
;------------------------------------------------------------------
```

(a) Write the complete subroutine. Streamline your code as much as possible. **As an exception, please skip the comments this time to make grading in Carmen easier.**

(b) What is the **largest value of n** that this subroutine can handle **correctly** (i.e., return a correct output)?

Question 2 – Status Flags (1 + 1 + 1 + 4 pts)

Can you predict the outcome of the snippet of code given below?

```
        mov.b   #71, R4
        add.b   #73, R4
```

In particular, after executing these two lines of code, what will be the status flags?
(a) V=?
(b) N=?
(c) C=?
(d) The more important question is: Why do we get these values for the status flags? Give a short but precise explanation.

Question 3 – Things that can go wrong (1 + 5 + 5 pts)
For Quiz 5, you were asked to write a subroutine that finds the remainder after integer division. Below is a subroutine that attempts to do that.

```
281 ;------------------------------------------------------------
282 ; Subroutine: mod
283 ; input: R5 unsigned 16-bit integer x
284 ;        R8 unsigned 16-bit nonzero integer N
285 ;
286 ; output: R5 unsigned 16-bit integer y
287 ;           y = x % N is a number 0 <= y < N s.t. x=y mod N
288 ;           y is the remainder when x is divided by N
289 ;------------------------------------------------------------
290 mod:
291         cmp.w   R8, R5              ; set status bits for (x-N)
292         jn      end_mod             ; if (x-N) < 0 we are done
293                                     ; otherwise
294         sub.w   R8, R5              ; x <- x-N
295         jmp     mod                 ; and repeat
296 end_mod:
297         ret
```

Sometimes it works correctly (e.g., it can correctly compute 1024 % 5 = 4), and at other times it does not. **Can you find the error and fix the subroutine?**

(a)   What line needs to be fixed?
(b)   What should the correct line be?
(c)   What is one input for which the subroutine does not give a correct result? There are many such pairs x and N, you need to give only one example.

Question 4 – Things that can go wrong 2 (8 pts)
We have seen that multiplication by a power of two is easy with binary numerals: To multiply a number by 2^p corresponds to p left shifts. This works both for signed and unsigned numbers as long as we are careful with the range.

Below is an attempt on a subroutine to multiply by a positive power of two. As noted in the contract, the assumption is that p is a strictly positive integer (i.e., 1, 2, 3 etc.).

```
;----------------------------------------------------------------
; Subroutine: times_power_of_two
; input:  R5 16-bit strictly positive number p -- returned unchanged
;         R6 16-bit number x
;
; output: R6 16-bit number y = x*2^p
;            y is obtained by p left-shifts
;----------------------------------------------------------------
times_power_of_2:

            push    R5

            rla.w   R6
            dec.w   R5
            jne     times_power_of_2

            pop     R5
            ret
```

What happens when this subroutine is called with R5 = 3 and R6 = 11? **Give a short but precise answer at the register level of the MCU.**

Question 5 – Things that can go wrong 3 (8 pts)

In class, we have discussed the ultimate ISR as an homage to Claude Shannon's ultimate machine: The ultimate ISR does nothing except clearing the interrupt flag.

Below is an ultimate ISR that serves push button S1. You can assume that the IVT has been correctly populated.

```
81 ultimate_ISR_for_S1:
82
83             bit.b   #BIT1, &P1IFG
84             jnc     fin
85
86             bic.b   #BIT1, &P1IFG
87 fin:
88             ret
```

What will happen when you press push button S1? **Give a short but precise explanation at the register level of the MCU.**

Question 6 – Parting question (5 pts)
I like to ask this question in all of my classes: What is the *best thing* you have learned in this class? You get to decide what *best* means: it could be the most interesting, most important, most useful, or it could be the one thing you think you will still remember many years from today. There are no right or wrong answers, every opinion is valid.

When you are ready start to enter your answers to Carmen. You have only one attempt and one hour to enter all your answers once you start the online quiz.

<p style="text-align:center;color:red;">Thank you for a great semester.</p>

<p style="text-align:center;color:red;">Best of luck with all your finals!</p>

<p style="text-align:center;color:red;">Happy Holidays!</p>