HW2
CMPE364: Microprocessor Based Design
Spring 2017

Question 1

Book question 4.4

Note that there are multiple possible correct answers.

```
; r0 stores the value we want to take the ABS of
cmp r0, #0
bge done
rsb r0, r0, #0
done:
```

Question 2

Book question 4.8

In this problem it calculates r1-r4 and sets the status bits accordingly. That result is negative, is not zero, does result in a carry, and does not overflow. So the important four bits of the CPSR are 1010. Combining this with the other bits of the CPSR leads to…

```
0xA0000010
```

Question 3

Book question 5.2

After the instruction runs:
R1 = 0x00008A0C
R3 = 0xFFFFFFFE
R4 = 0xA9C25541

The memory map will be: (modified items in bold)

0x00008A04	0xE285
0x00008A06	0x5001
0x00008A08	0xFFFC
0x00008A0A	**0xA341**
0x00008A0C	0x4EA9
0x00008A0E	0xC786
0x00008A10	0xAF01
0x00008A12	0x5613

Question 4

Book question 5.8

There are a LOT of answers to this question. Here is a simple one:

```
mov r0, r0
```

Question 5

Book question 5.10

```
R6 =  0xFF014680
R13 = 0x00009CC0
R14 = 0xA804194E
```

There are no changes to memory.

Question 6

Write a simple ARM assembly language program that finds the mean (as an integer) of all values of an integer array. When your program starts, you should assume that `r0` contains the address of the array and `r1` contains the number of integers in the array. When you program finishes, the mean should be stored in `r2`. You may use other registers as scratch registers.

There are many possible correct solutions. Here is one:

```
mov r2, #0
mov r3, #0

sum_loop:
        ; Sum the array, put the result in r2

        ; termination condition
        cmp r1, r3
        beq sum_done

        ; Add the next array entry to our sum
        ldr r4, [r0], #4
        add r2, r2, r4

        ; increment our counter and go again
        add r3, r3, #1
        b sum_loop

sum_done:
        ; We finished summing, now we need to divide
        ; This is a simple loop to do a power of 2 division
        mov r1, r1, lsr #1
        mov r2, r2, lsr #1

        cmp r1, #1
        bne sum_done

swi 0x11
```