Homework 5: Answers

Profs. John McLeod & Arash Reyhani

ECE3375, Winter 2022

Please note that the particular homework assignment that you answered will have the same options as listed here, but the order of these options may have been randomized for each quiz. So your option (b) may instead be written here as option (c), etc. Please read the questions and options in full.
The answers to these questions are all found in the course notes, textbook, and video lessons.

1. Problem: Consider the following code.

```
/* main code here */
mov r4, #0
label_1:
/* more code */
add r4, #1
cmp r4, #10
bne label_1
```

Which of the following best describes this code?

(a). A block of code that is looped over 10 times before the program continues sequentially.

(b). A block of code that is looped over endlessly.

(c). A conditional branch causes the program to skip over a sequential block of code.

(d). The link register overflows, causing the stack to push untimed interrupts to the program status register.

Answer: Option (a). Here `r4` is used as a counter that is incremented up to $(10)_{10}$. Of course, if the `/*more code*/` part also messes with value of `r4` then this code may not execute ten times, but the question says "best describes" — without further information we should assume `r4` is not affected by any unspecified code.

2. *Problem*: Consider the following code.

```
/* main code here */
mov r4, #10
label_1:
  /* more code */
  subs r4, #1
  bne label_1
```

Which of the following best describes this code?

(a). A block of code is looped over 10 times before the program continues sequentially.

(b). A block of code is looped over 9 times before the program continues sequentially.

(c). There is no `cmp` instruction, so the loop is endless.

(d). A block of code is looped over until the stack overflows.

Answer: Option (a). The loop will exit when `r4` holds the value of 1. A `cmp` instruction is not necessary for to evaluate a condition — as long as there is code that updates the status flags (as `subs` does), a condition can be evaluated.

3. Problem: Consider the following code.

```
/* main code here */
mov r4, #10
label_1:
  /* more code */
  adds r4, #1
  bne label_1
```

Which of the following best describes this code?

(a). A block of code will be looped over 10 times before the program continues sequentially.

(b). A block of code will be looped over $(2^{32} - 10)$ times before the program continues sequentially.

(c). A block of code will be looped over endlessly.

(d). A block of code will be looped over (2^{32}) times before the program continues sequentially.

Answer: Option (b). The `ne` conditional flag is always evaluated the same way: it is true if the zero status flag is clear (`Z=0`). As `r4` started at $(10)_{10}$ and is 32-bits in size, it will overflow back to zero after adding $(2^{32} - 10)$ to `r4`.

4. *Problem*: Consider the following code.

```
    mov r0, #8
    mov r1, #8
part_a:
    cmp r0, r1
    beq part_c
part_b:
    adds r0, r1
    bmi part_d
    lsl r1, #2 @equivalent to r1 = r1 * 4
    b part_b
part_c:
    add r0, #2
    b part_a
part_d:
    /*rest of code*/
```

How many times does the program reach the label `part_b`? You may assume there are no further branches after `part_d`.

Answer: The program reaches `part_b` 15 times. The code jumps away to `part_d` when the value in `r0` would be a negative number in signed 2's complement, meaning a value $\geq 2^{32-1}$. The value in `r0` at each pass through label `label_b` is:

Cycle by `part_b`	Value in `r0`	Cycle by `part_b`	Value in `r0`
1	`0x0000 000A`	9	`0x0002 AAB2`
2	`0x0000 0012`	10	`0x000A AAB2`
3	`0x0000 0032`	11	`0x002A AAB2`
4	`0x0000 00B2`	12	`0x00AA AAB2`
5	`0x0000 02B2`	13	`0x02AA AAB2`
6	`0x0000 0AB2`	14	`0x0AAA AAB2`
7	`0x0000 2AB2`	15	`0x2AAA AAB2`
48	`0x0000 AAB2`		

After 15 loops, the addition is performed one last time, so `r0` has a value of `0x0xAAAA AAB2` when `bmi part_d` is finally executed.