Solutions 4

Q1:

a) Thumb-2 can freely intermix 16-bit and 32-bit instructions. This means is has the code density advantage of 16-bit instructions as well as the power of 32-bit instructions
b) As all instructions are 16-bit aligned, all instruction addresses are multiples of 2. Hence, the lsb of an instruction address is always 0. As the bit is always 0, we can assign a different use to it.

Q2:

a)
MOVS: 1
loop:
ADDS: 1
MOVS: 1
CMP: 1
BEQ: 1 (typically not taken)
B: 3

Loop iterations: FF/3 = 85
Cycles = 1 + 85 * (1 + 1 + 1 + 1 + 3) = 596

However, on the last iteration of the loop, the BEQ IS taken. That introduces two additional cycles, but removes the cost of the B. Net effect: -1.
Hence, total = 595

b) 595 / 8e6 = 0.000074375 seconds = 74.375 us

Q3:

a) 1 147 483 647 + 1 234 567 890
= 0x4465 35FF + 0x4996 02D2
= 0x8DFB 38D1
-1 912 915 759

For the following, be weary of subtraction operations. Subtractions are implemented as additions of the inverse, which introduces additional complexity to the flags.
b) The result must have the msb set. Also, it must exceed the maximum or minimum signed number, but not overflow when considered unsigned. Eg: 0x7FFF FFFF + 1
c) Must overflow both signed and unsigned. 0x8000 0000 + 0x8000 0000

d) No. For an instruction to set N, the msb must be 1. For an instruction to set 0, all bits must be 0.

e)	Z = 1,	N = 0,	C = unchanged,	V = unchanged
f)	Z = 1,	N = 0,	C = 1,	V = 0
g)	Z = 1,	N = 0,	C = 1,	V = 0
h)	Z = 0,	N = 1,	C = 0,	V = 0

Q4:

a) Attempts to load a word on an unaligned address.

b) First loads data at label foobar. Second loads address of data labeled foobar.

Q5

```
CMP R0, R1
BHS foo        // if R0 is higher than or equal to R1. (BCS also works)
B bar
```

or

```
CMP R1, R0
BLO foo        // if R1 is lower than R0
B bar
```