1. Using some test cases, match these bit operations to their associated function:

1. `x & 1`	a) Return x without trailing 1s (e.g. 11011111 becomes 11000000)
2. `x & (1 << n)`	b) Unset the n_{th} bit
3. `x & ~(1 << n)`	c) Return true if n_{th} bit is set
4. `(x ^ y) < 0`	d) Return the minimum of x and y
5. `y ^ ((x ^ y) & -(x < y))`	e) Return true if x and y have opposite signs
6. `x & (x - 1)`	f) Return true if x is odd, false if x is even
7. `x & (x + 1)`	g) Return 0 if x is a power of 2 for $x > 0$

Solution:

2. The following C "optimizations" are said to improve the performance of embedded systems. In reality, some of them are useless or even counterproductive on certain architectures. For each of the "optimizations" given,

   - Find out why it optimizes performance on some architectures
   - Find out if there are any targets on which it does not improve performance, or decreases performance
   - On the architectures on which it improves performance, how great is the improvement? (e.g., one instruction overall, one instruction per iteration of a loop, etc.) Is the improvement significant or trivial?

   Here are the "optimizations":

   (a) Count down to zero, not up to N, in `for()` loops

   (b) Avoid the % operation

   (c) Use an 8-bit `unsigned char` whenever you have a value that you know won't go beyond 0-255 (e.g., some loop index variables)

Solution:

3. Refer to the JPL Institutional Coding Standard for the C Programming Language (http://lars-lab.jpl.nasa.gov/JPL_Coding_Standard_ext.pdf). This standard describes their rules for mission critical flight software written in the C programming language. (The NASA Jet Propulsion Laboratory was responsible for the Mars Curiosity rover.)
   (a) Why is recursion not permitted in mission critical flight software?
   (b) Why is dynamic memory allocation disallowed after task initialization in mission critical flight software?

Solution:

4. Fill in the blanks with the word “signed” or “unsigned”:

(a) In ________________ arithmetic, if the overflow flag (V in CPSR) is set on an operation, the result is wrong.

(b) In ________________ arithmetic, the overflow flag (V in CPSR) does not indicate anything meaningful about the result of the operation.

(c) In ________________ arithmetic, if the carry flag (C in CPSR) is set on an operation, the result is wrong.

(d) In ________________ arithmetic, the carry flag (C in CPSR) does not indicate anything meaningful about the result of the operation.

Solution:

5. Describe the status of the N, Z, C, and V flags of the CPSR after each of the following:

(a)
```
ldr     r1, =0xffffffff
ldr     r2, =0x00000001
add     r0, r1, r2
```

(b)
```
ldr     r1, =0xffffffff
ldr     r2, =0x00000001
cmn     r1, r2
```

(c)
```
ldr     r1, =0xffffffff
ldr     r2, =0x00000001
adds    r0, r1, r2
```

(d)
```
ldr     r1, =0xffffffff
ldr     r2, =0x00000001
addeq   r0, r1, r2
```

(e)
```
ldr     r1, =0x7fffffff
ldr     r2, =0x7fffffff
adds    r0, r1, r2
```

Solution:

6. The following C code implements the Euclid algorithm for calculating the greatest common divisor:

```
int gcd(int a, int b)
{
    while (a != b)
      {
        if (a > b)
            a = a - b;
        else
            b = b - a;
      }
    return a;
}
```

Here is an equivalent ARM assembly routine that only uses conditional execution on the branch instructions:

```
gcd
        CMP        r1, r2
        BEQ        end
        BLT        lessthan
        SUB        r1, r1, r2
        B          gcd
lessthan
        SUB        r2, r2, r1
```

```
        B           gcd
end
        ...
```

And here is an equivalent ARM assembly routine that uses full conditional execution :

```
gcd
        CMP          r1, r2
        SUBGT        r1, r1, r2
        SUBLT        r2, r2, r1
        BNE          gcd
```

Assume `a` is 54 and is loaded into `r1`, `b` is 24 and is loaded into `r2`.

(a) Run through the C algorithm until its completion to find the greatest common divisor.

Solution:

(b) Run through the ARM assembly version without full conditional execution.

(c) Run through the ARM assembly version with full conditional execution.

(d) Refer to the ARM Cortex-M4 Technical Reference Manual (available online) to find out the timing of each instruction. How many cycles does the first ARM routine take? How many cycles does the second ARM routine take?

Solution: