These questions should be done by hand calculations. You can use some types of hand-held or computer calculators to check your answers. You can also check your answers using WolframAlpha on the web (http://www.wolframalpha.com/) by entering, for example, 101110_2 *as* `101110_2` *(using the underscore character), and WolframAlpha will respond with decimal, hexadecimal and other equivalents. You can also do modular arithmetic and conversions between bases (e.g. "*`convert octal 35 to hexadecimal`*") etc. there.*

1. Write the following numbers in base 10.

(a) 101110_2 $2+4+8+32 = \mathbf{46.}$

(b) $12C_{16}$ $\mathbf{C} + 2\times 16 + 1\times 16^2 = 12+32+256 = \mathbf{300.}$

(c) 127_8 $7 + 2\times 8 + 1\times 8^2 = \mathbf{87.}$

2. Compute the following directly (without any change of base):

(a) $1011_2 + 1010_2$

(b) $1011_2 \times 1100_2$

(c) $A2_{16} + B3_{16}$

(d) $1A_{16} \times 23_{16}$

(a)
$$\begin{array}{r} \mathbf{1011} \\ \mathbf{1010} \\ \hline \mathbf{10101} \\ \mathbf{1\ \ 1} \end{array}$$

(b)
$$\begin{array}{r} \mathbf{1011} \\ \mathbf{1100} \\ \hline \mathbf{101100} \\ \mathbf{1011000} \\ \hline \mathbf{10000100} \\ \mathbf{1111} \end{array}$$

(c)
$$\begin{array}{r} \mathbf{A2} \\ \mathbf{B3} \\ \hline \mathbf{155} \\ \mathbf{1} \end{array}$$

(d)
$$\begin{array}{r} \mathbf{1A} \\ \mathbf{23} \\ \hline \mathbf{4E} \\ \mathbf{340} \\ \hline \mathbf{38E} \end{array}$$

3. Write the following in base 2.

(a) 151_{10} $128+16+4+2+1 = 2^7+2^4+2^2+2^1+2^0 = \mathbf{10010111_2}$

(b) BAD_{16} $1011_2\ 1010_2\ 1101_2 = \mathbf{101110101101_2}$

(c) 747_8 $111_2\ 100_2\ 111_2 = \mathbf{111100111_2}$

4. Recall that to represent negative integers in computer words a special convention is used. For 8-bit words, any negative integer x in the range $-2^7 = -128 \le x \le -1$, can be represented. To represent say $x = -25$:

1) write $|x|$ in 8-bit binary (pad with leading zeroes if necessary): 00011001;
2) toggle all the bits ($0 \to 1$, $1 \to 0$): 11100110;
3) add 1 to the resulting binary number: 11100111.

Provided x is in the stated range, the representation will automatically have 1 as the left-most bit. So if the left-most bit of an 8-bit word is 0 the word is interpreted as a binary representation of an integer x in the range $0 \le x \le 127 = 2^7 - 1$.

Using this representation:

(a) Write -123_{10} as an 8-bit word. $123 = 64{+}32{+}16{+}8{+}2{+}1 = 1111011_2 = 01111011_2$
So, using toggle-and-add-one, -123 is represented by $10000100{+}1 = \boxed{\mathbf{10000101}}$

(b) Write -128_{10} as an 8-bit word. $128 = 2^7 = 10000000_2$
So, using toggle-and-add-one, -128 is represented by $01111111{+}1 = \boxed{\mathbf{10000000}}$

(c) Evaluate `11001100` **The leading digit is a 1, so number is negative.**
(i.e. express it in ordinary decimal notation) **Again using toggle-and-add-one:**
$11001100 \longrightarrow -(00110011_2 + 1) = -00110100_2 = -(32+16+4) = \mathbf{-52_{10}}$

(d) Evaluate `01111000` **The leading digit is a 0, so number is positive.**
(i.e. express it in ordinary decimal notation) $01111000_2 = 64{+}32{+}16{+}8 = \mathbf{120_{10}}$

5. Write the following numbers (from $\mathbb{Q}$, in base 10) as 8-bit words, with sign bit s followed by 5 bits of mantissa ("5 bit precision") m followed 2 bits of (non-negative) exponent n. For example -6.5 would be written `11101010`.

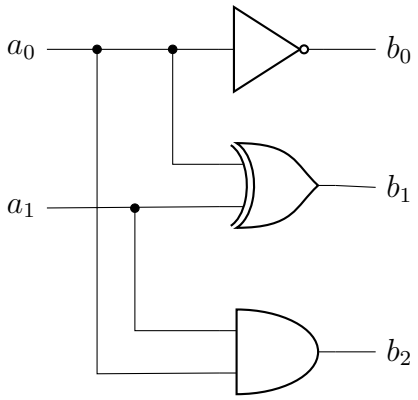
(a) 5.25 $5\frac{1}{4} = 2^2+2^0+2^{-2} = 101.01_2 = 1.0101_2 \times 2^2 = 1.0101_2 \times 2^{10_2} \rightarrow$ **01010110**

(b) -2 $= -1 \times 2 = -1.0000_2 \times 2^1 \rightarrow$ **11000001**

(c) 5.3 (requires approximating) $0.3 = \frac{3}{10} = \frac{1}{4} + (\frac{3}{10} - \frac{1}{4}) = \frac{1}{4} + \frac{1}{20}$
$= \frac{1}{4} + \frac{1}{32} + (\frac{1}{20} - \frac{1}{32}) = \frac{1}{4} + \frac{1}{32} + \frac{3}{160}$
So $5.3 = 101.01001....._2 = 1.0101001....._2 \times 2^2 \approx 1.0101 \times 2^2$ in 5-bit precision.
Thus we get the same answer as for (a): 01010110

(d) -15.5 $-15\frac{1}{2} = -1111.1_2 = -1.1111 \times 2^3 = -1.1111 \times 2^{11_2} \rightarrow$ **11111111**

6. From the top down the circuit at right has a NOT gate, an XOR gate and an AND gate. The input is the integer x represented in binary as $(a_1a_0)_2$ and the output is the integer $y = f(x)$ represented in binary by $(b_2b_1b_0)_2$. Find y for each $x \in \{0,1,2,3\}$ and hence determine the purpose of the circuit by finding the rule for the function $f : \{0,\ldots,3\} \rightarrow \{0,\ldots,7\}$.



x	a_1	a_0	b_2	b_1	b_0	y
0	0	0	0	0	1	1
1	0	1	0	1	0	2
2	1	0	0	1	1	3
3	1	1	1	0	0	4

So $y = f(x) = x + 1$ and hence the purpose of the circuit is to ADD ONE.

7. Carry out the following binary arithmetic operations. Do not change base.

(a)
$$\begin{array}{r} 00001101 \\ +00000110 \\ \hline \mathbf{00010011} \\ {\scriptstyle 1} \end{array}$$

(b)
$$\begin{array}{r} 0000{\scriptstyle 1}1{\scriptstyle 1}01 \\ -00000{\scriptstyle 1}{\scriptstyle 1}10 \\ \hline \mathbf{00000111} \end{array}$$

(c)
$$\begin{array}{r} 00001101 \\ \times 00000110 \\ \hline 11010 \\ 110100 \\ \hline \mathbf{01001110} \\ {\scriptstyle 1} \end{array}$$

8. Calculate the following.

(a) 128 div 5 = **25** $\because 128 = 25\times5 + 3$

(b) 128 mod 5 = **3** $\because 128 = 25\times5 + 3$

(c) 12365489 mod 2 = **1** $\because$ 12365489 is odd

(d) -20 mod 3 = **1** $\because -20 = (-7)\times3 + 1$

9. Write an algorithm that takes $n \in \mathbb{N}$ as input, and returns "true" as an output if n is prime, and "false" otherwise. Make use of the mod operation.

Input: natural number n
Output: "true" if n is prime, "false" otherwise
Method:
If $n = 1$, stop and return "false"
Set $j = 2$
Loop: If $j^2 > n$, stop and return "true".
If n mod $j = 0$, stop and return "false"
Replace j by $j + 1$
Repeat loop.

10. Determine the truth or falsity of the following statements. Give reasons.

(a) $765 \equiv 654 \pmod{37}$
True
$\because\ 765-654=111$
$= 3\times 37$

(b) $765 \equiv -654 \pmod{37}$
False
$\because\ 765--654=1419$
$= 38\times 37+13$

(c) $-765 \equiv -654 \pmod{37}$
True
$\because\ -765--654=-111$
$= -3\times 37$

11. Calculate each of the following, using techniques of modular arithmetic.

(a) $13579 \times 24680 \bmod 9$
In base 10, every natural number is congruent modulo 9 to the sum of its digits. So
$13579 \equiv (1+3+5+7+9) = 25 \equiv 2+5 = 7 \pmod 9$ and
$24680 \equiv (2+4+6+8+0) = 20 \equiv 2+0 = 2 \pmod 9$.

Hence $13579 \times 24680 \equiv 7 \times 2 = 14 \equiv (1+4) = 5 \pmod 9$.
Since $0 \le 2 < 9$ it follows that $\mathbf{13579 \times 24680 \bmod 9 = 5}$.

(b) $13579 \times 24680 \bmod 11$
In base 10, every natural number is congruent modulo 11 to the sum of every second of its digits (starting from the right) minus the sum of the other digits. So
$13579 \equiv (9+5+1)-(7+3) = 15-10 = 5 \pmod{11}$ and
$24680 \equiv (0+6+2)-(8+4) = 8-12 = -4 \pmod{11}$.

Hence $13579 \times 24680 \equiv 5 \times -4 = -20 \equiv -(0-2) = 2 \pmod{11}$.
Since $0 \le 2 < 11$ it follows that $\mathbf{13579 \times 24680 \bmod 11 = 2}$.

(c) $13579^8 \bmod 13$
In base 10 there is no convenient rule for congruence modulo 13, so we first need to find 13579 mod 13 by division-plus remainder. We find that $13579 = 1044\times 13+7$, and so $13579 \bmod 13 = 7$. It now follows that

$13579^8 \equiv 7^8 = (7^2)^4 = 49^4 \equiv (-3)^4 = 9^2 \equiv (-4)^2 = 16 \equiv 3 \pmod{13}$.
Since $0 \le 3 < 13$ it follows that $\mathbf{13579^8 \bmod 13 = 3}$.

(d) $13579^{24680} \bmod 11$. (Use Fermat's little theorem.)
Since 11 is prime Fermat's little theorem tells us that $a^{10} \equiv 1 \pmod{11}$ whenever $a \not\equiv 0 \pmod{11}$. From (b) we know that $13579 \equiv 5 \not\equiv 0 \pmod{11}$, so

$13579^{24680} \equiv 5^{24680} = (5^{10})^{2468} \equiv 1^{2468} = 1 \pmod{11}$.
Since $0 \le 1 < 11$ it follows that $\mathbf{13579^{24680} \bmod 11 = 1}$.