1. What is the decimal equivalent of the largest binary integer that can be obtained with (a) 9 bits and (b) 18 bits?

(a) $2^9 - 1 = 511$. (b) $2^{18} - 1 = 262,143.$

2. Convert the following binary numbers to decimal: (a) 1100110, (b) 11010111.1101, and (c) 111001001.

(a) $1100110 = 2^6 + 2^5 + 2^2 + 2^1 = 102.$
(b) $11010111.1101 = 2^7 + 2^6 + 2^4 + 2^3 + 2^1 + 2^0 + 2^{-1} + 2^{-2} + 2^{-4} = 215.8125.$
(c) $111001001 = 2^8 + 2^7 + 2^6 + 2^3 + 2^0 = 457.$

3. Convert the following decimal numbers to hexadecimal: (a) 731, (b) 1776, and (c) 1865.

(a) $731/2 = 365$ with remainder 1,
   $365/2 = 182$ with remainder 1,
   $182/2 = 91$ with remainder 0,
   $91/2 = 45$ with remainder 1,
   $45/2 = 22$ with remainder 1,
   $22/2 = 11$ with remainder 0,
   $11/2 = 5$ with remainder 1,
   $5/2 = 2$ with remainder 1,
   $2/2 = 1$ with remainder 0,
   $1/2 = 0$ with remainder 1,
So $731_{10} = 101101011_2 = 2db_{16}$.

(b) $1776/2 = 888$ with remainder 0,
   $888/2 = 444$ with remainder 0,
   $444/2 = 222$ with remainder 0,
   $222/2 = 111$ with remainder 0,
   $111/2 = 55$ with remainder 1,
   $55/2 = 27$ with remainder 1,
   $27/2 = 13$ with remainder 1,
   $13/2 = 6$ with remainder 1,
   $6/2 = 3$ with remainder 0,
   $3/2 = 1$ with remainder 1,
   $1/2 = 0$ with remainder 1,
So $1776_{10} = 11011100000_2 = 7f0_{16}$.

(c) $1865/2 = 932$ with remainder 1,
   $932/2 = 466$ with remainder 0,
466/2 = 233 with remainder 0,
233/2 = 116 with remainder 1,
116/2 =  58 with remainder 0,
 58/2 =  29 with remainder 0,
 29/2 =  14 with remainder 1,
 14/2 =   7 with remainder 0,
  7/2 =   3 with remainder 1,
  3/2 =   1 with remainder 1,
  1/2 =   0 with remainder 1,
So 1865_{10} = 111 0100 1001_{2} = 749_{16}.

4. Add and subtract the following numbers without converting to decimal: (a) (373)_{8} and
(266)_{8} (b) (bc2)_{16} and (99)_{16} (c) (110111)_{2} and (101110)_{2}.

\[
\begin{array}{ccc}
\text{110} & \text{carry} & \text{100} & \text{carry} & \text{1111100} & \text{carry} \\
\text{373} & & \text{bc2} & & \text{110111} & \\
\text{+266} & & \text{+ 99} & & \text{+ 101110} & \\
\text{661}_{8} & & \text{c5b}_{16} & & \text{1100101}_{2} & \\
\text{010} & \text{borrow} & \text{110} & \text{borrow} & \text{1010000} & \text{borrow} \\
\text{373} & & \text{bc2} & & \text{110111} & \\
\text{−266} & & \text{− 99} & & \text{+ 101110} & \\
\text{105}_{8} & & \text{b29}_{16} & & \text{001001}_{2} & \\
\end{array}
\]

5. Multiply the following numbers without converting to decimal. (a) (23)_{8} and (54)_{8} (b) (11011)_{2} and (1011)_{2}.

\[
\begin{array}{ccc}
\text{23} & \times \text{54} & \text{11011} \\
\text{114} & & \text{11011} \\
\text{1370} & & \text{+ 110110} \\
\text{1504}_{8} & & \text{1010001} \\
& & \text{+11011000} \\
& & \text{100101001}_{2} \\
\end{array}
\]

6. A computer represents information in groups of 40 bits. How many different integers can
be represented in (a) binary, (b) BCD, and (c) 8-bit ASCII, using all 40 bits in a word?

(a) \(2^{40} \approx 10^{12}\) (b) \(10^{40/4} = 10^{10}\) (c) \(10^{40/8} = 10^{5}\).

7. Express the −22, 135, −67 and −98 as 8 bit two's-complement numbers.

135 has no 8 bit two's-complement representation, since we can only represent positive numbers up
to 127 in this format. The two's-complements for the other values are 1+(0001 0110)′ = 1110 1010,
1+(0100 0011)′ = 1011 1101, 1+(0110 0010)′ = 1001 1110.

8. What pattern of bits is used to represent the integer 317 (a) in binary notation (b) in BCD
notation (c) in ASCII notation?
9. Write a machine-language program similar to the one on page 1-21 of the course notes. Your program should add the values stored in a sequential block of memory locations. When your program begins execution, the address of the first memory location is stored in word 1. The number of values to be added is stored in word 2. When your program completes, the sum should be written in word 3. Place the first instruction of your program at location 10. You may use words 4-9 to store any program variables you require. Start your program text at word 000a. Use the “hexadecimal” version of the machine language, as on page 1-21. Write your program text in the same style as the program on page 1-21. That is, show not just the actual machine language instruction, but the location where it is stored, the mnemonic version of the instruction and comments that provide a higher level pseudo-code for each group of instructions.

What is the sum computed, if when the program starts, 000b is stored in word 1 and 0004 is stored in word 2?

You may assume that the value stored in word 2 is non-negative. How would you tell if it were negative?

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td></td>
<td>base = start of array</td>
</tr>
<tr>
<td>0002</td>
<td></td>
<td>n = number of values to add</td>
</tr>
<tr>
<td>0003</td>
<td></td>
<td>sum = result</td>
</tr>
<tr>
<td>0004</td>
<td></td>
<td>i = number left to add</td>
</tr>
<tr>
<td>0005</td>
<td></td>
<td>p = pointer to next value</td>
</tr>
<tr>
<td>000a (start)</td>
<td>8 000 (load “00”)</td>
<td>sum = 0</td>
</tr>
<tr>
<td>000b</td>
<td>1 003 (store 03)</td>
<td></td>
</tr>
<tr>
<td>000c</td>
<td>0 002 (load 02)</td>
<td>i = n</td>
</tr>
<tr>
<td>000d</td>
<td>1 004 (store 04)</td>
<td></td>
</tr>
<tr>
<td>000e</td>
<td>0 001 (load 01)</td>
<td>p = base</td>
</tr>
<tr>
<td>000f</td>
<td>1 005 (store 05)</td>
<td></td>
</tr>
<tr>
<td>0010 (loop)</td>
<td>0 004 (load 04)</td>
<td>if i = 0, then quit</td>
</tr>
<tr>
<td>0011</td>
<td>5 01d (if 0 goto 1d)</td>
<td></td>
</tr>
<tr>
<td>0012</td>
<td>6 005 (load *05)</td>
<td>sum = sum + *p</td>
</tr>
<tr>
<td>0013</td>
<td>2 003 (add 03)</td>
<td></td>
</tr>
<tr>
<td>0014</td>
<td>1 003 (store 03)</td>
<td></td>
</tr>
<tr>
<td>0015</td>
<td>8 001 (load “1”)</td>
<td>p = p + 1</td>
</tr>
<tr>
<td>0016</td>
<td>2 005 (add 05)</td>
<td></td>
</tr>
<tr>
<td>0017</td>
<td>1 005 (store 05)</td>
<td></td>
</tr>
<tr>
<td>0018</td>
<td>8 001 (load “1”)</td>
<td>i = i – 1</td>
</tr>
<tr>
<td>0019</td>
<td>3 000 (negate)</td>
<td></td>
</tr>
<tr>
<td>001a</td>
<td>2 004 (add 04)</td>
<td></td>
</tr>
<tr>
<td>001b</td>
<td>1 004 (store 04)</td>
<td></td>
</tr>
<tr>
<td>001c</td>
<td>4 010 (goto 10)</td>
<td>goto loop</td>
</tr>
<tr>
<td>001d</td>
<td>3 001 (halt)</td>
<td></td>
</tr>
</tbody>
</table>
The sum computed is $200a$ if when the program starts, $000b$ is stored in word 1 and $0003$ is stored in word 2?

There is no simple way to determine if a number is negative, but it can be done, using a special subroutine written for this purpose. To determine if a value $x$ is negative, let $y = x$ and $z = -x$. Now repetitively subtract 1 from $y$ and $z$. Eventually one of them will become zero. If $y$ becomes zero first, then $x$ is positive. If $z$ becomes zero first then $x$ is negative, and if they both become zero together, then either $x$ is zero, or $x$ is equal to the most negative value that can be represented in 16 bits. These two cases are easy to tell apart, so it is possible (but time consuming) to determine if $x$ is negative.