HW01 Model Solutions

1. Make the following base conversions. Use shortcuts when applicable.

(a) 101100101_2 to decimal ***357***

$$\begin{aligned} 101100101_2 &= 1\times 2^0 + 0\times 2^1 + 1\times 2^2 + 0\times 2^3 + 0\times 2^4 + 1\times 2^5 + 1\times 2^6 + 0\times 2^7 + 1\times 2^8 \\ &= 1\times 2^0 + 1\times 2^2 + 1\times 2^5 + 1\times 2^6 + 1\times 2^8 \\ &= 1\times 1 + 1\times 4 + 1\times 32 + 1\times 64 + 1\times 256 \\ &= 357 \end{aligned}$$

(b) 101110101111010_2 to hexadecimal ***5D7A***

Conversion from binary to hex is a matter of grouping bits

$\overbrace{0101}^{5}\ \overbrace{1101}^{D}\ \overbrace{0111}^{7}\ \overbrace{1010}^{A}$

(c) 101110101111010_2 to octal ***56572***

Similarly

$\overbrace{101}^{5}\ \overbrace{110}^{6}\ \overbrace{101}^{5}\ \overbrace{111}^{7}\ \overbrace{010}^{2}$

(d) 539_{10} to binary ***10 0001 1011***

$$\begin{array}{r|l} 539 & 2 \\ \hline -4 & 269 \\ \hline 13 & \\ -12 & \\ \hline 19 & \\ -18 & \\ \hline 1 & \end{array} \quad \begin{array}{r|l} 269 & 2 \\ \hline -2 & 134 \\ \hline 06 & \\ -6 & \\ \hline 09 & \\ -8 & \\ \hline 1 & \end{array} \quad \begin{array}{r|l} 134 & 2 \\ \hline -12 & 67 \\ \hline 14 & \\ -14 & \\ \hline 0 & \end{array} \quad \begin{array}{r|l} 67 & 2 \\ \hline -6 & 33 \\ \hline 07 & \\ -6 & \\ \hline 1 & \end{array} \quad \begin{array}{r|l} 33 & 2 \\ \hline -2 & 16 \\ \hline 13 & \\ -12 & \\ \hline 1 & \end{array}$$

$$\begin{array}{r|l} 16 & 2 \\ \hline -16 & 8 \\ \hline 0 & \end{array} \quad \begin{array}{r|l} 8 & 2 \\ \hline -8 & 4 \\ \hline 0 & \end{array} \quad \begin{array}{r|l} 4 & 2 \\ \hline -4 & 2 \\ \hline 0 & \end{array} \quad \begin{array}{r|l} 2 & 2 \\ \hline -2 & 1 \\ \hline 0 & \end{array}$$

(e) 6527_{10} to octal ***14577***

$$\begin{array}{r|l} 6527 & 8 \\ \hline -64 & 815 \\ \hline 12 & \\ -8 & \\ \hline 47 & \\ -40 & \\ \hline 7 & \end{array} \quad \begin{array}{r|l} 815 & 8 \\ \hline -8 & 101 \\ \hline 01 & \\ -0 & \\ \hline 15 & \\ -8 & \\ \hline 7 & \end{array} \quad \begin{array}{r|l} 101 & 8 \\ \hline -8 & 12 \\ \hline 21 & \\ -16 & \\ \hline 5 & \end{array} \quad \begin{array}{r|l} 12 & 8 \\ \hline -8 & 1 \\ \hline 4 & \end{array}$$

(f) 18107_{10} to hexadecimal ***46BB***

$$
\begin{array}{r|l}
1\,8\,1\,0\,7 & 1\,6 \\
\cline{2-2}
-\,1\,6 & 1\,1\,3\,1 \\
\cline{1-1}
\quad 2\,1 & \\
-\,1\,6 & \\
\cline{1-1}
\quad 5\,0 & \\
-\,4\,8 & \\
\cline{1-1}
\quad 2\,7 & \\
-\,1\,6 & \\
\cline{1-1}
\quad 1\,1 &
\end{array}
\qquad
\begin{array}{r|l}
1\,1\,3\,1 & 1\,6 \\
\cline{2-2}
-\,1\,1\,2 & 7\,0 \\
\cline{1-1}
\quad 1\,1 & \\
-\,\quad 0 & \\
\cline{1-1}
\quad 1\,1 &
\end{array}
\qquad
\begin{array}{r|l}
7\,0 & 1\,6 \\
\cline{2-2}
-\,6\,4 & 4 \\
\cline{1-1}
\quad 6 &
\end{array}
$$

(g) 365_8 to binary ***11 110 101***

$$\overbrace{011}^{3}\ \overbrace{110}^{6}\ \overbrace{101}^{0}$$

(h) 5022_8 to decimal ***2578***

$$\begin{aligned} 5022_8 &= 2\times 8^0 + 2\times 8^1 + 0\times 8^2 + 5\times 8^3 \\ &= 2\times 1 + 2\times 8 + 0\times 64 + 5\times 512 \\ &= 2578 \end{aligned}$$

(i) 467_8 to hexadecimal ***137***

Conversion from octal to hex is a matter of regrouping bits. Requires an intermediate conversion to binary.

$$\overbrace{100}^{4}\ \overbrace{110}^{6}\ \overbrace{111}^{7}$$

$$\overbrace{0001}^{1}\ \overbrace{0011}^{3}\ \overbrace{0111}^{7}$$

(j) $D7A_{16}$ to binary ***1101 0111 1010***

$$\overbrace{1101}^{\mathrm{D}}\ \overbrace{0111}^{7}\ \overbrace{1010}^{\mathrm{A}}$$

(k) $E49F_{16}$ to decimal ***58527***

$$\begin{aligned} E49F_{16} &= 15\times 16^0 + 9\times 16^1 + 4\times 16^2 + 14\times 16^3 \\ &= 15 + 9\times 16 + 4\times 256 + 14\times 4096 \\ &= 15 + 144 + 1024 + 57344 \\ &= 58527 \end{aligned}$$

(l) $3G2_{17}$ to 13-base notation ***69A***

Convert to decimal first, and then convert to the desired base.

$$\begin{aligned} 3G2_{17} &= 2\times 17^0 + 16\times 17^1 + 3\times 17^2 \\ &= 2 + 16\times 17 + 3\times 289 \\ &= 1141 \end{aligned}$$

$$
\begin{array}{r|l}
1141 & 13 \\
\cline{2-2}
-\,104 & 87 \\
\cline{1-1}
\;\;101 & \\
-\;\;91 & \\
\cline{1-1}
\;\;\;10 &
\end{array}
\qquad
\begin{array}{r|l}
87 & 13 \\
\cline{2-2}
-\,78 & 6 \\
\cline{1-1}
\;\;9 &
\end{array}
$$

2. What is the twos complement representation of 68 in 8-bit, 16-bit, 32-bit and 64-bit notations?.

The absolute value of the number in binary is

1000100

In order to get the two's complement representation of the number for a given number of bits, take the number and fill the remaining bits with 0.
0100 0100
0000 0000 0100 0100
0000 0000 0000 0000 0000 0000 0100 0100
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0100 0100

Please note that the two's complement operation that consisted of flipping bits and adding one, is the equivalent of multiplying something by -1.
No further steps are necessary for positive numbers.

3. Convert -11 to binary using 8-bit, 16-bit, 32-bit and 64-bit two's complement notations. Have you encountered any of the values during the previous problems, and if so, where? Explain the reason the values coincide.

Take the absolute value of the number and convert to binary.

1011

Then construct the two's complement representations of the positive part.
0000 1011
0000 0000 0000 1011
0000 0000 0000 0000 0000 0000 0000 1011
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 1011

If the given number is a positive number, **stop here.**

Everything from here onwards, applies to **negative numbers only.**

Perform the two's complement operation on each representation in order to get the desired negative values.

Flipping the bits results in:

1111 0100
1111 1111 1111 0100
1111 1111 1111 1111 1111 1111 1111 0100
1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 0100

Adding one results in

1111 0101
1111 1111 1111 0101
1111 1111 1111 1111 1111 1111 1111 0101
1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 0101

The 8-bit representation of -11, is the same as the result from 1.g, which was a positive number. These numbers are in fact two's complements of each other for 8 bits.

Please also note that an 8-bit representation of -11 in 32 bits would be

0000 0000 0000 0000 0000 0000 1111 0100

and is the original positive value.

4. Describe a model that can fully represent and store the state of the board for a game of tic tac toe in a computer. How many bytes of memory would your model require? Is your model optimal?

The board of a game of tic-tac-toe, comprises of 9 cells, each of which can contain one of the three values of X, O and empty. Each of these values needs 2 bits in order to be represented. Having 9 of these values requires a total of 18 bits, but since the unit for storing information is a byte, the model requires 3 whole bytes even though there are 6 unused bits.

While this model is fully sufficient as a stand-alone solution for practical purposes, there are ways to reduce the memory footprint at the expense of more calculation and overhead memory by designating values for valid states of the game.

There are 756 unique states for the game. Each has 7 more equivalent variations that are obtained by rotating and flipping the original, resulting in a total of 6048 variations. This number can be stored in 13 bits, which is under 2 bytes.

5. Take the ASCII values of the first three letters of your surname and use them as hex values in order to produce a 24-bit web color. What is your color like?

Capital letters come before small letters in ASCII, and all letters have a value that is

smaller than 127. Therefore, answers will result in a sort of a dark teal.

6. Is it possible to play console games (such as those made for PlayStation) on a PC? How?

IBM compatible computers (almost every PC not recently made by Apple) rely on Intel architecture (Intel and AMD chips). If a console also uses the same architecture, then the binary executables made for those consoles can run on that PC, natively. However, there are always additional software dependencies that impose further limitations and that is why Windows programs will not run on Linux without workarounds, even though they are running on the same hardware.

If the console has a different architecture (Play Station), then the short answer is no. A binary executable made for a different architecture will only run on that platform.

There are, however, software solutions that "emulate" other platforms, making executables "feel" as if they are in their native environment, and translate the binary instructions with some performance pentalty. This approach is often used during development for making these games and various applications in the first place and for testing and debugging them.

Sometimes, developers opt for embedding an emulator in the program itself in order to reduce their efforts, at the expense of some loss of functionality.

This document was produced using LaTeX

$$\hat{A}_b(\gamma) = -\int_0^{\infty} \frac{x^{e^x}}{\sqrt{1-\frac{\phi^2}{\theta^2}}} \partial\sigma K_L e(\tau) d\sigma \tag{1}$$