

# 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

$\overset{5}{\underbrace{0101}} \overset{D}{\underbrace{1101}} \overset{7}{\underbrace{0111}} \overset{A}{\underbrace{1010}}$

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

Similarly

$\overset{5}{\underbrace{101}} \overset{6}{\underbrace{110}} \overset{5}{\underbrace{101}} \overset{7}{\underbrace{111}} \overset{2}{\underbrace{010}}$

(d)  $593_{10}$  to binary **10 0101 0001**

$$\begin{array}{r|l}
 \begin{array}{r}
 593 \\
 -4 \\
 \hline
 19 \\
 -18 \\
 \hline
 13 \\
 -12 \\
 \hline
 1
 \end{array} & \begin{array}{l} 2 \\ 296 \end{array} \\
 \hline
 \end{array}
 \quad
 \begin{array}{r|l}
 \begin{array}{r}
 296 \\
 -2 \\
 \hline
 09 \\
 -8 \\
 \hline
 16 \\
 -16 \\
 \hline
 0
 \end{array} & \begin{array}{l} 2 \\ 148 \end{array} \\
 \hline
 \end{array}
 \quad
 \begin{array}{r|l}
 \begin{array}{r}
 148 \\
 -14 \\
 \hline
 08 \\
 -8 \\
 \hline
 0
 \end{array} & \begin{array}{l} 2 \\ 74 \end{array} \\
 \hline
 \end{array}
 \quad
 \begin{array}{r|l}
 \begin{array}{r}
 74 \\
 -6 \\
 \hline
 14 \\
 -14 \\
 \hline
 0
 \end{array} & \begin{array}{l} 2 \\ 37 \end{array} \\
 \hline
 \end{array}
 \quad
 \begin{array}{r|l}
 \begin{array}{r}
 37 \\
 -2 \\
 \hline
 17 \\
 -16 \\
 \hline
 1
 \end{array} & \begin{array}{l} 2 \\ 18 \end{array} \\
 \hline
 \end{array}$$
  

$$\begin{array}{r|l}
 \begin{array}{r}
 18 \\
 -18 \\
 \hline
 0
 \end{array} & \begin{array}{l} 2 \\ 9 \end{array} \\
 \hline
 \end{array}
 \quad
 \begin{array}{r|l}
 \begin{array}{r}
 9 \\
 -8 \\
 \hline
 1
 \end{array} & \begin{array}{l} 2 \\ 4 \end{array} \\
 \hline
 \end{array}
 \quad
 \begin{array}{r|l}
 \begin{array}{r}
 4 \\
 -4 \\
 \hline
 0
 \end{array} & \begin{array}{l} 2 \\ 2 \end{array} \\
 \hline
 \end{array}
 \quad
 \begin{array}{r|l}
 \begin{array}{r}
 2 \\
 -2 \\
 \hline
 0
 \end{array} & \begin{array}{l} 2 \\ 1 \end{array} \\
 \hline
 \end{array}$$

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

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

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

$$\begin{array}{r|l}
 \begin{array}{r}
 18107 \\
 -16 \\
 \hline
 21 \\
 -16 \\
 \hline
 50 \\
 -48 \\
 \hline
 27 \\
 -16 \\
 \hline
 11
 \end{array}
 &
 \begin{array}{r}
 16 \\
 1131
 \end{array}
 \end{array}
 \quad
 \begin{array}{r|l}
 \begin{array}{r}
 1131 \\
 -112 \\
 \hline
 11 \\
 -0 \\
 \hline
 11
 \end{array}
 &
 \begin{array}{r}
 16 \\
 70
 \end{array}
 \end{array}
 \quad
 \begin{array}{r|l}
 \begin{array}{r}
 70 \\
 -64 \\
 \hline
 6
 \end{array}
 &
 \begin{array}{r}
 16 \\
 4
 \end{array}
 \end{array}$$

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

$$\begin{array}{ccc}
 \overset{3}{\underbrace{011}} & \overset{6}{\underbrace{110}} & \overset{0}{\underbrace{101}}
 \end{array}$$

(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.

$$\begin{array}{ccc}
 \overset{4}{\underbrace{100}} & \overset{6}{\underbrace{110}} & \overset{7}{\underbrace{111}} \\
 \overset{1}{\underbrace{0001}} & \overset{3}{\underbrace{0011}} & \overset{7}{\underbrace{0111}}
 \end{array}$$

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

$$\begin{array}{ccc}
 \overset{D}{\underbrace{1101}} & \overset{7}{\underbrace{0111}} & \overset{A}{\underbrace{1010}}
 \end{array}$$

(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.

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

$$\begin{array}{r|l}
\begin{array}{r}
- \begin{array}{r} 1141 \\ 104 \\ \hline 101 \\ - \quad 91 \\ \hline 10 \end{array}
\end{array}
&
\begin{array}{r}
\begin{array}{r} 13 \\ 87 \end{array}
\end{array}
&
\begin{array}{r|l}
\begin{array}{r} - \begin{array}{r} 87 \\ 78 \\ \hline 9 \end{array}
\end{array}
&
\begin{array}{r} 13 \\ 6 \end{array}
\end{array}$$

2. What is the two's 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 was a positive number, **stop here.**

Steps from here onwards, apply to **negative numbers only.**

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

Flipping the bits for each of our representations, 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 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 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. 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.

The long answer, however, includes 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 penalty. 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.

In case you were curious, this document was produced using L<sup>A</sup>T<sub>E</sub>X

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