Chapter 1 A Tour of Computer Systems

Problem 1.1

A. We use the formula with $\alpha = \frac{3}{5}$ and $k = \frac{150}{100} = \frac{3}{2}$.

$$\begin{aligned} S &= \frac{1}{\left(1 - \frac{3}{5}\right) + \frac{3}{5} \cdot \frac{2}{3}} \\ &= \frac{1}{\frac{2}{5} + \frac{2}{5}} \\ &= \frac{5}{4} \\ &= 1.25\times \end{aligned}$$

B. We use the formula and work our way back:

$$\begin{aligned} \frac{5}{3} &= \frac{1}{\left(1 - \frac{3}{5}\right) + \frac{3}{5k}} \\ \frac{3}{5} &= \frac{2}{5} + \frac{3}{5k} \\ \frac{1}{5} &= \frac{3}{5k} \\ 1 &= \frac{3}{k} \\ k &= 3 \end{aligned}$$

So the drive through Montana needs a speedup of $3\times$ which is 300 km/hr.

Problem 1.2

Use the formula with $\alpha = \frac{4}{5}$ and $S = 2$ and solve for k.

$$\begin{aligned} 2 &= \frac{1}{\left(1 - \frac{4}{5}\right) + \frac{4}{5k}} \\ \frac{2}{5} + \frac{8}{5k} &= 1 \\ \frac{8}{5k} &= \frac{3}{5} \\ \frac{1}{k} &= \frac{3}{8} \\ k &= \frac{8}{3} \end{aligned}$$

Chapter 2 Representing and Manipulating Information

Problem 2.1

A. `0x39A7F8` to binary: `0011 1001 1010 0111 1111 1000`

B. `1100100101111011` to hexadecimal: `0xC97B`

C. `0xD5E4C` to binary: `1101 0101 1110 0100 1100`

D. `1001101110011110110101` to hexadecimal: `0x26E7B5`

Problem 2.2

n	2^n (decimal)	2^n (hexadecimal)
`9`	`512`	`0x200`
`19`	`524288`	`0x80000`
`14`	`16384`	`0x4000`
`16`	`65536`	`0x10000`
`17`	`131072`	`0x20000`
`5`	`32`	`0x20`
`7`	`128`	`0x80`

Problem 2.3

Decimal	Binary	Hexadecimal
`0`	`0000 0000`	`0x00`
`167`	`1010 0111`	`0xA7`
`62`	`0011 1110`	`0x3E`
`188`	`1011 1100`	`0xBC`
`55`	`0011 0111`	`0x37`
`136`	`1000 1000`	`0x88`
`243`	`1111 0011`	`0xF3`
`82`	`0101 0010`	`0x52`
`172`	`1010 1100`	`0xAC`

Decimal	Binary	Hexadecimal
`231`	`1110 0111`	`0xE7`

Problem 2.4

A. `0x503c` $+$ `0x8` $=$ `0x5044`

B. `0x503c` $-$ `0x40` $=$ `0x4ffc`

C. `0x503c` $+$ `64` $=$ `0x507c`

D. `0x50ea` $-$ `0x503c` $=$ `0xae`

Problem 2.5

	Little endian	Big endian
A.	`21`	`87`
B.	`21 43`	`87 65`
C.	`21 43 65`	`87 65 43`

Problem 2.6

A.

`0x00359141` in binary: `0000 0000 0011 0101 1001 0001 0100 0001`

`0x4A564504` in binary: `0100 1010 0101 0110 0100 0101 0000 0100`

B.

```
00000000001101011001000101000001
  01001010010101100100010100000100
           *********************
```

There are 21 matching bits.

C.

The whole integer occurs in the float representation, except for the most-significant bit which is a 1. Similarly, some of the most-significant bits of the float representation do not occur in the int representation.

Problem 2.7

It prints `61 62 63 64 65 66` (it does not print the terminating null character because the `strlen` function does not count it).

Problem 2.8

Operation	Result
`a`	`[01101001]`
`b`	`[01010101]`
`~a`	`[10010110]`
`~b`	`[10101010]`
`a & b`	`[01000001]`
`a | b`	`[01111101]`
`a ^ b`	`[00111100]`

Problem 2.9

A. The following colors completent each other:

Black ↔ White
Blue ↔ Yellow
Green ↔ Magenta
Cyan ↔ Red

B.

Blue | Green = Cyan
Yellow & Cyan = Green
Red ^ Magenta = Blue

Problem 2.10

Step	$*x$	$*y$
Initially	`a`	`b`
Step 1	`a`	`a ^ b`
Step 2	`a ^ (a ^ b) = b`	`a ^ b`
Step 3	`b`	`b ^ (a ^ b) = a`

Problem 2.11

A. In the final iteration we have `first = k` and `last = k` (swap the middle element with itself).

B. In this case `*x` and `*y` point to the same address and the steps become:

Step	*x	*y
Initially	`a`	`a`
Step 1	`a ^ a = 0`	`a ^ a = 0`
Step 2	`0 ^ 0 = 0`	`0 ^ 0 = 0`
Step 3	`0 ^ 0 = 0`	`0 ^ 0 = 0`

C. We can fix it by changing the condition to `first < last` since the middle element does not need to be swapped anyway.

Problem 2.12

A. `x & 0xFF` leaves the least significant byte and sets everything else to zero.

B. `x ^ ~0xFF` inverts everything except the least significant byte.

C. `x | 0xFF` sets the least significant byte to ones and leaves everything else.

Problem 2.13

`x | y` is equivalent to `bis(x, y)`.

`x ^ y` is equivalent to `bis(bic(x, y), bic(y, x))`.

Problem 2.14

We have `x = 0110 0110` and `y = 0011 1001`.

Expression	Value	Expression	Value
`x & y`	`0010 0000`	`x && y`	`1`
`x | y`	`0111 1111`	`x || y`	`1`
`~x | ~y`	`1111 1111 1111 1111 1111 1111 1101 1111` (assuming 32-bit int)	`!x || !y`	`0`
`x & !y`	`0`	`x && ~y`	`1`

Problem 2.15

`!(x ^ y)` is equivalent to `x == y` because `x ^ y` will be `0` only if all the bits match.

Problem 2.16

x	x << 3	x >> 2 (logical)	x >> 2 (arithmetic)
`0xC3 = 1100 0011`	`0001 1000 = 0x18`	`0011 0000 = 0x30`	`1111 0000 = 0xF0`
`0x75 = 0111 0101`	`1010 1000 = 0xA8`	`0001 1101 = 0x1D`	`0001 1101 = 0x1D`
`0x87 = 1000 0111`	`0011 1000 = 0x38`	`0010 0001 = 0x21`	`1110 0001 = 0xE1`
`0x66 = 0110 0110`	`0011 0000 = 0x30`	`0001 1001 = 0x19`	`0001 1001 = 0x19`

Problem 2.17

Hexadecimal	Binary	$B2U_4(x)$	$B2T_4(x)$
`0xE`	`[1110]`	$2^3 + 2^2 + 2^1 = 14$	$-2^3 + 2^2 + 2^1 = -2$
`0x0`	`[0000]`	0	0
`0x5`	`[0101]`	$2^2 + 2^0 = 5$	$2^2 + 2^0 = 5$
`0x8`	`[1000]`	$2^3 = 8$	$-2^3 = -8$
`0xD`	`[1101]`	$2^3 + 2^2 + 2^0 = 13$	$-2^3 + 2^2 + 2^0 = -3$
`0xF`	`[1111]`	$2^3 + 2^2 + 2^1 + 2^0 = 15$	$-2^3 + 2^2 + 2^1 + 2^0 = -1$

Problem 2.18

A. `0x2e0 = 736`

B. `-0x58 = -88`

C. `0x28 = 40`

D. `-0x30 = -48`

E. `0x78 = 120`

F. `0x88 = 136`

G. `0x1f8 = 504`

H. `0xc0 = 192`

I. `-0x48 = -72`

Problem 2.19

x	$T2U_4(x)$
-8	8
-3	$2^3 + 2^2 + 2^0 = 13$
-2	$2^3 + 2^2 + 2^1 = 14$
-1	$2^3 + 2^2 + 2^1 + 2^0 = 15$
0	0
5	5

Problem 2.20

Equation 2.5 can be used to solve the previous problem. Since $\omega = 4$, we need to add $2^4 = 16$ to all negative numbers in Two's Complement. For example, $-8 + 16 = 8$ and $-1 + 16 = 15$. Positive numbers (and zero) stay the same.

Problem 2.21

Expression	**Type**	**Evaluation**
$-2147483647 - 1 == 2147483648\text{U}$	Unsigned	1
$-2147483647 - 1 < 2147483647$	Signed	1
$-2147483647 - 1\text{U} < 2147483647$	Unsigned	0
$-2147483647 - 1 < -2147483647$	Signed	1
$-2147483647 - 1\text{U} < -2147483647$	Unsigned	1