

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	<code>0x200</code>
19	524288	<code>0x80000</code>
14	16384	<code>0x4000</code>
16	65536	<code>0x10000</code>
17	131072	<code>0x20000</code>
5	32	<code>0x20</code>
7	128	<code>0x80</code>

Problem 2.3

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

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 complement each other:

Black \leftrightarrow White

Blue \leftrightarrow Yellow

Green \leftrightarrow Magenta

Cyan \leftrightarrow 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	$a \wedge a = 0$	$a \wedge a = 0$
Step 2	$0 \wedge 0 = 0$	$0 \wedge 0 = 0$
Step 3	$0 \wedge 0 = 0$	$0 \wedge 0 = 0$

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
<code>x & y</code>	0010 0000	<code>x && y</code>	1
<code>x y</code>	0111 1111	<code>x y</code>	1
<code>~x ~y</code>	1111 1111 1111 1111 1111 1111 1101 1111 (assuming 32-bit int)	<code>!x !y</code>	0
<code>x & !y</code>	0	<code>x && ~y</code>	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	$x \ll 3$	$x \gg 2$ (logical)	$x \gg 2$ (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
$-2147483647 - 1 == 2147483648U$	Unsigned	1
$-2147483647 - 1 < 2147483647$	Signed	1
$-2147483647 - 1U < 2147483647$	Unsigned	0
$-2147483647 - 1 < -2147483647$	Signed	1
$-2147483647 - 1U < -2147483647$	Unsigned	1

Problem 2.22

- A. $[1011] = -2^3 + 2^1 + 2^0 = -5$
 B. $[11011] = -2^4 + 2^3 + 2^1 + 2^0 = -5$
 C. $[111011] = -2^5 + 2^4 + 2^3 + 2^1 + 2^0 = -5$

Problem 2.23

w	$\text{fun1}(w)$	$\text{fun2}(w)$
0x00000076	0x00000076	0x00000076
0x87654321	0x00000021	0x00000021
0x000000C9	0x000000C9	0xFFFFFC9
0xEDCBA987	0x00000087	0xFFFFF87

`fun1` keeps only the least significant byte and sets the other three to all zeroes, resulting in a value between 0 and 255. `fun2` also extracts the least significant byte, but it performs sign extension instead of zero extension, which results in a value between -128 and 127.

Problem 2.24

Hex		Unsigned		Two's complement	
Original	Truncated	Original	Truncated	Original	Truncated
0	0	0	0	0	0
2	2	2	2	2	2
9	1	9	1	-7	1
B	3	11	3	-5	3
F	7	15	7	-1	-1

We can use the equations to verify these results. For example, in hex F truncates to 7, in unsigned $B2U_4(1111) \bmod 2^3 = 7$ and in two's complement $U2T_3(B2U_4(1111) \bmod 2^3) = -1$.

Problem 2.25

Because `length` is unsigned the expression $0 - 1$ evaluates to `UMax`. The comparison has an unsigned integer on one side, which means the other side will also be treated as unsigned. Of course every unsigned number is $\leq \text{UMax}$ and so we try to access invalid array elements.

We can fix it by changing the condition to `i < length` or changing `length` to a signed integer.

Problem 2.26

A. The function returns wrong results in case `t` is longer than `s`.

B. The problem is that `strlen` returns a `size_t` which is unsigned. When calculating `strlen(s) - strlen(t)` where `t` is longer than `s` unsigned arithmetic is used, resulting in a number close to `UMax` instead of a negative number. This is obviously greater than 0 so the function incorrectly says that `s` is longer.

C. We can fix it by changing the condition to `strlen(s) > strlen(t)`.