Data representation
Solved Exercises

Exercise 1. Indicate the representation of the following numbers, reasoning your answer:
a) -16 in two's complement with 5 bits
b) -16 in one's complement with 5 bits
c) +13 in sign-magnitude with 5 bits
d) -14 in two's complement with 5 bits

Solution:
a) The range of representation of numbers in two's complement with 5 bits is $[-2^{5-1}..2^{5-1}-1] = [-16..15]$. 16 in binary is 10000. We have to complement, 01111 and add 1. Therefore, -16 in two's complement with 5 bits is 10000.
b) The range of number representation in one's complement with 5 bits is $[-(2^{5-1}-1)..2^{5-1}-1] = [-15..15]$. Therefore, the number -16 cannot be represented.
c) 13 in binary is 1101. Sign-magnitude introduce at the beginning a sign bit, in this case 0 to indicate that it is positive. The number 13 is sign-magnitude with 5 bits is 01101.
d) 14 in binary with 5 bits is 01110. We complement, 10001 and add 1, so we get 10010.

Exercise 2. Indicate the representation of the following numbers:
a) -64 in one's complement with 7 bits
b) -64 two's complement with 7 bits
c) 12 in sign-magnitude with 6 bits
d) 18 in two's complement with 5 bits

Solution:
a) In one's complement with 7 bits, the range of representation is [-63, 63], so the number -64 is not representable.
b) 64 in binary with 7 bits is 1000000. Complement 0111111 and add 1, the result being 100000
c) 12 in sign-magnitude with 6 bits 001100
d) In two's complement with 5 bits, the range of representation is [-16, 15], therefore, the number is not representable.

Exercise 3. How do you detect an overflow in two's complement when performing an addition operation?

Solution:

It is detected when the two operands have the same sign and the result has a different sign from the operands.

Exercise 4. Represent in the IEEE 754 standard of single precision the value -36

Solution:

36 in binary is 100100. $100100 = 1.00100 \times 2^5$. So:

- The sign bit is 1, because the number is negative.
- The exponent is 5, therefore, the exponent stored is 5 + 127 = 132. In binary is 10000100
- The mantissa is 001000000 …. 00000

Therefore, the number -36 is represented as 11000010000100000000000000000000 in IEEE 754.

Exercise 5. Indicate the decimal value of the following hexadecimal number 0x00600000 representing a floating point number according to IEEE 754 (single precision)

Solution:

0x00600000 in binary is 00000000011000000000000000000000

Sign = 0, positive number
Exponent = 00000000
Mantissa = 1100000…0000

This is a non-normalized number whose value is $0,11 \times 2^{-126} = 0,75 \times 2^{-126}$

Exercise 6. Represent the number -24.50 using the IEEE 754 (single precision) floating point standard. Express this representation in binary and hexadecimal.

Solution:

$24,5_{(10} = 11000.1_{(2} = 1,10001 \times 2^4$

Sign = 1, negative number
Exponent = 4 + 127 = 131 = 10000011
Mantissa = 1000100000 .. 00000

In binary: 11000000111000100000 …….00000
In Hexadecimal: 0xC1C40000

Exercise 7. We want to represent integers within the range -8191...8191:

a) What is the number of bits needed if you want to use a representation in one's complement?
b) What is the number of bits needed if you want to use a representation in sign-magnitude?

Solution:

We need in both cases 14 bits. With 14 bits the range of representation in both cases is $-(2^{13}-1) \ldots 2^{13}-1 = -8191 \ldots 8191$

Exercise 8. What is the smallest normalized positive number that can be represented using the IEEE 754 single precision standard? Justify your answer. Also indicate the smallest non-normalized positive number that can be represented. Please justify your answer in the same way.

Solution:

a) The smallest normalized positive number that can be represented in the standard IEEE 754 (32 bits) is:
0 00000001 00000000000000000000000
positive smaller normalized

The value is $1.0 \times 2^{1-127} = \mathbf{2^{-126}}$

b) The smallest non-normalized positive number that can be represented in the standard is:
0 00000000 00000000000000000000001
positive non-normalized smaller

the value is $2^{-23} \times 2^{-126} = \mathbf{2^{-149}}$

Exercise 9. Represent in the 32-bit IEEE 754 floating point standard the values 10,25 and 6,75. Express the result in hexadecimal. Next, make the sum of the previous numbers represented in IEEE 754, indicating the steps that you are making in each moment.

Solution:

a) $10{,}25_{(10} = 1010{,}01_{(2} = 1{,}01001 \times 2^3{}_{(2}$
Sign = 0
Exponent = 127+3 = $130_{(10} = 10000010_{(2}$
Mantissa = 010010000…00

In binary: 0100 0001 0010 0100 00…00
In Hexadecimal: 0x41240000

$6{,}75_{(10} = 110{,}11_{(2} = 1{,}1011 \times 2^2{}_{(2}$
Sign = 0
Exponent = 127+2 = $129_{(10} = 10000001_{(2}$
Mantissa = 10110000…00

In binary: 0100 0000 1101 1000 00…00
In Hexadecimal: 0x40D80000

To add both numbers in IEEE754, the first thing we have to do is to equalize exponents and take into account when adding the implicit bit of the mantissa. Then perform the addition, and if the result is not normalized, normalize it.

10,25 = 0 10000010 1.01001000000000000000000
6,75 = 0 10000010 0.11011000000000000000000
Add = 0 10000010 10.00100000000000000000000
Normalized result = 0 10000011 00010000000000000000000
In Hexadecimal: 0x41880000

Exercise 10. Indicate the decimal value of the following number represented in the IEEE 754 standard of single precision: 0xBF400000.

Solution:

The decimal value of 0xBF400000 is:

In binary: 1011 1111 0100 0000…00

Sign = Negative
Stored exponent = $01111110_{(2} = 126_{(10}$ => Exponent = -1
Stored mantissa = 1

Number: $-1{,}1 \times 2^{-1}{}_{(2} = -0{,}11_{(2} = -0{,}75_{(10}$

Exercise 11. According to IEEE 754 standard answer the following questions:

a) In a 32-bit IEEE 754 representation, indicate in a reasoned manner the number of non-normalized values that can be represented.

b) In a 32-bit computer, can the value $2^{27}+1$ be exactly represented in a `float` type variable? and in an `int` type variable? Reason your answer.

c) Represent in the IEEE 754 double precision standard the value 12,5. Express the result in hexadecimal.

Solution:

a) A non-normalized value corresponds to an exponent (8 bits) whose value is 0 and a mantissa (23 bits) with a value other than zero. The number of elements that can be represented is, therefore, 2 (positive and negative) $\times\ 2^{23}-1 = 2 \cdot (2^{23}-1)$

b) The value $2^{27} + 1 = 1000000000000000000000000001_{(2} = 1.000000000000000000000000001 \times 2^{27}{}_{(2}$

Since in the mantissa of a float type number (IEEE 754 of 32 bits) only 23 bits can be stored, apart from the implicit bit, for the previous number the 4 less significant bits of the number could not be stored, so the number could not be represented exactly. In an int variable, which uses two's complement, the range of representation is $[-2^{31}, 2^{31}-1$. In this case, it can be represented exactly the number.

c)

$12{,}5_{(10} = 1100{,}1_{(2} = 1{,}1001 \times 2^{3}{}_{(2}$

Sign = 1
Exponent = 1023+3 = $1026_{(10}$ $1024+2_{(10}$ = $10000000010_{(2}$
Mantissa = 1001…. 000000 (52 bits)

The representation of 12,5 ss:

0 100 0000 0010 1001 00000000 …… 00000

In hexadecimal: 0x4029000000000000

Exercise 12. Consider the following fragment written in C, running on a 32-bit computer.

```
double A;
float  B;
int    C;

A =  pow(2, 28) + 5;    // 2^28 +5
B = (float) A;
C = (int)   A;
```

After executing the previous code fragment, indicate in a reasoned way the value, in hexadecimal, that is stored in the variables A, B and C.

Solution:

A = pow(2, 28) + 5; // 2^{28} +5

The value $2^{28} + 5 = 10000000000000000000000000101_{(2}$
$= 1.0000000000000000000000000101 \times 2^{28}{}_{(2}$

When this value is stored in a double type variable, which corresponds to the IEEE 754 double precision format, it is obtained:

Sign = 0
Exponent = 1023+28 = $1051_{(10}$ = $1024+16+11_{(10}$ = $10000011011_{(2}$

Mantissa = 0000000000000000000000000101000000000 ….. 000000 (52 bits)

The stored value in binary is:

0 10000011011 0000000000000000000000000101000000000 ….. 000000

And in hexadecimal: 0x41B0000005000000

B = (float) A;

The value $2^{28} + 5 = 1.0000000000000000000000000101 \times 2^{28}{}_{(2}$.

When this value is stored in a float type variable, which corresponds to the IEEE 754 format of simple precision, we obtain:

Sign= 0
Exponent = 127+28 = $155_{(10} = 10011011_{(2}$
Mantissa = 0000…. 000 (23 bits, t he last less significant bits are lost)

The stored value is in binary:

0 10011011 0000…. 000

In hexadecimal: 0x4D8000000000000

The mantissa of a `float` type number (IEEE 754 of 32 bits) can only store 23 bits (without considering the implicit bit) so the 4 less significant bits of the number could not be stored.

`C = (int) A;`

When the value of A (double type) is stored $10000000000000000000000000101_{(2}$ in a 32-bit variable in two's complement is obtained:

0001 0000 0000 0000 0000 0000 0000 0101

In hexadecimal: 0x10000005

In a variable of type `int`, which uses complement to 2, the range of representation does allow to represent the number exactly this value

Exercise 13. In relation to the IEEE 754 standard, please answer the following questions:

a) In a 32-bit IEEE 754 representation, what is the number of values between number 8 and 9?
b) Is the number of values that can be represented for any interval [N, N+1] kept constant in this standard, being N an integer value? Reason your answer.
c) Where does a `float` type variable have more precision, in a 32-bit computer or in a 64-bit one? Give a reason for your answer.
d) Indicate in a reasoned and justified way if you can store exactly any 32-bit integer in two's complement in a `float` variable?

Solution:

a) In order to calculate the number of values between 8 and 9 it is necessary to represent both numbers in a floating point:

$8_{(10} = 1000_{(2} = 1,000 \times 2^3{}_{(2}$

Sign = 1
Exponent = 127+3 = $130_{(10} = 10000010_{(2}$
Mantissa = 000000…. 000000 (23 bits)

$9_{(10} = 1001_{(2} = 1{,}001 \times 2^3{}_{(2}$

Sign = 1
Exponent = 127+3 = $130_{(10} = 10000010_{(2}$
Mantissa = 001000…. 000000 (23 bits)

The representation of 8 es: 0 10000010 00000000000000000000000
The representation of 9 es: 0 10000010 00100000000000000000000

All you have to do is calculate the number of values between the two representations. The first bit (starting from the right) with a different value is the one at position 21, so the number of values is 2^{20}.

b) This value does not remain constant for any interval, since the resolution of the representation, i.e. the difference between two representable values is not kept constant for the IEEE 754 standard4.

c) A float variable uses the IEEE 754 32-bit standard, so its resolution does not depend on the word width of the computer used.

d) Any integer value cannot be accurately represented. Consider the value $2^{30} + 5$. This value can be stored in a 32-bit integer using two's complement, since the range of representation is $[-2^{31}, 2^{31} -1]$. When you store the value $2^{30} + 5$ in a float variable you get the following:
$2^{30} + 5 = 1000000000000000000000000000101_{(2} = 1.000000000000000000000000000101 \times 2^{30}{}_{(2}$

Only the first 23 bits of the fractional part can be stored in the mantissa, so the less significant bits are lost.

Exercise 14. In relation to the IEEE 754 standard, please answer the following questions:
a) Indicate the bit content of a double type variable when the value 17,25 is stored in it.
b) What happens when a double type variable is stored in a float type variable?

Solution:

a) A double type variable in IEEE 754 format occupies 64 bits, and is formed by 1 bit for the sign, 11 for the exponent and 52 for the mantissa (instead of the 1, 8 and 23 bits respectively of the 32-bit format).

$17{,}25_{10} = 16 + 1 + 0.25 = 2^4 + 2^0 + 2^{-2} = 10001{,}01_{(2}$
$10001{,}01_{(2} = 1{,}000101_{(2} \times 2^4$

Sign: 0 (positive)
Exponent = 4; The exponent stored is $4 + 2^{10} - 1 = 4 + 1023 = 1027_{10} = 10000000011_{(2}$
Mantissa: $000101000000…..00000000_{(2}$

In hexadecimal, 17,25 in IEEE 754 ss: 4031400000000000

b) That it is necessary to adapt the representation of the number, going from 64 bits to 32 bits. It is possible that:
- The number is unrepresentable (as it is the largest range) .
- That accuracy is lost (by dedicating less bits for the mantissa).

NOTE: You can see the result in http://babbage.cs.qc.edu/IEEE-754/

Exercise 15. In relation to the IEEE 754 standard, please answer the following questions:

a) In a 32-bit IEEE 754 representation, indicate in a reasoned manner the number of non-normalized values that can be represented.
b) Represent in the IEEE 754 double precision standard the value -20,5. Express the result in hexadecimal.

Solution:

a) A non-normalized value corresponds to an exponent (8 bits) whose value is 0 and a mantissa (23 bits) with a value other than zero. The number of elements that can be represented is, therefore, 2 (positive and negative)$\times$ 2^{23}-1 = $2(2^{23}$-1)

b) $-20{,}5_{(10} = 10100{,}1_{(2} = 1{,}01001 \times 24_{(2}$

Sign = 1
Exponent = 1023+4 = $1027_{(10}$ = $1024+3_{(10}$ = $10000000011_{(2}$
Mantissa = 01001…. 000000 (52 bits)

The representation of -20,5 is:

1 100 0000 0011 0100 1000000…….000000

In hexadecimal: 0xC034800000000000

Exercise 16. In relation to the IEEE 754 standard, please answer the following questions:

a) Given the number 0,6, in which of the formats (single precision or double precision) can the number 0,6 be represented exactly? Reason your answer.
b) Represent in the IEEE 754 double precision standard the value -18,25. Express the result in hexadecimal.

Solution:

a) The number 0,6 cannot be represented exactly in binary:

$0{,}6_{(10} = 0{,}100110011001\ldots.$

Therefore, it cannot be accurately represented in either single or double precision.

b) $-18{,}25_{(10} = 10010{,}01_{(2} = 1{,}001001 \times 24_{(2}$

Sign = 1
Exponent = 1023+4 = $1027_{(10}$ = $1024+3_{(10}$ = $10000000011_{(2}$
Mantissa = 00100100000…..0000000 (52 bits)

The representation of -20.5 is:

1 100 0000 0011 0010 0100 0000 0000 …. 0000