LAB WORK NO 2
THE INTERNAL DATA REPRESENTATION

1. Object of lab work

The purpose of this work is to understand the internal representation of different types of data in the computer. We will study and illustrate different ways of representing integer numbers (in magnitude and sign (MS), in complement to 1 (C1), complement to 2 (C2), in binary, decimal, packed and unpacked (BCD)), and real numbers (the IEEE short, long and temporary format)

2. Theoretical considerations

2.1 The representation of the integers in magnitude and sign (MS) in Complement to 1 and 2 (C1, C2)

Integer numbers can be represented on byte, on word (2 bytes) double words (4 bytes) or quadwords (8 bytes). For all representations, the most significant bit represents the sign bit, and the rest of the representation (the other bits) are used for representing in binary the number (the negative numbers have a different representation in the 3 representation forms)

There are two parts in representing the whole numbers: the sign bit and the absolute value. In all the tree forms of representation if the sign bit is 0 it represents a positive numbers and 1 in the sign bit represents negative numbers.

The field for the absolute value is represented like this:

- In the magnitude and sign (MS) representation, the module of the number is represented, so a number is represented putting 0 or 1 on the sign byte, according to the positive or negative value of the number and in the rest of the representation it is going to be used the value of the module in binary.
- In the complement to 1 (C1) representation, if the number is positive, the representation is identical as in magnitude and sign, the module of the number is represented and the sign bit is implicit 0. If the number is

negative then all the representation bits of the number in absolute value are complemented, the 1→0 and 0→1. The sign bit will be 1.

- In C2 representation, if the number is positive the representation is identical as in magnitude and sign and in complement on 1. If the number is negative, then the representation of the number in absolute value is complemented to 2, namely the representation of the module is subtracted from the value 2^{n+1} (n represents the number of bits to be represented, the sign bit will become 1). Another way of obtaining the representation in C2 of a negative number is by adding 1 to the representation in C1.

From the modes of representing the numbers in the three forms, results that the positive numbers have the same representation in magnitude and sign as in complement on 1 and complement on 2.

A greater attention must be accorded to the minimum space (the minimum number of bytes) on witch a number can be represented in the three modes of representation. For example when we want to find the minimum number of bytes on which the number 155 can be represented, we must defer to the fact that for representing the module there is one bit less (the sign bit) from the representation space. In this case, even if the value of its module fits on a byte (155 = 9Bh), the number can not be represented on a byte in either of the three representation modes, because the sign bit must be represented also, so at the interpretation of the 9Bh representation, the first bit being 1, the representation will be of a negative number instead of the desired number. To represent the number 155 we will need minimum 2 bytes (the representation is done on multiple of bytes), and the number will be represented like this: 009Bh, being positive in all the 3 representation modes.

Examples:

Represent on 4 bytes the following numbers: 157, 169, -157 and -169

157(D) = 1001 1101(B) = 9D(H)

So the representation in MS, C1 and C2 is 00 00 00 9D (H)

169(D) = 1010 1001(B) = A9(H)

So the representation in MS, C1 and C2 is: 00 00 00 A9 (H)

For –157, the module is represented firs (it has been computed above) and the result is:

MS: 1000 0000 0000 0000 0000 0000 1001 1101 (B) = 80 00 00 9D (H)

C1: 1111 1111 1111 1111 1111 1111 0110 0010 (B) = FF FF FF 62 (H)

C2: 1111 1111 1111 1111 1111 1111 0110 0011 (B) = FF FF FF 63 (H)
For –163 analogous:
MS: 80 00 00 A9 (H)
C1: FF FF FF 56 (H)
C2: FF FF FF 57 (H)

2.2. Representing the real numbers in the IEEE format

The IEEE standard of representing the real numbers proposes 3 modes of representing for real numbers:

- The short format on 4 bytes
- The long format on 8 bytes
- The temporary format on 10 bytes

The real numbers are represented in the short an long formats in the computer's memory, and the temporary format are found in loading the real numbers in the mathematic coprocessor.

All the three formats contain 3 parts:

Sign	Characteristic	**Mantissa**

- The sign bit S
- Characteristic C (on 8, 11, or 15 bits, for short, long and temporary format)
- Mantissa M (on 23, 52, or 64 bits)

For each representation:
S is 0 if the number is positive and 1 if the number is negative.
Characteristic = Exponent + 7Fh (or 3FFh for long IEEE and 3FFFh for temporary format)
In order to compute mantissa, first the number is represented in binary. This representation is normalized and written in the following format:
NO = 1.<binary digits> * $2^{exponent}$.

For the IEEE short and long format, the mantissa is formed by the digits after the decimal point, so the first 1 before the decimal point is not represented in the mantissa. For the temporary format all the digits of the mantissa are represented (including the leading 1).

Examples:

Represent in the IEEE short and long format the number 17,6(D).

The integer and the fractional part are converted separately, the results are:

The integer part: 17(D) = 11(H) = 1 0001(B)

The fractional part: 0,6(D) = 0,(1001)(B) (to be observed that the number is periodic), has infinite number of bits.

So 17,6(D) = 10001,(1001)(B)

The number is normalized: 17,6(D) = 10001,(1001)(B) = 1,0001(1001) * 2^4 (instead of 2^4 it would be more correct to have 10^{100}(B) because the notation was in binary, the fact that the characteristic is easer to calculate in hex then in binary can be a motivated excuse.

From this representation we can deduce the mantissa (the part after the decimal point, without that 1 before the point which is not represented by convention):

M = 0001(1001)(B).

After that the characteristic is calculated: C = exponent + 7F(H) = 4+7F(H) = 83(H) = 1000 0011(B)

The bit 0 for the sign will be written and we can write the representation:

0	1000 0011	00011001100110011001100
sign	charact.	mantissa

In order to write the representation in hex we will group 4 binary digits. Attention to the fact that grouping 4 digits will not correspond to the characteristic because of the sign bit that shifts a position. So the hex digits of the characteristic will not be found in the representation written in hex.

The result of the representation is: 41 8C CC CC(H).

In practice, a rounding will appear at the last bit, and the representation is:

41 8C CC CD(H).

Similarly we will represent –23,5(D):

23(D) = 17(H) = 1 0111(B)

0,5(D) = 0,1(B)

So 23,5(D) = 10111,1 = 1,01111 * 2^4 it results that M = 0111100000000000… (23 bits)

Characteristic = 7F(H) + 4(H) = 83(H)

The sign bit becomes 1.
The representation directly in hex is C1 BC 00 00(H).

In the following t you have the representation of a number in IEEE short format and you have to find the real number that is represented.

Example:
Given the representation 43 04 33 33 (H), you have to calculate the decimal value of the number represented
The representation in binary:
0100 0011 0000 0100 0011 0011 0011 0011
From here we deduce that:
The sign is 0
The characteristic is C = 1000 0110 (B) = 86(H)
The exponent is 86(H) – 7F(H) = 7 (H)
Mantissa M = 0000 1000 0110 0110…
The number is No=1,mantissa * $2^{exponent}$ = 1,0000 1000 0110…*2^{7}=
= 1000 0100,00110011…= 128 + 4 + 0.125 + 0.0625 + … ≈
≈ 132,1875 which approximates 132,2.

2.3. The representation of the numbers in packed BCD and unpacked BCD (Binary Coded Decimal)

Beside the modes of representation of the integers in MS, C1 and C2, there is the representation in Packed BCD and unpacked BCD.
In the Packed BCD representation one decimal digit is represented on 4 bits, so there are 2 decimal digits in a byte.
In the Unpacked BCD representation unpacked one decimal digit is represented on one byte (so we put 0 on the first 4 bits).
This representation modes are used for a better readability of the numbers from the programmers point of view, even if this is done by losing part of the available memory space (for packed BCD, only values 0-9 are used on 4 bits, and for unpacked BCD 4 more bits are left unused).
In order to compute operations with numbers represented in BCD, there are additional instructions for correcting the result after addition, multiplication, that will be studied in labs regarding instructions for arithmetic operations.
Example:
The number 3912(D) is going to be represented in BCD
- packed: 39 12(H) on 2 bytes;

- unpacked: 03 09 01 02(H) on 4 bytes.

3. Lab tasks

- Represent in MS, C1 and C2 the +35, -127 and 0.
- Represent in IEEE short format two real numbers.
- Given a representation in IEEE short format, find the represented number.