Computer Design — Data Representation

Computer Science (2nd year B.Sc.) — Seminar #1

## 1. Check-up

This part is designed as a check-up to allow you to determine if you understand the concepts already seen in “structures machine” (1st year). Please answer "True" or "False" to the following questions and include an explanation:

1.1. Depending on the context, the same sequence of bits can represent different things.
True! The same sequence of bits can be interpreted in different ways! This can represent an unsigned or signed number or even something else like a character. It all depends on the interpretation we give to this set.

1.2. In the two’s complement representation, it is possible to get an "overflow" error when adding two numbers of opposite signs.
False! "Overflow" errors occur only when the correct result of addition is outside the range of $[-2^{n-1}, 2^{n-1}-1]$, for an n-bit representation. Adding numbers of opposite signs will never produce numbers outside of this range.

1.3. If you interpret the negative numbers in the two’s complement representation as unsigned integers, then the values of those numbers would be smaller than the positive integers.
False! In the two’s complement representation, the most significant bit is always equal to 1 for a negative number. This means that all negative numbers will be larger than the positive numbers in an unsigned representation.

## 2. Unsigned integers

An unsigned integer $v$ can be written as the sum $v = \sum_{i=0}^{n-1} b^i d_i$, where $0 \leq d_i < b$ and $b$ is the base order. This mathematical notation simply means that to represent a given number in a base $b$, we use the units $b, b^2, ..$, etc. In the case of the binary, decimal and hexadecimal bases, $b$ will have the values 2, 10 and 16, respectively.

2.1. Convert the following numbers from their initial base to the other two bases. That is, If the initial basis is a binary representation, then give the equivalent representations in decimal and hexadecimal and so on.

a) $(10010011)_2 = (147)_{10} = (93)_{16}$

b) $(63)_{10} = (0011\ 1111)_2 = (3F)_{16}$

c) $(00100100)_2$ = $(36)_{10}$ = $(24)_{16}$

d) $(0)_{10}$ = $(0)_2$ = $(0)_{16}$

e) $(39)_{10}$ = $(0010\ 0111)_2$ = $(27)_{16}$

f) $(437)_{10}$ = $(0001\ 1011\ 0101)_2$ = $(1B5)_{16}$

g) $(0123)_{16}$ = $(0000\ 0001\ 0010\ 0011)_2$ = $(291)_{10}$

2.2. Convert the following numbers from the hexadecimal representation to the equivalent binary representation

a) $(BAD)_{16}$ = $(1011\ 1010\ 1101)_2$

b) $(F00D)_{16}$ = $(1111\ 0000\ 0000\ 1101)_2$

c) $(FACE)_{16}$ = $(1111\ 1010\ 1100\ 1110)_2$

d) $(0FF)_{16}$ = $(0000\ 1111\ 1111)_2$

## 3. Signed integers

In binary, the unsigned schema is not very well suited to represent, at the same time, positive and negative numbers. In this sense, several schemes have been invented to represent signed numbers, but we will limit ourselves to the two's complement encoding method.

- The most significant bit in a two's complement representation encodes a negative number. All other bits encode a standard positive integer. Thus, the value of an n-digit number in two's complement can be written as: $-2^{n-1}d_{n-1} + \sum_{i=0}^{n-2} 2^i d_i$.
- A trick to find the two's complement of a number: Flip all bits and add 1.
- Addition in the two's complement representation is performed in the same way as with unsigned numbers.
- The number zero has a single representation in two's complement: $(000\ldots0)_2$.

For questions 3.1 to 3.3, assume **an 8-bit width** and give your answers for both signed and unsigned cases. If you feel that the question does not have an answer, then indicate this with a "N/A" (meaning : Not Applicable).

3.1. What is the biggest integer? What will be the result if we add 1 to this number?

unsigned? 255 , 0

signed? 127 , -128

3.2. Give the encoding of the numbers $(0)_{10}$, $(1)_{10}$ and $(-1)_{10}$

| | | | |
|---|---|---|---|
| unsigned? | $(0000\ 0000)_2$ | , $(0000\ 0001)_2$ | , N / A |
| signed? | $(0000\ 0000)_2$ | , $(0000\ 0001)_2$ | , $(1111\ 1111)_2$ |

3.3. Give the representations of the numbers $(33)_{10}$ et $(-33)_{10}$

| | | |
|---|---|---|
| unsigned? | $(0010\ 0001)_2$ | , N / A |
| signed? | $(0010\ 0001)_2$ | , $(1101\ 1111)_2$ |

3.4. What is the smallest negative integer we can encode on 8 bits?

| | |
|---|---|
| in binary? | $(1000\ 0000)_2$ |
| in decimal? | -128 |

3.5. In binary on 16 bits, what's the encoding of the numbers $(33)_{10}$ and $(-33)_{10}$

a) $(33)_{10}$ = $(0000\ 0000\ 0010\ 0001)_2$

b) $(-33)_{10}$ = $(1111\ 1111\ 1101\ 1111)_2$

3.6. How do we move from an 8-bit representation to a 16-bit representation? Conversely, under what conditions can we switch from a 16-bit representation to an 8-bit representation?

In two's complement, to go from an 8-bit to a 16-bit representation we do a sign extension. That is, bit 7 of the byte is replicated in positions 8 through 15 in the 16-bit representation.

To truncate a 16-bit representation to an 8-bit representation, bits 8 through 15 must be the same as bit 7.

## 4. Arithmetic in binary

4.1. Still assuming 8-bit data width and the two's complement encoding format, compute the following additions and indicate the carry output. Give the decimal value of each result as well:

```
a)   14 + 59     = ( 0100 1001 )₂  = ( 73 )₁₀   , carry = 0

b)   59 + 80     = ( 1000 1011 )₂  = (-117)₁₀   , carry = 0

c)   59 + (-80)  = ( 1110 1011 )₂  = (-21 )₁₀   , carry = 0
```

```
d)  -59 + (-14)  = ( 1011 0111 )₂  = (-73 )₁₀   , carry = 1

e)  -59 + (-80)  = ( 0111 0101 )₂  = ( 117)₁₀   , carry = 1

f)   59 + (-59)  = ( 0000 0000 )₂  = (  0 )₁₀   , carry = 1
```

4.2. Some of the results obtained in the previous question are not as expected. What happened? What simple test can be used to detect these errors?
We have an "overflow" in b) and e). In two's complement representation, an overflow occurs when we add two values with the same sign, but the obtained result has an opposite sign. Thus, to detect an overflow, it is sufficient to compare the sign bits of the operands with that of the result.

The carry bit indicates an "overflow" for UNSIGNED operations only.

## 5. Data encoding

What is the smallest number of bits required to represent the following range values using any numerical encoding scheme (explain your answers!)

a) 0 to 256
*n* bits can be used to represent at most $2^n$ distinct values. Thus, 8 bits can represent $2^8 = 256$ values. However, the range 0 to 256 actually contains 257 numbers, so, we need 9 bits.

b) -7 to 56
Range of 64 numbers that can be encoded with 6 bits since $2^6 = 64$.

c) 64 to 127
Here, $127 - 64 + 1 = 64$ numbers to encode in total, so 6 bits.

d) -64 to -127
Same answer, 6 bits for 64 distinct values.

## 6. ASCII code

The **American Standard Code for Information Interchange** (ASCII) is a computer standard for character encoding that appeared in the 1960s. This standard defines 128 7-bit codes to represent characters that include synchronization codes (0 to 31), the digits 0 to 9, the 26 letters of the Latin alphabet in lowercase and uppercase, and mathematical and punctuation symbols (Figure below).

| Bits | | $b_6b_5b_4$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | **000** | **001** | **010** | **011** | **100** | **101** | **110** | **111** |
| $b_3b_2b_1b_0$ | **0000** | (NUL) | (DLE) | (SP) | 0 | @ | P | ` | p |
| | **0001** | (SOH) | (DC1) | ! | 1 | A | Q | a | q |
| | **0010** | (STX) | (DC2) | " | 2 | B | R | b | r |
| | **0011** | (ETX) | (DC3) | # | 3 | C | S | c | s |
| | **0100** | (EOT) | (DC4) | $ | 4 | D | T | d | t |
| | **0101** | (ENQ) | (NAK) | % | 5 | E | U | e | u |
| | **0110** | (ACK) | (SYN) | & | 6 | F | V | f | v |
| | **0111** | (BEL) | (ETB) | ' | 7 | G | W | g | w |
| | **1000** | (BS) | (CAN) | ( | 8 | H | X | h | x |
| | **1001** | (HT) | (EM) | ) | 9 | I | Y | i | y |
| | **1010** | (LF) | (SUN) | * | : | J | Z | j | z |
| | **1011** | (VT) | (ESC) | + | ; | K | [ | k | { |
| | **1100** | (FF) | (FS) | , | < | L | \ | l | \| |
| | **1101** | (CR) | (GS) | - | = | M | ] | m | } |
| | **1110** | (SOH) | (RS) | . | > | N | ^ | n | ~ |
| | **1111** | (SI) | (US) | / | ? | O | _ | o | (DEL) |

Figure 1. Table ASCII

6.1. From the table, give the 7-bit binary code of the characters 'a' and 'A'

‘a’ : $(110\ 0001)_2$ , ‘A’ : $(100\ 0001)_2$

6.2. Do the same for the character pairs ('b', 'B') and ('c', 'C'). What do you notice?

‘b’ : $(110\ 0010)_2$ , ‘B’ : $(100\ 0010)_2$ , ‘c’ : $(110\ 0011)_2$ , ‘C’ : $(100\ 0011)_2$

6.3. If an ASCII character represented a decimal digit, what value, in decimals, could the code of this character be? How do you infer the decimal value represented by this character?
The decimal characters '0' to '9' are encoded with the values $(011\ 0000)_2$ to $(011\ 1001)_2$ in binary. In decimal, this translates to the range of values 48 to 57. To find the decimal value associated with a decimal character, one solution is to subtract the value 48 from the ASCII code of that character (what’s the other solution?).