CS Basics - Exercises
Codification of numbers
SOLUTIONS

E. Benoist

Fall Term 2018-19

1 Encoding

1.1 Little endian

Exercise 1 *Compute in decimal numbers the following 32-bit integers in little endian convention.*

- *00H 00H 01H 00H = 00 01 00 00H*= $1 * 16^4 = 65'536$
- *AFH 01H 00H 00H = 1AFH* = $16^2 + 10 * 16 + 15 = 256 + 160 + 15 = 431$
- *00H 01H 00H 00H = 100H* = $16^2 = 256$
- *10H 10H 00H 00H = 1010H* = $16^3 + 16 = 4096 + 16 = 4112$
- *00H FFH 01H 00H = 1FF00H* = $16^4 + 15 * 16^3 + 15 * 16^2 = 130'816$

Exercise 2 *Write the value in memory of the following numbers in little endian on a 32-bit computer.*

- *35 =23H is written in memory:* 23H 00H 00H 00H
- *390 =186H is written in memory:* 86H 01H 00H 00H
- *450 = 1C2H is written in memory:* C2H 01H 00H 00H
- *1003 = 3*256+14*16+11 = 3EBH is written in memory:* EBH 03H 00H 00H

1.2 Signed integers

Exercise 3 *Using the two's complement notation write the representation of the following numbers in one byte. Write the result in hexadecimal.*

- *100 = 64 + 32 + 4 = 0000 0000 0110 0100 = 0064H*
- *67 = 64 + 2 + 1 = 0000 0000 0100 0011 = 0043H*
- *-10*

  First, we need the representation of 10 = 0000 0000 0000 1010B

  We negate this number (not on each bit) 1111 1111 1111 0101B

  And finally we add one 1111 1111 1111 0101B + 1= 1111 1111 1111 0110B

  The representation of -10 is 1111 1111 1111 0110B and FFF6H in hexadecimal
- *-5*

  5 is represented by 5 = 0000 0000 0000 0101B

  Negated it becomes 1111 1111 1111 1010B

  And if we add 1 it becomes:

  the representation of -5 is 1111 1111 1111 1011B and in hexadecimal FFFBH
- *-67*

  We use the Hexadecimal representation of 67 = 43H.

  We make a NOT on 0043H and obtain FFBCH (NOT 0043H = FFBCH)

  We add one and obtain FFBDH

  -67 is represented by the binary which corresponds to FFBDH
- *-130*

  130=0082H

  NOT 0082H = FF7DH

  FF7DH+1 = FF7EH

  The representation of -130 is FF7EH
- *-89*

  89=0059H

  NOT 0059H = FFA6H

  FFA6+1 = FFA7H

  The representation of -89 is FFA7H

- *-255*

  255=00FFH

  NOT 00FFH = FF00H

  FF00H+1 = FF01H

  The representation of -255 is FF01H

Exercise 4 *Execute the following additions in hexadecimal (using two's compelement notation):*

- *100 + (-67)*

```
   00 64 h
+  FF BD h
--------
(1) 00 21 h
```

  *21h = 2*16+1=33 (and 100-67=33)*

- *67 + (-5)*

```
   00 43 h
+  FF FB h
--------
(1) 00 3E h
```

  *3EH= 3*16+14 = 62*

- *(-67) + (-5)*

```
   FF BDh
+  FF FBh
------
(1) FF B8h
```

  *FF B8H represents a negative number (since F¿7 its first bit is 1) We negate its binary representation and add one: Not(FF B8H) + 1 = 00 47H + 1 = 48H=4*16+8 The result is -72*

1.3 Unsigned integers with bias

Exercise 5 *Compute the value of the following unsigned integers with the bias 127 (e.g -10 is represented by the number 117 and 20 is represented by the number 147). Write the number in hexadecimal form.*

- *0*

  0 + 127 = 127 = 0111 1111B = 7FH

- *10*

  10+127 = 137 = 128 + 8+2+1= 1000 1011 = 89H

- *120*

  120+127 = 247 = 1111 0111 = F7H

- *-20*

  -20+127=107 = 0110 1011 = 6BH

- *-15*

  -15+127 = 112 = 0111 0000 = 70H

1.4 Conversion of decimal to binary of floating point numbers

Exercise 6 *Convert the following decimal numbers into binary.*

- Example: *0.5*

  For a number smaller than 1, each time, we multiply the number by 2. If the result is larger than 1, we note 1 (for the result) and subtract 1. If the result is smaller than 1, we go on with the same number. At each step we multiply the resulting number by 2.

  ```
  0.5          | 0.
  1.0 (=0.5*2) | 1 -> 0.0
  0 -> End
  ```

  Result: ***0.5 = 0.1B***

- *0.125*

  ```
  0.125            | 0.
  0.250 (=0.125*2) | 0
  0.500 (=0.250*2) | 0
  1.0   (=0.500*2) | 1 -> 0.0
  0 -> End
  ```

  Result: ***0.125 = 0.001B***

- *0.4*

```
0.4 | 0.
0.8 | 0
1.6 | 1 -> 0.6
1.2 | 1 -> 0.2
0.4 | 0
0.8 | 0
1.6 | 1 -> 0.6
...
```

Result: ***0.4***$=0.01\overline{1001}$***B***

- *0.89*

```
0.89 | 0.
1.78 | 1 -> 0.78
1.56 | 1 -> 0.56
1.12 | 1 -> 0.12
0.24 | 0
0.48 | 0
0.96 | 0
1.92 | 1 -> 0.92
1.84 | 1 -> 0.84
1.68 | 1 -> 0.68
...
```

Result: ***0.89=0.111000111...B***

- *25.1*
  - *First we take care of 25, each step we divide 25 by 2 and note the reste.*

    ```
    25 | 1
    12 | 0
    6  | 0
    3  | 1
    1  | 1
    ```

    The integer part is 25 = 11001B
  - *Then we take care of the decimal part : 0.1*

    ```
    0.1 | 0.
    0.2 | 0
    0.4 | 0
    0.8 | 0
    1.6 | 1 -> 0.6
    1.2 | 1 -> 0.2
    0.4 | 0
    ```

```
0.8 | 0
1.6 | 1 -> 0.6
...
```

The decimal part is 0.1=$0.0001\overline{1001}B$

Result: ***25.1*** $= 11001.0001\overline{1001}\boldsymbol{B}$

- *9.90*

  Integer Part: 9 = 1001B = (8+1)

  Then we take care of the decimal part : 0.1

```
0.9 | 0.
1.8 | 1 -> 0.8
1.6 | 1 -> 0.6
1.2 | 1 -> 0.2
0.4 | 0
0.8 | 0
1.6 | 1 -> 0.6
...
```

The decimal part is 0.9=$0.11\overline{1001}B$ *The result:* ***9.90*** $=1001.11\overline{1001}\boldsymbol{B}$

1.5 Floating point numbers

Exercise 7 *Compute the representation of the following numbers on 32 bits (write it in bits, then in hexadecimal notation):*

- Example:*0.5*

  Sign bit is = 0;

  *0.5 = 1.00000000 * $2^{-1}B$*

  Mantissa is 1.00000000000000000000000, but without its first one (hidden bit) it becomes 00000000000000000000000

  Exponent is -1, and is encoded -1+127 = 126 = 0111 1110B

  Encoding of 0.5 in float is

```
0 0111 1110 00000000000000000000000
```

  This gives the binary representation: ***0011'1111'0000'0000'0000'0000'0000'0000B***
  This gives the Hexadecimal representation: ***3F 00 00 00 H***

- *12*

  12= 1100B= 1100.00000000000000000000B=1.10000000000000000000$*2^3$

Mantissa is 1.10000000000000000000000, and without the hidden bit it is encoded 10000000000000000000000

The exponent is 3 and is encode 127+3 = 130 = 1000 0010B

Encoding of 12 in float is 0 1000 0010 10000000000000000000000 B

This gives the binary representation: ***0100'0001'0100'0000'0000'0000'0000'0000 B***

This gives the Hexadecimal representation: ***41 40 00 00 H***

- *34.25*

  34 is 100010

  and 0.25 = $1/2^2$ *= 0.01b*

  *34.25 = 100010.01B= 1.0001001B**2^5

  Mantissa (without hidden bit) 0001001

  Exponent = 5, which is encoded (with bias 127) as 1000 0100

  34.25 is encoded:

  ```
  0 1000 0100 00010010000000000000000 B
  ```

  This gives the binary representation: ***0100'0010'0000'1001'0000'0000'0000'0000 B*** *This gives the Hexadecimal representation:* ***42 09 00 00 H***

- *0.1*

  0.1 is 0.0001100110011001100110011001100110011...B (0011 repeated infinitely)

  *0.1 = 1.100110011... ** 2^{-4}

  0.1 is represented as a float by

  ```
  0 0111 1011 10011001100110011001100
  ```

  This gives the binary representation: ***0011'1101'1100'1100'1100'1100'1100'1100 B***

  This gives the hexadecimal representation: ***3D CC CC CC H***

- *-10.98*

  10 = 1010 B = (2+8)

  ```
  0.98 | 0.
  1.96 | 1 -> 0.96
  1.92 | 1 -> 0.92
  1.84 | 1 -> 0.84
  ```

```
1.68 | 1 -> 0.68
1.36 | 1 -> 0.36
0.72 | 0
1.44 | 1 -> 0.44
0.88 | 0
1.76 | 1 -> 0.76
1.52 | 1 -> 0.52
1.04 | 1 -> 0.04
0.08 | 0
0.16 | 0
0.32 | 0
0.64 | 0
1.28 | 1 -> 0.28
0.56 | 0
1.12 | 1 -> 0.12
0.24 | 0
0.48 | 0
```

-10.98 = -1010.1111101011100010100....B

-10.98 = -1.01011111010111000010100.... x 2^3B 3 is the exponent and is represented by 3+127 = 130 = 1000'0010 B

The sign is negative and is therefore 1. The representation of -10.98 is ***1 1000 0010 1010 1111 1010 1110 0001 0100 B***

This gives the binary number: ***1100'0001'1010'1111'1010'1110'0001'0100 B***
This gives the hexadecimal number: ***C1 2F AE 14 H***

Exercise 8 *Compute the biggest number that can be represented by a 32 bit float.*
Hint : $(1/2) + (1/2^{2)} + (1/2^3) + (1/2^4) + ... + (1/2^n) = (2^n - 1)/(2^n) = 1 - (1/2^n)$
*The bigest number is positive has an exponent FEH (FFH is forbidden) and only 1's in the mantissa. FEH=15*16+14=254. So the exponent is 254-127 = 127*

```
0 1111 1110 1111 1111 1111 1111 1111 1111
```

it represents the sum:

$$(1+1/2+1/4+1/8+...+1/2^{23})*2^{127} = (1+1-1/2^{23})*2^{127} = (2-2^{-23})*2^{127} = 2^{128}-2^{104} \quad (1)$$