# BITWISE OPERATION EXERCISES

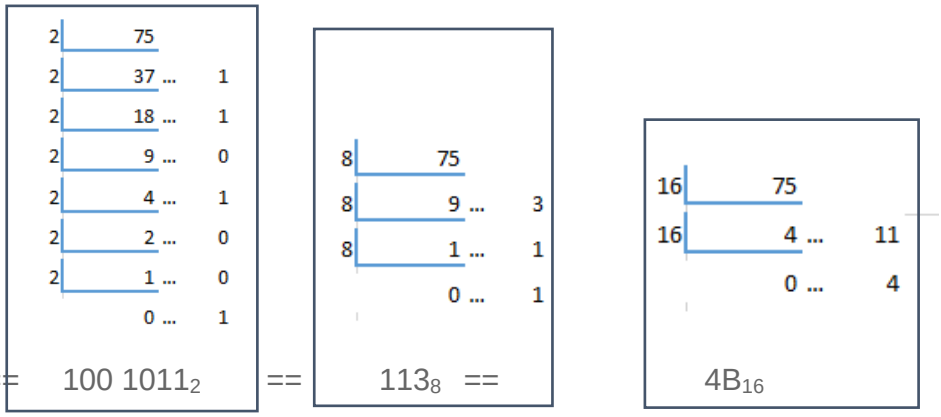
## Table of Contents

# REVIEW BASE CONVERSION

**Common Bases :**

- Decimal == base-10. Range: 0,…, 9
- Hexadecimal == base-16. Range : 0 … A, B, C, D, E, F
- Octal == base-8, Range: 0, …, 7
- Binary == base-2 . Range: 0, 1

**Convert** **base-10 to base-2** **base-10 to base-8** **base-10 to base-16**




So $75_{10}$ == $100\ 1011_2$ == $113_8$ == $4B_{16}$

**Convert other bases back to base-10**

Binary to Decimal

$1\ 1\ 1\ 1$

$2^3\ 2^2\ 2^1\ 2^0$ $= 15_{10}$

Octal to Decimal

$1\ 1\ 2\ 1$

$8^3\ 8^2\ 8^1\ 8^0$ $= 593_{10}$

Hex to Decimal

$1\ 2\ 1\ 1$

$16^3\ 16^2\ 16^1\ 16^0$ $= 4625_{10}$

**Tips:**

- A single hex digits == 4 bits
- When converting binary to hex, first break the bits into 4 bits chunk.

e.g. Take $1011001010_2$ , break it into $10\ 1100\ 1010_2$.

Dec. value for each chunk … … 2 12 $10_{10}$

Hex. value = 2 C $A_{16}$

∴ $10\ 1100\ 1010_2$ == $2CA_{16}$

## PRACTICE EXERCISES

1) Decimal (base-10) to Hexadecimal (base-16) and Binary (base-2)

| Decimal (base-10) | Hexadecimal (base-16) | Binary (base-2) |
|---|---|---|
| 12 | | |
| 14 | | |
| 13 | | |
| 15 | | |
| 16 | | |
| 127 | | |
| 255 | | |
| 256 | | |
| 1023 | | |
| 1024 | | |
| 2047 | | |
| 2048 | | |
| 65535 | | |
| 65536 | | |

2) Binary (base-2) to Hexadecimal (base-16) and Decimal (base-10)

| Binary (base-2) | Hexadecimal (base-16) | Decimal (base-10) |
|---|---|---|
| 00001 | | |
| 00011 | | |
| 00111 | | |
| 01001 | | |
| 110011 | | |
| 1001110 | | |
| 00001001 | | |
| 10001001 | | |
| 01001001 | | |
| 10101000 | | |

3) (If you have yet done this before, you need to work on this one.)
Write a function to allow user to print out the based-8, based-16 and base-2 representation of a number. For based-2, the display must include a simple "-" every single 4 bits.

Example input: 56
Example output: Octet of 56 = 70
Hex of 56 = 38
Binary of 56 = 0011-1000

# BITWISE OPERATIONS EXERCISES

Review the Text book – chapter 20 – low level programming

- Bitwise operations
- Bit-Fields in Structures
- Using Unions to Provide Multiple Views of Data
- Using Pointers as Addresses
- The volatile Type Qualifier

1) About Parity bit .
   When data is transmitted from one device to another, one among many error detection and cryptography methods involved is to check the parity bit. Parity of a number can be odd or even. If the number contains odd number of 1-bits, it is a odd-parity; otherwise, "even parity".:

2) Write a program to ask user for a number, and you will display it's value in base-10, base-16, and base-2, and its parity bit.

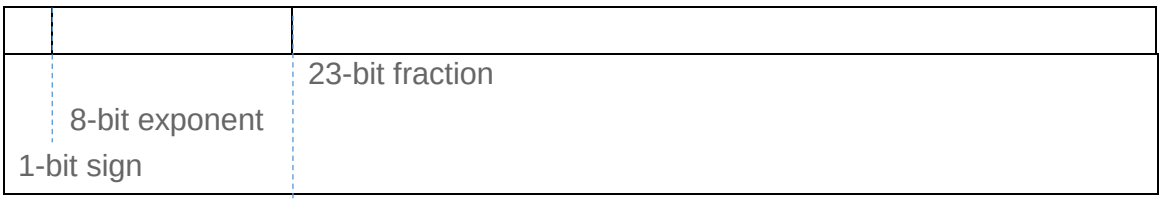
   Example input: You entered number: 13
   Output ; 13 (Dec) == D (Hex) == 1101 (Binary). Odd Parity

   Example input: You entered number: 80
   Output ; 80 (Dec) == 80 (Hex) == 0101 0000 (Binary). Even Parity

3) Write 4 functions to perform +, -, x, /, % with using ONLY bitwise operations.

4) Write a swap function without creating a separate variables to hold the temp.

5) IEEE floating-point standard: a floating point number consist of




   Design a structure type that occupies 32 bits, with bit-field members corresponding to the sign, exponent and fraction.
   Check out whether your compiler is big-endian or little-endian border.

6) Given this struct below and printf statement, some compliers may display 1, or -1. What do you need to do to avoid this from happening.

```
struct {
Int flag : 1;
} s;
…
s.flag = 1;
printf("%d\n", s.flag);
```