TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO BÀI TẬP LỚN

CHUYỂN ĐỔI CƠ SỐ

(BASE CONVERTER)

Các thành viên trong nhóm:

1. Lê Trọng Anh Tú MSSV: 20127091
2. Phan Tuấn Khải MSSV: 20127524

Giáo viên hướng dẫn: Ths. Thái Hùng Văn

Mục lục

Listings

1 THÔNG TIN CHUNG

Người hướng dẫn:

– Ths. Thái Hùng Văn (Khoa Công nghệ Thông tin)

Nhóm sinh viên thực hiện:

1. Lê Trọng Anh Tú MSSV: 20127091
2. Phan Tuấn Khải MSSV: 20127524

Môn học: Hệ Thống Máy Tính - **Lớp:** 20CLC07

Thời gian thực hiện: Từ 2/11/2021 đến 12/11/2021

2 NỘI DUNG THỰC HIỆN

2.1 Giới thiệu

Chuyển đổi cơ số là chuyển đổi biểu diễn của một số ở dạng biểu diễn này sang dạng biểu diễn khác nhưng vẫn thể hiện cùng một giá trị. Để làm rõ hơn trong việc chuyển đổi cơ số nhóm chúng tôi thực hiện bài tập lớn về vấn đề chuyển đổi cơ số từ hệ thập phân sang nhị phân và ngược lại.

2.2 Mục tiêu

Với yêu cầu đề bài, nhóm chúng tôi thực hiện viết chương trình cho phép chuyển đổi số thực ở dạng thập phân sang biểu diễn nhị phân 32/64 bit, số nguyên sang dạng 8/16/32 bit bao gồm cả có dấu và không dấu. Tương tự là chuyển ngược từ 32/64 bit về số thực và 8/16/32 bit về số nguyên có dấu và không dấu.

2.3 Phân chia công việc và mức độ hoàn thành

Số thứ tự	Nội dung	Người thực hiện	Mức độ hoàn thành
1	Chuyển đổi số thực sang 32/64 bit	Phan Tuấn Khải	100%
2	Chuyển đổi số nguyên sang 8/16/32 bit	Phan Tuấn Khải	100%
3	Chuyển đổi dãy bit 8/16/32 bit sang số nguyên	Lê Trọng Anh Tú	100%
4	Chuyển đổi dãy bit 32/64 bit sang số thực	Lê Trọng Anh Tú	100%

2.4 Một số lưu ý

2.4.1 Biểu diễn

Trong bài tập này nhóm chúng tôi thực hiện biểu diễn số thực 32/64 bit theo định dạng của tổ chức IEEE và số nguyên có dấu được biểu diễn ở dạng bù 2.

Do quá trình làm tròn của ngôn ngữ nên khi biểu diễn từ số thực sang bit có thể làm lệch kết quả nhưng tạm có thể chấp nhận độ sai lệch.

2.4.2 Mã nguồn

Chương trình được nhóm chúng tôi viết bằng ngôn ngữ C/C++.

Các mã nguồn được viết trong báo cáo chỉ là minh họa về thuật toán và cần một số hàm phụ để hoạt động. Tất cả mã nguồn được gửi kèm trong file.

2.4.3 Cài đặt

Cài đặt chương trình được biên dịch bằng trình biên dịch GNU C++
Compiler: g++ version 8.1.0 với lệnh biên dịch:

g++ *.cpp *.h -std=c++17 -o main.exe

2.5 Mô tả thuật toán

2.5.1 Chuyển đổi số thực sang dạng 32 bit

- **Mô tả thuật toán**

  Một số thực bất kỳ nào đều có thể biểu diễn được dưới dạng: $\pm 1.M * B^E$ trong đó M là phần định trị, B là cơ số đang xét và E là số mũ.
  Như đã đề cập trong phần lưu ý, nhóm chúng tôi thực hiện chuyển đổi theo chuẩn IEEE và cách thực hiện như sau
  Chuẩn IEEE biễn diễn số thực R ở dạng nhị phân:

$$R = (-1)^S * (1.f_1 f_2 ... f_n) * 2^E = (-1)^S * (1 + f_1 * 2^{-1} + f_2 * 2^{-2} + ... + f_n * 2^{-n}) * 2^E.$$

  Cần biểu diễn số thực R thành dạng biểu diễn nhị phân theo định dạng: $(1.f_1 f_2 ... f_n) * 2^E$
  Ví dụ: $R = -5.25_{10} = -101.01_2$ ta chuẩn hóa thành $-1.0101 * 2^2$
  Ta cần xác định bit dấu S, như số ở ví dụ trên ta thấy R là số âm do đó bit dấu sẽ là bit 1.
  Tiếp đến là phần mũ được biểu diễn bằng 8 bit, ở đây phần mũ được biểu diễn theo kiểu quá 127 vì vậy phần mũ = 127 + E.
  Cuối cùng là phần định trị M, được biểu diễn bằng 23 bit là phần $f_1 f_2 ... f_n$.
  Cuối cùng ta có được chuỗi biểu diễn 32 bit của 1 số thực với bit đầu tiên là bit dấu, 8 bit tiếp theo thể hiện phần mũ và cuối cùng là 23 bit cuối trong định trị.[1]

- **Mã nguồn**

```
vector<int> DoubleToBin::to32Bit(const double& b) {
    long long _b = (long long)b;
    vector<int> getBit = IntegerToBin::toBit(_b);
    int need = 24 - getBit.size();
    _b =(long long) (b * Math::mpow(2, need));
    vector<int> man = IntegerToBin::toBit(_b);
    int e = 127 + 23 - need;
    vector<int> res;
    res.push_back(b < 0 ? 1 : 0);
    res += IntegerToBin::to8Bit(e);
    res += vector<int>(man.begin() + 1, man.end());

    if(res.size() < 32)
        res += vector<int>(32 - res.size(), 0);

    return res;
}

```

Listing 1: Double To 32 bit

2.5.2 Chuyển đổi số thực sang dạng 64 bit

- **Mô tả thuật toán**

  Tương tự như biểu diễn 32 bit nhưng 64 bit cần 11 bit biểu diễn phần mũ và biểu diễn ở dạng quá 1023, phần định trị được biểu diễn bằng 52 bit và cách làm hoàn toàn tương tự như 32 bit.

  Cách tính các phần S, E, M tương tự như cách tính S, E, M của biểu diễn 32 bit nên chúng tôi không giải thích chi tiết thêm.

- **Mã nguồn**

```
vector<int> DoubleToBin::to64Bit(const double& b) {
    unsigned long long _b = (unsigned long long)b;
    vector<int> getBit = IntegerToBin::toBit(_b);
    int need = 53 - getBit.size();
    _b = (unsigned long long)(b * Math::mpow(2, need));
    vector<int> man = IntegerToBin::toBit(_b);
    int e = 1023 + 52 - need;
    vector<int> res;
    res.push_back(b < 0 ? 1 : 0);
    res += IntegerToBin::to11Bit(e);
    res += vector<int>(man.begin() + 1, man.end());

    if(res.size() < 64)
        res += vector<int>(64 - res.size(), 0);

    return res;
}

```

Listing 2: Double To 64 bit

2.5.3 Chuyển đổi số nguyên sang dạng 8 bit

- **Mô tả thuật toán**

  Một số nguyên N bất kỳ của thể biểu diễn bằng tổng của các lũy thừa 2 do đó muốn biểu diễn số nguyên N nào đó dưới dạng nhị phân ta chỉ cần tìm các lũy thừa của 2 thỏa mãn $N = 2^{k_1} + 2^{k_2} + .. + 2^{k_n}$ từ đó ta có ý tưởng thuật toán như sau:

  Chia N cho 2 lưu lại kết quả là K_1 và R_1 (K_1 là phần nguyên của phép chia và R_1 là phần dư của phép chia nguyên).

  Tiếp tục lấy K_1 chia cho 2 lưu lại kết quả K_2 và R_2.

  Tiếp tục thực hiện cho đến khi $K_n = 0$ khi đó ta sẽ có dãy bit biểu diễn $N = R_nR_{n-1}...R_2R_1$ ở hệ nhị phân.[2]

  Vì cần biểu diễn ở 8 bit do đó nếu $n < 8$ ta thêm vào phía trước R_n các số 0 sao cho đủ 8 bit.

  Ví dụ: Biểu diễn 8 bit của số 7 ta làm như sau

  - $7/2 = 3$ dư 1
  - Lấy $3/2 = 1$ dư 1
  - Lấy $1/2 = 0$ dư 1
  - Tới đây ta dừng lại và kết quả là 111 nhưng do biểu diễn 8 bit nên ta cần thêm vào phía trước 5 số 0 để có đủ 8 bit

– Vậy kết quả cuối cùng là 0000'0111

- **Mã nguồn**

```
vector<int> IntegerToBin::to64Bit(const long long &d) {
    vector<int> res;
    res.clear();
    long long _d = abs(d);
    for (; _d; _d >>= 1) {
        res.push_back((_d & 1 ? 1 : 0));
    }
    for (int i = res.size(); i <= 63; ++i)
        res.push_back(0);

    reverse(res.begin(), res.end());

    if(res.size() == 0)
        res = vector<int>(64, 0);

    return res;
}

vector<int> IntegerToBin::to8Bit(const long long &d) {
    vector<int> res = to64Bit(d);

    return vector<int>(res.begin() + 64 - 8, res.end());
}

```

Listing 3: Integer To 8 bit

2.5.4 Chuyển đổi số nguyên sang dạng 16 bit

- **Mô tả thuật toán**

  Thực hiện tương tự dạng 8 bit và thêm vào các số 0 phía trước sao cho đủ 16 bit. Cách làm chi tiết như biểu diễn 8 bit nên chúng tôi không giải thích thêm cách làm chi tiết.

- **Mã nguồn**

```
vector<int> IntegerToBin::to64Bit(const long long &d) {
    vector<int> res;
    res.clear();
    long long _d = abs(d);
    for (; _d; _d >>= 1) {
        res.push_back((_d & 1 ? 1 : 0));
    }
    for (int i = res.size(); i <= 63; ++i)
        res.push_back(0);

    reverse(res.begin(), res.end());

    if(res.size() == 0)
        res = vector<int>(64, 0);

    return res;
}
```

```
18
19      vector<int> IntegerToBin::to16Bit(const long long &d) {
20          vector<int> res = to64Bit(d);
21
22          return vector<int>(res.begin() + 64 - 16, res.end());
23      }
24
```

Listing 4: Integer To 16 bit

2.5.5 Chuyển đổi số nguyên sang dạng 32 bit

- **Mô tả thuật toán**

  Bằng cách thực hiện tương tự dạng 8 bit và 16 bit và thêm vào phía trước các số 0 sao cho đủ 32 bit. Cách làm chi tiết như biểu diễn 8 bit và 16 bit nên chúng tôi không giải thích thêm cách làm chi tiết.

- **Mã nguồn**

```
1       vector<int> IntegerToBin::to64Bit(const long long &d) {
2           vector<int> res;
3           res.clear();
4           long long _d = abs(d);
5           for (; _d; _d >>= 1) {
6               res.push_back((_d & 1 ? 1 : 0));
7           }
8           for (int i = res.size(); i <= 63; ++i)
9               res.push_back(0);
10
11          reverse(res.begin(), res.end());
12
13          if(res.size() == 0)
14              res = vector<int>(64, 0);
15
16          return res;
17      }
18
19      vector<int> IntegerToBin::to32Bit(const long long &d) {
20          vector<int> res = to64Bit(d);
21
22          return vector<int>(res.begin() + 64 - 32, res.end());
23      }
24
```

Listing 5: Integer To 32 bit

2.5.6 Chuyển đổi số nguyên có dấu sang dạng 8 bit

- **Mô tả thuật toán**

  Như đã đề cập ở phần lưu ý, nhóm chúng tôi thực hiện chuyển đổi ở dạng bù 2.
  Để biểu diễn được một số nguyên có dấu N dầu tiên ta cần tìm được dạng biểu diễn 8 bit của $|N|$.
  Tiếp theo ta tiến hành đảo toàn bộ dãy bit trên và cuối cùng là cộng thêm 1 vào kết quả.[3]
  Ví dụ: Muốn tìm dạng biểu diễn 8 bit theo bù 2 của số -7 ta làm như sau

  – $|-7| = 7 = 0000'0111_2$

- Đảo toàn bộ dãy bit ta được kết quả: $0000'0111_2 = 1111'1000_2$
- Cuối cùng cộng 1 ta có kết quả: $1111'1000_2 + 1_2 = 1111'1001_2$

- **Mã nguồn**

```
vector<int> IntegerToBin::to2s8Bit(const long long &d) {
    vector<int> a = to8Bit(d);
    if (d < 0) {
        a = ~a; //a is a vector type that needs overloaded operator ~
        a++; //a is a vector type that needs overloaded operator +
    }
    //This condition to check the performance bit sequence
    if ((long long)(1 << 8) - 1 <= abs(d))
        throw runtime_error("over size of 8bits");

    return a;
}

```

Listing 6: Integer To 2's Complement 8 bit

2.5.7 Chuyển đổi số nguyên có dấu sang dạng 16 bit

- **Mô tả thuật toán**
  Vẫn như cách làm khi biểu diễn bù 2 của 8 bit nên chúng tôi không giải thích chi tiết mà chỉ đi nhanh qua cách làm như sau:
  Đầu tiên cần tìm được dạng biểu diễn 16 bit và tiến hành đảo bit sau đó cộng 1 và có được kết quả.

- **Mã nguồn**

```
vector<int> IntegerToBin::to2s16Bit(const long long &d) {
    vector<int> a = to16Bit(d);
    if (d < 0) {
        a = ~a; //a is a vector type that needs overloaded operator ~
        a++; //a is a vector type that needs overloaded operator +
    }
    //This condition to check the performance bit sequence
    if ((long long)(1 << 16) - 1 <= abs(d))
        throw runtime_error("over size of 16bits");

    return a;
}

```

Listing 7: Integer To 2's Complement 16 bit

2.5.8 Chuyển đổi số nguyên có dấu sang dạng 32 bit

- **Mô tả thuật toán**
  Vẫn như cách làm khi biểu diễn bù 2 của 8 bit và 16 bit nên chúng tôi không giải thích chi tiết mà chỉ đi nhanh qua cách làm như sau:
  Đầu tiên cần tìm được dạng biểu diễn 32 bit và tiến hành đảo bit sau đó cộng 1 và có được kết quả.

- **Mã nguồn**

```
vector<int> IntegerToBin::to2s32Bit(const long long &d) {
    vector<int> a = to32Bit(d);
    if (d < 0) {
        a = ~a; //a is a vector type that needs overloaded operator ~
        a++; //a is a vector type that needs overloaded operator +
    }
    //This condition to check the performance bit sequence
    if ((long long)(1ULL << 32) - 1 <= abs(d))//Warning: (1 << 32) overflow
        throw runtime_error("over size of 32bits");

    return a;
}

```

Listing 8: Integer To 2's Complement 32 bit

2.5.9 Chuyển đổi dãy 8 bit sang số nguyên không dấu

- **Mô tả thuật toán**

  Từ một dãy 8 bit có định dạng:

  $$x_7x_6x_5x_4x_3x_2x_1x_0$$

  Ta có thể chuyển đổi về số nguyên có dấu như sau:

  $SN = x_0 * 2^0 + x_1 * 2^1 + x_2 * 2^2 + x_3 * 2^3 + x_4 * 2^4 + x_5 * 2^5 + x_6 * 2^6 + x_7 * 2^7$[4]

  Và cuối cùng đáp án của dãy bit khi chuyển sang số nguyên là SN.

- **Mã nguồn**

```
unsigned int BinaryToInteger::convertToUnsignedInteger() {
    unsigned int res = 0;
    int idx = 0;
    // assume that bit sequence have length is 8
    for (int i = _numberOfBit - 1; i >= 0; i--) {
        if (_binarySequence[i] == '1') {
            res += pow(2, idx);
        }
        idx++;
    }

    return res;
}

```

Listing 9: 8 bit To Unsigned Integer

2.5.10 Chuyển đổi dãy 16 bit sang số nguyên không dấu

- **Mô tả thuật toán**

  Từ một dãy 16 bit có định dạng:

  $$x_{15}x_{14}x_{13}x_{12}x_{11}x_{10}x_9x_8x_7x_6x_5x_4x_3x_2x_1x_0$$

Ta có thể chuyển đổi về số nguyên có dấu như sau:

$SN = x_0 * 2^0 + x_1 * 2^1 + x_2 * 2^2 + x_3 * 2^3 + x_4 * 2^4 + x_5 * 2^5 + x_6 * 2^6 + x_7 * 2^7 + x_8 * 2^8 + x_9 * 2^9 + x_{10} * 2^{10} + x_{11} * 2^{11} + x_{12} * 2^{12} + x_{13} * 2^{13} + x_{14} * 2^{14} + x_{15} * 2^{15}$

Và cuối cùng đáp án của dãy bit khi chuyển sang số nguyên là SN.

- **Mã nguồn**

```
unsigned int BinaryToInteger::convertToUnsignedInteger() {
    unsigned int res = 0;
    int idx = 0;
    // assume that bit sequence have length is 16
    for (int i = _numberOfBit - 1; i >= 0; i--) {
        if (_binarySequence[i] == '1') {
            res += pow(2, idx);
        }
        idx++;
    }

    return res;
}

```

Listing 10: 16 bit To Unsigned Integer

2.5.11 Chuyển đổi dãy 32 bit sang số nguyên không dấu

- **Mô tả thuật toán**

Cách làm tương tự như chuyển đôi dãy 8 bit và 16 bit, chúng tôi sẽ không ghi chi tiết thêm nữa.

- **Mã nguồn**

```
unsigned int BinaryToInteger::convertToUnsignedInteger() {
    unsigned int res = 0;
    int idx = 0;
    // assume that bit sequence have length is 32
    for (int i = _numberOfBit - 1; i >= 0; i--) {
        if (_binarySequence[i] == '1') {
            res += pow(2, idx);
        }
        idx++;
    }

    return res;
}

```

Listing 11: 32 bit To Unsigned Integer

2.5.12 Chuyển đổi dãy 8 bit sang số nguyên có dấu

- **Mô tả thuật toán**

Một dãy 8 bit biểu diễn một số nguyên có dấu có dạng như sau:

$$x_7x_6x_5x_4x_3x_2x_1x_0$$

Trong đó x_7 là bit dấu.

Cách chuyển đổi như sau:

$SN = (-1)^{x_7}(x_0 * 2^0 + x_1 * 2^1 + x_2 * 2^2 + x_3 * 2^3 + x_4 * 2^4 + x_5 * 2^5 + x_6 * 2^6)$[5]

Và cuối cùng đáp án của dãy bit khi chuyển sang số nguyên là SN.

- **Mã nguồn**

```
int BinaryToInteger::convertToSignedInteger() {

    //assume that bit sequence have length is 8
    int res = (_binarySequence[0] == '1') ? -pow(2, _numberOfBit - 1) : 0;
    int idx = 0;

    for (int i = _numberOfBit - 1; i > 0; i--) {
        if (_binarySequence[i] == '1') res += pow(2, idx);
        idx++;
    }

    return res;
}

```

Listing 12: 8 bit To Signed Integer

2.5.13 Chuyển đổi dãy 16 bit sang số nguyên có dấu

- **Mô tả thuật toán**

Một dãy 16 bit biểu diễn số nguyên có dấu có định dạng như sau:

$$x_{15}x_{14}x_{13}x_{12}x_{11}x_{10}x_9x_8x_7x_6x_5x_4x_3x_2x_1x_0$$

Trong đó x_{15} là bit dấu

Ta có thể chuyển đổi về số nguyên có dấu như sau:

$SN = (-1)^{x_{15}} * (x_0 * 2^0 + x_1 * 2^1 + x_2 * 2^2 + x_3 * 2^3 + x_4 * 2^4 + x_5 * 2^5 + x_6 * 2^6 + x_7 * 2^7 + x_8 * 2^8 + x_9 * 2^9 + x_{10} * 2^{10} + x_{11} * 2^{11} + x_{12} * 2^{12} + x_{13} * 2^{13} + x_{14} * 2^{14})$

Và cuối cùng đáp án của dãy bit khi chuyển sang số nguyên là SN.

- **Mã nguồn**

```
int BinaryToInteger::convertToSignedInteger() {

    //assume that bit sequence have length is 16
    int res = (_binarySequence[0] == '1') ? -pow(2, _numberOfBit - 1) : 0;
    int idx = 0;

    for (int i = _numberOfBit - 1; i > 0; i--) {
        if (_binarySequence[i] == '1') res += pow(2, idx);
        idx++;
    }
```

```
11
12          return res;
13      }
14
```

Listing 13: 16 bit To Signed Integer

2.5.14 Chuyển đổi dãy 32 bit sang số nguyên có dấu

- **Mô tả thuật toán**

  Cách làm tương tự như chuyển đôi dãy 8 bit và 16 bit, chúng tôi sẽ không ghi chi tiết thêm nữa.

- **Mã nguồn**

```
1   int BinaryToInteger::convertToSignedInteger() {
2
3       //assume that bit sequence have length is 16
4       int res = (_binarySequence[0] == '1') ? -pow(2, _numberOfBit - 1) : 0;
5       int idx = 0;
6
7       for (int i = _numberOfBit - 1; i > 0; i--) {
8           if (_binarySequence[i] == '1') res += pow(2, idx);
9           idx++;
10      }
11
12      return res;
13  }
14
```

Listing 14: 32 bit To Signed Integer

2.5.15 Chuyển đổi dãy 32 bit sang số thực

- **Mô tả thuật toán** Ta có dãy 32 bit biểu diễn một số thực có định dạng như sau:

  seeeeeeemmmmmmmmmmmmmmmmmmmmmmm

  Trong đó s là bit đầu tiên và là bit dấu, tiếp theo gồm 8 số e và là 8 bit mũ, cuối cùng là 23 số m là 23 bit định trị.

  Để chuyển được một dãy 32 bit về số thực trước hết ta cần tìm được các thành phần S, E và M trong đó S là bit dấu, E là bit mũ và M là bit định trị.

  Sau đó tính phần mũ E bằng cách đổi dãy 8 bit ở phần mũ sang số nguyên và trừ đi 127 do đang được biểu diễn ở kiểu quá 127.

  Tiếp theo là tính phần định trị M, cách tính dãy bit đã được ghi chi tiết ở trên nên chúng tôi cũng sẽ không đi chi tiết thêm.

  Và cuối cùng đáp án khi chuyển đổi dãy 32 bit sang số thực là: $F = (-1)^S * (2^E + M)$[6]

- **Mã nguồn**

```
float BinaryToFloatingPoint::convertToSinglePrecisionFloat() {
    const int sign = 1;
    const int expo = 8;
    const int mantis = 23;
    float res = 0;

    string exponent = _binarySequence.substr(1, expo);
    BinaryToInteger bti(8, exponent);
    int exp = bti.convertToUnsignedInteger() - 127;
    res += pow(2, exp);

    string mantissa = _binarySequence.substr(9, mantis);
    for (int i = 0; i < mantis; i++) {
        if (mantissa[i] == '1') {
            res += pow(2, exp - 1 - i);
        }
    }

    if (_binarySequence[0] == '1') res = -res;

    return res;
}

```

Listing 15: 32 bit To Float

2.5.16 Chuyển đổi dãy 64 bit sang số thực

- **Mô tả thuật toán**

  Cách chuyển đổi hoàn toàn tương tự như chuyển đổi dãy 32 bit.

  Một số điểm khác:

  – Phần mũ thay vì biểu diễn 8 bit thì biểu diễn bằng 11 bit ở dạng quá 1023 do đó khi tính ta cần - 1023.

  – Phần định trị bao gồm 52 bit.

  Như đã đề cập chi tiết phía trên nên chúng tôi sẽ không đi sâu vào cách tính chi tiết thêm nữa.

- **Mã nguồn**

```
double BinaryToFloatingPoint::convertToDoublePrecision() {
    const int sign = 1;
    const int expo = 11;
    const int mantis = 52;
    double res = 0;

    string exponent = _binarySequence.substr(1, expo);
    BinaryToInteger bti(16, exponent);
    int exp = bti.convertToUnsignedInteger() - 1023;
    res += pow(2, exp);

    string mantissa = _binarySequence.substr(12, mantis);
    for (int i = 0; i < mantis; i++) {
        if (mantissa[i] == '1') {
```

```
                res += pow(2, exp - 1 - i);
            }
        }

        if (_binarySequence[0] == '1') res = -res;

        return res;
    }

```

Listing 16: 64 bit To Float

Tài liệu

[1] https://courses.ctda.hcmus.edu.vn/pluginfile.php/66829/mod_resource/content/2/HTMT.Ch02.DataRepresentation.pdf p.32.

[2] https://stackoverflow.com/questions/29647301/have-to-convert-integer-to-binary

[3] https://en.wikipedia.org/wiki/Two%27s_complement

[4] https://stackoverflow.com/questions/52718059/change-binary-to-decimal/52718226

[5] https://stackoverflow.com/questions/30051173/twos-complement-of-decimal-number

[6] https://courses.ctda.hcmus.edu.vn/pluginfile.php/66829/mod_resource/content/2/HTMT.Ch02.DataRepresentation.pdf p.37.