

ĐẠI HỌC QUỐC GIA TP HCM

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

---

# Báo cáo Homework 01

Đề tài: Data Representation

---

Môn học: Hệ thống máy tính

*Sinh viên thực hiện:*

Võ Hữu Tuấn (22127439)



# Mục lục

<b>A Write a program</b>	<b>2</b>
A.1 Source code . . . . .	2
A.2 Idea . . . . .	2
A.2.1 Convert a signed integer X (16-bit) to the binary bit pattern of X (2's complement form) . . . . .	2
A.2.2 Convert a single precision Y to the binary bit pattern of Y . . . . .	2
A.3 Capture the picture . . . . .	2
<b>B Complete these exercises</b>	<b>4</b>
B.1 . . . . .	4
B.2 . . . . .	4
B.3 . . . . .	5
B.4 . . . . .	5
<b>C TÀI LIỆU THAM KHẢO</b>	<b>8</b>

## A Write a program

### A.1 Source code

Source code ở đây

### A.2 Idea

#### A.2.1 Convert a signed integer X (16-bit) to the binary bit pattern of X (2's complement form)

- Tạo một biến x để nhận giá trị đầu vào và một biến bool (flag) cho biết x âm hay dương (Quy ước: bool == true là x âm).
- Tạo một mảng một chiều có 16 phần tử để lưu trữ dạng nhị phân của |x|.
- Sử dụng cách chia 2 lấy dư để tìm dạng nhị phân của |x|.
- Kiểm tra biến bool, nếu là false thì bỏ qua. Nếu là true, lật tất cả các bit trong mảng rồi +1 cho bit cuối bên phải.
- Cuối cùng, xuất ra dãy bit vừa chuyển đổi.

#### A.2.2 Convert a single precision Y to the binary bit pattern of Y

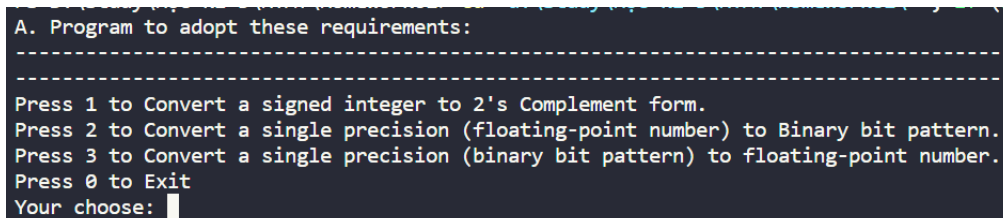
**Ý tưởng:** Sử dụng cấu trúc dữ liệu **bitset** có sẵn trong thư viện của c++ để thực hiện. Nhưng vì **bitset** chỉ nhận vào giá trị là số nguyên không dấu nên cần phải ép kiểu y từ **float** thành **unsigned int** để **bitset** có thể đọc được.

- Tạo một biến y (kiểu float) để nhận giá trị đầu vào.
- Tạo một **bitset** chứa 32 bit để lưu trữ dạng nhị phân của số y.
- Ép kiểu con trỏ trỏ đến vị trí của y từ kiểu con trỏ **float** sang con trỏ kiểu **unsigned int** bằng cách:

```
1 *reinterpret_cast<unsigned int*>(&y)
2
```

- dãy bit trong **bitset** chính là kết quả cần tìm.

### A.3 Capture the picture



```
A. Program to adopt these requirements:
-----
Press 1 to Convert a signed integer to 2's Complement form.
Press 2 to Convert a single precision (floating-point number) to Binary bit pattern.
Press 3 to Convert a single precision (binary bit pattern) to floating-point number.
Press 0 to Exit
Your choose: █
```

Hình 1: Menu hiển thị khi chạy chương trình

```
A. Program to adopt these requirements:
-----
Press 1 to Convert a signed integer to 2's Complement form.
Press 2 to Convert a single precision (floating-point number) to Binary bit pattern.
Press 3 to Convert a single precision (binary bit pattern) to floating-point number.
Press 0 to Exit
Your choose: 1
Input a signed integer X: 2004
The binary bit pattern of X(2s complement form): 0000011111010100
-----
```

Hình 2: Kết quả khi chuyển signed integer X sang binary bit pattern of X

```
-----
Press 1 to Convert a signed integer to 2's Complement form.
Press 2 to Convert a single precision (floating-point number) to Binary bit pattern.
Press 3 to Convert a single precision (binary bit pattern) to floating-point number.
Press 0 to Exit
Your choose: 2
Input a single precision Y (floating-point number): 23.022004
Binary bit pattern of Y: 01000001101110000010110100010000
-----
```

Hình 3: Kết quả khi chuyển single precision Y sang binary bit pattern of Y

```
A. Program to adopt these requirements:
-----
Press 1 to Convert a signed integer to 2's Complement form.
Press 2 to Convert a single precision (floating-point number) to Binary bit pattern.
Press 3 to Convert a single precision (binary bit pattern) to floating-point number.
Press 0 to Exit
Your choose: 1
Input a signed integer X: 2004
The binary bit pattern of X(2s complement form): 0000011111010100
-----
Press 1 to Convert a signed integer to 2's Complement form.
Press 2 to Convert a single precision (floating-point number) to Binary bit pattern.
Press 3 to Convert a single precision (binary bit pattern) to floating-point number.
Press 0 to Exit
Your choose: 2
Input a single precision Y (floating-point number): 23.022004
Binary bit pattern of Y: 01000001101110000010110100010000
-----
Press 1 to Convert a signed integer to 2's Complement form.
Press 2 to Convert a single precision (floating-point number) to Binary bit pattern.
Press 3 to Convert a single precision (binary bit pattern) to floating-point number.
Press 0 to Exit
Your choose: █
```

Hình 4: Sau khi chuyển đổi xong có thể tiếp tục chuyển đổi tiếp ngay

## B Complete these exercises

### B.1

**Phân tích:**

1. char chiếm 1 byte.
2. struct minipoint uint8\_t x; uint8\_t y; uint8\_t z; mỗi uint8\_t chiếm 1 byte mà struct có 3 biến kiểu này nên chiếm 3 byte.
3. int chiếm 4 byte.
4. unsigned short[1]: unsigned short chiếm 2 byte mà có 1 phần tử nên tổng chiếm là 2 byte.
5. char\*\* là con trỏ kiểu char nên sẽ tùy vào hệ thống mà chiếm 4 hoặc 8 byte.
6. double[0] vì không có phần tử nên chiếm 0 byte.

Từ trên ta có:  $6 < 1 < 4 < 2 < 3 \leq 5$ .

### B.2

**Ý tưởng:** Phép  $\wedge$  hay XOR là phép biến đổi trả về 0 khi 2 bit giống nhau và trả về 1 khi 2 bit khác nhau. Do đó, để dễ dàng ta cần chuyển các số thập lục phân  $\rightarrow$  thập phân  $\rightarrow$  nhị phân để giải dễ hơn.

Theo đề bài ta có:

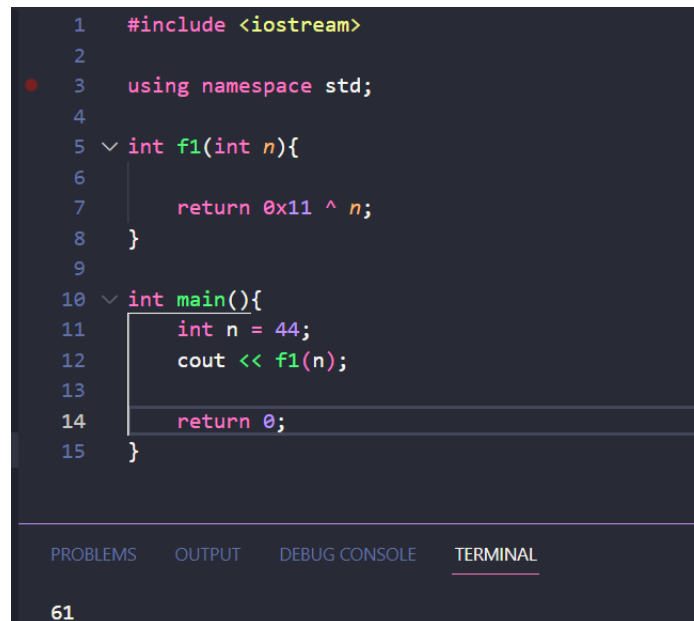
$$0x11_{16} \wedge n = 0x3d_{16}$$

$$\Leftrightarrow 17_{10} \wedge n = 61_{10} \text{ (đổi về kiểu thập phân)}$$

$$\Leftrightarrow 00010001_2 \wedge n = 00111101_2 \text{ (đổi về kiểu nhị phân)}$$

$$\Rightarrow n = 00101100_2 = 44_{10} = 0x2c_{16}$$

**Kiểm tra:** Chạy thử hàm `f1(int n)` với  $n = 44$ :



```

1  #include <iostream>
2
3  using namespace std;
4
5  int f1(int n){
6
7      return 0x11 ^ n;
8  }
9
10 int main(){
11     int n = 44;
12     cout << f1(n);
13
14     return 0;
15 }

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

61

Hình 5: Kết quả khi chạy với  $n = 44$

**Kết luận:**  $n = 00101100_2 = 44_{10} = 0x2c_{16}$ .

### B.3

```
1      struct s {  
2          char f1[7];  
3          char *f2;  
4          short f3;  
5          int f4;  
6      };
```

Dựa vào bảng chuyển dữ liệu của ENIAC cho sẵn ta dễ dàng biết được:

- char f1[7] có 7 phần tử kiểu char nên  $\text{sizeof}(f1[7]) = 7$ .
- char \*f2 là con trỏ nên  $\text{sizeof}(f2) = 16$ .
- short f3 là phần tử kiểu short nên  $\text{sizeof}(f3) = 4$ .
- int f4 là phần tử kiểu int nên  $\text{sizeof}(f4) = 8$ .

Tổng kích thước các phần tử  $= 7 + 16 + 4 + 8 = 35$ .

Và theo đề bài, ta chọn căn chỉnh kích thước của struct theo bội số của 16 (theo kích thước của char\*). Lúc này kích thước của struct s sẽ được làm tròn lên 48 (48 là bội của 16).

Hay  $\text{sizeof}(\text{struct s}) = 7 + 9 \text{ (byte trống)} + 16 + 4 + 8 + 4 \text{ (byte trống)} = 48$ .

### B.4

**Ý tưởng:** Đầu tiên sẽ chuyển số từ hệ thập lục phân sang hệ nhị phân rồi sử dụng chuẩn 32-bit IEEE 754 để đưa về dạng thập phân như đề bài yêu cầu.

\*Hệ thập lục phân  $\rightarrow$  nhị phân: Cứ 1 ký tự trong hệ 16 sẽ tương ứng với 4 bit nhị phân nên ta có bảng chuyển đổi sau:

binary	hexa
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Bảng 1: Bảng chuyển đổi giữa hệ nhị phân và hệ thập lục phân

**a. 0x4000 0000**

Dạng nhị phân: 0 10000000 000000000000000000000000

Signed: 0 => số dương

Exponent:  $10000000_2 = 128_{10}$

=>  $E = 128 - 127 = 1$ .

Significand: 000000000000000000000000

=> Fixed-point:  $1.00 * 2^1 = 10.00_2$

=> Dạng thập phân =  $2^1 = 2$

**b. 0x3d80 0000**

Dạng nhị phân: 0 01111011 000000000000000000000000

Signed: 0 => số dương

Exponent:  $01111011_2 = 123_{10}$

=>  $E = 123 - 127 = -4$

Significand: 000000000000000000000000

=> Fixed-point number:  $1.000 * 2^{-4} = 0.0001_2$

=> Dạng thập phân:  $2^{-4} = 0.0625$

**c. 0xc259 48b4** Dạng nhị phân: 1 10000100 10110010100100010110100

Signed: 1 => số âm

Exponent:  $10000100_2 = 132_{10}$

=>  $E = 132 - 127 = 5$

Significand: 10110010100100010110100

=> Fixed-point number:  $-1.10110010100100010110100 * 2^5 = -110110.010100100010110100_2$

=> Dạng thập phân:  $-(2^1 + 2^2 + 2^5 + 2^6 + 2^{-2} + 2^{-4} + 2^{-7} + 2^{-11} + 2^{-13} + 2^{-14} + 2^{-16}) = -54.3209991455$

\*Có thể dùng tính năng "Convert a single precision (binary bit pattern) to floating-point number" trong [source code đính kèm ở mục A.1](#) để kiểm tra kết quả chuyển đổi theo chuẩn 32-bit IEEE 754 của các câu trên.

```
-----
Press 1 to Convert a signed integer to 2's Complement form.
Press 2 to Convert a single precision (floating-point number) to Binary bit pattern.
Press 3 to Convert a single precision (binary bit pattern) to floating-point number.
Press 0 to Exit
Your choose: 3
Input a single precision Y (binary bit pattern): 01000000000000000000000000000000
Signed: 0
Exponent: 10000000
Significand: 000000000000000000000000
Fixed point number: 10.0000000000000000000000
Decimal value: 2.000000
```

Hình 6: Kết quả khi chạy bằng tính năng trên với giá trị ở câu B.4a



## C TÀI LIỆU THAM KHẢO

Chapter1 - Introduction Data Representation, COMPUTER SYSTEM, FITUS.

[Tìm hiểu về reinterpret\\_cast](#)

[Tìm hiểu về Structure Alignment trong lập trình c/c++](#)