BÁO CÁO BÀI TẬP

**Môn học: Lập trình hệ thống**

**Buổi báo cáo: Lab 1**

**Tên chủ đề:** Data Lab: Tính toán Bits

*GVHD: Đỗ Thị Hương Lan*

*Ngày thực hiện: 29/09/2025*

**THÔNG TIN CHUNG:**

*(Liệt kê tất cả các thành viên trong nhóm)*

Lớp: NT209.Q12.ANTT

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Họ và tên** | **MSSV** | **Email** |
| 1 | Trần Ngọc Danh | 22520200 |  |
| 2 | Đoàn Minh Khang | 22520609 |  |
| 3 | Phạm Huỳnh Tấn Khang | 22520624 |  |

1. **ĐÁNH GIÁ KHÁC:**

|  |  |
| --- | --- |
| **Nội dung** | **Kết quả** |
| Tổng thời gian thực hiện bài thực hành trung bình |  |
| Link Video thực hiện  *(nếu có)* |  |
| Ý kiến *(nếu có)*  + Khó khăn  + Đề xuất … |  |
| Điểm tự đánh giá |  |

**Phần bên dưới của báo cáo này là báo cáo chi tiết của nhóm thực hiện.**

BÁO CÁO CHI TIẾT

1. **Tính toán bit đơn giản**
   1. Hàm bitOr(x, y)

Hàm sử dụng định luật De Morgan để tính toán phép OR chỉ với toán tử ~ và &

Định lý De Morgan được phát biểu như sau:



Nên ta suy ra hàm:

*int bitOr(int x, int y)*

*{*

*return ~(~x & ~y);*

*}*

* 1. Hàm negative(x)

Hàm dựa trên biểu diễn **bù 2** trong hệ nhị phân để tính số đối. Cụ thể, ~x + 1 cho ta giá trị nghịch đảo của x dẫn đến cho ta kết quả là số đối của x.

Lưu số âm bằng bù 2. Lấy tất cả bit đảo lại rồi cộng thêm 1 sẽ cho ra giá trị âm tương ứng

*int negative(int x)*

*{*

*return ~x + 1; // dùng bù 2*

*}*

* 1. Hàm getHexcha(x, n)

Mỗi chữ số trong hệ 16 chiếm đúng 4 bit. Khi ta muốn lấy chữ số thứ n, chỉ cần dịch sang phải để nó nằm ở cuối, rồi dùng mặt nạ để loại bỏ các bit thừa

Phương pháp: Dịch số x sang phải n\*4 bit để đưa nhóm 4 bit cần lấy về cuối, rồi AND với 0xF để giữ lại đúng 4 bit đó

*int getHexcha(int x, int n)*

*{*

*int shiftAmount = n << 2; // Tính số bit cần dịch: n \* 4, tương đương n << 2*

*// Dịch phải x để đưa byte thứ n về vị trí thấp nhất*

*int result = x >> shiftAmount;*

*// Dùng mask 0xF để xóa tất cả các bit trừ 8 bit cuối cùng*

*return result & 0xF;*

*}*

* 1. Hàm flipByte(x, n)
  2. Hàm divpw2(x, n)

1. **Các phép kiểm tra dựa trên tính toán bit**
   1. Hàm isEqual(x, y)

Dùng XOR để kiểm tra: nếu x ^ y = 0 thì hai số bằng nhau. Sau đó phủ định kết quả bằng !

XOR của hai số bằng 0 nghĩa là từng bit của chúng giống nhau. Nhờ vậy, ta xác định được hai số có bằng nhau mà không cần so sánh trực tiếp

*int isEqual(int x, int y)*

*{*

*return !(x ^ y);*

*}*

* 1. Hàm is16x(x)
  2. Hàm isPositive(x)

Bit dấu trong bù 2 cho biết số âm hay dương, còn (x | -x) khai thác tính chất bù 2 để phân biệt 0 với số khác 0. Nhờ vậy hàm trả về chính xác 1 khi x dương và 0 trong các trường hợp còn lại

Dịch phải 31 bit để lấy bit dấu. Nếu kết quả = 0 thì số không âm, đảo lại bằng ! để được NotNegative

Dùng (x | -x) >> 31 & 1 để kiểm tra khác 0: kết quả = 1 nếu x ≠ 0, ngược lại = 0 ( vì bù 2 của 0 vẫn bằng 0 )

Kết hợp hai điều kiện bằng & để ta có kết quả bằng 1 khi x > 0

*int isPositive(int x)*

*{*

*// Điều kiện 1: x không phải là số âm (tức là x >= 0).*

*int NotNegative = !(x >> 31);*

*// Điều kiện 2: x không bằng 0.*

*int NotZero = ((x | -x) >> 31) & 1;*

*return NotNegative & NotZero;*

*}*

* 1. isGE2n(x, n)