

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Khoa Công Nghệ Thông Tin

-----000-----



BÁO CÁO ĐỒ ÁN

Môn: Kiến trúc máy tính và hợp ngữ

Đề Tài: Biểu diễn và tính toán số nguyên lớn

Giáo viên hướng dẫn: Nguyễn Thanh Quân

Sinh viên thực hiện: Nguyễn Hữu Tú - 1612766

Nguyễn Anh Tuấn - 1612778

TPHCM, ngày 19 tháng 4 năm 2018

MỤC LỤC

1. Phân công công việc
2. Môi trường lập trình
3. Ý tưởng và phạm vi biểu diễn
4. Chạy kiểm tra
5. Tỷ lệ hoàn thành
6. Các nguồn tài liệu tham khảo

1. Phân công công việc:

- Nguyễn Hữu Tú
 - Tìm hiểu đề tài.
 - Thiết kế cấu trúc dữ liệu.
 - Chuyển đổi số QInt từ hệ thập phân sang hệ nhị phân và ngược lại
 - Các operator=, operator+, operator-, operator*, operator/
 - Các toán tử AND “&”, OR “|”, XOR “^”, NOT “~”
 - Các phép xoay trái “rol”, xoay phải “ror” mỗi lần xoay chỉ xử lý cho đúng 1 bit, không xử lý cho trường hợp tổng quát xoay k bit.
- Nguyễn Anh Tuấn
 - Tìm hiểu đề tài.
 - Thiết kế cấu trúc dữ liệu.
 - Chuyển đổi số QInt từ hệ nhị phân sang hệ thập lục phân và ngược lại.
 - Các toán tử: dịch trái “<<”, dịch phải “>>”
 - Viết hàm main
 - Viết báo cáo

2. Môi trường lập trình:

Microsoft Visual Studio 2015

3. Ý tưởng và phạm vi biểu diễn:

a. Ý tưởng:

- Thiết kế cấu trúc dữ liệu: Yêu cầu của đề bài là phải biểu diễn số nguyên lớn có độ lớn 16 byte, trong khi đó ngôn ngữ lập trình chỉ hỗ trợ lưu trữ tối đa 8 byte. Do đó ta suy nghĩ đến việc dùng 1 mảng gồm có 2 phần tử, mỗi phần tử 8 byte để lưu được hết số bit dữ liệu của số Nguyên lớn 16 byte.
- Hàm chuyển chuỗi nhị phân sang kiểu QInt: Đầu tiên ta thêm bit 0 ở đầu sao cho chuỗi nhị phân có đủ 128 bit. Sau đó dùng các phép dịch bit như trong hàm setBit để tính ra 1

số QInt(nếu số cần chuyển là dương), kiểm tra xem nó là số âm dương để chuyển về số QInt đúng.

- Hàm chuyển QInt sang chuỗi nhị phân: dùng kiểu dữ liệu `bitset<64>` để chuyển mỗi phần tử long long thành string nhị phân, từ đó suy ra chuỗi nhị phân.
- Hàm chuyển chuỗi thập phân sang chuỗi nhị phân: Chuyển từ chuỗi thập phân sang QInt rồi từ QInt đến chuỗi nhị phân: dùng thuật toán Double-Dabble
- Hàm chuyển chuỗi nhị phân sang chuỗi thập phân:
- Hàm chuyển chuỗi thập lục phân sang chuỗi nhị phân.
- Hàm chuyển chuỗi nhị phân sang thập lục phân: thêm vài bit 0 ở đầu sao cho $\text{length} \% 4 = 0$
- Các hàm cộng trừ 2 số QInt: Dùng phép cộng trừ thông thường trên 2 phần tử của mảng long long.
- Hàm nhân 2 số QInt: Chia số nguyên 128 bit thành 4 phần, mỗi phần chứa 32 bit. Số đầu tiên chia thành 4 phần, ký hiệu abcd. Số thứ hai gồm bốn phần xyzt.
- Hàm chia 2 số QInt: ta thực hiện như phép chia thủ công với số thập phân bằng tay.
- Các hàm and, or, xor, not thì ta lần lượt thực hiện các thao tác đó trên 2 phần tử của mảng long long.
- Các thao tác dịch trái và dịch phải: xét xem thao tác dịch bit sẽ dịch bao nhiêu bit để xem độ ảnh hưởng của nó với các phần tử của mảng long long.
- Các thao tác xoay bit: nếu rol thì xét bit đầu là bit gì, còn ror thì xét bit cuối là bit gì để kết hợp với phép dịch rồi or với 1 để ra kết quả.

b. Phạm vi biểu diễn:

Ta biểu diễn được số nguyên từ -2^{127} đến $2^{127}-1$.

4. Chạy kiểm tra:

```

1 #include "QInt.h"
2
3 int main(int argc, char** argv)
4 {
5     fstream fpInput, fpOutput;
6     string str;// Chuỗi lưu từng dòng của input

```

input - Notepad

```

2 1111100011101010111 + 01101110110111
2 110110111 * 0101011111
2 10 011010101111011111
10 2 8793278316383117319
16 85AF + 90BC
10 5678 >> 2
2 ror 0111000110101110
10 ~ -2000

```

output - Notepad

```

1111110001100001110
1001010110110101
438207
111101000001000000000000000101000111011001001000000000111
11668
1419
11100011010111
1999

```

Đây là kết quả của chương trình khi chạy ví dụ mẫu của thầy.

```

1 #include "QInt.h"
2
3 int main(int argc, char** argv)
4 {
5     fstream fpInput, fpOutput;
6     string str;// Chuỗi lưu từng dòng của input

```

input_test - Notepad

```

2 10 1110111011010110101
10 2 7383430404029402320
10 16 233988923933324442
16 10 76CA
2 16 010101110101010001
16 2 47AE
10 281293633310313 + 2320420130093293
16 48DC - 32AC
2 011101010111010101 * 01000111110010
10 21328329383342424493 / 222392932822323
2 101010100101111 & 010101001010101
16 57CB | 24CF
2 101010101010101 ^ 101010000101011
16 ~ 34FC
10 383332229238432 << 2
2 101010010100101001 >> 4
16 ror 46FA
10 rol 29828392932832021

```

output - Notepad

```

489141
1100110011101110011010101010101111111011100110011010000
33F4BAFC46BF49A
30410
57551
0100011110101110
2601713763403606
1630
100000111111000010000100101010
12957
101
77CF
101111110
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFCB03
1533328916953728
10101001010010
237D
59656785865664042

```

Đây là kết quả của chương trình khi chạy 1 số test case để thực hiện các chức năng như đề bài yêu cầu.

5. Tỷ lệ hoàn thành:

Đã làm đầy đủ tất cả các chức năng, kiểm thử chạy tốt.

Tỷ lệ hoàn thành là 100%.

6. Các nguồn tài liệu tham khảo:

- Các thao tác trên bit:
https://vi.wikipedia.org/wiki/Ph%C3%A9p_to%C3%A1n_t%C3%A1c_tr%C3%AAn_bit
- Nhiều nguồn tài liệu khác trên stackoverflow và github
<https://stackoverflow.com>
<https://github.com>
- File hướng dẫn đồ án.