BÁO CÁO ĐỒ ÁN 1
KIẾN TRÚC MÁY TÍNH VÀ HỢP NGỮ
BIỂU DIỄN VÀ TÍNH TOÁN SỐ NGUYÊN LỚN

1. Phân công công việc

Họ tên	MSSV	Đảm nhận
Nguyễn Minh Đức	1712358	layout chương trình chính; các phép toán trên bit (&, |, ^, ~, << , >>, rol, ror)
Phạm Quốc Dũng	1712369	các thuật toán chính; các phép toán +, -, *, / trên số có dấu;
Đặng Thành Duy	1712379	các phép chuyển đổi cơ số qua lại: 10 → 2, 2 → 10, 16 → 2, 2 → 16 trên số có dấu;

2. Môi trường lập trình: Visual studio 2017

3. Ý tưởng thiết kế và thực hiện đồ án

Ý tưởng:

⁂ Dùng kiểu bitset để lưu trữ số nguyên lớn theo dạng nhị phân

⁂ Các phép toán được thực hiện trực tiếp trên hệ cơ số 2, các phép toán trên hệ cơ số 10 và 16 được thực hiện gián tiếp qua việc chuyển đổi sang cơ số 2 → tính toán → chuyển đáp số về cơ số ban đầu.

Thực hiện:

⁂ Dựng class Qint đại diện cho kiểu số nguyên lớn với trường dữ liệu là bitset<128> và trường method là các phép chuyển đổi cơ số, các operator phép toán. Việc chuyển các hệ cơ số 2, 10, 16 → 2 được thực hiện ngay tại constructor của class, constructor này nhận vào string số đầu vào và cơ số base của số đầu vào. Qint quá tải tất cả các toán tử + - * / & | ^ ~ rol ror,... rồi trả về 1 biến cùng kiểu Qint. Class Qint

cung cấp các hàm chuyển đổi cơ số 2 → 2, 10, 16 trả về 1 string cho số sau khi chuyển đổi.

⁂ Dòng dữ liệu đầu vào lấy từ file argv[1] được xử lý sao cho chương trình đưa được dữ liệu cho class Qint để xử lý thành string kết quả rồi in ra file argv[2]. Như vậy từ 1 dòng dữ liệu tụi em phải phân tách thành:

* Hệ cơ số đang dùng.
* Thao tác cần xử lý (là chuyển đổi cơ số hay thực hiện phép tính + -,..)
* string chuỗi số cần xử lý (1 string với phép toán 1 biến như phép lấy not, 2 tring với phép toán 2 biến như phép + -...)

→ Từ ý tưởng trên tụi em thấy việc đầu tiên là phải phân tách dòng dữ liệu đầu vào thành các substring qua các dấu cách rồi lưu vào vector tên data. Ví dụ dòng "10 1 + 2" được phân tách thành 4 substring "10", "1", "+", "2". Các substring được lưu trong 1 vector rồi truyền vào cho hàm execute xử lý. Từ form của dữ liệu thầy cung cấp, tụi em chia dữ liệu thành 2 dạng: dạng 4 substring và dạng 3 substring. Dạng 4 substring là dạng phép toán 2 biến: data[0] là cơ số ban đầu, operator nằm ở data[2], còn 2 biến số nằm ở data[1] và data[3]. Dạng 3 substring là dạng phép toán 1 biến hoặc phép chuyển đổi cơ số: data[0] là cơ số ban đầu, data[1] chứa operator hoặc chỉ thị p2 (cơ số sau), biến số nằm ở data[2]. Từ việc nhận biết các substring này, chương trình sẽ thực hiện yêu cầu của dòng dữ liệu ban đầu.

⁂ Các phép chuyển đổi:

* 2 → 2 (string → bitset): được thực hiện dễ dàng nhờ kiểu bitset có sẵn của c++
* 10 → 2 (string → bitset): được thực hiện bằng quy tắc chia 2
* 16 → 2 (string → bitset): được thực hiện bằng cách đổi từng đơn vị thành bin 4 bit rồi gom lại
* 2 → 2 (bitset → string): được thực hiện dễ dàng nhờ kiểu bitset có sẵn của c++
* 2 → 10 (bitset → string): được thực hiện bằng thuật toán double dabble để đổi sang BCD rồi đổi từng 4 bit sang hệ 10 rồi gom lại
* 2 → 16 (bitset → string): được thực hiện bằng cách đổi từng 4 bit sang hex rồi gom lại

⁂ Các phép toán + - * /:

* Phép cộng:

1. Đặt một biến kiểu bool tên là **temp** để nhớ vào cột sau (được gán sẵn 0)
2. Tạo đối tượng **C** kiểu Qint để trả về kết quả cộng.
3. Chạy vòng lặp biến **i** từ 0 đến 127, i cộng 1 đơn vị mỗi lần lặp.
    - Nếu 2 bit ở vị trí i của 2 số hạng giống nhau (hay phép XOR giữa 2 bit đó có kết quả bằng 0) :
        - Gán bit ở vị trí i của C bằng temp.
        - Gán temp bằng bit ở vị trí i của 1 trong 2 số hạng.
    - Nếu 2 bit ở vị trí i của 2 số hạng khác nhau:
        - Gán bit ở vị trí i của C bằng NOT của temp.
4. Trả về C.

* Phép trừ:
    1. Đổi dấu số trừ (bằng cách sử dụng phép NOT cho số trừ, sau đó cộng thêm 1)
    2. Thực hiện phép cộng giữa số bị trừ và số trừ vừa được đổi dấu.
* Phép nhân:
    1. Để dễ hình dung, ta sẽ thực hiện phép nhân theo kiểu đặt tính rồi tính, một thừa số gọi là **P** nằm trên, một thừa số gọi là **Q** nằm dưới.
    2. Khởi tạo đối tượng **C** kiểu Qint. Đây sẽ là đối tượng trả về kết quả phép nhân.
    3. Khởi tạo mảng **temp** kiểu bool lưu giá trị tính toán tạm thời ở từng cột.
    4. Khởi tạo mảng **memory** kiểu int lưu số lần nhớ ở từng cột
    5. Cho chạy vòng lặp biến **i** từ 0 đến 127, i tăng 1 đơn vị (chạy từ cuối bitset lên đầu bitset vì bitset lưu giá trị ngược với giá trị hiển thị)
        - Nếu phần từ Q[i] = 0 thì bỏ qua vòng lặp này.
        - Khởi tạo một vòng lặp khác trong vòng lặp cũ với biến **j** chạy từ 0 cho đến khi điều kiện i + j < 128 không còn đúng.

- Tạo 1 biến kiểu bool tên là **x**, gán giá trị là phép AND giữa P[j] và Q[i].
- Nếu temp[i + j] = 1 và x cũng = 1 thì gán temp[i + j] = 0 và tăng giá trị của số lần nhớ cột kế bên (tức là memory[i + j + 1]) lên 1 đơn vị
- Nếu temp[i + j] = 0 thì gán temp[i + j] = x.

6. Chạy vòng lặp với biến **i** từ 0 đến 127.
7. Nếu temp[i] = 0:
   - Nếu memory[i] là số chẵn thì gán C.core[i] = 0 và cộng memory[i + 1] thêm 1 giá trị bằng memory[i] / 2.
   - Nếu memory[i] là số lẻ thì gán C.core[i] = 1 và cộng memory[i + 1] thêm 1 giá trị bằng (memory[i] – 1) / 2.
8. Nếu temp[i] = 1:
   - Nếu memory[i] là số chẵn thì gán C.core[i] = 1 và cộng memory[i + 1] thêm 1 giá trị bằng memory[i] / 2.
   - Nếu memory[i] là số lẻ thì gán C.core[i] = 0 và cộng memory[i + 1] thêm 1 giá trị bằng (memory[i] + 1) / 2.
9. Trả về C.

❋ Phép chia:

1. Khởi tạo đối tượng **C** kiểu Qint. Đây sẽ là đối tượng trả về thương của phép chia.
2. Để dễ hình dung, gọi **P** là số bị chia và **Q** là số chia.
3. Tạo 1 biến kiểu bool tên là **sign**. Đây là biến để lưu dấu của thương. Gán sign = phép XOR giữa 2 bit đầu tiên của P và Q.
4. Nếu dấu của P âm thì tìm gán P là số đối của chính nó bằng cách sử dụng phép bù 2, tương tự với Q.
5. Nếu P < Q, trả về C ngay lập tức.
6. Khai báo đối tượng **temp** kiểu Qint dùng để lưu số dư.
7. Tạo 2 biến kiểu int tên là **index_sobichia** và **index_sochia** dùng để lưu vị trí mà bit 1 xuất hiện đầu tiên trong P và Q.
8. Chạy 2 vòng lặp liên tiếp để tìm vị trí bit 1 xuất hiện đầu tiên trong P và Q.

9. Dịch trái Q đi (index_sobichia – index_sochia) vị trí (mục đích để bit 1 đầu tiên của P và Q nằm ở cùng 1 vị trí)
10. Tạo biến **i** kiểu int và gán bằng index_sobichia.
11. Chạy vòng lặp với biến i có sẵn với điều kiện

i > index_sobichia – index_sochia, i giảm đi 1 đơn vị.

- Gán từng bit của temp ở vị trí i bằng P ở vị trí i.

12. Chạy vòng lặp với biến i có sẵn, điều kiện là i >= 0, i giảm đi 1 đơn vị.
    - Gán bit ở vị trí i của temp = bit ở vị trí i của P.
    - Nếu temp lớn hơn hay bằng Q:
        - Gán bit ở vị trí i của C bằng 1.
        - Gán temp bằng kết quả phép trừ giữa chính nó và Q.
    - Dịch Q sang phải 1 vị trí.
13. Nếu sign = 1, tức là dấu của C là âm thì lấy C bằng số đối của nó (dùng phép lấy số bù 2)
14. Trả về C.

⁂ Các phép toán trên dãy bit:

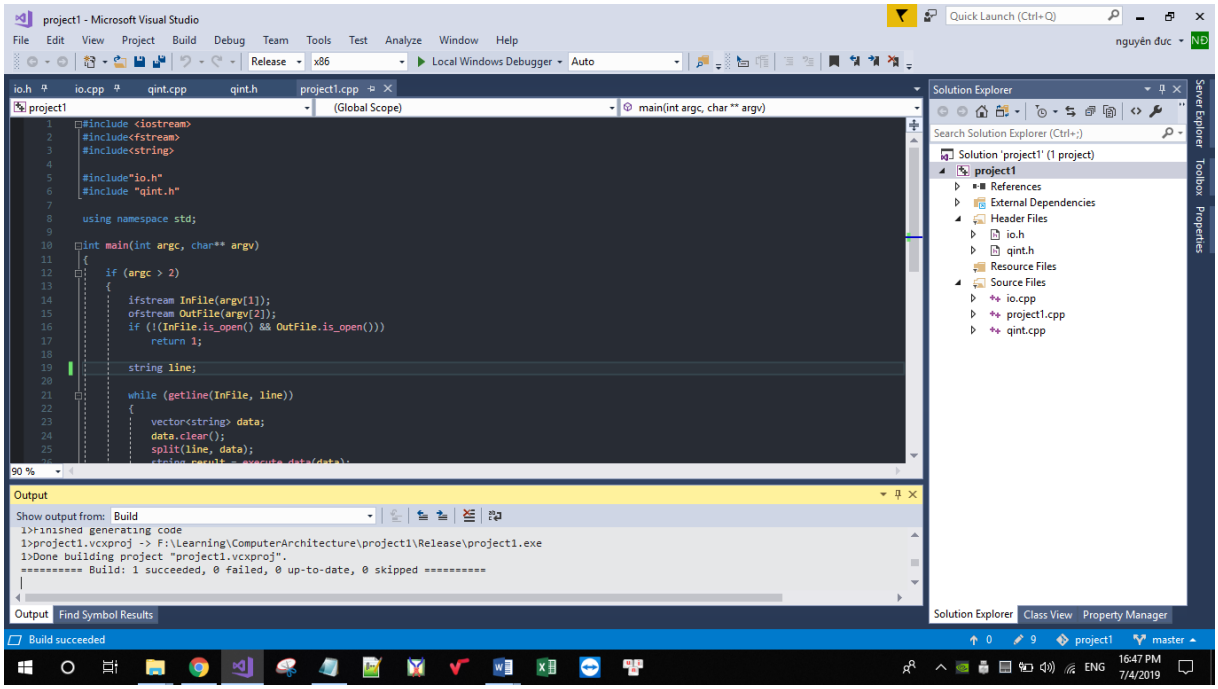
⁕ Các phép & | ^ ~ << được thực hiện dễ dàng nhờ kiểu bitset có sẵn của c++

⁕ Phép rol (xoay trái) được thực hiện theo 3 bước: lưu bit đầu tiên → dịch trái 1 bit → set bit cuối cùng bằng bit đầu tiên đã lưu. Tương tự cho phép ror.

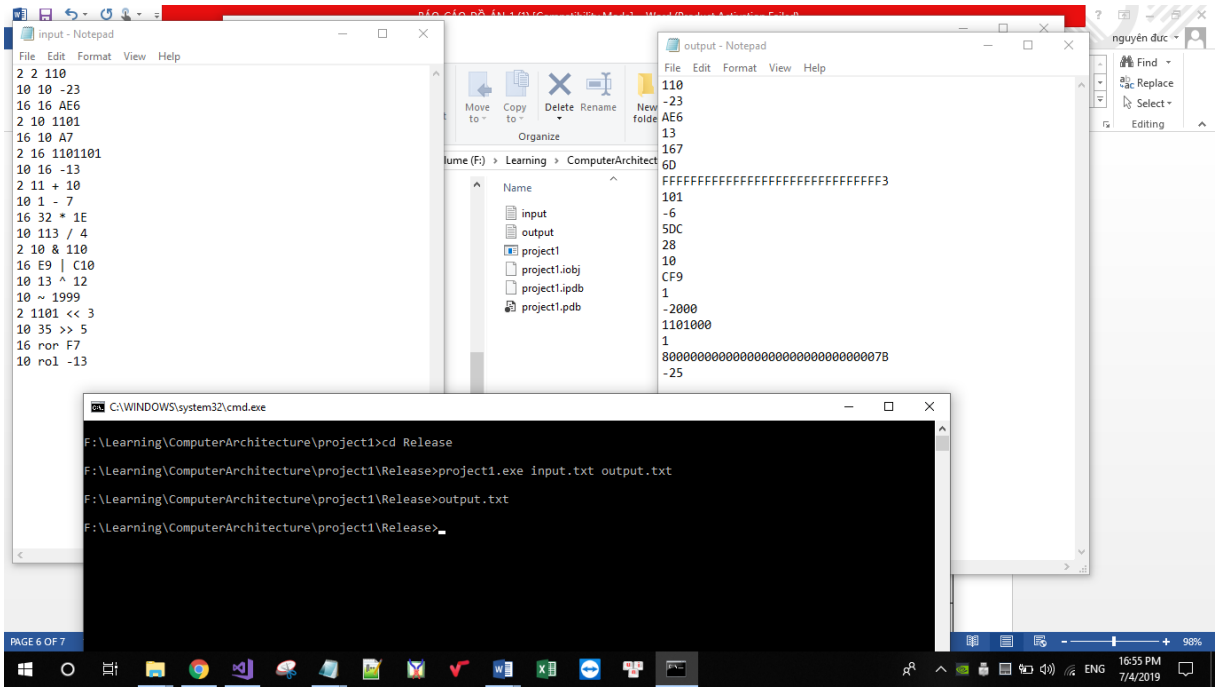
⁕ Phép >> k (dịch phải k bit) được thực hiện đơn giản nhờ kiểu bitset nhưng với số có dấu thì ta phải thêm 1 bước. Nếu số thực hiện là số âm (bit đầu bằng 1) thì cứ mỗi lần dịch phải 1 bit ta phải gán bit đầu bằng 1.

4. Chạy kiểm tra

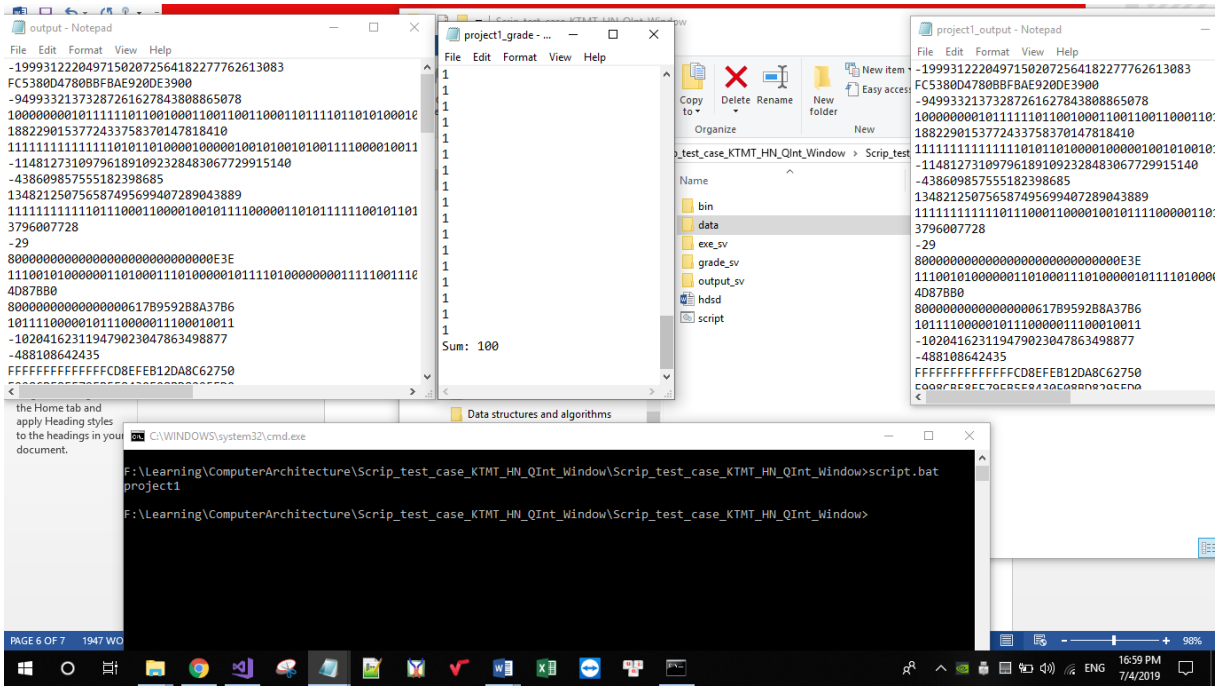
Build thành công ở chế độ release



Chạy thử thành công trên command line của windows các phép chuyển đổi cơ bản cùng các phép tính cơ bản



Chạy thử thành công đúng 100 test case trong bộ test thầy đưa



5. Chức năng

Chức năng	Đã làm được (✓)
Chuyển đổi cơ số 10 → 2	✓
Chuyển đổi cơ số 16 → 2	✓
Chuyển đổi cơ số 2 → 10	✓
Chuyển đổi cơ số 16 → 10	✓
Chuyển đổi cơ số 10 → 16	✓
Chuyển đổi cơ số 2 → 16	✓
Phép cộng	✓
Phép trừ	✓
Phép nhân	✓
Phép chia	✓
Các phép toán trên bit & | ^ ~ << >> ror rol	✓

6. Mức độ hoàn thành

Dựa trên bảng chức năng ở trên thì nhóm em đã làm hết các chức năng yêu cầu của đồ án. Kết quả đầu ra được xóa hết các bit 0 ở đầu (theo yêu cầu), nếu kết quả bằng 0 thì in ra 0. Nhóm em cũng lập trình các chức năng theo form số có dấu dạng bù 2 nên hoàn toàn đáp ứng được các phép toán số có dấu dạng bù 2. Nhóm em đã chạy thử 100 test case thầy post và đúng hết 100 test. Nhưng không ngoại trừ trường hợp chương trình chạy sai nên nhóm em ước lượng mức độ hoàn thành đồ án là 95%.

7. Tài liệu tham khảo

- Giáo trình kiến trúc máy tính – Nguyễn Minh Tuấn
- Slide bài giảng kiến trúc máy tính và hợp ngữ - Chương 2: Biểu diễn số nguyên.
- https://www.geeksforgeeks.org/c-bitset-and-its-application/
- http://www.cplusplus.com/reference/bitset/bitset/