

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

Đồ án 1
Kiến trúc máy tính và hợp ngữ

BIỂU DIỄN VÀ TÍNH TOÁN SỐ HỌC TRÊN MÁY TÍNH

Học kỳ II
Năm học: 2017- 2018

LỜI CẢM ƠN

Trong quá trình thực hiện đồ án đề tài, nhóm em đã nhận được rất nhiều sự giúp đỡ, hỗ trợ từ các thầy cô Trường Đại học Khoa học Tự nhiên – ĐHQG TP.HCM và các bạn bè trong trường. Nhóm em xin bày tỏ lòng cảm ơn chân thành về sự hướng dẫn, chỉ bảo của mọi người.

Đặc biệt, nhóm em xin bày tỏ lòng biết ơn sâu sắc đến các thầy cô khoa Công nghệ thông tin, cụ thể hơn chúng em xin cảm ơn thầy Phạm Tuấn Sơn đã hỗ trợ, giảng dạy rất kỹ lưỡng từng phần nhỏ để chúng em có một đồ án thật hoàn chỉnh và phù hợp nhất. Chính những thứ tưởng chừng như nhỏ nhặt này đã góp phần to lớn giúp chúng em hoàn thành đồ án và bảng báo cáo này thật hoàn chỉnh nhất. Một lần nữa, chúng em xin bày tỏ lòng biết ơn sâu sắc đến với các thầy cô và bạn bè.

Ngoài ra, nhóm chúng em còn nhận được rất nhiều sự khích lệ tinh thần, động viên cổ vũ không chỉ từ các bạn đồng trang lứa mà còn đến từ các anh chị khoa Công nghệ thông tin khóa trước và các thầy cô.

Chúng em xin chân thành cảm ơn!

Mục lục

Mục lục.....	2
1 CHƯƠNG I: MỞ ĐẦU	4
1.1 Giới thiệu nhóm và phân công công việc.....	4
1.2 Mô tả đồ án.....	4
2 CHƯƠNG II: GIỚI THIỆU ĐỒ ÁN	4
2.1 Thời gian thực hiện	4
2.2 Các bước thực hiện đồ án	4
3 CHƯƠNG III: NỘI DUNG ĐỒ ÁN.....	5
3.1 Sơ đồ UML	5
3.2 Lớp BigInt	7
3.2.1 Get_bit(int index, bool value)	7
3.2.2 Get_bit(int).....	7
3.3 Lớp BigInt	7
3.3.1 Chuyển giữa hệ thập phân và nhị phân	7
3.3.2 Chuyển giữa hệ nhị phân và hệ thập lục phân	8
3.3.3 Toán tử cộng	8
3.3.4 Toán tử trừ.....	8
3.3.5 Toán tử nhân	8
3.3.6 Toán tử chia.....	9
3.3.7 Phép xử lý bit	9
3.3.8 Phép so sánh.....	9
3.4 Lớp BigFloat	9
3.4.1 Chuyển giữa hệ thập phân và nhị phân	10
3.4.2 Chuyển giữa hệ nhị phân và hệ thập lục phân	10
3.4.3 Toán tử cộng	10
3.4.4 Toán tử trừ.....	11
3.4.5 Toán tử nhân	11
3.4.6 Toán tử chia.....	11
3.4.7 Phép so sánh.....	11
4 CHƯƠNG V: ĐÁNH GIÁ VÀ TỔNG KẾT QUÁ TRÌNH	11

4.1	Đánh giá hoàn thành đồ án	11
4.2	Giao diện chương trình với testcase	12
5	TÀI LIỆU THAM KHẢO	17

1 CHƯƠNG I: MỞ ĐẦU

1.1 Giới thiệu nhóm và phân công công việc

- Số thành viên: 3 người

STT	MSSV	Họ và tên
1	1612840	Dương Nguyễn Thái Bảo
2	1612052	Phạm Minh Chiến
3	1612899	Hoàng Xuân Trường

1.2 Mô tả đồ án

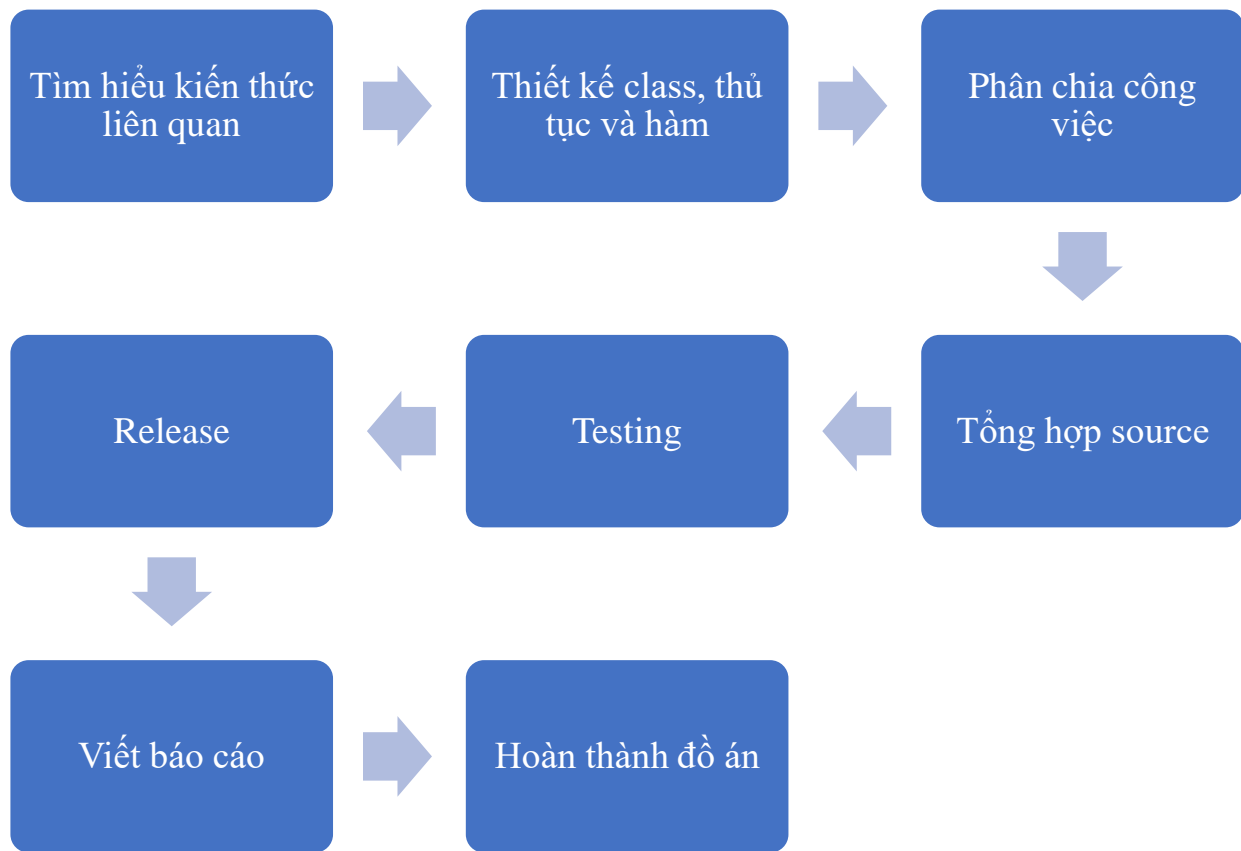
- Biểu diễn và tính toán số học trên máy tính.
 - o Thiết kế kiểu dữ liệu biểu diễn có số nguyên lớn có dấu 16 bytes (128bit).
 - o Thiết kế kiểu dữ liệu biểu diễn số chấm động có độ chính xác Quadruple-precision (độ chính xác gấp 4 lần) độ lớn 128bit.
- Viết chương trình calculator đơn giản gồm các chức năng sau:
 - o Chuyển đổi giá trị qua lại giữa các hệ 2,10,16 (số chấm động không chuyển đổi qua hệ 16)
 - o Tính toán cộng, trừ, nhân, chia giữa 2 số bất kỳ.

2 CHƯƠNG II: GIỚI THIỆU ĐỒ ÁN

2.1 Thời gian thực hiện

- Đồ án bắt đầu từ ngày 10/3 đến ngày 31/3

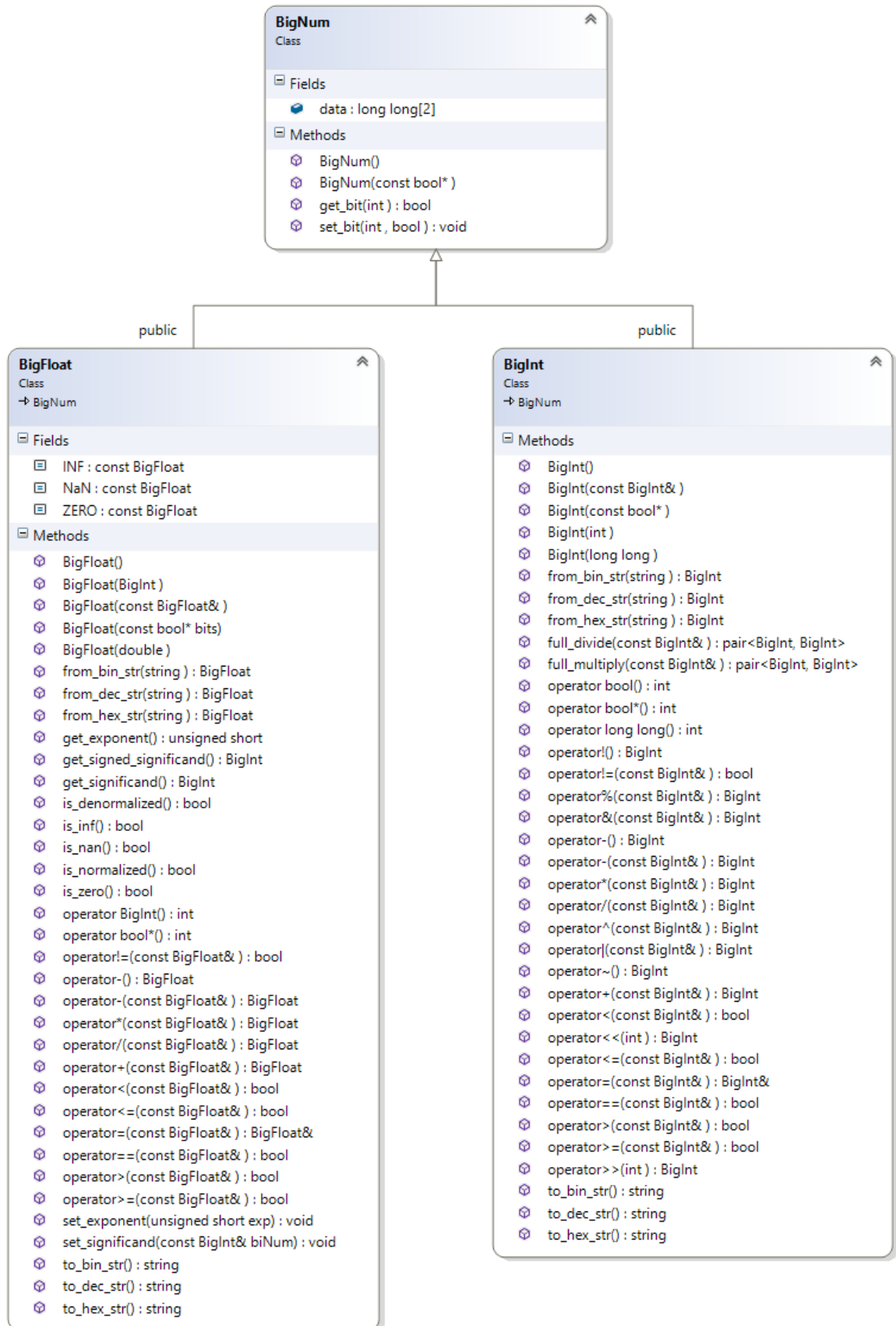
2.2 Các bước thực hiện đồ án



3 CHƯƠNG III: NỘI DUNG ĐỒ ÁN

3.1 Sơ đồ UML

- Sơ đồ UML thể hiện thiết kế class



3.2 Lớp BigInt

- Là lớp số cơ sở của 2 lớp số nguyên lớn và số thực lớn.
- BigInt gồm 2 phần tử kiểu long long để tạo thành 128 bit dữ liệu (kích thước long long là 64 bit).
- Lớp BigInt hỗ trợ 2 phương thức quan trọng là `get_bit()` - dùng để lấy giá trị 1 trong 128 bit, `set_bit()` - dùng để sửa giá trị của 1 bit.

3.2.1 `Get_bit(int index, bool value)`

Thuật toán:

- Xác định được phần tử data cần thay đổi là $\text{index} / 64$, tức là $\text{index} \gg 6$.
- Xác định bit cần thay đổi của $\text{data}[\text{index} \gg 6]$ là $\text{index} \% 64$, tức là $\text{index} \& 63$.
- Như vậy, nếu $\text{value} = \text{true}$ thì ta sẽ áp dụng phép or giữa $\text{data}[\text{index} \gg 6]$ với dãy bit toàn 0, chỉ trừ bit thứ $\text{index} \& 63$ là bật, tức là $1 \ll (\text{index} \& 63)$.
- Ngược lại nếu $\text{value} = \text{false}$ thì and giữa $\text{data}[\text{index} \gg 6]$ với dãy bit toàn 1, chỉ trừ bit thứ $(\text{index} \& 63)$ là 0, tức là phần bù của $1 \ll (\text{index} \& 63)$, hay $\sim(1 \ll (\text{index} \& 63))$.

3.2.2 `Get_bit(int)`

Thuật toán:

- Tương tự như `get_bit`, ta xác định được data cần xét là $\text{index} \gg 6$, và vị trí bit cần lấy là $(\text{index} \& 63)$.
- Để lấy được giá trị bit này, ta lấy $\text{data}[\text{index} \gg 6]$ dịch phải $(\text{index} \& 63)$ bit, lúc này bit cần xét nằm ở vị trí 0 nên chỉ cần and với 1 là xác định được kết quả.

3.3 Lớp BigInteger

- Lớp BigInteger dùng để lưu giá trị của một số nguyên lớn có dấu có 128 bit.
- Lớp BigInteger được kế thừa từ lớp cha BigInt với mục đích sử dụng lại các hàm ví dụ như `get_bit()`, `set_bit()` để dễ dàng hơn trong các thao tác xử lý.
- Phạm vi biểu diễn số trong lớp BigInteger:
 - Từ -2^{127} đến $2^{127} - 1$.
 - Có tối đa 39 chữ số thập phân.

3.3.1 Chuyển giữa hệ thập phân và nhị phân

Từ hệ nhị phân sang chuỗi thập phân: Dựa vào nhận xét số BigInteger có tối đa 39 chữ số thập phân nên chia 39 chữ số này thành 3 phần 13 chữ số và lưu

bằng 3 biến long long, sử dụng các phép / và %: Gọi p là số cần chuyển (kiểu BigInt):

- 13 chữ số nhỏ nhất là $p \% 10^{13}$.
- 13 chữ số tiếp theo là $(p / 10^{13}) \% 10^{13}$.
- 13 chữ số lớn nhất là $p / 10^{13} / 10^{13}$.

Từ chuỗi thập phân sang nhị phân: Ta làm ngược lại ở trên, chuyển từng 13 chữ số thập phân thành 3 số long long rồi nối lại bằng phép * và +.

3.3.2 Chuyển giữa hệ nhị phân và hệ thập lục phân

Gom từng nhóm 4 bit của biểu diễn nhị phân để biểu diễn một chữ số thập lục phân, và ngược lại.

3.3.3 Toán tử cộng

- Dựa trên [Full adder](#), ta chạy từng bit từ trái sang phải trên data[2]. Trong mỗi lần lặp, ta cần tính tổng của 3 bit: 2bit của hai số BigInt cho trước và 1bit của *Carry*. Biến *Carry* được tính bằng cách OR của tất cả các cặp. Thuật toán cụ thể:

```
1. Sum = a XOR b XOR carry; //a và b là hai bit đang xét của hai số BigInt
2. carry = (a & b) | (a & c) | (b & c);
```

3.3.4 Toán tử trừ

- Duyệt từng bit của số BigInt, ta thực hiện phép tính trừ trên hai bit của hai số BigInt và số mượn *borrow*. Mô tả cách thực hiện:

```
1. Kết quả = (a - b - borrow) and 1;
2. If(kết quả < 0) thì borrow = 1 ngược lại borrow = 0;
```

3.3.5 Toán tử nhân

Sử dụng thuật toán Booth.

Trong cài đặt có 2 hàm nhân:

- Hàm `full_multiply` (nhân đầy đủ): trả về đủ 256 bit của kết quả, biểu diễn bằng 2 biến BigInt (1 biến lưu 128 bit cao và 1 biến lưu 128 bit thấp của kết quả). Cài đặt hàm này để ứng dụng trong thuật toán nhân của số thực lớn.
- Toán tử nhân: là 128 byte thấp từ hàm `full_multiply`.

3.3.6 Toán tử chia

Sử dụng thuật toán chia không dấu trong slide bài giảng Số chấm động - thầy Phạm Tuấn Sơn - Khoa Công nghệ Thông tin trường Đại học Khoa học Tự nhiên TP.HCM.

Trước khi nhân, đưa 2 toán hạng về số dương.

Trong cài đặt cũng có 2 phép chia:

- Hàm `full_divide` (chia đầy đủ): trả về phần dư và phần nguyên của kết quả chia.
- Toán tử chia: là phần nguyên từ hàm `full_divide`.
- Ngoài ra còn có toán tử chia lấy phần dư, đây là phần dư từ hàm `full_divide`.

3.3.7 Phép xử lý bit

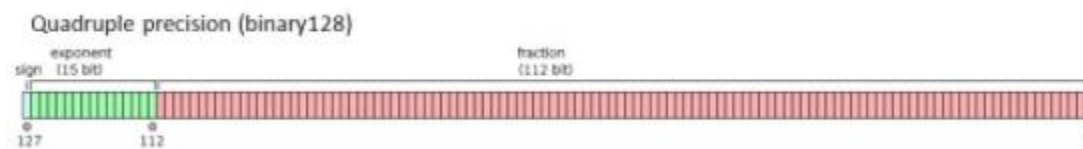
Các phép xử lý bit sử dụng các toán tử xử lý bit có sẵn `&`, `|`, `^`, `~` trên từng phần tử data.

3.3.8 Phép so sánh

- So sánh bằng: 2 số `BigInt` bằng nhau khi data tương ứng bằng nhau.
- So sánh khác: dựa trên so sánh bằng.
- So sánh bé hơn: xét bit đầu tiên khác nhau (từ cao xuống). Nếu bit đó là cao nhất (127) thì số nào có bit 127 là 1 thì nhỏ hơn. Ngược lại 0 thì nhỏ hơn.
- So sánh lớn hơn: tương tự so sánh bé hơn.
- So sánh bé hơn hoặc bằng và lớn hơn hoặc bằng: tận dụng so sánh lớn hơn và bé hơn.

3.4 Lớp `BigFloat`

- Lớp `BigFloat` là kiểu dữ liệu biểu diễn số chấm động có độ chính xác Quadruple-precision độ lớn 128bit.
- Lớp `BigFloat` được kế thừa từ lớp `BigNum`, có quy ước cấu trúc như sau (chuẩn IEEE 754-2008): bit 127 là bit dấu, bit 112 tới 126 là 15 bit mũ (số bias), bit 0 tới 111 là 112 bit phần trị.



- Phạm vi biểu diễn của lớp `BigFloat`:

Số	Dấu	Mũ	Trị
Chuẩn max (+)	0	111111111111110 (= 16383)	1111...111 (= $1 - 2^{-112}$)
Chuẩn min (+)	0	000000000000001 (= -16384)	0000...0000 (= 0)

Chuẩn min (-)	1	111111111111110 (= 16383)	1111...111 (= $1 - 2^{-112}$)
Chuẩn max (-)	1	000000000000001 (= -16384)	0000...0000 (= 0)
Không chuẩn max (+)	0	0000000000000000 (quy ước là -16384)	11111...1111 ($1 - 2^{-112}$)
Không chuẩn min (-)	0	0000000000000000 (quy ước là -16384)	0000....0001 (= 2^{-112})
Không chuẩn min (-)	1	0000000000000000 (quy ước là -16384)	11111...1111 ($1 - 2^{-112}$)
Không chuẩn max (-)	1	0000000000000000 (quy ước là -16384)	0000....0001 (= 2^{-112})
Số vô cực	0	1111111111111111	0
Số 0	0	0000000000000000	0
Số NaN		1111111111111111	Khác 0

3.4.1 Chuyển giữa hệ thập phân và nhị phân

Từ nhị phân sang thập phân:

- Cài đặt phương thức ép kiểu BigFloat sang BigInt để làm tròn.
- Liên tục nhân số cần chuyển với 10, lấy phần nguyên đưa vào kết quả và loại bỏ phần nguyên này, cho đến khi số này trở thành 0.

Từ thập phân sang nhị phân: Tách phần trị ra riêng, chuyển sang nhị phân rồi nhân 10 hoặc chia 10 cho đến khi phần mũ thập phân về 0.

3.4.2 Chuyển giữa hệ nhị phân và hệ thập lục phân

Gom 4 bit nhị phân thành 1 chữ số thập lục phân, và ngược lại.

3.4.3 Toán tử cộng

- Cài đặt hàm get_signed_significand: dùng để lấy phần trị, kết hợp với số 1 (nếu là số chuẩn), lưu ở dạng số có dấu BigInt. Ví dụ: $1.11 \Rightarrow 111000...000$ (113 bit).
- Lấy signed_significand của 2 số, đưa mũ về bằng nhau, đồng thời dịch trái phải tương ứng cho phần signed significand.
- Sau đó cộng 2 phần đó lại với nhau.
- Chuẩn hóa kết quả bằng cách đưa mũ về khoảng cho phép, kết hợp dịch trái dịch phải tương ứng với phần signed significand.
 - Dùng hàm set_exponent và set_significand để khởi tạo kết quả.

3.4.4 Toán tử trừ

- Gọi lại thuật toán cộng.

3.4.5 Toán tử nhân

- Lấy phần `signed_significand`, cộng 2 phần mũ với nhau.
- Nhân 2 phần `signed_significand` bằng hàm `full_multiply` của lớp `BigInt`, sau đó lấy các bit cao nhất (kể từ bit 1 đầu tiên của kết quả này).

3.4.6 Toán tử chia

- Lấy phần `signed_significand`, cộng 2 phần mũ với nhau.
- Liên tục dịch trái `signed_significand` của số bị chia rồi so sánh và trừ `signed_significand` của số chia. Kết quả lấy những bit cao nhất (kể từ lần đầu `signed_significand` của số bị chia lớn hơn) của kết quả.

3.4.7 Phép so sánh

Lấy phần `signed_significand`, đưa 2 số về cùng số mũ rồi so sánh trên phần `signed_significand` sau khi chuẩn hóa.

4 CHƯƠNG V: ĐÁNH GIÁ VÀ TỔNG KẾT QUÁ TRÌNH

4.1 Đánh giá hoàn thành đồ án

Tên lớp	Yêu cầu		Mức độ hoàn thành
BigInt	Hàm nhập		100%
	Hàm xuất		100%
	Chuyển đổi từ hệ thập phân sang hệ nhị phân		100%
	Chuyển đổi từ hệ nhị phân sang hệ thập phân		100%
	Chuyển đổi từ nhị phân sang hệ thập lục phân		100%
	Chuyển đổi từ thập phân sang thập lục phân		100%
	Các phép tính	Cộng	100%
		Trừ	100%
		Nhân	100%

		Chia	Lấy phần nguyên	100%
			Lấy phần dư	100%
BigFloat	Hàm nhập			100%
	Hàm xuất			100%
	Chuyển đổi từ hệ thập phân sang hệ nhị phân			100%
	Chuyển đổi từ hệ nhị phân sang hệ thập phân			100%
	Các phép tính	Cộng		100%
		Trừ		100%
		Nhân		100%
		Chia		100%
			Đánh giá tổng quát: 100%	

4.2 Giao diện chương trình với testcase

- Gồm có 100 testcase

Kiến trúc máy tính và hợp ngữ

The image displays two screenshots of a computer screen, illustrating the execution of assembly code. The left window shows a text editor with assembly code, and the right window shows the command prompt output.

Left Window (Text Editor):

The text editor displays assembly code. The first screenshot shows lines 1 through 14, and the second screenshot shows lines 12 through 24. The code includes hexadecimal values, decimal values, and binary representations.

Right Window (Command Prompt):

The command prompt shows the output of the assembly code execution. It displays results for tests 1 through 22, including hexadecimal values and binary representations.

Assembly Code (Left Window):

1 16 2 22B037215517285138801C518DFCEC
2 16 10 E883F4E12401F3F02CBA3C952B4F5B
3 16 2 876508769A80091FE626278B1D05B6
4 10 16 18945041307199288224336403637367
5 10 2 708946847915651328287193
6 16 10 68DBB1BB4BD4A7F0286D6D7C22A75
7 2 10
000100101101111101101010100010100101010011110
00101000001100110111100100101001011110111010
00100000011100010001001101111100101010
8 2 10
0100110111110110001001101101001000000010001
0100110100011011110110001110000001000010001
00100010110011100011001110011001100101
9 2 10
100011101000011100110101001111001111010010101
01001000100011011100010
10 16 B9B474A520042B4FEE6225 +
C5DE09E2E3BB1C7F97B7C462844CDE
11 16 10 C3B74CAEA318C8D0E398A6EE50F033
12 16 10 C3399E1FE89E8D18A3F4474C8A
13 10 2 -94945752613449537398204590348729
14 10 2492269372675610243184735 +

Command Prompt Output (Right Window):

*Result test 1:
10001010110000001101110010000101010100010110010100001010001001110
0111111001101100
*Result test 2:
1207289268187469620683457194894249819
*Result test 3:
1000011101100101000010000111011010011011000000000100100011111110
11010000010110110110
*Result test 4:
EF1EBADF92A78A51CE8C922477
*Result test 5:
100101100010000000010101111011000010000111100111000111101100101010
*Result test 6:
544454752950182628161069698089364085
*Result test 7:
25086147018774568094066132733302988586
*Result test 8:
103654215871837624337880656699619862117
*Result test 9:
164323776178573035234
*Result test 10:
C5DE09E39D6F912487BBEF8272AF03
*Result test 11:
1016215643759674386048358589840551987
*Result test 12:
15467323496805292020049769876618
*Result test 13:
10010101110011000100010111001110010110010101010111011
10111001
*Result test 14:
2492269349235421052590161
*Result test 15:
56514051001000925378986974736688
*Result test 16:
210079872684108820183669850533119072
*Result test 17:
-110101000001110011100001001011000001110000101010110010000100110
0111000
*Result test 18:
-1101111011100101001111001000011101101000001101010000000110110111
000100011011011001111110
*Result test 19:
-1001100101010101001001110101110101100000110010100011110110
100001100101000100101111110
*Result test 20:
-86304327708315866994928277149579627955
*Result test 21:
849424596627611567803676441398249865
*Result test 22:

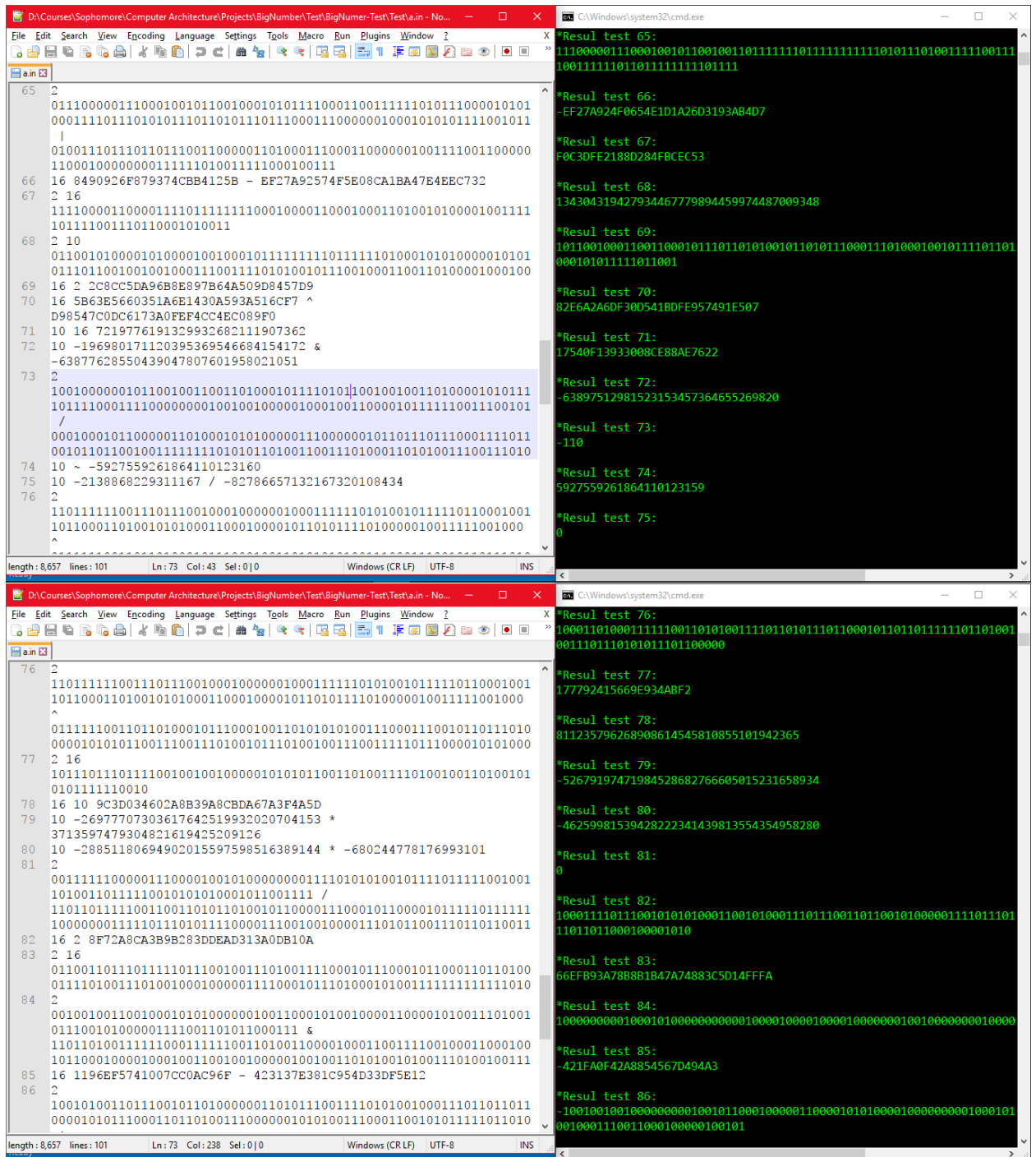
Kiến trúc máy tính và hợp ngữ

The image shows a Windows desktop with two Notepad++ windows open. The left window, titled 'D:\Courses\Sophomore\Computer Architecture\Projects\BigNumber\Test\BigNumer-Test\Test\ain - No...', displays a file named 'a.in'. The file contains a sequence of numbers and binary strings, including a long decimal number '10 -9105928424755024673180212123681 << 1' and a long binary string '10101100101010001000011000111010110100010100110000100000110101'. The right window, titled 'C:\Windows\system32\cmd.exe', displays the output of a program. It shows results for 32 tests, with each test result preceded by '*Resul test [number]:'. The output includes binary strings and decimal numbers, some of which are highlighted in green, such as '182118568495100493463604242487362' and '11010010101100000110001000111100011100011000100100110011010110000110'.

Kiến trúc máy tính và hợp ngữ

[illegible]

Kiến trúc máy tính và hợp ngữ



The image displays two screenshots of a development environment. The left screenshot shows a code editor with assembly code for a program named 'BigNumber-Test'. The code includes various instructions like 'mov', 'add', 'sub', 'mul', 'div', and 'print', along with hexadecimal and decimal constants. The right screenshot shows a command prompt window displaying the output of the program, which consists of a series of test results, each preceded by a label like '*Resul test 65:'. The output shows the results of various arithmetic operations performed on large numbers represented in binary and hexadecimal formats.

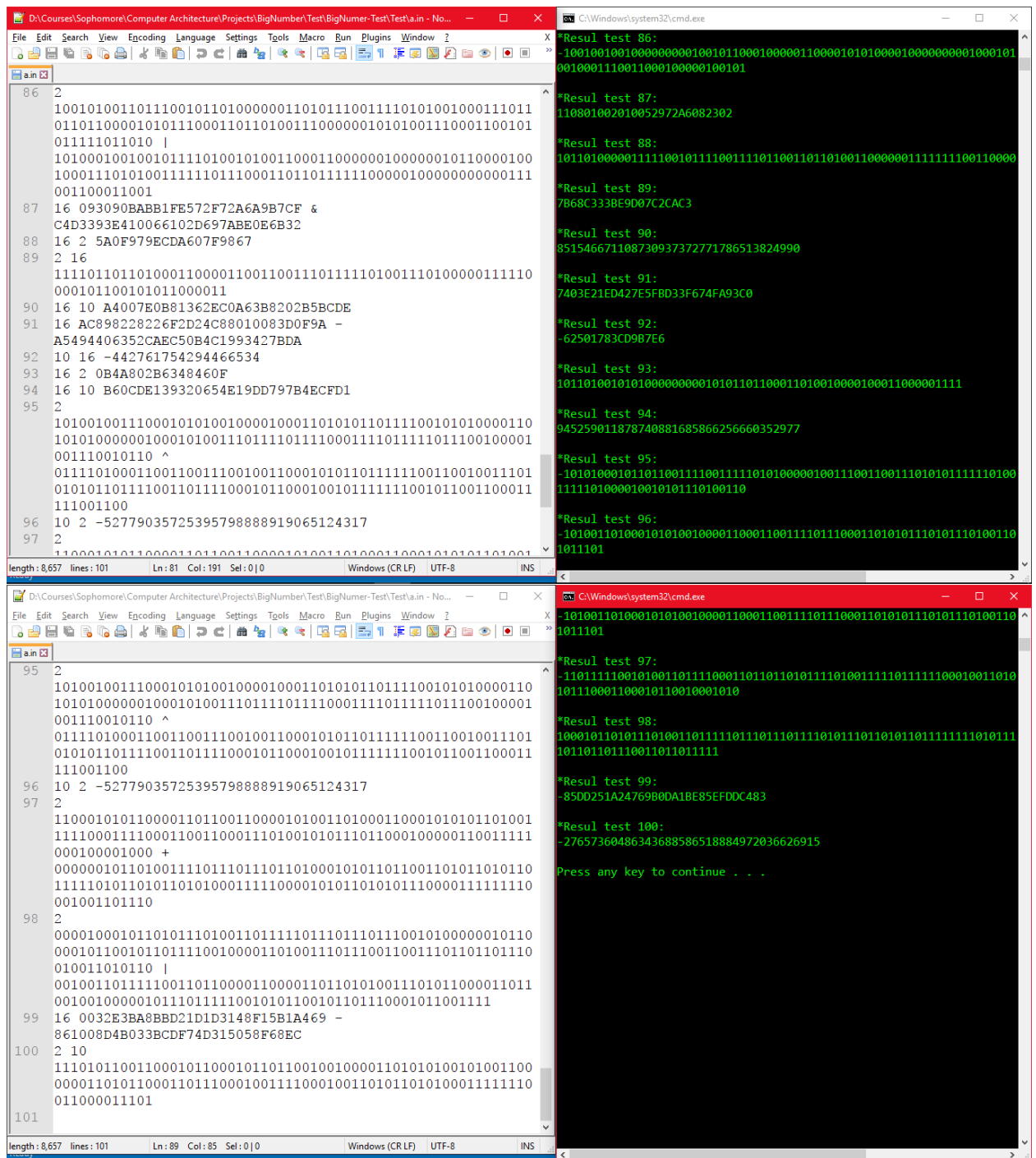
```
length: 8,657 lines: 101 Ln: 73 Col: 43 Sel: 0|0 Windows (CR LF) UTF-8 INS
```

```
*Resul test 65:
11100000111000100101100100010101111000110011111010111000010101
1001111011101010111010101110001110000001000101010111001011
100111111011011111101111
*Resul test 66:
-EF27A924F0654E1D1A26D3193AB4D7
*Resul test 67:
F0C3DFE2188D284FBCEC53
*Resul test 68:
134304319427934467779894459974487009348
*Resul test 69:
10110010001100110010110101001011010111000111010001001011101101
00010101111011001
*Resul test 70:
82E6A2A6DF30D5418DFE957491E507
*Resul test 71:
17540F13933008CE88AE7622
*Resul test 72:
-63897512981523153457364655269820
*Resul test 73:
-110
*Resul test 74:
5927559261864110123159
*Resul test 75:
0
```

```
length: 8,657 lines: 101 Ln: 73 Col: 238 Sel: 0|0 Windows (CR LF) UTF-8 INS
```

```
*Resul test 76:
100010100011111100110101001111010111011000101101101111101101001
001101110101110110100000
*Resul test 77:
177792415669E934ABF2
*Resul test 78:
811235796268908614545810855101942365
*Resul test 79:
-52679197471984528682766605015231658934
*Resul test 80:
-4625998153942822341439813554354958280
*Resul test 81:
0
*Resul test 82:
1000111011100101010001100101000111011100110110010100000111011101
1101101100010001010
*Resul test 83:
66EFB93A78B8B1B47A74883C5D14FFFA
*Resul test 84:
1000000001000101000000000100010000100001000100000001000000010000
*Resul test 85:
-421FA0F42A8854567D49A3
*Resul test 86:
-10010010010000000010010100010000010000101010001000000001000101
0010001100110001000001001
```

Kiến trúc máy tính và hợp ngữ



```
86 2
1001010011011100101101000000110101110011110101001000111011
0110110000101011100011011010011100000010101001110001100101
01111011010 |
1010001001001011110100101001100011000000100000010110000100
1000111010100111110111000110110111110000010000000000111
001100011001
87 16 093090BABB1FE572F72A6A9B7CF &
C4D3393E410066102D697ABE0E6B32
88 16 2 5A0F979ECDA607F9867
89 2 16
11110110110100011000011001100111011110100111010000011110
000101100101011000011
90 16 10 A4007E0B81362EC0A63B8202B5BCDE
91 16 AC898228226F2D24C88010083D0F9A -
A5494406352CAEC50B4C1993427BDA
92 10 16 -442761754294466534
93 16 2 0B4A802B6348460F
94 16 10 B60CDE139320654E19DD797B4ECFD1
95 2
1010010011100010101001000010001101010110111100101010000110
10101000000100010100111011110111100011101111011100100001
001110010110 ^
01111010001100110011100100110010101111100110010011101
010101101111001101111000101100010010111111001011001100011
111001100
96 10 2 -52779035725395798888919065124317
97 2
1100010101100001101100110000101001101000110001010101101001
length: 8,657 lines: 101 Ln: 81 Col: 191 Sel: 0|0 Windows (CR LF) UTF-8 INS

C:\Windows\system32\cmd.exe
*Result test 86:
-10010010010000000001001011000100000110000101010000100000000100101
0010001110011000100000100101
*Result test 87:
110801002010052972A6082302
*Result test 88:
10110100000111110010111100111101100110110011000000111111100110000
*Result test 89:
7B68C33BE9D07C2CAC3
*Result test 90:
851546671108730937372771786513824990
*Result test 91:
7403E21ED427E5FBD33F674FA93C0
*Result test 92:
-62501783CD9B7E6
*Result test 93:
10110100101010000000010101101100011010010000100011000001111
*Result test 94:
94525901187840881685866256660352977
*Result test 95:
-10101000101011001111001111010100000100110011001101010111110100
1111101000010010101110100110
*Result test 96:
-10100110100010101001000011000110011110111000110101110101110100110
1011101
-10100110100010101001000011000110011110111000110101110101110100110
1011101
*Result test 97:
-11011111001010011011110001101101011110100111101111100010011010
101110001100010110010001010
*Result test 98:
100010110101110100110111101110111011101110110110111111010111
1011011011100110111111
*Result test 99:
-850D251A24769B00A1BE85EFD0C483
*Result test 100:
-27657360486343688586518884972036626915
Press any key to continue . . .
```

5 TÀI LIỆU THAM KHẢO

[https://en.wikipedia.org/wiki/Adder %28electronics%29#Full_adder](https://en.wikipedia.org/wiki/Adder_%28electronics%29#Full_adder)