

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 nhất 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	3
1.1 Giới thiệu nhóm và phân công công việc.....	3
1.2 Mô tả đồ án.....	3
2 CHƯƠNG II: GIỚI THIỆU ĐỒ ÁN	3
2.1 Thời gian thực hiện	3
2.2 Các bước thực hiện đồ án.....	3
3 CHƯƠNG III: NỘI DUNG ĐỒ ÁN.....	4
3.1 Sơ đồ UML.....	4
3.2 Lớp BigInt.....	6
3.2.1 Hàm get_bit(int index, bool value)	6
3.2.2 Hàm get_bit(int).....	6
3.3 Lớp BigInt	6
3.3.1 Chuyển giữa hệ thập phân và nhị phân	6
3.3.2 Chuyển giữa hệ nhị phân sang hệ thập lục phân.....	7
3.3.3 Toán tử cộng	7
3.3.4 Toán tử trừ	7
3.3.5 Toán tử nhân	7
3.3.6 Toán tử chia	8
3.4 Lớp BigInt	8
3.4.1 Chuyển giữa hệ thập phân và nhị phân	9
3.4.2 Hàm chuyển từ hệ nhị phân sang hệ thập lục phân.....	9
3.4.3 Toán tử cộng	9
3.4.4 Toán tử trừ	10
3.4.5 Toán tử nhân	10
3.4.6 Toán tử chia	10
4 CHƯƠNG V: ĐÁNH GIÁ VÀ TỔNG KẾT QUÁ TRÌNH.....	10
4.1 Đánh giá hoàn thành đồ án	10
4.2 Giao diện chương trình với testcase	11
5 TÀI LIỆU THAM KHẢO	16

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

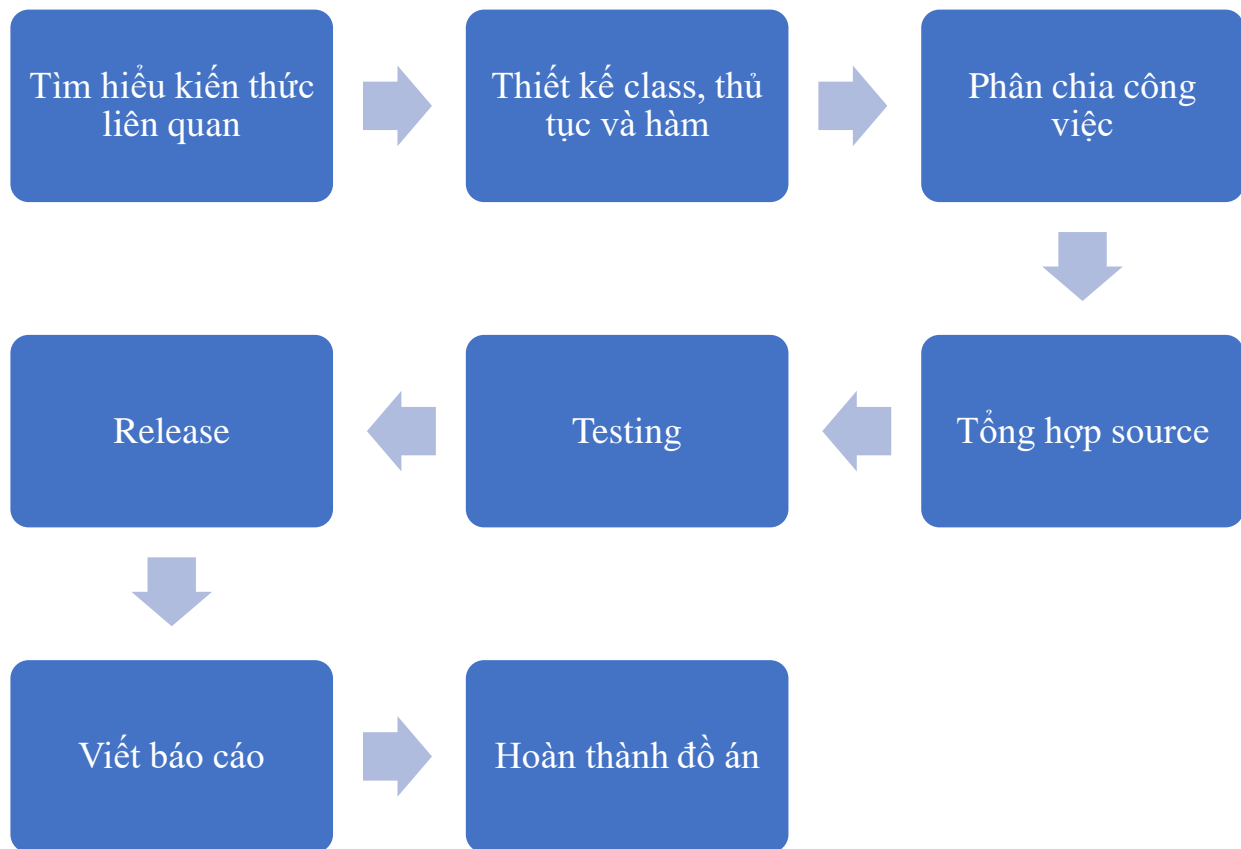
- Biểu diễn và tính toán số học trên máy tính.
 - o Thiết kế kiểu dữ liệu biểu diễn có số nguyên lớn có dấu 16 bytes (128bit).
 - o Thiết kế kiểu dữ liệu biểu diễn số chấm động có độ chính xác Quadruple-precision (độ chính xác gấp 4 lần) độ lớn 128bit.
- Viết chương trình calculator đơn giản gồm các 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

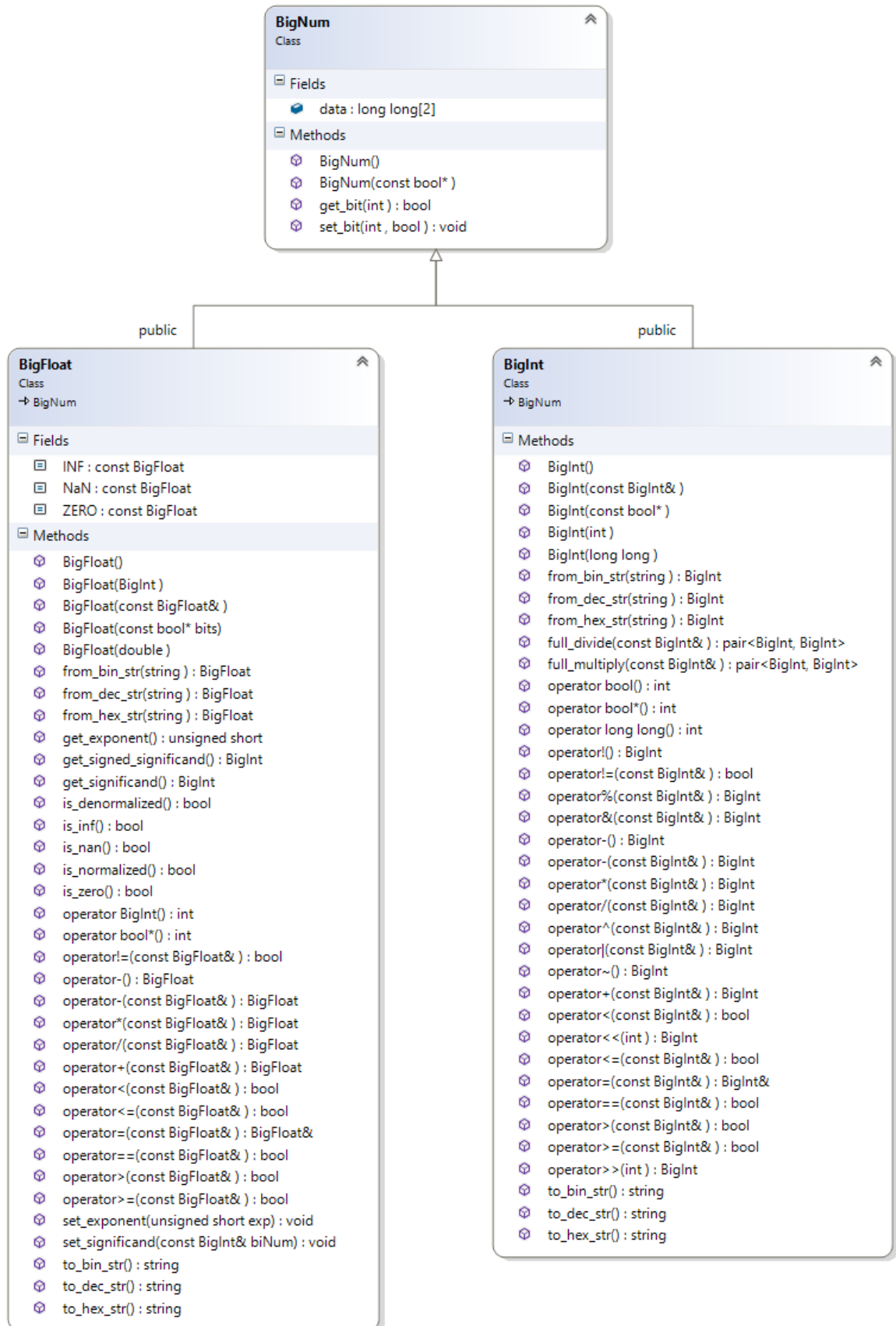
2.2 Các bước thực hiện đồ án



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 Hàm `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 `data[index >> 6]` là $\text{index} \% 64$, tức là $\text{index} \& 63$.
- Như vậy, nếu `value = true` thì ta sẽ áp dụng phép or giữa `data[index >> 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 `value = false` thì or giữa `data[index >> 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 Hàm `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 `data[index >> 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 sang 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

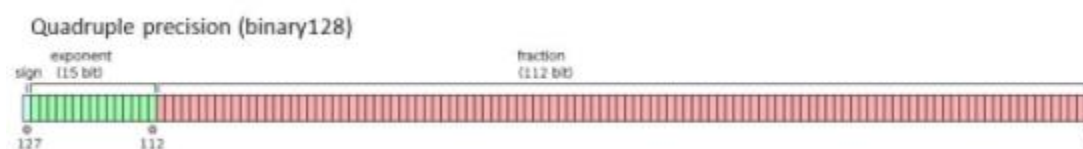
Các phép xử lý bit sử dụng các toán tử xử lý bit có sẵn `&`, `|`, `^`, `~` trên từng phần tử data.

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	0000000000000001 (=-16384)	0000...0000 (= 0)
Chuẩn min (-)	1	1111111111111110 (= 16383)	1111...111 (= $1 - 2^{-112}$)
Chuẩn max (-)	1	0000000000000001 (=-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 Hàm chuyển từ hệ nhị phân sang 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 => 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



```
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
01001101000110111101100011110000001000010001
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 +

length: 8,657 lines: 101 Ln: 9 Col: 74 Sel: 0|0 Windows (CR LF) UTF-8 INS

*Result test 1:
10001010110000001101110010000101010100010110010100001010001001110
01111110011101100

*Result test 2:
1207289268187469620683457194894249819

*Result test 3:
10000111011001010000100001110110100110110000000001001000111111110
11010000010110110110

*Result test 4:
EF1EBADF92A78A51CE8C922477

*Result test 5:
1001011000100000000101011110110000100001111001110001111011100101010

*Result test 6:
544454752950182628161069098089364085

*Result test 7:
25086147018774568094066132733302988586

*Result test 8:
103654215871837624337880656699619862117

*Result test 9:
164323776178573035234

*Result test 10:
C5DE09E39D6F912487BBEF8272AF03

*Result test 11:
1016215643759674386048358589840551987

12 16 10 C3399E1FE89E8D18A3F4474C8A
13 10 2 -94945752613449537398204590348729
14 10 2492269372675610243184735 + -23440189190594574
15 10 56514051001042769323047716624726 ^
116207664003497781350
16 16 10 2875BCC6CBE7381EDD818DC8C9C460
17 10 2 -74176401439756444056200741243576
18 2
1110001001100001001101000111011010001010000000111
011010000100111110110101100010010010 -
000110111110110010100111100100001111011010111100
0100000001011010110110110000101110001000011111100
101001010000100100010110010000
19 2
1011101110010101110011111101010100001011100110110
0010010001010011100011011001111111110011010011101
010100010001110101101101000111 *
011100010101011110010000010010000000111110001001
10101001010110110001100001111001111101110
20 2 10
101111100010010011001000001111010110110011000100
000101010111110010001111100010100110011010000000
111111101010011011001001001101
21 16 10 A397DDB30C094340091292AD408989
22 10 -91059284247550246731802121243681 << 1
23 10 16 63105068371767627977311516949108
24 16 10 939AAC44817209F6675EAEAE783EE8

length: 8,657 lines: 101 Ln: 19 Col: 38 Sel: 0|0 Windows (CR LF) UTF-8 INS

*Result test 12:
15467323496805292020049769876618

*Result test 13:
-10010101110011000100010111001110010110010100110010101111011
10111001

*Result test 14:
2492269349235421052590161

*Result test 15:
56514051001000925378986974736688

*Result test 16:
210079872684108820183669850533119072

*Result test 17:
-1101010000011100111000010010110000101010110010000100110
0111000

*Result test 18:
-11011110111001010011110010000111011010000011010110111
0001000110111011001111110

*Result test 19:
-10011001010101010100100111101011100000110010100011110110
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\A.in - No...', displays a file named 'a.in'. The file contains a sequence of numbers and binary strings, with some lines highlighted in blue. The right window, titled 'C:\Windows\system32\cmd.exe', displays the output of a program, showing results for 32 tests. The output is a long string of numbers and binary strings, with some lines highlighted in green.

Left Window (a.in):

```
22 10 -9105928424755024673180212123681 << 1
23 10 16 63105068371767627977311516949108
24 16 10 939AAC44817209F6675EAE783EE8
25 2 10
000010010001100011110101010010011001101110010010111
10101100101010001000011000111010101000101001100001000011
01
26 16 2 B2B41C47C0F1C6499D92FD277DD786
27 2
00001111111011110111100011000010110010001011000000111000
100010100110011011010001110111111011110000010001111010
010111000110 ^
111010100101010101010101110111011101101110111001101101
1011000101010000010101110001100010010010101010001101001
01010001
28 2
100000101010100100000001000111001101101001011101101110110
1111000111110000011100 +
0110100101010011000000010100110100110100110101110001111
1101110111010010001010000011000000110010000010000010100
011011101010
29 10 2 49082708443894389892
30 2
00101100011110110100101101001000000010101110010100000001
11000011 -
10100011001010101011111110110110110001101110000000101001
1101110011000111110101011100001110010010
31 16 2 69877B6141BA2E1
10 762274791023557497910848303755 /
-66533727505066272124709593080942
16 756330097155CFC5DE088C36AA50A /
59437598704BD30E229B291D194E3E
16 137E3616AA2E19B4E2FC5BA6 |
2E4EC3A22A954344CC1A74E91
16 94044E2D64BD52BF422935CE35E7D8 |
337FC70DDA970861C2CC0B78989AA2
10 16 95092073547211027533615517473023
10 616033269074703289 -
34259506915032157941955452609439
10 ~ 22248712075166530227504266166925
16 10 BA990652092764D6C7350
10 -4146256647942562288831460883355 *
-23834115431360230510005568853733
2 10
100110100011100010001110000000111011100000010
0101001110011011101011101001001010101101110111
10101001110101010001110111011000110101
16 2 AC5F1DF5202AA0A9B7C21E92853
10 -8295157940082033638286115599721 |
-66287038714561555012865
16 A9AD4D4BFA8F757116A97B3F00404E >> 5
```

Right Window (cmd.exe):

```
*Result test 22:
-182118568495100493463604242487362

*Result test 23:
31C7F791C67D67AD4CD7674FA74

*Result test 24:
766404777729005104242096726691299048

*Result test 25:
11809228205499847437201080998740493

*Result test 26:
101100101010100000111000100011110001110001100010011001
1101110101110000110

*Result test 27:
101010010101000111110101010101000010111110101010001011100100
01011100110010010111

*Result test 28:
1101001010100110000000101001010010100101010010101011111
1110001111100001010000110

*Result test 29:
1010101001001010001100111101000111010001000110010000100

*Result test 30:
-10100011001101010111111101010101010000101001010100001110100
11

*Result test 31:
1101001100001110111011011000010100000110111010001011100001

*Result test 32:

*Result test 33:
0

*Result test 34:
2F7FE3E36AB7E3DFCE3FF5FB7

*Result test 35:
B77FCF2DFEBF5AFFC2ED3FFEBDFFFA

*Result test 36:
4B03B0FA54DE0D1851FC9918FF

*Result test 37:
-34259506915031541908686377906150

*Result test 38:
-22248712075166530227504266166926

*Result test 39:
14098927569497739026527056

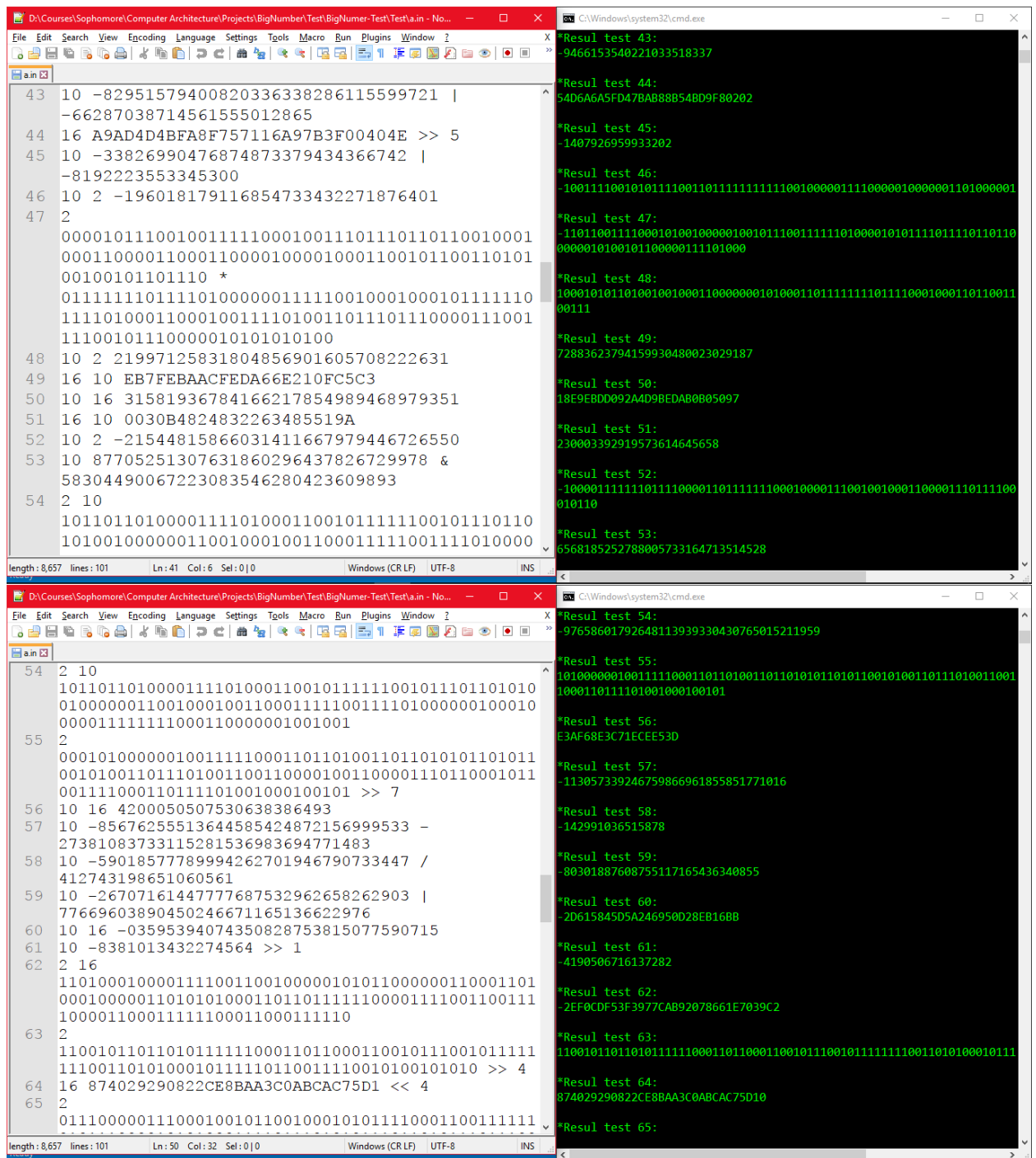
*Result test 40:
115145133650009576727582509758051509159

*Result test 41:
-135287606549167519362813928984909744587

*Result test 42:
10101100010111110001110111110101001000000101010100000101010011011
01010011

*Result test 43:
```

Kiến trúc máy tính và hợp ngữ



The image displays two side-by-side windows from a Windows operating system. The left window is a text editor titled "D:\Courses\Sophomore\Computer Architecture\Projects\BigNumber\Test\BigNumber-Test\Test\Main - No...", showing assembly code. The right window is a command prompt titled "C:\Windows\system32\cmd.exe", showing the output of the assembly code execution.

Left Window (Assembly Code):

```
43 10 -82951579400820336338286115599721 |  
-66287038714561555012865  
44 16 A9AD4D4BFA8F757116A97B3F00404E >> 5  
45 10 -33826990476874873379434366742 |  
-8192223553345300  
46 10 2 -196018179116854733432271876401  
47 2  
00001011100100111110001001110110110110010001  
000110000110001100001000010001100101100110101  
00100101101110 *  
0111111011101000000111110010001000101111110  
111010001100010011101001101101110000111001  
11100101110000010101010100  
48 10 2 21997125831804856901605708222631  
49 16 10 EB7FEBAA CFDA66E210FC5C3  
50 10 16 31581936784166217854989468979351  
51 16 10 0030B4824832263485519A  
52 10 2 -21544815866031411667979446726550  
53 10 87705251307631860296437826729978 &  
58304490067223083546280423609893  
54 2 10  
10110110100001111010001100101111100101110110  
101001000000110010001001100011111001111010000
```

Right Window (Execution Results):

```
*Result test 43:  
-9466153540221033518337  
  
*Result test 44:  
54D6A6A5FD47B8B88B548D9F80202  
  
*Result test 45:  
-1407926959933202  
  
*Result test 46:  
-10011100101111001101111111111001000011110000010000001101000001  
  
*Result test 47:  
-11011001111000101001000001001110011111101000010101110110110110  
0000010100101100000111101000  
  
*Result test 48:  
1000101011010010001100000001010011011111101110001000110110011  
00111  
  
*Result test 49:  
72883623794159930480023029187  
  
*Result test 50:  
18E9EBD0092A4D9BEDAB0B05097  
  
*Result test 51:  
230003392919573614645658  
  
*Result test 52:  
-10000111111011110000110111111100010000111001000110000111011100  
010110  
  
*Result test 53:  
6568185252788005733164713514528  
  
*Result test 54:  
-976586017926481113939330430765015211959  
  
*Result test 55:  
1010000001001111100011011010011011010101010101010101011011010011001  
1000110111101001000100101  
  
*Result test 56:  
E3AF68E3C71ECE53D  
  
*Result test 57:  
-113057339246759866961855851771016  
  
*Result test 58:  
-142991036515878  
  
*Result test 59:  
-80301887608755117165436340855  
  
*Result test 60:  
2D615845D5A246950D28E816BB  
  
*Result test 61:  
-4190506716137282  
  
*Result test 62:  
-2EF0CDF53F3977CAB92078661E7039C2  
  
*Result test 63:  
11001011011010111111000110110001100101110010111111100110100010111  
  
*Result test 64:  
874029290822CE8BAA3C0ABCAC75D1  
  
*Result test 65:
```


Kiến trúc máy tính và hợp ngữ

The screenshot displays two side-by-side Windows Command Prompt windows. The left window shows the assembly code from a file named 'Test\BigNumber-Test\Bin\Win32\x86\bin\test.exe'. The right window shows the corresponding execution output.

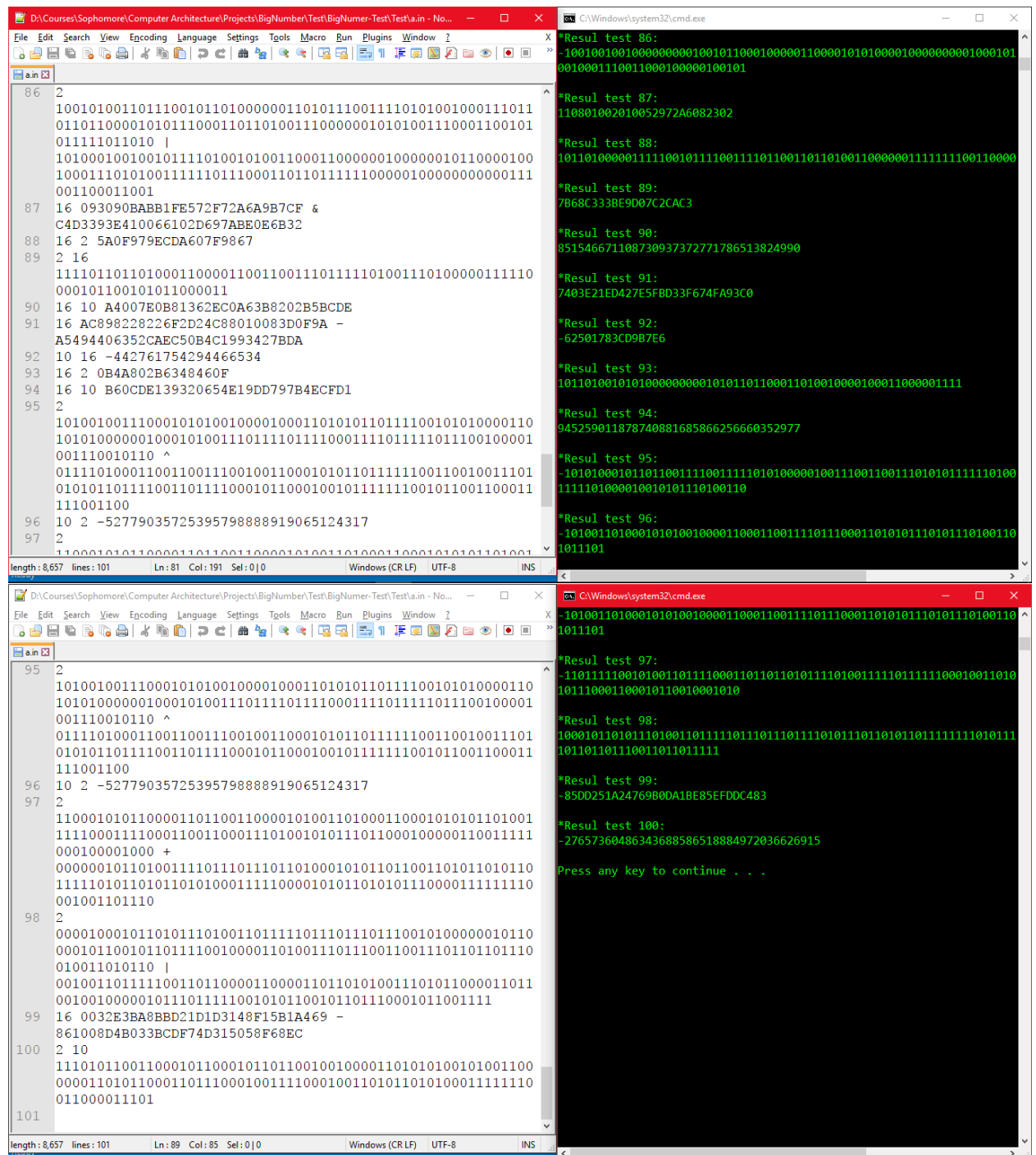
Left Window (Assembly Code):

```
D:\Courses\Sophomore\Computer Architecture\Projects\BigNumber\Test\BigNumber-Test\Bin\Win32\x86\bin\test.exe
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
[Icons]
65 2
011100000111000100101100100010101111000110011111010111000010101
0001111011010101110101011101100011100000010001010101111001011
|
01001110110101110011000100000101000111000110000001001111001100000
11000100000000011111010011111000100111
66 16 8490926F879374CBB4125B - EF27A92574F5E08CA1BA47E4EEC732
67 2 16
1111000011000011101111110001000011000100011010010100001001111
101111001110110001010011
68 2 10
0110010100001010000100100010111111101111101000101010000010101
011101001001001000111001110101001011001000100110100001000100
69 16 2 2C8CC5DA96B8E897B64A509D8457D9
70 16 5B63E5660351A6E1430A593A516CF7 ^
D98547C0DC6173A0FEF4CC4EC089F0
71 10 16 7219776191329932682111907362
72 10 ~ -196980171120395369546684154172 &
-63877628550439047807601958021051
73 2
10010000001011001001100110100010111010110010010011010000101011
10111100011110000000010010010000100010011000010111110011100101
/
000100010110000011010001010100001110000001011011011000111011
0010101100100111111101010110100110011101000101010011100111010
74 10 ~ -5927559261864110123160
75 10 ~ -2138868229311167 / -82786657132167320108434
76 2
1101111100111011100100010000001000111110101001011110110001001
1011000110100101010001100010000101101011101000001001111001000
^
length : 8,657 lines: 101 Ln: 73 Col: 43 Sel: 0|0 Windows (CR LF) UTF-8 INS
```

Right Window (Execution Output):

```
C:\Windows\system32\cmd.exe
*Resul test 65:
1110000011100010010110010010111111011111110111110111011100111100111
1001111011011111111101111
*Resul test 66:
-EF27A924F0654E1D1A26D3193AB4D7
*Resul test 67:
F0C3DFE2188D284FBCE53
*Resul test 68:
134304319427934467779894459974487009348
*Resul test 69:
1011001000110011000101110110100101101011000111010001001011101101
00010101111011001
*Resul test 70:
82E6A2A6DF30D541BD0FE957491E507
*Resul test 71:
17540F13933008CE88AE7622
*Resul test 72:
-63897512981523153457364655269820
*Resul test 73:
-110
*Resul test 74:
5927559261864110123159
*Resul test 75:
0
*Resul test 76:
10001101000111111001101000111101101101101100010110111111011001
001110110101011101100000
*Resul test 77:
177792415669E934ABF2
*Resul test 78:
811235796268908614545810855101942365
*Resul test 79:
-52679197471984528682766605015231658934
*Resul test 80:
-46259981539428222341439813554354958280
*Resul test 81:
0
*Resul test 82:
100011101110010101000011001100011101100110100000111011101
11011011000100001010
*Resul test 83:
66EFB93A78BBB1B47A74883CD14FFFA
*Resul test 84:
1000000001000101000000000010001000010000100000001000000010000
*Resul test 85:
-421FA0F42A8854567D49A3
*Resul test 86:
-1001001001000000000010010100010000101010001000000001000101
0010001110011000100000100101
length : 8,657 lines: 101 Ln: 73 Col: 238 Sel: 0|0 Windows (CR LF) UTF-8 INS
```


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
111101101101000110000110011001110111101001110100000111110
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 ^
011110100011001100111001001100101011111100110010011101
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:
-100100100100000000010010110001000001100001010100001000000001000101
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:
-101010001011011001111010100000100110011001101010111110100
1111101000010010101110100110
*Result test 96:
-10100110100010101001000011000110011110111000110101110101110100110
1011101
-10100110100010101001000011000110011110111000110101110101110100110
1011101
*Result test 97:
-11011111001010011011110001101101011110100111101111100010011010
101110001100010110010001010
*Result test 98:
1000101101011101001101111011101110111011101101111111010111
1011011011100110111111
*Result test 99:
-85DD251A24769B00A1BE85EFD0C483
*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)