

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO ĐỒ ÁN 1

Bộ môn: Kiến trúc máy tính và Hợp ngữ

Giảng viên bộ môn: Lê Quốc Hòa

Thành viên nhóm:

18120356: Phan Anh Hào

18120372: Lê Minh Hiếu

18120376: Phạm Trung Hiếu

Hồ Chí Minh, ngày 13 tháng 06 năm 2020

NỘI DUNG

A. Phân chia công việc và đánh giá mức độ hoàn thành.....	3
B. Tổng quan đề án	4
I. Môi trường lập trình và ý tưởng thiết kế	4
II. Phân tích các chức năng	9
1. Chuyển đổi giữa các hệ số	9
2. Các phép toán trên QInt	10
a. <i>Toán tử AND(&), OR(), XOR(^), NOT(~), SHIL(<<),SHIR(>>)</i>	10
b. <i>Toán tử CỘNG(+), TRỪ(-), NHÂN(*), CHIA(/), BẰNG(=)</i>	11
c. <i>Phép XOAY TRÁI(rol), XOAY PHẢI(ror)</i>	12
III. Đánh giá mức độ hoàn thành của đề án	13
IV. Tài liệu tham khảo	14

A. PHÂN CHIA CÔNG VIỆC VÀ MỨC ĐỘ HOÀN THÀNH

STT	Họ và tên	MSSV	Phân công	Mức độ hoàn thành
1	Phan Anh Hào	18120356	_ Chuyển đổi số QInt từ hệ 2 sang 16 và ngược lại. _ Chuyển đổi số QInt từ hệ 16 sang 10 và ngược lại. _ Các opertor =,+,-,*,/ trên các hệ số.	Hoàn thành tốt
2	Lê Minh Hiếu	18120372	_ Viết báo cáo đồ án. _ Chuyển đổi số QInt từ hệ 2 sang 10 và ngược lại. _ Hàm main để đọc dữ liệu và thực hiện các phép tính.	Hoàn thành tốt
3	Phạm Trung Hiếu	18120376	_ Các toán tử & ^ ~ << >>. _ Các phép xoay rol, ror. _ Optimize code.	Hoàn thành tốt

B. TỔNG QUAN ĐỒ ÁN

I. MÔI TRƯỜNG LẬP TRÌNH VÀ Ý TƯỞNG THIẾT KẾ:

1. Môi trường lập trình:

Microsoft Visual C++ 2015-2019 Redistributable.

2. Ý tưởng thiết kế:

- Sử dụng 2 biến kiểu long long là `_low` và `_high` để biểu diễn số QInt, trong đó biến `_low` chứa 64 bit đầu tiên (0 - 63) còn biến `_high` chứa 64 bit còn lại (64 - 127).

- Phạm vi biểu diễn: $-2^{127} \rightarrow 2^{127} - 1$

- **Chuyển từ hệ 2 sang QInt:**

- _ Gọi hàm `QInt(string str, int radix)` với `str` là dãy bit hệ 2 truyền vào, `radix` là cơ số 2.

- _ Chuẩn hóa dãy bit `str` vừa đưa vào bằng việc thêm vào các bit 0 sao cho đủ 128 bit. Các bit từ $i = 0 - 63$ của dãy bit `string` thuộc phần `_high` nên ta lấy lần lượt 2^{63-i} rồi cộng vào `_high`. Các bit từ $i = 64 - 127$ thuộc phần `_low` nên ta lấy lần lượt 2^{127-i} rồi cộng vào `low`.

- **Chuyển từ hệ 10 sang QInt:**

- _ Gọi hàm `QInt(string str, int radix)` với `str` là dãy string hệ 10 truyền vào, `radix` là cơ số 10.

- _ Đầu tiên ta chuyển về hệ 2 trước: Lần lượt lấy phần dư khi chia dãy `str` ở hệ 10 cho 2 bằng hàm `getDevinedBy2(string str)`, sau đó cộng vào dãy string bit của hệ 2. Nếu số ở hệ 10 là số âm thì ta chuyển dãy bit về dạng bù 2.

- _ Tiếp theo, thực hiện chuyển từ hệ 2 sang QInt.

- **Chuyển từ hệ 16 sang QInt:**

- _ Gọi hàm `QInt(string str, int radix)` với `str` là dãy bit hệ 16 truyền vào, `radix` là cơ số 16.

- _ Đầu tiên ta chuyển về hệ 2 trước: Lần lượt so các bit của dãy bit `str` hệ 16 và chuyển thành 4 bit hệ 2 dựa vào bảng map quy định các giá trị chuyển đổi từ hệ 16 sang hệ 2, sau đó cộng lần lượt vào dãy bit hệ 2.

- _ Tiếp theo, thực hiện chuyển từ hệ 2 sang QInt.

- **Chuyển đổi QInt sang hệ 2:**

- _ Gọi hàm `string toBinary()`

- _ Tạo 2 mảng `str1`, `str2`. Lần lượt chia lấy dư các phần `_high` và `_low` của QInt, sau đó lưu lần lượt vào `str1` và `str2`.

_ Nối mảng str2 vào str1, ta được dãy bit hệ 2. Nếu `_high < 0`, tức là số âm thì thêm 1 bước chuyển dãy bit hệ 2 qua dạng bù 2.

Cắt bỏ hết các bit 0 thừa ở đầu, ta được dãy bit hệ 2 hoàn chỉnh.

- **Chuyển đổi QInt từ hệ sang hệ 10:**

_ Gọi hàm `string toDecimal()`

_ Hàm chuyển `Qint > 0` sang hệ 2 rồi chuyển hệ 2 sang hệ 10 bằng phương pháp nhân. Nếu `Qint < 0` thì lấy đối số rồi thực hiện như `Qint > 0`, cuối cùng thêm “-” vào trước chuỗi.

- **Chuyển đổi QInt từ hệ sang hệ 16:**

_ Gọi hàm `string toHexa()`

_ Hàm chuyển `Qint` sang hệ 2, sau đó chuyển từng substring có chiều dài bằng 4 thành hệ 10, sau đó ánh xạ sang hệ 16.

- **Phép gán =:**

_ Gọi hàm `operator=(const QInt& q)`

_ Gán 1 số `QInt` `q` cho 1 số `QInt` `a` gọi hàm `operator=`, tức là gán `_high` và `_low` của `q` cho `_high` và `_low` của `a`.

- **Phép cộng(+):**

_ Gọi hàm `operator+(const QInt& q1, const QInt& q2)`

_ Đầu tiên, chuyển số `QInt` `q1` và `q2` về dãy bit hệ 2 `str1` và `str2`. Tạo dãy bit sum để tính tổng `str1` và `str2`.

_ Sau đó cộng lần lượt từng bit của dãy bit `str1` và `str2` với nhau, tuân thủ quy luật: $1 + 0 = 1$ hoặc $0 + 1 = 1$, $1 + 1 = 0$ nhớ 1, tạo 1 biến nhớ để ghi lại trường hợp $1 + 1$.

_ Cuối cùng, chuyển dãy bit sum vào 1 số `QInt` và xuất ra kết quả.

- **Phép trừ(-):**

_ Gọi hàm `operator-(const QInt& q1, const QInt& q2)`

_ Đầu tiên, gọi hàm `operator-(const QInt& q)` để chuyển số `q2` thành bù 2 của nó.

_ Sau đó, gọi hàm `operator+(const QInt& q1, const QInt& q2)` cho `q1` và bù 2 của `q2`.

- **Phép nhân(*):**

_ Gọi hàm `operator*(const QInt& q1, const QInt& q2)`

_ Tạo ra 1 số `QInt` `result` để lưu kết quả, lưu số `QInt` `q2` dưới dạng dãy bit nhị phân `str` bằng cách gọi hàm `toBinary()`.

_ Với mỗi bit trong tất cả 128 bit của `str`, nếu là bit 1 thì cộng kết quả `result` với `q1`, sau đó dịch trái `q1` 1 bit, nếu là bit 0 thì chỉ dịch trái `q1` 1 bit.

_ Cuối cùng trả về `QInt` `result` là kết quả của phép nhân.

- **Phép chia(/):**
 - _ Gọi hàm `operator/(const QInt& q1, const QInt& q2)`
 - _ Lần lượt lấy dãy bit nhị phân của q1 và q2 lưu vào string1 và string2 bằng cách gọi hàm `toBinary()`.

Phép chia

- Giả sử ta muốn thực hiện Q / M với


```

Khởi tạo: A = n bit 0 nếu Q > 0; A = n bit 1 nếu Q < 0; k = n
Lặp khi k > 0
{
    Shift left (SHL) [A, Q]

    A - M → A
    # Nếu A < 0: Q0 = 0 và A + M → A
    # Ngược lại: Q0 = 1

    k = k - 1
}
      
```

Kết quả: Q là thương, A là số dư

41

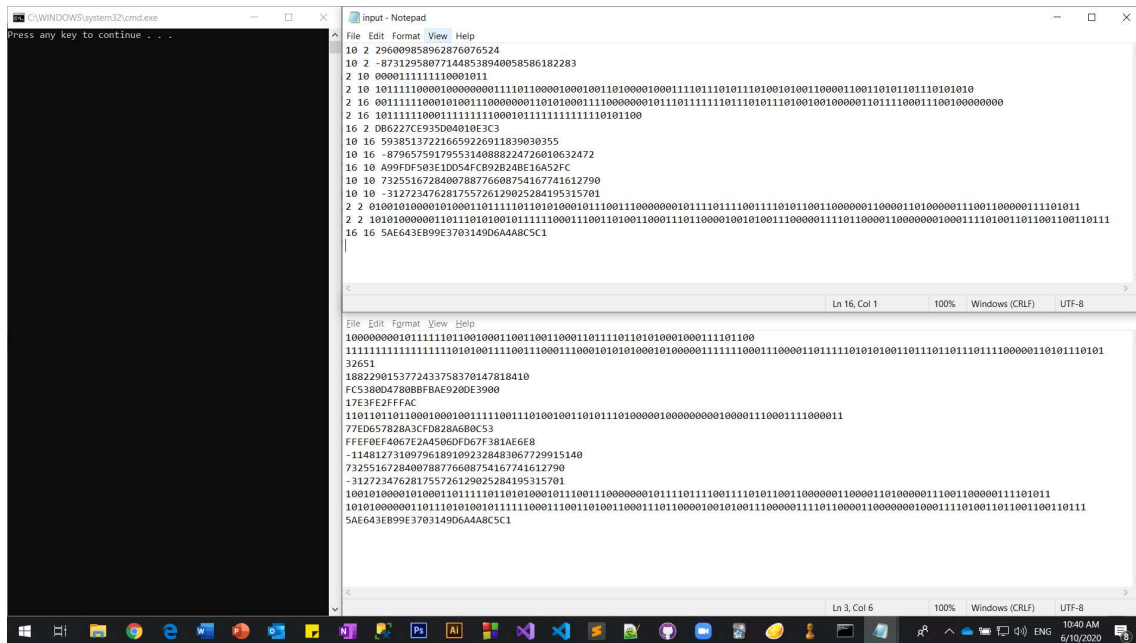
- _ Thực hiện thuật toán chia như trên với $Q = \text{string1}$, $M = \text{string2}$ và $k = 128$.
- **Toán tử AND &:**
 - _ Gọi hàm `operator&(const QInt& q1, const QInt& q2)`
 - _ Lần lượt lấy dãy bit nhị phân của q1 và q2 lưu vào string1 và string2 bằng cách gọi hàm `toBinary()`.
 - _ Khởi tạo dãy bit str toàn bit 0 để lưu kết quả. Với mỗi bit của bit kết quả, nếu bit của string1 và string2 đều bằng 1 thì gán 1 vào bit của biến kết quả đó, ngược lại gán 0.
 - _ Cuối cùng ta lưu dãy bit nhị phân str vào số QInt và return kết quả.
- **Toán tử OR |:**
 - _ Gọi hàm `operator|(const QInt& q1, const QInt& q2)`
 - _ Lần lượt lấy dãy bit nhị phân của q1 và q2 lưu vào string1 và string2 bằng cách gọi hàm `toBinary()`.

- _ Khởi tạo dãy bit str toàn bit 0 để lưu kết quả. Với mỗi bit của bit kết quả, nếu bit của string1 và string2 đều bằng 0 thì gán 0 vào bit của biến kết quả đó, ngược lại gán 1.
- _ Cuối cùng ta lưu dãy bit nhị phân str vào số QInt và return kết quả.
- **Toán tử XOR ^:**
 - _ Gọi hàm `operator^(const QInt& q1, const QInt& q2)`
 - _ Lần lượt lấy dãy bit nhị phân của q1 và q2 lưu vào string1 và string2 bằng cách gọi hàm `toBinary()`.
 - _ Khởi tạo dãy bit str toàn bit 0 để lưu kết quả. Với mỗi bit của bit kết quả, nếu bit của string1 bằng 1 và string2 bằng 0 hoặc bit của string1 bằng 0 và string2 bằng 1 thì gán 1 vào bit của biến kết quả đó, ngược lại gán 0.
 - _ Cuối cùng ta lưu dãy bit nhị phân str vào số QInt và return kết quả.
- **Toán tử NOT ~:**
 - Gọi hàm `operator~()`
 - _ Khởi tạo dãy bit nhị phân str bằng cách gọi hàm `toBinary()` chuyển số QInt đang gọi hàm này về dạng dãy bit nhị phân.
 - _ Với mỗi bit trong dãy bit nhị phân str đó, nếu là bit 1 thì gán lại bằng bit 0 và ngược lại.
 - _ Cuối cùng ta lưu dãy bit nhị phân vào số QInt và gán số QInt này vào số QInt thực hiện gọi hàm này.
- **Toán tử dịch trái <<:**
 - Gọi hàm `operator<<(const int& q)`
 - _ Khởi tạo dãy bit nhị phân str bằng cách gọi hàm `toBinary()` chuyển số QInt đang gọi hàm này về dạng dãy bit nhị phân.
 - _ Với q lần dịch bit, mỗi lần dịch ta dịch chuyển chuỗi str sang trái 1 đơn vị và thêm bit 0 vào bên phải cùng.
 - _ Cuối cùng ta lưu dãy bit nhị phân str vào số QInt và return kết quả.
- **Toán tử dịch phải >>:**
 - Gọi hàm `operator>>(const int& q)`
 - _ Khởi tạo dãy bit nhị phân str bằng cách gọi hàm `toBinary()` chuyển số QInt đang gọi hàm này về dạng dãy bit nhị phân, tạo 1 biến sign để lưu bit dấu.
 - _ Với q lần dịch bit, mỗi lần dịch ta dịch chuyển chuỗi str sang phải 1 đơn vị và thêm bit dấu sign vào bên trái cùng.
 - _ Cuối cùng ta lưu dãy bit nhị phân str vào số QInt và return kết quả.
- **Phép xoay trái ROL:**
 - _ Khởi tạo dãy bit nhị phân binary bằng cách gọi hàm `toBinary()` chuyển số QInt đang gọi hàm này về dạng dãy bit nhị phân, tạo 1 biến temp để lưu giá trị của bit trái cùng.
 - _ Dịch trái dãy bit nhị phân binary 1 đơn vị, sau đó gán bit phải cùng của binary bằng bit temp.

- _ Cuối cùng ta lưu dãy bit nhị phân binary vào số QInt và return kết quả.
- **Phép xoay phải ROR:**
 - _ Khởi tạo dãy bit nhị phân binary bằng cách gọi hàm `toBinary()` chuyển số QInt đang gọi hàm này về dạng dãy bit nhị phân, tạo 1 biến temp để lưu giá trị của bit phải cùng.
 - _ Dịch phải dãy bit nhị phân binary 1 đơn vị, sau đó gán bit trái cùng của binary bằng bit temp.
 - _ Cuối cùng ta lưu dãy bit nhị phân binary vào số QInt và return kết quả.

II. PHÂN TÍCH CÁC CHỨC NĂNG:

1. Chuyển đổi giữa các cơ số:



The screenshot shows a Windows desktop with a command prompt window titled 'C:\WINDOWS\system32\cmd.exe' and a Notepad window titled 'Input - Notepad'. The command prompt is displaying a series of binary and hexadecimal data. The Notepad window is also displaying a series of binary and hexadecimal data, which appears to be a continuation or a different view of the same data shown in the command prompt. The data consists of long strings of 0s and 1s, followed by hexadecimal representations of these strings.

Dòng:

1. Chuyển số dương hệ 10 sang hệ 2.
2. Chuyển số âm hệ 10 sang hệ 2.
3. Chuyển số dương hệ 2 sang hệ 10.
4. Chuyển số âm hệ 2 sang hệ 10.
5. Chuyển số dương hệ 2 sang hệ 16.
6. Chuyển số âm hệ 2 sang hệ 16.
7. Chuyển số từ hệ 16 sang hệ 2.
8. Chuyển số dương hệ 10 sang hệ 16.
9. Chuyển số âm hệ 10 sang hệ 16.
10. Chuyển số tự hệ 16 sang hệ 10.
11. Chuyển từ số dương hệ 10 sang hệ 10.
12. Chuyển từ số âm hệ 10 sang hệ 10.
13. Chuyển từ số dương hệ 2 sang hệ 2.
14. Chuyển từ số âm hệ 2 sang hệ 2.
15. Chuyển từ số hệ 16 sang hệ 16.

2. Các phép toán trên QInt:

a. Toán tử AND(&), OR(|), XOR(^), NOT(~), SHIL(<<),SHIR(>>):

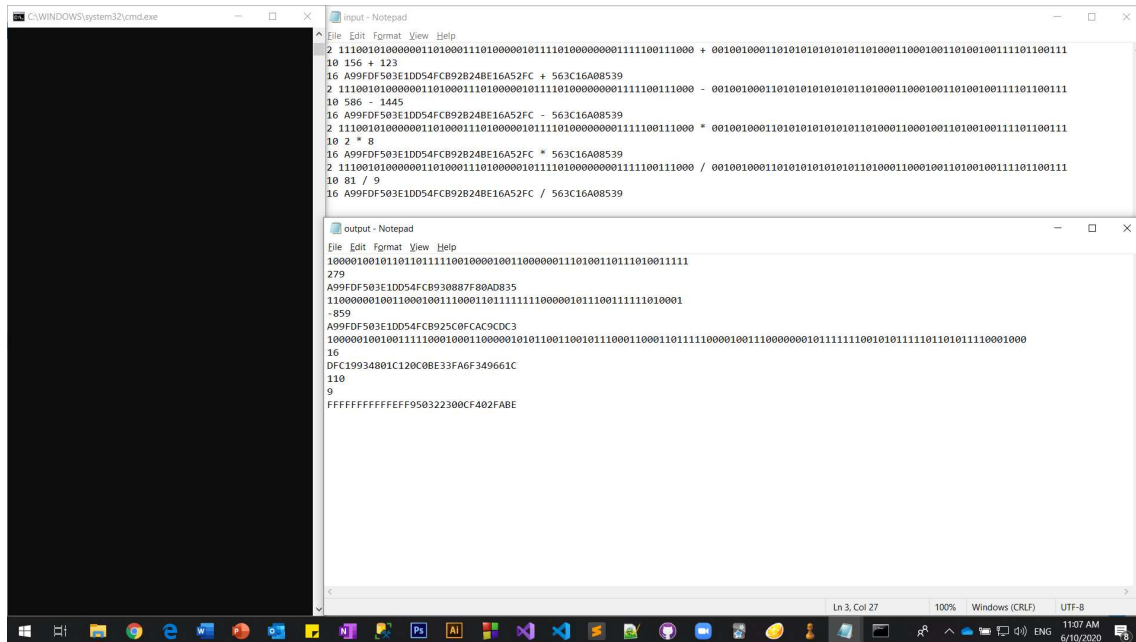
```
input - Notepad
File Edit Format View Help
2 1001010101 & 100101011
10 1945651 & 564981313
16 9C448 & 97175C77FBE08B64B36D512D6F2
2 1001010101 | 100101011
10 1945651 | 564981313
16 9C448 | 97175C77FBE08B64B36D512D6F2
2 1001010101 ^ 100101011
10 1945651 ^ 564981313
16 519 ^ 83C088536
2 ~ 100011001111011010000111100101000001101001010010110111011100101111110010010100001110101100010011110
10 ~ 488108642434
16 ~ 6673417108604A17BCF0F7427D6A12F
2 0100100011011001111011110001001011010010000101110010000101111010001110001001011111000101111010111101111000100 << 9
10 1505464646546513213299 << 23
16 0127F4D6EC0F7A4601 << 41
2 01001000110110011110111100010010110100100001011000100101111010001110001001011111000101111 >> 16
10 ~6529624872154135214755416638567 >> 67
16 97175C77FBE08B64B36D512D6F2 >> 73

output - Notepad
File Edit Format View Help
1
827393
C440
1101111111
566099571
97175C77FBE08B64B36D512D6FA
1101111110
565272178
83C08809F
1111111111011100011000100101110000011010111100101101011010010010001010000000110110010110100111100010011101100001
-488108642435
F998C8E8F79F85B438F08080295ED0
1101100111011111000100101101001000101110010000101110100011100010010111110001011101011110111100010000000000
1313206925773725313185697792
24FE9ADD81FF48C020000000000
10010001101100111011111000100101101001000010111000100101111010001110001001
-44246459200
4B8BAE3BF
```

Dòng:

1. Phép AND(&) trên hệ 2.
2. Phép AND(&) trên hệ 10.
3. Phép AND(&) trên hệ 16.
4. Phép OR(|) trên hệ 2.
5. Phép OR(|) trên hệ 10.
6. Phép OR(|) trên hệ 16.
7. Phép XOR(^) trên hệ 2.
8. Phép XOR(^) trên hệ 10.
9. Phép XOR(^) trên hệ 16.
10. Phép NOT(~) trên hệ 2.
11. Phép NOT(~) trên hệ 10.
12. Phép NOT(~) trên hệ 16.
13. Phép SHIL(<<) 9 bit trên hệ 2.
14. Phép SHIL(<<) 23 bit trên hệ 10.
15. Phép SHIL(<<) 41 bit trên hệ 16.
16. Phép SHIR(>>) 16 bit trên hệ 2.
17. Phép SHIR(>>) 67 bit trên hệ 10.
18. Phép SHIR(>>) 73 bit trên hệ 16.

b. Toán tử CỘNG(+), TRỪ(-), NHÂN(*), CHIA(/):



```
input - Notepad
File Edit Format View Help
2 111001010000001101000111010000000101110100111000 + 00100100011010101010101010000100010011010011101100111
10 156 + 123
16 A99FDF503E1DD54FCB92B24BE16A52FC + 563C16A08539
2 111001010000001101000111010000000101110100111000 - 00100100011010101010101010001100010011010011101100111
10 586 - 1445
16 A99FDF503E1DD54FCB92B24BE16A52FC - 563C16A08539
2 111001010000001101000111010000000101110100111000 * 00100100011010101010101010001100010011010011101100111
10 2 * 8
16 A99FDF503E1DD54FCB92B24BE16A52FC * 563C16A08539
2 111001010000001101000111010000000101110100111000 / 001001000110101010101010001100010011010011101100111
10 81 / 9
16 A99FDF503E1DD54FCB92B24BE16A52FC / 563C16A08539

output - Notepad
File Edit Format View Help
1000010010110110111100100001001100000011010011011010011111
279
A99FDF503E1DD54FCB930887F80AD835
110000001001100010011100010111111100000101110011111010001
-859
A99FDF503E1DD54FCB925C0FCAC9C0DC3
100000100100111100010001100000101011001100111100010001101111100001001111110010101111101011110001000
16
DFC19934801C120C0BE33FA6F349661C
110
9
FFFFFFFFFFFF950322300CF402FABE
```

Dòng:

1. Cộng 2 dãy bit nhị phân.
2. Cộng 2 số thập phân.
3. Cộng 2 dãy bit thập lục phân.
4. Trừ 2 dãy bit nhị phân.
5. Trừ 2 số thập phân.
6. Trừ 2 dãy bit thập lục phân.
7. Nhân 2 dãy bit nhị phân.
8. Nhân 2 số thập phân.
9. Nhân 2 dãy bit thập lục phân.
10. Chia 1 dãy bit nhị phân cho 1 dãy bit nhị phân.
11. Chia 1 số thập phân cho 1 số thập phân.
12. Chia 1 dãy bit thập lục phân cho 1 dãy bit thập lục phân.

c. Phép XOAY TRÁI(rol), XOAY PHẢI(ror):

```
C:\WINDOWS\system32\cmd.exe
Press any key to continue . . .

input - Notepad
File Edit Format View Help
2 rol 000110000100110011100110100001100010110001110000001001101010100100111001010111110111101011100110000000001110
10 rol -51020011559739511523931749439
16 rol 168371020318101142321F7788796
2 ror 11100101000000110100011101000001011110100000001111100111000
10 ror -57
16 ror 1C7D

output - Notepad
File Edit Format View Help
1100001001100111001101000011000101100011100000010011010101001001110010101111101111010111001100000000011100
-102041623119479023047803498877
2D66E205A6362022846A3EEF70F2C
1110010100000011010001110100000101111010000000111110011100
-29
80000000000000000000000000000000E3E
```

Dòng:

1. Xoay trái dãy bit nhị phân.
2. Xoay trái số thập phân.
3. Xoay trái dãy bit thập lục phân
4. Xoay phải dãy bit nhị phân.
5. Xoay phải số thập phân.
6. Xoay phải dãy bit thập lục phân.

III. Mức độ hoàn thành đồ án:

Hoàn thành	Chưa hoàn thành
1. Chuyển đổi số QInt từ hệ 10 sang 2 và ngược lại. 2. Chuyển đổi số QInt từ hệ 2 sang 16 và ngược lại. 3. Chuyển đổi số QInt từ hệ 16 sang 10 và ngược lại. 4. Các toán tử + - * /. 5. Các toán tử & ^ ~ << >>. 6. Các phép rol và ror.	Không có.

⇒ Hoàn thành 100% đồ án.

IV. Tài liệu tham khảo:

https://stackoverflow.com/questions/13166785/how-to-convert-binary-string-to-integer-string?fbclid=IwAR2O01posFWKJ_D5Q2czwemwciN_3xwzlhPhjF-gsPDjyhMUQtbcaUeR86c

Slide bài giảng “Biểu diễn số nguyên” của thầy Lê Quốc Hòa.