



**ĐẠI HỌC QUỐC GIA TP.HCM**

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

## KIẾN TRÚC MÁY TÍNH VÀ HỢP NGỮ

## BÁO CÁO ĐỒ ÁN I

Giáo viên hướng dẫn: **Thái Hùng Văn**

**Thành viên nhóm**

NGUYỄN CHÍ THUẬN – 20127346

LƯƠNG VŨ THÁI - 20127325

**Mục lục**

[KIẾN TRÚC MÁY TÍNH VÀ HỢP NGỮ 1](#_Toc87209918)

[BÁO CÁO ĐỒ ÁN I 1](#_Toc87209919)

[**MÔ TẢ** 3](#_Toc87209920)

[**1)** **Thông tin thành viên và mức độ đóng góp** 4](#_Toc87209921)

[**2)** **Đánh giá mức độ hoàn thành(%) ứng với từng yêu cầu và toàn bộ project:** 4](#_Toc87209922)

[**3)** **Cấu trúc chương trình** 5](#_Toc87209923)

[**4)** **Nguồn tham khảo (phục vụ cho mục đích học tập)** 18](#_Toc87209924)

**MÔ TẢ**

Báo cáo thể hiện kết quả của quá trình tìm hiểu, cài đặt cách thức nhập dữ liệu, chuyển hệ, các phép toán tử để tính toán trên các kiểu dữ liệu thập phân hay nhị phân có độ lớn 8, 16, 32 hay 64 byte.

Tất cả quá trình hoàn thiện project trên, từ source code, phân công công việc, tự đánh giá mức độ hoàn thành,…

1. **Thông tin thành viên và mức độ đóng góp**

|  |  |  |
| --- | --- | --- |
| **STT** | **Họ và tên** | **Mã số sinh viên** |
| **1** | Nguyễn Chí Thuận | 20127346 |
| **2** | Lương Vũ Thái | 20127325 |

**Mức độ đóng góp đối với project:**

**Danh sách công việc chính nhóm đã thực hiện:**

* Tìm hiểu, nghiên cứu, tổ chức lưu trữ, cài đặt nhập xuất, chuyển hệ từ thập phân sang nhị phân dưới dạng có dấu, không dấu chấm động 32 bit và 64 bit: Lương Vũ Thái
* Tìm hiểu, nghiên cứu, tổ chức, lưu trữ, cài đặt nhập xuất, chuyển hệ nhị phân sang thập phân dưới dạng có dấu, không dấu chấm động 32 bit và 64 bit: Nguyễn Chí Thuận
* Tổng hợp source code và nghiên cứu, cài đặt hàm main: Nguyễn Chí Thuận
* Viết báo cáo: cả nhóm.

1. **Đánh giá mức độ hoàn thành(%) ứng với từng yêu cầu và toàn bộ project:**
2. **Thập phân sang nhị phân**

|  |  |
| --- | --- |
| **Các yêu cầu** | **Mức độ hoàn thành** |
| Nhập | **100%** |
| Xuất | **100%** |
| Chuyển đổi số từ thập phân sang nhị phân có dấu | **100%** |
| Chuyển đổi số từ thập phân sang nhị phân không dấu | **100%** |
| Chuyển đổi số từ thập phân sang nhị phân chấm động 32 bit | **100%** |
| Chuyển đổi số từ thập phân sang nhị phân chấm động 64 bit | **100%** |

1. **Nhị phân sang thập phân**

|  |  |
| --- | --- |
| **Các yêu cầu** | **Mức độ hoàn thành** |
| Nhập | **100%** |
| Xuất | **100%** |
| Chuyển đổi số từ nhị phân sang thập phân có dấu | **100%** |
| Chuyển đổi số từ nhị phân sang thập phân không dấu dấu | **100%** |
| Chuyển đổi số từ nhị phân sang thập phân chấm động 32 bit | **100%** |
| Chuyển đổi số từ nhị phân sang thập phân chấm động 64 bit | **99%** |

1. **Đánh giá mức độ hoàn thành đối với toàn bộ Project:** **99%**
2. **Cấu trúc chương trình**

**a) Các hàm trong chương trình đổi từ thập phân sang nhị phân**

string intToBinary(int n)

Hàm chuyển đổi từ hệ thập phân sang chuỗi nhị phân để xuất ra màn hình nhận vào đối số là 1 số nguyên

string dec\_to\_bin(float number)

Hàm chuyển đổi thập phân sau dấu “.” sang chuỗi nhị phân để xuất ra màn hình nhận đối số là một số thập phân

string to\_IEEE75\_32bit(float number)

Hàm chuyển đổi một số thập phân sang chấm động 32 bit nhận đối số là một số thập phân các bước thực hiện lần lượt là:

* Bước 1: Lấy sign ở bit đầu tiên bằng cách so sánh số đó có âm không
* Bước 2: Lần lượt lấy các phần khác như exponent, mantisa bằng hàm string intToBinary(int n) để nhận được chuỗi nhị phân của exponent, mantisa
* Bước 3: Cộng các chuối string sign + exponent + mantisa ta sẽ nhận được chuỗi string binary chấm động 32 bit

string to\_IEEE75\_64bit(float number)

Hàm chuyển đổi một số thập phân sang chấm động 64 bit nhận đối số là một số thập phân các bước thực hiện lần lượt là:

* Bước 1: Lấy sign ở bit đầu tiên bằng cách so sánh số đó có âm không
* Bước 2: Lần lượt lấy các phần khác như exponent, mantisa bằng hàm string intToBinary(int n) để nhận được chuỗi nhị phân của exponent, mantisa
* Bước 3: Cộng các chuối string sign + exponent + mantisa ta sẽ nhận được chuỗi string binary chấm động 64 bit

string unsign\_interger(float num)

Hàm chuyển đổi thập phân sang chuỗi nhị phân nhận đối số là một số thập phân

* Bước 1: Nếu số này là 1 số thập phân thì sẽ làm tròn số đến số hàng đơn vị
* Bước 2: Yêu cầu người dùng chọn bit muốn biểu diễn (8 bit,16 bit, 32 bit), nếu nhập không đúng sẽ yêu cầu nhập lại
* Bước 3: Nếu số biểu diễn có số bit lớn hơn số được chọn sẽ chọn số bit bên trái nhất để biểu diễn

string sign\_interger(float num)

Hàm chuyển đổi thập phân sang chuỗi nhị phân nhận đối số là một số thập phân

* Bước 1: Nếu số này là 1 số thập phân lớn hơn 0 thì sẽ làm tròn số đến số hàng đơn vị còn nếu nhỏ hơn 0 sẽ làm tròn ngược lại
* Bước 2: Yêu cầu người dùng chọn bit muốn biểu diễn (8 bit,16 bit, 32 bit), nếu nhập không đúng sẽ yêu cầu nhập lại
* Bước 3: Nếu là số dương sẽ biểu diễn như số không dấu nếu số âm thì sẽ thêm bit 0 vào đầu của chuối nhị phân

**b) Các hàm trong chương trình đổi từ thập phân sang nhị phân**

long long unsignedBinToDec(string binaryNumber)

Hàm chuyển từ chuỗi nhị phân sang thập phân không dấu với đối số nhận vào là một chuỗi nhị phân được người dùng nhập vào

* Bước 1: Biểu diễn sang số nguyên không dấu
* Bước 2: Sau đó lần lượt lấy từ bit đầu đến bit cuối nhân với 2 mũ phù hợp với công thức và trả về số thập phân

bool checksize(int size)

Kiểm tra xem số bit của người dùng có đúng với 8 bit, 16 bit, 32 bit, hay 64 bit không nếu khôg đúng sẽ yêu cầu nhập lại

string handleSpaceInputString(string binary)

Hàm chuẩn hóa số liệu nhận vào chuỗi string kể cả dấu cách để người dùng thuận lợi hơn sẽ trả về một chuỗi nhị phân chỉ còn 1,0 và không còn bất cứa dấu cách nào

string handlingMissingInputString(string binary,int size)

Hàm chuẩn hóa số liệu sẽ bổ sung thêm vào các số không nếu như người dùng chưa nhập đủ số bit mà người dùng đã chọn

string bitIsland(string binary)

Hàm sẽ thay đổi từ bit 0 sang bit 1 và bit 1 sẽ thành bit không thuận lợi cho việc chuyển sang số bù 2

long double BitFloatingPoint32(string binary)

Hàm sẽ chuyển từ chuỗi nhị phân 32 bit sang số thập phân nhận đối số là 1 chuỗi nhị phân do người dùng nhập vào

Bước 1: Bổ sung bit 0 vào đầu nếu người dùng nhập chưa đủ

Bước 2: Lấy sign là bit đầu của chuỗi sau khi chuẩn hóa (nếu có) tiếp đến là 8 bit Exponent và 23 bit Mantissa

Bước 3: Tính theo công thức và trả ra chuỗi thập phân chấm đông 32 bit

long double BitFloatingPoint64(string binary)

Hàm sẽ chuyển từ chuỗi nhị phân 64 bit sang số thập phân nhận đối số là 1 chuỗi nhị phân do người dùng nhập vào

Bước 1: Bổ sung bit 0 vào đầu nếu người dùng nhập chưa đủ

Bước 2: Lấy sign là bit đầu của chuỗi sau khi chuẩn hóa (nếu có) tiếp đến là 11 bit Exponent và 52 bit Mantissa

Bước 3: Tính theo công thức và trả ra chuỗi thập phân chấm động 64 bit

1. **Phần chưa thực hiện được của chương trình**

* Không thể tính số chấm động 64 với phần exponent lớn hơn 11111111101

1. **Source code chương trình**

string intToBinary(int n)

{

string r;

while (n != 0) { r = (n % 2 == 0 ? "0" : "1") + r; n /= 2; }

return r;

}

string dec\_to\_bin(float number)

{

string bin = "";

while (number > 0)

{

number = number \* 2;

if (number >= 1)

{

bin = bin + "1";

number--;

}

else {

bin = bin + "0";

}

}

return bin;

}

string to\_IEEE75\_32bit(float number)

{

// get the sign bit

string sign\_bit;

if (number > 0)

{

sign\_bit = "0";

}

else {

sign\_bit = "1";

}

int num = abs(number);

string exponent;

exponent = intToBinary((intToBinary(num)).size() + 127 - 1);

while (exponent.size() < 8)

{

exponent = "0" + exponent;

}

string decimal\_part = dec\_to\_bin(abs(number) - num);

string mantisa;

string pos = intToBinary(num) + decimal\_part;

mantisa = string(pos.begin() + 1, pos.end());

while (mantisa.size() < 23)

{

mantisa = mantisa + "0";

}

string iEE75 = sign\_bit + " " + exponent + " " + mantisa;

return iEE75;

}

string to\_IEEE75\_64bit(float number)

{

// get the sign bit

string sign\_bit;

if (number > 0)

{

sign\_bit = "0";

}

else {

sign\_bit = "1";

}

int num = abs(number);

string exponent;

exponent = intToBinary((intToBinary(num)).size() + 1023 - 1);

while (exponent.size() < 11)

{

exponent = "0" + exponent;

}

string decimal\_part = dec\_to\_bin(abs(number) - num);

string mantisa;

string pos = intToBinary(num) + decimal\_part;

mantisa = string(pos.begin() + 1, pos.end());

while (mantisa.size() < 52)

{

mantisa = mantisa + "0";

}

string iEE75 = sign\_bit + " " + exponent + " " + mantisa;

return iEE75;

}

string unsign\_interger(float num)

{

int a;

if (num - int(num) >= 0.5)

{

a = int(num) + 1;

}

else {

a = int(num);

}

cout << "chon so bit bieu dien" << endl;

cout << "1: 8bit 2: 16 bit 3:32bit" << endl;

int choice;

do {

cin >> choice;

if (choice > 3)

{

cout << " so khong hop le. Vui long nhap lai ";

}

} while (choice > 3);

if (choice == 1)

{

string result = intToBinary(a);

while (result.size() < 8)

{

result = "0" + result;

}

if (result.size() > 8)

{

result = result.substr(0, 8);

}

return result;

}

if (choice == 2)

{

string result = intToBinary(a);

while (result.size() < 16)

{

result = "0" + result;

}

if (result.size() > 16)

{

result = result.substr(0, 16);

}

return result;

}

if (choice == 3)

{

string result = intToBinary(a);

while (result.size() < 32)

{

result = "0" + result;

}

if (result.size() > 32)

{

result = result.substr(0, 32);

}

return result;

}

}

string sign\_interger(float num)

{

string positive = "0";

string negative = "1";

string result;

if (num > 0)

{

int a;

if (num - int(num) >= 0.5)

{

a = int(num) + 1;

}

else {

a = int(num);

}

cout << "chon so bit bieu dien" << endl;

cout << "1: 8bit 2: 16 bit 3:32bit" << endl;

int choice;

do {

cin >> choice;

if (choice > 3)

{

cout << " so khong hop le. Vui long nhap lai ";

}

} while (choice > 3);

if (choice == 1)

{

result = intToBinary(a);

while (result.size() < 7)

{

result = "0" + result;

}

result = positive + result;

if (result.size() > 8)

{

result = result.substr(0, 8);

}

cout << "Signed: ";

return result;

}

if (choice == 2)

{

result = intToBinary(a);

while (result.size() < 15)

{

result = "0" + result;

}

result = positive + result;

if (result.size() > 16)

{

result = result.substr(0, 16);

}

cout << "Signed: ";

return result;

}

if (choice == 3)

{

result = intToBinary(a);

while (result.size() < 31)

{

result = "0" + result;

}

result = positive + result;

if (result.size() > 32)

{

result = result.substr(0, 32);

}

}

cout << "Signed: ";

return result;

}

else if (num < 0)

{

int b;

if (num - int(num) <= -0.5)

{

b = -int(num) + 1;

}

else {

b = -int(num);

}

cout << "chon so bit bieu dien am" << endl;

cout << "1:8bit 2:16 bit 3:32bit" << endl;

int choice;

do {

cin >> choice;

if (choice > 3)

{

cout << " so khong hop le. Vui long nhap lai ";

}

} while (choice > 3);

int num = abs(b);

if (choice == 1)

{

result = intToBinary(num);

while (result.size() < 7)

{

result = "0" + result;

}

for (int i = 0; i < result.size(); i++)

{

if (result[i] == '0')

{

result[i] = '1';

}

else {

result[i] = '0';

}

}

for (int i = result.size() - 1; i >= 0; i--)

{

if (result[i] == '0')

{

result[i] = '1';

break;

}

else if (result[i] == '1')

{

result[i] = '0';

}

}

result = result.substr(0, 7);

result = negative + result;

cout << "Signed: ";

return result;

}

if (choice == 2)

{

result = intToBinary(num);

while (result.size() < 15)

{

result = "0" + result;

}

for (int i = 0; i < result.size(); i++)

{

if (result[i] == '0')

{

result[i] = '1';

}

else {

result[i] = '0';

}

}

for (int i = result.size() - 1; i >= 0; i--)

{

if (result[i] == '0')

{

result[i] = '1';

break;

}

else if (result[i] == '1')

{

result[i] = '0';

}

}

result = result.substr(0, 15);

result = negative + result;

cout << "Signed: ";

return result;

}

if (choice == 3)

{

result = intToBinary(num);

while (result.size() < 31)

{

result = "0" + result;

}

for (int i = 0; i < result.size(); i++)

{

if (result[i] == '0')

{

result[i] = '1';

}

else {

result[i] = '0';

}

}

for (int i = result.size() - 1; i >= 0; i--)

{

if (result[i] == '0')

{

result[i] = '1';

break;

}

else if (result[i] == '1')

{

result[i] = '0';

}

}

result = result.substr(0, 31);

result = negative + result;

cout << "Signed: ";

return result;

}

}

}

long long unsignedBinToDec(string binaryNumber)

{

long long decimal = 0;

if (binaryNumber.size() > 32) {

for (int i = 0; i < binaryNumber.size()-32; i = i + 1) {

string temp;

temp = temp + binaryNumber[i];

decimal = decimal + stoll(temp) \* pow(2, binaryNumber.size() - i);

}

}

else {

for (int i = 0; i < binaryNumber.size(); i = i + 1) {

string temp;

temp = temp + binaryNumber[i];

decimal = decimal + stoll(temp) \* pow(2, binaryNumber.size() - i);

}

}

return decimal;

}

bool checksize(int size) {

if (size != 8 && size != 16 && size != 32 && size != 64)

{

return false;

}

else {

return true;

}

}

string handleSpaceInputString(string binary) {

while (binary.find(' ') != -1) {

binary.replace(binary.find(' '), 1, "");

}

return binary;

}

string handlingMissingInputString(string binary,int size) {

string str;

if (binary.size() < 16 && size != 8) {

for (int i = 0; i < 16 - binary.size(); i = i + 1) {

str = str + "0";

}

}

str = str + binary;

return str;

}

string bitIsland(string binary) {

string temp;

for (int i = 1; i < binary.size(); i = i + 1) {

temp = temp + binary[i];

}

for (int i = 0; i < temp.size(); i = i + 1) {

if (temp[i] == '1') {

temp[i] = '0';

}

else {

temp[i] = '1';

}

}

return temp;

}

long double BitFloatingPoint32(string binary) {

string temp;

string Sign;

string Exponent;

string Mantissa;

if (binary.size() < 32) {

for (int i = 0; i < 32 - binary.size(); i = i + 1) {

temp = temp + '0';

}

}

temp = temp + binary;

Sign = temp[0];

long double sum = 0;

for (int i = 1; i <= 8; i = i + 1) {

Exponent = Exponent + temp[i];

}

for (int i = 9; i <= 31; i = i + 1) {

Mantissa = Mantissa + temp[i];

}

float a = 1;

long long f = pow(-1, stoi(Sign));

int x = unsignedBinToDec(Exponent) - 127;

long double m ;

if (x > 0) {

m = pow(2, x);

}

else {

m = 1 / pow(2, abs(x));

}

for (int i = 0; i < 23; i = i + 1) {

string h = "";

h = h + Mantissa[i];

a = a + stof(h) \* 1.0 \* pow(2, -i - 1);

}

sum = f \* m \* a;

return sum;

}

long double BitFloatingPoint64(string binary) {

string temp;

string Sign;

string Exponent;

string Mantissa;

if (binary.size() < 64) {

for (int i = 0; i < 64 - binary.size(); i = i + 1) {

temp = temp + '0';

}

}

temp = temp + binary;

Sign = temp[0];

long double sum = 0;

for (int i = 1; i <= 11; i = i + 1) {

Exponent = Exponent + temp[i];

}

for (int i = 12; i <= 63; i = i + 1) {

Mantissa = Mantissa + temp[i];

}

double a = 1;

long long f = pow(-1, stoi(Sign));

long double x = unsignedBinToDec(Exponent) - 1023;

long double m=0;

if (x > 0) {

m = powl(2, x);

}

else {

m = 1 / powl(2, abs(x));

}

for (int i = 0; i < 52; i = i + 1) {

string h = "";

h = h + Mantissa[i];

a = a + stof(h) \* 1.0 \* pow(2, -i - 1);

}

sum = f \* m \* a;

return sum;

}

int main()

{

int choose;

while (true) {

cout << "1.Ban muon doi binary sang demacial" << endl;

cout << "2.Ban muon doi demacial sang binary" << endl;

cout << "3.De thoat khoi chuong trinh" << endl;

cout << "Nhap lua chon: ";

cin >> choose;

if (choose == 1) {

double value;

cout << "Ban hay nhap so thap phan muon doi" << endl;

cin >> value;

cout << "Floating point 32 bit: ";

cout << to\_IEEE75\_32bit(value);

cout << endl;

cout << "Floating point 64 bit: ";

cout << to\_IEEE75\_64bit(value);

cout << endl;

cout << sign\_interger(value)<<endl;

cout << endl;

cout << endl;

}

else if (choose == 2) {

int size = 0;

cout << "Ban muon nhap bao nhieu byte(8,16,32 hay 64): ";

cin >> size;

while (true) {

if (checksize(size) == false)

{

cout << "Ban hay nhap lai sai size: ";

cin >> size;

}

else {

break;

}

}

cout << "Nhap chuoi nhi phan (co the nhap dau cach vd(0101 000)): ";

string n;

getline(cin, n);

getline(cin, n);

n = handleSpaceInputString(n);

string str;

str = handlingMissingInputString(n, size);

regex regex("\\b[0-1]+\\b");

long long a = 0;

bool result = regex\_match(n, regex);

while (true) {

if (result == false) {

cout << "Nhap lai chuoi nhi phan: ";

cin >> n;

result = regex\_match(n, regex);

}

else {

break;

}

}

if (result == true) {

a = unsignedBinToDec(str);

cout << "Unsigned: " << a << endl;

}

if (result == true && str[0] == '1') {

string temp = bitIsland(str);

cout << "signed: " << "-" << unsignedBinToDec(temp) + 1 << endl;

}

else {

cout << "signed: " << a << endl;

}

if (size == 32) {

cout << "Floating point 32 bit: " << BitFloatingPoint32(n)<<endl;

cout << endl;

cout << endl;

}

else if (size == 64) {

cout << "Floating point 64 bit: " << BitFloatingPoint64(n)<<endl;

cout << endl;

cout << endl;

}

}

else if (choose == 3) {

break;

}

else {

cout << "Nhap lai lua chon: ";

cin >> choose;

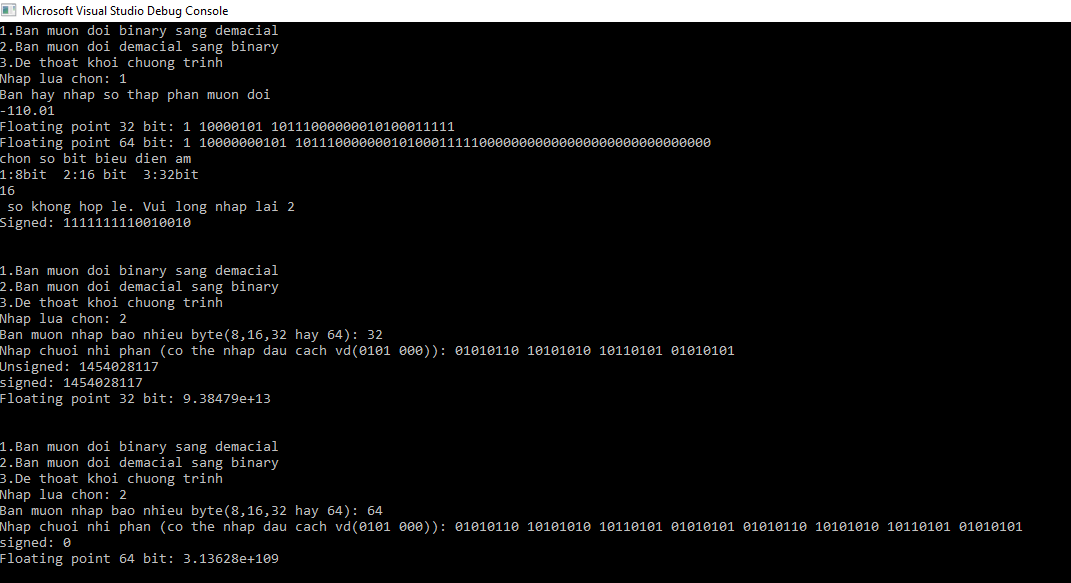
}

}

cin.get();

}

1. **Demo chương trình**

****

1. **Nguồn tham khảo (phục vụ cho mục đích học tập)**

* [**stackoverflow.com**](https://stackoverflow.com/)
* [**github.com**](https://github.com/)
* [**How to multiplying ieee-754 floating point numbers**](https://www.youtube.com/results?search_query=multiplying+ieee-754+floating+point+numbers)
* [**Quadruple-precision floating-point format**](https://en.wikipedia.org/wiki/Quadruple-precision_floating-point_format)
* [**Floating Point Arithmetic.**](http://pages.cs.wisc.edu/~smoler/x86text/lect.notes/arith.flpt.html)
* [**Floating Point Multiplication/Division**](http://www.ecs.umass.edu/ece/koren/arith/simulator/FPMul/)
* **Tài liệu lí thuyệt trên Moodle.**