TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

BÀI TẬP LỚN MÔN CẤU TRÚC RỜI RẠC

Phép tính nhị phân

Người hướng dẫn: **ThS Trần Hồng Tài**

Người thực hiện: **Nguyễn Đức Dũng- 51900732**

Lớp **: 19050302**

Khoá **: 23**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2020

LỜI CẢM ƠN

Để có đươc bài báo cáo như ngày hôm nay em xin cảm ơn thầy Trần Hồng Tài một giáo viên xuất tận tình đã giảng dạy cho em hiểu về môn thực hành cấu trúc rời rạc.

MỤC LỤC

CHƯƠNG 1 – GIỚI THIỆU

1.1 Các phép tính nhị phân:

Chúng ta đã quen với việc sử dụng các phép tính toán trên hệ cơ số 10 với các chữ số từ 0 đến 9. Nhưng bây giờ sẽ tìm hiểu về hệ nhị phân gồm là hệ chỉ gồm 2 chữ số 0 và 1. Các phép tính hệ nhị phân dường như đơn giản hơn các hệ khác vì chỉ có 2 bit 0 và 1. Vậy câu hỏi đặt ra: "Chúng được dùng ở đâu?" . Chúng được dùng trong các máy tính vì như vậy máy tính hay các máy móc sẽ dễ dàng hơn. Sau đây chúng em sẽ trình bày các thuật toán được yêu cầu.

1.2 Các hàm tính toán trong bài tập này:

1. sum(A,B): Đầu vào là 2 chuỗi A,B và các kí tự trong mỗi chuỗi sẽ là '0' hoặc '1'. Đầu ra là chuỗi C là kết quả phép **cộng** của hai số nhị phân được truyền vào dạng chuỗi.

2. dif(A,B): Đầu vào là 2 chuỗi A,B và các kí tự trong mỗi chuỗi sẽ là '0' hoặc '1'. Đầu ra là chuỗi C là kết quả phép **trừ** của hai số nhị phân được truyền vào dạng chuỗi.

3. prod(A,B): Đầu vào là 2 chuỗi A,B và các kí tự trong mỗi chuỗi sẽ là '0' hoặc '1'. Đầu ra là chuỗi C là kết quả phép **nhân** của hai số nhị phân được truyền vào dạng chuỗi.

4. bitwiseAnd(A,B): Đầu vào là 2 chuỗi A,B và các kí tự trong mỗi chuỗi sẽ là '0' hoặc '1'. Đầu ra là chuỗi C là kết quả phép **And** của hai số nhị phân được truyền vào dạng chuỗi.

5. bitwiseOr(A,B): Đầu vào là 2 chuỗi A,B và các kí tự trong mỗi chuỗi sẽ là '0' hoặc '1'. Đầu ra là chuỗi C là kết quả phép **Or** của hai số nhị phân được truyền vào dạng chuỗi.

6. bitwiseXor(A,B): Đầu vào là 2 chuỗi A,B và các kí tự trong mỗi chuỗi sẽ là ‘0’ hoặc ‘1’. Đầu ra là chuỗi C là kết quả phép **Xor** của hai số nhị phân được truyền vào dạng chuỗi.

7. bitwiseNot(A): Đầu vào là chuỗi A và các kí tự trong chuỗi sẽ là ‘0’ hoặc ‘1’. Đầu ra là chuỗi C là kết quả phép **Not** của số nhị phân được truyền vào dạng chuỗi.

8. bitwiseLeftShift(A): Đầu vào là chuỗi A và các kí tự trong chuỗi sẽ là ‘0’ hoặc ‘1’. Đầu ra là chuỗi C là kết quả phép **dịch trái** của số nhị phân được truyền vào dạng chuỗi.

9. bitwiseRightShift(A): Đầu vào là chuỗi A và các kí tự trong chuỗi sẽ là ‘0’ hoặc ‘1’. Đầu ra là chuỗi C là kết quả phép **dịch phải** của số nhị phân được truyền vào dạng chuỗi.

10. bin2Hex(A):Đầu vào là chuỗi A và các kí tự trong chuỗi sẽ là ‘0’ hoặc ‘1’. Đầu ra là chuỗi C là kết quả phép **biến đổi từ nhị phân sang thập lục phân** của số nhị phân được truyền vào dạng chuỗi.

CHƯƠNG 2 – MIÊU TẢ THUẬT TOÁN

Ghi chú:

MSSV: 51900732

Chuỗi A bằng từng kí tự trong MSSV chia lấy dư cho 2.

A = ‘11100110’

B = A+A Sau đó bỏ kí tự đầu tiên ra khỏi chuỗi từ bên trái

‘111001100’ => ‘11001100’. Suy ra B = ‘11001100’

Miêu tả các thuật toán:

2.1 Hàm sum(A,B):

Bước 1: Gán biến x bằng độ dài lớn nhất của chuỗi A và B.

Bước 2: Gọi hàm zfill() cho cả 2 chuỗi A và B. Hàm zfill là thêm số 0 vào phía trái của chuỗi sao cho độ dài của chuỗi bằng với độ dài được gọi (Trong bài này là x). Mục đích của bước 2 là để cho 2 chuỗi A và B có độ dài bằng nhau.

Bước 3: Tạo biến carry = 0, tạo biến result và tạo vòng lặp chạy từ phần tử thứ x-1 đến phần tử thứ 0 với câu lệnh for i in range(x-1, -1, -1) : .Trong vòng lặp khai báo biến r = carry(là biến nhớ), r tăng lên 1 nếu phần tử thứ i của chuỗi A là ‘1’ ngược lại không tăng. Tương tự ta xét phần tử thứ i của chuỗi B, nếu bằng ‘1’ thì tăng r lên 1 ngược lại thì không tăng. Xét điều kiện r chia lấy dư cho 2, nếu bằng 1 thì chuỗi result sẽ thêm 1 phần tử ‘1’ vào phía bên trái ngược lại thì thêm phần tử ‘0’ vào phía bên trái. Cuối cùng trong vòng lặp, nếu r < 2 thì gán carry bằng 0 ngược lại thì gán bằng 1 và quay trở lại vòng lặp cho đến khi hết phần tử thứ 0 thì thoát vòng lặp.

Bước 4: Xét biến nhớ carry, nếu khác 0 thì thêm vào bên trái result 1 phần tử '1'

Bước 5: Gán t bằng hàm removezero(result) để xóa phần tử '0' đằng trước nếu có và trả về t .

A + B = 110110010

Code:

```
def sum(A,B) :
    x = max(len(A),len(B))
    A = A.zfill(x)
    B = B.zfill(x)
    result = ''
    t = ''
    carry  = 0
    for i in range(x-1,-1,-1):
        r = carry
        r += 1 if A[i] == '1' else 0
        r += 1 if B[i] == '1' else 0
        result = ('1' if r%2 ==1 else '0') + result
        carry  = 0 if r < 2 else 1
    if carry !=0: result = '1' + result
    t = removezero(result)
    return t
```

2.2 Hàm dif(A,B):

Bước 1: Gọi hàm removezero(A) và removezero(B) để xóa kí tự '0' trong chuỗi A và B nếu có

Bước 2: - Xét độ dài của hai chuỗi A và B bằng ham len(). Nếu độ dài của A bé hơn độ dài của B thì trả về 'error' vì A bé hơn nên không thể trừ nhị phân.

- Nếu độ dài A bằng độ dài của B thì chạy vòng lặp đi từ 0 đến phần tử thứ len(A) -1 (vì A, B có độ dài bằng nhau).

- Trong vòng lặp xét trượng hợp tại phần tử thứ i, Nếu A[i] bằng ‘1’ và B[i] bằng ‘0’ thì thoát vòng lặp. Nếu A[i] bằng ‘0’ và B[i] bằng ‘0’ thì trả về ‘error’ do trường hợp này A bé hơn B

Bước 3: Gán biến x bằng độ dài lớn nhất của chuỗi A và B.

Bước 4: Gọi hàm zfill() cho cả 2 chuỗi A và B. Hàm zfill là thêm số 0 vào phía trái của chuỗi sao cho độ dài của chuỗi bằng với độ dài được gọi (Trong bài này là x). Mục đích của bước 2 là để cho 2 chuỗi A và B có độ dài bằng nhau.

Bước 5: Tạo biến carry = 0, tạo biến result và tạo vòng lặp chạy từ phần tử thứ x-1 đến phần tử thứ 0 với câu lệnh for i in range(x-1, -1, -1) : .Trong vòng lặp khai báo biến r = carry(là biến nhớ), r tăng lên 1 nếu phần tử thứ i của chuỗi A là ‘0’ ngược lại không tăng. Tương tự ta xét phần tử thứ i của chuỗi B, nếu bằng ‘1’ thì tăng r lên 1 ngược lại thì không tăng. Xét điều kiện r chia lấy dư cho 2, nếu bằng 1 thì chuỗi result sẽ thêm 1 phần tử ‘1’ vào phía bên trái ngược lại thì thêm phần tử ‘0’ vào phía bên trái. Cuối cùng trong vòng lặp, nếu r < 2 thì gán carry bằng 0 ngược lại thì gán bằng 1 và quay trở lại vòng lặp cho đến khi hết phần tử thứ 0 thì thoát vòng lặp.

Bước 6: Xét biến nhớ carry, nếu khác 0 thì thêm vào bên trái result 1 phần tử ‘1’

Bước 7: Gán t bằng hàm removezero(result) để xóa phần tử ‘0’ đằng trước nếu có và trả về t

A-B = 11010

Code:

```
def dif(A,B) :
    A = removezero(A)
    B = removezero(B)
    if (len(A) < len(B)):
        return 'error'
    if len(A) == len(B):
        for i in range(len(A)):
            if A[i] == '1' and B[i] == '0':
                break
            elif A[i] == '0' and B[i] == '1':
                return 'error'
    x = max(len(A),len(B))
    A = A.zfill(x)
    B = B.zfill(x)
    result = ''
    carry  = 0
    t = ''
    for i in range(x-1,-1,-1):
        r = carry
        r += 1 if A[i] == '0' else 0
        r += 1 if B[i] == '1' else 0
        result = ('0' if r %2 == 1 else '1') + result
        carry  = 0 if r < 2 else 1
    t = removezero(result)
    return t
```

2.3 Hàm prod(A,B):

Bước 1: Gán biến x bằng độ dài lớn nhất của chuỗi A và B.

Bước 2: Gọi hàm zfill() cho cả 2 chuỗi A và B. Hàm zfill là thêm số 0 vào phía trái của chuỗi sao cho độ dài của chuỗi bằng với độ dài được gọi (Trong bài này là x). Mục đích của bước 2 là để cho 2 chuỗi A và B có độ dài bằng nhau.

Bước 3: Tạo biến t dạng chuỗi, biến a và b dạng int.

- Mục đích của biến a để đếm số lần chạy của vòng lặp

- Mục đích của biến b để đếm số lần kí tự ‘1’ xuất hiện trong chuỗi B và để thêm sô phần tử ‘0’ vào trong chuỗi A bằng với b khi B[i] bằng ‘1’

Bước 4: Xét chuỗi A và B, nếu một trong hai chuỗi có một chuỗi bằng ‘0’ thì trả về ‘0’

Bước 5: Tạo vòng lặp for đi từ vị trí thứ cuối đến vị trí đầu của hai chuỗi A và B với câu lệnh for i in range(x-1, -1, -1) :

- Trong vòng lặp for: Nếu B[i] bằng ‘0’ thì tăng a lên 1, ngược lại thì tạo một vòng lặp while() để thêm phần tử ‘0’

- Khi thoát vòng lặp while() tăng biến a lên 1 và gọi hàm sum(t,A) để cộng tại mỗi vị trí B[i] băng ‘1’

Bước 6: Trả về t

A x B = 1011011101001000

Code:

```
def prod(A,B) :
    x = max(len(A),len(B))
    A = A.zfill(x)
    B = B.zfill(x)
    t = '0'
    a = 0
    b = 0
    if removezero(A)== '0' or removezero(B) == '0':
        return '0'
    for i in range(x-1,-1,-1):
        if B[i] == '0':
            a = a+1
        else:
            while b < a:
                A = A + '0'
                b = b +1
            a = a + 1
            t = sum(t,A)
    return t
```

2.4 Hàm bitwiseAnd(A,B):

Bước 1: Gán biến x bằng độ dài lớn nhất của chuỗi A và B.

Bước 2: Gọi hàm zfill() cho cả 2 chuỗi A và B. Hàm zfill là thêm số 0 vào phía trái của chuỗi sao cho độ dài của chuỗi bằng với độ dài được gọi (Trong bài này là x). Mục đích của bước 2 là để cho 2 chuỗi A và B có độ dài bằng nhau.

Bước 3: Tạo biên t dạng chuỗi tạo vòng lặp for đi từ phần tử cuối đến đầu

for i in range(x-1, -1, -1) :

- Trong vòng lặp xét điều kiện nếu A[i] bằng '0' hoặc B[i] bằng '0' thì cộng vào phía bên trái của t phần tử '0' ngược lại thì cộng vào bên trái phần tử '1'

Bước 4: Trả về hàm removezero(t) (để xóa '0' đầu nếu có)

A and B = 11000100

Code:

```
def bitwiseAnd(A,B):
    x = max(len(A),len(B))
    A = A.zfill(x)
    B = B.zfill(x)
    t = ''
    for i in range(x-1,-1,-1) :
        if A[i] == '0' or B[i] == '0':
            t = '0'+ t
        else:
            t = '1'+ t
    t =  removezero(t)
    return t
```

2.5 Hàm bitwiseOr(A,B):

Bước 1: Gán biến x bằng độ dài lớn nhất của chuỗi A và B.

Bước 2: Gọi hàm zfill() cho cả 2 chuỗi A và B. Hàm zfill là thêm số 0 vào phía trái của chuỗi sao cho độ dài của chuỗi bằng với độ dài được gọi (Trong bài này là x). Mục đích của bước 2 là để cho 2 chuỗi A và B có độ dài bằng nhau.

Bước 3: Tạo biên t dạng chuỗi tạo vòng lặp for đi từ phần tử cuối đến đầu for i in range(x-1, -1, -1) :

- Trong vòng lặp xét điều kiện nếu A[i] bằng '1' hoặc B[i] bằng '1' thì cộng vào phía bên trái của t phần tử '1' ngược lại thì cộng vào bên trái phần tử '0'

Bước 4: Trả về hàm removezero(t) (để xóa '0' đầu nếu có)

A or B = 11101110

Code:

```
def bitwiseOr(A,B):
    x = max(len(A),len(B))
    A = A.zfill(x)
    B = B.zfill(x)
    t = ''
    result =''
    for i in range(x-1,-1,-1) :
        if A[i] == '1' or B[i] == '1':
            t = '1'+ t
        else:
            t = '0'+ t
    return removezero(t)
```

2.6 Hàm bitwiseXor(A,B):

Bước 1: Gán biến x bằng độ dài lớn nhất của chuỗi A và B.

Bước 2: Gọi hàm zfill() cho cả 2 chuỗi A và B. Hàm zfill là thêm số 0 vào phía trái của chuỗi sao cho độ dài của chuỗi bằng với độ dài được gọi (Trong bài này là x). Mục đích của bước 2 là để cho 2 chuỗi A và B có độ dài bằng nhau.

Bước 3: Tạo biên t dạng chuỗi và tạo vòng lặp for đi từ phần tử cuối đến đầu for i in range(x-1, -1, -1) :

- Trong vòng lặp xét điều kiện nếu A[i] bằng B[i] thì cộng vào phía bên trái của t phần tử '0' ngược lại thì cộng vào bên trái phần tử '1'

Bước 4: Trả về hàm removezero(t) (để xóa ‘0’ đầu nếu có)

A xor B = 101010

Code:

```
def bitwiseXor(A,B):
    x = max(len(A),len(B))
    A = A.zfill(x)
    B = B.zfill(x)
    t = ''
    result =''
    for i in range(x-1,-1,-1) :
        if A[i] == B[i] :
            t = '0' + t
        else:
            t = '1'+ t
    return removezero(t)
```

2.7 Hàm bitwiseNot(A) :

Bước 1: Tạo biến t kiểu chuỗi và x bằng độ dài của chuỗi với hàm len(A)

Bước 2: Tạo biên t dạng chuỗi tạo vòng lặp for đi từ phần tử cuối đến đầu for i in range(x-1, -1, -1) :

- Xét điều kiện nếu A[i] bằng ‘0’ thì cộng vào phía bên trái của t thêm phần tử ‘1’ngược lại thì cộng vào phía bên trái phần tử ‘0’

Bước 3: Trả về hàm removezero(t) (để xóa ‘0’ đầu nếu có)

Not A = 11001

Code:

```
def bitwiseNot(A):
    t = ''
    x = len(A)
    for i in range(x-1,-1,-1):
        if A[i] == '0':
            t = '1' + t
        else:
            t = '0' + t
    return removezero(t)
```

2.8 Hàm bitwiseLeftShift(A):

Bước 1: Tạo biến t kiểu chuỗi

Bước 2: Gán t bằng phần tử thứ 1 của chuỗi A đến phần tử cuối cùng cộng với phần tử thứ 0

Bước 3: Trả về hàm removezero(t) (để xóa '0' đầu nếu có)

Left shift A : 11001101

Code:

```
def bitwiseLeftShift(A):
    t = ''
    t = A[1:] + A[0]
    return removezero(t)
```

2.9 Hàm bitwiseRightShift(A):

Bước 1: Tạo biến t kiểu chuỗi

Bước 2: Gán t bằng phần tử cuối cùng của chuỗi A cộng với phần tử thứ 0 đến phần tử kế cuối

Bước 3: Trả về hàm removezero(t) (để xóa '0' đầu nếu có)

Right Left A: 1110011

Code:

```
def bitwiseRightShift(A):
    t = ''
    t = A[len(A) - 1] + A[: len(A)-1]
    return removezero(t)
```

2.10 Hàm bin2Hex(A):

Bước 1: Gọi hàm removezero(A) để xóa ‘0’ đứng đầu

Bươc 2: Gán t biến kiểu dictionary bằng việc gọi hàm createDict() (hàm gọi biến chứa dữ liệu chuyển tư nhị phân sang thập lục phân)

Bước 3: Tạo biến c kiểu chuỗi

Bước 4: Gán biến l là độ dài phần bù của chuỗi A để l chia hết cho 4 bằng cách l bằng 4-(len(A) % 4)

Bước 5: Nếu l khác 0 thì thêm phần tử ‘0’ vào chuỗi A để độ dài chuỗi A chia hết cho 4

Bước 6: Tạo vòng lặp for bắt đầu từ phần tử đầu tiên đến kết thúc, bước nhảy là 4 for i in range(0,len(A),4):

- Trong vòng lặp thì thêm vào chuỗi c các phần tử trong t tương ứng từ phần tử thứ i đến i+4 của chuỗi A

Bước 7: Trả về c

Bin to Hex A :

Từ 11100110 -> E6

Thập phân (Decimal)	Nhị phân (Binary)	Thập lục phân (Hexadecimal)	Bát phân (Octal)
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	8	10
9	1001	9	11
10	1010	A	12
11	1011	B	13
12	1100	C	14
13	1101	D	15
14	1110	E	16
15	1111	F	17

Code:

```
def bin2Hex(A):
    A = removezero(A)
    t = createDict()
    c = ''
    l = 4-(len(A) %4)
    if(l%4 !=0):
        A = A.zfill(l+len(A))
    for i in range(0,len(A),4):
        c += t[A[i:i+4]]

    return c
```

2.11 Các hàm phụ:

2.11.1 Hàm removezero(A):

Tạo vòng lặp đi từ phần tử đầu tiền đến phần tử cuối cùng trong chuỗi A

- Trong vòng lặp xét điều kiện A[i] bằng ‘1’ thì trả về chuỗi từ vị trí i đến hết ngược lại trả về ‘0’

Code:

```
def removezero(A):
    for i in range(len(A)) :
        if A[i] == '1':
            return A[i:]
    return '0'
```

2.11.2 Hàm createDict() :

Bước 1: Tạo chuỗi a bằng ‘0’ và biến t kiểu dictionary (key và value)

Bước 2: Tạo vòng lặp for đi từ 0 đến 15 for i in range(16):

- Trong vòng lặp nếu i <10 thì gán key bằng a.zfill(4) (tạo 1 chuỗi 4 phần tử) và value bằng str(i) ngược lại thì gán key bằng chuỗi a và value bằng các chữ cái từ A đến F tương ứng.

-Tăng biến a lên 1 bằng cách gọi hàm sum(a,’1’)

Bước 3: Sau khi chạy hết vòng lặp thì trả về t

Code:

```
def createDict():
    a='0'
    t = {}
    for i in range(16):
        if i<10:

            t[a.zfill(4)]= str(i)

        else:

            t[a] = chr(ord('A')+i-10)
        a = sum(a,'1')
    return t
```

CHƯƠNG 3 – KẾT QUẢ