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

TÌM HIỂU
CÁC PHÉP TOÁN TRÊN SỐ NHỊ PHÂN

Người hướng dẫn: **THẦY DUNG CẨM QUANG**

Người thực hiện: **NGUYỄN THẾ TRƯỜNG – 51900780**

Lớp : **19050302**

Khoá : **23**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2020

LỜI CẢM ƠN

Em xin chân thành cảm ơn thầy Dung Cẩm Quang đã hướng dẫn em cách làm bài báo cáo này. Sự hướng dẫn nhiệt tình của thầy và những kiến thức học được từ thầy đã giúp em hoàn thành bài báo cáo này một cách hoàn thiện nhất. Chúc thầy luôn có sức khỏe và đạt nhiều thành công hơn trong cuộc sống.

ĐỒ ÁN ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Tôi xin cam đoan đây là sản phẩm đồ án của riêng tôi và được sự hướng dẫn của Thầy Dung Cẩm Quang. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình. Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày tháng năm

Tác giả

(ký tên và ghi rõ họ tên)

Nguyễn Thế Trường

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

Phần xác nhận của GV hướng dẫn

_____

_____

_____

_____

_____

_____

_____

Tp. Hồ Chí Minh, ngày tháng năm
(kí và ghi họ tên)

Phần đánh giá của GV chấm bài

_____

_____

_____

_____

_____

_____

_____

Tp. Hồ Chí Minh, ngày tháng năm
(kí và ghi họ tên)

TÓM TẮT

Bài báo cáo này giới thiệu về hệ nhị phân, các hàm tính toán có trong bài tập lớn. Mô tả, giải thích các thuật toán sử dụng để thực hiện các hàm trên. Dựa vào những kiến thức thao tác với hệ nhị phân đã học ở môn Tổ chức máy tính, Cấu trúc rời rạc và sử dụng ngôn ngữ lập trình Python để thực hiện bài tập lớn này.

MỤC LỤC

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

DANH MỤC BẢNG

DANH MỤC HÌNH

CHƯƠNG 1 – GIỚI THIỆU HỆ NHỊ PHÂN

1.1 Định nghĩa và chức năng

1.1.1 Định nghĩa

Hệ nhị phân (hay hệ đếm cơ số hai hoặc mã nhị phân) là một hệ đếm dùng hai ký tự để biểu đạt một giá trị số, bằng tổng số các lũy thừa của 2. Hai ký tự đó thường là 0 và 1; chúng thường được dùng để biểu đạt hai giá trị hiệu điện thế tương ứng (có hiệu điện thế, hoặc hiệu điện thế cao là 1 và không có, hoặc thấp là 0).

1.1.2 Chức năng

Do có ưu điểm tính toán đơn giản, dễ dàng thực hiện về mặt vật lý, chẳng hạn như trên các mạch điện tử, hệ nhị phân trở thành một phần kiến tạo căn bản trong các máy tính đương thời.

1.2 Các hàm tính toán trong bài tập lớn

1.2.1 Hàm sum(A,B)

- Đầu vào là 2 chuỗi A, B được biểu diễn dưới dạng nhị phân. Đầu ra là 1 chuỗi biểu diễn kết quả của phép cộng A và B.
- Quy tắc cộng 2 số nhị phân:
    - 0 + 0 = 0
    - 0 + 1 = 1
    - 1 + 0 = 1
    - 1 + 1 = 0 (Nhớ 1)

1.2.2 Hàm dif(A,B)

- Đầu vào là 2 chuỗi A, B được biểu diễn dưới dạng nhị phân. Đầu ra là 1 chuỗi biểu diễn kết quả của phép trừ A cho B.
- Quy tắc trừ 2 số nhị phân:
    - 0 – 0 = 0
    - 0 – 1 = 1 (Mượn 1)
    - 1 – 0 = 1
    - 1 – 1 = 0

1.2.3 Hàm prod(A,B)

- Đầu vào là 2 chuỗi A, B được biểu diễn dưới dạng nhị phân. Đầu ra là 1 chuỗi biểu diễn kết quả của phép nhân A và B.
- Phép tính nhân trong hệ nhị phân cũng tương tự như phương pháp làm trong hệ thập phân. Hai số A và B được nhân với nhau bởi những tích số cục bộ: với mỗi con số ở B, tích của nó với một con số trong A được tính và viết xuống một hàng mới, mỗi hàng mới phải dịch chuyển vị trí sang bên trái, con số cuối cùng ở bên phải đứng cùng cột với vị trí của con số ở trong B đang dùng. Tổng của các tích cục bộ này cho ta kết quả tích số cuối cùng. Vì chỉ có hai con số trong hệ nhị phân, nên chỉ có 2 kết quả khả quan trong tích cục bộ:
    - Nếu con số trong B là 0, tích cục bộ sẽ là 0
    - Nếu con số trong B là 1, tích cục bộ sẽ là A

1.2.4 Hàm bitwiseAnd(A,B)

Đầu vào là 2 chuỗi A, B được biểu diễn dưới dạng nhị phân. Đầu ra là 1 chuỗi biểu diễn kết quả phép AND của A và B.

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

Bảng 1.1: Bảng chân trị phép AND

1.2.5 Hàm bitwiseOr(A,B)

Đầu vào là 2 chuỗi A, B được biểu diễn dưới dạng nhị phân. Đầu ra là 1 chuỗi biểu diễn kết quả phép OR của A và B.

A	B	A OR B
0	0	0
0	1	1

1	0	1
1	1	1

Bảng 1.2: Bảng chân trị phép OR

1.2.6 Hàm bitwiseXor(A,B)

Đầu vào là 2 chuỗi A, B được biểu diễn dưới dạng nhị phân. Đầu ra là 1 chuỗi biểu diễn kết quả phép XOR của A và B.

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Bảng 1.3: Bảng chân trị phép XOR

1.2.7 Hàm bitwiseNot(A)

Đầu vào là chuỗi A được biểu diễn dưới dạng nhị phân. Đầu ra là 1 chuỗi biểu diễn kết quả phép NOT của A.

A	NOT A
0	1
1	0

Bảng 1.4: Bảng chân trị phép NOT

1.2.8 Hàm bitwiseLeftShift(A)

Đầu vào là chuỗi A được biểu diễn dưới dạng nhị phân. Đầu ra là 1 chuỗi biểu diễn kết quả phép Left Shift của A.

1.2.9 Hàm bitwiseRightShift(A)

Đầu vào là chuỗi A được biểu diễn dưới dạng nhị phân. Đầu ra là 1 chuỗi biểu diễn kết quả phép Right Shift của A.

1.2.10 Hàm bin2Hex(A)

Đầu vào là chuỗi A được biểu diễn dưới dạng nhị phân. Đầu ra là 1 chuỗi biểu diễn kết quả của phép chuyển A từ hệ nhị phân sang hệ thập lục phân.

Hệ nhị phân	Hệ thập lục phân
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Bảng 1.5: Bảng quy tắc chuyển đổi hệ nhị phân sang hệ thập lục phân

CHƯƠNG 2 – MÔ TẢ THUẬT TOÁN

2.1 Lấy ví dụ tính toán

MSSV: 51900780

Lấy chuỗi A bằng cách lấy từng ký tự trong MSSV chia lấy dư cho 2.

⇨ A = "11100100"

Lấy chuỗi B bằng cách: B = A + A, sau đó bỏ kí tự đầu tiên bên trái ra khỏi chuỗi.

⇨ B = "11001000"

2.2 Mô tả các hàm trong bài

2.2.1 Hàm sum(A,B)

- Bước 1
  - Lấy giá trị lớn hơn giữa kích thước chuỗi A và kích thước chuỗi B, sau đó gán cho biến maxLen.
  - Dùng hàm zfill() để bổ sung các ký tự "0" vào bên trái chuỗi A (hoặc B) nếu kích thước chuỗi A (hoặc B) nhỏ hơn maxLen. Bây giờ, A và B có cùng kích thước.
- Bước 2
  - Khởi tạo chuỗi rỗng có tên là result, chuỗi này dùng để lưu kết quả.
  - Khởi tạo biến đếm count và gán cho nó giá trị 0.
- Bước 3
  - Dùng vòng lặp For, cho i chạy từ maxLen – 1 đến 0, qua mỗi lần lặp giảm i đi 1 đơn vị.
  - Khởi tạo biến tạm temp và lấy giá trị của count.
  - Ta xét phần tử A[i] và B[i], nếu A[i] = "1" thì tăng temp lên 1 đơn vị, ngược lại nếu A[i] = "0" thì giữ nguyên temp. Tương tự với B[i].
  - Nếu temp chia hết cho 2 thì thêm "0" vào bên trái result, ngược lại nếu temp không chia hết cho 2 thì thêm "1" vào bên trái result.
  - Nếu temp < 2 thì gán count cho 0, nếu temp ≥ 2 thì gán count cho 1.
  - Thoát vòng lặp.
- Bước 4

- Kiểm tra nếu count ≠ 0 thì thêm "1" vào bên trái result.
- Trả về hàm removeLeadZero() có tham số truyền vào là result để xóa các phần tử "0" dư thừa ở đầu chuỗi result.

```
def sum(A, B):
    maxLen = max(len(A), len(B))
    A = A.zfill(maxLen)
    B = B.zfill(maxLen)
    result = ""
    count = 0
    for i in range(maxLen-1, -1, -1):
        temp = count
        temp += 1 if A[i] == "1" else 0
        temp += 1 if B[i] == "1" else 0
        result = ("0" if temp%2 == 0 else "1") + result
        count = 0 if temp < 2 else 1
    if count != 0:
        result = "1" + result
    return removeLeadZero(result)
```

Hình 2.1 Hình ảnh code hàm sum(A,B)

2.2.2 Hàm dif(A,B)

- Bước 1: Dùng hàm int() để chuyển A, B sang dạng số nguyên rồi kiểm tra xem A < B hay không. Nếu A < B thì trả về chuỗi "error".
- Bước 2
  - Lấy giá trị lớn hơn giữa kích thước chuỗi A và kích thước chuỗi B, sau đó gán cho biến maxLen.
  - Dùng hàm zfill() để bổ sung các ký tự "0" vào bên trái chuỗi A (hoặc B) nếu kích thước chuỗi A (hoặc B) nhỏ hơn maxLen. Bây giờ, A và B có cùng kích thước.
- Bước 3
  - Khởi tạo chuỗi rỗng có tên là result, chuỗi này dùng để lưu kết quả.
  - Khởi tạo biến đếm count và gán cho nó giá trị 0.
- Bước 4
  - Dùng vòng lặp For, cho i chạy từ maxLen – 1 đến 0, qua mỗi lần lặp giảm i đi 1 đơn vị.

- Khởi tạo biến tạm temp. Tiếp tục dùng hàm int() để chuyển A[i], B[i] sang số nguyên. Sau đó gán (A[i] – B[i]) cho temp.

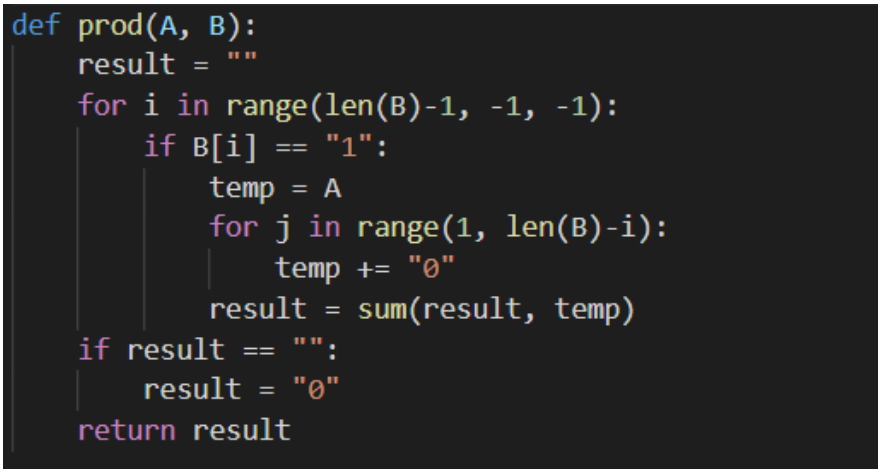
- Bước 5: Ta xét giá trị của temp và count.
  - Trường hợp temp < 0. Nếu count = 0 thì thêm "1" vào bên trái result và gán 1 cho count. Nếu count ≠ 0 thì thêm "0" vào bên trái result.
  - Trường hợp temp = 0. Nếu count = 0 thì thêm "0" vào bên trái result. Còn nếu count ≠ 0 thì thêm "1" vào bên trái result.
  - Trường hợp còn lại. Nếu count = 0 thì thêm "1" vào bên trái result. Còn nếu count ≠ 0 thì thêm "0" vào bên trái result và gán 0 cho count.
  - Thoát vòng lặp.
- Bước 6: Trả về hàm removeLeadZero() có tham số truyền vào là result để xóa các phần tử "0" dư thừa ở đầu chuỗi result.

```
def dif(A, B):
    if(int(A) < int(B)):
        return "error"
    else:
        maxLen = max(len(A), len(B))
        A = A.zfill(maxLen)
        B = B.zfill(maxLen)
        result = ""
        count = 0
        for i in range(maxLen-1, -1, -1):
            temp = int(A[i]) - int(B[i])
            if temp < 0:
                if count == 0:
                    result = "1" + result
                    count = 1
                else:
                    result = "0" + result
            elif temp == 0:
                if count == 0:
                    result = "0" + result
                else:
                    result = "1" + result
            else:
                if count == 0:
                    result = "1" + result
                else:
                    count = 0
                    result = "0" + result
        return removeLeadZero(result)
```

Hình 2.2: Hình ảnh code hàm dif(A,B)

2.2.3 Hàm prod(A,B)

- Bước 1: Khởi tạo chuỗi rỗng có tên là result, chuỗi này dùng để lưu kết quả.
- Bước 2:
    - Dùng vòng lặp For, cho i chạy từ (kích thước của chuỗi B – 1) đến 0, qua mỗi lần lặp giảm i đi 1 đơn vị.
    - Xét phần tử B[i], nếu B[i] = "1" thì khởi tạo biến temp và gán chuỗi A cho nó.
    - Tiếp tục dùng vòng lặp For, cho j chạy từ 1 đến (kích thước của chuỗi B – i – 1), qua mỗi vòng lặp tăng j lên 1 đơn vị và thêm "0" vào bên phải temp. Sau đó kết thúc vòng lặp này.
    - Dùng hàm sum() đã định nghĩa ở trên, tham số truyền vào là result và temp. Sau đó gán lại cho result.
    - Thoát vòng lặp For.
- Bước 3: Thoát vòng lặp For còn lại, sau đó kiểm tra xem result có phải là chuỗi rỗng hay không. Nếu result là chuỗi rỗng thì trả về chuỗi "0", còn nếu không phải thì trả về chuỗi result.



```
def prod(A, B):
    result = ""
    for i in range(len(B)-1, -1, -1):
        if B[i] == "1":
            temp = A
            for j in range(1, len(B)-i):
                temp += "0"
            result = sum(result, temp)
    if result == "":
        result = "0"
    return result
```

Hình 2.3: Hình ảnh code hàm prod(A,B)

2.2.4 Hàm bitwiseAnd(A, B)

- Bước 1
    - Lấy giá trị lớn hơn giữa kích thước chuỗi A và kích thước chuỗi B, sau đó gán cho biến maxLen.

  - Dùng hàm zfill() để bổ sung các phần tử "0" vào bên trái chuỗi A (hoặc B) nếu kích thước chuỗi A (hoặc B) nhỏ hơn maxLen. Bây giờ, A và B có cùng kích thước.
- Bước 2: Khởi tạo 1 chuỗi rỗng có tên là result, chuỗi này dùng để lưu kết quả.
- Bước 3
  - Dùng vòng lặp For, cho i chạy từ maxLen – 1 đến 0, qua mỗi vòng lặp giảm i đi 1 đơn vị.
  - Ta xét A[i] và B[i]. Nếu A[i] = "1" và B[i] = "1" thì thêm "1" vào bên trái result, còn nếu không phải thì thêm "0" vào bên trái result.
- Bước 4: Thoát vòng lặp và trả về hàm removeLeadZero() có tham số truyền vào là result để xóa các phần tử "0" dư thừa ở đầu chuỗi result.

```
def bitwiseAnd(A, B):
    maxLen = max(len(A), len(B))
    A = A.zfill(maxLen)
    B = B.zfill(maxLen)
    result = ""
    for i in range(maxLen-1, -1, -1):
        if A[i] == "1" and B[i] == "1":
            result = "1" + result
        else:
            result = "0" + result
    return removeLeadZero(result)
```

Hình 2.4: Hình ảnh code hàm bitwiseAnd(A,B)

2.2.5 Hàm bitwiseOr(A,B)

- Bước 1
  - Lấy giá trị lớn hơn giữa kích thước chuỗi A và kích thước chuỗi B, sau đó gán cho biến maxLen.
  - Dùng hàm zfill() để bổ sung các phần tử "0" vào bên trái chuỗi A (hoặc B) nếu kích thước chuỗi A (hoặc B) nhỏ hơn maxLen. Bây giờ, A và B có cùng kích thước.
- Bước 2: Khởi tạo 1 chuỗi rỗng có tên là result, chuỗi này dùng để lưu kết quả.
- Bước 3

- Dùng vòng lặp For, cho i chạy từ maxLen – 1 đến 0, qua mỗi vòng lặp giảm i đi 1 đơn vị.
- Ta xét A[i] và B[i]. Nếu A[i] = "0" và B[i] = "0" thì thêm "0" vào bên trái result, còn nếu không phải thì thêm "1" vào bên trái result.

- Bước 4: Thoát vòng lặp và trả về hàm removeLeadZero() có tham số truyền vào là result để xóa các phần tử "0" dư thừa ở đầu chuỗi result.

```
def bitwiseOr(A,B):
    maxLen = max(len(A), len(B))
    A = A.zfill(maxLen)
    B = B.zfill(maxLen)
    result = ""
    for i in range(maxLen-1, -1, -1):
        if A[i] == "0" and B[i] == "0":
            result = "0" + result
        else:
            result = "1" + result
    return removeLeadZero(result)
```

Hình 2.5: Hình ảnh code hàm bitwiseOr(A,B)

2.2.6 Hàm bitwiseXor(A,B)

- Bước 1
  - Lấy giá trị lớn hơn giữa kích thước chuỗi A và kích thước chuỗi B, sau đó gán cho biến maxLen.
  - Dùng hàm zfill() để bổ sung các phần tử "0" vào bên trái chuỗi A (hoặc B) nếu kích thước chuỗi A (hoặc B) nhỏ hơn maxLen. Bây giờ, A và B có cùng kích thước.
- Bước 2: Khởi tạo 1 chuỗi rỗng có tên là result, chuỗi này dùng để lưu kết quả.
- Bước 3
  - Dùng vòng lặp For, cho i chạy từ maxLen – 1 đến 0, qua mỗi vòng lặp giảm i đi 1 đơn vị.
  - Ta xét A[i] và B[i]. Nếu A[i] = B[i] thì thêm "0" vào bên trái result, còn nếu không phải thì thêm "1" vào bên trái result.

- Bước 4: Thoát vòng lặp và trả về hàm removeLeadZero() có tham số truyền vào là result để xóa các phần tử "0" dư thừa ở đầu chuỗi result.

```
def bitwiseXor(A,B):
    maxLen = max(len(A), len(B))
    A = A.zfill(maxLen)
    B = B.zfill(maxLen)
    result = ""
    for i in range(maxLen-1, -1, -1):
        result = ("0" if A[i] == B[i] else "1") + result
    return removeLeadZero(result)
```

Hình 2.6: Hình ảnh code hàm bitwiseXor(A,B)

2.2.7 Hàm bitwiseNot(A)

- Bước 1: Khởi tạo chuỗi rỗng có tên là result, chuỗi này dùng để lưu kết quả.
- Bước 2
    - Dùng vòng lặp For, cho i chạy từ 0 đến (kích thước chuỗi A – 1), qua mỗi vòng lặp tăng i lên 1 đơn vị.
    - Ta xét phần tử A[i], nếu A[i] = "1" thì thêm "0" vào bên phải result, còn nếu A[i] = "0" thì thêm "1" vào bên phải result.
- Bước 3: Thoát vòng lặp và trả về hàm removeLeadZero() có tham số truyền vào là result để xóa các phần tử "0" dư thừa ở đầu chuỗi result.

```
def bitwiseNot(A):
    result = ""
    for i in range(len(A)):
        result += "0" if A[i] == "1" else "1"
    return removeLeadZero(result)
```

Hình 2.7: Hình ảnh code hàm bitwiseNot(A)

2.2.8 Hàm bitwiseLeftShift(A)

- Bước 1: Khởi tạo chuỗi rỗng có tên là result, chuỗi này dùng để lưu kết quả.
- Bước 2:

  - Dùng vòng lặp For, cho i chạy từ 1 đến (kích thước chuỗi A – 1), qua mỗi vòng lặp tăng i lên 1 đơn vị.
  - Thêm giá trị của phần tử A[i] vào bên phải chuỗi result.
- Bước 3: Thoát vòng lặp, sau đó thêm giá trị của phần tử A[0] vào bên phải chuỗi result.
- Bước 4: Trả về hàm removeLeadZero() có tham số truyền vào là result để xóa các phần tử "0" dư thừa ở đầu chuỗi result.

```
def bitwiseLeftShift(A):
    result = ""
    for i in range(1, len(A)):
        result += A[i]
    result += A[0]
    return removeLeadZero(result)
```

Hình 2.8: Hình ảnh code hàm bitwiseLeftShift(A)

2.2.9 Hàm bitwiseRightShift(A)

- Bước 1: Khởi tạo chuỗi rỗng có tên là result, chuỗi này dùng để lưu kết quả.
- Bước 2: Thêm giá trị của phần tử A[kích thước chuỗi A – 1] vào chuỗi result.
- Bước 3:
  - Dùng vòng lặp For, cho i chạy từ 0 đến (kích thước chuỗi A – 2), qua mỗi vòng lặp tăng i lên 1 đơn vị.
  - Thêm giá trị của phần tử A[i] vào bên phải chuỗi result.
- Bước 4: Thoát vòng lặp và trả về hàm removeLeadZero() có tham số truyền vào là result để xóa các phần tử "0" dư thừa ở đầu chuỗi result.

```
def bitwiseRightShift(A):
    result = ""
    result += A[len(A)-1]
    for i in range(len(A)-1):
        result += A[i]
    return removeLeadZero(result)
```

Hình 2.9: Hình ảnh code hàm bitwiseRightShift(A)

2.2.10 Hàm bin2Hex(A)

- Bước 1
  - Khởi tạo 1 dictionary rỗng có tên là dic dùng để lưu các quy tắc chuyển từ hệ nhị phân sang hệ thập lục phân.
  - Khởi tạo chuỗi tmp có giá trị là "0".
- Bước 2
  - Dùng vòng lặp For, cho i chạy từ 0 đến 15, qua mỗi vòng lặp tăng i lên 1 đơn vị.
  - Kiểm tra nếu i < 10, ta dùng hàm zfill(4) cho tmp để nếu tmp có số phần tử < 4 thì sẽ thêm các phần tử "0" vào bên trái sao cho có đủ 4 phần tử, dùng hàm str() để chuyển 1 số nguyên sang chuỗi, sau đó gán chuỗi i cho dic[tmp.zfill(4)]. Còn nếu i > 10 thì ta dùng hàm chr() để chuyển 1 số sang 1 chữ cái trong mã ASCII, sau đó gán chữ đó cho dic[tmp.zfill(4)].
  - Dùng hàm sum() có tham số truyền vào là tmp và "1". Thực hiện hàm này rồi gán nó cho tmp.
- Bước 3: Kết thúc vòng lặp, khởi tạo biến maxLen và gán (kích thước chuỗi A + 4 – (kích thước chuỗi A chia lấy dư cho 4)) cho nó.
- Bước 4: Dùng hàm zfill() với tham số truyền vào là maxLen cho chuỗi A.
- Bước 5: Khởi tạo 1 chuỗi rỗng có tên là result, chuỗi này dùng để lưu kết quả.
- Bước 6
  - Dùng vòng lặp For, cho i chạy từ maxLen – 1 đến 0, qua mỗi vòng lặp giảm i đi 4 đơn vị.
  - Khởi tạo chuỗi rỗng có tên là temp.

- Tiếp tục dùng vòng lặp For, cho j chạy từ i đến i – 4, qua mỗi vòng lặp giảm j đi 1 đơn vị. Thêm phần tử A[j] vào bên trái chuỗi temp. Sau đó kết thúc vòng lặp này.
- Tiếp tục dùng thêm vòng lặp For nữa, dùng biến item để duyệt tất cả các phần tử của dic. Kiểm tra xem có phần tử item nào bằng temp không, nếu có thì thêm phần tử dic[item] vào bên trái result.

- Bước 7: Thoát tất cả các vòng lặp và trả về hàm removeLeadZero() có tham số truyền vào là result để xóa các phần tử "0" dư thừa ở đầu chuỗi result.

```
def bin2Hex(A):
    dic = {}
    tmp = "0"
    for i in range(16):
        if i < 10:
            dic[tmp.zfill(4)] = str(i)
        else:
            dic[tmp.zfill(4)] = chr(i%10 + 65)
        tmp = sum(tmp, "1")
    maxLen = len(A) + (4 - len(A)%4)
    A = A.zfill(maxLen)
    result = ""
    for i in range(maxLen-1, -1, -4):
        temp = ""
        for j in range(i, i-4, -1):
            temp = A[j] + temp
        for item in dic:
            if item == temp:
                result = dic[item] + result
    return removeLeadZero(result)
```

Hình 2.10: Hình ảnh code hàm bin2hex(A)

2.2.11 Các hàm phụ được sử dụng trong bài

2.2.11.1 Hàm removeLeadZero(A)

- Hàm này em tự định nghĩa. Đầu vào là chuỗi A được biểu diễn dưới dạng nhị phân. Đầu ra là 1 chuỗi đã loại bỏ những phần tử "0" dư thừa ở đầu chuỗi A.

- Bước 1: Dùng vòng lặp For, cho i chạy từ 0 đến kích thước chuỗi A – 1. Kiểm tra nếu A[i] ≠ "0" thì trả về chuỗi A từ phần tử thứ i trở về sau. Nếu không có phần tử A[i] ≠ "0" nào thì chuyển qua bước 2.
- Bước 2: Thoát vòng lặp và trả về chuỗi "0".

```
def removeLeadZero(A):
    for i in range(len(A)):
            if A[i] != "0":
                return A[i:]
    return "0"
```

Hình 2.11: Hình ảnh code hàm removeLeadZero(A)

2.2.11.2 Hàm zfill(a)

Hàm này dùng để thêm các phần tử "0" vào bên trái chuỗi sao cho chuỗi có số phần tử đúng bằng a.

2.2.11.3 Hàm int(s)

Hàm này dùng để chuyển chuỗi s sang số nguyên.

2.2.11.4 Hàm str(n)

Hàm này dùng để chuyển số nguyên n sang dạng chuỗi.

2.2.11.5 Hàm chr(n)

Hàm này dùng để chuyển đổi số nguyên n sang chữ cái theo mã ASCII.

CHƯƠNG 3 – KẾT QUẢ

51900780.py

```python
# Exercise 10
def bin2Hex(A):
    dic = {}
    tmp = "0"
    for i in range(16):
        if i < 10:
            dic[tmp.zfill(4)] = str(i)
        else:
            dic[tmp.zfill(4)] = chr(i%10 + 65)
        tmp = sum(tmp, "1")
    maxLen = len(A) + (4 - len(A)%4)
    A = A.zfill(maxLen)
    result = ""
    for i in range(maxLen-1, -1, -4):
        temp = ""
        for j in range(i, i-4, -1):
            temp = A[j] + temp
        for item in dic:
            if item == temp:
                result = dic[item] + result
    return removeLeadZero(result)

A = "11100100"
B = "11001000"
print(" A = " + A)
print(" B = " + B)
print(" A + B = " + sum(A, B))
print(" A - B = " + dif(A, B))
print(" A * B = " + prod(A, B))
print(" A AND B = " + bitwiseAnd(A, B))
print(" A OR B = " + bitwiseOr(A, B))
print(" A XOR B = " + bitwiseXor(A, B))
print(" NOT A = " + bitwiseNot(A))
print(" LEFT SHIFT OF A: " + bitwiseLeftShift(A))
print(" RIGHT SHIFT OF A: " + bitwiseRightShift(A))
print(" CONVERT A TO HEX: " + bin2Hex(A))
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\Documents\TDTU\Discrete Structures\Lab\Big Exercise> & "C:/U
900780.py"
 A = 11100100
 B = 11001000
 A + B = 110101100
 A - B = 11100
 A * B = 1011001000100000
 A AND B = 11000000
 A OR B = 11101100
 A XOR B = 101100
 NOT A = 11011
 LEFT SHIFT OF A: 11001001
 RIGHT SHIFT OF A: 1110010
 CONVERT A TO HEX: E4
PS D:\Documents\TDTU\Discrete Structures\Lab\Big Exercise>
```

Hình 3.1: Kết quả thực hiện các hàm