Objective

The objective of this lab is to implement C programs to perform binary addition and subtraction of two decimal numbers, including handling of negative numbers using two's complement representation.

Algorithm

Common Functions

1. Convert the decimal number to binary and store it in an array.
2. If the number is negative, compute the two's complement for the binary number.
3. Print the binary number.

Addition

1. Start.
2. Initialize and take number inputs in decimal.
3. Convert the first decimal number to binary. If the number is negative, compute the two's complement for the number and store it.
4. Convert the second decimal number to binary. If the number is negative, compute the two's complement for the number and store it.
5. Initialize an array to store the sum of the two binary numbers.
6. Add the two binary numbers and store the result.
7. Print the binary sum.
8. Stop.

Subtraction

1. Start.
2. Initialize and take number inputs in decimal.
3. Convert the first decimal number to binary. If the number is negative, compute the two's complement for the number and store it.
4. Convert the second decimal number to binary. If the number is negative, compute the two's complement for the number and store it.
5. Compute the two's complement of the second binary number.
6. Add the first binary number to the two's complement of the second binary number.
7. Print the binary difference.
8. Stop.

Header File: binary_operations.h

```
#ifndef BINARY_OPERATIONS_H
#define BINARY_OPERATIONS_H

#define SIZE 5

void decimalToBinary(int n, int *binary, int size);
void twosComplement(int *binary, int size);
void printBinary(int *binary, int size);
void addTwoBinaries(int *binary1, int *binary2, int size);

#endif
```

Source File: binary_operations.c

```
#include "binary_operations.h"
#include <stdio.h>
#include <stdlib.h>

void decimalToBinary(int n, int *binary, int size) {
    if (n < 0) {
        n = abs(n);
    }

    for (int i = size - 1; i >= 0; i--) {
        binary[i] = n % 2;
        n = n / 2;
    }
}

void twosComplement(int *binary, int size) {
    int carry = 1;

    for (int i = 0; i < size; i++) {
        binary[i] = binary[i] == 0 ? 1 : 0;
    }

    for (int i = size - 1; i >= 0; i--) {
        binary[i] = binary[i] + carry;
        if (binary[i] == 2) {
            binary[i] = 0;
            carry = 1;
        } else {
            carry = 0;
        }
    }
}

void printBinary(int *binary, int size) {
    for (int i = 0; i < size; i++) {
        printf("%d", binary[i]);
    }
```

```
    printf("\n");
}

void addTwoBinaries(int *binary1, int *binary2, int size) {
    int carry = 0;
    for (int i = size - 1; i >= 0; i--) {
        binary1[i] = binary1[i] + binary2[i] + carry;
        if (binary1[i] == 2) {
            binary1[i] = 0;
            carry = 1;
        } else if (binary1[i] == 3) {
            binary1[i] = 1;
            carry = 1;
        } else {
            carry = 0;
        }
    }
}
```

Source File: `addition.c`

```
#include <stdio.h>
#include "binary_operations.h"

int main() {
    int a1, b1, a[SIZE], b[SIZE];

    printf("Enter the first number: ");
    scanf("%d", &a1);
    printf("Enter the second number: ");
    scanf("%d", &b1);
    printf("\n");

    decimalToBinary(a1, a, SIZE);
    if (a1 < 0) {
        twosComplement(a, SIZE);
    }

    printf("First number in binary: ");
    printBinary(a, SIZE);

    decimalToBinary(b1, b, SIZE);
    if (b1 < 0) {
        twosComplement(b, SIZE);
    }

    printf("Second number in binary: ");
    printBinary(b, SIZE);
    printf("\n");

    int sum[SIZE];
    for (int i = 0; i < SIZE; i++) {
```

```
        sum[i] = a[i];
    }
    addTwoBinaries(sum, b, SIZE);

    printf("Sum of the given two numbers: ");
    printBinary(sum, SIZE);

    return 0;
}
```

Source File: `subtraction.c`

```
#include <stdio.h>
#include "binary_operations.h"

int main() {
    int a1, b1, a[SIZE], b[SIZE];

    printf("Enter the first number: ");
    scanf("%d", &a1);
    printf("Enter the second number: ");
    scanf("%d", &b1);
    printf("\n");

    decimalToBinary(a1, a, SIZE);
    if (a1 < 0) {
        twosComplement(a, SIZE);
    }

    printf("First number in binary: ");
    printBinary(a, SIZE);

    decimalToBinary(b1, b, SIZE);
    if (b1 < 0) {
        twosComplement(b, SIZE);
    }

    printf("Second number in binary: ");
    printBinary(b, SIZE);
    printf("\n");

    // To subtract b from a, we add a to the two's complement of b
    twosComplement(b, SIZE);
    int diff[SIZE];
    for (int i = 0; i < SIZE; i++) {
        diff[i] = a[i];
    }
    addTwoBinaries(diff, b, SIZE);

    printf("Difference of the given two numbers: ");
    printBinary(diff, SIZE);
```

```
    return 0;
}
```

Sample Input/Output for Addition

- **Input:** 7 and -3
- **Output:**

```
Enter the first number: 7
Enter the second number: 3

First number in binary: 00111
Second number in binary: 00011

Sum of the given two numbers: 01010
```

Sample Input/Output for Subtraction

- **Input:** 7 and -3
- **Output:**

```
Enter the first number: 7
Enter the second number: 3

First number in binary: 00111
Second number in binary: 00011

Difference of the given two numbers: 00100
```

Discussion

The programs successfully convert decimal numbers to binary, perform addition and subtraction using two's complement for negative numbers, and display the results in binary format. The approach ensures correct handling of negative numbers by using two's complement, which simplifies binary arithmetic operations.

Conclusion

The implemented C programs correctly perform binary addition and subtraction of two decimal numbers, including handling negative numbers using two's complement representation. This lab demonstrates the process of binary arithmetic operations and the importance of two's complement in representing negative numbers.