**COMPUTER ORGANIZATION AND ARCHITECTURE**

BATCH-07

SECTION-2I-CSE

S.KIRANMAI-221FA04624

M.SUPRIYA-221FA04611

A.REVANTH-221FA04628

A.NIKILESH-221FA04644

**ABSTRACT:**

This comparison study explores different methods of comparing two signed binary numbers represented in signed-2's complement form. The comparison is conducted using three different techniques: signed-magnitude representation, pairwise bit comparison from left to right, and implementing a flowchart algorithm. Each method is analyzed in terms of its effectiveness and efficiency in determining the relationship between the two numbers.

**INTRODUCTION:**

In this study, we'll explore three different ways to compare these special numbers. First, we'll use something called signed-magnitude representation, where we look at the signs of the numbers first and then compare their magnitudes separately. Then, we'll try a method where we compare pairs of bits from left to right, just like how you might compare numbers by looking at each digit. Finally, we'll dive into a flowchart algorithm, which is like following a step-by-step recipe to find the answer.

By understanding and applying these different techniques, we'll see which one works best for comparing signed-2's complement numbers. So, let's dive in and unravel the mystery of comparing these intriguing binary numbers

**QUESTION:**

You have to do comparison of two signed binary numbers when negative numbers are in signed-2's

complement representation.

a) Use signed-magnitude representation to find a solution to the given problem. (Understand) (3)

b) Obtain the solution by comparing pairs of bits from left to right. (Apply) (3)

c). Implement the flowchart to obtain the solution using signed-2's complement numbers. (Analyze) (4)

.

**SOLUTION:**

SIGN-MAGNITUDE REPRESENTATION

* To represent negative and positive integers , treat the most significant (leftmost) bit in the word as a sign bit
* If the sign bit , MSB=0,the number is positive;
* If the sign bit , MSB=1,the number is negative.
* **Example:**
* +18=00010010
* -18=11101110

**A)**

Using signed-magnitude representation:

* Binary number is divided in to two parts.

**Sign-Bit:**

* If the left most bit is 0 then the number is positive.
* If the left most bit is 1 then the number is negative.

**Magnitude:**

The remaining bits of binary represents the absolute value of the number.

**Comparision:**

* If the signs of two binary numbers are different, then the number with sign 0 is> sign 1
* If the signs are same , compare the magnitudes.

**Example:**

A=>(1101)2

B=>(0010)2

Sign for A=1

Sign for B=0

Both signs are diffrent number with sign 0>sign 1

Therefore : B>A

When signs are same:

A=>(1101)2

B=>(1010)2

Sign for A=1

Sign for B=1

same sign values so we need to check magnitude values

magnitude value for A

A=1101

Therefore magnitude value for A=13;

Magnitude value for B

B=1010

Therefore magnitude for B=10;

So A>B

**B)Comparing from left to right**:

Example:

A=>(1100)2

B=>(1011)2

Sign for A=1

Sign for B=1

Here both signs are same,so we need to next bit.

Sign for A=1

Sign for B=0

Here both signs are different

So sign bit is B>A

C)**C LANGUAGE CODE:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

void main(){

char a[16];

int i, j, k, len;

printf ("Enter a binary number: ");

gets (a);

len= strlen(a);

for (k=0; a[k]!='\0';k++)

{

if (a[k] != '0' && a[k]!='1’)

{

printf ("\nIncorrect Binary format...this program will quit");

exit(0);

}

}

for (i=len-1; a[i]!='1'; i--)

for (j=i-1; j>=0; j--)

{ if (a[j]=='1')

a[j]= '0';

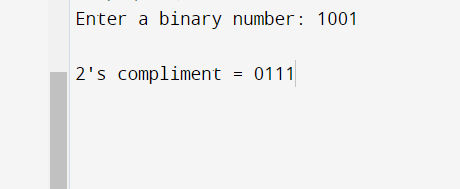
else a[j]= '1';

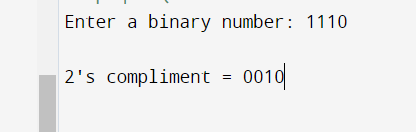
}

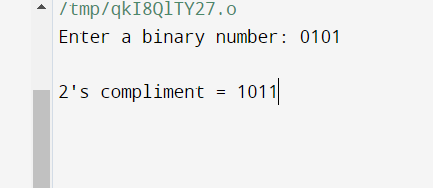
printf ("\n2's compliment = %s", a);

}

**OUTPUT:**





****

**CONCLUSION:**

In summary, we've examined three approaches for comparing signed binary numbers in signed-2's complement representation: signed-magnitude, pairwise bit comparison, and flowchart implementation.

Signed-magnitude representation allows for separate analysis of sign and magnitude but adds complexity in handling sign bits.

Pairwise bit comparison, by contrast, proves straightforward and efficient, starting from the most significant bits and progressing to the least significant. This method offers simplicity and clarity.

Flowchart implementation provides a structured algorithmic approach but may be more complex, suited for systematic handling of various cases.

**References:**

* <https://www.geeksforgeeks.org/register-transfer-language-rtl/>
* Textbook:

Essentials of Computer Organization and Architecture, 5th Edition by Linda Null, Julia Lobur.