Compute the integer absolute value (abs) without branching

Difficulty Level : HardLast Updated : 25 Mar, 2021

We need not to do anything if a number is positive. We want to change only negative numbers. Since negative numbers are stored in 2's complement form, to get the absolute value of a negative number we have to toggle bits of the number and add 1 to the result.
For example -2 in a 8 bit system is stored as follows 1 1 1 1 1 1 1 0 where leftmost bit is the sign bit. To get the absolute value of a negative number, we have to toggle all bits and add 1 to the toggled number i.e, 0 0 0 0 0 0 0 1 + 1 will give the absolute value of 1 1 1 1 1 1 1 0. Also remember, we need to do these operations only if the number is negative (sign bit is set).

Method 1

1) Set the mask as right shift of integer by 31 (assuming integers are stored using 32 bits).

```
mask = n>>31
```

2) For negative numbers, above step sets mask as 1 1 1 1 1 1 1 1 and 0 0 0 0 0 0 0 0 for positive numbers. Add the mask to the given number.

```
mask + n
```

3) XOR of mask +n and mask gives the absolute value.

```
(mask + n)^mask
```

Implementation:

C++

```
#include <bits/stdc++.h>
using namespace std;
#define CHARBIT 8

/* This function will return absolute value of n*/
unsigned int getAbs(int n)
{
    int const mask = n >> (sizeof(int) * CHARBIT - 1);
    return ((n + mask) ^ mask);
}

/* Driver program to test above function */
int main()
{
    int n = -6;
    cout << "Absoute value of " << n << " is " << getAbs(n);
    return 0;
}

// This code is contributed by rathbhupendra
```

C

```c
#include <stdio.h>
#define CHAR_BIT 8

/* This function will return absolute value of n*/
unsigned int getAbs(int n)
{
    int const mask = n >> (sizeof(int) * CHAR_BIT - 1);
    return ((n + mask) ^ mask);
}

/* Driver program to test above function */
int main()
{
    int n = -6;
    printf("Absoute value of %d is %u", n, getAbs(n));

    getchar();
    return 0;
}
```

Java

```java
// Java implementation of above approach
class GFG {

    static final int CHAR_BIT = 8;
    static final int SIZE_INT = 8;

    /* This function will return absolute value of n*/
    static int getAbs(int n)
    {
        int mask = n >> (SIZE_INT * CHAR_BIT - 1);
        return ((n + mask) ^ mask);
    }
```

```
    /* Driver code */
    public static void main(String[] args)
    {
        int n = -6;
        System.out.print("Absoute value of " + n + " is " + g
    }
}

// This code is contributed by Rajput-Ji
```

Python3

```
# Python3 implementation of above approach
CHARBIT = 8;
SIZE_INT = 8;

# This function will return
# absolute value of n
def getAbs(n):
    mask = n >> (SIZE_INT * CHARBIT - 1);
    return ((n + mask) ^ mask);

# Driver Code
n = -6;
print("Absolute value of",n,"is",getAbs(n));

# This code is contributed by mits
```

C#

```
// C# implementation of above approach
using System;

lass GFG {

    static int CHAR_BIT = 8;
```

```
    static int SIZE_INT = 8;

    /* This function will return absolute value of n*/
    static int getAbs(int n)
    {
        int mask = n >> (SIZE_INT * CHAR_BIT - 1);
        return ((n + mask) ^ mask);
    }

    /* Driver code */
    static void Main()
    {
        int n = -6;
        Console.Write("Absoute value of " + n + " is " + getA
    }
}

// This code is contributed by mits
```

PHP

```
<?php
// PHP implementation of above approach
$CHARBIT = 8;
$SIZE_INT = 8;

// This function will return
// absolute value of n
function getAbs($n)
{
    global $SIZE_INT, $CHARBIT;
    $mask = $n >> ($SIZE_INT * $CHARBIT - 1);
    return (($n + $mask) ^ $mask);
}

// Driver Code
$n = -6;
echo "Absolute value of " . $n .
            " is " . getAbs($n);
```

```
// This code is contributed by mits
?>
```

Javascript

```
<script>
// javascript implementation of above approach
var CHAR_BIT = 8;
var SIZE_INT = 8;

    /* This function will return absolute value of n */
    function getAbs(n)
    {
        var mask = n >> (SIZE_INT * CHAR_BIT - 1);
        return ((n + mask) ^ mask);
    }

    /* Driver code */
        var n = -6;
        document.write("Absoute value of " + n + " is " + get

// This code is contributed by shikhasingrajput
</script>
```

Output:

```
Absolute value of -6 is 6
```

ethod 2:

) Set the mask as right shift of integer by 31 (assuming integers

are stored using 32 bits).

```
mask = n>>31
```

2) XOR the mask with number

```
mask ^ n
```

3) Subtract mask from result of step 2 and return the result.

```
(mask^n) - mask
```

Implementation:

C

```c
/* This function will return absolute value of n*/
unsigned int getAbs(int n)
{
    int const mask = n >> (sizeof(int) * CHAR_BIT - 1);
    return ((n ^ mask) - mask);
}
```

On machines where branching is expensive, the above expression can be faster than the obvious approach, r = (v < 0) ? (unsigned)v : v, even though the number of operations is the same.

Please see this for more details about the above two methods.