# Operations on Signed Numbers

Naika Sateesh Kumar
Department of Computer Science & Engineering
GITAM University
Hyderabad, India
Email: Sateesh1835@gmail.com

Muduganti Rathan Reddy
Department of Computer Science & Engineering
GITAM University
Hyderabad, India
Email: rathan.reddy.muduganti@gmail.com

Y. Srinivas
Department of Information Engineering
GITAM University
Hyderabad, India
Email: sri9229@gmail.com

J. Vijayasekhar
Department of Mathematics
GITAM University
Hyderabad, India
Email: vijayjaliparthi@gmail.com

V K Narla
Department of Mathematics
GITAM University
Hyderabad, India
Email: vknarla@gmail.com

*Abstract*- **Signed integers are normally represented using 2's complement representation. Addition and subtraction of signed numbers is done similar to that of unsigned numbers. However carry (or borrow) is simply ignored. Unlike unsigned number carry (or borrow) does not mean overflow or error. Doubling of a signed number can be done by shift left. However, halving of a signed number cannot be done by shift right. Hence special arithmetic instruction SAR (Shift arithmetic right) is needed.**
**We have defined an alternative representation for signed numbers. Here a positive number is represented by appended a zero (0) at right. Here a negative number is represented by inverting all bits in corresponding positive number. Two signed numbers are added by adding corresponding binary representation. After that carry is added to the result. Similarly two signed numbers are subtracted by subtracting corresponding binary representation. After that borrow is subtracted. Doubling and halving is done by ROL (Rotate left) and ROR (Rotate right) respectively.**

**Keywords-SAR, ROL, ROR, Overflow**

Description

Following are drawbacks of our system.
(A) Addition is done in two stages. In the first stage the numbers are added. In the second stage carry is added. Carry cannot be ignored as in 2's complement representation.
(B) Same holds for subtraction.
(C) When an odd number is halved then error results. In 2's complement representation approximate answer appears.
The advantage of our system is that entire arithmetic can be carried using ordinary logical instructions. No special instruction is needed. In 2's complement representation a special instruction SAR is needed. Thisinstruction is not used for any other purpose.
1. Representation

Method to represent the signed numbers in 2's complement form

I. Find binary representation of corresponding unsigned number.

II. A positive number is represented by putting 0 at the beginning.
III. A negative number is represented by putting 1 at the beginning. Moreover all bits at the left of right most 1 are inverted (Complemented).

Method to represent the signed numbers in present scheme form

I. Find binary representation of the corresponding unsigned number.
II. A positive number is represented by putting 0 at the end.
III. A negative number is represented by putting 1 at the end. All other bits are inverted.
If two numbers of same magnitude with different signs are to be represented then they are

Signed Number Binary Representation

| Unsigned Numbers | Binary Representation | 2's Complement Values | | Rotation Scheme | |
|---|---|---|---|---|---|
| | | **Positive** | **Negative** | **Positive** | **Negative** |
| 12 | 1100 | +12=01100 | -12=10100 | +12=11000 | -12=00111 |
| 15 | 1111 | +15=01111 | -15=10001 | +15=11110 | -15=00001 |
| 17 | 10001 | +17=010001 | -17=101111 | +17=100010 | -17=011101 |
| 19 | 10011 | +19=010011 | -19=101101 | +19=100110 | -19=011001 |
| 50 | 110010 | +50=0110010 | -50=1001110 | +50=1100100 | -50=0011011 |
| 37 | 100101 | +37=0100101 | -37=1011011 | +37=1001010 | -37=0110101 |

Representation of Signed number from binary number

| Binary Number | 2's complement form | Rotation Scheme |
|---|---|---|
| 0101100 | The LSB is 0. Hence the given number is positive. No changes in rest of bits.The absolute value is + 44 | The RSB is 0. So, the number is positive. No changes in rest of bits. Drop the right most bit. The absolute value is +22 |
| 1001101 | The LSB is 1. Hence the given number is negative, socomplement all bits before last one is 0110011. The absolute value of 0110011 is -51. | The RSB is 1. Hence the given number is negative. Hence invert all bits 0110010. Drop right most bit 011001. The absolute value is -25 |

Addition

The procedure for addition of two numbers is same in both the schemes, only difference is method of handling carry.

| NUMBERS | 2'S Complement Method | Rotation Scheme |
|---|---|---|
| +12 and -15 | +12=01100<br>-15=10001<br>Addition : 11101<br>Value is -3(10011) | +12=11000<br>-15=00001<br>Addition:11001(drop last 1)<br>It is -3(00111) |
| -17and +19 | -17=101111<br>+19=010011<br>Addition: 000010(Carry ignored)<br>It is +2 | -17=011101<br>+19=100110<br>Addition: 1000011(ADD ignored)<br>1<br>000100(drop last 1)<br>It is +2 |

Expansion and Compression

When two numbers of different size are added, the size of the smaller number (in magnitude) is expanded, in order to make them into same size.Size of an unsigned number is increased by the placing 0's are put in the beginning (LSB).

- In 2's complement scheme the size of a number is expanded by putting the copy of the MSB before MSB. +15 is 01111. It can be also written as 0001111. Similarly for -15 is 10001, it can also be written as 1110001.
- In rotation scheme the size of a number is expanded by placing the copy of LSB at MSB. +15 is 11110. It can also be written as 0011110. Similarly for −15 is 00001. It can also be written as 1100001.

Following table gives the representation of increasing unsigned numbers size

| Unsigned Numbers | Binary Representation | 2's Complement Values | | Rotation Scheme | |
|---|---|---|---|---|---|
| | | Positive | Negative | Positive | Negative |
| 12 | 001100 | +12=0001100 | -12=110100 | +12=011000 | -12=100111 |
| 15 | 001111 | +15=0001111 | -15=110001 | +15=011110 | -15=100001 |
| 17 | 010001 | +17=0010001 | -17=1101111 | +17=0100010 | -17=1011101 |
| 19 | 010011 | +19=0010011 | -19=1101101 | +19=0100110 | -19=1011001 |
| 50 | 0110010 | +50=00110010 | -50=11001110 | +50=01100100 | -50=10011011 |
| 37 | 0100101 | +37=00100101 | -37=11011011 | +37=01001010 | -37=10110101 |

Addition of two numbers of different sizes can be illustrated as:

| Numbers | 2'sComplementScheme | Rotation Scheme |
|---|---|---|
| +12and+2 | +12is01100and+2is010.Sizeof+2 isincreased.<br>+12= 01100<br>+2=00010<br>Add: 01110<br>The value is +14 | +12is 11000and +2is 100. Size of +2is increased.<br>+12=11000<br>+2 =00100<br>Add: 11100<br>The value is+14 |
| +19and-15 | +19=010011<br>-15=1 10001<br>Add: 0 0 0 1 0 0<br>The value is+4<br>(carry Ignored) | +19=100110<br>-15=100001(expand)<br>Add: 000111<br>1<br>001000<br>The value is+4 |

In some cases the size of result may also get reduced.Hencesizecompressiontakes place.Itisdonebyremovingmostsignificantbits.

- In2'scomplementrepresentationifMSBandsecondmostsignificantbitsaresamethenMSBis removed.
- InRotational scheme the most significant bit can be removed when MSB=LSB

| | | |
|---|---|---|
| +19and-15 | +19=010011<br>-15=110001<br>Add: 000100<br>Eliminate MSB values, since three bits are repeated.<br>The value is +4. (Carry Ignored) | +19=100110<br>-15=100001(expand)<br>Add: 000111<br>1<br>001000<br>Eliminate MSB values, since MSB and LSB are same.<br>The value is +4. (Carry Ignored) |

Subtraction:

Subtraction will be processed similar to addition

| NUMBERS | 2'S Complement Method | Rotation Scheme |
|---|---|---|
| +12 and +2 | +12=01100<br>+2=00010<br>subtraction : 01010<br>The value is +10(01010) | +12=11000<br>+2=00100<br>subtraction: 10100 (drop last 1)<br>The value is +10(1010) |
| -12 and -15 | -12=10100<br>-15=10001<br>subtraction:00011(Carry ignored)<br>The value is +3 | -12=00111<br>-15=00001<br>subtraction: 00110(drop last 1)<br>The value is +3 |

In the above cases there is a situation, where the number of bits in the resultant are more than the number if bits in the given values on which operations are to be performed. This can be solved in 2's complement method by placing MSB in MSB position and in Rotation scheme placing LSB in MSB position.

| Numbers | 2'sComplementScheme | Rotation Scheme |
|---|---|---|
| +12and+13 | +12=01100<br>+13=01101<br>Add: 11001<br>The value is-9 | +12=11000<br>+13 =11010<br>Add: 10010<br>The value is +9 |
| After incrementing the size of bits | | |
| +12and+13 | (placing MSB in MSB position)<br>+12=001100<br>+13=001101<br>Add: 011001<br>The valueis+25 | (placing LSB in MSB position)<br>+12=0 11000<br>+13 =011010<br>Add: 110010<br>The value is+25 |

Halving and Doubling

A signed number can be doubled and halved by using two methods.

a. 2's complement Method b. Rotation method

Doubled of signed number in 2's complement method can be done by SHL (shift left) operation, where every bitis shifted left. The left most bit is removed. Zero (0) is appended to the right.

Halving of signed number in 2's complement method can be done by SAR (shift arithmetic right) operation, where every bit is shifted right. The right most bit is removed. Place the left most bit in LSB position. The only difference between SHR and SAR is replacement of zero and LSB bits respectively.

Illusion

Let us consider a word UNIVERSITY

(SL)

UNIVERSITYNIVERSITY0 ⟶

(SAR)

↓

UUNIVERSIT

In rotation schema the signed numbers can be doubled by Rotate Left (ROL) operation and the number can be halved by Rotate Right (ROR). In rotate left operation every bit is shifted left and the left most bit (MSB) is transferred to the right (LSB).In case of ROR, every bit is shifted to the right and the right most bit (MSB) is transferred to the left (LSB).

Illusion

Let us consider a word UNIVERSITY

(ROR)

UNIVERSITY YUNIVERSIT ⟶

(ROL)

↓

NIVERSITYU

As the number is doubled the size of the number increases, to avoid overflow the size is increase by 1 in the beginning. Similarly after halving a number the size decreases. Hence MSB can be removed.

| Unsigned Numbers | Binary Representation | 2's Complement Values | | Rotation Scheme | |
|---|---|---|---|---|---|
| | | Shift Left | | Rotate Left | |
| 12 | 1100 | +12=001100 | +24=011000 | +12=011000 | +24=110000 |
| 15 | 1111 | +15=001111 | +30=011110 | +15=011110 | +30=111100 |
| 17 | 10001 | +17=0010001 | +34=0100010 | +17=0100010 | +34=1000100 |
| 19 | 10011 | +19=0010011 | +38=00100110 | +19=0100110 | +38=1001100 |
| 50 | 110010 | +50=00110010 | +100=001100100 | +50=01100100 | +100=11001000 |
| Unsigned Numbers | Binary Representation | 2's Complement Values | | Rotation Scheme | |
| | | Arithmetic Shift Right | | Rotate Right | |
| 12 | 1100 | +12=01100 | +6=00110 | +12=11000 | +6=01100 |
| 15 | 1111 | +15=01111 | +7=00111 | +15=11110 | -7=01111 |
| 17 | 10001 | +17=010001 | +8=001000 | +17=100010 | -8=010001 |
| 19 | 10011 | +19=010011 | +9=001001 | +19=100110 | -9=010011 |
| 50 | 110010 | +50=0110010 | +25=0011001 | +50=1100100 | +25=0110010 |

From the above table we can conclude that halving and doubling of an even number produces the correct result.The doubling of odd numbers gives required output but halving produces the error in result in Rotation Method, where as in the 2's complement produces nearest integer value.

## Conclusions

The representation of binary numbers can be done by 2's complement method, placing zero at MSB position for positive numbers and negatives are represented by placing one at MSB position and complementing all the bits to the left of last one's occurrence. Dividing and halving of numbers can be shown in both 2's complement method and rotation method. In 2's complement method doubling is done by shift left and halving is done by Arithmetic shift right. Halving through this method produces nearest integer value. In rotation method doubling is done by Rotation left and halving is done by Rotation right. Halving through this method produces error.

## References


[1] Ivan Flores, the Logic of Computer Arithmetic, Prentice-Hall (1963)

[2] Ytha Yu and Charles marut, Assembly language programming and organization of IBM PC, Mc Graw-Hill, 1992

[3] Peter Norton and John Socha, Peter Norton's Assembly language book for the IBMPC, Prentice-Hall, 1993.

[4] John F. Wakerly, Digital Design Principles & Practices, Prentice Hall, 3rd edition 2000, page 47

[5] Israel Koren, Computer Arithmetic Algorithms, A.K. Peters (2002), ISBN 1-56881-160-8

[6] David J. Lilja and Sachin S. Sapatnekar, Designing Digital Computer Systems with Verilog, Cambridge University Press, 2005online

[7] Digital Design and Computer Architecture" by David Harris, David Money Harris, Sarah L. Harris. 2007. Page 18. [8] Digital Logic and Computer Design,M. Morris Mano.

[8] J. Vijayasekhar, Y. Srinivas, N. Vamsi Krishna,Signed Numbers Conversions, Indian Journal of Computer science and Engineering, Vol 2, Issue1, (2011)43-47.