

Binary Addition in C (2C)

Copyright (c) 2011-2013 Young W. Lim.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice and Octave.

Make a binary string

```
//.....  
// n : bit width  
// 1 << n : shift left by n bits (2^n)  
// mask : n bit filled with all one's (2^n -1)  
//.....  
#define nn 3  
int n = nn;  
int mask = (1 << nn) - 1;
```

```
// make a binary string from the given num  
void bstr(int num, int nbit, char * s)  
{  
    int i, j;  
  
    for (i=0; i<nbit; ++i) { // i-th bit  
        j = nbit - i - 1; // j-th position in s[]  
        if (num & (1 << i)) s[j] = '1'; // the i-th bit of  
num  
        else s[j] = '0';  
    }  
    s[nbit] = '\0';  
}
```

Get a 2's complementary number

```
// make a 2's complementary number from
unsigned number.
int snum(int x, int nbit)
{
    int num;

    if (x < (1 << (nbit-1))) num = x;
    else num = x - (1 << nbit);

    return (num);
}
```

List 3-bit binary numbers

ui =	0	i =	+0	000
ui =	1	i =	+1	001
ui =	2	i =	+2	010
ui =	3	i =	+3	011
ui =	4	i =	-4	100
ui =	5	i =	-3	101
ui =	6	i =	-2	110
ui =	7	i =	-1	111

```
void main()
{
    unsigned char ui, uj;
    signed char i, j;
    char s[10];

    int x, y, z;

    for (x=0; x<(1<<n); ++x) {
        // ui: unsigned number
        ui = x;

        // i: signed 2's complement number
        i = snum(x, n);

        // s: binary string of the given number i
        bstr(i, n, s);

        printf("ui = %4d i= %+4d %4s \n", mask & ui, i, s);
    }
}
```

Addition Table for 3-bit binary numbers

```
for (x=0; x<(1<<n); ++x) {
  for (y=0; y<(1<<n); ++y) {
    ui = x;
    uj = y;
    z = mask & (ui + uj);
    bstr(z, n+1, s);

    printf(" %4s(%+1d)", s, snum(z, n));
  }
  printf("\n");
}

}
```

0000(+0)	0001(+1)	0010(+2)	0011(+3)	0100(-4)	0101(-3)	0110(-2)	0111(-1)
0001(+1)	0010(+2)	0011(+3)	<u>0100(-4)</u>	0101(-3)	0110(-2)	0111(-1)	0000(+0)
0010(+2)	0011(+3)	<u>0100(-4)</u>	<u>0101(-3)</u>	0110(-2)	0111(-1)	0000(+0)	0001(+1)
0011(+3)	<u>0100(-4)</u>	<u>0101(-3)</u>	<u>0110(-2)</u>	0111(-1)	0000(+0)	0001(+1)	0010(+2)
<u>0100(-4)</u>	<u>0101(-3)</u>	<u>0110(-2)</u>	<u>0111(-1)</u>	0000(+0)	0001(+1)	0010(+2)	0011(+3)
0101(-3)	0110(-2)	0111(-1)	0000(+0)	<u>0001(+1)</u>	<u>0010(+2)</u>	<u>0011(+3)</u>	0100(-4)
0110(-2)	0111(-1)	0000(+0)	0001(+1)	<u>0010(+2)</u>	<u>0011(+3)</u>	0100(-4)	0101(-3)
0111(-1)	0000(+0)	0001(+1)	0010(+2)	<u>0011(+3)</u>	0100(-4)	0101(-3)	0110(-2)

Laplace Equation

References

- [1] <http://en.wikipedia.org/>
- [2] <http://planetmath.org/>
- [3] M.L. Boas, "Mathematical Methods in the Physical Sciences"