Amol Gulhane
Robolab Technologies,
Pune - 411005 (India)
#Tel: +91 879 310 2996
E-mail: amol.robolab@gmail.com

Program for finding whether a number is power of 2 or not.

```
if((n) & (n-1))
  {n is a power of 2}
```

Program for finding whether a number is even or odd.

```
if((n) & (1))
  {odd}
else
  {even}
```

Write a function to swap even bits with consecutive odd bits in a number.
e.g. b0 swapped with b1, b2 swapped with b3 and so on.

```
unsigned int swap_bits(unsigned int num)
{
   return (num>>1 & 0x55555555) | (num<<1 & 0xAAAAAAAA);
}
```

Write a function to swap odd bits in a number.
e.g. b1 swapped with b3, b5 swapped with b7 and so on.

```
unsigned int swap_odd_bits(unsigned int num)
{
  return (num>>2 & 0x22222222) |
         (num<<2 & 0x88888888) |
         ( num   & 0x55555555) ;
}
```

Write a function to swap even bits in a number.
e.g. b0 swapped with b2, b4 swapped with b6 and so on.

```
unsigned int swap_even_bits(unsigned int num)
{
  return (num>>2 & 0x11111111) |
         (num<<2 & 0x44444444) |
         ( num   & 0xAAAAAAAA) ;
}
```

Write a function to find out the number of 1s in a number.

```
unsigned int num_of_ones(unsigned int num)
{
  unsigned int count = 0;
  while (num != 0) {
    num = (num) & (num-1);
    count++;
  }
  return count;
}
```

Write a function to check whether the number of 1s present in a number are even or odd.

Amol Gulhane
Robolab Technologies,
Pune - 411005 (India)
#Tel: +91 879 310 2996
E-mail: amol.robolab@gmail.com

```
enum {
  EVEN,
  ODD
} even_odd;

unsigned int check_even_odd_no_of_ones(unsigned int num)
{
  if(num_of_ones(num) & 1)
    return ODD;
  else
    return EVEN;
}
```

Write a function for finding the first lowest bit set in a number.

```
unsigned int first_lowest_bit_set(unsigned int num)
{
  unsigned int count = 0;

  while(num) {
    count++;
    if(num&1 == 1)
      break;
    num = num >> 1;
  }

  return count;
}
```

Write a function for finding the higest bit set in a number.

```
unsigned int first_highest_bit_set(unsigned int num)
{
  unsigned int count = 0;

  while(num) {
    count++;
    if(num&(1<<31) == 1)
      break;
    num = num << 1;
  }

  return count;
}
```

Write a function for reversing the bits in a number.

```
unsigned int bit_reverse(unsigned int n)
{
  unsigned int m = 0, i;
```

Amol Gulhane
Robolab Technologies,
Pune - 411005 (India)
#Tel: +91 879 310 2996
E-mail: amol.robolab@gmail.com

```
  for (i = 0; i < 32; i++) {
    m |= (n & 1) << (31-i);
    n >>= 1;
  }

  return m;
}
```

Write the code for extracting nth to mth bits, where n < m.

```
31                          0

 ------------------------

   |        |

   m        n


(num >> n) & ~(~0 << (m-n+1))
```

Write the code for toggling nth to mth bits, where n < m.

```
e.g.

   4 2

   | |

10101010 - num

11111111 ~0

11111100 (~0<<n)

00011111 (~0>>(31-m))


00011100 (~0<<n) & (~0>>(31-m))    --> mask


Solution: num ^ ((~0<<n) & (~0>>(31-m)))
```

Amol Gulhane
Robolab Technologies,
Pune - 411005 (India)
#Tel: +91 879 310 2996
E-mail: amol.robolab@gmail.com

Write the code for setting nth to mth bits, where n < m.

```
num | ((~0<<n) & (~0>>(31-m)))
```

Write the code for clearing nth to mth bits, where n < m.

```
num & ~((~0<<n) & (~0>>(31-m)))
```

Write a piece of code for sizeof() implementation.

```
#define sizeof(a) ((char *)(&a+1)-(char *)&a)
```

Explain about ffs and ffz implementations in ARM linux.

```
/*
 * On ARMv5 and above those functions can be implemented around
 * the clz instruction for much better code efficiency.
 */

static inline int fls(int x)
{
        int ret;

        if (__builtin_constant_p(x))
                return constant_fls(x);

        asm("clz\t%0, %1" : "=r" (ret) : "r" (x));
        ret = 32 - ret;
        return ret;
}

#define __fls(x) (fls(x) - 1)
#define ffs(x) ({ unsigned long __t = (x); fls(__t & -__t); })
#define __ffs(x) (ffs(x) - 1)
#define ffz(x) __ffs( ~(x) )
```

Explain about container_of() and offsetof() implementations.

Amol Gulhane
Robolab Technologies,
Pune - 411005 (India)
#Tel: +91 879 310 2996
E-mail: amol.robolab@gmail.com

```
/**
 * container_of - cast a member of a structure out to the
containing structure
 * @ptr:        the pointer to the member.
 * @type:       the type of the container struct this is
embedded in.
 * @member:     the name of the member within the struct.
 *
 */
#define container_of(ptr, type, member) ({
 \
        const typeof( ((type *)0)->member ) *__mptr = (ptr);
 \
        (type *)( (char *)__mptr - offsetof(type,member) );})


#define offsetof(TYPE, MEMBER) ((size_t) &((TYPE *)0)->MEMBER)
```

How to implement bit-wise operations without using bit-wise operators?

```
/* a ^ b */
c = 0;
for (x = 0; x <= 15; ++x) {
    c += c;
    if (a < 0) {
        if (b >= 0) {
            c += 1;
        }
    } else if (b < 0) {
        c += 1;
    }
    a += a;
    b += b;
}

/* a & b */
c = 0;
for (x = 0; x <= 15; ++x) {
    c += c;
```

Amol Gulhane
Robolab Technologies,
Pune - 411005 (India)
#Tel: +91 879 310 2996
E-mail: amol.robolab@gmail.com

```
    if (a < 0) {
        if (b < 0) {
            c += 1;
        }
    }
    a += a;
    b += b;
}

/* a | b */
c = 0;
for (x = 0; x <= 15; ++x) {
    c += c;
    if (a < 0) {
        c += 1;
    } else if (b < 0) {
        c += 1;
    }
    a += a;
    b += b;
}
```

Data Structures

Write a program for reversing a singly linked list?

```
struct node {
  int data;
  struct node *next;
}

void reverse(struct node **head)
{
  struct node *t0, *t1, *t2 = NULL;

  t0 = *head;

  while(t0 != NULL) {
    t1 = t2;
    t2 = t0;
    t0 = t0->next;
    t2->next = t1;
   }
```

Amol Gulhane
Robolab Technologies,
Pune - 411005 (India)
#Tel: +91 879 310 2996
E-mail: amol.robolab@gmail.com

```
  *head = t2;
}
```