Mask & Bitwise Operators

A mask is a bit pattern used with the bitwise operators to modify another bit pattern. Mask is used to set or clear individual bits, complement a particular bit, or check the status of a particular or all bits.

Create a mask:

1. Set a bit pattern explicitly.

Example:	unsigned mm = 0x0000; mm = 0x0005; /* First bit and 3rd bits are set */

2. Use the Left Shift operator.

Example 1:	unsigned mm = 0x0000;
	mm = 1 <<15; /* Sets 16th bit */

Example2 :	unsigned mm = 0x0000;
	mm = 1 << (n-1); /* Sets nth bit */

Preserve or filter particular bits:

1. Create a mask with 1 for those bits which you want to preserve or filter, and zero for all others.
2. Perform Bitwise AND operation (&) with the operands: the mask and the bit pattern you want to modify.

Example:	unsigned aa, bb, mm = 0x0000;
	
	aa = 0xF339; /* aa = 1111 0011 0011 1001 */
	mm = 0x0005; /* First bit and 3rd bit are set – 0000 0000 0000 0101 */
	bb = aa & mm; /* bb = 0000 0000 0000 0001 */

Set particular bits and leave other bits as they are:

1. Create a mask with 1 in the locations that you want to set to 1 and zeros in the other locations.
2. Perform Bitwise OR operation (|) with the operands: the mask and the bit pattern you want to modify.

Example:	unsigned aa, bb, mm = 0x0000;
	
	aa = 0xF338; /* aa = 1111 0011 0011 1000 */
	mm = 0x0005; /* First bit and 3rd bit are set – 0000 0000 0000 0101 */
	bb = aa | mm; /* bb = 1111 0011 0011 1101 */

Clear particular bits and leave other bits as they are:

Option 1

1. Create a mask with zeros for those bits which you want to clear, and 1 for all others.
2. Perform Bitwise AND operation (&) with the operands: the mask and the bit pattern you want to modify.

Example:	unsigned aa, bb, mm = 0x0000;
	
	aa = 0xF339; /* aa = 1111 0011 0011 1001 */
	mm = 0x3FFF; /* 15th and 16th bits are zero – 0011 1111 1111 1111 */
	bb = aa & mm; /* bb = 0011 0011 0011 1001 */

Option 2 – Commonly used

1. Create a mask with 1 in the locations that you want to clear and zeros in the other locations.
2. Use Bitwise Complement operator (~) to reverse all bits of the mask.
3. Perform Bitwise AND operation (&) with the operands: the mask and the bit pattern you want to modify.

Example:	unsigned aa, bb, mm = 0x0000;
	
	aa = 0xF339; /* aa = 1111 0011 0011 1001 */
	mm = 1 << 15; /* 16th bit is zero – 1000 0000 0000 0000 */
	bb = aa & (~mm); /* bb = 0111 0011 0011 1001 */

Complement particular bits and leave other bits as they are:

1. Create a mask with 1 in the locations that you want to complement and zeros in the other locations.
2. Perform Bitwise eXclusive OR operation (^) with the operands: the mask and the bit pattern you want to modify.

Example:

```
unsigned aa, bb, mm = 0x0000;

aa = 0xF338;              /* aa = 1111 0011 0011 1000 */
mm = 0x0005;              /* First bit and 3rd bit are set – 0000 0000 0000 0101 */
bb = aa ^ mm;             /* bb = 1111 0011 0011 1101 */
```

Check if any bit of a bit patter has changed:

1. Perform Bitwise eXclusive OR operation (^) with the operands: the original bit pattern and the new bit pattern of the same variable.
2. Determine if the resulting variable is greater than zero or equal to zero. If the variable is greater than zero, one or more bits have changed.

Example:

```
unsigned aa, bb;

aa = 0xF338;              /* aa = 1111 0011 0011 1000 */
scanf(%x", &bb);
bb = bb ^ aa;
if (bb > 0)               /* Has any bit changed? */
  printf("One or more bits have changed");
```

Check the status of the leftmost bit of a bit pattern (integer variable):

1. Create a mask with 1 in the leftmost location (Most Significant Bit) and zeros in the other locations.
2. Perform Bitwise AND operation (&) with the operands: the mask and the integer variable. Store the result in an integer variable.
3. Use the Right Shift operator (>>) to move the leftmost bit to the rightmost location in the newly created integer.
4. Determine if the new integer variable has value of 1 or zero.

Example:

```
unsigned aa, bb, mm=0x0000;

aa = 0xF338;              /* aa = 1111 0011 0011 1000 */
mm = 1<< 15;              /* Mask – 1000 0000 0000 0000 */
bb = aa & mm;             /* Bitwise AND to isolate 16th bit */
bb = bb >>15;             /*Right Shift to move the leftmost bit to the rightmost position */
if (bb == 1)      printf("The bit is set");
if (bb == 0)      printf("The bit is clear");
```

Determine and print the status of all bits of a bit pattern:

Example1:

Starts with MSB

```
unsigned aa, bb, mm=0x0000;
int i;

aa = 0xF338;              /* aa = 1111 0011 0011 1000 */
mm = 1<< 15;              /* Mask – 1000 0000 0000 0000 */
for (i = 16; i >= 1; i--)     /* Work with one bit at the time starting with 16th bit */
  {
  bb = aa & mm;           /* Bitwise AND to isolate a bit */
  bb >>= (i - 1) ;        /*Right Shift to move the bit to the rightmost position */
  printf("%u ", bb);
  mm >>= 1;               /* Prepare the mask to check the next bit */
  }
```

Example2:

Starts with LSB

```
unsigned int aa, bb, mm=0x0000;
int i;

aa = 0xF338;              /* aa = 1111 0011 0011 1000 */
mm = 1;                   /* Mask – 0000 0000 0000 0001 */

printf ("\nBits set to 1 are: ");
for (i = 1; i <= 16; i++)     /* Work with one bit at the time starting with the first bit */
  {
  bb = aa & mm;           /* Bitwise AND to isolate a bit */
  if (bb == 1)      printf("Bit %d \n ", i);
  aa >>= 1;               /*Right Shift to move the bit to the rightmost position */
  }
```

References

Tan, H.H., and T.B. D'Orazio. *C Programming for Engineering & Computer Science*. USA: WCB McGRaw-Hill. 1999. Print.