PRINT BITS IN AN INTEGER

PRINT BITS IN AN INTEGER

EXTRACT THE BITS ONE BY ONE
FROM THE INTEGER

GET WHETHER THEY ARE 0 OR 1 -
PRINT THAT OUT TO SCREEN

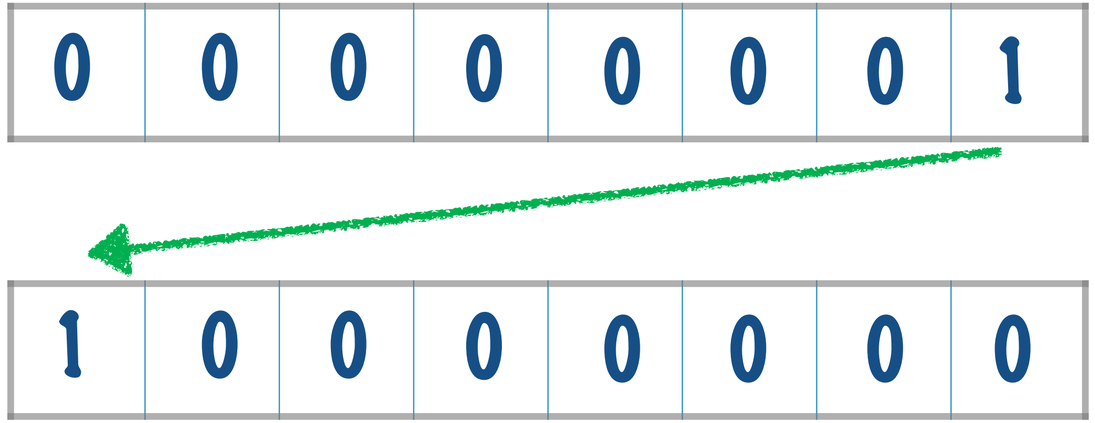
WE'VE ALREADY SEEN HOW TO GET
THE VALUE OF AN INTEGER AT THE
NTH POSITION

THE TRICK HERE IS TO START AT THE
LEFTMOST POSITION SO WE PRINT
THE BITS IN A LOGICAL ORDER

PRINT BITS IN AN INTEGER

HOW DO WE SET THE LEFT MOST BIT OF AN INTEGER TO 1?

START WITH 1



SHIFT IT LEFT SIZEOF(INT) * 8 - 1 TIMES

SIZEOF WILL RETURN THE SIZE OF AN INTEGER IN BYTES

MULTIPLY BY 8 TO GET THE SIZE IN BITS

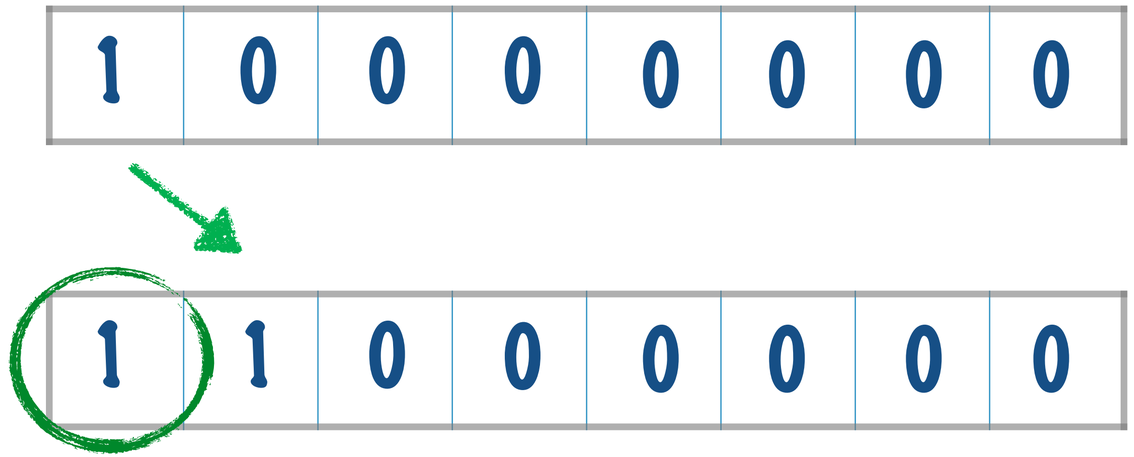
SUBTRACT 1 TO ACCOUNT FOR THE RIGHT MOST POSITION BEING AT INDEX 0

PRINT BITS IN AN INTEGER

AFTER WE GET THE 1-BIT IN THE LEFT MOST POSITION

IT SHOULD BE SHIFTED RIGHT

ONE SUBTLE DETAIL IN SHIFTING RIGHT



IF THE LEFT MOST BIT IS 1, THE FILL BIT TENDS TO BE 1 WHEN SHIFTED RIGHT

THIS IS BECAUSE RIGHT SHIFT PRESERVES THE SIGN OF THE INTEGER - A 1 IN THE RIGHT MOST POSITION IS A NEGATIVE NUMBER

USE AN UNSIGNED INTEGER AS A CHECK BIT TO AVOID THIS ISSUE!

PRINT BITS IN AN INTEGER

```c
void print_bits(int num) {
  unsigned int check_bit = 1 << (sizeof(int) * 8 - 1);

  while (check_bit != 0) {
    int result = num & check_bit;
    if (result == check_bit) {
      printf("%d ", 1);
    } else {
      printf("%d ", 0);
    }

    check_bit = check_bit >> 1;
  }

  printf("\n");
}
```

THE RIGHT MOST POSITION IN THE CHECK BIT SHOULD BE 1 - NOTE THE UNSIGNED INT!

ONCE THE RIGHT MOST 1 IS MOVED OUT TO THE LEFT AND IS SHIFTED OUT THEN CHECK_BIT WILL BE 0

MOVE THE 1 TO THE RIGHT

COUNT THE NUMBER OF 1 BITS

COUNT THE NUMBER OF 1 BITS

THE SIMPLE WAY IS VERY SIMILAR TO THE PRINT BITS EXAMPLE

THIS METHOD INVOLVES CHECKING EVERY BIT IN AN INTEGER

THERE IS ANOTHER, MORE OPTIMIZED TECHNIQUE

CHECK EACH BIT, INCREMENT A COUNT IF THE BIT IS 1

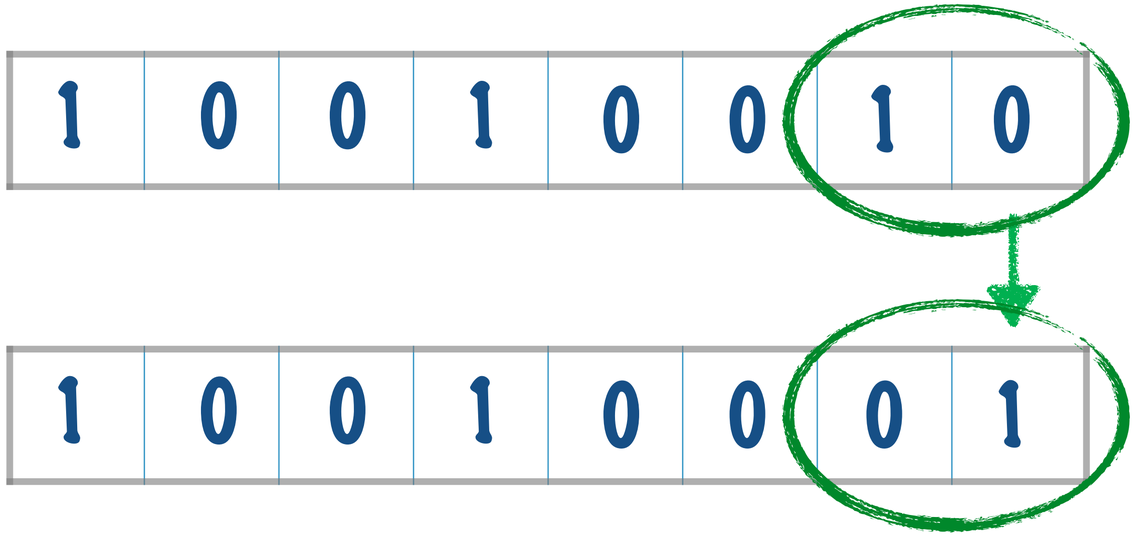
IT'S COMPLEXITY IS O(NUMBER OF BITS)

WITH A COMPLEXITY OF O(NUMBER OF 1S)

COUNT THE NUMBER OF 1 BITS

CONSIDER A NUMBER REPRESENTED IN BINARY FORM

NOW IF YOU SUBTRACT 1 FROM THIS NUMBER THIS IS THE RESULT

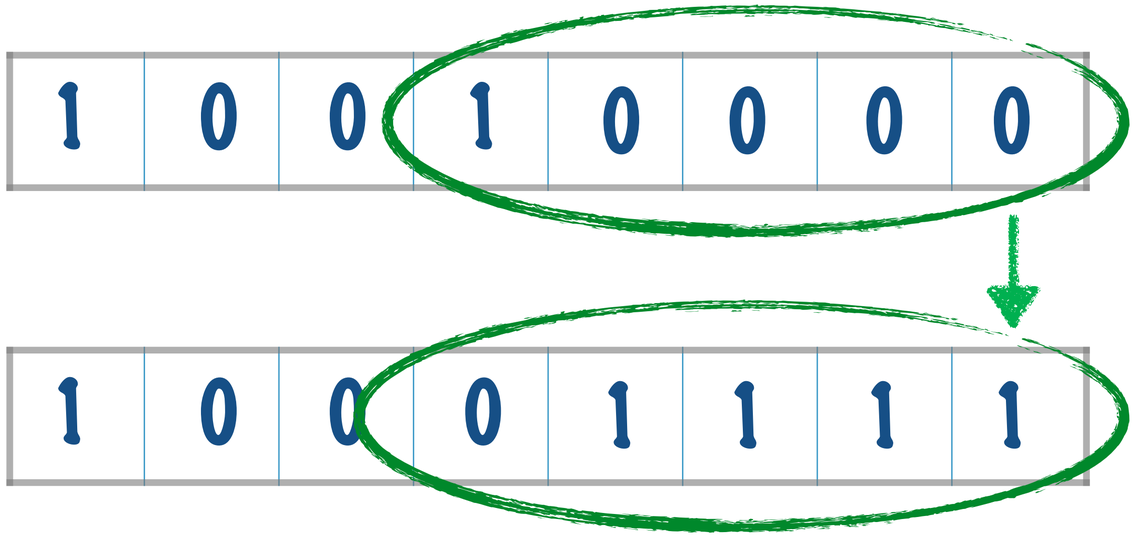


ALL THE BITS FROM THE RIGHT UP TO THE VERY FIRST 1 ARE TOGGLED

COUNT THE NUMBER OF 1 BITS

CONSIDER ANOTHER EXAMPLE

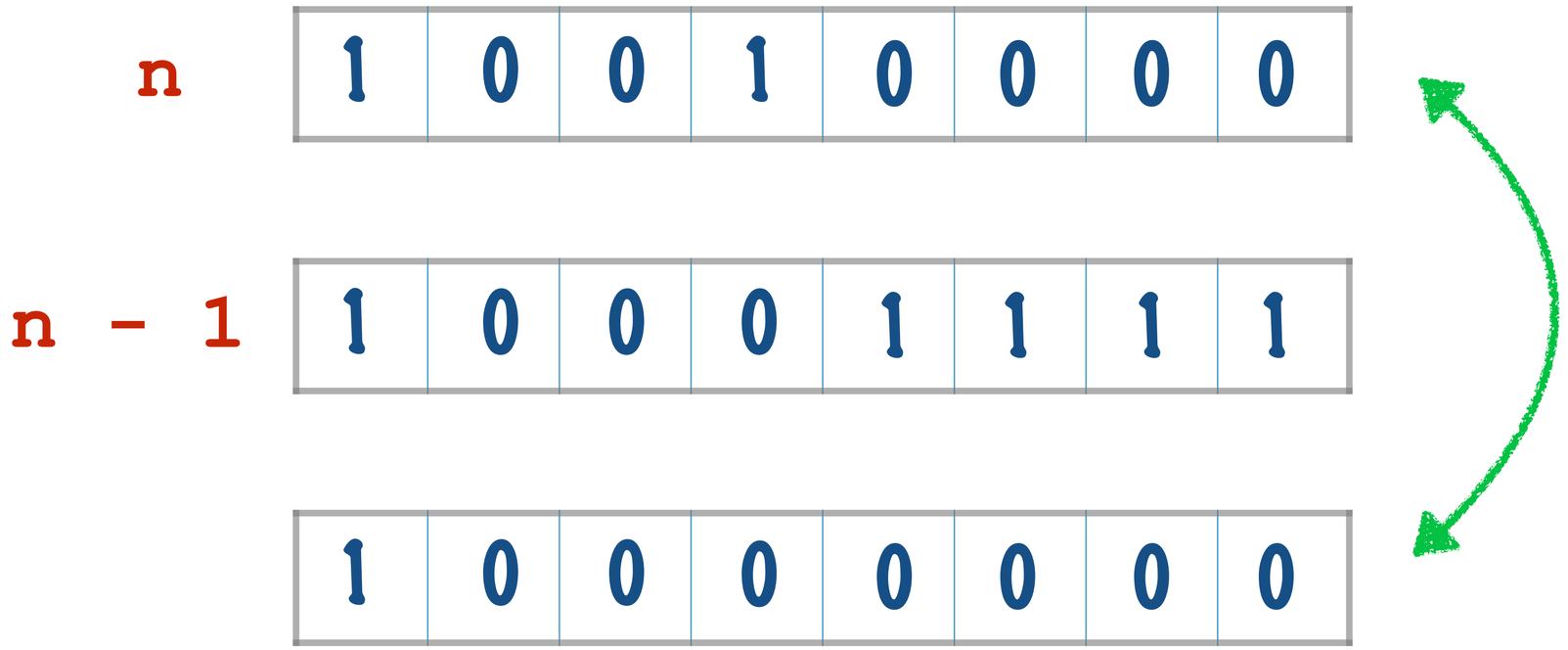
SUBTRACT 1



ALL THE BITS FROM THE RIGHT UP TO THE VERY FIRST 1 ARE TOGGLED

COUNT THE NUMBER OF 1 BITS

NOW IF WE BITWISE AND THESE TWO `n & (n -1)`



THE ORIGINAL NUMBER HAD TWO 1-BITS

THE RESULT HAS JUST ONE

COUNT THE NUMBER OF 1 BITS

CONTINUE THE BITWISE AND TILL THE NUMBER IS ALL ZEROES

A COUNTER CAN KEEP TRACK OF HOW OFTEN THIS OPERATION IS PERFORMED

THE COUNT WILL GIVE YOU THE NUMBER OF 1-BITS IN THE INTEGER

WITH A COMPLEXITY OF
O(NUMBER OF 1S)

COUNT THE NUMBER OF 1 BITS

```
int count_1s_optimized(int num) {
  int count = 0;
  while (num != 0) {
    num = num & (num - 1);
    count++;
  }

  return count;
}
```

INITIALIZE THE COUNT TO TRACK THE NUMBER OF 1-BITS

CONTINUE THE BITWISE AND WITH NUM - 1

THE COUNT FINALLY HOLDS THE NUMBER OF 1-BITS IN THIS NUMBER