Recitation 3

Computer Architecture (section 1)

Number Systems

- A standard way to represent values.
  - You are all familiar with the *decimal number system.*
  - More importantly, we will discuss *positional number systems.*
- *Base* of a number system refers to the number of symbols.

$2{,}839{,}239{,}794_{10}$

110100010111_{2}

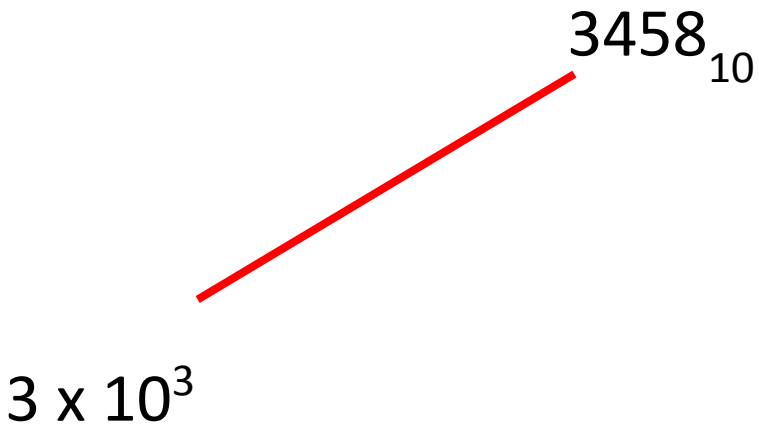
$0123456789ABCDEF_{16}$

1111111_{1}

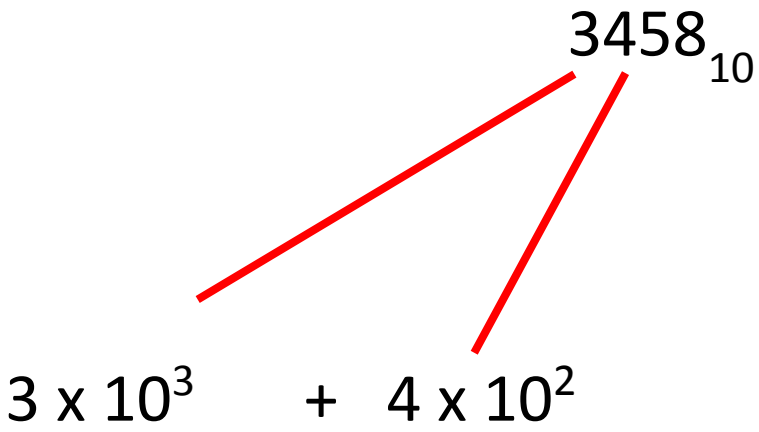
The Decimal Number System

3458_{10}

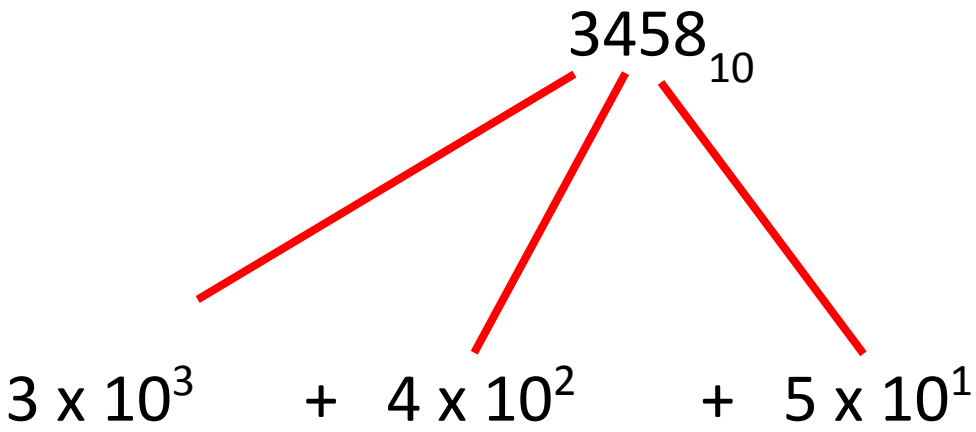
The Decimal Number System



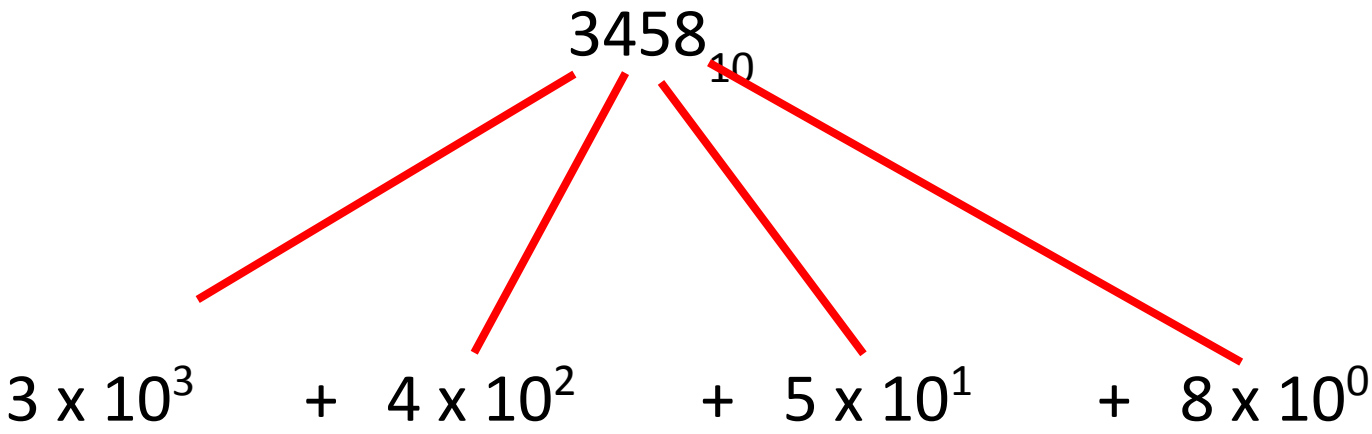
The Decimal Number System



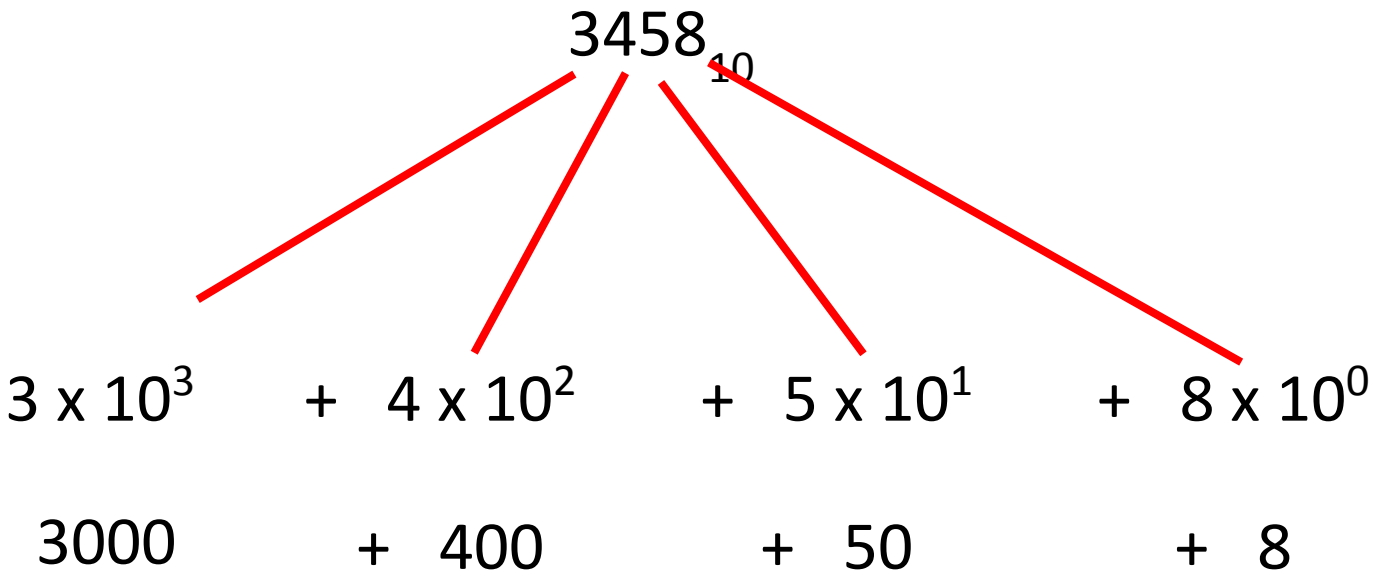
The Decimal Number System



The Decimal Number System



The Decimal Number System



Decimal to Binary Example

91_{10}

Decimal to Binary Example

Just keep dividing by 2

91_{10}

Decimal to Binary Example

Just keep dividing by 2

91_{10}

45 **R** 1

Decimal to Binary Example

Just keep dividing by 2

91_{10}

45 **R** 1

22 **R** 1

Decimal to Binary Example

Just keep dividing by 2

91_{10}

45 **R** 1

22 **R** 1

11 **R** 0

Decimal to Binary Example

Just keep dividing by 2

91_{10}

45 **R** 1

22 **R** 1

11 **R** 0

5 **R** 1

Decimal to Binary Example

Just keep dividing by 2

91_{10}

45 **R** 1

22 **R** 1

11 **R** 0

5 **R** 1

2 **R** 1

Decimal to Binary Example

Just keep dividing by 2

91_{10}

45 **R** 1

22 **R** 1

11 **R** 0

5 **R** 1

2 **R** 1

1 **R** 0

Decimal to Binary Example

Just keep dividing by 2

91_{10}

45 **R** 1

22 **R** 1

11 **R** 0

5 **R** 1

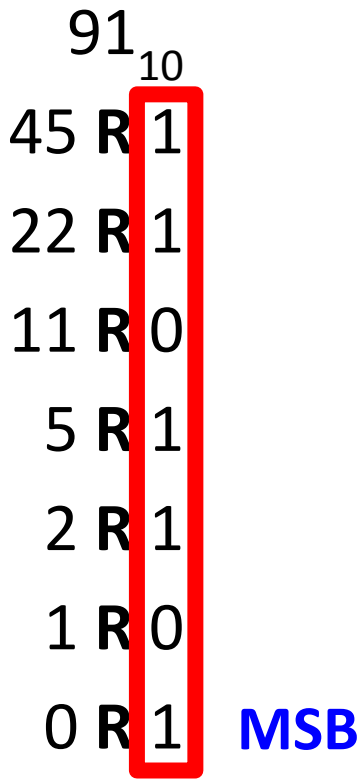
2 **R** 1

1 **R** 0

0 **R** 1

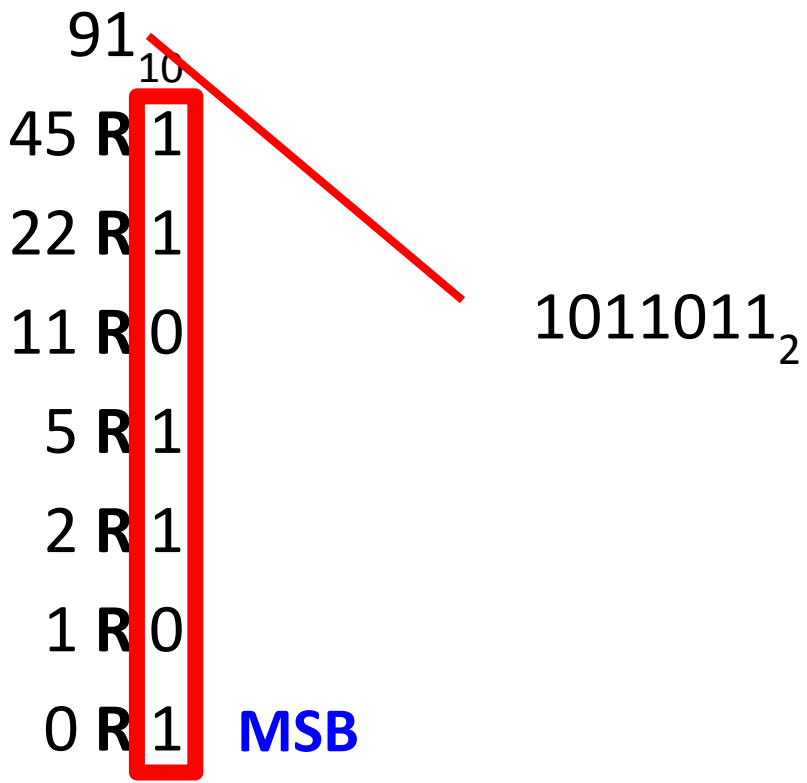
Decimal to Binary Example

Just keep dividing by 2



Decimal to Binary Example

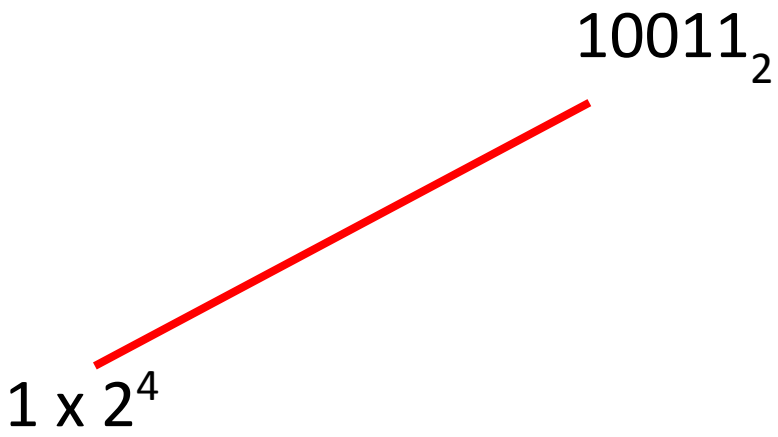
Just keep dividing by 2



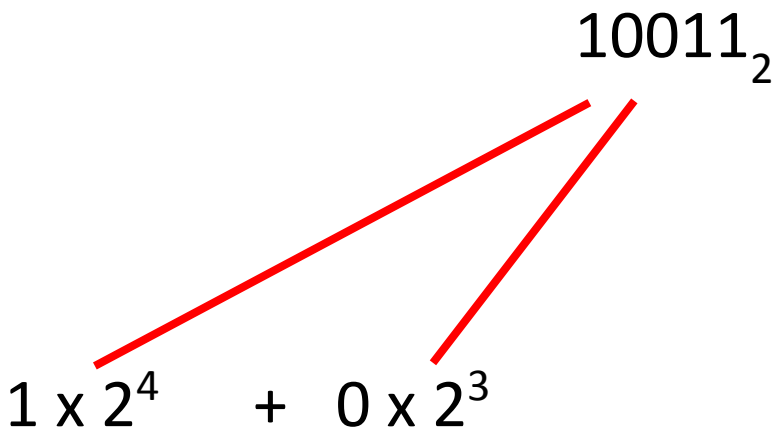
Binary to Decimal Example

10011_2

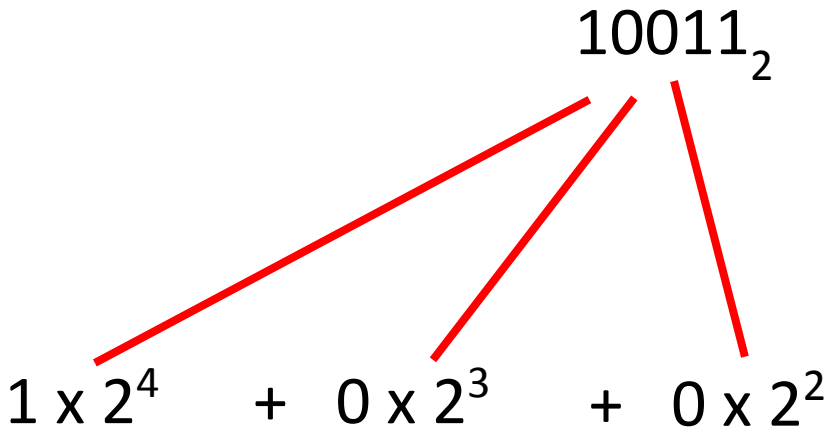
Binary to Decimal Example



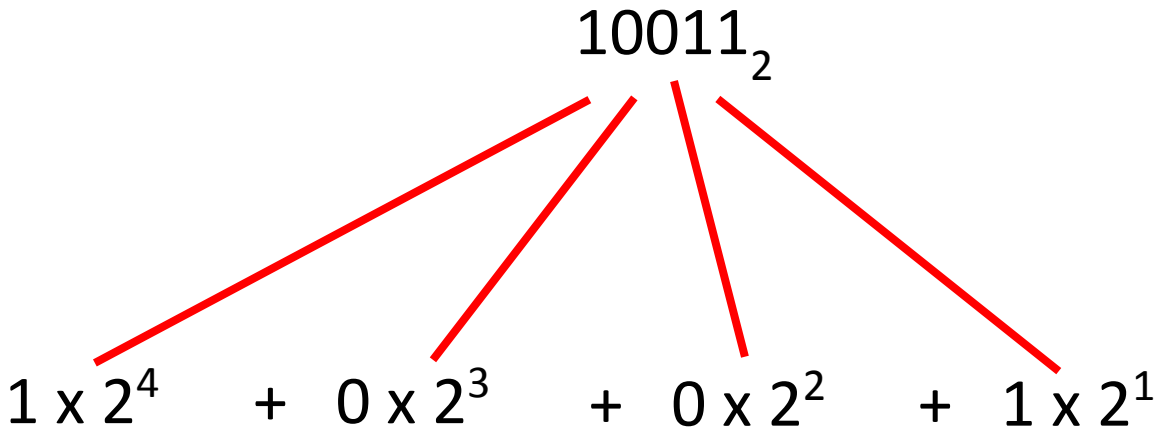
Binary to Decimal Example



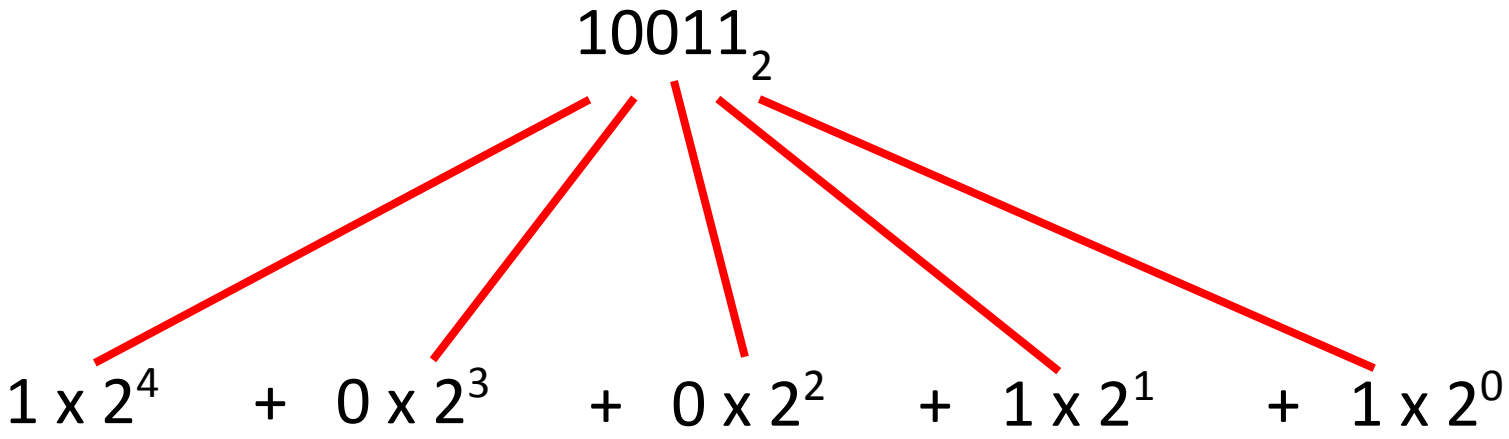
Binary to Decimal Example



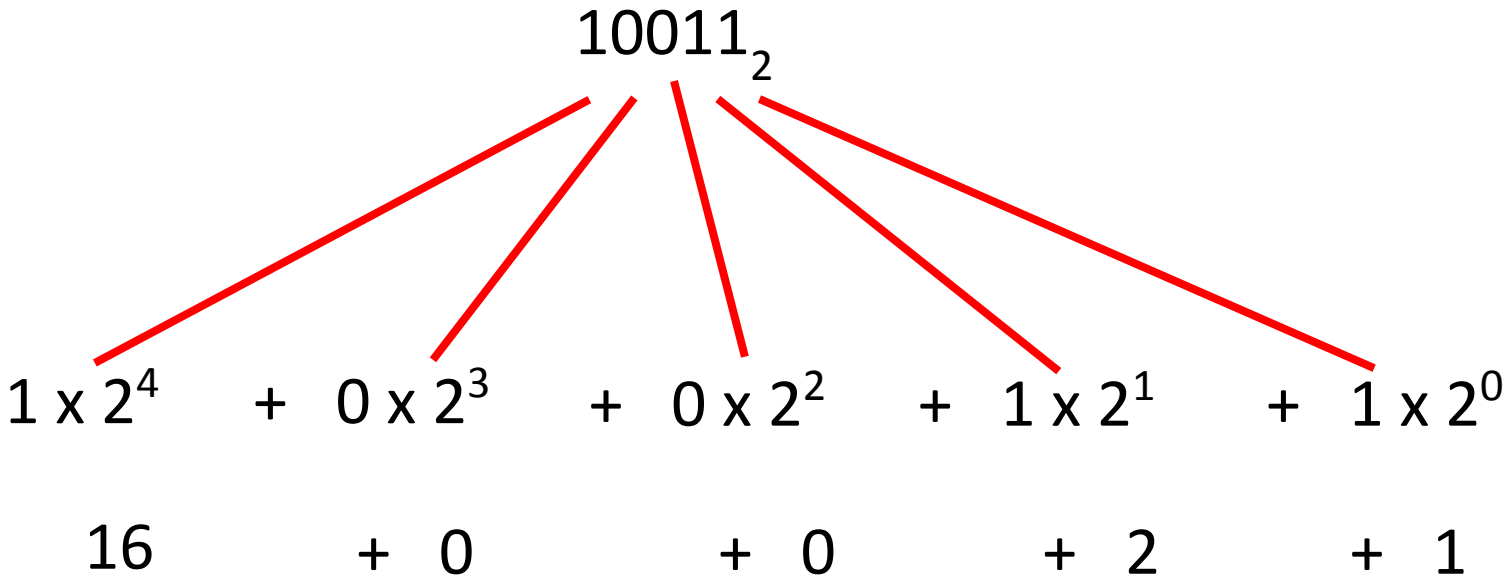
Binary to Decimal Example



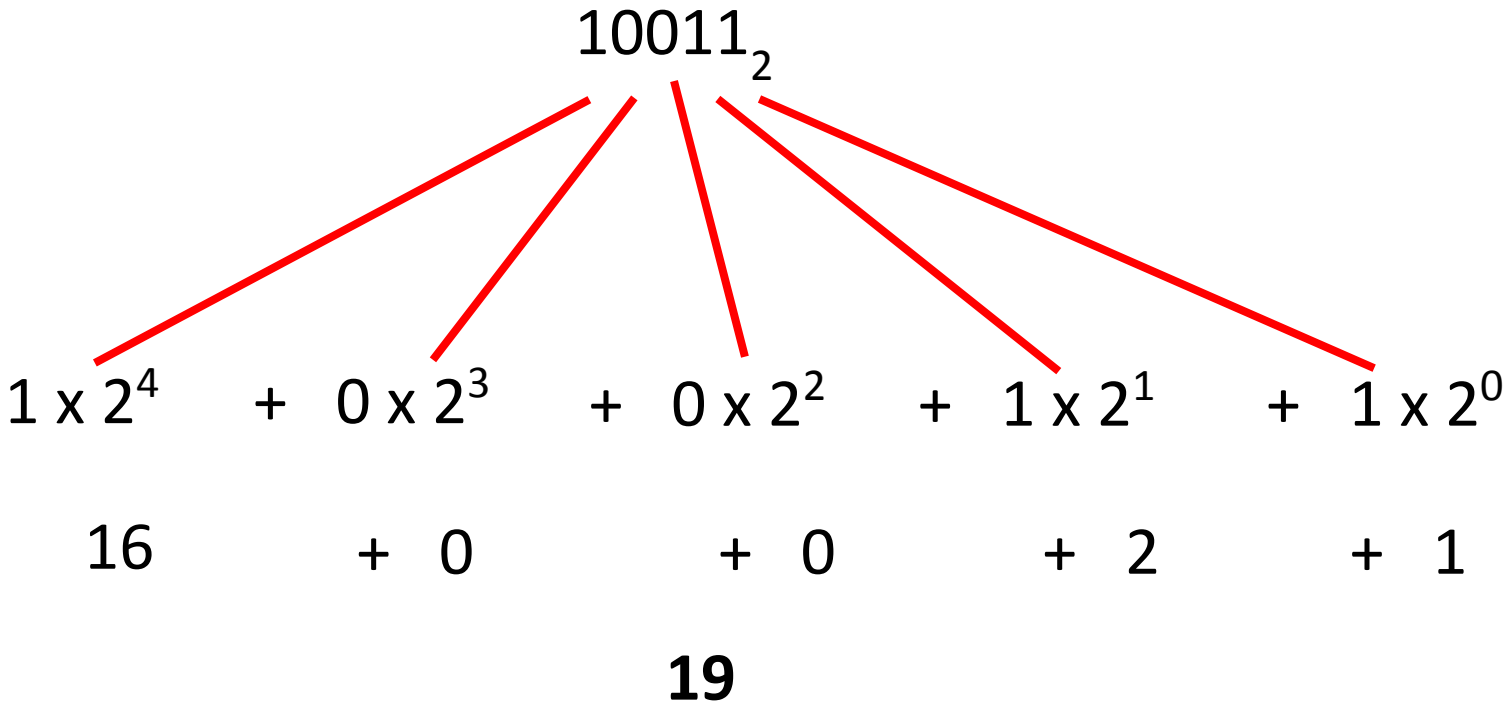
Binary to Decimal Example



Binary to Decimal Example

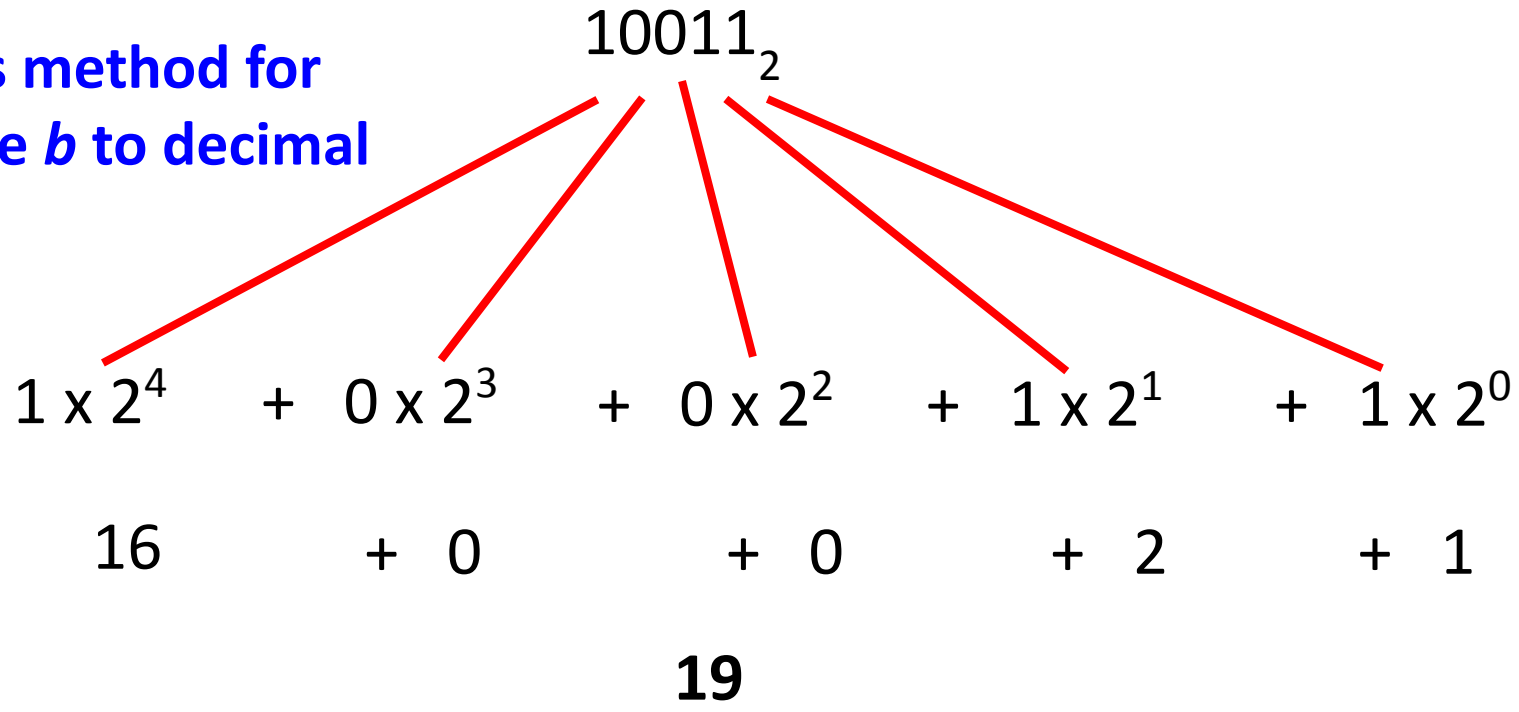


Binary to Decimal Example



Binary to Decimal Example

Use this method for any base *b* to decimal



Hexadecimal to Decimal Example

$2DF_{16}$

Hexadecimal to Decimal Example

0	1	2	3	4	5	6	7
0	1	2	3	4	5	6	7

8	9	10	11	12	13	14	15
8	9	A	B	C	D	E	F

$2DF_{16}$

Hexadecimal to Decimal Example

0	1	2	3	4	5	6	7
0	1	2	3	4	5	6	7

8	9	10	11	12	13	14	15
8	9	A	B	C	D	E	F

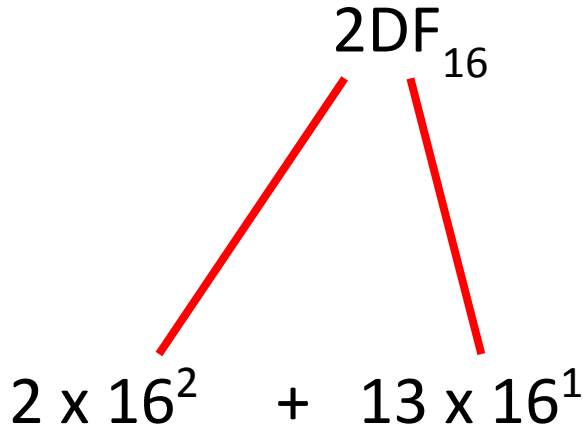
$2DF_{16}$

2×16^2

Hexadecimal to Decimal Example

0	1	2	3	4	5	6	7
0	1	2	3	4	5	6	7

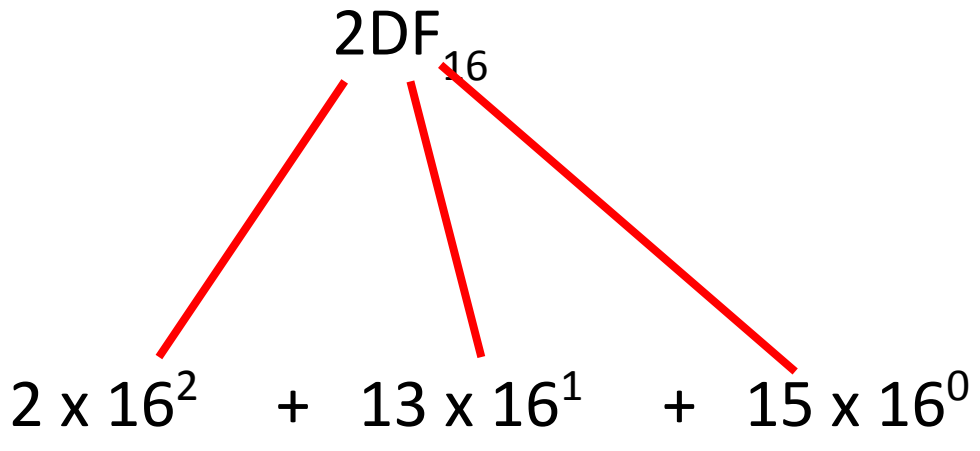
8	9	10	11	12	13	14	15
8	9	A	B	C	D	E	F



Hexadecimal to Decimal Example

0	1	2	3	4	5	6	7
0	1	2	3	4	5	6	7

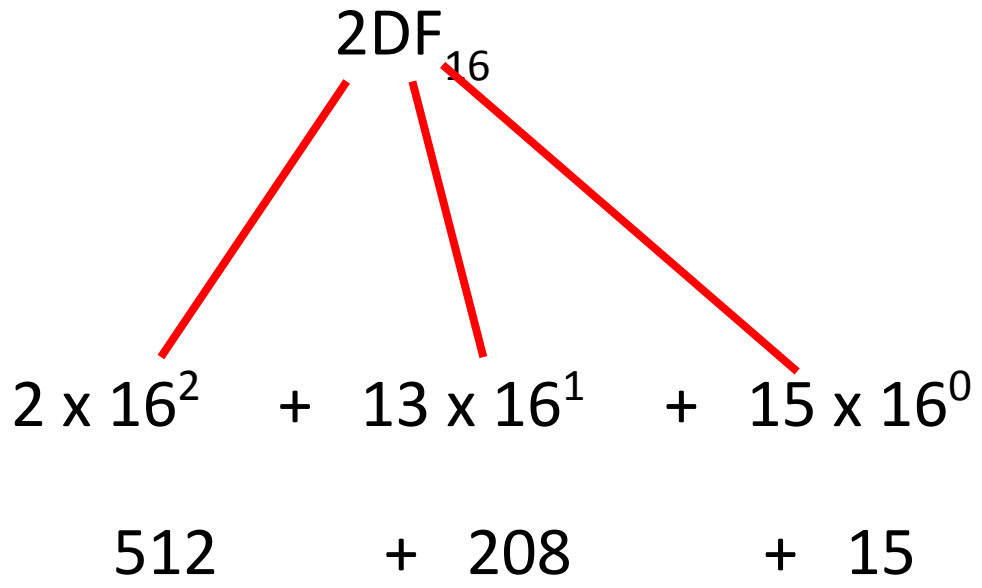
8	9	10	11	12	13	14	15
8	9	A	B	C	D	E	F



Hexadecimal to Decimal Example

0	1	2	3	4	5	6	7
0	1	2	3	4	5	6	7

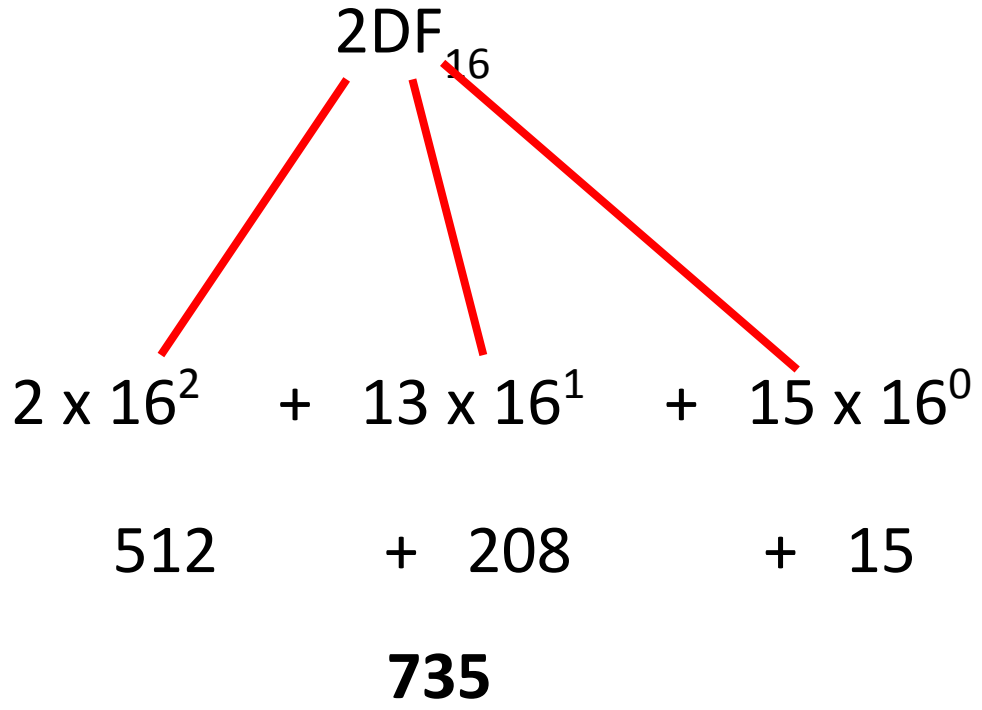
8	9	10	11	12	13	14	15
8	9	A	B	C	D	E	F



Hexadecimal to Decimal Example

0	1	2	3	4	5	6	7
0	1	2	3	4	5	6	7

8	9	10	11	12	13	14	15
8	9	A	B	C	D	E	F



Binary to Hexadecimal Example

11010010_2

Binary to Hexadecimal Example

1 hex symbol can represent 4 bits (2^4)

$$11010010_2$$

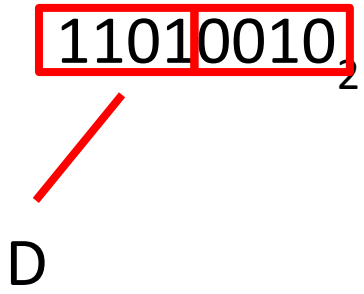
Binary to Hexadecimal Example

1 hex symbol can represent 4 bits (2^4)

11010010_2

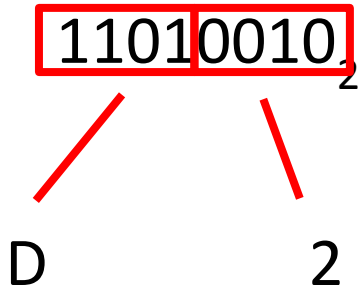
Binary to Hexadecimal Example

1 hex symbol can represent 4 bits (2^4)



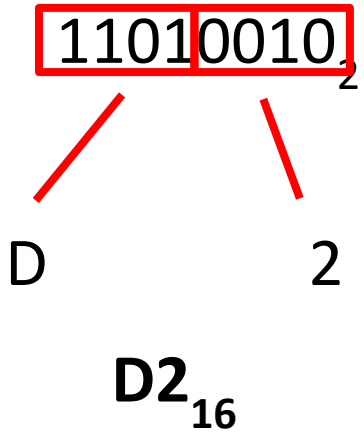
Binary to Hexadecimal Example

1 hex symbol can represent 4 bits (2^4)



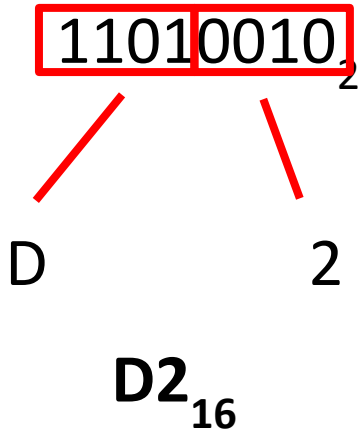
Binary to Hexadecimal Example

1 hex symbol can represent 4 bits (2^4)



Binary to Hexadecimal Example

1 hex symbol can represent 4 bits (2^4)



Hex to binary is no different

Properties of Binary Representation

- Even or odd?
  - Look at the LSB.

$1001011\mathbf{1}_2$ **Odd: 151**

Properties of Binary Representation

- Even or odd?
  - Look at the LSB.
- Bitwise shifts
  - Left = Multiply by 2.
  - Right = Divide by 2.

10010111_2 **Odd: 151**

100101110_2 **302**

001001011_2 **75**

Properties of Binary Representation

- Even or odd?
  - Look at the LSB.
- Bitwise shifts
  - Left = Multiply by 2.
  - Right = Divide by 2.
- Multiplying any two *n* bit binary values requires to most 2n bits.
  - Adding requires at most n+1 bits.

10010111_2 **Odd: 151**

100101110_2 **302**

001001011_2 **75**

Computing in Binary

- The smallest addressable unit of data on computer is a byte.
  - This is precisely 8 bits.
  - We can represent 2^8 (256) values.
- If we want to talk bits, we need to be a *bit* more clever.

```
// sum primes up to limit by sieving
long long sum_primes(long long limit){
   long long sum = 0;
   bool* marks = malloc(sizeof(bool)*(limit+1));
   if(marks == NULL)
   {
      perror("Malloc failed!");
      abort();
   }
   for(long long i=0; i < limit+1; i++){
      marks[i] = false;
   }
   for(long long i = 2; i <= limit; i++){

      if(!marks[i]){
         sum+=i;

         for(long long m=2*i; m <= limit; m += i)
         {
            marks[m] = true;
         }
      }
   }
   free(marks);
   return sum;
}
```

Computing in Binary

- The smallest addressable unit of data on computer is a byte.
  - This is precisely 8 bits.
  - We can represent 2^8 (256) values.
- If we want to talk bits, we need to be a *bit* more clever.

```
// sum primes up to limit by sieving
long long sum_primes(long long limit){
   long long sum = 0;
   bool* marks = malloc(sizeof(bool)*(limit+1));
   if(marks == NULL)
   {
      perror("Malloc failed!");
      abort();
   }
   for(long long i=0; i < limit+1; i++){
      marks[i] = false;
   }
   for(long long i = 2; i <= limit; i++){

      if(!marks[i]){
         sum+=i;

         for(long long m=2*i; m <= limit; m += i)
         {
            marks[m] = true;
         }
      }
   }
   free(marks);
   return sum;
}
```

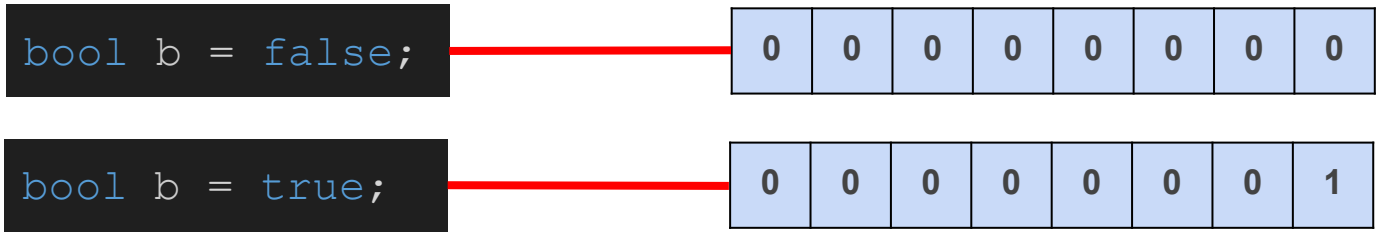
7 bits are unused…

The Bitmap

Idea: Each bit of a byte represents a true or false value.

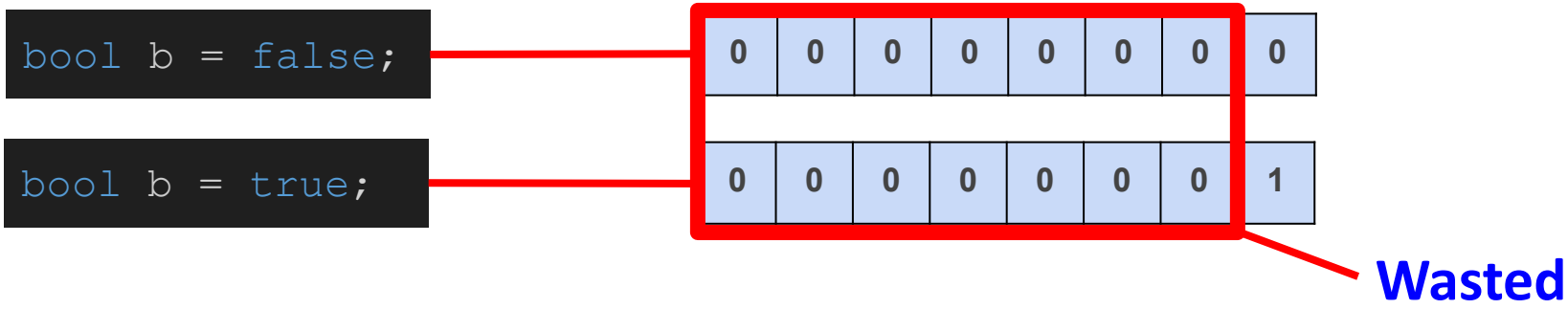
The Bitmap

Idea: Each bit of a byte represents a true or false value.



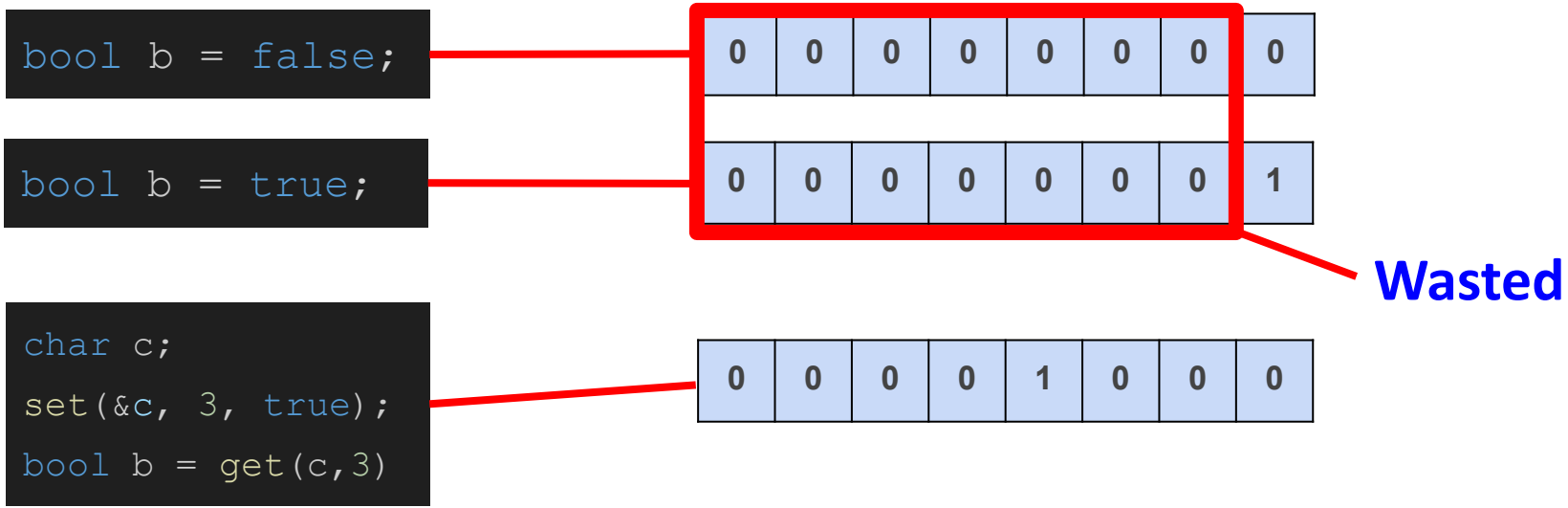
The Bitmap

Idea: Each bit of a byte represents a true or false value.



The Bitmap

Idea: Each bit of a byte represents a true or false value.



The Bitmap

Idea: Each bit of a byte represents a true or false value.