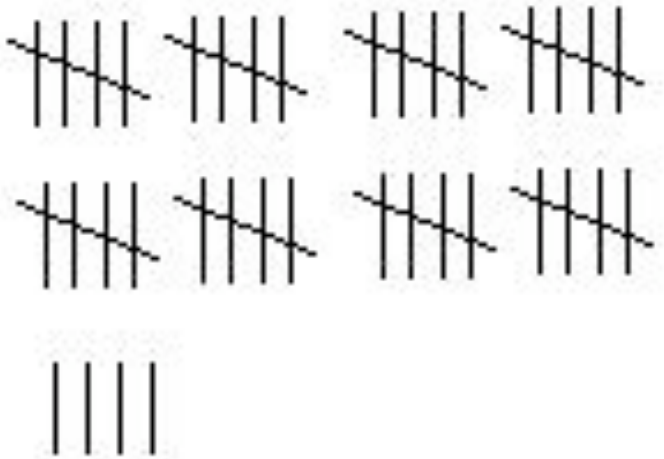
Representing Numbers

The Digital World

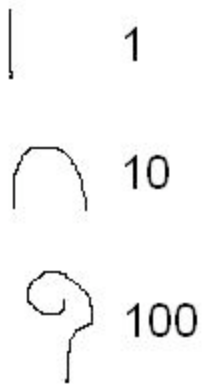
Number-writing Schemes

- Tally marks



Number-writing Schemes

- Tally marks
- Additive number systems—each symbol has a fixed value, e.g., additive decimal system of ancient Egypt:



- Twenty-three
- Fifteen
- (Somebody please find me an image of an inscription showing higher-order numerals!)

Number-writing Schemes

- Tally marks
- Additive number systems—each symbol has a fixed value, e.g., additive decimal system of ancient Egypt.
- **Positional number systems**—the value of a symbol depends on its position.

Positional Number Systems

- In our system, there are ten symbols, with unweighted values zero, one,…,nine.
- Positions correspond to powers of ten (one, ten, one hundred, one thousand,…)
- In “30185”, the 3 has value thirty-thousand, the 8 has value eighty, etc. Note how important zero is.
- Since the weights are powers of ten, this is a *base ten*, or *radix ten*, or *decimal system*.

Positional Number Systems

- Any base > 1 can be used. For instance in base 5, there are 5 symbols: 0,1,2,3,4. “1473” then denotes $1x5^3+4x5^2+7x5+3=263$
- Need to write something like

  $1473_{five}=263_{ten}$

  to avoid ambiguity.
- First positional number system was base 60 in Babylonia c. 4000 years ago. Base 20 system used in Central America 2000 (?) years ago.
- Our system (Hindu-Arabic numerals) originated in India about 700 AD.

Hindu-Arabic Numerals

4703 means

4 thousands +

7 hundreds +

0 tens +

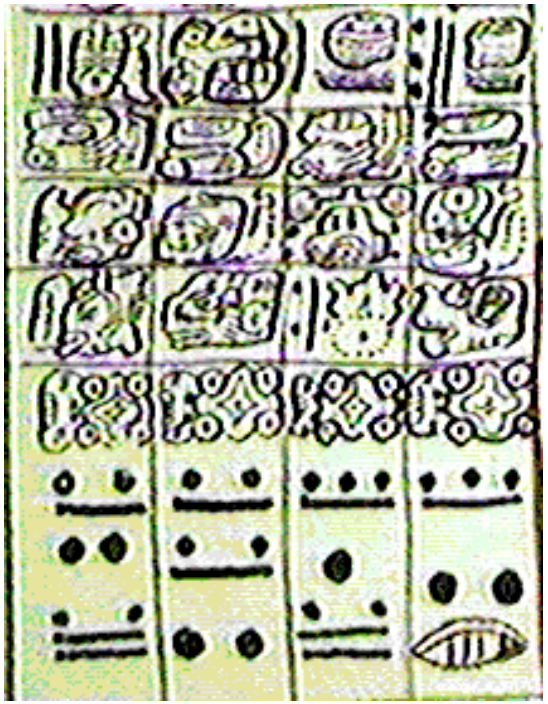
3 ones

$4x10^3+7x10^2+0x10^1+3x10^0$

Positional decimal system

(a symbol's value depends on its position).

Mayan base twenty positional number system



- Each “digit” built with additive system in which dot is 1 and line is 5.
- The shell-like thing is zero.
- The most significant digit is at the top.
- What are these four numbers?

Computers usually represent whole numbers in positional **binary** (radix 2) system. There are two distinct digits, 0 and 1. (BIT = “Binary digIT”)

For instance, 1001 is the binary representation of 1x8 + 0x4 + 0x2 +1x1 = 9. 1101 is the binary representation of 1x8 + 1x4 + 0x2 + 1x1=13, i.e.,

$1101_{two}=13_{ten}$

Note connection with hex digits: 0000,..,1001 represent values zero through nine. 1010,…, 1111 represent ten through fifteen, written as A,B,C,D,E,F.

Algorithms for Base Conversion

Decimal to Binary

Algorithm 1: Starting with most significant bit

Make a three-row table. In the second row, list the powers of 2 from from right to left, until the largest power of 2 that does not exceed the number to convert. Write the number to convert in the upper left cell of the table.

Repeatedly perform the operation illustrated below, moving from left to right.

N	N if $N<2^K$, $N-2^K$ otherwise
2^k	2^{k-1}
0 if $N<2^k$, 1 otherwise	

Example: Convert 2743 to binary.

2743											
2048	1024	512	256	128	64	32	16	8	4	2	1

2743	695										
2048	1024	512	256	128	64	32	16	8	4	2	1
1											

2743	695	695									
2048	1024	512	256	128	64	32	16	8	4	2	1
1	0										

2743	695	695	183								
2048	1024	512	256	128	64	32	16	8	4	2	1
1	0	1									

2743	695	695	183	183							
2048	1024	512	256	128	64	32	16	8	4	2	1
1	0	1	0								

2743	695	695	183	183	55						
2048	1024	512	256	128	64	32	16	8	4	2	1
1	0	1	0	1							

2743	695	695	183	183	55	55					
2048	1024	512	256	128	64	32	16	8	4	2	1
1	0	1	0	1	0						

2743	695	695	183	183	55	55	23				
2048	1024	512	256	128	64	32	16	8	4	2	1
1	0	1	0	1	0	1					

2743	695	695	183	183	55	55	23	7			
2048	1024	512	256	128	64	32	16	8	4	2	1
1	0	1	0	1	0	1	1				

2743	695	695	183	183	55	55	23	7	7		
2048	1024	512	256	128	64	32	16	8	4	2	1
1	0	1	0	1	0	1	1	0			

2743	695	695	183	183	55	55	23	7	7	3	
2048	1024	512	256	128	64	32	16	8	4	2	1
1	0	1	0	1	0	1	1	0	1		

2743	695	695	183	183	55	55	23	7	7	3	1
2048	1024	512	256	128	64	32	16	8	4	2	1
1	0	1	0	1	0	1	1	0	1	1	

2743	695	695	183	183	55	55	23	7	7	3	1
2048	1024	512	256	128	64	32	16	8	4	2	1
1	0	1	0	1	0	1	1	0	1	1	1

2743	695	695	183	183	55	55	23	7	7	3	1
2048	1024	512	256	128	64	32	16	8	4	2	1
1	0	1	0	1	0	1	1	0	1	1	1

Algorithm 2: Starting with least significant bit

Make a two-row table, with the number to convert in the upper right. Repeatedly divide by 2, retaining both remainder and quotient, until the quotient becomes 0

N/2 (Just the whole number part, without remainder.)	N
	N%2 (0 if N even, 1 if N odd)

Example: Convert 2743 to binary

										1371	2743
											1

Example: Convert 2743 to binary

									685	1371	2743
										1	1

Example: Convert 2743 to binary

								342	685	1371	2743
									1	1	1

Example: Convert 2743 to binary

							171	342	685	1371	2743
								0	1	1	1

Example: Convert 2743 to binary

						85	171	342	685	1371	2743
							1	0	1	1	1

Example: Convert 2743 to binary

					42	85	171	342	685	1371	2743
						1	1	0	1	1	1

Example: Convert 2743 to binary

				21	42	85	171	342	685	1371	2743
					0	1	1	0	1	1	1

Example: Convert 2743 to binary

			10	21	42	85	171	342	685	1371	2743
				1	0	1	1	0	1	1	1

Example: Convert 2743 to binary

		5	10	21	42	85	171	342	685	1371	2743
			0	1	0	1	1	0	1	1	1

Example: Convert 2743 to binary

	2	5	10	21	42	85	171	342	685	1371	2743
		1	0	1	0	1	1	0	1	1	1

Example: Convert 2743 to binary

1	2	5	10	21	42	85	171	342	685	1371	2743
	0	1	0	1	0	1	1	0	1	1	1

Example: Convert 2743 to binary

1	2	5	10	21	42	85	171	342	685	1371	2743
1	0	1	0	1	0	1	1	0	1	1	1

Binary to Decimal Conversion

Algorithm 1

Just write down the successive powers of 2 under the bits of the binary representation, and add those powers corresponding to a 1.

1	1	0	0	1	0	0	1
128	64	32	16	8	4	2	1
201	73			9			1

Algorithm 2-Starting with most significant bit

Make a two-row table with the binary representation in the top row, an extra column at left, and 0 in the lower left corner. Repeatedly perform the operation below, doubling the value in the second row and adding the next bit.

	b
X	2X+b

Example:

	1	1	0	0	1	0	0	1
0								

Example:

	1	1	0	0	1	0	0	1
0	1							

Example:

	1	1	0	0	1	0	0	1
0	1	3						

Example:

	1	1	0	0	1	0	0	1
0	1	3	6					

Example:

	1	1	0	0	1	0	0	1
0	1	3	6	12				

Example:

	1	1	0	0	1	0	0	1
0	1	3	6	12	25			

Example:

	1	1	0	0	1	0	0	1
0	1	3	6	12	25	50		

Example:

	1	1	0	0	1	0	0	1
0	1	3	6	12	25	50	100	

Example:

	1	1	0	0	1	0	0	1
0	1	3	6	12	25	50	100	201

Hex to Decimal and Back

You can convert between hex and binary very easily, and use the above algorithms.

Alternatively, you can use the fact that the hex representation is radix sixteen.

For instance A3(hex) represents 10x16+3=163 (decimal).

475 (decimal) = 256 +208+11=

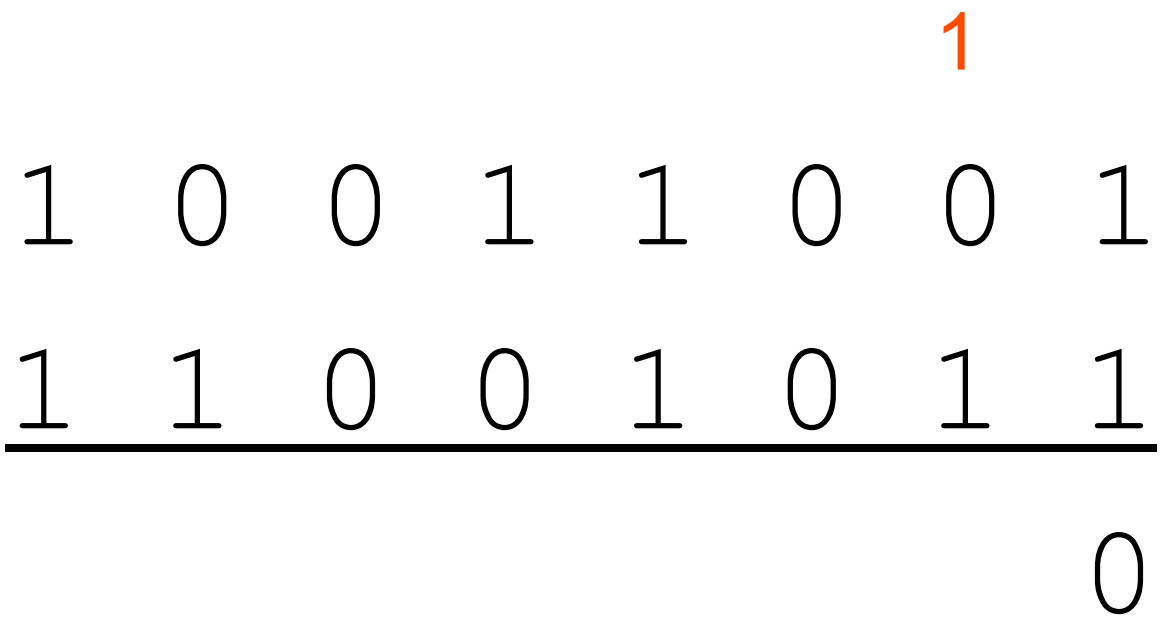
16^2+13x16+11=1DB (hex)

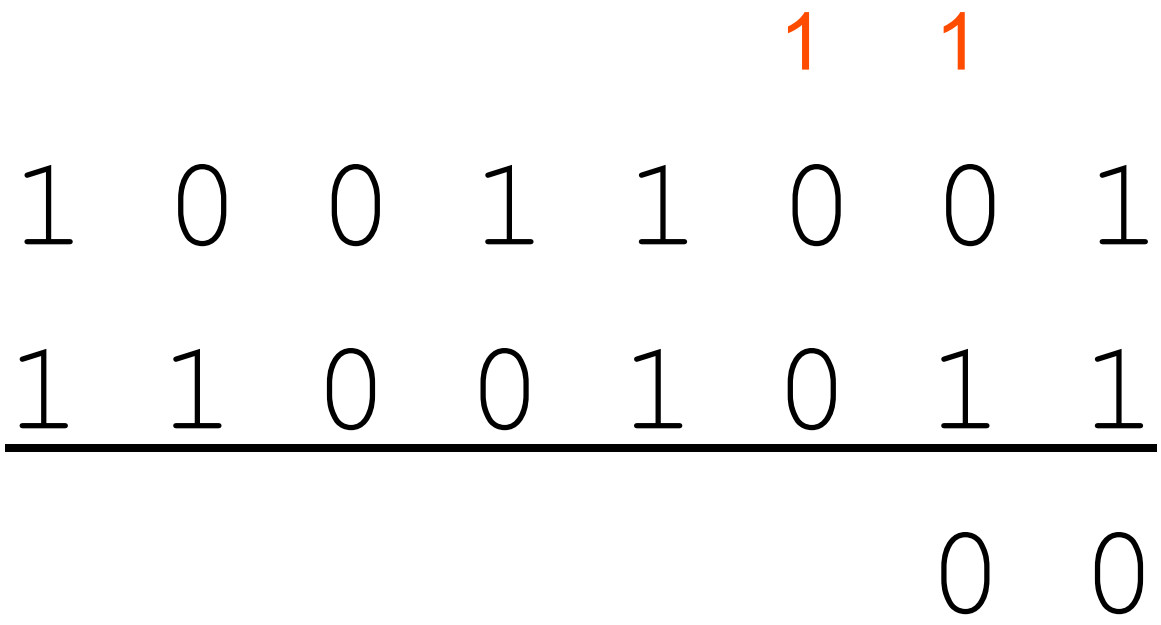
Arithmetic

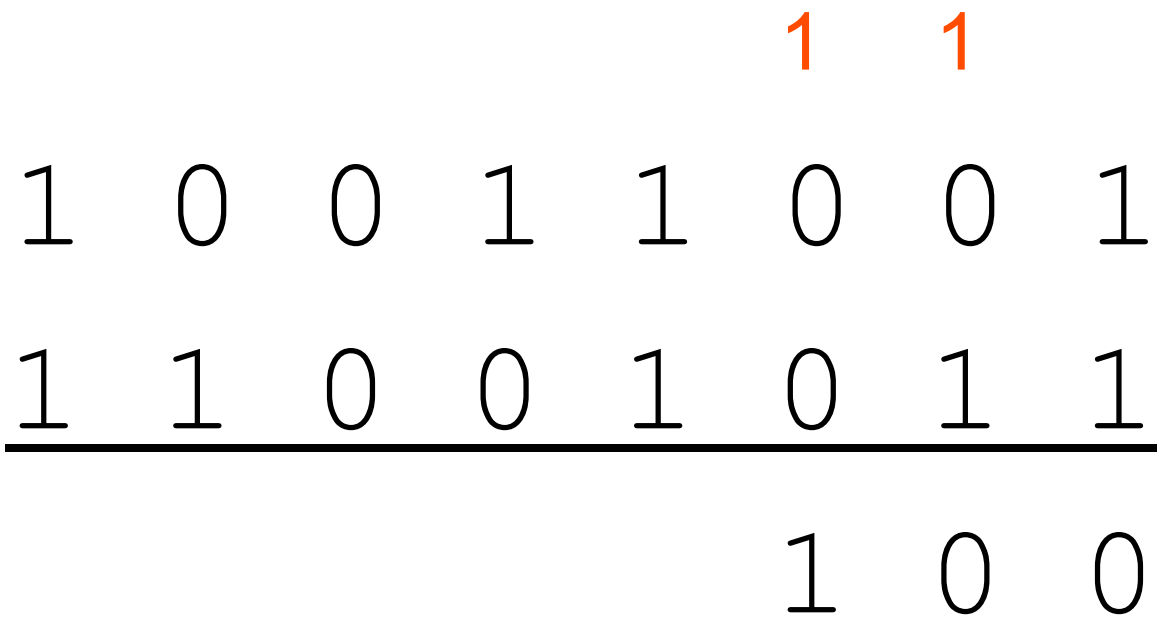
- Standard algorithms for addition, subtraction, multiplication and division work in every radix.
- Easier in binary (no multiplication tables, no guessing in long division).
- Binary versions are easier to implement in electric machines.

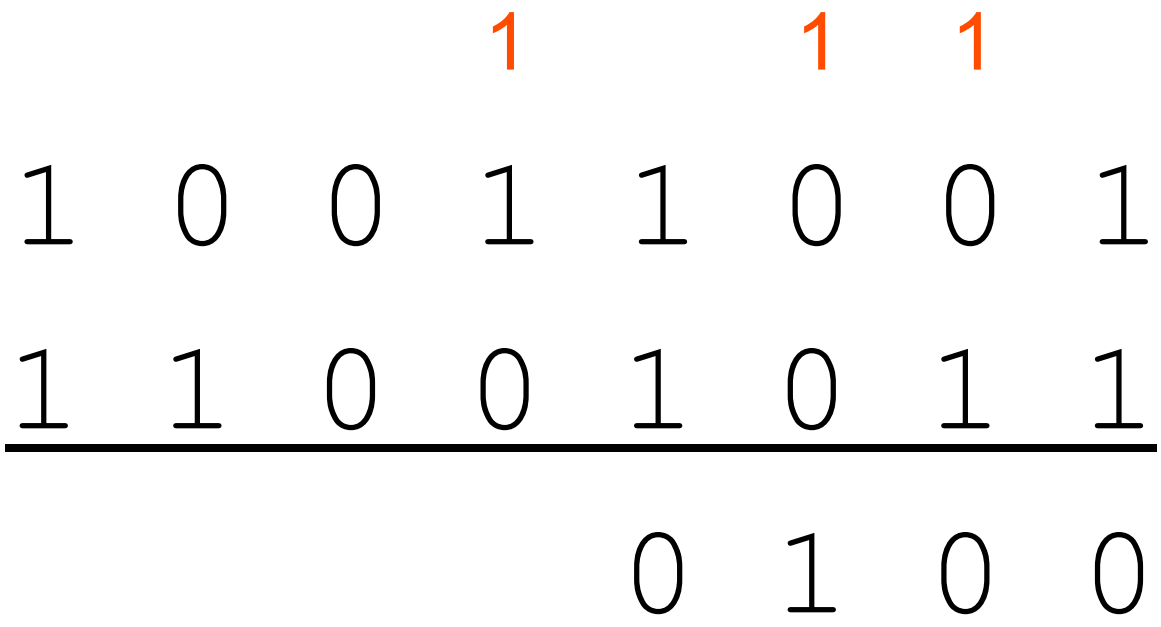
Addition

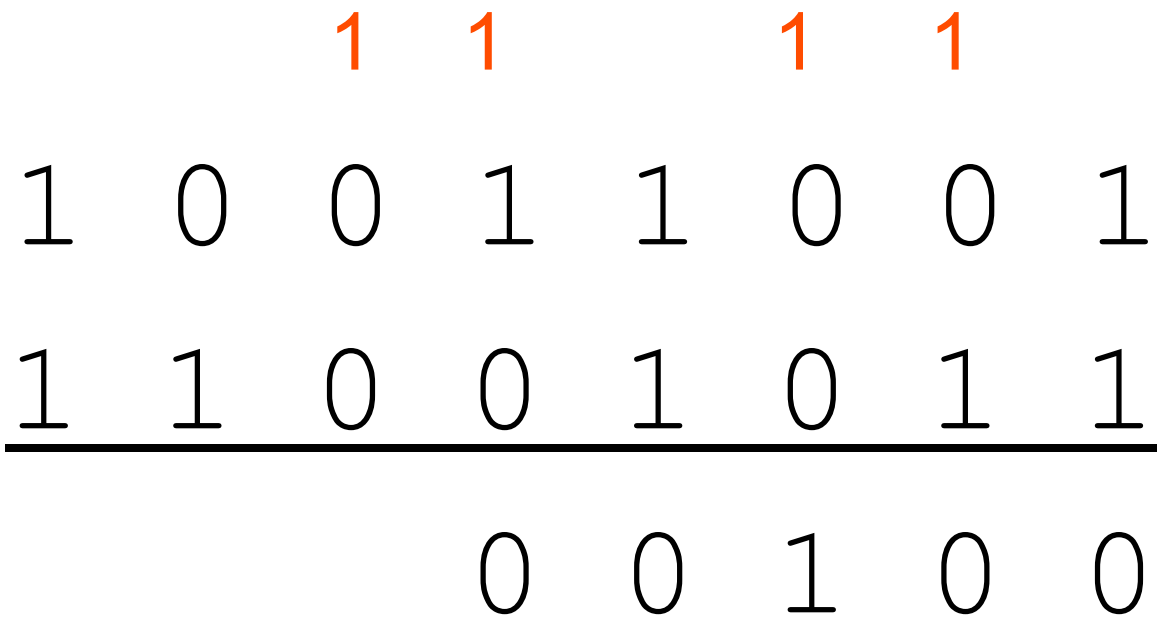
```
  1  0  0  1  1  0  0  1
  1  1  0  0  1  0  1  1
-------------------------
```

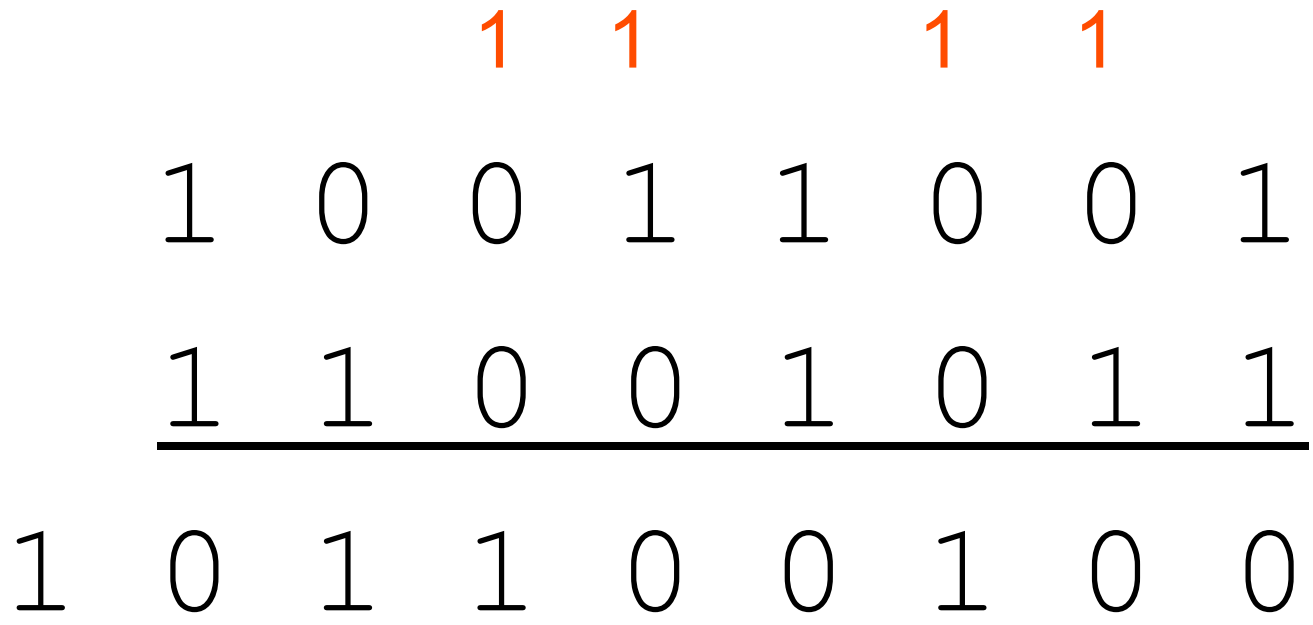

1
1 0 0 1 1 0 0 1
1 1 0 0 1 0 1 1
0


1 1
1 0 0 1 1 0 0 1
1 1 0 0 1 0 1 1
0 0


1 1
1 0 0 1 1 0 0 1
1 1 0 0 1 0 1 1
1 0 0


1 1 1
1 0 0 1 1 0 0 1
1 1 0 0 1 0 1 1
0 1 0 0

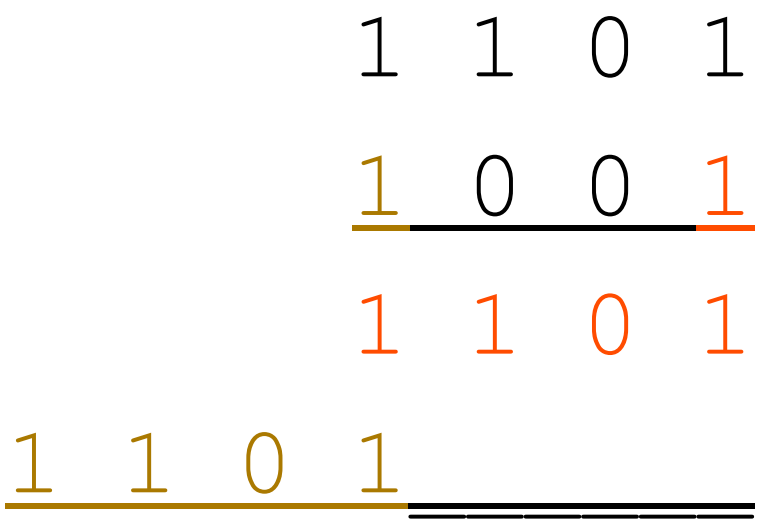

1 1 1 1
1 0 0 1 1 0 0 1
1 1 0 0 1 0 1 1
0 0 1 0 0


1 1 1 1
1 0 0 1 1 0 0 1
1 1 0 0 1 0 1 1
1 0 1 1 0 0 1 0 0

Multiplication

$$\begin{array}{r} 1\ 1\ 0\ 1 \\ \times\ 1\ 0\ 0\ 1 \\ \hline \end{array}$$

Write shifted first factor for every 1 in second factor



Then add

```
        1 1 0 1
        1 0 0 1
      ---------
        1 1 0 1
  1 1 0 1
---------------
1 1 1 0 1 0 1
```