ECE 150 Binary and hexadecimal

Douglas Harder

December 2023

1 Introduction

In this topic, we will discuss counting in binary and hexadecimal and conversions between binary, decimal and hexadecimal.

2 Decimal and positional number systems

We use base-10 arithmetic because we have ten unique digits where the digits 1 through 9 are used to represent the number of items seen here:

$$| \quad || \quad ||| \quad |||| \quad ||||| \quad |||||| \quad ||||||| \quad |||||||| \quad |||||||||$$

We represent ten items (i.e., ||||||||||) by 10, and ten groups of ten items by 100, and so on. We then represent a number like 50 or 300 as being as five groups of 10 items and three groups of 100 items, respectively. The number 1970 represents one group of 1000, nine groups of 100, seven groups of 10 and no groups of 1.

The idea that the significance of a number depends on its position is both oddly old and yet relatively new, and is called a *positional number system.* The ancient Babylonians had a positional number system, but that idea was lost for millennia.

Thus, $258437 = 200000 + 50000 + 8000 + 400 + 30 + 7$, and therefore this number represents

$$2 \times 10^5 + 5 \times 10^4 + 8 \times 10^3 + 4 \times 10^2 + 3 \times 10^1 + 7 \times 10^0.$$

In this representation, the order of the digits matters.

Not on the examination

Older systems were not positional. In Hebrew, each letter represented a number as shown below:

א	alef	1
ב	bet	2
ג	gimel	3
ד	dalet	4
ה	hey	5
ו	vav	6
ז	zayin	7
ח	chet	8
ט	tet	9
י	yod	10
ך	kaf	20
ל	lamed	30
ם	mem	40
ן	nun	50
ס	samekh	60
ע	ayin	70
ף	pe	80
ץ	tsade	90
ק	qof	100
ר	resh	200
שׁ	shin	300
ת	tav	400
ך	final kaf	500
ם	final mem	600
ן	final nun	700
ף	final pe	800
ץ	final tsade	900

Thus, םיו, וים and ימו all represent 616. You could even replace the ו with two letters that sum up to six, so you could also use בםדי. It would be as if be as if 325, 523 and 91 all represent $5 + 3 + 2$ or $9 + 1$ which we would describe as *ten*. For example, a transliteration of Nero Caesar into Hebrew is נרוקסר where the letters have the numeric value $50 + 200 + 6 + 100 + 60 + 200 = 616$.

Not on the examination

Roman numerals is similar to Hebrew, but used two letters per power of ten:

I	1
V	5
X	10
L	50
C	100
D	500
M	1000
$\overline{\mathrm{V}}$	5000
$\overline{\mathrm{X}}$	10000
$\overline{\mathrm{L}}$	50000
$\overline{\mathrm{C}}$	100000
$\overline{\mathrm{D}}$	500000
$\overline{\mathrm{M}}$	1000000

In Hebrew, there would be a default representation, but any combination of the letters worked. For Roman numerals, there is a fixed representation for each value with no option to rearrange the numerals. For each power of 10, the two letters were combined as follows:

1-9	I II III IV V VI VII VIII IX
10-90	X XX XXX XL L LX LXX LXXX XC
100-900	C CC CCC CD D DC DCC DCCC CM
1000-9000	M MM MMM $\mathrm{M}\overline{\mathrm{V}}$ $\overline{\mathrm{V}}$ $\overline{\mathrm{V}}\mathrm{M}$ $\overline{\mathrm{V}}\mathrm{MM}$ $\overline{\mathrm{V}}\mathrm{MMM}$ $\mathrm{M}\overline{\mathrm{X}}$
10000-90000	$\overline{\mathrm{X}}$ $\overline{\mathrm{XX}}$ $\overline{\mathrm{XXX}}$ $\overline{\mathrm{XL}}$ $\overline{\mathrm{L}}$ $\overline{\mathrm{LX}}$ $\overline{\mathrm{LXX}}$ $\overline{\mathrm{LXXX}}$ $\overline{\mathrm{XC}}$
100000-900000	$\overline{\mathrm{C}}$ $\overline{\mathrm{CC}}$ $\overline{\mathrm{CCC}}$ $\overline{\mathrm{CD}}$ $\overline{\mathrm{D}}$ $\overline{\mathrm{DC}}$ $\overline{\mathrm{DCC}}$ $\overline{\mathrm{DCCC}}$ $\overline{\mathrm{CM}}$
1000000-3000000	$\overline{\mathrm{M}}$ $\overline{\mathrm{MM}}$ $\overline{\mathrm{MMM}}$

If you wanted to represent 99, you could not use IC (one before one hundred), but rather you would use XCIX. If scribes needed numbers equal to or greater than four million, they likely came up with there own notation, but there was no standard. To create 1970, you would combine the various values: MCMDXX. To represent 1888, you would require MDCCCLXXXVIII. There was no representation of 0.

3 Binary

In binary, we only allow two digits: 0 and 1. This is also called a base-2 number system. As with decimal, where it is positional on powers of 10, in binary, it is powers of two, so $1001101 = 1000000 + 1000 + 100 + 1$. To indicate that this is a binary number, we will use a subscript two after the number, so 1001101_2, or we will prefix it with a `0b` using a fixed-width font, such as `0b1001101`.

Now, in binary, $2^0 = 1 = 1_2$, $2^1 = 2 = 10_2$, $2^2 = 4 = 100_2$, $2^3 = 8 = 1000_2$, $2^4 = 16 = 10000_2$, and so on, so

$$1001101_2 = 1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0.$$

Because it is only multipliers by 0 or 1, it's easier just to add the powers of two not multiplied by 0:

$$1001101_2 = 2^6 + 2^3 + 2^2 + 2^0 = 77_{10}.$$

Similarly, 110011101_2 represents the number

$$2^8 + 2^7 + 2^4 + 2^3 + 2^2 + 2^0 = 413_{10}.$$

We will emphasize that a number is in binary by either adding a subscript 2 (so 10 is ten, but 10_2 is two), or prefix a fixed font with `0b`, so `0b10` representing two.

Not on the examination

The reason we use only two digits is because 0 is represented by 0 V and 1 is represented by 5 V (although in some systems designed for embedded systems, 3.3 V and even 1.8 V are being used).
It is much more difficult to design a system where there are ten different voltages representing ten different digits.

An unsigned integer is either 16, 32 or 64 bits, so the largest value that can be stored as an `unsigned int` is $11111111111111111111111111111111_2$ or $1 + 2 + 2^2 + 2^3 + 2^4 + \cdots + 2^{31}$. Calculating this would be tedious at best, but we use the theorem that gives a formula for a geometric sum:

$$\sum_{k=0}^{n} r^k = \frac{r^{k+1} - 1}{r - 1}.$$

If you replace r with 2, the denominator becomes 1 and we have

$$\sum_{k=0}^{n} 2^k = 2^{k+1} - 1,$$

so the above sum is $2^{32} - 1$, being the largest integer that can be stored as an `unsigned int`. Similarly, the largest value that can be stored as an `unsigned short` is $2^{16}-1$ and the largest value that can be stored as an `unsigned long` is $2^{64}-1$.

When you add 1 to 1 in decimal, you get the next highest digit, 2. When you add 1 to 9, this results in a carry, so we write down the 0 and carry the 1, and in the left-most column, there are two implied zeros, so we have 1 (the carry) + $0+0=1$, so the result is 10.

The same occurs in binary, but carries happen more frequently:

0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111
16	10000
17	10001
18	10010
19	10011
20	10100
21	10101
22	10110
23	10111
24	11000
25	11001
26	11010
27	11011
28	11100
29	11101
30	11110
31	11111
32	100000
33	100001

You will notice that each power of two equaling 2^n is a 1 followed by n zeros. In decimal, a number equaling 10^n is represented by a 1 followed by n zeros. Thus, we have that powers of two in binary are:

$1_2=1, 10_2=2, 100_2=4, 1000_2=8, 10000_2=16, 100000_2=32, 1000000_2=64,$

$$10000000_2 = 128, 100000000_2 = 256, 1000000000_2 = 512, 10000000000_2 = 1024$$

The last one is significant: $2^{10} = 1024 \approx 1000 = 10^3$, and therefore 1 million $\approx 2^{20}$ and 1 billion $\approx 2^{30}$.

3.1 Conversions between binary and decimal

To convert a binary integer to a decimal integer, just add the relevant powers of two. For example:

1. 101010_2 is equal to $2^5 + 2^3 + 2^1 = 32 + 8 + 2 = 42$.
2. 10001001011 is equal to $2^{10}+2^6+2^3+2^1+2^0 = 1024+64+8+2+1 = 1099$.

Not on the examination

To convert a decimal number n from decimal to binary, we use the following algorithm:

1. If the decimal number is 0, the binary number is 0.
2. Otherwise, start with an empty result, and iterate while $n \neq 0$:
   (a) Calculate the quotient and the remainder when calculating $n \div 2$. The remainder will be either 0 or 1. Prepend the remainder to the result, and assign to n the value of the quotient.

For example, to convert 147 to binary:

1. $147 = 2 \times 73 + 1$, so the result is now $\underline{1}$.
2. $73 = 2 \times 36 + 1$, so the result is now $\underline{1}1$.
3. $36 = 2 \times 18 + 0$, so the result is now $\underline{0}11$.
4. $18 = 2 \times 9 + 0$, so the result is now $\underline{0}011$.
5. $9 = 2 \times 4 + 1$, so the result is now $\underline{1}0011$.
6. $4 = 2 \times 2 + 0$, so the result is now $\underline{0}10011$.
7. $2 = 2 \times 1 + 0$, so the result is now $\underline{0}010011$.
8. $1 = 0 \times 1 + 1$, so the result is now $\underline{1}0010011$.

With the last step $n \leftarrow 0$. Thus, 147 in binary is 10010011_2.

4 Hexadecimal

Hexadecimal, also known as a base-16 number system, is where counting uses sixteen digits, and these are generally shown as 0, 1, 2, 3, 4, 5,6, 7, 8, 9, a, b, c,d, e and f. Now, when you add 1 to f, you get a carry, resulting in 10_{16}, and thus 10_{16} in hexadecimal represents 16, just like 10_2 in binary represents 2, and 10_{10} in decimal represents, well, ten.

To indicate that a number is in hexadecimal, we will use a subscript sixteen after the number, so $47fe2_{16}$, or we will prefix it with a `0x` using a fixed-width typeface, such as `0x47fe2`.

The numbers on the left are the first 33 integers in decimal, and their representation in hexadecimal is on the right.

0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	a
11	b
12	c
13	d
14	e
15	f
16	10
17	11
18	12
19	13
20	14
21	15
22	16
23	17
24	18
25	19
26	1a
27	1b
28	1c
29	1d
30	1e
31	1f
32	20
33	11

4.1 Conversions between hexadecimal and decimal

Not on the examination

To convert a hexadecimal number to decimal requires you to add the corresponding values: $23ac5 = 2 \times 16^4 + 3 \times 16^3 + 10 \times 16^2 + 12 \times 16^1 + 5 \times 16^0$ which equals $2 \times 65536 + 3 \times 4096 + 10 \times 256 + 12 \times 16 + 5 =$ which equals 146117. You will notice that a is interpreted as multiplication by 10, and c is interpreted as multiplciation by 12.

Not on the examination

To convert a decimal number n from decimal to hexadecimal, we use the following algorithm:

1. If the decimal number is 0, the hexadecimal number is 0.
2. Otherwise, start with an empty result, and iterate while $n \neq 0$:
   (a) Calculate the quotient and the remainder when dividing $n \div 16$. The remainder will be a value between 0 and 15. Prepend the remainder as a hexadecimal digit to the result (so if the remainder is 11, prepend a b), and assign to n the value of the quotient.

For example, to convert 197036 to hexadecimal:

1. $197036 = 16 \times 12314 + 12$ and 12 is c, so the result is now $\underline{c}$.
2. $12314 = 16 \times 769 + 10$ and 10 is a, so the result is now $\underline{a}c$.
3. $769 = 16 \times 48 + 1$, so the result is now $\underline{1}ac$.
4. $48 = 3 \times 16 + 0$, so the result is now $\underline{0}1ac$.
5. $3 = 0 \times 16 + 3$, so the result is now $\underline{3}01ac$.

With the last step $n \leftarrow 0$. Thus, 197036 in hexadecimal is $301ac$ or `0x301ac`.

4.2 Conversions between hexadecimal and binary

Hexadecimal is a convenient way for humans to read binary. The conversion from hexadecimal to binary is straight-forward:

1. For each hexadecimal digit, replace it with the following four bits according to this table:

0	`0000`
1	`0001`
2	`0010`
3	`0011`
4	`0100`
5	`0101`
6	`0110`
7	`0111`
8	`1000`
9	`1001`
a	`1010`
b	`1011`
c	`1100`
d	`1101`
e	`1110`
f	`1111`

2. If you want the binary value, remove the leading zeros.

For example, in binary $301ac_{16}$ is `0011 0000 0001 1010 1100` or `0b110000000110101100`.

Note: if you are converting an `int` to hexadecimal, you will keep all 32 bits, which will be converted into eight hexadecimal characters. In the following table, remember that negative numbers are stored using two's complement.

`unsigned short`	50000	`0b1100001101010000`	`0xc350`
`short`	−23456	`0b1010010001100000`	`0xa460`
`unsigned int`	123456789	`0b00000111010110111100110100010101`	`0x075bcd15`
`int`	−42	`0b11111111111111111111111111010110`	`0xffffffd6`

When converting `0b00000111010110111100110100010101` to hexadecimal, it is easiest to put bars or spaces between each grouping of four bits:

```
0000|0111|0101|1011|1100|1101|0001|0101
```

and then convert each quartet of bits to the corresponding hexadecimal character: `0x075bcd15`.

When converting a hexadecimal number to binary, just replace each hexadecimal character with the corresponding four bits, so the hexadecimal number $123456789abcdef0_{16}$ is the binary number

0001|0010|0011|0100|0101|0110|0111|1000|1001|1010|1011|1100|1101|1110|1111|0000 or

1001000110100010101100111100010011010101111001101111011110000

without the three leading zeros.

4.3 Arithmetic in hexadecimal

If you compare the two hexadecimal numbers `3ff` and `400`, you will notice the second is `3ff + 1`. Thus, `407` becomes `3ff + 8`. You could learn to count and add in hexadecimal, but for the most part, it should be clear that the next three numbers after `1b3d` are `1b3e`, `1b3f` and `1b40`, while the three numbers before `83ab2` are `83ab1`, `83ab0` and `83aaf`.

The most you may do is consider the addresses of the entries of an array of `long` or `double`, where each entry is eight bytes. Thus, if the array is at `0xb0b88`, then the second and subsequent entries of that array are the addresses `0xb0b90`, `0xb0b98`, `0xb0ba0`, `0xb0ba8`, `0xb0bb0`, `0xb0bb8`, `0xb0bc0`, and so on. The entries at indices 14, 15 and 16 are `0xb0bf8`, `0xb0c00` and `0xb0c08`, respectively.

Not on the examination

5 Bonus: duodecimal

10 is a miserable number, as it is only divisible by two primes: 2 and 5. We use it as a base for the most anthropocentric possible choices: we (usually) have that many fingers on both our hands. In reality, it is a horrible choice: how often do we want to divide by five? On the other hand, dividing by three or four is much more difficult in base 10.
I would propose the most ideal system that works in the real world is base 12, or duodecimal system. The symbols would be 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, X and E. In this system, $10 \div 2 = 6$, $10 \div 3 = 4$ and $10 \div 4 = 3$. The addition table is as straight-forward as that for decimal, but the multiplication table is actually more pleasing:

+	1	2	3	4	5	6	7	8	9	X	E
1	2	3	4	5	6	7	8	9	X	E	10
2	3	4	5	6	7	8	9	X	E	10	11
3	4	5	6	7	8	9	X	E	10	11	12
4	5	6	7	8	9	X	E	10	11	12	13
5	6	7	8	9	X	E	10	11	12	13	14
6	7	8	9	X	E	10	11	12	13	14	15
7	8	9	X	E	10	11	12	13	54	15	16
8	9	X	E	10	11	12	43	14	15	16	17
9	X	E	10	11	12	13	14	15	16	17	18
X	E	10	11	12	13	14	15	16	17	18	19
E	10	11	12	13	14	15	16	17	18	19	1X

×	1	2	3	4	5	6	7	8	9	X	E
1	1	2	3	4	5	6	7	8	9	X	E
2	2	4	6	8	X	10	12	14	16	18	1X
3	3	6	9	10	13	16	19	20	23	26	29
4	4	8	10	14	18	20	24	28	30	34	38
5	5	X	13	18	21	26	2E	34	39	42	47
6	6	10	16	20	26	30	36	40	46	50	56
7	7	12	19	24	2E	36	41	48	53	5X	65
8	8	14	20	28	34	40	48	54	60	68	74
9	9	16	23	30	39	46	53	60	69	76	83
X	X	18	26	34	42	50	5X	68	76	84	92
E	E	1X	29	38	47	56	65	74	83	92	X1

You may appreciate that 8 out of 81 entries in the times table base 10 result in a multiple of 10, while here 17/121 end up as a multiple of 10_{12}. But this is for entertainment, only.
We already use a duodecimal system with hours on the clock and inches.