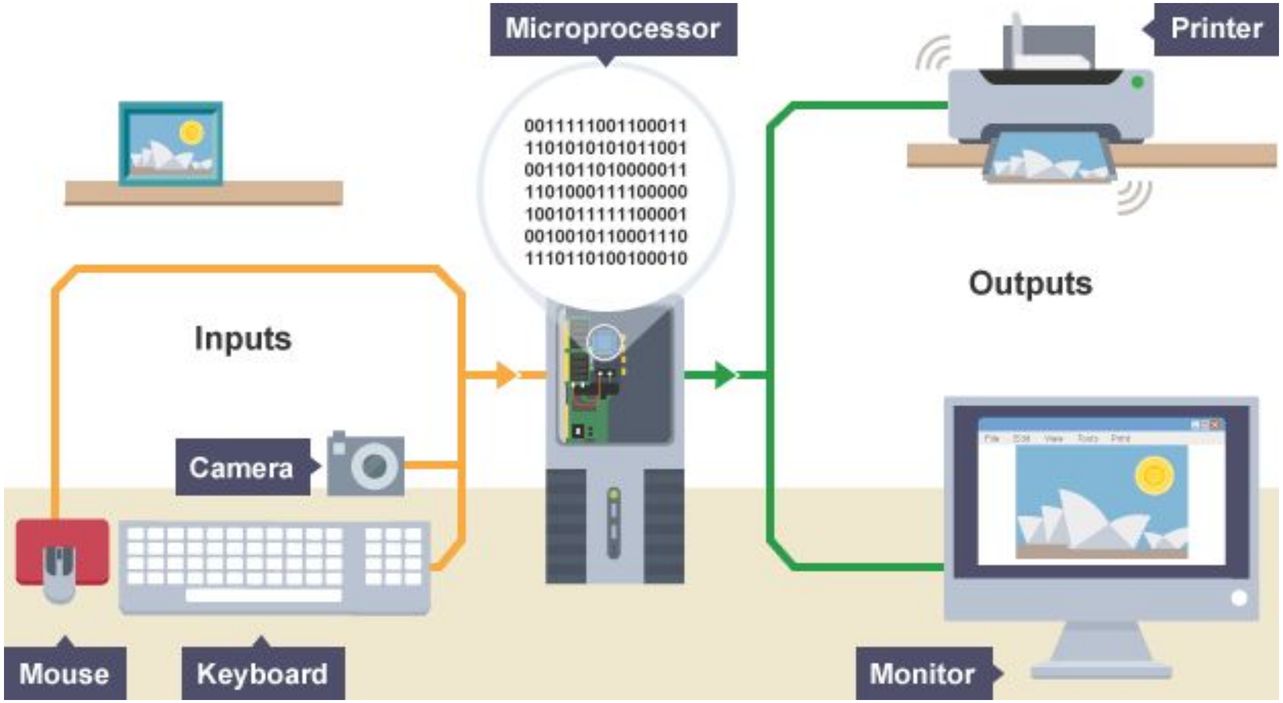
Binary and Hex Representations: Text, Images, Music, Videos, ….

Computers “communicate” in Base 2 (binary)



Early computers had switches made from vacuum tubes. The invention of the transistor in the 1950s started a revolution in creating smaller, faster and cheaper computers.

The reduction of decimal to binary does increase the length of the number, but this is more than made up for in the increase in speed, memory and utilisation.

Decimal System - Base 10

- Ten symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- These symbols are called digits
- We represent numbers with place value notation: 1’s place, 10’s place 100’s place, …
- 2573 = 2 X 1000 + 5 X 100 + 7 X 10 + 3 X 1

$$= 2 \text{ X } 10^3 + 5 \text{ X } 10^2 + 7 \text{ X } 10^1 + 3 \text{ X } 10^0$$

10^3	10^2	10^1	10^0

Conversions...

1	1	0	1
2^3	2^2	2^1	2^0

Binary - 1101 represents 13 in Base 10 (Base 2 to Base 10/Decimal)

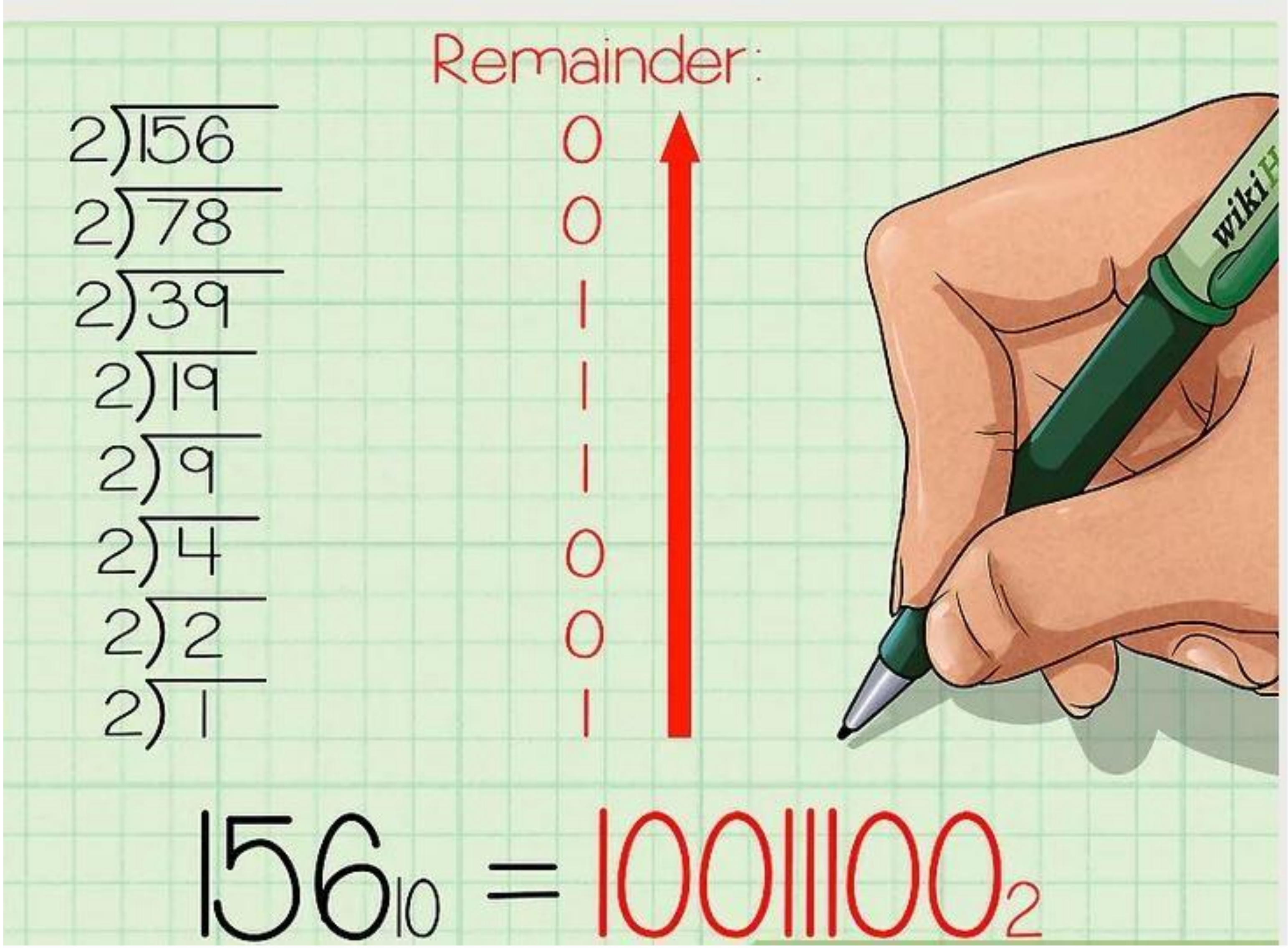
5	2	3
8^2	8^1	8^0

Octal- represents 339 in Base 10 (Base 8 to Base 10/Decimal)

2	5	4
6^2	6^1	6^0

Base 6 - represents 106 in Base 10

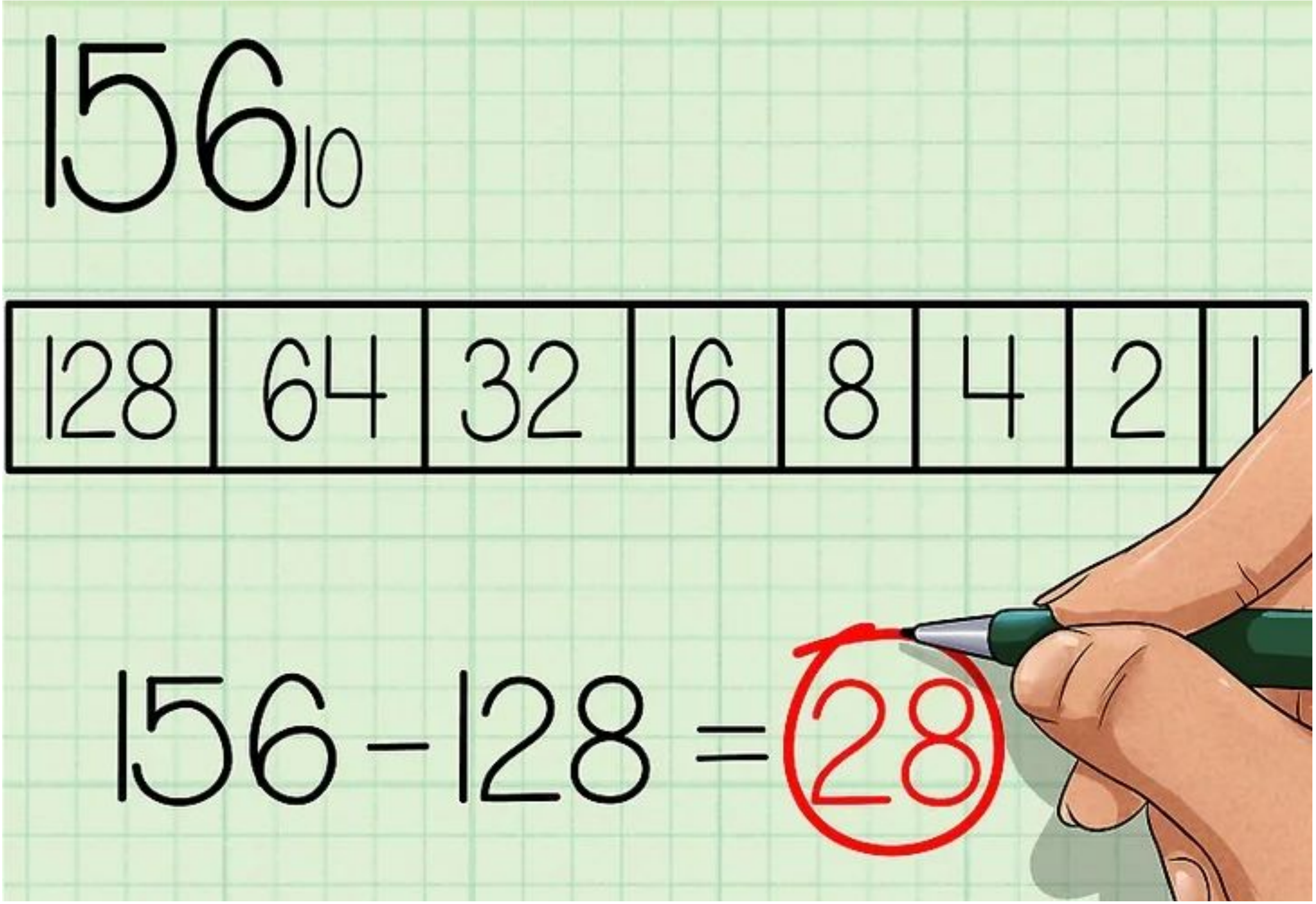
Conversion from decimal to binary:



Try It - Convert Dec to Bin

1. 11
2. 188
3. 204
4. 367
5. 1567

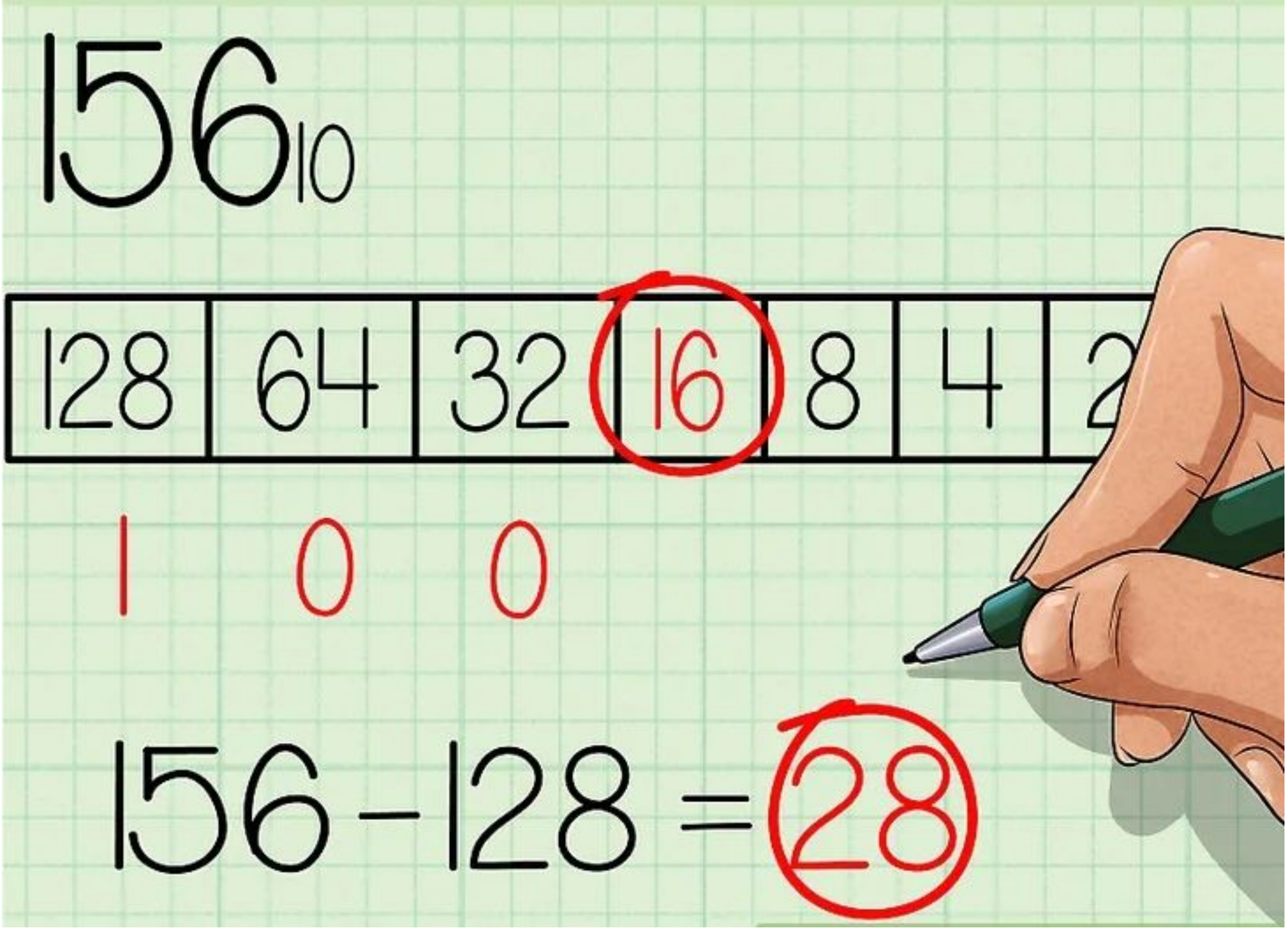
Conversion from decimal to binary: Method 2 - Descending powers of 2



156_{10}

128	64	32	16	8	4	2	1

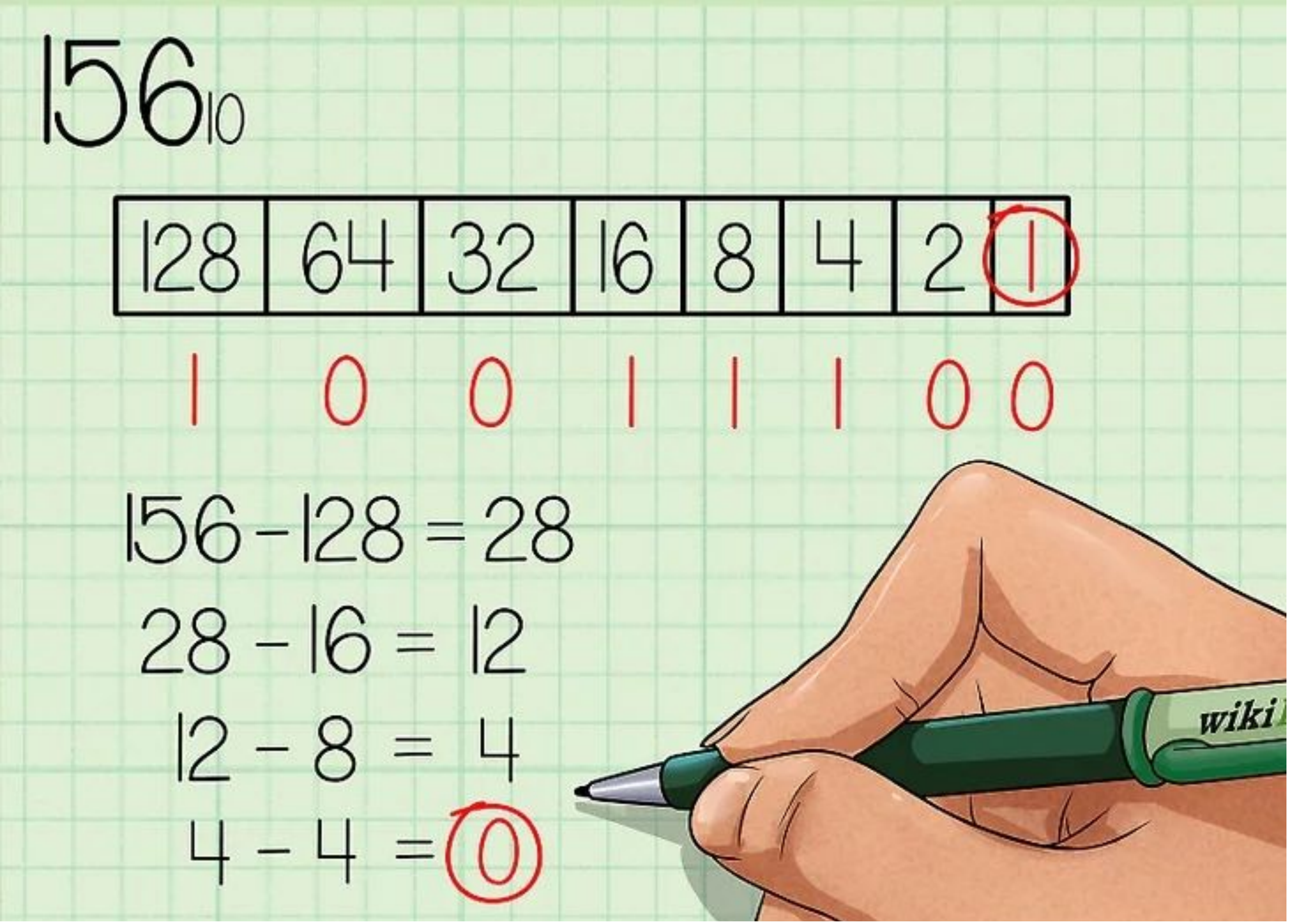
$$156 - 128 = \boxed{28}$$



156_{10}

128	64	32	16	8	4	2
1	0	0				

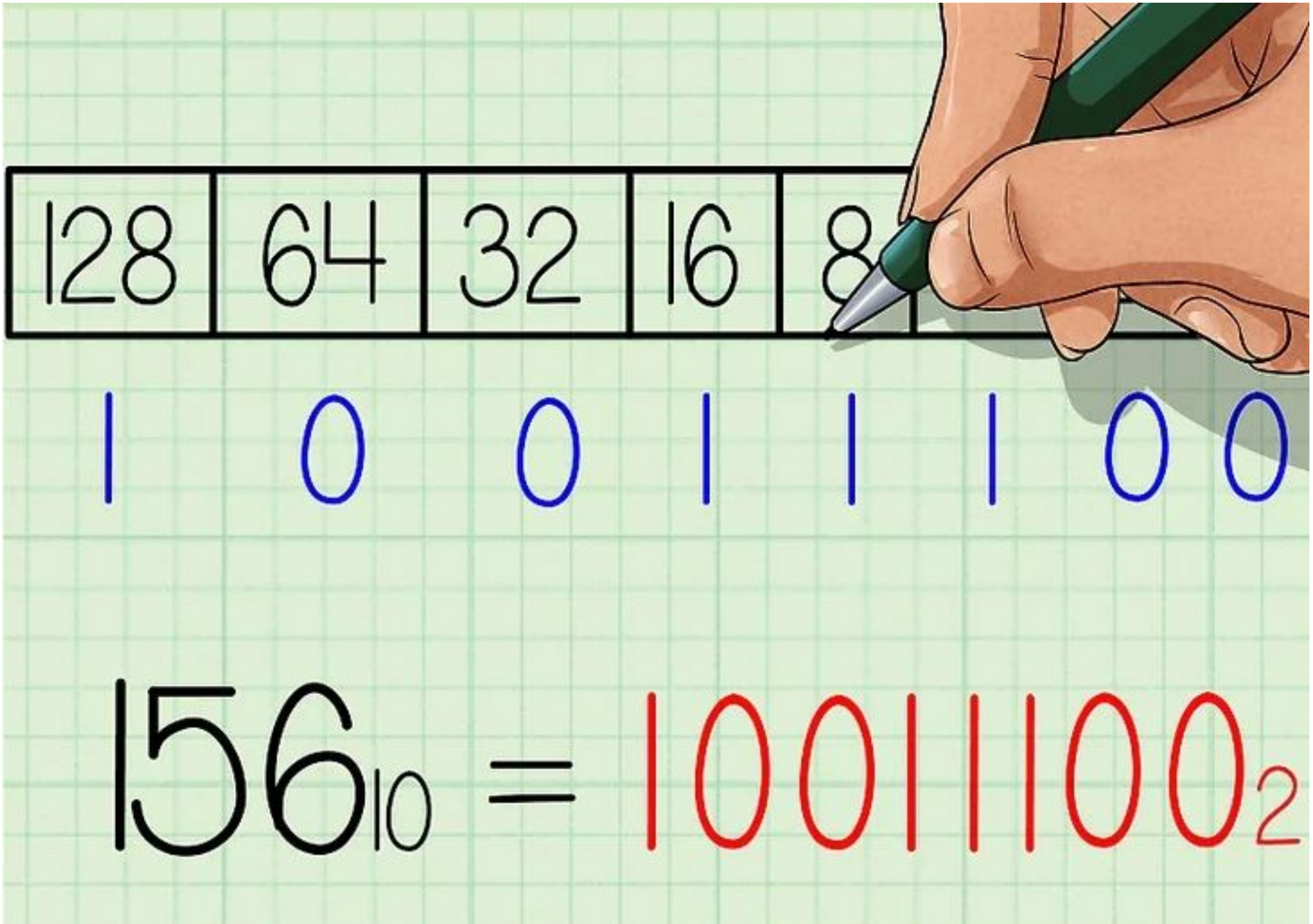
$156 - 128 = 28$



156_{10}

128	64	32	16	8	4	2	1
1	0	0	1	1	1	0	0

$$156-128=28$$
$$28-16=12$$
$$12-8=4$$
$$4-4=\boxed{0}$$



128	64	32	16	8			
1	0	0	1	1	1	0	0

$$156_{10} = 10011100_2$$

Convert Dec to Bin with Method 2

1. 12
2. 7
3. 2
4. 14
5. 6

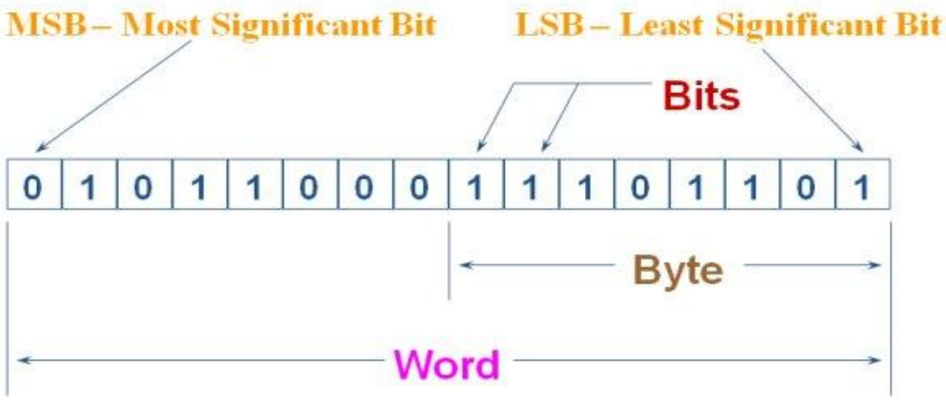
Convert Bin to Dec

1. 1111
2. 101100
3. 10111
4. 110
5. 101

Bits, Bytes, Words

Computers work with fixed sized bit sequences:

- Bit -> 0 or 1
- Byte = 8 bits
- Computer Word = several bytes
- Common word sizes on modern computers are
  - –32 bit words (4 bytes)
  - –64 bit words (8 bytes)



Some Useful Facts

- We can represent 10^n numbers with n digits
  - Ex: with two digits numbers 0, 1, …, 99
  - Note that $99 = 10^2 - 1$
- We can represent 2^n numbers with n bits
  - Ex: with 8 bits (one byte) : 0, 1, …, 255
  - Note that $255 = 2^8 - 1$
- Binary numbers that end in 0 are even; those that end in 1 are odd

“There are 10 kinds of people in the world: those who understand binary and those who don’t!”

Hexadecimal (base 16)

- Binary numbers can be impractical

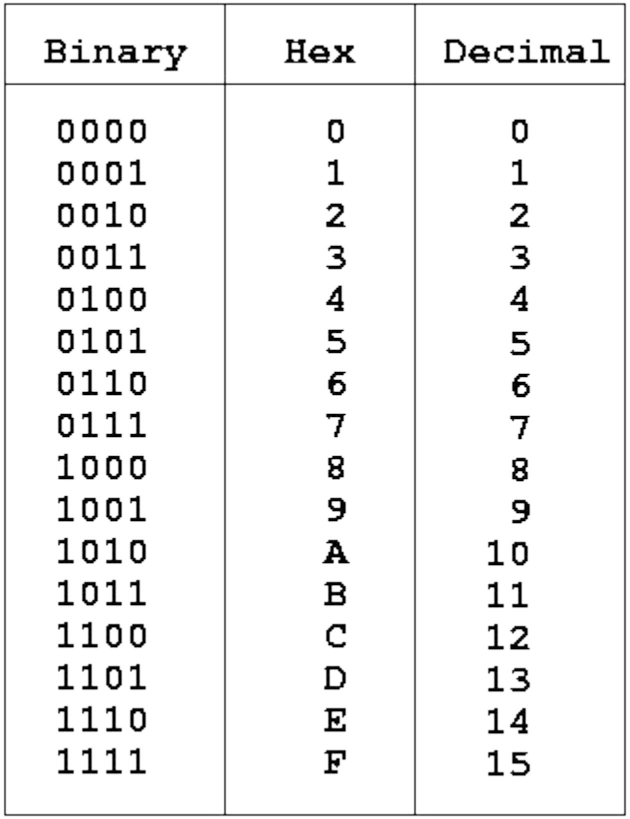
  1100101001000100011101011001100111010
- Base 16 is convenient because 16 numbers can be represented with 4 bits: 0000 to 1111 (also known as 0 to 15). They always start with 0x.
- But we don’t have enough digits to represent all 16 of these symbols

  So, how do we do that??

WE USE LETTERS !

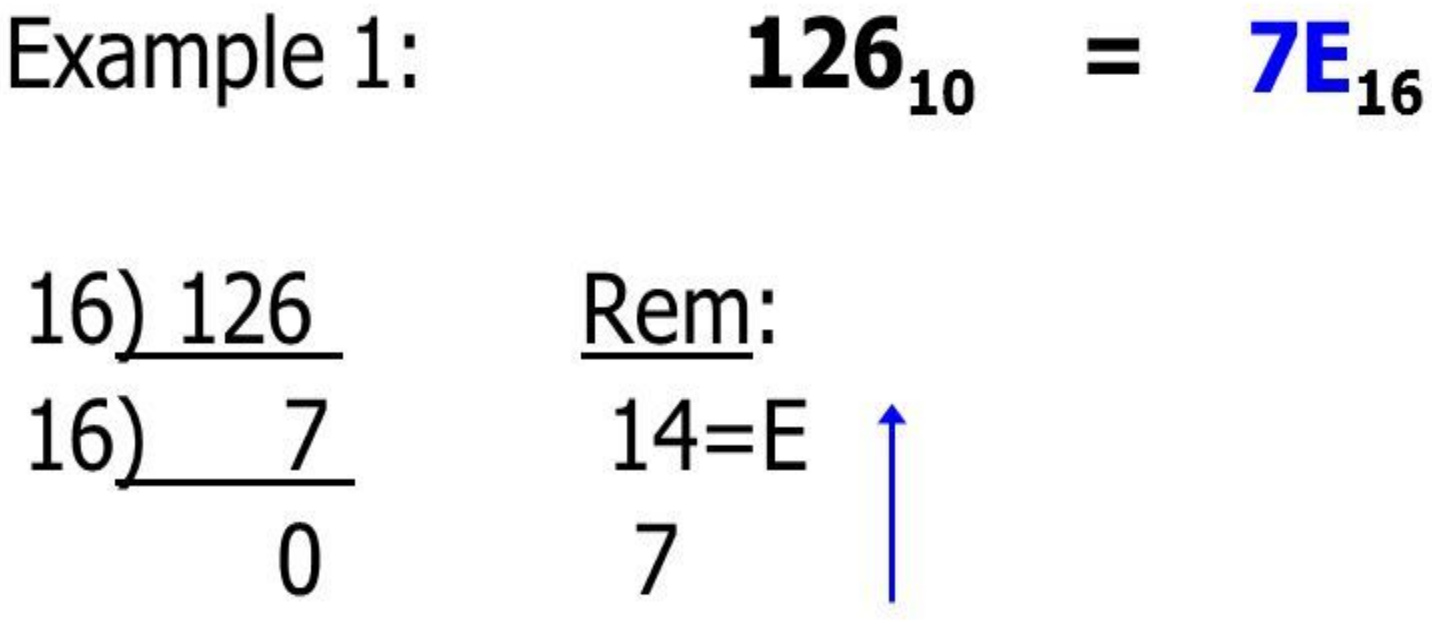
Sixteen symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

0x2AB3 = 2 X 16^3 + 10 X 16^2 + 11 X 16^1 + 3 = ? (Ask Python!)



Binary	Hex	Decimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

Conversion from Decimal to Hexadecimal



Example 1: 126_{10} = $7E_{16}$

```
16) 126          Rem:
16)   7           14=E
      0            7
```

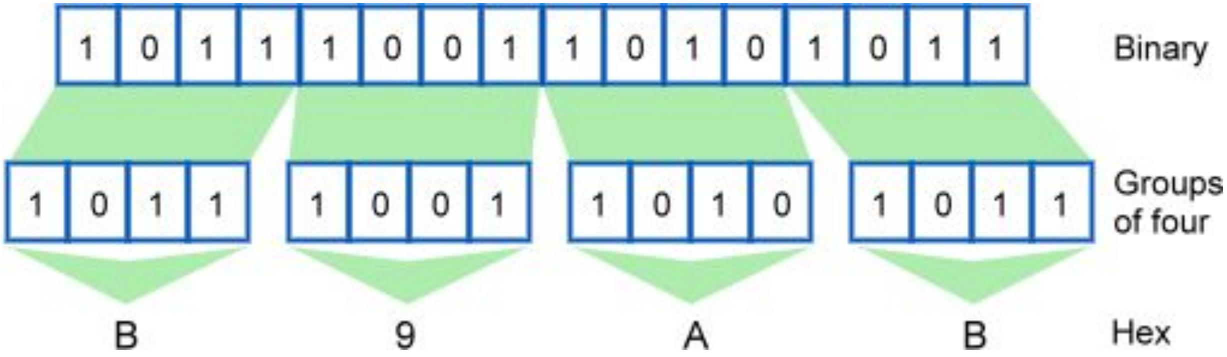
Convert Dec to Hex

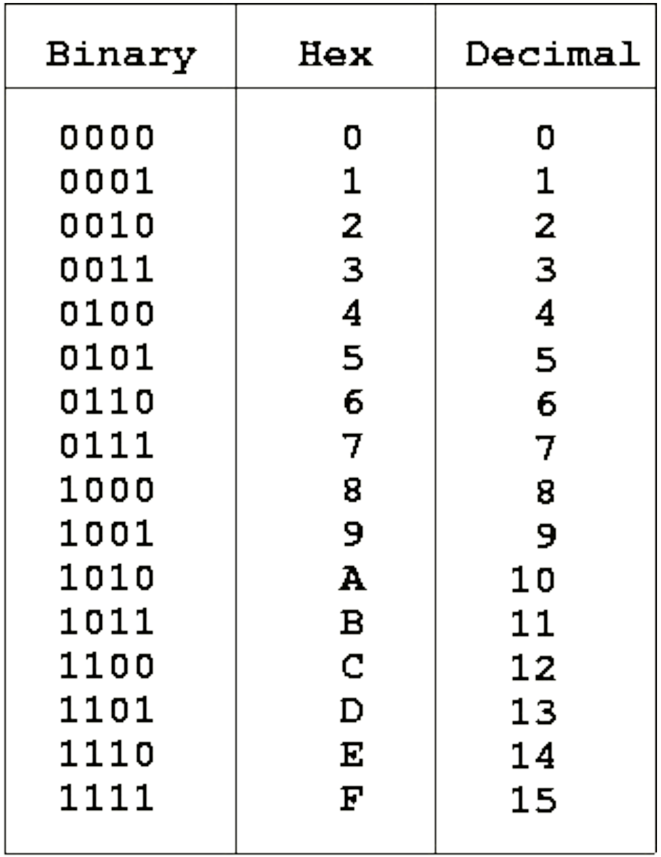
1. 15
2. 72
3. 10
4. 4
5. 109

Convert Hex to Dec

1. A
2. 8D
3. 7
4. 98
5. 2F

Binary to Hex: Easy!!!





Binary	Hex	Decimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

Step 1: Split into groups of 4 from the right

Step 2: Find each group of 4 in this table!

Convert Bin to Hex

1. 10 1110 1110 0111 0100 0111
2. 10010
3. 10110110101
4. 1110101010111010
5. 1101011110101

Convert Hex to Bin

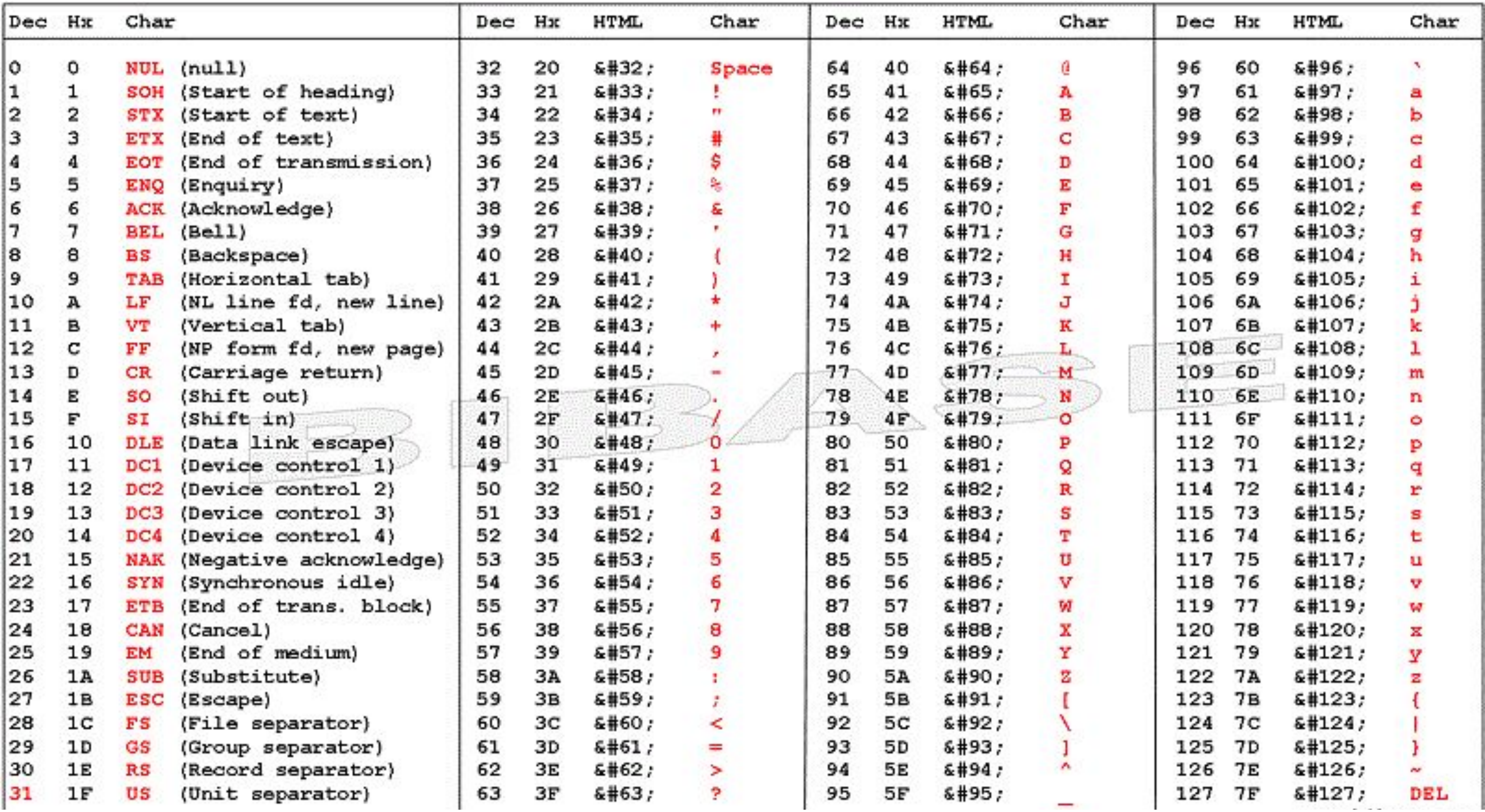
1. A1
2. 24
3. AD62FF
4. 985FA
5. B0987C

Importance of Binary

- Text, photos, music, video, get encoded into bytes that are represented in binary (and more easily readable in hex).
- For each type of media there are (several) standard representations
  - Text characters: ASCII, UTF-8, others
  - Images: jpg, png, bmp, …
  - Music: mp3, midi …

ASCII

- Each character (letter, digit, punctuation mark, space, tab, …) corresponds to a number.
- Original ASCII uses 7 bits to represent 128 different characters.
- Python uses an extension of ASCII called UTF-8 (but we'll just call it ASCII)
- Capital letters start at decimal 65 with 'A'
- 'A' : 65 : 01000001 : 41 (hex)
- 'B' : 66 : 01000010 :42 (hex)



Dec	Hx	Char	Dec	Hx	HTML	Char	Dec	Hx	HTML	Char	Dec	Hx	HTML	Char
0	0	NUL (null)	32	20	&#32;	Space	64	40	&#64;	@	96	60	&#96;	`
1	1	SOH (Start of heading)	33	21	&#33;	!	65	41	&#65;	A	97	61	&#97;	a
2	2	STX (Start of text)	34	22	&#34;	"	66	42	&#66;	B	98	62	&#98;	b
3	3	ETX (End of text)	35	23	&#35;	#	67	43	&#67;	C	99	63	&#99;	c
4	4	EOT (End of transmission)	36	24	&#36;	$	68	44	&#68;	D	100	64	&#100;	d
5	5	ENQ (Enquiry)	37	25	&#37;	%	69	45	&#69;	E	101	65	&#101;	e
6	6	ACK (Acknowledge)	38	26	&#38;	&	70	46	&#70;	F	102	66	&#102;	f
7	7	BEL (Bell)	39	27	&#39;	'	71	47	&#71;	G	103	67	&#103;	g
8	8	BS (Backspace)	40	28	&#40;	(	72	48	&#72;	H	104	68	&#104;	h
9	9	TAB (Horizontal tab)	41	29	&#41;	)	73	49	&#73;	I	105	69	&#105;	i
10	A	LF (NL line fd, new line)	42	2A	&#42;	*	74	4A	&#74;	J	106	6A	&#106;	j
11	B	VT (Vertical tab)	43	2B	&#43;	+	75	4B	&#75;	K	107	6B	&#107;	k
12	C	FF (NP form fd, new page)	44	2C	&#44;	,	76	4C	&#76;	L	108	6C	&#108;	l
13	D	CR (Carriage return)	45	2D	&#45;	-	77	4D	&#77;	M	109	6D	&#109;	m
14	E	SO (Shift out)	46	2E	&#46;	.	78	4E	&#78;	N	110	6E	&#110;	n
15	F	SI (Shift in)	47	2F	&#47;	/	79	4F	&#79;	O	111	6F	&#111;	o
16	10	DLE (Data link escape)	48	30	&#48;	0	80	50	&#80;	P	112	70	&#112;	p
17	11	DC1 (Device control 1)	49	31	&#49;	1	81	51	&#81;	Q	113	71	&#113;	q
18	12	DC2 (Device control 2)	50	32	&#50;	2	82	52	&#82;	R	114	72	&#114;	r
19	13	DC3 (Device control 3)	51	33	&#51;	3	83	53	&#83;	S	115	73	&#115;	s
20	14	DC4 (Device control 4)	52	34	&#52;	4	84	54	&#84;	T	116	74	&#116;	t
21	15	NAK (Negative acknowledge)	53	35	&#53;	5	85	55	&#85;	U	117	75	&#117;	u
22	16	SYN (Synchronous idle)	54	36	&#54;	6	86	56	&#86;	V	118	76	&#118;	v
23	17	ETB (End of trans. block)	55	37	&#55;	7	87	57	&#87;	W	119	77	&#119;	w
24	18	CAN (Cancel)	56	38	&#56;	8	88	58	&#88;	X	120	78	&#120;	x
25	19	EM (End of medium)	57	39	&#57;	9	89	59	&#89;	Y	121	79	&#121;	y
26	1A	SUB (Substitute)	58	3A	&#58;	:	90	5A	&#90;	Z	122	7A	&#122;	z
27	1B	ESC (Escape)	59	3B	&#59;	;	91	5B	&#91;	[	123	7B	&#123;	{
28	1C	FS (File separator)	60	3C	&#60;	<	92	5C	&#92;	\	124	7C	&#124;	|
29	1D	GS (Group separator)	61	3D	&#61;	=	93	5D	&#93;	]	125	7D	&#125;	}
30	1E	RS (Record separator)	62	3E	&#62;	>	94	5E	&#94;	^	126	7E	&#126;	~
31	1F	US (Unit separator)	63	3F	&#63;	?	95	5F	&#95;	_	127	7F	&#127;	DEL

Conversion Using Python

Binary to Hexadecimal

```
binary_string = "1010"

decimal_representation = int(binary_string, 2)

hexadecimal_string = hex(decimal_representation)

print(hexadecimal_string)
```

THIS IS HOW YOU CONVERT A BINARY TO INT

Conversion Using Python

Decimal to Binary

```
binary = int(input('enter a number: '))

print(bin(binary))
```

REMEMBER THESE HACKS:

1. ANY NUMBER SYSTEM CONVERTING TO DECIMAL ->

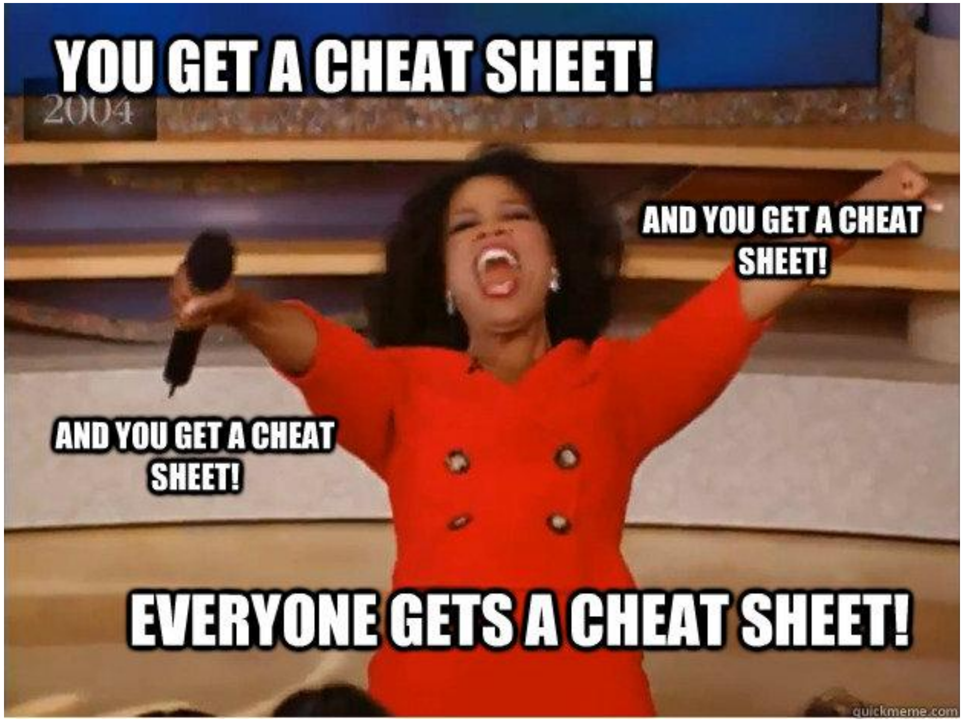
   Write powers of the base from right to left, multiply with digits and add

2. DECIMAL TO ANY NUMBER SYSTEM ->

   Repeatedly divide by the base of the number system, write the remainders from bottom to top

3. BINARY TO HEX->

   Groups of 4 bits and then just write in hex!


YOU GET A CHEAT SHEET!
2004
AND YOU GET A CHEAT SHEET!
AND YOU GET A CHEAT SHEET!
EVERYONE GETS A CHEAT SHEET!
quickmeme.com