AP CSP

SouthLake Christian Academy
2022-2023

Text Data

ASCII

Apart from numbers, we use many different types of data. If we are writing an essay in Microsoft Word or Google Docs, our data is in the form of *text*, or alphabetical letters.

Here's the problem: if computers only understand binary, how can we represent text?

We solve this problem by mapping letters (or **characters**) to numbers. Here, humans are interpreting letters as numbers, and vice versa – the computer doesn't care about letters; in fact, letters don't really exist for a computer. Computers will only see the binary mapping of the letter.

An obvious mapping is to say `A` → `0`, `B` → `1`, `C` → `2`, and so on.

Instead of this trivial mapping, however, we often use American Standard Code for Information Interchange, or **ASCII**. ASCII dedicates 7 bits to represent one character (the extended ASCII dedicates 8 bits, for more characters). Characters include uppercase letters, lowercase letters, punctuation, and special functions.

As we can see below, the 7-bit ASCII allows for 128 different characters:

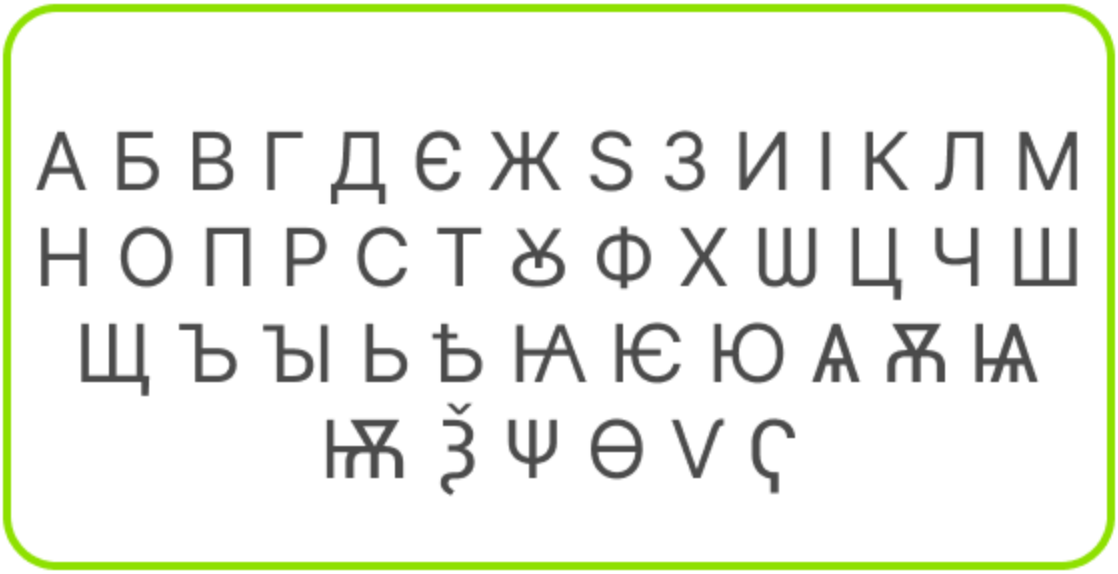
dec	hex	oct	char	dec	hex	oct	char	dec	hex	oct	char	dec	hex	oct	char
0	0	000	NULL	32	20	040	space	64	40	100	@	96	60	140	`
1	1	001	SOH	33	21	041	!	65	41	101	A	97	61	141	a
2	2	002	STX	34	22	042	"	66	42	102	B	98	62	142	b
3	3	003	ETX	35	23	043	#	67	43	103	C	99	63	143	c
4	4	004	EOT	36	24	044	$	68	44	104	D	100	64	144	d
5	5	005	ENQ	37	25	045	%	69	45	105	E	101	65	145	e
6	6	006	ACK	38	26	046	&	70	46	106	F	102	66	146	f
7	7	007	BEL	39	27	047	'	71	47	107	G	103	67	147	g
8	8	010	BS	40	28	050	(	72	48	110	H	104	68	150	h
9	9	011	TAB	41	29	051	)	73	49	111	I	105	69	151	i
10	a	012	LF	42	2a	052	*	74	4a	112	J	106	6a	152	j
11	b	013	VT	43	2b	053	+	75	4b	113	K	107	6b	153	k
12	c	014	FF	44	2c	054	,	76	4c	114	L	108	6c	154	l
13	d	015	CR	45	2d	055	-	77	4d	115	M	109	6d	155	m
14	e	016	SO	46	2e	056	.	78	4e	116	N	110	6e	156	n
15	f	017	SI	47	2f	057	/	79	4f	117	O	111	6f	157	o
16	10	020	DLE	48	30	060	0	80	50	120	P	112	70	160	p
17	11	021	DC1	49	31	061	1	81	51	121	Q	113	71	161	q
18	12	022	DC2	50	32	062	2	82	52	122	R	114	72	162	r
19	13	023	DC3	51	33	063	3	83	53	123	S	115	73	163	s
20	14	024	DC4	52	34	064	4	84	54	124	T	116	74	164	t
21	15	025	NAK	53	35	065	5	85	55	125	U	117	75	165	u
22	16	026	SYN	54	36	066	6	86	56	126	V	118	76	166	v
23	17	027	ETB	55	37	067	7	87	57	127	W	119	77	167	w
24	18	030	CAN	56	38	070	8	88	58	130	X	120	78	170	x
25	19	031	EM	57	39	071	9	89	59	131	Y	121	79	171	y
26	1a	032	SUB	58	3a	072	:	90	5a	132	Z	122	7a	172	z
27	1b	033	ESC	59	3b	073	;	91	5b	133	[	123	7b	173	{
28	1c	034	FS	60	3c	074	<	92	5c	134	\	124	7c	174	|
29	1d	035	GS	61	3d	075	=	93	5d	135	]	125	7d	175	}
30	1e	036	RS	62	3e	076	>	94	5e	136	^	126	7e	176	~
31	1f	037	US	63	3f	077	?	95	5f	137	_	127	7f	177	DEL

www.alpharithms.com

So when we hit the Tab key, the computer really sees the number `0b0001001`, or `9`. When we hit the `A` key, the computer really sees `0b1000001`.

Therefore, the binary sequence `0b10010001101001` can be interpreseted as the message “Hi” (`H` → `0b1001000`; `i` → `0b1101001`).

ASCII is a great system and widely used; however, it maps numbers to only characters from the Latin alphabet. What are all the people writing in Cyrillic or Arabic scripts supposed to do?



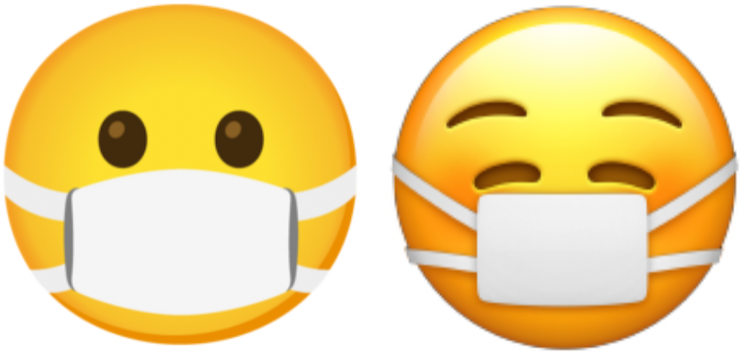
Unicode

Unicode is another way to map numbers to character. While ASCII dedicates at most a byte for each character, Unicode sets aside **32 bits**, or 4 bytes, for one character. That means Unicode can represent 4294967296 characters total – remember, ASCII can only represent 128 characters.

Unicode can represent characters from a variety of alphabets as well as emojis.

Emojis

To represent emojis, Unicode maps *descriptions* to numbers. For example, the description "face with medical mask" is mapped to `0b11110000100111111001100010110111`. Whenever you text your friend an emoji, what you're actually sending is the string of bits. Your friend's phone receives this binary number, looks up the description "face with medical mask", and then displays the image associated with that description:



Key Takeaway

Computers do not interpret data; they just consume binary numbers. Humans interpret binary as either decimal numbers or text. The value of the data does not have to change, but its representation can change wildly, depending on context.

For example, `0b10010001101001` can be interpreted as *either* "Hi" or `9321`, depending if we are in Microsoft Word or the Calculator App.

Extra Resources

- Slides (https://docs.google.com/viewer?url=https://github.com/APCSP-SLCA/slides

/raw/main/ascii/slides.pdf)