Chapter 2a

Number System

Decimal – 0-9
Hexadecimal 0-9, A,B, C, D, E, F
Octal – 0-7
Binary - 0,1

Decimal	0	1	2	3	4	5	6	7	8	9					
Hexadecimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E
Binary	0	1													

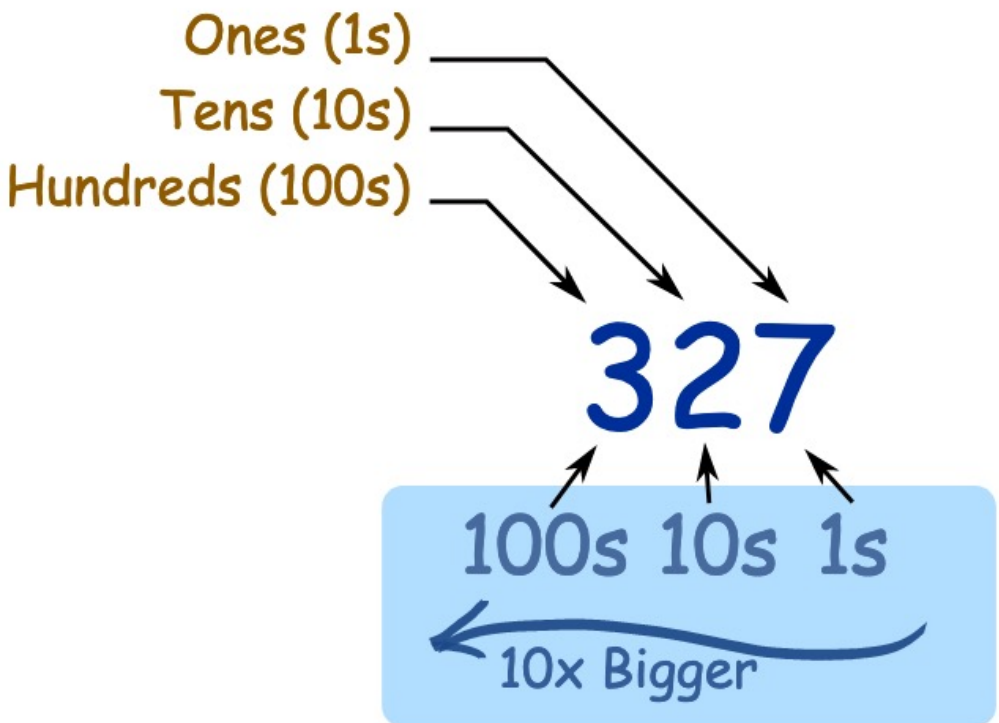
Definition of

Decimal Number System

more ...

The number system we use every day, based on 10 digits (0,1,2,3,4,5,6,7,8,9).

Position is important, with the first position being units, then next on the left being tens, then hundreds and so on.



Hexadecimal

16 Different Values

There are **16** Hexadecimal digits. They are the same as the decimal digits up to 9, but then there are the letters A, B, C, D, E and F in place of the decimal numbers 10 to 15:

Hexadecimal:	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Decimal:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

So a single Hexadecimal digit can show 16 different values instead of the normal 10.

Octal

An Octal Number uses only these 8 digits: 0, 1, 2, 3, 4, 5, 6 and 7

Examples:

- **7** in Octal equals 7 in the Decimal Number System
- **10** in Octal equals 8 in the Decimal Number System
- **11** in Octal equals 9 in the Decimal Number System

...

- **167** in Octal equals 119 in the Decimal Number System

Binary Number

A Binary Number is made up of only **0**s and **1**s.

110100

Example of a Binary Number

There is no 2, 3, 4, 5, 6, 7, 8 or 9 in Binary!

Base 10 (Denary)

0 1 2 3 4 5 6 7 8 9

digits

1000

Decimal System - Poistional Number System

10000	1000	100	10	1
0	0	5	3	7

$$(5 * 100) + (3 * 10) + (7 * 1)$$

Base 2 (Binary)

0 1

Electrial Signal – On & Off

bits

11111111

Binary System - Poistional Number System

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

(1 * 32) + (1 * 16) + (1 * 4) + (1 * 1) = 53

Largest Binary – 8 Bits

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

(1 * 128) + (1 * 64) + (1 * 32) + (1 * 16) + (1 * 8) + (1 * 4) + (1 * 2) + (1 * 1) = 255

Smallest Binary – 8 Bits

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

Convert the 8 bit binary number 00011010 into denary

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

16 + 8 + 2 = 26

$00011010_2 = 26_{10}$

Excerise - 1

00000101	01111111
01111001	10000000
01010100	10101010
01101101	11000011

Excerise - 2

17

44

132

101

Excerise - 3

What is the biggest denary number that you can represent in binary using 8 bits?

How many different denary numbers can you represent in binary using 8 bits?

The ASCII and Unicode Character Sets

List of characters a computer can understand

American Standard Code for Information Interchange (ASCII)

Number	Character		Number	Character	Number	Character	Number	Character
0	NULL	NULL CHARACTER	32	SPACE	64	@	96	`
1	SOH	START OF HEADING	33	!	65	A	97	a
2	STX	START OF TEXT	34	"	66	B	98	b
3	ETX	END OF TEXT	35	#	67	C	99	c
4	EOT	END OF TRANSMISSION	36	$	68	D	100	d
5	ENQ	ENQUIRY	37	%	69	E	101	e
6	ACK	ACKNOWLEDGE	38	&	70	F	102	f
7	BEL	BELL	39	'	71	G	103	g
8	BS	BACKSPACE	40	(	72	H	104	h
9	HT	HORIZONTAL TAB	41	)	73	I	105	i
10	LF	LINE FEED	42	*	74	J	106	j
11	VT	VERTICAL TAB	43	+	75	K	107	k
12	FF	FORM FEED	44	,	76	L	108	l
13	CR	CARRIAGE RETURN	45	-	77	M	109	m
14	SO	SHIFT OUT	46	.	78	N	110	n
15	SI	SHIFT IN	47	/	79	O	111	o
16	DLE	DATALINK ESCAPE	48	0	80	P	112	p
17	DC1	DEVICE CONTROL 1	49	1	81	Q	113	q
18	DC2	DEVICE CONTROL 2	50	2	82	R	114	r
19	DC3	DEVICE CONTROL 3	51	3	83	S	115	s
20	DC4	DEVICE CONTROL 4	52	4	84	T	116	t
21	NAK	NEGATIVE ACKNOWLEDGE	53	5	85	U	117	u
22	SYN	SYNCHRONOUS IDLE	54	6	86	V	118	v
23	ETB	END OF TRANS, BLOCK	55	7	87	W	119	w
24	CAN	CANCEL	56	8	88	X	120	x
25	EM	END OF MEDIUM	57	9	89	Y	121	y
26	SUB	SUSTITUTE	58	:	90	Z	122	z
27	ESC	ESCAPE	59	;	91	[	123	{
28	FS	FILE SEPARATOR	60	<	92	\	124	|
29	GS	GROUP SEPARATOR	61	=	93	]	125	}
30	RS	RECORD SEPARATOR	62	>	94	^	126	~
31	US	UNIT SEPARATOR	63	?	95	_	127	DEL

American Standard Code for Information Interchange (ASCII)

Decimal	Hex	Binary	Character		Decimal	Hex	Binary	Character	Decimal	Hex	Binary	Character	Decimal	Hex	Binary	Character
0	00	0000000	NULL	NULL CHARACTER	32	20	0100000	SPACE	64	40	1000000	@	96	60	1100000	`
1	01	0000001	SOH	START OF HEADING	33	21	0100001	!	65	41	1000001	A	97	61	1100001	a
2	02	0000010	STX	START OF TEXT	34	22	0100010	"	66	42	1000010	B	98	62	1100010	b
3	03	0000011	ETX	END OF TEXT	35	23	0100011	#	67	43	1000011	C	99	63	1100011	c
4	04	0000100	EOT	END OF TRANSMISSION	36	24	0100100	$	68	44	1000100	D	100	64	1100100	d
5	05	0000101	ENQ	ENQUIRY	37	25	0100101	%	69	45	1000101	E	101	65	1100101	e
6	06	0000110	ACK	ACKNOWLEDGE	38	26	0100110	&	70	46	1000110	F	102	66	1100110	f
7	07	0000111	BEL	BELL	39	27	0100111	'	71	47	1000111	G	103	67	1100111	g
8	08	0001000	BS	BACKSPACE	40	28	0101000	(	72	48	1001000	H	104	68	1101000	h
9	09	0001001	HT	HORIZONTAL TAB	41	29	0101001	)	73	49	1001001	I	105	69	1101001	i
10	0A	0001010	LF	LINE FEED	42	2A	0101010	*	74	4A	1001010	J	106	6A	1101010	j
11	0B	0001011	VT	VERTICAL TAB	43	2B	0101011	+	75	4B	1001011	K	107	6B	1101011	k
12	0C	0001100	FF	FORM FEED	44	2C	0101100	,	76	4C	1001100	L	108	6C	1101100	l
13	0D	0001101	CR	CARRIAGE RETURN	45	2D	0101101	-	77	4D	1001101	M	109	6D	1101101	m
14	0E	0001110	SO	SHIFT OUT	46	2E	0101110	.	78	4E	1001110	N	110	6E	1101110	n
15	0F	0001111	SI	SHIFT IN	47	2F	0101111	/	79	4F	1001111	O	111	6F	1101111	o
16	10	0010000	DLE	DATALINK ESCAPE	48	30	0110000	0	80	50	1010000	P	112	70	1110000	p
17	11	0010001	DC1	DEVICE CONTROL 1	49	31	0110001	1	81	51	1010001	Q	113	71	1110001	q
18	12	0010010	DC2	DEVICE CONTROL 2	50	32	0110010	2	82	52	1010010	R	114	72	1110010	r
19	13	0010011	DC3	DEVICE CONTROL 3	51	33	0110011	3	83	53	1010011	S	115	73	1110011	s
20	14	0010100	DC4	DEVICE CONTROL 4	52	34	0110100	4	84	54	1010100	T	116	74	1110100	t
21	15	0010101	NAK	NEGATIVE ACKNOWLEDGE	53	35	0110101	5	85	55	1010101	U	117	75	1110101	u
22	16	0010110	SYN	SYNCHRONOUS IDLE	54	36	0110110	6	86	56	1010110	V	118	76	1110110	v
23	17	0010111	ETB	END OF TRANS, BLOCK	55	37	0110111	7	87	57	1010111	W	119	77	1110111	w
24	18	0011000	CAN	CANCEL	56	38	0111000	8	88	58	1011000	X	120	78	1111000	x
25	19	0011001	EM	END OF MEDIUM	57	39	0111001	9	89	59	1011001	Y	121	79	1111001	y
26	1A	0011010	SUB	SUSTITUTE	58	3A	0111010	:	90	5A	1011010	Z	122	7A	1111010	z
27	1B	0011011	ESC	ESCAPE	59	3B	0111011	;	91	5B	1011011	[	123	7B	1111011	{
28	1C	0011100	FS	FILE SEPARATOR	60	3C	0111100	<	92	5C	1011100	\	124	7C	1111100	|
29	1D	0011101	GS	GROUP SEPARATOR	61	3D	0111101	=	93	5D	1011101	]	125	7D	1111101	}
30	1E	0011110	RS	RECORD SEPARATOR	62	3E	0111110	>	94	5E	1011110	^	126	7E	1111110	~
31	1F	0011111	US	UNIT SEPARATOR	63	3F	0111111	?	95	5F	1011111	_	127	7F	1111111	DEL

American Standard Code for Information Interchange (ASCII)

Decimal	Hex	Binary	Character		Decimal	Hex	Binary	Character	Decimal	Hex	Binary	Character	Decimal	Hex	Binary	Character
0	00	0000000	NULL	NULL CHARACTER	32	20	0100000	SPACE	64	40	1000000	@	96	60	1100000	`
1	01	0000001	SOH	START OF HEADING	33	21	0100001	!	65	41	1000001	A	97	61	1100001	a
2	02	0000010	STX	START OF TEXT	34	22	0100010	"	66	42	1000010	B	98	62	1100010	b
3	03	0000011	ETX	END OF TEXT	35	23	0100011	#	67	43	1000011	C	99	63	1100011	c
4	04	0000100	EOT	END OF TRANSMISSION	36	24	0100100	$	68	44	1000100	D	100	64	1100100	d
5	05	0000101	ENQ	ENQUIRY	37	25	0100101	%	69	45	1000101	E	101	65	1100101	e
6	06	0000110	ACK	ACKNOWLEDGE	38	26	0100110	&	70	46	1000110	F	102	66	1100110	f
7	07	0000111	BEL	BELL	39	27	0100111	'	71	47	1000111	G	103	67	1100111	g
8	08	0001000	BS	BACKSPACE	40	28	0101000	(	72	48	1001000	H	104	68	1101000	h
9	09	0001001	HT	HORIZONTAL TAB	41	29	0101001	)	73	49	1001001	I	105	69	1101001	i
10	0A	0001010	LF	LINE FEED	42	2A	0101010	*	74	4A	1001010	J	106	6A	1101010	j
11	0B	0001011	VT	VERTICAL TAB	43	2B	0101011	+	75	4B	1001011	K	107	6B	1101011	k
12	0C	0001100	FF	FORM FEED	44	2C	0101100	,	76	4C	1001100	L	108	6C	1101100	l
13	0D	0001101	CR	CARRIAGE RETURN	45	2D	0101101	-	77	4D	1001101	M	109	6D	1101101	m
14	0E	0001110	SO	SHIFT OUT	46	2E	0101110	.	78	4E	1001110	N	110	6E	1101110	n
15	0F	0001111	SI	SHIFT IN	47	2F	0101111	/	79	4F	1001111	O	111	6F	1101111	o
16	10	0010000	DLE	DATALINK ESCAPE	48	30	0110000	0	80	50	1010000	P	112	70	1110000	p
17	11	0010001	DC1	DEVICE CONTROL 1	49	31	0110001	1	81	51	1010001	Q	113	71	1110001	q
18	12	0010010	DC2	DEVICE CONTROL 2	50	32	0110010	2	82	52	1010010	R	114	72	1110010	r
19	13	0010011	DC3	DEVICE CONTROL 3	51	33	0110011	3	83	53	1010011	S	115	73	1110011	s
20	14	0010100	DC4	DEVICE CONTROL 4	52	34	0110100	4	84	54	1010100	T	116	74	1110100	t
21	15	0010101	NAK	NEGATIVE ACKNOWLEDGE	53	35	0110101	5	85	55	1010101	U	117	75	1110101	u
22	16	0010110	SYN	SYNCHRONOUS IDLE	54	36	0110110	6	86	56	1010110	V	118	76	1110110	v
23	17	0010111	ETB	END OF TRANS, BLOCK	55	37	0110111	7	87	57	1010111	W	119	77	1110111	w
24	18	0011000	CAN	CANCEL	56	38	0111000	8	88	58	1011000	X	120	78	1111000	x
25	19	0011001	EM	END OF MEDIUM	57	39	0111001	9	89	59	1011001	Y	121	79	1111001	y
26	1A	0011010	SUB	SUSTITUTE	58	3A	0111010	:	90	5A	1011010	Z	122	7A	1111010	z
27	1B	0011011	ESC	ESCAPE	59	3B	0111011	;	91	5B	1011011	[	123	7B	1111011	{
28	1C	0011100	FS	FILE SEPARATOR	60	3C	0111100	<	92	5C	1011100	\	124	7C	1111100	|
29	1D	0011101	GS	GROUP SEPARATOR	61	3D	0111101	=	93	5D	1011101	]	125	7D	1111101	}
30	1E	0011110	RS	RECORD SEPARATOR	62	3E	0111110	>	94	5E	1011110	^	126	7E	1111110	~
31	1F	0011111	US	UNIT SEPARATOR	63	3F	0111111	?	95	5F	1011111	_	127	7F	1111111	DEL

$2^7 = 128$

1 1 1 1 1 1 1

American Standard Code for Information Interchange (ASCII)

Decimal	Hex	Binary	Character		Decimal	Hex	Binary	Character	Decimal	Hex	Binary	Character	Decimal	Hex	Binary	Character
0	00	0000000	NULL	NULL CHARACTER	32	20	0100000	SPACE	64	40	1000000	@	96	60	1100000	`
1	01	0000001	SOH	START OF HEADING	33	21	0100001	!	65	41	1000001	A	97	61	1100001	a
2	02	0000010	STX	START OF TEXT	34	22	0100010	"	66	42	1000010	B	98	62	1100010	b
3	03	0000011	ETX	END OF TEXT	35	23	0100011	#	67	43	1000011	C	99	63	1100011	c
4	04	0000100	EOT	END OF TRANSMISSION	36	24	0100100	$	68	44	1000100	D	100	64	1100100	d
5	05	0000101	ENQ	ENQUIRY	37	25	0100101	%	69	45	1000101	E	101	65	1100101	e
6	06	0000110	ACK	ACKNOWLEDGE	38	26	0100110	&	70	46	1000110	F	102	66	1100110	f
7	07	0000111	BEL	BELL	39	27	0100111	'	71	47	1000111	G	103	67	1100111	g
8	08	0001000	BS	BACKSPACE	40	28	0101000	(	72	48	1001000	H	104	68	1101000	h
9	09	0001001	HT	HORIZONTAL TAB	41	29	0101001	)	73	49	1001001	I	105	69	1101001	i
10	0A	0001010	LF	LINE FEED	42	2A	0101010	*	74	4A	1001010	J	106	6A	1101010	j
11	0B	0001011	VT	VERTICAL TAB	43	2B	0101011	+	75	4B	1001011	K	107	6B	1101011	k
12	0C	0001100	FF	FORM FEED	44	2C	0101100	,	76	4C	1001100	L	108	6C	1101100	l
13	0D	0001101	CR	CARRIAGE RETURN	45	2D	0101101	-	77	4D	1001101	M	109	6D	1101101	m
14	0E	0001110	SO	SHIFT OUT	46	2E	0101110	.	78	4E	1001110	N	110	6E	1101110	n
15	0F	0001111	SI	SHIFT IN	47	2F	0101111	/	79	4F	1001111	O	111	6F	1101111	o
16	10	0010000	DLE	DATALINK ESCAPE	48	30	0110000	0	80	50	1010000	P	112	70	1110000	p
17	11	0010001	DC1	DEVICE CONTROL 1	49	31	0110001	1	81	51	1010001	Q	113	71	1110001	q
18	12	0010010	DC2	DEVICE CONTROL 2	50	32	0110010	2	82	52	1010010	R	114	72	1110010	r
19	13	0010011	DC3	DEVICE CONTROL 3	51	33	0110011	3	83	53	1010011	S	115	73	1110011	s
20	14	0010100	DC4	DEVICE CONTROL 4	52	34	0110100	4	84	54	1010100	T	116	74	1110100	t
21	15	0010101	NAK	NEGATIVE ACKNOWLEDGE	53	35	0110101	5	85	55	1010101	U	117	75	1110101	u
22	16	0010110	SYN	SYNCHRONOUS IDLE	54	36	0110110	6	86	56	1010110	V	118	76	1110110	v
23	17	0010111	ETB	END OF TRANS, BLOCK	55	37	0110111	7	87	57	1010111	W	119	77	1110111	w
24	18	0011000	CAN	CANCEL	56	38	0111000	8	88	58	1011000	X	120	78	1111000	x
25	19	0011001	EM	END OF MEDIUM	57	39	0111001	9	89	59	1011001	Y	121	79	1111001	y
26	1A	0011010	SUB	SUSTITUTE	58	3A	0111010	:	90	5A	1011010	Z	122	7A	1111010	z
27	1B	0011011	ESC	ESCAPE	59	3B	0111011	;	91	5B	1011011	[	123	7B	1111011	{
28	1C	0011100	FS	FILE SEPARATOR	60	3C	0111100	<	92	5C	1011100	\	124	7C	1111100	|
29	1D	0011101	GS	GROUP SEPARATOR	61	3D	0111101	=	93	5D	1011101	]	125	7D	1111101	}
30	1E	0011110	RS	RECORD SEPARATOR	62	3E	0111110	>	94	5E	1011110	^	126	7E	1111110	~
31	1F	0011111	US	UNIT SEPARATOR	63	3F	0111111	?	95	5F	1011111	_	127	7F	1111111	DEL

American Standard Code for Information Interchange (ASCII)

Decimal	Hex	Binary	Character		Decimal	Hex	Binary	Character	Decimal	Hex	Binary	Character	Decimal	Hex	Binary	Character
0	00	0000000	NULL	NULL CHARACTER	32	20	0100000	SPACE	64	40	1000000	@	96	60	1100000	`
1	01	0000001	SOH	START OF HEADING	33	21	0100001	!	65	41	1000001	A	97	61	1100001	a
2	02	0000010	STX	START OF TEXT	34	22	0100010	"	66	42	1000010	B	98	62	1100010	b
3	03	0000011	ETX	END OF TEXT	35	23	0100011	#	67	43	1000011	C	99	63	1100011	c
4	04	0000100	EOT	END OF TRANSMISSION	36	24	0100100	$	68	44	1000100	D	100	64	1100100	d
5	05	0000101	ENQ	ENQUIRY	37	25	0100101	%	69	45	1000101	E	101	65	1100101	e
6	06	0000110	ACK	ACKNOWLEDGE	38	26	0100110	&	70	46	1000110	F	102	66	1100110	f
7	07	0000111	BEL	BELL	39	27	0100111	'	71	47	1000111	G	103	67	1100111	g
8	08	0001000	BS	BACKSPACE	40	28	0101000	(	72	48	1001000	H	104	68	1101000	h
9	09	0001001	HT	HORIZONTAL TAB	41	29	0101001	)	73	49	1001001	I	105	69	1101001	i
10	0A	0001010	LF	LINE FEED	42	2A	0101010	*	74	4A	1001010	J	106	6A	1101010	j
11	0B	0001011	VT	VERTICAL TAB	43	2B	0101011	+	75	4B	1001011	K	107	6B	1101011	k
12	0C	0001100	FF	FORM FEED	44	2C	0101100	,	76	4C	1001100	L	108	6C	1101100	l
13	0D	0001101	CR	CARRIAGE RETURN	45	2D	0101101	-	77	4D	1001101	M	109	6D	1101101	m
14	0E	0001110	SO	SHIFT OUT	46	2E	0101110	.	78	4E	1001110	N	110	6E	1101110	n
15	0F	0001111	SI	SHIFT IN	47	2F	0101111	/	79	4F	1001111	O	111	6F	1101111	o
16	10	0010000	DLE	DATALINK ESCAPE	48	30	0110000	0	80	50	1010000	P	112	70	1110000	p
17	11	0010001	DC1	DEVICE CONTROL 1	49	31	0110001	1	81	51	1010001	Q	113	71	1110001	q
18	12	0010010	DC2	DEVICE CONTROL 2	50	32	0110010	2	82	52	1010010	R	114	72	1110010	r
19	13	0010011	DC3	DEVICE CONTROL 3	51	33	0110011	3	83	53	1010011	S	115	73	1110011	s
20	14	0010100	DC4	DEVICE CONTROL 4	52	34	0110100	4	84	54	1010100	T	116	74	1110100	t
21	15	0010101	NAK	NEGATIVE ACKNOWLEDGE	53	35	0110101	5	85	55	1010101	U	117	75	1110101	u
22	16	0010110	SYN	SYNCHRONOUS IDLE	54	36	0110110	6	86	56	1010110	V	118	76	1110110	v
23	17	0010111	ETB	END OF TRANS, BLOCK	55	37	0110111	7	87	57	1010111	W	119	77	1110111	w
24	18	0011000	CAN	CANCEL	56	38	0111000	8	88	58	1011000	X	120	78	1111000	x
25	19	0011001	EM	END OF MEDIUM	57	39	0111001	9	89	59	1011001	Y	121	79	1111001	y
26	1A	0011010	SUB	SUSTITUTE	58	3A	0111010	:	90	5A	1011010	Z	122	7A	1111010	z
27	1B	0011011	ESC	ESCAPE	59	3B	0111011	;	91	5B	1011011	[	123	7B	1111011	{
28	1C	0011100	FS	FILE SEPARATOR	60	3C	0111100	<	92	5C	1011100	\	124	7C	1111100	|
29	1D	0011101	GS	GROUP SEPARATOR	61	3D	0111101	=	93	5D	1011101	]	125	7D	1111101	}
30	1E	0011110	RS	RECORD SEPARATOR	62	3E	0111110	>	94	5E	1011110	^	126	7E	1111110	~
31	1F	0011111	US	UNIT SEPARATOR	63	3F	0111111	?	95	5F	1011111	_	127	7F	1111111	DEL

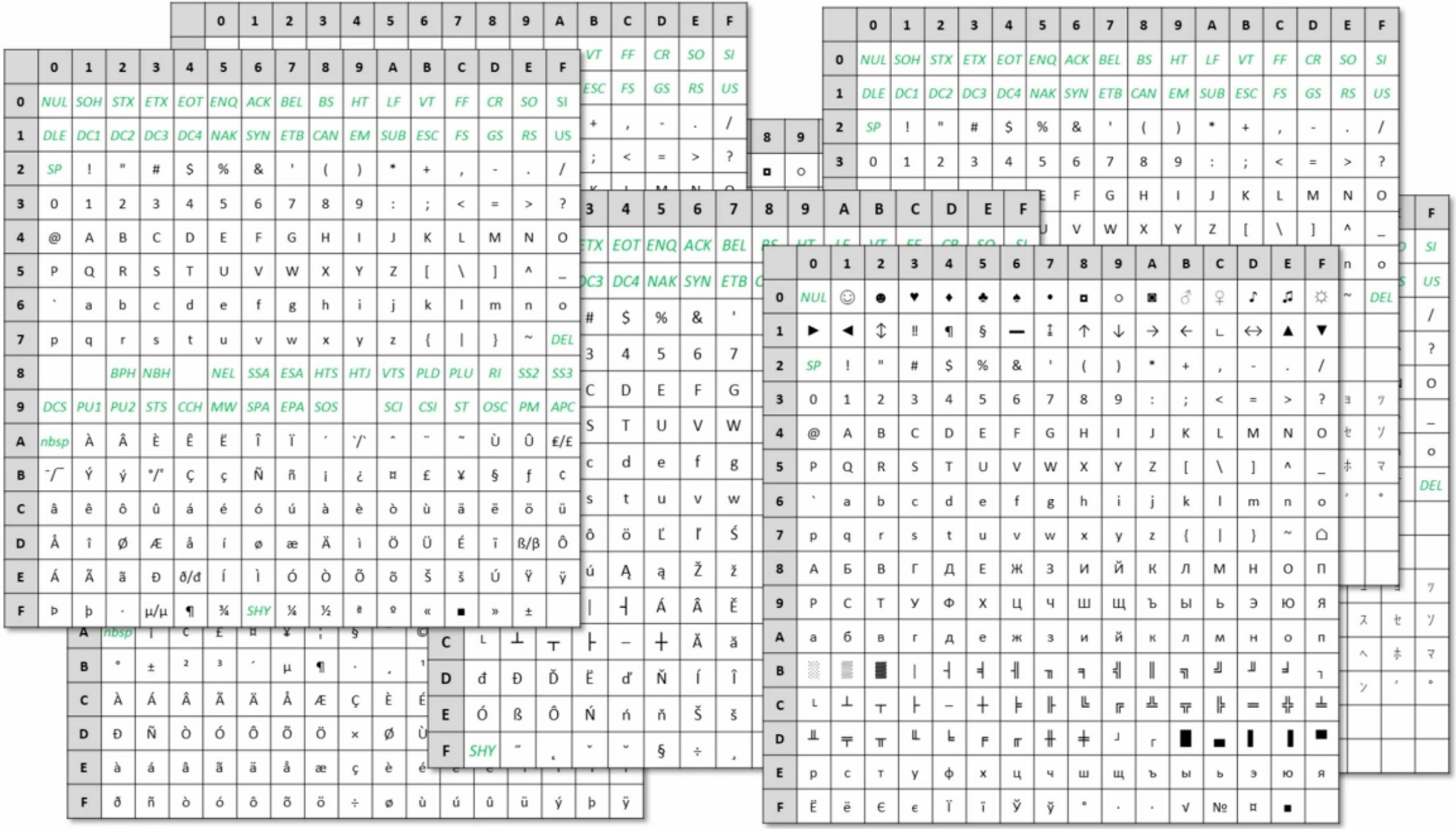
8	4	2	1
1	0	0	1

American Standard Code for Information Interchange (ASCII)

Decimal	Hex	Binary	Character		Decimal	Hex	Binary	Character	Decimal	Hex	Binary	Character	Decimal	Hex	Binary	Character
0	00	0000000	NULL	NULL CHARACTER	32	20	0100000	SPACE	64	40	1000000	@	96	60	1100000	`
1	01	0000001	SOH	START OF HEADING	33	21	0100001	!	65	41	1000001	A	97	61	1100001	a
2	02	0000010	STX	START OF TEXT	34	22	0100010	"	66	42	1000010	B	98	62	1100010	b
3	03	0000011	ETX	END OF TEXT	35	23	0100011	#	67	43	1000011	C	99	63	1100011	c
4	04	0000100	EOT	END OF TRANSMISSION	36	24	0100100	$	68	44	1000100	D	100	64	1100100	d
5	05	0000101	ENQ	ENQUIRY	37	25	0100101	%	69	45	1000101	E	101	65	1100101	e
6	06	0000110	ACK	ACKNOWLEDGE	38	26	0100110	&	70	46	1000110	F	102	66	1100110	f
7	07	0000111	BEL	BELL	39	27	0100111	'	71	47	1000111	G	103	67	1100111	g
8	08	0001000	BS	BACKSPACE	40	28	0101000	(	72	48	1001000	H	104	68	1101000	h
9	09	0001001	HT	HORIZONTAL TAB	41	29	0101001	)	73	49	1001001	I	105	69	1101001	i
10	0A	0001010	LF	LINE FEED	42	2A	0101010	*	74	4A	1001010	J	106	6A	1101010	j
11	0B	0001011	VT	VERTICAL TAB	43	2B	0101011	+	75	4B	1001011	K	107	6B	1101011	k
12	0C	0001100	FF	FORM FEED	44	2C	0101100	,	76	4C	1001100	L	108	6C	1101100	l
13	0D	0001101	CR	CARRIAGE RETURN	45	2D	0101101	-	77	4D	1001101	M	109	6D	1101101	m
14	0E	0001110	SO	SHIFT OUT	46	2E	0101110	.	78	4E	1001110	N	110	6E	1101110	n
15	0F	0001111	SI	SHIFT IN	47	2F	0101111	/	79	4F	1001111	O	111	6F	1101111	o
16	10	0010000	DLE	DATALINK ESCAPE	48	30	0110000	0	80	50	1010000	P	112	70	1110000	p
17	11	0010001	DC1	DEVICE CONTROL 1	49	31	0110001	1	81	51	1010001	Q	113	71	1110001	q
18	12	0010010	DC2	DEVICE CONTROL 2	50	32	0110010	2	82	52	1010010	R	114	72	1110010	r
19	13	0010011	DC3	DEVICE CONTROL 3	51	33	0110011	3	83	53	1010011	S	115	73	1110011	s
20	14	0010100	DC4	DEVICE CONTROL 4	52	34	0110100	4	84	54	1010100	T	116	74	1110100	t
21	15	0010101	NAK	NEGATIVE ACKNOWLEDGE	53	35	0110101	5	85	55	1010101	U	117	75	1110101	u
22	16	0010110	SYN	SYNCHRONOUS IDLE	54	36	0110110	6	86	56	1010110	V	118	76	1110110	v
23	17	0010111	ETB	END OF TRANS, BLOCK	55	37	0110111	7	87	57	1010111	W	119	77	1110111	w
24	18	0011000	CAN	CANCEL	56	38	0111000	8	88	58	1011000	X	120	78	1111000	x
25	19	0011001	EM	END OF MEDIUM	57	39	0111001	9	89	59	1011001	Y	121	79	1111001	y
26	1A	0011010	SUB	SUSTITUTE	58	3A	0111010	:	90	5A	1011010	Z	122	7A	1111010	z
27	1B	0011011	ESC	ESCAPE	59	3B	0111011	;	91	5B	1011011	[	123	7B	1111011	{
28	1C	0011100	FS	FILE SEPARATOR	60	3C	0111100	<	92	5C	1011100	\	124	7C	1111100	|
29	1D	0011101	GS	GROUP SEPARATOR	61	3D	0111101	=	93	5D	1011101	]	125	7D	1111101	}
30	1E	0011110	RS	RECORD SEPARATOR	62	3E	0111110	>	94	5E	1011110	^	126	7E	1111110	~
31	1F	0011111	US	UNIT SEPARATOR	63	3F	0111111	?	95	5F	1011111	_	127	7F	1111111	DEL

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	*NUL*	*SOH*	*STX*	*ETX*	*EOT*	*ENQ*	*ACK*	*BEL*	*BS*	*TAB*	*LF*	*VT*	*FF*	*CR*	*SO*	*SI*
1	*DLE*	*DC1*	*DC2*	*DC3*	*DC4*	*NAK*	*SYN*	*ETB*	*CAN*	*EM*	*SUB*	*ESC*	*FS*	*GS*	*RS*	*US*
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	'	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	*DEL*

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	☺	☻	♥	♦	♣	♠	•	◘	○	◙	♂	♀	♪	♫	☼
1	►	◄	↕	‼	¶	§	▬	↨	↑	↓	→	←	∟	↔	▲	▼
2	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	⌂
8	Ç	ü	é	â	ä	à	å	ç	ê	ë	è	ï	î	ì	Ä	Å
9	É	æ	Æ	ô	ö	ò	û	ù	ÿ	Ö	Ü	¢	£	¥	₧	ƒ
A	á	í	ó	ú	ñ	Ñ	ª	º	¿	⌐	¬	½	¼	¡	«	»
B	░	▒	▓	|	┤	╡	╢	╖	╕	╣	‖	╗	╝	╜	╛	┐
C	└	┴	┬	├	─	┼	╞	╟	╚	╔	╩	╦	╠	═	╬	╧
D	╨	╤	╥	╙	╘	╒	╓	╫	╪	┘	┌	▉	▄	▌	▐	▀
E	α	ß	Γ	π	Σ	σ	µ	τ	Φ	Θ	Ω	δ	∞	φ	ε	∩
F	≡	±	≥	≤	⌠	⌡	÷	≈	°	∙	·	√	ⁿ	²	■	nbsp



After www – unicode standard was published in 1991.

The design goals of Unicode

- A unique code point for every possible character

$2^{16} = 65536$

The design goals of Unicode

- A unique code point for every possible character
- Backwards compatibility with ASCII
- Space efficient

00000000 00000000 00000000 01000001

The design goals of Unicode

- A unique code point for every possible character
- Backwards compatibility with ASCII
- Space efficient
- Allow for efficient data transmission

00000000 00000000 00000000 01000001

Unicode Transformation Format (UTF-8)

A

0	1	0	0	0	0	0	1

U+0041

Control Bit

Ω

			1024	512	256	128	64
1	**1**	**0**	0	1	1	1	0

		32	16	8	4	2	1
1	**0**	1	0	1	0	0	1

U+03A9

Unicode Transformation Format (UTF-8)

A

0	1	0	0	0	0	0	1

U+0041

Ω

			1024	512	256	128	64
1	1	0	0	1	1	1	0

		32	16	8	4	2	1
1	0	1	0	1	0	0	1

U+03A9

♫

				32768	16384	8192	4096
1	1	1	0	0	0	1	0

		2048	1024	512	256	128	64
1	0	0	1	1	0	0	1

		32	16	8	4	2	1
1	0	1	0	1	0	1	1

U+266B

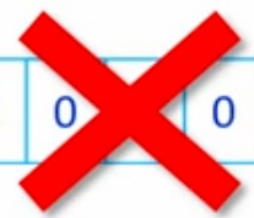
For Chinese, Koren + other symbols

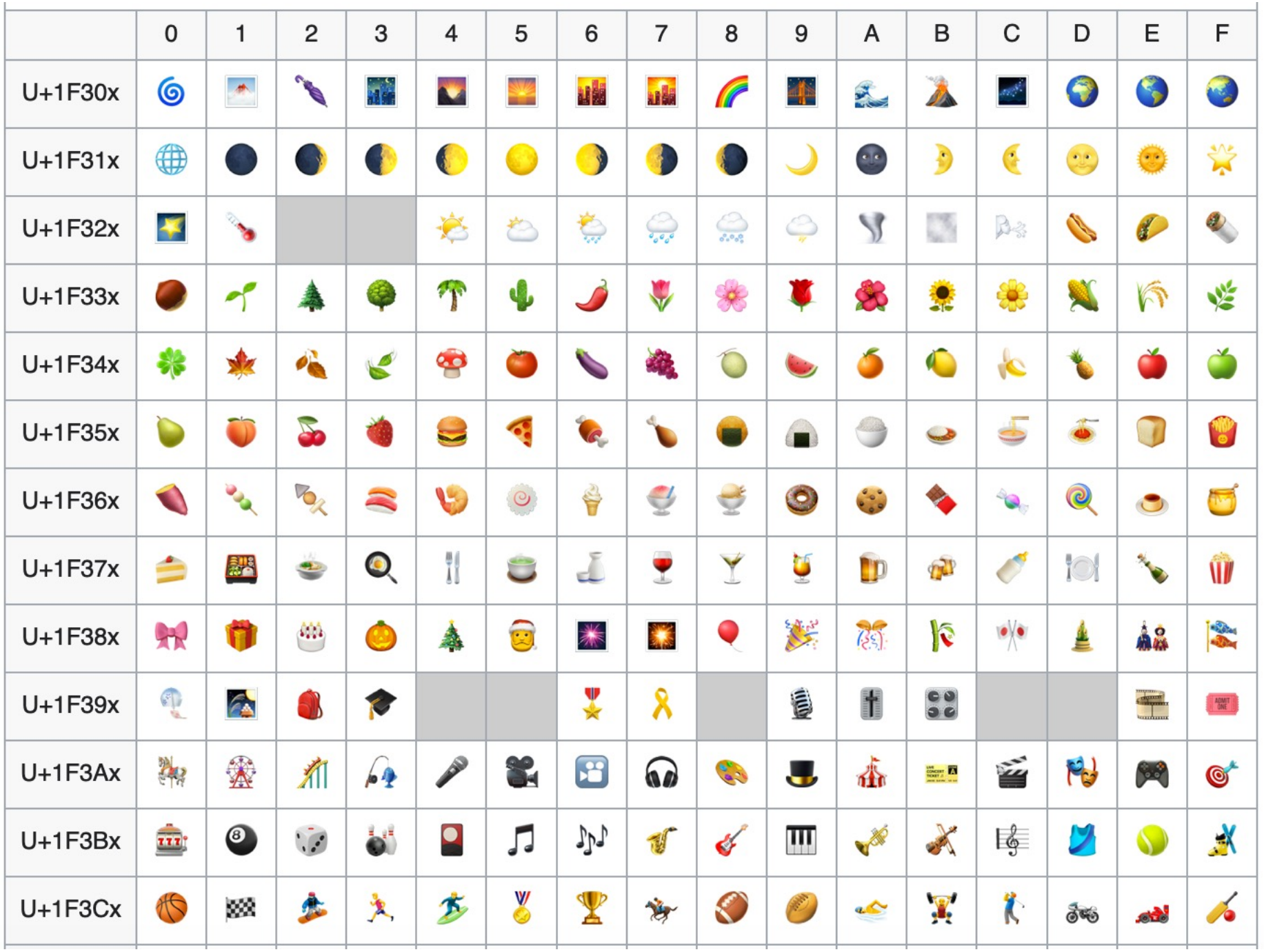
Byte 1	Byte 2	Byte 3	Byte 4	Bits available
0XXXXXXX				7
110XXXXX	10XXXXXX			11
1110XXXX	10XXXXXX	10XXXXXX		16
11110XXX	10XXXXXX	10XXXXXX	10XXXXXX	21

Emojis

A | 0 1 0 0 0 0 0 1 | U+0041

A | 1 1 1 0 0 0 0 0 | 1 0 0 0 0 0 0 | 0 1 0 0 0 0 0 1 | U+0041




0 1 2 3 4 5 6 7 8 9 A B C D E F
U+1F30x
U+1F31x
U+1F32x
U+1F33x
U+1F34x
U+1F35x
U+1F36x
U+1F37x
U+1F38x
U+1F39x
U+1F3Ax
U+1F3Bx
U+1F3Cx

Byte 1	Byte 2	Byte 3	Byte 4	Bits available
0XXXXXXX				7
110XXXXX	10XXXXXX			11
1110XXXX	10XXXXXX	10XXXXXX		16
11110XXX	10XXXXXX	10XXXXXX	10XXXXXX	21

More than 1 million possible characters

Summary

- ASCII is a 7 bit encoding system for a limited number of characters
- Extended ASCII resulted in lots of incompatible code pages
- Unicode allows every character in every written language to be encoded
- Unicode is backwards compatible with ASCII
- Unicode is space efficient
- Unicode Transformation Format (UTF-8) uses 1, 2, 3 or 4 bytes
- Unicode is universally supported

Representing Negative Integers in Binary

Two's Complement

Convert 01011010 into denary

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

64 + 16 + 8 + 2 = 90

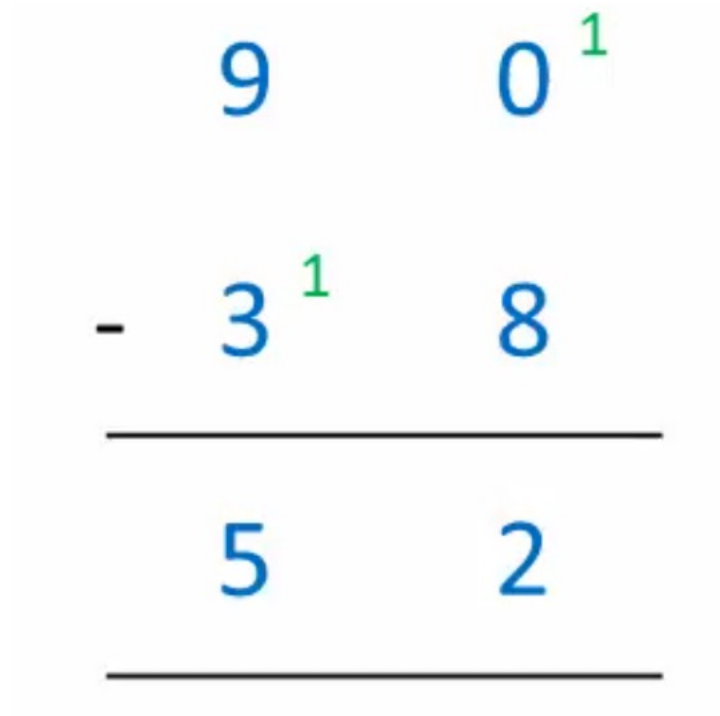
$01011010_2 = 90_{10}$

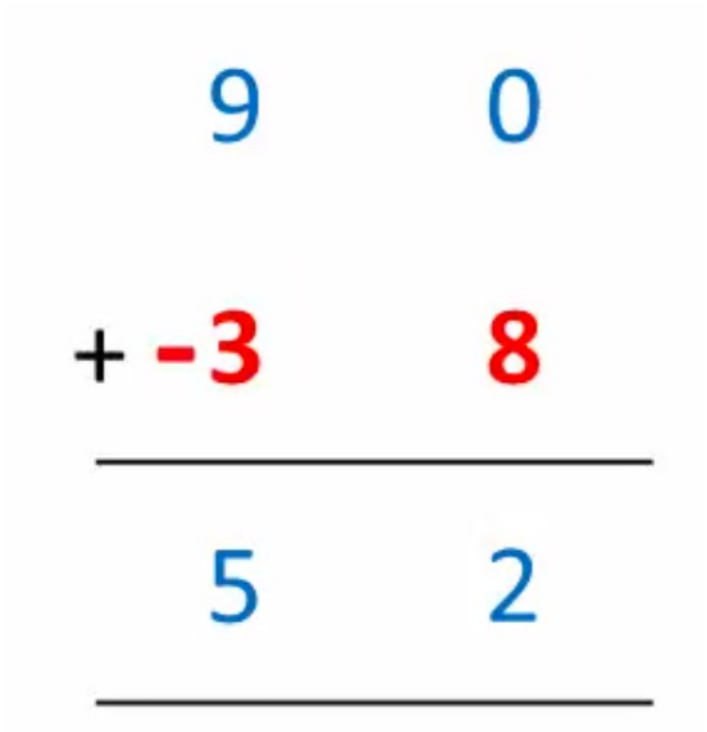
Convert 11011010 into denary

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

$$-128 + 64 + 16 + 8 + 2 = -38$$

$$11011010_2 = -38_{10}$$


9 0 1
- 3 1 8
5 2


9 0
+ -3 8
5 2

	0	1	0	1	1	0	1	0	90
^{1} +	1 $_{1}$	1	0 $_{1}$	1 $_{1}$	1	0 $_{1}$	1	0	-38
	0	0	1	1	0	1	0	0	

Convert 00110100 into denary

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

32 + 16 + 4 = 52

$00110100_2 = 52_{10}$

Convert -6 into 8 bit binary

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

$$-128 + 64 + 32 + 16 + 8 + 2 = -6$$

$$-6_{10} = 11111010_{2}$$

Convert -49 into 8 bit binary

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

$-128 + 64 + 8 + 4 + 2 + 1 = -49$

$-49_{10} = 11001111_{2}$

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

$$-128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = -1$$

$$11111111_{2} = -1_{10}$$

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

$$64 + 32 + 16 + 8 + 4 + 2 + 1 = 127$$

$$01111111_{2} = 127_{10}$$

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

-128

$10000000_{2} = -128_{10}$

0	00000000	32	00100000	64	01000000	96	01100000	128	10000000	160	10100000	192	11000000	224	11100000
1	00000001	33	00100001	65	01000001	97	01100001	129	10000001	161	10100001	193	11000001	225	11100001
2	00000010	34	00100010	66	01000010	98	01100010	130	10000010	162	10100010	194	11000010	226	11100010
3	00000011	35	00100011	67	01000011	99	01100011	131	10000011	163	10100011	195	11000011	227	11100011
4	00000100	36	00100100	68	01000100	100	01100100	132	10000100	164	10100100	196	11000100	228	11100100
5	00000101	37	00100101	69	01000101	101	01100101	133	10000101	165	10100101	197	11000101	229	11100101
6	00000110	38	00100110	70	01000110	102	01100110	134	10000110	166	10100110	198	11000110	230	11100110
7	00000111	39	00100111	71	01000111	103	01100111	135	10000111	167	10100111	199	11000111	231	11100111
8	00001000	40	00101000	72	01001000	104	01101000	136	10001000	168	10101000	200	11001000	232	11101000
9	00001001	41	00101001	73	01001001	105	01101001	137	10001001	169	10101001	201	11001001	233	11101001
10	00001010	42	00101010	74	01001010	106	01101010	138	10001010	170	10101010	202	11001010	234	11101010
11	00001011	43	00101011	75	01001011	107	01101011	139	10001011	171	10101011	203	11001011	235	11101011
12	00001100	44	00101100	76	01001100	108	01101100	140	10001100	172	10101100	204	11001100	236	11101100
13	00001101	45	00101101	77	01001101	109	01101101	141	10001101	173	10101101	205	11001101	237	11101101
14	00001110	46	00101110	78	01001110	110	01101110	142	10001110	174	10101110	206	11001110	238	11101110
15	00001111	47	00101111	79	01001111	111	01101111	143	10001111	175	10101111	207	11001111	239	11101111
16	00010000	48	00110000	80	01010000	112	01110000	144	10010000	176	10110000	208	11010000	240	11110000
17	00010001	49	00110001	81	01010001	113	01110001	145	10010001	177	10110001	209	11010001	241	11110001
18	00010010	50	00110010	82	01010010	114	01110010	146	10010010	178	10110010	210	11010010	242	11110010
19	00010011	51	00110011	83	01010011	115	01110011	147	10010011	179	10110011	211	11010011	243	11110011
20	00010100	52	00110100	84	01010100	116	01110100	148	10010100	180	10110100	212	11010100	244	11110100
21	00010101	53	00110101	85	01010101	117	01110101	149	10010101	181	10110101	213	11010101	245	11110101
22	00010110	54	00110110	86	01010110	118	01110110	150	10010110	182	10110110	214	11010110	246	11110110
23	00010111	55	00110111	87	01010111	119	01110111	151	10010111	183	10110111	215	11010111	247	11110111
24	00011000	56	00111000	88	01011000	120	01111000	152	10011000	184	10111000	216	11011000	248	11111000
25	00011001	57	00111001	89	01011001	121	01111001	153	10011001	185	10111001	217	11011001	249	11111001
26	00011010	58	00111010	90	01011010	122	01111010	154	10011010	186	10111010	218	11011010	250	11111010
27	00011011	59	00111011	91	01011011	123	01111011	155	10011011	187	10111011	219	11011011	251	11111011
28	00011100	60	00111100	92	01011100	124	01111100	156	10011100	188	10111100	220	11011100	252	11111100
29	00011101	61	00111101	93	01011101	125	01111101	157	10011101	189	10111101	221	11011101	253	11111101
30	00011110	62	00111110	94	01011110	126	01111110	158	10011110	190	10111110	222	11011110	254	11111110
31	00011111	63	00111111	95	01011111	127	01111111	159	10011111	191	10111111	223	11011111	255	11111111

0	00000000	32	00100000	64	01000000	96	01100000	-128	10000000	-96	10100000	-64	11000000	-32	11100000
1	00000001	33	00100001	65	01000001	97	01100001	-127	10000001	-95	10100001	-63	11000001	-31	11100001
2	00000010	34	00100010	66	01000010	98	01100010	-126	10000010	-94	10100010	-62	11000010	-30	11100010
3	00000011	35	00100011	67	01000011	99	01100011	-125	10000011	-93	10100011	-61	11000011	-29	11100011
4	00000100	36	00100100	68	01000100	100	01100100	-124	10000100	-92	10100100	-60	11000100	-28	11100100
5	00000101	37	00100101	69	01000101	101	01100101	-123	10000101	-91	10100101	-59	11000101	-27	11100101
6	00000110	38	00100110	70	01000110	102	01100110	-122	10000110	-90	10100110	-58	11000110	-26	11100110
7	00000111	39	00100111	71	01000111	103	01100111	-121	10000111	-89	10100111	-57	11000111	-25	11100111
8	00001000	40	00101000	72	01001000	104	01101000	-120	10001000	-88	10101000	-56	11001000	-24	11101000
9	00001001	41	00101001	73	01001001	105	01101001	-119	10001001	-87	10101001	-55	11001001	-23	11101001
10	00001010	42	00101010	74	01001010	106	01101010	-118	10001010	-86	10101010	-54	11001010	-22	11101010
11	00001011	43	00101011	75	01001011	107	01101011	-117	10001011	-85	10101011	-53	11001011	-21	11101011
12	00001100	44	00101100	76	01001100	108	01101100	-116	10001100	-84	10101100	-52	11001100	-20	11101100
13	00001101	45	00101101	77	01001101	109	01101101	-115	10001101	-83	10101101	-51	11001101	-19	11101101
14	00001110	46	00101110	78	01001110	110	01101110	-114	10001110	-82	10101110	-50	11001110	-18	11101110
15	00001111	47	00101111	79	01001111	111	01101111	-113	10001111	-81	10101111	-49	11001111	-17	11101111
16	00010000	48	00110000	80	01010000	112	01110000	-112	10010000	-80	10110000	-48	11010000	-16	11110000
17	00010001	49	00110001	81	01010001	113	01110001	-111	10010001	-79	10110001	-47	11010001	-15	11110001
18	00010010	50	00110010	82	01010010	114	01110010	-110	10010010	-78	10110010	-46	11010010	-14	11110010
19	00010011	51	00110011	83	01010011	115	01110011	-109	10010011	-77	10110011	-45	11010011	-13	11110011
20	00010100	52	00110100	84	01010100	116	01110100	-108	10010100	-76	10110100	-44	11010100	-12	11110100
21	00010101	53	00110101	85	01010101	117	01110101	-107	10010101	-75	10110101	-43	11010101	-11	11110101
22	00010110	54	00110110	86	01010110	118	01110110	-106	10010110	-74	10110110	-42	11010110	-10	11110110
23	00010111	55	00110111	87	01010111	119	01110111	-105	10010111	-73	10110111	-41	11010111	-9	11110111
24	00011000	56	00111000	88	01011000	120	01111000	-104	10011000	-72	10111000	-40	11011000	-8	11111000
25	00011001	57	00111001	89	01011001	121	01111001	-103	10011001	-71	10111001	-39	11011001	-7	11111001
26	00011010	58	00111010	90	01011010	122	01111010	-102	10011010	-70	10111010	-38	11011010	-6	11111010
27	00011011	59	00111011	91	01011011	123	01111011	-101	10011011	-69	10111011	-37	11011011	-5	11111011
28	00011100	60	00111100	92	01011100	124	01111100	-100	10011100	-68	10111100	-36	11011100	-4	11111100
29	00011101	61	00111101	93	01011101	125	01111101	-99	10011101	-67	10111101	-35	11011101	-3	11111101
30	00011110	62	00111110	94	01011110	126	01111110	-98	10011110	-66	10111110	-34	11011110	-2	11111110
31	00011111	63	00111111	95	01011111	127	01111111	-97	10011111	-65	10111111	-33	11011111	-1	11111111

Excersise -4

10000101	127
11111001	-56
01010110	-86
11101101	-61

Two’s Complement

Convert -6 into 8 bit binary

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

One’s Compliment

Convert -6 into 8 bit binary

Add 1 to One’s Compliment to get its Two’s Compliment

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

Convert -6 into 8 bit binary

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

$$-128 + 64 + 32 + 16 + 8 + 2 = -6$$

$$11111010_2 = -6_{10}$$

Convert -109 into 8 bit binary

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

Convert -109 into 8 bit binary

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

Convert -109 into 8 bit binary

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

Convert -109 into 8 bit binary

1 0 0 1 0 0 1 1

Convert -77 into 8 bit binary

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

Convert -77 into 8 bit binary

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

Convert -77 into 8 bit binary

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

Convert -77 into 8 bit binary

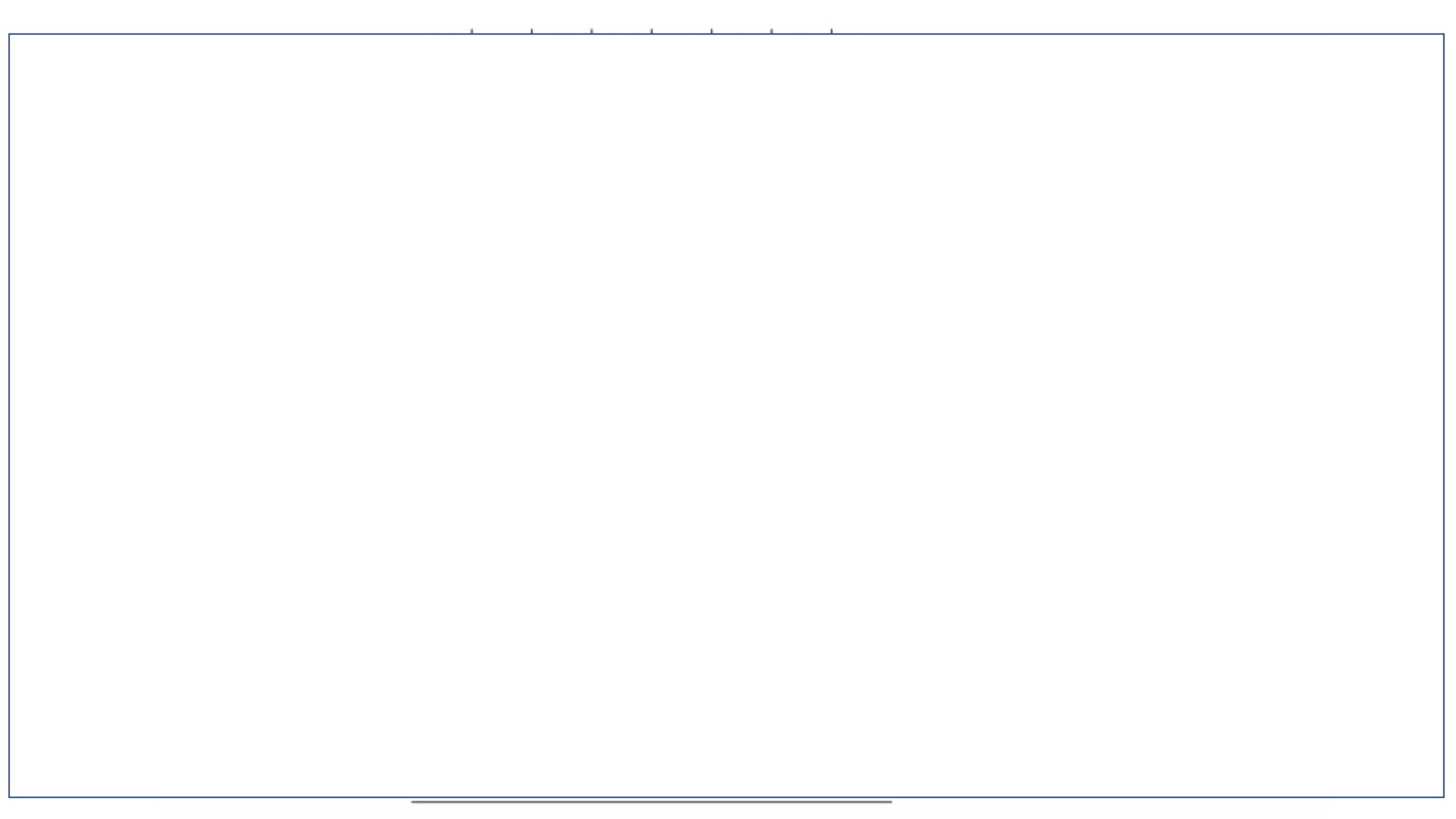
1 0 1 1 0 0 1 1

Excersise 5: Convert them using Two’s Compliment.

-17

-44

-122



Convert -44 into 8 bit binary

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

$-44_{10} = 11010100_{2}$

Summary

- Computers use two's complement to represent negative numbers in binary
- Half of the available combinations of bits are used to represent negative numbers
- Three methods to convert negative denary numbers into binary

Decimal to Binary

Conversion steps:

1. Divide the number by 2.
2. Get the integer quotient for the next iteration.
3. Get the remainder for the binary digit.
4. Repeat the steps until the quotient is equal to 0.

Example #1

Convert 13_{10} to binary:

Division by 2	Quotient	Remainder	Bit #
13/2	6	1	0
6/2	3	0	1
3/2	1	1	2
1/2	0	1	3

So $13_{10} = 1101_2$

Binary to Decimal

For binary number with n digits:

$$d_{n-1} \ldots d_3\, d_2\, d_1\, d_0$$

The decimal number is equal to the sum of binary digits (d_n) times their power of 2 (2^n):

$$\text{decimal} = d_0 \times 2^0 + d_1 \times 2^1 + d_2 \times 2^2 + \ldots$$

Example

Find the decimal value of 111001_2:

binary number:	1	1	1	0	0	1
power of 2:	2^5	2^4	2^3	2^2	2^1	2^0

$$111001_2 = 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 57_{10}$$

Decimal to Hexadecmial

Conversion steps:

1. Divide the number by 16.
2. Get the integer quotient for the next iteration.
3. Get the remainder for the hex digit.
4. Repeat the steps until the quotient is equal to 0.

Example #1

Convert 7562_{10} to hex:

Division by 16	Quotient (integer)	Remainder (decimal)	Remainder (hex)	Digit #
7562/16	472	10	A	0
472/16	29	8	8	1
29/16	1	13	D	2
1/16	0	1	1	3

So $7562_{10} = 1D8A_{16}$

Representing Real Numbers in Binary

Fixed Point Binary Fractions

Decimal Point in Denary
Binary Point in Binary

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

$$64 + 8 + 1/2 + 1/4 = 72\ 3/4$$

$$01001000110{0}_{2} = 72.75_{10}$$

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

-128	64	32	16	8	4	2	1	0.5	0.25	0.125	0.0625
0	1	1	0	1	0	1	1	1	0	1	0

-128	64	32	16	8	4	2	1	0.5	0.25	0.125	0.0625
0	1	1	0	1	0	1	1	1	0	1	0

$$64 + 32 + 8 + 2 + 1 + 0.5 + 0.125 = 107.625$$

$$011010111010_{2} = 107.625_{10}$$

-128	64	32	16	8	4	2	1	0.5	0.25	0.125	0.0625
1	0	1	0	0	1	1	0	1	1	0	0

$$-128 + 32 + 4 + 2 + 0.5 + 0.25 = -89.25$$

$$101001101100_{2} = -89.25_{10}$$

Excersise 6

The following binary numbers are stored using two's complement in a 12 bit register with 4 bits after the binary point. Convert them into decimal fractions.

011111111111

111111111111

000001110010

Excersise 7

Using two's complement, convert the following denary numbers into fixed point binary to be stored in a 12 bit register with 4 bits after the binary point.

27.5

-55.75

-1.75

Given a 4 bit register, with 1 bit before and 3 bits after the binary point, using two's complement, calculate:

- The largest positive number that can be represented

-1	0.5	0.25	0.125
0	1	1	1

0.5 + 0.25 + 0.125 = 0.875

- The smallest positive number that can be represented (not including 0)

-1	0.5	0.25	0.125
0	0	0	1

0.125

- The smallest magnitude negative number that can be represented (closest to 0)

-1	0.5	0.25	0.125
1	1	1	1

-1 + 0.5 + 0.25 + 0.125 = -0.125

- The largest magnitude negative number that can be represented

-1	0.5	0.25	0.125
1	0	0	0

-1

Summary

- Fixed point binary is used in Digital Signal Processing
- Simpler and therefore cheaper processor hardware
- Greatly simplified arithmetic means much faster processing
- Trade off between range and precision
- Some numbers can never be represented accurately

Floating Point Binary

To represent very large values
To represent very small values
To represent values with great accuracy

Standard Scientific Notation

2.99×10^{8}	m/s	Speed of light
6.02×10^{23}	/mol	Avogadro's number
1.60×10^{-19}	C	Charge of electron
4.35×10^{17}	s	Age of the universe

mantissa exponent

$$6.02 \times 10^{23}$$

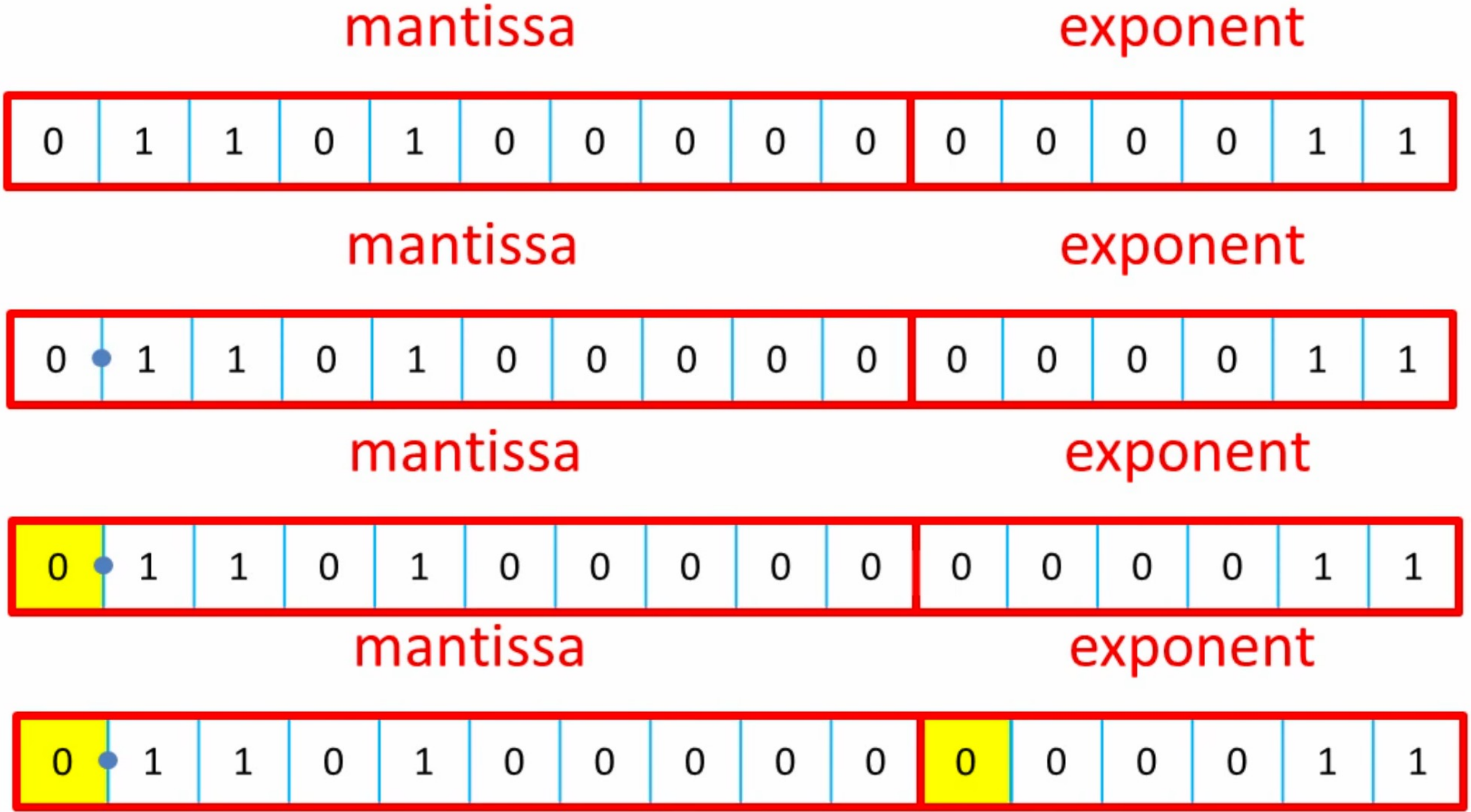
Governs the precisiion

Governs the range

$$6.022140857 \times 10^{23}$$

$$1.60 \times 10^{-19}$$

0.0000000000000000001160


mantissa
exponent
0 1 1 0 1 0 0 0 0 0 0 0 0 0 1 1
mantissa
exponent
0 1 1 0 1 0 0 0 0 0 0 0 0 0 1 1
mantissa
exponent
0 1 1 0 1 0 0 0 0 0 0 0 0 0 1 1
mantissa
exponent
0 1 1 0 1 0 0 0 0 0 0 0 0 0 1 1

Floating-Point Binary

IEEE Short Real: 32 bits	1 bit for the sign, 8 bits for the exponent, and 23 bits for the mantissa. Also called *single precision*.
IEEE Long Real: 64 bits	1 bit for the sign, 11 bits for the exponent, and 52 bits for the mantissa. Also called *double precision*.

Both formats use essentially the same method for storing floating-point binary numbers, so we will use the Short Real as an example in this tutorial.
The bits in an IEEE Short Real are arranged as follows, with the most significant bit (MSB) on the left:

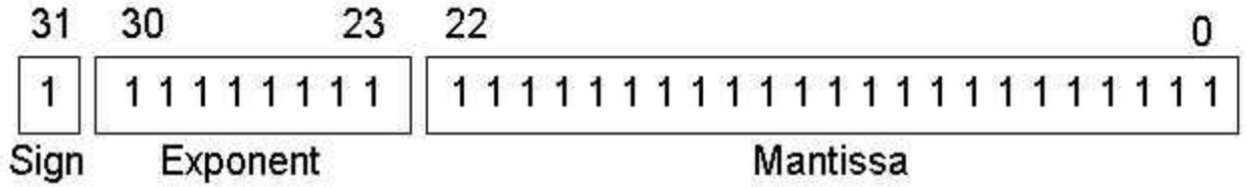


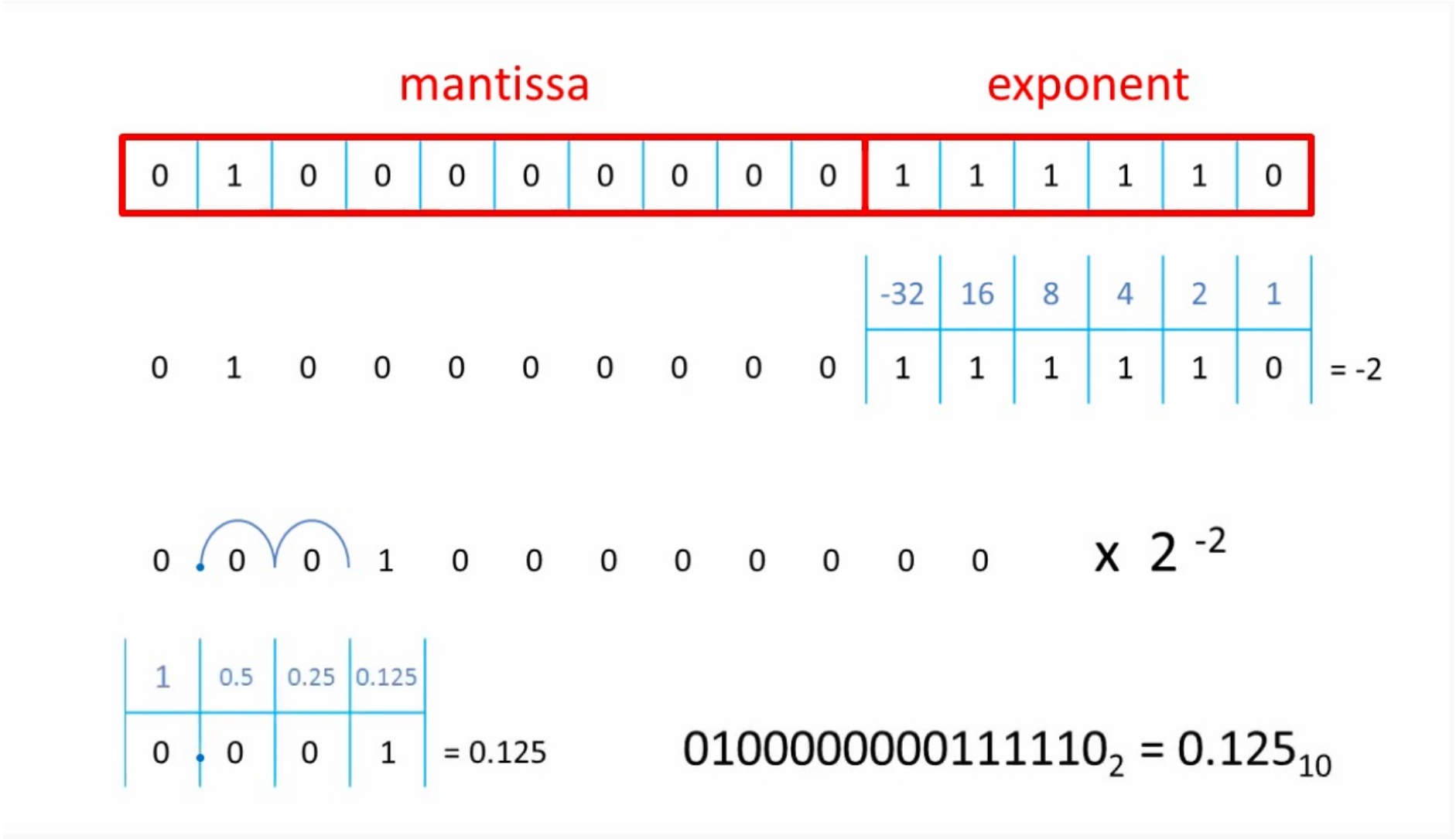
Fig. 1

http://cstl-csm.semo.edu/xzhang/Class%20Folder/CS280/Workbook_HTML/FLOATING_tut.htm

Floating-Point Binary

Floating-Point Binary

Floating-Point Binary



Exercise 8

Convert the following floating point binary numbers into denary. Assume 10 bits for the mantissa and 6 bits for the exponent, both in two's complement

0110110000000100

0101000000111111

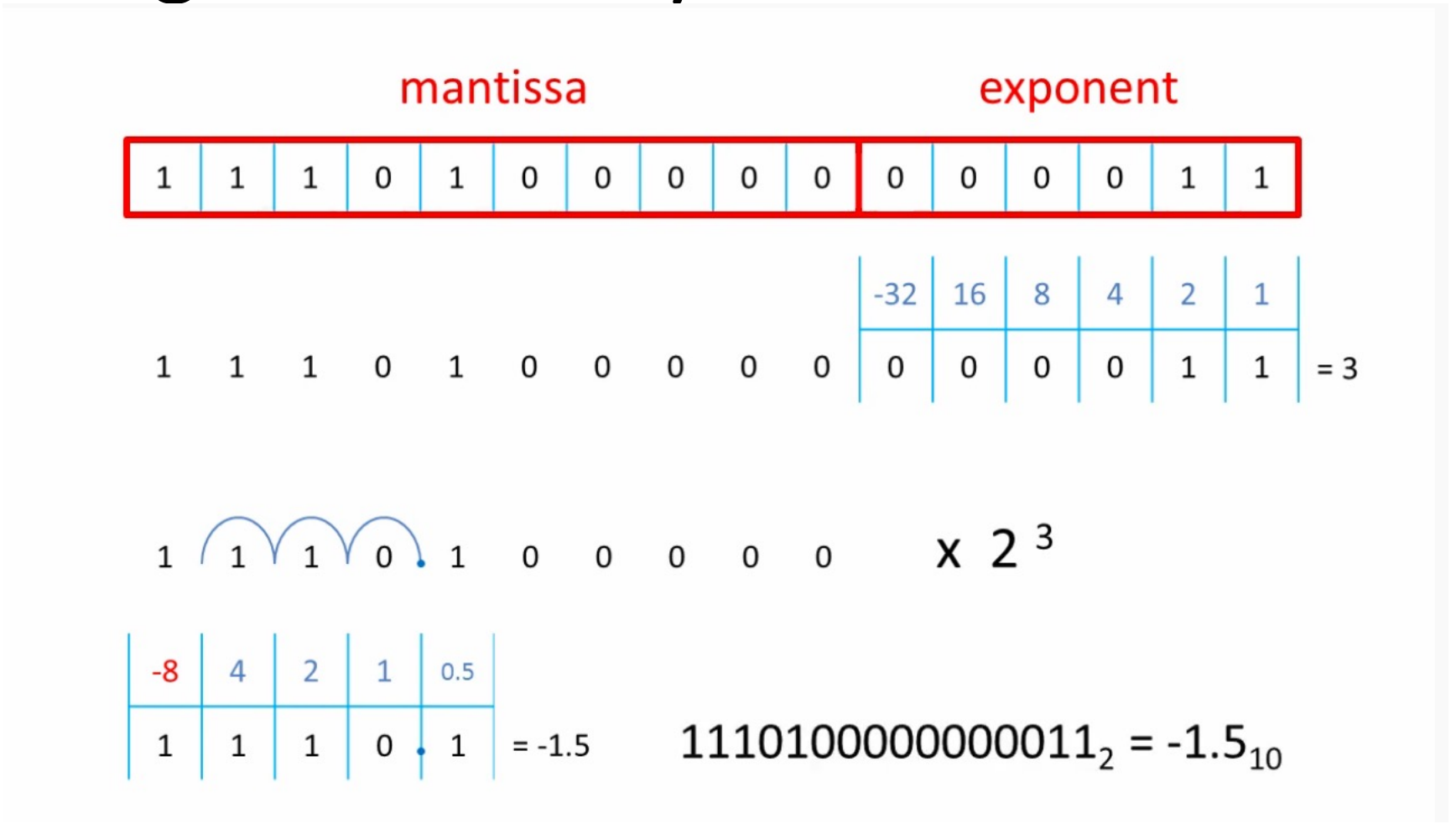
Exercise 9

Convert the following floating point binary numbers into denary. Assume 4 bits for the mantissa and 4 bits for the exponent, both in two's complement

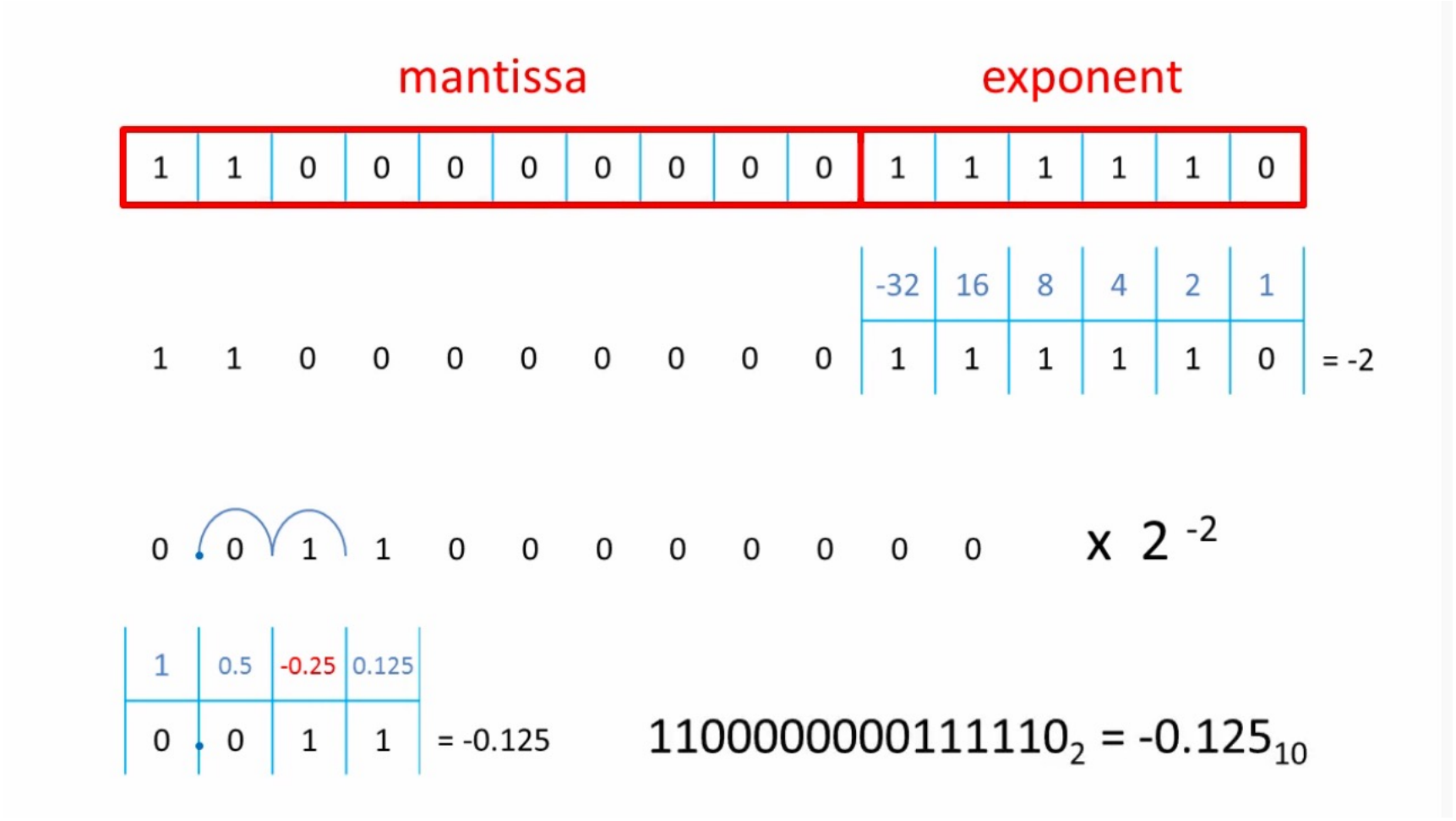
01110011

01111110

Floating-Point Binary



Floating-Point Binary



Exercise 10

Convert the following floating point binary numbers into denary. Assume 10 bits for the mantissa and 6 bits for the exponent, both in two's complement

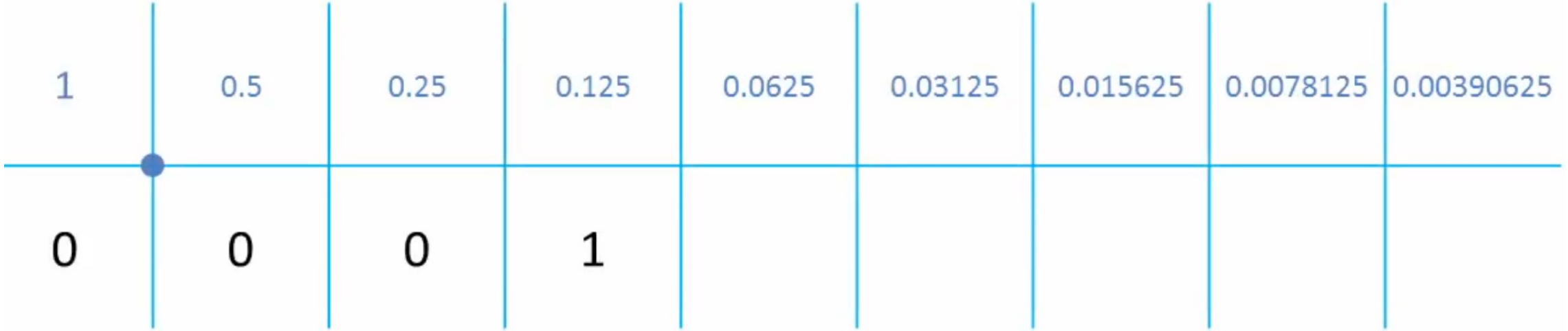
1101000000 111111

1001101000000110

Floating Point Binary

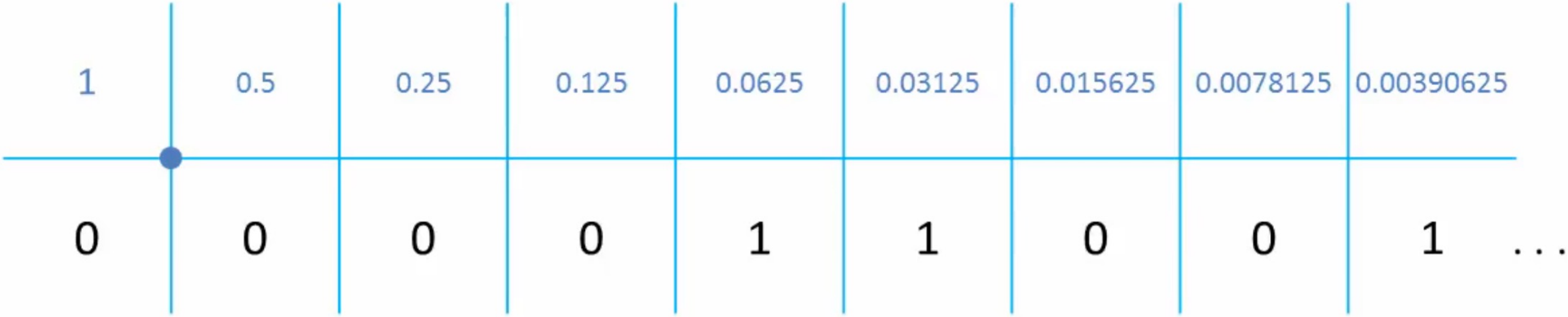
Range versus Precision

$$0.125_{10} = 0.001_{2}$$



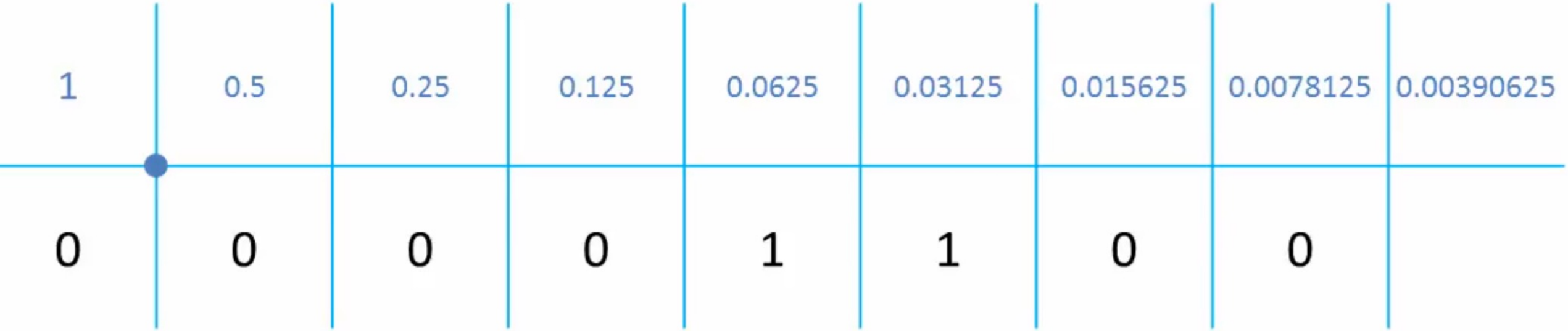
Precision relates to numbers of bits used to represent a binary.
These two values are exactly the same.
With the precision of 4 bits you can represent accuratley.

$$0.1_{10} = 0.0001100110011001100110011\ldots_2$$



1/10 in Decimal cannot be accuratley represented in Binary as
it is recurring sequance of bits.
Cannot be respenested accurately in binary with any number of bits.

$$0.1_{10} = 0.00011_2$$



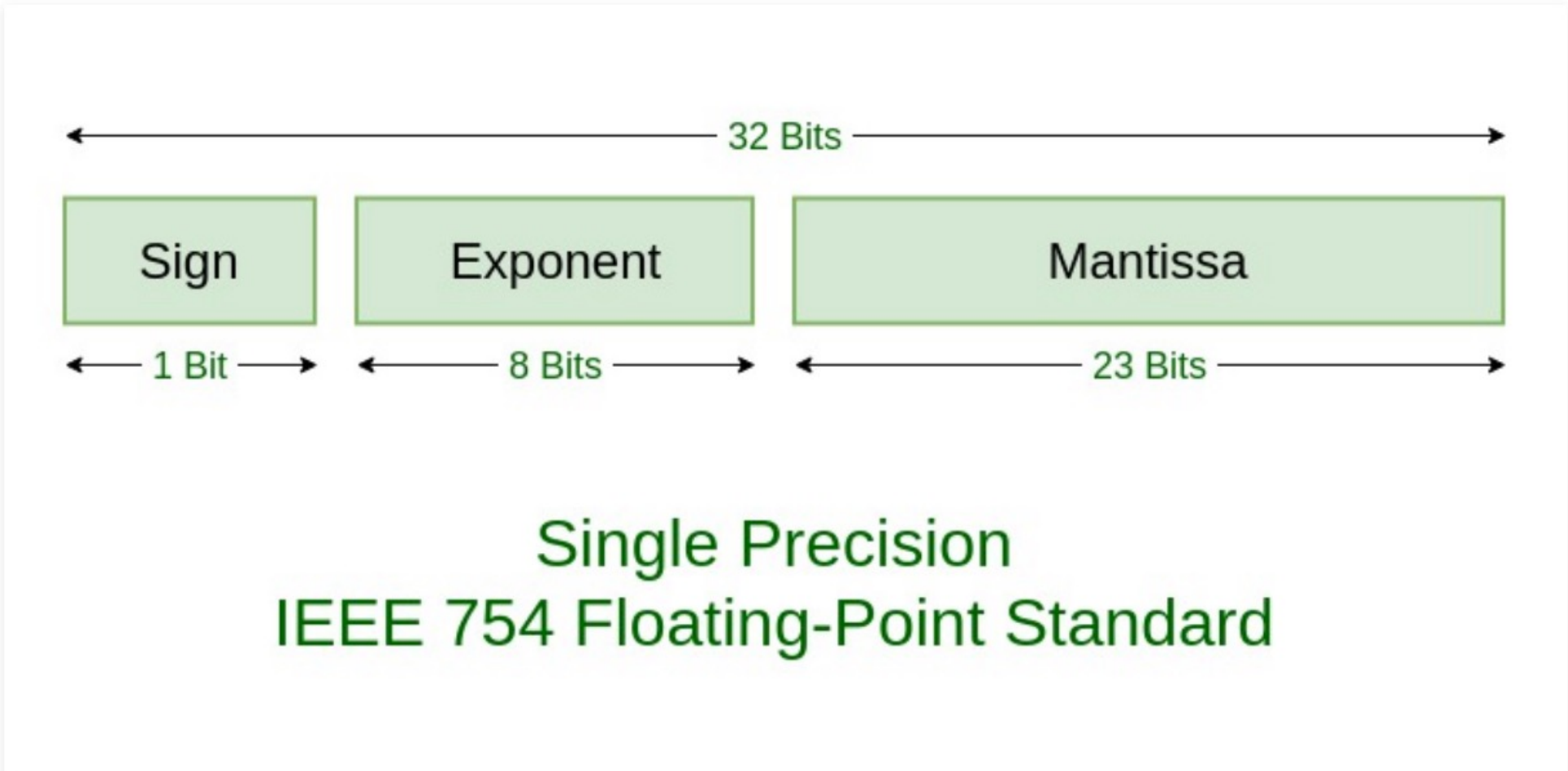
$$0.00011_2 = 0.0625 + 0.03125 = 0.09375_{10}$$

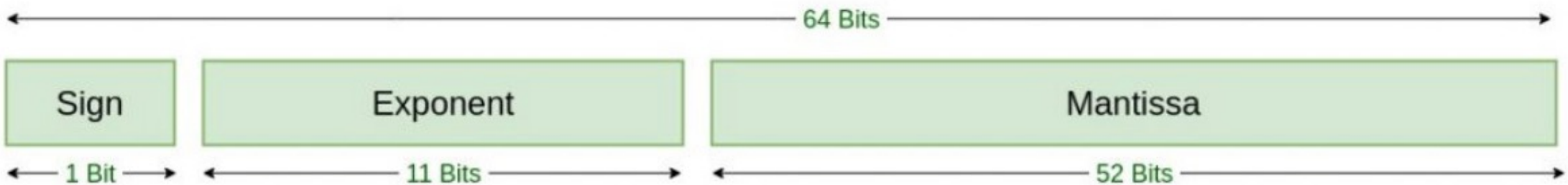
Only an approximation

Precission and Accuracy are used interchangeably.
Low precision means less accuracy. However, this not always true.
Forexample- 0.5 in Decimal any easily be represented accurately with 2 bits.

In floting point binary precision is governed by the no of bits allocated to The mantissa.

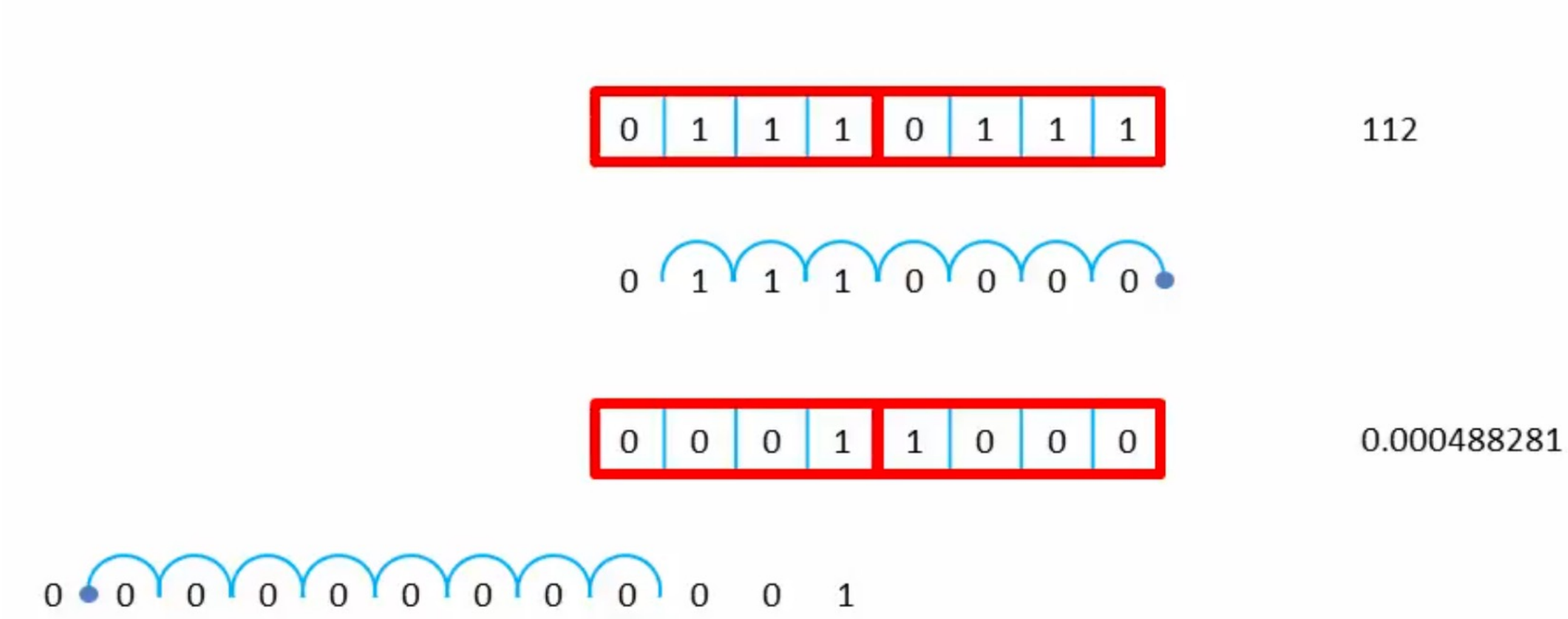
IEEE 754 numbers are divided into two based on the above three components: single precision and double precision.





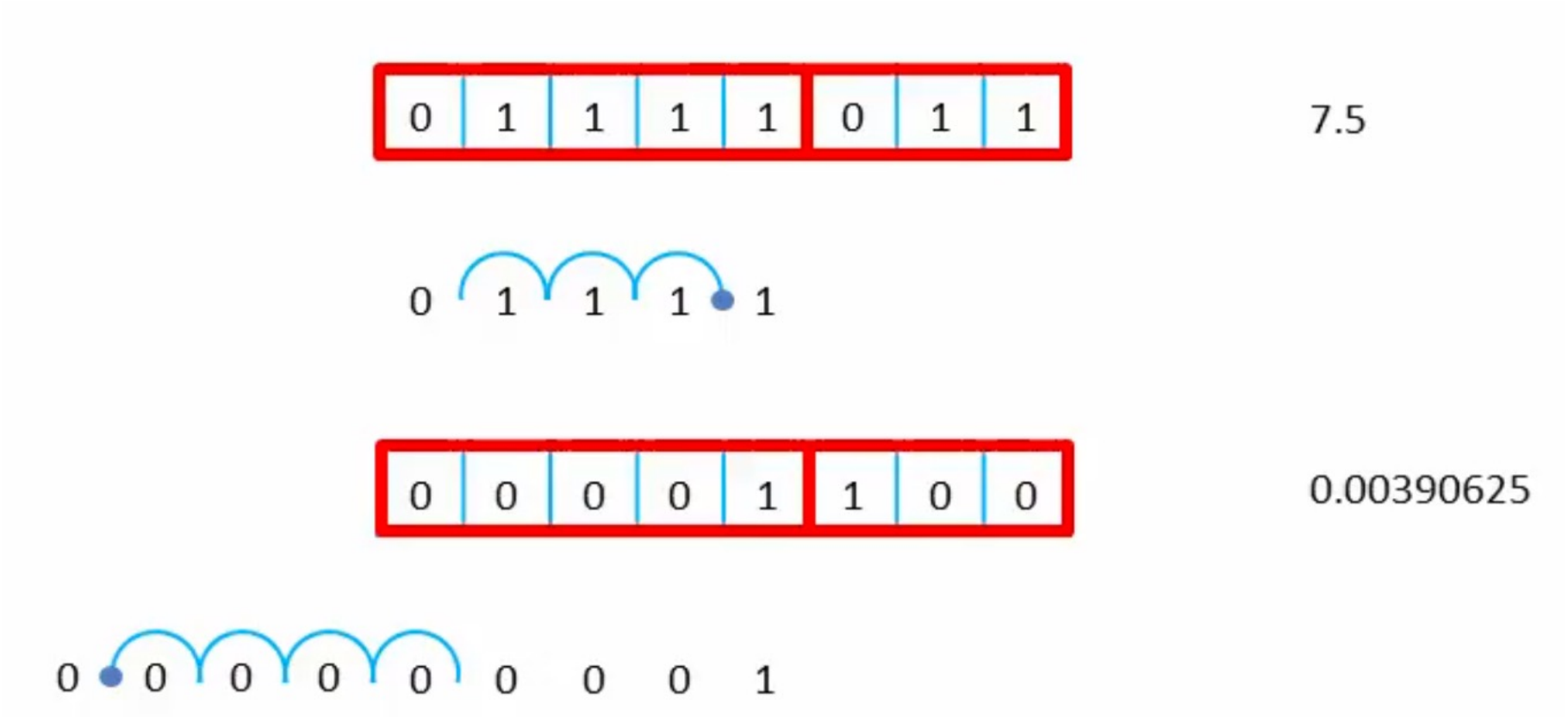
Double Precision
IEEE 754 Floating-Point Standard

Largest and Smallest Value Possible



Can we represent accuratley 7.5 in the register?

Largest and Smallest Value Possible



Summary

- For a given sized register, the number of values that can be represented is limited
- Greater precision comes at the expense of range
- Greater range comes at the expense of precision
- Accuracy often depends on precision, but not always
- There will always be values that can’t be represented accurately in binary
- Programmers should understand how floating point binary works

```
using System;

namespace FloatingNumber
{
    class Program
    {
        static void Main(string[] args)
        {
            float a = 12.345F;
            float b = 12;
            float c = a - b;
            Console.WriteLine(c);
        }
    }
}
```

```
using System;

namespace FloatingNumber
{
    class Program
    {
        static void Main(string[] args)
        {
            decimal a = 12.345M;
            decimal b = 12;
            decimal c = a - b;
            Console.WriteLine(c);

        }
    }
}
```

Floating Point Binary

Convert from Denary to Normalised Binary

$0\ 1\ 1\ 1\ 0\ 0\ 1\ 1$

$= 0\ .\ 1\ 1\ 1\ \ 0\ 0\ 1\ 1$

$= 0\ .\ 1\ 1\ 1\ \ \times 2^{3}$

$= 0\ \ 1\ 1\ 1\ .$

$= 0\ 1\ 1\ 1\ .\ 0$

$= 4 + 2 + 1$

$= 7$

$0\ 1\ 1\ 1\ 0\ 0\ 1\ 1_{2} = 7_{10}$

$0\ 1\ 1\ 1\ 1\ 1\ 1\ 0$

$= 0\ .\ 1\ 1\ 1\ \ 1\ 1\ 1\ 0$

$= 0\ .\ 1\ 1\ 1\ \ \times 2^{-2}$

$= .\ 0\ 0\ \ 1\ 1\ 1$

$= 0\ .\ 0\ 0\ 1\ 1\ 1$

$= 0.125 + 0.0625 + 0.03125$

$= 0.21875$

$0\ 1\ 1\ 1\ 1\ 1\ 1\ 0_{2} = 0.21875_{10}$

Convert positive denary numbers into floating point binary

With 4 bits for the mantissa and 4 bits for the exponent, both in two's complement, convert the value 7 into floating point binary

8	4	2	1	0.5
0	1	1	1	0

= 7.0

With 4 bits for the mantissa and 4 bits for the exponent, both in two's complement, convert the value 7 into floating point binary

0 1 1 1 . 0 = 7.0

x 2^3

8	4	2	1
0	0	1	1

0 . 1 1 1

With 4 bits for the mantissa and 4 bits for the exponent, both in two's complement, convert the value 7 into floating point binary

0 1 1 1 . 0 = 7.0

0 . 1 1 1 0 0 1 1

$7_{10} = 01110011_2$

With 4 bits for the mantissa and 4 bits for the exponent, both in two's complement, convert the value 1.5 into floating point binary

2	1	0.5
0	1	1

= 1.5

With 4 bits for the mantissa and 4 bits for the exponent, both in two's complement, convert the value 1.5 into floating point binary

0 1 . 1 = 1.5

0 1 1 0

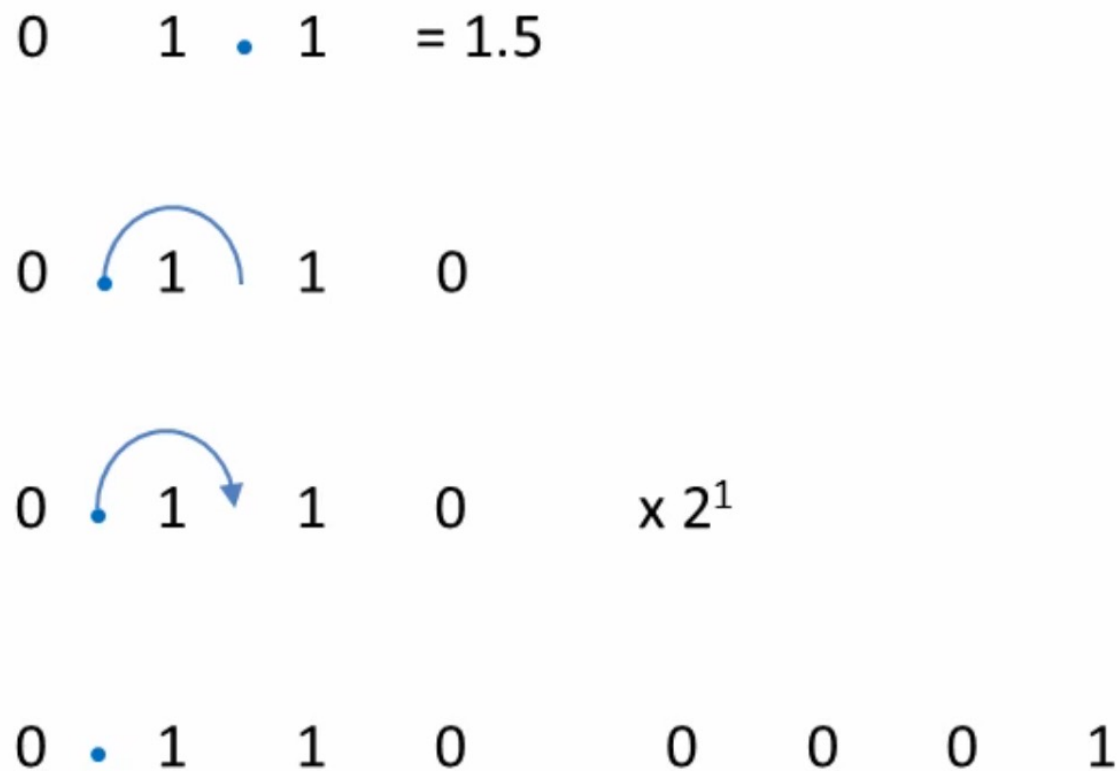
0 1 1 0

x 2^1

0 . 1 1 0

8	4	2	1
0	0	0	1

With 4 bits for the mantissa and 4 bits for the exponent, both in two's complement, convert the value 1.5 into floating point binary



$1.5_{10} = 01100001_2$

With 4 bits for the mantissa and 4 bits for the exponent, both in two's complement, convert the value 0.25 into floating point binary

1	0.5	0.25	
0	0	1	= 0.25

With 4 bits for the mantissa and 4 bits for the exponent, both in two's complement, convert the value 0.25 into floating point binary

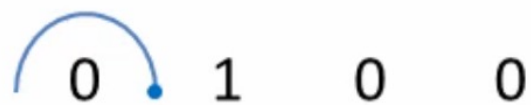
0 . 0 1 = 0.25

With 4 bits for the mantissa and 4 bits for the exponent, both in two's complement, convert the value 0.25 into floating point binary

0 . 0 1 = 0.25

With 4 bits for the mantissa and 4 bits for the exponent, both in two's complement, convert the value 0.25 into floating point binary

0 . 0 1 = 0.25



With 4 bits for the mantissa and 4 bits for the exponent, both in two's complement, convert the value 0.25 into floating point binary

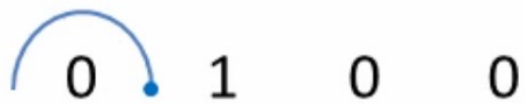
0 . 0 1 = 0.25

0 1 0 0

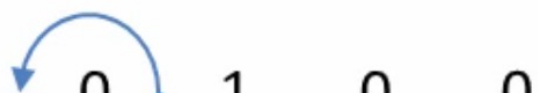
0 1 0 0 x 2^{-1}

With 4 bits for the mantissa and 4 bits for the exponent, both in two's complement, convert the value 0.25 into floating point binary

0 . 0 1 = 0.25



0 . 1 0 0



0 . 1 0 0 x 2^{-1}

- 8	4	2	1
1	1	1	1

With 4 bits for the mantissa and 4 bits for the exponent, both in two's complement, convert the value 0.25 into floating point binary

0 . 0 1 = 0.25

0 1 0 0

0 1 0 0 x 2^{-1}

0 . 1 0 0 1 1 1 1

$$0.25_{10} = 01001111_2$$

Exercise 11

Convert the following denary numbers into floating point binary. Assume 4 bits for the mantissa and 4 bits for the exponent, both in two's complement

2.5

1.75

0.375

Exercise 12

Convert the following denary numbers into floating point binary. Assume 10 bits for the mantissa and 6 bits for the exponent, both in two's complement

19.25

0.0625

Convert negative denary numbers into floating point binary

With 4 bits for the mantissa and 4 bits for the exponent, both in two's complement, convert the value -6 into floating point binary

-8	4	2	1	0.5
1	0	1	0	0

= -6.0

With 4 bits for the mantissa and 4 bits for the exponent, both in two's complement, convert the value -6 into floating point binary

1 0 1 0 . 0 = -6.0

x 2^3

1 . 0 1 0 0 0 1 1

$-6_{10} = 10100011_2$

With 4 bits for the mantissa and 4 bits for the exponent, both in two's complement, convert the value -1.5 into floating point binary

-2	1	0.5	
1	0	1	= -1.5

With 4 bits for the mantissa and 4 bits for the exponent, both in two's complement, convert the value -1.5 into floating point binary

-2	1	0.5
1	0 .	1

= -1.5

-4	2	1	0.5
1	1	0 .	1

= -1.5

With 4 bits for the mantissa and 4 bits for the exponent, both in two's complement, convert the value -1.5 into floating point binary

-2	1	0.5
1	0 .	1

= -1.5

-8	4	2	1	0.5
1	1	1	0 .	1

= -1.5

With 4 bits for the mantissa and 4 bits for the exponent, both in two's complement, convert the value -1.5 into floating point binary

1 0 . 1 = -1.5

1 0 1 0

1 0 1 0 x 2^1

1 . 0 1 0 0 0 0 1

$$-1.5_{10} = 10100001_2$$

Exercise 13

Convert the following denary numbers into floating point binary. Assume 4 bits for the mantissa and 4 bits for the exponent, both in two's complement

-4 -0.25 -1.75

Exercise 14

Convert the following denary numbers into floating point binary. Assume 10 bits for the mantissa and 6 bits for the exponent, both in two's complement

-12.75

-0.125

Normalisation

4 bit Mantissa + 4 bit Exponent

0 1 0 0 1 1 1 1	0 0 0 1 0 0 0 1	0 0 1 0 0 0 0 0
= 0 . 1 0 0 1 1 1 1	= 0 . 0 0 1 0 0 0 1	= 0 . 0 1 0 0 0 0 0
= 0 . 1 0 0 x 2^{-1}	= 0 . 0 0 1 x 2^{1}	= 0 . 0 1 0 x 2^{0}
= .0 1 0 0	= 0 0.0 1	= 0 . 0 1 0
= 0 . 0 1 0 0	= 0 0 . 0 1	= 0 . 0 1 0
= 0.25	= 0.25	= 0.25

Speed of Light in Vacuum (m/s)

2.99×10^{8}

299000000

29.9×10^{7}

$299. \times 10^{6}$

29.9×10^{7}

2.99×10^{8}

0.29×10^{9}

0.02×10^{10}

Normalized form – best balance between precision and range

$299. \times 10^{6}$

29.9×10^{7}

2.99×10^{8}

0.29×10^{9}

0.02×10^{10}

0 1 0 0 1 1 1 1	0 0 0 1 0 0 0 1	0 0 1 0 0 0 0 0
= 0 . 1 0 0 1 1 1 1	= 0 . 0 0 1 0 0 0 1	= 0 . 0 1 0 0 0 0 0
= 0 . 1 0 0 x 2^{-1}	= 0 . 0 0 1 x 2^{1}	= 0 . 0 1 0 x 2^{0}
= .0 1 0 0	= 0 0.0 1	= 0 : 0 1 0
= 0 . 0 1 0 0	= 0 0 . 0 1	= 0 . 0 1 0
= 0.25	= 0.25	= 0.25

Normalized form – sign bit 0 is followed immediately by 1

Objectives of normalisation

- Maximise precision
- Unambiguous representation
- Simplify arithmetic

- For POSITIVE numbers, the normalised form starts with a 0 followed immediately by a 1

- For NEGATIVE numbers, the normalised form starts with a 1 followed immediately by a 0

The floating point binary number 00010001 is stored using 4 bits for the mantissa and 4 bits for the exponent, both in two's complement. Normalise it.

$$0\,0\,0\,1\,0\,0\,0\,1$$

$$= 0.0\,0\,1 \quad 0\,0\,0\,1$$

$$= 0.0\,0\,1 \times 2^{1}$$

$$= 0 \quad 0\,0.1$$

$$= 0.1\,0\,0$$

$$= 0.1\,0\,0 \times 2^{1-2=-1}$$

$$= 0\,1\,0\,0\,1\,1\,1\,1$$

The floating point binary number 00111111 is stored using 4 bits for the mantissa and 4 bits for the exponent, both in two's complement. Normalise it.

$$0\,0\,1\,1\,1\,1\,1\,1$$

$$= 0\,.\,0\,1\,1 \quad 1\,1\,1\,1$$

$$= 0\,.\,0\,1\,1 \quad \times 2^{-1}$$

$$= 0 \quad 0.1\,1$$

$$= 0\,.\,1\,1\,0$$

$$= 0\,.\,1\,1\,0 \quad \times 2^{-1\,-\,1\,=\,-2}$$

$$= 0\,1\,1\,0\,1\,1\,1\,0$$

The floating point binary number 11000001 is stored using 4 bits for the mantissa and 4 bits for the exponent, both in two's complement. Normalise it.

11000001

= 1 . 1 0 0 0 0 0 0 1

= 1 . 1 0 0 x 2^{1}

= 1 1.0 0

= 1 . 0 0 0

= 1 . 0 0 0 x $2^{1 - 1 = 0}$

= 1 0 0 0 0 0 0 0

The floating point binary number 11101011 is stored using 5 bits for the mantissa and 3 bits for the exponent, both in two's complement. Normalise it.

$$11101011$$

$$= 1.1101\ \ 011$$

$$= 1.1101 \times 2^{3}$$

$$= 1\ \ 11.01$$

$$= 1.0100$$

$$= 1.0100 \times 2^{3-2=1}$$

$$= 10100001$$

Exercise 15

The following floating point binary numbers are stored using 5 bits for the mantissa and 3 bits for the exponent, Normalise them.

11011001

00011111

Exercise 16

The following floating point binary numbers are stored using 10 bits for the mantissa and 6 bits for the exponent, Normalise them.

0000000110000111

1111101000001011

Floating Point Binary Addition

$5.2 \times 10^3 + 2.34 \times 10^3 = 7.54 \times 10^3$

$$\begin{array}{r} 5.2 \\ +\ 2.34 \\ \hline 7.54 \end{array} \times 10^3$$

$$5.2 \times 10^4 + 2.34 \times 10^3 = 5.434 \times 10^4$$

$$0.234 \times 10^4$$

$$\begin{array}{r} 5.2 \\ +\ 0.234 \\ \hline 5.434 \end{array} \times 10^4$$

$8.2 \times 10^4 + 123.25 \times 10^3 = 2.0525 \times 10^5$

12.325×10^4

8.2

$+ 12.325$

20.525×10^4

2.0525×10^5

Floating point binary addition

- Make sure both numbers are normalised
- Make exponents the same
- Add mantissas together
- Normalise result if necessary

Show how you would add 0100000011 to 0100100010. Both numbers are in floating point binary format using 6 bits for the mantissa and 4 bits for the exponent, both in two's complement. Show your result in the same format.

010000 0011	+	010010 0010	
0.10000×2^3	+	0.10010×2^2	
0.10000×2^3	+	$0\,0.10010 \times 2^3$	Make exponents the same
0.10000×2^3	+	0.01001×2^3	

$$\begin{array}{r} 0.10000 \\ +\ 0.01001 \\ \hline 0.11001 \end{array}$$

Add mantissas together

0.11001×2^3

011001 0011 Result already normalised

Double check result

$0.10000 \times 2^3 = 100.0 = 4$

$0.10010 \times 2^2 = 10.01 = 2.25$

$4 + 2.25 = 6.25$

$6.25 = 110.01$

$110.01 = 0.11001 \times 2^3 = 011001\ \ 0011$

Show how you would add 010000001111 to 010101000100. Both numbers are in floating point binary format using 8 bits for the mantissa and 4 bits for the exponent, both in two's complement. Show your result in the same format.

01000000 1111	+	01010100 0100	
0.1000000×2^{-1}	+	0.1010100×2^{4}	
0 0 0 0 0 0.1000000×2^{4}	+	0.1010100×2^{4}	Make exponents the same
0.0000010×2^{4}	+	0.1010100×2^{4}	

0.0000010	Add mantissas together
+ 0.1010100	
0.1010110	

0.1010110×2^{4}

01010110 0100 Result already normalised

Double check result

$0.1000000 \times 2^{-1} = 0.01 = 0.25$

$0.1010100 \times 2^{4} = 1010.1 = 10.5$

$0.25 + 10.5 = 10.75$

$10.75 = 1010.11$

$1010.11 = 0.101011 \times 2^{4} = 01010110\ 0100$

Show how you would add 0100100100 to 0100100010. Both numbers are in floating point binary format using 6 bits for the mantissa and 4 bits for the exponent, both in two's complement. Show your result in the same format.

010010 0100	+	010010 0010	
0.10010×2^4	+	0.10010×2^2	
0.10010×2^4	+	$0.0\,0.10010 \times 2^4$	Make exponents the same
0.10010×2^4	+	0.0010010×2^4	TRUNCATION ERROR

0.10010
+ 0.00100
0.10110

0.10110×2^4

010110 0100 Normalise result

Double check result

$0.10010 \times 2^4 = 1001.0 = 9$

$0.10010 \times 2^2 = 10.010 = 2.25$

9 + 2.25 = 11.25

11.25 = 1011.01

1011.01 = 0.101101×2^4 = 0101101 0100

Convert result to denary

010110 0100 = $0.10110 \times 2^4 = 1011.0 = 11$

Exercise 17

Show how you would add 0101000110 to 0010100101. Both numbers are in floating point binary format using 6 bits for the mantissa and 4 bits for the exponent, both in two's complement. Show your result in the same format.

Show how you would add 0100100011 to 1001000010. Both numbers are in floating point binary format using 6 bits for the mantissa and 4 bits for the exponent, both in two's complement. Show your result in the same format.

Floating point binary addition

- Make sure both numbers are normalised
- Make smaller exponent match the larger exponent
- Add mantissas together
- Normalise result if necessary

Floating point binary subtraction

- Make sure both numbers are normalised
- Make exponents the same
- If performing subtraction, negate the number to subtract
- Add mantissas together
- Normalise result if necessary

Show how you would subtract 0110000010 from 0111000011. Both numbers are in floating point binary format using 6 bits for the mantissa and 4 bits for the exponent, both in two's complement. Show your result in the same format.

011100 0011	-	011000 0010	
0.11100×2^3	-	0.11000×2^2	
0.11100×2^3	-	$0\,0.11000 \times 2^3$	Make exponents the same
0.11100×2^3	-	0.01100×2^3	
	invert bits	1.10011	Negate the number to subtract
	add 1	+ 1	
		1.10100	
0.11100×2^3	+	1.10100×2^3	
		0.11100	Add mantissas together
	carry 1 overflow	+ 1.10100	
		0.10000	
		0.10000×2^3	
		010000 0011	Result already normalised

Double check result

$0.11100 \times 2^3 = 111.0 = 7$

$0.11000 \times 2^2 = 11.0 = 3$

$7 - 3 = 4$

$4 = 100.0$

$100.0 = 0.10000 \times 2^3 = 0.10000\ \ 0011$

Show how you would subtract 0100100010 from 0100100100. Both numbers are in floating point binary format using 6 bits for the mantissa and 4 bits for the exponent, both in two's complement. Show your result in the same format.

010010 0100 - 010010 0010

0.10010×2^4 - 0.10010×2^2

0.10010×2^4 - $0.0\,0.10010 \times 2^4$ Make exponents the same

0.10010×2^4 - 0.00100×2^4 TRUNCATION ERROR

invert bits 1.11011 Negate the number to subtract

add 1 + 1

1.11100

0.10010×2^4 + 1.11100×2^4

0.10010 Add mantissas together

carry 1 overflow + 1.11100

0.01110

0.01110×2^4

0.01110×2^3 Normalise result

0.11100×2^3

011100 0011

Double check result

$0.10010 \times 2^4 = 1001.0 = 9$

$0.10010 \times 2^2 = 10.010 = 2.25$

9 - 2.25 = 6.75

6.75 = 0110.11

$0110.11 = 0.11011 \times 2^3 = 011011\ \ 0011$

Show how you would subtract 0100100010 from 0100100100. Both numbers are in floating point binary format using 6 bits for the mantissa and 4 bits for the exponent, both in two's complement. Show your result in the same format.

010010 0100 - 010010 0010

0.10010×2^4 - 0.10010×2^2

0.10010×2^4 - $0.0\,0.10010 \times 2^4$ Make exponents the same

0.10010×2^4 - 0.00100×2^4 TRUNCATION ERROR

invert bits 1.11011 Negate the number to subtract

add 1 + 1

1.11100

0.10010×2^4 + 1.11100×2^4

0.10010 Add mantissas together

carry 1 overflow + 1.11100

0.01110

0.01110×2^4

0.01110×2^3 Normalise result

0.11100×2^3

011100 0011

Double check result

$0.10010 \times 2^4 = 1001.0 = 9$

$0.10010 \times 2^2 = 10.010 = 2.25$

9 - 2.25 = 6.75

6.75 = 0110.11

$0110.11 = 0.11011 \times 2^3 = 011011\ \ 0011$

Convert result to denary

$0.11100 \times 2^3 = 111.0 = 7$

Exercise 18

Show how you would subtract 010000100101 from 010001000110. Both numbers are in floating point binary format using 8 bits for the mantissa and 4 bits for the exponent, both in two's complement. Show your result in the same format.

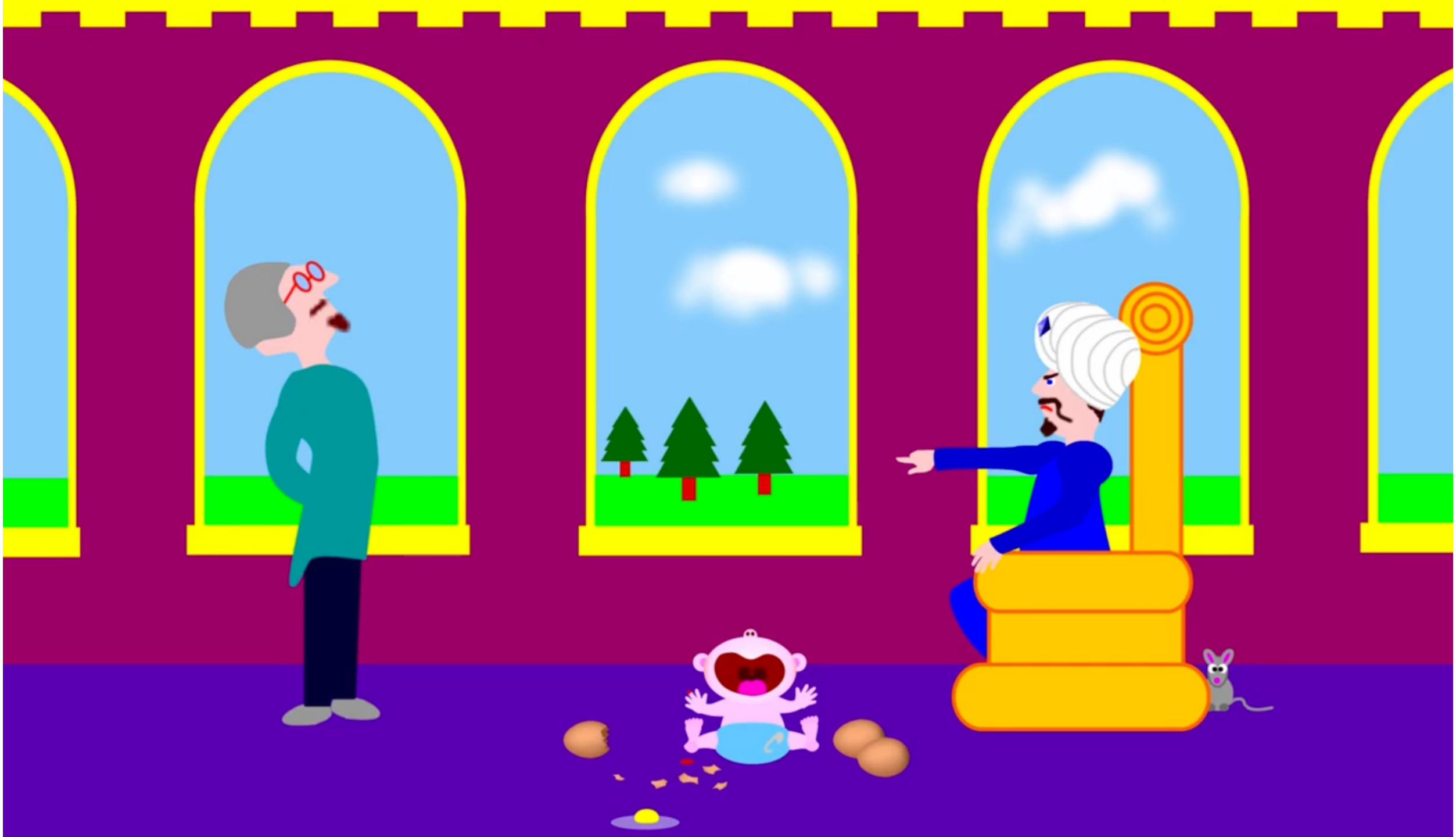
Show how you would subtract 010010000011 from 110100001111. Both numbers are in floating point binary format using 8 bits for the mantissa and 4 bits for the exponent, both in two's complement. Show your result in the same format.

Floating point binary addition

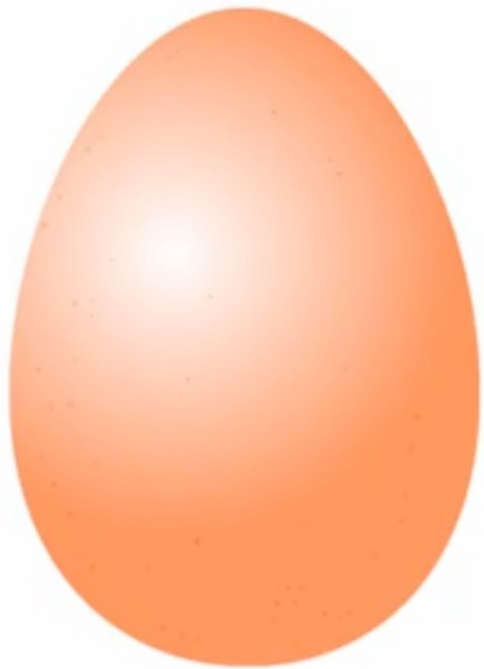
- Make sure both numbers are normalised
- Make smaller exponent match the larger exponent
- Add mantissas together
- Normalise result if necessary

Byte Ordering

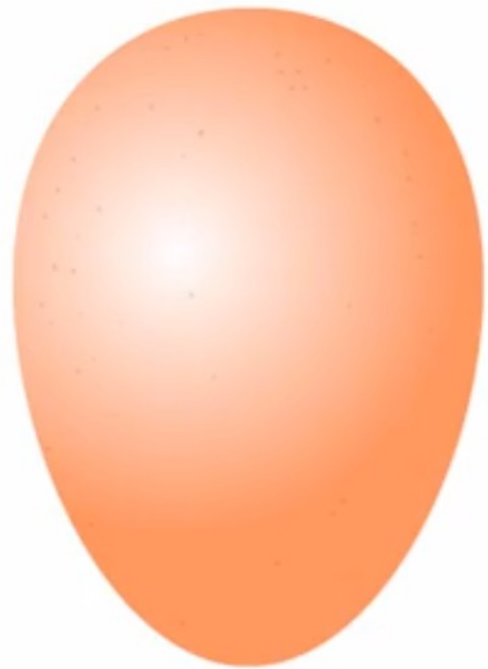
Endianness



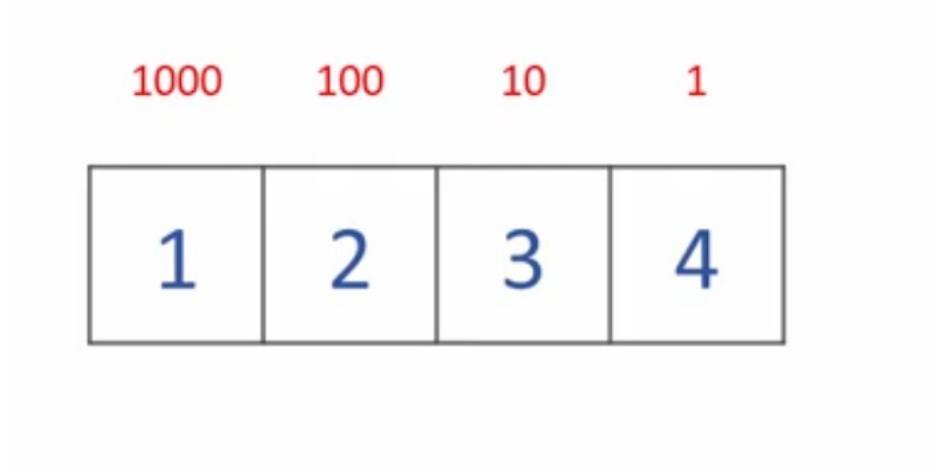
ON HOLY WARS AND A PLEA FOR PEACE

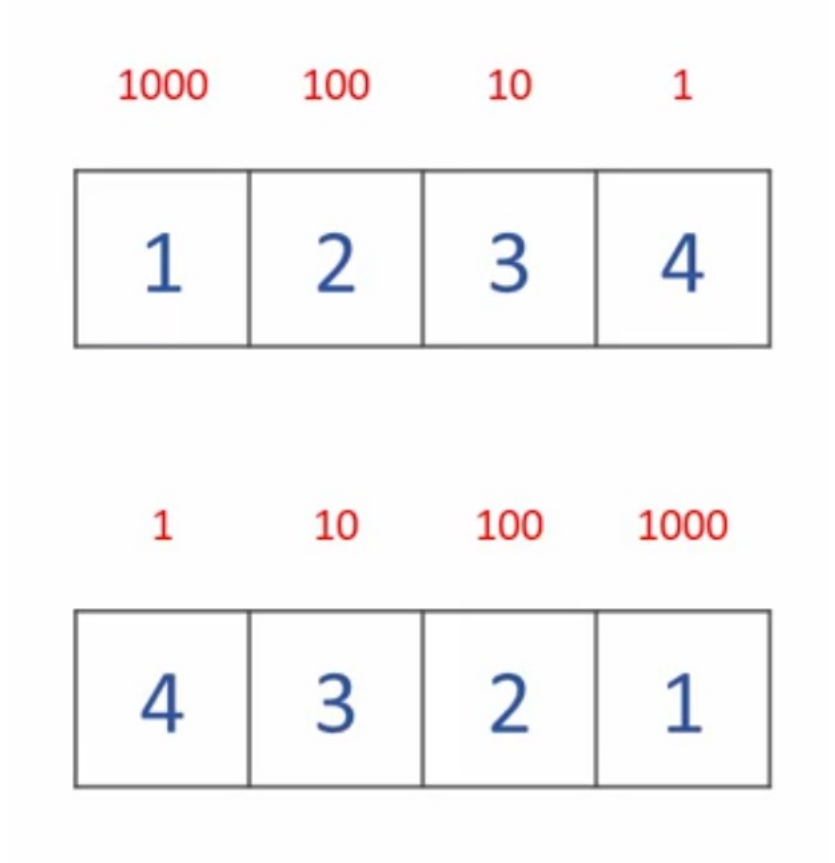


Little Endian

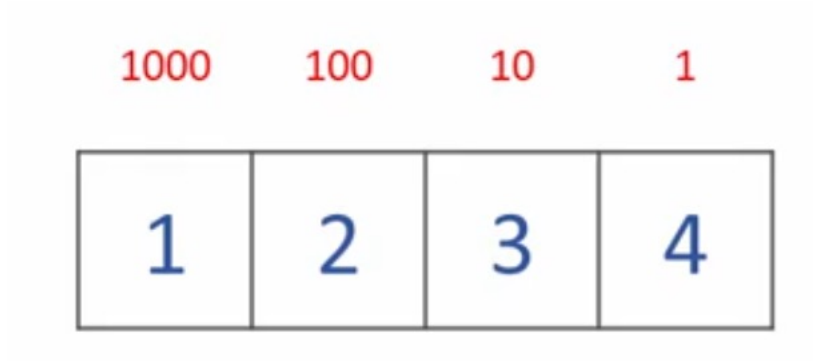


Big Endian

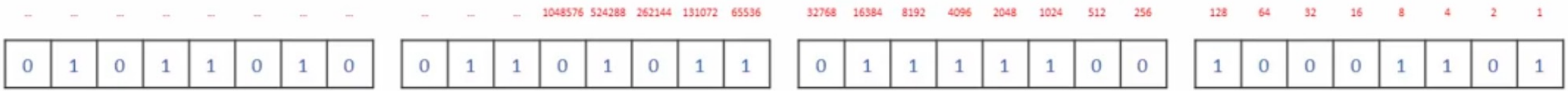

1000
100
10
1
1
2
3
4


1000
100
10
1
1
2
3
4
1
10
100
1000
4
3
2
1

Decimal system, also called **Hindu-Arabic number system** or **Arabic number system**, in mathematics, positional numeral system employing 10 as the base and requiring 10 different numerals, the digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. It also requires a dot (decimal point) to represent decimal fractions. In this scheme, the numerals used in denoting a number take different place values depending upon position. In a base-10 system the number 543.21 represents the sum $(5 \times 10^{2}) + (4 \times 10^{1}) + (3 \times 10^{0}) + (2 \times 10^{-1}) + (1 \times 10^{-2})$. *See* numerals and numeral systems.



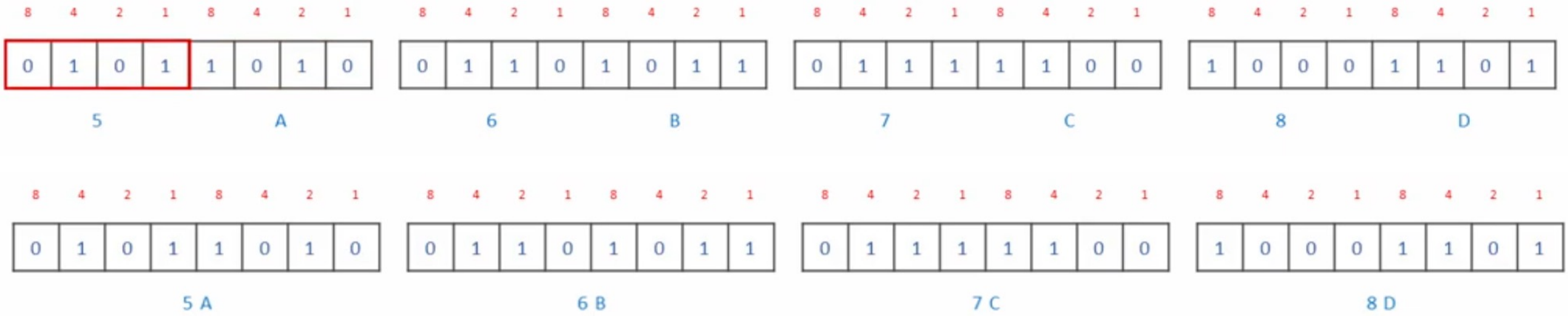
$64 + 16 + 8 + 4 + 1 = 93_{10}$



$= 1516993677_{10}$

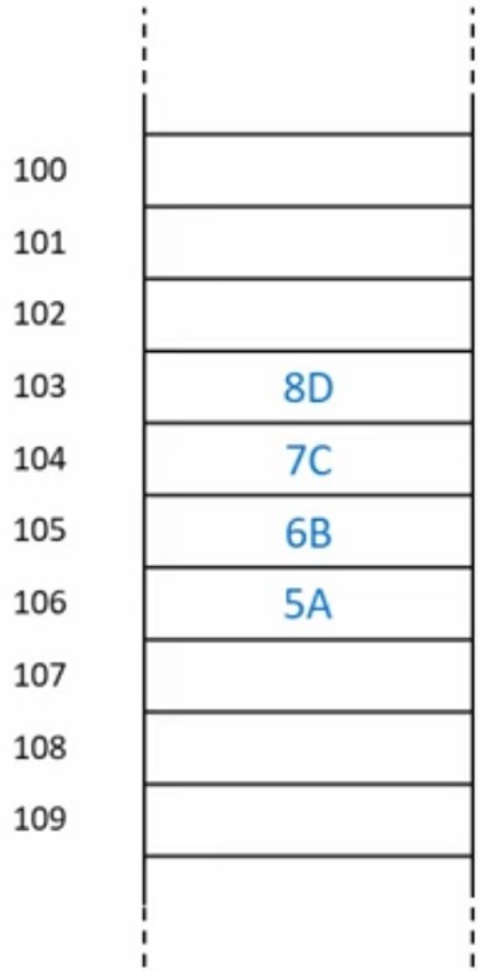
$= 5A6B7C8D_{16}$

Hexadecimal and Binary

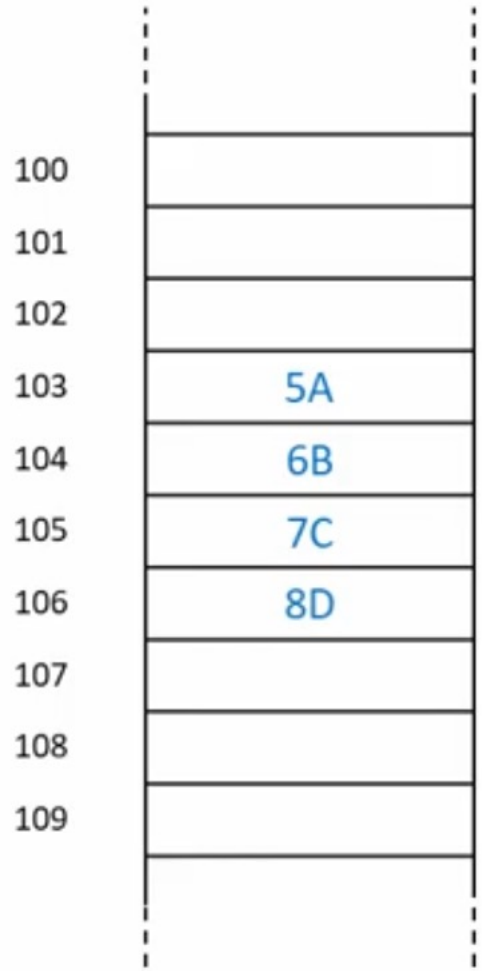


5A 6B 7C 8D

Least significant byte at lower address



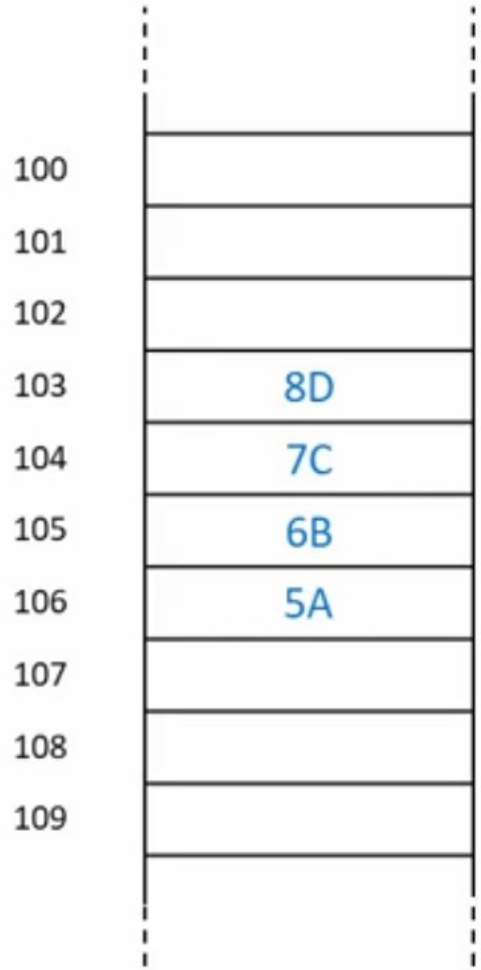
Least significant byte at higher address



5A 6B 7C 8D

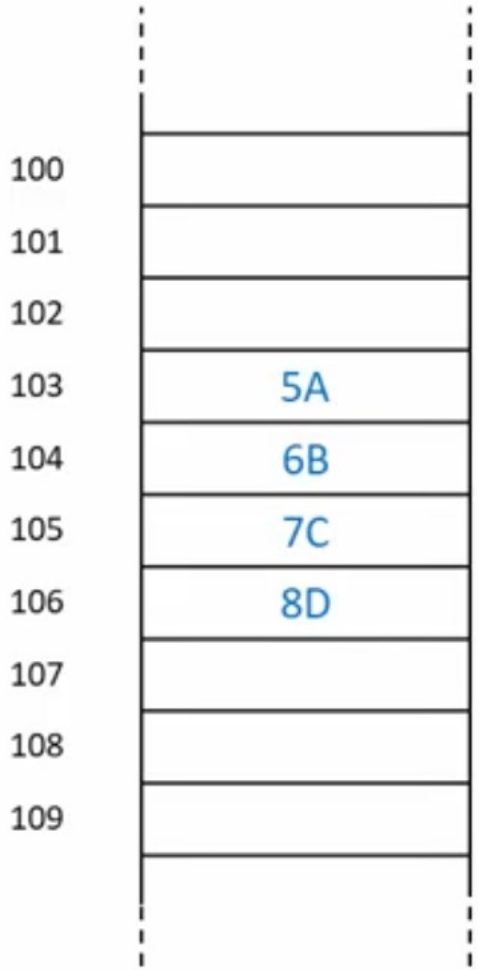
Little endian

Least significant byte at lower address



Big endian

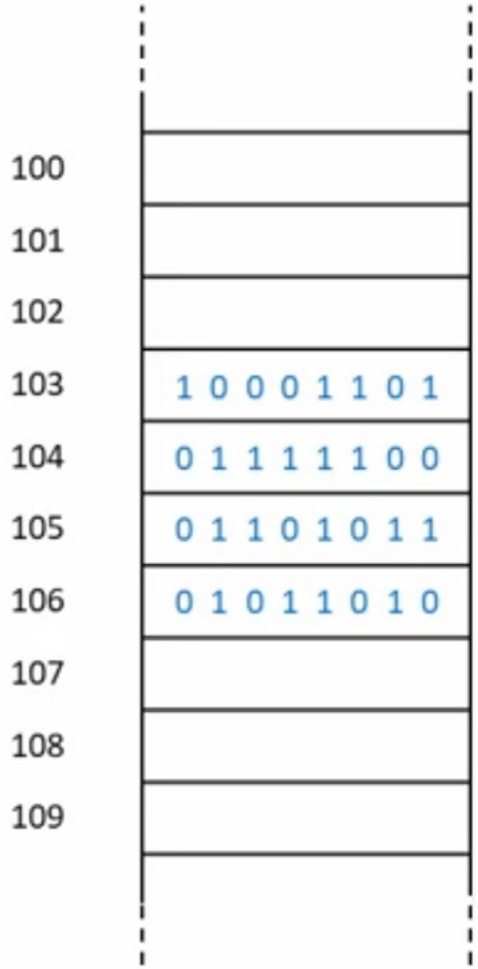
Least significant byte at higher address



5A 6B 7C 8D

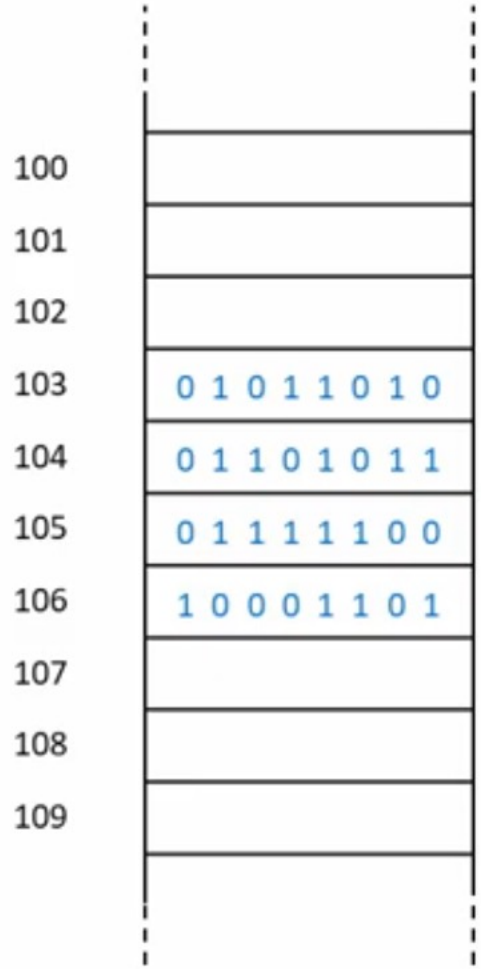
Little endian

Least significant byte at lower address



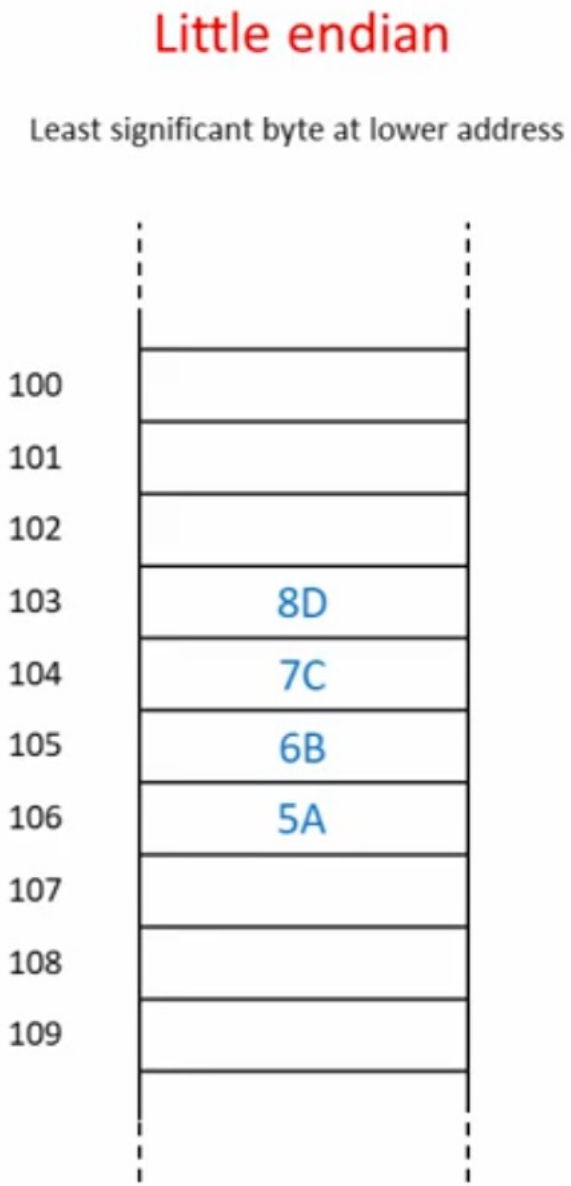
Big endian

Least significant byte at higher address



5A 6B 7C 8D

Erlier Main Frame + IBM + Apple are Big Endian
Intel processor are Little Endian due to efficiency
Of some processes



106	105	104	103
5A	6B	7C	8D
0 1 0 1 1 0 1 0	0 1 1 0 1 0 1 1	0 1 1 1 1 1 0 0	1 0 0 0 1 1 0 1

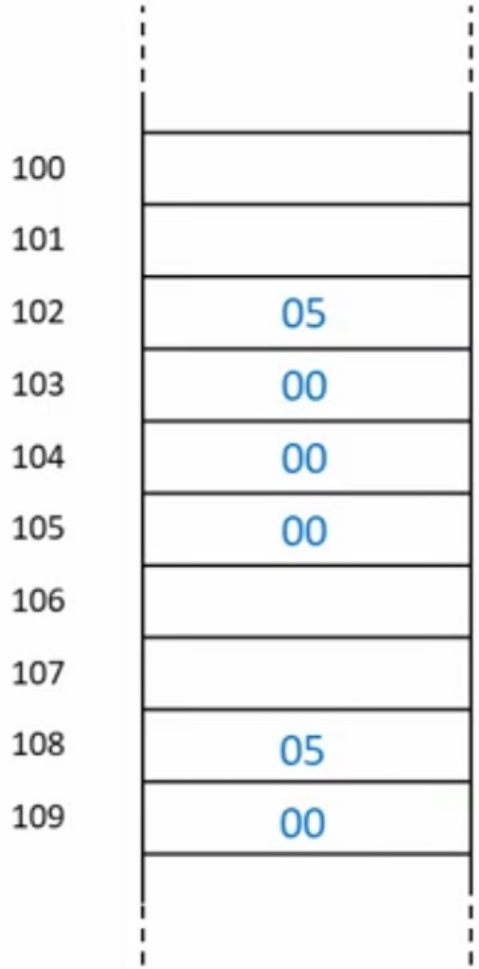
206	205	204	203
B2	C2	D2	E2
1 0 1 1 0 0 1 0	1 1 0 0 0 0 1 0	1 1 0 1 0 0 1 0	1 1 1 0 0 0 1 0

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

```
Dim ShoeSize As Integer

ShoeSize = 5

Dim NewShoeSize as Short

NewShoeSize = CShort(ShoeSize)
```

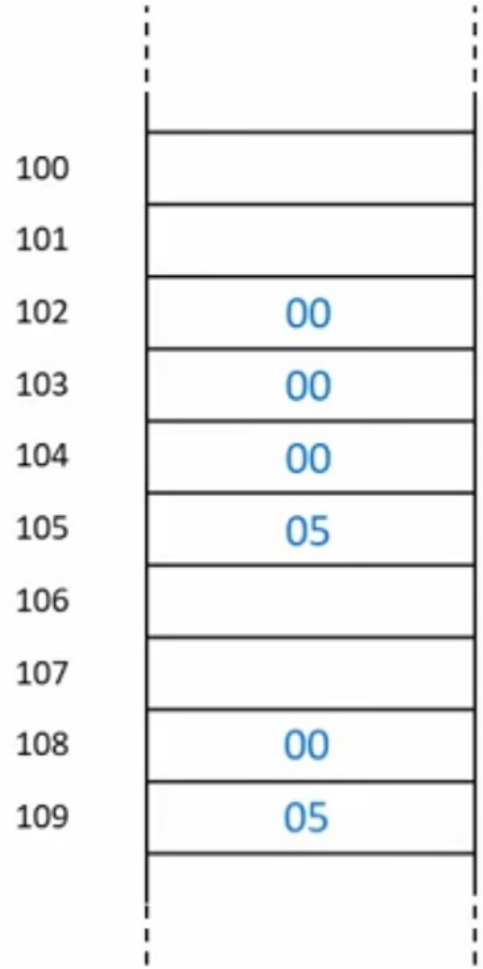
Little endian

Least significant byte at lower address

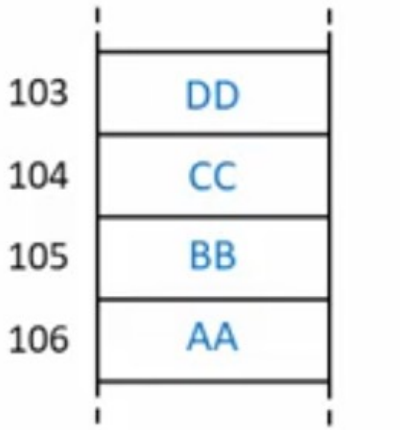


Big endian

Least significant byte at higher address



Network Byte Order (Big Endian)



Network Byte Order (Big Endian)

"We agree that the difference between sending eggs with the little- or the big-end first is trivial, but we insist that everyone must do it in the same way, to avoid anarchy. Since the difference is trivial we may choose either way, but a decision must be made." Danny Cohen 1 April 1980


JPEG
Big Endian
GIF
Little Endian
PNG
Big Endian
BMP
Little Endian
MPEG-4
Big Endian

Big Endian

Little Endian

Big Endian

Little Endian

Big Endian

Either

Unicode Transformation Formats and Byte Order

Summary

- Normalisation maximises precision
- Normalisation allows for an unambiguous representation
- For positive numbers, the normalised form starts with 01
- For negative numbers, the normalised form starts with 10

IEEE 754 Standard for Floating-Point Arithmetic

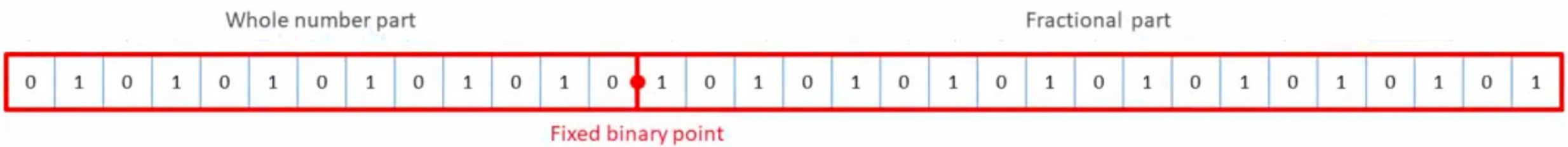
Representing 32 bit Single Precision Numbers

Representing Real Numbers

Fixed point binary

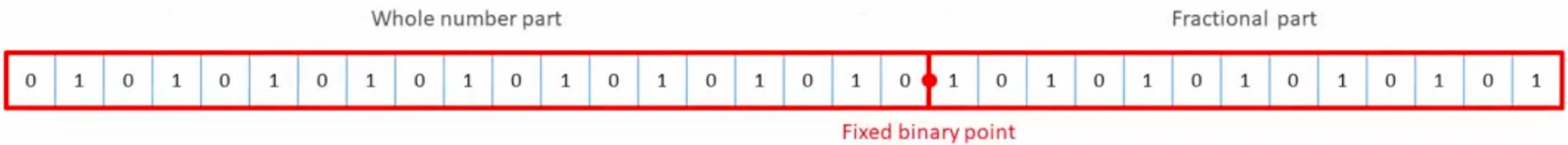
Representing Real Numbers

Fixed point binary



Representing Real Numbers

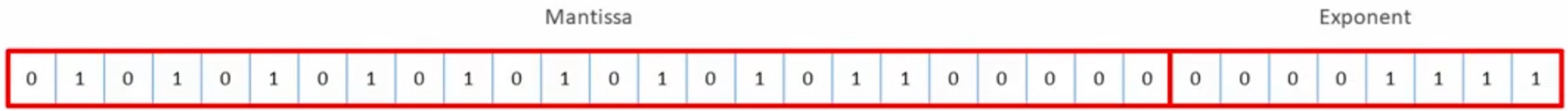
Fixed point binary



Representing Real Numbers

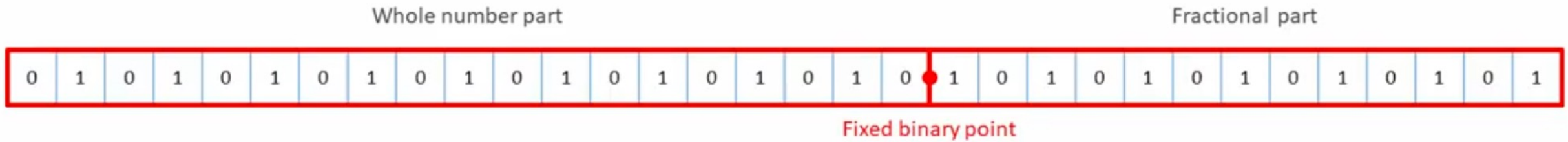
Fixed point binary

Floating point binary

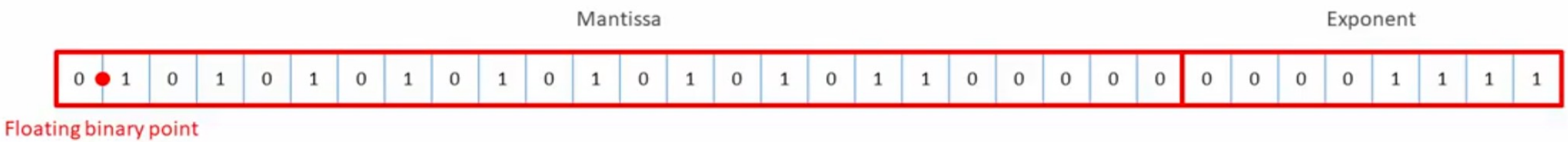


Representing Real Numbers

Fixed point binary

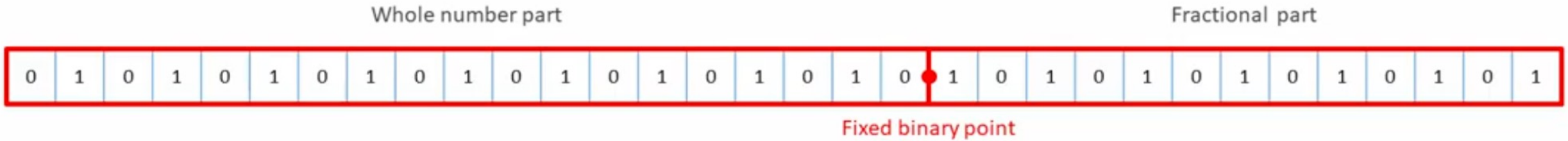


Floating point binary

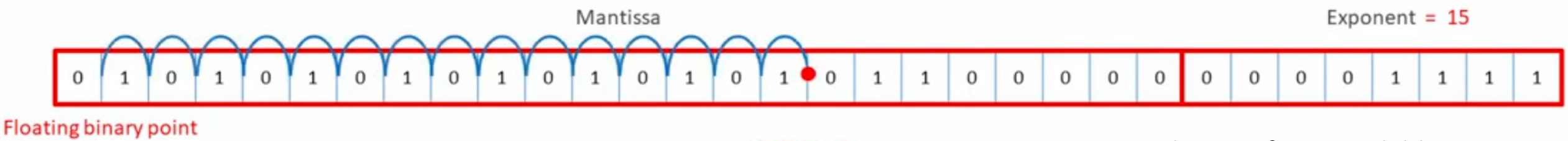


Representing Real Numbers

Fixed point binary



Floating point binary



Precision is govern by No of Bits available Mantisa

21845.375_{10}

Range is govern by No of Bits available Exponent

Negative values are represented by the Tow's Compliment

IEEE 754 Standard for Single Precision 32 bit floating point binary

Convert the denary value 19.59375 into IEEE 754 standard 32 bit floating point binary

Step 1: Determine the sign bit (0 if positive, 1 if negative)

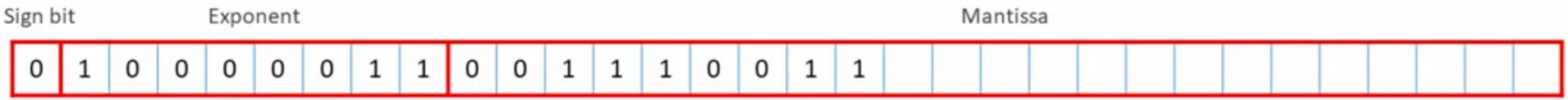
Sign bit = 0

Step 2: Convert to pure binary

	16	8	4	2	1	0.5	0.25	0.125	0.0625	0.03125
19.59375_{10} =	1	0	0	1	1 •	1	0	0	1	1

19	÷	2	=	9	remainder	1	0.59375	x	2	=	1.1875	1
9	÷	2	=	4	remainder	1	0.1875	x	2	=	0.375	0
4	÷	2	=	2	remainder	0	0.375	x	2	=	0.75	0
2	÷	2	=	1	remainder	0	0.75	x	2	=	1.5	1
					remainder	1	0.5	x	2	=	1.0	1

Convert the denary value 19.25 into IEEE 754 standard 32 bit floating point binary



Step 1: Determine the sign bit (0 if positive, 1 if negative)

Sign bit = 0

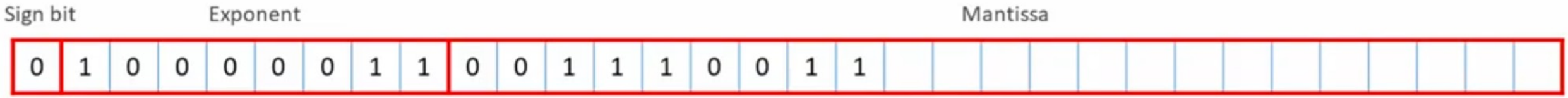
Step 2: Convert to pure binary

	16	8	4	2	1	0.5	0.25	0.125	0.0625	0.03125
19.59375_{10} =	1	0	0	1	1	1	0	0	1	1

Step 3: Normalise to determine the mantissa and the unbiased exponent ***(place the binary point after leftmost 1)***

1 0 0 1 1 . 1 0 0 1 1 = 1.001110011×2^4

Convert the denary value 19.25 into IEEE 754 standard 32 bit floating point binary



Step 1: Determine the sign bit (0 if positive, 1 if negative)

Sign bit = 0

Step 2: Convert to pure binary

	16	8	4	2	1	0.5	0.25	0.125	0.0625	0.03125
19.59375_{10} =	1	0	0	1	1	1	0	0	1	1

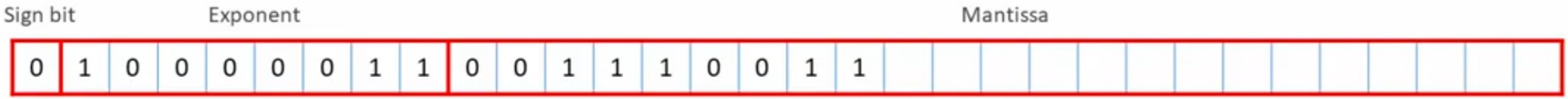
Step 3: Normalise to determine the mantissa and the unbiased exponent ***(place the binary point after leftmost 1)***

1 0 0 1 1 1 0 0 1 1 = 1.001110011×2^4

Step 4: Determine the biased exponent ***(add 127 then convert to an unsigned binary integer)***

$4 + 127 = 131_{10} = 10000011_2$

Convert the denary value 19.25 into IEEE 754 standard 32 bit floating point binary



Step 1: Determine the sign bit (0 if positive, 1 if negative)

Sign bit = 0

Step 2: Convert to pure binary

	16	8	4	2	1	0.5	0.25	0.125	0.0625	0.03125
19.59375_{10} =	1	0	0	1	1	1	0	0	1	1

Step 3: Normalise to determine the mantissa and the unbiased exponent ***(place the binary point after leftmost 1)***

1 0 0 1 1 1 0 0 1 1 = 1.001110011×2^4

Step 4: Determine the biased exponent ***(add 127 then convert to an unsigned binary integer)***

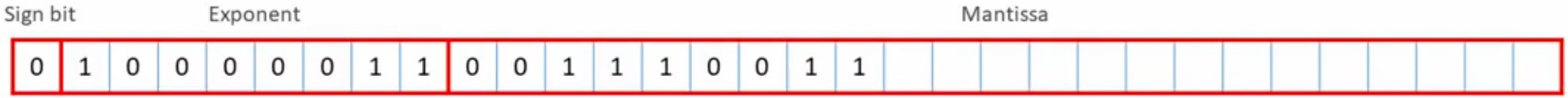
$4 + 127 = 131_{10} = 10000011_2$

Step 5: Remove the leading 1 from the mantissa ***(remove the leftmost 1)***

1.001110011 = 001110011

Left most bit of the mantisa will always be 1 – therefore no need to store it. Just put back when doing any calculations.

Convert the denary value 19.25 into IEEE 754 standard 32 bit floating point binary



Step 1: Determine the sign bit (0 if positive, 1 if negative)

Sign bit = 0

Step 2: Convert to pure binary

	16	8	4	2	1	0.5	0.25	0.125	0.0625	0.03125
19.59375_{10} =	1	0	0	1	1	1	0	0	1	1

Step 3: Normalise to determine the mantissa and the unbiased exponent ***(place the binary point after leftmost 1)***

1 0 0 1 1 • 1 0 0 1 1 = 1.001110011×2^4

Step 4: Determine the biased exponent ***(add 127 then convert to an unsigned binary integer)***

$4 + 127 = 131_{10} = 10000011_2$

Step 5: Remove the leading 1 from the mantissa ***(remove the leftmost 1)***

1.001110011 = 001110011

Convert the denary value 19.25 into IEEE 754 standard 32 bit floating point binary

Sign bit	Exponent								Mantissa																			
0	1	0	0	0	0	0	1	1	0	0	1	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0

Step 1: Determine the sign bit (0 if positive, 1 if negative)

Sign bit = 0

Step 2: Convert to pure binary

	16	8	4	2	1	0.5	0.25	0.125	0.0625	0.03125
19.59375_{10} =	1	0	0	1	1 •	1	0	0	1	1

Step 3: Normalise to determine the mantissa and the unbiased exponent ***(place the binary point after leftmost 1)***

1 0 0 1 1 • 1 0 0 1 1 = 1.001110011×2^4

Step 4: Determine the biased exponent ***(add 127 then convert to an unsigned binary integer)***

$4 + 127 = 131_{10} = 10000011_2$

Step 5: Remove the leading 1 from the mantissa ***(remove the leftmost 1)***

1.001110011 = 001110011

Baised Exponent – 4 Bit Exponent

0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

8	4	2	1
0	0	1	1

$= 3_{10}$

0	0	0	0	0	0
0	0	0	1	1	1
0	0	1	0	2	2
0	0	1	1	3	3
0	1	0	0	4	4
0	1	0	1	5	5
0	1	1	0	6	6
0	1	1	1	7	7
1	0	0	0	8	-8
1	0	0	1	9	-7
1	0	1	0	10	-6
1	0	1	1	11	-5
1	1	0	0	12	-4
1	1	0	1	13	-3
1	1	1	0	14	-2
1	1	1	1	15	-1

-8	4	2	1	
1	0	1	1	= -5_{10}

0 0 0 0	0	0
0 0 0 1	1	1
0 0 1 0	2	2
0 0 1 1	3	3
0 1 0 0	4	4
0 1 0 1	5	5
0 1 1 0	6	6
0 1 1 1	7	7
1 0 0 0	8	-8
1 0 0 1	9	-7
1 0 1 0	10	-6
1 0 1 1	11	-5
1 1 0 0	12	-4
1 1 0 1	13	-3
1 1 1 0	14	-2
1 1 1 1	15	-1

-8	4	2	1	
1	0	1	1	$= -5_{10}$

0	0	0	0	0	0	-5
0	0	0	1	1	1	-4
0	0	1	0	2	2	-3
0	0	1	1	3	3	-2
0	1	0	0	4	4	-1
0	1	0	1	5	5	0
0	1	1	0	6	6	1
0	1	1	1	7	7	2
1	0	0	0	8	-8	3
1	0	0	1	9	-7	4
1	0	1	0	10	-6	5
1	0	1	1	11	-5	6
1	1	0	0	12	-4	7
1	1	0	1	13	-3	8
1	1	1	0	14	-2	9
1	1	1	1	15	-1	10

When we need more +ve values and few –ve values

0 0 0 0	0	0	-5
0 0 0 1	1	1	-4
0 0 1 0	2	2	-3
0 0 1 1	3	3	-2
0 1 0 0	4	4	-1
0 1 0 1	5	5	0
0 1 1 0	6	6	1
0 1 1 1	7	7	2
1 0 0 0	8	-8	3
1 0 0 1	9	-7	4
1 0 1 0	10	-6	5
1 0 1 1	11	-5	6
1 1 0 0	12	-4	7
1 1 0 1	13	-3	8
1 1 1 0	14	-2	9
1 1 1 1	15	-1	10

0	0	0	0	0	0	-5
0	0	0	1	1	1	-4
0	0	1	0	2	2	-3
0	0	1	1	3	3	-2
0	1	0	0	4	4	-1
0	1	0	1	5	5	0
0	1	1	0	6	6	1
0	1	1	1	7	7	2
1	0	0	0	8	-8	3
1	0	0	1	9	-7	4
1	0	1	0	10	-6	5
1	0	1	1	11	-5	6
1	1	0	0	12	-4	7
1	1	0	1	13	-3	8
1	1	1	0	14	-2	9
1	1	1	1	15	-1	10

0	0	0	0	0	0	-5
0	0	0	1	1	1	-4
0	0	1	0	2	2	-3
0	0	1	1	3	3	-2
0	1	0	0	4	4	-1
0	1	0	1	5	5	0
0	1	1	0	6	6	1
0	1	1	1	7	7	2
1	0	0	0	8	-8	3
1	0	0	1	9	-7	4
1	0	1	0	10	-6	5
1	0	1	1	11	-5	6
1	1	0	0	12	-4	7
1	1	0	1	13	-3	8
1	1	1	0	14	-2	9
1	1	1	1	15	-1	10

Need to add 5 to convert 3 into unsigned binary integer

0	0	0	0	0	0	-5
0	0	0	1	1	1	-4
0	0	1	0	2	2	-3
0	0	1	1	3	3	-2
0	1	0	0	4	4	-1
0	1	0	1	5	5	0
0	1	1	0	6	6	1
0	1	1	1	7	7	2
1	0	0	0	8	-8	3
1	0	0	1	9	-7	4
1	0	1	0	10	-6	5
1	0	1	1	11	-5	6
1	1	0	0	12	-4	7
1	1	0	1	13	-3	8
1	1	1	0	14	-2	9
1	1	1	1	15	-1	10

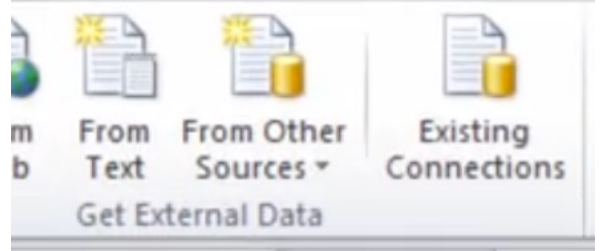
0 0 0 0	0	0	-5	-9
0 0 0 1	1	1	-4	-8
0 0 1 0	2	2	-3	-7
0 0 1 1	3	3	-2	-6
0 1 0 0	4	4	-1	-5
0 1 0 1	5	5	0	-4
0 1 1 0	6	6	1	-3
0 1 1 1	7	7	2	-2
1 0 0 0	8	-8	3	-1
1 0 0 1	9	-7	4	0
1 0 1 0	10	-6	5	1
1 0 1 1	11	-5	6	2
1 1 0 0	12	-4	7	3
1 1 0 1	13	-3	8	4
1 1 1 0	14	-2	9	5
1 1 1 1	15	-1	10	6

0	0	0	0	0	0	-5	-9	-8	-7
0	0	0	1	1	1	-4	-8	-7	-6
0	0	1	0	2	2	-3	-7	-6	-5
0	0	1	1	3	3	-2	-6	-5	-4
0	1	0	0	4	4	-1	-5	-4	-3
0	1	0	1	5	5	0	-4	-3	-2
0	1	1	0	6	6	1	-3	-2	-1
0	1	1	1	7	7	2	-2	-1	0
1	0	0	0	8	-8	3	-1	0	1
1	0	0	1	9	-7	4	0	1	2
1	0	1	0	10	-6	5	1	2	3
1	0	1	1	11	-5	6	2	3	4
1	1	0	0	12	-4	7	3	4	5
1	1	0	1	13	-3	8	4	5	6
1	1	1	0	14	-2	9	5	6	7
1	1	1	1	15	-1	10	6	7	8

$2^{(4-1)} - 1 = 7$ negative values

Offset of bais

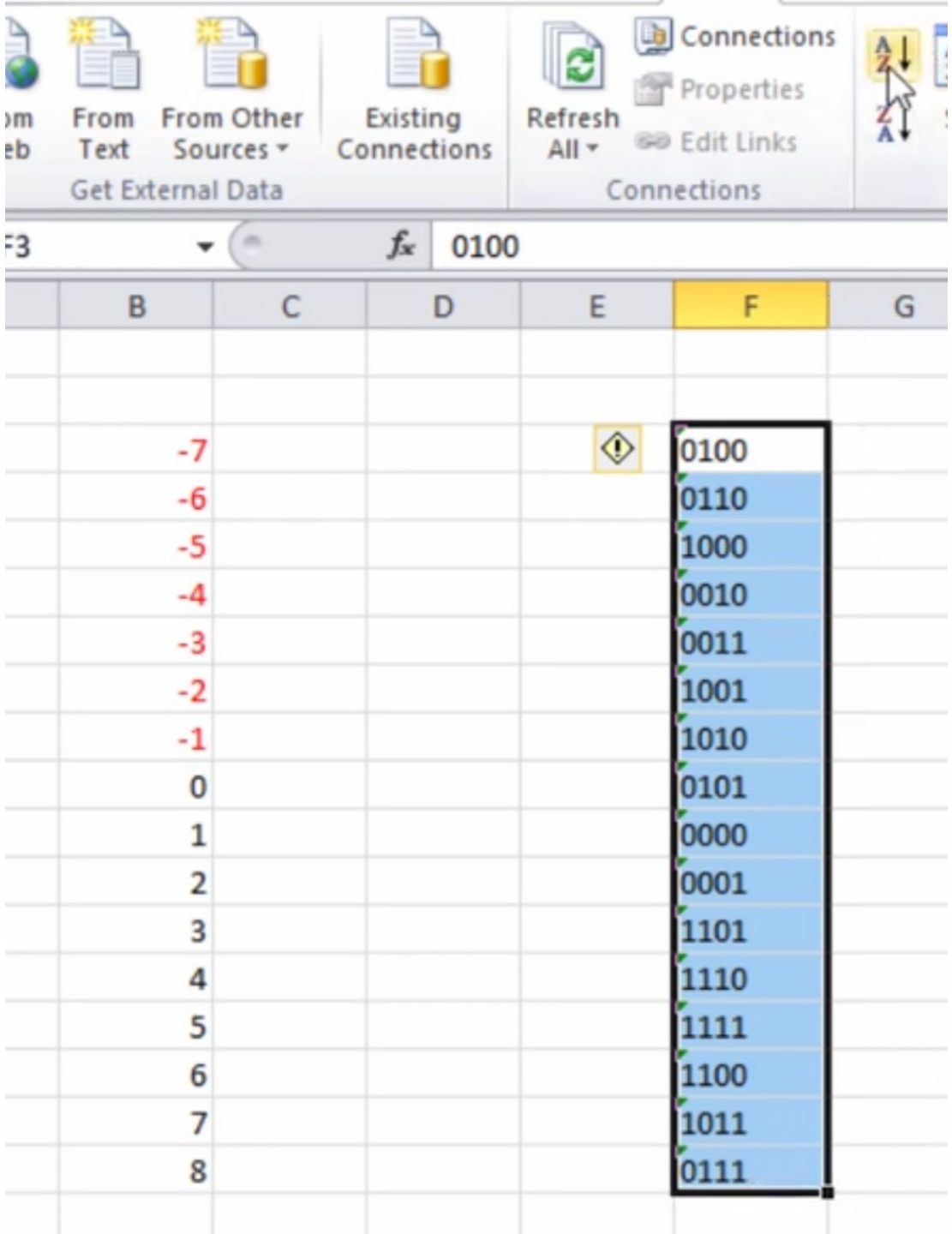
IEEE 754 Format	Sign	Exponent	Mantissa	Exponent Bias
32 bit single precision	1 bit	8 bits	23 bits (+ 1 not stored)	$2^{(8-1)} - 1 = 127$
64 bit double precision	1 bit	11 bits	52 bits (+ 1 not stored)	$2^{(11-1)} - 1 = 1023$
128 bit quadruple precision	1 bit	15 bits	112 bits (+ 1 not stored)	$2^{(15-1)} - 1 = 16383$

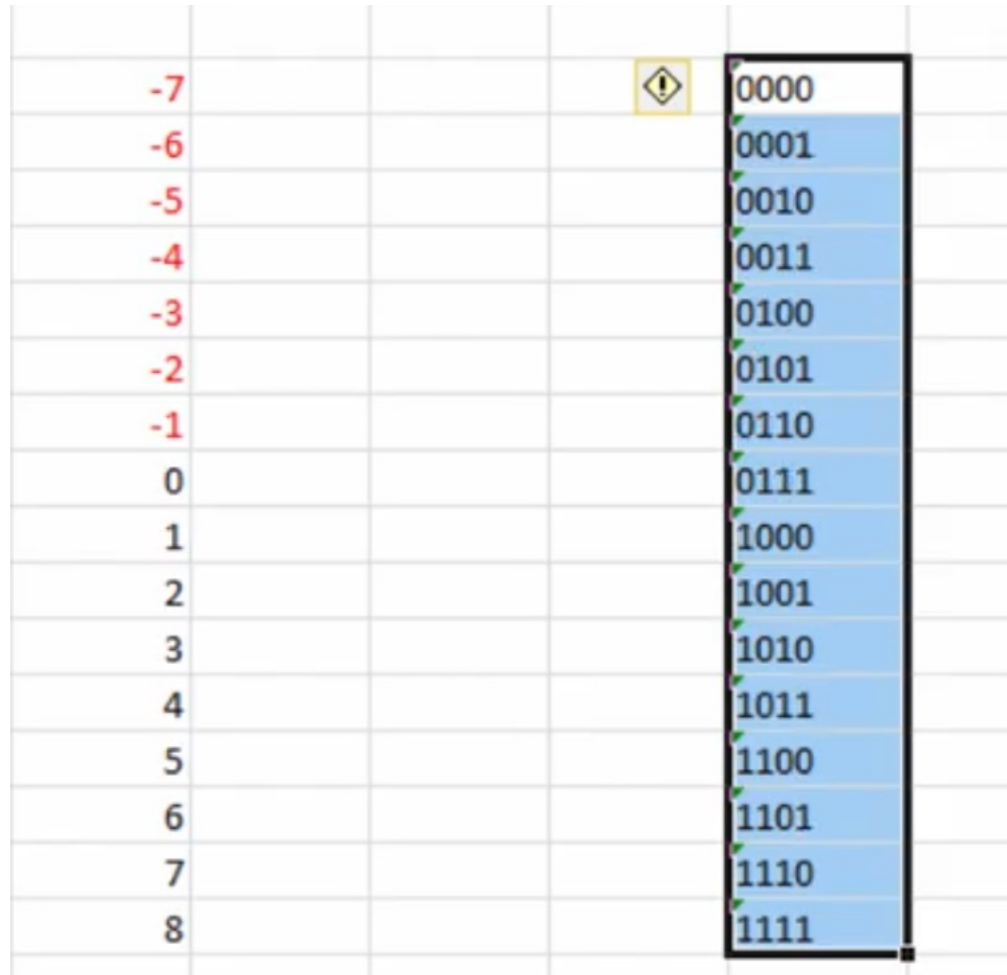


3

	B	C	D
	-7		0000
	-6		0001
	-5		0010
	-4		0011
	-3		0100
	-2		0101
	-1		0110
	0		0111
	1		1000
	2		1001
	3		1010
	4		1011
	5		1100
	6		1101
	7		1110
	8		1111

-7				0100
-6				0110
-5				1000
-4				0010
-3				0011
-2				1001
-1				1010
0				0101
1				0000
2				0001
3				1101
4				1110
5				1111
6				1100
7				1011
8				0111


From Text
From Other Sources
Existing Connections
Get External Data
Refresh All
Connections
Properties
Edit Links
Connections
F3
0100
B C D E F G
-7 0100
-6 0110
-5 1000
-4 0010
-3 0011
-2 1001
-1 1010
0 0101
1 0000
2 0001
3 1101
4 1110
5 1111
6 1100
7 1011
8 0111


-7
-6
-5
-4
-3
-2
-1
0
1
2
3
4
5
6
7
8
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111

Exercise 19

Convert 0.09375 into IEEE 754 single precision floating point binary

Convert -123.3 into IEEE 754 single precision floating point binary

Convert 0.09375 into IEEE 754 single precision floating point binary

Step 1: Determine the sign bit

Sign bit = 0

Step 2: Convert to pure binary

$0.09375_{10} = 0.00011_2$

0	÷	2	=	0	remainder	0	

0.09375	x	2	=	0.1875	0
0.1875	x	2	=	0.375	0
0.375	x	2	=	0.75	0
0.75	x	2	=	1.5	1
0.5	x	2	=	1.0	1

Step 3: Normalise for mantissa and unbiased exponent

$0.00011 = 1.1 \times 2^{-4}$

Step 4: Determine biased exponent

$-4 + 127 = 123_{10} = 01111011_2$

Step 5: Remove leading 1 from mantissa

1.1 = 1

0 01111011 10000000000000000000000

Convert -123.3 into IEEE 754 single precision floating point binary

Step 1: Determine the sign bit

Sign bit = 1

Step 2: Convert to pure binary

$123.3_{10} = 1111011.0100110011001\overline{1001}\ldots_2$

123	÷	2	=	61	remainder	1	
61	÷	2	=	30	remainder	1	
30	÷	2	=	15	remainder	0	
15	÷	2	=	7	remainder	1	
7	÷	2	=	3	remainder	1	
3	÷	2	=	1	remainder	1	
1	÷	2	=	0	remainder	1	

0.3	x	2	=	0.6	0
0.6	x	2	=	1.2	1
0.2	x	2	=	0.4	0
0.4	x	2	=	0.8	0
0.8	x	2	=	1.6	1
0.6	x	2	=	1.2	1
0.2	x	2	=	0.4	0

Step 3: Normalise for mantissa and unbiased exponent

$1111011.0100110011\overline{1001}\ldots = 1.1110110100110011\overline{1001}\ldots \times 2^6$

Step 4: Determine biased exponent

$6 + 127 = 133_{10} = 10000101_2$

Step 5: Remove leading 1 from mantissa

$1.1110110100110011\overline{1001}\ldots = 11101101001100110011001$

Step 6: Round mantissa up or down if necessary

$$\begin{array}{r} 11101101001100110011001 \\ +\ 1 \\ \hline 11101101001100110011010 \end{array}$$

1 10000101 11101101001100110011010

Exercise 20

Convert 01000010011010100000000000000000 into denary

Step 1: Determine the sign in denary

Convert 10000100100011001000000000000000 into denary

Convert 01000010011010100000000000000000 into denary

Step 1: Determine the sign in denary

Sign = + (positive)

Step 2: Determine the exponent in denary

$10000100_2 = 132_{10}$

Step 3: Remove the exponent bias

132 - 127 = 5

Step 4: Convert the mantissa to denary

0.5	0.25	0.125	0.0625	0.03125	0.015625
1	1	0	1	0	1

$0.5 + 0.25 + 0.0625 + 0.015625 = 0.828125_{10}$

Step 5: Add 1 to the mantissa and include the sign

1. 828125

Step 6: Calculate final result

$1.828125 \times 2^5 = 58.5$

58.5

Convert 10000100100011001000000000000000 into denary

Step 1: Determine the sign in denary

Sign = - (negative)

Step 2: Determine the exponent in denary

$00001001_2 = 9_{10}$

Step 3: Remove the exponent bias

9 - 127 = -118

Step 4: Convert the mantissa to denary

0.5	0.25	0.125	0.0625	0.03125	0.015625	0.0078125	0.00390625
0	0	0	1	1	0	0	1

$0.0625 + 0.03125 + 0.00390625 = 0.09765625_{10}$

Step 5: Add 1 to the mantissa and include the sign

-1.09765625

Step 6: Calculate final result

$-1.09765625 \times 2^{-118} = -3.303139125810627897392149669594 5 \times 10^{-36}$

-0.00000000000000000000000000000003303

Reserved Exponent Values

Exponent Value	Mantissa	Represents
11111111	All zeros	Infinity (∞)
11111111	Not all zeros	Not a number (NAN)
00000000	All zeros	Zero
00000000	Not all zeros	Subnormal (very small)

NaN

From Wikipedia, the free encyclopedia

For other uses, see Nan (disambiguation).

In computing, **NaN** (/næn/), standing for **Not a Number**, is a member of a numeric data type that can be interpreted as a value that is undefined or unrepresentable, especially in floating-point arithmetic. Systematic use of NaNs was introduced by the IEEE 754 floating-point standard in 1985, along with the representation of other non-finite quantities such as infinities.

In mathematics, zero divided by zero is undefined as a real number, and is therefore represented by NaN in computing systems. The square root of a negative number is not a real number, and is therefore also represented by NaN in compliant computing systems. NaNs may also be used to represent missing values in computations.[1][2]

Two separate kinds of NaNs are provided, termed *quiet NaNs* and *signaling NaNs*. Quiet NaNs are used to propagate errors resulting from invalid operations or values. Signaling NaNs can support advanced features such as mixing numerical and symbolic computation or other extensions to basic floating-point arithmetic.

Subnormal Numbers

https://stackoverflow.com/questions/8341395/what-is-a-subnormal-floating-point-number

Unicode Transformation Formats and Byte Order

Big Endian

Little Endian

Big Endian

Little Endian

Big Endian


UNICODE

Code Point	Denary Value	Character
U+0045	69	E
U+03A6	934	Φ
U+86CB	34507	蛋
U+1F95A	129370	

American Standard Code of Information Interchange

ASCII

Code Point	Denary Value	Character	ASCII
U+0045	69	E	1000101
U+03A6	934	Φ	✗
U+86CB	34507	蛋	✗
U+1F95A	129370		✗

American Standard Code of Information Interchange

ASCII

Code Point	Denary Value	Character	ASCII
U+0045	69	E	1000101
U+03A6	934	Φ	✗
U+86CB	34507	蛋	✗
U+1F95A	129370	🥚	✗

$2^7 = 128$ characters

Universal Character Set

UCS-2

Code Point	Denary Value	Character	UCS-2 encoding
U+0045	69	E	00000000 01000101
U+03A6	934	Φ	00000011 10100110
U+86CB	34507	蛋	10000110 11001011
U+1F95A	129370	🥚	✗

2^{16} = 65536 characters

U+0000 to U+FFFF

Basic Multilingual Plane

Unicode Transformation Format

UTF-16

Year - 1996

Code Point	Denary Value	Character	UCS-2 encoding	UTF-16 encoding
U+0045	69	E	00000000 01000101	00000000 01000101
U+03A6	934	Φ	00000011 10100110	00000011 10100110
U+86CB	34507	蛋	10000110 11001011	10000110 11001011
U+1F95A	129370	🥚	✗	1101100000111110 1101110101011010

Variable length encoding which can use 16 or 32 bits to address the limitations of USC-2 encoding.

Unicode Transformation Format

UTF-16

Code Point	Denary Value	Character	UCS-2 encoding	UTF-16 encoding
U+0045	69	E	00000000 01000101	00000000 01000101
U+03A6	934	Φ	00000011 10100110	00000011 10100110
U+86CB	34507	蛋	10000110 11001011	10000110 11001011
U+1F95A	129370		✗	1101100000111110 1101110101011010

High Surrogate Low Surrogate

110110XXXXXXXXXX 110111XXXXXXXXXX

$1F95A_{16} - 10000_{16} = F95A_{16}$

$F95A_{16} = 0000111110010101101_0$

Unicode Transformation Format

UTF-16

Code Point	Denary Value	Character	UCS-2 encoding	UTF-16 encoding	UTF-16 (Hex)
U+0045	69	E	00000000 01000101	00000000 01000101	00 45
U+03A6	934	Φ	00000011 10100110	00000011 10100110	03 A6
U+86CB	34507	蛋	10000110 11001011	10000110 11001011	86 CB
U+1F95A	129370	🥚	✗	1101100000111110 1101110101011010	D8 3E DD 5A

High Surrogate Low Surrogate

1101100000111110 1101110101011010

$1F95A_{16} - 10000_{16} = F95A_{16}$

$F95A_{16} = 0000111110010101101{0}_{2}$

Why we need these control bits?

Unicode Transformation Format

UTF-16

Code Point	Denary Value	Character	UCS-2 encoding	UTF-16 encoding	UTF-16 (Hex)
U+0045	69	E	00000000 01000101	00000000 01000101	00 45
U+03A6	934	Φ	00000011 10100110	00000011 10100110	03 A6
U+86CB	34507	蛋	10000110 11001011	10000110 11001011	86 CB
U+1F95A	129370	🥚	✗	1101100000111110 1101110101011010	D8 3E DD 5A

High Surrogate	Low Surrogate
110110XXXXXXXXXX	110111XXXXXXXXXX

Unicode Transformation Format

UTF-16

Code Point	Denary Value	Character	UCS-2 encoding	UTF-16 encoding	UTF-16 (Hex)
U+0045	69	E	00000000 01000101	00000000 01000101	00 45
U+03A6	934	Φ	00000011 10100110	00000011 10100110	03 A6
U+86CB	34507	蛋	10000110 11001011	10000110 11001011	86 CB
U+1F95A	129370	🥚	✗	1101100000111110 1101110101011010	D8 3E DD 5A

High Surrogate 110110XXXXXXXXXX

Low Surrogate 110111XXXXXXXXXX

Unicode Transformation Format

UTF-16

Code Point	Denary Value	Character	UCS-2 encoding	UTF-16 encoding	UTF-16 (Hex)
U+0045	69	E	00000000 01000101	00000000 01000101	00 45
U+03A6	934	Φ	00000011 10100110	00000011 10100110	03 A6
U+86CB	34507	蛋	10000110 11001011	10000110 11001011	86 CB
U+1F95A	129370	🥚	✗	1101100000111110 1101110101011010	D8 3E DD 5A

High Surrogate 110110XXXXXXXXXX

Low Surrogate 110111XXXXXXXXXX

Unicode Transformation Format

UTF-16

Code Point	Denary Value	Character	UCS-2 encoding	UTF-16 encoding	UTF-16 (Hex)
U+0045	69	E	00000000 01000101	00000000 01000101	00 45
U+03A6	934	Φ	00000011 10100110	00000011 10100110	03 A6
U+86CB	34507	蛋	10000110 11001011	10000110 11001011	86 CB
U+1F95A	129370	🥚	✗	1101100000111110 1101110101011010	D8 3E DD 5A

High Surrogate 110110XXXXXXXXXX

Low Surrogate 110111XXXXXXXXXX

Unicode Transformation Format

UTF-16

Code Point	Denary Value	Character	UCS-2 encoding	UTF-16 encoding	UTF-16 (Hex)
U+0045	69	E	00000000 01000101	00000000 01000101	00 45
U+03A6	934	Φ	00000011 10100110	00000011 10100110	03 A6
U+86CB	34507	蛋	10000110 11001011	10000110 11001011	86 CB
U+1F95A	129370		✗	1101100000111110 1101110101011010	D8 3E DD 5A

High Surrogate 110110XXXXXXXXXX

Low Surrogate 110111XXXXXXXXXX

High surrogate range 1101100000000000 to 1101101111111111, D800 to DBFF, unavailable for 16 bit encodings

Unicode Transformation Format

UTF-16

Code Point	Denary Value	Character	UCS-2 encoding	UTF-16 encoding	UTF-16 (Hex)
U+0045	69	E	00000000 01000101	00000000 01000101	00 45
U+03A6	934	Φ	00000011 10100110	00000011 10100110	03 A6
U+86CB	34507	蛋	10000110 11001011	10000110 11001011	86 CB
U+1F95A	129370	🥚	✗	1101100000111110 1101110101011010	D8 3E DD 5A

High Surrogate Low Surrogate

110110XXXXXXXXXX 110111XXXXXXXXXX

High surrogate range 1101100000000000 to 1101101111111111, D800 to DBFF, unavailable for 16 bit encodings

Low surrogate range 1101110000000000 to 1101111111111111, DC00 to DFFF, unavailable for 16 bit encodings

Unicode Transformation Format

UTF-16

Code Point	Denary Value	Character	UCS-2 encoding	UTF-16 encoding	UTF-16 (Hex)
U+0045	69	E	00000000 01000101	00000000 01000101	00 45
U+03A6	934	Φ	00000011 10100110	00000011 10100110	03 A6
U+86CB	34507	蛋	10000110 11001011	10000110 11001011	86 CB
U+10000	65536	𐀀	✗	1101100000000000 1101110000000000	D8 00 DC 00
U+1F95A	129370	🥚	✗	1101100000111110 1101110101011010	D8 3E DD 5A

High Surrogate Low Surrogate

1101100000000000 1101110000000000

$10000_{16} - 10000_{16} = 00_{16}$

$00_{16} = 00000000000000000000_{2}$

Unicode Transformation Format

UTF-32

Code Point	Denary Value	Character	UTF-32 encoding	UTF-32 (Hex)
U+0045	69	E	00000000 00000000 00000000 01000101	00 00 00 45
U+03A6	934	Φ	00000000 00000000 00000011 10100110	00 00 03 A6
U+86CB	34507	蛋	00000000 00000000 10000110 11001011	00 00 86 CB
U+10000	65536	𐀀	00000000 00000001 00000000 00000000	00 01 00 00
U+1F95A	129370	🥚	00000000 00000001 11111001 01011010	00 01 F9 5A

Unicode Transformation Format

UTF-32

Code Point	Denary Value	Character	UTF-32 encoding	UTF-32 (Hex)
U+0045	69	E	00000000 00000000 00000000 01000101	00 00 00 45
U+03A6	934	Φ	00000000 00000000 00000011 10100110	00 00 03 A6
U+86CB	34507	蛋	00000000 00000000 10000110 11001011	00 00 86 CB
U+10000	65536	𐀀	00000000 00000001 00000000 00000000	00 01 00 00
U+1F95A	129370	🥚	00000000 00000001 11111001 01011010	00 01 F9 5A

Used in unix based OSs

$1F95A_{16}$ = 00000000 00000001 11111001 01011010$_2$

0045_{16} = **00000000 00000000 00000000** 01000101$_2$

$86CB_{16}$ = **00000000 00000000** 10000110 11001011$_2$

Text files in UTF-32 are need very large in size, however, very easy to process as compared to variable length encoding.

Unicode Transformation Format

UTF-8

Code Point	Denary Value	Character	UTF-8 encoding	UTF-8 (Hex)
U+0045	69	E	01000101	45
U+03A6	934	Φ	11001110 10100110	CE A6
U+86CB	34507	蛋	11101000 10011011 10001011	E8 9B 8B
U+10000	65536	𐀀	11110000 10010000 10000000 10000000	F0 90 80 80
U+1F95A	129370	🥚	11110000 10011111 10100101 10011010	F0 9F A5 9A

Most domient encoding scheme

Unicode Transformation Format

UTF-8

Code Point	Denary Value	Character	UTF-8 encoding	UTF-8 (Hex)
U+0045	69	E	01000101	45
U+03A6	934	Φ	11001110 10100110	CE A6
U+86CB	34507	蛋	11101000 10011011 10001011	E8 9B 8B
U+10000	65536	𐀀	11110000 10010000 10000000 10000000	F0 90 80 80
U+1F95A	129370	🥚	11110000 10011111 10100101 10011010	F0 9F A5 9A

110XXXXX 10XXXXXX

Unicode Transformation Format

UTF-8

Code Point	Denary Value	Character	UTF-8 encoding	UTF-8 (Hex)
U+0045	69	E	01000101	45
U+03A6	934	Φ	11001110 10100110	CE A6
U+86CB	34507	蛋	11101000 10011011 10001011	E8 9B 8B
U+10000	65536	𐀀	11110000 10010000 10000000 10000000	F0 90 80 80
U+1F95A	129370	🥚	11110000 10011111 10100101 10011010	F0 9F A5 9A

11001110 10100110

$03A6_{16} = 1110100110_2$

Unicode Transformation Format

UTF-8

Code Point	Denary Value	Character	UTF-8 encoding	UTF-8 (Hex)
U+0045	69	E	01000101	45
U+03A6	934	Φ	11001110 10100110	CE A6
U+86CB	34507	蛋	11101000 10011011 10001011	E8 9B 8B
U+10000	65536	𐀀	11110000 10010000 10000000 10000000	F0 90 80 80
U+1F95A	129370	🥚	11110000 10011111 10100101 10011010	F0 9F A5 9A

1110XXXX 10XXXXXX 10XXXXXX

$86CB_{16}$ = 1000011011001011_2

Unicode Transformation Format

UTF-8

Code Point	Denary Value	Character	UTF-8 encoding	UTF-8 (Hex)
U+0045	69	E	01000101	45
U+03A6	934	Φ	11001110 10100110	CE A6
U+86CB	34507	蛋	11101000 10011011 10001011	E8 9B 8B
U+10000	65536	𐀀	11110000 10010000 10000000 10000000	F0 90 80 80
U+1F95A	129370	🥚	11110000 10011111 10100101 10011010	F0 9F A5 9A

11101000 10011011 10001011

$86CB_{16} = 1000011011001011_2$

Unicode Transformation Format

UTF-8

Code Point	Denary Value	Character	UTF-8 encoding	UTF-8 (Hex)
U+0045	69	E	01000101	45
U+03A6	934	Φ	11001110 10100110	CE A6
U+86CB	34507	蛋	11101000 10011011 10001011	E8 9B 8B
U+10000	65536	𐀀	11110000 10010000 10000000 10000000	F0 90 80 80
U+1F95A	129370	🥚	11110000 10011111 10100101 10011010	F0 9F A5 9A

11110XXX 10XXXXXX 10XXXXXX 10XXXXXX

$1F95A_{16} = 11111100101011010_2$

Unicode Transformation Format

UTF-8

Code Point	Denary Value	Character	UTF-8 encoding	UTF-8 (Hex)
U+0045	69	E	01000101	45
U+03A6	934	Φ	11001110 10100110	CE A6
U+86CB	34507	蛋	11101000 10011011 10001011	E8 9B 8B
U+10000	65536	𐀀	11110000 10010000 10000000 10000000	F0 90 80 80
U+1F95A	129370	🥚	11110000 10011111 10100101 10011010	F0 9F A5 9A

11110000 10011111 10100101 10011010

$1F95A_{16} = 11111100101011010_2$

UTF-8, UTF-16 and UTF-32 Compared

Code Point	Character	UTF-8 (Hex)	UTF-16 (Hex)	UTF-32 (Hex)
U+0045	E	45	00 45	00 00 00 45
U+03A6	Φ	CE A6	03 A6	00 00 03 A6
U+86CB	蛋	E8 9B 8B	86 CB	00 00 86 CB
U+1F95A	🥚	F0 9F A5 9A	D8 3E DD 5A	00 01 F9 5A

UTF-8 is good for English characters whereas UTF-16 is for Asian languages

Byte Order

UTF-16 Byte Order

Code Point	Character	UTF-16 encoding	UTF-16 (Hex)
U+0045	E	00000000 01000101	00 45
U+0049	I	00000000 01000101	00 49
U+03A6	Φ	00000011 10100110	03 D5
U+2764	❤	00000011 11010101	27 64
U+65E5	日	10000110 11001011	65 E5
U+672C	本	01100111 00101100	67 2C
U+86CB	蛋	10000110 11001011	86 CB
U+1F63D	😽	11011000 00111101 11011110 00111101	D8 3D DE 3D
U+1F95A	🥚	11011000 00111110 11011101 01011010	D8 3E DD 5A

UTF-16 Byte Order

Code Point	Character	UTF-16 encoding	UTF-16 (Hex)	UTF-16 encoding	UTF-16 (Hex)
U+0045	E	00000000 01000101	00 45	01000101 00000000	45 00
U+0049	I	00000000 01000101	00 49	01000101 00000000	49 00
U+03A6	Φ	00000011 10100110	03 D5	10100110 00000011	A6 03
U+2764	❤	00000011 11010101	27 64	11010101 00000011	64 27
U+65E5	日	10000110 11001011	65 E5	11001011 10000110	E5 65
U+672C	本	01100111 00101100	67 2C	00101100 01100111	2C 67
U+86CB	蛋	10000110 11001011	86 CB	11001011 10000110	CB 86
U+1F63D	😽	11011000 00111101 11011110 00111101	D8 3D DE 3D	00111101 11011000 00111101 11011110	3D D8 3D DE
U+1F95A	🥚	11011000 00111110 11011101 01011010	D8 3E DD 5A	00111110 11011000 01011010 11011101	3E D8 5A DD

UTF-16 Byte Order

Big Endian

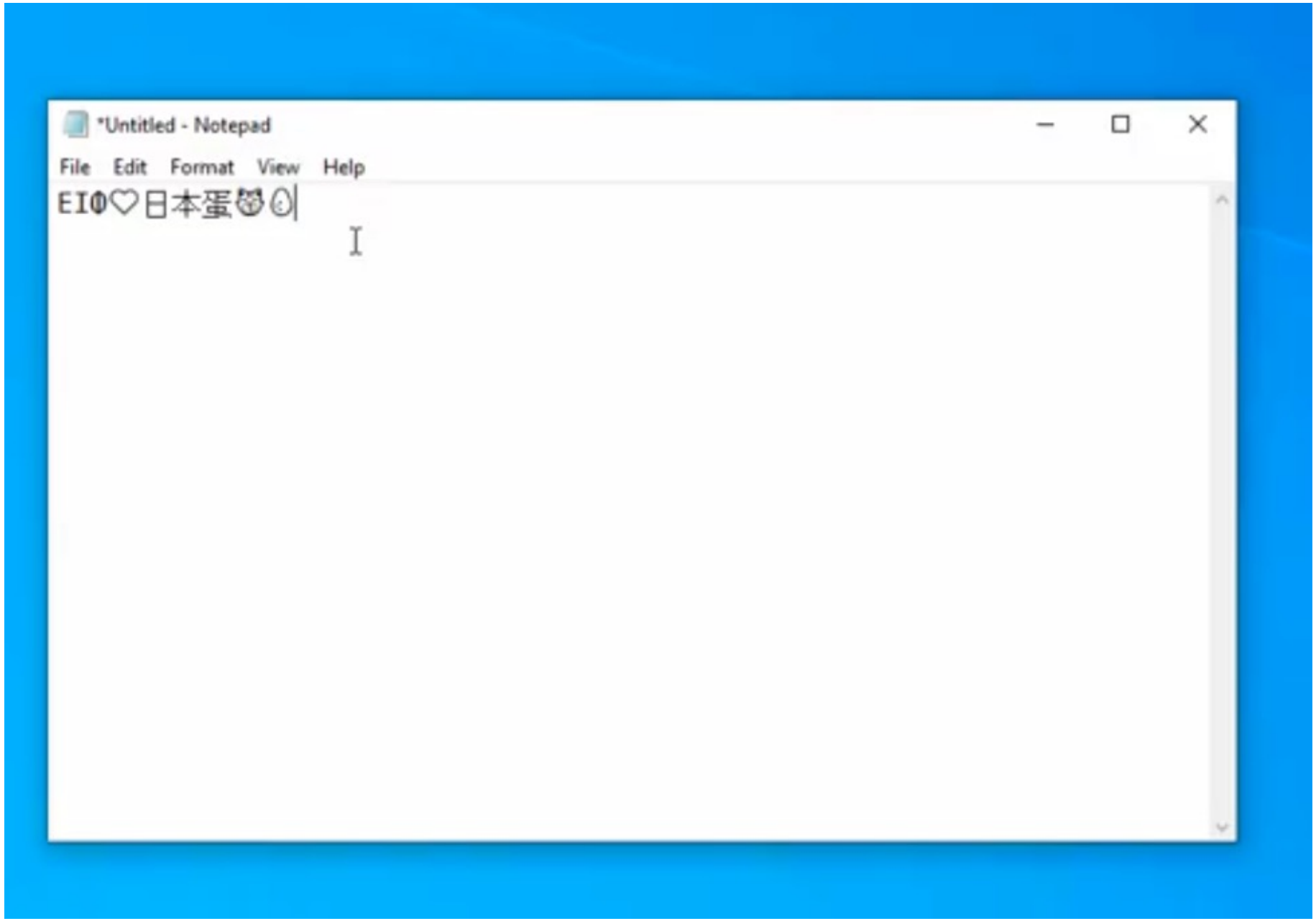
Code Point	Character	UTF-16 encoding	UTF-16 (Hex)	UTF-16 encoding	UTF-16 (Hex)
U+0045	E	00000000 01000101	00 45	01000101 00000000	45 00
U+0049	I	00000000 01000101	00 49	01000101 00000000	49 00
U+03A6	Φ	00000011 10100110	03 D5	10100110 00000011	A6 03
U+2764	❤	00000011 11010101	27 64	11010101 00000011	64 27
U+65E5	日	10000110 11001011	65 E5	11001011 10000110	E5 65
U+672C	本	01100111 00101100	67 2C	00101100 01100111	2C 67
U+86CB	蛋	10000110 11001011	86 CB	11001011 10000110	CB 86
U+1F63D	😽	11011000 00111101 11011110 00111101	D8 3D DE 3D	00111101 11011000 00111101 11011110	3D D8 3D DE
U+1F95A	🥚	11011000 00111110 11011101 01011010	D8 3E DD 5A	00111110 11011000 01011010 11011101	3E D8 5A DD

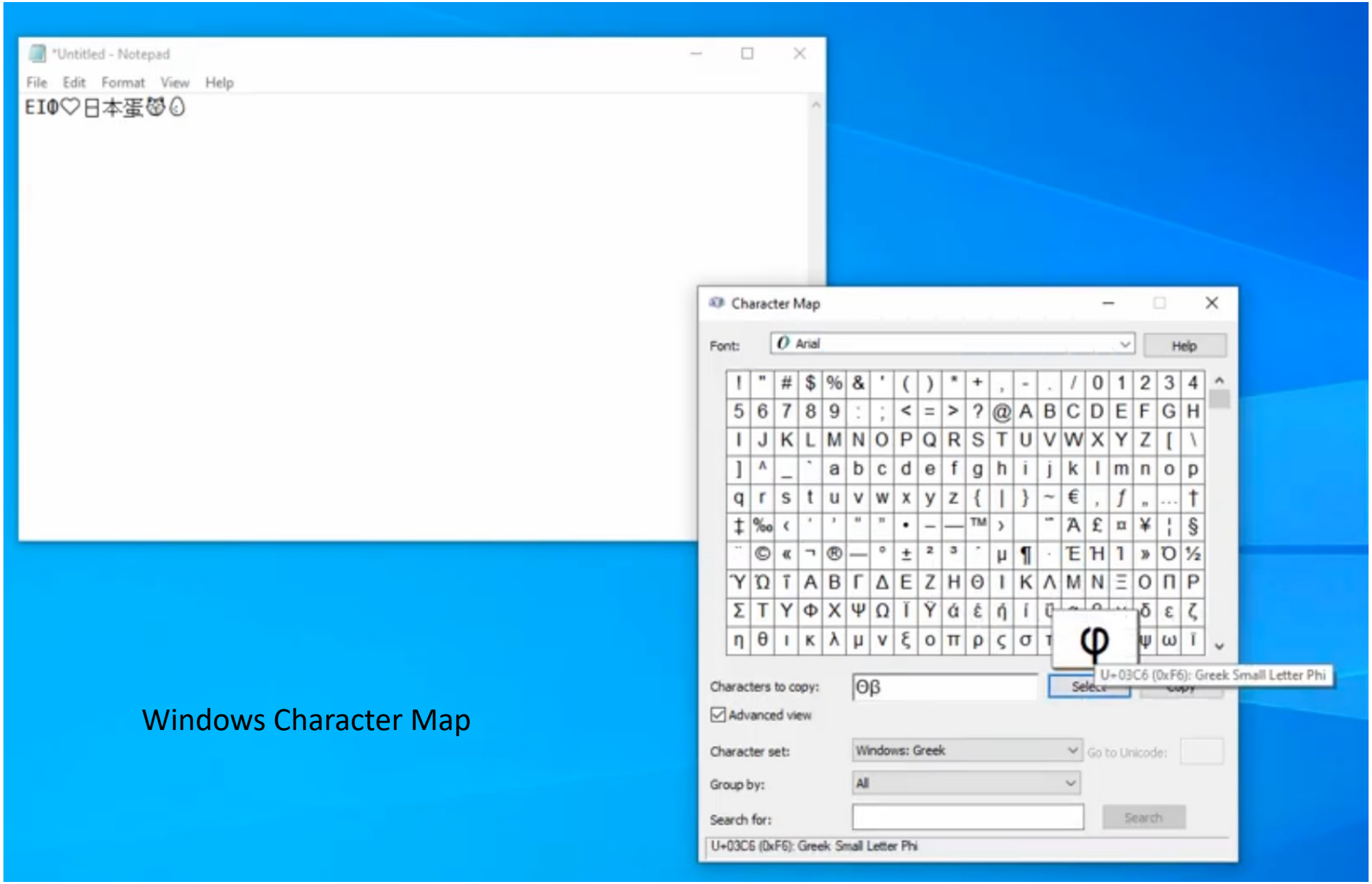
UTF-16 Byte Order

		Big Endian		Little Endian	
Code Point	**Character**	**UTF-16 encoding**	**UTF-16 (Hex)**	**UTF-16 encoding**	**UTF-16 (Hex)**
U+0045	E	00000000 01000101	00 45	01000101 00000000	45 00
U+0049	I	00000000 01000101	00 49	01000101 00000000	49 00
U+03A6	Φ	00000011 10100110	03 D5	10100110 00000011	A6 03
U+2764	❤	00000011 11010101	27 64	11010101 00000011	64 27
U+65E5	日	10000110 11001011	65 E5	11001011 10000110	E5 65
U+672C	本	01100111 00101100	67 2C	00101100 01100111	2C 67
U+86CB	蛋	10000110 11001011	86 CB	11001011 10000110	CB 86
U+1F63D	😽	11011000 00111101 11011110 00111101	D8 3D DE 3D	00111101 11011000 00111101 11011110	3D D8 3D DE
U+1F95A	🥚	11011000 00111110 11011101 01011010	D8 3E DD 5A	00111110 11011000 01011010 11011101	3E D8 5A DD

UTF-16 Byte Order

		Big Endian		Little Endian	
Code Point	**Character**	**UTF-16 encoding**	**UTF-16 (Hex)**	**UTF-16 encoding**	**UTF-16 (Hex)**
U+0045	E	00000000 01000101	00 45	01000101 00000000	45 00
U+0049	I	00000000 01000101	00 49	01000101 00000000	49 00
U+03A6	Φ	00000011 10100110	03 D5	10100110 00000011	A6 03
U+2764	❤	00000011 11010101	27 64	11010101 00000011	64 27
U+65E5	日	10000110 11001011	65 E5	11001011 10000110	E5 65
U+672C	本	01100111 00101100	67 2C	00101100 01100111	2C 67
U+86CB	蛋	10000110 11001011	86 CB	11001011 10000110	CB 86
U+1F63D	😽	11011000 00111101 11011110 00111101	D8 3D DE 3D	00111101 11011000 00111101 11011110	3D D8 3D DE
U+1F95A	🥚	11011000 00111110 11011101 01011010	D8 3E DD 5A	00111110 11011000 01011010 11011101	3E D8 5A DD

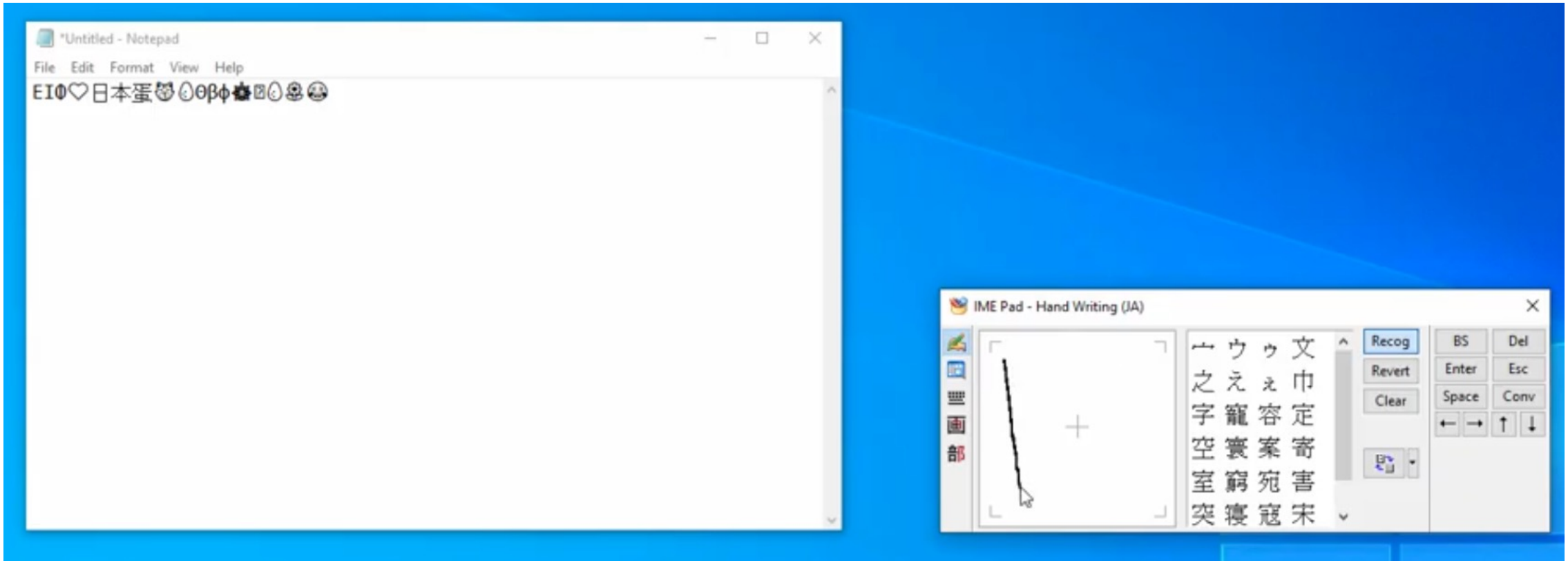

*Untitled - Notepad
File Edit Format View Help
EI🅾♡日本蛋🐱

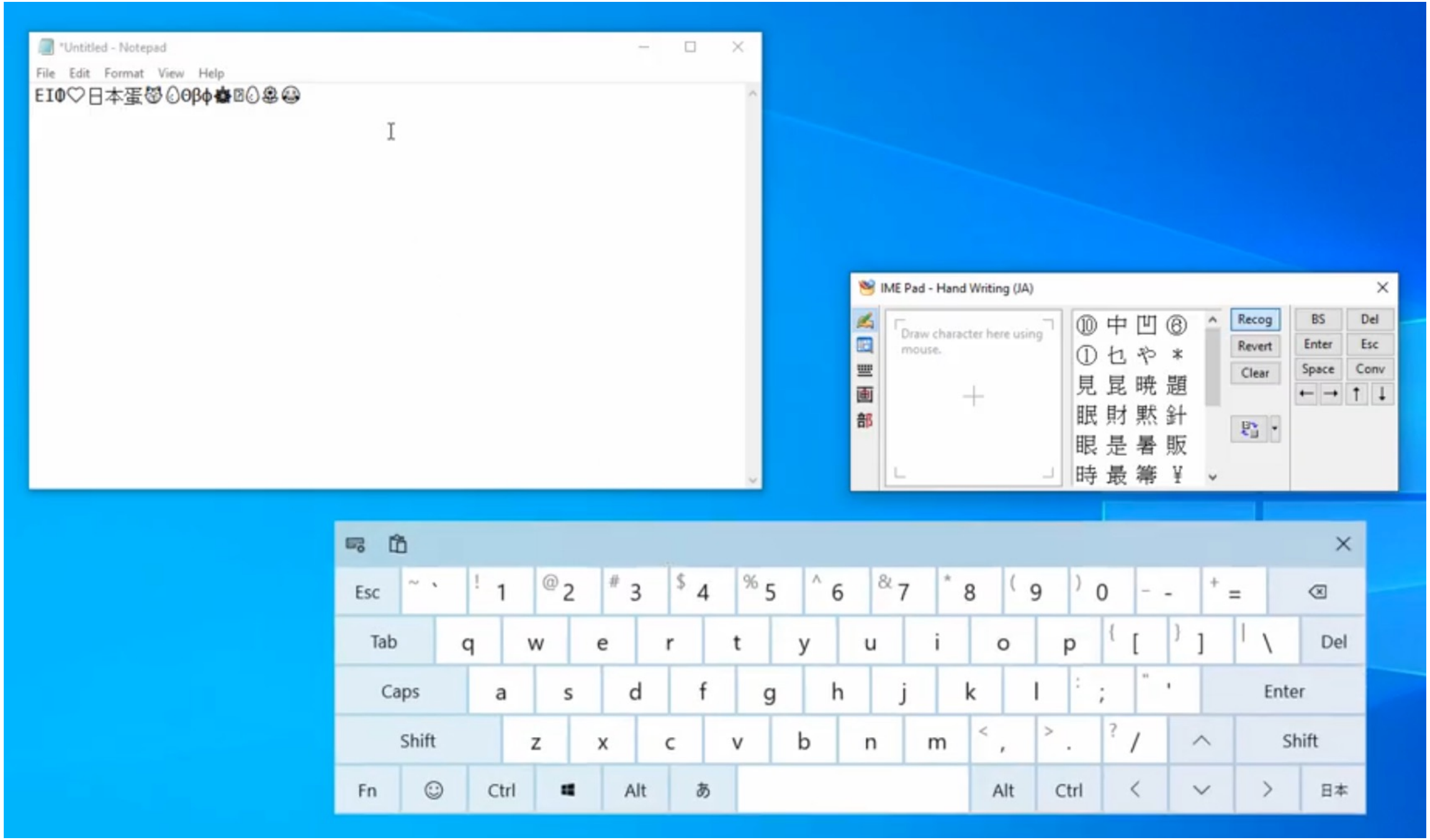


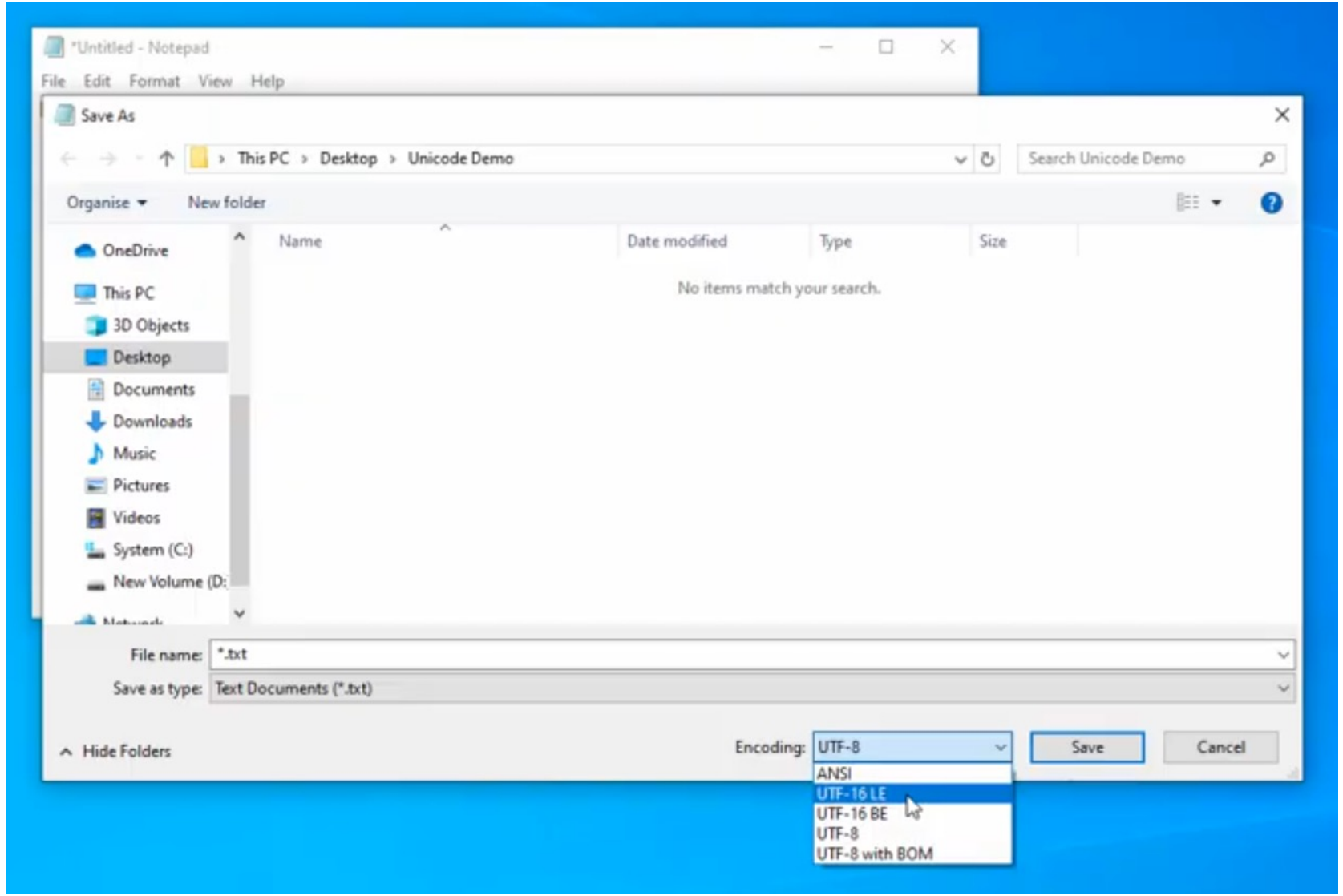
Windows Character Map

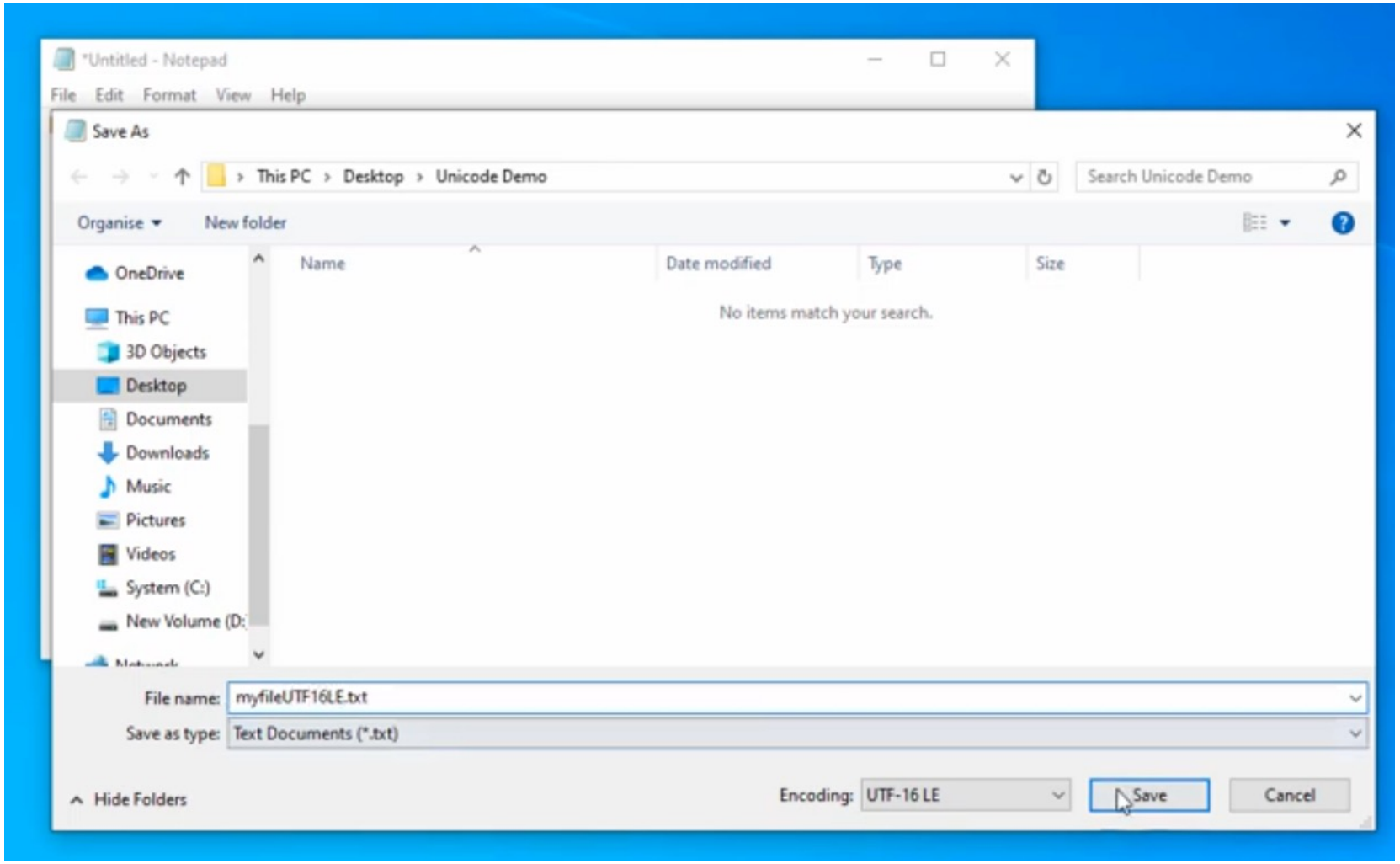

*Untitled - Notepad
File Edit Format View Help
Emoji - Keep typing to find an emoji
;-)
Ω

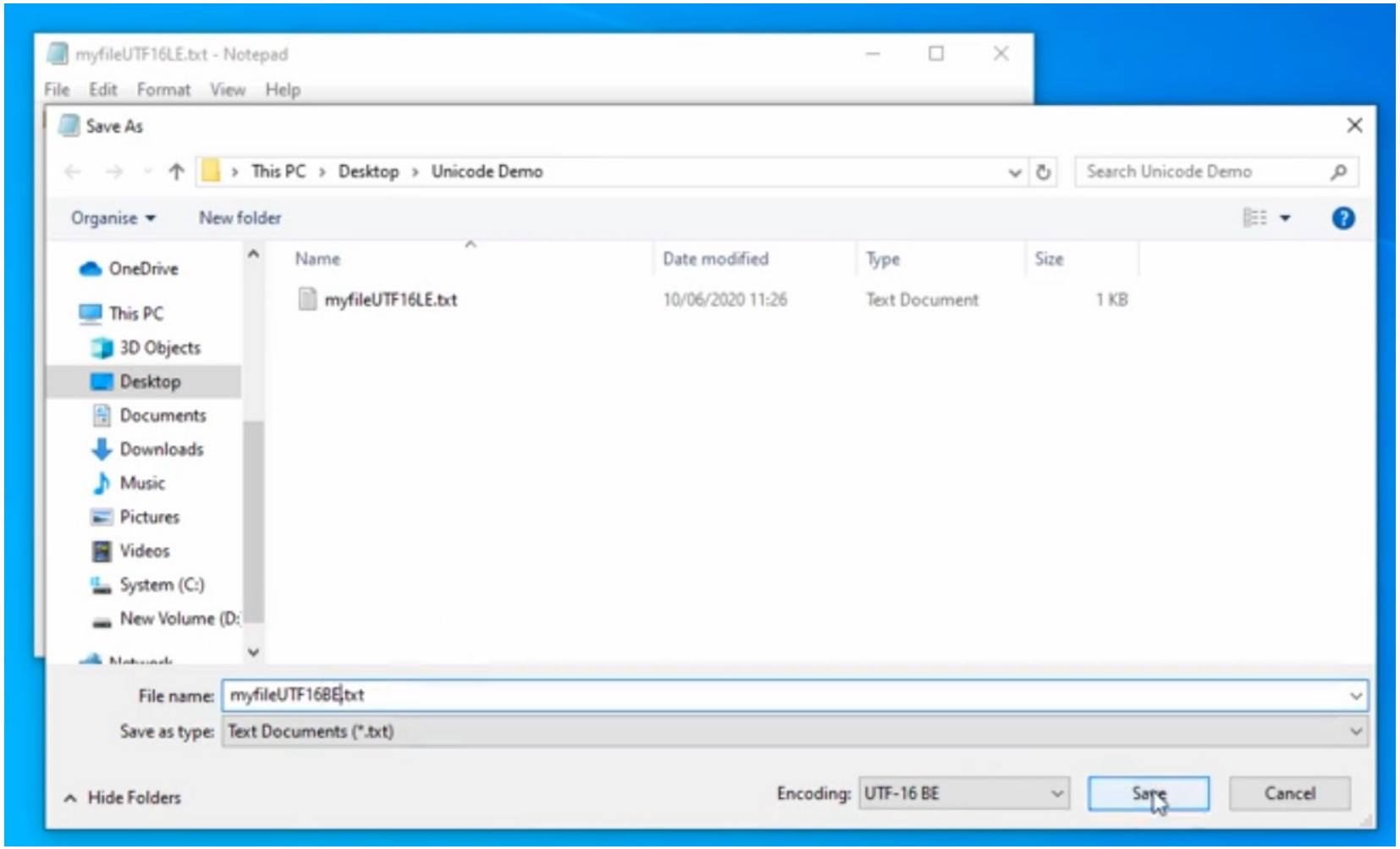
Windows Key + .

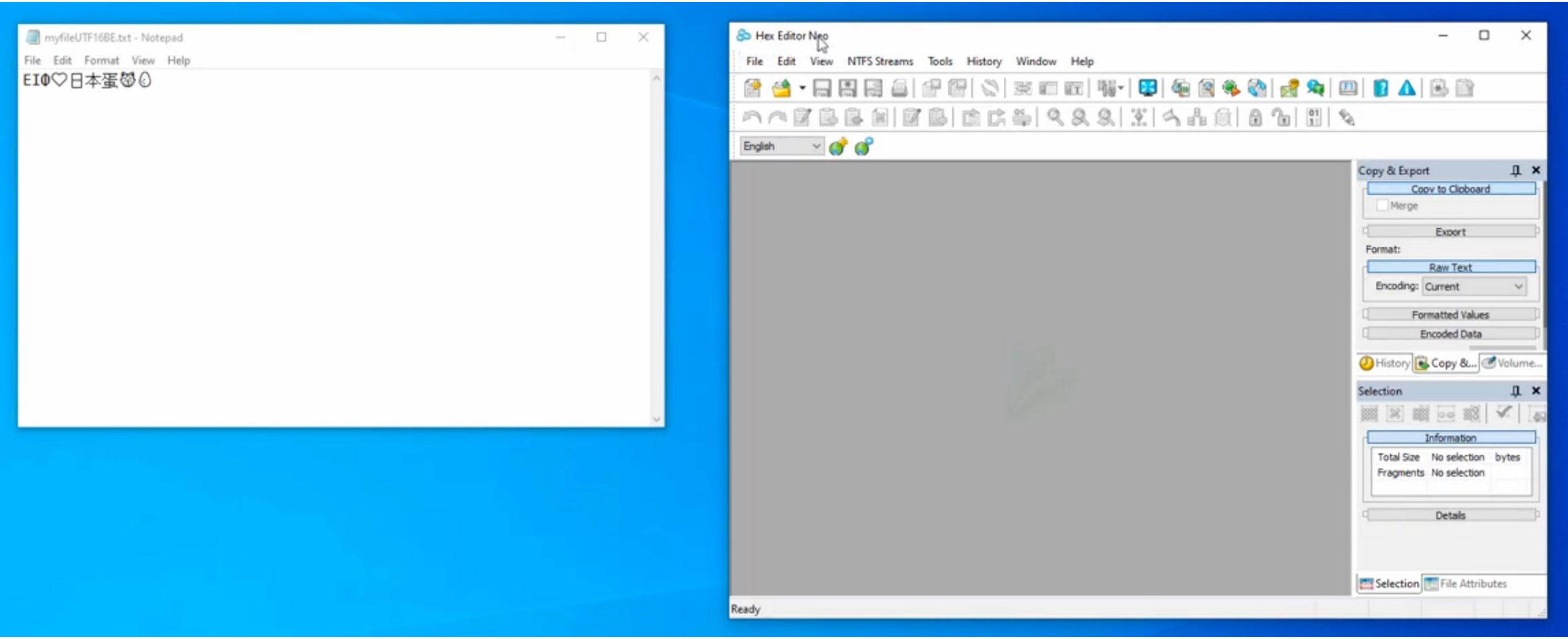

*Untitled - Notepad
File Edit Format View Help
EI♡日本蛋
IME Pad - Hand Writing (JA)
Recog
Revert
Clear
BS
Del
Enter
Esc
Space
Conv

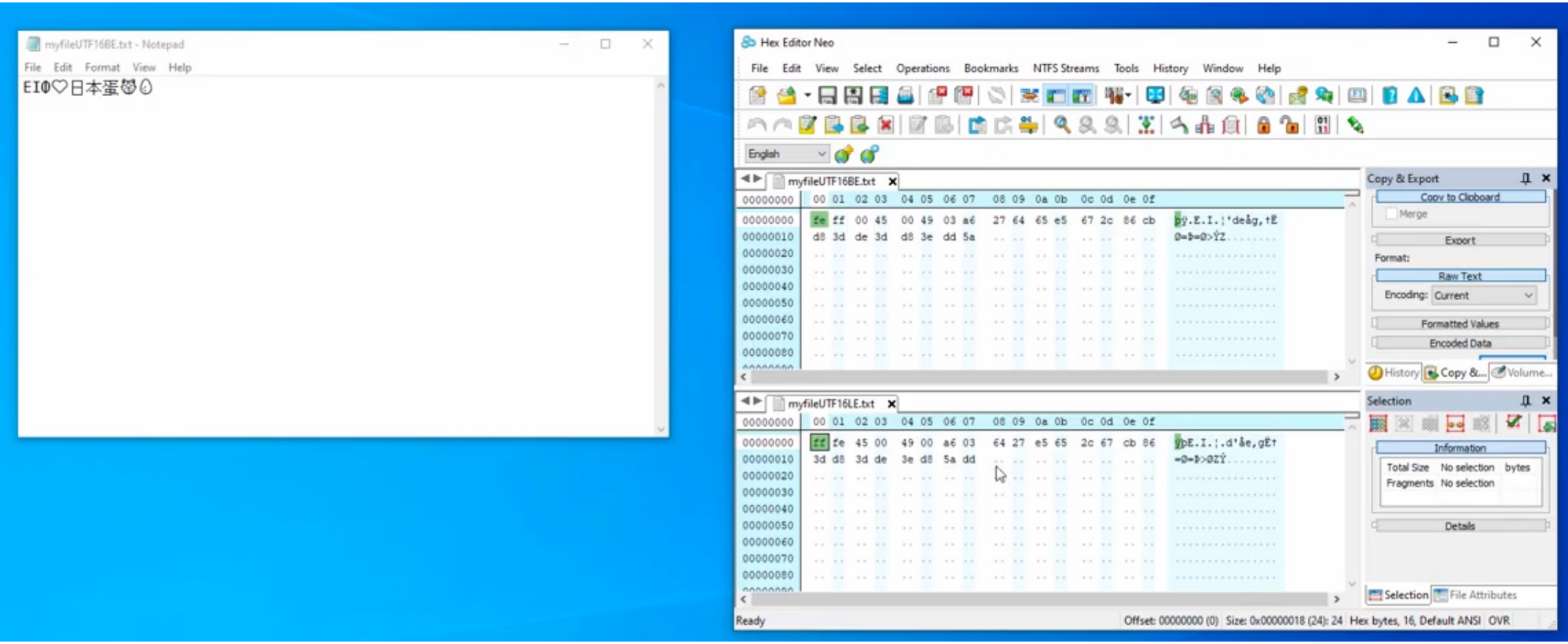

*Untitled - Notepad
File Edit Format View Help
IME Pad - Hand Writing (JA)
Draw character here using mouse.
Recog
Revert
Clear
BS
Del
Enter
Esc
Space
Conv
Esc Tab Caps Shift Fn Ctrl Alt あ Alt Ctrl Enter Shift Del 日本

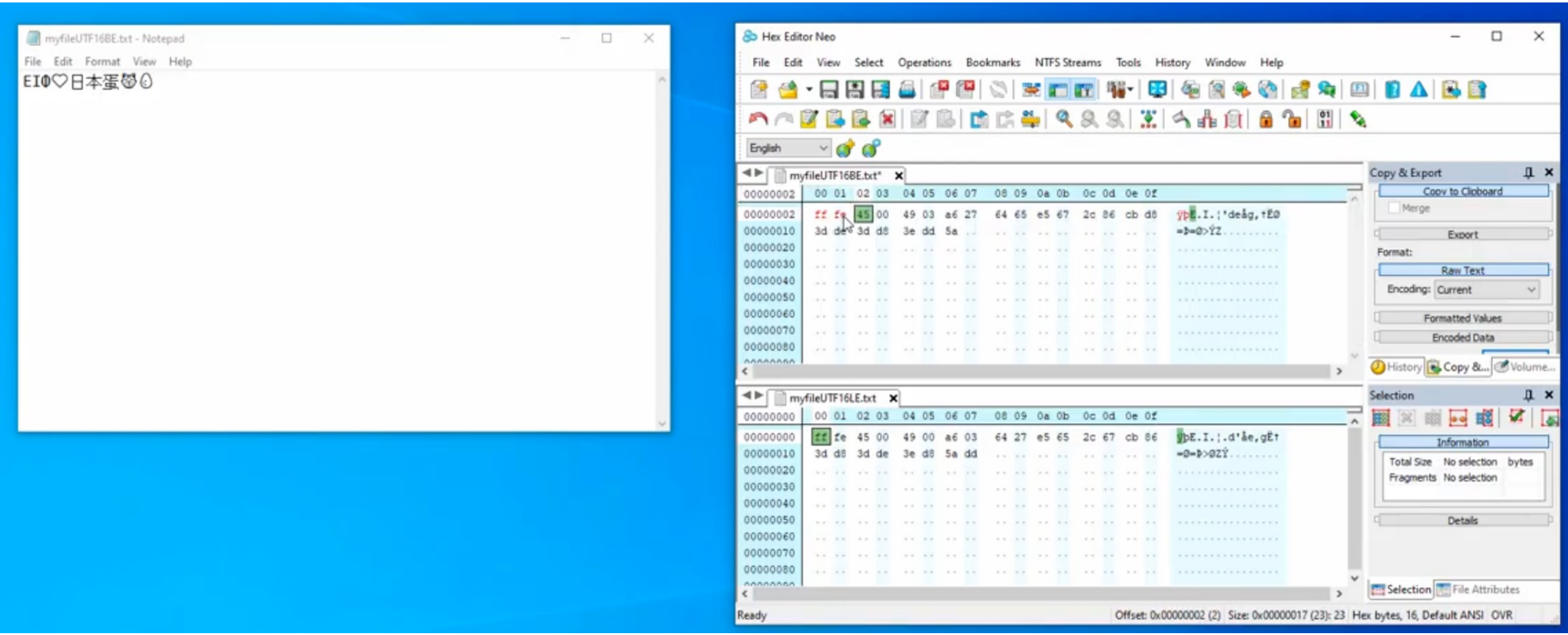

*Untitled - Notepad
File Edit Format View Help
Save As
This PC › Desktop › Unicode Demo
Search Unicode Demo
Organise
New folder
OneDrive
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
System (C:)
New Volume (D:
Name
Date modified
Type
Size
No items match your search.
File name: *.txt
Save as type: Text Documents (*.txt)
Hide Folders
Encoding: UTF-8
Save
Cancel
ANSI
UTF-16 LE
UTF-16 BE
UTF-8
UTF-8 with BOM

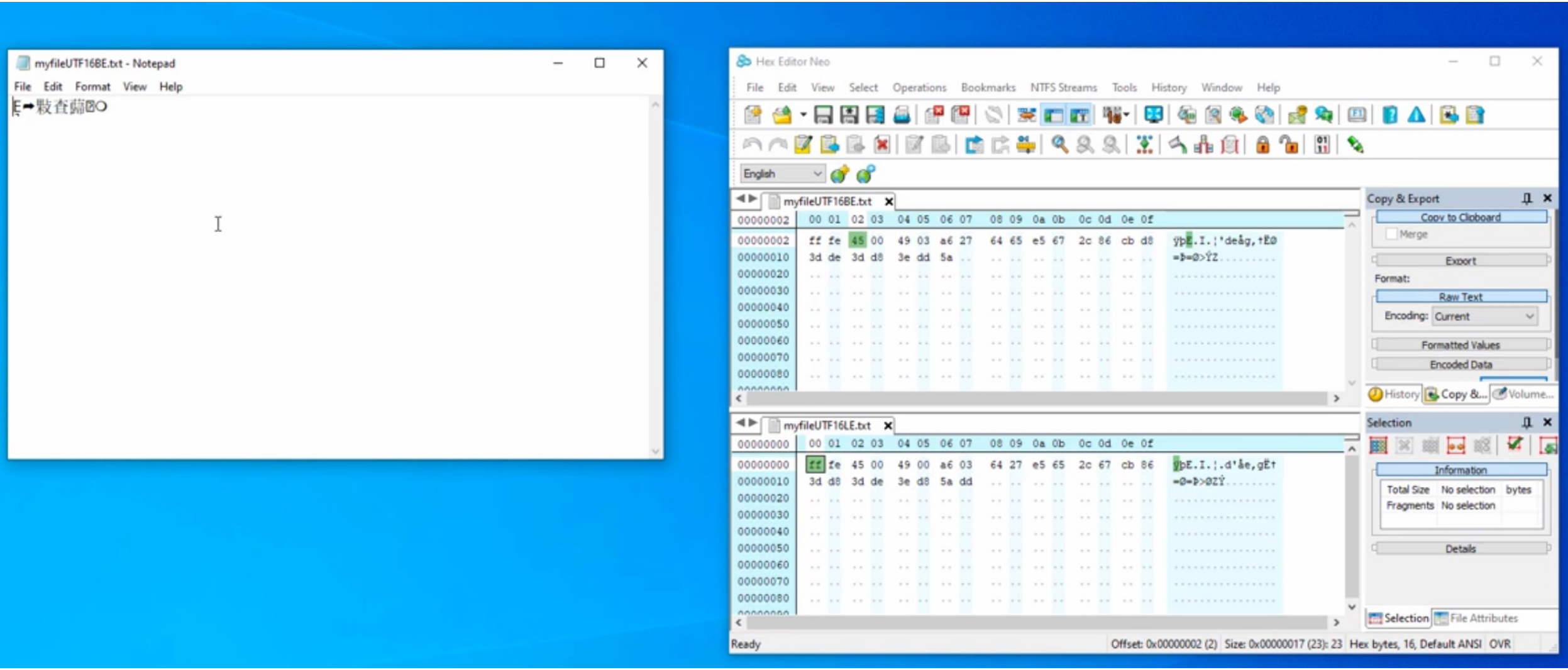

*Untitled - Notepad
File Edit Format View Help
Save As
This PC › Desktop › Unicode Demo
Search Unicode Demo
Organise
New folder
OneDrive
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
System (C:)
New Volume (D:
Name
Date modified
Type
Size
No items match your search.
File name: myfileUTF16LE.txt
Save as type: Text Documents (*.txt)
Hide Folders
Encoding: UTF-16 LE
Save
Cancel

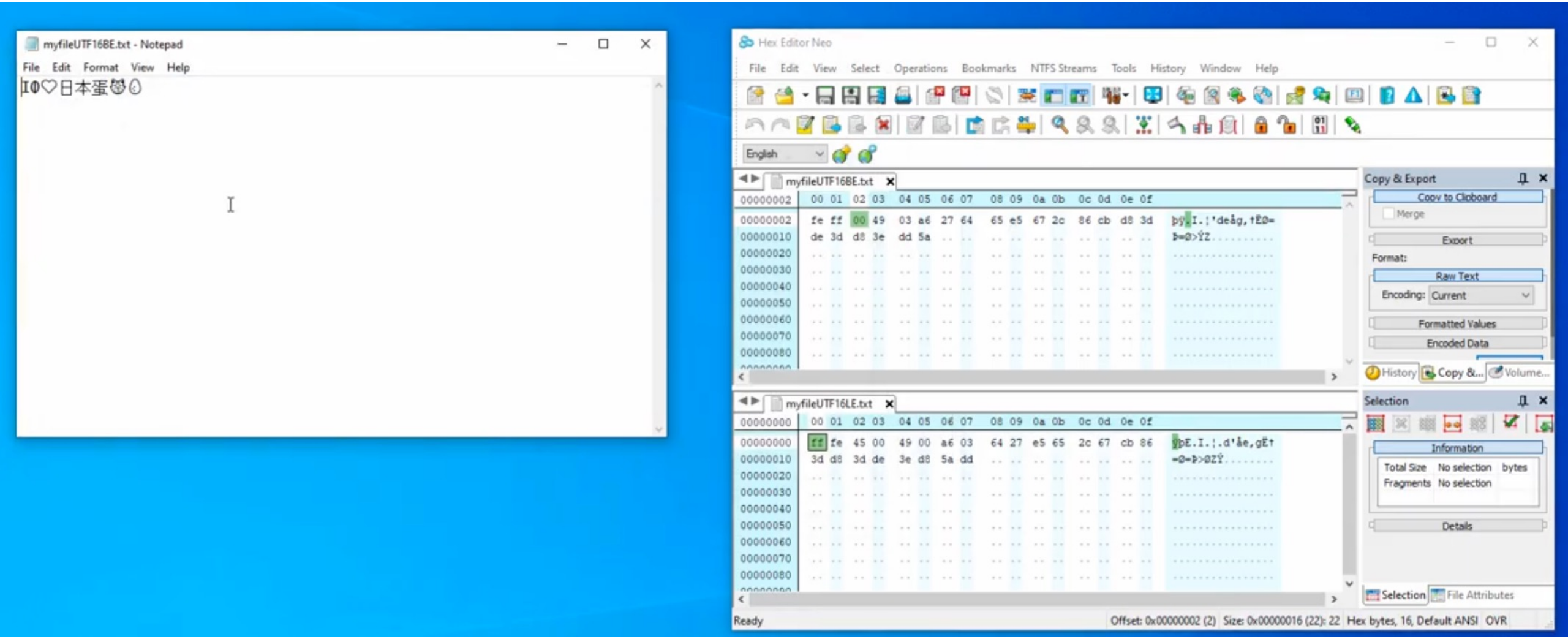

myfileUTF16LE.txt - Notepad
File Edit Format View Help
Save As
This PC › Desktop › Unicode Demo
Search Unicode Demo
Organise
New folder
OneDrive
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
System (C:)
New Volume (D:
Name
Date modified
Type
Size
myfileUTF16LE.txt
10/06/2020 11:26
Text Document
1 KB
File name: myfileUTF16BE.txt
Save as type: Text Documents (*.txt)
Hide Folders
Encoding: UTF-16 BE
Save
Cancel


myfileUTF16BE.txt - Notepad
File Edit Format View Help
EI0♡日本蛋🐻💧
Hex Editor Neo
File Edit View NTFS Streams Tools History Window Help
English
Copy & Export
Copy to Clipboard
Merge
Export
Format:
Raw Text
Encoding: Current
Formatted Values
Encoded Data
History
Copy &...
Volume...
Selection
Information
Total Size No selection bytes
Fragments No selection
Details
Selection
File Attributes
Ready


myfileUTF16BE.txt - Notepad
File Edit Format View Help
EI0♡日本蛋🐼💧
Hex Editor Neo
File Edit View Select Operations Bookmarks NTFS Streams Tools History Window Help
English
myfileUTF16BE.txt
00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
fe ff 00 45 00 49 03 a6 27 64 65 e5 67 2c 86 cb
d8 3d de 3d d8 3e dd 5a
myfileUTF16LE.txt
ff fe 45 00 49 00 a6 03 64 27 e5 65 2c 67 cb 86
3d d8 3d de 3e d8 5a dd
Copy & Export
Copy to Clipboard
Merge
Export
Format:
Raw Text
Encoding: Current
Formatted Values
Encoded Data
History
Copy &...
Volume...
Selection
Information
Total Size No selection bytes
Fragments No selection
Details
Selection
File Attributes
Ready
Offset: 00000000 (0)
Size: 0x00000018 (24): 24
Hex bytes, 16, Default ANSI
OVR


myfileUTF16BE.txt - Notepad
File Edit Format View Help
EI🅾♡日本蛋🐻💧
Hex Editor Neo
File Edit View Select Operations Bookmarks NTFS Streams Tools History Window Help
English
myfileUTF16BE.txt*
00000002 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
00000002 ff fe 45 00 49 03 a6 27 64 65 e5 67 2c 86 cb d8 ÿþE.I.¦'deåg,†ËØ
00000010 3d de 3d d8 3e dd 5a =Þ=Ø>ÝZ
myfileUTF16LE.txt
00000000 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
00000000 ff fe 45 00 49 00 a6 03 64 27 e5 65 2c 67 cb 86 ÿþE.I.¦.d'åe,gË†
00000010 3d d8 3d de 3e d8 5a dd =Ø=Þ>ØZÝ
Copy & Export
Copy to Clipboard
Merge
Export
Format:
Raw Text
Encoding: Current
Formatted Values
Encoded Data
History Copy &... Volume...
Selection
Information
Total Size No selection bytes
Fragments No selection
Details
Selection File Attributes
Ready
Offset: 0x00000002 (2) Size: 0x00000017 (23): 23 Hex bytes, 16, Default ANSI OVR


myfileUTF16BE.txt - Notepad
File Edit Format View Help
Hex Editor Neo
File Edit View Select Operations Bookmarks NTFS Streams Tools History Window Help
English
myfileUTF16BE.txt
00000002 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
00000002 ff fe 45 00 49 03 a6 27 64 65 e5 67 2c 86 cb d8
00000010 3d de 3d d8 3e dd 5a
Copy & Export
Copy to Clipboard
Merge
Export
Format:
Raw Text
Encoding: Current
Formatted Values
Encoded Data
History
Copy &...
Volume...
myfileUTF16LE.txt
00000000 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
00000000 ff fe 45 00 49 00 a6 03 64 27 e5 65 2c 67 cb 86
00000010 3d d8 3d de 3e d8 5a dd
Selection
Information
Total Size No selection bytes
Fragments No selection
Details
Selection
File Attributes
Ready
Offset: 0x00000002 (2) Size: 0x00000017 (23): 23
Hex bytes, 16, Default ANSI OVR


myfileUTF16BE.txt - Notepad
File Edit Format View Help
Hex Editor Neo
File Edit View Select Operations Bookmarks NTFS Streams Tools History Window Help
English
myfileUTF16BE.txt
00000002 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
00000002 fe ff 00 49 03 a6 27 64 65 e5 67 2c 86 cb d8 3d
00000010 de 3d d8 3e dd 5a
Copy & Export
Copy to Clipboard
Merge
Export
Format:
Raw Text
Encoding: Current
Formatted Values
Encoded Data
History
Copy &...
Volume...
myfileUTF16LE.txt
00000000 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
00000000 ff fe 45 00 49 00 a6 03 64 27 e5 65 2c 67 cb 86
00000010 3d d8 3d de 3e d8 5a dd
Selection
Information
Total Size No selection bytes
Fragments No selection
Details
Selection
File Attributes
Ready
Offset: 0x00000002 (2)
Size: 0x00000016 (22): 22
Hex bytes, 16, Default ANSI
OVR

Unicode Transformation Format

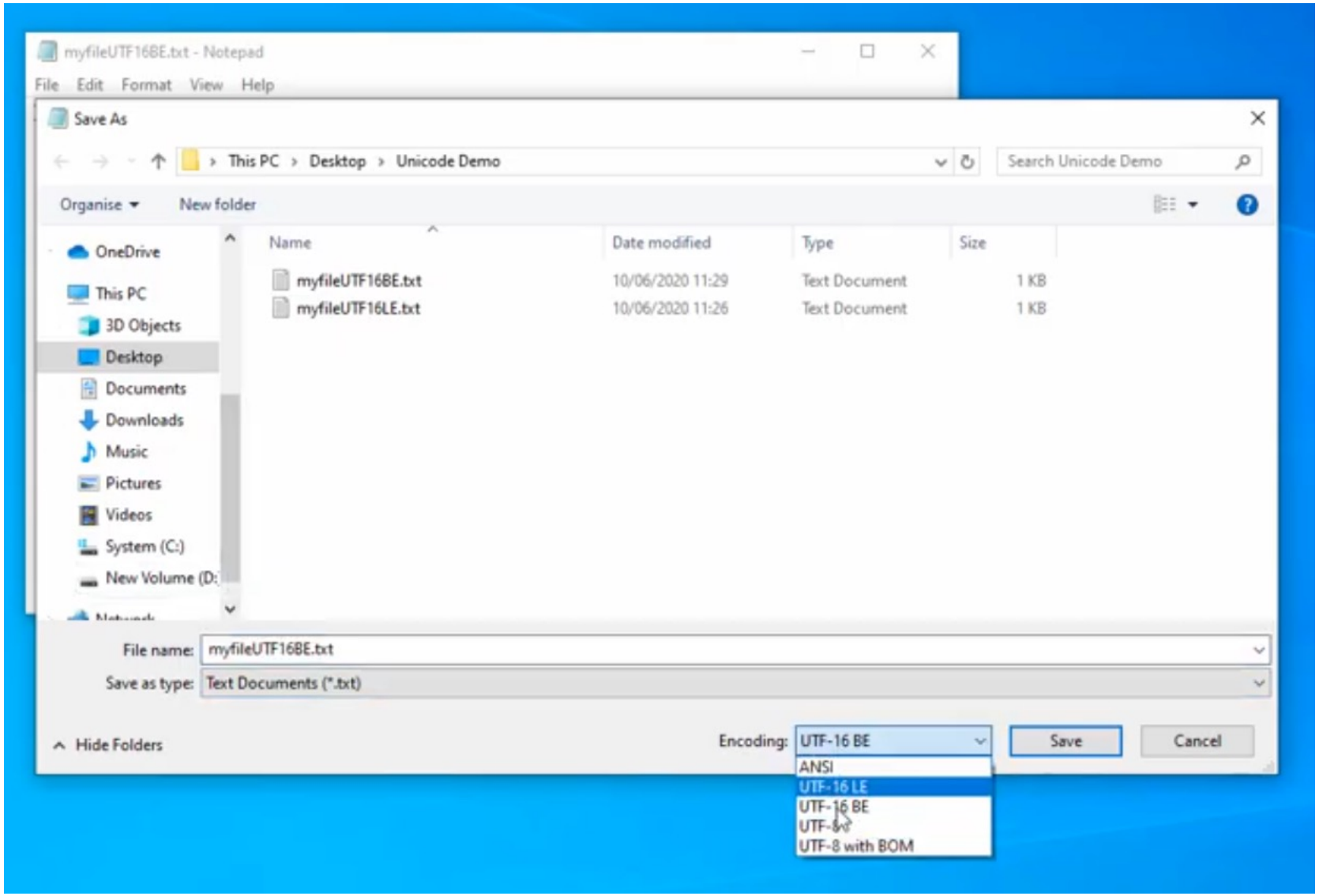
UTF-32

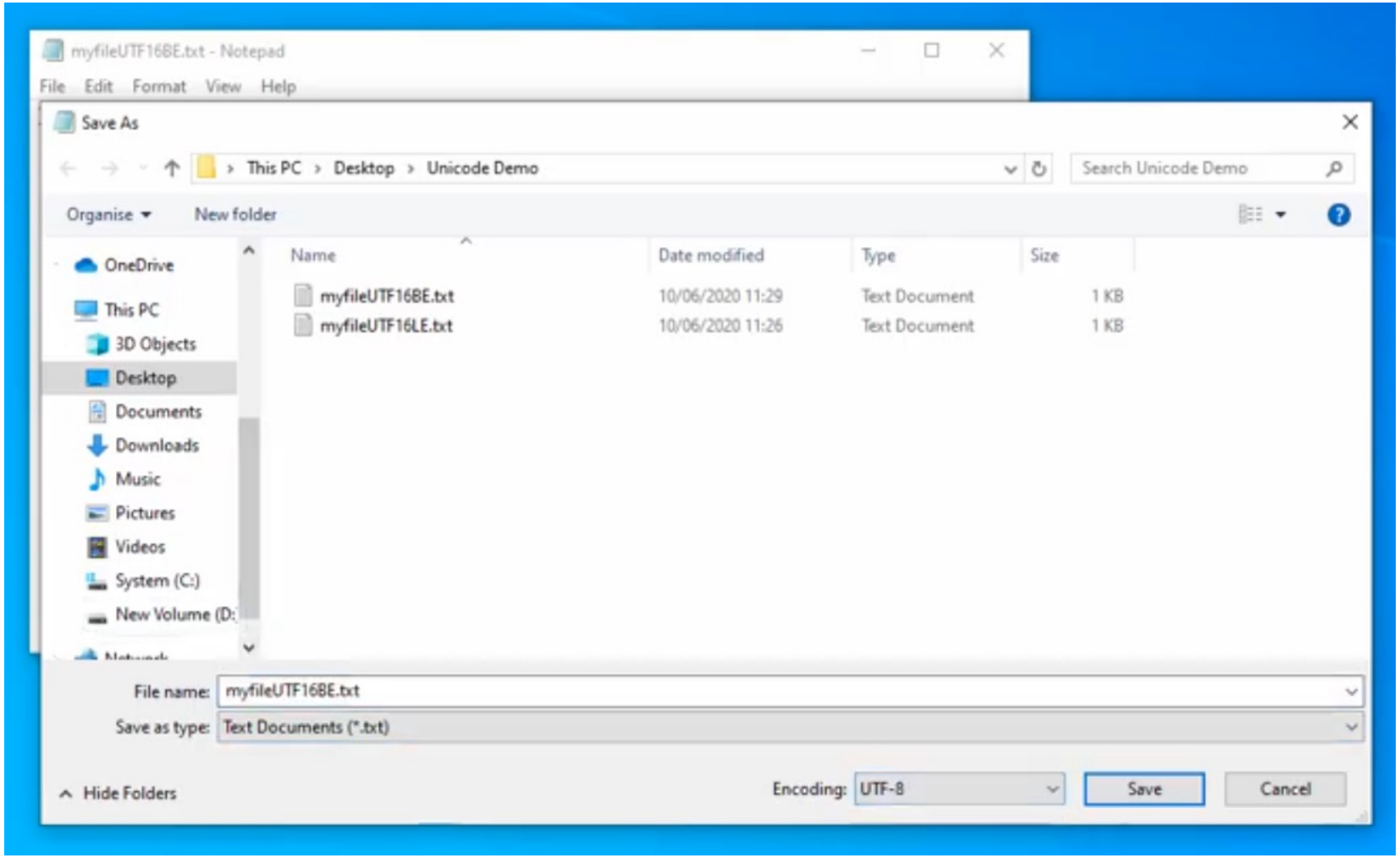
		Big Endian		Little Endian	
Code Point	Character	UTF-32 encoding	UTF-32 (Hex)	UTF-32 encoding	UTF-32 (Hex)
U+0045	E	00000000 00000000 00000000 01000101	00 00 00 45	01000101 00000000 00000000 00000000	45 00 00 00
U+0049	I	00000000 00000000 00000000 01000101	00 00 00 49	01000101 00000000 00000000 00000000	49 00 00 00
U+03A6	Φ	00000000 00000000 00000011 10100110	00 00 03 D5	10100110 00000011 00000000 00000000	D5 03 00 00
U+2764	❤	00000000 00000000 00000011 11010101	00 00 27 64	11010101 00000011 00000000 00000000	64 27 00 00
U+65E5	日	00000000 00000000 10000110 11001011	00 00 65 E5	11001011 10000110 00000000 00000000	E5 65 00 00
U+672C	本	00000000 00000000 01100111 00101100	00 00 67 2C	00101100 01100111 00000000 00000000	2C 67 00 00
U+86CB	蛋	00000000 00000000 10000110 11001011	00 00 86 CB	11001011 10000110 00000000 00000000	CB 86 00 00
U+1F63D	😽	00000000 00000001 11110110 00111101	00 01 F6 3D	00111101 11110110 00000001 00000000	3D F6 01 00
U+1F95A	🥚	00000000 00000001 11111001 01011010	00 01 F9 5A	01011010 11111001 00000001 00000000	5A F9 01 00

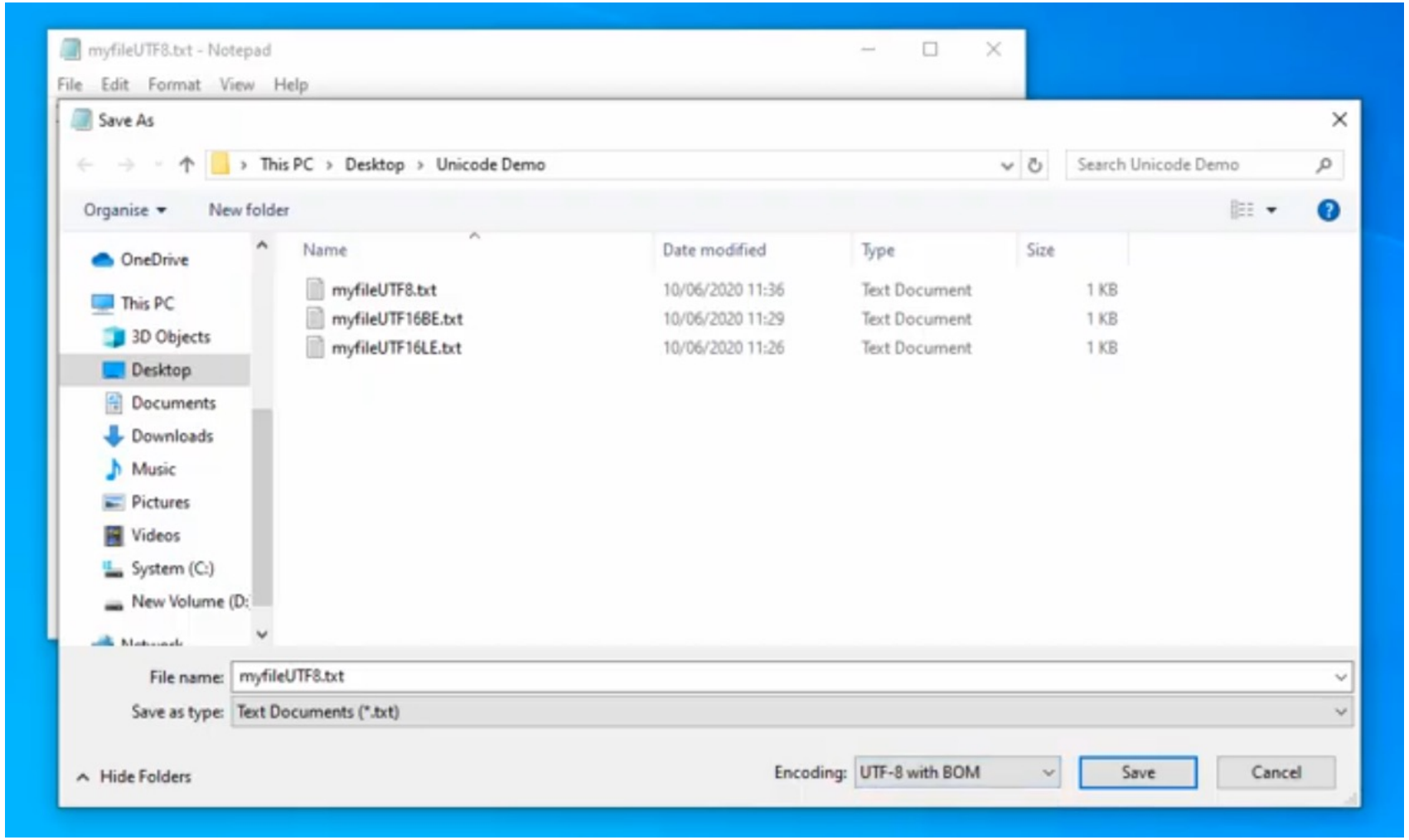
Unicode Transformation Format

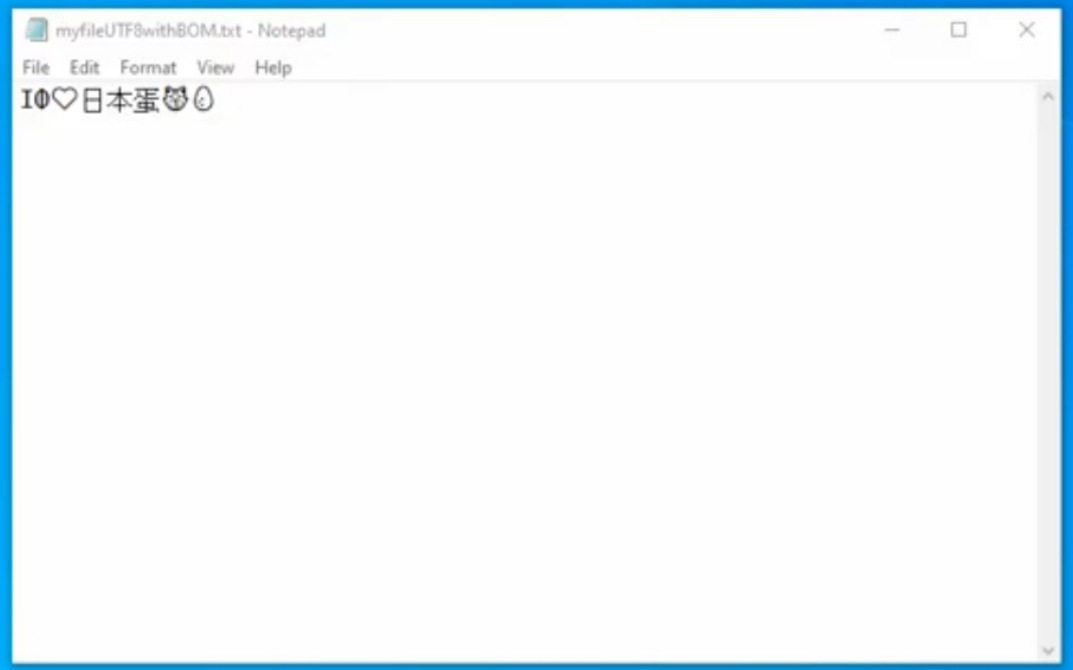
UTF-8

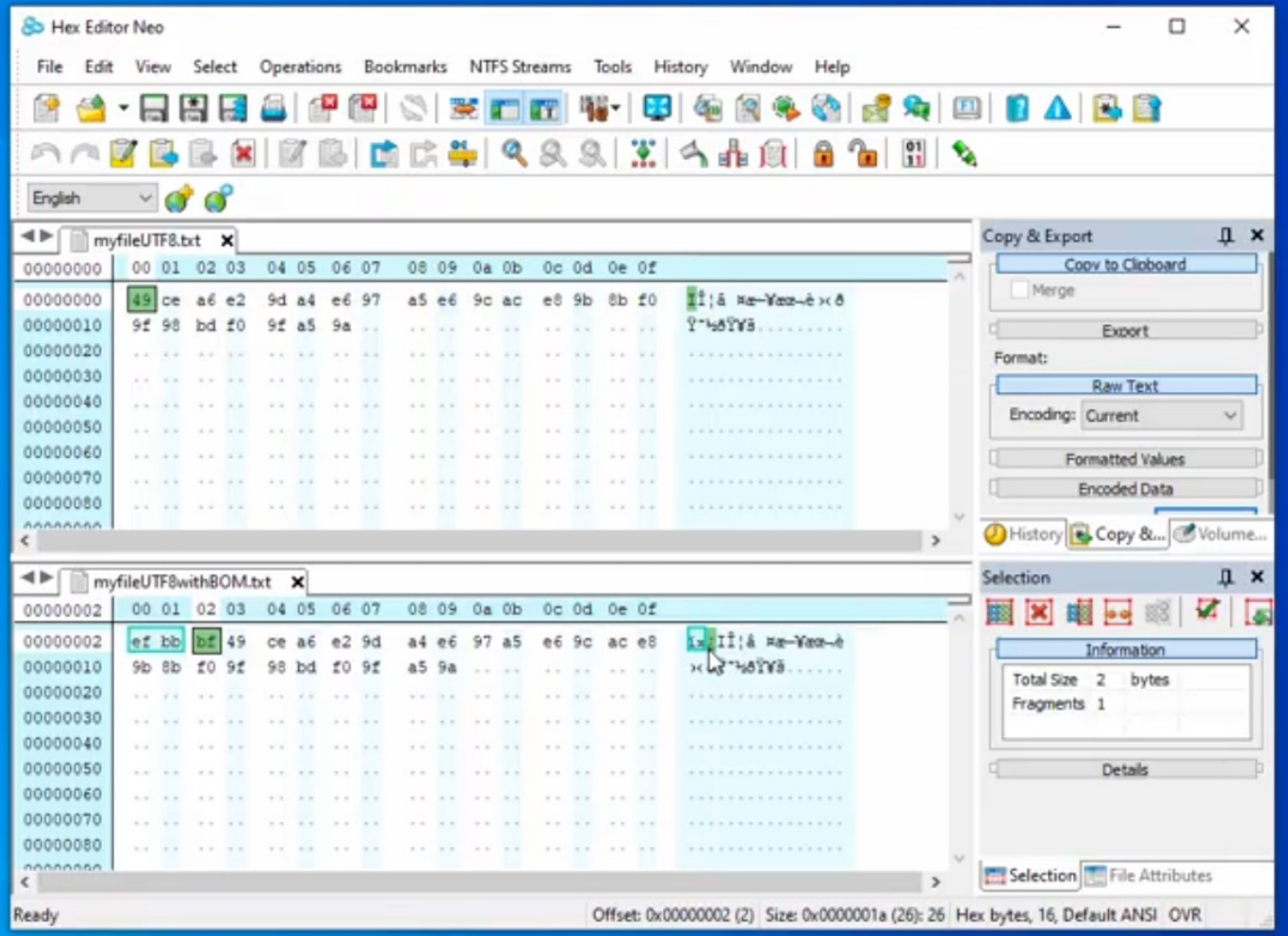
Code Point	Character	UTF-8 encoding	UTF-8 (Hex)
U+0045	E	01000101	45
U+0049	I	01001001	49
U+03A6	Φ	11000011 10100110	CE A6
U+2764	❤	11100010 10011101 10100100	E2 9D A4
U+65E5	日	11100110 10010111 10100101	E6 97 A5
U+672C	本	11100110 10011100 10101100	E6 9C AC
U+86CB	蛋	11101000 10011011 10001011	E8 9B 8B
U+1F63D	😽	11110000 10011111 10011000 10111101	F0 9F 98 BD
U+1F95A	🥚	11110000 10011111 10100101 10011010	F0 9F A5 9A


myfileUTF16BE.txt - Notepad
File Edit Format View Help
Save As
This PC › Desktop › Unicode Demo
Search Unicode Demo
Organise
New folder
OneDrive
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
System (C:)
New Volume (D:
Name
Date modified
Type
Size
myfileUTF16BE.txt
10/06/2020 11:29
Text Document
1 KB
myfileUTF16LE.txt
10/06/2020 11:26
Text Document
1 KB
File name: myfileUTF16BE.txt
Save as type: Text Documents (*.txt)
Hide Folders
Encoding: UTF-16 BE
Save
Cancel
ANSI
UTF-16 LE
UTF-16 BE
UTF-8
UTF-8 with BOM


myfileUTF16BE.txt - Notepad
File Edit Format View Help
Save As
This PC › Desktop › Unicode Demo
Search Unicode Demo
Organise
New folder
OneDrive
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
System (C:)
New Volume (D:
Name
Date modified
Type
Size
myfileUTF16BE.txt
10/06/2020 11:29
Text Document
1 KB
myfileUTF16LE.txt
10/06/2020 11:26
Text Document
1 KB
File name: myfileUTF16BE.txt
Save as type: Text Documents (*.txt)
Hide Folders
Encoding: UTF-8
Save
Cancel


myfileUTF8.txt - Notepad
File Edit Format View Help
Save As
This PC › Desktop › Unicode Demo
Search Unicode Demo
Organise
New folder
OneDrive
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
System (C:)
New Volume (D:
Name
Date modified
Type
Size
myfileUTF8.txt 10/06/2020 11:36 Text Document 1 KB
myfileUTF16BE.txt 10/06/2020 11:29 Text Document 1 KB
myfileUTF16LE.txt 10/06/2020 11:26 Text Document 1 KB
File name: myfileUTF8.txt
Save as type: Text Documents (*.txt)
Hide Folders
Encoding: UTF-8 with BOM
Save
Cancel

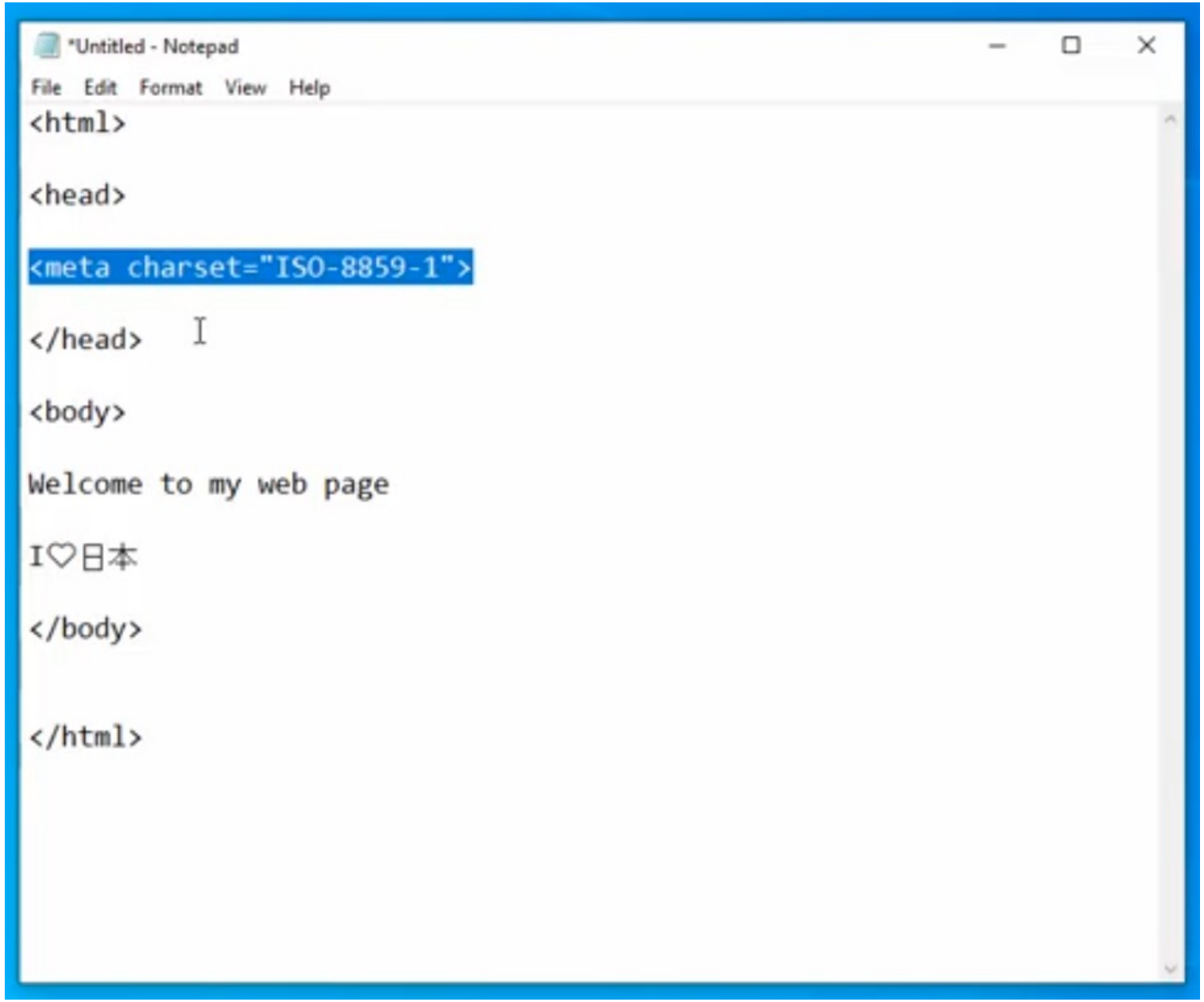




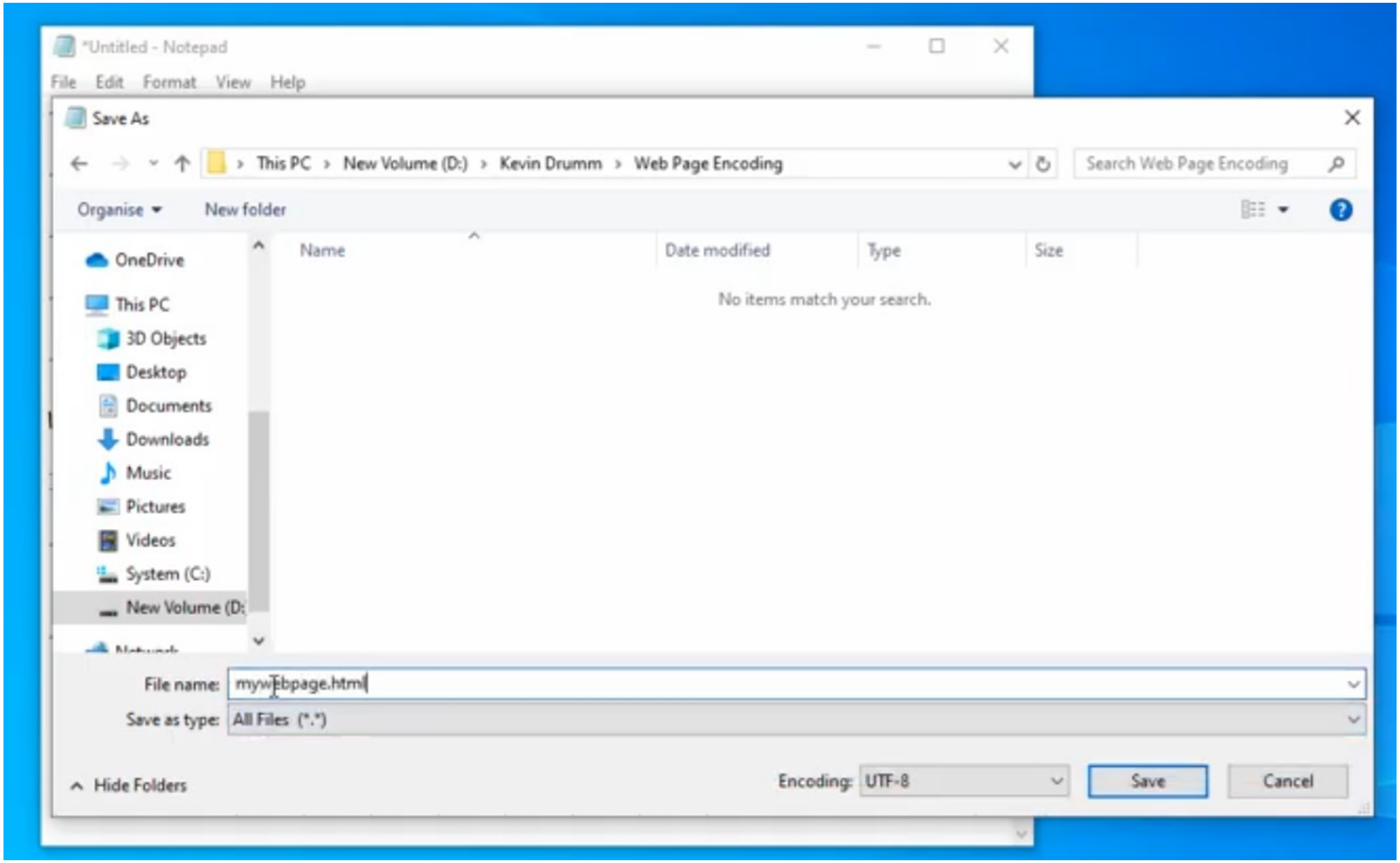
Byte Order mark

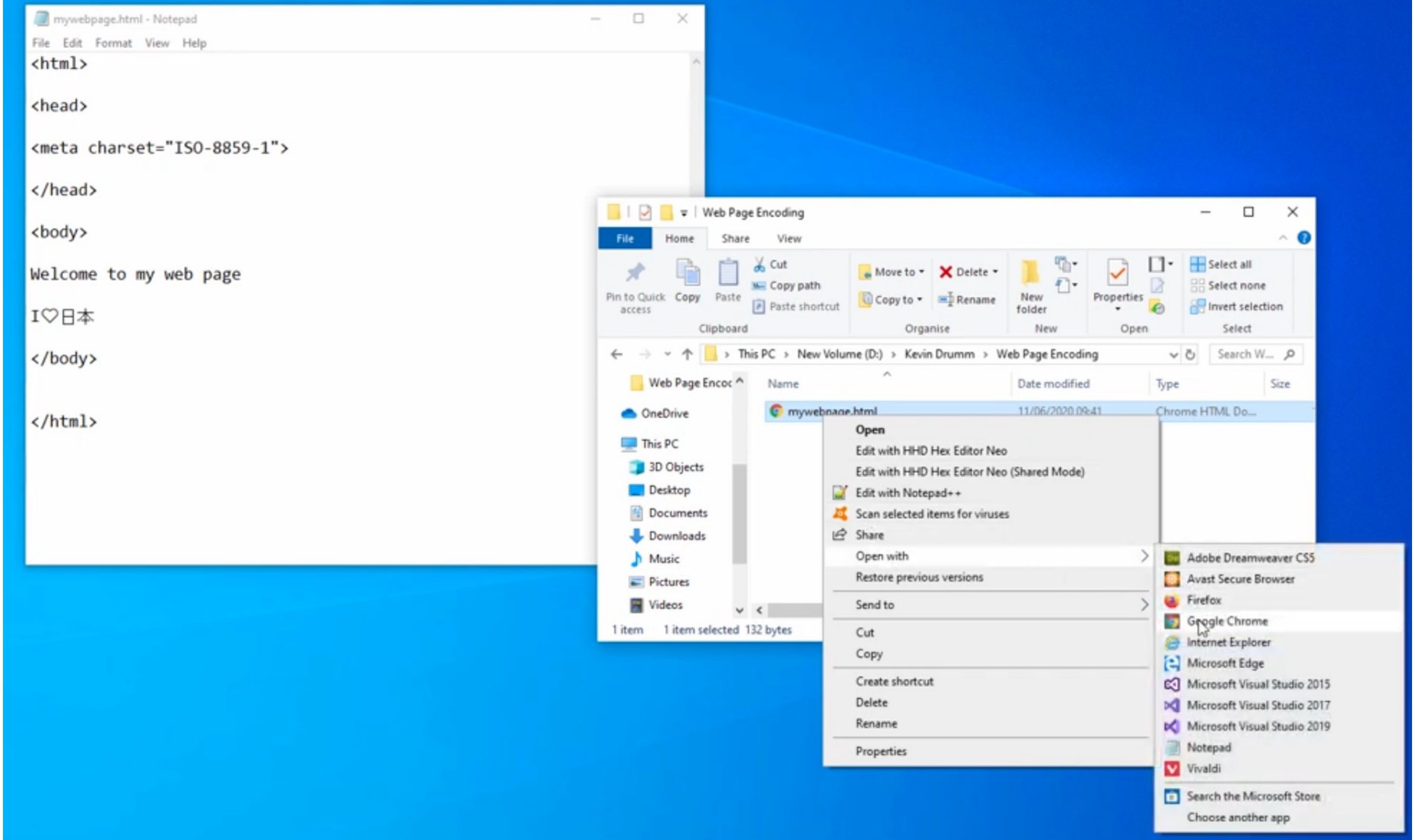
This is single byte fixed end encoding. This is the defualt browser encoding used by some browsers.

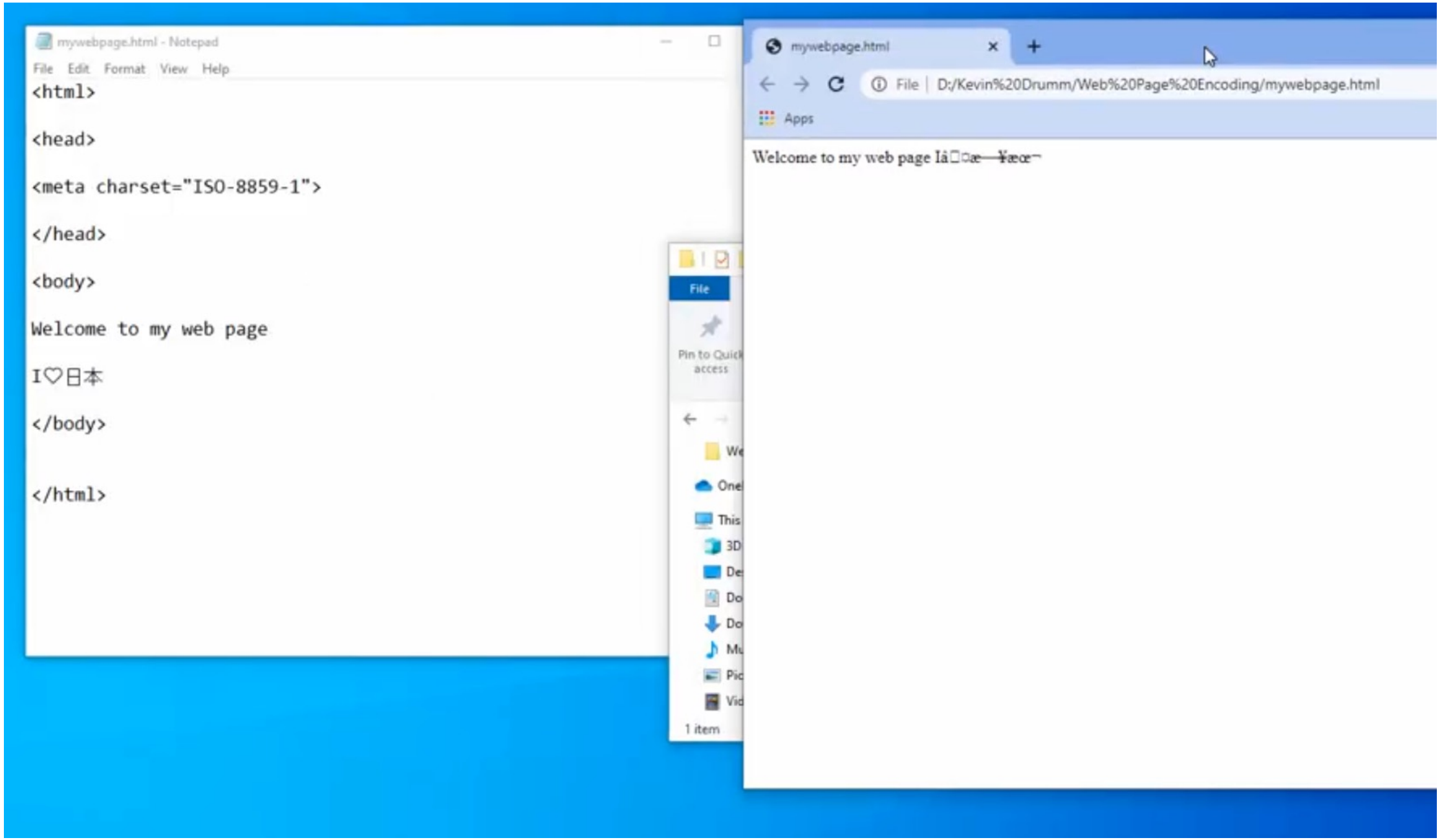
These characters have large code points.



When we save MS word document in html - the default encoding is Windows-1252


*Untitled - Notepad
File Edit Format View Help
Save As
This PC › New Volume (D:) › Kevin Drumm › Web Page Encoding
Search Web Page Encoding
Organise
New folder
OneDrive
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
System (C:)
New Volume (D:
Name
Date modified
Type
Size
No items match your search.
File name: mywebpage.html
Save as type: All Files (*.*)
Hide Folders
Encoding: UTF-8
Save
Cancel


mywebpage.html - Notepad
File Edit Format View Help
<html>
<head>
<meta charset="ISO-8859-1">
</head>
<body>
Welcome to my web page
I♡日本
</body>
</html>
Web Page Encoding
File Home Share View
Pin to Quick access
Copy
Paste
Cut
Copy path
Paste shortcut
Clipboard
Move to
Delete
Copy to
Rename
Organise
New folder
New
Properties
Select all
Select none
Invert selection
Select
This PC › New Volume (D:) › Kevin Drumm › Web Page Encoding
Search W...
Web Page Encoc
OneDrive
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Name
Date modified
Type
Size
mywebpage.html
11/06/2020 09:41
Chrome HTML Do...
1 item
1 item selected 132 bytes
Open
Edit with HHD Hex Editor Neo
Edit with HHD Hex Editor Neo (Shared Mode)
Edit with Notepad++
Scan selected items for viruses
Share
Open with
Restore previous versions
Send to
Cut
Copy
Create shortcut
Delete
Rename
Properties
Adobe Dreamweaver CS5
Avast Secure Browser
Firefox
Google Chrome
Internet Explorer
Microsoft Edge
Microsoft Visual Studio 2015
Microsoft Visual Studio 2017
Microsoft Visual Studio 2019
Notepad
Vivaldi
Search the Microsoft Store
Choose another app


mywebpage.html - Notepad
File Edit Format View Help
<html>
<head>
<meta charset="ISO-8859-1">
</head>
<body>
Welcome to my web page
I♡日本
</body>
</html>
mywebpage.html
File D:/Kevin%20Drumm/Web%20Page%20Encoding/mywebpage.html
Apps
Welcome to my web page Iâ□¤æ—¥æœ¬
File
Pin to Quick access
1 item

mywebpage.html - Notepad

File Edit Format View Help

```
<html>

<head>

<meta charset="ISO-8859-1">

</head>

<body>

Welcome to my web page

&#73 &#10084 &#26085 &#26412

I♡日本

</body>


</html>
```

mywebpage.html

File | D:/Kevin%20Drumm/Web%20Page%20Encoding/mywebpage.html

Apps

ome to my web page Iâ□¤æ—¥æœ¬

Base 10 values of code points

mywebpage.html - Notepad

File Edit Format View Help

```
<html>

<head>

<meta charset="ISO-8859-1">

</head>

<body>

Welcome to my web page

&#73 &#10084 &#26085 &#26412

I♡日本

</body>


</html>
```

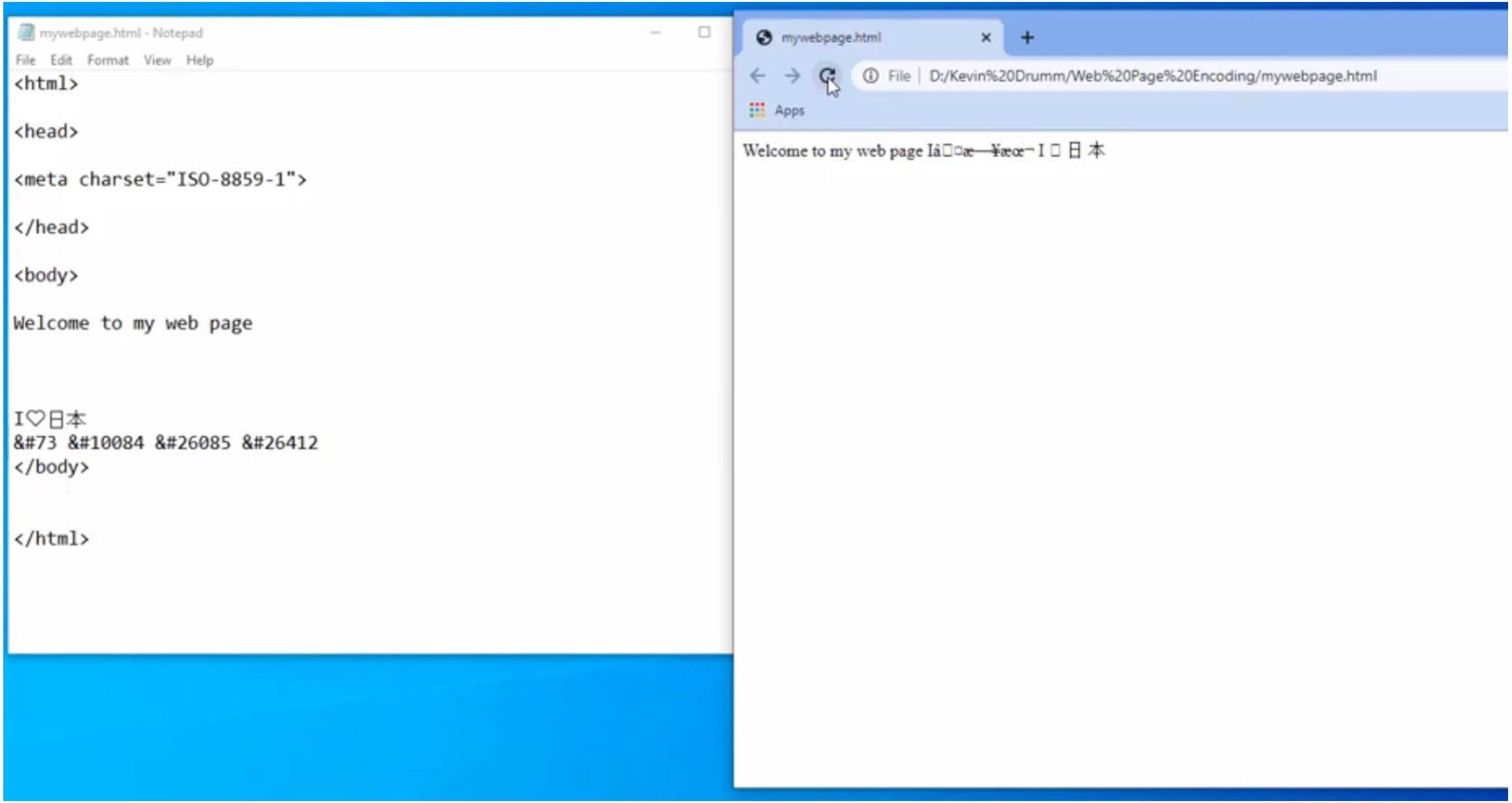
mywebpage.html

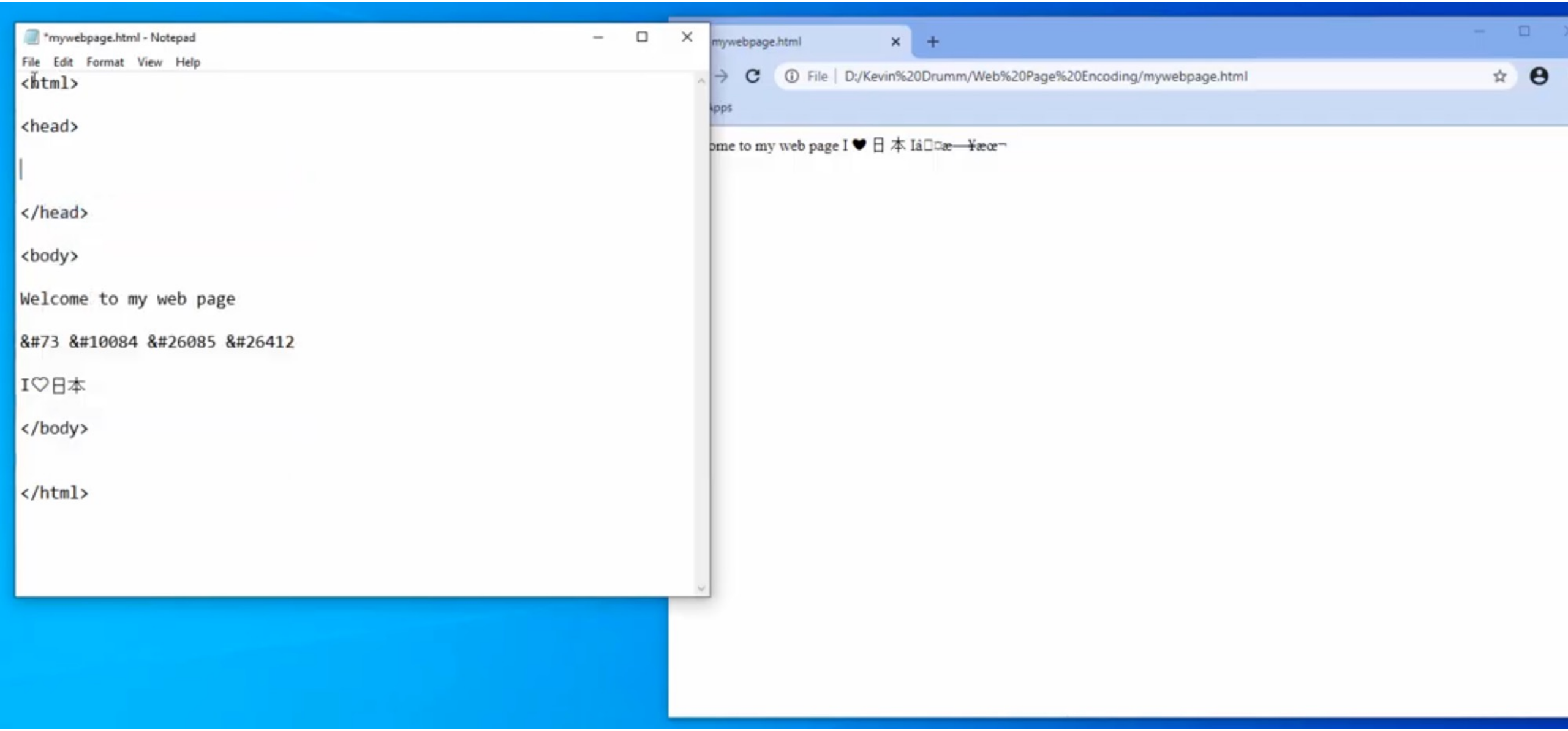
File | D:/Kevin%20Drumm/Web%20Page%20Encoding/mywebpage.html

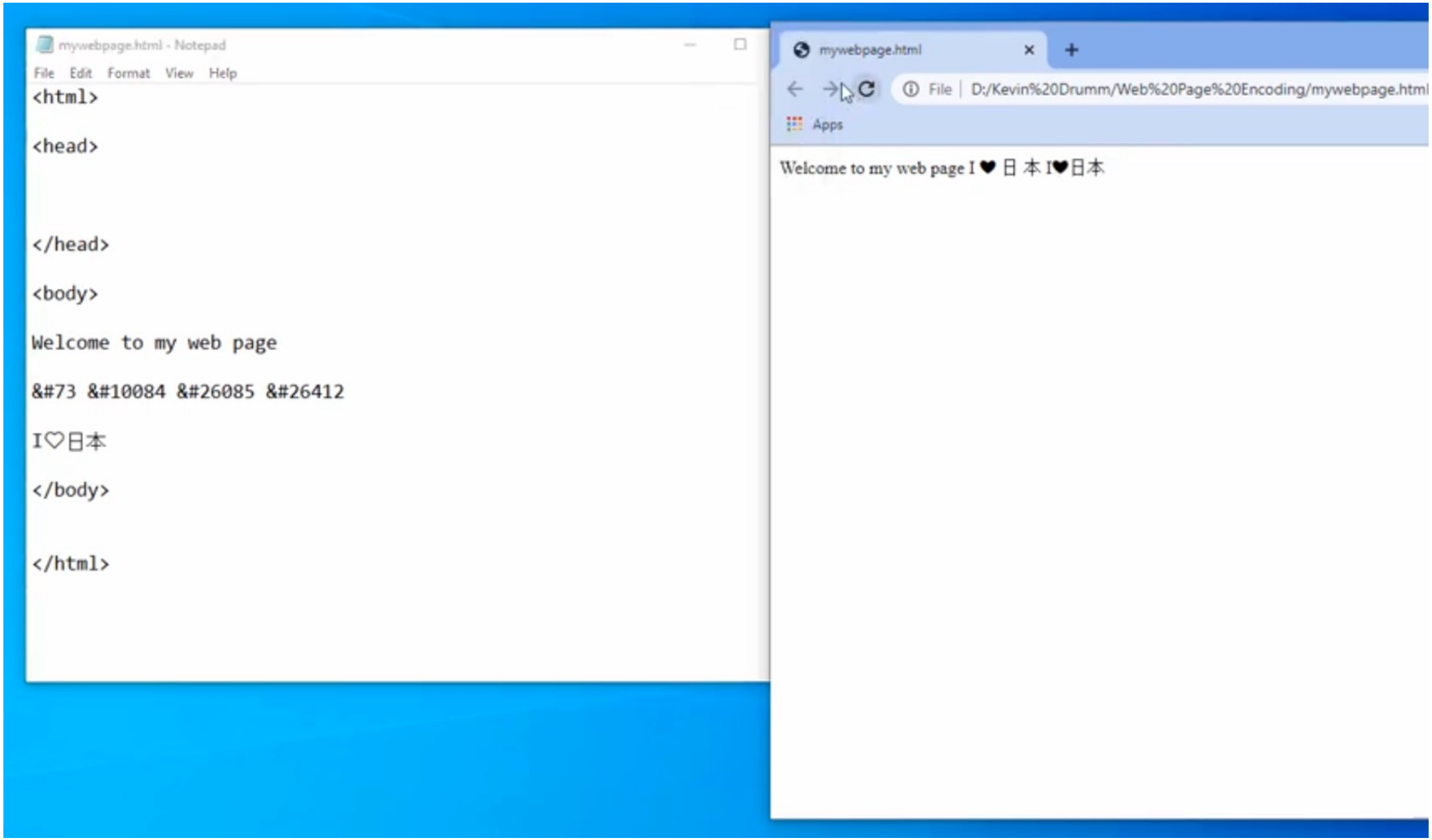
Apps

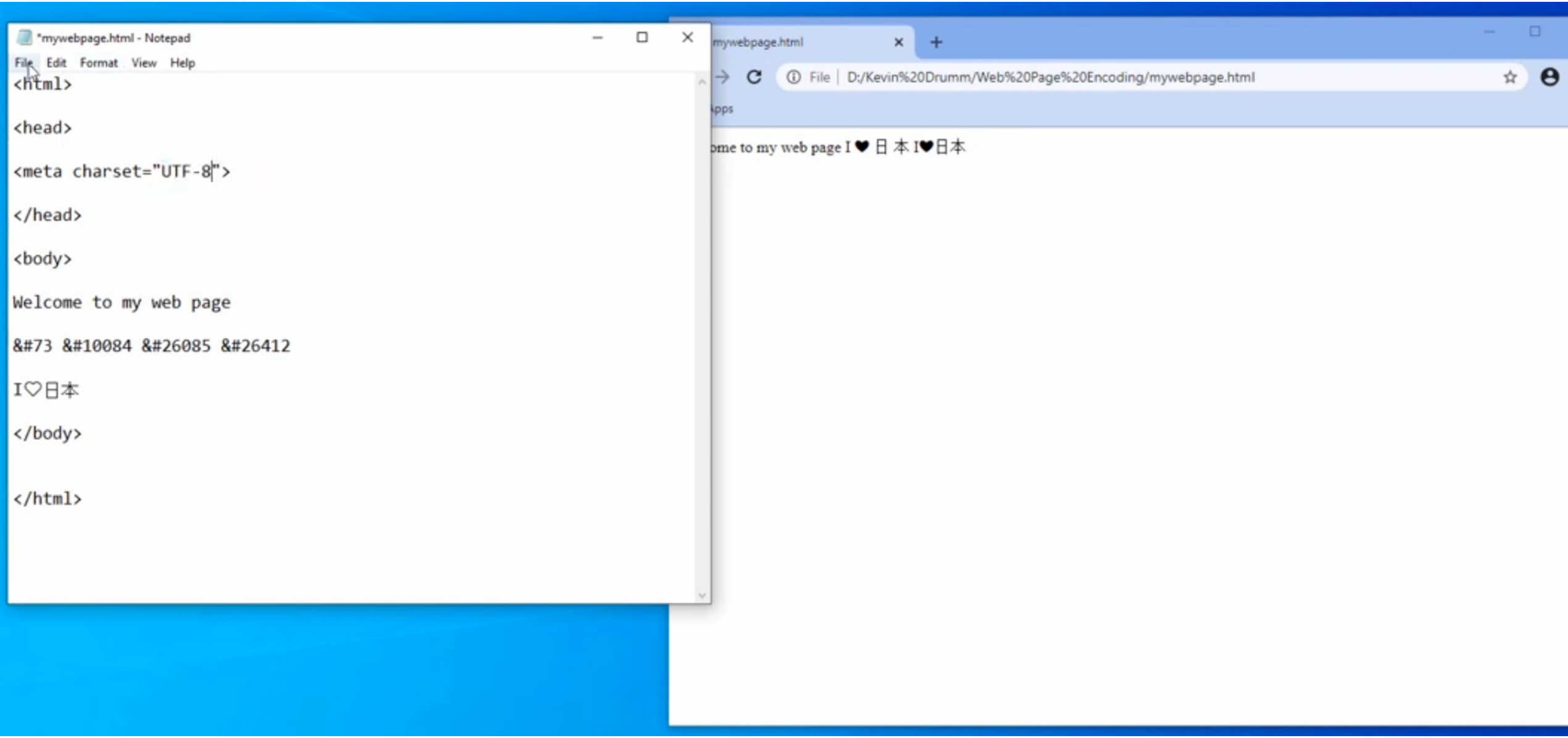
Welcome to my web page I ❤ 日 本 Iá□□æ—¥æœ¬

Browser interp them correctly


mywebpage.html - Notepad
File Edit Format View Help
<html>
<head>
<meta charset="ISO-8859-1">
</head>
<body>
Welcome to my web page
I♡日本
&#73 &#10084 &#26085 &#26412
</body>
</html>
mywebpage.html
File | D:/Kevin%20Drumm/Web%20Page%20Encoding/mywebpage.html
Apps
Welcome to my web page Iá□□æ—¥æœ¬ I □ 日 本


*mywebpage.html - Notepad
File Edit Format View Help
<html>
<head>
</head>
<body>
Welcome to my web page
&#73 &#10084 &#26085 &#26412
I♡日本
</body>
</html>
mywebpage.html
File | D:/Kevin%20Drumm/Web%20Page%20Encoding/mywebpage.html
Apps
ome to my web page I ♥ 日 本 Iá□□æ—¥æœ¬


mywebpage.html - Notepad
File Edit Format View Help
<html>
<head>
</head>
<body>
Welcome to my web page
&#73 &#10084 &#26085 &#26412
I♡日本
</body>
</html>
mywebpage.html
File | D:/Kevin%20Drumm/Web%20Page%20Encoding/mywebpage.html
Apps
Welcome to my web page I ❤ 日 本 I❤日本


*mywebpage.html - Notepad
File Edit Format View Help
<html>
<head>
<meta charset="UTF-8">
</head>
<body>
Welcome to my web page
&#73 &#10084 &#26085 &#26412
I♡日本
</body>
</html>
mywebpage.html
File D:/Kevin%20Drumm/Web%20Page%20Encoding/mywebpage.html
Apps
ome to my web page I ❤ 日 本 I❤日本

Unicode Transformation Format

UTF-8

Code Point	Character	UTF-8 encoding	UTF-8 (Hex)
U+0045	E	01000101	45
U+0049	I	01001001	49
U+03A6	Φ	11000011 10100110	CE A6
U+2764	❤	11100010 10011101 10100100	E2 9D A4
U+65E5	日	11100110 10010111 10100101	E6 97 A5
U+672C	本	11100110 10011100 10101100	E6 9C AC
U+86CB	蛋	11101000 10011011 10001011	E8 9B 8B
U+1F63D	😽	11110000 10011111 10011000 10111101	F0 9F 98 BD
U+1F95A	🥚	11110000 10011111 10100101 10011010	F0 9F A5 9A

0011001011110000100111111001000010011101111000101000101010010101110000101010110000110010111100001001111110010000100111010011111

Exercise 21

Create files with different encoding schemes using nodepad. View and comapre them using Hexa Editior Neo and upload the files and screenshots in google class room.

Exercise 22

What is the sense and repesentaion of the following string in UTF-8, UTF-16, and UTF-32

```
0011001011110000100111111001000010011101111000101000101010010101110000101010110000110010111100001001111110010000100111010011111
```