Numeric Representation II

CSCI 2050U - Computer Architecture

Randy J. Fortier
@randy_fortier

Outline

- Binary
- Hexadecimal
- Representing fractional numbers
- Representing characters

DECIMAL REPRESENTATION

$$297 = 2\times10^{2} + 9\times10^{1} + 7\times10^{0}$$

position

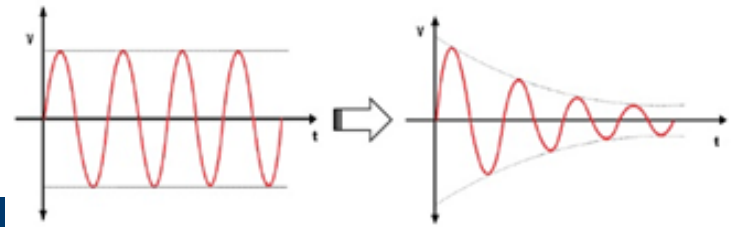
base - 10

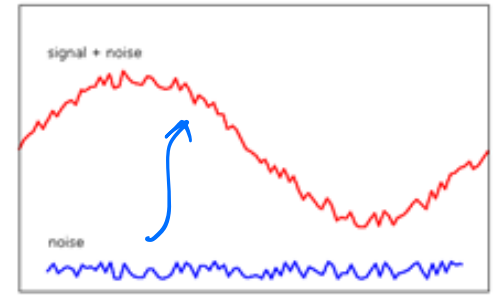
Binary

CSCI 2050U - Computer Architecture

Binary - Why?

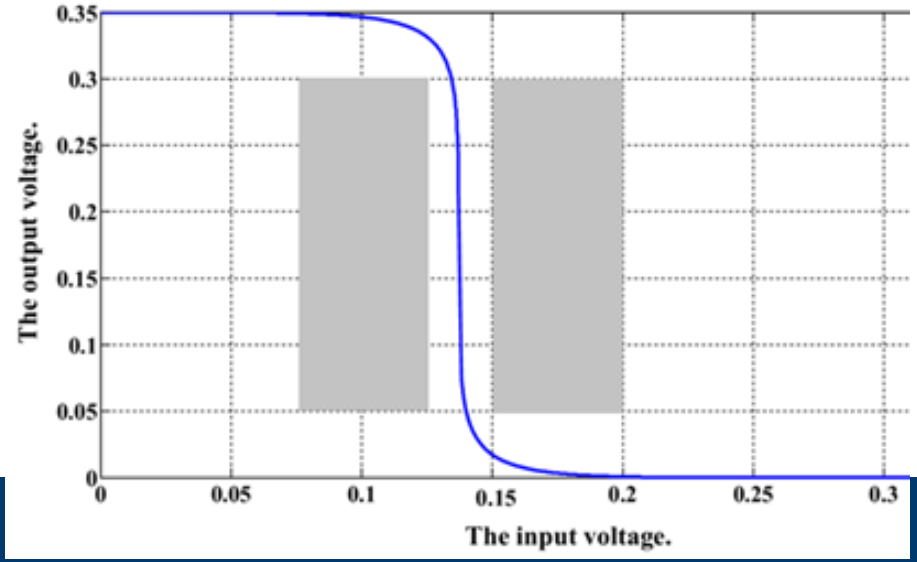
- A signal needs to differentiate one value from another
  - Electronic: Voltage or no voltage, current or no current
  - Electromagnetic: Varying frequency, varying amplitude, varying phase
- It might be possible to differentiate more than 2 values
  - e.g. Base 4: 0v, 1.5v, 3v, 4.5v
- Attenuation and noise can make these differences harder to discern
  - Binary maximizes the likelihood of detecting the correct value





Voltage Characteristics

- Let's say we decide that 0.0v is a 0 and 0.35v is a 1
  - What if we receive 0.32v?
- We typically design circuits that are not so black and white:
  - 0: 0.00-0.05v
  - 1: 0.30-0.35v
  - Undefined, otherwise



A Binary (Base 2) System

- A base 2 system would have 2 unique digits (0 and 1)
  - This is an 8-bit (binary digit) binary number:

$$01101101_2 = 0x2^7 + 1x2^6 + 1x2^5 + 0x2^4 + 1x2^3 + 1x2^2 + 0x2^1 + 1x2^0$$

Binary: 0, 1

1 1 0 1 → Decimal notation

3 2 1 0

$= 1 \times 2^3$

$+ 1 \times 2^2$

$+ 0 \times 2^1$

$+ 1 \times 2^0$

$= 8 + 4 + 0 + 1$

$= 13_{10}$

---

1 0 1 1 0 1 0 0 0 1

Text

(i) Convert Binary numbers to Decimal Numbers

(ii) Decimal to binary

192 64 0 0 0 0 0 0

1	1	0	0	0	0	0	0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
(128)	(64)	(32)	(16)	(8)	(4)	(2)	(1)

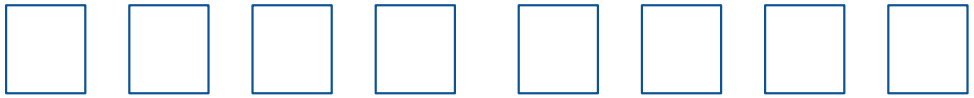
With 8-bits, the largest number we can represent is $2^8 - 1$

→ $1100\,0000_2 = 192_{10}$

$$
\begin{array}{r}
1 \\
128\overline{)192} \\
128 \\
\hline
64
\end{array}
\qquad
\begin{array}{r}
1 \\
64\overline{)64} \\
64 \\
\hline
0
\end{array}
\qquad
\begin{array}{r}
0 \\
32\overline{)0} \\
0 \\
\hline
0
\end{array}
$$

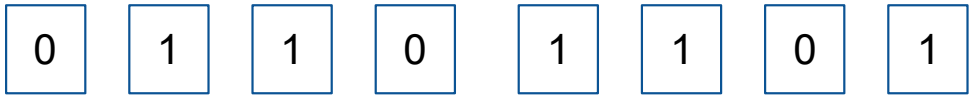
Converting from Binary to Decimal

- Converting from binary to decimal is rather easy:



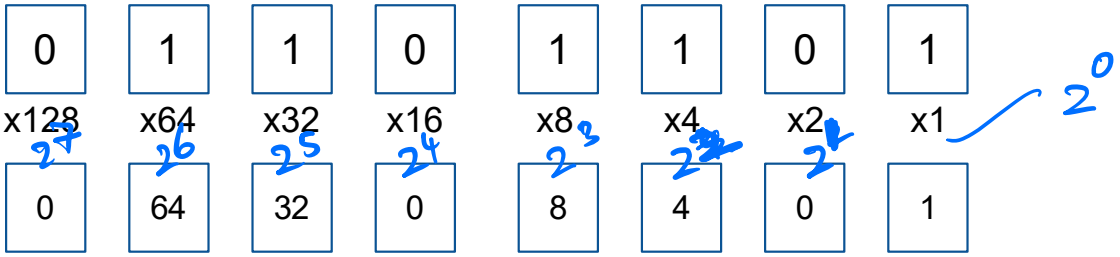
Converting from Binary to Decimal

- 1. Add each binary digit to the boxes below:



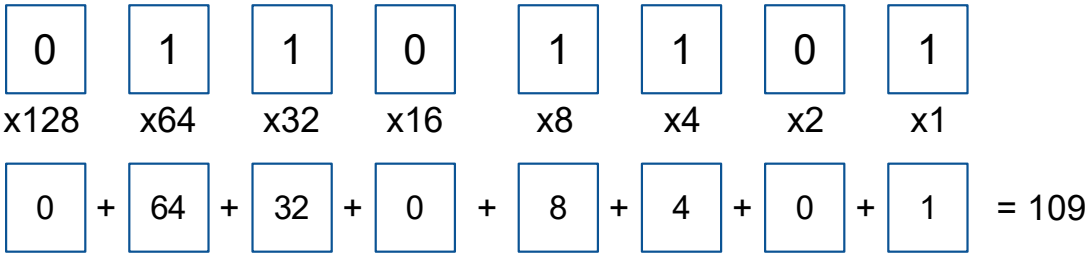
Converting from Binary to Decimal

- 2. Multiply each times the corresponding power of 2:



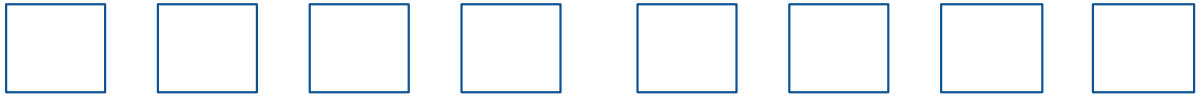
Converting from Binary to Decimal

- 3. Add up the results:



Converting from Decimal to Binary

- Converting from decimal to binary is also easy:

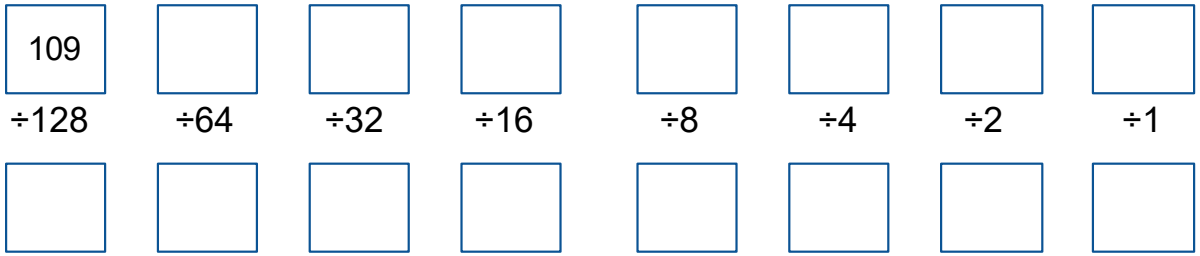


Converting from Decimal to Binary

- 1. Write the number into the left-most box

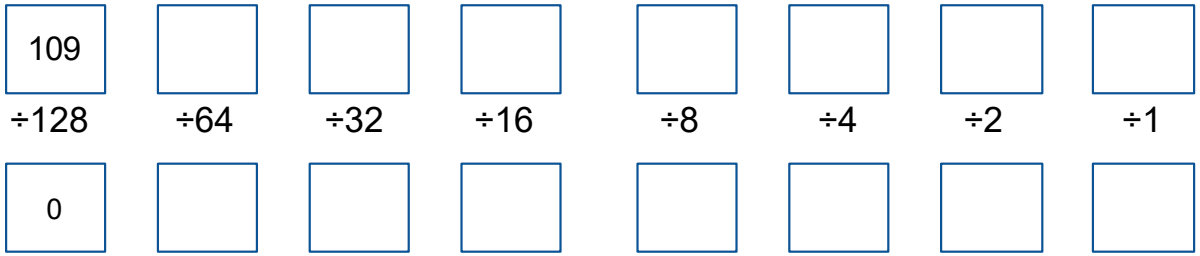
Converting from Decimal to Binary

- 2. Divide by the largest possible power of 2



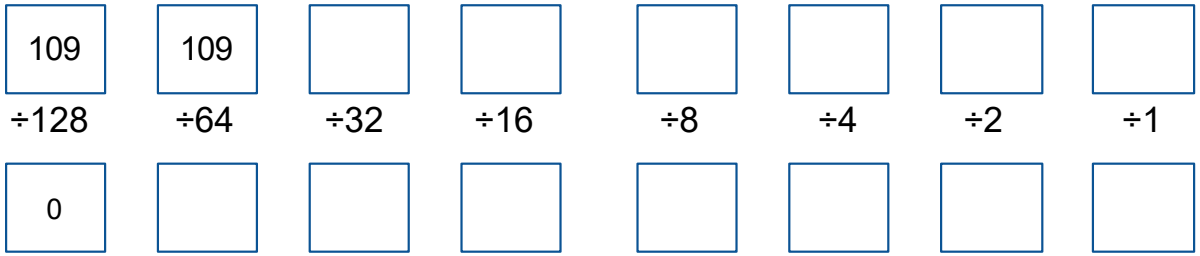
Converting from Decimal to Binary

- 3. Write the (integer) quotient in the box, below



Converting from Decimal to Binary

- 4. Write the remainder in the box to the right



Converting from Decimal to Binary

- 5. Repeat for successively smaller powers of 2

109	109	45					
÷128	÷64	÷32	÷16	÷8	÷4	÷2	÷1
0	1						

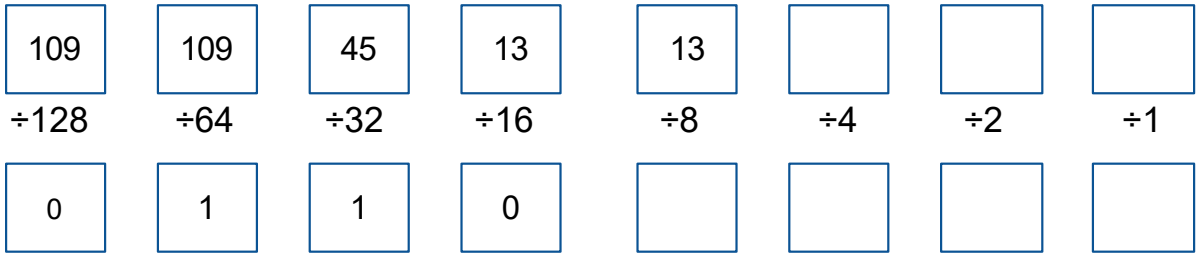
Converting from Decimal to Binary

- 5. Repeat for successively smaller powers of 2

109	109	45	13				
÷128	÷64	÷32	÷16	÷8	÷4	÷2	÷1
0	1	1					

Converting from Decimal to Binary

- 5. Repeat for successively smaller powers of 2



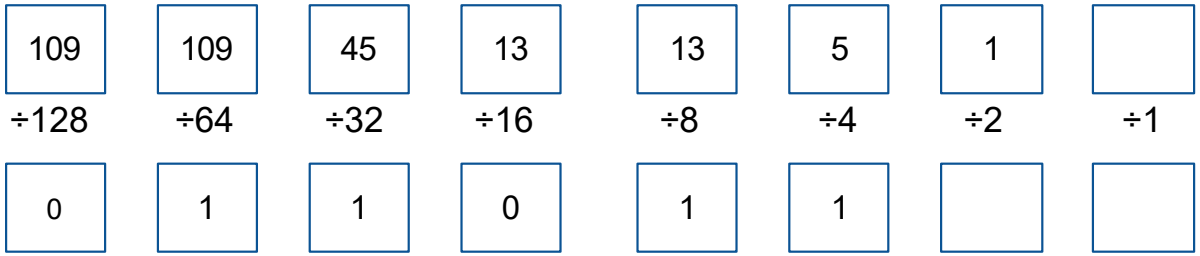
Converting from Decimal to Binary

- 5. Repeat for successively smaller powers of 2

109	109	45	13	13	5		
÷128	÷64	÷32	÷16	÷8	÷4	÷2	÷1
0	1	1	0	1			

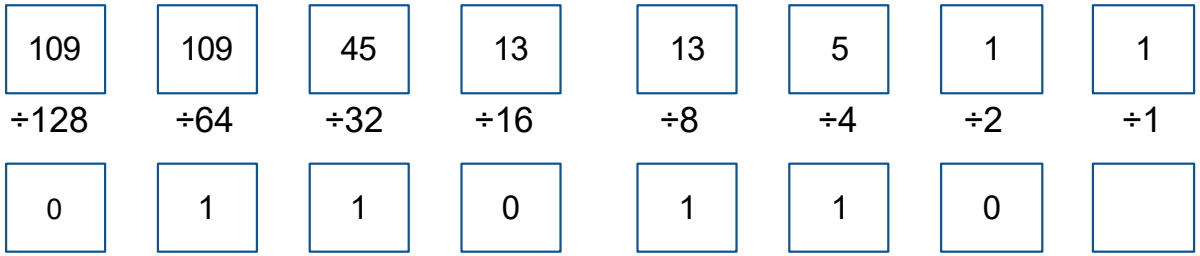
Converting from Decimal to Binary

- 5. Repeat for successively smaller powers of 2



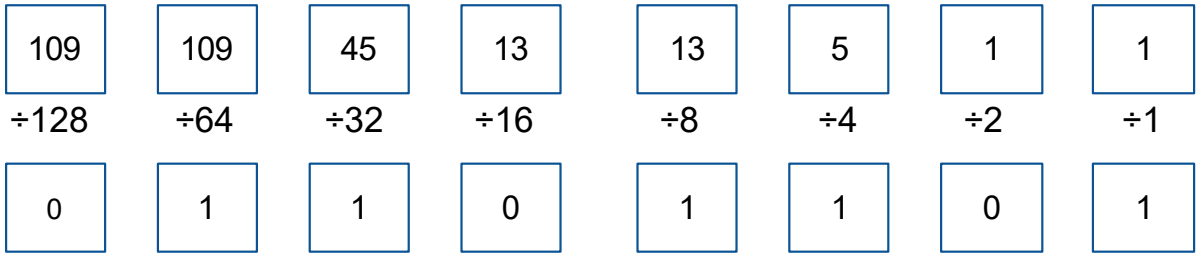
Converting from Decimal to Binary

- 5. Repeat for successively smaller powers of 2



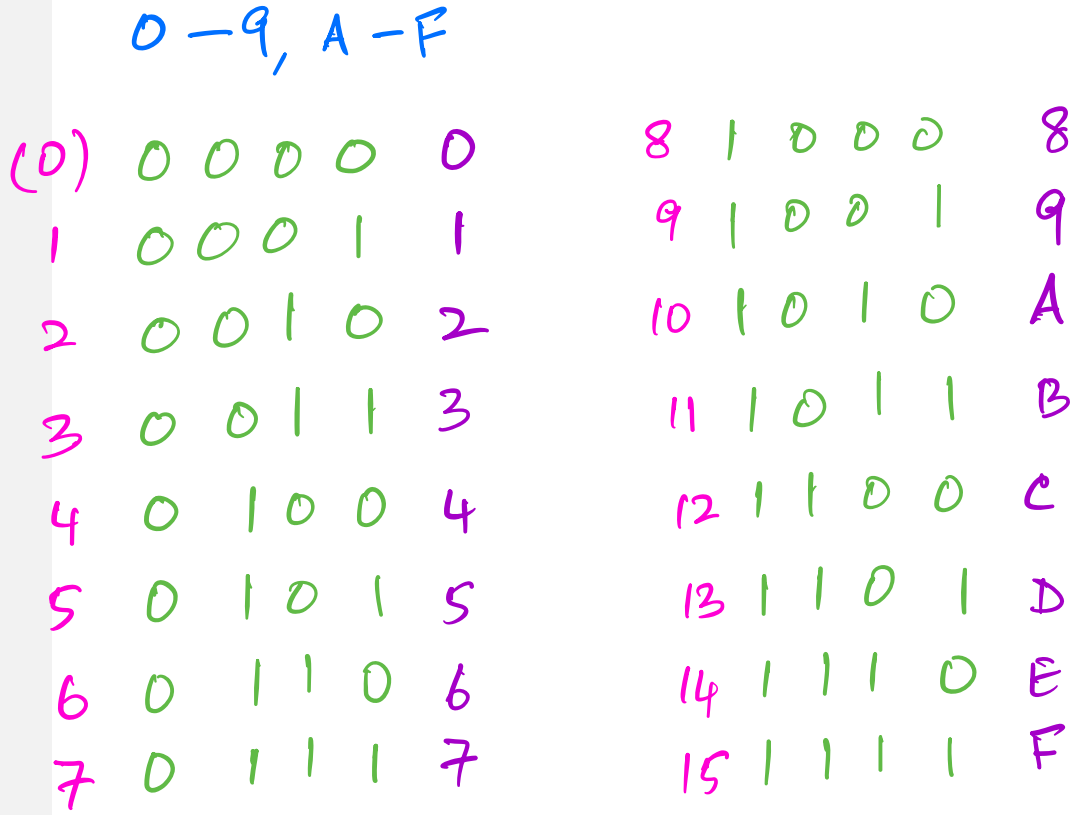
Converting from Decimal to Binary

- 5. Repeat for successively smaller powers of 2



Hexadecimal

CSCI 2050U - Computer Architecture



OntarioTech UNIVERSITY

Octal 0-7

0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

101|010|111|001|001

(iii) Add numbers in base-2

```
 1 1 1
 1 1 0 1   ---> 13
     1 1   ---> 3
-----------
 1 0 0 0 0 ---> 16
```

(iv) What is the biggest or smallest number that you can represent given n bits

e.g. 8

Hexadecimal

- Modern computers only use binary
  - Memory addresses
  - Encrypted or compressed data
  - Integers (signed or unsigned)
  - Floating points
  - Text
- Sometimes, us humans need to look at binary data
  - Binary numbers are too long - it is easy to make mistakes

Hexadecimal

- Modern computers only use binary
  - Memory addresses
  - Encrypted or compressed data
  - Integers (signed or unsigned)
  - Floating points
  - Text
- Sometimes, us humans need to look at binary data
  - Binary numbers are too long - it is easy to make mistakes

```
0110110110010110
1100010110100011
1011011101000110
0001110110100110
```

Hexadecimal

- Modern computers only use binary
  - Memory addresses
  - Encrypted or compressed data
  - Integers (signed or unsigned)
  - Floating points
  - Text
- Sometimes, us humans need to look at binary data
  - Binary numbers are too long - it is easy to make mistakes

```
0110110110010110
1100010110100011
1011011101000110
0001110110100110
```

```
6D96 C5A3 B746 1DA6
```

Hexadecimal

- Hexadecimal is a base-16 number system
- Advantage:
  - As 16 is a power of 2, binary ↔ hexadecimal
  - We can use a table for conversion:
  - There are no unmapped numbers
  - 2 hex digits per byte (8 bits)
- Disadvantage:
  - We don't have 16 digit symbols
  - Hexadecimal uses 0-9, and A-F

Binary	***Hex***
0000	0x0
0001	0x1
0010	0x2
0011	0x3
0100	0x4
0101	0x5
0110	0x6
0111	0x7

Binary	***Hex***
1000	0x8
1001	0x9
1010	0xA
1011	0xB
1100	0xC
1101	0xD
1110	0xE
1111	0xF

Hexadecimal vs. Binary

- Which is easier to read?
  - Hexadecimal: `C07E`
  - Binary: `1100000001111110`

Representing Fractional Numbers

CSCI 2050U - Computer Architecture

132.57

1	3	2	.	5	7
10^2	10^1	10^0		10^{-1}	10^{-2}

1	1	0	1	.	1	0	1
2^3	2^2	2^1	2^0		2^{-1}	2^{-2}	2^{-3}

Representing Fractional Numbers

- Decimal has been extended to include negative powers of 10:

$$109.4075_{10}$$
$$= 1x10^{2} + 0x10^{1} + 9x10^{0}$$
$$+ 4x10^{-1} + 0x10^{-2} + 7x10^{-3} + 5x10^{-4}$$

Representing Fractional Numbers

- Binary can be extended to include negative powers of 2 also:

$$01101101.01101_2$$
$$= 0x2^7 + 1x2^6 + 1x2^5 + 0x2^4$$
$$+ 1x2^3 + 1x2^2 + 0x2^1 + 1x2^0$$
$$+ 0x2^{-1} + 1x2^{-2} + 1x2^{-3} + 0x2^{-4} + 1x2^{-5}$$

Floating Point - Scientific Notation

- In decimal, we often use scientific notation:

$32,413,788.729 = 0.32413788729 \times 10^{8}$

$-0.000005382971 = -0.5382971 \times 10^{-5}$

Floating Point - Binary

- In binary, we could do the same thing:

$$01101101.01101$$

$$= 0.1101101011 01 \times 2^{7}$$

Representing Fractional Numbers

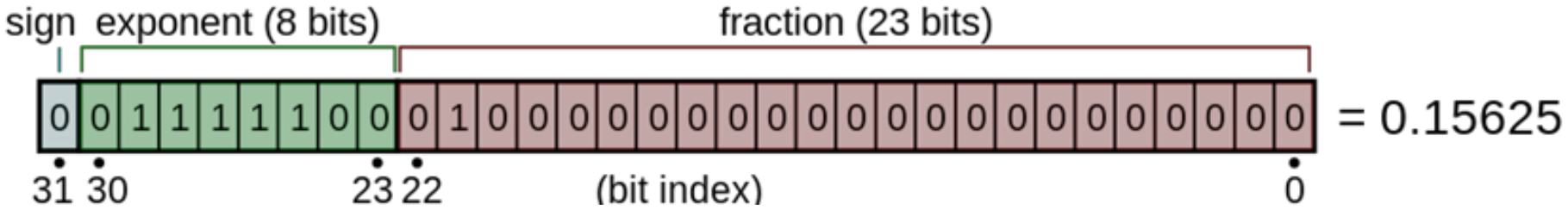
- IEEE 754 defines a set of standards for representing floating point numbers
  - Floating point - similar to scientific notation, but with a limited number of digits
  - 32-bit (single precision)
  - 64-bit (double precision)

23.45
234.5
2.345

https://en.wikipedia.org/wiki/IEEE_754

Representing Fractional Numbers

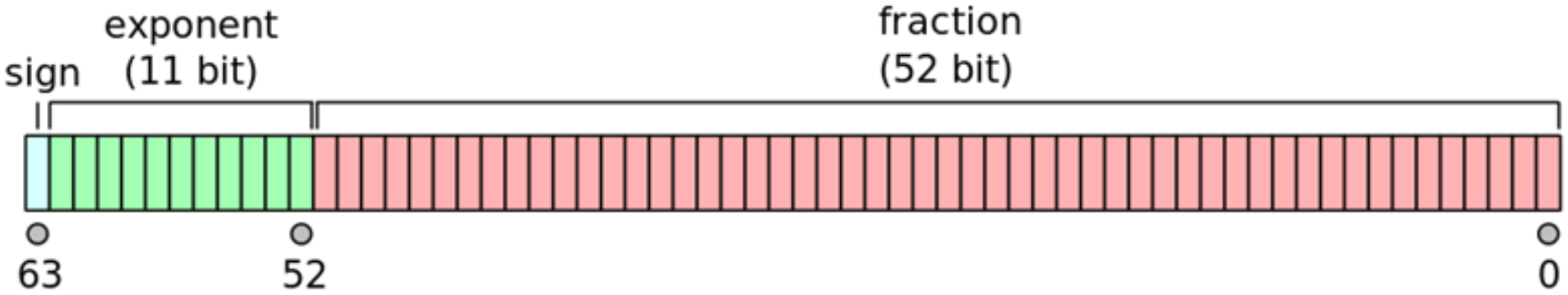
- IEEE 754 single precision format (C++ `float`)
  - Significand - the actual digits
  - Exponent - the power of 2 (i.e. the floating *point*)
  - Sign bit - 1 if negative



https://en.wikipedia.org/wiki/IEEE_754

Representing Fractional Numbers

- IEEE 754 double precision format (C++ `double`)
  - Significand - the actual digits
  - Exponent - the power of 2 (i.e. the floating *point*)
  - Sign bit - 1 if negative



https://en.wikipedia.org/wiki/IEEE_754

Representing Characters

CSCI 2050U - Computer Architecture

Representing Characters - ASCII

- One early encoding for characters is ASCII
- Each letter, digit, punctuation is represented by an 8-bit number
- Advantages:
  - Uppercase letters are contiguous
  - Lowercase letters are contiguous
  - Digits are contiguous
- Disadvantages:
  - Really only English is supported

https://www.cs.cmu.edu/~pattis/15-1XX/common/handouts/ascii.html

Representing Characters - ASCII

Dec	Hex	Bin	Char
0	00	00000000	NUL
1	01	00000001	SOH
2	02	00000010	STX
3	03	00000011	ETX
4	04	00000100	EOT
5	05	00000101	ENQ
6	06	00000110	ACK
7	07	00000111	BEL

https://www.cs.cmu.edu/~pattis/15-1XX/common/handouts/ascii.html

Representing Characters - ASCII

Dec	Hex	Bin	Char
0			
1			
2			
3			
4			
5			
6			
7			

Dec	Hex	Bin	Char
8	08	00001000	BS
9	09	00001001	HT
10	**0A**	**00001010**	**LF**
11	0B	00001011	VT
12	0C	00001100	FF
13	**0D**	**00001101**	**CR**
14	0E	00001110	SO
15	0F	00001111	SI

https://www.cs.cmu.edu/~pattis/15-1XX/common/handouts/ascii.html

Representing Characters - ASCII

Dec	Hex	Bin	Char
0			
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16	10	00010000	DLE
17	11	00010001	DC1
18	12	00010010	DC2
19	13	00010011	DC3
20	14	00010100	DC4
21	15	00010101	NAK
22	16	00010110	SYN
23	17	00010111	ETB

Representing Characters - ASCII

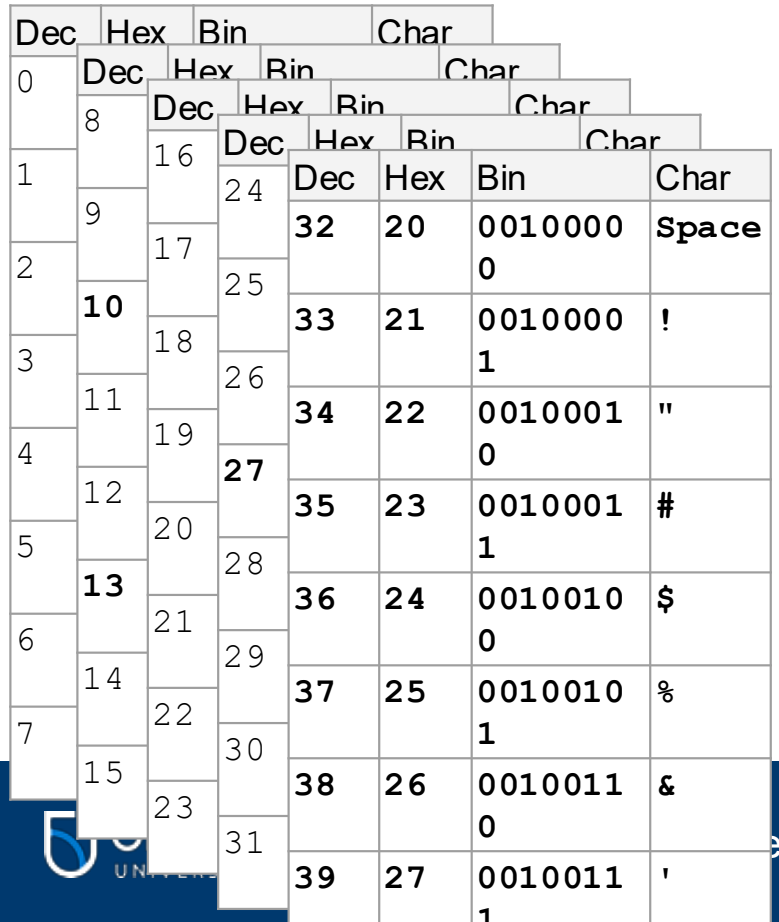
Dec	Hex	Bin	Char
0			
1			
2			
3			
4			
5			
6			
7			

Dec	Hex	Bin	Char
8			
9			
10			
11			
12			
13			
14			
15			

Dec	Hex	Bin	Char
16			
17			
18			
19			
20			
21			
22			
23			

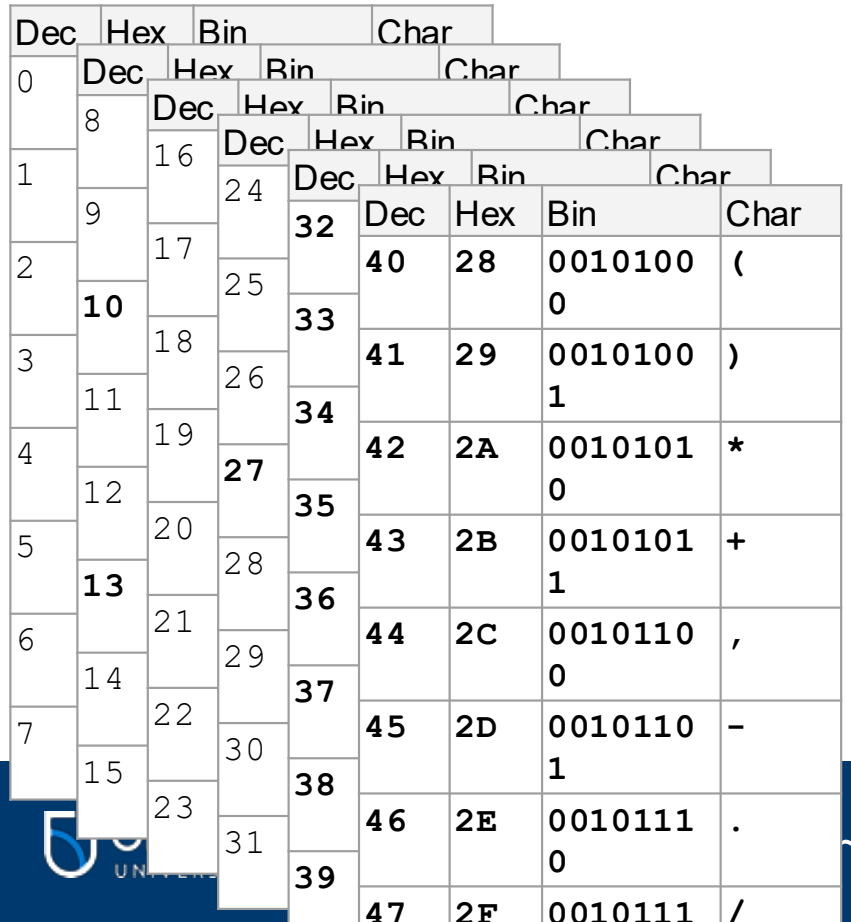
Dec	Hex	Bin	Char
24	18	00011000	CAN
25	19	00011001	EM
26	1A	00011010	SUB
27	**1B**	**00011011**	**ESC**
28	1C	00011100	FS
29	1D	00011101	GS
30	1E	00011110	RS
31	1F	00011111	US

Representing Characters - ASCII



Dec	Hex	Bin	Char
32	20	00100000	Space
33	21	00100001	!
34	22	00100010	"
35	23	00100011	#
36	24	00100100	$
37	25	00100101	%
38	26	00100110	&
39	27	00100111	'

Representing Characters - ASCII



Dec	Hex	Bin	Char
0			
1			
2			
3			
4			
5			
6			
7			

Dec	Hex	Bin	Char
8			
9			
10			
11			
12			
13			
14			
15			

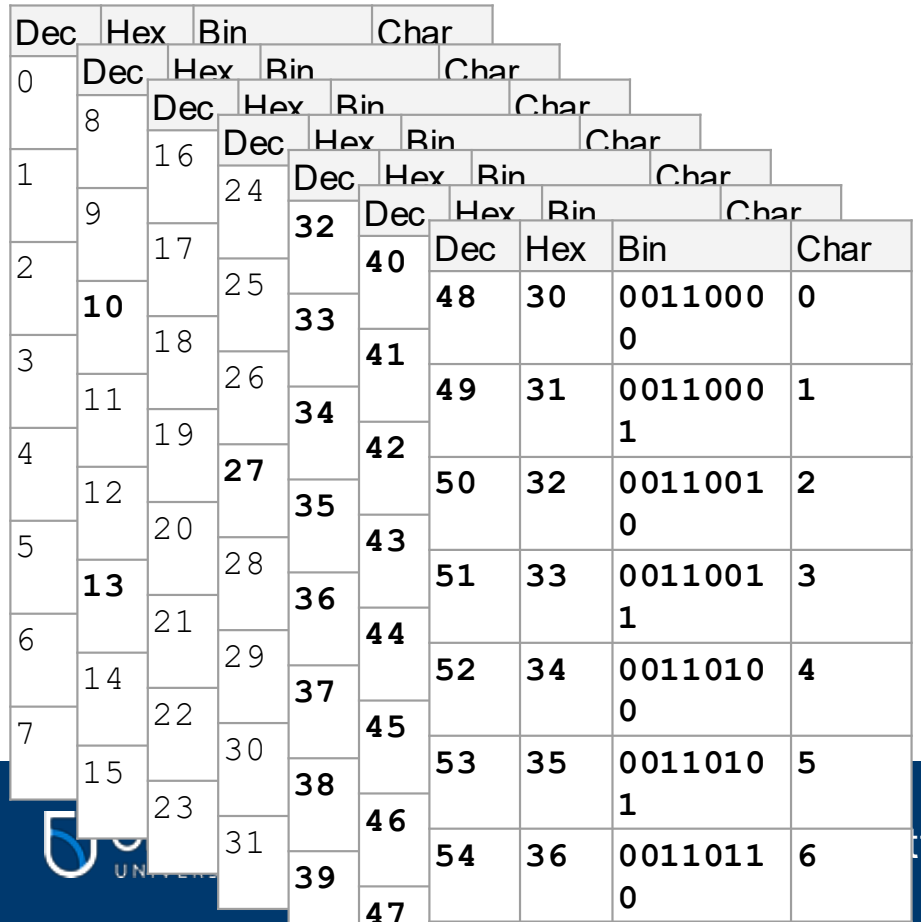
Dec	Hex	Bin	Char
16			
17			
18			
19			
20			
21			
22			
23			

Dec	Hex	Bin	Char
24			
25			
26			
27			
28			
29			
30			
31			

Dec	Hex	Bin	Char
32			
33			
34			
35			
36			
37			
38			
39			

Dec	Hex	Bin	Char
40	28	00101000	(
41	29	00101001	)
42	2A	00101010	*
43	2B	00101011	+
44	2C	00101100	,
45	2D	00101101	-
46	2E	00101110	.
47	2F	00101111	/

Representing Characters - ASCII



Dec	Hex	Bin	Char
0			
1			
2			
3			
4			
5			
6			
7			

Dec	Hex	Bin	Char
8			
9			
10			
11			
12			
13			
14			
15			

Dec	Hex	Bin	Char
16			
17			
18			
19			
20			
21			
22			
23			

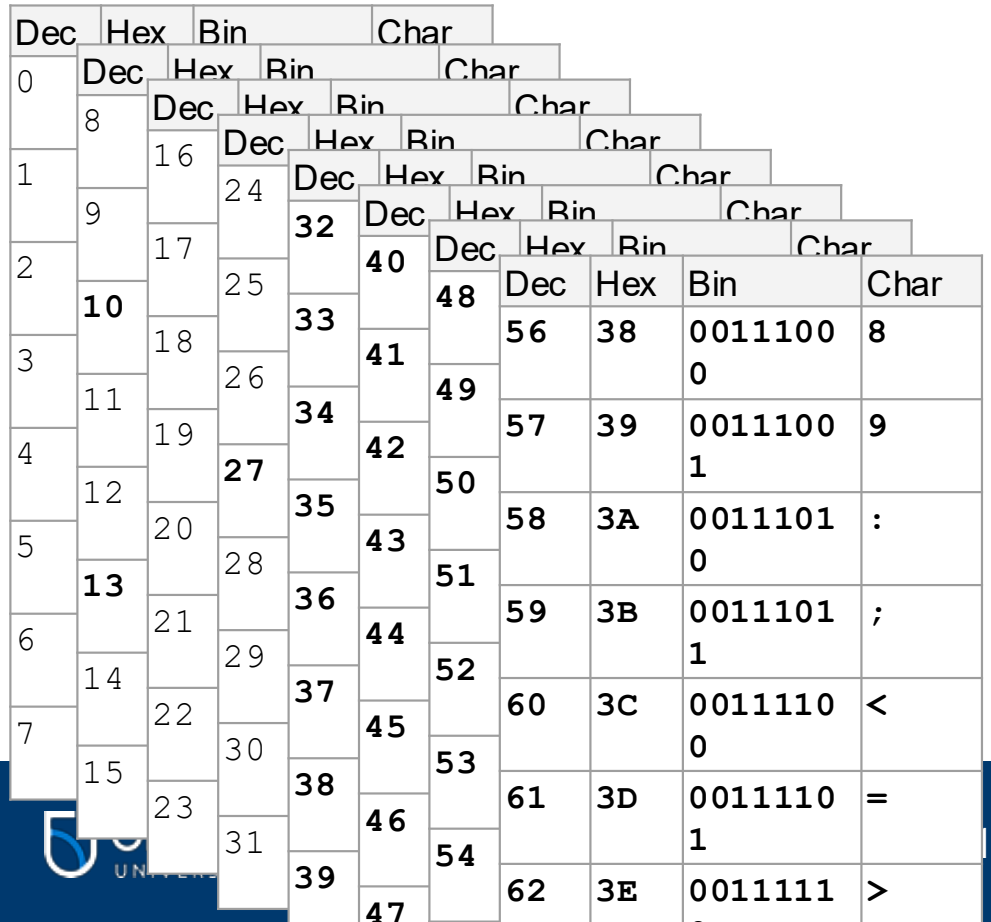
Dec	Hex	Bin	Char
24			
25			
26			
27			
28			
29			
30			
31			

Dec	Hex	Bin	Char
32			
33			
34			
35			
36			
37			
38			
39			

Dec	Hex	Bin	Char
40			
41			
42			
43			
44			
45			
46			
47			

Dec	Hex	Bin	Char
48	30	00110000	0
49	31	00110001	1
50	32	00110010	2
51	33	00110011	3
52	34	00110100	4
53	35	00110101	5
54	36	00110110	6

Representing Characters - ASCII



Dec	Hex	Bin	Char
56	38	00111000	8
57	39	00111001	9
58	3A	00111010	:
59	3B	00111011	;
60	3C	00111100	<
61	3D	00111101	=
62	3E	00111111	>

Representing Characters - ASCII

Dec	Hex	Bin	Char
0			
1			
2			
3			
4			
5			
6			
7			

Dec	Hex	Bin	Char
8			
9			
10			
11			
12			
13			
14			
15			

Dec	Hex	Bin	Char
16			
17			
18			
19			
20			
21			
22			
23			

Dec	Hex	Bin	Char
24			
25			
26			
27			
28			
29			
30			
31			

Dec	Hex	Bin	Char
32			
33			
34			
35			
36			
37			
38			
39			

Dec	Hex	Bin	Char
40			
41			
42			
43			
44			
45			
46			
47			

Dec	Hex	Bin	Char
48			
49			
50			
51			
52			
53			
54			

Dec	Hex	Bin	Char
56	38	00111000	8
57	39	00111001	9
58	3A	00111010	:
59	3B	00111011	;
60	3C	00111100	<
61	3D	00111101	=
62	3E	00111110	>
63	3F	00111111	?

Dec	Hex	Bin	Char
64	40	01000000	@
65	41	01000001	A
66	42	01000010	B
67	43	01000011	C
68	44	01000100	D
69	45	01000101	E
70	46	01000110	F
71	47	01000111	G

Representing Characters - ASCII

Dec	Hex	Bin	Char
56	38	00111000	8
57	39	00111001	9
58	3A	00111010	:
59	3B	00111011	;
60	3C	00111100	<
61	3D	00111101	=
62	3E	00111110	>

Dec	Hex	Bin	Char
72	48	01001000	H
73	49	01001001	I
74	4A	01001010	J
75	4B	01001011	K
76	4C	01001100	L
77	4D	01001101	M
78	4E	01001110	N
79	4F	01001111	O

Representing Characters - ASCII

Dec	Hex	Bin	Char
56	38	00111000	8
57	39	00111001	9
58	3A	00111010	:
59	3B	00111011	;
60	3C	00111100	<
61	3D	00111101	=
62	3E	00111110	>

Dec	Hex	Bin	Char
80	50	01010000	P
81	51	01010001	Q
82	52	01010010	R
83	53	01010011	S
84	54	01010100	T
85	55	01010101	U
86	56	01010110	V
87	57	01010111	W

Representing Characters - ASCII

Dec	Hex	Bin	Char
56	38	00111000	8
57	39	00111001	9
58	3A	00111010	:
59	3B	00111011	;
60	3C	00111100	<
61	3D	00111101	=
62	3E	00111110	>

Dec	Hex	Bin	Char
88	58	01011000	X
89	59	01011001	Y
90	5A	01011010	Z
91	5B	01011011	[
92	5C	01011100	\
93	5D	01011101	]
94	5E	01011110	^
95	5F	01011111	_

Representing Characters - ASCII

Dec	Hex	Bin	Char
56	38	00111000	8
57	39	00111001	9
58	3A	00111010	:
59	3B	00111011	;
60	3C	00111100	<
61	3D	00111101	=
62	3E	00111110	>
63	3F	00111111	?
64	40	01000000	@
65	41	01000001	A
66	42	01000010	B
67	43	01000011	C
68	44	01000100	D
69	45	01000101	E
70	46	01000110	F
71	47	01000111	G
72	48	01001000	H
73	49	01001001	I
74	50	01001010	J
75	51	01001011	K
76	52	01001100	L
77	53	01001101	M
78	54	01001110	N
79	55	01001111	O
80	56	01010000	P
81	57	01010001	Q
82	58	01010010	R
83	59	01010011	S
84	5A	01010100	T
85	5B	01010101	U
86	5C	01010110	V
87	5D	01010111	W
88	5E	01011000	X
89	5F	01011001	Y
90	60	01011010	Z
91	61	01011011	[
92	62	01011100	\
93	63	01011101	]
94	64	01011110	^
95	65	01011111	_
96	60	01100000	`
97	61	01100001	a
98	62	01100010	b
99	63	01100011	c
100	64	01100100	d
101	65	01100101	e
102	66	01100110	f
103	67	01100111	g

Representing Characters - ASCII

Dec	Hex	Bin	Char
56	38	00111000	8
57	39	00111001	9
58	3A	00111010	:
59	3B	00111011	;
60	3C	00111100	<
61	3D	00111101	=
62	3E	00111110	>

Dec	Hex	Bin	Char
104	68	01101000	h
105	69	01101001	i
106	6A	01101010	j
107	6B	01101011	k
108	6C	01101100	l
109	6D	01101101	m
110	6E	01101110	n
111	6F	01101111	o

Representing Characters - ASCII

Dec	Hex	Bin	Char
56	38	00111000	8
57	39	00111001	9
58	3A	00111010	:
59	3B	00111011	;
60	3C	00111100	<
61	3D	00111101	=
62	3E	00111110	>
63	3F	00111111	?
64	40	01000000	@
65	41	01000001	A
66	42	01000010	B
67	43	01000011	C
68	44	01000100	D
69	45	01000101	E
70	46	01000110	F
71	47	01000111	G
72	48	01001000	H
73	49	01001001	I
74	50	01001010	J
75	51	01001011	K
76	52	01001100	L
77	53	01001101	M
78	54	01001110	N
79	55	01001111	O
80	56	01010000	P
81	57	01010001	Q
82	58	01010010	R
83	59	01010011	S
84	60	01010100	T
85	61	01010101	U
86	62	01010110	V
87	63	01010111	W
88	64	01011000	X
89	65	01011001	Y
90	66	01011010	Z
91	67	01011011	[
92	68	01011100	\
93	69	01011101	]
94	70	01011110	^
95	71	01011111	_
96	72	01100000	`
97	73	01100001	a
98	74	01100010	b
99	75	01100011	c
100	76	01100100	d
101	77	01100101	e
102	78	01100110	f
103	79	01100111	g
104	80	01101000	h
105	81	01101001	i
106	82	01101010	j
107	83	01101011	k
108	84	01101100	l
109	85	01101101	m
110	86	01101110	n
111	87	01101111	o
112	70	01110000	p
113	71	01110001	q
114	72	01110010	r
115	73	01110011	s
116	74	01110100	t
117	75	01110101	u
118	76	01110110	v

Representing Characters - ASCII

Dec	Hex	Bin	Char
56	38	00111000	8
57	39	00111001	9
58	3A	00111010	:
59	3B	00111011	;
60	3C	00111100	<
61	3D	00111101	=
62	3E	00111111	>

Dec	Hex	Bin	Char
120	78	01111000	x
121	79	01111001	y
122	7A	01111010	z
123	7B	01111011	{
124	7C	01111100	|
125	7D	01111101	}
126	7E	01111111	~

Representing Characters - Other Encodings

- Unicode
  - Supports most writing systems, math symbols, emojis
  - Can store these characters in one of two ways:
    - UTF-8: Stores unicode characters using 8-32 characters
      - English (8-bit) and latin (16-bit) characters are prioritized
    - UTF-16: Stores unicode characters using 16-32 characters

https://home.unicode.org/

Wrap-up

- Binary
- Hexadecimal
- Representing fractional numbers
- Representing characters

What is next?

- Error detection and correction
- Prefix codes
- Huffman's algorithm