Fundamentals of Computer Systems

Summer 2024

Columbia University

The subjects of this class

0 **1**

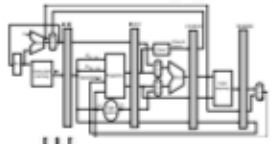
Computer Systems Work Because of Abstraction

Application Software

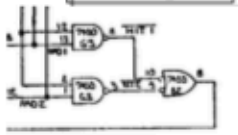
Operating Systems

```
  ;; voice 1 wave select
ld      a,(#CH1_W_NUM)
and     a
ld      a,(#CH1_W_SEL)
jr      nz,#00b4
ld      a,(#CH1_E_TABLE0)
```

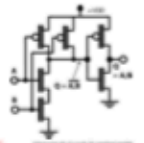
Architecture



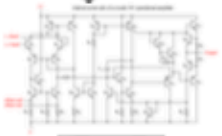
Micro-Architecture



Logic

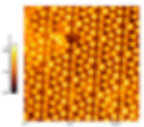


Digital Circuits



Analog Circuits

Devices

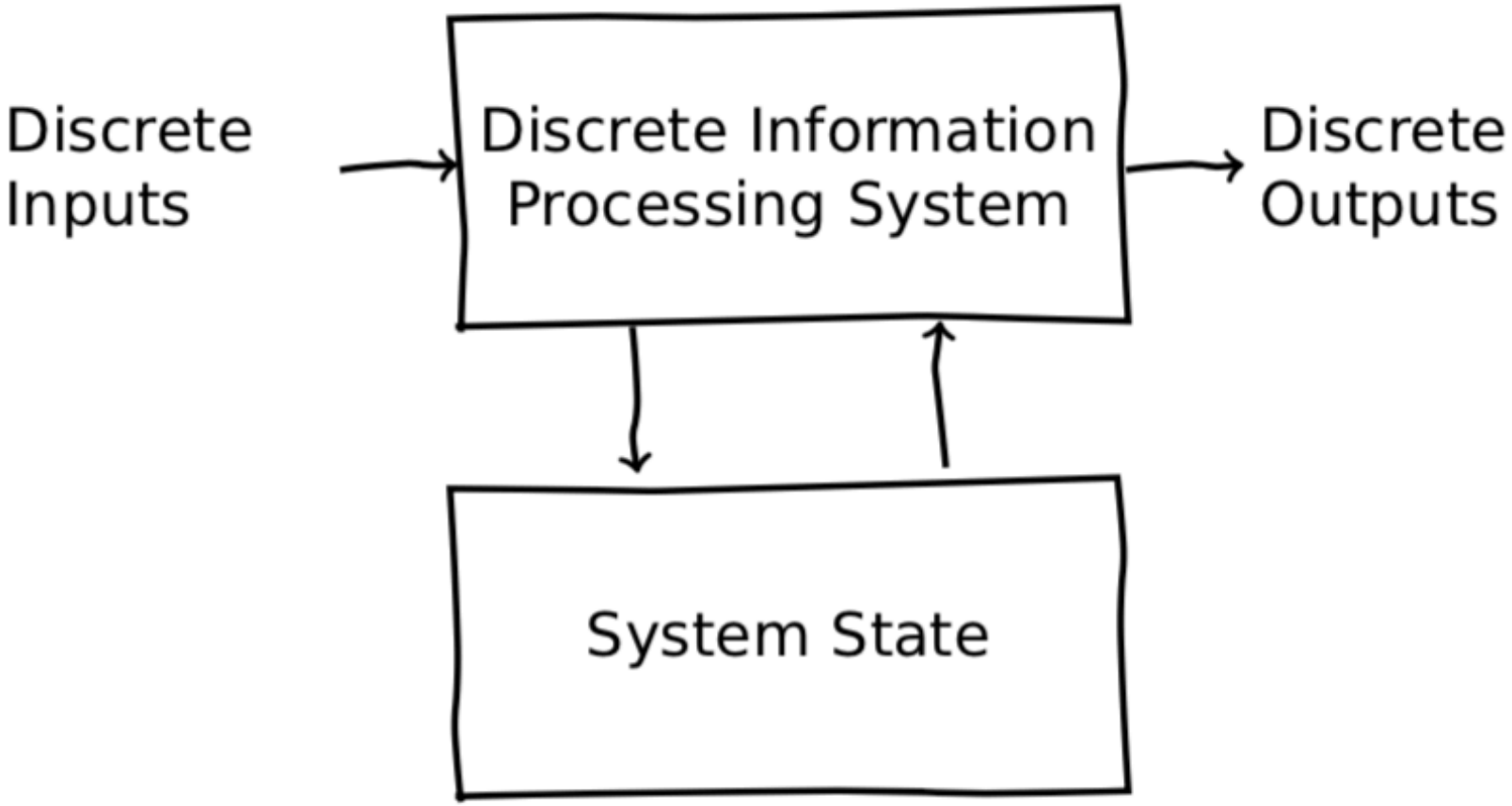


Physics

Computer Systems Work Because of Abstraction

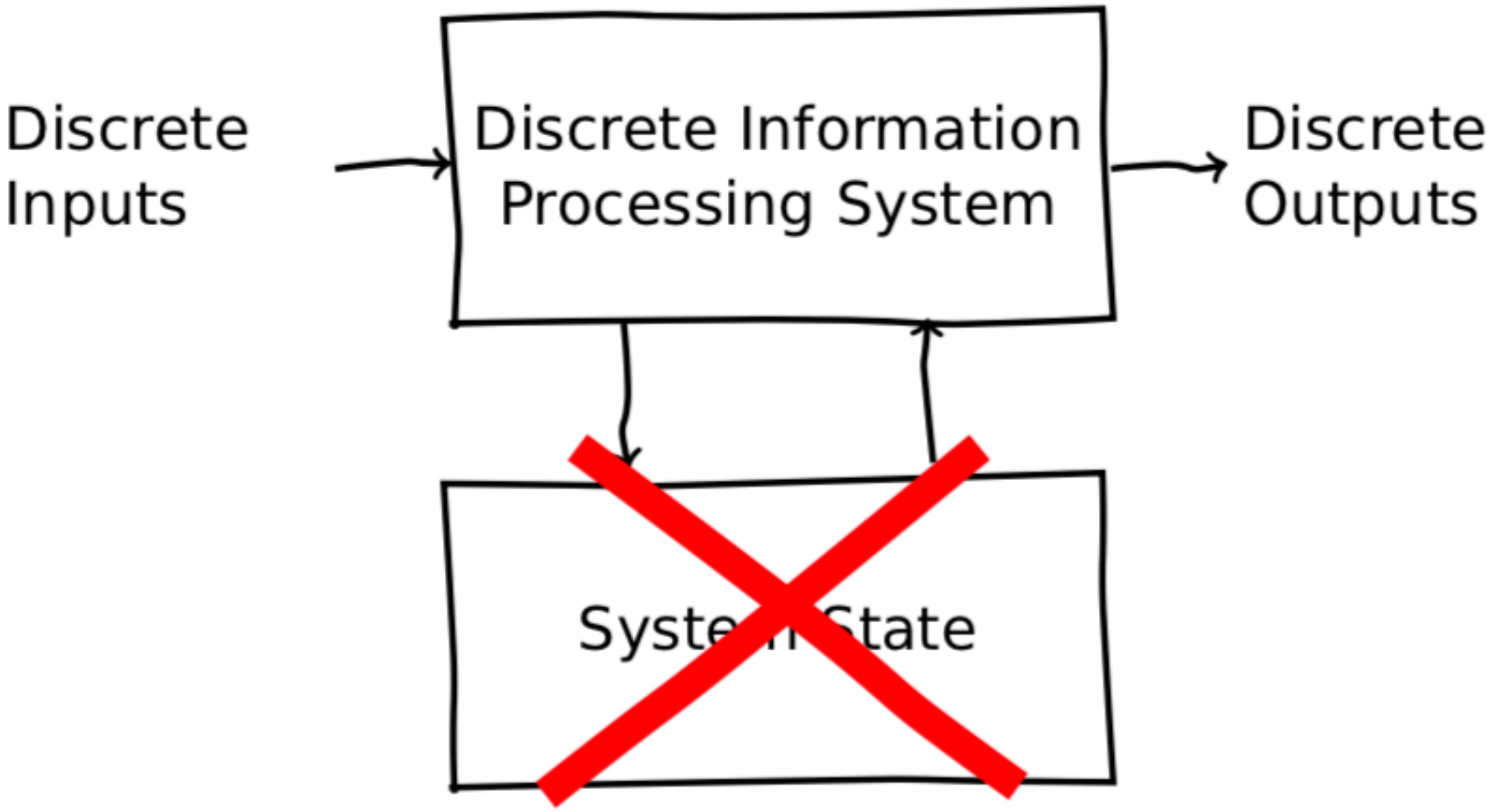
Application Software	COMS 3157, 4156, et al.
Operating Systems	COMS W4118
Architecture	Second Half of 3827
Micro-Architecture	Second Half of 3827
Logic	First Half of 3827
Digital Circuits	First Half of 3827
Analog Circuits	ELEN 3331
Devices	ELEN 3106
Physics	ELEN 3106 et al.

Information Processing System



First half of the course

Information Processing System



First quarter of the course

Number Systems

Before we can begin to understand simple computer systems, we must first understand Binary, the foundation of these systems

Binary Number Systems

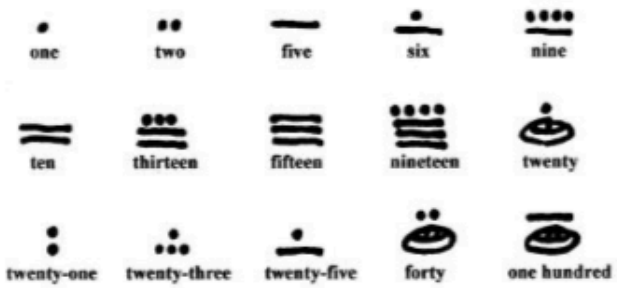
- All modern digital devices rely on a simple scheme, intimately tied to transistors:
  - 1 - On
  - 0 - Off

The Decimal Number System

- Ten figures: 0 1 2 3 4 5 6 7 8 9
- $1{,}354 = 1 \times 10^3 + 3 \times 10^2 + 5 \times 10^1 + 4 \times 10^0$
- Why base 10?

Other Number Systems

Roman: I II III IV V VI VII VIII IX X



Mayan: base 20, Shell = 0

Babylonian: base 60

Hexadecimal, Decimal, Octal, and Binary

Hex	Dec	Oct	Bin
0	0	0	0
1	1	1	1
2	2	2	10
3	3	3	11
4	4	4	100
5	5	5	101
6	6	6	110
7	7	7	111
8	8	10	1000
9	9	11	1001
A	10	12	1010
B	11	13	1011
C	12	14	1100
D	13	15	1101
E	14	16	1110
F	15	17	1111

Binary and Octal

- Binary: 2 figures: 0 1
- $11111010 = 1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$
- Octal: 8 figures: 0 1 2 3 4 5 6 7
- 011=3, 111=7, 010=2
- $= 3 \times 8^2 + 7 \times 8^1 + 2 \times 8^0$
- $= 2 \times 10^2 + 5 \times 10^1 + 0 \times 10^0$

Binary and Octal

- Walkthrough on board

Hexadecimal

Base 16: 0 1 2 3 4 5 6 7 8 9 A B C D E F

Instead of groups of 3 bits (octal), Hex uses groups of 4.

$$\begin{aligned} \text{CAFEF00D}_{16} &= 12 \times 16^7 + 10 \times 16^6 + 15 \times 16^5 + 14 \times 16^4 + \\ &\quad 15 \times 16^3 + 0 \times 16^2 + 0 \times 16^1 + 13 \times 16^0 \\ &= 3,405,705,229_{10} \end{aligned}$$

C	A	F	E	F	0	0	D	Hex
1100	1010	1111	1110	1111	0000	0001	1101	Binary
3	1	2	7	7	5	7	0	0	1	5	Octal

Computers Rarely Manipulate True Numbers

Infinite memory still very expensive

Finite-precision numbers typical

32-bit processor: naturally manipulates 32-bit numbers

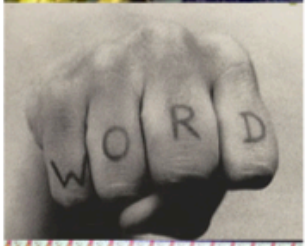
64-bit processor: naturally manipulates 64-bit numbers

How many different numbers can you represent with 5 binary / octal / decimal / hexadecimal digits?

Jargon

Bit — Binary digit: 0 or 1

Byte — Eight bits



Word — Natural number of bits for the processor, e.g., 16, 32, 64

LSB — Least Significant Bit (“rightmost”)

MSB — Most Significant Bit (“leftmost”)

Decimal Addition Algorithm

$$\begin{array}{r} 434 \\ +628 \\ \hline \end{array}$$

$4 + 8 = 12$

+	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	10
2	2	3	4	5	6	7	8	9	10	11
3	3	4	5	6	7	8	9	10	11	12
4	4	5	6	7	8	9	10	11	12	13
5	5	6	7	8	9	10	11	12	13	14
6	6	7	8	9	10	11	12	13	14	15
7	7	8	9	10	11	12	13	14	15	16
8	8	9	10	11	12	13	14	15	16	17
9	9	10	11	12	13	14	15	16	17	18
10	10	11	12	13	14	15	16	17	18	19

Decimal Addition Algorithm

$$\begin{array}{r} {\scriptstyle 1} \\ 434 \\ +628 \\ \hline 2 \end{array}$$

$$4 + 8 = 12$$

$$1 + 3 + 2 = 6$$

+	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	10
2	2	3	4	5	6	7	8	9	10	11
3	3	4	5	6	7	8	9	10	11	12
4	4	5	6	7	8	9	10	11	12	13
5	5	6	7	8	9	10	11	12	13	14
6	6	7	8	9	10	11	12	13	14	15
7	7	8	9	10	11	12	13	14	15	16
8	8	9	10	11	12	13	14	15	16	17
9	9	10	11	12	13	14	15	16	17	18
10	10	11	12	13	14	15	16	17	18	19

Decimal Addition Algorithm

$$\begin{array}{r} {\scriptstyle 1} \\ 434 \\ +628 \\ \hline 62 \end{array}$$

$$\begin{aligned} 4+8 &= 12 \\ 1+3+2 &= 6 \\ 4+6 &= 10 \end{aligned}$$

+	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	10
2	2	3	4	5	6	7	8	9	10	11
3	3	4	5	6	7	8	9	10	11	12
4	4	5	6	7	8	9	10	11	12	13
5	5	6	7	8	9	10	11	12	13	14
6	6	7	8	9	10	11	12	13	14	15
7	7	8	9	10	11	12	13	14	15	16
8	8	9	10	11	12	13	14	15	16	17
9	9	10	11	12	13	14	15	16	17	18
10	10	11	12	13	14	15	16	17	18	19

Decimal Addition Algorithm

$$\begin{array}{r} {\scriptstyle 1\ \ 1\ \ \ } \\ 434 \\ +628 \\ \hline 062 \end{array}$$

$$\begin{aligned} 4+8 &= 12 \\ 1+3+2 &= 6 \\ 4+6 &= 10 \end{aligned}$$

+	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	10
2	2	3	4	5	6	7	8	9	10	11
3	3	4	5	6	7	8	9	10	11	12
4	4	5	6	7	8	9	10	11	12	13
5	5	6	7	8	9	10	11	12	13	14
6	6	7	8	9	10	11	12	13	14	15
7	7	8	9	10	11	12	13	14	15	16
8	8	9	10	11	12	13	14	15	16	17
9	9	10	11	12	13	14	15	16	17	18
10	10	11	12	13	14	15	16	17	18	19

Decimal Addition Algorithm

$$\begin{array}{r} {\scriptstyle 1\ \ 1\ \ \ } \\ 434 \\ +628 \\ \hline 1062 \end{array}$$

$$\begin{aligned} 4+8 &= 12 \\ 1+3+2 &= 6 \\ 4+6 &= 10 \end{aligned}$$

+	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	10
2	2	3	4	5	6	7	8	9	10	11
3	3	4	5	6	7	8	9	10	11	12
4	4	5	6	7	8	9	10	11	12	13
5	5	6	7	8	9	10	11	12	13	14
6	6	7	8	9	10	11	12	13	14	15
7	7	8	9	10	11	12	13	14	15	16
8	8	9	10	11	12	13	14	15	16	17
9	9	10	11	12	13	14	15	16	17	18
10	10	11	12	13	14	15	16	17	18	19

Binary Addition Algorithm

$$\begin{array}{r} 10011 \\ +11001 \\ \hline \end{array}$$

$1 + 1 = 10$

+	0	1
0	00	01
1	01	10
10	10	11

Binary Addition Algorithm

$$\begin{array}{r} 1 \\ 10011 \\ +11001 \\ \hline 0 \end{array}$$

$$1 + 1 = 10$$

$$1 + 1 + 0 = 10$$

+	0	1
0	00	01
1	01	10
10	10	11

Binary Addition Algorithm

$$
\begin{array}{r}
 11 \\
 10011 \\
+11001 \\
\hline
 00
\end{array}
$$

$$
\begin{aligned}
1 + 1 &= 10 \\
1 + 1 + 0 &= 10 \\
1 + 0 + 0 &= 01
\end{aligned}
$$

+	0	1
0	00	01
1	01	10
10	10	11

Binary Addition Algorithm

$$
\begin{array}{r}
011 \\
10011 \\
+11001 \\
\hline
100
\end{array}
$$

$$
\begin{aligned}
1+1 &= 10 \\
1+1+0 &= 10 \\
1+0+0 &= 01 \\
0+0+1 &= 01
\end{aligned}
$$

+	0	1
0	00	01
1	01	10
10	10	11

Binary Addition Algorithm

$$
\begin{array}{r}
0011 \\
10011 \\
+11001 \\
\hline
1100
\end{array}
$$

$$
\begin{aligned}
1+1 &= 10 \\
1+1+0 &= 10 \\
1+0+0 &= 01 \\
0+0+1 &= 01 \\
0+1+1 &= 10
\end{aligned}
$$

+	0	1
0	00	01
1	01	10
10	10	11

Binary Addition Algorithm

$$
\begin{array}{r}
10011 \\
10011 \\
+11001 \\
\hline
101100
\end{array}
$$

$$
\begin{aligned}
1+1 &= 10 \\
1+1+0 &= 10 \\
1+0+0 &= 01 \\
0+0+1 &= 01 \\
0+1+1 &= 10
\end{aligned}
$$

+	0	1
0	00	01
1	01	10
10	10	11

Representing Negative Numbers

- How do we represent both positive and negative numbers?
- In decimal, we use the (-) sign

Representing Negative Numbers - Signed Binary

You are most familiar with this:
negative numbers have a leading −

In binary, a leading 1 means negative:

$0000_2 = 0$

$0010_2 = 2$

$1010_2 = -2$

$1111_2 = -7$

$1000_2 = -0?$

Can be made to work, but addition is annoying:

If the signs match, add the magnitudes and use the same sign.

If the signs differ, subtract the smaller number from the larger; return the sign of the larger.

Representing Negative Numbers - One's Complement

Like Signed Magnitude, a leading 1 indicates a negative One's Complement number.

To negate a number, complement (flip) each bit.

$0000_2 = 0$

$0010_2 = 2$

$1101_2 = -2$

$1000_2 = -7$

$1111_2 = -0?$

Addition is nicer: just add the one's complement numbers as if they were normal binary.

Really annoying having a -0: two numbers are equal if their bits are the same or if one is 0 and the other is -0.

Representing Negative Numbers

Fix the double zero issue?

Representing Negative Numbers

Two's Complement

Representing Negative Numbers - Two's Complement

Really neat trick: make the most significant bit represent a *negative* number instead of positive:

$$1101_2 = -8 + 4 + 1 = -3$$

$$1111_2 = -8 + 4 + 2 + 1 = -1$$

$$0111_2 = 4 + 2 + 1 = 7$$

$$1000_2 = -8$$

Representing Negative Numbers - Two's Complement

Easy addition: just add in binary and discard any carry.

Negation: complement each bit (as in one's complement) then add 1.

Very good property: no −0

Two's complement numbers are equal if all their bits are the same.

Representing Negative Numbers - Two's Complement

Bits	Binary	Signed Mag.	One's Comp.	Two's Comp.
0000	0	0	0	0
0001	1	1	1	1
⋮				
0111	7	7	7	7
1000	8	−0	−7	−8
1001	9	−1	−6	−7
⋮				
1110	14	−6	−1	−2
1111	15	−7	−0	−1

Smallest number

Largest number

Representing Negative Numbers - Two's Complement

Examples on the board

Fixed Point vs Floating Point

How do we represent fractional numbers?

Fixed Point

How to represent fractional numbers? In decimal, we continue with negative powers of 10:

$$\begin{aligned} 31.4159 &= 3 \times 10^1 + 1 \times 10^0 + \\ &\quad 4 \times 10^{-1} + 1 \times 10^{-2} + 5 \times 10^{-3} + 9 \times 10^{-4} \end{aligned}$$

The same trick works in binary:

$$\begin{aligned} 1011.0110_2 &= 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + \\ &\quad 0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} + 0 \times 2^{-4} \\ &= 8 + 2 + 1 + 0.25 + 0.125 \\ &= 11.375 \end{aligned}$$

Floating Point

Floating point can represent very large numbers in a compact way.

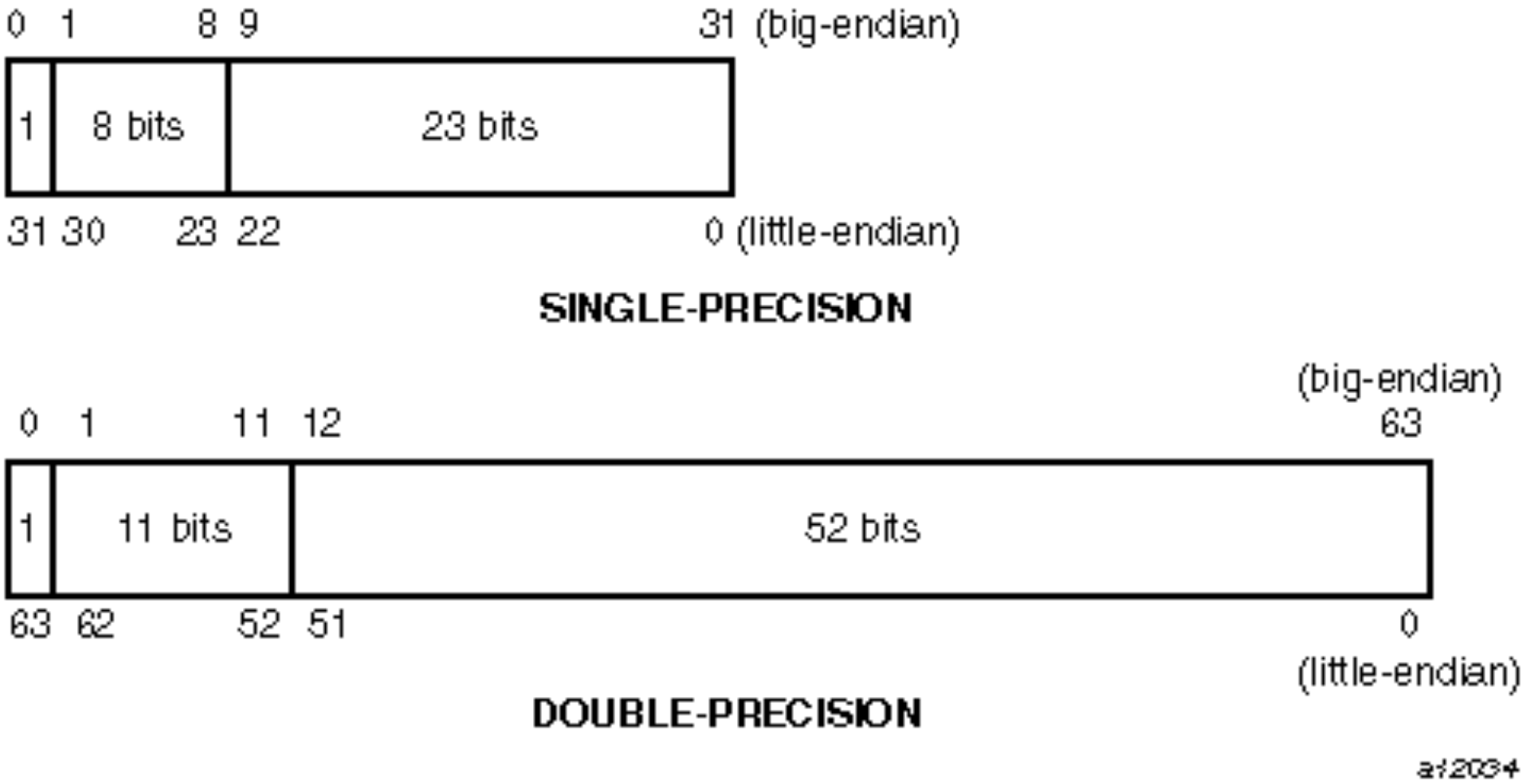
A lot like scientific notation, -7.776×10^3, where you have the *mantissa* (-7.776) and *exponent* (3).

But for this course, think in binary: $-1.10x2^{0111}$

The bits of a 32-bit word are separated into fields. The IEEE 754 standard specifies

- which bits represent which fields (bit 31 is sign, bits 30-23 are 8-bit exponent, bits 22-00 are 23-bit fraction)
- how to interpret each field

Floating Point



Characters and Strings

How do we represent characters and letters?

Characters and Strings

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

Source: www.LookupTables.com