Uvod v računalništvo

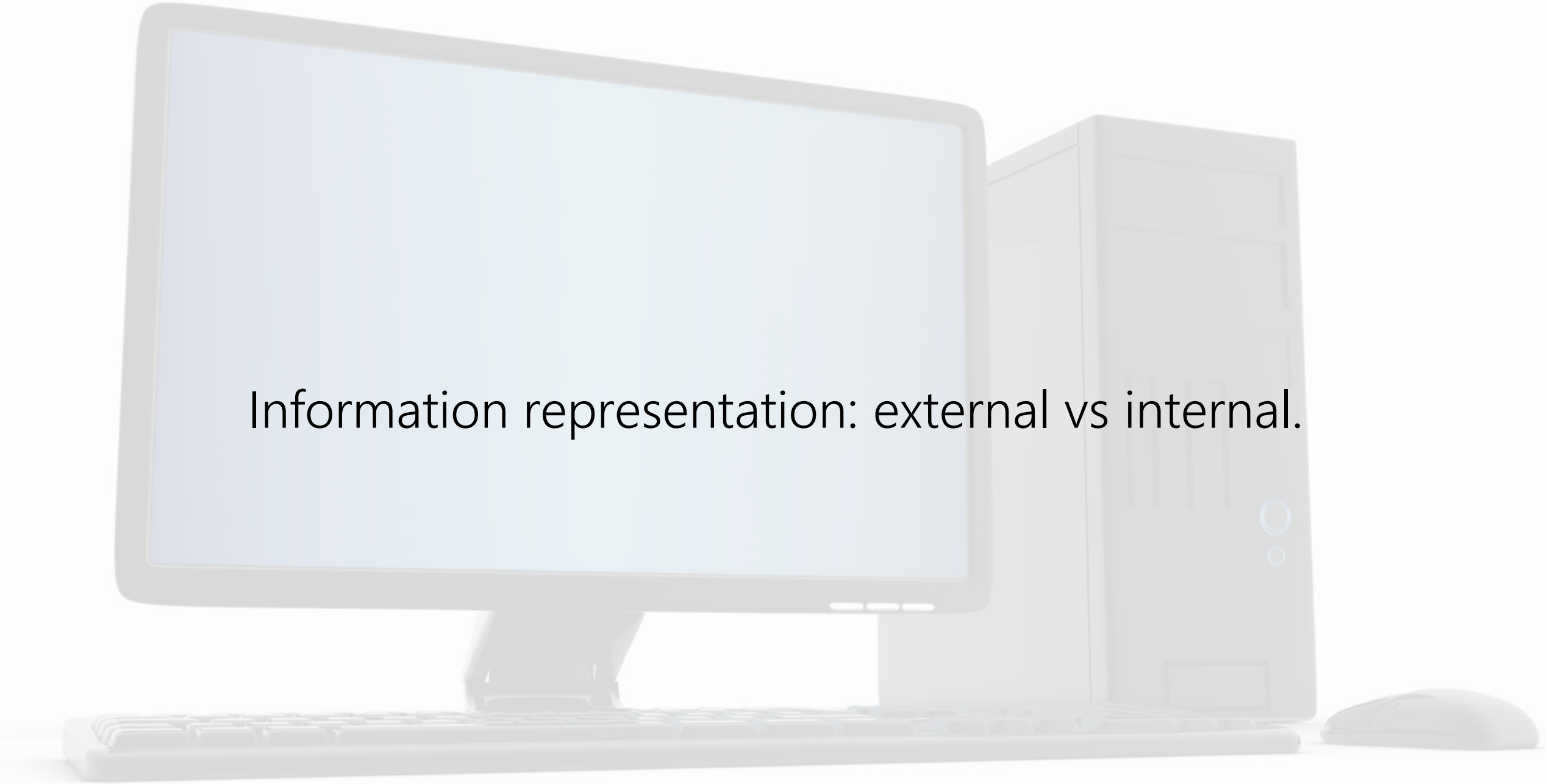
Aleksander Sadikov

2015/2016

0

What is a bit?

1



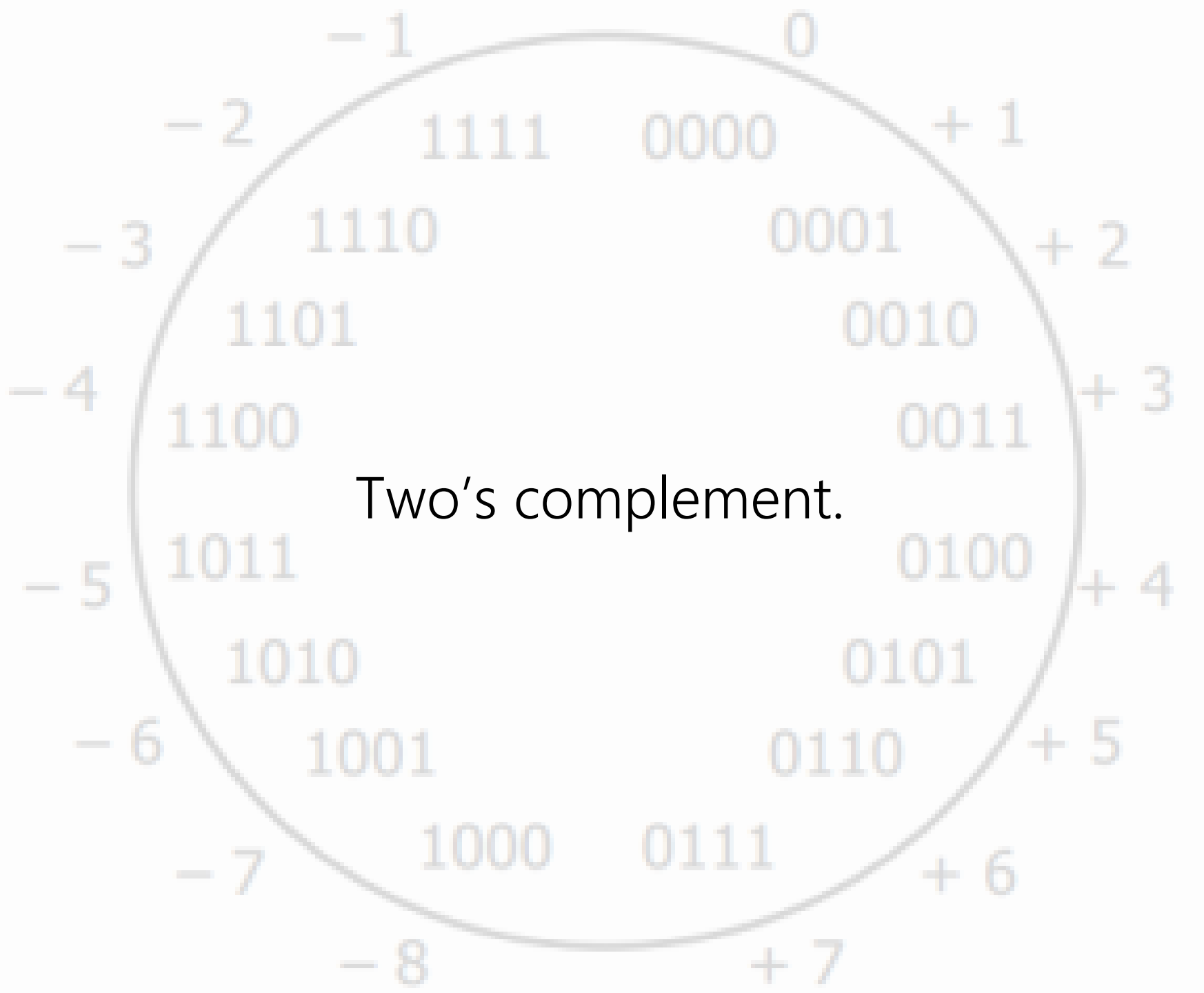
Information representation: external vs internal.


80
100
120
60
140
40
160
20
180
0
km/h
50
60
70
40
80
30
90
20
100
10
110
0
MPH
ODO
998665

Arithmetic is simple in binary.

10010101

Sign & magnitude notation:
easy for humans, very rare in computers.



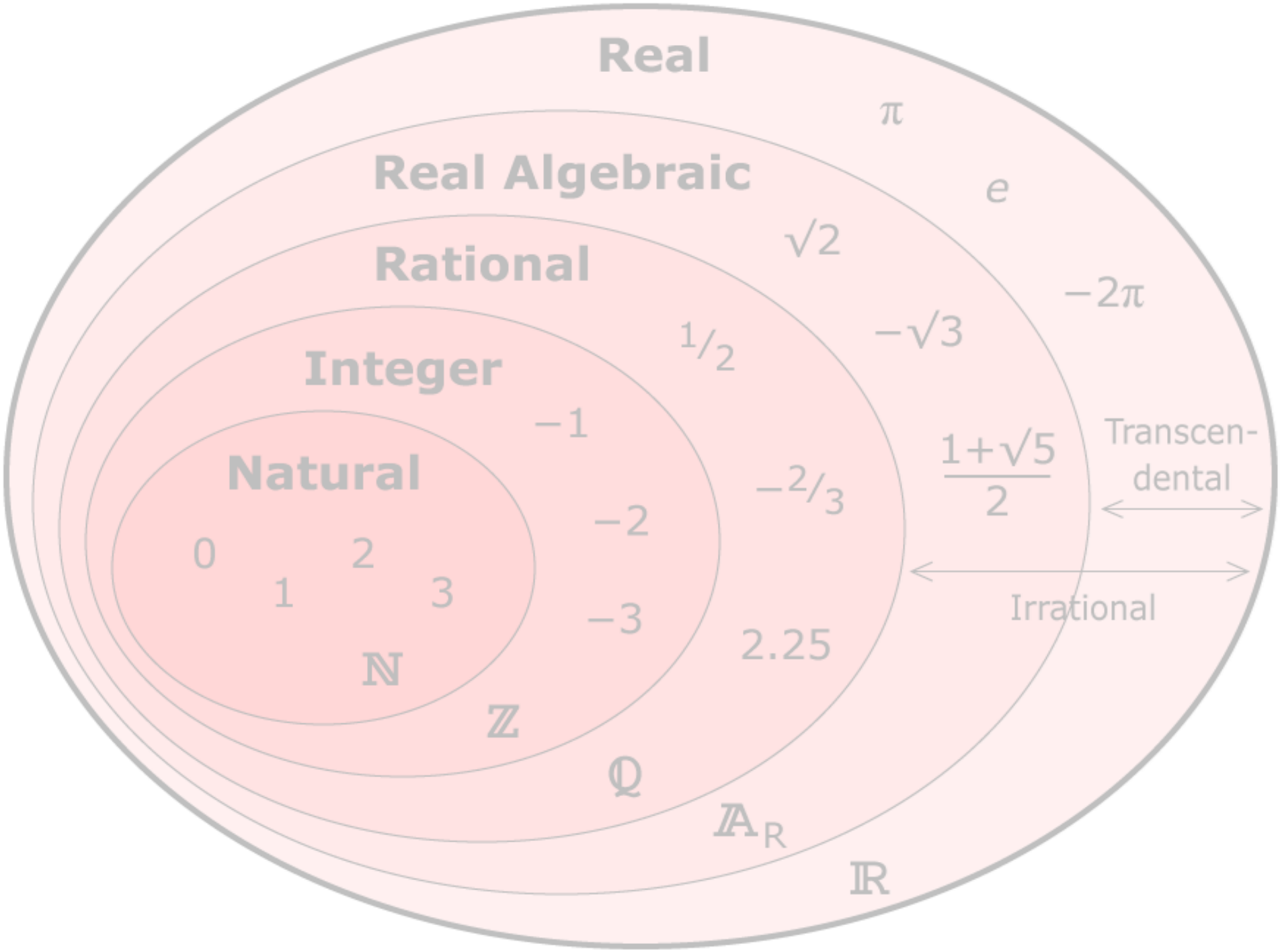
Two’s complement.

I don’t need subtraction!

Fractional numbers.

$$\pm M \times B^{\pm E}$$

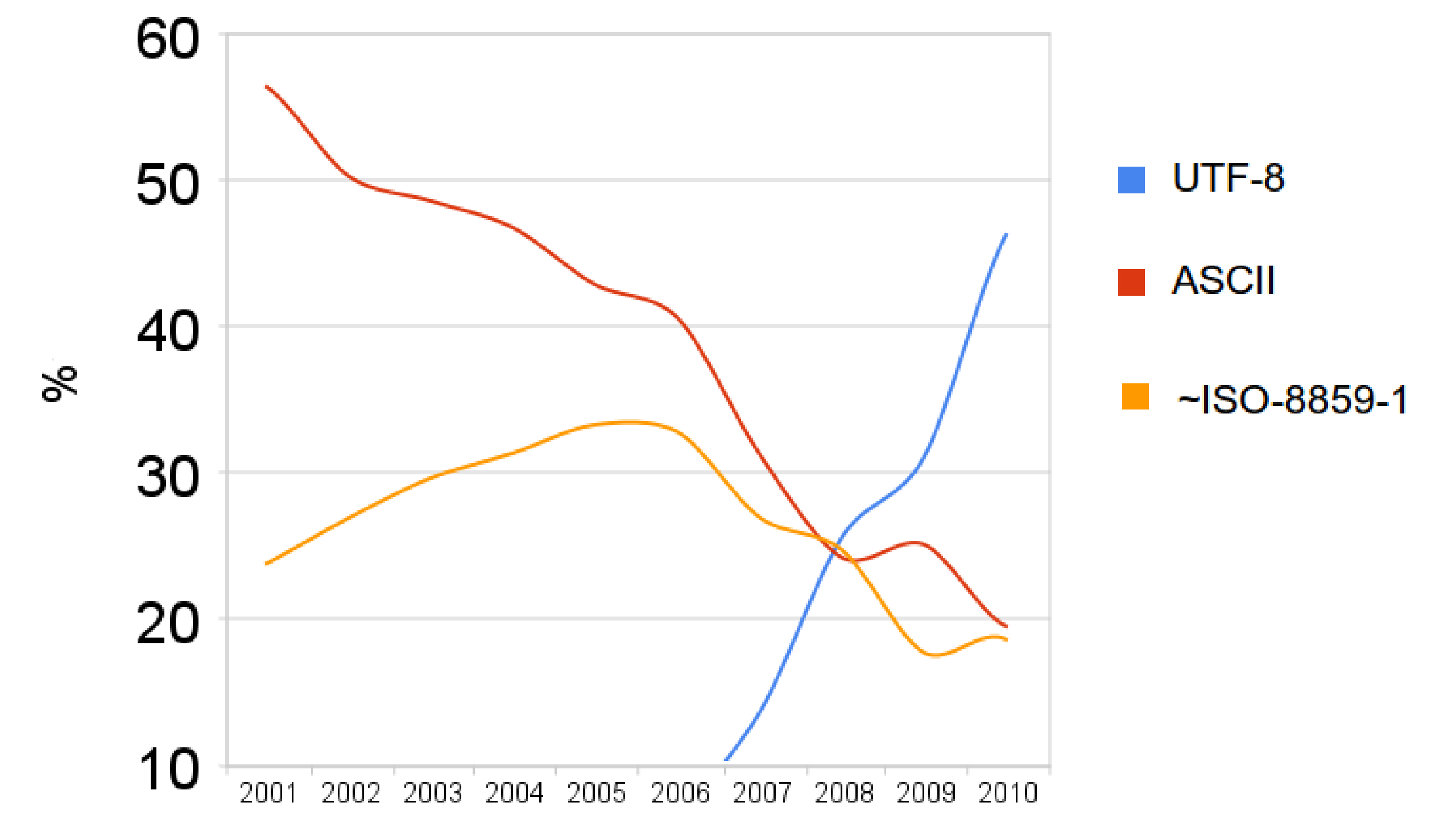
Integers
vs
floats.



Textual information.

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

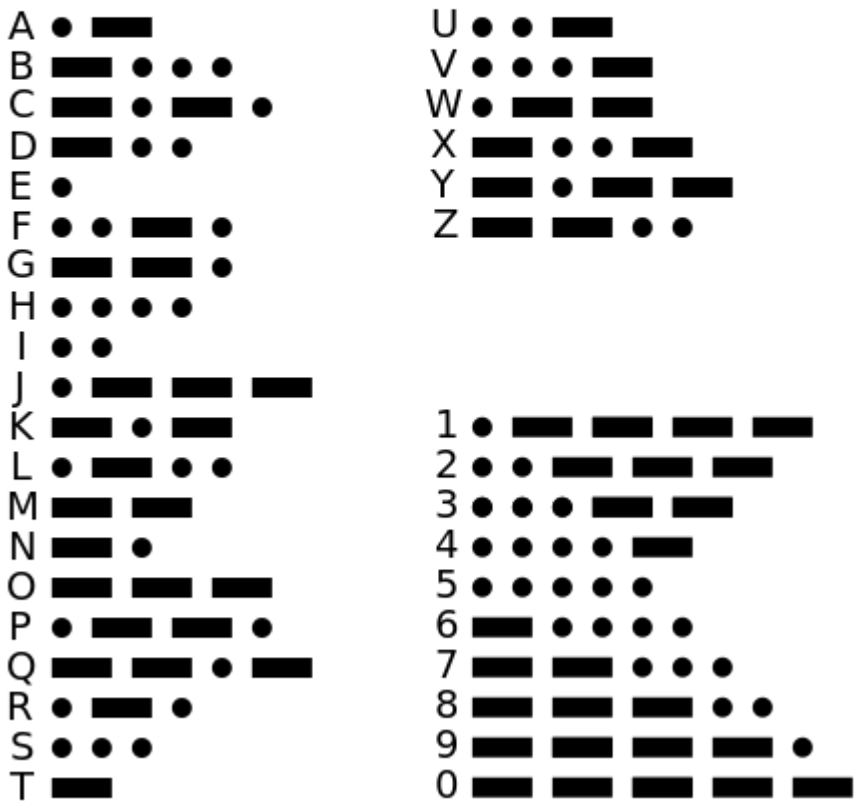
www.unicode.org

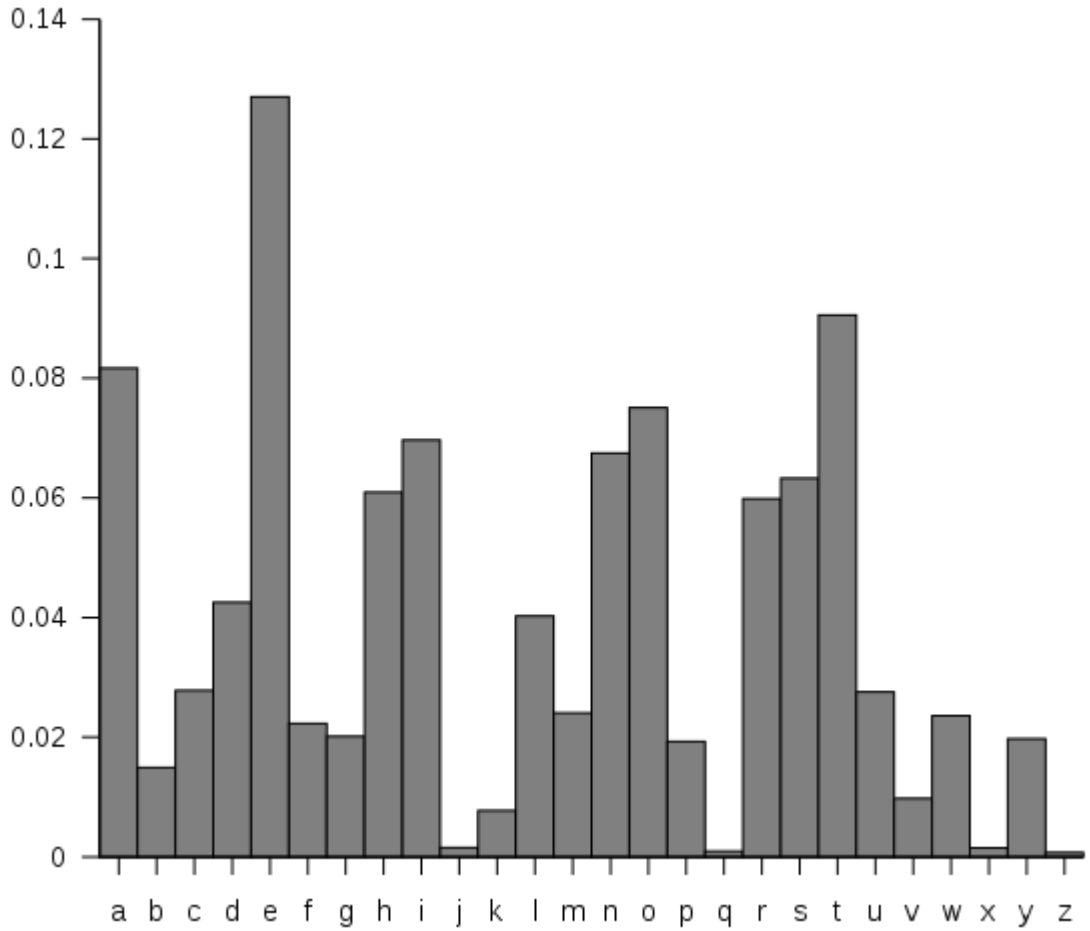

60
50
40
30
20
10
%
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
UTF-8
ASCII
~ISO-8859-1

Characters with lower numerical values,
which tend to occur more frequently,
are encoded using fewer bytes.

International Morse Code

1. The length of a dot is one unit.
2. A dash is three units.
3. The space between parts of the same letter is one unit.
4. The space between letters is three units.
5. The space between words is seven units.

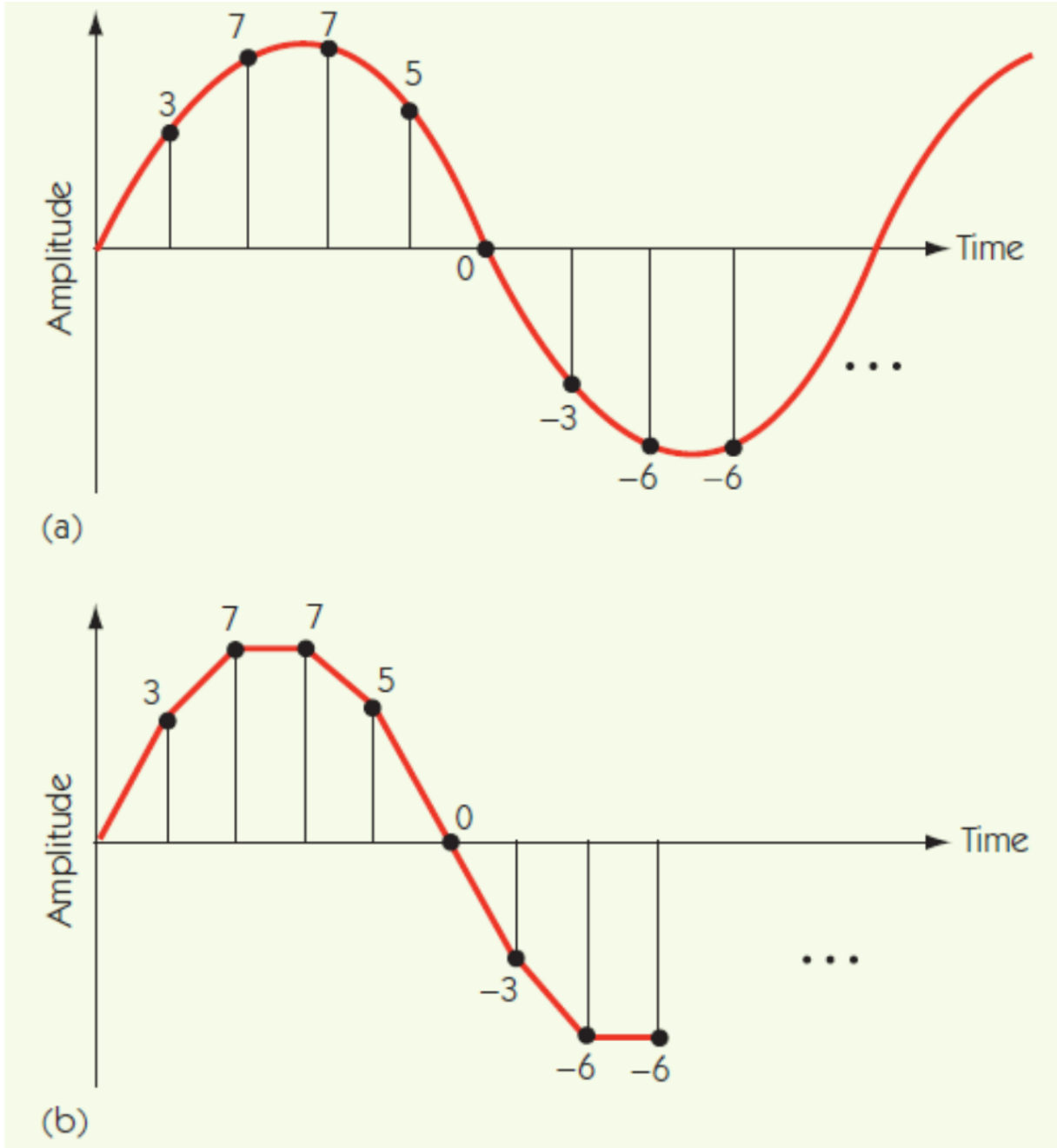




Sound.

Sound is analog information
as opposed to numbers and text which are digital.

… and sound is also a time series.


Amplitude
Time
3
7
7
5
0
−3
−6
−6
(a)
Amplitude
Time
3
7
7
5
0
−3
−6
−6
(b)

Sampling rate & bit depth.

The Nyquist–Shannon Sampling Theorem

44.1 kHz

44.1 kHz

What is a pixel?

So how many pixels/mm can your camera do?

How much information to store for each pixel?

A picture is worth a thousands words.
Only a thousand?!?

Let’s talk about compression a little bit…


PILOT


PILOT


PILOT

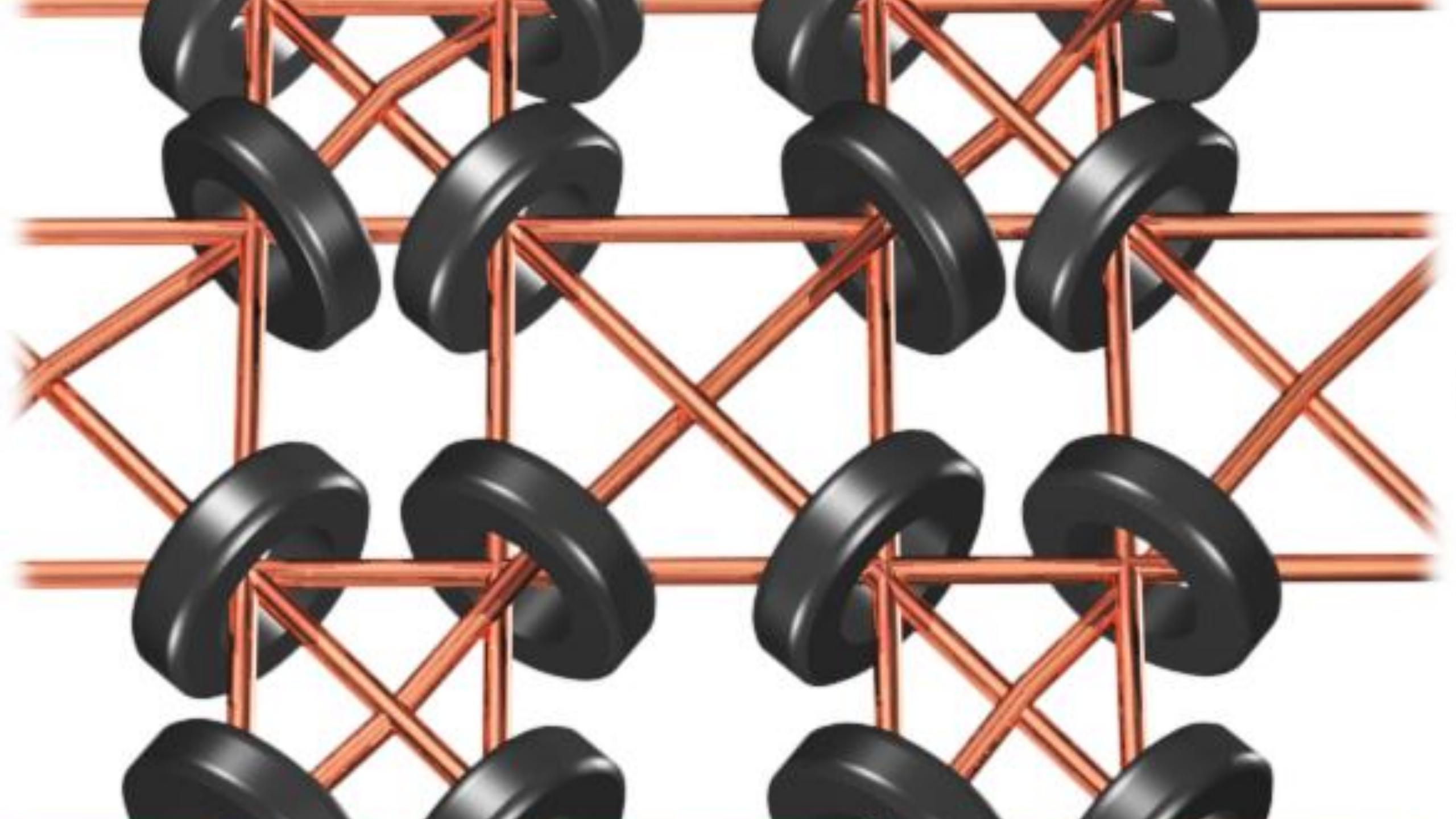
Hey, why binary?

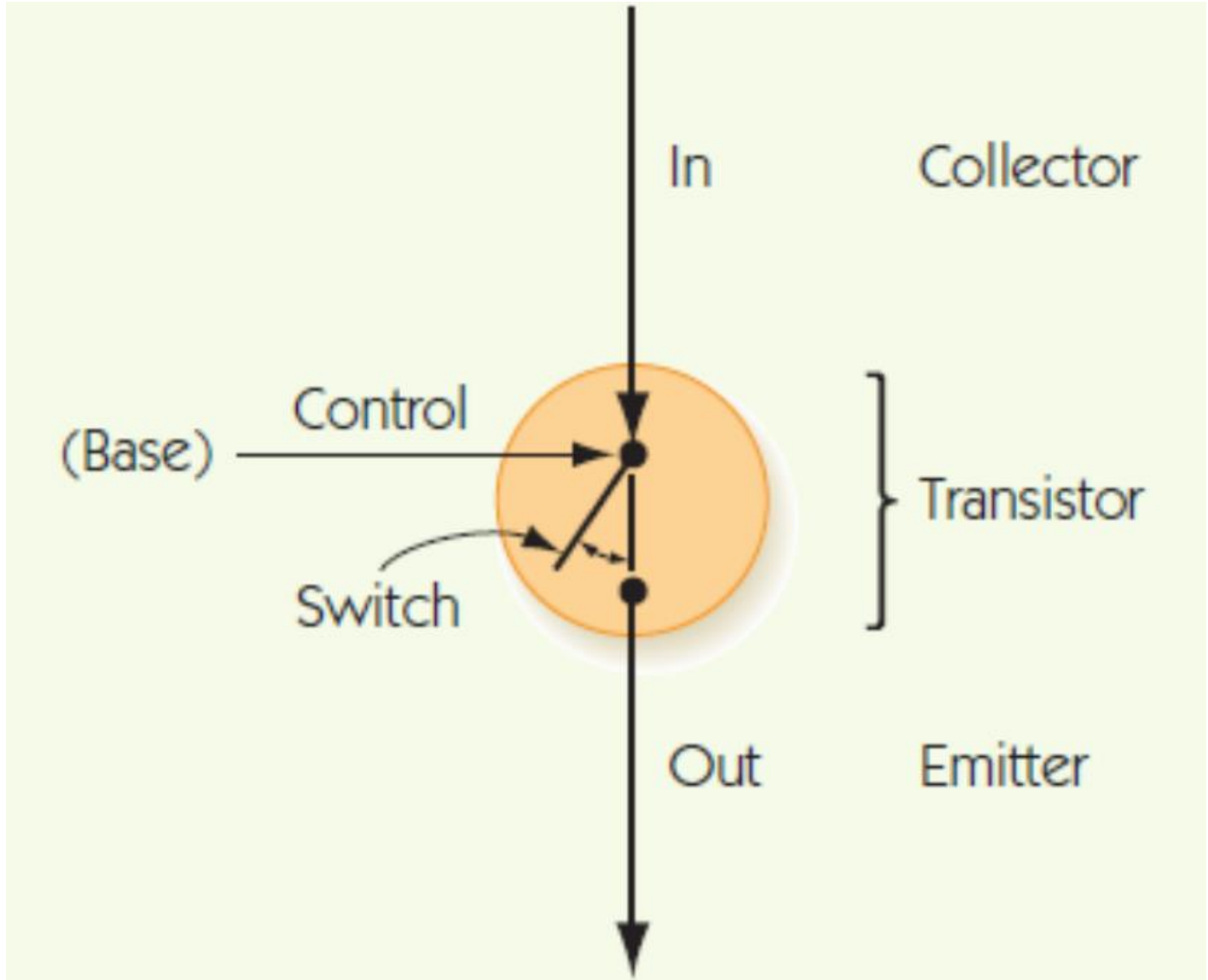
Reliability.

Building a reliable decimal machine would be an engineering nightmare.

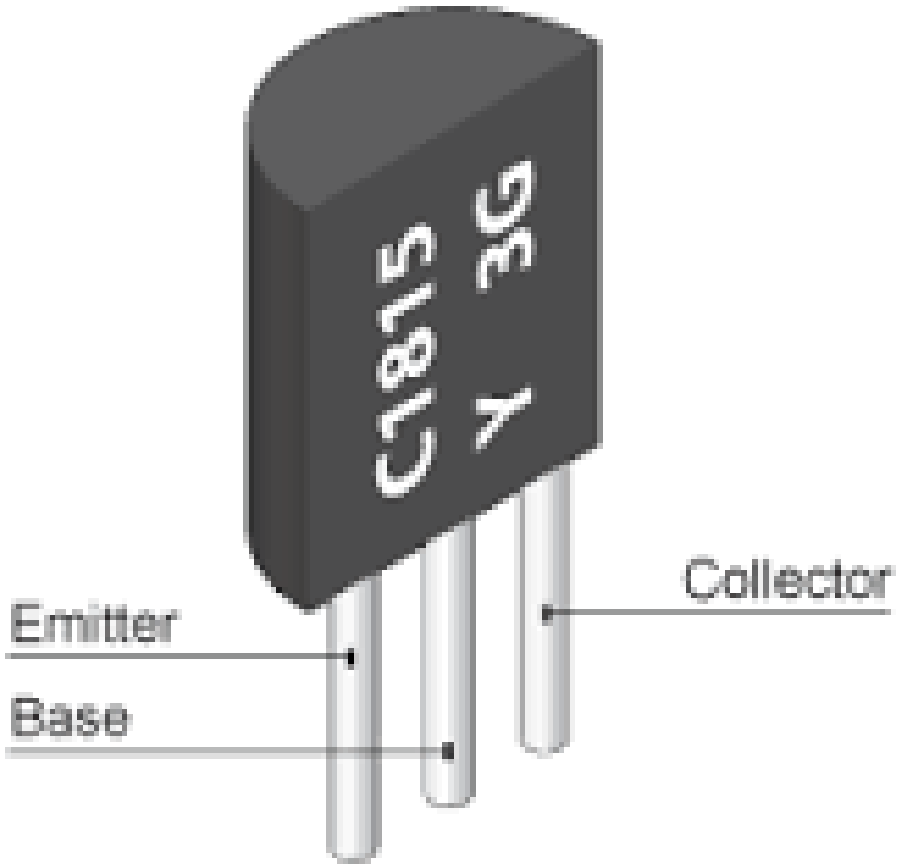
Hardware device should meet these four criteria:

1. Two stable energy states.
2. Large energy barrier between both states.
3. Possible to detect/sense state without destroying it.
4. Possible to switch state by applying enough energy.

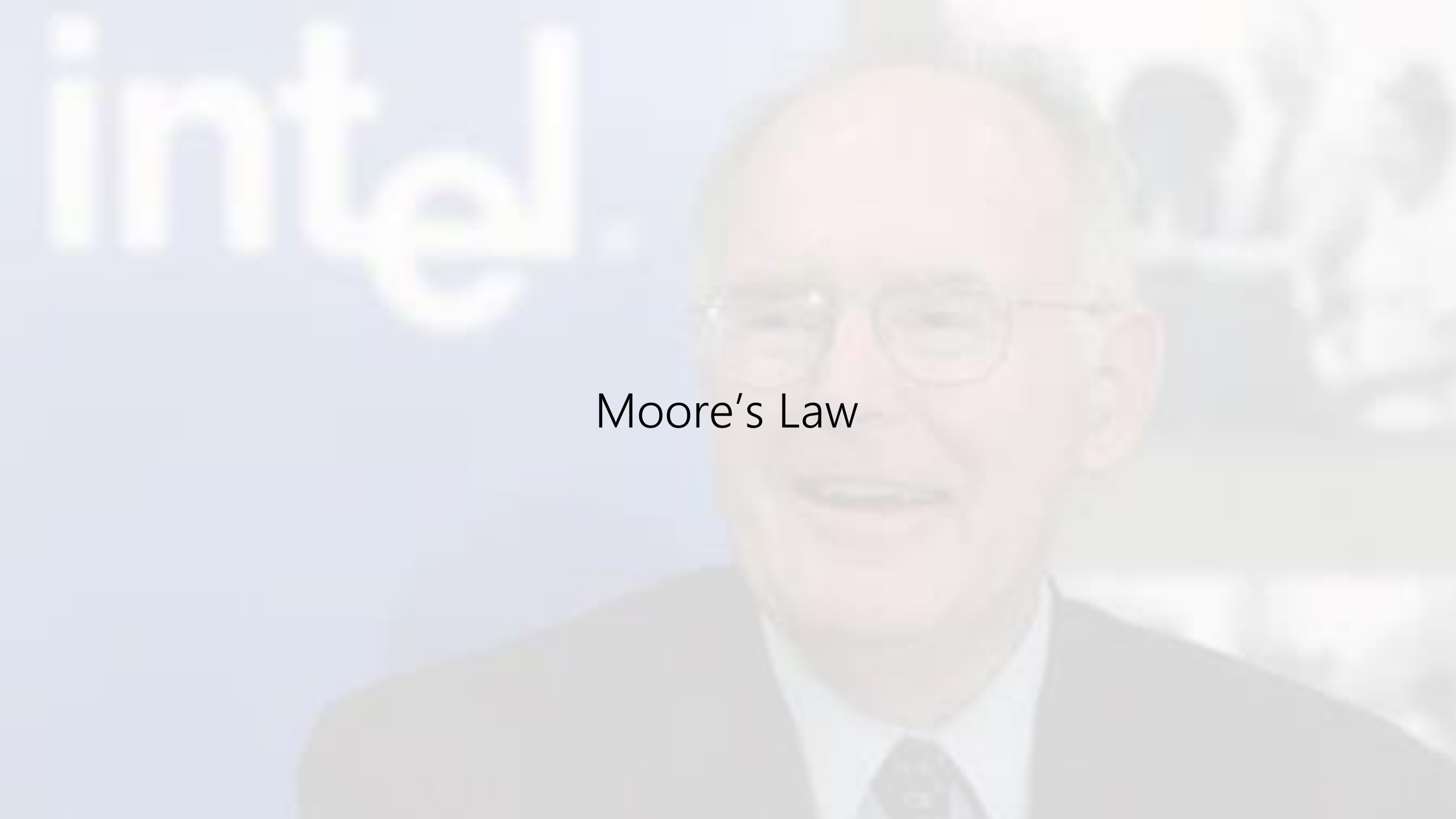




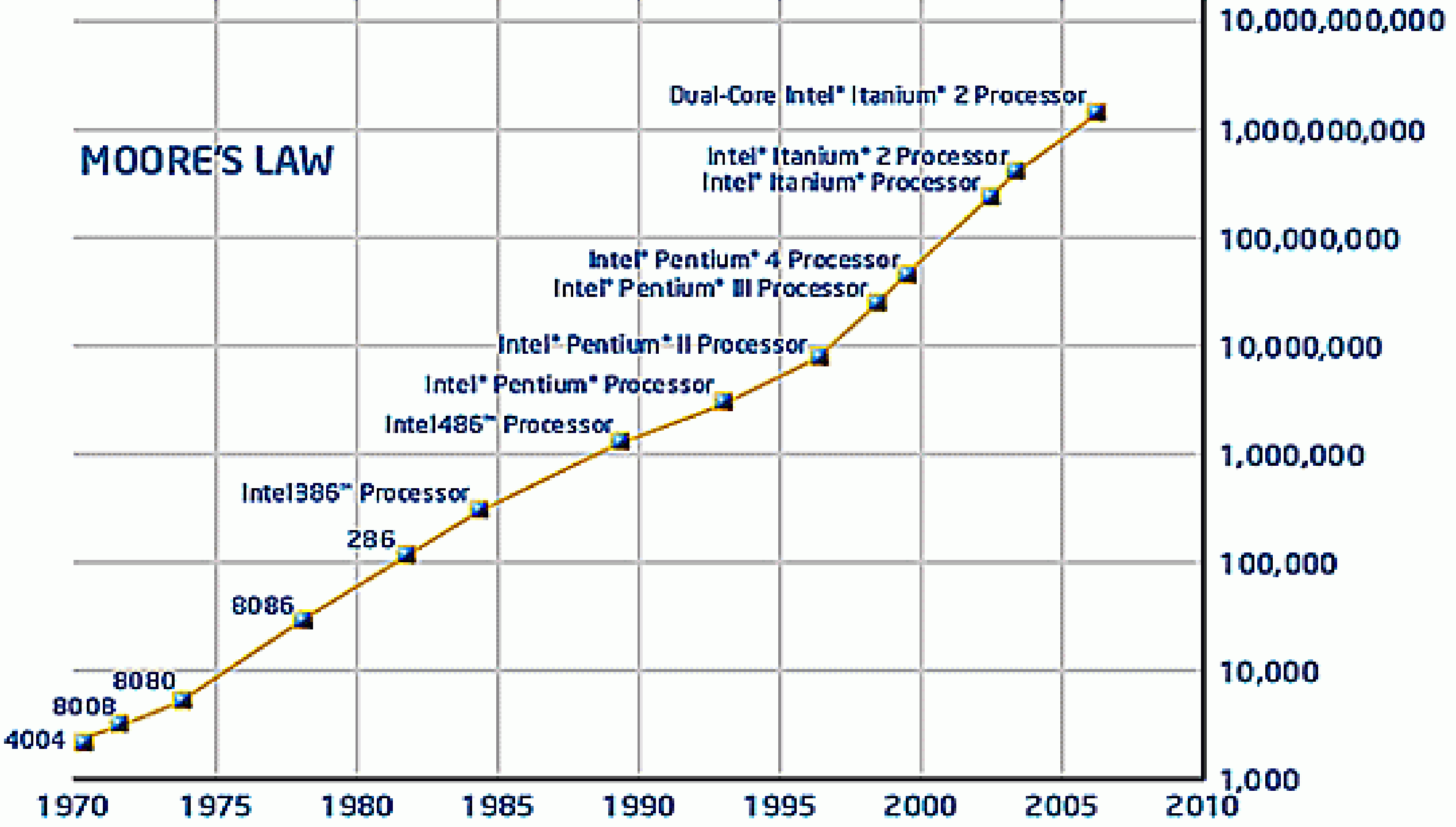
Simplified model of a transistor

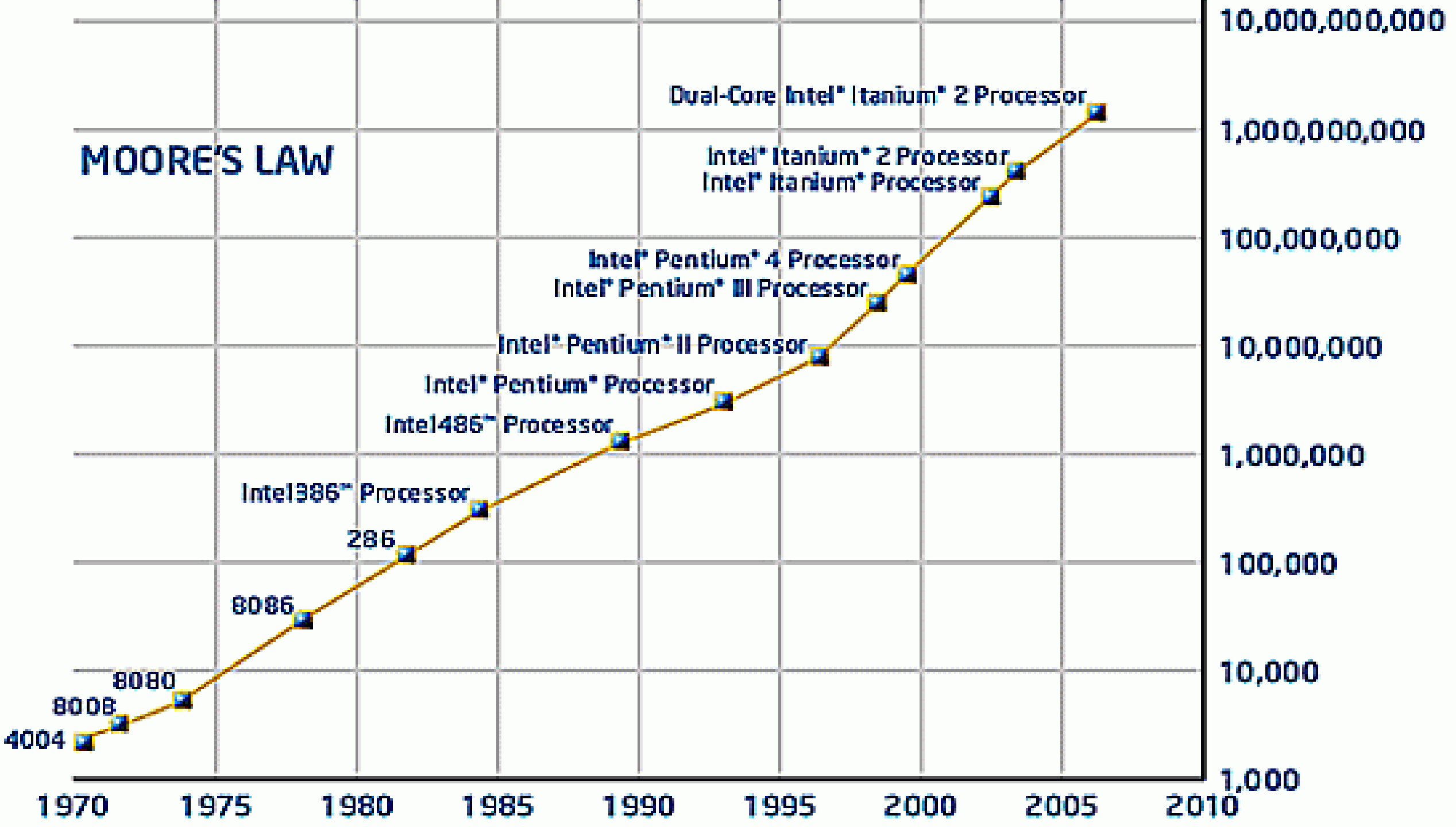

C1815
Y
SG
Emitter
Base
Collector

Today transistors are the building blocks for…
… well, everything. But…



Moore’s Law


MOORE'S LAW
Dual-Core Intel® Itanium® 2 Processor
Intel® Itanium® 2 Processor
Intel® Itanium® Processor
Intel® Pentium® 4 Processor
Intel® Pentium® III Processor
Intel® Pentium® II Processor
Intel® Pentium® Processor
Intel486™ Processor
Intel386™ Processor
286
8086
8080
8008
4004
10,000,000,000
1,000,000,000
100,000,000
10,000,000
1,000,000
100,000
10,000
1,000
1970
1975
1980
1985
1990
1995
2000
2005
2010


MOORE'S LAW
Dual-Core Intel® Itanium® 2 Processor
Intel® Itanium® 2 Processor
Intel® Itanium® Processor
Intel® Pentium® 4 Processor
Intel® Pentium® III Processor
Intel® Pentium® II Processor
Intel® Pentium® Processor
Intel486™ Processor
Intel386™ Processor
286
8086
8080
8008
4004
10,000,000,000
1,000,000,000
100,000,000
10,000,000
1,000,000
100,000
10,000
1,000
1970
1975
1980
1985
1990
1995
2000
2005
2010

Boolean logic.

Inputs		Output a AND b
a	b	(also written a · b)
False	False	False
False	True	False
True	False	False
True	True	True

Inputs		Output a OR b
a	b	(also written a + b)
False	False	False
False	True	True
True	False	True
True	True	True

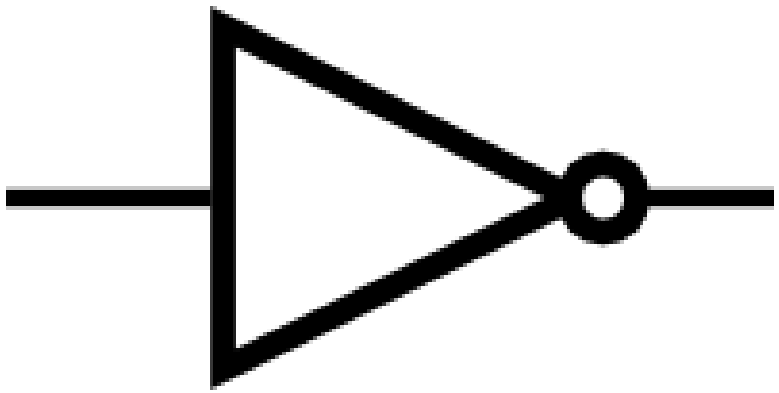
Input	Output NOT a
a	(also written $\overline{a}$)
False	True
True	False

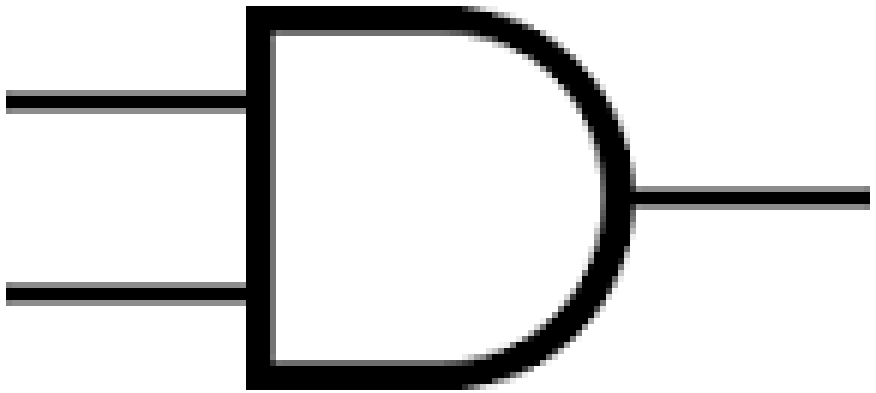
Why AND, OR, and NOT?

Gates.

No, not Bill.

Not this time.





Gates just seen are **not** abstract ideas
as mathematical definitions were.

Combinational Circuit.

Every Boolean expression
can be represented as a circuit diagram,
and every output value in a circuit diagram
can be written as a Boolean expression.

Sum of Products algorithm.

1. Truth table construction.
2. Subexpression construction
   using AND and NOT gates (product phase).
3. Subexpression combination using OR gates.
4. Circuit diagram production.

The resulting circuit is **not** necessarily optimal in the number of logic gates used.

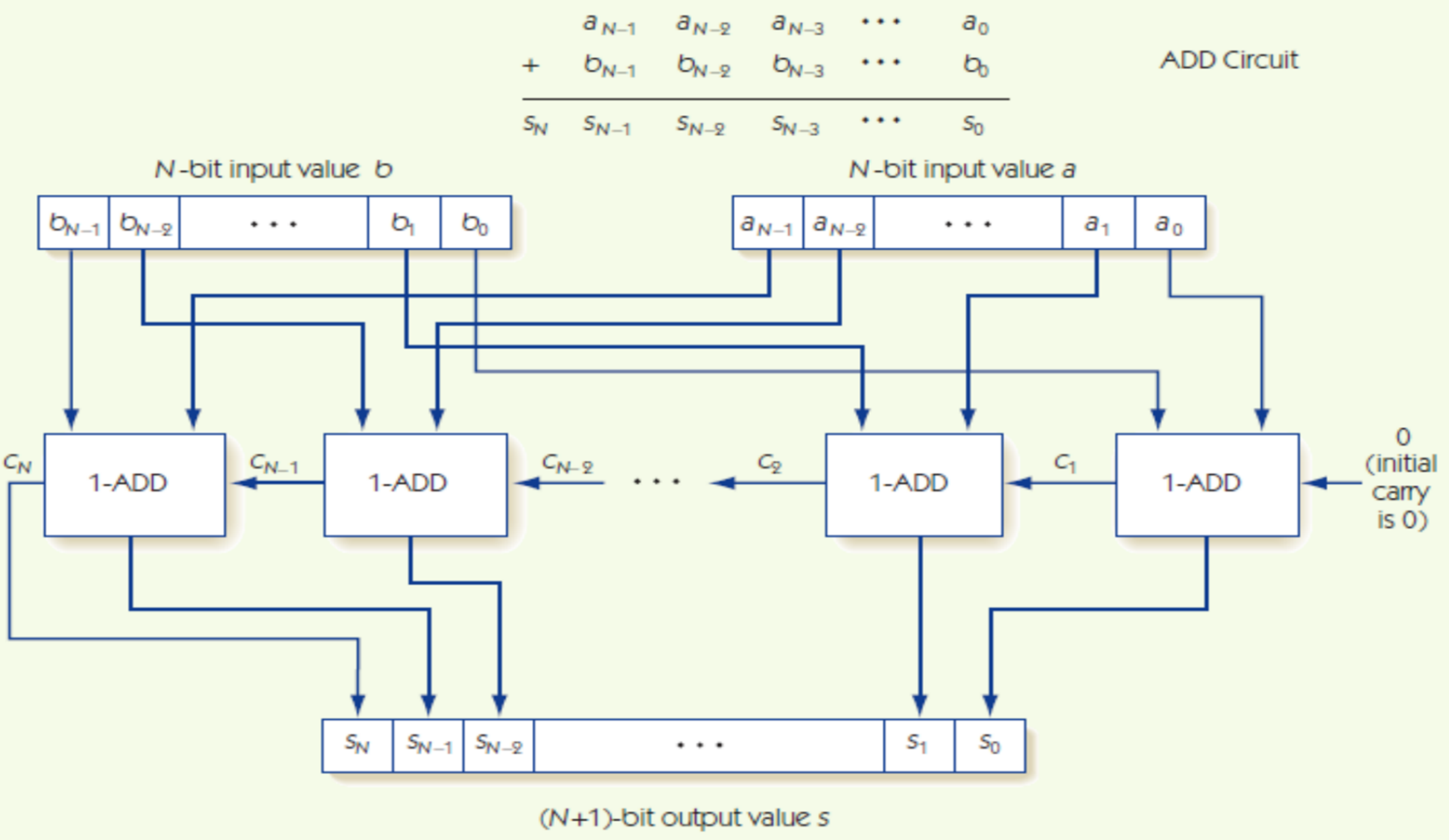
(money, space, power, heat)

Compare for Equality (CE) circuit.

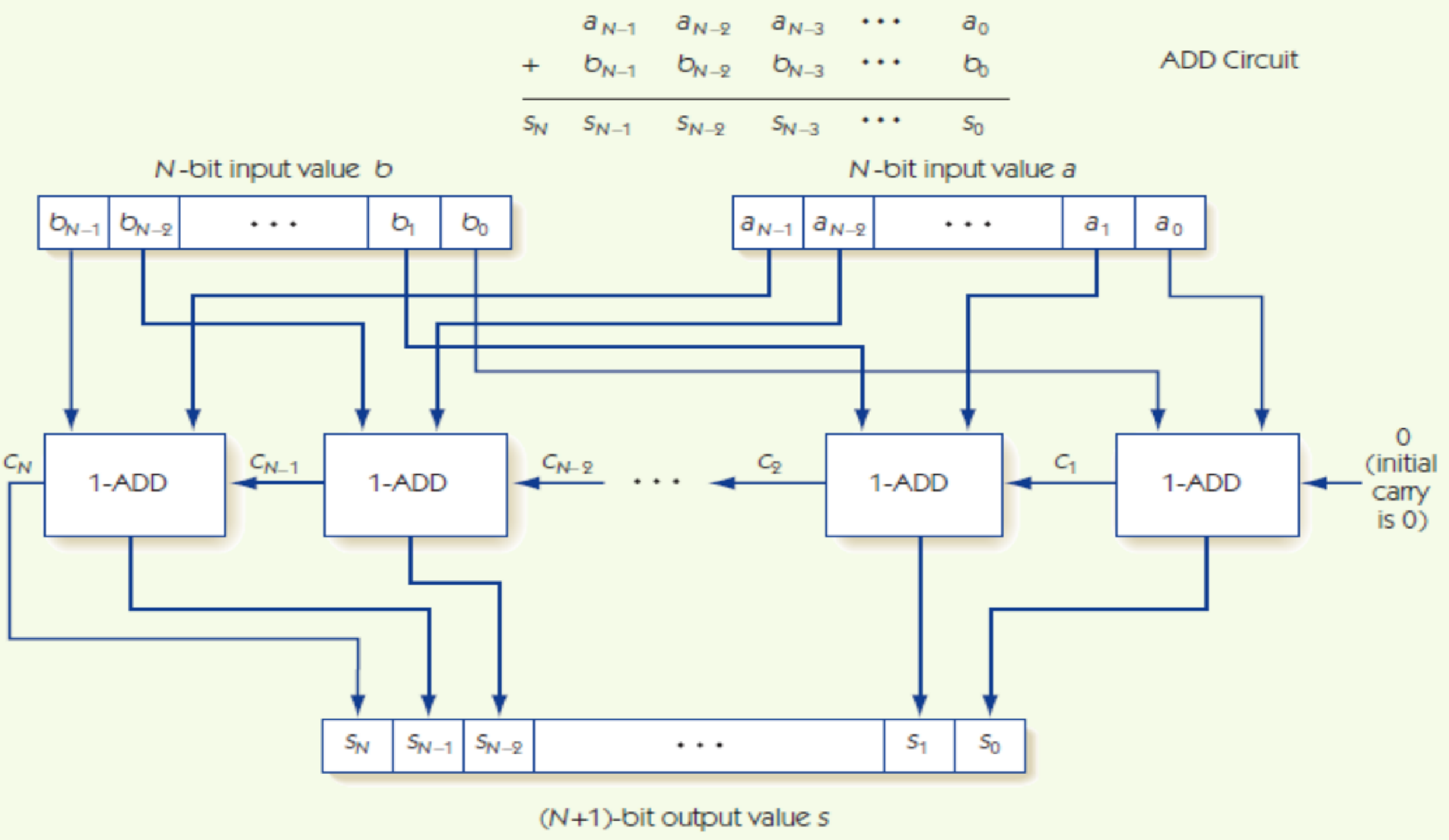
Complement circuit.

1-ADD circuit.

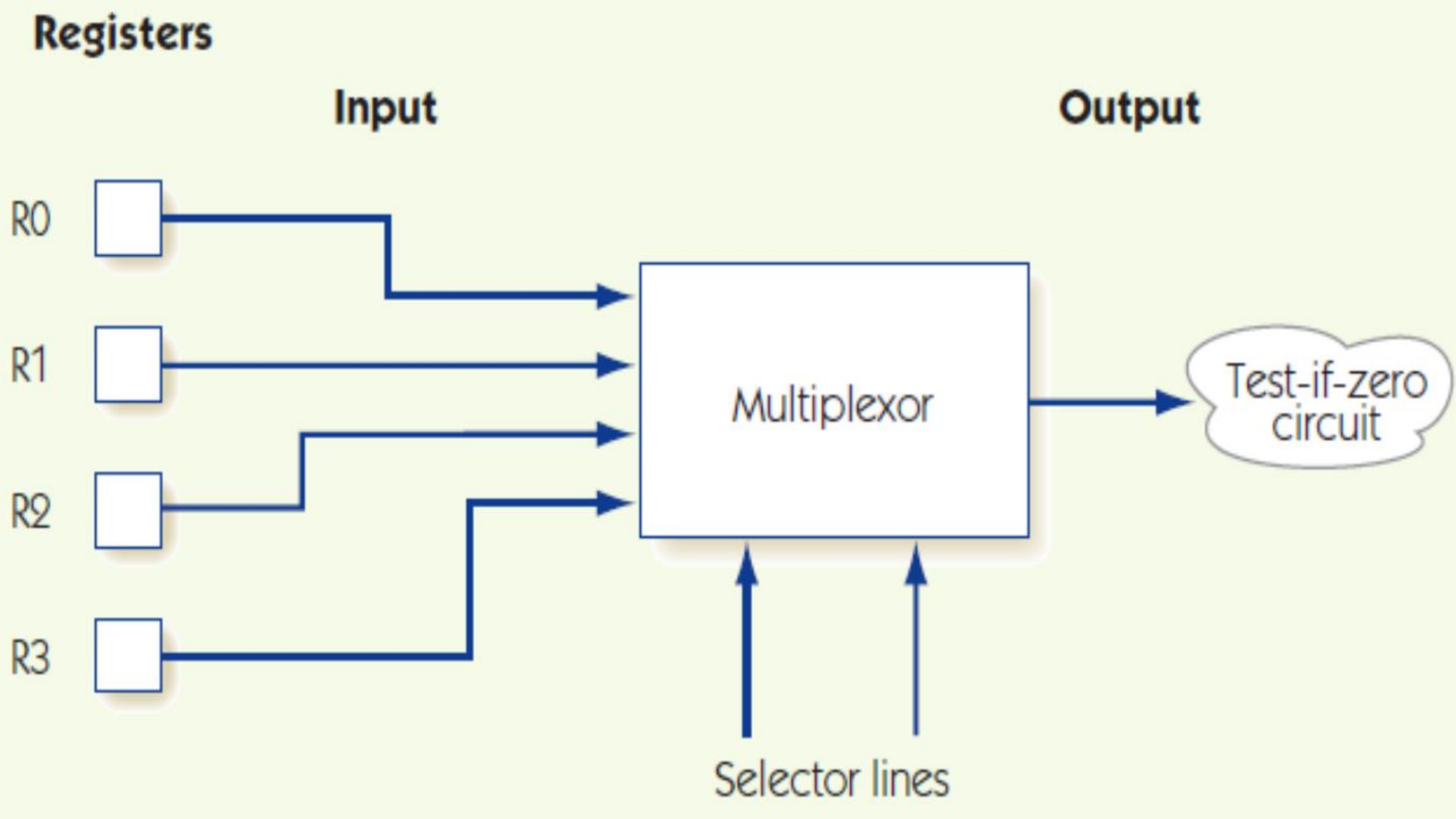
ADD Circuit

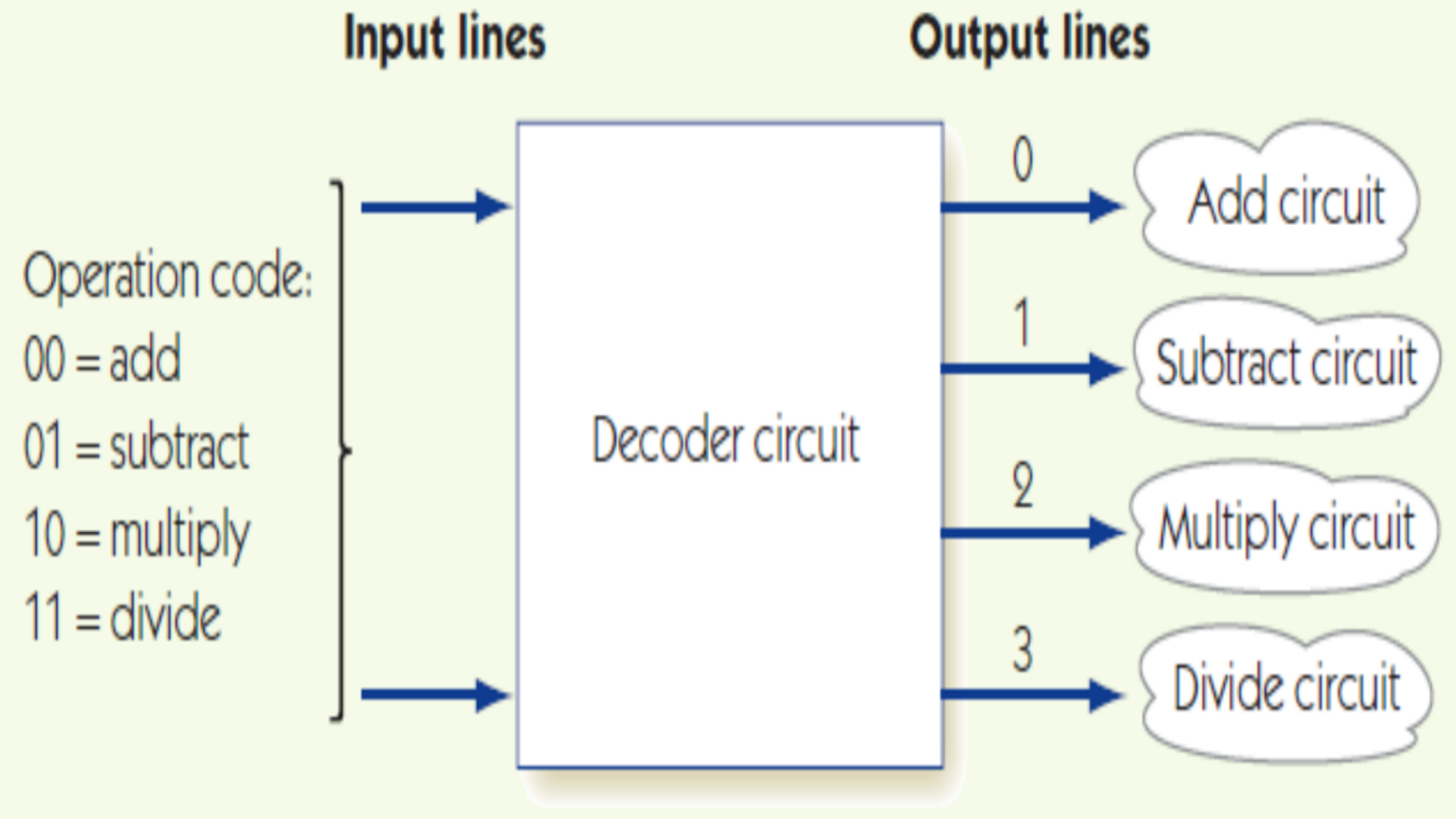

a_{N-1} a_{N-2} a_{N-3} ••• a_0
+ b_{N-1} b_{N-2} b_{N-3} ••• b_0
s_N s_{N-1} s_{N-2} s_{N-3} ••• s_0
N-bit input value b
N-bit input value a
b_{N-1} b_{N-2} ••• b_1 b_0
a_{N-1} a_{N-2} ••• a_1 a_0
c_N
1-ADD
c_{N-1}
1-ADD
c_{N-2}
•••
c_2
1-ADD
c_1
1-ADD
0 (initial carry is 0)
s_N s_{N-1} s_{N-2} ••• s_1 s_0
(N+1)-bit output value s

ADD Circuit


a_{N-1} a_{N-2} a_{N-3} ••• a_0
+ b_{N-1} b_{N-2} b_{N-3} ••• b_0
s_N s_{N-1} s_{N-2} s_{N-3} ••• s_0
N-bit input value b
N-bit input value a
b_{N-1} b_{N-2} ••• b_1 b_0
a_{N-1} a_{N-2} ••• a_1 a_0
c_N
1-ADD
c_{N-1}
1-ADD
c_{N-2}
•••
c_2
1-ADD
c_1
1-ADD
0 (initial carry is 0)
s_N s_{N-1} s_{N-2} ••• s_1 s_0
(N+1)-bit output value s

Control circuits.
Multiplexor & Decoder.


Registers
Input
Output
R0
R1
R2
R3
Multiplexor
Test-if-zero circuit
Selector lines


Input lines
Output lines
Operation code:
00 = add
01 = subtract
10 = multiply
11 = divide
Decoder circuit
0
1
2
3
Add circuit
Subtract circuit
Multiply circuit
Divide circuit

Levels of abstraction:
from transistors to gates, from electricity to arithmetic.

Levels of abstraction:
from transistors to gates, from electricity to arithmetic.

Take away lessons #P4/P5

Practically everything can be represented with ones and zeros (with a little imagination). If it is presented in this way, we can algorithmically manipulate such data.

Abstraction. No need to think of electrical (or other) details once we build basic gates. Think logically instead!

Electricity is not the only way.