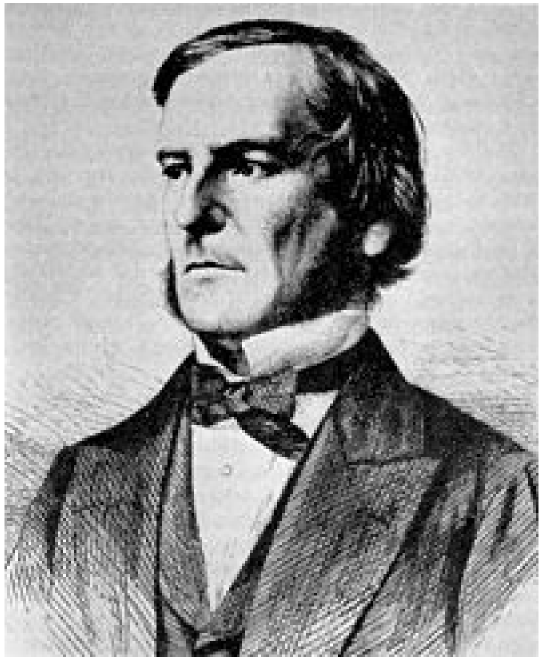
CS E1: Section 1: Counting in Binary

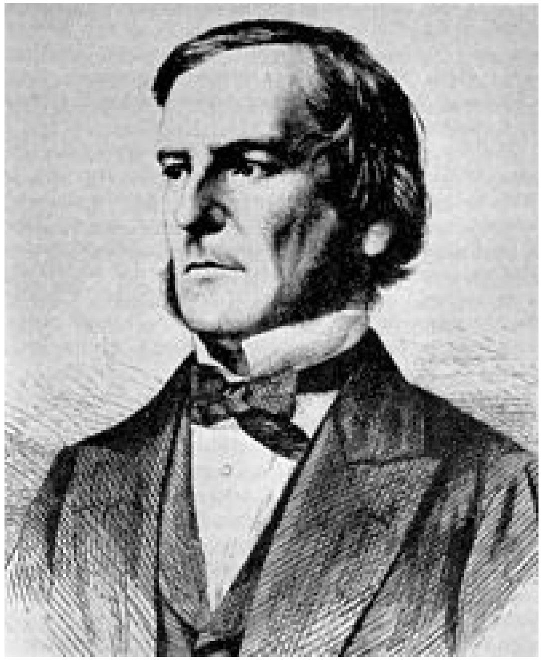
CS E1, Section 1:
The History of 1 and 0

George Boole (1815-1864)

- English mathematician
- Invented boolean logic



George Boole (1815-1864)

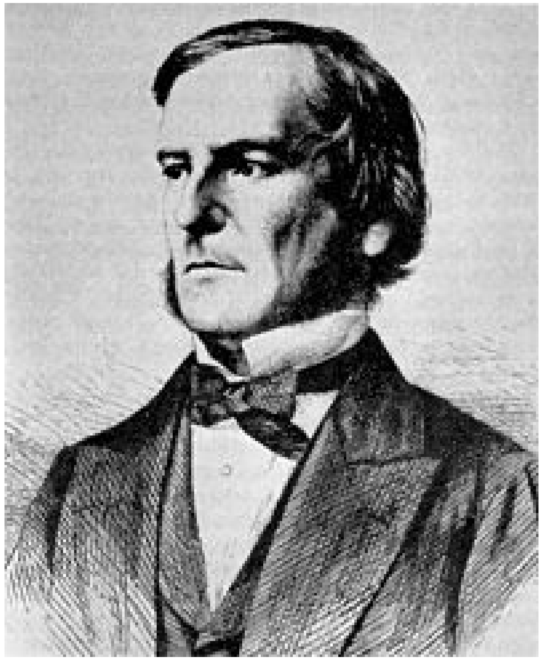


- English mathematician
- Invented boolean logic

Boolean Logic is everywhere□

- boolean tools are built into search engines:
  - Boston Public Library
  - Google
  - Gmail

George Boole (1815-1864)

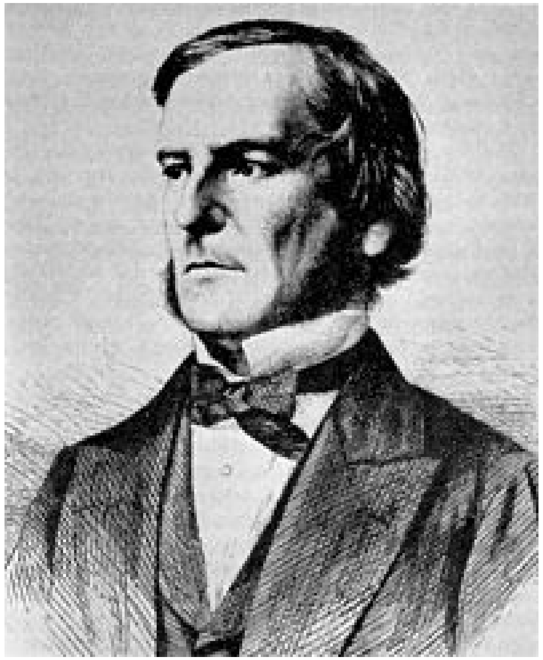


- English mathematician
- Invented boolean logic

Boolean Logic is everywhere□

- lets you search for
  - Boston Public Library
    - sharks NOT attack
  - Google
  - Gmail

George Boole (1815-1864)

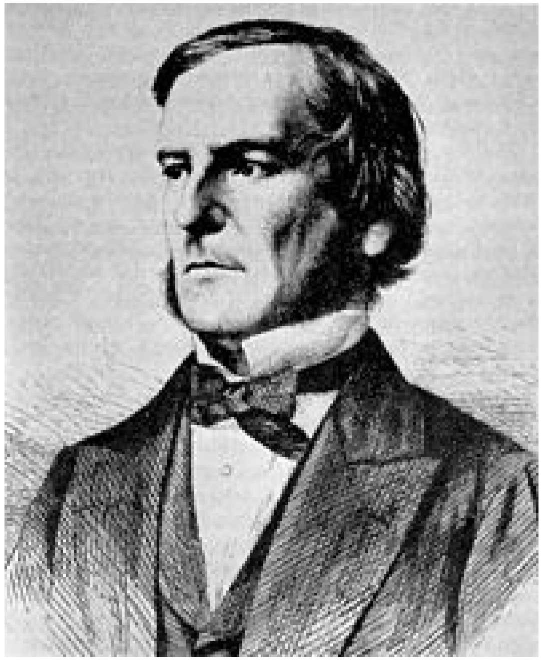


- English mathematician
- Invented boolean logic

Boolean Logic is everywhere□

- lets you search for
  - Boston Public Library
    - sharks NOT attack
  - Google
    - "David Malan" AND "Dan Armendariz"
  - Gmail

George Boole (1815-1864)



- English mathematician
- Invented boolean logic

Boolean Logic is everywhere□

- lets you search for
  - Boston Public Library
    - sharks NOT attack
  - Google
    - "David Malan" AND "Dan Armendariz"
  - Gmail
    - from:pnore@fas.harvard.edu OR from: contact@tombarasso.com

Claude Shannon (1916-2001)

- American Mathematician
- Founded circuit design at 21 with his Master's thesis at MIT
- Called the "most important master's thesis of all time"

Claude Shannon (1916-2001)

- American Mathematician
- Founded circuit design at 21 with his Master's thesis at MIT
- Called the "most important master's thesis of all time"

The thesis describes how Boolean logic circuits can represent any logical or numerical relationship

Claude Shannon (1916-2001)

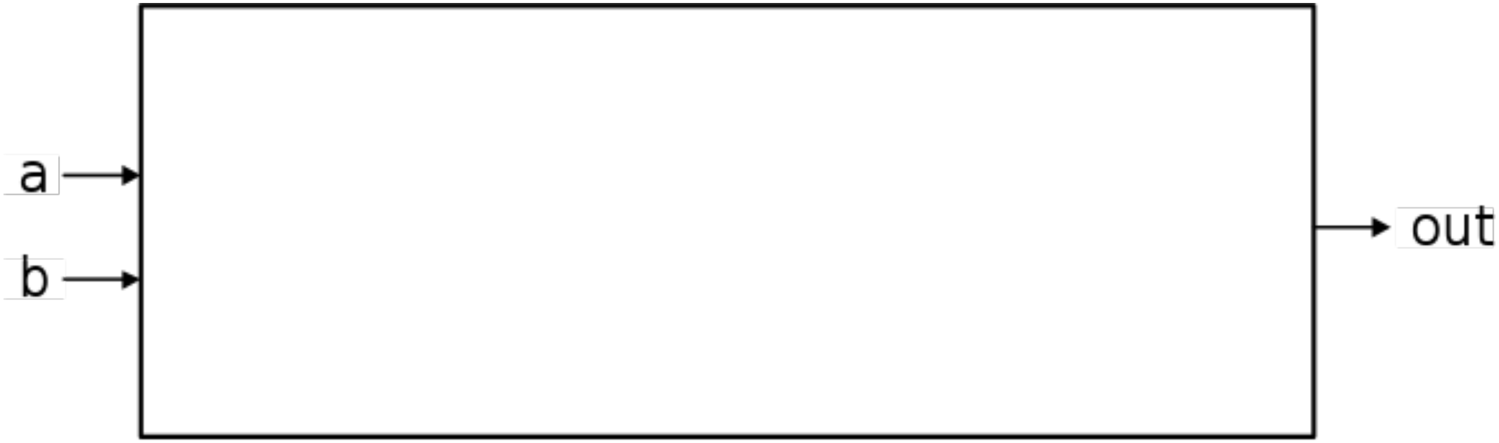
- American Mathematician
- Founded circuit design at 21 with his Master's thesis at MIT
- Called the "most important master's thesis of all time"

The thesis describes how Boolean logic circuits can represent any logical or numerical relationship

(wow - that's what makes computers possible)

What is a function?

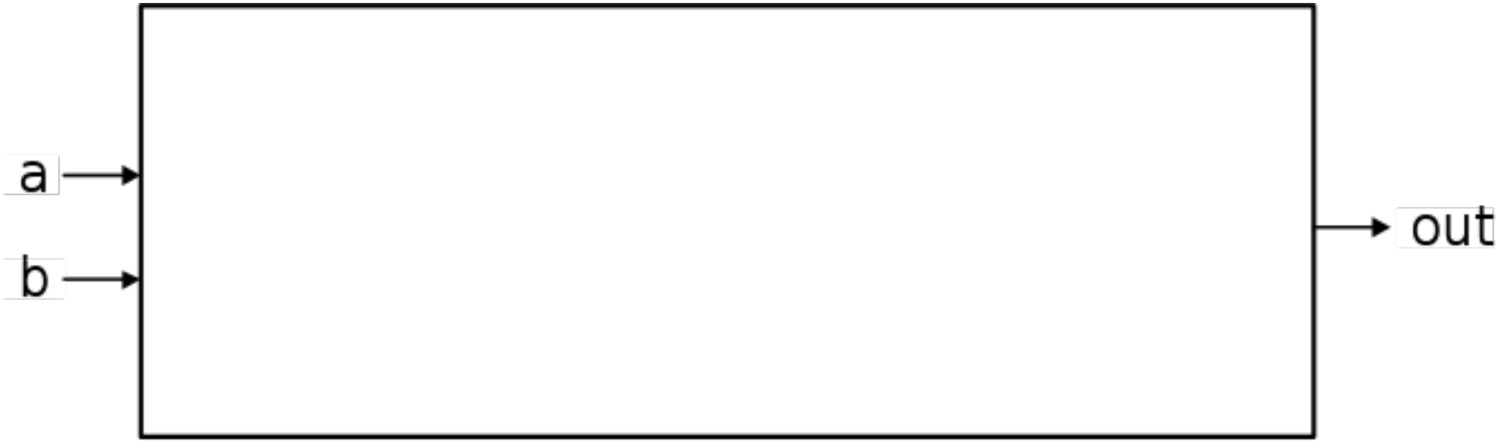
A function is ...



What is a function?

A function is ...

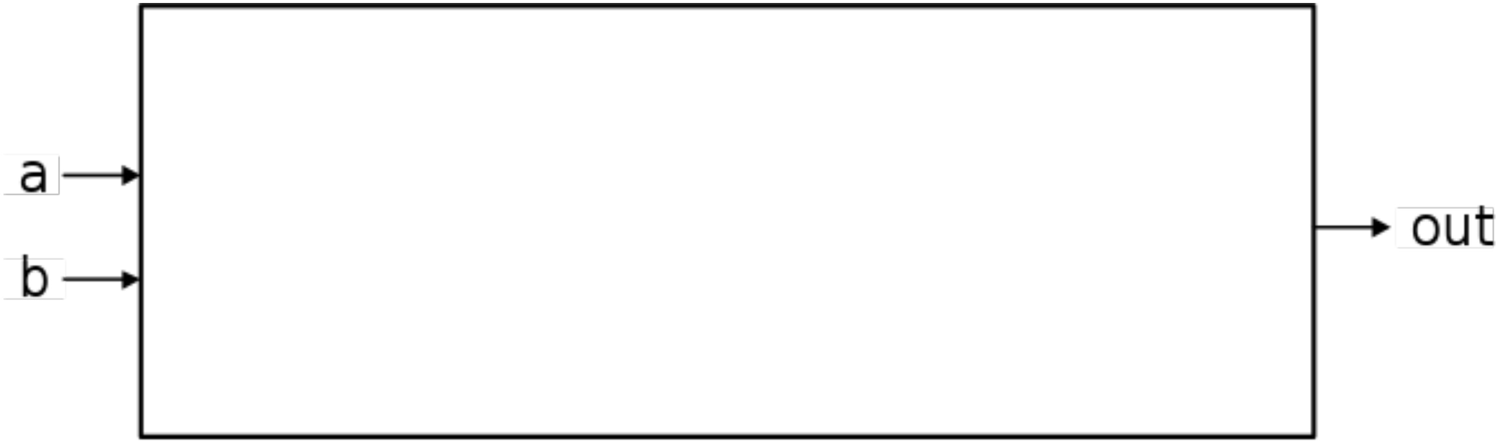
- like a "machine" that turns inputs into an output



What is a function?

A function is ...

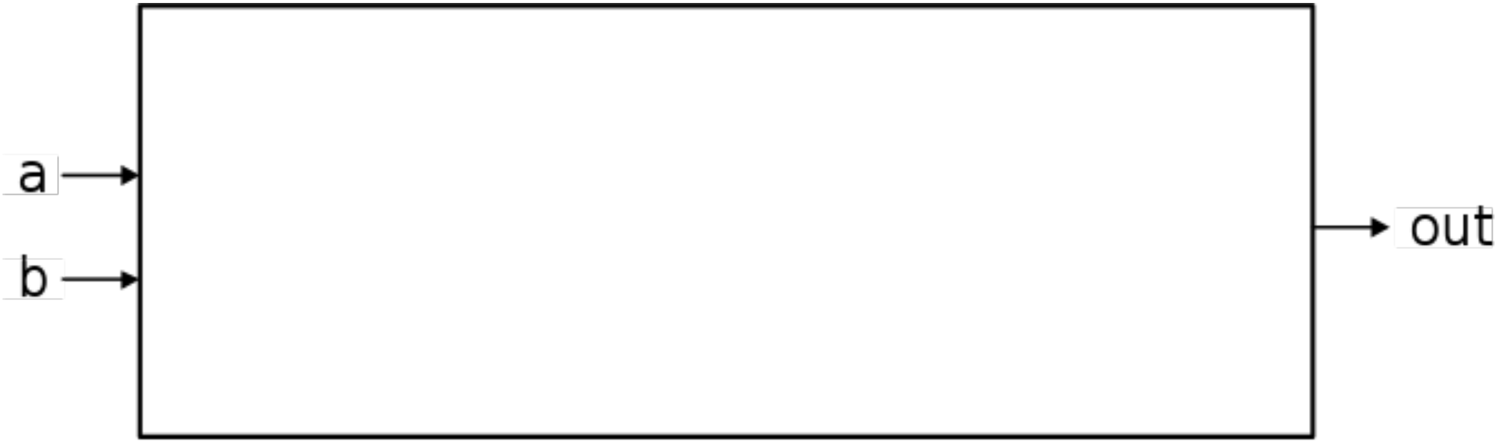
- like a "machine" that turns inputs into an output
- like a "black box" - to use it we don't need to know how it works, we just need to know what it needs (inputs) and what it does (output)



What is a function?

For our purposes, a function is ...

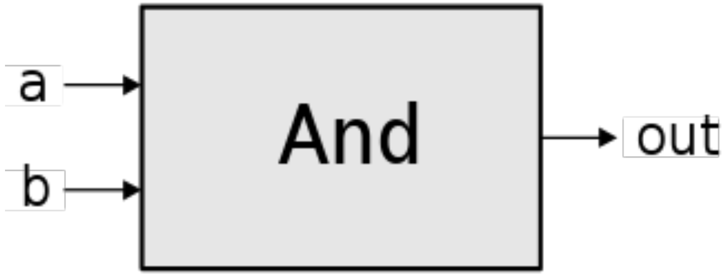
- like a "machine" that turns inputs into an output
- like a "black box" - to use it we don't need to know how it works, we just need to know what it needs (inputs) and what it does (output)
- repeatable. The same inputs lead to the same output.



How do we describe a function? With a truth table.

a and b are inputs of function
"out" is the output of the function

a	b	out
0	0	0
0	1	0
1	0	0
1	1	1



What is the truth table of the "NOT" function?

NOT

How many inputs does it have?

What is the truth table of the "NOT" function?

How many inputs does it have?

ONE

How many outputs does it have?

NOT

a
0
1

What is the truth table of the "NOT" function?

How many inputs does it have?

ONE

How many outputs does it have?

ONE

What is the relationship between them?

NOT

a	out
0	
1	

What is the truth table of the "NOT" function?

How many inputs does it have?

ONE

How many outputs does it have?

ONE

What is the relationship between them?

OPPOSITE

NOT

a	out
0	1
1	0

What is the truth table of the "OR" function?

How many inputs does it have?

TWO

How many outputs does it have?

ONE

What is the relationship between them?

OR

a	b	out
0	0	
0	1	
1	0	
1	1	

What is the truth table of the "OR" function?

How many inputs does it have?

TWO

How many outputs does it have?

ONE

What is the relationship between them?

OR

a	b	out
0	0	0
0	1	
1	0	
1	1	

What is the truth table of the "OR" function?

How many inputs does it have?

TWO

How many outputs does it have?

ONE

What is the relationship between them?

OR

a	**b**	**out**
0	**0**	**0**
0	**1**	**1**
1	**0**	
1	**1**	

What is the truth table of the "OR" function?

How many inputs does it have?

TWO

How many outputs does it have?

ONE

What is the relationship between them?

OR

a	b	out
0	0	0
0	1	1
1	0	1
1	1	

What is the truth table of the "OR" function?

How many inputs does it have?

TWO

How many outputs does it have?

ONE

What is the relationship between them?

OR

a	**b**	**out**
0	**0**	**0**
0	**1**	**1**
1	**0**	**1**
1	**1**	**1**

What is the truth table of the "OR" function?

How many inputs does it have?

TWO

How many outputs does it have?

ONE

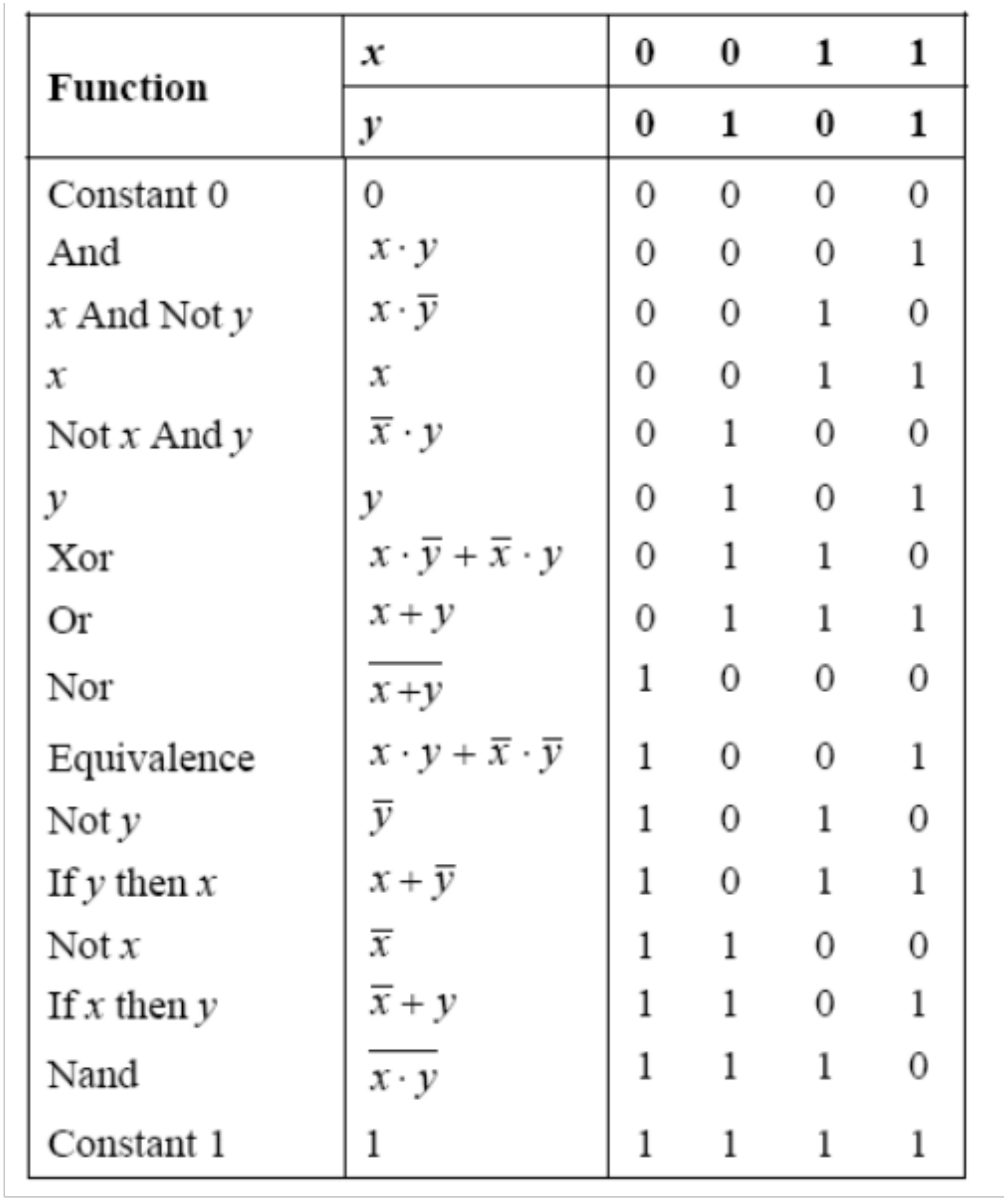
What is the relationship between them?

TRUE IF EITHER OR BOTH ARE TRUE, FALSE OTHERWISE

OR

a	b	out
0	0	0
0	1	1
1	0	1
1	1	1

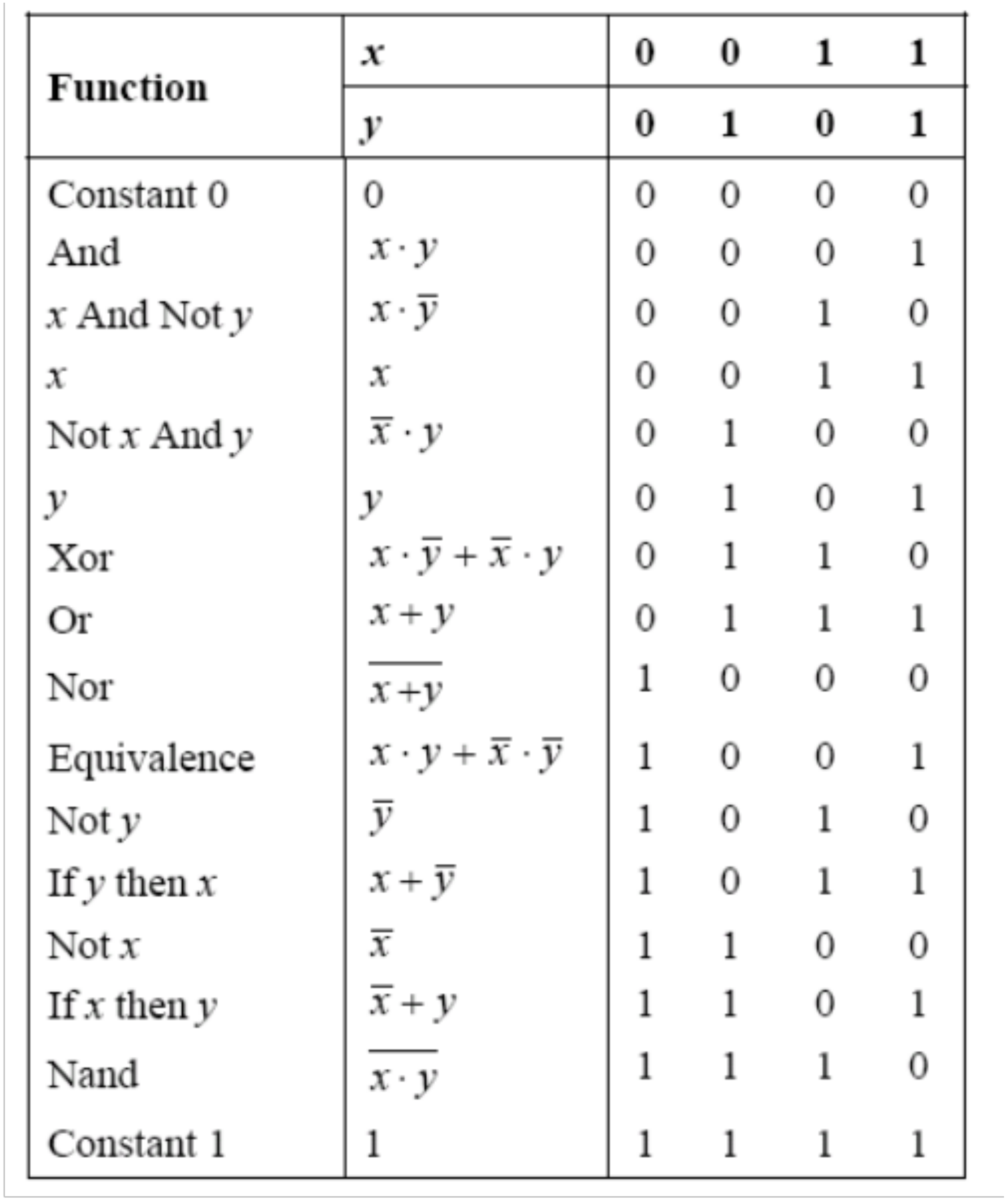
All Boolean functions of 2 variables



Function	x	0	0	1	1
	y	0	1	0	1
Constant 0	0	0	0	0	0
And	$x \cdot y$	0	0	0	1
x And Not y	$x \cdot \bar{y}$	0	0	1	0
x	x	0	0	1	1
Not x And y	$\bar{x} \cdot y$	0	1	0	0
y	y	0	1	0	1
Xor	$x \cdot \bar{y} + \bar{x} \cdot y$	0	1	1	0
Or	$x + y$	0	1	1	1
Nor	$\overline{x+y}$	1	0	0	0
Equivalence	$x \cdot y + \bar{x} \cdot \bar{y}$	1	0	0	1
Not y	$\bar{y}$	1	0	1	0
If y then x	$x + \bar{y}$	1	0	1	1
Not x	$\bar{x}$	1	1	0	0
If x then y	$\bar{x} + y$	1	1	0	1
Nand	$\overline{x \cdot y}$	1	1	1	0
Constant 1	1	1	1	1	1

All Boolean functions of 2 variables

What do you notice about the pattern?

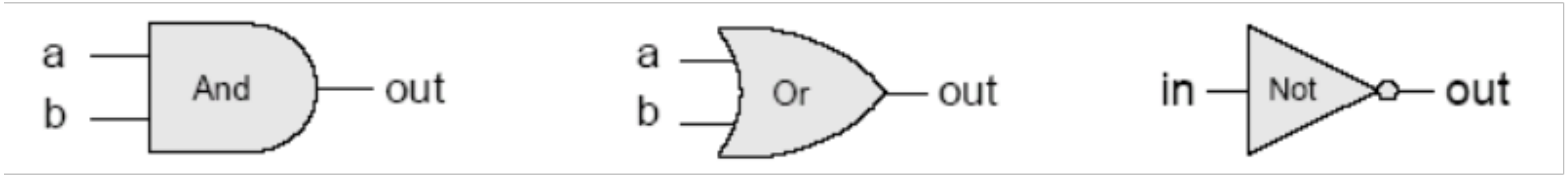


Function	x	**0**	**0**	**1**	**1**
	y	**0**	**1**	**0**	**1**
Constant 0	0	0	0	0	0
And	$x \cdot y$	0	0	0	1
x And Not y	$x \cdot \bar{y}$	0	0	1	0
x	x	0	0	1	1
Not x And y	$\bar{x} \cdot y$	0	1	0	0
y	y	0	1	0	1
Xor	$x \cdot \bar{y} + \bar{x} \cdot y$	0	1	1	0
Or	$x + y$	0	1	1	1
Nor	$\overline{x + y}$	1	0	0	0
Equivalence	$x \cdot y + \bar{x} \cdot \bar{y}$	1	0	0	1
Not y	$\bar{y}$	1	0	1	0
If y then x	$x + \bar{y}$	1	0	1	1
Not x	$\bar{x}$	1	1	0	0
If x then y	$\bar{x} + y$	1	1	0	1
Nand	$\overline{x \cdot y}$	1	1	1	0
Constant 1	1	1	1	1	1

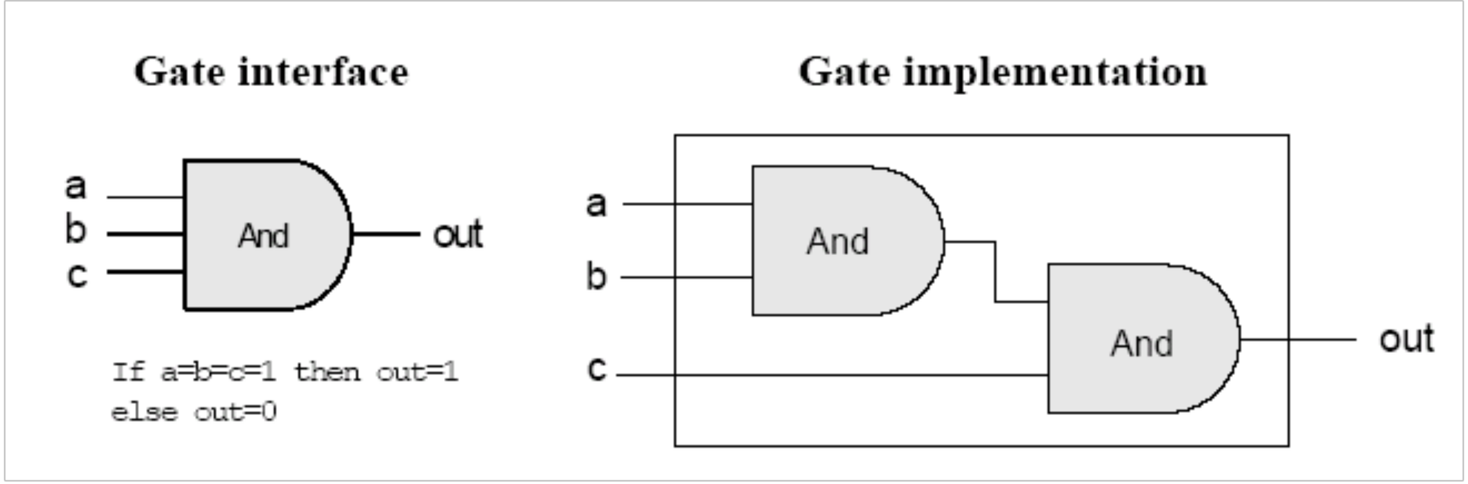
Gate logic - each function has its own "chip" inside the cpu

- Gate logic – a gate architecture designed to implement a Boolean function

- Elementary gates:



- Composite gates:



```
If a=b=c=1 then out=1
else out=0
```

- Important distinction: Interface (what) VS implementation (how).

CS E1, Section 1: number vocabulary

bit, byte, kilo-, mega-, giga-, tera-

Vocab

- BIT

Vocab

- BIT
    - a one or a zero

Vocab

- BIT
  - a one or a zero
- BYTE
  -

Vocab

- BIT
  - a one or a zero
- BYTE
  - eight bits: 0 0 0 0 0 0 0 0

Vocab

- BIT
  - a one or a zero
- BYTE
  - eight bits: 0 0 0 0 0 0 0 0
  - can store 2^8 numbers:

Vocab

- BIT
    - a one or a zero
- BYTE
    - eight bits: 0 0 0 0 0 0 0 0
    - can store 2^8 numbers: 0-255
- KILOBYTE
    -

Vocab

- BIT
    - a one or a zero
- BYTE
    - eight bits: 0 0 0 0 0 0 0 0
    - can store 2^8 numbers: 0-255
- KILOBYTE
    - ~1000 bytes

Vocab

- BIT
  - a one or a zero
- BYTE
  - eight bits: 0 0 0 0 0 0 0 0
  - can store 2^8 numbers: 0-255
  - stores one character
- KILOBYTE
  - ~1000 bytes
  - actually 2^10 numbers: 0-1023
  - stores less than eight twitter posts

Vocab

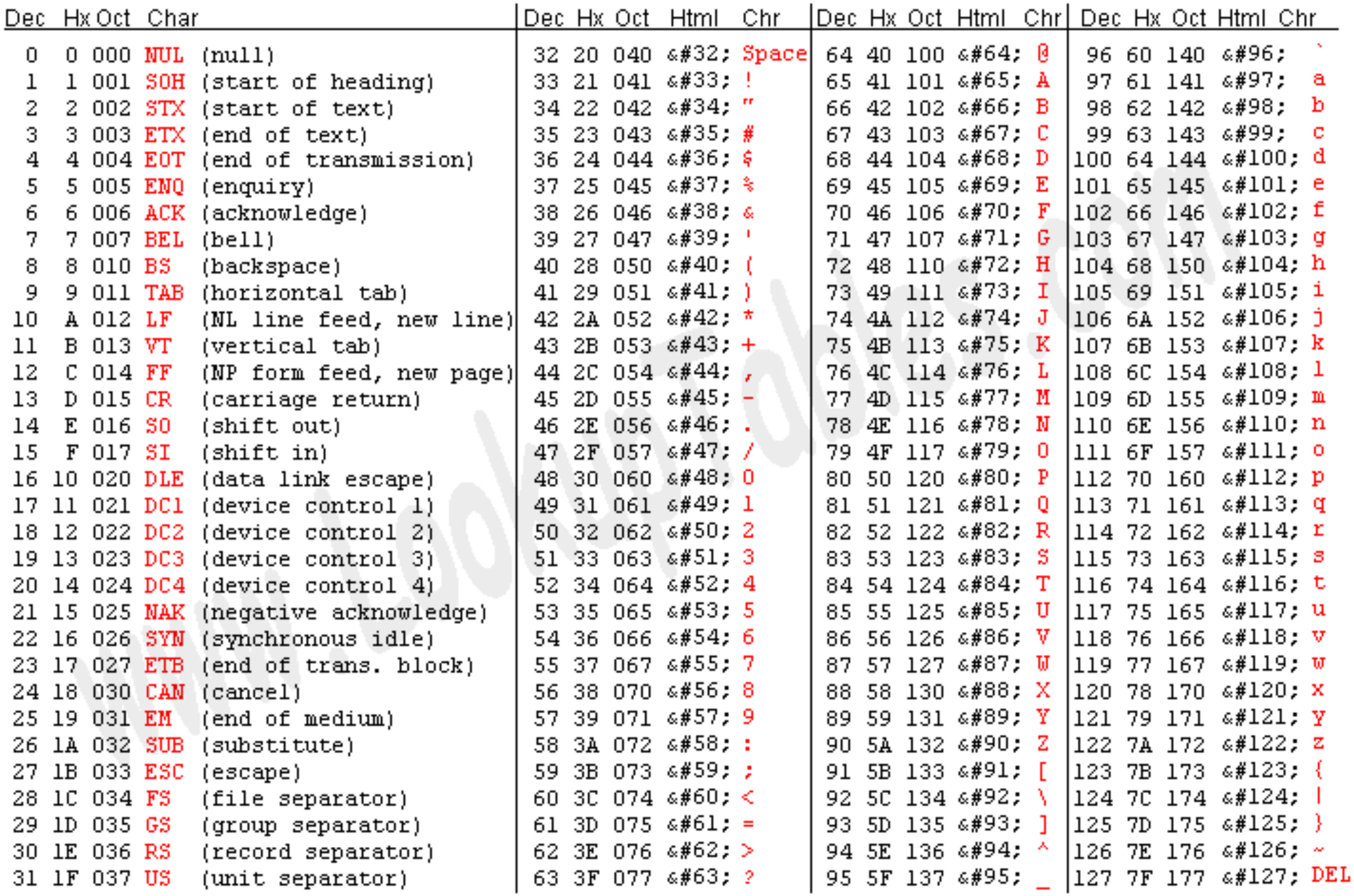
- BIT
  - a one or a zero
- BYTE
  - eight bits: 0 0 0 0 0 0 0 0
  - can store 2^8 numbers: 0-255
  - stores one character
- KILOBYTE
  - ~1000 bytes
  - actually 2^10 numbers: 0-1023
  - stores less than eight twitter posts
- MEGABYTE
  - ~1,000,000 bytes
  - actually 2^30 numbers: 0-1,048,575 - less than 1/3 of a typical mp3 song

Vocab

- http://en.wikipedia.org/wiki/Megabyte

CS E1, Section 1: ascii

American Standard Code for Information Interchange



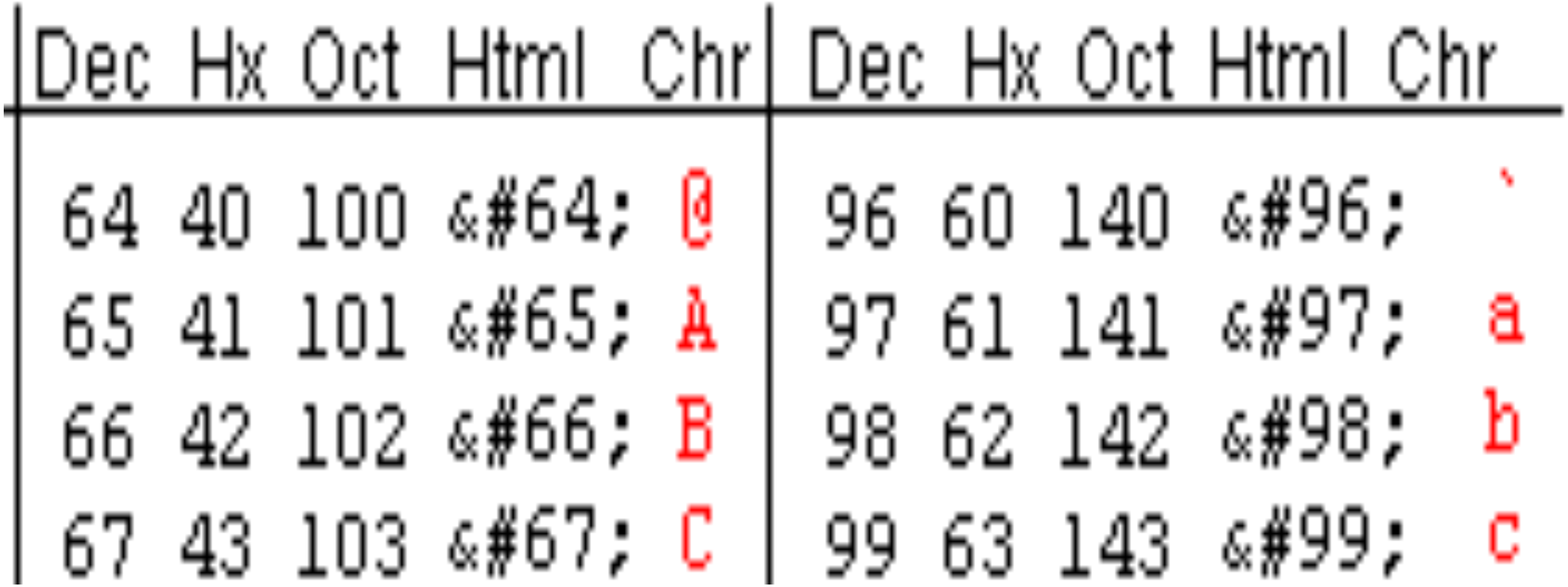
Dec	Hx	Oct	Char		Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL	(null)	32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
1	1	001	SOH	(start of heading)	33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
2	2	002	STX	(start of text)	34	22	042	&#34;	"	66	42	102	&#66;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

Source: www.LookupTables.com

CS E1, Section 1: ascii

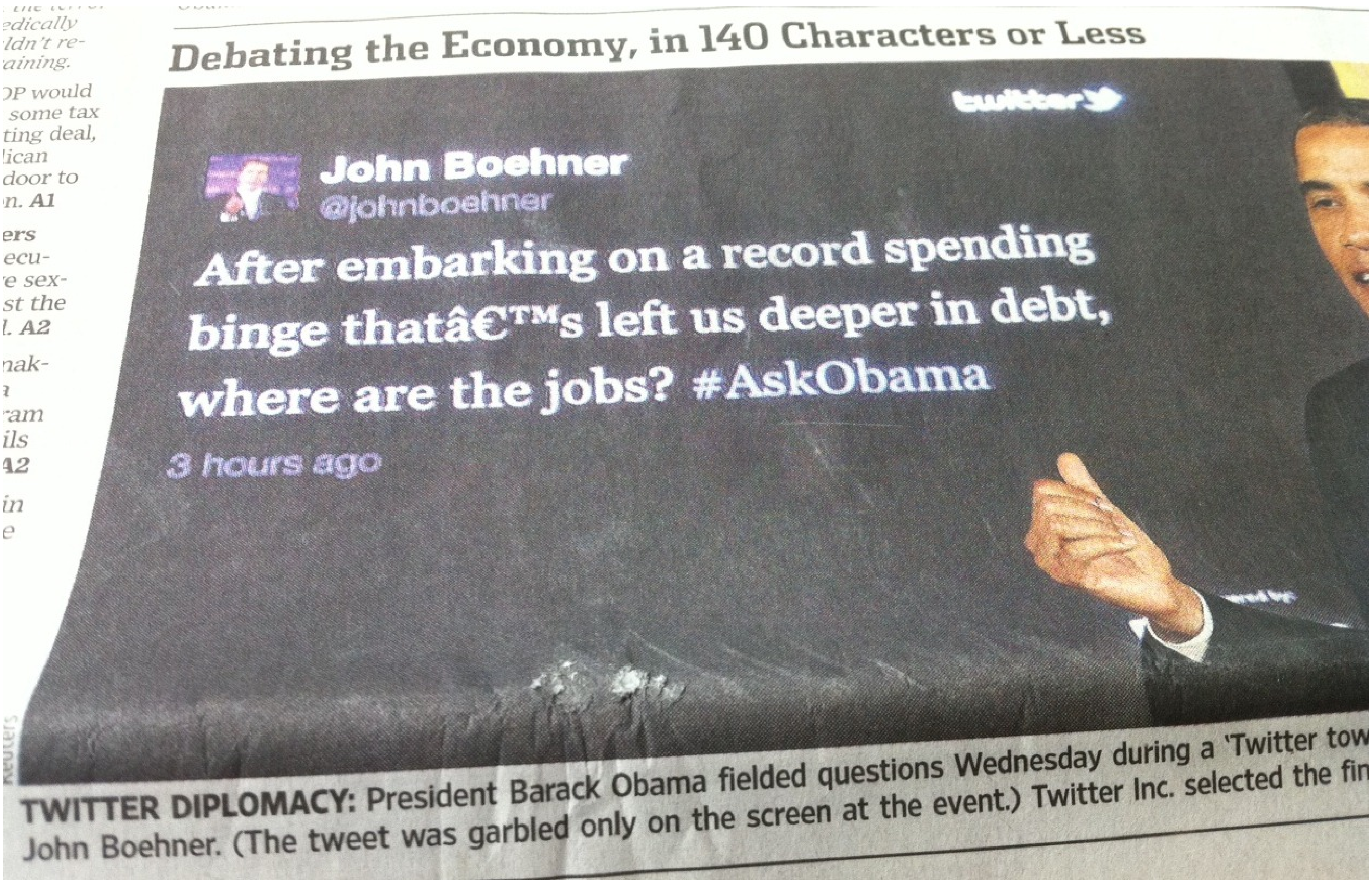
Ascii numbers to remember:

65 is capital 'A'
97 is lowercase 'a'



Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
64	40	100	&#64;	@	96	60	140	&#96;	`
65	41	101	&#65;	A	97	61	141	&#97;	a
66	42	102	&#66;	B	98	62	142	&#98;	b
67	43	103	&#67;	C	99	63	143	&#99;	c

Source, explanation: link



edically
ldn't re-
aining.

OP would
some tax
ting deal,
lican
door to
n. A1

ers
ecu-
e sex-
st the
l. A2

nak-
a
ram
ils
A2

in
e

Debating the Economy, in 140 Characters or Less

John Boehner
@johnboehner

After embarking on a record spending binge thatâ€™s left us deeper in debt, where are the jobs? #AskObama

3 hours ago

TWITTER DIPLOMACY: President Barack Obama fielded questions Wednesday during a 'Twitter tow
John Boehner. (The tweet was garbled only on the screen at the event.) Twitter Inc. selected the fin

https://docs.google.com/spreadsheet/ccc?key=0ApKUsT5wTHHgdHpDNkEtOUlDUUNkeVF4cmlhSlVXYVE&hl=en_US

example: ascii in a spreadsheet