Computer Architecture

A DRIVE-BY …

Fundamentals

Yes, you know what a computer is … but how well and at what level?

Understanding the basics of how things work relate to why we make certain architectural choices

Main components

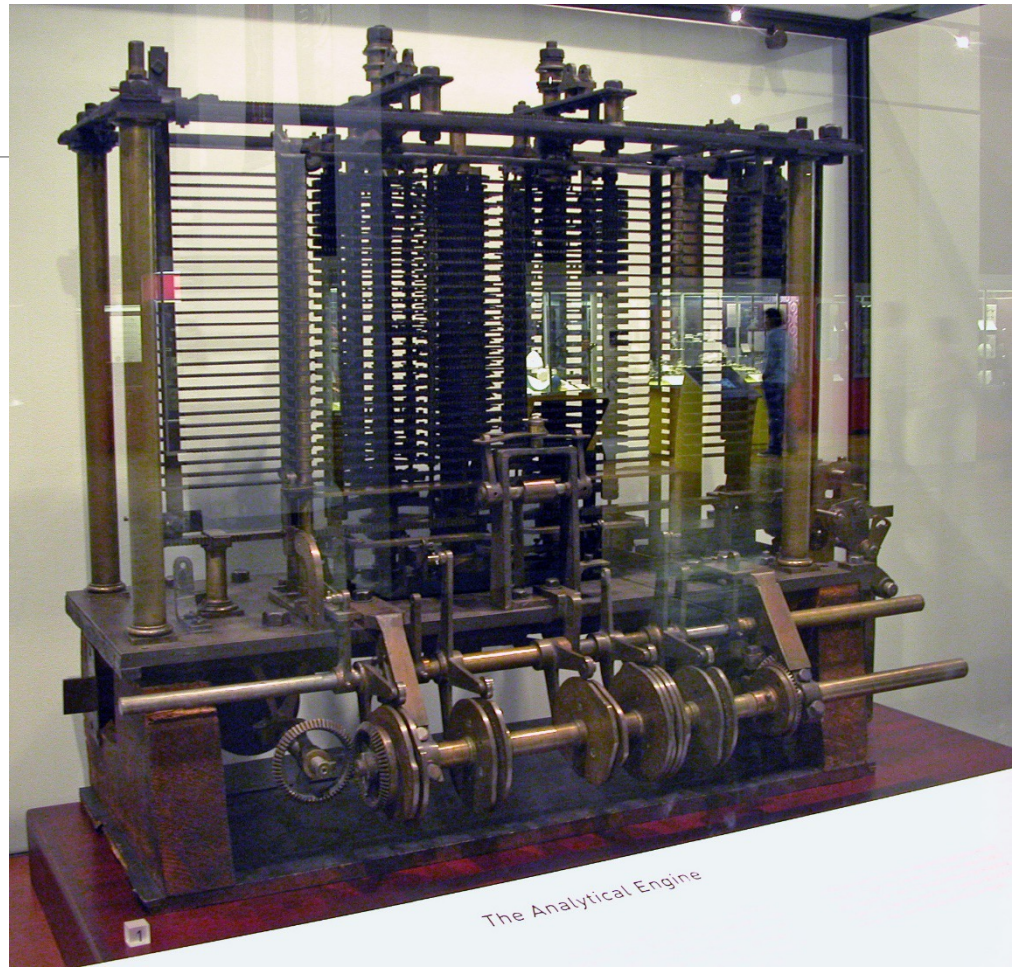
- CPU
- Memory
- Storage
- Bus

Where it began …

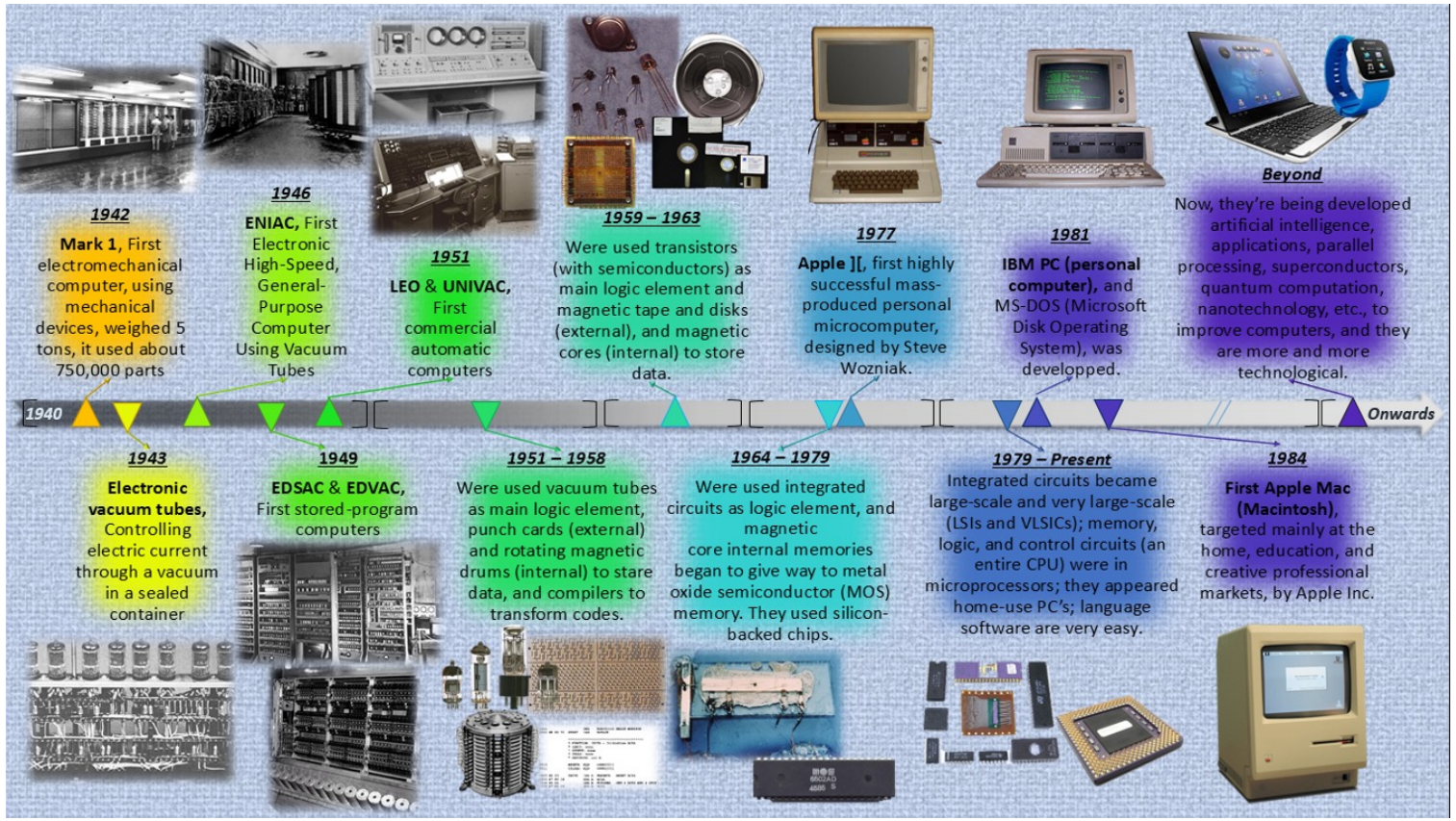
Charles Babbage and Ada Lovelace (circa 1800s)

- Babbage devises his ‘analytic engine’. A mechanical computer with an Arithmetic Control Unit, Flow control, Branching, Loops, Memory…
- Ada Lovelace contributed the ideas and concepts which became a general programming language for the analytic engine. (Ever heard of the programming language ‘Ada’??)

All analog and mechanical; no electronic components!



Ancestry.computer …



ENIAC: The first electronic computer. Completed in 1946 at the University of Pennsylvania.

EDVAC: von Neumann architecture

ICs (Integrated circuits aka 'chips') didn't show up until the 60s!

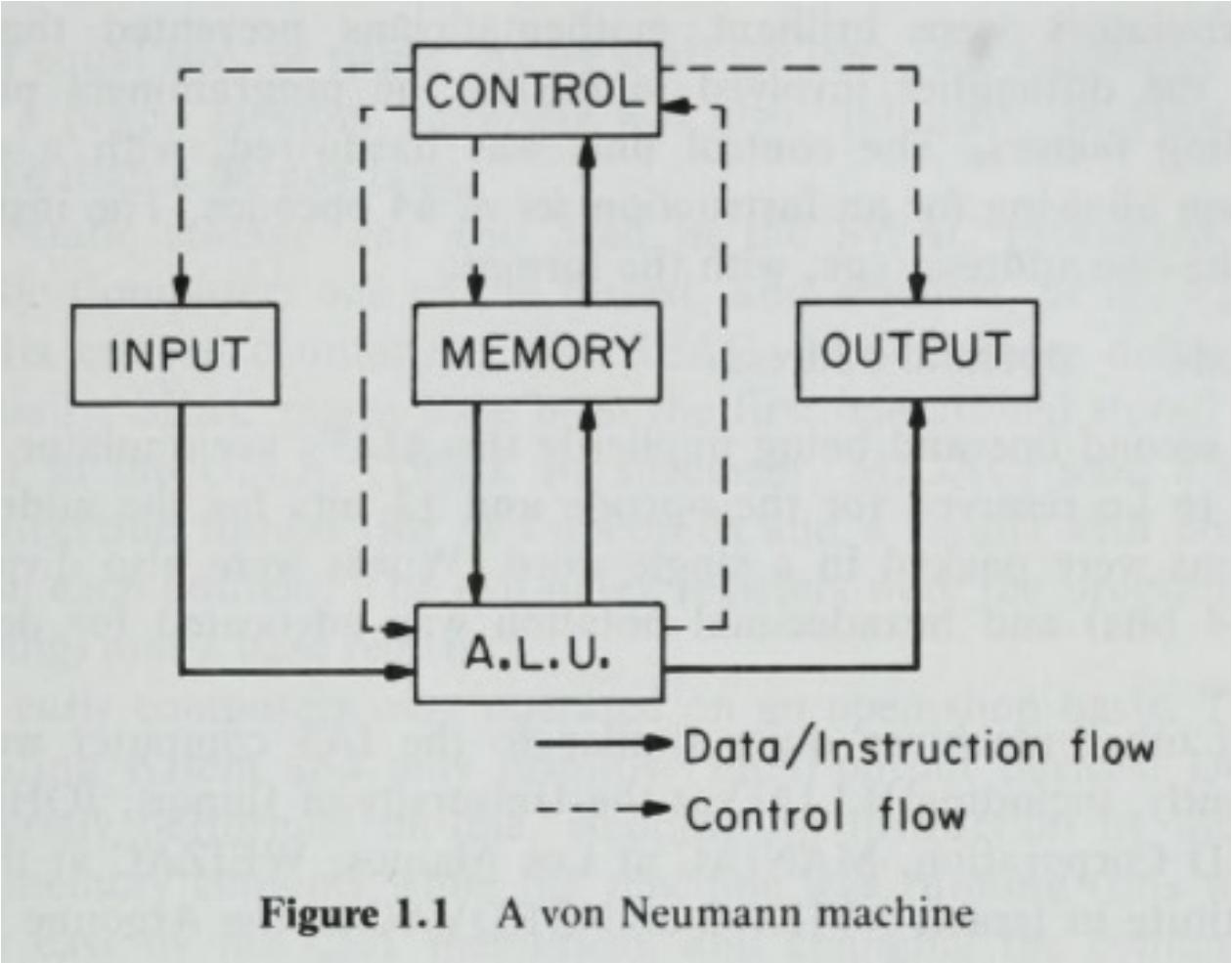
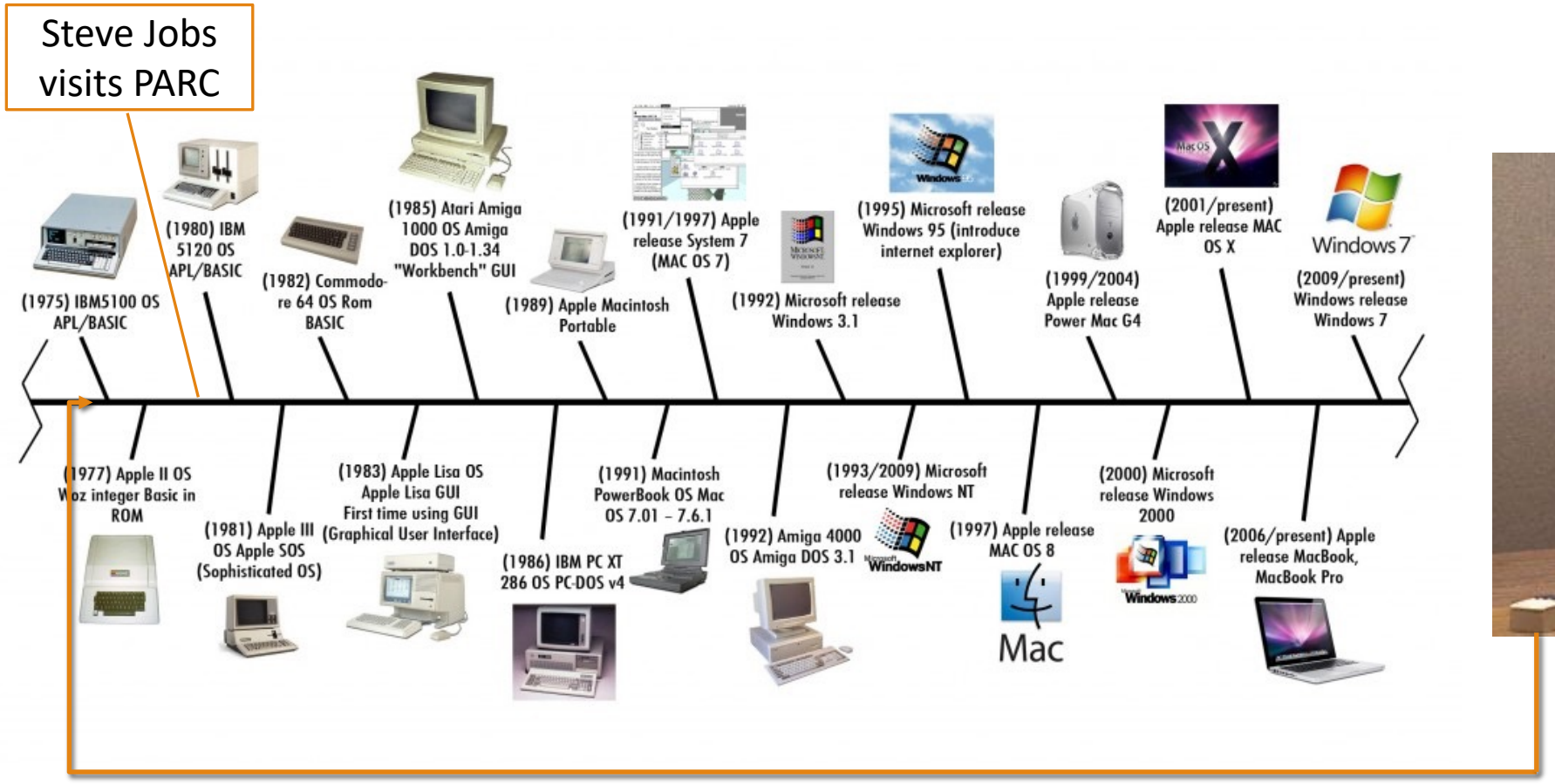
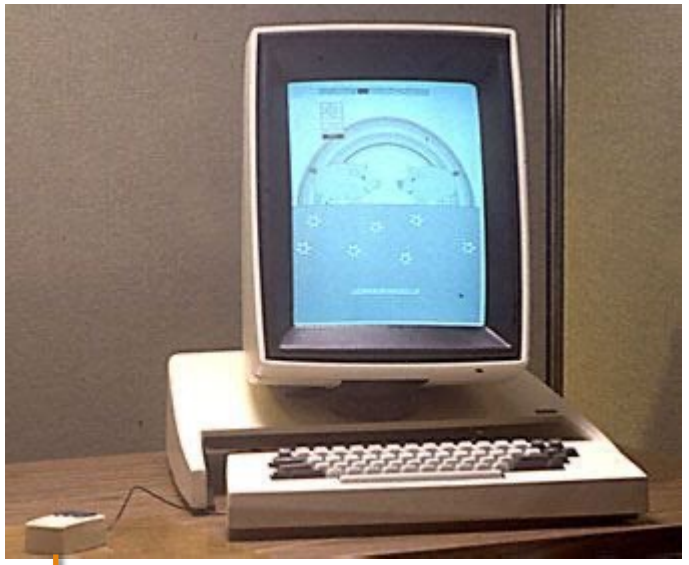


Figure 1.1 A von Neumann machine

J. Baer, Computer Systems Architecture, Computer Science Press, p. 5, 1980

Ancestry.computer … for OS





Xerox PARC Alto (1973!)

The evolution

Computers did not always look the way they do now

- In fact they were once base-10, not base-2
- EBCDIC was used instead of ASCII
- The original goal was just crunch numbers (math) very very fast (e.g. Alan Turing's Enigma project)

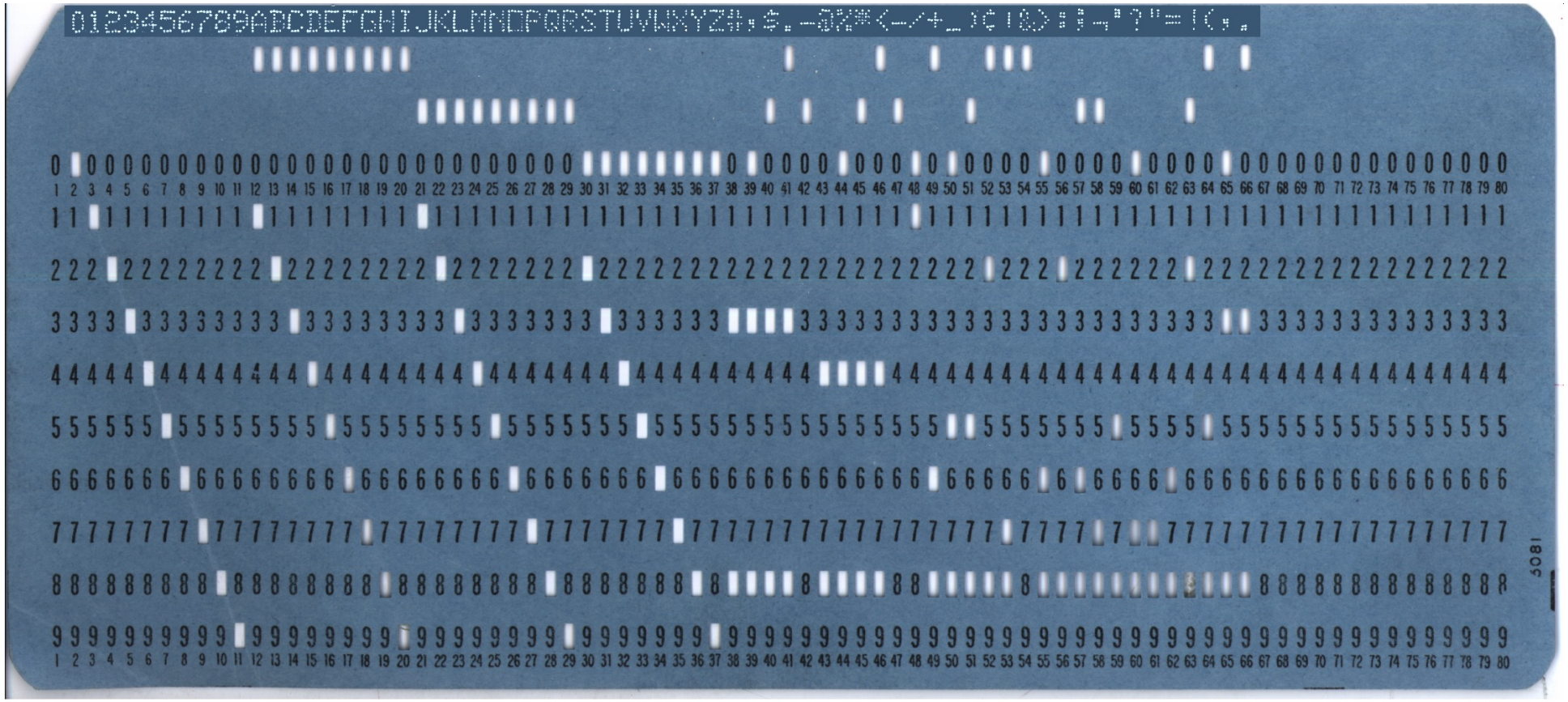
The standard today, is binary, high density chips, multiprocessing

The future?

- Quantum Computing (https://www.ibm.com/quantum-computing/learn/what-is-quantum-computing)

- Neuromorphic Computing (R. A. Nawrocki, R. M. Voyles and S. E. Shaheen, "A Mini Review of Neuromorphic Architectures and Implementations," in IEEE Transactions on Electron Devices, vol. 63, no. 10, pp. 3819-3829, Oct. 2016, doi: 10.1109/TED.2016.2598413)

BCD – Binary Coded Decimal



History of Punched Cards: https://homepage.divms.uiowa.edu/~jones/cards/codes.html

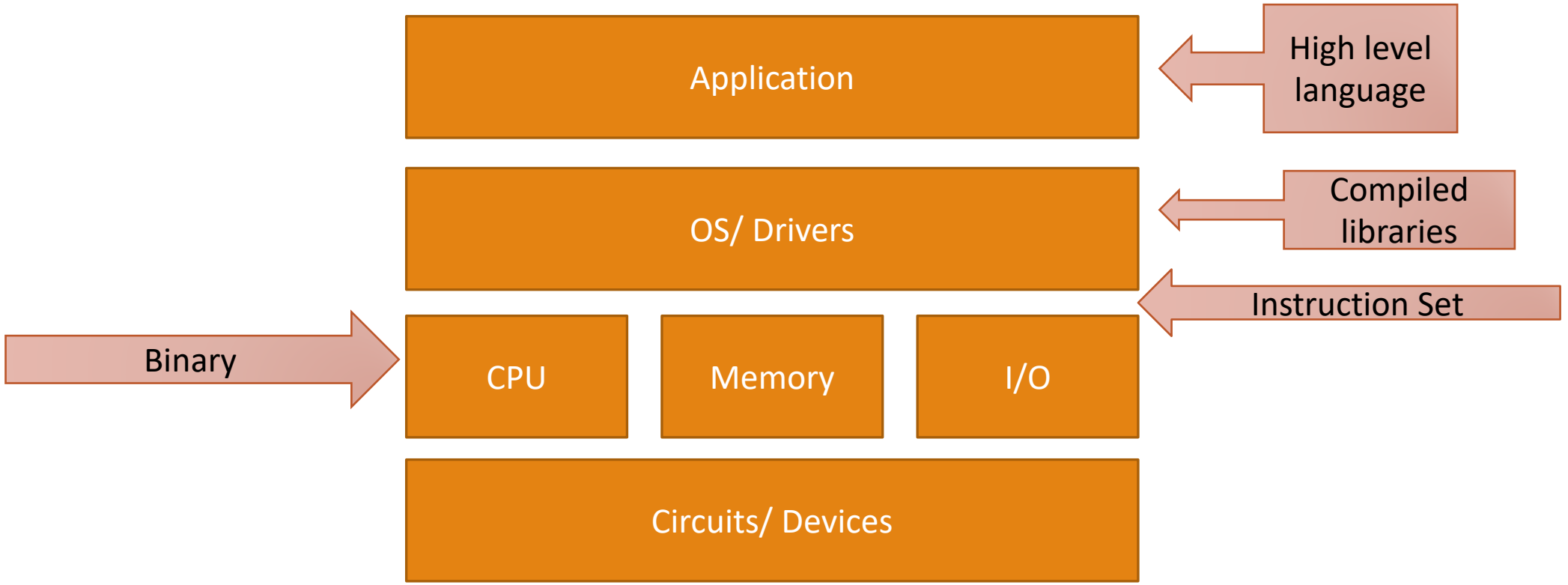
EBCDIC vs ASCII

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX		PT			GE				FF	CR		
1	DLE	SBA	EUA	1C		NL				EM			DUP	SF	FM	ITB
2							ETB	ESC						ENQ		
3			SYN					EOT					RA	NAK		
4	SP										¢	.	<	(	+	
5	&										!	$	*	)	;	¬
6	-	/									|	,	%	_	>	?
7											:	#	@	'	=	"
8		a	B	c	d	e	f	g	h	I						
9		j	K	l	m	n	o	p	q	r						
A		~	S	t	u	v	w	x	y	z						
B																
C	{	A	B	C	D	E	F	G	H	I						
D	}	J	K	L	M	N	O	P	Q	R						
E	\		S	T	U	V	W	X	Y	Z						
F	0	1	2	3	4	5	6	7	8	9						

most significant nibble

least significant nibble	0_	1_	2_	3_	4_	5_	6_	7_
_0	NUL	DLE	SP	0	@	P	'	p
_1	SOH	DC1	!	1	A	Q	a	q
_2	STX	DC2	"	2	B	R	b	r
_3	ETX	DC3	#	3	C	S	c	s
_4	EOT	DC4	$	4	D	T	d	t
_5	ENQ	NAK	%	5	E	U	e	u
_6	ACK	SYN	&	6	F	V	f	v
_7	BEL	ETB	'	7	G	W	g	w
_8	BS	CAN	(	8	H	X	h	x
_9	HT	EM	)	9	I	Y	i	y
_A	LF	SUB	*	:	J	Z	j	z
_B	VT	ESC	+	;	K	[	K	}
_C	FF	FS	,	<	L	\	l	|
_D	CR	GS	-	=	M	]	m	}
_E	SO	RS	.	>	N	^	n	~
_F	SI	US	/	?	O	_	o	DEL

But today: Your software on a computer … looks like this



From code to CPU

We all know that CPUs run architecture specific instruction sets

Compilers take programming languages and convert to assembly and then binary

X86, ARM, PowerPC all use different instruction sets (with similar behavior)

8085 programming item	Description
Registers	These are 8 bit general purpose registers such as B,C,D,E,H,L as mentioned.
Accumulator	It is a 8 bit register used to store the result as well as intermediate operations of the mathematical operation.
Flags	These are five flags , set and reset according to certain arithmetic and logical conditions.
Program Counter(PC)	It is 16 bit in size. The microprocessor uses this register to sequence the executions of the instructions. It is also used as memory pointer. It holds memory address from where next byte is to be fetched. It basically points to the assembly codes stored in the form of opcodes. When a byte is fetched , PC is incremented by 1 to point to the next memory location.
Stack Pointer (SP)	It is 16 bit in size. It points to the memory location in R/W memory known as stack. The beginning of stack is defined by loading 16 bit address in the SP.

From code to CPU

x86 examples

```
MVI D, 8CH
MVI C, 6EH
MOV A, C
ADD D
OUT PORT1
HLT
```

What does this do?

Adds x8C and x6E = xFA (250)

```
MVI A, 40H
RLC
RLC
RLC
OUT PORT1
HLT
```

What does this do?

Multiply by 8 (shift is x2)

From there …

CPUs are circuitry

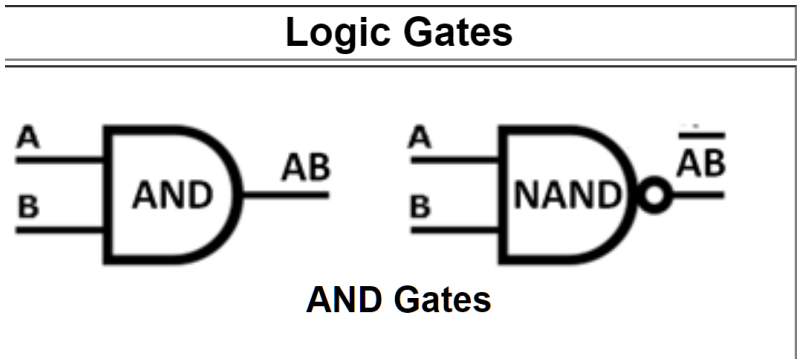
Transistors are the foundation of circuitry

Computers are digital logic; logic gates built from transistors

CPUs are made of millions of (transistors/ logic gates)

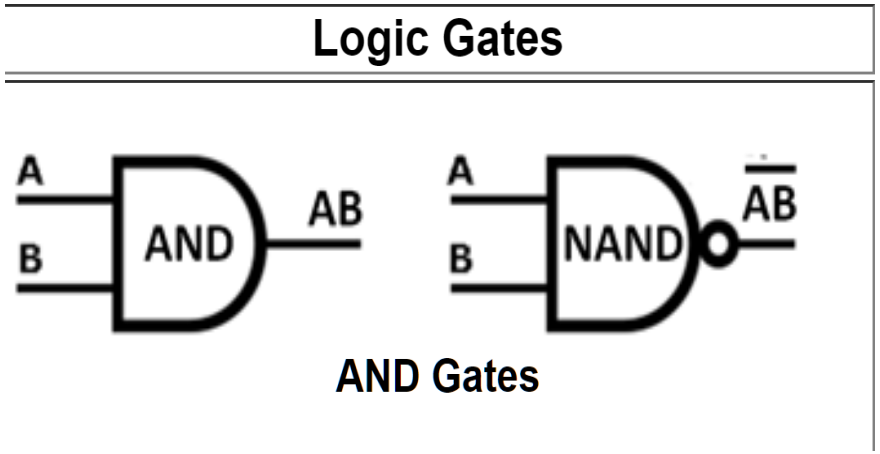
Binary instructions operate on these logic gates

What is a logic gate?
AND, NAND, OR, NOR, NOT, XOR …



Concept behind Logic … truth tables

	A	B	AB	$\overline{AB}$
Row1	0	0	0	1
Row2	0	1	0	1
Row3	1	0	0	1
Row4	1	1	1	0



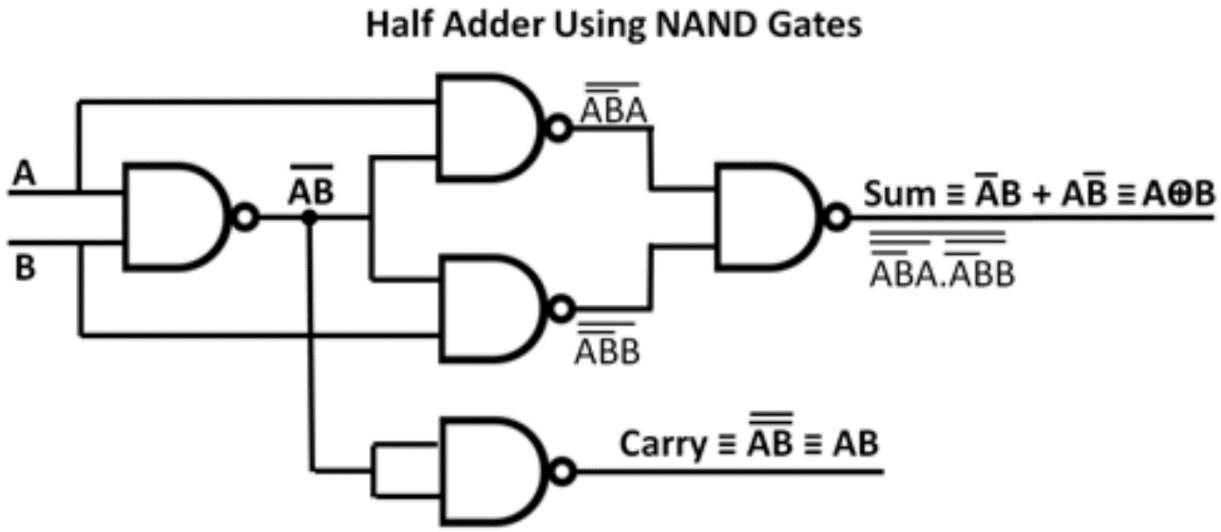
What is the Software statement that corresponds to the logic gate diagram for output AB?

if (A && B) { …}

What is the Software statement that corresponds to the logic gate diagram for output $\overline{AB}$

if ! (A && B) { …}

All math operations subset to digital logic



Subtraction is also addition in digital logic

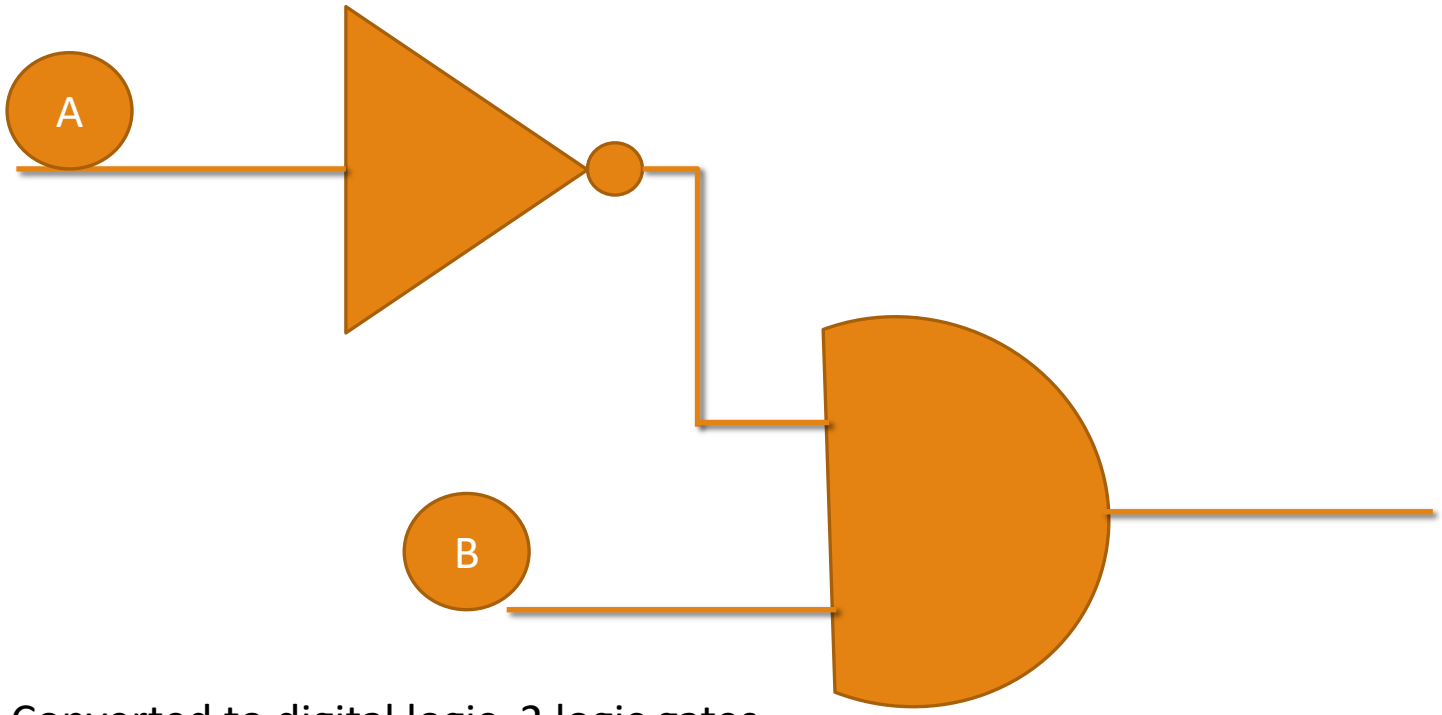
- **Subtraction**

To subtract binary number B (the subtrahend) from binary number A (the minuend).

- Add the 2's complement of B to A
- If there is a final carry bit, discard it and the result is positive.
- If B greater than A there will be no carry bit and the result will be the 2's complement of the sum and is negative.

A simple ‘if’ statement

```
if (a == 0 and b != 0)
```



(MOSTLY CORRECT) ASSEMBLY CODE
&a = 0x5100; & b = 0x5200

```
LDA 0x5100
CMP 0
JZ SKIP
LDA 0x5200
JZ SKIP
BLAH, BLAH
SKIP: OTHER STUFF
```

Converted to digital logic, 2 logic gates
Roughly 8 transistors (2-5 per gate)
One simple line of code needs 8 transistors
Linux (just the kernel) has 10,000,000 lines of code

Reading material

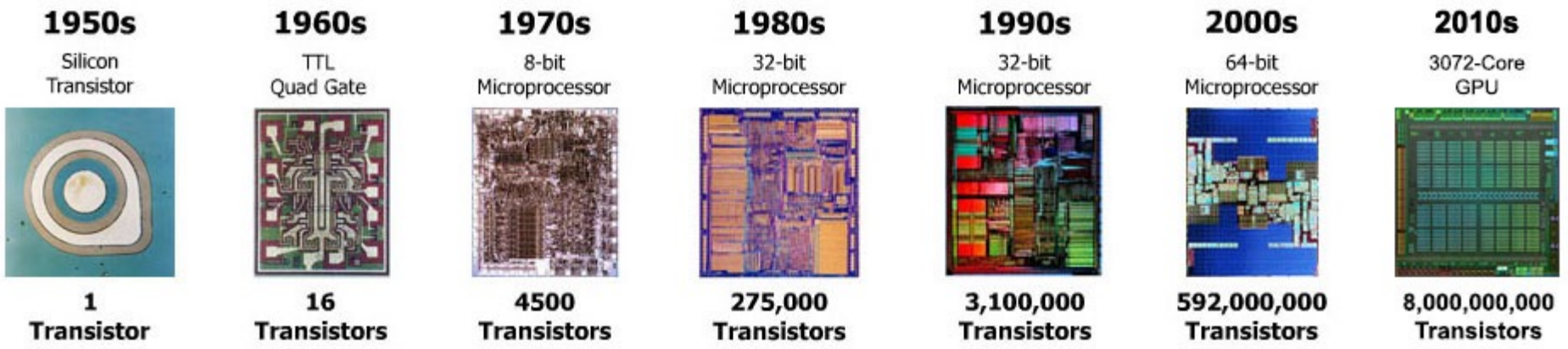
Digital Logic:

https://mpoweruk.com/digital_logic.htm

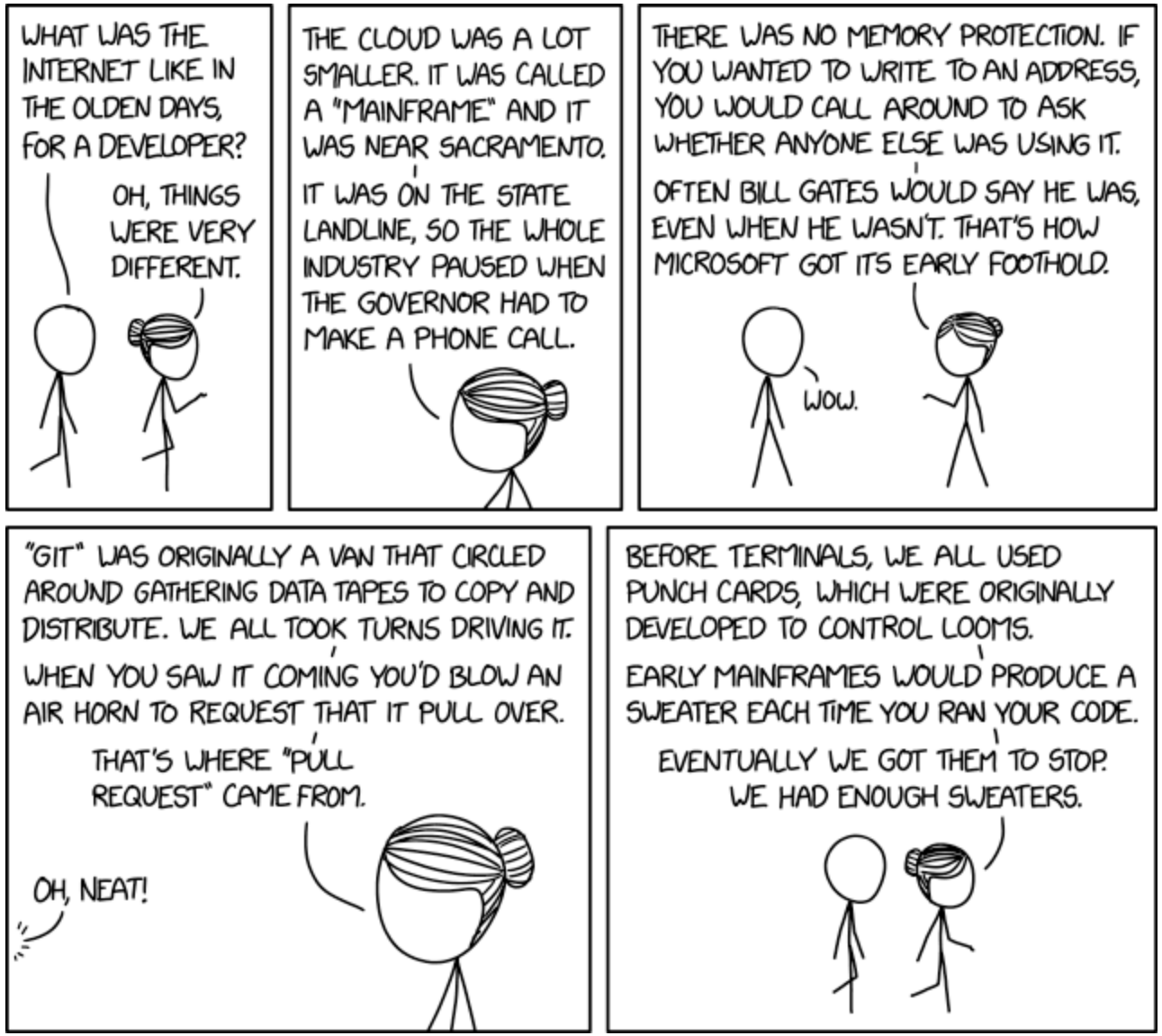
Digital logic gates simulator:

https://www.kolls.net/gatesim/

And this is why Moore’s law is important!



Always know your history so you can build the future



https://xkcd.com/2324/