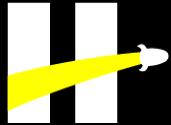
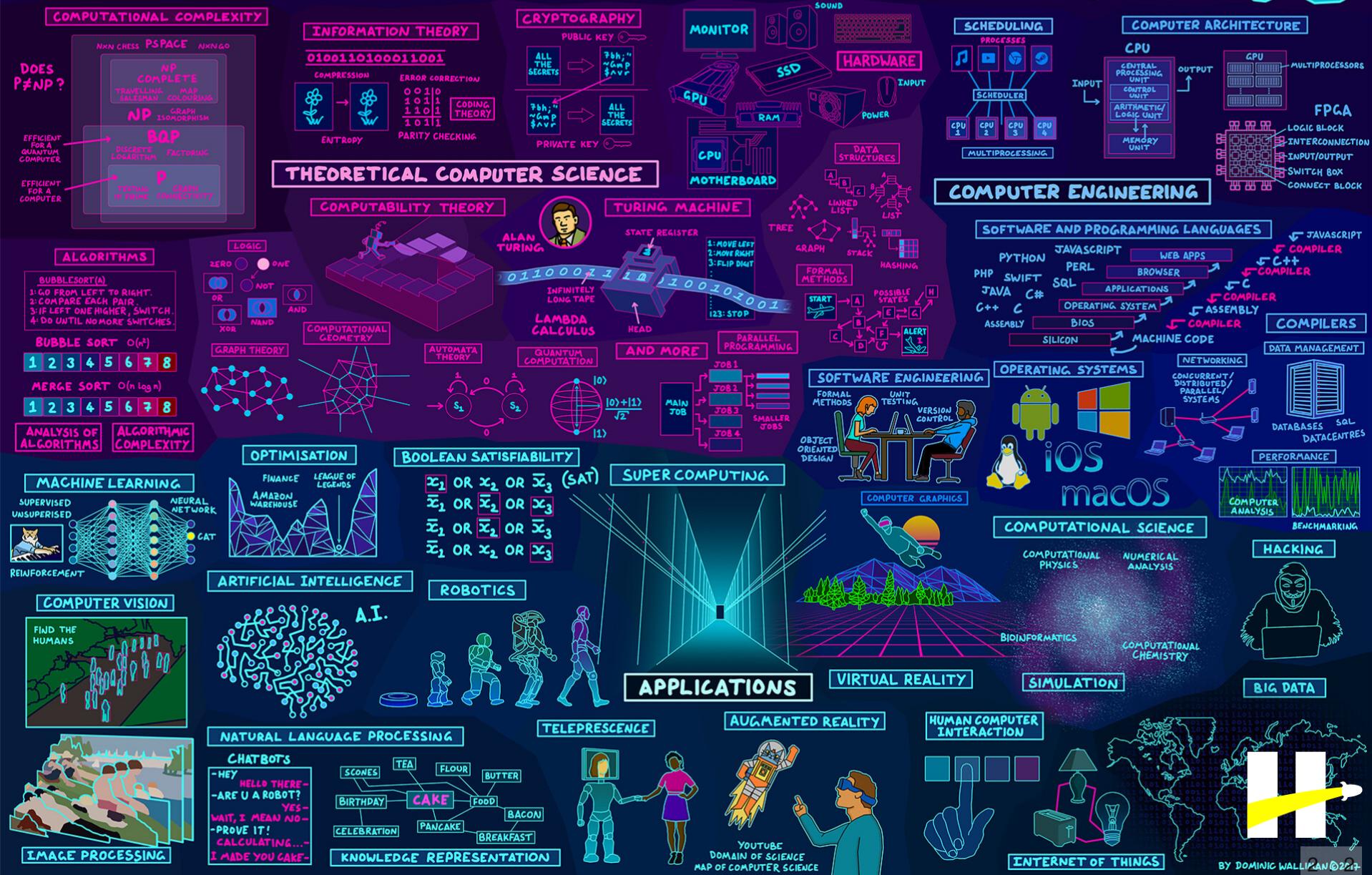


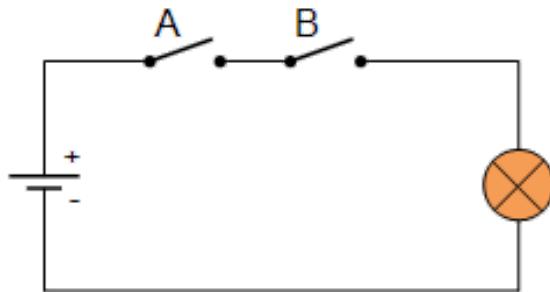
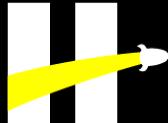
HENRY



# Intro to CS

# MAP OF COMPUTER SCIENCE

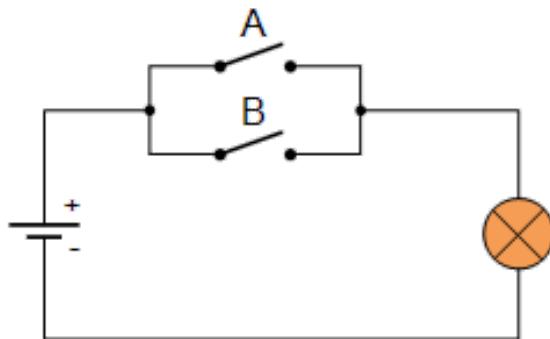




Lamp - ON = "1"  
Lamp - OFF = "0"

Switch A - Open = "0", Closed = "1"

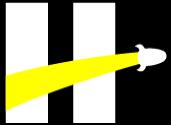
Switch B - Open = "0", Closed = "1"



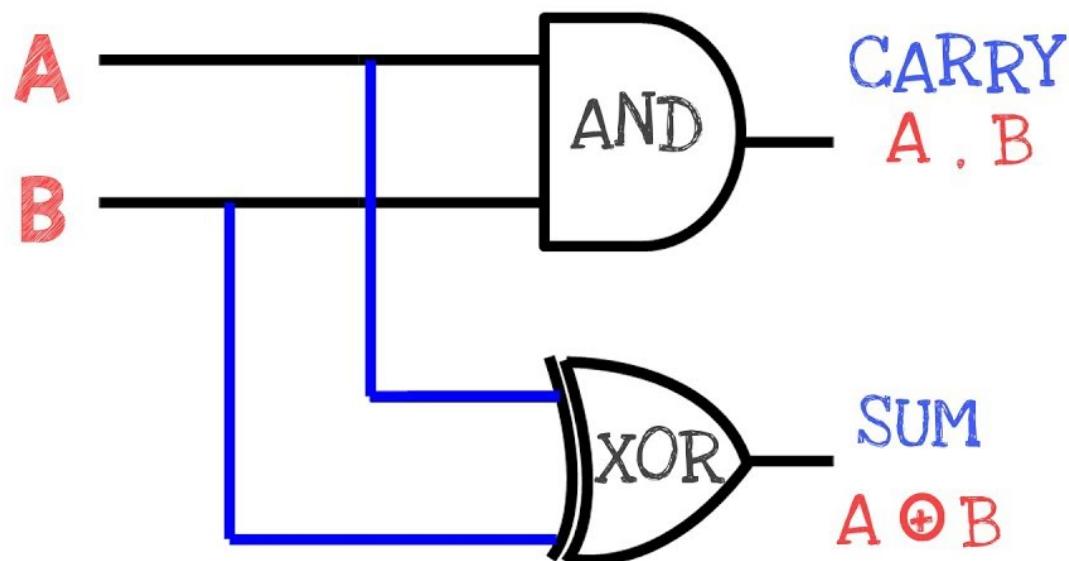
Lamp - ON = "1"  
Lamp - OFF = "0"

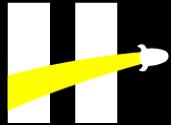
Switch A - Open = "0", Closed = "1"

Switch B - Open = "0", Closed = "1"

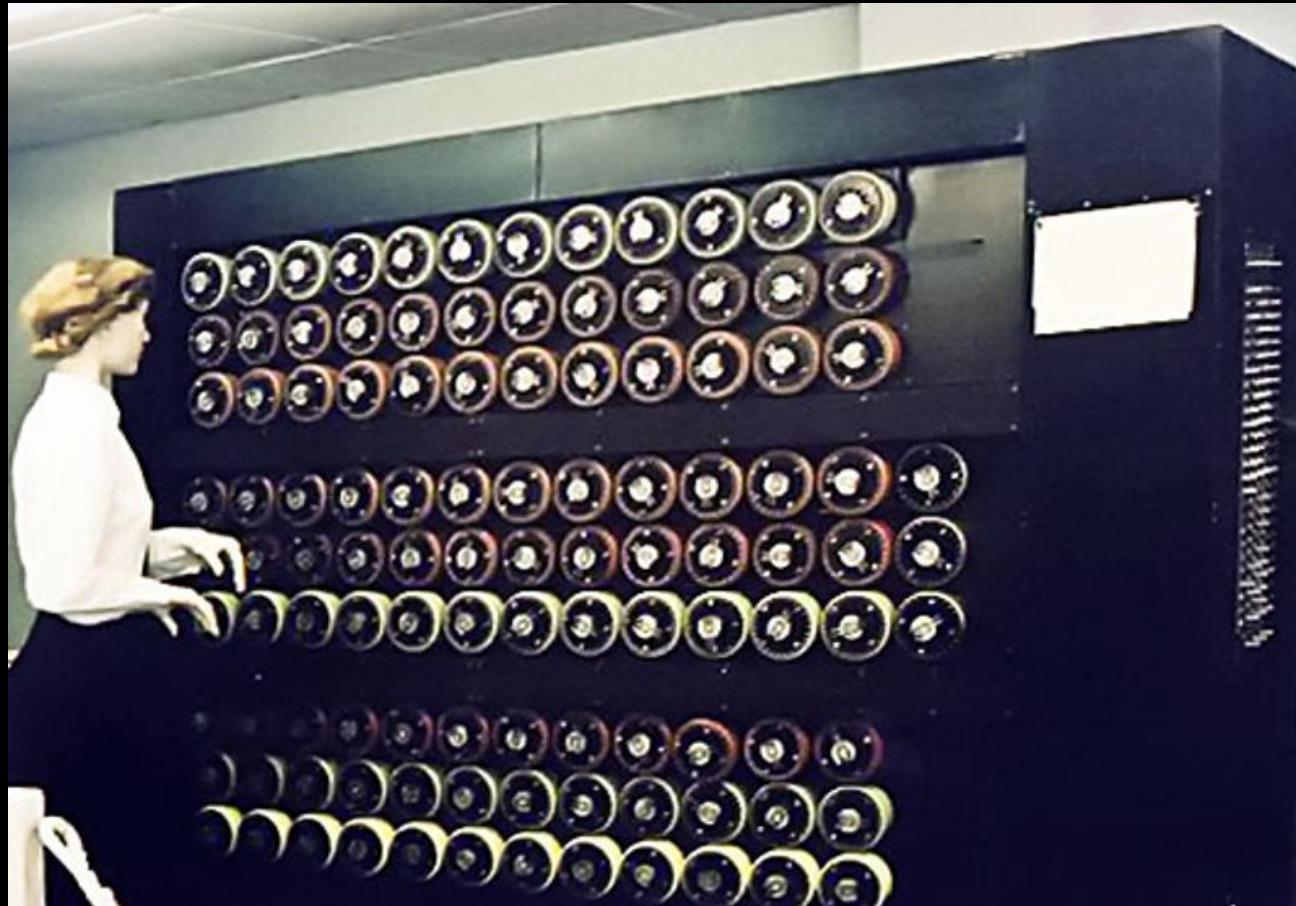
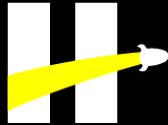


# Half Adder

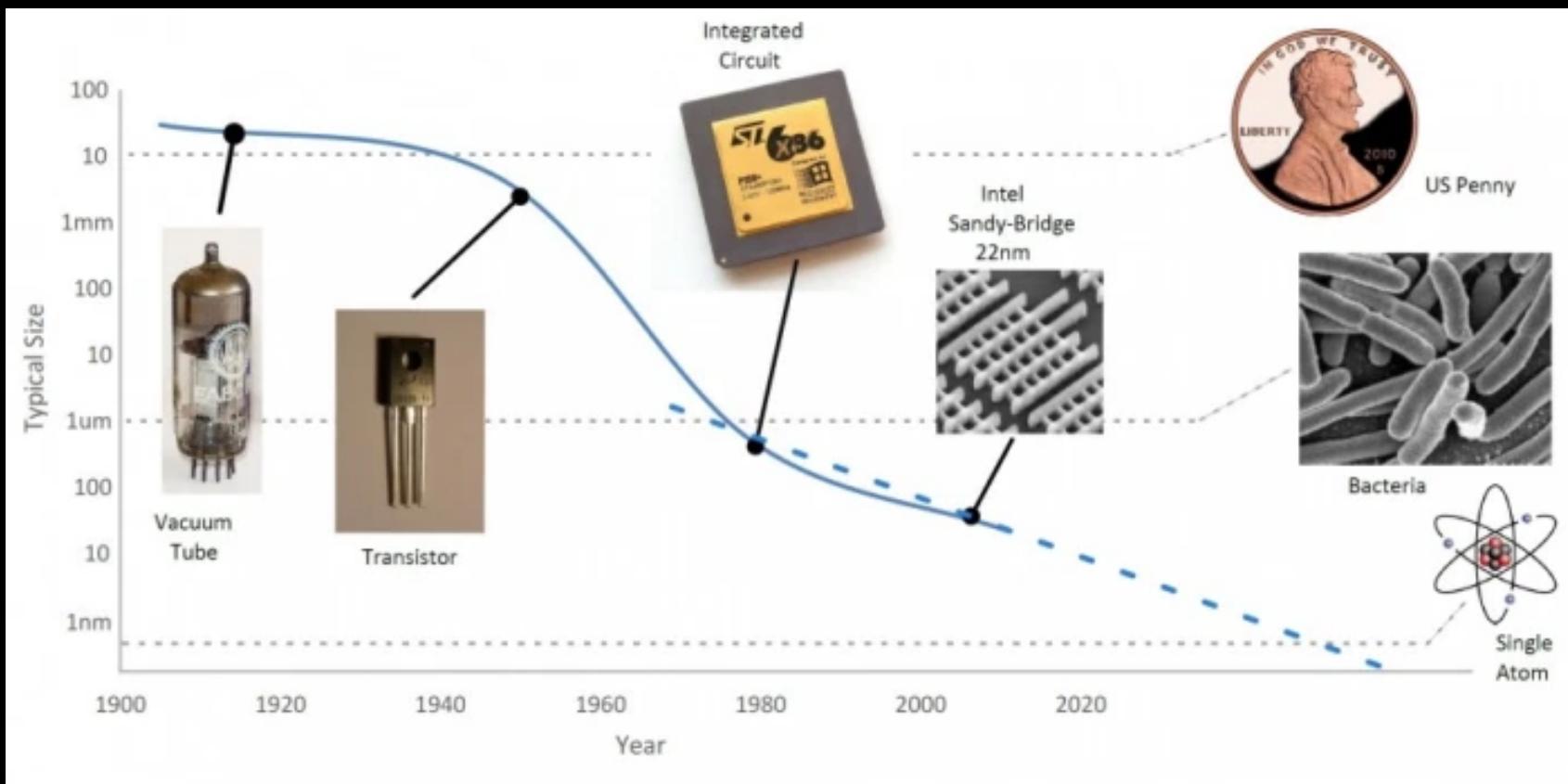
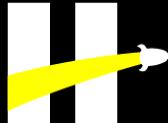




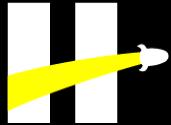
1937 - Adder



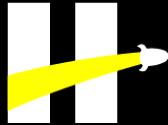
1937 - British Bombe  
Alan Turing



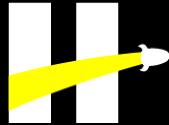
Ley de Moore.



# Números Binarios



```
1 // Sistema UNARIO
2
3 I I I I I I I I
4
5 // utiliza un sólo tipo de símbolo. Su desventaja es que no permite simbolizar
6 // cómoda y rápidamente conjuntos con muchos elementos.
7
8 // Numeros ROMANOS
9
10 // Parecido al Unario, pero disponemos de más simbolos
11 // y operaciones implicitas
12
13 CXVII = cien + diez + cinco + uno + uno
14
15 MCMV = mil + (mil - cien) + cinco
16
17 // SISTEMAS POSICIONALES
18
19 // En estos sistemas, cada símbolo, además del número de unidades que representa
20 // considerado en forma aislada, tiene un significado o peso distinto
21 // según la posición que ocupa en el grupo de caracteres del que forma parte.
```



POSICIÓN

7<sup>a</sup>

6<sup>a</sup>

5<sup>a</sup>

4<sup>a</sup>

3<sup>a</sup>

2<sup>a</sup>

1<sup>a</sup>

Unidad

Decena

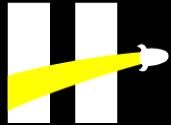
Centena

Unidad de Millar

Decena de Millar

Centena de Millar

Unidad de Millón



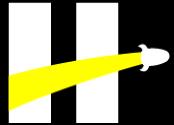
# Símbolos del sistema

Cada sistema utiliza sus propios símbolos:

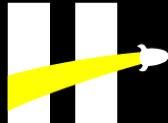
- Decimal: 0, 1, 2, 3, 4, 5, 6, 7, 8 y 9.
- Binario: 0 y 1.
- Quinario: 0, 1, 2, 3 y 4.
- Octal: 0, 1, 2, 3, 4, 5, 6 y 7.
- Hexadecimal: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E y F.

$$N = \sum_{i=-k}^{n-1} d_i \cdot 10^i$$

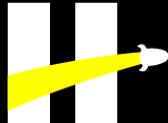
$$N = \sum_{i=-k}^n d_i \cdot 2^i$$



# Representaciones



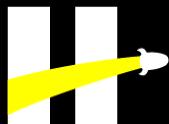
Physical State	OFF	ON	ON	OFF	OFF	ON	ON	OFF
Binary Notation	0	1	1	0	0	1	1	0
Order of Magnitude	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
Decimal Value	128	64	32	16	8	4	2	1
Applicable Value	0	64	32	0	0	4	2	0
Total Decimal Value	$102 = 64 + 32 + 4 + 2$							



# ASCII

↗

Binary	Decimal	Glyph	Binary	Decimal	Glyph
0010 1110	46	.	0011 1010	58	:
0010 1111	47	/	0011 1011	59	;
0011 0000	48	o	0011 1100	60	<
0011 0001	49	I	0011 1101	61	=
0011 0010	50	2	0011 1110	62	>
0011 0011	51	3	0011 1111	63	?
0011 0100	52	4	0100 0000	64	@
0011 0101	53	5	0100 0001	65	A
0011 0110	54	6	0100 0010	66	B
0011 0111	55	7	0100 0011	67	C
0011 1000	56	8	0100 0100	68	D
0011 1001	57	9	0100 0101	69	E

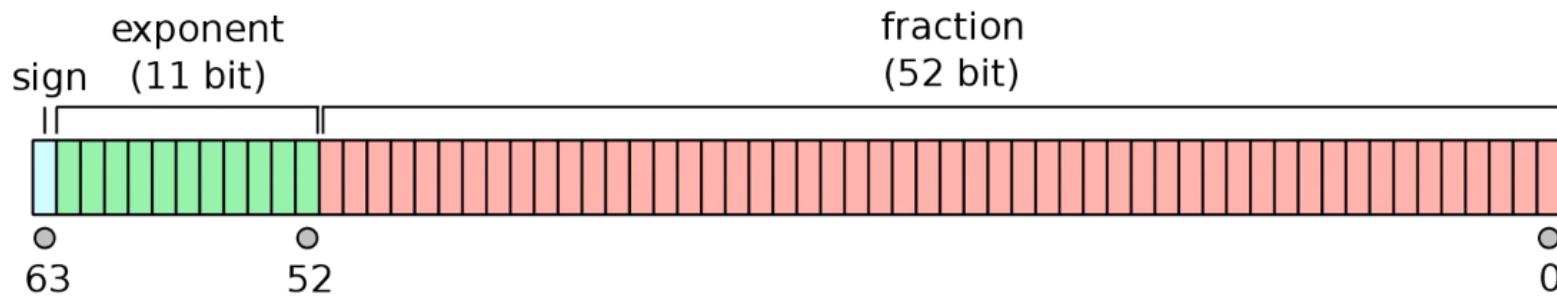


# Unicode

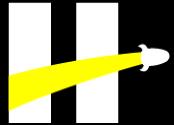
Number of bytes	Bits for code point	First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
1	7	U+0000	U+007F	0xxxxxxxx			
2	11	U+0080	U+07FF	110xxxxx	10xxxxxxxx		
3	16	U+0800	U+FFFF	1110xxxx	10xxxxxxxx	10xxxxxxxx	
4	21	U+10000	U+10FFFF	11110xxx	10xxxxxxxx	10xxxxxxxx	10xxxxxxxx



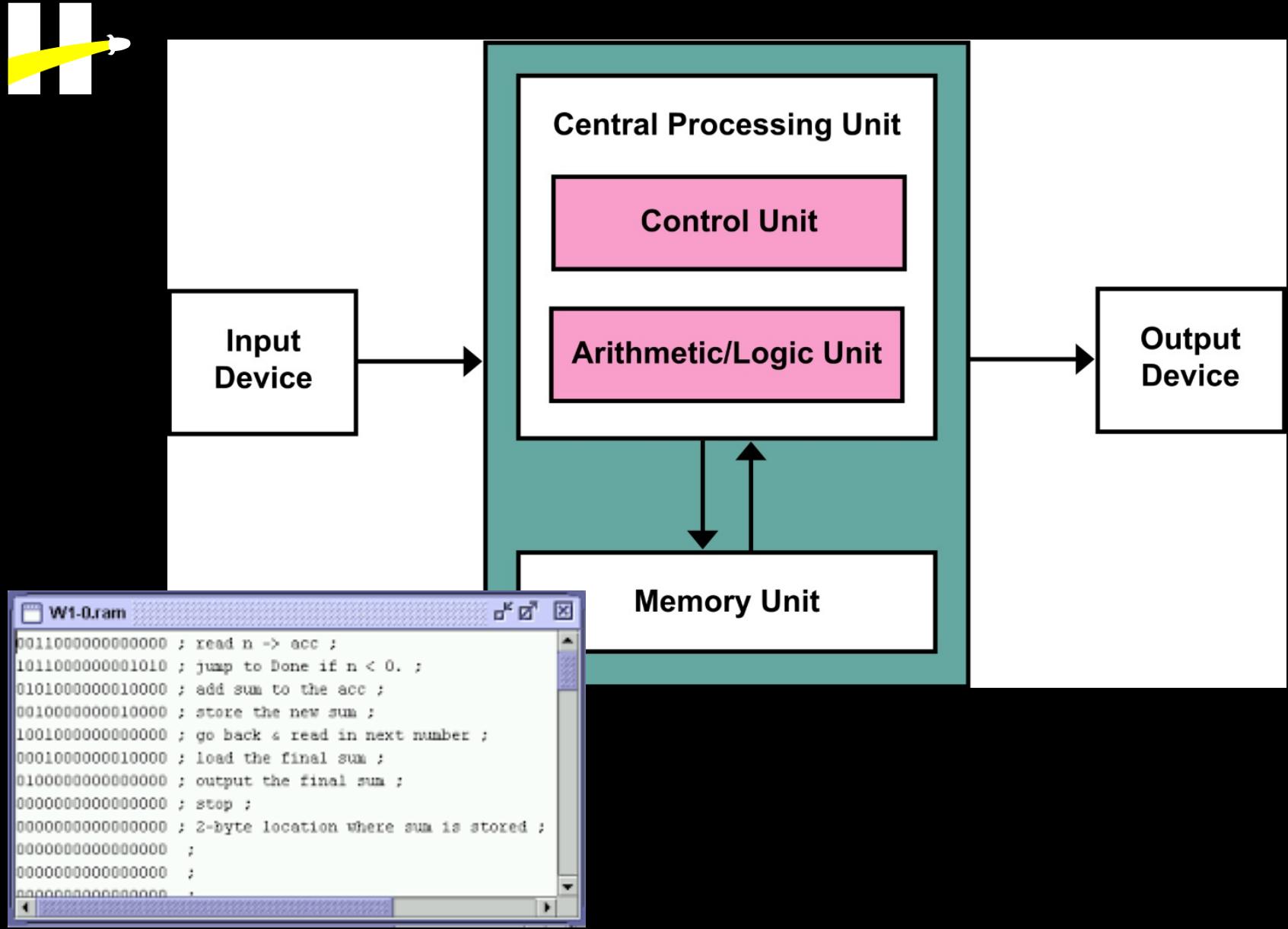
## IEEE 754: Floating-Point Signed Double

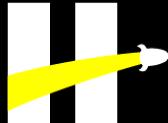


[https://www.h-  
schmidt.net/FloatConverter/IEEE754.html](https://www.h-schmidt.net/FloatConverter/IEEE754.html)

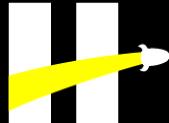


# Lenguaje de Máquina

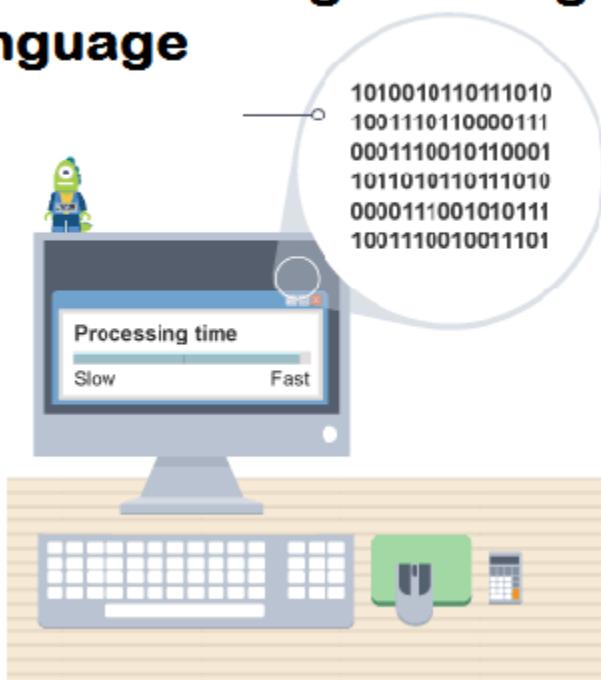




	op. code	source register	target reg.	destination reg.	shift amount	function type
instruction	000000	00001	00010	00110	00000	100000
hex	0x0	0x1	0x2	0x6	0x0	0x20
decimal	0	1	2	6	0	32
meaning	arithmetic	register 1	register 2	register 6	no offset	addition
English	"take the value in register 1, add the value in register 2, place the result in register 6"					
assembly	add \$t6, \$t1, \$t2					



## Low Level Programming Language



## High Level Programming Language

