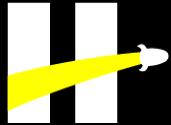
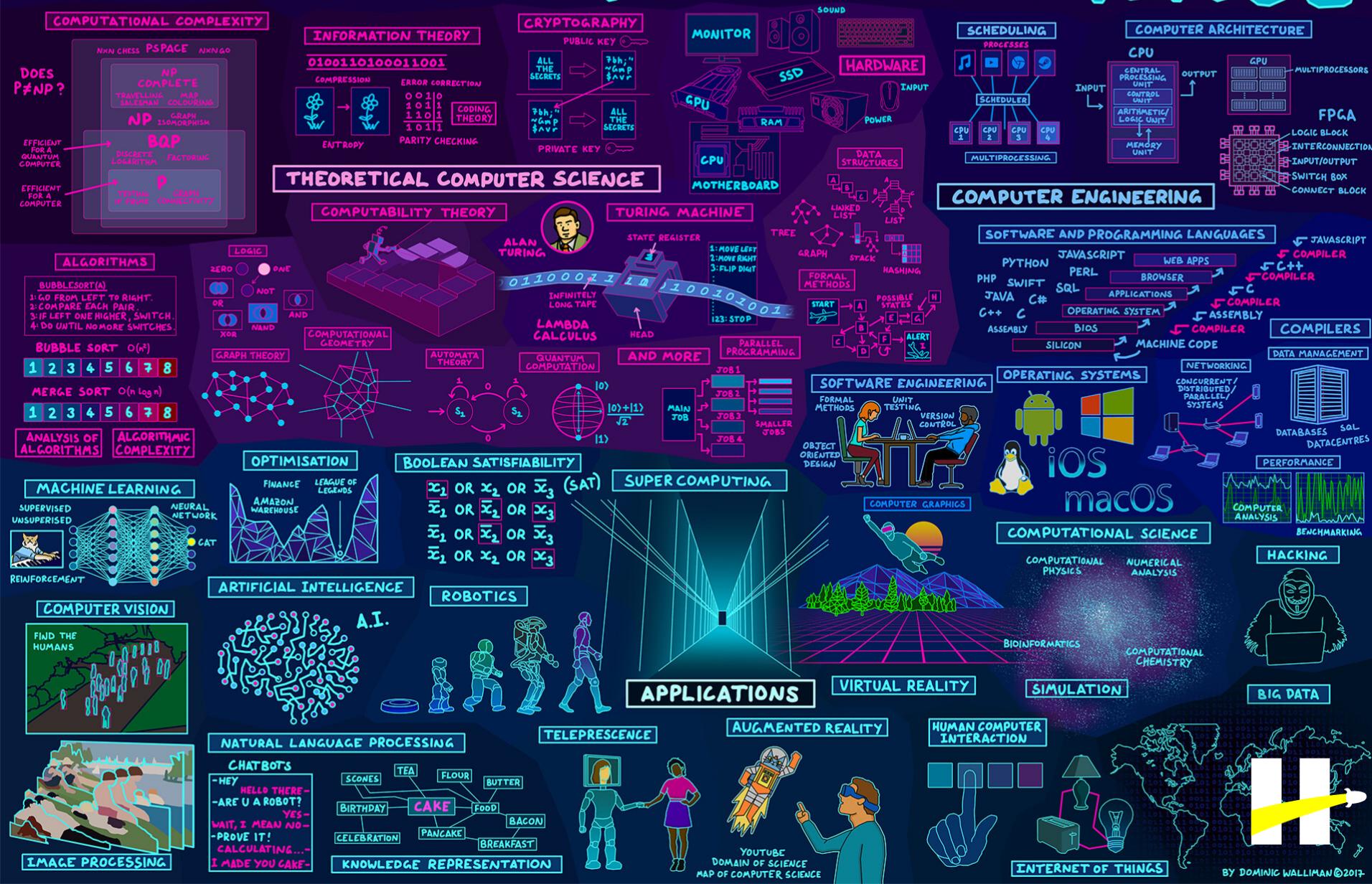


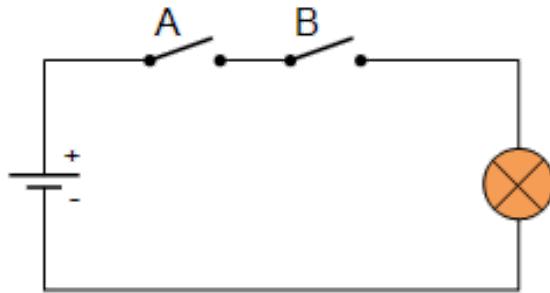
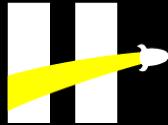
HENRY



# Intro to CS

# MAP OF COMPUTER SCIENCE

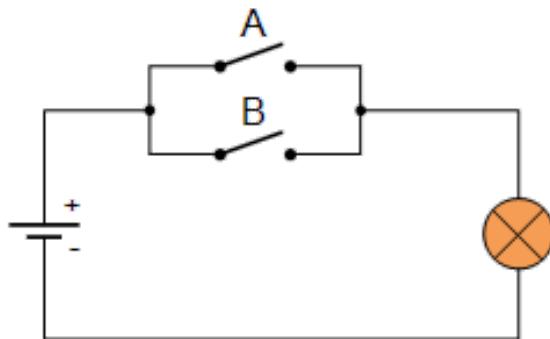




Lamp - ON = "1"  
Lamp - OFF = "0"

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

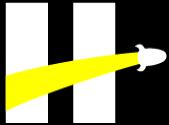
Switch B - Open = "0", Closed = "1"



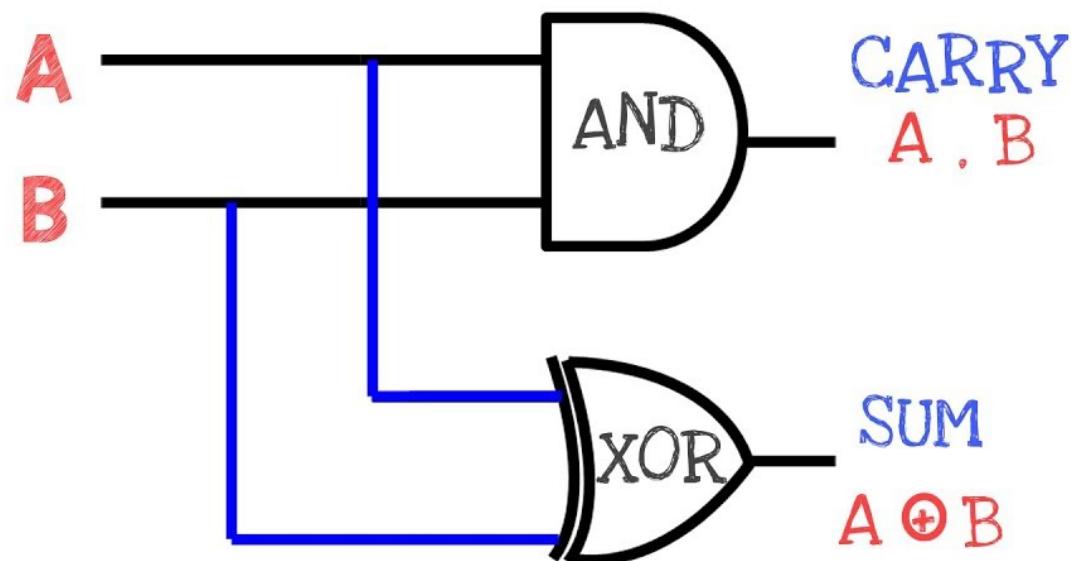
Lamp - ON = "1"  
Lamp - OFF = "0"

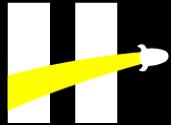
Switch A - Open = "0", Closed = "1"

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

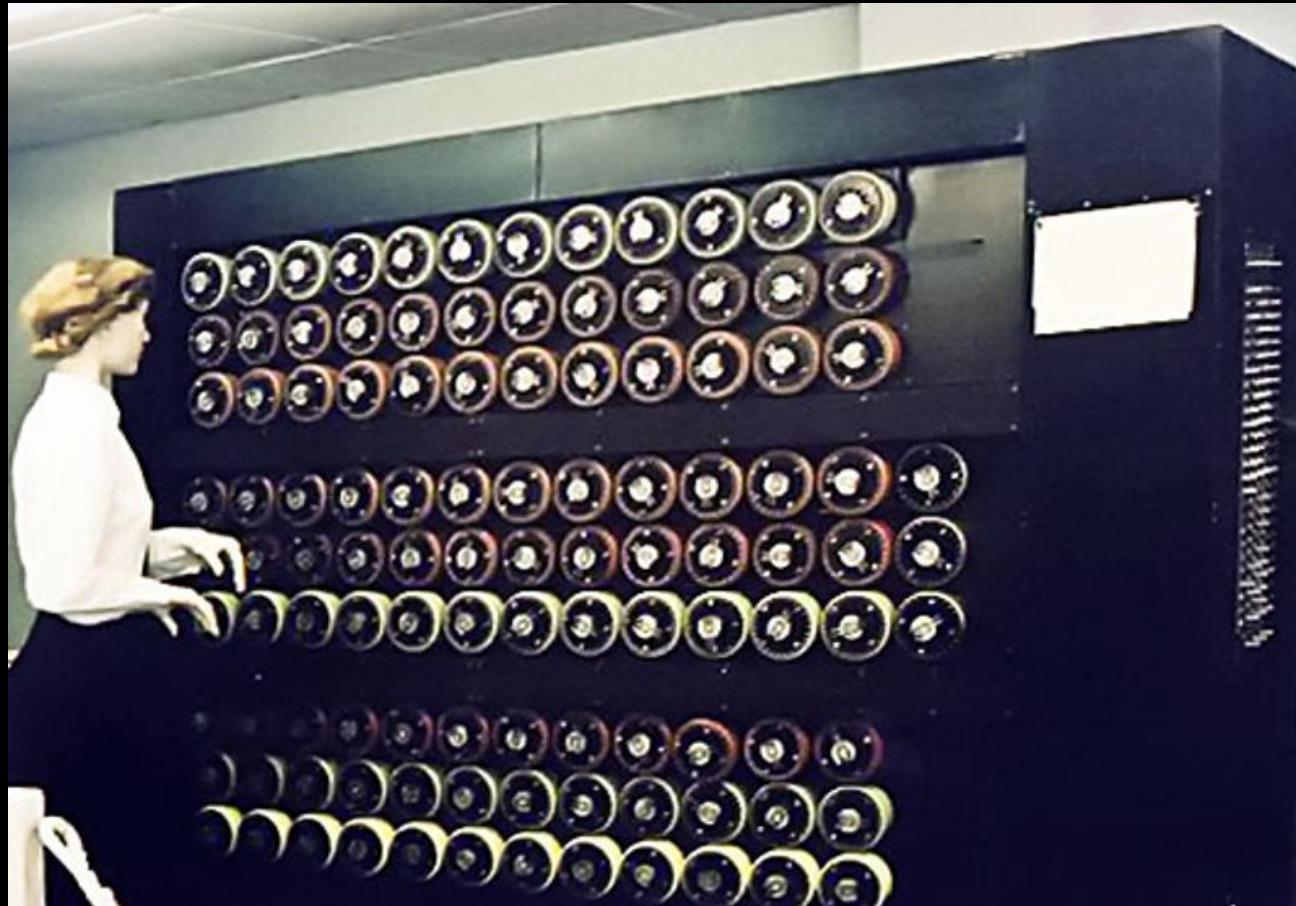
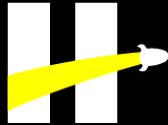


# Half Adder

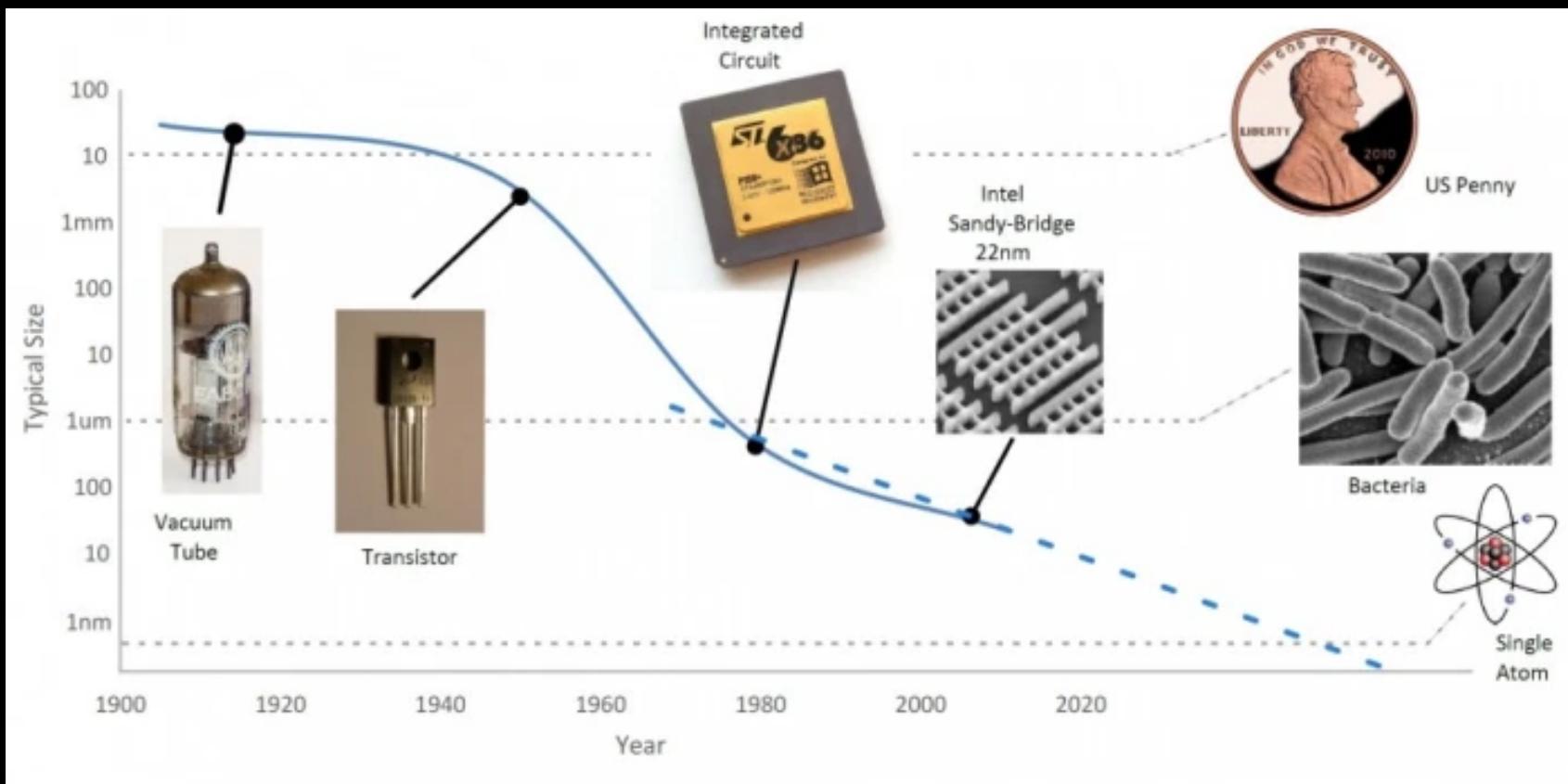
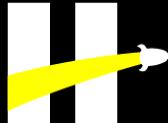




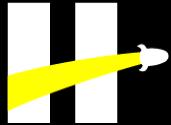
1937 - Adder



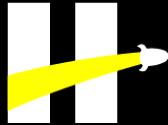
1937 - British Bombe  
Alan Turing



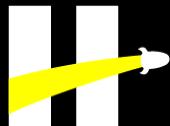
Ley de Moore.



# 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

70

6

5

4

6

4

1

Unidad

Decena

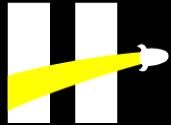
Centena

Unidad de Millar

Decena de Millar

Centena de Millar

## Unidad de Millón



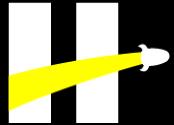
# Símbolos del sistema

Cada sistema utiliza sus propios símbolos:

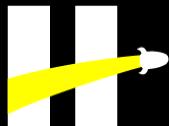
- Decimal: 0, 1, 2, 3, 4, 5, 6, 7, 8 y 9.
- Binario: 0 y 1.
- Quinario: 0, 1, 2, 3 y 4.
- Octal: 0, 1, 2, 3, 4, 5, 6 y 7.
- Hexadecimal: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E y F.

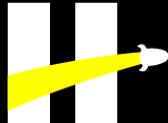
$$N = \sum_{i=-k}^{n-1} d_i \cdot 10^i$$

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



# Representaciones

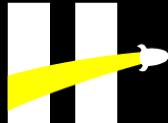




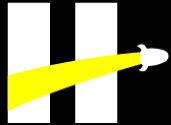
# ASCII

↑

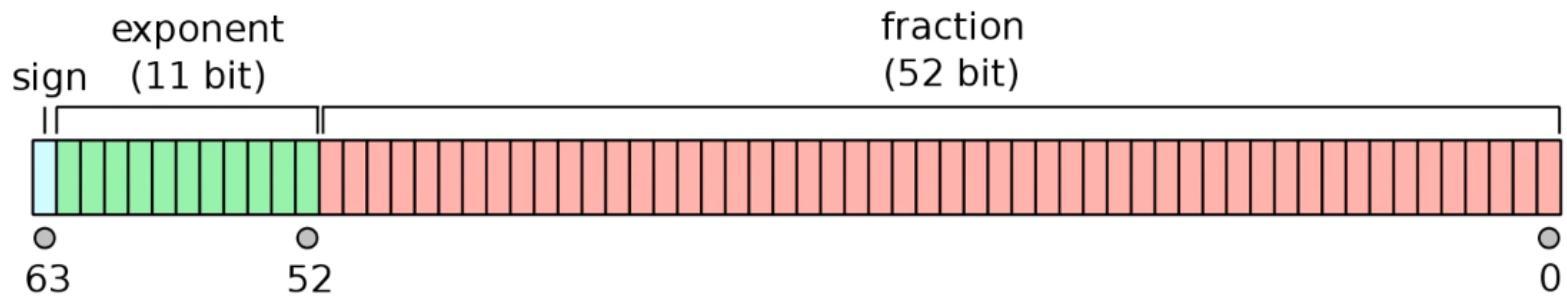
Binary	Decimal	Glyph	Binary	Decimal	Glyph
0010 1110	46	.	0011 1010	58	:
0010 1111	47	/	0011 1011	59	;
0011 0000	48	o	0011 1100	60	<
0011 0001	49	I	0011 1101	61	=
0011 0010	50	2	0011 1110	62	>
0011 0011	51	3	0011 1111	63	?
0011 0100	52	4	0100 0000	64	@
0011 0101	53	5	0100 0001	65	A
0011 0110	54	6	0100 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



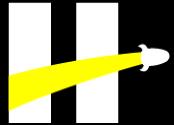
Glyph	Unicode Code Point	Code Unit 1: <i>singleton or lead</i>	Code Unit 2: <i>trail 1</i>	Code Unit 3: <i>trail 2</i>	Code Unit 4: <i>trail 3</i>
I	U+0049	0100 1001			
♥	U+2764	1110 0010	1001 1101	1010 0100	
J	U+004A	0100 1010			
S	U+0053	0101 0011			



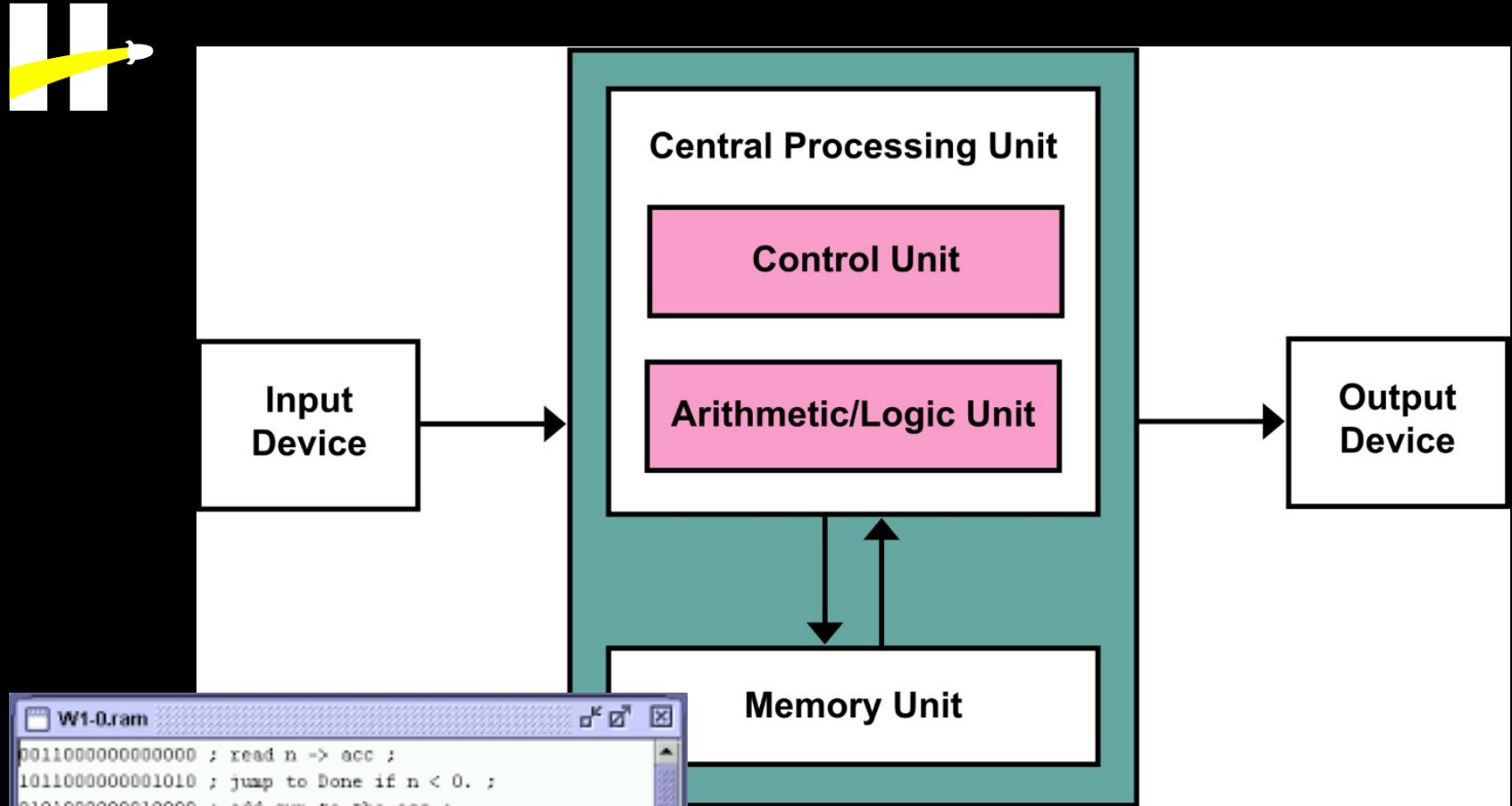
## IEEE 754: Floating-Point Signed Double



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

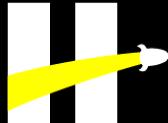


# Lenguaje de Máquina

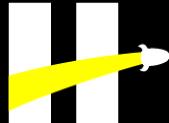


The screenshot shows a software window titled "W1-0.ram". The window contains assembly language code with comments explaining the steps of a program. The code is as follows:

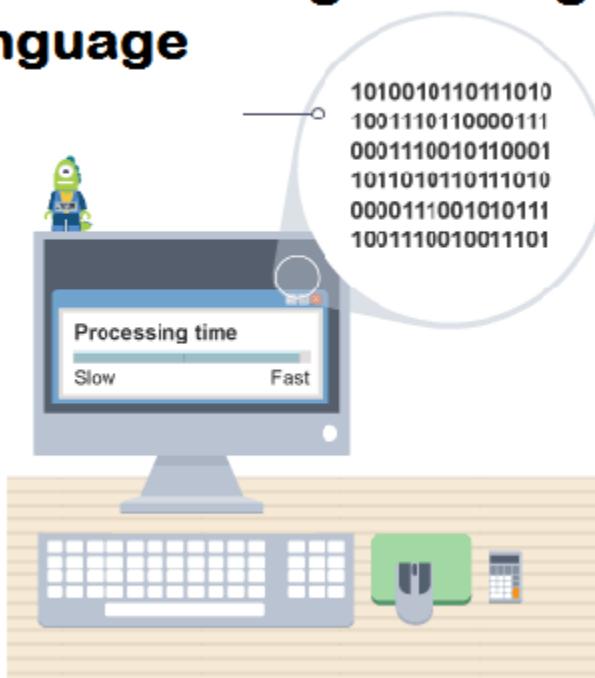
```
W1-0.ram
0011000000000000 ; read n -> acc ;
1011000000001010 ; jump to Done if n < 0. ;
0101000000010000 ; add sum to the acc ;
0010000000010000 ; store the new sum ;
1001000000000000 ; go back & read in next number ;
0001000000010000 ; load the final sum ;
0100000000000000 ; output the final sum ;
0000000000000000 ; stop ;
0000000000000000 ; 2-Byte location where sum is stored ;
0000000000000000 ;
0000000000000000 ;
0000000000000000 ;
```



	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

