

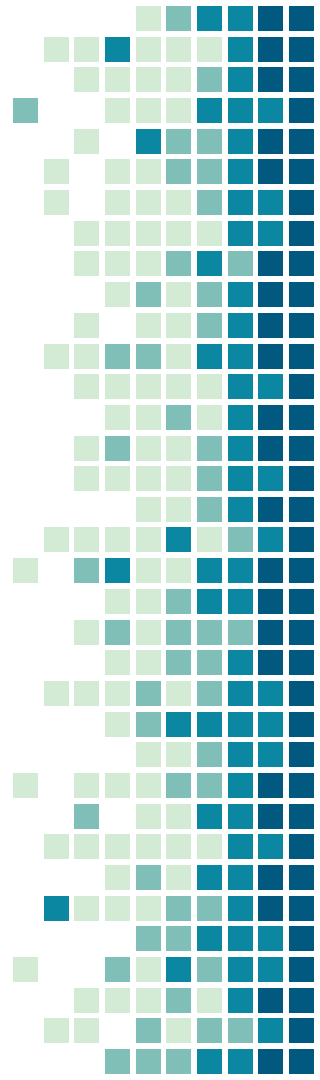
Arquitectura del Computador II

Repaso.



Lógica Booleana

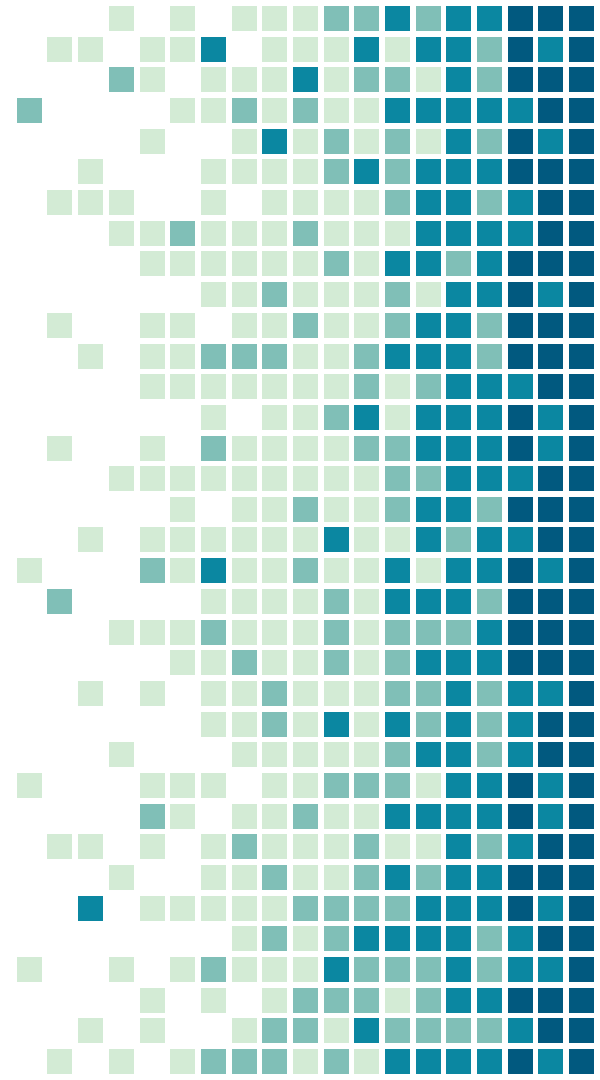
- Compuertas lógicas
- Mapas de Karnaugh
- Simplificación de funciones.
- Representación booleana y en forma de circuitos.
- Construcción de compuertas en base a las compuertas básicas.
- Simbología
- Qué es un HDL?



x	Not (x)
0	1
1	0

x	y	And (x, y)
0	0	0
0	1	0
1	0	0
1	1	1

x	y	Or (x, y)
0	0	0
0	1	1
1	0	1
1	1	1



Álgebra de Boole

1. $A + 0 = A$

2. $A \cdot 1 = A$

3. $A + 1 = 1$

4. $A \cdot 0 = 0$

5. $A + A = A$

6. $A \cdot A = A$

7. $A + \bar{A} = 1$

8. $A \cdot \bar{A} = 0$

9. $\bar{\bar{A}} = A$

10. $A + B = B + A$

11. $A \cdot B = B \cdot A$

12. $A + (B + C) = (A + B) + C$

13. $X(YZ) = (X \cdot Y)Z$

14. $A \cdot (B + C) = AB + AC$

15. $A + BC = (A + B) \cdot (A + C)$

16. $\overline{A+B} = \bar{A} \cdot \bar{B}$

17. $\overline{A \cdot B} = \bar{A} + \bar{B}$

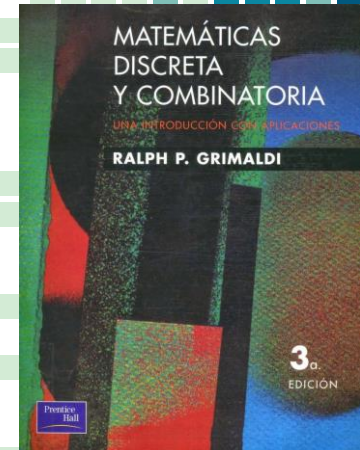
Lei comutativa

Lei associativa

Lei distributiva

DeMorgan

<https://slideplayer.com.br/slide/338032/1/images/7/Axiomas+e+Teoremas+da+%C3%81lgebra+Booleana+de+Chaveamento.jpg>



Mapas de Karnaugh

la variable que cambia es la que se desecha

C \ AB	0	1
	00	01
0	0	1
1	0	1
2	0	1
3	0	1

(a) $X = C$

CD \ AB	00	01	11	10
	00	01	11	10
0	0	0	0	0
1	0	0	0	0
2	1	1	1	1
3	0	0	0	0

(b) $X = AB$

CD \ AB	00	01	11	10
	00	01	11	10
0	0	0	0	0
1	0	1	1	0
2	0	1	1	0
3	0	0	0	0

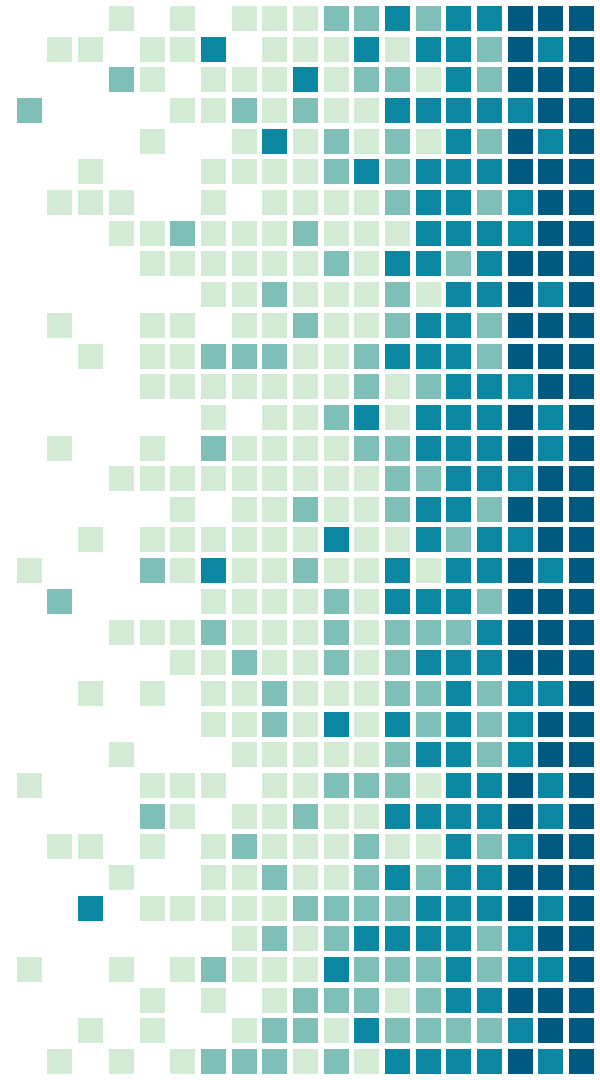
(c) $X = BD$

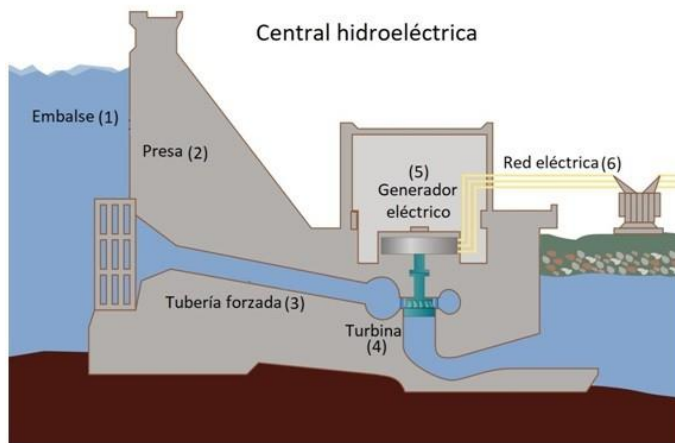
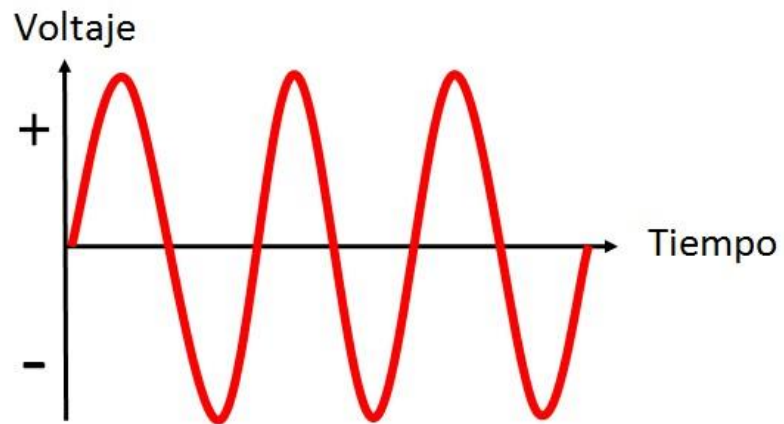
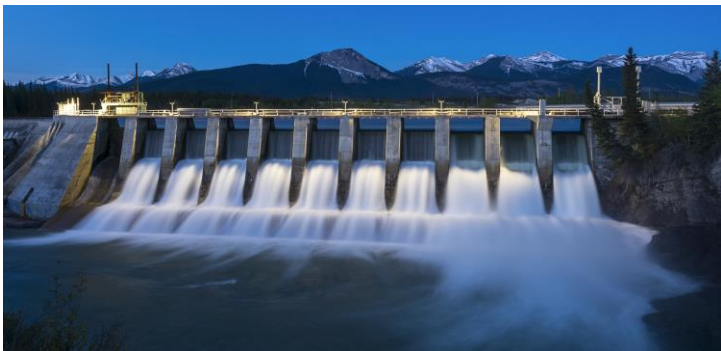
CD \ AB	00	01	11	10
	00	01	11	10
0	0	0	0	0
1	0	0	0	0
2	1	0	0	1
3	1	0	0	1

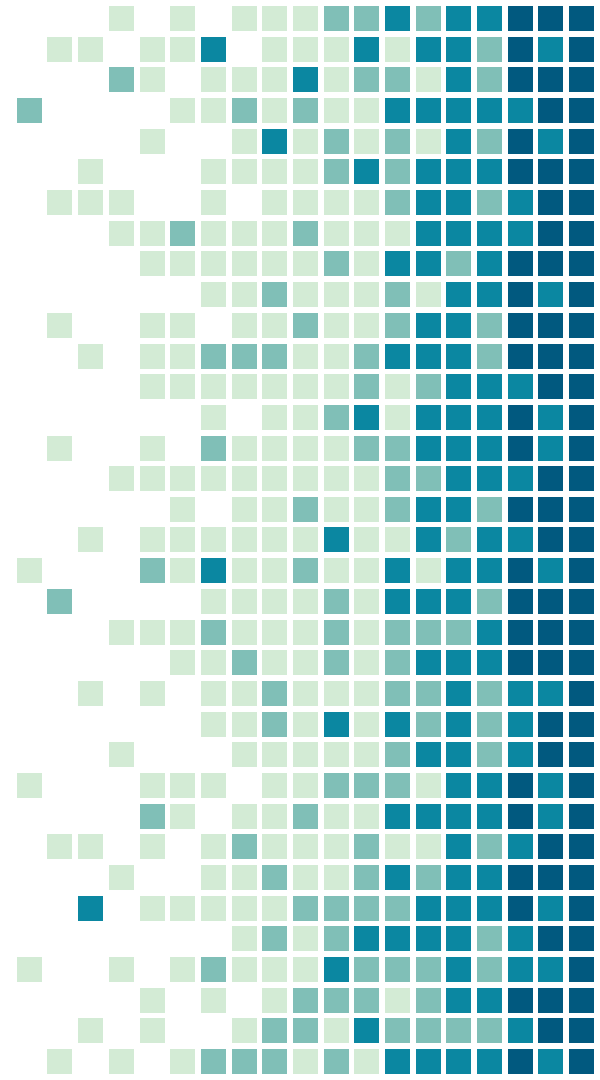
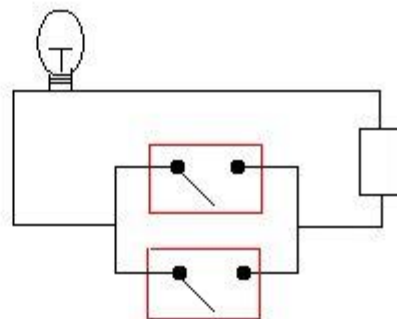
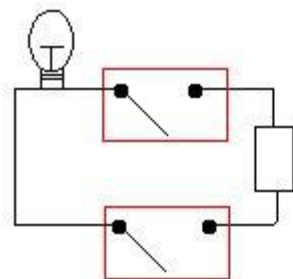
(d) $X = AD$

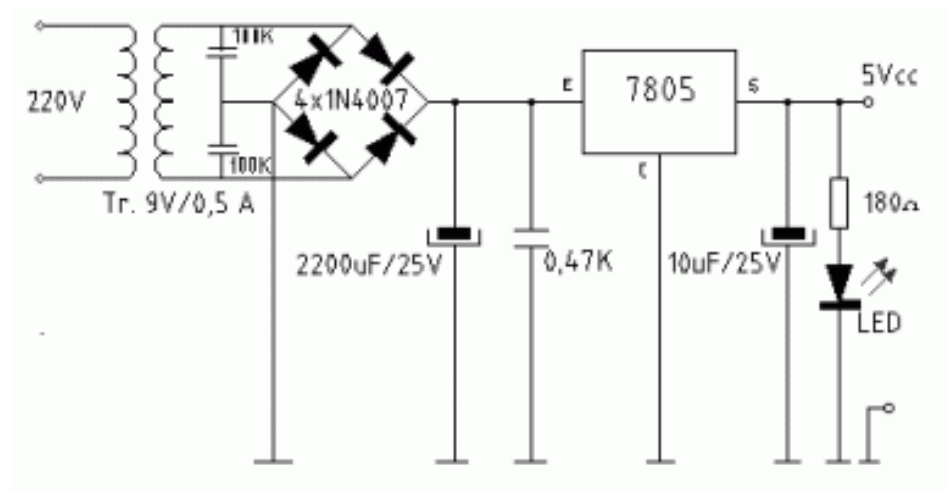
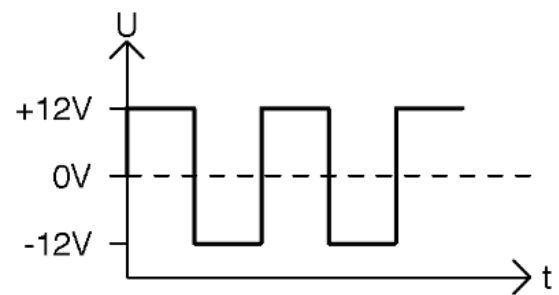
CD \ AB	00	01	11	10
	00	01	11	10
0	1	0	0	1
1	0	0	0	0
2	0	0	0	0
3	1	0	0	1

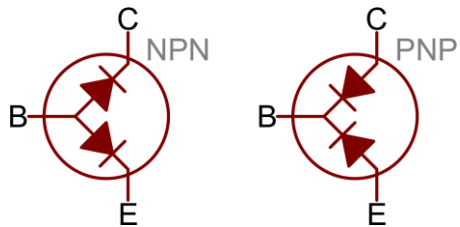
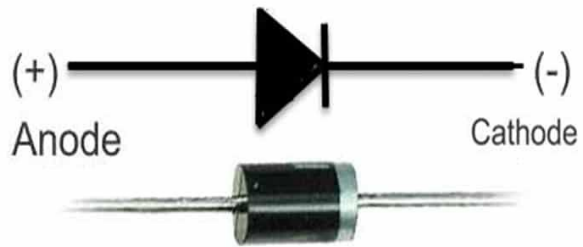
(e) $X = \overline{BD}$



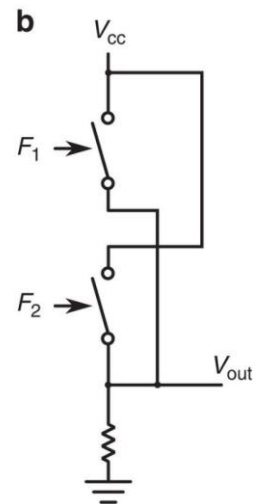
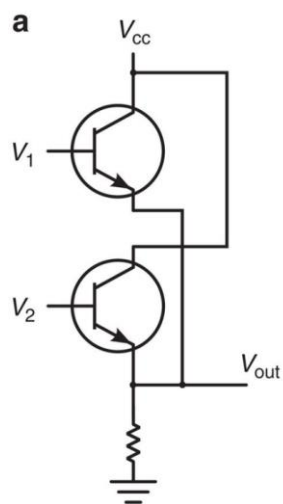
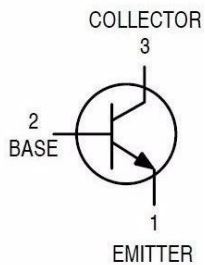
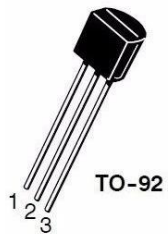




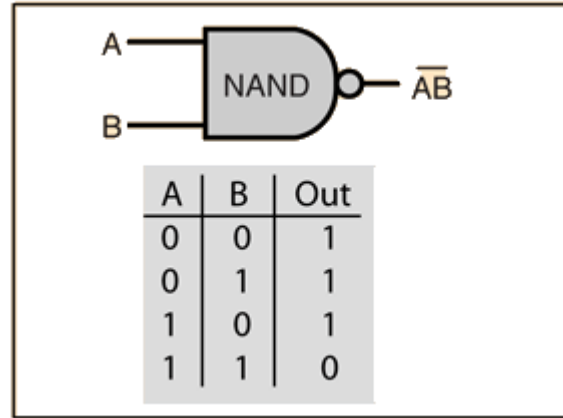




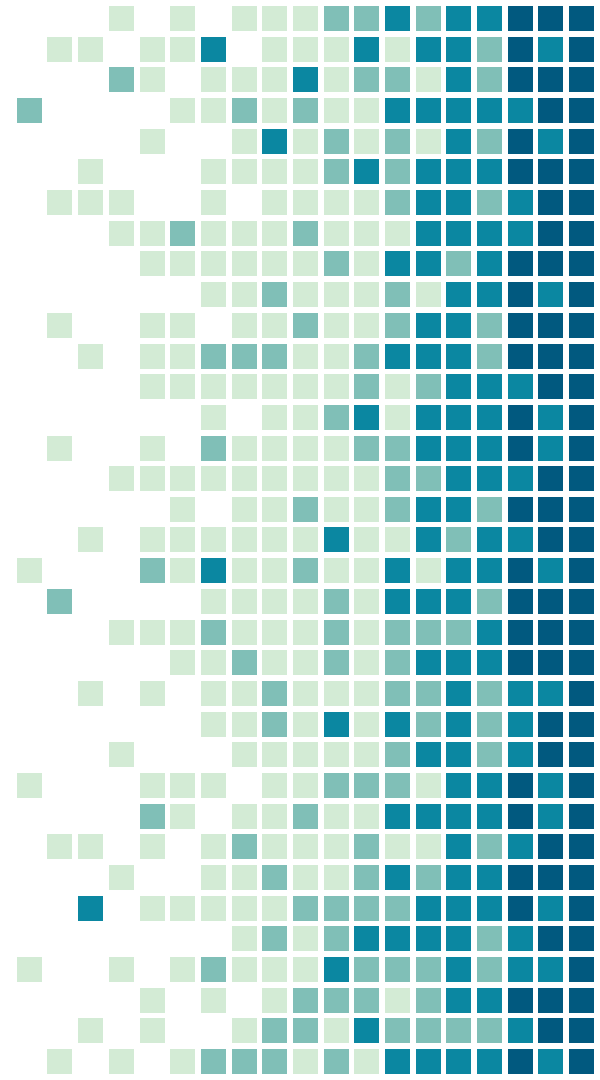
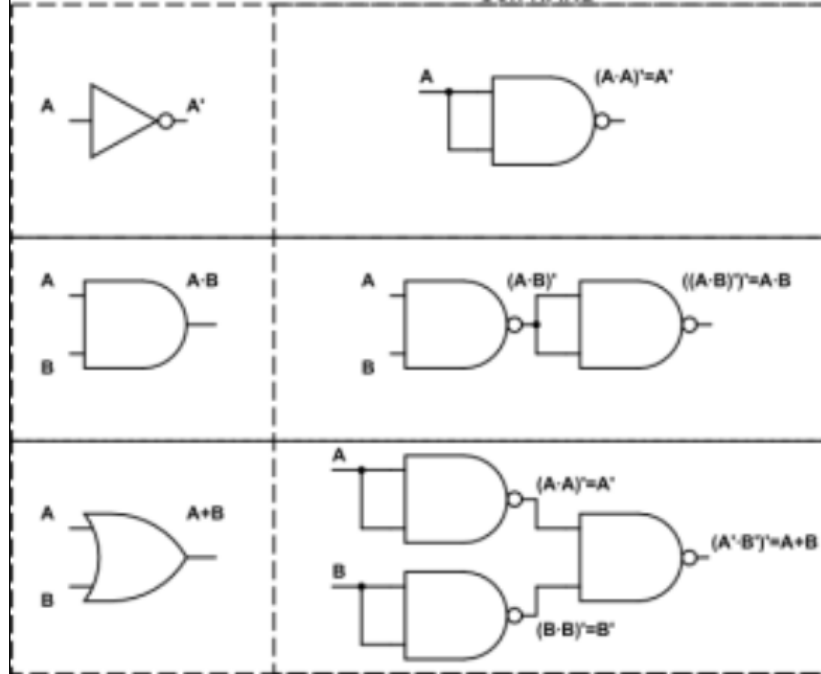
2N2222



NAND



Compuertas básicas con NAND



Theory

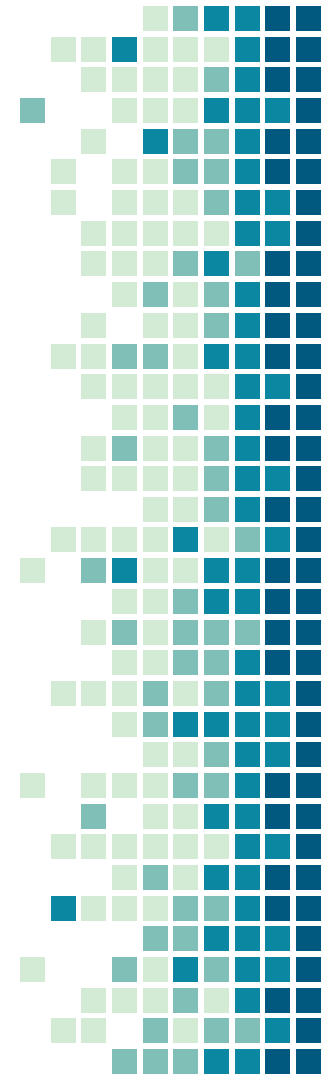


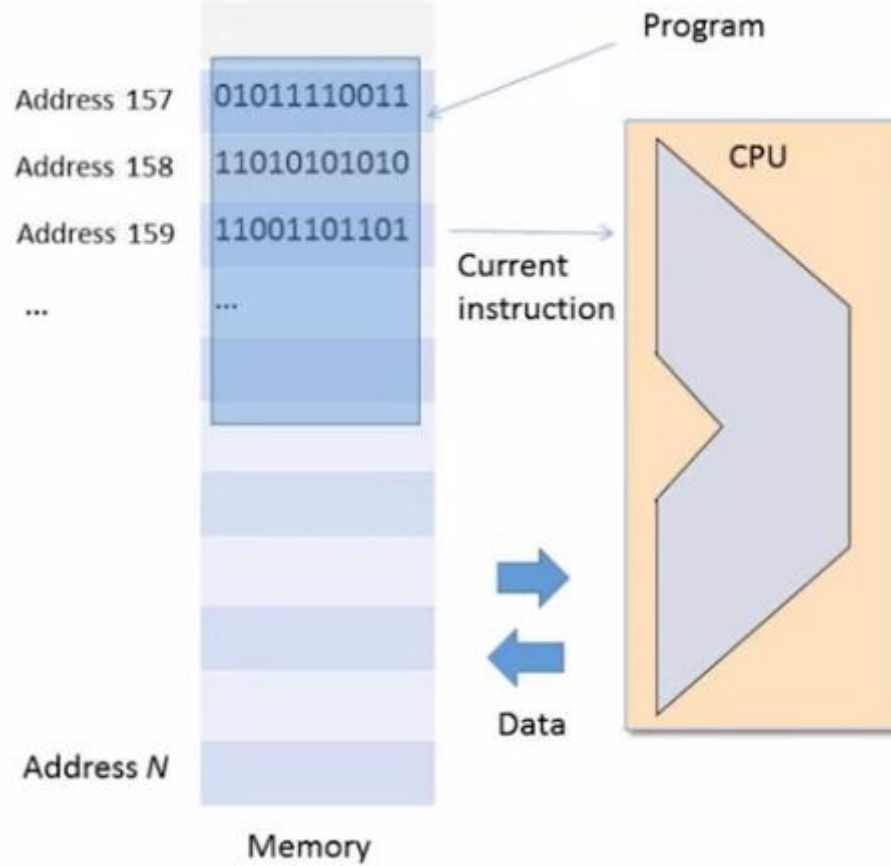
Universal Turing Machine

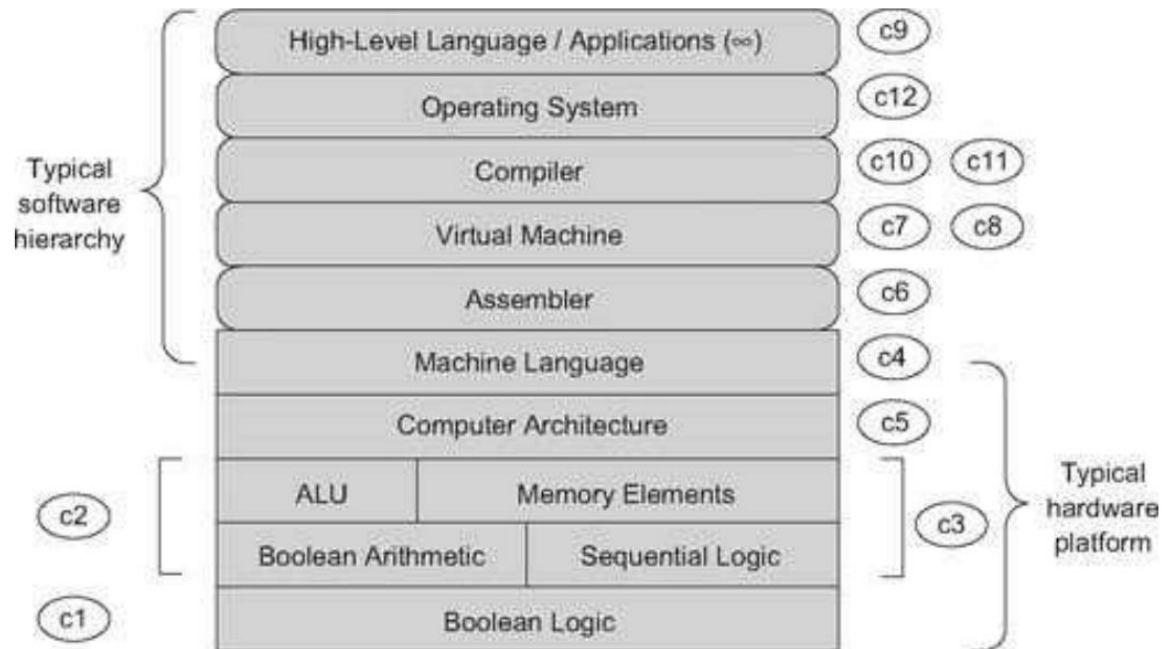
Practice



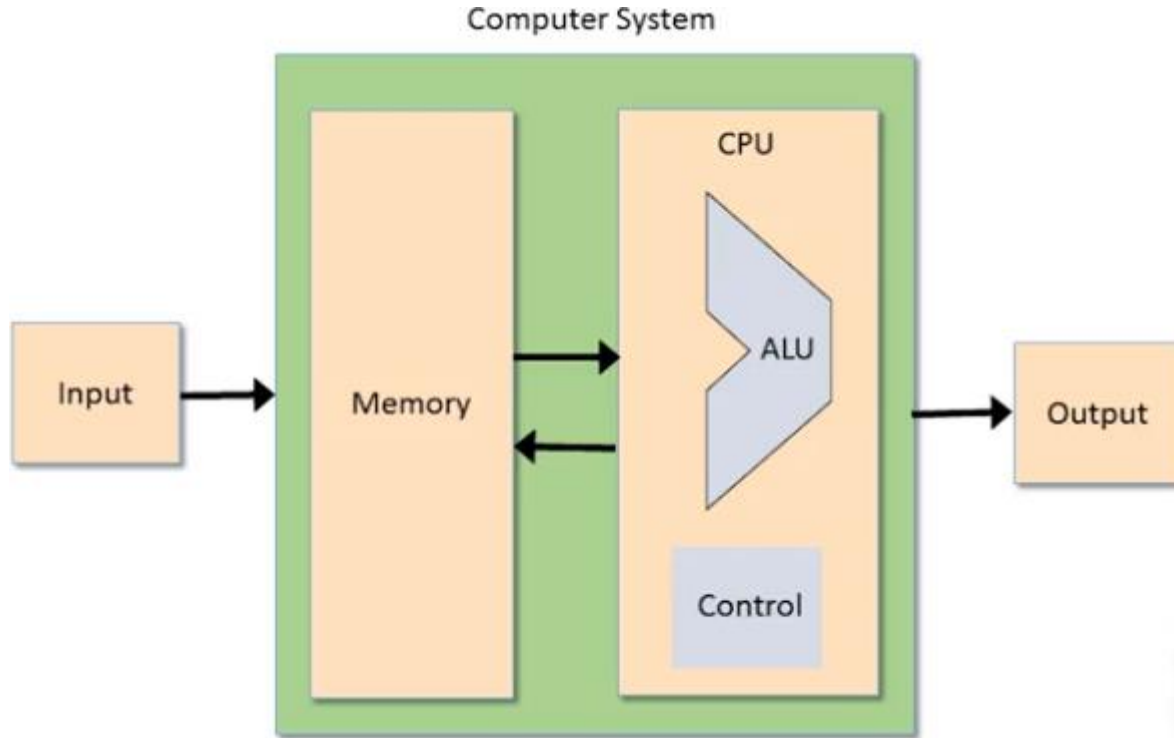
von Neumann Architecture







Von Neumann Architecture:



Ejercicio rápido:

Cuánto es:

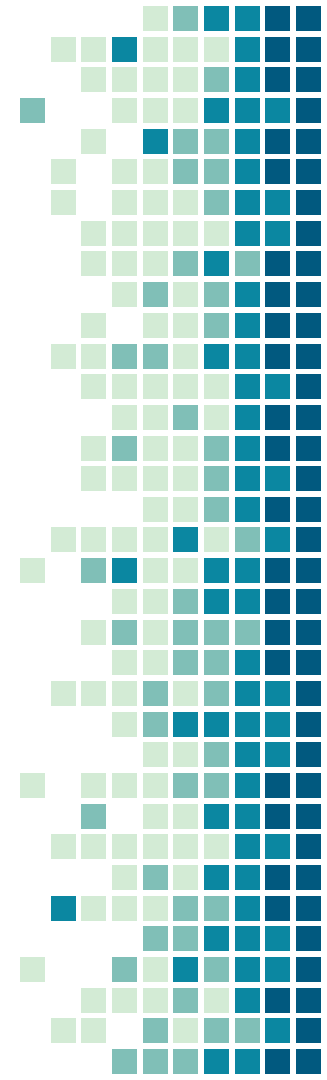
01111101 +

00110110

01111101 +

00110110

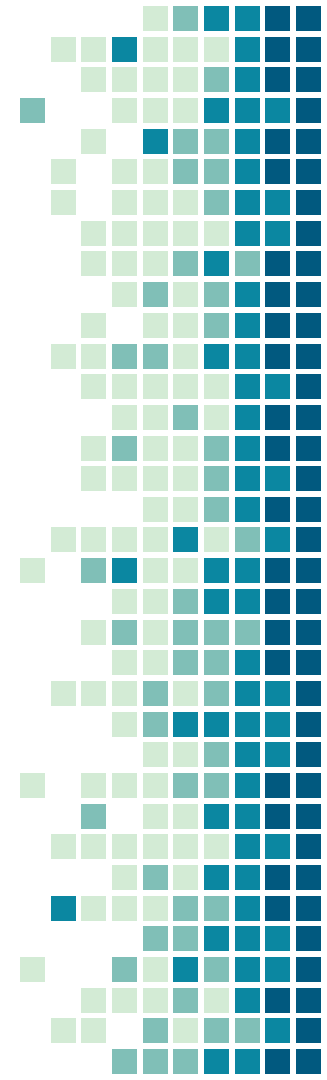
10110011



Overflow

$$\begin{array}{r} 1 \quad 1 \quad 1 \quad 1 \\ 10010101 \\ + \\ 11011100 \\ \hline 101110001 \end{array}$$

Exceder el tamaño de la palabra.

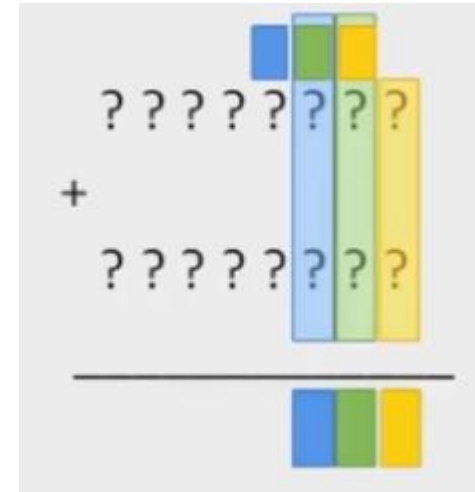
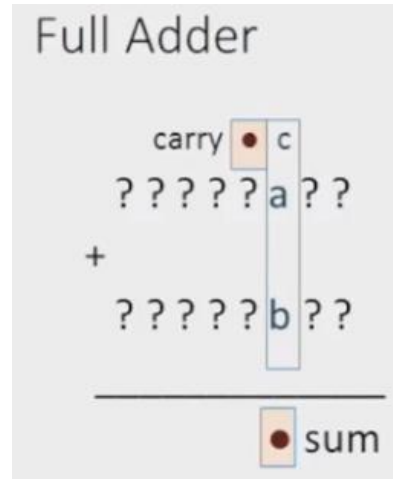
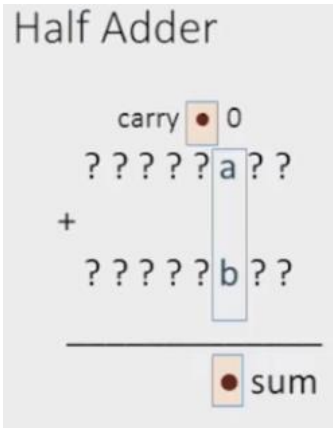


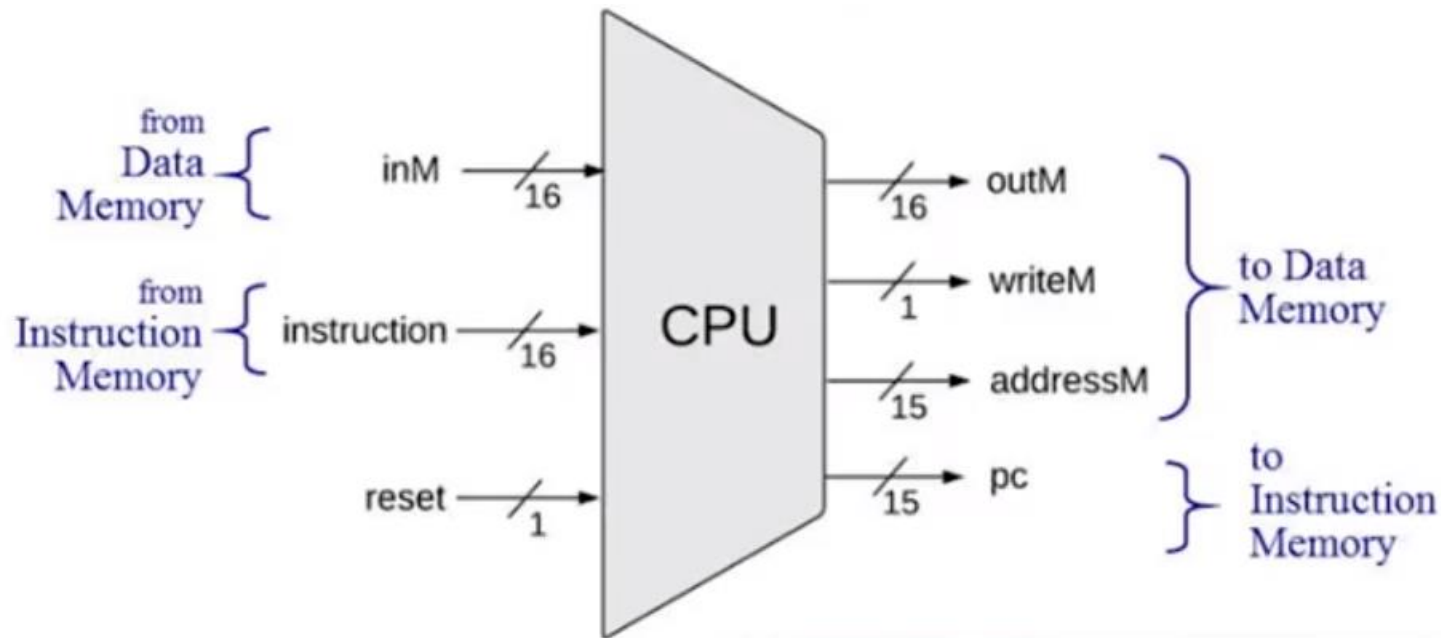
Sumadores:

Medio Sumador – suma de 2 bits.

Sumador completo – Suma de 3 bits.

Sumador – Suma 2 números.



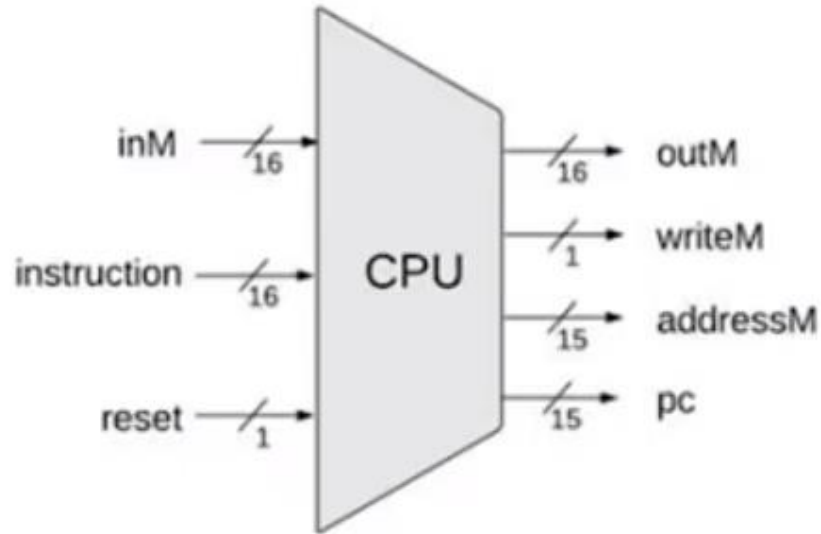


Sample Hack
instructions:

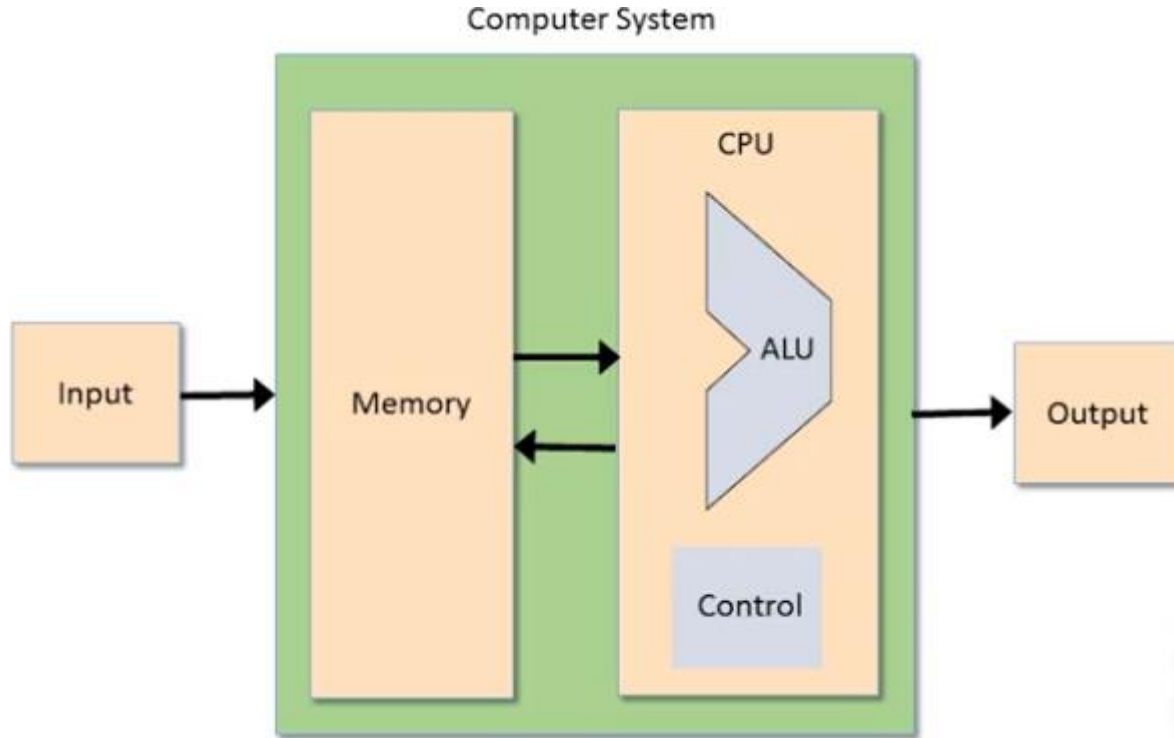
$D = D - A$

@17

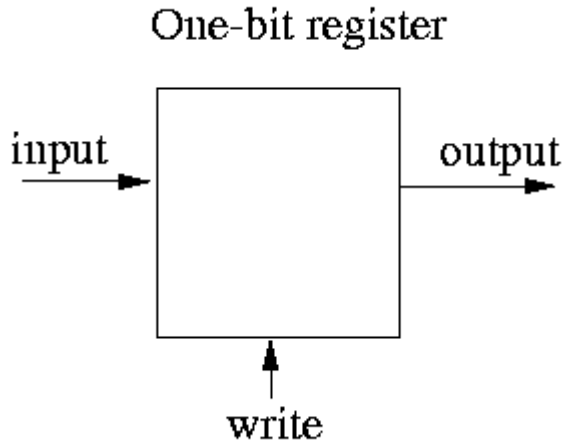
$M = M + 1$



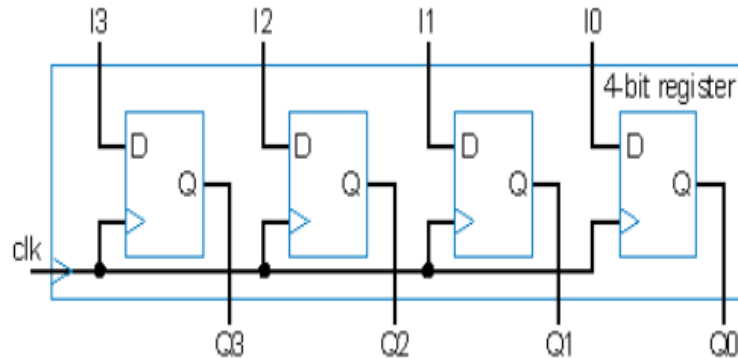
Von Neumann Architecture:

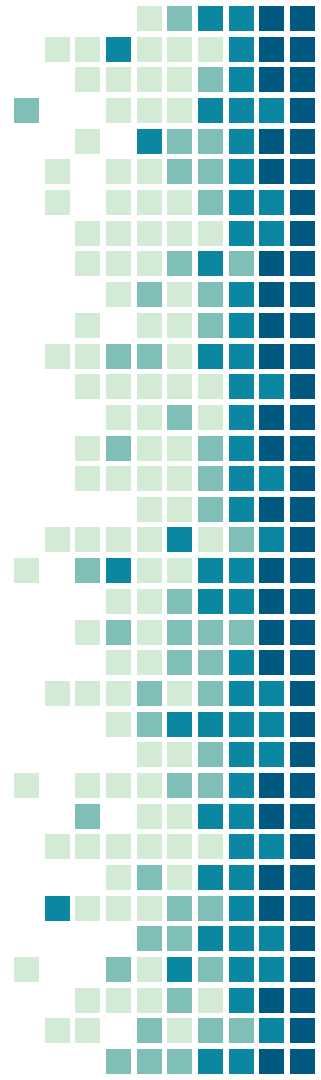
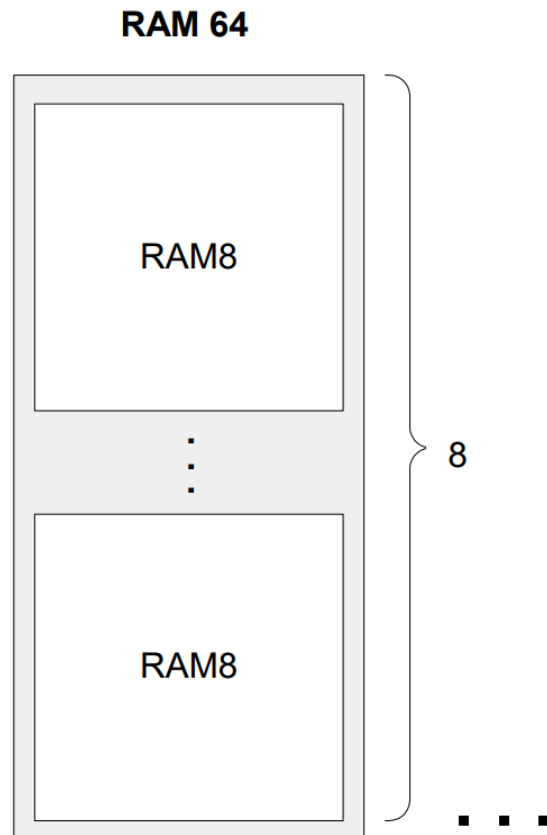
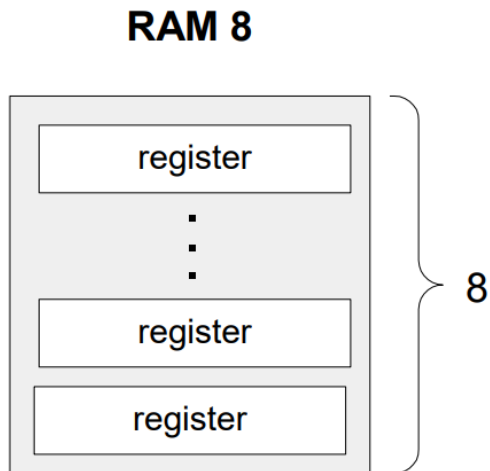
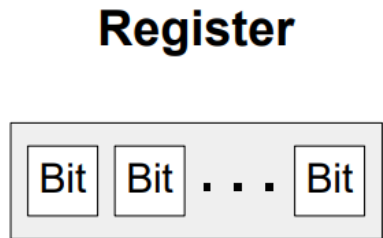


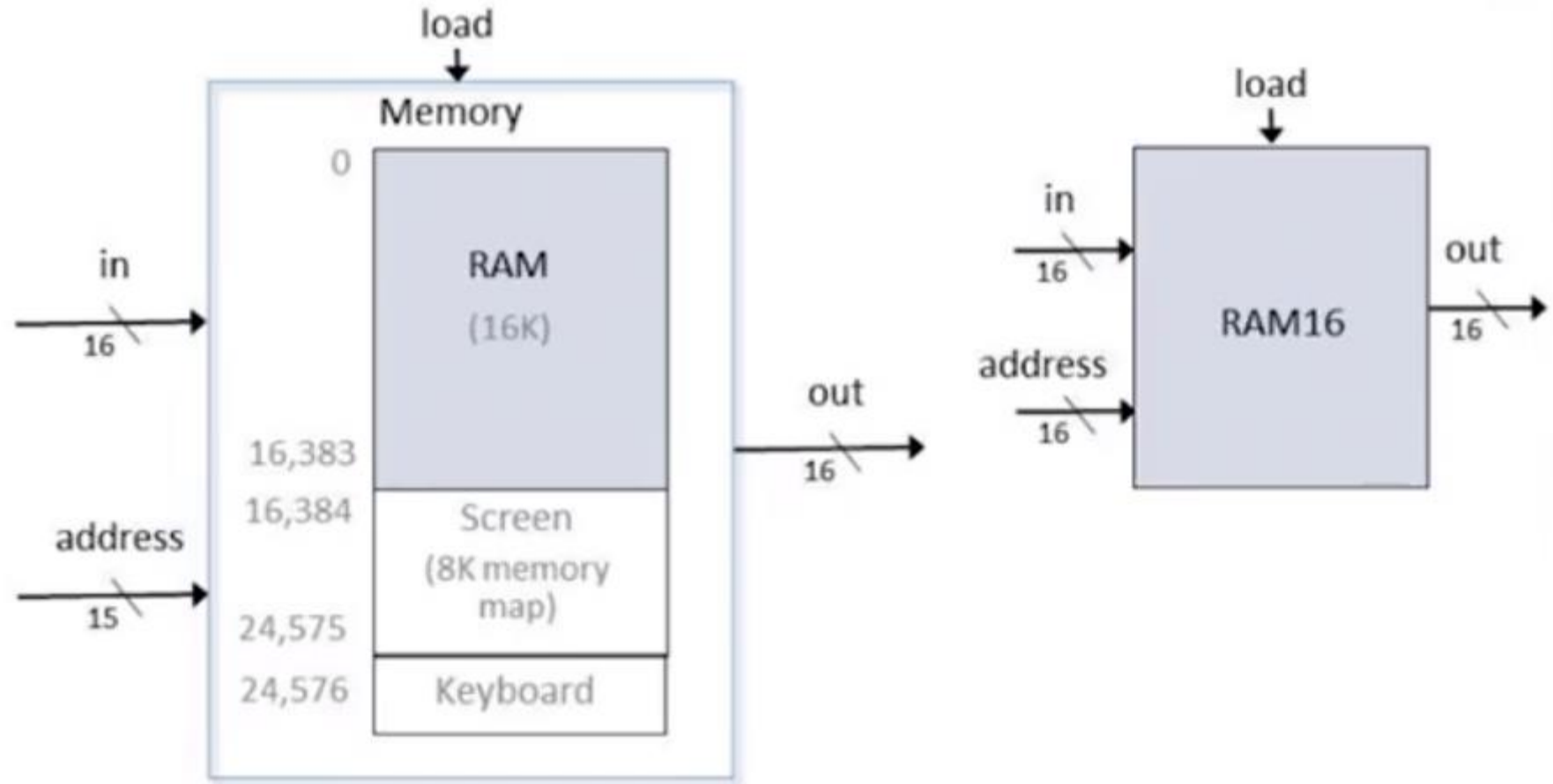
One Bit and Multi Bit Register



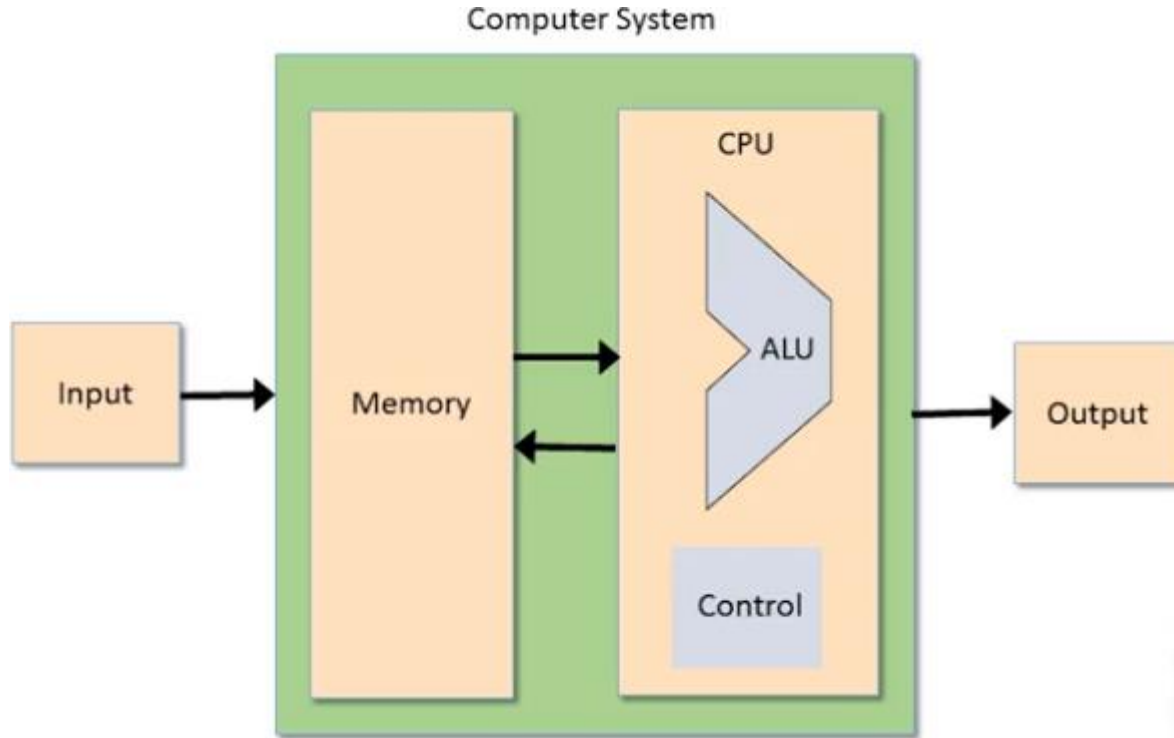
- Word (16bit, 32 ...)
- Register State (Qué tiene guardado el registro)

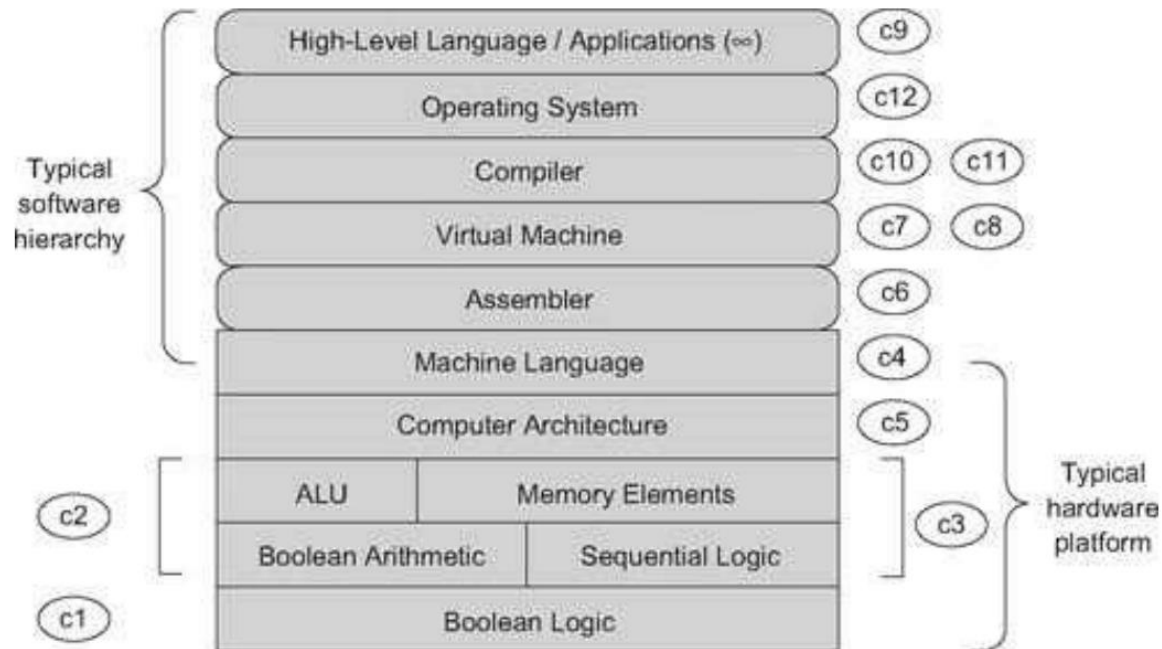






Von Neumann Architecture:





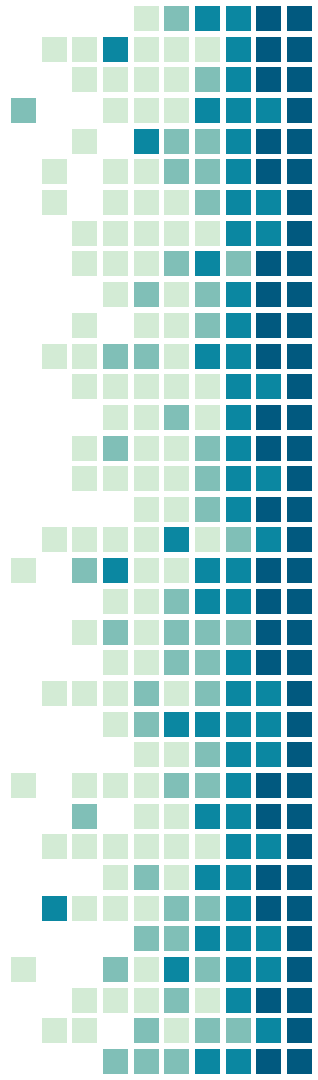
Símbolos

Machine Language

1010000110000001

Assembly Language

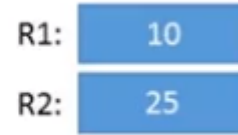
ADD 1, Mem[129]



Registros

- Data Registers

- Add R1, R2



- Address Registers

- Store R1, @A



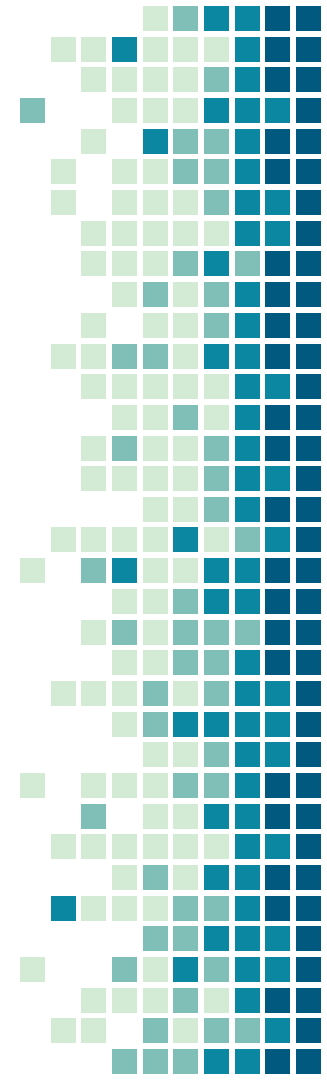
Representaciones

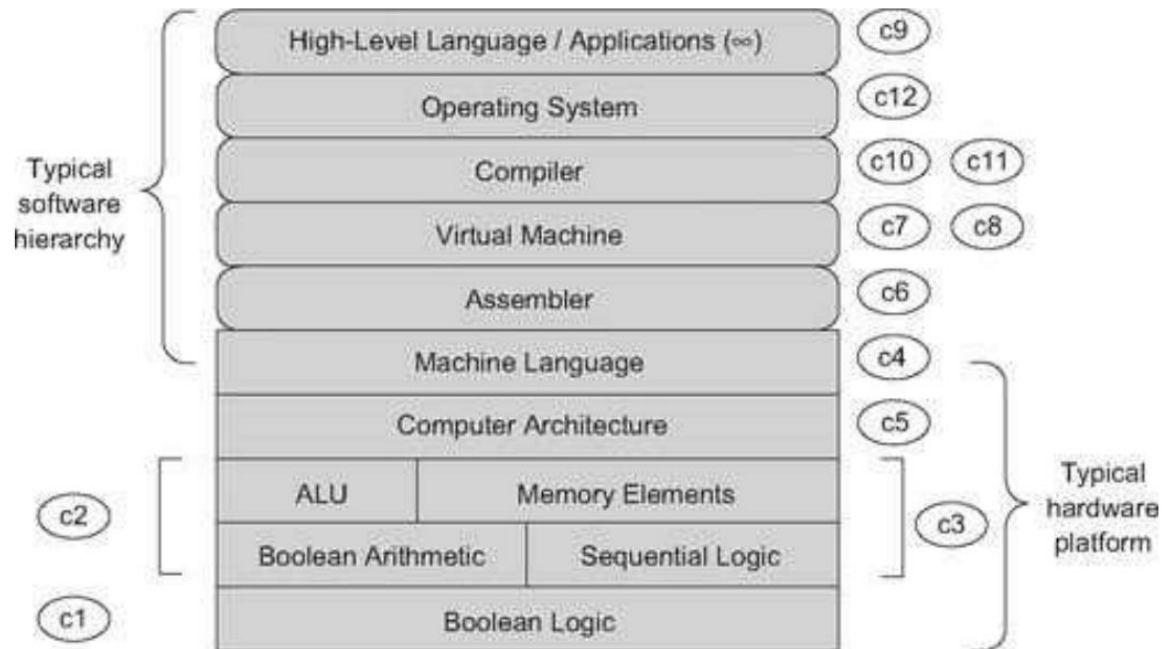
Symbolic:

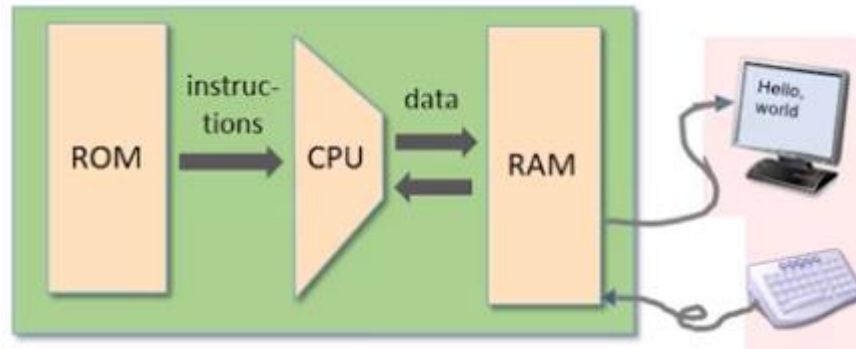
```
@17  
D+1;JLE
```

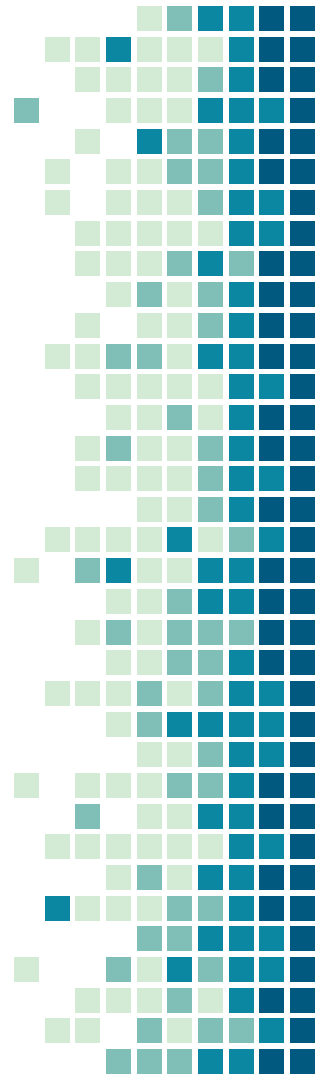
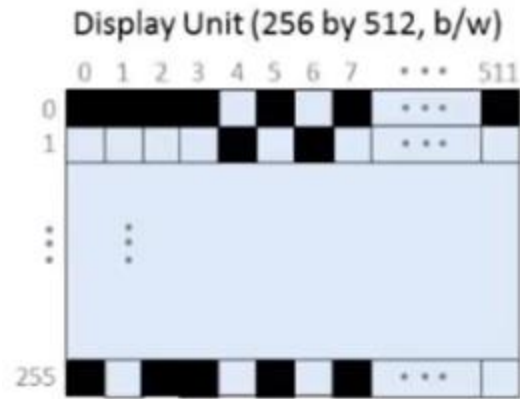
Binary:

```
000000000000010001  
1110011111000110
```

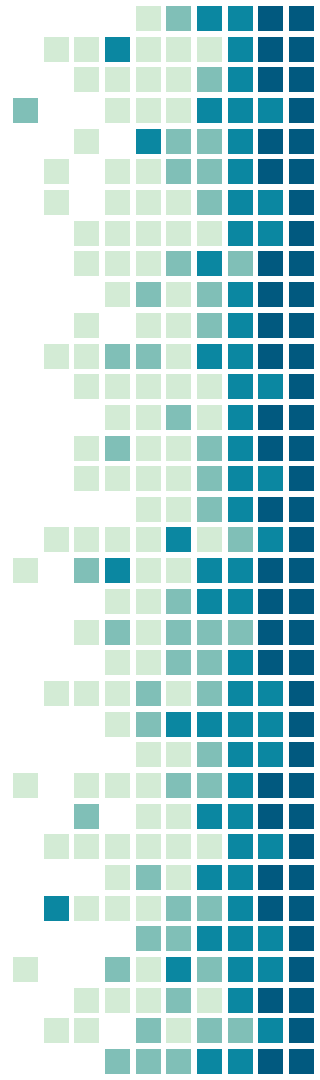


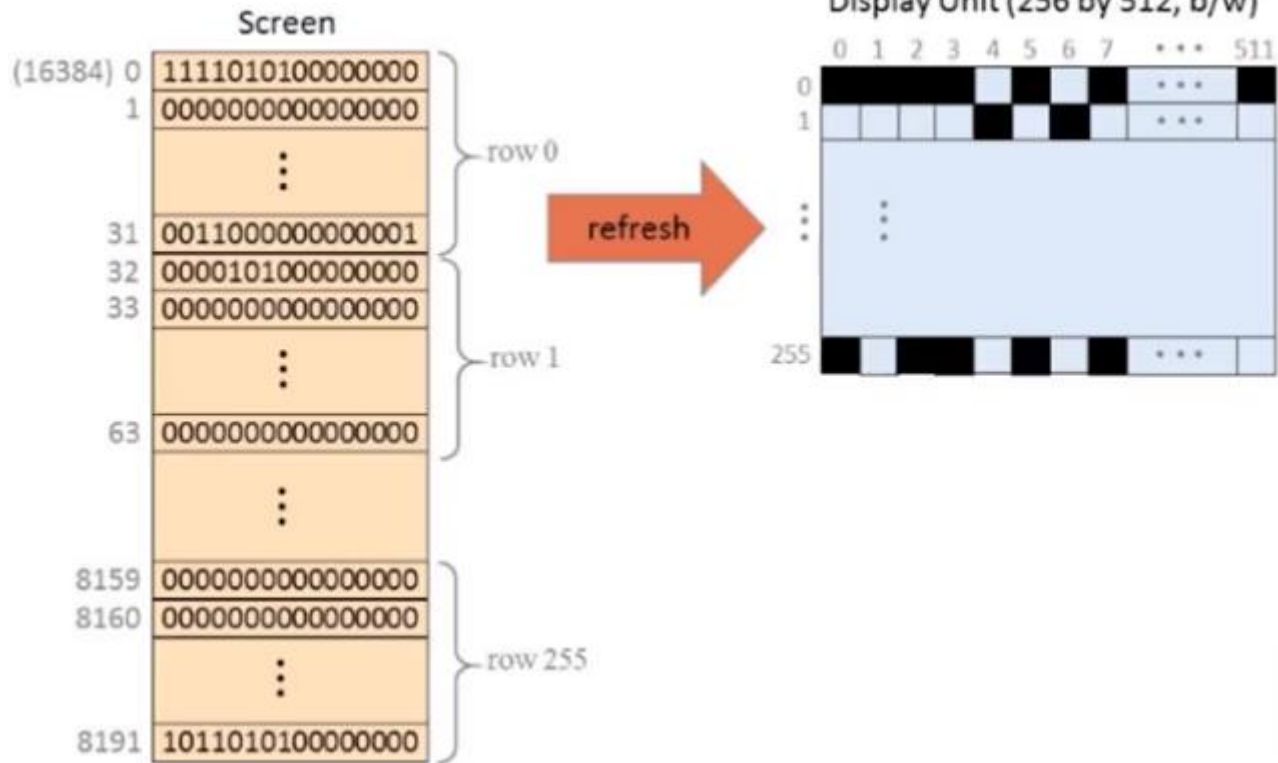


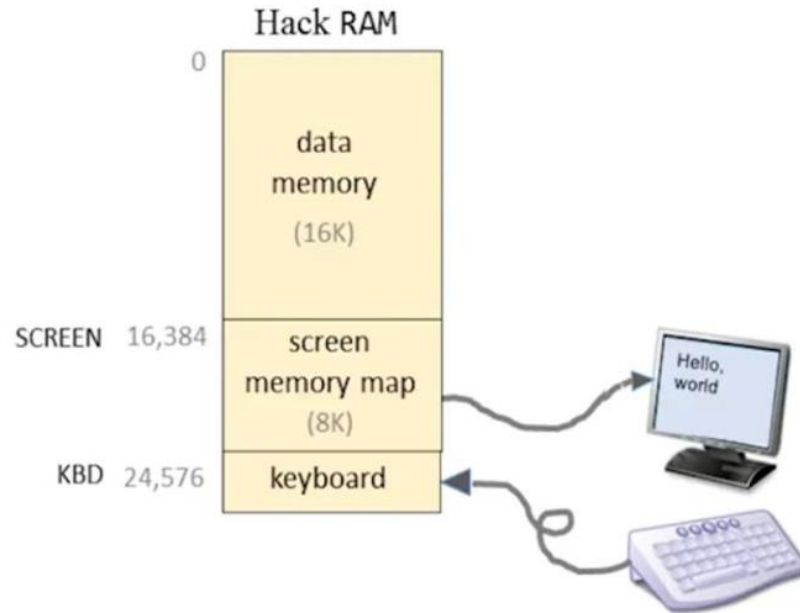




Screen	
(16384) 0	1111010100000000
1	0000000000000000
	⋮
31	0011000000000001
32	0000101000000000
33	0000000000000000
	⋮
63	0000000000000000
	⋮
8159	0000000000000000
8160	0000000000000000
	⋮
8191	1011010100000000

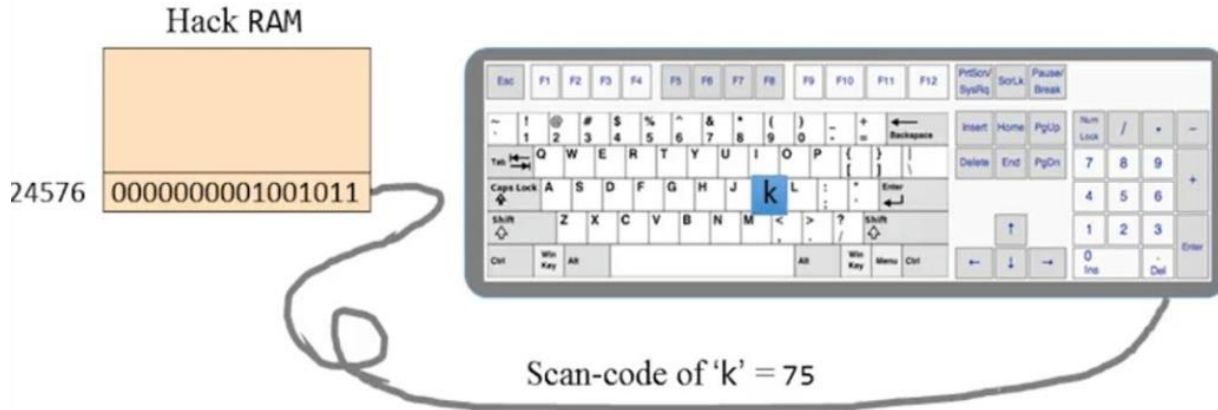






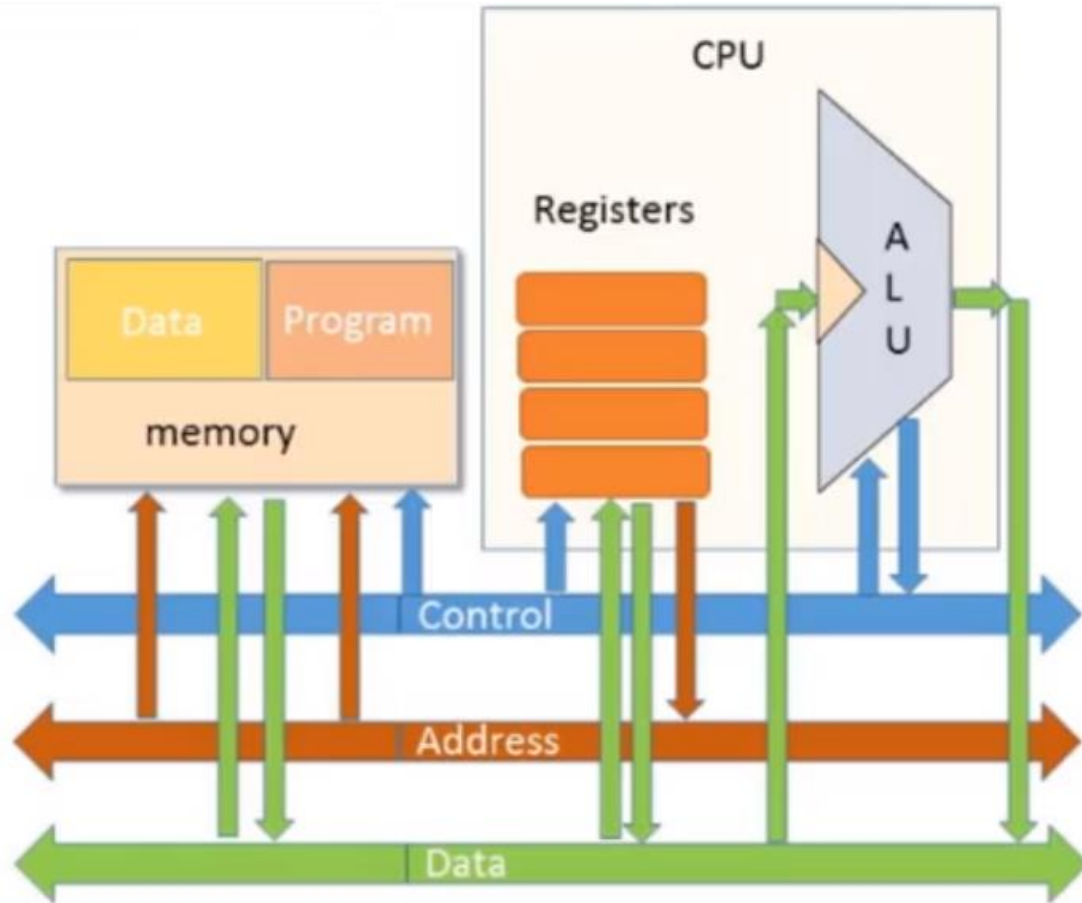
Hack language convention:

- SCREEN: base address of the screen memory map
- KBD: address of the keyboard memory map



To check which key is currently pressed:

- Read the contents of RAM[24576] (address KBD)
- If the register contains 0, no key is pressed
- Otherwise, the register contains the scan code of the currently pressed key.



THANKS!

Any questions?

