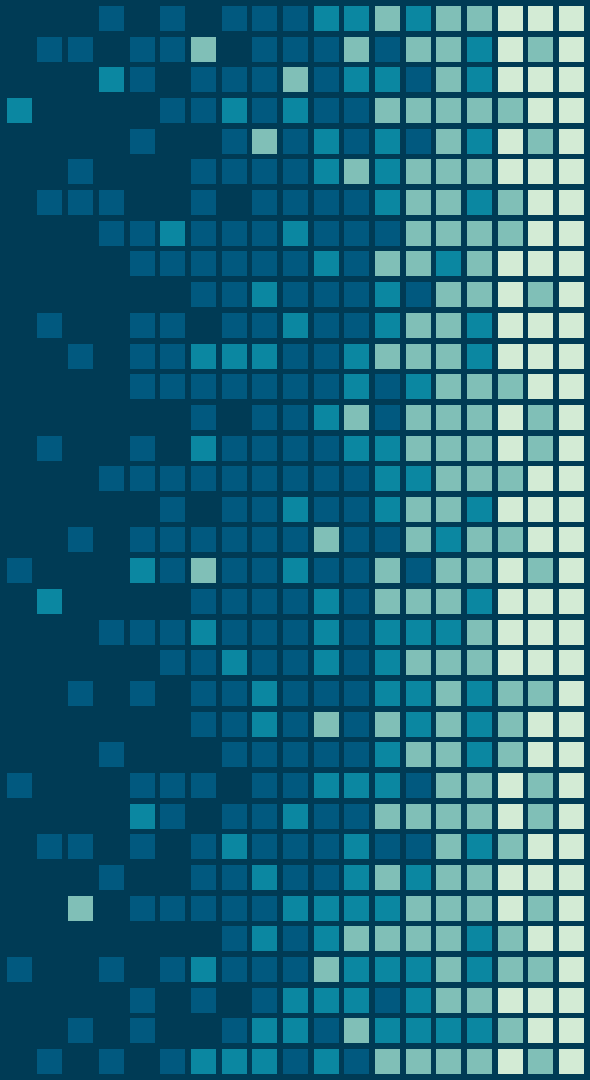
Arquitectura del Computador II

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

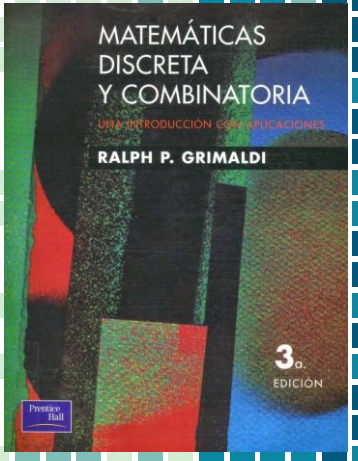
x	y	And(x,y)
0	0	0
0	1	0
1	0	0
1	1	1

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

Álgebra de Boole

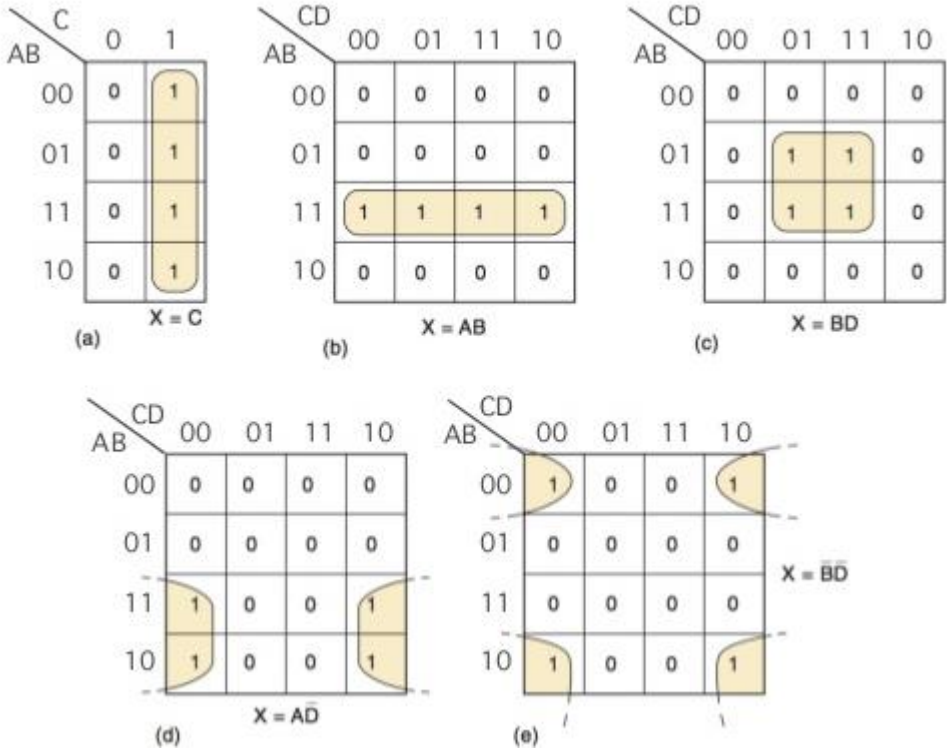
1. $A + 0 = A$
2. $A \cdot 1 = A$
3. $A + 1 = 1$
4. $A \cdot 0 = 0$
5. $A + A = A$
6. $A \cdot A = A$
7. $A + \overline{A} = 1$
8. $A \cdot \overline{A} = 0$
9. $\overline{\overline{A}} = A$
10. $A + B = B + A$ — 11. $A.B = B.A$ — **Lei comutativa**
12. $A + (B + C) = (A + B) + C$ — 13. $X(YZ) = (X.Y)Z$ — **Lei associativa**
14. $A \cdot (B + C) = AB + AC$ — 15. $A + BC = (A + B).(A + C)$ — **Lei distributiva**
16. $\overline{A + B} = \overline{A} \cdot \overline{B}$ — 17. $\overline{A \cdot B} = \overline{A} + \overline{B}$ — **DeMorgan**

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

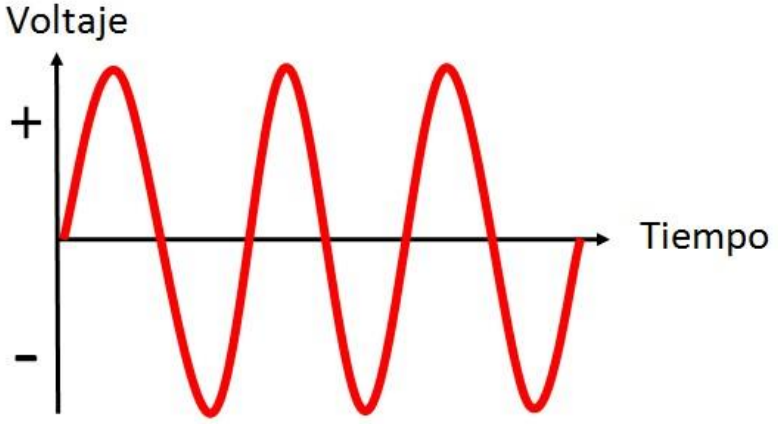


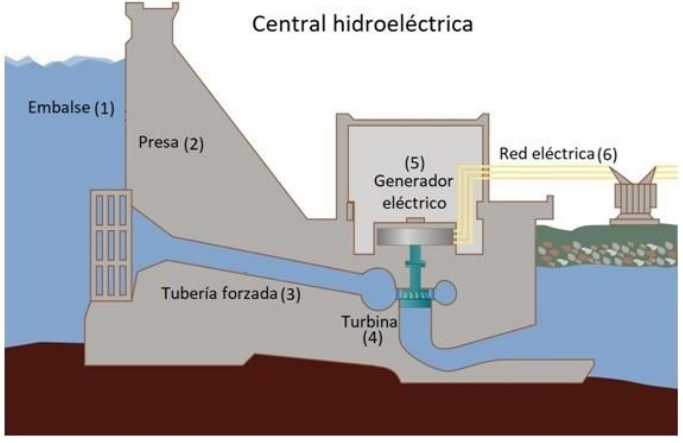
Mapas de Karnaugh

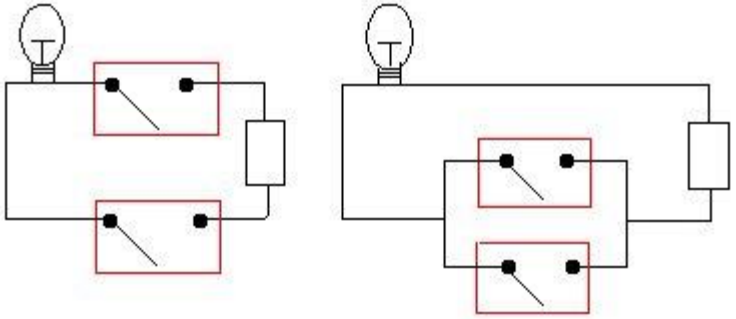
la variable que cambia es la que se desecha

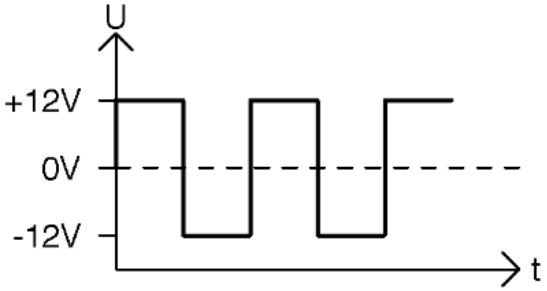


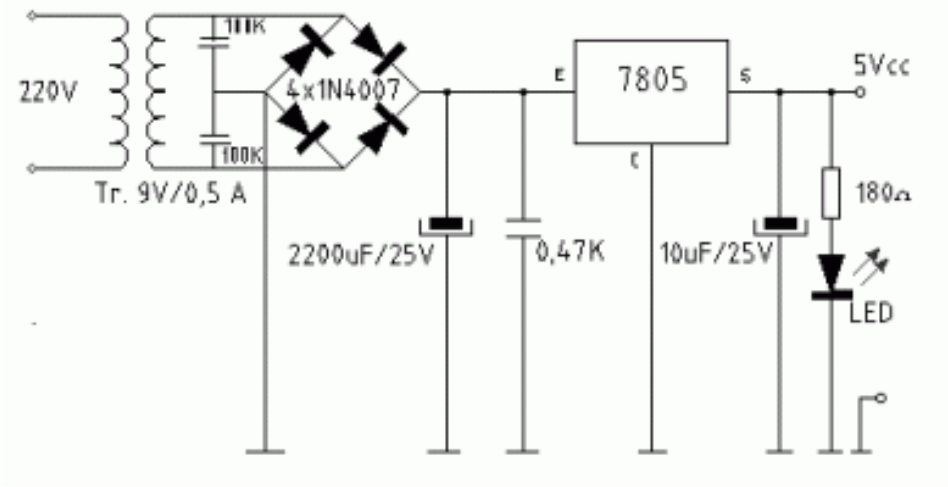
corriente alterna

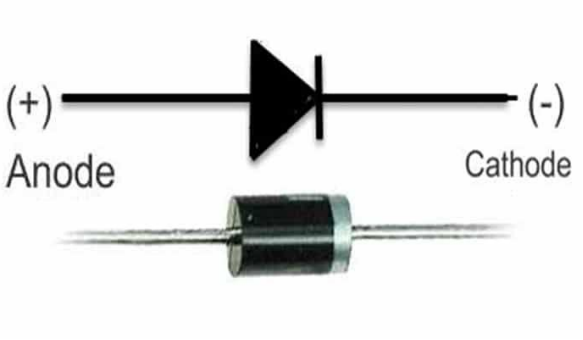


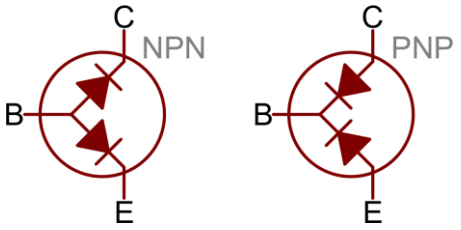


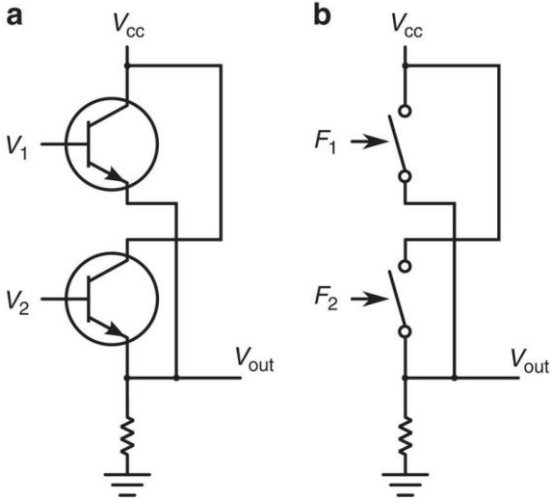



U
+12V
0V
-12V
t

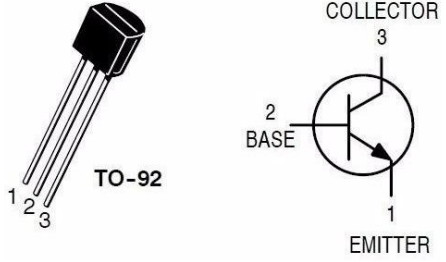

220V
100K
4x1N4007
100K
Tr. 9V/0,5 A
E
7805
S
C
5Vcc
180Ω
2200uF/25V
0,47K
10uF/25V
LED


(+)
(-)
Anode
Cathode

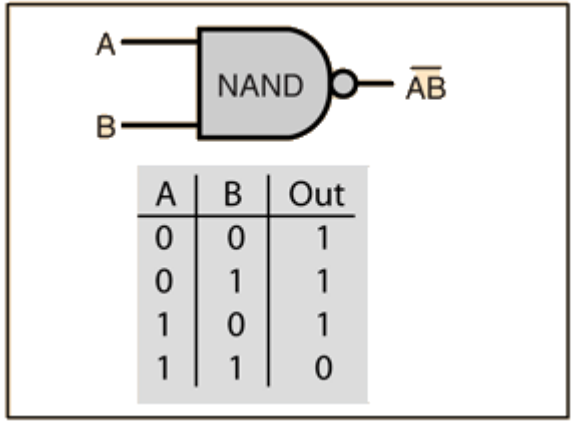

C
NPN
B
E
C
PNP
B
E


a
V_{cc}
V_1
V_2
V_{out}
b
V_{cc}
F_1
F_2
V_{out}

2N2222

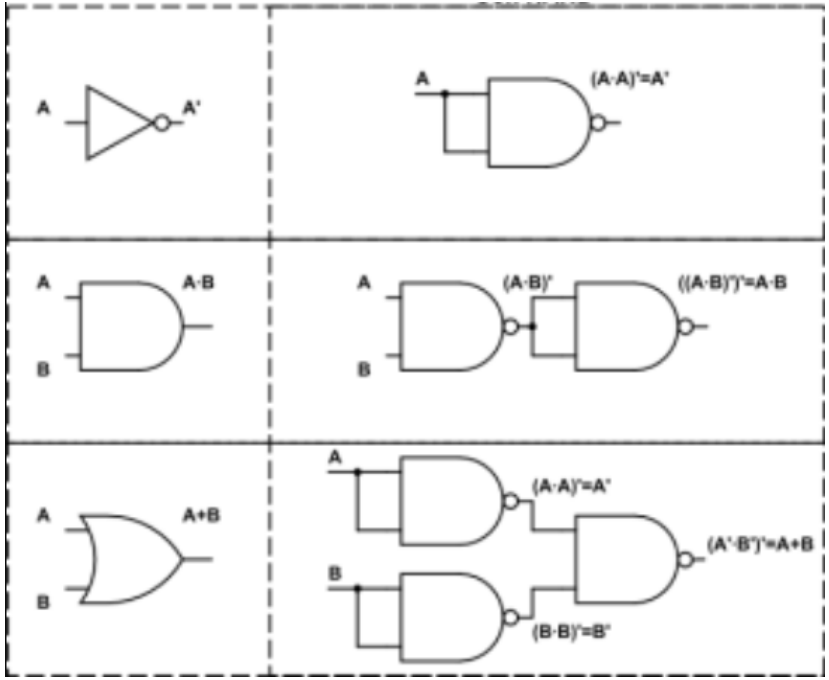

TO-92
1
2
3
COLLECTOR
3
2
BASE
1
EMITTER

NAND



A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

Compuertas básicas con NAND

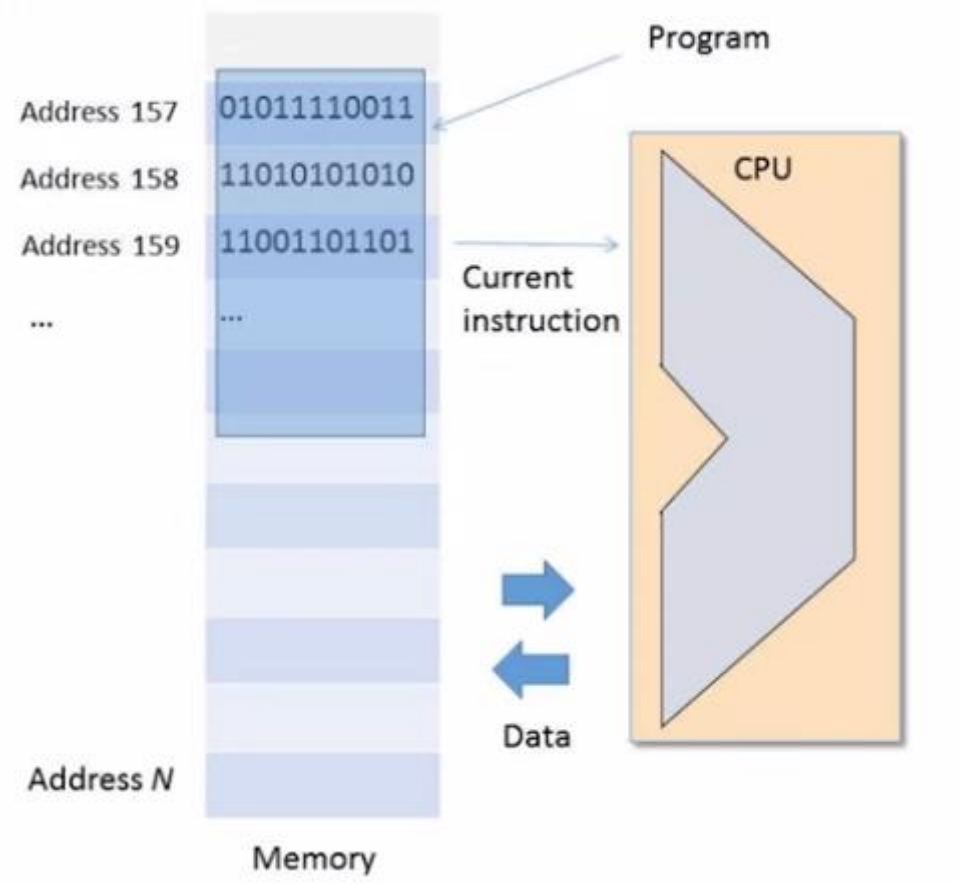


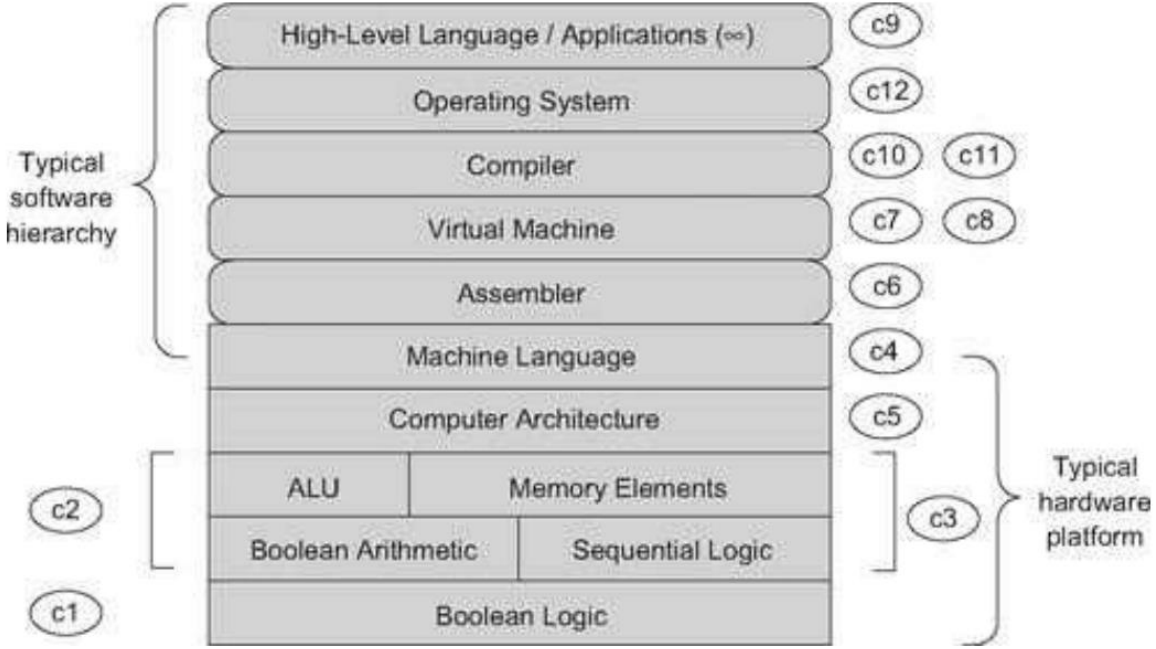
Theory

Universal Turing Machine

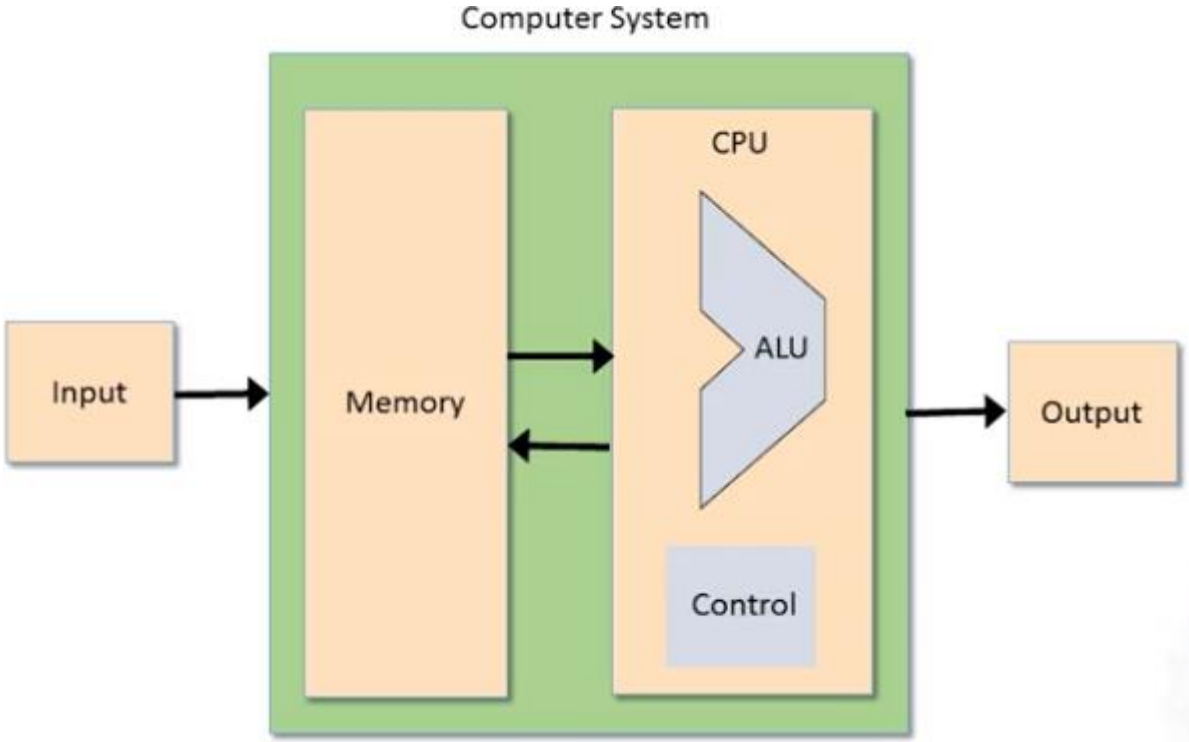
Practice

von Neumann Architecture


Program
Address 157
01011110011
Address 158
11010101010
Address 159
11001101101
...
...
CPU
Current
instruction
Data
Address N
Memory


High-Level Language / Applications (∞)
Operating System
Compiler
Virtual Machine
Assembler
Machine Language
Computer Architecture
ALU
Memory Elements
Boolean Arithmetic
Sequential Logic
Boolean Logic
Typical software hierarchy
Typical hardware platform
c1
c2
c3
c4
c5
c6
c7
c8
c9
c10
c11
c12

Von Neumann Architecture:



Ejercicio rápido:

Cuánto es:

01111101 +

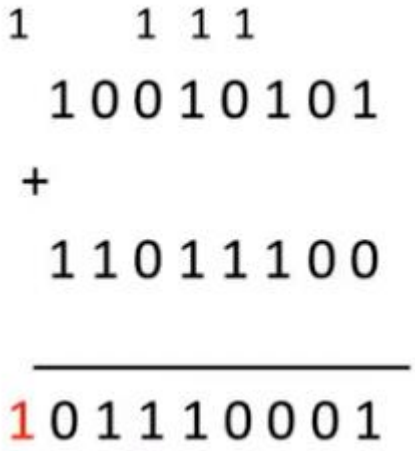
00110110

01111101 +

00110110

10110011

Overflow



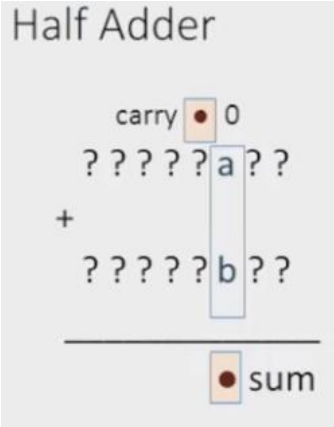
Exceder el tamaño de la palabra.

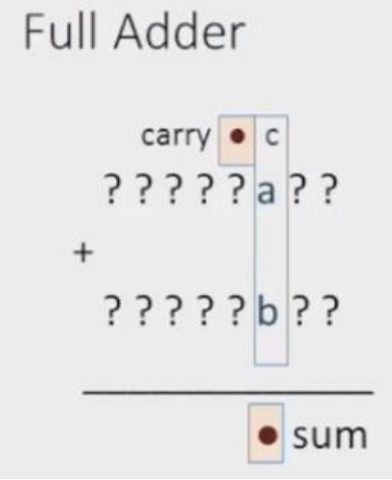
Sumadores:

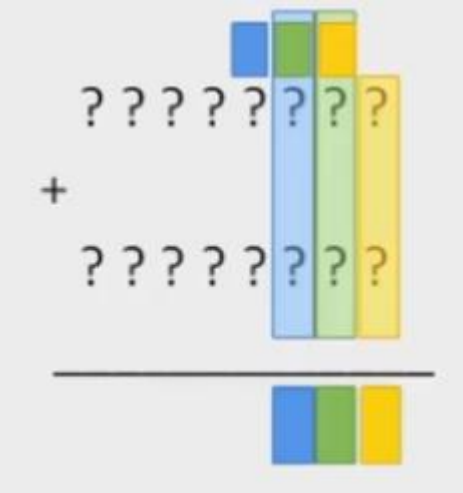
Medio Sumador – suma de 2 bits.

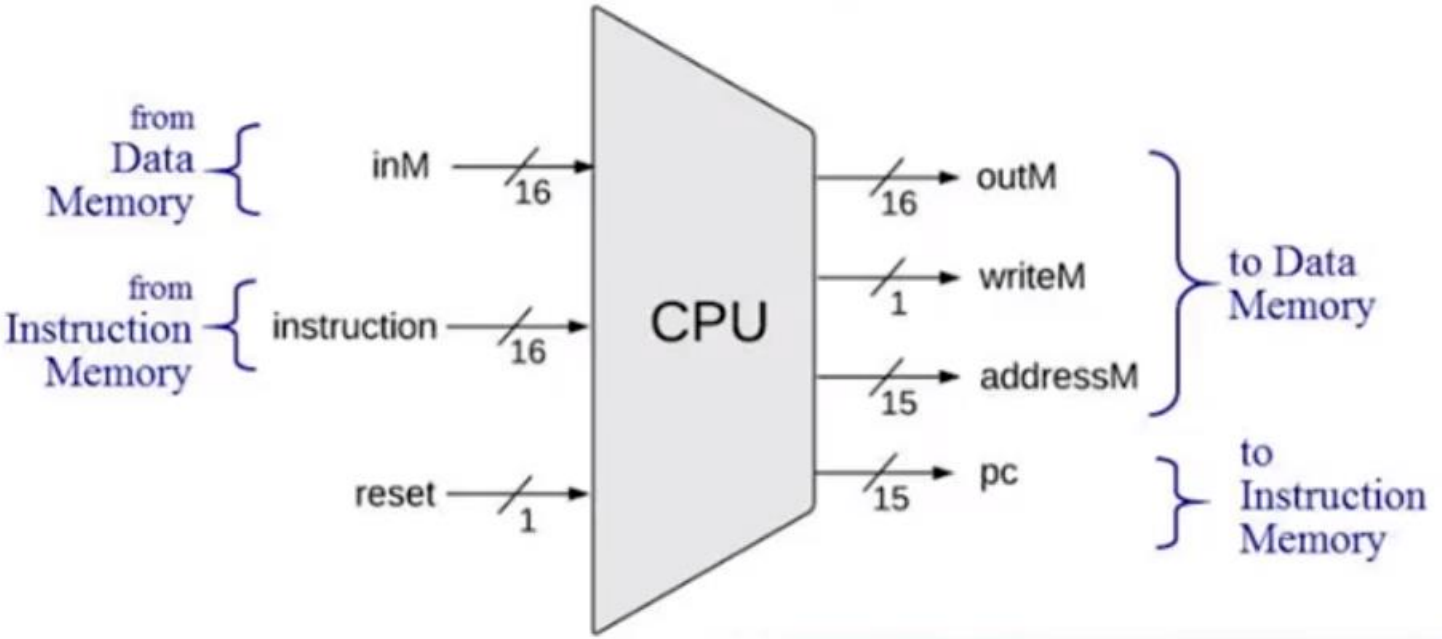
Sumador complete – Suma de 3 bits.

Sumador – Suma 2 números.







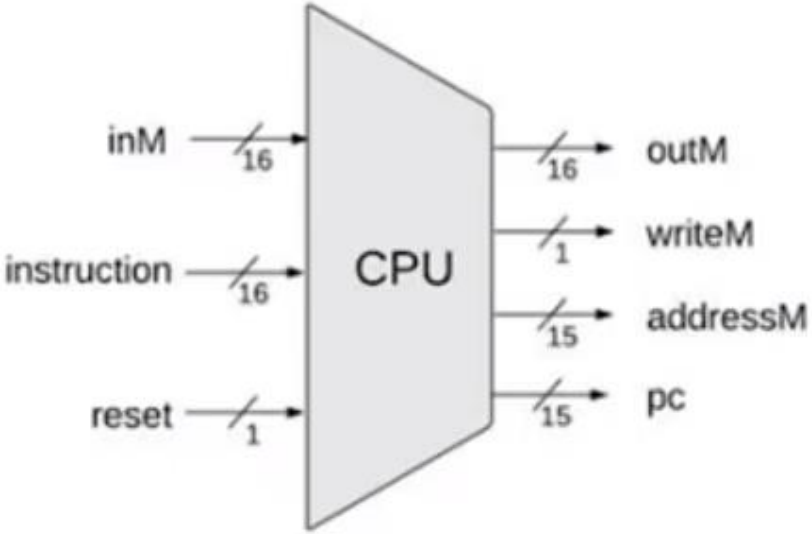

from
Data
Memory
from
Instruction
Memory
inM
16
instruction
16
reset
1
CPU
16
outM
1
writeM
15
addressM
15
pc
to Data
Memory
to
Instruction
Memory

Sample Hack instructions:

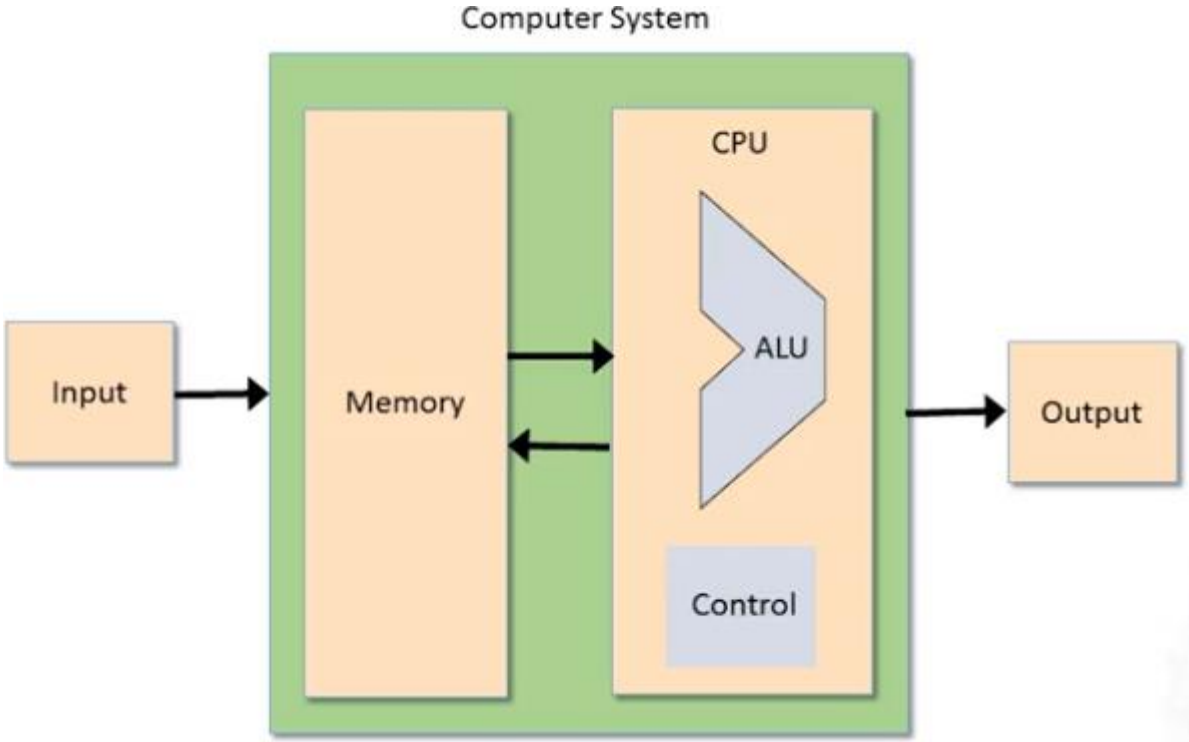
D = D-A

@17

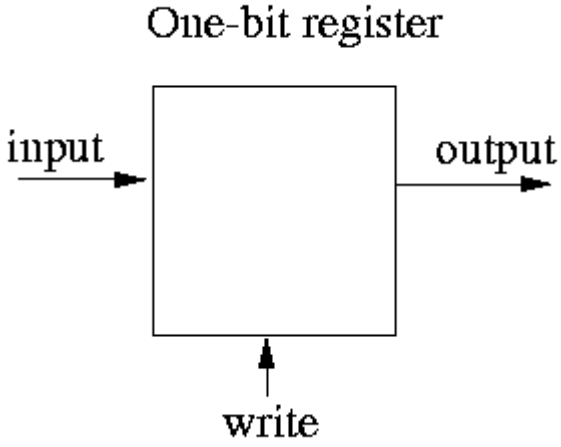
M = M+1



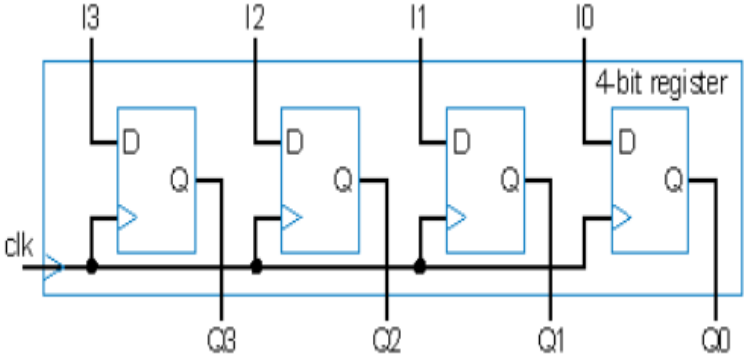
Von Neumann Architecture:

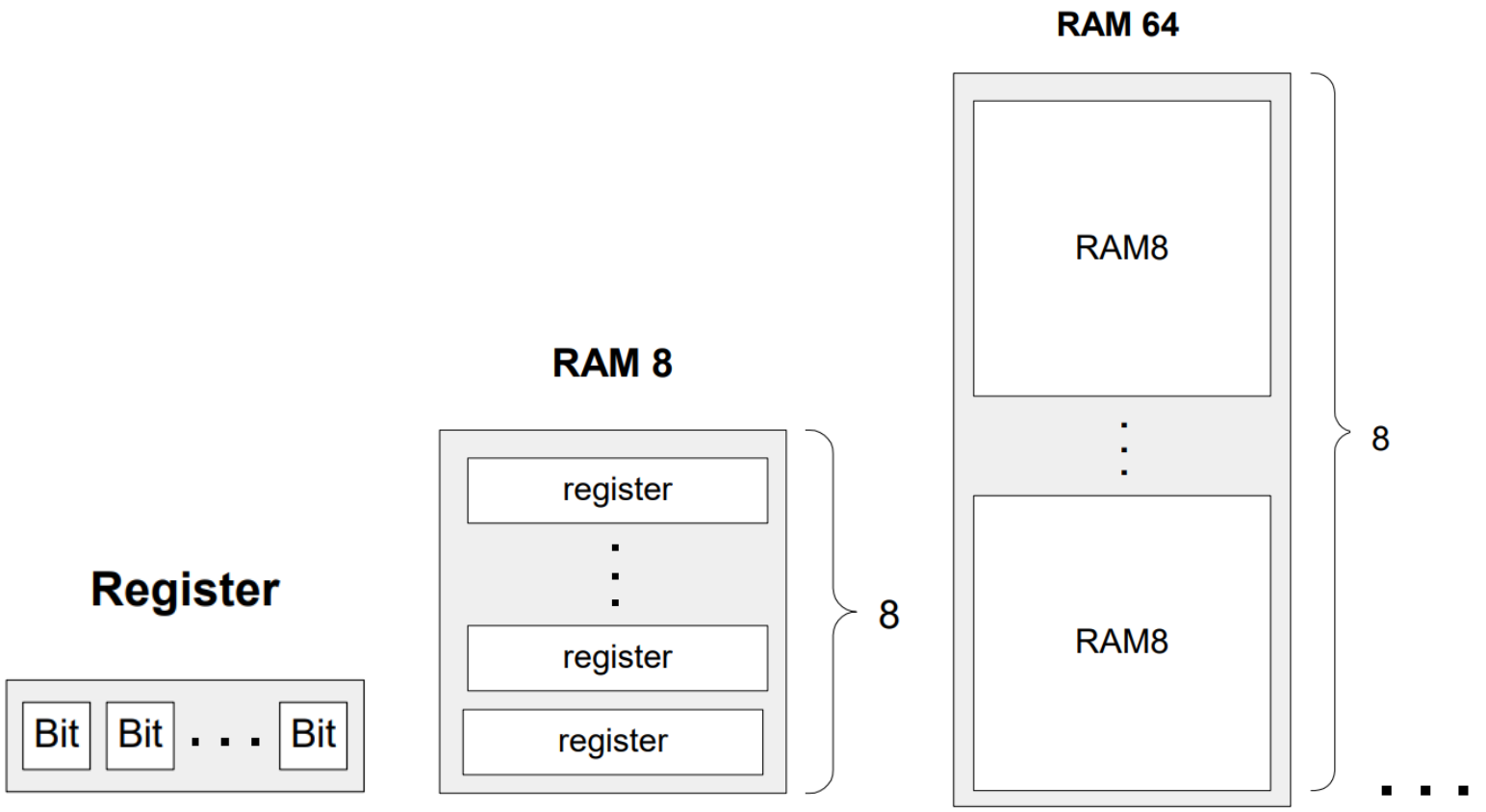


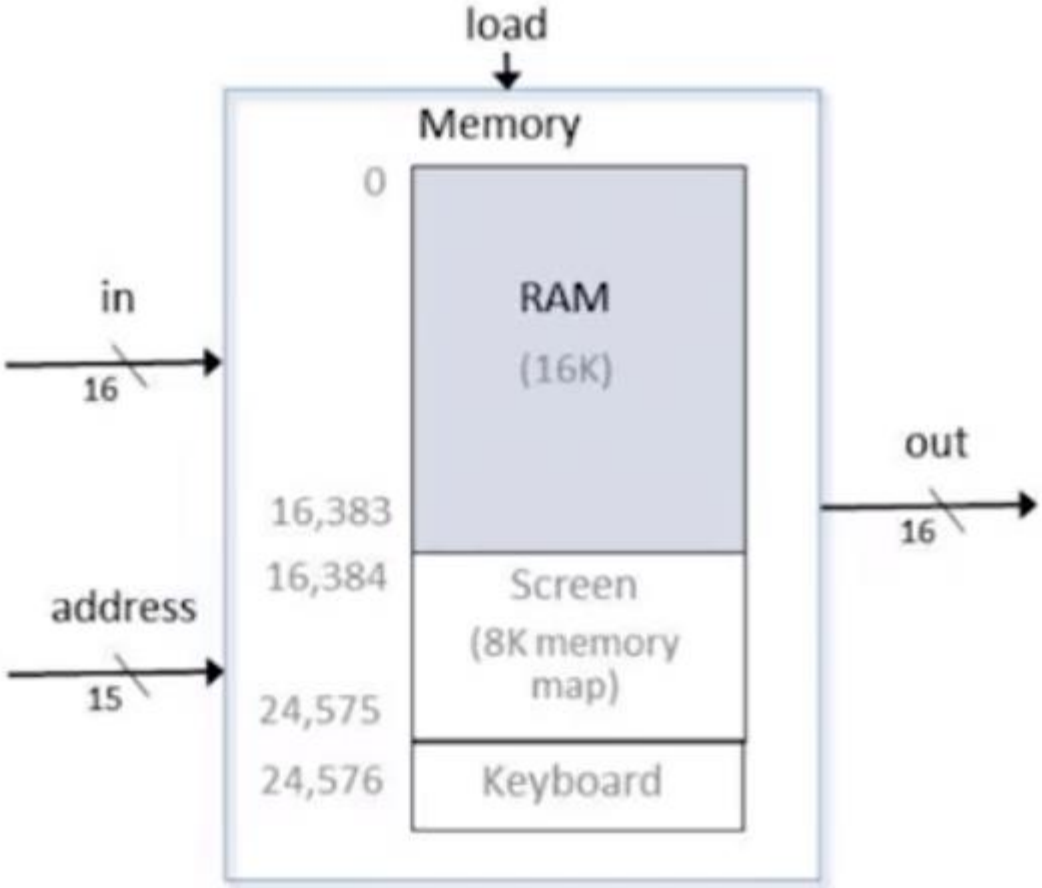
One Bit and Multi Bit Register

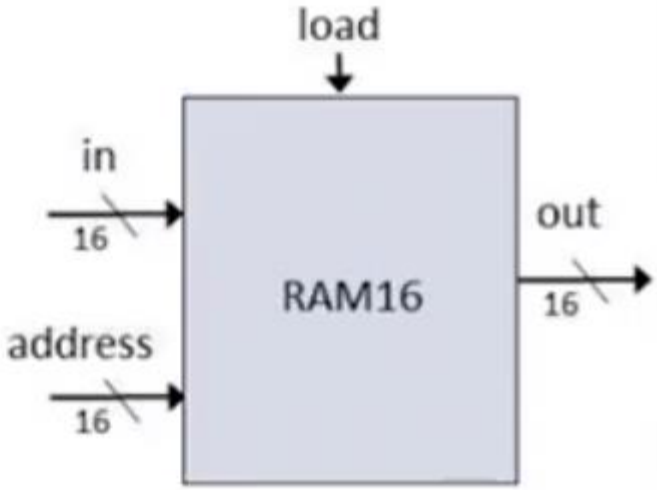


- Word (16bit, 32 ...)
- Register State (Qué tiene guardado el registro)

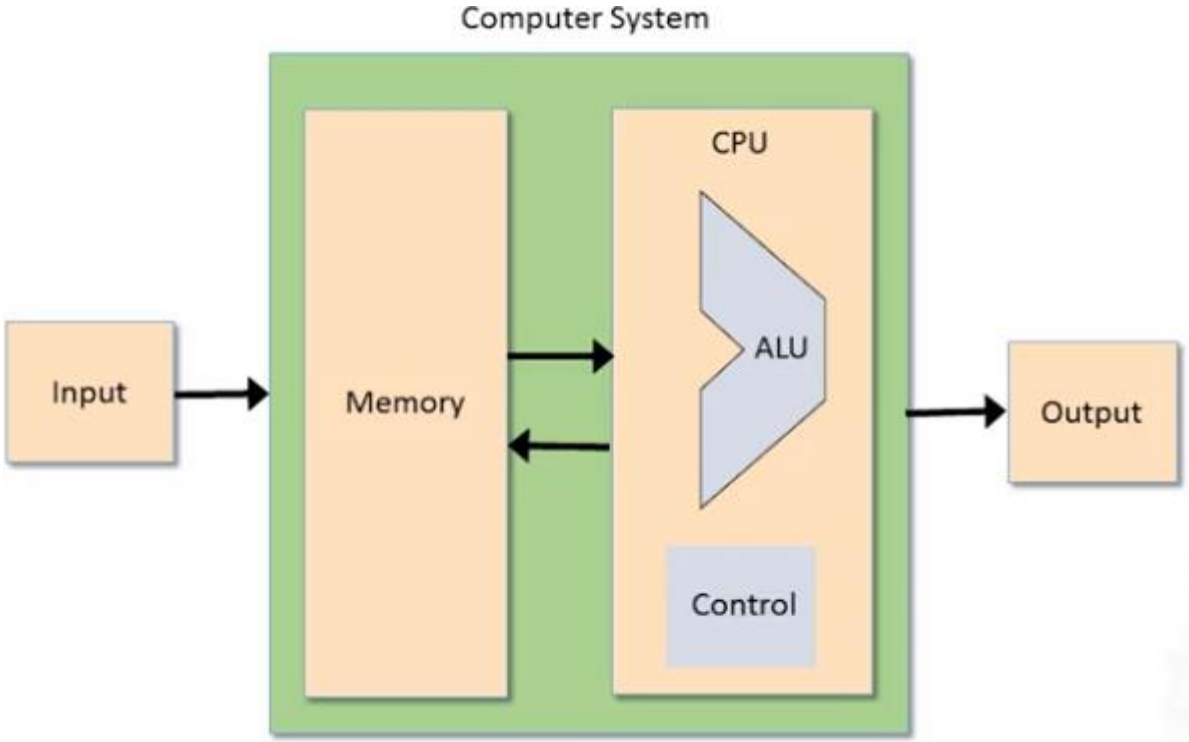


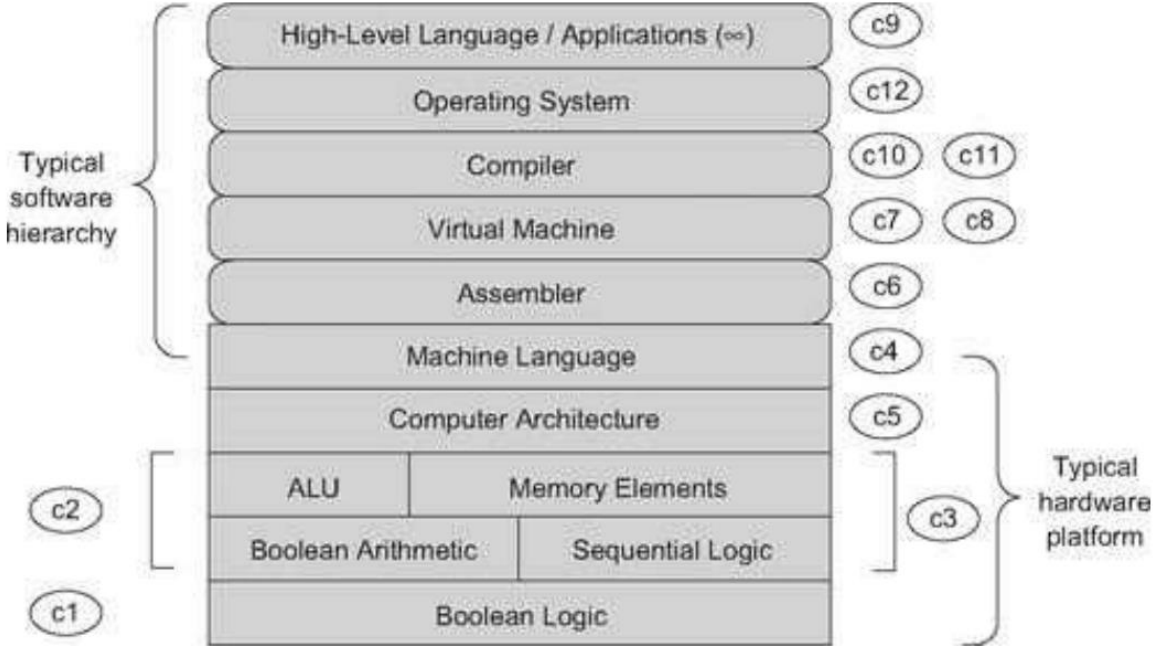

Register
Bit
Bit
Bit
RAM 8
register
register
register
8
RAM 64
RAM8
RAM8
8


load
Memory
0
RAM
(16K)
16,383
16,384
Screen
(8K memory
map)
24,575
24,576
Keyboard
in
16
address
15
out
16


load
in
16
address
16
RAM16
out
16

Von Neumann Architecture:



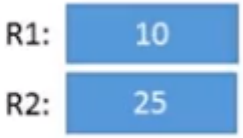

High-Level Language / Applications (∞)
Operating System
Compiler
Virtual Machine
Assembler
Machine Language
Computer Architecture
ALU
Memory Elements
Boolean Arithmetic
Sequential Logic
Boolean Logic
Typical software hierarchy
Typical hardware platform
c1
c2
c3
c4
c5
c6
c7
c8
c9
c10
c11
c12

Símbolos

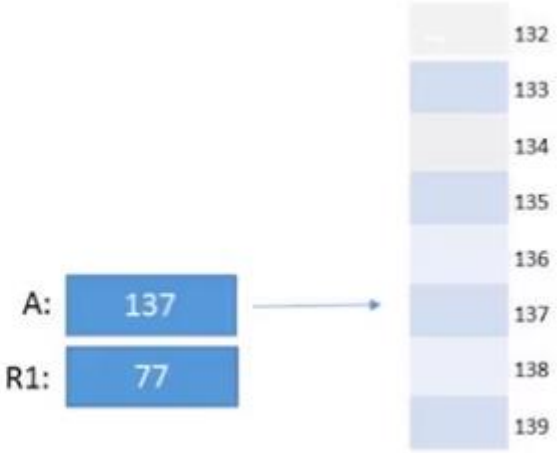
Machine Language	Assembly Language
10100001 10000001	ADD 1, Mem[129]

Registros

- Data Registers
  - Add R1, R2



- Address Registers
  - Store R1, @A



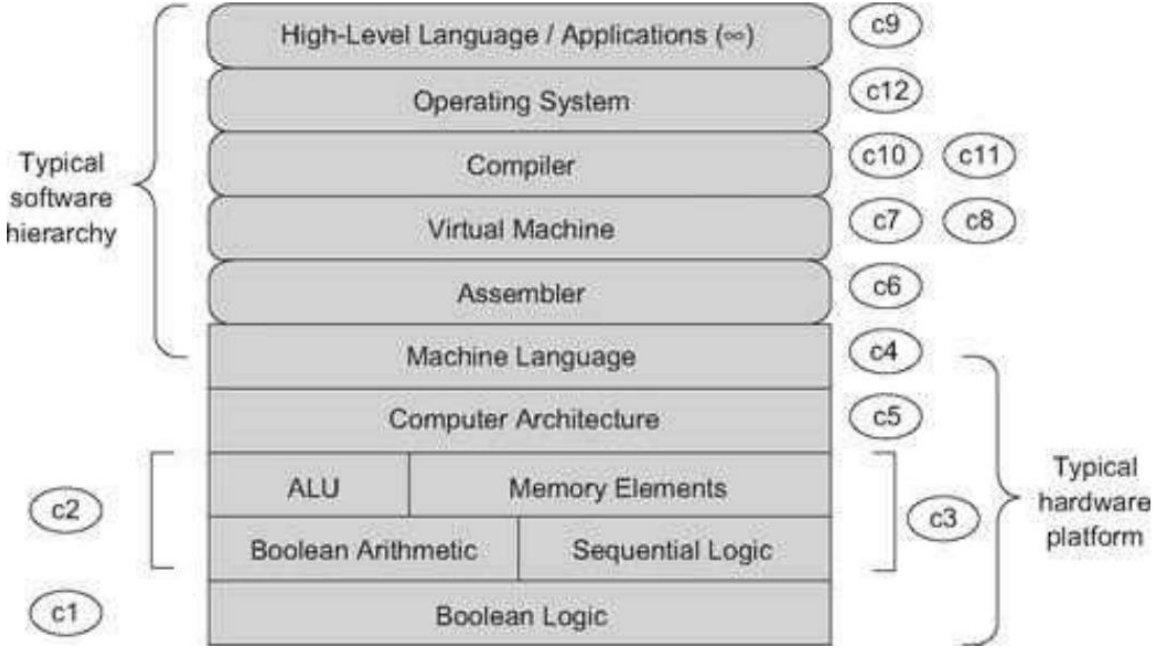
Representaciones

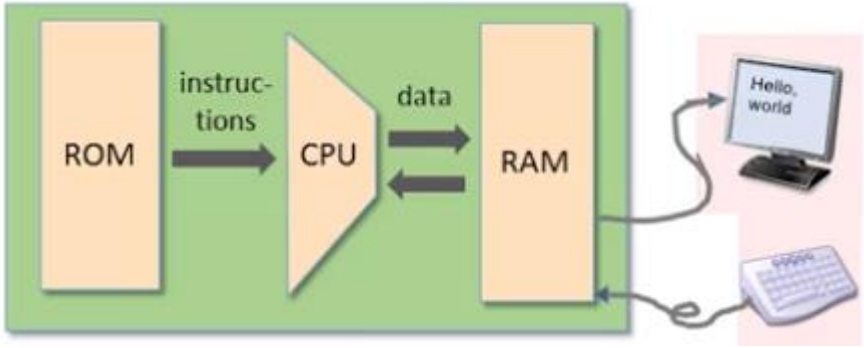
Symbolic:

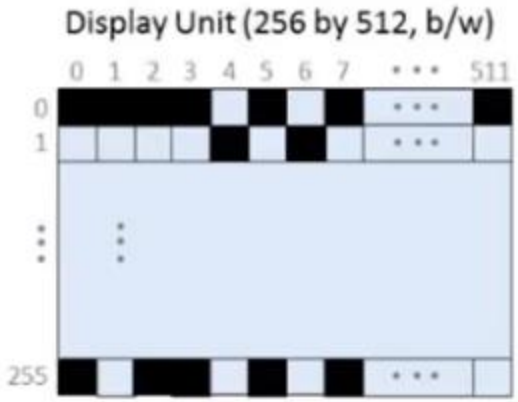
```
@17
D+1;JLE
```

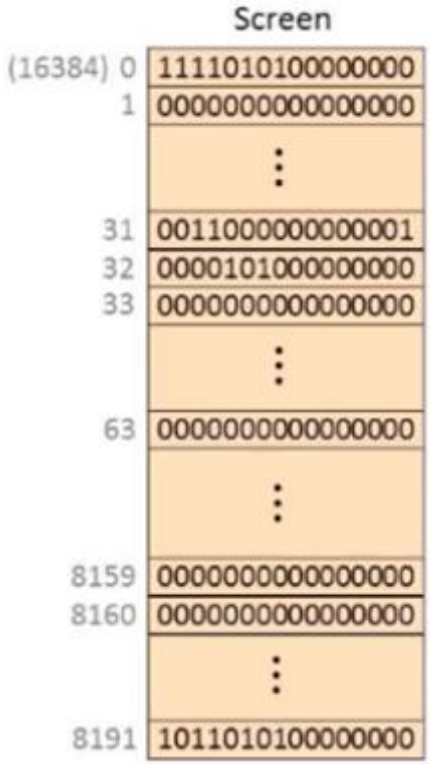
Binary:

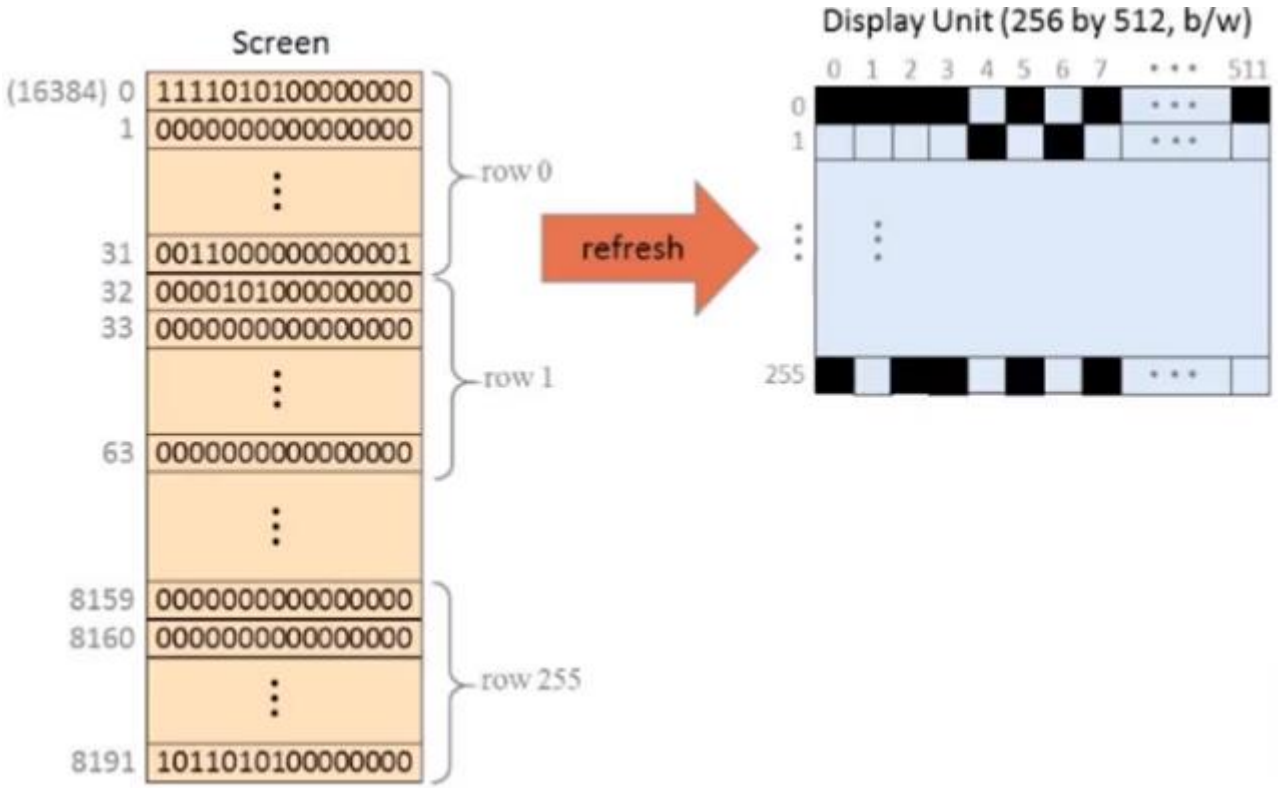
```
0000000000010001
1110011111000110
```

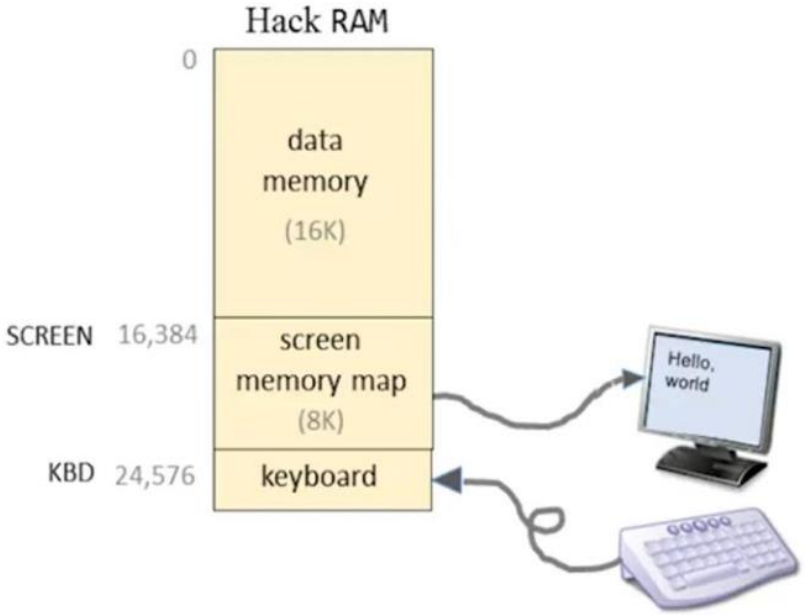

High-Level Language / Applications (∞)
Operating System
Compiler
Virtual Machine
Assembler
Machine Language
Computer Architecture
ALU
Memory Elements
Boolean Arithmetic
Sequential Logic
Boolean Logic
Typical software hierarchy
Typical hardware platform
c1
c2
c3
c4
c5
c6
c7
c8
c9
c10
c11
c12


ROM
instruc-
tions
CPU
data
RAM
Hello,
world


Display Unit (256 by 512, b/w)
0 1 2 3 4 5 6 7 • • • 511
0
1
255

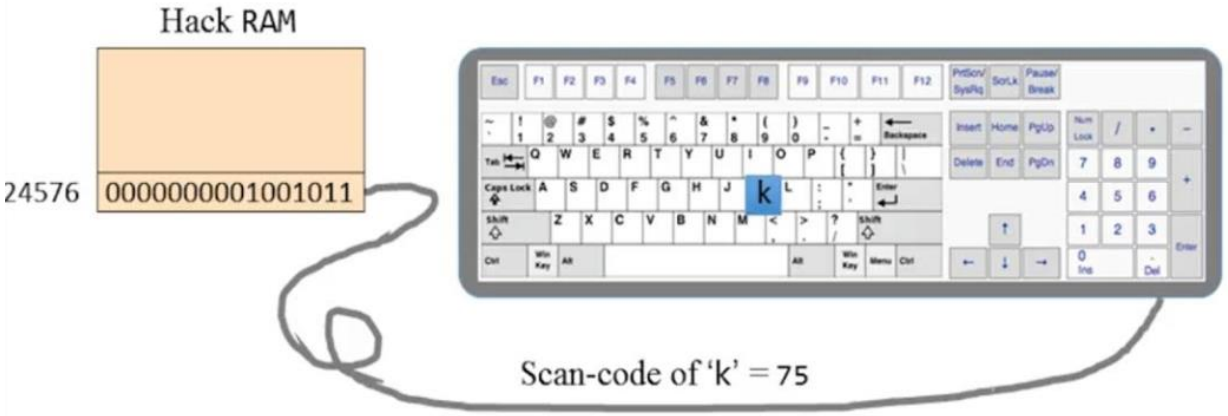

Screen
(16384) 0
1111010100000000
1
0000000000000000
31
0011000000000001
32
0000101000000000
33
0000000000000000
63
0000000000000000
8159
0000000000000000
8160
0000000000000000
8191
1011010100000000


Screen
(16384) 0
1
31
32
33
63
8159
8160
8191
1111010100000000
0000000000000000
0011000000000001
0000101000000000
0000000000000000
0000000000000000
0000000000000000
1011010100000000
row 0
row 1
row 255
refresh
Display Unit (256 by 512, b/w)
0 1 2 3 4 5 6 7 • • • 511
0
1
255



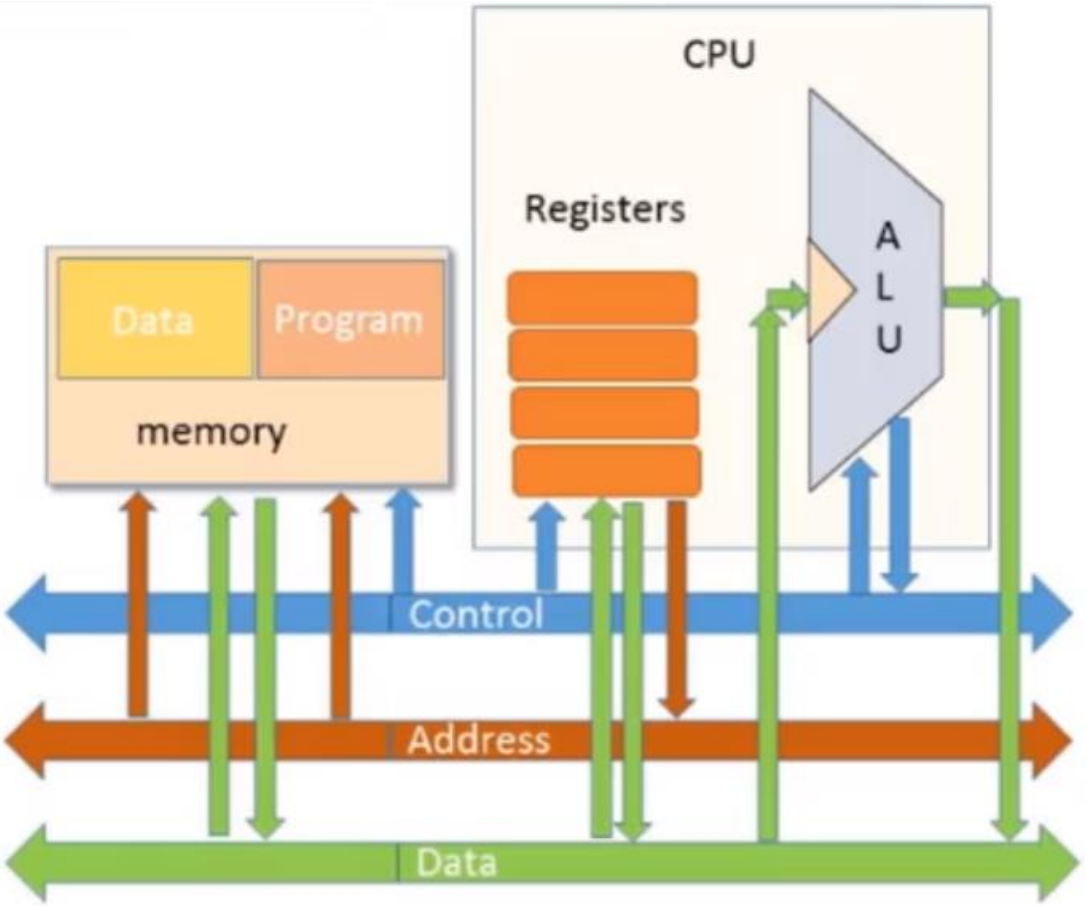
Hack language convention:

- 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.


CPU
Registers
A
L
U
Data
Program
memory
Control
Address
Data

THANKS!

Any questions?