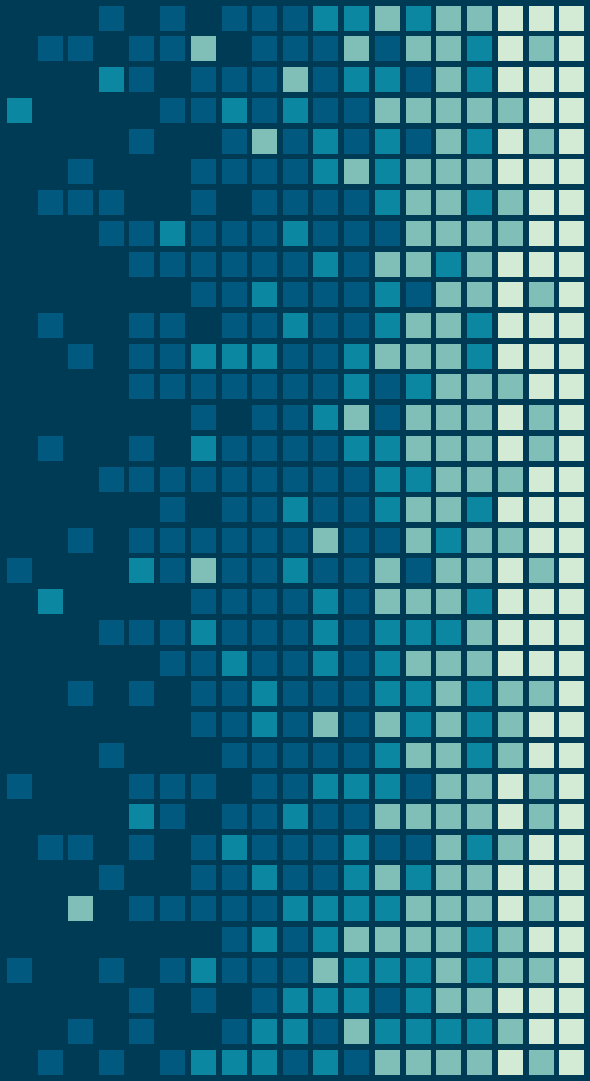
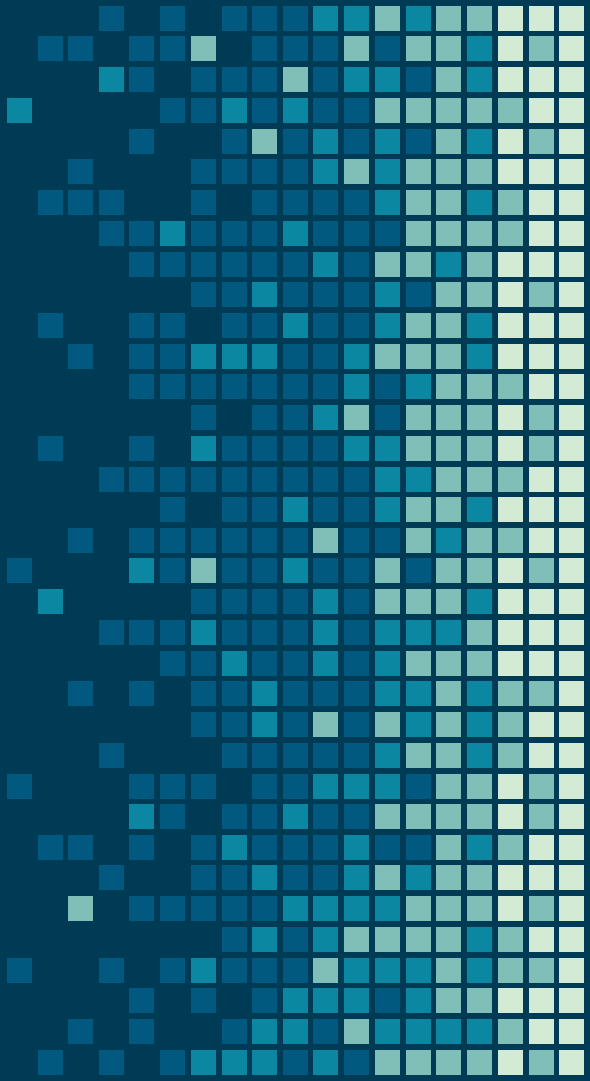
Arquitectura del Computador II

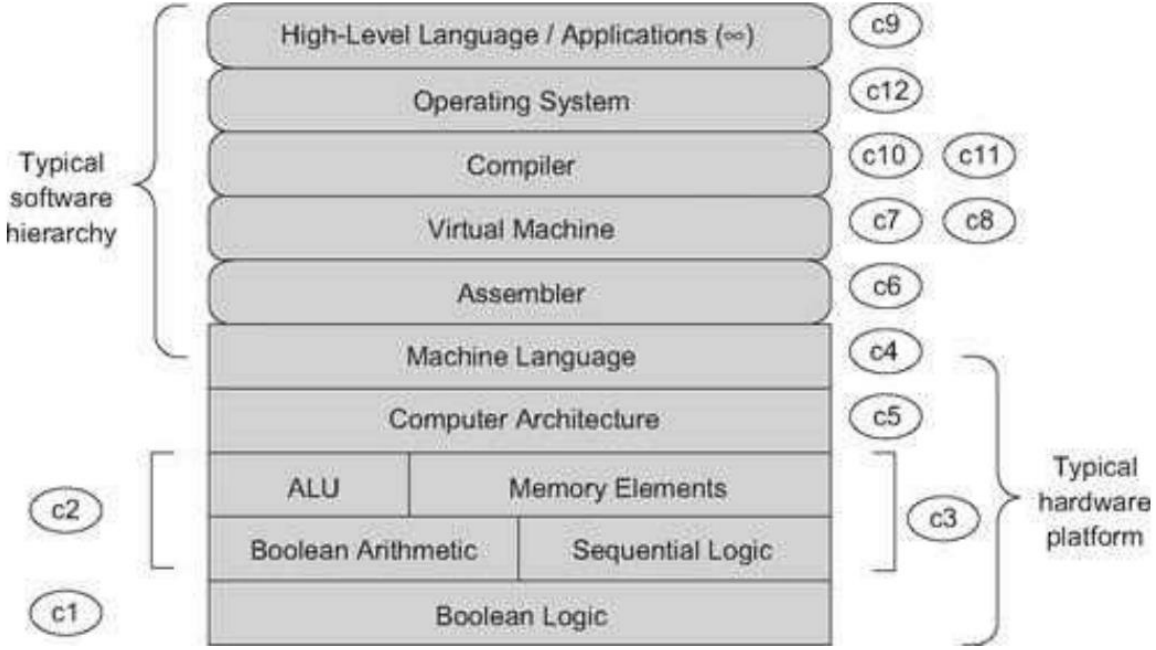
Introducción.



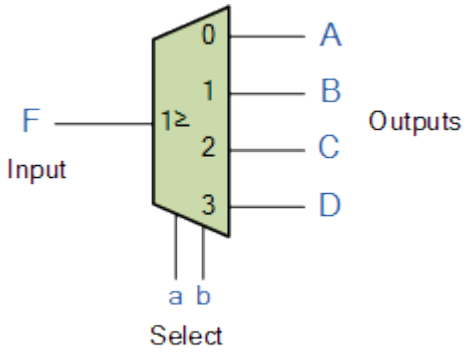
Arquitectura del Computador I

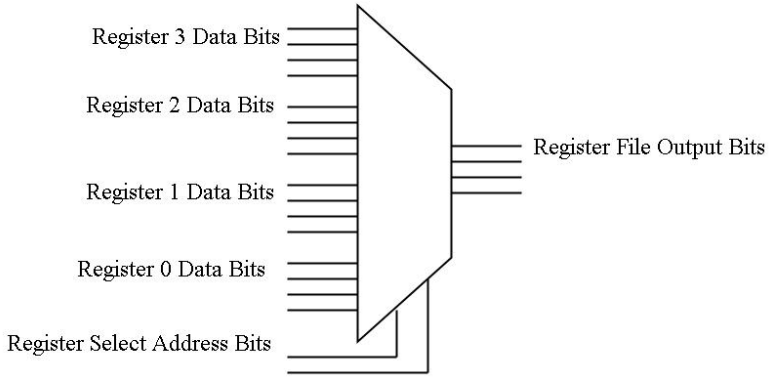
(recap)

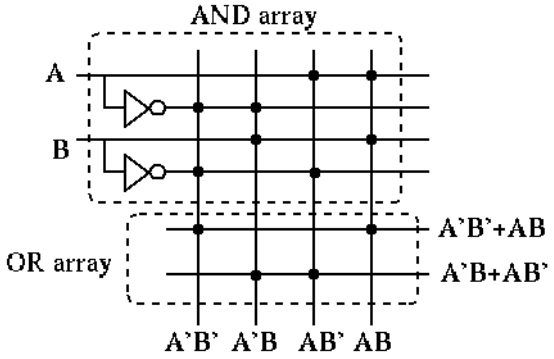



High-Level Language / Applications (∞)
Operating System
Compiler
Virtual Machine
Assembler
Machine Language
Computer Architecture
ALU
Memory Elements
Boolean Arithmetic
Sequential Logic
Boolean Logic
Typical software hierarchy
Typical hardware platform
c1
c2
c3
c4
c5
c6
c7
c8
c9
c10
c11
c12

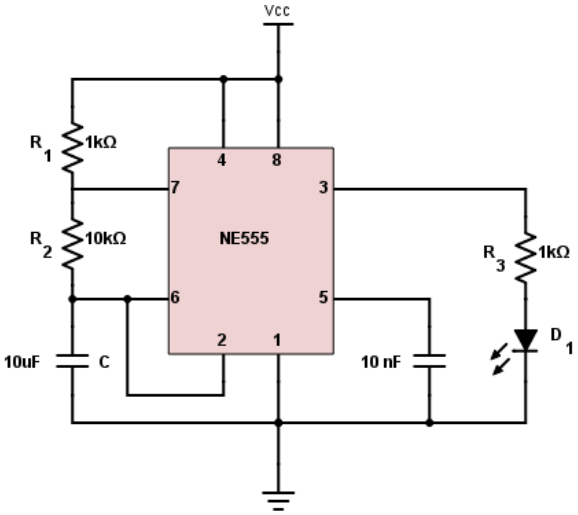
- Lógica combinacional
  - Sumadores
  - Sustractores
  - Comparadores de magnitudes
  - Decodificadores
  - Multiplexores
  - Demultiplexores
  - Read Only Memory (ROM)
  - Programmable Logic Array (PLA)

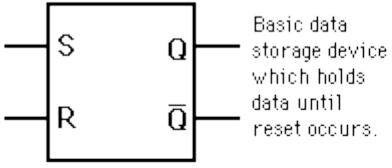






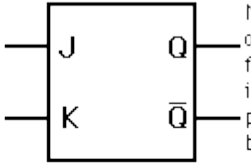
- Lógica secuencial
  - Flip flops
  - Circuitos secuenciales
  - Registros
  - Contadores sincrónicos
  - Secuencias de tiempo
  - Unidad de memoria
  - Random Access Memory (RAM)





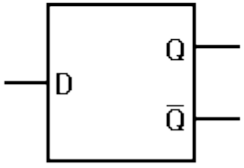
Set/Reset Type

Basic data storage device which holds data until reset occurs.



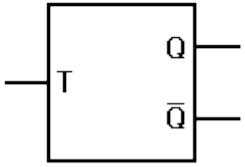
J-K Type

Most versatile of basic flip-flops. Two input behavior plus toggle type behavior.



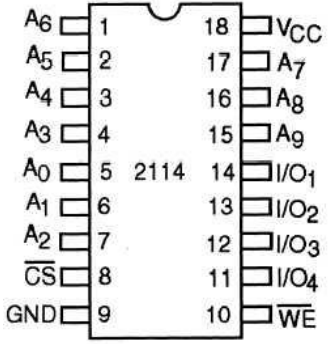
D-Type

Basic data flip-flop has advantage of simpler wiring than J-K version.



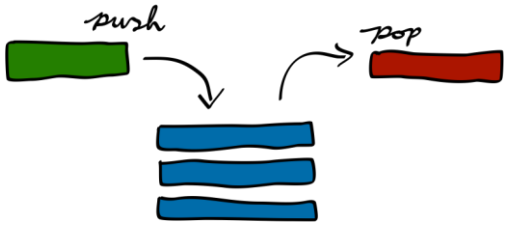
Toggle type

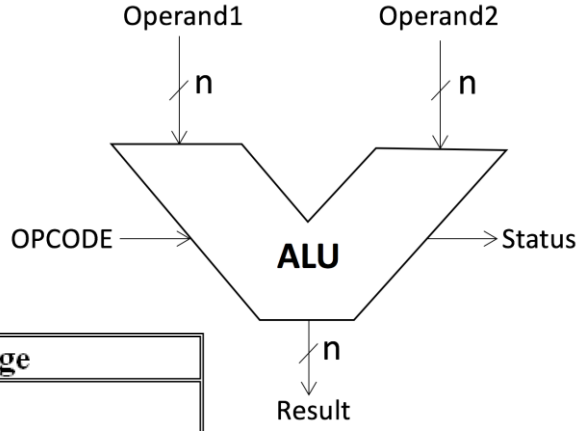
"Toggle" action is useful for counters. Can be constructed from any of the basic flip-flops.



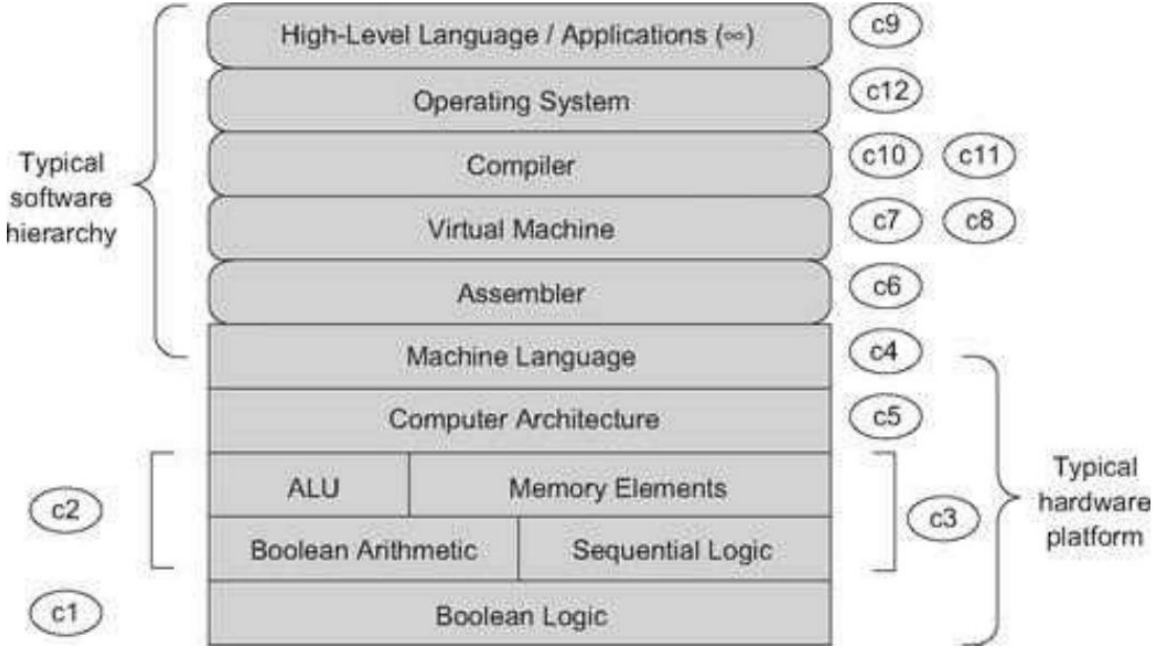
35NVM026

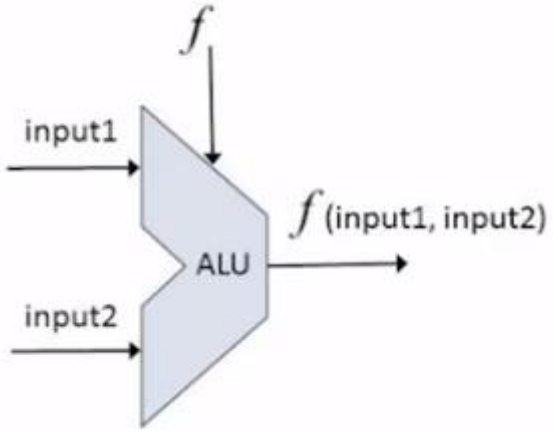
- Procesamiento
  - Logical arithmetic unit (ALU)
  - Procesamiento
  - Secuencia de microprograma
  - Ejecución de instrucciones
  - Organización de un microprocesador
  - Pila, subrutina e interrupción.
  - Organización de la memoria
  - Interconexión de entrada y salida





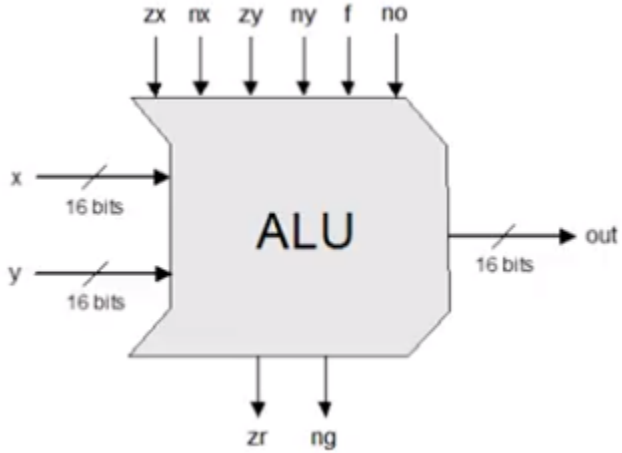
Address	Machine Language				Assembly Language
0000 0000	0000	0000	0000	0000	TOTAL .BLOCK 1
0000 0001	0000	0000	0000	0010	ABC .WORD 2
0000 0010	0000	0000	0000	0011	XYZ .WORD 3
0000 0011	0001	1101	0000	0001	LOAD REGD, ABC
0000 0100	0001	1110	0000	0010	LOAD REGE, XYZ
0000 0101	0101	1111	1101	1110	ADD REGF, REGD, REGE
0000 0110	0010	1111	0000	0000	STORE REGF, TOTAL
0000 0111	1111	0000	0000	0000	HALT


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

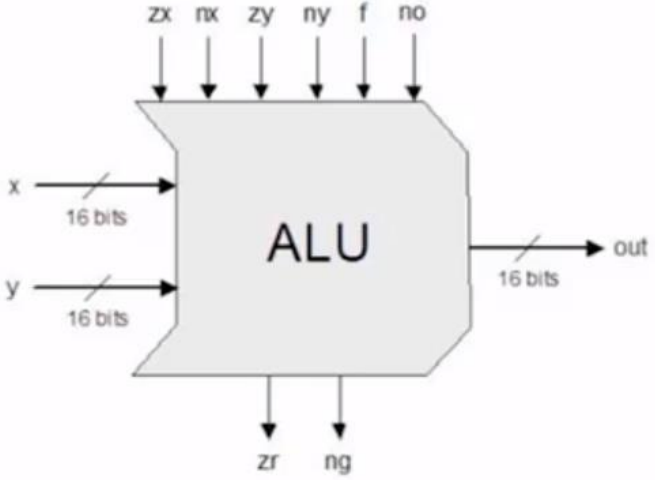


- Arithmetic operations: integer addition, multiplication, division, ...
- logical operations: And, Or, Xor, …

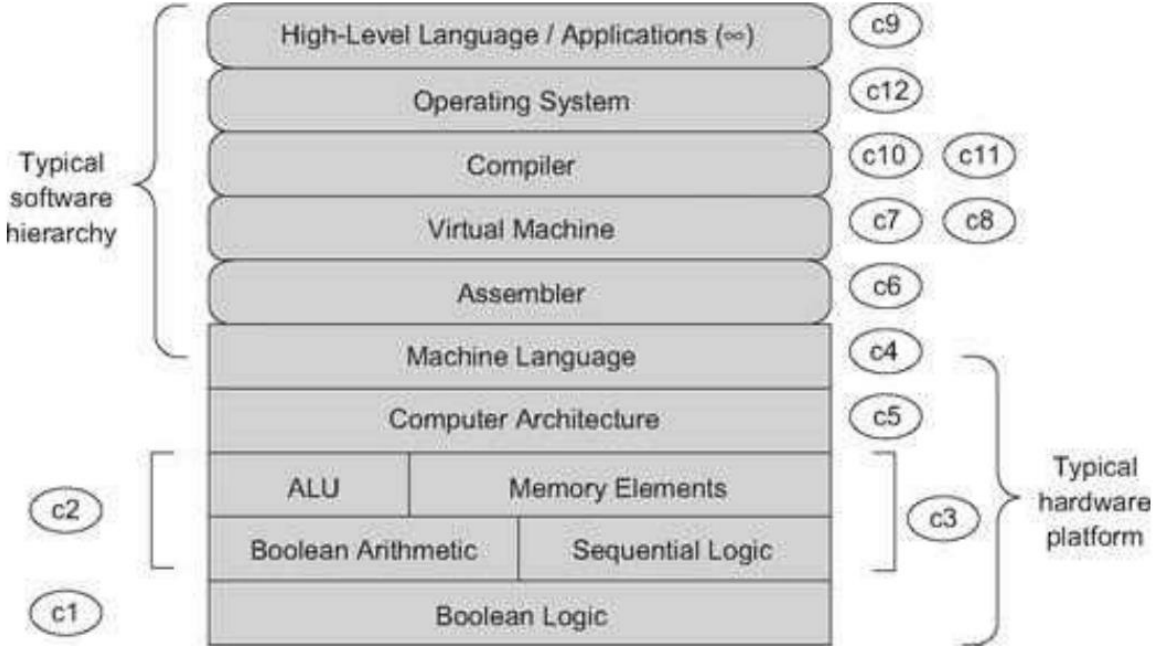
Hack ALU:

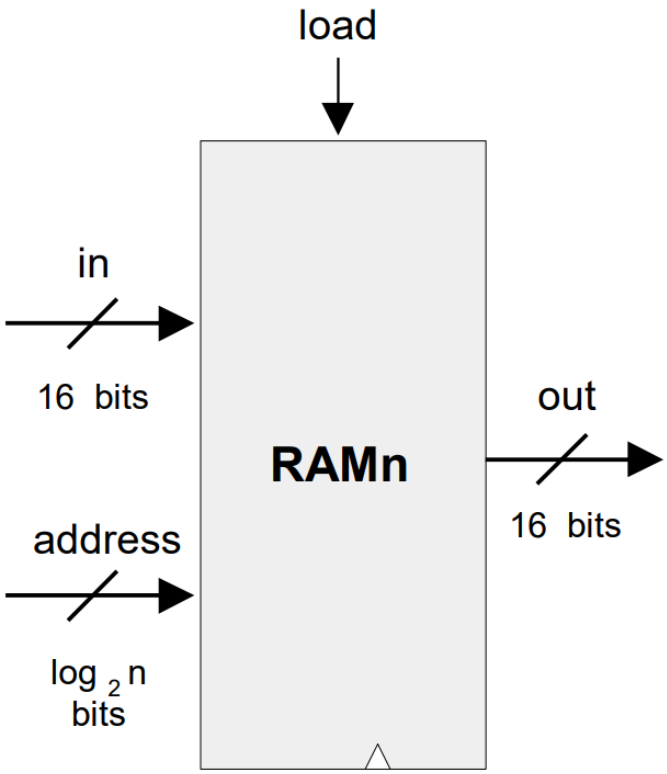


pre-setting the x input		pre-setting the y input		selecting between computing + or &	post-setting the output	Resulting ALU output
zx	nx	zy	ny	f	no	out
if zx then x=0	if nx then x=!x	if zy then y=0	if ny then y=!y	if f then out=x+y else out=x&y	if no then out=!out	out(x,y)=



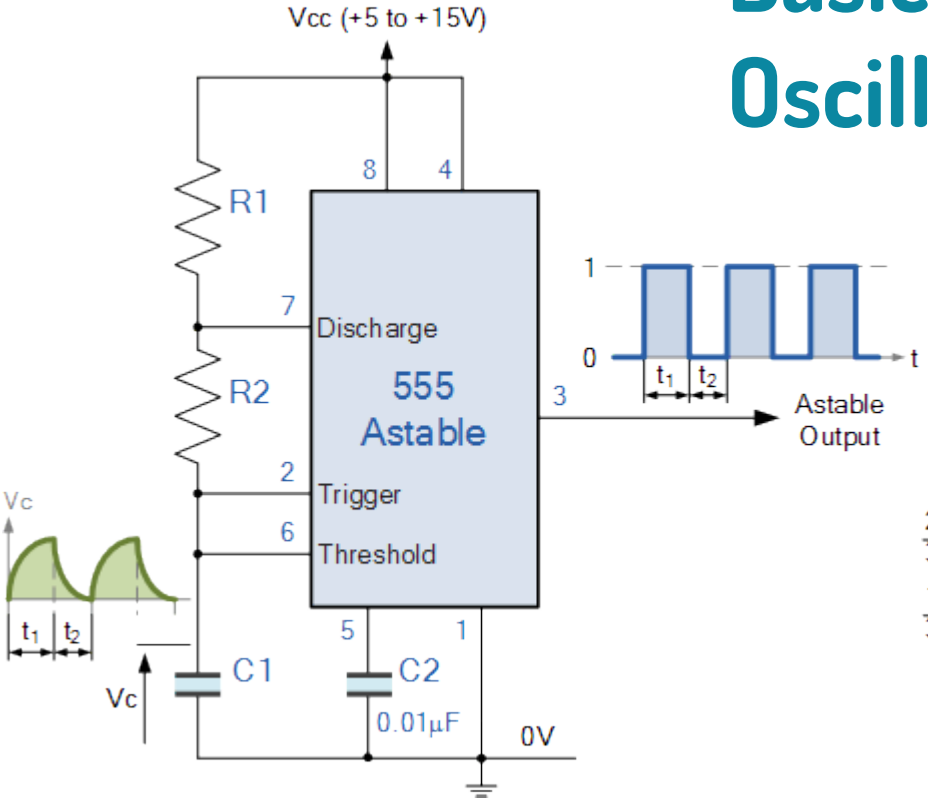
pre-setting the x input		pre-setting the y input		selecting between computing + or &	post-setting the output	Resulting ALU output
zx	nx	zy	ny	f	no	out
if zx then x=0	if nx then x=!x	if zy then y=0	if ny then y=!y	if f then out=x+y else out=x&y	if no then out=!out	out(x,y)=
1	0	1	0	1	0	0
1	1	1	1	1	1	1
1	1	1	0	1	0	-1
0	0	1	1	0	0	x
1	1	0	0	0	0	y
0	0	1	1	0	1	!x
1	1	0	0	0	1	!y
0	0	1	1	1	1	-x
1	1	0	0	1	1	-y
0	1	1	1	1	1	x+1
1	1	0	1	1	1	y+1
0	0	1	1	1	0	x-1
1	1	0	0	1	0	y-1
0	0	0	0	1	0	x+y
0	1	0	0	1	1	x-y
0	0	0	1	1	1	y-x
0	0	0	0	0	0	x&y
0	1	0	1	0	1	x|y

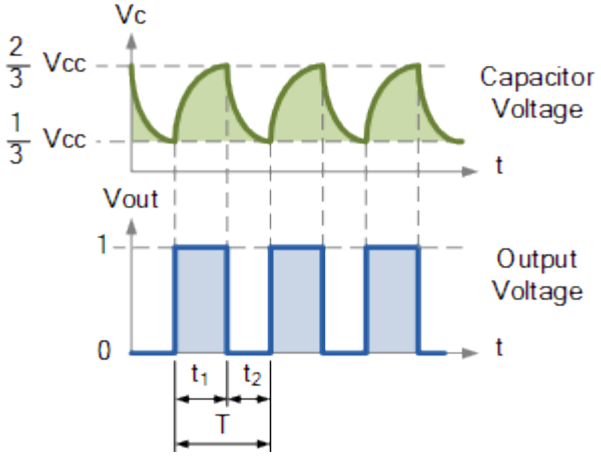

High-Level Language / Applications (∞)
Operating System
Compiler
Virtual Machine
Assembler
Machine Language
Computer Architecture
ALU
Memory Elements
Boolean Arithmetic
Sequential Logic
Boolean Logic
Typical software hierarchy
Typical hardware platform
c1
c2
c3
c4
c5
c6
c7
c8
c9
c10
c11
c12



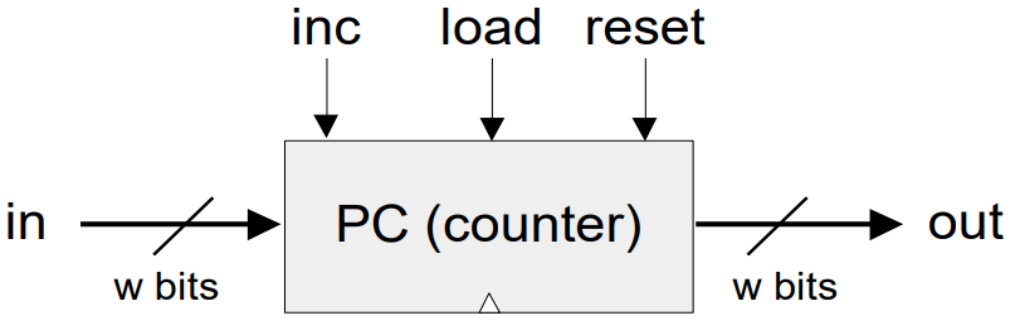
- Secuencia de Lectura
- Secuencia de Escritura

Basic Astable 555 Oscillator Circuit



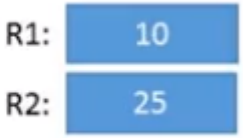


Counters

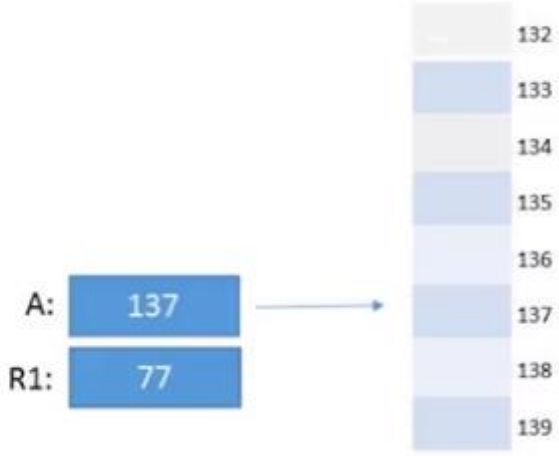


Registros

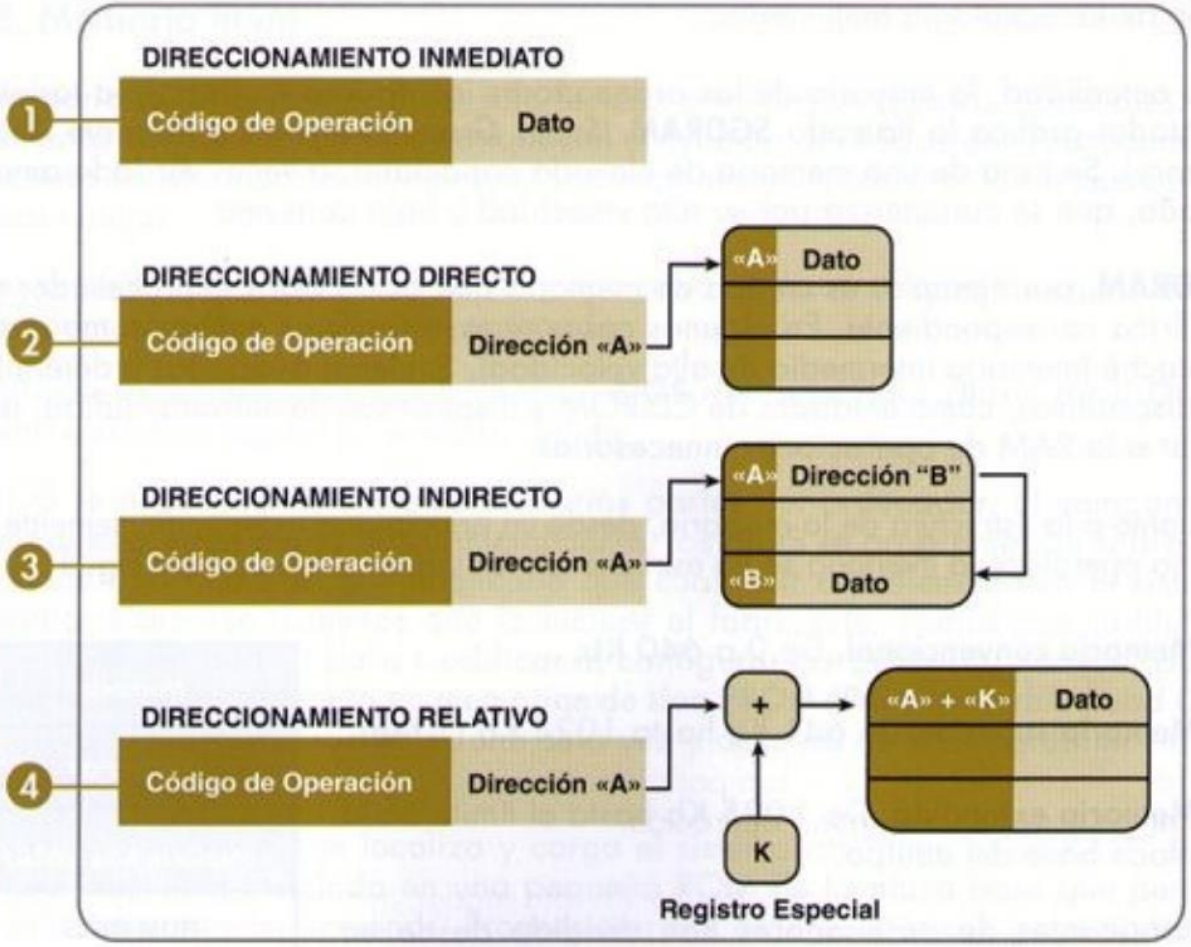
- Data Registers
  - Add R1, R2

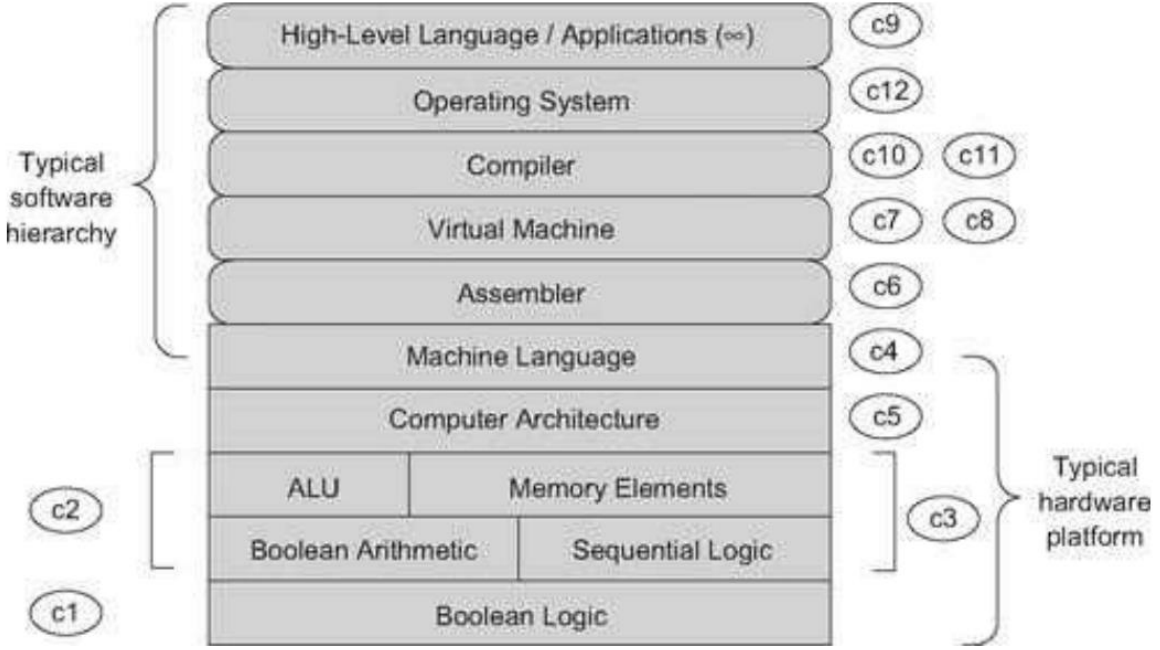


- Address Registers
  - Store R1, @A



- Register
  - Add R1, R2 // R2 ←R2 + R1
- Direct
  - Add R1, M[200] // Mem[200] ← Mem[200] + R1
- Indirect
  - Add R1, @A // Mem[A] ← Mem[A] + R1
- Immediate
  - Add 73, R1 // R1 ← R1 + 73


DIRECCIONAMIENTO INMEDIATO
1
Código de Operación
Dato
DIRECCIONAMIENTO DIRECTO
2
Código de Operación
Dirección «A»
«A»
Dato
DIRECCIONAMIENTO INDIRECTO
3
Código de Operación
Dirección «A»
«A»
Dirección "B"
«B»
Dato
DIRECCIONAMIENTO RELATIVO
4
Código de Operación
Dirección «A»
+
«A» + «K»
Dato
K
Registro Especial


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

Flow Control

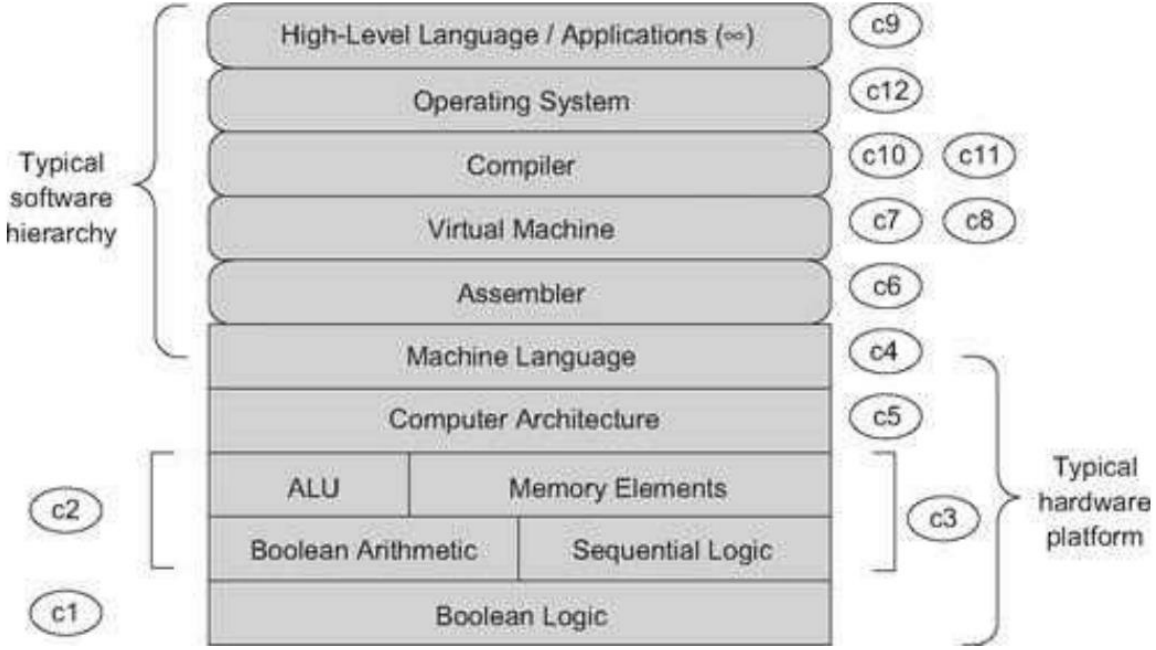
- Se siguen instrucciones en orden (top down)
- Sin embargo es básico que se pueda cambiar este orden.

```
        Load R1, 0
loop:   Add 1, R1
        ...
        // Do something with R1's value
        ...
        Jump loop
```

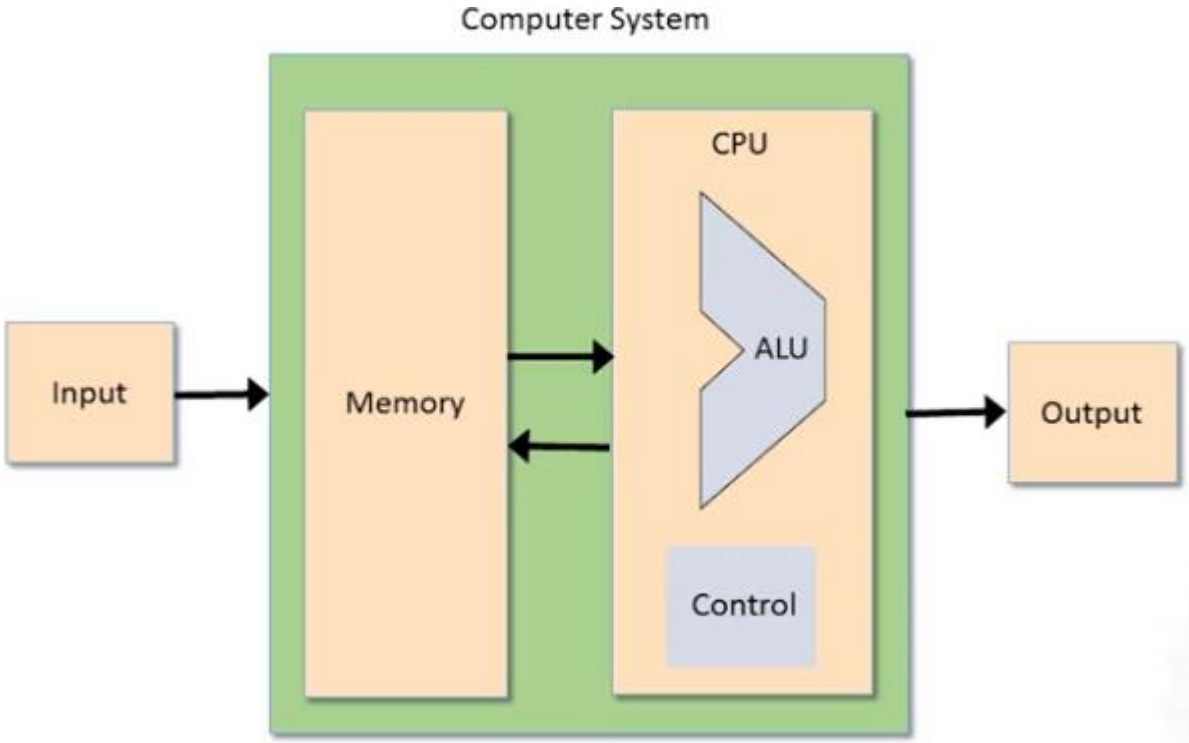
```
// In what follows R1,R2,R3 are registers, PC is program counter,
// and addr is some value.

ADD R1,R2,R3      // R1 ← R2 + R3

ADDI R1,R2,addr   // R1 ← R2 + addr

AND R1,R1,R2      // R1 ← R1 and R2 (bit-wise)

JMP addr          // PC ← addr

JEQ R1,R2,addr    // IF R1 == R2 THEN PC ← addr ELSE PC++

LOAD R1, addr     // R1 ← RAM[addr]

STORE R1, addr    // RAM[addr] ← R1

NOP               // Do nothing
```

Operaciones Lógicas/Aritméticas

```
ADD R1,R2,R3      // R1 ← R2 + R3

ADDI R1,R2,addr   // R1 ← R2 + addr

AND R1,R1,R2      // R1 ← R1 and R2 (bit-wise)
```


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:



THANKS!

Any questions?