Sequence Logic and ALU

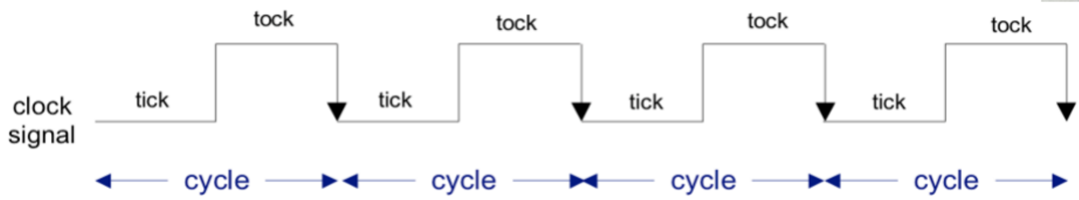
Memory

Implementation of memory elements involves synchronization, clocking and feedback loops

Clock

The hardware implementation is based on an **oscillator** that alternates between the beginning phases labelled:

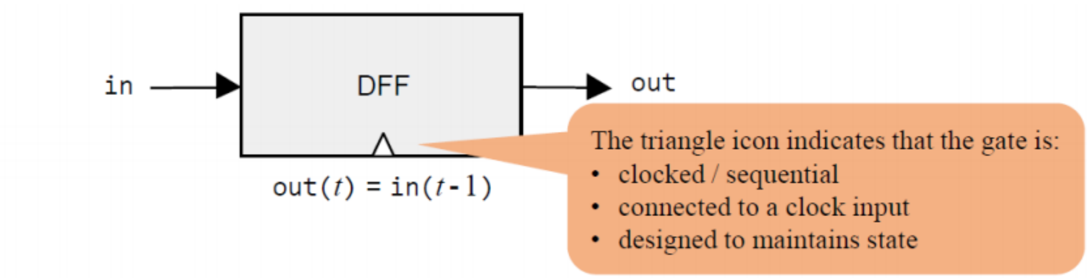
The elapsed time between the beginning of a tick and the end of a subsequent tock is called a **cycle**

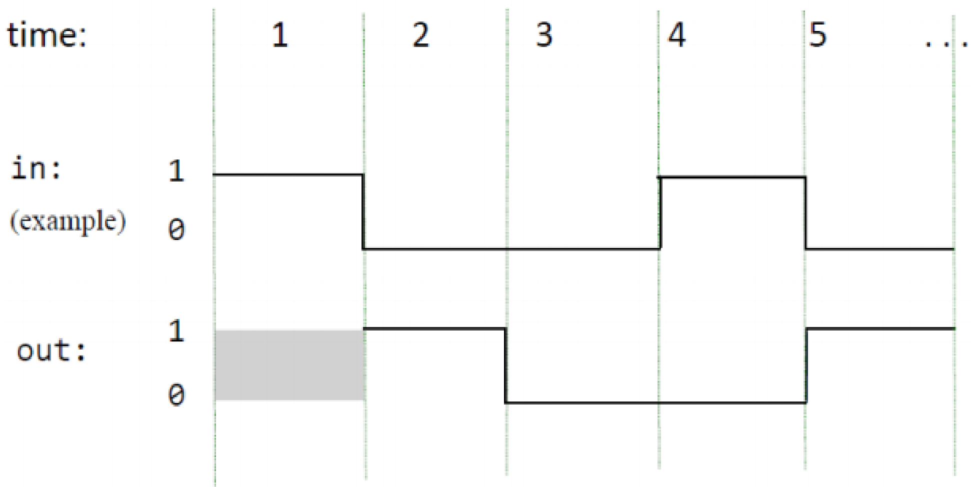


Flip Flops

Data Flip Flop (DFF): the simplest **state keeping** gate (built-in)

The gate outputs its previous input: out(*t*)= in(*t*-1)





Data and Time in DFF

out(t)= in(t-1)

• **in** is the gate's input value

• **out** is the gate's output value

• **t** is the current clock cycle

• **t-1** is the previous clock cycle

• **t+1** is the next clock cycle

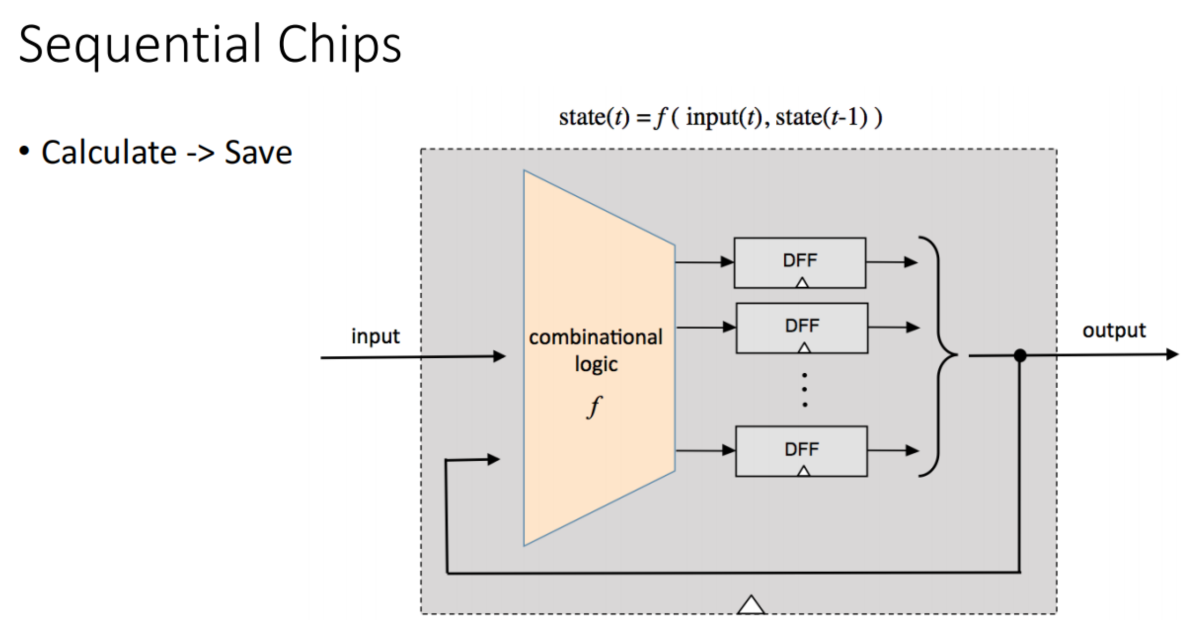
> This elementary behavior can form the basis of all the hardware devices that computers use to maintain state

Sequential Chips

Sequential chips are capable of:

- Maintaining state
- Acting on the state, and on the current inputs

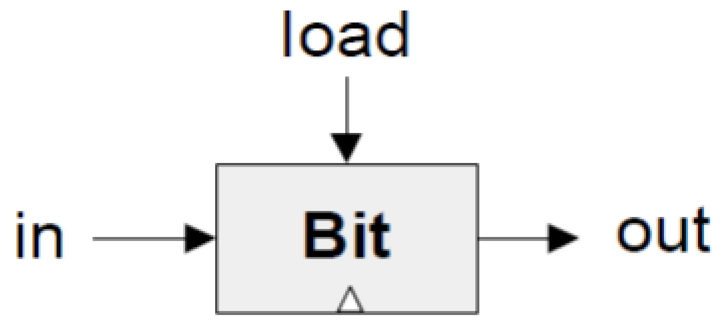
$$state(t) = f(state(t-1), input(t))$$



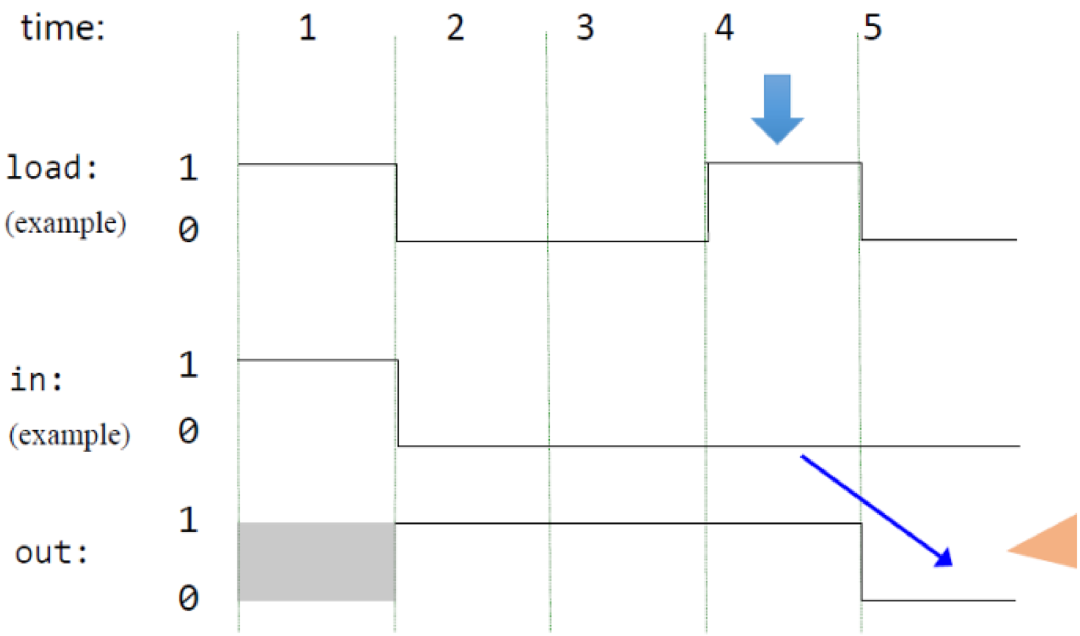
Register

A register is a storage device that can "**store**" or "**remember**" a value over time

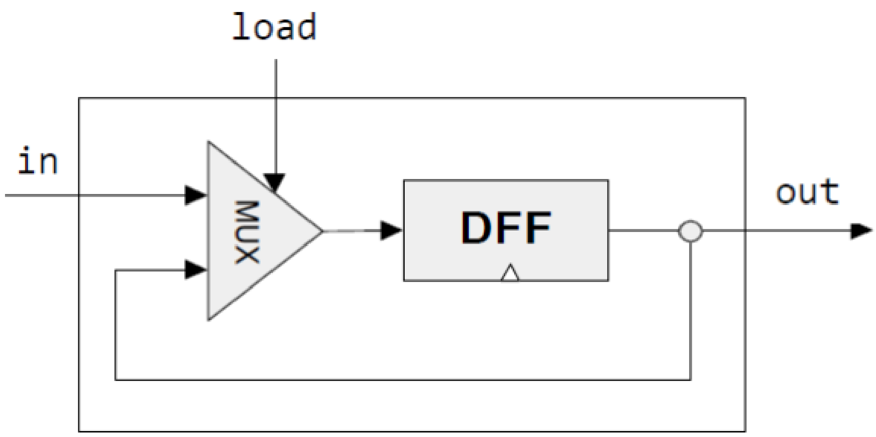
1-bit register



$$\text{if } \mathtt{load}(t) \text{ then } \mathtt{out}(t+1) = \mathtt{in}(t)$$
$$\text{else } \quad \mathtt{out}(t+1) = \mathtt{out}(t)$$

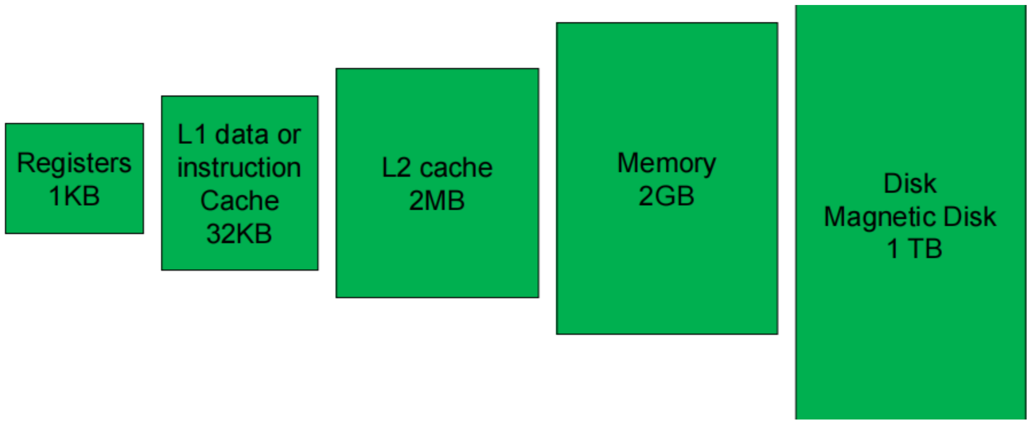


1-bit Register: Implementation



Memory Hierarchy

As it goes further, capacity and latency increase



Random-Access Memory (RAM)

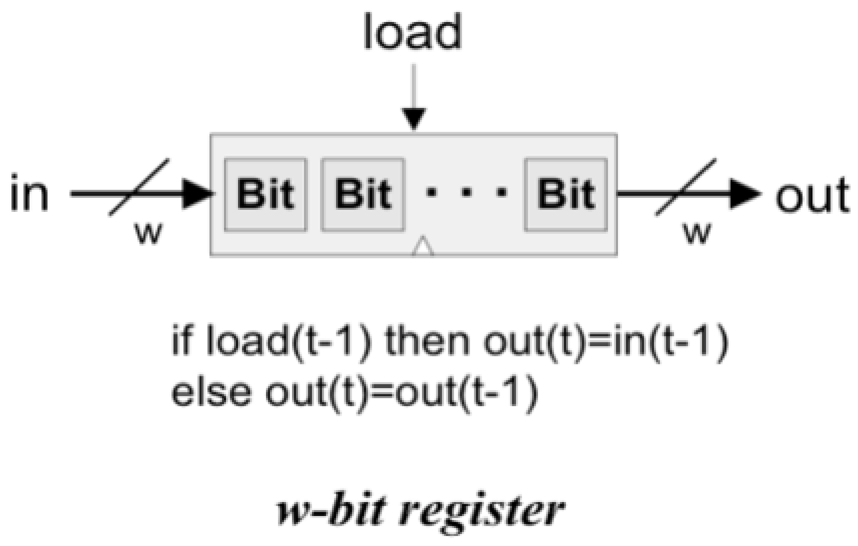
RAM should be able to access randomly chosen words, with no restriction in the order in which they are accessed

only maintains its data while the device is powered

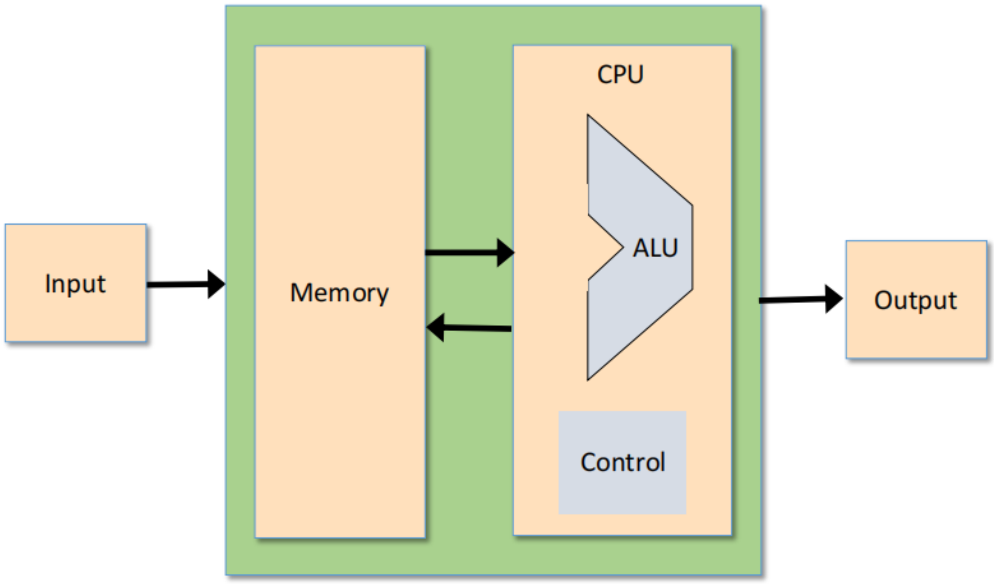
Random-Access Memory has:

- A data input
- An address input
- A load bit (read is load=0, write is load=1)

Multi-bit Registers



The IAS (von Neumann) Machine



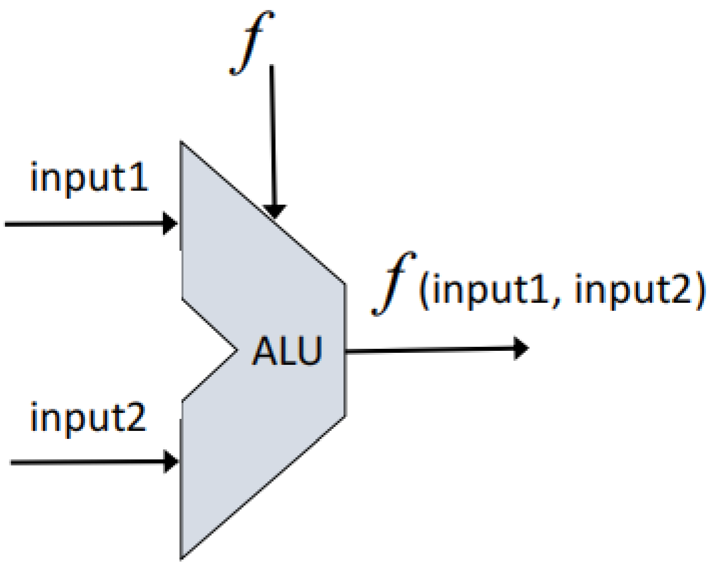
Computer System

Arithmetic Logical Unit

• **A combinational circuit** that performs arithmetic and bitwise operations on integers represented as binary numbers.

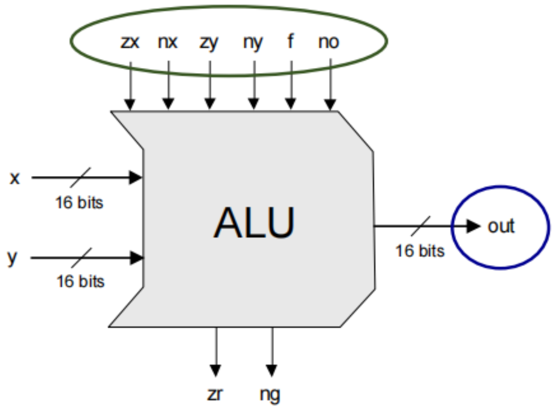
• Input the **data** and some **code for the operation**

• Output will be some **data** and any **additional information**



The Hack ALU

To cause the ALU to compute a function, set the control bits to one of the binary combinations listed in the table.



control bits

zx	nx	zy	ny	f	no	out
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