[Von Neumann] Computers

- CPU
- memory (disk drives, DRAM, SRAM, CD)
- input (mouse, keyboard)
- output (display, printer)
- network
- software

We need abstraction ...

Levels of abstraction

Software:

- Application
- Operating system
- Firmware

Instruction set architecture:

- Data type and structures: encodings and machine representation
- Instruction set
- Instruction formats
- Addressing modes and accessing data and instructions

Hardware:

- Instruction set processing
- I/O System
- Digital design
- Circuit design
- Layout

What is “Computer Architecture”

Computer Architecture =

Instruction Set Architecture +

Machine Organization

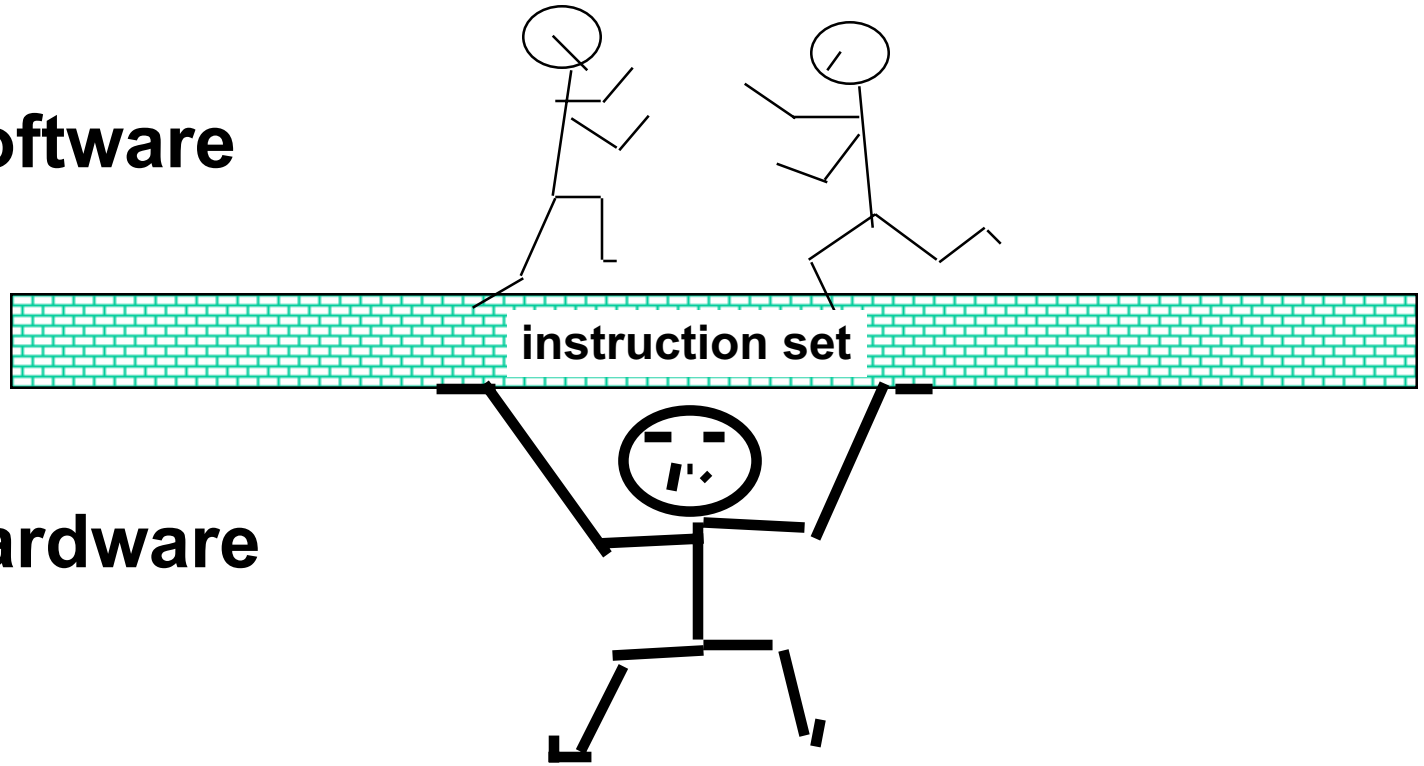
Instruction Set Architecture

- Organization of Programmable Storage
- Data type and Structures: encodings and machine representation
- Instruction set
- Instruction Formats
- Addressing Modes and Accessing Data and Instructions
- Exception Handling

Computer Organization

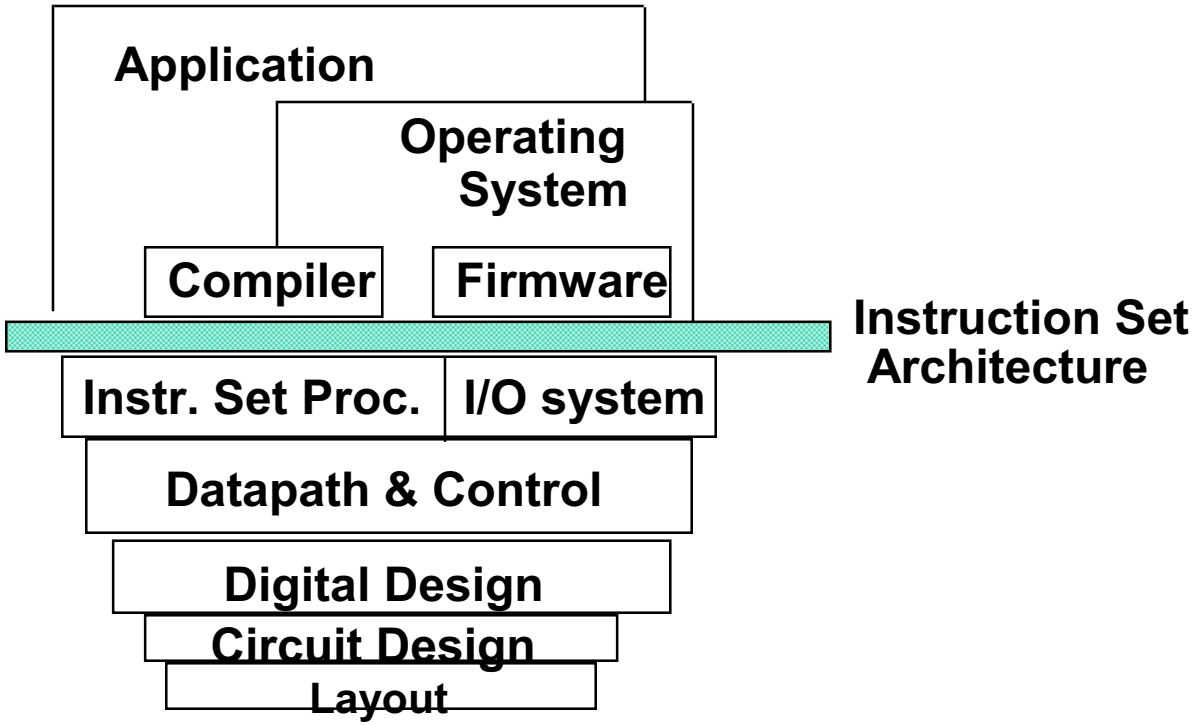
- Capabilities and Performance of the Basic Functional Units
- The Way These Units are Interconnected
- Information Flow between components
- Information Flow Control

software

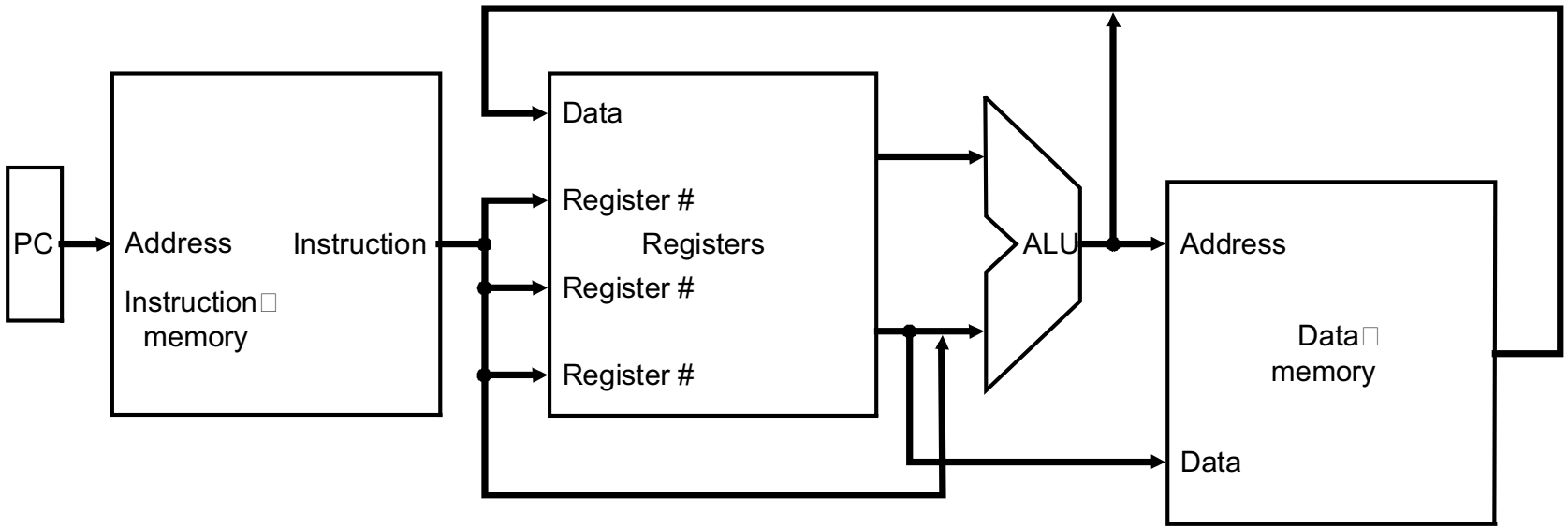


hardware

What is “Computer Architecture”?



MIPS machine



Example: adding two variables

Software level

C:

- A = B + C

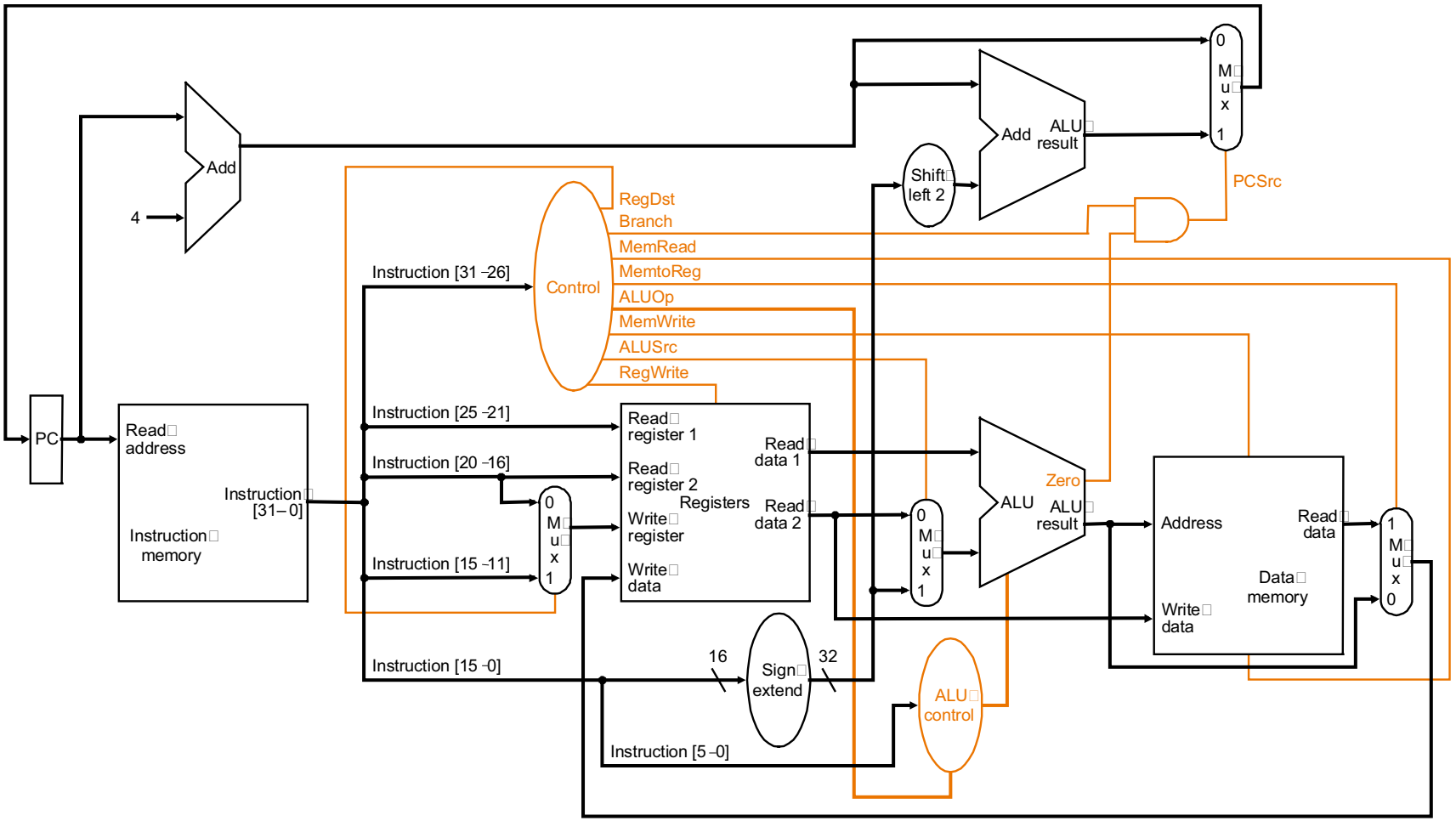
Assembler

- B -> $s1, C -> $s2
- add $t0, $s1, $s2
- $t0 -> A

Machine instruction:

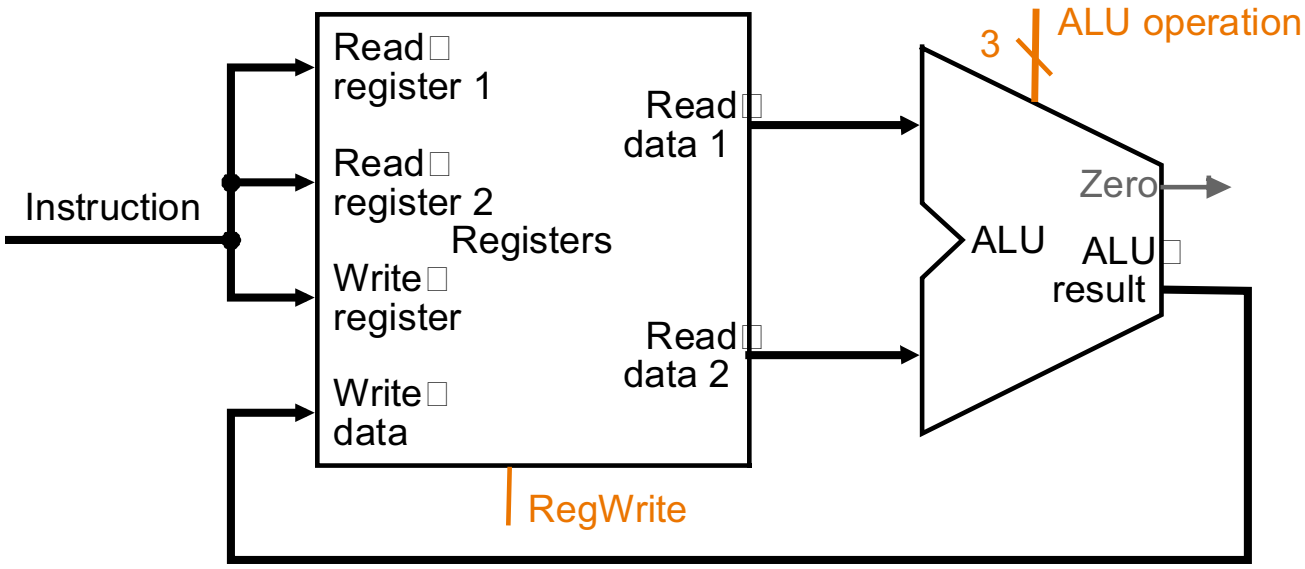
	op	rs	rt	rd	…	funct
• decimal:	0	17	18	8	0	32
• binary:	000000	10001	10010	01000	00000	100000

Example: Datapath and control



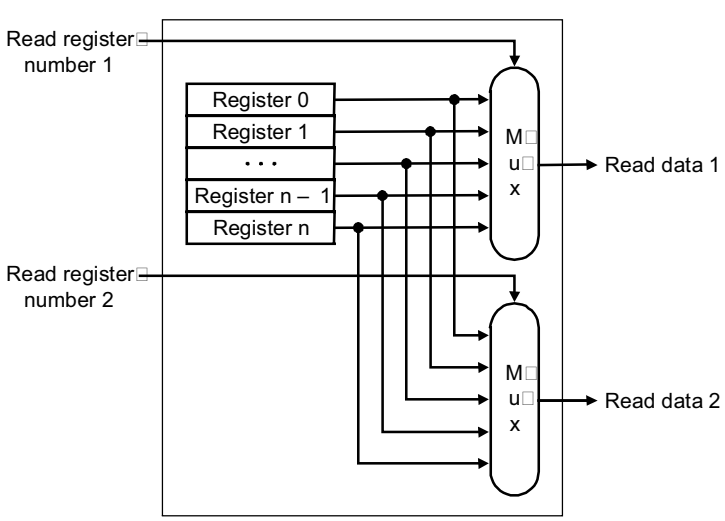
Example: from instructions to gates

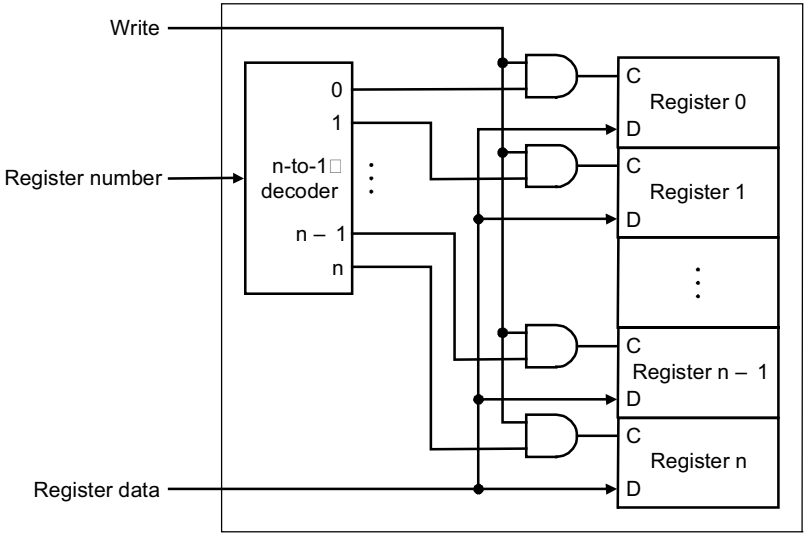
Register file and ALU



Example: from instructions to gates

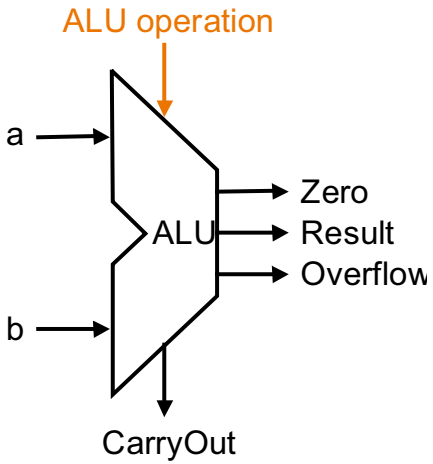
Inside register file

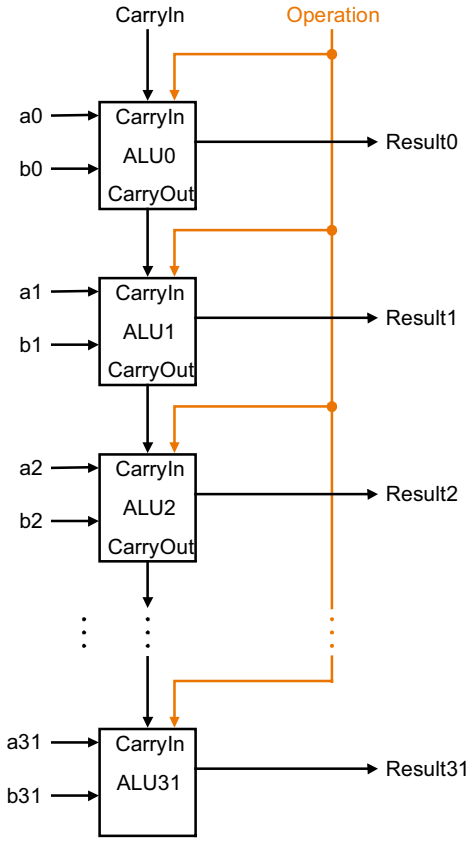


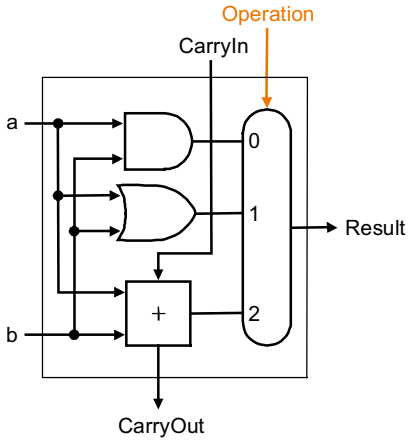


Example: from instructions to gates

Arithmetic Logic Unit (ALU)







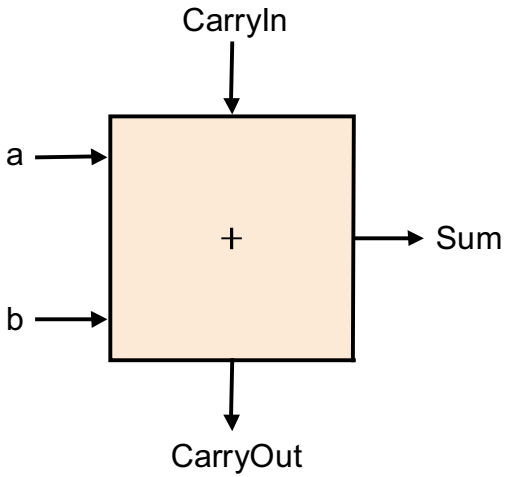
Example: from instructions to gates

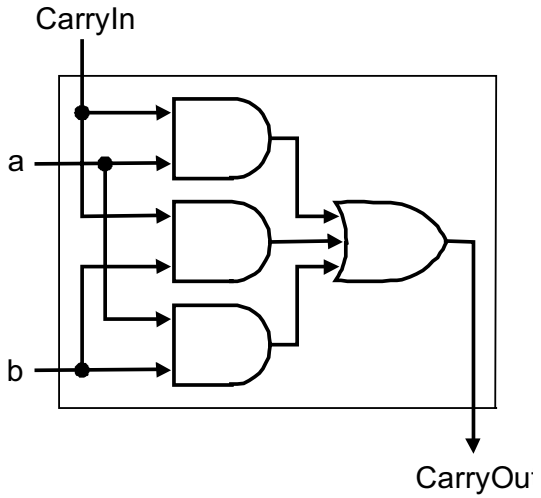
ALU: Carry Out logic

$$CarryOut = b.CarryIn + a.CarryIn + a.b + a.b.CarryIn$$

or

$$CarryOut = b.CarryIn + a.CarryIn + a.b$$

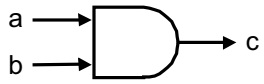




Example: from instructions to gates

Logic gates

1. AND gate (c = a . b)



a	b	c = a . b
0	0	0
0	1	0
1	0	0
1	1	1

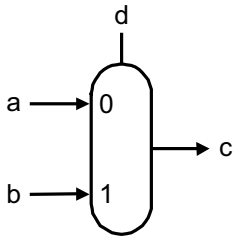
2. OR gate (c = a + b)

a	b	c = a + b
0	0	0
0	1	1
1	0	1
1	1	1

3. Inverter (c = $\bar{a}$)

a	c = $\bar{a}$
0	1
1	0

4. Multiplexor
   (if d = = 0, c = a;
   else c = b)



d	c
0	a
1	b