

## how computers work (3)

inside the ALU

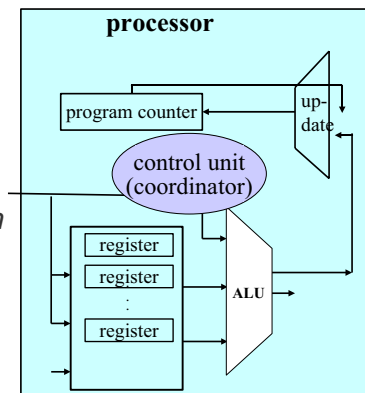
## goals

- last time, we saw the fetch/execution cycle
- focus was on
  - flow of instructions and data between memory and the computer processor
  - the units within the processor and their roles
- today: see how one component, the ALU (arithmetic and logic unit) works

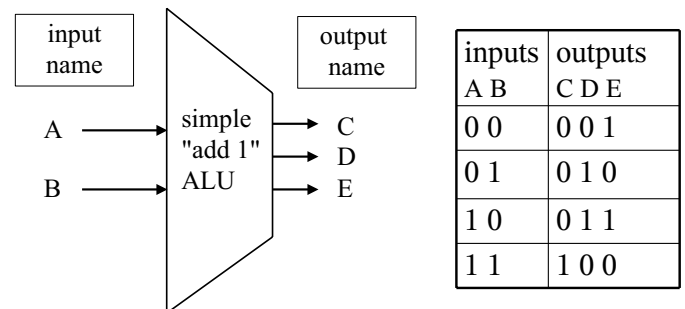
## the arithmetic and logic unit

recall: the ALU

- takes as *input* some *data* (bits) from some *registers*
- performs an *operation* on the data, such as addition, subtraction, multiplication, division
- produces the *output* of the operation



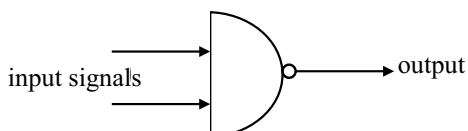
## our task: design a unit that adds 1 to a 2-bit input



**Note:** the function of our "add-1" ALU can be specified as an input/output table

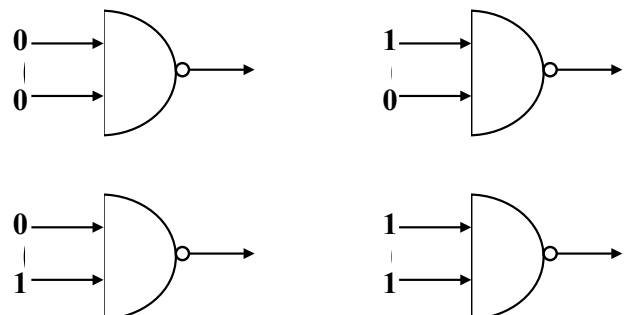
## basic building block for the ALU

- the ALU (as well as the control unit and the unit that updates the program counter) are built from components called *gates*
- a versatile gate is the *NAND gate*, which takes two input signals (bits) and produces a single output bit:



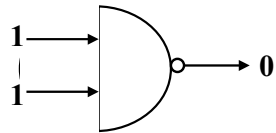
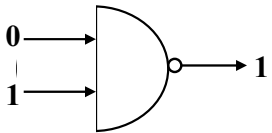
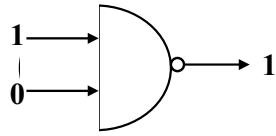
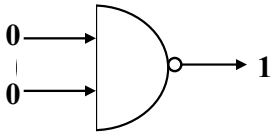
## NAND gate

there are four possible input values:



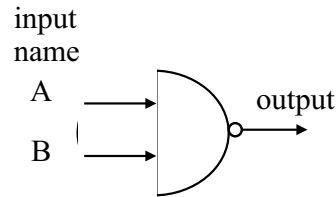
## NAND gate

for these input values, the output value is:



## NAND gate

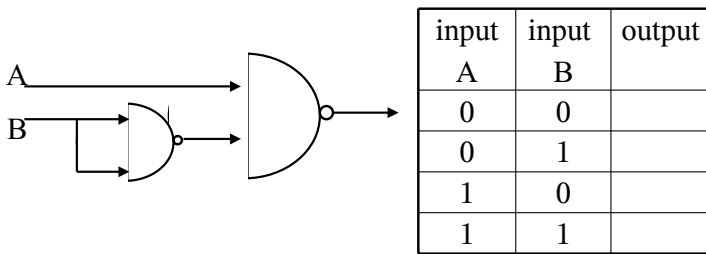
the NAND gate's function can be summarised in a table:



| input A | input B | output |
|---------|---------|--------|
| 0       | 0       | 1      |
| 0       | 1       | 1      |
| 1       | 0       | 1      |
| 1       | 1       | 0      |

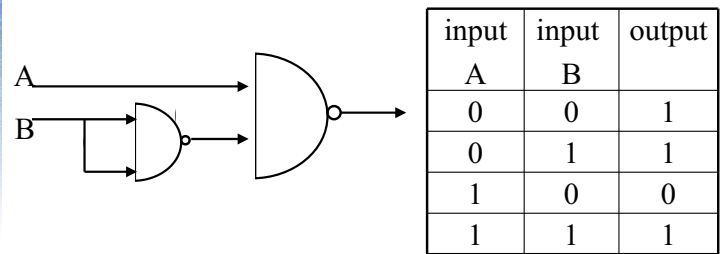
## circuits

- just as brains are built from brain cells (called neurons), *circuits* are built from gates
- here is a circuit with two inputs and two gates; what is its input output table?

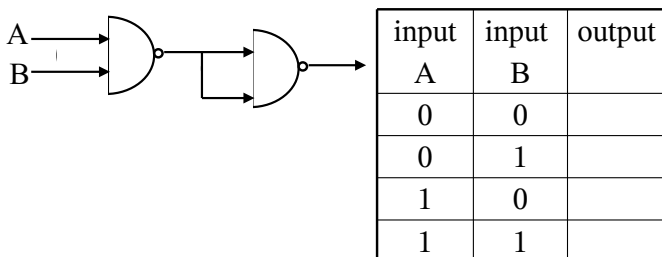
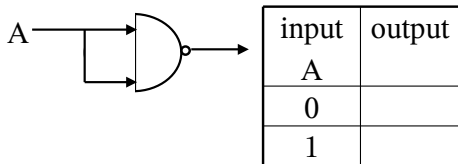


## circuits

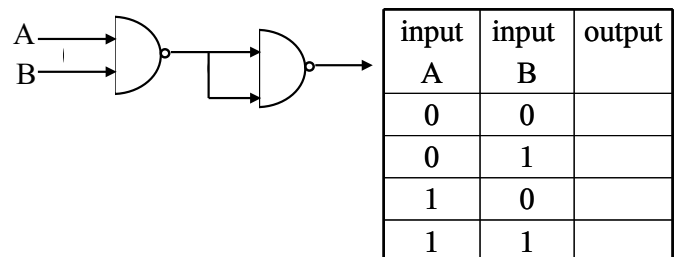
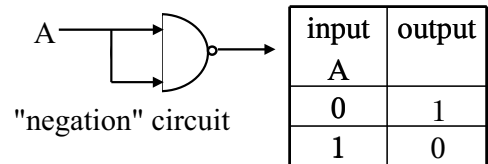
- just as brains are built from brain cells (called neurons), *circuits* are built from gates
- here is a circuit with two inputs and two gates; what is its input output table?



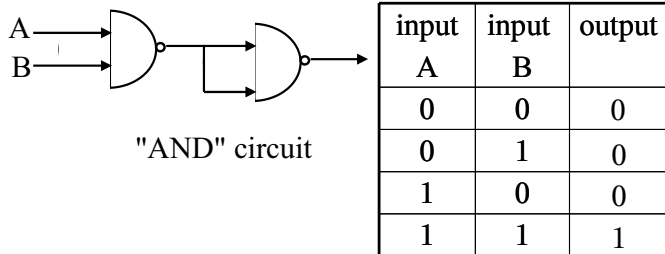
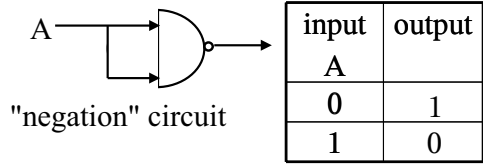
## more circuits



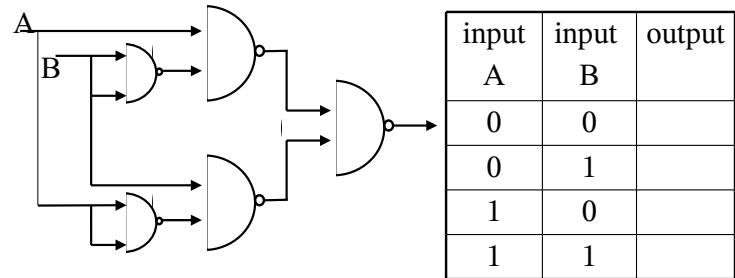
## more circuits



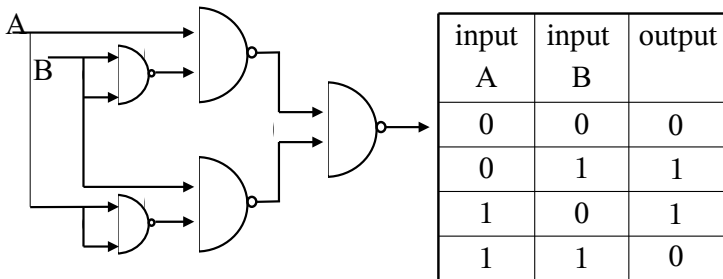
### more circuits



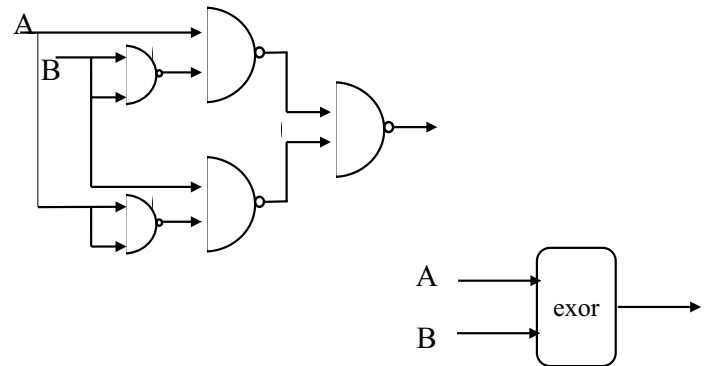
### the EXOR circuit: what is its table?



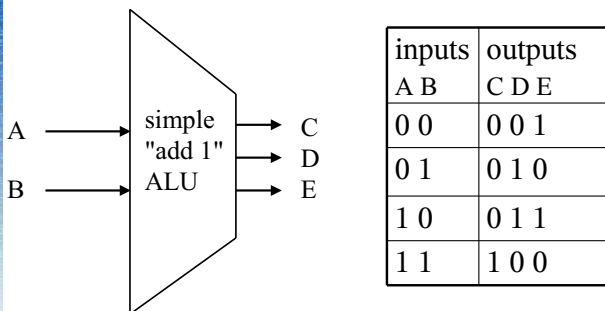
### the EXOR circuit: what is its table?



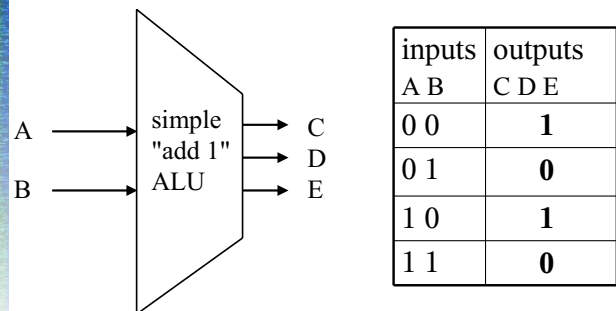
### an abstraction of the EXOR circuit:



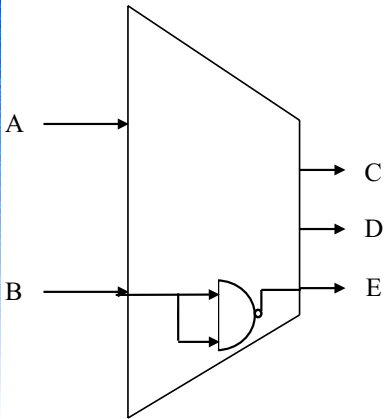
### back to our task: first focus on output C



### back to our task: first focus on output E

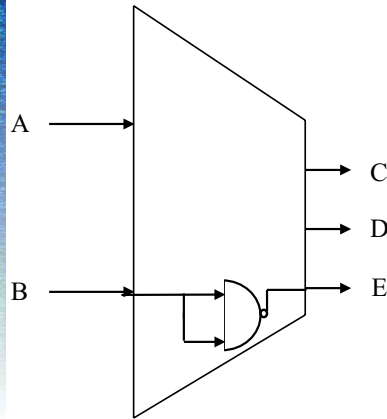


output E is the negation of input B!



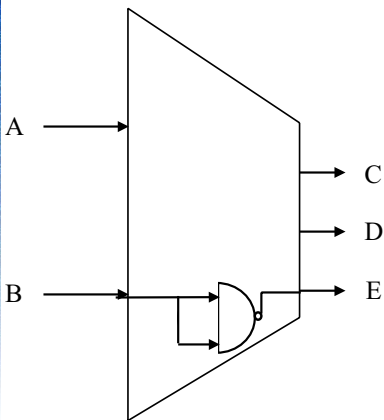
| inputs |   | outputs |   |   |
|--------|---|---------|---|---|
| A      | B | C       | D | E |
| 0      | 0 | 0       | 0 | 1 |
| 0      | 1 | 0       | 1 | 0 |
| 1      | 0 | 0       | 1 | 1 |
| 1      | 1 | 1       | 0 | 0 |

next look at output D



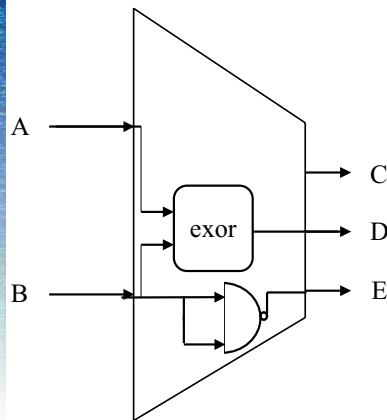
| inputs |   | outputs |   |   |
|--------|---|---------|---|---|
| A      | B | C       | D | E |
| 0      | 0 | 0       | 0 | 1 |
| 0      | 1 | 0       | 1 | 0 |
| 1      | 0 | 0       | 1 | 1 |
| 1      | 1 | 1       | 0 | 0 |

next look at output D



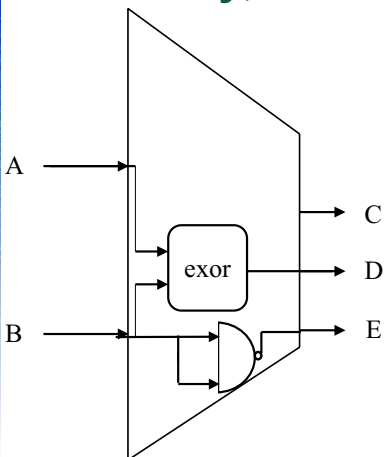
| inputs |   | outputs |   |   |
|--------|---|---------|---|---|
| A      | B | C       | D | E |
| 0      | 0 | 0       | 0 | 1 |
| 0      | 1 | 0       | 1 | 0 |
| 1      | 0 | 0       | 1 | 1 |
| 1      | 1 | 1       | 0 | 0 |

output D is the EXOR of inputs A and B



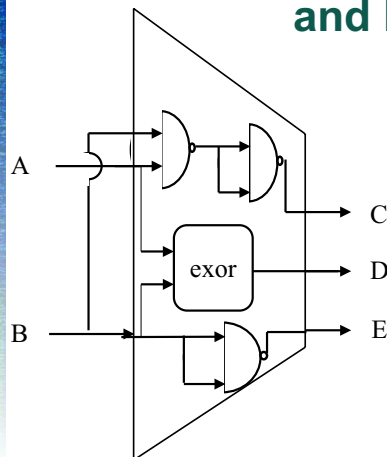
| inputs |   | outputs |   |   |
|--------|---|---------|---|---|
| A      | B | C       | D | E |
| 0      | 0 | 0       | 0 | 1 |
| 0      | 1 | 0       | 1 | 0 |
| 1      | 0 | 0       | 1 | 1 |
| 1      | 1 | 1       | 0 | 0 |

finally, look at output C



| inputs |   | outputs |   |   |
|--------|---|---------|---|---|
| A      | B | C       | D | E |
| 0      | 0 | 0       | 0 | 1 |
| 0      | 1 | 0       | 1 | 0 |
| 1      | 0 | 0       | 1 | 1 |
| 1      | 1 | 1       | 0 | 0 |

output C is the AND of inputs A and B



| inputs |   | outputs |   |   |
|--------|---|---------|---|---|
| A      | B | C       | D | E |
| 0      | 0 | 0       | 0 | 1 |
| 0      | 1 | 0       | 1 | 0 |
| 1      | 0 | 0       | 1 | 1 |
| 1      | 1 | 1       | 0 | 0 |

## summary

- a circuit that adds 1 to a binary number can be built from NAND gates
- circuits that add or multiply two binary numbers are based on exactly the same principles
- you can buy NAND gates online! 43 cents for 4 NAND gates on a chip at [www.semiconductors.philips.com](http://www.semiconductors.philips.com)