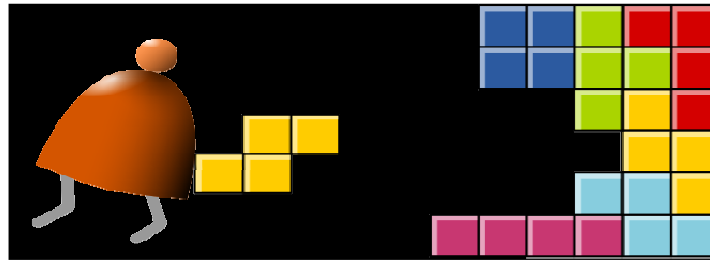


# Boolean Arithmetic



*Building a Modern Computer From First Principles*

[www.nand2tetris.org](http://www.nand2tetris.org)

# Counting systems

---

quantity	decimal	binary	3-bit register
	0	0	000
*	1	1	001
**	2	10	010
***	3	11	011
****	4	100	100
*****	5	101	101
*****	6	110	110
*****	7	111	111
*****	8	1000	overflow
*****	9	1001	overflow
*****	10	1010	overflow

# Rationale

---

$$(9038)_{ten} = 9 \cdot 10^3 + 0 \cdot 10^2 + 3 \cdot 10^1 + 8 \cdot 10^0 = 9038$$

$$(10011)_{two} = 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 19$$

$$(x_n x_{n-1} \dots x_0)_b = \sum_{i=0}^n x_i \cdot b^i$$

# Binary addition

---

Assuming a 4-bit system:

$$\begin{array}{r} \underline{0\ 0\ 0\ 1} \\ 1\ 0\ 0\ 1 \\ 0\ 1\ 0\ 1 \\ \hline \end{array} +$$

0 1 1 1 0

no overflow

$$\begin{array}{r} \underline{1\ 1\ 1\ 1} \\ 1\ 0\ 1\ 1 \\ 0\ 1\ 1\ 1 \\ \hline \end{array} +$$

1 0 0 1 0

overflow

- Algorithm: exactly the same as in decimal addition
- Overflow (MSB carry) has to be dealt with.



# Building an Adder chip

---

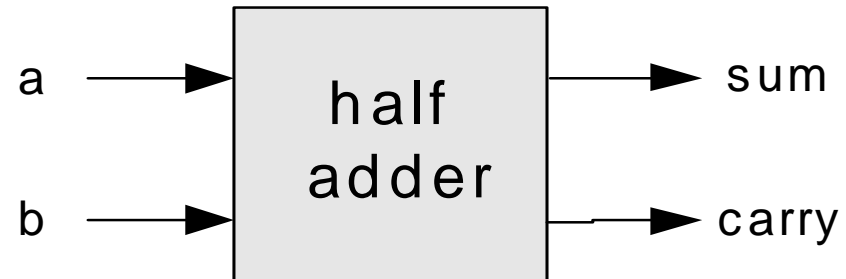


- Adder: a chip designed to add two integers
- Proposed implementation:
  - **Half adder**: designed to add 2 bits
  - **Full adder**: designed to add 3 bits
  - **Adder**: designed to add two  $n$ -bit numbers.

## Half adder (designed to add 2 bits)

---

a	b	sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

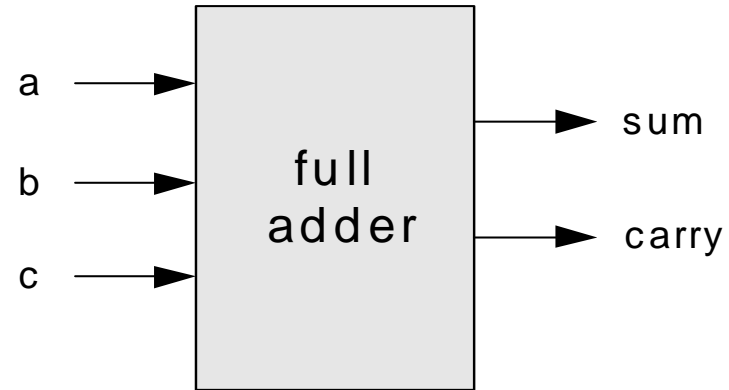


Implementation: based on two gates that you've seen before.

# Full adder (designed to add 3 bits)

---

a	b	c	sum	carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



Implementation: can be based on half-adder gates.



## $n$ -bit Adder (designed to add two 16-bit numbers)

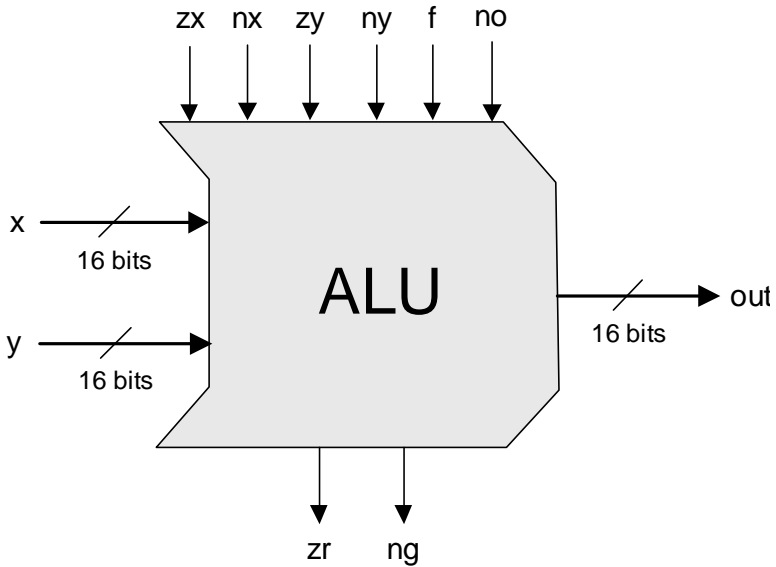
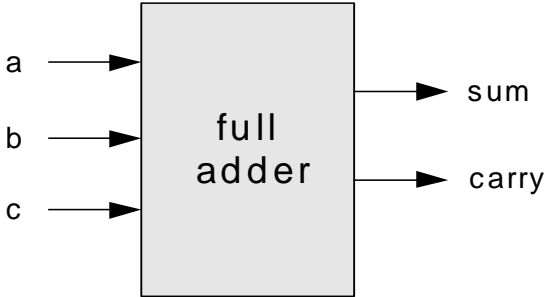
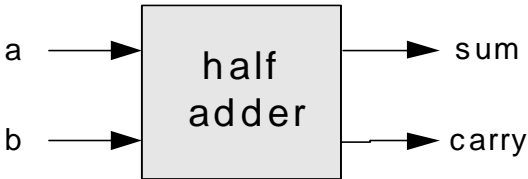
---



...	1	0	1	1	a
					+
...	0	0	1	0	b
<hr/>					
...	1	1	0	1	out

Implementation: array of full-adder gates.

# The ALU (of the Hack platform)



$out(x, y, \text{control bits}) =$

- $x+y, x-y, y-x,$
- $0, 1, -1,$
- $x, y, -x, -y,$
- $x!, y!,$
- $x+1, y+1, x-1, y-1,$
- $x\&y, x|y$

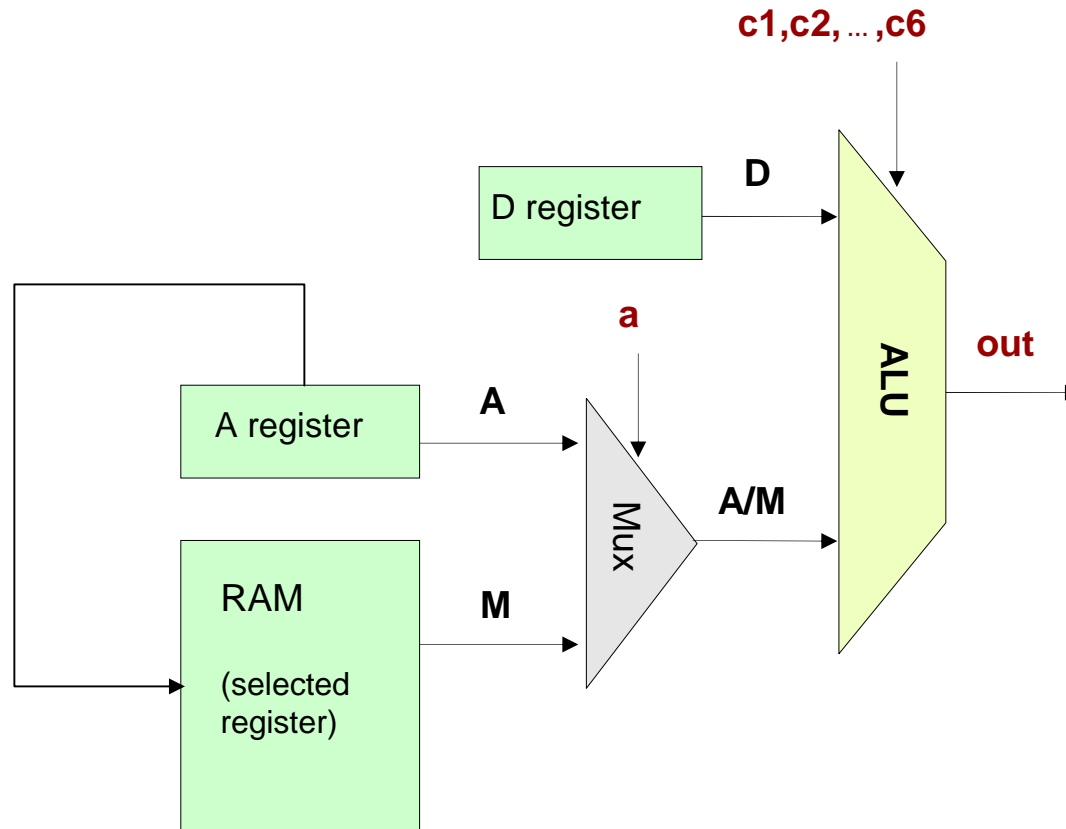
# ALU logic (Hack platform)

These bits instruct how to pre-set the x input		These bits instruct how to pre-set the y input		This bit selects between + / And	This bit inst. how to post-set out	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 And y	if no then out=!out	f (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						y
0						!x
1						!y
0						-x
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

Implementation: build a logic gate architecture that "executes" the control bit "instructions":  
if zx==1 then set x to 0 (bit-wise), etc.

# The ALU in the CPU context (a sneak preview of the Hack platform)

---



# Perspective

---

- Combinational logic
- Our adder design is very basic: no parallelism
- It pays to optimize adders
- Our ALU is also very basic: no multiplication, no division
- Where is the seat of more advanced math operations?  
a typical hardware/software tradeoff.

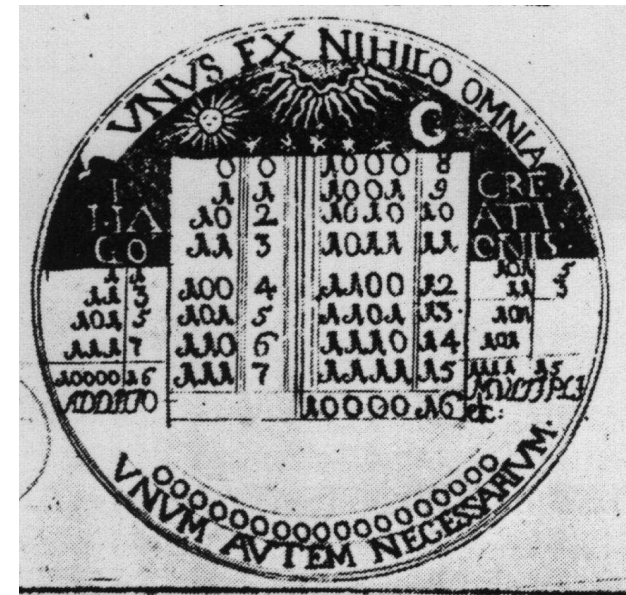
## Historical end-note: Leibnitz (1646-1716)



- “The binary system may be used in place of the decimal system; express all numbers by unity and by nothing”
- 1679: built a mechanical calculator (+, -, \*, /)



- CHALLENGE: “All who are occupied with the reading or writing of scientific literature have assuredly very often felt the want of a common scientific language, and regretted the great loss of time and trouble caused by the multiplicity of languages employed in scientific literature:
- SOLUTION: “*Characteristica Universalis*”: a universal, formal, and decidable language of reasoning
- The dream's end: Turing and Gödel in 1930's.



Leibniz's medallion  
for the Duke of Brunswick