

Boolean Arithmetic

Usage and Copyright Notice:

Copyright 2005 © Noam Nisan and Shimon Schocken

This presentation contains lecture materials that accompany the textbook “The Elements of Computing Systems” by Noam Nisan & Shimon Schocken, MIT Press, 2005.

We provide both PPT and PDF versions.

The book web site, www.idc.ac.il/tecs , features 13 such presentations, one for each book chapter. Each presentation is designed to support about 3 hours of classroom or self-study instruction.

You are welcome to use or edit this presentation as you see fit for instructional and non-commercial purposes.

If you use our materials, we will appreciate it if you will include in them a reference to the book’s web site.

If you have any questions or comments, you can reach us at tecs.ta@gmail.com

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:

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

no overflow

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

overflow

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

Representing negative numbers (4-bit system)

0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
	1000 -8
	1111 -1
	1110 -2
	1101 -3
	1100 -4
	1011 -5
	1010 -6
	1001 -7

- The codes of all positive numbers begin with a "0"
- The codes of all negative numbers begin with a "1"
- To convert a number:
leave all trailing 0's and first 1 intact,
and flip all the remaining bits

Example: $2 - 5 = 2 + (-5) = 0010$

$$\begin{array}{r} + 1011 \\ \hline 1101 = -3 \end{array}$$

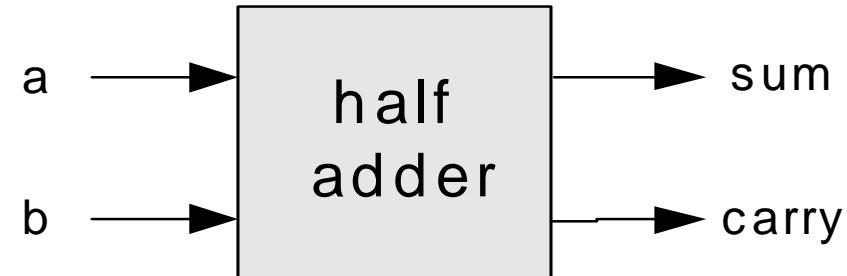
Building an Adder chip



- 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)

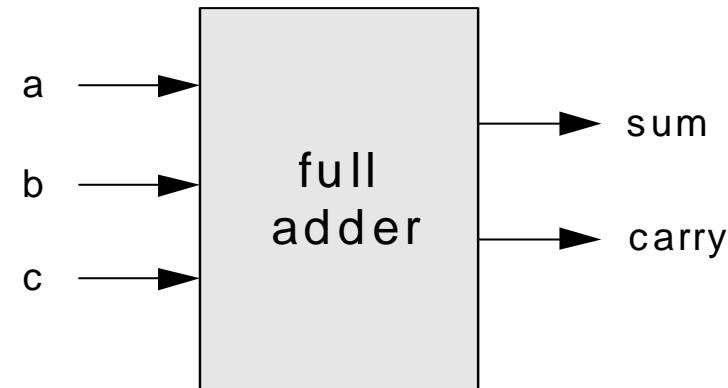
a	b	carry	sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



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

Full adder (designed to add 3 bits)

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



- Implementation: can be based on half-adder gates.

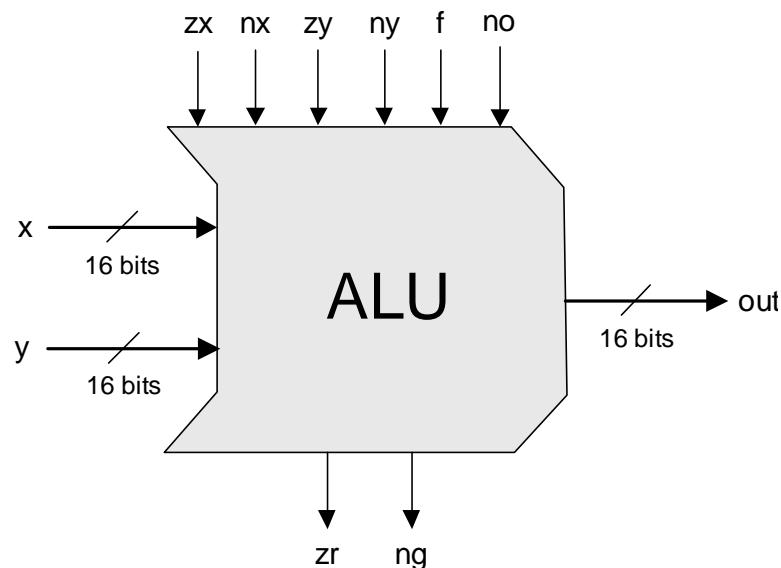
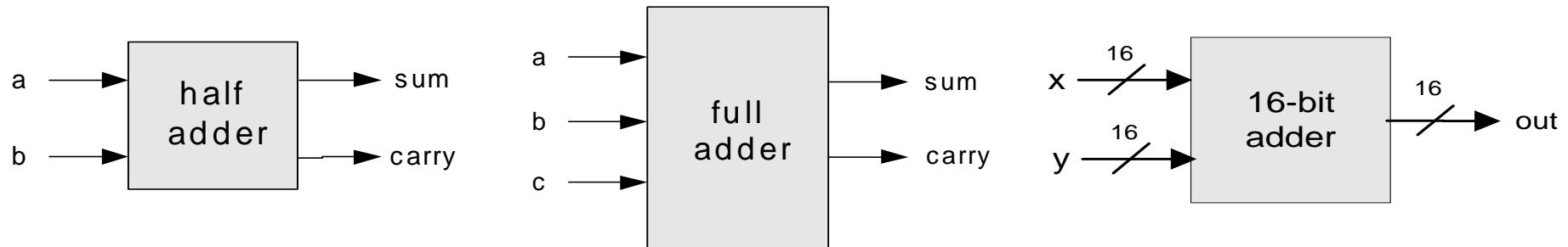
n-bit Adder



$$\begin{array}{r} \dots 1 0 1 1 a \\ + \\ \dots 0 0 1 0 b \\ \hline \dots 1 1 0 1 \text{out} \end{array}$$

- Implementation: array of full-adder gates.

The ALU (of the Hack platform)



```
out(x, y, 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
```

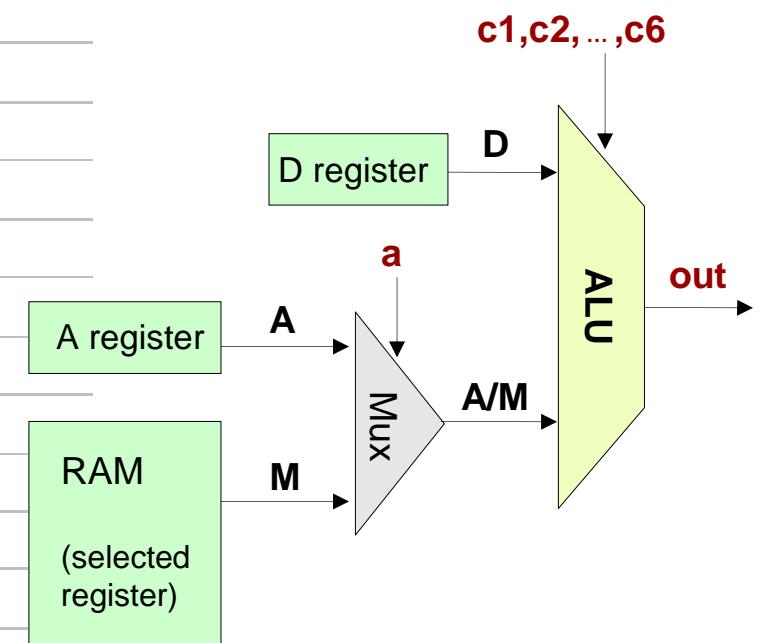
ALU logic (Hack platform)

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 "reads" each control bit and does what the table specifies: if zx=1 then set x to 0, etc.

The ALU in the CPU context (Hack platform)

out (when $a=0$)	c1	c2	c3	c4	c5	c6	out (when $a=1$)
0	1	0	1	0	1	0	
1	1	1	1	1	1	1	
-1	1	1	1	0	1	0	
D	0	0	1	1	0	0	
A	1	1	0	0	0	0	M
!D	0	0	1	1	0	1	
!A	1	1	0	0	0	1	!M
-D	0	0	1	1	1	1	
-A	1	1	0	0	1	1	-M
D+1	0	1	1	1	1	1	
A+1	1	1	0	1	1	1	M+1
D-1	0	0	1	1	1	0	
A-1	1	1	0	0	1	0	M-1
D+A	0	0	0	0	1	0	D+M
D-A	0	1	0	0	1	1	D-M
A-D	0	0	0	1	1	1	M-D
D&A	0	0	0	0	0	0	D&M
D A	0	1	0	1	0	1	D M

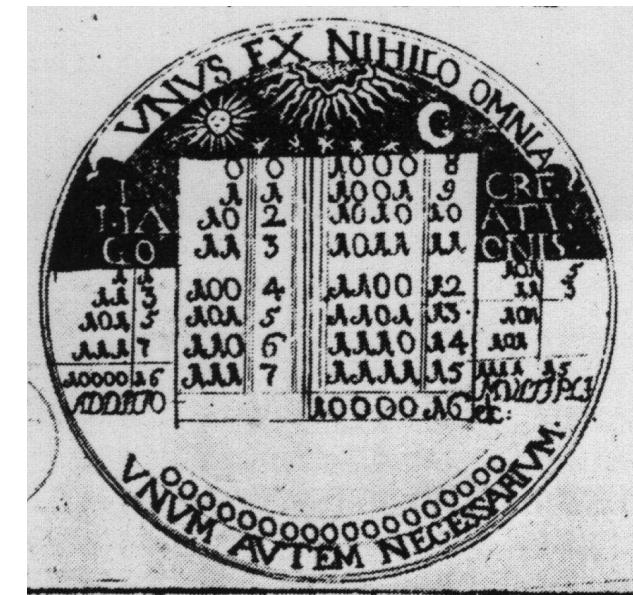


Perspective

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

Historical 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 (+, -, *, /)



Leibniz's medallion
for the Duke of Brunswick

- 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, language of reasoning
- The dream's end: Turing and Goedl in 1930's.