Bits, Data Types, Arithmetic Operations
Chapter 2 Sept. 5, 2018

- Review: Algorithms
- Using building blocks
- Binary system
- Unsigned integers
  - Unsigned addition
- Negative numbers
  - Two's complement representation
- Conversion between binary and decimal
- More on addition
  - Subtraction
  - Sign extension
  - Overflow
- Review of Hexadecimal representation

Algorithms

Definition:

Process or set of rules to be followed in calculations or other problem-solving operations, especially by a computer.

STATED IN A FORMAL LANGUAGE

EX: NATURAL LANGUAGE: "TIME FLIES"

Example of Algorithm

Problem Statement:

Find the factorial of a given integer, n

n! = 1 x 2 x 3 x .... x n-1 x n

= n x n-1 x n-2 x ..... 2 x 1

n - RESULT - (F) > VARIABLES

Algorithm:

IF n=0, F=1

STOP

OTHERWISE

F = n

n = n-1

IF n=0, STOP

OTHERWISE F=FxN

RESULT = F

EXAMPLE

n = 4

F=4
n=3

F=12
n=2

F=24
n=1

F=2
n=0

STOP

Computer to execute algorithm

Has memory to store n, factorial

Units which can subtract, multiply

Unit to test whether a number is 0, OR NEGATIVE, =, ...

DATA PATH

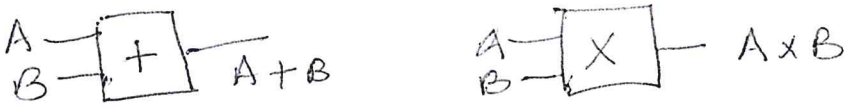
Has Instructions to:

- Get data from memory
- Put data into memory
- Add, subtract, ...
- Test whether a number is zero
- Change flow of execution
- Print answer

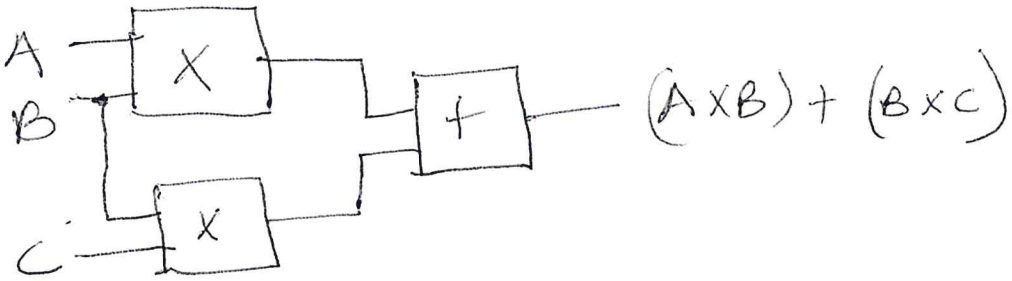
CONTROL

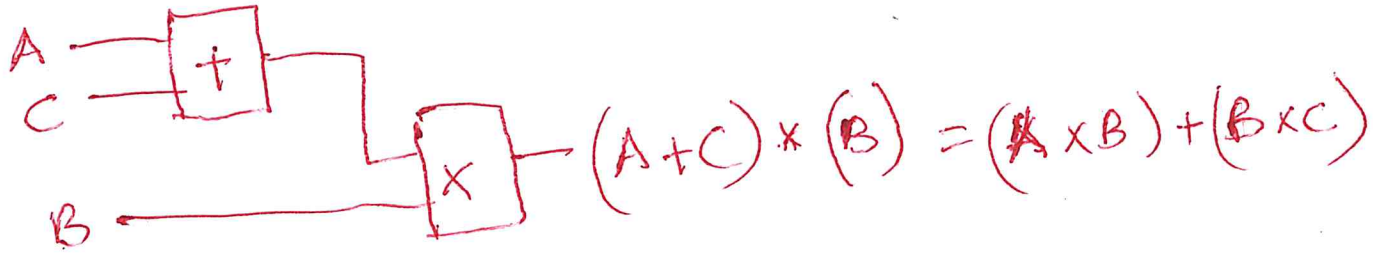
Using "black-box" functions (ABSTRACTION)

Have "boxes" for + and x



Implement AxB + BxC





NUMBERS

UNARY - 1 1 1 1 1 (DECIMAL 5)

DECIMAL - 2 9

$\times 10^1$ $\times 10^0$

ROMAN : X X I X

BINARY : 1 1 1 0 1 $= 29_{10}$

$\times 2^4$ $\times 2^3$ $\times 2^2$ $\times 2^1$ $\times 2^0$

Representing Data Inside a Computer: Start with Unsigned Integers

(IN BINARY)

A CLASS HAS 108 PEOPLE

HOW MANY BITS TO GIVE EACH ONE A UNIQUE ID? → 7 ($2^7 > 108$)

HOW MANY MORE STUDENTS CAN BE ADDED WITH A 7-BIT ID? 20

Unsigned Integers (cont.)

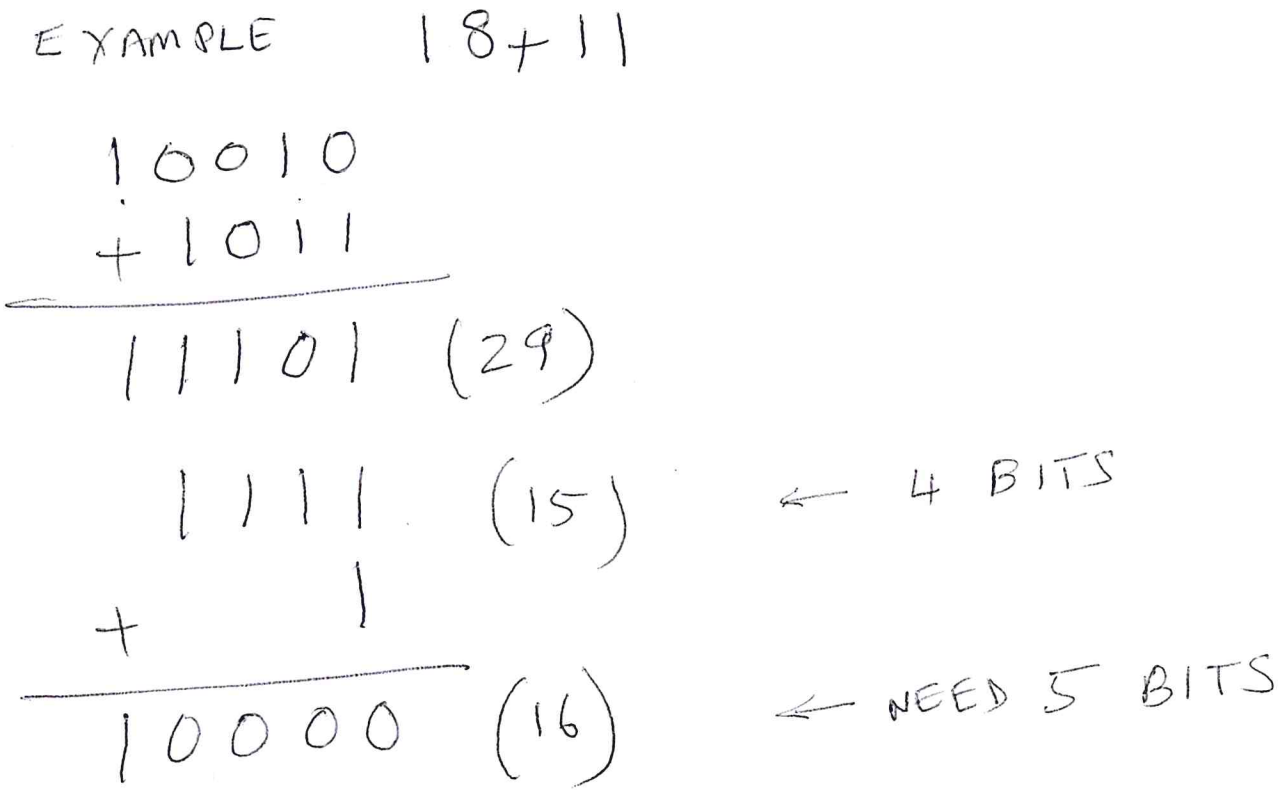
An *n*-bit unsigned integer represents 2^n values: from 0 to 2^n-1.

2^2	2^1	2^0	
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

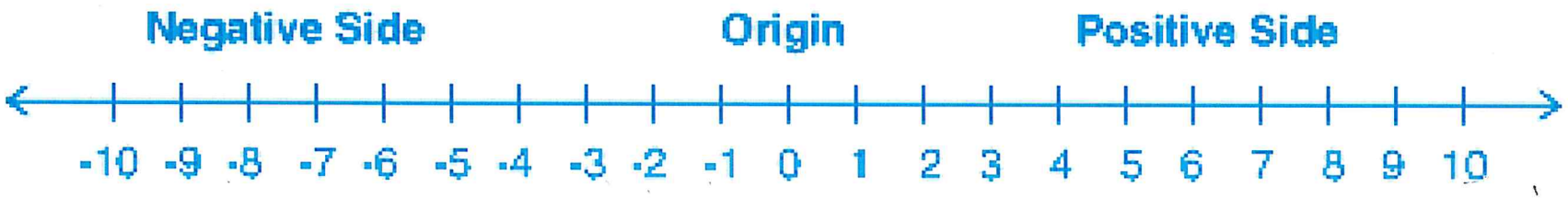
Unsigned Binary Arithmetic

Base-2 addition – just like base-10!

- add from right to left, propagating carry



Positive and Negative Numbers



Signed Integers - Represent Negative Numbers

SIGN MAGNITUDE

0 : POSITIVE
1 : NEGATIVE

SIGN

+5 : 0 | 1 0 1

−5 : 1 | 1 0 1

PROBLEMS:

−5 + 5 ≠ 0

TWO ZEROS 1 0 0 0

0 0 0 0

1's COMPLEMENT

NEGATIVE = COMPLEMENT OF +VE NUMBER

+5 : 0 | 1 0 1

−5 : 1 | 0 1 0

TWO ZEROS

TWOs COMPLEMENT

SIGN

+5 : 0 | 0 1 0 1

−5 : 1 | 1 0 1 1 → 1's COMPLEMENT + 1

Two's Complement Signed Integers

Most Significant bit is sign bit – it has weight -2^{n-1}.

Range of an n-bit number: -2^{n-1} through $2^{n-1} - 1$.

- **The most negative number (-2^{n-1}) has no positive counterpart.**

-2^3	2^2	2^1	2^0	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7

-2^3	2^2	2^1	2^0	
1	0	0	0	-8
1	0	0	1	-7
1	0	1	0	-6
1	0	1	1	-5
1	1	0	0	-4
1	1	0	1	-3
1	1	1	0	-2
1	1	1	1	-1

ADD A NUMBER TO ITSELF

3 + 3

```
  0 0 1 1
+ 0 0 1 1
---------
  0 1 1 0   ← 6
```

0 0 1 1 SHIFTED LEFT

= X 2

SIGN EXTENSION

4-BIT NUMBER: +4 = 0100

PUT IT INTO A 8-BIT MEMORY

+4 = 0100

11

00000100 (8 BITS)

SIGN EXTENSION

$-4 : 1100$

$+4 = 0100$
$\text{COMP} = 1011$
$+1$
1100

$\Downarrow$

8-BIT MEMORY

$1111\,1100$

SIGN EXTENSION

ADDITION

[HOW DO YOU COMPUTE A - B?

A + (-B)]

$104_{10} + (-16_{10})$

$$\begin{array}{rll} & 01101000 & (104_{10}) \\ + & 11110000 & (-16_{10}) \\ \hline 1 \leftarrow & 01011000 & \end{array}$$

10000

$\hookrightarrow (+88_{10})$ COMP = 00 1111

$+1$

$11\,0000$

2's COMPLEMENT

ANOTHER RULE

1. COPY BITS FROM R (RIGHT) TO L (LEFT) TILL THE FIRST "1"
2. THEN FLIP THE REMAINING BITS TO THE LEFT

$X = D1101\,0000$

FIRST "1"

$-X = 1001\,1\,0000$