ELL782 – Computer Architecture

Department of Electrical Engineering, IIT-Delhi
Kaushik Saha
ksaha@ee.iitd.ac.in

High Level List of Topics

Kaushik Saha

- Review of Boolean logic
  - Arithmetic using Boolean logic
    - Number systems, Adders, Subtractors
- RISC V ISA (Instruction set architecture)
  - Instruction format
  - Assembly language
  - Execution units – Multipliers, Dividers
- Single cycle CPU architecture

Prof. Smruti Sarangi

- In Order CPU Pipeline
- Out of order CPU Pipeline
- Multiprocessor systems
- Other advanced topics

Grading

- 2 assignments – 2 x 15 marks = 30 marks
- 1 minor = 30 marks
- 1 major = 40 marks
- Total marks for grading = 100 marks

Audit criteria: Student opting for Audit must obtain 40/100 in aggregate for Audit Pass

Grading: Relative grading on the marks distribution curve

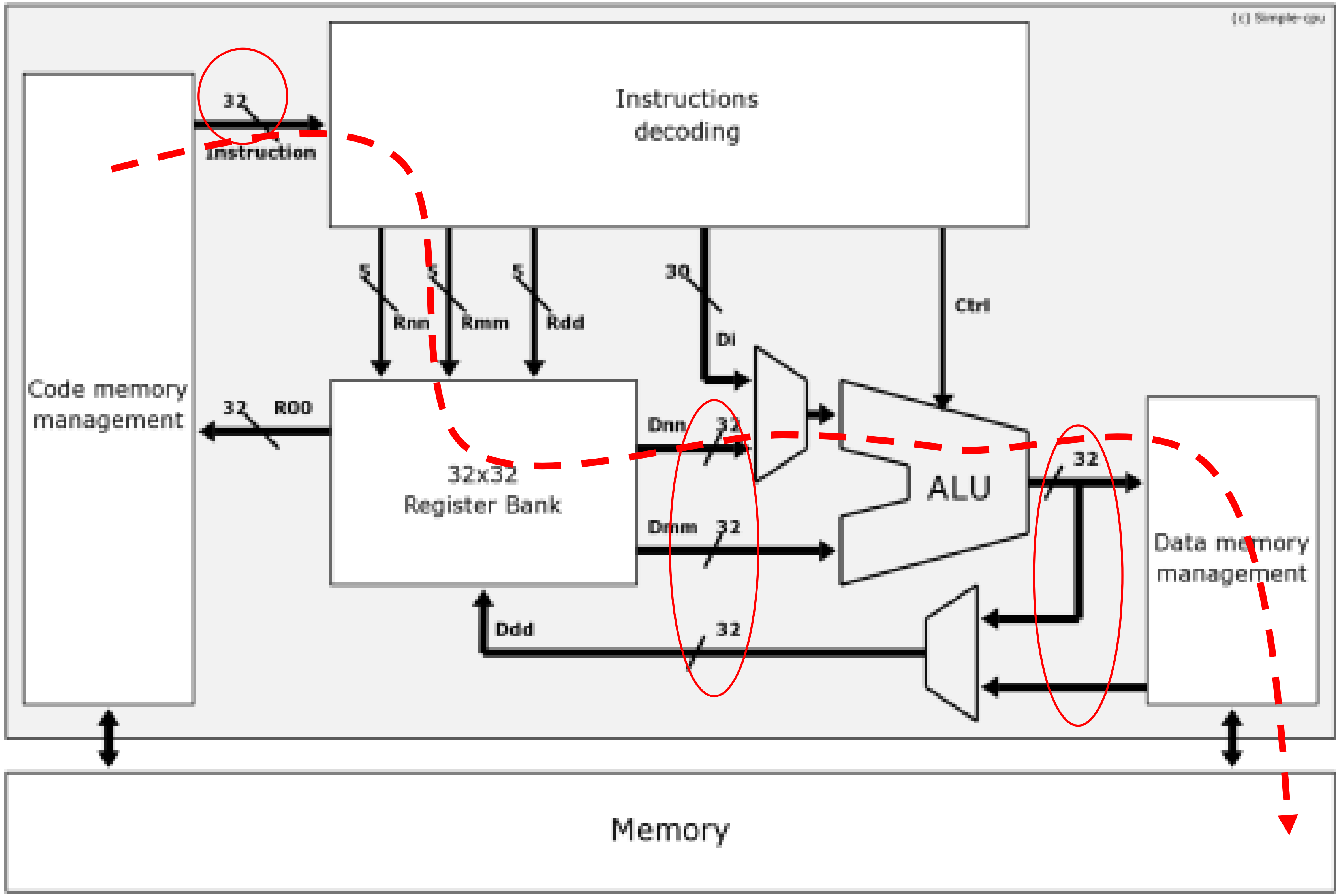
Course Policy

1. All deadlines are final. No deadline will be extended under any circumstances. Delay of even 1 minute beyond deadline will not be permitted.
2. A student who misses the major exam must apply for an I grade. The academic section will decide the eligibility for a re-major
3. If any student getting an E or I grade will only be eligible to write the re-major. The student will not be allowed to submit any assignment after the deadline.
4. If a student who misses the minor exam or an assignment deadline is eligible for a re-minor or the third assignment. These will be considerably more difficult than the regular minor or any of the assignments, respectively. No medical certificate needs to be submitted.
5. Instructors will not entertain any direct e-mail communication. All the questions need to be asked via the Piazza group for this course.
    1. https://piazza.com/iit_delhi/summer2024/ell782 ; access code: ell782
6. Zero-tolerance policy for plagiarism. A plagiarized submission will be given 0.0 marks. The instructor's judgement is final.
7. All standard rules and penalties of IIT-Delhi regarding academic dishonesty apply. They are tough, so do not break the rules.

Texts

- Basic Computer Architecture – S.R.Sarangi
- Next-Gen Computer Architecture – S.R.Sarangi
- URL https://www.cse.iitd.ac.in/~srsarangi/archbooksoft.html
- Computer Arithmetic: Algorithms and Hardware Designs – Behrooz Parhami -

Simple CPU Architecture



Why 32 bits?

Scope of Computer Arithmetic

Hardware

Design of efficient digital circuits for primitive and other arithmetic operations such as +, –, ×, ÷, √, log, sin, and cos

Issues: Algorithms
Precision (Error analysis)
Dynamic range (smallest to largest val)
Speed/cost trade-offs
Hardware implementation
Testing, verification

Software

Numerical methods for solving systems of linear equations, partial differential eq'ns, and so on

Issues: Algorithms
Computational complexity
Programming
Testing, Verification

General-purpose

Flexible data paths
Fast primitive operations like +, –, ×, ÷, √

Special-purpose

Tailored to application areas such as:
Digital filtering
Image processing
Graphics & gaming
Machine learning (neuromorphic processing)

Example of Finite Precision Problems

Failure of Patriot Missile (1991 Feb. 25)

Source http://www.ima.umn.edu/~arnold/disasters/disasters.html

American Patriot Missile battery in Dharan, Saudi Arabia, failed to intercept incoming Iraqi Scud missile

The Scud struck an American Army barracks, killing 28

Cause, per GAO/IMTEC-92-26 report: "software problem" (inaccurate calculation of the time since boot)

Problem specifics:

Time in tenths of second as measured by the system's internal clock was multiplied by 1/10 to get the time in seconds

Internal registers were 24 bits wide

1/10 = 0.0001 1001 1001 1001 1001 100 (chopped to 24 b)

Error $\approx 0.1100\ 1100 \times 2^{-23} \approx 9.5 \times 10^{-8}$

Error in 100-hr operation period

$\approx 9.5 \times 10^{-8} \times 100 \times 60 \times 60 \times 10 = 0.34$ s

Distance traveled by Scud = $(0.34\text{ s}) \times (1676\text{ m/s}) \approx 570$ m

Example of Inadequate Dynamic Range Problem

Example: Explosion of Ariane Rocket (1996 June 4)

Source http://www.ima.umn.edu/~arnold/disasters/disasters.html

Unmanned Ariane 5 rocket of the European Space Agency veered off its flight path, broke up, and exploded only 30 s after lift-off (altitude of 3700 m)

The $500 million rocket (with cargo) was on its first voyage after a decade of development costing $7 billion

Cause: "software error in the inertial reference system"

Problem specifics:

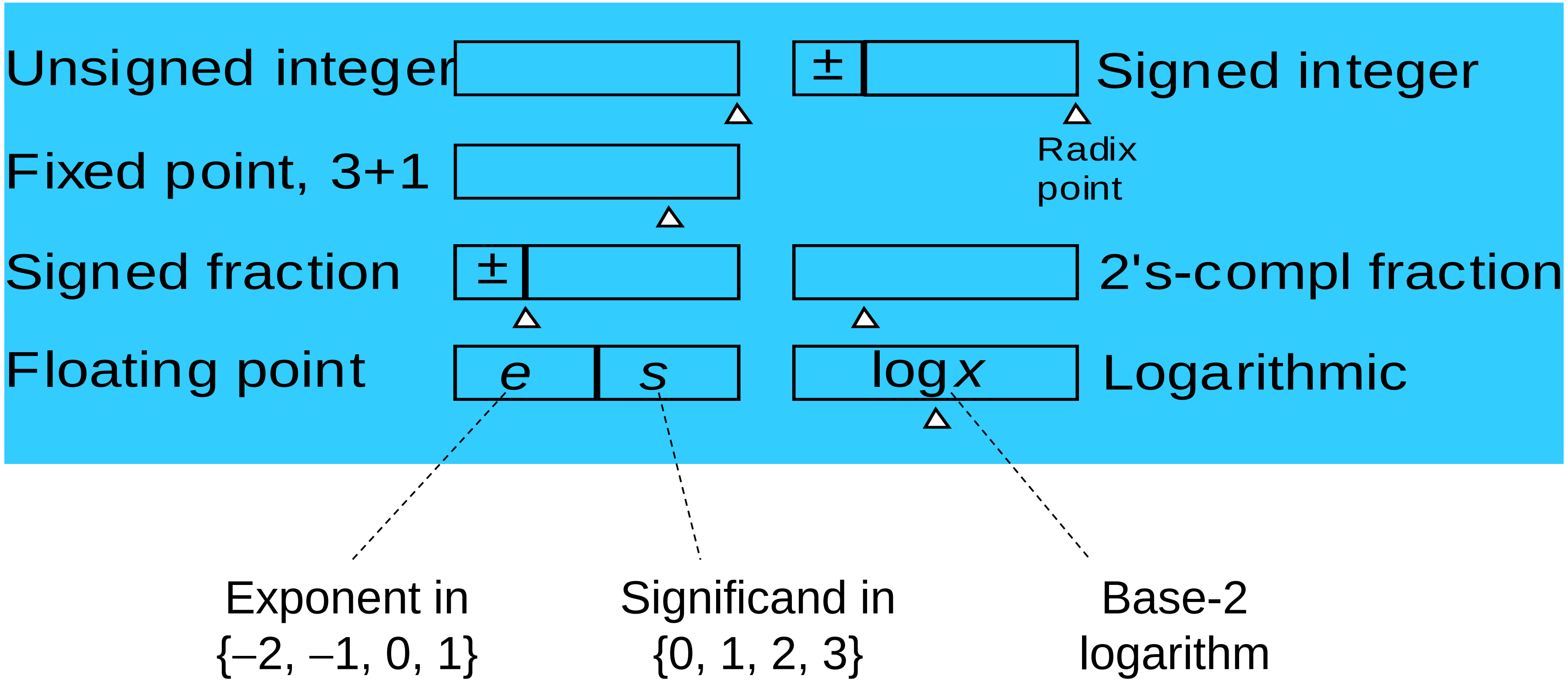
A 64 bit floating point number relating to the horizontal velocity of the rocket was being converted to a 16 bit signed integer

An SRI* software exception arose during conversion because the 64-bit floating point number had a value greater than what could be represented by a 16-bit signed integer (max 32 767)

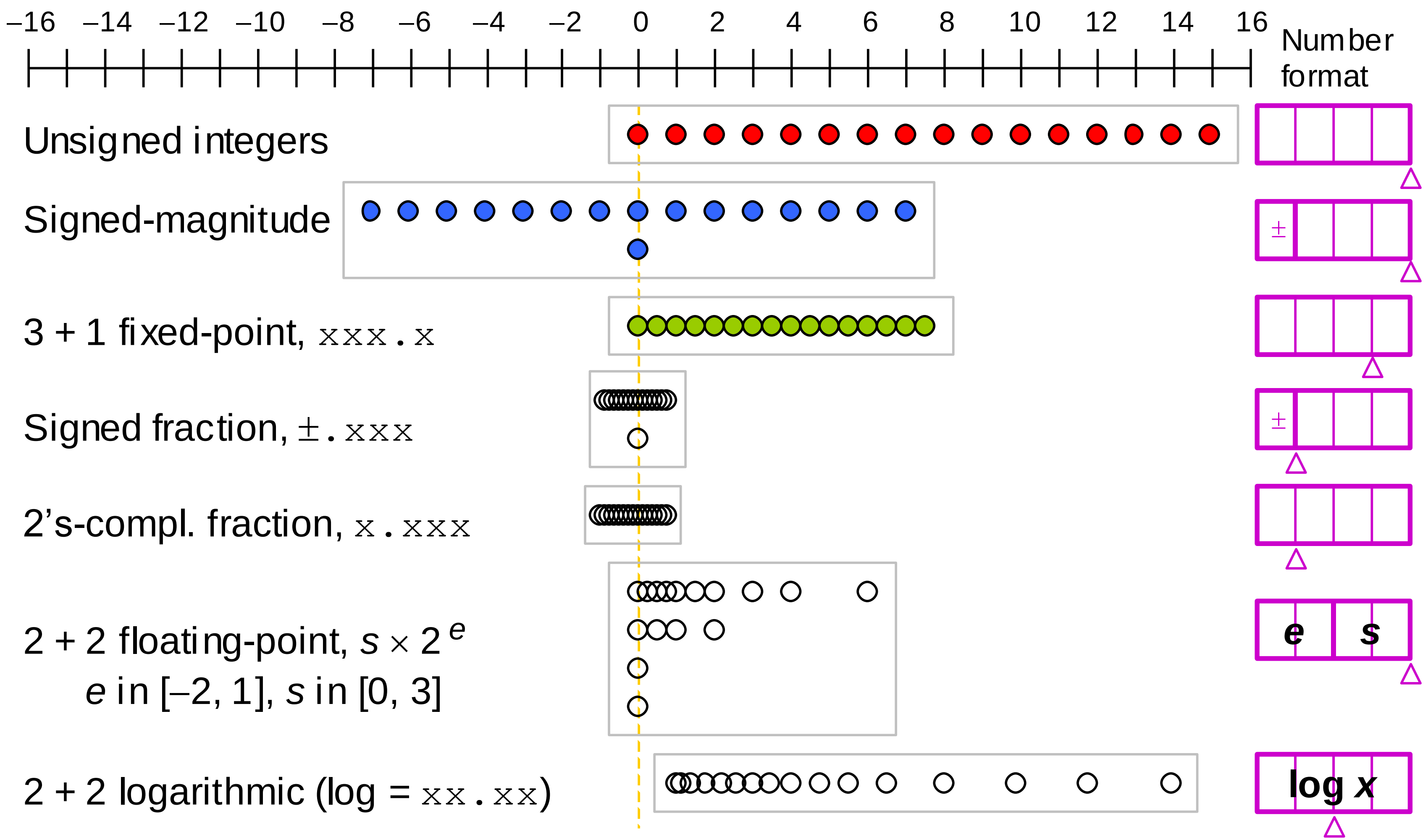
*SRI = Système de Référence Inertielle or Inertial Reference System

Numbers and Their Encodings

Some 4-bit number representation formats



Possible Ways of Encoding Numbers in 4 Bits



Some of the possible ways of assigning 16 distinct codes to represent numbers. Small triangles denote the radix point locations.

1.4 Fixed-Radix Positional Number Systems

$$(x_{k-1}x_{k-2} \dots x_1x_0 \,.\, x_{-1}x_{-2} \dots x_{-l})_r = \sum_{i=-l}^{k-1} x_i r^i$$

One can generalize to:

Arbitrary radix (not necessarily integer, positive, constant)

Arbitrary digit set, usually $\{-\alpha, -\alpha+1, \dots, \beta-1, \beta\} = [-\alpha, \beta]$

Example Balanced ternary number system:

Radix $r = 3$, digit set = $[-1, 1]$

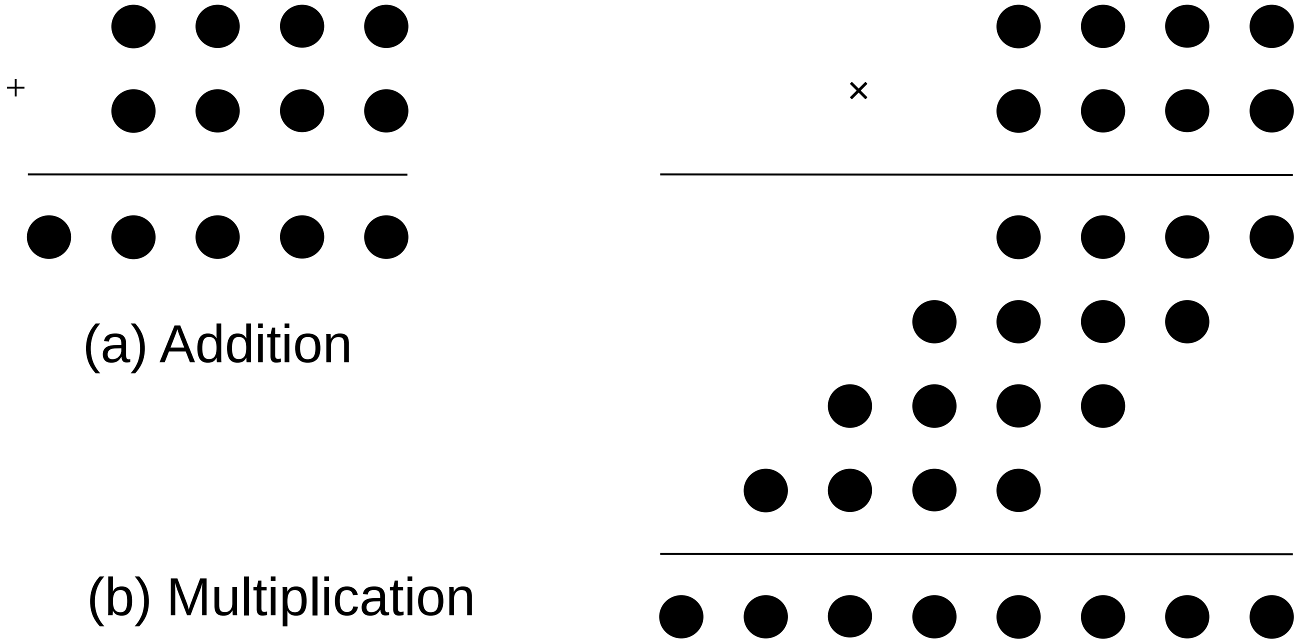
Example Digit set $[-4, 5]$ for $r = 10$:

$(3\ {}^{-}1\ \ 5)_{ten}$ represents $295 = 300 - 10 + 5$

Example Digit set $[-7, 7]$ for $r = 10$:

$(3\ {}^{-}1\ \ 5)_{ten} = (3\ \ 0\ {}^{-}5)_{ten} = (1\ {}^{-}7\ \ 0\ {}^{-}5)_{ten}$

Dot Notation: A Useful Visualization Tool

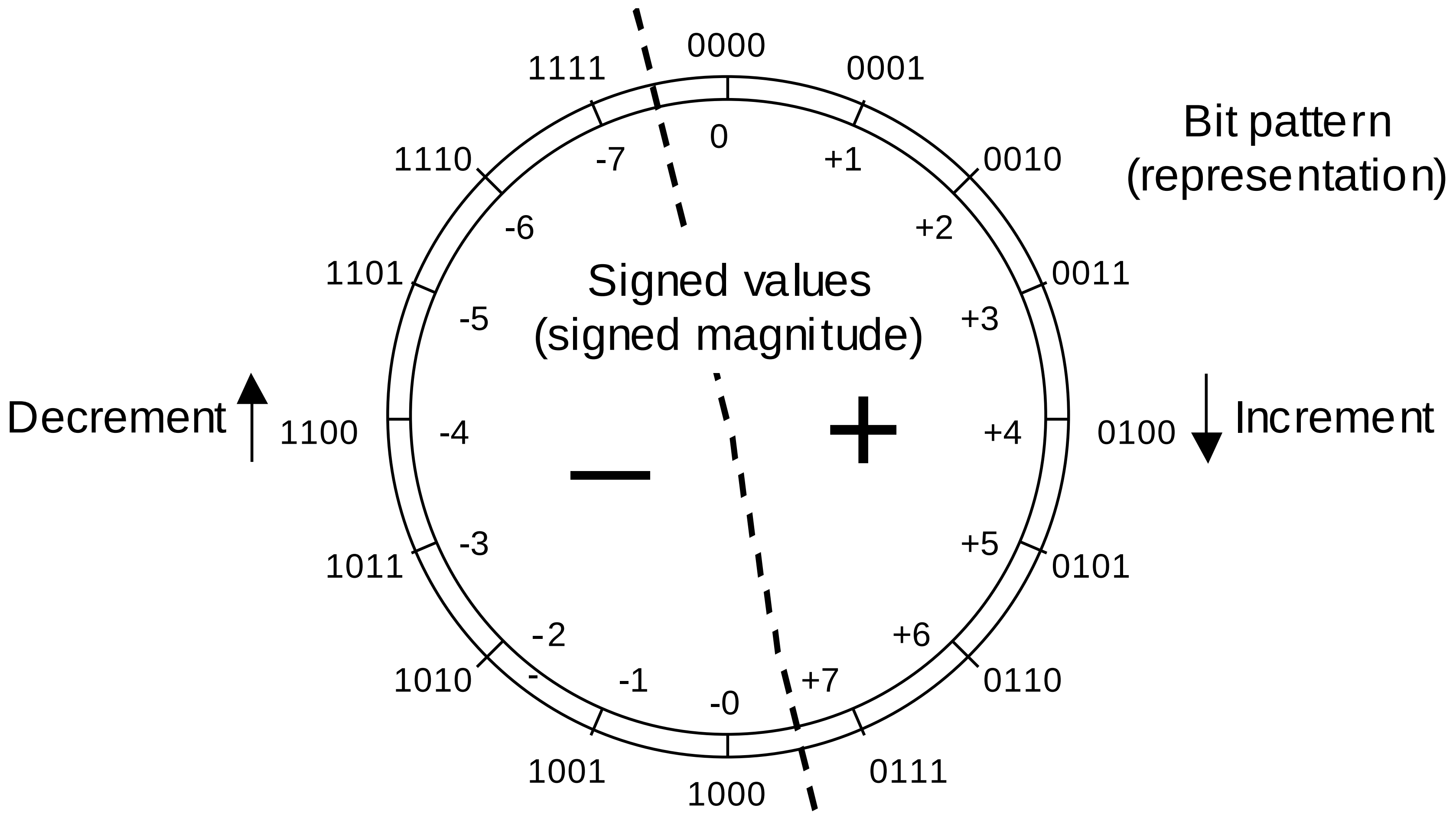


(a) Addition

(b) Multiplication

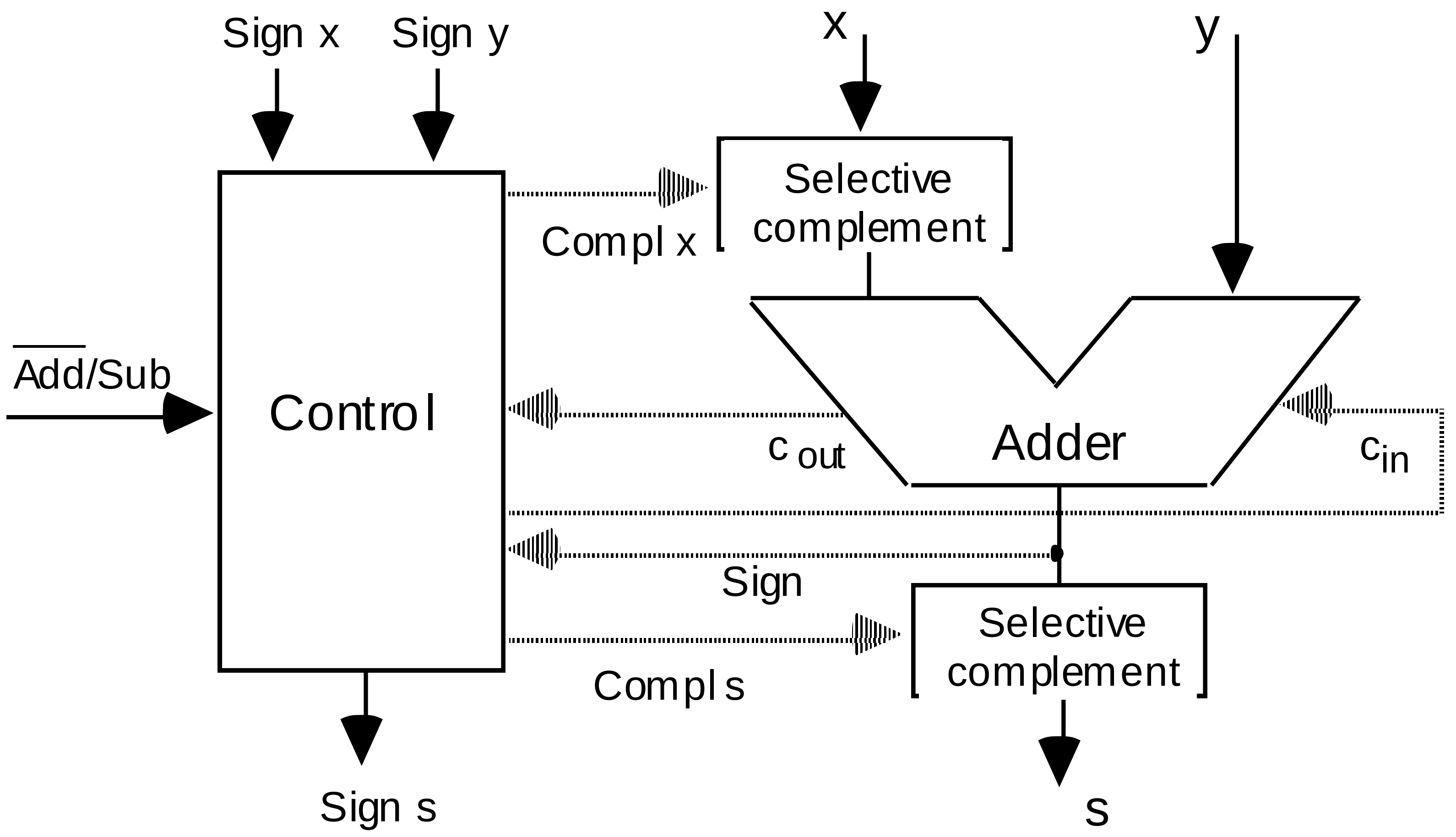
Dot notation to depict number representation formats and arithmetic algorithms.

Signed-Magnitude Representation



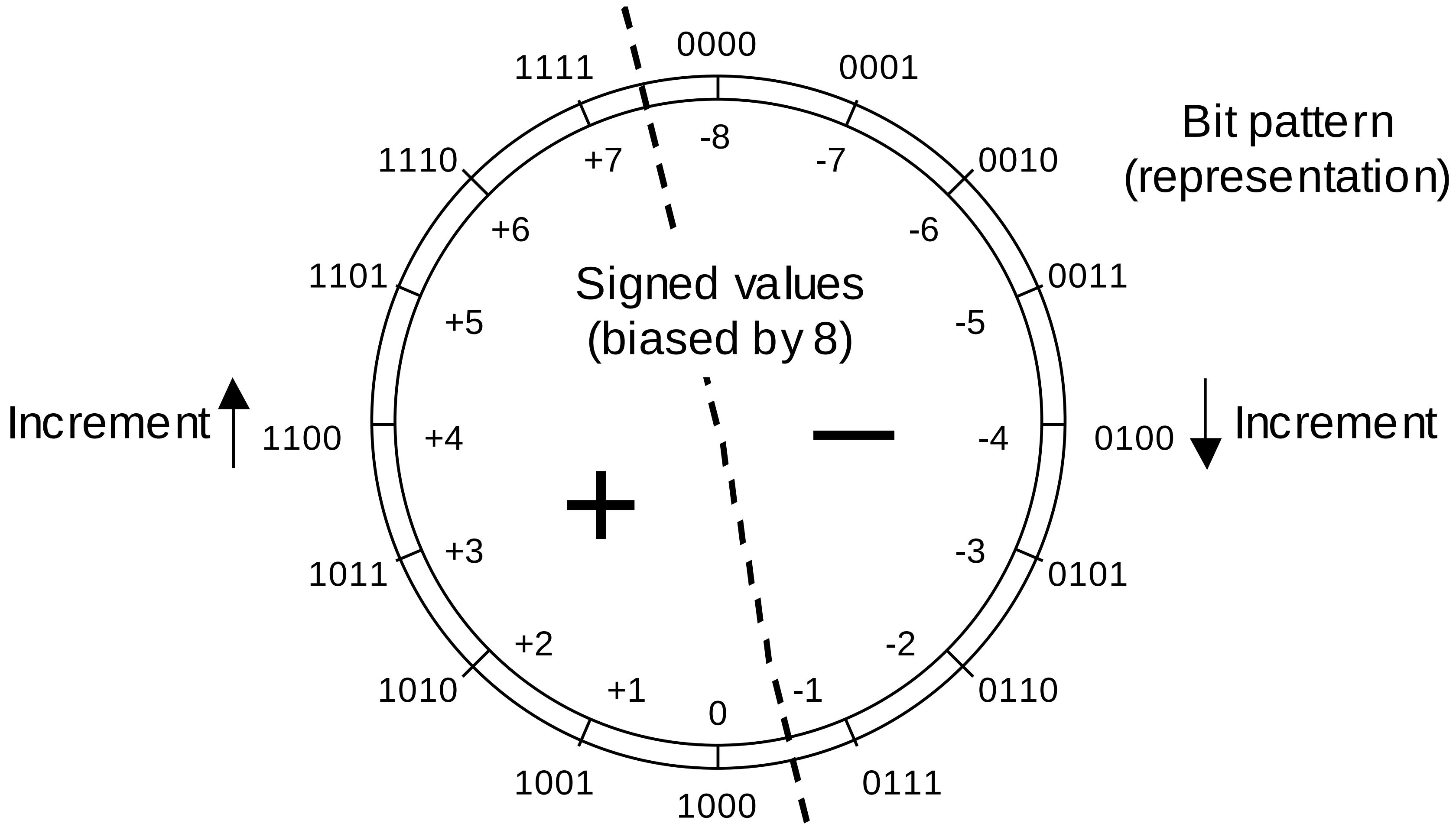
4-bit signed-magnitude number representation system for integers.

Signed-Magnitude Adder – y ± x



Adding signed-magnitude numbers using pre-complementation and post-complementation.

Biased Representations



A 4-bit biased integer number representation system with a bias of 8.

Arithmetic with Biased Numbers

Addition/subtraction of biased numbers

$$x + y + bias = (x + bias) + (y + bias) - bias$$
$$x - y + bias = (x + bias) - (y + bias) + bias$$

A power-of-2 (or $2^a - 1$) bias simplifies addition/subtraction

Comparison of biased numbers:

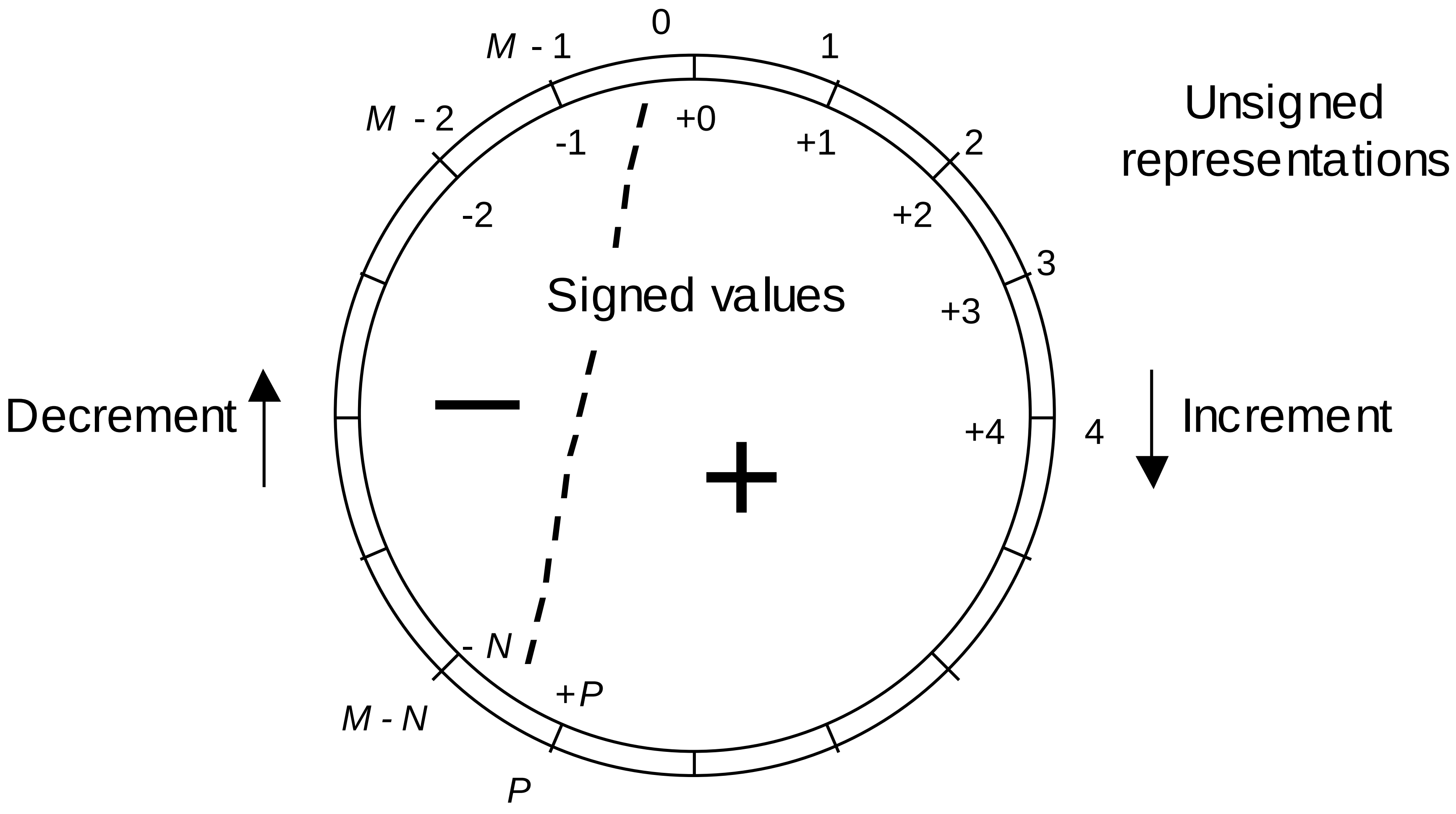
Compare like ordinary unsigned numbers

find true difference by ordinary subtraction

We seldom perform arbitrary arithmetic on biased numbers

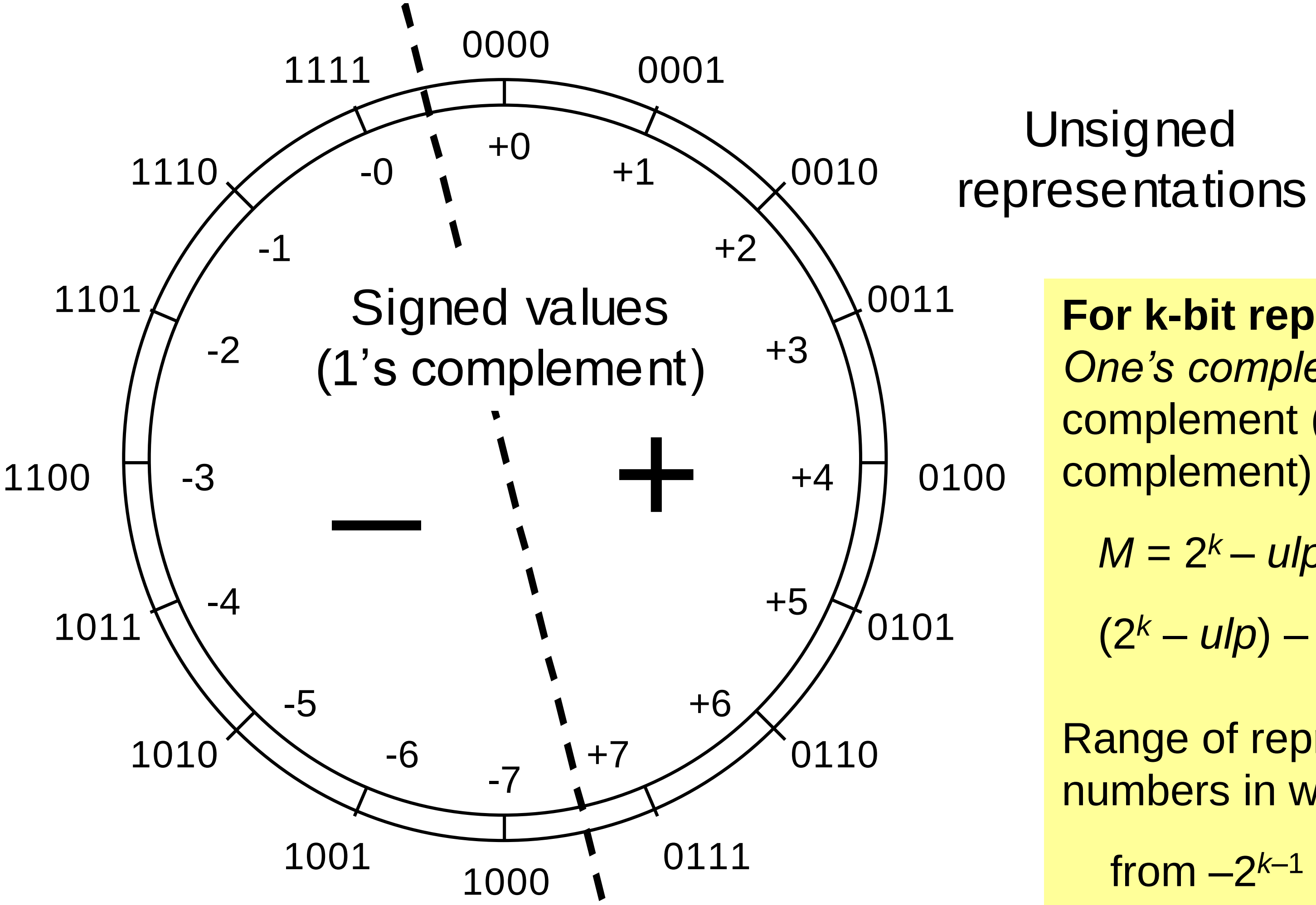
Main application: Exponent field of floating-point numbers

Complement Representations



Complement representation of signed integers.

1's-Complement Number Representation (k-bit registers)



For k-bit representation
One's complement = digit complement (diminished radix complement) system for $r = 2$

$M = 2^k - ulp$ (e.g. ulp =1)

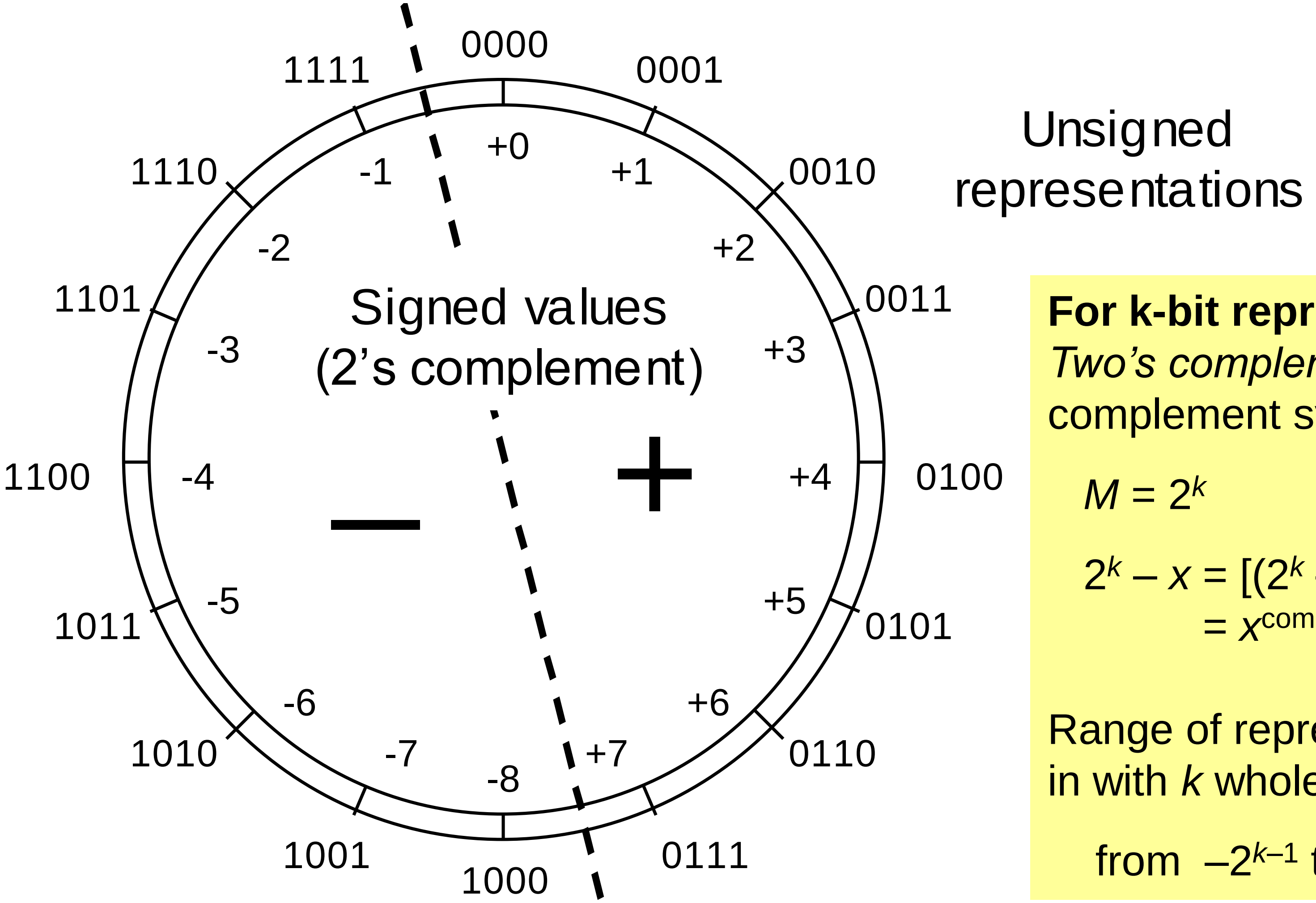
$(2^k - ulp) - x = x^{compl}$

Range of representable numbers in with k whole bits:

from $-2^{k-1} + ulp$ to $2^{k-1} - ulp$

A 4-bit 1's-complement number representation system for integers.

2's-Complement Numbers



For k-bit representation
Two's complement = radix complement system for $r = 2$

$M = 2^k$

$2^k - x = [(2^k - ulp) - x] + ulp$
$= x^{compl} + ulp$ *(e.g. ulp=1)*

Range of representable numbers in with k whole bits:

from -2^{k-1} to $2^{k-1} - ulp$

A 4-bit 2's-complement number representation system for integers.

Arithmetic with Complement Representations

Addition in a complement number system with complementation constant *M* and range [–*N*, +*P*]

Desired operation	Computation to be performed mod M	Correct result with no overflow	Overflow condition
$(+x) + (+y)$	$x + y$	$x + y$	$x + y > P$
$(+x) + (–y)$	$x + (M – y)$	$x – y$ if $y \le x$ $M – (y – x)$ if $y > x$	N/A
$(–x) + (+y)$	$(M – x) + y$	$y – x$ if $x \le y$ $M – (x – y)$ if $x > y$	N/A
$(–x) + (–y)$	$(M – x) + (M – y)$	$M – (x + y)$	$-x + -y > -N$

Some Details for 2's- and 1's Complement

Range/precision extension for 2's-complement numbers

$$\ldots x_{k-1}\, x_{k-1}\, x_{k-1}\, \boxed{x_{k-1}\, x_{k-2} \ldots x_1\, x_0 \,.\, x_{-1}\, x_{-2} \ldots x_{-l}}\; 0\;\; 0\;\; 0 \ldots$$

← Sign extension → Sign bit LSD ← Extension →

Range/precision extension for 1's-complement numbers

$$\ldots x_{k-1}\, x_{k-1}\, x_{k-1}\, \boxed{x_{k-1}\, x_{k-2} \ldots x_1\, x_0 \,.\, x_{-1}\, x_{-2} \ldots x_{-l}}\; x_{k-1}\, x_{k-1}\, x_{k-1} \ldots$$

← Sign extension → Sign bit LSD ← Extension →

Mod-2^k operation needed in 2's-complement arithmetic is trivial:
Simply drop the carry-out (subtract 2^k if result is 2^k or greater)

Mod-(2^k – *ulp*) operation needed in 1's-complement arithmetic is done via end-around carry

$(x + y) - (2^k - ulp) = (x - y - 2^k) + ulp$ Connect c_{out} to c_{in}

Which Complement System Is Better?

Comparing radix- and digit-complement number representation systems

Feature/Property	Radix complement	Digit complement
Symmetry ($P = N$?)	Possible for odd r (radices of practical interest are even)	Possible for even r
Unique zero?	Yes	No, there are two 0s
Complementation	Complement all digits and add *ulp*	Complement all digits
Mod-M addition	Drop the carry-out	End-around carry

Using Signed Positions or Signed Digits

A key property of 2’s-complement numbers that facilitates direct signed arithmetic:

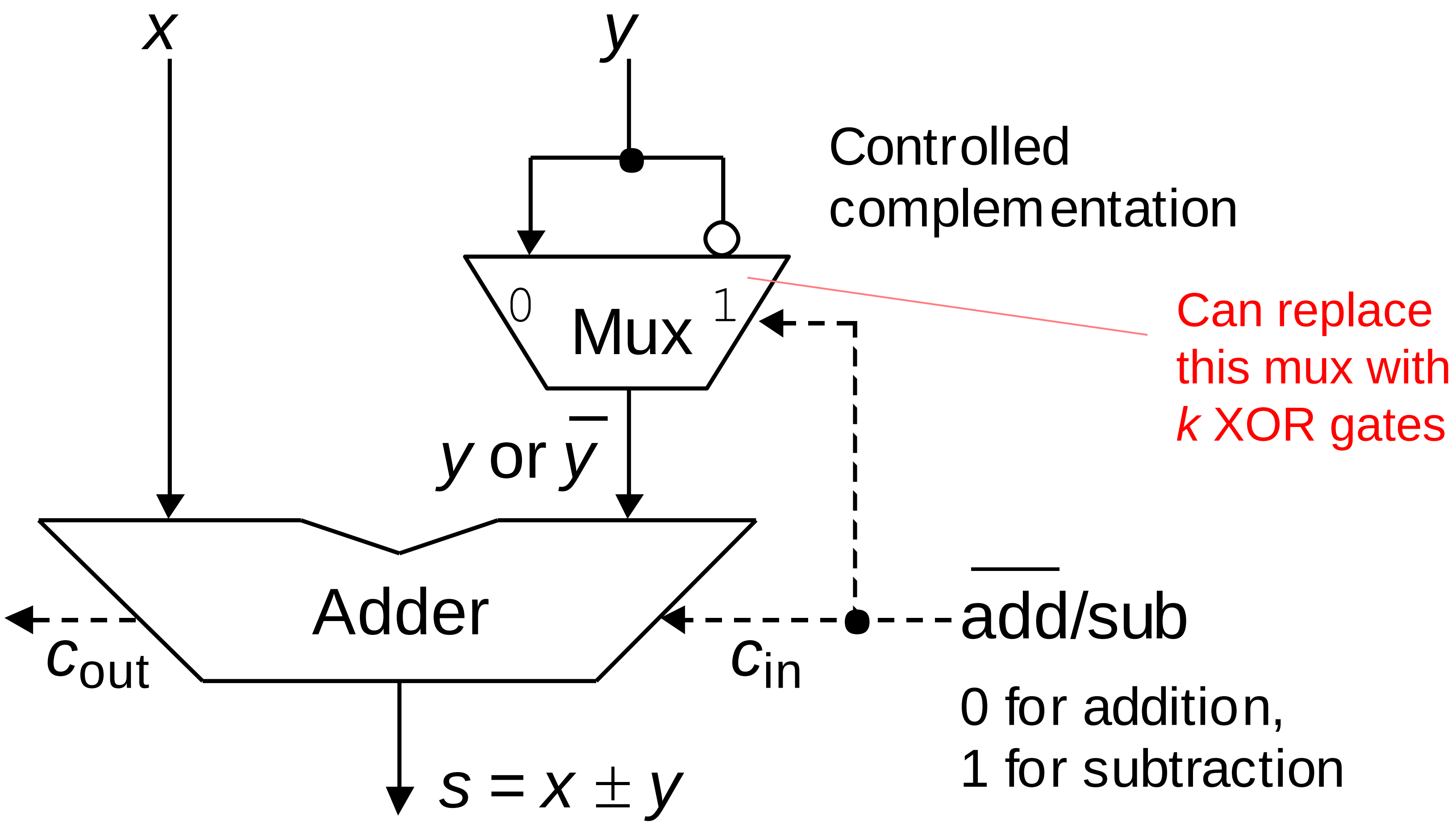
x =	(1	0	1	0	0	1	1	$0)_{\text{two's-compl}}$
	-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	–128	+	32	+		4 +	2	= –90

Check:

x =	(1	0	1	0	0	1	1	$0)_{\text{two's-compl}}$
$-x$ =	(0	1	0	1	1	0	1	$0)_{\text{two}}$
	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
		64	+	16 +	8	+	2	= 90

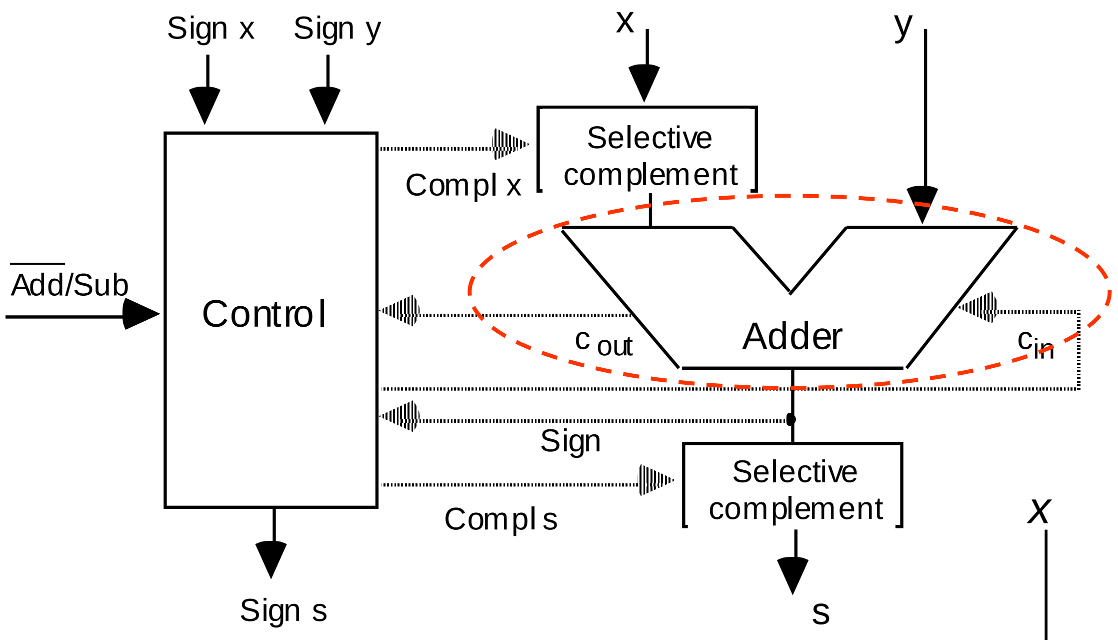
Interpreting a 2’s-complement number as having a negatively weighted most-significant bit.

Why 2's-Complement Is the Universal Choice



Adder/subtractor architecture for 2's-complement numbers.

Signed-Magnitude vs 2's-Complement



Signed-magnitude adder/subtractor is significantly more complex than a simple adder

2's-complement adder/subtractor needs very little hardware other than a simple adder