UNSIGNED NUMBERS

Computer Arithmetic-II

Signed binary numbers

Binary number representation

1. Representing binary signed integers:
   a) Sign-magnitude
   b) One’s complement signed magnitude
   c) Two's complement signed magnitude

2. Representing binary integer/fractions:
   a) Fixed-point numbers
   b) Floating-point numbers

Review

Unsigned (positive) numbers

Range of unsigned binary numbers

- The range is: 0 up to $2^n - 1$ (min – max)

- Example: n = 3:

$$0 \rightarrow [2^3 - 1] = 7$$

3-bit unsigned binary number

Decimal	Unsigned
7	111
6	110
5	100
4	100
3	011
2	010
1	001
0	000

8-bit unsigned binary numbers

Binary	Unsigned
00000000	0
00000001	1
...	...
01111111	127
10000000	128
10000001	129
...	...
11111111	255

$$255 = 2^8 - 1$$

Largest unsigned 32-bit integer

- For a 32-bit computer …
- What is the largest unsigned integer that can hold?

11111111-11111111-11111111-11111111

32-bits

$2^{32}-1$ = 4, 294, 967, 295

4 Giga

Binary unsigned numbers

$(54.5)_{10} = (110110.1)_2$

2^5	2^4	2^3	2^2	2^1	2^0		2^{-1}	2^{-2}	2^{-3}
1	**1**	**0**	**1**	**1**	**0**	•	**1**	**0**	...

$$(110110.1)_2 = 1 * 2^5 + 1 * 2^4 + 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 0 * 2^0 + 1 * 2^{-1}$$
$$= 32 + 16 + 0 + 4 + 2 + 0 + 0.5$$
$$= 54.5$$

1. Representing binary signed integers

a) Signed-magnitude (review)
b) 1’s complement signed-magnitude
c) 2’s complement signed-magnitude

Unsigned-magnitude (review)

- A n-bit binary number has 2^n distinct values.

Decimal	Unsigned
+7	111
+6	110
+5	100
+4	100
+3	011
+2	010
+1	001
0	000

Signed-magnitude

a) Signed-magnitude

- 1 << leftmost bit = Negative number
- 0 << leftmost bit = Positive number

Decimal	signed	
-3	1 11	NEGATIVE
+3	0 11	POSITIVE

a) Signed-magnitude

- A n-bit binary number has 2^n distinct values.
  - Assign **Half to negative (One MSB)**
  - and **Half to positive integers (Zero MSB)**
  - with two values **for the Zero (0**)

Decimal	Unsigned
+7	111
+6	110
+5	101
+4	100
+3	011
+2	010
+1	001
0	000

Decimal	signed
-3	1 11
-2	1 10
-1	1 01
-0	1 00
+3	0 11
+2	0 10
+1	0 01
+0	0 00

NEGATIVE

POSITIVE

3-bit signed-magnitude numbers

- A 3-bit binary number has a total of $\mathbf{2^3}$ = 8 numbers.
- Using signed number representation, we have negative and positive numbers: Divide 8 by 2 ➔ 4. We have:
  - 4-positive (+0 → +3)
  - 4-negative (-0 → -3) numbers.
- The only «problem» is that we have 2 different zeros (-0, +0).

The range is: +3 → -3
We have 2 ways to represent zero

Decimal	Unsigned	Signed
+7	111	
+6	110	
+5	100	
+4	100	
+3	011	011
+2	010	010
+1	001	001
+0	000	000
-0		100
-1		101
-2		110
-3		111
-4		

Range: Signed-magnitude numbers

The range is: +3 → -3

In general

$2^{n-1}-1$ to $-(2^{n-1}-1)$

Decimal	Unsigned	Signed
+7	111	
+6	110	
+5	100	
+4	100	
+3	011	011
+2	010	010
+1	001	001
+0	000	000
-0		100
-1		101
-2		110
-3		111
-4		

Signed-magnitude

Not used in modern computing … because there are two representations for the zero.

IBM7090/ 1959

In general: Range of n-bit numbers

bits	Binary			
	Unsigned		Sign-magnitude	
	Min	Max	Min	Max
n	0	$2^{n} - 1$	$-(2^{n-1} - 1)$	$2^{n-1} - 1$

b) 3-bit 1's complement signed-magnitude

The range is: +3 → -3

In general

$2^{n-1}-1$ to $-(2^{n-1}-1)$

We have 2 ways to represent zero

Decimal	Unsigned	Signed	1's Comp.
+7	111		
+6	110		
+5	100		
+4	100		
+3	011	011	011
+2	010	010	010
+1	001	001	001
→ +0	000	000	000
→ -0		100	111
-1		101	110
-2		110	101
-3		111	100
-4			

1’s complement signed-magnitude

Not used in modern computing … because there are two representations for the zero

CDC160A/1960
UNIVAC 1100/2200 /1962

c) 3-bit 2's complement signed magnitude

Decimal	Unsigned	Signed	1's Comp.	2's Comp
+7	111			
+6	110			
+5	100			
+4	100			
+3	011	011	011	011
+2	010	010	010	010
+1	001	001	001	001
+0	000	000	000	000
-0		100	111	000
-1		101	110	111
-2		110	101	110
-3		111	100	101
-4				100

The range is:
Positive (+0 → +3)
Negative (-1 → -4)
… we have 1 way to represent zero

2’s complement signed-magnitude

- Today’s processors represent signed integers using two’s complement …

- Why?

- Because a 2’s complement signed-magnitude representation has a single representation for zero (0)

Range for n-bit 2's complement signed

Range: $-2^{n-1} \rightarrow 2^{n-1} - 1$

For our 3-bit [$2^3 = 8$ … divide by 2 $\rightarrow$ $4 = 2^2$]
For our example the range is ($-4 \rightarrow +3$):

-

$2^{3-1} \rightarrow 2^{3-1} - 1 = -2^2 \rightarrow 2^2 - 1 = -4 \rightarrow +3$

Modulo-16 system (4-bits)

B	Value represented		
$b_3 b_2 b_1 b_0$	Sign and magnitude	1's complement	2's complement
0 1 1 1	+ 7	+ 7	+ 7
0 1 1 0	+ 6	+ 6	+ 6
0 1 0 1	+ 5	+ 5	+ 5
0 1 0 0	+ 4	+ 4	+ 4
0 0 1 1	+ 3	+ 3	+ 3
0 0 1 0	+ 2	+ 2	+ 2
0 0 0 1	+ 1	+ 1	+ 1
0 0 0 0	+ 0	+ 0	+ 0
1 0 0 0	- 0	- 7	- 8
1 0 0 1	- 1	- 6	- 7
1 0 1 0	- 2	- 5	- 6
1 0 1 1	- 3	- 4	- 5
1 1 0 0	- 4	- 3	- 4
1 1 0 1	- 5	- 2	- 3
1 1 1 0	- 6	- 1	- 2
1 1 1 1	- 7	- 0	- 1

In general: Range of n-bit numbers

bits	Binary					
	Unsigned		Sign/Magnitude		2's Complement	
	Min	Max	Min	Max	Min	Max
n	0	$2^n - 1$	$-(2^{n-1} - 1)$	$2^{n-1} - 1$	-2^{n-1}	$2^{n-1} - 1$

Range of 32-bit 2's comp. signed numbers

$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_2 = 0_{10}$

$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001_2 = 1_{10}$

…

$0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_2 = 2^{31}-1$

$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_2 = -2^{31}$

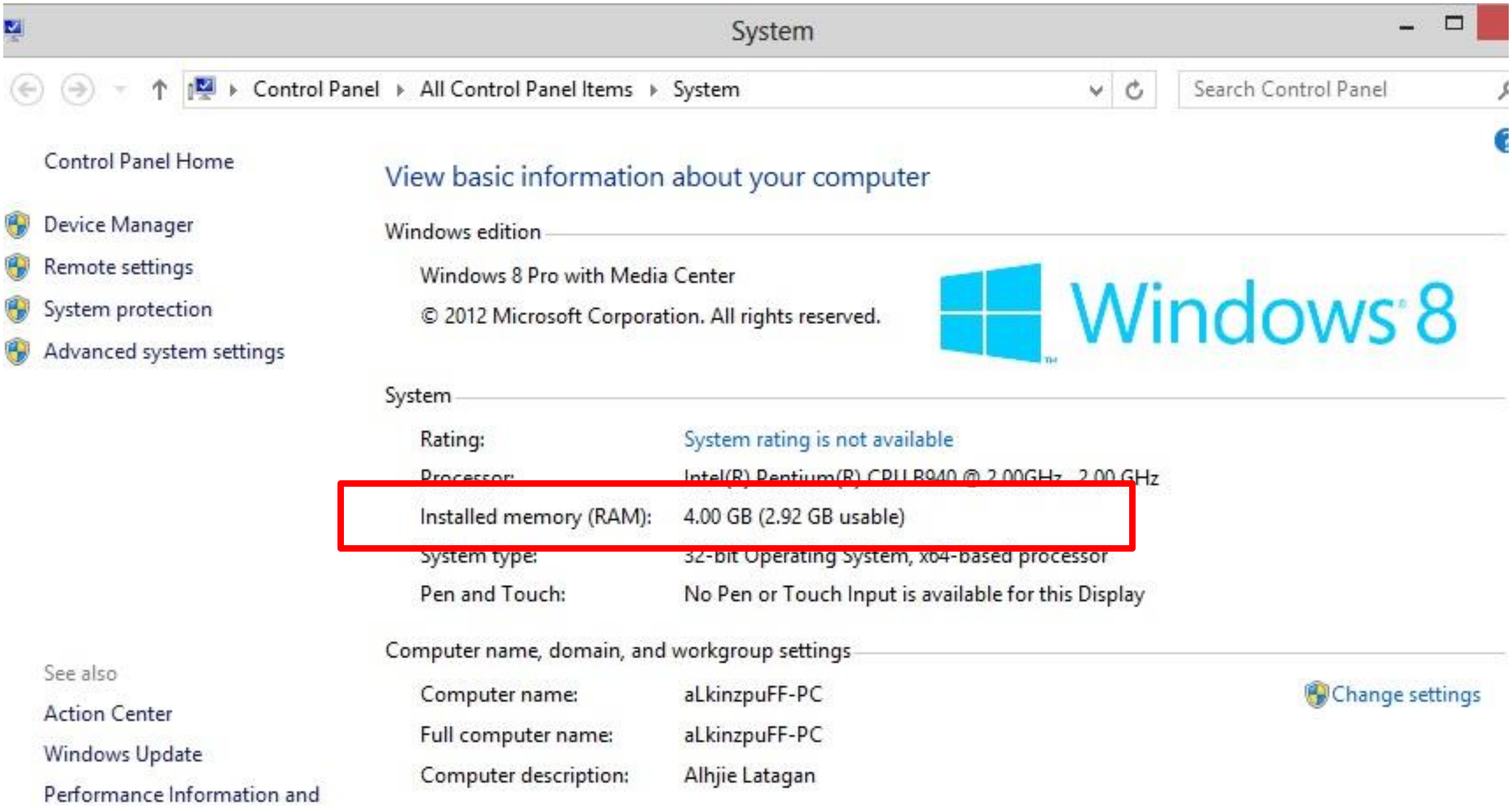
$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001_2 = -(2^{31} - 1)$

$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010_2 = -(2^{31} - 2)$

$2^{31}-1 = 4{,}294{,}967{,}295$

4 Giga

32 bit OS - 4 GB RAM



32-bit and 64-bit

32-bit:

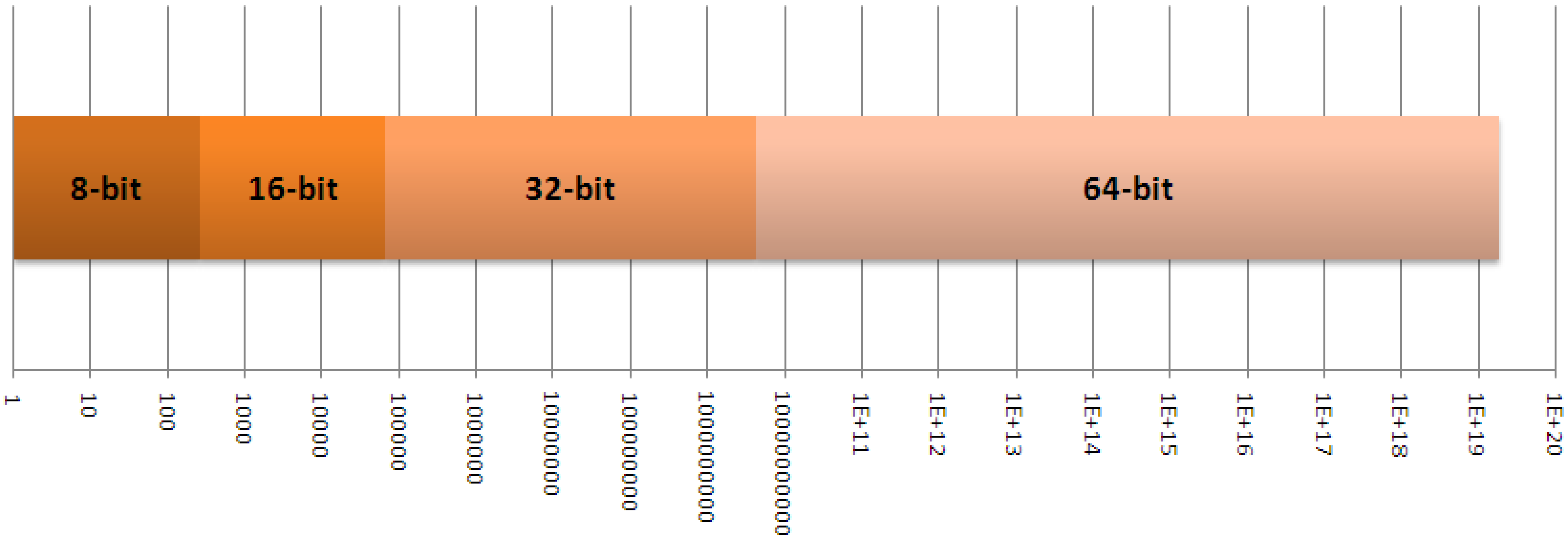
2^{32} = 4,294,967,296
4,294,967,296 / (1,024 x 1,024) = 4,096 MB = 4GB (gigabytes)

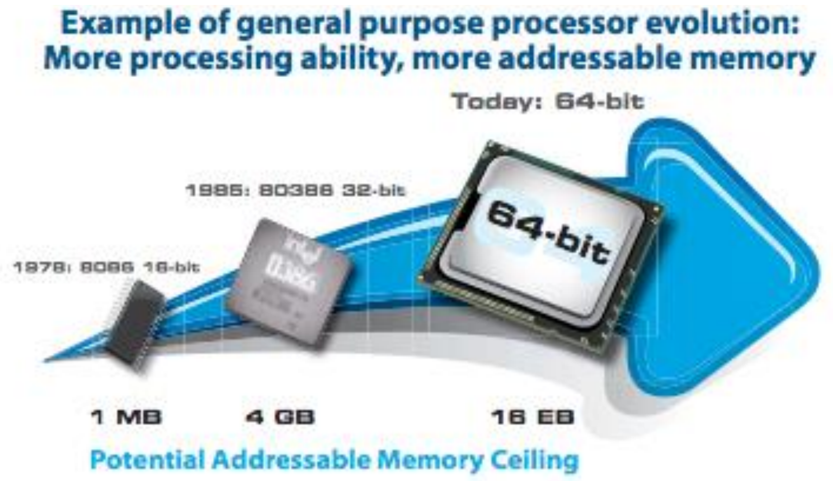
64-bit:

2^{64} = 18,446,744,073,709,551,616
18,446,744,073,709,551,616 / (1,024 x 1,024) = 16EB (exabytes)

Note that ... **giga** >> tera >> peta >> **exa**

32-bit and 64-bit





64-bit computers can realistically access 4 GB - 128 GB of RAM.

64-bit CPU

Example

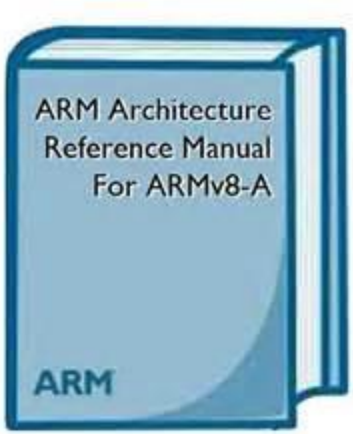
Apple: A11 Bionic CPU (iPhone-8, X)

https://en.wikipedia.org/wiki/Apple_A11#cite_note-iPhone_8_A11-4

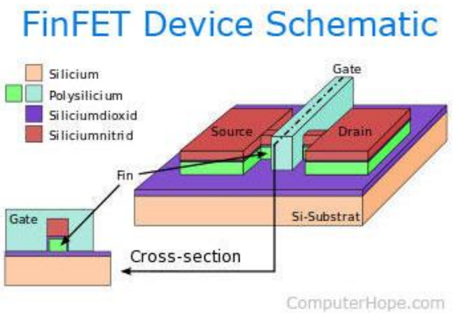
Apple: A11 Bionic CPU (iPhone-8, X)

Apple: A11 Bionic CPU (iPhone-8, X)

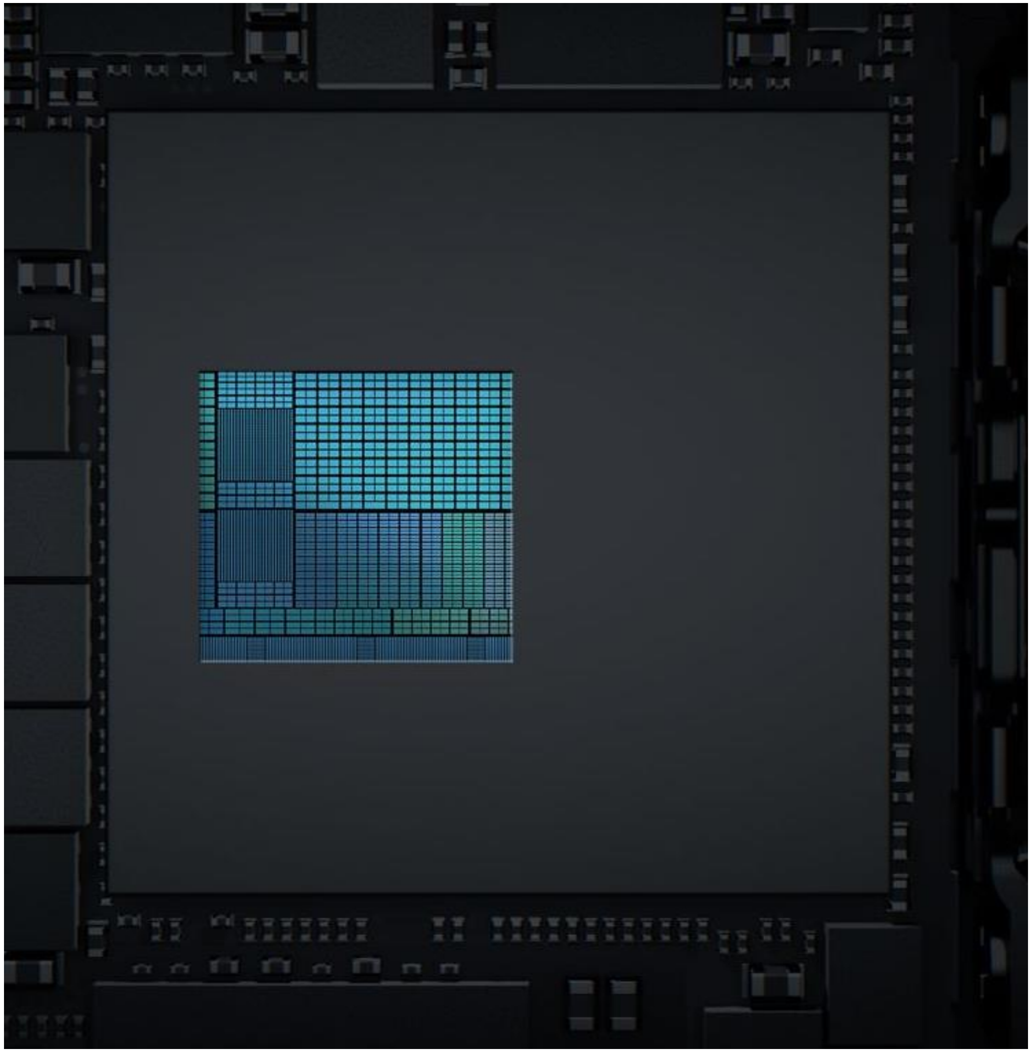
- The A-11 Bionic features an Apple-designed 64-bit ARMv8-A six-core CPU
  - Two high-performance cores: called Monsoon
  - Four energy-efficient cores: called Mistral
- The A11 is manufactured by TSMC using a 10 nm FinFET process



FinFET (Fin Field Effect Transistor)

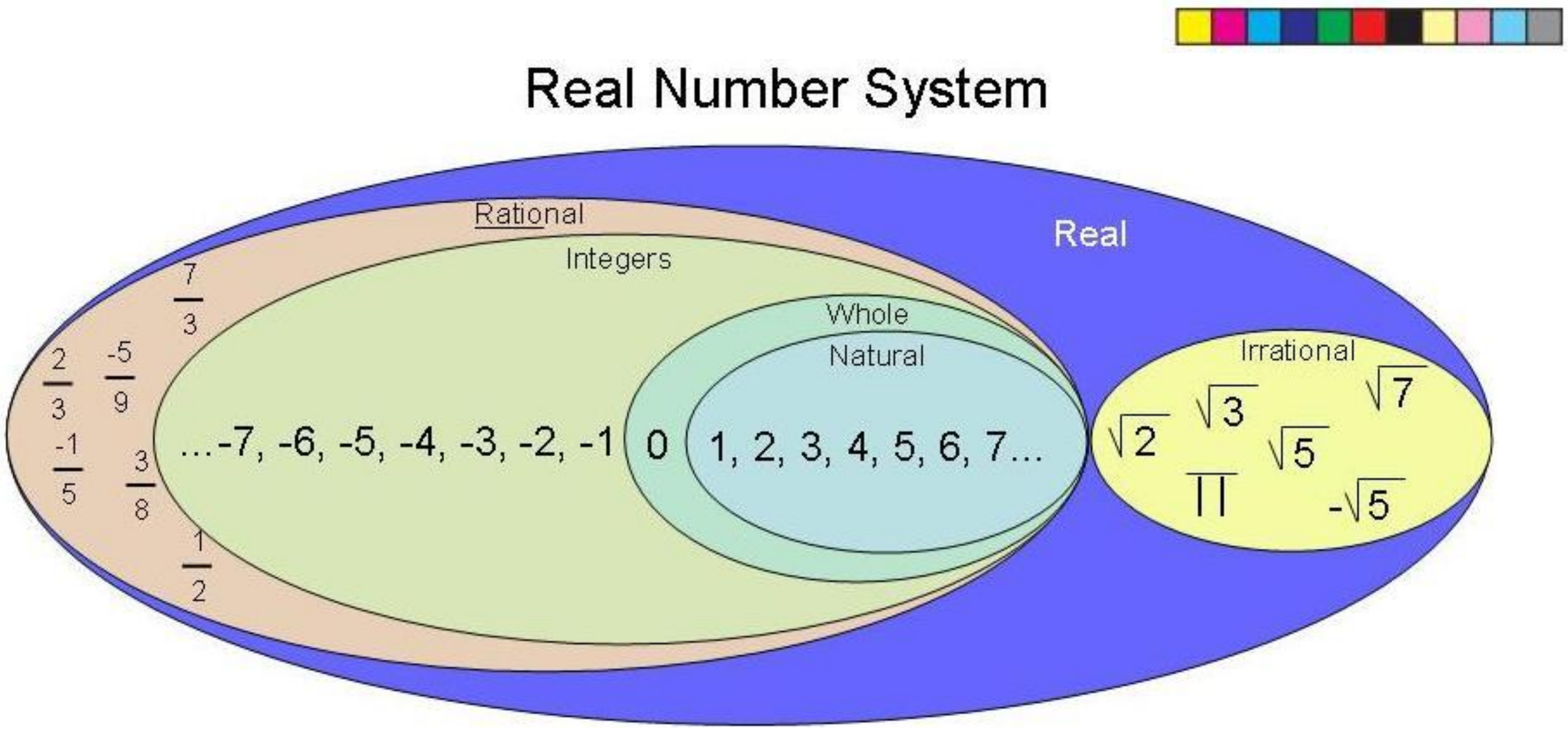


Apple: A11 Bionic: The neural engine



Real-Numbers

Real-Numbers



For Real-Numbers: Fixed-Point and **Floating-Point Number representations are used**

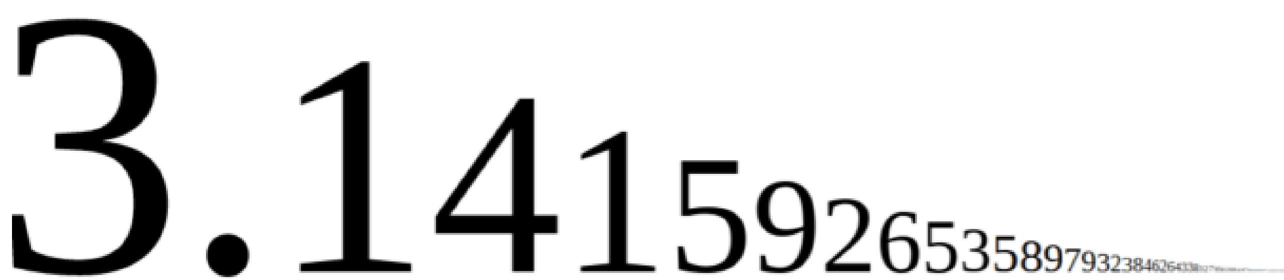
Why?

Why?

- Because we need to …
  - Expand the number range … and… and include smaller numbers than 1

    0.000000000000000000000000000012

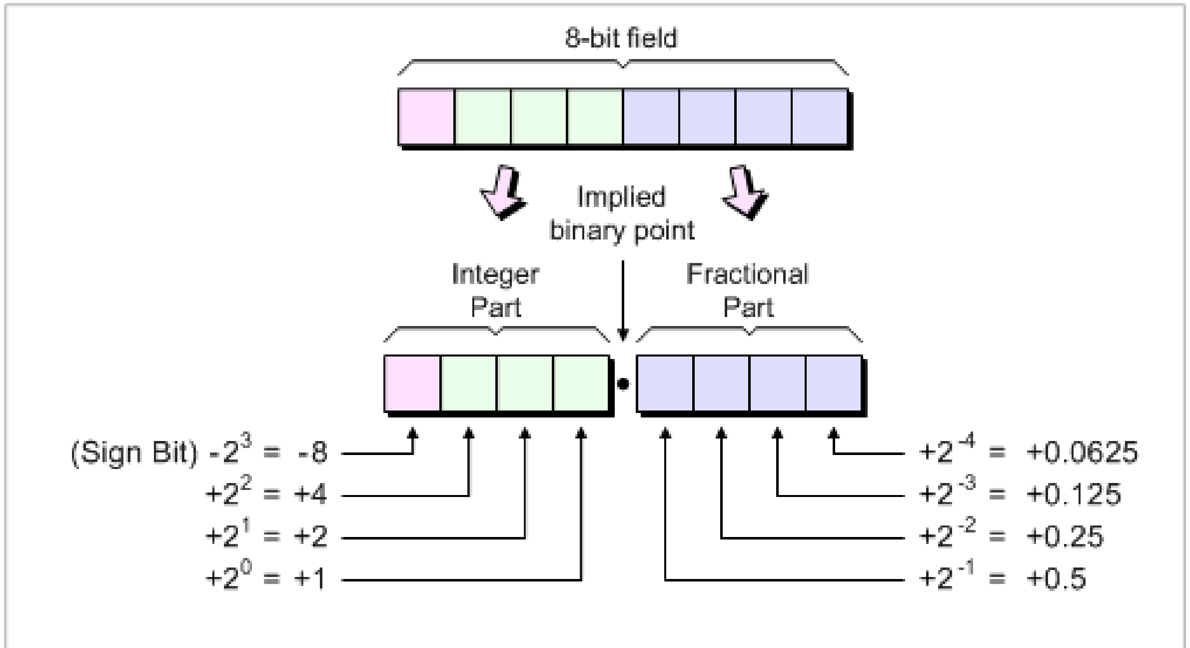
  - Use integers, positive and negative numbers in decimal notation.



2. Fixed-Point Numbers

Fixed-Point Numbers

Fixed point number representation: Every word has the same number of digits and **the binary point** is always **fixed** at the same position.



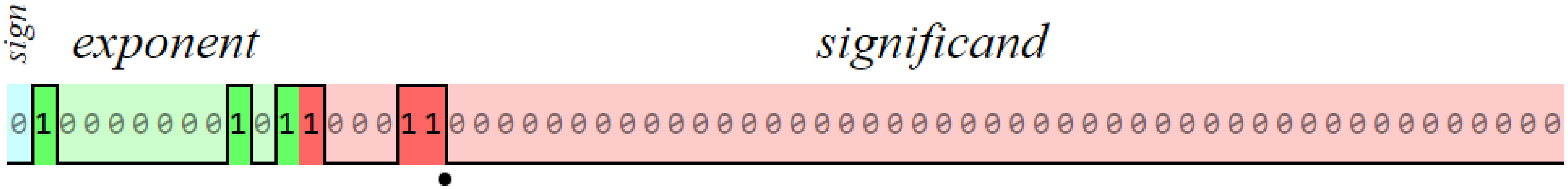
8-bit signed binary 4.4 fixed-point representation

http://www.diycalculator.com/sp-round.shtml

Fixed-Point Arithmetic is used …

- Fixed-Point arithmetic is used in applications where **speed** is more important than precision:
  - Digital Signal/Image Processing
  - Control Systems
  - Mobile smartDevices
  - Games
- Fixed-point calculations require less memory and less processor time
- Fixed-point hardware are much less complicated than those of **floating-point** hardware.

Floating-Point Numbers (FPN)



- FPN are: 32/64/128 bits long …

  - ○ Sign bit
  - ○ Exponent part
  - ○ Significant part

FPN (next lecture)