DSP Software and Number Systems

Dr. Clark Hochgraf

EEET-425

What Are The Main Points To Learn Today

- How a number is represented in a digital system can create “noise”
- Round off error is a type of noise.
- Round off error comes from the finite precision of floating point numbers
- Floating point numbers have a huge dynamic range (i.e. the ratio between the biggest number and the smallest number)
- Fixed point numbers have a limited dynamic range but allow for faster processing and cheaper chips.

Today's Topics

- Number Representations
  - Fixed Point
  - Floating Point
- Round off error
- Programming in Assembly versus C-language
- In Class Problem
- Making Software Run Faster
- Introduction to TI C6713 32-bit processor
- Summary

DSP Number Systems

- Numbers in a DSP are represented by bit patterns. The interpretation of the bit patterns is the critical part. The bits may represent fixed point numbers in one of four common formats
  - Unsigned Integer (can't represent negative numbers)
  - Two's complement (closest to actual hardware)
  - Sign and magnitude (easy for humans to understand)
  - Offset binary (used for ADC and DAC)

Fixed Point Number Representation

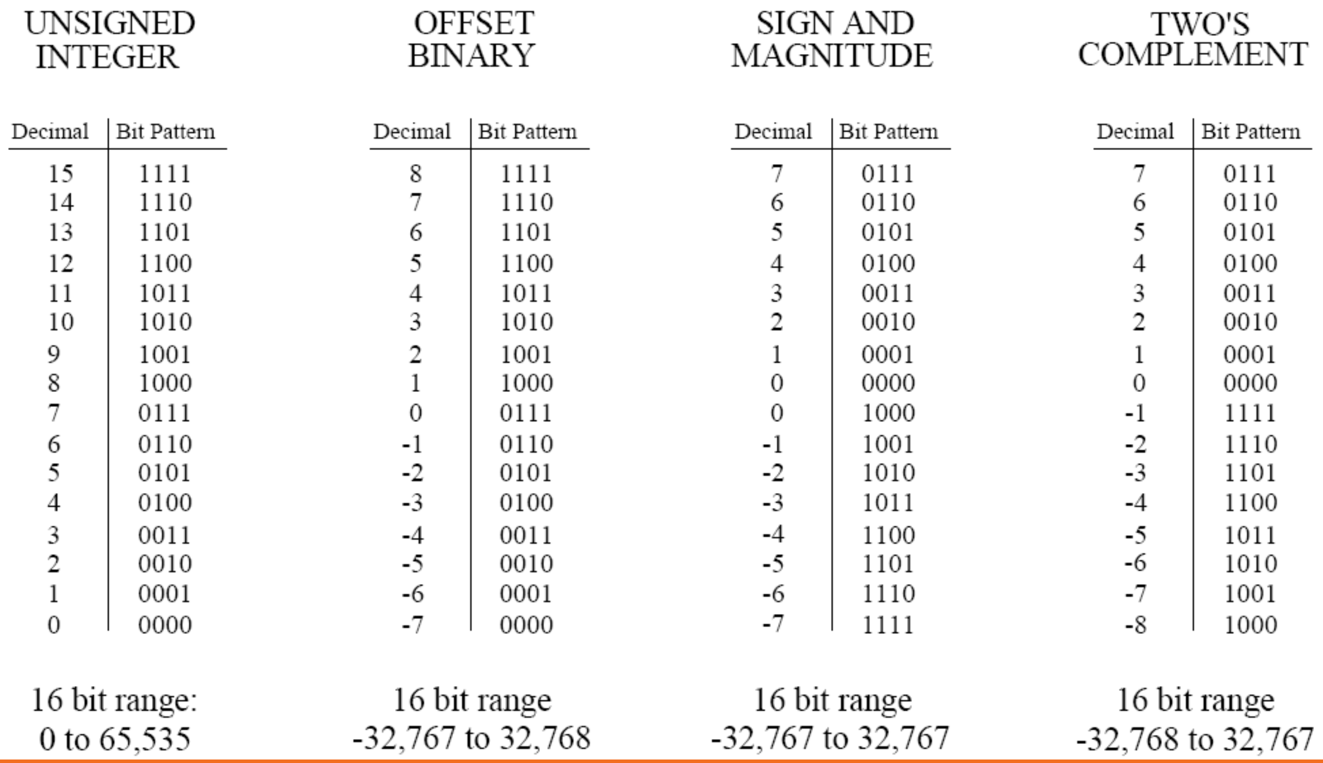
- Fixed point = integer representation
- E.g. represent 16 bit representation of a signed number -32768 to +32767
  - Or 65,536 possibilities

Fixed Point Formats

- Sign and Magnitude format (also called 1.15)
  - Normalize number to +/-1.0
    - 1 bit for sign
    - 15 bits for magnitude
  - 2^15 =32768, however largest positive number in 1.15 format is 0111111111111111 = 32767, therefore use 32767 for bit size calculations.
  - bit step size is 0.00003052 (=1/32,767)

Number Representation

- 1.15 is very similar to two complement in how it is represented in integer format, but different in how it is stored as a bit representation.



UNSIGNED INTEGER		OFFSET BINARY		SIGN AND MAGNITUDE		TWO'S COMPLEMENT	
Decimal	Bit Pattern	Decimal	Bit Pattern	Decimal	Bit Pattern	Decimal	Bit Pattern
15	1111	8	1111	7	0111	7	0111
14	1110	7	1110	6	0110	6	0110
13	1101	6	1101	5	0101	5	0101
12	1100	5	1100	4	0100	4	0100
11	1011	4	1011	3	0011	3	0011
10	1010	3	1010	2	0010	2	0010
9	1001	2	1001	1	0001	1	0001
8	1000	1	1000	0	0000	0	0000
7	0111	0	0111	0	1000	-1	1111
6	0110	-1	0110	-1	1001	-2	1110
5	0101	-2	0101	-2	1010	-3	1101
4	0100	-3	0100	-3	1011	-4	1100
3	0011	-4	0011	-4	1100	-5	1011
2	0010	-5	0010	-5	1101	-6	1010
1	0001	-6	0001	-6	1110	-7	1001
0	0000	-7	0000	-7	1111	-8	1000

16 bit range: 0 to 65,535

16 bit range -32,767 to 32,768

16 bit range -32,767 to 32,767

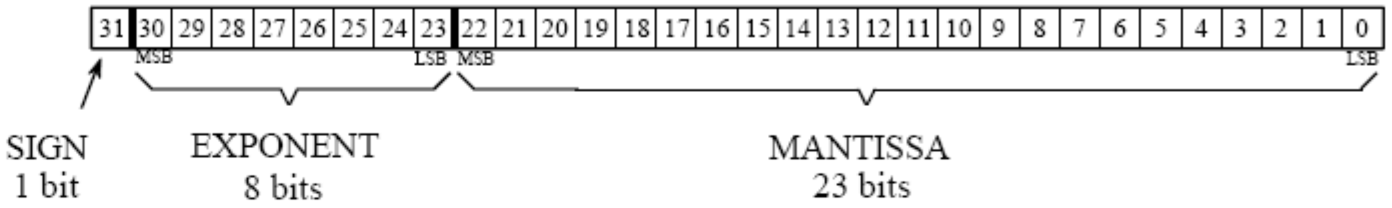
16 bit range -32,768 to 32,767

Floating Point

- Numbers can be stored as bit patterns representing floating point numbers.
- Floating point is similar to scientific notation
- Normalized to base 2 rather than base 10
- Described fully in ANSI/IEEE Standard 754-1985
- Defines how each of the bits in the word are interpreted e.g. sign, magnitude, exponent
- The advantage of floating point numbers is their wide dynamic range. Single precision floating point numbers can range from as large as +/- 3.4 x 10^38 to as small as +/- 1.2 x 10^-38.

Floating Point

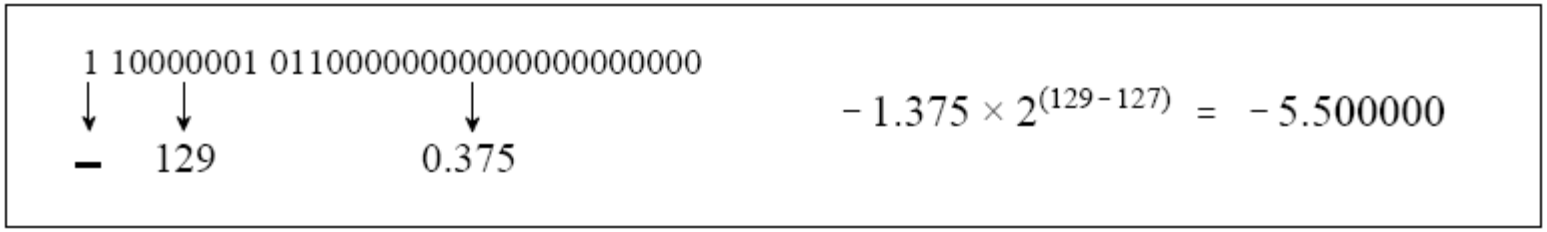
- 32bit = single precision (double prec. is 64 bits)
- 1 sign bit (S), 8 exponent bits (E), 23 mantissa bits (M)



$$v = (-1)^S \times M \times 2^{E-127}$$

$$M = 1 + m_{22}2^{-1} + m_{21}2^{-2} + m_{20}2^{-3}\cdots.$$

M as a decimal



$$-1.375 \times 2^{(129-127)} = -5.500000$$

Floating Point

- S=0 for positive
- E stored in offset binary. Subtract 127 to get the actual exponent of 2.
- M mantissa is a binary fraction.
  - Mantissa is normalized so that it has only one digit left of the binary point (similar to decimal point)

$$v = (-1)^S \times M \times 2^{E-127}$$

IEEE Standard Bit Patterns

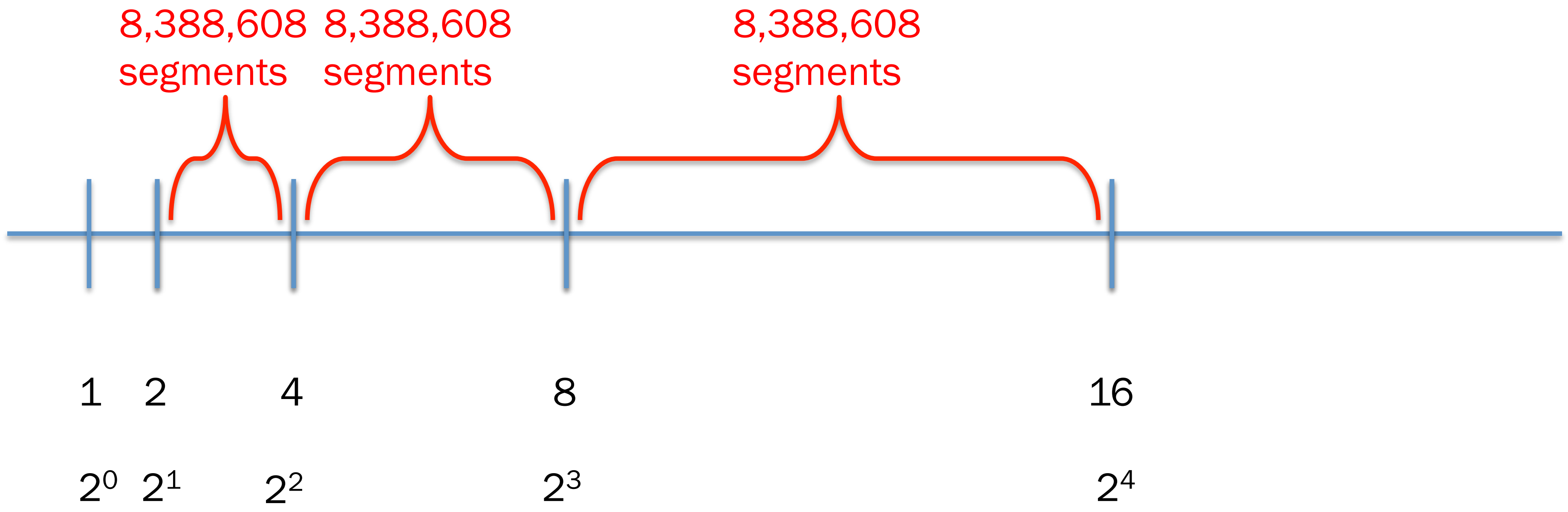
- Certain bit patterns in Floating Point are assigned special meaning.
- +/-0 assigned when all mantissa and all exponent bits are zero.
- +/- infinity when all mantissa bits are zero and all exponent bits are 1.
- NAN not a number for a small range of unnormalized small numbers

Number Precision

- In integer math, the spacing between numbers is always 1.
- In floating point math, the spacing between numbers varies over the number range.
  - Large numbers have large gaps number to number
  - Small numbers have small gaps number to number
- The spacing between two floating point numbers is about one 10 millionth of the number.

Floating Point Interval Between Numbers

Each interval between exponent values (e.g $2^{E-127} = 2^0, 2^1, 2^2, 2^3, 2^4$…) is divided into 2^{23} = 8,388,608 segments. The size of a given segment gets bigger as the exponent multiplier gets bigger

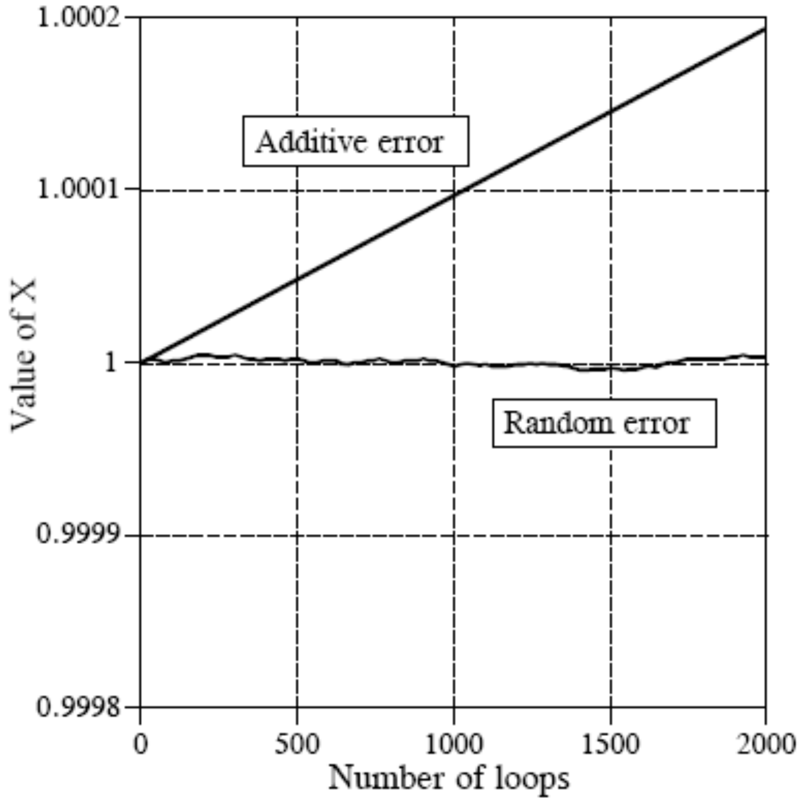


Round off Error Due to Finite Precision

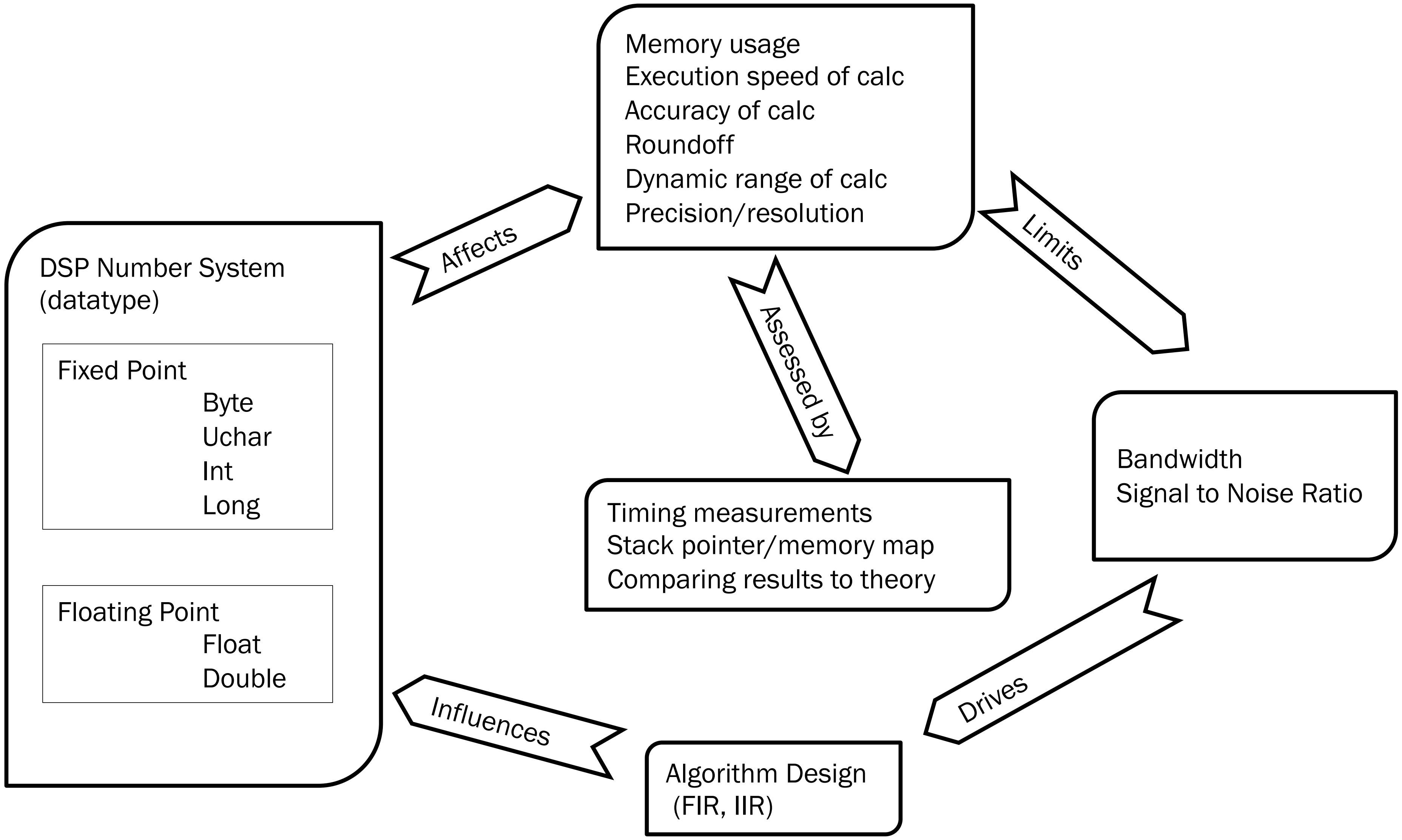
- Calculations with floating point numbers can accumulate round-off error. The errors associated with finite precision are very similar to quantization errors.
- Computer has to decide which of the limited number of bit patterns should represent the output of the calculation.
  - With 32 bits, only 4,294,967,296 possible bit patterns
- An approximate value of the round off error is one 40 millionth of the number.

Floating Point Round-off Error Accumulation

```
100  X = 1                    'initialize X
110  '
120  FOR I% = 0 TO 2000
130    A = RND                'load random numbers
140    B = RND                'into A and B
150    '
160    X = X + A              'add A and B to X
170    X = X + B
180    X = X - A              'undo the additions
190    X = X - B
200    '
210    PRINT X                'ideally, X should be 1
220 NEXT I%
230 END
```



Inter-play of Number System and Algorithm



Filter Design Vs Filter Realization

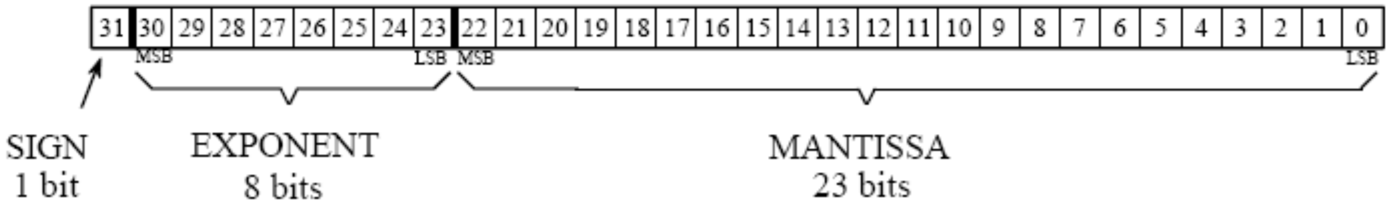
- A given digital filter design (e.g. fourth order butterworth lowpass filter) can be <u>realized</u> by many different implementations.
- Realization of a Digital filter relates to how it is implemented, not what its frequency response specification is.
- Filter realization is driven by execution speed, memory usage and sensitivity to round off error.

Dealing with Finite Precision

- Use proper typing for variables
  - Loop index should be integer not floating point (termination condition error – see textbook)
- Plan for error
  - Every single precision floating point number will have an error of one part in forty million multiplied by the number of math operations it has been through.
- Use knowledge of number system
  - Note that whole between +/- 16.8 million (2^24) have exact floating point binary representations. This allows floating point whole numbers staying within this range to be added, subtracted, multiplied without round off error.

Example

- Find the decimal number that corresponds to the following floating point bit pattern.
  - 1 01110001 01010100000000000000000

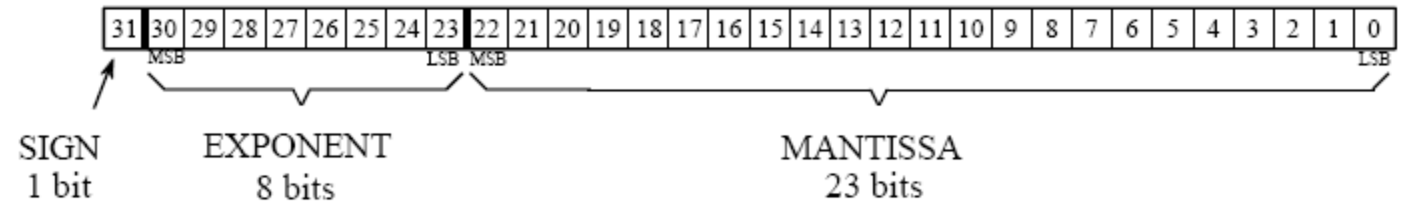


$$v = (-1)^S \times M \times 2^{E-127}$$

$$M = 1 + m_{22}2^{-1} + m_{21}2^{-2} + m_{20}2^{-3}\cdots.$$

In Class Problem - Solution

- 1 01110001 01010100000000000000000



$$v = (-1)^S \times M \times 2^{E-127}$$

$$M = 1.m_{22}m_{21}m_{20}m_{19}\cdots m_2m_1m_0$$

$$M = 1 + m_{22}2^{-1} + m_{21}2^{-2} + m_{20}2^{-3}\dots.$$

-1^1

$2^{(113-127)}$

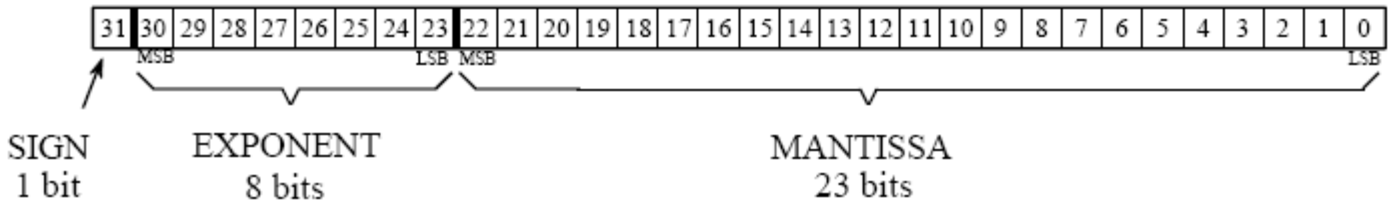
$1+1(2^{-2})+1(2^{-4})+1(2^{-6})$

or 1.328125

Putting it all together yields:
$-1.328125 \times 2^{-14} =$
-0.000081062

In Class Problem

- Find the decimal number that corresponds to the following floating point bit patterns.
- a. 1 01110001 01010100000000000000000
- b. 1 00000000 00000000000000000000000
- c. 0 11110011 11110010000000000000000
- d. 0 11111111 00000000000000000000000

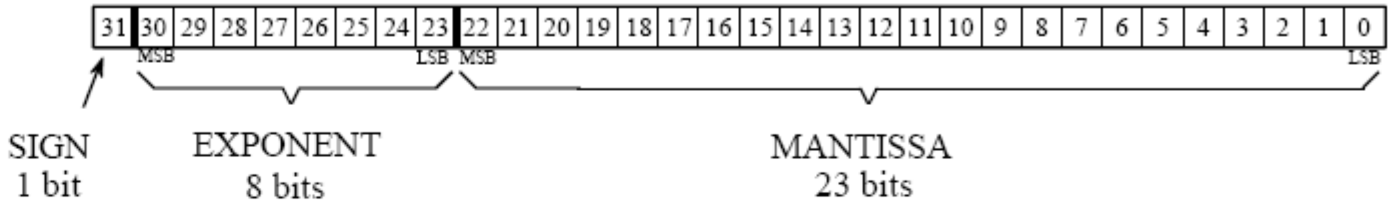


$$M = 1.m_{22}m_{21}m_{20}m_{19}\cdots m_2m_1m_0$$

$$v = (-1)^S \times M \times 2^{E-127}$$

In Class Problem

- Convert the following decimal numbers into their IEEE floating point bit patterns:
- a. 1
- b. 2
- c. 4
- d. -5

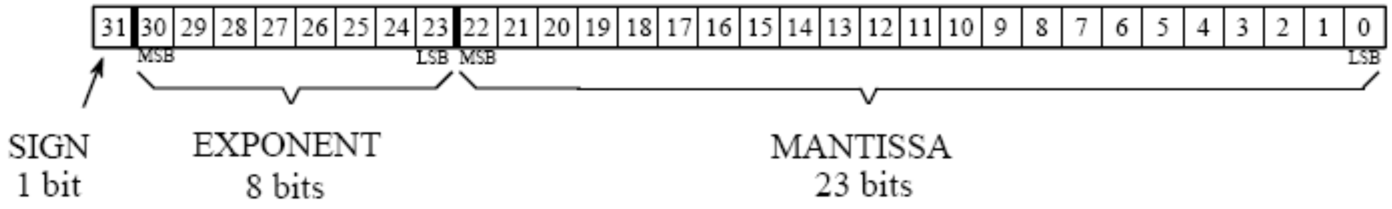


$$M = 1.m_{22}m_{21}m_{20}m_{19} \cdots m_2 m_1 m_0$$

$$v = (-1)^S \times M \times 2^{E-127}$$

In Class Problem - Solution

- Convert the following decimal numbers into their IEEE floating point bit patterns:



- a. 1
- b. 2
- c. 4
- d. -5

$$v = (-1)^S \times M \times 2^{E-127}$$

Tradeoff Between Round-off Error and Dynamic Range

- Floating point numbers have a wide but limited dynamic range. They also have finite precision.
- The dynamic range is determined by the number of exponent bits and the precision is set by the number of mantissa bits.

$$Dynamic\ Range = 6\ dB \times 2^{N_{exponent\ bits}}$$

$$SNR = 6\ dB \times N_{mantissa\ bits}$$

Example DR and SNR for 32 Bit IEEE Floating Point Numbers

- Nexponent bits =8+1 for the sign bit
- Nmantissa bits =23

$$Dynamic\ Range \ = \ 6\ dB \times \ 2^{N_{exponent\ bits}}$$

$$Dynamic\ Range \ = \ 6\ dB \times \ 2^9 = 3072\ dB$$

$$SNR \ = \ 6\ dB \times \ N_{mantissa\ bits}$$

$$SNR \ = \ 6\ dB \times \ 23 = 138\ dB$$

Programming in Assembly versus C-language

- Programming in C, abstracts you away from thinking of the number system but does not make the errors go away. E.g. precision, rounding error accumulation, indexing issues. Knowing these issues, you can write better algorithms in C.
- Programming in assembly keeps you closer and makes code faster, but often is not required.
  - Assembly is closer to the hardware and generally not portable from one family of processors to another.
- C compilers are fairly good at making efficient code, but the human programmer must understand the number system limitations.

Example Round-off Error Computation

- In an FIR digital filter, each sample in the output signal is found by multiplying M samples from the input signal by M predetermined coefficients, and adding the products. The characteristics of these filters (high-pass, low-pass, etc.) are determined by the coefficients used. For this problem, assume M = 5000, and that single precision floating point math is used.

- a. How many math operations (the number of multiplications plus the number of additions) need to be conducted to calculate each point in the output signal?
- b. If the output signal has an average amplitude of about one-hundred, what is the expected error on an individual output sample. Assume that the round-off errors combine by addition. Give your answer both as an absolute number, and as a fraction of the number being represented.

Example Round-off Error Computation - Solution

- a. How many math operations (the number of multiplications plus the number of additions) need to be conducted to calculate each point in the output signal?

- b. If the output signal has an average amplitude of about one-hundred, what is the expected error on an individual output sample. Assume that the round-off errors combine by addition. Give your answer both as an absolute number, and as a fraction of the number being represented.

Why All the Attention to Number Formats?

- In the lab, you will be working with a variety of number formats, e.g. short, integer, float, double in your C-code.
  - It is very common to get confused if you try to plot a number as a float when it is a 32 bit integer. The resulting plot looks like noise. It is important to know exactly how you, and how the computer, are interpreting the bit pattern. Pay particular attention when you are plotting variables in the DSP memory.

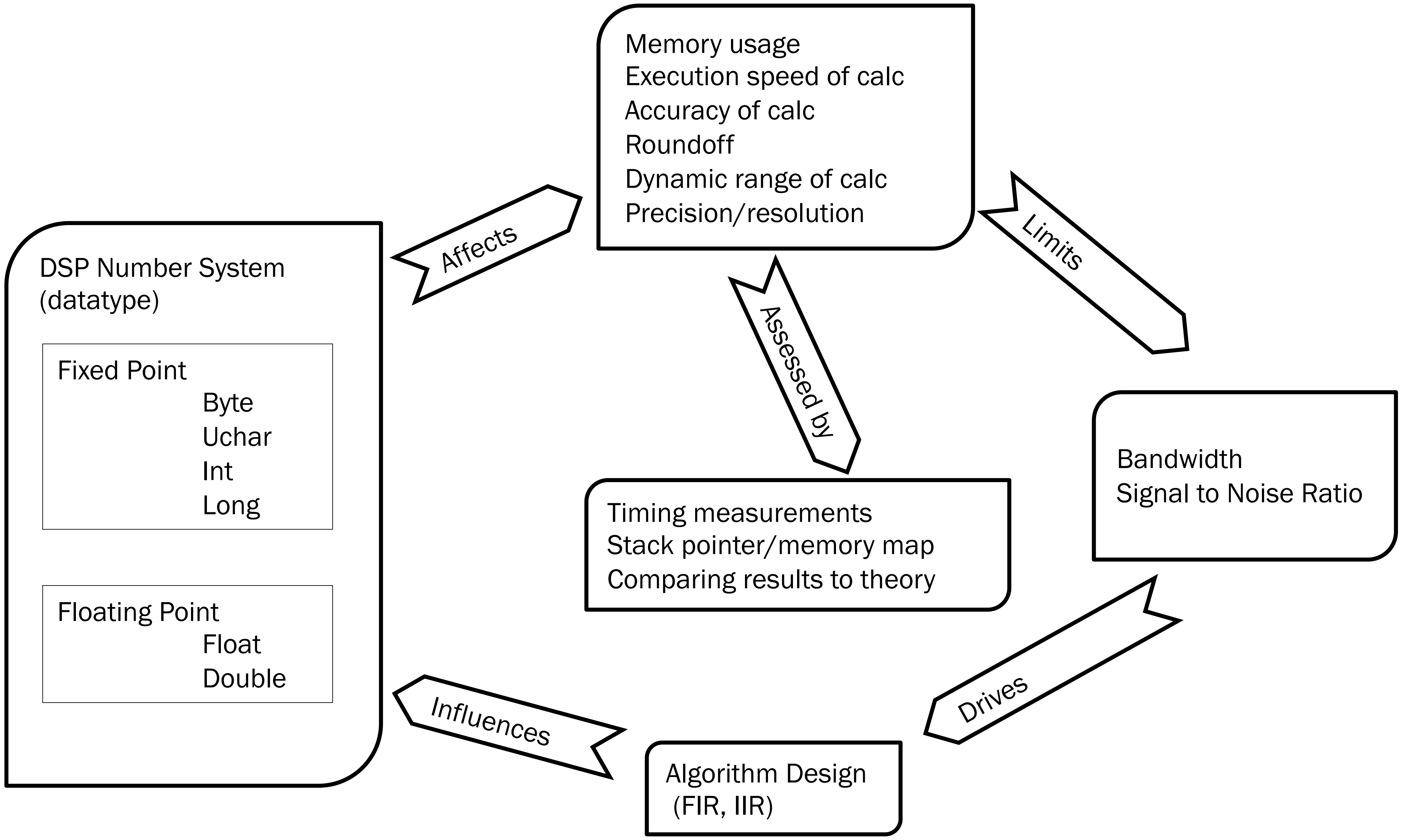
Summary of Today's Lecture (1/2)

- Numbers in a DSP are represented by bit patterns. The interpretation of the bit patterns is the critical part. The bits may represent fixed point numbers in one of four common formats
  - Unsigned Integer (can't represent negative numbers)
  - Two's complement (closest to actual hardware)
  - Sign and magnitude (easy for humans to understand)
  - Offset binary (used for ADC and DAC)

Summary of Today's Lecture (2/2)

- Bit patterns may also represent floating point numbers. The advantage of floating point numbers is their wide dynamic range. Single precision floating point numbers can range from as large as +/- 3.4 x 10^38 to as small as +/- 1.2 x 10^-38.
- The spacing between two floating point numbers is about one 10 millionth of the number.
- Calculations with floating point numbers can accumulate round-off error. An approximate value of the round off error is one 40 millionth of the number.

Inter-play of Number System and Algorithm


DSP Number System
(datatype)
Fixed Point
Byte
Uchar
Int
Long
Floating Point
Float
Double
Affects
Memory usage
Execution speed of calc
Accuracy of calc
Roundoff
Dynamic range of calc
Precision/resolution
Limits
Assessed by
Bandwidth
Signal to Noise Ratio
Timing measurements
Stack pointer/memory map
Comparing results to theory
Drives
Algorithm Design
(FIR, IIR)
Influences