10.2 1s Complement (2/3)

- Technique to negate a value: invert all the bits.
- Largest value: 01111111 = $+127_{10}$
- Smallest value: 10000000 = -127_{10}
- Zeros: 00000000 = $+0_{10}$
  11111111 = -0_{10}
- Range (for 8 bits): -127_{10} to $+127_{10}$
- Range (for *n* bits): $-(2^{n-1} - 1)$ to $2^{n-1} - 1$
- The most significant bit (MSB) still represents the sign: 0 for positive, 1 for negative.

10.3 2s Complement (2/3)

- Technique to negate a value: invert all the bits, then add 1.
- Largest value: 01111111 = $+127_{10}$
- Smallest value: 10000000 = -128_{10}
- Zero: 00000000 = $+0_{10}$
- Range (for 8 bits): -128_{10} to $+127_{10}$
- Range (for *n* bits): -2^{n-1} to $2^{n-1} - 1$
- The most significant bit (MSB) still represents the sign: 0 for positive, 1 for negative.

10.8 Excess Representation (1/2)

- Besides sign-and-magnitude and complement schemes, the **excess representation** is another scheme.
- It allows the range of values to be distributed evenly between the positive and negative values, by a simple translation (addition/subtraction).
- Example: Excess-4 representation on 3-bit numbers. See table on the right.

Excess-4 Representation	*Value*
000	-4
001	-3
010	-2
011	-1
100	0
101	1
110	2
111	3

11.2 IEEE 754 Floating-Point Rep. (3/4)

- 3 components: **sign**, **exponent** and **mantissa (fraction)**

sign	exponent	mantissa

- Sign bit: 0 for positive, 1 for negative.
- Mantissa is **normalised** with an implicit leading bit 1
  - 110.1_2 → normalised → $1.101_2 \times 2^2$ → only **101** is stored in the mantissa field
  - 0.00101101_2 → normalised → $1.01101_2 \times 2^{-3}$ → only **01101** is stored in the mantissa field

11.2 IEEE 754 Floating-Point Rep. (2/4)

- 3 components: **sign**, **exponent** and **mantissa (fraction)**

sign	exponent	mantissa

- The base (radix) is assumed to be 2.
- Two formats:
  - Single-precision (32 bits): 1-bit sign, 8-bit exponent with bias 127 (excess-127), 23-bit mantissa
  - Double-precision (64 bits): 1-bit sign, 11-bit exponent with bias 1023 (excess-1023), and 52-bit mantissa
- We will focus on the single-precision format
- Reading
  - DLD pages 32 - 33
  - IEEE standard 754 floating point numbers: http://steve.hollasch.net/cgindex/coding/ieeefloat.html

11.2 IEEE 754 Floating-Point Rep. (3/4)

- 3 components: **sign**, **exponent** and **mantissa (fraction)**

sign	exponent	mantissa

- Sign bit: 0 for positive, 1 for negative.
- Mantissa is **normalised** with an implicit leading bit 1
  - 110.1_2 → normalised → $1.101_2 \times 2^2$ → only **101** is stored in the mantissa field
  - 0.00101101_2 → normalised → $1.01101_2 \times 2^{-3}$ → only **01101** is stored in the mantissa field

Function Prototype

- It is a good practice to put function prototypes at the top of the program, before the main() function, to inform the compiler of the functions that your program may use and their return types and parameter types.
- **A function prototype includes only the function's return type, the function's name, and the data types of the parameters** (names of parameters are optional).
- Function definitions to follow after the main() function.
- Without function prototypes, you will get error/warning messages from the compiler.