ECE2050 Homework 1

Due: Feb. 20, 2025

Q1 What is the largest 14-bit binary number that can be represented with.

1. unsigned numbers?
2. two's complement numbers?
3. sign/magnitude numbers?

A1:

1.Unsigned: 16383

2.Two's Complement: 8191

3.Sign/Magnitude: 8191

Q2 What is the smallest (most negative) 15-bit binary number that can be represented with

1. unsigned numbers?
2. two's complement numbers?
3. sign/magnitude numbers?

A2:

1.Unsigned: 0

2.Two's Complement: -16384

3.Sign/Magnitude: -16383

Q3 Convert the following hexadecimal numbers to unsigned binary. Show your work.

1. $A6_{16}$;
2. $3C_{16}$;
3. $FFFC_{16}$;
4. $E0000000_{16}$.

A3:

1. $A6_{16} \rightarrow 10100110_2$
2. $3C_{16} \rightarrow 0011\ 1100_2$
3. $FFFC_{16} \rightarrow 1111\ 1111\ 1111\ 1100_2$
4. $E0000000_{16} \rightarrow 1110\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_2$

Q4 Convert the following two's complement binary numbers to decimal.

1. 1011_2;
2. 110111_2;
3. 01110010_2;
4. 10001111_2.

A4:

1.
Starts with 1 (negative).
Invert: 0100_2
Add 1: $0101_2 = 5$
Answer: -5_{10}

2.
Starts with 1 (negative).
Invert: 001000_2
Add 1: $001001_2 = 9$

Answer: -9_{10}

3.
Starts with 0 (positive).
Convert normally:
0+64+32+16+0+0+2+0=114
Answer: 114_{10}

4.
Starts with 1 (negative).
Invert: 01110000_2
Add 1: $01110001_2 = 113$
Answer: -113_{10}

Q5 Convert the following decimal numbers to 8-bit two's complement numbers or indicate that the decimal number would overflow the range.

1. -51_{10};
2. -67_{10};
3. -151_{10};
4. -128_{10}.

A5:

1.-51_{10}

$|51| = 51 \rightarrow 00110011_2$

Invert: 11001100_2

Add 1: 11001101_2

Answer: 11001101_2

2.-67_{10}

$|67| = 67 \rightarrow 01000011_2$

Invert: 10111100_2

Add 1: 10111101_2

Answer: 10111101_2

3.-151_{10}

The range for 8-bit two's complement is −128 to 127, but −151 is out of range.
Overflow!

4.-128_{10}

Special case: −128 is the lowest possible value in 8-bit two's complement.

Answer: 10000000_2

Q6 Convert the following 4-bit 2's complement numbers to 8-bit 2's complement numbers.

1. 1100_2;
2. 1011_2.

A6:

1.1100_2 (4-bit)
Starts with 1 (negative).
Extend the sign bit (1) to make it 8-bit:
$1111\ 1100_2$
Answer: 11111100_2

2.1011_2 (4-bit)
Starts with 1 (negative).
Extend the sign bit (1) to make it 8-bit:
$1111\ 1011_2$
Answer: 11111011_2

Q7 Perform each addition in the 2's complement form. Indicate whether the sum overflows a 7-bit result.

1. $18_{10} + 9_{10}$;
2. $29_{10} + 32_{10}$;
3. $-6_{10} + 23_{10}$;
4. $-3_{10} + 31_{10}$;
5. $-18_{10} + -10_{10}$;
6. $-28_{10} + -36_{10}$.

A7:

1. $18_{10} + 9_{10}=0011011_2$ (No Overflow)
2. $29_{10} + 32_{10}=0111101_2$ (No Overflow)
3. $-6_{10} + 23_{10}=0010001_2$ (No Overflow)
4. $-3_{10} + 31_{10}=0011100_2$ (No Overflow)
5. $-18_{10} + -10_{10}=1100100_2$ (No Overflow)
6. $-28_{10} + -36_{10}=1000000_2$ (No Overflow)

Set question 3 as an example:

3. $-6_{10} + 23_{10}$

- $-6_{10} = 1111010_2$
- $23_{10} = 0010111_2$
- Addition:

```
  1111010
+ 0010111
--------
  0001001
```

- Result: $0001001_2 = 17_{10}$
- Overflow: No (carry into MSB = carry out of MSB)

Q8 The IEEE 754 standard specifies a 32-bit binary as having: 1 bit Sign; 8 bits Exponent width; 24 bits (23 explicitly stored) Mantissa precision. The sign bit determines the sign of the number, which is the sign of the mantissa as well. The exponent is an 8-bit unsigned integer from 0 to 255, in biased form: an exponent value of 127 represents the actual zero. Exponents range from -126 to +127 because exponents of -127 (all 0s) and +128 (all 1s) are reserved for special numbers. The true mantissa includes 23 fraction bits to the right of the binary point and an implicit leading bit (to the left of the binary point) with value 1, unless the exponent is stored with all zeros. Thus only 23 fraction bits of the mantissa appear in the memory format, but the total precision is 24 bits. The bits are laid out as shown in Figure 1: The real value assumed by a given 32-bit binary data with a given sign, biased exponent (the 8-bit unsigned integer), and a 23-bit mantissa is

$$(-1)^{b_{31}} \times 2^{(b_{30}b_{29}\cdots b_{23})_2 - 127} \times (1.b_{22}b_{21}\cdots b_0)_2.$$

1. Convert the decimal number 102.25_{10} to IEEE 754 standard 32-bit binary. Show your work;
2. The IEEE 754 standard specifies a 64-bit binary as having: 1 bit Sign; 11 bits Exponent width; 53 bits (52 explicitly stored) Mantissa precision. The sign bit determines the sign of the number. The exponent is an 11-bit unsigned integer from 0 to 2047, in biased form: an exponent value of 1023 represents the actual zero. Exponents range from -1022 to +1023 because exponents of -1023 (all 0s) and +1024 (all 1s) are reserved for special numbers. The true mantissa includes 52 fraction bits to the right of the binary point and an implicit leading bit (to the left of the binary point) with value 1, unless the exponent is stored with all zeros. Thus only 52 fraction bits of the mantissa appear in the memory format, but the total precision is 53 bits. Sketch IEEE 754 standard 64-bit binary layout;
3. Convert the decimal number 102.25_{10} to IEEE 754 standard 64-bit binary. Show your work.

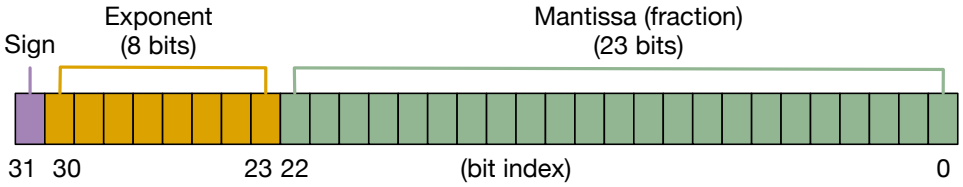


Figure 1: IEEE 754 standard 32-bit binary.

1. Convert the Decimal Number 102.25_{10} to IEEE 754 32-bit Binary Format

Steps:

1. **Sign Bit**: Since the number is positive, the sign bit is 0.
2. **Decimal to Binary Conversion**:
   - Integer part: $102_{10} = 1100110_2$.
   - Fractional part: $0.25_{10} = 0.01_2$.
   - Combined: $102.25_{10} = 1100110.01_2$.
3. **Normalization**: Convert the binary into scientific notation:
   - $1100110.01_2 = 1.10011001 \times 2^6$ (Shift the decimal point left by 6 places).
4. **Exponent Calculation**:
   - Actual exponent $E = 6$, and the bias for 32-bit IEEE 754 is 127.
   - Biased exponent $E_{\text{bias}} = 6 + 127 = 133_{10}$.
   - Binary of 133_{10} = 10000101_2.
5. **Mantissa**:
   - Remove the leading 1 from the normalized value, leaving the mantissa: $10011001000000000000000_2$.
   - Padded to 23 bits to fit in the mantissa field.

Final Result (32-bit):

$$\boxed{0\ 10000101\ 10011001000000000000000}$$

2. IEEE 754 64-bit Binary Layout

64-bit Double Precision Format:

- **Sign Bit**: 1 bit (the most significant bit).
- **Exponent**: 11 bits (after the sign bit), with a bias of 1023.
- **Mantissa**: 52 bits (the remaining bits), with an implicit leading 1.

Layout Diagram:

$$\boxed{\underbrace{\text{Sign (1 bit)}}_{\text{Sign Bit}}\ \underbrace{\text{Exponent (11 bits)}}_{\text{Exponent}}\ \underbrace{\text{Mantissa (52 bits)}}_{\text{Mantissa}}}$$

3. Convert the Decimal Number 102.25_{10} to IEEE 754 64-bit Binary Format

Steps:

1. **Sign Bit**: Since the number is positive, the sign bit is 0.
2. **Decimal to Binary Conversion**:
   - The binary conversion is the same as for the 32-bit format: $102.25_{10} = 1100110.01_2$.
3. **Normalization**: Convert to scientific notation:
   - $1100110.01_2 = 1.10011001 \times 2^6$.
4. **Exponent Calculation**:
   - The actual exponent $E = 6$, and the bias for 64-bit IEEE 754 format is 1023.
   - The biased exponent is $E_{\text{bias}} = 6 + 1023 = 1029_{10}$.
   - In binary, $1029_{10} = 10000000101_2$ (11 bits).
5. **Mantissa**:
   - Remove the leading 1 from the normalized form, leaving the mantissa: $1001100100000000000000000000000000000000000000000000_2$ (padded to 52 bits).

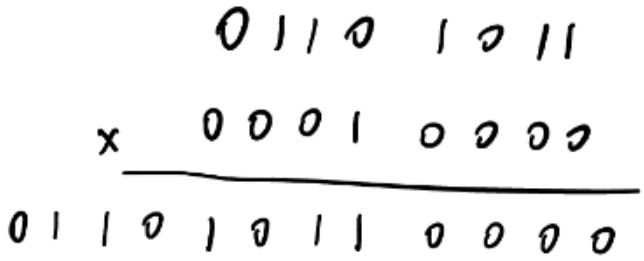
Final Result (64-bit):

$$\boxed{0\ 10000000101\ 1001100100000000000000000000000000000000000000000000}$$

Q9 Multiply 01101011 by 11110000 in the two's complement form.

A9:

1.Sign bit=1
2.2's complement of 11110000 = 00010000
3.Multiplication



4. According to the sign of two numbers, the result is negative. So right answer is -011010110000, then converter it into new format. The right answer is:

100101010000

Q10 Convert each of the following octal numbers to binary, hexadecimal, and decimal.

1. 23_8;
2. 45_8;
3. 371_8;
4. 2560_8.

A10:

23 (Octal) = Binary: 010011, Hex: 13, Decimal: 19
45 (Octal) = Binary: 100101, Hex: 25, Decimal: 37
371 (Octal) = Binary: 011111001, Hex: F9, Decimal: 249 2560
(Octal) = Binary: 010101110000, Hex: 570, Decimal: 1392

Q11 How many 7-bit two's complement numbers are greater than 0? How many are less than 0? How would your answers differ for sign/magnitude numbers?

A11:

Two's Complement:
Numbers greater than 0: 63.
Numbers less than 0: 64.

Sign/Magnitude:
Numbers greater than 0: 63.
Numbers less than 0: 63.

The answers differ because two's complement has an asymmetric range (one extra negative number), while sign/magnitude has a symmetric range but a redundant representation for *0*