Lab2_Part2 & Chapter3 Review

Yuanqing Miao

https://github.com/myuanqing/CMPEN331_Spring2025

Lab2_part2 Hints

UTF-8 Encoding Rules:

Bytes	Bit Pattern	Unicode Range
1 byte	`0xxxxxxx`	U+0000 to U+007F
2 bytes	`110xxxxx 10xxxxxx`	U+0080 to U+07FF
3 bytes	`1110xxxx 10xxxxxx 10xxxxxx`	U+0800 to U+FFFF
4 bytes	`11110xxx 10xxxxxx 10xxxxxx 10xxxxxx`	U+10000 to U+10FFFF

```
Smaller than Bound1
JMP HANDLE_1BYTE

Smaller than Bound2
JMP HANDLE_2BYTE

Smaller than Bound3
JMP HANDLE_3BYTE
......

JMP HANDLE_ERROR
```

Example: Encoding U+0800 (0x0800) to UTF-8

Step 1: Determine Encoding Size

- \- U+0800 falls in the range `0x0800 - 0xFFFF` → 3-byte encoding

Step 2: Binary Representation of Code Point

- \- U+0800 (hex) = `0000 1000 0000 0000` (binary)

Step 3: UTF-8 Encoding (3 Bytes)

First byte: `1110xxxx`

- Extract top 4 bits: `0000` → `11100000` (`0xE0`)

Second byte: `10xxxxxx`

- Next 6 bits: `100000` → `10100000` (`0xA0`)

Third byte: `10xxxxxx`

- Last 6 bits: `000000` → `10000000` (`0x80`)

Final UTF-8 Encoding:

\- Binary: `11100000 10100000 10000000`

\- Hexadecimal: `0xE0 A0 80`

1. Assume 185 and 122 are signed 8-bit decimal integers stored in sign-magnitude format. Calculate 185 + 122. Is there overflow, underflow, or neither?

1. Understanding Sign-Magnitude Format

In **8-bit sign-magnitude format**, an integer is represented as:

- **1 bit for the sign** (Most Significant Bit, MSB)
  - **0** indicates a **positive** number.
  - **1** indicates a **negative** number.
- **7 bits for the magnitude**, allowing values from **0 to 127**.

2. Range of Representable Values

In **8-bit sign-magnitude format**, the representable range is:

- **Positive Range:** $+0$ to $+127$
- **Negative Range:** -0 to -127

1. Interpretation of 185 in Sign-Magnitude Format

Sign-Magnitude Decoding:

- **Binary of 185: 10111001**
- **Sign Bit: 1**, indicating a **negative** number.
- **Magnitude: 57**, derived from the binary **0111001**.

Final Interpretation:

$$185 \text{ in sign-magnitude format} = -57$$

---

2. Interpretation of 122 in Sign-Magnitude Format

- **Binary of 122: 01111010**
- **Sign Bit: 0**, indicating a **positive** number.
- **Magnitude: 122**, directly from the binary representation.

Final Interpretation:

$$122 \text{ in sign-magnitude format} = +122$$

---

3. Perform the Addition

$$-57 + 122 = 65$$

Neither Overflow nor Underflow

2. Assume 185 and 122 are signed 8-bit decimal integers stored in sign-magnitude format. Calculate 185 - 122. Is there overflow, underflow, or neither?

Mathematical Operation:

$$-57 - 122 = -179$$

Result Validation:

- The result **-179** is **outside** the representable range for **8-bit sign-magnitude integers**, as it is **less than -127**.

Conclusion:

$$\text{The operation results in } \textbf{underflow}.$$

4. Assuming single precision IEEE 754 format, what decimal number is represent by this word:

   1 01111101 00100000000000000000000

The **IEEE 754 single-precision format** consists of:

- **1 bit** for the **sign.**
- **8 bits** for the **exponent.**
- **23 bits** for the **mantissa** (fraction).

1. Determine the Sign Bit

- **Sign Bit: 1**, indicating the number is **negative.**

$$\text{Sign} = (-1)^1 = -1$$

2. Decode the Exponent

- Exponent Bits: 01111101

Convert to Decimal:

$$01111101_2 = 125_{10}$$

Apply the Bias:

- **IEEE 754** single-precision uses a **bias of 127**:

$$\text{Exponent} = 125 - 127 = -2$$

3. Decode the Mantissa (Fraction)

- Mantissa Bits: 00100000000000000000000

Constructing the Fraction:

The **mantissa** has an **implicit leading 1**, so:

$$1.00100000000000000000000_2$$

Convert to Decimal:

$$1 + 2^{-3} = 1 + 0.125 = 1.125$$

4. Calculate the Final Value

$$\text{Value} = \text{Sign} \times \text{Mantissa} \times 2^{\text{Exponent}}$$

$$\text{Value} = -1 \times 1.125 \times 2^{-2}$$

$$2^{-2} = \frac{1}{4} = 0.25$$

$$\text{Value} = -1.125 \times 0.25 = -0.28125$$

5. The floating-point format to be used in this problem is an 8-bit IEEE 754 normalized format with 1 sign bit, 4 exponent bits, and 3 mantissa bits. It is identical to the 32-bit and 64-bit formats in terms of the meaning of fields and special encodings. The exponent field employs an excess- 7coding. The bit fields in a number are (sign, exponent, mantissa). Assume that we use unbiased rounding to the nearest even specified in the IEEE floating point standard.

Field	Bits	Description
Sign	1	**0** for positive, **1** for negative
Exponent	4	**Excess-7** encoding (bias = 7)
Mantissa	3	**Normalized form** with an **implicit leading 1**

a) Encode the following numbers to the 8 bit IEEE format

1) $0.0011011)_{binary}$

Normalize the Binary Number

$$0.0011011 = 1.1011 \times 2^{-3}$$

Components of the Normalized Form:

- **Sign Bit: 0** (since the number is positive)
- **Mantissa:** The **3 bits** after the leading **1**:
  - **1.1011 → 101**
- **Exponent: -3** (due to shifting the binary point **3** places to the right)

Calculate the Biased Exponent:

$$\text{Biased Exponent} = -3 + 7 = 4$$

Convert 4 to 4-bit Binary:

$$4 = 0100_2$$

Final 8-bit Representation:

$$0\ 0100\ 101$$

i) Decode the following 8-bit IEEE number into their decimal value: 1 1010 101

- **Binary Exponent: 1010**
- **Convert to Decimal:**

$$1010_2 = 10_{10}$$

- **Apply Excess-7 Bias:**

$$\text{Exponent} = 10 - 7 = 3$$

- The **IEEE 754 format** uses an **implicit leading 1**, giving us:

$$1.101$$

Convert to Decimal:

$$1 + 0.5 + 0.125 = 1.625$$

$$\text{Value} = \text{Sign} \times \text{Mantissa} \times 2^{\text{Exponent}}$$

$$\text{Value} = -1 \times 1.625 \times 2^3$$

$$2^3 = 8$$

$$\text{Value} = -1.625 \times 8 = -13.0$$

j) Decide which number in the following pairs are greater in value (the numbers are in 8-bit IEEE 754 format):

1) 0 1000 100 and 0 1000 111

The second number is greater in value

b) Perform the computation 1.011binary + 0.0011011binary showing the correct state of the guard, round bits and sticky bits. There are three mantissa bits.

1. What are Guard, Round, and Sticky Bits?

Guard Bit (G):

- The **first bit** beyond the **mantissa bits**.
- Helps in **rounding decisions**.

Round Bit (R):

- The **second bit** beyond the **mantissa bits**.
- Assists in **rounding** by indicating whether the **truncated portion** is **closer to 0 or 1**.

Sticky Bit (S):

- Represents **whether any bits beyond the round bit** are **non-zero**.
- If **any bits** shifted out during **normalization** are **1**, the **sticky bit** is **set to 1**, otherwise, it remains **0**.
- **Captures information** about whether the discarded bits are **non-zero**, contributing to **correct rounding**.

2. Problem Recap: Adding $1.011_2 + 0.0011011_2$

1. Normalize the Numbers:

Binary	Normalized Form	Exponent	Mantissa
1.011_2	1.011×2^0	0	011
0.0011011_2	1.1011×2^{-3}	-3	101

2. Align the Exponents:

The **higher exponent** is **0**, so **shift** the **second number right by 3 places** to align:

$$1.1011 \times 2^{-3} = 0.0011011 \times 2^0$$

$$0.0011011 \times 2^0$$

3. Bit Representation (Including Guard, Round, and Sticky Bits)

After Shifting to Align Exponents:

$$\text{Mantissa (3 bits)} = 001$$

$$\text{Guard Bit (G)} = 1$$

$$\text{Round Bit (R)} = 0$$

$$\text{Sticky Bit (S)} = 1$$

Binary Addition:

$$1.011 + 0.001 = 1.100$$

Bit Values After Addition:

Bit Type	Bit Value
Mantissa	100
Guard Bit (G)	1
Round Bit (R)	0
Sticky Bit (S)	1

$$1.101_2 = 1.625$$

G R S	Action	Easy Rule
1 1 X	Round Up	If round bit is 1, always round up
1 0 1	Round Up	Sticky bit 1? Round up!
1 0 0	Round to Nearest Even	Mantissa odd? Round up
0 X X	Do Not Round Up	Guard bit 0 means no rounding

6. Using 32-bit IEEE 754 single precision floating point with one(1) sign bit, eight (8) exponent bits and twenty three (23) mantissa bits, show the representation of -11/16 (-0.6875).

Field	Bits	Description
Sign Bit	1	**0** for positive, **1** for negative
Exponent	8	**Excess-127** encoding (bias = **127**)
Mantissa	23	**Normalized** with an **implicit leading 1**

Fractional Conversion:

$$-0.6875 = -\left(\frac{11}{16}\right)$$

Convert the **fractional part** to **binary**:

$$0.6875 \times 2 = 1.375 \to 1$$

$$0.375 \times 2 = 0.75 \to 0$$

$$0.75 \times 2 = 1.5 \to 1$$

$$0.5 \times 2 = 1.0 \to 1$$

Combine the bits:

$$0.6875_{10} = 0.1011_2$$

Normalized Binary Form:

$$0.1011 = 1.011 \times 2^{-1}$$

- **Mantissa: 011**
- **Exponent: -1**

The **exponent** in **excess-127** format:

$$\text{Biased Exponent} = -1 + 127 = 126$$

Convert **126** to **8-bit binary**:

$$126 = 01111110_2$$

The **normalized mantissa** is **1.011**, but we store only the **fractional part**:

$$\text{Mantissa} = 01100000000000000000000$$

The **remaining bits** are **padded with zeros** to complete **23 bits.**

$$\text{Sign Bit} = 1$$

$$\text{Exponent} = 01111110$$

$$\text{Mantissa} = 01100000000000000000000$$

Combined Binary Representation:

$$1\ 01111110\ 01100000000000000000000$$

Feedback Survey