Exercise 3.3

3.3.1

For a. $A-B=-21$, underflow

For b. $A-B=58$, neither

3.3.2.

For a. $A+B=159>127$, overflow

For b. $A+B=146>127$, overflow

3.3.3.

For a. $A-B=-21 \in [-127,127]$ Neither

For b. $A-B=58 \in [-127,127]$ Neither

3.3.4.

For a. $A+B=303>127$, result $=127$ (due to saturation)

For b. $A+B=484>127$, result $=127$ (due to saturation)

3.3.5.

For a. $A-B=97$ (with in range)

For b. $A-B=10$ (within range)

3.3.6.

For a. $A+B=303>255$, result $=255$ (due to saturation)

For b. $A+B=484>255$, result $=255$ (due to saturation)

Exercise 3.10.

3.10.1.

For a. As an unsigned integer: 61485875b
As a two's-complement integer: 61485875b

For b. As an unsigned integer: 2948530176
As a two's-complement integer: -1346437120

3.10.2.

For a. addiu $6. $5. 0x4

For b. sw $31, 0x0($29)

3.10.3

For a. 7.199×10^{-17}

For b. -3.474×10^{-10}

3.10.4.

For a. $(C4C93000)_H$

For b. $(C46AB400)_H$

3.10.5.

For a. $(C0992600\,00000000)_H$

For b. $(C08D5680\,00000000)_H$

3.10.6.

For a. $(C2400000)_H$

For b. $(C2640000)_H$

Exercise 3.11

3.11.1

For a. $(12E8C4400)_H$

For b. $(FQAC00000)_H$

Comparison: This format provides greater precision due to the 24-bit mantissa compared to single precision IEEE 754. However, using magnitude for the exponent rather than two's-complement might limit its efficient representa-tion of small numbers.

3.11.2.

For a $(40004000)_H$

For b. $(ABOO)_H$

Comparison: The "half" format provides signifi-cantly less precision with only 10 bits in the manitissa and a smaller range due to the 5-bit exponent It's more compact but less precise and has a narrower range than IEEE 754.

3.11.3.

For a. $(3A3|011|2)_H$

For b. $(FAC00000)_H$

Comparison: The HP 21xx format allows for higher precision with a 24-bit mantissa compared to single precision's 23-bit mantissa

(plus 1 implicit bit). However, the use of the entire 8 bits for a two's-complement exponent could limit the efficient range compared to IEEE754, which uses an exponent in bias format.

3.11.4 Recall NVIDIA "Half" Format Specifications:

Sign bit: 1 bit
Exponent: 5 bits (excess-16)
Mantissa: 10 bits
Total: 16 bits

For a.

Step 1. Convert the Numbers to Half-Precision Format

A: 1 10010 0011011110
B: 1 01110 1001000000

Step 2. Align Exponents and Add Mantissas

B's mantissas needs to be right-shifted 22 places to match the exponents.

Step 3. Rounding and Final Result

After shifting, B effectively becomes zero and the final result remains A unchanged

For b.

Step1. A: 0 10100 0111000110

B: 0 01100 0000111010

Step2: Adjust B's exponents from 12 to 20 by right-shifting the mantissa by 8 places

Step3. the result remains 23.173291

Exercise 3.13

3.13.1

For a.

Step1. Convert Values to Half-Precision

A: 1 11110 0000000000

B: 0 11110 0000000000

C: 0 01111 0000000000

Step2. Perform (A+B)

A+B=0

Step3. Perform ((A+B)+C)

Result in Decimal: 1.0

Result in 16-bit: 0 01111 0000000000

For b.

Step1. A: 0 10011 1000110011

B: 00110 0101001100

C: 0 10011 1100100110

Step 2. A's mentissa dominates due to larger magnitude and precision loss in B

Step3.

Result in Decimal: 40.703125

Result in 16-bit: 0|01000|01|1000

3.13.2

For a.

Step1. remain the same

Step2. perform B+C

Step3. Perform (A+(B+C))

Result in Decimal: 1.0

Result in 16-bit : 0011110000000000

For b.

Step1. The same

Step2. Perform B+C

Step3. Perform (A+(B+C))

Result in Decimal: 40.72105850

Result in 16-bit: 0|01000|11|01|00111

3.13.3

In these calculations, due to the precision limitations and exponent scaling in the half format, (A+B)+C and A+(B+C) could yield different results especially handling values that are close to the format's precision limits

or exponent overflow/underflow conditions.

3.13.4.

For a.

Step1. Convert each number to binary and normalize

A: 0 00011 1111000000

B: 0 11001 0111100000

C: 0 10110 1111000001

Step2. Perform A×B

0 10101 0000000000

Step3 Perform (A×B) ×C

Result in Decimal: 104857.6

Result in 16-bit: 0110101000000000

For b.

Step1. A: 0 10100 0111 0011

B: 0 10011 0000 0110

C: 0 10100 0011 1100

Step2. Exponent Calculation: 20+19-15=24

Multiply 0 1111 0011 x 1 1000 0110

Step3. Exponent Calculation: 24+20-15=29

This multiplication exceed the normal range and lead to an overflow

3.14.5.

For a.

Step 1. the same

Step 2. Perform B×C

0 1111 0000 0000

Step 3. Perform A×(B×C)

0 110 | 0000 0000

$1.0 \times 2^{20} \approx 1048576$

For b.

Step 1. the same

Step 2. Exponent Calculation: 19+20−15=24

Multiply 1100 0011 0 × 0001 1110

Step 3. Exponent Calculation: 20+24−15=29

this too seems beyond the maximum representable value.

3.14.6. The result suggests that they may indeed be equal, assuming precise handling of exponent alignment and mantissa rounding. Under certain conditions, floating-point multiplication can show associative properties despite general cases where rounding errors can introduce discrepancies.