CheggSolutions - Thegdp

Floating-Point Arithmetic and IEEE-754 Standard

Part a: Convert 27.8125 into IEEE-754 32-bit Single Precision Representation

Given: Number to be converted: 27.8125

Step-by-Step Solution:

1. **Step 1: Convert to binary**
   1. Convert the integer part (27) to binary: $27_{10} = 11011_2$
      Explanation: Convert the decimal integer 27 to its binary equivalent.
   2. Convert the fractional part (0.8125) to binary: $0.8125_{10} = 0.1101_2$
      Explanation: Multiply the fractional part by 2 repeatedly, taking the integer part of the product as each binary digit until the fractional part is 0 or within desired precision.
   3. Combining both parts: $27.8125_{10} = 11011.1101_2$
2. **Step 2: Normalize the binary number**

   Normalized form: $1.10111101_2 \times 2^4$

   Explanation: Move the binary point to immediately after the first 1, adjusting the exponent accordingly.

3. **Step 3: Determine the sign bit**

   Sign bit (S): $0 \quad (\text{positive number})$

   Explanation: The sign bit is 0 for positive numbers and 1 for negative numbers.

4. **Step 4: Determine the exponent**
   1. Unbiased exponent for 2^4 is 4.
   2. Bias for IEEE-754 single-precision (127): $\text{Exponent} = 4 + 127 = 131$
      Explanation: Adjust the exponent to account for the IEEE-754 bias of 127.
   3. Exponent in binary: $131_{10} = 10000011_2$
5. **Step 5: Determine the fraction (mantissa)**

   Mantissa: $10111101000000000000000_2$

   Explanation: Use the normalized form mantissa (without the leading 1), filled to 23 bits.

6. **Step 6: Combine all parts**
   1. IEEE-754 representation: $0 \quad 10000011 \quad 10111101000000000000000$
   2. In hexadecimal: $41DE8000_{16}$

Final Solution:

$$27.8125_{10} \text{ in IEEE-754 format} = 0\ 10000011\ 10111101000000000000000 \ (\text{binary})$$

$$= 41DE8000_{16} \ (\text{hexadecimal})$$

Explanation: This part converts the decimal number to its IEEE-754 32-bit single-precision format, which is helpful for understanding floating point representation in computing.

Part b: Carry out the addition $27.8125 + 13.5$ in IEEE-754 Single Precision Arithmetic

Given:

$\begin{aligned} &27.8125 \rightarrow 0 \ 10000011 \ 10111101000000000000000 \ (\text{binary}) \rightarrow 41DE8000_{16} \\ &13.5 \rightarrow 0 \ 10000010 \ 1011000000000000000000 \ (\text{binary}) \rightarrow 41580000_{16} \end{aligned}$

Step-by-Step Solution:

1. **Step 1: Align the exponents**

   Convert both to binary and align exponents by shifting the mantissa:

   - $(27.8125) \rightarrow 1.10111101 \times 2^4 \rightarrow 0\ \underline{10000011}\ 10111101000000000000000$
   - $(13.5) \rightarrow 1.101 \times 2^3 \rightarrow 0\ \underline{10000011}\ 01101000000000000000000$

   Explanation: Normalize the numbers and align the exponents by shifting the mantissa of the smaller exponent number to the right.

2. **Step 2: Add the mantissas**

   Add the mantissas with the aligned exponents:

   $1.10111101 + 0.11010000 = 10.10001101$

Explanation: Binary addition of the mantissas.

3. **Step 3: Normalize the result**

   Normalize the sum:

   $10.10001101 \times 2^4 = 1.010001101 \times 2^5$

   Explanation: Normalize the resulting sum to maintain a single leading 1.

4. **Step 4: Adjust the exponent**

   Exponents need adjustment due to normalization:

   $4 + 1 = 5 \implies \text{Exponent} = 5 + 127 = 132 \rightarrow 10000100_2$

   Explanation: Recalculate the biased exponent after normalization.

5. **Step 5: Forming the result**

   Combine the final sign, exponent, and mantissa:

   $\text{Result} = 0\ 10000100\ 01000110100000000000000$

   In hexadecimal:

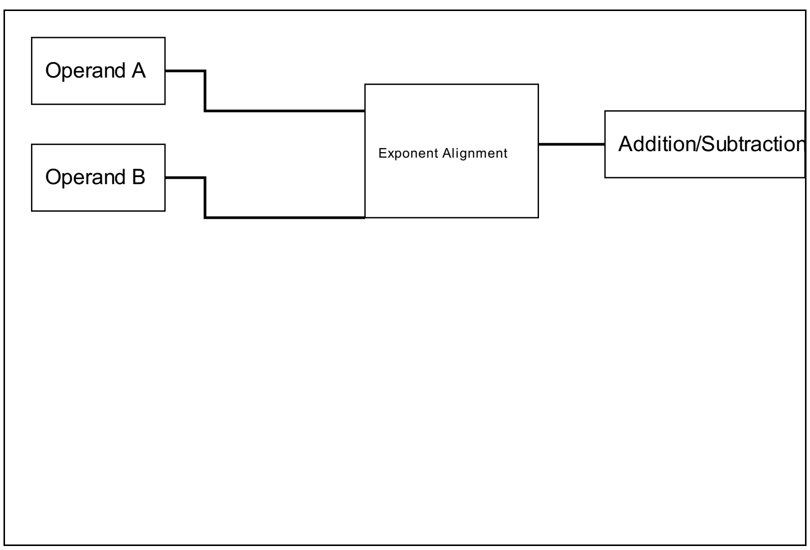
   $\approx 42892000_{16} \quad (\text{Note: a rough representation})$

Explanation: This part performs the binary addition of two IEEE-754 floating-point numbers, illustrating the steps of alignment, mantissa addition, normalization, and combination.

Part ii: Draw the data flow for floating point addition and subtraction corresponding to the hardware implementation

Data Flow Diagram (simplified):

1. **Input operands:** Two floating-point numbers.
2. **Unpack:** Extract sign, exponent, and mantissa for each operand.
3. **Exponent alignment:** Shift mantissa of the smaller exponent to align exponents.
4. **Arithmetic operation:** Perform addition or subtraction on the aligned mantissas.
5. **Normalization:** Normalize the result to maintain standard form.
6. **Rounding if necessary.**
7. **Pack result:** Combine sign, exponent, and mantissa to form the IEEE-754 result.



Explanation: The data flow diagram summarizes the hardware stages for floating-point addition/subtraction from unpacking the numbers through to the final normalized result.