**QUESTION 1**

Octal 62 = 0011 0010 🡪 Multiplicand

Octal 12 = 0000 1010 🡪 Multiplier

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Step | Multiplier | Multiplicand | Product |
| 0 | Initial Values | 1010 | 0011 0010 | 0000 0000 |
| 1 | SL Multiplicand  SR Multiplier | 1010  0101 | 0110 0100  0110 0100 | 0000 0000  0000 0000 |
| 2 | Add Multiplicand To Product  SL Multiplicand  SR Multiplier | 0101  0101  0010 | 0110 0100  11001000  11001000 | 0000 0110 0100  0000 0110 0100  0000 0110 0100 |
| 3 | SL Multiplicand  SR Multiplier | 0010  0001 | 0001 1001 0000  0001 1001 0000 | 0000 0110 0100  0000 0110 0100 |
| 4 | Add Multiplicand To Product  SL Multiplicand  SR Multiplier | 0001  0001  0000 | 0001 1001 0000  0011 0010 0000  0011 0010 0000 | 0001 1111 0100  0001 1111 0100  0001 1111 0100 |

**QUESTION 2**

Hex 62 = 0110 0010 🡪 Multiplicand

Hex 12 = 0001 0010 🡪 Multiplier

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Step | Multiplier | Multiplicand | Product |
| 0 | Initial Values | 0001 0010 | 0000 0110 0010 | 0000 000**0** 0000 0000 |
| 1 | 0 🡪 Nothing  SL Multiplicand  SR Product + Multiplier | 0001 0010  0001 0010  0000 1001 | 0000 0110 0010  0000 1100 0100  0000 1100 0100 | 0000 000**0** 0000 0000  0000 000**0** 0000 0000  0000 0000 **0**000 0000 |
| 2 | 1 🡪 Product = Product + Multiplicand  SL Multiplicand  SR Product + Multiplier | 0000 1001  0000 1001  0000 0100 | 0000 1100 0100  0001 1000 1000  0001 1000 1000 | 0110 0010 **0**000 0000  0110 0010 **0**000 0000  0011 0001 0**0**00 0000 |
| 3 | 0 🡪 Nothing  SL Multiplicand  SR Product + Multiplier | 0000 0100  0000 0100  0000 0010 | 0001 1000 1000  0011 0001 0000  0011 0001 0000 | 0011 0001 0**0**00 0000  0011 0001 0**0**00 0000  0001 1000 10**0**0 0000 |
| 4 | 0 🡪 Nothing  SL Multiplicand  SR Product + Multiplier | 0000 0010  0000 0001  0000 0001 | 0011 0001 0000  0110 0010 0000  0110 0010 0000 | 0001 1000 10**0**0 0000  0001 1000 10**0**0 0000  0000 1100 010**0** 0000 |
| 5 | 1 🡪 Product = Product + Multiplicand  SL Multiplicand  SR Product + Multiplier | 0000 0001  0000 0001  0000 0000 | 0110 0010 0000  1100 0100 0000  1100 0100 0000 | 0110 1110 010**0** 0000  0110 1110 010**0** 0000  0011 0111 0010 **0**000 |

**QUESTION 3**

Octal 74 = 0011 1100 🡪 Remainder/Dividend

Octal 21 = 0001 0001 🡪 Divisor

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Step | Quotient | Divisor | Remainder |
| 0 | Initial Values | 000000 | 010001 000000 | 000000 111100 |
| 1 | Remainder = Remainder – Divisor  Remainder < 0 🡪 +Div, sll Quotient, Q0 = 0  SR Divisor | 000000  000000  000000 | 010001 000000  010001 000000  001000 100000 | 101111 111100  000000 111100  000000 111100 |
| 2 | Remainder = Remainder – Divisor  Remainder < 0 🡪 +Div, sll Quotient, Q0 = 0  SR Divisor | 000000  000000  000000 | 001000 100000  001000 100000  000100 010000 | 111000 011100  000000 111100  000000 111100 |
| 3 | Remainder = Remainder – Divisor  Remainder < 0 🡪 +Div, sll Quotient, Q0 = 0  SR Divisor | 000000  000000  000000 | 000100 010000  000100 010000  000010 001000 | 111100 101100  000000 111100  000000 111100 |
| 4 | Remainder = Remainder – Divisor  Remainder < 0 🡪 +Div, sll Quotient, Q0 = 0  SR Divisor | 000000  000000  000000 | 000010 001000  000010 001000  000001 000100 | 111110 110100  000000 111100  000000 111100 |
| 5 | Remainder = Remainder – Divisor  Remainder < 0 🡪 +Div, sll Quotient, Q0 = 0  SR Divisor | 000000  000000  000000 | 000001 000100  000001 000100  000000 100010 | 111111 111000  000000 111100  000000 111100 |
| 6 | Remainder = Remainder – Divisor  Remainder >= 0 🡪 sll Q, Q0 = 1  SR Divisor | 000000  000001  000001 | 000000 100010  000000 100010  000000 010001 | 000000 011010  000000 011010  000000 011010 |
| 7 | Remainder = Remainder – Divisor  Remainder >= 0 🡪 sll Q, Q0 = 1  SR Divisor | 000001  000011  000011 | 000000 010001  000000 010001  000000 001000 | 000000 001001  000000 001001  000000 001001 |

**QUESTION 4**

− 1.5625 X 10-1 🡪 -0.15625 x 100

− = Sign Bit

0 = (0)2

0.15625 = (00101)2

0.00101 🡪 1.01 x 2-3

Sign Bit = 1

Exponent = -3 + 15 = 12

12 = (01100)2

5 bits for exponent

Mantissa = 0100 0000 00

10 bits left for mantissa

IEEE 754-2008 Representation: 1011 0001 0000 0000

IEEE 754 Representation: 1011 1110 0010 0000 0000 0000 0000 0000

The single precision (2nd one) is better than half precision (1st one) in terms of range and accuracy.

**QUESTION 5**

**Step 1:**

-8.0546875 x 100

8 = (1000)2

0.0546875 = (0000111)2

1000.0000111🡺1.0000000111 x 23

**Step 2:**

-1.79931640625x10-1 🡪 0.179931640625 x 100

0 = (0)2

0.179931640625 = (001011100001)2

**Step 3:**

0.001011100001🡺1.011100001 x 2-3

10000000111

x 10111000010

-----------------

1.01110011000001001110

Note: Answer is positive b/c the numbers above are negative.

Negative x negative = positive (-1 x -1 = 1).

**Step 4:**

Mantissa = 0111001100

Exponent = 3 – 3 = 0 + 15 = 15

15 = (10000)2

0 10000 0111001101 = (1.44921875)2

**Step 5:**

(-8.0546875 x 100) (-1.79931640625x10-1) = 1.449293137

(Calculated On A McMaster Certified Calculator [Casio – fx-991MS])

**Step 6:**

epsilon = | binary – decimal |

= 1.44921875 - 1.449293137

= 7.4387x10-5

Since epsilon != 0, precision was lost during conversion

**Step 7:**

No overflow or underflow 🡪 No further comments

My result is accurate enough for normal use because epsilon is a very small number (10-5). It’s accurate for 4 decimal places which is good enough. However, it’s not accurate enough for professional use (i.e. Defense Systems).

Compared to my calculator, the answer is off b/c information was lost when converting from decimal to binary and performing computations.

**QUESTION 6**

8.625 x 101 🡪 86.25 x 100

– 4.875 x 100 🡪 -4.875 x 100

86.25 x 100

86 = (1010110)2

0.25 = (01)2

1010110.01 🡪 1.01011001 x 26

-4.875 x 100

– = Sign Bit

4 = (100)2

0.875 = (111)2

100.111 🡪 1.00111 x 22

(1.01011001) / (1.00111)

Exponent = 6 – 2 = 4

= 1000110110…

Sign Bit = – 🡪 1

Exponent = 15 + 4 = 19 🡪 (10011)2

Mantissa = 1000110110

16-Bit Float Point Format As Described In Q4:

1 10011 1000110110 🡪 1100 1110 0011 0110

= -17.69230769 (Calculated On A McMaster Certified Calculator [Casio – fx-991MS])

1100 1110 0011 0110 = (-)2

Epsilon = | binary – decimal |

= | x – (-17.69230769) |

Since epsilon != 0, there is a loss of precision and information. Therefore, my calculator is more accurate and precise. See Q5 for more details. The precision/accuracy/results are the similar.