Keegan Smith

Computer Architecture and Design

HW2

10/24/2023

1. This problem covers 4-bit binary multiplication. Fill in the table for the Product, Multiplier and Multiplicand for each step. You need to provide the description of the step being performed (shift left, shift right, add, no add). The value of M (Multiplicand) is 1011, Q (Multiplier) is initially 1010.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Product | Multiplicand | Multiplier | Description | Step |
| 0000 0000 | 0000 1011 | 0000 1010 | Initial Values | Step 0 |
| 0000 0000 | 0000 1011 | 0000 1010 | Multiplying first digit of multiplier with second of multiplicand | Step 1 |
| 0000 0000 | 1011 | 1010 | Multiplying first digit of multiplier with third of multiplicand | Step 2 |
| 0000 0000 |  |  | Multiplying first digit of multiplier with fourth of multiplicand | Step 3 |
| X |  | 1 | Place don’t care due to shift | Step 4 |
| 1X | 1 | 1 | Multiplying second digit of multiplier with first of multiplicand | Step 5 |
| 11X | 1 | 1 | Multiplying second digit of multiplier with first of multiplicand | Step 6 |
| 011X | 0 | 1 | Multiplying second digit of multiplier with third of multiplicand | Step 7 |
| 1011X | 1 | 1 | Multiplying second digit of multiplier with fourth of multiplicand | Step 8 |
| XX | 1 | 0 | Double shift for the third digit of the multiplier | Step 9 |
| 0000XX | 1011 | 0 | Multiply the zero through the multiplicand | Step 10 |
| XXX |  | 1 | Triple shift for the fourth digit of the multiplier | Step 11 |
| 1XXX | 1 | 1 | Multiplying fourth digit of multiplier with first of multiplicand | Step 12 |
| 11XXX | 1 | 1 | Multiplying fourth digit of multiplier with second of multiplicand | Step 13 |
| 011XXX | 0 | 1 | Multiplying fourth digit of multiplier with third of multiplicand | Step 14 |
| 1011XXX | 1 | 1 | Multiplying fourth digit of multiplier with fourth of multiplicand | Step 15 |
| 0000 0000  0001 011X  0000 00XX  0101 1XXX |  |  | Add up products | Step 16 |
| 0110 1110 |  |  | Final Result | Step 17 |

1. This problem covers floating-point IEEE format.
   1. [10pts] list four floating-point operations that cause NaN to be created?
   2. [10pts] Assuming single precision IEEE 754 format, what decimal number is represented by this word?

1 01111101 00100000000000000000000

1. 1. Divide 0 by 0

2. Divide infinity by infinity

3. Multiply 0 by infinity

4. Multiply infinity by 0

1. 1 01111101 00100000000000000000000 to decimal.

|  |  |  |
| --- | --- | --- |
| 1 | 01111101 | 00100000000000000000000 |
| Negative number | 125 for bias exponent | 2^(-3) = 0.125 |

1. Perform the following operations by converting the operands to 2’s complement binary numbers and then doing the addition or subtraction shown. Please show all work in binary, operating on 16-bit numbers.
   1. 3+13
   2. 13-2
   3. 5-8
   4. -7-(-7)

|  |  |  |  |
| --- | --- | --- | --- |
| Number | Binary conversion | 2’s compliment solution | solution |
| 3 + 13 | 0 000 0000 0000 0011  + 0 000 0000 0000 1101 | 0 000 0000 0001 0000 | 0 000 0000 0001 0000 |
| 13 – 2 | 0 000 0000 0000 1101  – 1 000 0000 0000 0010 | 0 000 0000 0000 0010 -1’s comp of 2  1 111 1111 1111 1101  + 1 -2’s comp of 2  1 111 1111 1111 1110  0 000 0000 0000 1101 – 13  + 1 111 1111 1111 1110 -2’s of 2  0 000 0000 0000 1011 | 0 000 0000 0000 1011 |
| 5 – 8 | 0 000 0000 0000 0101  – 0 000 0000 0000 1000 | 0 000 0000 0000 1000 -8  1 111 1111 1111 0111 -1’s comp of 8  + 1 -2’s comp of 8  1 111 1111 1111 1000  0 000 0000 0000 0101 -5  + 1 111 1111 1111 1000 -2’s of 8  1 … 1101 | 1 000 0000 0000 0011 |
| -7 – (-7) | 1 000 0000 0000 0111  – 1 000 0000 0000 0111 | 1 000 0000 0000 0111  0 111 1111 1111 1000 -1’s of 7  + 1 -2’s of 7  0 111 1111 1111 1001  0 111 1111 1111 1001  + 0 000 0000 0000 0111  0000 with overflow | 0 |

1. The floating-point format to be used in this problem is a normalized format with 1 sign bit, 3 exponent bits, and 4 mantissa bits. The exponent field employs an excess-4 coding. 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.
   1. [10pts] Encode the following numbers in the above format:

|  |  |  |
| --- | --- | --- |
| Sign (1) | Exponent (3) | Mantissa (4) |
| 0 | 000 | 0000 |

* + 1. 1.0binary = 1decimal

Sign = 0

Exp = 0 + 4 = 4, 100

Mantissa = 0000

Result: 0 100 0000

* + 1. 0.0011001binary => Normalized = 1.1001 x 2^(-3)

Sign = 0

Exp = -3 + 4 = 1

Mantissa = 1001

Result: 0 001 1001

* 1. Perform rounding on the following fractional binary numbers, use the bit positions in italics to determine rounding (use the rightmost 3 bits):

* + 1. Round to positive infinity: +0.100101*110*binary

Rounded: 0.10011

* + 1. Round to negative infinity: -0.001111*001*binary

Rounded: -0.01000

1. This problem covers 4-bit binary unsigned division (similar to Fig. 3.11 in the text). Fill in the table for the Quotient, Divisor and Dividend for each step. You need to provide the DESCRIPTION of the step being performed (shift left, shift right, sub). The value of Divisor is 4 (0100, with additional 0000 bits shown for right shift), Dividend is 6 (initially loaded into the Remainder).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Quotient | Divisor | Remainder | Description | Step |
| 0000 | 0100 0000 | 0000 0110 | Initial values | 0 |
| 0000 | 0100 0000 | 1100 0110 | Rem = rem – div | 1 |
| 0000 | 0100 0000 | 0000 0110 | Rem < 0 => + div, sll Q, Q0 = 0 | 2 |
| 0000 | 0010 0000 | 0000 0110 | Shift Divisor to the right | 3 |
| 0000 | 0010 0000 | 1110 0110 | Rem = Rem – div | 4 |
| 0000 | 0010 0000 | 0000 0110 | Rem < 0 => + div, sll Q, Q0 = 0 | 5 |
| 0000 | 0001 0000 | 0000 0110 | Shift divisor to the right | 6 |
| 0000 | 0001 0000 | 1111 0110 | Rem = Rem – div | 7 |
| 0000 | 0001 0000 | 0000 0110 | Rem < 0 => + div, sll Q, Q0 = 0 | 8 |
| 0000 | 0000 1000 | 0000 0110 | Shift div to the right | 9 |
| 0000 | 0000 1000 | 1111 1110 | Rem = Rem – div | 10 |
| 0000 | 0000 1000 | 0000 0110 | Rem < 0 => + div, sll Q, Q0 = 0 | 11 |
| 0000 | 0000 0100 | 0000 0110 | Shift div to the right | 12 |
| 0000 | 0000 0100 | 0000 0010 | Rem = Rem – div | 13 |
| 0001 | 0000 0100 | 0000 0010 | Rem < 0 => +Div, sll Q, Q0 = 0 | 14 |
| 0001 | 0000 0010 | 0000 0010 | Shift div to the right | 15 |