INTEL APPROACH DOCUMENT

Design and Implementation of Slow and Fast Division algorithms in Computer Architechture

# STEPS TO BE FOLLOWED IN DIVISION ALGORITHMS:

# SLOW DIVISION ALGORITHM(RESTORING ALGORITHM):

The restoring division works by restoring the dividend after each subtraction, in such a way that the dividing process continues until the needed precision and accuracy is reached.

The algorithm starts by determining the sign of the result based on the signs of the dividend and divisor and then converts both the values to positive numbers for simplicity.

The division process operates in a loop that iterates over the bit positions of the dividend, from the most significant bit to the least significant bit. The algorithm shifts the remainder one position to the left and sets the least significant bit of the remainder to the current bit of the dividend.

If the remainder is greater than or equal to the divisor, the divisor is subtracted from the remainder and the corresponding bit in the quotient is set to 1. Otherwise, the bit in the quotient is set to 0.After the completion of the loop,the algorithm applies the sign to the quotient and returns both the quotient and remainder.

Algorithm :

1. Get the dividend N and divisor D from the user.
2. Initialize a new variable R=N
3. Compute R=2\*R-D
4. Use a for loop and perform n-1 iterations starting from the i value 1.
5. Using if clause check R>=0

Then Q(i)=1

Using else if clause check R<0

Then Q(i)=0

R=R+D

1. End if clause
2. End for loop

Here is an example of output waveform for the division of 13 divided by 3:

┌──┐

Dividend: 13 │11│ 1

Divisor: 3 └──┘

┌──┐┌──┐

│ 1││10│ 1

└──┘└──┘

┌──┐┌──┐┌──┐

│ 3││ 1││ 8│ 1

└──┘└──┘└──┘

┌──┐┌──┐┌──┐┌──┐

│ 9││ 2││ 6││ 6│ 1

└──┘└──┘└──┘└──┘

Quotient: 4 3 2 1

Remainder: 1

# FAST DIVISION ALGORITHM(GOLDSMIDT ALGORITHM):

The Goldsmidt algorithm starts by initializing registers and counters to keep track of the calculations.Loops are created to perform the number of iterations needed to get the needed results. In each iteration of the loop, we take the remainder from the previous step and multiply it by 2, which helps us get closer to the answer.Then, the quotient is multiplied by the divisor, giving us a new remainder smaller than the previous one.These calculations and updations on the remainder are performed until we have gone through the loop for a fixed number of times, based on the number of bits in the numbers.

Once the loop is complete, we extract the lower 8 bits of the quotient and remainder and assign them to the output signals. Finally, we set a flag called "done" to indicate that the division is complete.

Algorithm is as follows:

1. Initialize the value of R (reciprocal estimate) to the initial guess, which is nothing but 1/divisor.
2. Perform iterations until the desired precision and accuracy is reached.
3. a. Compute P= divisor\*R

b. Find the difference D = 2-P

c. Update the value of R as R=R\*D

4. Repeat step 3 for desired number of times.

An example of Goldschmidt’s division algorithm:

Given inputs,

Dividend:

N−1=8610=01010110.0000000000002

Divisor:

D−1=710=00000111.0000000000002

RESULT OBTAINED:

Q=N−1D−1=12.285714285714285714285714285714285714285714285714

Initial reciprocal is the inverse of divisor which is calculated by shifting bits around the fraction point:

F−1=1D−1≈0.054687510=00000000.0000111000002

In this case we have 8 bits for integers and 12 bits for fraction digits.

ALGORITHM STEPS

* Iteration 1:

N0=F−1×N−1=4.70312510=00000100.1011010000002

D0=F−1×D−1=0.382812510=00000000.0110001000002

F0=2−D0=1.617187510=00000001.1001111000002

* Iteration 2:

N1=F0×N0=7.60571289062510=00000111.1001101100012

D1=F0×D0=0.61889648437510=00000000.1001111001112

F1=2−D1=1.38110351562510=00000001.0110000110012

* Iteration 3:

N2=F1×N1=10.50415039062510=00001010.1000000100012

D2=F1×D1=0.85473632812510=00000000.1101101011012

F2=2−D2=1.14526367187510=00000001.0010010100112

* Iteration 4:

N3=F2×N2=12.0297851562510=00001100.0000011110102

D3=F2×D2=0.97875976562510=00000000.1111101010012

F3=2−D3=1.02124023437510=00000001.0000010101112

* Iteration 5:

N4=F3×N3=12.2851562510=00001100.0100100100002

D4=F3×D3=0.9995117187510=00000000.1111111111102

F4=2−D4=1.0004882812510=00000001.0000000000102

Conclusion

The result after 5 iterations is:

N5=12.2851562510=00001100.0100100100002

Which deviates from the correct result by:

100%−|N5Q|=0.0045%

So the resultant output is almost accurate.

OUTCOMES OF BOTH SLOW AND FAST DIVISION FOR SAME INPUTS:

INPUT:

Dividend: 57

Divisor: 8

Restoring Division Algorithm:

Step 1: Initialize the dividend and divisor.

Dividend = 57

Divisor = 8

Step 2: Set the quotient to 0 and the remainder

to the dividend.

Quotient = 0

Remainder = 57

Step 3: Repeat the following steps until the remainder is smaller than the divisor:

a. Subtract the divisor from the remainder.

b. Increment the quotient by 1.

Iteration 1:

Remainder = Remainder - Divisor

= 57 - 8

= 49

Quotient = Quotient + 1

= 0 + 1

= 1

Iteration 2:

Remainder = Remainder - Divisor

= 49 - 8

= 41

Quotient = Quotient + 1

= 1 + 1

= 2

Iteration 3:

Remainder = Remainder - Divisor

= 41 - 8

= 33

Quotient = Quotient + 1

= 2 + 1

= 3

Iteration 4:

Remainder = Remainder - Divisor

= 33 - 8

= 25

Quotient = Quotient + 1

= 3 + 1

= 4

Iteration 5:

Remainder = Remainder - Divisor

= 25 - 8

= 17

Quotient = Quotient + 1

= 4 + 1

= 5

Iteration 6:

Remainder = Remainder - Divisor

= 17 - 8

= 9

Quotient = Quotient + 1

= 5 + 1

= 6

Iteration 7:

Remainder = Remainder - Divisor

= 9 - 8

= 1

Quotient = Quotient + 1

= 6 + 1

= 7

Step 4: The final quotient obtained is the result of the division, and the remainder is the remainder of the division operation.

Quotient = 7

Remainder = 1

Goldschmidt’s Algorithm:

Step 1: Initialize the dividend and divisor.

Dividend = 57

Divisor = 8

Step 2: Calculate the reciprocal of the divisor (1/divisor).

Reciprocal of 8 = 0.125

Step 3: Multiply the dividend by the reciprocal to obtain an intermediate result.

Intermediate Result = Dividend \* Reciprocal of 8

= 57 \* 0.125

= 7.125

Step 4: Round the intermediate result to the nearest integer.

Rounded Intermediate Result = Round(7.125)

= 7

Step 5: The rounded intermediate result is the quotient.

Quotient = 7

Step 6: Calculate the remainder by subtracting the product of the rounded quotient and divisor from the dividend.

Remainder = Dividend - (Quotient \* Divisor)

= 57 - (7 \* 8)

= 57 - 56

= 1

Both the algorithms provides the same outcome but the iterations differs.

Restoring Division Algorithm:

Outcome: The restoring division algorithm takes a longer time to compute the division result compared to other more efficient algorithms. It performs division by repeatedly subtracting the divisor from the dividend until the remainder becomes smaller than the divisor.

Reasoning: The restoring division algorithm is a straightforward approach that emulates the manual long division process. It relies on repeated subtraction, which makes it time-consuming for larger numbers. The algorithm iteratively subtracts the divisor from the dividend, incrementing the quotient each time, until the remainder is smaller than the divisor. This process continues until the entire dividend is exhausted, providing the quotient and remainder as the result. While it is conceptually simple, it becomes inefficient when dealing with large numbers due to the repetitive subtractions involved.

Goldschmidt’s Division Algorithm:

Outcome:

The Goldschmidt’s division algorithm provides a quicker way to compute division results by using more advanced mathematical techniques, such as multiplication and bit shifting.

Reasoning:

The Goldschmidt’s division algorithm is designed to reduce the number of iterations required for division, resulting in faster computations. It relies on various optimizations to improve efficiency. One common approach is to convert the division operation into a multiplication by the reciprocal of the divisor. This eliminates the need for repeated subtractions and replaces them with multiplications, which are typically faster to compute. Additionally, the algorithm may utilize techniques like bit shifting to further optimize the calculations. These optimizations help reduce the number of steps needed to find the quotient and remainder, resulting in a faster division process.