| Experiment No. 5 |
| --- |
| Implement restoring division algorithm |
| Date of Performance: |
| Date of Correction: |

**Aim:** To implement a program that performs binary division using the restoring division algorithm.

**Objective:** To understand and implement the restoring division algorithm used in digital systems for binary division operations.

**Theory:**

**Introduction:**

In computer architecture, binary division is a fundamental arithmetic operation performed by the Arithmetic Logic Unit (ALU) of a processor. Unlike multiplication and addition, which are relatively straightforward, division in hardware requires more complex logic.

The Restoring Division Algorithm is one such technique used for implementing binary division in hardware. It simulates long division, similar to how we do it manually, but in binary form using shifting and subtraction.

**How the Restoring Division Algorithm Works:**

**The algorithm operates on unsigned binary integers and performs division by:**

* Shifting the dividend left.
* Subtracting the divisor from the partial remainder.
* If the result is negative, it restores the previous value by adding the divisor back (hence the name *restoring*).
* A 0 is placed in the quotient in this case; otherwise, a 1 is placed.

**Step-by-Step Explanation:**

**Let’s define:**

* A (Accumulator / Remainder) — initialized to 0.
* Q (Dividend) — the number to be divided.
* M (Divisor) — the number by which we divide.
* n — number of bits.

**Steps:**

1. Initialize A = 0, and Q contains the dividend.
2. Repeat n times:
   * Left shift A and Q (combine them as AQ).
   * A = A − M
   * If A < 0:
     + Set Q₀ = 0
     + A = A + M (restore A)
   * Else:
     + Set Q₀ = 1
3. After n cycles, Q holds the quotient and A holds the remainder.

**Example (4-bit):**

**Let’s divide 13 (1101) by 3 (0011):**

* Q = 1101, M = 0011, A = 0000
* Shift AQ left → Subtract M from A
* If result negative → Restore A, set Q₀ = 0
* Else → Keep A, set Q₀ = 1
* Repeat for 4 cycles
* Final Q = Quotient, A = Remainder

**Applications in Computer Architecture:**

* Used in hardware-level division in older processors.
* Foundation for understanding non-restoring and SRT division algorithms.
* Useful for teaching binary arithmetic, control logic, and ALU design.
* Important in embedded systems and custom hardware where standard division instructions may not be available.

Although modern CPUs often use faster or more optimized algorithms (like non-restoring or Newton-Raphson), restoring division is crucial for understanding the basics of binary division in digital circuits.

**Solution:**

**Conclusion:** We implemented the restoring division algorithm and understood how binary division is performed in hardware using subtraction and shifting techniques.