**Non-Restoring Division**

**Write up:**

**roll no.1: ee18btech11029**

**roll no.2: ee18btech11049**

Binary division is basically a procedure to determine how many times the divisor D divides the dividend B thus resulting in the quotient Q. At each step in the process the divisor D either divides B into a group of bits or it does not. The divisor divides a group of bits when the divisor has a value less than or equal to the value of those bits. Therefore, the quotient is either 1 or 0. The division algorithm performs either an addition or subtraction based on the signs of the divisor and the partial remainder. There are number of binary division algorithm like Digit Recurrence Algorithm restoring, non-restoring and SRT Division, Multiplicative Algorithm etc

The non-restoring divide does not “restore” the remainder to the correct value but leaves it incorrect until the next cycle.

**Note:** The non-restoring algorithm can result in a negative remainder, which is incorrect. Therefore, a correction step is needed to obtain the correct remainder.

The Algorithm to perform non-restoring division is as follows:

**Algorithm details:**

* **Step-1:** First the registers are initialized with corresponding values (Q = Dividend, M = Divisor, R = 0, n = number of bits in dividend)
* **Step-2:** Then the content of register A and Q is shifted right as if they are a single unit
* **Step-3:** Then content of register M is subtracted from A and result is stored in A
* **Step-4:** Then the most significant bit of the A is checked if it is 0 the least significant bit of Q is set to 1 otherwise if it is 1 the least significant bit of Q is set to 0 and value of register A is restored i.e. the value of A before the subtraction with M
* **Step-5:** The value of counter n is decremented
* **Step-6:** If the value of n becomes zero, we get of the loop otherwise we repeat for step 2
* **Step-7:** Finally, the register Q contain the quotient and R contain remainder

**Restoring vs. Non-restoring division:**

It is less complex than the restoring one because simpler operation is involved i.e. addition and subtraction, also restoring step is performed.

In each step of division calculation, the result of the step is either 1 or 0, depending if the dividend is less than or larger than the divisor. We generally do a test subtraction for each digit step; if the result is positive or zero, we note down a 1 as next digit of our quotient.

If the result is negative, we proceed with one of two strategies:

**Restoring vs. Non-restoring division:**

* **restoring method:** add the divisor back, and put 0 as your next quotient digit
* **non-restoring method**: (don’t do that) keep negative remainder and a digit 1, and basically correct things by a supplementary addition afterwards.
* non-restoring is faster (max n+1 step for n digits), whereas restoring method (which is more natural) needs up to 2n-1 steps (you may be forced to do a correction addition per each step)

**Example:**

Dividend = x = 11

Divisor = y = 3

| **n** | **M** | **r** | **q** | **Operation** |
| --- | --- | --- | --- | --- |
| 4 | 00011 | 00000 | 1011 | initialize |
|  | 00011 | 00001 | 011\_ | shift left AQ |
|  | 00011 | 11110 | 011\_ | A=A-M |
|  | 00011 | 00001 | 0110 | Q [0] =0 And restore A |
| 3 | 00011 | 00010 | 110\_ | shift left AQ |
|  | 00011 | 11111 | 110\_ | A=A-M |
|  | 00011 | 00010 | 1100 | Q [0] =0 |
| 2 | 00011 | 00101 | 100\_ | shift left AQ |
|  | 00011 | 00010 | 100\_ | A=A-M |
|  | 00011 | 00010 | 1001 | Q [0] =1 |
| 1 | 00011 | 00101 | 001\_ | shift left AQ |
|  | 00011 | 00010 | 001\_ | A=A-M |
|  | 00011 | 00010 | 0011 | Q [0] =1 |

Remember to restore the value of r most significant bit of r is 1. As that register q contain the quotient, i.e. 3 and register r contain remainder 2.

**Circuit diagram:**





