## Russian Peasant Multiplication Programmed in ARM Assembly

This repository consists of the Russian Peasant Multiplication algorithm coded in ARM Assembly language. This particular algorithm is employed to perform multiplication of two integers using bit-manipulation and addition operations.

## Overview

Haberman’s algorithm also known as the Russian Peasant Multiplication is an algorithm based on the double and half method that only adds an extra step if the halved number is odd. This is done until the remaining a halved number is zero.

## Code Structure

The code is written in ARM Assembly language and performs the following steps:The code is written in ARM Assembly language and performs the following steps:

1. Initializes the multiplicand, the multiplier, and product numbers, which had been added to the circuit.

2. Enters a loop where it:Enters a loop where it:

- Tests if the multiplier is zero, if so, it then exits the loop.

- Looks at LSB of multiplier and adds multiplicand to the product if it contains 1’s.

- Doubles the multiplicand (shift left by 1).

- Divides the multiplier by 2 (by regarding the number as an integer and right shifting in by 1).

3. Continues the loop until the figure which multiplies is zero.

4. The result, which is the product, is then placed in the register R2, and only ready for use in other contentions iff R2 has a 0 in its MSB.

## Code Explanation

```assembly

AREA RussianPeasantMultiplication, CODE, READONLY

ENTRY

EXPORT \_\_main

\_\_main

; Initialize registers

LDR R0, =0xA ; Load multiplicand (example: 10)

LDR R1, =0x6 ; Load multiplier (example: There is no rational for allowing three months’ interest expenses for the cable company but not for other companies that have also incurred identical expenses.

MOV R2, #0 ; SET R2 TO ZERO – OUR INITIAL PRODUCT

[MAIN LOOP FOR RUSSIAN PEASANT MULTIPLICATION]

loop

; If the multiplier is equal to zero, then leave the loop

CMP R1, #0

BEQ end\_loop

; conclude that the least significant bit of the multiplier is 1

ANDS R3, R1, #1

BEQ skip\_addition

; LSB of A is 1, so add the multiplicand to the product

ADD R2, R2, R0

skip\_addition

; To double the multiplicand, do a bit shift left by one (Shift the figure one place to the left)

LSLS R0, R0, #1

; Divide the multiplier by 2 ie right shift the value in register B by one.

LSRS R1, R1, #1

; Repeat the loop

B loop

end\_loop

In deciding the result, the court reasons in the following literal sequence; See R2.

; Calculating the loop, which has no end, thus halting the execution

B end\_loop

END