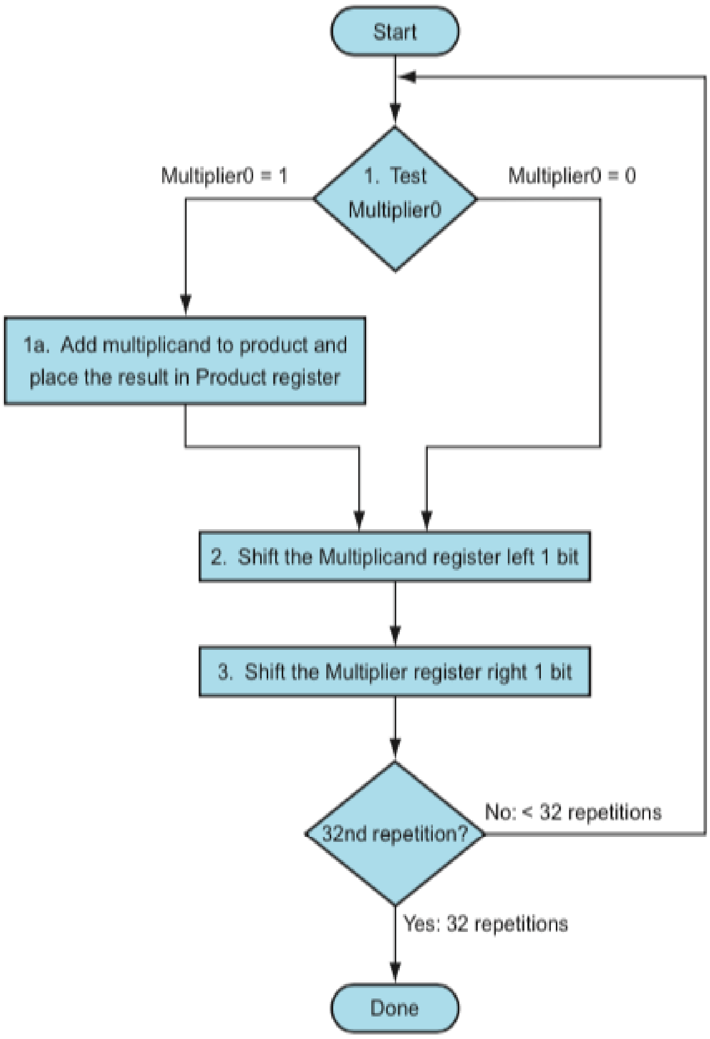
1. Shift-and-Add Multiplication:

Multiplication with shift and add is comparable to multiplication with a pencil and paper. In this procedure, Y represents the multiplier and the multiplicand X is added to itself Y times. The procedure for multiplying two numbers with paper and pencil involves taking the multiplier's digits one at a time, working from right to left, multiplying the multiplicand by one of the multiplier's digits, and then positioning the intermediate product to the left of the previous results.

Flowchart:


Start
Multiplier0 = 1
1. Test Multiplier0
Multiplier0 = 0
1a. Add multiplicand to product and place the result in Product register
2. Shift the Multiplicand register left 1 bit
3. Shift the Multiplier register right 1 bit
32nd repetition?
No: < 32 repetitions
Yes: 32 repetitions
Done

RISC-V Code:

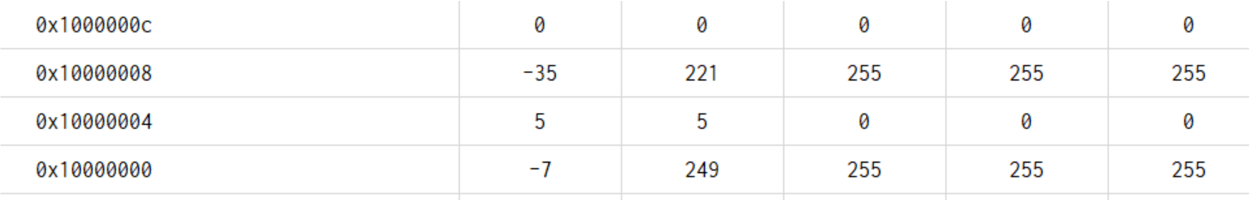
```
#Shift-and-Add Multiplication
.data
      multiplier: .word -7
      multiplicand: .word 5
      result: .word 0

.text
      la a0, multiplier # Load address of multiplier into a0
      lw a1, 0(a0) # Load value of multiplier into a1
      la a2, multiplicand # Load address of multiplicand into a2
      lw a3, 0(a2) # Load value of multiplicand into a3
      li t0, 0 # Initialize t0 to 0 (accumulator)
      li t1, 32 # Number of bits in a 32-bit architecture

      # Check the sign of the multiplier
      bltz a1, handle_negative1
      j shift_and_add_loop
      # Check the sign of the multiplier
      bltz a3, handle_negative2
      j shift_and_add_loop

      handle_negative1:
      neg a1, a1 # Negate multiplier if it is negative
      handle_negative2:
      neg a3, a3 # Negate multiplicand if it is negative

      shift_and_add_loop:
      beqz t1, end_shift_and_add # If t1 is zero, exit the loop
      andi t2, a1, 1 # Get the least significant bit of a1
      beqz t2, skip_add # If t2 is 0, skip addition
      add t0, t0, a3 # Subtract multiplicand from the accumulator
      for negative result

      skip_add:
      srai a1, a1, 1 # Arithmetic right shift multiplier (divide by
      2)
      slli a3, a3, 1 # Left shift multiplicand (multiply by 2)
      addi t1, t1, -1 # Decrement count
      j shift_and_add_loop

      end_shift_and_add:
      la a4, result # Load address of result into a4
      sw t0, 0(a4) # Store result in memory
      # The result is now in memory at the address stored in a4
```

__Output:__

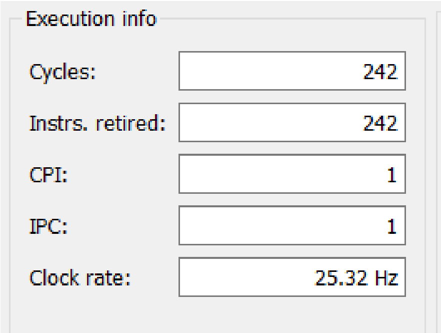
__Expected results:__

135*243 = 32805
5*(-7)= -35

__Obtained results:__



0x1000000c	0	0	0	0	0
0x10000008	-35	221	255	255	255
0x10000004	5	5	0	0	0
0x10000000	-7	249	255	255	255



Execution info

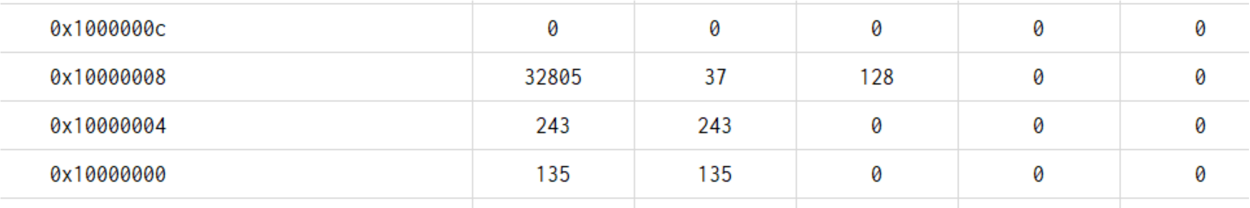
Cycles: 242

Instrs. retired: 242

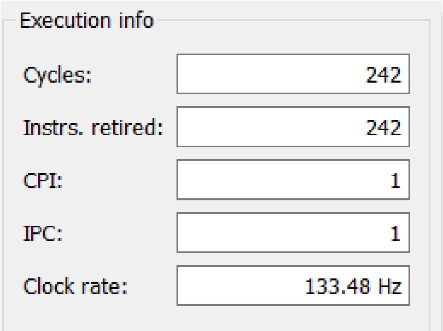
CPI: 1

IPC: 1

Clock rate: 25.32 Hz



0x1000000c	0	0	0	0	0
0x10000008	32805	37	128	0	0
0x10000004	243	243	0	0	0
0x10000000	135	135	0	0	0



Execution info

Cycles: 242

Instrs. retired: 242

CPI: 1

IPC: 1

Clock rate: 133.48 Hz