DOCUMENTATION

IMPLEMENTING BOOTH'S ALGORITHM

Project By:
Shruti Jha (2019274)

The project statement is - "Implement the Booth's algorithm for multiplying binary integers."
This document contains information about how the code developed in java implements the Booth's algorithm.

1) About Booth's Algorithm Implementation:

Booth's algorithm gives a procedure for multiplying binary integers in signed 2's complement representation in an efficient way.

The implementation of Booth's algorithm in the java code is done according to the following procedure:

- The *procedure for Booth's algorithm* is explained below:
    - Let M be the n-bit multiplicand.
    - Let Q be the n-bit multiplier.
    - Let Q1 be the last bit of Q.
    - Consider a 1-bit register Qo and initialise it to 0.
    - Consider a n-bit register A and initialize it to 0.
    - n: the step count = n-bit size of A
- The *conditions* are as follows:
    - If Q1Qo are the same, that is 00 or 11, then perform arithmetic right shift by 1-bit. Reduce step count by 1.
    - If Q1Qo is 01, then perform

      A=A+M,

      and then perform arithmetic right shift by 1-bit. Reduce step count by 1.
    - If Q1Qo is 10, then perform

A=A-M,

and then perform arithmetic right shift by 1-bit. Reduce step count by 1.

- The procedure is continued till step count does not reduce to 0.
- When step count becomes equal to 0, then the final product of the 2 integers is obtained. The product is 2n-bit, and is stored in AQ.

2) Machine Requirements:

- Should have at least 2^32 bits of memory.

3) Workflow:

- First the 2 integers to be multiplied are taken as inputs in decimal form. If they are in the input range mentioned below, then they are converted to their n-bit binary equivalent, otherwise the execution is aborted with an error message thrown.
- If the integer is positive, then it is simply converted to its binary form. But if the integer is negative, then the binary equivalent of its positive counterpart is calculated and then the 2's complement of this binary equivalent is taken. This is the final binary form of the negative integer.
- The integers are passed to a function (along with n) where size-n integer arrays A and Q (binary form of multiplier), 1-bit array Qo and Q1 are initialised as mentioned above.
- This function then checks the value of Q1Qo and according to it decides the operation to be done (as mentioned above). If it is 00 or 11, then the arrays A, Q and Qo are right shifted by 1-bit and step count reduced by 1. If it is 01, then the size-n array M (binary form of multiplicand) is added to the size-n array A. A,Q,Qo are right shifted by 1-bit and step count reduced by 1. And if it is 10, then the size-n array M' (binary form of the negative of multiplicand) is added to A. A,Q,Qo are right shifted by 1-bit and step count reduced by 1.

- When n becomes 0, then a new size-2n integer array is declared, containing array A and Q elements in order. This is the binary form of the final multiplication product.
- If the 1st bit(msb) of this binary form is 1, then the product is negative. In this case, first the 2's complement of this binary form is taken and then its decimal equivalent is calculated by another function. If the msb is 0, then the product is positive and simply the decimal equivalent of the binary form is calculated.

3) Input Format:

- Integers have to be provided in decimal form for input.

4) Input Range:

- The range of integers that are allowed as inputs to the algorithm is -2^31+1 to +2^31-1.
- This is because java stores integers in 32-bit 2's complement form.
- *Clarification*: Although the range of integers that can be represented in 32-bits in signed 2's complement form is -2^31 to +2^31-1, -2^31 cannot be allowed as multiplicand in the algorithm because the algorithm also uses negative of the multiplicand, and thus +2^31 will be required which can be represented in minimum 33-bits and not 32-bits.

5) Output Format:

- Each step of the algorithm is displayed in the order: step count, A, Q, Qo, action performed in that step.
- After step count becomes 0, the result of multiplication is provided in both 2n-bit binary form and its decimal equivalent.

6) Error Handled:

- If either of the integers provided as input does not lie in the input range mentioned above, then an error message is printed and execution is aborted.

Here is a screenshot of the output of of the documented java code implementing booth's algorithm: