CSL251 (2024-25-W)

Computer Organization and Architecture

(Supplementary note)

Souradyuti Paul

NYD Publishing Company

CSL251 (2024-25-W)

Computer Organization and Architecture

(Supplementary note)

Souradyuti Paul

NYD Publishing Company

NYD Publishing Company

Preface

Supplementary note.

Contents

I

Arithmetic Operations in a Computer

Arithmetic operations in a computer system are slightly tricky. We are habituated to two kinds of arithmetic system: decimal (base-10) and binary (base-2) systems.

In the binary system, we have four symbols: $0, 1, -, \cdot$, whereas in the decimal system, we have 12 symbols (e.g., $0, 1, 2, \ldots, 9, -, \cdot$). But in a computer system (CPU, Chipset, Bus, etc.) there are only two symbols available: 0 and 1.

In the computer system, the binary-to-decimal system works in the following way.

1. Suppose that only n-bit integers are considered.

2. Suppose that $b = (b_{n-1}b_{n-2} \ldots b_1 b_0)$ is an n-bit integer.

3. The binary-to-decimal conversion of b inside any electronic device works as follows: $-b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + \ldots + b_2 2^2 + b_1 2^1 + b_0 2^0$.

4. The binary-to-decimal conversion of b in our pen-and-paper system works as follows: $b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + \ldots + b_2 2^2 + b_1 2^1 + b_0 2^0$.

The binary integer $b = (b_{n-1}b_{n-2}\ldots b_1b_0)$ is called the twos complement representation if it is converted to the decimal integer as follows: $-b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + \ldots + b_2 2^2 + b_1 2^1 + b_0 2^0$.

Question 1 *Suppose that only n-bit integers are considered in our computer system which are represented as twos complement binary integers. Then how many of them are negative integers and how many of them are nonnegative integers?*

1.1 Negation, Addition, Subtraction, Multiplication and Division with Twos Complement Binary Integers

Negation algorithm. Suppose $a = -b$, where b is a binary integer: $b = (b_{n-1}b_{n-2}\ldots b_1b_0)$. How to compute a.
Answer. $a = (00\ldots1) + (\overline{b_{n-1}b_{n-2}\ldots b_1b_0})$. The MSB overflow bit is ignored.

Question 2 *Prove that the above negation algorithm is correct.*

Addition algorithm. Compute $c = a+b$, given a and b using an *adder* circuit. Ignore the overflow bit.

Subtraction algorithm. Suppose $c = a - b$. First, compute $d = -b$ using the *negationalgorithm* , then compute $c = a + d$ using the *addition* algorithm.

Question 3 *Prove that the above addition and subtraction algorithms are correct, taking into account all cases.*

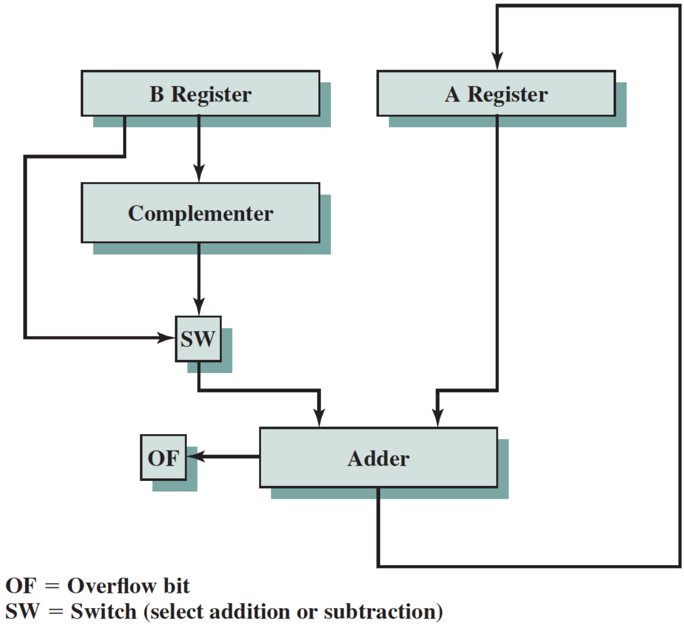

Figure I.1: Hardware circuit for addition and subtraction

I.2 Multiplication using Booth's algorithm

Key Observation-1. Convert the unsigned binary integer $b = 1111$ to a decimal integer. The answer is: $2^0 1 + 2^1 1 + 2^2 1 + 2^3 1$. How many operations were needed? 3 additions and 4 exponentiations. Can we reduce it? We can even compute it using 2 exponentiations and 1 subtraction. How $2^0 1 + 2^1 1 + 2^2 1 + 2^3 1 = 2^4 - 2^0$.

Key Observation-2: How to extend the bitsize of a negative integer

I.3 Booth's Algorithm.

How to extend an m-bit -ve integer into (m+1)-bit negative integer?

Example $\quad m = 5$ $\quad$ sign bit

$\boxed{\text{2's complement representation}}$

$$A = \begin{array}{|c|c|c|c|c|} \hline 1 & 0 & 1 & 0 & 1 \\ \hline \end{array}$$
$$\begin{array}{ccccc} 4 & 3 & 2 & 1 & 0 \end{array}$$

- What is the decimal value of A?

Ans: $-2^4\cdot 1 + 2^3\cdot 0 + 2^2\cdot 1 + 2^1\cdot 0 + 2^0\cdot 1 = -16 + 0 + 4 + 0 + 1 = -11$

- What is the 6-bit extension of A?
- What is the 7-bit extension of A?

sign bit

$$\begin{array}{|c|c|c|c|c|c|} \hline 1 & 1 & 0 & 1 & 0 & 1 \\ \hline \end{array} \leftarrow \text{6-bit extension of A}$$

sign bit

$$\begin{array}{|c|c|c|c|c|c|c|} \hline 1 & 1 & 1 & 0 & 1 & 0 & 1 \\ \hline \end{array} \leftarrow \text{7-bit extension of A}$$

Check out that both the above decimal conversions are producing −11. [The msb is the ~~msb~~ sign-bit as always]

Figure I.2: How to extend the bitsize of a negative integer

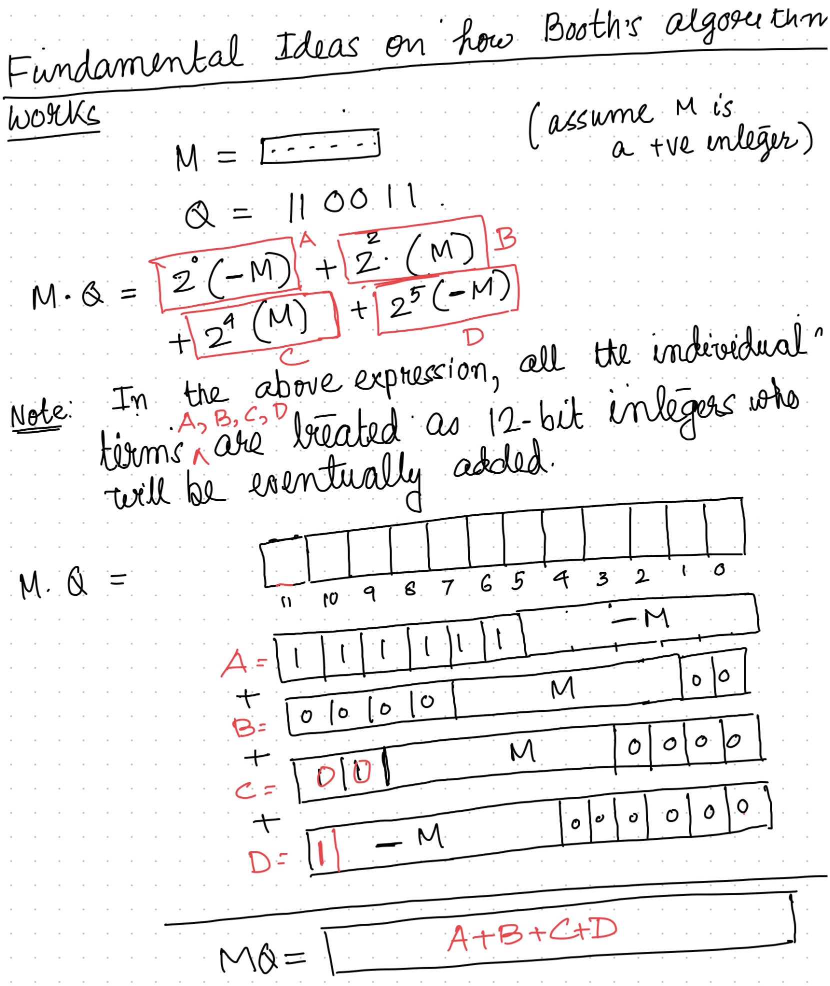

Figure I.3: Basic Idea of Booth's Algorithm

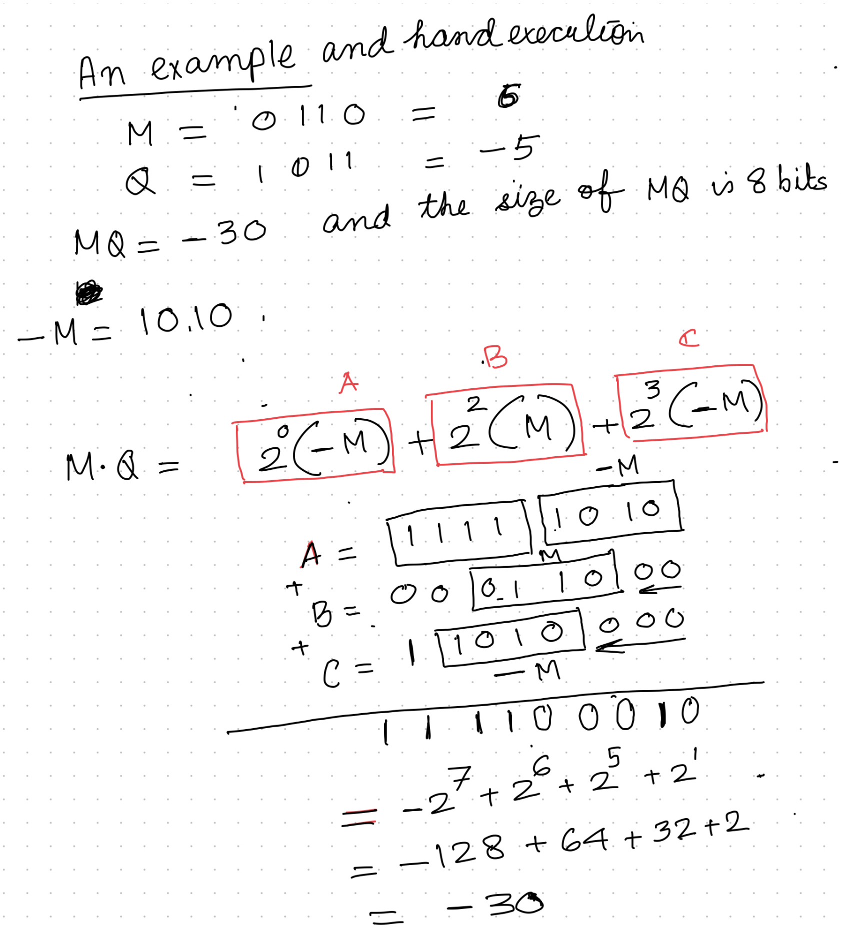

Figure I.4: An example and hand execution of Booth's algorithm

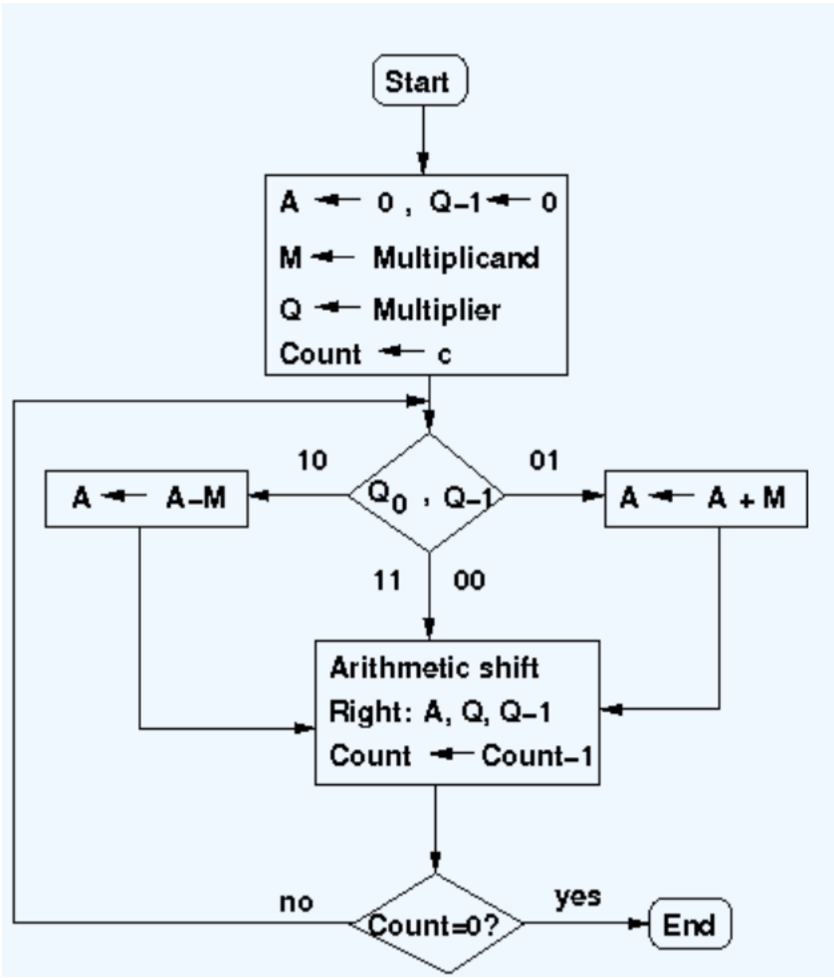

Figure I.5: Flowchart: Booth's Algorithm

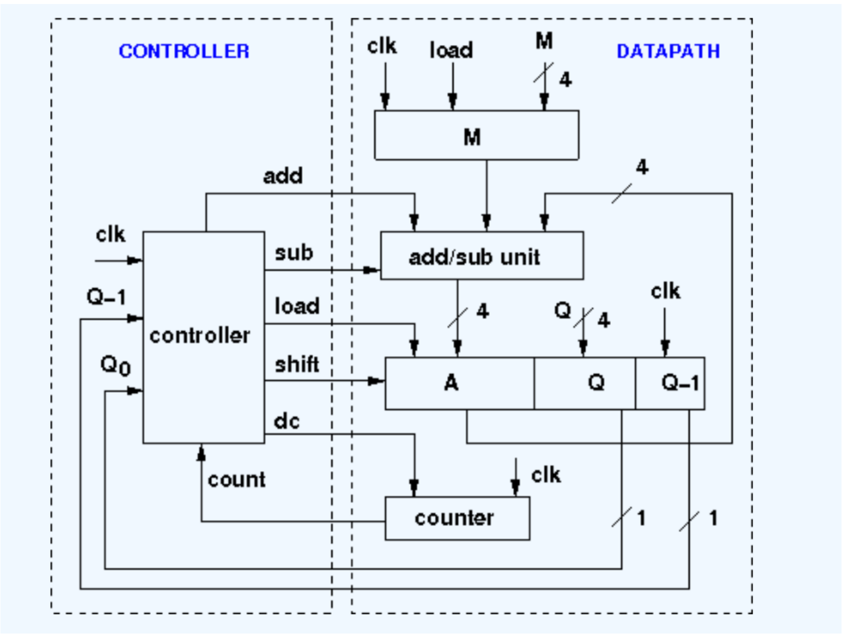

Figure I.6: Booth's Algorithm: Hardware Implementation

SUPPLEMENTARY NOTE.