Digital Logic Design: a rigorous approach ©

Chapter 5: Binary Representation

Guy Even Moti Medina

School of Electrical Engineering Tel-Aviv Univ.

April 10, 2012

Book Homepage:
http://www.eng.tau.ac.il/~guy/Even-Medina

Division and Modulo

Suppose we divide a natural number a by a positive natural number b. If a is divisible by b, then we obtain a quotient q that is a natural number. Namely, $a = q \cdot b$, with $q \in \mathbb{N}$.
However, we also want to consider the case that a is not divisible by b. In this case, division is defined as follows. Consider the two consecutive integer multiples of b that satisfy

$$q \cdot b \leq a < (q+1) \cdot b.$$

The quotient is defined to be q. The remainder is defined to be $r \triangleq a - q \cdot b$. Clearly, $0 \leq r < b$. Note that the quotient q simply equals $\left\lfloor \frac{a}{b} \right\rfloor$.
Notation: Let $(a \bmod b)$ denote the remainder obtained by dividing a by b.

Examples

1. $3 \bmod 5 = 3$ and $5 \bmod 3 = 2$.
2. $999 \bmod 10 = 9$ and $123 \bmod 10 = 3$.
3. $a \bmod 2$ equals 1 if a is odd, and 0 if a is even. Indeed, if a is even, then $a = 2x$, and then
   $a - 2 \cdot \left\lfloor \frac{a}{2} \right\rfloor = a - 2 \cdot \left\lfloor \frac{2x}{2} \right\rfloor = a - 2x = 0.$
   If a is odd, then $a = 2x + 1$, and then
   $a - 2 \cdot \left\lfloor \frac{a}{2} \right\rfloor = a - 2 \cdot \left\lfloor \frac{2x+1}{2} \right\rfloor = a - 2 \left\lfloor x + \frac{1}{2} \right\rfloor = a - 2x = 1.$
4. $a \bmod b \geq 0$. Indeed, $b \cdot \left\lfloor \frac{a}{b} \right\rfloor \leq b \cdot \frac{a}{b} = a$. Therefore, $a - b \cdot \left\lfloor \frac{a}{b} \right\rfloor \geq a - a = 0$.
5. $a \bmod b \leq b - 1$. Let $q = \left\lfloor \frac{a}{b} \right\rfloor$. This means that $b \cdot q \leq a < b \cdot q + b$. Hence, $a - b \cdot \left\lfloor \frac{a}{b} \right\rfloor = a - b \cdot q < a - (a - b) = b$, which implies that $a \bmod b < b$. Since $a \bmod b$ is an integer, we conclude that $a \bmod b \leq b - 1$.

In decimal numbers, the basic unit of information is a digit, i.e., a number in the set $\{0, 1, \ldots, 9\}$. In digital computers, the basic unit of information is a bit.

Definition

A bit is an element in the set $\{0, 1\}$.

Since bits are the basic unit of information, we need to represent numbers using bits. How is this done? Numbers are represented in many ways in computers: binary representation, BCD, floating-point, two's complement, sign-magnitude, etc. The most basic representation is binary representation. To define binary representation, we first need to define binary strings.

Definition

A binary string is a finite sequence of bits.

There are many ways to denote strings: as a sequence $\{A_i\}_{i=0}^{n-1}$, as a vector $A[0 : n-1]$, or simply by $\vec{A}$ if the indexes are known. We often use $A[i]$ to denote A_i.

- Let us consider the string $\{A_i\}_{i=0}^{3}$, where $A_0 = 1$, $A_1 = 1$, $A_2 = 0$, $A_3 = 0$. We often wish to abbreviate and write $A[0:3] = 1100$. This means that when we read the string 1100, we assign the indexes 0 to 3 to this string from left to right.
- Consider the string $A[0:5] = 100101$. The string $\vec{A}$ has 6 bits, hence $n = 6$. The notation $A[0:5]$ is zero based, i.e., the first bit in $\vec{A}$ is $A[0]$. Therefore, the third bit of $\vec{A}$ is $A[2]$ (which equals 0).

A basic operation that is applied to strings is called concatenation. Given two strings $A[0:n-1]$ and $B[0:m-1]$, the concatenated string is a string $C[0:n+m-1]$ defined by

$$C[i] \triangleq \begin{cases} A[i] & \text{if } 0 \le i < n, \\ B[i-n] & \text{if } n \le i \le n+m-1. \end{cases}$$

We denote the operation of concatenating string by $\circ$, e.g., $\vec{C} = \vec{A} \circ \vec{B}$.

Examples of concatenation of strings. Let $A[0:2] = 111$, $B[0:1] = 01$, $C[0:1] = 10$, then:

$$\vec{A} \circ \vec{B} = 111 \circ 01 = 11101\,,$$
$$\vec{A} \circ \vec{C} = 111 \circ 10 = 11110\,,$$
$$\vec{B} \circ \vec{C} = 01 \circ 10 = 0110\,,$$
$$\vec{B} \circ \vec{B} = 01 \circ 01 = 0101\,.$$

Let $i \leq j$. Both $A[i:j]$ and $A[j:i]$ denote the same sequence $\{A_k\}_{k=i}^{j}$. However, when we write $A[i:j]$ as a string, the leftmost bit is $A[i]$ and the rightmost bit is $A[j]$. On the other hand, when we write $A[j:i]$ as a string, the leftmost bit is $A[j]$ and the rightmost bit is $A[i]$.

Example

The string $A[3:0]$ and the string $A[0:3]$ denote the same 4-bit string. However, when we write $A[3:0] = 1100$ it means that $A[3] = A[2] = 1$ and $A[1] = A[0] = 0$. When we write $A[0:3] = 1100$ it means that $A[3] = A[2] = 0$ and $A[1] = A[0] = 1$.

least/most significant bits

Definition

The least significant bit of the string $A[i:j]$ is the bit $A[k]$, where $k \stackrel{\triangle}{=} \min\{i,j\}$. The most significant bit of the string $A[i:j]$ is the bit $A[\ell]$, where $\ell \stackrel{\triangle}{=} \max\{i,j\}$.

The abbreviations LSB and MSB are used to abbreviate the least significant bit and the most significant bit, respectively.

LSB/MSB - examples

1. The least significant bit (LSB) of $A[0:3] = 1100$ is $A[0] = 1$. The most significant bit (MSB) of $\vec{A}$ is $A[3] = 0$.
2. The LSB of $A[3:0] = 1100$ is $A[0] = 0$. The MSB of $\vec{A}$ is $A[3] = 1$.
3. The least significant and most significant bits are determined by the indexes. In our convention, it is not the case that the LSB is always the leftmost bit. Namely, if $i \leq j$, then LSB in $A[i:j]$ is the leftmost bit, whereas in $A[j:i]$, the leftmost bit is the MSB.

Binary Representation

We are now ready to define the binary number represented by a string $A[n-1:0]$.

Definition

The natural number, a, represented in binary representation by the binary string $A[n-1:0]$ is defined by

$$a \triangleq \sum_{i=0}^{n-1} A[i] \cdot 2^i.$$

In binary representation, each bit has a weight associated with it. The weight of the bit $A[i]$ is 2^i.

Notation

Consider a binary string $A[n-1:0]$. We introduce the following notation:

$$\langle A[n-1:0] \rangle \triangleq \sum_{i=0}^{n-1} A[i] \cdot 2^i .$$

To simplify notation, we often denote strings by capital letters (e.g., A, B, S) and we denote the number represented by a string by a lowercase letter (e.g., a, b, and s).

Examples

Consider the strings: $A[2:0] \triangleq 000, B[3:0] \triangleq 0001$, and $C[3:0] \triangleq 1000$. The natural numbers represented by the binary strings A, B and C are as follows.

$$\begin{aligned}
\langle A[2:0]\rangle &= A[0]\cdot 2^0 + A[1]\cdot 2^1 + A[2]\cdot 2^2 \\
&= 0\cdot 2^0 + 0\cdot 2^1 + 0\cdot 2^2 = 0\,, \\
\langle B[3:0]\rangle &= B[0]\cdot 2^0 + B[1]\cdot 2^1 + B[2]\cdot 2^2 + B[3]\cdot 2^3 \\
&= 1\cdot 2^0 + 0\cdot 2^1 + 0\cdot 2^2 + 0\cdot 2^3 = 1\,, \\
\langle C[3:0]\rangle &= C[0]\cdot 2^0 + C[1]\cdot 2^1 + C[2]\cdot 2^2 + C[3]\cdot 2^3 \\
&= 0\cdot 2^0 + 0\cdot 2^1 + 0\cdot 2^2 + 1\cdot 2^3 = 8\,.
\end{aligned}$$

Leading Zeros

Consider a binary string $A[n-1:0]$. Extending $\vec{A}$ by leading zeros means concatenating zeros in indexes higher than $n-1$. Namely,

1. extending the length of $A[n-1:0]$ to $A[m-1:0]$, for $m > n$, and
2. defining $A[i] = 0$, for every $i \in [m-1:n]$.

Leading Zeros

The following lemma states that extending a binary string by leading zeros does not change the number it represents in binary representation.

Lemma

Let $m > n$. If $A[m-1:n]$ is all zeros, then $\langle A[m-1:0]\rangle = \langle A[n-1:0]\rangle$.

Example

Consider $C[6:0] = 0001100$ and $D[3:0] = 1100$. Note that $\langle C\rangle = \langle D\rangle = 12$. Since the leading zeros do not affect the value represented by a string, a natural number has infinitely many binary representations.

Representable Ranges

The following lemma bounds the value of a number represented by a k-bit binary string.

Lemma

Let $A[k-1:0]$ denote a k-bit binary string. Then,
$0 \le \langle A[k-1:0] \rangle \le 2^k - 1.$

What is the largest number representable by the following number of bits: (i) 8 bits, (ii) 10 bits, (iii) 16 bits, (iv) 32 bits, and (v) 64 bits?

Computing a Binary Representation

Fix k the number of bits (i.e., length of binary string).
Goals:

1. show how to compute a binary representation of a natural number using k bits.
2. prove that every natural number has a unique binary representation that uses k bits.

Algorithm $BR(x, k)$ for computing a binary representation is specified as follows:

Inputs: $x \in \mathbb{N}$ and $k \in \mathbb{N}^+$, where x is a natural number for which a binary representation is sought, and k is the length of the binary string that the algorithm should output.

Output: The algorithm outputs "fail" or a k-bit binary string $A[k-1:0]$.

Functionality: The relation between the inputs and the output is as follows:

1. If $0 \leq x < 2^k$, then the algorithm outputs a k-bit string $A[k-1:0]$ that satisfies $x = \langle A[k-1:0] \rangle$.
2. If $x \geq 2^k$, then the algorithm outputs "fail".

Algorithm 1 $BR(x,k)$ - An algorithm for computing a binary representation of a natural number a using k bits.

1. Base Cases:
   1. If $x \geq 2^k$ then return (fail).
   2. If $k = 1$ then return (x).
2. Reduction Rule:
   1. If $x \geq 2^{k-1}$ then return $(1 \circ BR(x - 2^{k-1}, k-1))$.
   2. If $x \leq 2^{k-1} - 1$ then return $(0 \circ BR(x, k-1))$.

example: execution of $BR(2,1)$ and $BR(7,3)$

Theorem

If $x \in \mathbb{N}$, $k \in \mathbb{N}^+$, and $x < 2^k$, then algorithm $BR(x,k)$ returns a k-bit binary string $A[k-1:0]$ such that $\langle A[k-1:0] \rangle = x$.

unique binary representation

corollary

Every positive integer x has a binary representation by a k-bit binary string if $k \geq \lfloor \log_2(x) \rfloor + 1$.

unique binary representation

corollary

Every positive integer x has a binary representation by a k-bit binary string if $k \geq \lfloor \log_2(x) \rfloor + 1$.

Theorem (unique binary representation)

The binary representation function $\langle\rangle_k : \{0,1\}^k \to \mathbb{N}$ *defined by*

$$\langle A[k-1:0] \rangle_k \triangleq \sum_{i=0}^{k-1} A[i] \cdot 2^i$$

is a bijection (i.e., one-to-one and onto) from $\{0,1\}^k$ *to* $\{0,\ldots,2^k-1\}$.

We claim that when a natural number is multiplied by two, its binary representation is "shifted left" while a single zero bit is padded from the right. That property is summarized in the following lemma.

Lemma

Let $a \in \mathbb{N}$. Let $A[k-1:0]$ be a k-bit string such that $a = \langle A[k-1:0]\rangle$. Let $B[k:0] \triangleq A[k-1:0] \circ 0$, then

$$2 \cdot a = \langle B[k:0]\rangle.$$

Example

$\langle 1000\rangle = 2 \cdot \langle 100\rangle = 2^2 \cdot \langle 10\rangle = 2^3 \cdot \langle 1\rangle = 8.$