COMP250: Binary Numbers

Jérôme Waldispühl
School of Computer Science
McGill University
(Slides from M. Langer)

Decimal (base 10)

Digits = { 0, 1, 2, 3, 4, 5, 7, 8, 9 }

Example of numerals: 11, 923, 5548, etc.

$$11 = 1 * 10^1 + 1 * 10^0$$
$$923 = 9 * 10^2 + 2 * 10^1 + 3 * 10^0$$
$$5548 = 5 * 10^3 5 * 10^2 + 4 * 10^1 + 8 * 10^0$$

$$m = \sum_{i=0} d[i] * 10^i$$

digit

Binary (base 2)

Bits = { 0, 1 }

Example of numerals: 11, 101, 1010, etc.

$$11 = 1 * 2^1 + 1 * 2^0$$
$$101 = 1 * 2^2 + 0 * 2^1 + 1 * 2^0$$
$$1010 = 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 0 * 2^0$$

$$m = \sum_{i=0} b[i] * 2^i$$

bit

Relationship

Decimal	Binary
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000

Fixed size representation

Decimal	Binary
0	00000000
1	00000001
2	00000010
3	00000011
4	00000100
5	00000101
6	00000110
7	00000111
8	00001000

Fixed number of bits (typically 8, 16, 32, 64…).

8 bits is called “byte”.

Conversion

- How to convert from binary to decimal?

$$
\begin{aligned}
(11010)_2 &= 1 * 2^4 + 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 0 * 2^0 \\
&= 16 + 8 + 0 + 2 + 0 \\
&= 26
\end{aligned}
$$

Note: You need to know powers of 2…

- How to convert from decimal to binary?

$$(241)_{10} = ???$$

Operations on decimals

Use this property of any positive integer m：

$$m = m\%10 + (m/10) * 10$$

(integer) division by 10 = dropping rightmost digit

Ex: 238/10 = 23

Multiplication by 10 = shifting left by one digit

Ex: 23*10 = 230

Remainder of integer division by 10 = rightmost digit

Ex: 238%10 = 8

Operations on binary

Same property holds for binary:

$$m = m\%2 + (m/2) * 2$$

Example:

$$\begin{aligned} m &= (1011)_2 \\ m/2 &= (0101)_2 \\ (m/2) * 2 &= (1010)_2 \\ m\ \%\ 2 &= (0001)_2 \end{aligned}$$

Decimal⟶Binary (Algorithm)

Algorithm decimal2binary(m)
Input: a decimal *m*
Output: a binary *b*
i⟵0
while m>0 **do**
    b[i] ⟵m%2
    m ⟵m/2
    i ⟵i+1

Decimal⟶Binary (Example)

i	m/2	m%2 **(b[i])**
	241	
0	120	1
1	60	0
2	30	0
3	15	0
4	7	1
5	3	1
6	1	1
7	0	1
8	-	-

Answer:

b[ ] = …011110001

Why is the algorithm working?

$$m = \underbrace{m/2 * 2}_{(\ldots b[3]b[2]b[1])_2} + m \,\%\, 2$$

$$\underbrace{(\ldots b[3]b[2]b[1]0)_2}_{} \qquad \underbrace{(b[0])_2}_{}$$

$$\underbrace{(\ldots b[3]b[2]b[1]b[0])_2}_{}$$

Additions

Decimal	**Binary**
0+1=1	0+1=1
1+1=2	1+1=10
1+2=3	1+10=11

Additions

Decimal	Binary
26	11010
$+\quad 15$	$+\quad 01111$
$=\quad 41$	$=\quad ?????$

Addition in binary

$$
\begin{array}{cccccc}
 & 1 & 1 & 0 & 1 & 0 \\
+ & 0 & 1 & 1 & 1 & 1 \\
\hline
= & ? & ? & ? & ?_{0} & 1
\end{array}
$$

Addition in binary

$$
\begin{array}{cccccc}
 & 1 & 1 & 0 & 1 & 0 \\
+ & 0 & 1 & 1 & 1 & 1 \\
\hline
= & ? & ? & ?_{1} & 0_{0} & 1
\end{array}
$$

Addition in binary

$$
\begin{array}{cccccc}
 & 1 & 1 & 0 & 1 & 0 \\
+ & 0 & 1 & 1 & 1 & 1 \\
\hline
= & ? & ?_{1} & 0_{1} & 0_{0} & 1
\end{array}
$$

Addition in binary

$$
\begin{array}{cccccc}
 & 1 & 1 & 0 & 1 & 0 \\
+ & 0 & 1 & 1 & 1 & 1 \\
\hline
= & ?_{1} & 1_{1} & 0_{1} & 0_{0} & 1
\end{array}
$$

Addition in binary

$$
\begin{array}{cccccccc}
 & & & 1 & 1 & 0 & 1 & 0 \\
+ & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\
\hline
= & 1 & 0_{1} & 1_{1} & 0_{1} & 0_{0} & 1 &
\end{array}
$$

Addition in binary

$$\begin{array}{rcccccc} & 1 & 1 & 0 & 1 & 0 & = 26 \\ + & 0 & 0 & 1 & 1 & 1 & = 15 \\ \hline = & 1 & 0 & 1 & 0 & 0 & 1 \quad = 41 \end{array}$$

$$\begin{array}{cccccc} 2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \\ \hline 1 & 0 & 1 & 0 & 0 & 1 \end{array} = 2^5 + 2^3 + 2^0 = 32 + 8 + 1 = 41$$

Operation in binary

Recall grade-school algorithm for addition, subtraction, multiplication, and division.

There is nothing special about base 10.

These algorithms work for binary (base 2), and work for other bases too!

Representation size

$$m = \sum_{i=0}^{N-1} b[i] * 2^i$$

What is the relationship between m and *N* ?

(How many bits *N* do we need to represent a positive integer *m*?)

Upper bound

$$\sum_{i=0}^{N-1} 2^i = 1 + 2 + 4 + \cdots + 2^{N-1} = 2^N - 1$$

(This is a special case of $\sum_{i=0}^{N-1} x^i = \frac{x^N - 1}{x-1}$ where $x = 2$)

$$\begin{aligned} m = \sum_{i=0}^{N-1} b[i] * 2^i \quad &\leq \quad \sum_{i=0}^{N-1} 1 * 2^i \\ &= \quad 2^N - 1 \\ &< \quad 2^N \end{aligned}$$

$\log_2 m < N$ (apply log on both sides)

Lower bound

We can assume that $N - 1$ **is the index** i **of the leftmost bit** b **[i] such that** $b[i] = 1$ (we ignore leftmost 0's: 00001101).

$$m = \sum_{i=0}^{N-1} b[i] * 2^i \quad \geq \quad 2^{N-1}$$

$$\log_2 m \quad \geq \quad N - 1$$

$$\log_2 m + 1 \quad \geq \quad N$$

How many bit do we **need**?

$$\log_2 m < N \leq (\log_2 m) + 1$$

Answer: The largest integer less than or equal to $(\log_2 m) + 1$.

We write it as $N = \lfloor (\log_2 m) + 1 \rfloor$ (a.k.a. ”floor” that means “round down”)

Examples

m (decimal)	m (binary)	$N = \lfloor(\log_2 m) + 1\rfloor$
0	0	-
1	1	1
2	10	2
3	11	2
4	100	3
5	101	3
6	110	3
7	111	4
8	1000	4
9	1001	4

To think about…

- How are negative integers represented?
- How many bits are used to represent int, short, long in a computer?
- How are non-integers (fractional numbers) represented?
- How are characters represented?