# COMP 250

# Lecture 2

# Binary number representations

# Sept. 11, 2017

# Base 10 (decimal) “digits” {0,1,2,…, 9}

e.g. $$5819 = 5 * 10^3 + 8 * 10^2 + 1 * 10^1 + 9 * 10^0$$

# Base 10 (decimal) "digits" {0,1,2,…, 9}

e.g. $$5819 = 5 * 10^3 + 8 * 10^2 + 1 * 10^1 + 9 * 10^0$$

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

digits

# Base 2 (binary) "bits" {0, 1}

e.g.

$$(11010)_2 = 1 * 2^4 + 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 0 * 2^0$$

Base 2 (binary) "bits" {0, 1}

e.g.

$$(11010)_2 = 1 * 2^4 + 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 0 * 2^0$$

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

| decimal | binary |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 10 |
| 3 | 11 |
| 4 | 100 |
| 5 | 101 |
| 6 | 110 |
| 7 | 111 |
| 8 | 1000 |
| 9 | 1001 |
| 10 | 1010 |
| 11 | 1011 |
| : | : |

| decimal | binary |
|---|---|
| 0 | 00000000 |
| 1 | 00000001 |
| 2 | 00000010 |
| 3 | 00000011 |
| 4 | 00000100 |
| 5 | 00000101 |
| 6 | 00000110 |
| 7 | 00000111 |
| 8 | 00001000 |
| 9 | 00001001 |
| 10 | 00001010 |
| 11 | 00001011 |
| : | : |

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

8 bits is called a "byte".

How to convert *from binary to decimal* ?

You need to know the powers of 2.

| $i$ | $2^i$ |
|---|---|
| 0 | 1 |
| 1 | 2 |
| 2 | 4 |
| 3 | 8 |
| 4 | 16 |
| 5 | 32 |
| 6 | 64 |
| 7 | 128 |
| 8 | 256 |
| 9 | 512 |
| 10 | 1024 |
| 11 | 2048 |
| : | : |

# Converting from binary to decimal

$$(11010)_2 = 1 * 2^4 + 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 0 * 2^0$$

$$= 16 + 8 + 0 + 2 + 0$$

$$= 26$$

# How to convert from decimal to binary?

$$(241)_{10} = ( \ ? \ )_2$$

I will present an algorithm for doing so.

Use this property of any positive integer $m$ :

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

Use this property of any positive integer $m$ :

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

(integer) division by 10 $=$ dropping rightmost digit

238 / 10 = 23

Multiplication by 10 $=$ shifting left by one digit

23 * 10 = 230

Remainder of integer division by 10 $=$ rightmost digit

238 % 10 = 8

The same property for binary.

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

The same property for binary.

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

e.g.

$$m = (10011)_2$$

$$m/2 = (1001)_2$$

$$(m/2) * 2 = (10010)_2$$

$$m \% 2 = (00001)_2$$

Algorithm to convert a given $m$ from decimal to binary where ... $b[3]\ b[2]\ b[1]\ b[0]$ are the bits of the binary representation.

$$
\begin{array}{l}
i \leftarrow 0 \\
\textbf{while } m > 0 \textbf{ do} \\
\quad b[i] \leftarrow m \;\%\; 2 \\
\quad m \leftarrow m \;/\; 2 \\
\quad i \leftarrow i + 1 \\
\textbf{end while}
\end{array}
$$

Sometimes I will use the arrow notation rather than equals sign for assigning a value to a variable. (No significance to this.)

Example:  Convert 241 from decimal to binary

| $i$ | | $b[i\ ]$ |
|---|---|---|
| | 241 | |
| 0 | 120 | 1 |

241 = (241/2)*2 + 1

Example: Convert 241 from decimal to binary

| $i$ | | $b[i]$ |
|---|---|---|
| | 241 | |
| 0 | 120 | 1 |
| 1 | 60 | 0 |

120 = (120/2)*2 + 0

Example: Convert 241 from decimal to binary

| $i$ | | $b[i\,]$ |
|---|---|---|
| | 241 | |
| 0 | 120 | 1 |
| 1 | 60 | 0 |
| 2 | 30 | 0 |

60 = (60/2)*2 + 0

Example: Convert 241 from decimal to binary

| $i$ | | $b[i\ ]$ |
|---|---|---|
| | 241 | |
| 0 | 120 | 1 |
| 1 | 60 | 0 |
| 2 | 30 | 0 |
| 3 | 15 | 0 |
| 4 | 7 | 1 |

15 = (15/2)*2 + 1

Example: Convert 241 from decimal to binary

| $i$ | | $b[i\,]$ |
|---|---|---|
| | 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 | 0 | 0 |
| 9 : | | |

Answer:

$b$[ ] = …011110001

Recall:

$$m \;=\; m/2 * 2 \;+\; m \,\%\, 2$$

Why the algorithm works:

$$m \;=\; (\ldots b[3]\; b[2]\; b[1]\; b[0]\;)_2$$

$$m/2 \;=\; (\ldots b[3]\; b[2]\; b[1]\;)_2$$

$$(m/2) * 2 \;=\; (\ldots b[3]\; b[2]\; b[1]\; 0\;)_2$$

$$m \,\%\, 2 \;=\; b[0]$$

# Addition in binary

$$\begin{array}{r} 11010 \\ +\quad 1111 \\ \hline ? \end{array} \qquad\qquad \begin{array}{r} 26 \\ +\ \ 15 \\ \hline 41 \end{array}$$

# Addition in binary

| | | |
|---|---|---|
| carry | 11110 | |
| | 11010 | 26 |
| + | 1111 | + 15 |
| | 101001 | 41 |

# Grade school arithmetic  in binary

Recall addition, subtraction, multiplication, division.

There is nothing special about base 10.

*These algorithms work for binary (base 2) too.*

*Indeed they work for other bases too (Assignment 1).*

# How many bits $N$ do we need to represent a positive integer $m$?

$$m = \sum_{i=0}^{N-1} b_i \, 2^i$$

What is the relationship between $m$ and $N$?

# To answer this question, we use:

$$\sum_{i=0}^{N-1} 2^i = 1 + 2 + 4 + 8 + 16 + \cdots$$

$$\ldots + 2^{N-3} + 2^{N-2} + 2^{N-1}$$

$$= 2^N - 1$$ (see next slide)

$$\sum_{i=0}^{N-1} 2^i \quad = \quad 2^N - 1$$

is a special case of

$$1 + x + x^2 + x^3 + \ldots \quad x^{N-1} \quad = \quad \frac{x^N - 1}{x - 1}$$

where $x = 2$. (See lecture notes for a derivation of this.)

How many bits $N$ do we need to represent an integer $m$?

$$m = \sum_{i=0}^{N-1} b_i \, 2^i \quad \leq \sum_{i=0}^{N-1} 1 * 2^i$$

$$= \quad 2^N - 1 \qquad \text{(previous slide)}$$

$$< \quad 2^N$$

Take the log (base 2) of both sides:

$$\boxed{log_2 m < N}$$

How many bits $N$ do we need to represent an integer $m$ ?

$$m = \sum_{i=0}^{N-1} b_i \, 2^i$$

We can assume that $N-1$ is the index $i$ of the leftmost bit $b_i$ such that $b_i = 1$.

e.g. We ignore leftmost 0's of $(\ldots 00000010011)_2$

$$m = \sum_{i=0}^{N-1} b_i \, 2^i \qquad \geq \quad 2^{N-1}$$

Take the log (base 2) of both sides:

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

Rewrite as:

$$N \quad \leq \ (log_2 \, m) + \ 1$$

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

Q: How many bits $N$ do we ***need*** to represent $m$?

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

We write:

$$N = floor( (log_2 m) + 1)$$

where "floor" means "round down".

| $m$ (decimal) | $m$ (binary) | $N = floor\,(1 + \,log_2\, m)$ |
|---|---|---|
| 0 | 0 | |
| 1 | 1 | 1 |
| 2 | 10 | 2 |
| 3 | 11 | 2 |
| 4 | 100 | 3 |
| 5 | 101 | 3 |
| 6 | 110 | 3 |
| 7 | 111 | 3 |
| 8 | 1000 | 4 |
| 9 | 1001 | 4 |
| 10 | 1010 | 4 |
| 11 | 1011 | 4 |
| : | : | : |

# Other number representations

(covered in detail in COMP 273 –
see my lecture notes if you are curious)

Q: How are negative integers represented ?

Q: How many bits are used to represent `int`, `short`, `long` in a computer?
(These include negative valued integers)

Q: How are non-integers (fractional numbers) represented ?

# TODO

- Install Eclipse. Make sure it runs. But you don't need to start using it yet.

http://www.eclipse.org/downloads/packages/eclipse-ide-java-developers/oxygenr

There will be a basic tutorial next class:

(Wed for Sec. 001 and Thurs for Sec. 002)