Computer (Literacy) Skills

Number representations and memory

Lubomír Bulej
KDSS MFF UK

Number representations? What for?

Recall: computer works with binary numbers

- Groups of zeroes and ones
  - 8 bits (byte), 16 bits (2 bytes), … , 64 bits (8 bytes), ...
  - Number of bits determines range of numbers
  - If a number does not fit $\Rightarrow$ overflow!
- So far we have used **unsigned integer** numbers
  - Natural numbers with *zero* included

People usually need other kinds of numbers

- Positive, negative, fractions, ...
- Groups of zeroes and ones… again?

We can interpret bits in different ways

- Some interpretations more useful than others

Positive/negative integer numbers

Things to consider

- Distinguishing positive/negative numbers
- Difficulty working with the numbers (in HW)
  - Arithmetic, negation, detecting overflow
  - Extending/truncating fixed-size representation

Sign and magnitude representation

- Explicit sign bit (where to put it?)
- N bits $\Rightarrow \{ -(2^{N-1}), \ldots, \mathbf{-0}, \mathbf{+0}, \ldots, 2^{N-1} \}$

Biased representation

- Implicit sign bit (only for bias of $2^{N-1} - 1$)
- N bits with bias B $\Rightarrow \{ -B, \ldots, \mathbf{0}, \ldots, 2^{N} - 1 - B \}$

Positive/negative integer numbers

One's complement representation

- Implicit sign bit, symmetric range
- Negation is "flip all bits"
- N bits ⇒ $\{ -(2^{N-1}), \ldots, -0, +0, \ldots, 2^{N-1} \}$

Two's complement representation

- Implicit sign bit, asymmetric range
- Negation is "**flip all bits, then add 1**"
- N bits ⇒ $\{ -(2^{N-1}), \ldots, 0, \ldots, 2^{N-1} - 1 \}$
  - $\mathbf{b_{N-1} \times -(2^{N-1})} + b_{N-2} \times 2^{N-2} + \ldots + b_1 \times 2^1 + b_0 \times 2^0$
- Modular arithmetic!
  - Subtract by adding in unsigned arithmetic

Extending/truncating numbers

General principle

- Ensure that interpreting the extended/truncated representation gives the same number
- Truncation can result in **loss of information!**

Unsigned integers

- Extend with zero bits to the left, truncation trivial

Sign and magnitude representation

- Strip sign, extend/truncate as unsigned, set sign

One's and two's complement representations

- Extend using the value of the highest bit, truncation trivial

Representing fractional numbers

Fixed point representation

- Decimal analogy: numbers 0 … 99 divided by 10 allow representing 0.0, 0.1, 0.2, …, 9.9
- In binary, the fractional part of a number is the sum of negative powers of 2
- Works also with two's complement

Example: 4.4 fixed-point representation

- Integral part: 0, 1, …, 15
- Fractional part: 0×0.0625, …, 15×0.0625 = 0.9375

$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$	$2^{-1} = 0.5$	$2^{-2} = 0.25$	$2^{-3} = 0.125$	$2^{-4} = 0.0625$
b_7 = MSB	b_6	b_5	b_4	b_3	b_2	b_1	b_0 = LSB

(Imaginary) fixed point

Decimal ↔ binary conversion

Fractional part

- Convert separately from integral part

Simple algorithm

- Multiply fractional part by 2
- Value before decimal point provides next fraction bit, starting with MSB
- Strip of the fractional part and repeat until zero, or …
  - … the pattern starts repeating
  - … we have enough bits

$0.678_{10} = ???_2$

$0.678 \times 2 = 1.356$ $(1 = b_{-1})$
$0.356 \times 2 = 0.712$ $(0 = b_{-2})$
$0.712 \times 2 = 1.424$ $(1 = b_{-3})$
$0.424 \times 2 = 0.848$ $(0 = b_{-4})$
$0.848 \times 2 = 1.696$ $(1 = b_{-5})$
$0.696 \times 2 = 1.392$ $(1 = b_{-6})$
$0.392 \times 2 = 0.784$ $(0 = b_{-7})$
$0.784 \times 2 = 1.568$ $(1 = b_{-8})$
…
$0.678_{10} \cong 0.10101101_2$

Approximating real numbers

Floating point representation

- Decimal analogy: normalized scientific notation
  $D_0, D_1D_2 \ldots D_{P-1} \times 10^E$ (P valid digits, $1 \le D_0 \le 9$)
- Similarly in binary
  $B_0, B_1B_2 \ldots B_{P-1} \times 2^E$ (P valid bits, $1 \le B_0 \le 1$)

B_0 is always 1 in this form

In-memory representation

Sign	Exponent (with bias)	Significand (value of B_0 not stored $\Rightarrow$ hidden 1)

SP	DP
Bias = 127	Bias = 1023
P = 24	P = 53

- $(-1)^{Sign} \times \text{Significand} \times 2^{(\text{Exponent - Bias})}$
- Half (16-bit) / Single (32-bit) / Double (64-bit)

Real number → IEEE floating point

1. **Convert to binary fractional number**
   - Integral and fractional part, ignore sign
2. **Normalize the binary representation**
   - Move binary point to get 1.ssss $\times 2^{Exp}$
3. **Depending on the target FP representation**
   - Round significand to desired precision
   - Convert the exponent to biased representation
4. **Set the sign bit to reflect the sign**
5. **For in-memory representation**
   - Drop initial 1 from the significand (hidden 1)

Speaking of memory...

Programmer's perspective (logical view)

- 1-D array of N bytes numbered 0, … , N-1
- Individual bytes can be read or written to
- A particular byte is identified by its index

Index => Address

- Numbers occupy multiple bytes in memory
- Address of something = address of first byte

Memory is "visible" to CPU

- CPU sends address to memory controller, requesting bytes to be read or written
- Memory controller uses parts of the address to determine which part of memory to access

How to store multi-byte numbers?

Memory = array of bytes

- Chop up number into sequence of N bytes
  - B_{N-1} (Most Signif. Byte), …, B_1, B_0 (Least Signif. Byte)
- Store N bytes at consecutive addresses in memory
  - Addresses A, A+1, A+2, …, A+(N-1) for N-byte number
- CPU does this when storing/loading contents of its registers to/from memory at a given address

The order of bytes matters!

- Similar to sending bits over serial line.

Big Endian = MSB first

A	A+1	...	A+(N-1)
B_{N-1}	B_{N-2}	...	B_0

Little Endian = LSB first

A	A+1	...	A+(N-1)
B_0	B_1	...	B_{N-1}

Understanding a memory dump

Lists contents of memory

- Or any other address space, e.g., storage device (hard disk, solid state drive), or a file
- Contents of starting address and fixed number of consecutive addresses (usually all in hexadecimal)

We must know the interpretation!

- What is stored at `0x03194A7B`? Or at `0x03194A84`?

Address	Byte at (Address + 0), (Addres + 1), …, (Address + 7)							
...								
03194A78	AF	BC	39	F6	D0	24	91	34
03194A80	81	C9	A3	7C	00	80	B7	C2
03194A88	E2	6C	71	EA	59	FE	F5	49
...								