Digital Computers and Machine Representation of Data

K. Cooper[1]

[1]Department of Mathematics
Washington State University

2023

Bits and Bytes

Computers run on base 2

- 12v vs. 0v

Bits are organized into groups of eight: *bytes*

Bits and Bytes

Computers run on base 2

- 12v vs. 0v
- positive or negative magnetic charge

Bits are organized into groups of eight: *bytes*

Bits and Bytes

Computers run on base 2

- 12v vs. 0v
- positive or negative magnetic charge
- reflection or no reflection

Bits are organized into groups of eight: *bytes*

Bits and Bytes

Computers run on base 2

- 12v vs. 0v
- positive or negative magnetic charge
- reflection or no reflection
- lights on or off

Bits are organized into groups of eight: *bytes*

Words

Bytes are organized into groups that characterize the machine architecture: *words*

Words

Bytes are organized into groups that characterize the machine architecture: *words*

- One byte only enough to count numbers from 0 to 255

Words

Bytes are organized into groups that characterize the machine architecture: *words*

- One byte only enough to count numbers from 0 to 255
- Earliest microcomputers used eight-bit words, but computed with two bytes.

Words

Bytes are organized into groups that characterize the machine architecture: *words*

- One byte only enough to count numbers from 0 to 255
- Earliest microcomputers used eight-bit words, but computed with two bytes.
- 1980s – 16-bit words

Words

Bytes are organized into groups that characterize the machine architecture: *words*

- One byte only enough to count numbers from 0 to 255
- Earliest microcomputers used eight-bit words, but computed with two bytes.
- 1980s – 16-bit words
- 1990s – 32-bit words

Words

Bytes are organized into groups that characterize the machine architecture: *words*

- One byte only enough to count numbers from 0 to 255
- Earliest microcomputers used eight-bit words, but computed with two bytes.
- 1980s – 16-bit words
- 1990s – 32-bit words
- 2003→ 64-bit words.

Words

Bytes are organized into groups that characterize the machine architecture: *words*

- One byte only enough to count numbers from 0 to 255
- Earliest microcomputers used eight-bit words, but computed with two bytes.
- 1980s – 16-bit words
- 1990s – 32-bit words
- 2003→ 64-bit words.

Words

Bytes are organized into groups that characterize the machine architecture: *words*

- One byte only enough to count numbers from 0 to 255
- Earliest microcomputers used eight-bit words, but computed with two bytes.
- 1980s – 16-bit words
- 1990s – 32-bit words
- 2003→ 64-bit words. . . . big enough?

Data Types

Computers are required to support various types of data.

- Characters

Data Types

Computers are required to support various types of data.

- Characters
  - Letters,

Data Types

Computers are required to support various types of data.

- Characters
  - Letters,

Data Types

Computers are required to support various types of data.

- Characters
  - Letters, diacritical marks,

Data Types

Computers are required to support various types of data.

- Characters
  - Letters, diacritical marks, digits
  - Punctuation

Data Types

Computers are required to support various types of data.

- Characters
  - Letters, diacritical marks, digits
  - Punctuation
  - Symbols

Data Types

Computers are required to support various types of data.

- Characters
  - Letters, diacritical marks, digits
  - Punctuation
  - Symbols
- Integers

Data Types

Computers are required to support various types of data.

- Characters
  - Letters, diacritical marks, digits
  - Punctuation
  - Symbols
- Integers
  - Arithmetic

Data Types

Computers are required to support various types of data.

- Characters
  - Letters, diacritical marks, digits
  - Punctuation
  - Symbols
- Integers
  - Arithmetic
  - Addresses

Data Types

Computers are required to support various types of data.

- Characters
  - Letters, diacritical marks, digits
  - Punctuation
  - Symbols
- Integers
  - Arithmetic
  - Addresses
- Real and Complex Numbers

Data Types

Computers are required to support various types of data.

- Characters
  - Letters, diacritical marks, digits
  - Punctuation
  - Symbols
- Integers
  - Arithmetic
  - Addresses
- Real and Complex Numbers
  - Complex numbers are just ordered pairs of real numbers.

Characters

- ASCII - American Standard Code for Information Interchange

Characters

- ASCII - American Standard Code for Information Interchange
- Maps 7-bit integers to characters

Characters

- ASCII - American Standard Code for Information Interchange
- Maps 7-bit integers to characters

ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Unicode

- ASCII proved limited

Unicode

- ASCII proved limited
- Unicode now supports much larger character sets

Unicode

- ASCII proved limited
- Unicode now supports much larger character sets
- UTF-8, UTF-16, UTF-32

Unicode

- ASCII proved limited
- Unicode now supports much larger character sets
- UTF-8, UTF-16, UTF-32
- UTF-8 now dominates the Internet

Unicode

- ASCII proved limited
- Unicode now supports much larger character sets
- UTF-8, UTF-16, UTF-32
- UTF-8 now dominates the Internet
- UTF-8 is designed so that ASCII is subsumed.

Integers

- Typically one word

Integers

- Typically one word
- One bit for sign: 0 $\rightarrow$positive, 1 $\rightarrow$negative.

Integers

- Typically one word
- One bit for sign: 0 $\rightarrow$positive, 1 $\rightarrow$negative.
- In a 16-bit integer that leaves $2^{15} = 32768$ numbers.

Integers

- Typically one word
- One bit for sign: 0 $\rightarrow$positive, 1 $\rightarrow$negative.
- In a 16-bit integer that leaves $2^{15} = 32768$ numbers.
- The larger the number of bits to store the number, the more integers can be represented.

Integers

- Typically one word
- One bit for sign: 0 $\rightarrow$positive, 1 $\rightarrow$negative.
- In a 16-bit integer that leaves $2^{15} = 32768$ numbers.
- The larger the number of bits to store the number, the more integers can be represented.
- There is a largest computer integer that a computer can represent.

Integers

- Typically one word
- One bit for sign: 0 $\rightarrow$positive, 1 $\rightarrow$negative.
- In a 16-bit integer that leaves $2^{15} = 32768$ numbers.
- The larger the number of bits to store the number, the more integers can be represented.
- There is a largest computer integer that a computer can represent.
- Memory addresses are one-word integers

Integers

- Typically one word
- One bit for sign: 0 $\rightarrow$positive, 1 $\rightarrow$negative.
- In a 16-bit integer that leaves $2^{15} = 32768$ numbers.
- The larger the number of bits to store the number, the more integers can be represented.
- There is a largest computer integer that a computer can represent.
- Memory addresses are one-word integers
- A 32-bit machine can have at most 4GB of memory

N.B.

Some languages allow us to declare integers

N.B.

Some languages allow us to declare integers, such as C, C++, Python 2.x. . .

N.B.

Some languages allow us to declare integers, such as C, C++, Python 2.x. . .
In these languages, 1/2 = 0, but 1./2 = 0.5.

N.B.

Some languages allow us to declare integers, such as C, C++, Python 2.x. . .
In these languages, 1/2 = 0, but 1./2 = 0.5.

In Matlab and Python 3.x we do not have to worry about this.

Floating Point

What about non-integers - scientific computation?

- Use decimal notation

Floating Point

What about non-integers - scientific computation?

- Use decimal notation
- Only finitely many digits

Floating Point

What about non-integers - scientific computation?

- Use decimal notation
- Only finitely many digits
- Must represent very tiny numbers, and very large ones

Floating Point

What about non-integers - scientific computation?

- Use decimal notation
- Only finitely many digits
- Must represent very tiny numbers, and very large ones
- Use scientific notation $b \times 10^k$

Floating Point

What about non-integers - scientific computation?

- Use decimal notation
- Only finitely many digits
- Must represent very tiny numbers, and very large ones
- Use scientific notation $b \times 10^k$
- b and k are binary, and need a sign bit

Floating Point

What about non-integers - scientific computation?

- Use decimal notation
- Only finitely many digits
- Must represent very tiny numbers, and very large ones
- Use scientific notation $b \times 10^k$
- b and k are binary, and need a sign bit
- b is called the *mantissa*; k is the *exponent*.

Floating Point

What about non-integers - scientific computation?

- Use decimal notation
- Only finitely many digits
- Must represent very tiny numbers, and very large ones
- Use scientific notation $b \times 10^k$
- b and k are binary, and need a sign bit
- b is called the *mantissa*; k is the *exponent*.
- There are largest and smallest numbers that can be represented.

16-bit Floating Point

- 10-bit mantissa with 1 sign bit

16-bit Floating Point

- 10-bit mantissa with 1 sign bit
- Integers from -1024 to 1024, basically 3-digit mantissas

16-bit Floating Point

- 10-bit mantissa with 1 sign bit
- Integers from -1024 to 1024, basically 3-digit mantissas
- 5 bits for exponent: numbers from 0 to 31, shifted

16-bit Floating Point

- 10-bit mantissa with 1 sign bit
- Integers from -1024 to 1024, basically 3-digit mantissas
- 5 bits for exponent: numbers from 0 to 31, shifted

16-bit Floating Point

- 10-bit mantissa with 1 sign bit
- Integers from -1024 to 1024, basically 3-digit mantissas
- 5 bits for exponent: numbers from 0 to 31, shifted

Note that there is a sign bit for the mantissa, but the exponent is shifted.

16-bit Floating Point

- 360×10^0 is in this set.

16-bit Floating Point

- 360×10^0 is in this set.
- 36×10^1 would be the same number.

16-bit Floating Point

- 360×10^0 is in this set.
- 36×10^1 would be the same number.
- Mantissas are normalized for maximal significance

16-bit Floating Point

- 360×10^0 is in this set.
- 36×10^1 would be the same number.
- Mantissas are normalized for maximal significance
- Largest number: 1111111111×10^M, where M is max exponent

16-bit Floating Point

- 360×10^0 is in this set.
- 36×10^1 would be the same number.
- Mantissas are normalized for maximal significance
- Largest number: 1111111111×10^M, where M is max exponent
- Smallest number: 1000000000×10^m, m min exponent

16-bit Floating Point

- 360×10^0 is in this set.
- 36×10^1 would be the same number.
- Mantissas are normalized for maximal significance
- Largest number: 1111111111×10^M, where M is max exponent
- Smallest number: 1000000000×10^m, m min exponent
- If we used sign bit, then exponents would go from -16 to 16

16-bit Floating Point

- 360×10^0 is in this set.
- 36×10^1 would be the same number.
- Mantissas are normalized for maximal significance
- Largest number: 1111111111×10^M, where M is max exponent
- Smallest number: 1000000000×10^m, m min exponent
- If we used sign bit, then exponents would go from -16 to 16
- . . . numbers from 512×10^{-16} to 1023×10^{16}

16-bit Floating Point

- 360×10^0 is in this set.
- 36×10^1 would be the same number.
- Mantissas are normalized for maximal significance
- Largest number: 1111111111×10^M, where M is max exponent
- Smallest number: 1000000000×10^m, m min exponent
- If we used sign bit, then exponents would go from -16 to 16
- . . . numbers from 512×10^{-16} to 1023×10^{16}
- i.e. $\sim 5e-14$ to $\sim 1e19$

16-bit Floating Point

- 360×10^0 is in this set.
- 36×10^1 would be the same number.
- Mantissas are normalized for maximal significance
- Largest number: 1111111111×10^M, where M is max exponent
- Smallest number: 1000000000×10^m, m min exponent
- If we used sign bit, then exponents would go from -16 to 16
- ...numbers from 512×10^{-16} to 1023×10^{16}
- i.e. $\sim 5e-14$ to $\sim 1e19$
- For balance, we shift, instead of using sign bit.

16-bit Floating Point

- 360×10^0 is in this set.
- 36×10^1 would be the same number.
- Mantissas are normalized for maximal significance
- Largest number: 1111111111×10^M, where M is max exponent
- Smallest number: 1000000000×10^m, m min exponent
- If we used sign bit, then exponents would go from -16 to 16
- ...numbers from 512×10^{-16} to 1023×10^{16}
- i.e. $\sim 5e-14$ to $\sim 1e19$
- For balance, we shift, instead of using sign bit.
- Avogadro's number (6.022×10^{23}) cannot be represented in 16-bit floating point.

IEEE 754

- IEEE $\rightarrow$ Institute of Electrical and Electronics Engineers - standards organization

IEEE 754

- IEEE $\rightarrow$ Institute of Electrical and Electronics Engineers - standards organization
- 32-bit floating point

IEEE 754

- IEEE $\rightarrow$ Institute of Electrical and Electronics Engineers - standards organization
- 32-bit floating point
  - 23-bit mantissa, 1 sign bit

IEEE 754

- IEEE $\rightarrow$ Institute of Electrical and Electronics Engineers - standards organization
- 32-bit floating point
  - 23-bit mantissa, 1 sign bit
  - 8-bit exponent, shifted range from -126 to +127

IEEE 754

- IEEE $\rightarrow$ Institute of Electrical and Electronics Engineers - standards organization
- 32-bit floating point
  - 23-bit mantissa, 1 sign bit
  - 8-bit exponent, shifted range from -126 to +127
  - Effectively represents about 7 decimal digits

IEEE 754

- IEEE $\rightarrow$ Institute of Electrical and Electronics Engineers - standards organization
- 32-bit floating point
  - 23-bit mantissa, 1 sign bit
  - 8-bit exponent, shifted range from -126 to +127
  - Effectively represents about 7 decimal digits
- 64-bit floating point

IEEE 754

- IEEE $\rightarrow$ Institute of Electrical and Electronics Engineers - standards organization
- 32-bit floating point
  - 23-bit mantissa, 1 sign bit
  - 8-bit exponent, shifted range from -126 to +127
  - Effectively represents about 7 decimal digits
- 64-bit floating point
  - 52-bit mantissa, 1 sign bit

IEEE 754

- IEEE $\rightarrow$ Institute of Electrical and Electronics Engineers - standards organization
- 32-bit floating point
  - 23-bit mantissa, 1 sign bit
  - 8-bit exponent, shifted range from -126 to +127
  - Effectively represents about 7 decimal digits
- 64-bit floating point
  - 52-bit mantissa, 1 sign bit
  - 11-bit exponent, shifted range from -1022 to +1023

IEEE 754

- IEEE $\rightarrow$ Institute of Electrical and Electronics Engineers - standards organization
- 32-bit floating point
  - 23-bit mantissa, 1 sign bit
  - 8-bit exponent, shifted range from -126 to +127
  - Effectively represents about 7 decimal digits
- 64-bit floating point
  - 52-bit mantissa, 1 sign bit
  - 11-bit exponent, shifted range from -1022 to +1023
  - Effectively represents almost 16 decimal digits