Computer Science 210

Computer Systems 1

Lecture Notes

Lecture 5,

Representation of Fractions & Floating Point Numbers

Credits: Adapted from slides prepared by Gregory T. Byrd, North Carolina State University

Text: ASCII Characters

- ASCII: Maps 128 characters to 7-bit code.
  - both printable and non-printable (ESC, DEL, …) characters
  - "ASCIIbetical" order

00	nul	10	dle	20	sp	30	0	40	@	50	P	60	`	70	p
01	soh	11	dc1	21	!	31	1	41	A	51	Q	61	a	71	q
02	stx	12	dc2	22	"	32	2	42	B	52	R	62	b	72	r
03	etx	13	dc3	23	#	33	3	43	C	53	S	63	c	73	s
04	eot	14	dc4	24	$	34	4	44	D	54	T	64	d	74	t
05	enq	15	nak	25	%	35	5	45	E	55	U	65	e	75	u
06	ack	16	syn	26	&	36	6	46	F	56	V	66	f	76	v
07	bel	17	etb	27	'	37	7	47	G	57	W	67	g	77	w
08	bs	18	can	28	(	38	8	48	H	58	X	68	h	78	x
09	ht	19	em	29	)	39	9	49	I	59	Y	69	i	79	y
0a	nl	1a	sub	2a	*	3a	:	4a	J	5a	Z	6a	j	7a	z
0b	vt	1b	esc	2b	+	3b	;	4b	K	5b	[	6b	k	7b	{
0c	np	1c	fs	2c	,	3c	<	4c	L	5c	\	6c	l	7c	|
0d	cr	1d	gs	2d	-	3d	=	4d	M	5d	]	6d	m	7d	}
0e	so	1e	rs	2e	.	3e	>	4e	N	5e	^	6e	n	7e	~
0f	si	1f	us	2f	/	3f	?	4f	O	5f	_	6f	o	7f	del

Interesting Properties of ASCII Code

- What is relationship between a decimal digit (‘0’, ‘1’, …) and its ASCII code?
- What is the difference between an upper-case letter (‘A’, ‘B’, …) and its lower-case equivalent (‘a’, ‘b’, …)?
- Given two ASCII characters, how do we tell which comes first in alphabetical order?
- Is 128 characters enough?
  (http://www.unicode.org/)

No new operations -- integer arithmetic and logic.

Representing More Characters

- Unicode – International standard
  - More than 110,000 characters defined
  - 8-, 16-, and 32-bit codes (UTF-8, UTF-16 & UTF-32)
  - “ASCII” is a subset of Unicode

Today: Representation of non-Integers

- Text, strings
- Fractions
- Scientific notation/Floating point representation

Other Data Types

- Text strings
  - sequence of characters, terminated with NULL (0)
  - typically, no hardware support
- Image
  - array of pixels
    - monochrome: one bit (1/0 = black/white)
    - color: red, green, blue (RGB) components (e.g., 8 bits each)
    - other properties: transparency
  - hardware support:
    - typically none, in general-purpose processors
    - MMX -- multiple 8-bit operations on 32- or 64-bit word
- Sound
  - sequence of fixed-point numbers

LC-3 Data Types

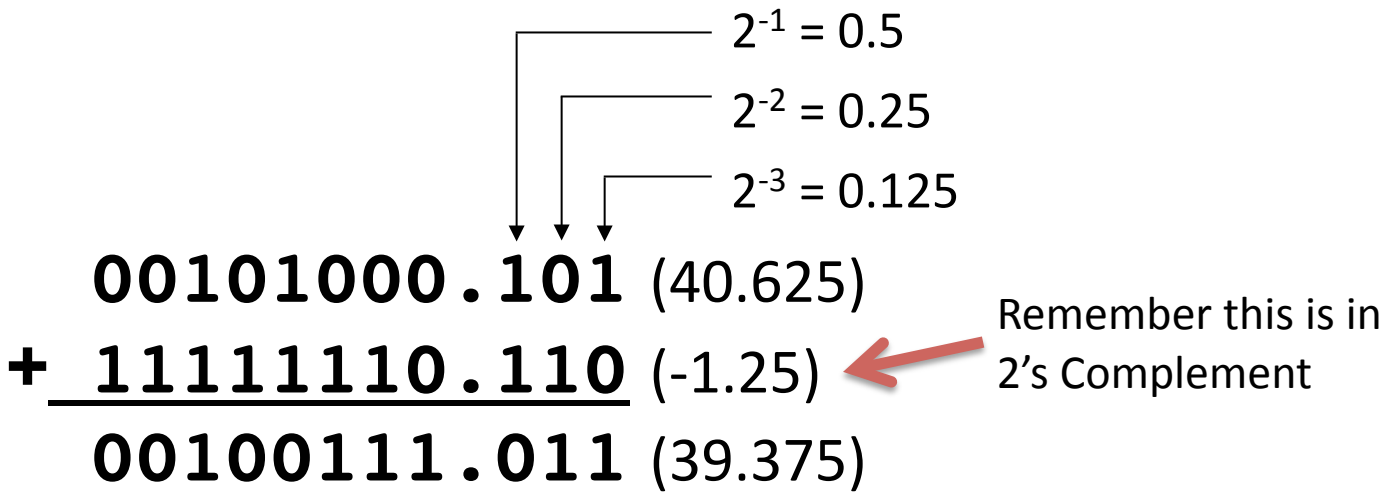
- Some data types are supported directly by the instruction set architecture.

- For LC-3, there is only one hardware-supported data type:
  - 16-bit 2's complement signed integer
  - Operations: ADD, AND, NOT

- Other data types are supported by *interpreting* 16-bit values as logical, text, fixed-point, etc., in the software that we write.

Fractions: Fixed-Point

- How can we represent fractions?
  - Use a "binary point" to separate positive from negative powers of two -- just like "decimal point."
  - 2's comp addition and subtraction still work
    - only if binary points are aligned



No new operations -- same as integer arithmetic

Video: how to convert decimal fractions to binary http://youtu.be/Y4Q9PnjKhac

Scientific Notation

- We can only represent 2^{32} (~ 4 billion) or maybe 2^{64} (~ 18 quintillion) or even 2^{128} (~ 3.4×10^{38} or 340 decillion) unique values, but there are
  - Infinitely many numbers between any two integers
  - Infinitely many numbers between any two real numbers!
- We can only represent a (small) finite number of values.
  - These values are not spread uniformly along number line
  - Infinity of numbers between zero and one

Video on Scientific Notation http://youtu.be/QtN7l3nlkhs

Scientific Notation

$975.25 \rightarrow 9.7525 \times 10^2$

- Conventional (decimal) notation:
  - $\pm$ mantissa x 10^{exponent}
    - $1 \leq \text{mantissa} < 10$
  - exponent is signed integer

- Binary notation:
  - $\pm$ mantissa x 2^{exponent}
    - $1 \leq \text{mantissa} < 2$
  - exponent is signed integer

Significant Digits

- Accuracy of measurement leads to notion of *Significant Digits*
  - For most purposes, we don't need high precision
  - Accuracy of calculations is generally limited by least precise numbers
  - Can represent numbers with a few significant digits
    - $6.0221415 * 10^{23}$ Avogadro's Number (approximately)
    - 299,792,458 meters/sec -- Speed of Light (exactly!)
      - By definition, a meter is the distance light travels through a vacuum in exactly 1/299792458 seconds
    - 3.141592…
      - Computable to arbitrary accuracy, but
      - More digits probably won't improve result

Representation of Floating Point Numbers (Reals)

- As with integers and chars, we ask
- Which reals? There is an infinite number between two adjacent integers.

  In fact, there are an infinite number between any two reals!!!!!!!
- Which bit patterns for selected reals?
- Answer for both strongly related to scientific notation.

Very Large and Very Small: Floating-Point

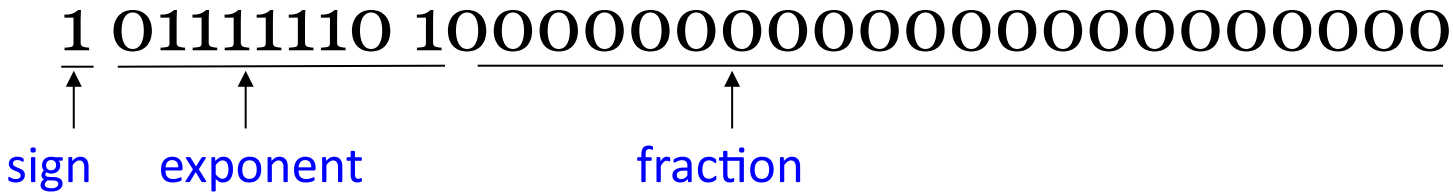
- Large values: 6.023×10^{23} -- requires 79 bits
- Small values: 6.626×10^{-34} -- requires >110 bits
- Use equivalent of "scientific notation": $F \times 2^{E}$
- Need to represent F (*fraction*), E (*exponent*), and sign.
- IEEE 754 Floating-Point Standard (32-bits):

- Exponent uses "biased" representation (no sign bit)
- Fraction has implicit 1

Video converting decimal to floating-point binary representation
http://youtu.be/iQFG7sAa7i4

Floating Point Example

- Single-precision IEEE floating point number:



  - Sign is 1 – number is negative.
  - Exponent field is 01111110 = 126 (decimal).
  - Fraction is 0.10000000000… = 0.5 (decimal).

- Value = $-1.5 \times 2^{(126-127)} = -1.5 \times 2^{-1} =$ -0.75

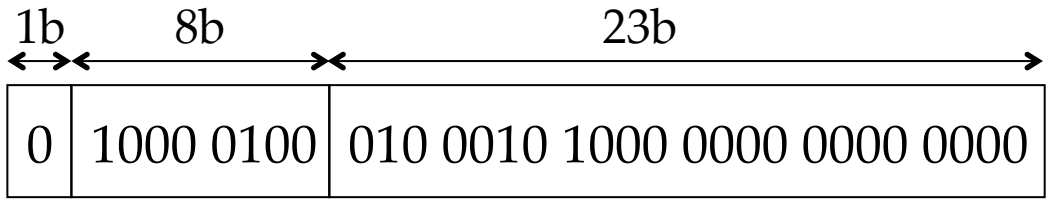
IEEE Floating-Point Standard (32-bit)

$$N = (-1)^S \times 1.\text{fraction} \times 2^{\text{exponent}-127},\ 1 \le \text{exponent} \le 254$$

$$N = (-1)^S \times 0.\text{fraction} \times 2^{-126},\ \text{exponent} = 0$$

Example: FP Representation for 40.625_{ten}

1. From earlier slide $40.625_{ten} = 00101000.101_{two}$
2. Put the binary rep. into normal form (make it look like scientific notation): $+101000.101 \times 2^0 = +1.01000101 \times 2^5$
5. 5 is the true exponent; with bias: $5 + 127 = 132_{ten}$
   $= 10000100_{two}$
6. Mantissa/Fraction occupies 23 bits:
   (1.) 010 0010 1000 0000 0000 0000



Floating-Point Operations

- Will regular 2's complement arithmetic work for
  Floating Point numbers?
- (*Hint*: In decimal, how do we compute $3.07 \times 10^{12} + 9.11 \times 10^{8}$?)

Floating-Point Arithmetic

- Floating point operations may overflow but, more importantly, floating point operations are inherently *inexact*
- ➢ Some numbers (*e.g.* "repeating decimal") cannot be represented exactly.
- ➢ Introduces the "Rounding" problem
  - Every inexact result creates a difference between the mathematical value and the computed value.
  - Errors accumulate, often benignly by cancelling out.
  - Worst-case accumulation of error can be enormous.