REPRESENTATION OF NUMBERS

CMPT 110

TENTATIVE SYLLABUS

Week	Topic
1 (Sept. 4th)	Introduction to Course
2 (11th)	Introduction to Programming
3 (18th)	Programming in VB
4 (29th)	Events
5 (Oct. 2nd)	Data Types and Operators Review for Midterm
6 (9th)	Representing and Storing Values **MIDTERM – Thursday, October 11th**
7 (16th)	Decisions
8 (23rd)	Iterations
9 (30th)	Arrays
10 (Nov. 6th)	I/O
11 (13th)	Graphics
12 (21st)	Advanced Topics
13 (27th)	Review

Study Guide

- Unit 1: What is programming about?
- Unit 2: Programming in Visual Basic
- Unit 3: Events
- Unit 4: Representing and Storing Values
- Unit 5: Subprograms
- Unit 6: More About Subprograms
- Unit 7: Decisions, Decisions
- Unit 8: Please Repeat That
- Unit 9: Representing Lists and Tables with Arrays
- Unit 10: File Input and Output
- Unit 11: Graphs and Simulation

CMPT 110 Study Guide

Unit 1: What is programming about?

Unit 2: Programming in Visual Basic

Unit 3: Events

Unit 4: Representing and Storing Values

PRELIMINARY QUESTIONS FROM STUDY GUIDE 4

- What does the range (–2,147,483,648 to +2,147,483,647) represent in VB?
- What is does the range (-1.79769313486231570E+308 through -4.94065645841246544E-324) for negative values and from (4.94065645841246544E-324 through 1.79769313486231570E+308) for positive values represent?

PRELIMINARY QUESTIONS FROM STUDY GUIDE 4

- What does the range (–2,147,483,648 to +2,147,483,647) represent in VB?
  - **The Integer data type.**
- What is does the range (-1.79769313486231570E+308 through -4.94065645841246544E-324) for negative values and from (4.94065645841246544E-324 through 1.79769313486231570E+308) for positive values represent?
  - **The range of signed IEEE 64-bit (8-byte) double-precision floating-point numbers.**

PRELIMINARY QUESTIONS FROM STUDY GUIDE 4

- What are possible variable names from the phrase "seasonal adjusted amount":

PRELIMINARY QUESTIONS FROM STUDY GUIDE 4

- What are possible variable names from the phrase "seasonal adjusted amount":

```
seasonal_adjusted_amount

seasonalAdjustedAmount
```

As both examples show, it is also common practice not to capitalize the first letter of a variable name. The reason for not doing so is that another role for symbolic names is to label methods. You have already seen instances of such names in the previous units. By always capitalizing method names and never capitalizing variable names, it is easy to recognize at a glance whether an identifier in a program is the name of a method or a variable.

OBJECTIVES

- The binary, hexadecimal, and octal number systems
- Finite representation of unsigned integers
- Finite representation of signed integers
- Finite representation of real numbers

BITS, BYTES, AND WORDS

BIT: A Binary digit (either 0 or 1) – the minimum unit of communication.

BYTE: 8 bits – a common unit of computer memory based on ASCII.

WORD: A computer word is a group of bits which are passed around together during computation. The word length of the computer's processor is how many bits are grouped together (registers):

- 8-bit machine (e.g. Nintendo Gameboy, 1989)
- 16-bit machine (e.g. Sega Genesis, 1989)
- 32-bit machines (e.g. Sony PlayStation, 1994)
- 64-bit machines (e.g. Nintendo 64, 1996)
    - *Do you remember the quiz question on 64-bit representation of a number?*

ASCII

Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	[NULL]	48	30	110000	60	0	96	60	1100000	140	`
1	1	1	1	[START OF HEADING]	49	31	110001	61	1	97	61	1100001	141	a
2	2	10	2	[START OF TEXT]	50	32	110010	62	2	98	62	1100010	142	b
3	3	11	3	[END OF TEXT]	51	33	110011	63	3	99	63	1100011	143	c
4	4	100	4	[END OF TRANSMISSION]	52	34	110100	64	4	100	64	1100100	144	d
5	5	101	5	[ENQUIRY]	53	35	110101	65	5	101	65	1100101	145	e
6	6	110	6	[ACKNOWLEDGE]	54	36	110110	66	6	102	66	1100110	146	f
7	7	111	7	[BELL]	55	37	110111	67	7	103	67	1100111	147	g
8	8	1000	10	[BACKSPACE]	56	38	111000	70	8	104	68	1101000	150	h
9	9	1001	11	[HORIZONTAL TAB]	57	39	111001	71	9	105	69	1101001	151	i
10	A	1010	12	[LINE FEED]	58	3A	111010	72	:	106	6A	1101010	152	j
11	B	1011	13	[VERTICAL TAB]	59	3B	111011	73	;	107	6B	1101011	153	k
12	C	1100	14	[FORM FEED]	60	3C	111100	74	<	108	6C	1101100	154	l
13	D	1101	15	[CARRIAGE RETURN]	61	3D	111101	75	=	109	6D	1101101	155	m
14	E	1110	16	[SHIFT OUT]	62	3E	111110	76	>	110	6E	1101110	156	n
15	F	1111	17	[SHIFT IN]	63	3F	111111	77	?	111	6F	1101111	157	o
16	10	10000	20	[DATA LINK ESCAPE]	64	40	1000000	100	@	112	70	1110000	160	p
17	11	10001	21	[DEVICE CONTROL 1]	65	41	1000001	101	A	113	71	1110001	161	q
18	12	10010	22	[DEVICE CONTROL 2]	66	42	1000010	102	B	114	72	1110010	162	r
19	13	10011	23	[DEVICE CONTROL 3]	67	43	1000011	103	C	115	73	1110011	163	s
20	14	10100	24	[DEVICE CONTROL 4]	68	44	1000100	104	D	116	74	1110100	164	t
21	15	10101	25	[NEGATIVE ACKNOWLEDGE]	69	45	1000101	105	E	117	75	1110101	165	u
22	16	10110	26	[SYNCHRONOUS IDLE]	70	46	1000110	106	F	118	76	1110110	166	v
23	17	10111	27	[ENG OF TRANS. BLOCK]	71	47	1000111	107	G	119	77	1110111	167	w
24	18	11000	30	[CANCEL]	72	48	1001000	110	H	120	78	1111000	170	x
25	19	11001	31	[END OF MEDIUM]	73	49	1001001	111	I	121	79	1111001	171	y
26	1A	11010	32	[SUBSTITUTE]	74	4A	1001010	112	J	122	7A	1111010	172	z
27	1B	11011	33	[ESCAPE]	75	4B	1001011	113	K	123	7B	1111011	173	{
28	1C	11100	34	[FILE SEPARATOR]	76	4C	1001100	114	L	124	7C	1111100	174	|
29	1D	11101	35	[GROUP SEPARATOR]	77	4D	1001101	115	M	125	7D	1111101	175	}
30	1E	11110	36	[RECORD SEPARATOR]	78	4E	1001110	116	N	126	7E	1111110	176	~
31	1F	11111	37	[UNIT SEPARATOR]	79	4F	1001111	117	O	127	7F	1111111	177	[DEL]
32	20	100000	40	[SPACE]	80	50	1010000	120	P					
33	21	100001	41	!	81	51	1010001	121	Q					
34	22	100010	42	"	82	52	1010010	122	R					
35	23	100011	43	#	83	53	1010011	123	S					
36	24	100100	44	$	84	54	1010100	124	T					
37	25	100101	45	%	85	55	1010101	125	U					
38	26	100110	46	&	86	56	1010110	126	V					
39	27	100111	47	'	87	57	1010111	127	W					
40	28	101000	50	(	88	58	1011000	130	X					
41	29	101001	51	)	89	59	1011001	131	Y					
42	2A	101010	52	*	90	5A	1011010	132	Z					
43	2B	101011	53	+	91	5B	1011011	133	[					
44	2C	101100	54	,	92	5C	1011100	134	\					
45	2D	101101	55	-	93	5D	1011101	135	]					
46	2E	101110	56	.	94	5E	1011110	136	^					
47	2F	101111	57	/	95	5F	1011111	137	_					

ASCII

Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	[NULL]
1	1	1	1	[START OF HEADING]
2	2	10	2	[START OF TEXT]
3	3	11	3	[END OF TEXT]
4	4	100	4	[END OF TRANSMISSION]
5	5	101	5	[ENQUIRY]
6	6	110	6	[ACKNOWLEDGE]
7	7	111	7	[BELL]
8	8	1000	10	[BACKSPACE]
9	9	1001	11	[HORIZONTAL TAB]
10	A	1010	12	[LINE FEED]
11	B	1011	13	[VERTICAL TAB]
12	C	1100	14	[FORM FEED]
13	D	1101	15	[CARRIAGE RETURN]
14	E	1110	16	[SHIFT OUT]
15	F	1111	17	[SHIFT IN]
16	10	10000	20	[DATA LINK ESCAPE]

SIGNED AND UNSIGNED INTEGERS

- The term "unsigned" in computer programming indicates a variable that can hold only positive numbers.
- The term "signed" in computer code indicates that a variable can hold negative and positive values.
- In 32-bit integers, an unsigned integer has a range of 0 to 2^{32}-1 = 0 to 4,294,967,295 or about 4 billion.
- The signed version goes from -2^{31}-1 to 2^{31}, which is –2,147,483,648 to 2,147,483,647 or about -2 billion to +2 billion. The range is the same, but it is shifted on the number line.

https://www.thoughtco.com/definition-of-unsigned-958174

REPRESENTATION OF INTEGERS

REPRESENTATION OF INTEGERS

Let b be an integer greater than 1. Then if n is a positive integer, it can be expressed uniquely in the form

$$n = a_k b^k + a_{k-1} b^{k-1} + \cdots + a_1 b + a_0,$$

where k is a nonnegative integer, $a_0, a_1, \ldots, a_k$ are nonnegative integers less than b, and $a_k \neq 0$.

Rosen

REPRESENTATION OF INTEGERS

- With d decimal digits, we can represent 10^d different values, usually the numbers 0 to (10^d -1) inclusive
- In binary with n bits this becomes 2^n values, usually the range 0 to (2^n -1)
- Computers usually assign a set number of bits (physical switches) to an instance of a type:
  - An integer is often 32 bits, so we can represent positive integers from 0 to 4,294,967,295 inclusive.
  - Or a range of negative and positive integers.

REPRESENTATION OF INTEGERS

- Higher bases make for shorter numbers that are easier for humans to manipulate.

  e.g. $6654733_d = 11001011000101100001101_b$

- We traditionally choose powers-of-2 bases because this corresponds to whole chunks of binary.
  - **Octal** is base-8 ($8=2^3$ digits, which means 3 bits per digit)
  - $6654733_d = 011\text{-}001\text{-}011\text{-}000\text{-}101\text{-}100\text{-}001\text{-}101_b = 31305415_o$
  - **Hexadecimal** is base-16 ($16=2^4$ digits so 4 bits per digit)
    - Our ten decimal digits aren't enough, so we add 6 new ones: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
    - $6654733_d = 0110\text{-}0101\text{-}1000\text{-}1011\text{-}0000\text{-}1101_b = 658B0D_h$
- Because we constantly slip between binary and hex, we have a special marker for it:
  - Prefix with '0x' (zero-x). So 0x658B0D=6654733_d, 0x123=291_d

REPRESENTATION OF INTEGERS

- A de-facto standard of 8 bits has now emerged
  - 256 values
  - 0 to 255 inclusive.
  - It takes two hexadecimal digits to describe this
  - 0x00=0, 0xFF=255
- Check: What does 0xBD represent?

REPRESENTATION OF INTEGERS

- A de-facto standard of 8 bits has now emerged
  - 256 values
  - 0 to 255 inclusive.
  - It takes two hexadecimal digits to describe this
  - 0x00=0, 0xFF=255
- Check: What does 0xBD represent?
  - B → 11 or 1011
  - D → 13 or 1101
  - Result is $11 \times 16^1 + 13 \times 16^0$ = **189** or 10111101

THE BINARY NUMBER SYSTEM

Binary to decimal: expand using positional notation

$$
\begin{aligned}
100101_B &= (1*2^5)+(0*2^4)+(0*2^3)+(1*2^2)+(0*2^1)+(1*2^0) \\
&= 32 + 0 + 0 + 4 + 0 + 1 \\
&= 37
\end{aligned}
$$

THE BINARY NUMBER SYSTEM

Decimal to binary: do the reverse

- Determine largest power of 2 ≤ number; write template

$$37 = (?*2^5)+(?*2^4)+(?*2^3)+(?*2^2)+(?*2^1)+(?*2^0)$$

- Fill in template

$$37 = (1*2^5)+(0*2^4)+(0*2^3)+(1*2^2)+(0*2^1)+(1*2^0)$$

```
 37
-32
  5
 -4
  1
 -1
  0
```

100101_B

THE BINARY NUMBER SYSTEM

Decimal to binary: do the reverse

- Determine largest power of 2 ≤ number; write template

$$37 = (?*2^5)+(?*2^4)+(?*2^3)+(?*2^2)+(?*2^1)+(?*2^0)$$

- Fill in template

$$37 = (1*2^5)+(0*2^4)+(0*2^3)+(1*2^2)+(0*2^1)+(1*2^0)$$

5
-4
1
-1
0

Not Responsible for the inverse calculation

THE BINARY NUMBER SYSTEM

EXAMPLE 1 What is the decimal expansion of the integer that has $(1\ 0101\ 1111)_2$ as its binary expansion?

THE BINARY NUMBER SYSTEM

EXAMPLE 1 What is the decimal expansion of the integer that has $(1\ 0101\ 1111)_2$ as its binary expansion?

Solution: We have

$$\begin{aligned}(1\ 0101\ 1111)_2 &= 1 \cdot 2^8 + 0 \cdot 2^7 + 1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 \\ &\quad + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 351.\end{aligned}$$

◀

THE OCTAL NUMBER SYSTEM

Name

- “octo” (Latin) => eight

Characteristics

- Eight symbols
  - 0 1 2 3 4 5 6 7

Computer programmers often use the octal number system

Why?

THE OCTAL NUMBER SYSTEM

From Wikipedia, the free encyclopedia

The **octal** numeral system, or **oct** for short, is the base-8 number system, and uses the digits 0 to 7. Octal numerals can be made from binary numerals by grouping consecutive binary digits into groups of three (starting from the right). For example, the binary representation for decimal 74 is 1001010. Two zeroes can be added at the left: (00)1 001 010, corresponding the octal digits 1 1 2, yielding the octal representation 112.

In the decimal system each decimal place is a power of ten. For example:

$$\mathbf{74}_{10} = \mathbf{7} \times 10^1 + \mathbf{4} \times 10^0$$

In the octal system each place is a power of eight. For example:

$$\mathbf{112}_8 = \mathbf{1} \times 8^2 + \mathbf{1} \times 8^1 + \mathbf{2} \times 8^0$$

By performing the calculation above in the familiar decimal system we see why 112 in octal is equal to 64+8+2 = 74 in decimal.

THE OCTAL NUMBER SYSTEM

Decimal	Octal
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	10
9	11
10	12
11	13
12	14
13	15
14	16
15	17

Decimal	Octal
16	20
17	21
18	22
19	23
20	24
21	25
22	26
23	27
24	30
25	31
26	32
27	33
28	34
29	35
30	36
31	37

Decimal	Octal
32	40
33	41
34	42
35	43
36	44
37	45
38	46
39	47
40	50
41	51
42	52
43	53
44	54
45	55
46	56
47	57
...	...

THE OCTAL NUMBER SYSTEM

Octal to decimal: expand using positional notation

$$
\begin{aligned}
37_{O} &= (3*8^{1}) + (7*8^{0}) \\
&= 24 + 7 \\
&= 31
\end{aligned}
$$

THE OCTAL NUMBER SYSTEM

Observation: $8^1 = 2^3$

- Every 1 octal digit corresponds to 3 binary digits

Binary to octal

```
001010000100111101B
 1  2  0  4  7  5O
```

Digit count in binary number not a multiple of 3 => pad with zeros on left

Octal to binary

```
 1  2  0  4  7  5O
001010000100111101B
```

Discard leading zeros from binary number if appropriate

THE OCTAL NUMBER SYSTEM

EXAMPLE 2 What is the decimal expansion of the number with octal expansion $(7016)_8$?

THE OCTAL NUMBER SYSTEM

EXAMPLE 2 What is the decimal expansion of the number with octal expansion $(7016)_8$?

Solution: Using the definition of a base b expansion with $b = 8$ tells us that

$$(7016)_8 = 7 \cdot 8^3 + 0 \cdot 8^2 + 1 \cdot 8 + 6 = 3598.$$

Rosen

THE HEXADECIMAL NUMBER SYSTEM

Name

- "hexa" (Greek) => six
- "decem" (Latin) => ten

Characteristics

- Sixteen symbols
  - **0 1 2 3 4 5 6 7 8 9 A B C D E F**
- Positional
  - $\mathbf{A13D_H \neq 3DA1_H}$

Computer programmers often use the hexadecimal number system

Why?

THE HEXADECIMAL NUMBER SYSTEM

Decimal	Hex
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F

Decimal	Hex
16	10
17	11
18	12
19	13
20	14
21	15
22	16
23	17
24	18
25	19
26	1A
27	1B
28	1C
29	1D
30	1E
31	1F

Decimal	Hex
32	20
33	21
34	22
35	23
36	24
37	25
38	26
39	27
40	28
41	29
42	2A
43	2B
44	2C
45	2D
46	2E
47	2F
...	...

THE HEXADECIMAL NUMBER SYSTEM

Hexadecimal to decimal: expand using positional notation

$$\begin{aligned} 25_H &= (2*16^1) + (5*16^0) \\ &= 32 + 5 \\ &= 37 \end{aligned}$$

THE HEXADECIMAL NUMBER SYSTEM

Observation: $16^1 = 2^4$

- Every 1 hexadecimal digit corresponds to 4 binary digits

Binary to hexadecimal

1010000100111101_B
A 1 3 D_H

Digit count in binary number not a multiple of 4 => pad with zeros on left

Hexadecimal to binary

A 1 3 D_H
1010000100111101_B

Discard leading zeros from binary number if appropriate

THE HEXADECIMAL NUMBER SYSTEM

Observation: $16^1 = 2^4$

The **hexadecimal** system is commonly **used** by programmers to describe locations in memory because it can **represent** every byte (i.e., eight bits) as two consecutive **hexadecimal** digits instead of the eight digits that would be required by **binary** (i.e., base 2) **numbers** and the three digits that would be required with decimal ... Sep 14, 2005

Hexadecimal system: describes locations in memory, colors

www.linfo.org/hexadecimal.html

A 1 3 D_H

1010000100111101_B

from binary number if appropriate

THE HEXADECIMAL NUMBER SYSTEM

EXAMPLE 3 What is the decimal expansion of the number with hexadecimal expansion $(2AE0B)_{16}$?

Rosen

THE HEXADECIMAL NUMBER SYSTEM

EXAMPLE 3 What is the decimal expansion of the number with hexadecimal expansion $(2AE0B)_{16}$?

Solution: Using the definition of a base b expansion with $b = 16$ tells us that

$$(2AE0B)_{16} = 2 \cdot 16^4 + 10 \cdot 16^3 + 14 \cdot 16^2 + 0 \cdot 16 + 11 = 175627.$$

Rosen

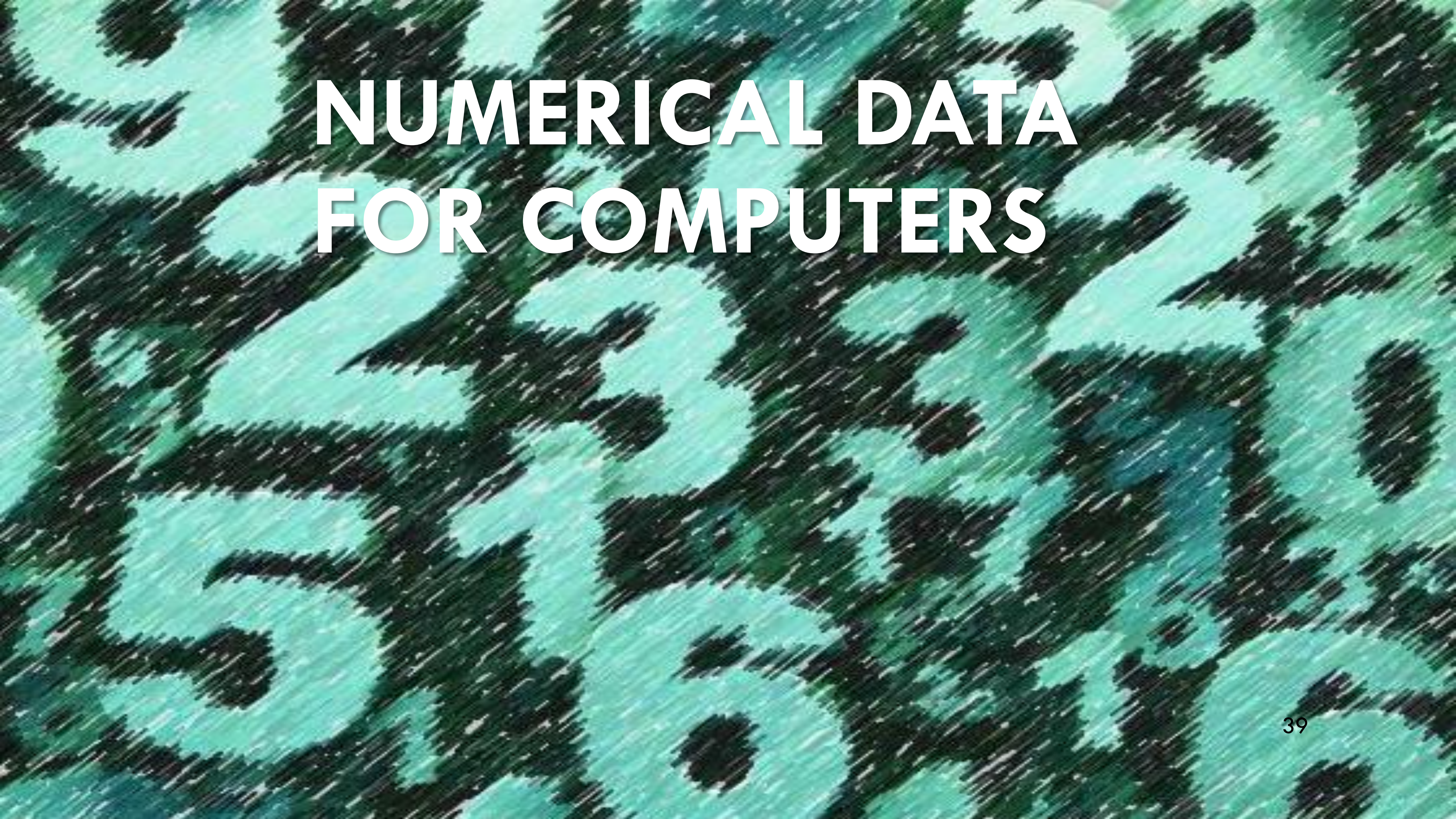
HEXADECIMAL, OCTAL, AND BINARY REPRESENTATION OF THE INTEGERS

TABLE 1 Hexadecimal, Octal, and Binary Representation of the Integers 0 through 15.

Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Hexadecimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Octal	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17
Binary	0	1	10	11	100	101	110	111	1000	1001	1010	1011	1100	1101	1110	1111

Rosen

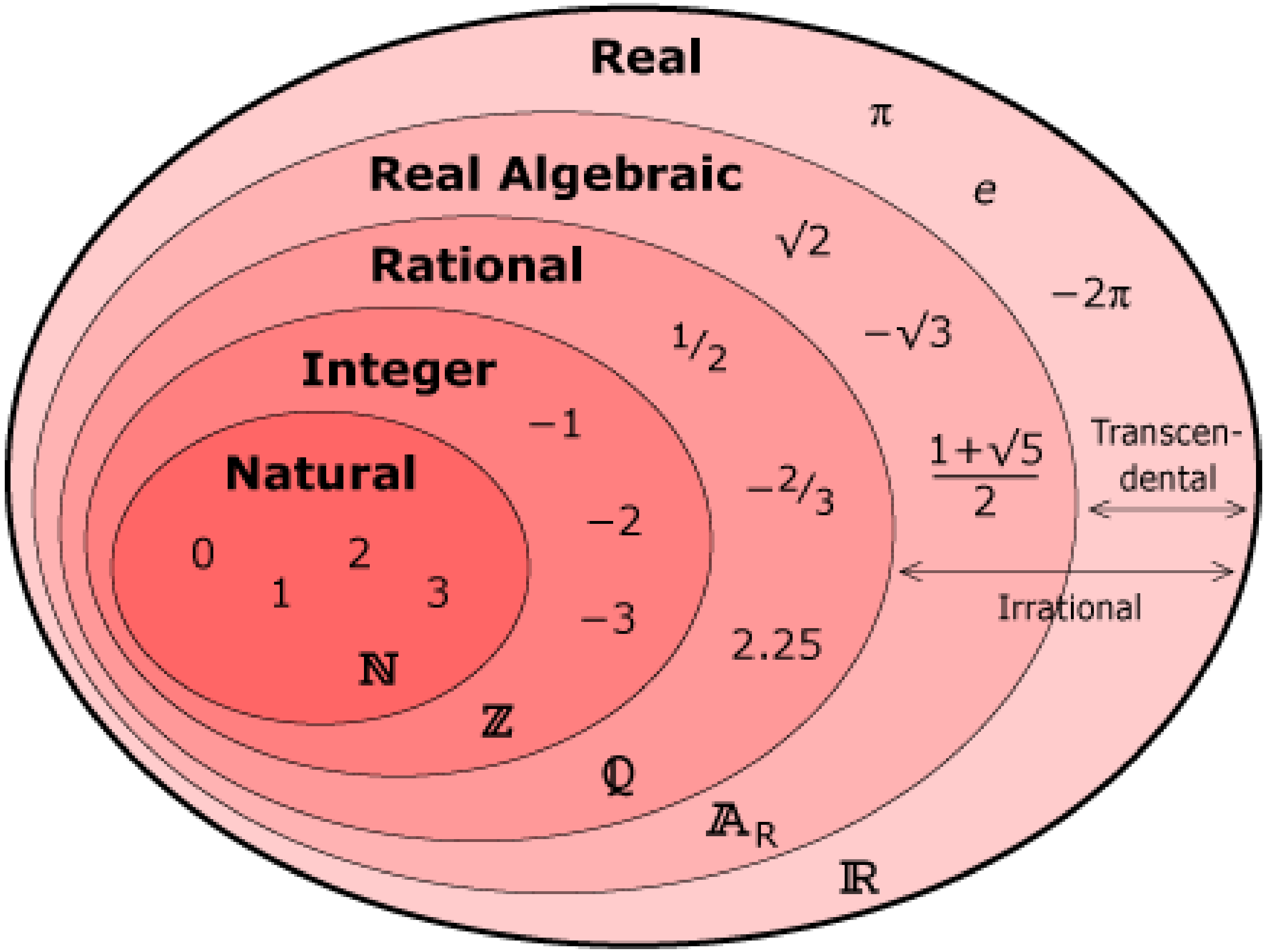
NUMERICAL DATA FOR COMPUTERS



NUMERICAL DATA FOR COMPUTERS

- Recall that the Central Processing Unit (CPU) contains the Arithmetic/Logic Unit (ALU – the circuit that performs the basic arithmetic and logic operations) and the CU (traffic cop).
- A computer program mostly processes numerical and logical data into useful information of one sort or another, in part, through these circuits.
- Von Neumann computers are binary and, therefore, *integer in nature*.
- Data comes in many forms such as **integers**, **real numbers** (floating-point), **Booleans**, **characters**, and **alphanumeric strings**.

RECALL SET THEORY



http://thinkzone.wlonk.com/Numbers/NumberSets.htm

REPRESENTATION OF INTEGERS

Integers are *whole numbers* or *fixed-point numbers* with the radix point *fixed* after the least-significant bit. They are contrast to *real numbers* or *floating-point numbers*, where the position of the radix point varies. It is important to take note that integers and floating-point numbers are treated differently in computers. They have different representation and are processed differently (e.g., floating-point numbers are processed in a so-called floating-point processor). Floating-point numbers will be discussed later.

Computers use *a fixed number of bits* to represent an integer. The commonly-used bit-lengths for integers are 8-bit, 16-bit, 32-bit or 64-bit. Besides bit-lengths, there are two representation schemes for integers:

1. *Unsigned Integers*: can represent zero and positive integers.
2. *Signed Integers*: can represent zero, positive and negative integers. Three representation schemes had been proposed for signed integers:
    a. Sign-Magnitude representation
    b. 1's Complement representation
    c. 2's Complement representation

You, as the programmer, need to decide on the bit-length and representation scheme for your integers, depending on your application's requirements. Suppose that you need a counter for counting a small quantity from 0 up to 200, you might choose the 8-bit unsigned integer scheme as there is no negative numbers involved.

http://www.ntu.edu.sg/home/ehchua/programming/java/datarepresentation.html

REPRESENTATION OF INTEGERS - UNSIGNED

Unsigned integers can represent zero and positive integers, but not negative integers. The value of an unsigned integer is interpreted as "*the magnitude of its underlying binary pattern*".

Example 1: Suppose that *n*=8 and the binary pattern is `0100 0001B`, the value of this unsigned integer is `1×2^0 + 1×2^6 = 65D`.

Example 2: Suppose that *n*=16 and the binary pattern is `0001 0000 0000 1000B`, the value of this unsigned integer is `1×2^3 + 1×2^12 = 4104D`.

Example 3: Suppose that *n*=16 and the binary pattern is `0000 0000 0000 0000B`, the value of this unsigned integer is `0`.

An *n*-bit pattern can represent `2^n` distinct integers. An *n*-bit unsigned integer can represent integers from `0` to `(2^n)-1`, as tabulated below:

n	Minimum	Maximum
8	0	(2^8)-1 (=255)
16	0	(2^16)-1 (=65,535)
32	0	(2^32)-1 (=4,294,967,295) (9+ digits)
64	0	(2^64)-1 (=18,446,744,073,709,551,615) (19+ digits)

http://www.ntu.edu.sg/home/ehchua/programming/java/datarepresentation.html

REPRESENTATION OF INTEGERS – SIGNED (2'S COMPLEMENT EXAMPLE

Suppose we're working with 8 bit quantities (for simplicity's sake) and suppose we want to find how -28 would be expressed in two's complement notation. First we write out 28 in binary form.

0 0 0 1 1 1 0 0

Then we invert the digits. 0 becomes 1, 1 becomes 0.

1 1 1 0 0 0 1 1

Then we add 1.

1 1 1 0 0 1 0 0

That is how one would write -28 in 8 bit binary.

Mano

REPRESENTATION OF INTEGERS – SIGNED (2'S COMPLEMENT EXAMPLE

An *n*-bit 2's complement signed integer can represent integers from `-2^(n-1)` to `+2^(n-1)-1`, as tabulated. Take note that the scheme can represent all the integers within the range, without any gap. In other words, there is no missing integers within the supported range.

n	minimum	maximum
8	-(2^7) (=-128)	+(2^7)-1 (=+127)
16	-(2^15) (=-32,768)	+(2^15)-1 (=+32,767)
32	-(2^31) (=-2,147,483,648)	+(2^31)-1 (=+2,147,483,647)(9+ digits)
64	-(2^63) (=-9,223,372,036,854,775,808)	+(2^63)-1 (=+9,223,372,036,854,775,807)(18+ digits)

http://www.ntu.edu.sg/home/ehchua/programming/java/datarepresentation.html

BIG ENDIAN VERSUS LITTLE ENDIAN

Modern computers store one byte of data in each memory address or location, i.e., byte addressable memory. An 32-bit integer is, therefore, stored in 4 memory addresses.

The term"Endian" refers to the *order* of storing bytes in computer memory. In "Big Endian" scheme, the most significant byte is stored first in the lowest memory address (or big in first), while "Little Endian" stores the least significant bytes in the lowest memory address.

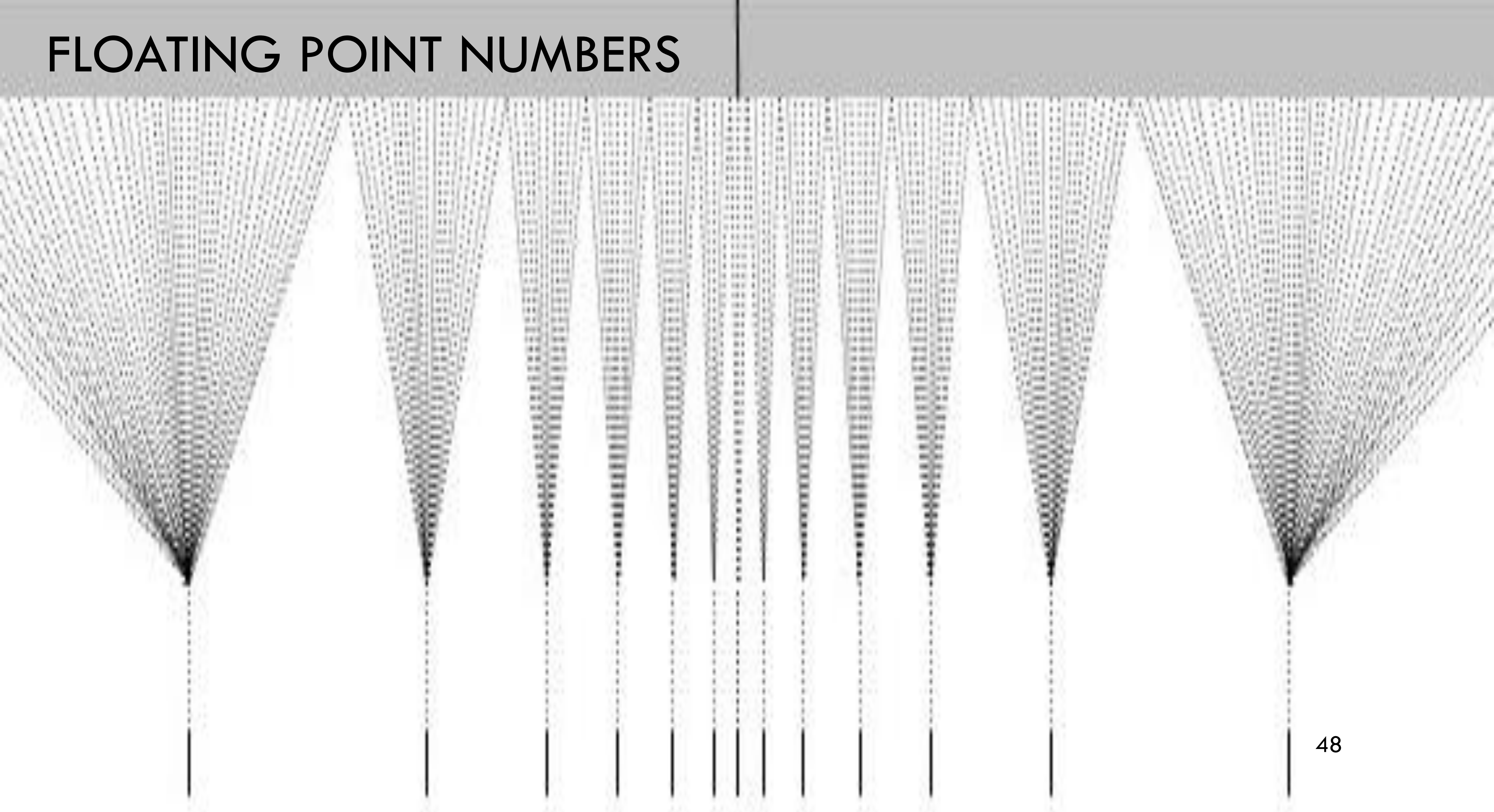
For example, the 32-bit integer 12345678H (305419896_{10}) is stored as 12H 34H 56H 78H in big endian; and 78H 56H 34H 12H in little endian. An 16-bit integer 00H 01H is interpreted as 0001H in big endian, and 0100H as little endian.

http://www.ntu.edu.sg/home/ehchua/programming/java/datarepresentation.html

SECTION SUMMARY EXERCISES

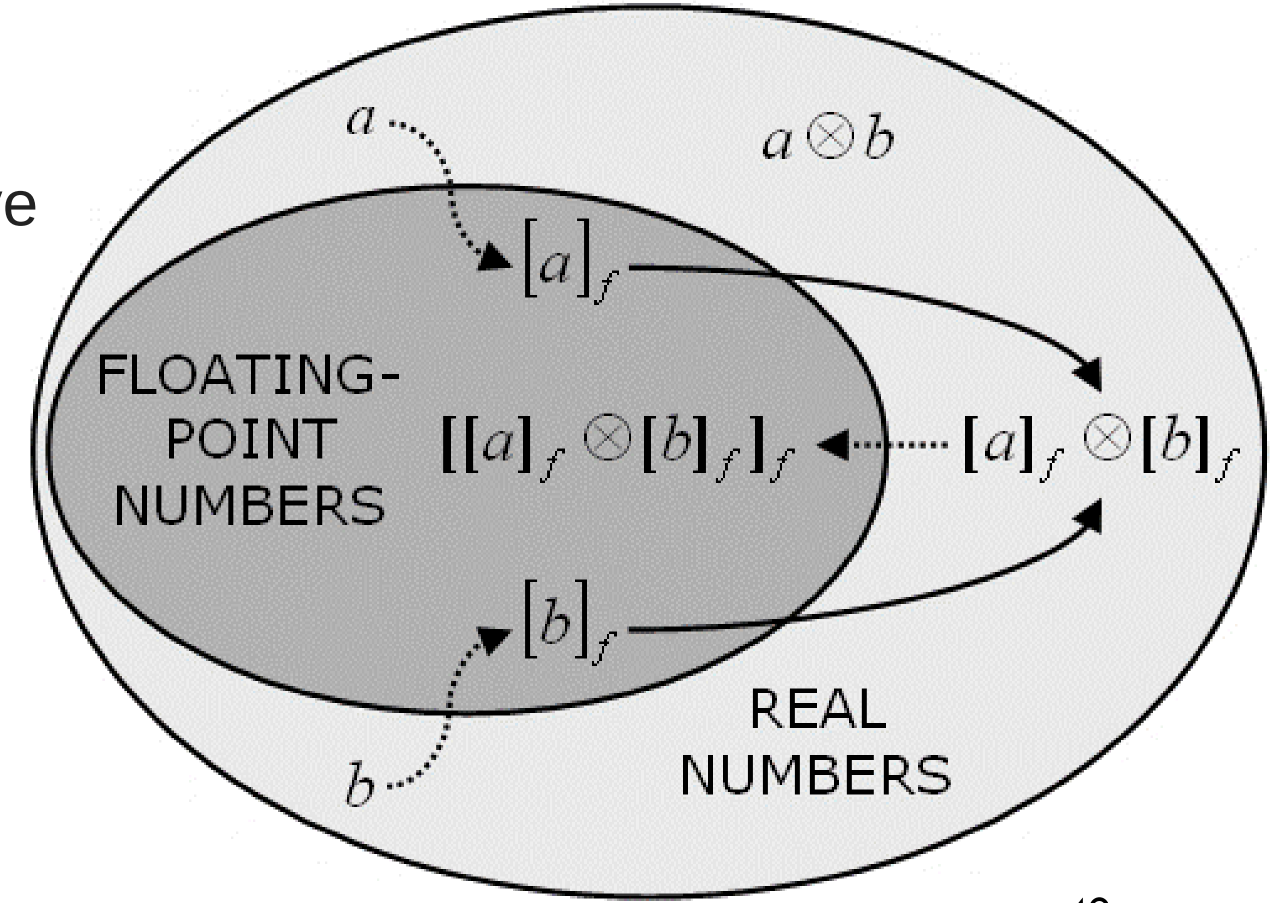
1. What are the ranges of 8-bit, 16-bit, 32-bit and 64-bit integer, in "unsigned" and "signed" representation?
2. Give the value of 88, 0, 1, 127, and 255 in 8-bit unsigned representation.
3. Give the value of +88, -88 , -1, 0, +1, -128, and +127 in 8-bit 2's complement signed representation.
4. Give the value of +88, -88 , -1, 0, +1, -127, and +127 in 8-bit sign-magnitude representation.
5. Give the value of +88, -88 , -1, 0, +1, -127 and +127 in 8-bit 1's complement representation.

FLOATING POINT NUMBERS



FLOATING POINT NUMBERS

Because the size and **number** of registers that any computer can have is limited, only a **subset** of the the **real-number** continuum can be used in **real-number** calculations.



http://jasss.soc.surrey.ac.uk

FLOATING POINT NUMBERS

- Scientific programs rarely survive on integers alone, but representing fractional parts efficiently is complicated.
- Option one: **fixed point:**
  - Set the point at a known location. Anything to the left represents the integer part; anything to the right the fractional part.
  - But where do we set it??
- Option two: **floating point:**
  - Let the point 'float' to give more capacity on its left or right as needed.
  - Much more efficient, but harder to work with.

FLOATING POINT NUMBERS

- Clearly, real numbers are more challenging to represent in a computer than integers.
- We represent real numbers on computers as **floating point decimal numbers:**
  - The term, floating point, means there are no fixed number of digits before and after the decimal point (e.g., the decimal point can *float*).
- Note that real numbers require more computing power to process them – some computers have a *floating point unit* chip (FPU) dedicated to this task.
- **Most floating point numbers are only *approximations* when represented in a computer (good to 1:10^{15}).**

TYPE CONVERSION FUNCTIONS

These functions are compiled inline, meaning the conversion code is part of the code that evaluates the expression. Sometimes there is no call to a procedure to accomplish the conversion, which improves performance. Each function coerces an expression to a specific data type.

Syntax

```
CBool(expression)
CByte(expression)
CChar(expression)
CDate(expression)
CDbl(expression)
CDec(expression)
CInt(expression)
CLng(expression)
CObj(expression)
CSByte(expression)
CShort(expression)
CSng(expression)
CStr(expression)
CUInt(expression)
CULng(expression)
CUShort(expression)
```

Return Value Data Type

The function name determines the data type of the value it returns, as shown in the following table.

Function name	Return data type	Range for `expression` argument
`CBool`	Boolean Data Type	Any valid `Char` or `String` or numeric expression.
`CByte`	Byte Data Type	0 through 255 (unsigned); fractional parts are rounded.[1]
`CChar`	Char Data Type	Any valid `Char` or `String` expression; only first character of a `String` is converted; value can be 0 through 65535 (unsigned).
`CDate`	Date Data	Any valid representation of a date and time.

Microsoft

MORE GENERAL DATA CONVERSION FEATURES

To convert	Use this
Character code to character	Chr
String to lowercase or uppercase	Format, LCase, UCase, String.ToUpper, String.ToLower, String.Format
Date to a number	DateSerial, DateValue
Decimal number to other bases	Hex, Oct
Number to string	Format, Str
One data type to another	CBool, CByte, CDate, CDbl, CDec, CInt, CLng, CObj, CSng, CShort, CStr, Fix, Int
Character to character code	Asc
String to number	Val
Time to serial number	TimeSerial, TimeValue

PRESENTATION TERMINATED