Information and Data

CPSC 217: Introduction to Computer Science for Multidisciplinary Studies I
Fall 2020

Jonathan Hudson, Ph.D
Instructor
Department of Computer Science
University of Calgary

Tuesday, September 8, 2020

UNIVERSITY OF CALGARY

What is Information?

Etymology: Latin, “to give form to” or “to form an idea of”

Definition: The state of being of an object or system of interest

What is Data?

Data: raw facts, representation of information, no context

Encoding: The translation of information into data

(Decoding the other direction

Data represents information

Information Processing

A change of information in any manner detectable by an observer

Using a computer?

Encode information into data	Process the data	Translate data back into information

Moral: computers process **data**, not information – **it is our responsibility to interpret the data correctly.**

Storing Data

All data in a computer is either a 0 or 1

- Called a bit (binary digit)
- Electrically, this is a switch that is either open or closed

Encoding schemes translate integers, real numbers, letters, pictures, … into bits

Boolean Data

- Has two possible values
  - False
  - True
- Easily encoded using a single bit
  - 0: False
  - 1: True

Integer Data

How do we represent the numbers 5, 24, or 367 using only ones and zeros?

Simplest idea:

`11111 = 5`

`11111 11111 11111 11111 1111 = 24`

Not practical for large integers!

Other ideas?

Number Systems

- Decimal (Base 10)
  - 10 distinct symbols (0,1,2,3,4,5,6,7,8,9)
  - Each digit is a factor of 10 larger than the digit to its right

- Examples:

```
5 = 5 x 1
24 = 2 x 10 + 4 x 1
367 = 3 x 100 + 6 x 10 + 7 x 1
```

Number Systems

- Decimal (Base 10)
  - 10 distinct symbols (0,1,2,3,4,5,6,7,8,9)
  - Each digit is a factor of 10 larger than the digit to its right

- Examples:

$5 = 5 \times 10^0$

$24 = 2 \times 10^1 + 4 \times 10^0$

$367 = 3 \times 10^2 + 6 \times 10^1 + 7 \times 10^0$

UNIVERSITY OF CALGARY

Number Systems

THIS IS A POSITIONAL SYSTEM – THE POSITION WITHIN THE NUMBER IMPACTS THE FACTOR BY WHICH THE DIGIT IS MULTIPLIED.

CHOICE OF BASE 10 IS (SOMEWHAT) ARBITRARY – **CAN USE ANY INTEGER BASE >= 1**

NOTE: **THERE IS NOTHING SPECIAL ABOUT BASE 10** – IT'S JUST WHAT WE ARE USED TO!

UNIVERSITY OF CALGARY

Binary Data

Number Systems

Binary (Base 2)

- 2 distinct symbols (0,1)
- Each digit is a factor of 2 larger than the digit to its right

Base 10: hundreds, tens, ones

Base 2: eights, fours, twos, ones

Counting in Binary

```
   0   == 0
   1   == 1
  10   == 2
  11   == 3
 100   == 4
 101   == 5
 110   == 6
 111   == 7
1000   == 8
```

- You can see how when we have a single 1 in a column (ones, two, fours, eights) that it’s equivalent to that number in decimal (base 10)

Binary Numbers

- Consider the base 2 number 1001101_2

```
1:   ones (2^0)
0:   twos (2^1)
1:   fours (2^2)
1:   eights (2^3)
0:   sixteens (2^4)
0:   thirty-twos (2^5)
1:   sixty-fours (2^6)
```

Binary Numbers

- Consider the base 2 number 1001101_2

```
1:   ones (2^0)
0:   twos (2^1)
1:   fours (2^2)
1:   eights (2^3)
0:   sixteens (2^4)
0:   thirty-twos (2^5)
1:   sixty-fours (2^6)
```

- *$1 \times 2^0 + 1 \times 2^2 + 1 \times 2^3 + 1 \times 2^6 = 1 + 4 + 8 + 64 =$* $\mathbf{77_{10}}$ (base specified as a subscript)

Binary <-> Decimal

Binary to Decimal

- Convert 1111_2 to base 10:

- Convert 100010_2 to base 10:

- Convert 0_2 to base 10:

Binary to Decimal

- Convert 1111_2 to base 10:

$$1 \times 2^0 + 1 \times 2^1 + 1 \times 2^2 + 1 \times 2^3 = 1 + 2 + 4 + 8 = 15_{10}$$

- Convert 100010_2 to base 10:

$$1 \times 2^1 + 1 \times 2^5 = 2 + 32 = 34_{10}$$

- Convert 0_2 to base 10:

$$0_{10}$$

The Division Algorithm

- Allows us to convert from Decimal to Binary

```
Let Q represent the number to convert
Repeat
   Divide Q by 2, recording the Quotient, Q, and the remainder, R
Until Q is 0
Read the remainders from bottom to top
```

- Divide by the base to which we want to convert (algorithm works for conversion from decimal to **any** base)

Decimal to Binary

- Convert 191_{10} to Binary:

```
191 / 2 = 95, remainder 1
95 / 2 = 47, remainder 1
47 / 2 = 23, remainder 1
23 / 2 = 11, remainder 1
11 / 2 = 5, remainder 1
5 / 2 = 2, remainder 1
2 / 2 = 1, remainder 0
1 / 2 = 0, remainder 1
```

Decimal to Binary

- Convert 191_{10} to Binary:

```
191 / 2 = 95, remainder 1
95 / 2 = 47, remainder 1
47 / 2 = 23, remainder 1
23 / 2 = 11, remainder 1
11 / 2 = 5, remainder 1
5 / 2 = 2, remainder 1
2 / 2 = 1, remainder 0
1 / 2 = 0, remainder 1
```

- Reading from bottom to top: $1011\ 1111_2$
- ***Check:*** $1 + 2^1 + 2^2 + 2^3 + 2^4 + 2^5 + 2^7 = 1 + 2 + 4 + 8 + 16 + 32 + 128 = 191_{10}$

Integer Data

Integer Data

- Base 10 integers can be represented using sequences of bits
  - Common sizes:
    - 8 bits (referred to as a **byte**)
    - 32 bits (referred to as a **word**)
    - 64 bits (referred to as a **double word** / **long**)
    - 16 bits (referred to as a **half word** / **short**)

Integer Data

- Base 10 integers can be represented using sequences of bits

- **Byte** [8 bits]: 0000 0000 – 1111 1111 (0 to $2^8 - 1$)
- **Word** [32 bits]: 0 to $2^{32} - 1$
- **Double word** (**long**) [64 bits]: 0 to $2^{64} - 1$
- **Half word** (**short**) [16 bits]; 0 to 2^{16} - 1

Negative Numbers

- **Simple idea is called "Signed Magnitude".**
- Idea (SM byte): right-most 7 bits represent the magnitude, **8th bit represents the sign.**

- Example:

$65_{10} = 100\ 0001_2$

```
+65 as a byte:  0100 0001
-65 as a SM byte:  1100 0001
```


UNIVERSITY OF CALGARY

Other Bases

Other Bases

- A number system can have any base
  - **Decimal: Base 10 (0,1,2,3,4,5,6,7,8,9)**
  - **Binary: Base 2 (0,1)**
  - Octal: Base 8 (0,1,2,3,4,5,6,7)
  - **Hexadecimal: Base 16 (0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f)**
  - Vigesimal: Base 20 (0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f,g,h,I,j)
  - Base 6 (0,1,2,3,4,5)
  - Any other number we choose…

Hexadecimal

- Convert 0xA1 to decimal:

- Convert 44 base 16 to decimal:

- Convert $CAFE_{16}$ to base 10:

Hexadecimal

- Convert 0xA1 to decimal:

  $A \times 16^1 + 1 \times 16^0$ =

  $10 \times 16^1 + 1 \times 16^0$ =

  160 + 1 =

  161_{10}

- Convert 44 base 16 to decimal:

  $4 \times 16^1 + 4 \times 16^0$ =

  64 + 4 =

  68_{10}

- Convert $CAFE_{16}$ to base 10:

  $C \times 16^3 + A \times 16^2 + F \times 16^1 + E \times 16^0$ =

  $12 \times 16^3 + 10 \times 16^2 + 15 \times 16^1 + 14 \times 16^0$ =

  12 x 4095 + 10 x 256 + 15 x 16 + 14 x 1 =

  51966_{10}

Hexadecimal

- Convert 507_{10} to base 16:
- Use division method with 16 instead of 2:

Hexadecimal

- Convert 507_{10} to base 16:
- Use division method with 16 instead of 2:

```
507/16 = 31, remainder 11 = B
31/16 = 1, remainder 15 = F
1/16 = 0, remainder 1
```

Hexadecimal

- Convert 507_{10} to base 16:
- Use division method with 16 instead of 2:

```
507/16 = 31, remainder 11 = B
31/16 = 1, remainder 15 = F
1/16 = 0, remainder 1
```

- Reading from bottom to top: $1FB_{16}$
- **Check your work:**

$1 \times 16^2 + F \times 16^1 + B \times 16^0 = 1 \times 16^2 + 15 \times 16^1 + 11 \times 16^0 = 256 + 240 + 11 = 507_{10}$

Utility of Hexadecimal

- Common to have groups of 32 bits
  - 32 bits is cumbersome to write
  - easy to make mistakes

- Use hexadecimal as a shorthand
  - 8 hex digits instead of 32 bits
  - Group bits from the right
  - Memorize mapping from binary to hex for values between 0 and F

Utility of Hexadecimal

Convert 0xF51A to binary

Convert 10010010101010110101 00 from binary to hex

Utility of Hexadecimal

Convert 0xF51A to binary

F=1111_2, 5 = 0101_2, 1 =0001_2, A=1010_2

1111 0101 0001 1010_2

Convert 1001001010101011010100 from binary to hex

10 0100 1010 1010 1101 0100

0010=2 0100=4 1010=10 1010=10 1101=13 0100=4

0010=2 0100=4 1010=a 1010=a 1101=d 0100=4

0x24aad4

Character Data

Representing Characters

- **Standard encoding scheme called ASCII**
  - **American Standard Code for Information Interchange**
    - **7 bits per character** (2^7 = 128 possible characters)
  - Includes printable characters
  - Includes “control characters” that impact formatting (tab, newline), data transmission (mostly obsolete)
  - **Layout seems arbitrary, but actually contains some interesting patterns**

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Representing More Characters

- Limitation of ASCII?
  - Only supports Latin character set
  - No support for accents, additional character sets
  - Solutions?

Representing More Characters

- **UTF-8**
  - Another encoding scheme for characters
    - **Variable length – 1, 2, 3 or 4 bytes per character**
  - **Compatible with ASCII**
  - Consider each byte
    - **Left most bit is 0? Usual ASCII Character**
    - Left most bits are 110? 2 byte character
    - Left most bits are 1110? 3 byte character
    - Left most bits are 11110? 4 byte character

    - \xF0\x9F\x98\x82 → tears of joy

UTF-8

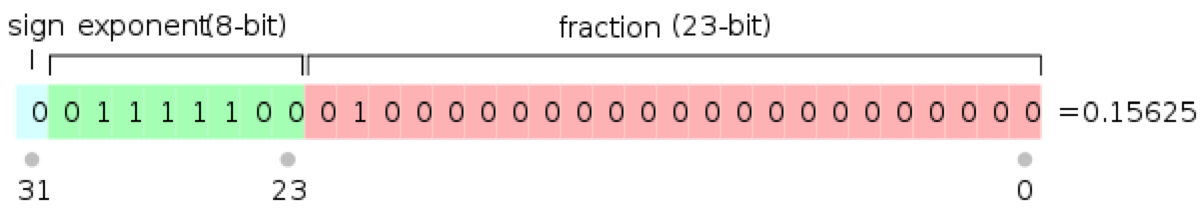
Number of bytes	Bits for code point	First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
1	7	U+0000	U+007F	0xxxxxxx			
2	11	U+0080	U+07FF	110xxxxx	10xxxxxx		
3	16	U+0800	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
4	21	U+10000	U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

Decimal Point Numbers

Representing Real Numbers

- Standard Representation: IEEE 754 Floating Point
  - **Express the number in scientific notation**
  - **-0.0002589 becomes -2.589 * 10^{-4}**
- Need to store **sign**, **exponent**, and **mantissa** (the fraction)
- 32-bit floating point representation:
- **sign (1 bit), exponent (8 bits), mantissa (23 bits)**
- 64-bits:
- sign (1 bit), exponent (11 bits), mantissa (52 bits)

IEEE 754 – 32 Bit



Problems with Real Numbers

- How many real numbers are there? **Infinity**
- How many real numbers are there between 0 and 1? **Infinity**
- How many values can be represented by 32 or 64 bits?
- **2^{32} = 4.2 billion,**
- **2^{64} = 1.8 x 10^{19}**
- **Largest values: $2^{32} – 1$ and $2^{64} – 1$**
- What's the problem?

Problems with Real Numbers

- Problem: some real numbers exist that cannot be represented exactly in floating point
- (eg. 1/3 = 0.3333333…., sqrt(2) = 1.414213…).

- Thus floating point numbers only **approximate** real numbers (and maintaining accuracy is a very important concern!).

UNIVERSITY OF CALGARY

Image Data

Encoding Images

- Common Techniques
  - **Vector Images**
    - Vector images: “line work” Image is encoded as a collection of geometric primitives such as points, lines, curves.
  - **Raster Images**
    - Raster images: constructed from a grid of pixels (picture elements), where each picture is assigned a color

Representing Colors

- How do we represent a color as a sequence of bits?
- Can represent almost any color as a combination of some red, some green, and some blue. Typically use a scale from 0 (no light of that color) to 255 (full on for that color). Yields 256x256x256 = 16 million different possible colors.
  - (256 = 16*16   or two hex symbols)
- To represent an image: 3 color components for **each pixel** (becomes a lot of bytes very quickly!)

UNIVERSITY OF CALGARY

Videos

- Raster image storage formats like jpg heavily use ‘compression’ to reduce storage size
  - Basic ideas, reduce quantity of colours stored, and group idea of ‘where colours are’ to store less information
- Video compression works similar but since video is a sequence of frames where each frame is an image, they also make use of reducing data by grouping idea of ‘colours stay the same and where’ across multiple frames
  - Great example of compression failure → confetti
  - When confetti is in image, the colour of spot changes every frame and nearby spots are different each frame
  - This means more info is needed per frame, as a result at the same data rate, the image quality will go down (boxy artifacts will appear, or even decoding breaks down)
  - This is the same reasons sports struggle with compressed video

Onward to … decisions.

Jonathan Hudson
jwhudson@ucalgary.ca
https://pages.cpsc.ucalgary.ca/~hudsonj/

UNIVERSITY OF CALGARY