2. Data Representation

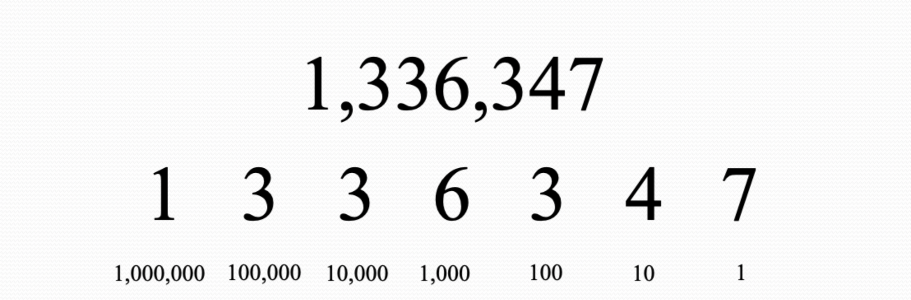
CT1100 - J. Duggan

Lecture Overview

- "There are 10 kinds of people in the world - those who understand binary and those who don't" (Anonymous)
- The most basic unit of information in a digital computer is called a bit **bi**nary digi**t**
- A bit is nothing more than a state of "on" or "off"
- Topics
  - Positional Numbering Systems
  - Base 10, Base 2 and Base 16
  - Conversion between bases
  - Unsigned whole numbers
  - Character Codes (ASCII and Unicode)

Lecture Overview

- All numbering systems use positional notation
- Number denoted by its absolute value, multiplied by its positional value



Decimal Numbering Systems

- Decimal Base 10 / Radix 10
- For example 973_{10}
- The set of valid numerals for a positioning numbering system is equal in size to teh radix of the system
- There are 10 digits in the decimals system (0-9), 2 in binary (0-1)

$$973_{10} = 9\text{x}10^2 + 7\text{x}10^1 + 3\text{x}10^0$$

$$216_{10} = 2\text{x}10^2 + 1\text{x}10^1 + 6\text{x}10^0$$

00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29

Binary Numbering Systems

- Two valid numerals (0 and 1)
- Base 2

$$011010_2 = 0x2^5+1x2^4+1x2^3+0x2^2+1x2^1+0x2^0$$

$$001111_2 = 0x2^5+0x2^4+1x2^3+1x2^2+1x2^1+1x2^0$$

000000 000001 000010 000011 000100
000101 000110 000111 001000 001001

The Scale of bits/bytes

Size	Term
Binary digit	bit
8 bits	byte
1024 (2^{10}) bytes	kilobyte
1,000 kilobytes	megabyte (million bytes)
1,000 megabytes	gigabyte (billion bytes)
1,000 gigabyte	terabyte (trillion bytes)

- A binary number with N bits can represent **unsigned integers** from 0 to 2^N-1
- For example, 15_{10} can be represented as 1111_2

Challenge 2.1

- Convert 10010011_2 to decimal
- Confirm the result with R (for example)

```r
library(binaryLogic)

n1 <- as.binary(111,n = 8)
n1
```

```
## [1] 0 1 1 0 1 1 1 1
```

- What is the maximum integer value (unsigned) for a 16 bit value?

Convert decimal to ternary

- Division/remainder technique, e.g. 104_{10} to base 3
- 104 / 3 = 34 with a remainder of 2
- 34 / 3 = 11 with a remainder of 1
- 11 / 3 = 3 with a remainder of 2
- $3/3 = 1$ with a remainder of 0
- $1/3 = 0$, with a remainder of 1
- Reading the remainders from bottom to top we have $104_{10} = 10212_3$

Challenge 2.2

- Convert 15_{10} to binary
- Confirm the result with R

```r
library(binaryLogic)

n1 <- as.binary(15,n = 4)
```

Binary Addition

- Facts for binary addition
  - $0 + 0 = 0$
  - $0 + 1 = 1$
  - $1 + 0 = 1$
  - $1 + 1 = 0$ 'and carry 1' $= 10$
  - $1 + 1 +$ [carry] $= 1$ 'and carry 1'
- Need to make sure there is enough space (otherwise "overflow")

Decimal	Binary	
19	10011	00111010
+ 17	+10001	+00011011

Binary Subtraction

- Our desire with computer arithmetic is to minimise the types of operations required
  - so we have devised a way to subtract using addition!
- We do this using complementary arithmetic
  - instead of a - b, think of a + (-b)
  - we find a way to express a number as negative, then add it
- To express a binary number as negative, get the *twos complement*
  - flip the bits and add 1
  - Then add, and throw away any high order bits

	Decimal	Binary	Twos Complement	Delete Higher Bit
	8	1000	1000	
-	3	0011 →	1101	
=	5		10101 →	0101

Challenge 2.3

- Add 4_{10} and 11_{10} in binary
- Subtract 8_{10} from 15_{10}

Representing Text

- ASCII
  - American Standard Code for Information Interchange
  - 8-bit, limited character set - built around Latin alphabet
  - 32 control characters, 10 digits, 52 letters (upper & lower), 32 special characters and the space character.
- Unicode
  - attempts to represent characters in every language
  - Uses 16 bits per character and is a superset of ASCII

ASCII Codes (0-127)

Dec	Hex	Name	Char	Ctrl-char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	Null	NUL	CTRL-@	32	20	Space	64	40	@	96	60	`
1	1	Start of heading	SOH	CTRL-A	33	21	!	65	41	A	97	61	a
2	2	Start of text	STX	CTRL-B	34	22	"	66	42	B	98	62	b
3	3	End of text	ETX	CTRL-C	35	23	#	67	43	C	99	63	c
4	4	End of xmit	EOT	CTRL-D	36	24	$	68	44	D	100	64	d
5	5	Enquiry	ENQ	CTRL-E	37	25	%	69	45	E	101	65	e
6	6	Acknowledge	ACK	CTRL-F	38	26	&	70	46	F	102	66	f
7	7	Bell	BEL	CTRL-G	39	27	'	71	47	G	103	67	g
8	8	Backspace	BS	CTRL-H	40	28	(	72	48	H	104	68	h
9	9	Horizontal tab	HT	CTRL-I	41	29	)	73	49	I	105	69	i
10	0A	Line feed	LF	CTRL-J	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	VT	CTRL-K	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	FF	CTRL-L	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage feed	CR	CTRL-M	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	SO	CTRL-N	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	SI	CTRL-O	47	2F	/	79	4F	O	111	6F	o
16	10	Data line escape	DLE	CTRL-P	48	30	0	80	50	P	112	70	p
17	11	Device control 1	DC1	CTRL-Q	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	DC2	CTRL-R	50	32	2	82	52	R	114	72	r
19	13	Device control 3	DC3	CTRL-S	51	33	3	83	53	S	115	73	s
20	14	Device control 4	DC4	CTRL-T	52	34	4	84	54	T	116	74	t
21	15	Neg acknowledge	NAK	CTRL-U	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	SYN	CTRL-V	54	36	6	86	56	V	118	76	v
23	17	End of xmit block	ETB	CTRL-W	55	37	7	87	57	W	119	77	w
24	18	Cancel	CAN	CTRL-X	56	38	8	88	58	X	120	78	x
25	19	End of medium	EM	CTRL-Y	57	39	9	89	59	Y	121	79	y
26	1A	Substitute	SUB	CTRL-Z	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	ESC	CTRL-[	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	FS	CTRL-\	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	GS	CTRL-]	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	RS	CTRL-^	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	US	CTRL-_	63	3F	?	95	5F	_	127	7F	DEL

ASCII Codes (128-255)

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
128	80	Ç	160	A0	á	192	C0	└	224	E0	α
129	81	ü	161	A1	í	193	C1	┴	225	E1	ß
130	82	é	162	A2	ó	194	C2	┬	226	E2	Γ
131	83	â	163	A3	ú	195	C3	├	227	E3	π
132	84	ä	164	A4	ñ	196	C4	─	228	E4	Σ
133	85	à	165	A5	Ñ	197	C5	┼	229	E5	σ
134	86	å	166	A6	ª	198	C6	╞	230	E6	µ
135	87	ç	167	A7	º	199	C7	╟	231	E7	τ
136	88	ê	168	A8	¿	200	C8	╚	232	E8	Φ
137	89	ë	169	A9	⌐	201	C9	╔	233	E9	Θ
138	8A	è	170	AA	¬	202	CA	╩	234	EA	Ω
139	8B	ï	171	AB	½	203	CB	╦	235	EB	δ
140	8C	î	172	AC	¼	204	CC	╠	236	EC	∞
141	8D	ì	173	AD	¡	205	CD	═	237	ED	φ
142	8E	Ä	174	AE	«	206	CE	╬	238	EE	ε
143	8F	Å	175	AF	»	207	CF	╧	239	EF	∩
144	90	É	176	B0	░	208	D0	╨	240	F0	≡
145	91	æ	177	B1	▒	209	D1	╤	241	F1	±
146	92	Æ	178	B2	▓	210	D2	╥	242	F2	≥
147	93	ô	179	B3	│	211	D3	╙	243	F3	≤
148	94	ö	180	B4	┤	212	D4	╘	244	F4	⌠
149	95	ò	181	B5	╡	213	D5	╒	245	F5	⌡
150	96	û	182	B6	╢	214	D6	╓	246	F6	÷
151	97	ù	183	B7	╖	215	D7	╫	247	F7	≈
152	98	ÿ	184	B8	╕	216	D8	╪	248	F8	°
153	99	Ö	185	B9	╣	217	D9	┘	249	F9	∙
154	9A	Ü	186	BA	║	218	DA	┌	250	FA	·
155	9B	¢	187	BB	╗	219	DB	█	251	FB	√
156	9C	£	188	BC	╝	220	DC	▄	252	FC	ⁿ
157	9D	¥	189	BD	╜	221	DD	▌	253	FD	²
158	9E	₧	190	BE	╛	222	DE	▐	254	FE	■
159	9F	ƒ	191	BF	┐	223	DF	▀	255	FF	

Challenge 2.4

Show that the following code makes sense, by converting the hexadecimal numbers to decimal.

```r
charToRaw("Hello World")
```

```
##  [1] 48 65 6c 6c 6f 20 57 6f 72 6c 64
```

Unicode

Types	Description	# Chars	Hex Vals
Alphabets	Latin, Cyrillic, Greek etc.	8192	0000-1FFF
Symbols	Dingbats, Mathematical etc.	4096	2000-2FFF
CJK	Chinese, Japenese, Korean	40,960	4000-DFFF
Han	Unified Chinese, Japenese, Korean	4096	3000-3FFF
	Expansion/spillover from Han	4096	E000-EFFF
User defined	Unified Chinese, Japenese, Korean	4095	F000-FFFE

Challenge 2.5

What kind of data might each of these variables be?

Time	Team	Scorer	From	Type	Points	Score
1	Dublin	Paul Mannion	Play	Point	1	1
2	Kerry	Sean O'Shea	Play	Point	1	1
3	Dublin	Dean Rock	Play	Point	1	2
4	Dublin	Dean Rock	Free	Point	1	3
10	Kerry	David Clifford	Play	Point	1	2
13	Kerry	Sean O'Shea	FortyFive	Point	1	3
14	Kerry	Stephen O'Brien	Play	Point	1	4
16	Dublin	Paul Mannion	Play	Point	1	4
18	Kerry	Sean O'Shea	Free	Point	1	5
19	Dublin	Jack McCaffrey	Play	Goal	3	7

Summary

- Key ideas
  - Positional numbering systems
  - Base 10, 16 and 2
  - Conversion processes
  - Addition & Subtraction of unsigned integers
  - Character codes
- Other Topcics
  - Signed integers and negative numbers
  - Floating point representation
  - Multiplication and division
  - IEEE Floating Point Standard
- Next steps
  - High level languages hides many of these details
  - Easier to work with numbers and characters
  - We will explore these in R, e.g. variable types in a **tibble**