CSC 101
Fluency with Information Technology and Computing

Chapter 7
Representing Data Digitally

Brian McBride
Department of Computer Science, Engineering and Physics (CSEP)
University of Michigan - Flint

Email: brmcbrid@umflint.edu

Chapter 7

Representing Information Digitally

Digitizing Discrete Information

- *Digitize*: Represent information with digits (normally base 10 numerals 0 through 9)
- Limitation of Digits
  - Alternative Representation: Any set of symbols could represent phone number digits, as long as the keypad is labeled accordingly
- Symbols, Briefly
  - Digits have the advantage of having short names (easy to say)
    - But computer professionals are shortening symbol names (exclamation point is pronounced "bang")

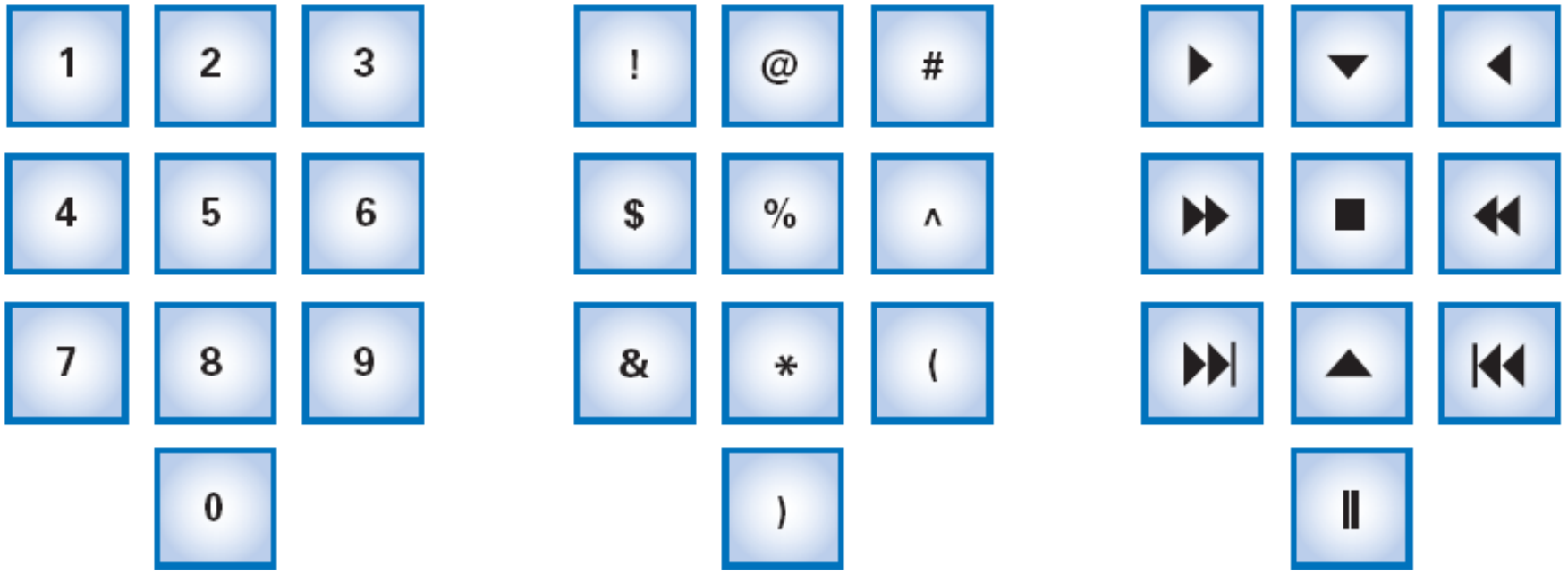


Figure 8.1 Three symbol assignments for a telephone keypad.

Ordering Symbols

- Advantage of digits for encoding info is that items can be listed in numerical order
- To use other symbols, we need an ordering system (*collating sequence*)
  - Agreed order from smallest to largest value
- In choosing symbols for encoding, consider how symbols interact with things being encoded

The Fundamental Representation of Information

- The fundamental patterns used in IT come when the physical world meets the logical world
- The most fundamental form of information is the presence or absence of a physical phenomenon
  - Is matter in a particular place is space and time?
  - Is light detected at a particular place and time? Or not?
  - Is magnetism sensed at a spot at a time? Or not?
- We don't care how much we sense (how bright the light is, how strong the magnetic pull is) we only care if it's there (or not)

The Fundamental Representation of Information

- In the logical world, the concepts of *true* and *false* are important
- Logic is the foundation of reasoning (human and mechanical/computing)
- By associating *true* with the presence of a phenomenon and *false* with its absence, we use the physical world to implement the logical world, and produce information technology
- We will refer to such association as a *PandA* (Presence and Absence) representation

The PandA Representation

- *PandA* is the mnemonic for "presence and absence" of some physical thing
- It is *discrete* (distinct or separable)—the phenomenon is present or it is not… there is no continuous gradation in between
- Give any names to the two basic PandA patterns as long as we are consistent (see table following)
- Sequences of P and A values are used to form symbols

Table 8.1 Possible interpretations of the two PandA patterns

Present	Absent
True	False
1	0
On	Off
Yes	No
+	–
Black	White
For	Against
Yang	Yin
X	Y
. . .	. . .

A Binary System

- Use of two basic patterns make a *binary system*
- The basic binary unit is known as a *"bit"* (short for binary digit)
- Bit values 0 and 1 are the most common form of PandA representation
- Bit sequences this way look like binary numbers from math (base 2) but we use them as general symbols
- 8 bits are grouped together to form a *byte*
  - *More on this special grouping later*

Combining Bit Patterns

- Since we only have two bit values, the bits alone can only represent two things, like (won, lost) or (yes,no)
- If we group bit values into sequences we can represent enough symbols to encode complex information (one sequence per symbol)
- Arranging 2 bit values into *n*-length sequences, we can create 2^n symbols

Table 8.2 Number of symbols when the number of possible patterns is two

n	2^n	Symbols
1	2^1	2
2	2^2	4
3	2^3	8
4	2^4	16
5	2^5	32
6	2^6	64
7	2^7	128
8	2^8	256
9	2^9	512
10	2^{10}	1,024

Figure 8.2 Sidewalk sections as a sequence of bits (1010 0010).

Using "P" for stone presence and "A" for stone absence, this 8-square sidewalk sequence forms the symbol "PAPA AAPA"

Table 8.3 Sixteen symbols of the 4-bit PandA representation

Symbol	Binary	Physical Bits	Hex	Symbol	Binary	Physical Bits	Hex
AAAA	0000		0	PAAA	1000		8
AAAP	0001		1	PAAP	1001		9
AAPA	0010		2	PAPA	1010		A
AAPP	0011		3	PAPP	1011		B
APAA	0100		4	PPAA	1100		C
APAP	0101		5	PPAP	1101		D
APPA	0110		6	PPPA	1110		E
APPP	0111		7	PPPP	1111		F

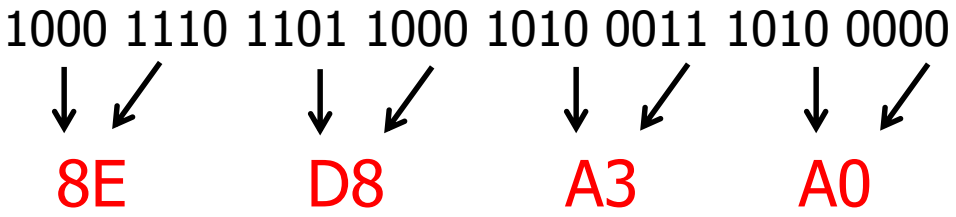
The 16 Hex Digits

- 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
  - A = 10, B = 11, … , F = 15
- Sixteen values can be represented perfectly by 4-bit sequences ($2^4 = 16$)
- Changing hex digits to bits and back again:
  - Given a sequence of bits, group them in 4's and write the corresponding hex digit
    - 0101 1100

      5 C
  - Given hex, write the associated group of 4 bits

Hex Explained

- Why use hex? Writing the sequence of bits is long, tedious, and error-prone



Hex (0-9,A-F)

Decimal	Hex	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0100
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Converting Between Decimal and Binary

114	0	1	1	1	0	0	1	0
- 64	128	64	32	16	8	4	2	1
50	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
- 32								
18								
- 16								
2								
- 2								
0								

Converting Between Decimal and Binary

54	0	0	1	1	0	1	1	0
- 32	128	64	32	16	8	4	2	1
22								
- 16								
6								
- 4								
2								
- 2								
0								

Digitizing Text

- Early binary representation—1 and 0—encoded numbers and keyboard characters
- Now representation for sound, video, and other types of information are also important
- For encoding text, what symbols should be included?
  - We want to keep the list small enough to use fewer bits, but we don't want to leave out critical characters

Assigning Symbols

- 26 uppercase and 26 lowercase Roman letters, 10 Arabic numerals, 10 arithmetic characters, 20 punctuation characters (including space), and 3 non-printable characters (new line, tab, backspace) = 95 characters, enough to represent English
- For 95 symbols, we need 7-bit sequences
  - $2^6 = 64$ $2^7 = 128$
- ASCII, an early standard 7-bit code, still used
  - *American Standard Code for Information Interchange*

Extended ASCII: An 8-bit Code

- 7-bit ASCII (132 characters) is not enough to represent non-English text
- Mid-60s, IBM extended ASCII to 8 bits (256 symbols), one 8-bit "byte" for one character
- Called "*Extended ASCII*," the first half is original 7-bit ASCII with a 0 added at the beginning of each byte… room for 132 more symbols
- Handles most Western languages and additional useful symbols

ASCII	0000 0000	0000 0001	0000 0010	0000 0011	0000 0100	0000 0101	0000 0110	0000 0111	0000 1000	0000 1001	0000 1010	0000 1011	0000 1100	0000 1101	0000 1110	0000 1111
0000	$^{N}_{U}$	$^{S}_{H}$	$^{S}_{X}$	$^{E}_{X}$	$^{E}_{T}$	$^{E}_{Q}$	$^{A}_{K}$	$^{B}_{L}$	$^{B}_{S}$	$^{H}_{T}$	$^{L}_{F}$	$^{Y}_{T}$	$^{F}_{F}$	$^{C}_{R}$	$^{S}_{O}$	$^{S}_{I}$
0001	$^{D}_{L}$	$^{D}_{1}$	$^{D}_{2}$	$^{D}_{3}$	$^{D}_{4}$	$^{N}_{K}$	$^{S}_{Y}$	$^{E}_{\Sigma}$	$^{C}_{N}$	$^{E}_{M}$	$^{S}_{B}$	$^{E}_{C}$	$^{F}_{S}$	$^{G}_{S}$	$^{R}_{S}$	$^{U}_{S}$
0010		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0011	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0101	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
0110	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0111	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	$^{D}_{T}$
1000	$^{S}_{0}$	$^{S}_{1}$	$^{S}_{2}$	$^{S}_{3}$	$^{I}_{N}$	$^{N}_{L}$	$^{S}_{S}$	$^{E}_{S}$	$^{H}_{S}$	$^{H}_{J}$	$^{Y}_{S}$	$^{P}_{D}$	$^{P}_{V}$	$^{R}_{I}$	$^{S}_{2}$	$^{S}_{3}$
1001	$^{D}_{C}$	$^{P}_{1}$	$^{P}_{Z}$	$^{S}_{E}$	$^{C}_{C}$	$^{M}_{M}$	$^{S}_{P}$	$^{E}_{P}$	$^{O}_{S}$	$^{O}_{O}$	$^{O}_{A}$	$^{C}_{S}$	$^{S}_{T}$	$^{O}_{S}$	$^{P}_{M}$	$^{A}_{P}$
1010	$^{A}_{O}$	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
1011	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
1100	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
1101	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
1110	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
1111	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Figure 8.3 ASCII, the American Standard Code for Information Interchange.

(An Aside) On Beyond ASCII

- Computing in the 21st century uses many languages with many different character sets
- 1990s, ***Unicode*** was created
  - used 2 bytes (16 bits) to represent over 65,000 characters (2^16)
  - Allows all modern scripts (Kanji, Arabic, Cyrillic, Hebrew, etc.)
  - Contained 8-bit ASCII as the low 256 characters for compatibility
- Current Unicode standards use even more bits to represent over 107,000 characters in 90 scripts
  - Allows ancient scripts like Egyptian hieroglyphics

ASCII Coding of Phone Numbers

- How would a computer represent in its memory, the phone number 888 555 1212?
- We want it stored not as a single large *number*, but as string of symbols telling what phone buttons to press
- Encode each digit with its ASCII byte

```
8        8        8        5        5        etc.
00111000 00111000 00111000 00110101 00110101 etc.
```

Final Tidbit: Why "bYte"?

- Why is BYTE spelled with a Y?
- Engineers at IBM were looking for a word for a quantity of memory between a bit and a word (usually 32 bits)
- Bite seemed appropriate, but typing errors could make a "bite" into a "bit"
- The "y" prevents this potential confusion

Summary

- Digitizing does not require digits… any symbols will do
- PandA encoding is creating patterns based on the *presence* and *absence* of physical phenomena, the two states of a *bit (binary digit)*
- Bit states are often represented as 0 and 1, but can be any two opposite terms or values
- 7-bit ASCII is an early assignment of bit sequences (symbols) to keyboard characters; extended, or 8-bit ASCII is the current standard
- Eight bit patterns for numbers can represent the quantity of 8 or the keyboard character of 8 and will not be the same pattern. As a result eight bit patterns must have their context identified before we know what they are encoding.
- *Byte* was historically spelled with a "y" to avoid confusion with "bit"