# Chapter 8: Bits and the "Why" of Bytes: Representing Information Digitally

**Fluency with Information Technology**
**Third Edition**

**by**
**Lawrence Snyder**



## 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



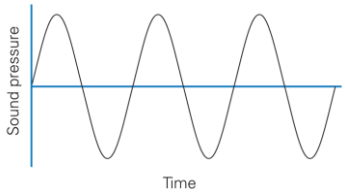

## The Fundamental Representation of Information

- The fundamental patters 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
- In the logical world, the concepts of true and false are important
  - 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



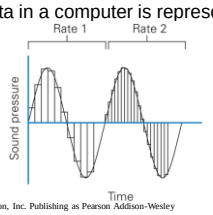

## Analog vs. Digital

- Analog is continuous data/information
  - Sound waves



## Analog vs. Digital

- Digital is discrete data/information
  - Many distinct samples of data
  - Stored in binary (0's and 1's)
    - All data in a computer is represented in binary







## The PandA Representation

- *PandA* is the mnemonic for "presence and absence"
- It is *discrete* (distinct or separable)—the phenomenon is present or it is not (true or false; 1 or 0). There in no continuous gradation in between.




## A Binary System

- Two patterns make a *binary system*
  - Base 2 (0 or 1)
- The basic binary unit is known as a *"bit"* (short for binary digit)
- 8 bits are grouped together to form a *byte*
  - Memory accessed by byte addresses
- We can give any names to these two patterns as long as we are consistent
  - PandA (Presence and Absence can represent 1 and 0, respectively)

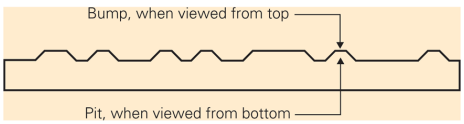



**Table 8.1** Possible interpretations of the two PandA patterns

| Present | Absent |
|---|---|
| True | False |
| 1 | 0 |
| On | Off |
| Yes | No |
| + | – |
| Black | White |
| For | Against |
| Yang | Yin |
| Lisa | Bart |
| . . . | . . . |



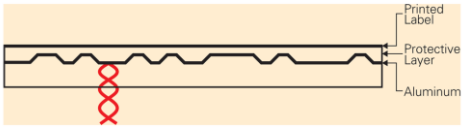

## Encoding Bits on a CD-ROM







## Encoding Bits on a CD-ROM



## Bits in Computer Memory

- Memory is arranged inside a computer in a very long sequence of bits (places where a phenomenon can be set and detected)
- Analogy: Sidewalk Memory
  - Each sidewalk square represents a memory slot (bit), and stones represent the presence or absence
  - If a stone is on the square, the value is 1, if not the value is 0




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




## Alternative PandA Encodings

- There are other ways to encode two states using physical phenomena
  - Use stones on all squares, but black stones for one state and white for the other
  - Use multiple stones of two colors per square, saying more black than white means 0 and more white than black means 1
  - Stone in center for one state, off-center for the other
  - etc.




## Combining Bit Patterns

- Since we only have two patterns, we must combine them into sequences to create enough symbols to encode necessary information
- Binary (PandA) has 2 patterns, arranging them 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 |




## Hex Explained

- Recall in Chapter 4, we specified custom colors in HTML using hex digits
  - e.g., <font color ="#FF8E2A">
  - *Hex* is short for hexadecimal, base 16
- Why use hex? Writing the sequence of bits is long, tedious, and error-prone

## 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 (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 |




## 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
  - $2^6 = 64$ $2^7 = 128$
- A standard 7-bit code is *ASCII(American Standard Code for Information Interchange)*




## Decimal ASCII Character Set

**Decimal - Character**

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




## Hexadecimal ASCII Character Set

**Hexadecimal - Character**

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | NUL | 01 | SOH | 02 | STX | 03 | ETX | 04 | EOT | 05 | ENQ | 06 | ACK | 07 | BEL |
| 08 | BS | 09 | HT | 0A | NL | 0B | VT | 0C | NP | 0D | CR | 0E | SO | 0F | SI |
| 10 | DLE | 11 | DC1 | 12 | DC2 | 13 | DC3 | 14 | DC4 | 15 | NAK | 16 | SYN | 17 | ETB |
| 18 | CAN | 19 | EM | 1A | SUB | 1B | ESC | 1C | FS | 1D | GS | 1E | RS | 1F | US |
| 20 | SP | 21 | ! | 22 | " | 23 | # | 24 | $ | 25 | % | 26 | & | 27 | ' |
| 28 | ( | 29 | ) | 2A | * | 2B | + | 2C | , | 2D | - | 2E | . | 2F | / |
| 30 | 0 | 31 | 1 | 32 | 2 | 33 | 3 | 34 | 4 | 35 | 5 | 36 | 6 | 37 | 7 |
| 38 | 8 | 39 | 9 | 3A | : | 3B | ; | 3C | < | 3D | = | 3E | > | 3F | ? |
| 40 | @ | 41 | A | 42 | B | 43 | C | 44 | D | 45 | E | 46 | F | 47 | G |
| 48 | H | 49 | I | 4A | J | 4B | K | 4C | L | 4D | M | 4E | N | 4F | O |
| 50 | P | 51 | Q | 52 | R | 53 | S | 54 | T | 55 | U | 56 | V | 57 | W |
| 58 | X | 59 | Y | 5A | Z | 5B | [ | 5C | \ | 5D | ] | 5E | ^ | 5F | _ |
| 60 | ` | 61 | a | 62 | b | 63 | c | 64 | d | 65 | e | 66 | f | 67 | g |
| 68 | h | 69 | i | 6A | j | 6B | k | 6C | l | 6D | m | 6E | n | 6F | o |
| 70 | p | 71 | q | 72 | r | 73 | s | 74 | t | 75 | u | 76 | v | 77 | w |
| 78 | x | 79 | y | 7A | z | 7B | { | 7C | \| | 7D | } | 7E | ~ | 7F | DEL |

## Extended ASCII: An 8-bit Code

- By the mid-1960's, it became clear that 7-bit ASCII was not enough to represent text from languages other than English
- IBM extended ASCII to 8 bits (256 symbols)
- Called "*Extended ASCII*," the first half is original ASCII with a 0 added at the beginning of each group of bits
- Handles most Western languages and additional useful symbols




| ASCII | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0000 | | | | 0 | @ | P | ` | p | | | | | | | | |
| 0001 | | | | 1 | A | Q | a | q | | | | | | | | |
| 0010 | | | ! | 2 | B | R | b | r | | | | | | | | |
| 0011 | | | " | 3 | C | S | c | s | | | | | | | | |
| 0100 | | | # | 4 | D | T | d | t | | | | | | | | |
| 0101 | | | $ | 5 | E | U | e | u | | | | | | | | |
| 0110 | | | % | 6 | F | V | f | v | | | | | | | | |
| 0111 | | | & | 7 | G | W | g | w | | | | | | | | |
| 1000 | | | ' | 8 | H | X | h | x | | | | | | | | |
| 1001 | | | ( | 9 | I | Y | i | y | | | | | | | | |
| 1010 | | | ) | : | J | Z | j | z | | | | | | | | |
| 1011 | | | * | ; | K | [ | k | { | | | | | | | | |
| 1100 | | | + | < | L | \ | l | \| | | | | | | | |
| 1101 | | | , | = | M | ] | m | } | | | | | | | | |
| 1110 | | | - | > | N | ^ | n | ~ | | | | | | | | |
| 1111 | | | . | ? | O | _ | o | | | | | | | | | |

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




## ASCII Coding of Phone Numbers

- How would a computer represent in its memory, the phone number 888 555 1212?
- Encode each digit with its ASCII byte




## Unicode

- Several languages around the world have more than 256 individual characters
- Unicode uses 16 bits; $2^{16}$ = 65536 characters
  - 1st 7 bits (128 chars) are ASCII chars
  - Different locales – different characters beyond 1st 7 bits




## NATO Broadcast Alphabet

- The code for broadcast communication is purposefully inefficient, to be distinctive when spoken amid noise

Table 8.4 NATO broadcast alphabet designed not to be minimal

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| A | Alpha | H | Hotel | O | Oscar | V | Victor |
| B | Bravo | I | India | P | Papa | W | Whiskey |
| C | Charlie | J | Juliet | Q | Quebec | X | X-ray |
| D | Delta | K | Kilo | R | Romeo | Y | Yankee |
| E | Echo | L | Lima | S | Sierra | Z | Zulu |
| F | Foxtrot | M | Mike | T | Tango | | |
| G | Golf | N | November | U | Uniform | | |




## The Oxford English Dictionary

- Extended ASCII encodes letters and characters well, but most documents contain more than just text.
  - Format information like font, font size, justification
- Formatting characters could be added to ASCII, but that mixes the content with the description of its form (*metadata*)
- Metadata is represented using tags, as in HTML

## Using Tags to Encode

- Oxford English Dictionary (OED) printed version is 20 volumes
- We could type the entire contents as ASCII characters (in about 120 years), but searching would be difficult
  - Suppose you search for the word "set." It is included in many other words like closet, horsetail, settle, etc.
  - How will the software know what characters comprise the definition of set?
    - Incorporate metadata




## Structure Tags

- Special set of tags was developed to specify OED's structure
  - <hw> means headword, the word being defined
  - Other tags label pronunciation <pr>, phonetic notation <ph>, parts of speech <ps>
- The tags do not print. They are there only to specify structure so the computer knows what part of the dictionary it is looking at




**byte** (bait). *Computers.* [Arbitrary, prob. influenced by bit sb.[4] and bite sb.] A group of eight consecutive bits operated on as a unit in a computer.

**1964 Blaauw** & **Brooks** in *IBM Systems Jrnl.* III. 122 An 8-bit unit of information is fundamental to most of the formats [of the System/360]. A consecutive group of *n* such units constitutes a field of length *n*. Fixed-length fields of length one, two, four, and eight are termed bytes, halfwords, words, and double words respectively. **1964** *IBM Jrnl. Res. & Developm.* VIII. 97/1 When a byte of data appears from an I/O device, the CPU is seized, dumped, used and restored. **1967** ***P. A. Stark*** *Digital Computer Programming* xix. 351 The normal operations in fixed point are done on four bytes at a time. **1968** *Datamation* 24 Jan. 1/1 Tape reading and writing is at from 34,160 to 192,000 bytes per second.

```
<e><hg><hw>byte</hw> <pr><ph>bait</ph></pr></hg>. <la> Computers</la>. <etym>
Arbitrary, prob. influenced by <xr><x>bit</x></xr> <ps>s.<hm>4</hm></ps>and
<xr><x>bite</x> <ps>n.</ps></xr></etym> <a4>A group of eight consecutive bits
operated on as a unit in a computer.</a4><qp><q><qd>1964</qd><a>Blaauw</a>
&amp. <a>Brooks</a><bib>in</bib> <w>IBM Systems Jrnl.</w> <lc>III.122</lc>
<qt>An 8-bit unit of information is fundamental to most of the formats <ed>of
the System/360</ed>.&es.A consecutive group of <i>n</i> such units constitutes
a field of length <i>n</i>.&es.Fixed-length fields of length one, two, four,
and eight are termed bytes, halfwords, words, and double words respectively.
</qt></q><q><qd>1964</qd> <w>IBM Jrnl. Res. &amp. Developm. </w> <lc>VIII.
97/1</lc> <qt>When a byte of data appears from an I/O device, the CPU is
seized, dumped, used and restored.</qt></q><q><qd>1967</qd> <a>P. A. Stark</a>
<w>Digital Computer Programming</w> <lc>xix. 351</lc> <qt>The normal operations
in fixed point are done on four bytes at a time.</qt></q> <q><qd> 1968</qd>
<w> Datamation</w> <lc>24 Jan. 1/1</lc> <qt>Tape reading and writing is at from
34,160 to 192,000 bytes per second.</qt></q></qp></e>
```

**Figure 8.4** The OED entry for the word byte, together with the representation of the entry in its digitized form with tags.

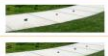



## Why "BYTE"

- Why is BYTE spelled with a Y?
- The 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 they changed the i to a y, to minimize typing errors.




**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 |