**Lecture 2. Data Representation and Policy for Searching Communications**

Read for Lecture 3:

* Chapter 5, “Tensions Between Cybersecurity and other Public Policy Concerns,” pp. 93-115, from *At The Nexus of Cybersecurity and Public Policy,* D. Clark, T. Berson, and H. Lin, Eds, National Research Council, 2014.
  + Available on Canvas in Files, Debate Materials.
  + *For each debate, the debaters are asked to consider the effect of the resolution from four perspectives: economics, innovation, civil liberties, and international relations. This chapter of this book addresses exactly these four areas and so should help you you're your debate preparation.*

**Exercise: Bits and their meaning: it all depends on how you look at it**

1. (10 points) We express years in decimal numbers, and so the transition from 1999 to 2000 was a big event. But suppose we counted years in binary:

a. What interesting property does 201510 have when expressed as a binary number?

b. When the New Year 2016 arrived, we got another interesting binary number. What interesting property does 201610 have when expressed in binary?

2. (10 points) Bits as logical values. In logical operations, a zero-bit means “false” and a one-bit means “true”. Logical operators include “NOT”, “AND”, “OR”, and “EXCLUSIVE OR” (XOR). A XOR B means “A is true or B is true, but not both”. If A and B are equal length strings of bits, these operations are applied to the corresponding bits in each string, so for example if A = 11, and B is 01, then NOT A is 00 and (A AND B) is 01.

With this in mind, for A = 0101 and B=1011, what is:

NOT A: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

NOT B:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

A OR B:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

A AND B:\_\_\_\_\_\_\_\_\_\_\_\_\_\_

A XOR B: \_\_\_\_\_\_\_\_\_\_\_\_\_

3. (15 points) Bits as numbers: machine arithmetic. Suppose our computer has 8-bit bytes, 16-bit words (i.e., 2 bytes), and uses 2’s complement notation for negative numbers (so negative numbers start with a ‘1’ bit in the leftmost bit position).

0000 1111 0000 01012

+

0000 1111 0000 01012

= \_2

4. (10 points) Same situation as problem 3.

Extra Credit (15 pts): convert each line to decimal

0111 1111 1111 11112 = \_[Hint: =215-1]\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 10

+ 0000 0000 0000 00102 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_10

= \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2  =\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_10

5. (15 points) Bits as characters: Table look-up problem. Suppose a word in another computer contains 4 bytes and has the following contents, expressed in hexadecimal:

50 48 49 4E

Suppose the computer word is interpreted as a sequence of characters from the UTF-8 (“Unicode”) character set (see Unicode table at the end of the exercise). Write in the space below the English characters they represent

\_\_ \_\_ \_\_ \_\_

6. (15 points) Bits as numbers: arithmetic problem: Now suppose the same word in the computer is interpreted as an integer value (i.e. a number). Convert the value to base 10 and write the result below.

For reference, in decimal (base 10) notation:

160 = 1

161 = 16

162 = 256

163 = 4096

164 = 65536

165 = 1,048,576

166 = 16,777,216

167 = 268,435,456

So the problem is, convert 5048494E16 to decimal notation. Enter your answer below. Show your work.

7 (15 points) Bits as machine instructions. Table look-up problem: Now suppose the same four bytes are interpreted as machine instructions in the Intel x86 instruction set, a portion of which is included below. As it happens, each of these codes corresponds to a single 1-byte instruction in the Intel x86 instruction set. What four instructions would these bytes represent? (give mnemonic and description for each).

5016 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4816 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4916 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4E16 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Find Intel x86 instruction set table following the UTF-8 character table.

8. (10 points) Bits as colors. Hexadecimal values can also be used to represent colors. One way this is done is for each byte to represent a value for each primary color Red, Green, and Blue. In this case it takes three bytes to represent a particular color. Each color strengths can range independently from 00 to FF in intensity, so a totally green color would be 00FF0016, and totally blue would be 0000FF. The resulting color is a mix of these values. You can see these effects at this website:

<http://www.colorschemer.com/online.html>

Pick three contiguous bytes of 50 48 49 4C16  and try them out in the web page color test area (left hand side of webpage).

Write down below: Hex value you used: \_\_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_\_\_16

Color you observed (your description): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

| **UTF-8 (hex.)** | **Printed**  **Char-acter** | **Character name** |
| --- | --- | --- |
| 00 |  |  |
| 01 |  |  |
| 02 |  |  |
| 03 |  | <control> |
| 04 |  | <control> |
| 05 |  | <control> |
| 06 |  | <control> |
| 07 |  | <control> |
| 08 |  | <control> |
| 09 |  | <control> |
| 0a |  | <control> |
| 0b |  | <control> |
| 0c |  | <control> |
| 0d |  | <control> |
| 0e |  | <control> |
| 0f |  | <control> |
| 10 |  | <control> |
| 11 |  | <control> |
| 12 |  | <control> |
| 13 |  | <control> |
| 14 |  | <control> |
| 15 |  | <control> |
| 16 |  | <control> |
| 17 |  | <control> |
| 18 |  | <control> |
| 19 |  | <control> |
| 1a |  | <control> |
| 1b |  | <control> |
| 1c |  | <control> |
| 1d |  | <control> |
| 1e |  | <control> |
| 1f |  | <control> |
| 20 |  | SPACE |
| 21 | ! | EXCLAMATION MARK |
| 22 | " | QUOTATION MARK |
| 23 | # | NUMBER SIGN |
| 24 | $ | DOLLAR SIGN |
| 25 | % | PERCENT SIGN |
| 26 | & | AMPERSAND |
| 27 | ' | APOSTROPHE |
| 28 | ( | LEFT PARENTHESIS |
| 29 | ) | RIGHT PARENTHESIS |
| 2a | \* | ASTERISK |
| 2b | + | PLUS SIGN |
| 2c | , | COMMA |
| 2d | - | HYPHEN-MINUS |
| 2e | . | FULL STOP |
| 2f | / | SOLIDUS |
| 30 | 0 | DIGIT ZERO |
| 31 | 1 | DIGIT ONE |
| 32 | 2 | DIGIT TWO |
| 33 | 3 | DIGIT THREE |
| 34 | 4 | DIGIT FOUR |
| 35 | 5 | DIGIT FIVE |
| 36 | 6 | DIGIT SIX |
| 37 | 7 | DIGIT SEVEN |
| 38 | 8 | DIGIT EIGHT |
| 39 | 9 | DIGIT NINE |
| 3a | : | COLON |
| 3b | ; | SEMICOLON |
| 3c | < | LESS-THAN SIGN |
| 3d | = | EQUALS SIGN |
| 3e | > | GREATER-THAN SIGN |
| 3f | ? | QUESTION MARK |
| 40 | @ | COMMERCIAL AT |
| 41 | A | LATIN CAPITAL LETTER A |
| 42 | B | LATIN CAPITAL LETTER B |
| 43 | C | LATIN CAPITAL LETTER C |
| 44 | D | LATIN CAPITAL LETTER D |
| 45 | E | LATIN CAPITAL LETTER E |
| 46 | F | LATIN CAPITAL LETTER F |
| 47 | G | LATIN CAPITAL LETTER G |
| 48 | H | LATIN CAPITAL LETTER H |
| 49 | I | LATIN CAPITAL LETTER I |
| 4a | J | LATIN CAPITAL LETTER J |
| 4b | K | LATIN CAPITAL LETTER K |
| 4c | L | LATIN CAPITAL LETTER L |
| 4d | M | LATIN CAPITAL LETTER M |
| 4e | N | LATIN CAPITAL LETTER N |
| 4f | O | LATIN CAPITAL LETTER O |
| 50 | P | LATIN CAPITAL LETTER P |
| 51 | Q | LATIN CAPITAL LETTER Q |
| 52 | R | LATIN CAPITAL LETTER R |
| 53 | S | LATIN CAPITAL LETTER S |
| 54 | T | LATIN CAPITAL LETTER T |
| 55 | U | LATIN CAPITAL LETTER U |
| 56 | V | LATIN CAPITAL LETTER V |
| 57 | W | LATIN CAPITAL LETTER W |
| 58 | X | LATIN CAPITAL LETTER X |
| 59 | Y | LATIN CAPITAL LETTER Y |
| 5a | Z | LATIN CAPITAL LETTER Z |
| 5b | [ | LEFT SQUARE BRACKET |
| 5c | \ | REVERSE SOLIDUS |
| 5d | ] | RIGHT SQUARE BRACKET |
| 5e | ^ | CIRCUMFLEX ACCENT |
| 5f | \_ | LOW LINE |
| 60 | ` | GRAVE ACCENT |
| 61 | a | LATIN SMALL LETTER A |
| 62 | b | LATIN SMALL LETTER B |
| 63 | c | LATIN SMALL LETTER C |
| 64 | d | LATIN SMALL LETTER D |
| 65 | e | LATIN SMALL LETTER E |
| 66 | f | LATIN SMALL LETTER F |
| 67 | g | LATIN SMALL LETTER G |
| 68 | h | LATIN SMALL LETTER H |
| 69 | i | LATIN SMALL LETTER I |
| 6a | j | LATIN SMALL LETTER J |
| 6b | k | LATIN SMALL LETTER K |
| 6c | l | LATIN SMALL LETTER L |
| 6d | m | LATIN SMALL LETTER M |
| 6e | n | LATIN SMALL LETTER N |
| 6f | o | LATIN SMALL LETTER O |
| 70 | p | LATIN SMALL LETTER P |
| 71 | q | LATIN SMALL LETTER Q |
| 72 | r | LATIN SMALL LETTER R |
| 73 | s | LATIN SMALL LETTER S |
| 74 | t | LATIN SMALL LETTER T |
| 75 | u | LATIN SMALL LETTER U |
| 76 | v | LATIN SMALL LETTER V |
| 77 | w | LATIN SMALL LETTER W |
| 78 | x | LATIN SMALL LETTER X |
| 79 | y | LATIN SMALL LETTER Y |
| 7a | z | LATIN SMALL LETTER Z |
| 7b | { | LEFT CURLY BRACKET |
| 7c | | | VERTICAL LINE |
| 7d | } | RIGHT CURLY BRACKET |
| 7e | ~ | TILDE |
| 7f |  | <control> |

Portion of Intel x86 Opcode table

|  |  |  |
| --- | --- | --- |
| Opcode (hex.) | Mnemonic | Meaning |
| 40 | INC | Increment (add 1) to register 0 |
| 41 | INC | Increment (add 1) to register 1 |
| 42 | INC | Increment (add 1) to register 2 |
| 43 | INC | Increment (add 1) to register 3 |
| 44 | INC | Increment (add 1) to register 4 |
| 45 | INC | Increment (add 1) to register 5 |
| 46 | INC | Increment (add 1) to register 6 |
| 47 | INC | Increment (add 1) to register 7 |
| 48 | DEC | Decrement (subtract 1) from register 0 |
| 49 | DEC | Decrement (subtract 1) from register 1 |
| 4a | DEC | Decrement (subtract 1) from register 2 |
| 4b | DEC | Decrement (subtract 1) from register 3 |
| 4c | DEC | Decrement (subtract 1) from register 4 |
| 4d | DEC | Decrement (subtract 1) from register 5 |
| 4e | DEC | Decrement (subtract 1) from register 6 |
| 4f | DEC | Decrement (subtract 1) from register 7 |
| 50 | PUSH | Push register 0 contents onto stack |
| 51 | PUSH | Push register 1 contents onto stack |
| 52 | PUSH | Push register 2 contents onto stack |
| 53 | PUSH | Push register 3 contents onto stack |
| 54 | PUSH | Push register 4 contents onto stack |
| 55 | PUSH | Push register 5 contents onto stack |
| 56 | PUSH | Push register 6 contents onto stack |
| 57 | PUSH | Push register 7 contents onto stack |