Chapter 2 Notes

Computer Science

2 Representing and Manipulating Information

2.1 Information Storage

Example (page 34):

2.1.1 Hexadecimal and Binary

Convert to binary: `0x173A4C`

0001 0111 0011 1010 0100 1100

How Hexadecimal works:
Bit representation of each digit:

1:0001 **7**:0111 **3**:0011 **A**:1010 **4**:0100 **C**:1100

`Each digit in Hexadecimal numbers are grouped into 8 byte parts.`
`When counting bytes you start from the end and group by 4.`
How binary works:

$$1_{128}1_{64}0_{32}0_{16} \quad 0_8 0_4 1_2 0_1 - \textit{every 8 digits is 1 byte}$$

`The next value is the current value x 2.  Add the subscripts to get the value.`

2.1.2 Words

Virtual addresses can range from $0 : 2_w - 1$. They have access to at most 2_w bytes.

2.1.3 Data Sizes

C declaration	32-bit	64-bit
char	1	1
short int	2	2
int	4	4
long int	4	8
long long int	8	8
char* int	4	8
float int	4	4
double	8	8

`Sizes (bytes) of different Data Types in C`

Computer memory can be read in two different ways. *big endian* and *little endian*.
Big endian will have to **most significant** byte in the **smallest** memory address.
Little endian will have to **least significant** byte in the **smallest** memory address.

```
Hex value 0x01234567 at address 0x100:
```

Big Endian

	0x100	0x101	0x102	0x103	
...	01	23	45	67	...

Little Endian

	0x100	0x101	0x102	0x103	
...	67	45	23	01	...

Which endian a computer uses usually doesn't matter except when they transfer information over a network. Code written for network applications must follow specific conventions for the program to work.

2.1.7 Introduction to Boolean Algebra

$$
\begin{array}{r} 0110 \\ +\ 1100 \\ \hline 0100 \end{array}
\qquad
\begin{array}{r} 0110 \\ |\ 1100 \\ \hline 0100 \end{array}
\qquad
\begin{array}{r} 0110 \\ \oplus\ 1100 \\ \hline 1010 \end{array}
\qquad
\begin{array}{r} \\ \sim\ 1100 \\ \hline 0011 \end{array}
$$