

Number_Systems

Module Code: GEEN1064

Module Name: Engineering Design and Implementation

Lecturer: Seb Blair BEng(H) PGCAP MIET MIHEEM FHEA

Outline

- Information Theory
- Binary
- Hexadecimal
- Octal

Information Theory

- [A Mathematical Theory of Communication, \(Shannon,1948\)](#)
 - Term *bit* was introduced as something that can have the representation of **1** or **0** most commonly an electrical signal
 - Foundations of the electrical computer was built off of this knowledge.
- Sequences of bits have no intrinsic meaning except for the representation that we assign to them, both by convention and by building particular operations into the hardware.

Binary

By choosing an appropriate representation, you can use bits to represent any value you can imagine:

- Characters are represented using numeric character codes.
- Floating-point representation supports real numbers.
- Two-dimensional arrays of bits represent images.
- Sequences of images represent video.
- and so on...

can be read right to left or left to right (an important distinction)

Binary

| Decimal | Binary | Decimal | Binary |
|---------|--------|---------|--------|
| 0 | 000000 | 10 | 001010 |
| 1 | 000001 | 11 | 001011 |
| 2 | 000010 | 12 | 001100 |
| 3 | 000011 | 13 | 001101 |
| 4 | 000100 | 14 | 001110 |
| 5 | 000101 | 15 | 001111 |
| 6 | 000110 | 16 | 001000 |
| 7 | 000111 | 17 | 010001 |
| 8 | 001000 | ... | ... |
| 9 | 001001 | 63 | 111111 |

Binary (2^n)

The number of combinations can be found by doing the following:

$$N_{combinations} = 2^n$$

You can calculate the maximum number you can have:

$$N_{maxValue} = 2^n - 1$$

| | MSB | | | | | | | | | | LSB |
|-----------|----------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| | 2^{10} | 2^9 | 2^8 | 2^7 | 2^6 | 2^5 | 2^4 | 2^3 | 2^2 | 2^1 | 2^0 |
| 2^n | 1024 | 512 | 256 | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| $2^n - 1$ | 1023 | 511 | 255 | 127 | 63 | 31 | 15 | 7 | 3 | 1 | 0 |

Binary to Decimal

| | | | | | | | | | | |
|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|
| 2^{10} | 2^9 | 2^8 | 2^7 | 2^6 | 2^5 | 2^4 | 2^3 | 2^2 | 2^1 | 2^0 |
| 1024 | 512 | 256 | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |

Therefore the decimal equivalent value is:

$$\begin{array}{r} 1 \cdot 2^7 = 128_{10} \\ 1 \cdot 2^4 = 16_{10} \\ 1 \cdot 2^2 = 4_{10} \\ 1 \cdot 2^1 = 2_{10} + \\ \hline [0]000\ 1001\ 0110_2 \equiv 150_{10} \end{array}$$

Your turn

| | | | | | | | | | | |
|----------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| 2^{10} | 2^9 | 2^8 | 2^7 | 2^6 | 2^5 | 2^4 | 2^3 | 2^2 | 2^1 | 2^0 |
| 1024 | 512 | 256 | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

- 1111

- ▶ Answer

- 11111101

- ▶ Answer

Little or Big Endian

- The order in which binary is read is very important.
- consider:
 - $10000000_2 = 256_{10} \parallel 1_{10}$
- or:
 - $10011001_2 = 153_{10} \parallel 153_{10}$
- Intel processors use little endian
- ISO standard for network address is big endian

Decimal to Binary

We can convert decimal to binary in a similar way eg: 51

- $64 < 51 = 0$
- $32 < 51 = 1$
- $(32 + 16) \rightarrow 48 \leq 51 = 1$
- $(48 + 8) \rightarrow 56 \leq 51 = 0$
- $(48 + 4) \rightarrow 52 \leq 51 = 0$
- $(48 + 2) \rightarrow 50 \leq 51 = 1$
- $(50 + 1) \rightarrow 51 \leq 51 = 1$

| 64 | 32 | 16 | 8 | 4 | 2 | 1 |
|----|----|----|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 1 | 1 |

Your Turn

| 2^{10} | 2^9 | 2^8 | 2^7 | 2^6 | 2^5 | 2^4 | 2^3 | 2^2 | 2^1 | 2^0 |
|----------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| 1024 | 512 | 256 | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |

- 75_{10}

- ▶ Answer

- 126_{10}

- ▶ Answer

- 487_{10}

- ▶ Answer

Hexadecimal (16^n)

Hexadecimal is base 16:

$$N_{combinations} = 16^n$$

You can calculate the maximum number you can have:

$$N_{maxValue} = 16^n - 1$$

| 16^6 | 16^5 | 16^4 | 16^3 | 16^2 | 16^1 | 16^0 |
|----------|---------|--------|--------|--------|--------|--------|
| 16777216 | 1048576 | 65536 | 4096 | 256 | 16 | 1 |

Hexadecimal, Binary, Decimal

| | | | | | | | | |
|---------------------|------|------|------|------|------|------|------|------|
| Binary | 1111 | 1110 | 1101 | 1100 | 1011 | 1010 | 1001 | 1000 |
| Unsigned Int | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Hexadecimal | F | E | D | C | B | A | 9 | 8 |

| | | | | | | | | |
|---------------------|------|------|------|------|------|------|------|------|
| Binary | 0111 | 0110 | 0101 | 0100 | 0011 | 0010 | 0001 | 0000 |
| Unsigned Int | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Hexadecimal | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Hexadecimal to Binary

Remember that binary and Hex have a relationship defined as 4 bits per Hexadecimal symbol.

| | | | | | | | |
|------|------|------|------|----------|----------|---|---|
| | | 1111 | 1111 | <i>F</i> | <i>F</i> | | |
| 1100 | 1010 | 0101 | 0111 | <i>C</i> | <i>A</i> | 5 | 7 |

Hexadecimal to Decimal

In hex each position's weight should be 16 times the previous.

| | | | | |
|--------|--------|--------|--------|---|
| 16^3 | 16^2 | 16^1 | 16^0 | $? = E5_{16}$ |
| | | | | $\rightarrow (E \cdot 16) + (5 \cdot 1)$ |
| | | | | $\rightarrow (14 \cdot 16) + (5 \cdot 1)$ |
| 4096 | 256 | 16 | 1 | $\rightarrow 224 + 5$ |
| | | | | $229_{10} =$ |

Hexadecimal to Decimal

The conversion can be done between 1 and 4 hexadecimal positions, much like Hexadecimal \longleftrightarrow Binary.

$$\begin{array}{cccc} 16^3 & 16^2 & 16^1 & 16^0 \\ 4096 & 256 & 16 & 1 \end{array}$$

$$? = B2F8_{16}$$

$$\rightarrow (B \cdot 4096) + (2 \cdot 256) + (F \cdot 16) + (8 \cdot 1)$$

$$\rightarrow (11 \cdot 4096) + (2 \cdot 256) + (15 \cdot 16) + (8 \cdot 1)$$

$$\rightarrow 45056 + 512 + 240 + 8$$

$$45816_{10} =$$

Octal (8^n)

Octal you'll see more commonly in Linux systems for file system permissions

| Binary | Octal | Permission |
|--------|-------|------------------|
| 111 | 7 | <code>rwX</code> |
| 110 | 6 | <code>rw-</code> |
| 101 | 5 | <code>r-X</code> |
| 100 | 4 | <code>r--</code> |
| 011 | 3 | <code>-wX</code> |
| 010 | 2 | <code>-w-</code> |
| 001 | 1 | <code>--X</code> |
| 000 | 0 | <code>---</code> |

```
rwXr-Xr-X seb users 4.0 KB Thu Jun 9 09:52:10 2022 .cache
rwXr-Xr-X seb users 4.0 KB Wed Jun 29 07:20:31 2022 .compose-cache
rwXr-Xr-X seb users 4.0 KB Wed Jun 29 08:01:09 2022 .config
rw----- seb users 16 B Mon Apr 25 15:22:27 2022 .esd_auth
rwXr-Xr-- seb users 143 B Wed Jun 29 09:36:24 2022 .fehbg
rw-r--r-- seb users 60 B Sun May 1 15:36:53 2022 .gitconfig
rwXr-Xr-X seb users 4.0 KB Sat May 21 11:57:51 2022 .java
rwXr-Xr-X seb users 4.0 KB Sat May 21 12:15:44 2022 .jssc
rw----- seb users 20 B Thu Jun 23 08:02:49 2022 .lessht
rwXr-Xr-X seb users 4.0 KB Mon Apr 25 11:47:39 2022 .local
rwx----- seb users 4.0 KB Mon May 23 18:43:32 2022 .mozilla
rwxrwxrwx seb users 4.0 KB Mon Mar 28 16:13:12 2022 .naivecalendar_events
rw-r--r-- seb users 45 B Wed Jun 8 10:41:10 2022 .nix-channels
rwXr-Xr-X seb users 4.0 KB Wed Jun 8 10:41:41 2022 .nix-defexpr
rwxrwxrwx seb users 42 B Mon Mar 28 12:58:15 2022 .nix-profile → /nix/var/nix/profile
rwXr-Xr-X seb users 4.0 KB Sat Apr 30 11:44:30 2022 .OTP
rwx----- seb users 4.0 KB Mon Mar 28 12:51:53 2022 .pki
rwx----- seb users 4.0 KB Mon May 9 15:02:53 2022 .ssh
rw----- seb users 14 KB Wed May 18 14:52:20 2022 .viminfo
rwXr-Xr-X seb users 4.0 KB Wed May 4 13:56:24 2022 .vscode
rwXr-Xr-X seb users 4.0 KB Wed May 4 13:57:07 2022 .vscode-oss
rw-r--r-- seb users 205 B Tue May 10 08:17:43 2022 .wget-hsts
rw----- seb users 152 B Wed Jun 29 07:20:31 2022 .Xauthority
rw----- seb users 310 B Wed May 18 15:40:06 2022 .xbindkeysrc
rw----- seb users 0 B Mon May 2 09:02:59 2022 .xsession-errors
rw----- seb users 0 B Sun May 1 16:30:08 2022 .xsession-errors.old
```

Octal \longleftrightarrow Binary

All you need to remember is the octal only uses three bits to represent its value per symbol.

Eg, 7301_8

$$001_2 = 1_8$$

$$000_2 = 0_8$$

$$011_2 = 3_8$$

$$111_2 = 7_8$$

$$111\ 011\ 000\ 001_2 \equiv 7301_8$$

Representing Characters

- Computers use numeric encodings to represent character data inside the memory of the machine, in which each character is assigned an integral value.
- Character codes, however, are not very useful unless they are standardised. When different computer manufacturers use different coding sequence (as was indeed the case in the early years), it is harder to share such data across machines.
- The first widely adopted character encoding was ASCII (American Standard Code for Information Interchange).
- With only 256 possible characters, the ASCII system proved inadequate to represent the many alphabets in use throughout the world. It has therefore been superseded by Unicode, which allows for a much larger number of characters.

American Standard Code for Information Interchange (ASCII)

| Dec | Char | Dec | Char | Dec | Char | Dec | Char | Dec | Char | Dec | Char | Dec |
|-----|-------|-----|------|-----|------|-----|------|-----|------|-----|------|-----|
| 32 | SPACE | 45 | - | 58 | : | 71 | G | 84 | T | 97 | a | 110 |
| 33 | ! | 46 | . | 59 | ; | 72 | H | 85 | U | 98 | b | 111 |
| 34 | " | 47 | / | 60 | < | 73 | I | 86 | V | 99 | c | 112 |
| 35 | # | 48 | 0 | 61 | = | 74 | J | 87 | W | 100 | d | 113 |
| 36 | \$ | 49 | 1 | 62 | > | 75 | K | 88 | X | 101 | e | 114 |
| 37 | % | 50 | 2 | 63 | ? | 76 | L | 89 | Y | 102 | f | 115 |
| 38 | & | 51 | 3 | 64 | @ | 77 | M | 90 | Z | 103 | g | 116 |
| 39 | ' | 52 | 4 | 65 | A | 78 | N | 91 | [| 104 | h | 117 |
| 40 | (| 53 | 5 | 66 | B | 79 | O | 92 | \ | 105 | i | 118 |