# COMP1917: Computing 1

# 6. Binary and Hexadecimal

Reading: Moffat, Section 13.2

# Outline

- Number Systems
- Binary Computation
- Converting between Binary and Decimal
- Octal and Hexadecimal
- Fractional Component

# Number Systems

- Most cultures have developed number systems based on 5 or 10, but there are some exceptions (often based on counting the spaces between the fingers as well as the fingers):
  - Base 8 (Yuki of northern California)
  - Base 9 (Nenet language of Russia)
  - Base 12 (hours of the day)
  - Base 20 (Maya of Central America)
  - Base 60 (Sumerians)
- For digital computers, it is convenient to use Base 2 (Binary), Base 8 (Octal) or Base 16 (Hexadecimal).

# Decimal Representation

- Can interpret decimal number 4705 as:

$$4 \times 10^3 + 7 \times 10^2 + 0 \times 10^1 + 5 \times 10^0$$

- The base or radix is 10
  Digits 0 – 9

- Place values:

$$\begin{array}{ccccc} \cdots & 1000 & 100 & 10 & 1 \\ \cdots & 10^3 & 10^2 & 10^1 & 10^0 \end{array}$$

- Write number as $4705_{10}$
  - Note use of subscript to denote base

# Binary Computation

- In a similar way, can interpret binary number `1011` as:

$$1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

- The base or radix is 2

  Digits `0` and `1`

- Place values:

$$\begin{array}{ccccc} \cdots & 8 & 4 & 2 & 1 \\ \cdots & 2^3 & 2^2 & 2^1 & 2^0 \end{array}$$

- Write number as $\mathtt{1011}_2$

  $(= \mathtt{11}_{10})$

# Binary Computation

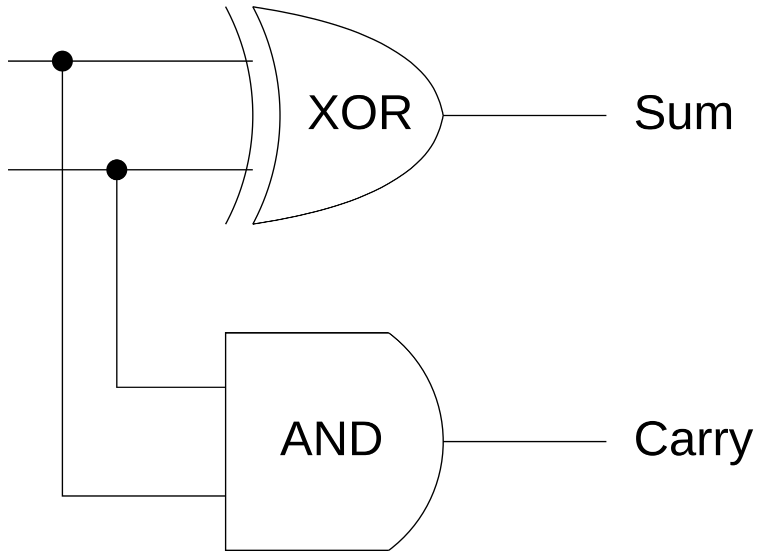
- Binary representation is convenient because
  - easily represented using two voltage levels: low and high
  - simple arithmetic tables that can be implemented using logic gates

| + | **0** | **1** |
|---|---|---|
| **0** | 00 | 01 |
| **1** | 01 | 10 |

- | $\times$ | **0** | **1** |
  |---|---|---|
  | **0** | 0 | 0 |
  | **1** | 0 | 1 |

# Binary Representation

- Addition of 1-bit binary numbers can be achieved by half-adder circuit.




- General addition requires combination of half-adder circuits

# Converting from base $x$ to Decimal

- Consider 4-digit number in arbitrary base $x$ : $\texttt{abcd}_x$
- Rewrite in polynomial form:

$$ax^3 + bx^2 + cx + d$$

- Or nested form:

$$((ax+b)x+c)x+d$$

- Perform necessary computations to achieve decimal conversion

# Examples

- Convert $1101_2$ to base 10 (Decimal):
  - $1101_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 13_{10}$
  - $1101_2 = ((1 \times 2 + 1) \times 2 + 0) \times 2 + 1 = 13_{10}$

- Convert $11001011_2$ to base 10:
  - $11001011_2$
    $= 1.2^7 + 1.2^6 + 0.2^5 + 0.2^4 + 1.2^3 + 0.2^2 + 1.2^1 + 1.2^0 = 203_{10}$
  - $11001011_2$
    $= ((((((1.2 + 1).2 + 0).2 + 0).2 + 1).2 + 0).2 + 1).2 + 1 = 203_{10}$

# Converting to base $x$

- a base $x$ number in the form $((\mathtt{a}x+\mathtt{b})x+\mathtt{c})x+\mathtt{d}$ can be written:

$$\begin{aligned} \mathtt{abcd}_x &= Cx+\mathtt{d} \\ C &= Bx+\mathtt{c} \\ B &= \mathtt{a}x+\mathtt{b} \\ \mathtt{a} &= 0x+\mathtt{a} \end{aligned}$$

- Numbers `a,b,c,d` are all less than base $x$
- Equations above imply that `d,c,b,a` are remainders when the number $\mathtt{abcd}_x$ is repeatedly divided by $x$
- To convert an integer to base $x$
  - repeatedly divide quotient by $x$ until quotient is 0
  - write the remainders in reverse

# Examples

$$\begin{array}{rclccc}
25_{10} \div 2 & = & 12 & r & 1 & \uparrow \\
12_{10} \div 2 & = & 6 & r & 0 & \uparrow \\
6_{10} \div 2 & = & 3 & r & 0 & \uparrow \\
3_{10} \div 2 & = & 1 & r & 1 & \uparrow \\
1_{10} \div 2 & = & 0 & r & 1 & \uparrow
\end{array}$$

- Write remainders from left to right reading from bottom to top.
- $$25_{10} \quad = \quad \overrightarrow{11001_2}$$
- **ALWAYS check your answer:**
  $11001_2 = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 25_{10}$

# Examples

- Convert $11972_{10}$ to base 8 (Octal)

$$\begin{array}{rcrcl}
11972_{10} \div 8 & = & 1496 & r & 4 \\
1496 \div 8 & = & 187 & r & 0 \\
187 \div 8 & = & 23 & r & 3 \\
23 \div 8 & = & 2 & r & 7 \\
2 \div 8 & = & 0 & r & 2
\end{array}$$

- $11972_{10} = 27304_8$
- $27304_8 = 2 \times 8^4 + 7 \times 8^3 + 3 \times 8^2 + 0 \times 8^1 + 4 \times 8^0 = 11972_{10}$

# Decimal to Binary — With Division

- Repeated division of quotient by 2 as in previous example
- repeat until quotient is 0
- output remainders in the reverse of the order in which they were generated

# Decimal to Binary — With Division

$$
\begin{array}{rcrcl}
573_{10} \div 2 & = & 286 & r & 1 \\
286 \div 2 & = & 143 & r & 0 \\
143 \div 2 & = & 71 & r & 1 \\
71 \div 2 & = & 35 & r & 1 \\
35 \div 2 & = & 17 & r & 1 \\
17 \div 2 & = & 8 & r & 1 \\
8 \div 2 & = & 4 & r & 0 \\
4 \div 2 & = & 2 & r & 0 \\
2 \div 2 & = & 1 & r & 0 \\
1 \div 2 & = & 0 & r & 1
\end{array}
$$

- $573_{10} = 1000111101_2$
- Check:
  $1.2^9 + 0.2^8 + 0.2^7 + 0.2^6 + 1.2^5 + 1.2^4 + 1.2^3 + 1.2^2 + 0.2^1 + 1.2^0 = 573_{10}$

# Decimal to Binary — Without Division

- find largest power of 2 smaller than current number;
  subtract this from the number and repeat on the difference
- usually not practical for larger numbers
- powers of 2 are: 128, 64, 32, 16, 8, 4, 2, 1

# Decimal to Binary — Without Division

- Convert $174_{10}$ to binary:

$$
\begin{array}{lcll}
174 - 128 = 46 & \longrightarrow & 1 & \text{in } 2^7 \text{ place} \\
64 > 46 & \longrightarrow & 0 & \text{in } 2^6 \text{ place} \\
46 - 32 = 14 & \longrightarrow & 1 & \text{in } 2^5 \text{ place} \\
16 > 14 & \longrightarrow & 0 & \text{in } 2^4 \text{ place} \\
14 - 8 = 6 & \longrightarrow & 1 & \text{in } 2^3 \text{ place} \\
6 - 4 = 2 & \longrightarrow & 1 & \text{in } 2^2 \text{ place} \\
2 - 2 = 0 & \longrightarrow & 1 & \text{in } 2^1 \text{ place} \\
1 > 0 & \longrightarrow & 0 & \text{in } 2^0 \text{ place} \\
= 10101110_2 & & &
\end{array}
$$

# Exercises

- Convert the following to binary:
  - $5_{10}$
  - $10_{10}$
  - $12_{10}$
  - $23_{10}$

# Octal Representation

- Can interpret octal number $\mathtt{7053}_8$ as:

$$7 \times 8^3 + 0 \times 8^2 + 5 \times 8^1 + 3 \times 8^0$$

- The base or radix is 8

  Digits `0,1,2,3,4,5,6,7`

- Place values:

$$\begin{array}{ccccc} \cdots & 512 & 64 & 8 & 1 \\ \cdots & 8^3 & 8^2 & 8^1 & 8^0 \end{array}$$

- Write number as $\mathtt{7053}_8$

  $(= \mathtt{3627}_{10})$

# Binary to Octal

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |

- Idea: Collect bits into groups of three starting from right to left
- "pad" out left-hand side with 0's if necessary
- Convert each group of three bits into its equivalent octal representation (given in table above)
- Example: Convert $1011111000101001_2$ to Octal:

| **00**1 | 011 | 111 | 000 | 101 | $001_2$ |
|---|---|---|---|---|---|
| 1 | 3 | 7 | 0 | 5 | $1_8$ |

# Octal to Binary

- Reverse the previous process
- Convert each octal digit into equivalent 3-bit binary representation
- Example: Convert $27015_8$ to Binary:

| | | | | |
|---|---|---|---|---|
| 2 | 7 | 0 | 1 | $5_8$ |
| 010 | 111 | 000 | 001 | $101_2$ |

# Hexadecimal Representation

- Can interpret hexadecimal number `3AF1` as:

$$3 \times 16^3 + 10 \times 16^2 + 15 \times 16^1 + 1 \times 16^0$$

- The base or radix is 16

  Digits `0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F`

- Place values:

| | | | | |
|---|---|---|---|---|
| $\cdots$ | 4096 | 256 | 16 | 1 |
| $\cdots$ | $16^3$ | $16^2$ | $16^1$ | $16^0$ |

- Write number as $\mathtt{3AF1}_{16}$

  $(= \mathtt{15089}_{10})$

# Binary to Hexadecimal

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 |
| 8 | 9 | A | B | C | D | E | F |
| 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |

- Idea: Collect bits into groups of four starting from right to left
- "pad" out left-hand side with `0`'s if necessary
- Convert each group of four bits into its equivalent hexadecimal representation (given in table above)

# Binary to Hexadecimal

- Example: Convert $\texttt{1011111000101001}_2$ to Hex:

| | | | |
|---|---|---|---|
| 1011 | 1110 | 0010 | $1001_2$ |
| B | E | 2 | $9_{16}$ |

- Example: Convert $\texttt{10111101011100}_2$ to Hex:

| | | | |
|---|---|---|---|
| **00**10 | 1111 | 0101 | 1100 |
| 2 | F | 5 | $C_{16}$ |

# Hexadecimal to Binary

- Reverse the previous process
- Convert each hex digit into equivalent 4-bit binary representation
- Example: Convert $\texttt{AD5}_{16}$ to Binary:

| | | |
|---|---|---|
| A | D | 5 |
| 1010 | 1101 | $\texttt{0101}_2$ |

# Exercises

- Convert to binary
  - $\texttt{53}_{10}$
  - $\texttt{5F3A}_{16}$
  - $\texttt{12D}_{16}$
  - $\texttt{3701}_{8}$
  - $\texttt{4232}_{8}$
- Convert to octal
  - $\texttt{10101111011}_{2}$
  - $\texttt{5F3A}_{16}$
- Convert to hexadecimal
  - $\texttt{10101111011}_{2}$
  - $\texttt{3701}_{8}$

# Fractions

- In the same way that we use a decimal point (.) to represent fractional quantities for decimal numbers, we use the radix point to represent fractional quantities in any base
- For example, $\mathtt{abcd.pqrs}_x$ (note: base $x$) represents

$$ax^3 + bx^2 + cx^1 + dx^0 + px^{-1} + qx^{-2} + rx^{-3} + sx^{-4}$$

- Numbers to the left of radix point represent integer component
- Numbers to the right of radix point represent fractional component
- Polynomial above can be evaluated to determine equivalent decimal representation.

# Converting Fractions to Base $x$

- Convert integer component as usual
- To convert fractional component use "separate and multiply" technique
- Consider fractional component of $\texttt{abcd.pqrs}_x$
  - $px^{-1} + qx^{-2} + rx^{-3} + sx^{-4}$
  - Multiplying by $x$ gives: $p + qx^{-1} + rx^{-2} + sx^{-3}$
  - Multiplying remaining fractional component by $x$ gives: $q + rx^{-1} + sx^{-2}$
- Repeat until fractional part exhausted or you have sufficient digits (Note: process is not guaranteed to terminate)

# Example

- Convert $23.3125_{10}$ to base 8
- Integer component

$$
\begin{array}{rcccc}
23_{10} \div 8 & = & 2 & \text{r} & 7 \\
2 \div 8 & = & 0 & \text{r} & 2
\end{array}
$$

- Fractional component

$$
\begin{array}{lcc}
.3125_{10} \times 8 & = & 2.5 \\
.5_{10} \times 8 & = & 4.0
\end{array}
$$

- Therefore: $23.3125_{10} = 27.24_8$

# Example

- Convert $23.3125_{10}$ to base 2
- Integer component

$$\begin{array}{rcrcr} 23_{10} \div 2 & = & 11 & \text{r} & 1 \\ 11 \div 2 & = & 5 & \text{r} & 1 \\ 5 \div 2 & = & 2 & \text{r} & 1 \\ 2 \div 2 & = & 1 & \text{r} & 0 \\ 1 \div 2 & = & 0 & \text{r} & 1 \end{array}$$

# Example *continued*

- Fractional component

$$
\begin{aligned}
.3125_{10} \times 2 &= 0.625 \\
.625 \times 2 &= 1.25 \\
.25 \times 2 &= 0.5 \\
.5 \times 2 &= 1.0
\end{aligned}
$$

- Therefore, $23.3125_{10} = 10111.0101_2$