Computer Organization
Secs 1.1 - 1.2
Lecture 2

Robb T. Koether

Hampden-Sydney College

Wed, Aug 28, 2019

1. The Main Components

2. Input and Output

3. Storing Numbers
   - Binary Numbers
   - Hexadecimal Numbers
   - Converting Between Binary and Hexadecimal

Outline

1. The Main Components
2. Input and Output
3. Storing Numbers
   - Binary Numbers
   - Hexadecimal Numbers
   - Converting Between Binary and Hexadecimal

The Main Components

- The Central Processing Unit (CPU)
- Memory (RAM)
- Secondary storage devices (e.g., hard drive, flash drive)
- Input devices (e.g., keyboard)
- Output devices (e.g., monitor)

The CPU

- The central processing unit (CPU).
  - The register file.
  - The arithmetic and logic unit (ALU).
  - The branch unit.
  - The memory read/write unit.
  - The control unit.

The CPU

- The CPU:
  - Performs arithmetic.
    - Addition, subtraction, logical operations, etc.
  - Makes logical decisions.
    - Branch if two values are equal, etc.
  - Loads data from memory to registers.
  - Stores data from registers to memory.

Memory

- Levels of memory.
- Registers (in the CPU).
- Level 1 Cache (L1)
  - Holds commonly used data.
  - “10% of the data is used 90% of the time.”
- Level 2 Cache (L2)
  - Holds less commonly used data.
- Main Memory
  - Holds data that are “seldom” used.

Outline

1. The Main Components
2. Input and Output
3. Storing Numbers
   - Binary Numbers
   - Hexadecimal Numbers
   - Converting Between Binary and Hexadecimal

Input and Output

- The keyboard is standard input, referred to as `cin` in C++ programs.
- The monitor is standard output, referred to as `cout` in C++ programs.
- There is also standard error, referred to as `cerr` in C++ programs.

Input and Output

- The keyboard is standard input, referred to as `cin` in C++ programs.
- The monitor is standard output, referred to as `cout` in C++ programs.
- There is also standard error, referred to as `cerr` in C++ programs.
- In C, they were called `stdin`, `stdout`, and `stderr`.

Outline

1. The Main Components

2. Input and Output

3. Storing Numbers
   - Binary Numbers
   - Hexadecimal Numbers
   - Converting Between Binary and Hexadecimal

Representing Data

- How can a machine “store” a number?
- We design a machine that can be put into different “states.”
- Each state represents a value.
- For example, the values 0 through 9 may be represented by 10 different states.

Representation of Data

- How could we represent the values 0 through 99?

Representation of Data

- How could we represent the values 0 through 99?
- Build a machine with 100 states.

Representation of Data

- How could we represent the values 0 through 99?
- Build a machine with 100 states.
- Or, build *two* machines with 10 states each.
  - One machine represents the 1's digit.
  - The other machine represents the 10's digit.
  - With six 10-state machines, we could represent values from 0 to 999999.

Binary Representation of Data

- In computers, it is more efficient to use only two states.
  - A light: on/off
  - A switch: open/closed
  - Voltage: high/low
- Therefore, numbers will be stored in binary (base 2) rather than decimal.

Outline

1. The Main Components
2. Input and Output
3. Storing Numbers
   - Binary Numbers
   - Hexadecimal Numbers
   - Converting Between Binary and Hexadecimal

Binary Representation of Data

- With 1 bit, we can represent 2 different values, namely 0 and 1.

Binary Representation of Data

- With 1 bit, we can represent 2 different values, namely 0 and 1.
- With 2 bits, we can represent 4 different values, namely 0 through 3.

Pattern	Value
00	0
01	1
10	2
11	3

Binary Representation of Data

- With 3 bits, we can represent 8 values, namely 0 through 7.

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

- With 4 bits, we can represent 16 values, namely 0 through 15.

Pattern	Value
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

Pattern	Value
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

- With 4 bits, we can represent 16 values, namely 0 through 15.

Pattern	Value
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

Pattern	Value
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

But what about negative numbers?

Negative Numbers

- We "reinterpret" 8 through 15 as -8 through -1.

Pattern	Value
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

Pattern	Value
1000	-8
1001	-7
1010	-6
1011	-5
1100	-4
1101	-3
1110	-2
1111	-1

This is called two's complement notation.

Fixed-Length Binary Numbers

- In general, n bits can represent any of 2^n different values.
- We can represent positive (unsigned) numbers from 0 to $2^n - 1$.
  - 4 bits represent values from 0 to 15.
  - 8 bits represent values from 0 to 255.
  - 16 bits represent values from 0 to 65535.
  - 32 bits represent values from 0 to 4294967296.
- Or we can represent signed numbers from -2^{n-1} to $2^{n-1} - 1$.
  - 8 bits represent values -128 to $+127$.
  - 16 bits represent values -32768 to $+32767$.
  - 32 bits represent values -2147483648 to $+2147483647$.
  - 64 bits represent values -9223372036854775808 to $+9223372036854775807$. (± 9 quintillion)

Binary Number System

- In the binary number system, each position in the number represents a power of 2.
- Example: 101011 represents

$$(1 \times 2^5) + (0 \times 2^4) + (1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0)$$

which equals 32 + 8 + 2 + 1 = 43.

The Structure of Memory

- 1 bit = 1 binary digit (0 or 1).
- 1 half-byte (nibble) = 4 bits.
- 1 byte = 2 nibbles = 8 bits.
- 1 half-word = 2 bytes = 16 bits.
- 1 word = 4 bytes = 32 bits (standard on 32-bit processor).
- 1 long word = 8 bytes = 64 bits (standard on 64-bit processor).
- 1 kilobyte (Kb) = 1024 bytes = 2^{10} bytes.
- 1 megabyte (Mb) = 1024 Kb = 2^{20} bytes.
- 1 gigabyte (Gb) = 1024 Mb = 2^{30} bytes.
- 1 terabyte (Tb) = 1024 Gb = 2^{40} bytes.

Outline

1. The Main Components

2. Input and Output

3. Storing Numbers
   - Binary Numbers
   - Hexadecimal Numbers
   - Converting Between Binary and Hexadecimal

Hexadecimal Numbers

- For convenience, we often display values in hexadecimal (base 16).
- The hexadecimal system has 16 digits with values 0 through 15.

Hex	Binary	Value
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7

Hex	Binary	Value
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

Examples of Hexadecimal Numbers

$$
\begin{aligned}
4A_{16} &= (4 \times 16) + 10 = 74, \\
FF_{16} &= (15 \times 16) + 15 = 255, \\
123_{16} &= (1 \times 16^2) + (2 \times 16) + 3 = 292, \\
FACE_{16} &= (15 \times 16^3) + (10 \times 16^2) + (12 \times 16) + 14 \\
&= 61440 + 2560 + 192 + 14 \\
&= 64206.
\end{aligned}
$$

Hexadecimal Numbers

- Hexadecimal numbers are convenient because
    - They are compact.
    - There is a very simple relation between them and binary numbers, namely, 0ne hexadecimal digit represents a 4-digit binary number (values 0 - 15), or half a byte.
    - They are much easier to read and transcribe than binary numbers. That is, consider copying 11111111111111100000000000000000 vs. copying FFFE0000.

- Example: `HexNumbers.cpp`

Outline

1. The Main Components
2. Input and Output
3. Storing Numbers
   - Binary Numbers
   - Hexadecimal Numbers
   - Converting Between Binary and Hexadecimal

Conversion from Hexadecimal to Binary

- Convert 2B6F to binary
- Write each hex digit as a 4-bit binary number.

Binary	Hex
0000	0
0001	1
0010	2
0011	3

Binary	Hex
0100	4
0101	5
0110	6
0111	7

Binary	Hex
1000	8
1001	9
1010	A
1011	B

Binary	Hex
1100	C
1101	D
1110	E
1111	F

  - 2 = 0010
  - B = 1011
  - 6 = 0110
  - F = 1111
  - Write the 4-bit numbers together as a single binary number

0010101101101111

Conversion from Binary to Hexadecimal

- Convert 1001101011101011 to hexadecimal.
- Break it up into groups of 4 bits (starting on the right).

1001 1010 1110 1011

- Write each block of 4 bits as a hex digit.

Binary	Hex
0000	0
0001	1
0010	2
0011	3

Binary	Hex
0100	4
0101	5
0110	6
0111	7

Binary	Hex
1000	8
1001	9
1010	A
1011	B

Binary	Hex
1100	C
1101	D
1110	E
1111	F

9 A E B

- Write the hex digits as a single hex number.

9AEB