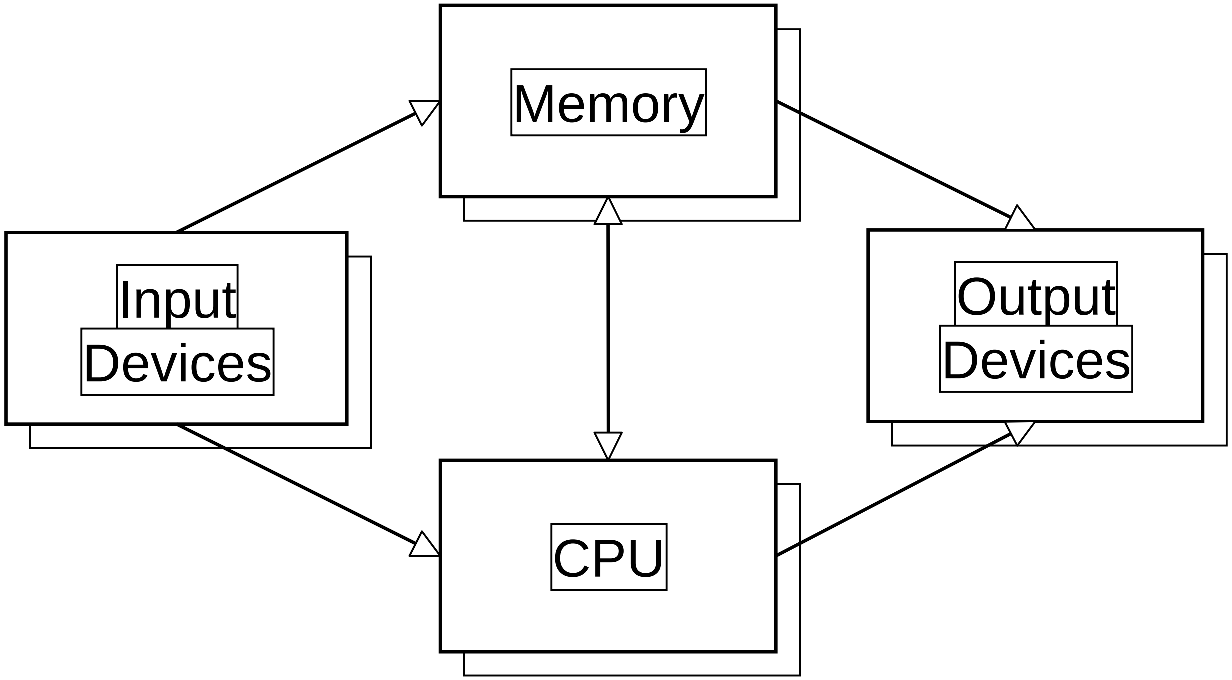
Chapter 1: Computing & the Object-Oriented Design Methodology

- Machine
- Software
- Program Design

Computer Organization

- CPU - central processing unit
  - Where decisions are made, computations are performed, and input/output requests are delegated
- Memory
  - Stores information being processed by the CPU
- Input devices
  - Allows people to supply information to computers
- Output devices
  - Allows people to receive information from computers

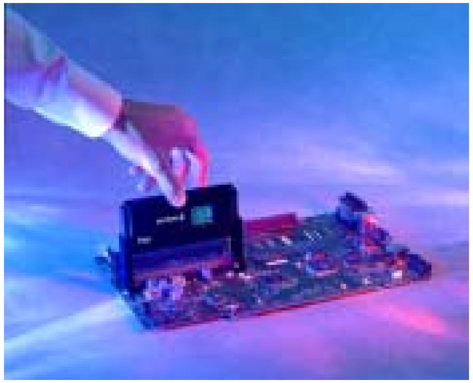
Computer Organization



CPU

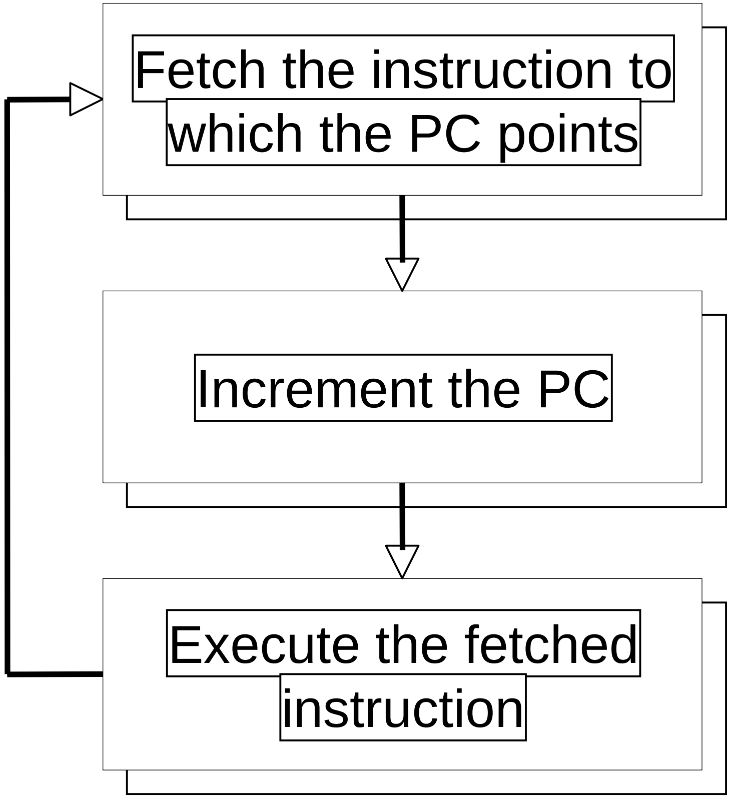
- *Brains* of the computer
  - Arithmetic calculations are performed using the Arithmetic/Logical Unit or ALU
  - Control unit decodes and executes instructions

- Arithmetic operations are performed using binary number system



Control Unit

- The fetch/execute cycle is the steps the CPU takes to execute an instruction
- Performing the action specified by an instruction is known as *executing the instruction*
- The program counter (PC) holds the memory address of the next instruction



Fetch the instruction to which the PC points

Increment the PC

Execute the fetched instruction

Input and Output Devices

- Accessories that allow computer to perform specific tasks
  - Receive information for processing
  - Return the results of processing
  - Store information

- Common input and output devices

Speakers	Mouse	Scanner
Printer	Joystick	CD-ROM
Keyboard	Microphone	DVD

- Some devices are capable of both input and output

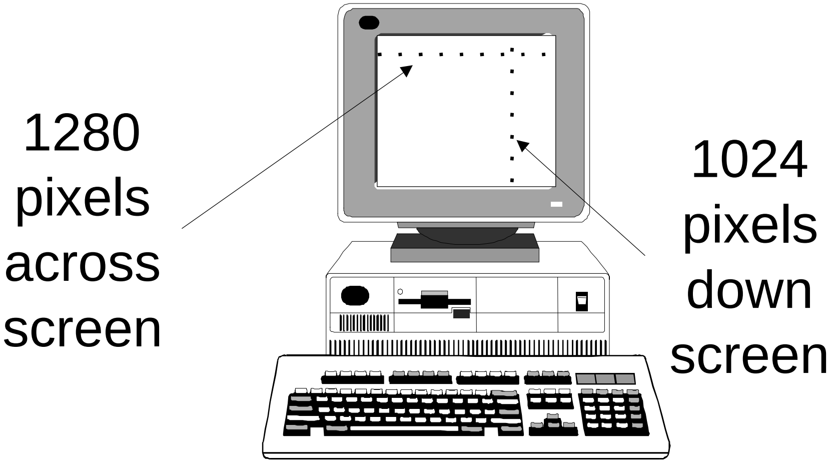
Floppy drive	Hard drive	Magnetic tape units

Monitor

- Display device that operates like a television
  - Also known as CRT (cathode ray tube)

- Controlled by an output device called a *graphics card*

- Displayable area
  - Measured in dots per inch, dots are often referred to as pixels (short for picture element)
  - Standard resolution is 640 by 480
  - Many cards support resolution of 1280 by 1024 or better
  - Number of colors supported varies from 16 to billions



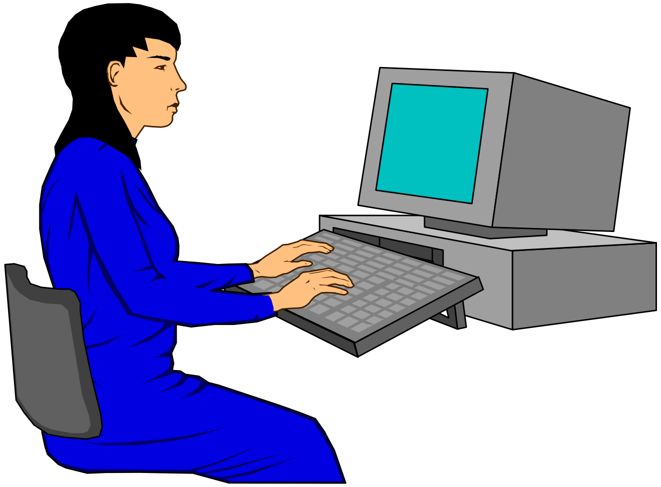
Software

- Application software
  - Programs designed to perform specific tasks that are transparent to the user
- System software
  - Programs that support the execution and development of other programs
  - Two major types
    - Operating systems
    - Translation systems

Application Software

- Application software is the software that has made using computers indispensable and popular

- Common application software
  - Word processors
  - Desktop publishing programs
  - Spreadsheets
  - Presentation managers
  - Drawing programs



- *Learning how to develop application software is our focus*

Operating System

- Examples
  - Windows®, UNIX®, Mac OS X®
- Controls and manages the computing resources
- Important services that an operating system provides
  - File system
    - Directories, folders, files
  - Commands that allow for manipulation of the file system
    - Sort, delete, copy
  - Ability to perform input and output on a variety of devices
  - Management of the running systems

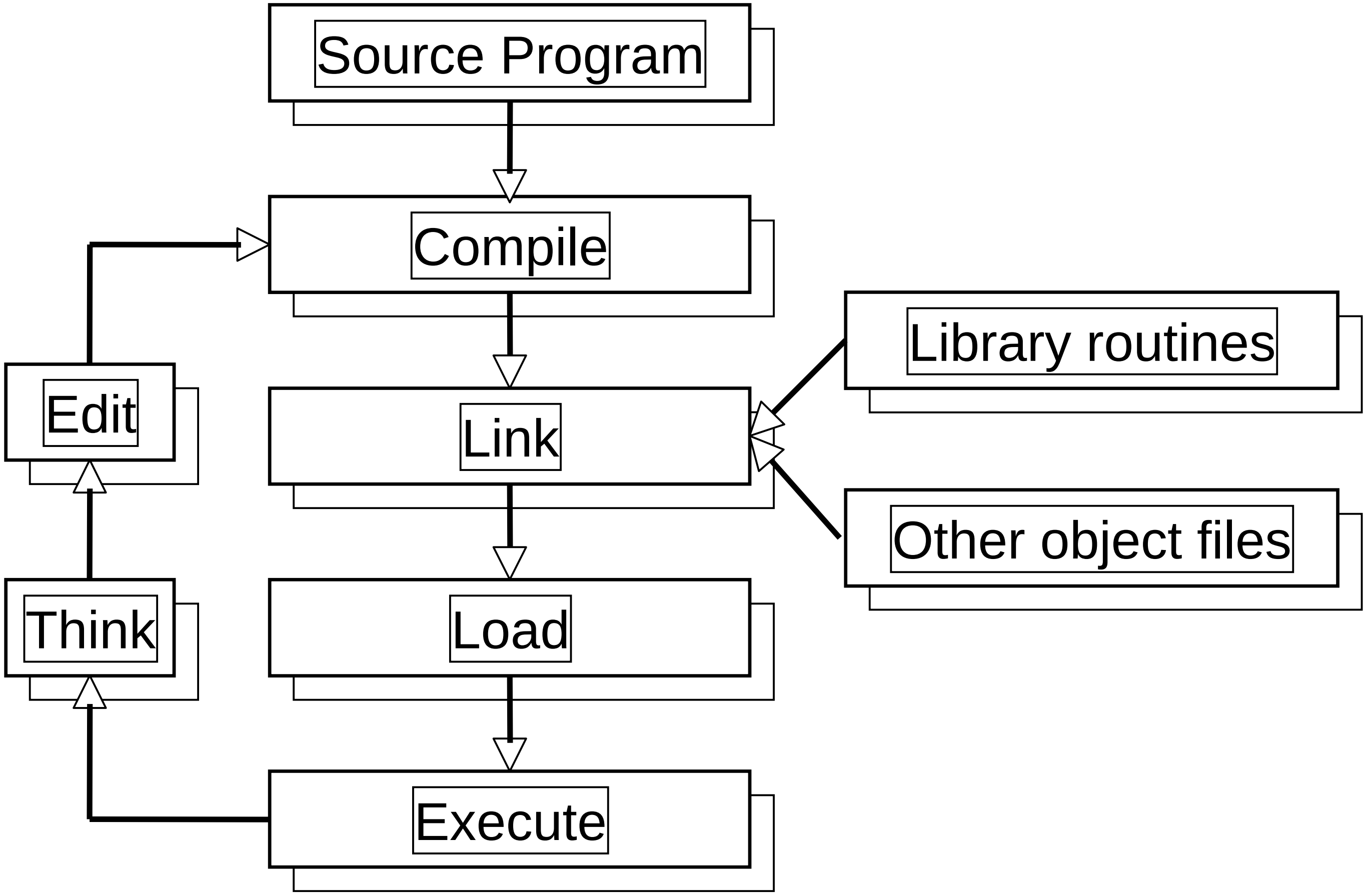
Translation System

- Set of programs used to develop software
- A key component of a translation system is a translator
- Some types of translators
  - Compiler
    - Converts from one language to another
  - Linker
    - Combines resources
- Examples
  - Microsoft Visual C++®, CBuilder®, g++, Code Warrior®
    - Performs compilation, linking, and other activities.

Software Development Activities

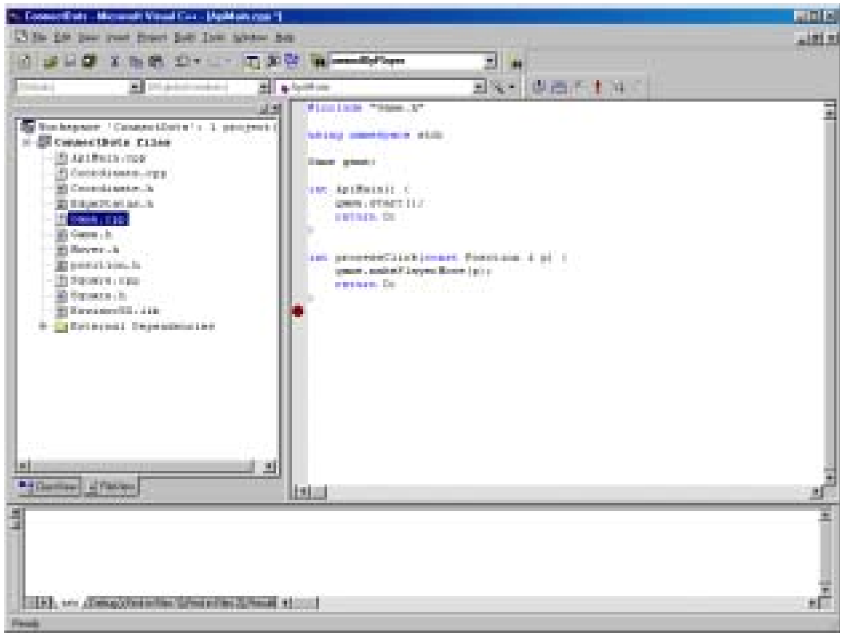
- Editing
- Compiling
- Linking with precompiled files
    - Object files
    - Library modules
- Loading and executing
- Viewing the behavior of the program

Software Development Cycle



IDEs

- Integrated Development Environments or IDEs
  - Supports the entire software development cycle
    - E.g., MS Visual C++, Borland, Code Warrior
- Provides all the capabilities for developing software
  - Editor
  - Compiler
  - Linker
  - Loader
  - Debugger
  - Viewer



```
#include "Game.h"

using namespace std;

Game game;

int ApiMain() {
    game.start();
    return 0;
}

int processClick(const Position & p) {
    game.makePlayerMove(p);
    return 0;
}
```

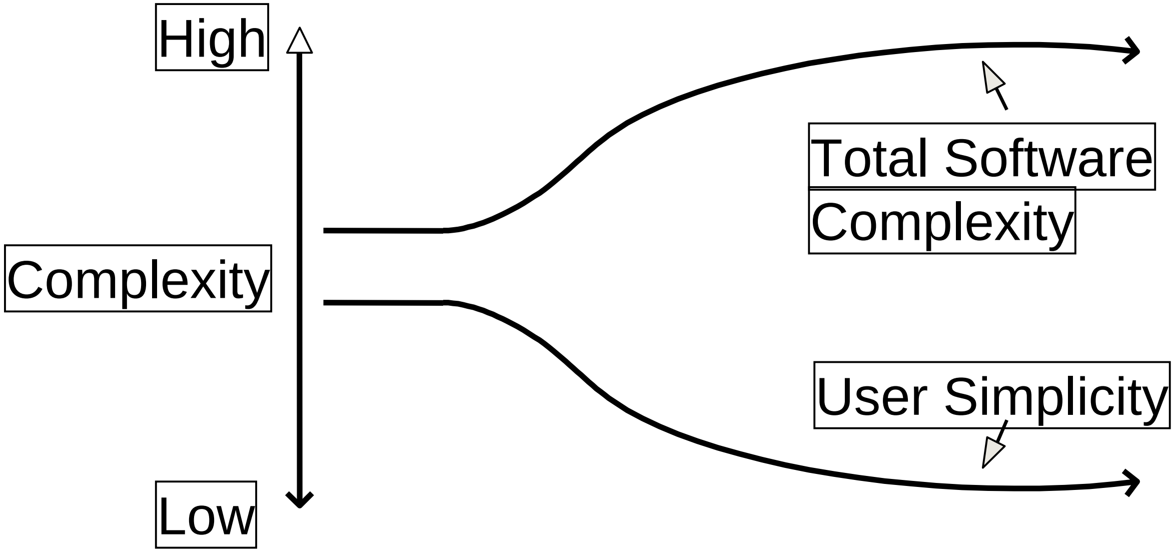
Engineering Software

- Software engineering
  - Area of computer science concerned with building large software systems

- Challenge
  - Tremendous advances in hardware have not been accompanied by comparable advances in software

Complexity Trade-off

- System complexity tends to grow as the system becomes more user friendly



Software Engineering Goals

- Reliability
  - An unreliable life-critical system can be fatal
- Understandability
  - Future development is difficult if software is hard to understand
- Cost Effectiveness
  - Cost to develop and maintain should not exceed profit
- Adaptability
  - System that is adaptive is easier to alter and expand
- Reusability
  - Improves reliability, maintainability, and profitability

Software Engineering Principles

- Abstraction
  - Extract the relevant properties while ignoring inessentials
- Encapsulation
  - Hide and protect essential information through a controlled interface
- Modularity
  - Dividing an object into smaller modules so that it is easier to understand and manipulate
- Hierarchy
  - Ranking or ordering of objects based on some relationship between them

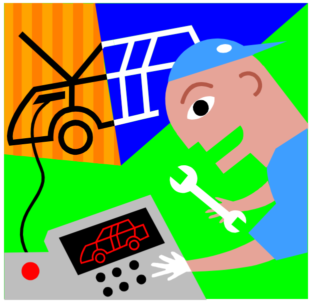
Abstraction

- Extract the relevant object properties while ignoring inessentials
  - Defines a view of the object

- Example - car
  - Car dealer views a car from selling features standpoint
    - Price, length of warranty, color, …
  - Mechanic views a car from systems maintenance standpoint
    - Size of the oil filter, type of spark plugs, …

Encapsulation

- Steps
  - Decompose an object into parts
  - Hide and protect essential information
  - Supply interface that allows information to be modified in a controlled and useful manner
- Internal representation can be changed without affecting other system parts
- Example - car radio
  - Interface consists of controls and power and antenna connectors
    - The details of how it works is hidden
  - To install and use a radio
    - Do not need to know anything about the radio’s electronics

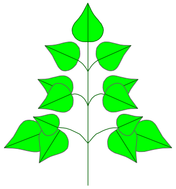


Modularity

- Dividing an object into smaller pieces or modules so that the object is easier to understand and manipulate
- Most complex systems are modular
- Example - Automobile can be decomposed into subsystems
  - Cooling system
    - Radiator Thermostat Water pump
  - Ignition system
    - Battery Starter Spark plugs

Hierarchy

- Hierarchy
  - Ranking or ordering of objects based on some relationship between them
- Help us understand complex systems
  - Example - a company hierarchy helps employees understand the company and their positions within it
- For complex systems, a useful way of ordering similar abstractions is a taxonomy from least general to most general



Northern Timber Wolf Taxonomy

Kingdom Animalia

Phylum Chordata

Class Mammalia

Order Carnivora

Family Caninae

Genus Canis

Species Canis lupus

Subspecies Canis lupus occidentalis

Northern Timber Wolf

OO Design and Programming

- Object-oriented design and programming methodology supports good software engineering
  - Promotes thinking in a way that models the way we think and interact with the real world
- Example - watching television

  - The remote is a physical object with properties
    - Weight, size, can send message to the television
  - The television is also a physical object with various properties

Objects

- An object is almost anything with the following characteristics
  - Name
  - Properties
  - The ability to act upon receiving a message
    - Basic message types
      - Directive to perform an action
      - Request to change one of its properties

Binary Arithmetic

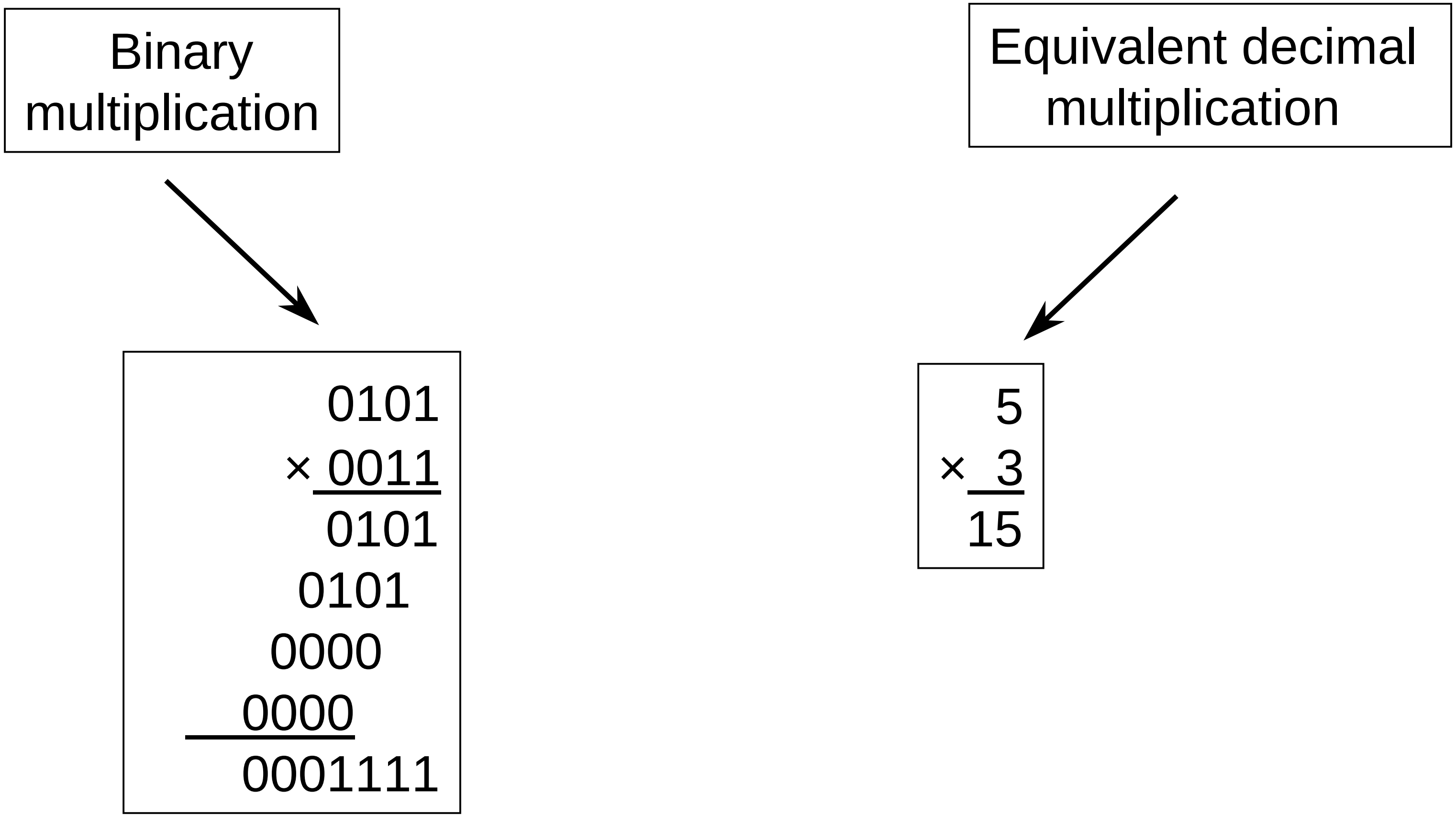
- The individual digits of a binary number are referred to as bits
  - Each bit represents a power of two

$$01011 = 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 11$$

$$00010 = 0 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 2$$

Binary addition → $\begin{array}{r} 00010 \\ \underline{+\ 01011} \\ 01101 \end{array}$ $\qquad$ $\begin{array}{r} 2 \\ \underline{+\ 11} \\ 13 \end{array}$ ← Equivalent decimal addition

Binary Arithmetic



Binary multiplication

Equivalent decimal multiplication

```
      0101
    × 0011
      0101
     0101
    0000
   0000
   0001111
```

```
   5
×  3
  15
```

Two’s Complement

- Representation for signed binary numbers
- Leading bit is a sign bit
  - Binary number with leading 0 is positive
  - Binary number with leading 1 is negative
- Magnitude of positive numbers is just the binary representation
- Magnitude of negative numbers is found by
  - Complement the bits
  - Replace all the 1's with 0's, and all the 0's with 1's
  - Add one to the complemented number
- The carry in the most significant bit position is thrown away when performing arithmetic

Two’s Complement

- Performing two's complement on the decimal 7 to get -7
    - Using a five-bit representation

```
7  =  00111   Convert to binary

      11000   Complement the bits

      11000   Add 1 to the complement
    + 00001
    -------
      11001   Result is  -7 in two's complement
```

Two's Complement Arithmetic

- Computing 8 - 7 using a two's complement representation with five-bit numbers

8 - 7 = 8 + (-7) = 1

01000 Two's complement of 8

11001 Two's complement of -7

$$\begin{array}{r} 01000 \\ +\ \ 11001 \\ \hline 100001 \end{array}$$

Add 8 and -7

Throw away the high-order carry as we are using a five bit representation

00001 Is the five-bit result