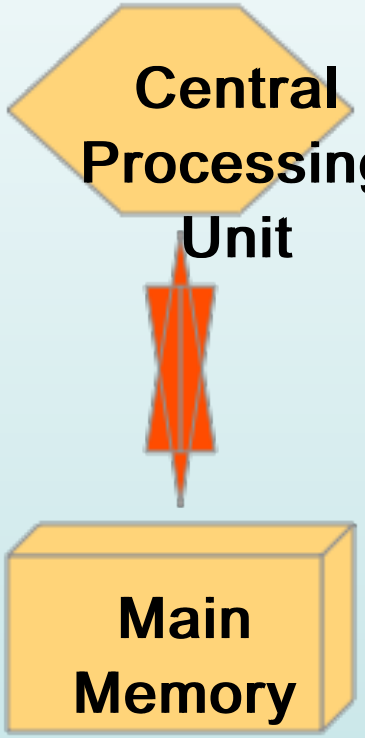
# Hardware and Software

- Hardware
  - the physical, tangible parts of a computer
  - keyboard, monitor, disks, wires, chips, etc.
- Software
  - programs and data
  - a *program* is a series of instructions
- A computer requires both hardware and software
- Each is essentially useless without the other

# CPU and Main Memory

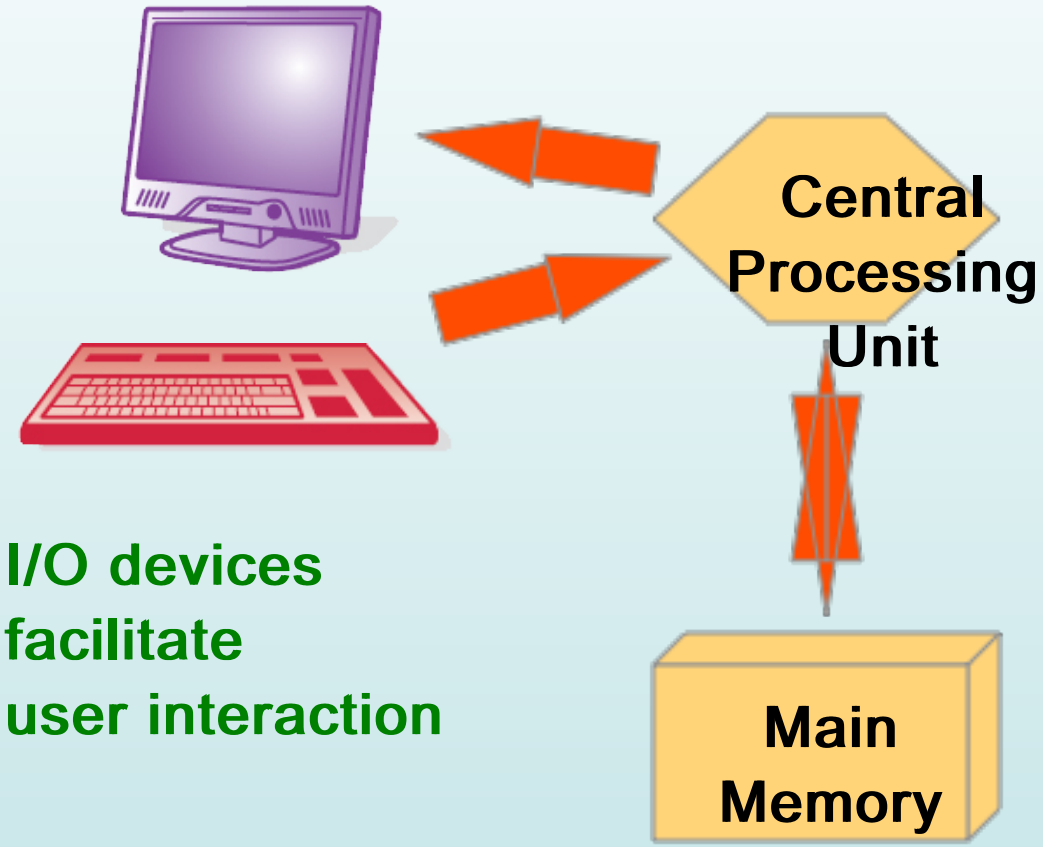



Chip that executes program commands

Primary storage area for programs and data that are in active use

Synonymous with RAM

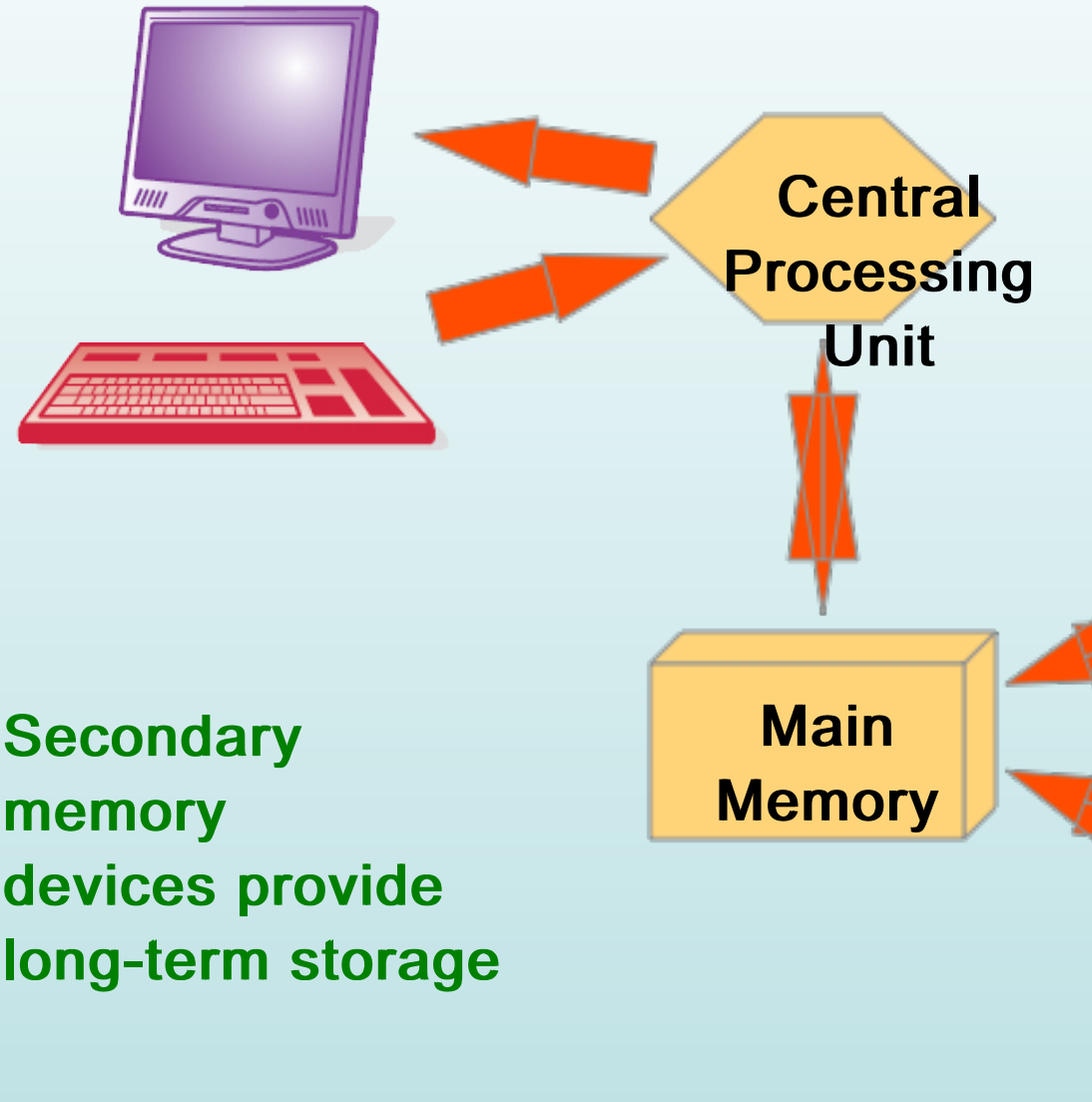
# Input / Output Devices




I/O devices
facilitate
user interaction

Monitor
screen
Keyboard
Mouse
Touch screen

# Secondary Memory Devices




Information is moved between main and secondary memory as needed

Secondary memory devices provide long-term storage

# Software Categories

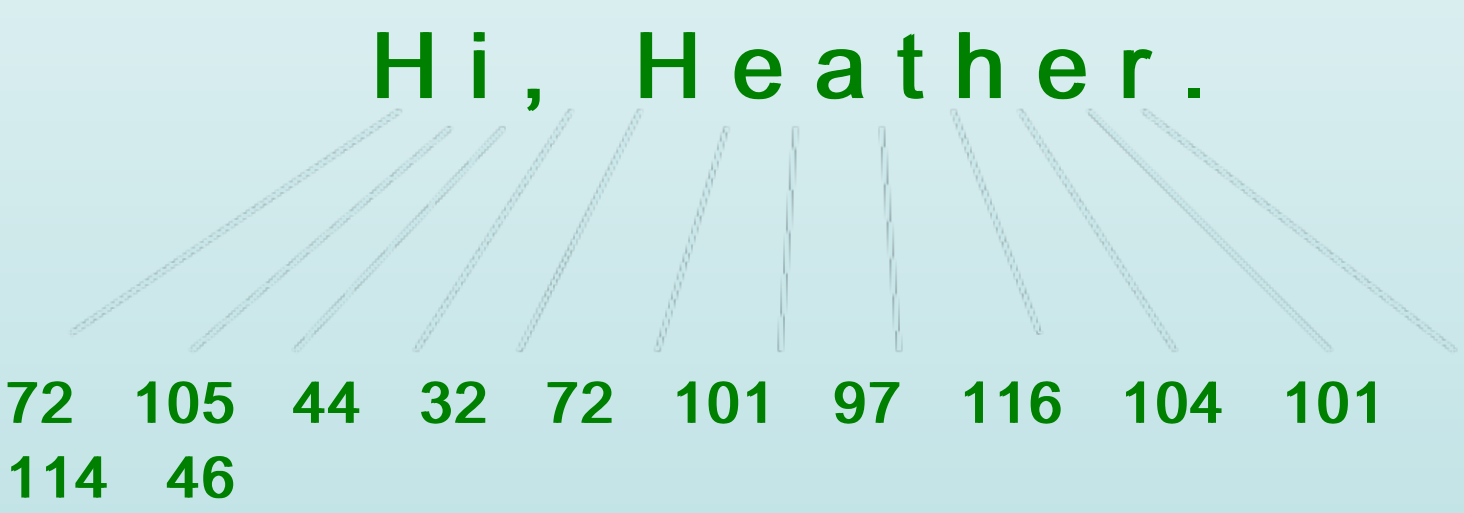
- Operating System
  - controls all machine activities
  - provides the user interface to the computer
  - manages resources such as the CPU and memory
  - Windows, Mac OS, Unix, Linux,
- Application program
  - generic term for any other kind of software
  - word processors, missile control systems, games
- Most operating systems and application programs have a *graphical user interface* (GUI)

# Digital Information

- Computers store all information digitally:
  - numbers
  - text
  - graphics and images
  - audio
  - video
  - program instructions
- In some way, all information is *digitized* - broken down into pieces and represented as numbers

# Representing Text Digitally

- For example, every character is stored as a number, including spaces, digits, and punctuation
- Corresponding upper and lower case letters are separate characters



# Binary Numbers

- Once information has been digitized, it is represented and stored in memory using the *binary number system*
- A single binary digit (0 or 1) is called a *bit*
- Devices that store and move information are cheaper and more reliable if they have to represent only two states
- A single bit can represent two possible states, like a light bulb that is either on (1) or off (0)
- Permutations of bits are used to store values

# Bit Permutations

| 1 bit | 2 bits | 3 bits | 4 bits | |
|---|---|---|---|---|
| 0 | 00 | 000 | 0000 | 1000 |
| 1 | 01 | 001 | 0001 | 1001 |
| | 10 | 010 | 0010 | 1010 |
| | 11 | 011 | 0011 | 1011 |
| | | 100 | 0100 | 1100 |
| | | 101 | 0101 | 1101 |
| | | 110 | 0110 | 1110 |
| | | 111 | 0111 | 1111 |

**Each additional bit doubles the number of possible permutations**

# Bit Permutations

- Each permutation can represent a particular item
- There are 2N permutations of N bits
- Therefore, N bits are needed to represent 2N unique items

**How many items can be represented by**

**1 bit ?** **21 = 2 items**

**2 bits** **22 = 4 items**

**?** **23 = 8 items**

**3 bits** **24 = 16 items**

**?** **25 = 32 items**

**4 bits**

**?**

# Quick Check

How many bits would you need to represent each of the 50 United States using a unique permutation of bits?

# Quick Check

How many bits would you need to represent each of the 50 United States using a unique permutation of bits?

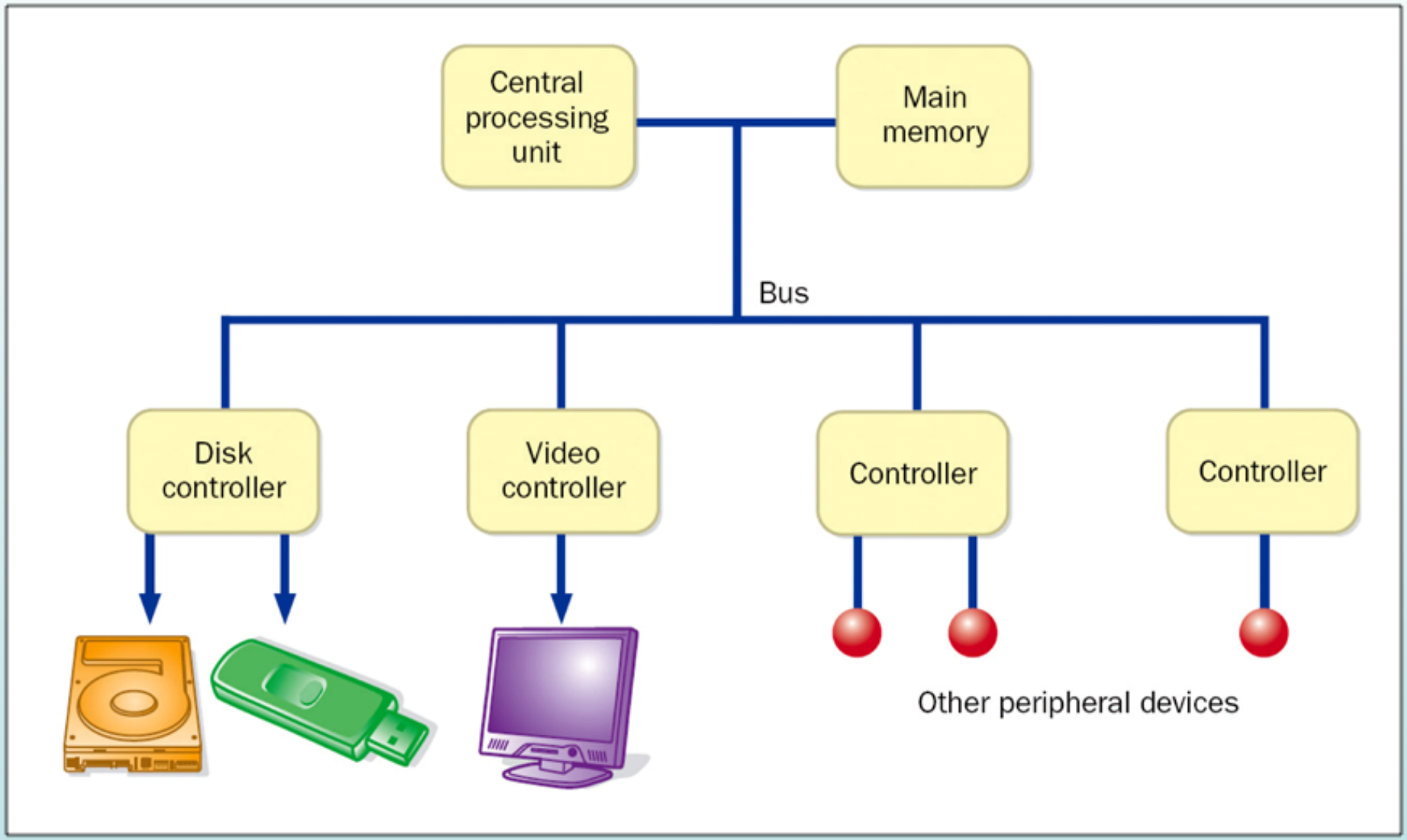
Five bits wouldn't be enough, because 25 is 32.

Six bits would give us 64 permutations, and some wouldn't be used.

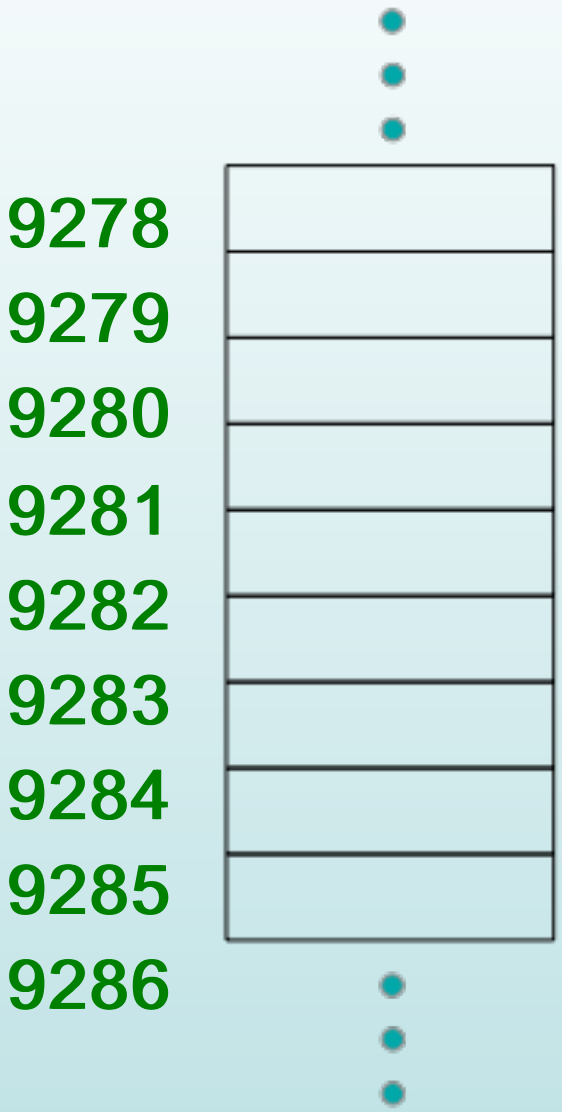
| | |
|---|---|
| 000000 | Alabama |
| 000001 | Alaska |
| 000010 | Arizona |
| 000011 | Arkansas |
| 000100 | California |
| 000101 | Colorado |

etc.

# Computer Architecture



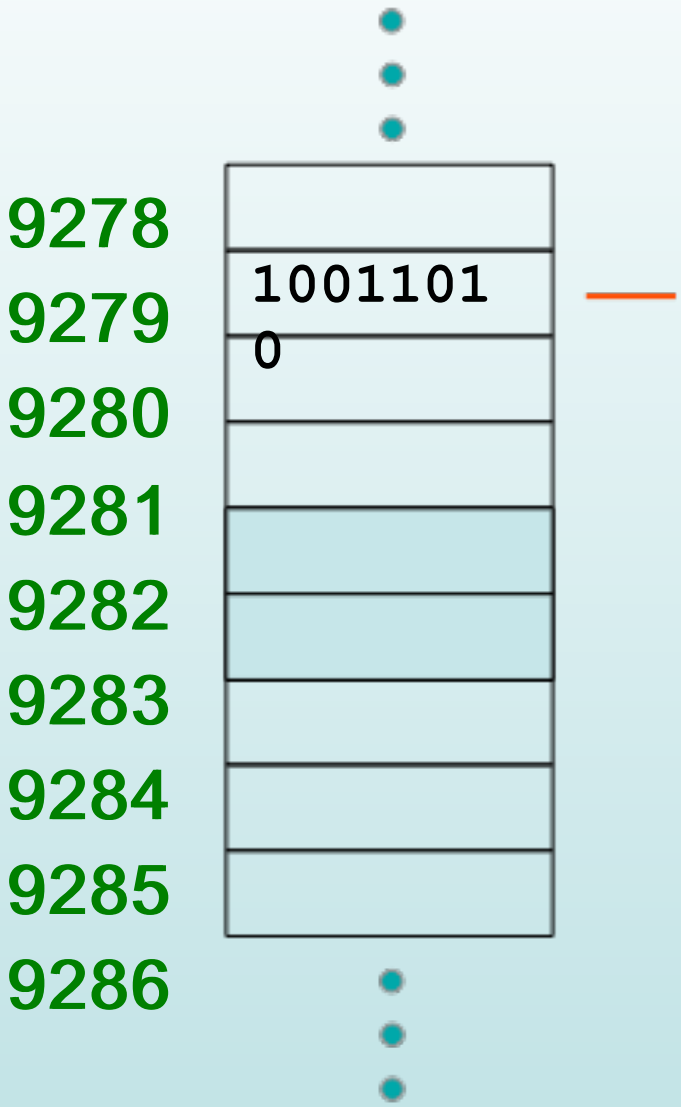
# Memory




Main memory is divided into many memory locations (or *cells*)

Each memory cell has a numeric *address*, which uniquely identifies it

# Storing Information




Each memory cell stores a set number of bits (usually 8 bits, or one *byte*)

Large values are stored in consecutive memory locations

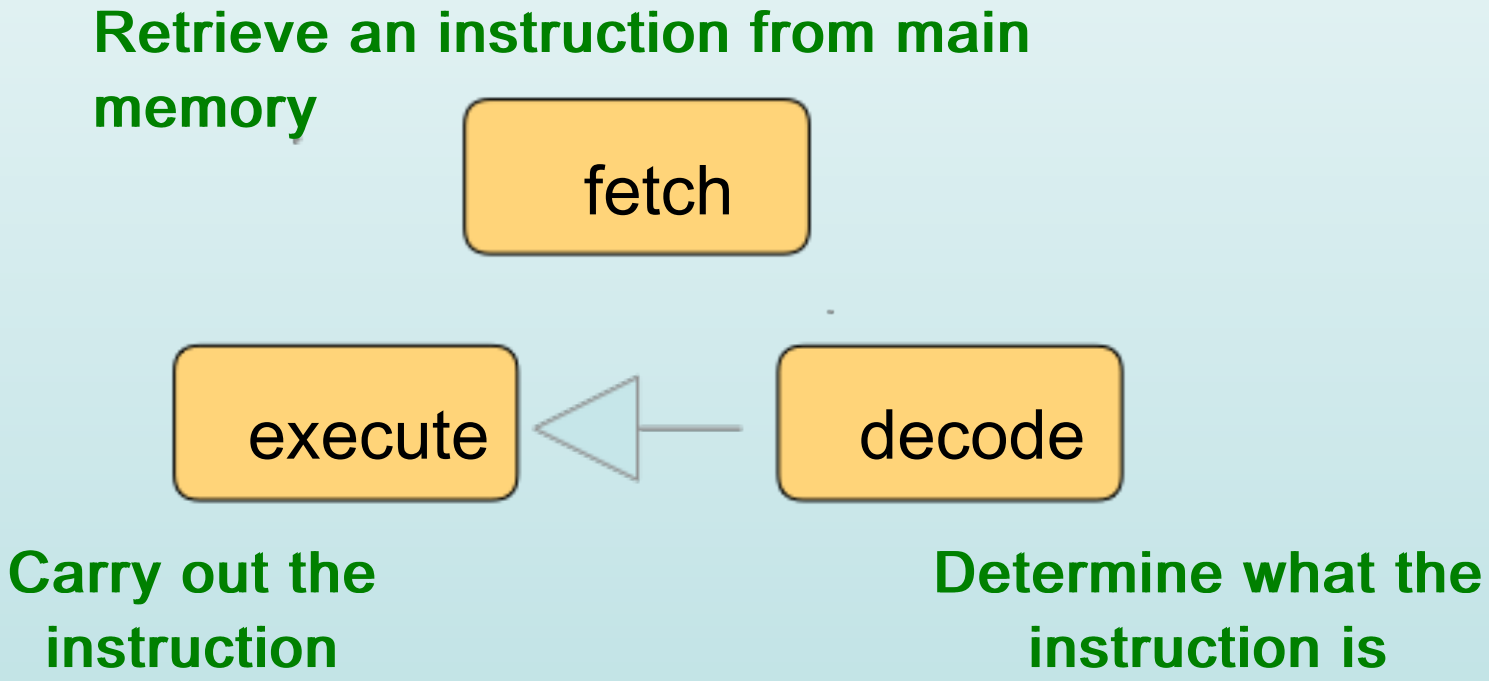
# Storage Capacity

- Every memory device has a *storage capacity*, indicating the number of bytes it can hold
- Capacities are expressed in various units:

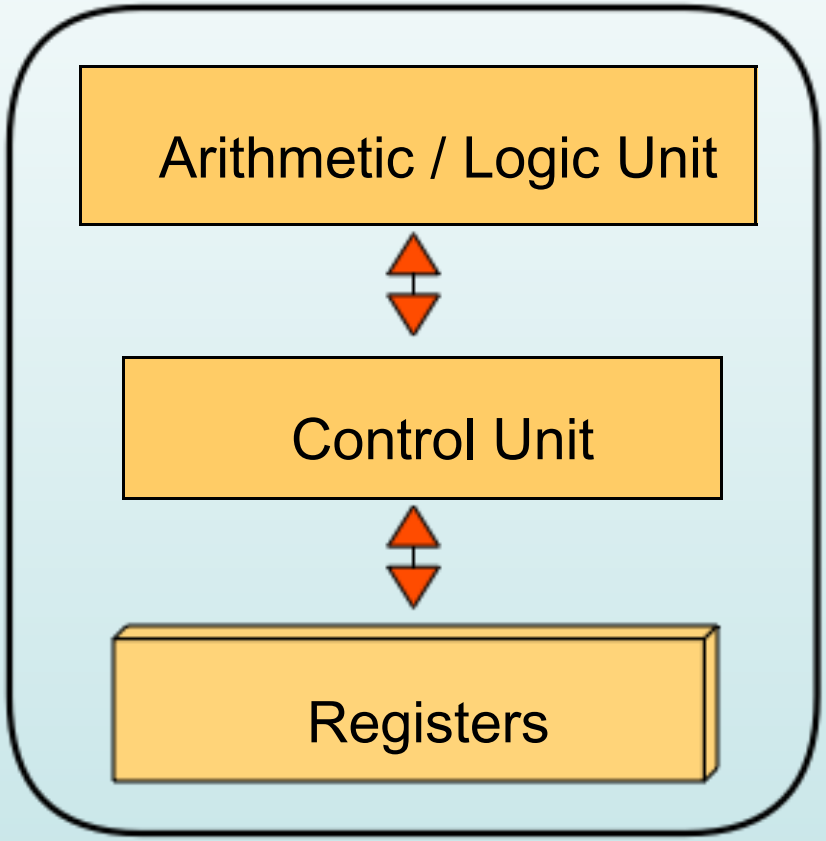
| Unit | Symbol | Number of Bytes |
|---|---|---|
| kilobyte | KB | 210 = 1024 |
| megabyte | MB | 220 (over one million) |
| gigabyte | GB | 230 (over one billion) |
| terabyte | TB | 240 (over one trillion) |
| petabyte | PB | 250 (a whole bunch) |

# The Central Processing Unit

- A CPU is on a chip called a *microprocessor*
- It continuously follows the *fetch-decode-execute cycle:*



# The Central Processing Unit




**Performs calculations and makes decisions**

**Coordinates processing steps**

**Small storage areas**

# Java

- The Java programming language was created by Sun Microsystems, Inc.
- It was introduced in 1995 and it's popularity has grown quickly since
- A *programming language* specifies the words and symbols that we can use to write a program
- A programming language employs a set of rules that dictate how the words and symbols can be put together to form valid *program statements*

# Identifiers

- *Identifiers* are the "words" in a program
- A Java identifier can be made up of letters, digits, the underscore character ( _ ), and the dollar sign
- Identifiers cannot begin with a digit
- Java is *case sensitive*: `Total`, `total`, and `TOTAL` are different identifiers
- By convention, programmers use different case styles for different types of identifiers, such as
  - *title case* for class names - `Lincoln`
  - *upper case* for constants - `MAXIMUM`

# Identifiers

- Sometimes the programmer chooses the identifer(such as `Lincoln`)
- Sometimes we are using another programmer's code, so we use the identifiers that he or she chose (such as `println`)
- Often we use special identifiers called *reserved words* that already have a predefined meaning in the language
- A reserved word cannot be used in any other way

# Reserved Words

- The Java reserved words:

```
abstract    else        interface   switch
assert      enum        long        synchronized
boolean     extends     native      this
break       false       new         throw
byte        final       null        throws
case        finally     package     transient
catch       float       private     true
char        for         protected   try
class       goto        public      void
const       if          return      volatile
continue    implements  short       while
default     import      static
do          instanceof  strictfp
double      int         super
```

# Program Development

- The mechanics of developing a program include several activities:
  - writing the program in a specific programming language (such as Java)
  - translating the program into a form that the computer can execute
  - investigating and fixing various types of errors that can occur
- Software tools can be used to help with all parts of this process

# Language Levels

- There are four programming language levels:
  - machine language
  - assembly language
  - high-level language
  - fourth-generation language
- Each type of CPU has its own specific *machine language*
- The other levels were created to make it easier for a human being to read and write programs

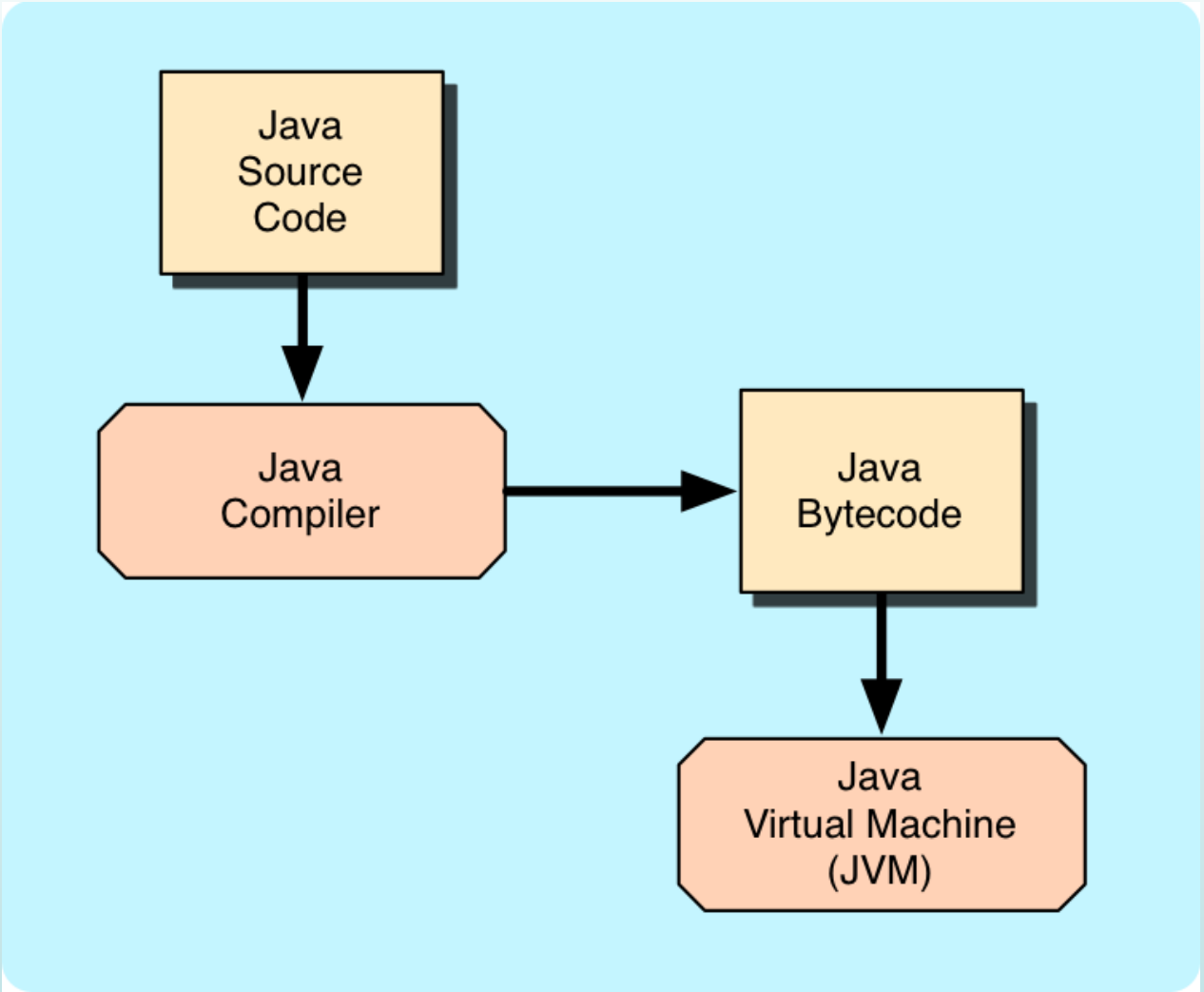
# Programming Languages

- Each type of CPU executes only a particular *machine language*
- A program must be translated into machine language before it can be executed
- A *compiler* is a software tool which translates *source code* into a specific target language
- Sometimes, that target language is the machine language for a particular CPU type
- The Java approach is somewhat different

# Java Translation

- The Java compiler translates Java source code into a special representation called *bytecode*
- Java bytecode is not the machine language for any traditional CPU
- Bytecode is executed by the *Java Virtual Machine* (JVM)
- Therefore Java bytecode is not tied to any particular machine
- Java is considered to be *architecture-neutral*

# Java Translation



# Development Environments

- There are many programs that support the development of Java software, including:
  - Java Development Kit (JDK)
  - Eclipse
  - NetBeans
  - BlueJ
  - jGRASP
- Though the details of these environments differ, the basic compilation and execution process is essentially the same

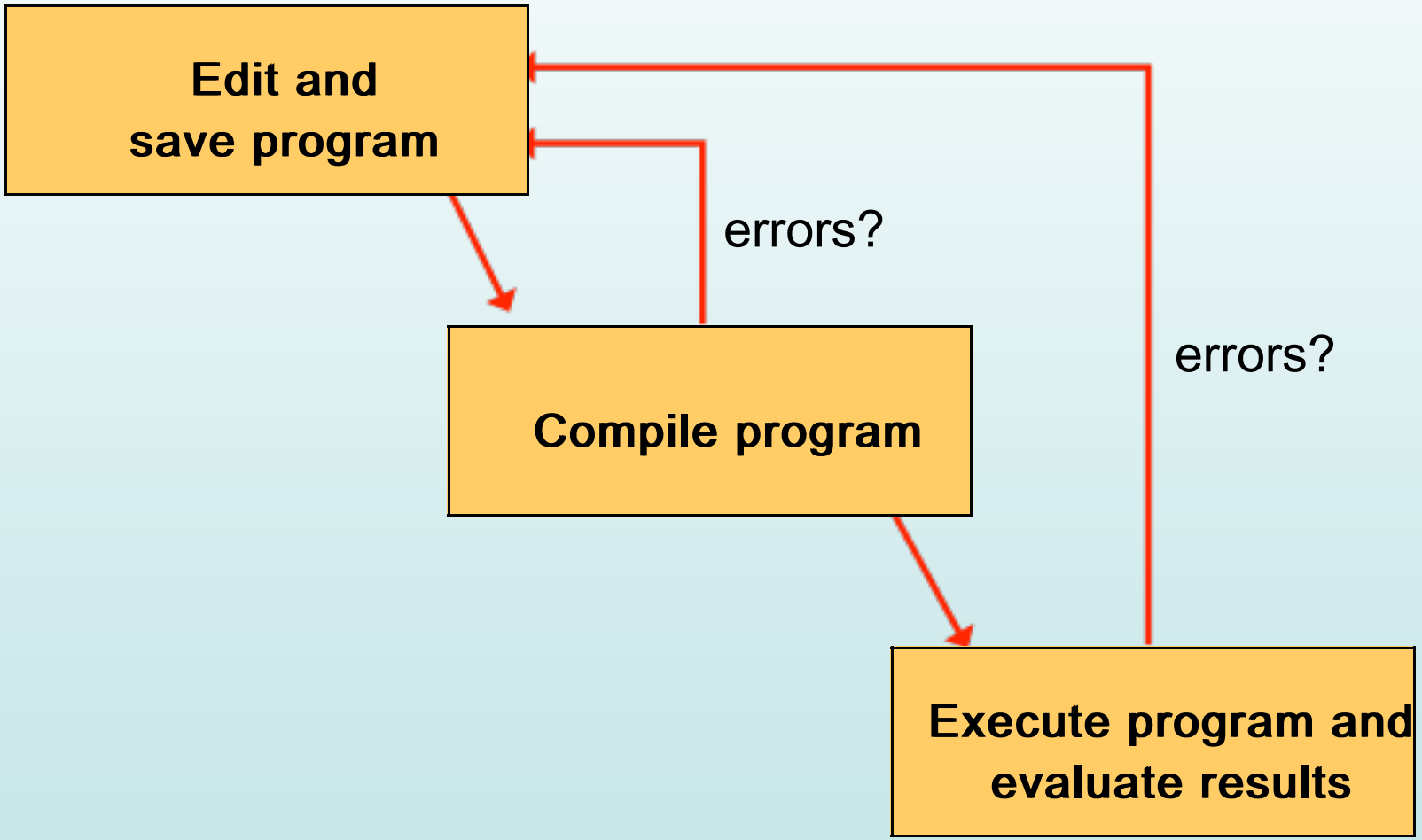
# Syntax and Semantics

- The *syntax rules* of a language define how we can put together symbols, reserved words, and identifiers to make a valid program
- The *semantics* of a program statement define what that statement means (its purpose or role in a program)
- A program that is syntactically correct is not necessarily logically (semantically) correct
- A program will always do what we tell it to do, not what we meant to tell it to do

# Errors

- A program can have three types of errors
- The compiler will find syntax errors and other basic problems (*compile-time errors*)
  - If compile-time errors exist, an executable version of the program is not created
- A problem can occur during program execution, such as trying to divide by zero, which causes a program to terminate abnormally (*run-time errors*)
- A program may run, but produce incorrect results, perhaps using an incorrect formula (*logical errors*)

# Basic Program Development



# Problem Solving

- The purpose of writing a program is to solve a problem
- Solving a problem consists of multiple activities:
  - Understand the problem
  - Design a solution
  - Consider alternatives and refine the solution
  - Implement the solution
  - Test the solution
- These activities are not purely linear – they overlap and interact

# Problem Solving

- The key to designing a solution is breaking it down into manageable pieces
- When writing software, we design separate pieces that are responsible for certain parts of the solution
- An *object-oriented approach* lends itself to this kind of solution decomposition
- We will dissect our solutions into pieces called objects and classes

# Object-Oriented Programming

- Java is an object-oriented programming language
- As the term implies, an object is a fundamental entity in a Java program
- Objects can be used effectively to represent real-world entities
- For instance, an object might represent a particular employee in a company
- Each employee object handles the processing and data management related to that employee

# Objects

- An object has:
  - *state* - descriptive characteristics
  - *behaviors* - what it can do (or what can be done to it)
- The state of a bank account includes its account number and its current balance
- The behaviors associated with a bank account include the ability to make deposits and withdrawals
- Note that the behavior of an object might change its state

# Classes

- An object is defined by a *class*
- A class is the blueprint of an object
- The class uses methods to define the behaviors of the object
- The class that contains the main method of a Java program represents the entire program
- A class represents a concept, and an object represents the embodiment of that concept
- Multiple objects can be created from the same class