Introduction to Algorithms and Data Structures

CSC 1051 – Algorithms and Data Structures I

Dr. Mary-Angela Papalaskari
Department of Computing Sciences
Villanova University

Course website:

www.csc.villanova.edu/~map/1051/

Some slides in this presentation are adapted from the slides accompanying Java Software Solutions by Lewis & Loftus

What is this course about?

- Computer Science
- Problem solving
- Algorithmic thinking
- Data representation
- Object oriented programming using Java

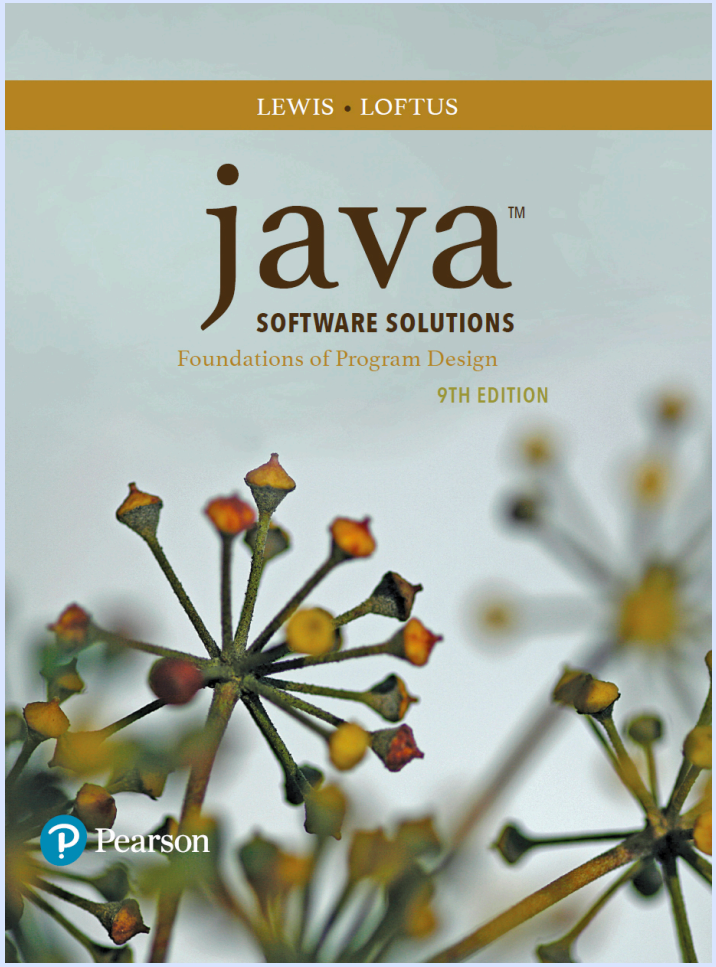
Course website

www.csc.villanova.edu/~map/1051/

Links to:

- **Schedule** – topics, slides, projects, labs, code, etc.
- **Syllabus** – course information
- **Piazza** – class discussions, announcements
- **Blackboard** – submit projects, check grades
- **Peer Tutors** – extra help available for this course
- **Exam archive** – past exams and quizzes & solutions

Our textbook



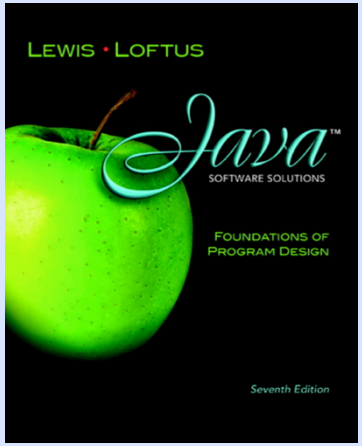
Java Software Solutions

John Lewis & William Loftus

9th Edition

On reserve at the library

Older Editions: (generally ok, except for graphics sections and some exercise numbers that may be different)



Addison-Wesley
is an imprint of
PEARSON

An old quote

A priest asked: “What is Fate, Master?”
And he answered:
“It is that which gives a beast of burden its reason for existence. It is that which men in former times had to bear upon their backs. It is that which has caused nations to build byways from City to City upon which carts and coaches pass, and alongside which inns have come to be built to stave off Hunger, Thirst and Weariness.”
“And that is Fate?” said the priest.
“Fate... I thought you said Freight,” responded the Master.
“That's all right,” said the priest. “I wanted to know what Freight was too.”

- Kehlog Albran

Source unknown. This quote appeared as one of the “fortunes” displayed by the fortune cookie program on old unix systems. (“fortune” was a program that ran automatically every time you logged out of a unix session and displayed a random, pithy saying.)

Java and other high-level programming languages

- Programmer writes **Source code**
- Translation produces **Machine code** (binary code suitable to run on the computer)
- Translation is performed by an assembler, compiler, or interpreter (stay tuned)

Java Program Structure

- In the Java programming language:
  - A program is made up of one or more *classes*
  - A class contains one or more *methods*
  - A method contains program *statements*
- These terms will be explored in detail throughout the course
- A Java application always contains a method called `main`
- See `Lincoln.java`

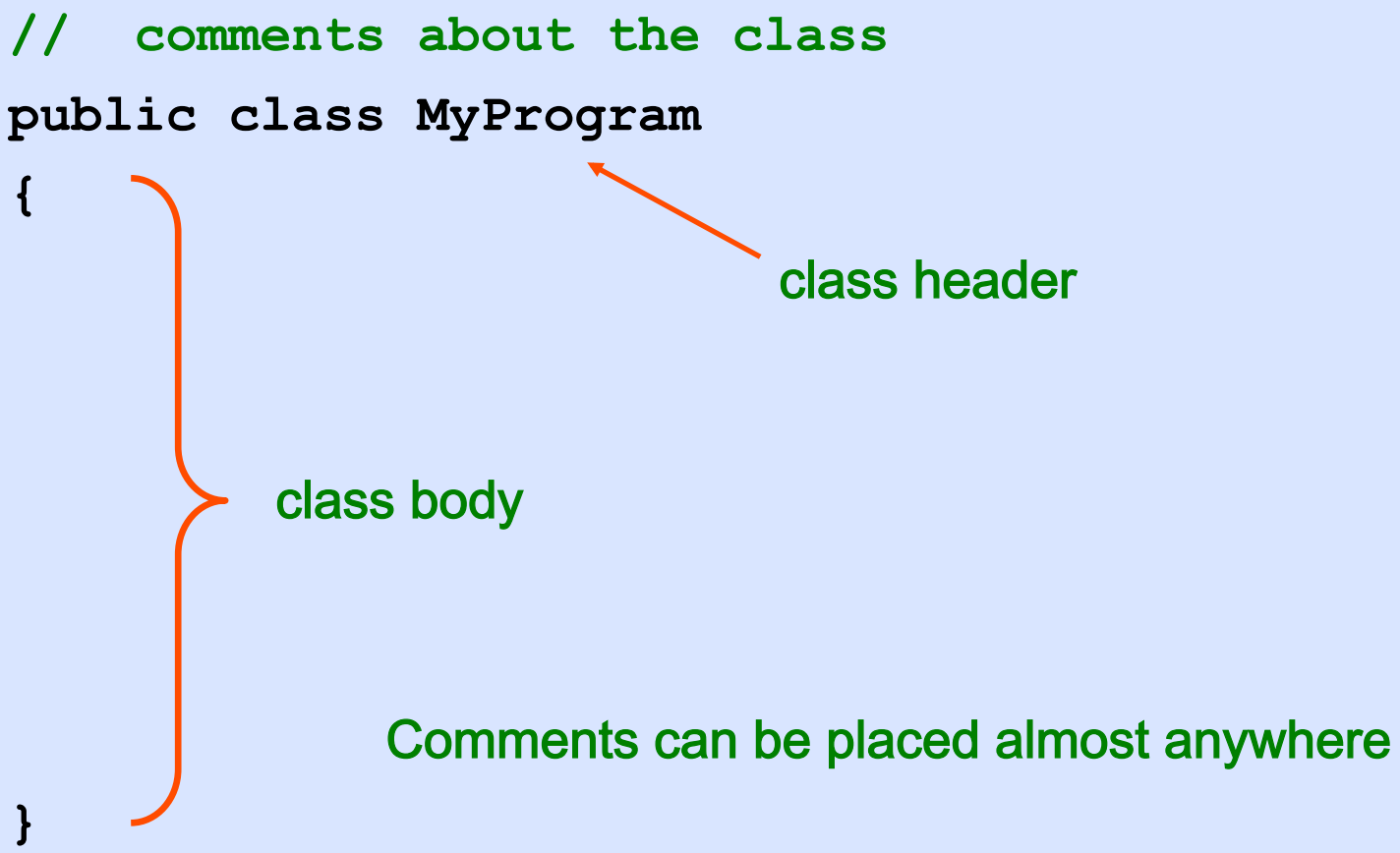
Java Program Example

```
//********************************************************************
//  Lincoln.java       Author: Lewis/Loftus
//
//  Demonstrates the basic structure of a Java application.
//********************************************************************

public class Lincoln
{
   //-----------------------------------------------------------------
   //  Prints a presidential quote.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      System.out.println ("A quote by Abraham Lincoln:");

      System.out.println ("Whatever you are, be a good one.");
   }
}
```

Java Program Structure



Java Program Structure

```
//  comments about the class
public class MyProgram
{
   //  comments about the method
   public static void main (String[] args)
   {

   }

}
```

method body

method header

Comments

- Comments in a program are called *inline documentation*
- They should be included to explain the purpose of the program and describe processing steps
- They do not affect how a program works
- Alternative ways of making Java comments:

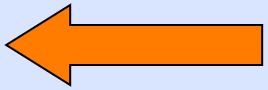
```
// This comment runs to the end of the line

/*  This comment runs to the terminating
    symbol, even across line breaks       */
```

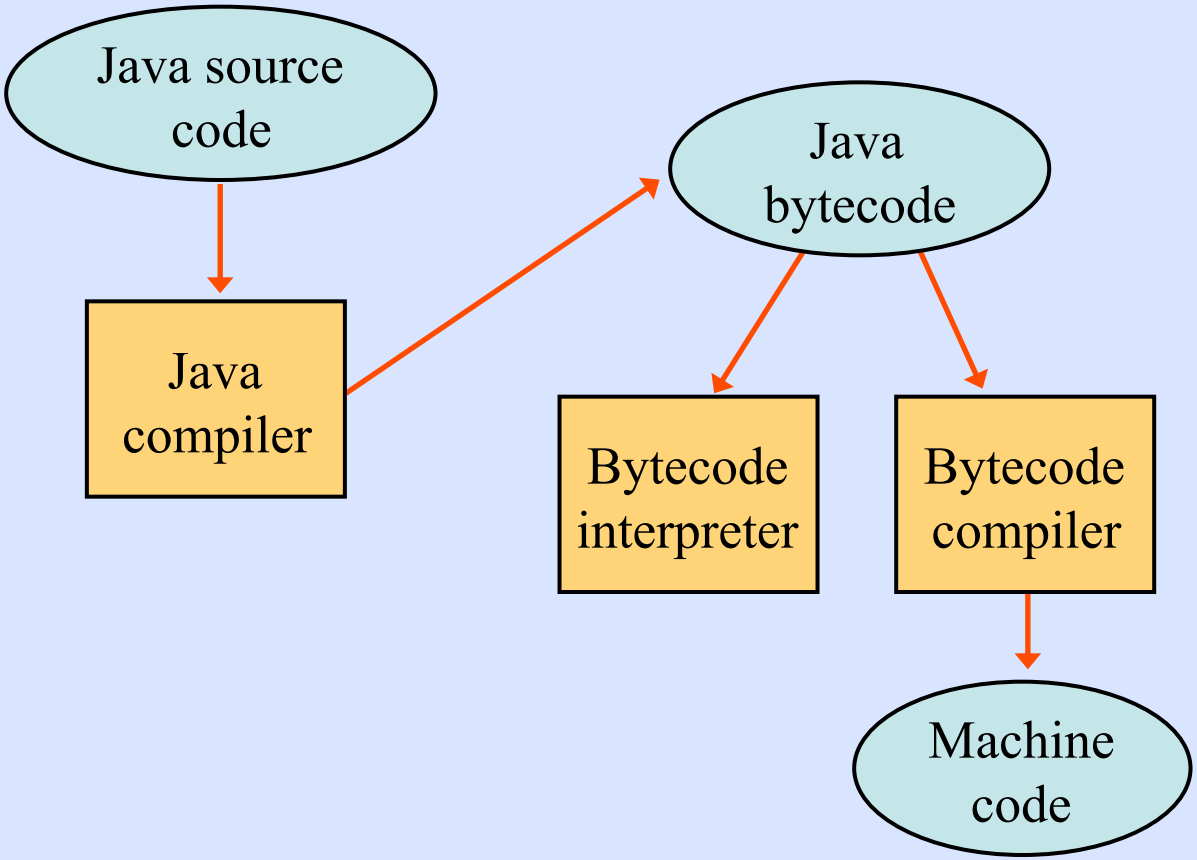
White Space (Spaces, blank lines, and tabs)

- Extra white space is ignored
- Programs should be formatted to enhance readability, using consistent indentation
- See Lincoln2.java, Lincoln3.java

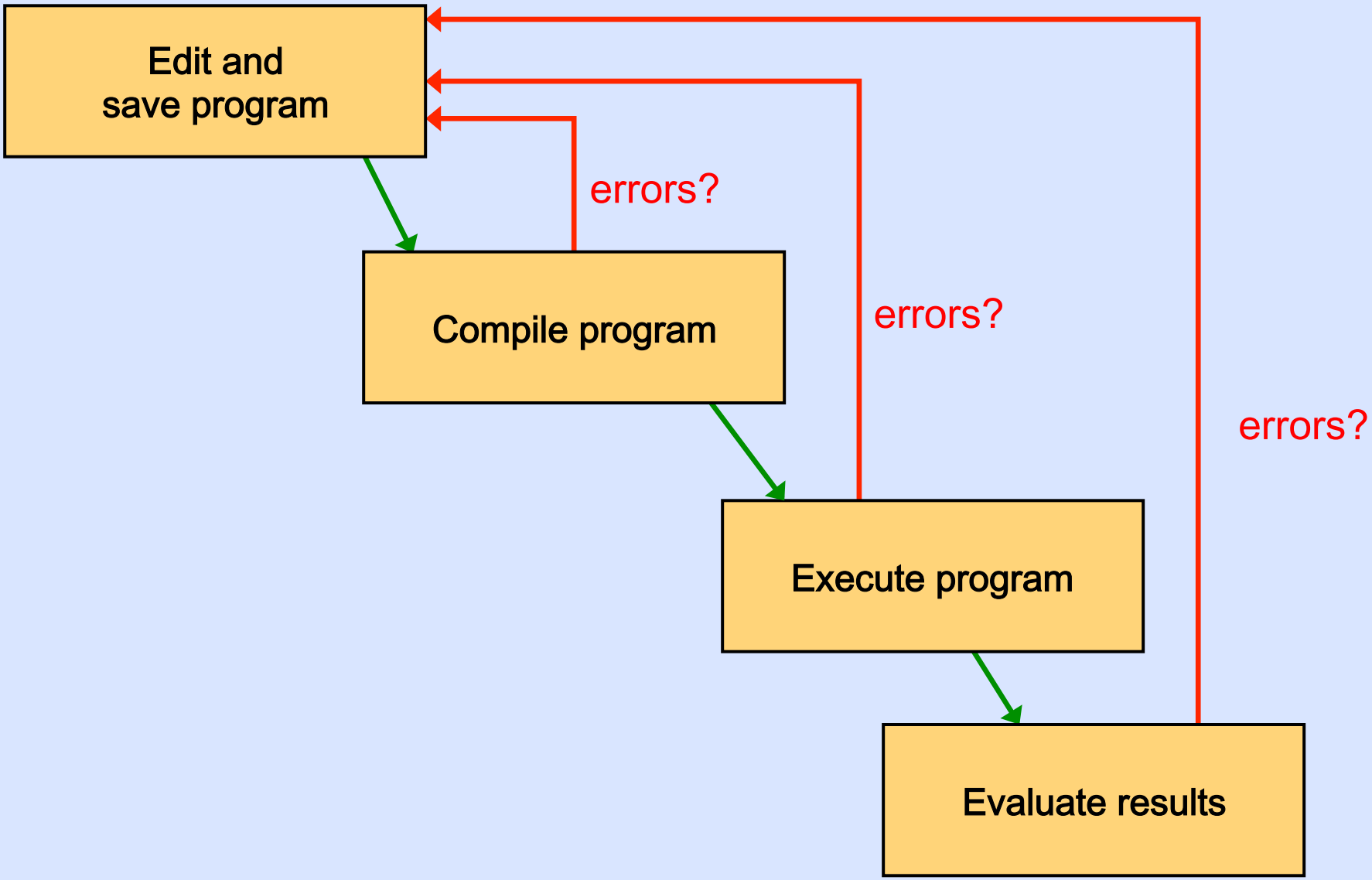
Development Environments

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

Java Translation



Basic Program Development

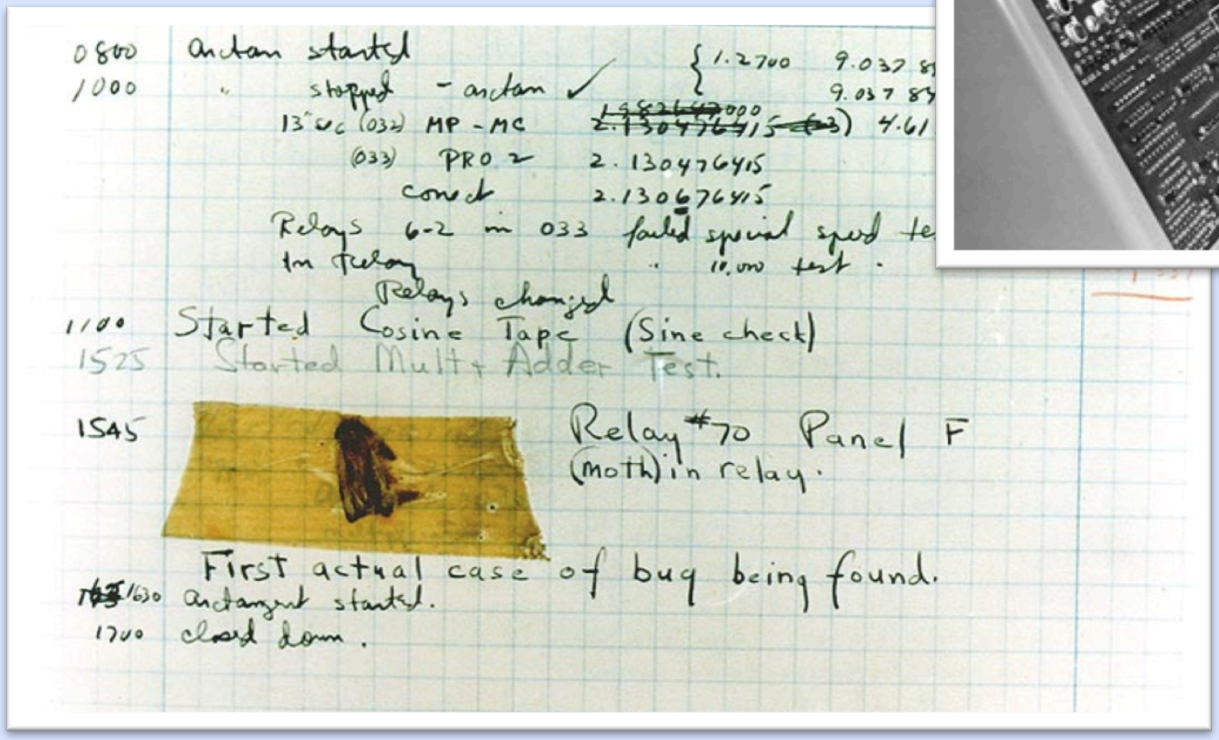


Errors

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*)

The original ”bug” found in the relays of Harvard’s Mark II computer by Admiral Grace Murray Hopper’s team.



Source: en.wikipedia.org/wiki/File:H96566k.jpg

Lab 1:

- Learn about jGrasp - the programming environment (IDE) that we will be using
- Compile  .. and run  java program
- Understand the relationship between a Java class name and file names
- Practice using basic Java output statements and adding comments
- Learn the basics of sequential execution, variables, and the assignment statement

```
System.out.println ("Howdy " + name);
System.out.println ("The answer is " + x);
System.out.print ("Counting... up: " + (count + 1));
System.out.println ("  ... and\n ... down: " + (count - 1));
```

- Experience some errors!

Identifiers

- Identifiers are used for naming variables, classes, and other components of a program.
- An identifier can be made up of:
  - letters (upper or lower case – *case sensitive!*)
  - digits (but cannot begin with a digit)
  - underscore character (_)
  - the dollar sign ($)
  - NOTHING ELSE!
- Example: `Total`, `total`, and `TOTAL` are different identifiers
- Conventions: use case to indicate whether it is a class or a variable etc.

Reserved Words

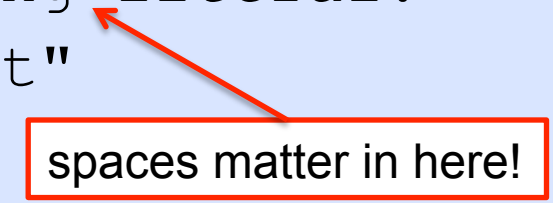
These identifiers have a special meaning in Java and cannot be used in any other way:

`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`	

Character Strings

- A *string literal* is represented by putting double quotes around the text
- Examples:

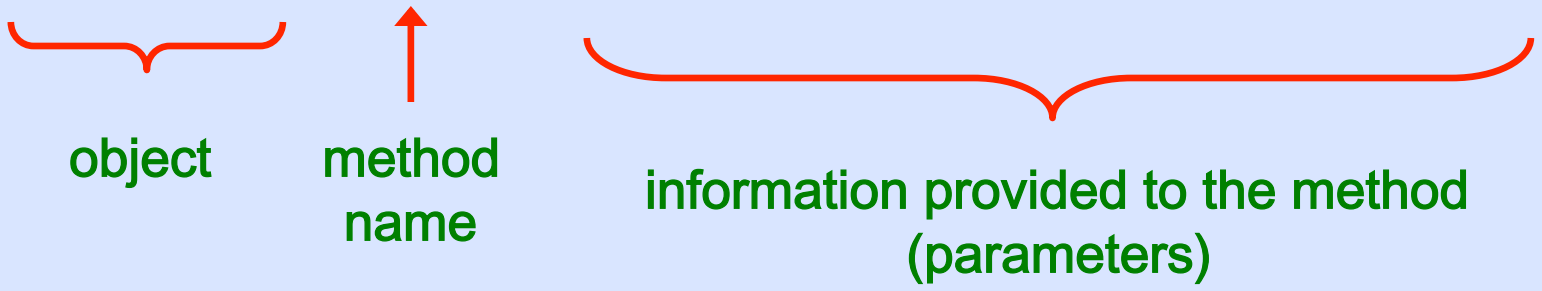
```
"This is a string literal."
"123 Main Street"
"X"
```



The println Method

- In the `Lincoln` program we invoked the `println` method to print a character string
- The `System.out` object represents a destination (the monitor screen) to which we can send output

```
System.out.println ("Whatever you are, be a good one.");
```



The print Method

- In the `Lincoln` program we invoked the `println` method to print a character string
- The `System.out` object represents a destination (the monitor screen) to which we can send output
- `print` is similar to `println` except that it does not advance to the next line

```
System.out.print ("Whatever you are, be a good one.");
```

object

method name

information provided to the method (parameters)

String Concatenation

- The *string concatenation operator* (+) is used to append one string to the end of another

```
"And one more " + "thing"
```

Hands on:

- Use MyQuote.java as a starting point (program from Lab 1), focus on this part of the code:

```
System.out.println ("Howdy " + name);
System.out.println ("The answer is " + x);
System.out.print ("Counting... up: " + (count + 1));
System.out.println ("  ... and\n ... down: " + (count - 1));
```

- Try the following:
  1) What if you remove the parentheses around (count + 1)?
  2) What happens if we try this way of breaking a line:

```
System.out.print ("Counting...
                  up: " + (count + 1));
```

  3) How can we get all this output to print all in one line?

- Other examples (textbook): Countdown.java Facts.java

Escape Sequences

- What if we wanted to print the quote character? e.g.,

```
System.out.println ("I said "Hello" to you."); // wrong!
```

- An *escape sequence* is a series of characters that represents a special character.
- Example:

```
System.out.println ("I said \"Hello\" to you.");
```

- Some Java escape sequences:

Escape Sequence	Meaning
`\t`	tab
`\n`	newline
`\"`	double quote
`\'`	single quote
`\\`	backslash

Example from textbook: Roses.java

```
//********************************************************************
//  Roses.java       Author: Lewis/Loftus
//
//  Demonstrates the use of escape sequences.
//********************************************************************

public class Roses
{
   //-----------------------------------------------------------------
   //  Prints a poem (of sorts) on multiple lines.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      System.out.println ("Roses are red,\n\tViolets are blue,\n" +
         "Sugar is sweet,\n\tBut I have \"commitment issues\",\n\t" +
         "So I'd rather just be friends\n\tAt this point in our " +
         "relationship.");
   }
}
```

Output

```
Roses are red,
        Violets are blue,
Sugar is sweet,
        But I have "commitment issues",
        So I'd rather just be friends
        At this point in our relationship.
```

Quick Check

Write a single `println` statement that produces the following output:

"Thank you all for coming to my home
tonight," he said mysteriously.

Context:
Reverse History of computing

Examine what we already know, travel backwards…

1. What we see now all around us – a connected world of computing

2. Focus on a single “traditional” computer

3. Dig deeper – data and processing

Computer Networks

- A network is two or more computers that are connected so that data and resources can be shared
  - Local Area Network (LAN) / Wide Area Network (WAN)
- The Internet is a WAN which spans the planet
  - The Internet Protocol (IP) determines how data are routed
  - devices have unique *IP addresses, e.g.,* `204.192.116.2`
- The *World Wide Web* provides a common interface to internet data:
  - text, graphics, video, sound, audio, executable programs
  - Web documents often use *HyperText Markup Language* (HTML)
  - Uses *Uniform Resource Locator* (URL), eg: **`http://www.whitehouse.gov/issues/education`**
- A *browser* accesses network resources
  - Popular browsers: Chrome, Internet Explorer, Safari, Firefox
  - My first browser: Mosaic <3

Historical Note: **Connecting the world**

The Internet

History: Started as a United States government project, sponsored by the Advanced Research Projects Agency (ARPA) in late 1960's

See also: http://www.internethalloffame.org/internet-history/timeline

- 1970's and 1980's: ***ARPANET***
  - wide area network
  - protocols for communication
- 1990's: ***World Wide Web***
  - html and web browsers

1969: first four nodes of the internet

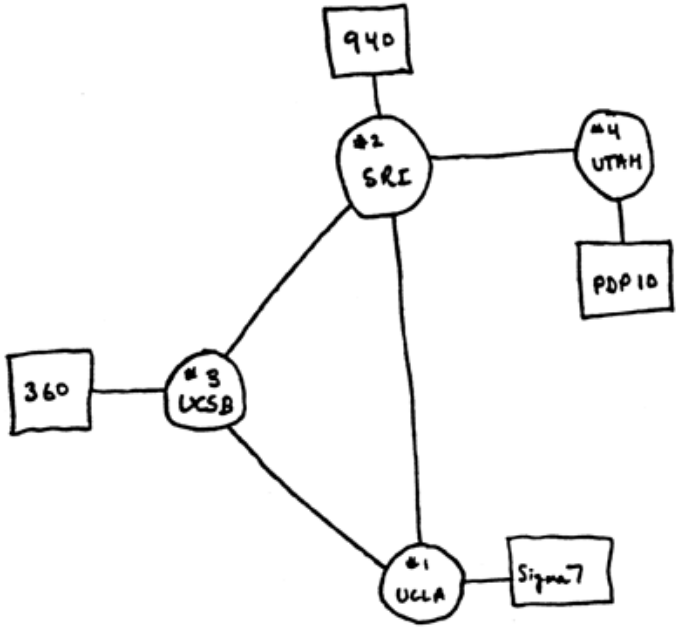


Image source: http://www.computerhistory.org/internet_history/full_size_images/1969_4-node_map.gif

Historical Note: **Connecting the world**

The Arpanet in 1971

'Interface Message Processor' (IMP) evolved into today's routers.

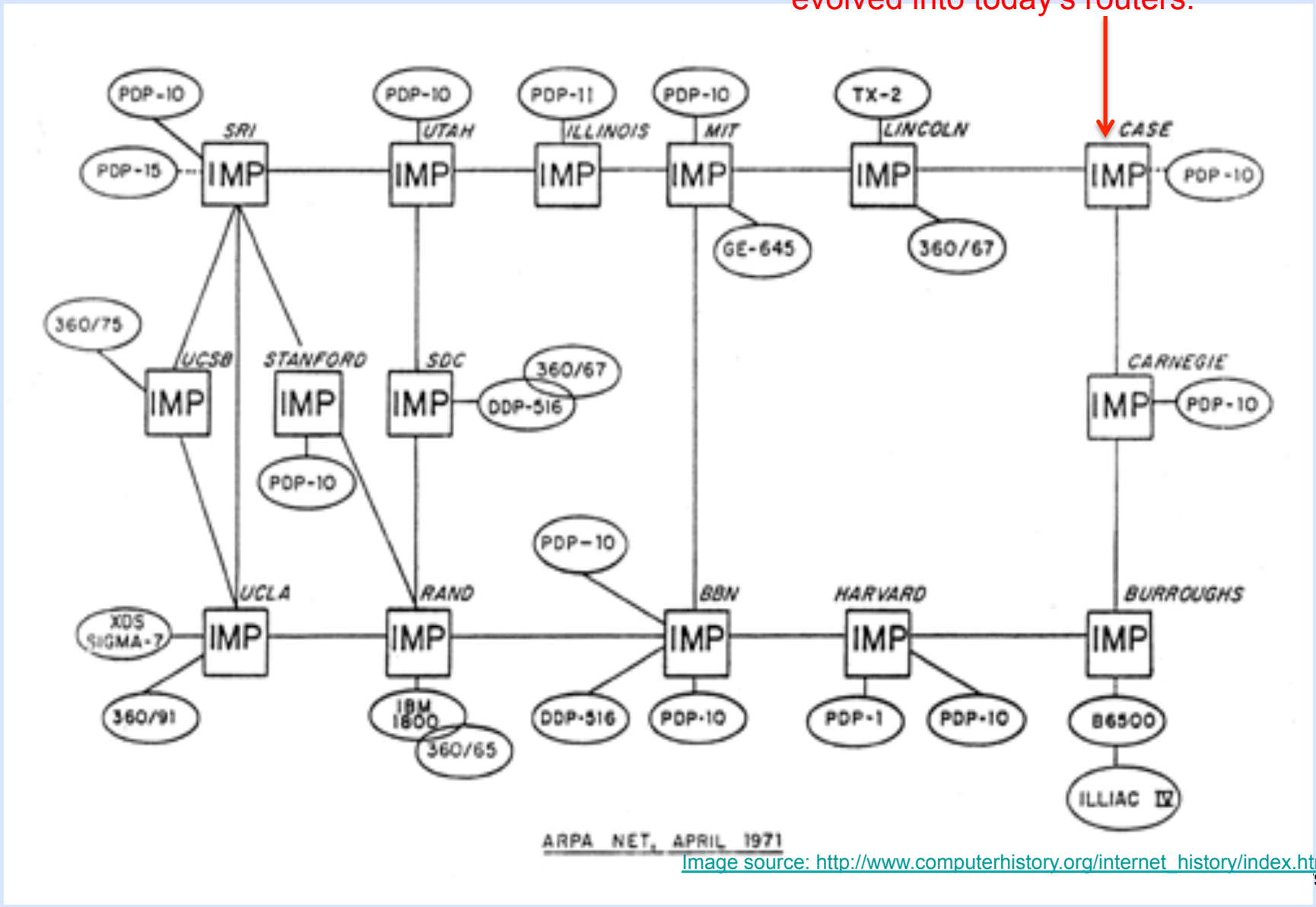


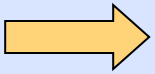
Image source: http://www.computerhistory.org/internet_history/index.html

sity

Reverse History of computing

Examine what we already know, travel backwards…

1. What we see now all around us – a connected world of computing



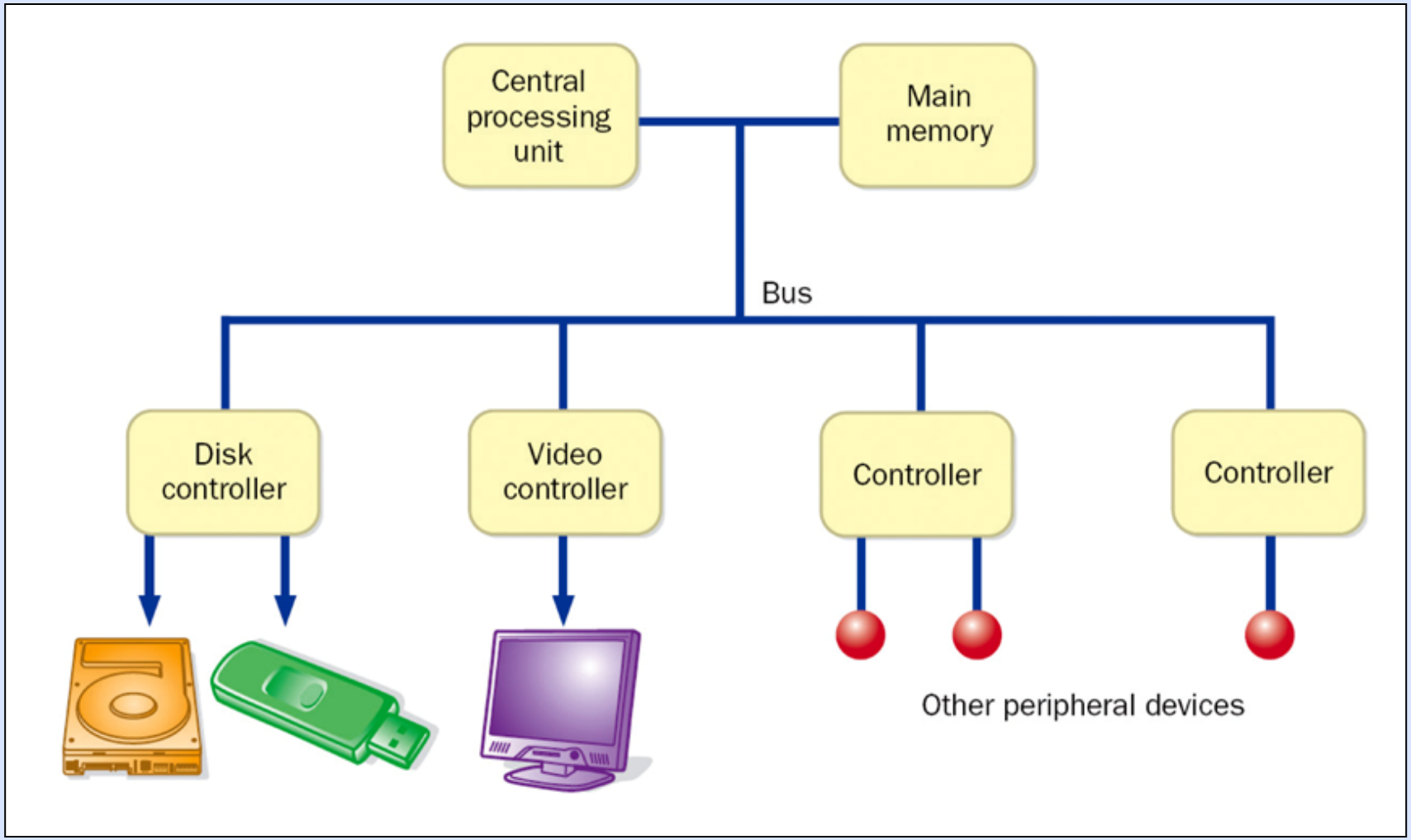
2. Focus on a single “traditional” computer

3. Dig deeper – data and processing

A Computer Specification

- Consider the following specification for a personal computer:
  - 3.07 GHz Intel Core i7 processor
  - 4 GB RAM
  - 750 GB Hard Disk
  - 16x Blu-ray / HD DVD-ROM & 16x DVD+R DVD Burner
  - 17” Flat Screen Video Display with 1280 x 1024 resolution
  - Network Card

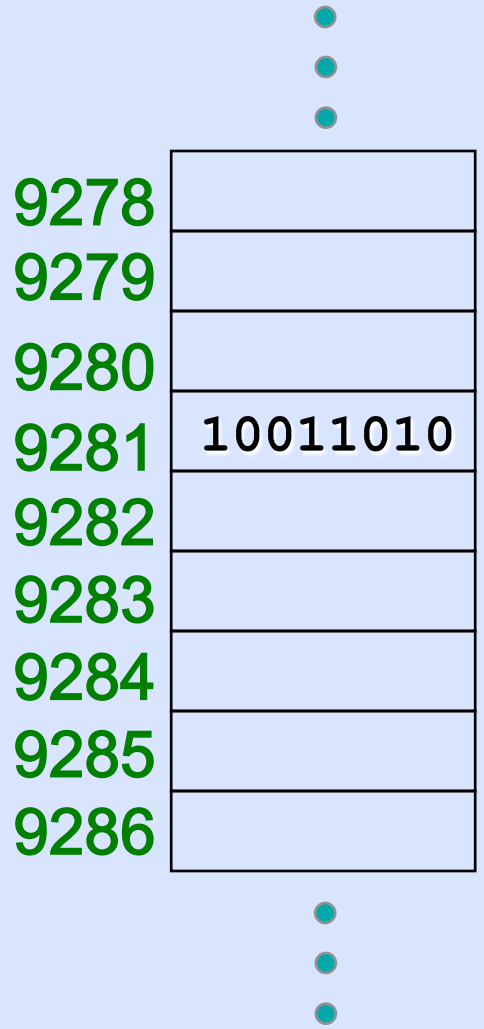
Computer Architecture



Memory

memory locations (or cells)

identified by a unique numeric address

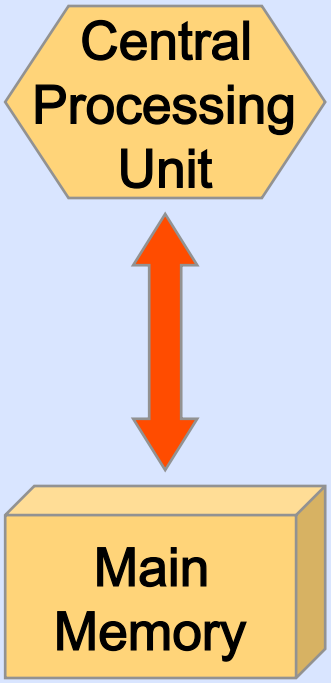


jargon alert!

Memory = Main Memory = Random Access Memory = RAM
("Random" because you don't have to scan the memory sequentially – go to data directly using the address)

CPU and Main Memory

Chip that executes program commands



Historical note:
Von Neuman architecture

John Von Neuman, USA 1945

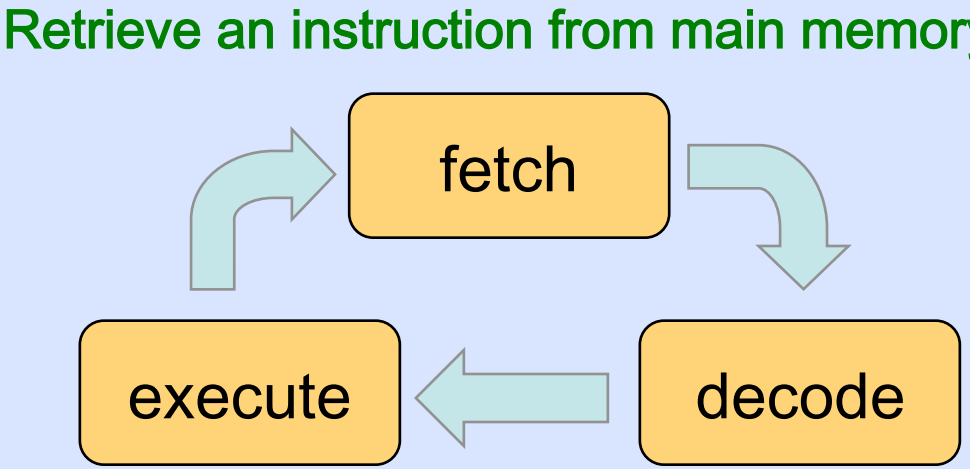
http://en.wikipedia.org/wiki/Von_Neumann

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

Synonymous with RAM

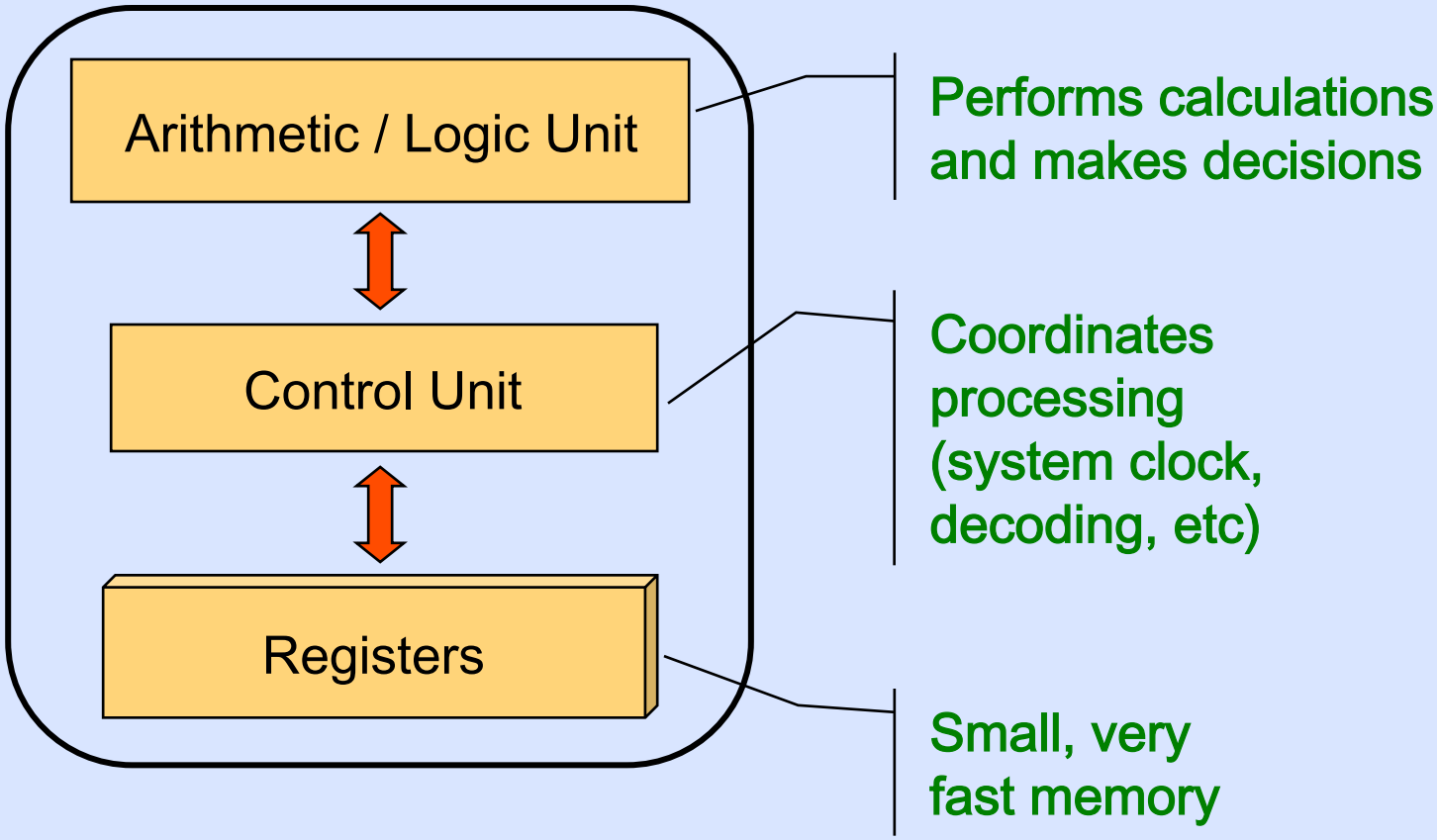
The Central Processing Unit

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



system clock *controls speed,* measured in *gigahertz* (GHz)

The Central Processing Unit



Historical Note: Automatic control of computation

A machine that can follow a series of steps - a "*program*"

- Early efforts:
  - Jacquard loom (France 1801)
  - Babbage's Difference engine and Analytical engine (England 1822)
  - Holerith's census machine (USA 1890)
- Colossus Mark I – first electronic computer to be programmable (**Alan Turing**, England 1944)
- Stored program and the fetch/decode/execute cycle (John von Neumann, USA 1945)
- ENIAC - first fully electronic digital computer (Eckert and Mauchley, University of Pennsylvania, 1946)

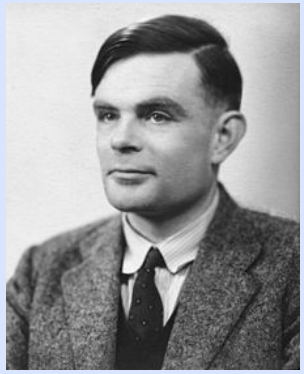


Image: http://www.biography.com/people/alan-turing-9512017

Historical Note: Automatic control of computation

Jacquard Loom

This portrait of Jacquard was woven in silk on a Jacquard loom using 24,000 punched cards (1839).

Charles Babbage owned one of these portraits; it inspired him in using punched cards in his analytical engine. Collection of the Science Museum in London, England.
(Source: Wikipedia)

punched cards determine the pattern

Historical Note: Automatic control of computation

Charles Babbage & Ada Lovelace

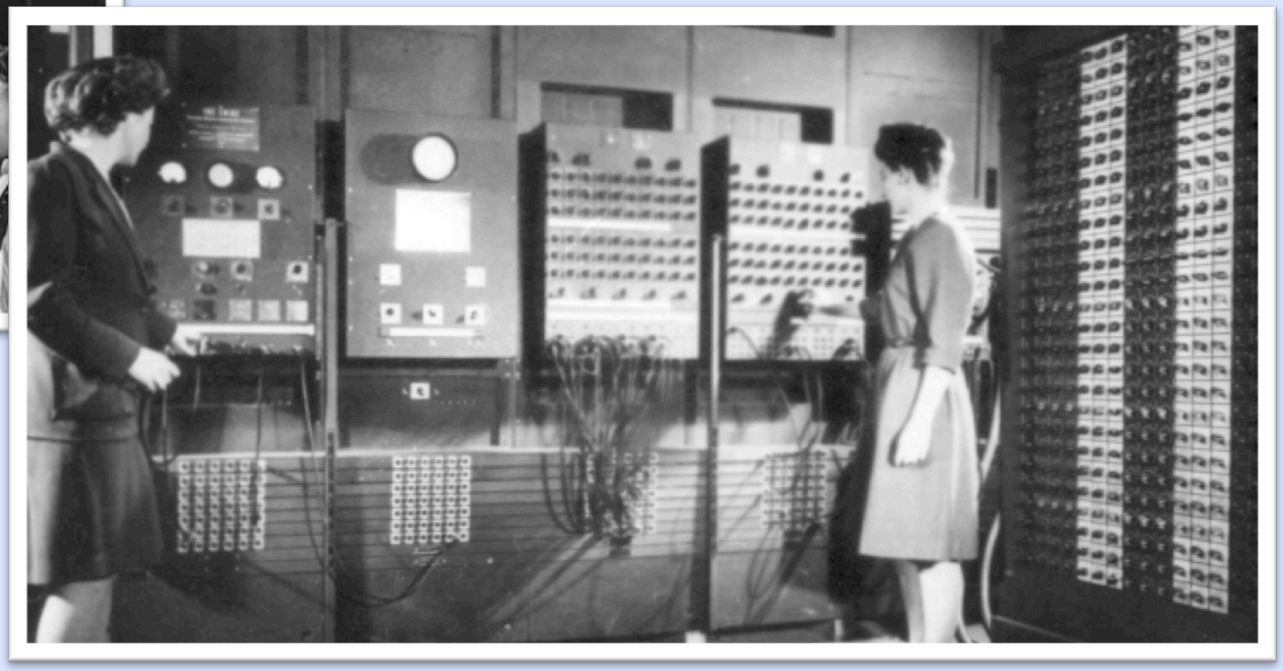
Designed the Analytical Engine

First “Programmer” for (not yet built) Analytical Engine

Historical Note: Automatic control of computation

1945: The word “computer” changes its meaning

Captain Grace Hopper
and other computers

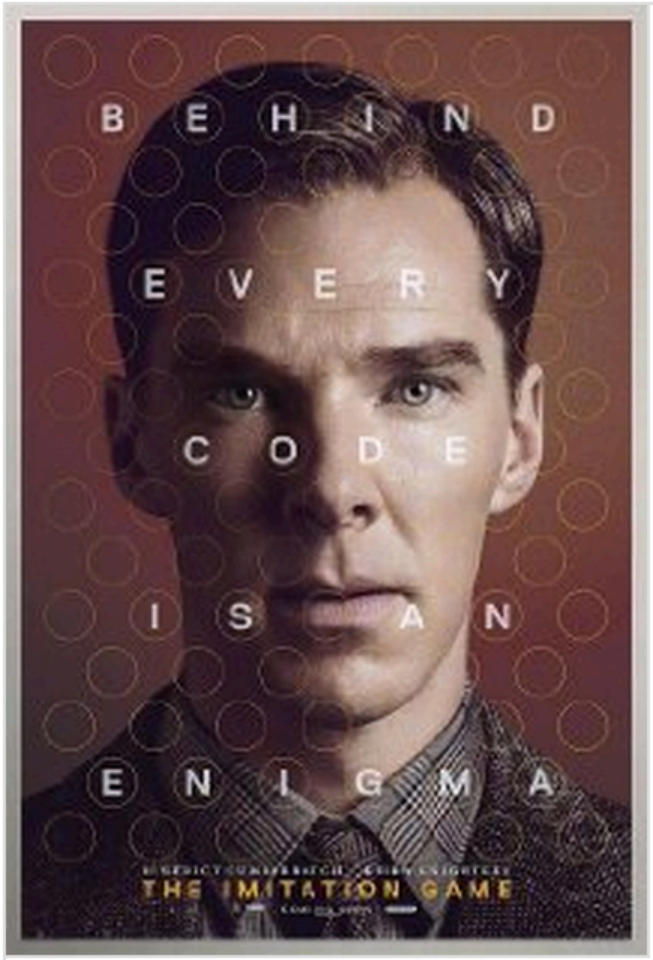


The Electronic Numeric Integrator and Calculator (ENIAC)

Programmers Betty Jean Jennings (left) and Fran Bilas (right) operate ENIAC's main control panel at the Moore School of Electrical Engineering. (U.S. Army photo from the archives of the ARL Technical Library)

Historical Note: Automatic control of computation

2014: Benedict Cumberbatch shows the world how cool Alan Turing was



The Imitation Game (2014)

Top 5(

PG-13 | 114 min | Biography, Drama, Thriller |
25 December 2014 (USA)

8.1

Your rating: -/10

Ratings: **8.1**/10 from 332,068 users Metascore: 73/100

Reviews: 555 user | 423 critic | 49 from Metacritic.com

During World War II, mathematician Alan Turing tries to crack the enigma code with help from fellow mathematicians.

Director: Morten Tyldum

Writers: Graham Moore, Andrew Hodges (book)

Stars: Benedict Cumberbatch, Keira Knightley, Matthew Goode | See full cast and crew »

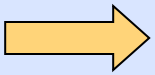
http://www.imdb.com/title/tt2084970/

Reverse History of computing

Examine what we already know, travel backwards…

1. What we see now all around us – a connected world of computing

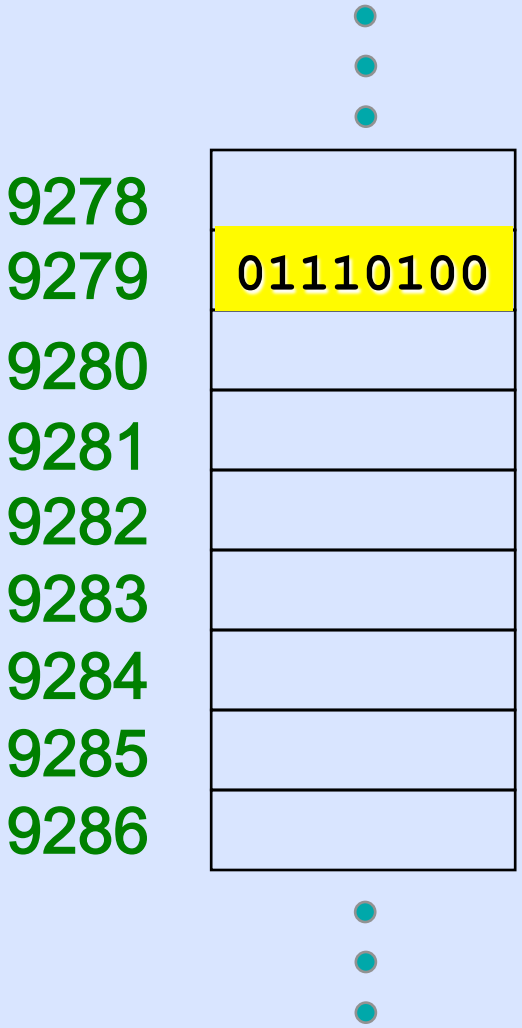
2. Focus on a single “traditional” computer



3. Dig deeper – data and processing

Data Representation

- Computers store all information ***digitally***, using ***binary*** codes:
  - numbers
  - text
  - images
  - audio
  - video
  - program instructions



A ***byte*** is a group of eight bits

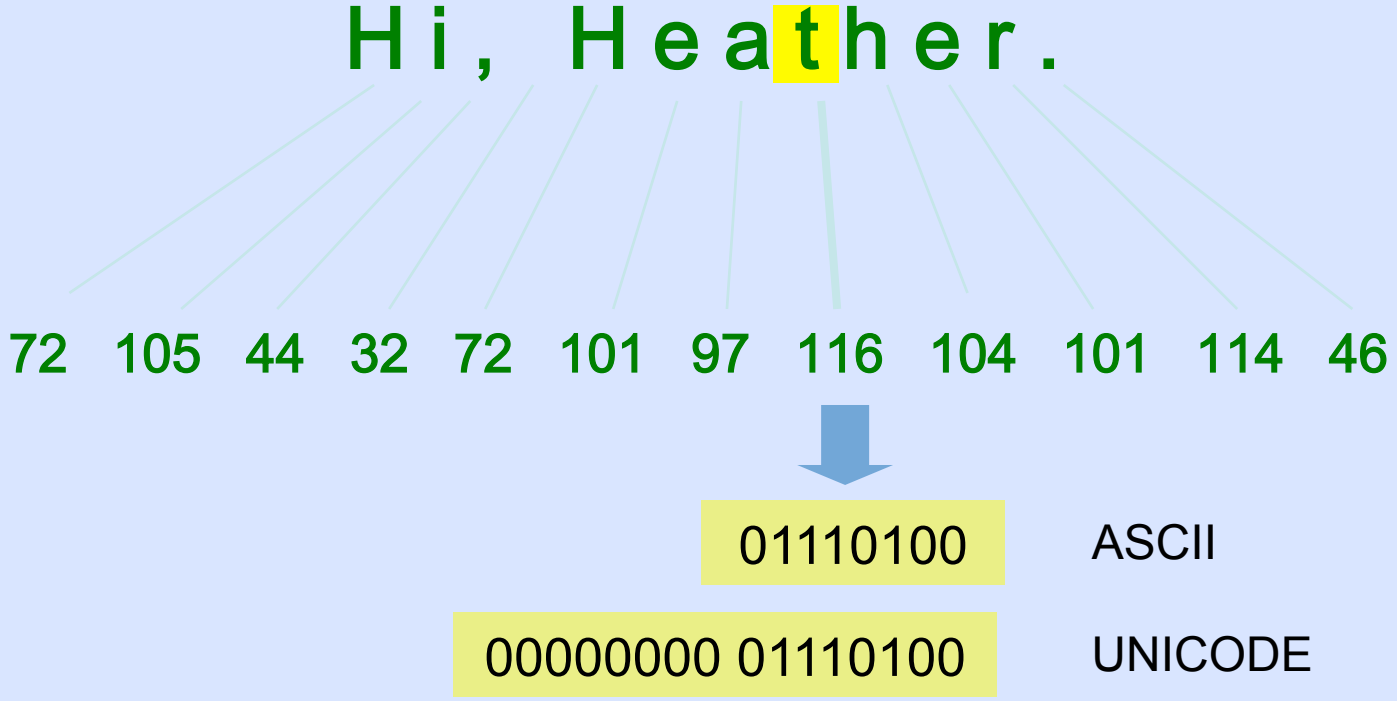
01110100

- a number?
- a letter?
- the red component of a pixel?
- a program instruction?

Computing devices store use binary codes to reprepresent data of ***all kinds***

Example: Representing Text

- Characters, including spaces, digits, and punctuation are represented by numeric codes



The **ASCII** (American Standard Code for Information Interchange) character set uses eight bits per character, allowing for 256 unique characters

The **Unicode** character set extends ASCII to sixteen bits per character, allowing for 65,536 unique characters.

Example: Representing Pixels

Example: Representing Program Instructions

Intel opcode for the instruction **JZ** (jump if zero):

```
01110100
```

Historical Note:
Symbolic Representation & Mechanization of Arithmetic

- Development of number systems & geometry
- The notion of an algorithm
- Creation of special purpose calculators

Historical Note:
Symbolic Representation & Mechanization of Arithmetic

Basic human needs: **counting & measuring**

http://www.dreamstime.com/royalty-free-stock-photography-counting-sheep-image129737

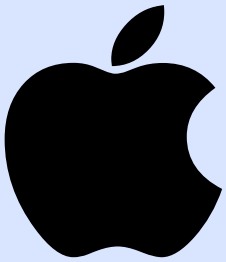
http://ghoststudy.com/new11_galleries/halloweve1067.jpg

Historical Note:
Symbolic Representation & Mechanization of Arithmetic

Basic human needs: **Symbolism**

:-)

π



∞

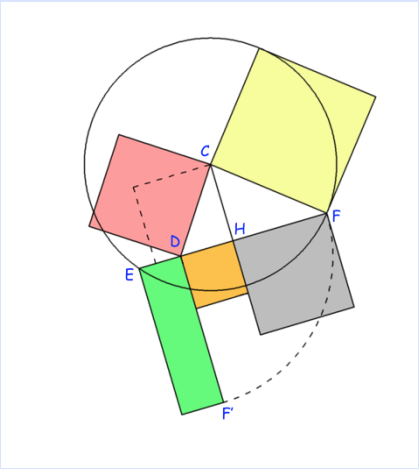
<3

Historical Note:
Symbolic Representation & Mechanization of Arithmetic

- **Development of number systems & geometry**
  - Abacus (China ~2400 BC)

  - Number systems (Babylonian, Greek, Roman, Arabic 1000 BC - 800 AD)

  - Geometry (Egypt/Greece 300 BC)



Historical Note:
Symbolic Representation & Mechanization of Arithmetic

- **The notion of an algorithm**

Euclid (300 BC)

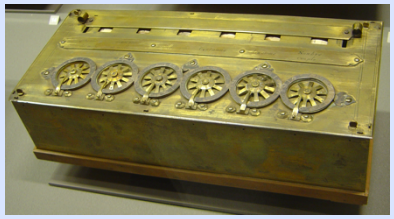
Muhammad ibn Musa al-Khwarizmi (800 AD)

images:http://www.thefamouspeople.com/profiles/euclid-436.php
https://en.wikipedia.org/wiki/Muhammad_ibn_Musa_al-Khwarizmi

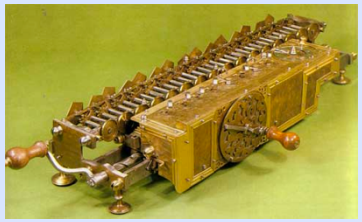
Historical Note:
Symbolic Representation & Mechanization of Arithmetic

- **Creation of special purpose calculators**

Stonehenge (1900-1600 BC)



Pascal's adder (1642)



Leibniz's calculator (1670s)

1975 Texas Instruments calculator

images:
https://upload.wikimedia.org/wikipedia/commons/8/80/Arts_et_Metiers_Pascaline_dsc03869.jpg
http://www.livescience.com/22427-stonehenge-facts.html
http://history-computer.com/MechanicalCalculators/Pioneers/Lebniz.html

Historical notes:

Trends that gave rise to the modern computer

- ***Symbolic representation and the mechanization of arithmetic*** – the concepts of numbers, symbols, algorithms, and computation

  +

- ***Automatic control of computation*** – a "program" to control operations (fetch/decode/execute cycle and the stored program concept)

  +

- ***Connecting the world*** – networks and telecommunications

= modern computer

Welcome to Computer Science!

A new paradigm in humanity's search for understanding of:

- Symbolic representation
- Computation
- Problem solving
- Mechanization

History Epilogue: Just like Physics and other sciences branched off from philosophy during the Renaissance, so CS emerged in the 20th century from the work of philosophers and mathematicians – with the help of dedicated, visionary practitioners, experimental scientists and engineers.

Summary

- An introduction to Java:
  - Identifiers
  - Comments
  - Errors
  - Strings and printing
- History of computing
- Computer hardware and software overview