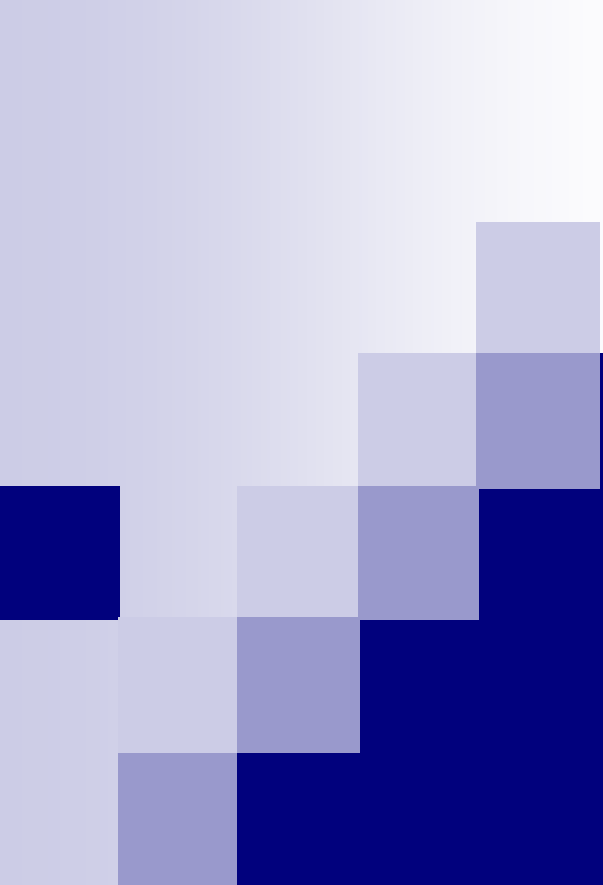




Review for CIS 1.0 Final

The Topics of CIS 1.0

- Computer system – hardware and software
- Machine architecture
 - How hardware components being organized
 - How computer program and data is being stored in memory
 - How computer instruction drives the behaviors of CPU, Memories, I/O (not in the course)
- Data representation – digitalize “everything”: numbers, characters, images, sounds, etc.
- Algorithms and computer programming languages (how software is built)
 - Algorithms - Human’s idea in terms of steps to drive a computer
 - Computer languages – The tool to write down human’s idea of steps to computer
- Limit of computer: problems with unsolvability and nonfeasibility; the halting problem
- HTML and JavaScript
 - JavaScript – An example of computer programming language
 - Variables, expressions, functions, if, loop (while, for)
 - HTML – a markup language to define webpages
- Networking – connecting the computers: packet switching, and layering architecture
- Security and privacy, encryption (secret key encryption and public key encryption)
- Some history of computer science, and practice on information searching



Computer System: Hardware and Software

An overview of computer system

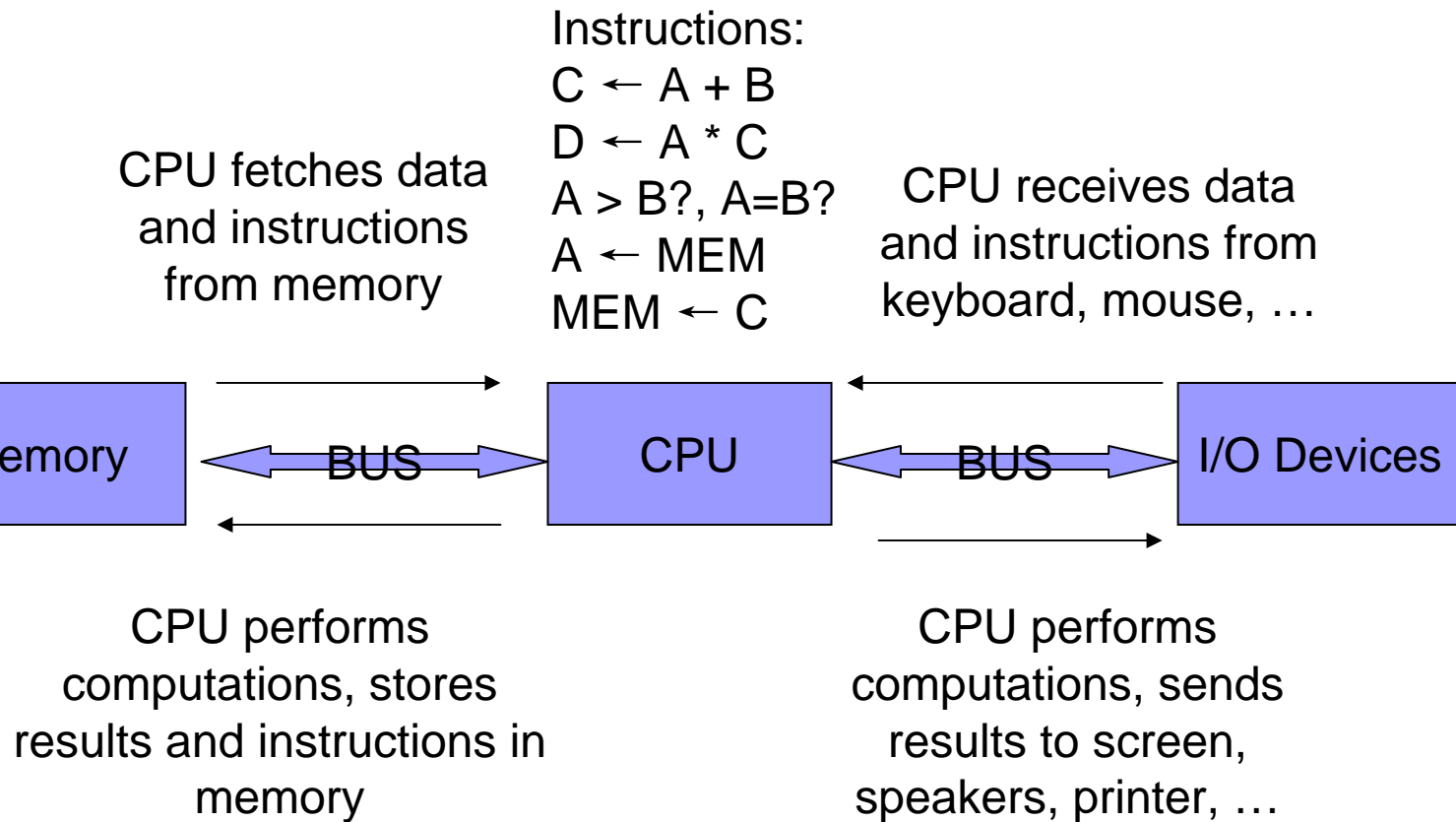
■ Hardware

- Central Processing Unit (CPU, or simply called processor)
- Memory
 - Internal memory (volatile)
 - CPU registers
 - CACHE
 - RAM (Random Access Memory): computer's main memory (or primary memory)
 - External memory (permanent), also called secondary memory
 - Hard drive
 - Floppy
 - Diskette
- Input / Output devices: keyboard, mouse, monitor, printer, etc.

■ Software

- Program
- Operating system
 - Manage the CPU and RAM allocation
 - File system: manage the secondary memory, directory, file
 - I/O devices management
 - Human-machine interface: GUI (Graphical User Interface)
- Application program: A computer program run on a computer directly targeting a task that the user wishes to perform

The von Neumann Architecture





Hardware and Software

- Hardware – Physical components of computer
 - E.g. CPU, RAM, keyboard, monitors, printers, speakers, etc.
- Software – Programs (series of computer instructions) that execute on computer, and the associated data
 - E.g. Microsoft windows, Internet Explorer, Netscape, Microsoft Word, etc.

CPU

- CPU is the brain of the computer, responsible for controlling the **internal workings** of the machine.
 - CPU is made of circuitry
 - E.g. A 1.8GHz CPU can execute approximately 1.8 billion simple instructions in a second
- Two tasks:
 - Fetching program instructions from memory
 - Executing the fetched instructions
- CPU is also called processor.

Memory

- **Memory** is the part of a computer that stores programs and data.
- Digital computer stores and processes information as **binary digits**, or **bits**.
- **Cache** is the memory which is built into the CPU chip; it utilizes high-speed-circuitry to provide extremely fast access to data.
- **RAM** (short for Random Access Memory) is the memory which is packaged on separate chips, communicates with CPU using lower-speed circuitry.
- **Main memory** is composed of Cache and RAM.
- Main memory is **volatile**, meaning that it requires a constant flow of electricity to maintain its stored values. When the computer is turned off the values stored in the main memory will be lost.
- **Secondary memory**
 - A **hard disk** is a metal platter that stores bits as magnetized and interprets them as bits. A hard disk is capable of permanently storing vast amounts of information (usually measured in gigabytes), which can be transferred into main memory when needed.
 - Other memory
 - Floppy disk
 - CD, etc.

Input/Output devices

- **Input devices** allow the computer to receive data and instructions from an external source, such as a person entering commands at a keyboard.
 - E.g. Keyboards, mice, track pads, microphones, scanners, etc.
- **Output devices** allow the computer to display or broadcast its result.
 - E.g. monitors, printers, speakers.

Software

- **A software program** is a collection of instructions for computer to carry out in order to complete some task.
- **Application software** is the software program which is designed to carry out tasks within a particular application area, such as word processing, graphical design, or Web access, etc.
- **Systems software** is the software program which is designed to manage the resources and behavior of the computer itself.
- **Operating system** is a systems software, a collection of programs, which controls how the CPU, memory, and I/O devices work together to execute programs.



Machine Architecture

Machine architecture

- Switches using transistors
- CPU
 - Control Unit - control the functioning of the other CPU components
 - IR
 - PC
 - Registers
 - Register address
 - ALU
 - Buses - Connections between registers, ALU, and memory interface
- Memory
 - Memory address
- Buses - Connection between memory and CPU
- The concept of stored program
 - Instructions are stored in memory
 - CPU's control unit loads instructions into CPU and executes them one by one

The Architecture

- CPU

- ☐ ALU
- ☐ Control Unit
 - PC
 - IR
 - MM – memory address
- ☐ Registers
- ☐ Buses connecting ALU and registers

- Main memory

- ☐ Memory cell
- ☐ Address

- Buses connecting CPU and Main memory

ALU

- The **arithmetic logic unit (ALU)** is the collection of circuitry that performs actual operations on data.
- **Basic operations** include
 - Addition
 - Subtraction
 - Bit manipulation (such as shifting or combining bits)



Registers

- **Registers** are memory locations that are built into the CPU.
- Data in registers can be accessed more quickly than the data in memory (as much as 5-10 times faster).
- Limited number of registers in CPU due to the cost (commonly 16 or 32 registers).



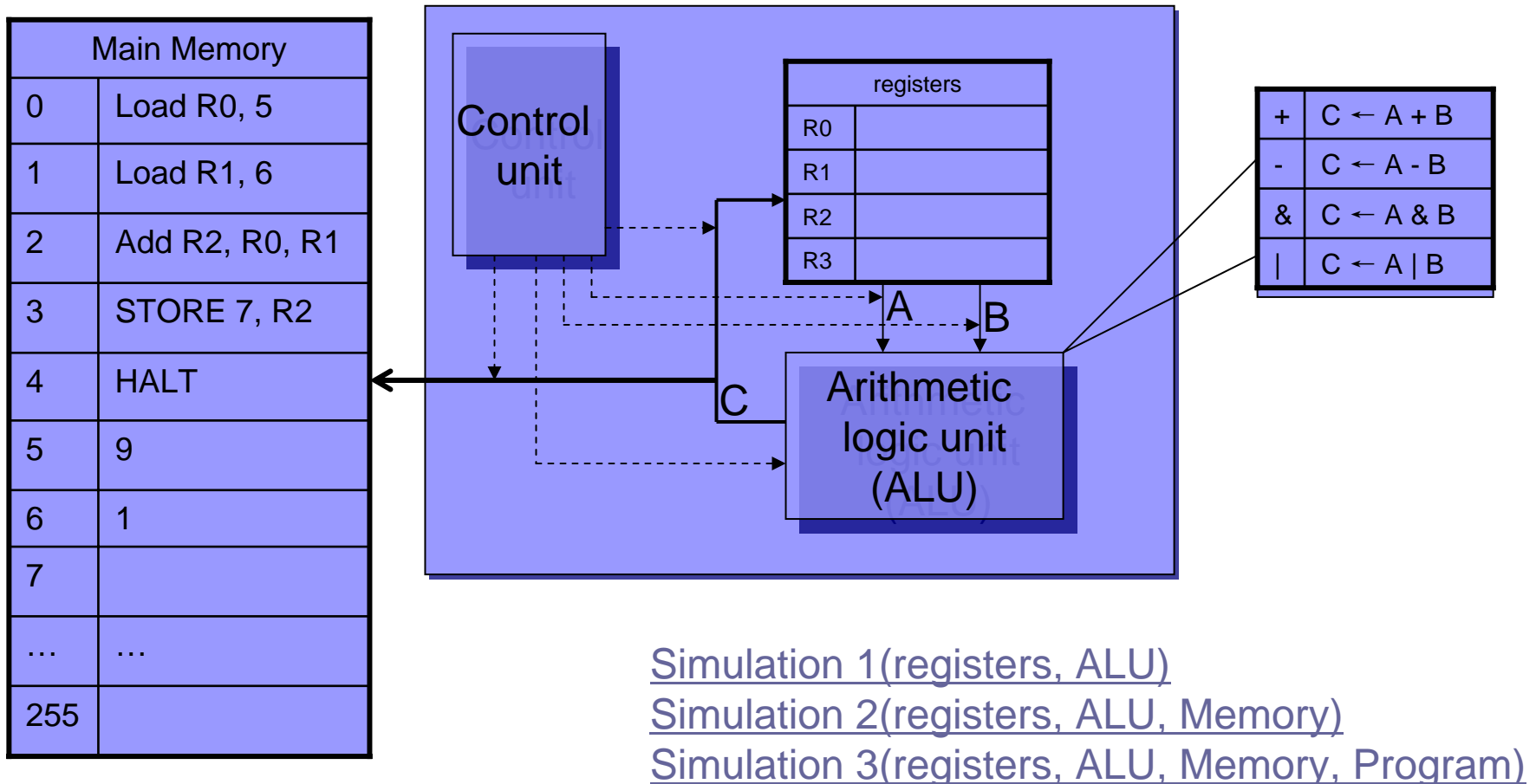
Buses

- **Bus** is a collection of **wires** which are responsible for transferring data between computer components.
- A set of buses inside CPU connect the registers to ALU.
- A bus Between CPU and memory connect the memory to CPU.

Control Unit (CU)

- The **control unit (CU)** can be thought as “the brain within the brain”, in that it oversees the various functions of the CPU.
- The control unit is a set of circuitry.
- Control unit is in charge of
 - Fetching **data** from main memory to CPU
 - Fetching **instructions** from main memory to CPU
 - Controlling the **flow of data** from **registers to the ALU** as well as from **ALU to registers**
 - Controlling the **operations of ALU**
 - Storing **data** from CPU to main memory
 - Storing **instructions** from CPU to main memory
- Basic controlling means are setting the **switches**.

CPU and Main memory



More Details about Control Unit

- **Program Counter (PC)** contains the memory address of next instruction to be executed.
- **Instruction register (IR)** contains the instruction that the control unit is currently executing.
- Configuration of switches for
 - ALU
 - ALU-register bus
 - Memory-CPU bus
 - Addresses of selected registers
 - Address of selected memory

Main Memory

- We can think of main memory as a large collection of **memory locations (cells)**.
- Each location is labeled by an address (binary number).
- Each location can be accessed by given its address.
- A **bus** connects main memory to CPU.
- A little bit details
 - A memory address (a sequence of 0s and 1s) will activate a **set of switches** which **select** a specified memory location.
 - The switches connect this specified memory location to **an interface** connecting to the Memory-CPU bus.

Instruction Cycle

- Fetch the instruction from main memory whose address is in the PC (program counter).
 - Store the instruction in IR (instruction register)
 - Increase the instruction address in PC by 1
- The control unit (CU) decodes the instruction and figure out the configuration of the switches to select the registers, buses, memory, etc.
- The control unit (CU) executes the instructions by activating the switches following the configuration given figured out above.
- A little bit more details: Some instruction (e.g. Jump xxx) is just intended to modify the content of PC, so as to modify the execution path of the program.



Speed of Today's Computer

- Millions or billions of instructions are being executed in one second in modern computer



Data Representation

Memory organization

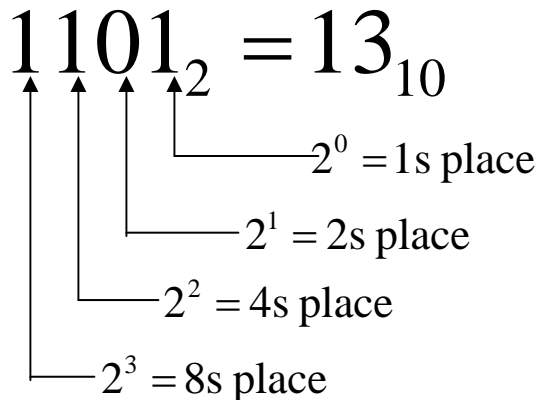
- **Bit** – Units of data that correspond to one of the two potential values: **0** and **1**.
 - Byte – a collection of 8 bits.
 - KB = 1024 bytes
 - MB = 1024 KBs
 - GB = 1024 MBs
- A **file** is a document that stores information, such as text (e.g., a term paper), and image, sound, or a program (Internet Explorer). The operating system keep track of where individual files are stored so that they can accessed when needed.
- A **directory**, or a **folder**, is a collection of files that are organized together and labeled with a common name.

Data representation

- A **binary** numeral system is a numeral system in which all values are represented using only **two binary digits**, **0** and **1**; these digits are called **bits**.
- **ASCII** is the standard code for representing characters; it maps each character to a specific 8-bit pattern.
- A document that contains only plain text (such as notepad file, html file) is called **ASCII** file or a **text** file.
- Size of ASCII file = number of characters stored in the file
- **Unicode** is a 16-bit encoding system capable of supporting most foreign-language character sets.

Binary numbers to decimal numbers

$$13_{10} = 1 \times 10^1 + 3 \times 10^0$$

$$1101_2 = 13_{10}$$


$2^0 = 1\text{s place}$
 $2^1 = 2\text{s place}$
 $2^2 = 4\text{s place}$
 $2^3 = 8\text{s place}$

$$\begin{aligned} 1101_2 &= 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\ &= (8 + 4 + 0 + 1)_{10} \\ &= 13_{10} \end{aligned}$$

- Multiply each bit (either 0 or 1) with the corresponding power of 2 and then sum the results.
- The corresponding power of the right most bit is 0, then 1, 2, 3,...

Representing Integers as bit patterns

- Integers are stored as bit patterns
 - 19 as 10011
 - 116 as 1110100
- Fixed-width
 - 32-bit width: 19 as 0000 0000 0000 0000 0000 0000 0001 0011 (with many leading 0s)
- Number of representable integers = number of bit patterns
 - 32 bit width representation has $2^{32} = 4,294,967,296$ bit patterns, namely 4,294,967,296 integers

Two-complement notation of integers

- Sign bit (usually the highest bit of the representation)
 - 0 means that this is a positive integer
 - 1 means that this is a negative integer
- The representation
 - Positive integer s is represented as its corresponding binary number
 - Negative integer s is represented as the corresponding binary number of $2^{32} - |s|$
- Examples (32-bit width integer):
 - 3: 0000 0000 0000 0000 0000 0000 0000 0011
 - -1: 1111 1111 1111 1111 1111 1111 1111 1110 = $2^{32} - 1$
- Why two-complement notation?
 - Avoid to two 0s: positive zero and negative zero
 - Simplify the binary addition

Binary representation of real number

- Binary-based Floating-point number $s = (\text{fractional part } m, \text{exponent } e)$, then the value of $s = m * 2^e$
 - m is the two-complement representation of the fractional part
 - e is the two-complement representation of the exponent part
- Fixed-width
 - Single precision (32-bit): 1bit for sign, 8 bits for exponent, 23 bits for fractional part
 - Double precision (64-bit): 1bit for sign, 11 bits for exponent, 52 bits for fractional part
- Example: $1110110.101 = 1.110110101 \times 2^6$
 - m will be 11011010100000000000000 (zeros are filled at the end to have 23 bits)
 - e will be 00000110 (6 in decimal), but in the practice it is 10000101 (6 + 127) which is out of the this course's scope.
- Round-off:
 - There are infinite number of real values but only a finite number of 32- or 64-bit patterns.
 - The real value will be round off to the pattern with closest value.
- Round-off examples
 - Single-precision floating-point number: roughly 7 decimal digits of precision
 - Double-precision floating-point number: roughly 16 decimal digits of precision

Representing Characters and Strings

- Character is encoded by binary bit patterns
- ASCII (American Standard Code for Information Interchange) codes
 - ASCII maps each character to a specific 8-bit pattern
- Unicode
 - Unicode maps each character to a specific 16-bit pattern
 - For non-English language with more than 256 characters

Representing Images

- Bitmap
 - partitions an image into a grid of dots, called **pixels**
 - represents each pixel with bit pattern
- Pixel bit pattern
 - 1bit Black-and-white image
 - 1 for white
 - 0 for black
 - 24-bit color image (R, G, B)
 - R: 0-255 intensity of red
 - G: 0-255 intensity of green
 - B: 0-255 intensity of blue
- Resolution: The sharpness or clarity of an image
 - E.g. It can be measured with how many pixels per square inch in an image

Image formats: BMP, GIF, JPEG, etc.

- BMP (bitmaps)
 - Simple
 - But require lot of memory
- GIF (Graphics Interchange Format) – a lossless format
 - Identify repetitive patterns in the image
 - Store those patterns only once with their markers
 - Replace the occurrence of such patterns with their markers
- JPEG (Joint Photographic Experts Group) – a lossy format
 - The compression is not fully reversible: e.g. compress several neighboring pixels by storing their average color value
- Usages
 - GIF format is used for line drawings and other images with discrete boundaries
 - JPEG format is mostly used for photographs

Representing Sound

- Each particular sound produces a pressure wave with
 - **Unique** amplitude (height, usually measured in pascals)
 - **Unique** frequency (duration over time)
- Digital sampling
 - Measure amplitudes of the sound wave at **regular intervals**
 - Store the amplitudes as sequence of discrete measurements in binary numbers
- CD-quality sound
 - 44,100 amplitude measurements per second
 - 16-bit representation of the amplitude

ASCII file (text file)

- A document that contains only plain text (such as notepad file, html file) is called **ASCII** file or a **text** file.
- The size of a file = number of bytes stored in the file
- Size of ASCII file = number of characters stored in the file

Binary file


- Files that contain data that is not plain text (e.g. image, sound, word document, etc.) are called binary files.
- The size of a file = number of bytes stored in the file
- Examples
 - An ASCII file with 1000 words is of size roughly 5,000bytes = **5KB** assuming average 5 characters per word.
 - Most graphics on web are over **30KB**.

Storage devices

- Floppy disks (1.44MB, 1.2MB, etc.):
http://en.wikipedia.org/wiki/Floppy_disk
- Zip drives (100MB, 250MB, 750MB):
http://en.wikipedia.org/wiki/Zip_drive
- Jaz drives (1GB, 2GB):
http://en.wikipedia.org/wiki/Jaz_drive
- Hard drives (0 – 500GB, or more):
http://en.wikipedia.org/wiki/Hard_disk
- CDs (650MB): http://en.wikipedia.org/wiki/Compact_disc
- DVDs(4.7GB): <http://en.wikipedia.org/wiki/DVD>



Computer Languages and Algorithmic Thinking



Machine language and high level language

- **Machine language** is a collection of different patterns of binary bits that correspond to different computer machine instructions.
- **High-level programming languages** are the languages that provide high-level abstractions and constructs (which correspond to machine level operations) for solving problems using computers.

Algorithm and Computer Program

- An **algorithm** is a series of steps that can be followed to solve the problem.
- A **program** is a series of instructions that specify exactly what the computer is supposed to do. Instructions can be
 - in high level programming languages
 - Or in machine language (binary coded instructions)
- **Stored program scheme:** The programs in machine language are stored in computer memory, the CPU fetches and executes the instructions in the program one by one.

Compiler and Interpreter

- An **interpreter** reads the statements in high-level language one at a time, immediately translating and executing each state before processing the next statement.
- A **compiler** translates the entire high-level language program into its equivalent machine-language instructions.

Software life-cycle

- Analyze the problem
- Specify the problem strictly
- Devise algorithm (and software architecture: how to organize the data, and organize the program.)
- Coding: implement the algorithm in some computer language(s)
- Testing
- (Documentation, and software training and support)
- Maintenance



JavaScript

JavaScript

- Building blocks
 - Variables (name, value)
 - Expressions (=, +, -, *, /, function call, if, while, for, etc.)
 - Functions (predefined, user-defined)
- Connecting to html Web-page
 - Objects, properties
 - Objects: window (status, location), document (bgColor, fgColor), image (src), link, form (text input object, button input object)
 - Web-page functions
 - alert, confirm, prompt, document.write, etc.
 - Event, event-driven programming
 - Event producers: link, button, etc.
 - Events and event handlers: onMouseOver, onMouseOut, onClick, etc.

Variable

- **Variable** is a “container” to store the information you want to store.
- Variable value: The content stored in the “container”; it can change during the execution.
- Variable name: A name to refer the information in the “container”. Naming rules:
 - Variable names are **case-sensitive**.
 - They must begin with letters (a-z, A-Z) or underscore character(_).

Data types

- The types of information that can be stored in variables are called data types.
- **Numbers**
 - Integers: positive, 0, or negative.
 - Representation in JS: As in math
 - Floating-point numbers
 - Representation in JS
 - With decimal point: 314.15
 - With scientific notation: 3.1415e2
- **Booleans: true or false** (the case does matter!)
- **Strings:** Strings are delineated by single (') or double quotation (") marks. (Use single quotes to type strings that contain quotation marks.)
 - E.g. "This is course CIS 1.0"
- Objects
- Null
- Undefined

Expressions

- Each JavaScript data type is associated with a specific set of predefined operators.
- Strings can be concatenated using the + operator.
 - E.g `str = "first half" + "second half"`
- Numbers have a predefined standard arithmetic operators +(addition), -(subtraction), *(multiplication), and / (division)
 - E.g `t = 10 + 4/2 + 3 * 5`
- An ***expression*** is any valid set of literals, variables, operators, and expressions that evaluates to a single value; the value can be a number, a string, or a logical value.

JavaScript Function

- A JavaScript function is an abstraction of sequence of instructions for a certain task.
- Parameters – Input of the task
- Local variables – Local memory to perform the task
- Return statement – Return the result of task back to the calling program (which will be accepted into the calling program's memory).
- Note: Some functions perform Input/Output tasks; the result of these functions are not only reflected by the return statement and memory, but also reflected in the reality (display in monitor, print in printer, input of keyboard, etc.)

Advantages of using functions

- Help minimize the amount of the detail that the programmers must keep track of. You only need to care about
 - Function name
 - Input of the function
 - Output of the function (result)
- Help minimize the length and complexity of the code.
 - The lengthy sequence of instructions for a task are organized as functions.
 - Every time the task is needed, you just need to call the function without repeating the instructions again and again.

JavaScript predefined functions

- JavaScript's predefined functions represent a collection of useful, general-purpose abstractions of tasks.
- Input/Output: `prompt`, `confirm`, `alert`
- Math functions: `Math.sqrt`

Syntax of User-defined Functions

```
function FUNCTION_NAME(PARAMETER_1, PARAMETER_2, ...,  
    PARAMETER_n)  
{  
    STATEMENT1;  
    STATEMENT2;  
    ....  
    STATEMENTm;  
    return VARIABLE_NAME;  
}
```

- Note: parameters and return statement are optional; you can define a function without parameters as well as a function without return statement.

Syntax

- The **syntax of JavaScript** is a set of rules that defines how a JavaScript program will be written and interpreted
 - Rules for variable names
 - Rules for valid data types
 - Rules for valid instructions
 - More...



Important Issues

- The language instructions must be written in **lower case**
- All the instructions must **be spelled correctly and exactly**
- Parts of an instruction need to be **separated by space** and not run together
- The correct **punctuations** are required

Structure of JavaScript

- Instructions are separated by semi-colons or line-breaks

```
<html>
  <head>
    <title> An JavaScript example </title>
    <script>
      name = prompt("Please enter your name");
      document.write("Hi "+ name);
    </script>
  </head>
  <body>
  </body>
</html>
```

The Places to Put JavaScript

■ Inside <head>...</head>

- Illustration: `<head>... <script>...</script>... </head>`
- The **functions** and **variables** defined in `<script>...</script>` inside **head** are guaranteed to be **established** before the execution of any JavaScript codes in `<body>...</body>`.
- The Javascript codes inside **head** are guaranteed to be executed before any codes in `<body>...</body>`.

■ Inside <body>...</body>

- Illustration: `<body>... <script>...</script>... </body>`

About Java Script

- Interpreted high level programming language
- Purpose
 - Dynamic changes to the webpage
 - Real time changes to the webpage
- History
 - Netscape with Sun Microsystems developed it as a web programming language
 - Since Netscape Navigator 2.0
 - Since Microsoft Internet Explorer 3.0
- Characteristics of the java script
 - **Allows interactive content on webpage**
 - **Client-based: work on the browser-side not the server-side**
 - No manipulation of files and directories
 - Does not carry out graphics

JavaScript Objects

- An object is a collection of **properties** and **methods** to access and modify the properties
- An example: html **document** in JavaScript
 - Properties
 - document.fgColor
 - document.bgColor
 - Etc.
 - Methods
 - document.write
 - Etc.

String concatenation for HTML element generation

■ Italics text:

- `variable_for_text = "DYNAMIC TEXT CONTENT";`
- `document.write("<i>" + variable_for_text + "</i>");`

■ General scheme

- `variable_for_the_dynamic_content = ...`
- `document.write("<tag>" + variable_for_the_dynamic_content + "</tag>");`

■ More general

- You can use string concatenation to produce any string.
- If the string is a valid piece of html elements (e.g. "``"), you can output the string for you intended html effects using **`document.write(string)`**.

window.prompt

- Function: Displays a Prompt dialog box with a message and an input field.
- **Syntax:** [userInput]=prompt(*message*, [*inputDefault*])
- **Parameters**
 - *message* is any string or a property of an existing object; the string is displayed as the message.
 - *inputDefault* is a string, integer, or property of an existing object that represents the default value of the input field. **InputDefault is optional; it can be omitted.**
- **Return value**
 - userInput is the string which the user inputs on the dialogue box.
- **Example**
 - **document.prompt(“Please enter a year”, “2006”);**

document.write

- **Function:** Writes one or more HTML expressions to a document in the specified window.
- **Syntax :** `document.write(expression1 [,expression2], ...[,expressionN])`
- **Parameters**
 - *expression1* through *expressionN* are any JavaScript expressions or the properties of existing objects.
- **Example:**
 - `document.write("This is a message produced by write method");`

window.alert(string)

- Function: Displays an Alert dialog box with a message and an **OK** button.
- **Syntax**
 - alert(*message*)
- **Parameters**
 - *message* is any string or a property of an existing object.
- Example: alert("This is an alert message");

document.bgColor

- A property of document: A string specifying the color of the document background.
- **Syntax**
 - document.bgColor
- Example: document.bgColor = “red”
 - This instruction will set the document background color to be red.

window.confirm

- Function: Displays a Confirm dialog box with the specified message and **OK** and **Cancel** buttons.
- **Syntax**
 - [userResponse]=confirm(*"message"*)
- **Parameters**
 - *message* is any string or a property of an existing object.
- **Return value**
 - userResponse, which is optional, will be true if the user press OK button, and will be false if the user press Cancel button.
- Example: confirm("Please confirm this message");

window.open

- Function: Opens a new web browser window.

- **Syntax**

- `[windowVar =][window].open("URL", "windowName", ["windowFeatures"])`

- **Parameters**

- *windowVar* is the name of a new window. Use this variable when referring to a window's properties, methods, and containership.
 - *URL* specifies the URL to open in the new window. See the [location](#) object for a description of the URL components.
 - *windowName* is the window name to use in the TARGET attribute of a FORM or <A> tag. *windowName* can contain only alphanumeric or underscore (_) characters.
- Example: `window.open(http://www.gc.cuny.edu, "aNewWindow");`

window.close

- Function: Closes the specified window.
- **Syntax**
 - *windowReference.close()*
- **Parameters**
 - *windowReference* is a valid way of referring to a window, as described in the [window object](#).
- Example: `window.close()`
 - Close the current window; `window` is a `windowReference` to the current active window.

window.moveBy

- Function: Moves the window by the specified horizontal and vertical offsets.
- Syntax: `window.moveBy(param1, param2)`
- Parameters:
 - param1: the horizontal offset in pixels.
 - param2: the vertical offset in pixels.

HTML form

- A **form** is a grouping of buttons and other event-driven elements within a page, delimited by the tags `<form name="FORM_NAME"> ... </form>`

```
<html>
```

```
<head>
```

```
</head>
```

```
<body>
```

```
  <form name="MyForm1">
```

```
    <input type="button" value="Click Here" />
```

```
    <input type="text" name="temperature" value="110" />
```

```
  </form>
```

```
</body>
```

```
</html>
```

Button and its event handler

```
<input type="button" value="BUTTON_LABEL" onClick =  
    "JAVASCRIPT_CODE" />
```

- **type = "button"** specifies to the browser that this is a button element.
- **value="BUTTON_LABEL"** specifies the label to display on the button face; BUTTON_LABEL should be substituted by any text you want to display on the button face.
- **onClick = "JAVASCRIPT_CODE"** specifies the JavaScript code to be executed when the button is clicked by a user.
- E.g. `<input type="button" value="Click me for today's temperature" onClick = "alert('93° F ');"/>`



Click me for today's temperature

Text object

`<INPUT TYPE="text" NAME="textName" VALUE="textValue" />`

- TYPE = "TEXT" specifies that this is text input element of a form.
- NAME="*textName*" specifies the name of the *Text* object. You can access this text object in JavaScript by
 - Document.formName.textName where formName is the name of a form which contains this text object.
- VALUE="*textValue*" specifies the initial value of the *Text* object. The value of the text object can be changed by user or by javascript. You can access the value of this text object in JavaScript by
 - Document.formName.textName.value
- E.g. `<INPUT TYPE="text" NAME="temperature" value="95° F" />`

95° F

HTML images with JavaScript

``

- An image is **an object** of document's images in JavaScript.
- `NAME="imageName"` specifies the name of the *Image* object. *imageName* should be replaced by any name you want to name your image. You can access this image in JavaScript by
 - `document.images.imageName`
- `SRC="Location"` specifies the URL of the image to be displayed in the document. You can access this value using the *src* property.
 - `document.images.imageName.src`
- Change the image
 - `document.images.imageName.src = "imageFilePath"`

Events and Event Handlers

- An **event** is a user action.
- An event handler is JavaScript code which responds to the user action.
- An example: onMouseOver

```
<A HREF="#"  
    onMouseOver = "document.bgColor = 'red'; return true"  
>  
Watch this link!  
</A>
```

Event handling using functions

```
<html>
<head>
<script>
function promptAndCalc()
{
    templnFhr = prompt("Please enter temperature in Fahrenheit");
    templnFhr = parseFloat(templnFhr);
    celius = (5/9) * (templnFhr - 32);
    alert("Temperature in celius is " + celius);
}
</script>
</head>
<body>
    <input type="button"
        value="Click me for your celius temperature"
        onClick="promptAndCalc();" />
</body>
</html>
```



The limit of computer

Unsolvability, nonfeasibility and halting problem

■ Unsolvable

□ Halting problem

- The existence of its solution will cause contradictory situations.
- Therefore no solution exist

■ Nonfeasible

□ Require too many steps

Computability

- A problem is **computable** if it is possible to write a computer program to solve it.
- A problem is **noncomputable** if it is impossible to write a computer program to solve it. Such a problem is also called **unsolvable**.



Nonfeasibility and Running Time

- **Non-feasible** – a computable problem that takes too long to solve (with the fastest algorithm).

Loop

- A repetition of a sequence of instructions
 - Loop conditional test: determine whether to execute the loop body
 - Loop body: the sequence of instructions to be repeated
- Infinite loop
 - Repeat the execution of the loop body forever
 - Loop condition test is always true

Loop at the machine level

Execution sequence: 0, 1, (2,3,4),(2,3,4)...., (2,3,4), 5

0: $x = 0$
1: $y = 0$
2: $x = x + 2$
3: $y = y + 1$
4: if $(y - 10 < 0)$ then
 jump to 2
5: halt

The task of above program is to compute 2×10 with using only the operations of additions.

- Program counter (PC) stores the **address of the instruction to be executed.**
- After an instruction has been executed, **the address in PC is increased by 1.**
- With the above settings, instructions are executed sequentially according to **their appearance ordering in the memory.**
- **Jump:** To **modify the address stored in PC**, so that change the instruction execution ordering
- **Loop:** Jump to **the address of previous executed instructions**, so that a **sequence of instructions will be executed again and again**

The Halting Problem

- Halting problem – given a **computer program** and an **input to the program**, will the **given program HALT** on the given input?
- Can we write a program to solve the halting problem? Namely, can we write a program **X** to tell whether a computer program will terminate given input **Y**?
 - **NO, WE CAN'T!**
- It is a problem **can not be solved** by computer! – It is a so-called **unsolvable problem**, or **non-computable** problem!

Why Halting Problem Unsolvable?

- Assume we can solve the halting problem, then we can write a function **will_halt**(program_X, input_Y), which
 - returns **true**, if program_X(input_Y) can terminate
 - return **false**, if program_X(input_Y) will run forever
- Some reminders about the inputs of **will_halt**
 - **program_X** is a sequence of instructions sitting in some area of the memory, essentially it is just **a sequence of 0s and 1s** just like the other data in memory.
 - **Input_Y** is the input data which program_X will work on. Since program_X is also data in memory, the setting allow the user use program_X as input to call program_X, namely the user can **set input_Y = program_X**.
- Can we have the function **will_halt**?
 - **NO!**

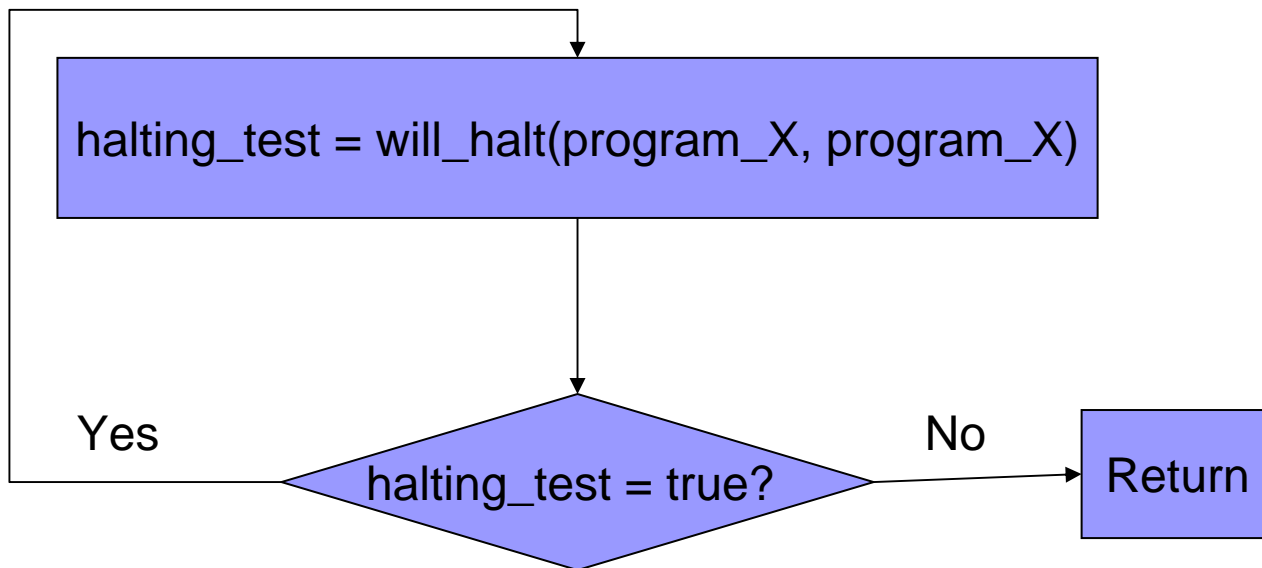
Solutions to halting problem cause contradictions!

```
function self_check(program_X)
{
    halting_test = true;
    while (halting_test == true)
    {
        halting_test = will_halt(program_X, program_X);
    }
}
```

- Contraction arises when calling **self_check(self_check)** (namely, **prograx_X=self_chck**):
 - If **self_check(self_check)** halts, then **halting_test = will_halt(self_check, self_check)** will be true forever, which means that the **self_check(self_check)** will run forever. **A contradiction!**
 - If **self_check(self_check)** doesn't halt, then **halting_test = will_halt(self_check, self_check)**, which means that the **self_check(self_check)** will terminate. **Another contradiction!**
- **This means a paradox**
 - If we can have a computer program to check whether another program halt or not, then such a halting checking program will always cause contradictory situations!
 - But contradictory situations can not happen in the real world, therefore we can not have a computer program to check whether another computer program halt or not!

Diagram of contradictory situations caused by the halting problem

function **self_check**(program_X)



- `self_check(self_check)` halts -> ("Yes" path) continue the loop -> means run forever! (**contradiction !**)
- `self_check(self_check)` runs forever -> ("No" path) stop and return -> terminate!! (**contradiction!!**)

Turing-Church Thesis

- *Thesis: "Every 'function' which would naturally be regarded as computable' can be computed by a Turing machine." (Turing machine is a model of computer.)*
- The **Church–Turing thesis** (also known as the **Church's thesis**, **Church's conjecture** and **Turing's thesis**) is a hypothesis about the **nature of computers**, such as a **digital computer** or a **human** with a pencil and paper following **a set of rules**.

Examples of feasible problems

- Adding to two integer numbers
 - several computer instructions
- Searching a list of n items
 - Need to do about n operations (to check the n items in the list one by one)
- Sorting a list of n items, one of many solutions:
 - Search for the smallest element, put it in the beginning (cost about n operations)
 - Do this n times for each i^{th} smallest element
 - Overall: about n^2 operations


Examples of non-feasible problems

- List all the possible score permutation of a class
 - Given the class size n , score are between 0-100
 - Need to do at least 100^n operations to output the result (assuming 1 operation for 1 permutation for simplicity)
 - For a class of 30 students, we at least need to do **10^{60}** operations
 - Modern computer can perform about 10^9 instructions in a second
 - There is roughly $3.15 * 10^7$ seconds per year
 - **$10^{60} / (3.15 * 10^7 * 10^9) > 3 * 10^{10}$ years!!!**
- Consider all the possible moves and outcome of chess playing
- Consider every possible seating arrangement for the class
- In general, any solution that considers every possible subset of a set requires exponential steps in term of the size of the set!

A list of steps in term of input size

Given a problem's input of size n

- Requiring n (linear) operations is feasible
- Requiring n^2 (quadratic) operations is feasible
- Requiring n^c (polynomial) operations is feasible, where c is constant in spite of n
- In general, requiring **less than n^c** operations is **feasible**
- Requiring **c^n (exponential: $2^n, 10^n, \dots$) operations** is **non-feasible** when n becomes large, because the requiring amount of operations increases rapidly as n increases.



Analysis of algorithms

- Analysis of algorithms – analyzing the running times of algorithms (or programs). This field of computer science studies algorithms and their efficiency, in terms of the amount of work (and space) needed to execute an algorithm.



Networking and Internet

LAN, WAN and Internetworking

- **LANs** (short for Local Area Network) are used to link computers over short distances, such as within the same room or building.
 - Ethernet is the most popular technology to build LANs.
- **WANs** (short for Wide Area Network) are used to connect computers over long distances, so it must include built-in controls for routing messages and adapting to the failures that will inevitably occur.
 - Internet as a whole is an example of WAN.
- **Internetworking** involves connecting two or more distinct computer networks together into an **internetwork** (often shortened to **internet**), using devices called routers to connect them together, to allow traffic to flow back and forth between them.
 - Historically, *Internet* and *internet* have had different meanings, with *internet* being a contraction of *internetwork* or *internetworking* and *Internet* referring to the worldwide network.

History of the Internet

■ ARPNET

- **Pre-birth:** J.C.R Licklider (MIT), 1960s, the “Galactic Network” idea: share computers (expensive), share and access information
- **Name obtained:** Larry Roberts’ team, 1967, finalized ARPANET plan (ARP – Advanced Research Project Agency, a U.S. Department of defense agency)
- **Became reality:** 1969, linking 4 computers at UCLA, UCSB, SRI (Stanford Research Institute), University of Utah
- Growth
 - 23 computers in 1971
 - 100 computers by 1980
 - More than 1,000 computers by 1984

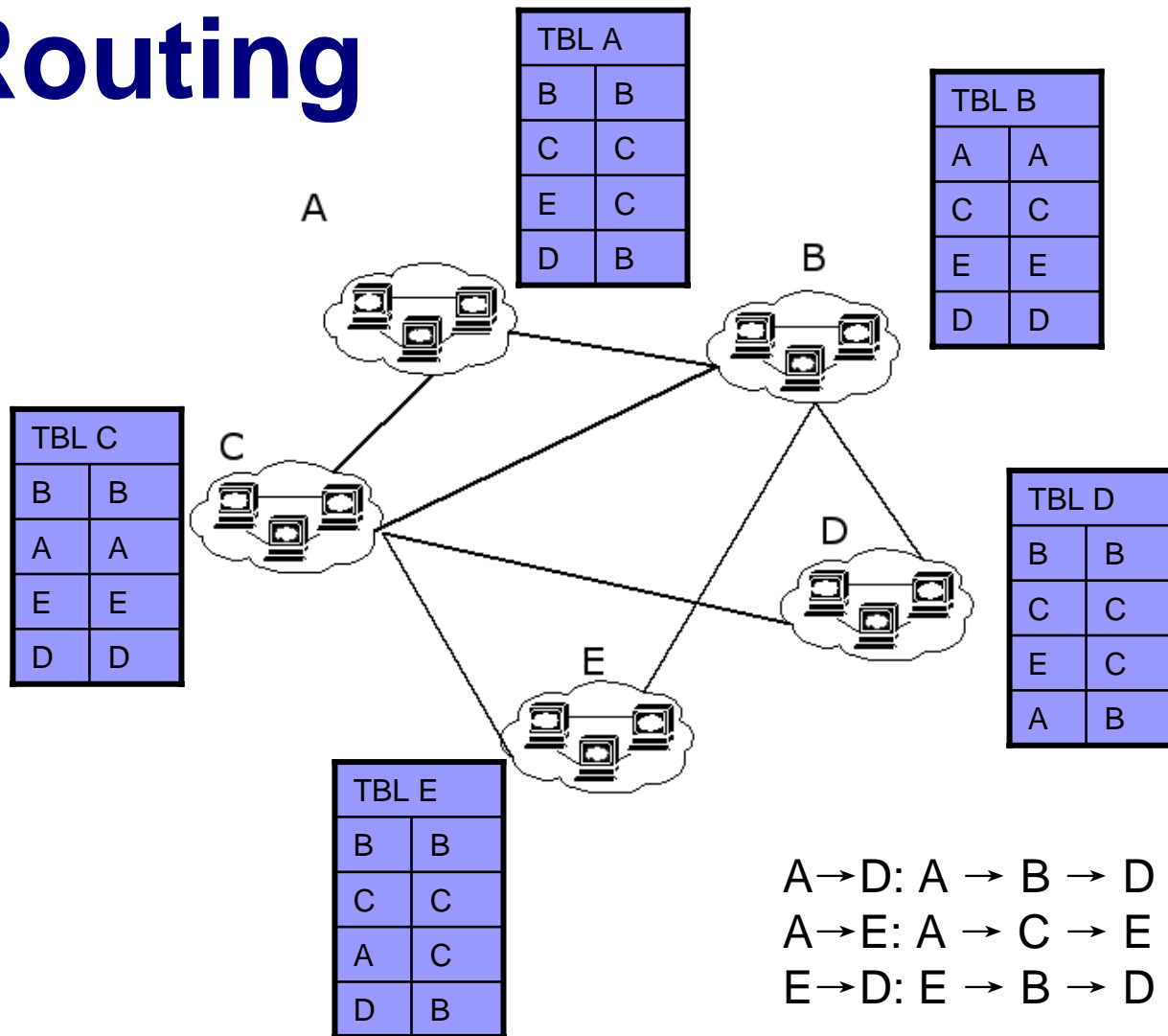
■ Internet

- NSF (National Science Foundation) became involved in ARPNET in 1984 and NSFNET was created, and later becomes the backbone of Internet. **Term Internet** was coined.
- In mid 1980s, the NSFNET became open to commercial interests
- In 1991, CA*net (Canadian) and CERN (European) were connected to the Internet backbone.
- Internet Society (ISOC) – Some nonprofit organizations
 - IETF (Internet Engineering Task Force)
 - IAB (Internet Architecture Board)
 - IESG (Internet Engineering Steering Group)
 - IRTF (Internet Research Task Force)

Packet switching

- It is the central idea of ARPANET architecture.
- In **packet switching**, messages to be sent over the network are first broken into small pieces known as **packets**, and these packets are sent independently to their final destination.

Routing

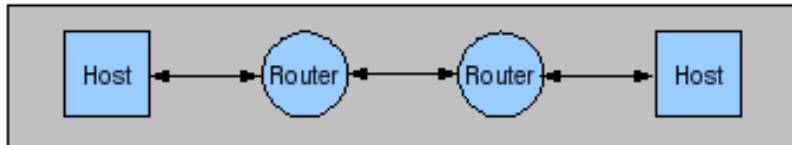


The Network Functions Layering

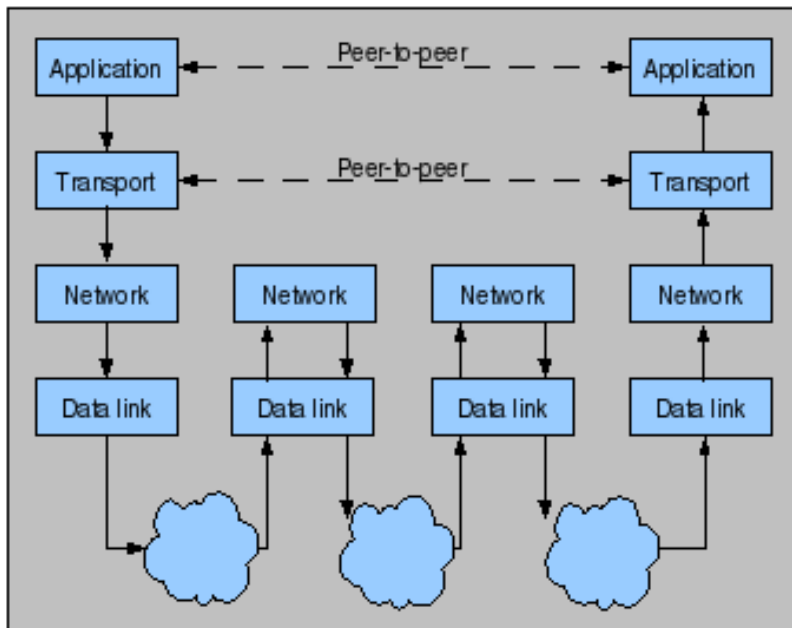
1. `http_deliver(url, html_file)`
 - Figure out the **IP address** and the **port number** of the receiving software
 - Call `tcp_deliver` with the IP address and port number, and the message is the html file
- 1.5. SSL: `https_deliver(url, html_file)` sits between `http_deliver` and `tcp_deliver` if “https://” is used instead of “[http://](#)” in the URL.
2. `tcp_deliver(ip_address, port, message)`
 - Break the message into packets
 - For each packet, call `ip_deliver(ip_address, ip_packet)`
3. `ip_deliver(ip_address, ip_packet)`
 - Routing through the networks by looking up the routing table
 - In each network, figure out the `mac_address` which corresponds to the `ip_address`, and assemble the `ip_packet` into `frame_packet`
 - Call `Ethernet_deliver(mac_address, frame_packet)`
4. `Ethernet_deliver(mac_address, frame_packet)`

Layers in the Internet protocols

Network connections

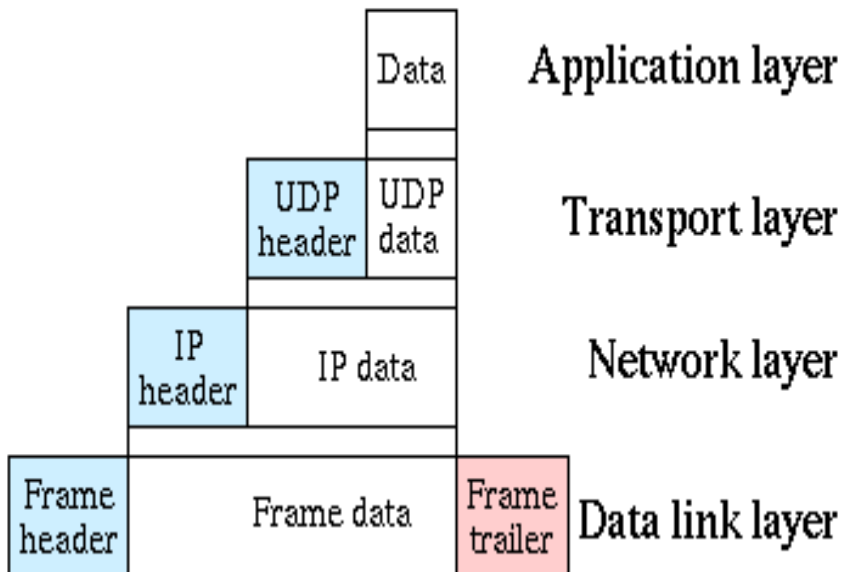


Stack connections



- **Application:** [DNS](#), [TLS/SSL](#), [FTP](#), [HTTP](#), [SMTP](#), ...
- **Transportation:** [TCP](#), [UDP](#), ...
- **Network:** [IP](#) ([IPv4](#), [IPv6](#))
- [ARP](#) and [RARP](#) operate underneath IP but above the link layer so they belong somewhere in between.
- **Link:** [Ethernet](#), [Wi-Fi](#), [PPP](#), [FDDI](#), [ATM](#), [Frame Relay](#), ...

Datagram (an UDP example)



- MAC addresses in frame header
- IP addresses in IP header
- UDP ports in UDP header (a port is used to identify an application within a computer)
- Data is the information intended for transmission

Internet Protocols: TCP/IP

- **Communication protocols** are sets of rules that describe how communication takes place.
- **IP addresses** are unique identifiers assigned to the computers (and devices) on the Internet. An IP address is a number, usually written as a dotted sequence such as “146.245.201.20”.
- The manner in which messages are sent and received over the Internet is defined by a pair of protocols called the **Transmission Control Protocol (TCP)** and **Internet Protocol (IP)**.
- **TCP** controls the method by which messages are broken down into packets and then reassembled when they reach their final destination.
- **IP** is concerned with labeling the packets (with IP addresses) for delivery and controlling the packets' paths (routing) from sender to recipient.
- In internetworking and computer network engineering, **Request for Comments (RFC)** documents are a series of memoranda encompassing new research, innovations, and methodologies applicable to Internet technologies.

TCP/IP software and Routing

- When a message is sent over the Internet, **TCP/IP software** uses the rules of TCP to break the message into packets and label the packets according to their sequence (e.g. packet 2 of 5).
- Then **TCP/IP software** follows the rules of IP to label these packets with routing information, including IP addresses of the source and destination computers. The labeled packets are called **IP datagram**.
- Once labeled, packets are sent independently.
- Special-purpose machines, called **routers**, receive the packets, access the routing information, and pass the packets on (possibly via the other routers) toward their destination. Routers use various types of information, including statistics on the network traffic pattern, to determine the best path for each packet to follow.
- When the packets arrive at the destination, **TCP software** running on the recipient's computer then reassembles the packets in the correct sequence to recreate the original message.

Domain Names and Domain-Name System (DNS)

- **Domain name** is the name assigned to each individual machine; it can be used in place of the machine's IP address.
- Domain names are hierarchical in nature
 - The leftmost part specifies the name of the machine
 - Subsequent parts indicate the organization to which the computer belongs to.
 - The right-most part is known as the **top-level domain** and identifies the type of organization with which the computer is associated.
 - E.g www.brooklyn.cuny.edu
- **Internet Corporation for Assigned Names and Numbers (ICANN)**, a nonprofit coalition of businesses, academic institutions, and individuals, accredits companies, known as **domain-name registrars**, which sell domain-name rights directly to consumers.
- **Domain-name System (DNS)** is a system where machine is assigned a name.
- **Domain-name servers** are used to store mappings between domain name and their corresponding IP addresses.

Email, Mailing lists, Email Viruses

- **Electronic mail**, abbreviated **e-mail** or **e-Mail** or **email**, is a method of composing, sending, storing, and receiving messages over electronic communication systems.
 - The Internet e-mail system is based on the Simple Mail Transfer Protocol (SMTP).
 - A modern Internet e-mail address (using SMTP or Usenet) is a string of the form *jsmith@example.com*. The part before the @ sign is the **local-part** of the address, often the username of the recipient, and the part after the @ sign is a domain name which can be looked up in the Domain Name System to find the Mail transfer agent accepting e-mail for that address.
- A **mailing list** is a collection of names and addresses used by an individual or an organization to send material to multiple recipients.

Web, URL, HTML

- **WEB:** An authorized individual on one computer can have access to files stored on another computer.
- Berners-Lee designed for the Web relied on two different types of software running on Internet-enabled computers.
 - **Web server** – a computer software that stores documents and serves them to other computers.
 - **Web browser** – a computer software that allows users to request and view the documents stored on the servers.
- **HTML (HyperText Markup Language):** Authors define the content of Web pages using HTML tags. The Web browser read HTML tags and render pages accordingly.
- **URL (Uniform Resource Locator):** A **Uniform Resource Locator** (URL) is a string of [characters](#) conforming to a standardized format, which refers to a [resource](#) on the [Internet](#) (such as a document or an image) by its location.
 - *Generic syntax: scheme://authority/path?query#fragment*
 - E.g. <http://www.cs.gc.cuny.edu/~tang/teaching/cis10/cis10.html>
- **HTTP (HyperText Transfer Protocol)** determines how messages are exchanged between browsers and servers using TCP/IP.
 - When the user clicks on a link in the browser, the browser identifies the Web server and sends a request for that page according to URL.
 - The server locates the specified page in its directories, and sends the page back to the browser for display using HTTP.
- A **hyperlink** is an element on a web page that connects to the page to another a webpage.
- Text that contains embedded hyperlinks is referred to as **hypertext**.

HTML tags

- `<html> ...</html>` defines a webpage.
- `<head>...</head>` defines a head of a webpage. The head element can contain information about the document. The browser does not display the "head information" to the user except `<title>...</title>`. The following tags can be in the head section: `<base>`, `<link>`, `<meta>`, `<script>`, `<style>`, and `<title>`.
- The title of a webpage defined by `<title> ...</title>` must be defined inside `<head>...</head>`, which is not part of the display of the webpage; it is displayed on the browser's title bar.
- `<body>...</body>` contains all the displayable elements of a webpage; it is inside `<html>...</html>`.
- `<H1>...</H1>`, ... define the headers inside a webpage.

<http://web.cs.gc.cuny.edu/~tang/teachings/cis10/html-lab-1.html>

<http://web.cs.gc.cuny.edu/~tang/teachings/cis10/html-lab-2.html>



Internet Searching and Some History

Search engines, Subject Directories, meta-search engines

- A **search engine** is a searchable database of Internet files collected by a computer program (called a wanderer, crawler, robot, worm, spider). Indexing is created from the collected files, e.g., title, full text, size, URL, etc. There is no selection criteria for the collection of files, though evaluation can be applied to ranking schemes that return the results of a query.
- A **subject directory** is a service that offers a collection of links to Internet resources submitted by site creators or evaluators and organized into subject categories. Directory services use selection criteria for choosing links to include, though the selectivity varies among services. Most directories are searchable.
- **Meta search engines** simultaneously search multiple search engines. They are also sometimes referred to as **parallel search engines, multithreaded search engines, or mega search engines**.

Other concepts in Internet Searching

- Deep web
- Advanced searches
- **Spider**: Program that traverses the Web from link to link, identifying and reading pages.
- **Index**: Database containing a copy of each Web page gathered by the spider
- Query logic: AND and OR
- Field search
 - TITLE:
 - URL:

History of computer science (LAB 6)

- People:
 - ☐ Pascal
 - ☐ Jacquard
 - ☐ Babbage
 - ☐ Ada Lovelace
 - ☐ John von Neumann
 - ☐ Alan Turing
 - ☐ Grace Hopper
 - ☐ Hollerith
 - ☐ Ritchie and Thompson: UNIX and the C programming language
- Machines and things
 - ☐ Pascaline
 - ☐ ENIAC
 - ☐ Moore's Law
 - ☐ RFC
 - ☐ Internet 2: **Internet2** or **UCAID** (University Corporation for Advanced Internet Development) is a [non-profit consortium](#) which develops and deploys advanced [network](#) applications and technologies, mostly for high-speed data transfer.
 - ☐ Unix
 - ☐ Difference Engine, Analytical Engine

GOOD LUCK! 😊