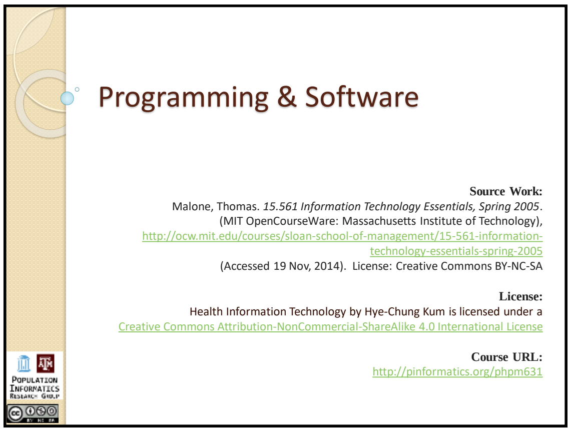
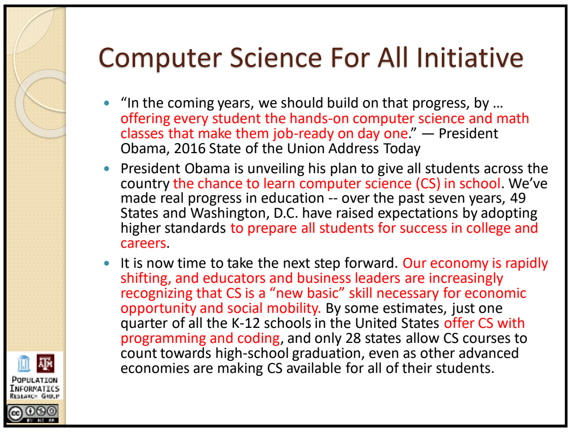
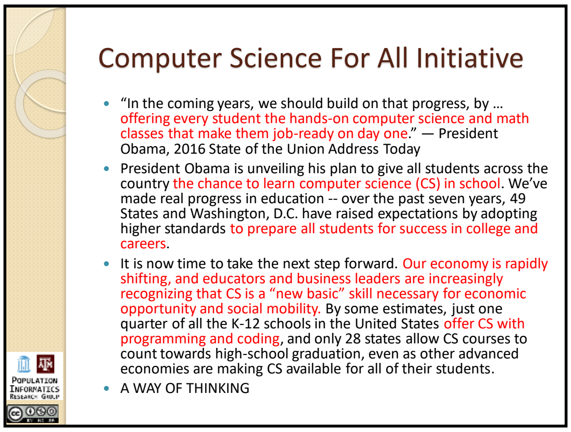


Programming & Software



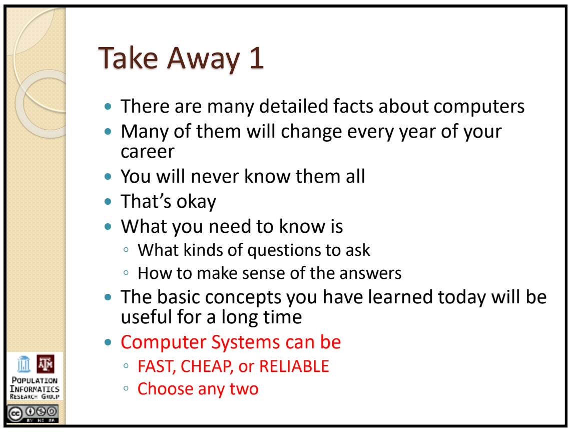
Computer Science For All Initiative

- "In the coming years, we should build on that progress, by … offering every student the hands-on computer science and math classes that make them job-ready on day one." — President Obama, 2016 State of the Union Address Today
- President Obama is unveiling his plan to give all students across the country the chance to learn computer science (CS) in school. We've made real progress in education -- over the past seven years, 49 States and Washington, D.C. have raised expectations by adopting higher standards to prepare all students for success in college and careers.
- It is now time to take the next step forward. Our economy is rapidly shifting, and educators and business leaders are increasingly recognizing that CS is a "new basic" skill necessary for economic opportunity and social mobility. By some estimates, just one quarter of all the K-12 schools in the United States offer CS with programming and coding, and only 28 states allow CS courses to count towards high-school graduation, even as other advanced economies are making CS available for all of their students.



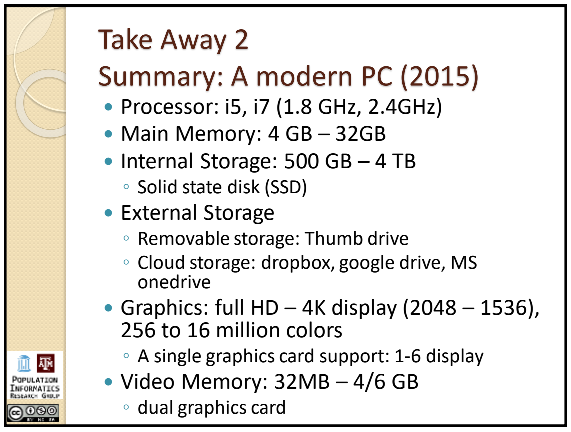
Computer Science For All Initiative

- "In the coming years, we should build on that progress, by … offering every student the hands-on computer science and math classes that make them job-ready on day one." — President Obama, 2016 State of the Union Address Today
- President Obama is unveiling his plan to give all students across the country the chance to learn computer science (CS) in school. We've made real progress in education -- over the past seven years, 49 States and Washington, D.C. have raised expectations by adopting higher standards to prepare all students for success in college and careers.
- It is now time to take the next step forward. Our economy is rapidly shifting, and educators and business leaders are increasingly recognizing that CS is a "new basic" skill necessary for economic opportunity and social mobility. By some estimates, just one quarter of all the K-12 schools in the United States offer CS with programming and coding, and only 28 states allow CS courses to count towards high-school graduation, even as other advanced economies are making CS available for all of their students.
- A WAY OF THINKING



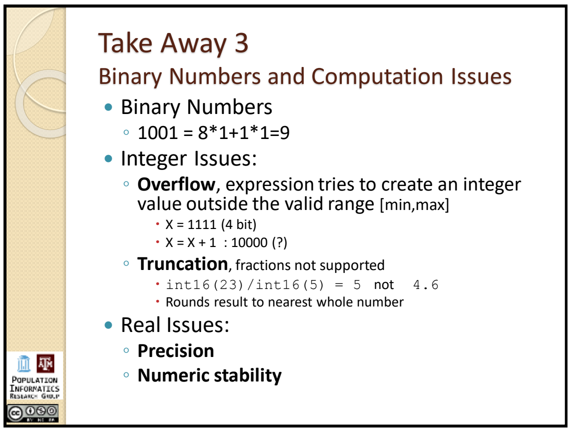
Take Away 1

- There are many detailed facts about computers
- Many of them will change every year of your career
- You will never know them all
- That's okay
- What you need to know is
  - What kinds of questions to ask
  - How to make sense of the answers
- The basic concepts you have learned today will be useful for a long time
- Computer Systems can be
  - FAST, CHEAP, or RELIABLE
  - Choose any two



Take Away 2
Summary: A modern PC (2015)

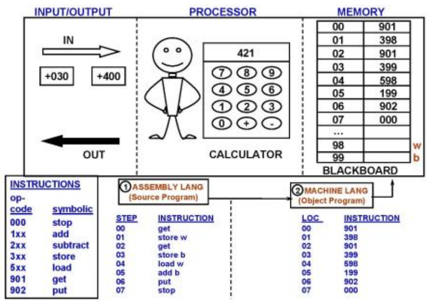
- Processor: i5, i7 (1.8 GHz, 2.4GHz)
- Main Memory: 4 GB – 32GB
- Internal Storage: 500 GB – 4 TB
  - Solid state disk (SSD)
- External Storage
  - Removable storage: Thumb drive
  - Cloud storage: dropbox, google drive, MS onedrive
- Graphics: full HD – 4K display (2048 – 1536), 256 to 16 million colors
  - A single graphics card support: 1-6 display
- Video Memory: 32MB – 4/6 GB
  - dual graphics card



Take Away 3
Binary Numbers and Computation Issues

- Binary Numbers
  - 1001 = 8*1+1*1=9
- Integer Issues:
  - **Overflow**, expression tries to create an integer value outside the valid range [min,max]
    - X = 1111 (4 bit)
    - X = X + 1 : 10000 (?)
  - **Truncation**, fractions not supported
    - `int16(23)/int16(5) = 5` not `4.6`
    - Rounds result to nearest whole number
- Real Issues:
  - **Precision**
  - **Numeric stability**

Lab 2
LMC MACHINGE LANGUAGE AND ASSEMBLY LANGUAGE



MEMORY	
00	901
01	398
02	901
03	399
04	598
05	199
06	902
07	000
...	
98	
99	

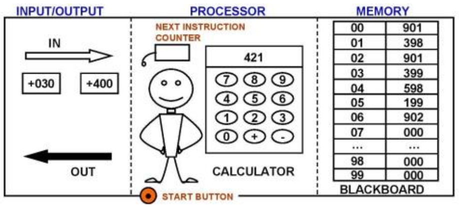
INSTRUCTIONS

op-code	symbolic
000	stop
1xx	add
2xx	subtract
3xx	store
5xx	load
901	get
902	put

STEP	INSTRUCTION
00	get
01	store w
02	get
03	store b
04	load w
05	add b
06	put
07	stop

LOC	INSTRUCTION
00	901
01	398
02	901
03	399
04	598
05	199
06	902
07	000

INSTRUCTION SEQUENCE



MEMORY	
00	901
01	398
02	901
03	399
04	598
05	199
06	902
07	000
...	...
98	000
99	000

Control Flow

- BRANCH always puts a new address in the instruction counter.
- BRANCH ZERO puts a new address in the instruction counter if the REGISTER is 0. Otherwise it increases the instruction counter by 1.

Examples: BRANCH 02
BRANCH ZERO 06

Topics

- Types of software
- What is a program (= software) ?
- Types of programming languages
- Software Stack
- Example: HTML5
- Tips for programming

Types of software

- System Software
  - Operating systems
  - Programming languages
    - Assemblers, compilers, interpreters (browser)
  - Database systems
- Application Software
  - General office tasks (word processing, etc.)
  - EHR
  - Accounting

Operating system

- Allocates and assigns:
  - Memory
    - e.g. file system, virtual memory
  - Processor time
    - e.g. multitasking (threading), multiprocessing
  - I/O devices
    - e.g. printer, keyboard, etc.
- May also provide other capabilities useful to many users or programs
  - Graphical User Interface (GUI) capabilities
  - Fonts, network protocols, …
  - Web browser?

Illusion #1: Multitasking/Threading

- Reality:
  - One CPU (multi-core)
  - One instruction at a time (one instruction per core)
- Illusion:
  - Several application programs executing concurrently
- Implementation:
  - Operating system divides CPU time among application programs (time sharing)
    - Each program “thinks” it is the only one running
    - OS copies Instruction Pointer and Registers back and forth as each program takes its turn (Thrashing)
  - Threading: multiple CPUs

Illusion #2: Virtual Memory

- Reality: finite memory
- Illusion:
  - Process (and its programmer) not aware that main memory is too small (the big memory illusion) and assumes infinite memory
- Implementation
  - Divide memory into a unit of data (called a "page") to hard disk and copy into memory when needed
  - Processes asks for a main memory location (Page #, offset on page)
  - OS has to get that page into main memory if not already there
  - OS basically, copies pages back from hard disk to main memory as they're needed

Illusion #3: File Systems

- Reality:
  - Sequence of 0/1
  - Packaged into blocks
- Illusion:
  - Disks are sets of directories
  - Directories contain other directories or files
  - Files are variable-size byte sequences
  - Directories and files have names

Illusion #4: Windows and Menus

- Reality: Screen is an array of pixels
- Illusion 1: Menus
  - Depending on where you click, different action happens
  - Technique: OS look up location where mouse was clicked, executes appropriate action
- Illusion 2: Overlapping windows
  - A window may cover part of or all of another
  - When a window is uncovered, its contents are redisplayed
  - Technique: OS saves bitmap of covered windows
    - Application does not need to know how to redraw the contents of its window

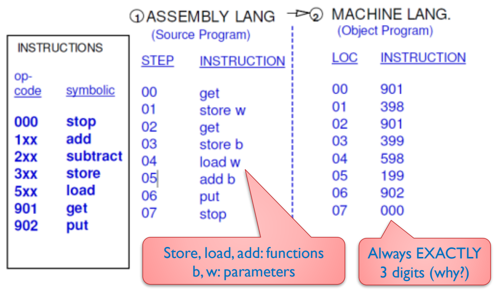
Selecting an Operating System

- Is our existing application software compatible with the OS?
- Does the OS have a large base of compatible software?
- How reliable is the OS? Does it crash frequently?
- Is the OS available for a wide variety of hardware?
- How quickly does it run?
- How easy it is to learn and use?
- How easy is it to install, configure, manage?
- How much does it cost?

Topics

- Types of software
- What is a program (= software) ?
- Types of programming languages
- Software Stack
- Example: HTML5
- Tips for programming

A Sample LMC Program



INSTRUCTIONS

op-code	symbolic
000	stop
1xx	add
2xx	subtract
3xx	store
5xx	load
901	get
902	put

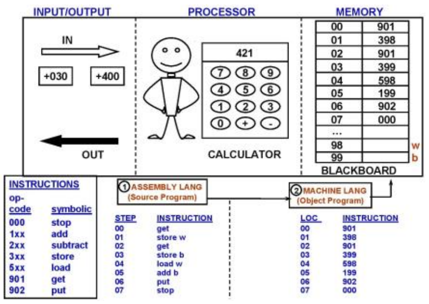
① ASSEMBLY LANG (Source Program) → ② MACHINE LANG. (Object Program)

STEP	INSTRUCTION	LOC	INSTRUCTION
00	get	00	901
01	store w	01	398
02	get	02	901
03	store b	03	399
04	load w	04	598
05	add b	05	199
06	put	06	902
07	stop	07	000

Basic Facts to Ask About Any Computer LMC Answers

- MEMORY
  - BASIC UNIT: 3 DECIMAL DIGIT NUMBER
  - MAXIMUM SIZE: 100 LOCATIONS
- REGISTERS
  - HOW MANY: 1
  - NUMBERS: 3 DECIMAL DIGIT NUMBER
- INSTRUCTIONS
  - NUMBER: 7 INSTRUCTIONS

LMC MACHINE LANGUAGE AND ASSEMBLY LANGUAGE



INSTRUCTIONS

op-code	symbolic
000	stop
1xx	add
2xx	subtract
3xx	store
5xx	load
901	get
902	put

STEP	INSTRUCTION
00	get
01	store w
02	get
03	store b
04	load w
05	add b
06	put
07	stop

LOC	INSTRUCTION
00	901
01	398
02	901
03	399
04	598
05	199
06	902
07	000

Is this correct?

INSTRUCTIONS

op-code	symbolic
000	stop
1xx	add
2xx	subtract
3xx	store
5xx	load
901	get
902	put

LOC	Instruction	Register
00	Get	400
01	Store W	400
02	Get	030
03	Store B	030
04	Load W	400
05	Add B	430
06	Put	430
07	Stop	430
00	Get	400
01	Store W	400
02	Get	030
03	Store B	030
	~~Load W~~	~~400~~
04	Add ~~B~~ W	430
05	Put	430
06	Stop	430

Kind of

- When is it not correct?
  - Timeshare (multitasking)
    - Run up to 03
    - Come back to 04 later
    - Modular
      - 00-01
      - 02-03
      - 04-06

LOC	Instruction	Reg
00	Get	400
01	Store W	400
02	Get	030
03	Store B	030
04	Load W	400
05	Add B	430
06	Put	430
07	Stop	430
00	Get	400
01	Store W	400
02	Get	030
03	Store B	030
	~~Load W~~	~~400~~
04	Add ~~B~~ W	430
05	Put	430
06	Stop	430

Kind of

- When is it not correct?
  - Timeshare (multitasking)
    - Run up to 03
    - Come back to 04 later
      - Is register going to be B (30)?
    - Modular
      - 00-01
      - 02-03
      - 04-06
  - Different Modules
    - Lose B, so have to run in one run (02-04)

LOC	Instruction	Reg
00	Get	400
01	Store W	400
02	Get	030
03	Store B	030
04	Load W	400
05	Add B	430
06	Put	430
07	Stop	430
00	Get	400
01	Store W	400
02	Get	030
	~~Store B~~	~~030~~
	~~Load W~~	~~400~~
03	Add ~~B~~ W	430
04	Put	430
05	Stop	430

Example Exam Questions

- What is the value in the register after executing memory location 04?
- What instruction can I delete and still get the same output?
- Do the following two programs output the same value?

LOC	Instruction	Reg
00	Get	400
01	Store W	400
02	Get	030
03	Store B	030
04	Load W	400
05	Add B	430
06	Put	430
07	Stop	430
00	Get	400
01	Store W	400
02	Get	030
	~~Store B~~	~~030~~
	~~Load W~~	~~400~~
03	Add ~~B~~ W	430
04	Put	430
05	Stop	430

Essential parts of programs (= software ≈ applications)

- Data: Variable – name, value pair

Location (Name)	Value
61 (wage)	400
62 (bonus)	30

- Procedure: Function/method

Location (Name)	Value	Assembly
03	561	Load wage (location 61)
04	162	Add bonus (location 62)

  - Sequences of actions (commands) for the computer
  - Pass parameters (variables) to functions
    - Load WAGE, Add BONUS
  - Usually, simple and iteratively developed
  - Building blocks. Reuse
  - Libraries: bundled functions. e.g., strcomp(str1, str2): is str1 same as str2?

Example: calculate avg

- Total = Sum (w_1 to w_n)
- N=count (w_1 to w_n)
- Avg=total/N

- Sum (w_1 to w_n)
  - Load w_1
  - Add w_2
  - …
  - Add w_n

- House
  - Windows
  - Doors
  - Rooms
  - etc

- Define Window
- Define Door
- Define Room
- etc

Variables?
Functions?
Parameters?

Programming Languages

- Machine language
- Assembly language
- High-level languages
- Fourth-generation languages
- Object-Oriented Programs (OOP)

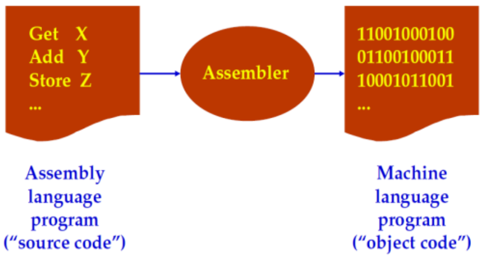
Machine Language

- Binary
- Executable
- Machine dependent
- Stored in the computer when the program is running
- Example:
  - 01110110001010010010 ….

Assembly Language

- Mnemonic
- Symbolic addressing
- One-to-one correspondence with machine language
- Example:
  - Get X
  - Add Y
  - Store Z

Automatically translating Assembly Language to Machine Language


Get X
Add Y
Store Z
…
Assembler
11001000100
01100100011
10001011001
…
Assembly language program ("source code")
Machine language program ("object code")

High-level Languages

- Closer to how people think about their problems
- No one-to-one correspondence to machine language
- Compiler
- General purpose
- Example:
  - Z = X + Y
- Fortran, Basic, Visual Basic, C, C++, Java

Example 1: Basic

```
'AVERAGING INTEGERS ENTERED THROUGH THE KEYBOARD
CLS
PRINT "THIS PROGRAM WILL FIND THE AVERAGE OF INTEGERS YOU ENTER"
PRINT "THROUGH THE KEYBOARD. TYPE 999 TO INDICATE THE END OF DATA."
PRINT
SUM=0
COUNTER =0
PRINT "PLEASE ENTER A NUMBER"
INPUT NUMBER
DO WHILE NUMBER <> 999
    SUM=SUM+NUMBER
    COUNTER=COUNTER+1
    PRINT "PLEASE ENTER THE NEXT NUMBER"
    INPUT NUMBER
LOOP
AVERAGE=SUM/COUNTER
PRINT "THE AVERAGE OF THE NUMBER IS"; AVERAGE
END
```

Example 2: C++

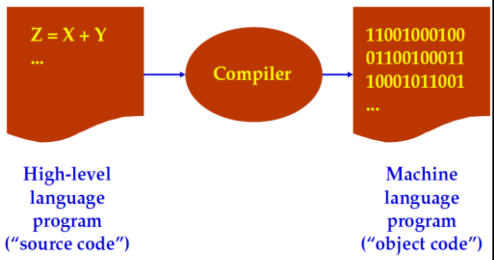
```
// AVERAGING INTEGERS ENTERED THROUGH THE KEYBOARD
#include <iostream.h>
main ( )
{
    float average;
    int number, counter = 0;  int sum = 0;
    cout << "THIS PROGRAM WILL FIND THE AVERAGE OF INTEGERS YOU ENTER \n";
    cout << "THROUGH THE KEYBOARD. TYPE 999 TO INDICATE END OF DATA. \n";
    cout << "PLEASE ENTER A NUMBER";
    cin   >> number;
    while (number !=999)
       {

            sum =sum + number;
            counter++;
            cout << "\nPLEASE ENTER THE NEXT NUMBER";
            cin   >>number;
       }
     average = sum / counter;
     cout << "\nTHE AVERAGE OF THE NUMBERS IS "<< average;
}
```

Example 3: Java

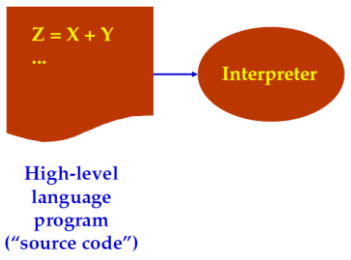
```
import java.io.*;
import java.lang.*;
/**
 ** Prompts user for list of numbers and outputs average
**/

class AverageNumbers {
   public static void main (String[] args) {
     float sum = 0;
     float average = 0;
     int counter = 0;

     System.out.println("THIS PROGRAM WILL FIND THE
     AVERAGE OF THE INTEGERS YOU ENTER
     THROUGH THE KEYBOARD.  TYPE 999 TO
     INDICATE END OF DATA.");

     try
```

```
{
   BufferedReader in = new BufferedReader(new
InputStreamReader(System.in));
   String cin = "0";
   System.out.println("Please enter a number.");
   while (!(cin=in.readLine()).equals("999"))
          {
             sum = sum + Integer.parseInt(cin);
             counter = counter + 1;
             System.out.println("Please enter another
number.");
          }
   in.close();
   average = sum/counter;
   System.out.println("The average of the numbers is :
"+average);
}
catch (IOException e)
{
          System.out.println("Ooops..");
}
  }
}
```

Automatically Translating High-Level Language to Machine Language


Z = X + Y
…
Compiler
11001000100
01100100011
10001011001
…
High-level language program ("source code")
Machine language program ("object code")

Interpreting High-Level Language



Interpreting High Level Language

- Advantages
  - Can give machine independence
    - e.g. one machine can "look" like another
  - Can be easier to debug and modify
  - Can give more flexibility at "run time"
- Disadvantages
  - Slower

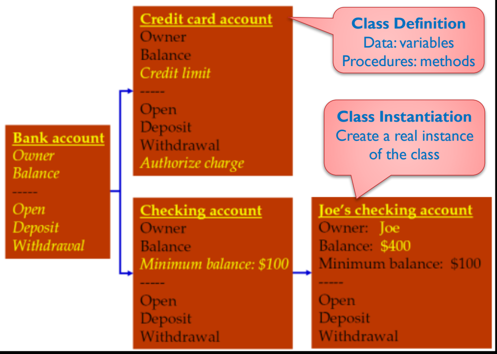
"Fourth-generation" Languages

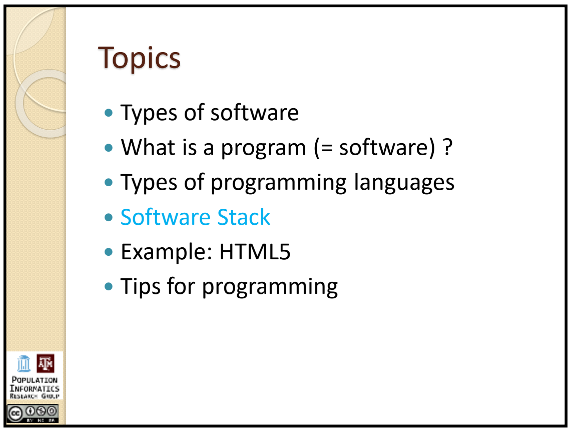
- Even closer to how people think about their problems
- Special purpose
- Examples:
  - General: PHP, Perl, Python, Ruby
  - Database query languages: SQL
    - FIND ALL RECORDS WHERE NAME IS "SMITH"
  - Data manipulation: R, SAS, SPSS, STATA, XML
  - Web development: HTML/CSS
  - Spreadsheet formulas?

Object-Oriented Programming

- A special kind of high-level language
- Can increase programming efficiency and software re-use
- Combines procedures and data into "objects"
- Arranges objects into "class hierarchies"
- "Inherits" properties of objects in this hierarchy

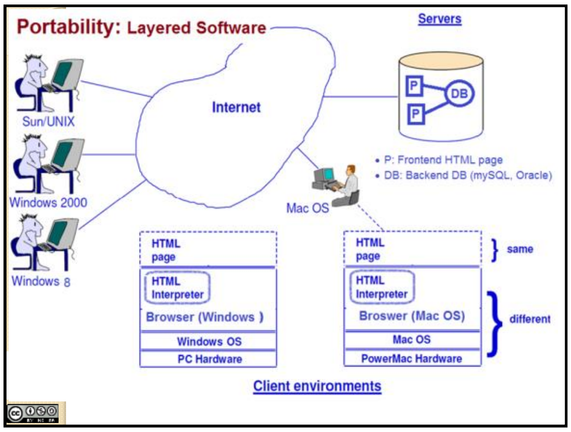
Class Inheritance in OOP

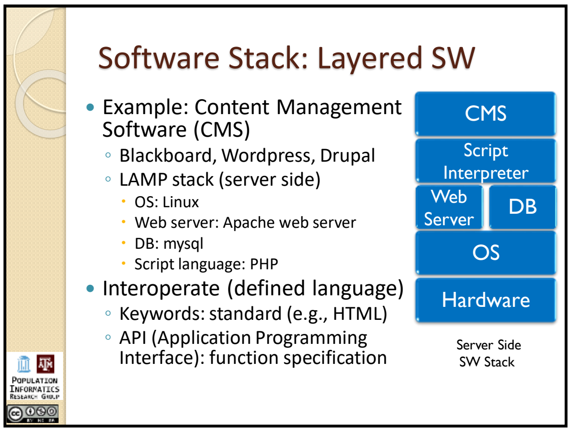




Topics

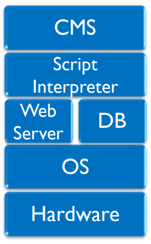
- Types of software
- What is a program (= software) ?
- Types of programming languages
- Software Stack
- Example: HTML5
- Tips for programming



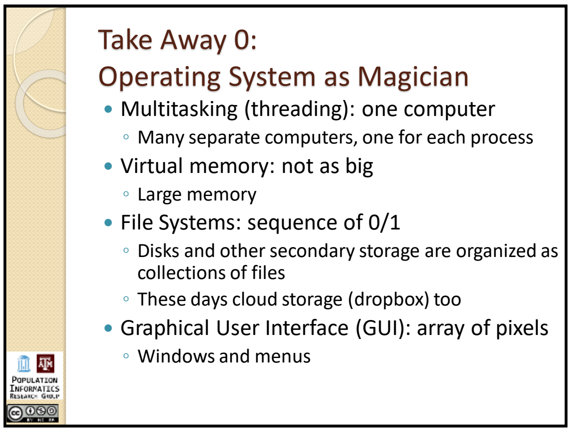


Software Stack: Layered SW

- Example: Content Management Software (CMS)
  - Blackboard, Wordpress, Drupal
  - LAMP stack (server side)
    - OS: Linux
    - Web server: Apache web server
    - DB: mysql
    - Script language: PHP
- Interoperate (defined language)
  - Keywords: standard (e.g., HTML)
  - API (Application Programming Interface): function specification

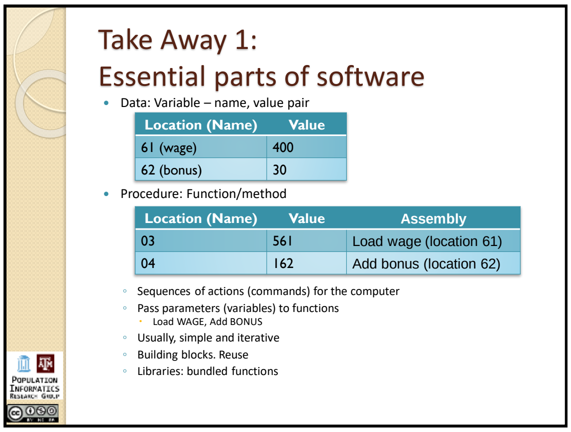


Server Side SW Stack



Take Away 0: Operating System as Magician

- Multitasking (threading): one computer
  - Many separate computers, one for each process
- Virtual memory: not as big
  - Large memory
- File Systems: sequence of 0/1
  - Disks and other secondary storage are organized as collections of files
  - These days cloud storage (dropbox) too
- Graphical User Interface (GUI): array of pixels
  - Windows and menus



Take Away 1: Essential parts of software

- Data: Variable – name, value pair

Location (Name)	Value
61 (wage)	400
62 (bonus)	30

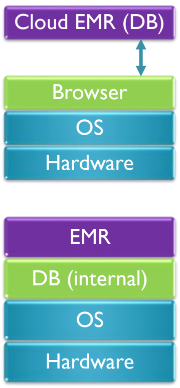
- Procedure: Function/method

Location (Name)	Value	Assembly
03	561	Load wage (location 61)
04	162	Add bonus (location 62)

  - Sequences of actions (commands) for the computer
  - Pass parameters (variables) to functions
    - Load WAGE, Add BONUS
  - Usually, simple and iterative
  - Building blocks. Reuse
  - Libraries: bundled functions

Take Away 2: Software Stack

- Computers operate by composing different layers of software
- Each layer is upgraded periodically
  - To accommodate new hardware, software (e.g., OS)
  - To block securities vulnerability
- Each layer have to interoperate
- When buying software for each layer consider
  - Interoperability
  - Expected life time
    - Assess support, required layers etc
  - Cost
    - Consider short, medium, long term



Take Away 3: Programming Software Development

- There are many ways to get to where you want to go
  - Some are better than others
  - Most have pros/cons
- You have to KNOW where you want to go
  - specification of the system
  - Must know roughly what is possible (current technology) and how difficult (cost)
- You have to plan out your steps
  - Key is partitioning the given task to appropriate size (too detailed or too broad is both not good)
- Iterative
  - Plan (start from nothing, or something similar)
  - Implement
  - Test, debug, evaluate
  - Readjust course as you go
- Expert (ask questions)
  - Know the level of detail at each step
  - Good decisions on key factors: What to use and why

break
Presentation & reading log

Topics (7:30)

- Types of software
- What is a program (= software) ?
- Types of programming languages
- Software Stack
- Example: HTML5
- Tips for programming

HTML5

- Hyper Text Markup Language
- Commands = Tags: <TAG></TAG>
  - <html></html>
  - XML: Extensible Markup Language
    - Main issue: filesize too big
- DOM: Document Object Model
  - Object Oriented
  - Page is composed of objects
    - Title, header, paragraph, table (row <tr>, cells <td>)
  - Objects have attributes
    - Color, font, size
  - Set attributes
    - Initially: css (layout) & html
    - Dynamically change: javascript

HTML5 components

- HTML: content
  - Objects: <head><body><footer>
  - Viewable (rendered) and nonviewable objects
- CSS: layout (color, format, font etc)
  - Define *attributes* of an object
- Javascript: action = dynamically change
  - Click a button, update data etc
  - Create, delete objects
  - Dynamically change *attributes* of an object
  - AJAX: asynchronous Javascript and XML
    - Group of interrelated web development techniques used on client-side to create asynchronous web applications

HTML5

- Objects
  - <head>: not rendered
  - <body>
    - <h1>
    - <p>: paragraph. Name (id)=demo

```
<html>
   <head>
   </head>
   <body>
      <h1>My First JavaScript</h1>
      <p id="demo">JavaScript can change the style of an HTML element.</p>
   </body>
</html>
```

CSS

```
<head>
        <style>
          body {background-color:lightgrey}
          h1   {color:blue}
          p    {color:green}
        </style>
</head>
```

```
<!-- rgb(211,211,211) rgb(0,0,255) rgb(0, 128,0)-->
<head>
        <style>
          body {background-color:#D3D3D3}
          h1   {color:#0000FF}
          p    {color:#008000}
        </style>
</head>
```

Javascript

http://www.w3schools.com/html/tryit.asp?filename=tryhtml_script_styles

```
<html>
   <head>
      <script>
         function myFunction() {
            document.getElementById("demo").style.color = "red";
            document.getElementById("demo").style.fontSize = "25px";
         }
      </script>
   </head>
   <body>
      <h1>My First JavaScript</h1>
      <p id="demo">JavaScript can change the style of an HTML element.</p>
      <button type="button" onclick="myFunction()">Click Me!</button>
   </body>
</html>
```

References

- Google 'HTML'
  - Look at ref: http://www.w3schools.com/html/
- Google 'chrome API'
  - https://developer.chrome.com/extensions/api_index
- Google 'firefox API'
  - https://developer.mozilla.org/en-US/docs/Web/API

Lab

- Do 1 together
- Submit
  - SOMETHING tonight (lab) – I want to see you have started
  - SOMETHING next Monday night (assignment & lab) – I want to see you are making progress
  - SUBMIT everything in two weeks – this is what is Graded
    - Requirement: minimal. Could learn as much as you like

Few tips

- Using text editors
  - Notepad, TextEdit
- Extension (file type): save as
  - .txt, .doc, .html, .pdf
- View source: class page
- Moving from w3 to working on your computer (for assignment)
- What is in a URL address?