INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY

HYDERABAD

Introduction to Software Systems

Who are we?

Y. Raghu Reddy

Associate Professor

Software Engineering Research Center

Co-Instructors:

Lalit Mohan (Wells Fargo, IDRBT, PhD research scholar);

Sai Anirudh (SkillSoft, PhD research scholar)

Course Details

- **Course Objective:** The aim of this course is to provide a working knowledge on tools and processes for building simple software systems.
- **Course Structure:** 13 classes (1 hr per class), Lab Work – 1 every week (3 hrs per week)
- **Grading split up:** Quiz (Lab Exam) – **10%**, Exam (Lab Exam) – **22%**, Assignment – **48%** (3 Assignments), Others – **20%** (Lab Activities, Surprise quiz/test, Inclass Activity)
- **Course Notes:** Reference Material and relevant notes will be made available on Moodle. Students are expected to read the notes/reading material, put on effort, work towards rising your problem-solving skills and learn things by doing.
- **Facilitators:** Instructors and Teaching Assistants.
- **Time:** Honour Time and Come with learning mindset. Ensure that you record your queries and discuss them offline when we run out of time.
- **Tutorials/Lab Work:** Linux Commands, Shell Scripting, HTML, CSS, JavaScript, Python
- **Reference Books/Materials:**
  - Mastering Linux Shell Scripting : A practical guide to Linux command-line, Bash scripting, and Shell programming, by Mokhtar Ebrahim, Andrew Mallett
  - Learning Python: Powerful Object-Oriented Programming, by Mark Lutz
  - JavaScript: The Definitive Guide, by David Flanagan
  - Software Engineering Principles (from various sources)
  - Workbook given by the course instructors

Academic Honesty

A helps B in task X

⇒B doesn't get opportunity to do task X

⇒B doesn't learn the skill to do task X

⇒B gets spoilt, dependent and unfit for jobs requiring skills of X

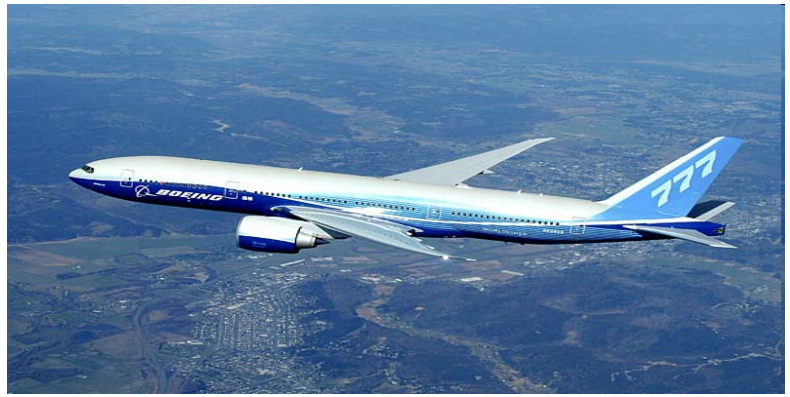
⇒You may think it is okay to do it only once and not repeat it. But when a thing is done once, it gets wired into the brain as being "okay"; and unless there is a strong reason, it *will* repeat.

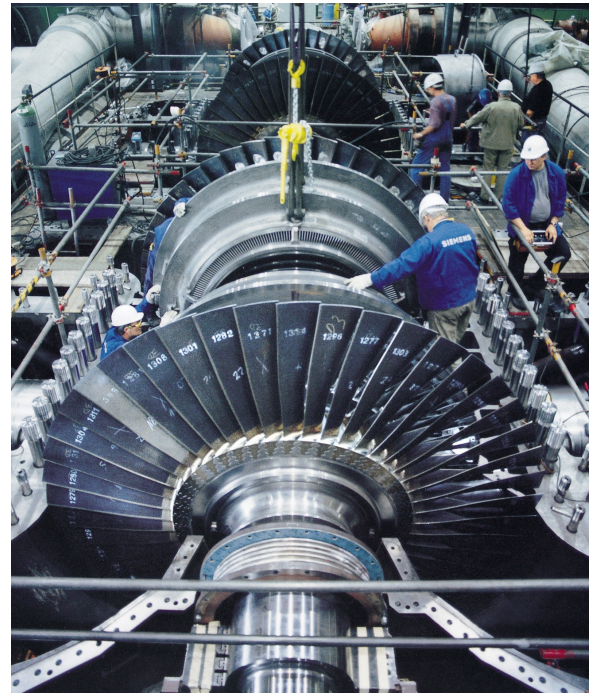
If you want to help, help to learn.

What's common in these?

• They are large complex "systems" with *lot* of software & hardware.

- The Boeing 777 flies with over 4,000,000 lines of code on-board.
- A typical top-level game has between 1 and 2 M SLOC (source lines of code)
- Thousands of devices





Programs

Teamwork

Process

Engineering design

Communication

What's a System?

- Commonly used/understood definition
  - ➢ Set of inter-related components working together to achieve a common objective
- A system may be "Natural" or "Engineered"
  - Solar system (Natural)
  - Telephone network system, power plants, etc. (Engineered)
  - Systems have boundaries – due to various reasons

Communication

Teamwork

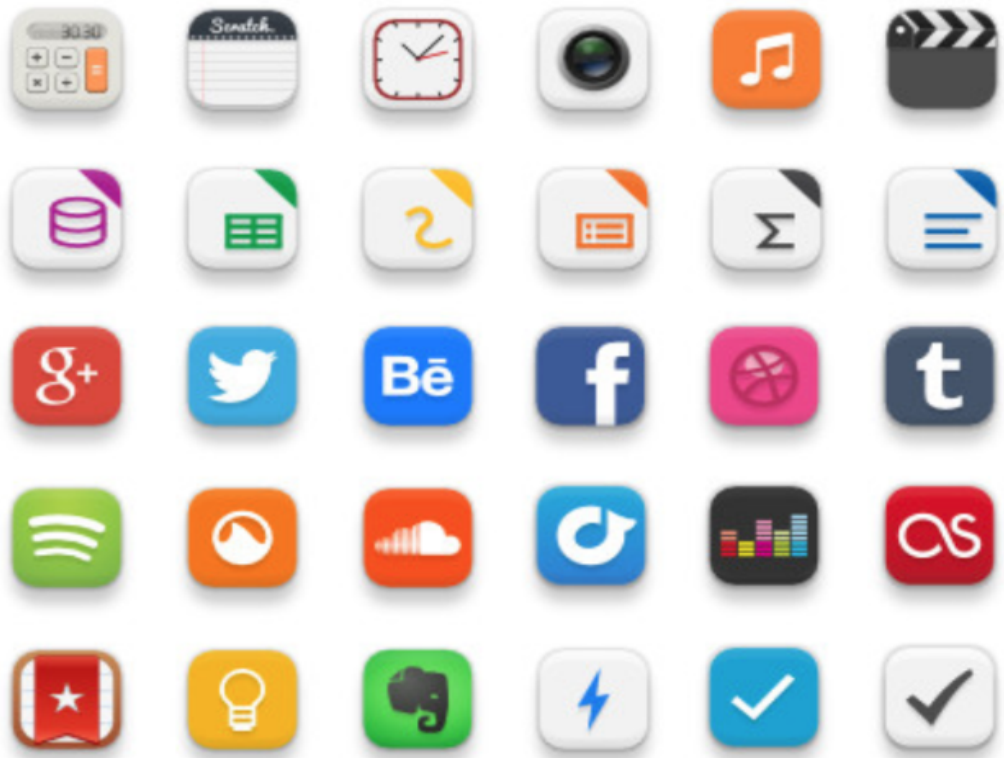
Process

Engineering design

Programs

This course is about… Tools, Technologies and processes for Software Systems.

Hardware



Software

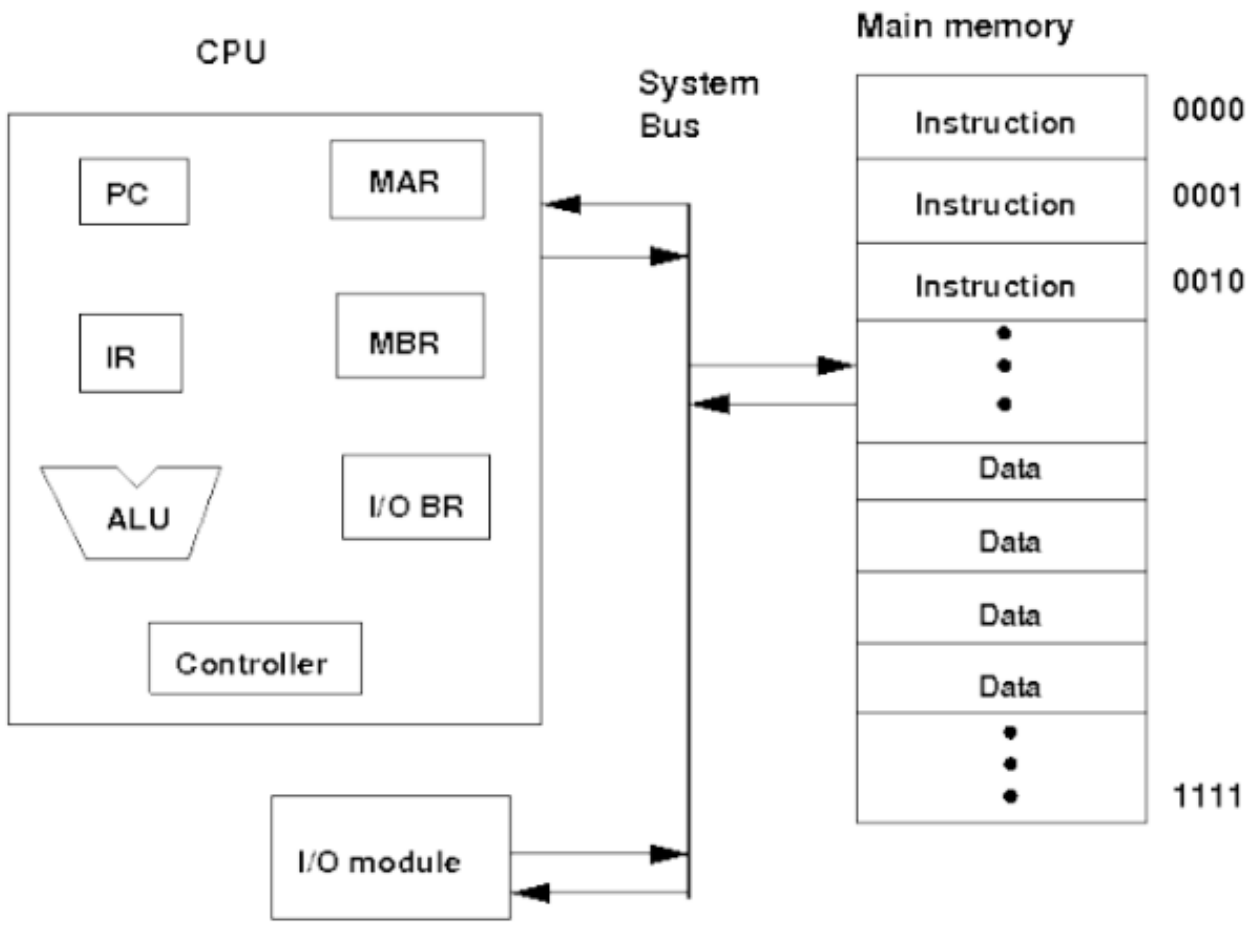
Requirements for Basic Computing Device

- Input and Output Unit
- Memory Unit - {primary, secondary}
- Processing Unit – {ALU, CU, Registers}
- Interconnection Structures - {control, address & data buses}

Ingredients for a Better Computation Power

- Organization of Hardware
- Mode of Processing
- Storage
- Speed
- Complexity
- Control Mechanism
- Resilience

HARDWARE

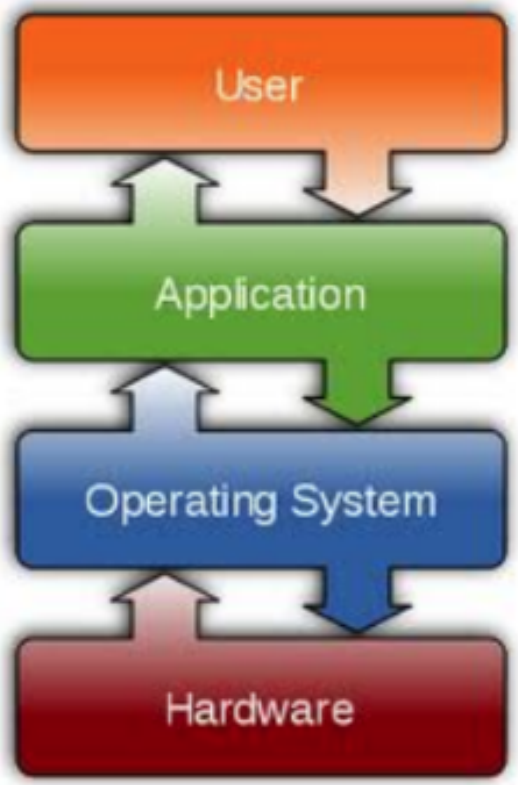


Software – A set of programs, procedures, algorithms and its documentation. It is written using programming languages

- *High-Level Languages* – e.g. C, C++, Java etc.
- *Assembly Language* – mnemonic-based e.g. ADD, SUB, MOV
- *Low-Level Language* – native language of computer circuitry

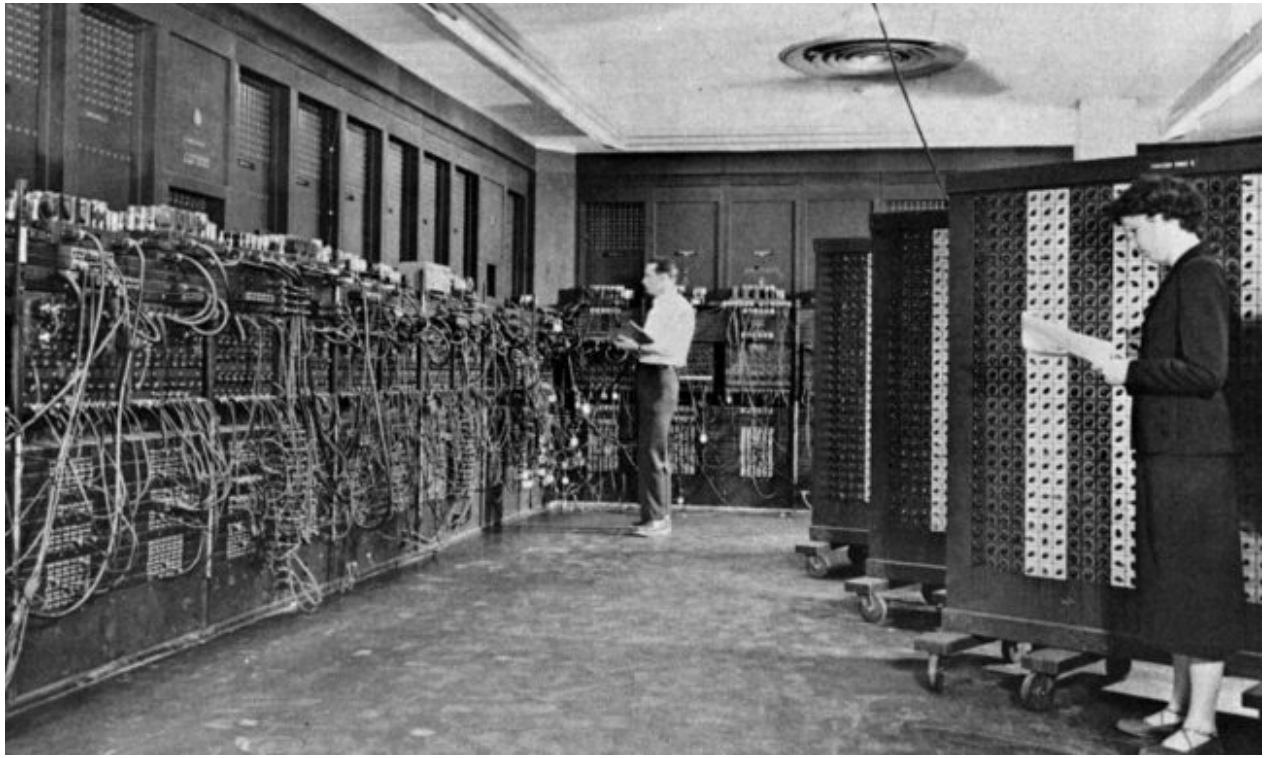
Language – has Syntactic + Semantic rules, otherwise called as Syntax, logical and/or runtime errors

Types – Complied Languages Vs Interpreted Languages
Usage – Application Software Vs System Software



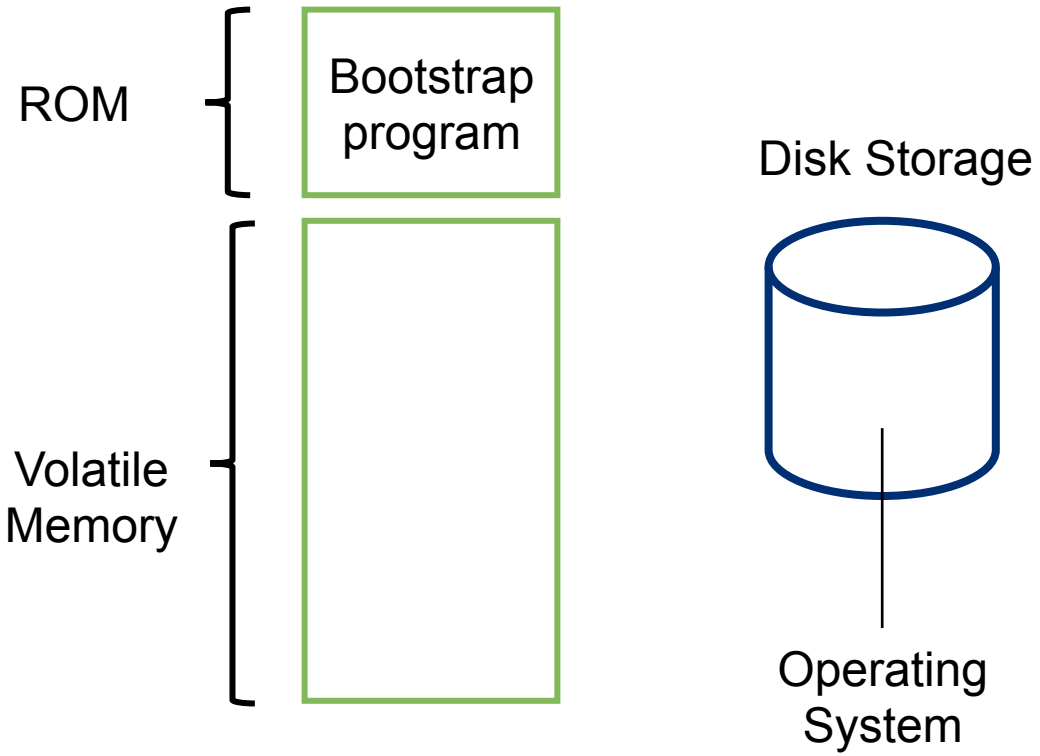
Operating System – A set of Software that ***manages*** computer hardware resources and ***serves*** other programs. It is a low-level software that supports a computer's basic functions, such as scheduling tasks and controlling peripherals. ***Examples*** – Windows, Unix, BSD, Linux, OS X, iOS, ChromeOS, Android etc.

Memory - Computer memory is any physical device capable of storing information temporarily or permanently. **RAM (Random Access Memory)** is an example for temporary memory where as **ROM (Read only Memory)** is an example for permanent memory

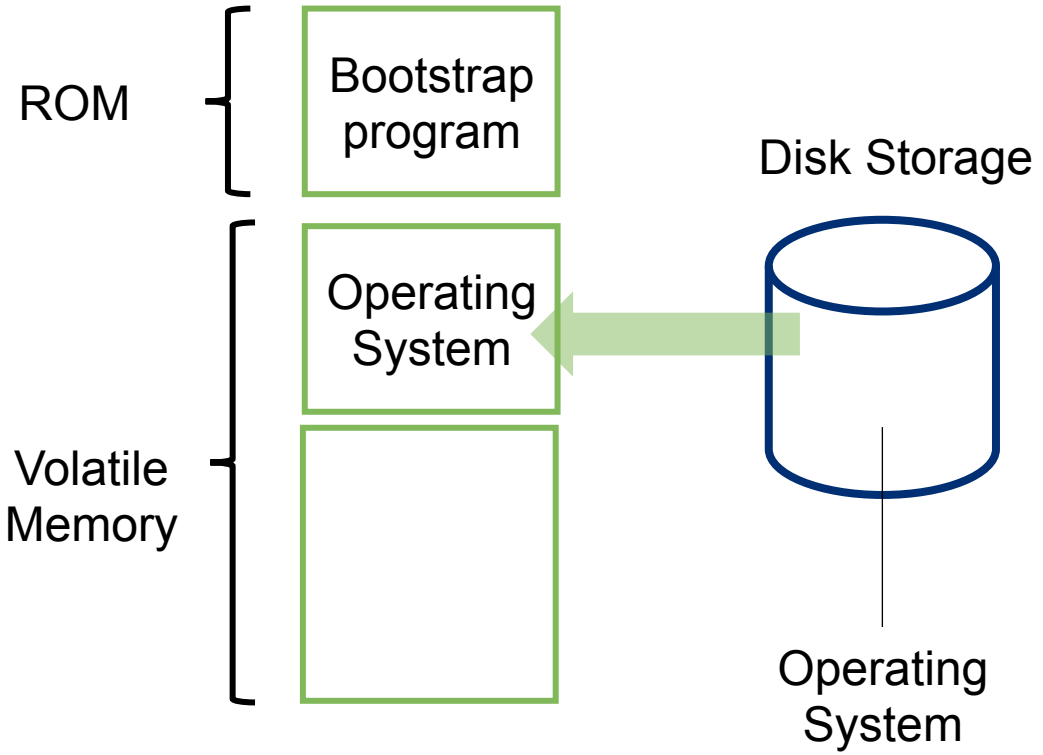


Did you know? – A Modern Smart Phone is 320,000,000 times "more powerful" than the ENIAC

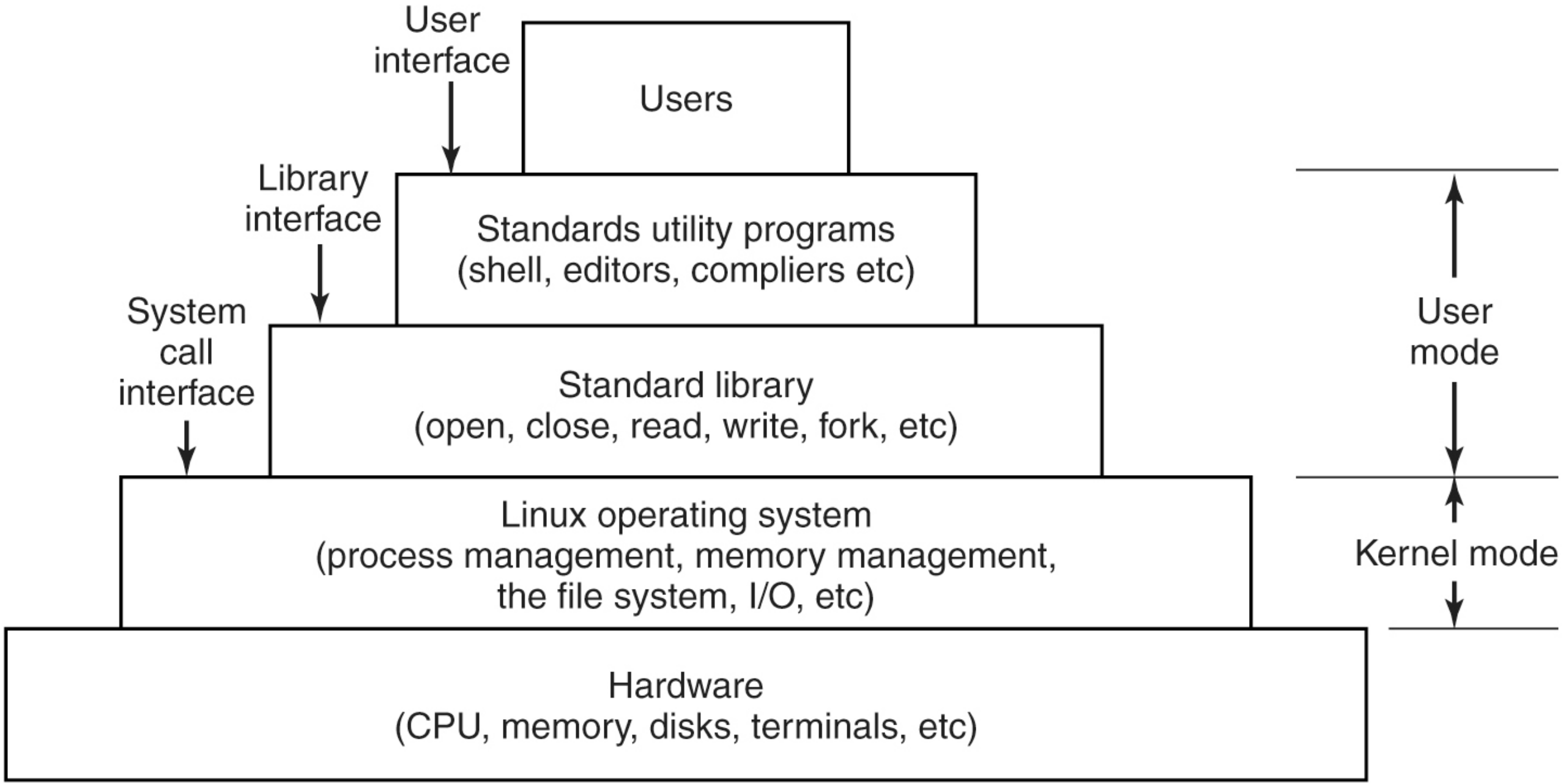
https://www.quora.com/How-many-ENIAC-computers-would-fit-into-an-iPhone



Step 1 – Machine starts by executing the bootstrap program already in ROM. Operating System is stored in Mass Storage (Disk Storage)

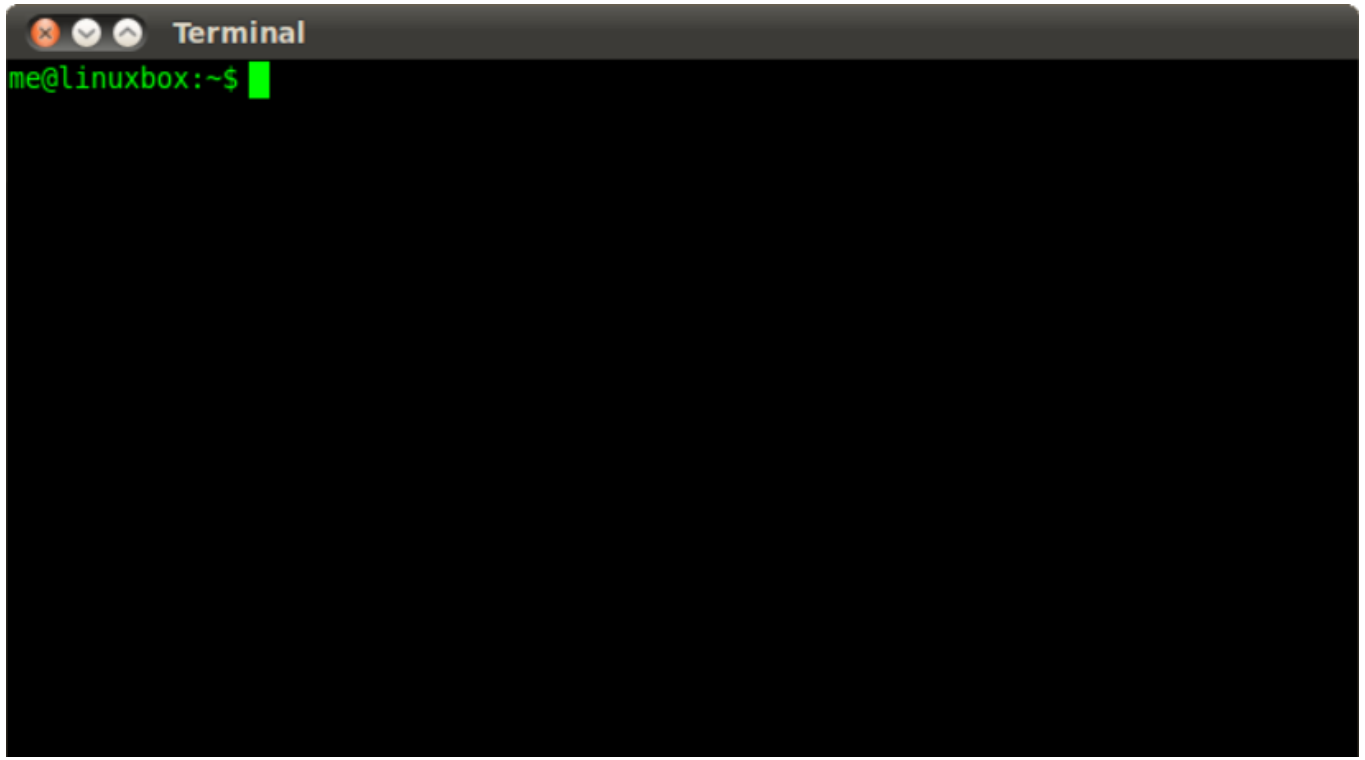


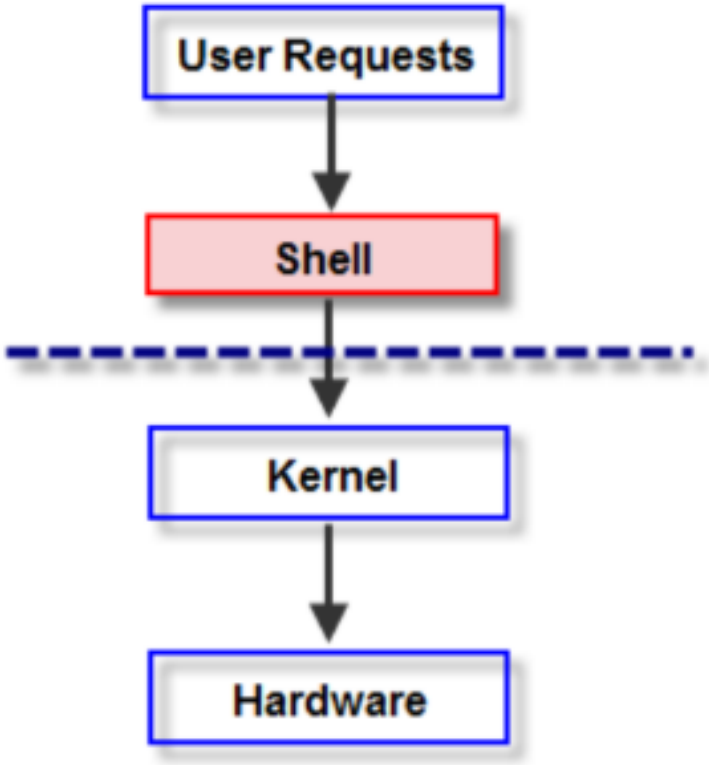
Step 2 – Bootstrap program directs the transfer of the operating system into main memory and then transfers control to it


User interface
Users
Library interface
Standards utility programs (shell, editors, compliers etc)
System call interface
Standard library (open, close, read, write, fork, etc)
Linux operating system (process management, memory management, the file system, I/O, etc)
Hardware (CPU, memory, disks, terminals, etc)
User mode
Kernel mode

SHELL - A program (a.k.a. command-line interpreter) that allows the user to interact with the UNIX/Linux system.

Examples: Bourne shell (sh), Bourne again shell (Bash), C shell (csh, tcsh), Korn shell (ksh), Powershell (windows)





Keep checking MOODLE !!!