Computer Organization and Assembly Language (COAL)

Lecture 1

Dr. Naveed Anwar Bhatti

Webpage: naveedanwarbhatti.github.io

Who am I? Dr. Naveed Anwar Bhatti

Hometown: Islamabad

Last Job:
Senior Researcher
RISE, Stockholm, Sweden
Joined on April, 2018
ERCIM Post-Doc (April, 2018 – Sep, 2019)

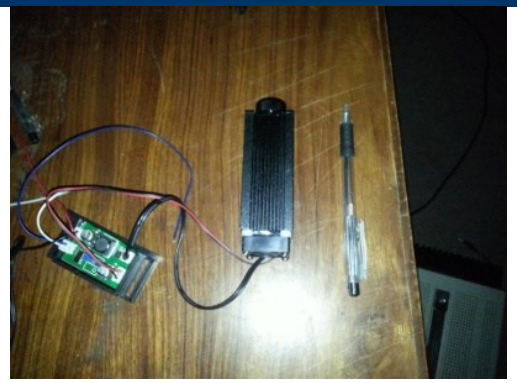
Education:

PhD 2018
Computer Science
Politecnico di Milano, Italy
System Support for Transiently Powered Embedded Systems

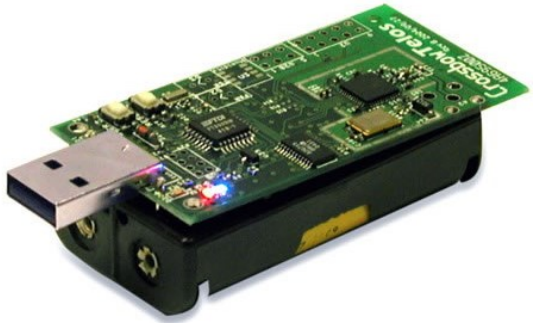
MS 2013
Computer Science
FAST-NUCES, Islamabad, Pakistan
Long range RFID System: Decoupling sensing and energy in sensor networks using energy transferenc

BS 2011
Telecom
FAST-NUCES, Islamabad, Pakistan
Internet Controlled Unmanned Ground Vehicle

Long range RFID-like System

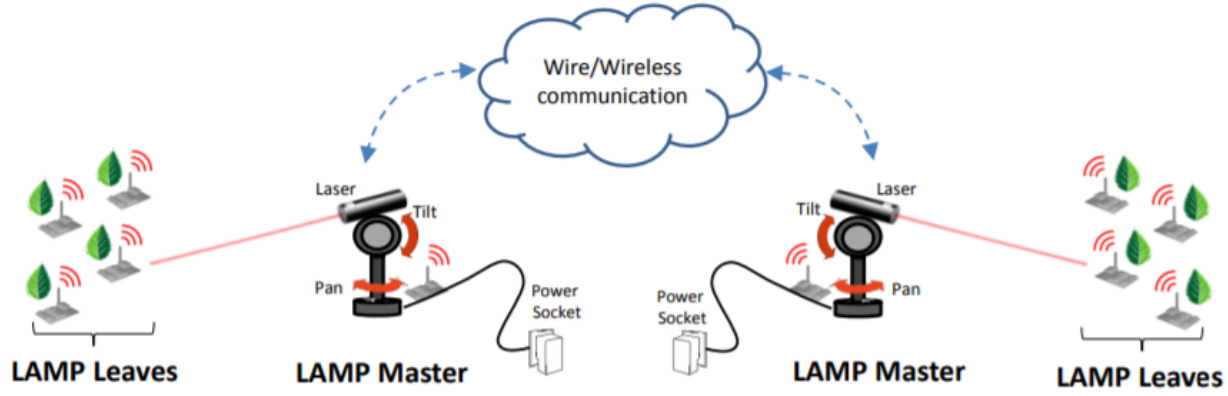


Laser Module



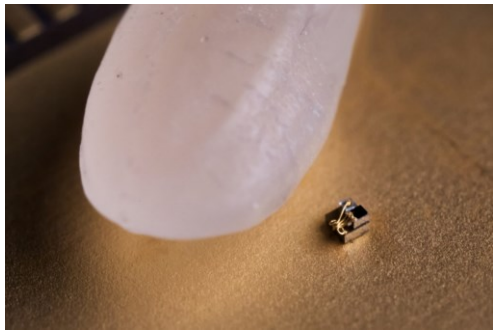
TelosB mote

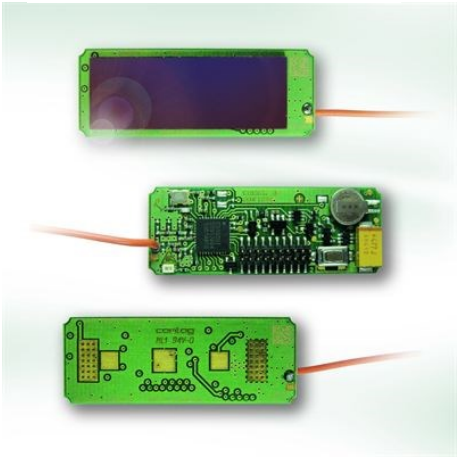
Solar Panel

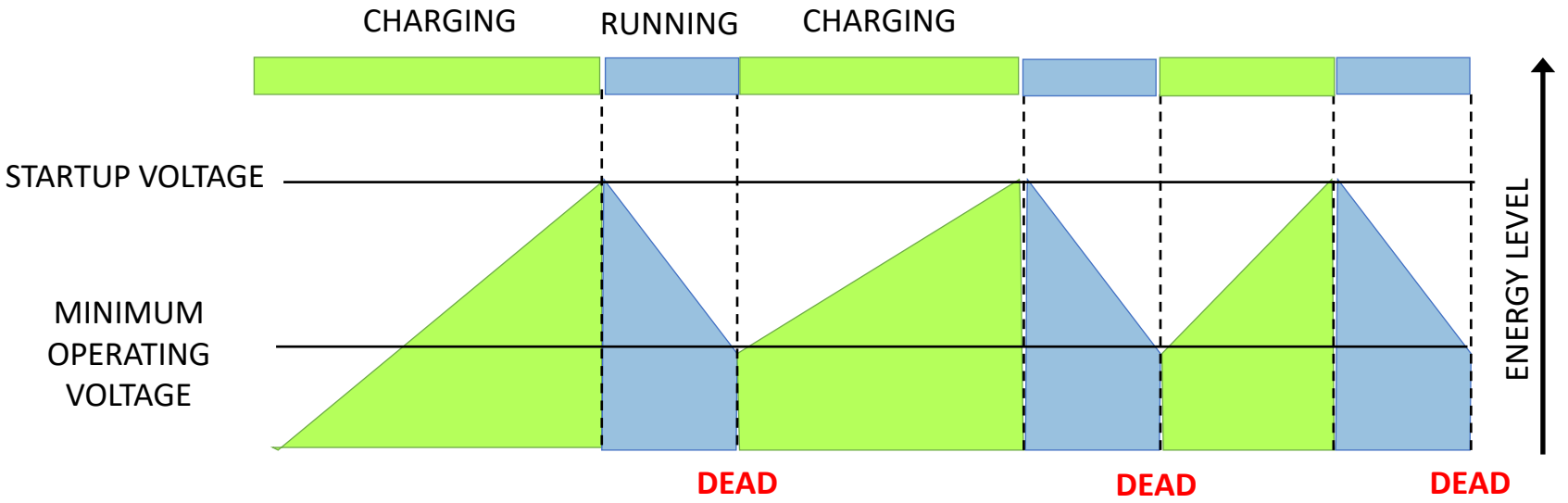


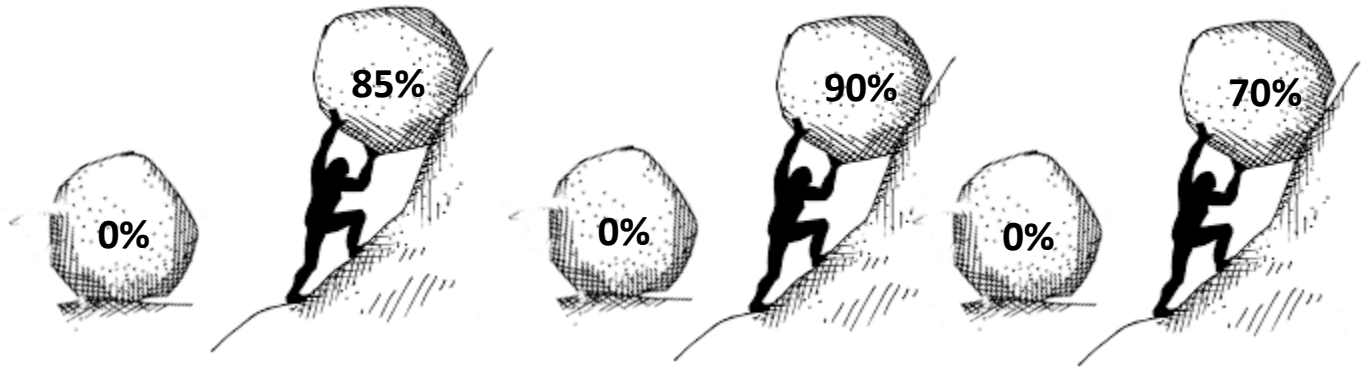
Long range RFID-like System

System Support for Transiently Powered Embedded Systems



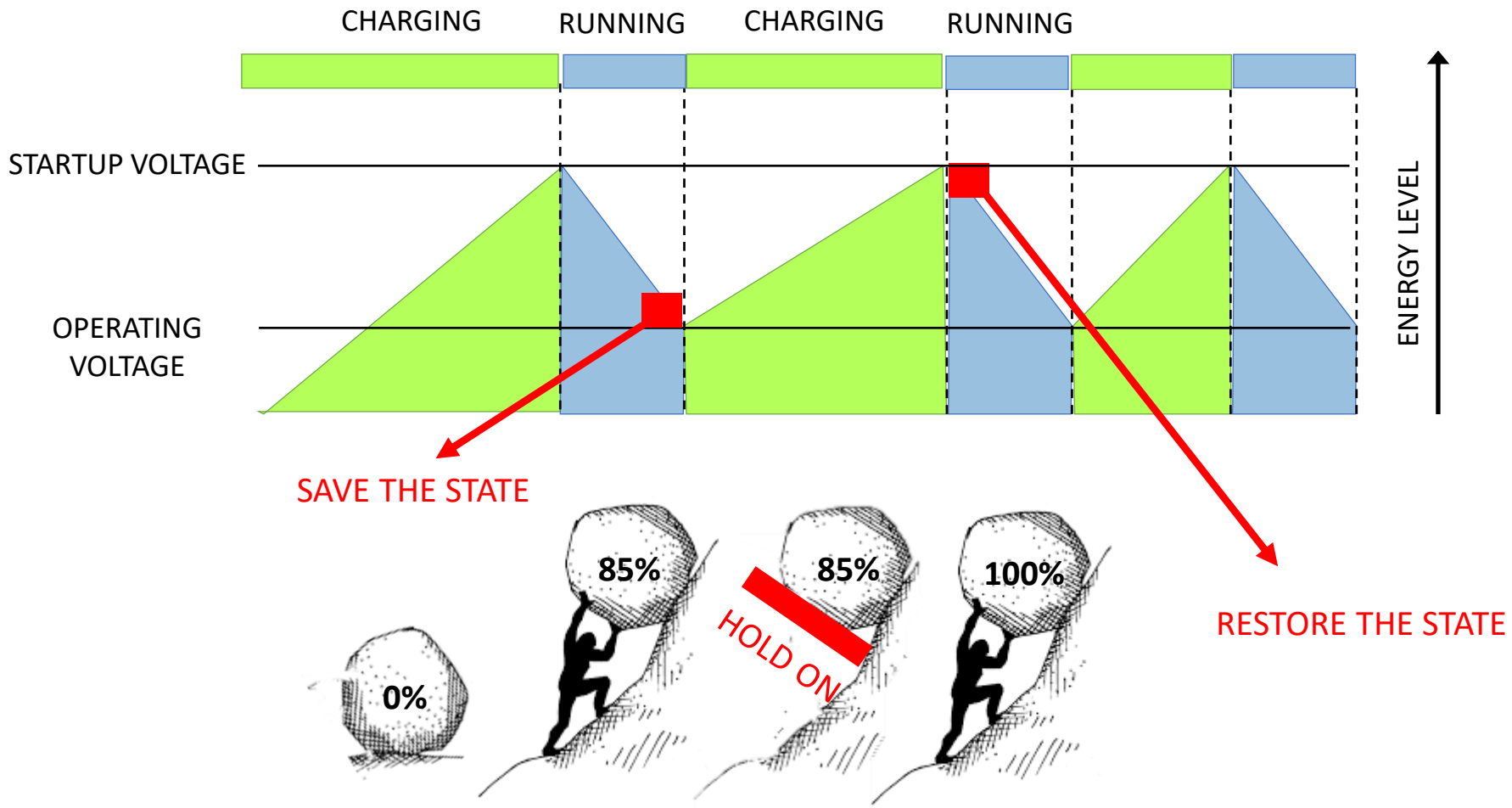




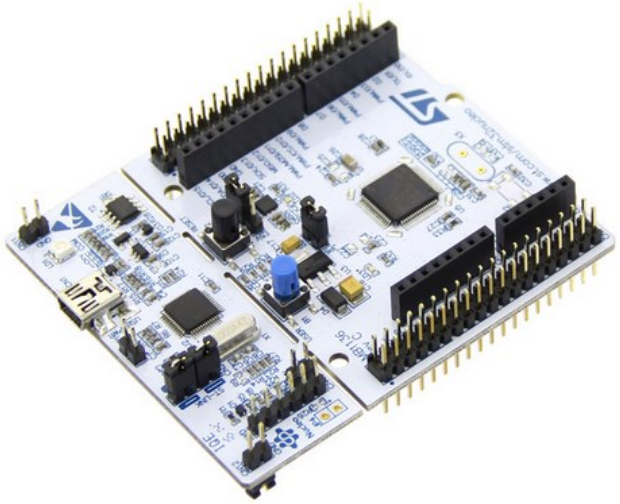


CHALLENGE: MAKE EMBEDDED DEVICES IMMUNE TO TRANSIENT POWER ENVIRONMENT

System Support for Transiently Powered Embedded Systems

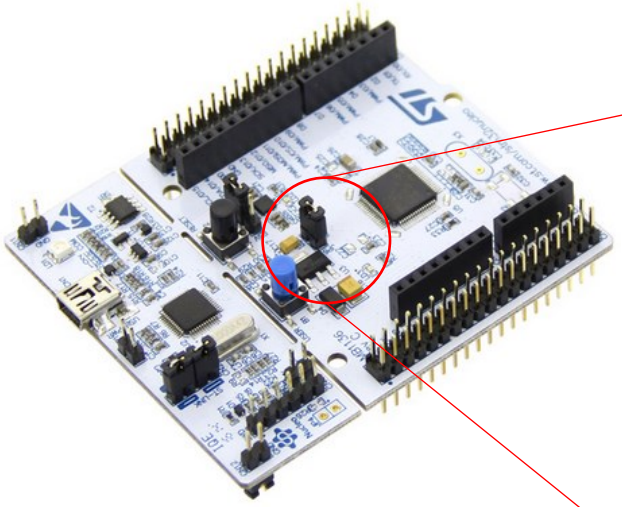


System Support for Transiently Powered Embedded Systems



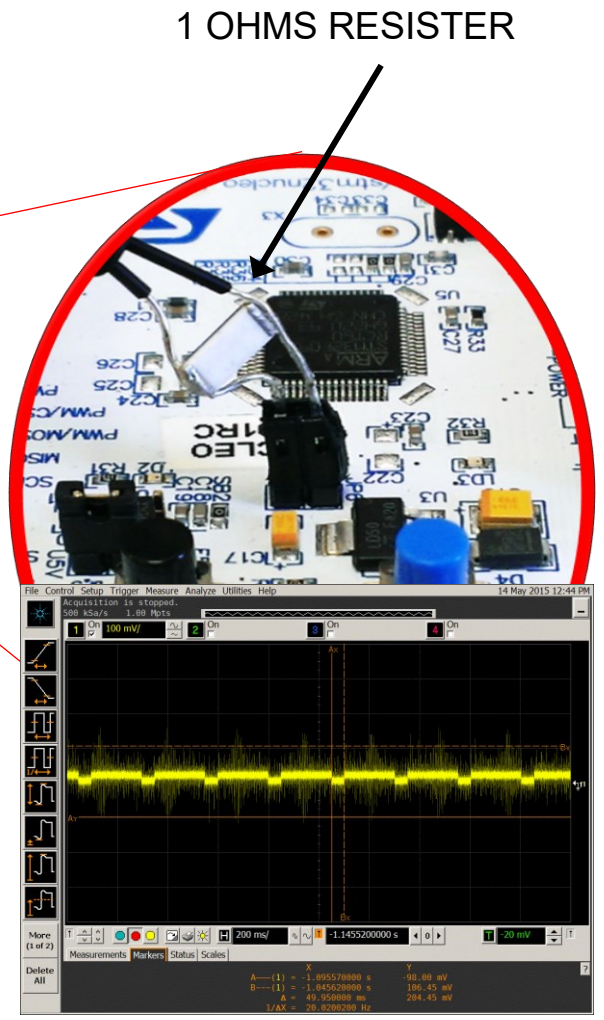
STM32 NUCLEO L152RE

ARM®32-bit Cortex®-M3 CPU
32 MHz max CPU frequency
512 KB Flash
80 KB SRAM

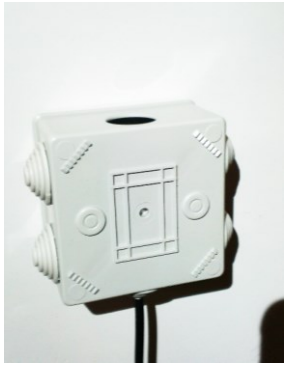


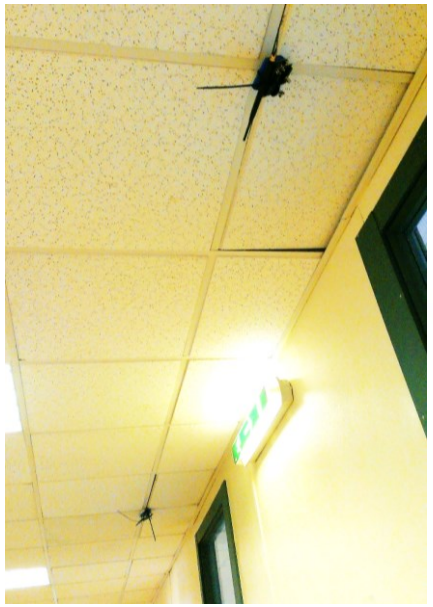
STM32 NUCLEO 91RC

ARM®32-bit Cortex®-M0 CPU
48 MHz max CPU frequency
256 KB Flash
32 KB SRAM



Other Sensor Deployments





Waspmote

How to reach me?

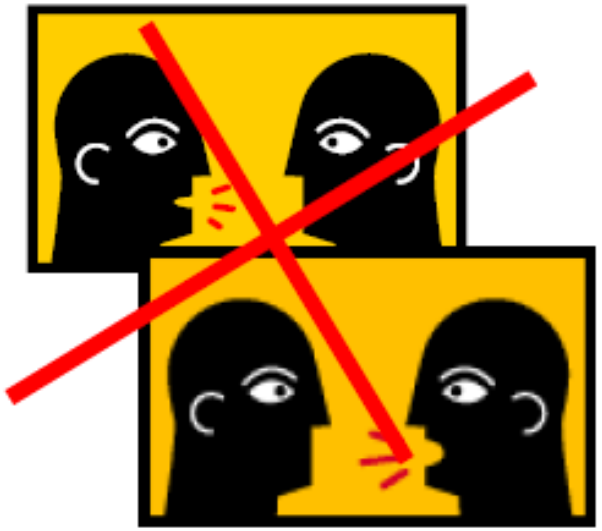
Email: naveed.bhatti@mail.au.edu.pk
Webpage: naveedanwarbhatti.github.io

Class page and slides: **zupq557** (Google classroom)

Administrivia and more

- **Grading split**
  - Assignments: 10%
  - Quizzes: 10%
  - Mid-Term Exam: 35%
  - Final Exam: 45%

Prohibitions



Class Attendance and Late arrival

- **University and HEC cares about it**
  - **I do not !**
  - I shall say you are present as long as you tell me before class
  - If you are not serious about the course, its your loss
    - Both money wise
    - And grade wise (directly: 10% participation, quizes indirectly: exams)

- **If you arrive late**
  - Be discrete (come in with minimal fanfare)
  - Be courteous (to other students trying to listen)

Computer Organization and Assembly Language

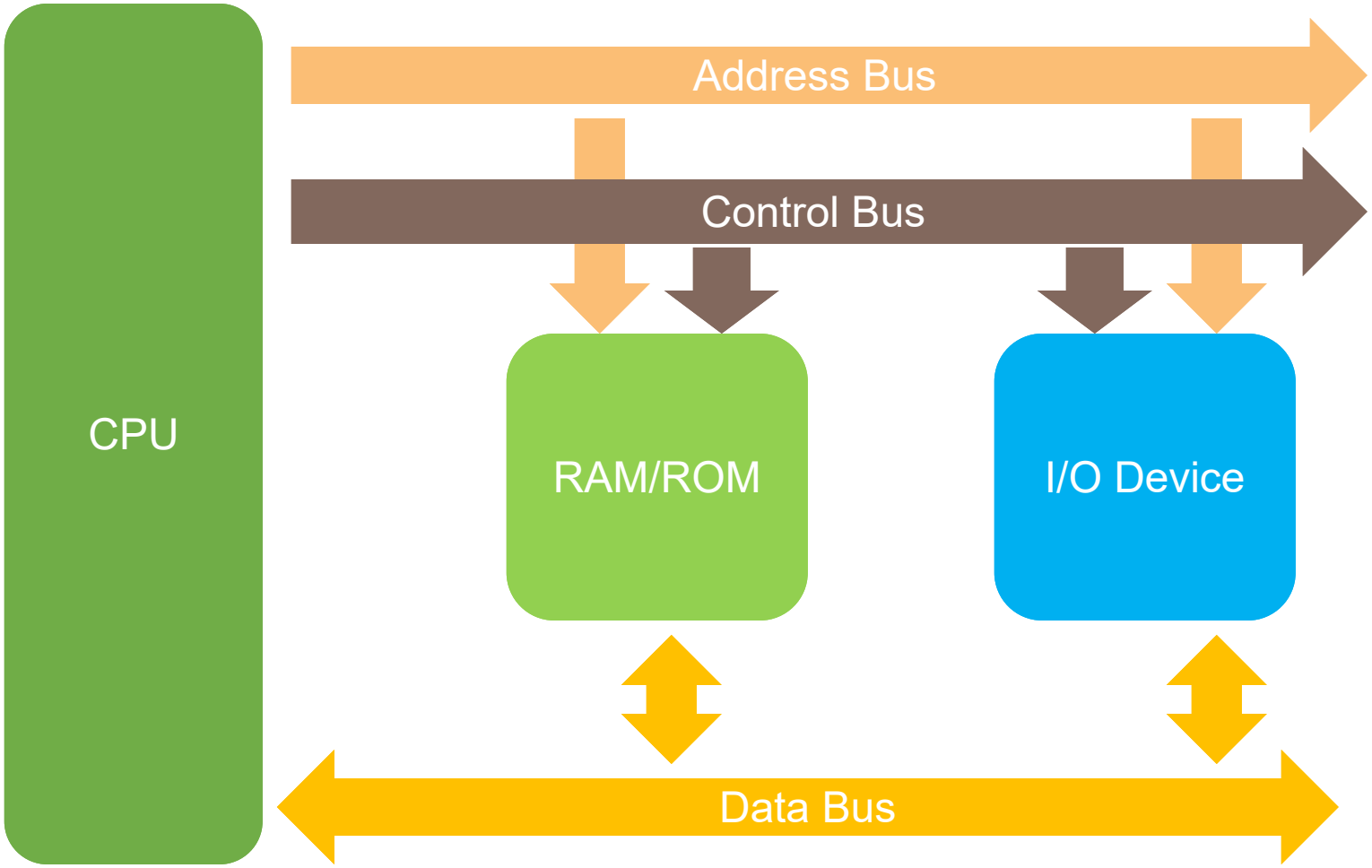
- Computer Organization vs Computer Architecture
- What is Assembly Language?
- Why Learn Assembly Language?

Computer System

Components

- CPU
- Memory
- I/O Devices
- Buses

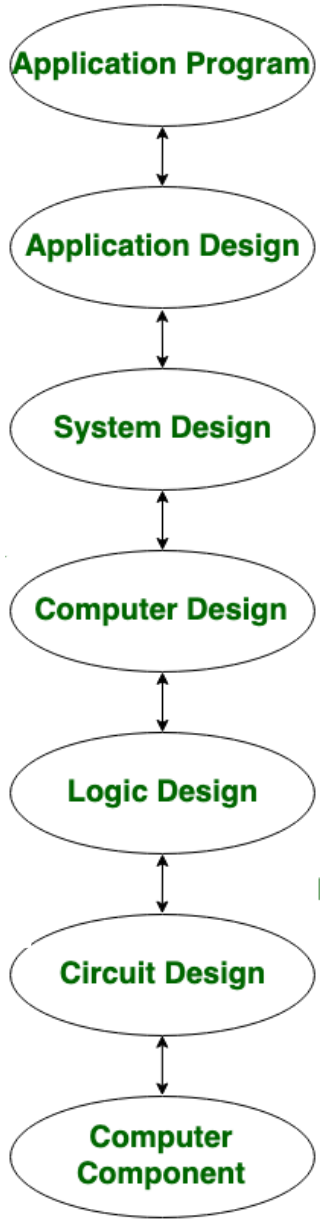
System Design



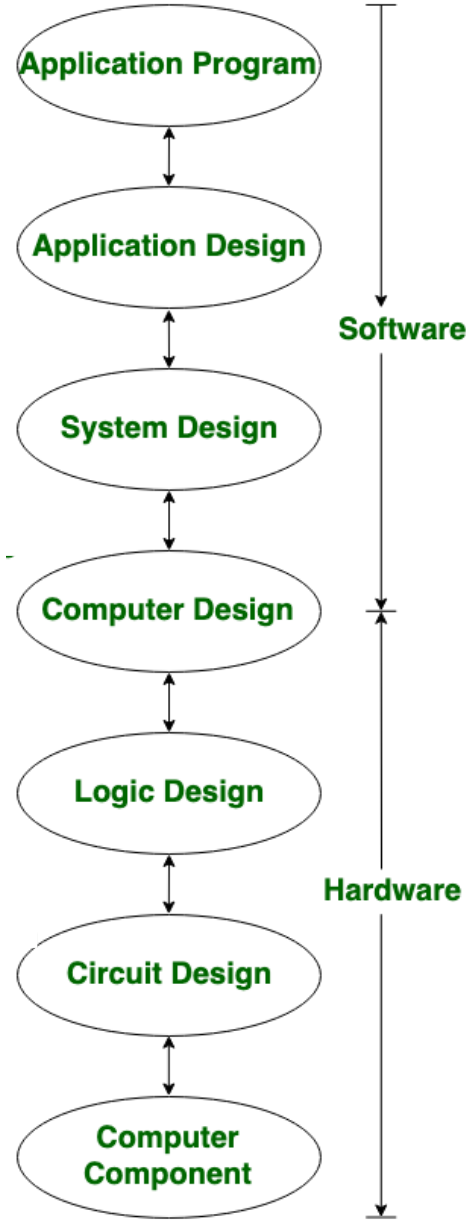
High-level Software

- Application Program
- Application Design

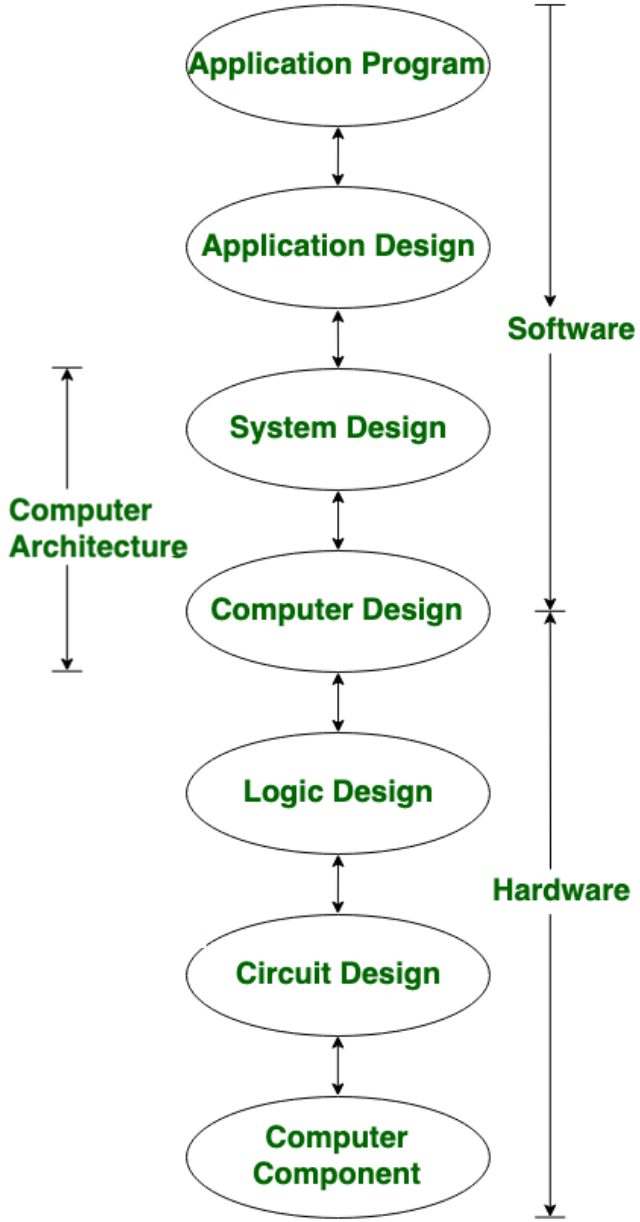
Computer System



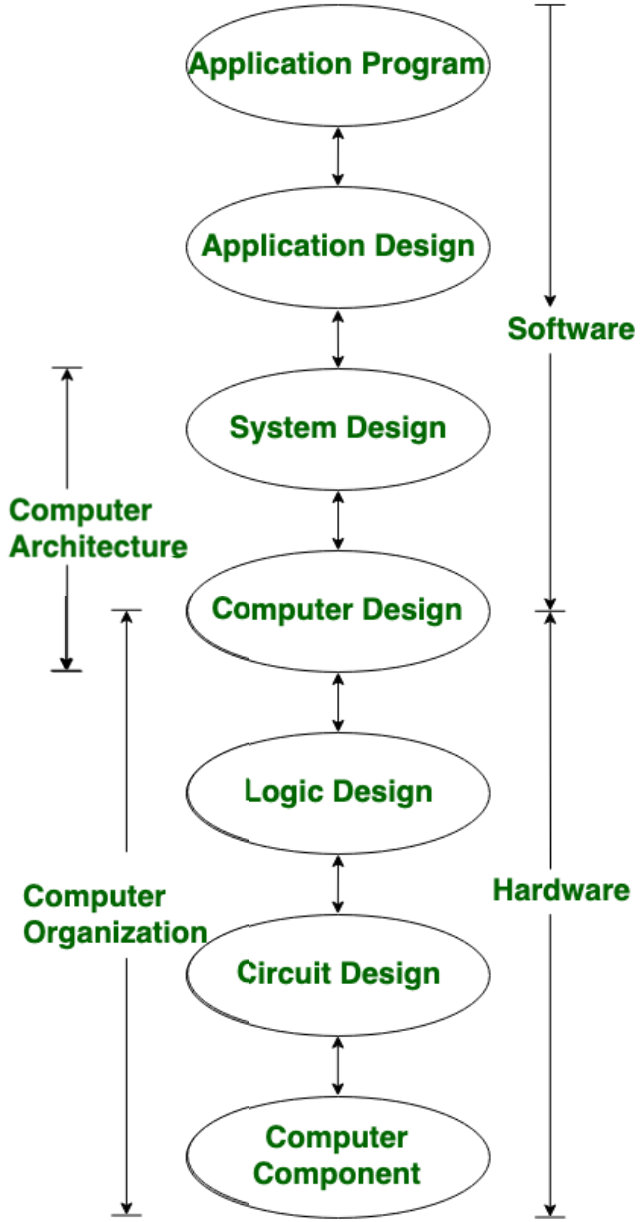
Computer System



Computer System



Computer System

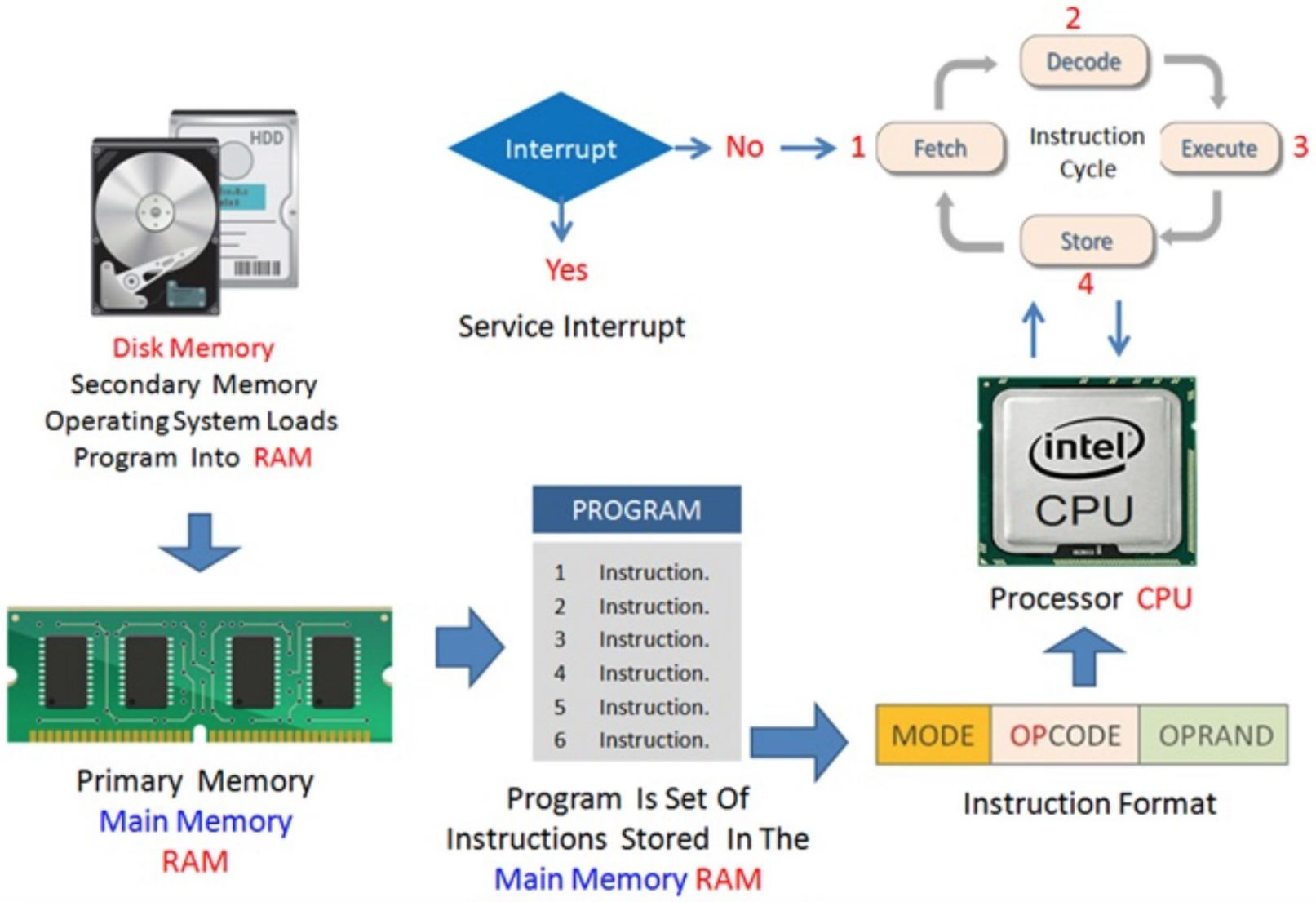


Computer Organization vs Computer Architecture

Architecture	Organization
What the computer does	**How** it does it
Deals with the **functional behavior**	Deals with a **structural relationship**
Deals with **high-level design** issues	Deals with **low-level design** issues.

"The implementation of the architecture is called organization"

Computer Organization

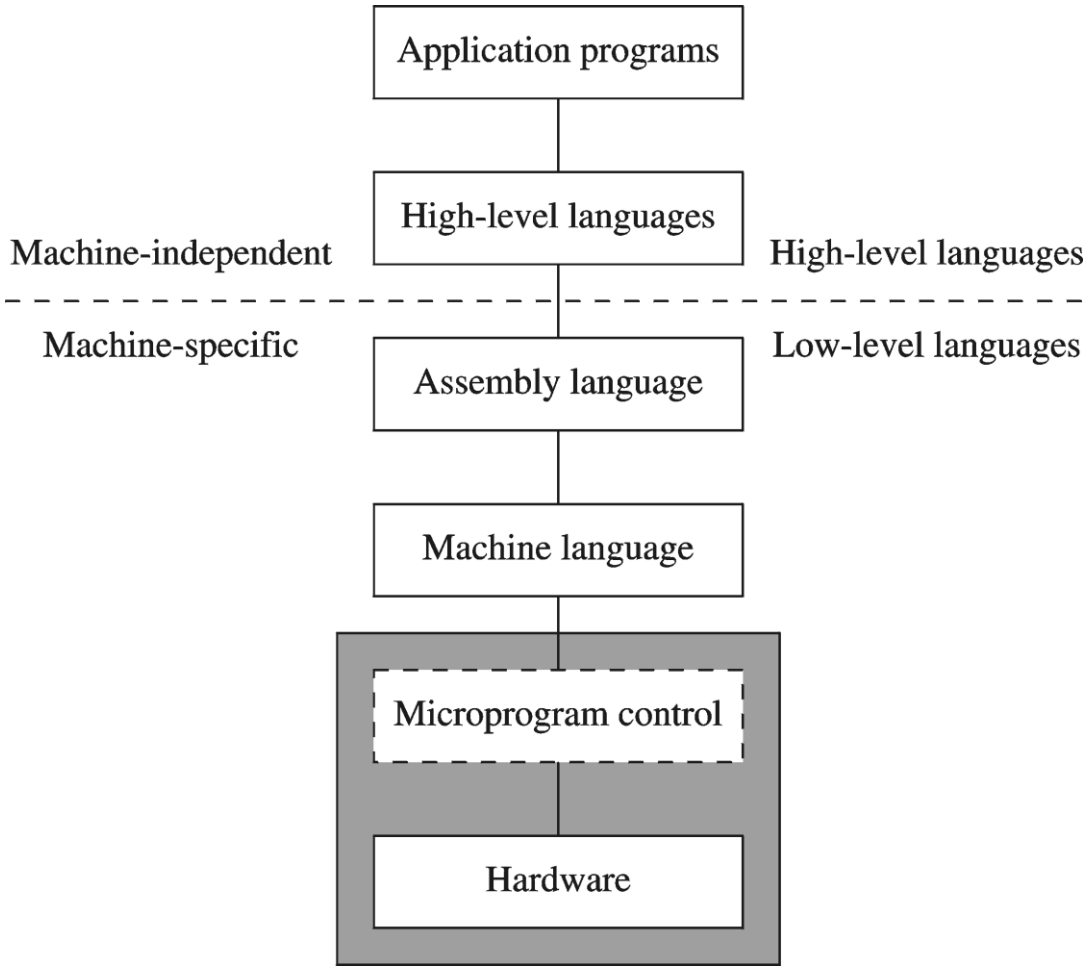


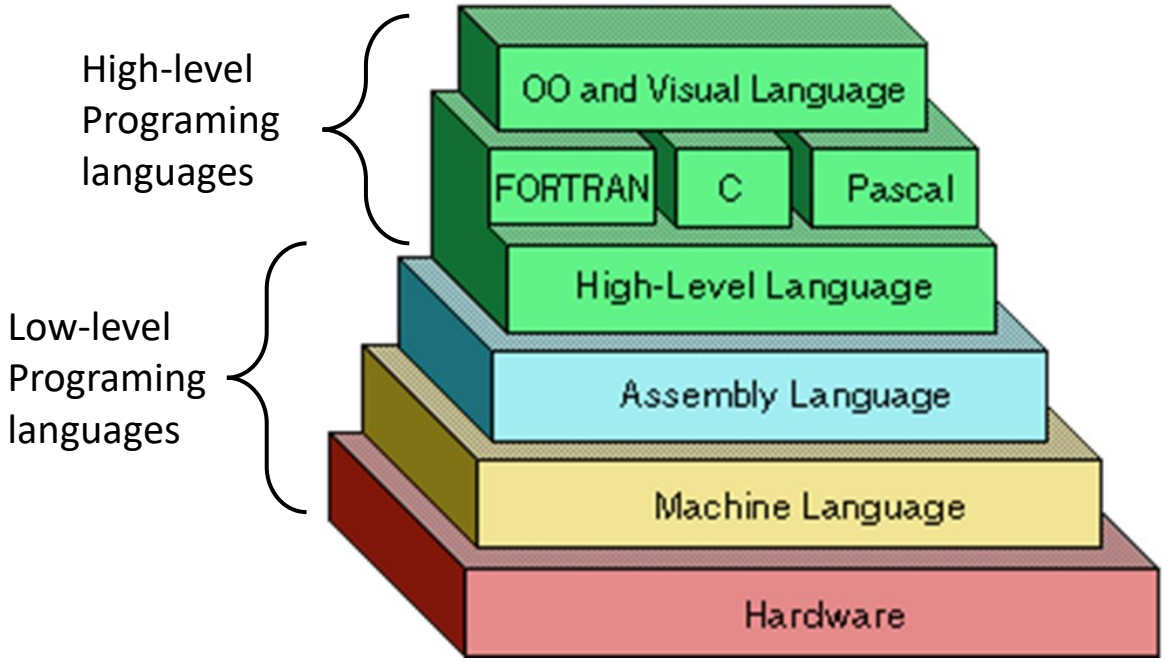
https://www.learncomputerscienceonline.com/computer-organization-and-architecture/

Some questions to ask?

- What is Assembly Language?
- Why Learn Assembly Language?
- What is Machine Language?
- How is Assembly related to Machine Language?
- What is an **Assembler** (Compiler vs Assembler vs Linker)?
- How is Assembly related to High-Level Language?
- Is Assembly Language portable?

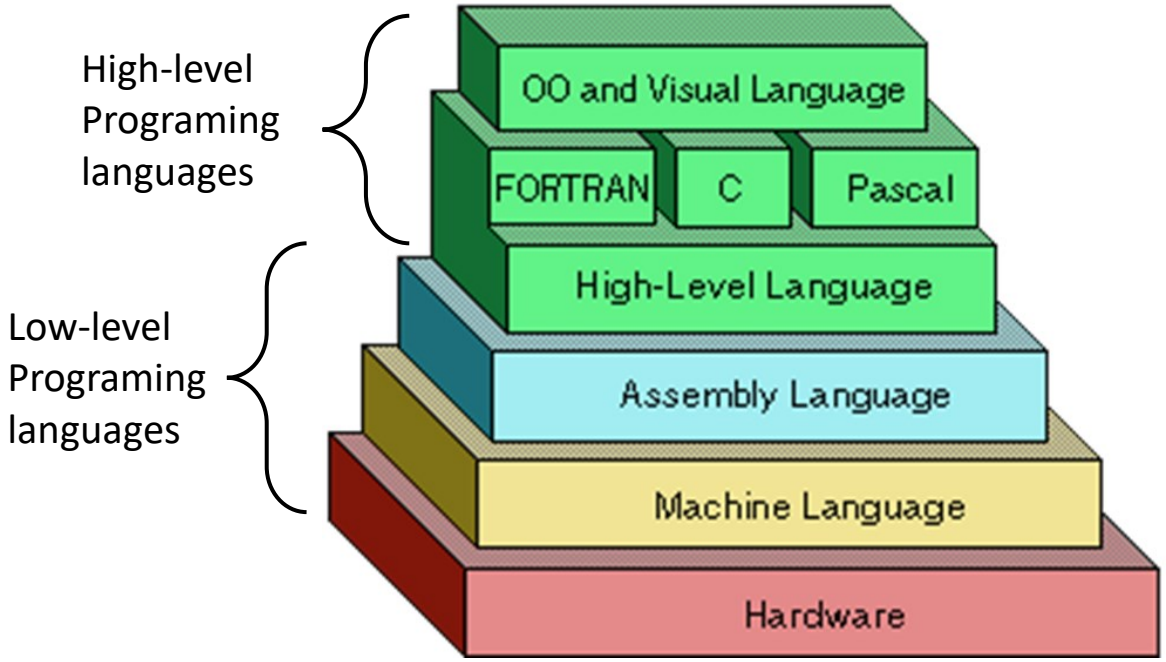
A Hierarchy of Languages





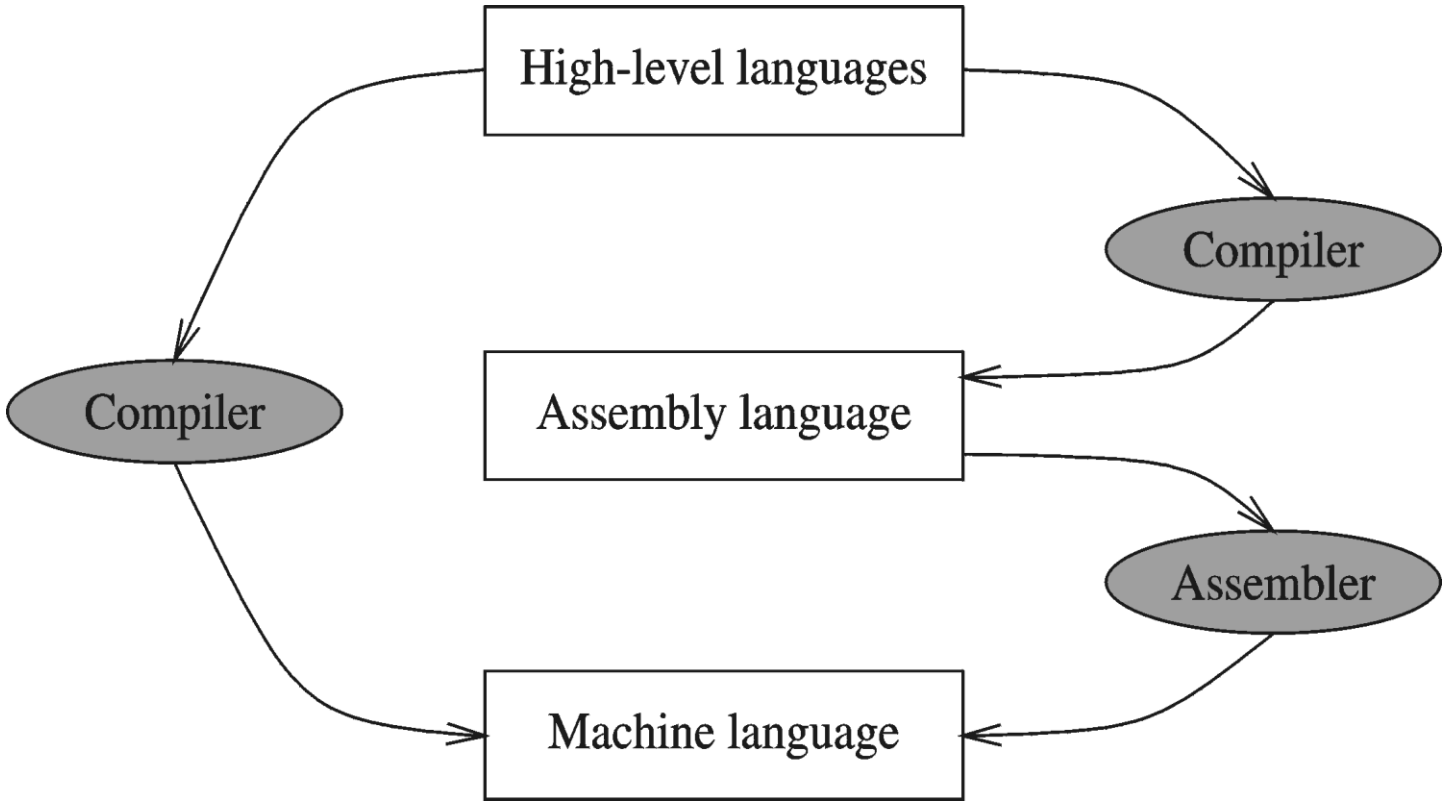
A Hierarchy of Languages

- Machine language
  - Native to a processor: executed directly by hardware
  - Instructions consist of binary code: 1s and 0s
- Assembly language
  - Slightly higher-level language
  - Readability of instructions is better than machine language
  - One-to-one correspondence with machine language instructions



Compiler and Assembler

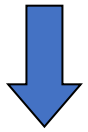
- Assemblers translate assembly to machine code
- Compilers translate high-level programs to machine code
  - Either directly, or
  - Indirectly via an assembler



Translating Languages

English: D is assigned the sum of A times B plus 10.

High-Level Language: D = A * B + 10



A statement in a high-level language is translated typically into several low-level instructions

Intel Assembly Language:

```
mov   eax, A
mul   B
add   eax, 10
mov   D, eax
```

Intel Machine Language:

```
A1 00404000
F7 25 00404004
83 C0 0A
A3 00404008
```

Advantages of High-Level Languages

- Program development is faster
  - High-level statements: fewer instructions to code
- Program maintenance is easier
  - For the same above reasons
- Programs are portable
  - Contain few machine-dependent details
    - Can be used with little or no modifications on different machines
  - Compiler translates to the target machine language
  - However, Assembly language programs are not portable

Why Learn Assembly Language?

- Two main reasons:
  - Accessibility to system hardware
  - Space and time efficiency

- **Accessibility to system hardware**
  - Assembly Language is useful for implementing system software (drivers)
  - Also useful for small embedded system applications

- **Space and Time efficiency**
  - Understanding sources of program inefficiency
  - Tuning program performance
  - Writing compact code

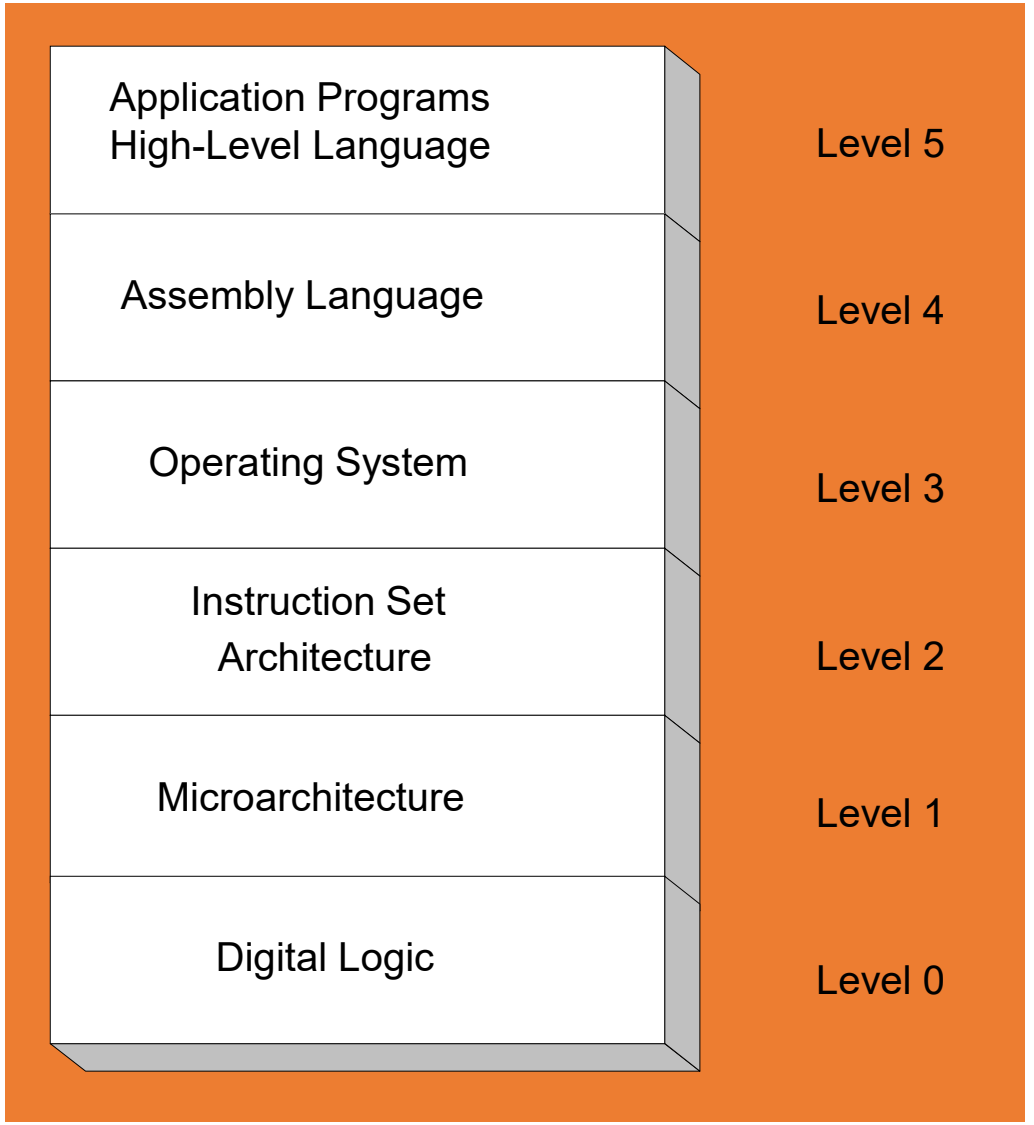
Assembly vs High-Level Languages

Some representative types of applications:

Type of Application	High-Level Languages	Assembly Language
Business application software, written for single platform, medium to large size.	Formal structures make it easy to organize and maintain large sections of code.	Minimal formal structure, so one must be imposed by programmers who have varying levels of experience. This leads to difficulties maintaining existing code.
Hardware device driver.	Language may not provide for direct hardware access. Even if it does, awkward coding techniques must often be used, resulting in maintenance difficulties.	Hardware access is straightforward and simple. Easy to maintain when programs are short and well documented.
Business application written for multiple platforms (different operating systems).	Usually very portable. The source code can be recompiled on each target operating system with minimal changes.	Must be recoded separately for each platform, often using an assembler with a different syntax. Difficult to maintain.
Embedded systems and computer games requiring direct hardware access.	Produces too much executable code, and may not run efficiently.	Ideal, because the executable code is small and runs quickly.

Programmer's View of a Computer System

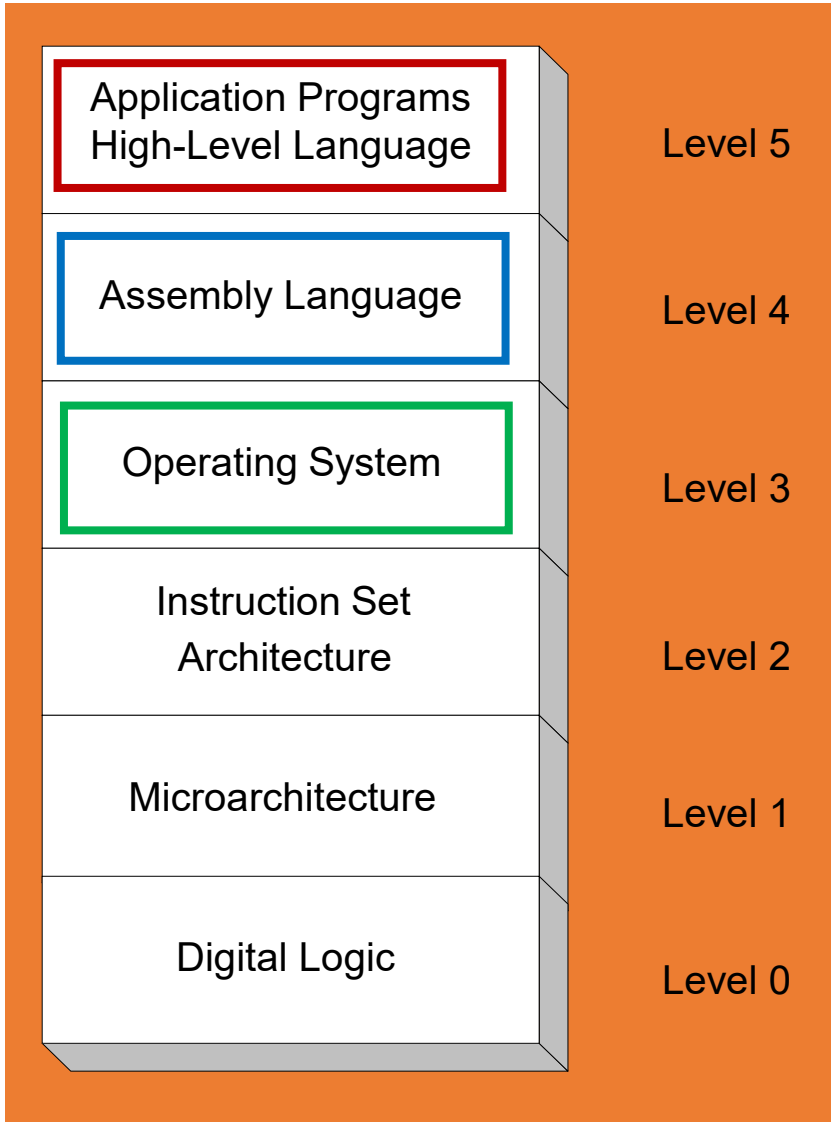
Increased level of abstraction



Each level hides the details of the level below it

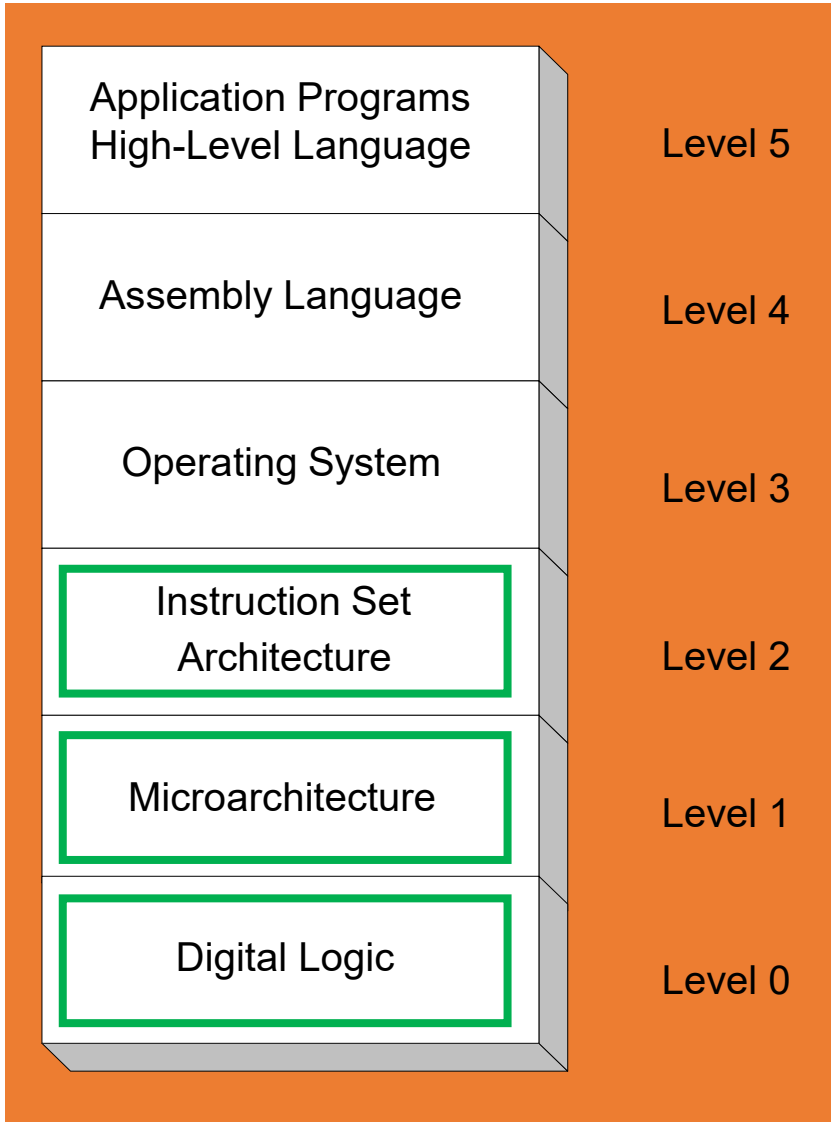
Programmer's View – 2

- Application Programs (Level 5)
  - Written in high-level programming languages
  - Such as Java, C++, Pascal, Visual Basic . . .
  - Programs compile into assembly language level (Level 4)
- Assembly Language (Level 4)
  - Instruction mnemonics are used
  - One-to-one correspondence to machine language
  - Calls functions written at the operating system level (Level 3)
  - Programs are translated into machine language (Level 2)
- Operating System (Level 3)
  - Provides services to level 4 and 5 programs
  - Translated to run at the machine instruction level (Level 2)



Programmer's View – 3

- Instruction Set Architecture (Level 2)
  - Specifies how a processor functions
  - Machine instructions, registers, and memory are exposed
  - Machine language is executed by Level 1 (microarchitecture)
- Microarchitecture (Level 1)
  - Controls the execution of machine instructions (Level 2)
  - Implemented by digital logic (Level 0)
- Digital Logic (Level 0)
  - Implements the microarchitecture
  - Uses digital logic gates
  - Logic gates are implemented using transistors



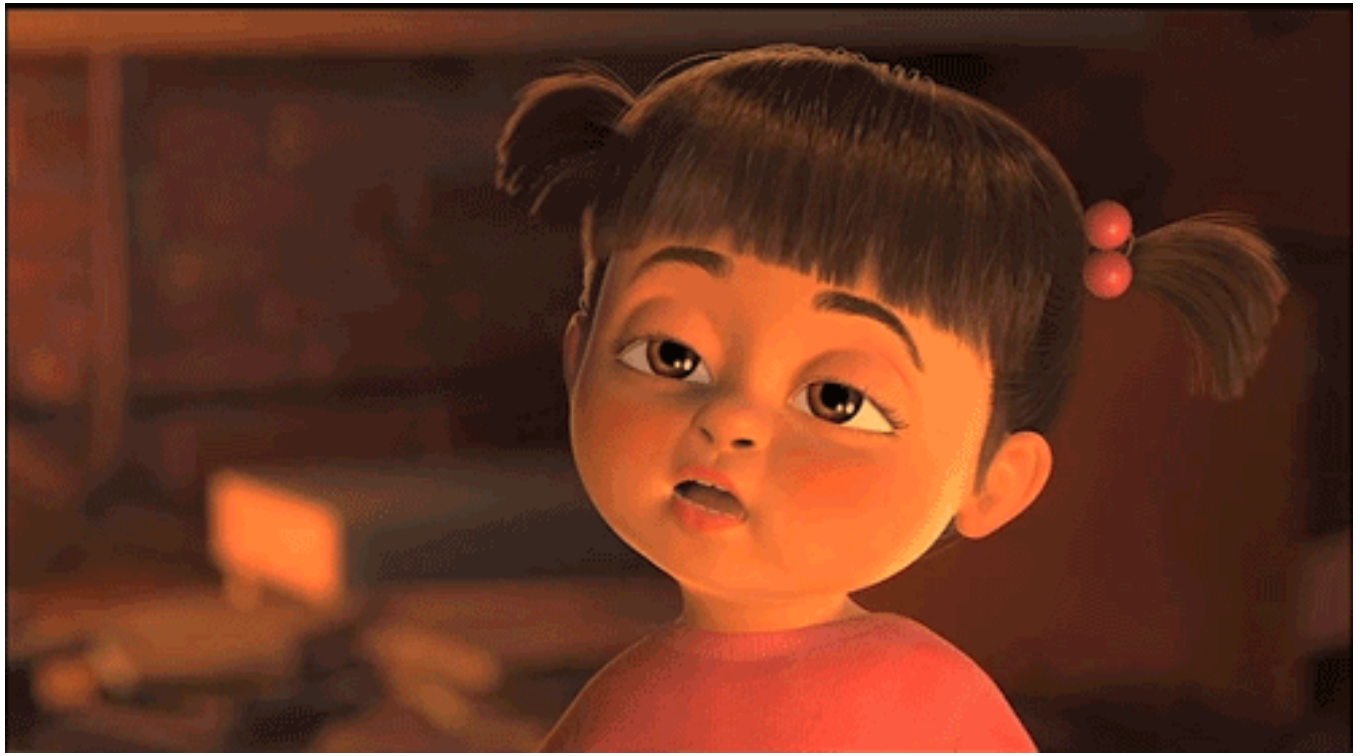
Next Time

- Data Representation
- Boolean Operations

Summary

- Assembly language helps you learn **how software is constructed at the lowest levels**
- Assembly language has a **one-to-one relationship with machine language**
- An **assembler** is a program that **converts assembly language programs into machine language**
- A **linker combines individual files created by an assembler into a single executable file**
- A computer system can be viewed as consisting of layers. Programs at one layer are translated or interpreted by the next lower-level layer

Thanks a lot



If you are taking a Nap, **wake up**.....Lecture Over

Acknowledgment and References

- Most of the slides *are borrowed from*
  - *Umme Hani previously taught course*