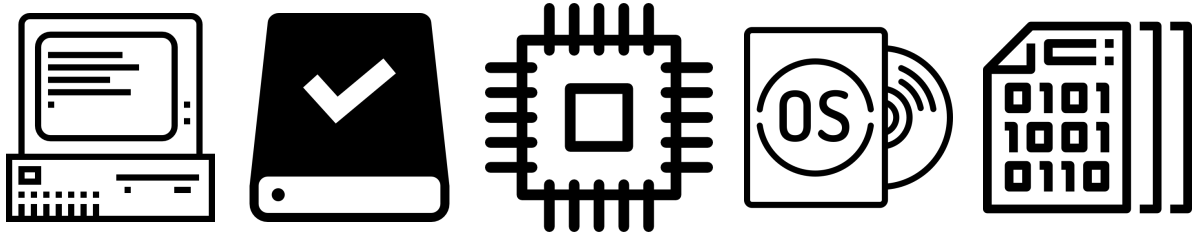
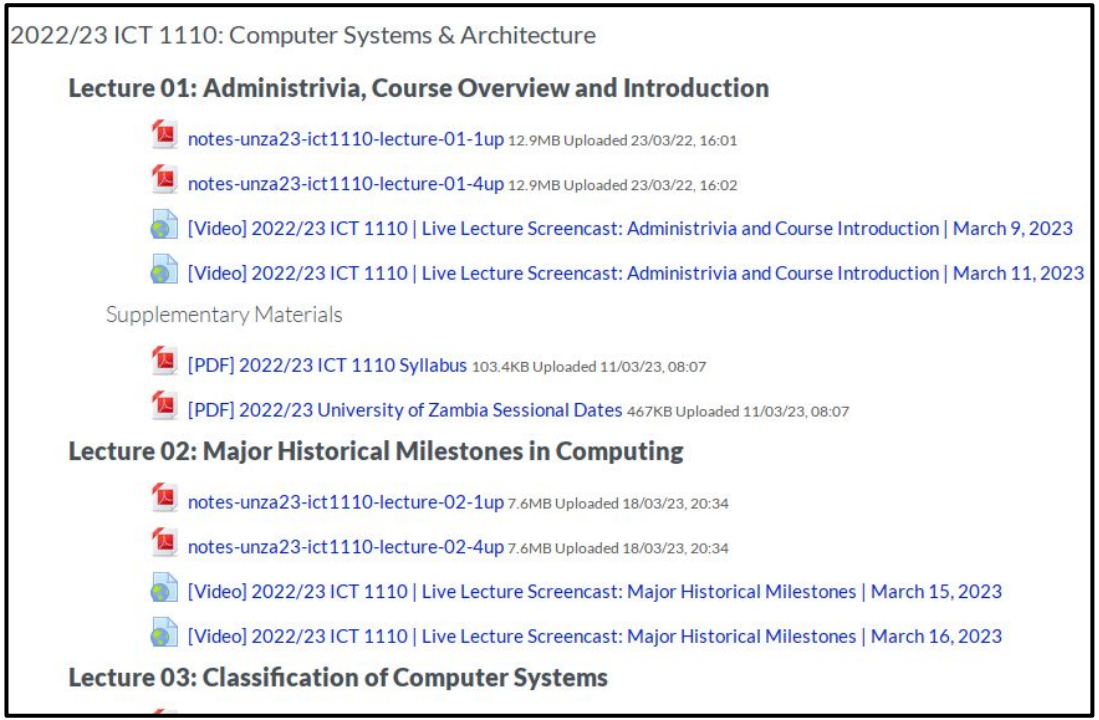


ICT 1110 (2022/23)
Computer Systems & Architecture
Lecture 4: Abstraction in Computer Systems

Lighton Phiri <lighton.phiri@unza.zm>
Department of Library & Information Science
University of Zambia
https://bit.ly/3uJBZRc

Announcements—March 25, 2023

- **The Moodle site is available**
  - Course notes and assessment scores available
- **Scheduled assessments**
  - Quiz #02: March 30, 2023
  - Classification of Computer Systems



Microcomputer (Personal Computer)

Lecture Outline

- Recap of ICT 1110 Topics
- Introduction
- Computer Abstraction
- Examples of Abstraction
- Summary

Recap of ICT 1110 Topics (1/2)

#	Topic/Theme	~Hours
1.	Introduction	1
2.	Major Historical Milestones	1
3.	Classification of Computer Systems	1
4.	Abstraction in Computing	1
5.	Computer Software	13
6.	Machine Structure	1
7.	Central Processing Unit	3
8.	I/O Subsystem	6

Recap of ICT 1110 Topics (2/2)

#	Topic/Theme	~Hours
9.	Computer Memory	16
10.	Number Systems	8
11.	Data Encoding	7
12.	MIPS ISA	24
13.	Digital Logic Structures	5

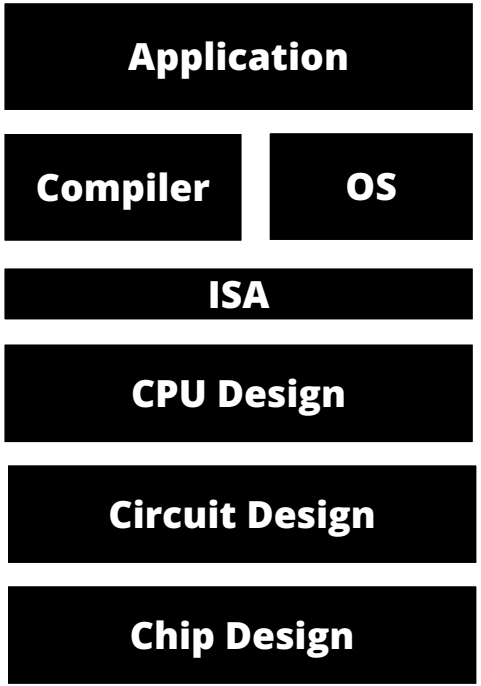
Introduction (1/2)

- **A computer system is any electronic device with the ability to accept input, process data, store data and produce output**
- **In actual fact, a computer is a complex devices comprising of hardware and software**
  - We abstract the complex nature of computer systems in order to gain an in-depth understand of how they function

Microcomputer (Personal Computer)

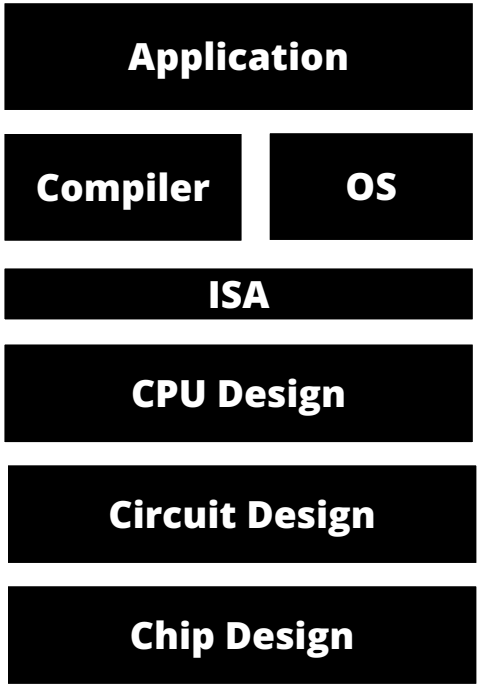
Introduction (2/2)

- **The layers of abstraction we will focus on in ICT 1110**
  - Software layer—different ways through which software interacts with hardware
  - High-level functional units—interaction of functional units during instruction execution
  - Instruction Set Architecture—basic building blocks of operations used to build software
  - Datapath and Control—instruction execution path on the CPU



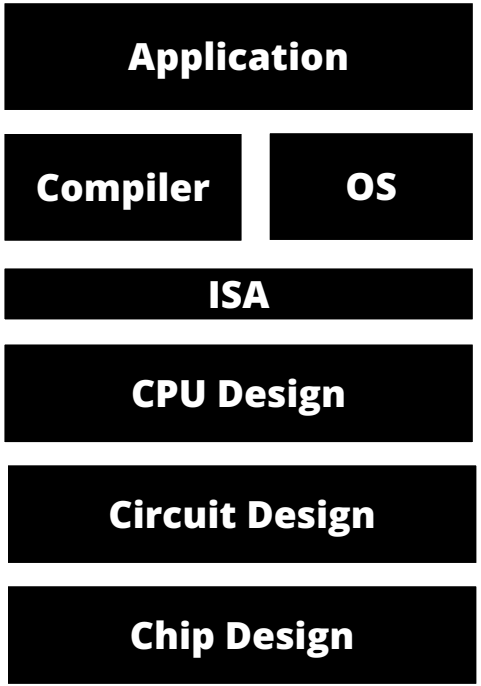
Computer Abstraction (1/4)

- **A top-down approach, beginning with highly abstracted view of the computer is helpful in understanding how computers work**
- **Complexity of computer systems require that we abstract certain view.**
  - e.g. Von Neumann Architecture.
- **Abstraction enables us cope with complexity by omitting unnecessary elements.**



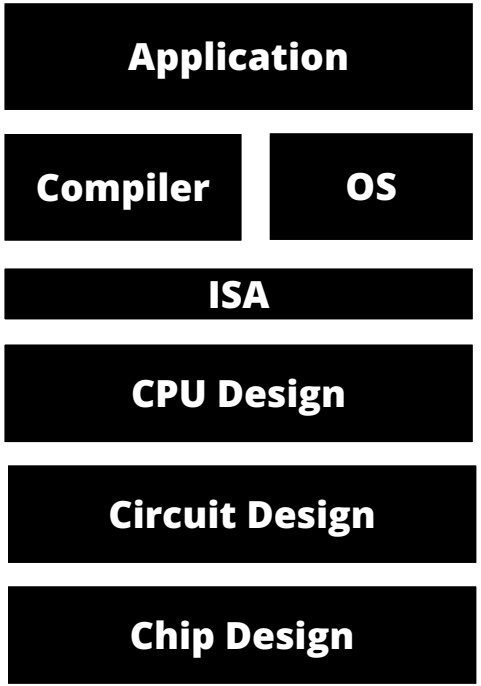
Computer Abstraction (2/4)

- **Abstraction is the process of omitting unnecessary detail in order to simply and focus on much needed attention.**
  - During abstraction, details that are not necessary are removed to simply an entity
  - During abstraction, some object properties are left out in order to focus on the more important ones
    - E.g. Von Neumann architecture and FETCH-DECODE-EXECUTE cycle



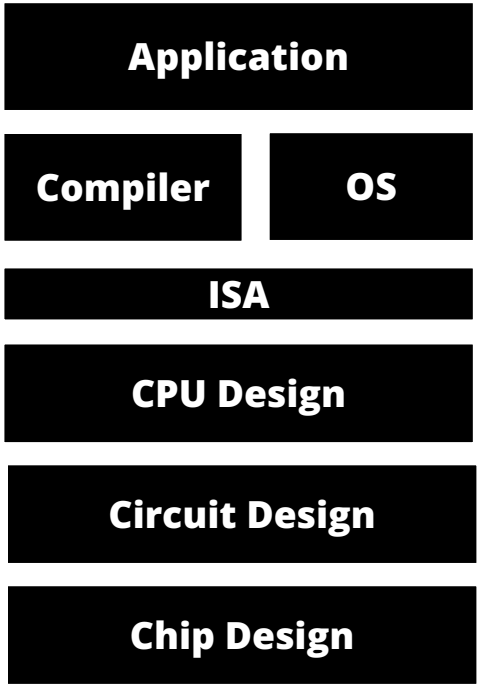
Computer Abstraction (3/4)

- **Abstraction allows for the generalisation of concepts.**
  - General concepts and common features can easily be identified from specific examples
    - For instance different programming languages, different types of computers
  - The abstracted view only comprises essential elements required to understand the layer of abstraction
    - Essential elements are a generalisation of the specific details



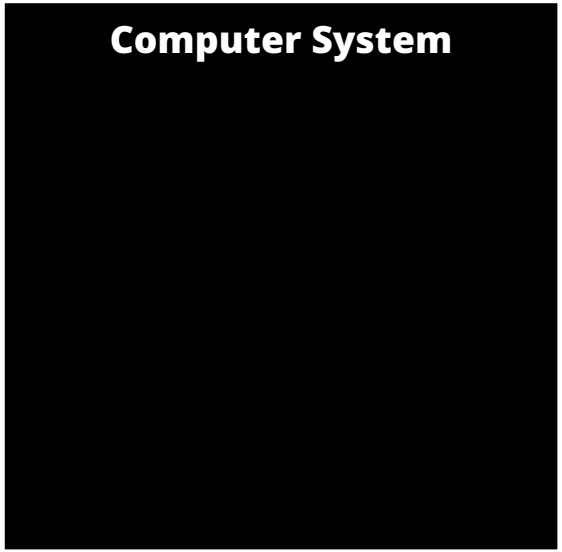
Computer Abstraction (4/4)

- **Advantages of abstraction in computing**
  - Focus on more important details, ignoring any irrelevant information
  - Focus on what is required to understand the concepts
  - Easier to handle complexity by excluding irrelevant details
  - Computing concepts are easier to understand



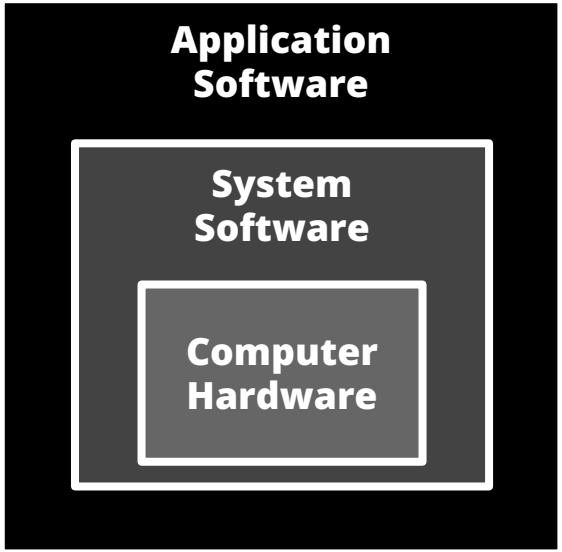
Computer High-level Abstraction (1/2)

- **A computer system is any electronic device with the ability to accept input, process data, store data and produce output**
  - Our current high-level overview has focused on what a computer does
    - Input, processing and output
  - Scenarios and use cases that require the use of computers



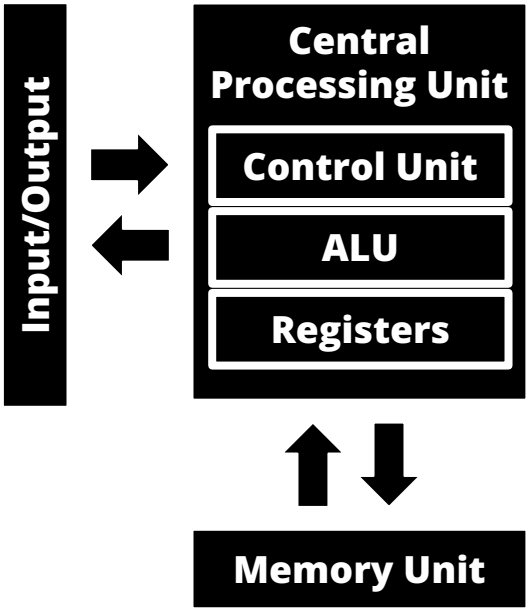
Computer High-level Abstraction (2/2)

- **In order to function successfully, a computer system leverages both hardware and software**
  - The software directs the hardware by specifying instructions to be executed
- **Hardware functionality is determined by the controlled flow of electricity**
  - Flow of electricity can be abstracted into a mathematical representation of two binary values: 1—flow—and 0—no flow



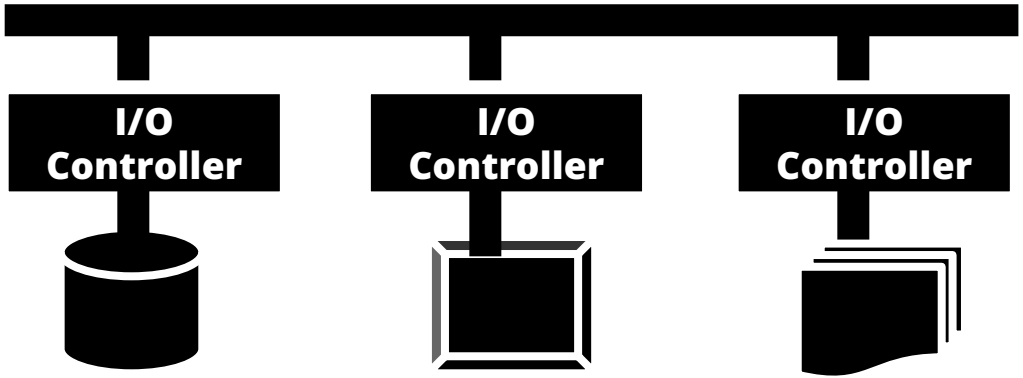
Hardware Abstraction (1/2)

- **Components of Von Neumann Architecture.**
  - Input Module (**Input**)
  - Output Module (**Output**)
  - Processing Module (**Process**)
    - Arithmetic Logic Unit
    - Control Unit
    - Registers
- **Memory Unit (Store)**
- **:**
- **Buses**



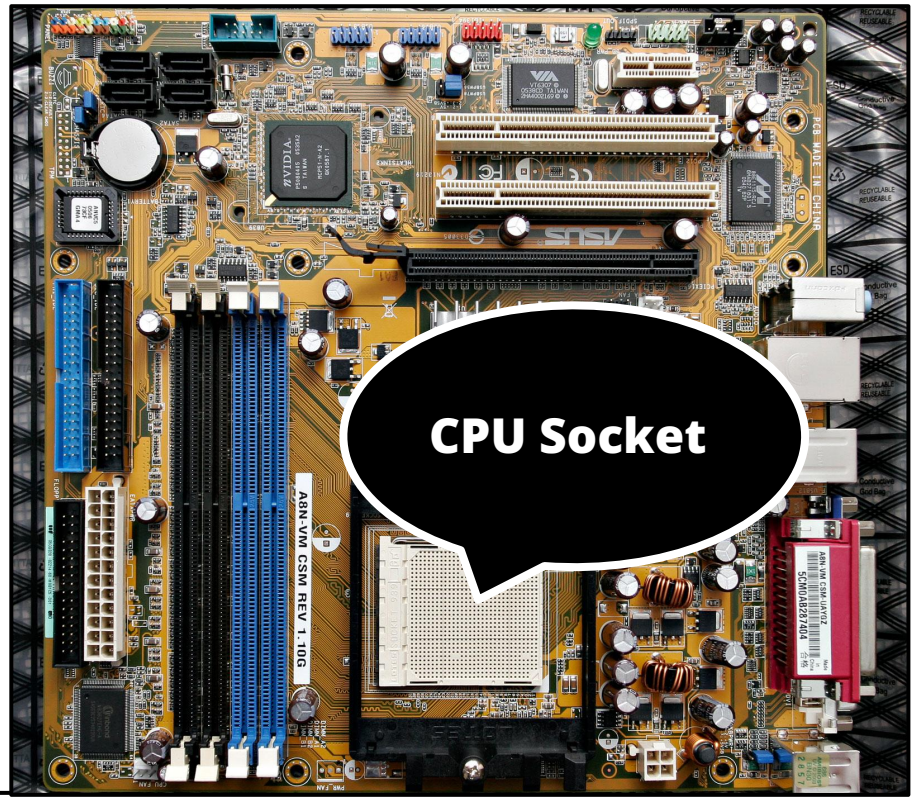
Hardware Abstraction (1/2)

- I/O subsystem is composed of a complex ecosystem that facilitates the interaction of peripherals and the CPU



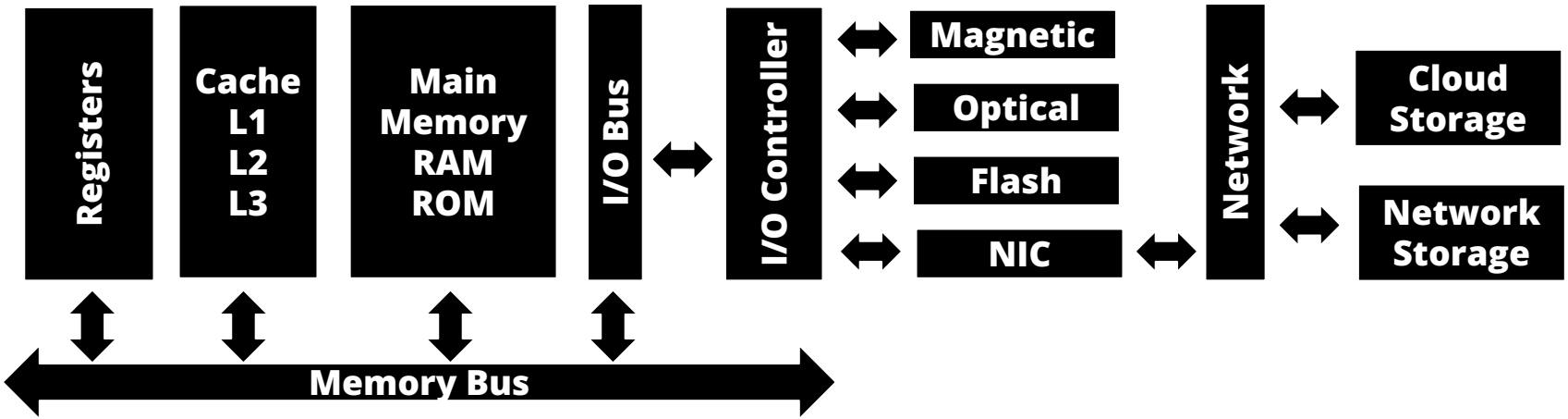
Hardware Abstraction (1/2)

Machine cycle occurs—for the most part—on the CPU [...]

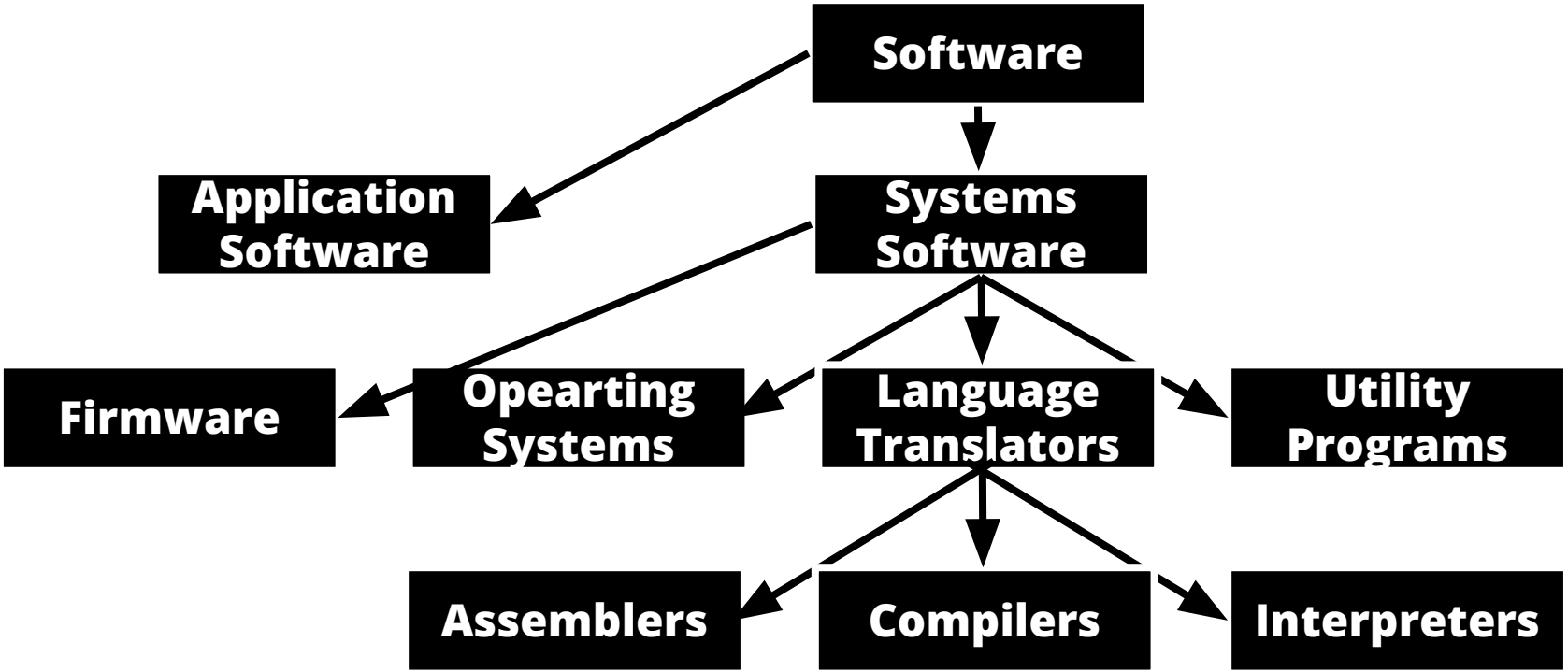


Hardware Abstraction (2/2)

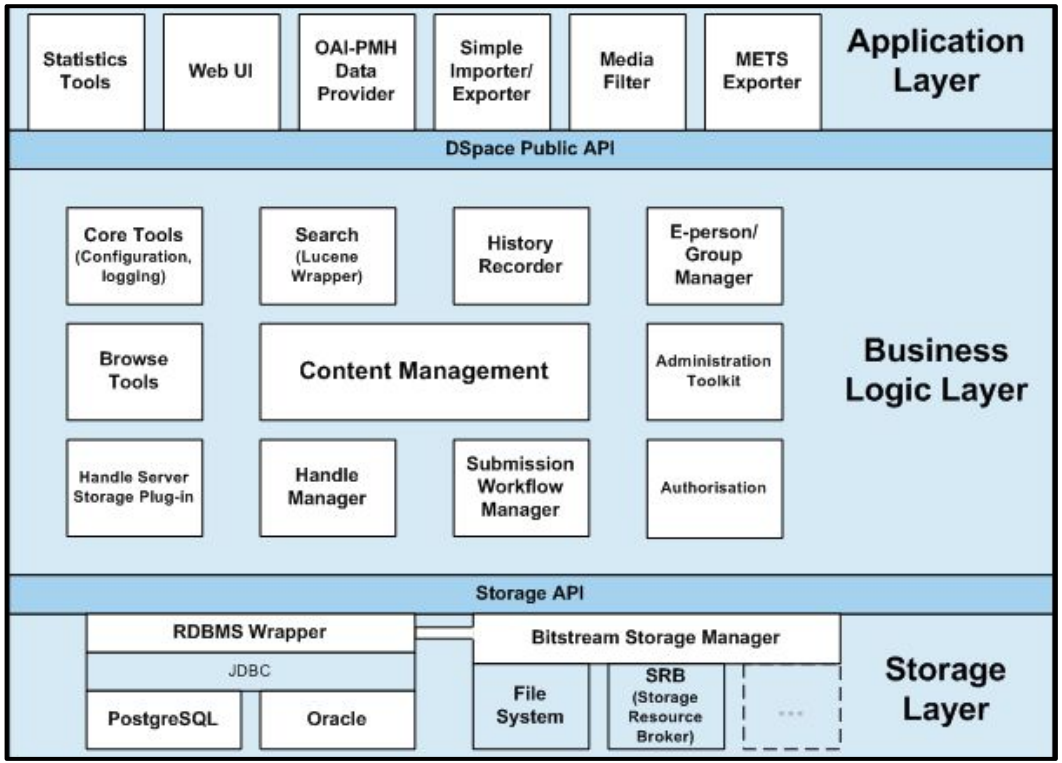
- **Abstracting the hardware views enables us to understand the most important components.**
  - Von Neumann Architecture presents memory as one block, however, there are various types of memory.



Software Abstraction (1/6)

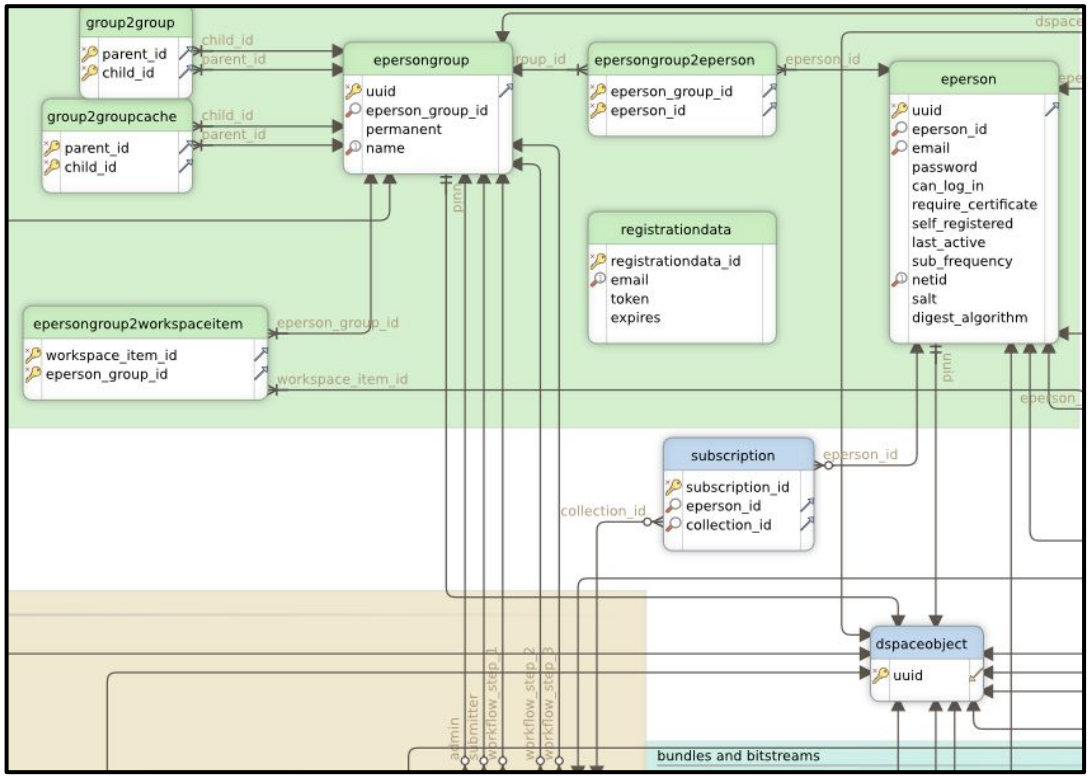


Software Abstraction (2/6)



- During development, software is abstracted by considering different representations of the software being developer

Software Abstraction (3/6)



- During development, software is abstracted by considering different representations of the software being developer

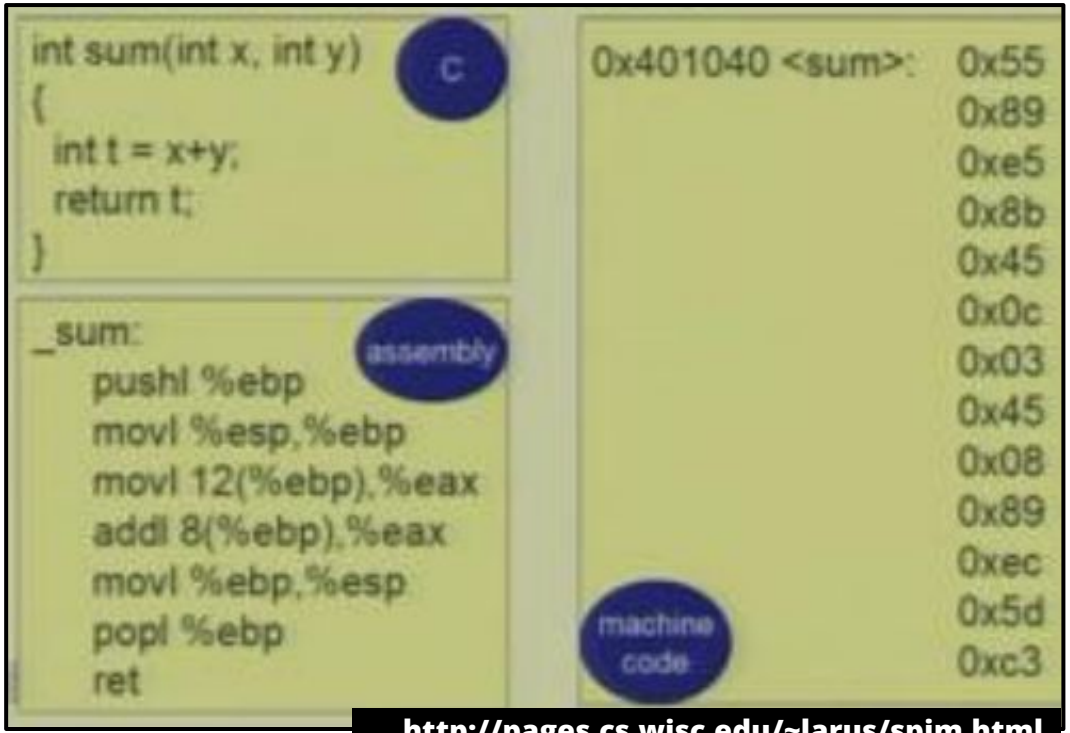
Software Abstraction (4/6)

```
import matplotlib.colorbar
import matplotlib.image
from matplotlib import _api
from matplotlib import rcsetup, style
from matplotlib import _pylab_helpers, interactive
from matplotlib import cbook
from matplotlib import _docstring
from matplotlib.backend_bases import (
    FigureCanvasBase, FigureManagerBase, MouseButton)
from matplotlib.figure import Figure, FigureBase, figa
from matplotlib.gridspec import GridSpec, SubplotSpec
from matplotlib import rcParams, rcParamsDefault, get_
from matplotlib.rcsetup import interactive_bk as _inte
from matplotlib.artist import Artist
from matplotlib.axes import Axes, Subplot
from matplotlib.projections import PolarAxes
```

- During development, software is abstracted by considering different representations of the software being developer

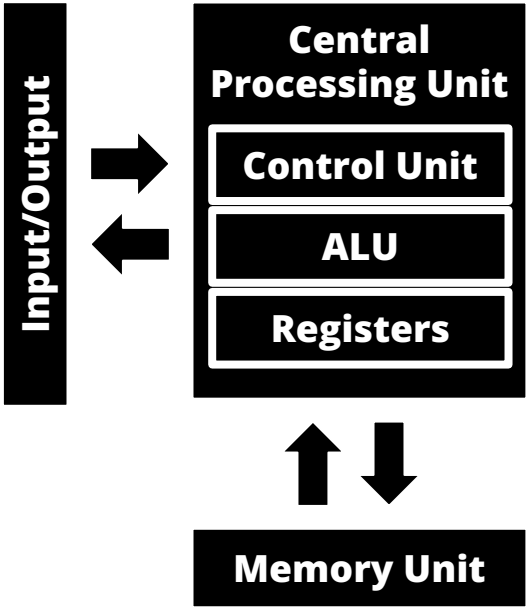
Software Abstraction (5/6)

- Programmers write typically write software using high-level languages. HLL abstracts machine code
- Translators convert code into different formats.



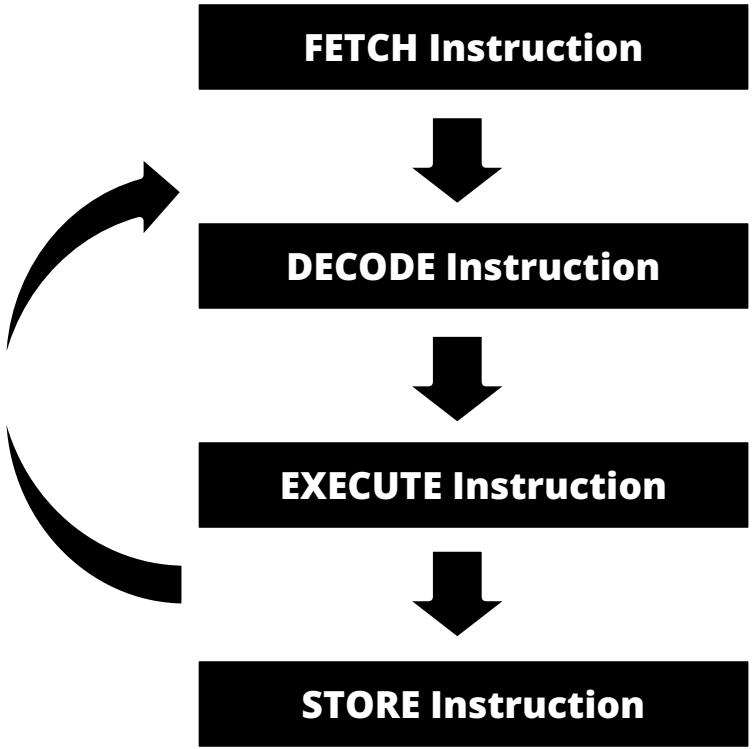
Software Abstraction (6/6)

- Instructions, from the computer system's point of view are strings of binary digits.
- When symbols are used for the binary strings, the instructions are called assembly language instructions.
- Components interpret the instructions and send signals to other components that cause the instruction to be carried out.



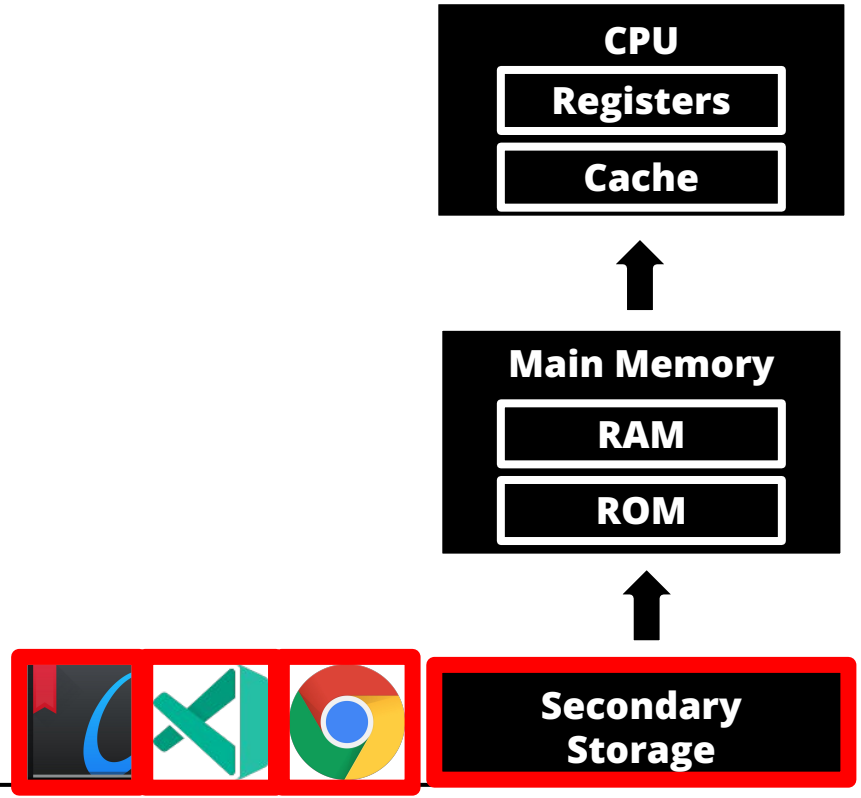
CPU Instruction Cycle (1/6)

- **Data processing in a computer occurs in three phases, before the results are stored into memory for further processing.**
  - Step 1: Instructions are fetched
  - Step 2: Instructions are decoded
  - Step 3: Instructions are executed



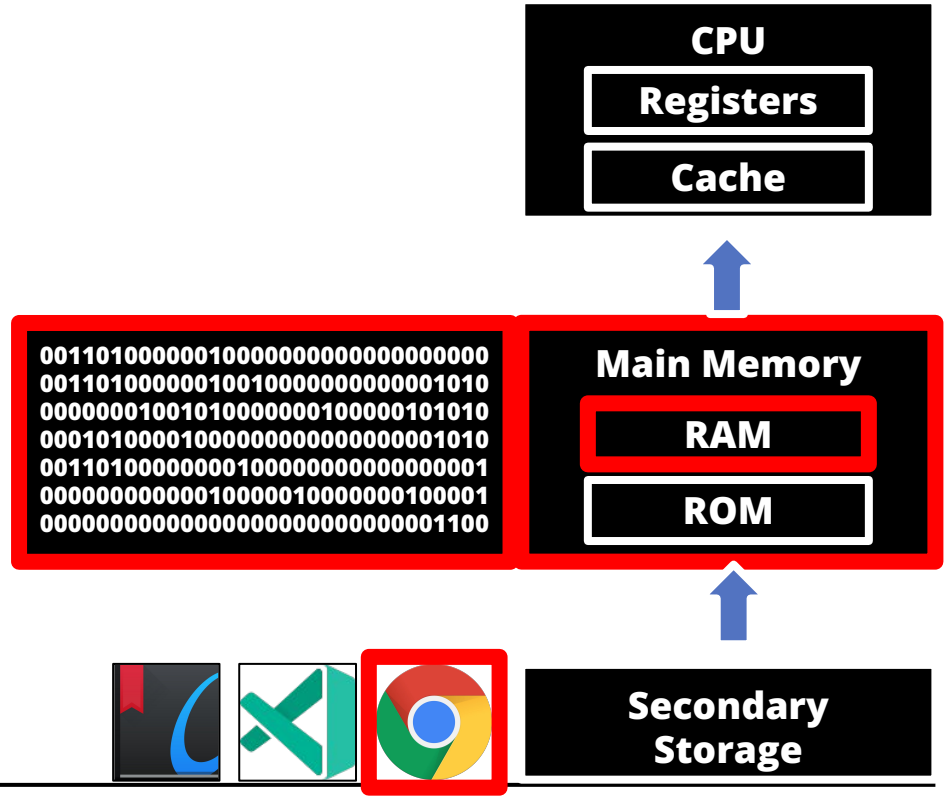
CPU Instruction Cycle (2/6)

- **Computer programs are executed from secondary storage.**
  - End users explicitly run/execute programs from secondary storage mediums
  - The average microcomputer will have a number of programs installed on disk



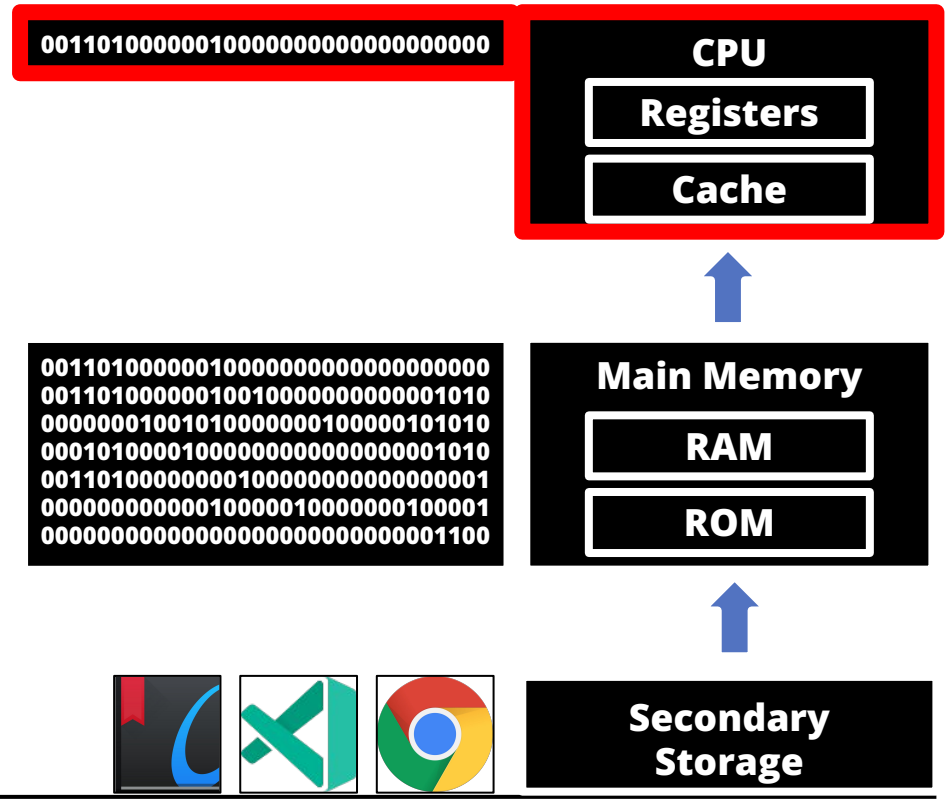
CPU Instruction Cycle (3/6)

- **Once explicitly run, the program is loaded into main memory.**
  - The entire program or portions of it will be loaded into main memory
  - The average microcomputer will have a number of programs loaded into main memory



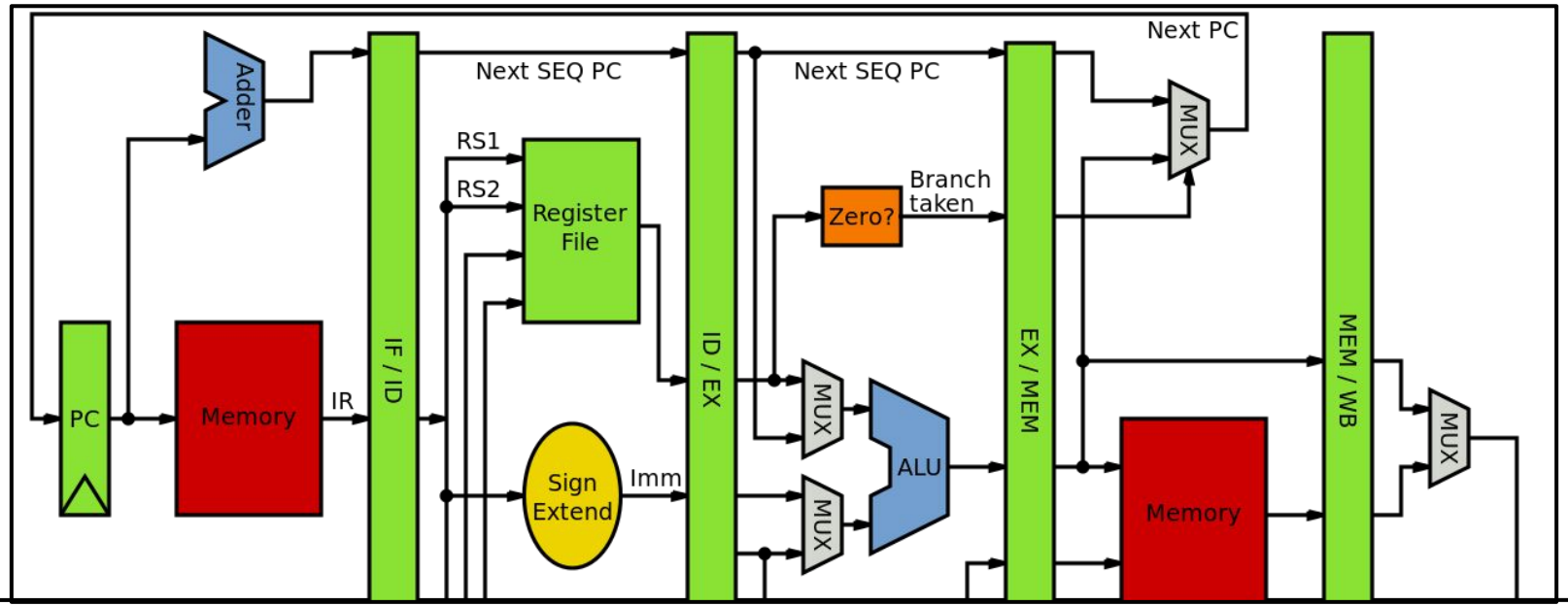
CPU Instruction Cycle (4/6)

- The actual execution of the program is done on the CPU, one instruction at a time
  - Each individual instruction associated with the program in memory is executed on the CPU in isolation
  - Execution is done at an extremely fast rate



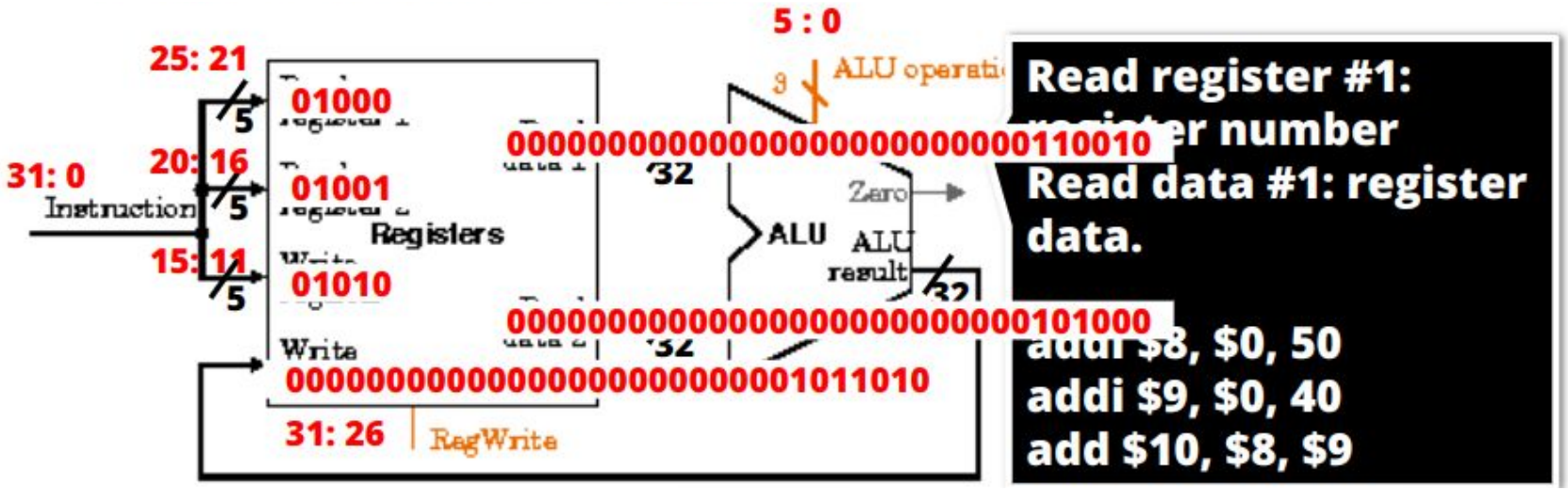
CPU Instruction Cycle (5/6)

- **Instruction cycle is much more complex than initially discussed.**



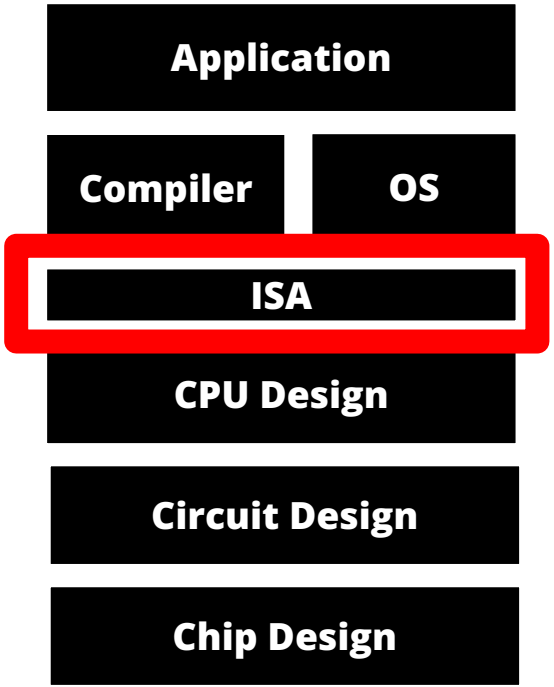
CPU Instruction Cycle (6/6)

- The execution path associated with execution of computer instructions involves a lot of detail



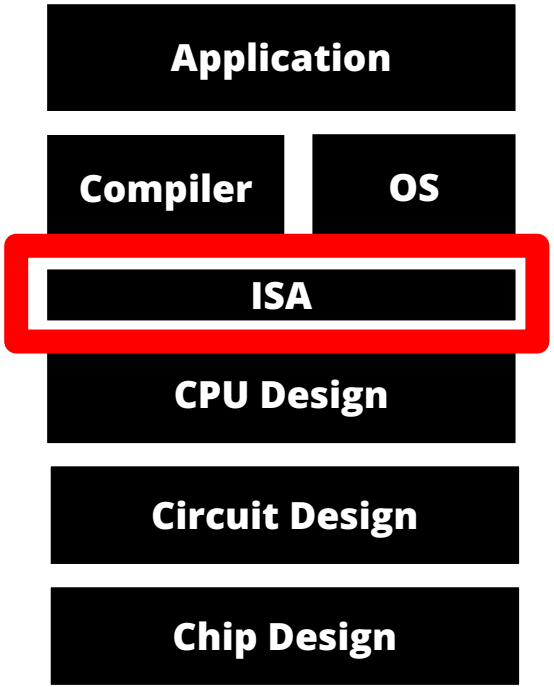
Instruction Set Architecture (1/4)

- **The abstracted hardware and software views enable us easily understand computer systems as a whole**
  - Above ISA: how to program the machine.
  - Below ISA: Designing and building hardware.
- **Our focus is going to be on the ISA.**
  - ISA is the lowest level visible to programmers.



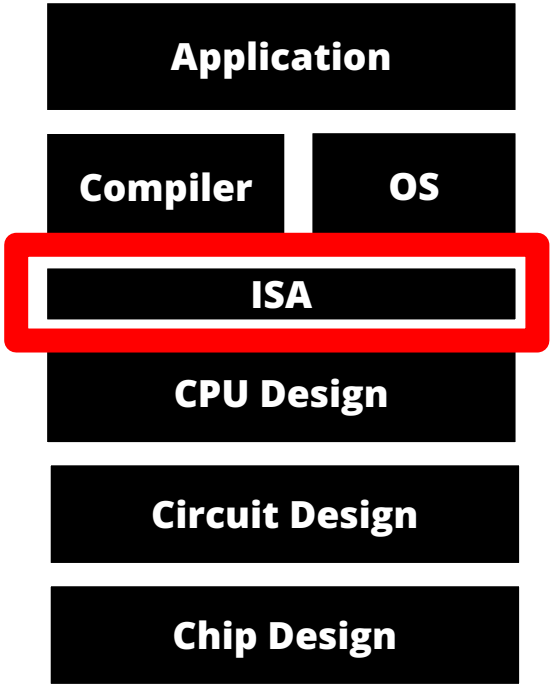
Instruction Set Architecture (2/4)

- **ISA can be viewed as the assembly language view.**
  - ISA defines the CPU state and how it changes as instructions are executed.
  - REMEMBER: CPU state is defined by contents of memory and registers.
- **When using assembly language the programmer has to understand how instructions are presented.**



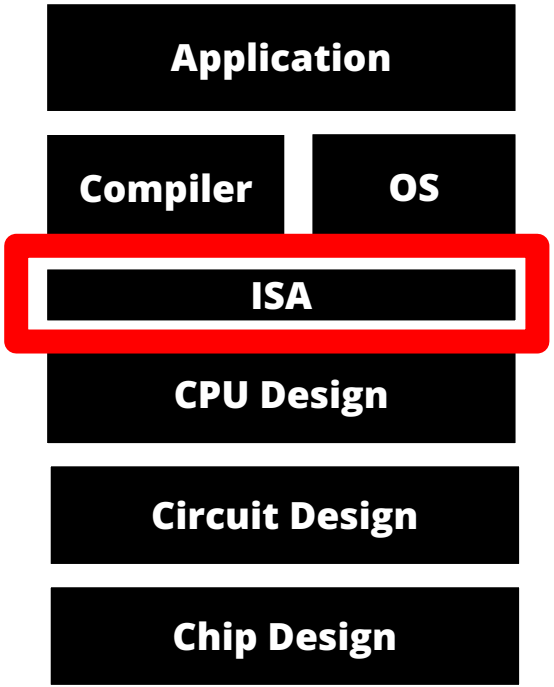
Instruction Set Architecture (3/4)

- **Instructions to be discussed.**
  - Instructions for arithmetic.
    - add, sub
  - Instructions for moving data.
    - lw, sw
  - Instructions for decision making.
  - Handling contact operands.
  - Implementing loops.
  - Switch statements.

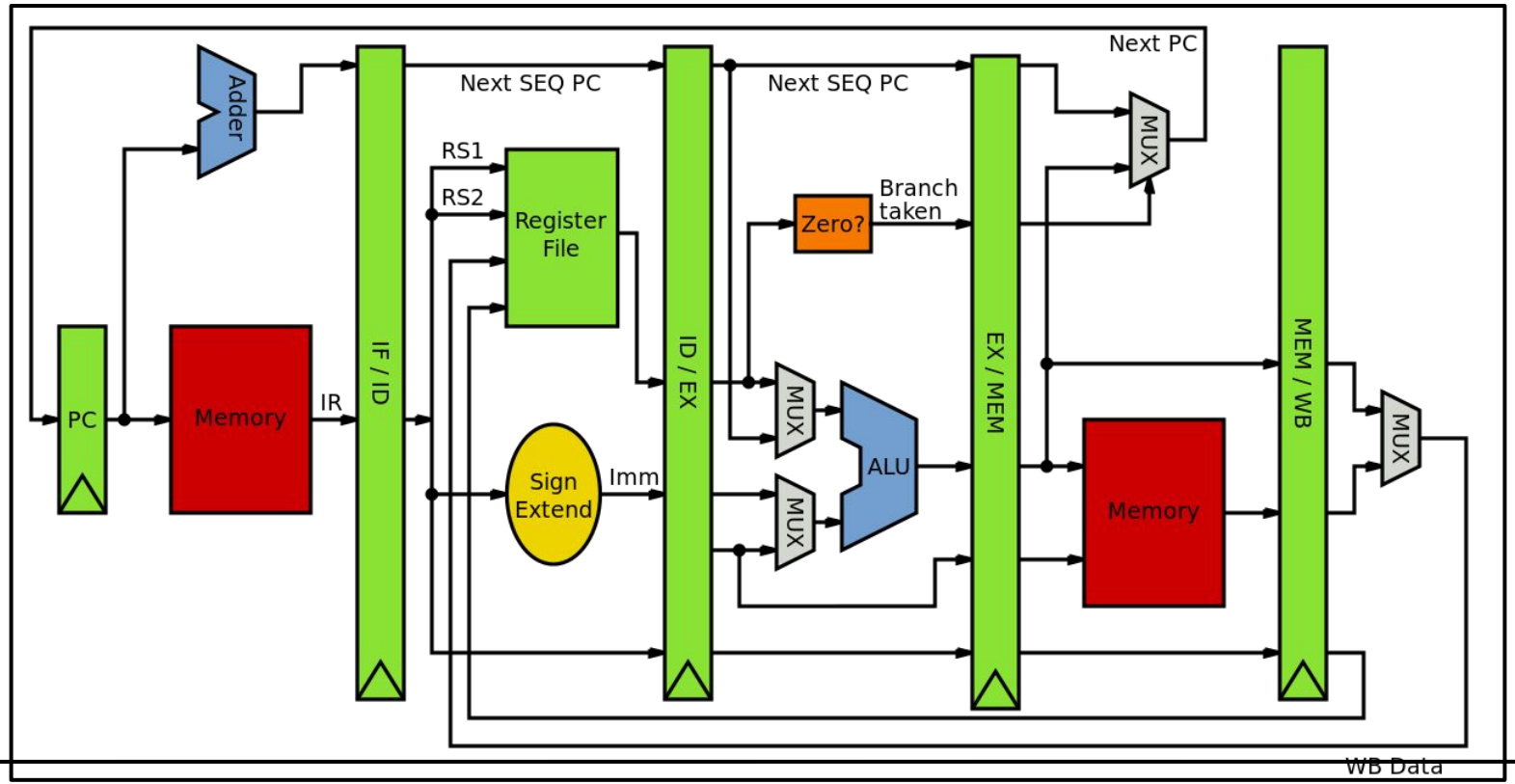


Instruction Set Architecture (4/4)

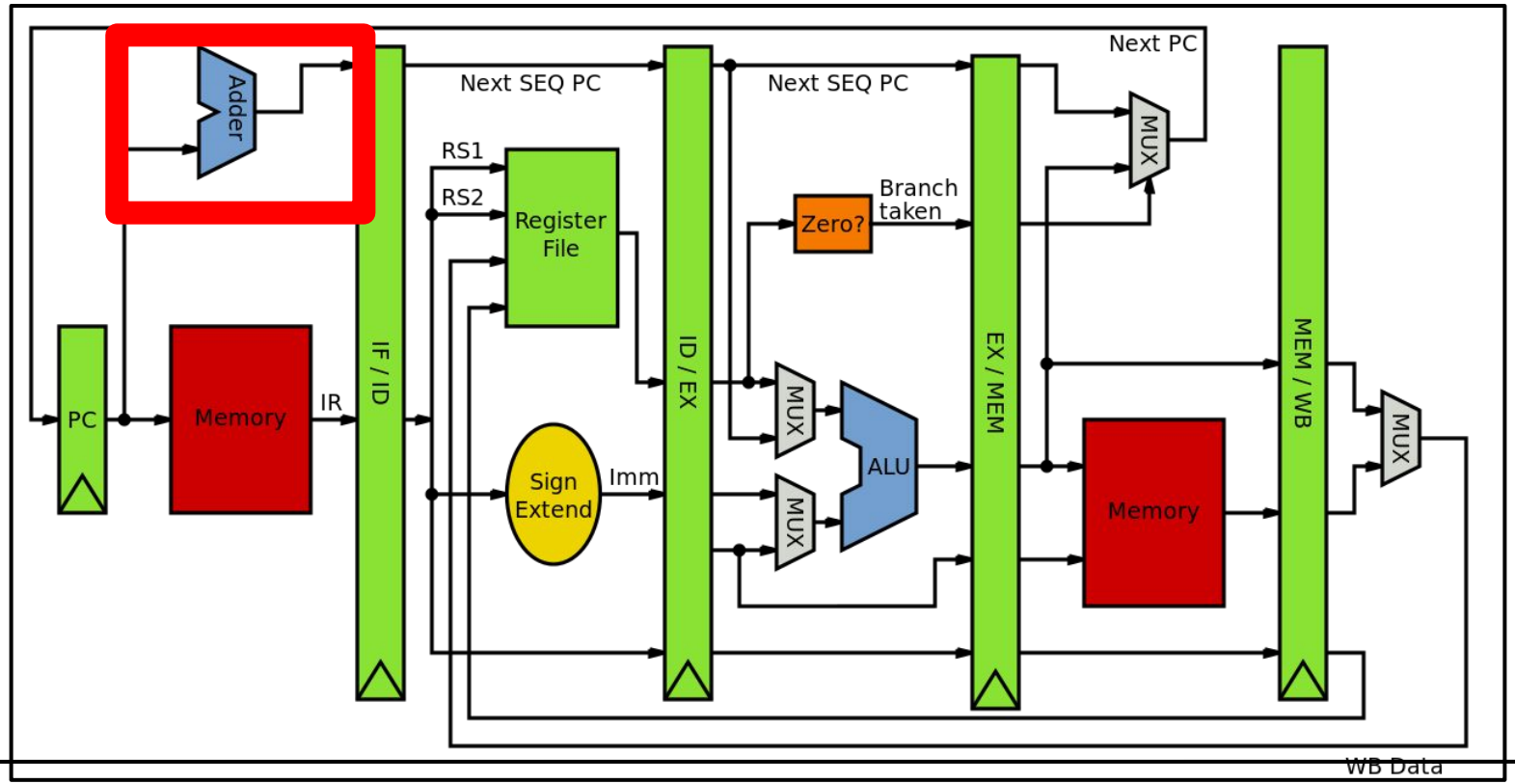
- **We will program with an assembly language using instructions of a real processor, MIPS, to understand the basics of hardware language.**
  - Our focus is going to be on the ISA.
  - Different classes of instructions will be discussed.
  - A mapping of HLL (C, C++, Java) to assembly will be discussed.



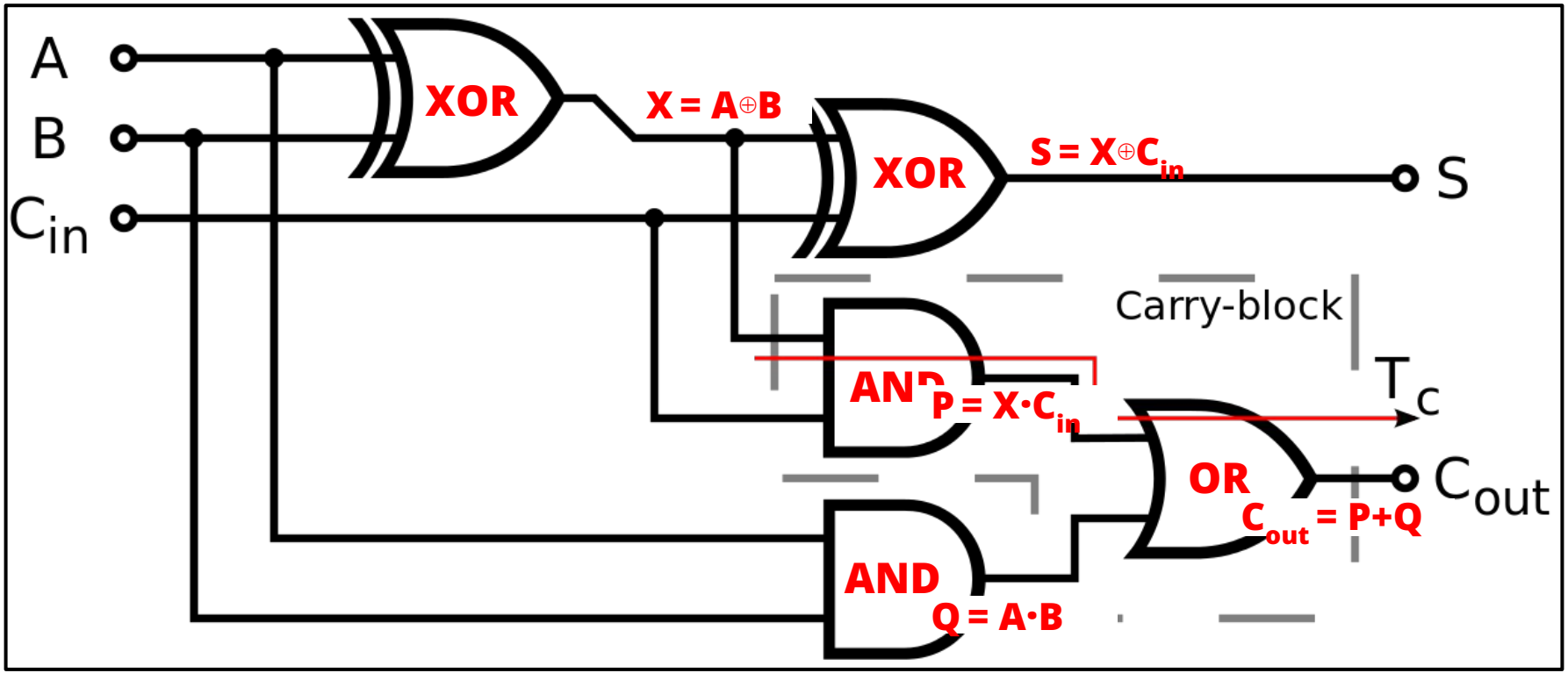
Digital Logic Structures (1/2)



Digital Logic Structures (1/2)



Digital Logic Structures (2/2)



Summary

- Computer Abstraction
- Examples of Abstraction

Q & A Session

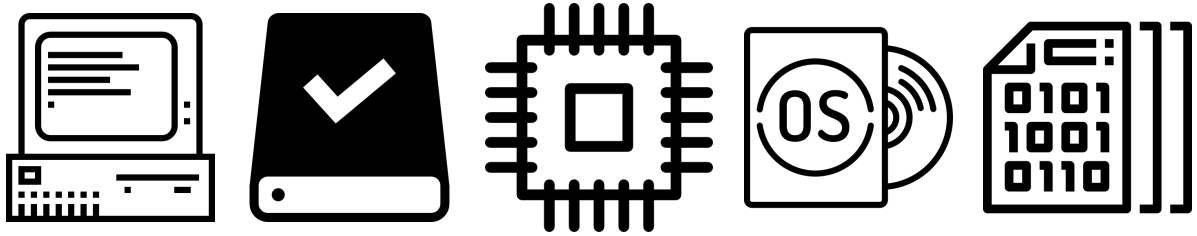
- Comments, concerns and complaints?

Bibliography

[1] William Stallings (2013), Instruction Sets: Characterists and Functions, Computer Organisation and Architecutre, Chapter 12

[2] Getting Started with SPIM
http://pages.cs.wisc.edu/~larus/spim.pdf

ict1110@unza.zm
https://bit.ly/3uJBZRc
http://bit.ly/2kK2ZkA



ICT 1110 (2022/23)
Computer Systems & Architecture
Lecture 4: Abstraction in Computer Systems

Lighton Phiri <lighton.phiri@unza.zm>
Department of Library & Information Science
University of Zambia
http://lis.unza.zm/~lightonphiri