Ollscoil na Gaillimhe
University of Galway

Dr Takfarinas Saber
takfarinas.saber@nuigalway.ie

CT213 Computing Systems & Organisation

Week 1: Introduction

Who Am I

- Takfarinas Saber
  - Call me: **Tak**, Takfarinas, or Dr SABER
  - Email: takfarinas.saber@universityofgalway.ie
  - Office: IT425 (4th floor, IT Building)

- BSc, MSc and PhD Computer Science
- Research Areas:
  - Resource optimisation in Cloud
  - Engineering and optimisation of efficient software applications:
    - Distributed over multiple hardware and geographical locations
    - with various real-time user interactions
    - programs processing large quantities of data

Overview

- **Course schedule:**
  - Monday 2 – 3 pm, **AC215**
  - Monday 3 – 4 pm, **McMunn Theatre**

- **Course material:** ***http://www.nuigalway.blackboard.com***
  - I will publish slides of the lecture every week with enough information on the slides to make them as clear as possible.
    - But more info in class

- **Textbooks:**
  - Digital Design and Computer Architecture (Second Edition), David Harris & Sarah Harris, ISBN: 978-0-12-394424-5
  - Computer Systems Organization & Architecture, John D. Carpinelli, ISBN: 0-201-61253-4
  - Computer Architecture: A Quantitive Approch, John L Hennessy and David A. Patterson, ISBN: 1-55860-329-8

Labs

- **Runs from**: Week 3 (Week starting 19/09/2022)
- **Focus**: Playing with Operating System Command Line, Using Raspberry Pi
- **Required**: Own laptop as advised by University of Galway (There will be no PCs in the lab, just docking stations)

- **Two Slots:**
  - Wednesdays 4pm to 6pm, IT101 Lab
  - Thursdays 4pm to 6pm, IT101 Lab
  - ➢ You need to choose **one slot** (the class has to be split into two groups of the same size)
- https://nuigalwayie-my.sharepoint.com/:x:/g/personal/0128261s_nuigalway_ie/EZknOSEhPqNJpSiGH3VI87oBh9W0IIUxni8mLEIOnlBdgA?e=fqfhbh

Assessment

- 70% Final Exam
- 30% Continuous Assessment:
  - Assignment 1: 15%
  - Assignment 2: 15%

- You **must** attend in-class assignments (to be announced in advance)

Contact Details

- For lecture/lab related questions
  - I have also create a **Forum** on Blackboard where you can ask questions directly to me.
  - Any questions related to lectures or labs must be asked in the Forum, if you think that all the class would like to know the answer.
    - You can ask questions anonymously if you wish

- For one-to-one interaction with me:
  - The best way to contact me is by email: takfarinas.saber@universityofgalway.ie
    - I will endeavour to respond within 48 hours.
    - If you would like to send me an email, use you University of Galway email account and state in the email: name, Module ID CT213, and Student ID.
  - I will also put in place a time period to discuss directly with you when necessary.

Learning Outcomes

Upon successful completion of this module, you will be able to:

- Explain the ***main concepts*** *behind* any **Operating System** and implement some of them
- ***Write shell scripts*** (system programming) to interact with Linux system to ***solve problems***
- Examine how **processes, memory**, **file**, and **device systems** are **managed** in a computer
- Investigate an **ARM processor** and **Raspberry Pi** Device

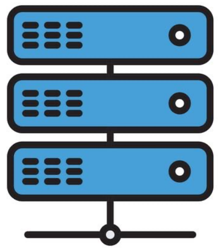
Syllabus

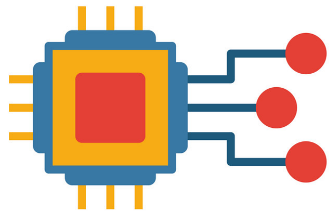
- Programming Models
- System software and Operating Systems
- Process Management and Process Synchronisation
  - ARM Processors
- Memory Management
- Device Management
- File Management
- Raspberry Pi

Overview of Computer Systems

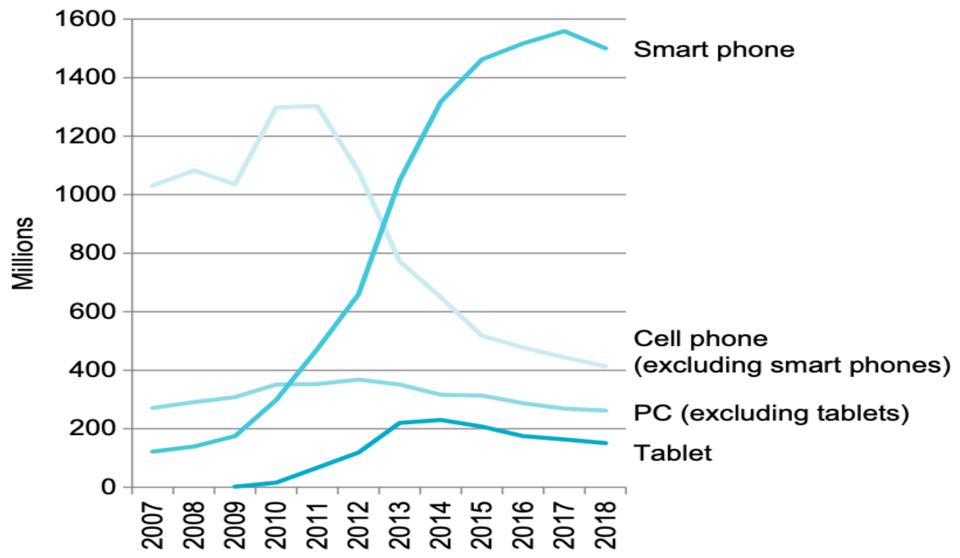
Traditional Classes of Computer Systems

- **Personal Computer (PC):** A computer designed for use by an individual, usually incorporating a graphics display, a keyboard, and a mouse.
- **Server:** A computer used for running larger programs for multiple users, often simultaneously, and typically accessed only via a network.
- **Supercomputer:** A class of computers with the highest performance and cost; they are configured as servers and typically cost tens to hundreds of millions of dollars.
- **Embedded computer**: A computer inside another device used for running one predetermined application or collection of software.





Post-PC Era



Personal mobile devices:

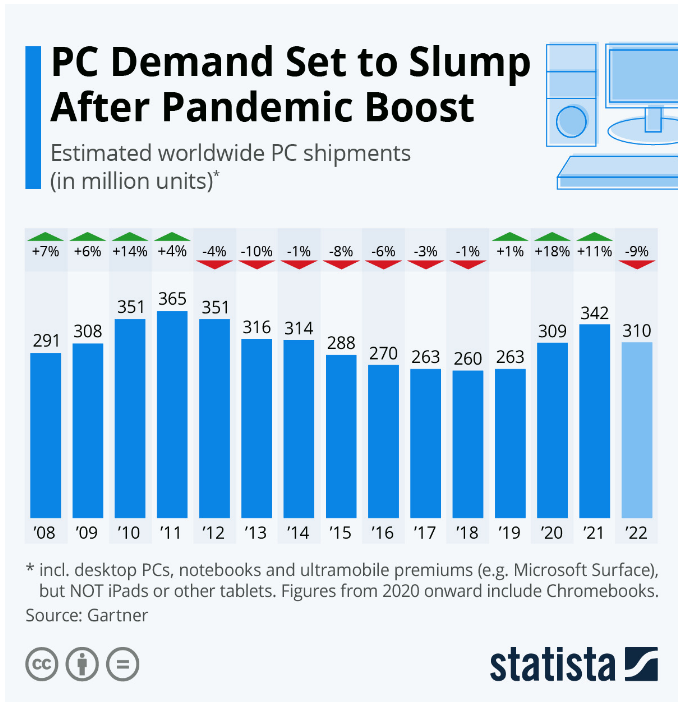
- Small wireless devices to connect to the Internet.
- They rely on batteries for power, and software is installed by downloading apps.
- Conventional examples are smart phones and tablets.

Cloud Computing:

- Refers to large collections of servers that provide services over the Internet;
- Some providers rent dynamically varying numbers of servers as a utility.

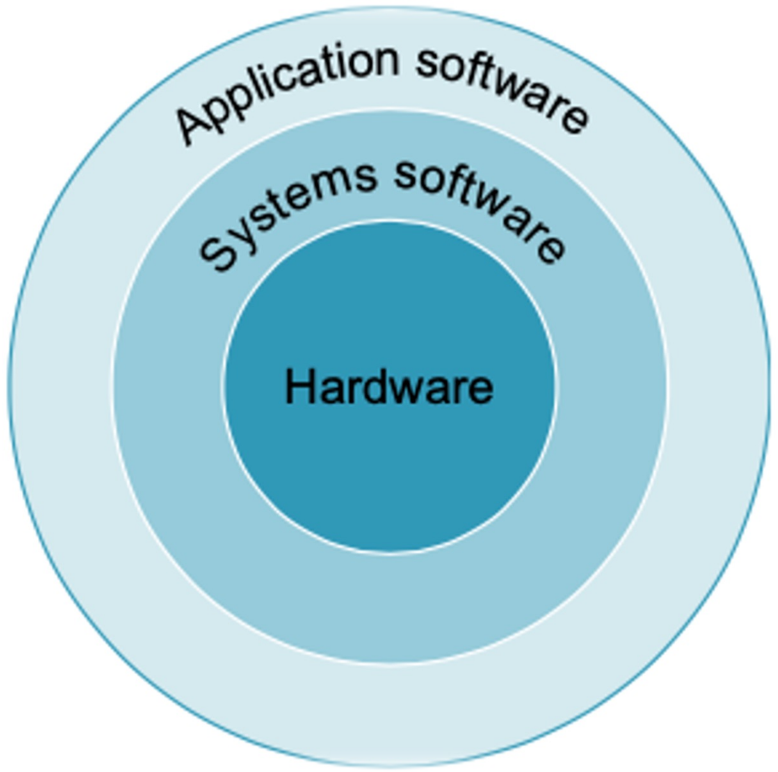
Software as a Service:

- Delivers software and data as a service over the Internet, usually via a thin program such as a browser.
- Examples include web search and email.



Computer Systems

- Application Software that provides services that are commonly useful.
- Operating system interfaces between a user's program and the hardware and provides a variety of services and supervisory functions.
- Hardware performs the tasks.
  - Which tasks?



Seven Great Ideas in Computer Organisation

1. Use Abstraction to Simplify Design

    A major productivity technique for hardware and software is to use **abstractions** to characterize the design at different levels of representation; lower-level details are hidden to offer a simpler model at higher levels.

2. Make the Common Case Fast

    Making the **common case fast** will tend to enhance performance better than optimizing the rare case. Ironically, the common case is often simpler than the rare case and hence is usually easier to enhance.

Seven Great Ideas in Computer Organisation

3. Performance via **Parallelism**

   Since the dawn of computing, computer architects have offered designs that get more performance by computing operations in parallel.

4. Performance via **Pipelining**

   A particular pattern of parallelism is so prevalent in computer architecture that it merits its own name: pipelining.

5. Performance via **Prediction**

   In some cases, it can be faster on average to guess and start working rather than wait until you know for sure, assuming that the mechanism to recover from a misprediction is not too expensive and your prediction is relatively accurate.

Seven Great Ideas in Computer Organisation

6. Hierarchy of Memories

Architects have found that they can address conflicting demands with a **hierarchy of memories:** the fastest, smallest, and the most expensive memory per bit at the top of the hierarchy and the slowest, largest, and cheapest per bit at the bottom.

7. Dependability via Redundancy

Since any physical device can fail, we make systems **dependable** by including redundant components that can take over when a failure occurs *and* to help detect failures.

Why Study Computing Systems & Organisation

- To get a job

Outline

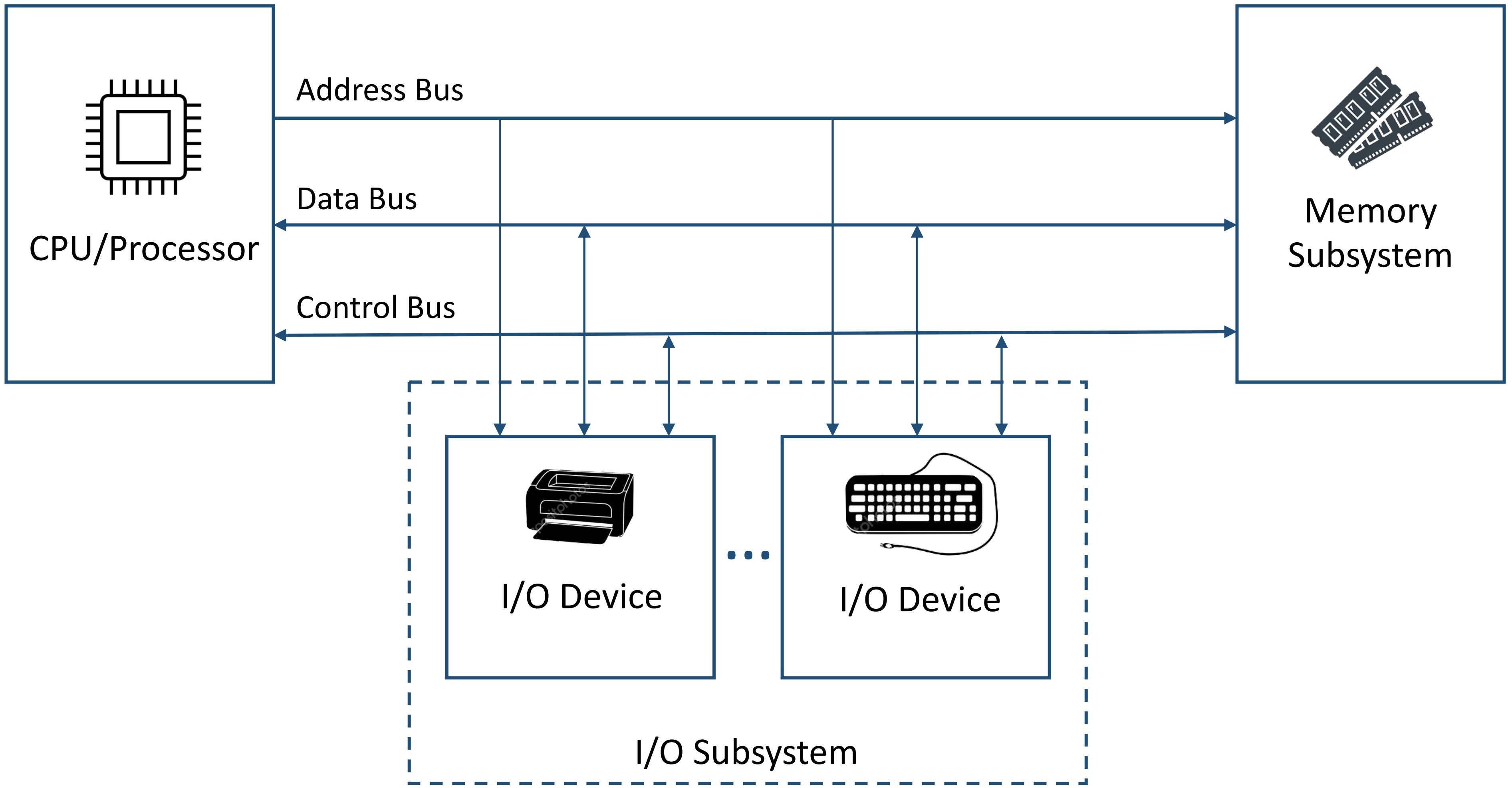
- Hardware Organisation
- Programs
- Operating System

- *Take home message:*
    - *A good understanding of how computing systems are organised is critical of IT professionals*

Hardware Organisation

How Does a Computer Look Like?

Basic Computer Organisation



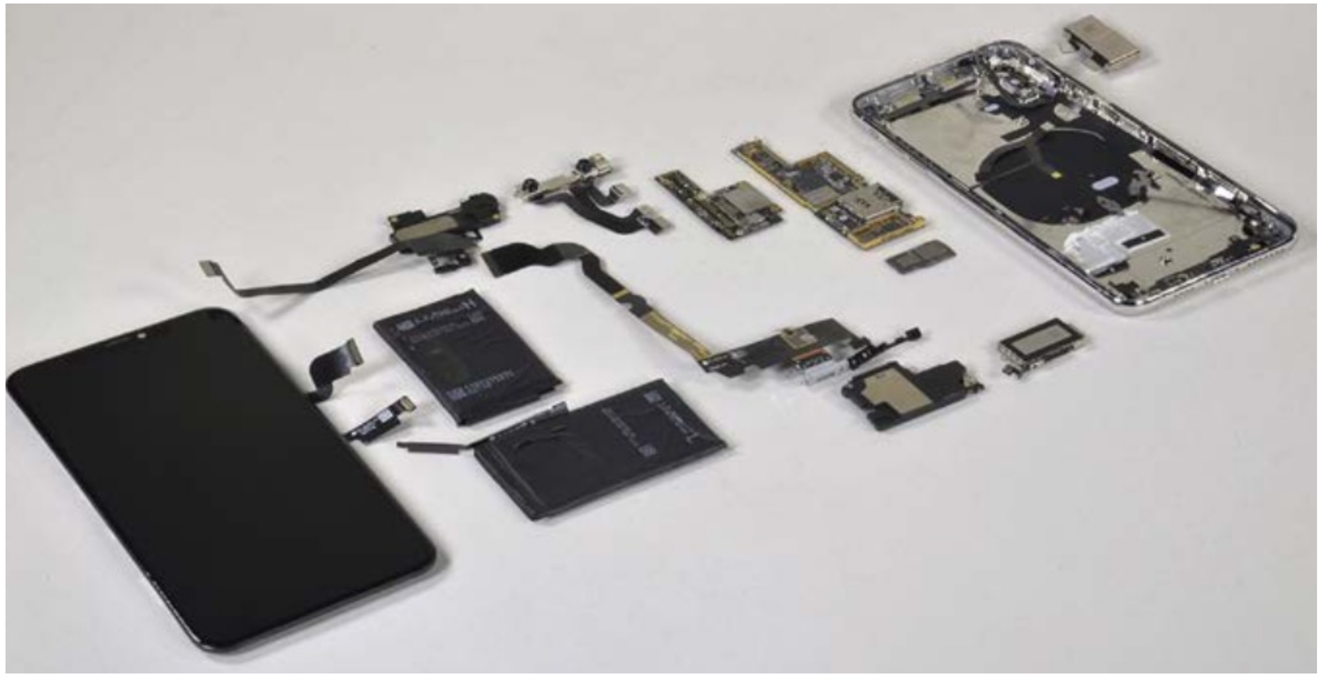
Apple iPhone XS Max smart phone

At the left is the capacitive multitouch screen and LCD display.

Next to it is the battery.

To the far right is the metal frame that attaches the LCD to the back of the iPhone.

The small components in the center are what we think of as the computer; Where?

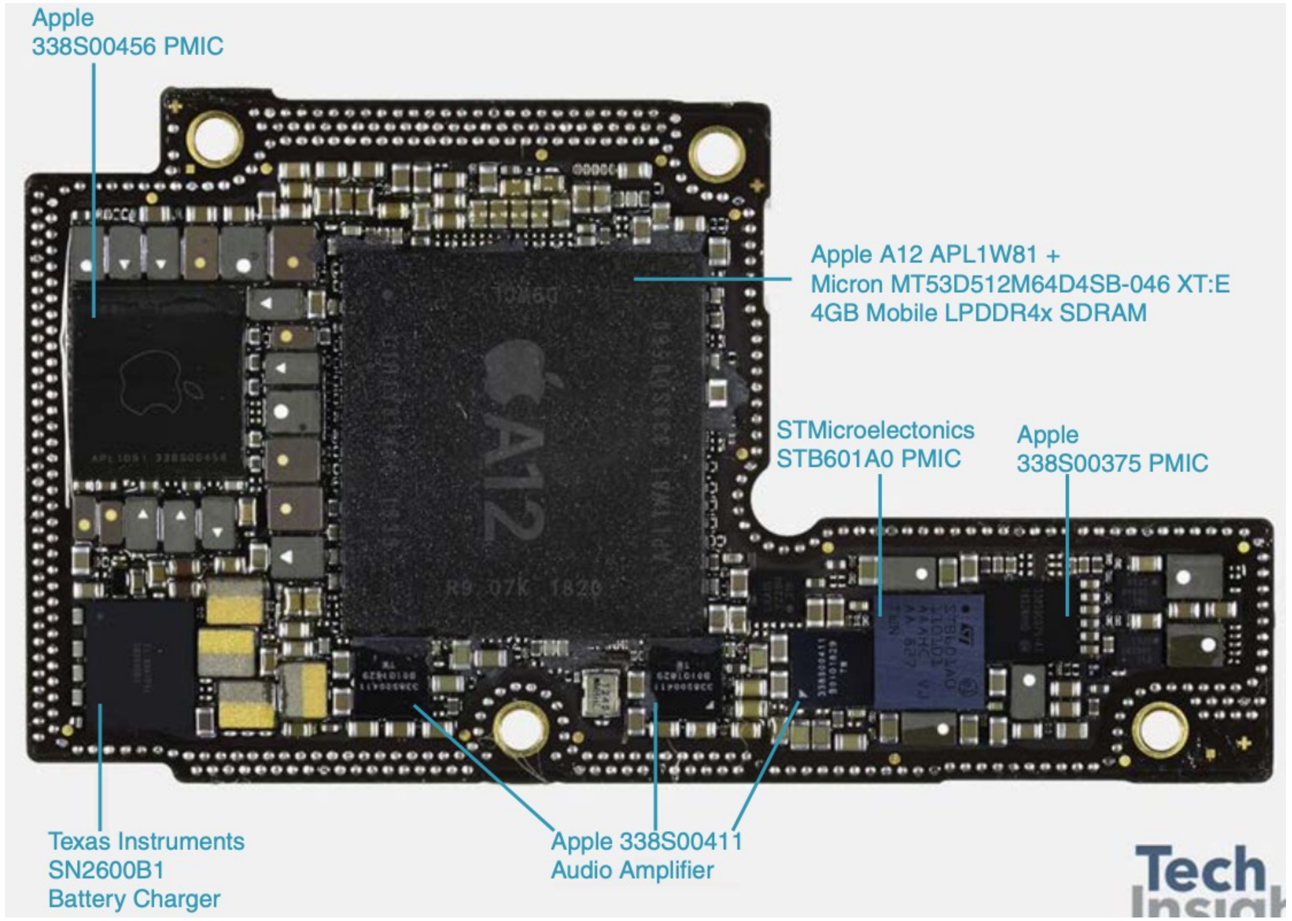


The logic board of Apple iPhone XS Max

Integrated circuit: also called a **chip**. A device combining dozens to millions of transistors.

Central Processor Unit (CPU): also called processor. The active part of the computer, which contains the datapath and control and which adds numbers, tests numbers, signals I/O devices to activate, and so on.

The other chips on the board include the **Power Management Integrated Controller (PMIC)** and **Audio Amplifier** chips.

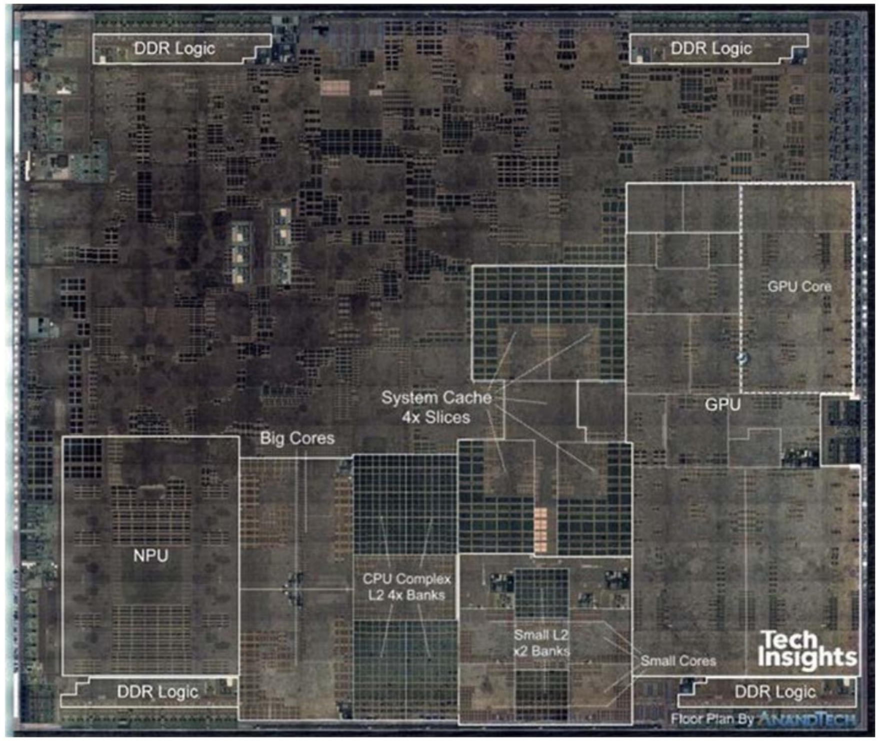


Central Processing Unit (CPU/Processor)

Responsible for executing programs

Processes programs in four steps:

1. **Fetch**: retrieve an instruction from program memory
2. **Decode**: break down the instruction into parts that have significance to specific sections of the CPU
3. **Execute**: Various portions of the CPU are connected to perform the desired operation
4. **Write Back**: Simply "writes back" the results of the execute step if necessary

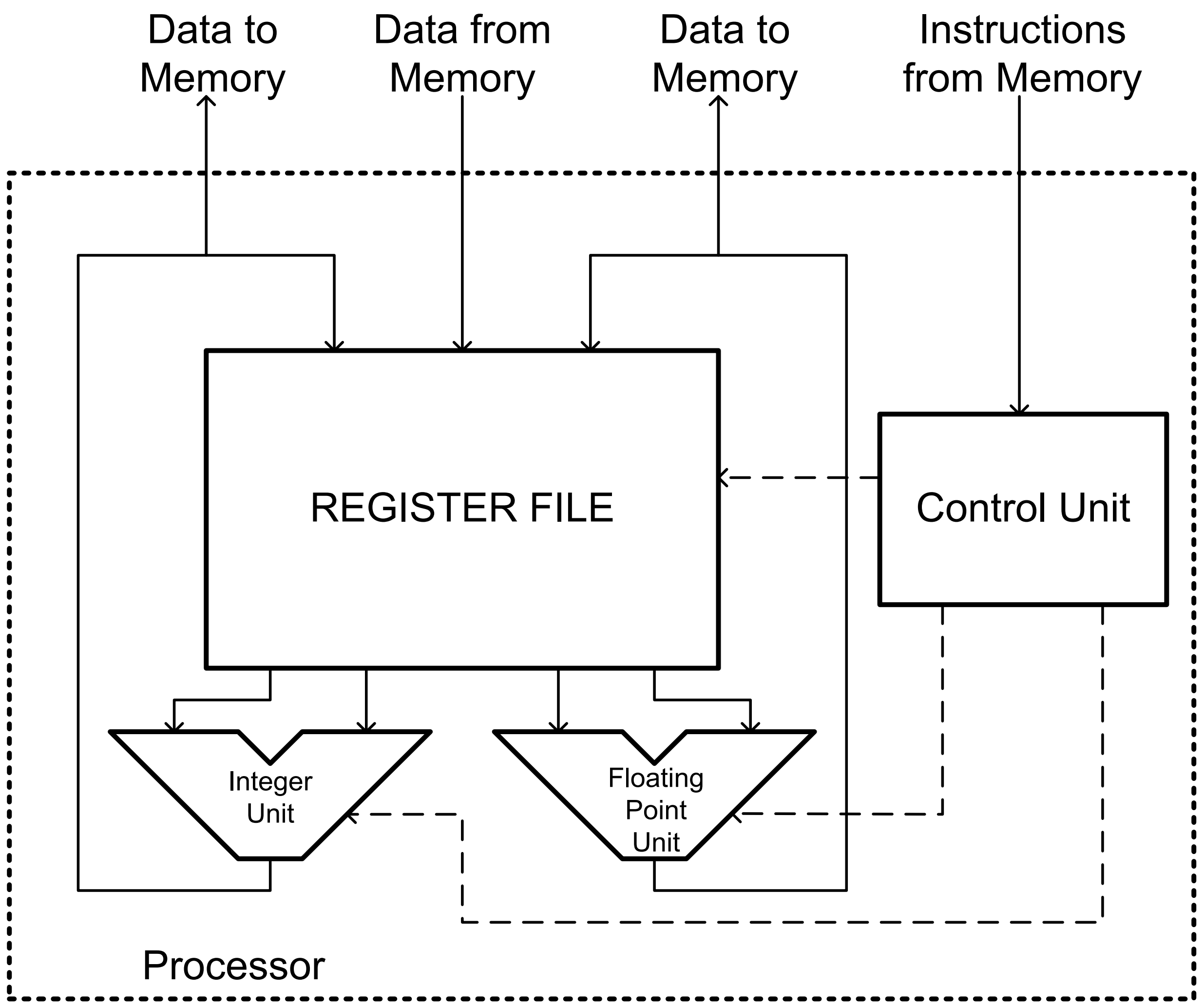


The processor integrated circuit inside the iPhone A12 package

CPU Organisation

Processors are made of:

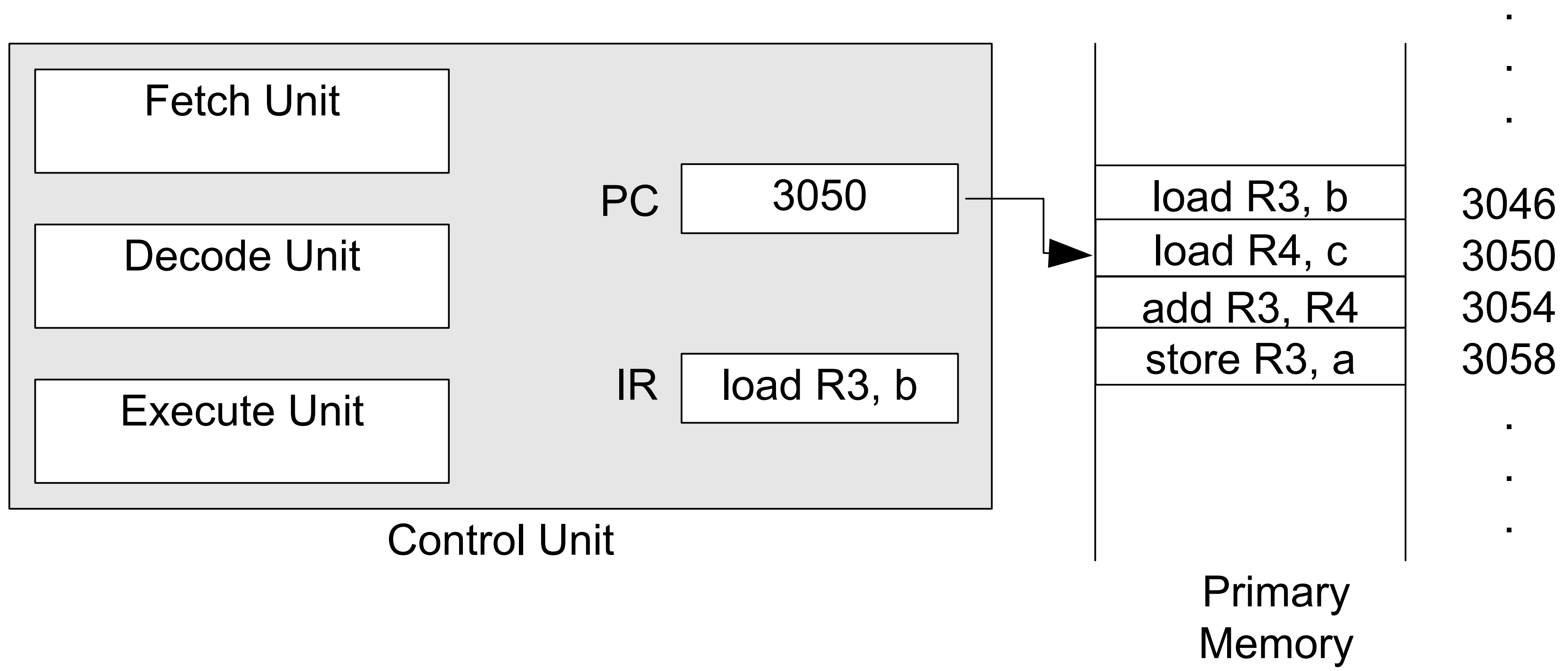
- Control Unit
- Execution Unit(s)
- Register file



Control Unit

The Control Unit controls the execution of the instructions stored in the main memory

- It retrieves and executes them



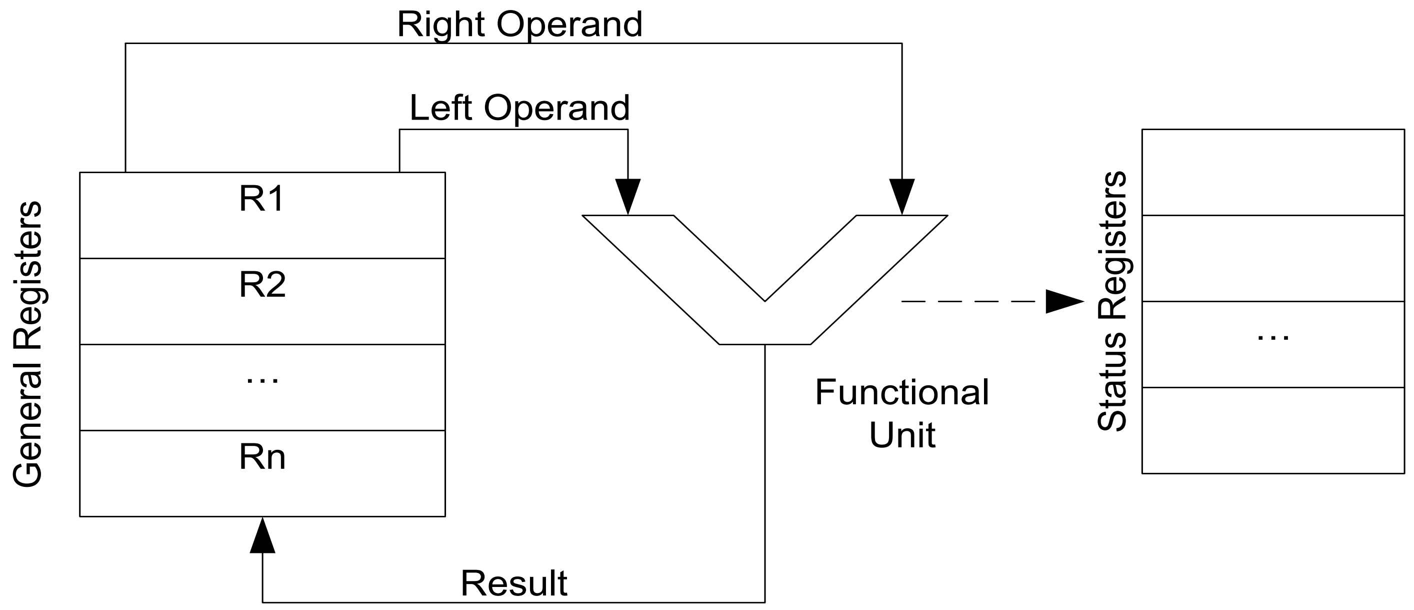
Special Registers:

- Program counter (PC): keeps the address of the next instruction
- Instruction Register (IR): keeps the instruction being executed

Execution Unit Example

Code for a = b + c :

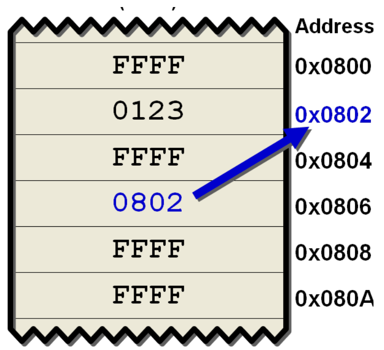
- LD R3, b //Load (copy) value b from memory to R3
- LD R4, c // Load (copy) value c from memory to R4
- add R3, R4 //sum placed in R3
- ST R3, a //store the result into memory as a



Memory Subsystem

- Memory is divided into a set of storage locations which can hold *data*.
  - Locations are numbered
  - Locations (i.e., *address*) are used to tell the memory which location the processor wants to access.

- There are two hierarchies of memory:
  1. **Nonvolatile / ROM (Read Only Memory):** Read only
     - Used to store the BIOS and/or a *bootstrap* or *boot loader* program
  2. **Volatile / RAM (Random Access Memory):** Read/write
     - Also called Primary Memory
     - Used to hold the programs, operating system and data required by the computer



Primary Memory

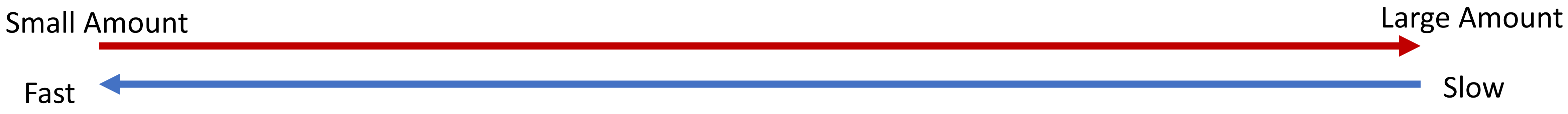
- Primary Memory is directly connected to the central processing unit of the computer.
  - ➢ It must be present for the CPU to function correctly.

- 3 types of primary storage:

Processors Register: Contains information that CPU needs to carry out the current instruction.

Cache memory: Special type of internal memory used by many CPUs to increase their performance or "throughput.

Main memory: Contains the programs that are currently being run and the data the programs are operating on.



Memory Subsystem (Cont'd)

- The width of **address** limits the amount of memory that a computer can access
  - Most current computers use a 64 bit address, which means that the maximum number of locations is 2^{64}, about (16 billion gigabytes).

- Memory subsystem supports **two operations**:
  - Load (or read) – address of the data location to be read
  - Store (or write) – address of the location and the data to be written

- Memory subsystem allows for more than 1 byte to be read or written at a time
  - Read and write operations operate at the width of system's data bus, usually 32 bit (4 bytes), or 64 bits (8 bytes).
  - The address contains the address of the lowest byte to be addressed
  - e.g., with a 4 byte read operation from address 0x1000 would return bytes stored at addresses: 0x1000, 0x1001, 0x1002 and 0x1003

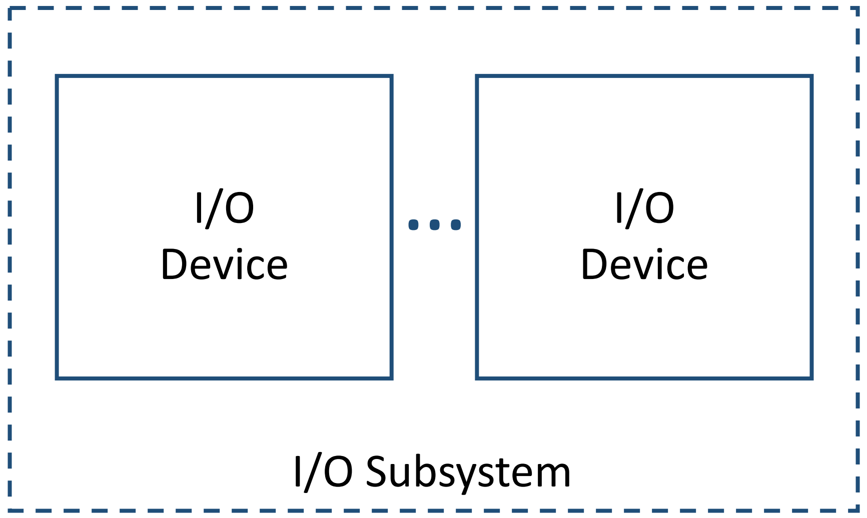
Memory Alignment and Word of Data

- When the computer's word size is 4 bytes the data to be read should be at a memory address which is some multiple of 4.
- When this is not the case, e.g. the data starts at address 14 instead of 16, then the computer has to read two or more 4 byte chunks and do some calculation before the requested data has been read, or it may generate an alignment fault.
- Even though the previous data structure end is at address 13, the next data structure should start at address 16. Two padding bytes are inserted between the two data structures at addresses 14 and 15 to align the next data structure at address 16.

Input/Output

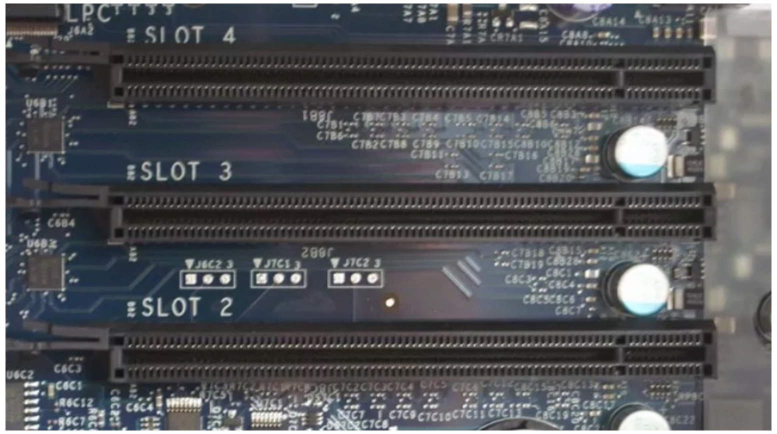
Input Devices: Anything that feeds data into the computer

Output Devices: Display / transmit information back to the user



The I/O Subsystem

- Contains devices that the computer uses to communicate with the outside world and to store data
- I/O devices are usually communicating with the processor using an I/O bus
  - PCs are using PCI Express (Peripheral Component Interconnect Express) bus for their I/O bus
  - OS needs a ***device driver*** to access a given I/O device
    - Program that allows the OS to control the I/O device



PCI Express (PCIe 5.0)

I/O Read/Write Operations

- The I/O read and write operations are similar to the memory read and write operations.

- A processor may use:
  - **memory mapped I/O** – when the address of the I/O device is in the direct memory space, and the sequence to read/write data in the device are the same with the memory read/write sequence
  - **isolated I/O** – the process is similar, but the processor has a second set of control signals to make the distinction between a memory access and an I/O access

- **IO/M** signal **is a status signal.**
  - When this signal is low (IO/M = 0) it denotes the memory related operations.
  - When this signal is high (IO/M = 1) it denotes an I/O operation.

Programs

Programs

- Sequences of instructions that tell computer what to do
- To the computer, a program is made out of a sequence of numbers that represent individual operations.
    - Those operations are known as ***machine instructions*** or just ***instructions***
    - A set of instructions that a processor can execute is known as ***instruction set***

Program Development Tools

From a High-Level Language to the Language of Hardware

High-level programming language: A portable language such as C, C++, Java, or Visual Basic that is composed of words and algebraic notation that can be translated by a compiler into assembly language.

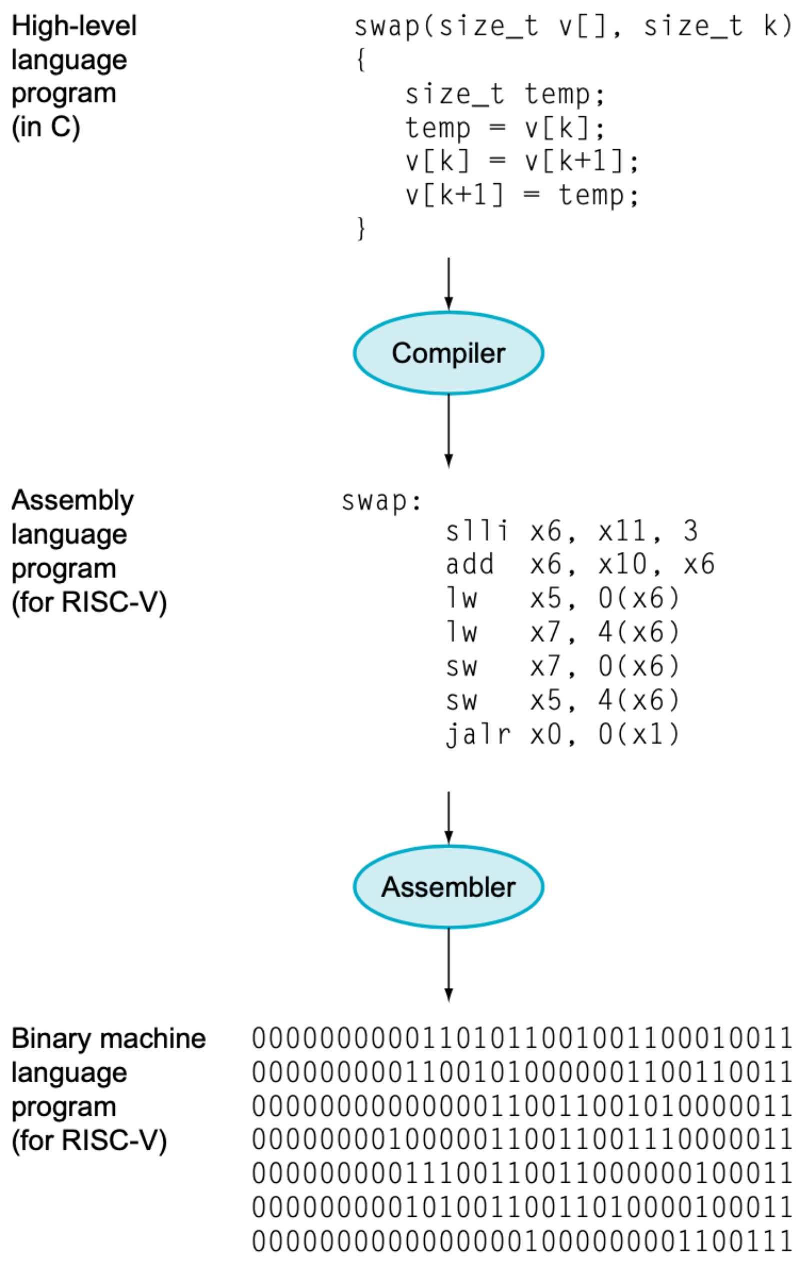
Compiler: A program that translates high-level language statements into assembly language statements.

Assembler: A program that translates a symbolic version of instructions into the binary version.

Assembly language: A symbolic representation of machine instructions.

Machine language: A binary representation of machine instructions.

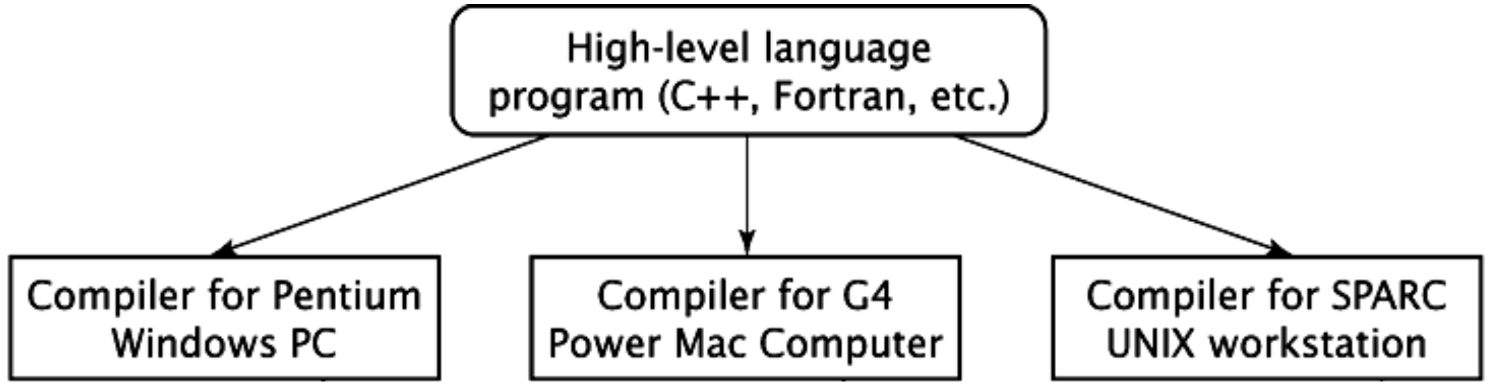
Instruction: A command that computer hardware understands and obeys.


High-level language program (in C)
Compiler
Assembly language program (for RISC-V)
Assembler
Binary machine language program (for RISC-V)

```
swap(size_t v[], size_t k)
{
   size_t temp;
   temp = v[k];
   v[k] = v[k+1];
   v[k+1] = temp;
}
```

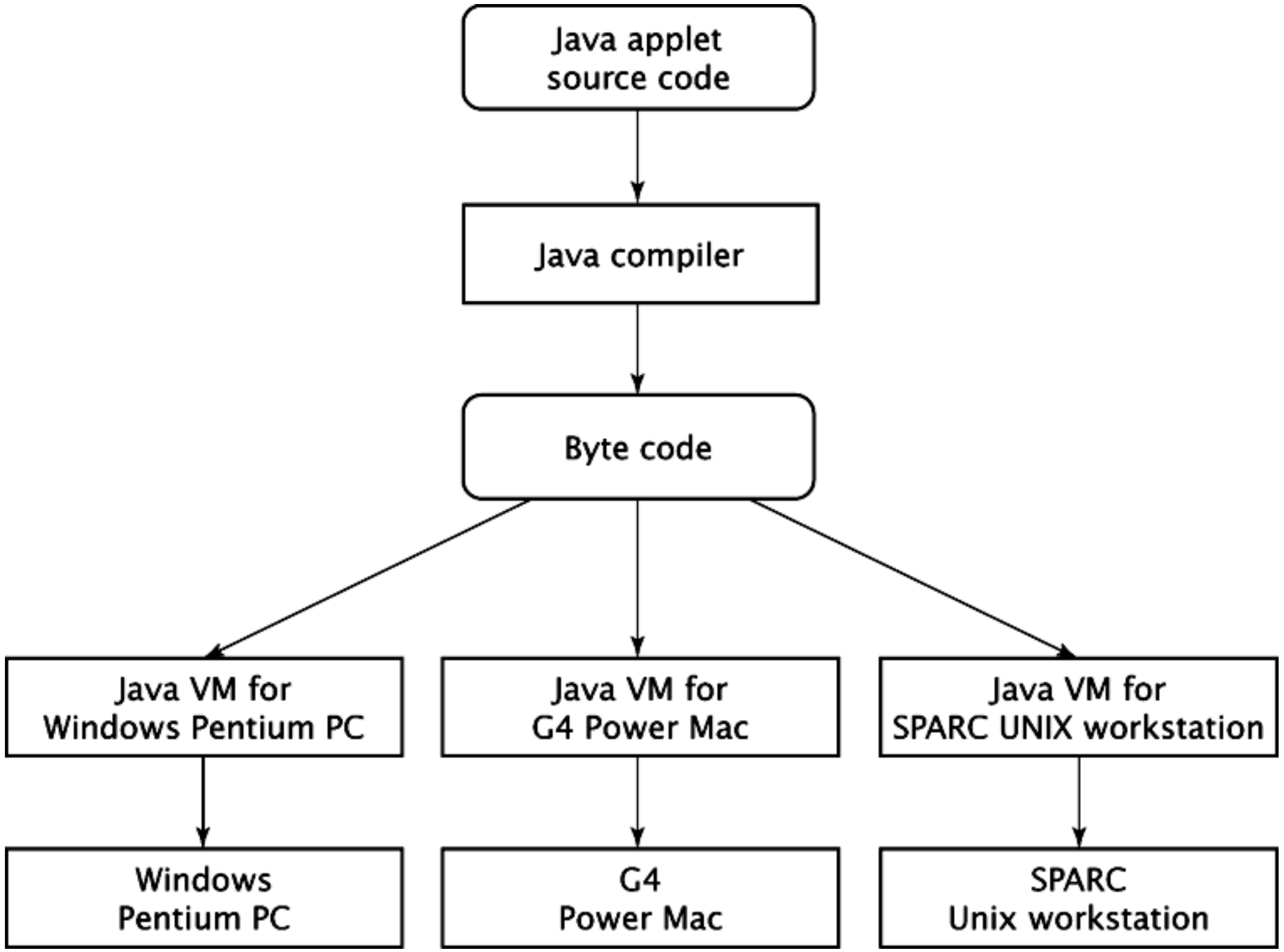
```
swap:
      slli x6, x11, 3
      add  x6, x10, x6
      lw   x5, 0(x6)
      lw   x7, 4(x6)
      sw   x7, 0(x6)
      sw   x5, 4(x6)
      jalr x0, 0(x1)
```

```
00000000001101011001001100010011
00000000011001010000001100110011
00000000000000110011001010000011
00000000100000110011001110000011
00000000011100110011000000100011
00000000010100110011010000100011
00000000000000001000000001100111
```


High-level language program (C++, Fortran, etc.)
Compiler for Pentium Windows PC
Compiler for G4 Power Mac Computer
Compiler for SPARC UNIX workstation

Compiling Programs

Java – Different way of processing



Operating Systems

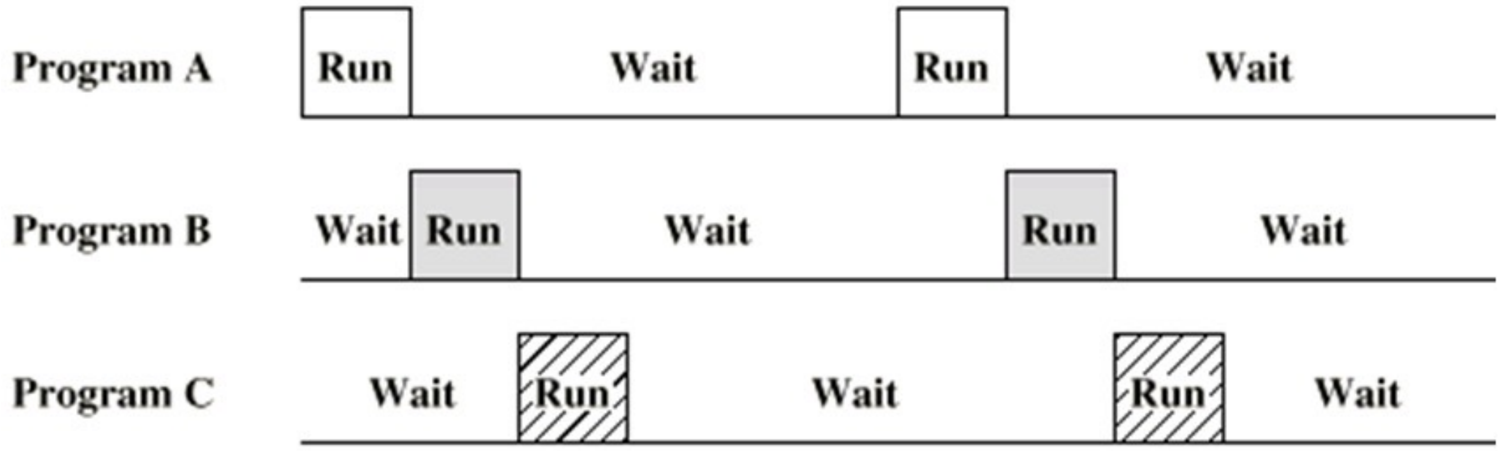
Operating System

- Responsible for managing the physical resources of complex systems (PCs, workstations, mainframe computers)
- Responsible for loading and executing programs and interfacing with the users
- Usually, no operating system for small embedded systems
  - Computers designed for one specific task
- A possible definition: It is a **program** that runs on the computer, that **knows about all the hardware** and usually runs in ***privileged*** (or supervisor) mode, **having access** to physical resources that user programs can't control and has the **ability to start and stop user programs**

Multiprogramming

- Technique that allows the system to present the illusion that multiple programs are running on the computer simultaneously

- Many multiprogrammed computers are ***multiuser***
  - Allow multiple persons to be logged on at a time



Multiprogramming (Cont'd.)

Multiprogramming is achieved by switching rapidly between programs

- FCFS – First Come, First Served (also called FIFO)
  - processes are moved to the CPU in the order in which they arrive

SJN – Shortest Job Next

looks at all processes in the ready state and dispatches the one with the smallest service time

Round Robin

distributes the processing time equitably among all ready processes

Advantages:

- simple
- easy to implement
- starvation-free

Disadvantages:

- set time too short => too much process switching => too slow.
- set time too long => unresponsive system, time wasting

Context Switch

- When a program timeslice ends, the OS stops it, removes it and gives another program control over processor
    - This is a ***context switch***

- To do a context switch the OS:
    - Copies the content of current program register file into memory
    - Restores the contents of the next program's register file into the processor
    - and starts executing the next program.

- From the program point of view, no program can tell that a context switch has been performed

Protection

- Three rules:
  1. The result of any program running on a multiprogram computer **must be the same** as if the program was the only program running on the computer
  2. Programs **must not be able to access** other program's data and must be confident that their data will not be modified by other programs.
     - For **security** and ***privacy***
  3. Programs **must not interfere** with other program's use of I/O devices

How to Achieve Protection

Protection is achieved by the operating system having full control over the resources of the system (processor, memory and I/O devices) through:

- **Privileged Mode:** the operating system is the only one that can control the physical resources it executes in privileged mode
  - ➢ User programs execute in ***user mode***
- **Virtual Memory:** each program operates as if it were the only program on the computer, occupying a full set of the address space in its virtual space.
  - The OS is ***translating*** memory addresses that the program references into physical addresses used by the memory system.

References

- “Computer Systems Organization & Architecture”, John D. Carpinelli, ISBN: 0-201-61253-4