COMP124

Computer Systems

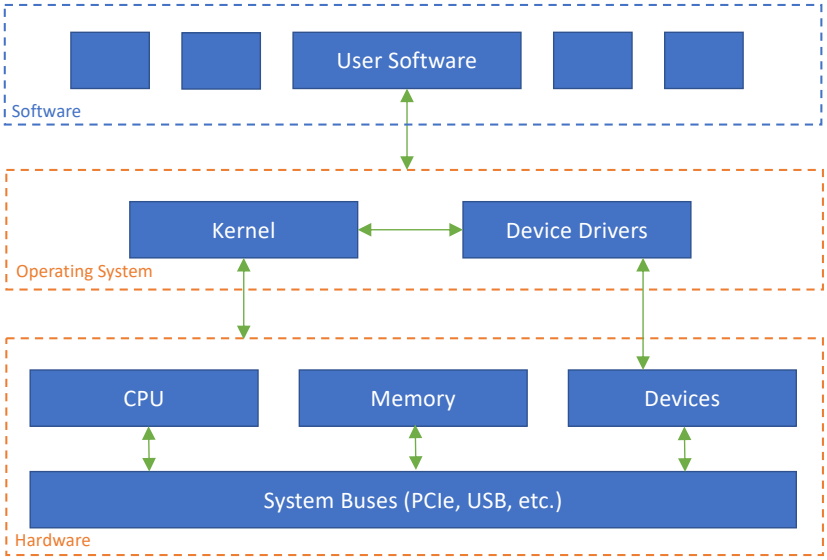
01 | Module Introduction | System Bus

Dr Stuart Thomason

Module Aims

- To introduce how computers function at the instruction operational level
- To introduce the relationships between the instruction operational level and both the higher (software) and lower (hardware) levels
- To introduce the structure and functionality of modern operating systems
- To explain how the principal components of computer systems perform their functions and how they interact with each other

Software & Hardware



Module Structure

- CPU architecture and machine instructions
- Assembly language programming
- Operating system concepts
- Process management
- File and device management
- Memory management
- Concurrent programming
- Compilers and code generation

Learning Outcomes

- LO1: Describe the structure and operation of computer hardware at the register transfer level
- LO2: Understand and reason about simple assembly language algorithms
- LO3: Describe the overall structure and functionality of a modern operating system and its interactions with computer hardware and user processes
- LO4: Explain how modern operating systems and programming languages implement concurrency and the issues that arise when working with concurrent processes
- LO5: Use the Linux command line and describe how files, devices and processes are managed by the Linux kernel

Module Delivery

- Delivery will be in person via lectures and lab drop-in sessions
  - Double lecture every week on Monday from 1300 to 1500 (weeks 1 to 11)
  - Lab drop-in session every Wednesday from 0900 to 1200 (weeks 2 to 11)
- Each week will have its own page on Canvas
  - Lecture slides published in advance (every Wednesday for the following week)
  - Live lecture recordings uploaded a few hours after each delivery
  - Lab sheet to work through in your own time (using lab PCs or your own device)
- Lab session is an optional drop-in (there is no attendance code)
  - Work through the lab tasks before the session
  - Come along to the session if you have any questions or problems
  - Chat to me if you have any questions about lectures or the module content

Recommended Reading

- The material on Canvas should be sufficient to complete this module
  - Wider reading is always encouraged but is not essential
  - Exam questions will only be based on things mentioned in lectures and lab sheets

- Lecture material is based on these books (all available in the Harold Cohen Library)
  - Williams – Computer Systems Architecture (2nd Ed)
  - Silberschatz – Operating System Concepts (9th Ed)
  - Flynn – Understanding Operating Systems (7th Ed)

- You can also search online for almost all concepts mentioned in the module
  - Wikipedia is a good source of material but not always explained for beginners
  - Make use of the drop-in sessions or Canvas discussion area to ask questions

Assessment

- This module is assessed with one exam in the formal exam period
  - The exam is weighted 100%
  - There is no coursework or other assessment
  - Practice exams will be released on Canvas throughout the semester

- The lab tasks are not assessed
  - Some exam questions will be based on things you do in those lab tasks
  - Most questions will be based on material from the lecture slides
  - Attempting the labs will give you a better chance to pass the exam

- Exam will be multiple choice
  - Select one correct answer for each question
  - Assembly coding questions might be based on code shown in lectures and labs

Contact Me

- If you have a general question that others would benefit from, post it in the Canvas discussion area

- I'm always very happy to chat in person about anything
  - Will hang around after lectures if you have any quick questions
  - Will be present at every lab drop-in for more in-depth questions or just to chat

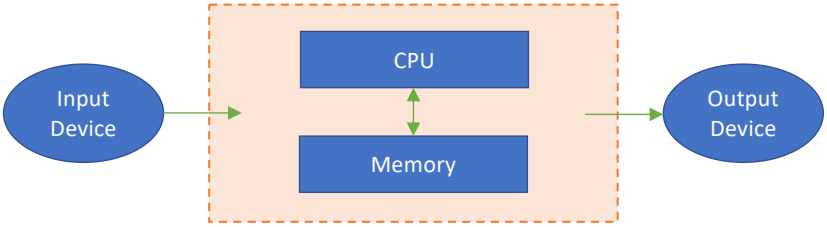
- You can also email me any time about the module or wider programme issues s.thomason@liverpool.ac.uk

- On Mondays and Wednesdays I will usually (but not always) be around in my office when not teaching, so pop along for a chat if you want to (Holt Building, Room 201B)

Module Content Begins Here

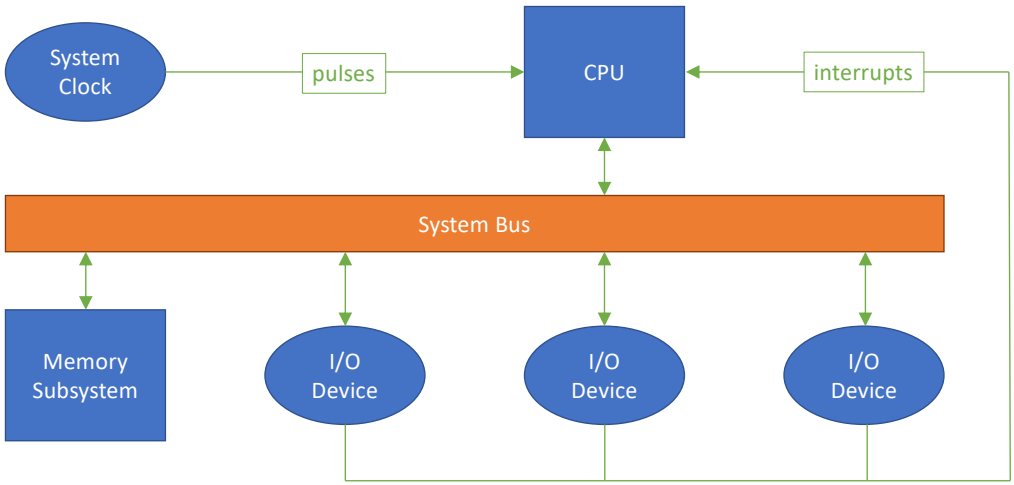
Von Neumann Model

- Basic model of a computer system was proposed by John von Neumann in the 1940s



- The input device is used to load programs and data into memory (the stored program concept)
- The CPU (central processing unit) fetches program instructions from memory, processes the data, and generates results
- The results are sent to an output device

Modern Computer Systems



Central Processing Unit (CPU)

- Fetches instructions from main memory and executes them
  - These instructions are very basic (eg. move a value, add two values, etc.)
  - Different types of CPU have different instruction sets (eg. AMD, Intel, Apple, etc.)
  - There is no standard instruction set or format, but operations are similar in all CPUs

- Two most common types of CPU instruction set
  - CISC – Complex Instruction Set – Such as Intel x86 and many desktop/server CPUs
  - RISC – Reduced Instruction Set – Such as ARM (smartphones and small IoT devices)

- Internal activity of the CPU is synchronised by a fast clock
  - Measured in hertz (megahertz, gigahertz)
  - For example, 3 GHz clock = 3 billion cycles (instructions) per second

System Bus

- The bus is a collection of wires allowing communication between the various components on the motherboard
- Without a bus, we'd have to directly connect every component to every other component, which is prohibitively complicated and expensive (point-to-point system)
- Sender places an item (data) on the bus and the receiver takes it off
- The bus can have multiple lines
  - Address lines – used to specify a memory address (or device) to be accessed
  - Data lines – carry the actual data to be transferred
  - Control lines – tell the receiver what to do with the data
- Almost all modern computers have multiple interconnected system buses (eg. SATA, PCIe, USB)
- There is a problem of bus contention because only one thing can be on the bus at once