CITY
ST GEORGE'S
UNIVERSITY OF LONDON

School of Science
& Technology

www.city.ac.uk

Systems Architecture

IN1006

Introduction to Computer systems Architecture-History

—Dr H. Asad

Learning objectives

- History of Computers
- (Computer) Systems Architecture
- von Neumann components of a computer system
- Basic hardware components of a computer
- Semiconductors
- Introduction to Moore’s law and technology trends
- Concept of Abstraction
- Computer system in terms layers of functionality

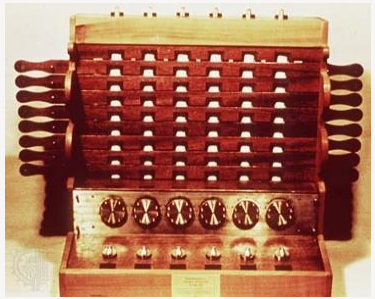
Can you compare Computers to a factory?

- Who will play the part of CPU?
- Can you think of a place in a factory that works like a RAM?
- What is the hard drive equivalent in a factory?
- What is a motherboard equivalent in a factory?

Early Devices

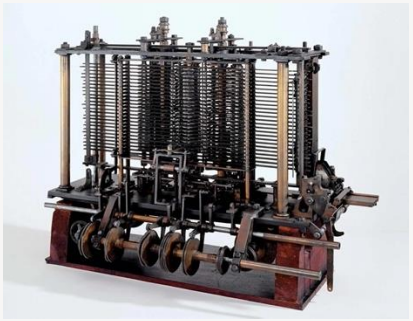
- **Mechanical calculating machines (1642 – 1945)**
  - Mechanical **Calculating Clock**:
    - Add, subtract 6-digit numbers (W. Schickard<1635)

- **Difference Engine** and **Analytical Engine** by Charles Babbage (1791-1871) (the father of computing?)
- **Analytical Engine** had many of the components of modern computers: the mill (like the ALU), the store (like memory) and input and output devices



Calculating Clock

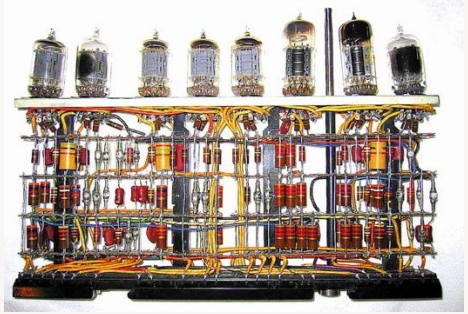
Difference Engine



Analytical Engine

History of Computers

- **First generation: Vacuum Tube Computers (1945 – 1953)**
  - Valves/**Vacuum tubes**
  - ENIAC 1946, the first all-electronic, general purpose computer with **17,468 vacuum tubes: 1800 square feet** of floor, 30 tons, **174 kilowatts of power**. It had **1,000 information bits of memory**.

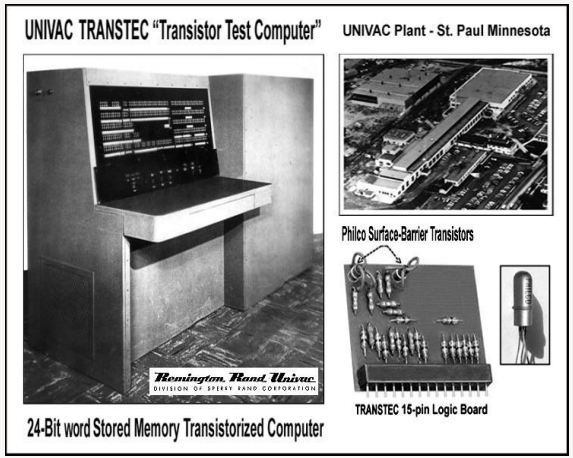


IBM Vacuum Tube Computer.

http://www.chipsetc.com/

History of Computers (cont'd)

- **Second generation: Transistorised computers (1955 – 1965)**
  - **Transistors** (1948) (televisions, radios, computers)
  - Computers became **smaller, faster** and consumed **less power**
  - But they still were **bulky** and **expensive.**
  - First computers: IBM 7094, DEC PDP-1, CDC 6600



History of Computers (cont’d)

- **Third generation: Integrated Circuit Computers (1965 – 1980)**
  - **Integrated circuits** or microchip, silicon chips
  - Each IC contains **dozens of ‘Integrated transistors'**
  - Computers became **faster,** smaller and **cheaper**
  - IBM 360, DEC PDP-11, Cray 1

Integrated Circuit Computers

www.timetoast.com

History of Computers (cont'd)

- **Fourth generation: VLSI computers (1980 – Now)**
  - Very Large Scale Integration **(VLSI): 10,000 components per chip**
  - 1981, IBM introduced the Personal Computer PC
  - Apple Mac (Graphics)

- **Modern computers**
  - **System on Chip (SoC)** when all components of a computer in one chip
  - **Special processors (graphics), new types of memory**
  - Laptops, tablets, smartphones, smart watches

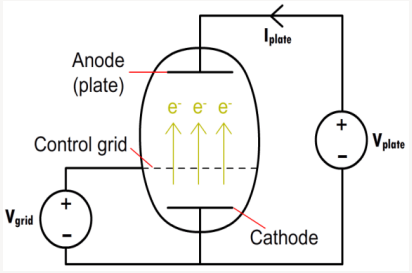
Very Large Scale Integration

Apple A4 chip built in on a smart phone

Computer History Driver: Technology of Computer Processors (& Memory)

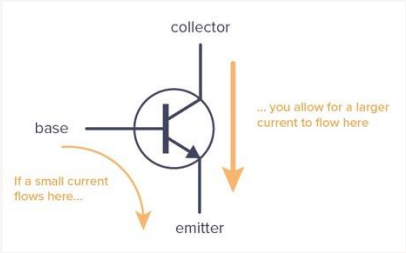
- Computers improved as the underlying technology changed:

Mechanical ➔ Vacuum tube ➔ Transistor ➔ Integrated Circuit ➔ Very Large Scale Integration ➔ System on Chip



[1]

Switch functionality



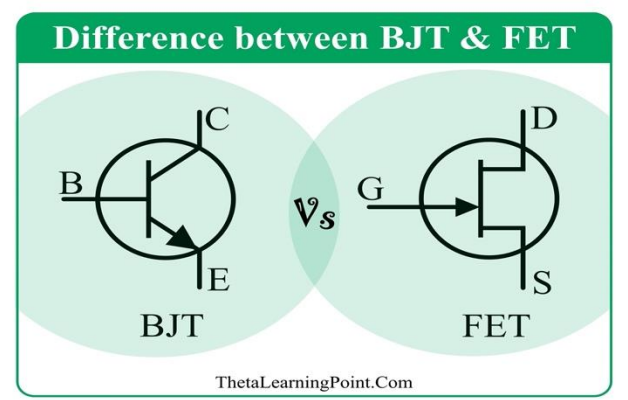
[2]

The most **significant factor** that allowed this progress was the move to the use of **Semiconductors**

[1] https://www.engineering.com/ElectronicsDesign/ElectronicsDesignArticles/ArticleID/16337/Vacuum-Tubes-The-World-Before-Transistors.aspx
[2] https://www.build-electronic-circuits.com/how-transistors-work/

Semiconductors

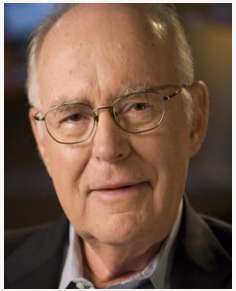
- A **type of material**
  - **Conductors** – allow electricity to flow
  - **Insulators** – prevent the flow of electricity
- They can **switch** between these two states **under external control**
- can be used to **build transistors that function as switches** and make all the basic logic of a computer
- **Silicon** is a semiconductor material that can be manufactured and processed efficiently and **inexpensively**



This is why all modern electronics are referred to as “Silicon Chips”

Moore's law

- Moore’s law: **Transistor count (in IC/Silicon Chips) will double approximately every 18-24 months.** Gordon Moore (1965)
- **Moore's law** predicts that this trend will continue into the foreseeable future
- Shrinking transistors have powered over **60 years advancement** in computing power, but its time to find other ways to improve computing power
- **(cloud computing, high performance computers, quantum computing?)**



Gordon Moore
Co-founder of Intel

Computer Systems Architecture

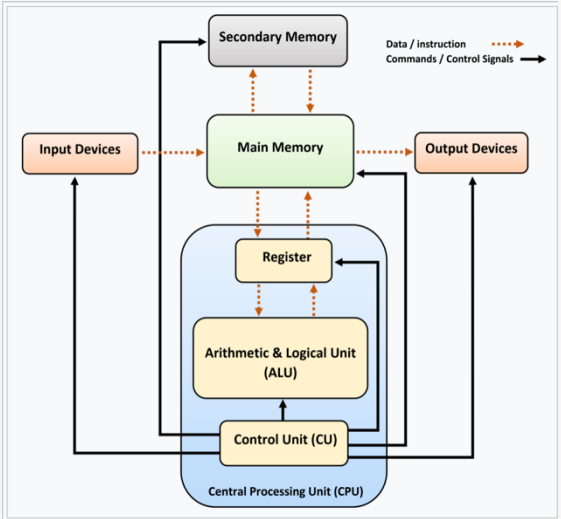
What is it?

- ***fundamental organization*** *of a (computer) system, embodied in its* ***components, their relationships*** *to each other and the* ***environment,*** *and the* ***principles*** *governing its* ***design*** *and* ***evolution.***

Why should it be studied?

It enables an **understanding** of

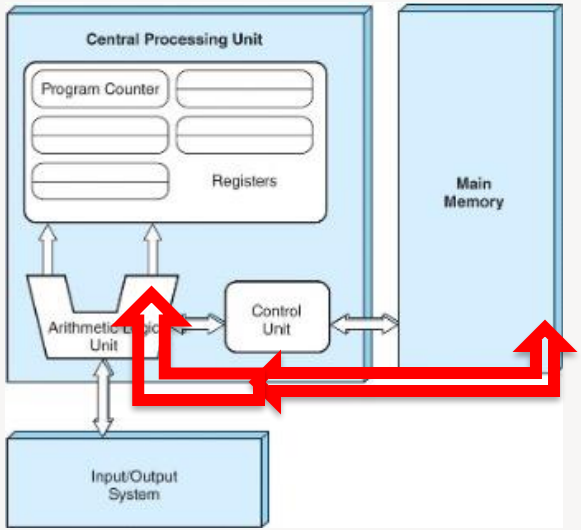
- the **fundamental capabilities** of a computer system
- how these capabilities **are realized** by the **components**
- how a computer **system** may be **altered a**nd **extended**
- **what effect such actions** may have on its capabilities



Source:https://en.wikipedia.org/wiki/Computer_architecture

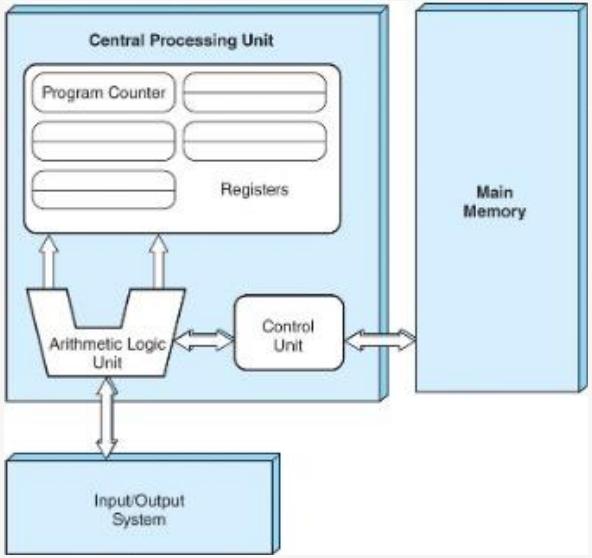
The von Neumann model: an abstract model

- All **modern stored-program computers** are based on the von **Neumann model**
- Called **stored program digital computer**.
- It consists of **five components:**
  - Control Unit (controls the flow of data between CPU and other the devices)
  - Arithmetic Logic Unit(ALU) binary numbers operations.
  - Registers
  - Main Memory System
  - I/O System
- carrying out s**equential instruction processing.**
- A single **datapath** between the **CPU** and main memory
- This single path is known as **the *von Neumann bottleneck***

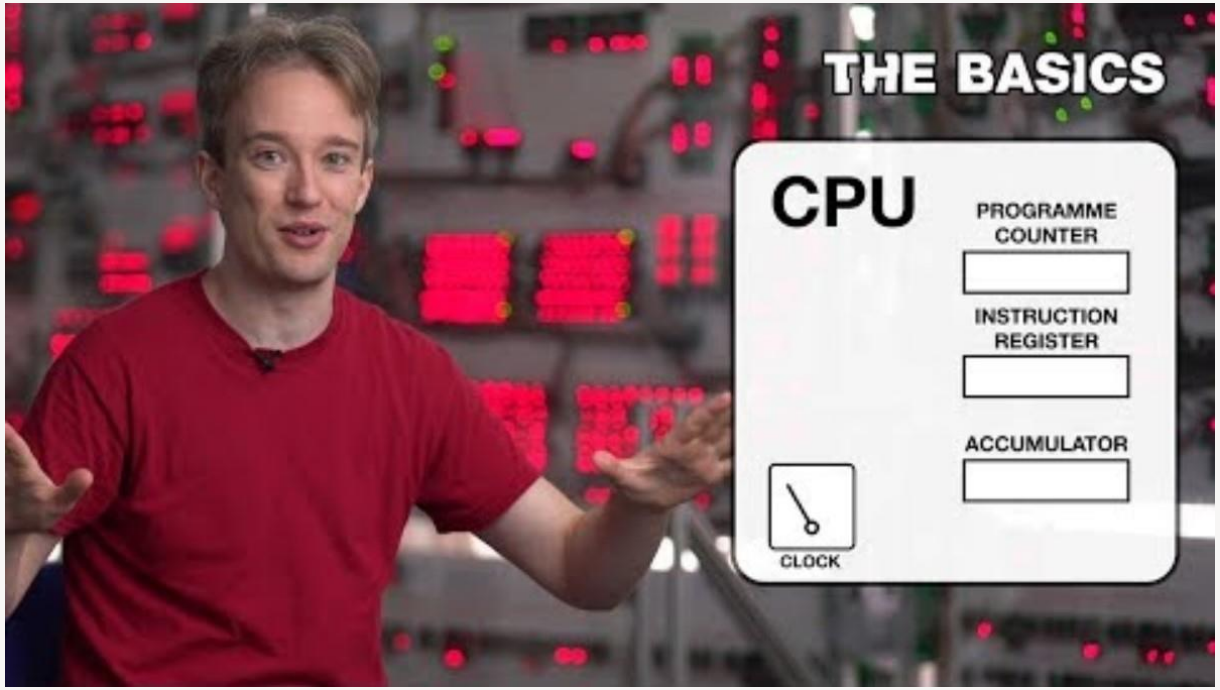


The von Neumann model (cont'd)

- The von Neumann model (**fetch-decode- execute** cycle):
- Control unit **fetches** the **next instruction** to execute from memory
- Instruction is **decoded**
- Any required **data operands** are **fetched** from memory to registers
- The ALU **executes** the instruction and puts results in registers/memory
- Go back to **(1)** for next instruction



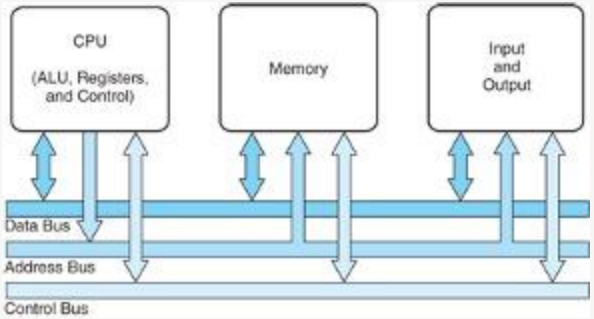
The Fetch-Execute Cycle: What's Your Computer Actually Doing?



The von Neumann model has evolved …

- **Incremental Improvements over the years**
- **specialised buses**, **floating-point units, cache memories**
- **require departure from** the classic von Neumann architecture:
- Adding processors
- Separate buses for data and instructions **(Harvard architecture)**

- cache memory - temporary storage for frequently used instructions and data for quicker processing



The system **bus model**

Computer Hardware Components

Personal Computer

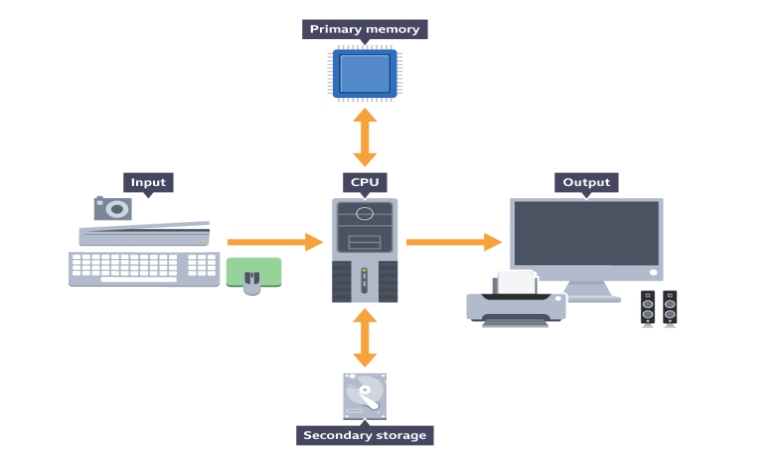
- Main System
- CPU, RAM(random access memory), disks, etc.

Input Devices

- E.g. keyboard, mouse, scanners, camera etc.

Output Devices

- E.g. monitor, speakers, printers etc.



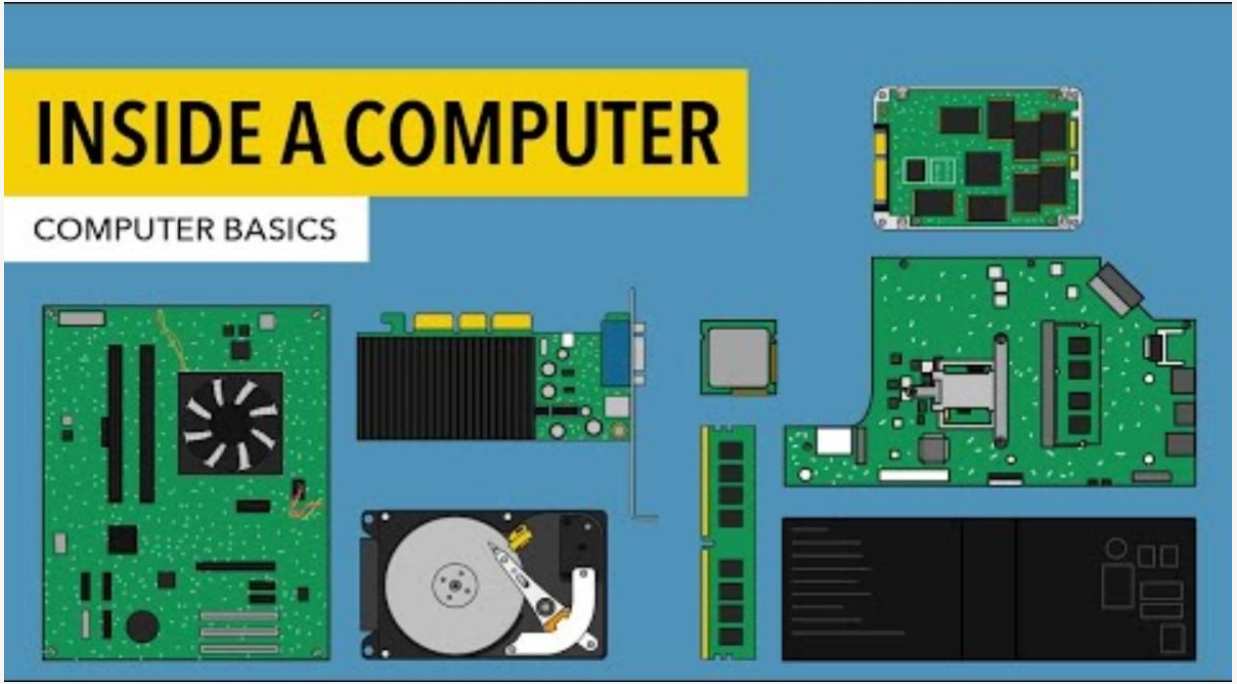
https://www.bbc.co.uk/bitesize/guides/zbfny4j/revision/1

CITY
ST GEORGE'S
UNIVERSITY OF LONDON

School of Science
& Technology

www.city.ac.uk

Inside a computer



Key Processor Components

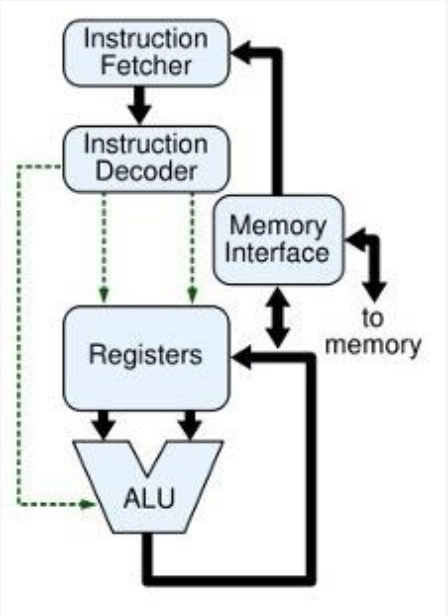
Registers
- local storage of key data

Memory Management Unit (MMU)
- Interface to main memory

Instruction fetcher/decoder
- Fetches and decodes instruction given at program counter (PC).

Arithmetic Logic Unit (ALU)
- Performs calculations/operations

Datapath / control
- Controls the flow of information that allows the instruction to be executed.



CITY
ST GEORGE'S
UNIVERSITY OF LONDON

School of Science
& Technology

www.city.ac.uk

Inside a Modern CPU

Another key factor in the evolution of computer history: “Abstraction”

- Act of representing **essential features without including the background details**
- the abstraction principle is used to **reduce complexity** and **allow efficient design and implementation** of complex software systems.

Abstraction in programming

Abstraction in modelling

Abstraction in networks

What is “Abstraction”? Practically speaking

rect();

```
jmp start
;=========================================
; Basic program to draw a rectangle
;=========================================
  mode          db      18      ;640 x 480
  x_start       dw      100
  y_start       dw      100
  x_end         dw      540
  y_end         dw      380
  colour        db      1       ;1=blue
;=========================================
start:
  mov ah,00                     ;subfunction 0
  mov al,mode                   ;select mode 18 (or 12h if prefer)
  int 10h                       ;call graphics interrupt
;=========================
  mov al,colour                 ;colour goes in al
  mov ah,0ch
  mov cx, x_start               ;start drawing lines along x
drawhoriz:
  mov dx, y_end                 ;put point at bottom
  int 10h
  mov dx, y_start               ;put point on top
  int 10h
  inc cx                        ;move to next point
  cmp cx, x_end                 ;but check to see if its end
  jnz drawhoriz
drawvert:                       ;(y value is already y_start)
  mov cx, x_start               ;plot on left side
  int 10h
  mov cx, x_end                 ;plot on right side
  int 10h
  inc dx                        ;move down to next point
  cmp dx, y_end                 ;check for end
  jnz drawvert
;=========================
readkey:
  mov ah,00
  int 16h                       ;wait for keypress
;=========================
end:
  mov ah,00                     ;again subfunc 0
  mov al,03                     ;text mode 3
  int 10h                       ;call int
  mov ah,04ch
  mov al,00                     ;end program normally
  int 21h
```

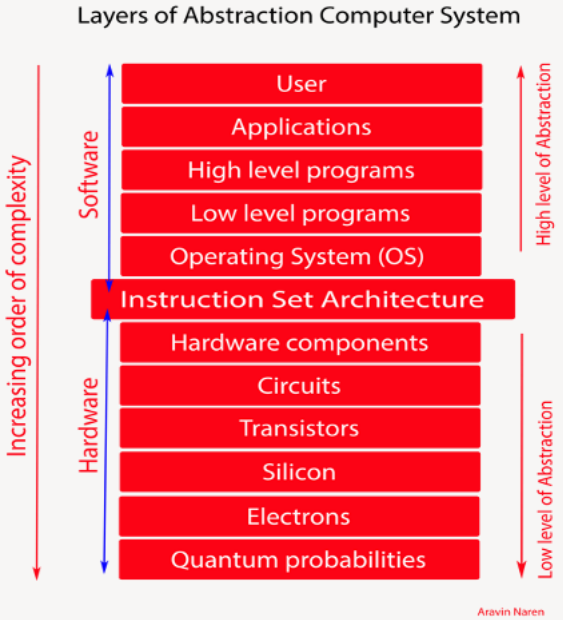
Machine code (0s and 1s !!!)

Key Layers of Abstraction in Computer Systems

- Systems can be **decomposed into layers**
- **abstraction** allows layers to **hide lower level information from higher levels**
  - higher levels to be **useful** and **consistent** without being cluttered by unnecessary detail
- You don't want to think about electron flow when writing a games program!

BUT

- Sometimes you need to break the abstraction to get things right – e.g., optimal performance of the game requires detailed knowledge of the caching architecture



In summary:

In this lecture, we have

- Briefly looked at the history of computers and technology trends
- Presented an abstract model of computers
- Presented the architecture of computer systems and key hardware components
- Introduced the need for and the key layers of abstraction in computer systems, and presented the advantages arising from abstraction

School of Science & Technology
City, University of London
Northampton Square
London
EC1V 0HB
United Kingdom

T: +44 (0)20 7040 5060
E: SST-ug@city.ac.uk
www.city.ac.uk/department

Acknowledgement:
Prof George Spanoudakis
Andrew Tuson
Kevin Jones
Eva Kalyvianaki
Aravin Nathan