3. The von Neumann Architecture

History

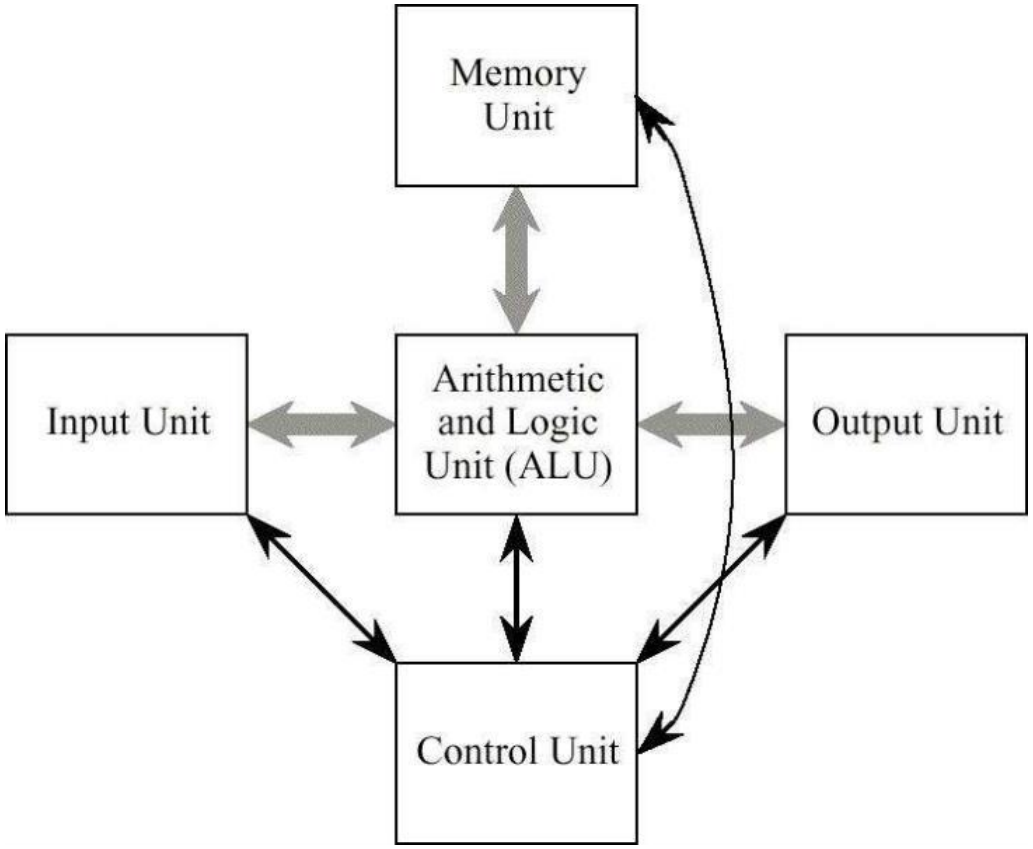
- Early computers were not (re)programmable
- To change the instructions to be performed, machine must be entirely rebuilt. E.g., ENIAC: It took three weeks to rewire in order to do a different calculation
- In 1945, just after the World War, John von Neumann proposed to build a more flexible computer
- Not only the data should be stored in memory, but also the program should be stored in the same memory (stored-program computer)
- In 1948, Manchester Small-Scale Experimental Machine (SSEM) made the first successful run of a stored-program

The von Neumann Architecture

- A stored program computer in which the program and the data are kept in the **same memory** is said to have the **von Neumann architecture**
- Effectively the program by itself is treated as data
- A computer built with this architecture would be much easier to re-program
- Virtually every general purpose modern day computer is based on the von Neumann architecture
- Non von Neumann architectures (e.g. Harvard architecture) are only used in systems with very specialist requirements

The von Neumann Architecture

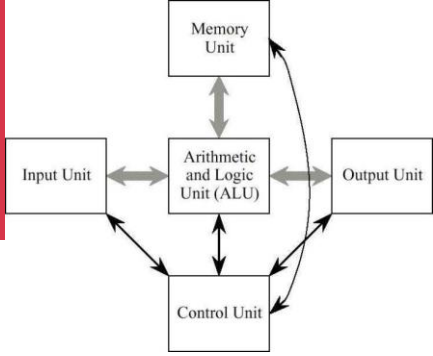
- Five key components:
  - Control unit
  - Arithmetic-logic unit
  - Memory unit
  - Input unit
  - Output unit



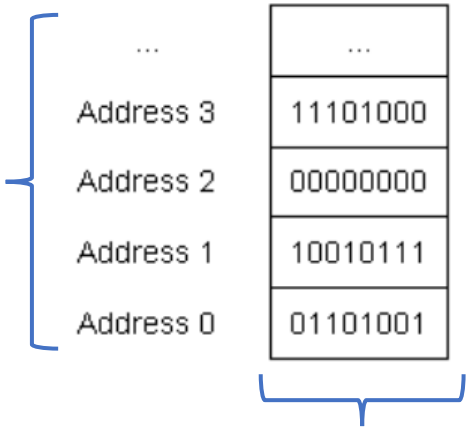
(Thick arrows show data paths and thin arrows represent control paths)

CARDIFF UNIVERSITY
PRIFYSGOL CAERDYDD

Memory Unit



- Memory is a collection of **cells**, each having an **address**



- An address uniquely identifies a memory cell. No two different memory cells can have the same address
- The number of bits in each addressable location is known as the memory's addressability
- Example: the x86 is byte-addressable, i.e., there is one address for every 8 bits (byte)

Memory Unit

- Most computers have a characteristic unit quantity of data, which can be handled most efficiently/naturally by the computer – the **machine word**. Usually:
  - Instructions operate on a word at once
  - Data is fetched one word at time
- Most machines are byte-addressable (1 byte = 8 bits)
- Typically a word is an integer number of bytes, and often are 2 bytes (16 bits), 4 bytes (32 bits), and 8 bytes (64 bits)

Memory Unit

- Example:

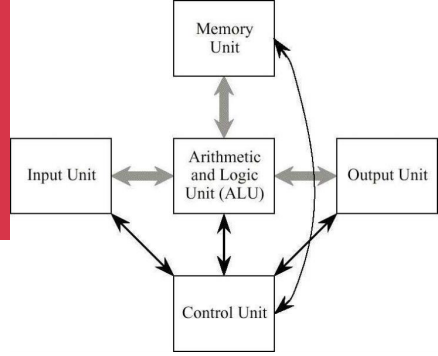
A computer has 9-bit words, in the sense that memory cells store words (word-addressable computer). The size of the addresses is 11 bits. What is the max memory capacity of this computer, in bytes?

Address	Data
00000000000	101010011
00000000001	110010110
…	…
11111111111	100011001

- Solution:

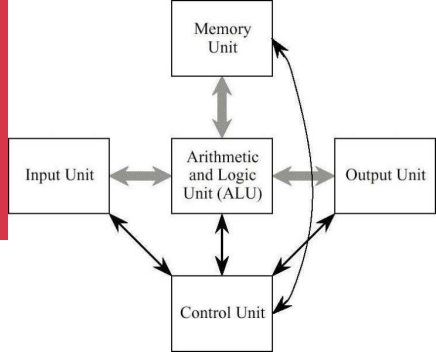
- There are 2^{11}=2048 possible addresses, which is the maximum number of (addressable) memory cells.
- Each cell stores one 9-bit word, which yields 9×2048=18432 bits.
- There are 8 bits in a byte, therefore the final answer is 18432/8=2304 bytes.

Input/Output Units



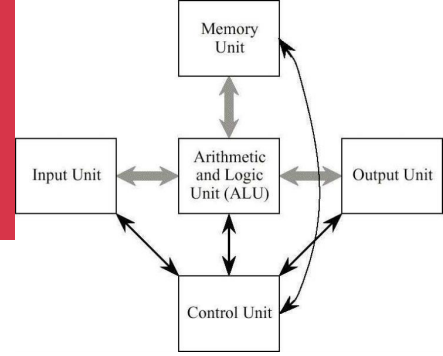
- The input unit:
  - Allows data and program to be entered into the computer
  - The very first input units received data via punched cards
  - Modern computers receive input into them from a range of sources, e.g., keyboard, mouse, scanner etc.
- The output unit:
  - Is a device through which results stored in the computer memory are made available to the outside world
  - Modern computers provide output to a range of devices, e.g., screen, printer etc.

Arithmetic-Logic Unit (ALU)



- Capable of performing basic arithmetic operations:
  - +, −, ×, ÷
- It is also capable of performing logic operations:
  - AND, OR, NOT, <, =, >
- Most modern ALUs have a small amount of special storage known as **registers**, which are used to store information that will be needed again immediately
- Typically, each register can hold one word of data

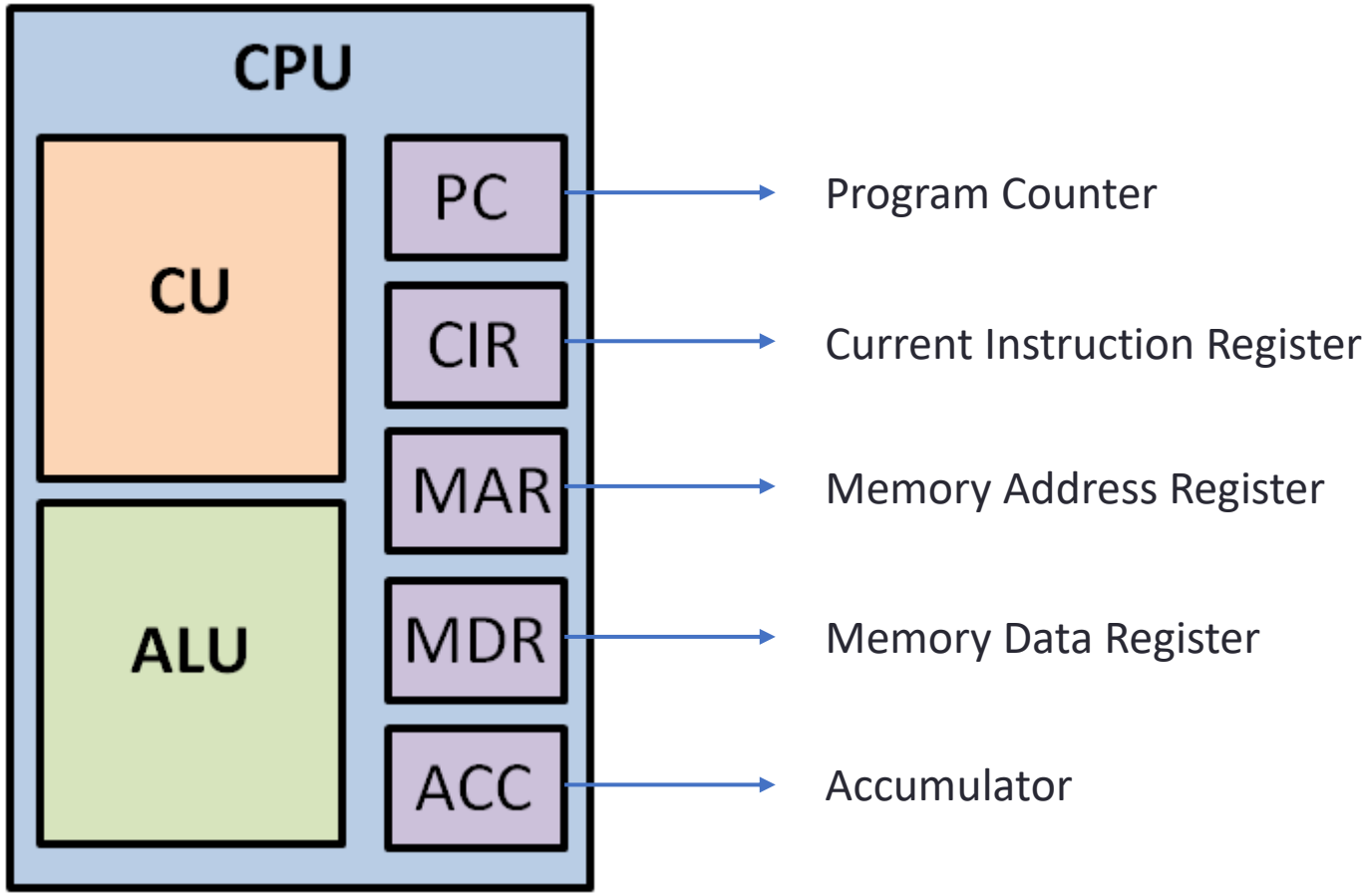
Control Unit (CU)



- This is the organising force within a computer system
- It controls the fetch-execute cycle (will be explained shortly)
- The control unit contains two very important registers:
  - The **Current Instruction Register (CIR)** – at any point in time this register holds the instruction that is currently being executed by the computer
  - The **Program Counter (PC)** – at any point in time this register holds the location of the next instruction to be executed by the computer

Central Processing Unit (CPU)

- CU and ALU are combined into Central Processing Unit (CPU)



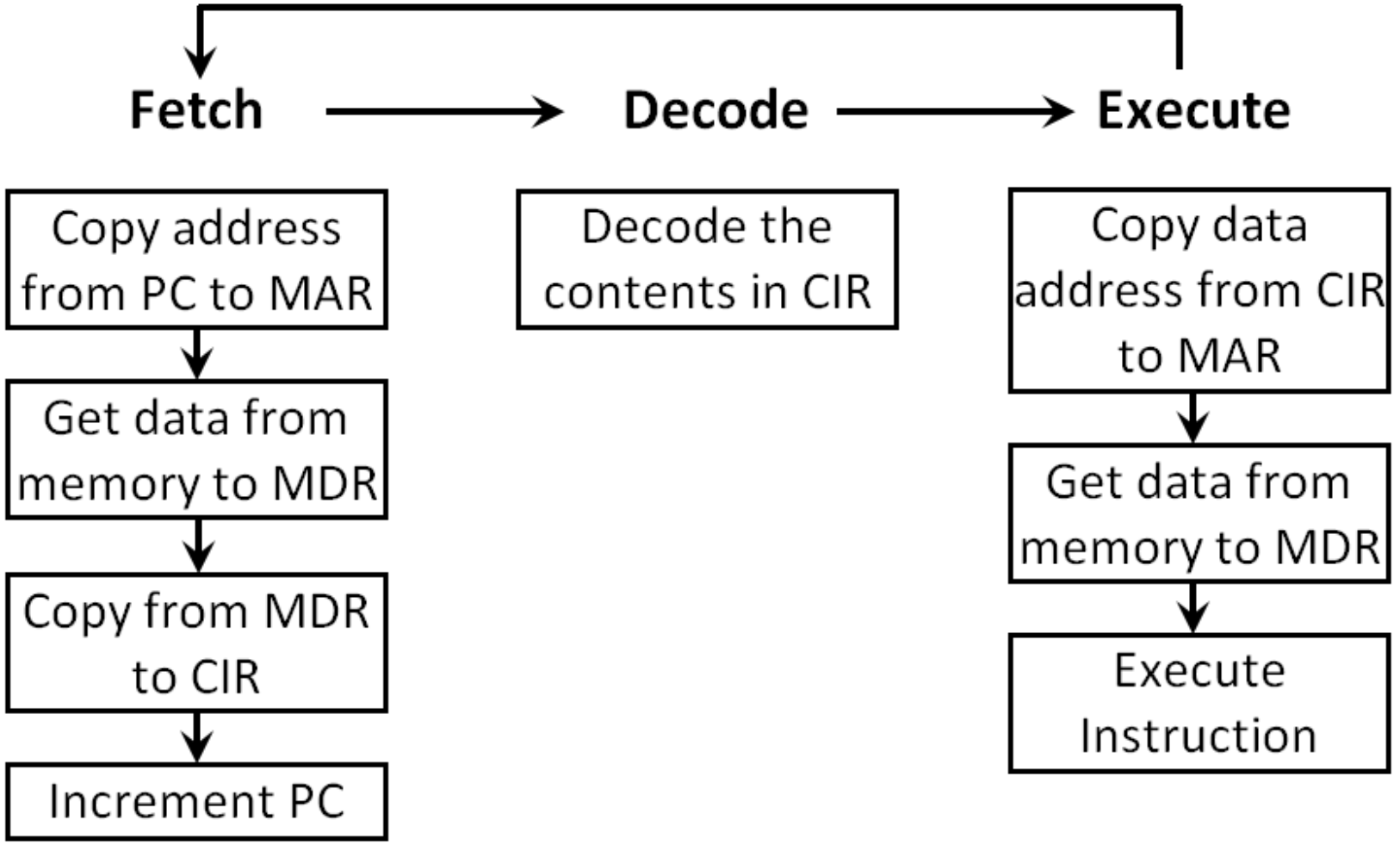
Execution of Instructions

- Let us assume that a program has been converted into a set of machine instructions, and the instructions have been loaded into the main memory
- The CPU will carry out the sequence of instructions
- It first records the address of the first instruction of the program in its **PC**, and then proceeds to carry out the following steps:

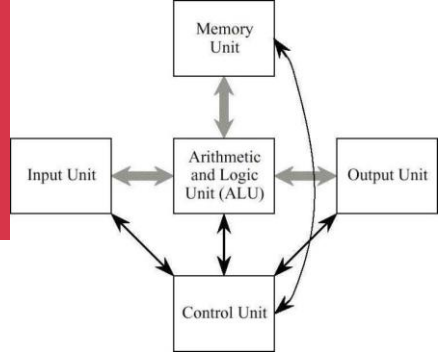
Fetch-Decode-Execute Cycle

1. Copy address from **PC** to **MAR**
2. Get data from memory to **MDR**
3. Copy from **MDR** to **CIR**
4. Increment **PC**
5. Decode the contents in **CIR**
6. Copy data address from **CIR** to **MAR**
7. Get data from memory to **MDR**
8. Execute Instruction
9. If required, result must be stored in main memory
10. Repeat from step 1

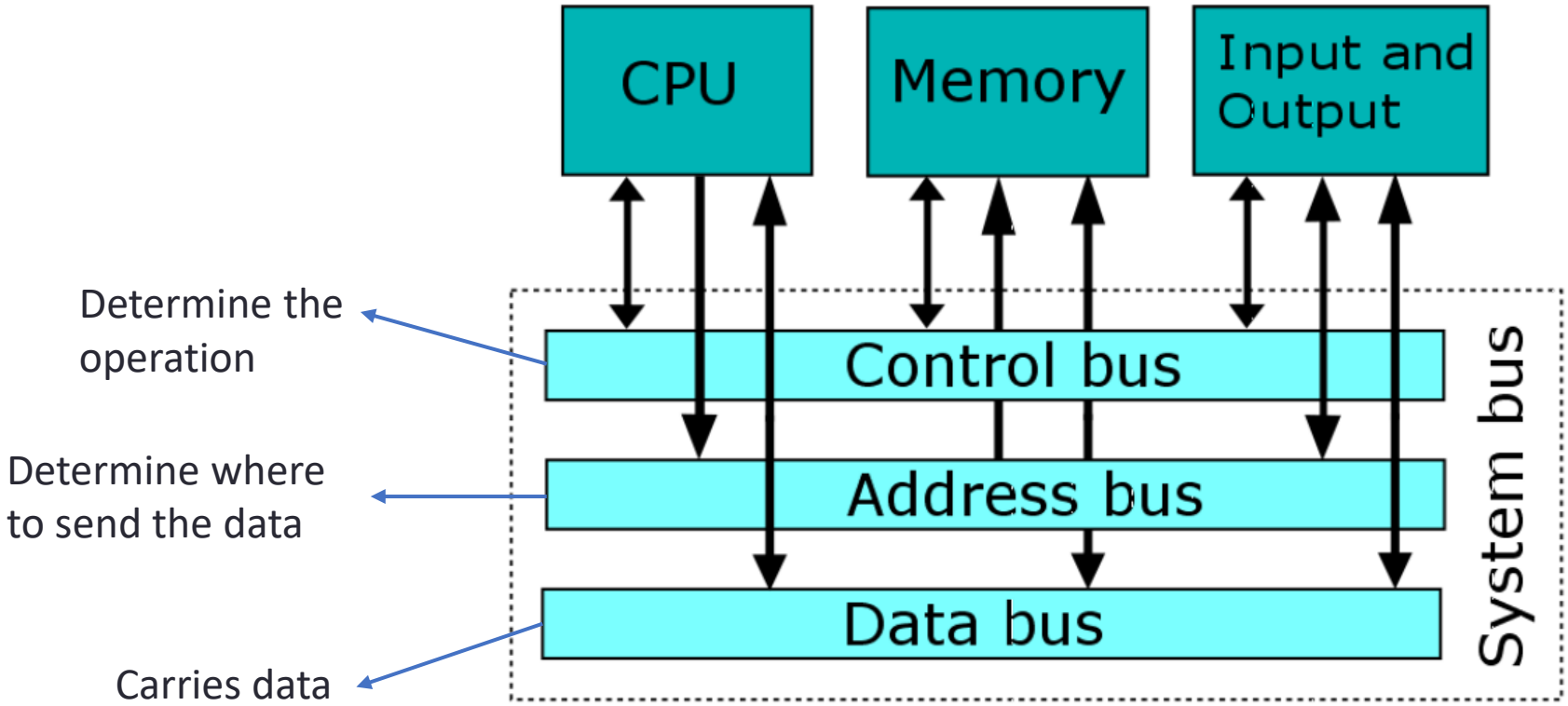
Fetch-Decode-Execute Cycle



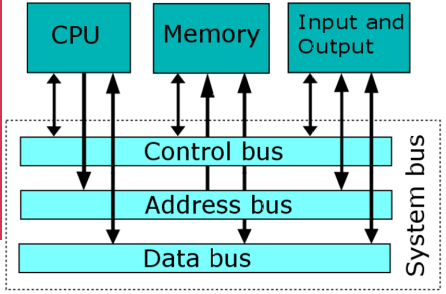
The System Bus Model



- The system bus model is a streamlined von Neumann model with a set of communication lines, known as a **bus**



The von Neumann Bottleneck



- The separation of the CPU and memory results in the **von Neumann bottleneck**
- Instructions are treated as data, and they share the data bus
- The shared bus is busy when fetching/writing data from/to memory, so the CPU has to wait before fetching the next instruction, and vice versa
- There is a limited throughput (transfer rate) between the CPU and main memory
- Memory is physically far away from the CPU and the signal propagation speed is finite

The von Neumann Bottleneck

- Since CPU speed and memory size have increased much faster than the throughput (transfer rate) between them, the bottleneck has become more of a problem on modern fast computers

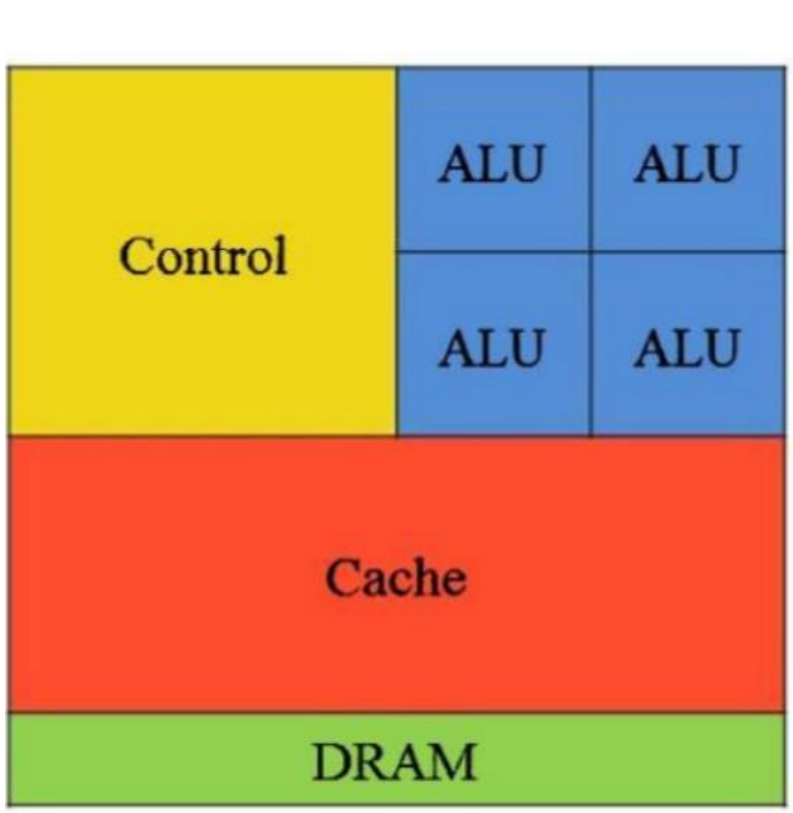
- The von Neumann bottleneck limits the effective processing speed of the CPU

- It is particularly pronounced in scenarios when it is required to carry out minimal processing on large quantities of data. The CPU is continually forced to wait for data transfer

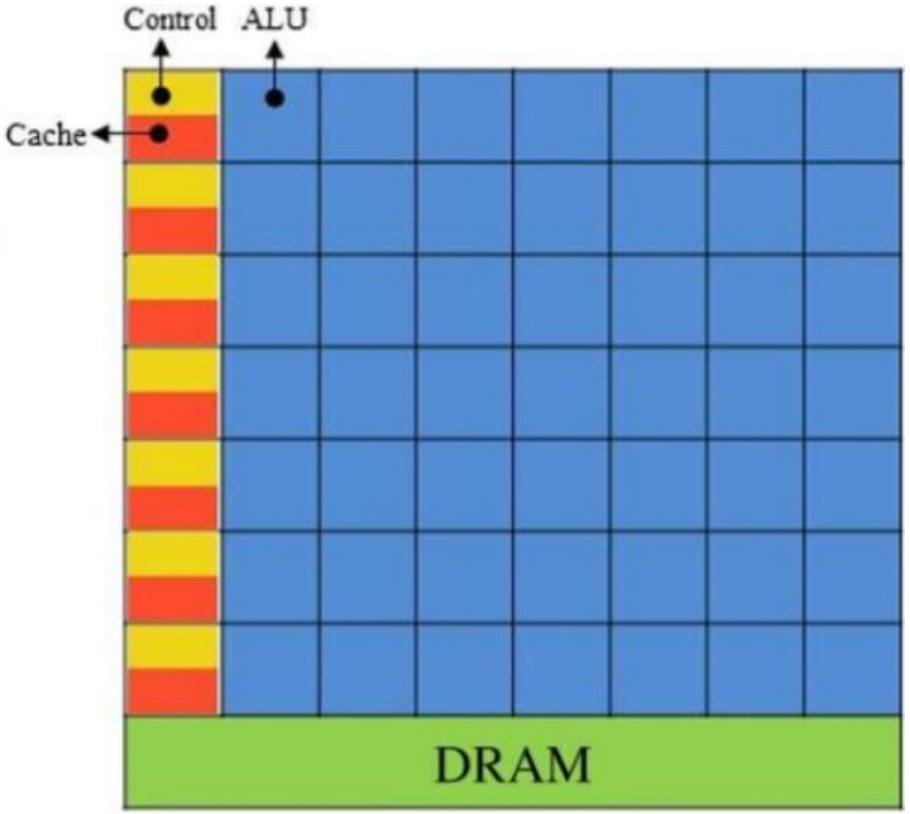
The von Neumann Bottleneck

- The bottleneck is a serious drawback for the computers we use today
- Possible solutions:
  - Data/instructions cache: a piece of smaller faster memory to store copies of frequently used data or instructions
  - Parallel computing: to do multiple calculations simultaneously. Levels of parallelism:
    - One instruction executed on multiple data
    - Multiple instruction executed on multiple data

CPU vs GPU



CPU



GPU

CPU vs GPU

"GPU computing is the use of a GPU (graphics processing unit) together with a CPU to accelerate general-purpose scientific and engineering applications. CPU + GPU is a powerful combination because **CPUs** consist of a **few cores** optimized for **serial processing**, while **GPUs** consists of **thousands of smaller, more efficient cores** designed for **parallel performance**. Serial portions of the code run on the CPU while parallel portions run on the GPU."
(Source: nvidia.com)

Can Fast Computers Be Big?

- Before the invention of ICs, more components meant physically larger computers
- Even sci-fi writers of the 1960s imagined powerful computers of the future to be big:

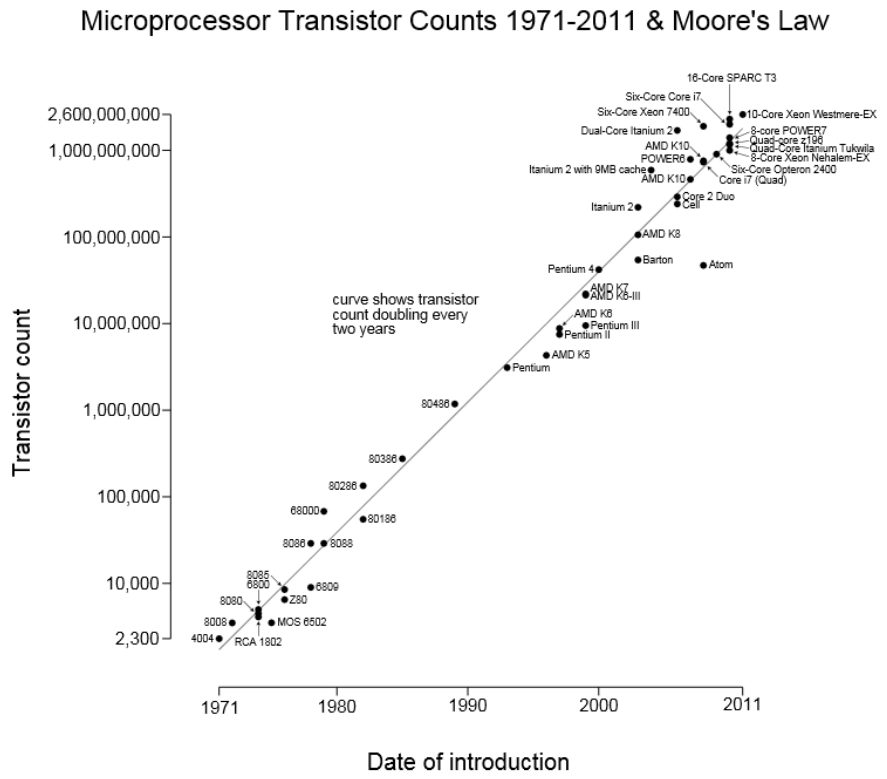
- But can fast computers really be big?

Size vs. Speed

- Signals (e.g. between parts of a computer) cannot travel faster than the speed of light, c = 299,792,458 m/s
- The larger the computer the more the time it takes to send information from one part to another!
- Example: Assume a signal has to be sent between terminals (3.5 cm apart) at 3 GHz
- s=v×t --> t=1/(3×10^9)sec
- Light travels only **10cm** during this time!
- Main reason for **miniaturisation**.

Moore's Law

- Attributed to an observation by Intel co-founder Gordon **Moore** in 1965.
- **Moore's law** is the observation that the number of transistors in a dense integrated circuit doubles approximately every two years
- But **Moore's law** is coming to an end!



Miniaturisation

- Size of a silicon atom = 111 pm = 1.11×10-10 m.
- Current transistor size on ICs ≈ 10nm = 1.00×10-8 m.
- So, the size of a transistor on ICs is only about 100 silicon atoms across!

  ...in terms of size of transistors you can see that we're approaching the size of atoms which is a fundamental barrier...

  – G. Moore

- So, fast computers must be small, but cannot be infinitely small. Solution? Probably parallel computing? Quantum Computing?

New Moore's Law?

- An observation: Advancements in GPU are growing at a rate much faster than with traditional CPU, in contrast to Moore's Law
- According to Jensen Huang (CEO of Nvidia), Nvidia's GPUs were "25 times faster than five years ago", whereas Moore's Law would have expected only a ten-fold increase
- Is this a new "Moore's Law"?