EXAMPLE OF AMDAHL'S LAW

Compute the Overall speedup if we make 90% of a program run 10 times faster.

$$SU_O = \frac{1}{(1-\Delta) + (\frac{\Delta}{SU_\Delta})}$$

$\Delta = 90\% = 0.9$

$SU_\Delta = 10$

$$SU_0 = \frac{1}{(1-0.9) + (\frac{0.9}{10})} = \frac{1}{0.1 + 0.09} = \frac{1}{0.19}$$

$SU_O = 5.26$

Measuring Performance

- Time needed to execute a chosen instruction mix
- Computer *X* is *n* times faster than computer *Y* means that for an instruction mix IM

$$\frac{ExecutionTime_{IM}(Y)}{ExecutionTime_{IM}(X)} = n$$

$$Performance_{IM}(X) = \frac{1}{ExecutionTime_{IM}(X)}$$

$$\frac{Performance_{IM}(X)}{Performance_{IM}(Y)} = n$$

"X is N% faster than Y."

$$\frac{\text{Execution Time of Y}}{\text{Execution Time of X}} = 1 + \frac{N}{100}$$

Compute the Overall speedup if we make 80% of a program run 20% faster.

Amdahl's law can be generalized as shown below to account for the case whereby a number of different independent enhancements can be applied separately and for different fractions of the time, $\Delta_1, \Delta_2, \Delta_3 \ldots . \Delta_n$ thus leading respectively to the speedup enhancements

$SU_{\Delta_1}, SU_{\Delta_2}, SU_{\Delta_3} \ldots . SU_{\Delta_n}$

$$SU_O = \frac{1}{(1-(\Delta_1+\Delta_2+\ldots+\Delta_n)+(\frac{(\Delta_1+\Delta_2+\ldots+\Delta_n)}{(SU_{\Delta_1}+SU_{\Delta_2}+\ldots SU_{\Delta_3})}}$$

MEMORY ARCHITECTURE

Information stored by the memory can be divide into two:

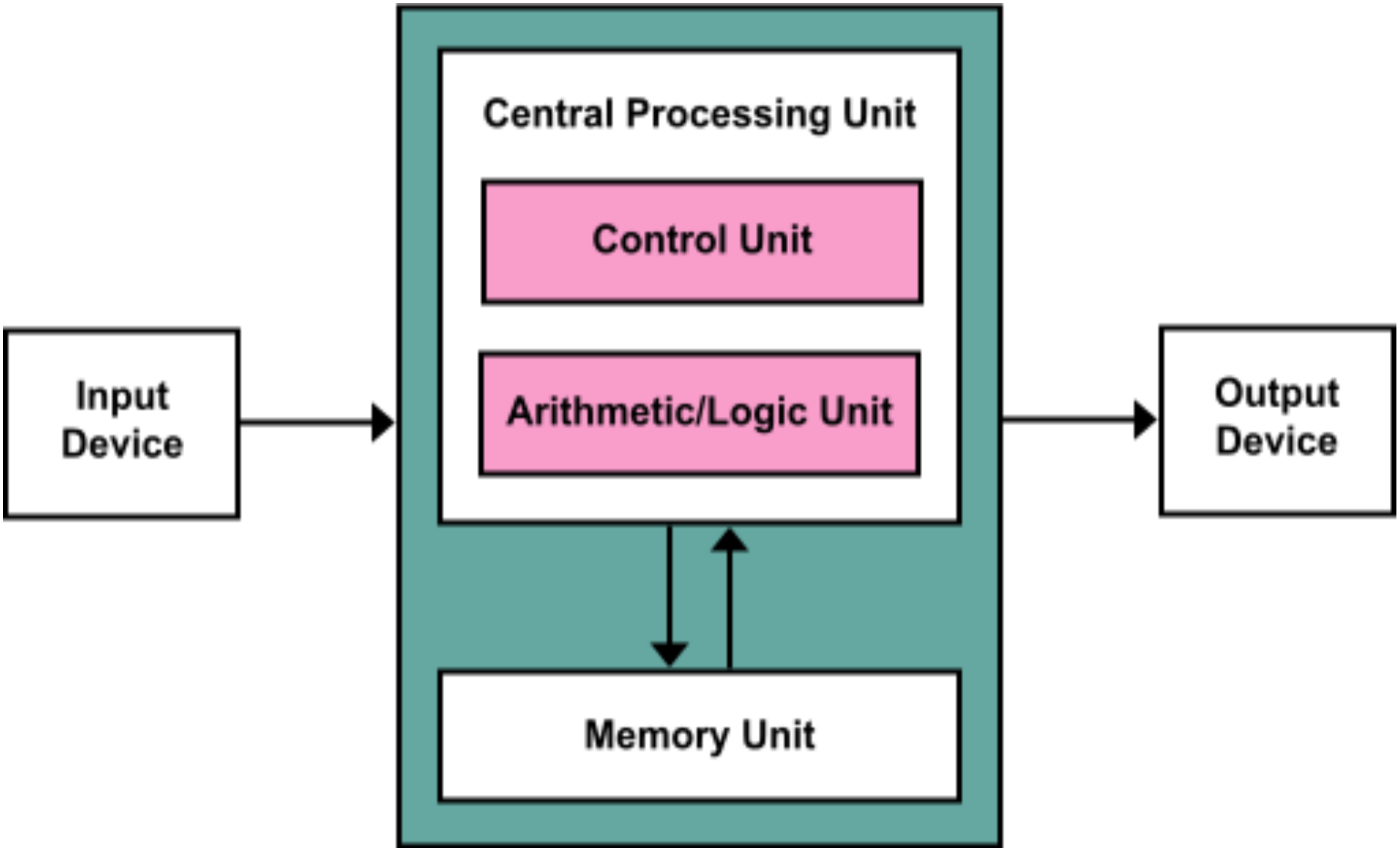
- Instruction/ program
- Data

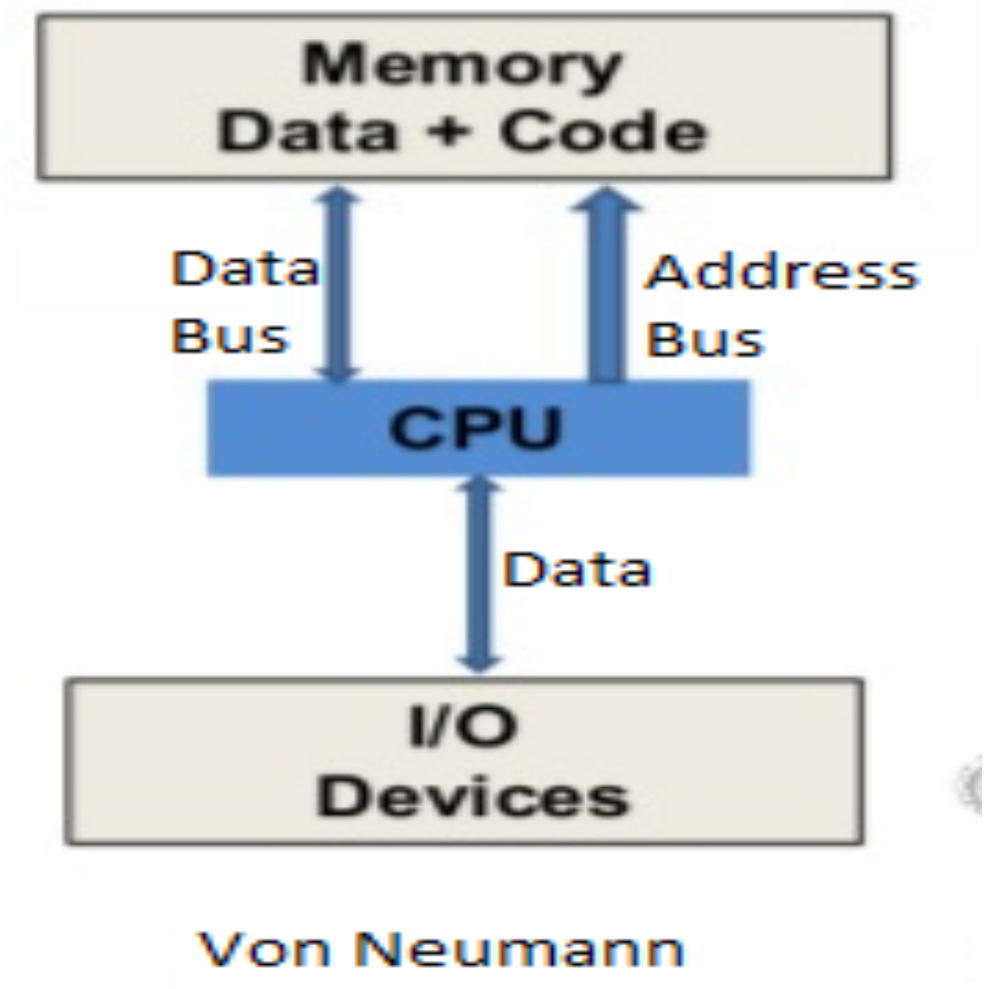
There are two different architectures employ by the memory to store data and instruction

- Von Neumann architecture
- Harvard Architecture

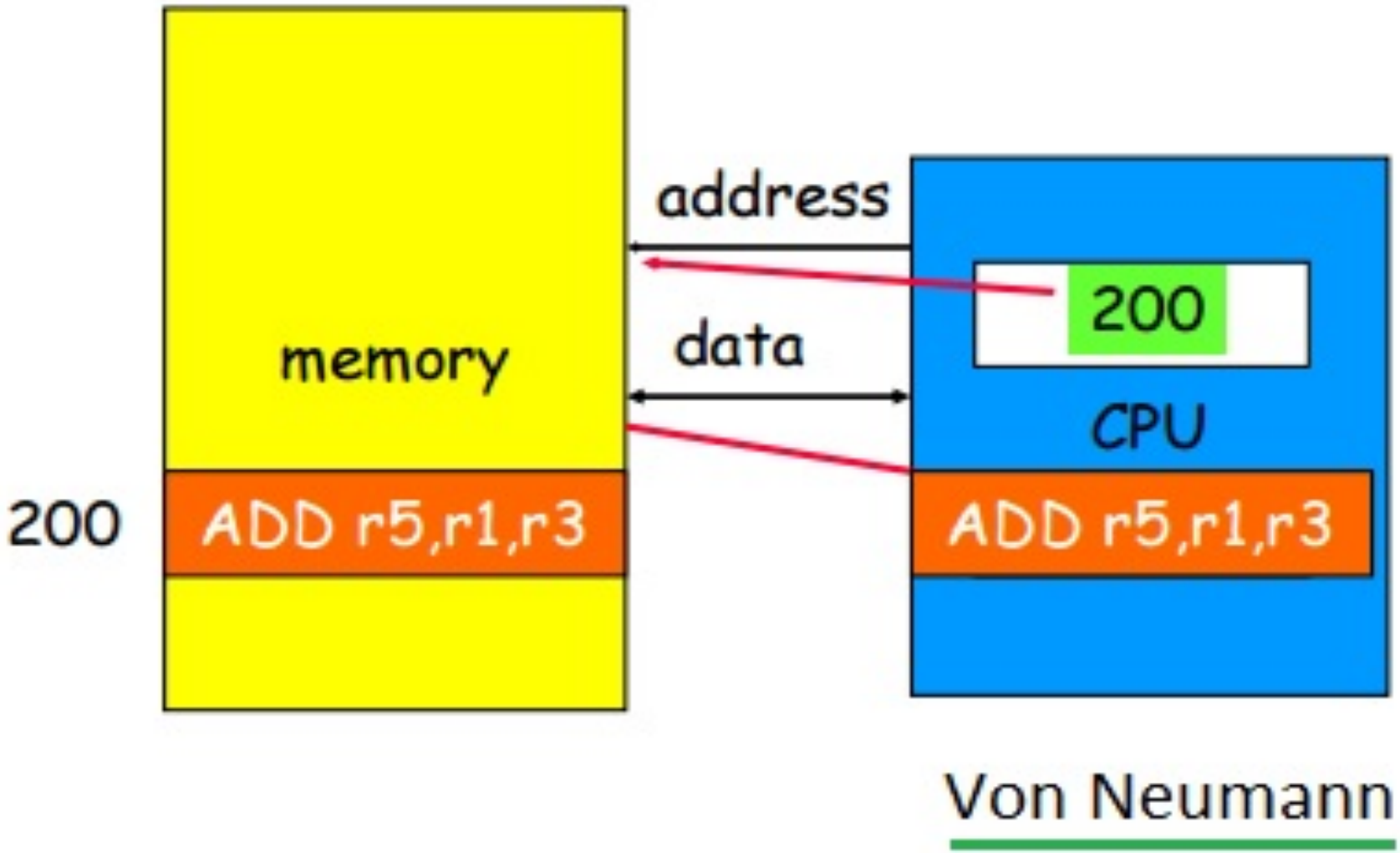
VON NEUMANN ARCHITECTURE

- The computer has single storage system(memory) for storing data as well as program to be executed.
- Processor needs two clock cycles to complete an instruction.
- In the first clock cycle the processor gets the instruction from memory and decodes it.
- In the next clock cycle the required data is taken from memory.
- For each instruction this cycle repeats and hence needs two cycles to complete an instruction.
- A single set of address/data bus is used fetching data and instruction by the CPU


Central Processing Unit
Control Unit
Arithmetic/Logic Unit
Input Device
Output Device
Memory Unit

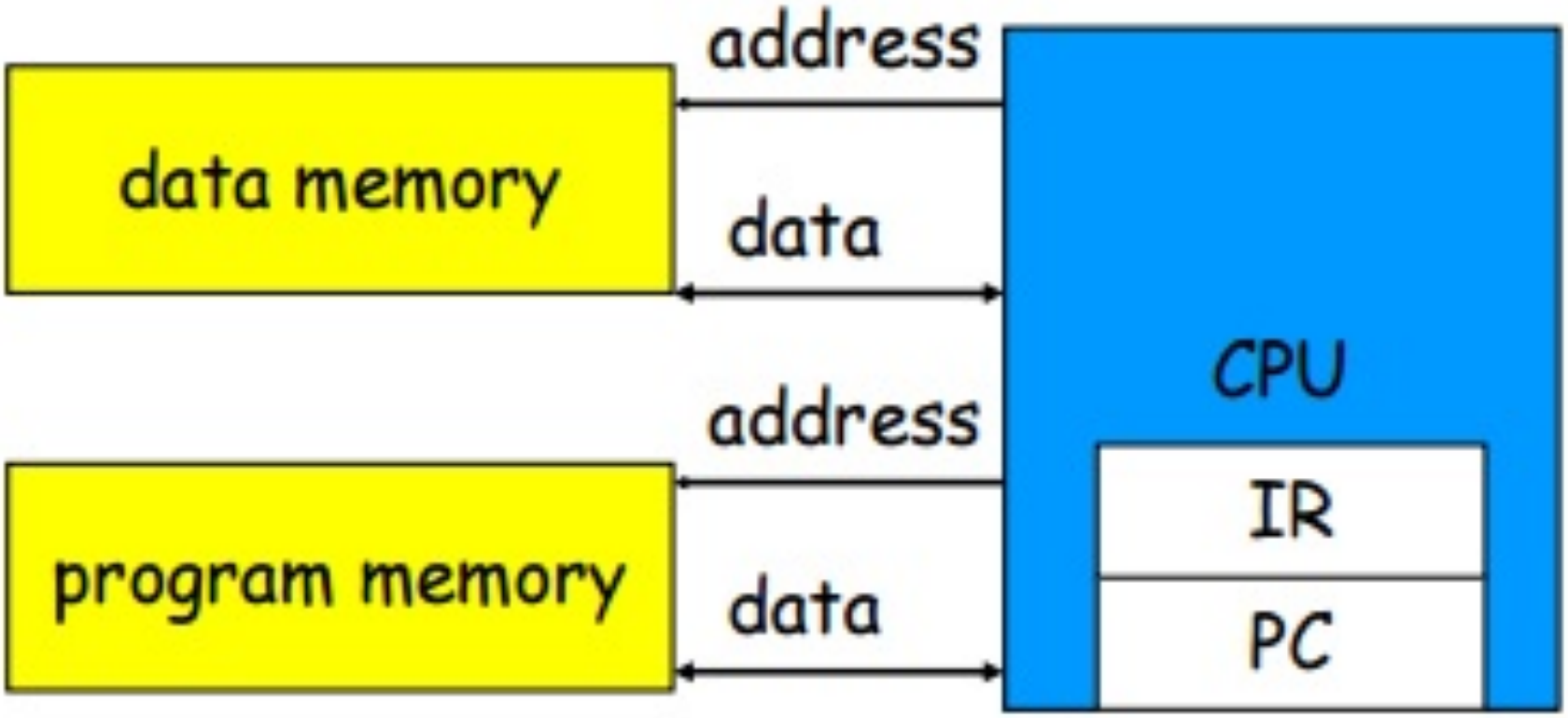


Von Neumann


address
200
memory
data
CPU
200
ADD r5,r1,r3
ADD r5,r1,r3
Von Neumann

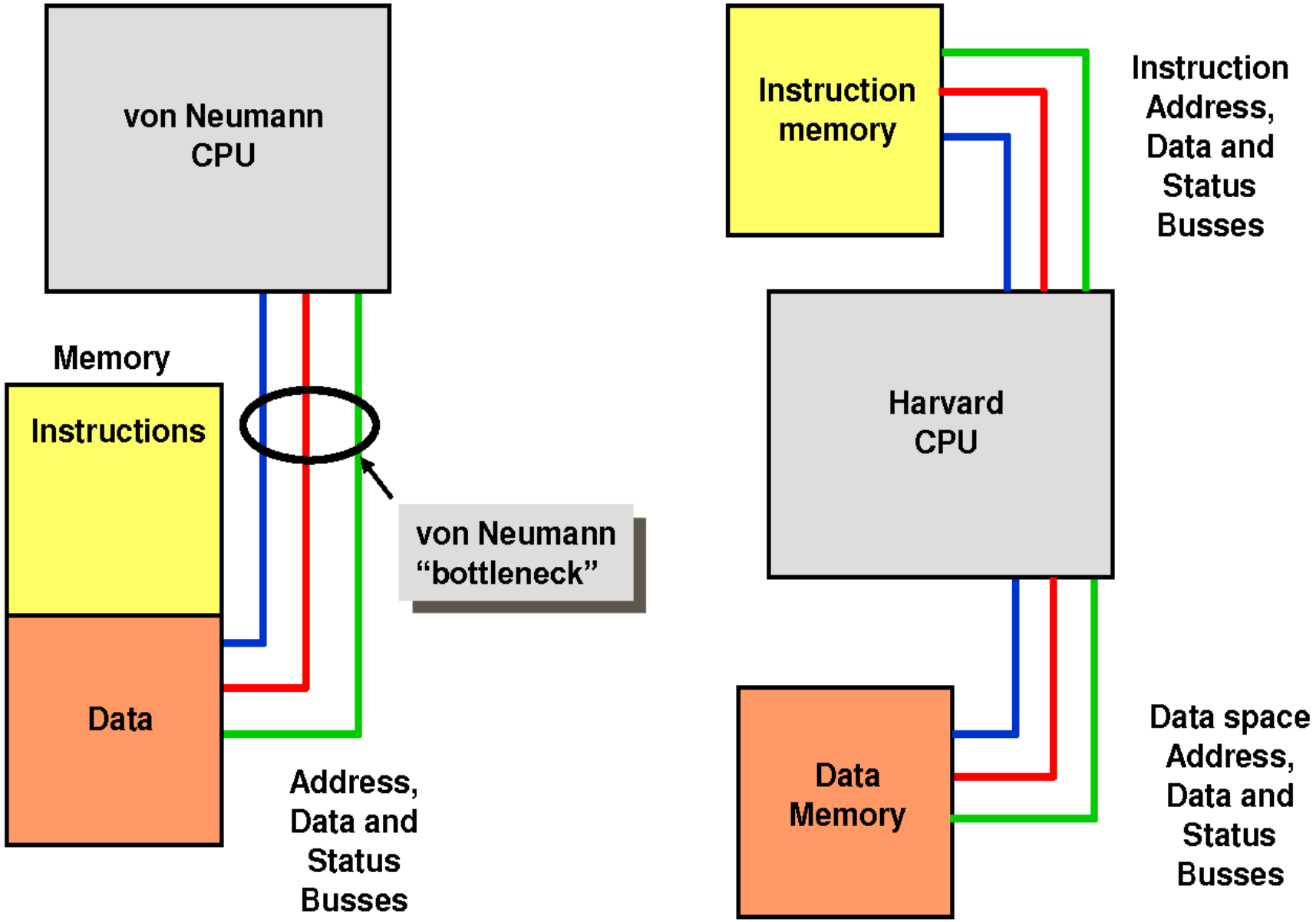
HARVARD ARCHITECTURE

- The computer has two separate memories for storing data and program.
- Processor can complete an instruction in one cycle if appropriate pipelining strategies are implemented.
- In the first stage of pipeline the instruction to be executed can be taken from program memory.
- In the second stage of pipeline data is taken from the data memory using the decoded instruction or address.
- Most of the modern computing architectures are basec on Harvard architecture.
- Two sets of address/data buses between CPU and memory

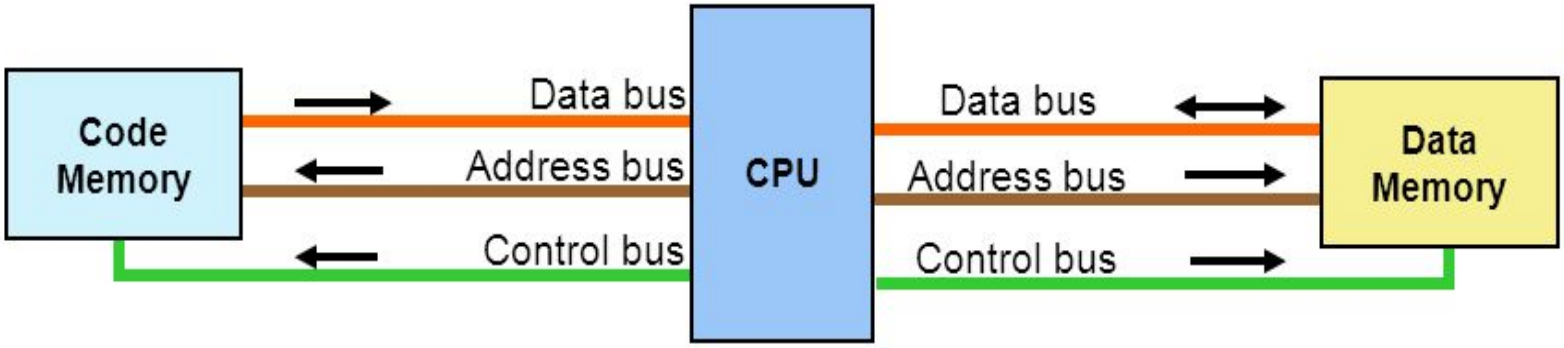


Harvard Architecture

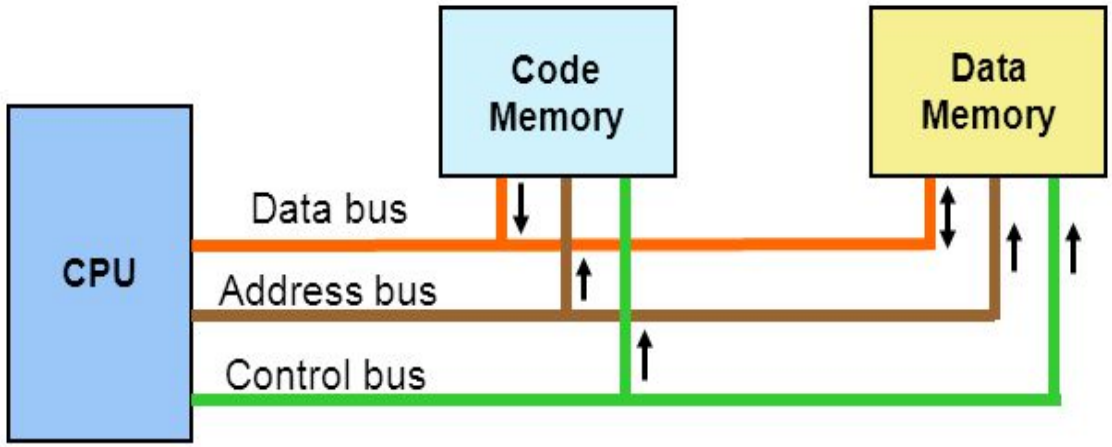
von Neumann and Harvard Architectures



Von Neumann vs. Harvard architecture



Harvard architecture



Von Neumann architecture

Von Neumann Bottleneck

This is the limitation imposed by the von Neumann architecture on computers that uses this model

MEMORY HIERARCHY

The design constraints on a computer's memory can be summed up by three questions: *How much? How fast? How expensive?*

- How much? relates to the storage capacity
- How fast? relates to performance in order to keep up with the processor.
- How expensive? relates to the cost of producing the memory.

As might be expected, there is a trade-off among the three key characteristics of memory: namely;

capacity, access time, and cost.

MEMORY HIERARCHY

- A variety of technologies are used to implement memory systems.
- Across this spectrum of technologies, the following relationships hold:
  - Faster access time, greater cost per bit
  - Greater capacity, smaller cost per bit
  - Greater capacity, slower access time

This leaves designers with a lot of dilemma.

The solution to this dilemma is not to rely on a single memory component or technology, but to employ a **memory hierarchy.**

MEMORY HIERARCHY

- Memory hierarchy is a strategic way of organizing computers' memory in order to optimize its use and to achieve greater efficiency and economy.
- Memory is organized in a hierarchy with the highest performance and in general the most expensive devices at the top, and with progressively lower performance and less costly devices in succeeding layers.
- Each level of the memory hierarchy is of higher speed and lower latency, and is of smaller size, than lower levels

MEMORY HIERARCHY

The memory hierarchy in most computers is as follows:

Processor registers

fastest possible access (usually 1 CPU cycle), only hundreds of bytes in size

Level 1 (L1) cache

often accessed in just a few cycles, usually tens of kilobytes

Level 2 (L2) cache

higher latency than L1 by 2× to 10×, often 512KB or more

✧ Level 3 (L3) cache

(optional) higher latency than L2, often multiple MB's

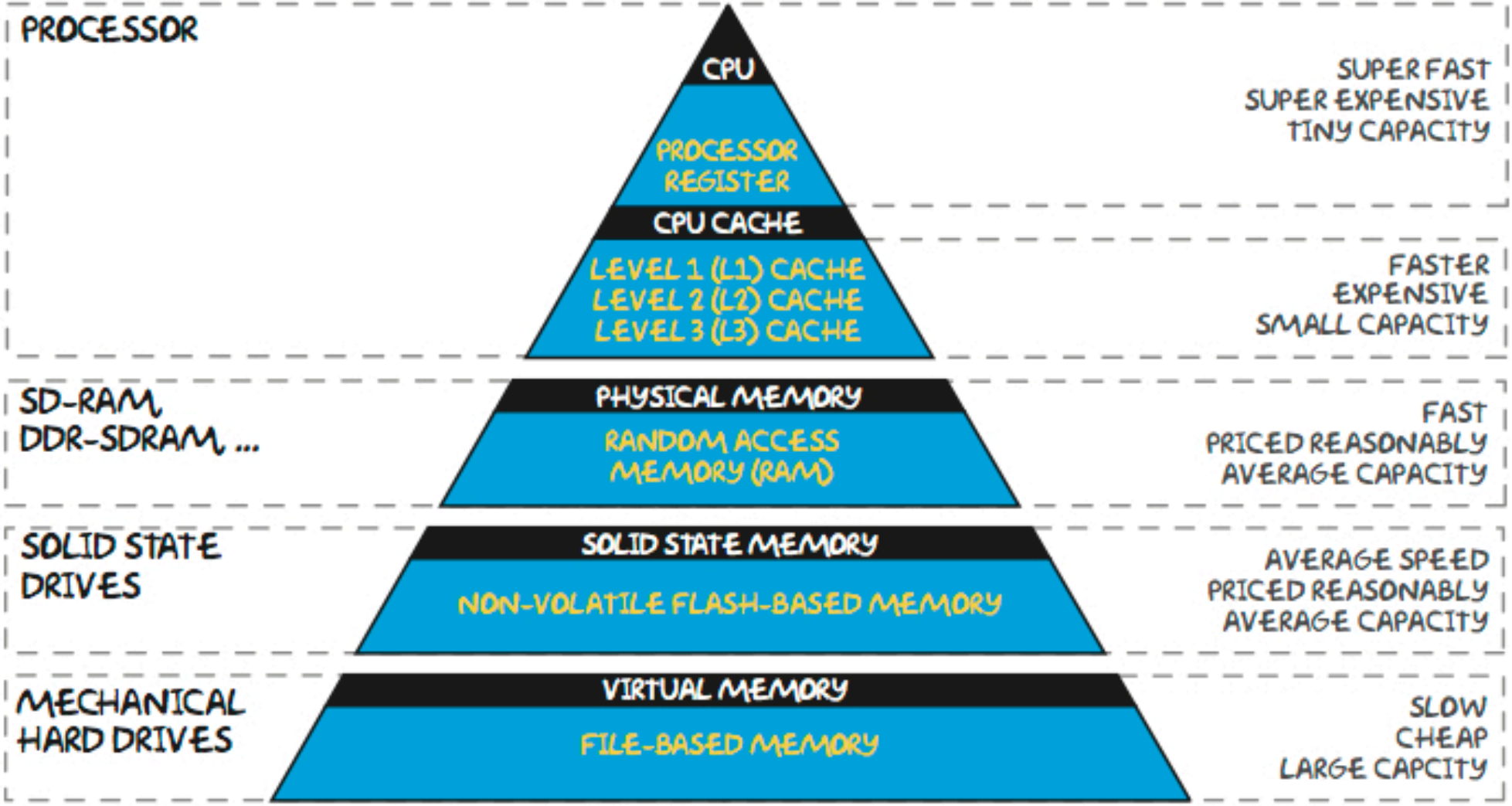
✧ Main memory (DRAM)

may take hundreds of cycles, but can be multiple gigabytes

✧ Disk storage

hundreds of thousands of cycles latency, but very large

THE MEMORY HIERARCHY



MEMORY HIERARCHY

As one goes down the hierarchy, the following occur:

- ✓ Decreasing cost per bit
- ✓ Increasing capacity
- ✓ Increasing access time
- ✓ Decreasing frequency of access of the memory by the processor

During the design of computer system, smaller, more expensive, faster memories are supplemented by larger, cheaper, slower memories, hence balancing the associated tradeoffs