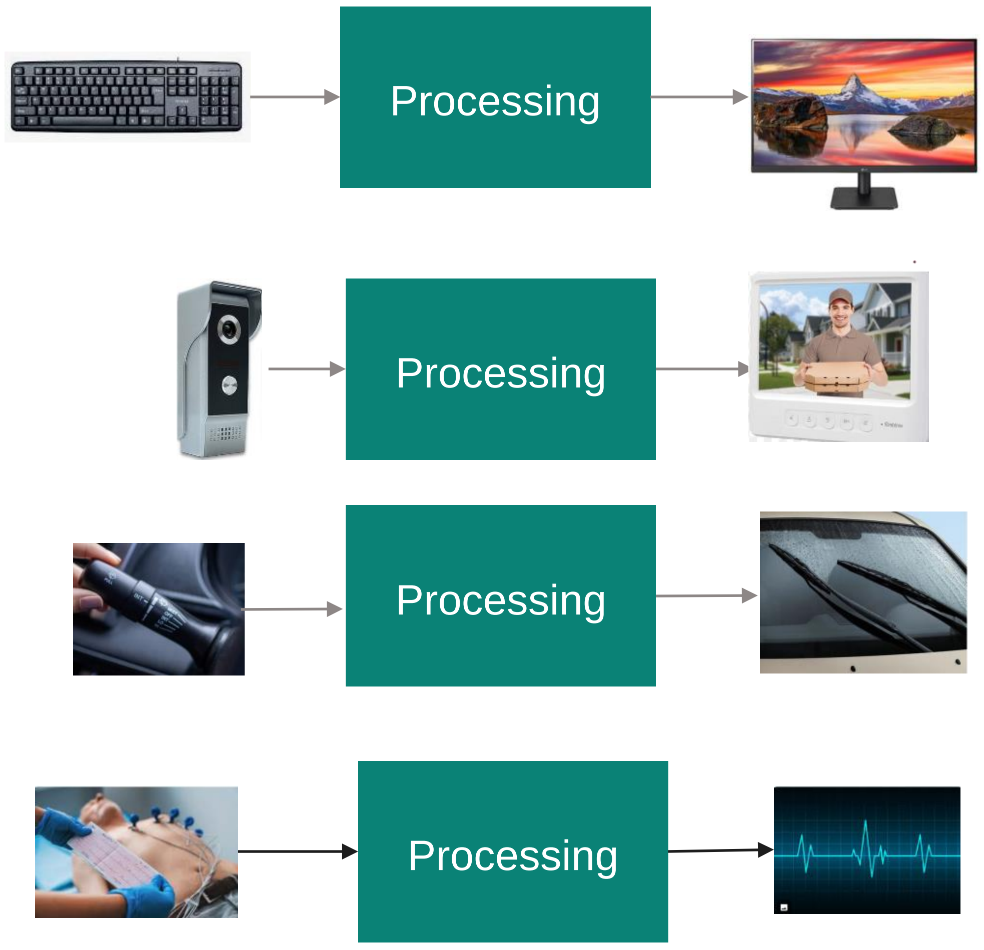
Computer architecture

Prakash Balasubramanian

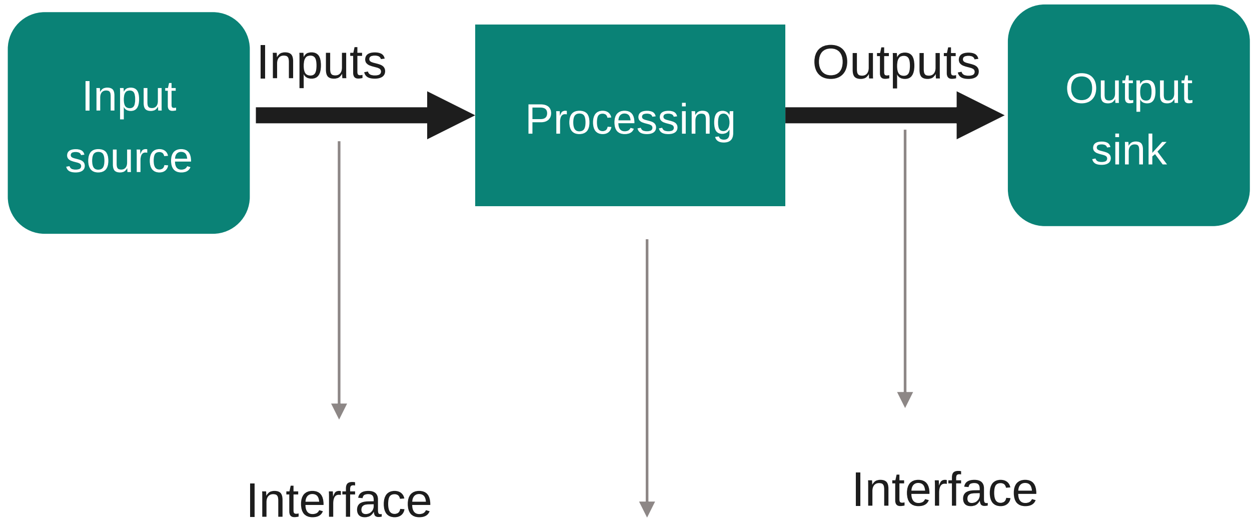
06/April/2024

Systems – 1/3

Examples



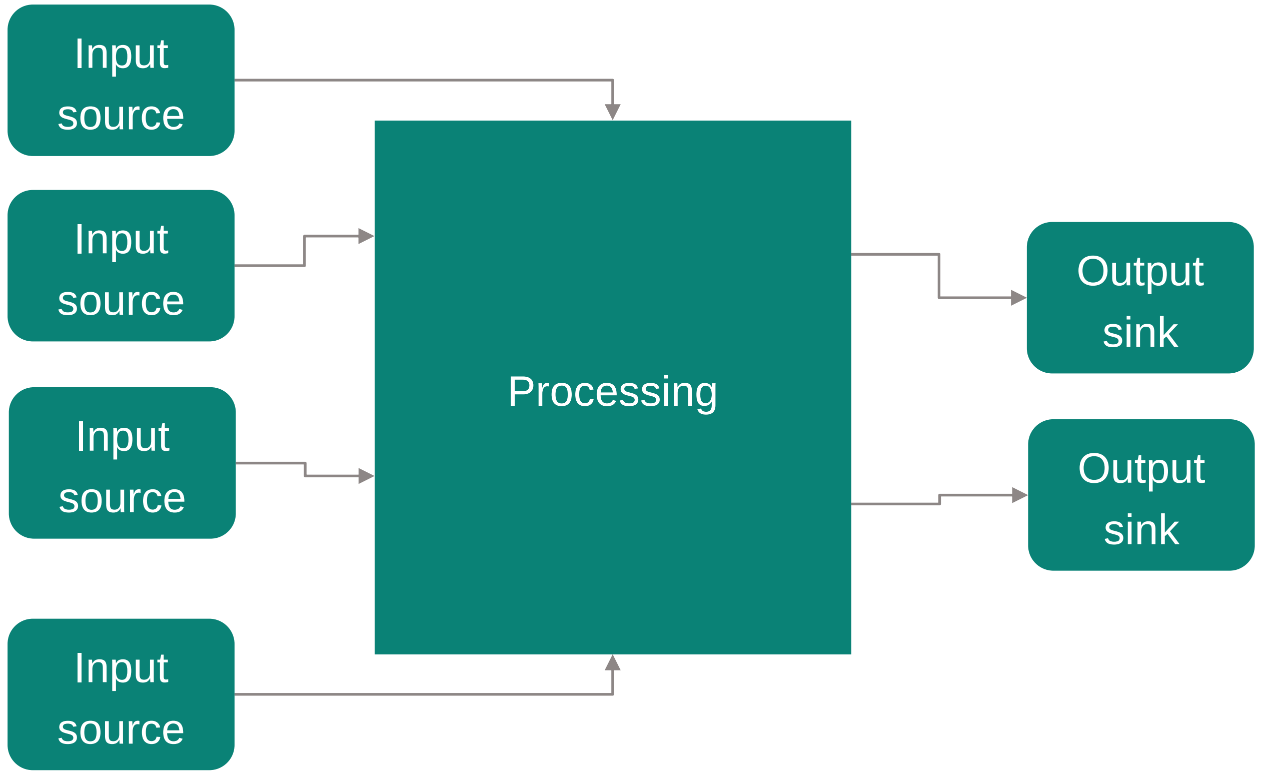
Generalization

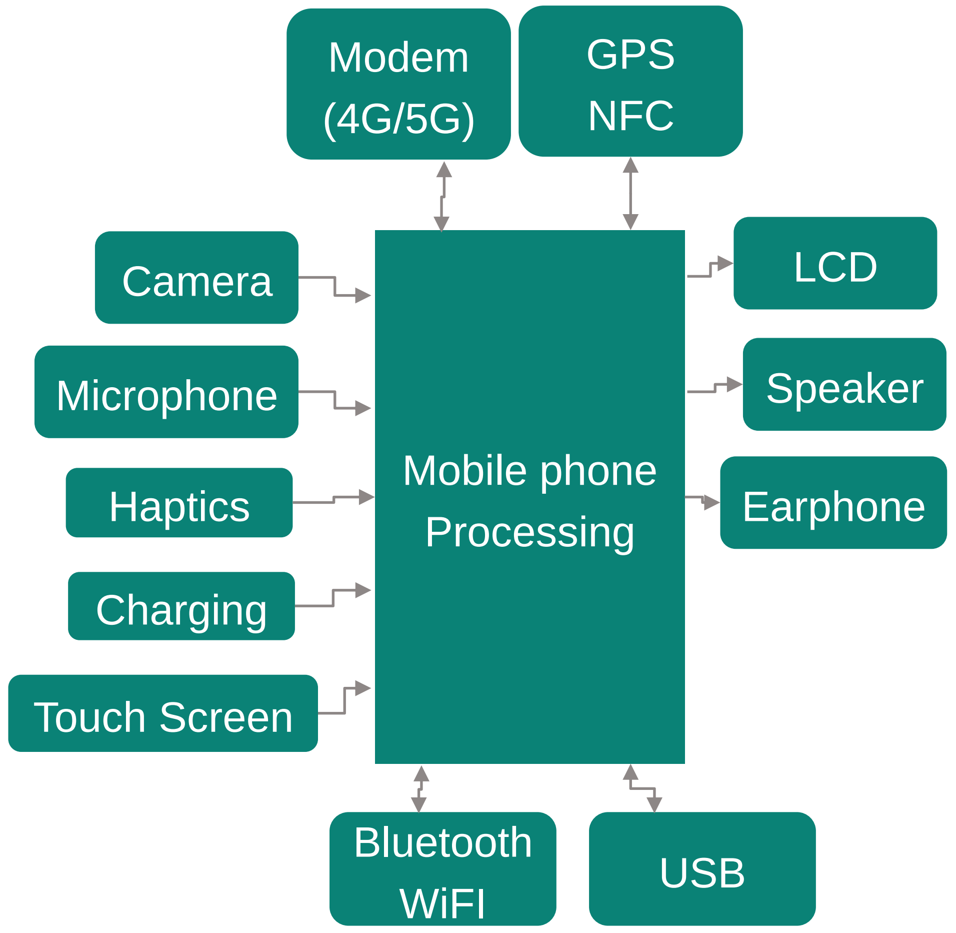


Input acquisition
Processing
Output generation

Systems – 2/3

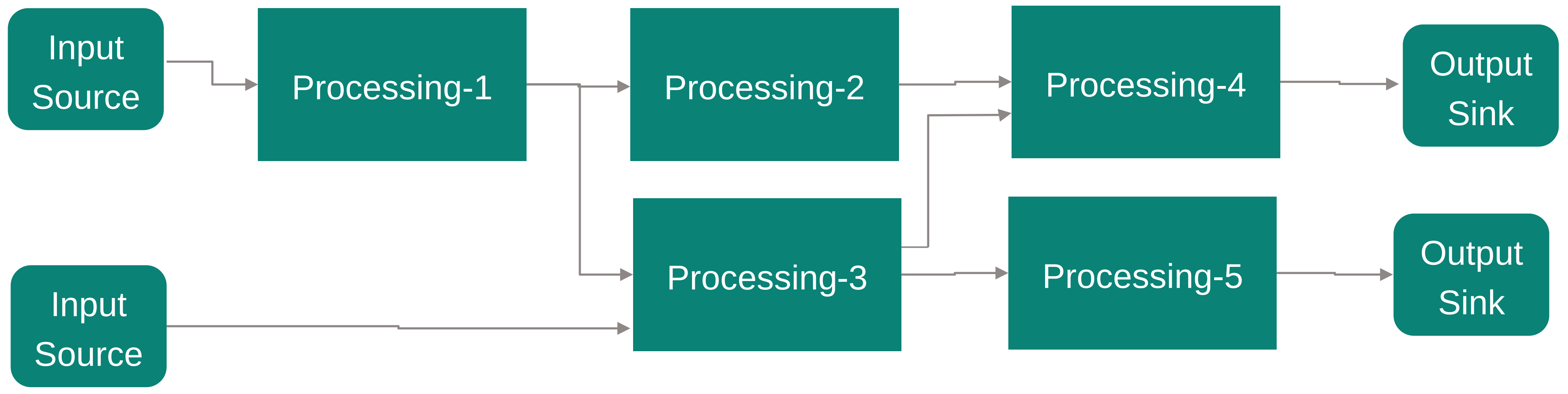
Multiple Input Multiple Output



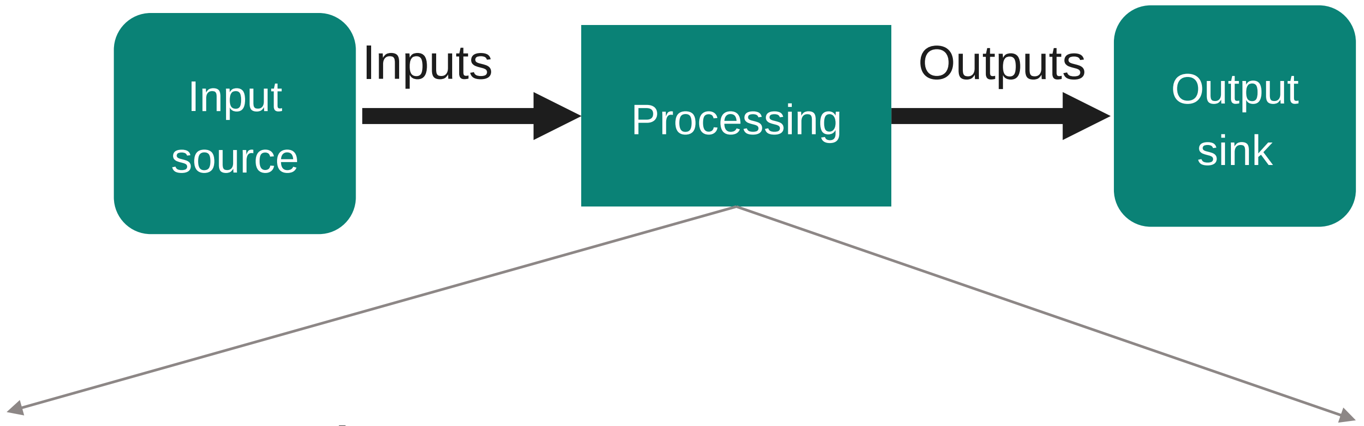


Systems – 3/3

Hierarchy of systems, cascaded systems



Real time system – 1/2



General purpose processing

- Working on an excel sheet on a Windows/Linux PC
- Watching a YouTube video

Objectives of the system are met even if there are lapses and delays in processing.
These are Non-Real time systems

Real Time processing

- Acquiring and processing an image frame before the next frame arrives
- Reading out a sensor value when a new input signal becomes available

Time is the essence
Processing must be performed at intended time
Processing must be completed within specified duration
MOST EMBEDDED SYSTEMS ARE REAL TIME SYSTEMS!

Real time system 2/2

Real time systems

Hard Real Time

Deadlines cannot be missed.
Deadlines are Non-Negotiable!

- The pressing of the brake pedal must be recognized
- Once recognized, braking action must be applied within 20ms.
- If not, there is a good probability that an accident may happen and perhaps even result in fatality!

Soft Real time

An occasional miss in deadline is acceptable

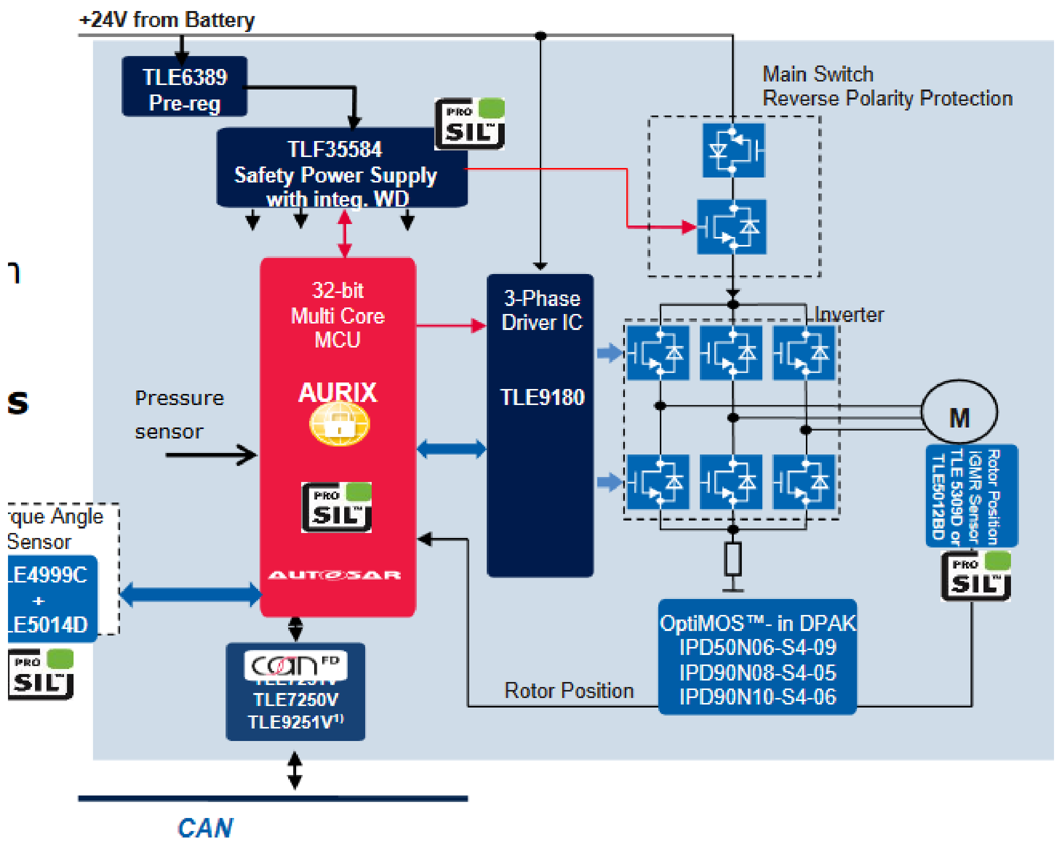
- A camera captures images at a rate of 30 frames per second
- Time gap between two frames is 1/30 = 33 ms
- Processing of a frame (e.g. Demosaicing) should therefore be done within 33 ms of its attival
- If every 472nd frame takes a little longer than 33ms for processing, there is barely any impact on end user

General purpose vs Embedded systems

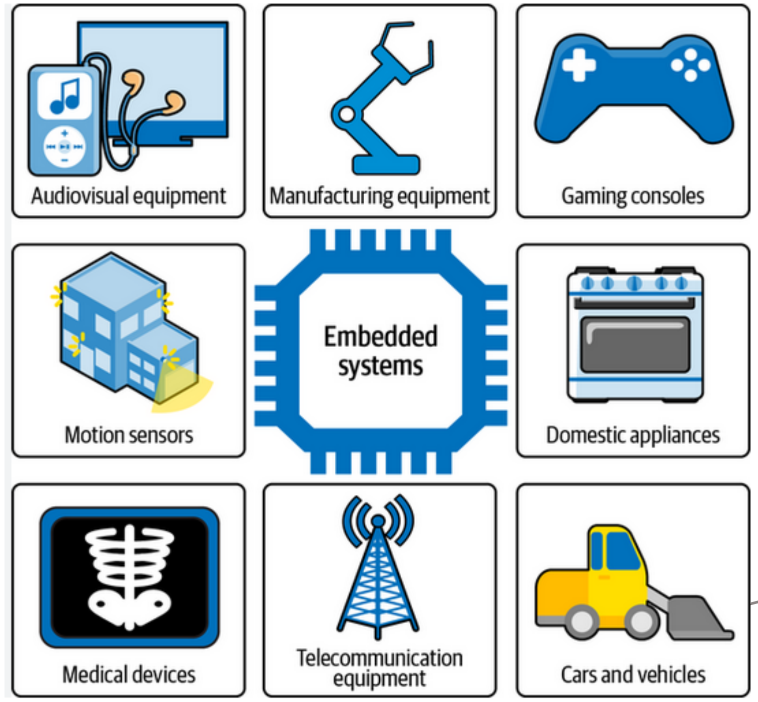
General purpose applications

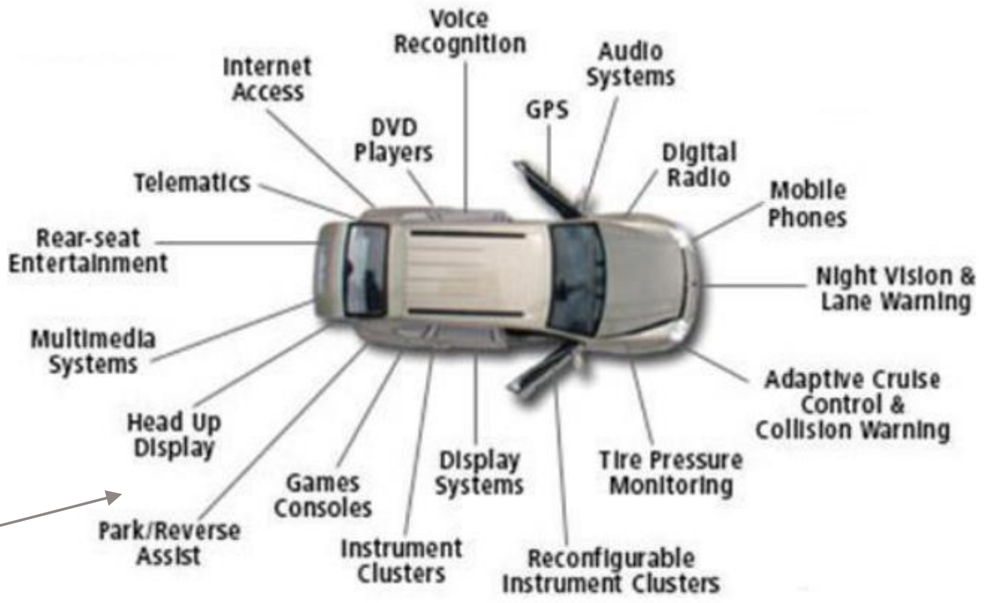
- Games, Video
- Browsing
- E-Mail
- Desktop publishing
- Star gazing app
- Stock market app

Specific purpose system based on a Microprocessor/Microcontroller/SoC

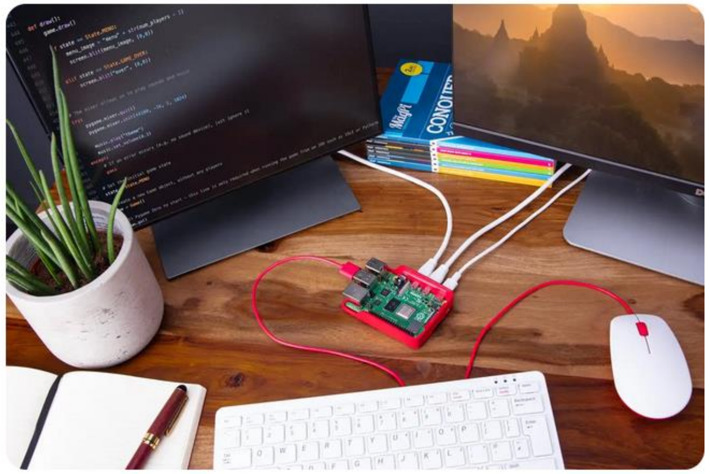

+24V from Battery
TLE6389 Pre-reg
TLF35584 Safety Power Supply with integ. WD
Main Switch Reverse Polarity Protection
32-bit Multi Core MCU
AURIX
AUTOSAR
3-Phase Driver IC
TLE9180
Inverter
M
Pressure sensor
que Angle Sensor
E4999C + E5014D
Rotor Position iGMR Sensor TLE 5309D or TLE5012BD
OptiMOS™- in DPAK IPD50N06-S4-09 IPD90N08-S4-05 IPD90N10-S4-06
Rotor Position
TLE7250V TLE9251V1)
CAN

Pervasiveness of embedded systems

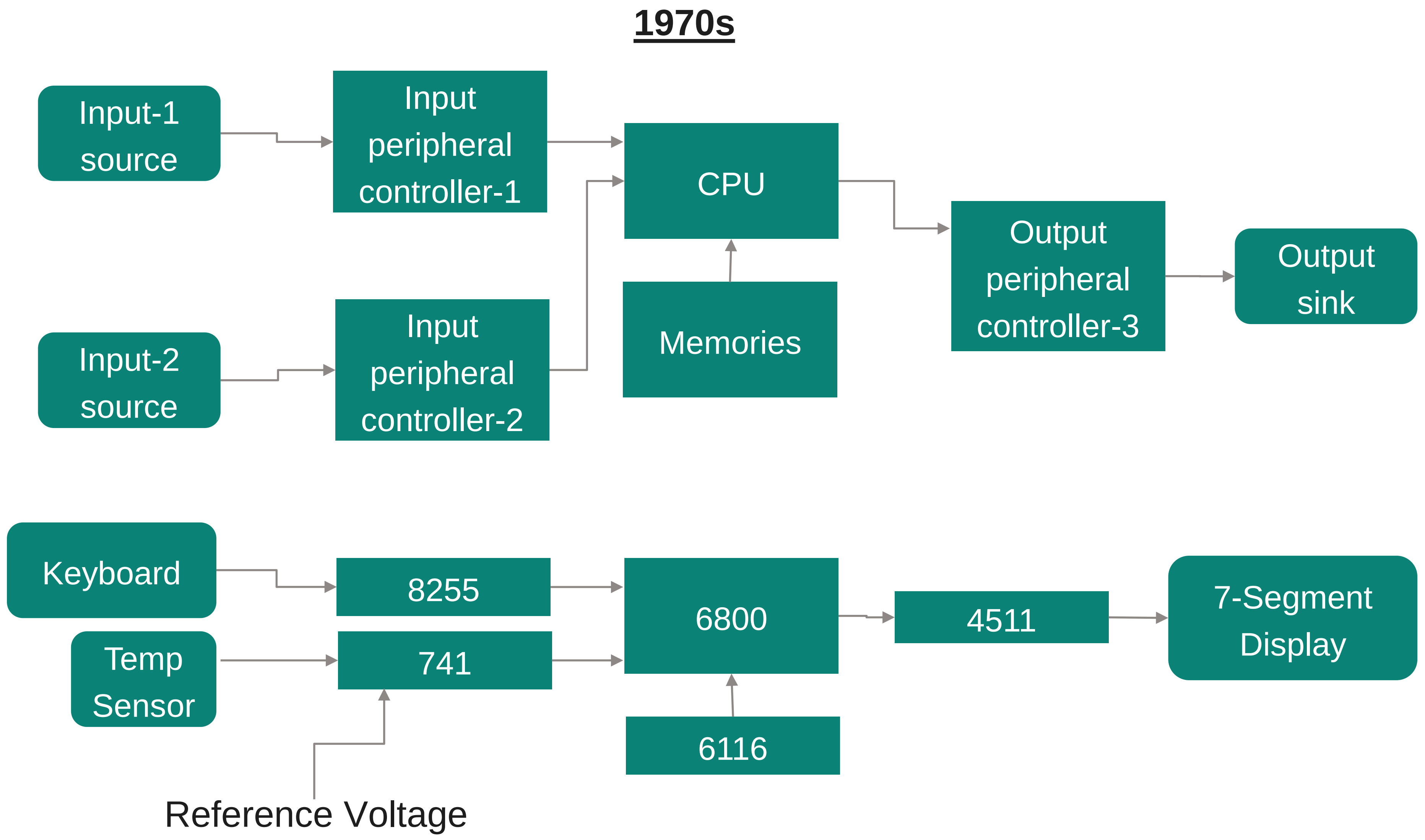




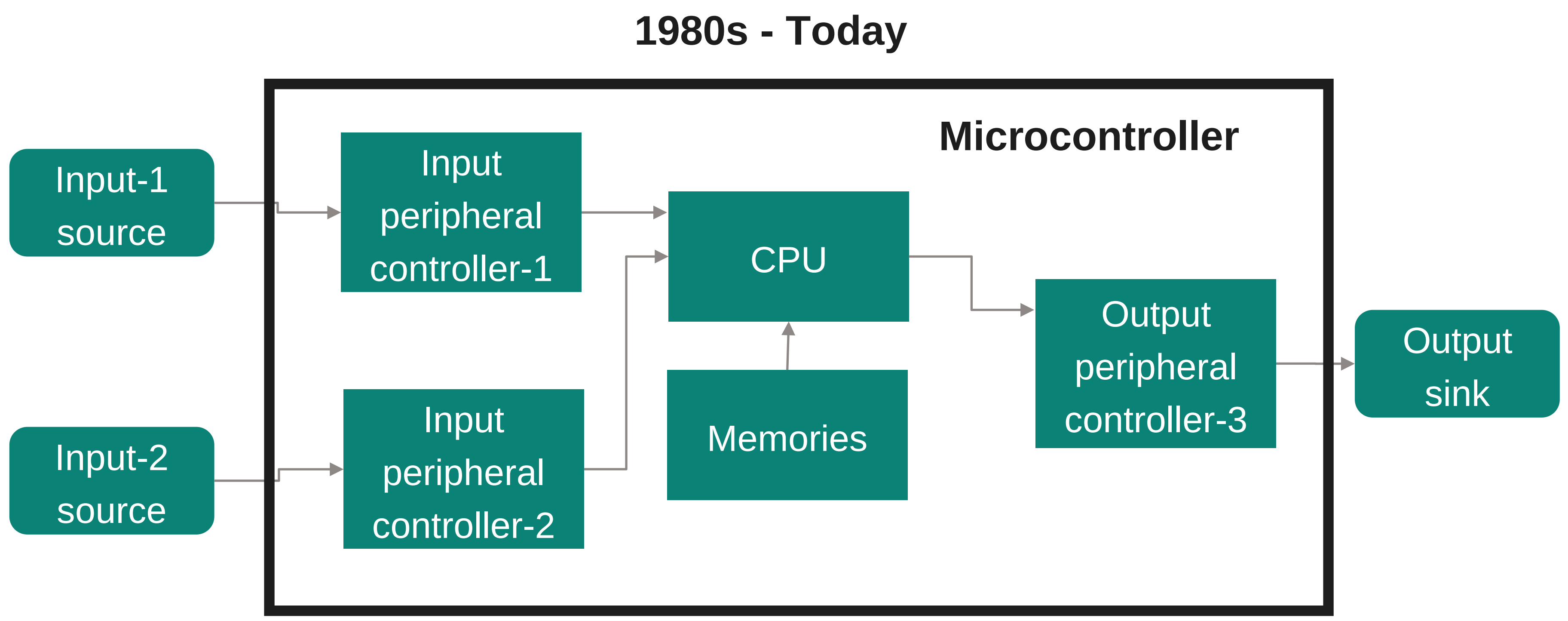
QUIZ: How would you classify this?



Embedded systems – 1/3

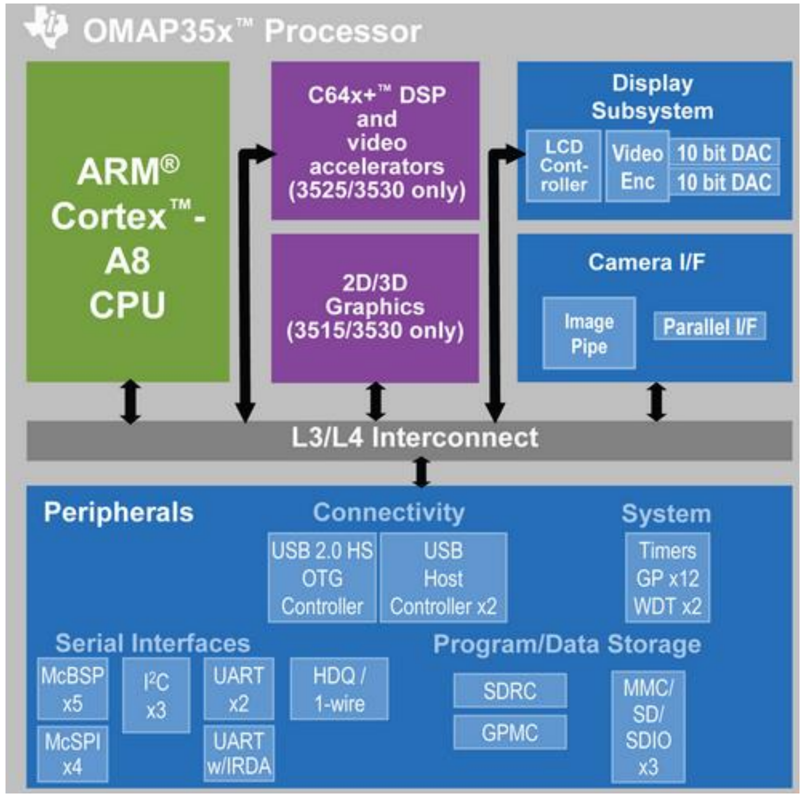


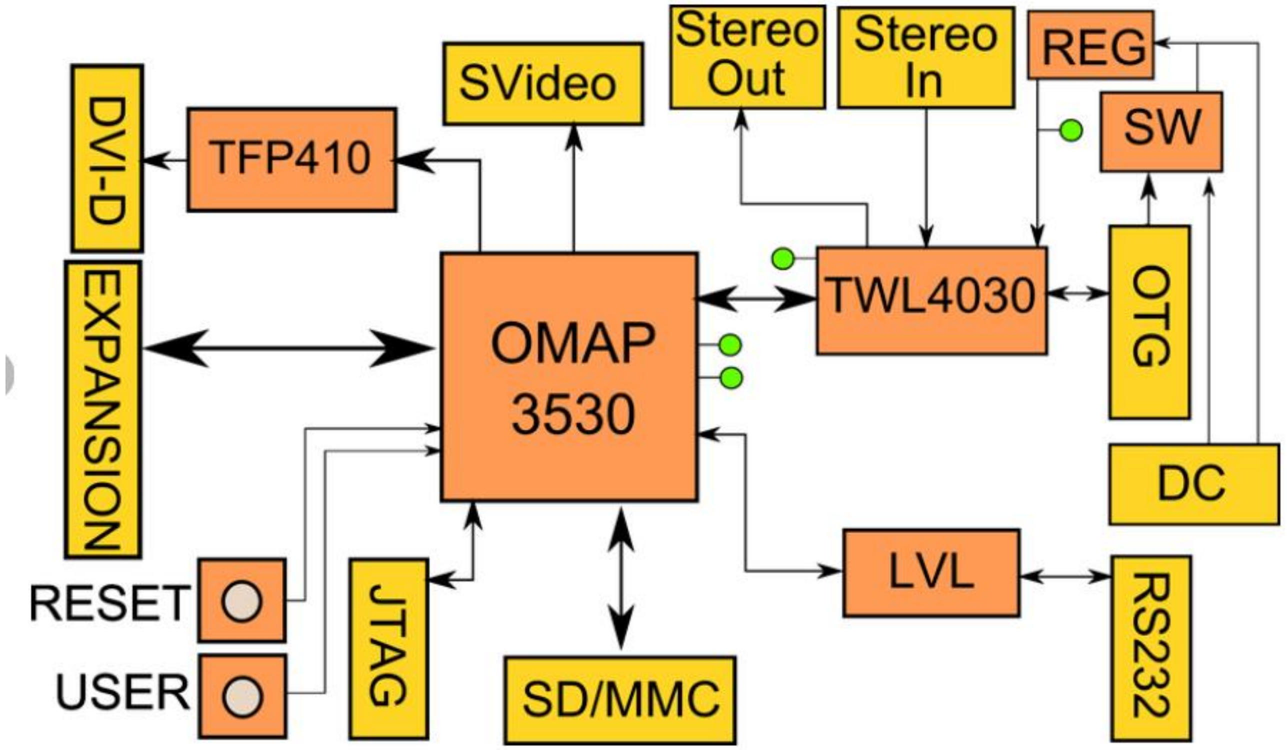
Embedded systems – 2/3


1980s - Today
Microcontroller
Input-1 source
Input-2 source
Input peripheral controller-1
Input peripheral controller-2
CPU
Memories
Output peripheral controller-3
Output sink

Embedded Systems – 3/3

Early 2000s until today

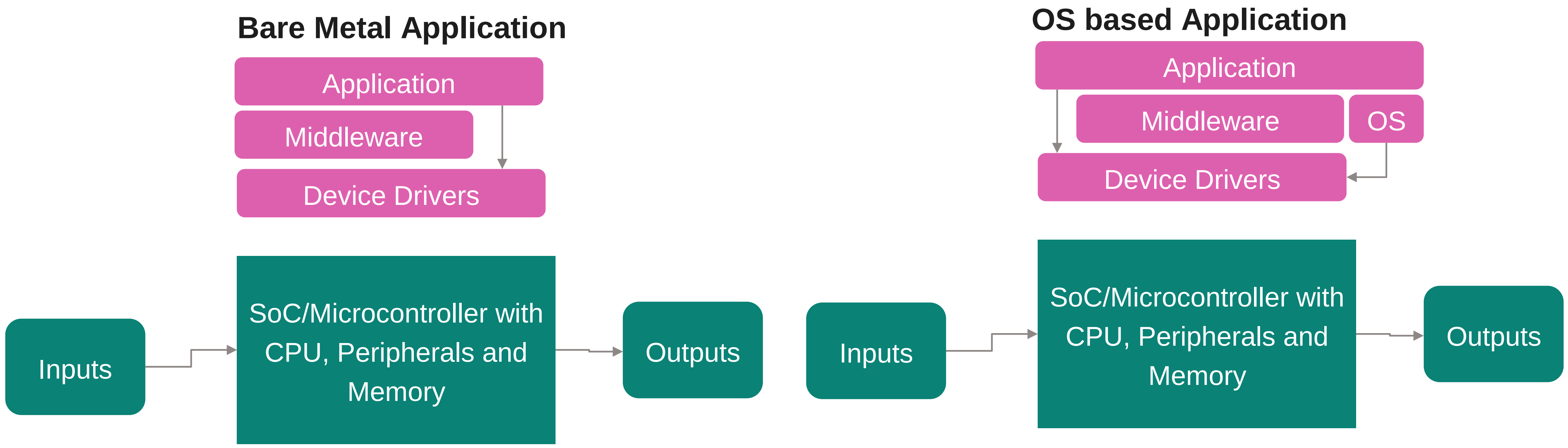




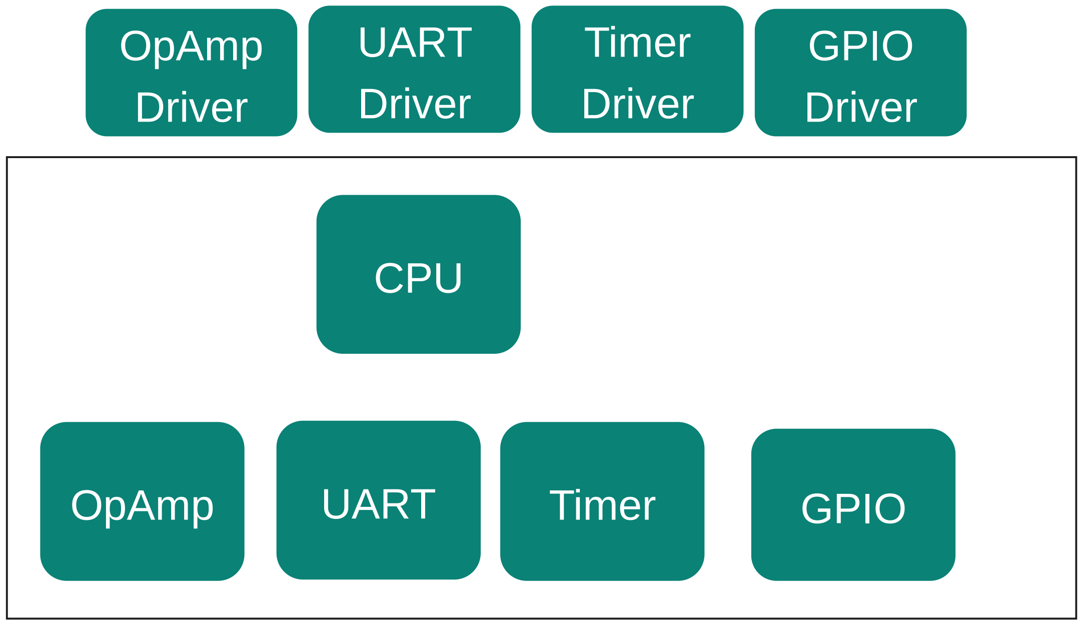
Massive integration

Single Board Computers

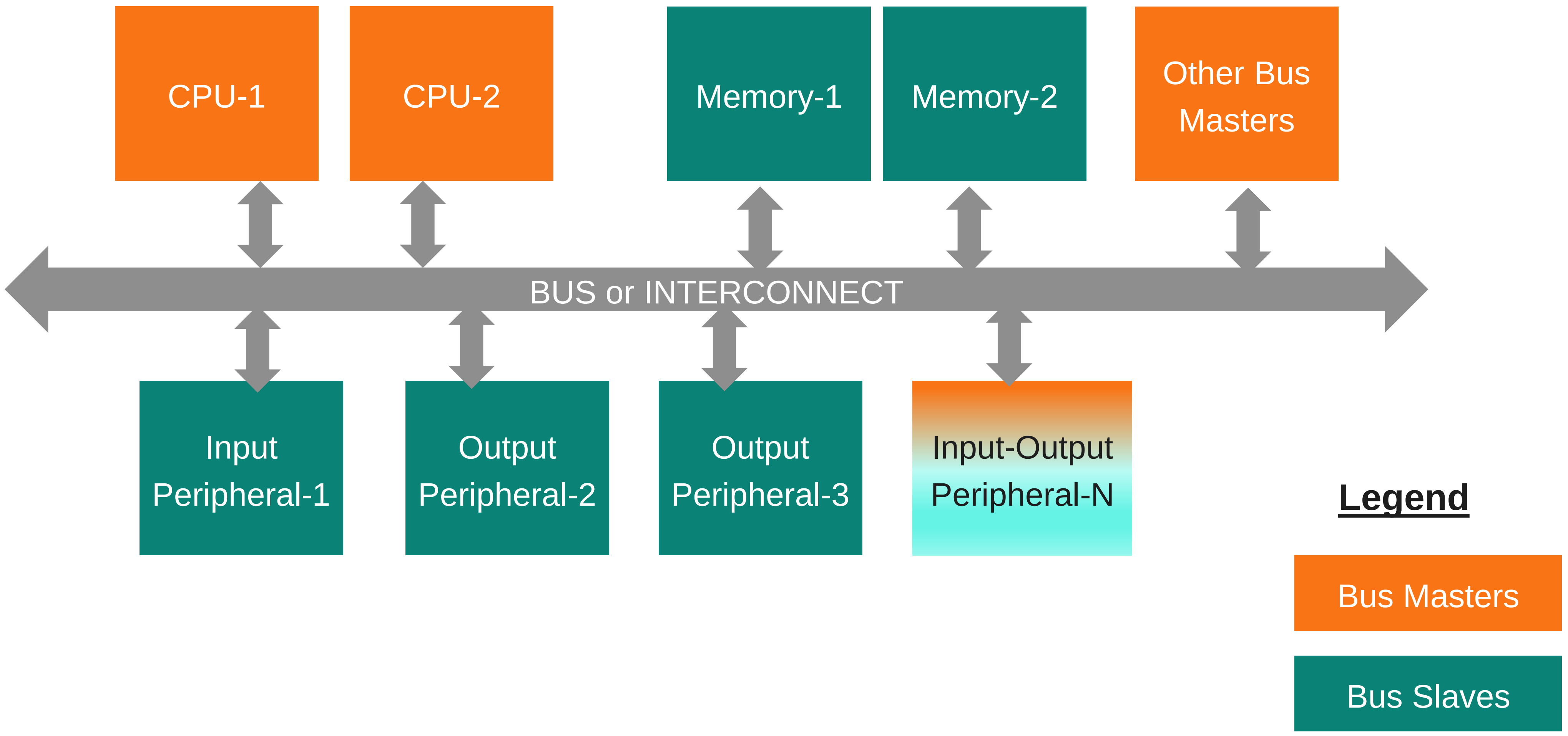
Typical software stack



Example of Peripherals and their device drivers

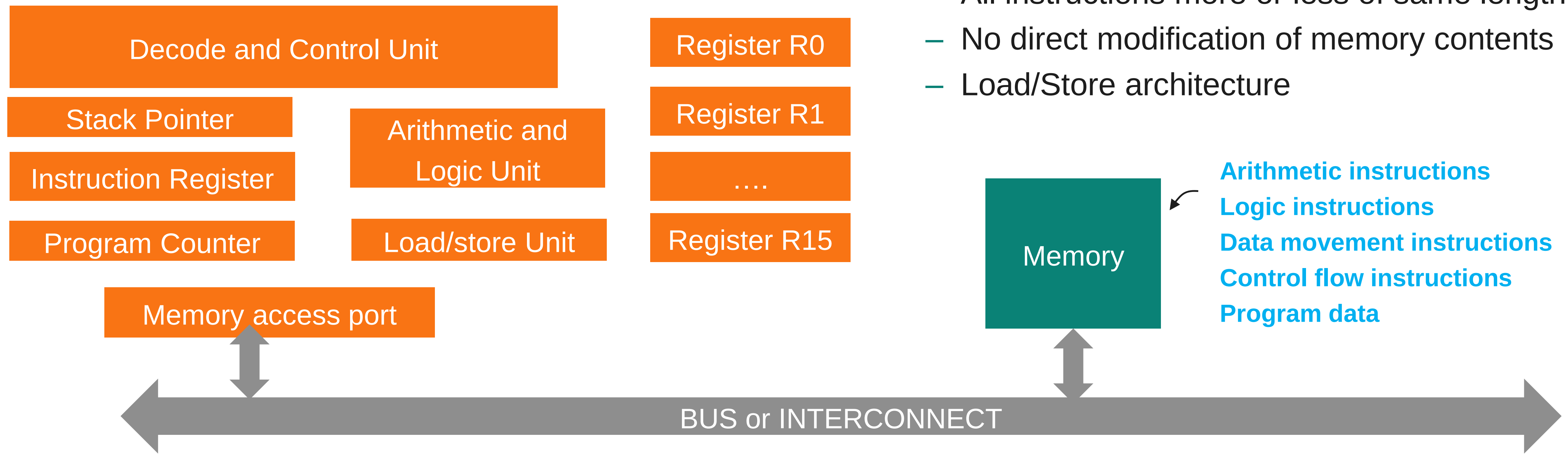


A typical Microcontroller



Von-Neumann CPU – 1/4

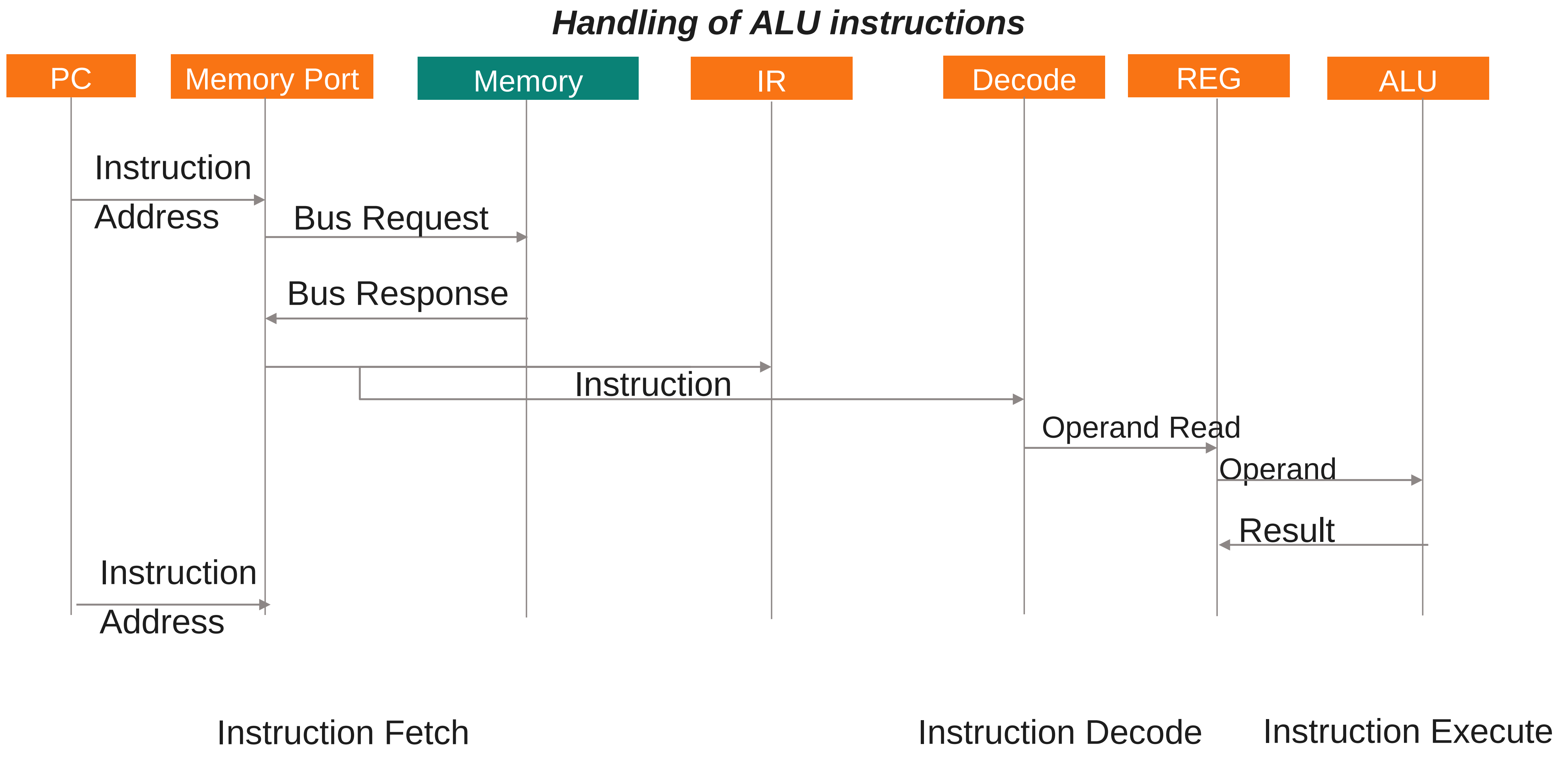
All CPUs used during the last 70+ years are Von-Neumann architectures!
All CPUs of today are improvisations of the basic RISC architecture



- All instructions more or less of same length
- No direct modification of memory contents
- Load/Store architecture

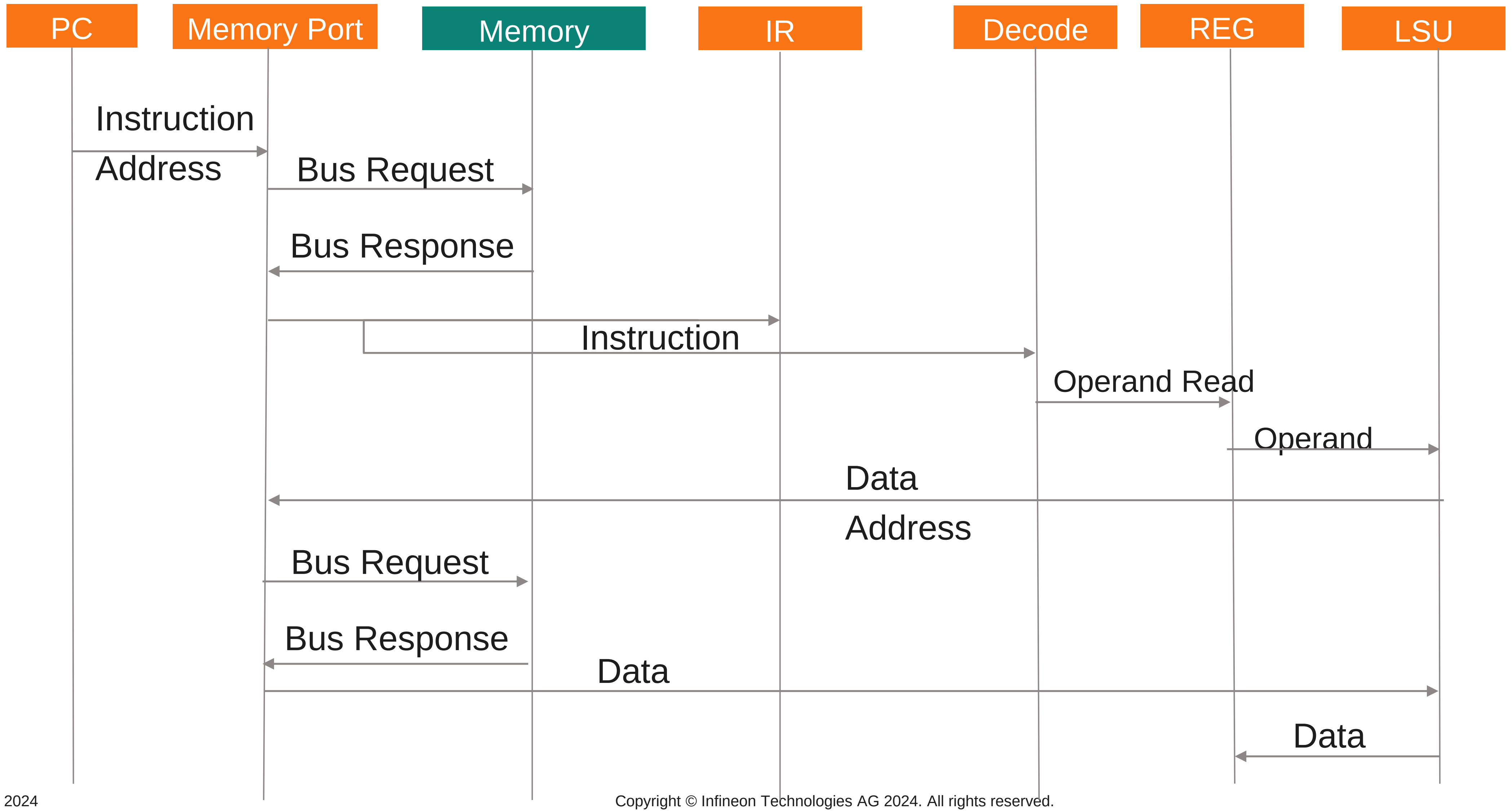
Arithmetic instructions
Logic instructions
Data movement instructions
Control flow instructions
Program data

Von-Neumann CPU – 2/4

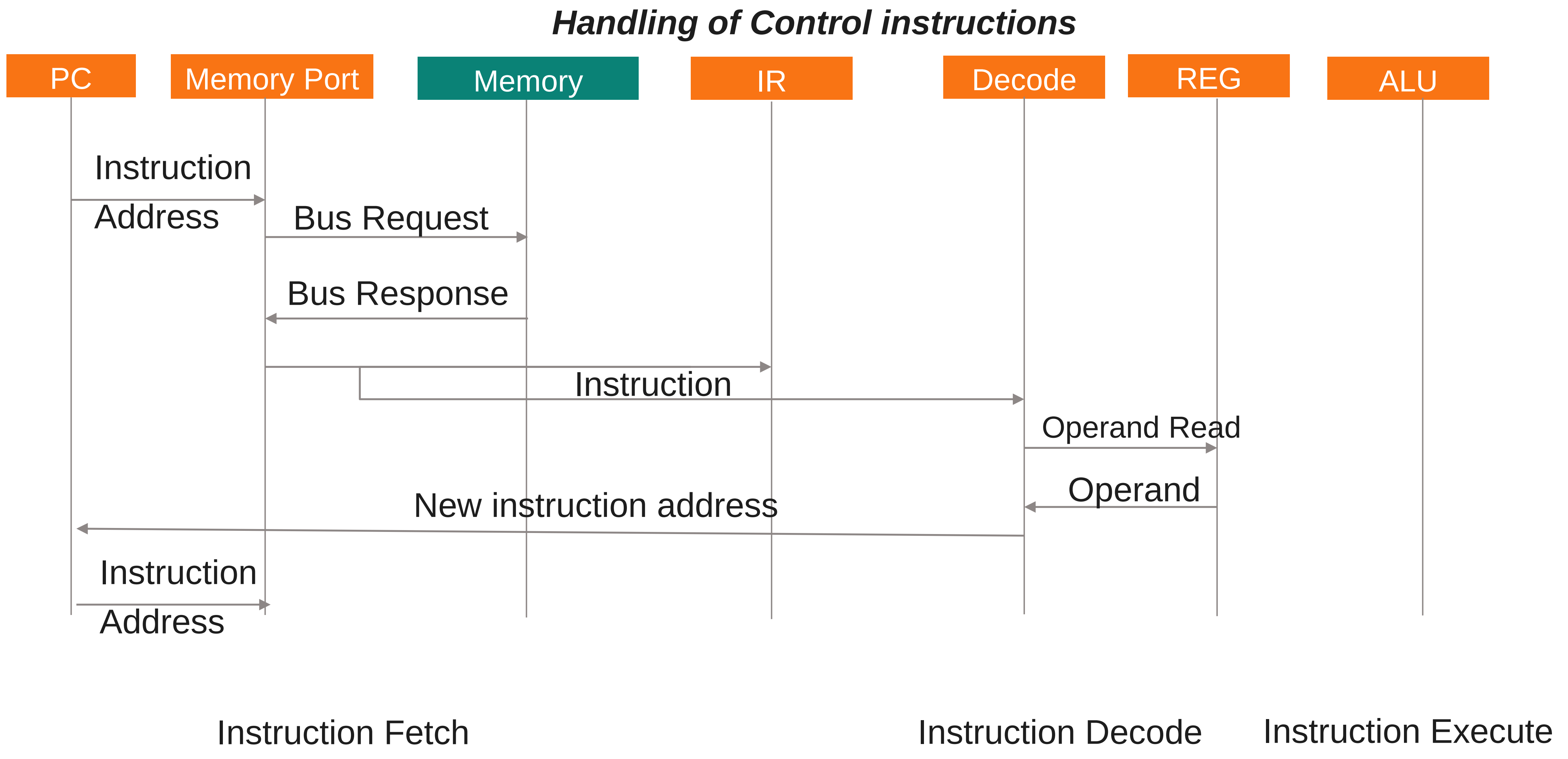


Von-Neumann CPU – 3/4

Handling of Load/Store instructions

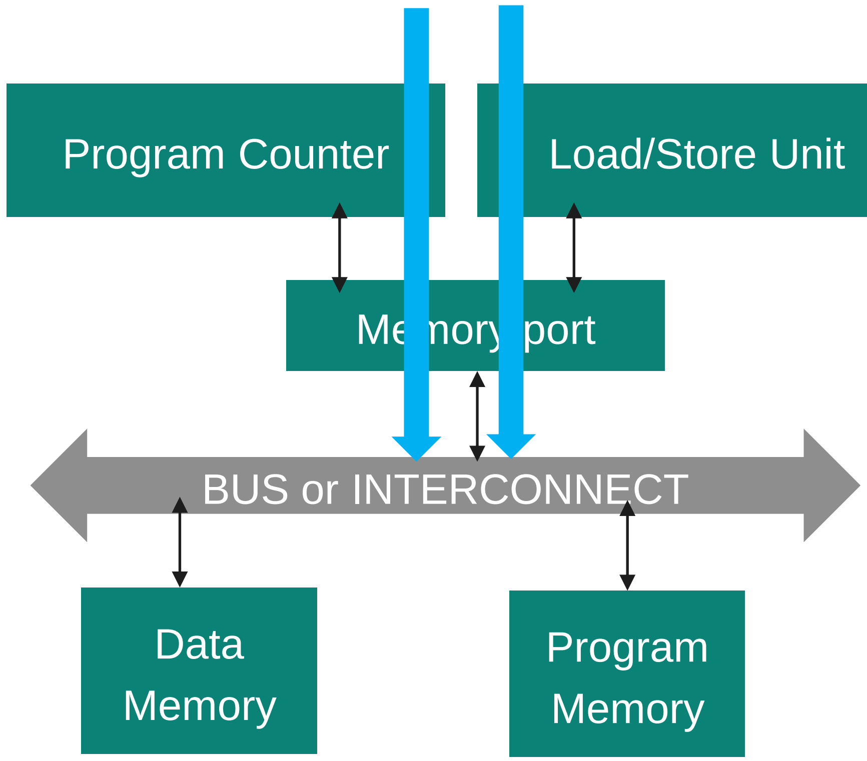


Von-Neumann CPU – 4/4



Von-Neumann vs Harvard architecture – 1/2

Harvard architecture IS essentially an improvisation of Von-Neumann architecture.
Harvard architecture removes a Von-Neumann bottleneck. That’s it!



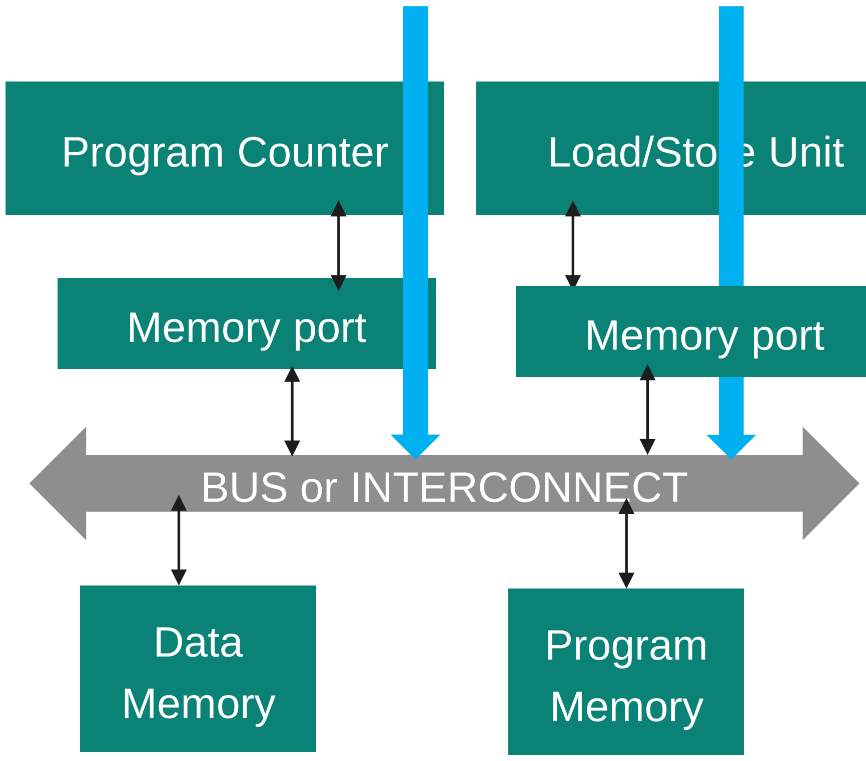
The same memory port is used for both fetching instructions during the Instruction Fetch Cycle and Data fetch/data store during the Execute cycle of a Load/Store instruction.

Only one can happen at a time and the other will have to wait!

This is the Von-Neumann bottleneck (among many others!)

Von-Neumann vs Harvard architecture – 2/2

Harvard architecture IS essentially an improvisation of Von-Neumann architecture.
Harvard architecture removes a Von-Neumann bottleneck. That’s it!



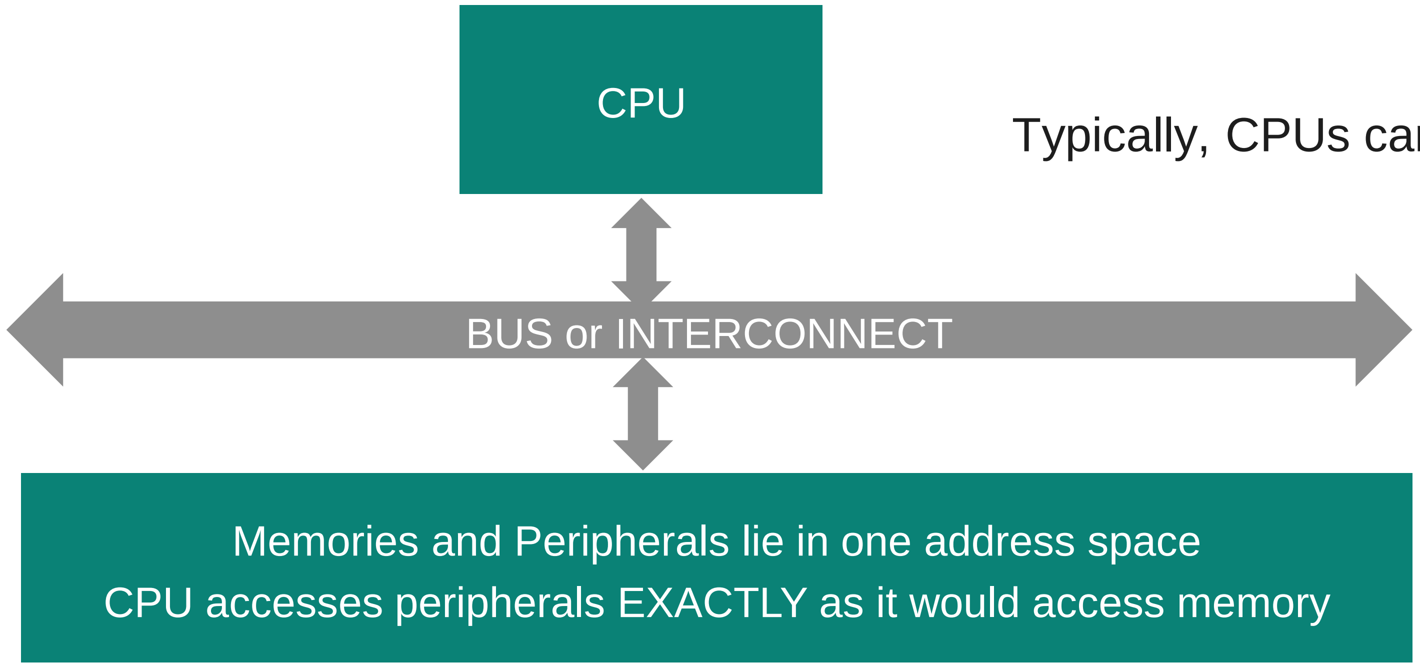
By providing two memory ports, both instruction fetch and data load/store can happen concurrently!

With this, we have eliminated the bottleneck!

Question:
Is the bottleneck truly eliminated? What is the problem with the picture?

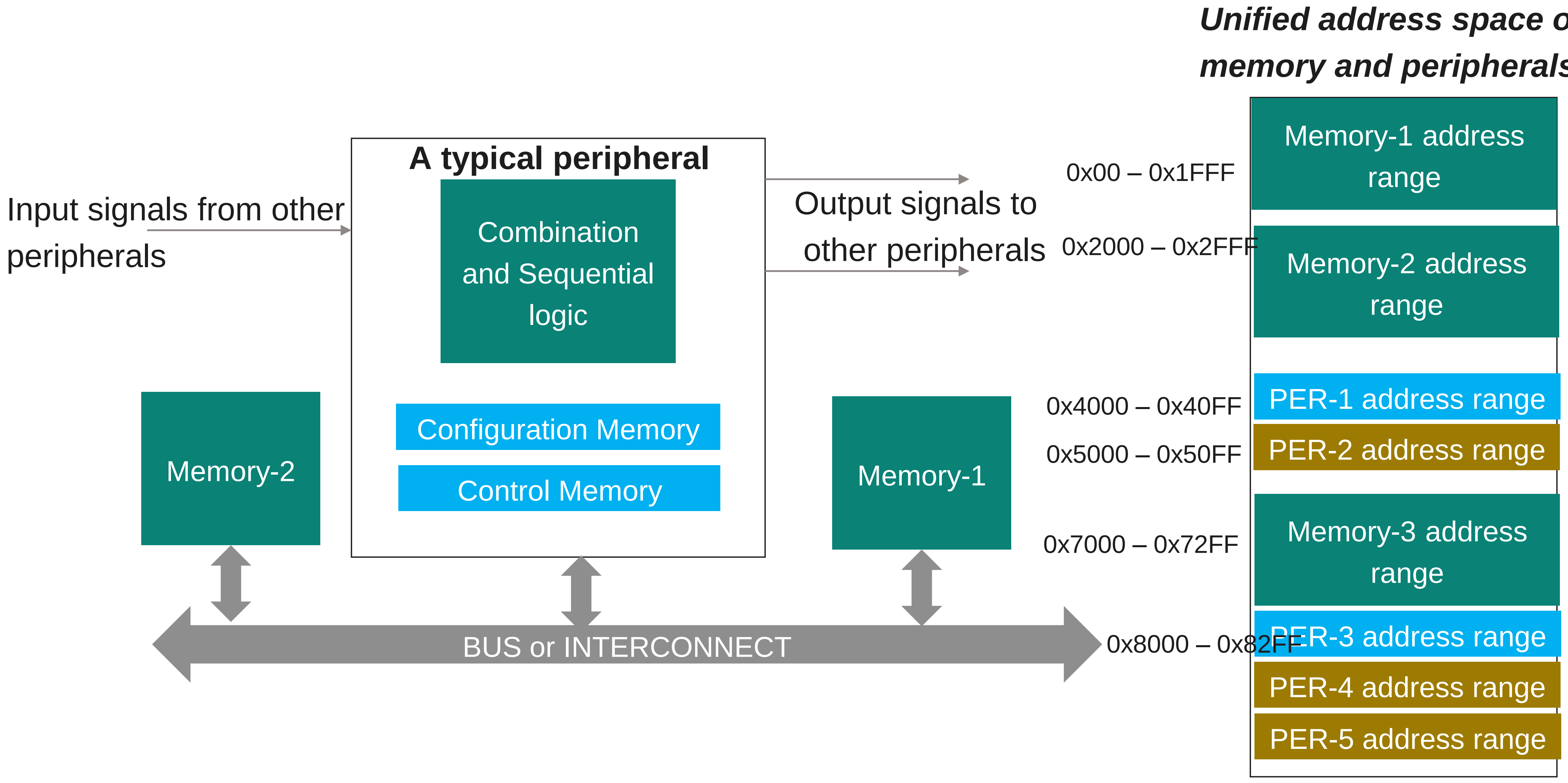
Address space of user program and peripherals – 1/2

REMEMBER THIS => **All modern CPUs follow Memory Mapped IO addressing scheme**

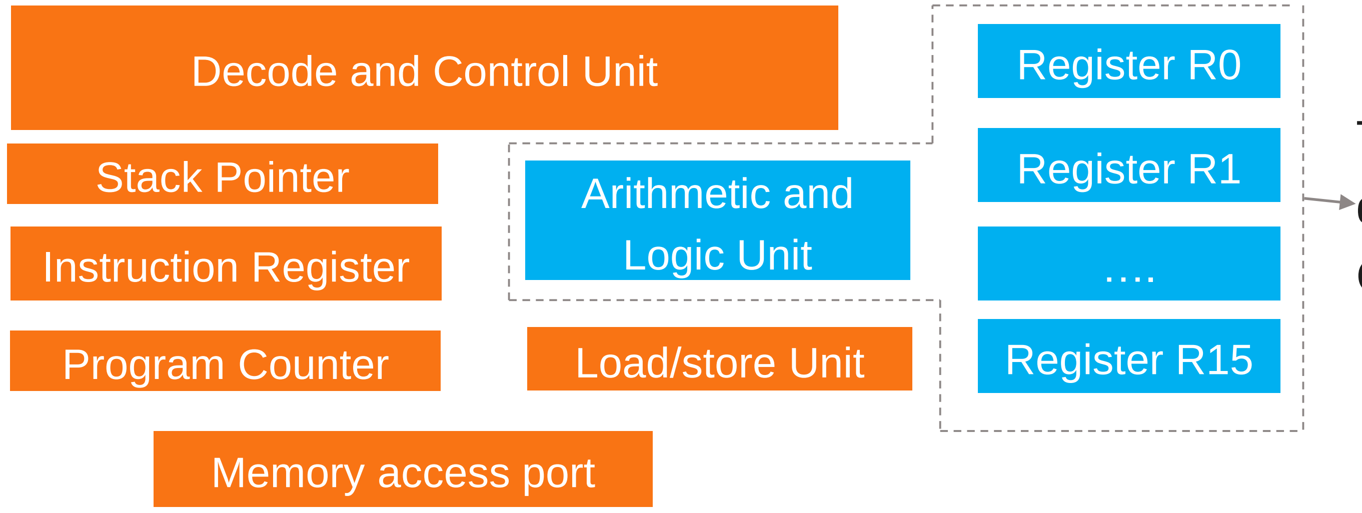

CPU
BUS or INTERCONNECT
Memories and Peripherals lie in one address space
CPU accesses peripherals EXACTLY as it would access memory

Typically, CPUs can access 2^{32} = 4GB address locations

Address space of user program and peripherals – 2/2



What type is my CPU?



The width of the register and the size of the data which ALU can handle determines the CPU type

If the register width is 32bit, then ALU is typically designed to operate on 32b operand at a time
If the register width is 16bit, then ALU is typically designed to operate on 16b operand at a time

Memories – 1/2

A toy application

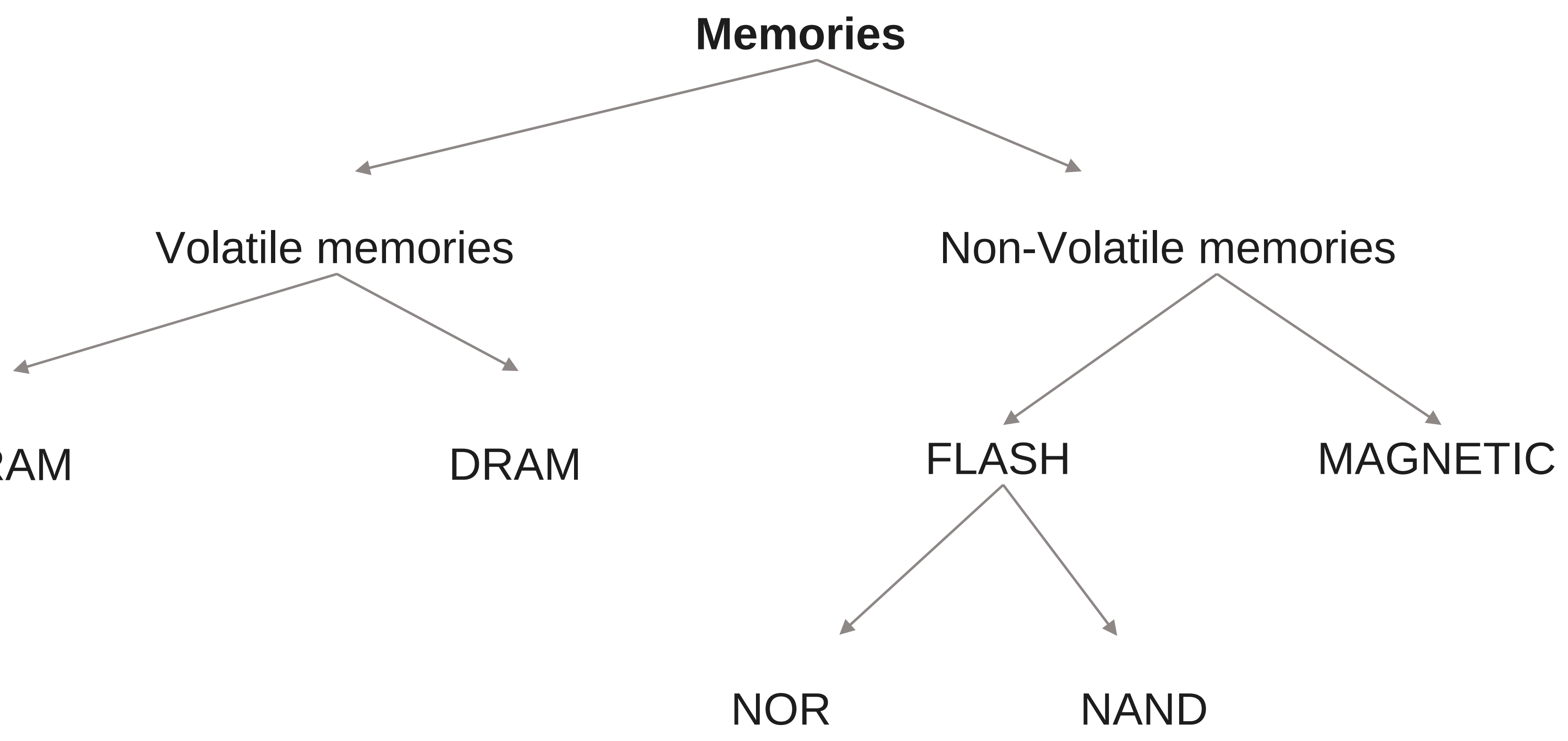
```
const int LOOP_COUNT = 5;
int b[5];
int a[5] = { 20, -19, 773, 47, -146};
void main(void)
{
   int *ptr;
   ptr = b;

  while(port->portA==0)
  {
    for(int i=0;i<LOOP_COUNT;i++)
    {
        ptr[i] = a[i];
    }
  }
}
```

Corresponding Program sections

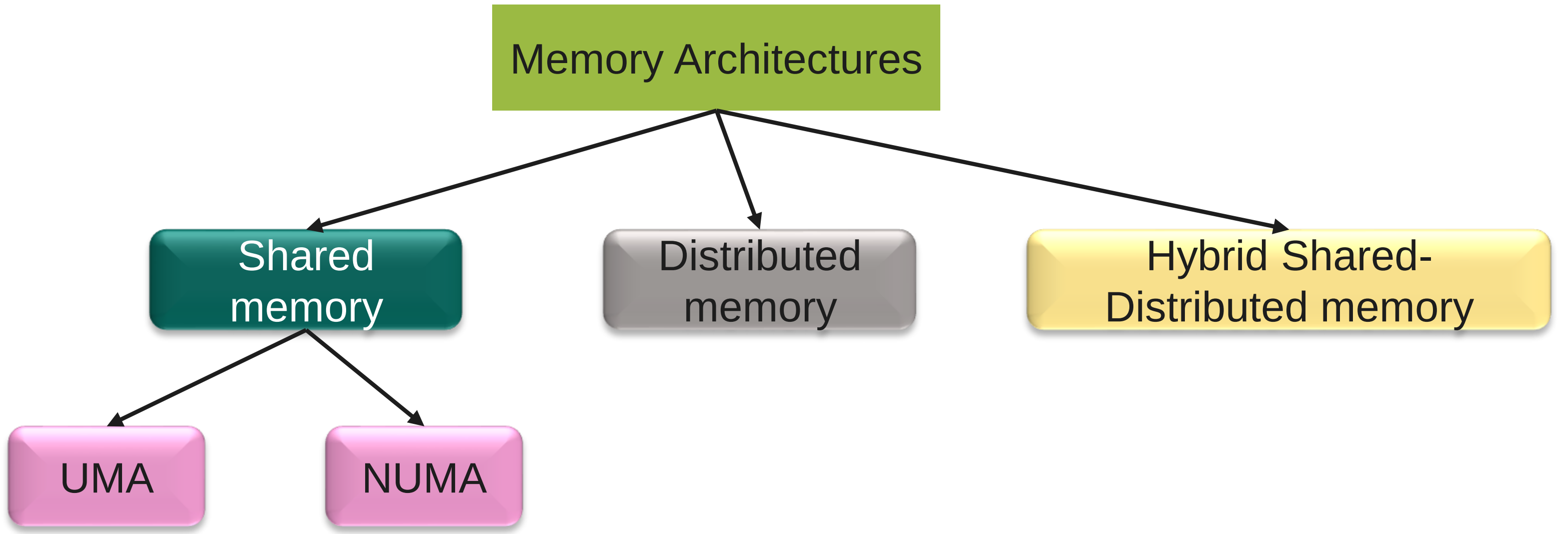
- Constant data section / RODATA section
- BSS section
- DATA section
- TEXT section

QUESTION: Where should these sections exist?

Memories-2/2

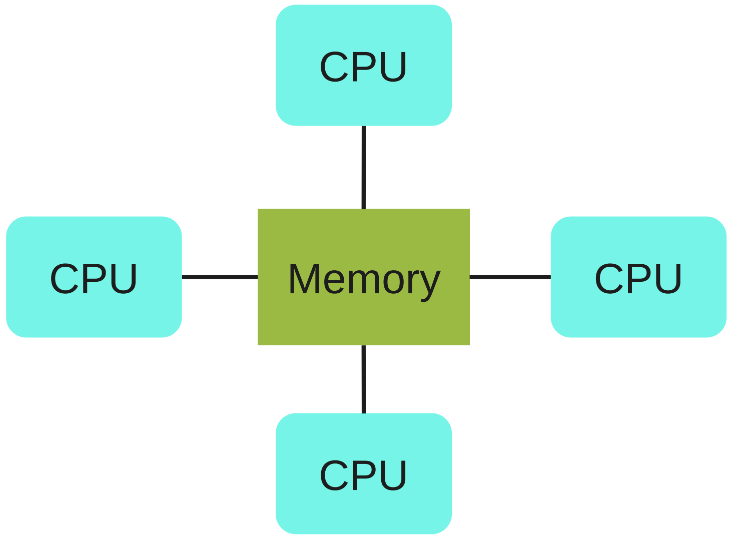

Memories
Volatile memories
Non-Volatile memories
SRAM
DRAM
FLASH
MAGNETIC
NOR
NAND

Memory architectures



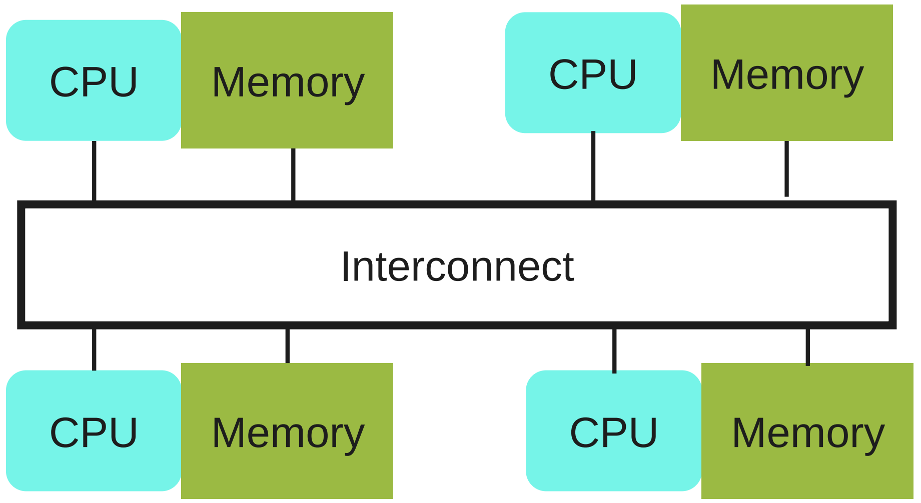
Shared Memory – Global address space

Uniform Memory Access



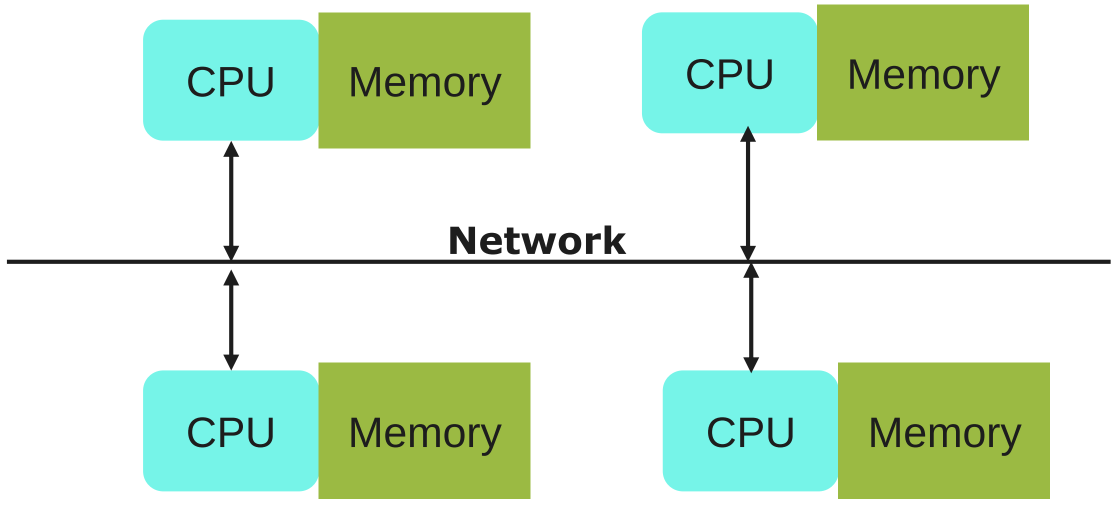
All CPUs have equal memory access latency

Non-Uniform Memory Access



Access latencies of local and remote memories are different

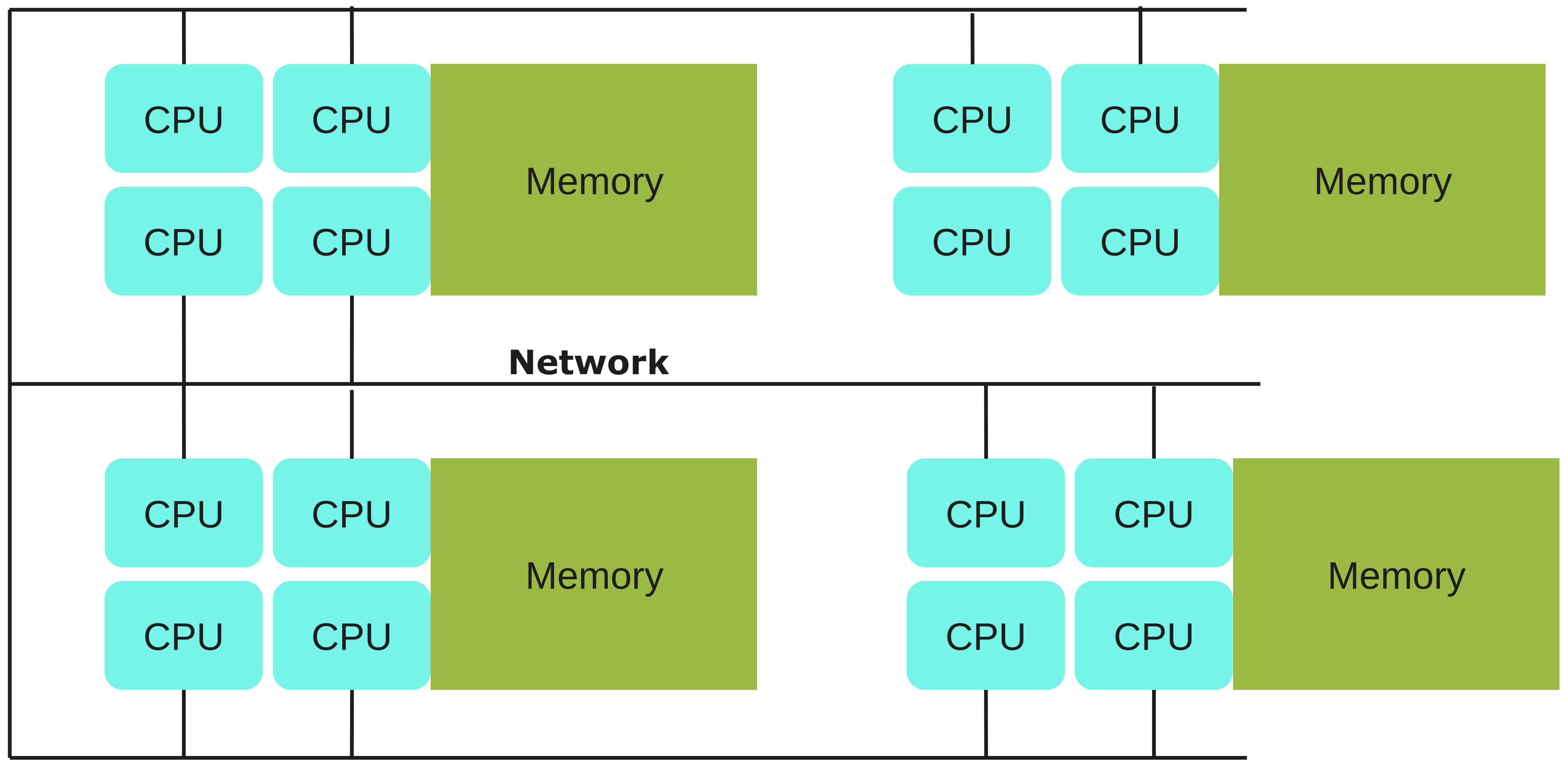
Distributed Memory – Private address space



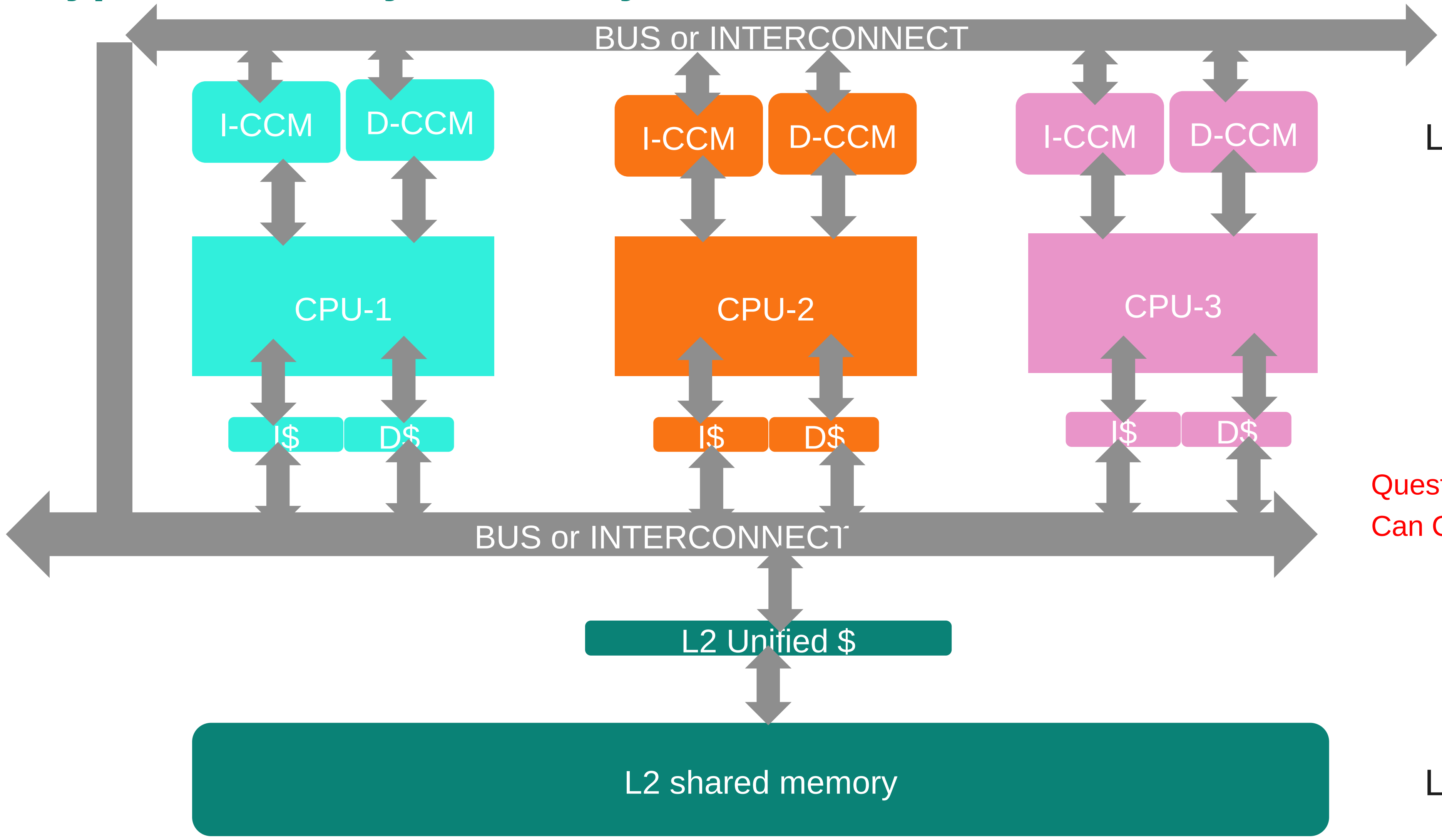
A CPU cannot access the memory space of another CPU

Typical HPC setup

Hybrid memory model



Typical memory hierarchy on Microcontrollers



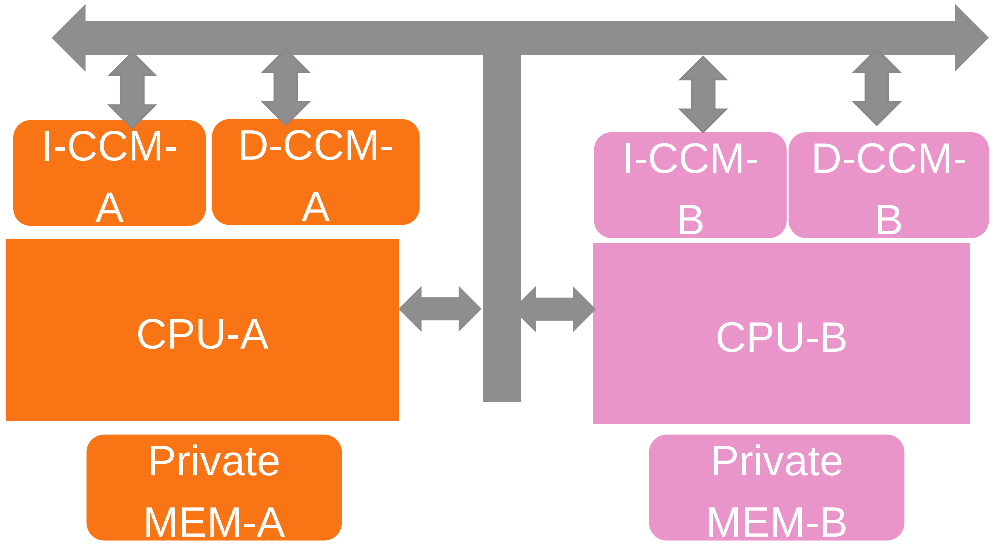
Level-1 memories (NUMA)

Level-0 memories

Question:
Can CPU-2 access caches of CPU-1?

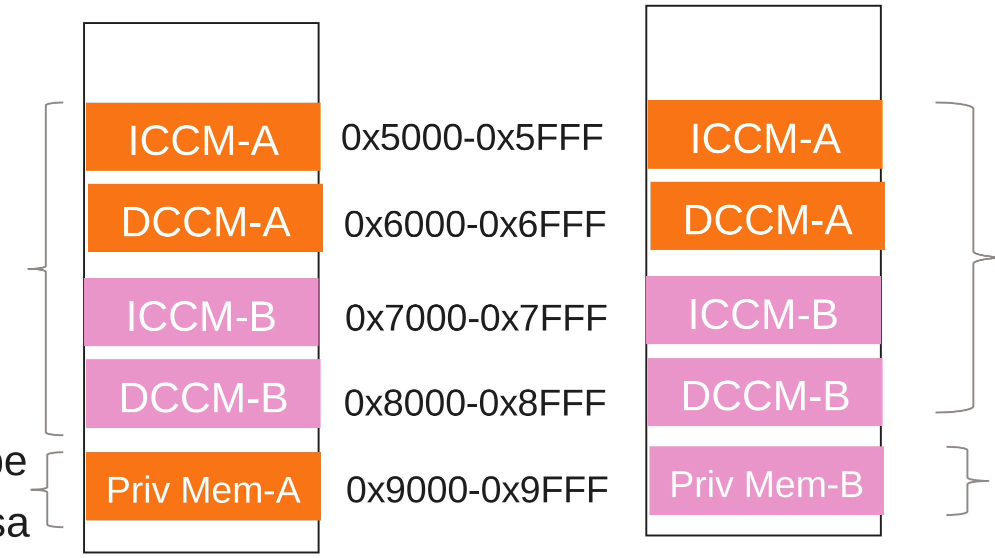
Level-2 memories (UMA)

Mapping of memories to CPU address space



These 4 memories are accessible to both CPUs.

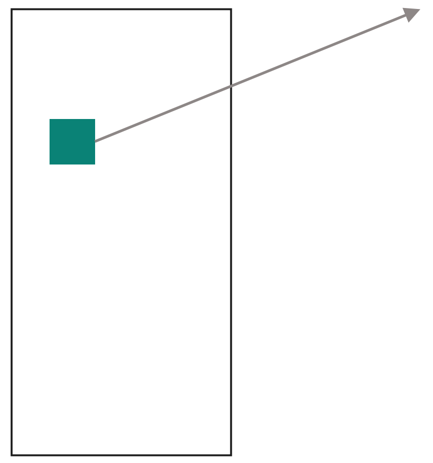
CPU-A’s private memory cannot be accessed by CPU-B and vice-versa



Both CPUs see ICCM-A at 0x5000, ICCM-B at 0x7000

When an address of 0x9000 is accessed, the two CPUs end up fetching stuff from their private memories only!

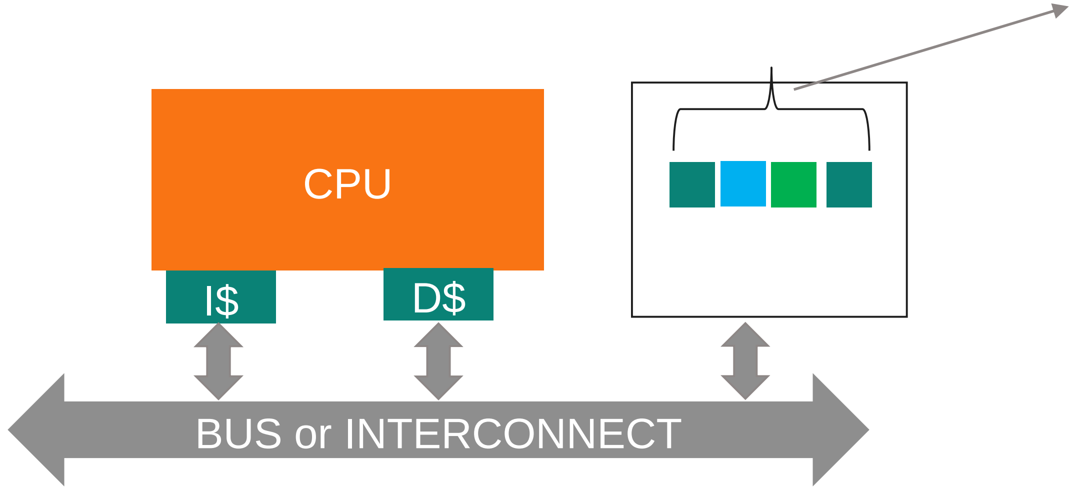
Cache memories – 1/2



1. Memories are slow, CPUs are fast
2. The pipeline of a CPU will stall waiting for data to be written to/returned from memory
3. A lot of energy is wasted in moving data between CPU and memory
4. If your program has accessed a certain location, it likely may access it yet again soon
5. If your program has accessed a certain location, it likely may access surrounding locations too
6. **What can we do to reduce the pains of slow memory access and energy?**

ANSWER: Cache memory

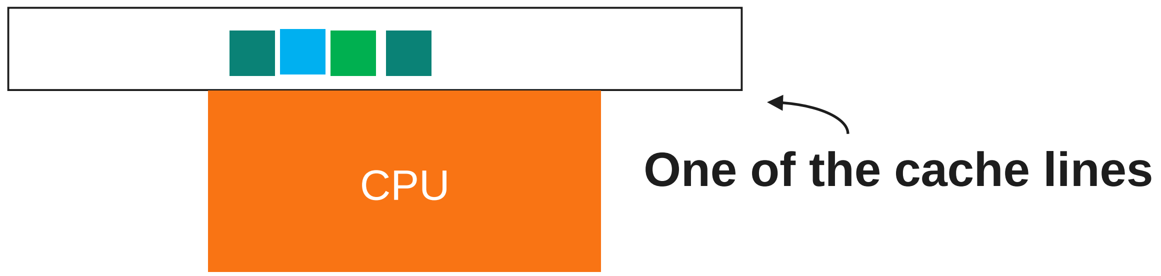
Cache memories – 2/2



1. When your program attempts to access any of these four words, all four words are brought into the cache

2. This is called CACHE LINE FILL

3. The requested word is now forwarded to CPU by cache

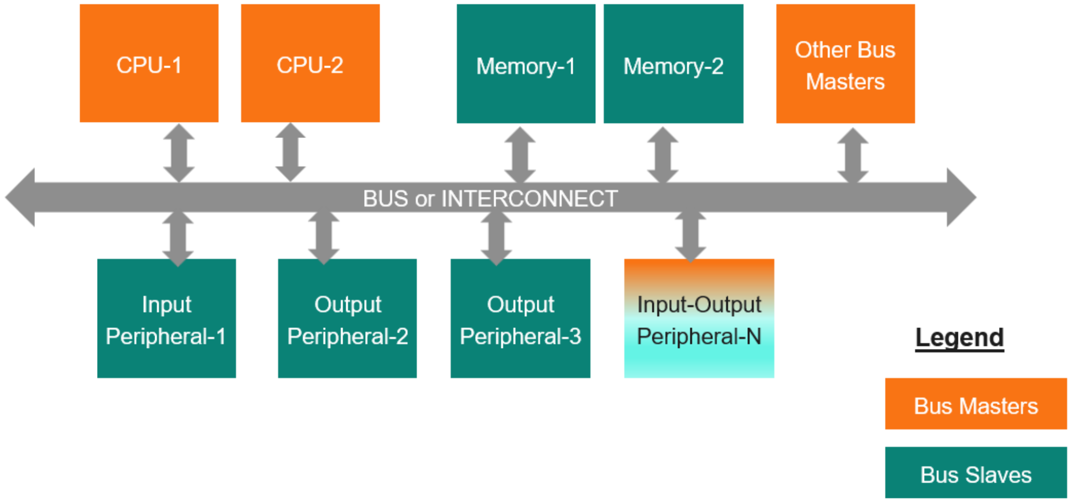


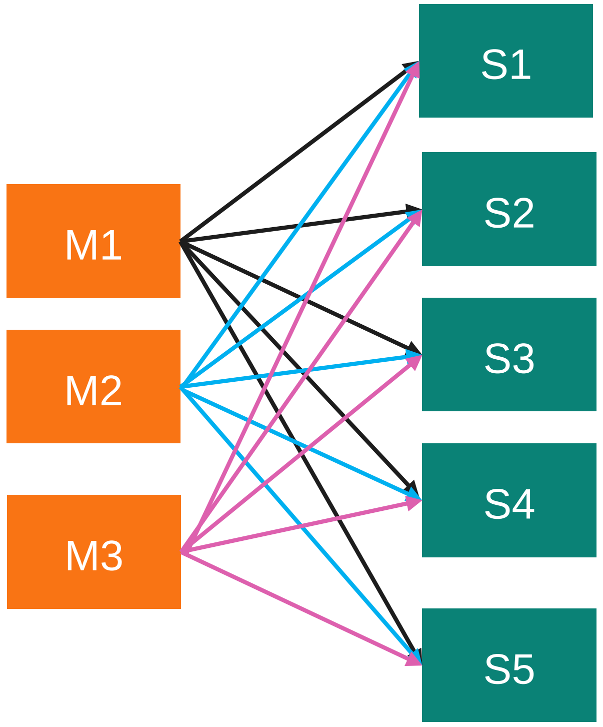
4. From now on, any access of the 4 address locations will result in fetches from the cache line

Homework topics:
Cache policies (Write Through, Write Back)
Cache line mapping (associativity), Eviction and refill policies,
Cache coherence

Interconnects – 1/4

$$\begin{tikzcd}
 & \text{Typical MCU Interconnect types} \arrow[dl] \arrow[dr] & & \\
\text{Shared bus} & & \text{Cross-Bar} \arrow[dl] \arrow[dr] & \\
 & \text{Partial} & & \text{Full}
\end{tikzcd}$$



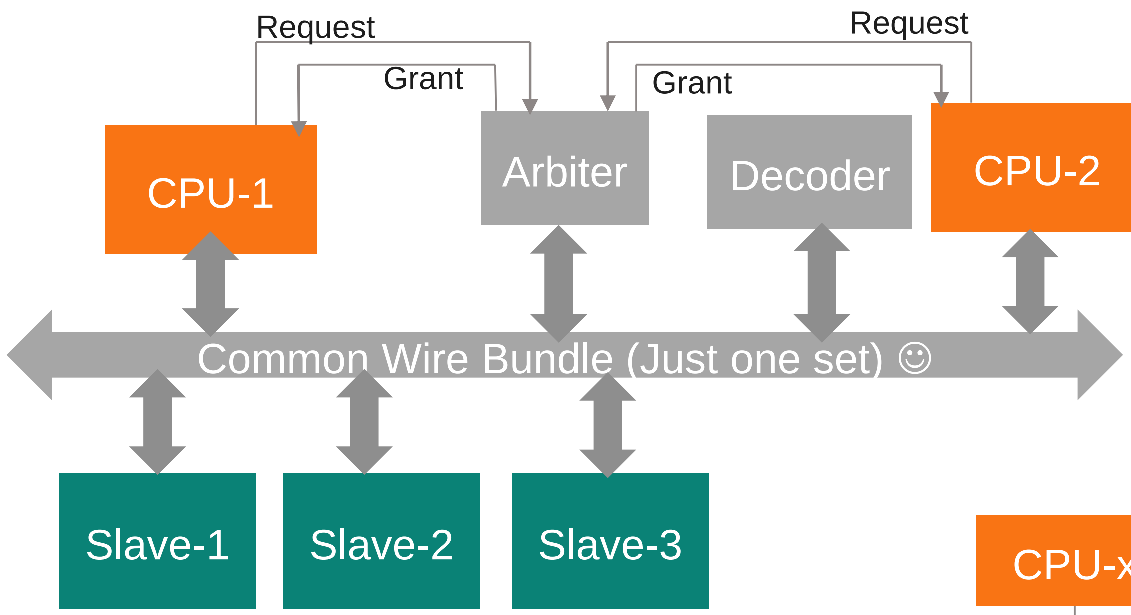


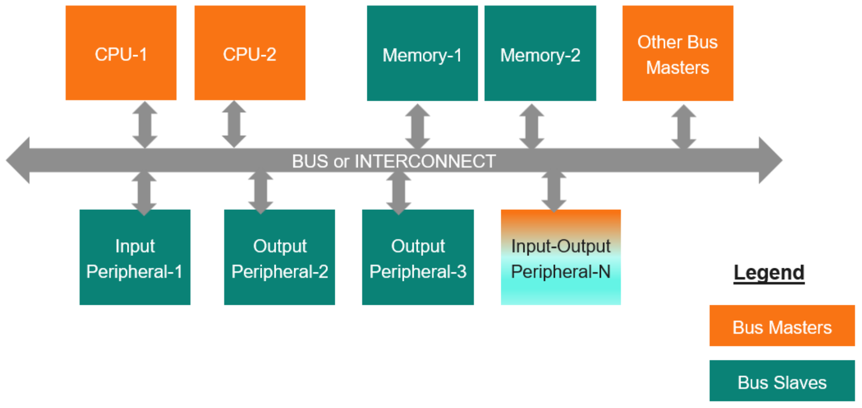
15 wire bundles for connecting 3 masters to 5 slaves!

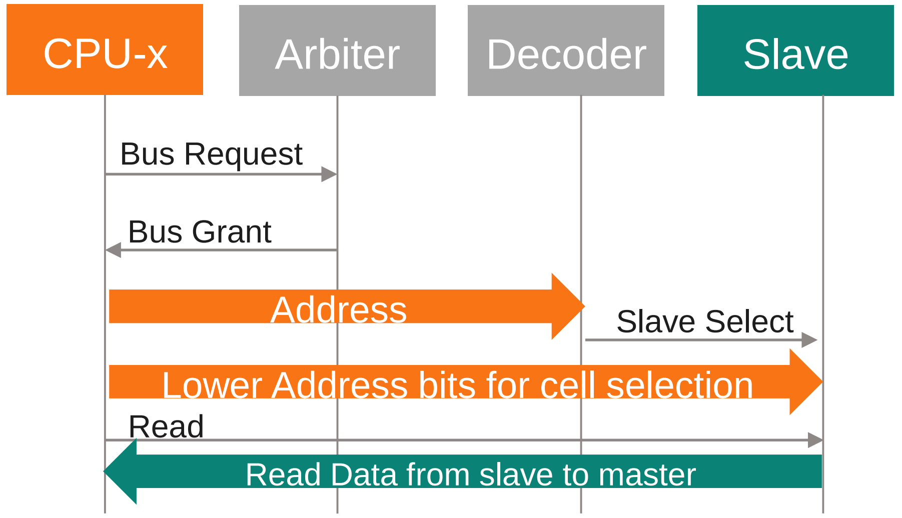
That is already quite a lot!

Need a mechanism to reduce the bundle count.

Interconnects – 2/4



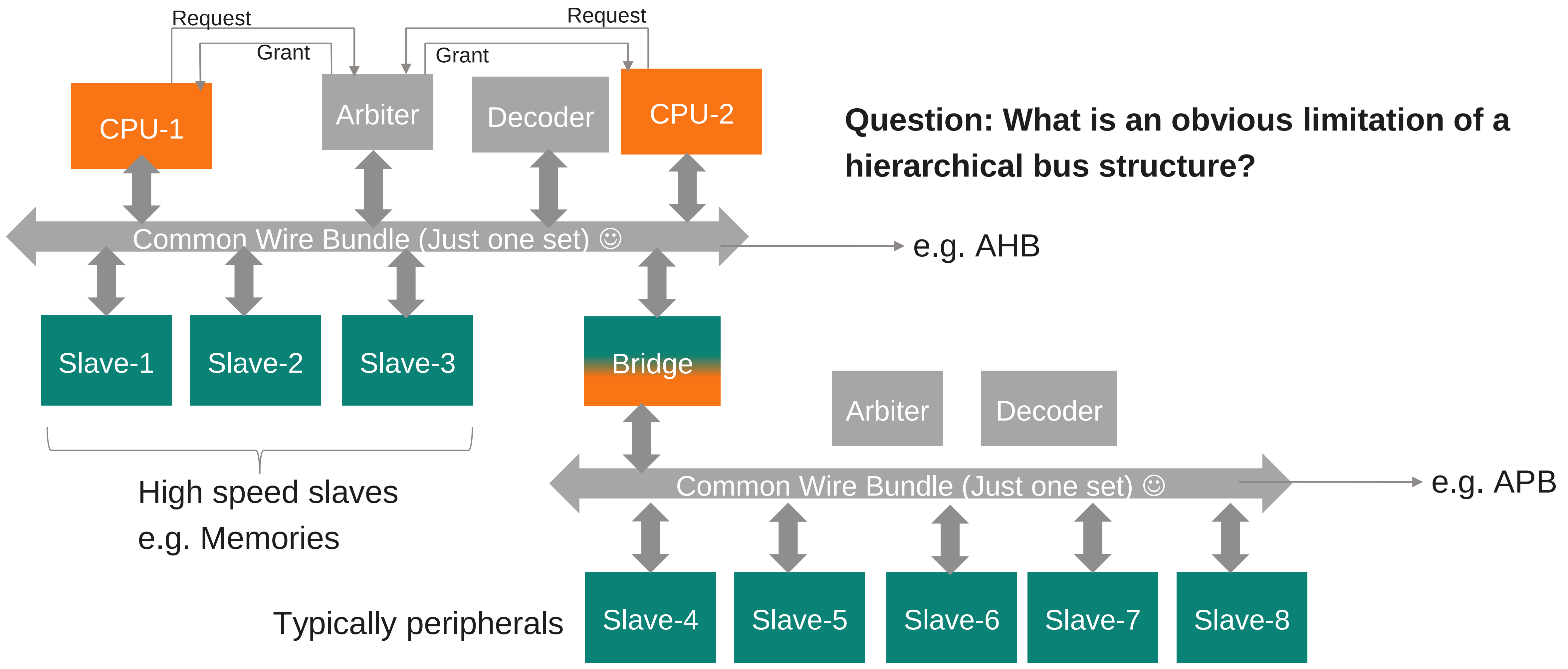




CPU-x
Arbiter
Decoder
Slave
Bus Request
Bus Grant
Address
Slave Select
Lower Address bits for cell selection
Read
Read Data from slave to master

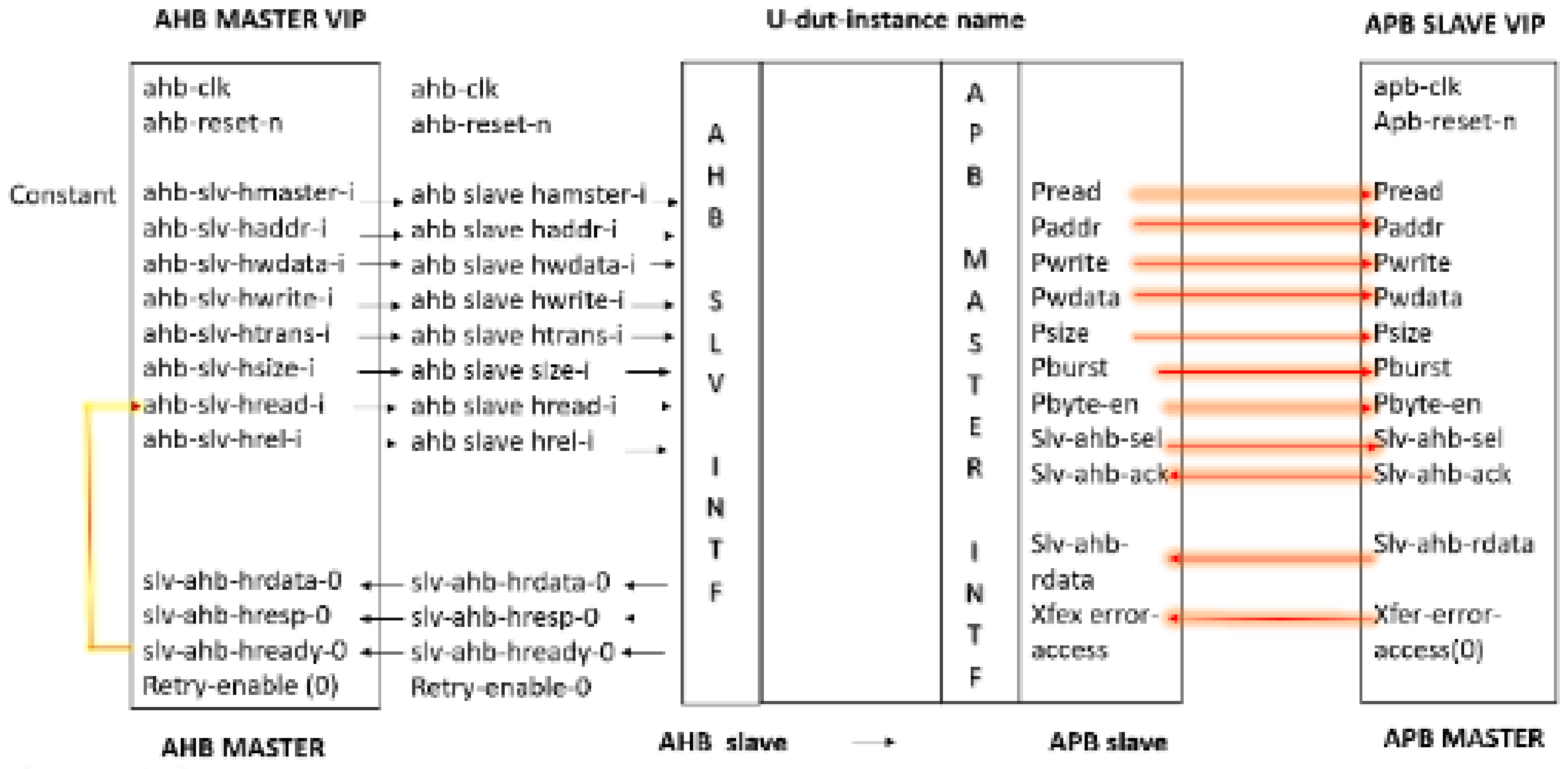
Interconnects – 3/4

On most MCUs, this is a typical multi-level bus structure!



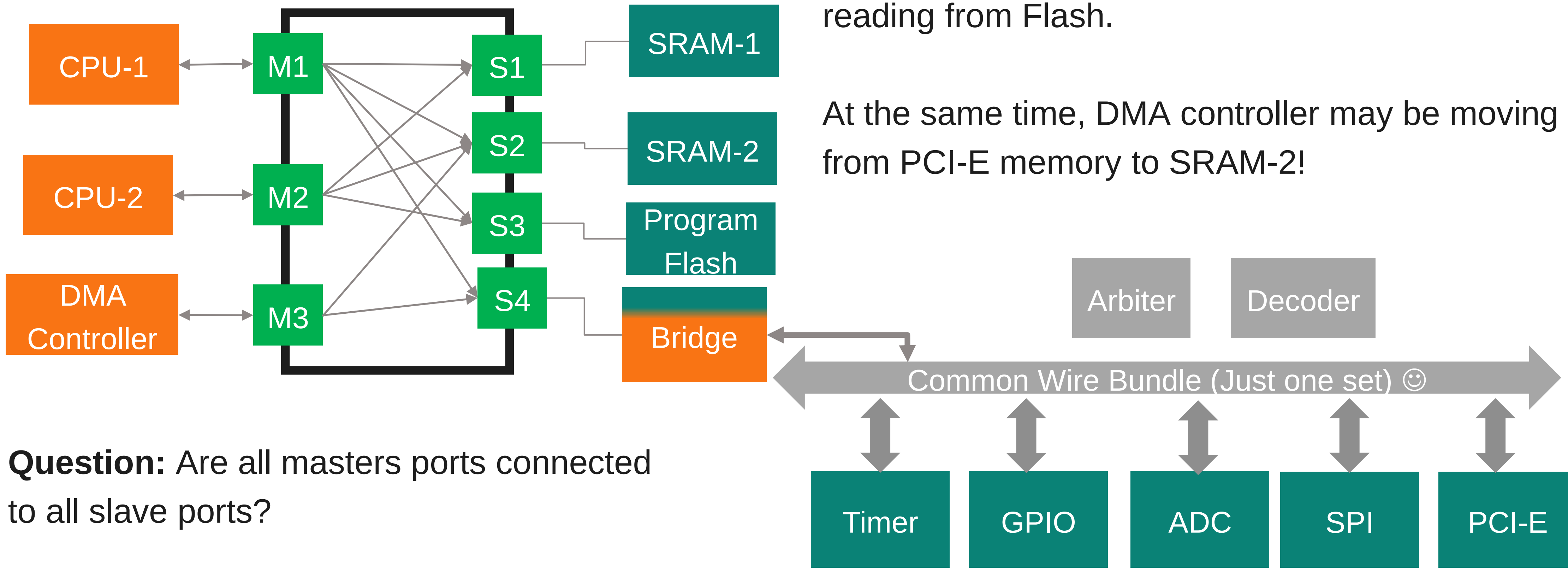
Question: What is an obvious limitation of a hierarchical bus structure?

AHB, APB, Bridge



Interconnects – 4/4

What we want is multiple bus masters and slaves to be transacting concurrently!

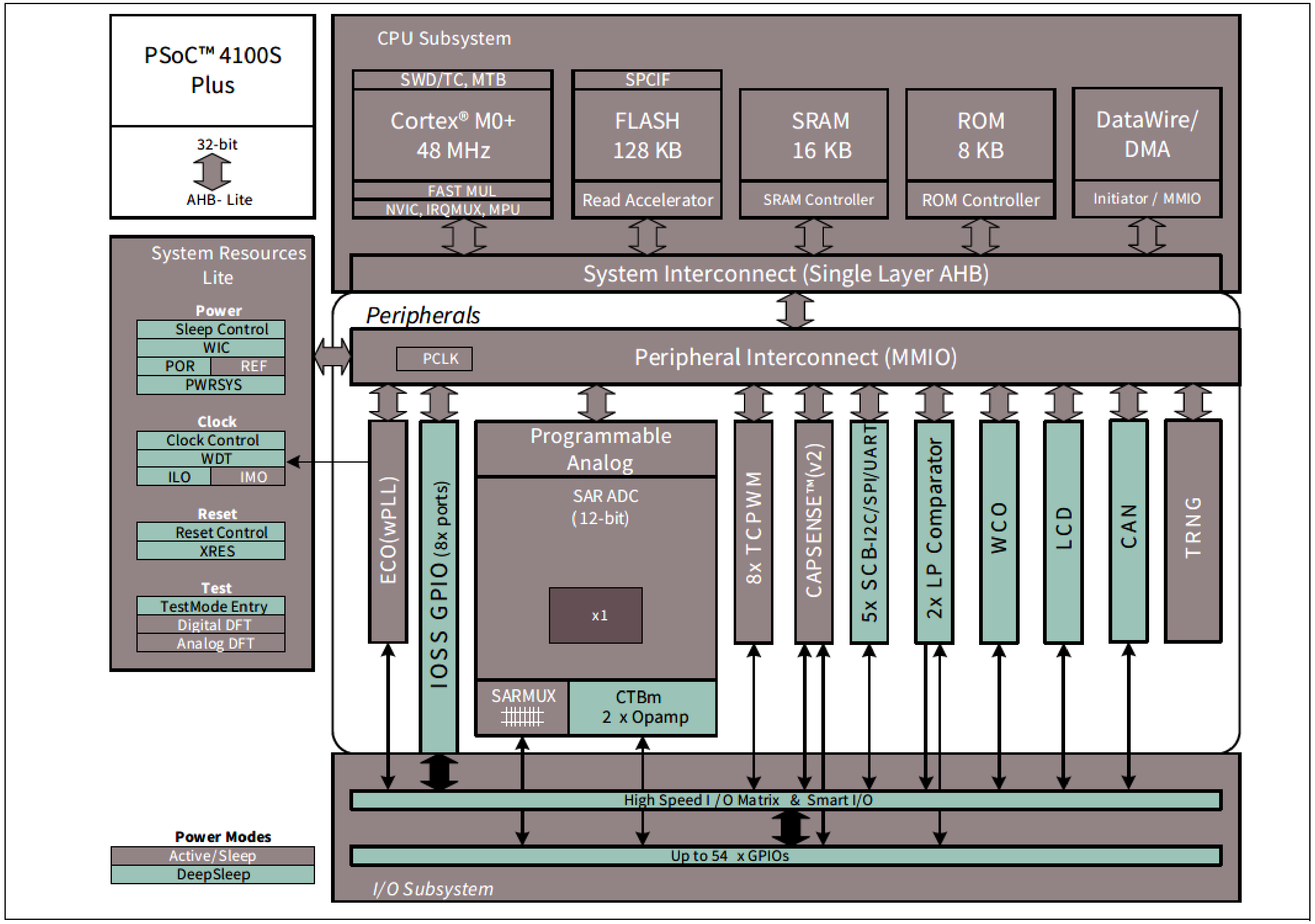


While CPU-1 is writing to SRAM-1, CPU-2 may be reading from Flash.

At the same time, DMA controller may be moving data from PCI-E memory to SRAM-2!

Question: Are all masters ports connected to all slave ports?

PSoC, Finally!



infineon