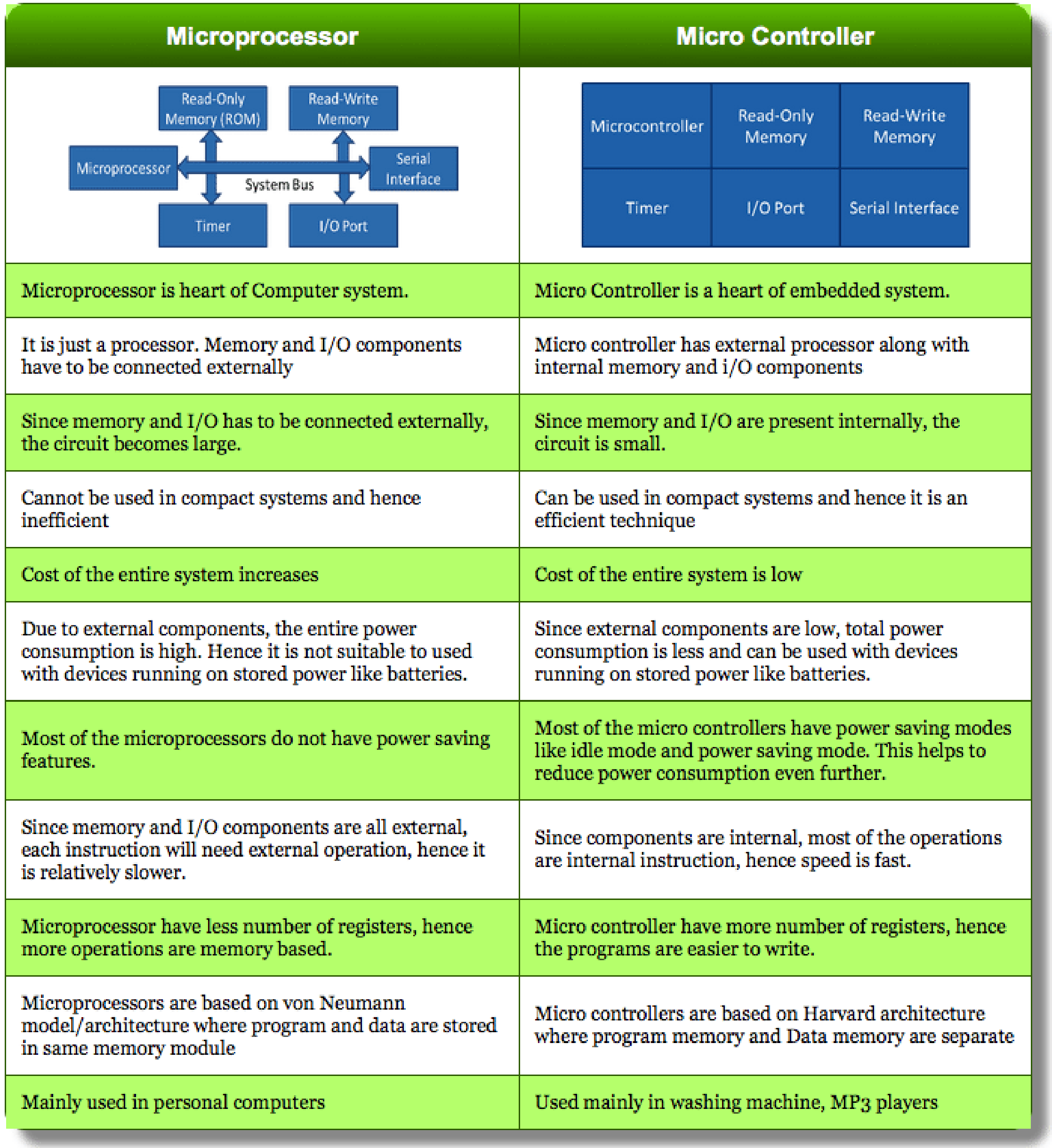
Live Lecture Series #1

Practical Differences - Microprocessor's vs Microcontroller's

What Does Google Say?



Microprocessor	Micro Controller
Read-Only Memory (ROM) Read-Write Memory Microprocessor System Bus Serial Interface Timer I/O Port	Microcontroller Read-Only Memory Read-Write Memory Timer I/O Port Serial Interface
Microprocessor is heart of Computer system.	Micro Controller is a heart of embedded system.
It is just a processor. Memory and I/O components have to be connected externally	Micro controller has external processor along with internal memory and i/O components
Since memory and I/O has to be connected externally, the circuit becomes large.	Since memory and I/O are present internally, the circuit is small.
Cannot be used in compact systems and hence inefficient	Can be used in compact systems and hence it is an efficient technique
Cost of the entire system increases	Cost of the entire system is low
Due to external components, the entire power consumption is high. Hence it is not suitable to used with devices running on stored power like batteries.	Since external components are low, total power consumption is less and can be used with devices running on stored power like batteries.
Most of the microprocessors do not have power saving features.	Most of the micro controllers have power saving modes like idle mode and power saving mode. This helps to reduce power consumption even further.
Since memory and I/O components are all external, each instruction will need external operation, hence it is relatively slower.	Since components are internal, most of the operations are internal instruction, hence speed is fast.
Microprocessor have less number of registers, hence more operations are memory based.	Micro controller have more number of registers, hence the programs are easier to write.
Microprocessors are based on von Neumann model/architecture where program and data are stored in same memory module	Micro controllers are based on Harvard architecture where program memory and Data memory are separate
Mainly used in personal computers	Used mainly in washing machine, MP3 players

My Definitions

MCU: Microcontroller

- Processor + Storage + Peripheral's* + Power Management.

MPU: Microprocessor

- Simply the Processor.

SBC: Single Board Computer

- Everything needed to run a desktop OS on a single board.

** Some peripherals will still require external PHY’s/Hardware (Ethernet)*

Common Examples

MCU’s

- Arduino (Microchip ATMEGA328).
- STMicro STM32 (Not like you can buy them).
- Espressif ESP32.

MPU’s

- Raspberry Pi.
- Beagle Bone.
- Jetson Nano.
- Computer CPU’s.

MCU/MPU Pro’s and Con’s

Commonly discussed MCU Pro’s:

- Low price.
- Easier to develop for.

Common discussed MPU Pro’s

- Higher processing power.
- More memory.

A direct Pro and Con list doesn’t tell the whole story.

MCU/MPU Pro’s and Con’s (Cont’d)

Real-World example: *Audio mixing board*

- Takes in several audio channels.
- Mixes, applies different filters/effects.
- Outputs a single audio stream.

MCU based approach

- Uses a mixture of analog components and digital IC’s.
- MCU controls the analog and IC’s.

MPU based approach

- Feed all audio channels directly into the processor.
- All mixing is done in software.

MCU/MPU Pro’s and Con’s (Cont’d)

MPU solution:

- Incredibly flexible.
- Less audio hardware.
- Potentially cheaper.
- Easy to implement on software.

MCU solution

- Limited in flexibility.
- Lots of analog audio hardware.
- Real time, minimal latency.

Operating Systems (OS)

MPU’s typically will be running a *true* OS.

MPU’s can also run without one, but it is not as common.

This results in a significant overhead, limits real-time performance.

MCU’s run either bare-metal code, or a Real Time OS (RTOS).

RTOS’ focus on time-based scheduling, making them deterministic.

Technically you can run a *true* OS on an MCU.

MPU’s Are the Kings of Interfaces

No matter the interface you need, an MPU has it.

DigiKey Search

What Makes Designing With MPU’s Difficult?

Why Don’t You See DIY SBC’s Like You Do MCU’s (Arduino/ESP32/STM32)?

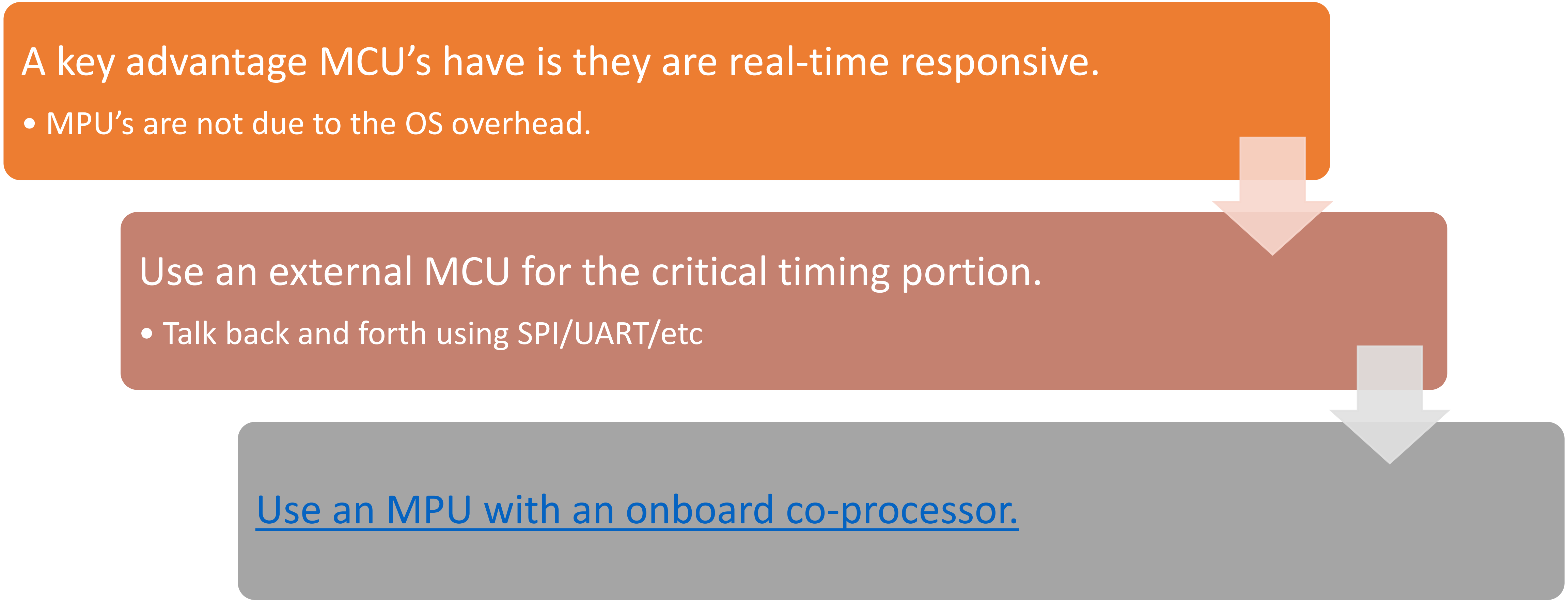
- Fun with BGA’s!
- No internal power management control
- Requires external flash* and RAM.
  - DDR memory often requires high layer count boards and controlled impedance

**Sometimes there will be some small flash onboard.*

What Makes Designing With MPU’s Difficult? (Cont’d)

- Getting Linux up and running can be an awful experience.
  - **Highly** dependent on vendor support. STM32MP1
  - Ubuntu/Debian, Yocto, BuildRoot.

Dealing With Real-Time Requirements



A key advantage MCU’s have is they are real-time responsive.

- MPU’s are not due to the OS overhead.

Use an external MCU for the critical timing portion.

- Talk back and forth using SPI/UART/etc

Use an MPU with an onboard co-processor.

Applications For A MPU Approach

Computationally Heavy

- Computer vision.
- Machine Learning.
- Anything with a lot of calculations.
- Onboard graphics engine.

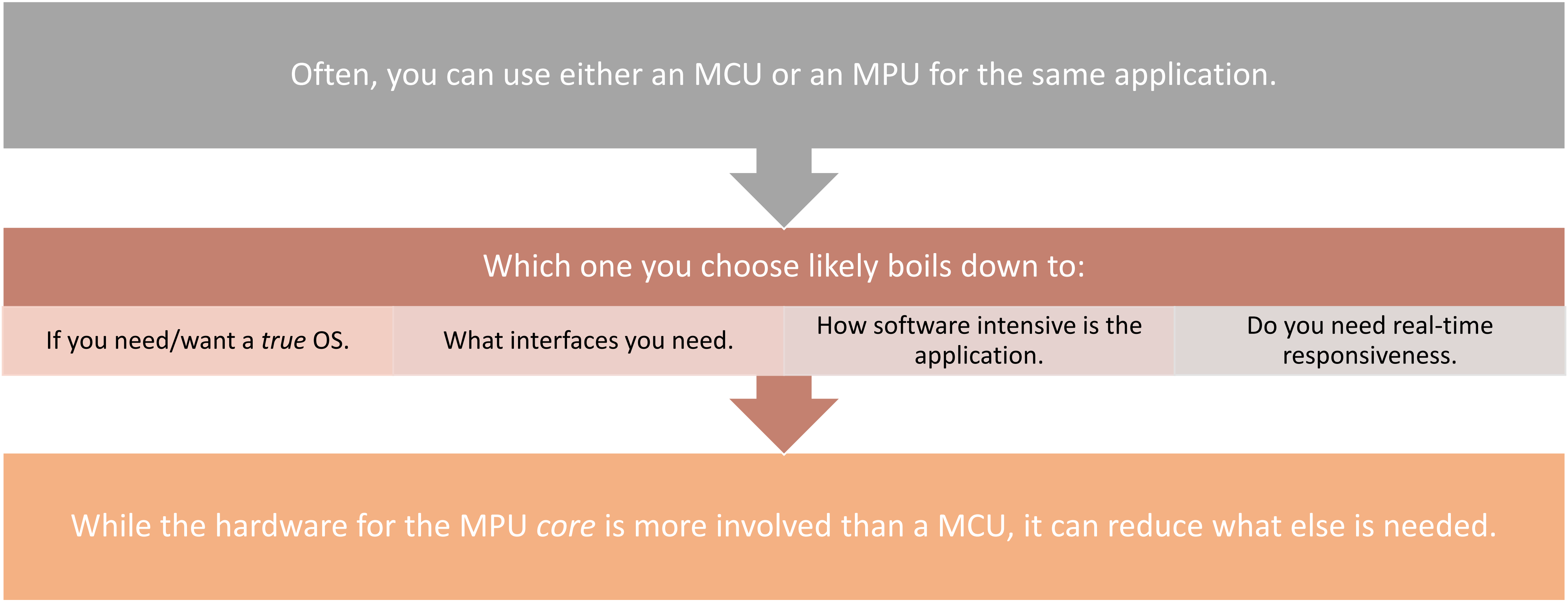
Memory Intensive

- Driving Displays.
- Large LED arrays.

Specific Interfaces

- High-Speed USB.
- HDMI.
- Ethernet.

Conclusion



Often, you can use either an MCU or an MPU for the same application.

Which one you choose likely boils down to:

- If you need/want a *true* OS.
- What interfaces you need.
- How software intensive is the application.
- Do you need real-time responsiveness.

While the hardware for the MPU *core* is more involved than a MCU, it can reduce what else is needed.