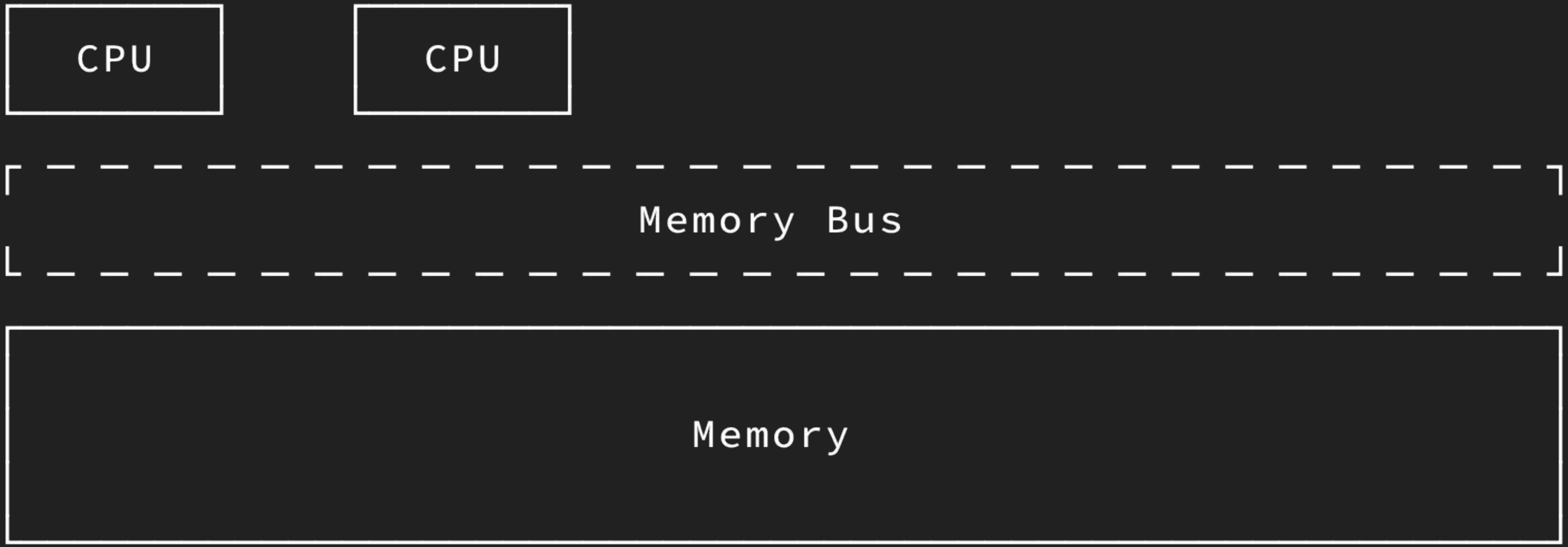
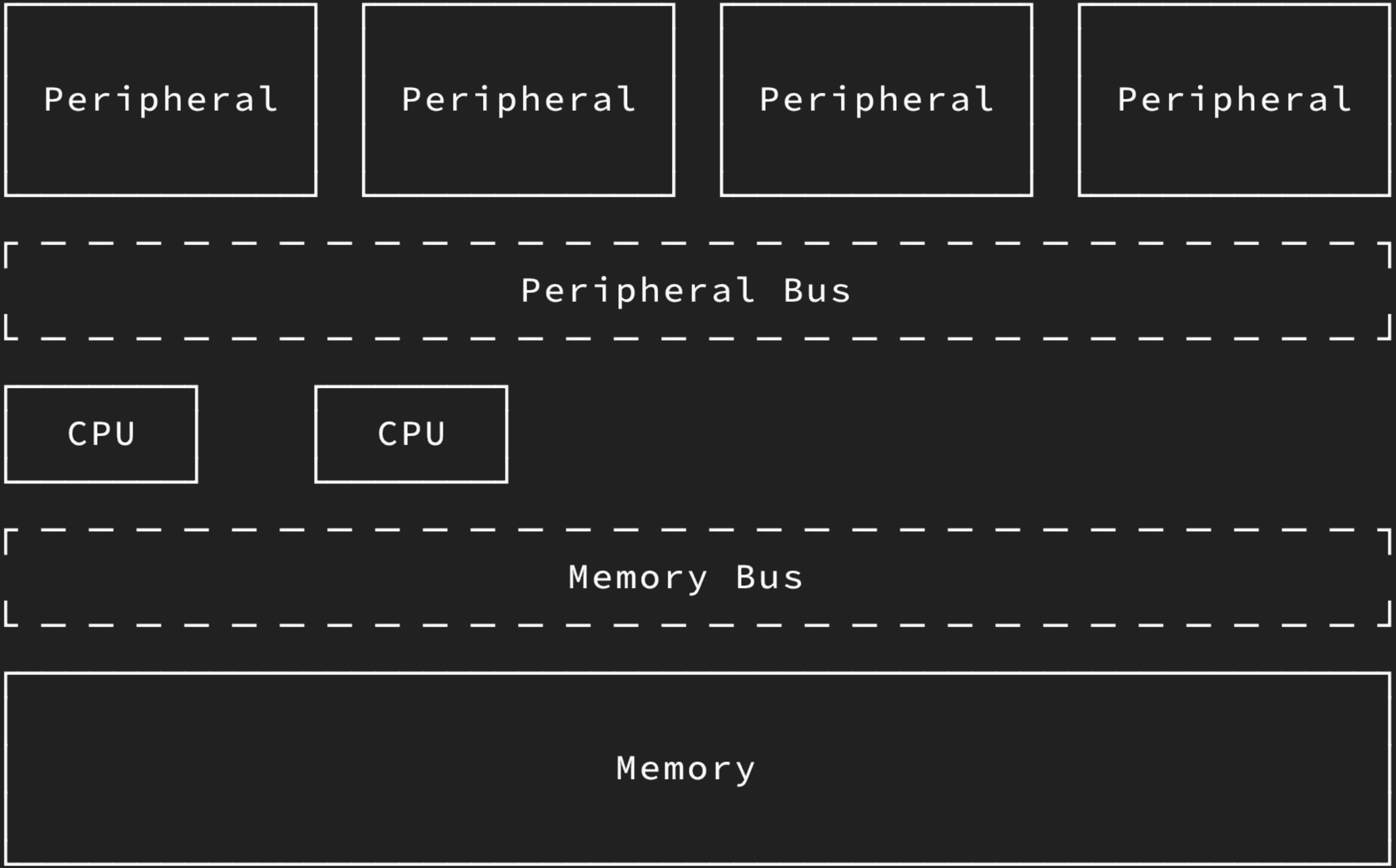
direct memory access for the uninitiated

DMA means
"direct memory access"

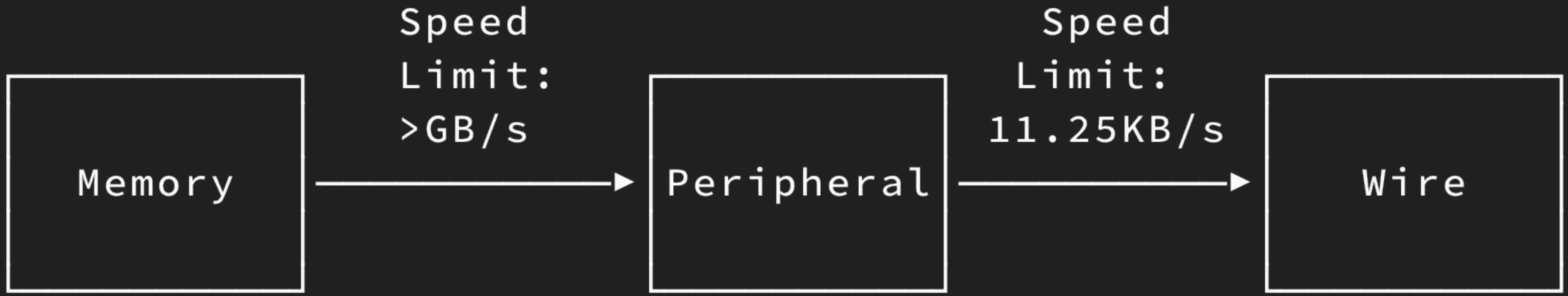
but what does memory access mean, anyway?

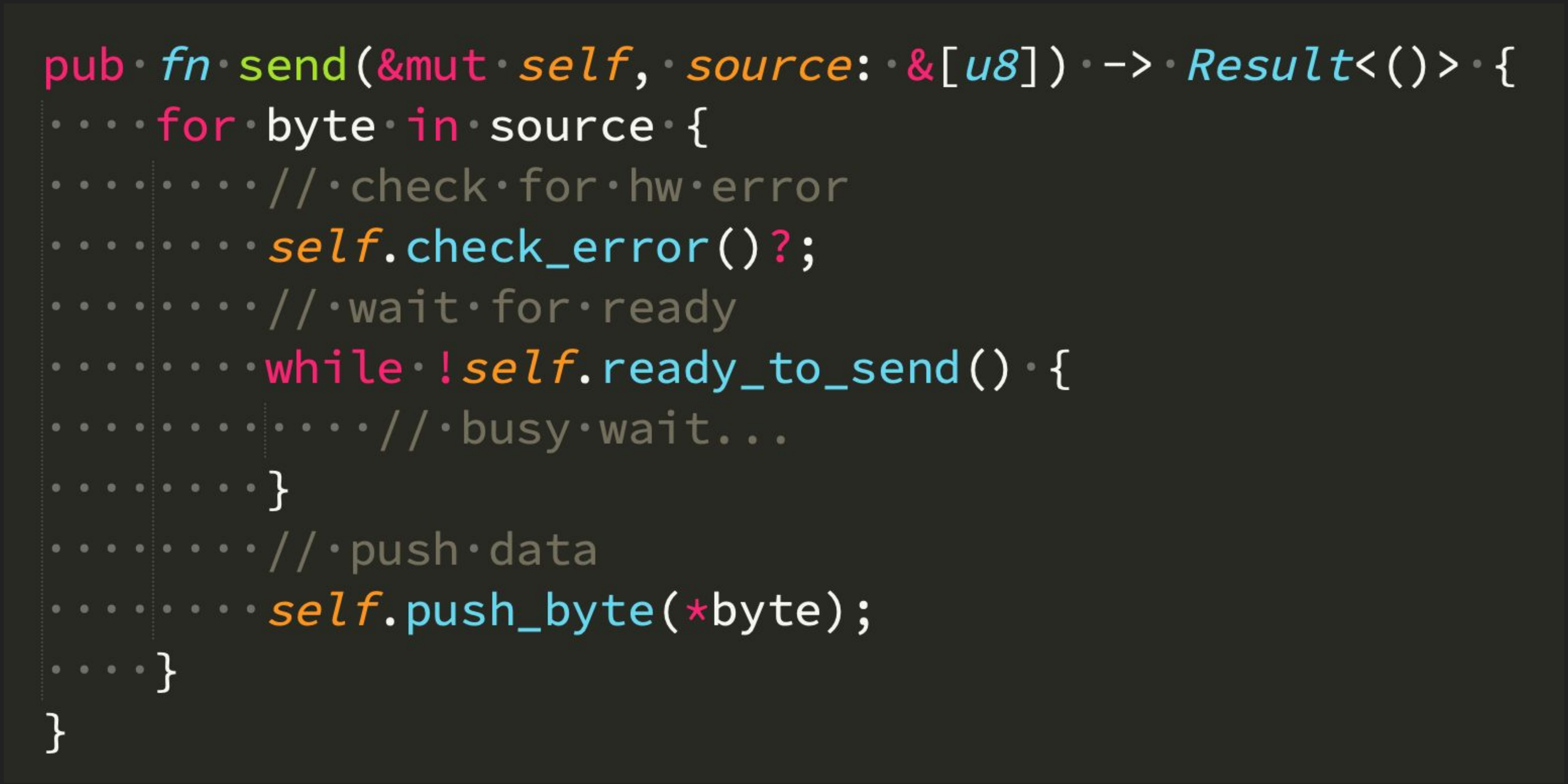

CPU
CPU
Memory Bus
Memory


Peripheral
Peripheral
Peripheral
Peripheral
Peripheral Bus
CPU
CPU
Memory Bus
Memory

problem: peripherals are sloooooooooooooooow

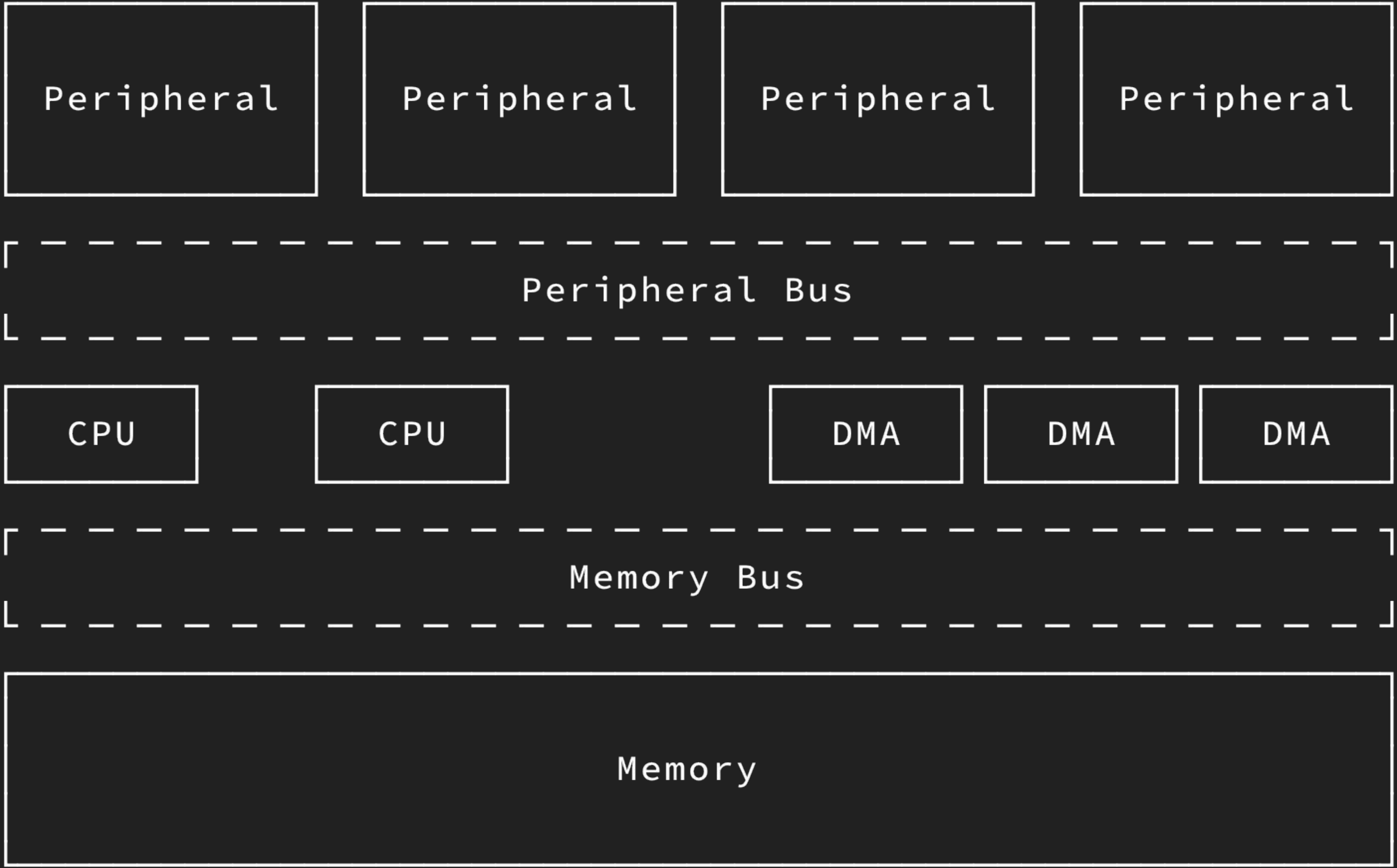
a desktop might have memory bandwidth of 100s of GB/s a microcontroller might have memory bandwidth of 100s of MB/s a “normal” serial port is 115200 baud or 11.25KB/s


Memory
Speed Limit: >GB/s
Peripheral
Speed Limit: 11.25KB/s
Wire



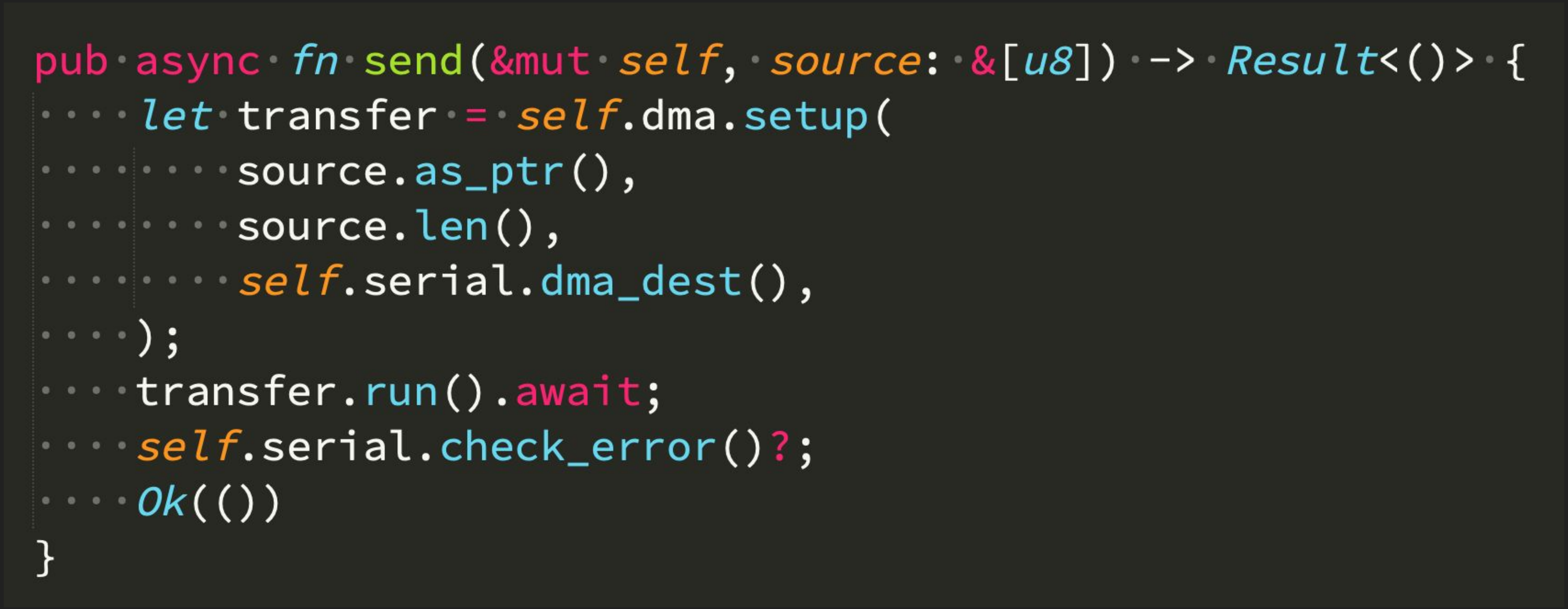
```rust
pub fn send(&mut self, source: &[u8]) -> Result<()> {
    for byte in source {
        // check for hw error
        self.check_error()?;
        // wait for ready
        while !self.ready_to_send() {
            // busy wait...
        }
        // push data
        self.push_byte(*byte);
    }
}
```

DMA is for babysitting memory copies


Peripheral
Peripheral
Peripheral
Peripheral
Peripheral Bus
CPU
CPU
DMA
DMA
DMA
Memory Bus
Memory

it's like a CPU core where the only instruction is "copy" the CPU gives DMA the source and destination, and says "go"

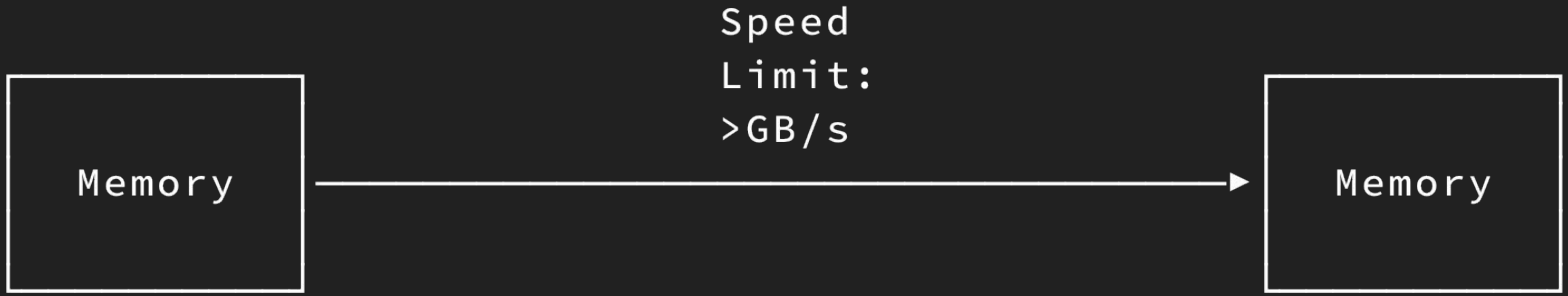
DMA takes over, and some time later, it says “done” this is great: we go from "busy polling" to "event driven" this is great for async: we *love* event driven in async

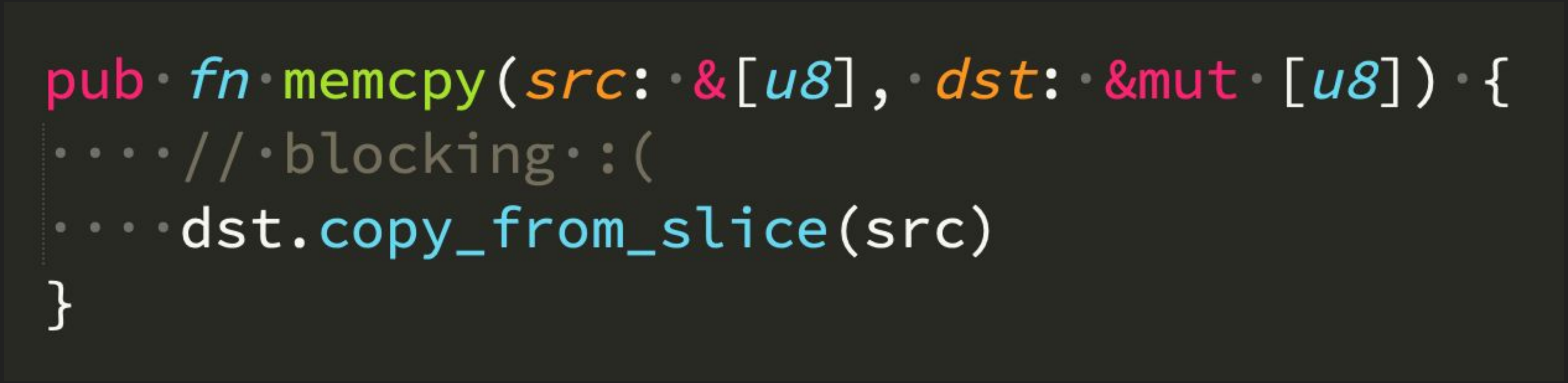


```rust
pub async fn send(&mut self, source: &[u8]) -> Result<()> {
    let transfer = self.dma.setup(
        source.as_ptr(),
        source.len(),
        self.serial.dma_dest(),
    );
    transfer.run().await;
    self.serial.check_error()?;
    Ok(())
}
```

one more thing...

DMA can also copy from memory to memory


Speed
Limit:
>GB/s
Memory
Memory



```rust
pub fn memcpy(src: &[u8], dst: &mut [u8]) {
    // blocking :(
    dst.copy_from_slice(src)
}
```

```rust
pub async fn memcpy(dma: &mut Dma, src: &[u8], dst: &mut [u8]) {
    let transfer = dma.setup(
        source.as_ptr(),
        source.len(),
        dst.as_mut_ptr(),
        dst.len(),
    );
    transfer.run().await;
}
```