

# Input and Output

## Fundamentals

Dr Andrew Scott  
a.scott@lancaster.ac.uk

1

---

---

---

---

---

---

---

# Overview

- Look at I/O mechanisms
  - How we implement and control access to devices
  - Clocks and Timers
    - Used to control and synchronise operations

2

---

---

---

---

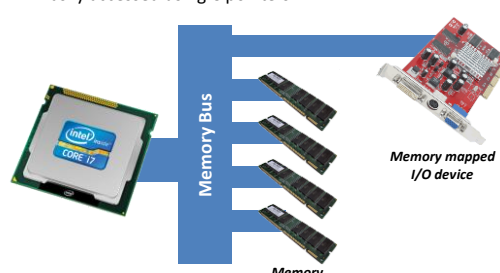
---

---

---

# Memory Mapped I/O

- I/O devices visible as normal memory locations
  - Easily accessed using C pointers



The diagram illustrates the Memory Mapped I/O architecture. A central blue vertical bar is labeled 'Memory Bus'. To the left of the bus is an Intel Core i7 processor. To the right of the bus are four memory modules, with the label 'Memory' at the bottom. Further to the right, connected to the bus, is a red circuit board labeled 'Memory mapped I/O device'.

3

---

---

---

---

---

---

---

### Isolated I/O

- Separate bus for I/O devices
  - Easier interfacing: distinct from high-speed memory bus
  - Accessed via special instructions
    - Generally not supported by compilers (must use assembler)

The diagram illustrates the Isolated I/O architecture. On the left, two I/O devices are shown, each connected to a separate I/O bus. The I/O bus is labeled 'I/O Bus' and has two ports: 'PORT 0x100' and 'PORT 0x180'. These ports are connected to 'Isolated I/O devices'. In the center, a CPU (Intel Core i7) is shown. To the right of the CPU is a 'Memory Bus' which connects to 'Memory' (RAM modules). The I/O bus and Memory bus are separate, allowing for easier interfacing of I/O devices without interfering with the high-speed memory bus.

4

---

---

---

---

---

---

---

---

### Optimising Memory Use

- Circular buffers help improve data throughput
  - Help decouple operation of processor and device
- Status Register holds flags for
  - Device ready
  - Data available, etc.

The diagram shows a green rectangular block representing a device interface. Inside the block, there are two circular buffers: 'Circular Buffer Input' and 'Circular Buffer Output'. To the right of the buffers are two registers: 'Status Register' and 'Control Register'. Below the buffers, there is a yellow and black striped bar representing data flow.

5

---

---

---

---

---

---

---

---

### Direct Memory Access

- Alternative to programmed I/O
- Relieves CPU of data transfer

The diagram illustrates the Direct Memory Access (DMA) process. It shows a CPU (Intel Core i7) connected to a 'DMA Controller'. The DMA Controller has three registers: 'Address', 'Count', and 'Control'. A 'Disk Controller' is connected to the DMA Controller and a 'Buffer'. The 'Memory' (RAM) is also connected to the DMA Controller. The process is shown in five steps: 1. Configure DMA controller (CPU to DMA Controller), 2. Transfer Request (Disk Controller to DMA Controller), 3. Data Transfer (DMA Controller to Memory), 4. ACK (Memory to DMA Controller), and 5. Interrupt (DMA Controller to CPU).

6

---

---

---

---

---

---

---

---

### Clocks and Timers

- Real Time Clock (RTC)
  - Returns encoding of current date
- Programmable Interrupt Timer (PIT)
  - Creates periodic event at set/ configurable interval
  - Used to invoke periodic kernel tasks
  - If frequency too
    - Low:           Kernel can seem unresponsive
    - High:           OS spends too much time servicing timer

---

---

---

---

---

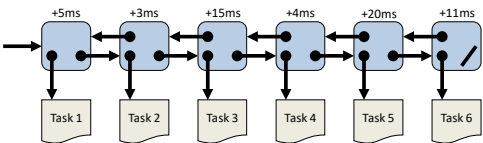
---

---

7

### Avoiding Fine Grained Timers

- Create event queue
  - Timer configured to fire when next task due, rather than at set period
  - Set interval timer to time delta (+n ms) to next event
- Technique used widely: OSs, simulators, etc.



---

---

---

---

---

---

---

8