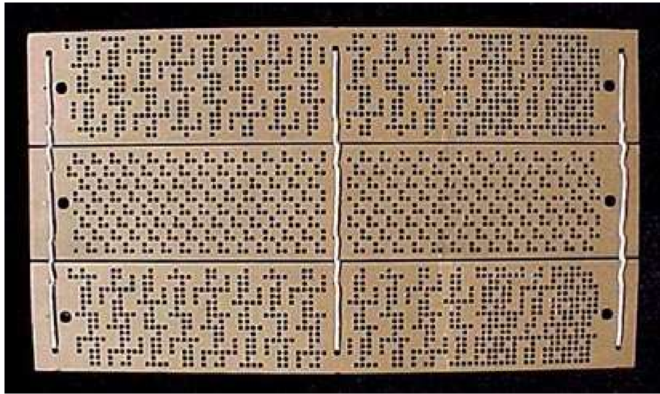
I/O Management Intro

Chapter 5

Learning Outcomes

- A high-level understanding of the properties of a variety of I/O devices.
- An understanding of methods of interacting with I/O devices.



commodore

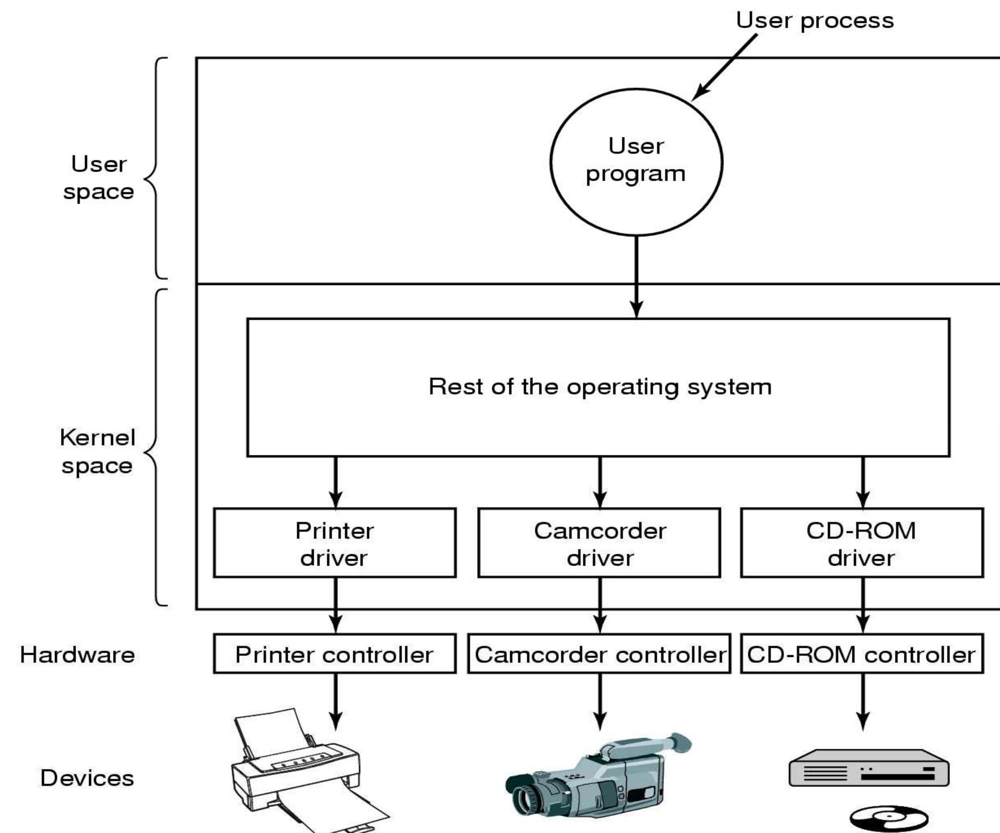
Seagate

I/O Devices

- There exists a large variety of I/O devices:
  - Many of them with different properties
  - They seem to require different interfaces to manipulate and manage them
    - We don't want a new interface for every device
    - Diverse, but similar interfaces leads to code duplication
- Challenge:
  - Uniform and efficient approach to I/O

Device Drivers

- Logical position of device drivers is shown here
- Drivers (originally) compiled into the kernel
  - Including OS/161
  - Device installers were technicians
  - Number and types of devices rarely changed
- Nowadays they are dynamically loaded when needed
  - Linux modules
  - Typical users (device installers) can't build kernels
  - Number and types vary greatly
    - Even while OS is running (e.g hot-plug USB devices)



Device Drivers

- **Drivers classified into similar categories**
  - Block devices and character (stream of data) device
- **OS defines a standard (internal) interface to the different classes of devices**
  - Example: USB HID (human interface device) class specifications
    - human input devices follow a set of rules making it easier to design a standard interface.

USB Device Classes

Base Class	Descriptor Usage	Description
00h	Device	Use class information in the Interface Descriptors
01h	Interface	Audio
02h	Both	Communications and CDC Control
03h	Interface	HID (Human Interface Device)
05h	Interface	Physical
06h	Interface	Image
07h	Interface	Printer
08h	Interface	Mass Storage
09h	Device	Hub
0Ah	Interface	CDC-Data
0Bh	Interface	Smart Card
0Dh	Interface	Content Security
0Eh	Interface	Video
0Fh	Interface	Personal Healthcare
10h	Interface	Audio/Video Devices
DCh	Both	Diagnostic Device
E0h	Interface	Wireless Controller
EFh	Both	Miscellaneous
FEh	Interface	Application Specific
FFh	Both	Vendor Specific

THE UNIVERSITY OF
NEW SOUTH WALES

I/O Device Handling

- Data rate
  - May be differences of several orders of magnitude between the data transfer rates
  - Example: Assume 1000 cycles/byte I/O
    - Keyboard needs 10 KHz processor to keep up
    - Gigabit Ethernet needs 100 GHz processor…..

Sample Data Rates

Device	Data rate
Keyboard	10 bytes/sec
Mouse	100 bytes/sec
56K modem	7 KB/sec
Telephone channel	8 KB/sec
Dual ISDN lines	16 KB/sec
Laser printer	100 KB/sec
Scanner	400 KB/sec
Classic Ethernet	1.25 MB/sec
USB (Universal Serial Bus)	1.5 MB/sec
Digital camcorder	4 MB/sec
IDE disk	5 MB/sec
40x CD-ROM	6 MB/sec
Fast Ethernet	12.5 MB/sec
ISA bus	16.7 MB/sec
EIDE (ATA-2) disk	16.7 MB/sec
FireWire (IEEE 1394)	50 MB/sec
XGA Monitor	60 MB/sec
SONET OC-12 network	78 MB/sec
SCSI Ultra 2 disk	80 MB/sec
Gigabit Ethernet	125 MB/sec
Ultrium tape	320 MB/sec
PCI bus	528 MB/sec
Sun Gigaplane XB backplane	20 GB/sec

USB 3.0 625 MB/s (5 Gb/s)
Thunderbolt 2.5GB/sec (20 Gb/s)
PCIe v3.0 x16 16GB/s

Device Drivers

- **Device drivers job**
  - translate request through the device-independent standard interface (open, close, read, write) into appropriate sequence of commands (register manipulations) for the particular hardware
  - Initialise the hardware at boot time, and shut it down cleanly at shutdown

Device Driver

- **After issuing the command to the device, the device either**
  - Completes immediately and the driver simply returns to the caller
  - Or, device must process the request and the driver usually blocks waiting for an interrupt indicating I/O completion.
- **Drivers are thread-safe** as they can be called by another process while a process is already blocked in the driver.
  - Thread-safe: Synchronised….

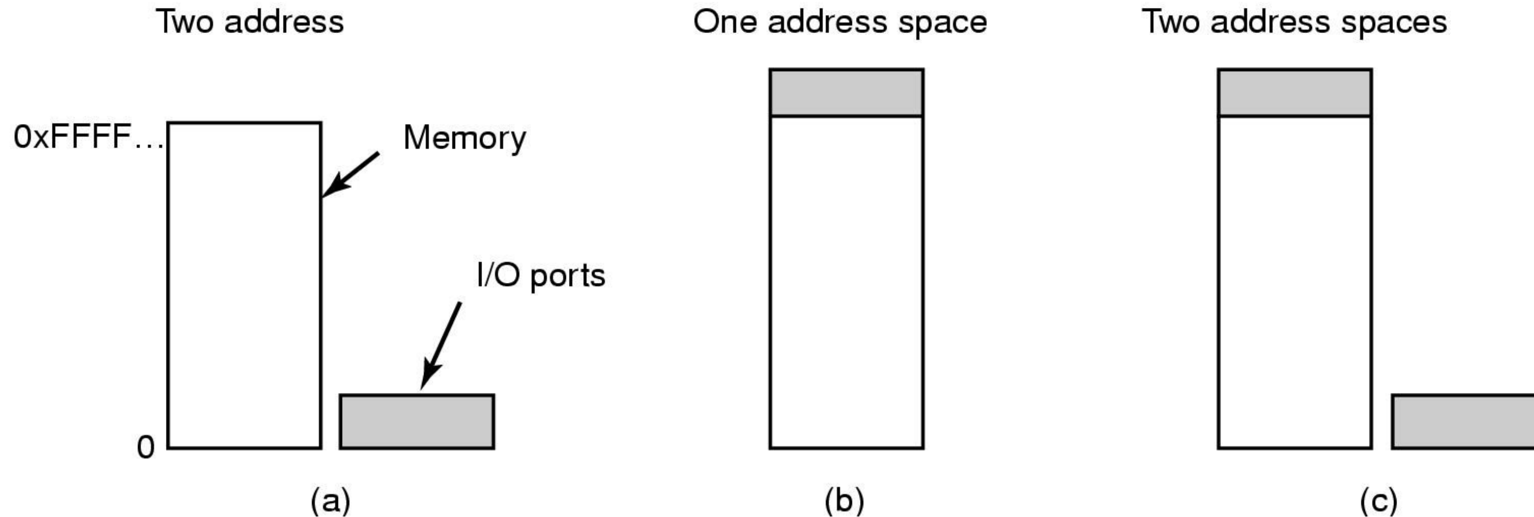
Device-Independent I/O Software

- There is commonality between drivers of similar classes
- Divide I/O software into device-dependent and device-independent I/O software
- Device independent software includes
  - Buffer or Buffer-cache management
  - TCP/IP stack
  - Managing access to dedicated devices
  - Error reporting

Driver ⇔ Kernel Interface

- Major Issue is uniform interfaces to devices and kernel
  - Uniform device interface for kernel code
    - Allows different devices to be used the same way
      - No need to rewrite file-system to switch between SCSI, IDE or RAM disk
    - Allows internal changes to device driver with fear of breaking kernel code
  - Uniform kernel interface for device code
    - Drivers use a defined interface to kernel services (e.g. kmalloc, install IRQ handler, etc.)
    - Allows kernel to evolve without breaking existing drivers
  - Together both uniform interfaces avoid a lot of programming implementing new interfaces
    - Retains compatibility as drivers and kernels change over time.

Accessing I/O Controllers



a) Separate I/O and memory space
- I/O controller registers appear as I/O ports
- Accessed with special I/O instructions

b) Memory-mapped I/O
- Controller registers appear as memory
- Use normal load/store instructions to access

c) Hybrid
- x86 has both ports and memory mapped I/O

Bus Architectures

CPU reads and writes of memory go over this high-bandwidth bus

CPU

Memory

I/O

All addresses (memory and I/O) go here

Bus

(a)

CPU

Memory

I/O

This memory port is to allow I/O devices access to memory

(b)

(a) A single-bus architecture

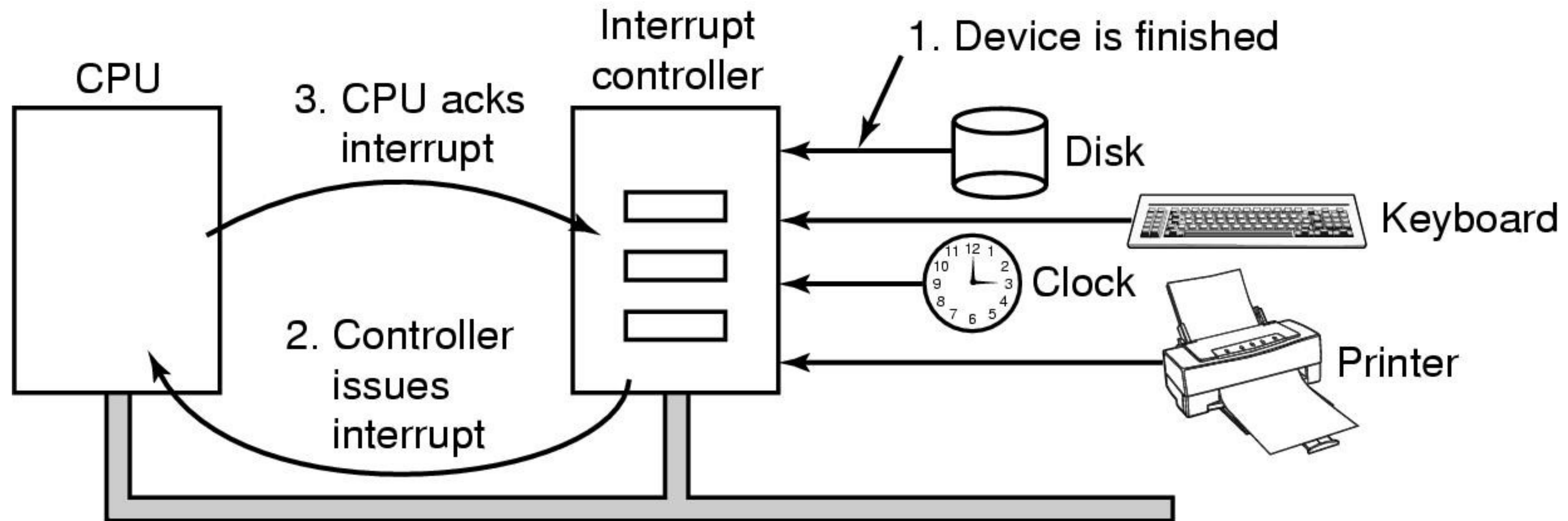
(b) A dual-bus memory architecture

Intel IXP420

MII-0
MII-1
Ethernet NPE A Ethernet MAC
Ethernet NPE B Ethernet MAC
133.32 MHz x 32 bits North Advance High-Performance Bus
Queue Status Bus
North AHB Arbiter
Queue Manager 8 KB SRAM
SDRAM Controller 8 - 256 MB
32-Bit
UART 921Kbaud
Interrupt Controller
Timers
North/South AHB Bridge
South AHB Arbiter
66.66 MHz Advanced Peripheral Bus
AHB/APB Bridge
133.32 MHz x 32 bits South Advance High-Performance Bus
UART 921Kbaud
GPIO Controller
USB Device V1.1
PMU (AHB)
Intel XScale® Core
266/400/533 MHz
32 KB Data Cache
32 KB Instruction Cache
2 KB Mini-Data Cache
PCI Controller
Expansion Bus Controller
Test Logic Unit
16 GPIO
JTAG
32-Bit
16-Bit

THE UNIVERSITY OF NEW SOUTH WALES

Interrupts

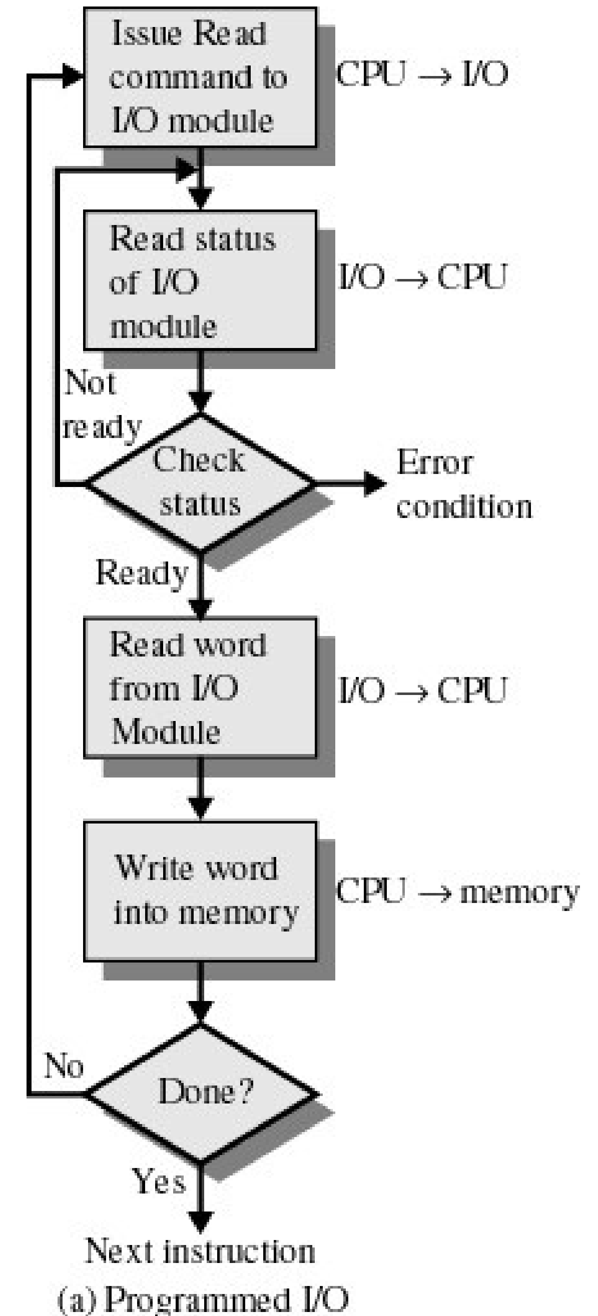


- Devices connected to an *Interrupt Controller* via lines on an I/O bus (e.g. PCI)
- Interrupt Controller signals interrupt to CPU and is eventually acknowledged.
- Exact details are architecture specific.

I/O Interaction

Programmed I/O

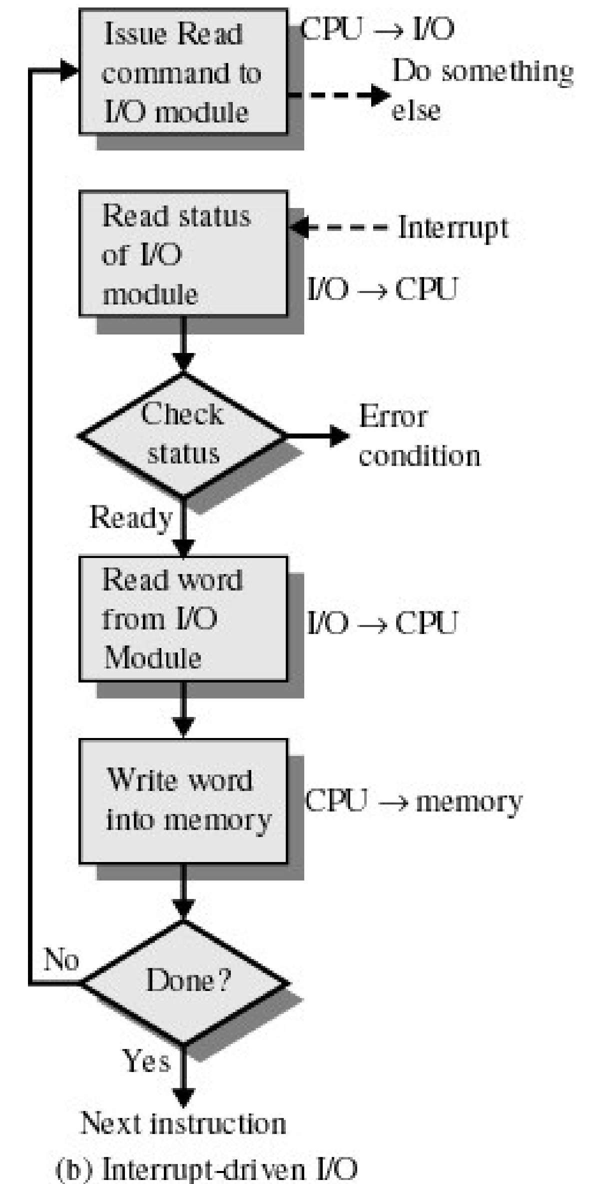
- Also called *polling*, or *busy waiting*
- I/O module (controller) performs the action, not the processor
- Sets appropriate bits in the I/O status register
- No interrupts occur
- Processor checks status until operation is complete
  - Wastes CPU cycles



(a) Programmed I/O

Interrupt-Driven I/O

- Processor is interrupted when I/O module (controller) ready to exchange data
- Processor is free to do other work
- No needless waiting
- Consumes a lot of processor time because every word read or written passes through the processor



(b) Interrupt-driven I/O

Direct Memory Access

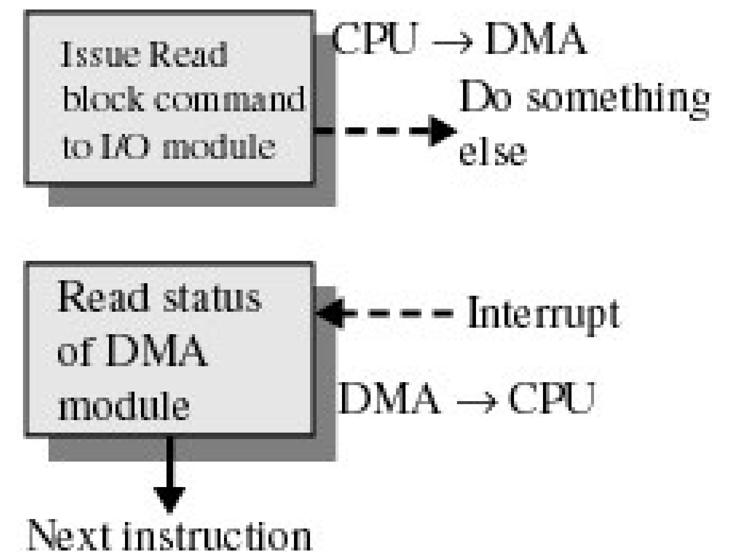
- Transfers data directly between Memory and Device
- CPU not needed for copying

DMA Controller in Device

Separate DMA Controller

Direct Memory Access

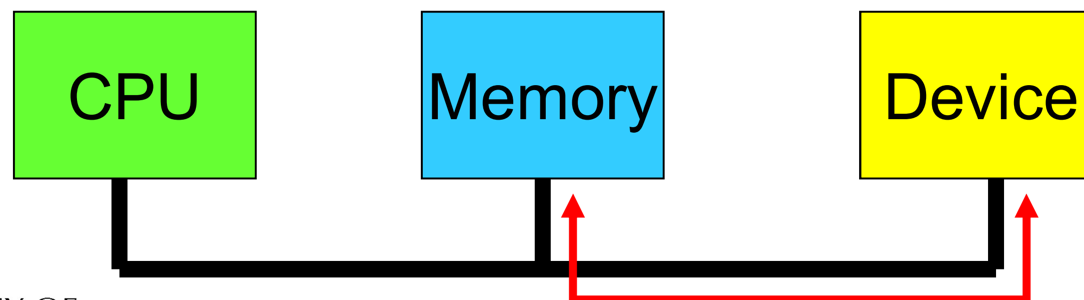
- Transfers a block of data directly to or from memory
- An interrupt is sent when the task is complete
- The processor is only involved at the beginning and end of the transfer



(c) Direct memory access

DMA Considerations

- ✓ Reduces number of interrupts
  - Less (expensive) context switches or kernel entry-exits
- ✗ Requires contiguous regions (buffers)
  - Copying
  - Some hardware supports “Scatter-gather”
- Synchronous/Asynchronous
- Shared bus must be arbitrated (hardware)
  - CPU cache reduces (but not eliminates) CPU need for bus



The Process to Perform DMA Transfer

1. device driver is told to transfer disk data to buffer at address X
2. device driver tells disk controller to transfer C bytes from disk to buffer at address X
3. disk controller initiates DMA transfer
4. disk controller sends each byte to DMA controller
5. DMA controller transfers bytes to buffer X, increasing memory address and decreasing C until C = 0
6. when C = 0, DMA interrupts CPU to signal transfer completion

CPU

cache

CPU memory bus

memory

x

buffer

DMA/bus/interrupt controller

PCI bus

IDE disk controller

disk

disk

disk

disk