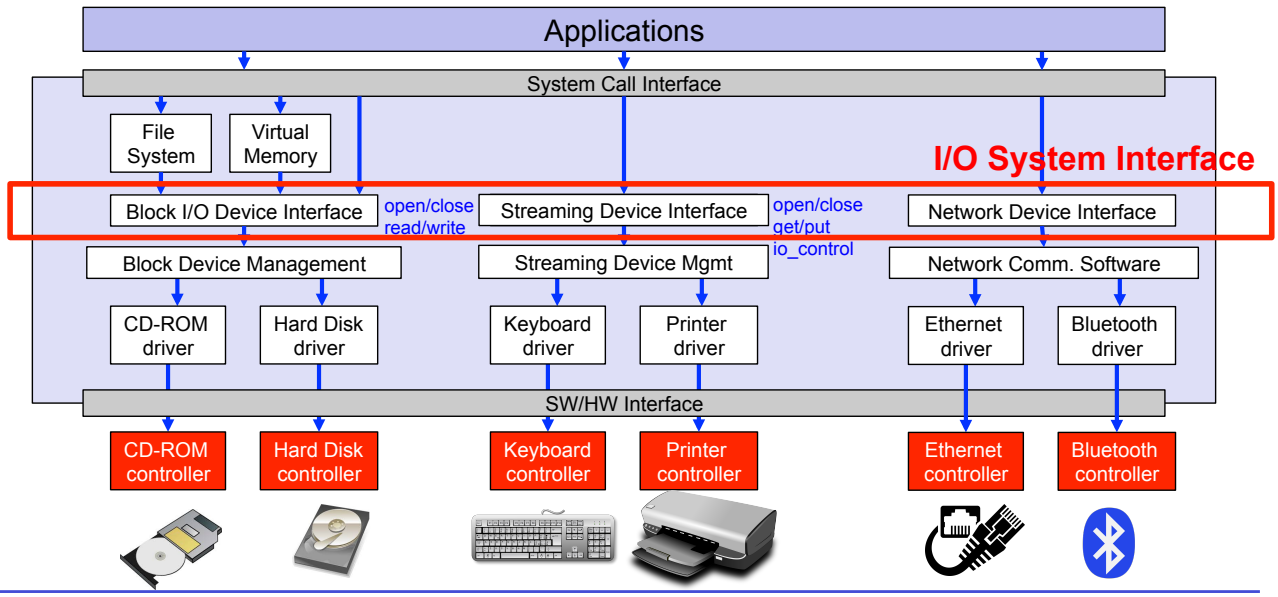
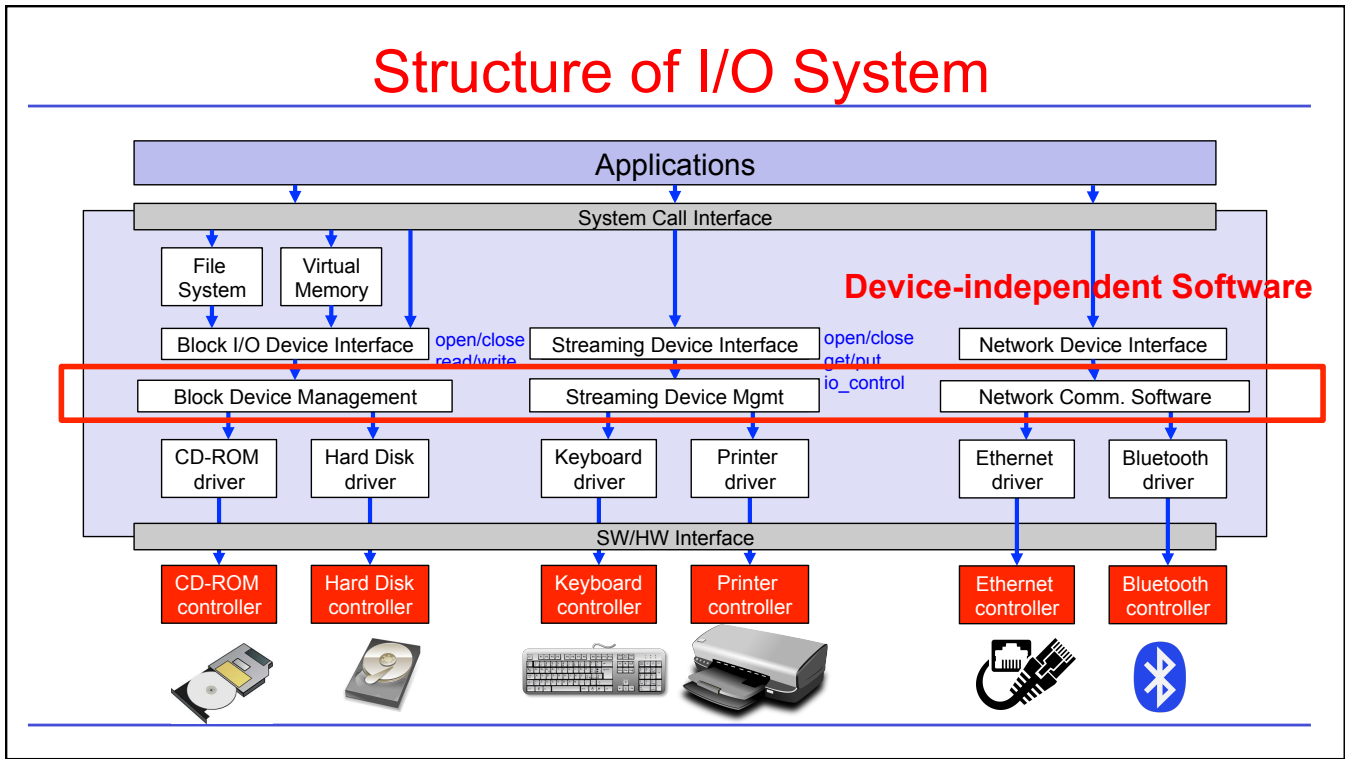
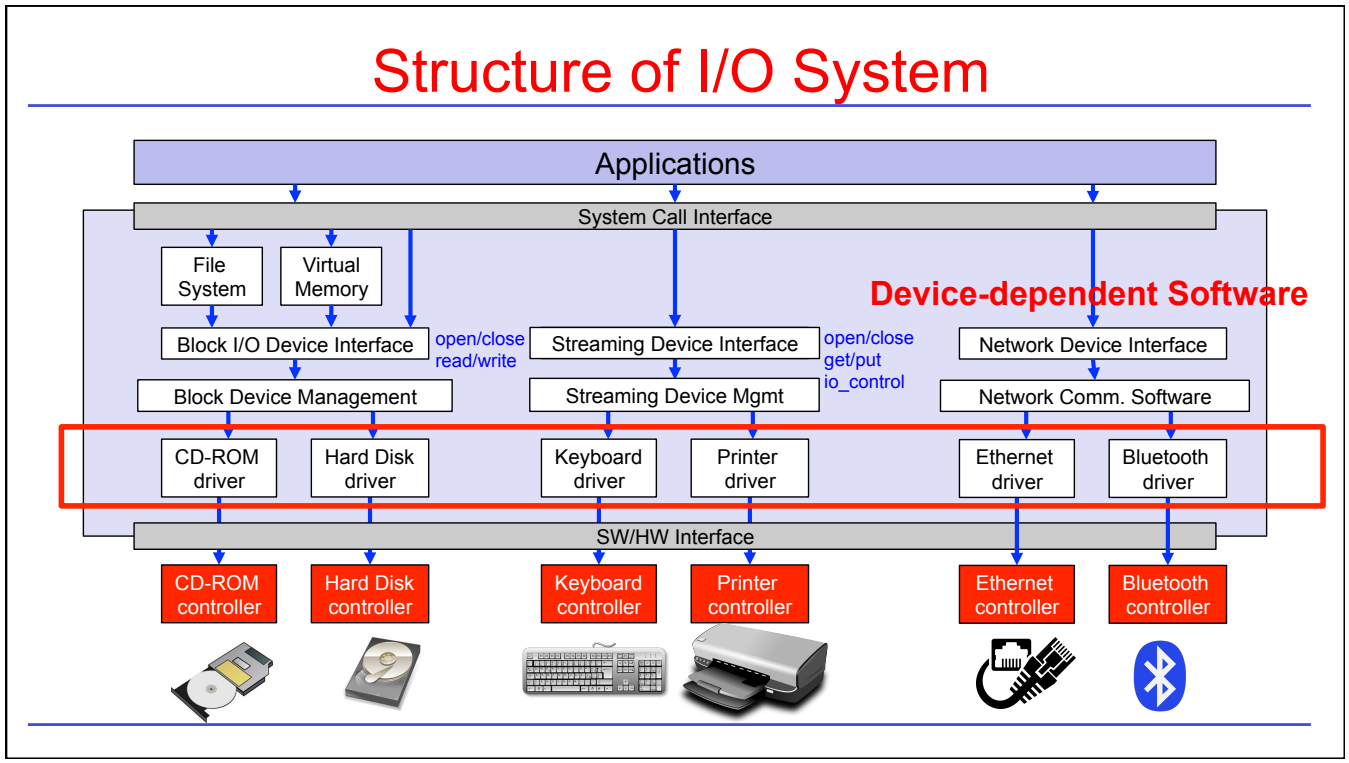
I/O Systems

- Structure of I/O System
- Device Controllers vs. Device Drivers
- Programmed I/O, Polling, Interrupts
- Direct Memory Access (DMA)
- Modern I/O Architectures
- Structure of Device Drivers

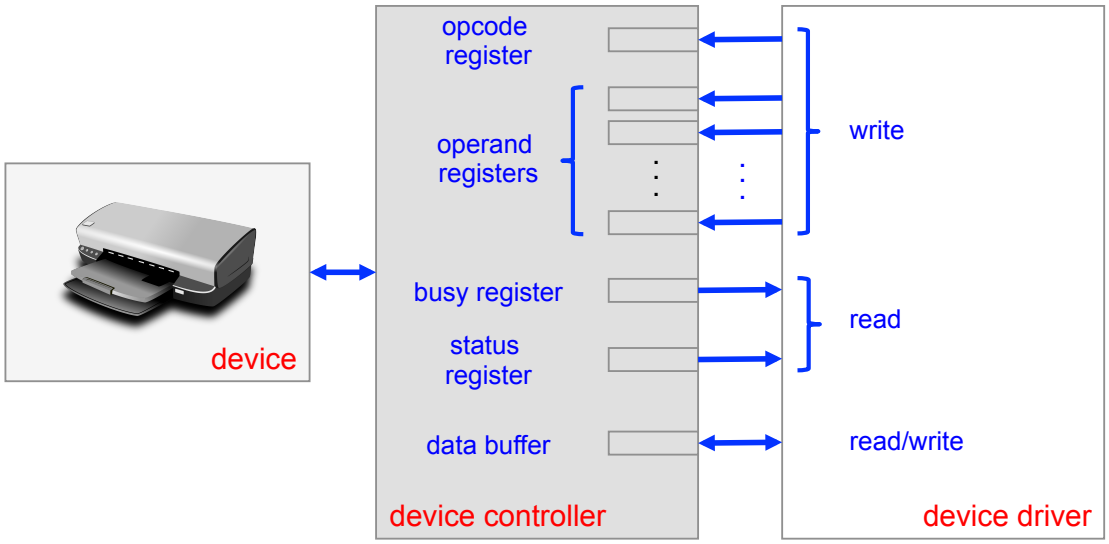
Structure of I/O System



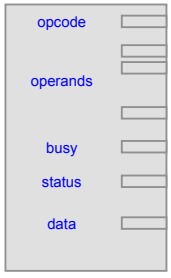

Structure of I/O System
Applications
System Call Interface
File System
Virtual Memory
Device-independent Software
Block I/O Device Interface
open/close
read/write
Streaming Device Interface
open/close
get/put
io_control
Network Device Interface
Block Device Management
Streaming Device Mgmt
Network Comm. Software
CD-ROM driver
Hard Disk driver
Keyboard driver
Printer driver
Ethernet driver
Bluetooth driver
SW/HW Interface
CD-ROM controller
Hard Disk controller
Keyboard controller
Printer controller
Ethernet controller
Bluetooth controller


Structure of I/O System
Applications
System Call Interface
File System
Virtual Memory
Device-dependent Software
Block I/O Device Interface
open/close
read/write
Streaming Device Interface
open/close
get/put
io_control
Network Device Interface
Block Device Management
Streaming Device Mgmt
Network Comm. Software
CD-ROM driver
Hard Disk driver
Keyboard driver
Printer driver
Ethernet driver
Bluetooth driver
SW/HW Interface
CD-ROM controller
Hard Disk controller
Keyboard controller
Printer controller
Ethernet controller
Bluetooth controller

Device Controllers



Device Controllers



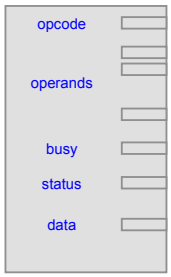
Three Questions:

Question 1: How do we access device registers?

Question 2: After an operation has been issued, and controller is busy, how do we know when controller is done?

Question 3: How is the data moved between controller data buffer and main memory?

Device Controllers



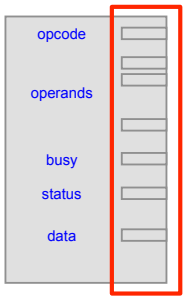
Three Questions:

Question 1: How do we access device registers?

Question 2: After an operation has been issued, and controller is busy, how do we know when controller is done?

Question 3: How is the data moved between controller data buffer and main memory?

Explicit vs. Memory-Mapped Device Interfaces



Explicit Device Interface:

```
io_store cpu_reg, dev_no, dev_reg
```

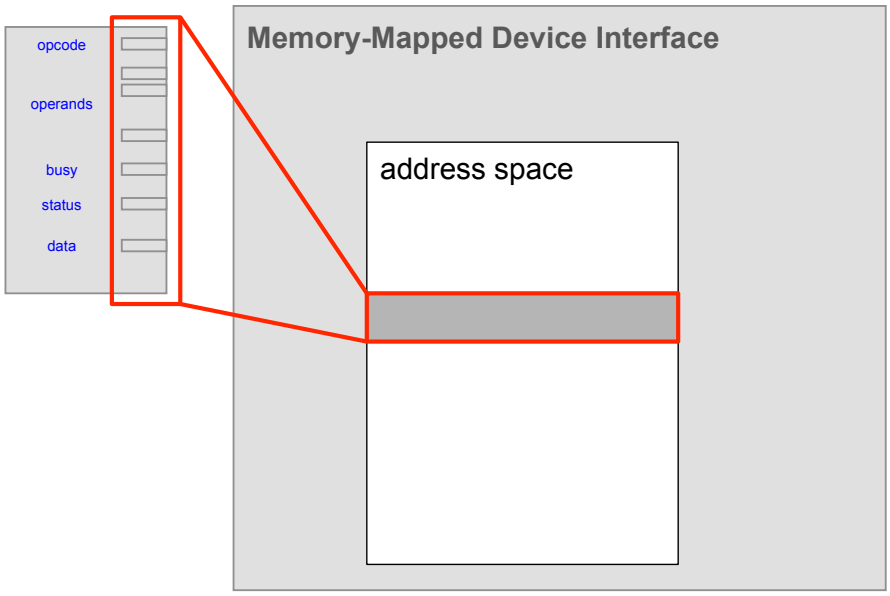
On x86:

```
out port, value
```

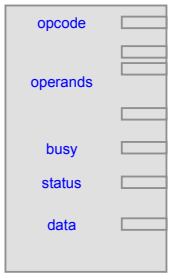
Example – Write to disk (vanilla!):

```
; position head
outb 0x1F1, 0x00
outb 0x1F2, 0x01
outb 0x1F3, 'block_no'          ; only low 8 bit
outb 0x1F4, 'block_no >> 8'     ; next 8 bit
outb 0x1F5, 'block_no >> 16'    ; next 8 bit
outb 0x1F6, '(block_no >> 24)&0x0F | 0xE0|(disk_id << 4)'
              ; ones, drive indicator, highest 4 bits of block_no
outb 0x1F7, 0x30                ; 0x30 for WRITE, 0x20 for READ
; start writing
outw 0x1F0, '16-bit data'       ; ...
```

Explicit vs. Memory-Mapped Device Interfaces



Device Controllers



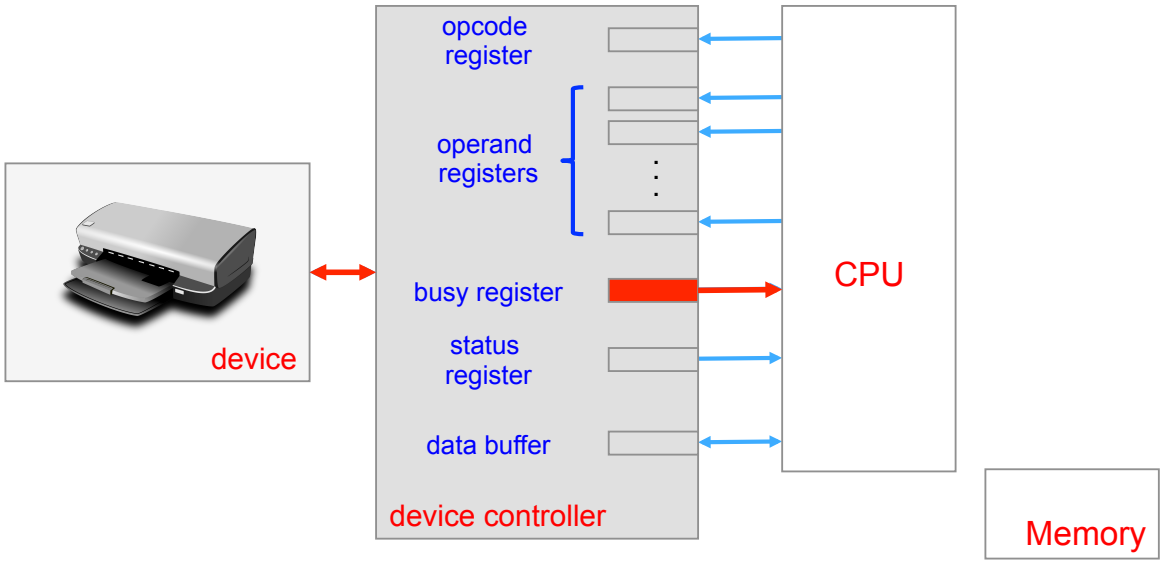
Three Questions:

Question 1: How do we access device registers?

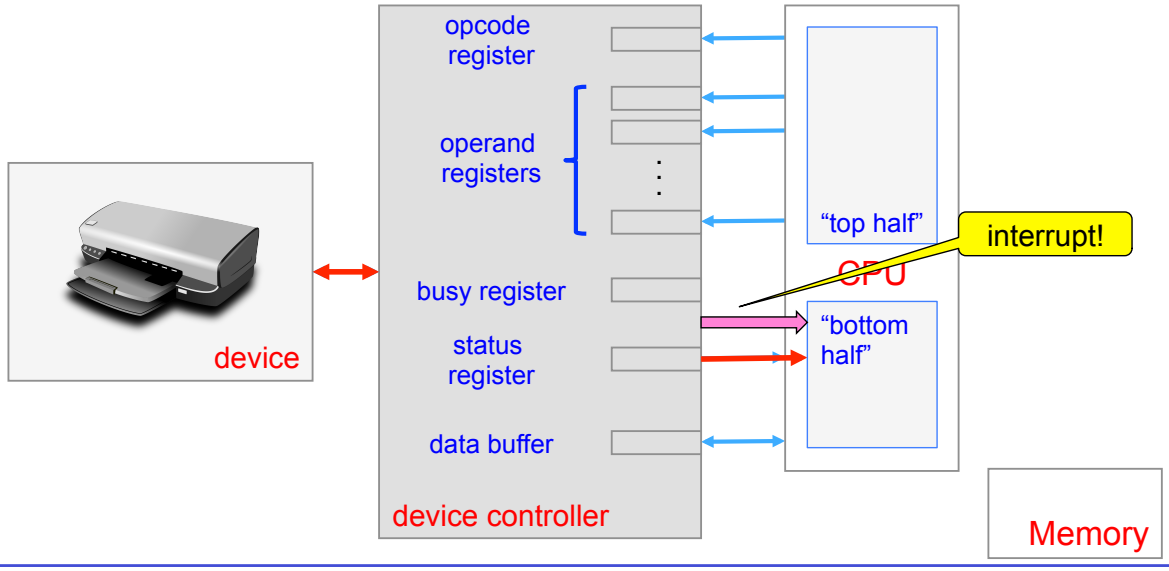
Question 2: After an operation has been issued, and controller is busy, how do we know when controller is done?

Question 3: How is the data moved between controller data buffer and main memory?

Programmed I/O with Polling



Programmed I/O with Interrupts



Device Controllers

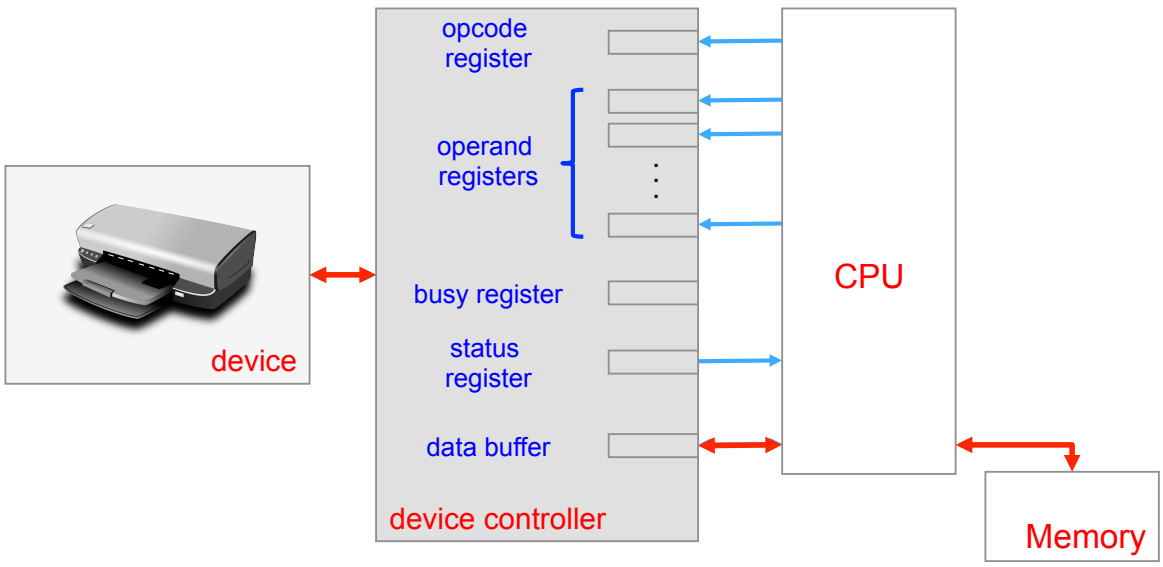
Three Questions:

Question 1: How do we access device registers?

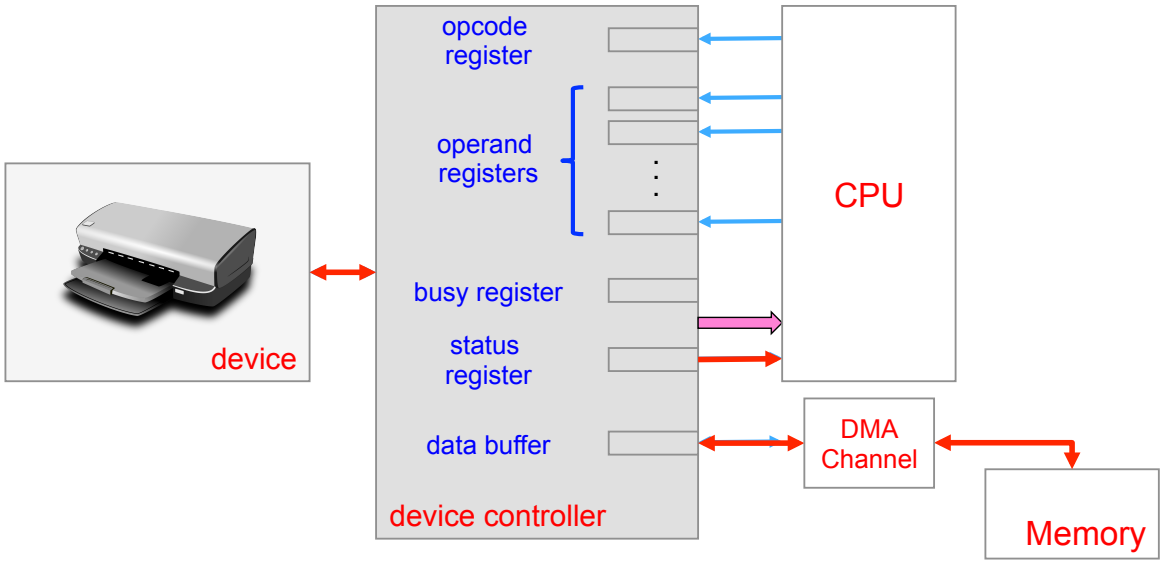
Question 2: After an operation has been issued, and controller is busy, how do we know when controller is done?

Question 3: How is the data moved between controller data buffer and main memory?

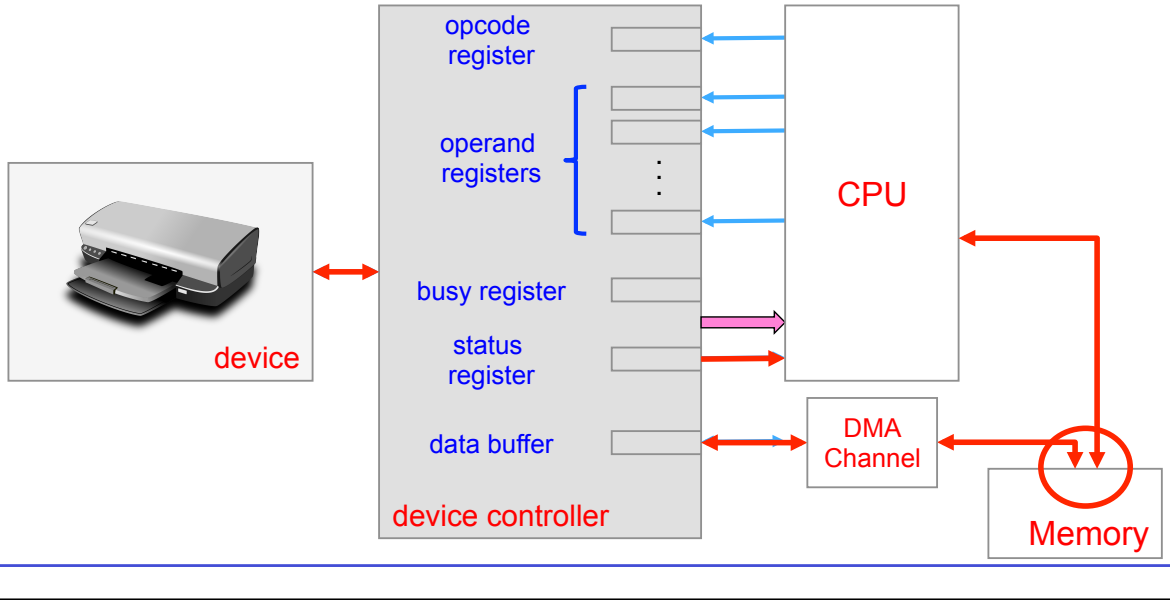
Data Transfer: Programmed I/O



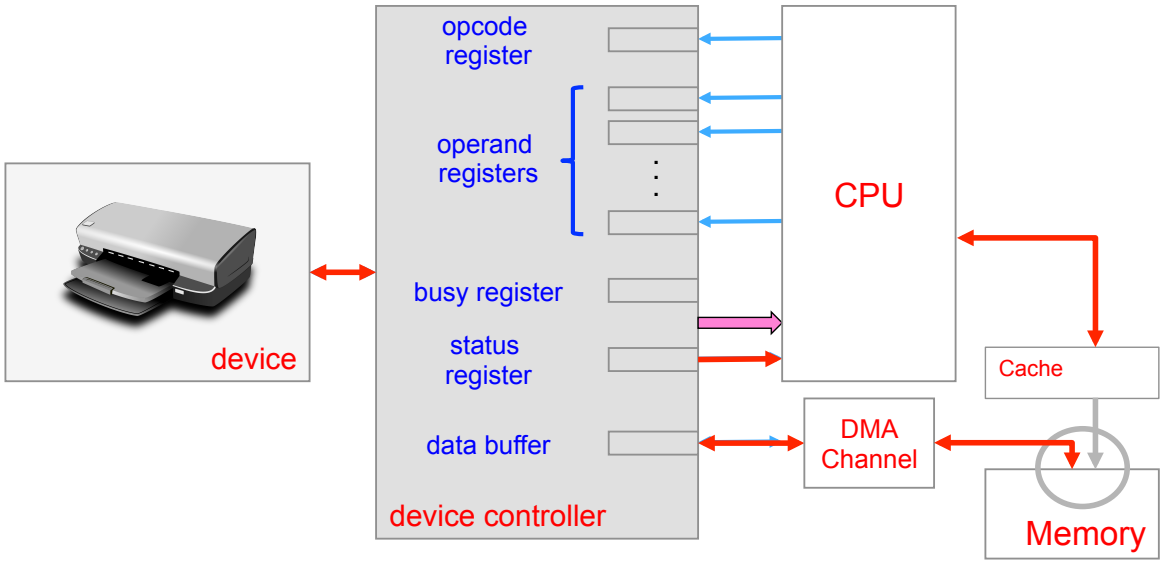
Data Transfer: Direct Memory Access (DMA)



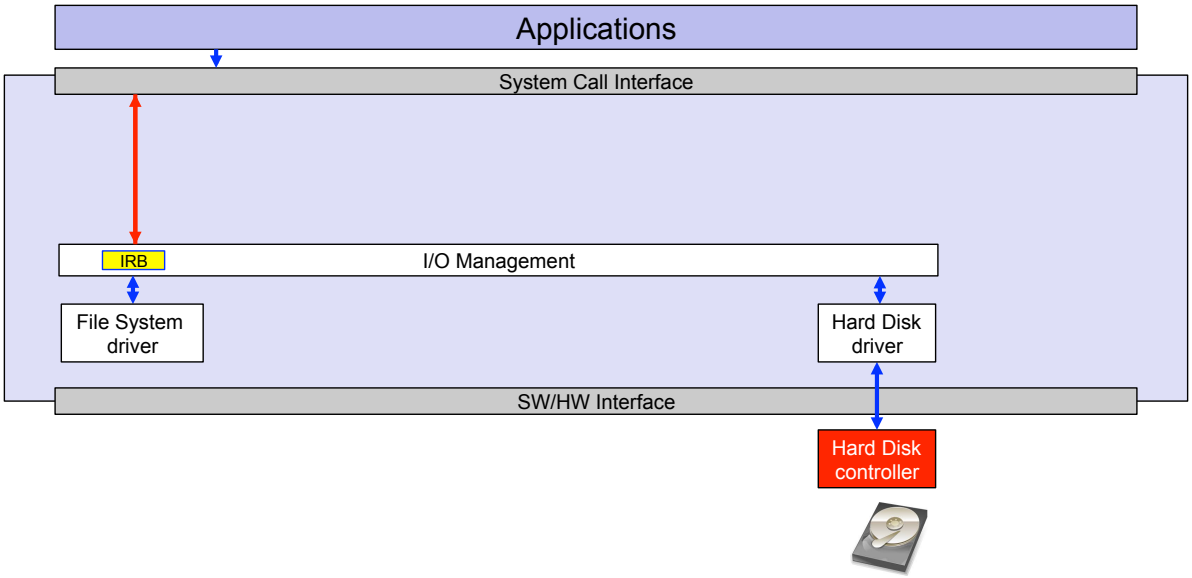
Data Transfer: Direct Memory Access (DMA)



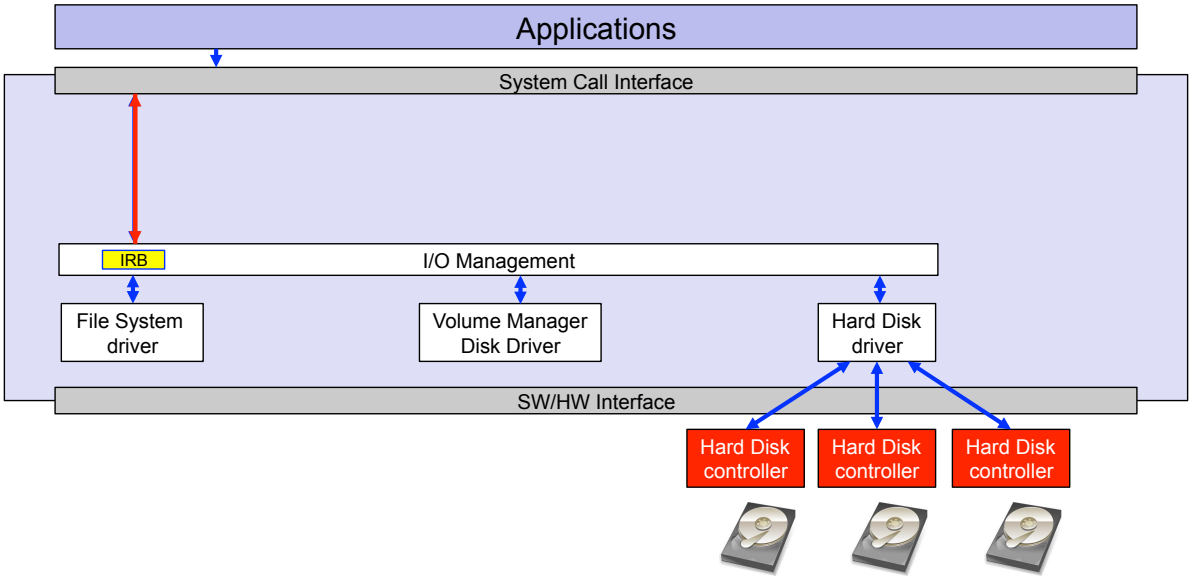
Data Transfer: Direct Memory Access (DMA)



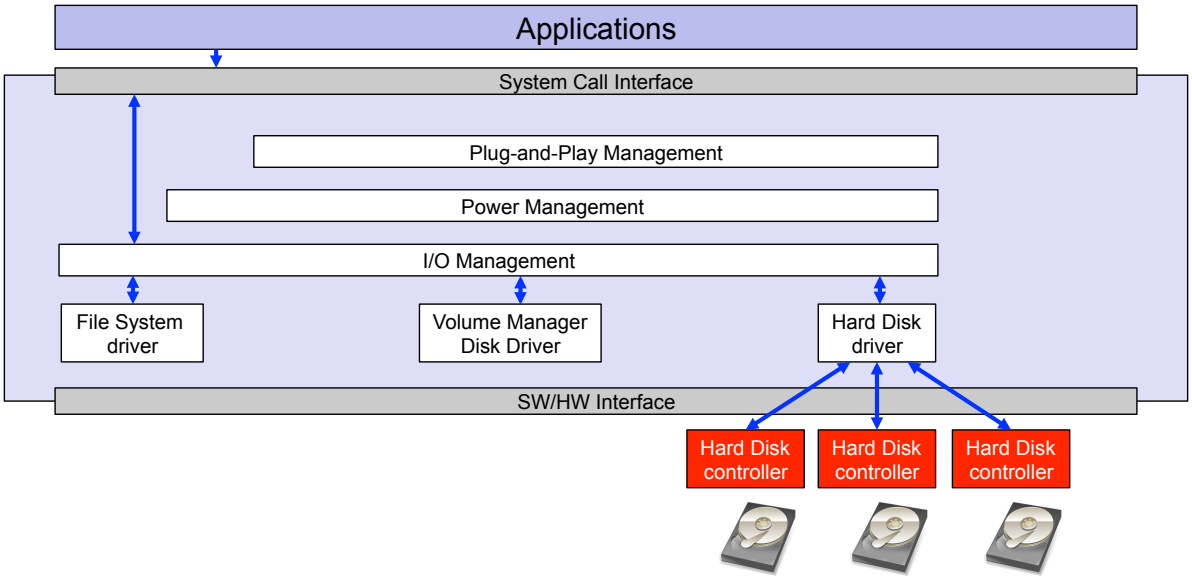
Modern Driver Architectures



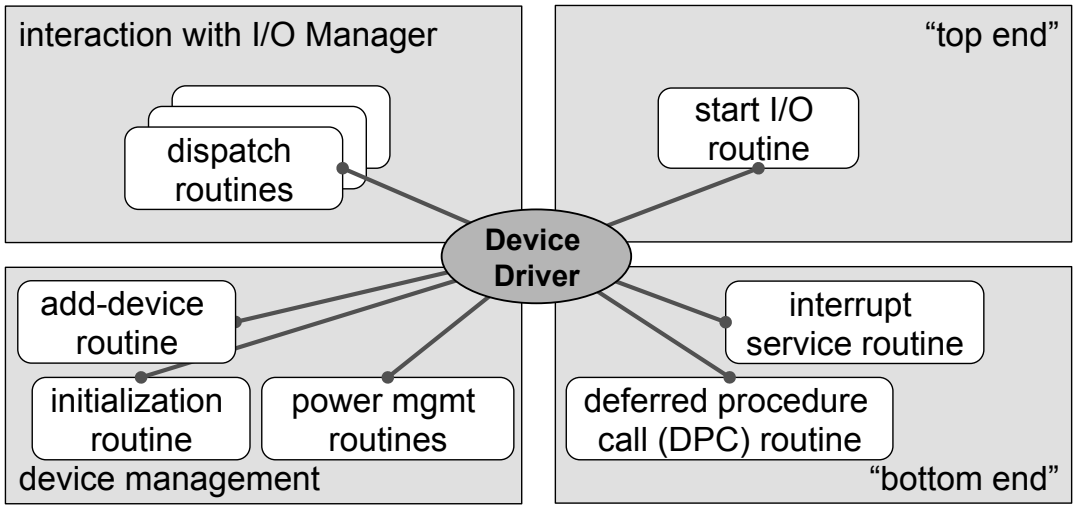
Modern Driver Architectures



Modern Driver Architectures



Generic Device Driver Structure



I/O Systems

- Structure of I/O System
- Device Controllers vs. Device Drivers
- Programmed I/O, Polling, Interrupts
- Direct Memory Access (DMA)
- Modern I/O Architectures
- Structure of Device Drivers