Unit 6: Device Management

6.1. Principles of I/O Systems

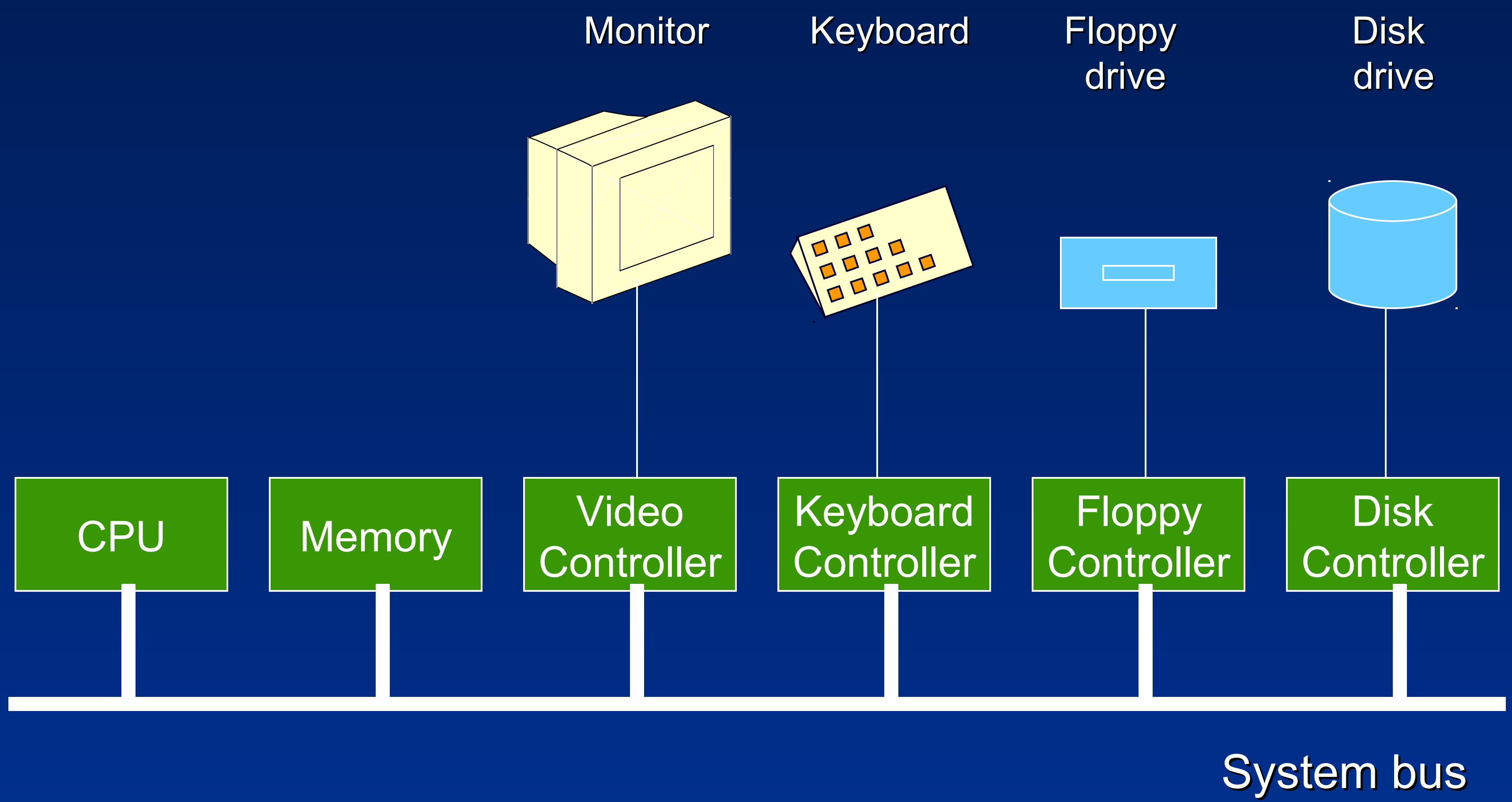
Roadmap for Section 6.1

- Principles of I/O Hardware
- Structuring of I/O Software
- Layers of an I/O System
- Operation of an I/O System

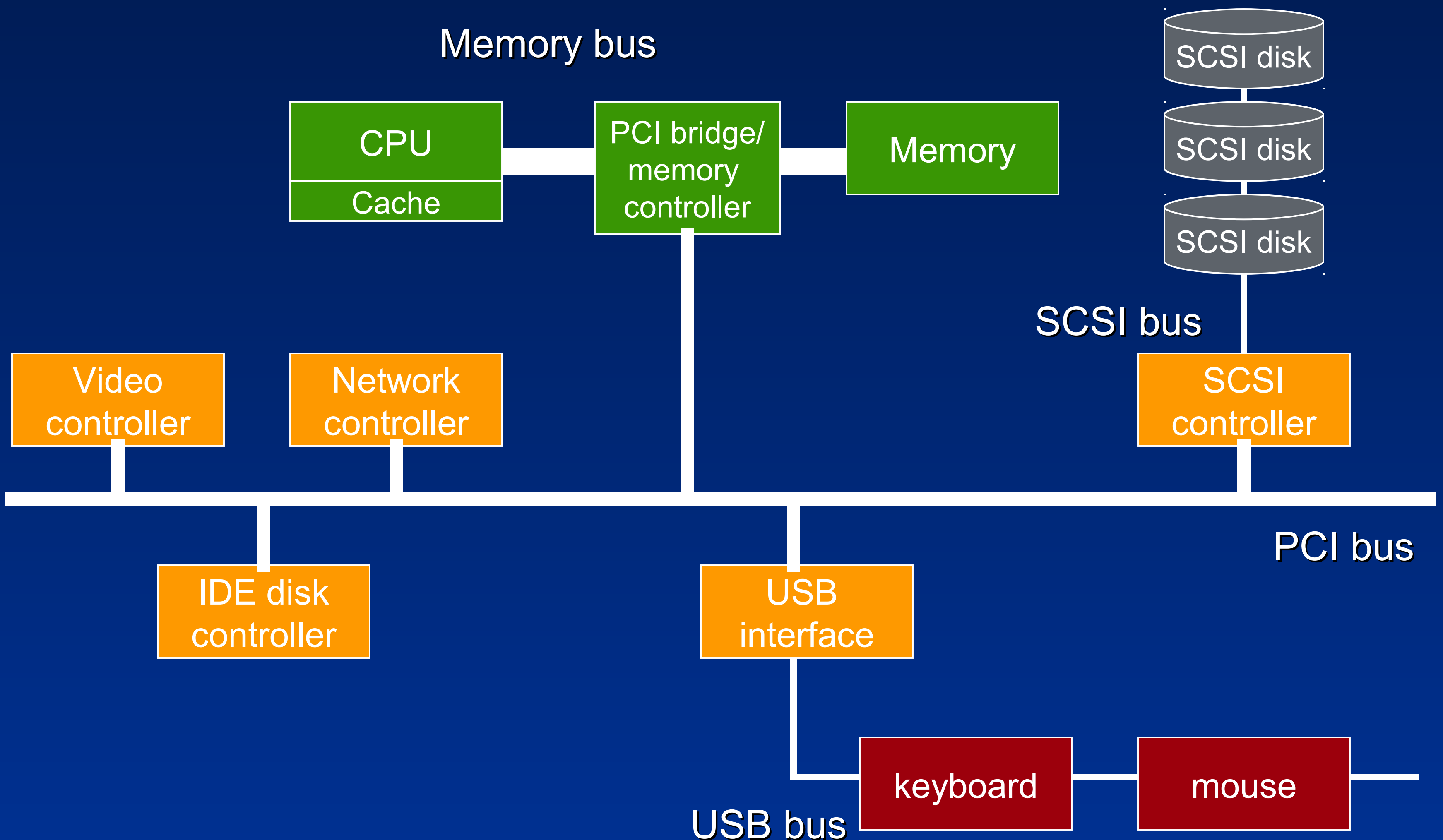
Input/Output – Principles of I/O Hardware

- Major components of a computer system:
  CPU, memories (primary/secondary), I/O system
- I/O devices:
  - Block devices – store information in fixed-sized blocks; typical sizes: 128-4096 bytes
  - Character devices – delivers/accepts stream of characters
- Device controllers:
  - Connects physical device to system bus (Minicomputers, PCs)
  - Mainframes use a more complex model:
    Multiple buses and specialized I/O computers (I/O channels)
- Communication:
  - Memory-mapped I/O, controller registers
  - Direct Memory Access (DMA)

I/O Hardware - Single Bus



I/O Hardware - Multiple Buses



Diversity among I/O Devices

The I/O subsystem has to consider device characteristics:

- Data rate:
  - may vary by several orders of magnitude
- Complexity of control:
  - exclusive vs. shared devices
- Unit of transfer:
  - stream of bytes vs. block-I/O
- Data representations:
  - character encoding, error codes, parity conventions
- Error conditions:
  - consequences, range of responses
- Applications:
  - impact on resource scheduling, buffering schemes

Organization of the I/O Function

- Programmed I/O with polling:
  - The processor issues an I/O command on behalf of a process
  - The process busy waits for completion of the operation before proceeding
- Interrupt-driven I/O:
  - The processor issues an I/O command and continues to execute
  - The I/O module interrupts the processor when it has finished I/O
  - The initiator process may be suspended pending the interrupt
- Direct memory access (DMA):
  - A DMA module controls exchange of data between I/O module and main memory
  - The processor requests transfer of a block of data from DMA and is interrupted only after the entire block has been transferred

Flow of a blocking I/O request

1. Thread issues blocking read() system call
2. Kernel checks parameters; may return buffered data and finish
3. Thread is removed from run queue if physical I/O required; added to wait queue for device; I/O request is scheduled
4. Device driver allocates kernel buffer; sends command to controller
5. Device controller operates the hardware to perform data transfer
6. Driver may poll for status and data; or set up DMA that will generate interrupt
7. Interrupt occurs; handler stores data; signals device driver
8. Device driver receives signal; determines request status; signals kernel I/O subsystem
9. Kernel transfers data or return code to user space; removes thread from wait queue
10. Thread resumes execution at completion of read() call

Principles of I/O Software

- Layered organization
- Device independence
- Error handling
  - Error should be handled as close to the hardware as possible
  - Transparent error recovery at low level
- Synchronous vs. Asynchronous transfers
  - Most physical I/O is asynchronous
  - Kernel may provide synchronous I/O system calls
- Sharable vs. dedicated devices
  - Disk vs. printer

Structuring of I/O software

1. User-level software
2. Device-independent OS software
3. Device drivers
4. Interrupt handlers

Interrupt Handlers

- Should be hidden by the operating system
- Every thread starting an I/O operation should block until I/O has completed and interrupt occurs
- Interrupt handler transfers data from device (controller) and un-blocks process

Device Driver

- Contains all device-dependent code
- Handles one device
- Translates abstract requests into device commands
  - Writes controller registers
  - Accesses mapped memory
  - Queues requests
- Driver may block after issuing a request:
  - Interrupt will un-block driver (returning status information)

Device-independent I/O Software

Functions of device-independent I/O software:

- Uniform interfacing for the device drivers
- Device naming
- Device protection
- Providing a device-independent block size
- Buffering
- Storage allocation on block devices
- Allocating and releasing dedicated devices
- Error reporting

Layers of the I/O System

User-Space I/O Software

- System call libraries (read, write,...)
- Spooling
  - Managing dedicated I/O devices in a multiprogramming system
  - Daemon process, spooling directory
  - lpd – line printer daemon, sendmail – simple mail transfer protocol

I/O request

I/O reply

Layer	I/O functions
User process	I/O calls, spooling format I/O
Device-independent software	Naming, protection buffering, blocking
Device drivers	Setup registers, Check status
Interrupt handlers	Wakeup driver
Hardware	Perform I/O op.

Application I/O Interfaces

The OS system call interface distinguished device classes:

- Character-stream or block
- Sequential or random-access
- Synchronous or asynchronous
- Sharable or dedicated
- Speed of operation
- Read/write, read only, write only

Example:
4.3 BSD kernel I/O structure

<table>
<tr><td colspan="6">System-call interface to the kernel</td></tr>
<tr><td>socket</td><td>Plain file</td><td rowspan="2">Cooked block interface</td><td rowspan="2">Raw block interface</td><td rowspan="2">Raw tty interface</td><td>cooked TTY</td></tr>
<tr><td>protocols</td><td>File system</td><td>Line discipline</td></tr>
<tr><td>Network interface</td><td colspan="3">Block-device driver</td><td colspan="2">character-device driver</td></tr>
<tr><td colspan="6">The hardware</td></tr>
</table>

Further Reading

- Abraham Silberschatz, Peter B. Galvin, and Greg Gagne, "*Operating System Concepts*", John Wiley & Sons, 9th Ed., 2013.
  - Chapter 2 – Operating-System Structures
  - Chapter 13 – I/O Systems