21 I/O Management (1)

I/O Design, I/O Subsystem, I/O-Handler

Device Driver, Buffering, Disks, RAID

January 26 2009

WT 2008/09

Recommended Reading

- Tanenbaum, A.: Modern Operating Systems 2nd (5)
- Stallings, W.: Operating Systems 5th (11)
- Silberschatz, A.: Operating System Concepts 7th (13)
- Nutt, G.: Operating Systems 2nd (5)
- Bacon, J.: Operating Systems (3)
- Nehmer, J.: Systemsoftware: Grundlagen moderner Betriebssysteme, (10)

Roadmap

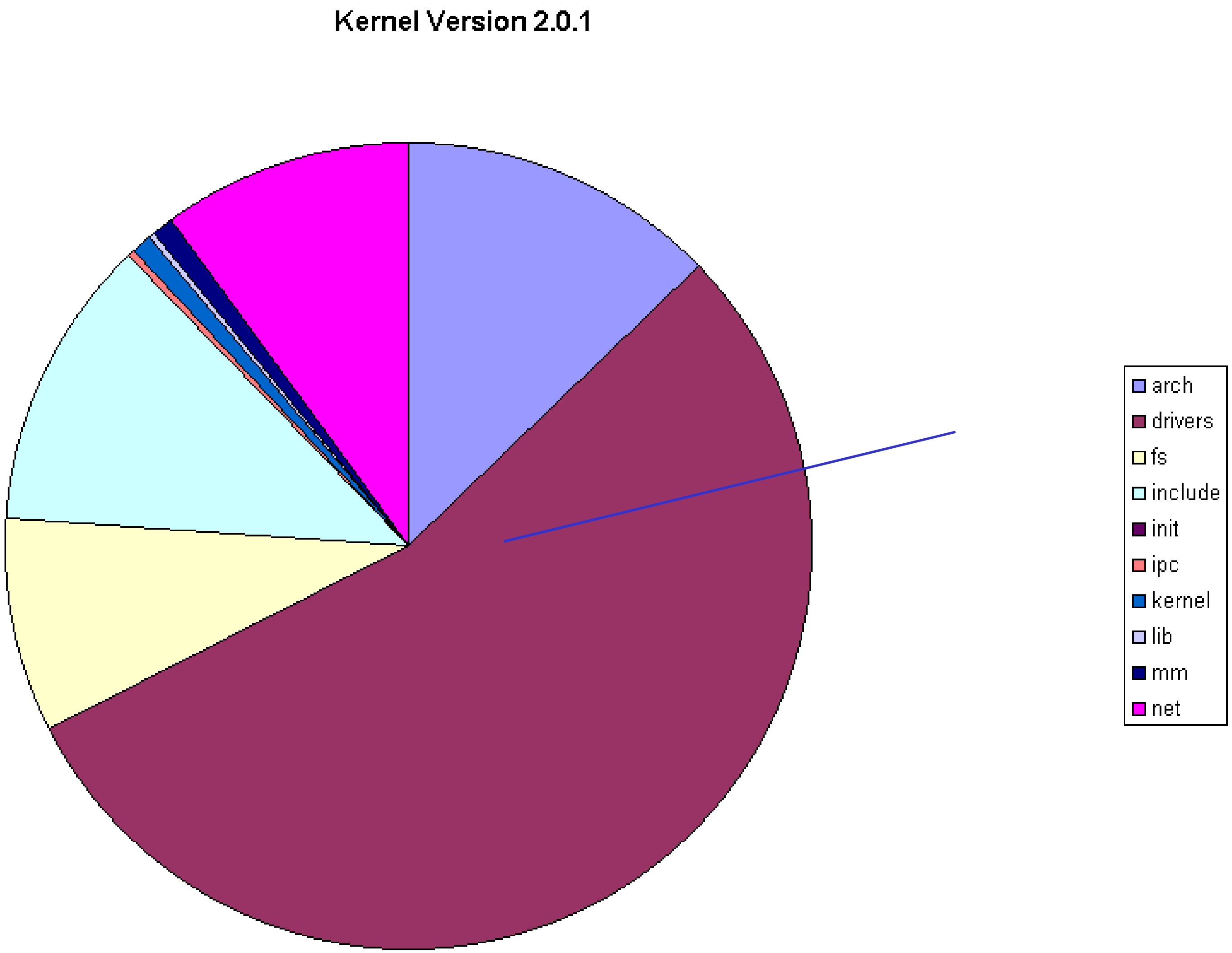
- Motivation
- Repetition: I/O-Devices
  - Device Categories
  - I/O-Functionality
  - Data Transfer
- I/O-Subsystem
  - Design Parameters
  - I/O Layering
  - I/O-Buffering
- Disk I/O Management
  - Disk, CD-Rom, ...
  - Disk Layouts and Formats
  - Disk Scheduling
  - RAID
  - Disk Caching
- Clocks and Timer

Problems of I/O Management

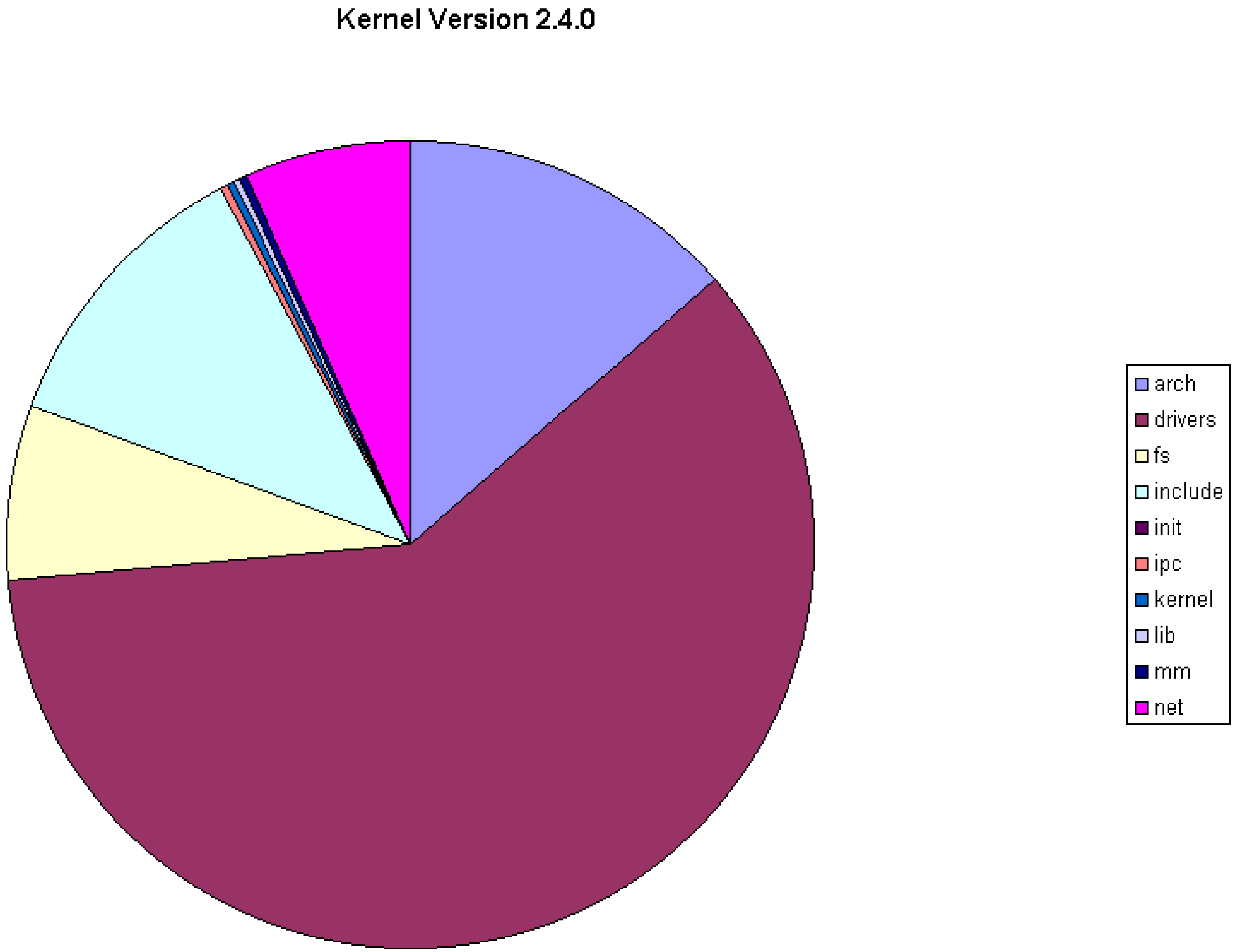
- There are many various types of I/O devices
  - Applications don‘t want to care about device specifics
  - Device independent I/O subsystems, e.g.
    - the file system or
    - the network stack

    do not want to care about device specifics
  - Most device management software will not be developed by OS suppliers, but by device vendors
  - I/O speed can’t keep up with CPU speed
  - On most computers, there is parallelism between I/O & CPU

Linux 2.0 Kernel SLOCs, I/O Portion



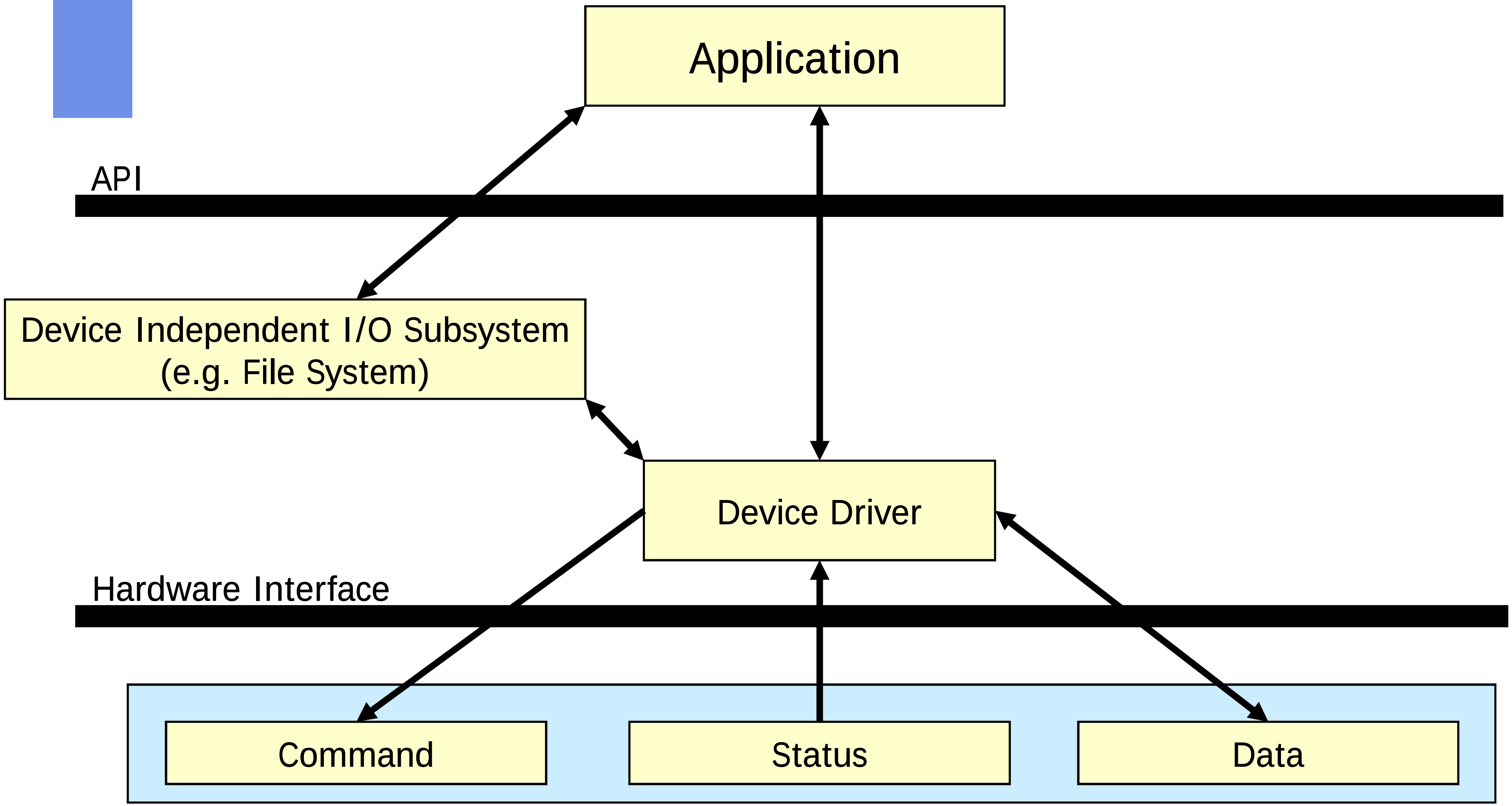
Linux 2.4 Kernel SLOCs, I/O Portion



Device Management Objectives

- Abstraction from details of physical devices
- Serialization of I/O-operations by concurrent applications
- Protection of standard-devices against unauthorized accesses
- Handling of sporadic device errors
- Virtualizing physical devices via memory and time multiplexing (e.g., pty, RAM disk)

I/O System Organization



Repetition: I/O-Devices

Categories of I/O Devices (Stallings: Operating Systems 11.1):

- “Noticeable” directly by humans
- Machine readable
- Communication devices

“Noticeable” by Humans

- Used to “communicate” with user
  - Video display terminals
  - Keyboard
  - Mouse
  - Printer
  - Headphone
  - Microphone
  - Force-feedback joystick
  - Data-glove
  - ...??

visual

audio

tactile

??

Machine Readable

- Used to communicate with local devices
  - Disk drives
  - Tape drives
  - Controllers (SCSI, CardBus, FC)
  - Actuators
  - Sensors
  - ...

Communication Devices

- Used to communicate with remote devices
  - Network adapters
  - Modems
  - ...

Characteristics of I/O Devices

aspect	variation	example
data-transfer mode	character block	terminal disk
access method	sequential random	modem CD-ROM
transfer schedule	synchronous asynchronous	tape keyboard
sharing	dedicated sharable	tape keyboard
device speed	latency seek time transfer rate delay between operations	next slide
I/O direction	read only write only read–write	CD-ROM graphics controller disk

Data Rates

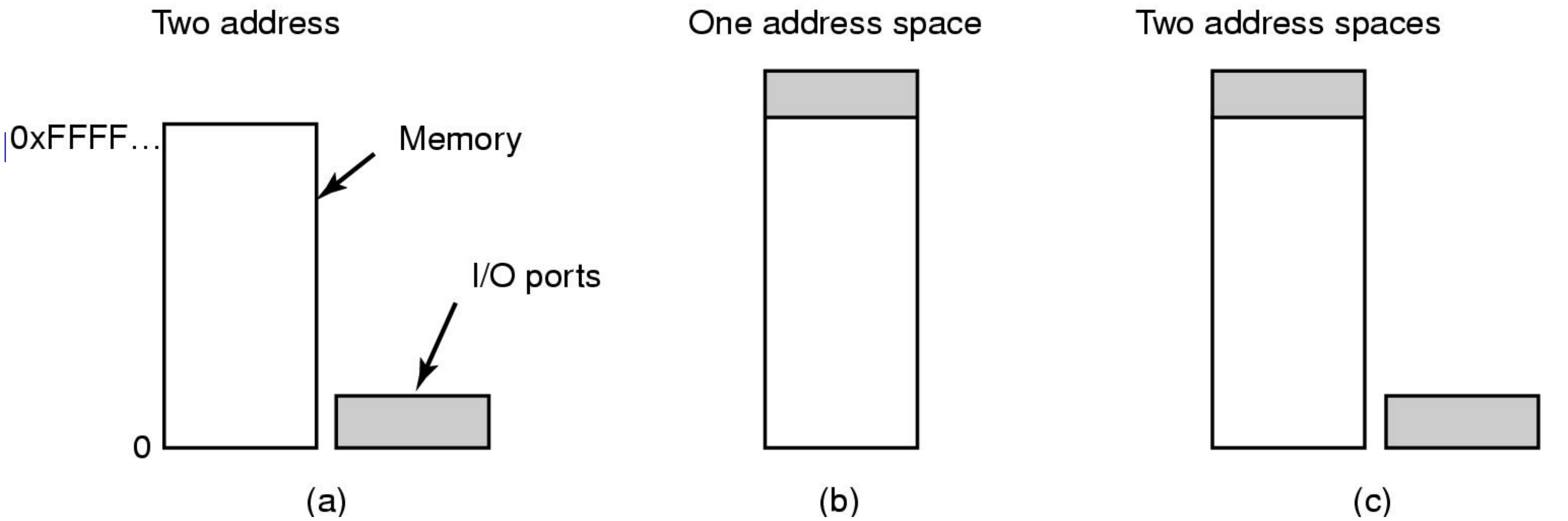
Device	Data rate
Keyboard	10 bytes/sec
Mouse	100 bytes/sec
56K modem	7 KB/sec
Telephone channel	8 KB/sec
Dual ISDN lines	16 KB/sec
Laser printer	100 KB/sec
Scanner	400 KB/sec
Classic Ethernet	1.25 MB/sec
USB (Universal Serial Bus)	1.5 MB/sec
Digital camcorder	4 MB/sec
IDE disk	5 MB/sec
40x CD-ROM	6 MB/sec
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

Device Controller

- I/O devices have two types of components:
  - the mechanical component(s)
    - often the major reason for high latency, i.e. low performance
  - the electronic component(s)

- Electronic component = device controller
  - Controller's tasks:
    - process device commands
    - convert between device specific data representation (e.g., bit serial, byte parallel) and block of bytes
    - perform error correction and handshake as necessary
    - make data available to main memory
  - may be able to handle multiple devices

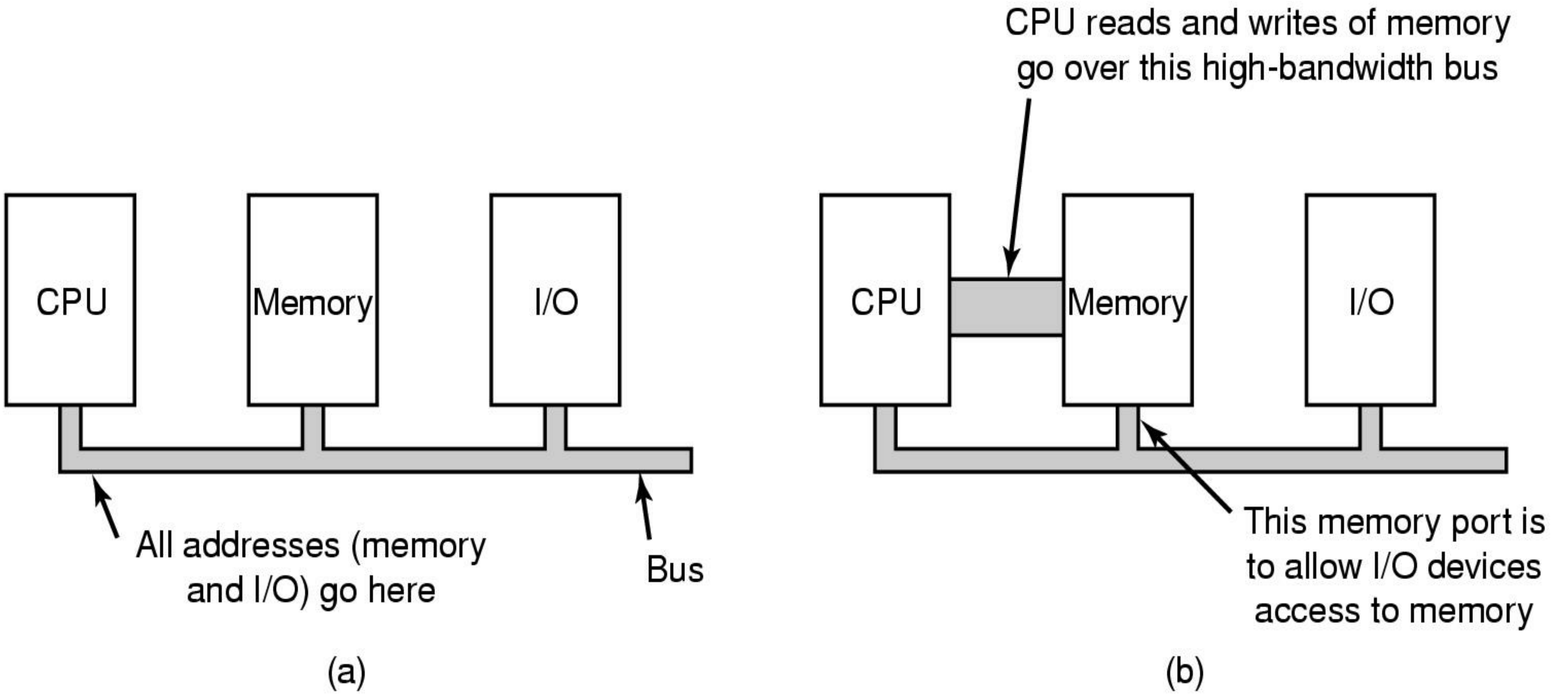
Memory-Mapped I/O (1)



- Separate I/O-address space and memory address space
  - MOV R0, 4 // <4> → R0
  - IN R0, 4 // <port 4> →R0
- Memory-mapped I/O // 1 common physical AS (PDP 11)
- Hybrid (Pentium) // part of I/O space in memory // part in an extra address space

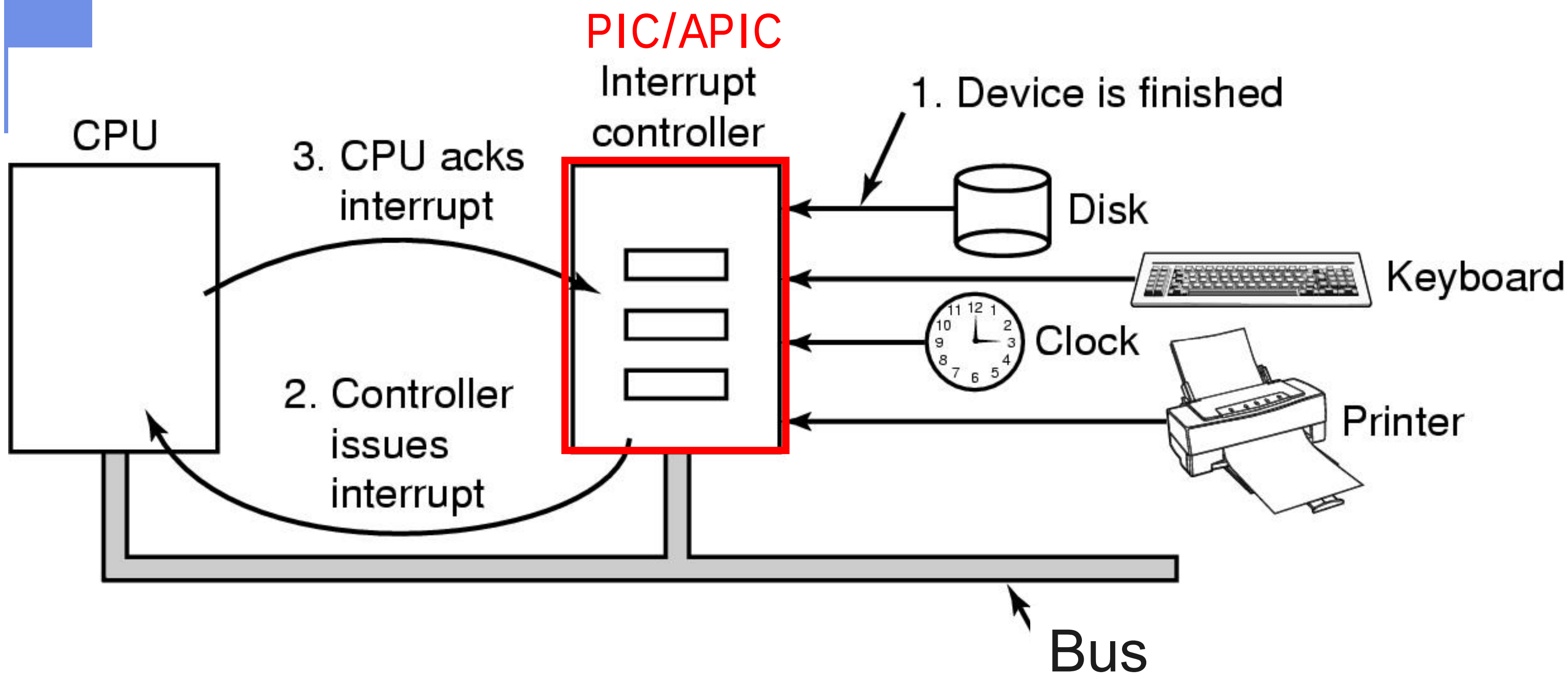
Hint: Discuss pros and cons

Memory-Mapped I/O (2)



- (a) Single-bus architecture
- (b) Dual-bus memory architecture

Repetition: Interrupts



Connections between devices and the interrupt controller use interrupt lines on the bus rather than dedicated wires

Goals of I/O-Software (1)

- Device independence
  - programs can access any I/O device without specifying device in advance
    - (floppy, hard drive, or CD-ROM)
- Uniform naming
  - name of a file or device either a string or an integer
  - not depending on machine- or device-type
- Error handling
  - handle as close to the hardware as possible $\Rightarrow$
    if hardware can handle the error, that's fine, e.g. just retry a "broken read from disk"

Goals of I/O-Software (2)

- Synchronous vs. asynchronous transfers
  - blocked transfers vs. interrupt-driven
- Buffering
  - data coming off a device cannot be stored in the final destination
  - avoid superfluous copying (see I/O-Lite*)
- Sharable vs. exclusive devices
  - disks are sharable
  - tape drives would not be

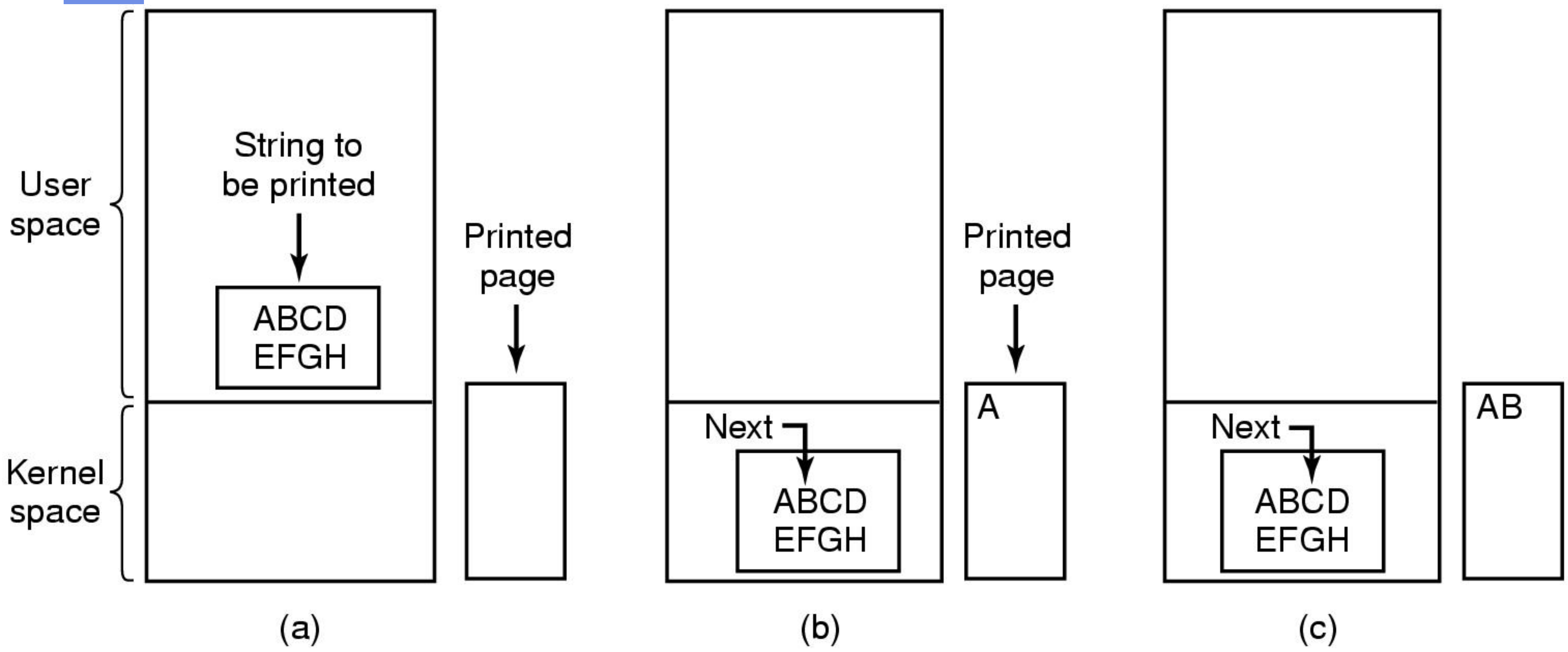
*V. Pai, P. Druschel, W. Zwaenepoel: I/O-Lite: "A Unified I/O-Buffering and Caching System", 3rd OSDI, New Orleans, 1999

Techniques for I/O-Management

- Programmed I/O
  - thread is busy-waiting for the I/O-operation to complete, processor cannot be used else where
- Interrupt-driven I/O
  - I/O-command is issued
  - processor continues executing instructions (of same or other thread)
  - I/O-device sends an interrupt when I/O-command is done
- Direct Memory Access (DMA)
  - DMA module controls exchange of data between main memory and I/O device
  - processor interrupted after entire block has been transferred

Asynchronous or synchronous I/O

Programmed I/O (1)



Steps in printing a string

Programmed I/O (2)

```
copy_from_user(buffer, p, count)    // p = kernel buf
for (i=0; i<count; i++{       // loop on each character
 while(*printer_status_reg!=READY); // loop until ready
                                    // busy waiting
    *printer_data_reg =p[i];        // output 1 character
}
return_to_user();
```

Printing string using programmed I/O

Interrupt Driven I/O

```
copy_from_user(buffer, p, count);
enable_interrupts();
while(*printer_status_reg!=READY);
 *printer_data_reg=p[0];
schedule();
```

```
if(count==0) {
   unblock_user();
}else{
 *printer_data_reg=p[i];
 count = count – 1;
 i=i+1;
}
acknowledge_interrupt();
return_from_iterrupt();
```

Printing string using interrupt-driven I/O

- Code executed when print system call is made
- Interrupt service procedure

I/O Using DMA

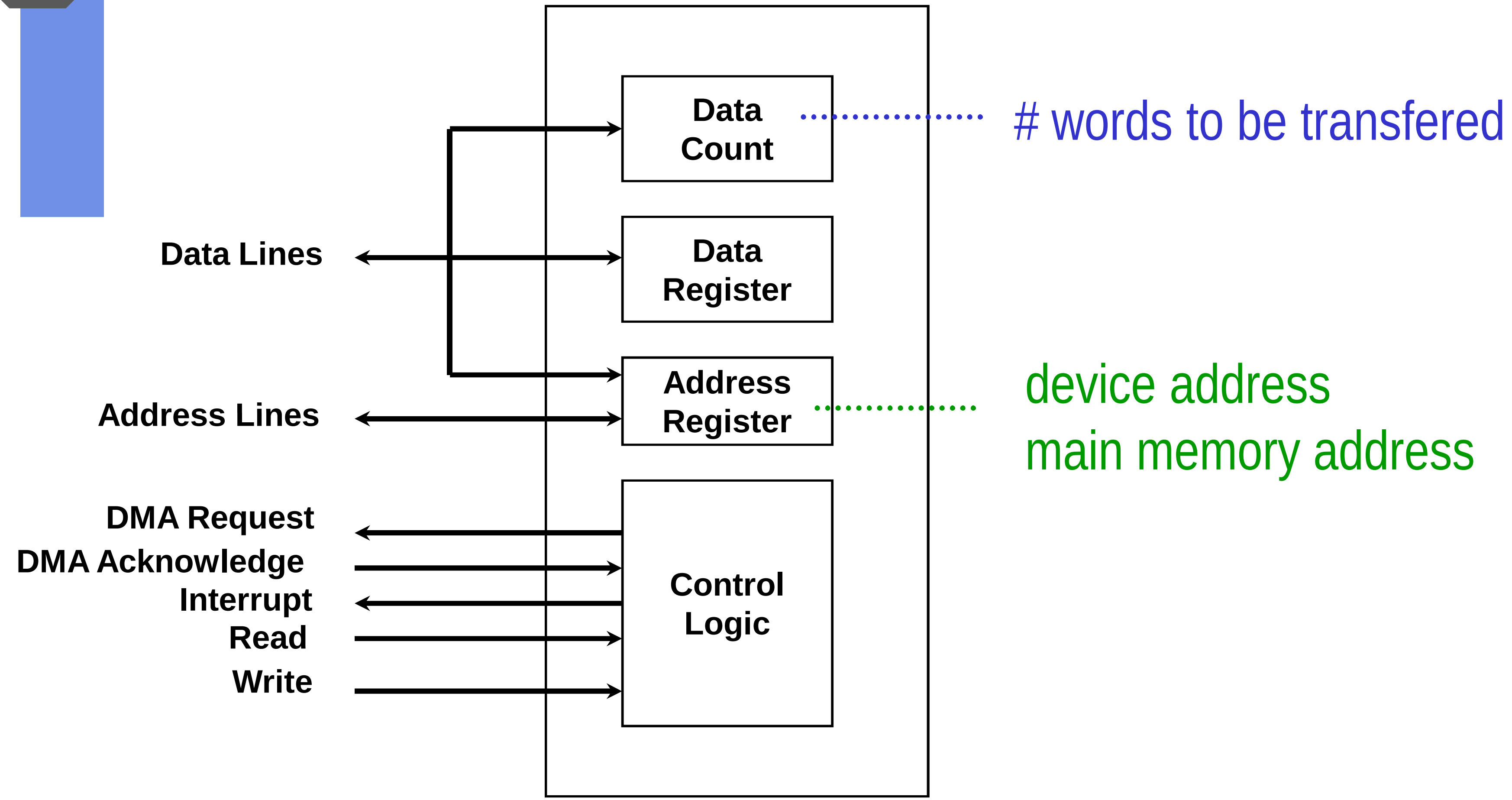
```
copy_from_user(buffer, p, count);
set_up_DMA_controller();
schedule();
```

```
acknowledge_interrupt();
unblock_user();
return_from_interrupt();
```

Printing string using DMA

- code executed when the print system call is made
- interrupt service procedure

Typical DMA Block Diagram

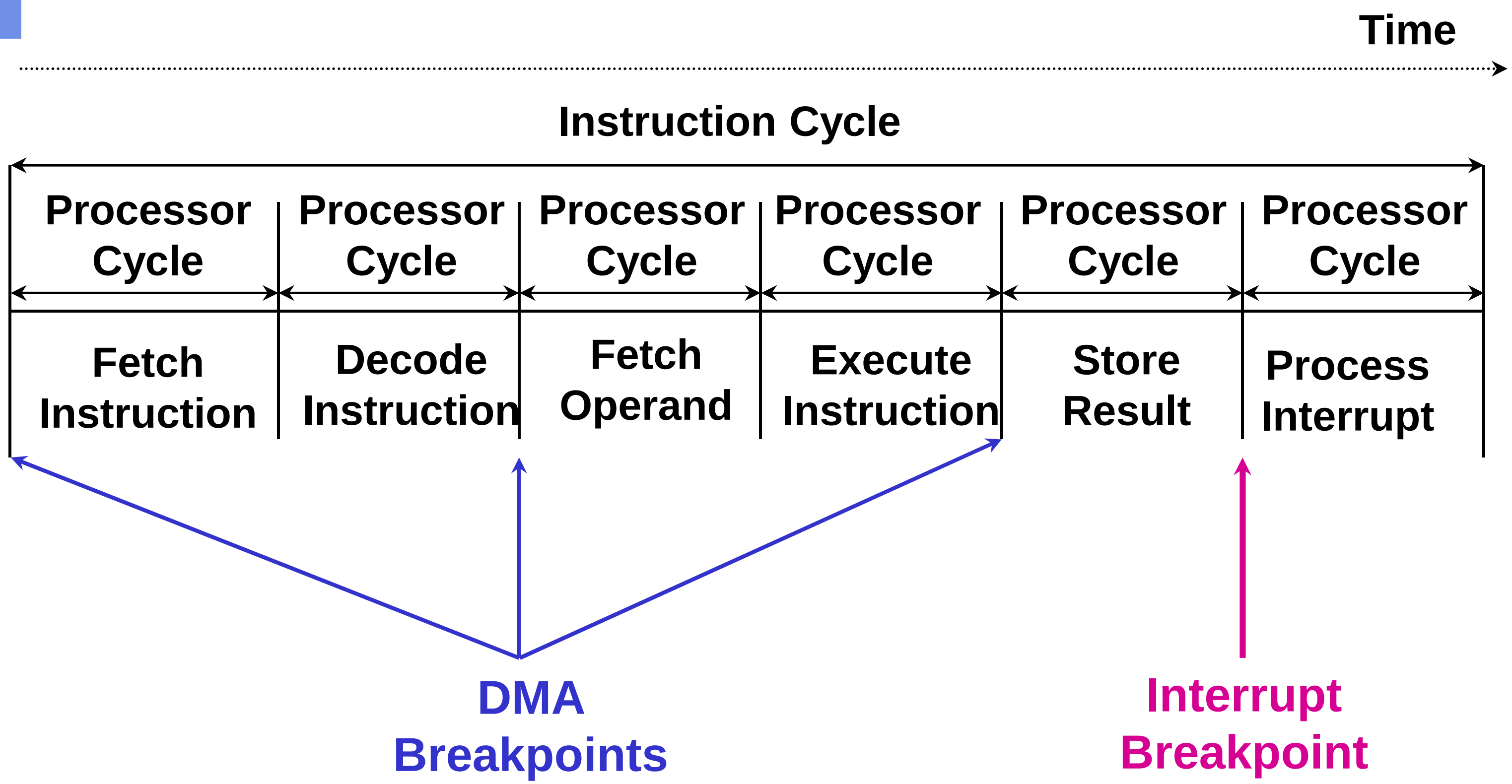


Direct Memory Access

- Takes control of system from the CPU to transfer a block of data to and from memory over system bus
- Cycle stealing: used to transfer data on the system bus, data is transferred “word” by “word*”
- Instruction cycle of CPU is suspended for a while so that a word can be transferred
- CPU pauses 1 bus cycle

* also multiple of words depending of system bus protocol

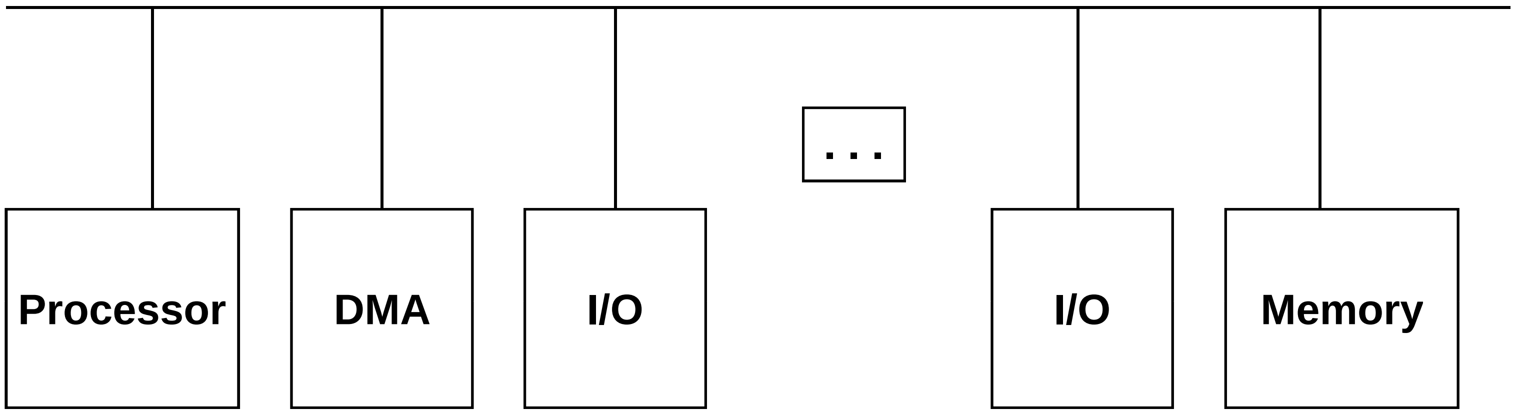
DMA and Interrupt Breakpoints



Direct Memory Access

- Cycle stealing causes CPU to slow down
- Number of required busy cycles can be reduced by integrating DMA and I/O functions
- Path between DMA module and I/O module that does not include the system bus

Single-bus, Detached DMA

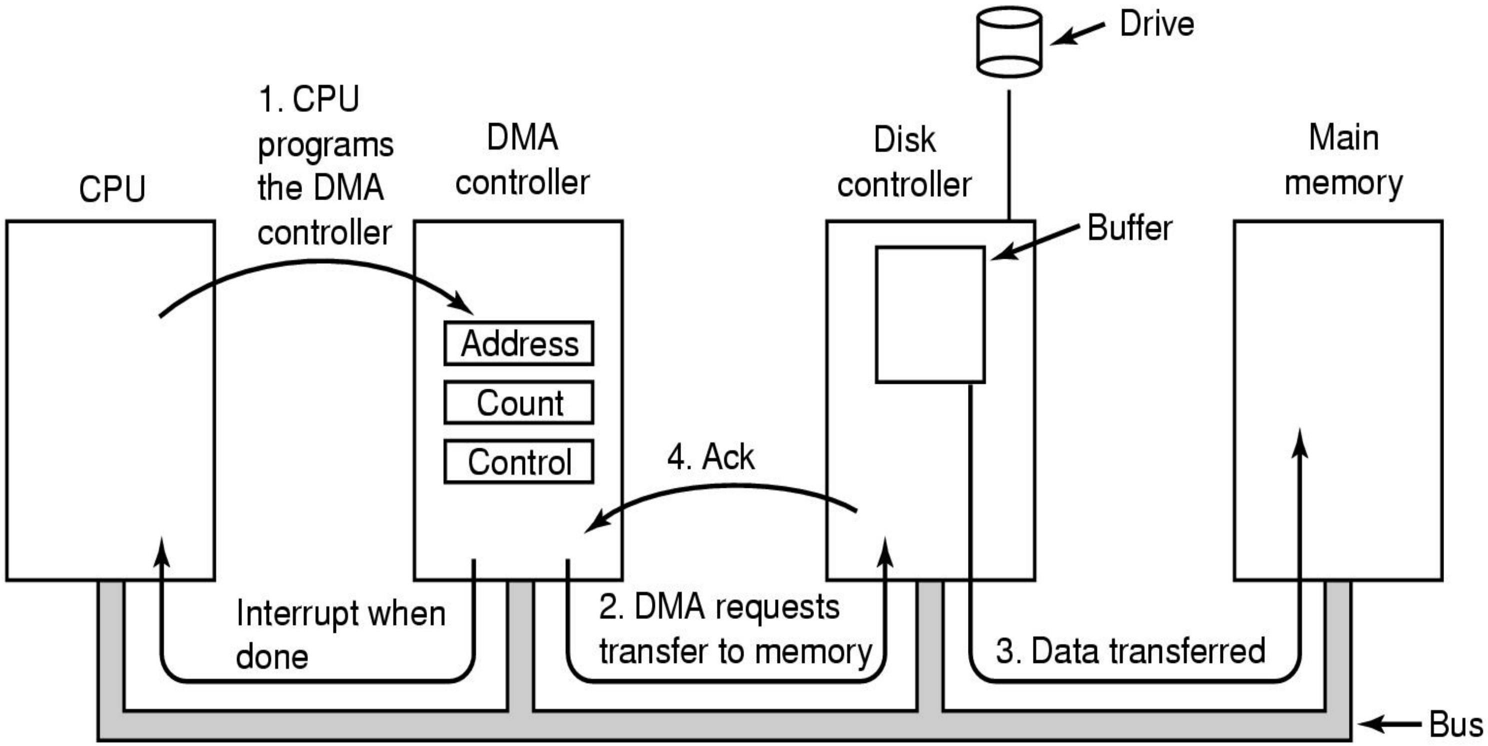


Analysis:

All modules **share same system-bus** ⇒ danger of system bus **contention**
(another example of low scalability)

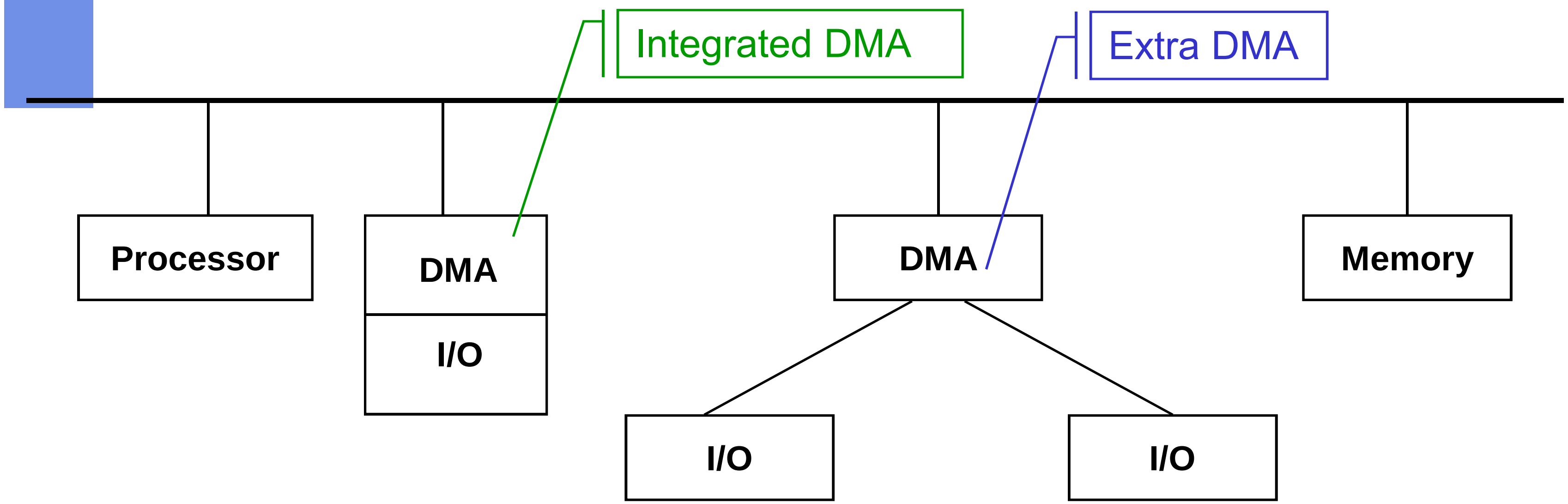
DMA uses programmed I/O to transfer data between memory and device
⇒ **each word** being transferred requires **2 bus-cycles**!

DMA Transfer with Fly-By Mode



- Word Mode (→ cycle stealing)
- Burst Mode

Single-bus, Integrated DMA-I/O



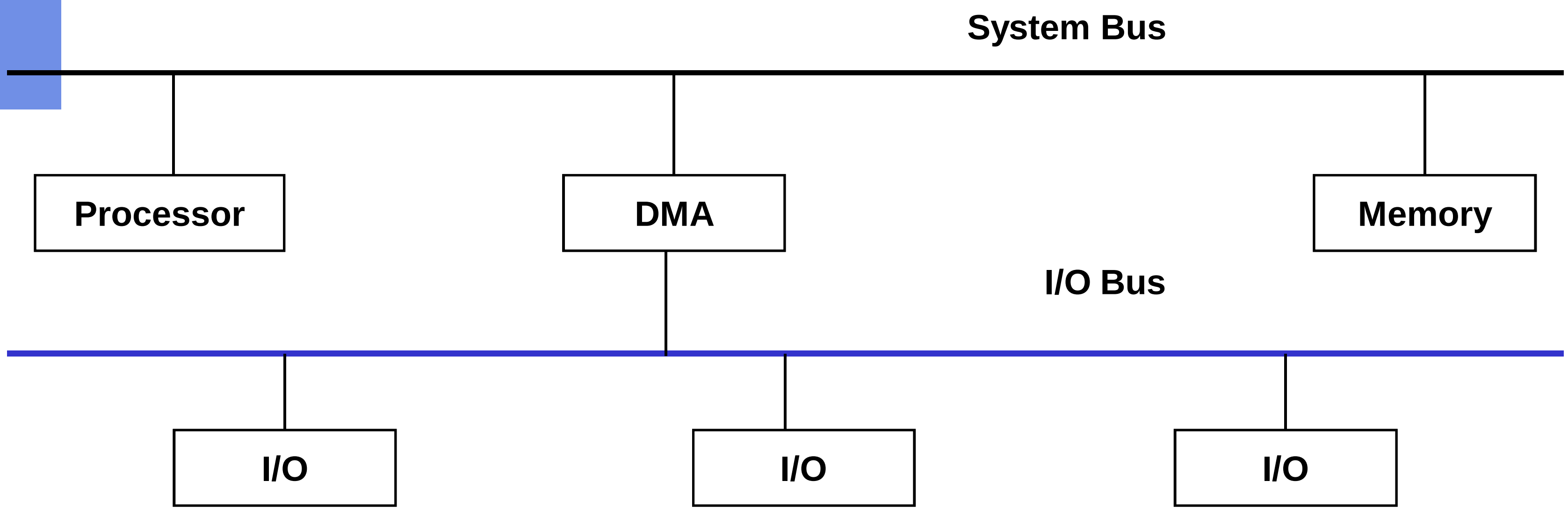
Analysis: Additional “data-lines” between DMA and I/O-devices

⇒ **fewer contention problems**

DMA competes for system bus only when it transfers a word from/to memory.

Due to extra data lines, one bus cycle is saved per word to be transferred

DMA and Separate I/O Bus



Analysis:

Additional peripheral-bus between DMA and I/O-devices

⇒ fewer contention problems on system bus and reducing the number of connections to 1 between DMA and I/O-Bus

Configuration is easily to expand (i.e. use hierarchical I/O-buses), ⇒ scalability