COMP4611: Design and Analysis of Computer Architectures

Input/Output

Lin Gu

CSE, HKUST

Motivation: Who Cares About I/O?

- CPU Performance: 60% improvement per year
- I/O system performance: < 10% improvement per year
  - sometimes limited by *mechanical* delays (disk I/O)
- 10% IO & 10x CPU =>   5x Performance (lose 50%)

  10% IO & 100x CPU => 10x Performance (lose 90%)
- I/O bottleneck:

  Diminishing value of faster CPUs

Input and Output Devices

I/O devices are diverse with respect to

- Behavior – input, output or storage
- User – human or machine
- Data rate – the rate at which data are transferred between the I/O device and the main memory or processor

Device	Behavior	User	Data rate (Mb/s)
Keyboard	input	human	0.0001
Mouse	input	human	0.0038
Laser printer	output	human	3.2000
Graphics display	output	human	800.0000-8000.0000
Network/LAN	input or output	machine	10.0000-10000.0000
Magnetic disk	storage	machine	240.0000-2560.0000

9 orders of magnitude range

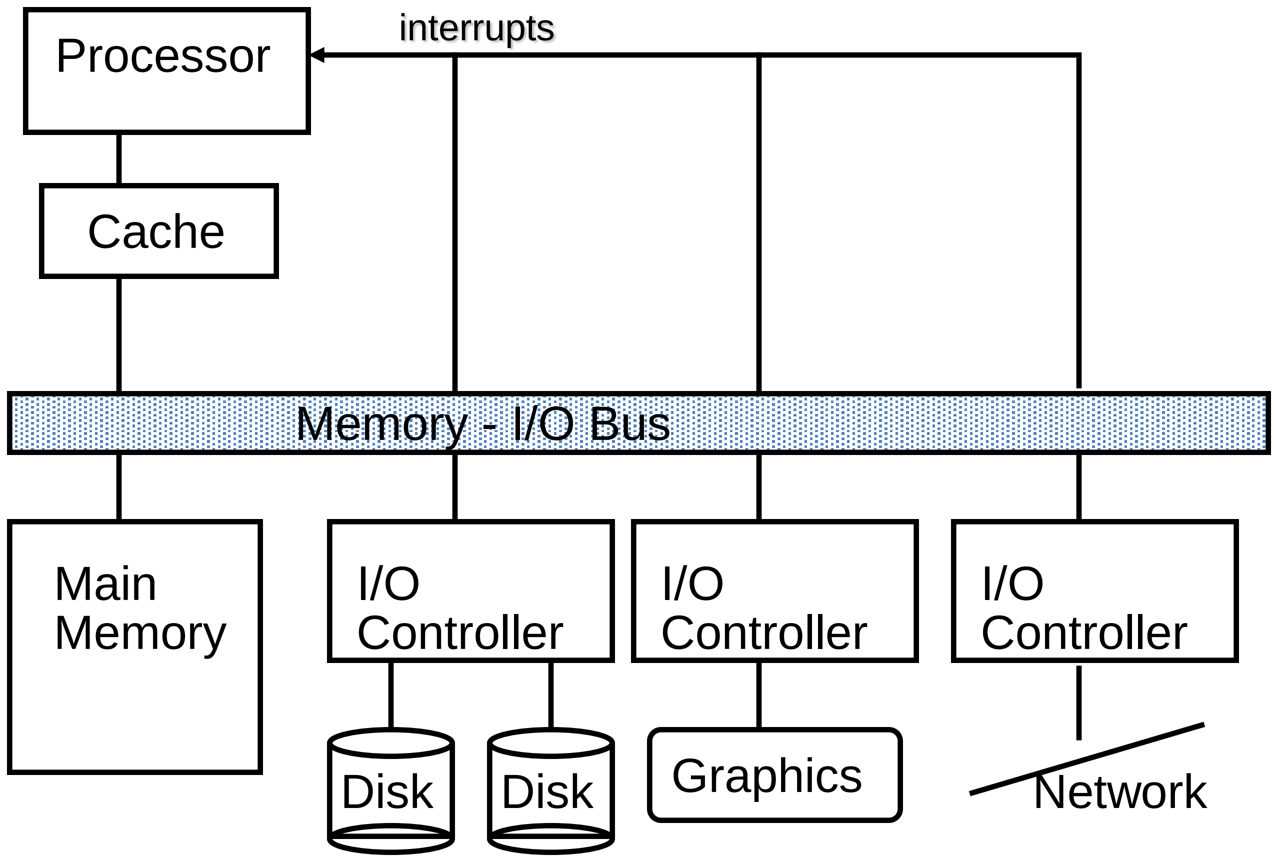
I/O Performance Measures

- I/O bandwidth (throughput) – amount of information that can be input (output) and communicated across an interconnect (e.g., a bus) to the processor/memory (I/O device) per unit time
  1. How much data can we move through the system in a certain time?
  2. How many I/O operations can we do per unit time?

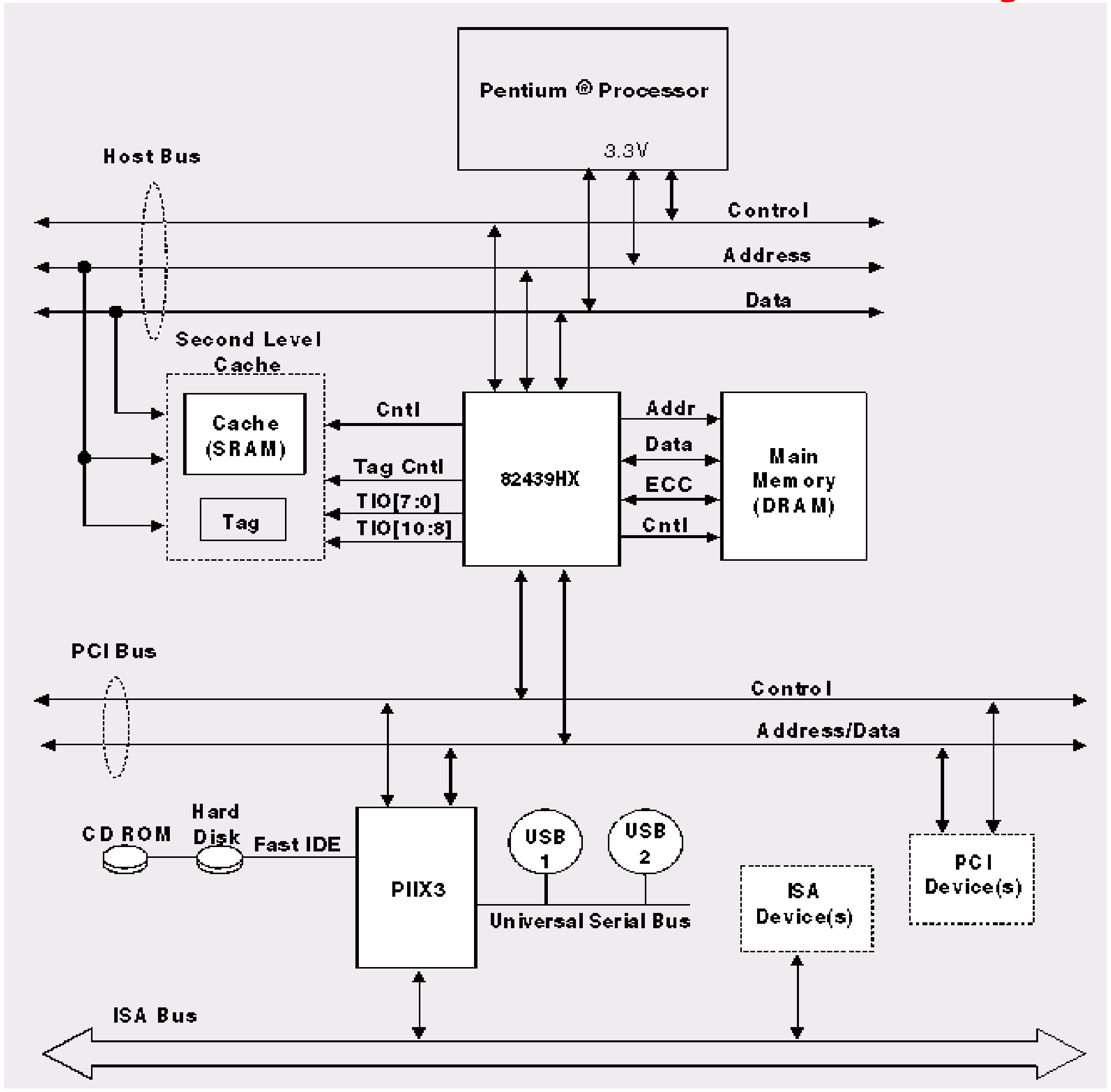
- I/O response time (latency) – the total elapsed time to accomplish an input or output operation
  - An especially important performance metric in real-time systems

- Many applications require both high throughput and short response times

A Simpe System with I/O



A More Recent System



Processor/Memory Bus

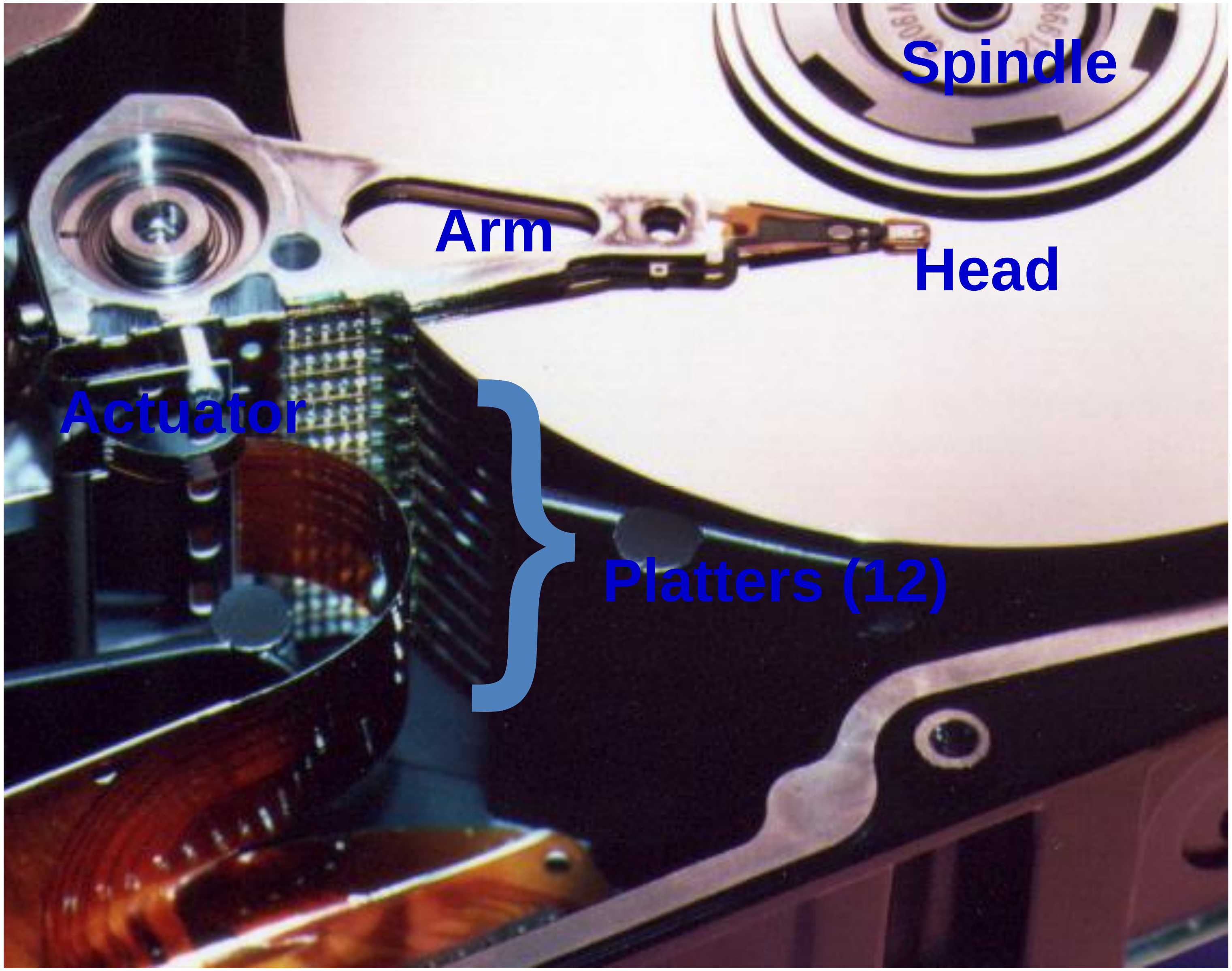
I/O Bus (PCI)

I/O Buses (ISA)

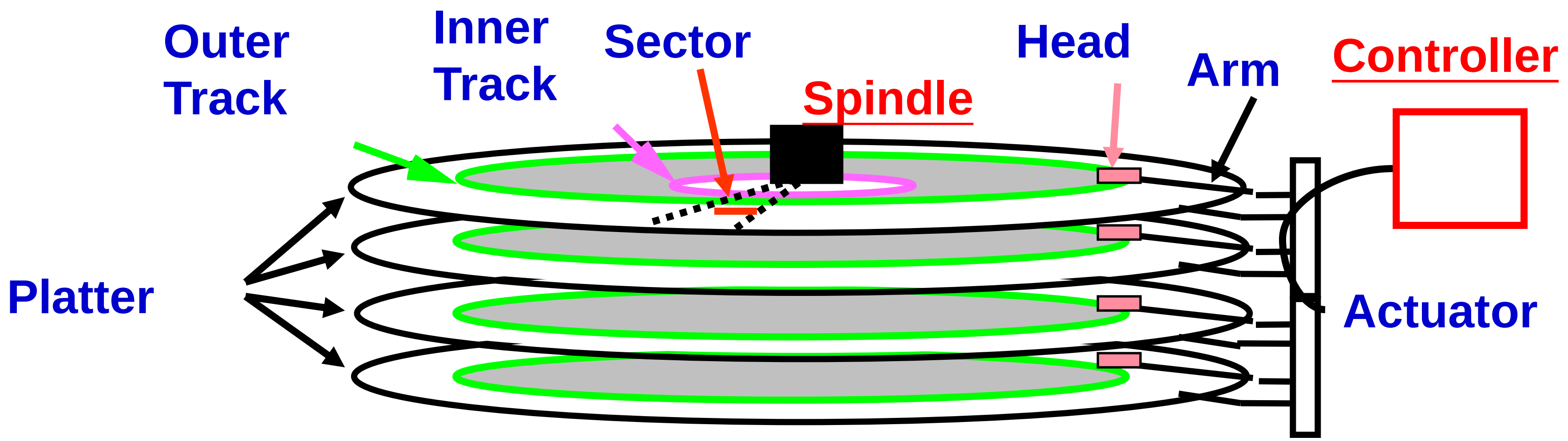
Introduction to I/O

- Most of the I/O devices are ***very slow*** compared to the cycle time of a CPU.
- The architecture of I/O systems is an active area of R&D.
  - I/O systems can define the performance of a system.
- *Computer architects* strive to design systems that do not tie up the CPU waiting for slow I/O systems (too many applications running simultaneously on processor).

Hard Disk



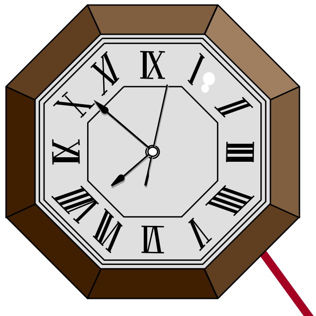
Hard Disk Performance



- Disk Latency = Seek Time + Rotation Time + Transfer Time + Controller Overhead
- Seek Time depends on the moving distance and moving speed of arms
- Rotation Time depends on how fast the disk rotates and how far sector is from head
- Transfer Time depends on data rate (bandwidth) of disk and size of request

Disk Access Time

I want block X →



→ block x in memory

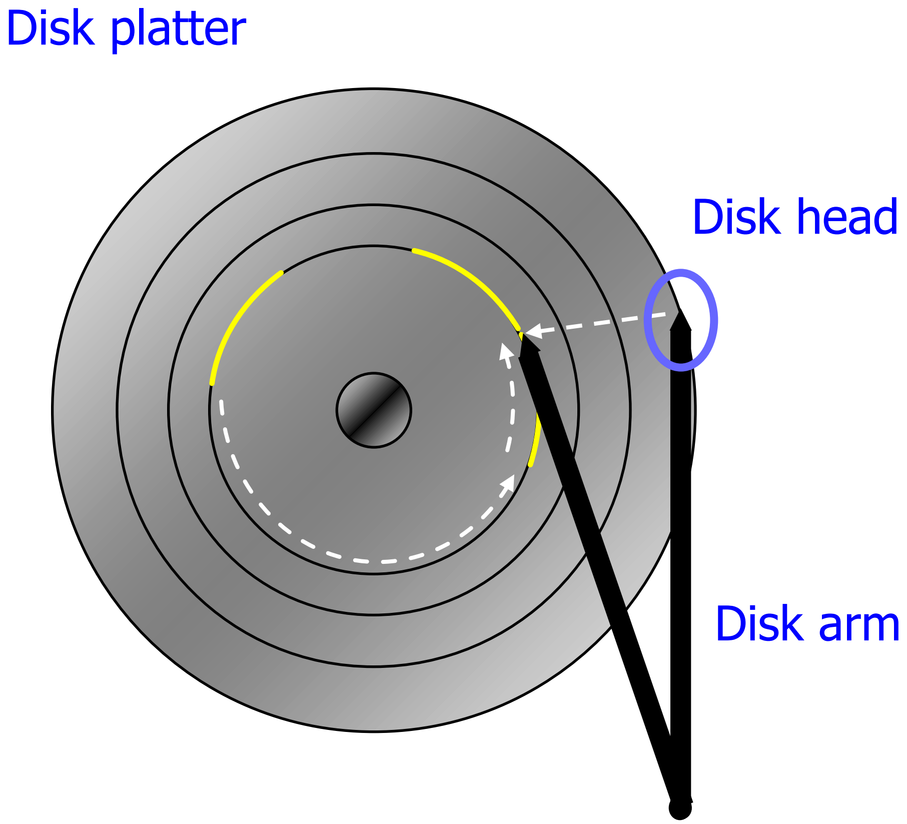
Disk access time =

Seek time

+ Rotational delay

+ Transfer time

+ Other delays



Example: Barracuda 180

- 181.6 GB, 3.5 inch disk
- 12 platters, 24 surfaces
- 24,247 cylinders
- 7,200 RPM; (4.2 ms avg. latency)
- 7.4/8.2 ms avg. seek (r/w)
- 65 to 35 MB/s (internal)
- 0.1 ms controller time

Latency =

$$\left.\begin{array}{r}\textit{per access} \\ + \\ \textit{per byte}\end{array}\right\{ \begin{array}{l}\text{Queuing Time +} \\ \text{Controller time +} \\ \text{Seek Time +} \\ \text{Rotation Time +} \\ \text{Size / Bandwidth}\end{array}$$

source: www.seagate.com

Disk Performance Example

Calculate time to read 64 KB (128 sectors) for Barracuda 180 X using advertised performance; sector is on outer track

Disk latency = average seek time + average rotational delay + transfer time + controller overhead

= 7.4 ms + 0.5 * 1/(7200 RPM)
+ 64 KB / (65 MB/s) + 0.1 ms

= 7.4 ms + 0.5 /(7200 RPM/(60000ms/M))
+ 64 KB / (65 KB/ms) + 0.1 ms

= 7.4 + 4.2 + 1.0 + 0.1 ms = 12.7 ms

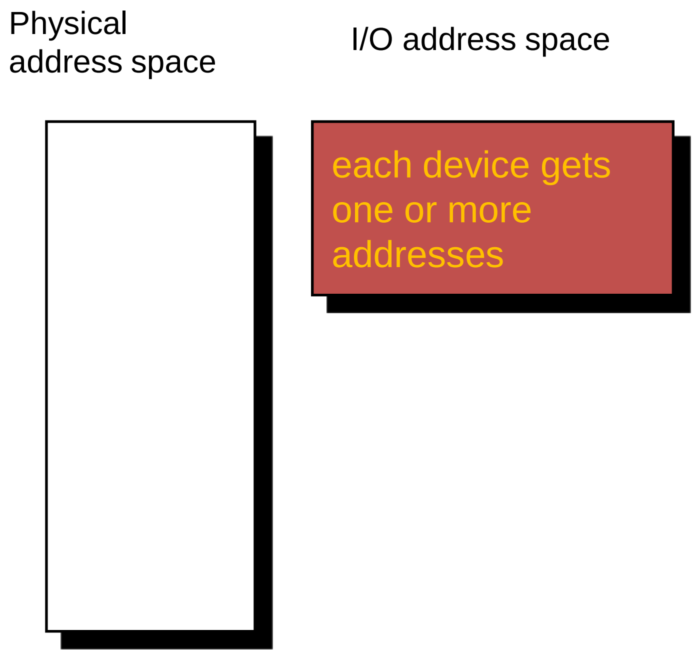
Communication of I/O Devices and Processor

How does the processor command the I/O devices?

- Special I/O instructions
  - Must specify both the device (port number) and the command
- For example:

inp reg, port; register←port

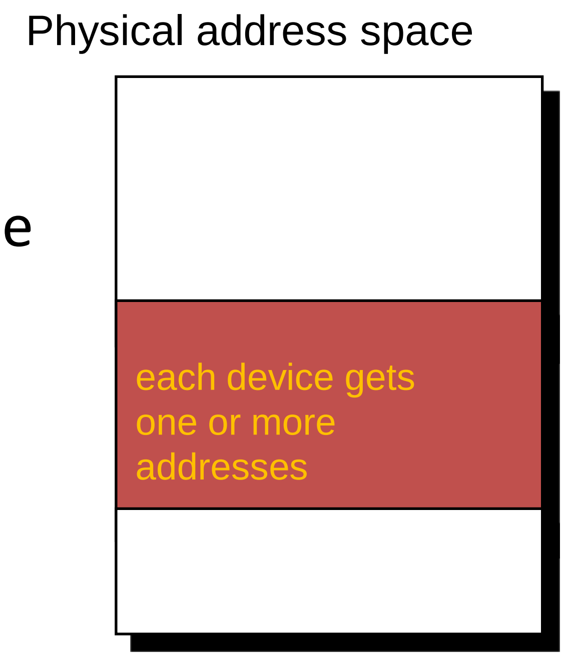
out port, reg; port→register



Communication of I/O Devices and Processor

How does the processor command the I/O devices?

- Memory-mapped I/O
  - Portions of the memory address space are assigned to I/O devices
  - Read and writes to those memory addresses are interpreted as commands to the I/O devices
  - Load/stores to the I/O address space can only be done by the OS

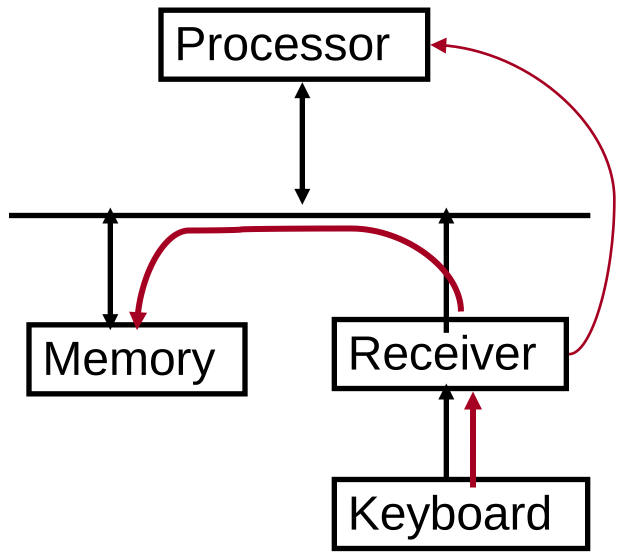


Communication of I/O Devices and Processor

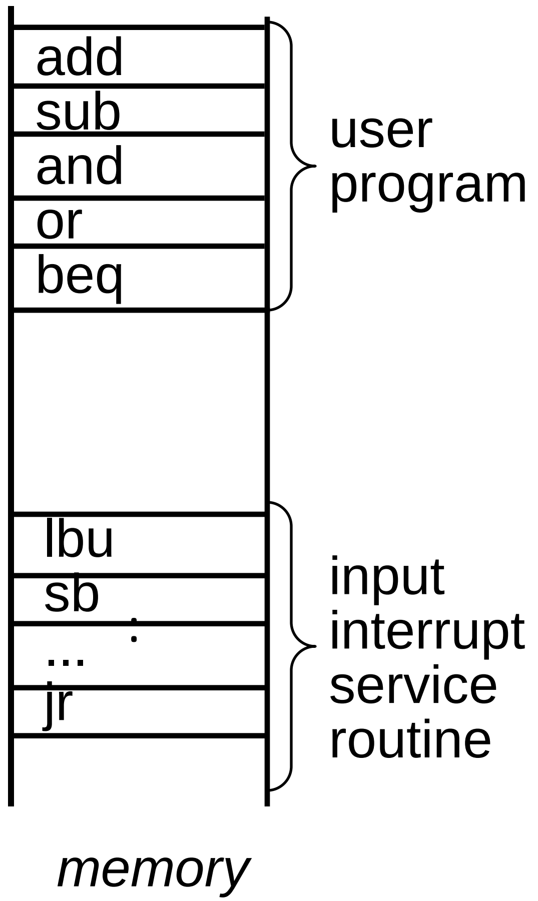
How does the I/O device communicate with the processor?

- Polling – the processor periodically checks the status of an I/O device to determine its need for service
  - Processor is totally in control – it also does all the work
  - Can waste a lot of processor time due to speed differences
- Interrupt-driven I/O – the I/O device issues an interrupt to the processor to indicate that it needs attention

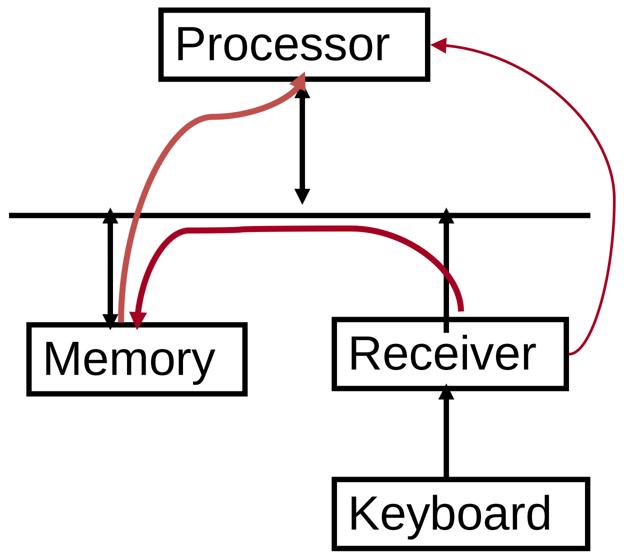
Interrupt-Driven Input

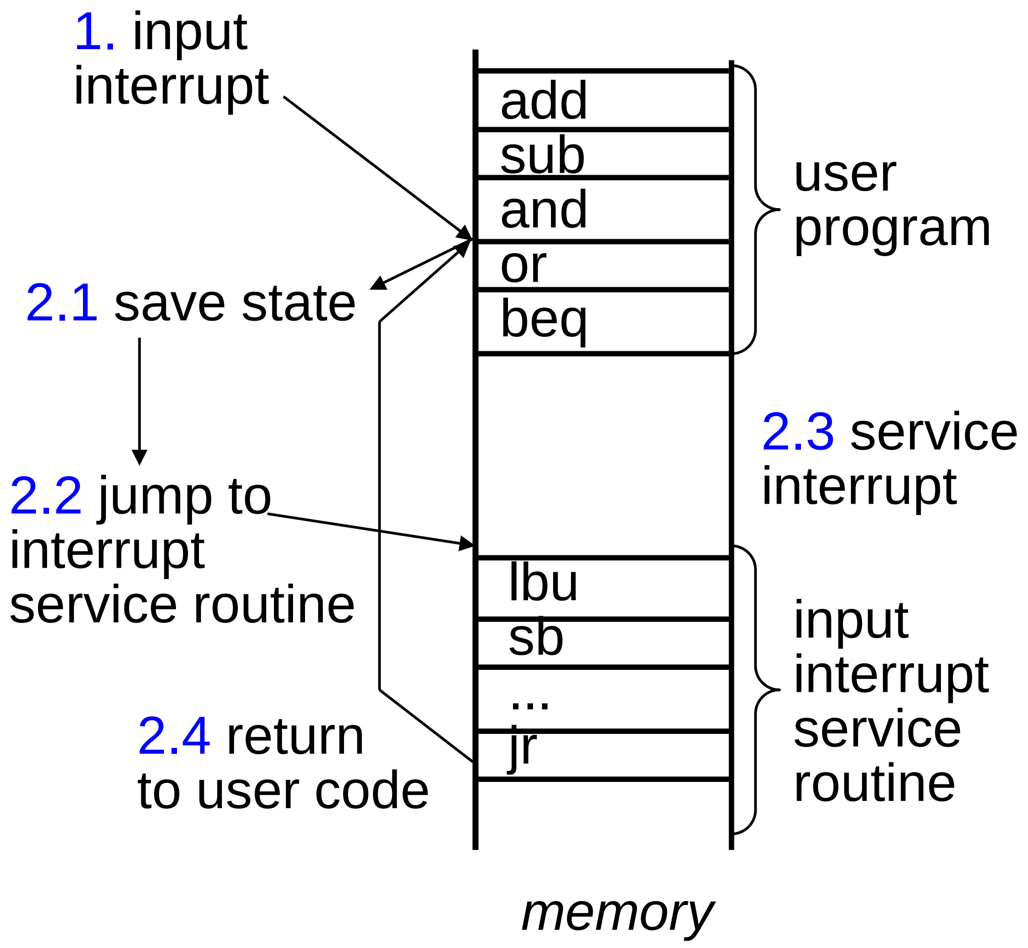


1. input interrupt

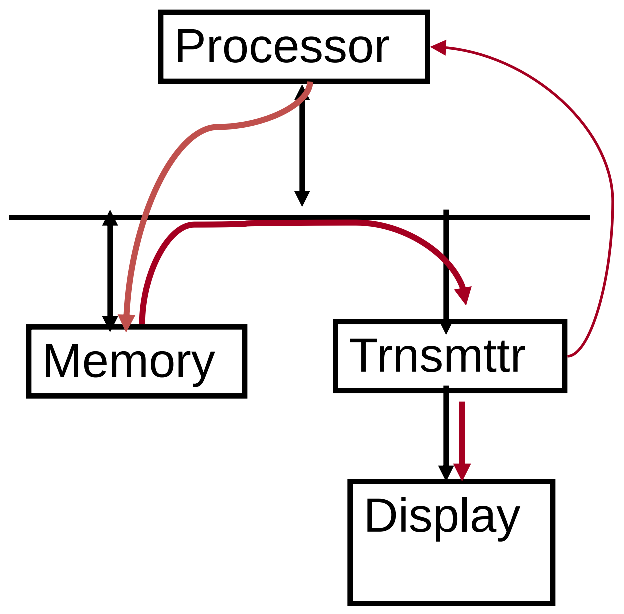


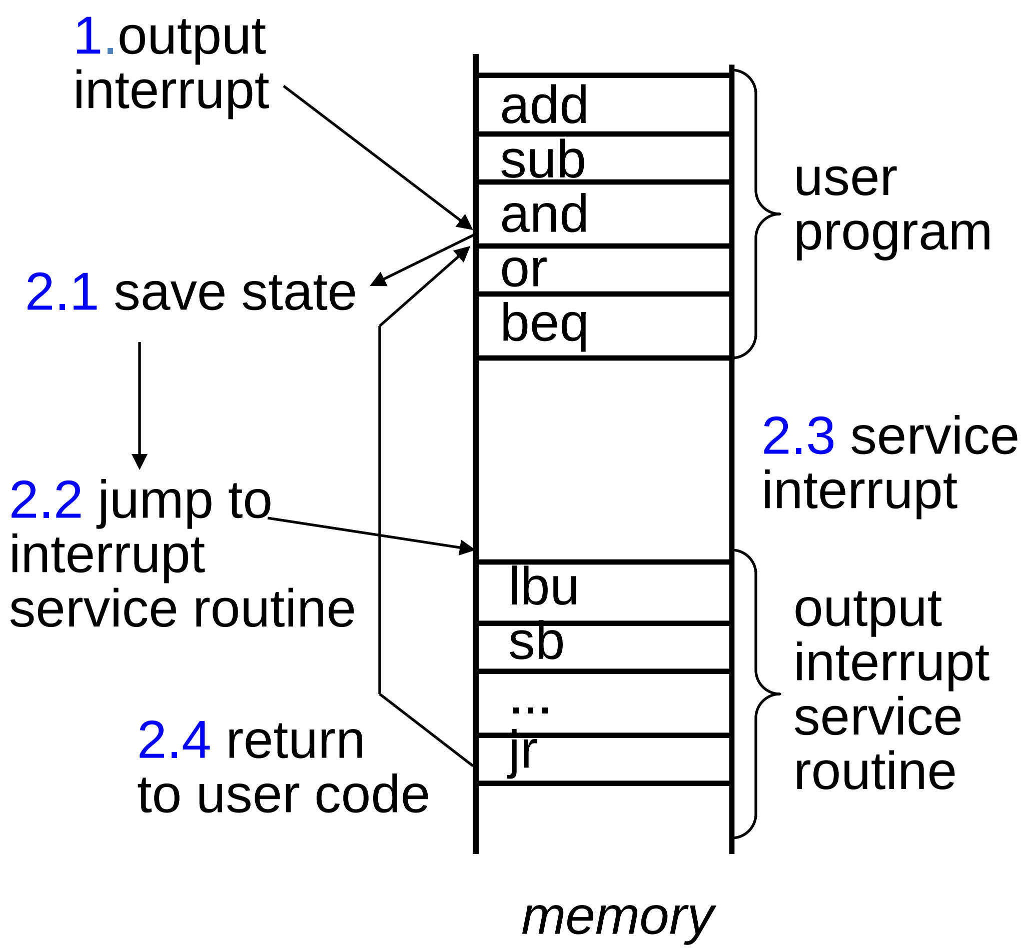
Interrupt-Driven Input





Interrupt-Driven Output



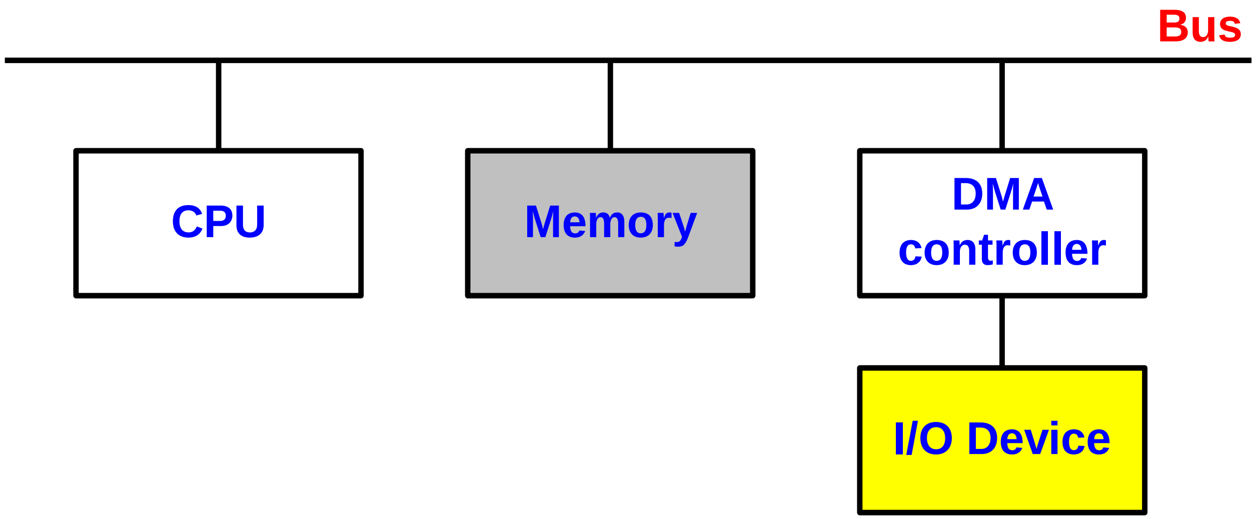


Direct-Memory Access (DMA)

- Interrupt-driven IO relieves the CPU from waiting for every IO event
- But the CPU can still be bogged down if it is used in transferring IO data.
  - Typically blocks of bytes.
- For high-bandwidth devices (like disks) interrupt-driven I/O would consume a *lot* of processor cycles

DMA

- DMA – the I/O controller has the ability to transfer data directly to/from the memory without involving the processor



DMA

- Consider printing a 60-line by 80-character page
- With no DMA:
  - CPU will be interrupted 4800 times, once for each character printed.
- With DMA:
  - OS sets up an I/O buffer and CPU writes the characters into the buffer.
  - DMA is commanded (includes the beginning address of the buffer and its size) to print the buffer.
  - DMA takes items from the buffer one-at-a-time and performs everything requested.
  - Once the operation is complete, the DMA sends a single interrupt signal to the CPU.

I/O Communication Protocols

- Typically *one* I/O channel controls *multiple* I/O devices.
- We need a two-way communication between the channel and the I/O devices.
  - The channel needs to send the command/data to the I/O devices.
  - The I/O devices need to send the data/status information to the channel whenever they are ready.

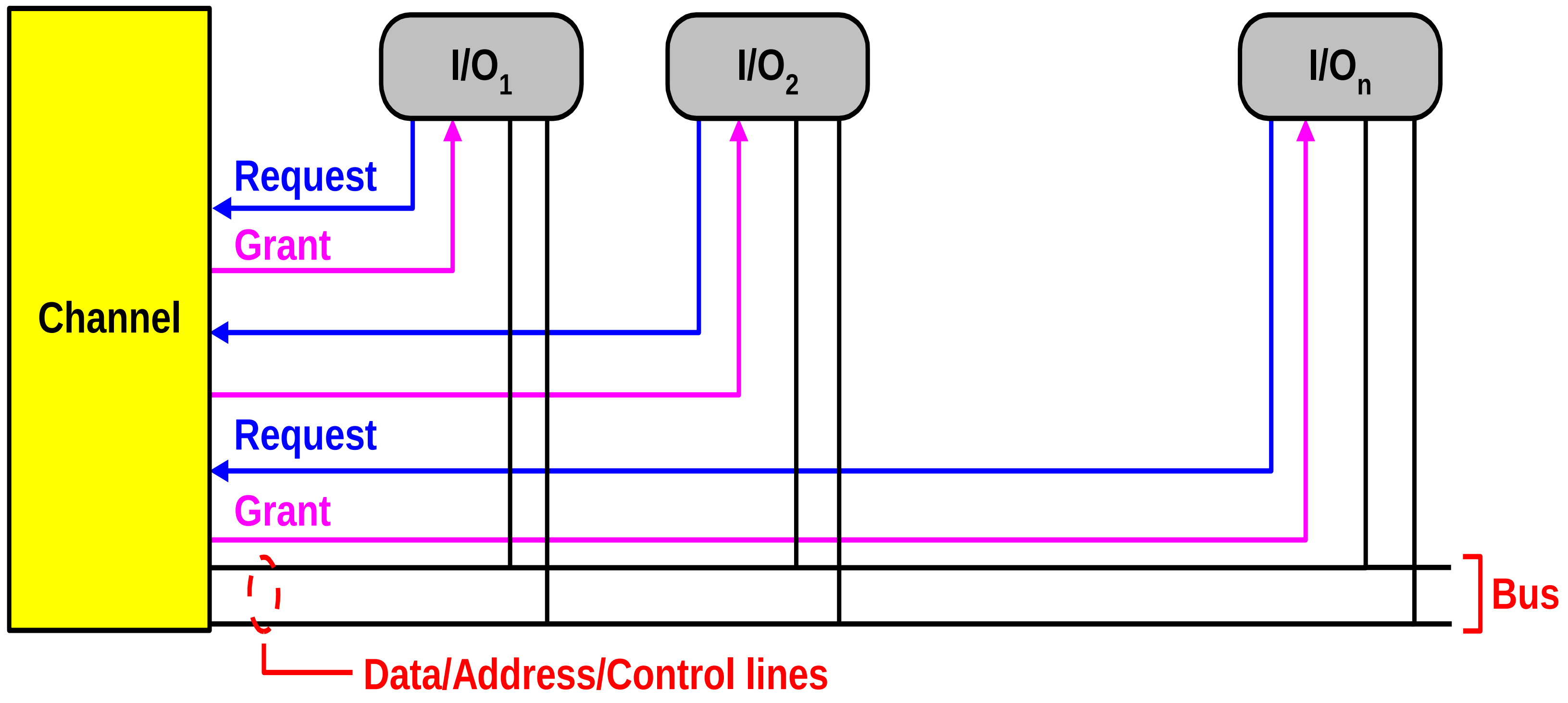
Channel to I/O Device Communication

- Channel sends the address of the device on the bus.
- All devices compare their addresses against this address.
  - Optionally, the device which has matched its address places its own address on the bus again.
    - First, it is an acknowledgement signal to the channel;
    - Second, it is a check of validity of the address.
- The channel then places the I/O command/data on the bus received by the correct I/O device.
- The command/data is queued at the I/O device and is processed whenever the device is ready.

I/O Devices to Channel Communication

- The I/O devices-to-channel communication is more complicated, since now several devices may require *simultaneous* access to the channel.
  - Need arbitration among multiple devices (bus master?)
  - May prefer a priority scheme
- Three methods for providing I/O devices-to-channel communication

Independent Requests



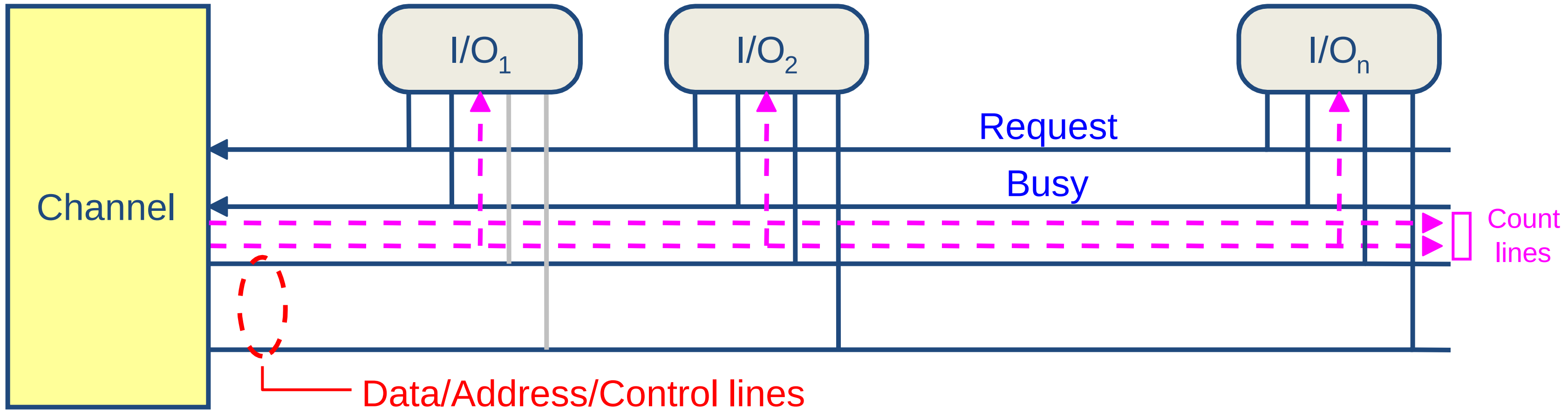
Independent Requests

Each device has its own Request-Grant lines:

- A device sends in its request, the channel responds by granting access
- Only the device that holds the grant signal can access the bus
- When a device finishes access, it lowers it request signal.
- The channel can use either a Priority scheme or Round-Robin scheme to grant the access.

Polling

The channel interrogates (polls) the devices to find out which one requested access:

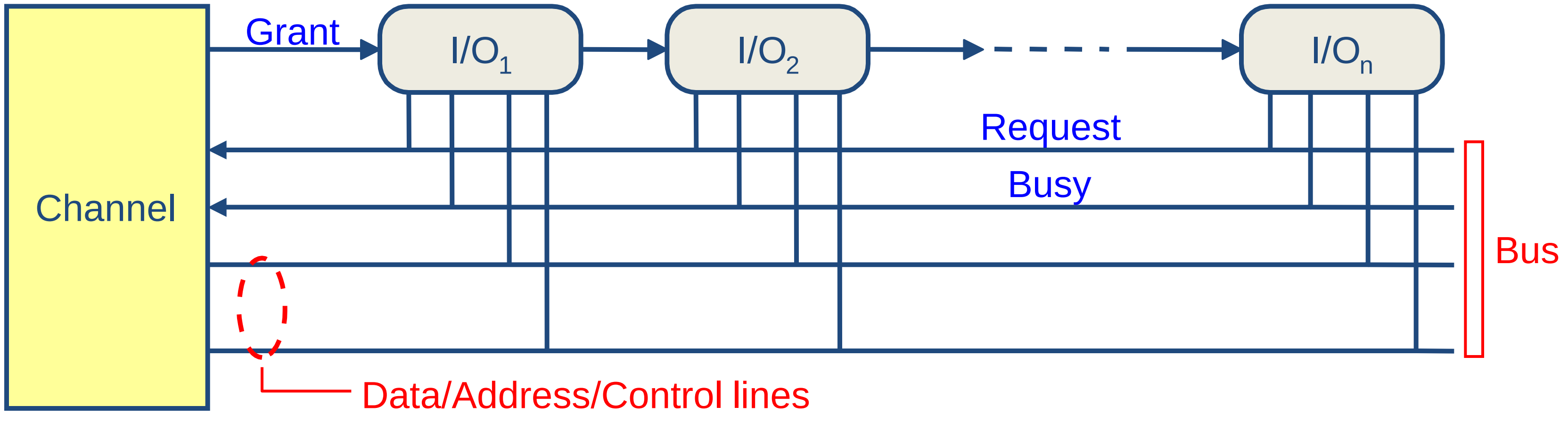


Polling

- Any device requesting access places a signal on request line.
- If the busy signal is off, the channel begins polling the devices to see which one is requesting access.
  - It does this by sequentially sending a count from 1 to *n* on $\log_2 n$ lines to the devices.
- Whenever a requesting device matches the count against its own number (address), it activates the busy line.
- The channel stops the count (polling) and the device has access over the bus.
- When access is over, the busy line is deactivated and the channel can either continue the count from the last device (Round-Robin) or start from the beginning (priority).

Daisy Chaining

- Two schemes
- Centralized control (priority scheme)

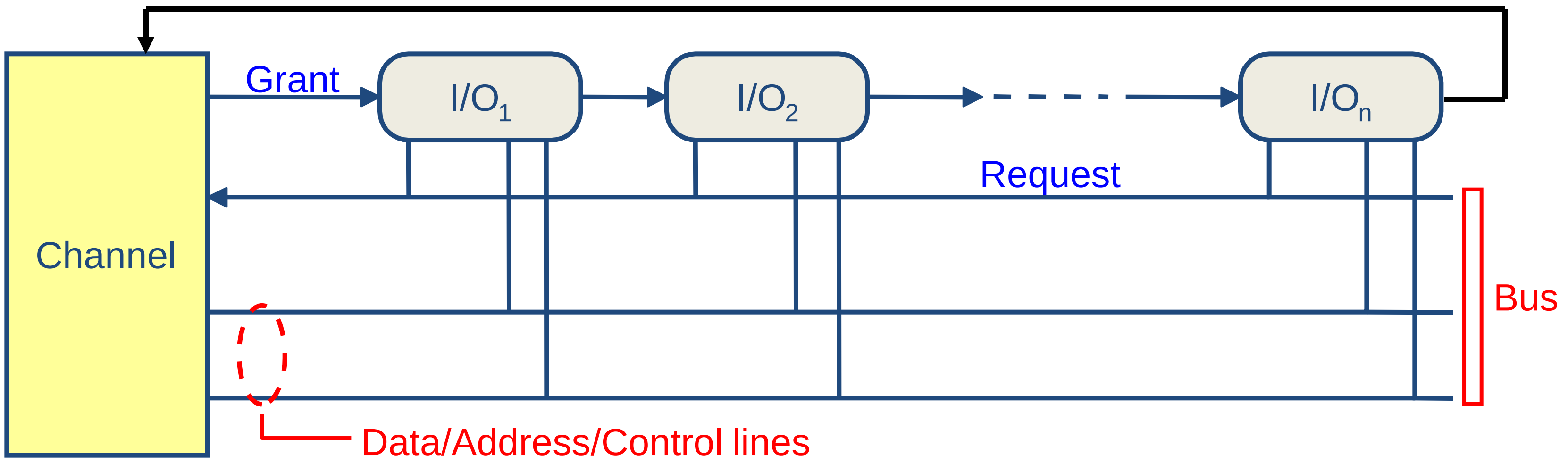


Daisy Chaining

- The I/O devices activate the request line for bus access.
- If the bus is not busy (indicated by no signal on busy line), the channel sends a Grant signal to the first I/O device (closest to the channel).
  - If the device is not the one that requested the access, it propagates the Grant signal to the next device.
  - If the device is the one that requested an access, it then sends a busy signal on the busy line and begins access to the bus.
- Only a device that holds the Grant signal can access the bus.
- When the device is finished, it resets the busy line.
- The channel honors the requests only if the bus is not busy.
- Obviously, devices closest to the channel have a higher priority and block access requests by lower priority devices.

Daisy Chaining

- Decentralized control (Round-robin Scheme)



Daisy Chaining

- The I/O devices send their request.
- The channel activates the Grant line.
- The first I/O device which requested access accepts the Grant signal and has control over the bus.
  - Only the devices that have received the grant signal can have access to the bus.
- When a device is finished with an access, it checks to see if the request line is activated or not.
- If it is activated, the current device sends the Grant signal to the next I/O device (Round-Robin) and the process continues.
  - Otherwise, the Grant signal is deactivated.

I/O Buses

- Connect I/O devices (channels) to memory.
  - Many types of devices are connected to a bus.
  - Have a wide range of bandwidth requirements for the devices connected to a bus.
  - Typically follow a bus standard, e.g., PCI, SCSI.
- Clocking schemes:
  - Synchronous: The bus includes a clock signal in the control lines and a fixed protocol for address and data relative to the clock
  - Asynchronous: The bus is self-timed and uses a handshaking protocol between the sender and receiver

Buses

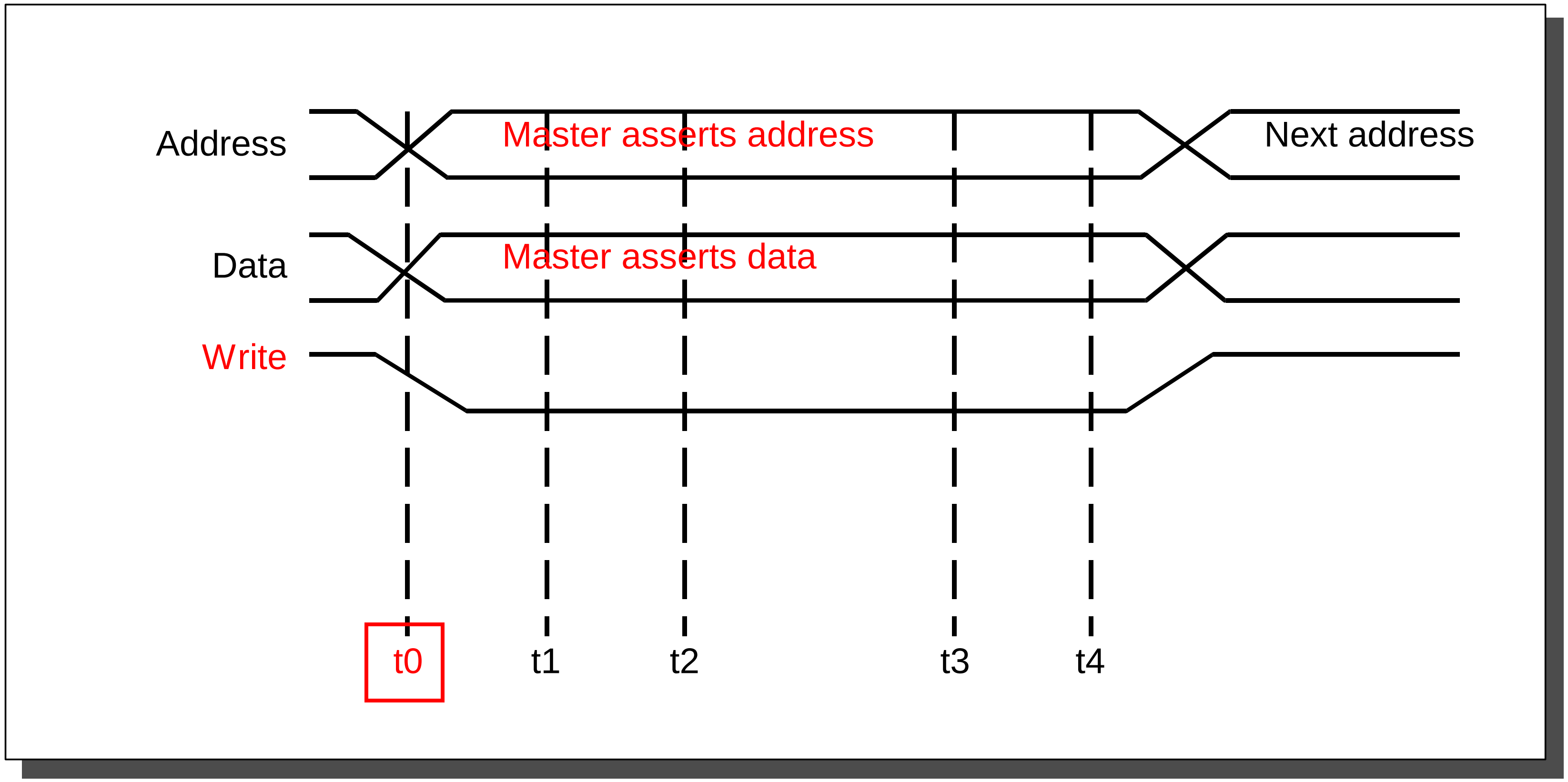
Synchronous buses are fast and inexpensive, but

- All devices on the bus must run at the same clock rate.
- Due to clock-skew problems, buses cannot be long.
- *CPU-Memory* buses are typically implemented as synchronous buses.
  - The front side bus (FSB) clock rate typically determines the clock speed of the memory you must install.

Buses

- Asynchronous buses are self-timed and use a handshaking protocol between the sender and receiver.
- This allows the bus to accommodate a wide variety of devices and to lengthen the bus.
- I/O buses are typically asynchronous.
  - A master (e.g., an I/O channel writing into memory) asserts address, data, and control and begins the handshaking process.

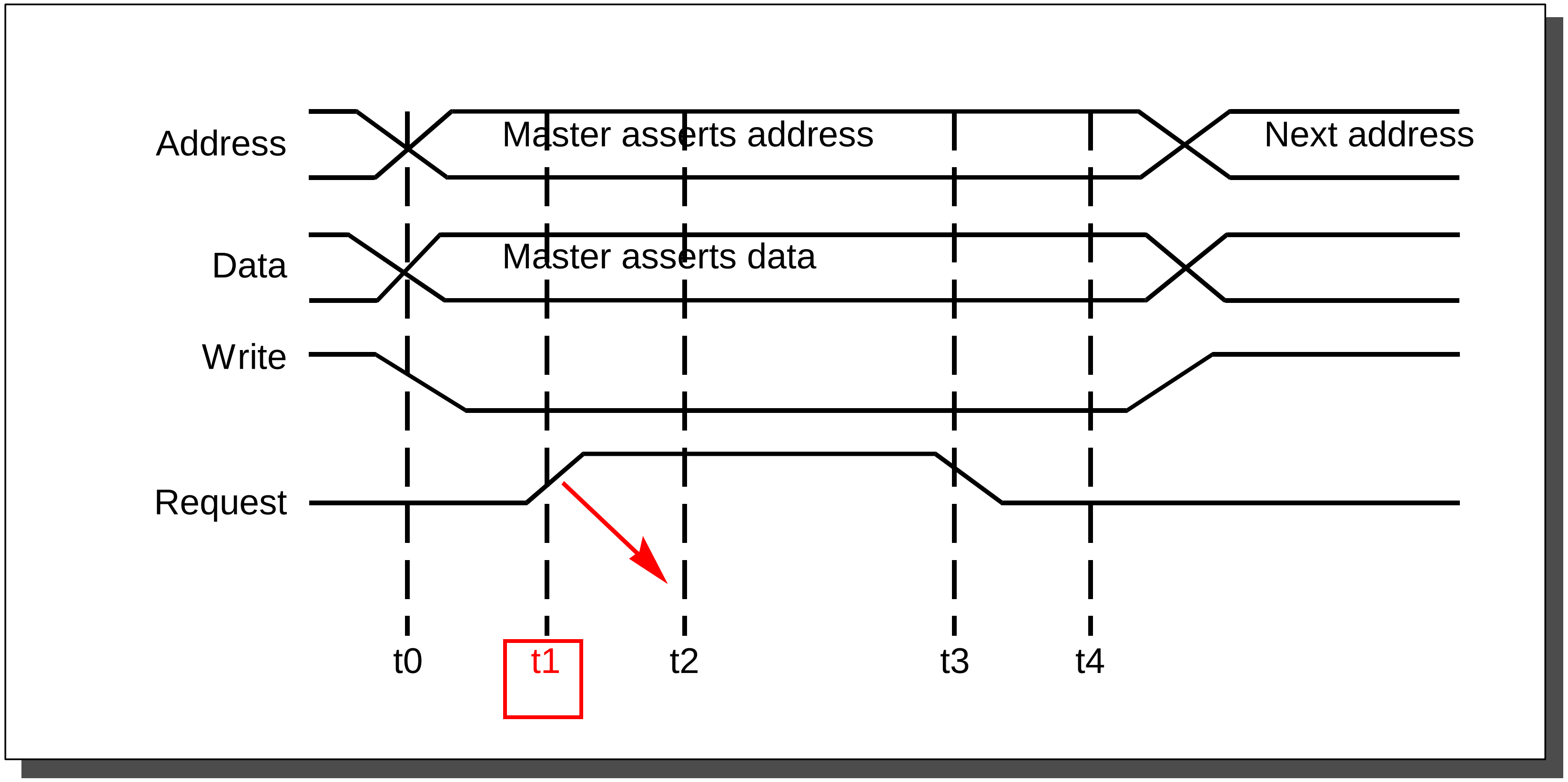
I/O Buses



Asynchronous write: master asserts address, data, write buses.

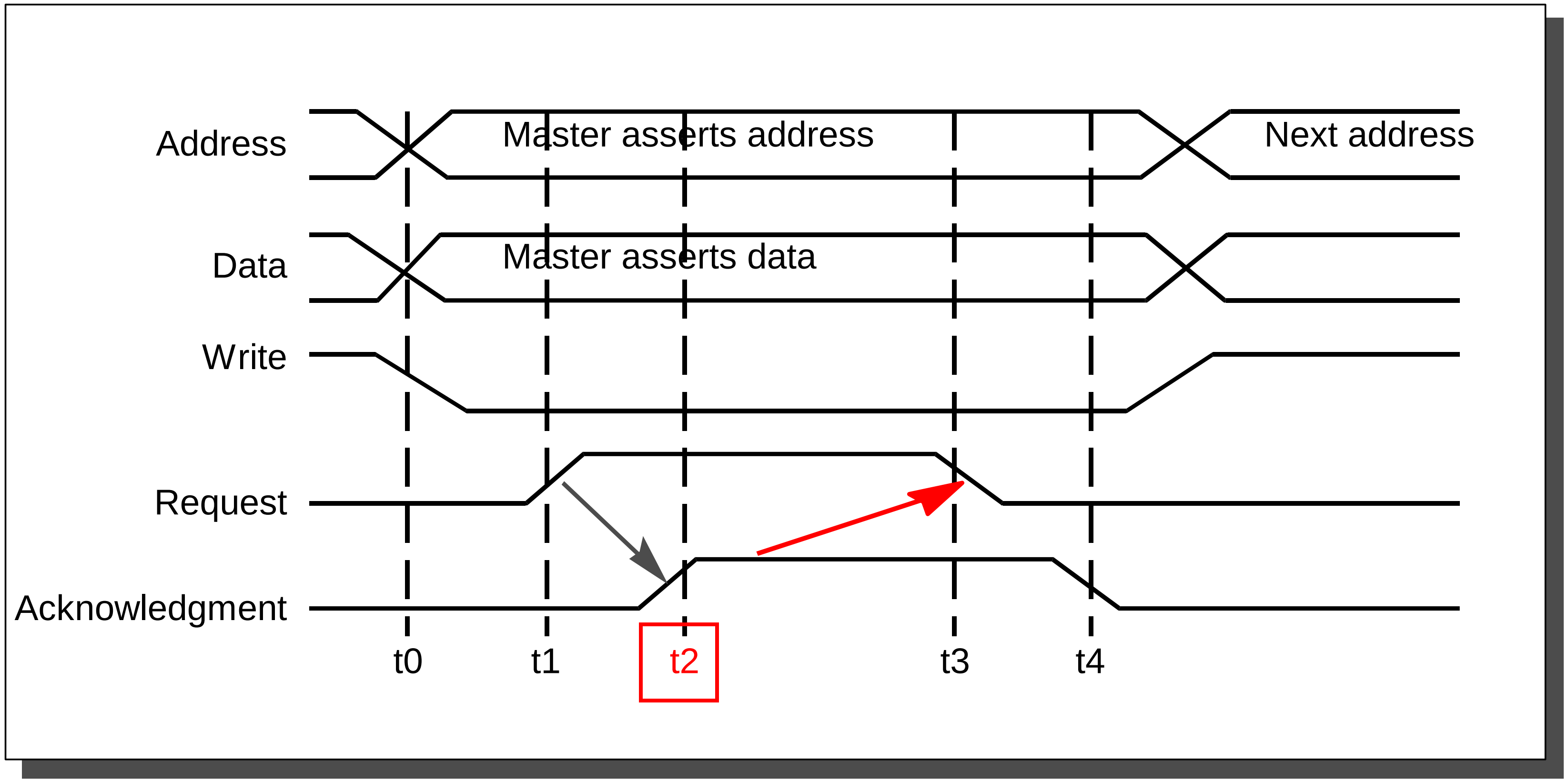
I/O Buses

Asynchronous write: master asserts request, expecting acknowledgement later.



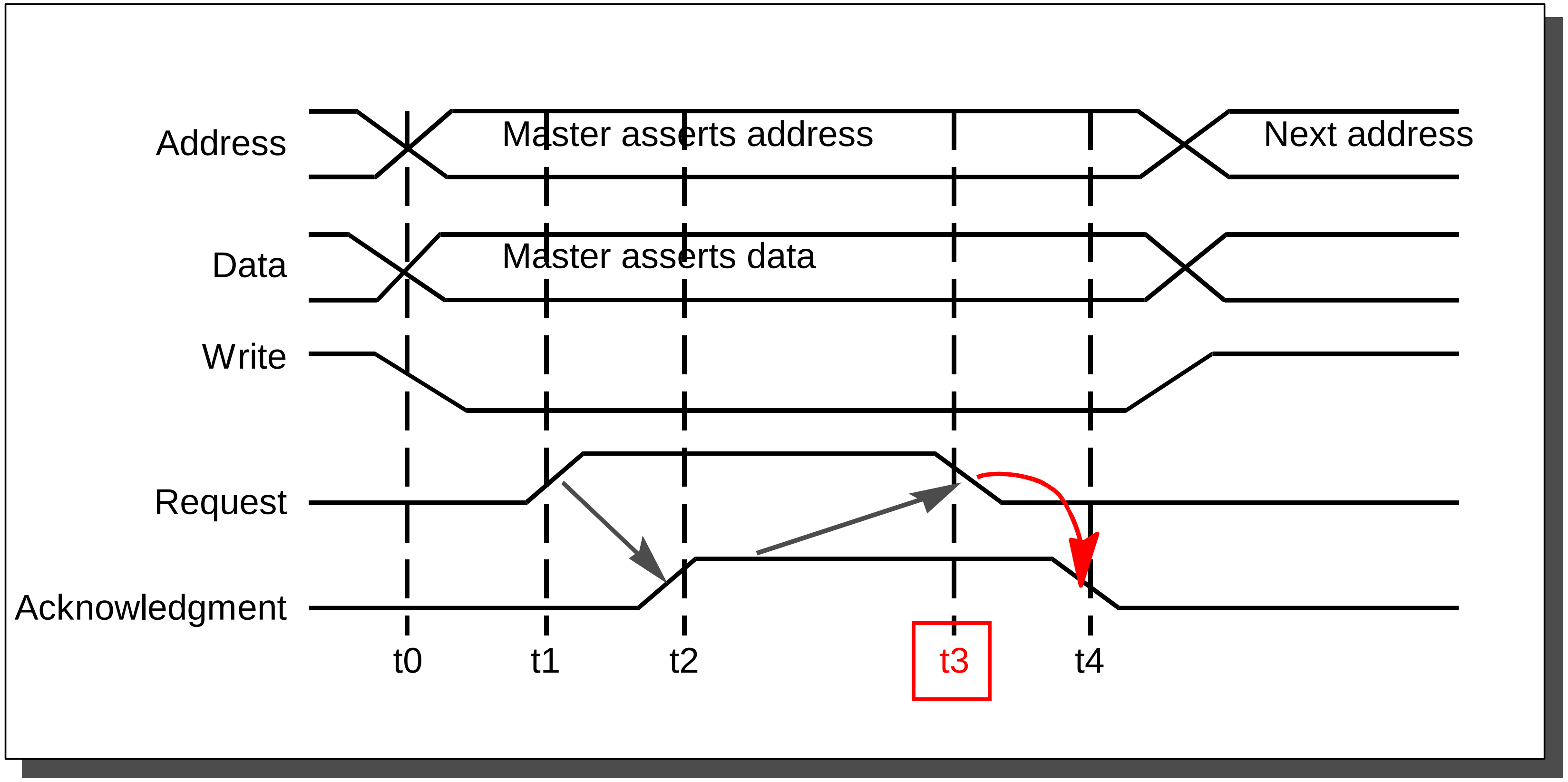
I/O Buses

Asynchronous write: slave (memory) asserts acknowledgment, expecting request to be deasserted later.



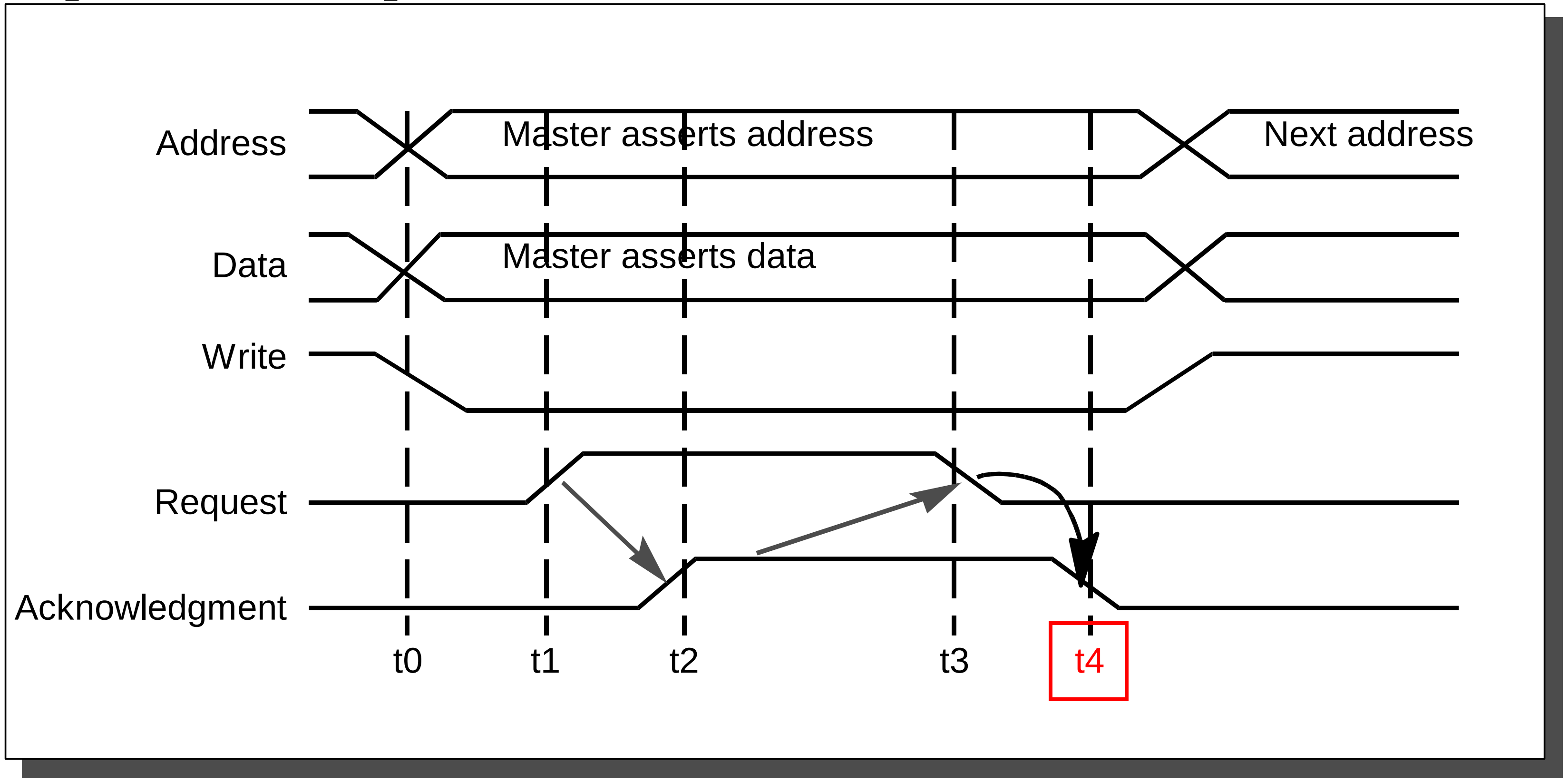
I/O Buses

Asynchronous write: master deasserts request and expects the acknowledgement to be deasserted later.



I/O Buses

Asynchronous write: slave deasserts acknowledgement and operation completes.



I/O Bus Examples

Multiple master I/O buses:

	Sun-S bus	IBM MicroChannel	PCI	SCSI 2
Data width	32 bits	32 bits	32 to 64 bits	8 to 16 bits
Clock rate	16 to 25 MHZ	Asynchronous	33 MHZ	10 MHZ or Asynch.
32-bit reads bandwidth	33 MB/Sec	20 MB/Sec	33 MB/Sec	20 MB/sec or 6 MB/Sec
Peak Bandwidth	89 MB/Sec	75 MB/Sec	132 MB/Sec	20 MB/sec or 6 MB/Sec

I/O Bus Examples

Multiple master CPU-memory buses:

	HP Summit	**SGI Challenge**	**Sun XDBus**
Data width	128 bits	256 bits	144 bits
Clock rate	60 MHZ	48 MHZ	66 MHZ
Peak Bandwidth	960 MB/Sec	1200 MB/Sec	1056 MB/Sec