7.4 I/O Architectures

- Input/output is a subsystem that moves coded data between external devices and a host system. The host consists of a CPU and main memory.
- I/O subsystems include:
  - Blocks of main memory that are devoted to I/O functions
  - Buses that provide the means of moving data into and out of the system
  - Control modules in the host and in peripheral devices
  - Interfaces to external components such as keyboards and disks
  - Cabling or communications links between the host system and its peripherals

- **Protocol** defines the signals by which sender and receiver effect a transfer of data.
- A **handshake** is the process in which the receiver acknowledges a command from the sender indicating that it is ready to receive data.
- Disk and tape are forms of **durable storage**. Durable storage is named so because of the volatile nature of electronic memory. The expected life of magnetic media is 5 to 30 years and for optical media, 100 years.

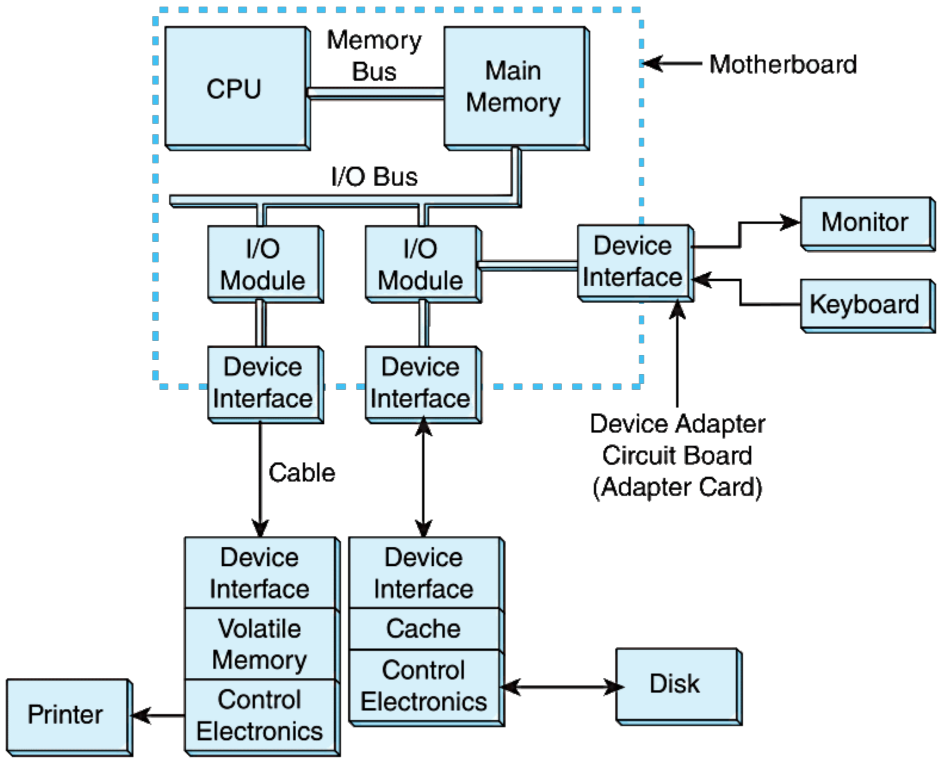


Figure 7.1 A Model I/O Configuration

7.4.1 I/O Control Methods

Programmed I/O

- Simplest way for a CPU to communicate with an I/O device.
- Sometimes call **polled I/O** (or **port I/O**)
- The CPU continually monitors (polls) a control register associated with each I/O port. When a byte arrives in the port, a bit is the control register is also set. The CPU eventually polls the port and notices that the "data ready" control bit is set. The CPU resets the control bit, retrieves the byte, and processes it according to instructions programmed for that particular port.
- Assets
    - Programmatic control over the behavior of each device.
    - Easily modifiable. By modifying a few lines of code, we can adjust the number and types of devices in the system, as well as their polling priorities and intervals.
- Liabilities
    - The CPU is denied more useful work because it is occupied in a continual "busy wait" loop until it starts servicing an I/O request.
    - The frequency that a device is polled is problem. One device may require a much higher frequency than another. With a single polling loop, only one frequency is available.

- If we decide to use programmed I/O, it is to our advantage to establish separate buses for memory traffic and I/O traffic so that the continued polling doesn't interfere with memory access.

Interrupt-Driven I/O

- Converse of programmed I/O
- Devices tell the CPU when they have data to send.
- The CPU proceeds with other tasks until a device requesting service sends an interrupt to the CPU. These interrupts are typically generated for every word of information that is transferred.
- In most interrupt-driven I/O implementations, this communication takes place through an intermediary interrupt controller. The circuit handles interrupt signals all I/O devices in the system. Once this circuit recognizes an interrupt signal from any of its attached devices, it raises a signal interrupt signal that activates a control line on the system bus.

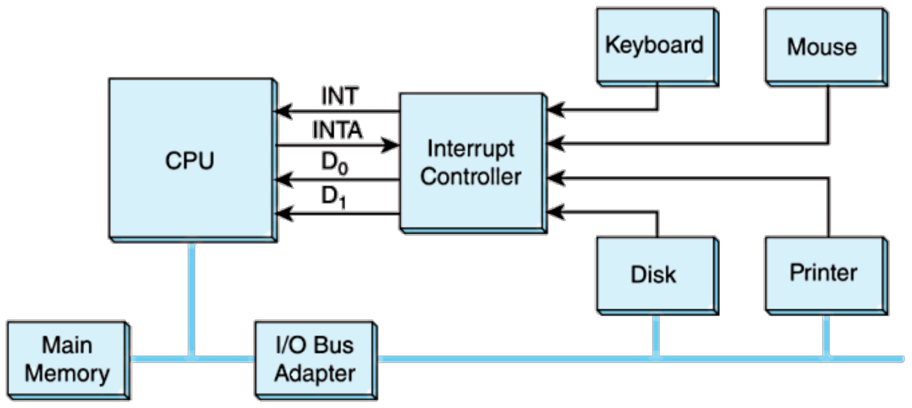


Figure 7.2 An I/O Subsystem Using Interrupts

Whenever an interrupt line is asserted, the controller decodes the interrupt and raises the `Interrupt (INT)` input on the CPU. When the CPU is ready to process the interrupt, it asserts the `Interrupt Acknowledge (INTA)` signal. Once the interrupt controller gets this acknowledgement it can lower its `INT` signal.

- Whey two or more I/O interrupts occur simultaneously, the interrupt controller determines which one should take precedence, based on the time-criticality of the device requesting the I/O. Keyboard and mouse I/O are usually the least critical.

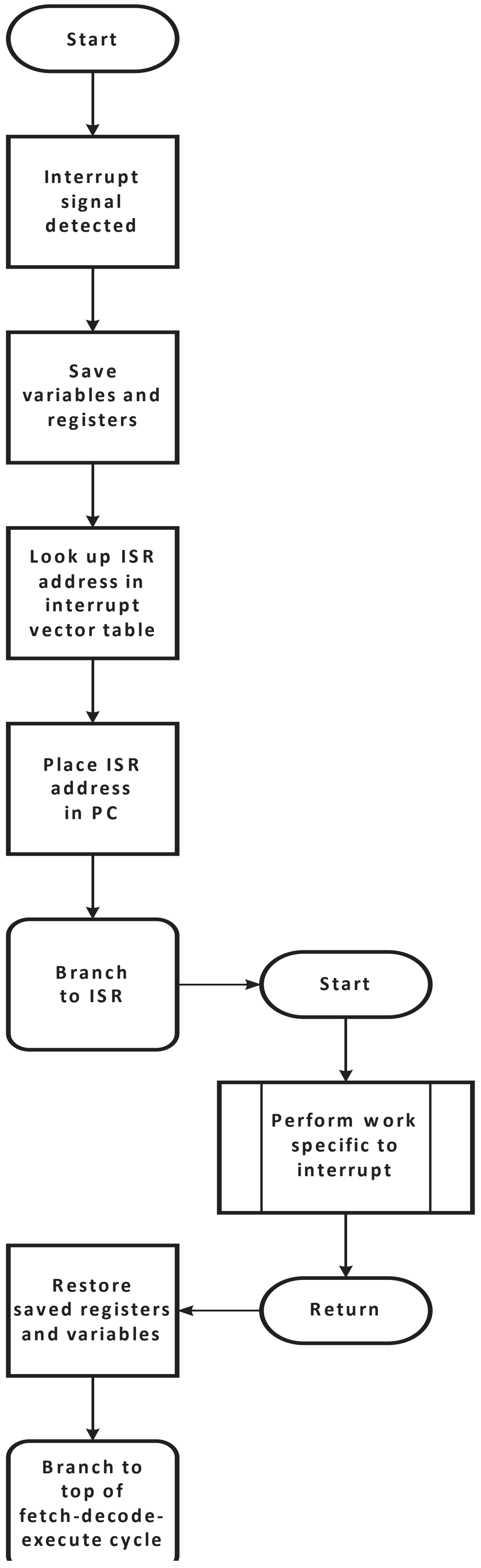

Start
Interrupt signal detected
Save variables and registers
Look up ISR address in interrupt vector table
Place ISR address in PC
Branch to ISR
Start
Perform work specific to interrupt
Return
Restore saved registers and variables
Branch to top of fetch-decode-execute cycle

Figure 7.4 Processing and Interrupt

Memory-Mapped I/O

- I/O devices and main memory share the same address space.
- Each I/O device has its own reserved block of memory.
- Data transfers to and from the I/O device involve moving bytes to or from the memory address that is mapped to the device.
- Memory-mapped I/O looks just like a memory access from the point of view of the CPU.

Direct Memory Access

- Both programmed I/O and interrupt-driven I/O move data to and from the I/O device employing code similar to:

```
while more-input and not (error or timeout)
    add 1 to byte-count
    if byte-count > total-bytes-to-be-transferred then exit endif
    place byte in destination buffer
    raise byte-ready signal
    initialize timer
    repeat
        wait
    until byte-acknowledged, timeout, or error
endwhile
```

- The foregoing instructions are simple enough to be programmed in a dedicated chip. This is the idea behind **direct memory access (DMA)**.
- The CPU provides the DMA controller with the location of the bytes to be transferred, the number of bytes to be transferred, and the destination device or memory address.

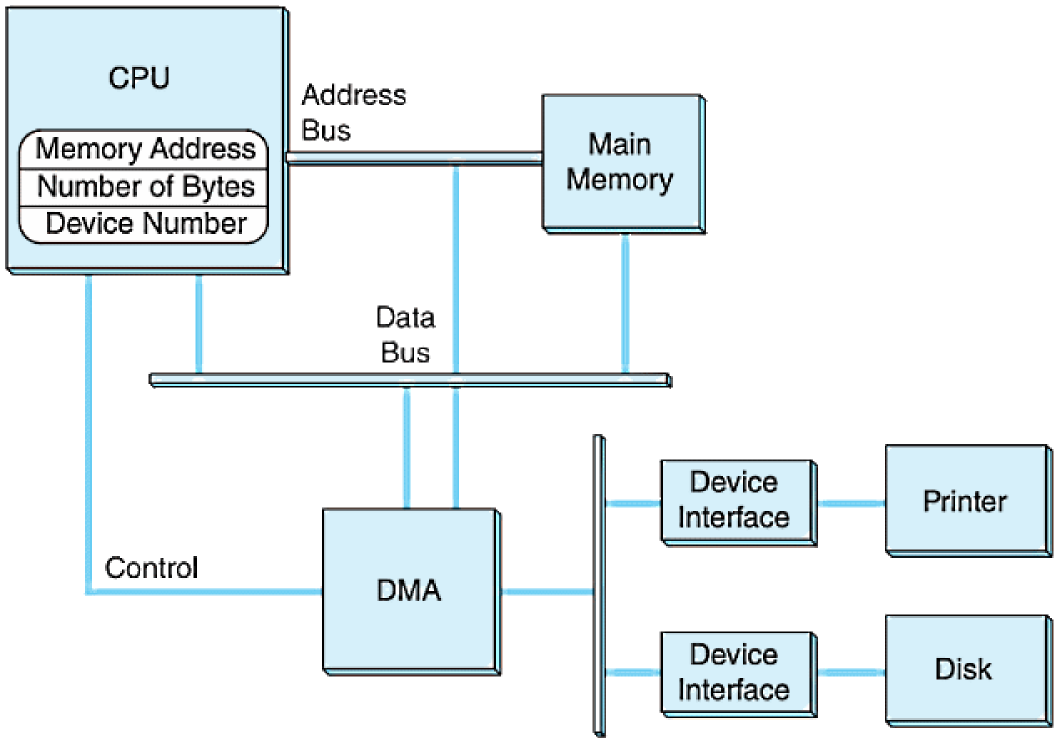


Figure 7.5 A Sample DMA Configuration

- The DMA controller and the CPU share the bus.
- Only one of them at a time can control the bus, that is, be the **bus master**.
- Generally, I/O takes priority over CPU memory fetches for program instructions and data because many I/O devices operate within tight timing parameters. If they detect no activity within a specified period, they **timeout** and abort the I/O process.
- To avoid device timeouts, the DMA uses memory cycles that would otherwise be used by the CPU. This is called cycle stealing.
- Fortunately, I/O tends to create **bursty** traffic on the bus: data are sent in blocks, or clusters.

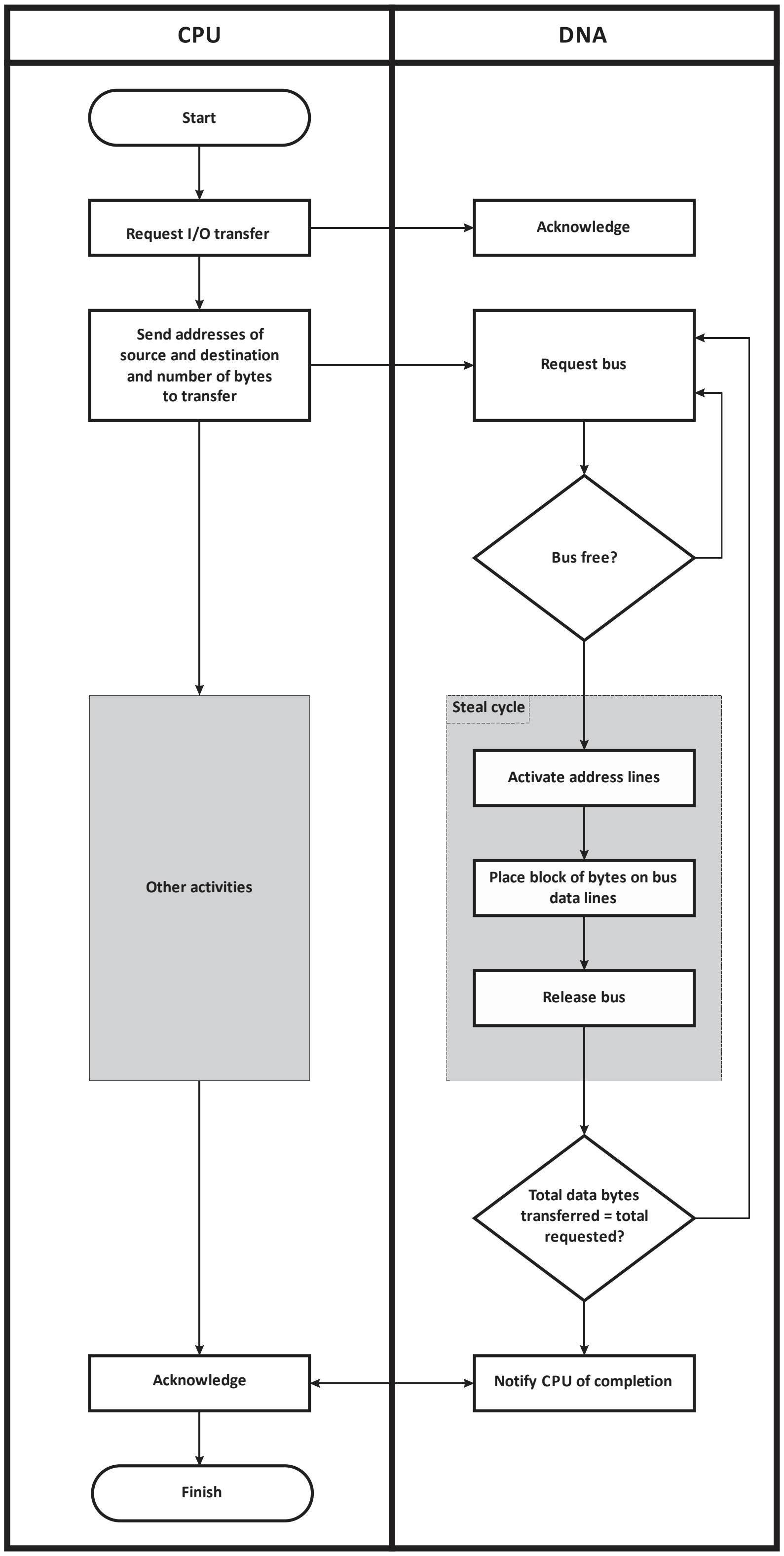


Figure 7.6 Swimlane Diagram Showing the Interaction of a CPU and a DMA

Channel I/O

- An **I/O Channel** is an intelligent DMA device typically employed on mainframe (IBM) computers.
- One or more I/O processors control various I/O pathways called **channel paths**. Channel paths for “slow” devices such as terminals and printers can be combined (**multiplexed**), allowing management of several of these devices through one controller.
- On IBM mainframes, a multiplexed channel path is called a **multiplexor channel**. Channels for disk drives and other “fast” devices are called **selector channels**.
- I/O Processors execute programs that are placed in main system memory by the host processor. These programs, consisting of a series of **channel command words** (**CCWs**), include not only the actual transfer instructions but also commands that control the I/O devices.
- Like standalone DMA, an I/O processor must steal memory cycles from the CPU. Unlike standalone DMA, channel I/O systems are equipped with separate I/O buses, which help to isolate the host from the I/O operation.
- Channel I/O is a type of isolated I/O. When copying a file from disk to tape, for example, the IOP uses the system memory bus only to fetch its instructions from main memory.

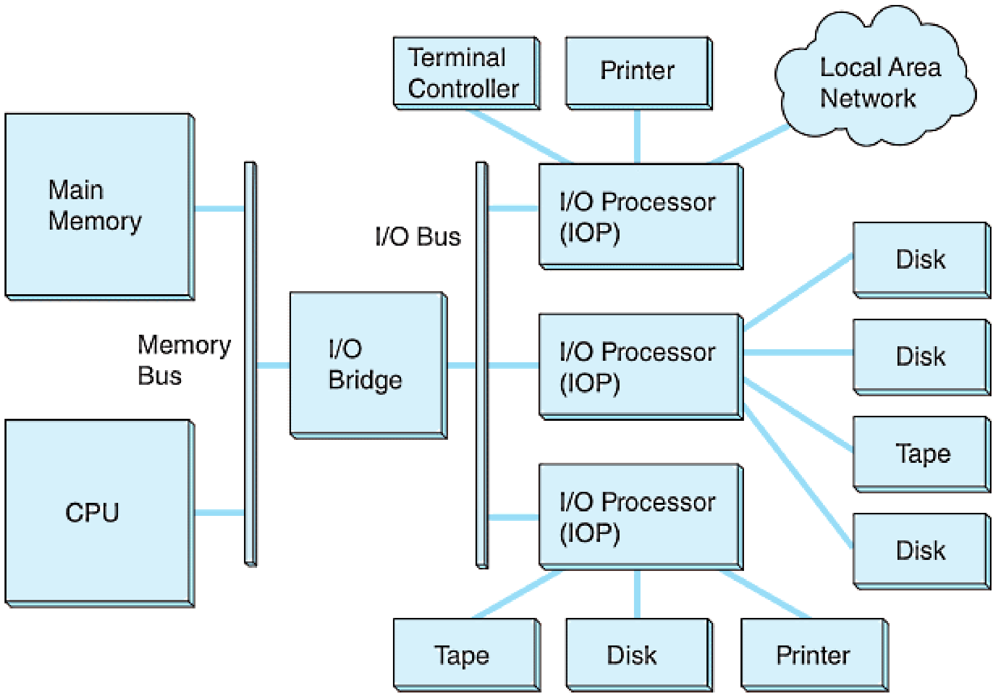
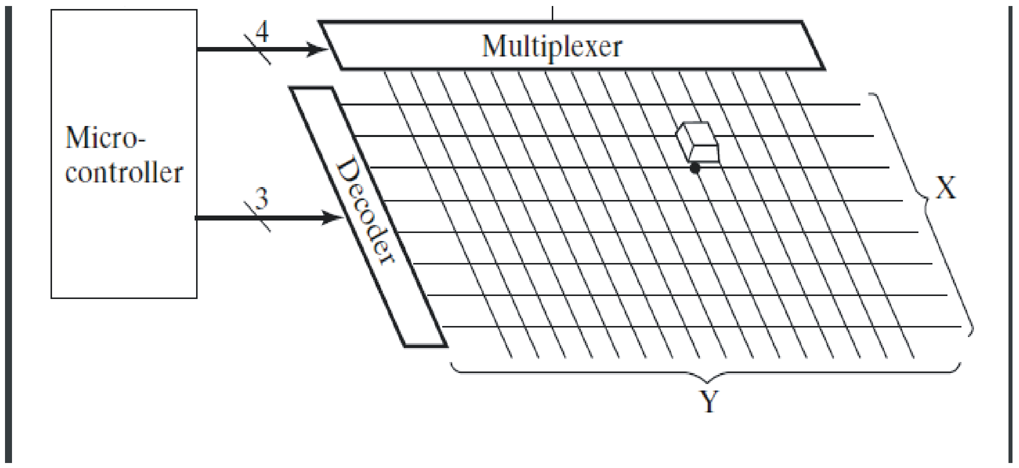
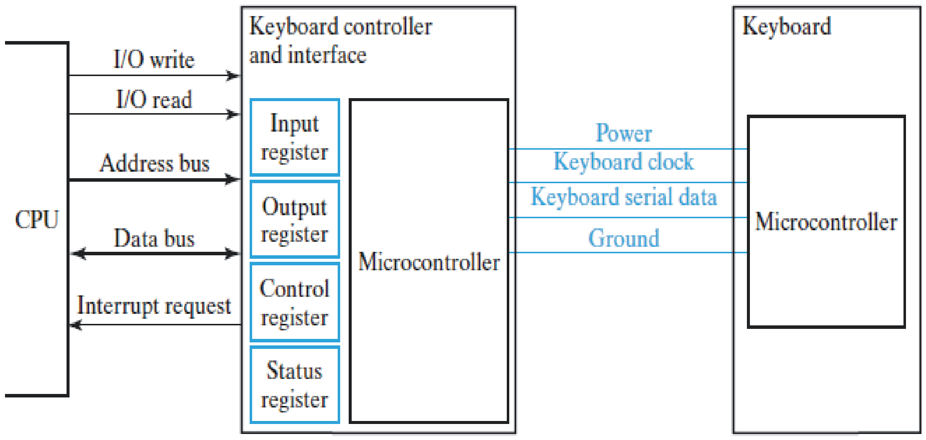


Figure 7.7 A Channel I/O Configuration

7.4.2 Character I/O Versus Block I/O

- Keyboard input
    1. When a key is depressed, a switch is closed. The switch is activates a **scan code** that reflects a row and column in a matrix.
    2. The scan code is then passed to a serial interface circuit. The serial interface circuit translates the scan code to a character code.
    3. The interface places the character code in a keyboard buffer that is maintained in low memory.
    4. An I/O interrupt signal is raised immediately after placing the character code in low memory.
    5. Characters are stored in the buffer until they are retrieved, one at a time, by a program (or until the buffer is reset)





- Magnetic disks and tapes
  - o Store data in blocks
  - o Transactions are performed in blocks rather than as single characters.
  - o Block I/O lends itself to DMA or channel I/O processing.

7.4.3 I/O Bus Operation

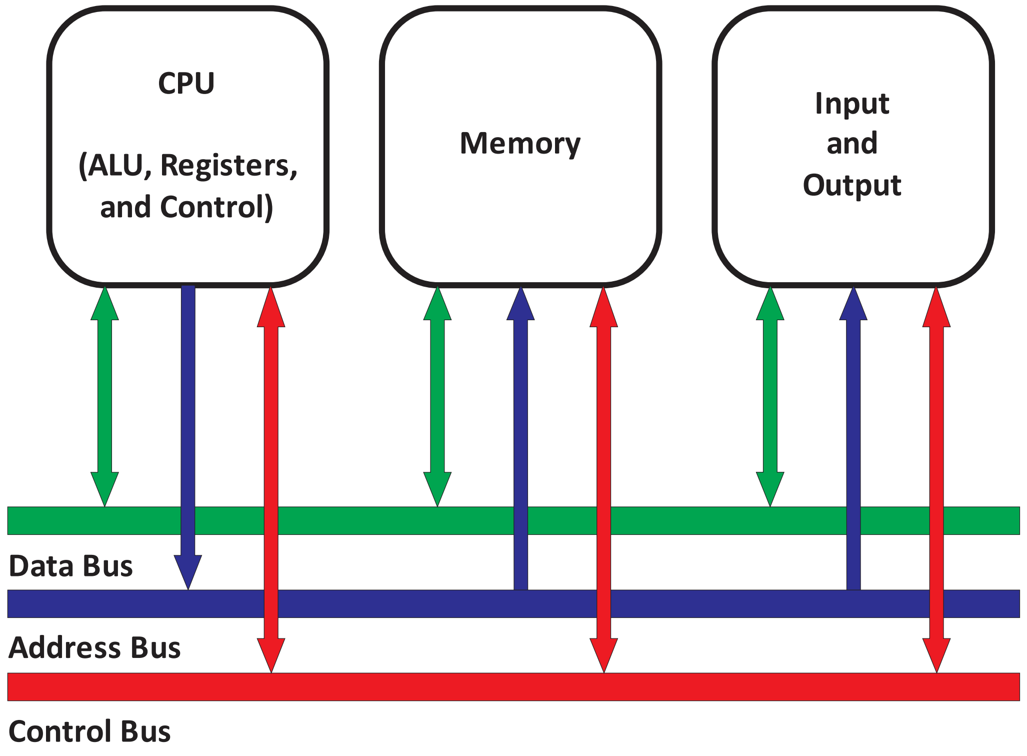


Figure 7.8 High-Level View of a System Bus

- A system bus is a resource shared among many components of a computer system.
- Access to this shared resource must be controlled. This is why a control bus is required.

- The memory bus and the I/O bus can be separate.
- Memory transfers can be **synchronous**, using some multiple of the CPU's clock cycle to retrieve data from main memory.
- I/O buses cannot operate synchronously. The must take into account the fact that I/O devices cannot always be ready to process an I/O transfer. I/O control circuits placed on the I/O bus and within the I/O devices negotiate with each other to determine the moment when each device may use the bus. Because these handshakes take place every time the bus is accessed, I/O buses are called **asynchronous**.

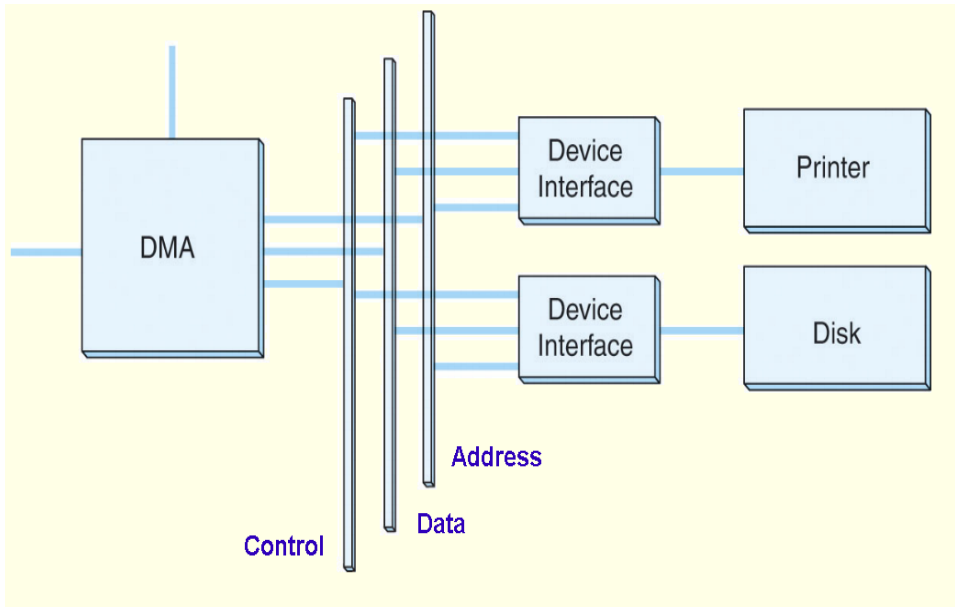


Figure 7.9 DMA Configuration Showing Separate Address, Data, and Control Lines

- Address and data buses consist of a number of individual conductors, each of which carries one bit of information.
- The number of data lines determines the **width** of the bus. A bus having eight data lines carries one byte at a time.

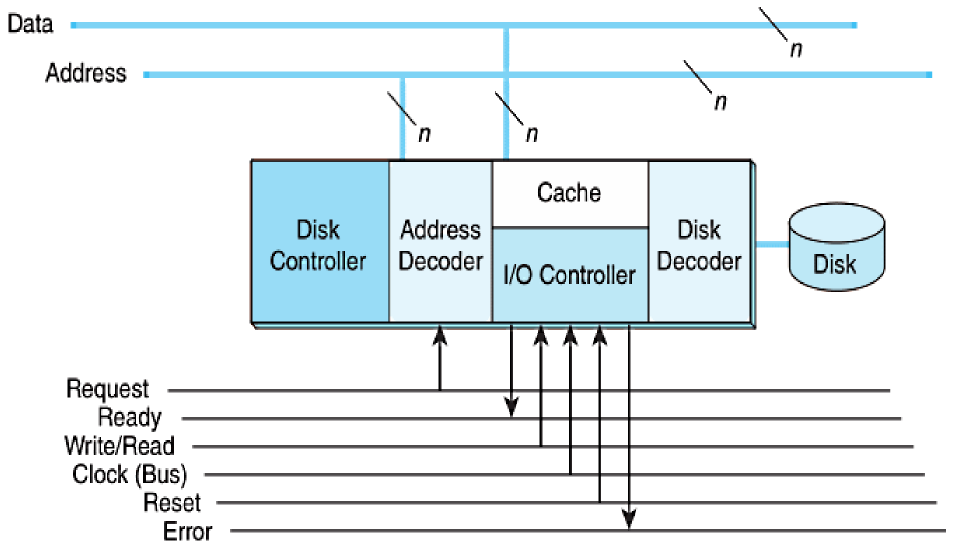


Figure 7.10 Disk Controller Interface with Connections to the I/O Bus

1. The DMA circuit places the address of the disk controller on the address lines and raises (asserts) the `Request` and `Write` signals.
2. With the `Request` signal asserted, decoder circuits in the controller interrogate the address lines.
3. Upon sensing its own address, the decoder enables the disk control circuits. If the disk is available for writing data, the controller asserts a signal on the `Ready` line.
4. The DMA circuits then place the data on the lines and lower the `Request` signal.
5. When the disk controller sees the `Request` signal drop, it transfers the byte from the data lines to the disk buffer, and then lowers its `Ready` signal.

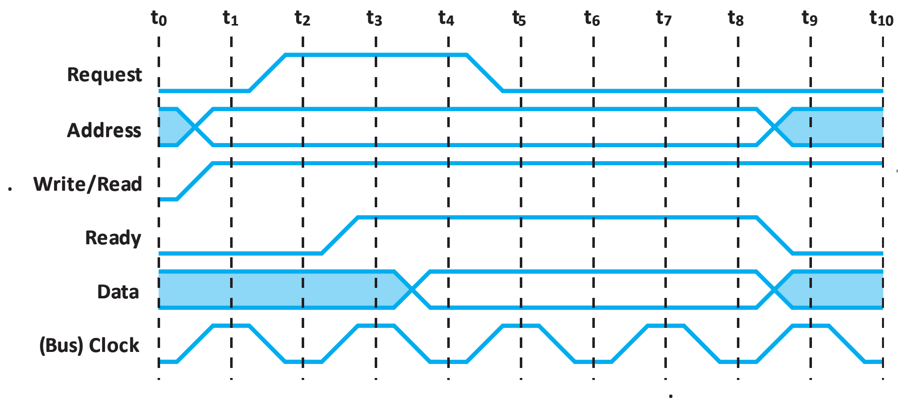


Figure 7.11 (a) A Bus Timing Diagram for Disk Write

Time	Salient Bus Signal	Meaning
t_0	Assert Write	Bus is needed for writing (not reading)
t_0	Assert Address	Indicates where bytes will be written
t_1	Assert Request	Request write to address on address lines
t_2	Assert Ready	Acknowledge write request, bytes placed on data lines
$t_3 - t_7$	Data Lines	Write data (requires several cycles)
t_8	Lower Ready	Release bus

Figure 7.11 (b) Bus Timing Legend for Disk Write

- The vertical lines, marked t_0 through t_{10}, specify the duration of the various signals. In a real timing diagram, an exact duration would be assigned to the timing intervals, usually in the neighborhood of 50ns (in 2014).
- A small amount of time must be allowed for the signal level to stabilize, or "settle down." This **settle time**, although small, contributes to a large delay over long I/O transfers.