INPUT/OUTPUT ORGANIZATION

Carl Hamacher, Zvonko Vranesic and Safwat Zaky, Computer Organization and Embedded Systems, (6e), McGraw Hill Publication, 2017.

Ch 3: 3.1, 3.1.1, 3.1.2

Accessing I/O Devices

- The components of a computer system communicate with each other through an interconnection network, as shown in Figure 3.1.
- The interconnection network consists of circuits needed to transfer information between the processor, the memory unit, and a number of I/O devices.
- Multiple I/O devices may be connected to the processor and the memory via a bus.
- Bus consists of three sets of lines to carry address, data and control signals.
- Each I/O device is assigned a unique address.
- To access an I/O device, the processor places the address on the address lines.
- The device recognizes the address and responds to the control signals.

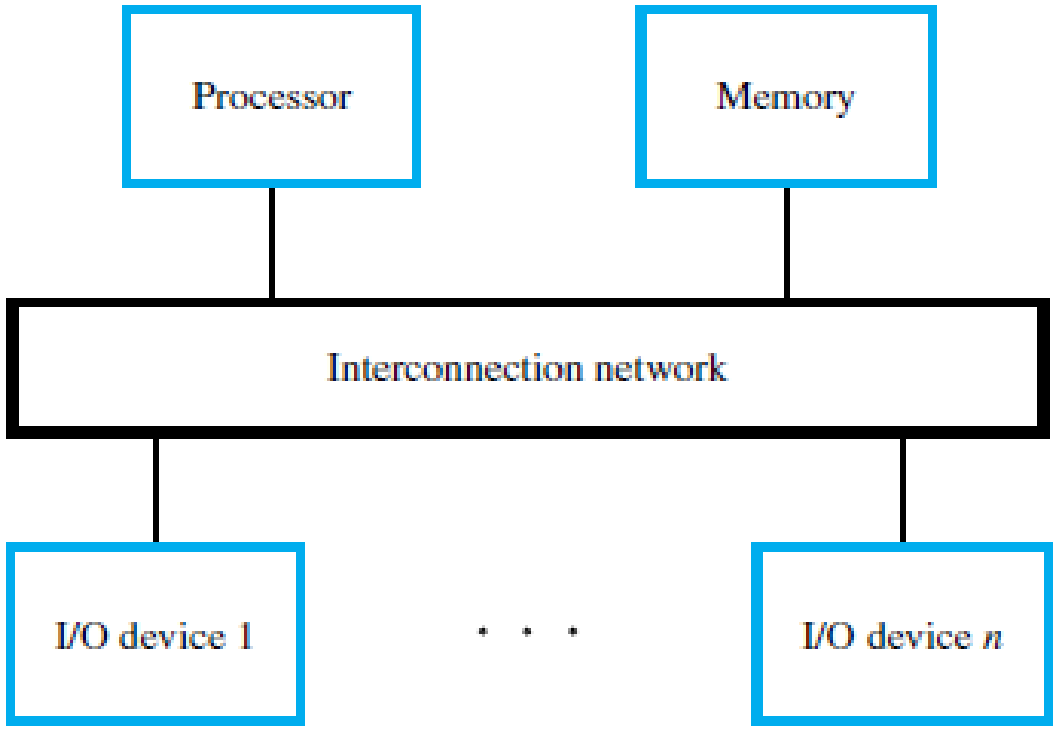


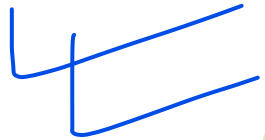
Figure 3.1 A computer system.

Accessing I/O Devices - ASSIGNMENT OF ADDRESS

- This idea of using addresses to access various locations in the memory can be extended to deal with the I/O devices as well.
- For this purpose, each I/O device must appear to the processor as consisting of some addressable locations, just like the memory.
- Some addresses in the address space of the processor are assigned to these I/O locations, rather than to the main memory.
- These locations are usually implemented as bit storage circuits (flip-flops) organized in the form of registers.
- It is customary to refer to them as I/O registers.
- Since the I/O devices and the memory share the same address space, this
- arrangement is called **memory-mapped I/O.**

Accessing I/O Devices - ASSIGNMENT OF ADDRESS

- Any machine instruction that can access memory can be used to transfer data to or from an I/O device.
- Load R2, DATAIN
- Store R2, DATAOUT
- I/O devices and the memory may have different address spaces **I/O mapped I/O**:
  - Special instructions to transfer data to and from I/O devices.
  - I/O devices may have to deal with fewer address lines.
  - I/O address lines need not be physically separate from memory address lines.
  - In fact, address lines may be shared between I/O devices and memory, with a control signal to indicate whether it is a memory address or an I/O address.



Accessing I/O Devices - I/O Device Interface

- An I/O device is connected to the interconnection network by using a circuit, called the *device interface*,
- This provides the means for data transfer and for the exchange of status and control information needed to facilitate the data transfers and govern the operation of the device.
- The interface includes some registers that can be accessed by the processor.
  - One register may serve as a buffer for data transfers
  - another may hold information about the current status of the device,
  - and yet another may store the information that controls the operational behavior of the device.
- These *data*, *status*, and *control* registers are accessed by program instructions as if they were memory locations.
- Typical transfers of information are between I/O registers and the registers in the processor

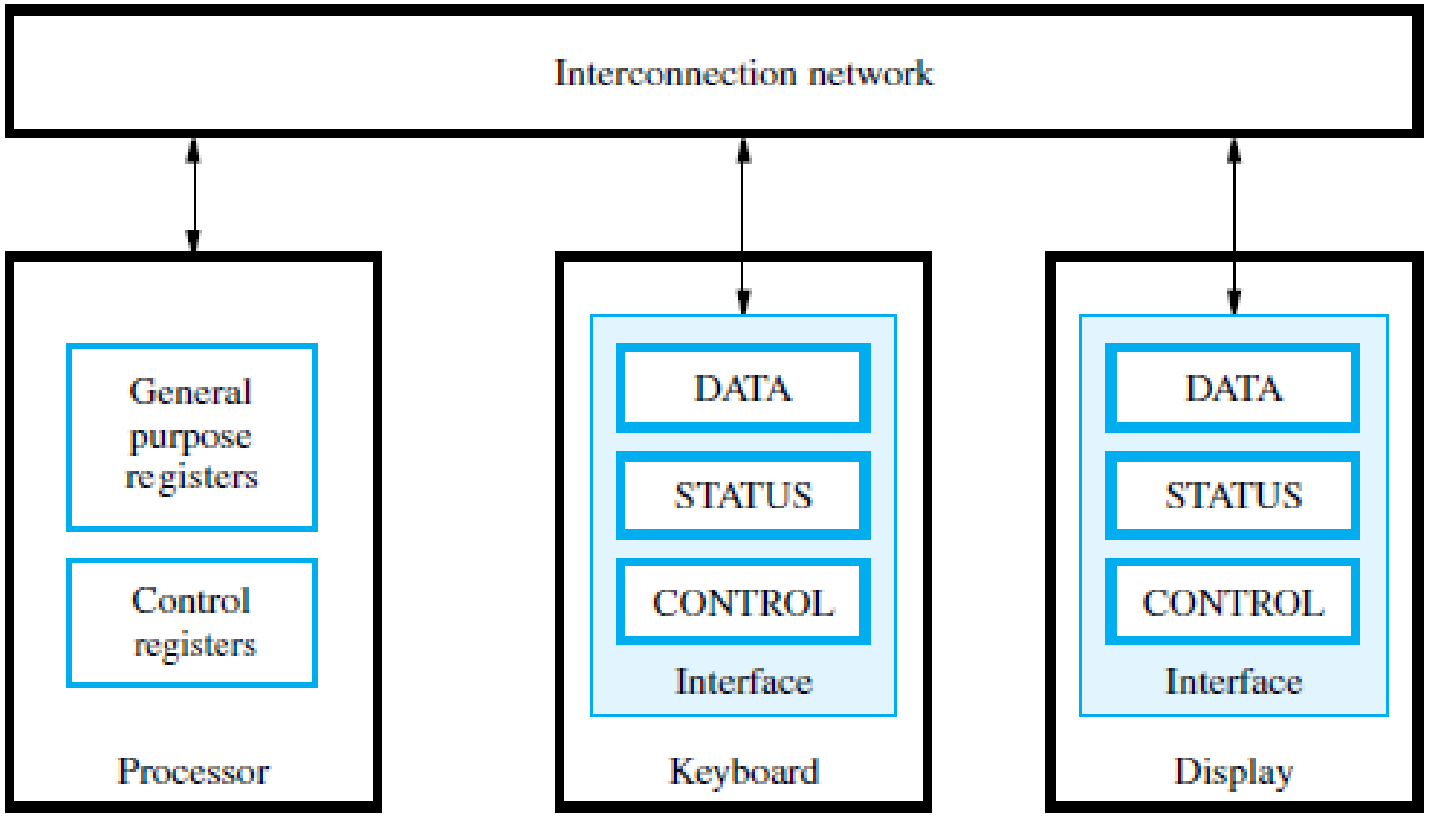


Figure 3.2 The connection for processor, keyboard, and display.

Accessing I/O Devices-PROGRAMMED Controlled I/O

- Program-controlled I/O
  - Consider a task that reads characters typed on a keyboard, stores these data in the memory, and displays the same characters on a display screen. A simple way of implementing this task is to write a program that performs all functions needed to realize the desired action. This method is known as **program-controlled I/O**.
  - Processor repeatedly monitors a status flag to achieve the necessary synchronization.
  - Processor polls the I/O device.
- Two other mechanisms used for synchronizing data transfers between the processor and memory:
  - Interrupts.
  - Direct Memory Access.

Interrupts

Carl Hamacher, Zvonko Vranesic and Safwat Zaky, Computer Organization and Embedded Systems, (6e), McGraw Hill Publication, 2017.

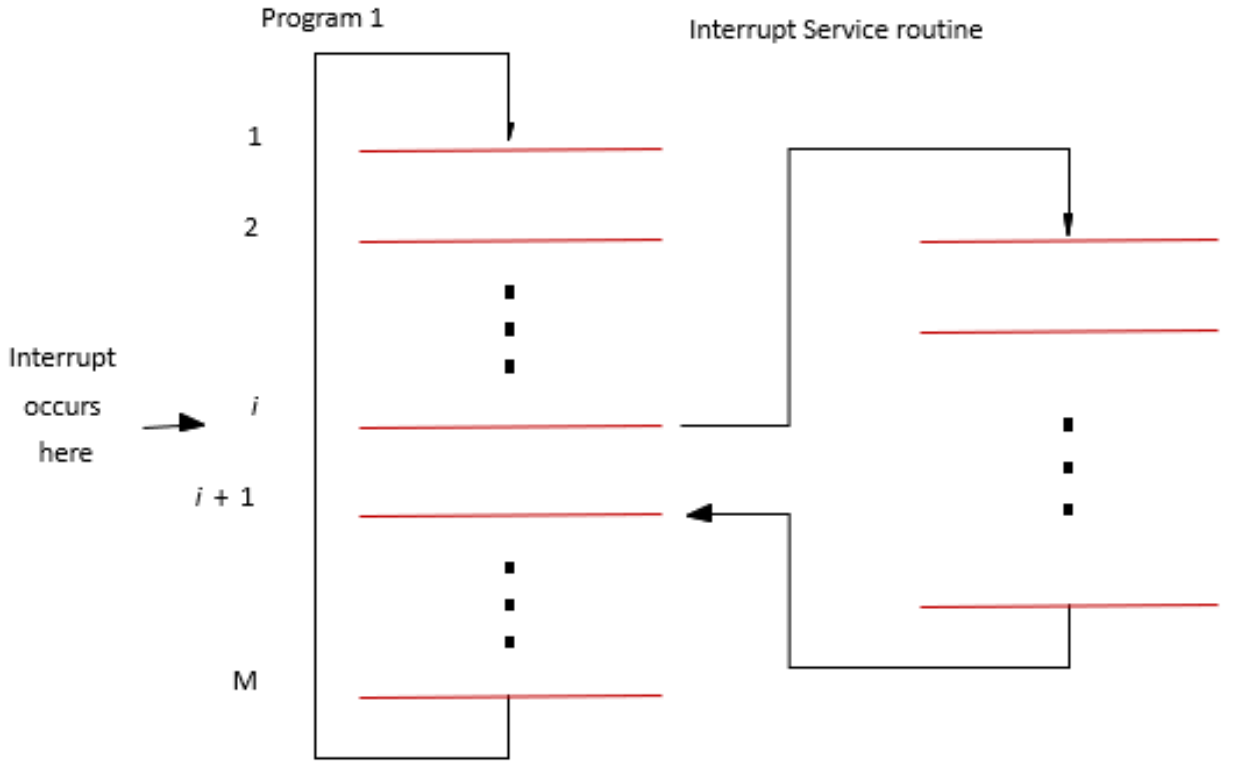
Ch 3:3.2, 3.2.1

Interrupts

- In program-controlled I/O, when the processor continuously monitors the status of the device, it does not perform any useful tasks.
- An alternate approach would be for the I/O device to alert the processor when it becomes ready.
- Do so by sending a hardware signal called an **interrupt request** to the processor.
- Since the processor is no longer required to continuously poll the status of I/O devices, it can use the waiting period to perform other useful tasks.
- Indeed, by using interrupts, such waiting periods can ideally be eliminated.

Interrupts

- Processor is executing the instruction located at address *i* when an interrupt occurs.
- Routine executed in response to an interrupt request is called the **interrupt-service routine**.
- When an interrupt occurs, control must be transferred to the ISR.
- But before transferring control, the current contents of the PC (*i+1*), must be saved in a known location.
- This will enable the return-from-interrupt instruction to resume execution at *i+1*.
- Return address, or the contents of the PC are usually stored on the processor stack.



Interrupts – Acknowledgement by the processor

- Processor must inform the device that its request has been recognized so that it may remove its interrupt-request signal.
  - accomplished through interrupt acknowledge signal, which is sent to the device through the interconnection network.
  - When the processor executes ISR, it is implicit that the interrupt request has been recognized

PIPELINING

Carl Hamacher, Zvonko Vranesic and Safwat Zaky, Computer Organization and Embedded Systems, (6e), McGraw Hill Publication, 2017.
Ch 6: 6.1, 6.2

Pipelining: Overview

- The speed of execution of programs is influenced by many factors.
- One way to improve performance is to use faster circuit technology to implement the processor and the main memory.
- Another possibility is to arrange the hardware so that more than one operation can be performed at the same time.
- In this way, the number of operations performed per second is increased, even though the time needed to perform any one operation is not changed.
- Pipelining is a particularly effective way of organizing concurrent activity in a computer system.
- Pipelining as a means for improving performance by overlapping the execution of
- machine instructions

Pipelining: Basic Concepts

- Pipelined Execution:

- Pipelining is widely used in modern processors.
- Pipelining improves system performance in terms of throughput.
- Pipelined organization requires sophisticated compilation techniques.

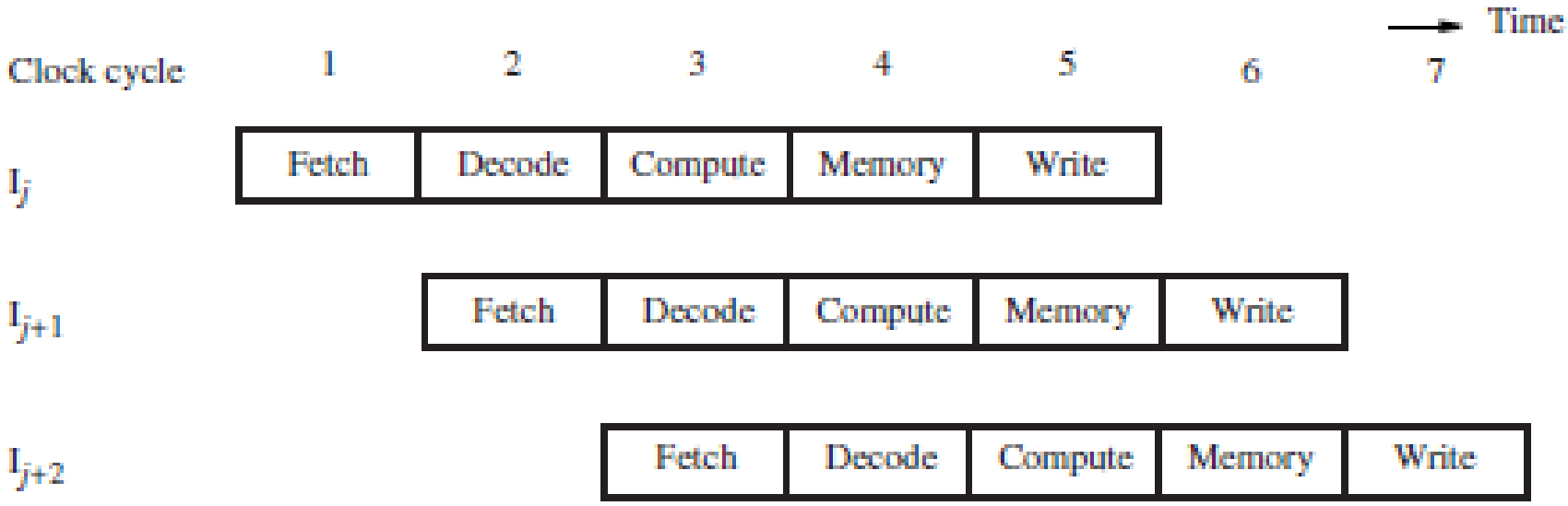


Figure 6.1 Pipelined execution—the ideal case.

Pipeline Organization

- The interstage buffers are used as follows:
  - Interstage buffer B1 feeds the Decode stage with a newly-fetched instruction.
  - Interstage buffer B2 feeds the Compute stage with the operands read from the register file
  - Interstage buffer B3 holds the result of the ALU operation, which may be data to be written into the register file or an address that feeds the Memory stage. In case of a write access to memory, buffer B3 holds the data to be written
  - Interstage buffer B4 feeds the Write stage with a value to be written into the register file.

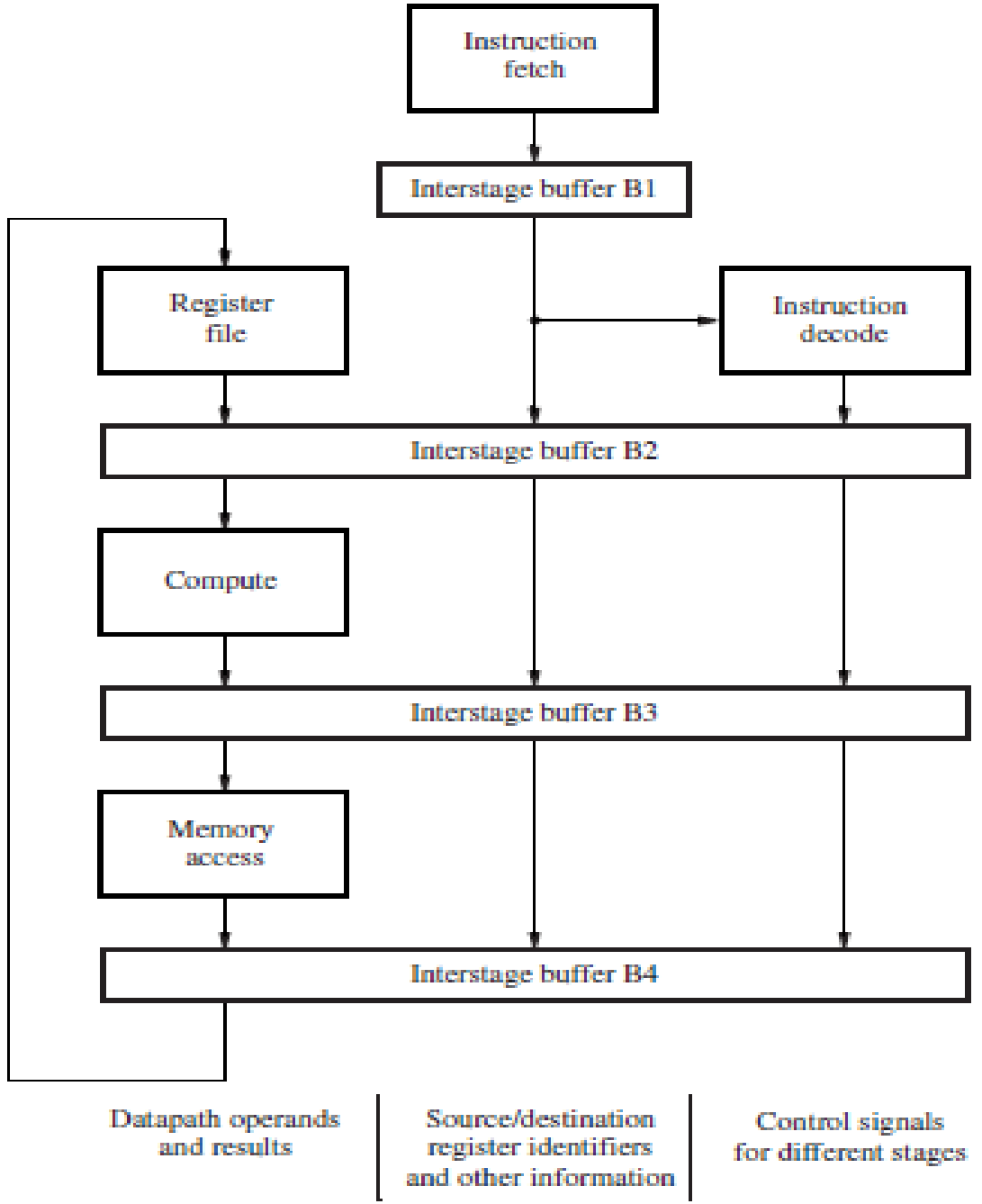


Figure 6.2 A five-stage pipeline.