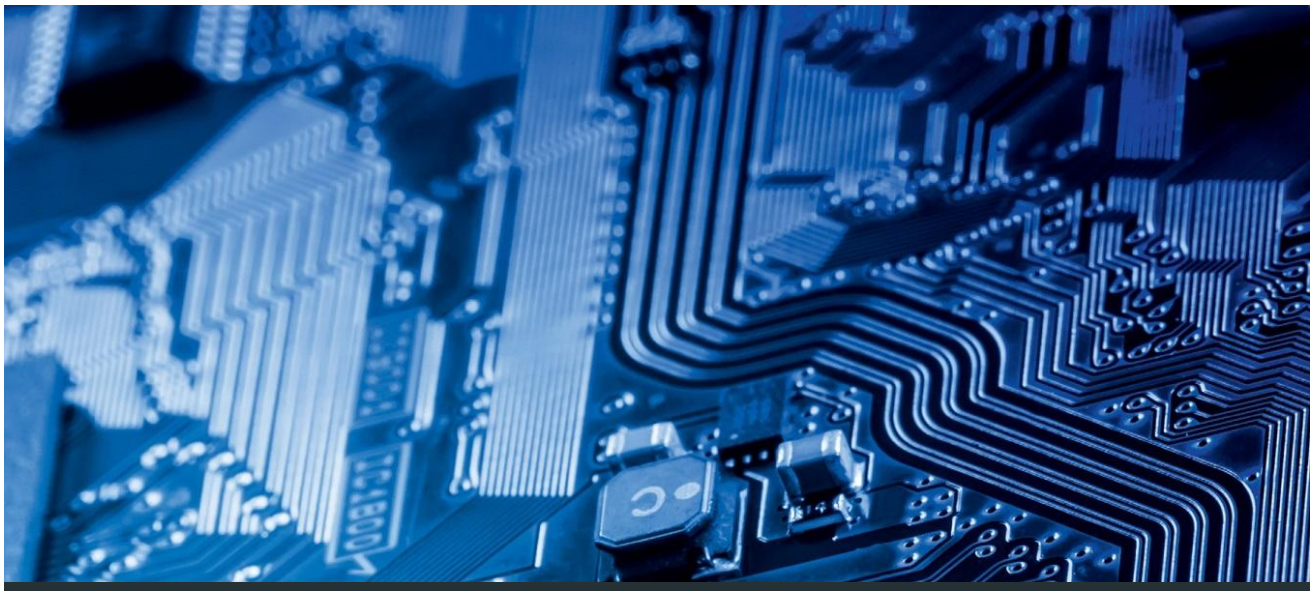
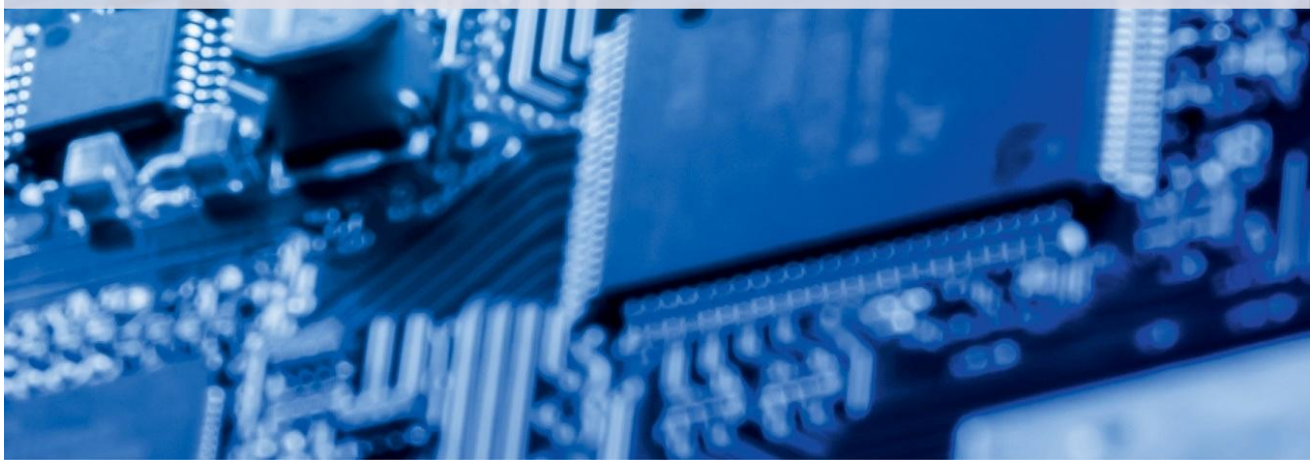


Chapter 5:
Processor fundamentals

Learning objectives

By the end of this chapter you should be able to:

- show understanding of the basic Von Neumann model for a computer system and the stored program concept
- show understanding of the purpose and role of registers, including the difference between general purpose and special purpose registers
- show understanding of the purpose and roles of the Arithmetic and Logic Unit (ALU), Control Unit (CU), system clock and Immediate Access Store (IAS)
- show understanding of how data are transferred between various components of the computer system using the address bus, data bus and control bus
- show understanding of how factors contribute to the performance of the computer system
- understand how different ports provide connection to peripheral devices
- describe the stages of the fetch-execute (F-E) cycle
- show understanding of the purpose of interrupts.



5.01 The von Neumann model of a computer system

John von Neumann was the first person to describe the basic principles of a computer system and its architecture in a publication.

The model von Neumann described has the following basic features.

- There is a processor - the central processing unit (CPU).
- The processor has direct access to memory.
- The memory contains a ‘stored program’ (which can be replaced by another at any time) and the data required by the program.
- The stored program consists of individual instructions.
- The processor executes instructions sequentially.

5.02 Central processing unit (CPU) architecture

In order to understand how the von Neumann model could be put into practice in a real computer system, we need first to identify the individual hardware components of a CPU and define their functions. Let's consider a system that has the minimum number of components needed. Figure 5.01 gives a simplified schematic diagram of a processor.

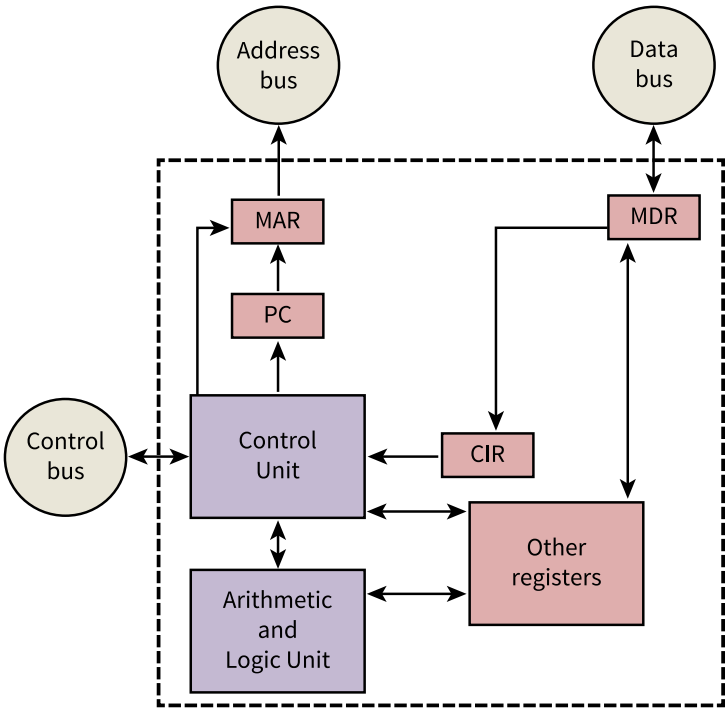


Figure 5.01 A schematic diagram of the architecture of a simple CPU

The dotted outline shows the boundary of the processor. The logical arrangement of some of the processor components is indicated. The arrows show possible directions of flow of data. As the following discussion will show, the data for some of the arrows is actually an address or an instruction. However, in general, data might be an address, an instruction or a value.

The active components of the CPU

The two components of the CPU that have an active role in its operation are the arithmetic and logic unit (ALU) (or Arithmetic Logic Unit) and the control unit. As its name implies, the ALU is responsible for the arithmetic or logic processing requirements of the instructions in a running program. The functions of the control unit are more diverse. One aspect is controlling the flow of data throughout the processor and the rest of the whole computer system. Another is ensuring that program instructions are handled correctly. A vital part of the control unit is a clock that is used by the unit to synchronise processes. Strictly speaking there are two clocks. The first is an **internal clock** that controls the cycles of activity within the processor. The other is the **system clock** that controls activities outside the processor. The CPU will have a defined frequency for its clock cycle, which is usually referred to as the clock speed. The frequency defines the minimum period of time that separates successive activities within the system.

Registers

The other components of the CPU are the registers. These are storage components which, because they are placed very close to the ALU, allow very short access times. Each register has limited storage capacity, typically 16, 32 or 64 bits. A register is either general purpose or special purpose. If there is only one general-purpose register it is referred to as the **Accumulator**. Here and in Chapter 6, we assume that the processor has just this one general-purpose register. The Accumulator is used to store a single value at any one time. A value is stored in the Accumulator that is to be used by the ALU for the

execution of an instruction. The ALU can then store a different value in the Accumulator after the execution of the instruction.

Figure 5.01 shows some of the special-purpose registers as individual components. The box labelled ‘other registers’ can be considered to comprise the Accumulator plus the special-purpose registers not identified individually. The full names of the special-purpose registers included in the simple CPU that we are considering are given in Table 5.01 with a brief description of their function.

Register name	Abbreviation	Register’s function
Current instruction register	CIR	Stores the current instruction while it is being decoded and executed
Index register	IX	Stores a value; only used for indexed addressing
Memory address register	MAR	Stores the address of a memory location or an I/O component which is about to have a value read from or written to
Memory data register (memory buffer register)	MDR (MBR)	Stores data that has just been read from memory or is just about to be written to memory
Program counter	PC	Stores the address of where the next instruction is to be read from
Status register	SR	Contains bits that are either set or cleared which can be referenced individually

Table 5.01 Registers in a simple CPU

There are three important points to remember. The first is that the MDR must act as a buffer. This is because transfers of data inside the processor take place much more quickly than transfers outside the processor. The second point to note is that the index register (IX) can be abbreviated as IR but in some sources the current instruction register (CIR) is abbreviated as ‘IR’. This is a potential cause of confusion. In this book, the index register is always IX and the current instruction register is CIR. Finally, there is also possible confusion if the abbreviation PC is used. This will only be used in this book when register transfer notation is being used, as you will see later in the chapter. Everywhere else, a PC is a computer.

For all of the special-purpose registers, except for the status register, the contents represent one value. For the status register each individual bit is used as a logical flag. The bit is set to 1 if a condition is detected. Examples are the carry flag, the negative flag and the overflow flag.

Chapter 6 (Section 6.07) contains some examples of the use of the accumulator and the status register.

5.03 The system bus

A bus is a parallel transmission component with each separate wire carrying a single bit. It is important not to describe a bus as a storage device. A bus does not hold data. Instead it is a mechanism for data to be transferred from one system component to another.

There will be buses inside the CPU. These are not considered here. The system bus connects the CPU to the memory and to the I/O system. In the simple computer system described in this chapter there will be a system bus that comprises three distinct components: the address bus, the data bus and the control bus. The schematic diagram of the CPU in Figure 5.01 shows the logical connection between each bus and a CPU component. The address bus is connected to the MAR; the data bus to the MDR; and the control bus to the control unit. The system bus allows data flow between the CPU, the memory and input or output (I/O) devices as shown in the schematic diagram in Figure 5.02.

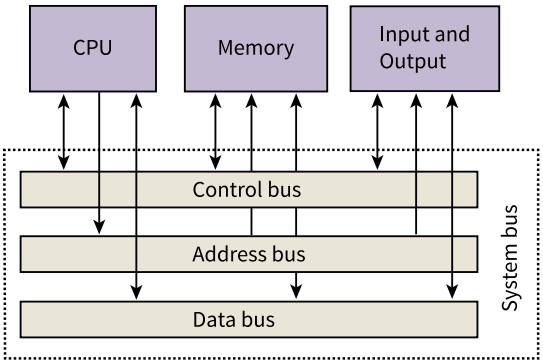


Figure 5.02 A schematic diagram of the system bus

The address bus

The sole function of the **address bus** is to carry an address. This address is loaded on to the bus from the MAR as and when directed by the control unit. The address specifies a location in memory or an I/O component which is due to receive data or from which data is to be read. The address bus is a ‘one-way street’. It can only be used to send an address to a memory controller or an I/O controller. It cannot be used to carry an address back to the CPU.

The data bus

The function of the **data bus** is to carry data. This might be an instruction, an address or a value. As can be seen from Figure 5.02, the data bus is two-way (bidirectional): it might be carrying data from the CPU to the memory or carrying data to the CPU. However, another option is to carry data to or from an I/O device. The diagram does not make clear whether, for instance, data coming from an input device is carried first to the CPU or directly to the memory. There is a good reason for this. Some computer systems will only allow input to the CPU before the data can be stored in memory. Other systems will allow direct transfer to memory.

The control bus

The control bus is another bidirectional bus which transmits a signal from the control unit to any other system component or transmits a signal to the control unit. There is no need for extended width, so the control bus typically has just eight wires. A major use of the control bus is to carry timing signals. As described in Section 5.02, the system clock in the control unit defines the clock cycle for the computer system. The control bus carries timing signals at time intervals dictated by the clock cycle. This ensures that the time that one component transmits data is synchronised with the time that another component reads it.

5.04 Factors contributing to system performance

The processor clock speed is a very important factor governing the processing speed of the system. This is because one clock cycle defines the shortest possible time that any action can take. Actually, none of the components outside of the processor can work anywhere near as fast as the processor can. The components that are directly addressable by the processor, which can be referred to as the immediate access store (IAS), can only accept data from or provide data to the processor at speeds much slower than the processor speed.

Because of this problem modern processors are far more complex than the simple example that has been discussed in this chapter. One example of this complexity is that the CPU chip or integrated circuit will be multi-core. Each core is a separate processor. Performance improves with increasing number of cores. A further factor is the use of cache memory which was briefly discussed in Chapter 3 (Sections 3.01 and 3.03). Cache memory is the fastest component of the IAS. Performance improves with increased storage size for the cache and with increased rate of access. Fastest access is obtained by having all or part of the cache on the CPU chip.

Before considering other factors, it is useful to introduce the concept of a **word**. A word consists of a number of bytes and for any system the word length is defined. The significance of the word length is that it defines a grouping that the system can handle as one unit. The word length might be stated as a number of bytes or as a number of bits. Typical word lengths are 16, 32 or 64 bits; that is, 2, 4 or 8 bytes, respectively. The word length will influence the system architecture design in regard to the capacity of the components. For example, it is usual for the size of registers to match the word length. Word length also has to be considered when making decisions about bus widths.

For the address bus, the bus width defines the number of bits in the address's binary code. In a very simple computer system the bus width might be 16 bits, allowing 65 536 memory locations to be directly addressed. Such a memory size would, of course, be totally inadequate for a modern computer system. Even doubling the address bus width to 32 bits would only allow direct addressing of a little over four billion addresses. As a result, special techniques are used when the storage capacity of the memory is too large for direct addressing. Their use affects system performance.

Bus width is again an important factor in considering how the data bus is used. For a given computer system, the data bus width is ideally the same as the word length. If this is not possible, the bus width can be half the word length so that a full word can be transmitted by two consecutive data transfers. Clearly the performance of the system is affected if the latter case applies.

Extension Question 5.01

In an advertisement for a laptop computer, the system is described as 4 GB, 1 TB, 1.7 GHz.

a Which three components are being referred to here?

b Calculate the minimum time period that could separate successive activities on this system.

Extension Question 5.02

Can you find out the bus widths used in the computer system you are using?

5.05 I/O ports

The schematic diagram in Figure 5.02 slightly misrepresents the system architecture because it looks as if the CPU, the memory and the I/O devices have similar access to the data and control buses. The reality is different. Each I/O device is connected to an interface called a port. Each port is connected to the I/O or device controller. This controller handles the interaction between the CPU and an I/O device. A port is described as ‘internal’ if the connected I/O device is an integral part of the computer system. An external port allows the computer user to connect a peripheral I/O device.

The Universal Serial Bus (USB)

In the early days of the PC, the process of connecting a peripheral was not something the ordinary user would try to do; it required technical expertise. The aim of the plug-and-play concept was to remove the need for technical knowledge so that any computer user could connect a peripheral and start using it straight away. The plug-and-play concept was only fully realised by the creation of the Universal Serial Bus (USB) standard. Nowadays anyone buying a new peripheral device will expect it to connect to a USB port. There is an alternative technology known as FireWire, but this is not so commonly used in computer systems.

TIP

Don’t forget that the USB is a bus. A USB drive stores data and is connected to a USB port which allows data to be transmitted along the bus.

The following is some information about the USB standard.

- A hierarchy of connections is supported.
- The computer is at the root of this hierarchy and can handle 127 attached devices.
- Devices can be attached while the computer is switched on and are automatically configured for use.
- The standard has evolved, with USB 3.2 being the latest version.

Discussion Point:

Carry out an investigation into storage devices that could be connected as a peripheral to a PC using the USB port.

For two representative devices find out which specific USB technology is being used and what the potential data transfer speed is. How do these speeds compare with the speed of access of a hard drive installed inside the computer?

Specialised multimedia ports

Despite the widespread use of USB ports there are some peripheral devices that require a different port, one that is specialised for the type of device. Although computer systems come packaged with a monitor for screen display there is sometimes a requirement for a second screen to be used. The connection of the second screen can be through a Video Graphics Array (VGA) port. This provides high-resolution screen display which is suitable for most display requirements. However, if the screen is needed to display a video, the VGA port is not suitable because it does not transmit the audio component.

A High Definition Multimedia Interface (HDMI) port will provide a connection to a screen and allow the transmission of high-quality video including the audio component.

5.06 The fetch–execute (F–E) cycle

The full name for this is the fetch, decode and execute cycle. This is illustrated by the flowchart in Figure 5.03.

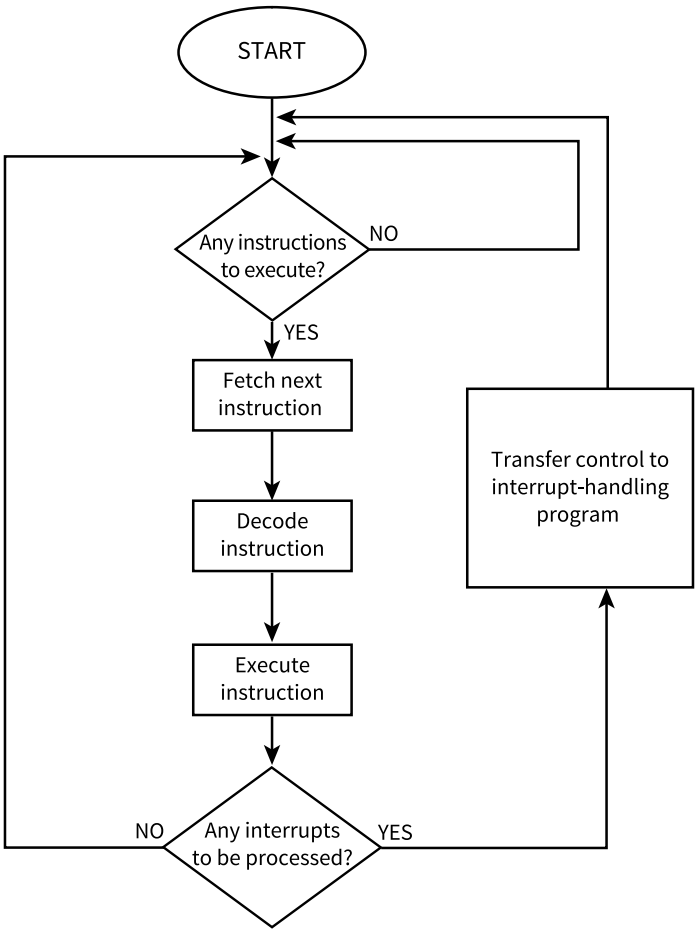


Figure 5.03 Flowchart for the fetch, decode and execute cycle

If we assume that a program is already running, then the program counter will already hold the address of the next instruction. In the fetch stage, the following steps will now happen.

1 This address in the program counter is transferred within the CPU to the MAR.

2 During the next clock cycle two things happen simultaneously:

- the instruction held in the address pointed to by the MAR is fetched into the MDR
- the address stored in the program counter is incremented.

3 The instruction stored in the MDR is transferred within the CPU to the CIR.

There are two points to note here.

- The clock cycle is the one controlled by the system clock which will have settings that allow one data transfer from memory to take place in the time defined for one cycle.
- In the final step the program counter is incremented by 1. However, the instruction just loaded might be a jump instruction. In this case, the program counter contents will have to be updated in accordance with the jump condition. This can only happen after the instruction has been decoded.

In the decode stage, the instruction stored in the CIR is received as input by the circuitry within the control unit. Depending on the type of instruction, the control unit will send signals to the appropriate components so that the execute stage can begin. At this stage, the ALU will be activated if the

instruction requires arithmetic or logic processing.

The description of the execute stage is given in Chapter 6, where a simple instruction set is introduced and discussed.

5.07 Register transfer notation

Operations involving registers can be described by register transfer notation. A simple example of this is a representation of the fetch stage of the fetch–execute cycle:

```
MAR ← [PC]
PC ← [PC] + 1; MDR ← [[MAR]]
CIR ← [MDR]
```

In register transfer notation the basic format for an individual data transfer is similar to that for variable assignment. The first item is the destination for the data. Here the appropriate abbreviation is used to identify the particular register. To the right of the arrow showing the transmission of data is the definition of this data. In this definition, the square brackets around a register abbreviation show that the content of the register is being moved. This movement might also include an arithmetical operation. When two data operations are placed on the same line separated by a semi-colon, this means that the two transfers take place simultaneously. The double pair of brackets around MAR on the second line needs careful interpretation. The content of the MAR is an address; it is the content of that address which is being transferred to the MDR.

5.08 Interrupt handling

There are many different reasons for an interrupt to be generated. Some examples are:

- a fatal error in a program
- a hardware fault
- a need for I/O processing to begin
- user interaction
- a timer signal.

Discussion Point:

Carry out an investigation into the different causes of an interrupt.

Interrupts are handled by a number of different mechanisms, but there are some clear overriding principles. Each different interrupt needs to be handled appropriately. Different interrupts might have different priorities. Therefore, the processor must have a means of identifying the type of interrupt. One way is to have an interrupt register in the CPU that works like the status register, with each individual bit operating as a flag for a specific type of interrupt.

As the flowchart in Figure 5.03 shows, the existence of an interrupt is only detected at the end of a fetch–execute cycle. This allows the current program to be interrupted and left in a defined state which can be returned to later. An interrupt is handled by the following steps.

- The contents of the program counter and any other registers are stored somewhere safe in memory.
- The appropriate interrupt handler or Interrupt Service Routine (ISR) program is initiated by loading its start address into the program counter.
- When the ISR program has been executed there is an immediate check to see if further interrupts need handling.
- Further interrupts are dealt with by repeated execution of the ISR program.
- If there are no further interrupts, the safely stored contents of the registers are restored to the CPU and the originally running program is resumed.

Reflection Point:

Have you worked out a method to remember all of the names and abbreviations for the special purpose registers?

Summary

- The von Neumann architecture for a computer system is based on the stored program concept.
- The CPU contains a control unit, an arithmetic and logic unit and registers.
- Registers can be special purpose or general purpose.
- The status register has individual bits acting as condition flags.
- The system bus contains the data, address and control buses.
- A universal serial bus (USB) port can be used to attach peripheral devices.
- Instructions are handled by the fetch–execute cycle.
- Register transfer notation is used to describe data transfers.
- If an interrupt is detected, control passes to an interrupt-handling routine.