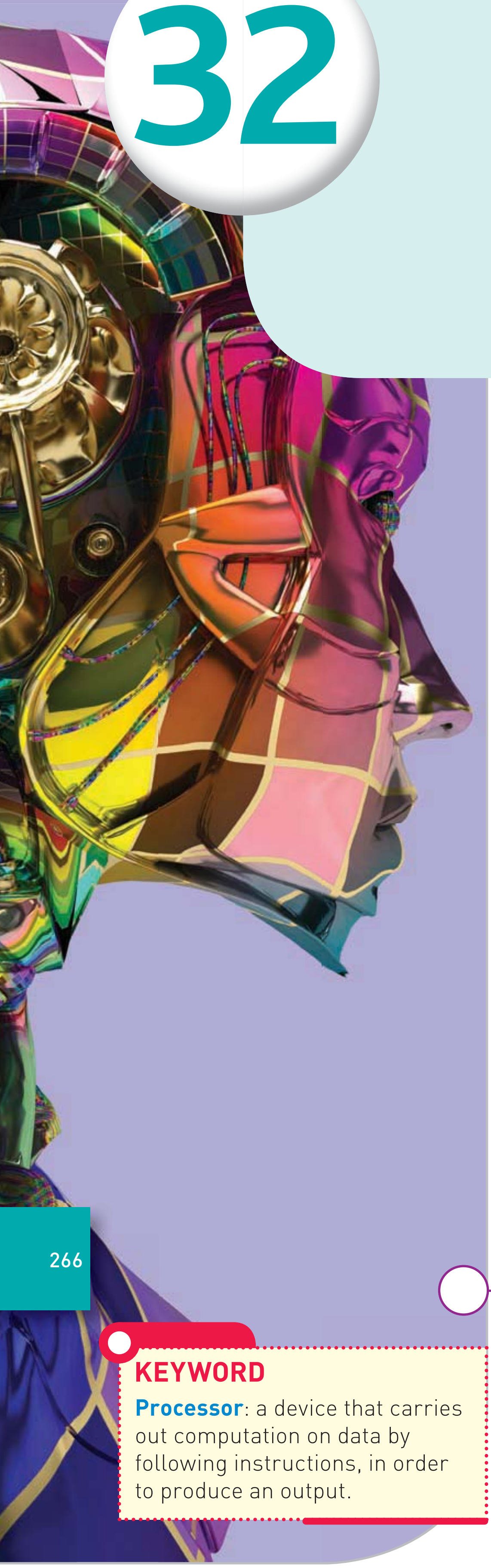


32 Internal hardware of a computer

SPECIFICATION COVERAGE

3.7.1.1 Internal hardware components of a computer

LEARNING OBJECTIVES

In this chapter you will learn:

- how the processor works
- about the different types of memory and what memory is used for
- how buses are used to pass data and instructions
- how external devices are handled
- the difference between the von Neumann and Harvard architectures.

INTRODUCTION

A computer is any machine or device that processes data. The word computer also implies that the machine is electronic or digital. In simple terms this means that it will contain one or more microprocessors that can be programmed to control the device. Microprocessors are made up of microscopic electronic circuits and belong to a group of devices commonly referred to as chips.

This definition is deliberately broad. It is important as A-level students that you realise that your PC (Personal Computer) is only one type of computer and that there are other types in common use. For example, you could look at any number of devices and describe them as computers – a burglar alarm, a microwave oven and a mobile phone all fit this definition. Computers used in this context are referred to as embedded systems as the chip is embedded within another device.

Processor

The **processor** is a device that carries out computation on data by following instructions. It handles all of the instructions that it receives from the user and from the hardware and software. For example, you may press 'A' on the keyboard in which case an electrical signal is sent either wirelessly or through the cable into the USB port at the back of the computer. This signal is routed through the processor which recognises the signal as an 'A'. It then sends a signal to the monitor which displays the letter.

KEYWORD

Processor: a device that carries out computation on data by following instructions, in order to produce an output.

Figure 32.1 A modern processor

Obviously, it is more complicated than this and the process of displaying a letter on a monitor requires the processor to carry out a number of processes which are invisible to the user. The processor will also be receiving instructions from other programs and other devices at the same time. A further complexity is added by the fact that many computers now contain processors with multiple cores and the actions of each will need coordinating.

Physically, the processor is made up of a thin slice of silicon approximately 2 cm square. Using microscopic manufacturing techniques, the silicon is implanted with millions of transistors. Microscopic wires called buses connect groups of transistors together. The transistors are used to control the flow of electrical pulses that are timed via the computer's clock. The pulses of electricity represent different parts of the instruction that the processor is carrying out. Each of these pulses is routed around the circuitry of these transistors at very high speeds. In theory a 3 GHz processor could process 3000 million instructions per second.

Generally speaking, the higher the clock speed of the processor, the faster it will carry out instructions and the faster your computer will work. There are other factors that affect performance, which are covered later. Manufacturers of processors such as Intel and AMD bring out newer, faster chips each year.

Main memory

Memory is used to store data and instructions. It is connected to the processor which will fetch the data and instructions it needs from memory, decode the instructions and then execute them. This is commonly known as the **fetch–execute cycle** and is a key principle in modern computing. There is more on this in the next chapter. Any new data created will be stored back into memory. Put simply, memory is a medium of storage. There are two main types: RAM and ROM.

KEYWORDS

Main memory: stores data and instructions that will be used by the processor.

Fetch–execute cycle: the continuous process carried out by the processor when running programs.

Random Access Memory (RAM): stores data and can be read to and written from.

Chip: an electronic component contained within a thin slice of silicon.

RAM – Random Access Memory

RAM is temporary storage space that can be accessed very quickly. This means that applications such as word processors and spreadsheets will run at high speed. Speed of operation becomes more apparent when you use your computer for games or videos as these require more memory in order to refresh the graphics. Physically, RAM is a chip or series of **chips** on which the data is stored electronically. It is made up of millions of cells, each of which has its own unique address. Each cell can contain either an instruction or some data.

The cells can be accessed as they are needed by the processor, by referencing the address. That is they can be accessed randomly, hence the name. Because they are electronic they are able to be accessed quickly. However, RAM is volatile, which means that when you turn your computer off, all of the contents of RAM are lost.

Figure 32.2 RAM chips

Whenever a program is run on your computer, the entire program or parts of it are loaded into RAM. The more memory you have, the more applications you can have loaded at any one time. For example, if you load a spreadsheet file, both the file and the spreadsheet application are stored in RAM. When you are creating formulae, these are stored temporarily in

RAM. If you turn the computer off without saving it will close down the spreadsheet and your work will be lost.

As for processors, manufacturers are always bringing out more powerful memory chips and the price of memory has actually fallen over the years. In 1997, 32 MB was fairly standard. By 2003, 512 MB was standard. By 2014, 4–8 GB became the standard range for laptops and desktops. Software manufacturers are also bringing out new products all the time that take advantage of the larger memory available. You may have noticed this yourself as certain programs will not run on machines that do not have enough memory.

KEYWORD

Read Only Memory (ROM): stores data and can be read from, but not written to (unless programmable ROM).

ROM – Read Only Memory

ROM is also a method of storing data and instructions. However, it is not volatile which means that the contents of ROM are not lost when you switch off. Unlike RAM, the user cannot alter the contents of ROM as it is read-only. It is important to note that it is possible to have programmable ROM, which is used in memory sticks and other devices. The definition here is of traditional ROM used within a PC.

In this case, ROM is used to store a limited number of instructions relating to the set up of the computer. These settings are stored in the BIOS which stands for Basic Input/Output System.

When you switch on your computer it carries out a number of instructions. For example, it checks the hardware devices are plugged in and it loads parts of the operating system. All of these instructions are stored in ROM. The instructions are programmed into ROM by the manufacturer of the PC.

Addressable memory

Memory is made up of millions of addressable cells and the various instructions and data that make up a program will be stored across a number of these cells. Each address can be uniquely identified. It is the job of the processor to retrieve each instruction and data item and to carry out instructions in a sequential manner.

Memory is organised in a systematic way. Using the addresses, different programs can be stored in different parts of memory. For example, a block of memory addresses will be allocated for the operating system, another block for the application software and so on. This way, the processor is able to find the data and instructions it needs much more quickly than if the programs were stored completely randomly.

A memory map can be produced which shows which programs are stored at which addresses. You will see that memory addresses are normally shown in hexadecimal format rather than binary as the hex version is shorter.

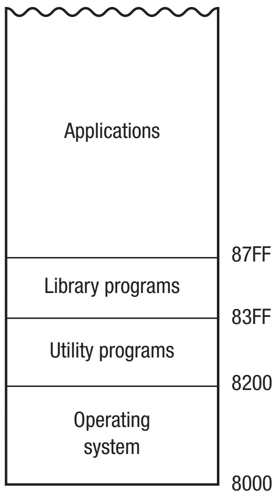


Figure 32.3 A basic memory map

Buses

Buses are groups of parallel microscopic wires that connect the processor to the various input and output controllers being used by the computer. They are also used to connect the internal components of a microprocessor, known as registers (more on these in Chapter 33), and to connect the microprocessor to memory. There are three types of **bus**: data, address and control.

KEYWORDS

Bus: microscopic parallel wires that transmit data between internal components.

Data bus: transfers data between the processor and memory.

Input/Output (I/O) controller: controls the flow of information between the processor and the input and output devices.

Data bus

The instructions and data that comprise a computer program pass back and forth between the processor and memory as the program is run. The **data bus** carries the data both to and from memory and to and from the **I/O controllers**, that is, they are bi-directional or two-way. The instructions and data held in memory will vary in size. Each memory cell will have a width measured in bits. For example, it may have a width of 32 bits.

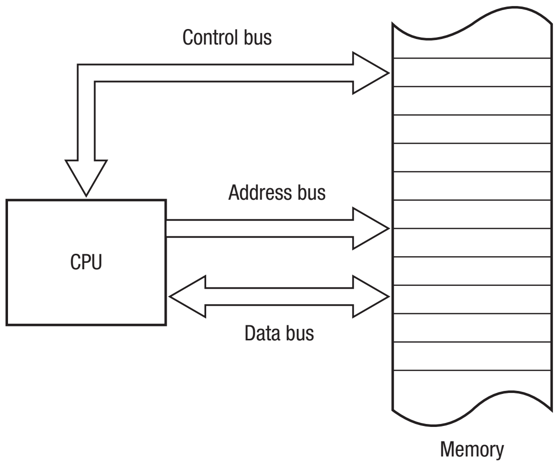


Figure 32.4 Buses connecting the processor to memory

The data bus connects the registers to each other and to memory. The amount of data that can be passed along the bus depends on how many wires are in the bus. An 8-bit data bus has eight wires. There are only two things that can pass down each wire, that is a 0 or a 1. Therefore, by using eight wires on the data bus, we can transmit any item of data that can be represented using 2^8 combinations which is 256. As we saw in Section Five, these patterns can be used to represent text, numbers, sound and graphics or instructions.

Therefore, when large data items are transmitted, the data will have to be split into smaller parts which are sent one after the other. The greater the width of the data bus, in terms of wires, the more data can be transmitted in one pulse of the clock. Consequently, the size of the data bus is a key factor in determining the overall speed and performance of the computer. 32-bit and 64-bit buses are the norm at the time of writing. The data bus width is usually the same as the **word length** of the processor and the same as the memory word length.

KEYWORDS

Word length: the number of bits that can be addressed, transferred or manipulated as one unit.

Address bus: used to specify a physical address in memory so that the data bus can access it.

Address bus

The **address bus** only goes in one direction – from the processor into memory. All the instructions and data that a processor needs to carry out a task are stored in memory. Every memory location has an address. The processor carries out the instructions one after the other. The address bus is used by the processor and carries the memory address of the next instruction or data item. The address bus therefore is used to access anything that is stored in memory, not just instructions.

KEYWORDS

Addressable memory: the concept that data and instructions are stored in memory using discrete addresses.

Control bus: controls the flow of data between the processor and other parts of the computer.

The size of the address bus is also measured in bits and represents the amount of memory that is addressable. An 8-bit bus would only give 256 directly **addressable memory** cells. This means that a program could only consist of a maximum of 256 separate instructions and/or data items. If we assume that each memory address can store 8 bits (one byte) of data then we would have 256 bytes of memory available. This would be useless on modern computers.

You may have realised from Section Five that each additional wire will double the capacity. Consequently 24 lines on the address bus would give 2^{24} combinations, which means it can access 16 MB of memory. A 32-bit address bus, which is common for most PCs, would provide 4 GB of addressable memory. A 64-bit address bus would provide, in theory, addressable memory of 16.8 million terabytes.

Control bus

The **control bus** is a bi-directional bus which sends control signals to the registers, the data and address buses. There is a lot of data flowing around the processor, between the processor and memory, and between the processor and the input and output controllers. Data buses are sending data to and from memory while address buses send only to memory.

The job of the control bus therefore is to ensure that the correct data is travelling to the right place at the right time. This involves the synchronisation of signals and the control of access to the data and address buses which are being shared by a number of devices.

For example, a signal on the control bus would dictate the direction of data transmission through the data bus; it would also indicate whether it was reading to or writing from an I/O port. The control bus will also be transmitting the pulses being delivered by the system's clock.

Input/Output (I/O) controllers

In addition to the direct link between the processor and main memory, the processor will also receive and send instructions and data to the various input and output devices connected to the computer. Basic I/O devices would be the keyboard, monitor, mouse and printer, though modern computer systems would typically include several other devices.

Physically, these I/O devices are connected via the I/O ports (usually USB ports) on your computer as shown in Figure 32.5. The ports are physical connections that allow I/O devices to be plugged in. For example, the printer will be plugged into one of the USB ports. Signals will be passed in both directions through the printer cable, via the port and through the processor to send and receive the instructions.

Inside the computer, the data buses carry the signals to and from the processor. In order to do this the processor is working in the same way as if it were sending data to or from memory. The difference, however, is that the processor does not communicate directly with the I/O devices. Instead, there is an interface called an I/O controller.

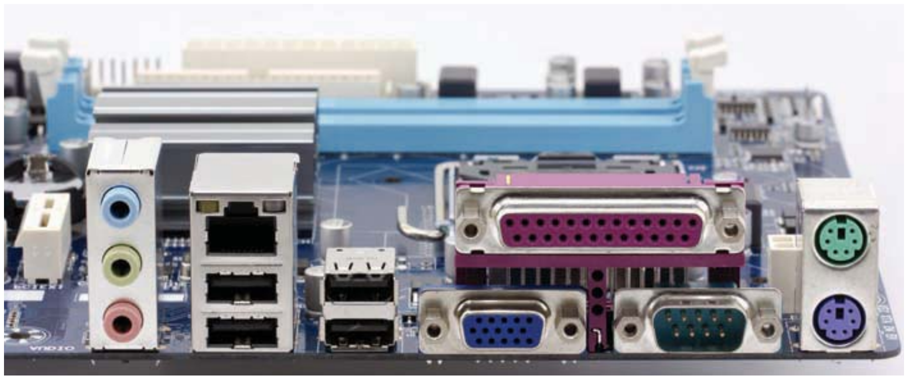


Figure 32.5 Physical ports on a standard PC

Controllers consist of their own circuitry that handle the data flowing between the processor and the device. Every device will have its own controller which allows new devices to be connected to the processor at any time. As a minimum, therefore, a typical computer will have a monitor controller, a mouse controller, a keyboard controller and a hard disk controller.

A key feature of an I/O controller is that it will translate signals from the device into the format required by the processor. There are many different devices and many different types of processor and it is the I/O controller that provides the flexibility to add new devices without having to redesign the processor.

Another important feature is that the I/O devices themselves respond relatively slowly compared to the speed at which a processor can work. Therefore the I/O controller is used to buffer data being sent between the processor and the device, so that the processor does not have to wait for the individual device to respond.

Von Neumann and Harvard architectures

In Section One we identified that a program is a series of instructions that the processor will carry out. Programs also require the data on which these instructions will be carried out. As we have seen, a program is loaded into main memory when it is run. In simple terms it means that the instructions and data that comprise a program are both stored in main memory and must both pass through the same bus (the data bus) in and out of memory.

The early computers that used this concept were known as 'von Neumann' machines after the man who first invented the technique in the 1940s. Most modern PCs use this technique and so are also von Neumann machines.

The word 'architecture' is widely used in computing and usually refers to the way that something is built. For example, a microprocessor has an architecture that refers to the way that the chip is built. The von Neumann method of building computers therefore is often referred to as **von Neumann architecture**.

An alternative method of building chips is the **Harvard architecture**. The key difference between this and von Neumann is that separate buses are used for data and instructions, both of which address different parts of memory. So rather than storing data and instructions in the same memory and then passing them through the same bus, there are two separate buses addressing two different memories.

KEYWORDS

Von Neumann architecture: a technique for building a processor where data and instructions are stored in the same memory and accessed via buses.

Harvard architecture: a technique for building a processor that uses separate buses and memory for data and instructions.

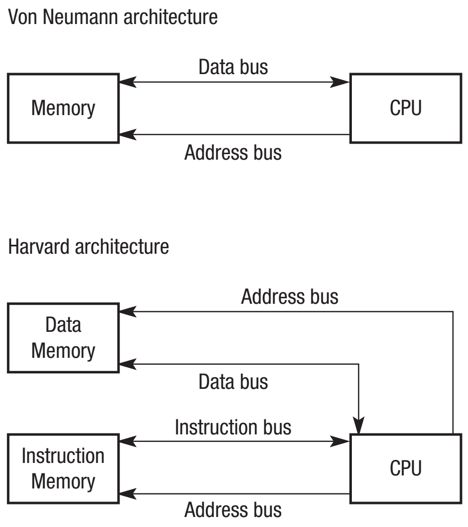


Figure 32.6 The von Neumann and Harvard architectures

The advantage of this is that the instructions and data are handled more quickly as they do not have to share the same bus. Therefore a program running on Harvard architecture can be executed faster and more efficiently. Harvard architecture is widely used on embedded computer systems such as mobile phones, burglar alarms etc. where there is a specific use, rather than being used within general purpose PCs.

Many such devices use a technique called Digital Signal Processing (DSP). The idea of DSP is to take continuous real world data such as audio or video data and then to compress it to enable faster processing. Chips that are optimised for DSP tend to have much lower power requirements, making them ideal for applications such as mobile phones where power consumption is critical.

Practice questions can be found at the end of the section on page 298.

TASKS

1 Identify the three other terms that are commonly used to mean main memory.

2 Identify one advantage of increasing the amount of RAM in a PC.

3 Describe the purpose of the following components:
   a) processor
   b) random access memory
   c) read only memory.

4 Explain why it is necessary to have ROM, RAM and a hard disk within a computer system.

5 Identify one advantage of increasing the capacity of the hard disk.

6 On a games console, explain why it can take over a minute to load a game.

7 What is the maximum amount of data that can pass down a 16-bit data bus in one stage of the fetch–execute cycle?

8 What is the largest amount of addressable memory available with a 16-bit address bus?

9 Name and describe the function of the three different buses.

10 Identify three functions carried out by the control bus.

11 Give two reasons why I/O devices are handled by controllers rather than being connected directly to the processor.

KEY POINTS

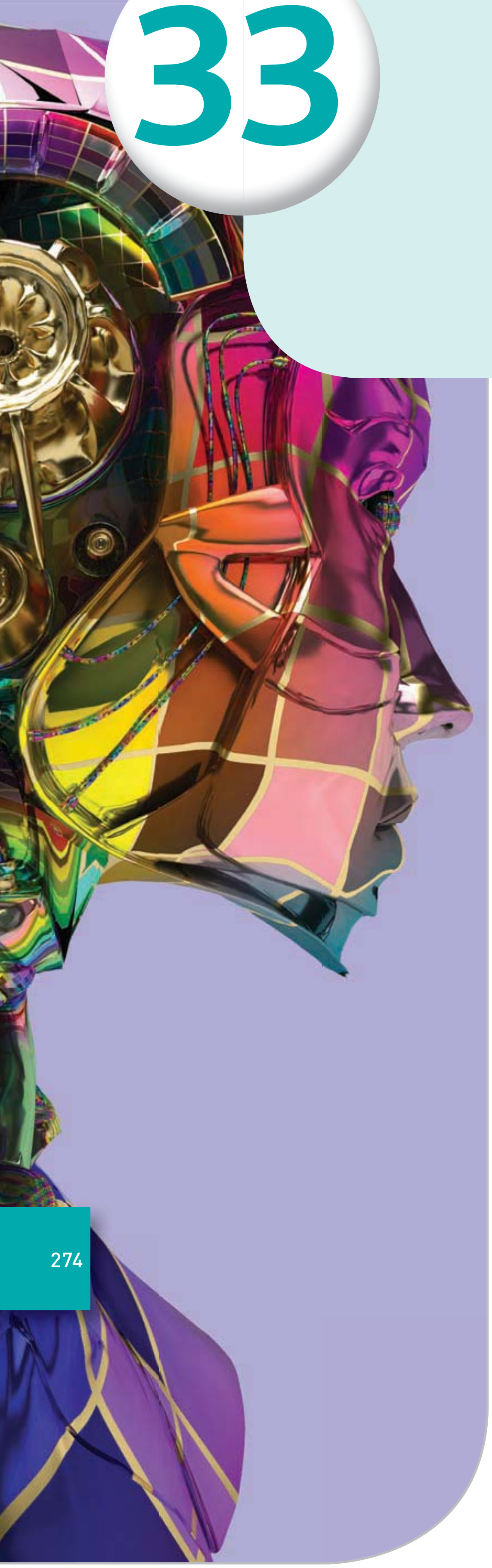
- The processor handles and processes instructions from the hardware and software.
- Processors can handle millions of instructions every second.
- Memory is made up of millions of addressable cells.
- Data and instructions are fetched, decoded and executed.
- There are two main types of main memory, RAM and ROM.
- The processor is connected to main memory and data and instructions are passed around circuitry known as buses.
- In von Neumann architecture, instructions and data are stored together in memory.
- In Harvard architecture, separate memory is used for data and instructions.

STUDY / RESEARCH TASKS

1 Recommend a suitable specification for a computer system for online gaming. How would the specification of this system vary from one that was designed to handle a large database?

2 Moore's Law broadly states that processor performance will double every two years. He made this prediction in 1965. Is it true and do you think it will continue to be true?

3 Some people believe that the next big advance in microprocessor technology is when we move on from the silicon chip. What are the limitations of the silicon chip and what might replace it in the future?



33 The stored program concept and processor components

SPECIFICATION COVERAGE

3.7.2.1 The meaning of the stored program concept

3.7.3.1 The processor and its components

3.7.3.2 The fetch–execute cycle and the role of registers within it

3.7.3.6 Factors affecting processor performance

3.7.3.6 Interrupts (A Level only)

LEARNING OBJECTIVES

In this chapter you will learn:

- how instructions are handles using the stored program concept
- what happens at each stage of the fetch–execute cycle
- that a processor is made up of components including registers and units
- how the registers and units are used to handle instructions
- what factors affect processor performance.

A-level students will also learn:

- what an interrupt is and how a processor handles it.

INTRODUCTION

In this chapter we look in more detail at what happens during the fetch–execute cycle, looking specifically at the physical components of the processor and how data and instructions are handled internally. The processor is made up of microscopic registers and units and each instruction will be passed around these components, manipulating 0s and 1s in order to create a result. This chapter explains what happens at each stage of the cycle. In addition, we consider the factors that affect the overall performance of the processor.

A-level students also need to be aware of the interrupt and this is covered at the end of the chapter.

KEYWORDS

Stored program concept: the idea that instructions and data are stored together in memory.

Fetch–execute cycle: the continuous process carried out by the processor when running programs.

The stored program concept

As you saw in the previous chapter, the von Neumann concept was to store instructions and data in the same memory unit. Each instruction or data item is fetched from memory, decoded and then executed, with any new data created being placed back into memory. Every time a program is run, the processor runs through this **fetch–execute cycle**.

Therefore, all the processor is doing is running through this cycle over and over again, millions of times every second. The computer's clock times the electrical pulses into the processor.

- Fetch – the processor fetches the program's next instruction from memory. The instruction will be stored at a memory address and will contain the instruction in binary code.
- Decode – the processor works out what the binary code at that address means.
- Execute – the processor carries out the instruction which may involve reading an item of data from memory, performing a calculation or writing data back into memory.

It is worth pointing out that a simple instruction for a user, for example, adding two numbers together, would actually involve a number of cycles for the processor. There is also an unanswered question in terms of how the processor fetches, decodes and executes each instruction. To understand this fully, you need to understand the architecture of the processor. Processors are made up of a number of components including the clock, control unit, arithmetic logic unit and various registers.

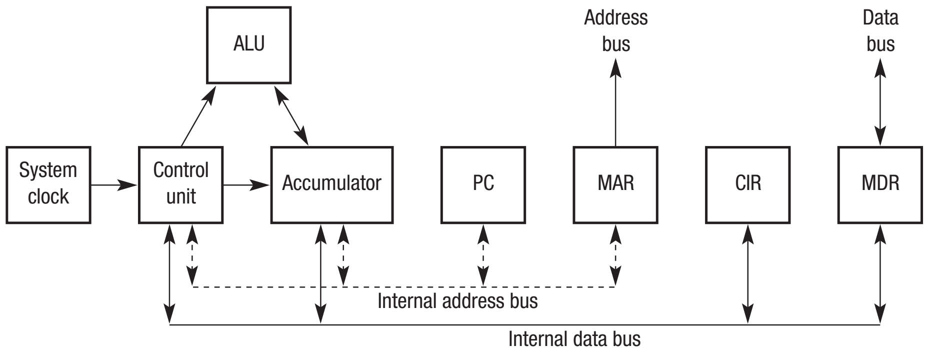


Figure 33.1 Machine architecture – the processor

KEYWORDS

Control unit: part of the processor that manages the execution of instructions.

Arithmetic Logic Unit (ALU): part of the processor that processes and manipulates data.

The control unit

The **control unit** is the part of the processor that supervises the fetch–execute cycle. The control unit also makes sure that all the data that is being processed is routed correctly – it is put in the correct register or section of memory.

The arithmetic logic unit (ALU)

The **ALU** carries out two types of operation – arithmetic and logic. The ALU can be used to carry out the normal mathematical functions such as add, subtract, multiply and divide, and some other less familiar processes such as shifting. This process is described in more detail later in the chapter.

The ALU is also used to compare two values and decide if one is less than, greater than or the same as another. Some comparisons will result in either TRUE or FALSE being recorded.

The ALU is sent an operation code (op-code) and the operands (the data to be processed). The ALU then uses logical operations such as OR, AND and NOT to carry out the appropriate process. In some computers, a separate arithmetic unit (AU) is used to cope with floating-point operations.

KEYWORD

Clock: a device that generates a signal used to synchronise the components of a computer.

The clock

All computers have an internal **clock**. The clock generates a signal that is used to synchronise the operation of the processor and the movement of data around the other components of the computer.

The speed of a clock is measured in either megahertz (MHz – millions of cycles per second) or gigahertz (GHz – 1000 million cycles per second). In 1990 a clock speed of between 4 and 5 MHz was the norm. In 2000, 1 GHz clock speeds were common. The typical clock speed at the time of writing is 2–3 GHz.

Registers

The control unit needs somewhere to store details of the operations being dealt with by the fetch–execute cycle and the ALU needs somewhere to put the results of any operations it carries out. There are a number of storage locations within the processor that are used to store this sort of data. They are called **registers** and although they have a very limited storage capacity they play a vital role in the operation of the computer.

A register must be large enough to hold an instruction – for example, in a 32-bit instruction computer, a register must be 32 bits in length. Some of these registers are general purpose but a number are used for a specific purpose:

- The **status register** keeps track of the status of various parts of the computer – for example, if an overflow error has occurred during an arithmetic operation.
- The **interrupt register** is a type of status register. It stores details of any signals that have been received by the processor from other components attached to it, for example, the I/O controller for the printer. This will receive input and output requests from processor and then send device-specific instructions to the printer. The I/O controller performs any necessary conversion of signals between the processor and a peripheral, ensuring that new peripherals can easily be connected. We will be looking at the role of interrupts later in the chapter.

There are four registers that are used by the processor as part of the fetch–execute cycle:

- The **Current Instruction Register (CIR)** stores the instruction that is currently being executed by the processor.
- The **Program Counter (PC)** stores the memory location of the next instruction that will be needed by the processor.
- The **Memory Buffer Register (MBR)**, also known as the **Memory Data Register (MDR)**, holds the data that has just been read from or is about to be written to main memory.
- The **Memory Address Register (MAR)** stores the memory location where data in the MBR is about to be written to or read from.

KEYWORDS

Register: a small section of temporary storage that is part of the processor. Stores data or control instructions during the fetch–decode–execute cycle.

Status register: keeps track of the various functions of the computer such as if the result of the last calculation was positive or negative.

Interrupt register: stores details of incoming interrupts.

Current Instruction Register (CIR): register that stores the instructions that the CPU is currently decoding/executing.

Program counter (PC): register that stores the address of the next instruction to be taken from main memory into the processor.

Memory Buffer Register (MBR): register that holds data that is either written to or copied from the CPU.

Memory Data Register (MDR): another name for the MBR.

Memory Address Register (MAR): register that stores the location of the address that data is either written to or copied from by the processor.

How the cycle works

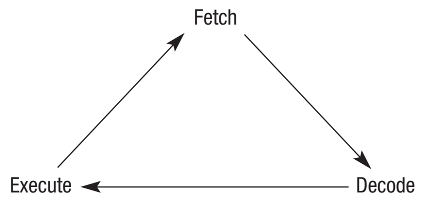


Figure 33.2 The fetch–execute cycle

- Fetch: The PC holds the address of the next instruction. The processor sends this address along the address bus to the main memory. The contents of the memory location at that address are sent via the data bus to the CIR and the PC is incremented. The details of addresses are initially loaded into the MAR and the data initially goes to the MBR. Some instructions need to load a number of bytes or words, so they may need to be fetched as successive parts of a single instruction.
- Execute: The processor then takes the instruction from the CIR and decides what to do with it. It does this by referring to the instruction set. These instruction sets are either classed as an RISC (reduced instruction set) or a CISC (complex instruction set). An instruction set is a library of all the things the processor can be asked to do. Each instruction in the instruction set is accompanied by details of what the processor should do when it receives that particular instruction. This might be to send the contents of the MBR to the ALU. There is a detailed example of how this works in the next chapter.

  Once the instruction that has just been taken from the memory has been decoded, the processor now carries out the instruction. It then goes back to the top of the cycle and fetches the next instruction. A simple instruction will require only a single clock cycle, whereas a complex instruction may need three or four. The results of any calculations are written either to a register or a memory location.

Factors affecting processor performance

There are a number of factors that affect processor performance. Often these factors have to be looked at in combination to understand how quickly a processor will work. For example, clock speed is seen as an important measure of performance, but increasing clock speed alone may not have a positive effect if other components within the processor are limited. There is no point fitting a faster clock to a computer if you do not change the components that are going to make use of that pulse as well.

- Clock speed: As we saw earlier, clock speed is one measure of the performance of the computer. It indicates how fast each instruction will be executed. In theory therefore, increasing the clock speed will increase the speed at which the processor executes instructions.
- Bus width: The processor needs to optimise the use of the clock pulse. One way of doing this is to increase the **bus width**. In the last chapter we looked at the data bus and address bus and saw how the width of the bus showed how many bits could be transferred in one pulse of the clock. Increasing the width of the data bus means that more bits and therefore more data can be passed down it with each pulse of the clock,

KEYWORD

Bus width: the number of bits that can be sent down a bus in one go.

which in turn means more data can be processed within a given time interval. Increasing the width of the address bus will increase the amount of memory that can be addressed and therefore allows more memory to be installed on the computer.

- Word length: Related to the data bus width is the **word length**. A word is a collection of bits that can be addressed and manipulated as a single unit. Computer systems may have a word length of 32 or 64 bits, indicating that 64 bits of data can be handled in one pulse of the clock. Word length and bus width are closely related in that a system with a 64-bit word length will need 64-bit buses.
- Multiple cores: Most computer systems have one processor. One way of increasing system performance is to use several processors. For convenience, multiple processors can be incorporated onto one single chip; this is known as a **multi-core** processor. A dual-core processor therefore has two processors on the one chip and will run much faster than a single-core system, which only has one processor. The term 'core' is used to define the components that enable instructions to be fetched and executed.
- Cache memory: Caching is a technique where instructions and data that are needed frequently, are placed into a temporary area of memory that is separate from main memory. The advantage of this is that the **cache** can be accessed much more quickly than main memory, so programs run faster. The key to this is ensuring that the most commonly used functions or data used in a program are placed into the cache.

KEYWORDS

Word length: the number of bits that can be addressed, transferred or manipulated as one unit.

Multi-core: a chip with more than one processor.

Cache: a high-speed temporary area of memory.

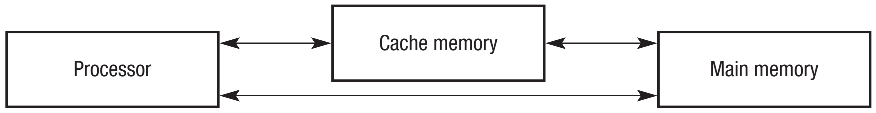


Figure 33.3 Cache memory

A level only

Interrupts

The processor in a computer is always working, irrespective of whether there is an application active or not. This is because the operating system, which is itself a large collection of programs, is always active. This means that the fetch–execute cycle is always in use. If an error occurs or a device wants the computer to start doing something else then we need some way to grab the processor's attention. The way to do this is to send an **interrupt**. An interrupt is a signal sent to the processor by a program or an external source such as a hard disk, printer or keyboard.

KEYWORD

Interrupt: a signal sent by a device or program to the processor requesting its attention.

There are a number of different sources of an interrupt. These are some typical examples:

- a printer sends a request for more data to be sent to it
- the user presses a key or clicks a mouse button
- an error occurs during the execution of a program, for example, if the program tries to divide by zero or tries to access a file that does not exist
- an item of hardware develops a fault
- the user sends a signal to the computer asking for a program to be terminated
- the power supply detects that the power is about to go off
- the operating system wants to pass control to another user in a

How the interrupt works

What happens is that an additional step is added to the fetch–execute cycle. This extra step fits between the completion of one execution and the start of the next. After each execution the processor checks to see if an interrupt has been sent by looking at the contents of the interrupt register.

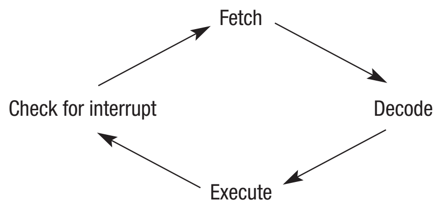


Figure 33.4 The fetch–execute cycle with interrupts

KEYWORDS

Interrupt Service Routine: calls the routine required to handle an interrupt.

Priorities: a method for assigning importance to interrupts in order to process them in the right order.

If an interrupt has occurred the processor will stop whatever it is doing in order to service the interrupt. It does this using the **Interrupt Service Routine (ISR)** which calls the routine required to handle the interrupt. Most interrupts are only temporary so the processor needs to be able to put aside the current task before it can start on the interrupt. It does this by placing the contents of the registers, such as the PC and CIR on to the system stack. Once the interrupt has been processed the CPU will retrieve the values from the stack, put them back in the appropriate registers and carry on.

Priorities

Sometimes the program that has interrupted the running of the processor is itself stopped by another interrupt. In this case the processor will either place details of its current task on the stack or it will assess the priority of the interrupts and decide which one needs to be serviced first. Assigning different interrupts different priority levels means that the really important signals, such as a signal indicating that the power supply is about to be lost, get dealt with first.

Table 33.1 shows some of the processes that can generate an interrupt, and the priority level that is attached to that interrupt. Level 1 is the highest priority, 5 the lowest. Interrupts with the same priority level are dealt with on a first-come first-served basis.

Table 33.1 Priorities, interrupts and possible causes

Level	Type	Possible causes
1	Hardware failure	Power failure – this could have catastrophic consequences if it is not dealt with immediately so it is allocated the top priority.
2	Reset interrupt	Some computers have a reset button or routine that literally resets the computer to a start-up position.
3	Program error	The current application is about to crash so the OS will attempt to recover the situation. Possible errors could be variables called but not defined, division by zero, overflow, misuse of command word, etc.
4	Timer	Some computers run in a multitasking or multiprogramming environment. A timer interrupt is used as part of the time slicing process.
5	Input/Output	Request from printer for more data, incoming data from a keyboard to a mouse key press, etc.

Vectored interrupt mechanism

Once the values of the registers have been pushed to the stack, the processor is then free to handle the interrupt. This can be done using a technique called a **vectored interrupt mechanism**.

Each interrupt has an associated section of code that tells the processor how to deal with that particular interrupt. When the processor receives an interrupt it needs to know how to find that code. Every type of interrupt has an associated memory address known as a vector. This vector points to the starting address of the code associated with each interrupt.

So when an interrupt occurs, the processor identifies what kind of interrupt it is, then finds its associated interrupt vector. It then uses this to jump to the address specified by the vector, from where it runs the Interrupt Service Routine (ISR).

Practice questions can be found at the end of the section on page 298.

KEYWORD

Vectored interrupt mechanism: a method of handling interrupts by pointing to the first memory address of the instructions needed.

KEY POINTS

- The stored program concept is the idea of instructions and data being stored together in memory.
- The fetch–execute cycle explains how an instruction is fetched from memory, and executed to produce a result and place this back into memory.
- There are a number of key components of the processor including: the clock, the control unit, the arithmetic logic unit and various registers. You need to know how an instruction passes through all of these components.
- There are a combination of factors that affect the performance of a processor including clock speed, bus width, word length and caching.
- An interrupt is a signal (e.g. from a hardware device) that stops the processor from carrying out its current instruction in order to deal with another task.

TASKS

1. Describe what happens at each stage of the fetch–execute cycle.
2. What is the maximum amount of data that can pass down a 16-bit data bus in one stage of the fetch–execute cycle?
3. What is the ALU and what function does it perform?
4. Explain what the clock in a computer does.
5. The processor has access to many registers. What is a register?
6. The control unit uses four registers to control the execution of a program. They are the CIR, PC, MAR and MBR. Explain what each of these is and the part it has to play in the execution of a program.
7. Why might clock speed be an inaccurate way of measuring the performance of a computer system?
8. Why are the contents of the registers put on the stack before an interrupt is processed?
9. Why is it important that different types of interrupts have different priorities?
10. Explain how the vectored interrupt mechanism works.

STUDY / RESEARCH TASKS

1. Some of the major causes of interrupts are listed in the chapter. Find out about other causes of interrupts and try to decide what priority level you would give each.
2. What is over-clocking and what are the positive and negative effects of it on your computer?
3. What are the implications for the operating system of a multi-core processor?
4. Explain why having a dual-core system does not make your computer twice as fast.