1 Computer platforms

Summary

The aim of this chapter of the book is to provide sufficient technical information to enable users to make informed purchasing or specification decisions without becoming over involved with detail. Each heading will present the information from the user's point of view and, where necessary, ignore or gloss over the most technical details. For instance, no real attempt will be made to describe the precise working of a disk drive but the benefits of its performance will be outlined. No attempt has been made to cover other aspects of the use of microprocessors such as embedded systems, mobile communications devices, etc. nor to look at larger machines, mainframe computers or other specialized areas. The focus will be on how to buy a PC but with sufficient content to be at Higher National level.

Introduction

The Edexcel unit says '*This unit is aimed at IT practitioners who need sufficient knowledge of computer architecture to make rational and commercial decisions on the selection and specification of systems. Learners will learn how to evaluate operating systems in order to create their own operating environment. Many IT practitioners communicate with specialist technical support staff during the specification and planning of systems implementation. This unit aims to give learners the confidence to communicate with technical and non-technical specialists to justify their recommendations.*

It is expected that centres will use current personal computer and networking resources. Learners should be encouraged to read current journals to investigate and evaluate new hardware and software developments.'

It is recognized that centres use a diverse range of hardware and software. For this reason, this chapter avoids specific software dependent items as far as possible.

1.1 The basic components of a PC

This part of the book is intended to describe the basic parts of a computer ready to fulfil the requirements of Unit 1, i.e. how to buy a PC.

Students who have embarked on an HNC in Computing will not need to be told that there is a monitor, a system box, etc.; all will have used these items, but they will need the knowledge of how to specify each one or to communicate effectively with experienced technicians.

The system box

This contains the motherboard with the CPU or *Central Processing Unit*, storage devices such as hard drives or CD-ROM drive, the memory, a power supply and any add-on components such as a video card, modem, etc.

Motherboard

So called because older computers were made from a large number of components organized onto several circuit boards. These were plugged into one ‘main’ or motherboard that contained the CPU and the circuits that communicated with the add-on boards. Modern machines have most of the principal components on one board but the name has stuck. The motherboard will house the CPU, the *chipset*, the memory connectors or expansion buses for the circuits that are still separate and often the I/O (input/output) ports. The chipset controls DMA or *Direct Memory Access*, the *bus interface*, memory, disks, keyboard, I/O ports and timing.

CPU

The CPU is the circuit that is able to execute the instructions stored in the memory. Modern CPUs in the Intel Pentium series and others are able to execute these instructions at high speed and provide considerable computing power in a small component.

Storage

All the instructions and data that form *software* must at one point be stored. This data is all in the form of logical 1s or 0s and any physical property of any substance or device that will remain in one state or another can be used to store this software. Most hard disks are magnetic devices that store 1s and 0s as changes in the patterns of magnetic particles held on a surface. CD-ROMs hold 1s and 0s by optical patterns on the surface of a simple material. The only real reason why magnetic hard drives are common is that they are cheap, fast and reliable. When newer, faster devices with higher capacity are manufactured, magnetic hard drives may become a thing of the past; purely optical devices hold this promise. The point is, there is nothing special about a *hard drive*, it is simply a device that can store a large number of 1s and 0s and deliver them to another circuits at an acceptable speed.

Memory

The memory stores software, i.e. program instructions and data. There are two broad kinds with the rather confusing names RAM for *Random Access Memory* and ROM for *Read Only Memory*. The problem is that both may be randomly accessed and some kinds of ROM can be written too! The key point is that RAM is *volatile*, i.e. it loses its data when the power is turned off, ROM does not, it is *non-volatile*.

Power supply

The power supply is another misleading name as the power to run the computer usually comes from the mains or from batteries. The job of the power supply is to provide 12 volts to run disk drive motors, etc. and lower (usually 5 or 3.3) volts to run the digital circuits. It must be able to provide enough current to run everything in the machine without overheating and to ensure the voltage is constant with defined limits.

Display

The most commonly used desktop display device is the CRT or *Cathode Ray Tube*, a device first widely used in the sciences and defence in the 1940s and extremely expensive at that time; they were also very unreliable. Now CRTs are made cheaply by the million and are amongst the most reliable devices in common use. With the rise in use of laptop computers and the need to save desk space, etc., newer screen types have also become available and the marketplace has become extremely competitive with LCD or *Liquid Crystal Display* screens currently being the most common. The common name for a CRT is a VDU or *Visual Display Unit*.

Key fact

If you need to buy a PC or just one of the components, you need to know more about its *performance* than how it works. A knowledge of how it works will help with your understanding of the performance and some of the difficulties overcome by the maker but this knowledge does not need to be in great detail. The remaining sections in this chapter are intended to provide the required knowledge.

The hierarchy of design

You know that computers are binary devices often made with silicon circuits and they work with logical 1s and 0s. It is hard to imagine the connection between this statement and seeing a wordprocessor in action with all the screen colours, text and clickable buttons, i.e. a program in action. To illustrate this, imagine you are given the task of explaining the idea of a ‘city’ to someone who has only ever lived on a desert island and never had need of permanent housing. If you started by describing ‘what is a brick’ and then immediately described the construction of a whole town using bricks, the connection between the small hard brick and the warm and comfortable rows of houses would be very difficult to follow. If you then took the view of a town planner and spoke of where the hospital should be in your town or how to route a road around a village, any connections with bricks would be entirely lost. The trouble is, towns are made of bricks!

The connection between 1s and 0s and tasks such as installing Windows is of a similar nature. The way to overcome this is to think of ‘layers’ of knowledge. Using the brick and the town example, consider these layers:

of bricks and towns	*of computer hardware*	*of computer software*
Bricks. Study what a brick is made of, how it is made, how strong it is, what will it cost.	1s and 0s, simple digital circuits and how logical arithmetic can be performed with a circuit.	Boolean logic.

of bricks and towns	*of computer hardware*	*of computer software*
Walls. Study how to mix cement, how	How a sequence of logical operations can be	How to perform arithmetic with
of bricks and towns	*of computer hardware*	*of computer software*
to lay bricks to make a wall, how strong is a wall.	achieved with a circuit, how to add, subtract, perform logical AND and OR operations, etc.	simple numbers.
How to make several walls into a building with spaces for windows and doors, etc. How to build a roof.	How to store many logical instructions and feed them in sequence to a circuit that can execute them.	How to perform arithmetic with multiple digit numbers.
How to install all the services a building needs, water, electricity, gas, heating, etc. and to move in the carpets, furniture, etc.	How to accept human inputs by devices such as a keyboard and to display outputs using devices like a colour monitor.	How to handle data such as text and to edit it, i.e. move a sentence within a paragraph.
How to build a row of houses, provide street lighting, public access, etc.	How to provide a complete set of devices such as a mouse, keyboard, printer, CPU etc. and to make them all connect correctly.	How to present a complete set of facilities in a word processor.
How to plan a town, provide libraries, shops, hospital, bus station, etc.	A complete PC.	How to control the entire machine – the operating system.

In normal life, we expect different people to be expert at these different layers, a brick layer is not a town planner. When studying for the HNC in Computing we do not expect you become expert in any one of the layers, rather to understand all the layers in the same way that you can imagine all the tasks required to build a town; specialization will come later in your studies.

What I am asking you to do here is to take on trust that descriptions of the 'bricks' will lead to an understanding of the 'town council'. It just takes some time.

1.2 Elements in the history of microprocessors

It is said that history is written by the victors. Although this really refers to political history and especially the result of political failure, war, it also applies to vast business areas like computing. Much of the history of computers has been written by the commercial 'victors', the Americans, but you should read it with care; often the rest of the world is left out. As an example, is it often quoted that the world's first computer was an American machine called Electronic Numerical Integrator and Computer (ENIAC) which came out in 1946. The problem is, this machine and others built around the 1940s were not like modern machines so we get into a discussion about what exactly *is* a computer. Amongst other computer projects around the world, in Britain, Germany and other countries, was a highly secret project that pre-dates ENIAC. The design and use of this project was carried out with brilliant success in Britain during the Second World War, it was called the *Colossus* computer; it was used at Bletchley Park in the decyphering of the German Lorenz cypher (not Enigma). It was so secret that its very existence was not officially published until 1974, long after some of the history books were written.

Here is the real problem, some people only use other history books as their source so some 'facts' are propagated with time. If you research the history of microprocessors, you will find that different sources quote 'facts' that vary greatly one from another. As you advance in your studies, you must develop the ability to look critically at these facts and to decide what is the most accurate. The world is a complex place, so simple claims are unlikely to be true. As an example, most people will consider it a fact that Isaac Newton discovered gravity when an apple hit him on his head whilst seated under an apple tree. The trouble is, this is a complete fiction but it is propagated as a fact. Beware!

The first Colossus computer, the Mark 1, was operational in January 1944. It was designed and built by the Post Office at Dollis Hill in London. Like ENIAC, Colossus is not a stored program machine, the 'program' was hard-wired and stored as switch settings. During the decryption process for which it was built, messages to be worked on were stored as a Baudot code (a 5 digit binary code) on a paper tape and these messages were read at 5000 characters a second. In the 200 microseconds it took between each character, Colossus could perform 100 Boolean calculations on each of the 5 binary digits of a character. It is difficult to compare exactly the speed difference between this kind of computer and a modern computer because Colossus did some of its work in parallel but if you take 100 Boolean calculations, one for each of the 5 digits and do this 5000 times a second you get $100 \times 5 \times 5000 = 2\,500\,000$ Boolean calculations per second. It would *not* be right to compare this directly with a speed of 2.5 MIPS (Million Instructions Per Second) but for the world's first computer it is a very respectable performance; this machine was not slow. See www.codesandciphers.org.uk/ for more information.

There is little doubt that the first commercial microprocessor was the Intel 4004 and that Intel has gone on to become a dominant force in the world of microcomputers but other brilliant designs remain less well known. The ubiquitous PC uses Intel microprocessors but Apple machines use the Motorola MC680x0 series and most mobile phones and many small devices use a British designed RISC (Reduced Instruction Set) chip called the ARM.

It is not the purpose of this book to describe in detail each of several hundred different microprocessors, but details can be seen directly from the makers web pages:

Intel	www.intel.com/products/index.htm
Intel past types	www.intel.com/intel/intelis/museum/Exhibits/hist_micro/hof/hof_main.htm (this link seems to change all the time. If it fails, start up www.intel.com and search for '80286', one of the resulting hit list should point to the Hall of Fame)

Motorola	The old link was to www.mot.com/SPS/MMTG/mp.html but this no longer works. Motorola seem to have removed microprocessor history links from their website. See www.motorola.com
ARM	www.arm.com (follow links to CPUs)
Cyrix	www.cyrix.com
AMD	www.amd.com and follow links to 'products'
IBM	www.chips.ibm.com/products/powerpc/
Sun	www.sun.com
Zilog	http://www.zilog.com/products/

In the preparation of this second edition, most of the links printed in the first edition were found to fail. If this happens with any of these links, use the main URI e.g. www.intel.com, and use their search facility.

Table 1.2.1 shows some important Intel microprocessors, the intention being to show the rapid development in terms of speed and complexity. (Intel actually produces many more types.)

Table 1.2.1 *Crude indicators for Intel microprocessors*

Intel chip	*Date*	*MIPS*	*Width of data bus*	*Number of transistors*
4004	1971	0.06	4	2300
8008	1972	0.06	8	3500
8080	1974	0.64	8	6000
8085	1976	0.37	8	6500
8086	1978	0.33	16	29 000
8088	1979	0.33	16	29 000
80286	1982	1.2	16	134 000
80386SX	1985	5.5	16	275 000
80486DX	1989	20	32	1 200 000
80486SX	1991	13	32	1 185 000
80486DX2	1992	41	32	1 200 000
80486DX4	1994	52	32	1 600 000
Pentium P5	1993	100	64	3 100 000
Pentium P54C	1994	150	64	3 200 000
Pentium MMX	1997	278	64	4 500 000
Pentium Pro	1995	337	64	5 500 000
Pentium II	1997	–	–	7 500 000
Pentium III	1999	–	–	9 500 000
Pentium III	1999	–	–	28 000 000
Pentium 4	2000	–	–	42 000 000
Pentium 4	2002	–	–	55 000 000

Beware!

1. The table shows speed for the earlier chips in *Millions of Instructions Per Second* or MIPS. As explained elsewhere in this chapter, this is a very crude means of testing microprocessor speed and is only included here with this warning. The speed in MIPS of these chips will change if the clock speed is changed and you will notice the clock speed has been left out. This is because comparing chips on clock speed is an even cruder means of evaluating performance. A 100 MHz 80486 is not four times slower than a 400 MHz Pentium, the speed difference is in fact a little tricky to pin down but it is certainly more than eight times slower. Using MIPS for later chips is even more misleading so has been left out.

2. The width of the data bus can also be given an inappropriate level of importance when comparing microprocessor performance. An 8-bit microprocessor will take many times longer to perform a 16-bit by 16-bit multiplication with a 32-bit answer than a 16-bit microprocessor, the 8-bit machine will take dozens of instructions, the other will take 1 instruction. Couple this with the fact that the 16-bit machine will perform this instruction in fewer clock cycles and the clock is running faster and you can see the microprocessor speed is indeed tricky to pin down!

Nonetheless, the table shows the remarkable development over nearly 30 years.

Moore's law

In 1965, Gordon Moore was the Research Director of the electronics company Fairchild Semiconductor. Some 3 years later he was to become one of the founders of Intel. He made an interesting prediction based on what had happened up to that time with memory chips (not microprocessors). He noticed that memory capacity doubled in every 18–24 months. He then predicted this would continue, leading to an exponential growth in size and hence computing power. Market trends have shown it to have come true up until now, with memory and with microprocessors; there are predictions that it will fail in the future but these predictions have themselves failed in the past as they have been made many times.

Mathematics in action

What is exponential growth?

It is when the next value in a series is multiplied by a factor, not added to. So if you start with a number, say 1 and multiply it by a factor, say 2, you get an answer of 2. If you then repeat this, the number grows slowly at the start but very soon becomes very fast.

For instance: if you start with 1 and keep multiplying by 2, you get the series 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024 and so on. The speed of increase is best displayed as a graph.

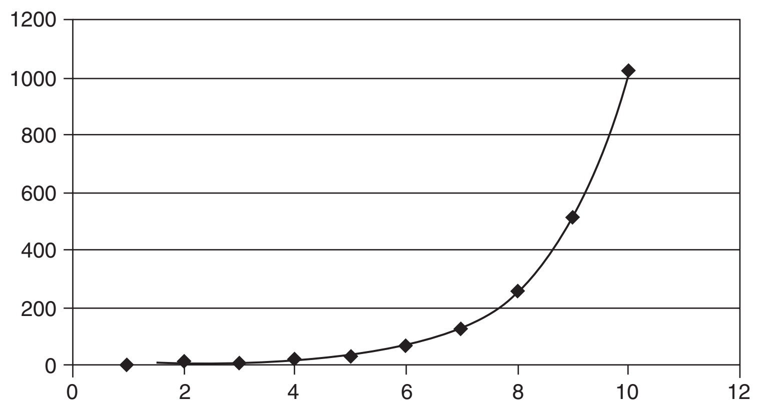

Figure 1.2.1 *Simple exponential growth*

Searching the Internet

Much detailed information on current microprocessor specifications can be seen on the Internet. The problem with giving too many URLs is that they are not guaranteed to exist after this book is printed. If you make good use of a search engine, you will find them very quickly.

There are different types of search engine so:

You could use a *keyword search engine* such as www.google.com, with a search string of

+microprocessor +history

The + signs show the word must be present (it is like the boolean AND operator)

You could also use a *subject-based search engine* such as Yahoo Advanced (search.yahoo.com/search/options) with the search string

microprocessor history

in the 'all of these words' box.

Finally you could choose a *meta search engine* such as Metacrawler (www.metacrawler.com/), the search string will be

microprocessor history

make sure the 'all' box is checked.

Be *very* careful who you believe!

1.3 What is a microprocessor?

A microprocessor is a complex circuit built from large number of simple circuits containing *transistors*, *diodes*, etc. made to perform *logic*. A Pentium 4 has approximately 55 million transistors, 10 times the Pentium Pro used as an example here in the first edition of this book! There are two key properties that a microprocessor has:

It can execute logical instructions in *sequence*

and

It can make *decisions.*

The instructions to carry these out (the program) are separately stored outside of the circuit in *memory*. The decisions or instructions are very modest in human terms, they usually take the form something like 'if number A is bigger than number B then execute instruction K else execute instruction X' or 'add 6 to number A'. The logical instructions are executed in sequence by the microprocessor, each is fetched from memory then executed, one at a time (complex modern microprocessors can execute several instructions at once). No *single instruction* does anything really complex like 'move paragraph to the bottom of the document', they all do relatively simple tasks. Complex tasks are built from hundreds or thousands of these simple tasks, just like a town is built from thousands of bricks; the town is complex, the bricks are simple.

All the operations of the microprocessor are controlled by a *clock* which means that a constant number of pulses or 1s and 0s are fed to the circuit and each instruction is executed on each pulse. A clock in this sense has nothing

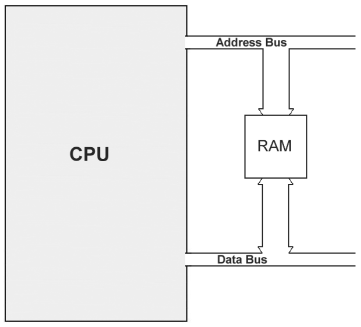

Figure 1.3.1 *Connection of a simple CPU and external RAM*

to do with devices that tell the time! When you see that a Pentium processor has a clock speed of 600 MHz, it means that 600 000 000 clock pulses are supplied to the circuit per second. A quick thought will tell you that if things are happening that fast, why is it that you can watch some operations take some time to execute. The answer is simple, complex tasks are made from a large number of very simple tasks, the simple tasks get executed at a speed hard to relate to human experience but there are *so* many to execute!

In Figure 1.3.1, there are two components, the CPU and the RAM. All the instructions are stored in the RAM and must be loaded one by one into the CPU. After a single instruction has been loaded, the CPU decides what it means, i.e. it *decodes* it, then *executes* the instruction. This is called the *Fetch–Execute cycle*. There is a clock input to the CPU which supplies a series of timed 1s and 0s as a square wave.

The CPU

What follows does *not* describe a real microprocessor. It is a simplified version of the original microprocessors and is presented to demonstrate the way that these devices work. Real microprocessors are more complex but share the same basic way of working.

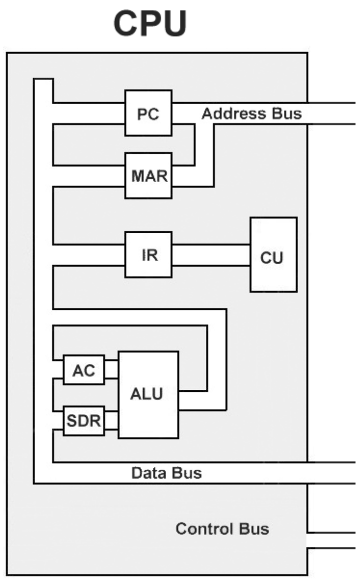

Figure 1.3.2 *Diagram of a simplified CPU*

Registers

The CPU contains *registers*. These are circuits that can remember individual numbers for a short time. Figure 1.3.2 shows these:

- The AC register is traditionally known as the *Accumulator*, it is the register where the results of calculations are held.
- The SDR is called the *Store Data Register* and is used to hold data ready for and instruction.
- The IR is the *Instruction Register* and is used to hold the latest instruction fetched from RAM.
- The MAR is the *Memory Address Register* and is used to hold addresses.
- The PC is the *Program Counter* and is used to store the location or address of the next instruction to be fetched.

Buses

The components of the CPU and those outside the CPU are connected together using *buses*. A bus is simply a collection of wires, so an 8-bit bus is just eight wires each carrying 1s or 0s. Remember that a byte is 8-bits so an 8-bit bus could carry a single byte of information. For example, the information at one moment *could* be the letter G which formed part of the data being processed by the CPU. ASCII for G is 64 + its position in the alphabet = 71 decimal or 47 hex. If you convert this to binary you get 01000111. If each of the wires takes on the value 1 or 0 in this pattern, the bus could be said to be holding a letter G. There is no way of telling if the pattern 01000111 is a ‘G’ or not, the ‘value’ of a piece of data is only applied by the software that is using it.

In this simple microprocessor, the data bus is 8 bits wide so the largest number it can store is $2^8 - 1 = 255$. If it is required, you can write programs to handle larger numbers by breaking them down into 8-bit values. The older 8-bit microprocessors did this and is one reason why they are much slower than modern microprocessors, arithmetic was laborious.

Question 1.3.1

If a microprocessor has a 20-bit address bus, what is the maximum size of RAM this can address?

Question 1.3.2

How many address lines are required to address 64 Mb of RAM?

The address bus is also just 8 bits wide. This causes a much more severe restriction on operations than an 8-bit data bus because you can only have 256 addresses. If some of the instructions need data (they usually do), you may have as little as 100 instructions in your program. When you consider that the main executable of Microsoft Word 97 comprises more than 5 million bytes and that this software needs even more support files to make it work, a 256-byte memory is very small! The width of the data bus and the address bus is an important consideration when specifying a microprocessor. The Pentium microprocessor has a 32-bit address bus and a 64-bit data bus. As $2^{32} = 4\,294\,967\,296$, a 32-bit address will allow 4 294 967 296 different addresses or 4000 Mb or 4 Gb.

Control unit

The control unit is the 'heart' of the CPU. When fed with an instruction from the IR, the microprocessor responds with the correct action, i.e. the right registers are used and if required, the ALU is brought into use.

Arithmetic and logic unit

The ALU is the *Arithmetic and Logic Unit* and as its name suggests, is where the microprocessor actually performs additions and subtractions and logical operations such as AND and OR instructions.

In this example, the CPU will fetch an instruction on one pulse of the clock, decode it right away and execute the instruction on 1, 2 or 3 of the next clock pulses. The reason it might use 1 or 2 or 3 clock pulses is because some instructions are a little more involved than others. The instructions are held in the RAM in a sequence of numbered locations, each of which is called an address. A possible sequence of instructions are shown in Table 1.3.1.

Fetch–execute cycle

Look carefully at Figure 1.3.3. You will see that the PC contains the value 161. The other registers have values that do not matter. Leaving out quite a lot of detail, the fetch–execute sequence will go like this:

Table 1.3.1 *Program for the simple CPU*

Address in RAM (numbered location)	*Code for instruction or data held in RAM*	*Meaning*
161	3A	Load the data at the next address into the AC register
162	23	Number to load, i.e. data not an instruction
163	3D	Load the data at the next address into the SDR register
164	12	Number to load, i.e. data not an instruction
165	8C	Add the numbers in the AC and SDR registers and store the result in the AC register
166	3E	Store the value in the AC register at the RAM address held at the next two addresses, low byte first
167	6E	Low byte of address
168	01	High byte of address
169	3A	Load the data at the next address into the AC register
16A	45	Number to load, i.e. data not an instruction

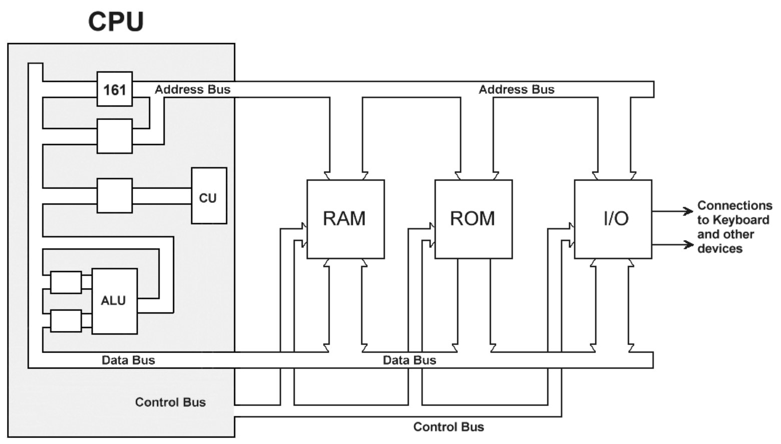

Figure 1.3.3 *The assembled microcomputer*

The fetch part of the fetch–execute sequence

- Instruct the RAM to give the contents of address 161 and place the number found there (3A) into the IR. This is done by putting the value 161 on the address bus and instructing the RAM to read, the value 3A will then appear on the data bus. The control unit provides all the signals for this to happen.
- Allow the IR to feed its value into the control unit.
- The control unit reacts by 'decoding' the instruction 3A which in turn has the effect of putting 162 into the MAR, i.e. the address of the next address where the data is held. The microprocessor is now ready to execute the newly fetched instruction.

Once this is complete, the instruction **Load the data at the next address into the AC register** will be in the CPU, so is the value of the next address but note that it has *not* been executed, i.e. the AC register has not been loaded with the data.

The execute part of the fetch–execute sequence

- Allow the contents of the MAR onto the address bus and instruct the RAM to read. This will result in the value 23 appearing on the data bus.
- Load the contents of the data bus into the AC register.
- Add 1 to the PC ready for the next fetch sequence.

The fetch–execute sequence has now completed and is ready for the next cycle. Remembering this is not a real processor, we can safely ignore some practical details; in this example the fetch sequence took two 'ticks' of the CPU clock and the execute sequence took three ticks. If the clock was running at 1 MHz, i.e. 1 000 000 ticks per second (or better, 1 000 000 1s and 0s), this would have taken 5/1 000 000 seconds or 5 millionths of a second. This is quick in human terms, but all that has happened is that a number has been loaded into a register!

The next instruction in the program is **Load the data at the next address into the SDR register** and this is fetched and executed in a similar way. The next instruction after that is **Add the numbers in the AC and SDR registers and store the result in the AC register**. You should note that this instruction does not have any data associated with it. This means it will take fewer ticks of the clock so will fetch and execute quicker because it does not have to read the RAM a second time.

In general, instructions will take the form of 'what to do' followed by the 'data to do it with' or more formally, *operation code* followed by *operand*. Other instructions will only have the operation code (often shortened to *op code*).

The actual program code and data is of course in binary but we humans do not like to see lists of binary numbers or for that matter, hex numbers. The consequence is that these programs are written down using *mnemonics* for op codes, so the instruction **Load the AC register with the contents of address 14** could be written as the mnemonic **LDA, 14** and the instruction **Add 21 to the contents of the AC register** would be written as the mnemonic **ADD, #21**. The mnemonic **STA, 25** would mean **Store the contents of AC register at address 25**.

A program fragment containing these instructions is:

```
LDA, 14
ADD, #21
STA, 25
```

This means:

load whatever data is at address 14 into the AC register
add the value 21
store the result at address 25

When a program is written in the form of

```
LDA, 14
ADD, #21
STA, 25
```

it is called assembly language.

Machine code

The program and data in RAM would in reality be just a set of numbers. If you were to write them down as numbers (in hex, decimal or binary, numbers are just numbers!), the resulting code is called *machine code*. This is what is actually run in the microprocessor using the fetch–execute sequence. Everything the CPU executes is machine code; when running Windows, the .EXE or .DLL files are machine codes.

Assembly code

If you write down the same program using mnemonics, the resulting program code is called *assembly code*. The reason is that the mnemonics must be converted or 'assembled' into machine code with another program called an 'assembler'. Writing a program using an assembler is much easier than writing directly in machine code. The sequence would be to use a text editor to type the assembly code then use the assembler to generate the machine code. This is then loaded into the RAM and the program run. With luck it will do what you want it to!

If the assembly language

```
LDA, 14
ADD, #21
STA, 25
```

is converted to machine code with an assembler, it might give something like

Address	*Machine code*
234	3A
235	14
236	8C
237	21
238	3E
239	25

What would be stored is just the machine code, i.e. 3A, 14, 8C, 21, 3E, 25. As you can see, machine code is not easy to read.

Actually writing commercial programs using assembly code (often just called assembler) is difficult but one or two applications in computing are still written this way. An example would be very small but include speed critical parts of an *operating system* or special high-speed animation sections of a game. Most programs are written using high-level languages such as C++ or Visual Basic but eventually, after all the compiling and processing of these languages, everything the CPU executes is machine code. *Absolutely* everything!

Assembly language on real processors

It is not the intention of this book or this unit of the HNC Computing to cover assembly language programming but some understanding of terms such as *16-bit operating system* will need to be explained. It is not at all clear why a 16-bit as against a 32-bit program should be different from each other but the section that follows will show there is a very large difference.

The simple processor shown above is not a real practical device. Several key points were ignored in order to put over the main idea of the fetch–execute sequence and what microprocessors actually do. Unlike the simple processor, real processors have a set of *general-purpose registers*, the ability to multiply and divide, to handle *floating point* (real) arithmetic, but they still fetch then execute instructions in sequence. Pentium processors can fetch more than one instruction at a time, can execute whilst fetching, can store instructions in the processor chip and operate in a very efficient way but still fetches then executes instructions even if the sequence is more complex than presented above.

If you need to write assembly language, you will need to know the *architecture* of the chip that will use the resulting machine code and its instruction set, the list of possible operations the chip will execute. Expertise in writing assembler for a Pentium chip will help you need to write for a MC68040 bit only in a general way. One will not directly transfer to the other in the same way that a C++ program would (or at least should!).

For the sake of comparison, one of the original series of Intel chips, the 8086, had four general-purpose registers called AX, BX, CX and DX so the chip could execute an instruction such as

MOV AX, 8000

which would move (or load) the number 8000 into the AX general-purpose register. This register is 16 bits wide so the largest number it can hold is $2^{16} - 1 = 65\,535$. If you were to multiply 8000 by 8, then the answer of 64 000 will still fit in the AX register, so the instruction

MUL 8

will perform the multiplication and place the answer back in AX.

The problem comes when you try a calculation that results in larger numbers than will fit in a 16-bit register. The instructions

MOV AX, 8000
MUL 9

will not give the correct answer as 8000×9 is greater than 65 535. The solution is to use two registers and hold the answer in two parts but you are still limited to the size of number you can handle in this way. It is quite possible to write routines that handle larger numbers, in fact numbers of

```
         PUSH     DS           ;Save caller's DS and DI
         PUSH     DI
         MOV      DI,DSEG      ;Initialise DS
         MOV      DS,DI
         MOV      NEGIND,O     ;Negative indicator 0
         CMP      DX,0         ;Multiplicand negative?
         JNS      CHKCX        ;No. Go check multiplier
         NOT      AX           ;Yes. 2s-comp. multiplicand
         NOT      DX
         ADD      AX,1
         MDC      DX,0
         NOT      NEGIND       ;and is-comp. indicator
CHKCX:   CMP      CX,0         ;Multiplier negative?
         JNS      GOMUL        ;No. Go multiply
         NOT      BX           ;Yes.2s-comp. multiplier
         NOT      CX
         ADD      BX,1
         ADC      CX,0
         NOT      NEGIND       ;and is-comp. indicator
GOMUL:   CALL     MULU32       ;Perform unsigned mult
         CMP      NEGIND,O     ;Is sign correct?
         JZ       DONE         ;Yes. Exit.
         NOT      AX           ;No. 2s-comp. product
         NOT      BX
         NOT      CX
         NOT      DX
         ADD      AX,1
         ADC      BX,O
         ADC      CX,O
         ADC      DX,O
DONE:    POP      DI           ;Restore caller's registers
         POP      DS
         RET
MULS32:  ENDP
CSEG:    ENDS
         END
```

Program 1.3.1 *8086 code for signed 32-bit multiplication with 64-bit answer.*

any size you choose but they require *routines*, i.e. whole sets of instructions rather than a single instruction.

A 32-bit processor can multiply two 32-bit numbers and give a 64-bit answer in a single instruction, a 16-bit processor will use a set of instructions to achieve the same result. See Program 1.3.1. It may seem that a 32-bit microprocessor is twice as powerful as a 16-bit version but in fact the 32-bit one is very much faster. Suppose you had a 16- and a 32-bit microprocessor that ran at the same clock speed and had the same complexity of instructions, the 32-bit machine would handle larger numbers a great deal quicker. Modern 64-bit machines can of course handle even larger numbers; they also execute each instruction quicker.

The assembly language program presented in Program 1.3.1 multiplies two 32-bit signed numbers and gives a 64-bit result. It is written for the 8086 processor, i.e. a 16-bit machine. It is shown here simply to demonstrate the difference between processors of different bit sizes. On a 32- or 64-bit machine, the same result is achieved using a **single instruction**. For the purposes of this unit, you should not attempt to understand how the program works. (Assembly language programmers will note that the top of the program is missing, the so-called assembler directives, etc. The program is shown here only to make the point between microprocessors, not to demonstrate assembly language itself.)

1.4 More complex processors, CISC and RISC

RISC means *Reduced Instruction Set* and CISC means *Complex Instruction Set* (The final C means Chip or Computing depending on what source material you read!)

In a RISC chip, there are fewer instructions which might lead you to believe the chip was in some way less powerful. There are several factors to consider:

- Fewer instructions mean the physical design in silicon is smaller with fewer electronic component and less complexity than a CISC chip so each instruction can execute quicker, the goal being one instruction per clock cycle.
- Analysis of actual program code written for CISC chips shows that only a small fraction of the large number of CISC instructions are actually used frequently.
- Modern ‘RISC’ chips have rich instruction sets.
- The Pentium series of CPUs have an architecture that gives, on average, better than one instruction per clock cycle, close to the original goal of RISC design.

In reality, modern CPUs described as RISC have little in common with the original idea of RISC. There are differences in the design approach of the chips and in the way code is written for each chip but the difference is not simple and clear cut.

Pipelining and superscalar architecture

A technique used to increase the performance of CPUs is *pipelining*. This means that the execution of each instruction is broken down into stages so when the second stage of execution is underway, the CPU can start the first stage of the next instruction *in the pipeline*. In this way, instruction execution overlaps; if an instruction takes five clock cycles to execute and there are five instructions in the pipeline, on average, it will take one clock cycle per instruction, providing a dramatic increase in performance. A CPU is *superscalar* when it has more than one pipeline.

1.5 Latest processors

Books such as this are not the best place to have information about the latest processors, the market moves so fast that by the time a book is in print, the market has moved on. The very best way to get easy-to-understand information about the latest technology is via the Internet. Have a look at the manufacturer's website, ARM, Intel, AMD, Cyrix, Motorola, Texas Instruments, etc.

The main developments are still in the quest for speed but recently there has been an increase in the number of application-specific CPUs for video cards, network switches and routers, etc. These chips are optimized for the particular application.

In the past, PCs had a single processor that did all the processing required to complete a task. Modern CPUs are not only very much faster but the total task is now split between the main CPU, graphics processors, CPUs on storage devices, on network cards, etc. so looked at from the viewpoint of the whole task that small computers are designed to undertake, PCs are now true multiprocessor machines.

1.6 Motherboards and expansion buses, PCI, ISA, etc., chipsets, BIOS and the bootup sequence, POST and operating system loader, I/O ports

What happens when you turn on your PC

When you turn on a PC, it takes some time before it is ready for work. It may not seem obvious but most of the software that runs the PC is stored on a disk and when the power is turned on, this must be loaded into memory before it can be used.

With one or two exceptions, everything that happens in a PC is *controlled* by software. The hardware actually performs the tasks but is 'told' what to do by software. This includes the process whereby the software is loaded into memory, so how does the software get into memory as it is software that does the loading? This was a problem for early computers, the very first ones had people loading a *loader* program by hand. When this was run, it could load the rest of the software. In a PC, this code is always present in a chip called the BIOS or *Basic Input Output System* and this starts to execute shortly after power-up. The BIOS is stored as machine code, usually in a device that acts like a ROM but can in fact be changed. Older machines used devices like *EPROMS* (Erasable Programmable Read Only Memories) which as the name implies, can be erased and then programmed. Newer machines use flash memory but the effect is the same, the machine code is there at start-up time.

An aside

The process of starting a computer is called 'booting' after an old philosophical problem that went something like this. *If I pull on your bootstraps, I can lift you off the ground. If you pull just as hard on your own bootstraps, nothing much happens. Why?* The problem is similar because the computer needs a piece of software (a loader) to load the problem, it is like pulling on its own bootstraps, the loader had to come from outside. For this reason, the original loader was called a 'bootstrap' loader and the process of starting a machine was called 'booting up'. It is not the same as turning on the machine, if you do that, all the electronics work but there is nothing to *control* it without software. Control is the central issue.

Experiment

With your PC turned off, pull out the keyboard connector then power up the PC. The POST will report a problem with the keyboard.

The first process to occur is the *POST* or *POwer up Self Test*. This code, stored in the BIOS chip, tries to obtain a response from each of the main components, video, RAM, keyboard, etc. As failure of the video may stop error messages appearing on the screen, success or failure of the POST is reported by sound: these are the two 'beeps' you should hear shortly after turning on the PC. Usually, if something is wrong, several beeps are sounded, either long or short beeps, the combination of which indicates what has gone wrong. Different makers use different combinations of sound but most use two beeps to mean all is OK.

The next stage is to initialize the system board and load the *interrupt vectors* into RAM (see the section on interrupts below). Having done this, a check is made for other devices and link the BIOS information in those devices to the main BIOS code. Most devices like video cards, disk controllers, etc. have BIOS code of their own. If you boot different PCs with various devices installed, you will see each BIOS display different text, right at the start of the process. If you have a modern power-saving monitor, it may not have started displaying before it 'warms up' so you may miss this. This is not to be confused with operating system text that appears later.

The next process is the *initial program load*. This is where the BIOS goes to a fixed place on a disk, the first sector, then loads whatever it finds there into memory and starts executing. The first sector is called the boot sector because it contains the bootstrap loader for the operating system to be used. At this point, the BIOS hands over control to the operating system stored on your disk. One of its first tasks is to configure the installed add-on boards using *plug and play*. This means that the hardware interrupts and machine addresses to be associated with each board are determined. (In the olden days, this was done when the board was installed by using switches on each board. It was the cause of much anguish and thankfully is now assigned to the history books!)

The operating system itself now loads using its own loader software. As much as some companies would like you to think otherwise, there are several different operating systems that work well with a standard PC (e.g. linux). The loading from now on is specific to each operating system.

Once this is complete, you can start loading application software and use your PC!

What is an interrupt?

Machine code resides in RAM in a set of addresses so each instruction can be fetched then executed in sequence. Decisions made in software cause the sequence to change, so you could have a decision like 'if A is greater than B, jump to address 3000'. If A is not greater than B, the code will continue at the next address in memory. This kind of execution is fine for small problems but is of little use when a large number of time critical tasks must be controlled. Suppose the code was running through the section that updated the video screen to place the next character in the right place. If during this process, a key was pressed on the keyboard, it would be ignored, the CPU would be busy with its video task. Remember, everything is controlled by software, even keyboard presses. A system designed in this manner (called *Polling*) would be of little use.

The solution adopted very early in the life of computers is to use interrupts. This means the CPU can be signalled to stop doing the current task, give some service to the device that interrupted it, then return to the original task. Usually the interrupting task does not take too long to be serviced, so normal operations continue at a good speed.

Interrupts, an analogy

Imagine you were at home waiting for a friend to call round but were not sure of the exact time she was coming. You could do one of two things. Firstly, you could go to the front door once every minute or so and check to see if she had arrived. That way she would never have to wait more than a minute to be let in. Alternatively, you could install a front door bell, continue with your other tasks until it rings, when it does, you stop what you are doing and answer the door. With luck, she has arrived. The first method is called 'polling' and the second method uses an 'interrupt'. Polling implies that the software executes in a fixed sequence and that each device is looked at during regular intervals. It does mean that if busy, the CPU will not service a task, one that could be important. This is the same as your friend arriving just after you have looked out, you miss her until the next 'polling' event.

In a PC, there are many tasks the system must service, so the CPU is interrupted many times per second. Typical interrupts come from the hard drive, the serial port, network cards, etc. and mean that something has happened that needs attending to. In the analogy of you being at home, this means that while you are waiting for your friend, you put the kettle on. It may boil over if you leave it on the gas, so you choose to use an interrupt, the whistle. There is another interrupt, the telephone; another friend could ring at any time. Of course you must decide in advance, which of these interrupts are the most important, what happens if the kettle boils and the front door bell rings at the same time? In a PC, interrupts have assigned priorities to cope with this problem.

When an interrupt occurs, program execution jumps to the right address in memory then returns when finished. In order to know the right address to jump to, the addresses are stored in a table. If the design of the PC was static, this table could be fixed, but every PC is configurable; you can add devices and services in a very flexible manner. This means the table of interrupt addresses must be changeable. At bootup time, a default set of addresses is copied from the BIOS chip into RAM. Later in the bootup process this can be changed to match the particular devices and programs to be used on your PC. The table of address is called the *interrupt vector*, and is situated at a low address in RAM.

Chipsets

The PC system box has three basic elements: the CPU, the memory system and the I/O subsystem.

The mother board is a PCB (*Printed Circuit Board*) that connects these elements together. In the past, these boards were quite large as there was little in the way of integration between the circuits, indeed the very first IBM PCs were built from 'off the shelf' ICs (*Integrated Circuits*) available for the (then) non-PC market. As the PC market grew, the supporting chips became more specific to their task and became known as *Application Specific ICs* or *ASICs*. Since they worked in sets of chips, they eventually became known as the *chipset*. Names have an odd way of sticking around; modern PCs may have all the functionality in one chip but it is still called a chipset. The function of the chipset is to control the flow of data around the motherboard, provide timing signals, handle DMA (see below) requests and a range of other housekeeping tasks.

The particular chipset available depends on the microprocessor socket or slot type. Each microprocessor has its own socket type so your choice of chipset will depend on the CPU type. Not all manufacturers make boards and chipsets to suit all microprocessors.

Most motherboards have circuits to control the mouse, keyboard, input and output from the serial ports, parallel port, floppy disks, and often the IDE hard disks.

Items not generally included in the chipset are video, sound, networks, NICs and expansion buses such as SCSI (Small Computers Systems Interface) controllers, modems and similar devices. These are fitted as extra boards or externally but a few motherbaords do have on board video or sound, etc.

DMA

DMA is *Direct Memory Access*, a system that allows data to be transferred in blocks. In the earliest pre-PC microcomputers, the process of moving a block of data was achieved by moving a byte at a time. The CPU would fetch the instruction, fetch the data byte, fetch the address to send it to, then send the data byte to its destination. This involved many memory operations per byte. Even the earliest PCs had DMA control, circuits that took over from the CPU and transferred whole blocks of data from devices to RAM, providing a dramatic improvement in performance.

CPU sockets

The microprocessor is fitted into a connector on the motherboard. The earliest microprocessors were built into DIPs (*Dual Inline Packages*) with a line of connecting pins down each on two sides. Modern CPUs have so many connecting pins that the package would be far too large, so the pins are now arranged in several rows or rectangular arrays. As microprocessors have developed so have the connectors.

Motherboard ‘form factors’

The size and shape of the motherboard is called the *form factor*. Very old 286-based PCs were called designated AT (meaning Advanced Technology!) and the board form factor was known as the AT board. This form factor was kept for the 486 microprocessors and some early Pentiums. ‘Baby AT’ boards were smaller than the ‘full AT’ boards. Most of these boards have ISA (Industry Standard Architecture) expansion slots, the old 8-bit and the slightly newer 16-bit versions. PCI (Peripheral Component Interconnect) expansion slots came later and are much faster than the ISA versions.

Pentium Socket 7 and the Pentium III chips use the ATX (AT eXtended) form factor and these boards will usually have PCI expansion slots, they may also have some 16-bit ISA slots to allow old (or legacy) boards to be fitted. Some boards use a *riser* where all the connectors and expansion slots are plugged in. The motherboard also plugs into this riser. These are designated NLX (New Low profile eXtended).

Motherboard memory sockets

Modern boards have memory slots of various designs to suit RAM chips (DIMM, SODIMM or RIMM), the older ones would use banks of either 32- or 72-pin SIMM sockets. This refers to the size and layout of the RAM chips. The reason the sockets have changed is to accommodate wider buses. Some boards allow some mixing between chip types but most set-ups avoid this complication.

ISA bus

The ISA bus is the Industry Standard Architecture. The oldest version had an 8-bit data bus and ran at 4.77 MHz, this was followed by a 16-bit version. These are the largest of the expansion slots found on all but the very latest motherboards. Old or legacy devices usually fit this slots. Because the bus runs at such a slow speed, data transfers are slow. The requirement for more and more performance has led to the development of faster buses but the ISA bus is still around to support the many legacy boards still in use. Some devices are so slow that no benefit will be obtained by connecting them to a faster bus.

Local buses

Simple machines use a single address and a single data bus controlled by the microprocessor. This leads to a bottleneck because if one device connected to the buses is transferring data, nothing else can happen. Indeed it is odd to think that most of the time, devices in these simple machine are doing nothing at all. PC have a much more complex architecture, many operations can and do happen at the same time. One of the means by which this can happen is to have a bus in addition to the main bus and allow the CPU to have direct access to it. This feature is called a local bus and there are different kinds available in the marketplace. One of the first was the VESA local bus but this has now been superseded by the PCI local bus. PCI is short for Peripheral Component Interconnect and was first introduced in 1993.

This original PCI was a 32-bit bus with a maximum speed of 33 MHz which means that 33 million times a second, a single piece of 32-bit data can flow along the bus to other devices. If it is set to run synchronously with the PC (the usual case), the actual speed is a set fraction of the PC bus speed. In machines that use a bus speed of 100 MHz, the PCI is running at just one-third that speed, 33 MHz. On some motherboards, the PCI bus is set to run asynchronously, i.e. not in time with the PC.

Most motherboards have three or four PCI slots, most will support *bus mastering*. This is where control of the bus is given to devices on the bus and allows data transfers to occur that are not under the direct control of the CPU. The advantage is that data can flow at the full speed of the bus when it is required. An arbitration circuit ensures that no one device can obtain complete control of the bus.

The most common devices found on the PCI bus are video cards, SCSI adapters, and networking cards. Hard disk controllers are on the PCI bus but are connected directly to the motherboard rather than occupying an expansion slot.

You will see the main system bus referred to as a *front side bus*. This is because the bus has two 'sides', one connecting the CPU to the L2 cache and the other, the *back side bus*, connecting the CPU to the main RAM. The back side bus has a typical speed of 66–133 MHz, the front side bus runs at either one-half of the processor speed or the full processor speed. As is the way with computers, the terms front side and back side bus are no longer used by Intel but are still in daily use. More modern terms are given below.

Modern system board layouts

Figure 1.6.1 shows a typical PC board layout.

- The CPU is the Central Processing Unit, i.e. the Pentium or AMD chip.
- The system RAM is the main memory typically fitted as DIMMs or similar.
- The FSB or Front Side Bus is the main communication to/from the CPU. The speed of the FSB is critical to the performance of the whole machine.
- The PCI or Peripheral Component Interconnect Bus is an Intel design to connect adapter cards to the main system.
- The ISA or Industry Standard Architecture Bus is also known as a legacy bus. This is the old, slow 8- or 16-bit bus fitted to original PCs. It has disappeared from some modern boards, it is only included on others to allow the fitting of older devices.
- The AGP or Accelerated Graphics Port is a high-speed link directly to/ from the CPU to allow improved graphics performance. This is missing from some older boards.
- A Bus Bridge is simply a device that 'converts' the signals on one type of bus to another, it will handle speed and signalling differences.

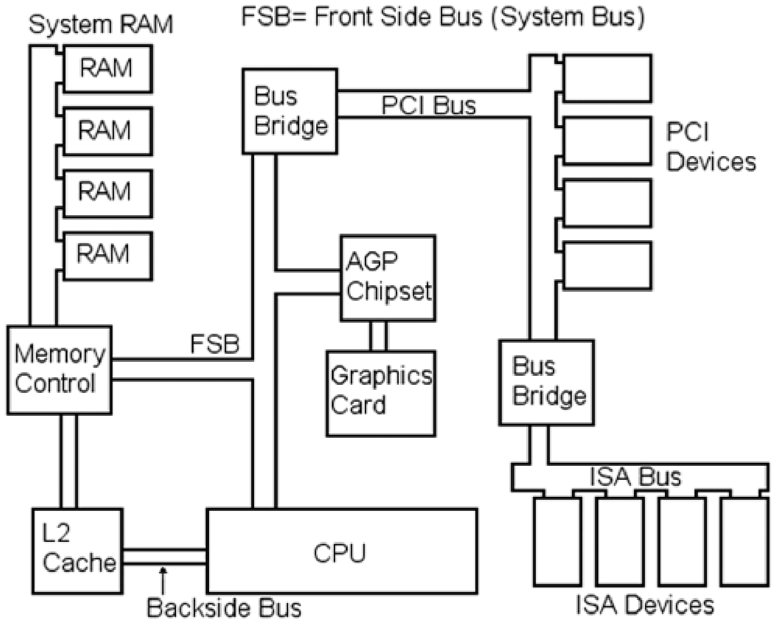

Figure 1.6.1 *Typical PC board layout*

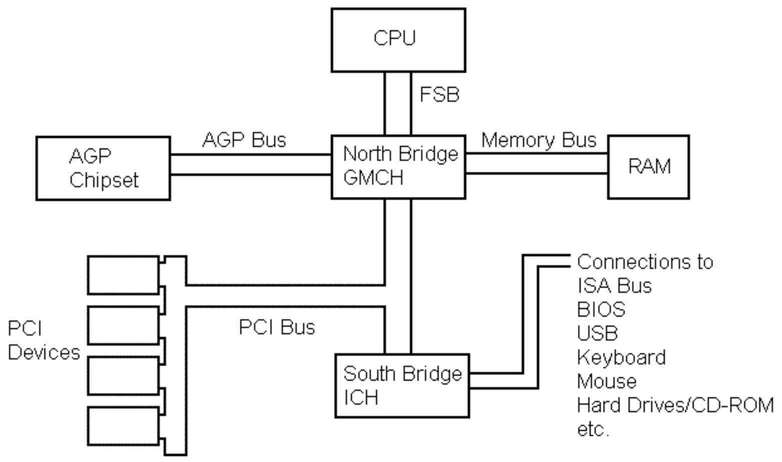

Figure 1.6.2 *PC chipset*

There is another way of looking at a PC motherboard layout. Figure 1.6.2 shows essentially the same PC as in Figure 1.6.1 but in a more modern way. Older PCs were made from numerous chips, assembled to make the complete machine. Modern machines contain most functions in dedicated *chipsets* that are used to connect the CPU with the rest of the machine. This chipset is split into two main parts, the *North Bridge* and the *South Bridge* as in Figure 1.6.2. Intel now calls the South Bridge the *Communications I/O Controller Hub* (C-ICH or just ICH). The North Bridge is now called the *Graphics Memory Control Hub* (GMCH).

The design and speed of the chipset plays a vital role in the overall performance of the machine.

Historical note

With PC specifications things may not always be what they seem. The original Pentium processor with its 64-bit data bus promised to offer PC-users the advantages of 64-bit processing. In fact, Pentium architecture is based on two interconnected 32-bit '486-type processors. When the Pentium was first launched, it was sobering to find that the first generation of these much heralded chips could only just matched the speed of the 'clock doubled' '486 chips that they were designed to replace (real benefits didn't materialize until the much faster Pentiums appeared). As far as memory is concerned and because of its 32-bit address bus, the Pentium is able to address exactly the same range as its predecessor. Not surprisingly, many people who rushed out to purchase the first Pentium-based systems were very disappointed with their performance – there must be a moral here somewhere!

You should be very careful about manufactures claims and be especially wary of 'the numbers', fantastic clock speeds and amounts of RAM. The only real indicator is how fast the machine does the work you want it to perform.

Compatible IDE/ATA (Integrated Drive Electronics/AT Attachment) hard disks (see Section 1.10) can be bus masters. Standard IDE drives use Programmable Input/ Output or PIO mode but bus mastering provides a performance improvement in some circumstances. It does not make everything quicker. Beware, if the hard drive you buy supports bus mastering it does not mean that it will work that way, you must ensure the operating system, motherboard and device drivers also support bus mastering.

AGP

AGP stands for *Accelerated Graphics Port* and is not really a local bus but provides similar benefits, i.e. data can be transferred to and from the AGP without other devices making demands on the circuit. The AGP is entirely independent of the PCI bus on the motherboard.

Even though the PCI bus is fast, the requirement to move very large amounts of data for 3D image effects, texturing, full motion video and ever higher resolution graphics means that the busy PCI bus cannot provide all the speed that is needed.

Graphics cards do a large amount of processing independently of the main CPU and this processing requires memory. Since VRAM (Video Ram) is expensive, the AGP allows the graphics processor to use main memory for these calculations; this results in more data being transferred along a bus from main RAM to the video card, data that is not part of the traffic that 'belongs' to the CPU. The AGP allows this transfer of data without slowing the PCI bus or tying up the CPU.

The original AGP specification was based on the PCI version 2.1 specification, it runs at 66 MHz and uses a 32-bit bus (4-bytes at once). This allows 254 Mbytes/sec. It can be made to run at twice this speed by providing two two data transfers per 66 MHz clock cycle. This is done by changing the way the data is encoded, i.e. by using the rising and falling edges of the 1s and 0s. The latest AGP specification allows data transfers of 2.1 Gbytes/sec .

Peripheral ports

USB

USB stands for *Universal Serial Bus*, a standard being worked on by Compaq, Hewlett Packard, Intel, Lucent, Microsoft, NEC and Philips.

The idea is that peripherals can be plugged in (or removed) whilst the PC is switched on, they do not need to be initialized during the boot up sequence. When a device is plugged in, the operating system recognizes the event and configures the required device driver software.

Many standard PCs are supplied with two USB ports. Attachment of more than two devices is achieved by USB Hubs that allow daisy-chaining, a technique where devices are plugged in one to the next forming a 'chain' thus reducing the amount of cable required. A further reduction in cabling is achieved because USB supplies the power to the devices in the data cable, up to 2.5 watts. Hubs may be cascaded up to five levels deep providing a connection for up to 127 peripheral devices at a transfer rate of either 12 Mb/sec (full-speed) and 1.5 Mb/sec (low-speed). The USB 1.1 standard has been superseded by USB 2.0. This has a raw data rate at 480 Mb/sec, which is 40 times faster than USB 1.1.

Firewire

Firewire is the common name for a standard called IEEE 1394. This is a serial connection aimed at very high data transfer speeds, at least, high for a serial link. Speeds between 100 and 800 Mbits/sec are possible and a speed of 3200 Mbits/sec is promised. Up to 16 devices can be connected via a single Firewire port. It commonly used to attach digital cameras to PCs, one reason being the very simple cable attachment and set-up that is used.

IrDA

This is an infrared serial communication standard that is intended to dispense with cables and run at a maximum of 4 Mbits/sec. IrDA will also work at standard serial speeds to mimic the old RS-232-C serial standard (see below). Since there is a clear possibility of several devices in one place using IrDA and the infrared signal is 'broadcast' around that place, the standard includes techniques similar to those used in networking, to avoid device conflicts and data being sent to the wrong device. It is common to find IrDA on notebook PCs or smaller devices to allow communication with desktop PCs without cabling.

Serial ports

Serial devices have been around for many years. The earliest machines could be connected to devices such as modems or printers using just three wires, a 'send' wire, a 'receive' wire and a signal return wire. Binary 1s and 0s were sent one after the other, i.e. serially. The maximum speed was quite low. To improve speed, extra wires were introduced to allow 'handshaking', signals that allowed or disallowed the sending of data depending in the readiness to receive. These data and handshake lines and the associated timings, etc. were incorporated into a standard called RS-232-C which used a 25-pin 'D' shaped connector. Since only a few of these pins were actually used, IBM introduced a similar 9-pin 'D' connector that is now common on modern PCs. Unfortunately, as a standard that has 'evolved' over the years, the 25-pin connectors are still common as are many different arrangements for interconnecting 25-pin, 9-pin old and new devices. Modern PCs with modern serial devices cause little problem but the use of legacy serial devices with any PC can prove to be problematic. The maximum speed of a serial is currently 115 200 bits/sec. With a simple serial link, each 8-bit byte has a 'start' and 'stop' bit added so using 10 bits/byte. 115 200 bits/sec would then give 11 520 bytes/sec. You may notice that some speeds are given as Mbytes/sec and others as Mbits/sec. This is because the number of extra bits (i.e. not data bits) is variable, depending on the application and the PC industries common practice of quoting the largest number to look attractive in advertisements! Also you should be wary of 'standards'.

Serial ports under Microsoft DOS or Windows have names COM1, COM2, etc. The set-up for these COM (Component Object Model) ports quote the speed in bits/sec, number of data bits, parity and number of stop bits. A typical set-up may be 9600, 8, none, 1. This means 9600 bits per second, 8 data bits, no parity and 1 stop bit. Parity is an old error checking system now little used, it is in the set-up to allow connection with legacy devices. You may see 9600 bits/sec quoted as 9600 Baud but the 'Baud rate' is not the same as bits/sec.

Parallel ports

Most PCs have a single port for the attachment of a local printer. This is a parallel port, i.e. it has control lines and eight data lines, one each for the 8-bits of a byte. Although designed as a single direction port for outputting to printers, some programmers have managed to allow two-way communication. The port is slow by modern standards but as the printers are even slower, no advantage is gained by using a high-speed link.

More modern machines use either an EPP or ECP port. The EPP or Enhanced Parallel Port was designed by Intel, Xircom and Zenith. It allows 500 Kbytes to 2 Mbytes, to be transferred each second, faster than the old Standard Parallel Port, SPP. Microsoft and Hewlett Packard introduced a specification called ECP, Extended Capabilities Port, that was designed to provide improved speed and functionality for printers.

Under Microsoft DOS or Windows, the parallel port is called LPT1 (for Line Printer 1). It is possible to add more parallel ports by plugging expansion cards into the ISA bus, they would then be called LPT2, etc.

1.7 Memory: RAM and ROM

Key fact

RAM

RAM is short for *Random Access Memory* and is one of the silliest names in computing! It really refers to the main memory of the PC, it is where most of the software and data is stored when the machine is in use. It is called random access because any byte can be read into the CPU in any order from any address but this is also true of ROM. A better term would be RWM or Read–Write Memory, indeed some people do use this name.

RAM is volatile, it loses it value as soon as the power is lost. That is one reason it takes so long to boot up a machine, once the power is restored, all the software and data must be loaded.

There are many different kinds of RAM on the market, the result of intense competition to satisfy the ever-increasing demands for speed, capacity and low cost. Some of the more important types are described below but a detailed knowledge is not required to fulfil the criteria for this unit of the HNC Computing.

Key fact

ROM

Unlike RAM, *Read Only Memory* (ROM) cannot be written to by the CPU but it has the advantage of retaining all the data when the power is lost. It is slower than RAM, i.e. it takes longer for the circuits to present data on the data bus after a read instruction from the CPU. It is ideal for the BIOS. As explained in Section 1.6, computers need software to control the loading of software, so software that is already present is very useful!

Types of RAM

For the purposes of the unit of the HNC Computing, a very detailed knowledge of memory types is not required but knowledge of how RAM works and in particular how this affects system performance is important. The points you should remember from the section below are related to this performance and they should inform your purchasing decisions when buying PCs.

Dynamic RAM or DRAM

This is the main type of RAM fitted to PCs, it is cheap but not as fast as other kinds of RAM. In technology there is nearly always a trade-off between conflicting requirements, in this case it is between cost and speed. DRAM is cheap but slow. It suffers from another problem, many times a second the memory contents need to be refreshed, i.e. the chip will forget or lose its contents unless a read or refresh operation is carried out. This means that separate refresh circuits are needed, adding to the complexity of the board.

Static RAM or SRAM

This kind of RAM holds its data without refresh signals and is faster than DRAM, but in the classic trade-off, SRAM loses on cost. It is implemented with between four and six transistors whereas DRAM is made from one transistor and a capacitor. This may not sound expensive but the six transistors are for just one bit! A byte is 8 bits, so 1 Mb of SRAM would take $6 \times 8 \times 1024 \times 1024 = 50\,331\,648$ transistors! Consider that the latest Pentium microprocessors use about a one-seventh of this, although it is really overstating it somewhat as RAM is much simpler to make than CPUs but the scale of the problem is obvious.

How is memory addressed?

In early microcomputers, RAM was addressed directly, so when an address was placed on the address bus, the RAM chips were read in one go and the data flowed onto the data bus. Modern PCs use a more complex arrangement. If you have a 32-bit address bus, this means there are 32-wires arranged in parallel. As $2^{32} = 4\,294\,967\,296$, you can have 4 294 967 296 or 4096 Mbytes of data, assuming one address location can hold 1 byte. 4 294 967 296 in decimal is 100 000 000 in hex. If a hex address of 500 317 is applied to the address bus, it means there is the pattern 10100000000001100000111 on the bus, where a ‘1’ is ‘on’ and a ‘0’ is ‘off’. 10100000000001100000111 in binary is 500 317 hex.

Memory speed

What do we mean by speed with respect to memory?

It is usually taken as the time for the DRAM chips to respond with a data request measured from the time of the request to the time the data is made available. This is quoted in nanoseconds and typical RAM chips have 60 or 70 ns ratings although some are much quicker. A time of 70 ns seems very fast but consider a 600 MHz Pentium CPU, slow by modern standards. 600 MHz means that 600 000 000 clock cycles occur each second. The time between each one is 1/600 000 000 or 1.67 ns, enough time for 70/1.67 = 42 clock cycles! Clearly, in a Pentium 600 machine, it would not be viable to have the CPU read each instruction one by one from the main memory. This is where memory cache is important.

Mathematics in action

What is a nanosecond?

A nanosecond is 10^{-9} seconds or 1/1 000 000 000 seconds. In language, this is a thousand times shorter than a millionth of a second. In human terms, this is an impossibly short period of time but in computing terms, is an 'everyday' unit of time. Table 1.7.1 shows the factors of 10 and the associated names and symbols. Table 1.7.2 shows the factors of 2, you should notice that some values, for instance 10^3 has the same name as 2^{10}, i.e. kilo. but $2^{20} = 1024$ and $10^3 = 1000$ so there is only an approximate equality.

It is sometimes difficult for humans to get a good mental picture of very large or very small numbers. A technique used to help with this problem is to imagine that 1 second is stretched out over a *whole year* and then to see what happens inside that year. If we take the 600-MHz clock speed as an example and imagine 1 second stretched out over a year. There are 60 seconds per minute, 60 minutes per hour, etc. so in 1 year we get $60 \times 60 \times 24 \times 365 = 31\,536\,000$ or 31.5 million seconds per real year. If 600 000 000 clock pulses occur per real second, we would get 600/31.5 = 19 pulses per stretched out second. In other words, even if we slowed down the Pentium by 31.5 million times, it would still be doing 19 things a second. If we now take a 3 GHz machine, this is 3/0.6 = 5 times faster still, so we would get $19 \times 5 = 95$ things a second!

Question 1.7.1

Assume the average size of a hard disk drive in a PC in the year 2000 is 10 Gb. If Moore's law were applied to average hard drive sizes and continued until the year 2012, what would the average size be in that year?

The answer may surprise you! Answer on page 409.

Nanoseconds and other named fractions

Asynchronous and synchronous DRAM

The DRAM fitted to most older PCs was called *asynchronous*. This means it does not operate in time with, i.e. it is not synchronized with, the system clock signals. This worked satisfactorily with slower machines, but newer machines need better performance. SDRAM or *synchronous* DRAM has its operation tied to the system clock so the timing of what happens is under better control. DRAM chips have typical speeds of 60 or 70 ns whereas SDRAM chips have typical values of 10 or 12 ns, i.e. much faster. Beware of the trap often found in computing, especially regarding speeds. A rating of 10 ns does not mean a system with SDRAM is six times faster than one fitted with 60 ns DRAM. The SDRAM can deliver that much faster but since it is now under the control of system timing,

Table 1.7.1 *Powers of 10 and their names*

Factor of 10	*Value*	*Prefix*	*Symbol*
10^{-18}	0.000 000 000 000 000 001	atto	a
10^{-15}	0.000 000 000 000 001	femto	f
10^{-12}	0.000 000 000 001	pico	p
10^{-9}	0.000 000 001	nano	n
10^{-6}	0.000 001	micro	μ
10^{-3}	0.001	milli	m
10^{-2}	0.01	centi	c
10^{-1}	0.1	deci	d
10	10	deca	da
10^{2}	100	hecto	h
10^{3}	1000	kilo	k
10^{6}	1 000 000	mega	M
10^{9}	1 000 000 000	giga	G
10^{12}	1 000 000 000 000	tera	T
10^{15}	1 000 000 000 000 000	peta	P
10^{18}	1 000 000 000 000 000 000	exa	E
10^{21}	1 000 000 000 000 000 000 000	zetta	Z
10^{24}	1 000 000 000 000 000 000 000 000	yotta	Y

Table 1.7.2 *Powers of 2 and their names*

Power of 2	*Number of bytes*	*Symbol*	*Name*
2^{10}	1024	kb	kilobytes
2^{20}	1 048 576	Mb	megabytes
2^{30}	1 073 741 824	Gb	gigabytes
2^{40}	1 099 511 627 776	Tb	terabytes
2^{50}	1 125 899 906 843 624	Pb	petabytes
2^{60}	1 152 921 504 607 870 976	Eb	exabytes
2^{70}	1 180 591 620 718 458 879 424	Zb	zettabytes
2^{80}	1 208 925 819 615 701 892 530 176	Yb	yottabytes

it may not get access to the bus for long enough so giving a slower effective speed. The settings of the BIOS, especially those that affect system timings, number of wait states, etc. are critical if the maximum performance is to be achieved from the PC. Wait states are when the memory system is told to wait for a clock cycle or more (depending on wait state setting) before acting. The fastest is zero wait states but the RAM in the PC may not be able to keep up. It is quite possible to have all the right hardware fitted to the machine but for it to still not run as fast as it should.

There are several kinds of SDRAM but this unit of the HNC Computing does not require such a level of detail. You may see the following term with respect to RAM types: Extended Data Out (EDO), Burst Extended Data Out (BEDO), Double Data Rate SDRAM (DDR SDRAM), Direct Rambus DRAM (DRDRAM), Synchronous-Link DRAM (SLDRAM).

In several sections of this book you will see warnings about making simple assumptions regarding speed ratings. Be very careful, the actual situation is much more complicated than it appears to be at first. The only real test of a PC is to run real life problems, the speed of sub-systems can become very academic.

Cache memory

As demonstrated above, even fast modern RAM chips cannot deliver anywhere near the speed required to feed a high-speed CPU with data. The solution adopted is to use cache memory.

Analysis of a large number of machine code programs (remember *all* programs run as machine code in the CPU) reveals that most of the time, the next instruction to be executed is next to or very close to the instruction being executed. This means that if a block of the program is copied into a section of high-speed memory, the CPU would be able to access instructions and data very much quicker. The problem is that not all instructions are in the high-speed RAM, the *cache*, so a process must go on 'behind the scenes' to load the correct sections of data from main RAM to the cache. About 90% of the time or even better, the next instruction is already in the cache so only 5% or 10% of memory requests will result in a direct memory read.

On Pentium systems, there are two levels of cache known as L1 and L2. L1 is very high-speed RAM inside the processor chip, L2 is a high-speed RAM cache outside the CPU but still with a high performance. With caching, doubling the speed of the main RAM has little effect on the overall machine performance, the critical speed is how fast the CPU can gain access to the cache.

Virtual memory and the amount of RAM required

If cache memory is very fast, virtual memory is very slow! In old mainframe computers, the main memory was very very expensive. It could be that only 1 kb was fitted (not a 1 Mb!). The programs to be run were much larger so techniques were evolved to have a small amount of code in memory and arrange for pieces of the main program to be loaded from disk only when required. This way programs of any size could be run but as the code pieces were loaded from disk each time, the process was quite slow. This is virtual memory, it is the memory that exists 'virtually', i.e. it is not real. Microsoft uses a similar technique with Windows. Windows needs a fair amount of space for itself and each application you have running needs its own space in memory. In the original PCs there was generally not sufficient RAM so a *swap file* was used. This means that sections of program code in memory that was not needed right away was 'swapped' onto the disk. As far as Windows was concerned, there was now enough memory, but of course, being disk bound, the process is slower. Fitting more RAM to a PC allows Windows to access the swap file less often so dramatically improving performance.

Experiment

You can detect when the swap file is accessed quite often, just keep an eye on the disk drive LED on the front of the PC. If there is much activity when you open a new window, it is likely that the swap file is being accessed.

If you were to load a simple operating system (not Windows!) then load and run a single program, the amount of RAM fitted would have no effect whatsoever. The amount of RAM fitted only affects performance when virtual memory techniques are made less necessary. If everything is in RAM, the disk is not accessed to get more data; accessing any part of main RAM takes the same time so no improvement is made by fitting more.

1.8 Video graphics

Pixels

When a graphical image is shown on a computer screen, it is made up from a large collection of dots. Each of these dots is called a *pixel*, a word which is short for *picture element*.

Pixels are arranged in rows and columns; typical numbers of rows and columns are shown in table below.

Name	*Columns*	*Rows*
VGA	640	480
SVGA	800	600
	1024	768
	1280	1024
	1600	1200

The number of pixels per screen is known as the resolution. The higher the number of pixels, the better the resolution or the finer the detail that can be shown.

Each pixel may have just one colour at a time, chosen from a set of colours. Suppose each pixel could be just one from a selection of 256 different colours. This would mean that a number must be assigned to that pixel. If bright red was colour number 37, then that one pixel would be stored as the value 37. Since 1 byte is made up of 8 bits, the largest number that can be stored in 8 bits is 255. If you include the value 0 then it is possible to store 1 of 256 different colour values in 1 byte, or the pixel can be one of 256 different colours. Another way to state this is that $2^8 = 256$.

If you use less memory than 1 byte per pixel, say 4 bits per pixel (half a byte), then you must be content with fewer colours. With 4 bits, the largest number you can store is 15, so (including the value 0) the number of possible colours is 16 or 24. (Half a byte is called a nibble!)

More generally, the number of colours that are available is given by 2^N. If you choose to use 8 bits per pixel then the number of colours is $2^8 = 256$, or with 24 bits per pixel, the number of colours is $2^{24} = 16.7$ million.

Bytes	*Bits*	*Number of colours*
¼	2	$2^2 = 4$
½	4	$2^4 = 16$
1	8	$2^8 = 256$
2	16	$2^{16} = 65\,536$ (64 k), Hicolor
3	24	$2^{24} = 16.7$ million, Trucolor

Most video graphics systems use a value of N that divides evenly into bytes, so values of 2, 4, 8, 16 or 24 are common, values such as 3, 5, 7 are not. The table below shows the number of colours available:

The term *colour planes* is sometimes used to describe the power of 2 so a $2^{24} = 16.7$ million colour setting would be described as a 24-bit colour plane. This comes from the design of the original VGA graphics card.

16-bit colour is called *Hicolor*, 24-bit colour is often known as *Trucolor* and is used where the better graphical image quality is required. Some scanners are now offering 30-bit colour although you could not realistically expect the full $2^{30} = 1\,073\,741\,824$ colours!

Consider some realistic limitations of human perception of image quality. If you have 16.7 million colours, can you see all of them? There are several answers to this.

1. Humans can perceive approximately 10 million colours.
2. The phosphors in the monitor cannot reproduce all the colours that you can see, for example, a really convincing brown colour is very hard to make.

3. If you have a screen set to 1024 × 768 pixels, you have less than 16.7 million pixels. To have enough pixels, to have one each of 16.7 million colours, you would need a resolution of 4730 × 3547 (maintaining the width to height ratio of 4:3).

The main reason to have 16.7 million colours is not to use them all but to have sufficient shades of each primary colour to reproduce a realistic shaded representation of an object.

Graphics cards

The original design of PCs had no facility to output graphics. The method used get around this problem was to have a separate *video card* or *graphics card* plugged into the main board. This card contained the video RAM and some ROM BIOS that contains the code required to write pixels, etc. More RAM allowed more colours or higher screen resolutions. Modern PCs may have the video card incorporated on the main board or as a separate component.

Modern video cards incorporate a CPU to speed up the graphics process. Imagine this problem: you wish to draw a single line at an angle on the screen and you know the colour and the start and end points of the line. Somehow the position of all the pixels that form the line must also be calculated. If this is done using the main CPU then that CPU is not available for calculating new graphics data so the system is relatively slow. If a CPU on the graphics card is dedicated to this task (known as *vector generation*), the main CPU is available for other calculations so speeding up the process greatly. A better enhancement is obtained when solid in-fill colours are required. A 100 by 100 pixel square needs 100 × 100 = 10 000 pixels to be coloured. When this is done by a dedicated CPU, the main CPU has only to calculate the corner positions, just four points.

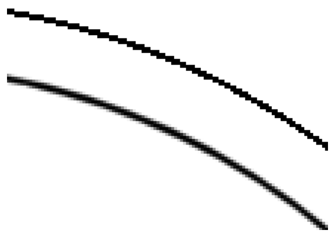
Figure 1.8.1 *Illustration of aliased and anti-aliased image*

Note that non-vertical or non-horizontal lines cannot be quite smooth as they are made up of pixels, a phenomenon known as *aliasing*. The same is true for circles, etc. and extra pixels can be added in different colours to smooth out the line; this is called anti-aliasing. There are a number of algorithms used to calculate these extra pixels which require a fairly large CPU ‘overhead’ and if not done by the graphics card, would seriously slow down the main tasks of the host PC. A disadvantage of anti-aliasing is that lines, etc. become wider so reducing the apparent crispness of detail in some images (Figure 1.8.1).

Most PCs are now sold with at least SVGA or *Super VGA* graphics cards giving 1024 by 768 screen resolution or better and at least 4 MB of video RAM. If you need more colours, then you have the option to increase the amount of video RAM. A video card set to 1024 by 768 Hicolor requires at least 2 MB, if you want this resolution and 24-bit Trucolor, you need 4 Mb as shown below.

Video RAM (VRAM) is a specialized type of RAM that allows *dual porting*, i.e. it allows the CPU to access the RAM at the same time as the video circuits. If a screen refresh of say 85 Hz is used, the video system must access the RAM 85 times a second. VRAM allows the CPU access during the same time.

Video RAM required

A screen resolution of 800 × 600 pixels will yield 800 × 600 = 480 000 pixels. If each pixel needs half a byte it will allow for 16 colours so the screen will require 480 000 × 0.5 = 240 000 bytes of storage.

Question 1.8.1

Calculate the video RAM required for a 1280 × 1024 resolution Trucolor screen.

Question 1.8.2

What is the best resolution possible on a machine with 8 MB of Video RAM and has Trucolor already set?

800 × 600 × 256 colours requires (800 × 600 × 1) = 480 000 bytes of storage because each pixel will need 1 byte. $2^8 = 256$ different colour combinations. 480 000 is roughly half a Megabyte of RAM.

In general, calculate the storage required for 1 pixel remembering that 2^N = number of colours where N = the number of bits required. Next multiply by the number of pixels on the screen.

Display types

The CRT

The glass screen you see uses essentially the same image forming technique as domestic television, the image is made up of several hundred lines of a glowing substance called a 'phosphor'. This substance is made to glow by being 'hit' by a beam of electrons radiating from a point source at the back of the CRT. A fairly sophisticated arrangement of components in the CRT causes the electrons being emitted from the point source to be formed into a thin beam. If the beam were to be kept still, all you would see is a single point of light where the beam hits the phosphor on the inside of the front of the screen. Other components allow this beam to be moved anywhere on the screen, either horizontally or vertically.

A picture is made up by very quickly moving or scanning the beam from one side of the screen to the other in a series of lines that cover the whole of the visible portion of the screen. The detail in the picture is provided by changing the brightness of the fast moving beam. The phosphor has a property that causes it to glow for a little while longer after the beam has passed, this gives the illusion that the screen is evenly illuminated at all times.

Raster scan screens

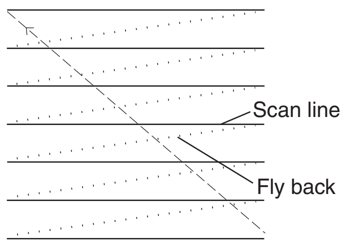

Figure 1.8.2 *Raster scan*

To display a screen image on a CRT, an electron beam is focused on to the front of the screen; this screen is coated with a material that glows when struck by a stream of electrons. The beam is scanned from left to right to form a line across the screen, a scan line. Once each line is formed, the beam is made to return (or fly back) to the side of the screen and down one line, ready for the next line. Once the bottom of the screen is reached, the beam is moved to the top and the process repeated, each screenful of lines is called a *raster*. A picture is produced by changing the intensity if the beam as it traverses the screen and synchronizing this change with the intensity of the image required (Figure 1.8.2).

The number of times the raster is repeated per second is called the refresh rate. Slow refresh rates are seen by humans as flickering so rates of at least 72 times a second or 72 Hz are used to display a steady image, 75 Hz is better. Normal TV screens use a raster scan but are slower than 72 Hz, a fact easily seen if you observe a TV screen in your peripheral vision where movement will be more obvious to you. This is even more obvious if you observe this at an electrical retailer's shop where you have many screens at once. If they are arranged down the side of the shop and you look down the centre of the shop slightly away from the TVs, you will see a very marked flicker in your peripheral vision.

Colour screens

Colour CRT screens use three electron beams, one each for red, green and blue parts of the image.

Chap-01.qxd 2/13/04 4:15 PM Page 33

An electron beam does not have a colour. The achieve colour, small areas of the screen are each allocated red, green and blue parts. These small areas are called slots and each of the three electron beams is made to hit exactly the right point, the 'red' beam hits a point that glows red, etc. The distance between each one is called the *slot pitch* or *dot pitch*. Generally, the smaller the slot pitch the sharper the image because there are more points of light to make up the image. Different shades of each colour are made up by varying the relative brightness of each of the red, green and blue parts of each slot.

Slots are only visible by using a powerful magnifying glass close to the screen. They can be observed on video monitors and domestic televisions and it can be seen that different makers use different shaped slots (shadow mask and aperture grille here).

Experiment

Use a magnifier to view a white portion of a monitor. You will see no white dots, only red, green and blue. All the colours you see are made of these three colours simply by mixing them in different amounts with different brightnesses. It can take some time to come to terms with the fact that there are no white portions of the screen. Most of your visual experience is concerned with your brain rather than your eyes, in this case your brain has 'manufactured' the white you see; your eyes are only receiving red, green and blue light.

Experiment

Observe the slots or dots carefully again with a powerful magnifier. Depending on the make of monitor you are using you will see dots or slots. Notice the slots have no effect on the image pixels, slots and pixels are completely different, pixels are a function of the computer graphics system and slots are fixed by the monitor maker. Move the image from side to side with the horizontal adjustments on the monitor. The slots stay still, the pixels move. You must not confuse slots with pixels; each pixel is a picture element and may take up different physical sizes on one computer by changing to different video modes. The number of slots available on one screen is fixed at time of manufacture of the screen. If you compare two video monitors with the same slot pitch but different sizes, the larger one will have more slots across the screen so will show each pixel more clearly. This is the reverse of the rule with domestic televisions, where larger screens show less sharp images.

Interlacing and refresh rates

On TV systems or old PCs, the beam cannot scan all the lines required to maintain a good number of full images per second, hence cannot provide

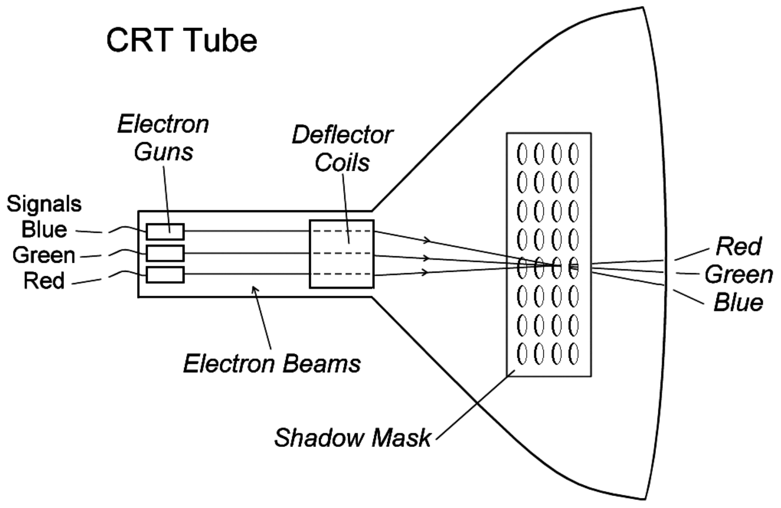

Figure 1.8.3 *Basic layout of a CRT*

an adequate refresh rate. To get around this, a system is used that scans only alternate lines, i.e. lines 1, 3, 5, 7, etc. to the bottom then the beam returns to the top and scans line 2, 4, 6, etc. The result is half the number of full frames per second and is called interlacing; it is the system used on domestic televisions. This is highly undesirable on video monitors as the image can be seen to vibrate slightly. The human response is for the eye to try to keep up with the moving image, a process that leads to sensations of 'tired eyes' because your eye muscles must continuously move your eyes to adjust of the slightly moving image. Better video systems use non-interlaced screens that produce very steady images. Most people now agree that screen refresh rates of 75 Hz or better is required to achieve a good quality, steady image.

Laptop and flat screens

There are many different kinds of LCD screen on the market, but they include:

- passive supertwist nematic displays;
- active-matrix displays.

Passive supertwist nematic displays

Passive supertwist nematic displays make use of a material, a nematic liquid crystal, that is able to change under the application of an electric field. The change in the nematic liquid crystal is in the way that light is polarized when passing through it.

In its normal state with no electric field applied, the polarization of the light that passes through it follows a twisted path. If the liquid crystal is put between two polarizing filters, each one with its axis of polarization at more than 90° to the other, the twisted path of the liquid crystals causes any light from one side to be transmitted out through the other side. When a voltage is applied across the polarizing screens, the light in the liquid crystals no longer travels in a twisted path and so is blocked by one of the polarizing filters. This is used to turn pixels on and off.

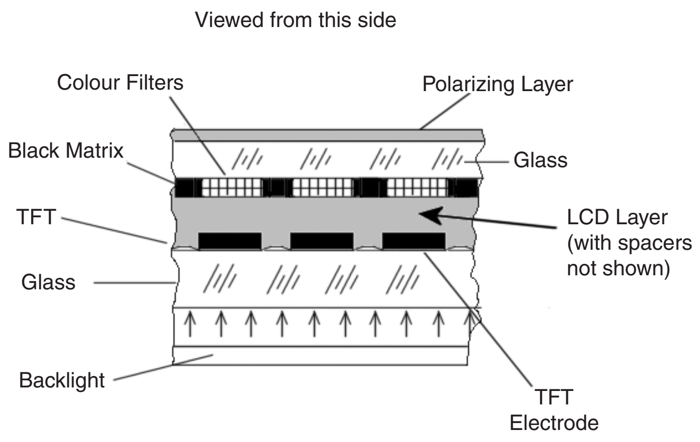

Figure 1.8.4 *Simple diagram of active LCD screen*

Active-matrix displays

Active-matrix displays incorporate transistor to control each pixel on the screen. These TFTs or Thin Film Transistors are made into a matrix, each point in the matrix forms one pixel. Each TFT has opaque parts so not all the area assigned to a pixel is able to produce light. The ratio of the area of light production to the opaque part is called the aperture ratio and is taken as a measure of quality of the screen. A value of 30% is good but a value of 50% is better and this is achieved in the better screens.

The backlight shown is cold cathode fluorescent light source. The light from this is attenuated by the black matrix. The higher the aperture ratio, the brighter the image.

Compared with CRTs, the LCD screens (Figure 1.8.4) found on notebook PCs are:

- physically smaller;
- have lower power consumption (allow batteries to be used);
- lighter;
- display good contrast with bright saturated colours;
- show a crisp image.

There are problems when compared with CRTs; LCD screens have a

- viewing angle;
- slow response time, they are not good for animated games, video etc.;
- high cost, the LCD screen accounts for most of the price difference between a desktop machine and a notebook PC, active LCD dispalys are more expensive than the passive type but produce a much brighter display.

1.9 Printers

What do you need from a printer?

The answer depends of course on the kind of business or task that you are undertaking.

Things to consider are:

- speed of text printing;
- speed of image printing;
- quality of text printing;

- quality of image printing;
- convenience;
- cost of purchase;
- cost of running.

Some of these are easy to assess, i.e. Will the printer produce a good output on normal office quality photocopy paper which is cheap and available everywhere? Is it easy to use and is it supplied with software to allow your computer to communicate with it?

Other factors are not so obvious and advertisements for printers often do not show the real answer; sometimes even the technical specifications do not show the answer. The best solution is to test one or at least see it running.

Print quality

The most commonly used measure of print quality is *resolution* measured in dots per inch (dpi) or dots per mm. Many laser printers have a resolution of 600 dots per inch or approximately 23.6 dots per mm. It is not sufficient to use this resolution as a measure of print quality. Other factors are also important such as how the ink 'sits' on the paper and the sophistication of the printer driver software. At least one major manufacturer of printers claims that the driver software is *more* important than a simple measure of resolution. One feature the use to support this claim is marketed under various names such as 'Resolution Enhancement Technology' or similar. This takes the idea of improving the resolution of individual characters instead of the whole page. The section below shows the large amount of RAM required to store a whole page. If mathematics are used to enhance each letter in a manner similar to anti-aliasing as used in video graphics, sharp letters can be produced by 300 dpi printers with a great saving in the RAM required and the associated processing. It can sometimes be difficult to tell simply by looking at a printed page, the difference between some 300 dpi and 600 dpi printers. Several reputable computer journals and magazines run tests on different printers models. These tests often show that choosing a printer on technical specification, especially the resolution alone, is not a reliable method of printer selection.

Colour reproduction (also see Appendix C, Colour)

The colour of an image on the screen may not closely match the colour of the final printed image. The reason is that one uses coloured phosphors to produce light, the other uses inks and dither patterns to simulate the same colours. Although modern printers do well in matching colours, often much experiment will be required to get the best results for your particular monitor/printer combination.

Question 1.9.1

For a 600 dpi printer, how many dots are there on an A4 page and how much RAM will be required to store this data? Allow a 10 mm margin all round. Answer on page 410.

Memory required

To produce large images, printers need large amounts of RAM.

As each dot on a mono laser can be encoded in just one bit, 1 byte can store 8 dots.

Table 1.9.1 *RAM required for full A4 page with 10 mm margin*

dpi	*Dots*	*Bytes*	*Kb*
300	7 341 900	917 737	896
360	10 572 336	1 321 542	1 291
600	29 367 599	3 670 950	3 585
720	42 289 342	5 286 168	5 162
1200	117 470 395	14 683 799	14 340
1440	169 157 369	21 144 671	20 649

Extending the calculation for other printer resolutions gives the results in the Table 1.9.1. As you can see, if you want a 1440 dpi printer to print a mono full-page image, you need a huge 20 Mb of RAM.

Printer speed

This is an area where great claims are made but real printers often fail to impress. Laser printers and some inkjet printers are quoted as printing ‘N pages per minute’. Eight pages per minute is pretty average for a small laser printer but can you really expect eight full pages to be finished after just 1 minute? Probably not. There are several reasons for this.

1. If you are printing images, does the printer or the PC ‘rasterize’ the image, i.e. turn it into a set of black dots suitable for printing. If this is done in the PC, it will produce a huge amount of data to be downloaded into the printer. The answer to the question above demonstrates just how large the amount of data could be, i.e. about 30 million dots. This is the extreme case of an image size of the page. In some cases this may take a few minutes, preventing the ‘eight pages per minute’ claim of the manufacturer. If done in the printer, it will probably be faster. See Methods used to control printers below.
2. Does the printer already contain all the fonts you are using? If not, each font will be downloaded which takes time.
3. In the case of a laser, has it warmed up yet?

If you start with a cold printer, i.e. just turned on and have a system where the PC rasterizes images instead of this being done in the printer, the time from when you ask a page to print to the time it is in the out tray may be say 4 minutes. If the manufacturer claims eight pages per minute, it should have been in the tray after less than 8 seconds. The truth is that the speed is quoted as the ‘engine speed’, i.e. the maximum throughput speed of the actual printing mechanism once the data has been prepared for printing. The quoted speed is almost reached if you measure it once the printer has started producing multiple copies of the same page.

At full speed, even an eight page per minute laser printer will outperform an old dot matrix printer *but* if you time a small print job, say a single invoice, from the time of request to the finished item, dot matrix printers are often faster, they do not suffer from the start up delay found in laser printers. If the task you must solve involves this kind of printing, especially if you need a carbon copy of every document, the old dot matrix machine still perform well.

Be very careful with claims of printer speeds.

Kinds of printer commonly used with PCs

Dot matrix	Cheap and cheerful and quite old fashioned but they are used in point of sale machines or tills and in many businesses for printing invoices at point of sale, simply because dot matrix printers will make carbon copies.
Inkjet	Cheap to buy, very popular for home use with the better models producing convincing photo quality. Will print on many different kinds of media but ink cartridge prices make it expensive to run.
Mono laser	Best quality from reasonably priced printer, makes a good crisp image at a good speed on cheap paper, good life from a toner cartridge.
Colour laser	Makes a good crisp image at a good speed on cheap paper. Machines currently more expensive than mono machines but prices are becoming more realistic.
Thermal wax	Good colours.
Dye sublimation	Close to photographic quality prints. Uses transparent colours allowing good colour reproduction (see Appendix C, Colour).

Methods used to control printers

When a computer sends data to a printing device, the data may take one of several forms. This affects the way the programmers must think about how they are to format their data and how to make the best use of the printer's differing functions; it also affects the speed of response as seen by the user. More complex print formatting requires more sophisticated communication with the printer. When you install software on a machine, one task is to load a *Printer Driver*. This piece of software has the task of taking the data from an application including any mark-up that defines formatting, etc. and translating it to the form the printer will accept. Many modern printers will accept all the forms shown below.

An aside – hexadecimal and binary numbers

Before continuing with this section, make sure you understand hex (hexadecimal) and binary numbers, you will find these in most technical areas of computing. An introduction is included at the end of this section.

Another aside – ASCII characters

Also before continuing with this section, you will need to understand the ASCII character set. An introduction is also included at the end of this section.

Method 1: Send plain ASCII codes to the printer?
The simplest method of outputting data is as a stream of ASCII codes, a set of binary codes that describe each letter. A problem arises when you wish to print graphical images since each pixel must be sent just as a dot, a slow and unsatisfactory process. Another problem arises when you wish to 'control' the printer. One code will send the print carriage back to the left (a Carriage Return, ASCII character 13 decimal, 0D hex), another will advance the paper one line (a Line Feed, ASCII character 10 decimal, 0A hex), but accurate fine resolution control is difficult. Underlining or bold printing is not possible as the printer will print everything it receives. Not a useful method.

Method 2: Send plain ASCII codes to the printer with Escape sequences
This is similar to Method 1 but overcomes the problem of underlining, etc. by using special codes to tell the printer when to underline, print double high, etc. These codes use ASCII character 27 (1B in hex), the *escape*

Question 1.9.2

Referring to the list of ASCII codes and the example above, write down the sequence of codes that should be sent to the printer to

reset the printer,
set italic underlined font,
print the letters 'HNC',
set underline off.

character. For instance, if the word 'CAT' is sent to the printer, the ASCII codes would be 67 65 84, (or 43, 41, 55 in hex) the codes for C, A and T. If you wished to have this underlined you would first send the escape sequence 27 45 1 that is 'ESC','–', '1'. The printer will not try to print these codes, it will simply print everything that comes after it underlined. To turn off the underlining you would send 'ESC', '–', '0'. The full sequence to print an underlined CAT and then to turn off underlining would then be

27, 45, 49, 67, 65, 84, 27, 45, 48, in decimal
1B, 2D, 31, 43, 41, 54, 1B, 2D, 30 in hex

Other examples of escape sequences:

ESC, 'W', 1 to turn on double wide printing, ESC, 'W', 0 to turn it off
ESC, '4', 1 to turn on italic printing, ESC, '5' to release
ESC, '@' resets the printer

Methods 3 and 4: Use a page description language.
The most common method used to talk to modern printers is to *describe* the page layout and contents then send that description to the printer. The printer usually contains its own computer, which interprets the description and forms an image to be printed. This computer is sometimes more powerful than the one used for wordprocessing, etc.!

The page description could be either in the PostScript language or the Hewlett Packard language called *PCL*. However, the when page is described, it must be *rasterized*, i.e. turned into a set of dots; this is the purpose of the computer in the printer. It is possible to do this in the main PC but this slows down the PC as far as the user is concerned and generates huge amounts of data to be downloaded to the printer, further slowing down the process.

The advantage gained by using a page description language is much better control of the printed image. For instance, the original software does not need to know how to place each dot needed to form a line; only where the line starts and ends, the printer works out the placement of each dot in the line, i.e. it generates the whole line from just the end point data. This also means that the printer can hold the shape of each letter in each size (called *fonts*), so the image of each letter does not need to be sent to the printer. If unusual letters are required, they can be sent to the printer hence allowing almost unlimited printing of characters and graphics images.

Examples of both PostScript and PCL are attached.

PCL is based on or evolved from an advanced series of escape sequences but PostScript is a stack-based computer language that may be written 'by hand' or interpreted by other software. You could if you wished write programs in PostScript. Software is available (such as Ghost Script) that will form an on-screen image of the document just by using the PostScript information. A feature of PostScript is that it is not dependant on the printer resolution. This means that if you send a PostScript file to a low-resolution printer it will be rendered according to that printer. If you send the same file to a high-resolution machine you get the same image in the same proportion but looking much sharper. This is the method of choice for many DTP operations. The image is composed on a PC and checked on a local laser printer but the file is sent to a professional typesetter who would use a printer with many times the resolution.

The use of a page description language allows much better control of the printed image and is most suitable for everyday office use but do not forget that not all printers are attached to PCs or bigger machines. The printers in point of sale machines that produce till receipts still need to be controlled as do numerous other small printing devices and many of these are still dot matrix printers. Many businesses choose to keep their dot matrix printers for printing invoices instead of using laser printers, they are simple, robust and very fast for small print jobs because they do not have a long start up time.

Hex or ASCII dumps from four methods

The hexdumps shown below are what were the outputs from Wordperfect 5.1 with different printers set-up. The data in the wordprocessor was just '**This is some sample text, some of it is underlined, some of it is not**.' A dump implies that a data file has been printed in a raw state as hex bytes and as readable ASCII, with no formatting or special treatment. It is used to inspect raw data and is often very useful to help understand what has gone on or gone wrong with a process. Most dumps are like these, with 16 bytes shown in hex and the same 16 bytes shown in the right hand column in readable ASCII with CTRL characters or other non-printable characters shown as dots.

It is clear that it is *not* important to remember details shown here, just that different printers accept data in different forms to achieve similar results.

1. As printed on a printer *with no ability to format text*. Note that all underline codes, etc. are missing, only plain ASCII is present. 0D is the Carriage Return and 0A is the Line Feed character.

```
0D 0D 0A 0D 0A 0D 0A 0D 0A 0D 0A 0D 0A 20 20 20 .............
20 20 20 20 20 20 20 54 68 69 73 20 69 73 20 73 This is
6F 6D 65 20 73 61 6D 70 6C 65 20 74 65 78 74 2C some sample text,
20 73 6F 6D 65 20 6F 66 20 69 74 20 69 73 20 75 some of it is
6E 64 65 72 6C 69 6E 65 64 2C 20 73 6F 6D 65 20 underlined, some
6F 66 20 69 74 20 69 73 0D 20 20 20 20 20 20 20 of it is not.
```

2. As printed on a Panasonic 1123 printer emulating a Epson LQ-850 24 pin dot-matrix printer using escape sequence control. The escape sequences are not the same as the examples above, each printer uses its own which is why a separate *printer driver* must be loaded on your machine to work properly with your printer. A printer driver 'knows' the sequences for the relevant printer.

```
1B 40 1B 36 1B 74 01 1B 52 00 1B 32 1B 6B 06 1B .@.6.t..R..2.k..
21 02 1B 78 01 1B 43 42 1B 2B FF 0A 0D 1B 2B 95 !..x..CB.+ÿ...  + •
0A 0D 1B 24 3C 00 54 68 69 73 1B 24 55 00 69 73 ...$<.This.$U.is
1B 24 62 00 73 6F 6D 65 1B 24 7D 00 73 61 6D 70 .$b.some.$}.samp
6C 65 1B 24 A1 00 74 65 78 74 2C 1B 24 BB 00 73 le.$¡.text,.$≫.s
6F 6D 65 1B 24 D6 00 6F 66 1B 24 E4 00 69 74 1B ome.$Ö.of.$ä.it.
24 F0 00 69 73 1B 24 FD 00 75 6E 64 65 72 6C 69 $∂.is.$ÿ.underli
6E 65 64 2C 1B 24 38 01 73 6F 6D 65 1B 24 53 01 ned,.$8.some.$S.
6F 66 1B 24 61 01 69 74 1B 24 6D 01 69 73 1B 24 of.$a.it.$m.is.$
7A 01 6E 6F 74 2E 1B 24 BB 00 1B 2D 01 20 20 20 z.not..$≫..-.
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
```

3. As printed on a Hewlett Packard LaserJet 5L Printer using the page description language PCL5. Printer control languages actually send a great deal of data to the printer. The text printed was 'This is some sample text, some of it is underlined, some of it is not.' which is only 70 bytes long but the output to the printer produced 504 bytes of data. Note how each escape (1B hex) is followed by a sequence of control bytes, followed by the text to be printed.

```
1B 25 2D 31 32 33 34 35 58 40 50 4A 4C 20 52 44 .%-12345X"PJL RD
59 4D 53 47 20 44 49 53 50 4C 41 59 20 3D 20 22 YMSG DISPLAY = "
57 6F 72 64 50 65 72 66 65 63 74 20 4A 6F 62 22 WordPerfect Job"
0D 0A 40 50 4A 4C 20 53 45 54 20 52 45 53 4F 4C ..@PJL SET RESOL
55 54 49 4F 4E 20 3D 20 36 30 30 0D 0A 40 50 4A UTION = 600..@PJ
4C 20 45 4E 54 45 52 20 4C 41 4E 47 55 41 47 45 L ENTER LANGUAGE
20 3D 20 50 43 4C 0D 0A 1B 26 6C 31 6F 30 6F 31  = PCL...&l1o0o1
74 30 6C 36 64 31 58 1B 2A 72 30 46 1B 2A 63 31 t0l6d1X.*r0F.*c1
30 30 47 1B 2A 76 32 54 1B 26 6C 30 4F 1B 26 61 00G.*v2T.&l0O.&a
30 50 1B 26 6C 30 45 1B 39 1B 2A 70 30 58 1B 2A 0P.&l0E.9.*p0X.*
70 30 59 1B 28 31 30 55 1B 28 73 31 70 31 32 76 p0Y.(10U.(s1p12v
73 62 34 31 34 38 54 1B 26 6C 32 61 30 45 1B 26 sb4148T.&l2a0E.&
6C 32 68 30 45 1B 39 1B 26 6C 30 4F 1B 26 61 30 l2h0E.9.&l0O.&a0
50 1B 26 6C 30 45 1B 39 1B 2A 70 30 58 1B 2A 70 P.&l0E.9.*p0X.*p
30 59 1B 2A 70 33 33 38 59 1B 2A 70 32 32 30 58 0Y.*p338Y.*p220X
54 68 69 73 1B 2A 70 33 33 34 58 69 73 1B 2A 70 This.*p334Xis.*p
33 38 38 58 73 6F 6D 65 1B 2A 70 35 32 39 58 73 388Xsome.*p529Xs
61 6D 70 6C 65 1B 2A 70 37 30 39 58 74 65 78 74 ample.*p709Xtext
2C 1B 2A 70 38 33 34 58 1B 26 64 44 73 6F 6D 65 ,.*p834X.&dDsome
1B 2A 70 39 37 35 58 6F 66 1B 2A 70 31 30 33 39 .*p975Xof.*p1039
58 69 74 1B 2A 70 31 30 38 36 58 69 73 1B 2A 70 Xit.*p1086Xis.*p
31 31 34 30 58 75 6E 64 65 72 6C 69 6E 65 64 1B 1140Xunderlined.
26 64 40 2C 1B 2A 70 31 34 31 35 58 73 6F 6D 65 &d@,.*p1415Xsome
1B 2A 70 31 35 35 36 58 6F 66 1B 2A 70 31 36 32 .*p1556Xof.*p162
30 58 69 74 1B 2A 70 31 36 36 37 58 69 73 1B 2A 0Xit.*p1667Xis.*
70 31 37 32 31 58 6E 6F 74 2E 0C 1B 26 6C 31 68 p1721Xnot...&l1h
30 45 1B 2A 72 42 1B 26 6C 30 70 31 6C 31 68 30 0E.*rB.&l0p1l1h0
6F 30 45 1B 28 38 55 1B 28 73 70 31 30 68 31 32 o0E.(8U.(sp10h12
76 73 62 33 54 1B 26 64 40 1B 45 1B 25 2D 31 32 vsb3T.&d@.E.%-12
33 34 35 58 40 50 4A 4C 20 52 44 59 4D 53 47 20 345X@PJL RDYMSG
44 49 53 50 4C 41 59 20 3D 20 22 22 0D 0A 1B 25 DISPLAY = ""...%
```

4. As printed on a Apple LaserWriter *PostScript Printer*. Printer control languages but especially PostScript actually send a great deal of data to the printer. The text printed was '*This is some sample text, some of it is underlined, some of it is not.*' which is only 70 bytes long but the output to the printer produced 13 867 bytes of data using PostScript from the wordprocessor used, Wordperfect 5.1. When set as 10 point Courier in Word 97, this page description prints on eight A4 pages! What is shown here is only some of the data actually sent to the printer in PostScript, much of it has been left out for clarity. You can see that there is no need for a hexdump as the data is sent as a computer language in plain ASCII, i.e. a description of the results rather than the actual page.

```
%!PS-Adobe
/wpdict  120 dict def
wpdict   begin
/bdef    {bind def} bind def

/bflg    false def
/Bfont   0 def
/bon     false def

.
.
.
```

much of the 13 867 bytes of the PostScript file removed to save space.

```
.
.
.
letter usertime 5000 add {dup usertime lt {pop exit} if} loop statusdict
/manualfeed true put usertime 5000 add {dup usertime lt {pop exit} if}
loop _bp 0 13200 10200 _ornt /HelveticaR 600 _ff
0 13200 10200 _ornt
/_r  { sflg {/_t {0 rmoveto}bdef /ron false def}
     { /_S /show load def /_t {0 rmoveto}bdef /ron false def}ifelse
   }bdef
1200 11849 _m
(This)_S 67 _t
```

```
(is)_S 67 _t
(some)_S 67 _t
(sample)_S 67 _t
(text,)_S 67 _t
_U
(some)_S 67 _t
(of)_S 67 _t
(it)_S 67 _t
(is)_S 67 _t
(underlined)_S _u (,)_S 67 _t
(some)_S 67 _t
(of)_S 67 _t
(it)_S 67 _t
(is)_S 67 _t
(not.)_S _ep
statusdict /manualfeed false put _ed end end
```

1.10 Hard drives

The hard drive in a PC employs a rotating disk or disks. These disks are coated with a material that has certain magnetic properties that are persistent, i.e. once changed they stay that way for a long period until changed again. This is all that is required to store data as 1s and 0s.

The purpose of this part of the book is to provide help with the specification of devices, so the precise details of how the various types of hard drive actually work will be ignored. The most important points to note are:

- capacity;
- performance;
- cost;
- reliability;
- compatibility with other systems/components.

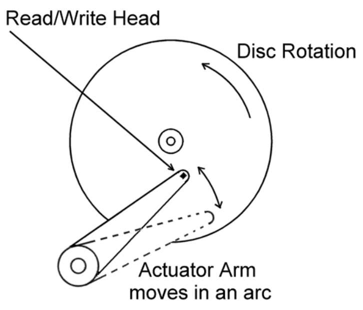

Figure 1.10.1 *Layout of simple disk drive*

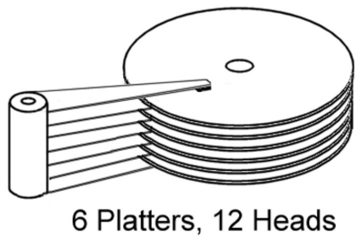

Figure 1.10.2 *Platters and heads*

Although we shall ignore the fine detail, some understanding of how a drive works is required to properly understand factors in the performance of a drive. The disk rotates at a constant speed and a *read/write head* is moved to nearly any point over the surface. If fed with the correct signals, this read/write head is able to affect the magnetic coating of the disk to store a binary 1 or a binary 0 or of course to read 1s and 0s from the disk. Clearly the speed at which this read/write head can be moved over the disk surface and the speed at which it rotates will have an affect on the speed of operation. The read/write head is usually mounted on a radial arm that can swing across the disk surface as shown in Figure 1.10.1. Multiple disks are mounted one about the other and are called ‘platters’. Each has two surfaces so a six platter drive will have 12 read/write heads as shown in Figure 1.10.2.

Disks are formatted into *tracks* and *sectors*. This is a process controlled by software that writes data onto the disk to allow control. Each track is concentric, the number of tracks depends on the bit density achievable on the disk, i.e. how much data can be stored in a small area. A sector is part of a track, it is usually numbered and it contains data and error checking code called a CRC. CRC is short for *Cyclic Redundancy Check*, it is used to test if data loaded from the disk has become corrupted in some way. If a drive has multiple disks or platters, the tracks are called cylinders, the read/write heads mounted on the actuator arm all move together so all the tracks of the same number are one above the other.

Disk performance

Amongst all the performance data for a drive that is available, there are two figures that are the most prominent, the *data transfer rate* and *average access time*.

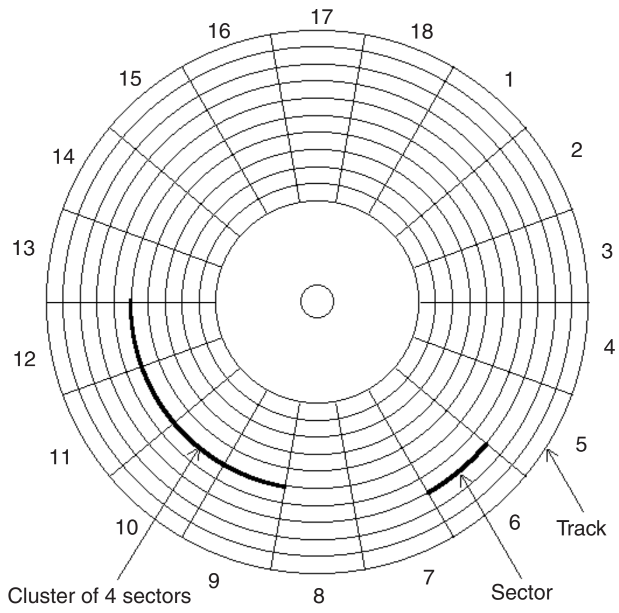

Figure 1.10.3 *Simple arrangement of tracks, sectors and clusters on a disk*

Data transfer rate

The data transfer rate is the speed that data may be read from the drive once it is spinning, the read/write head is correctly positioned and data is flowing, i.e. it is a measure that does not take into account the time to set up the reading process. Really old drives gave values of about 0.6 Mbytes/sec but newer drive can deliver 80 Mbytes/sec or more, about 130 times faster! Advertisers will often quote a *peak* transfer rate, the speed that can be achieved for a short time only. This figure is misleading, it is better to consider the *sustained transfer rate*, the speed the drive can deliver over an extended period, for instance, the time it takes to load a large file into RAM.

Average access time

If a request is made to the drive to supply data from a certain position on the disk, in all probability, the read/write head will not be in the correct place to start the read process. It must wait until the disk has rotated to the correct sector and be placed over the correct track or cylinder. It must then wait for a short time to 'settle' or for any residual movement to stop. If you measured the time it took to complete this whole process, it would not give you a useful figure as the distance the read/write head or the disk had to travel will affect the value obtained. To get around this problem, the process is tried many times; the read/write head is positioned at a random point, a data request is made and this is repeated a thousand times or more to obtain an average time. If a drive has an average access time of say 10 ms, it simply means that on average, the time from a data request to the time the data starts to flow is 10 milliseconds or 10^{-2} seconds, it tells you nothing about how fast the data will flow after that.

To get this time in proportion, consider of a disk that rotates at 3600 rev/min. 3600 rev/min = 60 rev/sec so 1 revolution takes 1/60 second or about 17 ms. Draw an imaginary line through the centre of the disk and through the centre of the read/write head; this line will cut the disk in half. Calling the halves A and B, now consider that on average, the sector that will be needed next will have a 50% chance of being in half A and a 50% chance of being in half B. It follows that the average time to access any sector will be approximately the time taken to rotate half a revolution providing the average is taken over a large number of data accesses. In this case you might expect a drive that rotates at 3600 rev/min to have an average access time of $17/2 = 8.5$ ms. The actual values are slower than this because the actuator arm travel has to be taken into account as does the 'settling time'.

If a drive has a modest data transfer rate but a fast average access time, you can be sure the advertisers will concentrate on the access time. They will say that 'drive x has a time of 8 ms which is faster than the competition value of 10 ms'. The problem you need to consider with buying or specifying drives is 'will I see a difference if I buy the 8 ms drive?' The answer depends on much more than this simple number.

In what follows, take track and cylinder to mean the same thing. Various ploys are used to improve disk performance. One is to increase the rotational speed but this is limited to the speed at which the read/write head and the associated circuits can cope with the data. Too fast and whole rotations are missed until the sector comes round again. Another ploy is to interleave sector numbering. If sector 8 is being loaded, it is a good guess that the next one that will be requested is sector 9. The trouble that once sector 8 has past the read/write head it may be just too late to request sector 9 so a complete rotation is lost. The answer is to number the sectors so they are 'interleaved', i.e. sector 9 is not next to sector 8 but is the next but one. Now when reading sector 8, after is has past, there is enough time within one rotation to request sector 9. A third ploy is called cylinder skewing. In a problem related to the need to interleave, the read/write head must change track at some point to get to the next sector. If the sector numbering goes say 0–25 then 0–25 in the same pattern on the next track, any read that needs sectors 23, 24, 25 then 0, 1, 2, etc. on the next track will result in waiting a full rotation between sectors 25 and 0 as the time it takes to move the actuator arm with the read/write head to the next track is too slow. Again, a full rotation is missed. Cylinder skewing simply aligns each sector 0 on a track differently from its closest neighbour, i.e. it is skewed. The result is a performance increase because the data can be read uninterrupted by track changes.

Disk addressing

The actual sectors on a disk are addressed according to the *cylinder* (or track of one disk), head number and sector number. This is called *CHS addressing* and is what happens inside the drive. Outside and as far as the operating system is concerned, the sectors are most often addressed by their *Logical Block Address* or LBA. The LBA is a simple number starting from 0 and counting all the available sectors. The drive hardware translates the LBA into the actual CHS. This has great advantages over the older system when the operating system itself used CHS, each drive had to be installed according to the number of heads, cylinders and sectors, the sector size (usually 512 bytes) and a few other parameters. You can see these old types listed in the BIOS of a PC as drive type 1 to drive type 46. Type 47 drives are those that are 'user definable'. Thankfully, all that is history. Logical Block Addressing means that drive makers are free to optimize their drives, have any combination of number of heads, cylinders, sectors, etc. and still be compatible with the PC and the operating system. Looked at another way, it places the responsibility of optimum use of the disk with the disk maker not the operating system programmer.

Disk cache

A disk can seem to be very fast if disk caching is present and active. This is a technique that copies sections of the disk into a buffer area in RAM on the assumption that it will be needed. If a subsequent data request for what is already in the cache, it is loaded from there so giving a very fast response. The disk cache can be either on the hard drive itself or in main RAM, if it is on the hard drive the maximum speed it limited by the drive interface, if it is main RAM, it is only limited by the RAM system itself.

Partitioning the disk

In the past, individual physical disks were addressed directly as single devices. This is often inconvenient so modern disks can be divided into logical sections called partitions. In this way a single physical drive can appear to be several different drives. Original DOS drives had a 16-bit FAT which imposes a size limit by having $2^{16} = 65\,536$ addressable elements. As explained below, the only way to address larger disks with a 16-bit FAT is to use cluster sizes greater than 1. This results in inefficient use of disk space so partitioning a larger disk into smaller logical drives can make more efficient use of the disk space, each partition will have its own FAT. The same efficient use of the disk space can be achieved by using a 32-bit FAT. Many people find that multiple disk partitions are a convenient way of organizing their software and data.

Drive interfaces

An important factor in determining disk performance is the *interface* used to connect the drive to the PC. The two competing interfaces that dominate the PC market are *IDE/ATA* and *SCSI.* Broadly, IDE/ATA drives offer a good speed at low cost, SCSI offers higher speed and the ability to address other devices but at a higher price. For this reason, most PCs are fitted with IDE/ATA interfaces but those required for performance critical applications are fitted with SCSI drives.

Key fact

‘Standards’

First a note about names and ‘standards’ in the computing business. A ‘standard’ is really an agreement usually written and published by a approved organization. International bodies such as the ISO do a great deal in this area. There exists a problem in the computing industry, there are many ‘*de facto*’ standards, i.e. standards that exist ‘in fact’ but have not been agreed internationally, they are simply adopted. This approach is fine except that anyone can modify the ‘standard’ and still claim they are conforming. This leads to an ever growing list of ‘standard’ names which can and often does lead to confusion. This is especially true with disk drive interfaces.

IDE/ATA interface

You will see many adverts for *IDE drives*, a term that was originally used to differentiate the old drives that used a separate controller board from the (then) new *Integrated Drive Electronics* (IDE) drives. The trouble is, other devices also have 'integral' electronics so the term is of little use. Drives that are called IDE are really using the 'AT Attachment' standard so should be called ATA but most adverts use the term IDE. IDE/ATA is by far the most popular interface commonly available but variations are also known as ATA, EIDE, ATA-2, Fast ATA or Ultra ATA. The ATA Packet Interface or ATAPI is an extension that allows connection to CD-ROM drives, etc.

Original ATA

The original IDE/ATA supported two hard disks that worked with 'Programmable I/O' or PIO modes and DMA.

ATA-2

With increasing performance demands the next standard, ATA-2 was brought out to support faster PIO, LBA and several other enhancements that need not concern us here. Ignoring a few minor details, ATA-2 is the same as Fast ATA, Fast ATA-2, or Enhanced IDE (EIDE). The differences are those between different manufactures and is an example of a 'standard' being applied differently!

ATA-3

ATA-3 is an improvement to ATA-2 that introduces Self-Monitoring Analysis and Reporting Technology (SMART). Although often confused with it, ATA-3 is not the same as Ultra ATA.

Ultra ATA

Ultra ATA (also called F, ATA-33, DMA-33, etc.) is an even faster interface that uses a 33.3 Mbytes/sec transfer mode using DMA. Ultra ATA is backwards compatible with previous ATA interfaces, so if you fit an Ultra ATA drive to a board that does not support the faster mode, it will still work at the slower speed.

ATAPI

The original ATA would not support CD-ROM drivers or floppy disks so the *ATA Packet Interface* (ATAPI) was introduced to bring the advantage of one standard to cover all the common drives. To make this work, an ATAPI driver must be installed; this is a piece of software called a 'device driver', and it communicates with devices using 'packets' of data. This is because CD-ROMs are quite unlike hard drives in their method of working so software makes up the difference.

PIO

IDE/ATA drives support both PIO and DMA transfer modes. Programmed I/O (PIO) is performed by the main CPU, it requires no support software but ties up the CPU when I/O is in progress. There are several PIO 'modes', each newer than the last, the fastest of which is supported by the latest IDE/ATA drives. As a performance comparison these modes gives maximum transfer rates of

PIO Mode 0	3.3 Mbytes/sec
PIO Mode 1	5.2 Mbytes/sec
PIO Mode 2	8.3 Mbytes/sec
PIO Mode 3	11.1 Mbytes/sec
PIO Mode 4	16.6 Mbytes/sec

DMA

Direct memory access or DMA achieves data transfer without the CPU, either using the old (and rather limited) DMA chips fitted as standard to all PCs or using bus mastering of the PCI bus.

DMA Mode

Single word 0	2.1 Mbytes/sec (no longer used)
Single word 1	4.2 Mbytes/sec (no longer used)
Single word 2	8.3 Mbytes/sec (no longer used)
Multiword 0	4.2 Mbytes/sec
Multiword 1	13.3 Mbytes/sec
Multiword 2	16.6 Mbytes/sec
Multiword 3	33.3 Mbytes/sec (DMA-33)

Do not think that PIO mode 4 and DMA Multiword 2 will give the same overall performance. Both achieve 16.6 Mbytes/sec but the PIO mode ties up the CPU whereas the DMA mode allows the CPU to complete other tasks at the same time, for instance, if an AGP device is fitted. However, an average user will probably not see this performance improvement, it will only be when heavy demands are made on the disk that such things will be useful. The disadvantage of using bus mastering is the added complexity of getting the devices and associated software to work properly.

Small Computer Systems Interface

In contrast with ATA, the Small Computer Systems Interface (SCSI) interface did not start in the PC world nor is it a disk drive interface, it is a standard that is used in Unix machines and others to achieve a fast and flexible means of data transfer between devices. Hard disks are only one of a range of devices that are 'skuzzy' compatible.

SCSI suffers from a similar problem to IDE/ATA, there are too many standards or names giving rise to considerable confusion in the marketplace.

There are main standards, SCSI-1, SCSI-2 and SCSI-3, that are compatible, i.e. older devices that conform to SCSI-1 will work on SCSI-2 adapters.

Table 1.10.1 *Main SCSI parameters*

	Bus width	*Transfer rate Mbyte/sec*	*Bus speed MHz*	*Number of devices per bus*
Regular SCSI-1	8 bits	4.77	5	8
Wide SCSI-2	16 bits	$2 \times 4.77 = 9.54$	5	16
Fast SCSI-2	8 bits	$2 \times 4.77 = 9.54$	10	8
Fast wide SCSI-2	16 bits	$2 \times 9.54 = 19$	10	16
Ultra SCSI-3	8 bits	$2 \times 9.54 = 19$	20	8
Ultra wide SCSI-3	16 bits	$2 \times 19 = 38$	20	16

SCSI-1

SCSI-1 defined a basic 8-bit bus (now called a 'narrow' bus), a 5 MHz bus and 4.77 Mbyte/sec transfer rate.

SCSI-2

SCSI-2 is an extension of SCSI-1 giving a 10 MHz bus, an increased bus width from the original 8-bit SCSI bus to 16 bits and support for 16 devices. It also defines new higher-density connections; unfortunately SCSI has suffered from a large range of 'standard' cables. If you look at hardware suppliers catalogues, you will see a large range of them! A 32-bit bus is defined and is called very wide SCSI but is not in common use.

SCSI-3

SCSI-3 is yet another extension giving a 20 MHz bus, improved Cabling enabling 'wide' SCSI (i.e. not 8 bit!) and for the first time, SCSI-3 contains the new serial SCSI, also called Firewire.

There are three different bus speeds used in SCSI, 5, 10 and 20 MHz, there is the slow and fast SCSI, all this gives rise to some confusion in the marketplace to add to the problems with cabling.

Table 1.10.1 shows a summary of SCSI speeds.

SCSI adapters

Because SCSI is an interface standard rather than a disk drive interface, there is no point in building the SCSI interface into the drive as is the practice with IDE/ATA drives.

The interface is usually on an expansion or accessory card connected either to the (old) ISA bus or more usually to the PCI bus. This expansion card is called a host adapter. Expansion cards connected to the PCI local bus support the use of bus mastering.

Comparative performance of SCSI and IDE/ATA

Some people will say that 'SCSI is better than IDE'. Whilst it is possible to demonstrate that SCSI is faster in some circumstances, in practice the difference is harder to pin down.

- The actual disk drives spin at comparable speeds so the time to get to and access a sector should be the same.

- The fastest SCSI bus is the ultra wide SCSI, which has a maximum transfer rate of 40 Mbytes/sec. This is better than the best IDE/ATA rate of 33 Mbytes/sec. As 40 is only 20% faster than 33, it is unlikely the difference will be spotted except when using demanding applications and they are being timed. It must also be remembered that 40 Mbytes/sec is the maximum of the SCSI interface and not necessarily that of the drive and interface together.
- The SCSI interface has a high 'overhead', i.e. it is much more complex than the IDE/ATA interface; more processing takes place. For this reason, if you ran a comparison of a single IDE/ATA drive running simple tasks against a SCSI drive fitted in an otherwise identical machine, the IDE/ATA drive may well perform better because the interface can react quicker.
- SCSI allows multitasking unlike ATA, so in set-ups with multiple disks that require simultaneous access, SCSI will handle multiple tasks better.
- In PCs fitted with IDE/ATA drives that support bus mastering, the difference between SCSI and IDE/ATA is less marked. In practice, not that many IDE/ATA drives are actually using bus mastering but this situation is likely to change as better device drivers and drives become available.

In summary, for PCs with single drives for desktop use, specify IDE/ATA because it is fast and cheap. For PCs with multiple disks that run demanding tasks or are used as a server, use SCSI.

Size limits

There are various factors that limit the size of disk that can be handled in a given PC. Most modern PCs take drives that are large enough for common applications and plenty of data but if you need more information it can be found on www.pcguide.com/ref/hdd/bios/size.htm.

High-level formatting

Operating systems must use at least one method to keep track of files on a disk. Although disks are formatted with tracks and sectors (the so-called low-level format), there is nothing about this structure that organizes files and directories (directories are called folders in Windows). This organization of files and directories is a function of the operating system, i.e. it is controlled entirely by software, the hardware plays no part at all in the *organization* although clearly it actually executes the task.

There are many different ways to arrange this organization but it is not the intention of this part of the book to describe them all. The section in Appendix D describes one of the simplest and almost certainly the most common, Microsoft DOS File Allocation Table. Operating systems like Windows NT are able to read and write to different disk filing systems whereas older operating systems use only their own. The purpose of the appendix is simply to illustrate the problems that must be solved.

1.11 CD-ROM and DVD drives

CDs work by storing large numbers of 1s and 0s as 'pits' in an optically reflective surface. Unlike hard drives, the 'tracks' are one continuous spiral and the rotational speed of the disk is adjusted to give a constant linear speed past the read head. This is called CLV or *Constant Linear*

Velocity. Hard drives use CAV or *Constant Angular Velocity*, i.e. the rotational speed is constant and the nearer the outside the read/write head is, the faster the linear speed past the head. You can hear the CD-ROM drive change speed as it works, indicating that the read head is moving in or out.

There are many different formats of CD, they conform to standards that are in 'books' named after colours, so you get 'orange book' for CD-R and CD-RW disks or 'red book' for audio or CD-DA disks. This unit of the HNC Computing does not cover such detail except that a drive must conform to a given standard to be useful. For instance, if you buy a CD-RW compatible drive, it will play CD-DA (audio) disks but beware, not all drives support all formats. The actual situation in the marketplace is far from standardized. Likewise, DVD is far from having common standards that are 'standard', i.e. work universally. DVD now stands for *Digital Versatile Disk* but did not start out like that, it was called a *Digital Video Disk*, an indication of the lack of standardization. The original format was for video data but now sound and data are to be found on DVD. Hopefully the 'standards' will become more standard in the near future. Older CD-ROM drives cannot read more modern formats.

Speed

You will have seen advertisements for '×32' or '×40' CD-ROM drives. This refers to the original speed of 150 Kbytes/sec drive. A '40 times' or '×40' drive should deliver a data transfer rate of $150 \times 40 = 6000$ Kbytes/sec or 5.86 Mbytes/sec. The drive listed below is a '×48' version so will provide $150 \times 48 = 7200$ Kbytes/sec. Do not be fooled by this number. It is the maximum the drive can deliver not the actual rate you will get once the drive has spun up to speed, the head has found the data you want and it has started flowing.

The average access time, just like hard drives, is the average time it takes to start delivering data after a data request has been made. Again, this figure is often misleading. Unlike hard drives, CD-ROM drives stop rotating soon after use, so if you measure the actual access time from a stationary disk, the result will be in whole seconds! The spin-up time increases as the 'times' speed increases to a '×48' drive will take longer to spin up to the high speed it needs before the data starts flowing. If your application accesses the CD infrequently, the spin-up time will dominate the average access time but will not figure in the specification at all.

A typical CD-ROM specification would look like this:

1. ATAPI-IDE interface
2. Supports Enhanced-IDE
3. ATAPI compatible
4. Supports ISO 9660 (High Sierra), CD-I, VCD, Multisession
5. Photo CD, CD+, CD-extra, i-trax, CD-UDF, CD-R, CD-RW
6. CD-DA, CD-ROM (mode 1 and 2) format
7. Supports Multiread, CD-RW format
8. Supports Multiread function packet writing format
9. Supports Digital Audio Extraction
10. High-speed audio playback
11. Supports Ultra-DMA mode 2

12. 48× speed drive with maximum 7200 Kbytes/sec transfer rate
13. Average access time: less than 80 ms.

Points 1, 2, 3 and 11 refer to the ATA interface that the drive can use. (See the section on hard drives.)

Points 4–10 refer to the format of CDs to be used. It will play audio disks, record on CD-R and CD-RW disks, play the Kodak Photo CDs, etc.

Point 12 gives the 'times' speed.

Point 13 gives the average access time in milliseconds.

DVD speeds are not the same as CD-ROM speeds, so a '×6' DVD is not slower than the '×48' CD-ROM. A '×6' DVD should be able to deliver 8100 Kbytes/sec or six times 1350 Kbytes/sec. At this rate a DVD will be $8100/150 = 54$ times the CD-ROM speed. Note from the specification below that when reading CD-ROMs and not DVDs, the data transfer rate is a lot lower.

A typical DVD specification would look like this:

1. 6 × (8100 Kbytes/sec) maximum speed at DVD-ROM drive
2. 32 × (4800 Kbytes/sec) maximum speed at CD-ROM drive
3. ATAPI/E-IDE interface
4. Supports PIO mode 4/Multiword DMA mode 2/Ultra-DMA 33
5. Compatible with CD-DA, CD-ROM/XA, Karaoke, CD/Video CD, CD-I/FMV, Single/Multisession photo CD, Enhanced CD and CD-RW
6. Interface type: E-IDE/ATAPI
7. Data transfer rate: 8100 Kbytes/sec maximum at DVD-ROM drive, 4800 Kbytes/sec maximum at CD-ROM drive
8. Average access time: 120 ms at DVD-ROM drive, 90 ms at CD-ROM drive
9. Disk formats: DVD single layer/dual layer, DVD-R, DVD-RW, CD-ROM, CD-ROM/XA, CD-R (CD-WO), CD-I/FMV, Single/Multisession photo CD, Enhanced CD.

Compare this specification with the one for the CD-ROM

CD-ROMs and DVDs either use the ATAPI or SCSI interface. SCSI is better for CD-RW because it offers multitasking and therefore does not interfere with the flow of data to the disk. If you are recording with a CD-RW disk on an ATAPI interface, it is often better not to use the PC for any other task as the recording process is time critical. If other tasks are running that might slow down the flow of data to the writing software, *buffer under-runs* can occur. CDs need a constant stream of data when writing. This data is stored temporarily in a buffer (a piece of memory), ready to write to the disk. If this buffer runs out of data, the disk cannot be fed with a constant stream so the writing process fails. This is called a buffer under-run.

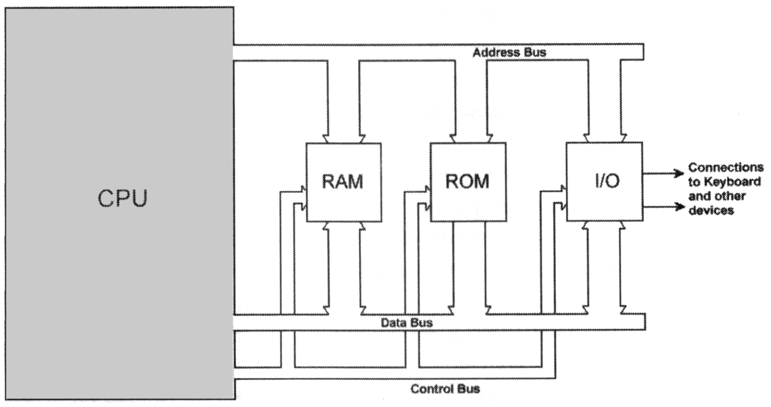

Figure 1.12.1 *A simple microcomputer architecture*

1.12 Computer performance

A human model of CPU performance

The classical model of a simple microcomputer is shown in the diagram below. The effect of this design may not be obvious from the start but the most important thing to notice is that the Central Processor (CPU) is separate from the memory (RAM). This results in the unfortunate fact that the program to be run must be fetched into the processor before the CPU knows what to do and it must be fetched in pieces. With normal programs, in no sense is the whole program ever found inside the CPU. This has quite a profound effect on the performance of the microcomputer because no matter how fast the CPU executes each instruction, the speed at which instructions are fetched from the RAM may well limit the speed at which the whole program is executed.

In this simple model, the processor fetches one instruction, decodes it, then executes it. It then fetches the next instruction, decodes and executes it and so on in an 'endless' sequence. Newer processors fetch instructions a few at a time and these can be executing whilst the processor fetches some more at the same time. Although this provides a very large increase in speed over the old designs, the fact remains that instructions in RAM must be fetched before they can be executed.

Computer performance, an analogy

Warning. Analogies such as this cannot be taken too far. For it to be of use, the humans involved are *not* using their own intelligence, they are simply following instructions blindly, acting as parts of a machine.

Imagine you were in a room with your friend, Fred and you are to perform a calculation. You are the processor, the 'CPU', Fred is the memory or RAM. You must ignore the fact that have intelligence and remember that you do not know what job is required of you; you can add, divide, multiply, keep a note of where you put answers, etc. but you cannot see the whole problem. Your friend Fred, in the same room, knows that a job must be completed but he does not know how to do it, all he knows is that the instructions are written down. He can read the written list of instructions from a reference book. It might go something like this.

Fred opens a tax reference book, finds the correct part then reads to you:

1. take the salary of £20 000;
2. subtract £5400;
3. remember what is left;
4. subtract 23% of this amount;
5. if 6% of what is left over is greater than £1250 then subtract £1250 otherwise subtract 6%;
6. Finally add £5400 to what is left;
7. divide it by 12;
8. tell me the result.

Points to note

The formal list of instructions is the program, it contains the problem.
You are the 'CPU', you do the calculating but do not have a complete picture of the problem.
The *written list* of instructions is the 'RAM'.
Your friend Fred is the connection between the RAM and you, i.e. he is the 'Bus'.

Now consider these points:

1. On a second run through the program, you perform the individual calculations faster but Fred reads them at the same speed, would the overall problem be solved quicker? No it would not, the speed is limited by either the 'RAM' speed or the 'Bus' speed even though you, the CPU, is now quicker.
2. On a third run through the program, if Fred reads the next instruction before you have finished the previous calculation, the speed of execution of the whole problem is now limited by the 'CPU' speed.
3. As a reminder, do not take this analogy too far!

Simple and some more complex computers are like this, the speed of execution depends on more than the speed of the processor. The message is, when you buy a PC, do not be overawed by the processor clock speed, the speed of the whole machine relies on more than this.

Caching

In old PCs with simple processors, the speed was limited more by the RAM because the processor could run faster. Processor design far outstripped the improvements in RAM speed. Modern PCs have RAM that is not all that much faster. How can they offer such a large increase in performance? This is achieved by *caching*. In our analogy, if Fred were to write the program down on a convenient piece of paper before execution then put the main book away, he would be able to access the 'program' quicker rather than looking it up in a book. Of course larger problems would not fit on his piece of paper but most of the time a useful speed increase would be achieved. Attached to a modern processor, there is a small amount of high-speed RAM that is used for this purpose, it is much faster than the main RAM. This is called level 1 or L1 caching.

A second and even better speed increase could be achieved by copying parts of the program onto smaller pieces of paper in your hand, i.e. in the 'CPU'. These are only small pieces of paper but the execution speed is now limited by how fast you can do the calculations, you need not wait for Fred to read them. The problem is that not all the program can be on your piece of paper. Fred must ensure that the next part of the program is available when you need it by copying it whilst you are doing the actual calculations. This is level 2 or L2 caching and is one of the reasons why modern machines are so fast. Caching is what squirrels do, in autumn they hide acorns in a cache for use later on in the winter. This is a similar idea because instructions are put into a cache for use a little later on.

Clock speed

The next thing to consider is the clock speed. In our analogy, the clock speed is roughly equivalent to the time that elapsed between the execution of each simple instruction. A processor does things step by step at a constant speed so if you have a 2 GHz processor, this is doing simple things 2000 million times a second. Although in human terms this seems impossibly fast, remember that each thing that is being done is as simple as 'add 3' or 'store a number', nothing as complex as 'check this spelling'. It does not matter what software is running, it is all made up of very simple machine level instructions called Machine Code. That includes Windows! When Windows is running, the processor cannot tell as all it does is to execute millions of simple instructions. A different analogy is one of bricks making up a building. If you imagine the detail required to understand how a single brick takes part in a building, when you consider Windows in the same analogy you would look at the design of a whole town not just a brick. Machine Code, as far as the processor is concerned, is just a vast array of 'bricks', Windows is the whole town.

What is meant by the term 'speed' in a computer?

If you look at advertisements you will find that companies selling computers quote the CPU speed and expect people to know how fast the machine is from that. For instance, you may think that a 2 GHz Pentium-based PC is 'faster' than a 1.7 GHz Pentium PC. The trouble is that it is often not true. The time taken by a computer to complete a task depends on many factors.

What most non-technical people mean by speed is 'how fast the computer will do a task just for me'. Many of these people would not know or care about the CPU speed. One of these users might be using a large database in which case the main speed of the 'computer' will be limited by the disk drive and memory systems. Another user may be playing a game, the speed here is at least partly limited by the video system.

Imagine you had to describe the complete performance of a car with just *one number*, is it possible? No, in real life it is not possible to define just one such number or *performance indicator*. A Formula 1 racing car may be able to reach 200 mph but if used on a rough farm track competing in a rally it would go several yards before getting stuck in the mud, even a Tractor would be faster! The problem is not just the power of the car, it is the whole design of the vehicle. One car may go very fast in a straight line but slowly around corners, a different car may do well on the corners but slow down the straight. Could you predict which one would win a race without knowing if the race track had lots of corners? No. The problem with defining the speed of the computer is much more complex even than this; you must realize that just quoting the CPU clock speed is almost useless as a measure of speed. There is perhaps one exception to this, modern PCs with faster CPUs tend to be fitted with faster subsystems so may be faster almost by default.

The speed of the processor

If a PC has a Pentium 2 GHz processor fitted it means that the chip is *clocked* or pulsed 2000 million times a second. On almost each one of these clock cycles, a low-level or machine instruction is completed. A typical instruction may be 'store number in memory' or 'add one number'. This means the number of machine level instructions performed per second is *roughly* 2000 million or 2000 MIPS. Something complicated like putting a single character on the screen takes more than one instruction, a task like moving a paragraph in a wordprocessor takes hundreds of machine instructions. Clearly the faster the CPU executes these instructions the better but there is a problem, the instructions are stored in memory so if the CPU needs them in a hurry, the memory system must be able to supply them at speed. The problem is, it can't! To be affordable, the RAM fitted to most machines is far too slow to supply instructions to the CPU. The solution is to copy a block of instructions to a 'cache' and the CPU looks in the cache for them. This cache is faster than the RAM and can (nearly) keep up with the CPU. Defining processor speed in MIPS is only a very crude measure of speed.

MIPS is an acronym for million instructions per second. An old measure of a computer's speed and power, MIPS measures roughly the number of machine instructions that a computer can execute in 1 second. However, different instructions require more or less time than others, and there is no standard method for measuring MIPS. In addition, MIPS refers only to the CPU speed, whereas real applications are generally limited by other factors, such as I/O speed. A machine with a high MIPS rating, therefore, might not run a particular application any faster than a machine with a low MIPS rating. For all these reasons, MIPS ratings are not used often anymore. In fact, some people jokingly claim that MIPS really stands for Meaningless Indicator of Performance.

Some processors have a Complex Instruction Set (CISC) and some have a Reduced Instruction Set (RISC). You could compare the speed of two processors with the same clock speed, one a RISC chip and the other a CISC chip. The problem is that such a comparison of the MIPS is completely meaningless unless you happen to want to sell computers to a gullible public and are happy telling straight lies! The reason is that a single instruction in a RISC processor may achieve more than a single instruction in the CISC processor so you find that a 30 MHz RISC machine executes a whole program faster than a 100 MHz CISC machine. The lesson is, do not compare clock speeds unless the processor has more or less the same design.

The speed of the internal buses

The computer uses *buses* to communicate between devices. These are simply collections of wires that transmit the pattern of 1s and 0s that represent the data and instructions. These buses operate on a clocked system, i.e. many times a second a signal is generated in the chipset that sets off one operation at a time. The buses in common PCs all operate slower than the CPU, typically 'only' 100 MHz or 100 million times a second. The result is that only a certain amount of data can be transmitted per second no matter how fast the CPU. This data may be on its way to the disk drive or the video system and hence some tasks are limited by the bus speed.

The speed of the disk drive system

Disks spin at a fixed speed so the maximum rate at which data can be delivered to or from a disk is partly the result of this speed. If you fit a faster CPU, the rate at which the data comes from a disk drive is not affected.

The speed of the video system

To view the result of running a program, the video system must display a large collection of dots called *pixels*. A screen with a resolution of 1024 by 768 is showing over ¾ of a million pixels, each one must be handled by the video system. If you increase the screen resolution to 1280 by 1024 you will now have over 1.3 million pixels or 66% more so it must take more time to compute the colour of each one. The video system in modern machines have their own dedicated set of chips to do this but some video systems are faster than others, this is not affected by the speed of the CPU.

The speed set in the power save system

If you have ever worked with a laptop computer you will have noticed that to save battery life the disk slows down after a short period of inactivity. If you want your machine to do a task, the disk must then spin up to speed before it starts. This will give the impression that the machine is quite slow, regardless of the speed of the CPU.

Summary

The only effective measure of speed is to run tasks that you require of the computer and to compare one machine with another. Some performance charts in computer journals demonstrate this technique. They would typically run these tasks:

- A large spreadsheet that requires loading and recalculation.
- Transformation of a large graphics image.
- Running a standard database enquiry on a large database.

The results are then displayed as a table. It is often the case that one machine will outperform the others in one task but not others.

Beware of adverts!

Other terms found in describing speed

- FLOPS = *Floating Point Operations per Second*, a common benchmark measurement for rating the speed of microprocessors. As we have seen, the CPU speed alone does not tell you everything about the performance of a whole computer.
- MFLOPS = Mega FLOPS or Million FLOPS, GFLOPS = Giga FLOPS or 1000 Million Flops (1 000 000 000), TFLOPS = Tera FLOPS or Million Million Flops (1 000 000 000 000). Floating-point numbers are those with decimal fractions. Integers whole numbers only.

Floating-point operations include any operations that involve fractional numbers. Such operations, which take much longer to compute than integer operations, occur often in some applications. Most modern microprocessors include a *floating-point unit* (FPU), which is a specialized part of the microprocessor responsible for executing floating-point operations. The FLOPS measurement, therefore, actually measures the speed of the FPU. One of the most common benchmark tests used to measure FLOPS is called Linpack. Many experts feel that FLOPS is not a relevant measurement because it fails to take into account factors such as the condition under which the microprocessor is running (e.g. heavy or light loads) and which exact operations are included as floating-point operations. For this reason, a consortium of vendors created the Standard Performance Evaluation Corporation (SPEC), which provides more meaningful benchmark values. Their Web URL is www.specbench.org/contents.html but many of the ideas shown are very technical and are not part of this unit.

1.13 User requirements

When considering the specification of PCs and associated equipment, there is more to consider than the machine itself. How do you know if you need the wonderful model on sale for £2000 or the lesser model selling for £500? A very large reason for the ever increasing power of computers being purchased is caused by ‘upgradeitus’. Some people will buy the latest computer/software simply because it is available. Operating systems like Windows are very hungry for disk space and RAM and work very slowly unless run in a very powerful machine. People often lose sight of the fact that Windows and many Windows applications offer features most do not even realize are present let alone use on need. In a competitive commercial environment, a sound knowledge of why computers and software are specified is very important. There is no real point in upgrading a system just because it becomes available. As an example, if an application in one office is running perfectly well using an old 80286 PC running MSDOS and a dot-matrix printer, why change it? What *need* is there to change? Simply upgrading the computer is very easy as is upgrading the software but successfully running and paying for the change in work practice and staff training is often very expensive and can be difficult.

Cost of ownership

In order to own and run computers in a business for a period of time, the following items of value must be considered:

- Hardware
- Software licences
- Staff training

- Installation and maintenance
- Business specific data and documents
- Staff experience and knowledge

Which of these is higher? After working for some time, much data is generated in the normal course of the business and much knowledge and experience of the computer systems is built up in the staff. After a very short time, this data and staff knowledge is much more valuable than the costs of the computers. Although the ongoing cost of IT support and maintenance is high, the value of the staff knowledge is probably greater. The cost of the hardware is often the lowest of these and its value falls to zero in a very short time.

It is not sensible to upgrade unless there is a clear business need.

Over the last few years, companies like Microsoft, Lotus, Corel, etc. have put more and more features into their software. This has resulted in the perceived 'need' to upgrade the machines and staff training, often without any real thought. It is interesting to note that the 'Cost of ownership' issue has become very prominent in recent times and that these software companies have started to change their policies, making their software easier to use rather than having more features that require ever more powerful machines.

The graph below (Figure 1.13.1) shows typical proportions of the cost of ownership found in many companies. The cost of the machines themselves is just a fifth of the total. Simply upgrading machines, then the software often causes grief for no real benefit to the organization because the extra training required and the cost of data conversion outweigh any benefits of newer machines. It makes sense to 'over specify' for a new installation so the machines will perform well for a reasonable period but it does not always make sense to upgrade when new hardware or software become available.

General guidelines

Microprocessor

Do not get overawed by CPU clock speeds. In practice, you are very unlikely to see the difference between a 1.7 and a 2 GHz machine, 2 is

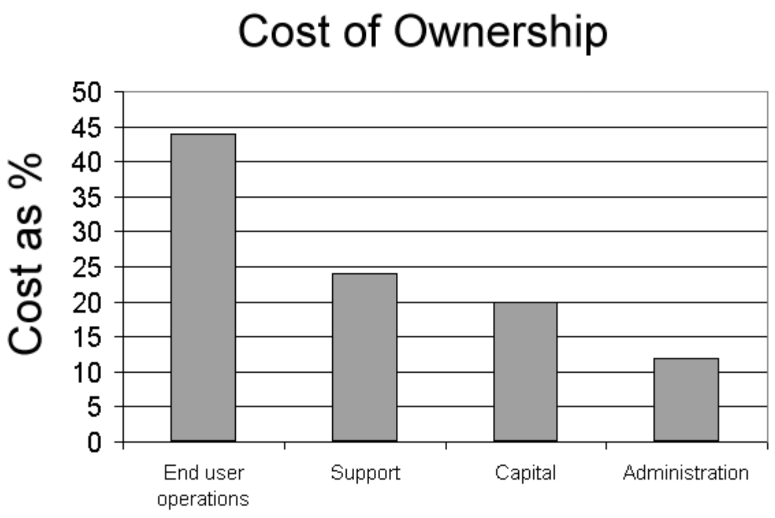

Figure 1.13.1 *An analysis of the cost of ownership*

only 2/1.7 = 1.176 or about 18% faster and the overall speed of a machine is a result of much more than the CPU speed. Machines fitted with faster CPUs are generally fitted with faster sub-systems at the same time so naïve users may be fooled into thinking it is all due to the heavily advertised processor 'inside'.

RAM

If running Microsoft Windows as a desktop operating system, performance improves up to about 128 Mbytes of RAM. Fitting above this is not likely to show a marked improvement unless the applications to be run all need simultaneous open windows or are themselves very demanding on memory. Running a PC as a server is very different but is not in the scope of this part of the book. You will probably not see much difference between the latest RAM types unless you enjoy running benchmark software or timing long tasks with a stopwatch.

Video

The amount of RAM fitted to the video card depends on what you need the machine for. Running standard office applications does not require animated 3D so you calculate the RAM required from the resolution you intend to set. Many people do not like the highest resolution set for office applications, they cannot see the screen fonts, so if the target is the typical 800×600 you will see no benefit from an 8 Mbyte video card. If you plan 3D applications and high-speed animated games, more video RAM the better but you will only see the benefit if the software designers make use of the hardware. More memory will have no effect on older software.

Monitor

Buy a good one! For the overall experience of using a PC, it is better to have a good stable image with crisp resolution and well-saturated colours than something that is a few percent faster. Spend the money you save by not specifying the 'latest, fastest processor' on the monitor. Users will thank you for it. If you run a price comparison on machines with the latest CPU you will see the price rise dramatically towards the faster end of the market. If one machine is 50% more expensive but only 20% faster, spend the difference on the monitor.

Disk drive

A while ago it was thought you could not buy a disk that was too large! This was mainly due to the ever increasing size of the software and data files. Now that MP3, graphics and video are becoming more important, it looks like the demand for disk space will now be caused more by your data than the software. At one extreme, if you are only using the machine for typing plain text and you type at a good 'office' speed of 45 words a minute all day and all night, 7 days a week without a break for 40 years, apart from being tired and hungry, you will still only generate less than 5 Gbytes of data! At the other extreme, you are likely to fill a 60 Gbytes drive very quickly if you store MP3, graphics and video files. One hour of full broadcast quality video can be stored on a 10 Gbyte drive. The best option is to go for size and only buy the more expensive fast drives if the applications *really* need it.

Floppy disk drive

Almost every PC has one, it is hardly ever used but as soon as you specify a machine without a floppy drive, someone will arrive with an important file on a floppy.

CD-ROM

Almost all PCs are fitted with a CD-ROM or DVD drive. CD-R is now cheap and is an excellent system for backups of key data. Unless the budget is very tight, specify a CD-RW compatible drive wherever possible and use it for CD-R writing. DVD drives are gaining acceptance and offer much larger capacity but most software is still distributed on CD. Speed is only an issue if you will use the drive as a continuous source of data; software installation speed is not seriously limited by CD-ROM drive speed. Buy quality rather than speed.

Sound

Specify the cheapest possible sound if the system is for office use. Most users turn it off after a while, especially if the office is open plan. If you need good quality sound to support games or to edit music, etc., it is better to output sound to a sound system than to spend on expensive PC speakers. In this case, buy a high quality sound card and leave the speakers in the shop.

Modem

Modems will soon be a thing of the past, at least that will be true if most people use broadband ADSL or cable modem connections. People who use laptops 'on the move' from hotel rooms, etc. will still need a modem, at least until these places have broadband connections. Most offices use a direct LAN connection. Until this happens, 56K modems are adequate. Buy quality and avoid the 'plain wrapper' kind.

A Modem is a 'MOdulator–DEModulator'. In English to *modulate* means to change (like modify). The old telephone system (known to some as POTS, *Plain Old Telephone System!*) could carry only analogue sound signals. It was not possible to put a digital signal through such a system. Modern digital telephone systems are very different but Modems were designed for the POTS. A modem modulated a sound with digital information and de-modulated this sound for the received signals. The speed of a modem is given in bits/second, most are now 56 Kbits/sec. As each byte is encoded in either 9 or 10 bits, this means a 56K modem will transmit about 5.6 Kbytes of data per second in an ideal world. Real world rates are nearly always slower and having CPUs that are claimed to 'speed up the Internet' will make no difference at all!

Printer

Printers. For home or SOHO use, inkjets are fine; they are expensive to run if your output is high and are not as reliable as laser printers. For office use, the only real choice is a laser printer, they are fast, quiet and reasonably cheap to run. Even for small offices, a printer with a large paper capacity will be appreciated by the users. (SOHO means *Small Office Home Office*.)

How to specify and buy a computer

First, clearly specify why you need a computer, what do you want to do with it and who needs to be able to share your data. Next, decide on your budget.

The next step is to decide on what software you will need to satisfy your business needs and only then to specify what hardware is required to run that software. It is a great mistake to think 'I will need a 3 GHz Pentium' just because they are available and heavily advertised.

Make sure that your user's or customer's requirements are *fully* met. For instance, one customer of the author (a travel tour operator) still uses a mixture of '486' and Pentium 100 machines running Windows 95 and 98. This is simply because, (1) they still work, (2) the specific software and hardware they use will not run under Windows XP. Specifying modern PCs may be a mistake. Even if you convince the hardware vendor to supply without Microsoft operating systems, the hardware sub-systems may not be compatible with the software, e.g. you may not be able to get Windows 98 (required for some old software) to run modern hardware. Even if you manage to get a modern machine and to 'upgrade' their old software, now you have an increased training cost to get staff to use the new system. Some customers are now very resistant to 'upgrading'.

Due to the ever increasing efficiency of hardware production and changes in far east economies, the actual costs of hardware are getting lower and lower. This means that the 'lowest' specification of some computer components now on sale is more than adequate for most people. For example, many machines now come with a disk drive with a 60 Gb capacity as standard. If you only need a wordprocessor this will meet your needs for a very long time. Some people seem to think that if they specify a 120 Gb drive the machine will be 'better' in some respect. This is not true. Many machines are fast enough for normal business activities and more speed is simply not required. Many people find that to run Windows 98 and Word 97 on a Pentium 266 with 32 Mb RAM fitted is quite adequate, why specify any more. In any case, what would you mean by 'more speed' in respect of a wordprocessor, always assuming that a 2 GHz Pentium is somehow 'faster', will it enable you to type faster?

Consider these typical applications running:

- Wordprocessing
- Spreadsheets
- Databases
- Graphic arts
- Technical design

Now consider these things that would appear on a list of components when specifying a computer.

- Disk speed
- Disk capacity
- Video resolution
- Video RAM size
- Monitor size
- Monitor resolution

- Monitor slot pitch
- Monitor refresh rate
- Main RAM size
- Processor type
- Processor clock speed
- Internal bus speed
- Internal bus type
- Motherboard features, buses, expansion slots, etc.

Now write down what is required (allowing for future expansion) rather than what you might 'like' to have. You will find that if you focus more on quality than on performance, the benefits will be higher. Clearly you need a machine that has sufficient performance but you should next consider:

- Cost/budget
- Performance
- Expandability
- Ergonomics, i.e. how well the components 'fit' with the people who will use them
- Needs of specific or specialist software, e.g. Autocad

Some people are specifying notebook computers in place of desktop machines, but you should consider these points that notebooks:

- cost at least 50% more for the same 'power', often twice as much;
- sometimes will not run some specialist software;
- are not as reliable as desktop machines, they are easy to damage;
- are not as expandable or configurable as desktops;
- are very portable (so thieves love them);
- have LCD screen (that many people prefer);
- newer versions of software make ever more demands on the hardware, notebooks go out of date quicker.

Now consider the questions shown in the sample assignment below and discuss your thoughts with your lecturer and fellow students. Keep your mind focused on what is *needed* not on what is desirable or what you may see advertised as 'The Computer Deal of the Century!'

Exercise 1.13.1

Sample assignment tasks

You are asked to specify computers for the six users below. For each of the users listed, choose a suitable machine and justify your choice. You should give a detailed explanation choices in terms of: cost, capabilities, performance and upgrade path.

The simplest way is to list the items fitted or specified in your chosen machine and explain the significance of each item and how it relates to the users' requirements. Write down the machine specification as a list of components in the same way you would present it to a supplier.

User 1

This company supplies artwork, graphics, etc. to the advertising industry, especially the glossy magazine trade. Their main expertise is in photo retouching using very high-resolution images. They only need machines for five graphic artists, the management function in the company is already computerized.

User 2

A small college runs 200 standalone PCs. A network company has offered a sponsorship deal and supplied a full network with cabling and servers to support the college provided that the college upgrades the users' machines. The current 200 machines are to be scrapped. The plan is to run Windows, MS Office and similar software on each user machine but with the user files stored on the servers; they have an extremely tight budget where every penny counts. You must achieve the cheapest possible machine that will run the software.

User 3

The PA to the finance director of a large shipping company requires a machine to do wordprocessing and e-mail. All the other computerized functions in the company are already running elsewhere.

User 4

A very experienced design engineer working on petrochemical plant designs has been on an Autocad course. The projects she works on involve 3D drawings of very complex pipework, etc. During the course of the next year, she will employ two assistants to computerize the existing paper drawings and to use Autocad themselves, so she needs three new networked PCs to run Autocad. The application requires that large amounts of data are stored and that the hidden line removal and other performance critical functions in Autocad are used to full effect.

User 5

A local private genealogy society has computer links to help in their research, they use an old PC with a 56 Kb/sec modem. To reduce costs and speed up enquiries, they have decided to start a large database of family genealogy details. The eventual size of the database may be 200 Gbytes with requirements to have at least one level of backup. Funds are very tight but users will require a good service. To limit the expenditure, only one member will use the machine at a time, linked via a fixed modem on a pre-arranged time slot.

User 6

A financial accountant uses spreadsheets to model the financial behaviour of companies. The spreadsheets are very large and she is hoping to make them even larger but is impatient with the recalculation time obtained with her current computer.