PC Architecture

- a book by Michael Karbo.

These web pages have been produced from a Microsoft Word file. Hence the design, which could be a lot beter. However, it is too much work for me to clean it all up. So you'll have to live with it as it is.

- Chapter 44. Hard disks, ATA and SATA.
- Chapter 45. System software. A small glossary.

- Please enjoy the 45 chapters, all highly illustrated ...

- Next chapter.
- Previous chapter.

PC Architecture. Preface

Foreword

Welcome to a guide which has very been exciting to write. I have spent many years learning to understand PC's and how they work, and this knowledge has been my starting point. I started working with computers in about 1983, and in 1986 I bought my first PC – a small, cheap, British computer (an "Amstrad Joyce"), with a whopping 256 KB RAM and 140 KB diskettes to store programs on. It was a glorious little machine which I used to write a lot of teaching material. As one of the very first things I did, I naturally tried to take the machine apart – the little bit I then dared. In 1987 I got a job which involved working with real PC's (Olivetti's), and this gave me real opportunities to repair, assemble and investigate the various components of a computer.

Since then, I have spent years studying the relationship between PC hardware and system software (BIOS, drivers and operating system). A subject which I found fascinating and which I still believe is important to understand. This lead to my first computer book, "The PC-DOS Book", which was published in January 1993. Since then I have published about 45 guides, on the pc hardware and software.

In 1996 I again began this book. Initially I decided to collect all my articles together on the "Click and Learn" website. The material was extended and translated into both English and German. One of the advantages of the web medium is that the author can continually update the material. And now, after many years, I am finally ready with the "PC Architecture" book. I hope you like it!

Assumptions

This guide is written in easy language and contains a lot of illustrations. It should therefore not be too difficult to understand the content. However, I am assuming that the reader already has some practical experience with PC's and is familiar with the most basic computer jargon (bits, bytes, RAM, etc.). I have also assumed some knowledge of Windows and the various system tools. Most PC's can easily be dismantled without needing special tools, and you should at least take the "lid" off your PC so that you can familiarise yourself with the electronics, preferably with a torch in hand. I'm not expecting you to immediately launch off into the complete disassembly of your PC into its individual pieces. You are welcome to do this, but at your own risk.

However, I would like to give you enough insight into and confidence about your PC's workings that you would dare to upgrade your PC, for example, with a new hard disk or more RAM. And if you should end up building your next PC yourself, I would be more than satisfied.

Structure of this guide

My explanations will shift from descriptions of the big picture to much more detailed analysis – and back again. After focusing on chips deep inside your PC, we may change perspective to more general and holistic observations, only to dig down again into the centre of some or other small circuit. My goal has been to keep my explanations at the popular level, and easy to understand – all the way through.

As I mentioned, this guide can be a great support tool for people who simply wish to build PC's

themselves. But my goal is more specifically to communicate a holistic understanding of the various components of a PC and its data sets, the logical system they are part of, and the technological developments in the field. It is very useful to have this insight, whether you are a programmer, support person, educator, or just a “super user”.

I would like to thank the companies and people who have helped me to obtain accurate, detailed information and photographs. It is not always easy to confirm technical data, so it is possible there may be occasional mistakes in my presentation. Should that be the case, I hope you will bear with me in these. Try instead to appreciate the explanation of the big picture, which it has been my primary goal to set forth!

My thanks to Fujitsu-Siemens, AMD, Intel and others, for pictures and other support! I have taken most of the photographs using my Canon G2 camera, and processed the images using the Photoshop program. The other graphics have been produced using Fireworks. My thanks to Ebbe, Peter, Mikkel, Carl and Jette for their support and, not least, repeated reviews of the manuscript.

The book is protected by copyright

The design ...

Thise web pages have been produced from a Microsoft Word file. Hence the design, which could be a lot beter. However, it is too much work for me to clean it all up. So you'll have to live with it as it is.

I hope you enjoy the book.
Michael B. Karbo. March 2005.
www.karbosguide.dk

- Next chapter.
- Previous chapter.

- Next chapter.
- Previous chapter.

Chapter 1. The PC, history and logic

The PC is a fascinating subject, and I want to take you on an illustrated, guided tour of its workings. But first I will tell you a bit about the background and history of computers. I will also have to introduce certain terms and expressions, since computer science is a subject with its own terminology. Then I will start to go through the actual PC architecture!

1. The historical PC

The PC is a microcomputer, according to the traditional division of computers based on size.

Microcomputers

No-one uses the expression *microcomputer* much anymore, but that is what the PC actually is. If we look at computers based on size, we find the PC at the bottom of the hierarchy.

- Mainframes and super computers are the biggest computers – million dollar machines, as big as a refrigerator or bigger. An example is the IBM model 390.

- Minicomputers are large, powerful machines which are often found at the centre of networks of “dumb” terminals and PC’s. For example, IBM’s AS/400. A definition that was used in the past, was that minicomputers cost between $10,000 and $100,000.

- Workstations are very powerful user machines. They have the capacity to execute technical/scientific programs and calculations, and typically use a UNIX variant or Windows NT as their operating system. Workstations used to be equipped with powerful RISC processors, like Digital Alpha, Sun Sparc or MIPS, but today workstations can be configured with one or more of Intel’s more powerful CPU’s.

- The PC is the baby of the family: Small, cheap, mass-produced computers which typically run Windows and which are used for standard programs which can be purchased anywhere.

The point of the story is that the baby has grown up, and has actually taken the lead! Today, PC’s are as powerful as minicomputers and mainframes were in the past. Powerful PC’s can now compete with the much more expensive workstations. How has the PC come so far?

Fig. 1. Data processing in 1970. Digital PDP 11/20.

The PC's childhood

Let's take a short look at the historical background of the modern PC, which originated in 1981. In less than 20 years, the PC went through a technological development which has surpassed everything we have seen before. The PC has simply revolutionised society's production and communication in just about every sector. And the revolution appears to be set to continue for many more years.

Today the PC is an industry standard. More than 90% of all microcomputers are based on Microsoft's software (Windows) and standardised hardware designed primarily by Intel. This platform or design is sometimes called *Wintel*, a combination of the two product names.

But at the time that the PC was introduced by IBM, it was just one of many 16-bit microcomputers. For example, the company, Digital, sold many of their "Rainbow" machines in the middle of the 1980's, which I have worked with myself. These other machines were *not IBM-compatible*, but they weren't very different from IBM's machines either, since they were all based on Intel's 8088 CPU. There were actually a number of different types of PC in the 1980's.

Fig. 2. DEC Rainbow from 1982. It costed around Euro 8.000 – then!

But over just a few years, late in the 1980's, the market got behind IBM's standards for PC architecture. Using the Intel 8086 and 8088 processors and Microsoft's operating systems (DOS initially, later Windows), the PC revolution got seriously underway. From that time on, we talked bout

IBM-*compatible* PCs, and as the years passed, the PC developed to become the triumphant industry standard.

In parallel with the IBM/Intel project, Apple developed the popular Macintosh computers, which from the very start were very user-friendly, with a graphical user interface. The Macintosh is a completely different platform from the platform of Windows-based pc's I am describing in this guide.

The Macintosh has also been released in generation after generation, but it is not compatible with IBM/ Intel/Microsoft's PC standard.

Fig. 3. An almost IBM-compatible PC from 1984.

In the table below you can see the development of the PC and it's associated operating systems. The PC was actually a further development of the 8-bit microprocessors (like the Commodore 64, etc.), which were very popular until late in the 1980's.

The computer shown in Fig. 2, is a very interesting hybrid. It marked the transition from 8 to 16-bit architecture. The computer contains two processors: an 8-bit Z80 and a 16-bit 8088. This enabled it to run several different operating systems, such as *CP/M* and MS-DOS 2. The two processors, each with their own bus, shared the 128 KB RAM. It was a particularly advanced machine.

Decade	Processors	Operating systems
1970	8-bit microprocessors (initially kits)	CP/M
1980	8086 8088 80286 80386	Various DOS-systems MS-DOS
1990	80486 Pentium K6 Pentium II Pentium III Athlon	Windows OS/2 Windows NT Windows 95 Windows 98 & 2000 Linux
2000	Pentium 4 Athlon XP Itanium Athlon 64	Windows XP Windows XP 64

Fig 4. The microprocessor has entered its fourth decade.

IBM and the PC's success

If we look back at the earlier PC, there are a number of factors which have contributed to its success:

- From the very beginning the PC had a *standardised* and *open* architecture.

- It was *well-documented* and had extensive *expansion options*.

- The PC was *cheap*, *simple* and *robust* (but definitely not advanced technology)

Initially, the PC was an IBM product. It was their design, built around an Intel processor (8088) and adapted to Microsoft's simple operating system, MS-DOS.

But other companies were quick to get involved. They found that they could freely copy the important BIOS system software and the central ISA bus. None of the components were patented. That wouldn't happen today! But precisely because of this open architecture, a whole host of companies gradually appeared, which developed and supplied IBM-compatible PC's and parts.

Clones

In the late 1980's there was a lot of talk about *clones*. A clone is a copycat machine. A machine which can do exactly the same things as an original PC (from IBM), and where the individual components (e. g. the hard disk) could be identical to the original's. The clone just has another name, or is sold without any name.

We don't distinguish as much today between the various PC manufacturers; but they can still be divided into two groups:

- Brand name PC's from IBM, Compaq, Dell, Fujitsu-Siemens, etc. Companies which are large enough to develop (potentially) their own hardware components.

- Clones, which are built from standard components. Anyone can build their own clone, like the one shown in Fig. 15 on page 10.

However, the technology is basically the same for all PC's – regardless of the manufacturer. And this common technology is the subject I am going to expound.

Finally, I just want to mention the term *servers*. They are special PC's built to serve networks. Servers can, in principle, be built using the same components that are used in normal PC's. However, other motherboards and a different type of RAM and other controllers are often used. My review will concentrate primarily on standard PC's.

Bit width

The very first microprocessor Intel produced (the model 4004, also discussed on page 26) was 4 bit. This meant that in a single operation, the processor could process numbers which were 4 bits long. One can say that the length of a *machine word* was 4 bits. The Intel 4004 was a 4-bit processor with a 4-bit architecture. Later came processors which could process 8 bits at a time, like the Intel 8008, 8080, and not least, the Zilog Z80 (a very large number were sold). These were used in a large number of 8-bit computers throughout the 1970's and well into the 1980's.

The PC (in the 1980's) was initially a 16-bit computer. With the development of the 80386 processor, there was a change to the 32-bit architecture which we are still using today.

Now there is a 64-bit architecture on the way, both from Intel (with the *Itanium* processor) and from AMD (with various *Athlon 64* processors). But it is still too early to predict the extent to which the 64-bit architecture will spread into normal, Windows-based PC's.

Width	Processor	Application
4 bit	4004	Pocket calculators

8 bit	8080	Small CP/M based home computers
16 bit	8086, 8088, 80286	IBM-compatible PC's running MS-DOS
32 bit	80386 - Pentium 4	32 bit versions of Windows (Windows 95/98/2000/XP)
64 bit	Athlon 64 Pentium 4 Itanium	Server software 64 bits versions of Windows, Linux etc.

Fig. 5. Today's PC's use mostly 32-bit architecture.

The pre-history of computers

Our PC's have "spiritual roots" going back 350 years. Mathematicians and philosophers like Pascal, Leibnitz, Babbage and Boole laid the foundations with their theoretical work.

The Frenchman, Blaise Pascal, lived from 1623-1662, and was a mathematical genius from a very young age.

As an 18-year-old, he constructed a calculating machine, and his mathematical theories have had enormous significance to all later scientific research.

The Englishman, George Boole (1815-1864), was also a natural talent. He grew up in very humble surroundings, and was largely self-taught.

When he was 20 years old, Boole founded a mathematics school and then began to develop the symbolic logic which is currently the cornerstone of every program.

Another Englishman, Charles Babbage, began developing various mechanical calculating machines in 1823, which are today considered to be the theoretical forerunners of the computer. Babbage's "analytical machine" could perform data calculations using punched cards. The machine was never fully realised; the plan was to power it using steam.

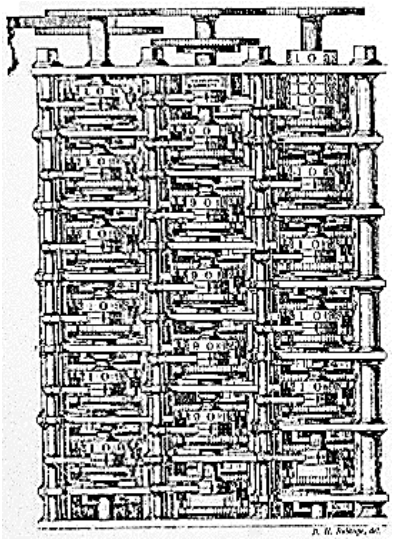


Fig. 6. A construction drawing for one of Babbage's calculating machines, which consisted of several tons of brass machinery.

Fig. 7. Charles Babbage (1791-1871) and his staff constructed various programs (software) for his calculating machine. Babbage is therefore called the "father of the computer" today.

However, it was only in the 20th century that electronics advanced sufficiently to make practical exploitation of these theories interesting.

The Bulgarian John Vincent Atanasoff (1903-1995) is the inventor of the *electronic digital computer*.

Atanasoff was a genius. At the age of nine, he studied algebra with the help of his mother Iva Lucena Purdy, a mathematics schoolteacher.

In the 1930'ies Atanasoff was a professor of mathematics and physics at Iowa State University in the USA. Here he used the existing tools like the Monroe calculater and IBM tabulator for his calculations, but he found these machines too slow and inaccurate. For years he worked on the idea that there should better machines for calculation. His thought was to produce a *digital machine*, since Atanasoff had concluded that mathematical devices fell into two classes, analog and digital. The term digital was not invented, so he called this class of devices "computing machines proper"

In the winter of 1939 Atanasoff was very frustrated from his lack of progress. After a long car ride (Atanasoff was fond of fast cars) he found himself drinking whisky in a bar (he was fond of scotch as well). Suddenly he had the solution. A machine built on four principles. It should work on base-two (binary) numbers instead of base-10 and use condensers for memory. Atanasoff teamed up with a brilliant young electrician Clifford Berry and later the 700 pounds machine called *Atanasoff-Berry Computer* was developed. This was the first digital computer.

Another pioneer was the German Konrad Zuse (1910-1995). He was only 18 when he constructed his own mechanical binary computer called Z1.

During the Second World War Zuse's computer Z3 was used in the German aircraft industry. It was the first computer in the world to be programmed with software. It is interesting, that Zuse's computers were developed entirely independent of other contemporary scientists work.

Figur 8. Konrad Zuse. One of the first scientists to produce working computers.

During the war, the Germans also used an advanced code machine (Fig. 8), which the English expended a great deal of effort on "hacking". They were successful, and this contributed to laying the foundation for the later development of computing.

An interesting piece of trivia: In 1947, the American computer expert, Howard Aiken, stated that there was only a need for six computers in the entire USA. History proved him wrong.

Fig. 9. The German ”ENIGMA” code machine.

- Next chapter.
- Previous chapter.

- Next chapter.
- Previous chapter.

Chapter 2. The Von Neumann model

The modern microcomputer has roots going back to USA in the 1940's. Of the many researchers, the Hungarian-born mathematician, John von Neumann (1903-57), is worthy of special mention. He developed a very basic model for computers which we are still using today.

Fig. 10. John von Neumann (1903-57). Progenitor of the modern, electronic PC.

Von Neumann divided a computer's hardware into 5 primary groups:

- CPU
- Input
- Output
- Working storage
- Permanent storage

This division provided the actual foundation for the modern PC, as von Neumann was the first person to construct a computer which had working storage (what we today call RAM). And the amazing thing is, his model is still completely applicable today. If we apply the von Neumann model to today's PC, it looks like this:

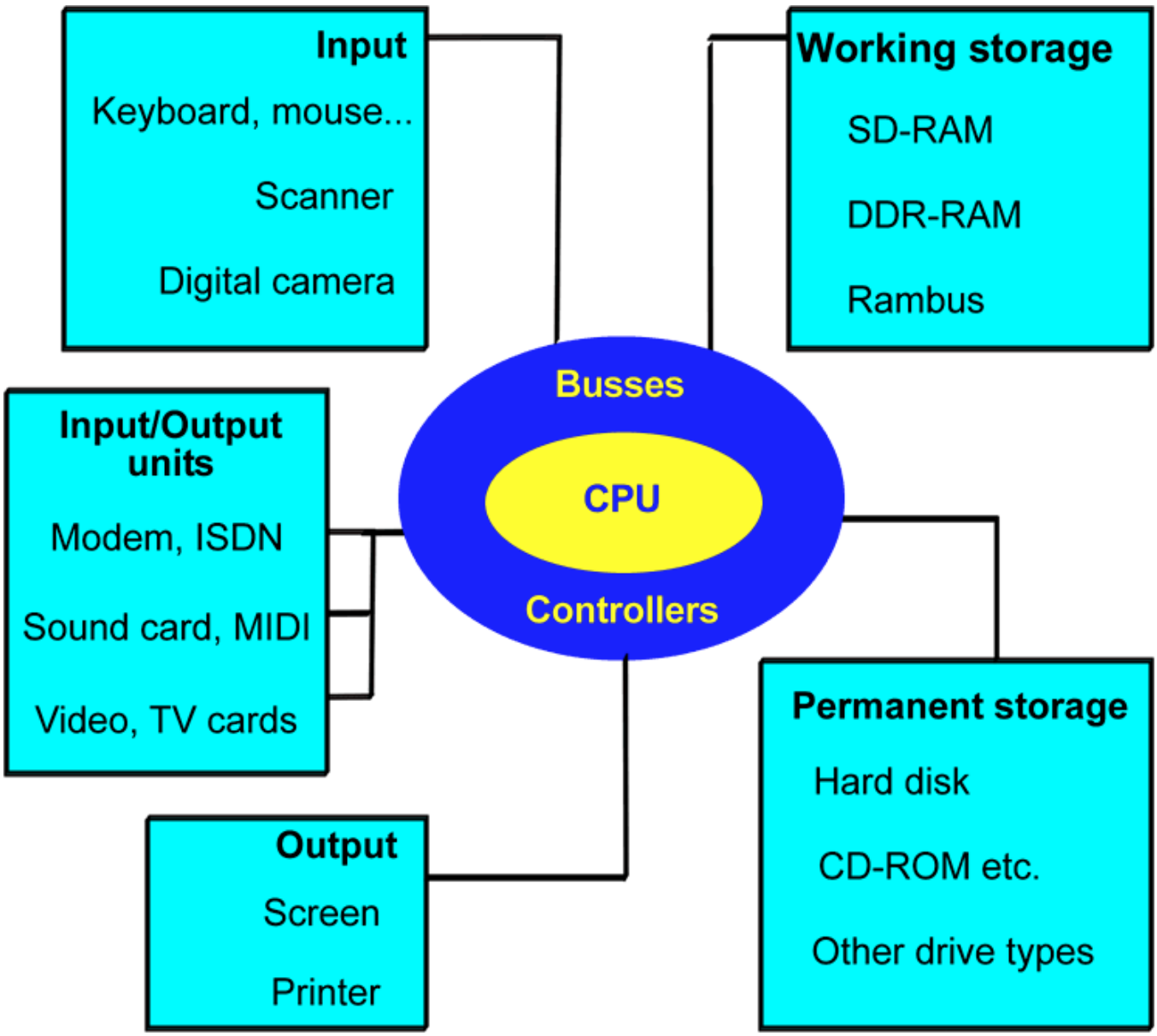


Fig. 11. The Von Neumann model in the year 2004.

Today we talk about multimedia PC's, which are made up of a wealth of interesting components. Note here that modems, sound cards and video cards, etc. all function as both input and output units. But this doesn't happen simultaneously, as the model might lead you to believe. At the basic level, the von Neumann model still applies today. All the components and terms shown in Fig. 11 are important to be aware of. The model generally helps in gaining a good understanding of the PC, so I recommend you study it.

Fig. 12. Cray supercomputer, 1976.

In April 2002 I read that the Japanese had developed the world's fastest computer. It is a huge thing (the size of four tennis courts), which can execute 35.6 billion mathematical operations per second. That's five times as many as the previous record holder, a supercomputer from IBM.

The report from Japan shocked the Americans, who considered themselves to be the leaders in the are of computer technology. While the American super computers are used for the development of new weapons systems, the Japanese one is to be used to simulate climate models.

2. The PC's system components

This chapter is going to introduce a number of the concepts which you have to know in order to understand the PC's architecture. I will start with a short glossary, followed by a brief description of the components which will be the subject of the rest of this guide, and which are shown in Fig. 11.

The necessary concepts

I'm soon going to start throwing words around like: *interface, controller and protocol*. These aren't arbitrary words. In order to understand the transport of data inside the PC we need to agree on various jargon terms. I have explained a handful of them below. See also the glossary in the back of the guide.

The concepts below are quite central. They will be explained in more detail later in the guide, but start by reading these brief explanations.

Concept	**Explanation**
Binary data	Data, be it instructions, user data or something else, which has been translated into sequences of 0's and 1's.
Bus width	The size of the packet of data which is processed (e.g. moved) in each work cycle. This can be 8, 16, 32, 64, 128 or 256 bits.
Band width	The data transfer capacity. This is measured in, for example, kilobits/second (Kbps) or megabytes/second (MBps).

Cache	A temporary storage, a buffer.
Chipset	A collection of one or more controllers. Many of the motherboard's controllers are gathered together into a chipset, which is normally made up of a north bridge and a south bridge.
Controller	A circuit which controls one or more hardware components. The controller is often part of the interface.
Hubs	This expression is often used in relation to chipset design, where the two north and south bridge controllers are called hubs in modern design.
Interface	A system which can transfer data from one component (or subsystem) to another. An interface connects two components (e.g. a hard disk and a motherboard). Interfaces are responsible for the exchange of data between two components. At the physical level they consist of both software and hardware elements.
I/O units	Components like mice, keyboards, serial and parallel ports, screens, network and other cards, along with USB, firewire and SCSI controllers, etc.
Clock frequency	The rate at which data is transferred, which varies quite a lot between the various components of the PC. Usually measured in MHz.
Clock tick (or clock cycle)	A single clock tick is the smallest measure in the working cycle. A working cycle (e.g. the transport of a portion of data) can be executed over a period of about 5 clock ticks (it "costs" 5 clock cycles).
Logic	An expression I use to refer to software built into chips and controllers. E.g. an EIDE controller has its own "logic", and the motherboard's BIOS is "logic".
MHz (Megahertz)	A "speed" which is used to indicate clock frequency. It really means: *million cycles per second*. The more MHZ, the more data operations can be performed per second.
North bridge	A chip on the motherboard which serves as a controller for the data traffic close to the CPU. It interfaces with the CPU through the Front Side Bus (FSB) and with the memory through the memory bus.
Protocols	Electronic traffic rules which regulate the flow of data between two components or systems. Protocols form part of interfaces.
South bridge	A chip on the motherboard which works together with the north bridge. It looks after the data traffic which is remote from the CPU (I/O traffic).

Fig. 13. These central concepts will be used again and again. See also the definitions on page PAGEREF Ordforklaringer2 \h 95.

- Next chapter.
- Previous chapter.

Chapter 3. A data processor

The PC is a digital data processor. In practise this means that all *analogue data* (text, sound, pictures) gets translated into masses of 0's and 1's. These numbers (*binary values*) exist as tiny electrical charges in microscopic circuits, where a transistor can take on two states: *charged* or *not charged*. This is one picture of a *bit*, which you can say is either turned *on* or *off*.

There can be billions of these microscopic bits hidden inside a PC, and they are all managed using electronic circuits (EDP stands for *electronic data processing*). For example, the letter "A" (like all other characters) can be represented by a particular 8-digit bit pattern. For "A", this 8-digit bit pattern is *01000001*.

When you type an "A" on your keyboard, you create the digital data sequence, 01000001. To put it simply, the "A" exists as a pattern in eight transistors, where some are "turned on" (charged) and others are not. Together these 8 transistors make up *one byte*.

The same set of data can be stored in the video card's electronics, in RAM or even as a magnetic pattern on your hard disk:

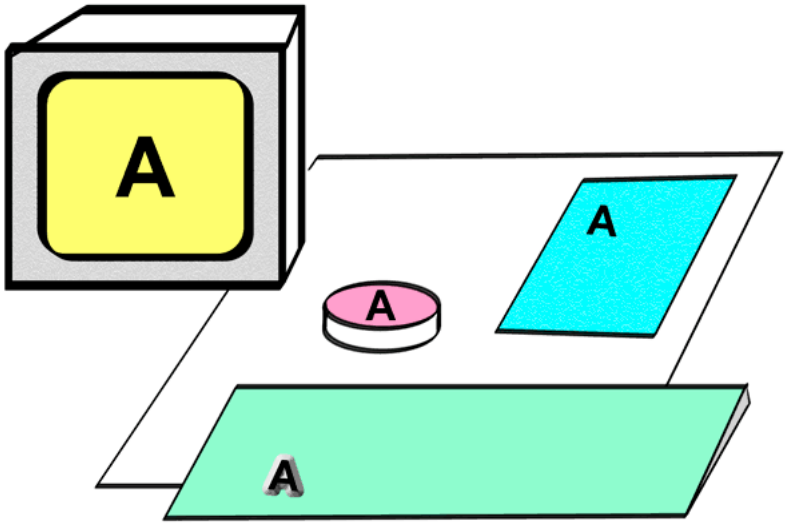


Fig. 14. The same data can be found on the screen, on the hard disk and in RAM.

The set of data can also be transferred to a printer, if you want to print out your text. The printer electronically and mechanically translates the individual bits into analogue letters and numbers which are printed on the paper. In this way, there are billions of bytes constantly circulating in your PC, while ever it is switched on. But how are these 0's and 1's moved around, and which components are responsible?

The physical PC

The PC is made up of a central unit (also called a system unit) and some external devices. The central unit is a box (a cabinet), which contains most of the computer's electronics (the *internal* devices). The *external* devices are connected to the central unit (shown below) using cables.

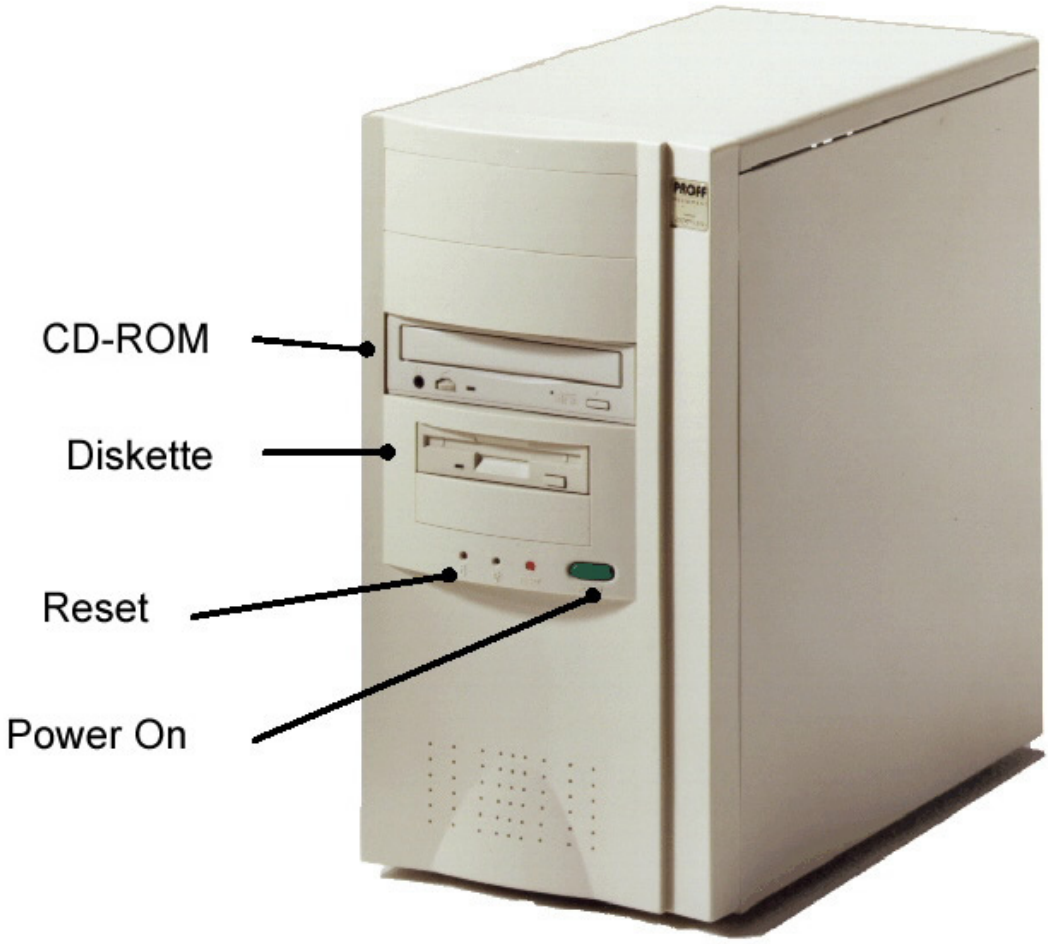


Fig. 15. The central unit contains the majority of a PC's electronics.

The cabinet shown in Fig. 15 is a *minitower*. In this cabinet, the motherboard is mounted vertically down one side. You can buy a taller cabinet of the same type. It's called a *tower*. If the cabinet is designed to be placed on a desk (under the monitor), it is called a *desktop* cabinet.

Fig. 16. A desktop cabinet.

Fig.17 shows a list of most of the components of the PC. Some of them are *internal*, i.e., they are inside the cabinet. Other components are *external*, they are located *outside* the cabinet.

Read through the list and think about what the words refer to. Do you know all these devices?

Internal devices		External devices
Motherboard	CPU, RAM, cache, ROM circuits containing the BIOS and startup programs. Chipsets (controllers). Ports, busses and slots. EIDE interface, USB, AGP, etc.	Keyboard Mouse Joystick Screen Printer Scanner Speakers External drives Tape drive MIDI units Modem Digital camera
Drives	Hard disk(s), diskette drive, CD-ROM, DVD, etc.	
Plug-in cards	Graphics card (video adapter), network card, SCSI controller. Sound card, video and TV card. Modem and ISDN card.	

Fig. 17. The PC's components can be divided into internal and external groups.

Speed – the more we get, the more we want

The PC processes data. It performs *calculations* and *moves* data between the various components. It all happens at our command, and we want it to happen fast.

It is interesting to note that current technological development is basically focusing exclusively on achieving *faster* data processing. The entire PC revolution over the last 20 years is actually just a sequence of ever increasing speed records in the area of data transfer. And there doesn't seem to be any upper limit to how much data transfer speed we need.

This continual speed optimisation is not just occurring in one place in the PC; it's happening everywhere that data is moved.

- The transfer from RAM to CPU – it has to be faster.
- The transfer between hard disk and motherboard – it has to be faster.
- Data to the screen – it has to be faster.
- Etc.

The PC can be viewed as a series of more or less independent subsystems, which can each be developed to permit greater capacity and higher speed. We constantly need new *standards*, because of the new, faster, interfaces, busses, protocols (which we all work out together), delivering better performance.

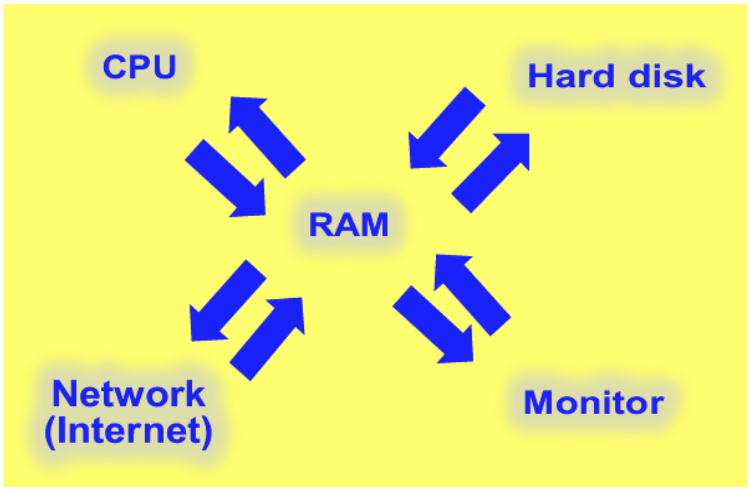


Fig. 18. Data transfer between all the components of the PC has to be fast.

Interfaces hold it all together

The PC is the sum of all these subsystems. At each boundary between one subsystem and another, we find an *interface*. That is, an electrical system which connects the two subsystems together and enables them to exchange data.

Fig. 19. The hardware components are connected to each other via interfaces.

The concept of an interface is a little abstract, as it most accurately refers to a *standard* (a set of rules for the exchange of data). In practise, an interface can consist of, for example, two *controllers* (one at each end of the connection), a cable, and some software (protocols, etc.) contained in the controllers.

The controllers are small electronic circuits which control the movement of data to and from the device.

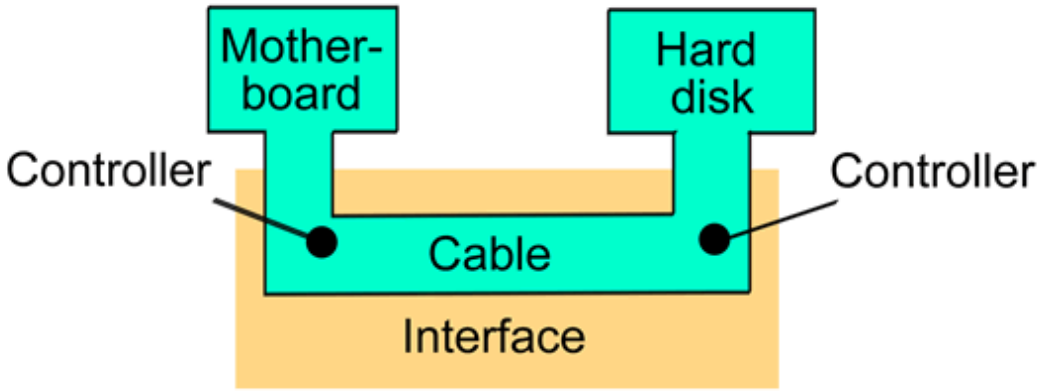


Fig. 20. An interface connects two hardware devices. An interface can consist of controllers with built-in software, cables, etc.

There are many interfaces in the PC, because there are many subsystems which have to be connected. Each interface is normally tailor-made for the job, and tuned to achieve maximum bandwidth (data transfer capacity) between the two components.

An example of an interface

Later in the guide I want to explore the EIDE interface in more detail, but I will use it here as a specific example of an interface. Keep your attention focused on the concept of an interface – you may not understand all the details, that doesn't matter here.

If we want to connect a hard disk to a motherboard, this is achieved using an EIDE interface. If we look more closely at this interface, it can be divided into a series of subcomponents. The interface consists of both hardware and logic: the most important being the two EIDE controllers. One is integrated into the hard disk's electronics, and the other is integrated into the motherboard, where it forms part of the chipset's south bridge.

Fig. 21. Underneath the hard disk you can see a small printed circuit board. This incorporates the controller functions which work together with the corresponding controller in the PC's motherboard.

The advantage of this system is that the hard disk can be connected directly to the motherboard with a cable. But the cable still runs from one controller to the other.

The two controllers work according to a common *standard*, which is the ATA standard. This standard includes a set of *protocols* which are continually being developed in new versions. Let's say our specific hard disk can use the *ATA/100* protocol. That means the controller on the motherboard has to also be compatible with ATA/100, and the cable as well. When all that is in place, we have a working ATA interface.

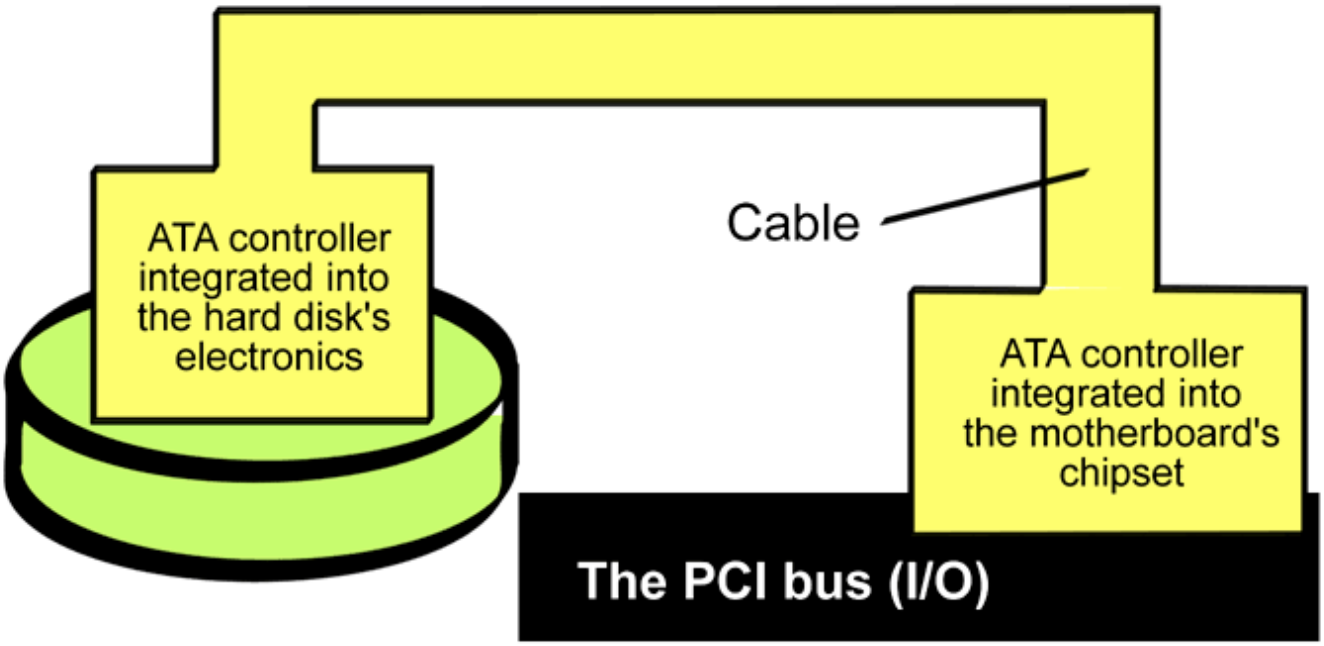


Fig. 22. A specific example of an interface.

- Next chapter.
- Previous chapter.

- Next chapter.
- Previous chapter.

Chapter 4. Intro to the motherboard

Now let's dive into the pc box. The whole computer is built up around a *motherboard*, and it is the most important component in the PC.

In this chapter I will introduce the motherboard and it's components.

- Construction of the motherboard.
- The CPU.
- The busses.
- Chipsets (controllers).

I will work through the individual components in more detail later in the guide. This chapter will describe the architecture in "broader" brush strokes.

Data exchange in the motherboard

The motherboard is a large printed circuit board, which has lots of chips, connectors and other electronics mounted on it. Computer nerds simply call it a *board*.

Inside the PC, data is constantly being exchanged between or via the various *devices* shown in Fig. 17. Most of the data exchange takes place on the motherboard itself, where all the components are connected to each other:

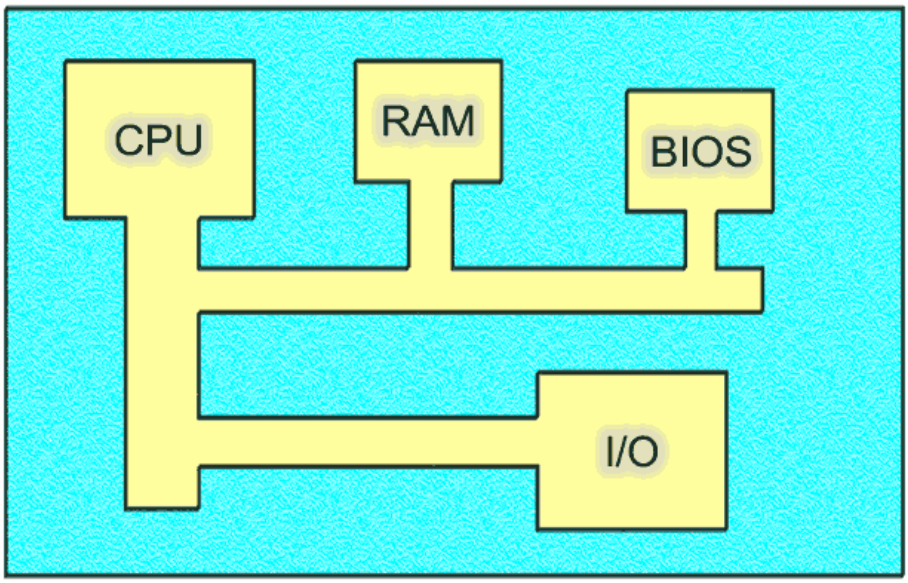


Fig. 23. Data exchange on the motherboard.

In relation to the PC's *external* devices, the motherboard functions like a central railway station.

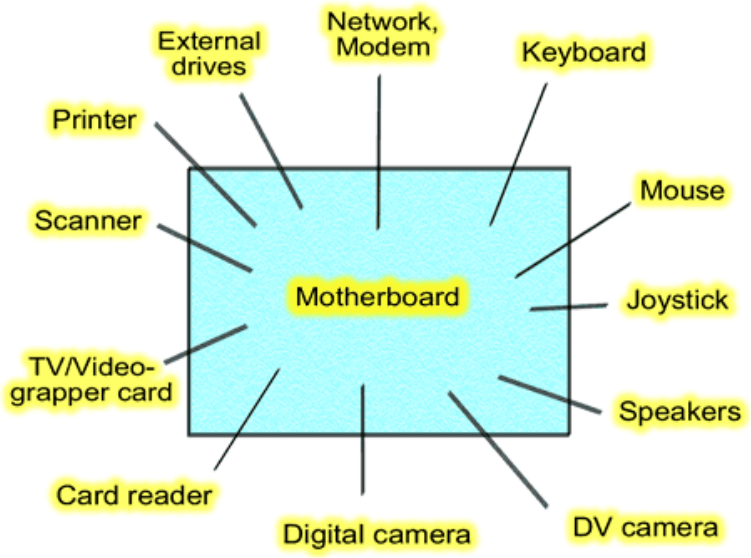


Fig. 24. The motherboard is the hub of all data exchange.

All traffic originates from or ends up in the motherboard; which is appropriately called the most important component of the PC. I will show you pictures of the individual components of the motherboard later, but this is what it looks like as a total unit:

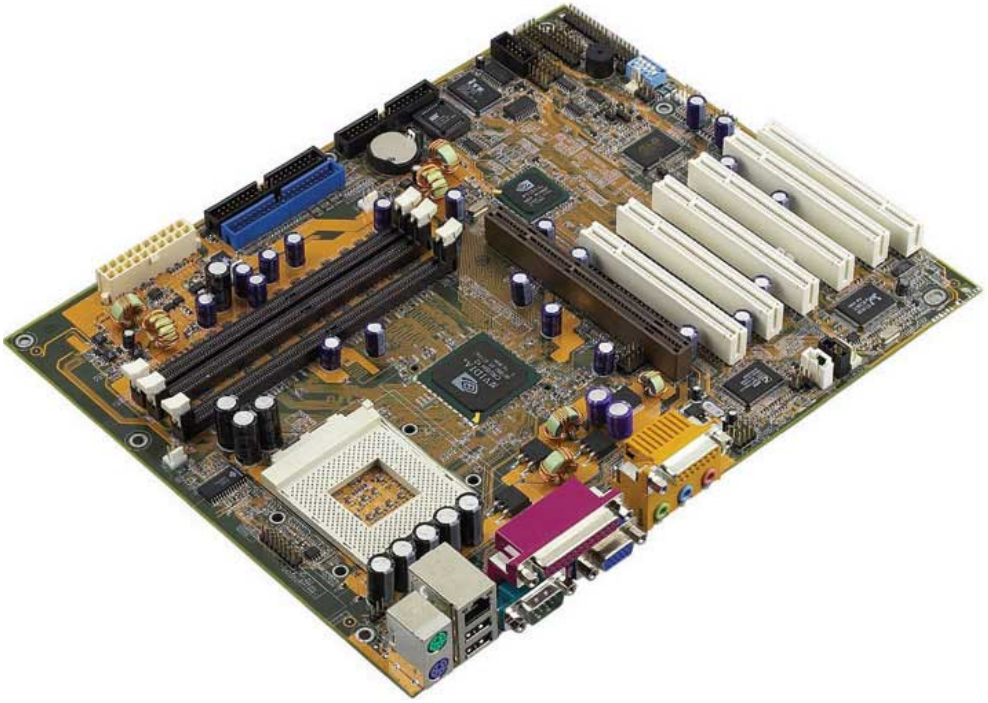


Fig. 25. A motherboard is a board covered with electronics.

Find your motherboard

If you are in position to look at a motherboard, I would recommend you do so. It is a very good exercise to try to identify the various components on a motherboard.

The motherboard is really just a big plastic sheet which is full of electrical conductors. The conductors (also called tracks) run across and down, and in several layers, in order to connect all the individual components, and transfer data between them.

The motherboard is mounted in the PC box using small plastic brackets and screws. The cabinet and the motherboard are made to suit each other, so there are holes in the metal for the connectors mounted on the board. Finally, the motherboard has to be connected to the PC's power supply installed in the cabinet. This is done using a standard connector:

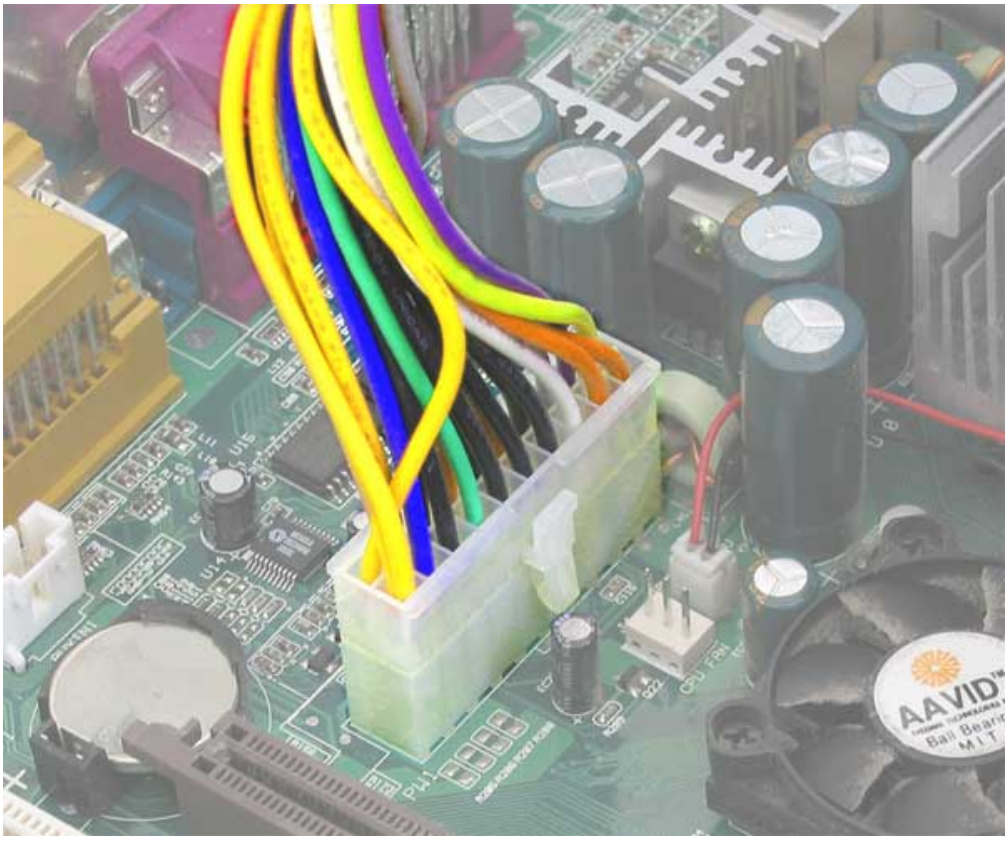


Fig. 26. The power supply is connected to the motherboard via a multicoloured cable and a large white plastic connector.

Now we'll look at the various types of components on the motherboard.

Chips

The active devices on the motherboard are gathered together in *chips*. These are tiny electronic circuits which are crammed with transistors. The chips have various functions. For example, there are:

- ROM chips, which store the BIOS and other *programs*.
- CMOS storage, which contains user-defined data used by the setup program.
- The chipset, which normally consists of two, so-called *controllers*, which incorporate a number of very essential functions.

You'll learn a lot about these chips and their functions later in the guide.

Sockets

You will also find *sockets* on the motherboard. These are holders, which have been soldered to the motherboard. The sockets are built to exactly match a card or a chip.

This is how a number of components are directly connected to the motherboard. For example, there are sockets (*slots*) to mount:

- The CPU and working storage (the RAM modules).
- Expansion cards, also called adapters (PCI, AGP and AMR slots, etc.).

The idea of a socket is, that you can install a component directly on the motherboard without needing special tools. The component has to be pushed carefully and firmly into the socket, and will then hopefully stay there.

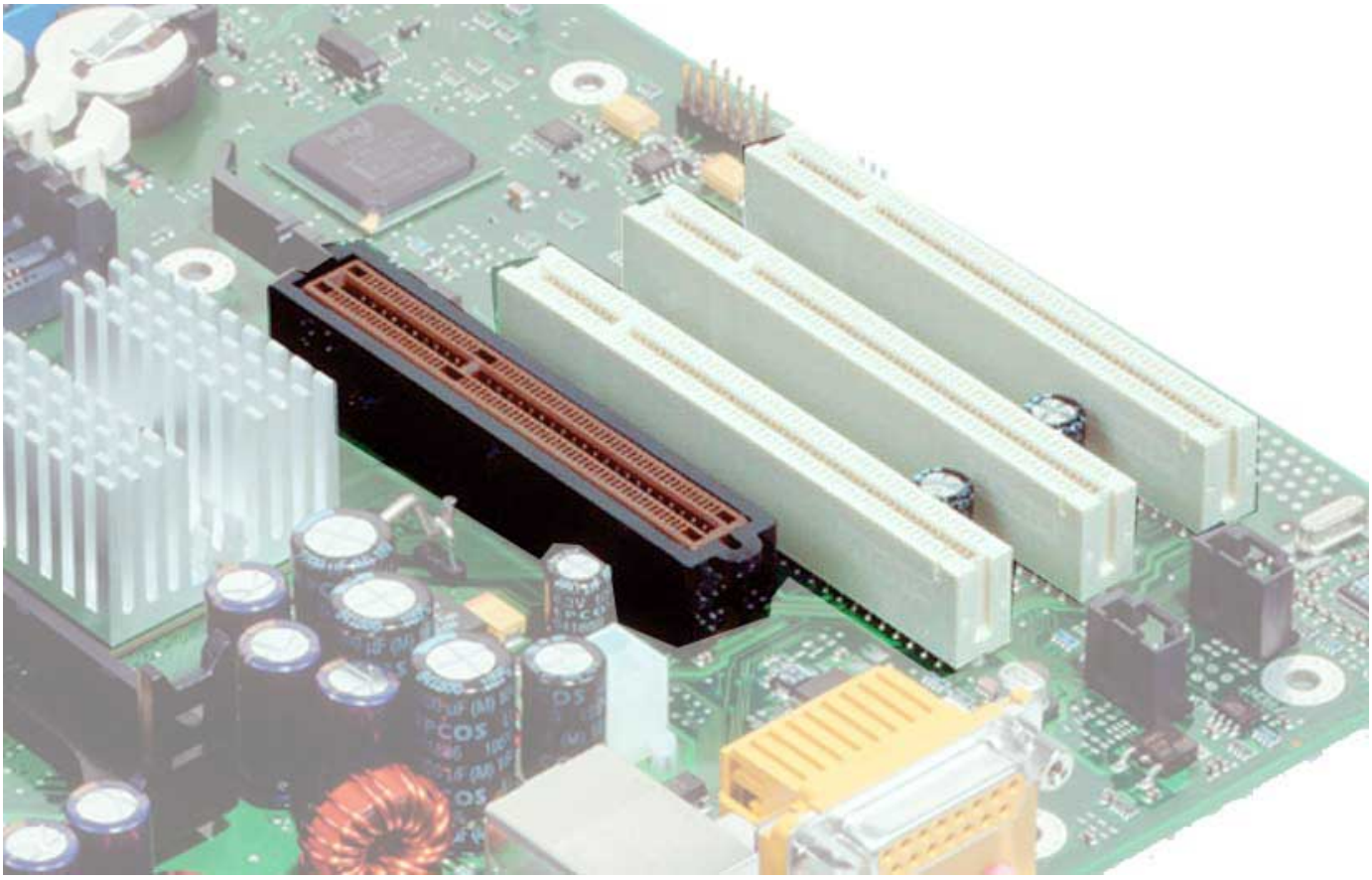


Fig. 27. Here you can see three (white) PCI sockets, in which plug-in cards can be installed.

Plugs, connectors and ports...

The motherboard also contains a number of inputs and outputs, to which various equipment can be connected. Most ports (also called I/O ports) can be seen where they end in a connector at the back of the PC. These are:

- Ports for the keyboard and mouse.
- Serial ports, the parallel port, and USB ports.
- Sockets for speakers/microphone etc.

Often, the various connectors are soldered onto the motherboard, so that the external components, like the keyboard, mouse, printer, speakers, etc., can be connected directly to the motherboard.

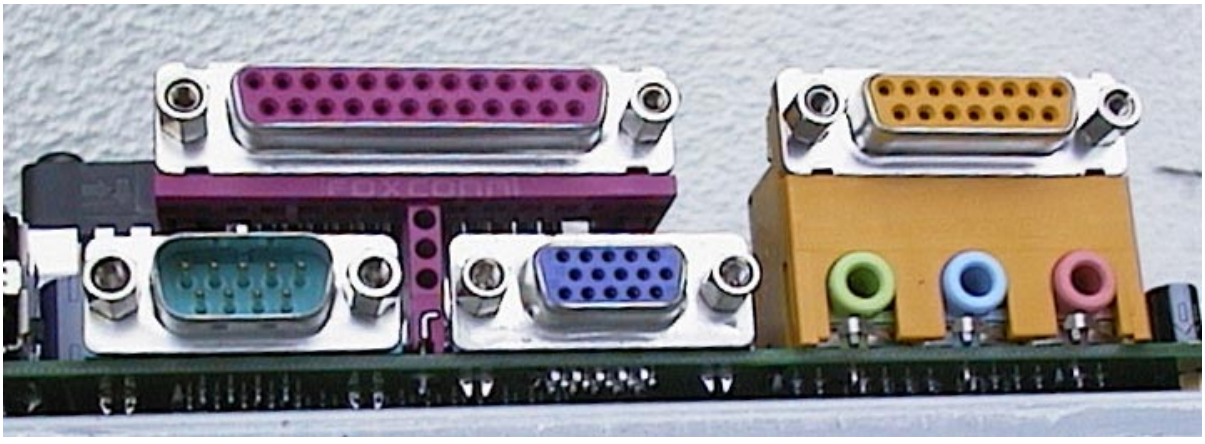


Fig. 28. Connectors mounted directly on a motherboard.

In addition to these sockets, connectors and ports, the motherboard contains a number of other contacts. These include:

- The big *connector* which supplies the motherboard with power from the power supply (see Fig. 26.
- Other connectors for the diskette drive, hard disk, CD-ROM drive, etc.
- So-called *jumpers*, which are used on some motherboards to configure voltage and various operating speeds, etc.
- A number of pins used to connect the reset button, LED for hard disk activity, built-in speaker, etc.

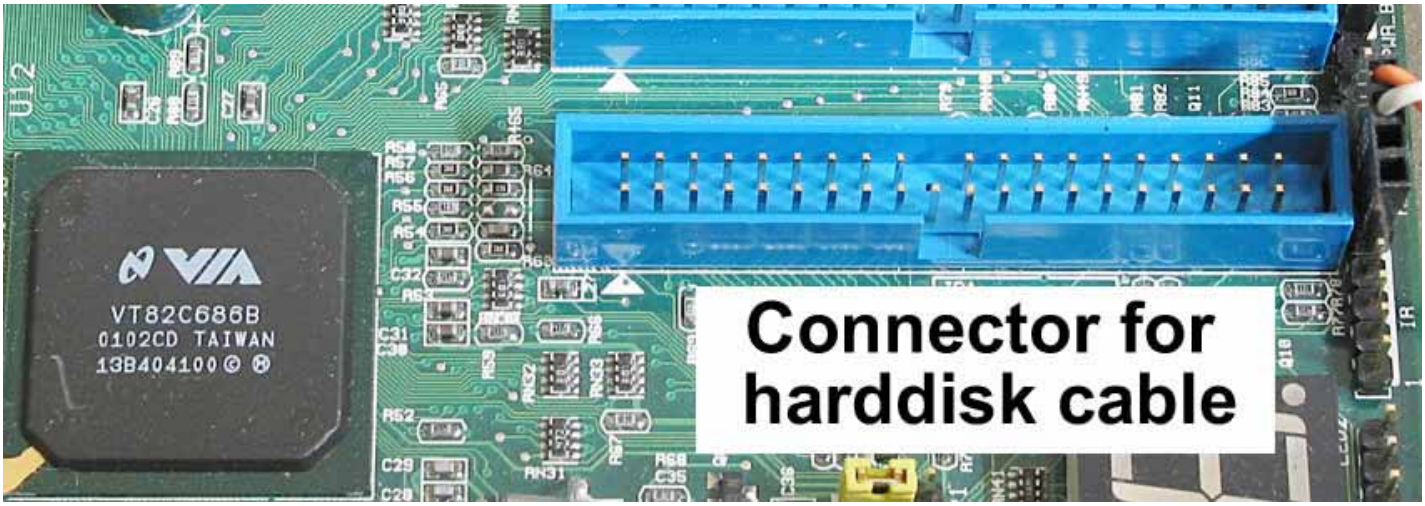


Fig. 29. A connector can be an array of pins like this, which suits a special cable.

Take a look at Fig. 30 and Fig. 31, which show connectors and jumpers from two different motherboards.

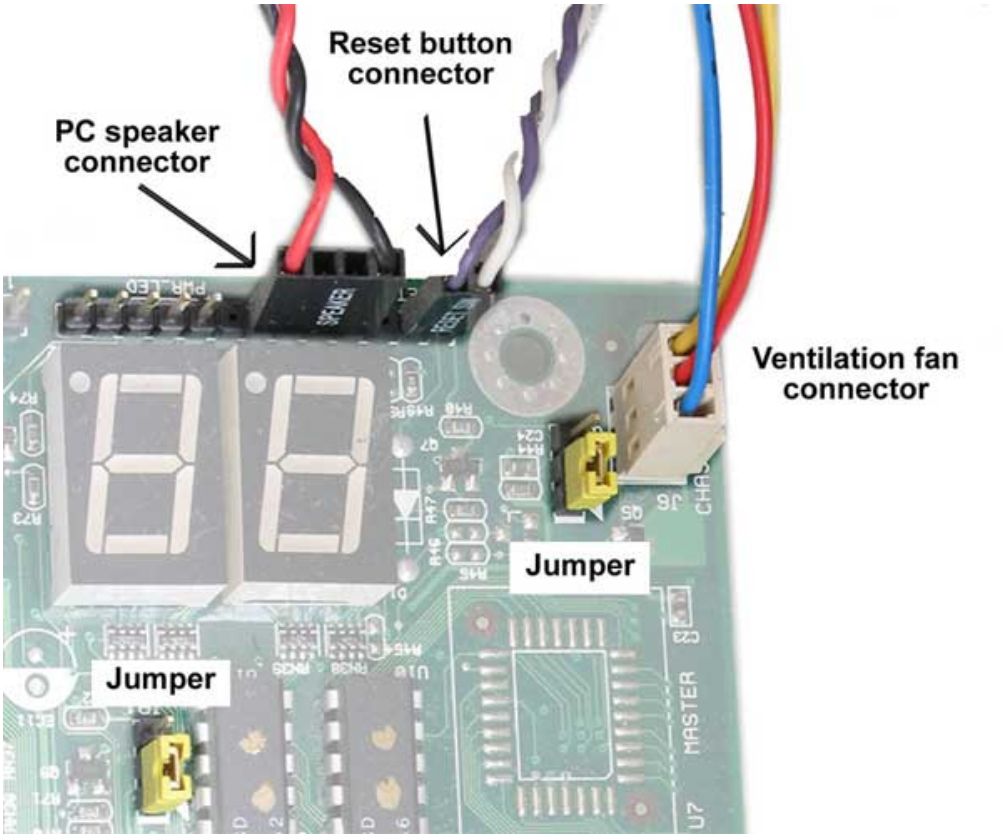


Fig. 30. The tiny connectors and jumpers that are hidden on any motherboard.

The ROM BIOS chip (Award brand), inFig. 31, contains a small collection of programs (software) which are permanently stored on the motherboard, and which are used, for example, when the PC starts up:

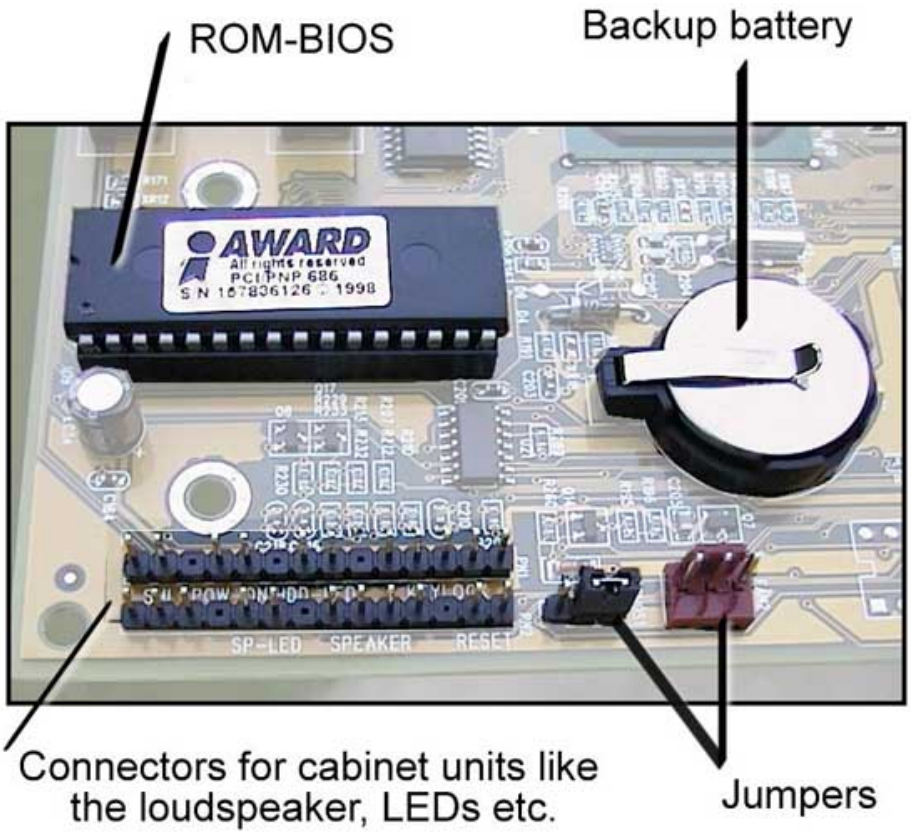


Fig. 31. At the bottom left, you can see the two rows of pins which connect, for example, to the little speaker inside the cabinet. On the bottom right you can see two “jumpers”.

The round thing in Fig. 31 is the motherboard battery, which maintains the clock function and any settings saved in the CMOS storage.

In a later chapter I will describe the motherboard seen through the eyes of a PC builder. But first we shall take a look at the motherboard’s architecture and the central components found on it.

- Next chapter.
- Previous chapter.

- Next chapter.
- Previous chapter.

Chapter 5. It all starts with the CPU

There are two very fundamental components to study on the motherboard. The CPU and the busses. The CPU does all the data processing, and the busses handle all data transfer.

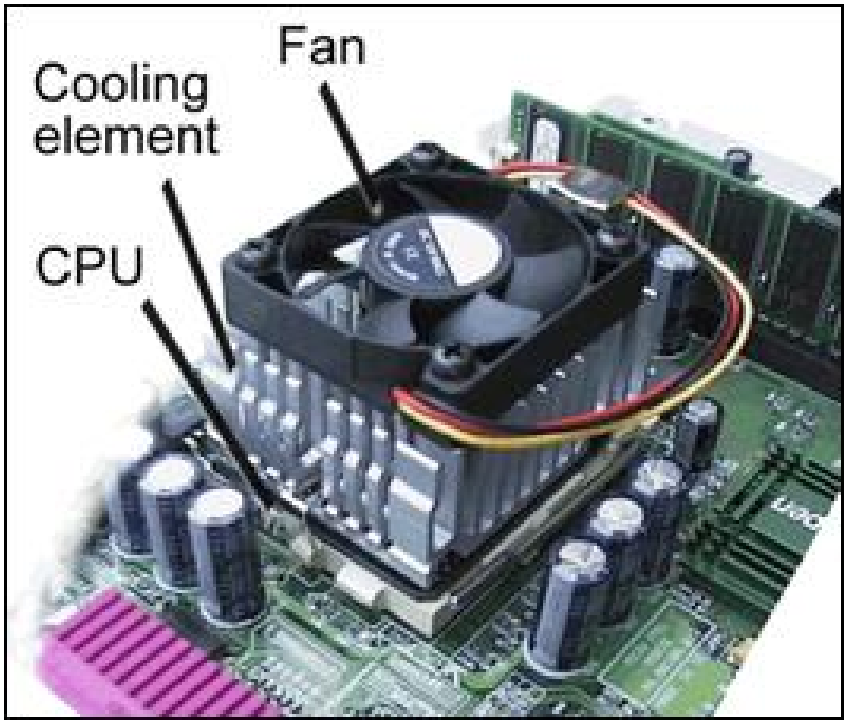


Fig. 32. The CPU is mounted on the motherboard, hidden under the cooling fan and heat sink.

What is a CPU?

CPU stands for Central Processing Unit. There can be several processors in a computer, but one of them is the central one – the CPU.

The reason the CPU is called a processor is because it can work with data. And it has two important jobs:

- It can do *calculations*.
- It can *move* data.

The CPU is very fast at doing both jobs. The faster the CPU can do calculations and move data, the faster we say the PC is. What follows is a short description of how to achieve faster data processing. Read it, and see if you understand all the concepts. There are three ways to improve a PC's performance:

- Higher *clock frequencies* (which means more clock ticks per second).

- Greater *bus width*.

- Optimising the core of the processor and other components so that the maximum amount of work is done for each *clock tick*.

All this can lead to better *bandwidth*, which is required throughout the PC. The entire development process is focused around the motherboard, and especially the CPU. But all of the electronics has to be able to keep up with the high pace, and that is what makes the motherboard so fascinating.

The CPU is physically quite small. At its core is an electronic circuit (called a *die*), which is no bigger than your little fingernail.

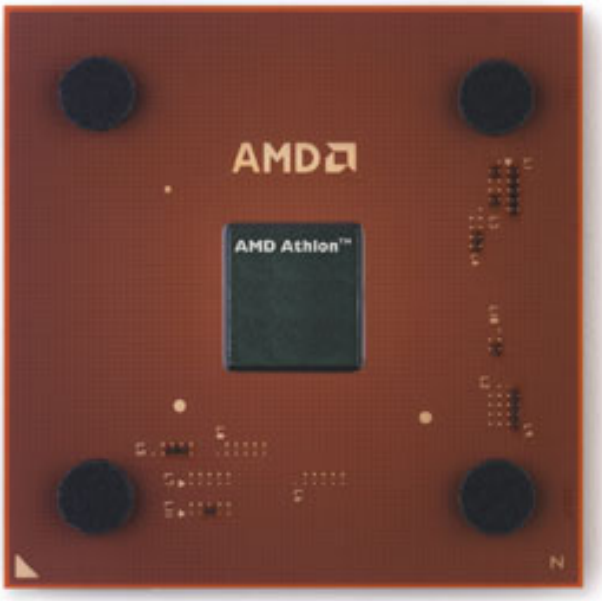


Fig. 33. The CPU circuit (the ”die”) can be seen in the middle of the chip (An AthlonXP shown close to actual size).

Despite its small size, the CPU is full of transistors. The *die* in a Pentium 4 CPU contains 125 million transistors, all squashed together into a very tight space. It is about 1 cm x 1 cm in size:

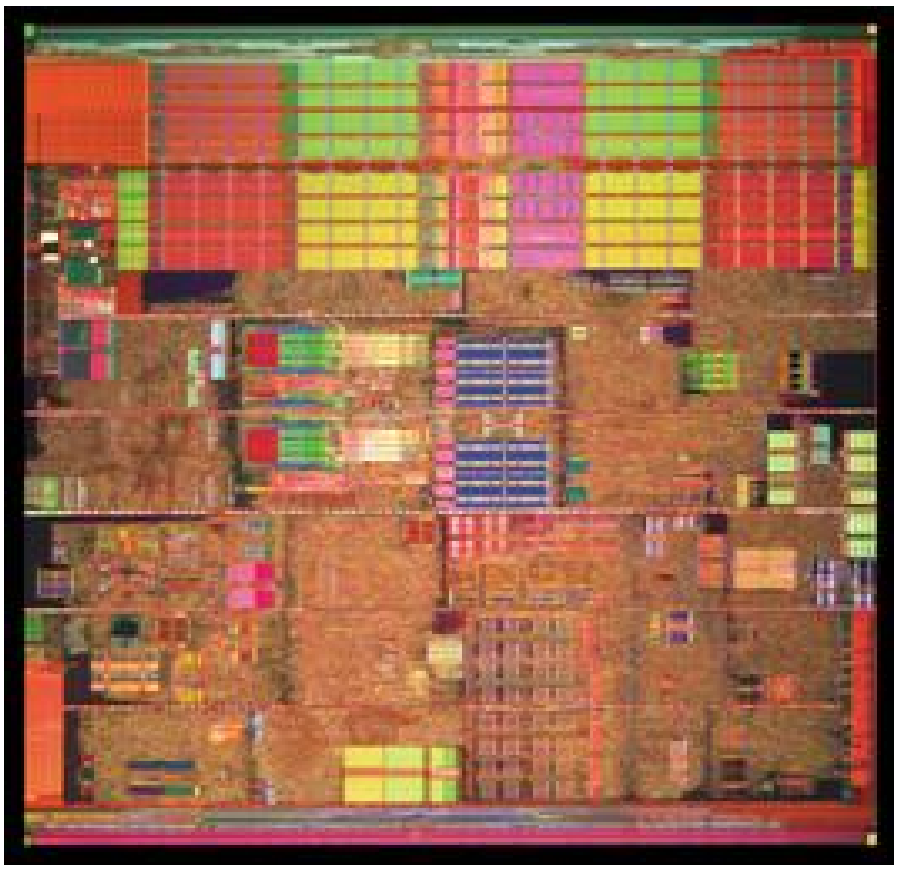
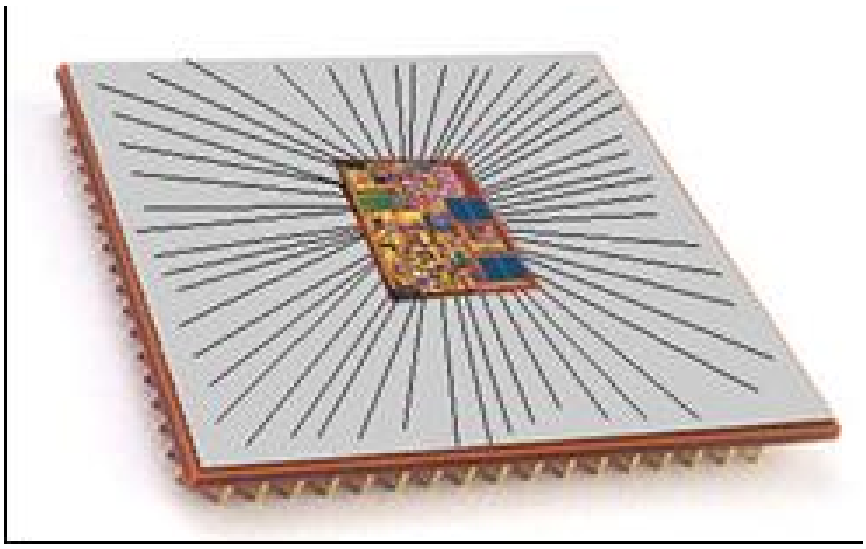


Fig. 34. Close up of a CPU circuit (die).

The electronic circuit is encapsulated in a much bigger plastic square. This is in order to make room for all the electrical contacts which are used to connect the CPU to the motherboard.



The individual contacts are called *pins*, and a CPU can have 478 of them (as does the Pentium 4). The large number of pins means that the socket has to be relatively large.

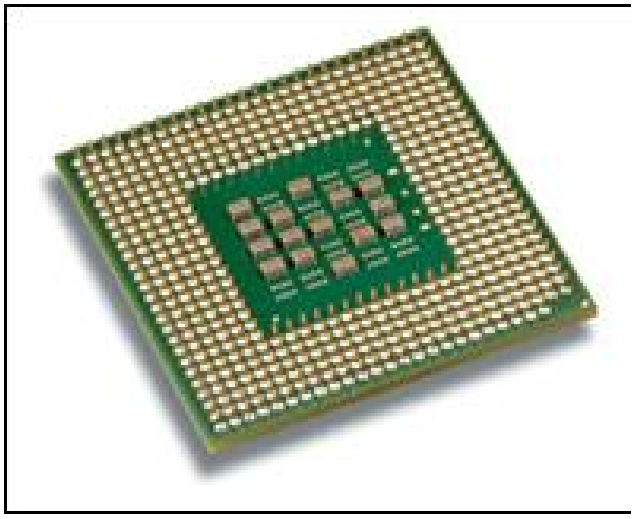


Fig. 35. The underside of a (Pentium 4) CPU, showing the many pins.

Which CPU?

The companies Intel and AMD make most CPU's. Intel laid the foundations for the development of CPU's for PCs with their more than 20 year old 8086 and 8088 processors.

CPU's are developed in series, or *generations*. Each series is known by its *name*. The last four generations of Intel processors, for example, have been the Pentium, Pentium II, Pentium III and Pentium 4. Running alongside these is the Celeron series, which are cheaper versions, typically with reduced L2 cache and a slower front side bus:

Fig. 36. A Celeron processor supplied in a box from Intel, with heat sink and fan.

Within each generation there are many variants with different *clock frequencies*. For example, when the Pentium 4 was released in the year 2000, it was as a 1400 MHz version. The original model was later followed up by versions with 1800, 2000, etc. MHz, up to 2400 MHz (the clock frequencies came in intervals of 100 MHz). In the year 2002, a new model came out for which the clock frequencies started at 2266, 2400 and 2533 MHz, and increased in intervals of 133 MHz. A year later the clock frequencies was raised to intervals of 200 MHz with the Pentium 4 chips running from 2600 to 3600 MHz. And so it continues.

The company, AMD, produces similar processors in the Sempron and Athlon 64 series, which also come with different clock frequencies.

Figur 37. The Pentium 4 socket 478 on a motherboard.

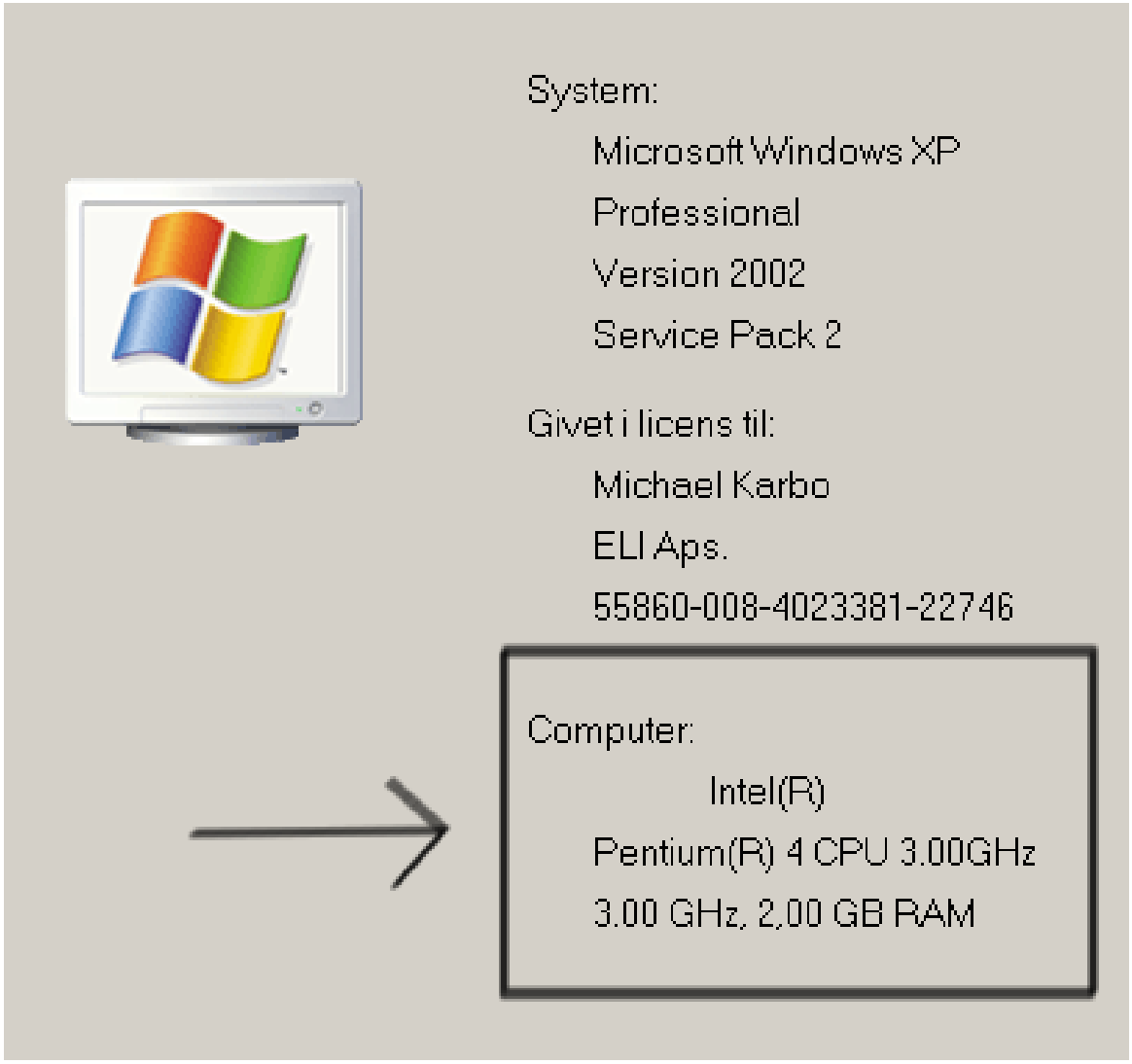
Find your CPU

If you are not sure which CPU your PC uses, you can investigate this in several ways. You could check your purchase receipt. The name of the CPU should be specified there.

You could look inside your PC and locate the CPU. But it is quite difficult to get to see the model name, because there is a fan mounted on the actual chip. The fan is often glued directly onto the processor, so that it is not easy to remove it.

Fig. 38. A CPU is shown here without a cooling fan. It is mounted in a small socket which it clicks into without needing any tools.

In Windows, you can select the System Properties dialog box, where you can see the processor name and clock frequency:



You can also watch carefully when your PC starts up. Your CPU name and clock frequency is shown as one of the first things displayed on the screen. You can press the P key to pause the startup process. Below you can see a picture of the startup screen for PC. This PC has an Intel Pentium 4, with a clock frequency (work rate) of 2553 MHz:

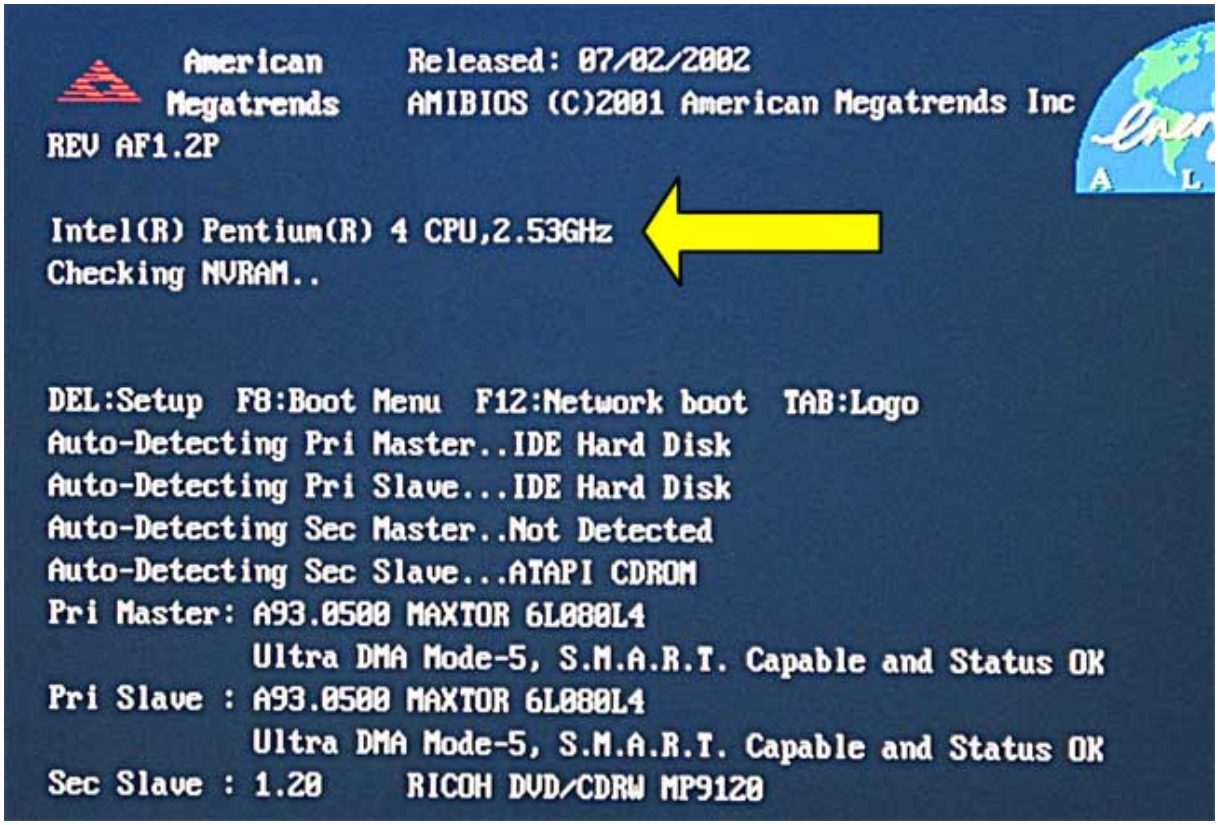
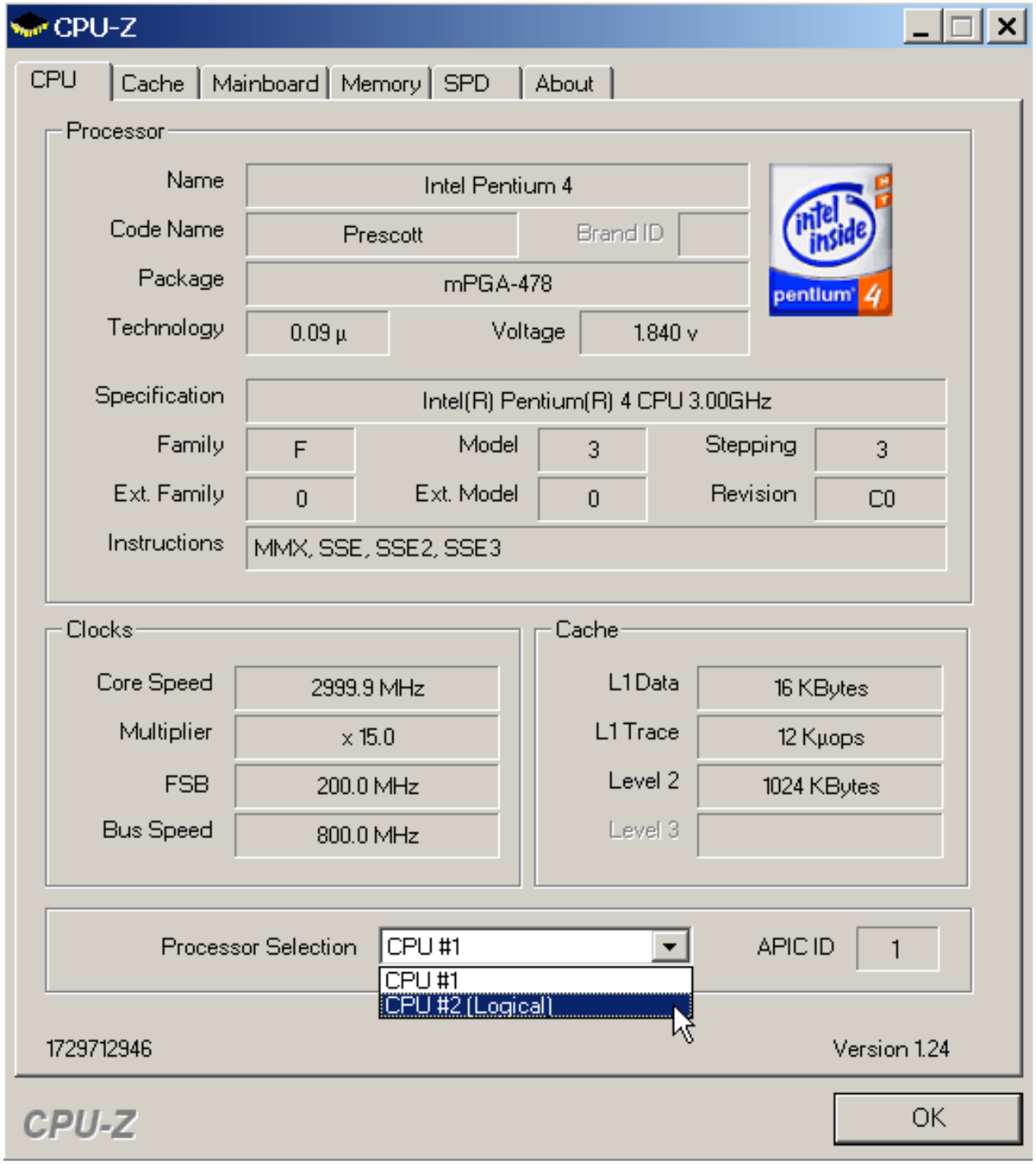


Fig. 39. If you are not sure which CPU your PC uses, you can see it on the screen, shortly after you switch on your PC.

CPU testing programs

Finally, let me just mention some small utility programs which you can download from the Internet (e. g. search for “WCPUID” or “CPU-Z” on www.google.com, and you’ll find it). The programs WCPUID and CPU-Z, reveals lots of information about your CPU, chipset, etc. They are used by motherboard nerds.



Figur 40. Here CPU-Z reports that the Pentium 4 processor is a ”Prescott” model. Due to Hyper Threading, the processor virtually holds two cores.

- Next chapter.
- Previous chapter.

- Next chapter.
- Previous chapter.

Chapter 6. The CPU and the motherboard

The heart and soul of the PC's data processing is the CPU. But the processor is not alone in the world, it communicates with the rest of the motherboard. There will be many new terms introduced in the following sections, so remember that you can find definitions for all the abbreviations in the back of the guide.

Busses do the transfers

Data packets (of 8, 16, 32, 64 or more bits at a time) are constantly being moved back and forth between the CPU and all the other components (RAM, hard disk, etc.). These transfers are all done using *busses*.

The motherboard is designed around some vary powerful data *channels* (or pathways, as they are also called). It is these busses which connect all the components to each other.

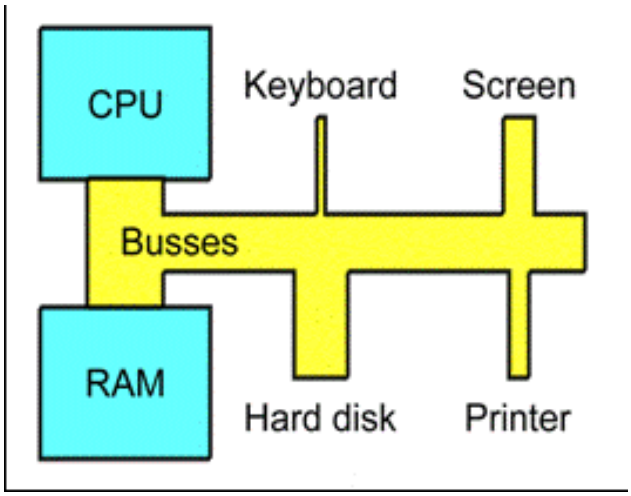


Fig. 41. The busses are the data channels which connect the PC's components together. Some are designed for small transfers, others for large ones.

Busses with varying capacities

There is not just one bus on a motherboard; there are several. But they are all connected, so that data can run from one to another, and hence reach the farthest corners of the motherboard.

We can say that a bus system is subdivided into several branches. Some of the PC components work with enormous amounts of data, while others manage with much less. For example, the keyboard only sends very few bytes per second, whereas the working storage (RAM) can send and receive several gigabytes per second. So you can't attach RAM and the keyboard to the same bus.

Two busses with different capacities (bandwidths) can be connected if we place a controller between them. Such a controller is often called a *bridge*, since it functions as a bridge between the two different

traffic systems.

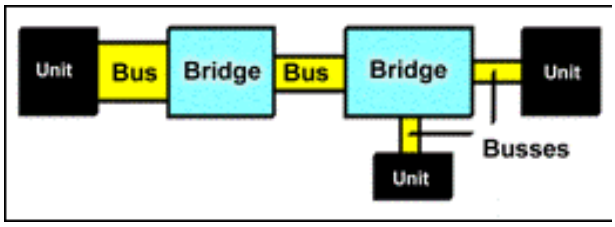


Fig. 42. Bridges connect the various busses together.

The entire bus system starts close to the CPU, where the load (traffic) is greatest. From here, the busses work outwards towards the other components. Closest to the CPU we find the working storage. RAM is the component which has the very greatest data traffic, and is therefore connected directly to the CPU by a particularly powerful bus. It is called the *front side bus* (FSB) or (in older systems) *the system bus*.

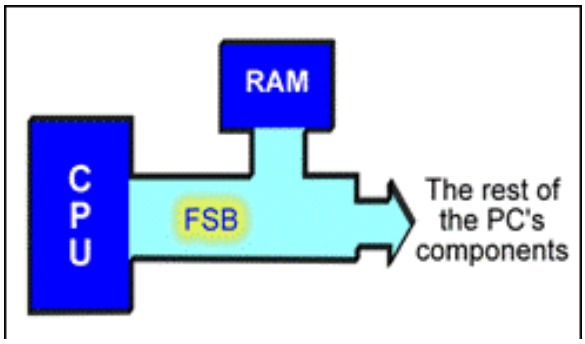


Fig. 43. The PC's most important bus looks after the "heavy" traffic between the CPU and RAM.

The busses connecting the motherboard to the PC's peripheral devices are called *I/O busses*. They are managed by the controllers.

The chip set

The motherboard's busses are regulated by a number of controllers. These are small circuits which have been designed to look after a particular job, like moving data to and from EIDE devices (hard disks, etc.).

A number of controllers are needed on a motherboard, as there are many different types of hardware devices which all need to be able to communicate with each other. Most of these controller functions are grouped together into a couple of large chips, which together comprise the *chip set*.

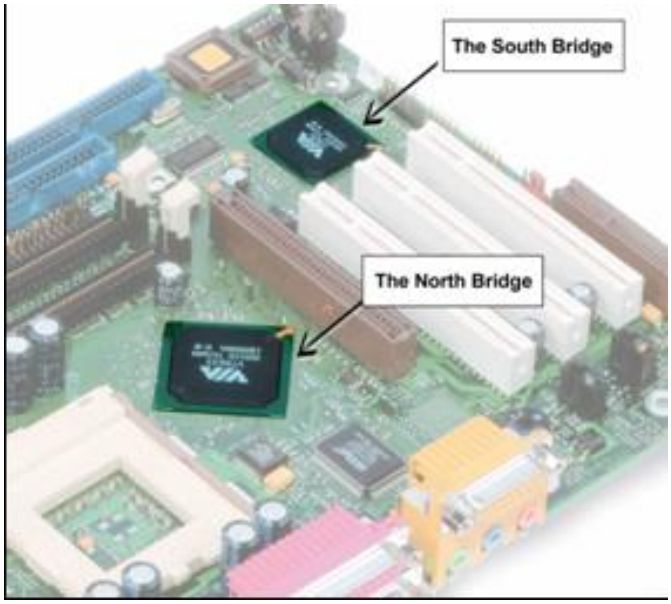


Fig. 44. The two chips which make up the chipset, and which connect the motherboard's busses.

The most widespread chipset architecture consists of *two* chips, usually called the *north* and *south bridges*. This division applies to the most popular chipsets from VIA and Intel. The north bridge and south bridge are connected by a powerful bus, which sometimes is called a *link channel*:

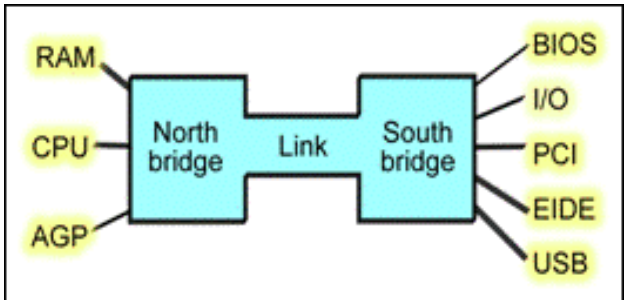


Fig. 45. The north bridge and south bridge share the work of managing the data traffic on the motherboard.

The north bridge

The north bridge is a controller which controls the flow of data between the CPU and RAM, and to the AGP port.

In Fig. 46 you can see the north bridge, which has a large heat sink attached to it. It gets hot because of the often very large amounts of data traffic which pass through it. All around the north bridge you can see the devices it connects:

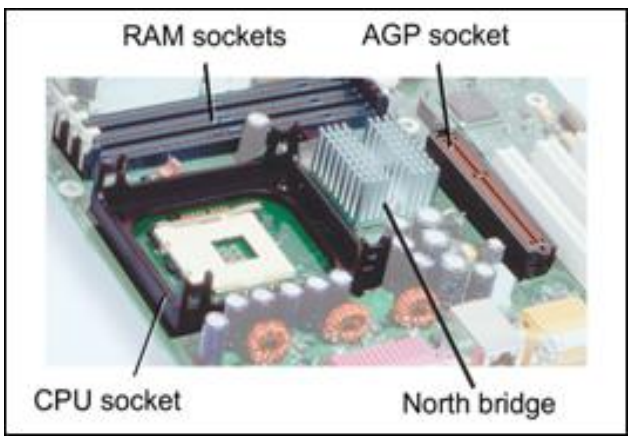


Fig. 46. The north bridge and its immediate surroundings. A lot of traffic runs through the north bridge, hence the heat sink.

The AGP is actually an I/O port. It is used for the video card. In contrast to the other I/O devices, the AGP port is connected directly to the north bridge, because it has to be as close to the RAM as possible. The same goes for the PCI Express x16 port, which is the replacement of AGP in new motherboards. But more on that later.

- Next chapter.
- Previous chapter.

- Next chapter.
- Previous chapter.

Chapter 7. The south bridge

The south bridge incorporates a number of different controller functions. It looks after the transfer of data to and from the hard disk and all the other I/O devices, and passes this data into the link channel which connects to the north bridge.

In Fig. 44 you can clearly see that the south bridge is physically located close to the PCI slots, which are used for I/O devices.

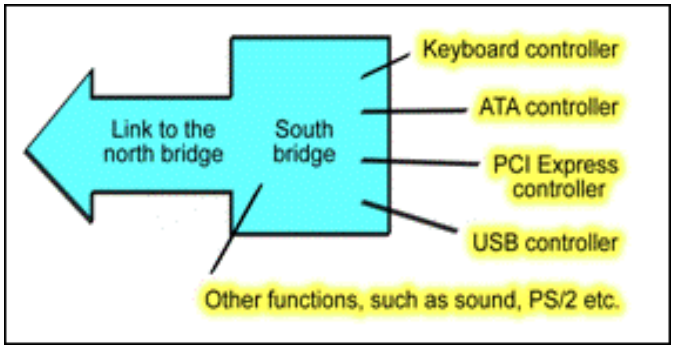


Fig. 47. The chipset's south bridge combines a number of controller functions into a single chip.

The various chipset manufacturers

Originally it was basically only Intel who supplied the chipsets to be used in motherboards. This was quite natural, since Intel knows everything about their own CPU's and can therefore produce chipsets which match them. But at the time the Pentium II and III came out, other companies began to get involved in this market. The Taiwanese company, VIA, today produces chipsets for both AMD and Intel processors, and these are used in a large number of motherboards.

Other companies (like SiS, nVidia, ATI and ALi) also produce chipsets, but these haven't (yet?) achieved widespread use. The CPU manufacturer, AMD, produces some chipsets for their own CPU's, but they also work together closely with VIA as the main supplier for Athlon motherboards.

Fig. 48. The Taiwanese company, VIA, has been a leader in the development of new chipsets in recent years.

Since all data transfers are managed by the chipset's two bridges, the chipset is the most important individual component on the motherboard, and new chipsets are constantly being developed.

The chipset determines the limits for clock frequencies, bus widths, etc. The chipset's built-in

controllers are also responsible for connecting I/O devices like hard disks and USB ports, thus the chipset also determines, in practise, which types of devices can be connected to the PC.

Fig. 49. The two chips which make up a typical chipset. Here we have VIA's model P4X266A, which was used in early motherboards for Pentium 4 processors.

Sound, network, and graphics in chipsets

Developments in recent years have led chipset manufacturers to attempt to place more and more functions in the chipset.

These extra functions are typically:

- Video card (integrated into the north bridge)
- Sound card (in the south bridge)
- Modem (in the south bridge)
- Network and Firewire (in the south bridge)

All these functions have traditionally been managed by separate devices, usually plug-in cards, which connect to the PC. But it has been found that these functions can definitely be incorporated into the chipset.

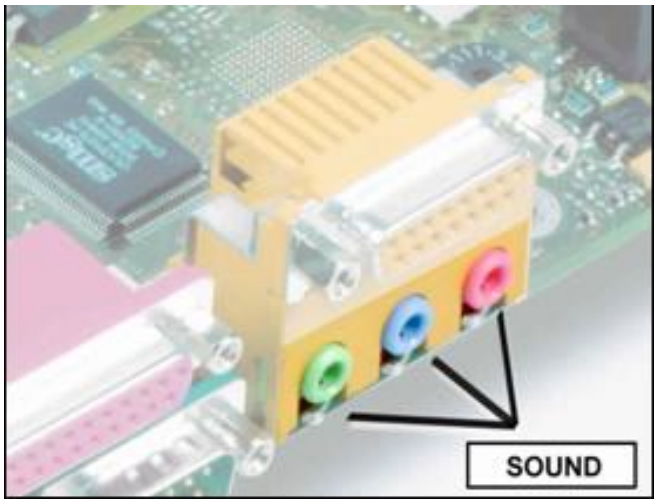


Fig. 50. Motherboard with built-in sound functionality.

Intel has, for many years, managed to produce excellent network cards (Ethernet 10/100 Mbps); so it is only natural that they should integrate this functionality into their chipsets.

Sound facilities in a chipset cannot be compared with “real” sound cards (like, for example, Sound Blaster Audigy). But the sound functions work satisfactorily if you only want to connect a couple of small speakers to the PC, and don’t expect perfect quality.

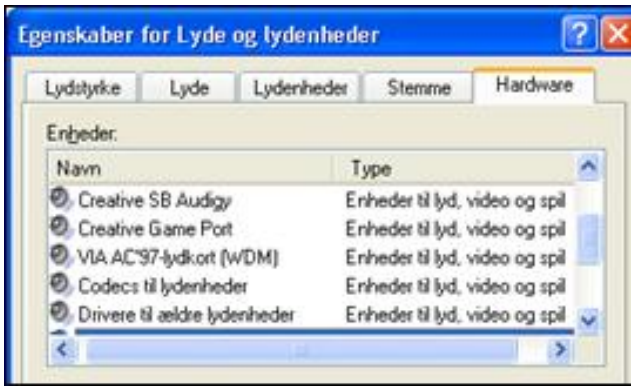


Fig. 51. This PC has two sound cards installed, as shown in this Windows XP dialog box. The VIA AC’97 is a sound card emulation which is built into the chipset.

Many chipsets also come with a built-in video card. The advantage is clear; you can save having a separate video card, which can cost a $100 or more.

Again, the quality can’t be compared with what you get with a separate, high quality, video card. But if you don’t particularly need support for multiple screens, DVI (for flat screens), super 3D performance for games, or TV-out, the integrated graphics controller can certainly do the job.

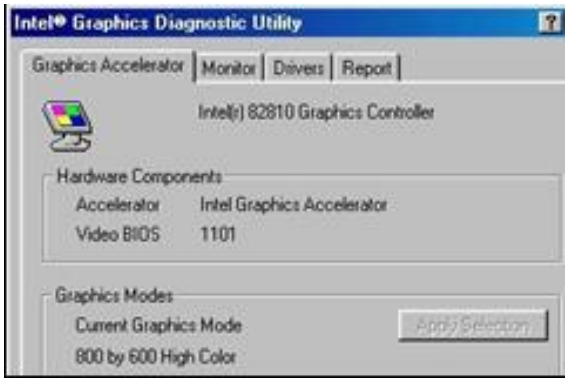


Fig. 52. This PC uses a video card which is built into the Intel i810 chipset.

It is important that the integrated sound and graphics functions can be *disabled*, so that you can replace them with a real sound or video card. The sound functions won’t cause any problems; you can always ask Windows to use a particular sound card instead of another one.

But the first Intel chipset with integrated graphics (the i810) did not allow for an extra video card to be installed. That wasn’t very smart, because it meant users were locked into using the built-in video card. In the subsequent chipset (i815), the problem was resolved.

Buying a motherboard

If you want to build a PC yourself, you have to start by choosing a motherboard. It is the foundation for the entire PC.

Most of the motherboards on the market are produced in Taiwan, where manufacturers like Microstar, Asus, Epox, Soltek and many others supply a wide range of different models. Note that a producer like Microstar supplies motherboards to brand name manufacturers like Fujitsu-Siemens, so you can comfortably trust in the quality. Taiwan is the leader in the area of motherboards.

The first issue to work out is, which CPU you want to use. For example, if you want to use a Pentium 4

from Intel, there is one line of motherboards you can choose between. If you choose an AthlonXP, there is another line. And the difference lies in which chipset is being used in the motherboard.

Fig. 53. A typical (technical) advertisement for a motherboard.

Once you have decided on a processor, you should try to get a motherboard with the latest chipset available, because new versions of chipsets continue to be released, with greater functionality. At the time of writing, for example, chipsets often include these functions:

- USB version 2.0.
- Dual channel RAM.
- Support for 400 and 533 MHz DDR2 RAM.
- Integrated Firewire ports.
- Serial ATA.
- Surround sound.
- Gigabit Ethernet.

You will most likely want to have these facilities (which are described later in the guide) on your PC. That is why it is important to choose the right motherboard with the latest generation chipset.

Extra facilities

In addition, it can sometimes be worth choosing a motherboard which has *extra facilities*. For example, all manufacturers have “luxury models” with built-in RAID controllers, making it possible to install several hard disks. There are motherboards around with loads of extra facilities, such as:

- Built-in RAID or (seldom) SCSI controller.

- Other network, screen and sound facilities.
- Wireless LAN.
- SmartCard/MemoryStick/etc. readers.

One of the advantages of building your own PC is that you can choose a really exciting motherboard.

Development is taking place rapidly, and by choosing the right motherboard, you can design the absolute latest PC on the market.

You can also find hundreds of articles on the Internet about each motherboard and chipset. So I can comfortably recommend you build your own PC, as long as you do your homework first! Make sure you read the rest of the guide before you start choosing a new motherboard!

- Next chapter.
- Previous chapter.

- Next chapter.
- Previous chapter.

Chapter 8. Inside and around the CPU

In this and the following chapters, I will focus on a detailed look at the CPU. One of the goals is help to you understand why manufacturers keep releasing new and more powerful processors. In order to explain that, we will have to go through what will at times be a quite detailed analysis of the CPU's inner workings.

Some of the chapters will probably be fairly hard to understand; I have spent a lot of time myself on my "research", but I hope that what I present in these chapters will shed some light on these topics.

Naturally, I will spend most of my time on the latest processors (the Athlon XP and Pentium 4). But we need to examine their internal architectures in light of the older CPU architectures, if we want to understand them properly. For this reason I will continually make comparisons across the various generations of CPU's.

I will now take you on a trip inside the CPU. We will start by looking at how companies like Intel and AMD can continue to develop faster processors.

Two ways to greater speed

Of course faster CPU's are developed as a result of hard work and lots of research. But there are two quite different directions in this work:

- More power and speed in the CPU, for example, from higher clock frequencies.
- Better exploitation of existing processor power.

Both approaches are used. It is a well-known fact that *bottlenecks* of various types drain the CPU of up to 75 % of its power. So if these can be removed or reduced, the PC can become significantly faster without having to raise the clock frequency dramatically.

It's just that it is very complicated to remove, for example, the bottleneck surrounding the front side bus, which I will show you later. So the manufacturers are forced to continue to raise the *working rate* (clock frequency), and hence to develop new *process technology*, so that CPU's with more power can come onto the market.

Clock frequencies

If we look at a CPU, the first thing we notice is the *clock frequency*. All CPU's have a working speed, which is regulated by a tiny *crystal*.

The crystal is constantly vibrating at a very large number of "beats" per second. For each clock tick, an impulse is sent to the CPU, and each pulse can, in principle, cause the CPU to perform one (or more) actions.

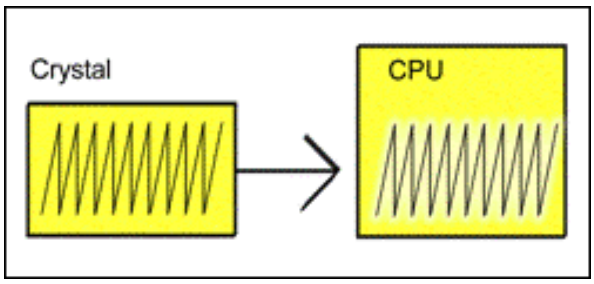


Fig. 54. The CPU's working speed is regulated by a crystal which "oscillates" millions of times each second.

The number of clock ticks per second is measured in *Hertz*. Since the CPU's crystal vibrates millions of times each second, the clock speed is measured in *millions* of oscillations (*megahertz* or *MHz*). Modern CPU's actually have clock speeds running into *billions* of ticks per second, so we have started having to use *gigahertz (GHz)*.

These are unbelievable speeds. See for yourself how short the period of time is between individual clock ticks at these frequencies. We are talking about billionths of a second:

Clock frequency	**Time period per clock tick**
133 MHz	0.000 000 008 000 seconds
1200 MHz	0.000 000 000 830 seconds
2 GHz	0.000 000 000 500 seconds

Fig. 55. The CPU works at an incredible speed.

The trend is towards ever increasing clock frequencies. Let's take a closer look at how this is possible.

More transistors

New types of processors are constantly being developed, for which the clock frequency keeps getting pushed up a notch. The original PC from 1981 worked at a modest 4.77 MHz, whereas the clock frequency 20 years later was up to 2 GHz.

In Fig. 56 you can see an overview of the last 20 years of development in this area. The table shows the seven *generations* of Intel processors which have brought about the PC revolution. The latest version of Pentium 4 is known under the code name Prescott.

Gen.	CPU	Yr (intr.)	Clock Frequency	No. of transistors
1	8088	1979	4.77- 8 MHz	29,000
2	80286	1982	6-12.5 MHz	134,000
3	80386	1985	16-33 MHz	275,000

4	80486	1989	25-100 MHz	1,200,000
5	Pentium Pentium MMX	1993 1997	60-200 MHz 166-300 MHz	3,100,000 4,500,000
6	Pentium Pro Pentium II Pentium III	1995 1997 1999	150-200 MHz 233-450 MHz 450-1200 MHz	5,500,000 7,500,000 28,000,000
7	Pentium 4 “Prescott“	2000 2002 2003 2004	1400-2200 2200-2800 2600-3200 2800-3600	42,000,000 55,000,000 55,000,000 125,000,000

Fig. 56. Seven generations of CPU’s from Intel. The number of transistors in the Pentium III and 4 includes the L2 cache.

Each processor has been on the market for several years, during which time the clock frequency has increased. Some of the processors were later released in improved versions with higher clock frequencies, I haven’t included the Celeron in the overview processor. Celerons are specially discount versions of the Pentium II, III, and 4 processors.

Anyone can see that there has been an unbelievable development. Modern CPU’s are one thousand times more powerful than the very first ones.

In order for the industry to be able to develop faster CPU’s each year, new manufacturing methods are required. *More and more transistors* have to be squeezed into smaller and smaller chips.

Fig. 57.
A photograph from one of Intel’s factories, in which a technician displays the Pentium 4 processor core. It is a tiny piece of silicon which contains 42 million transistors.

- Next chapter.
- Previous chapter.

- Next chapter.
- Previous chapter.

Chapter 9. Moores' Law

This development was actually described many years ago, in what we call Moores Law.

Right back in 1965, Gordon Moore predicted (in the Electronics journal), that the number of transistors in processors (and hence their speed) would be able to be doubled every 18 months.

Moore expected that this regularity would at least apply up until 1975. But he was too cautious; we can see that the development continues to follow Moores Law today, as is shown in Fig. 59.

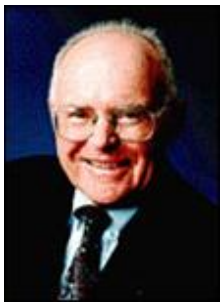


Fig. 58. In 1968, Gordon Moore helped found Intel.

If we try to look ahead in time, we can work out that in 2010 we should have processors containing 3 billion transistors. And with what clock frequencies? You'll have to guess that for yourself.

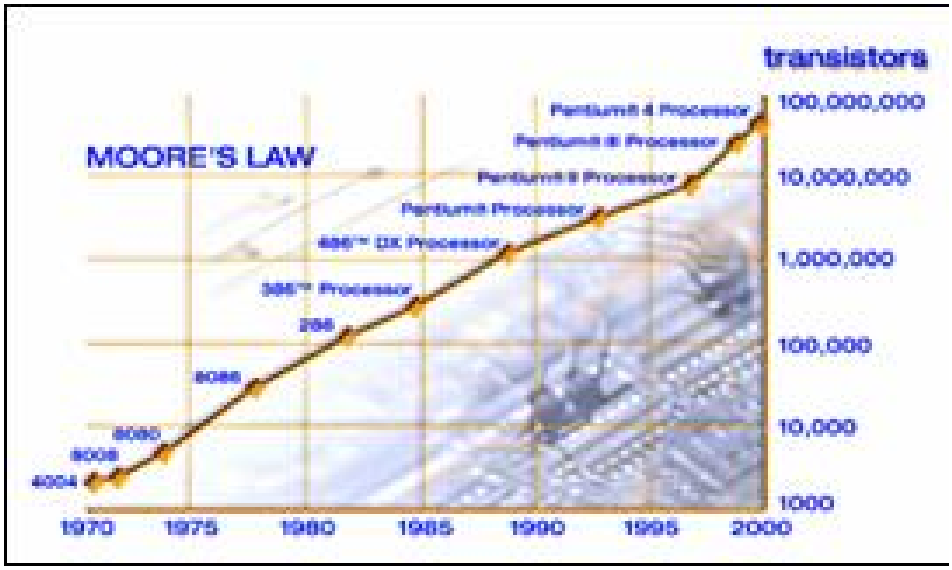


Fig. 59. Moores Law (from Intels website).

Process technology

The many millions of transistors inside the CPU are made of, and connected by, ultra thin electronic *tracks*. By making these electronic tracks even narrower, even more transistors can be squeezed into a small slice of silicon.

The width of these electronic tracks is measured in *microns* (or *micrometers*), which are millionths of a metre.

For each new CPU generation, the track width is reduced, based on new technologies which the chip manufacturers keep developing. At the time of writing, CPU's are being produced with a track width of 0.13 microns, and this will be reduced to 0.09 and 0.06 microns in the next generations.

Fig. 60. CPU's are produced in extremely high-technology environments ("clean rooms"). Photo courtesy of AMD.

In earlier generations aluminium was used for the current carrying tracks in the chips. With the change to 0.18 and 0.13-micron technology, aluminium began to be replaced with *copper*. Copper is cheaper, and it carries current better than aluminium. It had previously been impossible to insulate the copper tracks from the surrounding silicon, but IBM solved this problem in the late 1990's.

AMD became the first manufacturer to mass-produce CPU's with copper tracks in their chip factory *fab 30* in Dresden, Germany. A new generation of chips requires new chip factories (fabs) to produce it, and these cost billions of dollars to build. That's why they like a few years to pass between each successive generation. The old factories have to have time to pay for themselves before new ones start to be used.

Fig. 61. AMD's Fab 30 in Dresden, which was the first factory to mass-produce copper-based CPU's.

A grand new world ...

We can expect a number of new CPU's in this decade, all produced in the same way as they are now – just with smaller track widths. But there is no doubt that we are nearing the physical limits for how small the transistors produced using the existing technology can be. So intense research is underway to find new materials, and it appears that nanotransistors, produced using organic (carbon-based) semiconductors, could take over the baton from the existing process technology.

Bell Labs in the USA has produced nanotransistors with widths of just one molecule. It is claimed that this process can be used to produce both CPU's and RAM circuits up to 1000 times smaller than what we have today!

Less power consumption

The types of CPU's we have today use a fairly large amount of electricity when the PC is turned on and is processing data. The processor, as you know, is installed in the motherboard, from which it receives power. There are actually two different voltage levels, which are both supplied by the motherboard:

- One voltage level which powers the CPU core (*kernel voltage*).
- Another voltage level which powers the CPU's I/O ports, which is typically 3.3 volts.

As the track width is reduced, more transistors can be placed within the same area, and hence the voltage can be reduced.

As a consequence of the narrower process technology, the *kernel voltage* has been reduced from 3 volts to about 1 volt in recent years. This leads to lower power consumption per transistor. But since the number of transistors increases by a corresponding amount in each new CPU generation, the end result is often that the total power consumption is unchanged.

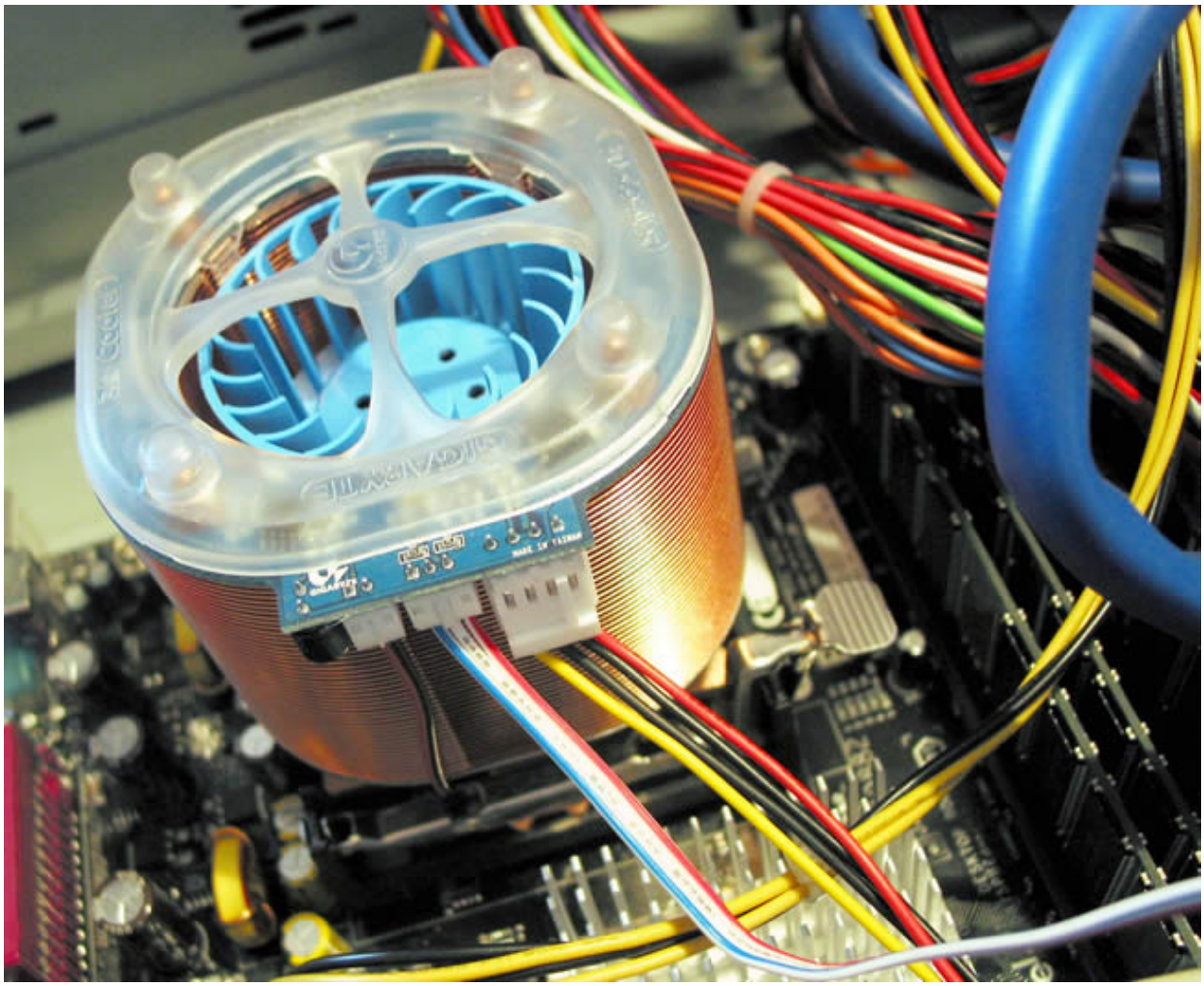
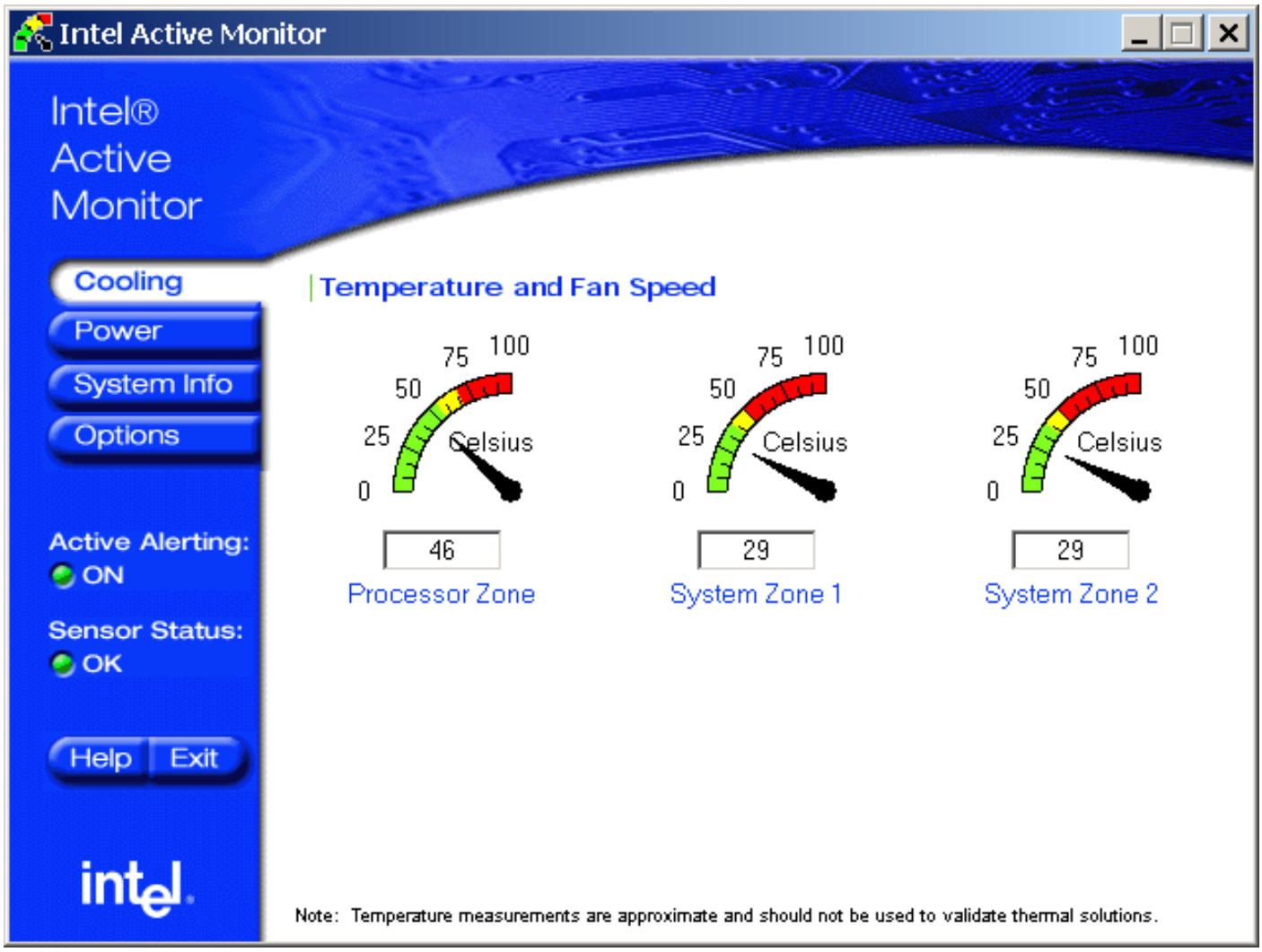


Fig. 62. A powerful fan. Modern CPU's require something like this.

It is very important to cool the processor; a CPU can easily burn 50-120 Watts. This produces a fair amount of heat in a very small area, so without the right cooling fan and motherboard design, a Gigahertz processor could quickly burn out.

Modern processors contain a thermal diode which can raise the alarm if the CPU gets to hot. If the motherboard and BIOS are designed to pay attention to the diode's signal, the processor can be shut down temporarily so that it can cool down.



Figur 63. The temperatures on the motherboard are constantly reported to this program..

Cooling is a whole science in itself. Many “nerds” try to push CPU’s to work at higher clock speeds than they are designed for. This is often possible, but it requires very good cooling – and hence often huge cooling units.

30 years development

Higher processor speeds require more transistors and narrower electronic tracks in the silicon chip. In the overview in Fig. 64 you can see the course of developments in this area.

Note that the 4004 processor was never used for PC’s. The 4004 was Intel’s first commercial product in 1971, and it laid the foundation for all their later CPU’s. It was a 4-bit processor which worked at 108 KHz (0.1 MHz), and contained 2,250 transistors. It was used in the first *pocket calculators*, which I can personally remember from around 1973-74 when I was at high school. No-one could have predicted that the device which replaced the slide rule, could develop, in just 30 years, into a Pentium 4 based super PC.

If, for example, the development in automobile technology had been just as fast, we would today be able to drive from Copenhagen to Paris in just 2.8 seconds!

Year	Intel CPU	Technology (track width)
1971	4004	10 microns
1979	8088	3 microns
1982	80286	1.5 microns
1985	80386	1 micron
1989	80486	1.0/0.8 microns

1993	Pentium	0.8/0.5/0.35 microns
1997	Pentium II	0.28/0.25 microns
1999	Pentium III	0.25/0.18/0.13 microns
2000-2003	Pentium 4	0.18/0.13 microns
2004-2005	Pentium 4 ”Prescott”	0.09 microns

Fig. 64. The high clock frequencies are the result of new process technology with smaller electronic ”tracks”.

A conductor which is 0.09 microns (or 90 nanometres) thick, is 1150 times thinner than a normal human hair. These are tiny things we are talking about here.

Wafers and die size

Another CPU measurement is its *die size*. This is the size of the actual silicon sheet containing all the transistors (the tiny area in the middle of Fig. 33 on page 15).

At the chip factories, the CPU cores are produced in so-called *wafers*. These are round silicon sheets which typically contain 150-200 processor cores (*dies*).

The smaller one can make each *die*, the more economical production can become. A big die is also normally associated with greater power consumption and hence also requires cooling with a powerful fan (e.g. see Fig. 63 on page 25 and Fig. 124 on page 50).

Figur 65. A technician from Intel holding a wafer. This slice of silicon contains hundreds of tiny processor cores, which end up as CPU's in everyday PC's.

You can see the measurements for a number of CPU's below. Note the difference, for example, between a Pentium and a Pentium II. The latter is much smaller, and yet still contains nearly 2½ times as many transistors. Every reduction in die size is welcome, since the smaller this is, the more processors can fit on a wafer. And that makes production cheaper.

CPU	Track width	Die size	Number of transistors
Pentium	0.80	294 mm^2	3.1 mil.
Pentium MMX	0.28	140 mm^2	4.5 mil.
Pentium II	0.25	131 mm^2	7.5 mil.
Athlon	0.25	184 mm^2	22 mil.
Pentium III	0.18	106 mm^2	28 mil.
Pentium III	0.13	80 mm^2	28 mil.
Athlon XP	0.18	128 mm^2	38 mil.
Pentium 4	0.18	217 mm^2	42 mil.
Pentium 4	0.13	145 mm^2	55 mil.
Athlon XP+	0.13	115 mm^2	54 mil.
Athlon 64 FX	0,13	193 mm^2	106 mill.
Pentium 4	0.09	112 mm^2	125 mil.

Fig. 66. The smaller the area of each processor core, the more economical chip production can be.

The modern CPU generations

As mentioned earlier, the various CPU's are divided into generations (see also Fig. 56 on page 23).

At the time of writing, we have started on the *seventh* generation. Below you can see the latest processors from Intel and AMD, divided into these generations. The transitions can be a bit hazy. For example, I'm not sure whether AMD's K6 belongs to the 5th or the 6^{th} generation. But as a whole, the picture is as follows:

Generation	CPU's
5th	Pentium, Pentium MMX, K5, K6
6th	Pentium Pro, K6-II, Pentium II, K6-3, Athlon, Pentium III
7th	Pentium 4, Athlon XP
8th.	Athlon 64 FX, Pentium 5

Fig. 67. The latest generations of CPU's.

- Next chapter.
- Previous chapter.

- Next chapter.
- Previous chapter.

Chapter 10. The cache

In the previous chapter, I described two aspects of the ongoing development of new CPU's – increased clock frequencies and the increasing number of transistors being used. Now it is time to look at a very different yet related technology – the processor's connection to the RAM, and the use of the *L1* and *L2 caches*.

Speed conflict

The CPU works internally at very high clock frequencies (like 3200 MHz), and no RAM can keep up with these.

The most common RAM speeds are between 266 and 533 MHz. And these are just a fraction of the CPU's working speed. So there is a great chasm between the machine (the CPU) which slaves away at perhaps 3200 MHz, and the "conveyor belt", which might only work at 333 MHz, and which has to ship the data to and from the RAM. These two subsystems are simply poorly matched to each other.

If nothing could be done about this problem, there would be no reason to develop faster CPU's. If the CPU had to wait for a bus, which worked at one sixth of its speed, the CPU would be *idle* five sixths of the time. And that would be pure waste.

The solution is to insert small, intermediate stores of high-speed RAM. These buffers (*cache* RAM) provide a much more efficient transition between the fast CPU and the slow RAM. Cache RAM operates at higher clock frequencies than normal RAM. Data can therefore be read more quickly from the cache.

Data is constantly being moved

The cache delivers its data to the CPU *registers*. These are tiny storage units which are placed right inside the processor core, and they are the absolute fastest RAM there is. The size and number of the registers is designed very specifically for each type of CPU.

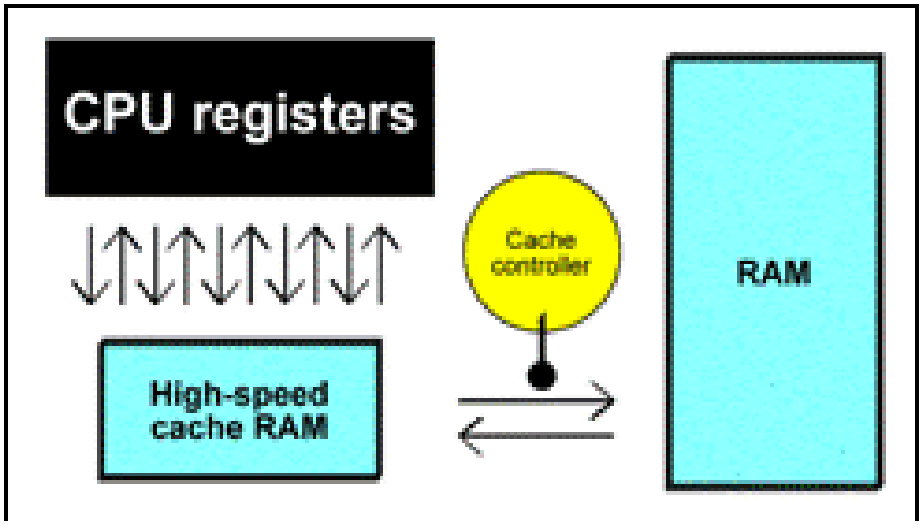


Fig. 68. Cache RAM is much faster than normal RAM.

The CPU can move data in different sized packets, such as *bytes* (8 bits), *words* (16 bits), *dwords* (32 bits) or *blocks* (larger groups of bits), and this often involves the registers. The different data packets are constantly moving back and forth:

- from the CPU registers to the Level 1 cache.
- from the L1 cache to the registers.
- from one register to another
- from L1 cache to L2 cache, and so on…

The cache stores are a central bridge between the RAM and the registers which exchange data with the processor's *execution units*.

The optimal situation is if the CPU is able to constantly work and fully utilize all clock ticks. This would mean that the registers would have to always be able to fetch the data which the execution units require. But this it not the reality, as the CPU typically only utilizes 35% of its clock ticks. However, without a cache, this utilization would be even lower.

Bottlenecks

CPU caches are a remedy against a very specific set of "bottleneck" problems. There are lots of "bottlenecks" in the PC – transitions between fast and slower systems, where the fast device has to wait before it can deliver or receive its data. These bottle necks can have a very detrimental effect on the PC's total performance, so they must be minimised.

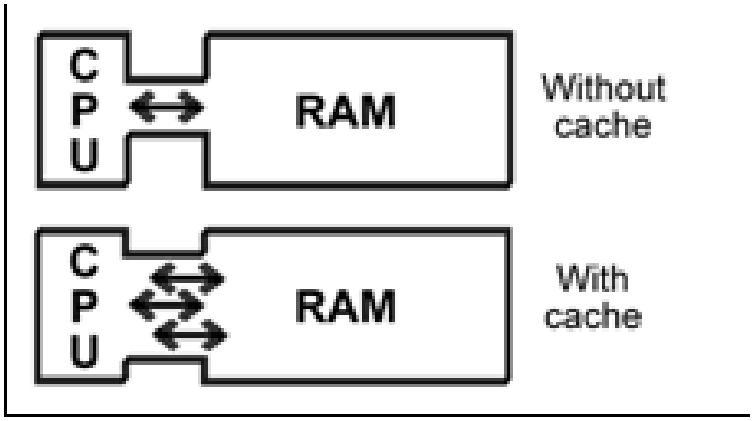


Fig. 69. A cache increases the CPU's capacity to fetch the right data from RAM.

The absolute worst bottleneck exists between the CPU and RAM. It is here that we have the heaviest data traffic, and it is in this area that PC manufacturers are expending a lot of energy on new development. Every new generation of CPU brings improvements relating to the front side bus.

The CPU's cache is "intelligent", so that it can reduce the data traffic on the front side bus. The cache controller constantly monitors the CPU's work, and always tries to read in precisely the data the CPU needs. When it is successful, this is called a *cache hit*. When the cache does not contain the desired data, this is called a *cache miss*.

Two levels of cache

The idea behind cache is that it should function as a "near store" of fast RAM. A store which the CPU can always be supplied from.

In practise there are always at least two close stores. They are called *Level 1, Level 2,* and (if applicable) *Level 3* cache. Some processors (like the Intel Itanium) have three levels of cache, but these are only used for very special server applications. In standard PC's we find processors with L1 and L2 cache.

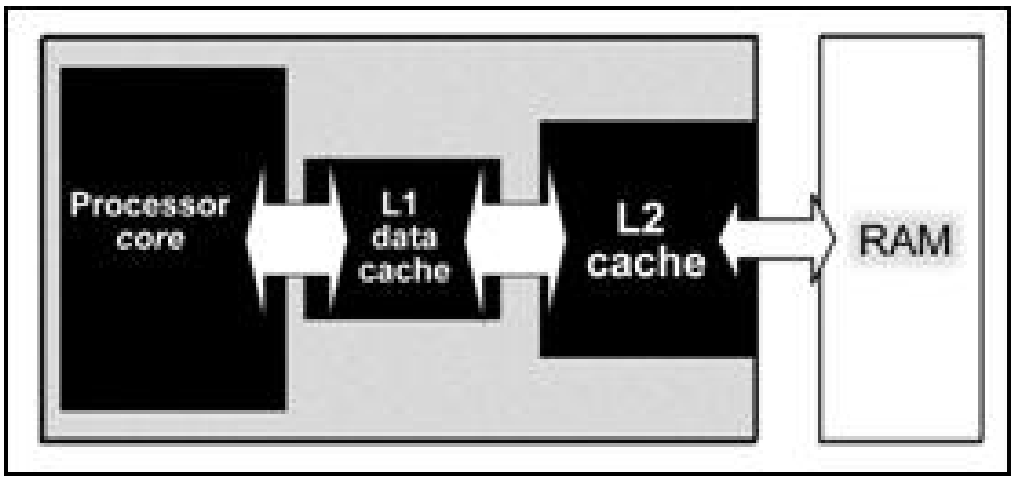


Fig. 70. The cache system tries to ensure that relevant data is constantly being fetched from RAM, so that the CPU (ideally) never has to wait for data.

L1 cache

Level 1 cache is built into the actual processor core. It is a piece of RAM, typically 8, 16, 20, 32, 64 or 128 Kbytes, which operates at the same clock frequency as the rest of the CPU. Thus you could say the L1 cache is part of the processor.

L1 cache is normally divided into two sections, one for *data* and one for *instructions*. For example, an Athlon processor may have a 32 KB data cache and a 32 KB instruction cache. If the cache is common for both data and instructions, it is called a *unified cache*.

- Next chapter.
- Previous chapter.

- Next chapter.
- Previous chapter.

Chapter 11. The L2 cache

The level 2 cache is normally much bigger (and unified), such as 256, 512 or 1024 KB. The purpose of the L2 cache is to constantly read in slightly larger quantities of data from RAM, so that these are available to the L1 cache.

In the earlier processor generations, the L2 cache was placed *outside* the chip: either on the motherboard (as in the original Pentium processors), or on a special module together with the CPU (as in the first Pentium II's).

Fig. 71. An old Pentium II module. The CPU is mounted on a rectangular printed circuit board, together with the L2 cache, which is two chips here. The whole module is installed in a socket on the motherboard. But this design is no longer used.

As process technology has developed, it has become possible to make room for the L2 cache inside the actual processor chip. Thus the L2 cache has been integrated and that makes it function much better in relation to the L1 cache and the processor core.

The L2 cache is not as fast as the L1 cache, but it is still much faster than normal RAM.

CPU	L2 cache
Pentium, K5, K6	External, on the motherboard
Pentium Pro	Internal, in the CPU

Pentium II, Athlon	External, in a module close to the CPU
Celeron (1st generation)	None
Celeron (later gen.), Pentium III, Athlon XP, Duron, Pentium 4	Internal, in the CPU

Fig. 72. It has only been during the last few CPU generations that the level 2 cache has found its place, integrated into the actual CPU.

Traditionally the L2 cache is connected to the front side bus. Through it, it connects to the chipset's north bridge and RAM:

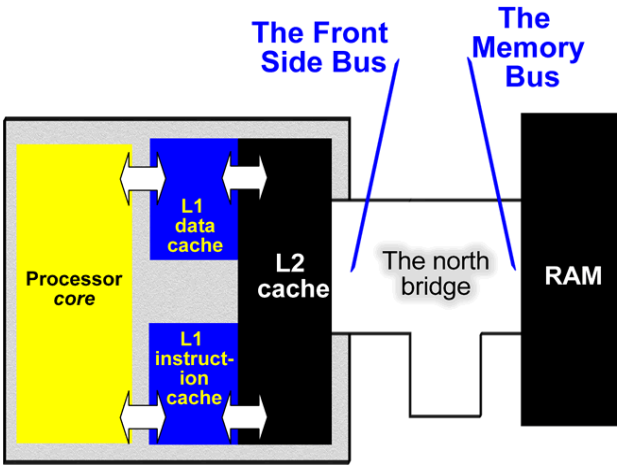


Fig. 73. The way the processor uses the L1 and L2 cache has crucial significance for its utilisation of the high clock frequencies.

The level 2 cache takes up a lot of the chip's die, as millions of transistors are needed to make a large cache. The integrated cache is made using SRAM (*static* RAM), as opposed to normal RAM which is *dynamic* (DRAM).

While DRAM can be made using one transistor per bit (plus a capacitor), it costs 6 transistors (or more) to make one bit of SRAM. Thus 256 KB of L2 cache would require more than 12 million transistors. Thus it has only been since fine process technology (such as 0.13 and 0.09 microns) was developed that it became feasible to integrate a large L2 cache into the actual CPU. In Fig. 66 on page 27, the number of transistors includes the CPU's integrated cache.

Powerful bus

The bus between the L1 and L2 cache is presumably THE place in the processor architecture which has the greatest need for high bandwidth. We can calculate the theoretical maximum bandwidth by multiplying the bus width by the clock frequency. Here are some examples:

CPU	Bus width	Clock frequency	Theoretical bandwidth
Intel Pentium III	64 bits	1400 MHz	11.2 GB/sek.
AMD Athlon XP+	64 bits	2167 MHz	17.3 GB/sek.

AMD Athlon 64	64 bits	2200 MHz	17,6 GB/sek.
AMD Athlon 64 FX	128 bits	2200 MHz	35,2 GB/sek.
Intel Pentium 4	256 bits	3200 MHz	102 GB/sek.

Fig. 74. Theoretical calculations of the bandwidth between the L1 and L2 cache.

Different systems

There are a number of different ways of using caches. Both Intel and AMD have saved on L2 cache in some series, in order to make cheaper products. But there is no doubt, that the better the cache – both L1 and L2 – the more efficient the CPU will be and the higher its performance.

AMD have settled on a fairly large L1 cache of 128 KB, while Intel continue to use relatively small (but efficient) L1 caches.

On the other hand, Intel uses a 256 bit wide bus on the “inside edge” of the L2 cache in the Pentium 4, while AMD only has a 64-bit bus (see Fig. 74).

Fig. 75. Competing CPU’s with very different designs.

AMD uses *exclusive caches* in all their CPU’s. That means that the same data can’t be present in both caches at the same time, and that is a clear advantage. It’s not like that at Intel.

However, the Pentium 4 has a more advanced cache design with *Execution Trace Cache* making up 12 KB of the 20 KB Level 1 cache. This instruction cache works with *coded instructions*, as described on page 35.

CPU	L1 cache	L2 cache
Athlon XP	128 KB	256 KB
Athlon XP+	128 KB	512 KB
Pentium 4 (I)	20 KB	256 KB
Pentium 4 (II, “Northwood”)	20 KB	512 KB
Athlon 64	128 KB	512 KB
Athlon 64 FX	128 KB	1024 KB

Pentium 4 (III, “Prescott”)	28 KB	1024 KB

Fig. 76. The most common processors and their caches.

Latency

A very important aspect of all RAM – cache included – is *latency*. All RAM storage has a certain latency, which means that a certain number of clock ticks (*cycles*) must pass between, for example, two reads. L1 cache has less latency than L2; which is why it is so efficient.

When the cache is bypassed to read directly from RAM, the latency is many times greater. In Fig. 77 the number of wasted clock ticks are shown for various CPU’s. Note that when the processor core has to fetch data from the actual RAM (when both L1 and L2 have failed), it costs around 150 clock ticks. This situation is called *stalling* and needs to be avoided.

Note that the Pentium 4 has a much smaller L1 cache than the Athlon XP, but it is significantly faster. It simply takes fewer clock ticks (*cycles*) to fetch data:

Latency	Pentium II	Athlon	Pentium 4
L1 cache:	3 cycles	3 cycles	2 cycles
L2 cache:	18 cycles	6 cycles	5 cycles

Fig. 77. Latency leads to wasted clock ticks; the fewer there are of these, the faster the processor will appear to be.

Intelligent ”data prefetch”

In CPU’s like the Pentium 4 and Athlon XP, a handful of support mechanisms are also used which work in parallel with the cache. These include:

A *hardware auto data prefetch unit*, which attempts to *guess* which data should be read into the cache. This device monitors the instructions being processed and predicts what data the next job will need.

Related to this is the *Translation Look-aside Buffer,* which is also a kind of cache. It contains information which constantly supports the supply of data to the L1 cache, and this buffer is also being optimised in new processor designs. Both systems contribute to improved exploitation of the limited bandwidth in the memory system.

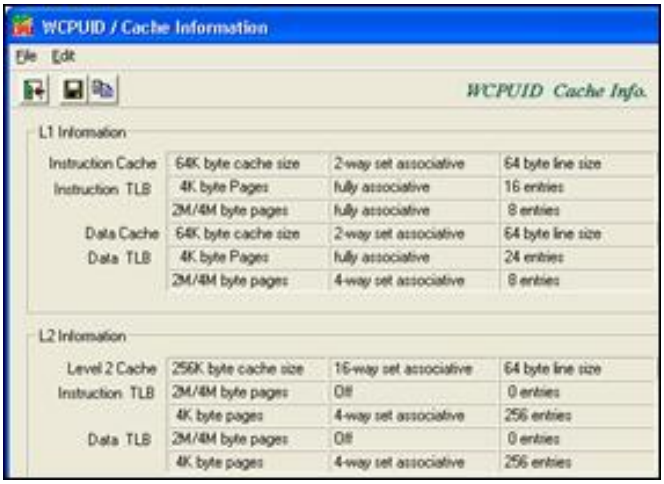


Fig. 78. The WCPUID program reports on cache in an Athlon processor.

Conclusion

L1 and L2 cache are important components in modern processor design. The cache is crucial for the utilisation of the high clock frequencies which modern process technology allows. Modern L1 caches are extremely effective. In about 96-98% of cases, the processor can find the data and instructions it needs in the cache. In the future, we can expect to keep seeing CPU's with larger L2 caches and more advanced memory management. As this is the way forward if we want to achieve more effective utilisation of the CPU's clock ticks. Here is a concrete example:

In January 2002 Intel released a new version of their top processor, the Pentium 4 (with the codename, "Northwood"). The clock frequency had been increased by 10%, so one might expect a 10% improvement in performance. But because the integrated L2 cache was also doubled from 256 to 512 KB, the gain was found to be all of 30%.

CPU	L2 cache	Clock freq.	Improvement
Intel Pentium 4 (0.18 micron)	256 KB	2000 MHz	
Intel Pentium 4 (0.13 micron)	512 KB	2200 MHz	+30%

Fig. 79. Because of the larger L2 cache, performance increased significantly.

In 2002 AMD updated the Athlon processor with the new "Barton" core. Here the L2 cache was also doubled from 256 to 512 KB in some models. In 2004 Intel came with the "Prescott" core with 1024 KB L2 cache, which is the same size as in AMD's Athlon 64 processors. Some *Extreme Editions* of Pentium 4 even uses 2 MB of L2 cache.

Xeon for servers

Intel produces special server models of their Pentium III and Pentium 4 processors. These are called Xeon, and are characterised by very large L2 caches. In an Intel Xeon the 2 MB L2 cache uses 149,000,000 transistors.

Xeon processors are incredibly expensive (about Euro 4,000 for the top models), so they have never achieved widespread distribution.

They are used in high-end servers, in which the CPU only accounts for a small part of the total price.

Otherwise, Intel's 64 bit server CPU, the Itanium. The processor is supplied in modules which include 4 MB L3 cache of 300 million transistors.

Multiprocessors

Several Xeon processors can be installed on the same motherboard, using special chipsets. By connecting 2, 4 or even 8 processors together, you can build a very powerful computer.

These MP (*Multiprocessor*) machines are typically used as servers, but can also be used as powerful workstations, for example, to perform demanding 3D graphics and animation tasks. AMD has the Opteron processors, which are server-versions of the Athlon 64. Not all software can make use of the PC's extra processors; the programs have to be designed to do so. For example, there are professional versions of Windows NT, 2000 and XP, which support the use of several processors in one PC.

See also the discussion of Hyper Threading, which allows a Pentium 4 processor to appear as an MP system. Both Intel and AMD also works on dual-core processors.

- Next chapter.
- Previous chapter.

- Next chapter.
- Previous chapter.

Chapter 12. Data and instructions

Now it's time to look more closely at the work of the CPU. After all, what does it actually do?

Instructions and data

Our CPU processes *instructions* and *data*. It receives orders from the software. The CPU is fed a gentle stream of binary data via the RAM.

These instructions can also be called *program code*. They include the commands which you constantly – via user programs – send to your PC using your keyboard and mouse. Commands to print, save, open, etc.

Data is typically *user data*. Think about that email you are writing. The actual contents (the text, the letters) is user data. But when you and your software say "send", your are sending program code (instructions) to the processor:

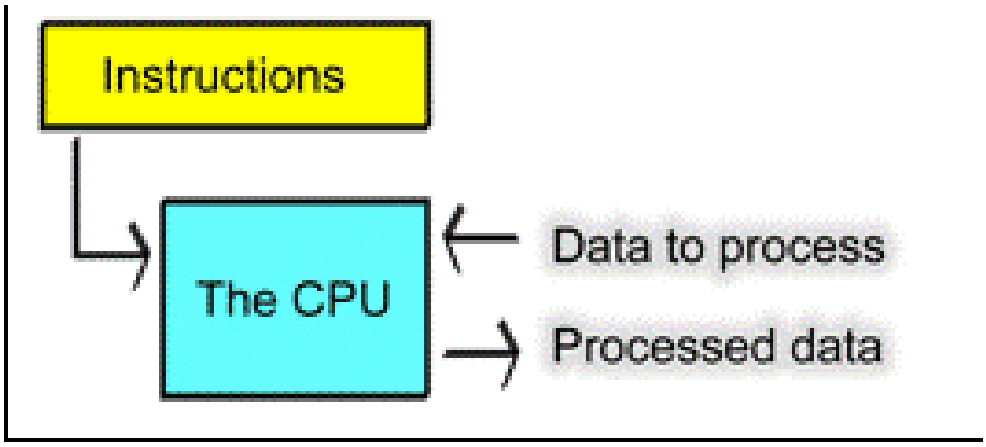


Fig. 80. The instructions process the user data.

Instructions and compatibility

Instructions are *binary code* which the CPU can understand. Binary code (machine code) is the mechanism by which PC programs communicate with the processor.

All processors, whether they are in PC's or other types of computers, work with a particular *instruction set*. These instructions are the language that the CPU understands, and thus all programs have to communicate using these instructions. Here is a simplified example of some "machine code" – instructions written in the language the processor understands:

```
proc near
```

```
mov AX,01
mov BX,01
inc AX
add BX,AX
```

You can no doubt see that it wouldn't be much fun to have to use these kinds of instructions in order to write a program. That is why people use programming tools. Programs are written in a programming language (like *Visual Basic* or *C++*). But these program lines have to be translated into machine code, they have to be *compiled*, before they can run on a PC. The compiled program file contains instructions which can be understood by the particular processor (or processor family) the program has been "coded" for:

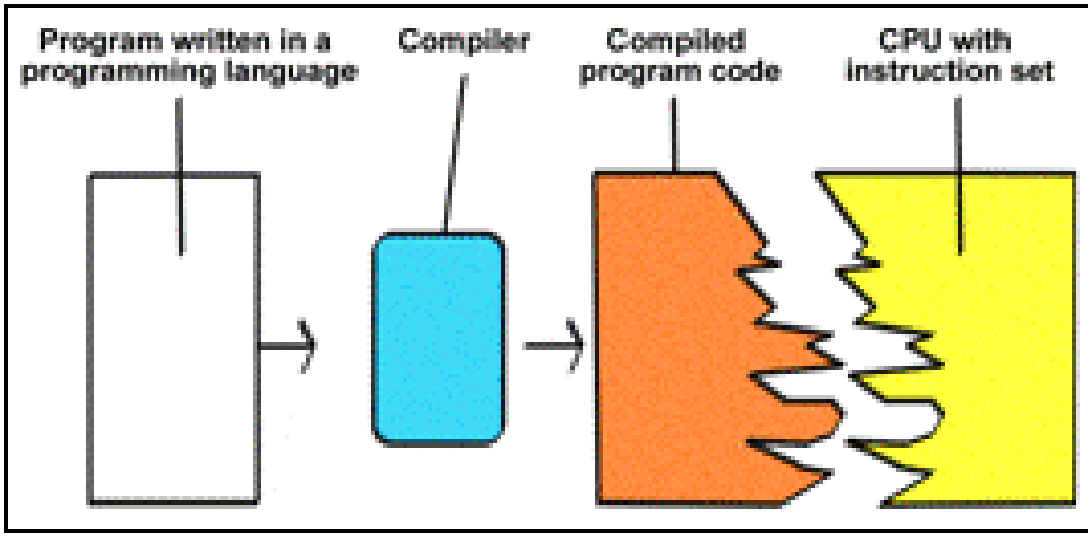


Fig. 81. The program code produced has to match the CPU's instruction set. Otherwise it cannot be run.

The processors from AMD and Intel which we have been focusing on in this guide, are *compatible*, in that they understand the same instructions.

There can be big differences in the way two processors, such as the Pentium and Pentium 4, process the instructions *internally*. But externally – from the programmer's perspective – they all basically function the same way. All the processors in the PC family (regardless of manufacturer) can execute the same instructions and hence the same programs.

And that's precisely the advantage of the PC: Regardless of which PC you have, it can run the Windows programs you want to use.

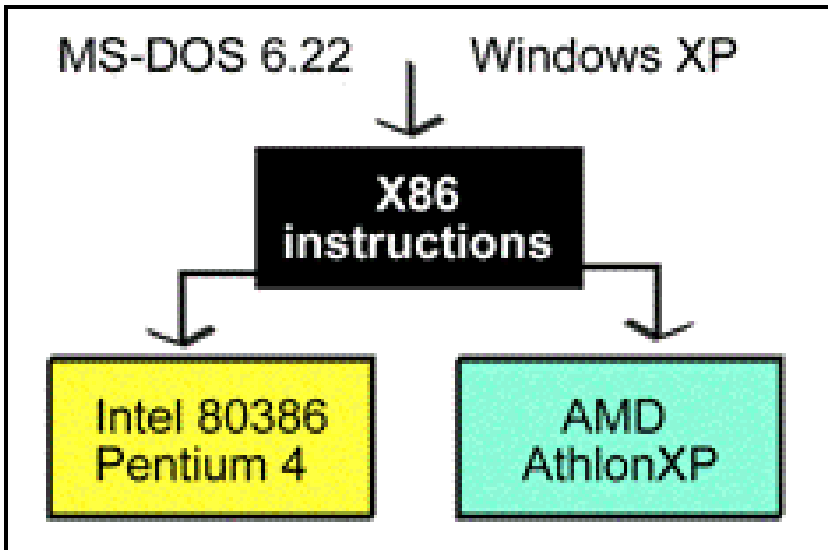


Fig. 82. The x86 instruction set is common to all PC's.

As the years have passed, changes have been made in the instruction set along the way. A PC with a Pentium 4 processor from 2002 can handle very different applications to those which an IBM XT with an 8088 processor from 1985 can. But on the other hand, you can expect all the programs which could run on the 8088, to still run on a Pentium 4 and on a Athlon 64. The software is *backwards compatible*.

The entire software industry built up around the PC is based on the common *x86* instruction, which goes back to the earliest PC's. Extensions have been made, but the original instruction set from 1979 is still being used.

x86 and CISC

People sometimes differentiate between RISC and CISC based CPU's. The (x86) instruction set of the original Intel 8086 processor is of the CISC type, which stands for *Complex Instruction Set Computer.*

That means that the instructions are quite diverse and complex. The individual instructions vary in length from 8 to 120 bits. It is designed for the 8086 processor, with just 29,000 transistors. The opposite of CISC, is RISC instructions.

RISC stands for *Reduced Instruction Set Computer*, which is fundamentally a completely different type of instruction set to CISC. RISC instructions can all have the same length (e.g. 32 bits). They can therefore be executed much faster than CISC instructions. Modern CPU's like the AthlonXP and Pentium 4 are based on a mixture of RISC and CISC.

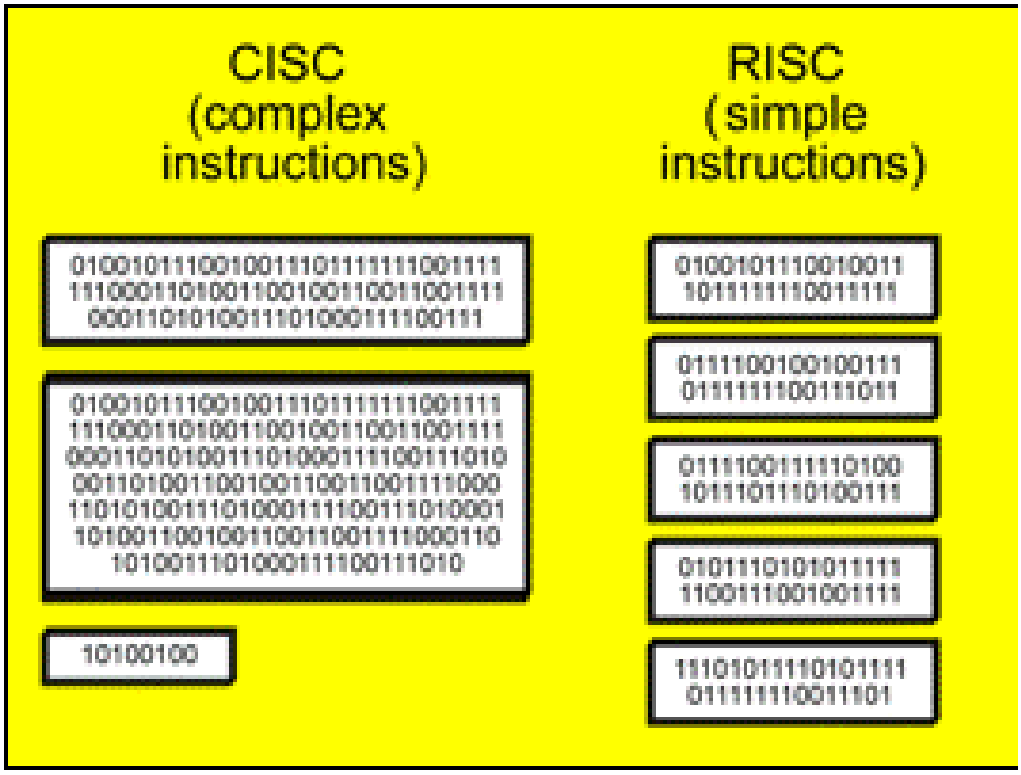


Fig. 83. PC's running Windows still work with the old fashioned CISC instructions.

In order to maintain compatibility with the older DOS/Windows programs, the later CPU's still understand CISC instructions. They are just converted to shorter, more RISC-like, sub-operations (called *micro-ops*), before being executed. Most CISC instructions can be converted into 2-3 micro-ops.

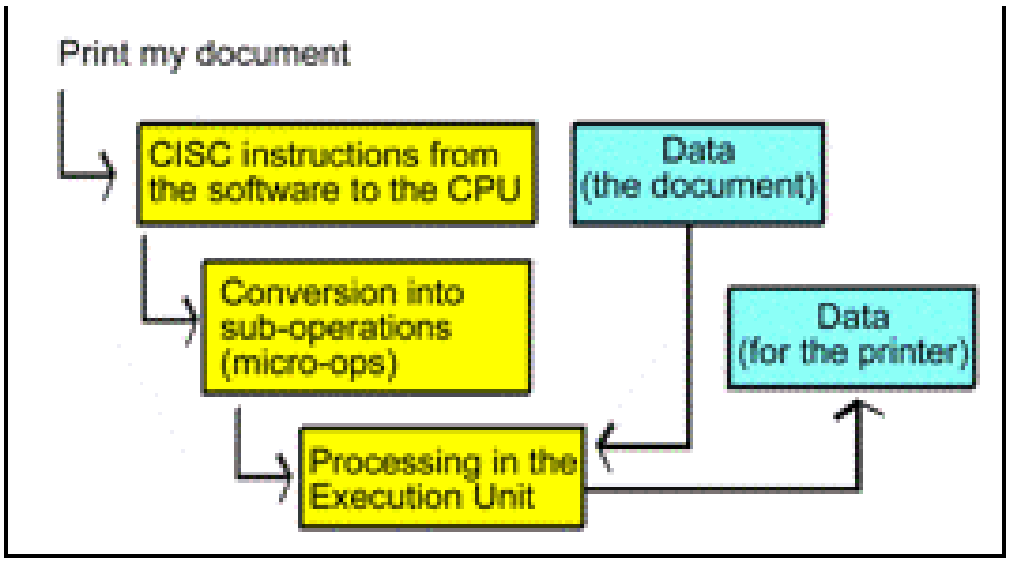


Fig. 84. The CISC instructions are decoded before being executed in a modern processor. This preserves compatibility with older software.

Extensions to the instruction set

For each new generation of CPU's, the original instruction set has been extended. The 80386 processor added 26 new instructions, the 80486 added six, and the Pentium added eight new instructions.

At the same time, execution of the instructions was made more efficient. For example, it took an 80386 processor six clock ticks to add one number to a running summation. This task could be done in the 80486 (see page 40), in just two clock ticks, due to more efficient decoding of the instructions.

These changes have meant that certain programs require at least a 386 or a Pentium processor in order to run. This is true, for example, of all Windows programs. Since then, the MMX and SSE extensions have followed, which are completely new instruction sets which will be discussed later in the guide. They can make certain parts of program execution much more efficient.

Another innovation is the 64-bit extension, which both AMD and Intel use in their top-processors. Normally the pc operates in 32-bit mode, but one way to improve the performance is using a 64-bit mode. This requires new software, which is not available yet.

9. Inside the CPU

Instructions have to be decoded, and not least, *executed*, in the CPU. I won't go into details on this subject; it is much too complicated. But I will describe a few factors which relate to the execution of instructions. My description has been extremely simplified, but it is relevant to the understanding of the microprocessor. This chapter is probably the most complicated one in the guide – you have been warned! It's about:

- Pipelines
- Execution units

If we continue to focus on speeding up the processor's work, this optimisation must also apply to the instructions – the quicker we can shove them through the processor, the more work it can get done.

Pipelines

As mentioned before, instructions are sent from the software and are broken down into *micro-ops* (smaller sub-operations) in the CPU. This decomposition and execution takes place in a *pipeline*.

The pipeline is like a reverse assembly line. The CPU's instructions are broken apart (decoded) at the start of the pipeline. They are converted into small sub-operations (*micro-ops*), which can then be processed one at a time in the rest of the pipeline:

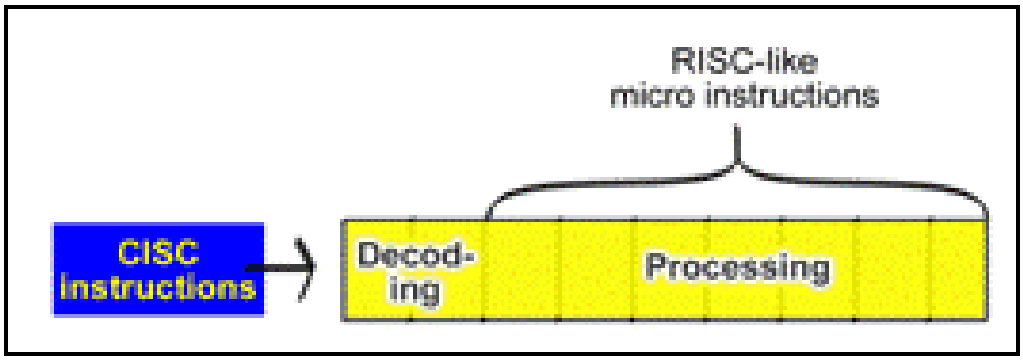


Fig. 85. First the CISC instructions are decoded and converted into more digestible micro instructions. Then these are processed. It all takes place in the pipeline.

The pipeline is made up of a number *stages*. Older processors have only a few stages, while the newer ones have many (from 10 to 31). At each stage "something" is done with the instruction, and each stage requires one clock tick from the processor.

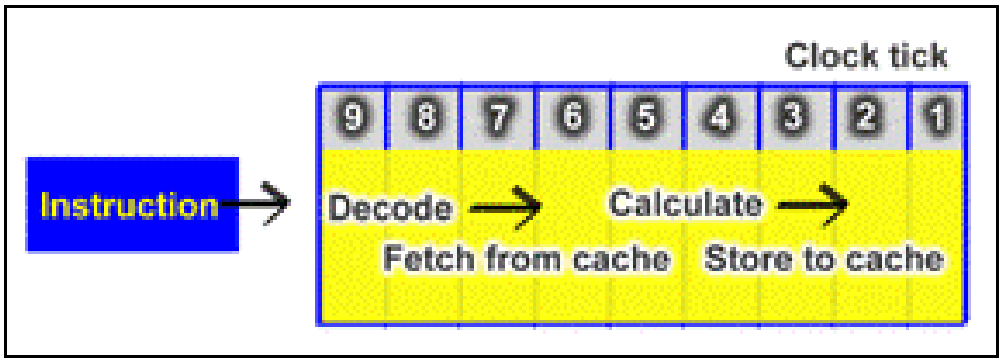


Fig. 86. The pipeline is an assembly line (shown here with 9 stages), where each clock tick leads to the execution of a sub-instruction.

Modern CPU's have more than one pipeline, and can thus process several instructions at the same time. For example, the Pentium 4 and AthlonXP can decode about 2.5 instructions per clock tick.

The first Pentium 4 has several very *long* pipelines, allowing the processor to hold up to *126 instructions* in total, which are all being processed *at the same time*, but at different stages of execution (see Fig. 88). It is thus possible to get the CPU to perform more work by letting several pipelines work in parallel:

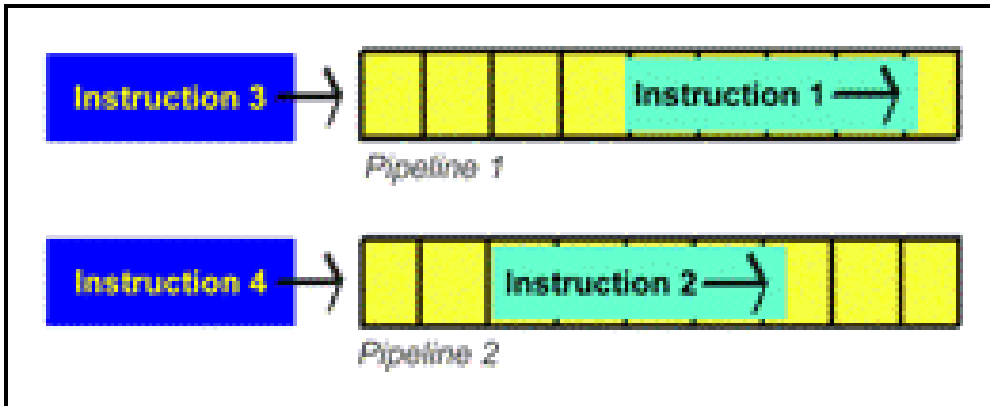


Fig. 87. Having two pipelines allows twice as many instructions to be executed within the same number of clock ticks.

CPU	Instructions executed at the same time
AMD K6-II	24
Intel Pentium III	40
AMD Athlon	72
Intel Pentium 4 (first generation)	126

Fig. 88. By making use of more, and longer, pipelines, processors can execute more instructions at the same time.

The problems of having more pipelines

One might imagine that the engineers at Intel and AMD could just make even more parallel pipelines in the one CPU. Perhaps performance could be doubled? Unfortunately it is not that easy.

It is not possible to feed a large number of pipelines with data. The memory system is just not powerful enough. Even with the existing pipelines, a fairly large number of clock ticks are wasted. The processor core is simply not utilised efficiently enough, because data cannot be brought to it quickly enough.

Another problem of having several pipelines arises when the processor can decode several instructions in parallel – each in its own pipeline. It is impossible to avoid the *wrong* instruction occasionally being read in (out of sequence). This is called *misprediction* and results in a number of wasted clock ticks, since another instruction has to be fetched and run through the “assembly line”.

Intel has tried to tackle this problem using a *Branch Prediction Unit*, which constantly attempts to guess the correct instruction sequence.

Length of the pipe

The number of “stations” (*stages*) in the pipeline varies from processor to processor. For example, in the Pentium II and III there are 10 stages, while there are up to 31 in the Pentium 4.

In the Athlon, the ALU pipelines have 10 stages, while the FPU/MMX/SSE pipelines have 15.

The longer the pipeline, the higher the processor’s clock frequency can be. This is because in the longer pipelines, the instructions are cut into more (and hence smaller) sub-instructions which can be executed more quickly.

CPU	Number of pipeline stages	Maximum clock frequency
Pentium	5	300 MHz

Motorola G4	4	500 MHz
Motorola G4e	7	1000 MHz
Pentium II and III	12	1400 MHz
Athlon XP	10/15	2500 MHz
Athlon 64	12/17	>3000 MHz
Pentium 4	20	>3000 MHz
Pentium 4 „Prescott“	31	>5000 MHz

Fig. 89. Higher clock frequencies require long “assembly lines” (pipelines).

Note that the two AMD processors have different pipeline lengths for integer and floating point instructions. One can also measure a processor’s efficiency by looking at the IPC number (*Instructions Per Clock*), and AMD’s Athlon XP is well ahead of the Pentium 4 in this regard. AMD’s Athlon XP processors are actually much faster than the Pentium 4’s at equivalent clock frequencies.

The same is even more true of the Motorola G4 processors used, for example, in Macintosh computers. The G4 only has a 4-stage pipeline, and can therefore, in principle, offer the same performance as a Pentium 4, with only half the clock frequency or less. The only problem is, the clock frequency can’t be raised very much with such a short pipeline. Intel have therefore chosen to future-proof the Pentium 4 by using a very long pipeline.

Execution units

What is it that actually happens in the pipeline? This is where we find the so-called execution units. And we must distinguish between to types of unit:

- ALU (*Arithmetic and Logic Unit*)
- FPU (*Floating Point Unit*)

If the processor has a brain, it is the ALU unit. It is the calculating device that does operations on whole numbers (*integers*). The computer’s work with ordinary *text*, for example, is looked after by the ALU.

The ALU is good at working with whole numbers. When it comes to decimal numbers and especially numbers with many decimal places (*real* numbers as they are called in mathematics), the ALU chokes, and can take a very long time to process the operations. That is why an FPU is used to relieve the load. An FPU is a number cruncher, specially designed for *floating point* operations.

There are typically several ALU’s and FPU’s in the same processor. The CPU also has other operation units, for example, the LSU (*Load/Store Unit*).

An example sequence

Look again at Fig. 73 on page 29. You can see that the processor core is right beside the L1 cache. Imagine that an instruction has to be processed:

- The processor core fetches a long and complex x86 instruction from the L1 instruction cache.
- The instruction is sent into the pipeline where it is broken down into smaller units.
- If it is an integer operation, it is sent to an ALU, while floating point operations are sent to an FPU.
- After processing the data is sent back to the L1 cache.

This description applies to the working cycle in, for example, the Pentium III and Athlon. As a diagram it might look like this:

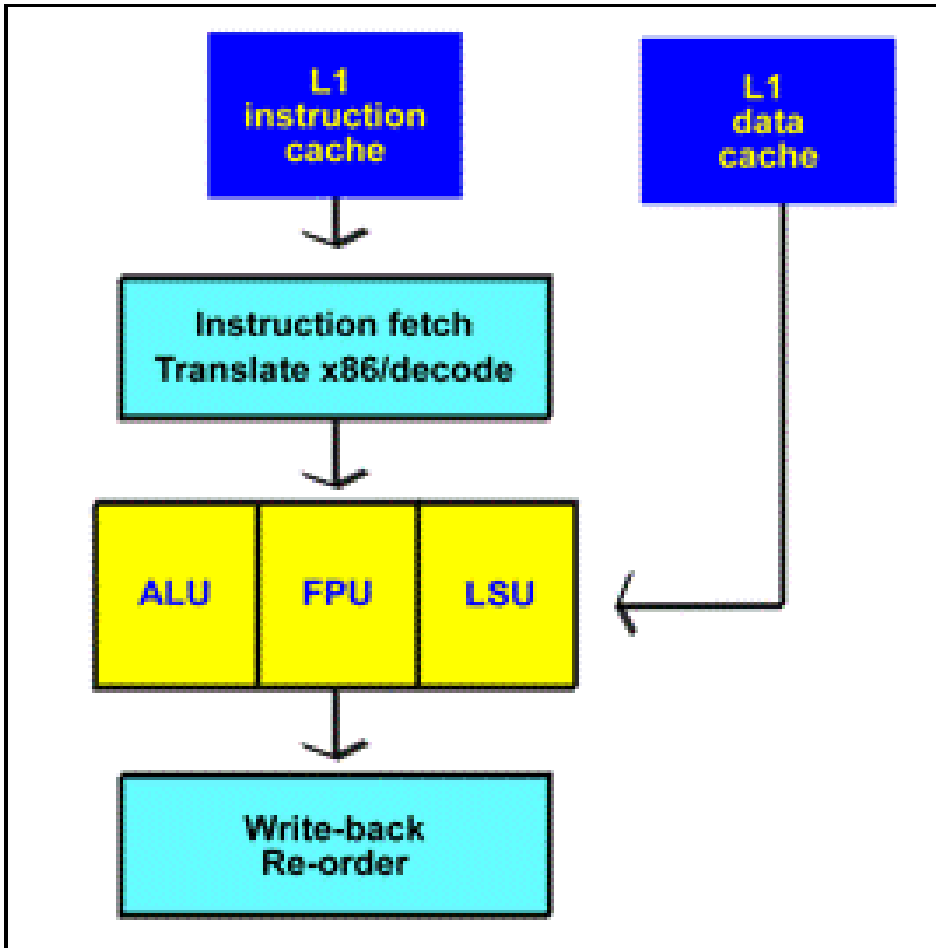


Fig. 90. The passage of instructions through the pipeline.

But the way the relationship between the pipeline and the execution units is designed differs greatly from processor to processor. So this entire examination should be taken as a general introduction and nothing more.

Pipelines in the Pentium 4

In the Pentium 4, the instruction cache has been placed between the “Instruction fetch/Translate” unit (in Fig. 90) and the ALU/FPU. Here the instruction cache (*Execution Trace Cache*) doesn’t store the actual instructions, but rather the “half-digested” micro-ops.

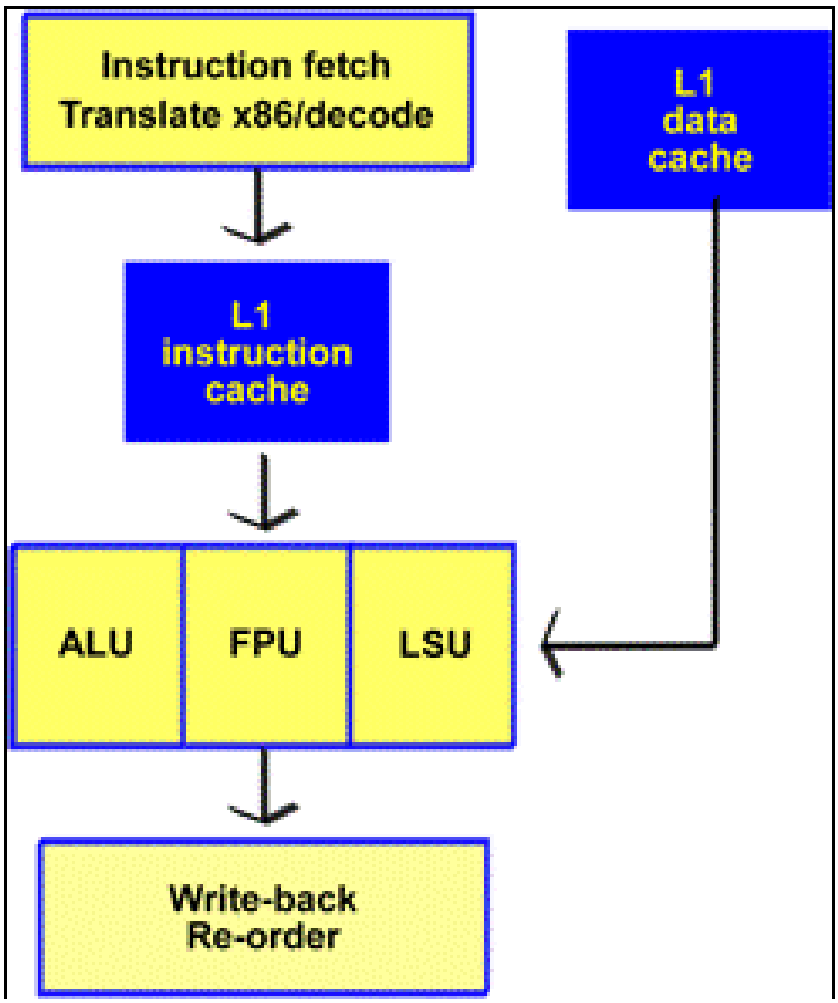


Fig. 91. In the Pentium 4, the instruction cache stores decoded micro instructions.

The actual pipeline in the Pentium 4 is longer than in other CPU's; it has 20 stages. The disadvantage of the long pipeline is that it takes more clock ticks to get an instruction through it. 20 stages require 20 clock ticks, and that reduces the CPU's efficiency. This was very clear when the Pentium 4 was released; all tests showed that it was much slower than other processors with the same clock frequency.

At the same time, the cost of reading the wrong instruction (*misprediction*) is much greater – it takes a lot of clock ticks to fill up the long assembly line again.

The Pentium 4's architecture must therefore be seen from a longer-term perspective. Intel expects to be able to scale up the design to work at clock frequencies of up to 5-10 GHz. In the "Prescott" version of Pentium 4, the pipeline was increased further to 31 stages.

AMD's 32 bit Athlon line can barely make it much above a clock frequency of 2 GHz, because of the short pipeline. In comparison, the Pentium 4 is almost "light years" ahead.

- Next chapter.
- Previous chapter.

- Next chapter.
- Previous chapter.

Chapter 13. FPU's and multimedia

The computer is constantly performing *calculations*, which can be divided into two groups.

- Whole numbers
- Floating point numbers

The whole number calculations are probably the most important, certainly for normal PC use – using office programs and the like. But operations involving floating point numbers have taken on greater significance in recent years, as 3D games and sound, image and video editing have become more and more a part of everyday computing. Let's have a brief look at this subject.

Floating point numbers

The CPU has to perform a lot of calculations on decimal (or *real*) numbers when the PC runs 3D games and other multimedia programs.

These decimal numbers are processed in the CPU by a special unit called an FPU (*Floating Point Unit*). In case you are wondering (as I did) about the name, *floating point* – here is an explanation: Real numbers (like the number π, pi) can have an infinite number of decimal places. In order to work with these, often very large, numbers, they are converted into a special format.

First the required level of *precision* is set. Since the numbers can have an infinite number of decimals, they have to be rounded. For example, one might choose to have *five* significant digits, as I have done in the examples below (inside the PC, one would probably choose to have many more significant digits).

Once the precision has been set, the numbers are converted as shown below (the decimal point *floats*):

- The number 1,257.45 is written as 0.12575×10^4.
- The number 0.00696784 is written as 0.69678×10^{-2}.

Now the FPU can manage the numbers and process them using the arithmetic operators.

Normal form	Rewritten	In the FPU
1,257.45	0.12575×10^4	12575 +4
0.00696784	0.69678×10^{-2}	69678 -2

Fig. 92. Re-writing numbers in floating point format.

FPU – the number cruncher

Floating point numbers are excessively difficult for the CPU's standard processing unit (the ALU) to process. A huge number of bits are required in order to perform a precise calculation. Calculations involving whole numbers (integers) are much simpler and the result is correct, every time.

That is why an FPU is used – a special calculating unit which operates with floating point numbers of various bit lengths, depending on how much precision is needed. FP numbers can be up to 80 bits long, whereas normal whole numbers can "only" be up to 32 bits (permitting 4,294 billion different numbers). So the FPU is a number cruncher, which relieves the load on the ALU's. You can experiment with large numbers yourself, for example, in a spreadsheet.

In Excel 2000, 2^{1023} (the number 2, multiplied by itself 1023 times), is the biggest calculation I can perform. The result is slightly less than 9 followed by 307 zeroes.

C4 = =A4^B4

	A	B	C
1		Potens	
2	2	32	4.294.967.296,00
3	2	1023	8,99E+307
4	2	1024	#NUM!
5			

Fig. 93. Experiments with big numbers in Excel.

Modern CPU's have a built-in FPU (*Floating Point Unit*) which serves as a number cruncher. But it hasn't always been like this.

For example, Intel's 80386 processor didn't have a built-in FPU calculating unit. All calculations were done using the processor's ALU. But you could buy a separate FPU (an *80387*), which was a chip which you mounted in a socket on the motherboard, beside the CPU. However, in the 80486 processor, the FPU was built-in, and it has been that way ever since.

Fig. 94. A “separate” FPU, Intel’s 80387 from 1986.

3D graphics

Much of the development in CPU’s has been driven by 3D games. These formidable games (like *Quake* and others) place incredible demands on CPU’s in terms of computing power. When these programs draw people and landscapes which can change in 3-dimensional space, the shapes are constructed from tiny *polygons* (normally triangles or rectangles).

Fig. 95. The images in popular games like Sims are constructed from hundreds of polygons.

A character in a PC game might be built using 1500 such polygons. Each time the picture changes, these polygons have to be drawn again in a new position. That means that every corner (*vortex*) of every polygon has to be re-calculated.

In order to calculate the positions of the polygons, floating point numbers have to be used (integer calculations are not nearly accurate enough). These numbers are called *single-precision floating points* and are 32 bits long. There are also 64-bit numbers, called *double-precision floating points*, which can be used for even more demanding calculations.

When the shapes in a 3D landscape move, a so-called matrix multiplication has to be done to calculate the new vortexes. For just one shape, made up of, say, 1000 polygons, up to 84,000 multiplications have to be performed on pairs of 32-bit floating point numbers. And this has to happen for each new position the shape has to occupy. There might be 75 new positions per second. This is quite heavy computation, which the traditional PC is not very good at. The national treasury’s biggest spreadsheet is child’s play compared to a game like Quake, in terms of the computing power required.

The CPU can be left gasping for breath when it has to work with 3D movements across the screen. What can we do to help it? There are several options:

- Generally faster CPUs. The higher the clock frequency, the faster the traditional FPU performance

will become.

- Improvements to the CPU's FPU, using more pipelines and other forms of acceleration. We see this in each new generation of CPU's.

- New *instructions* for more efficient 3D calculations.

We have seen that clock frequencies are constantly increasing in the new generations of CPU. But the FPU's themselves have also been greatly enhanced in the latest generations of CPU's. The Athlon, especially, is far more powerfully equipped in this area compared to its predecessors.

The last method has also been shown to be very effective. CPU's have simply been given new *registers* and new *instructions* which programmers can use.

MMX instructions

The first initiative was called MMX (*multimedia extension*), and came out with the Pentium MMX processor in 1997. The processor had built-in "MMX instructions" and "MMX registers".

The previous editions of the Pentium (like the other 32 bit processors) had two types of register: One for 32-bit integers, and one for 80-bit decimal numbers. With MMX we saw the introduction of a special 64-bit integer register which works in league with the new MMX instructions. The idea was (and is) that multimedia programs should exploit the MMX instructions. Programs have to be "written for" MMX, in order to utilise the new system.

MMX is an *extension* to the existing instruction set (*IA32*). There are 57 new instructions which MMX compatible processors understand, and which require new programs in order to be exploited.

Many programs were rewritten to work both with and without MMX (see Fig 96). Thus these programs could continue to run on older processors, without MMX, where they just ran slower.

MMX was a limited success. There is a weakness in the design in that programs either work with MMX, or with the FPU, and not both at the same time – as the two instruction sets share the same registers. But MMX laid the foundation for other multimedia extensions which have been much more effective.

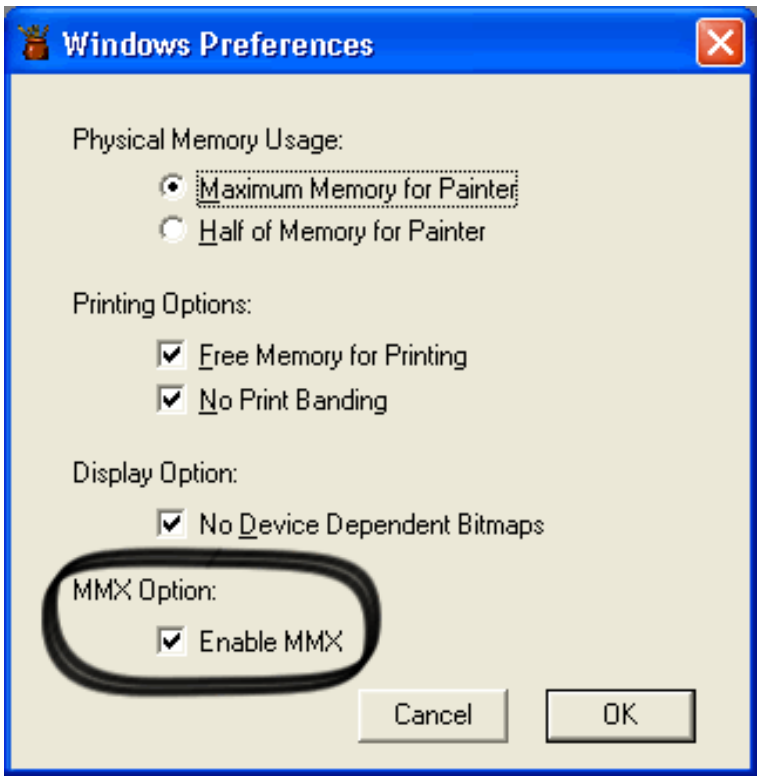


Fig 96. This drawing program (Painter) supports MMX, as do all modern programs.

3DNow!

In the summer of 1998, AMD introduced a collection of CPU instructions which improved 3D processing. These were 21 new SIMD (*Single Instruction Multiple Data*) instructions. The new instructions could process several chunks of data with one instruction. The new instructions were marketed under the name, *3DNow!*. They particularly improved the processing of the 32-bit floating point numbers used so extensively in 3D games.

Fig. 97. 3DNow! became the successor to MMX.

3DNow! was a big success. The instructions were quickly integrated into Windows, into various games (and other programs) and into hardware manufacturers' driver programs.

SSE

After AMD's success with 3DNow!, Intel had to come back with something else. Their answer, in January 1999, was SSE (Streaming SIMD Extensions), which are another way to improve 3D performance. SSE was introduced with the Pentium III.

In principle, SSE is significantly more powerful than 3DNow! The following changes were made in the CPU:

- 8 new 128-bit registers, which can contain four 32-bit numbers at a time.
- 50 new SIMD instructions which make it possible to do advanced calculations on several floating

point numbers with just one instruction.

- 12 *New Media Instructions*, designed, for example, for the encoding and decoding of MPEG-2 video streams (in DVD).

- 8 new *Streaming Memory* instructions to improve the interaction between L2 cache and RAM.

SSE also quickly became a success. Programs like Photoshop were released in new SSE optimised versions, and the results were convincing. Very processor-intensive programs involving sound, images and video, and in the whole area of multimedia, run much more smoothly when using SSE.

Since SSE was such a clear success, AMD took on board the technology. A large part of SSE was built into the AthlonXP and Duron processors. This was very good for software developers (and hence for us users), since all software can continue to be developed for one instruction set, common to both AMD and Intel.

SSE2 and SSE3

With the Pentium 4, SSE was extended to use even more powerful techniques. SSE2 contains 144 new instructions, including 128-bit SIMD integer operations and 128-bit SIMD *double-precision floating-point operations.*

SSE2 can reduce the number of instructions which have to be executed by the CPU in order to perform a certain task, and can thus increase the efficiency of the processor. Intel mentions *video, speech recognition, image/photo processing, encryption* and *financial/scientific programs* as the areas which will benefit greatly from SSE2. But as with MMX, 3DNow! and SSE, the programs have to be rewritten before the new instructions can be exploited.

SSE2 adopted by the competition, AMD, in the Athlon 64-processors. Here AMD even doubled up the number of SSE2 registers compared to the Pentium 4. Latest Intel has introduced 13 new instructions in SSE3, which Intel uses in the Prescott-version of Pentium 4.

We are now going to leave the discussion of instructions. I hope this examination has given you some insight into the CPU’s work of executing programs.

- Next chapter.
- Previous chapter.

- Next chapter.
- Previous chapter.

Chapter 14.Examples of CPU's

In this chapter I will briefly describe the important CPU's which have been on the market, starting from the PC's early childhood and up until today.

One could argue that the obsolete and discontinued models no longer have any practical significance. This is true to some extent; but the old processors form part of the "family tree", and there are still legacies from their architectures in our modern CPU's, because the development has been evolutionary. Each new processor extended and built "on top of" an existing architecture.

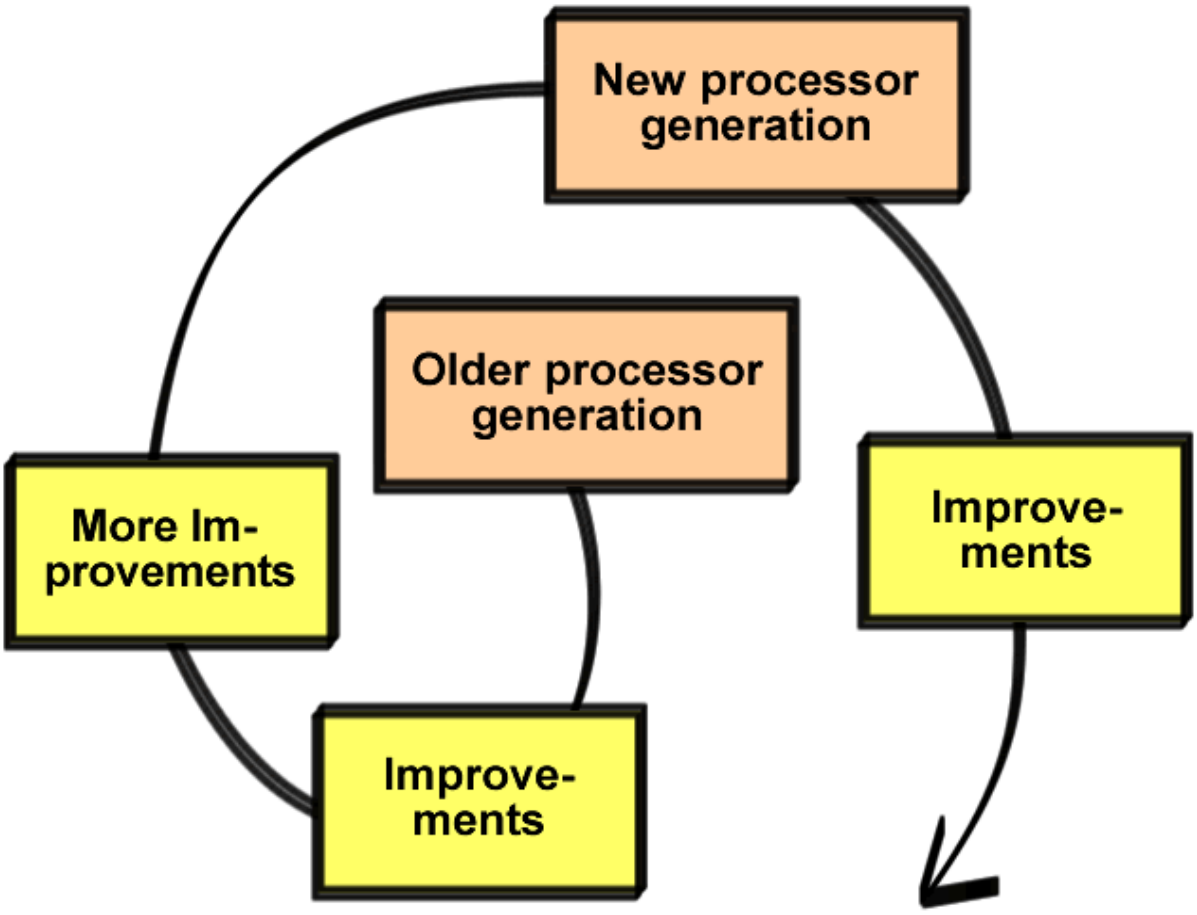


Fig. 98. The evolutionary development spirals ever outwards.

There is therefore value (one way or another) in knowing about the development from one generation of CPU's to the next. If nothing else, it may give us a feeling for what we can expect from the future.

16 bits – the 8086, 8088 and 80286

The first PC's were 16-bit machines. This meant that they could basically only work with text. They were tied to DOS, and could normally only manage one program at a time.

But the original 8086 processor was still "too good" to be used in standard office PC's. The Intel 8088 discount model was therefore introduced, in which the bus between the CPU and RAM was halved in width (to 8 bits), making production of the motherboard much cheaper. 8088 machines typically had 256 KB, 512 KB or 1 MB of RAM. But that was adequate for the programs at the time.

The Intel 80286 (from 1984) was the first step towards faster and more powerful CPU's. The 286 was much more efficient; it simply performed much more work per clock tick than the 8086/8088 did. A new feature was also the *32 bit protected mode* – a new way of working which made the processor much more efficient than under *real mode*, which the 8086/8088 processor forced programs to work in:

- Access to *all system memory* – even beyond the 1MB limit which applied to real mode.

Access to *multitasking*, which means that the operating system can run several programs at the same time.

- The possibility of *virtual memory*, which means that the hard disk can be used to emulate extra RAM, when necessary, via a *swap file*.

- 32 bit access to RAM and 32 bit drivers for I/O devices.

Protected mode paved the way for the change from DOS to Windows, which only came in the 1990's.

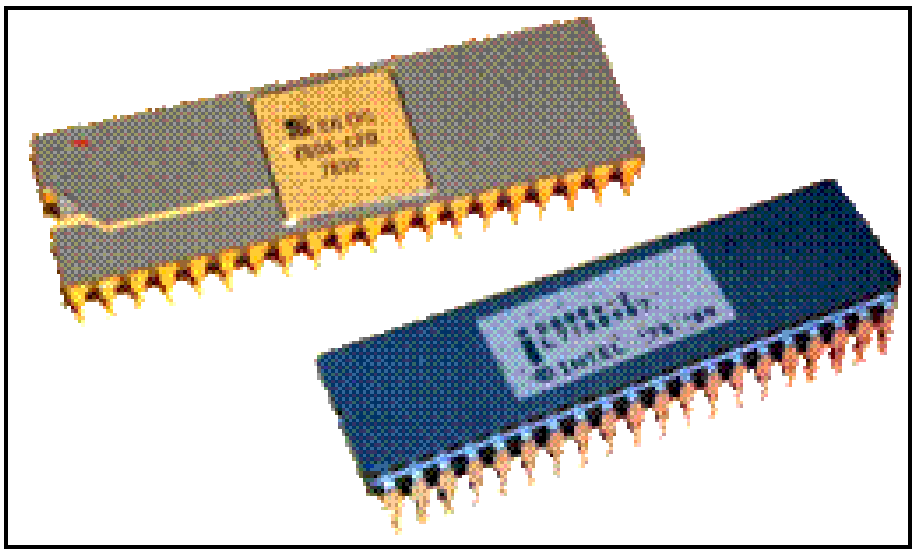


Fig. 99. Bottom: an Intel 8086, the first 16-bit processor. Top: the incredibly popular 8-bit processor, the Zilog Z80, which the 8086 and its successors out competed.

32 bits – the 80386 and 486

The Intel 80386 was the first 32-bit CPU. The 386 has 32-bit long registers and a 32-bit data bus, both internally and externally. But for a traditional DOS based PC, it didn't bring about any great revolution. A good 286 ran nearly as fast as the first 386's – under DOS anyway, since it doesn't exploit the 32-bit architecture.

The 80386SX became the most popular chip – a discount edition of the 386DX. The SX had a 16-bit external data bus (as opposed to the DX's 32-bit bus), and that made it possible to build cheap PC's.

Fig. 100. Discount prices in October 1990 – but only with a b/w monitor.

The fourth generation

The fourth generation of Intel's CPU's was called the 80486. It featured a better implementation of the x86 instructions – which executed faster, in a more RISC-like manner. The 486 was also the first CPU with built-in L1 cache. The result was that the 486 worked roughly twice as fast as its predecessor – for the same clock frequency.

With the 80486 we gained a built-in FPU. Then Intel did a marketing trick of the type we would be better off without. In order to be able to market a cheap edition of the 486, they hit on the idea of *disabling* the FPU function in some of the chips. These were then sold under the name, 80486SX. It was ridiculous – the processors had a built-in FPU; it had just been switched off in order to be able *to segment the market*.

Fig. 101. Two 486's from two different manufacturers.

But the 486 was a good processor, and it had a long life under DOS, Windows 3.11 and Windows 95. New editions were released with higher clock frequencies, as they hit on the idea of *doubling* the *internal* clock frequency in relation to the external (see the discussion later in the guide). These double-clocked processors were given the name, 80486DX2.

A very popular model in this series had an external clock frequency of 33 MHz (in relation to RAM), while working at 66MHz internally. This principle (*double-clocking*) has been employed in one way or

another in all later generations of CPU's. AMD, IBM, Texas Instruments and Cyrix also produced a number of 80486 compatible CPU's.

Pentium

In 1993 came the big change to a new architecture. Intel's Pentium was the first fifth-generation CPU. As with the earlier jumps to the next generation, the first versions weren't especially fast. This was particularly true of the very first Pentium 60 MHz, which ran on 5 volts. They got burning hot – people said you could fry an egg on them. But the Pentium quickly benefited from new process technology, and by using clock doubling, the clock frequencies soon skyrocketed.

Basically, the major innovation was a *superscalar* architecture. This meant that the Pentium could process several instructions at the same time (using several pipelines). At the same time, the RAM bus width was increased from 32 to 64 bits.

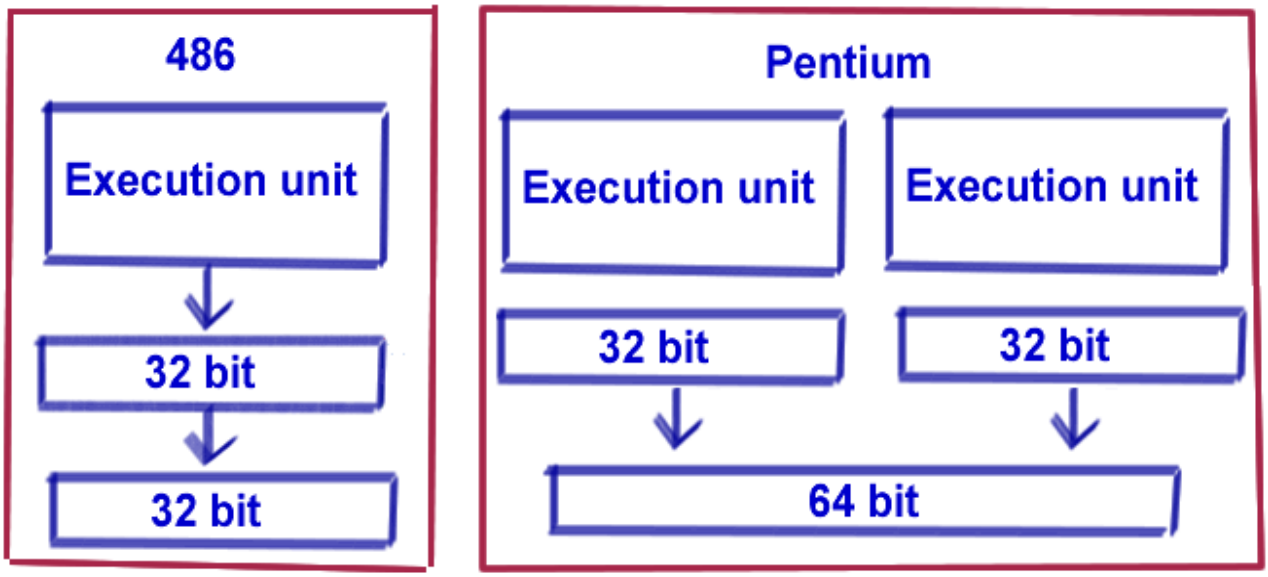
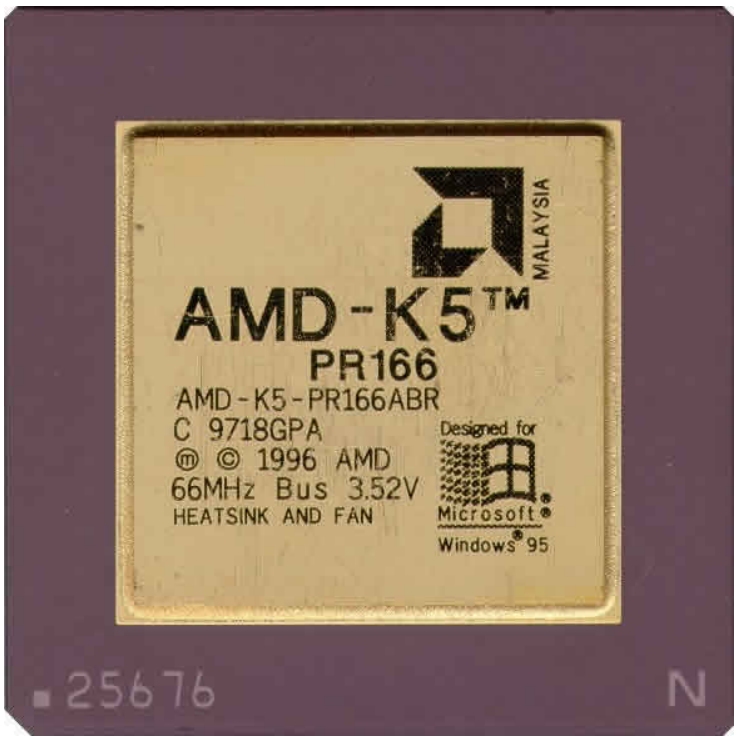


Fig. 102. The Pentium processor could be viewed as two 80486's built into one chip.

Throughout the 1990's, AMD gained attention with its *K5* and *K6* processors, which were basically cheap (and fairly poor) copies of the Pentium. It wasn't until the *K6-2* (which included the very successful 3DNow! extensions), that AMD showed the signs of independence which have since led to excellent processors like the AthlonXP.

Fig. 103. One of the earlier AMD processors. Today you'd hesitate to trust it to run a coffee machine…



In 1997, the Pentium MMX followed (with the model name P55), introducing the MMX instructions already mentioned. At the same time, the L1 cache was doubled and the clock frequency was raised.

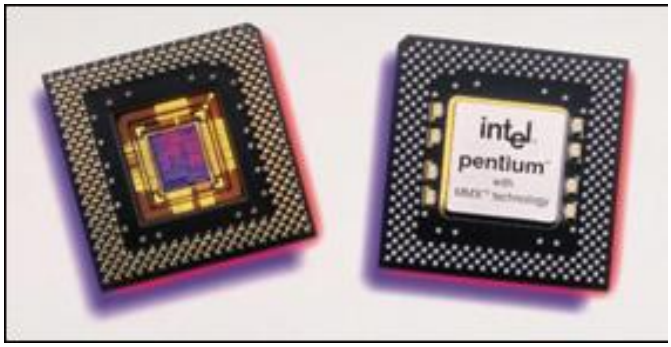


Fig. 104. The Pentium MMX. On the left, the die can be seen in the middle.

Pentium II with new cache

After the Pentium came the Pentium II. But Intel had already launched the Pentium Pro in 1995, which was the first CPU in the 6th generation. The Pentium Pro was primarily used in servers, but its architecture was re-used in the popular Pentium II, Celeron and Pentium III models, during 1997-2001.

The Pentium II initially represented a technological step backwards. The Pentium Pro used an integrated L2 cache. That was very advanced at the time, but Intel chose to place the cache outside the actual Pentium II chip, to make production cheaper.

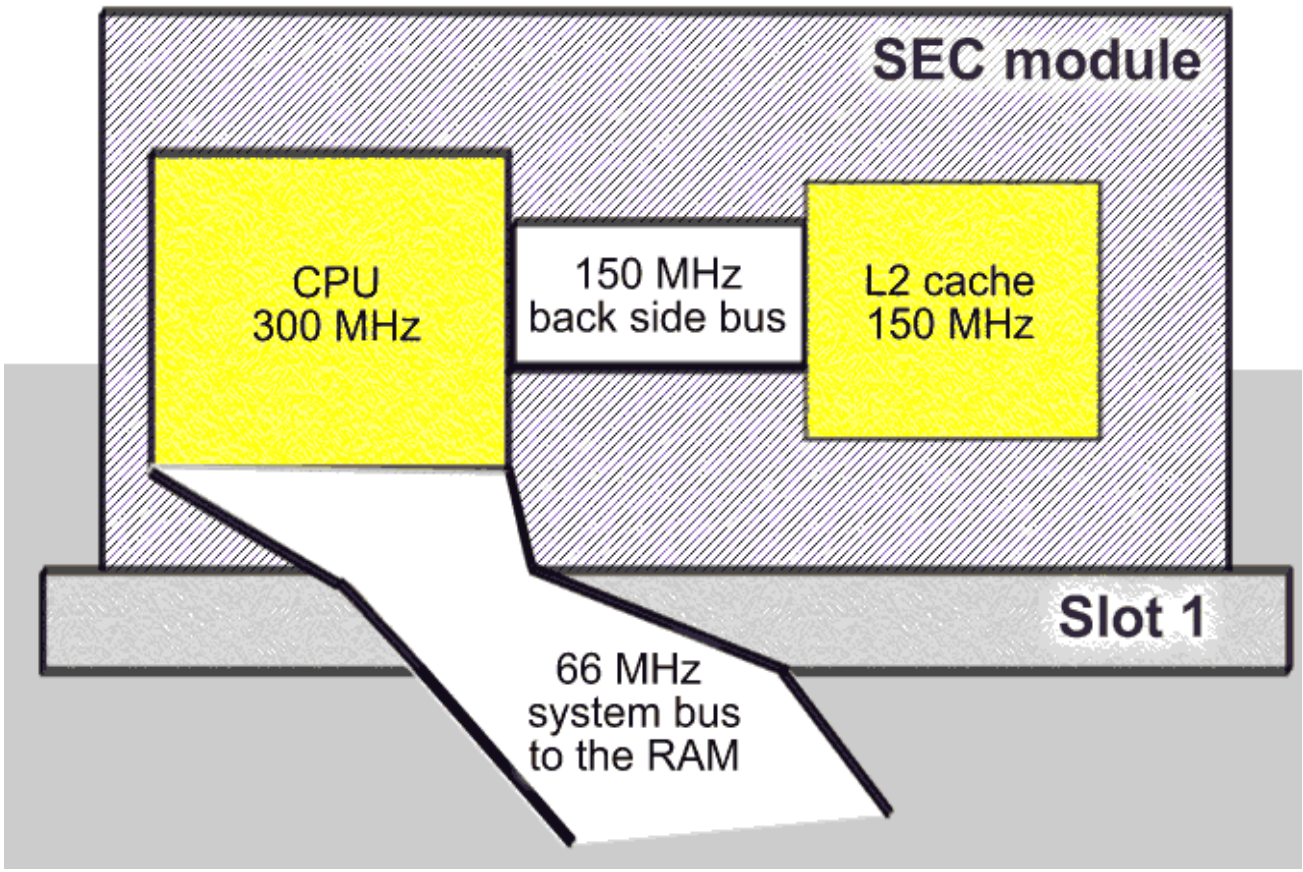


Fig. 105. L2 cache running at half CPU speed in the Pentium II.

The Level 2 cache was placed beside the CPU on a circuit board, an *SEC module* (e.g. see Fig. 71, on page 28). The module was installed in a long *Slot 1* socket on the motherboard. Fig. 106 shows the module with a cooling element attached. The CPU is sitting in the middle (under the fan). The L2 cache is in two chips, one on each side of the processor.

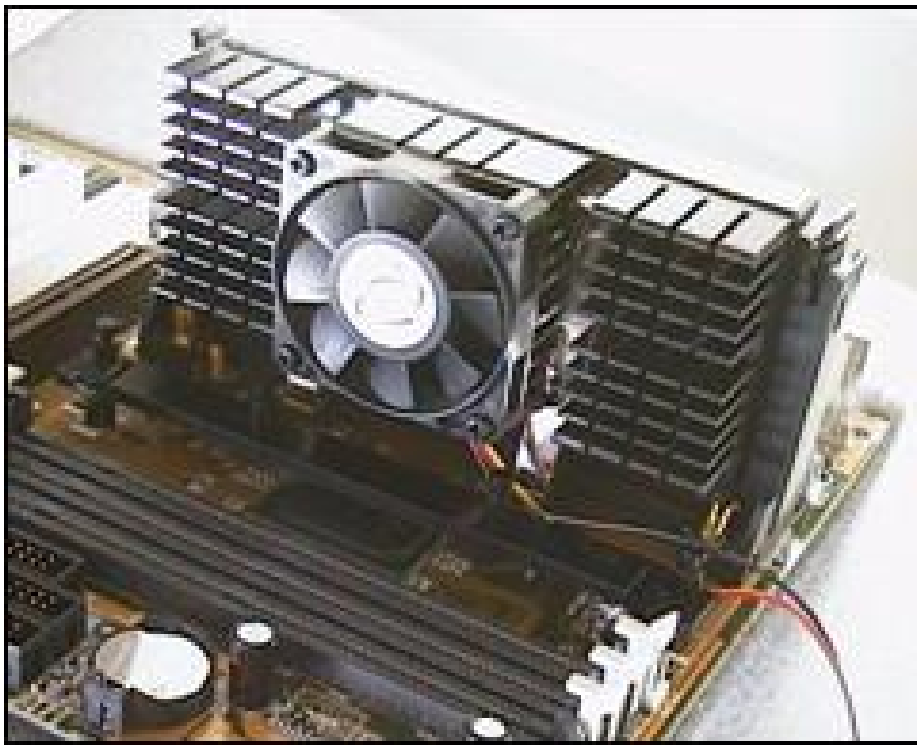


Fig. 106. Pentium II processor module mounted on its edge in the motherboard's Slot 1 socket (1997-1998).

The disadvantage of this system was that the L2 cache became markedly slower than it would have been if it was integrated into the CPU. The L2 cache typically ran at half the CPU's clock frequency. AMD used the same system in their first Athlons. For these the socket was called, *Slot A* (see Fig. 107).

At some point, Intel decided to launch a discount edition of the Pentium II – the Celeron processor. In the early versions, the L2 cache was simply scrapped from the module. That led to quite poor performance, but provided an opportunity for *overclocking*.

Overclocking means pushing a CPU to work at a higher frequency than it is designed to work at. It was a very popular sport, especially early on, and the results were good.

Fig. 107. One of the first AMD Athlon processors, mounted in a Slot A socket. See the large cooling element.

One of the problems of overclocking a Pentium II was that the cache chips couldn't keep up with the high speeds. Since these Celerons didn't have any L2 cache, they could be seriously overclocked (with the right cooling).

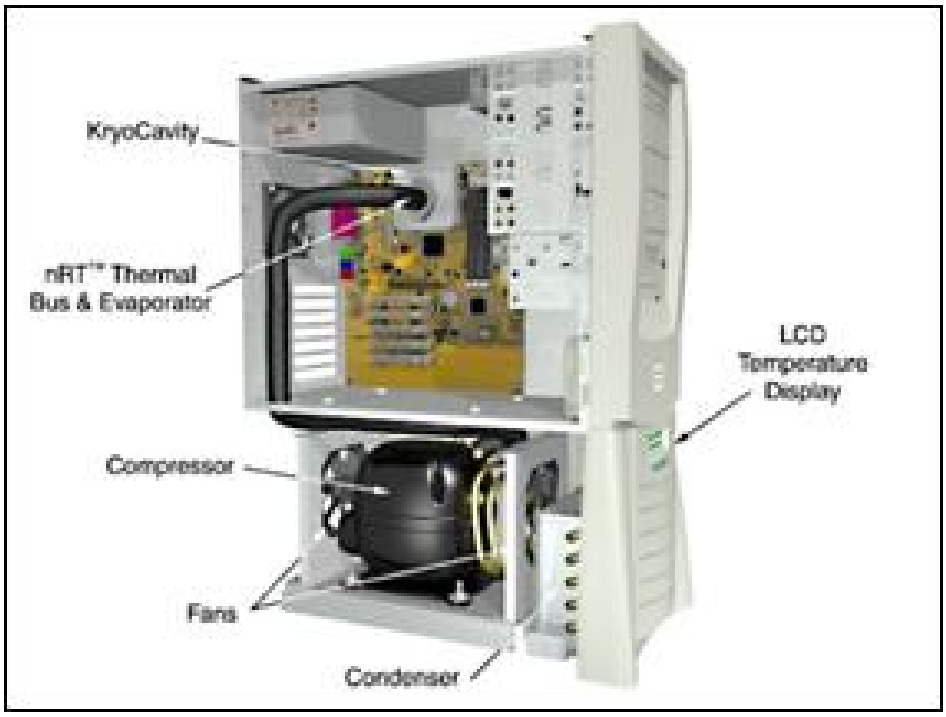


Fig. 108. Extreme CPU cooling using a complete refrigerator built into the PC cabinet. With equipment like this, CPU's can be pushed up to very high clock frequencies (See Kryotech.com and Asetek.com).

Intel later decided to integrate the L2 cache into the processor. That happened in a new versions of the Celeron in 1998 and a new versions of the Pentium III in 1999. The socket design was also changed so that the processors could be mounted directly on the motherboard, in a socket called *socket 370*. Similarly, AMD introduced their *socket A*.

Pentium 4 – long in the pipe

The Pentium III was really just (yet) another edition of the Pentium II, which again was a new version of the Pentium Pro. All three processors built upon the same core architecture (Intel *P6*).

It wasn't until the Pentium 4 came along that we got a completely new processor from Intel. The core (*P7*) had a completely different design:

- The L1 cache contained decoded instructions.
- The pipeline had been doubled to 20 stages (in later versions increased to 31 stages).
- The integer calculation units (ALU's) had been double-clocked so that they can perform two micro operations per clock tick.
- Furthermore, the memory bus, which connects the RAM to the north bridge, had been *quad-pumped*, so that it transfers four data packets per clock tick. That is equivalent to 4 x 100 MHz and 4 x 133 in the earliest versions of the Pentium 4. In later version the bus was pumped up to 4 x 200 MHz, and an update with 4 x 266 MHz is scheduled for 2005.
- The processor was Hyper Threading-enabled, meaning that it under certain circumstances may operate as two individual CPUs.

All of these factors are described elsewhere in the guide. The important thing to understand, is that the Pentium 4 represents a completely new processor architecture.

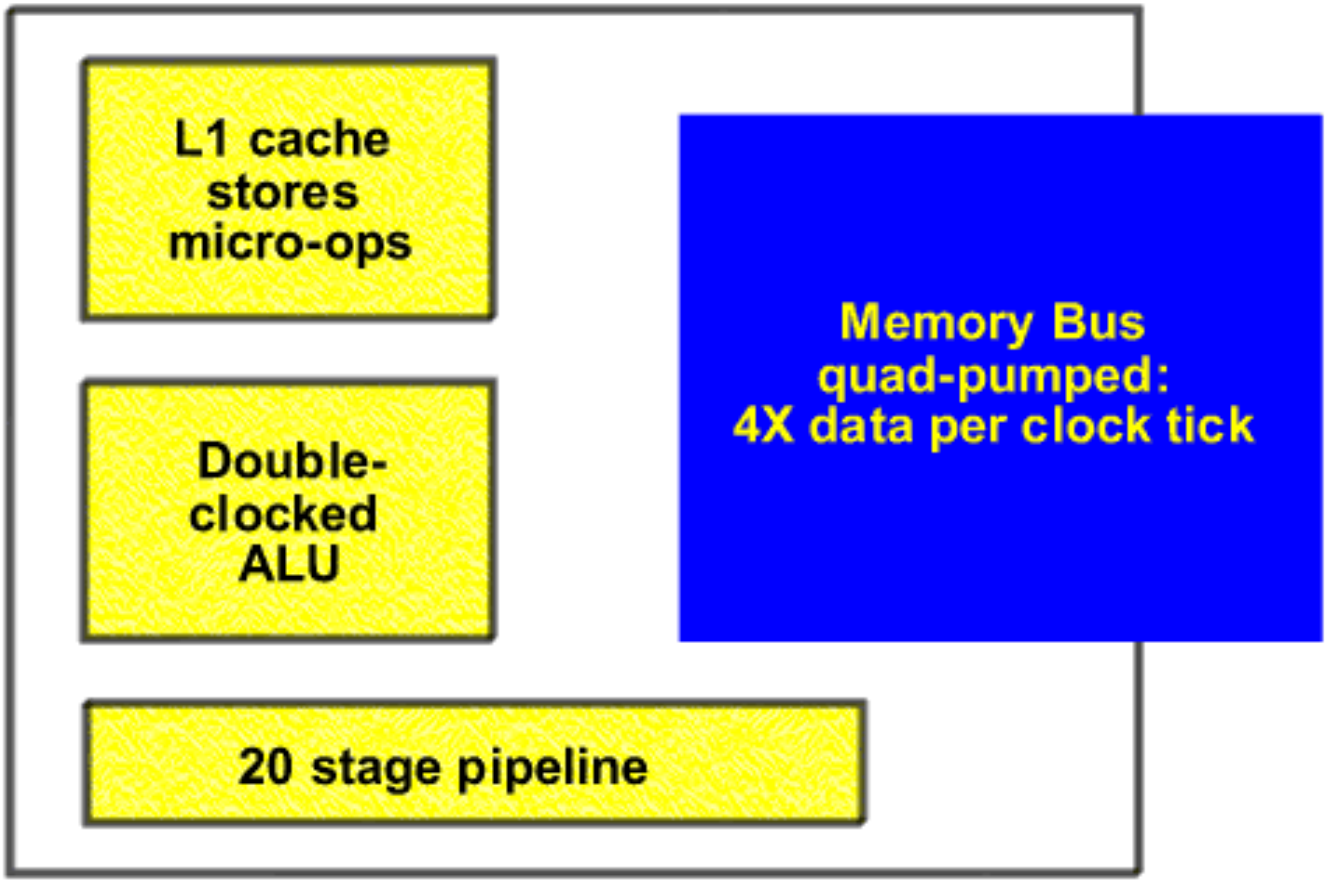


Fig. 109. The four big changes seen in the Pentium 4.

- Next chapter.
- Previous chapter.

- Next chapter.
- Previous chapter.

Chapter 15. Evolution of the Pentium 4

As was mentioned earlier, the older P6 architecture was released back in 1995. Up to 2002, the Pentium III processors were sold alongside the Pentium 4. That means, in practise, that Intel's sixth CPU generation has lasted 7 years.

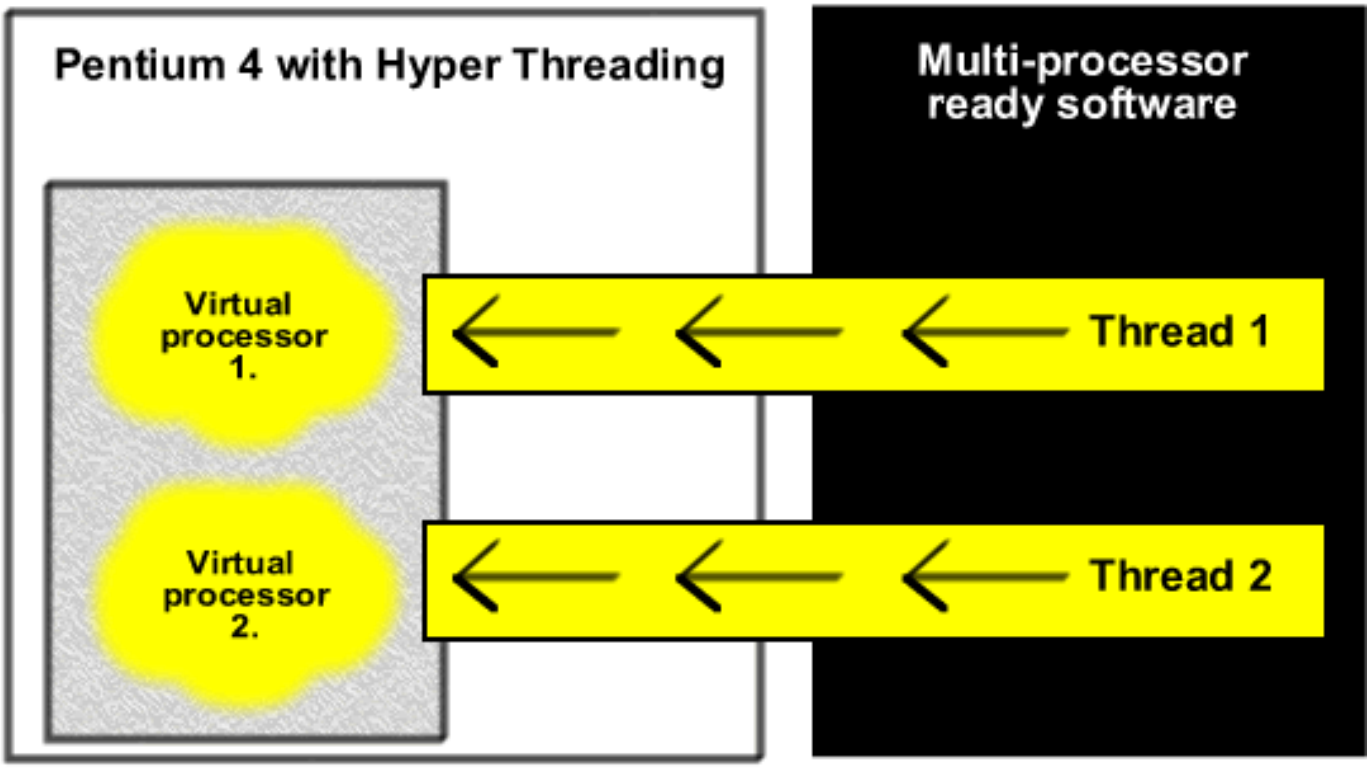
Similarly, we may expect this seventh generation Pentium 4 to dominate the market for a number of years. The processors may still be called Pentium 4, but it comes in al lot varietes.

A mayor modification comes with the version using 0.65 micron process technology. It will open for higher clock frequencies, but there will also be a number of other improvements.

Hyper-Threading Technology is a very exciting structure, which can be briefly outlined as follows: In order to exploit the powerful pipeline in the Pentium 4, it has been permitted to process *two threads at the same time*. Threads are series of software instructions. Normal processors can only process *one* thread at a time.

In servers, where several processors are installed in the same motherboard (MP systems), several threads can be processed at the same time. However, this requires that the programs be set up to exploit the MP system, as discussed on page 31.

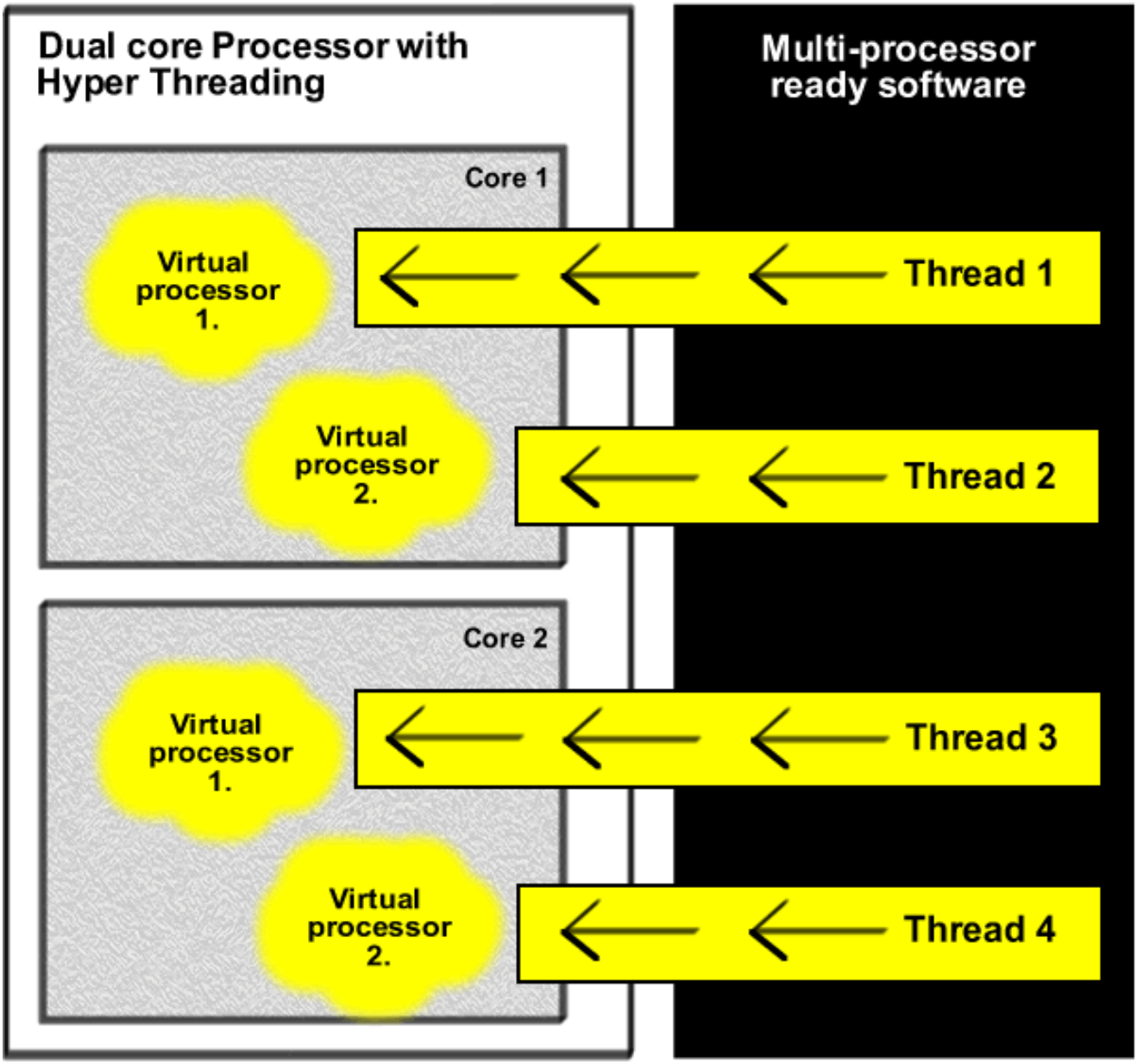
The new thing is that a single Pentium 4 logically can function as if there physically were two processors in the pc. The processor core (with its long pipelines) is simply so powerful that it can, in many cases, act as two processors. It's a bit like one person being able to carry on two independent telephone conversations at the same time.



Figur 110. The Pentium 4 is ready for MP functions.

Hyper-Threading works very well in Intel's Prescott-versions of Pentium 4. You gain performance when you operate more than one task at the time. If you have two programs working simultaneously, both putting heavy pressure on the CPU, you will benefit from this technology. But you need a MP-compatible operating system (like Windows XP Professional) to benefit from it.

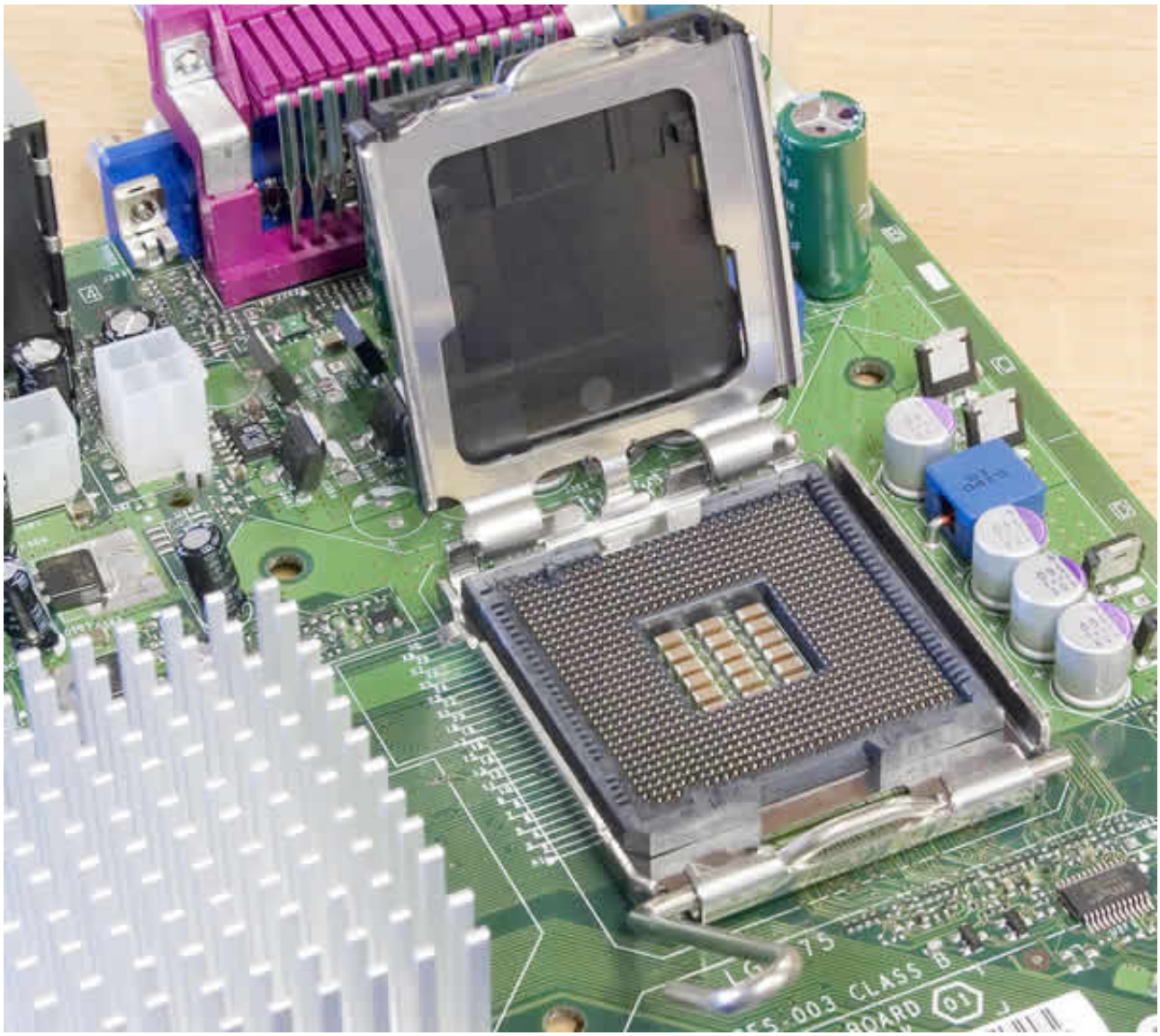
The next step in this evolution is the production of *dual-core processors.* AMD produces Opteron chips which hold two processors in one chip. Intel is working on dual core versions of the Pentium 4 (with the codename "Smithfield"). These chips will find use in servers and high performance pc's. A dual core Pentium 4 with Hyper-Threading enabled will in fact operate as a virtual *quad-core* processor.



Figur 111. A dual core processor with Hyper Threading operates as virtual quad-processor.

Intel also produces EE-versions of the Pentium 4. EE is for *Extreme Edition*, and these processors are extremely speedy versions carrying 2 MB of L2 cache.

In late 2004 Intel changed the socket design of the Pentium 4. The new processors have no "pins"; they connect directly to the socket using little contacts in the processor surface.



Figur 112. The LGA 775 socket for Pentium 4.

Athlon

The last processor I will discuss is the popular Athlon and Athlon 64 processor series (or *K7* and *K8*).

It was a big effort on the part of the relatively small manufacturer, AMD, when they challenged the giant Intel with a complete new processor design.

The first models were released in 1999, at a time when Intel was the completely dominant supplier of PC processors. AMD set their sights high – they wanted to make a better processor than the Pentium II, and yet cheaper at the same time. There was a fierce battle between AMD and Intel between 1999 and 2001, and one would have to say that AMD was the victor. They certainly took a large part of the market from Intel.

The original 1999 Athlon was very powerfully equipped with pipelines and computing units:

- Three instruction decoders which translated X86 program CISC instructions into the more efficient RISC instructions (ROP's) – 9 of which could be executed at the same time.

- Could handle up to 72 instructions (*ROP out of order*) at the same time (the Pentium III could manage 40, the K6-2 only 24).

- Very strong FPU performance, with three simultaneous instructions.

All in all, the Athlon was in a class above the Pentium II and III in those years. Since Athlon processors

were sold at competitive prices, they were incredibly successful. They also launched the *Duron* line of processors, as the counterpart to Intel's Celeron, and were just as successful with it.

Figur 113. Athlon was a huge success for AMD. During 2001-2002, the Athlon XP was in strong competition with the Pentium 4.

Athlon XP versus Pentium 4

The Athlon processor came in various versions. It started as a Slot A module (see Fig. 107 on page 42). It was then moved to Socket A, when the L2 cache was integrated.

In 2001, a new Athlon XP version was released, which included improvements like a new *Hardware Auto Data Prefetch Unit* and a bigger *Translation Look-aside Buffer*. The Athlon XP was much less advanced than the Pentium 4 but quite superior at clock frequencies less than 2000 MHz. A 1667 MHz version of AthlonXP was sold as 2000+. This indicates, that the processor as a minimum performs like a 2000 MHz Pentium 4.

Later we saw Athlons in other versions. The latest was based on a new kernel called "Barton". It was introduced in 2003 with a L2-cachen of 512 KB. AMD tried to sell the 2166 MHz version under the brand 3000+. It did not work. A Pentium 4 running at 3000 MHz had no problems outperforming the Athlon.

Opteron/ Athlon64

AMD's 8th generation CPU was released in 2003. It is based on a completely new core called *Hammer.*

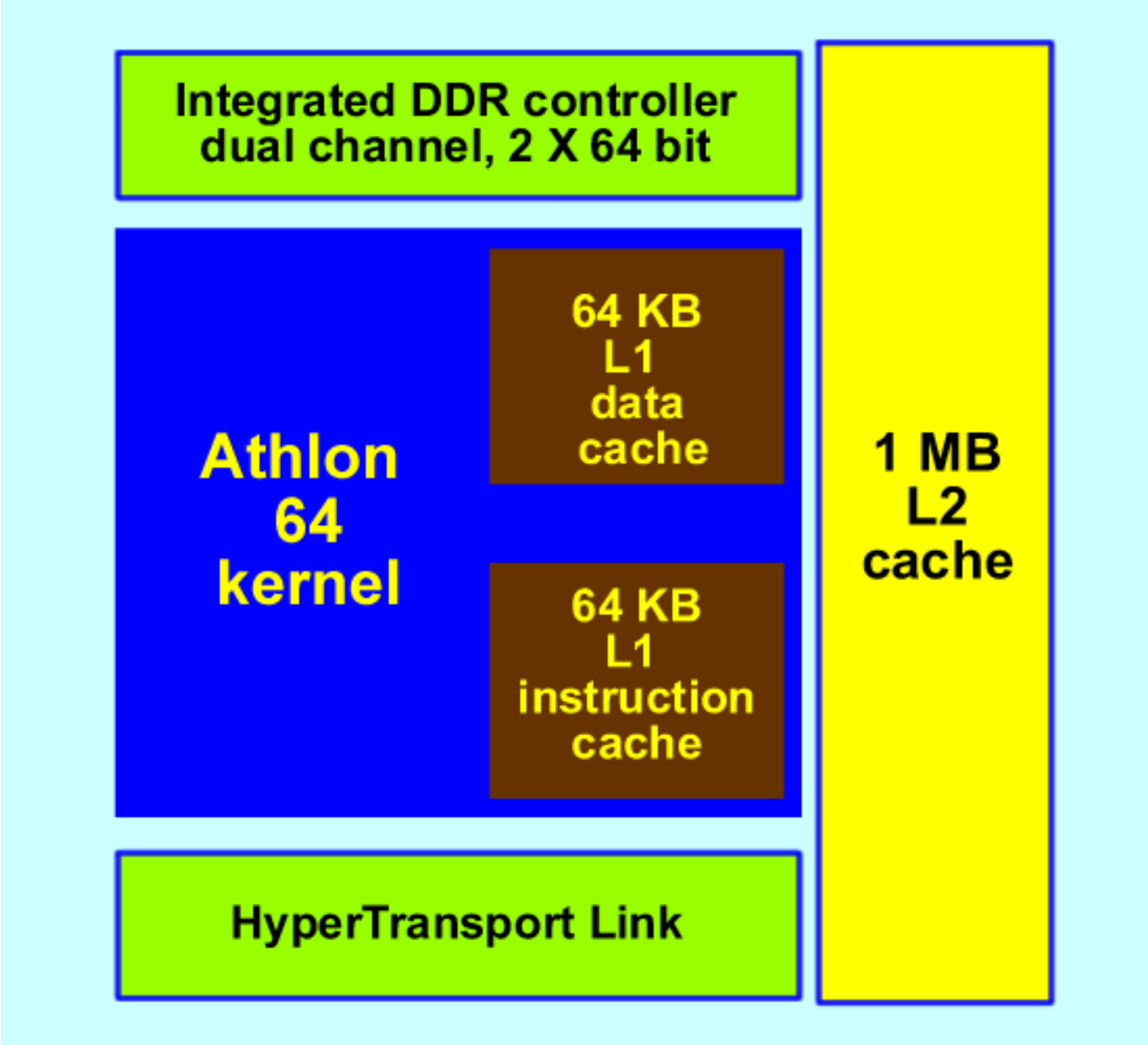
A new series of 64-bits processors is called Athlon 64, Athlon 64 FX and Opteron. These CPU's has a new design in two areas:

- The memory controller is integrated in the CPU. Traditionally this function has been housed in the north bridge, but now it is placed inside the processor.
- AMD introduces a completely new 64-bit set of instructions.

Moving the memory controller into the CPU is a great innovation. It gives a much more efficient communication between CPU and RAM (which has to be ECC DDR SDRAM – 72 bit modules with error correction).)

Every time the CPU has to fetch data from normal RAM, it has to first send a request to the chipset's controller. It has to then wait for the controller to fetch the desired data – and that can take a long time, resulting in wasted clock ticks and reduced CPU efficiency. By building the memory controller directly into the CPU, this waste is reduced. The CPU is given much more direct access to RAM. And that should reduce latency time and increase the effective bandwidth.

The Athlon 64 processors are designed for 64 bits applications. This should be more powerful than the existing 32 bit software. We will probably see plenty of new 64 bit software in the future, since Intel is releasing 64 bit processors compatible with the Athlon 64 series.



Figur 114. In the Athlon 64 the memory controller is located inside the processor. Hence, the RAM modules are interfacing directly with the CPU.

Overall the Athlon 64 is an updated Athlon-processor with integrated north bridge and 64 bits instructions. Other news are:

- Support for SSE2 instructions and 16 registers for this.
- Dual channel interface to DDR RAM giving a 128 bit memory bus, although the discount version Athlon 64 keeps the 64 bit bus.
- Communikationen to and from the south bridge via a new HyperTransport bus, operating with high-speed serial transfer.
- New sockets of 754 and 940 pins.

A complete line of chips

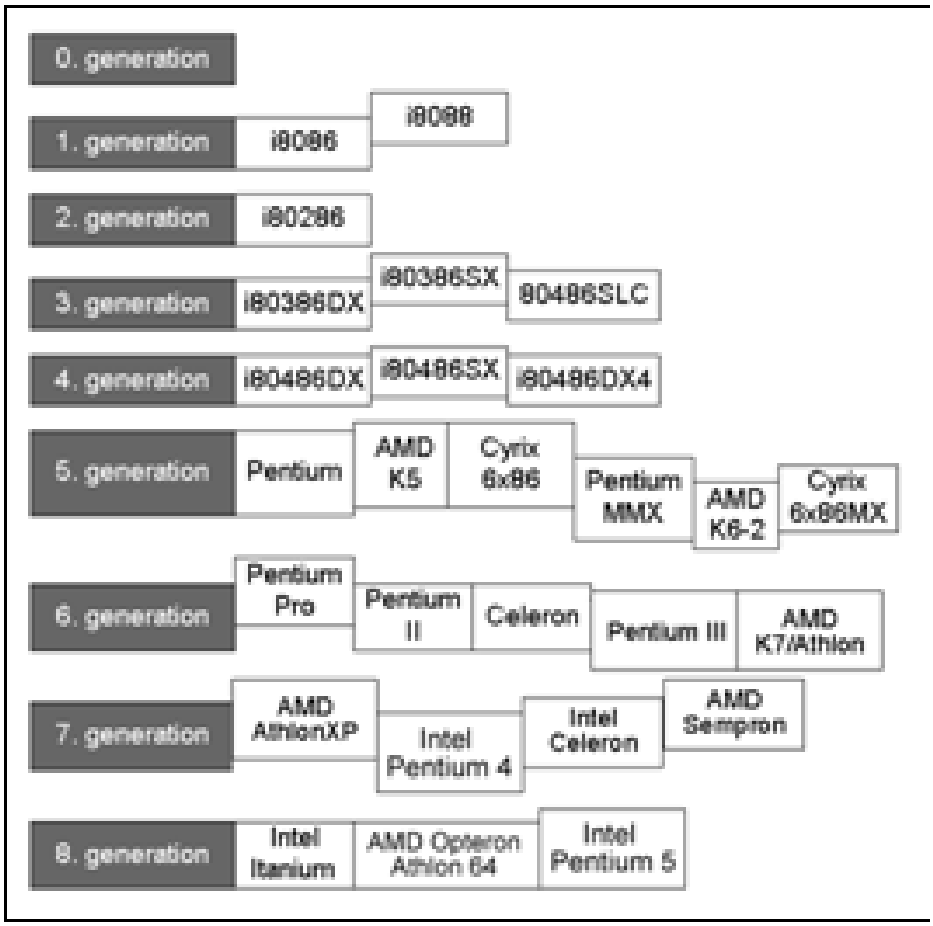
AMD expects to use the K8 kernel in all types of processors:

	The Opteron is the most expensive and advanced version to be used in multi-processor servers. The models are called 200, 400 and 800, and they use 2, 4 or 8 CPUs on the same motherboard – without use of a north bridge. All processors share a common memory of up to 64 GB. Each Opteron has three Hyper-Transport I/O channels, which each can move 6,4 GB/secund.
	The Athlon FX is a Opteron to be used in single processor configurations, high-end pc's and workstations. There is dual RAM interface, but only one channel of Hyper Transport Link.
	This is the discount version with reduced performance and lower prices. Only 64 bit RAM interface and smaller L2-cache.

Figur 115. Three versions of the latest AMD processor.

Historical overview

I will close off this review with a graphical summary of a number of different CPU's from the last 25 years. The division into generations is not always crystal clear, but I have tried to present things in a straightforward and reasonably accurate way:



Figur 116. There are scores of different processors. A selection of them is shown here, divided into generations.

But what is the most powerful CPU in the world? IBM's Power4 must be a strong contender. It is a monster made up of 8 integrated 64-bit processor cores. It has to be installed in a 5,200 pin socket, uses 500 watts of power (there are 680 million transistors), and connects to a 32 MB L3 cache, which it controls itself. Good night to Pentium.

- Next chapter.
- Previous chapter.

- Next chapter.
- Previous chapter.

Chapter 16. Choosing a CPU

If you happen to need to choose a CPU for your new PC, what should you choose? Let me give you a bit of food for thought.

	Processor	Euro
	INTEL Celeron 2.66 GHZ/533 256KB	70,00
	INTEL P4 520 2.8 GHZ/800 1 MB	130,00
	INTEL P4 530 3.0 GHZ/800 1 MB	145,00
	INTEL P4 540 3.2 GHZ/800 1 MB	175,00
	INTEL P4 550 3.2 GHZ/800 1 MB	220,00
	INTEL P4 560 3.2 GHZ/800 1 MB	330,00
	AMD Sempron 3100+ (1,8 GHZ)	105,00
	AMD ATHLON 64 3000+ (2,0 GHz)	209,00
	AMD ATHLON 64 3400+ (2,4 GHz)	227,00
	AMD ATHLON 64 FX-53 (2,4 GHz)	670,00

Figur. 117. Pricelist from October 2004 (without VAT).

How long does a CPU last?

The individual components have different lifetimes. The way development has gone up until the

present, CPU's and motherboards have been the components that have become *obsolete* the most quickly. CPU's and motherboards also go together -– you normally change them both at the same time.

You have by now read all of my review of new processor technology; new and faster models are constantly being developed. The question is, then, how important is it to have the latest technology? You have to decide that for yourself. But if you know that your PC has to last for many years, you probably should go for the fastest CPU on the market.

For the rest of us, who regularly upgrade and replace our PC's insides, it is important to find the most economic processor. There is usually a *price jump* in the processor series, such that the very latest models are disproportionately expensive.

By the time you read this, prices will have fallen in relation to those shown in Fig. 117. There will no doubt also be significantly faster CPU's to choose between. But the trend will most likely be the same: The latest models are a lot more expensive than the other ones. You have to find the model that gives the most power in proportion to the price.

Also, the amount of RAM is just as important as the speed of the processor. RAM prices fluctuate unbelievably, in just a year the price can double or halve. So it's a good idea to buy your RAM when the prices are low.

CPU-intensive operations

Back in the 1990's it was quite important to get the latest and fastest CPU, because CPU's were not fast enough for things like image processing. For example, if you try to work with fairly large images in Photoshop, on a PC with a 233 MHz processor, you will probably quickly decide to give up the project.

But whether you have 2400 or 3200 MHz – that's not as critical, especially if you have enough RAM and are working with normal tasks. A processor running at 3200 MHz is roughly 33% faster than a 2400 MHz model, but it doesn't always make that much difference to your daily work.

Here are some tasks, which might require more CPU power:

- Video editing.
- DVD ripping (often illegal).
- Photo editing, especially with high resolution images and 48 bit color depth.
- Production of PDF-files in high quality.
- Speech recognition.

Video (including DVD) contains huge amounts of data. The CPU and RAM therefore have to work very hard when you edit video footage. At the time of writing, it is legal to copy DVD films for personal use, but that may change. Legal or not – it's certainly possible. The actual *ripping* can take 10-20 minutes, during which the CPU slowly chews through the over 4 GB of data there are on a DVD. A 5 GHz CPU would obviously be able to reduce the time taken considerably.

Finally, *speech recognition* software is considered to be very CPU-intensive. This is probably only going to be a problem in the distant future. Some people imagine future versions of Windows and Office programs will have built-in speech recognition, which will place quite new demands on CPU's. However, it is far from certain that speech recognition will ever be introduced into PC's. There appear to be big problems in getting it to work well.

- Next chapter.
- Previous chapter.

Chapter 17. The CPU's immediate surroundings

In this part of this guide, we dug down into the inner workings of the CPU. We well let it rest in peace now, and concentrate on the processor's immediate surroundings. That is, the RAM and the chipset – or more precisely, the north bridge.

In the first section of the guide I introduced the chipset, including the *north bridge* (see, for example, Fig. 46 on page 19), which connects the CPU to the PC's memory — the RAM.

The pathway to RAM

The most important data path on the motherboard runs between the CPU and the RAM. Data is constantly pumped back and forth between the two, and this bus therefore often comes under focus when new generations of CPU's, chipset's and motherboards are released.

The RAM sends and receives data on a *bus*, and this work involves a *clock frequency*. This means that all RAM has a *speed*, just like a CPU does. Unfortunately RAM is much slower than the CPU, and the buses on the motherboard have to make allowance for this fact.

The XT architecture

In the original PC design (the IBM XT), the CPU, RAM and I/O devices (which we will come to later) were connected on one and the same bus, and everything ran *synchronously* (at a common speed). The CPU decided which clock frequency the other devices had to work at:

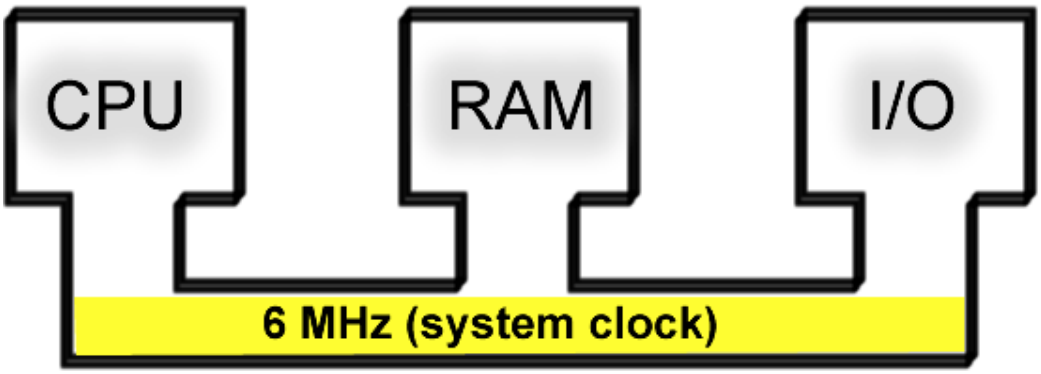


Fig. 118. In the original PC architecture, there was only one bus with one speed.

The problem with this system was that the three devices were "locked to each other"; they were forced to work at the lowest common clock frequency. It was a natural architecture in the first PC's, where the speed was very slow.

The first division of the bus

In 1987, Compaq hit on the idea of separating the system bus from the I/O bus, so that the two buses could work at different clock frequencies. By letting the CPU and RAM work on their own bus, independent of the I/O devices, their speeds could be increased.

In Fig. 119, the CPU and RAM are connected to a common bus, called the *system bus*, where in reality the CPU's clock frequency determines the working speed. Thus the RAM has the same speed as the CPU; for example, 12, 16 or 25 MHz.

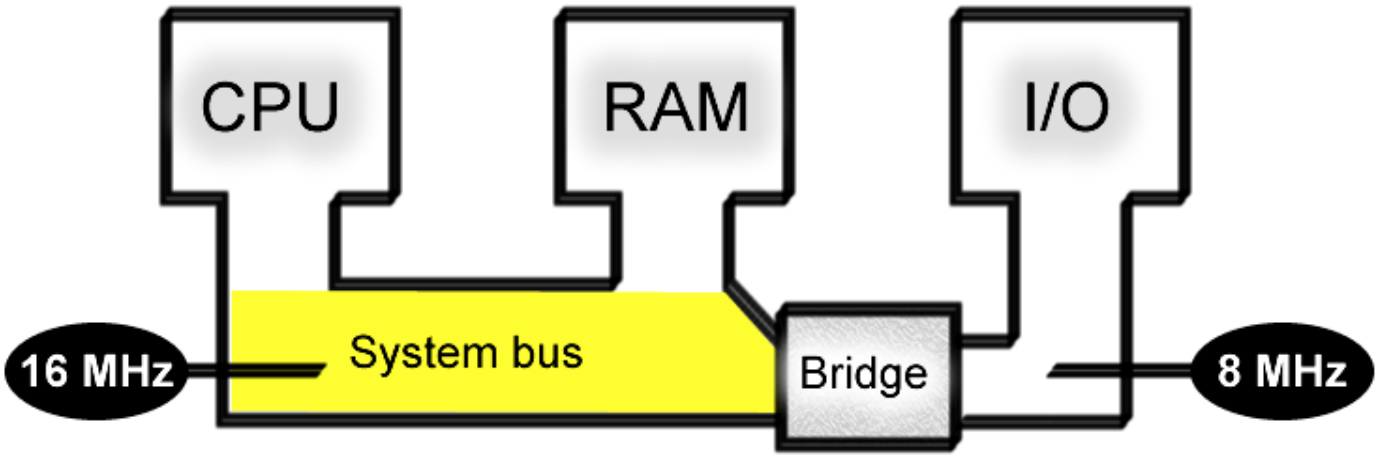


Fig. 119. With this architecture, the I/O bus is separate from the system bus (80386).

The I/O devices (graphics card, hard disk, etc.) were separated from the system bus and placed on a separate low speed bus. This was because they couldn't keep up with the clock frequencies of the new CPU versions.

The connection between the two buses is managed by a controller, which functions as a "bridge" between the two paths. This was the forerunner of the multibus architecture which all motherboards use today.

Clock doubling

With the introduction of the 80486, the CPU clock frequency could be increased so much that the RAM could no longer keep up. Intel therefore began to use *clock doubling* in the 80486 processor.

The RAM available at the time couldn't keep up with the 66 MHz speed at which an 80486 could work. The solution was to give the CPU two working speeds.

- An *external* clock frequency
- An *internal* clock frequency

Inside the processor, the clock frequency of the system bus is multiplied by a factor of 2, doubling the working speed.

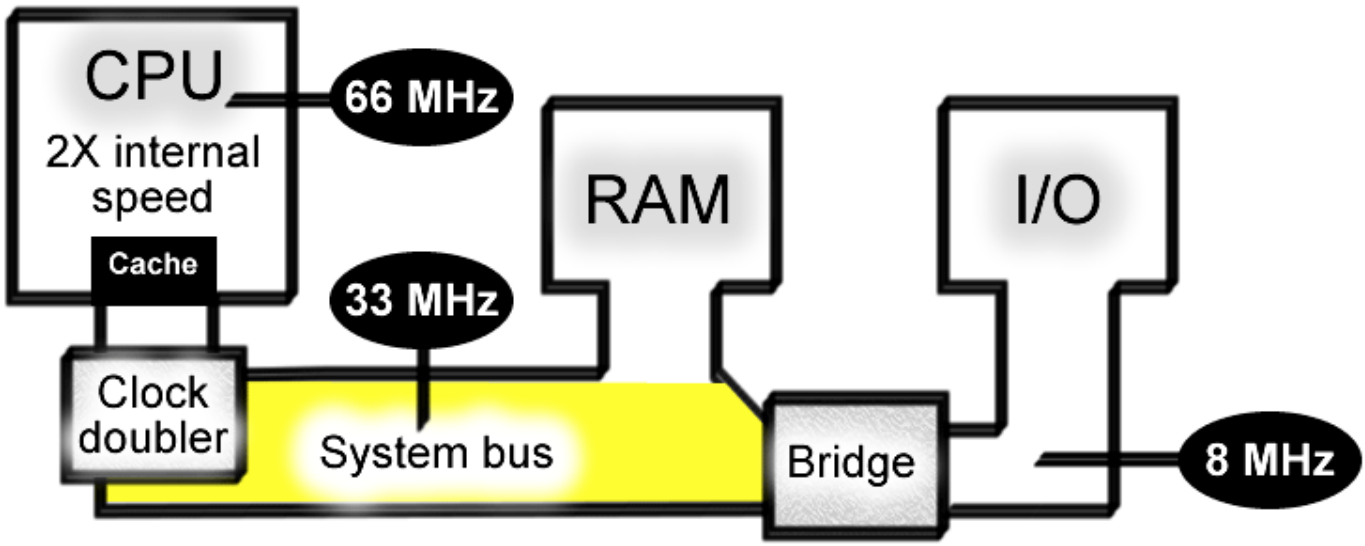
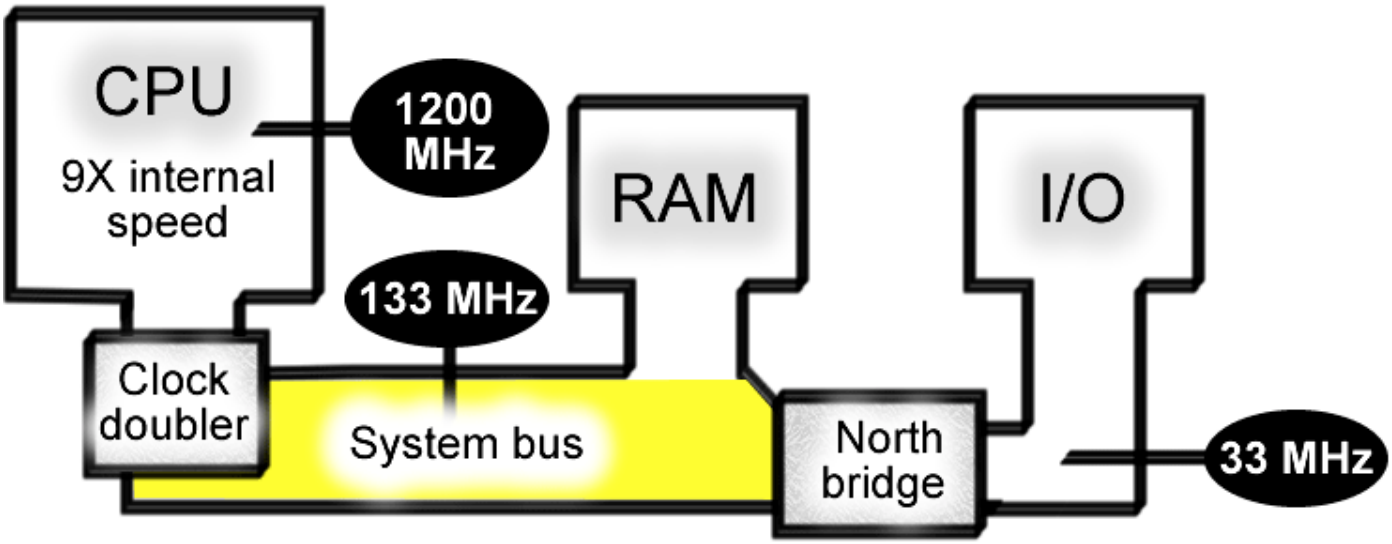


Fig. 120. The bus system for an 80486 processor.

But this system places heavy demands on the RAM, because when the CPU internally processes twice as much data, it of course has to be "fed" more often. The problem is, that the RAM only works half as fast as the CPU.

For precisely this reason, the 486 was given a built-in L1 cache, to reduce the imbalance between the slow RAM and the fast processor. The cache doesn't improve the bandwidth (the RAM doesn't work any faster), but it ensures greater efficiency in the transfer of data to the CPU, so that it gets the right data supplied at the right time.

Clock doubling made it possible for Intel to develop processors with higher and higher clock frequencies. At the time the Pentium was introduced, new RAM modules became available, and the system bus was increased to 66 MHz. In the case of the Pentium II and III, the system bus was increased to 100 and 133 MHz, with the internal clock frequency set to a multiple of these.



Figur 121. The bus system for a Pentium III processor.

- Next chapter.
- Previous chapter.

- Next chapter.
- Previous chapter.

Chapter 18. Overclocking

The Pentium II was subjected to a lot of *overclocking*. It was found that many of Intel's CPU's could be clocked at a higher factor than they were designed for.

If you had a 233 MHz Pentium II, you could set up the motherboard to, for example, run at 4.5 x 66 MHz, so that the processor ran at 300 MHz. I tried it myself for a while, it worked well. At a factor of 5 it didn't work, but at factor of 4.5 it functioned superbly.

CPU

System bus	Clock factor	Internal clock frequency	
Pentium	66 MHz	1.5	100 MHz
Pentium MMX	66 MHz	2.5	166 MHz
Pentium II	66 MHz	4.5	300 MHz
Pentium II	100 MHz	6	600 MHz
Celeron	100 MHz	8	800 MHz
Pentium III	133 MHz	9	1200 MHz
AthlonXP	133 MHz x 2	13	1733 MHz
AthlonXP+	166 MHz x 2	13	2167 MHz
Pentium 4	100 MHz x 4	22	2200 MHz
Pentium 4	133 MHz x 4	23	3066 MHz
Pentium 4	200 MHz x 4	18	3600 MHz

Figur 122. The CPU's internal clock frequency is locked to the system bus frequency.

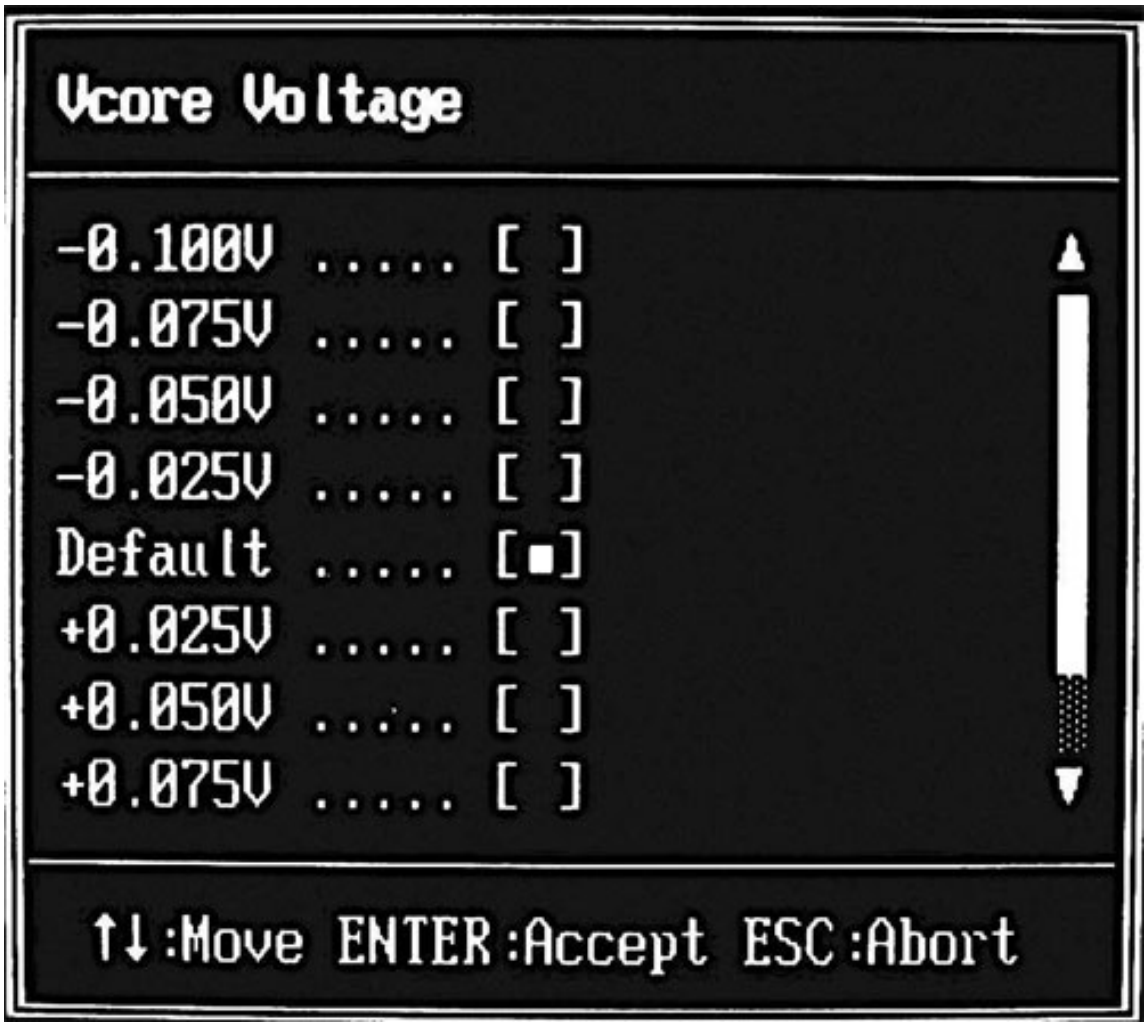
Overclocking the system bus

Another method of overclocking was to turn up the system bus clock frequency. In the early versions of the Pentium II, the system bus was at 66 MHz, which suited the type of RAM used at that time.

You could increase the bus speed, for example to 68 or 75 MHz, depending on how fast your RAM was. This type of tuning makes both the CPU and RAM faster, since it is the actual system clock speed which is increased.

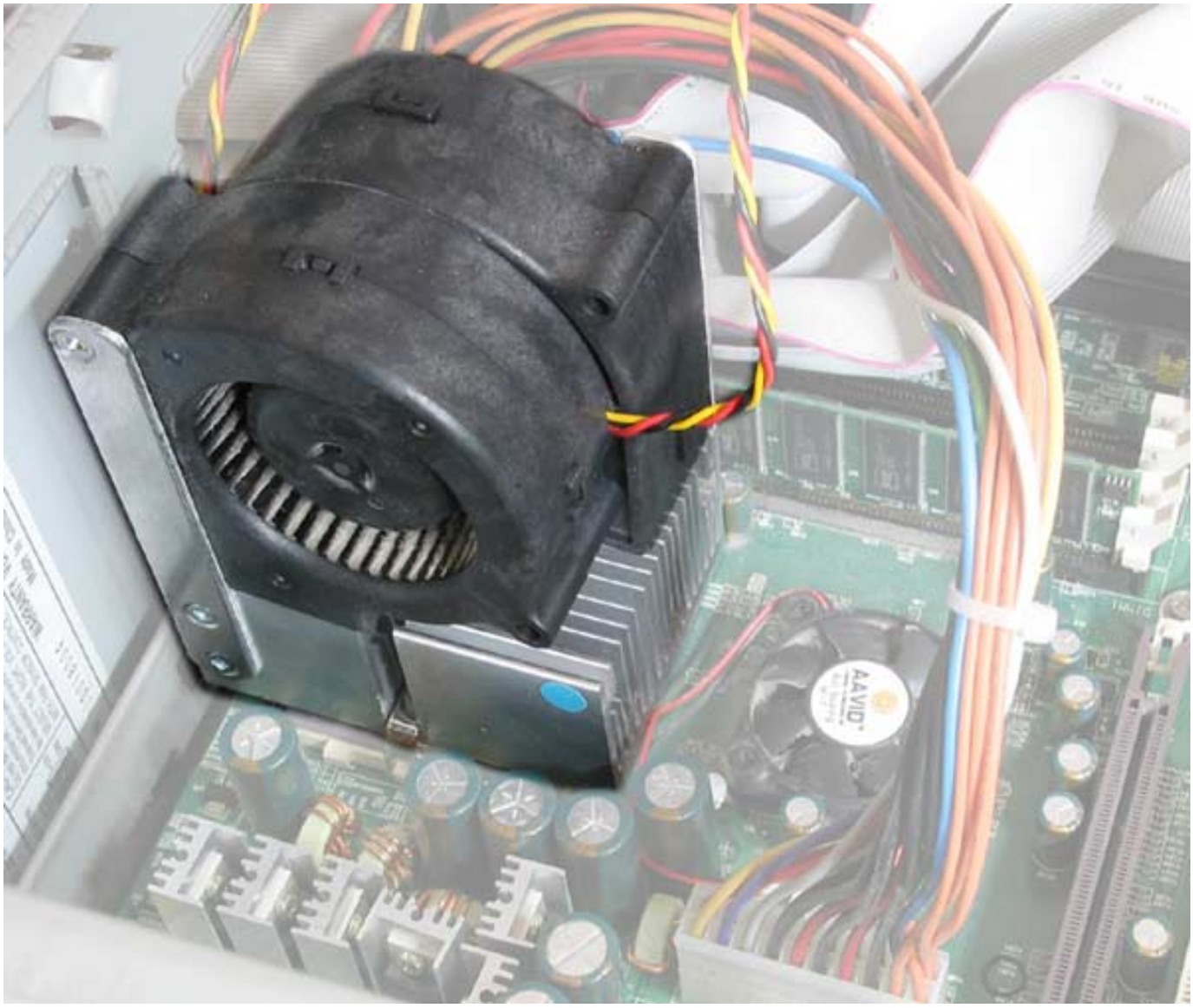
The disadvantage is that the system clock in these motherboard architectures also controls the I/O bus, which runs synchronously with the system bus. PCI bus devices (which we will come to in a later chapter) cannot handle being overclocked very much; otherwise faults can occur, for example in reading from the hard disk.

Overclocking typically requires a higher voltage for the CPU, and most motherboards can be set up to supply this:



Figur 123. Setting the CPU voltage using the motherboard's Setup program.

Many still use the same kind of overclocking on the Athlon XP and Pentium 4. The system clock has to be able to be adjusted in increments, which it can on many motherboards.



Figur 124. A gigantic cooler with two fans and pure silver contact surfaces. Silverado, a German product which is used for overclocking CPU's.

Example using the Pentium 4

A Pentium 4 processor is designed for a system clock of 200 MHz. If you can have a 3200 MHz model with a 200 MHz system bus, it can theoretically be clocked up to 4000 MHz by turning up the system clock. However, the processor needs a very powerful cooling system to operate at the increased frequencies:

System clock	CPU clock
200 MHz	3200 MHz
230 MHz	3700 MHz
250 MHz	4000 MHz

Fig. 125. Overclocking a Pentium 4 processor.

The manufacturers, Intel and AMD, don't like people overclocking their CPU's. They have sometimes attempted to prevent this by building a lock into the processors, so that the processor can only work at a specific clock frequency. In other cases the CPU's can be overclocked. In any case, you shouldn't expect

your warranty to apply if you play around with overclocking.

- Next chapter.
- Previous chapter.

- Next chapter.
- Previous chapter.

Chapter 19. Different types of RAM

RAM works in synch with the system bus, as I described in the previous chapter. But what is RAM actually? RAM is a very central component in a PC, for without RAM there can be no data processing. RAM is simply the storage area where all software is loaded and works from.

Silicon storage area

RAM is made in electronic chips made of so-called semiconductor material, just like processors and many other types of chips.

In RAM, transistors make up the individual storage cells which can each “remember” an amount of data, for example, 1 or 4 bits – as long as the PC is switched on.

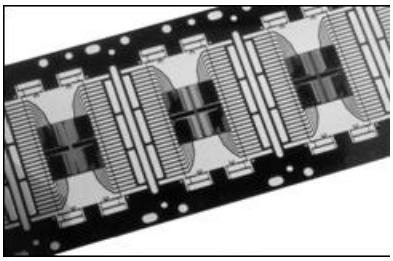


Fig. 126. Manufacturing RAM.

Normal RAM is *dynamic* (called DRAM), and requires constant electronic recharging to preserve its data contents. Without power, all RAM cells are cleared. RAM is very closely linked to the CPU, and it is very important to both have *enough* RAM, and to have *fast* RAM. If both conditions are not met, the RAM will be a bottleneck which will slow down the PC. What follows is an introduction to RAM, as it is used in modern PC’s. After this I will discuss the various types I more detail.

Several types

At the time of writing, there are several types of RAM, which are quite different. This means that they normally cannot be used on the same motherboard – they are not compatible.

However, some motherboards can have sockets for two types of RAM. You typically see this during periods where there is a change taking place from one type of RAM to another. Such motherboards are really designed for the new type, but are made “backwards compatible”, by making room for RAM modules of the old type.

The RAM types on the market at the moment are:

RAM type	Pins	Width	Usage
SD RAM	168	64 bit	Older and slower type. No use.
Rambus RAM	184	16 bit	Advanced RAM. Only used for very few Pentium 4’s with certain Intel chipsets.
DDR RAM	184	64 bit	A faster version of SD RAM. Used both for Athlon and Pentium 4’s. 2,5 Volt.
DDR2 RAM	240	64 bit	New version of DDR RAM with higher clock frequencies. 1,8 Volt.

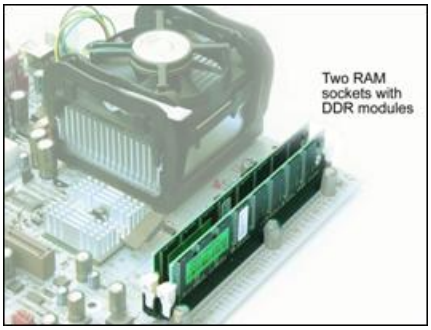
Figur 127. Very different types of RAM.

In any case, there is a lot of development taking place in DDR. A number of new RAM products will be released within the next few years. The modules are packaged differently, so they cannot be mixed. The notches in the sides are different as the he bottom edges of the modules are.

SDRAM is an old and proven type, which is used in the majority of existing PC’s. DDR RAM is a refinement of SDRAM, which is in reality double clocked. Rambus RAM is an advanced technology which in principle is superior to DDR RAM in many ways. However, Rambus has had a difficult birth. The technology has been patented by Rambus Inc., which has been involved in many legal suits. A number of important manufacturers (such as VIA) have opted out of Rambus, and only develop products which use DDR RAM. With the new DDR2 standard, there is no obvious need for Rambus RAM.

Notes on the physical RAM

RAM stands for *Random Access Memory*. Physically, RAM consists of small electronic chips which are mounted in modules (small printed circuit boards). The modules are installed in the PC's motherboard using sockets — there are typically 2, 3 or 4 of these. On this motherboard there are only two, and that's a bit on the low side of what is reasonable.



Figur 128. RAM modules are installed in sockets on the motherboard. In the background you see the huge fan on a Pentium 4 processor.

Each RAM module is a rectangular printed circuit board which fits into the sockets on the motherboard:

Fig. 129. A 512 MB DDR RAM module.

On a module there are typically 8 RAM chips which are soldered in place. There can also be 16 if it is a double-sided module. Below is a single RAM chip:

Fig. Figur 130.
A single RAM chip, a
256 megabit circuit.

On the bottom edge of the module you can see the copper coated tracks which make electrical contact (the edge connector). Note also the *profile* of the module; this makes it only possible to install it one way round and in the right socket.

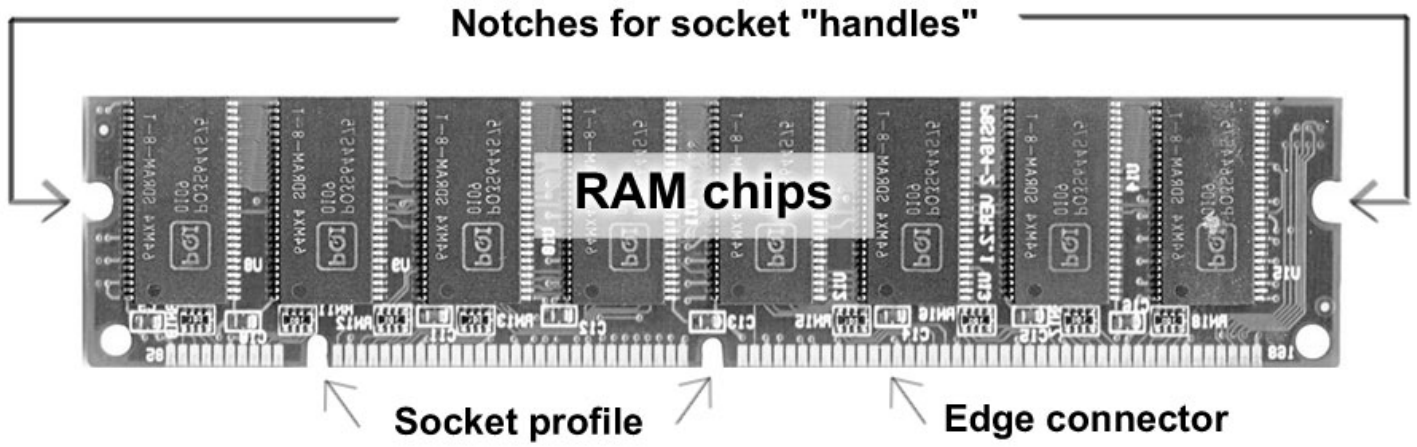


Fig. 131. Anatomy of the SD RAM module.

The notches in the sides of the module fit the brackets or "handles" which hold the module in place in the motherboard socket:

Module or chip size

All RAM modules have a particular *data width*, which has to match the motherboard, chipset, and ultimately the CPU. Modules using the two most common RAM types, SD and DDR RAM, are 64 bits wide.

The modules are built using chips which each contain a number of megabits. And since each byte needs 8 bits, more than one chip is needed to make a module. Look at the RAM chip in Fig. 130. You can see the text "64MX4":

Fig. 132. Text on a RAM chip.

The text in Fig. 132 indicates that the chip contains *64 x 4 mega bits* of data, which is the same as 256 megabits. If we want to do some calculations, each chip contains 1024 x 1024 x 64 = 67,108,864 cells, which can each hold 4 bits of data. That gives 268,435,456 bits in total, which (when divided by 8) equals 33,554,432 bytes = 32,768 KB = 32 MB.

So a 64MX4 circuit contains 32 MB, and is a standard product. This type of chip is used by many different manufacturers to make RAM modules. Modules are sold primarily in four common sizes, from 64 to 512 MB:

Number of chips per module	Module size
2 (single-sided)	2 x 256 Mbit = 64 Mbyte
4 (single-sided)	4 x 256 Mbit = 128 Mbyte
8 (single-sided)	8x 256 Mbit = 256 Mbyte
16 (double-sided)	16 x 256 Mbit = 512 Mbyte

Fig. 133. The module size is determined by what RAM chips are used.

RAM speeds

For each type of RAM, there are modules of various sizes. But there are also modules with various *speeds*. The faster the RAM chips are, the more expensive the modules.

Name	Type
PC700	2x 356 MHz Rambus RAM
PC800	2x 400 MHz Rambus RAM
PC1066	2 x 533 MHz Rambus RAM
DDR266 el. PC2100	2x 133 MHz DDR RAM
DDR333 el. PC2700	2x 166 MHz DDR RAM
DDR400 el. PC3200	2x 200 MHz DDR RAM
DDR2-400	400 MHz DDR2 RAM
DDR2-533	533 MHz DDR2 RAM
DDR2-667	667 MHz DDR2 RAM

Fig. 134. The speed is measured in megahertz, but it is not possible to compare directly between the three types of RAM shown in the table. The names shown are those which are being used on the market at the time of writing, but they may change.

- Next chapter.
- Previous chapter.

- Next chapter.
- Previous chapter.

Chapter 20. RAM technologies

Let’s look a little more closely at the technology which is used in the various types of RAM.

In the old days

Back in the 1980’s, DRAM was used. This was dynamic RAM, which was relatively slow. It was replaced by FPM (*Fast Page Mode*) RAM which was also dynamic, only a bit faster.

Originally, loose RAM chips were installed directly in large *banks* on the motherboard. Later people started combining the chips in modules. These came in widths of 8 bits (with 30 pins) and 32 bits (with 72 pins). The 32-bit modules were suited to the system bus for the 80486 processor, which was also 32 bits wide.

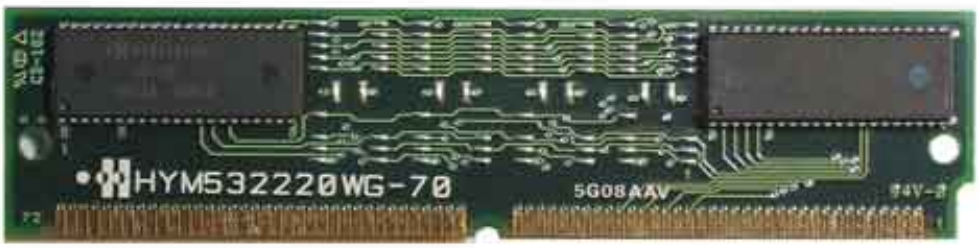


Fig. 135. Older RAM modules.

FPM RAM could not run any faster than 66 MHz, but that was fine for the system bus clock frequency in the original Pentium processors.

After FPM came EDO RAM (*Extended Data Out*). EDO is a bit faster than FPM because the data paths to and from the RAM cells have been optimised. The gain was a 3-5 % improvement in bandwidth. The clock frequency could be increased to 75 MHz, but basically, EDO is not very different to FPM RAM.

When Intel launched the Pentium processor, there was a change to using the 64 bit wide RAM modules (with 168 pins, as in Fig. 127 on page 51, which are still used for SDRAM today.

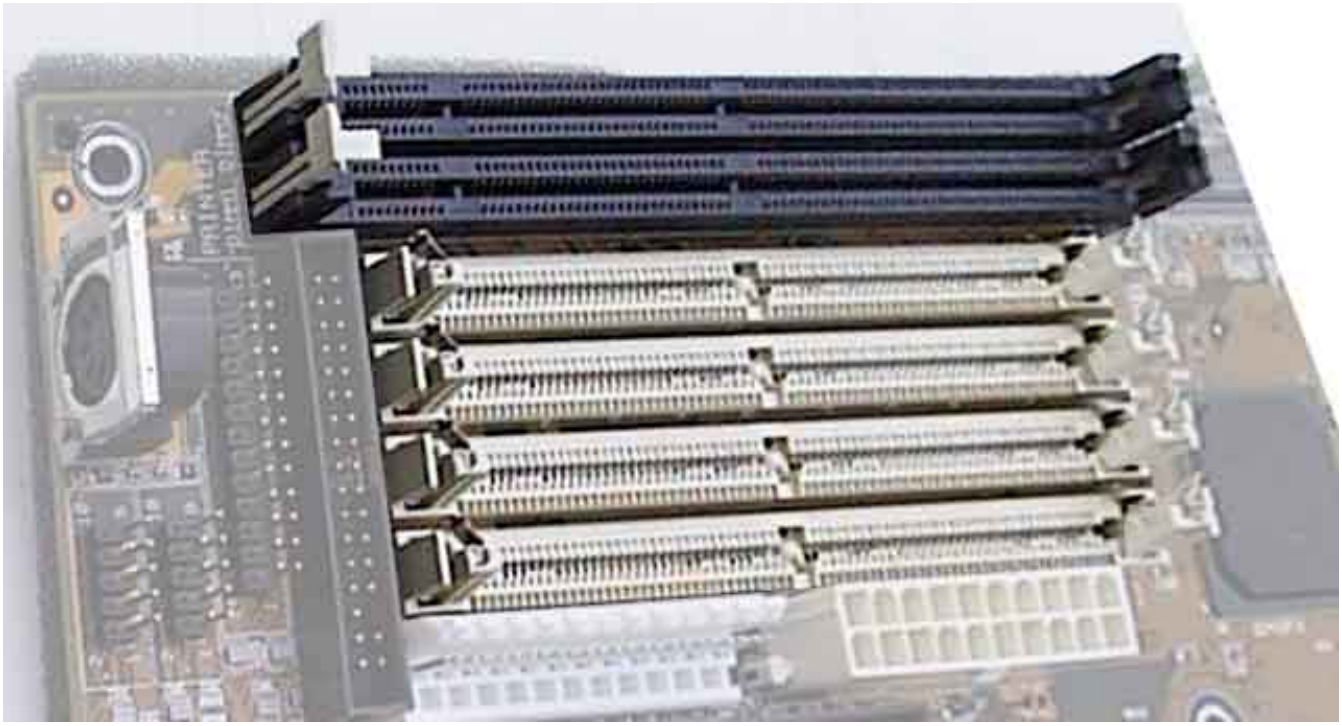


Fig. 136. An old motherboard with sockets for both 64-bit and 32-bit RAM modules. From the transition period between EDO and SDRAM.

SDRAM

The big qualitative shift came in around 1997, when SDRAM (*Synchronous DRAM*) began to break in. This is a completely new technology, which of course required new chipsets. SDRAM, in contrast to the earlier types of RAM, operates *synchronously* with the system bus.

Data can (in *burst mode*) be fetched on *every* clock pulse. Thus the module can operate fully synchronised with (at the same beat as) the bus – without so-called *wait states* (inactive clock pulses). Because they are linked synchronously to the system bus, SDRAM modules can run at much higher clock frequencies.

The 100 MHz SDRAM (PC100) quickly became popular, and with new processors and chipsets, the speed was brought up to 133 MHz (PC133).

Another innovation in SDRAM is the small EEPROM chip called the *Serial Presence Detect* chip, which is mounted on the modules. It is a very small chip containing data on the modules speed, etc.

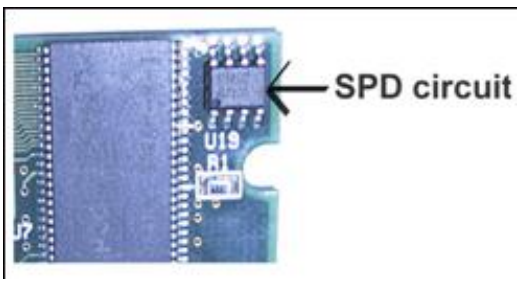


Fig. 137. The motherboard BIOS can now read SDRAM module specifications directly.

DDR RAM

It is expensive to produce fast RAM chips. So someone hit on a smart trick in 1999-2000, which in one blow made normal RAM twice as fast. That was the beginning of DDR RAM (*Double Data Rate*). See the module in Fig. 131.

In DDR RAM, the clock signal is used *twice*. Data is transferred *both* when the signal rises, and when it falls. This makes it possible to perform twice as many operations per clock pulse, compared to earlier RAM types:

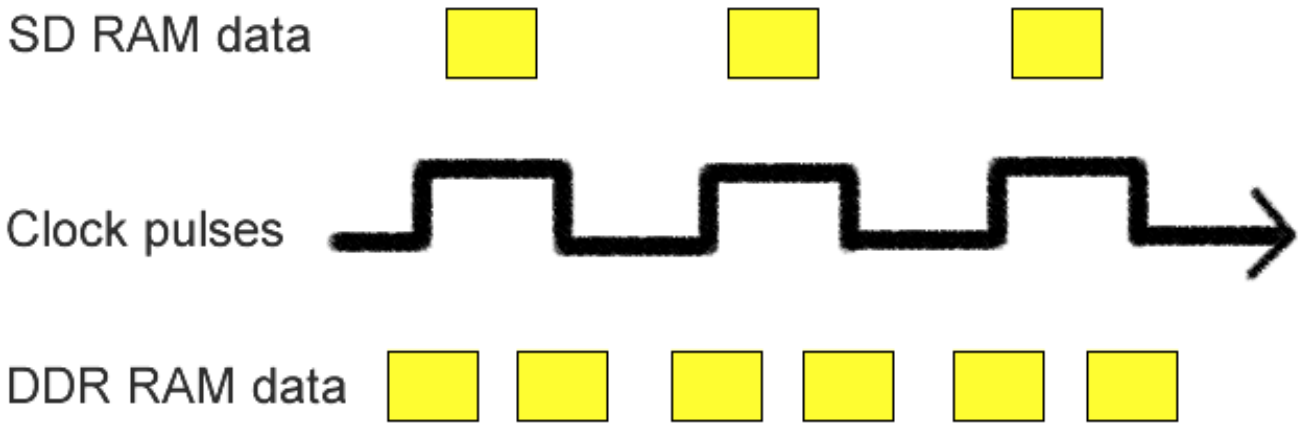


Fig. 138. DDR RAM sends off two data packets for each clock pulse.

Timings

DDR RAM exist in many versions, with different the clock frequencies and *timings.* The timing indicates how many clock cycles there are wasted, when the motherboard waits for the memory to deliver the requested data.

With smaller numbers, we have better timings and the CPU having fewer idle clock cycles. You may find memory modules of the same clock frequency but with different timings. The better timing, the more expensive the RAM module is.

Ordinary pc users need not to speculate in special RAM with fast timing; this is primary sold to gamers and over-clockers, who tries to achieve the maximum performance from their motherboards.

Module	Clock frequency	Timing
PC2100	2 x 133 MHz	2-2-2
PC 2700	2 x 166 MHz	2-2-2 2-3-3
PC 3200	2 x 200 MHz	2-3-2 2-3-3
PC 3700	2 x 233 MHz	3-4-4
PC 4000	2 x 250 MHz	3-4-4
PC 4400	2 x 275 MHz	3-4-4

Note that different timing means that the gain in terms of increased bandwidth doesn't quite match the clock speed. It is a bit less.

Next generation RAM

In the beginning the problem with DDR RAM, was that the RAM modules were poorly standardized. A module might work with a particular motherboard, but not with another. But this – which was typical for a new technological standard – is not a big problem anymore. Intel was initially against DDR RAM. They claimed that Rambus was a much better design, and that they wouldn't use DDR RAM. But consumers wanted DDR RAM, which Intel's competitors were able to deliver, and in the end even Intel had to give in. At the end of 2001, the i845 chipset was released, which uses DDR RAM for the Pentium 4, and later we had the i865 and i875 chip sets, which use dual channel DDR RAM.

The next generation of RAM is the *DDR2*, which is a new and better standardized version of DDR using less power. The DDR2 modules operates at higher clock speeds due to better design with higher signal integrity and a more advanced internal data bus. The first chip sets to use DDR2 was Intel's i915 and i925. Later *DDR4* is expected with clock frequencies of up to 1,6 GHz!

Rambus RAM

Rambus Inc., as already mentioned, has developed a completely new type of RAM technology. Rambus uses a completely different type of chip, which are mounted in *intelligent* modules that can operate at very high clock frequencies. Here is a brief summary of the system:

- The memory controller delivers data to a narrow high-speed bus which connects all the RDRAM modules in a long series.
- The modules contain logic (*Rambus ASIC*), which stores the data in the format the chips use.
- Data is written to one chip at a time, in contrast to SDRAM where it is spread across several chips.
- The modules work at 2.5 volts, which is reduced to 0.5 volts whenever possible. In this way, both the build up of heat, and electromagnetic radiation can be kept down. They are encapsulated in a heat conducting, aluminium casing.

Rambus RAM thus has a completely new and different design. The modules are only 16 bits wide. Less data is transferred per clock pulse, but the clock frequencies are much higher. The actual Rambus modules (also called RIMM modules) look a bit like the normal SDRAM modules as we know them. They have 184 pins, but as mentioned, the ships are protected by a heat-conducting casing:

Fig. 139. Rambus module.

As the advanced Rambus modules are quite costly to produce, the technology is on its way out of the market.

- Next chapter.
- Previous chapter.

- Next chapter.
- Previous chapter.

Chapter 21. Advice on RAM

RAM can be a tricky thing to work out. In this chapter I will give a couple of tips to anyone having to choose between the various RAM products.

Bandwidth

Of course you want to choose the best and fastest RAM. It's just not that easy to work out what type of RAM is the fastest in any given situation.

We can start by looking at the theoretical maximum bandwidth for the various systems. This is easy to calculate by *multiplying the clock frequency by the bus width*. This gives:

Module type	Max. transfer
SD RAM, PC100	800 MB/sec
SD RAM, PC133	1064 MB/sec
Rambus, PC800	1600 MB/sec
Rambus, Dual PC800	3200 MB/sec
DDR 266 (PC2100)	2128 MB/sec
DDR 333 (PC2700)	2664 MB/sec
DDR 400 (PC3200)	3200 MB/sec
DUAL DDR PC3200	6400 MB/sec
DUAL DDR2-400	8600 MB/sec
DUAL DDR2-533	10600 MB/sec

Fig. 140. The highest possible bandwidth (peak bandwidth) for the various types of RAM.

However, RAM also has to match the motherboard, chipset and the CPU system bus. You can try experimenting with overclocking, where you intentionally increase the system bus clock frequency. That will mean you need faster RAM than what is normally used in a given motherboard. However, normally, we simply have to stick to the type of RAM currently recommended for the chosen motherboard and CPU.

RAM quality

The type of RAM is one thing; the RAM quality is something else. There are enormous differences in RAM prices, and there are also differences in quality. And since it is important to have a lot of RAM, and it is generally expensive, you have to shop around.

One of the advantages of buying a clone PC (whether you build it yourself or buy it complete) is that you can use standard RAM. The brand name suppliers (like IBM and Compaq) use their own RAM, which can be several times more expensive than the standard product. The reason for this is that the RAM modules have to meet very specific specifications. That means that out a particular production run, only 20% may be “good enough”, and that makes them expensive.

Over the years I have experimented with many types of RAM in many combinations. In my experience, for desktop PC’s (not servers), you can use standard RAM without problems. But follow these precautions:

- Avoid mixing RAM from various suppliers and with various specifications in the same PC – even if others say it is fine to do so.
- Note that the RAM chips are produced at one factory, and the RAM modules may be produced at another.
- Buy standard RAM from a supplier you trust. You need to know who manufactured the RAM modules and the seller needs to have sold them over a longer period of time. Good brands are Samsung, Kingston and Corsair.
- The modules have to match the motherboard. Ensure that they have been tested at the speed you need to use them at.
- The best thing is to buy the motherboard and RAM together. It’s just not always the cheapest.
- Avoid modules with more than 8 chips on each side.

How much RAM?

RAM has a very big impact on a PC’s capacity. So if you have to choose between the fastest CPU, or more RAM, I would definitely recommend that you go for the RAM. Some will choose the fastest CPU, with the expectation of buying extra RAM later, “when the price falls again”. You can also go that way, but ideally, you should get enough RAM from the beginning. But how much is that?

If you still use Windows 98, then 256 MB is enough. The system can’t normally make use of any more, so more would be a waste. For the much better Windows 2000 operating system, you should ideally have at least 512 MB RAM; it runs fine with this, but of course 1024 MB or more is better. The same goes for Windows XP:

	128 MB	256 MB	512 MB	1024 MB
Windows 98	**	***	Waste	Waste

Windows 2000	*	**	***	****
Windows XP		*	***	****

Fig. 141. Recommended amount of PC RAM, which has to be matched to the operating system.

The advantage of having enough RAM is that you avoid *swapping*. When Windows doesn't have any more free RAM, it begins to artificially increase the amount of RAM using a *swap file*. The swap file is stored on the hard disk, and leads to a much slower performance than if there was sufficient RAM in the PC.

RAM addressing

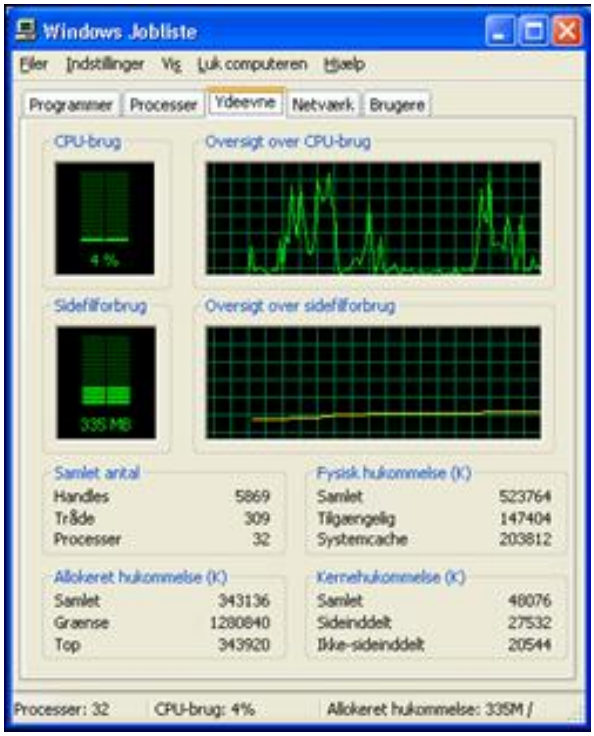
Over the years there have been many myths, such as "Windows 98 can't use more than 128 MB of RAM", etc. The issue is RAM *addressing*.

Below are the three components which each have an upper limit to how much RAM they can address (access):

- The operating system (Windows).
- The chipset and motherboard.
- The processor.

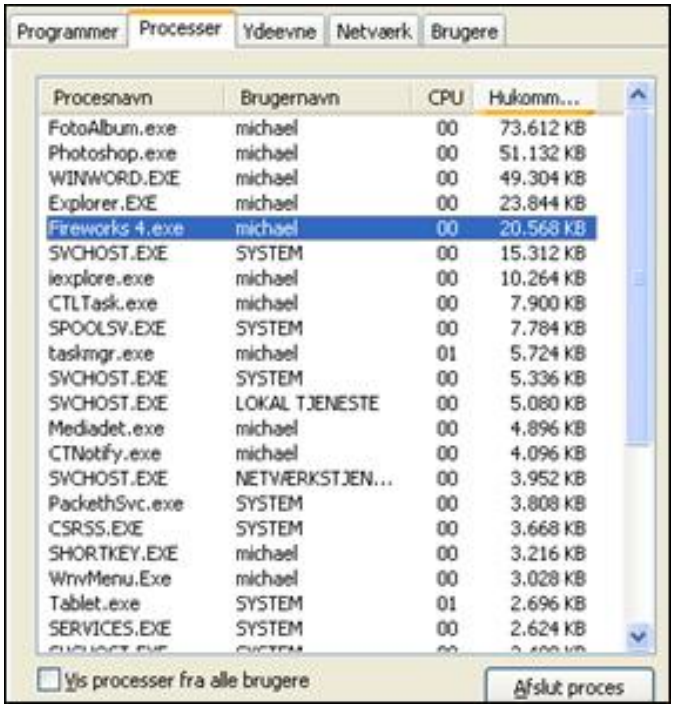
Windows 95/98 has always been able to access lots of RAM, at least in theory. The fact that the memory management is so poor that it is often meaningless to use more than 256 MB, is something else. Windows NT/2000 and XP can manage gigabytes of RAM, so there are no limits at the moment.

In Windows XP, you have to press Control+Alt+Delete in order to select the Job list. A dialog box will then be displayed with two tabs, Processes and Performance, which provide information on RAM usage:



Under the Processes tab, you can see how much RAM each program is using at the moment. In my case, the image browser, FotoAlbum is using 73 MB, Photoshop, 51 MB, etc., as shown in Fig. 143.

Modern motherboards for desktop use can normally address in the region of 1½-3 GB RAM, and that is more than adequate for most people. Server motherboards with special chipsets can address much more.



Figur 143. This window shows how much RAM each program is using (Windows XP).

Standard motherboards normally have a limited number of RAM sockets. If, for example, there are only three, you cannot use any more than three RAM modules (e.g. 3 x 256 MB or 3 x 512 MB).

CPU's have also always had an upper limit to how much RAM they can address:

Processor	Address bus width (bits)	Maximum System RAM
8088, 8086	20	1 MB
80286, 80386SX	24	16 MB
80386DX, 80486, Pentium, Pentium MMX, K5, K6 etc.	32	4 GB
Pentium Pro, Pentium II, III Pentium 4	36	64 GB

Fig. 144. The width of the CPU's address bus determines the maximum amount of RAM that can be used.

Let me conclude this examination with a quote. It's about RAM quantity:

"640K ought to be enough for anybody."
Bill Gates, 1981.

- Next chapter.
- Previous chapter.

Chapter 22. Chipsets and hubs

Since 1997, there has been more and more focus on refinement of the chipset, and not least the north bridge, which looks after data transfer to and from RAM. The south bridge has also been constantly developed, but the focus has been on adding new facilities.

For the north bridge, the development has focused on getting more bandwidth between the RAM and CPU. Let's look at a few examples of this.

Bridge or Hub

In a Pentium II motherboard, the I/O bus is directly linked to the system clock. The I/O bus (that is, in practise, the PCI bus) runs at 33 MHz, and that is typically a third or a quarter of the system clock speed (see Fig. 121 on page 49).

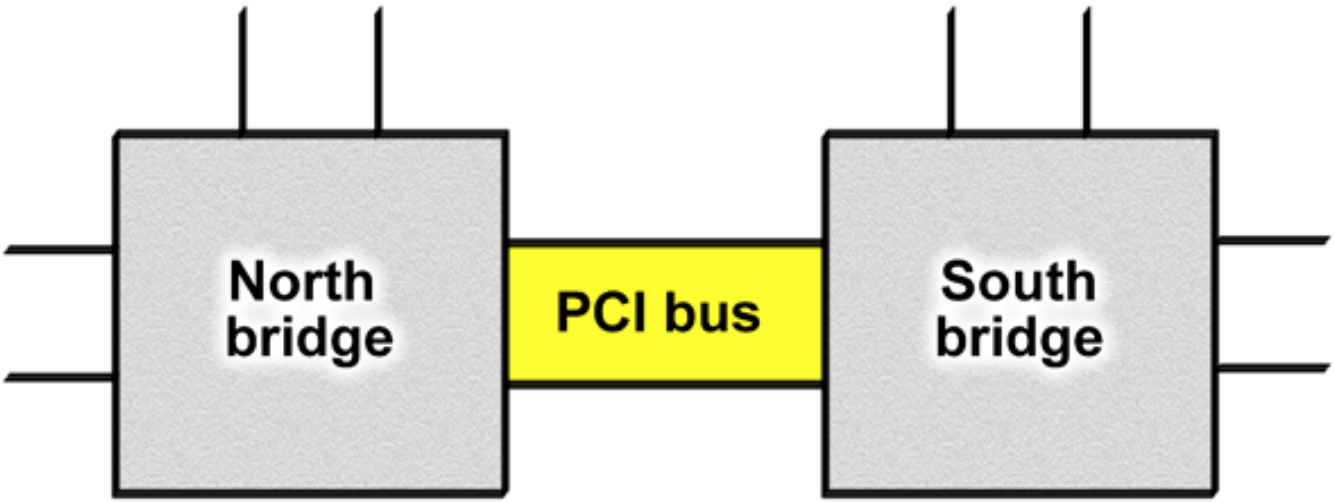


Fig. 145. In this architecture (from the Pentium II chipset), the PCI bus connects to the chipset's two bridges.

In 1998-99, new developments took place at both AMD and Intel. A new architecture was introduced based on a Memory Controller Hub (*MCH*) instead of the traditional north bridge and an I/O Controller Hub (*ICH*) instead of the south bridge. I am using Intel's names here; the two chips have other names at AMD and VIA, but the principle is the same. The first Intel chipset with this architecture was called *i810*.

The MCH is a controller located between the CPU, RAM and AGP. It regulates the flow of data to and from RAM. This new architecture has two important consequences:

- The connection between the two *hubs* is managed by a special bus (*link channel*), which can have a very high bandwidth.

- The PCI bus comes off the ICH, and doesn't have to share its bandwidth with other devices.

The new architecture is used for both Pentium 4 and Athlon processors, and in chipsets from Intel, VIA, and others. In reality, it doesn't make a great deal of difference whether the chipset consists of hubs or bridges, so in the rest of the guide I will use both names indiscriminately.

Figur 146. The MCH is the central part of the i875P chip set.

The i875P chipset

In 2003, Intel launched a chipset which work with the Pentium 4 and dual channel DDR RAM, each running at 200 MHz. This chip set became very popular, since it had a very good performance.

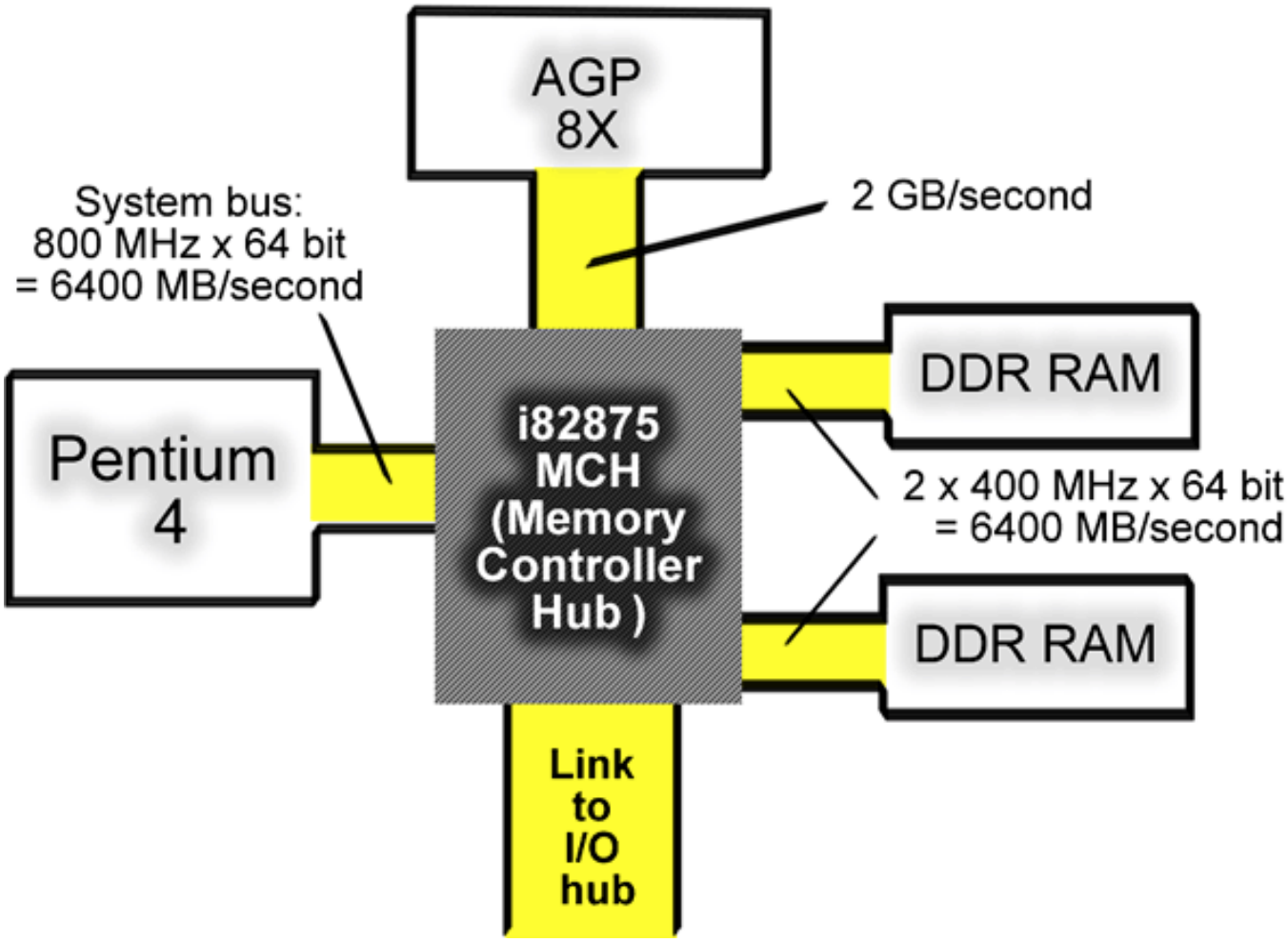
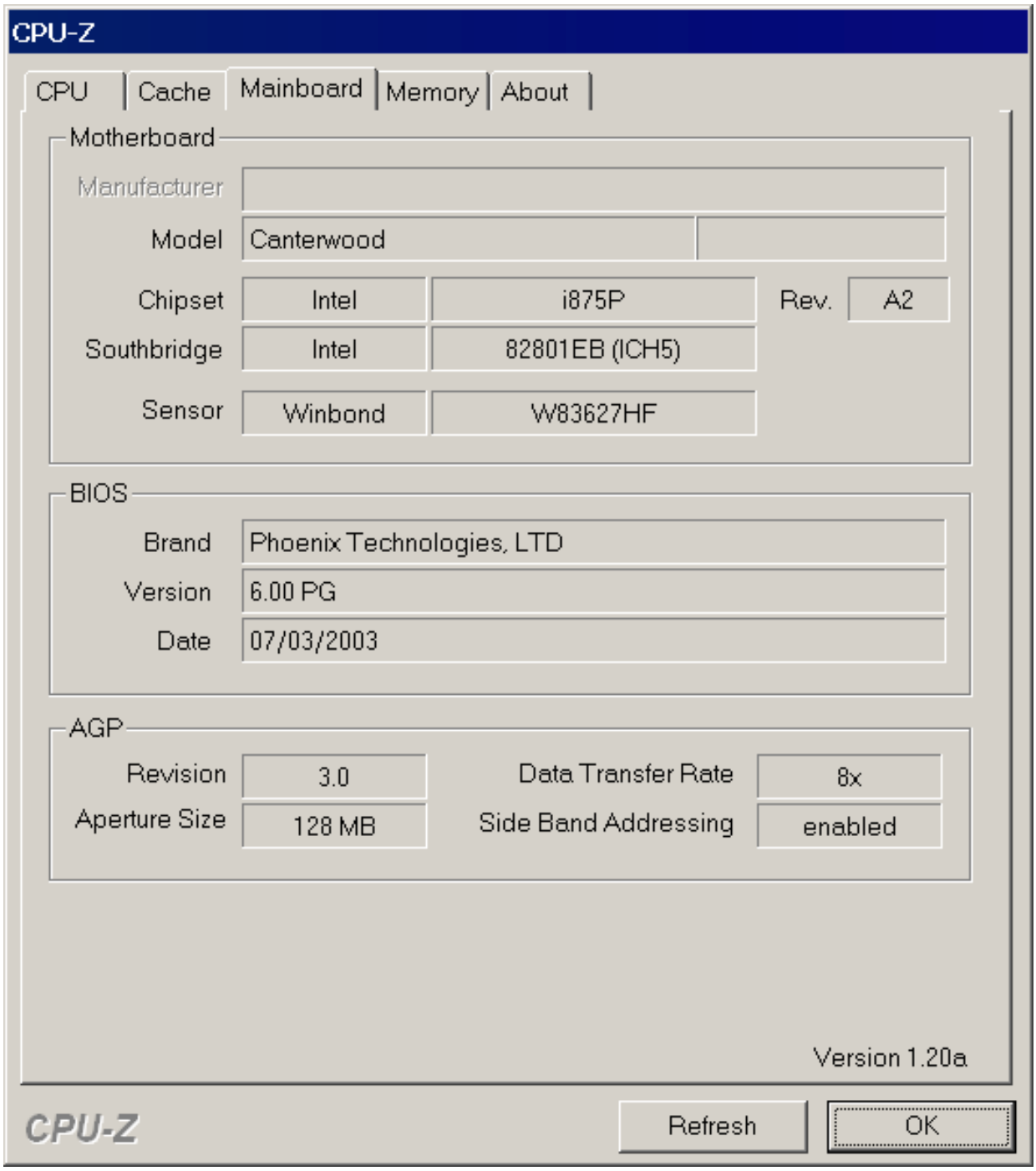


Fig. 147. The architecture surrounding the Intel® 82875P Memory Controller Hub (MCH).

Another new feature in this chip set is that a Gigabit Ethernet controller can have direct access to the MCH (the north bridge). Traditionally the LAN controller is connected to the PCI bus. But since a gigabit network controller may consume a great band width, it is better to plug it directly into north bridge. This new design is called Communication Streaming Architecture (CSA).



Figur 148. Report from the freeware program CPU-Z.

The i925 chipset

In late 2004 Intel introduced a new 900-series of chipsets. They were intended for the new generation of Pentium 4 and Celeron processors based on the LGA 775-socket (as in Figur 112 at page 44). The chip sets comes with support for the PCI Express bus, which is replacing the AGP bus and with support of DDR2 RAM:

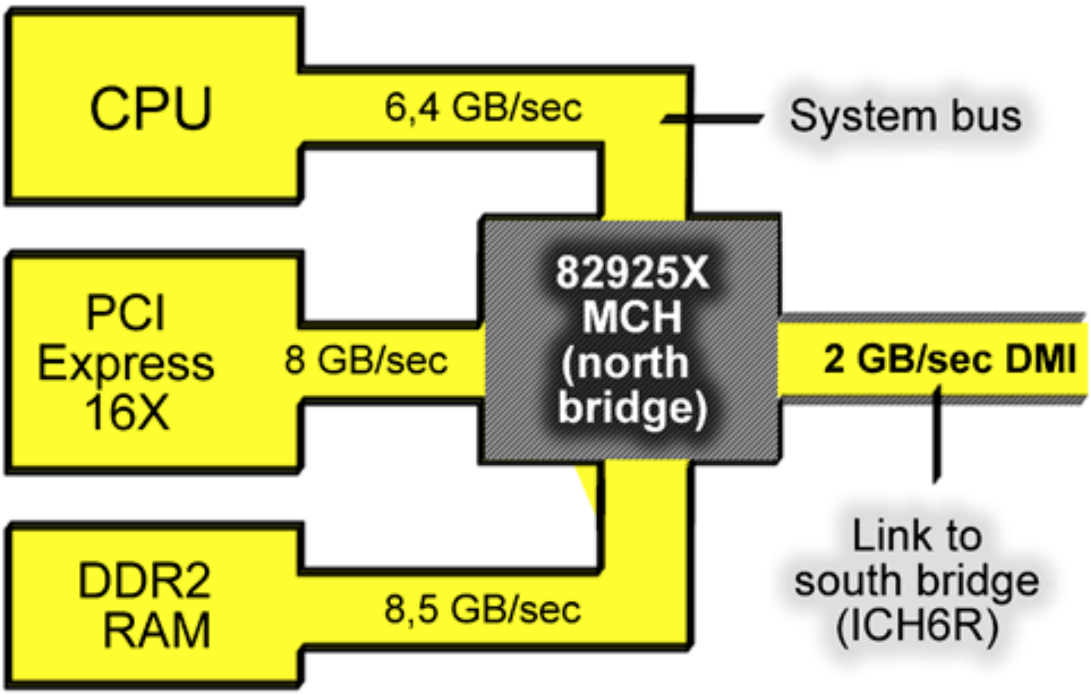


Fig. 149. The new chipset architecture, where the north bridge has become a hub. Here Intel chip set i925.

By making use of dual channel DDR2 RAM, a bandwidth of up to 8.5 GB/sec is achieved.

Big bandwidth for RAM

One might be tempted to think that the bandwidth to the RAM ought to be identical with that of the system bus. But that is not the case. It would actually be good if it was higher. That's because the RAM doesn't only deliver data to the CPU. Data also goes directly to the graphics port and to and from the I/O devices – bypassing the CPU. RAM therefore needs even greater bandwidth. In future architectures we will see north bridges for both Pentium 4 and Athlon XP processors which employ more powerful types of RAM, such as 533 MHz DDR2.

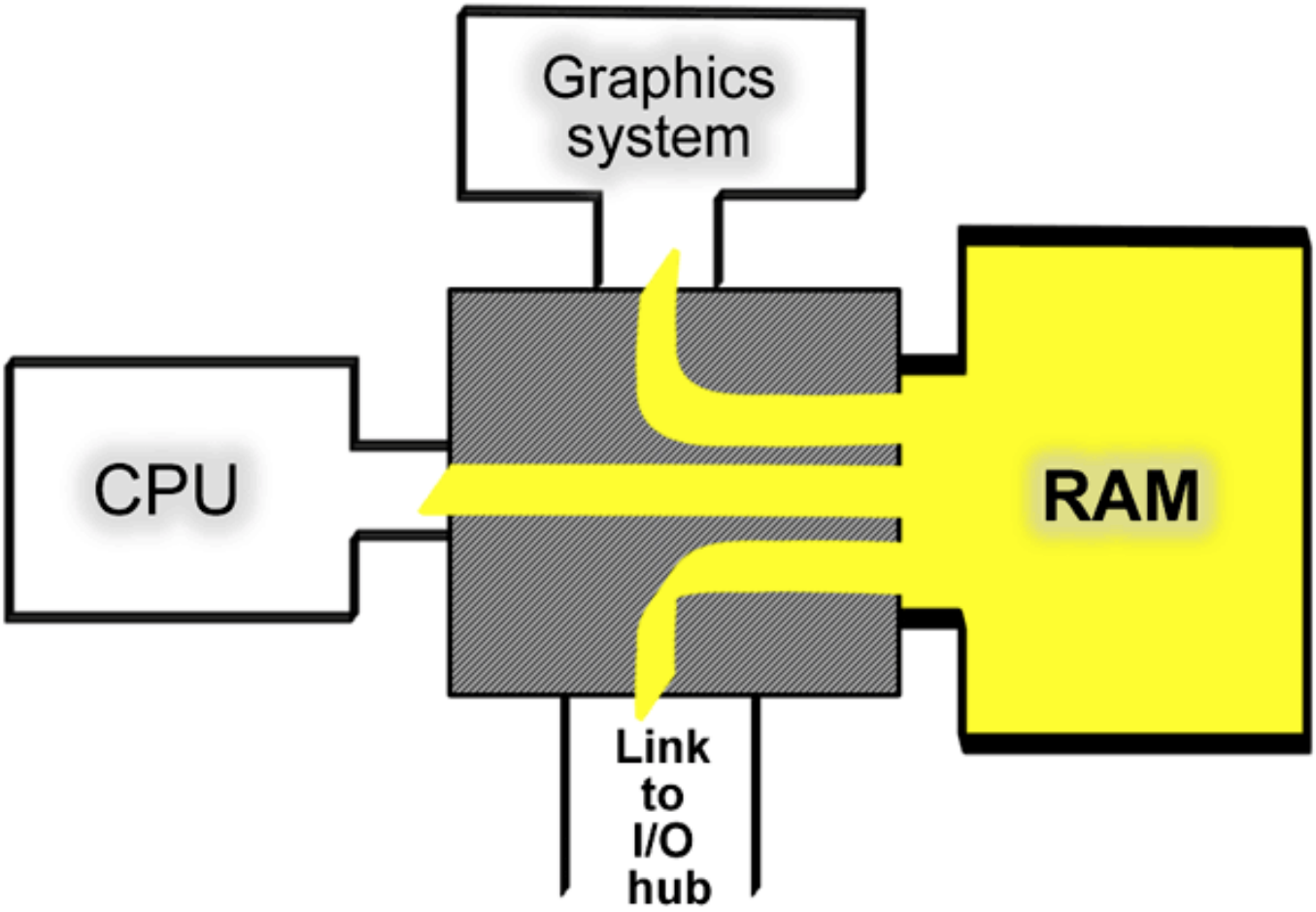


Fig. 150. In reality, the RAM needs greater bandwidth than the CPU.

- Next chapter.
- Previous chapter.

- Next chapter.
- Previous chapter.

Chapter 23. Data for the monitor

I have several times mentioned the AGP port, which is directly connected to the CPU and RAM. It is a high-speed port used for the video card, which has it's own RAM access.

Fig. Figur 151. AGP is a high-speed bus for the video card, developed by Intel.

About screens and video cards

As users, we communicate with PC programs via the screen. The screen shows us a graphical representation of the software which is loaded and active in the PC. But the screen has to be fed data in order to show a picture. This data comes from the *video card*, which is a controller.

Screens can be analogue (the traditional, big and heavy CRT monitors) or digital devices, like the modern, flat TFT screens. Whatever the case, the screen has to be controlled by the PC (and ultimately by the CPU). This screen control takes place using a video card.

A video card can be built in two ways:

- As a plug-in card (an *adapter*).
- Integrated in chips on the motherboard.

Finally, the video card can be connected either to the PCI bus, the AGP bus or the PCI Express x16 bus.

Big bandwidth

Traditionally, the video card (also called the *graphics card*) was connected as an I/O device. This meant that data for the video card was transferred using the same I/O bus which looks after data transfer to and from the hard disks, network card, etc.

In the late 1990's, the demands placed on the graphics system increased dramatically. This was especially due to the spread of the many 3D games (like Quake, etc.). These games require enormous amounts of data, and that places demands on the bus connection. At that time, the video card was connected to the PCI bus, which has a limited bandwidth of 133 MB/sec. The same bus, as just mentioned, also looks after the hard disk and other I/O devices, which all need bandwidth. The PCI bus therefore became a bottleneck, and the solution to the problem was to release the video card from the I/O bus.

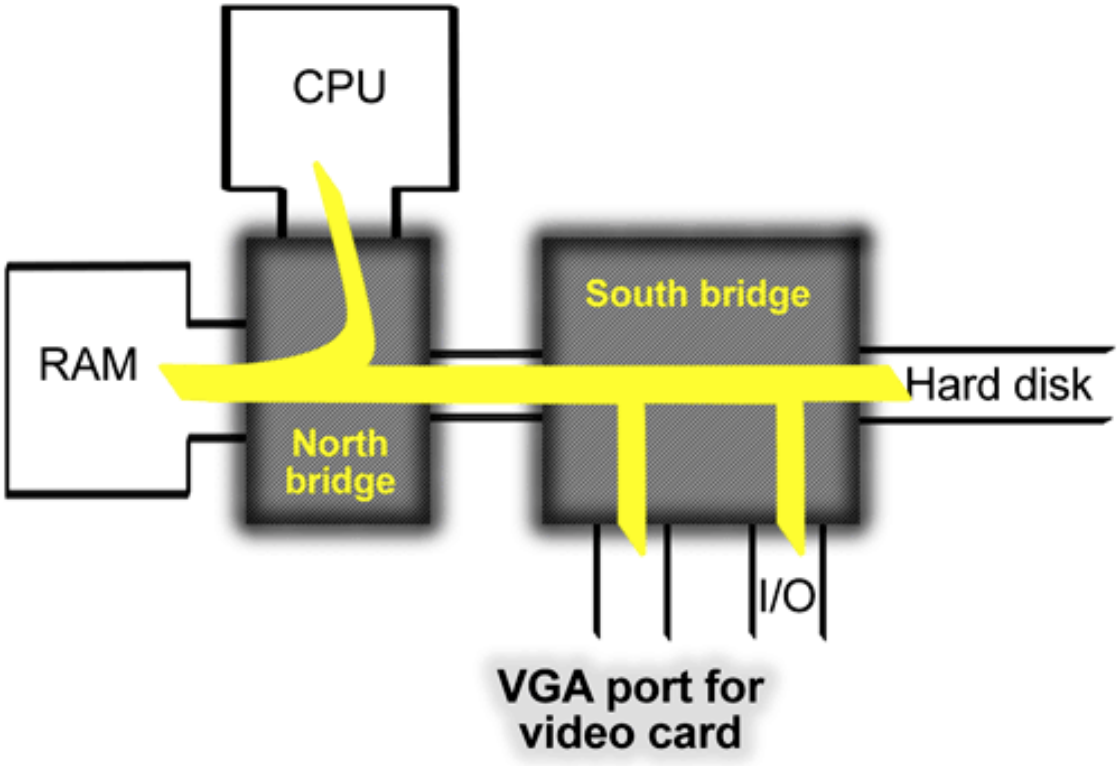


Fig. 152. The data path to the video card before the AGP standard.

AGP

AGP (*Accelerated Graphics Port*) is a special I/O port which is designed exclusively for video cards. AGP was developed by Intel. The AGP port is located close to the chipset's north bridge.

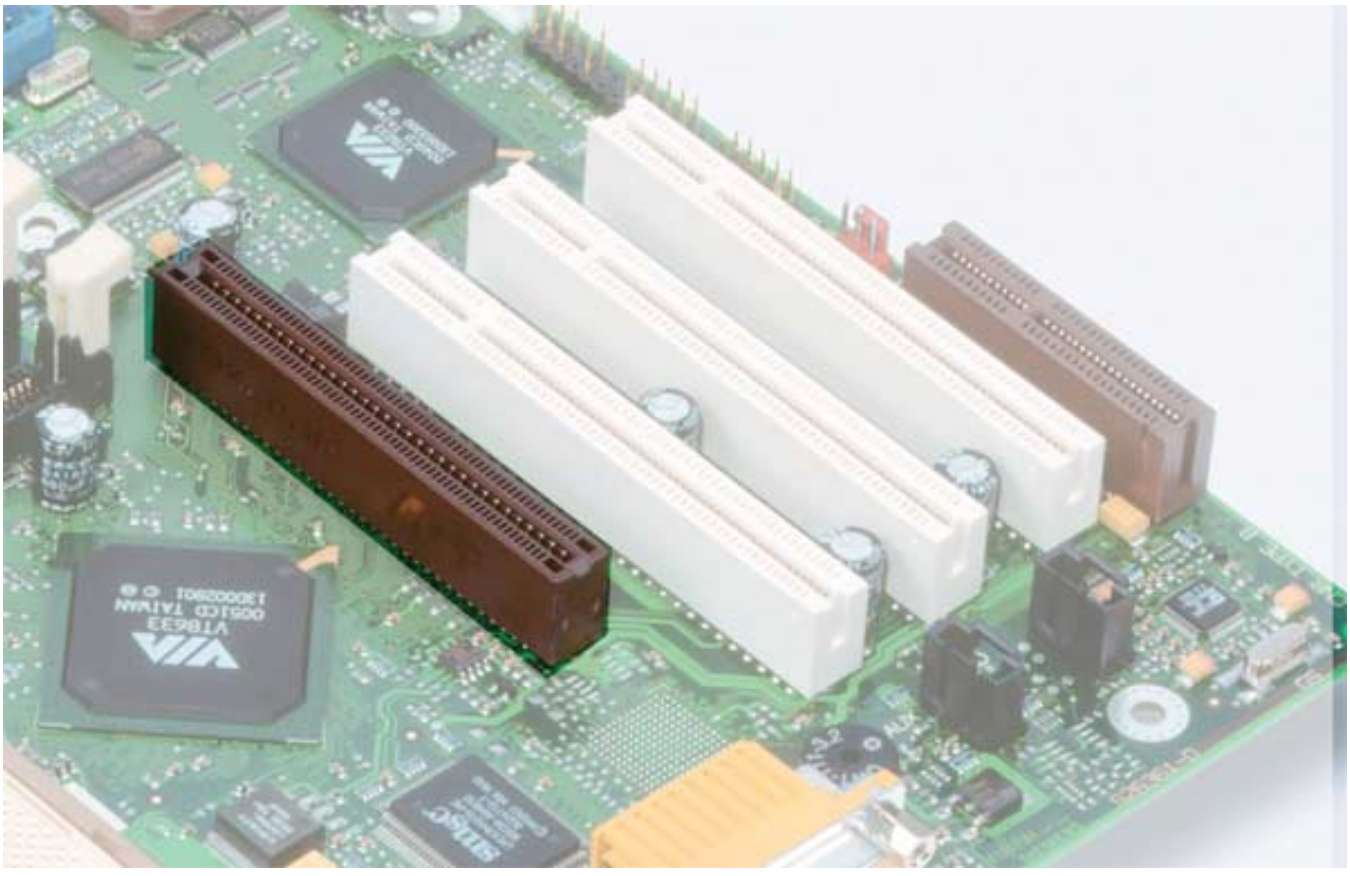


Fig. 153. The AGP slot can be seen on this motherboard, left of the three PCI slots.

The video card port was moved from the south to the north bridge. The new architecture gives optimal access to RAM and hence to the bandwidth which the 3D games require.

At the same time, the PCI system is spared from the large amount of graphic data traffic to and from the video card. It can now focus on the other intensive transfer tasks, such as transfer to and from the network adapter and disk drive.

Fig. 154. AGP Video card from ATI.

Technical details

AGP consists of a number of different technical elements, of which I will highlight two:

- A bus structure built around a double-clocked PCI bus.
- The ability to use the motherboard RAM as a *texture cache.*

The texture cache is used by games, and by giving access to the motherboard RAM, less RAM is needed on the cards.

The AGP bus is actually a 64-bit variant of the PCI bus. You can also see that on the surface, the motherboard AGP slot looks a fair bit like a PCI slot. But it is placed in a different position on the motherboard, to avoid confusion (see Fig. 156). The slot also has a different colour.

The first version of AGP was 1X, with a bandwidth of 254 MB/sec. But AGP was quickly released in a new mode, called 2X, with 508 MB/sec.

Later came 4X and 8X, which are the standards today. This involves a clock doubling, just as we have seen, for example, with DDR RAM. Two or four data packets are sent for each clock pulse. In this way, a bandwidth of 2,032 MB/sec has been reached.

Texture cache and RAMDAC

Textures are things like backgrounds in games. They can be uploaded directly from RAM to the video card. The system is called DIME (Direct Memory Execute). This allows the video card memory to be extended using standard RAM on the motherboard. In Fig. 155 you can see the system shown graphically.

In this figure you can also see the RAMDAC device. This is a chip on the video card which looks after the "translation" of digital data into *analogue* signals, when the card is connected to an analogue screen. The RAMDAC is a complete little processor in itself; the higher it's clock frequency, the higher the refresh rate with which the card can supply the screen image.

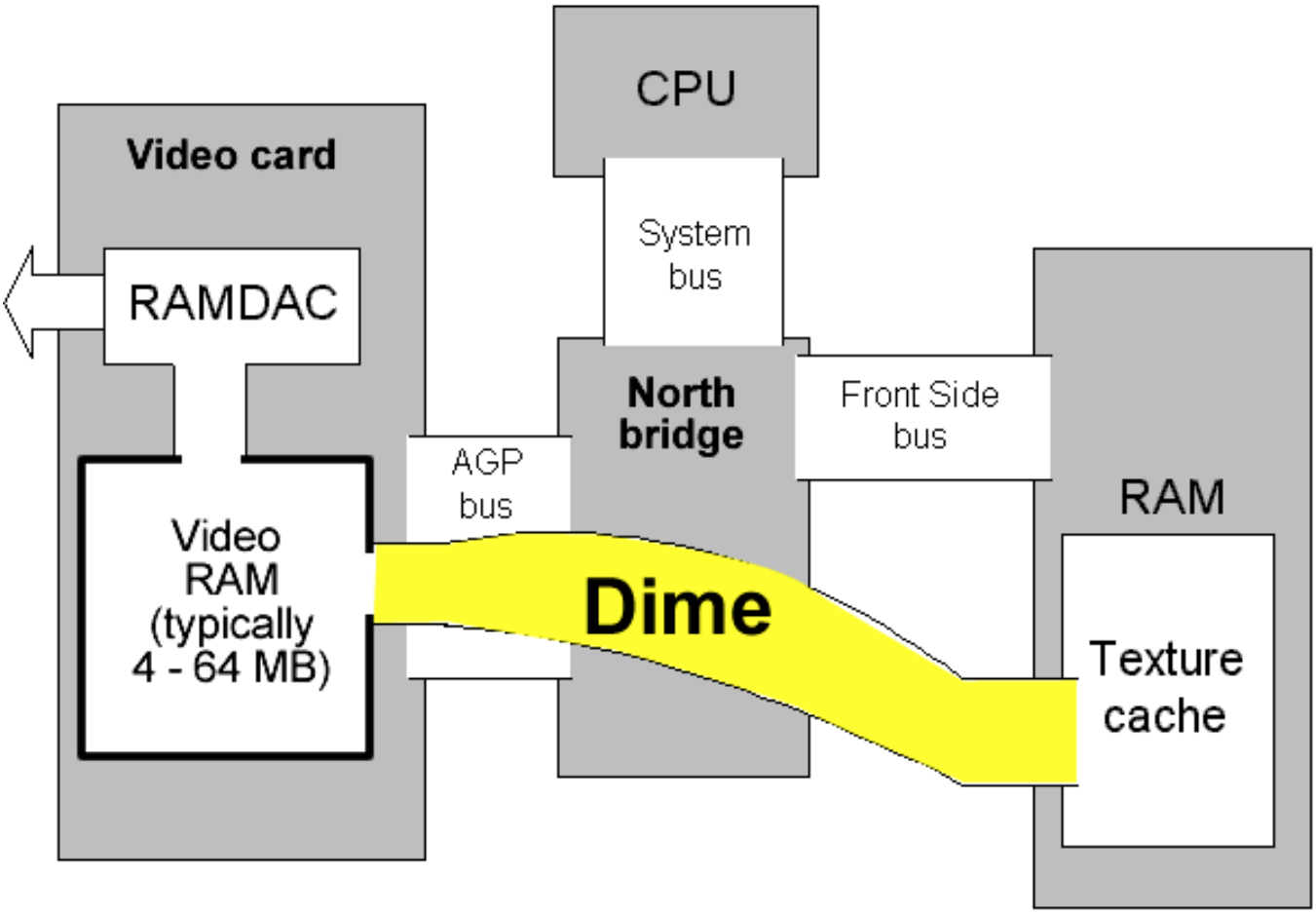


Fig. Figur 155. The AGP bus provides direct access to RAM.

Video card on PCI Express

With the new PCI Express bus, we get a new system for the video card. Replacing the AGP, the PCI Express X16-bus offers a transfer of 8 GB/sec, which leaves plenty of room for even the most graphical-intensive pc games.

Figur 156. The black PCI Express X16-slot to the left is for the video card.

The PC's I/O system

There are a lot of I/O ports in the PC's architecture, with their associated I/O devices and standards. I/O stands for Input/Output, and these ports can both send and receive data from the processor and RAM.

The I/O system provides flexibility

The use of I/O devices has contributed to making the PC an incredibly flexible machine. Computers can be used for anything from normal office tasks, processing text and numbers, to image processing using scanners and cameras, to producing video, light and music.

The PC can also be used industrially. In 1987-88 I worked in a company that produced special PC's which could control the production of concrete. This was achieved using special I/O cards which could monitor the weighing of sand, gravel, cement and water. The core of the system was a standard office PC from Olivetti.

This particularly flexible architecture is based on an I/O system which can be extended almost without limit. This is one place we really see the PC's open architecture: any engineer or technician can, in principle, develop their own plug-in cards and other special devices, if they just meet one of the I/O standards. The

opportunities for extension really are unlimited!

In the following chapters we will look at the various I/O buses which link the PC's other devices with the CPU and RAM.

- Next chapter.
- Previous chapter.

- Next chapter.
- Previous chapter.

Chapter 24. Intro to the I/O system

During the last 10-15 years we have seen numerous technological innovations, the goal of which has been to increase the amount of traffic in the PC. This increase in traffic has taken place in the motherboard – with the system bus at the centre.

But the high clock frequencies and the large capacity for data transfer also affect the I/O system. Demands are being made for faster hard disks and greater bandwidth to and from external devices such as scanners and cameras. This has led to ongoing development of the I/O controllers in the south bridge.

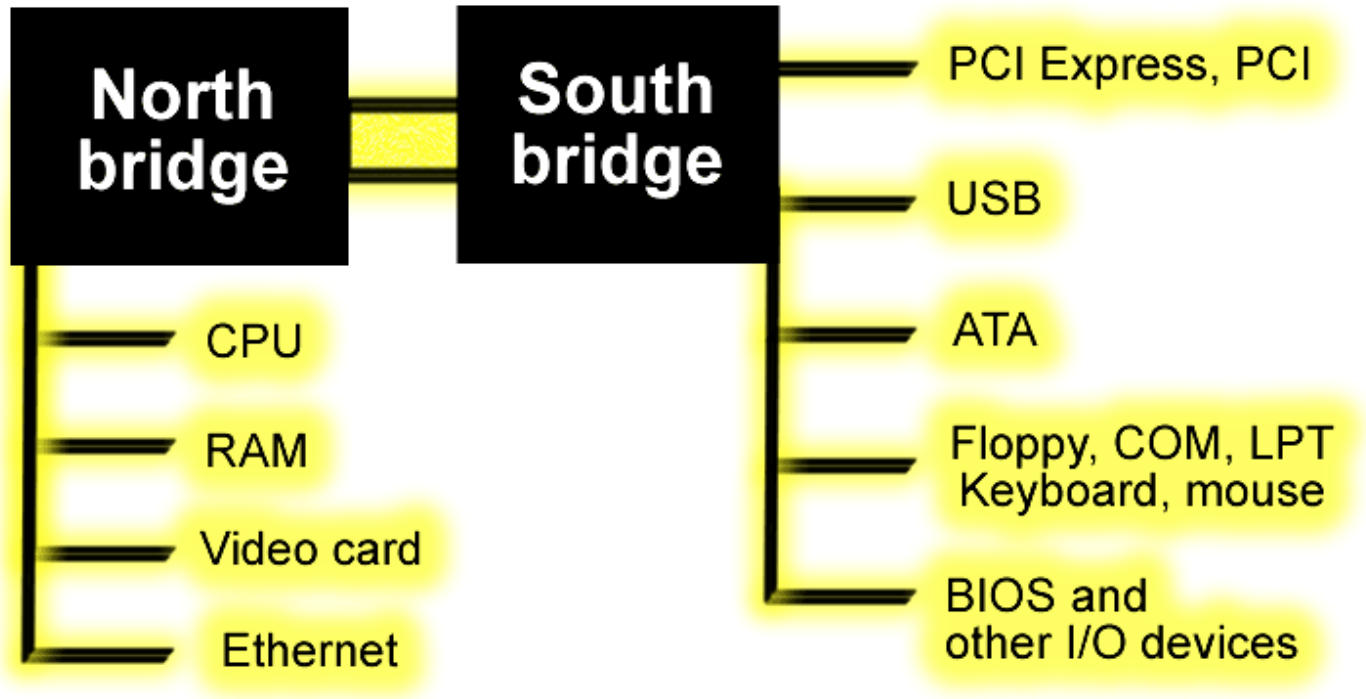


Fig. 157. The south bridge connects a large number of different devices with the CPU and RAM.

You can work out the capacity of a data bus based on data width and clock frequency. Here is a brief comparison between the system bus and the I/O buses:

Bus	**The north bridge's buses**	**The I/O buses**
Variants	FSB, RAM, AGP, PCI Express X16, CSA	ISA, PCI, PCI Express, USB, ATA, SCSI, FireWire
Connects	CPU, RAM, Video, Ethernet	All other devices.

Clock freq.	66 - 1066 MHz	Typically 10-33 MHz.
Maximum capacity	> 3 GB/sec.	Typically 20-500 MB/sec. per bus

Figur 158. The system bus is much faster than any other bus.

Function of the I/O buses

As I already mentioned, the south bridge was introduced back in 1987 (see Fig. 119 on page 49). The reason for this was that the I/O devices couldn't keep up with the high clock frequencies which the CPU and RAM could work at. There was electrical noise and other problems on the motherboard when the signals had to be carried to the plug-in cards, etc. Very few plug-in cards work at anything higher than 40 MHz – the electronics can't cope, the chips and conductors simply can't react that quickly. The I/O speed had to therefore be scaled down in relation to the system bus.

Since then, a number of different standards for I/O buses have been developed, which all emanate from the south bridge controllers. These include the older ISA, MCA, EISA and VL buses, and the current PCI, PCI Express and USB buses. The differences lie in their construction and architecture – in their connection to the motherboard.

I/O devices

The I/O buses run all over the motherboard, and connect a large number of different I/O devices. The south bridge controllers connect all the I/O devices to the CPU and RAM. Fig. 159 shows a brief overview of the various types of I/O devices. Note that AGP is not included, as it is tied to the north bridge.

Name	**Devices**
KBD, PS2, FDC, Game	Keyboard, mouse, floppy disk drive, joystick, etc.
ROM, CMOS	BIOS, setup, POST.
ATA	Hard disk, CD-ROM/RW, DVD etc.
PCI and PCI Express	Network card, SCSI controller, video grapper card, sound cards and lots of other adapters.
USB	Mouse, scanner, printers, modem, external hard disks and much more.
Firewire	Scanner, DV camera, external hard disk etc.
SCSI	Hard disks, CD-ROM drives, scanners, tape devices etc. (older)
LPT, COM	Parallel and serial devices such as printers, modems, etc.

Fig. Figur 159. Various types of I/O devices. The two last ones are not used much anymore.

The south bridge combines many functions

Originally, the various I/O devices could have *their own controller* mounted on the motherboard. If you look at a motherboard from the 1980's, there are dozens of individual chips – each with a particular function. As the years have passed, the controller functions have been gathered together into fewer and larger chips. The modern south bridge is now a large multi-controller, combining a number of functions previously managed by independent chips.

The south bridge is normally supplemented by a small *Super I/O* controller which takes care of a number of less critical functions that also used to be allotted to separate controller chips in the past. The Super I/O controller will be described in more detail later. It used to be connected to the south bridge via the ISA bus; in modern architectures the LPC (*Low Pin Count)* interface is used:

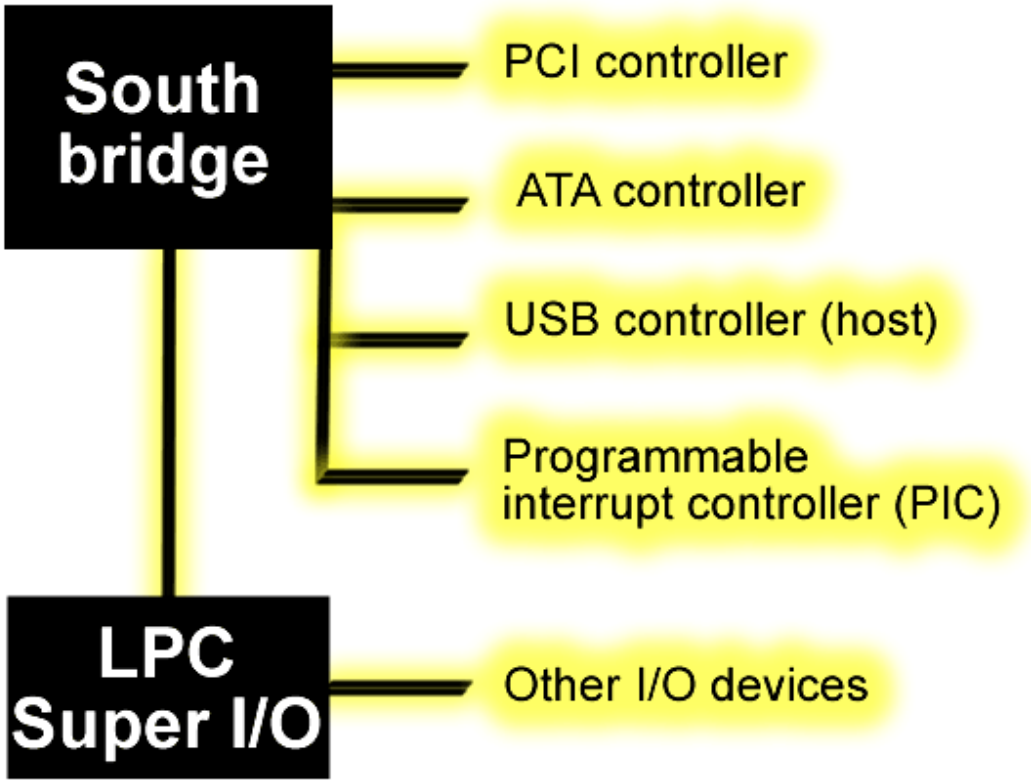


Fig. 160. The south bridge is part of the chipset, but is supplemented by the small Super I/O controller.

Several types of I/O bus

Throughout the years, several different I/O buses have been developed. These are the *proper* I/O buses:

- The ISA bus – an old, low-speed bus which is not used much any more.
- The MCI, EISA and VL buses – faster buses which are also not used any more.
- The PCI bus – a general I/O bus used in more modern PC's.
- The PCI Express – the most modern bus.

These "real" I/O buses are designed for mounting plug-in cards (*adapters*) inside the actual PC box. They therefore connect to a series of sockets (*slots*) on the motherboard:

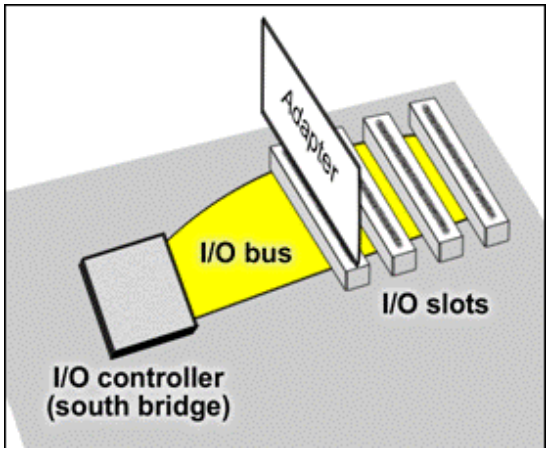


Fig. 161. Plug-in cards are mounted in the I/O slots on the motherboard.

The various I/O buses have gradually replaced each other throughout the PC's history. Motherboards often used to have several buses, but today basically only the PCI bus is used for installing adapters such as network cards, etc.

Figur 162. The features in this motherboard are all available through the choice of chipset. Here the south bridge delivers many nice features.

Other types of bus

The PC always needs a low-speed bus, used for the less intensive I/O devices. For many years that was the job of the ISA bus, but it has been replaced today by USB (Universal Serial Bus). USB is not a traditional motherboard bus, as it doesn't have slots for mounting plug-in cards. With USB, *external* devices are connected in *a series*. More on this later.

SCSI and FireWire are other types of high-speed bus which give the PC more expansion options. They are not part of the standard PC architecture, but they can be integrated into any PC. This is normally done using a plug-in card for the PCI bus, on which the SCSI or FireWire controller is mounted. Thus the two interfaces draw on the capacity of the PCI bus:

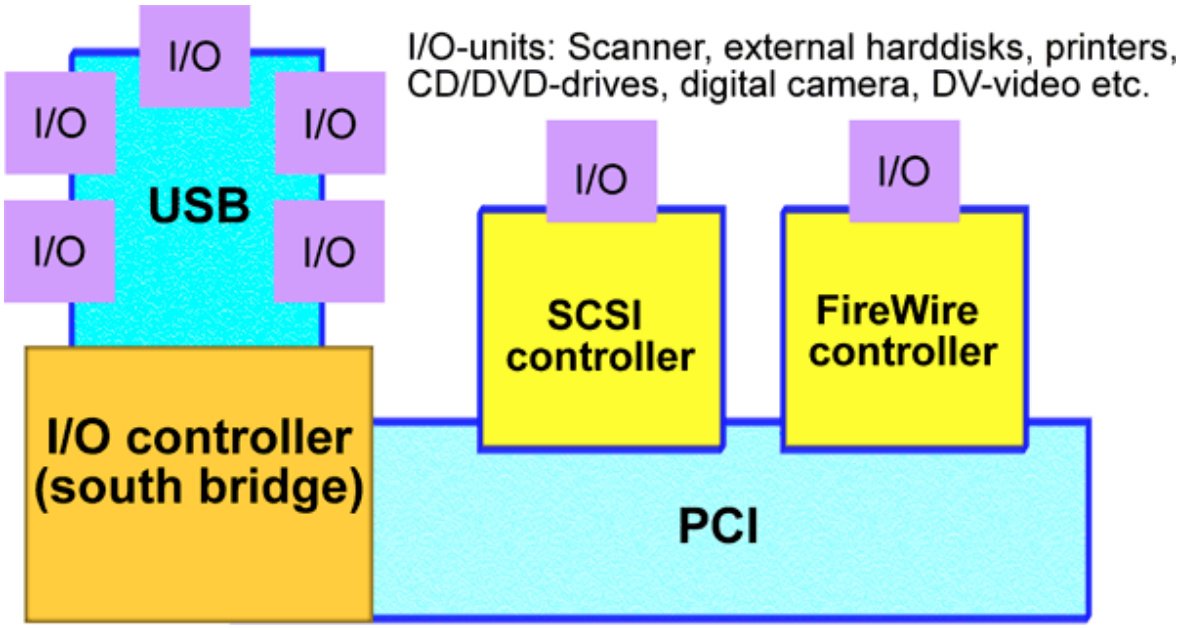


Fig. 163. SCSI and FireWire controllers are both normally connected to the PCI bus.

Finally, I should also mention the ATA hard disk interface, which is not normally called a bus, but which really could be called one. The ATA interface is only used for *drives*, and the standard configuration allows four devices to be connected directly to the motherboard, where the hard disk's wide cable, for example, fits inside an ATA connector.

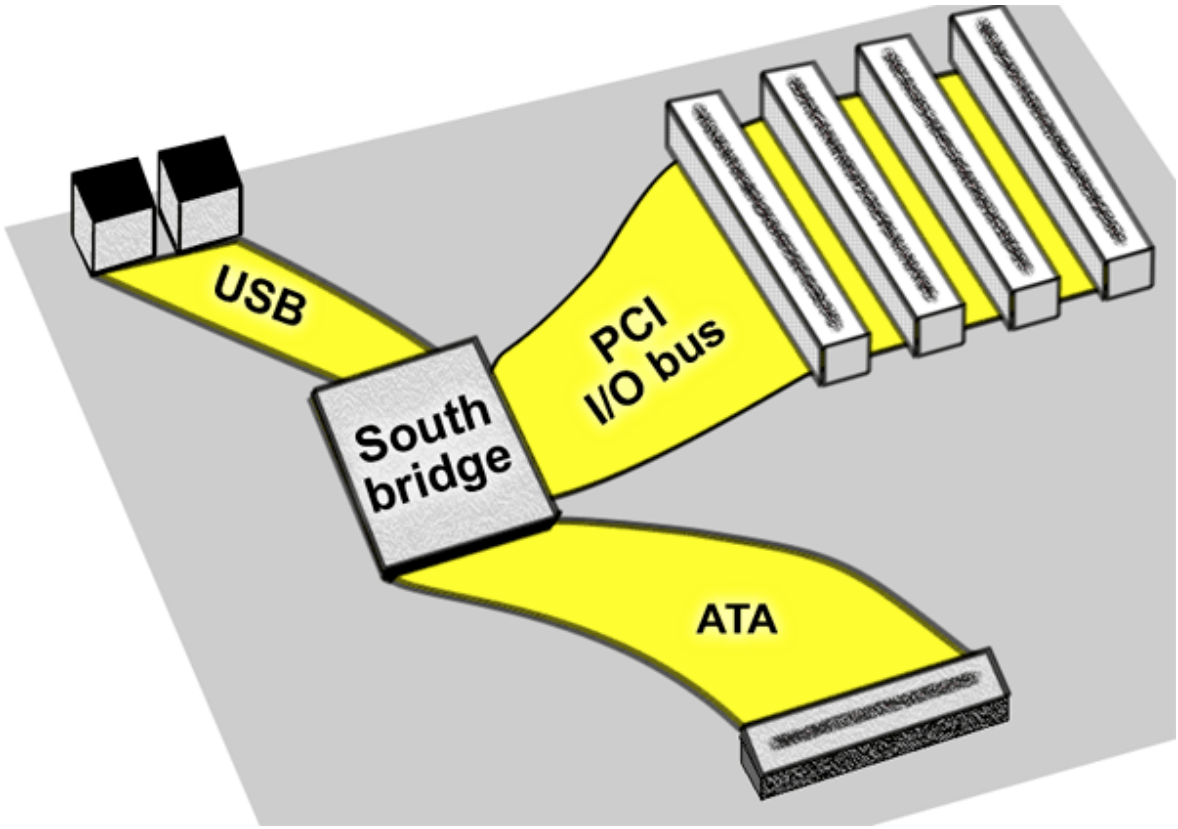


Fig. 164. The ATA interface works like a bus in relation to the south bridge.

- Next chapter.
- Previous chapter.

- Next chapter.
- Previous chapter.

Chapter 25. From ISA to PCI Express

From about 1984 on, every PC had a standard bus which was used for I/O tasks. That was the ISA (*Industry Standard Architecture*) bus.

Right up until about 1999 there were still ISA slots in most PC's. In the later years, however, they were only kept for compatibility, so that plug-in cards of the old ISA type could be re-used. This was particularly the case for *sound cards* from SoundBlaster; they worked quite well on the ISA bus, and many games were programmed to directly exploit this type of hardware. It therefore took many years to get away from the ISA bus, but we have managed to now.

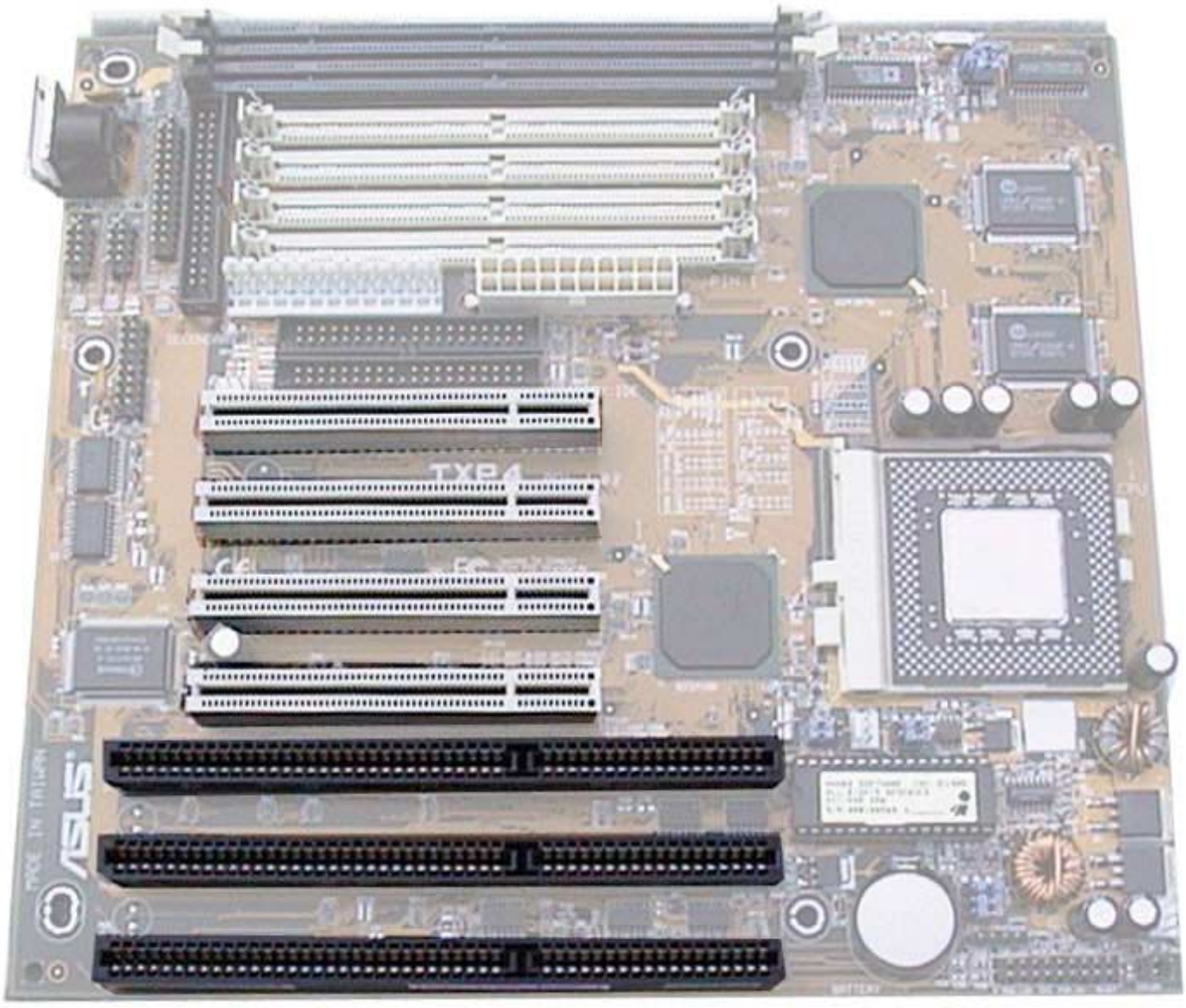


Fig. 165. Motherboard from 1998 with three (black) ISA slots and four (white) PCI slots.

History of the ISA bus

The ISA bus is thus the I/O bus which survived the longest. Here is some information about it:

ISA was an improvement to IBM's original XT bus (which was only 8 bits wide) IBM used the protected name "AT Bus", but in everyday conversation it was called the ISA bus.
The ISA bus was 16 bits wide, and ran at a maximum clock frequency of 8 MHz.
The bus has a theoretical bandwidth of about 8 MB per second. However in practise it never exceeds about 1-2 MB/sec. – partly because it takes 2-3 of the processor's clock pulses to move a packet (16 bits) of data.

Fig. 166. The ISA bus is not used much today, but it had enormous significance in the years up until the middle of the 1990's.

The ISA bus had two "faces" in the old PC architecture:

- An internal ISA bus, which the simple ports (the keyboard, floppy disk and serial/parallel ports) were connected to.
- An external expansion bus, to which 16-bit ISA adapters could be connected.

Sluggish performance

One of the reasons the ISA bus was slow was that it only had 16 data channels. The 486 processor, once it was introduced, worked with 32 bits each clock pulse. When it sent data to the ISA bus, these 32-bit packets (*dwords* or *doublewords*) had to be split into two 16-bit packets (two *words*), which were sent one at a time, and this slowed down the flow of data.

Bus	Time per packet	Amount of data per packet
ISA	375 ns	16 bits
PCI	30 ns	32 bits

Fig. 167. The PCI bus was a huge step forward.

The ISA bus was not "intelligent" either, since it was in principle the CPU which controlled all the work the bus was doing. This meant the CPU could only begin a new task when the transfer was complete. You may perhaps have experienced yourself, that when your PC works with the floppy disk drive – the rest of the PC virtually grinds to a halt. That's the ISA bus's fault, and it therefore only happens on older PC's.

These small delays are called *wait states.* If an ISA adapter cannot keep up with the data it is receiving, it sends wait states to the CPU. These are signals to the CPU telling it to do nothing. A wait state is a wasted clock pulse – the CPU skips over a clock pulse, without doing anything. Thus a slow ISA adapter could choke any PC.

Another problem was that the ISA bus often played up when you were installing an expansion card (e.g. a sound card). Many of the problems were associated with the handling of IRQ's and DMA channels (I will explain these terms later), which often had to be done "by hand" with the old ISA bus.

Every device takes up one particular IRQ, and possibly a DMA channel, and conflicts often arose with other devices. It was a big relief when Intel, in the late 1990's, finally dropped the ISA bus and replaced it with the smart USB bus.

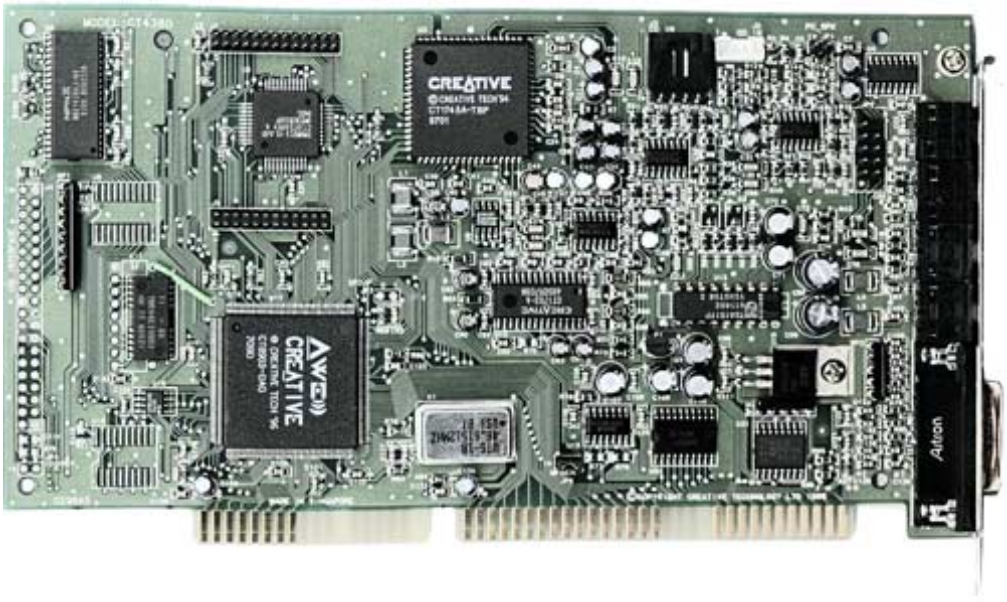


Fig. 168. ISA based Sound Blaster sound card.

The MCA, EISA and VL buses

The ISA bus was too slow, and the solution was to develop new standards for I/O devices. In 1987-88, two new I/O buses were put forward. First, IBM brought out their technologically advanced MCA bus. But since it was patented, a number of other companies pooled together to create the equivalent EISA bus.

But neither MCA or EISA had a big impact on the clone market. We were stuck with the ISA bus up until 1993, when the VL bus finally became reasonably widespread. It was a genuine *local bus*, which means it worked synchronously with the system bus. The VL bus was very primitive; it was really just an extension of the system bus.

The VL bus never managed to have a big impact, because almost at the same time, the robust and efficient PCI bus broke through. The various I/O buses are summarised below:

Bus	Description
PC-XT from 1981	Synchronous 8-bit bus which followed the CPU clock frequency of 4.77 or 6 MHz. Band width: 4-6 MB/sec.
ISA (PC-AT) from 1984	Simple, cheap I/O bus. Synchronous with the CPU. Band width: 8 MB/sec.
MCA from 1987	Advanced I/O bus from IBM (patented). Asynchronous, 32-bit, at 10 MHz. Band width: 40 MB/sec.
EISA From 1988	Advanced I/O bus (non-IBM), used especially in network servers. Asynchronous, 32-bit, at 8.33 MHz. Band width: 32 MB/sec.
VESA Local Bus from 1993	Simple, high-speed I/O bus. 32-bit, synchronised with the CPU's clock frequency: 33, 40, 50 MHz. Band width: up to 160 MB/sec.
PCI from 1993	Advanced, general, high-speed I/O bus. 32-bit, asynchronous, at 33 MHz. Band width: 133 MB/sec.
USB and Firewire, from 1998	Serial buses for external equipment.
PCI Express from 2004	A serial bus for I/O cards with very high speed. Replaces PCI and AGP.

	500 MB/sec. per. Channel.

Fig. 169. The PC's I/O buses, throughout the years.

The PCI bus

PCI stands for Peripheral Component Interconnect. The bus is an Intel product which is used in all PC's today, and also in other computers, as the PCI bus is processor independent. It can be used with all 32-bit and 64-bit processors, and is therefore found in many different computer architectures.

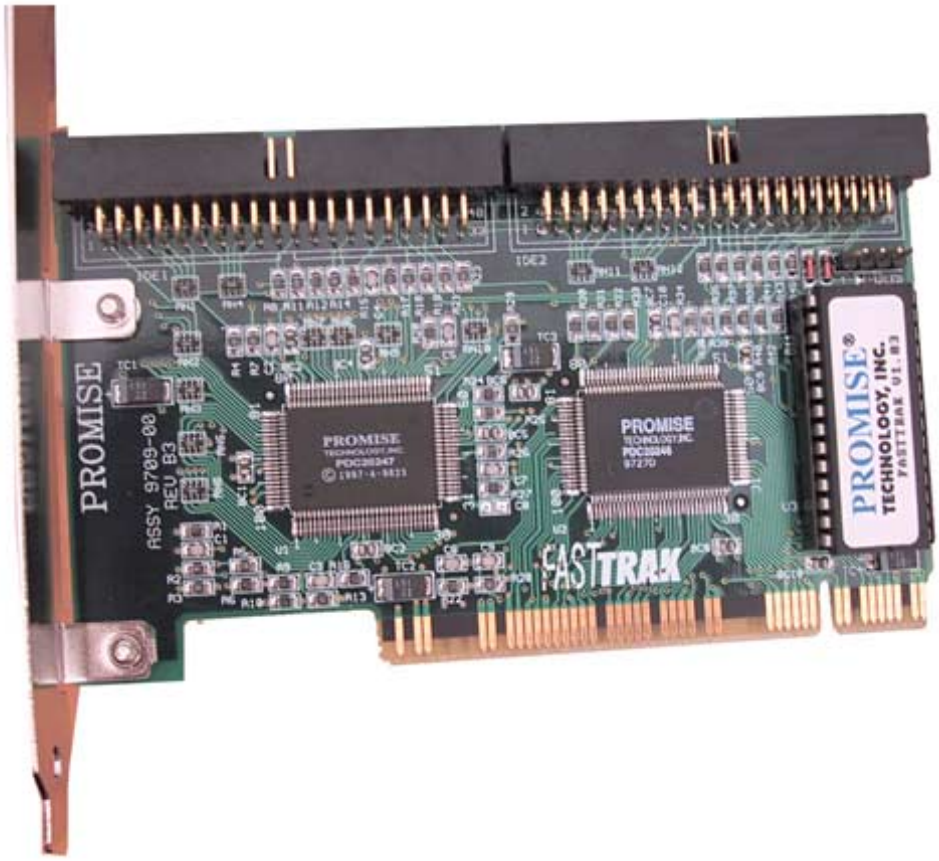


Fig. 170. PCI bus adapter.

At the same time, the bus is compatible with the ISA bus to a certain extent, since PCI devices can react to ISA bus signals, create the same IRQ's etc. One consequence of this was that Sound Blaster compatible sound cards could be developed, which was very important in the middle of the 1990's. In optimal conditions, the PCI bus sends one packet of data (32 bits) each clock pulse. The PCI bus therefore has a maximum bandwidth of 132 MB per second, as shown below:

Clock frequency:	33 MHz
Bus width:	32 bits
Bandwidth:	32 bits x 33,333,333 clock pulses/second = 4 bytes x 33,333,333 clock pulses/second = 132 MB per second

Fig. 171. The maximum bandwidth of the PCI bus.

There is also a more powerful versions of the PCI standard, which provides greater bandwidth, but most motherboards still use the original version. The PCI bus has a buffer which operates between the CPU and the peripheral devices (a kind of cache RAM). This allows the CPU to deliver its data to the buffer, and then perform other tasks. The bus looks after the rest of the delivery itself at its own pace. Alternatively, PCI adapters can also deliver data to the buffer, whether or not the CPU has time to process it. The data just stands in a queue and waits until there is room on the system bus, which then relays it to the CPU.

As a result of all this, the peripheral PCI devices operate *asynchronously* – at their own pace – in relation to the CPU. Thus the PCI bus (in contrast to the VL bus) is not a *local bus* from a technical perspective.

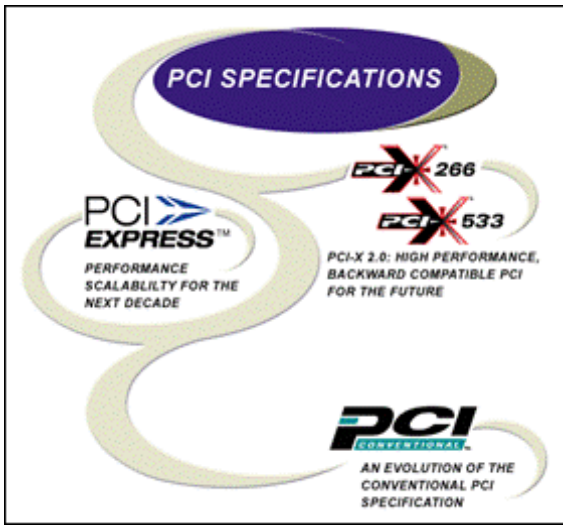


Fig. 172. The PCI bus is being refined by a Special Interest Group. You can follow their progress yourself on the Net (www.pcisig.com).

Plug and Play

The Plug and Play standard is part of the PCI specification. It means that all PCI adapter cards are *self-configuring*. The specification for Plug and Play was developed by Microsoft and Intel, among others, and the ideas was (as the name suggests) to provide a system where one can simply install an adapter and it will work. It's not quite this simple in practise; a software driver has to normally be installed before an adapter will work. But the actual cooperation between adapter, motherboard and operating system – happens automatically. During startup, communication takes place between the PC's startup programs, the PCI controller and each *PCI device* (adapter).

The adapter has to be able to inform the I/O bus which I/O addresses and IRQ's it can operate with. And it has to be able to configure itself to use the resources allocated to it by the I/O bus. When the exercise is successful, the adapter is configured automatically, and is ready to be used by the operating system.

All the components involved (adapter, motherboard and Windows) have to be Plug and Play compatible for the system to work.

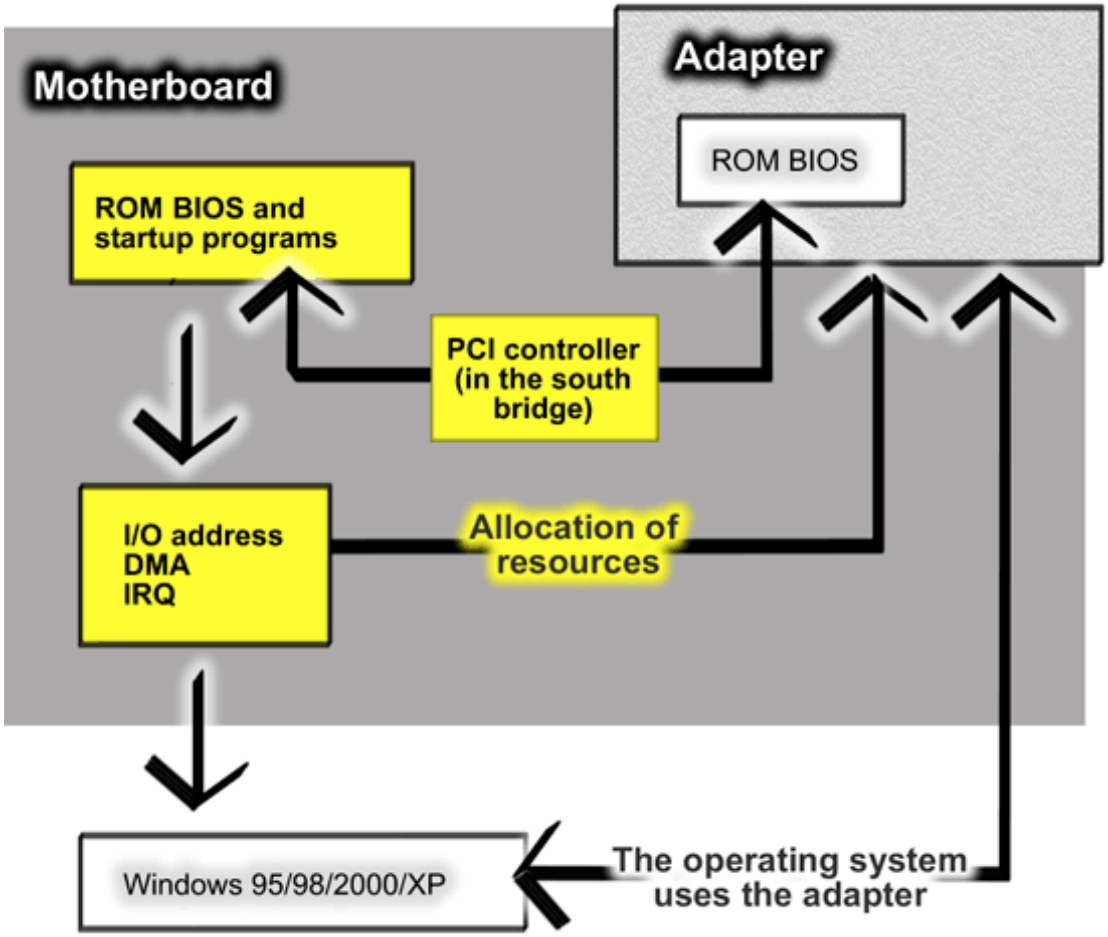


Fig. 173. Schematic overview of Plug and Play.

ESCD

In connection with Plug and Play, it is necessary also to discuss the *Extended System Configuration Data* (ESCD) system. This is a small data area which is stored in the motherboard's CMOS storage.

The ESCD store is used to save adapter configuration information. This means the motherboard doesn't have to go through the whole plug and play operation at each startup – information about the PC's configuration can be read from the CMOS storage.

The ESCD also allows the user to manually allocate an IRQ etc., for a particular adapter. This can be done using the motherboard's setup program. Read more about the CMOS on page 90.

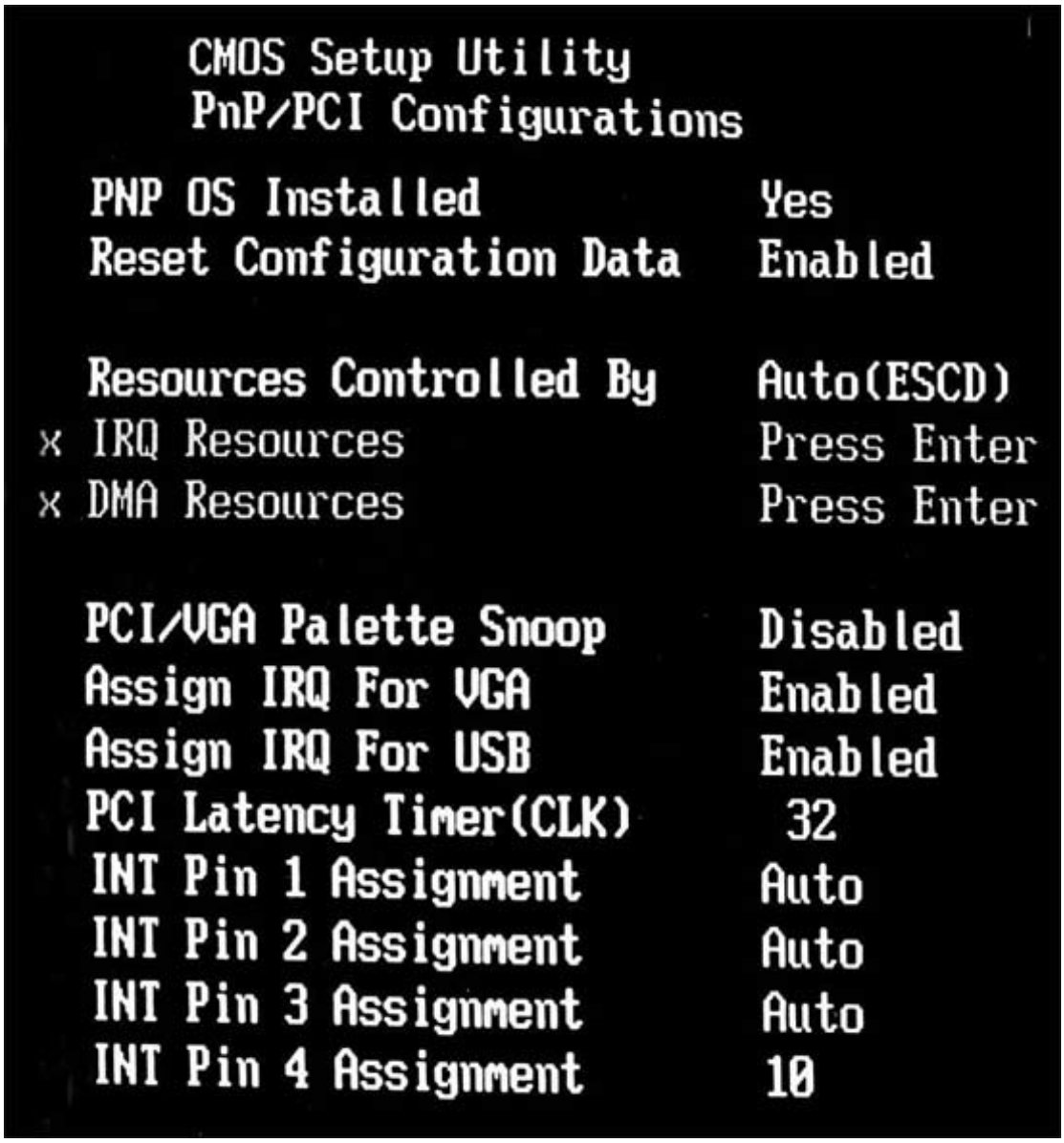


Fig. 174. Using the CMOS setup program, the user can directly allocate resources for PCI adapters.

See the devices during startup

All I/O devices have small, built-in registers (in ROM circuits), which contain various information. The registers can, for example, describe the nature of the device (e.g. a network card, video card or SCSI controller, etc.) and the manufacturer. You can see this information for yourself during PC startup, when the devices are configured:

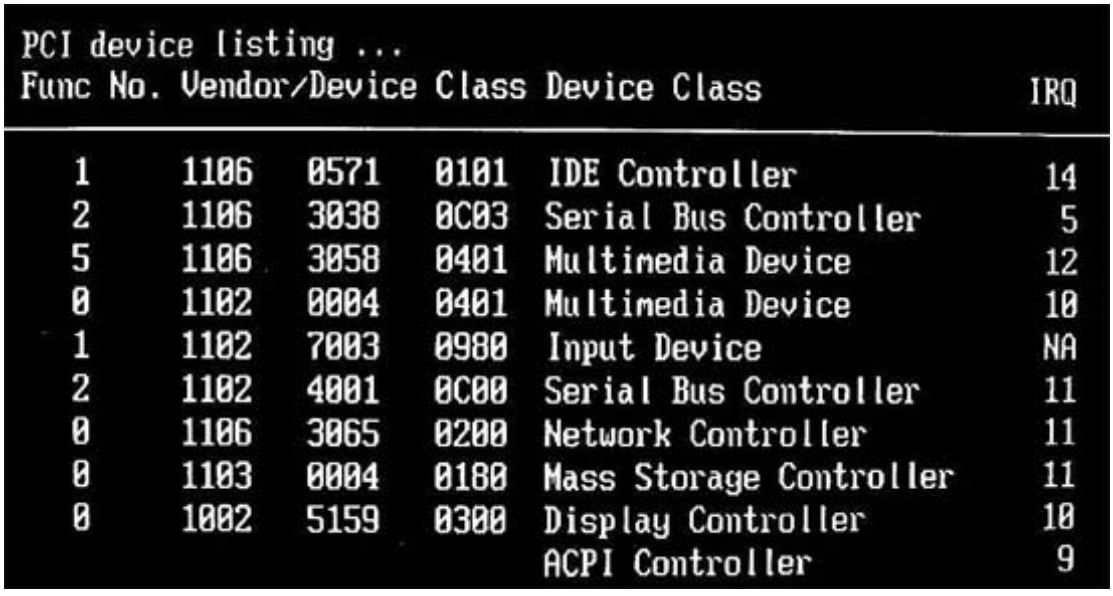


Fig. 175. The PCI devices ”identify themselves” during startup as they are configured.

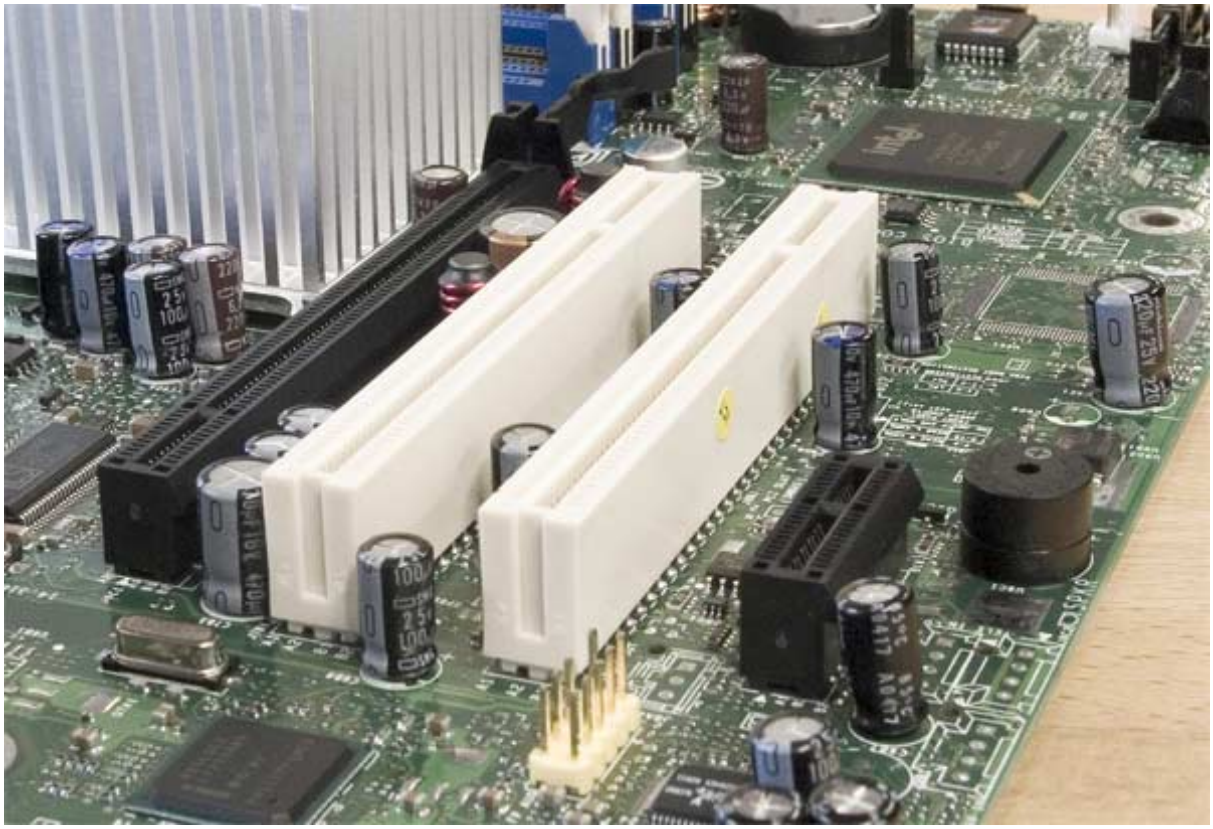
PCI Express development

In 2004 a new PCI bus was introduced. The PCI Special Interest Group (see www.pcisig.com) consists of the most important companies (Intel, IBM, Apple, etc.), who coordinate and standardize the bus via this forum. The PCI Express is the successor to the PCI bus. This is a completely new type of I/O bus using a serial connection (like the buses USB, Firewire and SATA).

This new I/O bus will be extremely scalable, as it works with a large number of channels (X1, X2, X16 etc.), each of which has a bandwidth of about 250 MB/second in both directions, simultaneously.

The standard plans for the use of plug-in cards and devices in various categories, with varying bandwidths and power consumption. A 16X video card, for example, will totally be able to pull about 8 GB/sec.

PCI Express is based on a serial architecture, making it possible to develop smaller and cheaper devices with many fewer pins. The new I/O bus will initially co-exist with the PCI interface, as we se it in the motherboards with Intel i900-series chip sets. But the goal is that PCI Express should replace both PCI and AGP.



Figur 176. The two black slots are for PCI Express cards. To the left a 16X card for the graphics controller and to the right a smaller 1X slot. Inbetween you se two traditional PCI slots.

- Next chapter.

- Next chapter.
- Previous chapter.

---

26. The CPU and the motherboard

The heart and soul of the PC's data processing is the CPU. But the processor is not alone in the world, it communicates with the rest of the motherboard. There will be many new terms introduced in the following sections, so remember that you can find definitions for all the abbreviations in the back of the guide.

Busses do the transfers

Data packets (of 8, 16, 32, 64 or more bits at a time) are constantly being moved back and forth between the CPU and all the other components (RAM, hard disk, etc.). These transfers are all done using *busses*.

The motherboard is designed around some vary powerful data *channels* (or pathways, as they are also called). It is these busses which connect all the components to each other.

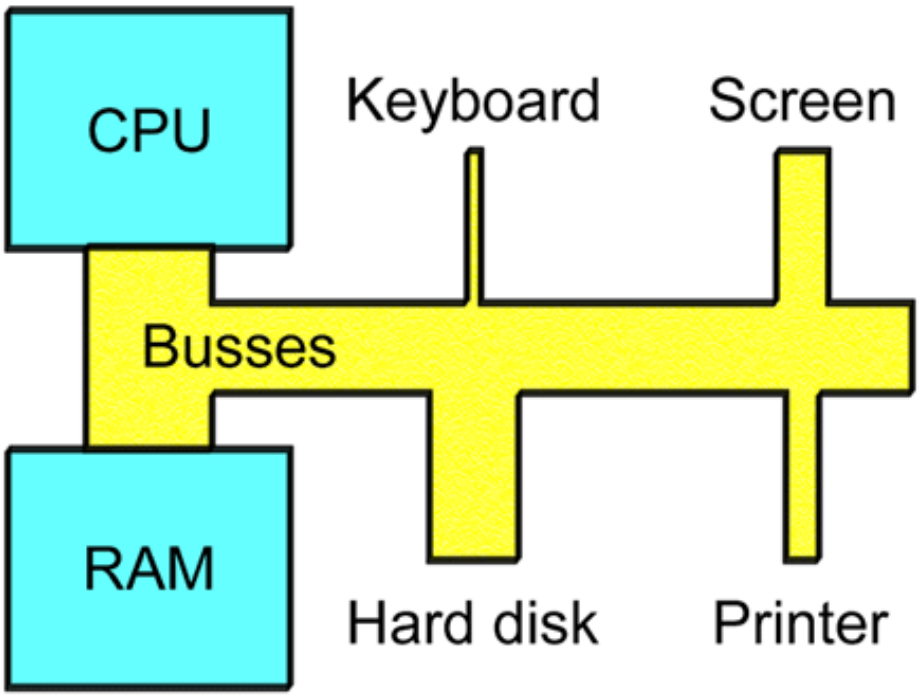


Figure 41. The busses are the data channels which connect the PC's components together. Some are designed for small transfers, others for large ones.

Busses with varying capacities

There is not just one bus on a motherboard; there are several. But they are all connected, so that data can run from one to another, and hence reach the farthest corners of the motherboard.

We can say that a bus system is subdivided into several branches. Some of the PC components work with enormous amounts of data, while others manage with much less. For example, the keyboard only sends very few bytes per second, whereas the working storage (RAM) can send and receive several gigabytes per second. So you can't attach RAM and the keyboard to the same bus.

Two busses with different capacities (bandwidths) can be connected if we place a controller between them. Such a controller is often called a *bridge*, since it functions as a bridge between the two different traffic systems.

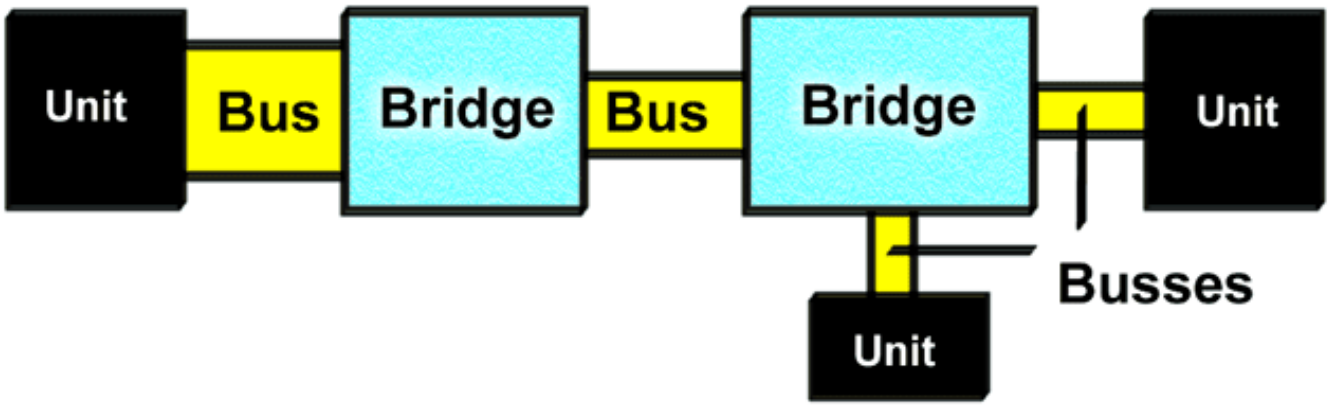


Figure 42. Bridges connect the various busses together.

The entire bus system starts close to the CPU, where the load (traffic) is greatest. From here, the busses work outwards towards the other components. Closest to the CPU we find the working storage. RAM is the component which has the very greatest data traffic, and is therefore connected directly to the CPU by a particularly powerful bus. It is called the *front side bus* (FSB) or (in older systems) *the system bus*.

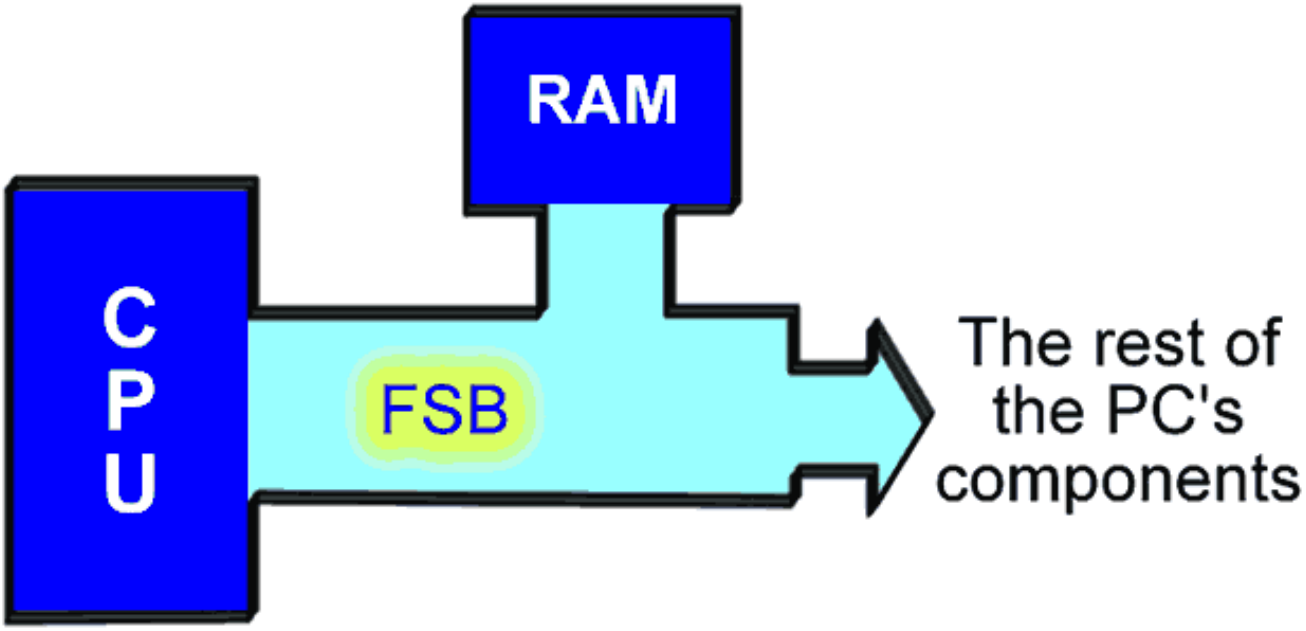


Figure 43. The PC’s most important bus looks after the “heavy” traffic between the CPU and RAM.

The busses connecting the motherboard to the PC’s peripheral devices are called *I/O busses*. They are managed by the controllers.

The chip set

The motherboard’s busses are regulated by a number of controllers. These are small circuits which have been designed to look after a particular job, like moving data to and from EIDE devices (hard disks, etc.).

A number of controllers are needed on a motherboard, as there are many different types of hardware devices which all need to be able to communicate with each other. Most of these controller functions are grouped together into a couple of large chips, which together comprise the *chip set*.

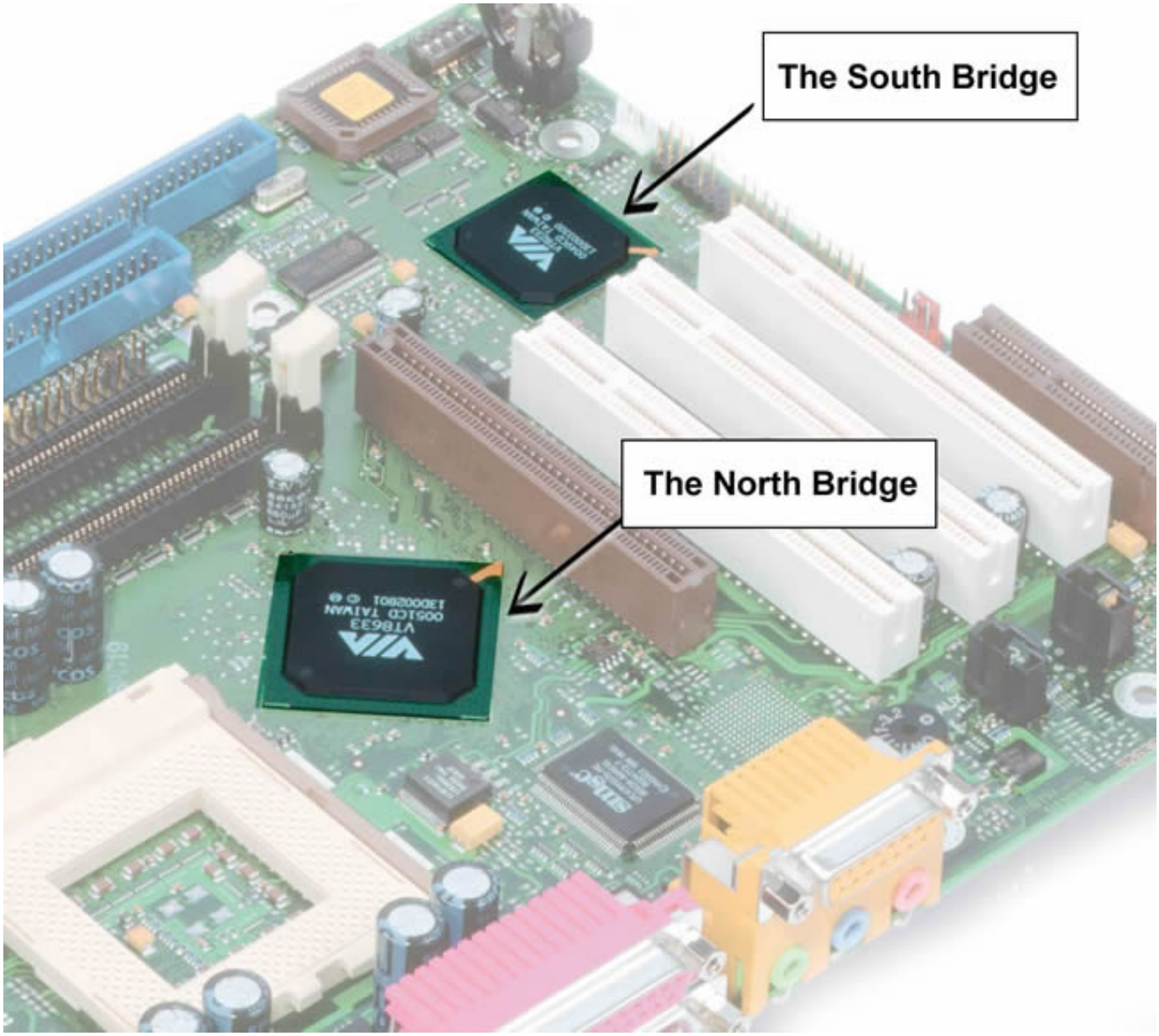


Figure 44. The two chips which make up the chipset, and which connect the motherboard's busses.

The most widespread chipset architecture consists of *two* chips, usually called the *north* and *south bridges*. This division applies to the most popular chipsets from VIA and Intel. The north bridge and south bridge are connected by a powerful bus, which sometimes is called a *link channel*:

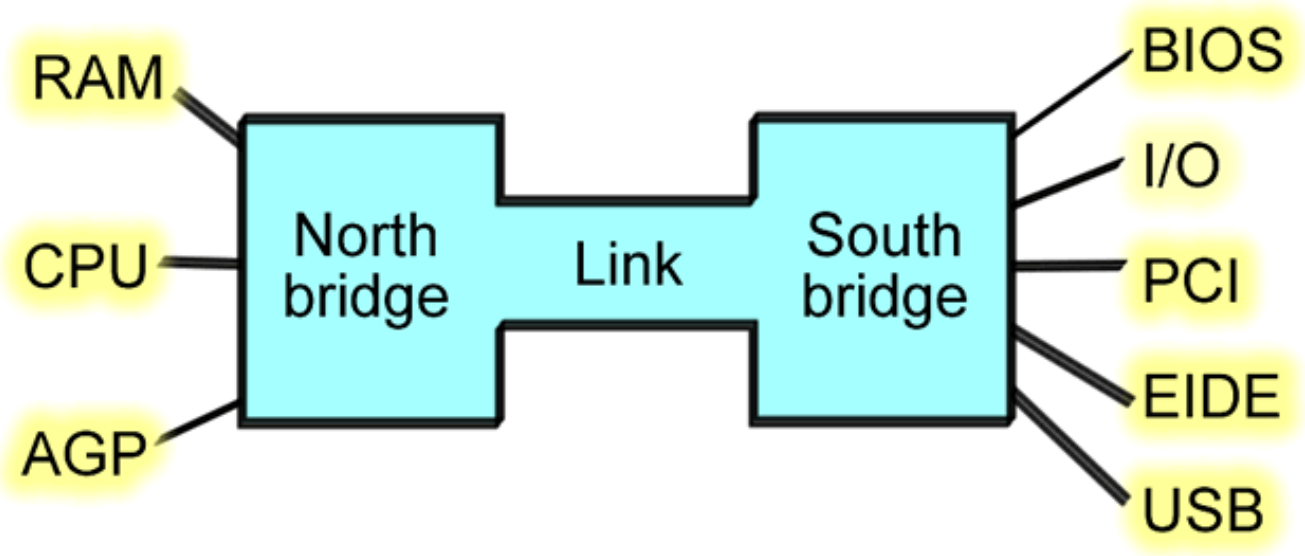


Figure 45. The north bridge and south bridge share the work of managing the data traffic on the motherboard.

The north bridge

The north bridge is a controller which controls the flow of data between the CPU and RAM, and to the AGP port.

In Fig. 46 you can see the north bridge, which has a large heat sink attached to it. It gets hot because of the often very large amounts of data traffic which pass through it. All around the north bridge you can see the devices it connects:

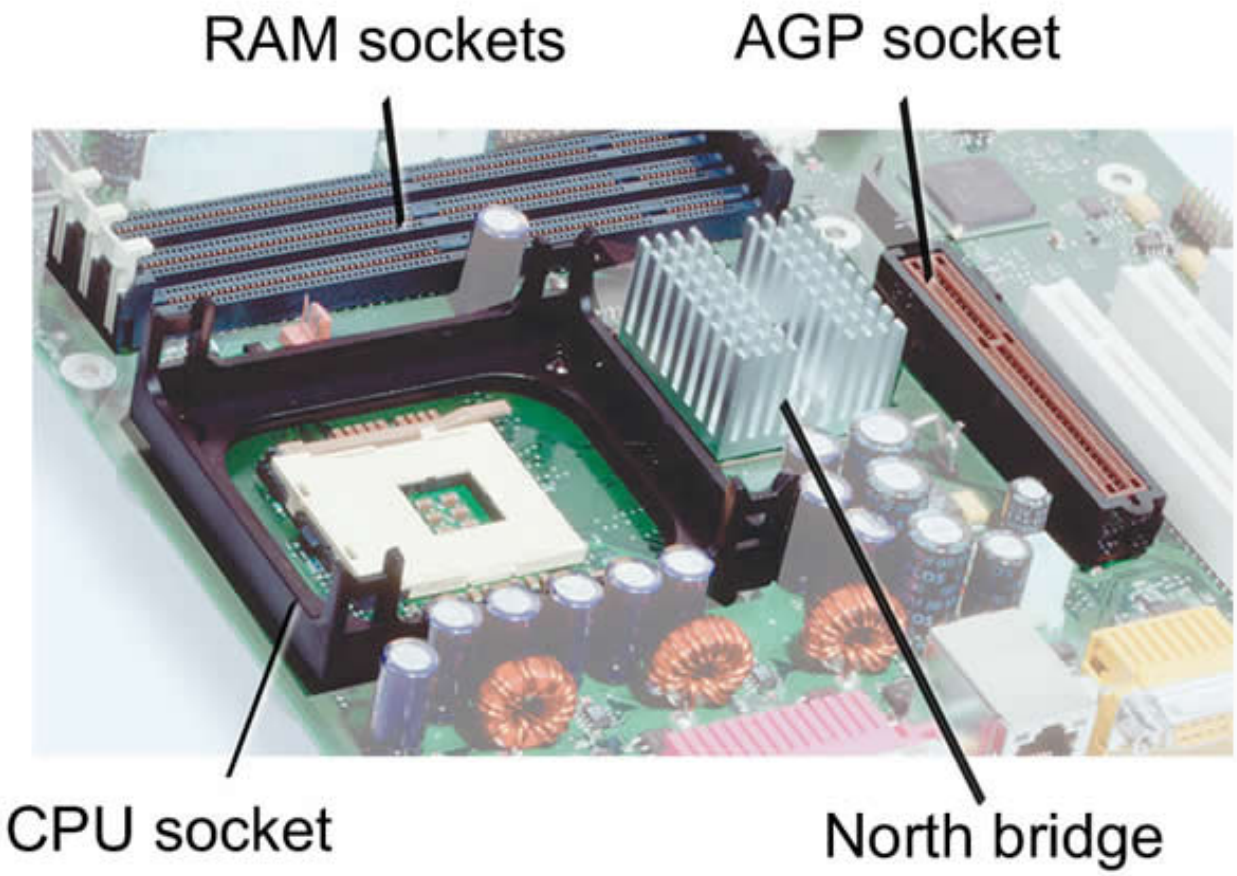


Figure 46. The north bridge and its immediate surroundings. A lot of traffic runs through the north bridge, hence the heat sink.

The AGP is actually an I/O port. It is used for the video card. In contrast to the other I/O devices, the AGP port is connected directly to the north bridge, because it has to be as close to the RAM as possible. The same goes for the PCI Express x16 port, which is the replacement of AGP in new motherboards. But more on that later.

The south bridge

The south bridge incorporates a number of different controller functions. It looks after the transfer of data to and from the hard disk and all the other I/O devices, and passes this data into the link channel which connects to the north bridge.

In Fig. 44 you can clearly see that the south bridge is physically located close to the PCI slots, which are used for I/O devices.

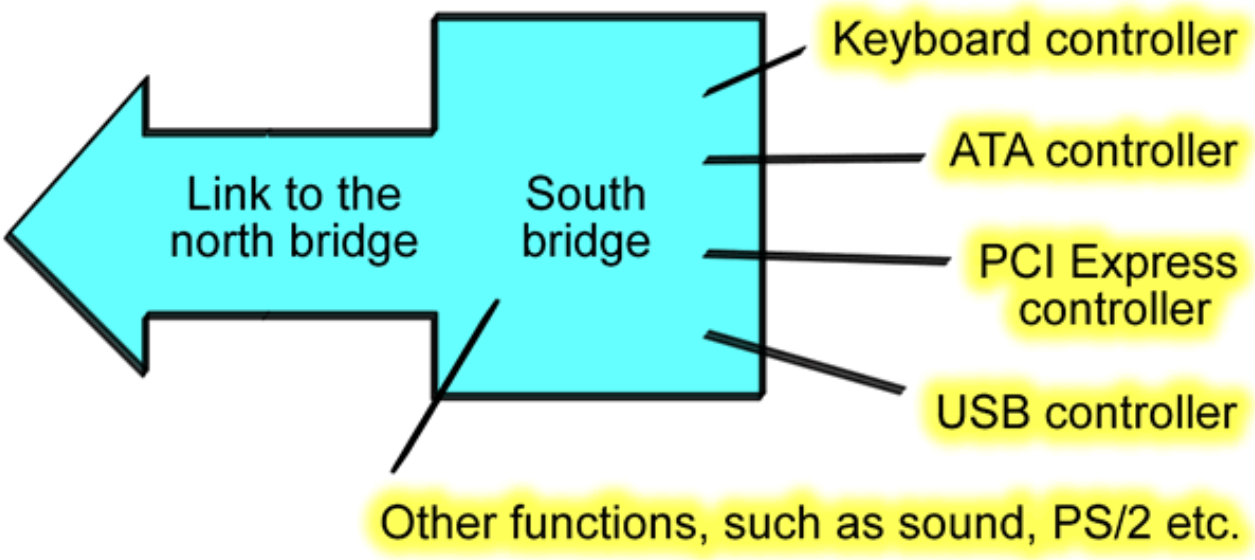


Figure 47. The chipset's south bridge combines a number of controller functions into a single chip.

The various chipset manufacturers

Originally it was basically only Intel who supplied the chipsets to be used in motherboards. This was quite natural, since Intel knows everything about their own CPU's and can therefore produce chipsets which match them. But at the time the Pentium II and III came out, other companies began to get involved in this market. The Taiwanese company, VIA, today produces chipsets for both AMD and Intel processors, and these are used in a large number of motherboards.

Other companies (like SiS, nVidia, ATI and ALi) also produce chipsets, but these haven't (yet?) achieved widespread use. The CPU manufacturer, AMD, produces some chipsets for their own CPU's, but they also work together closely with VIA as the main supplier for Athlon motherboards.

Figure 48. The Taiwanese company, VIA, has been a leader in the development of new chipsets in recent years.

Since all data transfers are managed by the chipset's two bridges, the chipset is the most important individual component on the motherboard, and new chipsets are constantly being developed.

The chipset determines the limits for clock frequencies, bus widths, etc. The chipset's built-in controllers are also responsible for connecting I/O devices like hard disks and USB ports, thus the chipset also determines, in practise, which types of devices can be connected to the PC.

Figure 49. The two chips which make up a typical chipset. Here we have VIA's model P4X266A, which was used in early motherboards for Pentium 4 processors.

Sound, network, and graphics in chipsets

Developments in recent years have led chipset manufacturers to attempt to place more and more functions in the chipset.

These extra functions are typically:

- Video card (integrated into the north bridge)
- Sound card (in the south bridge)
- Modem (in the south bridge)
- Network and Firewire (in the south bridge)

All these functions have traditionally been managed by separate devices, usually plug-in cards, which connect to the PC. But it has been found that these functions can definitely be incorporated into the chipset.

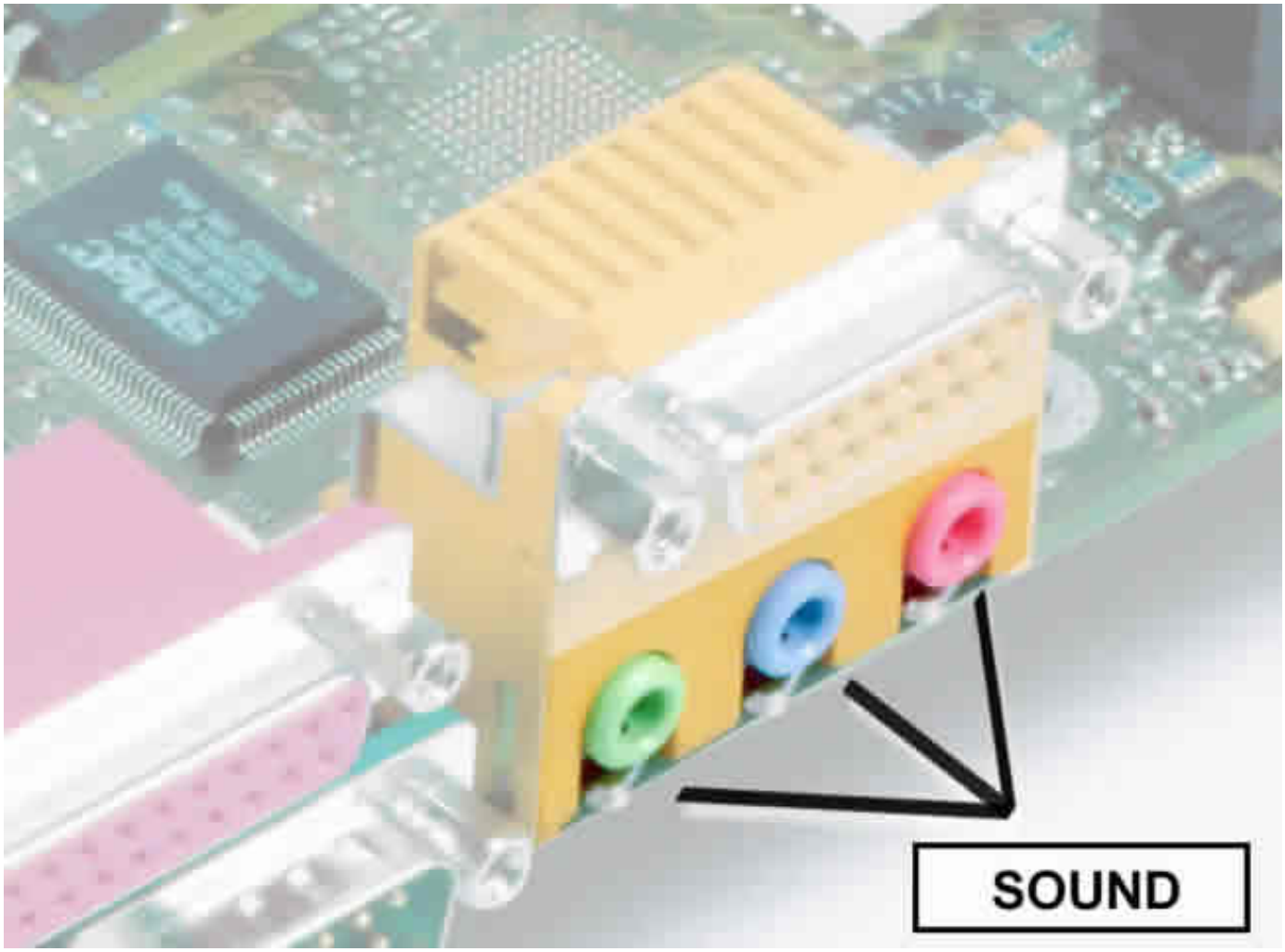


Figure 50. Motherboard with built-in sound functionality.

Intel has, for many years, managed to produce excellent network cards (Ethernet 10/100 Mbps); so it is only natural that they should integrate this functionality into their chipsets.

Sound facilities in a chipset cannot be compared with “real” sound cards (like, for example, Sound Blaster Audigy). But the sound functions work satisfactorily if you only want to connect a couple of small speakers to the PC, and don’t expect perfect quality.

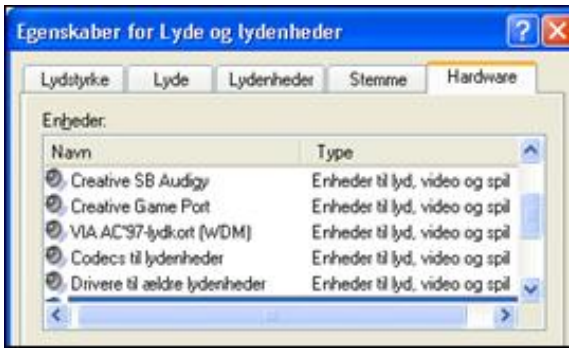


Figure 51. This PC has two sound cards installed, as shown in this Windows XP dialog box. The VIA AC'97 is a sound card emulation which is built into the chipset.

Many chipsets also come with a built-in video card. The advantage is clear; you can save having a separate video card, which can cost a $100 or more.

Again, the quality can't be compared with what you get with a separate, high quality, video card. But if you don't particularly need support for multiple screens, DVI (for flat screens), super 3D performance for games, or TV-out, the integrated graphics controller can certainly do the job.

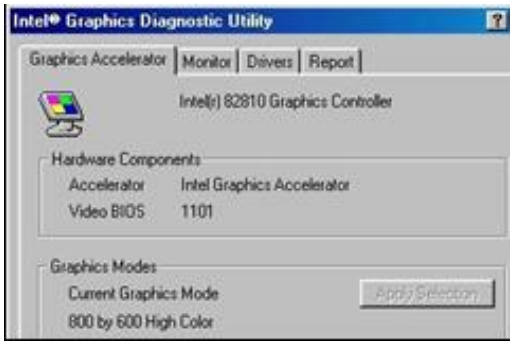


Figure 52. This PC uses a video card which is built into the Intel i810 chipset.

It is important that the integrated sound and graphics functions can be *disabled*, so that you can replace them with a real sound or video card. The sound functions won't cause any problems; you can always ask Windows to use a particular sound card instead of another one.

But the first Intel chipset with integrated graphics (the i810) did not allow for an extra video card to be installed. That wasn't very smart, because it meant users were locked into using the built-in video card. In the subsequent chipset (i815), the problem was resolved.

Buying a motherboard

If you want to build a PC yourself, you have to start by choosing a motherboard. It is the foundation for the entire PC.

Most of the motherboards on the market are produced in Taiwan, where manufacturers like Microstar, Asus, Epox, Soltek and many others supply a wide range of different models. Note that a producer like Microstar supplies motherboards to brand name manufacturers like Fujitsu-Siemens, so you can comfortably trust in the quality. Taiwan is the leader in the area of motherboards.

The first issue to work out is, which CPU you want to use. For example, if you want to use a Pentium 4 from Intel, there is one line of motherboards you can choose between. If you choose an AthlonXP, there is another line. And the difference lies in which chipset is being used in the motherboard.

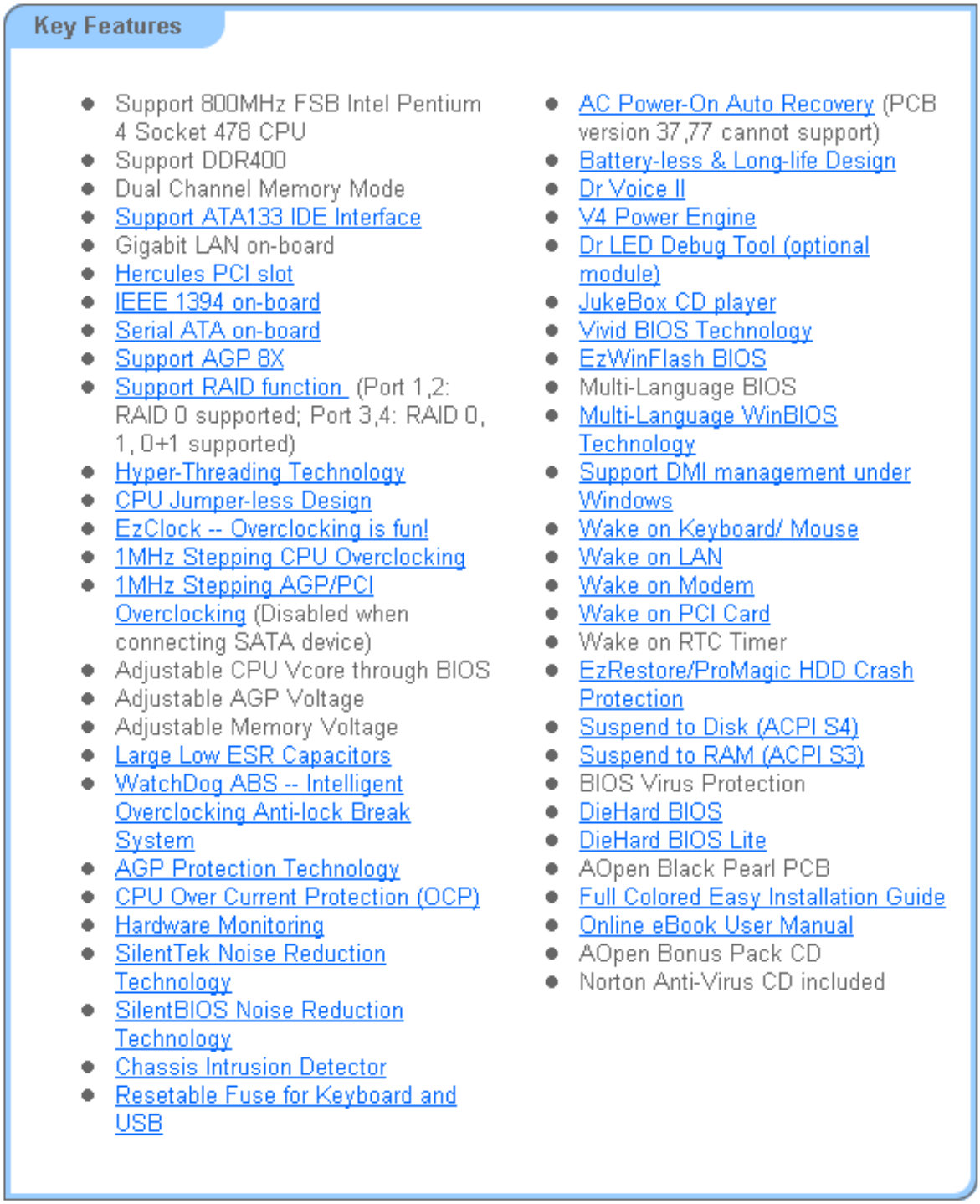


Figure 53. A typical (technical) advertisement for a motherboard.

Once you have decided on a processor, you should try to get a motherboard with the latest chipset available, because new versions of chipsets continue to be released, with greater functionality. At the time of writing, for example, chipsets often include these functions:

- USB version 2.0.
- Dual channel RAM.
- Support for the latest RAM like DDR2.
- Integrated Firewire ports.
- Serial ATA.
- Surround sound.
- Gigabit Ethernet.

You will most likely want to have these facilities (which are described later in the guide) on your PC. That is why it is important to choose the right motherboard with the latest generation chipset.

Extra facilities

In addition, it can sometimes be worth choosing a motherboard which has *extra facilities*. For example, all manufacturers have “luxury models” with built-in RAID controllers, making it possible to install several hard disks. There are motherboards around with loads of extra facilities, such as:

- Built-in RAID or (seldom) SCSI controller.
- Other network, screen and sound facilities.
- Wireless LAN.
- SmartCard/MemoryStick/etc. readers.

One of the advantages of building your own PC is that you can choose a really exciting motherboard.

Development is taking place rapidly, and by choosing the right motherboard, you can design the absolute latest PC on the market.

You can also find hundreds of articles on the Internet about each motherboard and chipset. So I can comfortably recommend you build your own PC, as long as you do your homework first! Make sure you read the rest of the guide before you start choosing a new motherboard!

- Next chapter.
- Previous chapter.

- Next chapter.
- Previous chapter.

Chapter 27. Inside and around the CPU

In this and the following chapters, I will focus on a detailed look at the CPU. One of the goals is help to you understand why manufacturers keep releasing new and more powerful processors. In order to explain that, we will have to go through what will at times be a quite detailed analysis of the CPU's inner workings.

Some of the chapters will probably be fairly hard to understand; I have spent a lot of time myself on my "research", but I hope that what I present in these chapters will shed some light on these topics.

Naturally, I will spend most of my time on the latest processors (the Athlon XP and Pentium 4). But we need to examine their internal architectures in light of the older CPU architectures, if we want to understand them properly. For this reason I will continually make comparisons across the various generations of CPU's.

Inside the CPU

I will now take you on a trip inside the CPU. We will start by looking at how companies like Intel and AMD can continue to develop faster processors.

Two ways to greater speed

Of course faster CPU's are developed as a result of hard work and lots of research. But there are two quite different directions in this work:

- More power and speed in the CPU, for example, from higher clock frequencies.
- Better exploitation of existing processor power.

Both approaches are used. It is a well-known fact that *bottlenecks* of various types drain the CPU of up to 75 % of its power. So if these can be removed or reduced, the PC can become significantly faster without having to raise the clock frequency dramatically.

It's just that it is very complicated to remove, for example, the bottleneck surrounding the front side bus, which I will show you later. So the manufacturers are forced to continue to raise the *working rate* (clock frequency), and hence to develop new *process technology*, so that CPU's with more power can come onto the market.

Clock frequencies

If we look at a CPU, the first thing we notice is the *clock frequency*. All CPU's have a working speed, which is regulated by a tiny *crystal*.

The crystal is constantly vibrating at a very large number of "beats" per second. For each clock tick, an impulse is sent to the CPU, and each pulse can, in principle, cause the CPU to perform one (or more) actions.

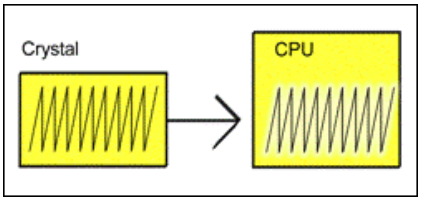


Figure 54. The CPU's working speed is regulated by a crystal which "oscillates" millions of times each second.

The number of clock ticks per second is measured in *Hertz*. Since the CPU's crystal vibrates millions of times each second, the clock speed is measured in *millions* of oscillations (*megahertz* or *MHz*). Modern CPU's actually have clock speeds running into *billions* of ticks per second, so we have started having to use *gigahertz (GHz)*.

These are unbelievable speeds. See for yourself how short the period of time is between individual clock ticks at these frequencies. We are talking about billionths of a second:

Clock frequency	**Time period per clock tick**
133 MHz	0.000 000 008 000 seconds
1200 MHz	0.000 000 000 830 seconds
2 GHz	0.000 000 000 500 seconds

Figure 55. The CPU works at an incredible speed.

The trend is towards ever increasing clock frequencies. Let's take a closer look at how this is possible.

More transistors

New types of processors are constantly being developed, for which the clock frequency keeps getting pushed up a notch. The original PC from 1981 worked at a modest 4.77 MHz, whereas the clock frequency 20 years later was up to 2 GHz.

In Fig. 56 you can see an overview of the last 20 years of development in this area. The table shows the seven *generations* of Intel processors which have brought about the PC revolution. The latest version of Pentium 4 is known under the code name Prescott.

Gen.	CPU	Yr (intr.)	Clock Frequency	No. of transistors
1	8088	1979	4.77- 8 MHz	29,000
2	80286	1982	6-12.5 MHz	134,000
3	80386	1985	16-33 MHz	275,000
4	80486	1989	25-100 MHz	1,200,000
5	Pentium Pentium MMX	1993 1997	60-200 MHz 166-300 MHz	3,100,000 4,500,000
6	Pentium Pro Pentium II Pentium III	1995 1997 1999	150-200 MHz 233-450 MHz 450-1200 MHz	5,500,000 7,500,000 28,000,000
7	Pentium 4 "Prescott"	2000 2002 2003 2004	1400-2200 2200-2800 2600-3200 2800-3600	42,000,000 55,000,000 55,000,000 125,000,000

Figure 56. Seven generations of CPU's from Intel. The number of transistors in the Pentium III and 4 includes the L2 cache.

Each processor has been on the market for several years, during which time the clock frequency has increased. Some of the processors were later released in improved versions with higher clock frequencies, I haven't included the Celeron in the overview processor. Celerons are specially discount versions of the Pentium II, III, and 4 processors.

Anyone can see that there has been an unbelievable development. Modern CPU's are one thousand times more powerful than the very first ones.

In order for the industry to be able to develop faster CPU's each year, new manufacturing methods are required. *More and more transistors* have to be squeezed into smaller and smaller chips.

Figure 57.
A photograph from one of Intel's factories, in which a technician displays the Pentium 4 processor core. It is a tiny piece of silicon which contains 42 million transistors.

Moores Law

This development was actually described many years ago, in what we call Moores Law.

Right back in 1965, Gordon Moore predicted (in the Electronics journal), that the number of transistors in processors (and hence their speed) would be able to be doubled every 18 months.

Moore expected that this regularity would at least apply up until 1975. But he was too cautious; we can see that the development continues to follow Moores Law today, as is shown in Fig. 59.

Figure 58. In 1968, Gordon Moore helped found Intel.

If we try to look ahead in time, we can work out that in 2010 we should have processors containing 3 billion transistors. And with what clock frequencies? You'll have to guess that for yourself.

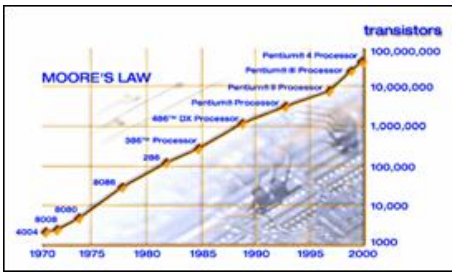


Figure 59. Moores Law (from Intels website).

Process technology

The many millions of transistors inside the CPU are made of, and connected by, ultra thin electronic *tracks*. By making these electronic tracks even narrower, even more transistors can be squeezed into a small slice of silicon.

The width of these electronic tracks is measured in *microns* (or *micrometers*), which are millionths of a metre.

For each new CPU generation, the track width is reduced, based on new technologies which the chip manufacturers keep developing. At the time of writing, CPU's are being produced with a track width of 0.13 microns, and this will be reduced to 0.09 and 0.06 microns in the next generations.

Figure 60. CPU's are produced in extremely high-technology environments ("clean rooms"). Photo courtesy of AMD.

In earlier generations aluminium was used for the current carrying tracks in the chips. With the change to 0.18 and 0.13-micron technology, aluminium began to be replaced with *copper*. Copper is cheaper, and it carries current better than aluminium. It had previously been impossible to insulate the copper tracks from the surrounding silicon, but IBM solved this problem in the late 1990's.

AMD became the first manufacturer to mass-produce CPU's with copper tracks in their chip factory *fab 30* in Dresden, Germany. A new generation of chips requires new chip factories (fabs) to produce it, and these cost billions of dollars to build. That's why they like a few years to pass between each successive generation. The old factories have to have time to pay for themselves before new ones start to be used.

Figure 61. AMD's Fab 30 in Dresden, which was the first factory to mass-produce copper-based CPU's.

A grand new world ...

We can expect a number of new CPU's in this decade, all produced in the same way as they are now – just with smaller track widths. But there is no doubt that we are nearing the physical limits for how small the transistors produced using the existing technology can be. So intense research is underway to find new materials, and it appears that nanotransistors, produced using organic (carbon-based) semiconductors, could take over the baton from the existing process technology.

Bell Labs in the USA has produced nanotransistors with widths of just one molecule. It is claimed that this process can be used to produce both CPU's and RAM circuits up to 1000 times smaller than what we have today!

Less power consumption

The types of CPU's we have today use a fairly large amount of electricity when the PC is turned on and is processing data. The processor, as you know, is installed in the motherboard, from which it receives power. There are actually two different voltage levels, which are both supplied by the motherboard:

- One voltage level which powers the CPU core (*kernel voltage*).
- Another voltage level which powers the CPU's I/O ports, which is typically 3.3 volts.

As the track width is reduced, more transistors can be placed within the same area, and hence the voltage can be reduced.

As a consequence of the narrower process technology, the *kernel voltage* has been reduced from 3 volts to about 1 volt in recent years. This leads to lower power consumption per transistor. But since the number of transistors increases by a corresponding amount in each new CPU generation, the end result is often that the total power consumption is unchanged.

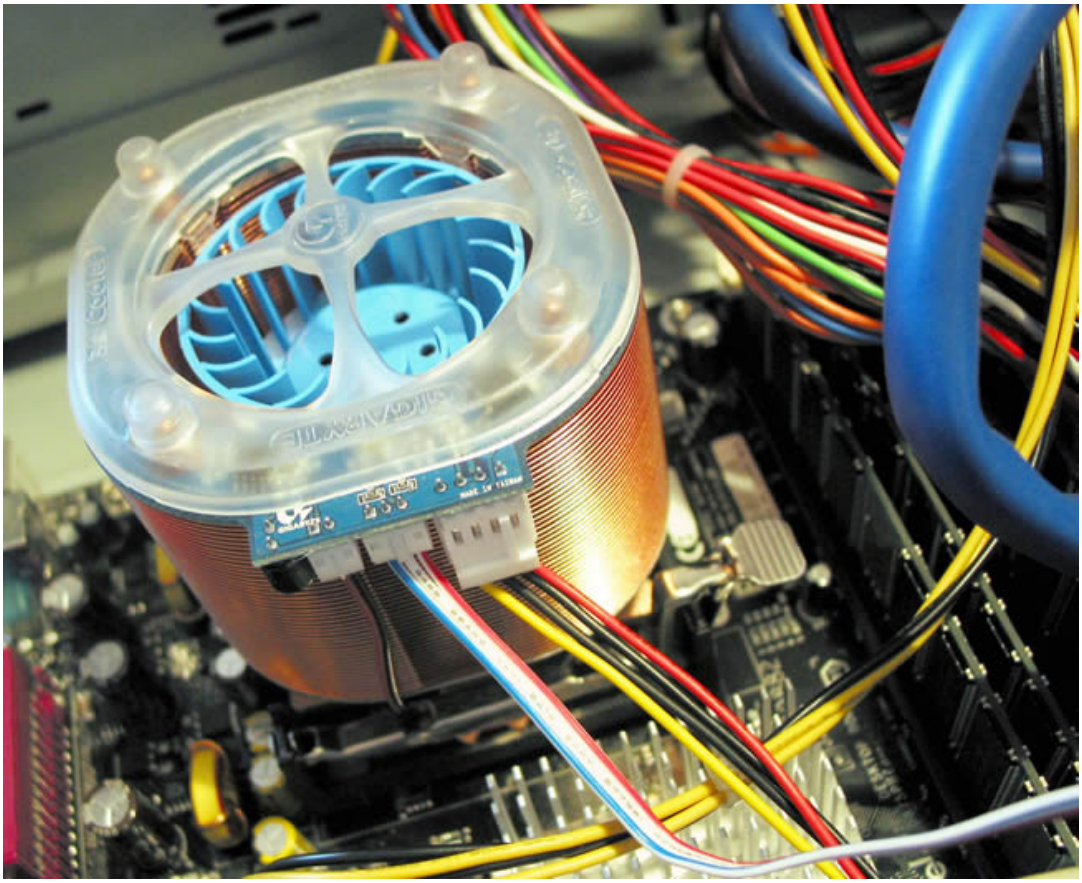
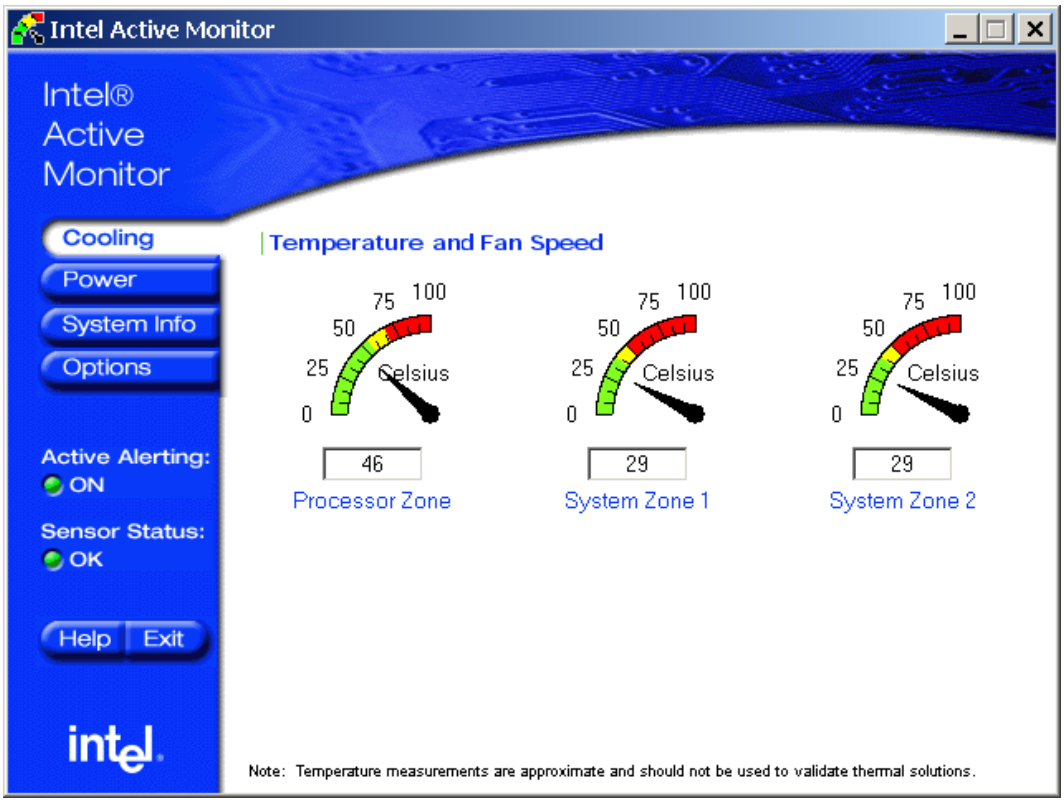


Figure 62. A powerful fan. Modern CPU's require something like this.

It is very important to cool the processor; a CPU can easily burn 50-120 Watts. This produces a fair amount of heat in a very small area, so without the right cooling fan and motherboard design, a Gigahertz processor could quickly burn out.

Modern processors contain a thermal diode which can raise the alarm if the CPU gets to hot. If the motherboard and BIOS are designed to pay attention to the diode's signal, the processor can be shut down temporarily so that it can cool down.



Figur Figure 63. The temperatures on the motherboard are constantly reported to this program..

Cooling is a whole science in itself. Many “nerds” try to push CPU’s to work at higher clock speeds than they are designed for. This is often possible, but it requires very good cooling – and hence often huge cooling units.

30 years development

Higher processor speeds require more transistors and narrower electronic tracks in the silicon chip. In the overview in Fig. 64 you can see the course of developments in this area.

Note that the 4004 processor was never used for PC’s. The 4004 was Intel’s first commercial product in 1971, and it laid the foundation for all their later CPU’s. It was a 4-bit processor which worked at 108 KHz (0.1 MHz), and contained 2,250 transistors. It was used in the first *pocket calculators*, which I can personally remember from around 1973-74 when I was at high school. No-one could have predicted that the device which replaced the slide rule, could develop, in just 30 years, into a Pentium 4 based super PC.

If, for example, the development in automobile technology had been just as fast, we would today be able to drive from Copenhagen to Paris in just 2.8 seconds!

Year	Intel CPU	Technology (track width)
1971	4004	10 microns
1979	8088	3 microns
1982	80286	1.5 microns
1985	80386	1 micron
1989	80486	1.0/0.8 microns
1993	Pentium	0.8/0.5/0.35 microns
1997	Pentium II	0.28/0.25 microns
1999	Pentium III	0.25/0.18/0.13 microns
2000-2003	Pentium 4	0.18/0.13 microns
2004-2005	Pentium 4 ”Prescott”	0.09 microns

Figure 64. The high clock frequencies are the result of new process technology with smaller electronic ”tracks”.

A conductor which is 0.09 microns (or 90 nanometres) thick, is 1150 times thinner than a normal human hair. These are tiny things we are talking about

here.

Wafers and die size

Another CPU measurement is its *die size*. This is the size of the actual silicon sheet containing all the transistors (the tiny area in the middle of Fig. 33 on page 15).

At the chip factories, the CPU cores are produced in so-called *wafers*. These are round silicon sheets which typically contain 150-200 processor cores (*dies*).

The smaller one can make each *die*, the more economical production can become. A big die is also normally associated with greater power consumption and hence also requires cooling with a powerful fan (e.g. see Fig. 63 on page 25 and Fig. 124 on page 50).

Figur Figure 65. A technician from Intel holding a wafer. This slice of silicon contains hundreds of tiny processor cores, which end up as CPU's in everyday PC's.

You can see the measurements for a number of CPU's below. Note the difference, for example, between a Pentium and a Pentium II. The latter is much smaller, and yet still contains nearly 2½ times as many transistors. Every reduction in die size is welcome, since the smaller this is, the more processors can fit on a wafer. And that makes production cheaper.

CPU	Track width	Die size	Number of transistors
Pentium	0.80	294 mm^2	3.1 mil.
Pentium MMX	0.28	140 mm^2	4.5 mil.
Pentium II	0.25	131 mm^2	7.5 mil.
Athlon	0.25	184 mm^2	22 mil.
Pentium III	0.18	106 mm^2	28 mil.
Pentium III	0.13	80 mm^2	28 mil.
Athlon XP	0.18	128 mm^2	38 mil.
Pentium 4	0.18	217 mm^2	42 mil.
Pentium 4	0.13	145 mm^2	55 mil.
Athlon XP+	0.13	115 mm^2	54 mil.
Athlon 64 FX	0,13	193 mm^2	106 mill.
Pentium 4	0.09	112 mm^2	125 mil.

Figure 66. The smaller the area of each processor core, the more economical chip production can be.

The modern CPU generations

As mentioned earlier, the various CPU's are divided into generations (see also Fig. 56 on page 23).

At the time of writing, we have started on the *seventh* generation. Below you can see the latest processors from Intel and AMD, divided into these generations. The transitions can be a bit hazy. For example, I'm not sure whether AMD's K6 belongs to the 5th or the 6th generation. But as a whole, the picture is as follows:

Generation	CPU's
5th	Pentium, Pentium MMX, K5, K6
6th	Pentium Pro, K6-II, Pentium II, K6-3, Athlon, Pentium III
7th	Pentium 4, Athlon XP
8th.	Athlon 64 FX, Pentium 5

Figure 67. The latest generations of CPU's.

- Next chapter.
- Previous chapter.

- Next chapter.
- Previous chapter.

Chapter 28. The cache

In the previous chapter, I described two aspects of the ongoing development of new CPU's – increased clock frequencies and the increasing number of transistors being used. Now it is time to look at a very different yet related technology – the processor's connection to the RAM, and the use of the *L1* and *L2 caches*.

Speed conflict

The CPU works internally at very high clock frequencies (like 3200 MHz), and no RAM can keep up with these.

The most common RAM speeds are between 266 and 533 MHz. And these are just a fraction of the CPU's working speed. So there is a great chasm between the machine (the CPU) which slaves away at perhaps 3200 MHz, and the "conveyor belt", which might only work at 333 MHz, and which has to ship the data to and from the RAM. These two subsystems are simply poorly matched to each other.

If nothing could be done about this problem, there would be no reason to develop faster CPU's. If the CPU had to wait for a bus, which worked at one sixth of its speed, the CPU would be *idle* five sixths of the time. And that would be pure waste.

The solution is to insert small, intermediate stores of high-speed RAM. These buffers (*cache* RAM) provide a much more efficient transition between the fast CPU and the slow RAM. Cache RAM operates at higher clock frequencies than normal RAM. Data can therefore be read more quickly from the cache.

Data is constantly being moved

The cache delivers its data to the CPU *registers*. These are tiny storage units which are placed right inside the processor core, and they are the absolute fastest RAM there is. The size and number of the registers is designed very specifically for each type of CPU.

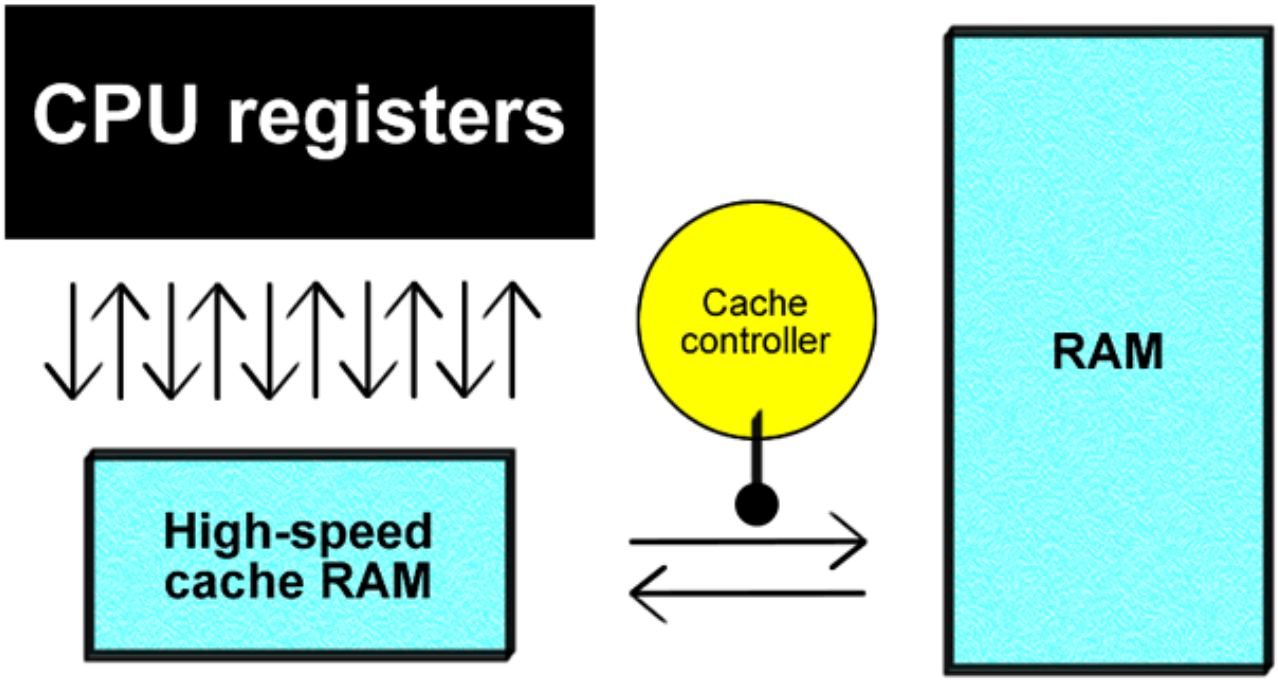


Figure 68. Cache RAM is much faster than normal RAM.

The CPU can move data in different sized packets, such as *bytes* (8 bits), *words* (16 bits), *dwords* (32 bits) or *blocks* (larger groups of bits), and this often involves the registers. The different data packets are constantly moving back and forth:

- from the CPU registers to the Level 1 cache.
- from the L1 cache to the registers.
- from one register to another
- from L1 cache to L2 cache, and so on...

The cache stores are a central bridge between the RAM and the registers which exchange data with the processor's *execution units*.

The optimal situation is if the CPU is able to constantly work and fully utilize all clock ticks. This would mean that the registers would have to always be able to fetch the data which the execution units require. But this it not the reality, as the CPU typically only utilizes 35% of its clock ticks. However, without a cache, this utilization would be even lower.

Bottlenecks

CPU caches are a remedy against a very specific set of "bottleneck" problems. There are lots of "bottlenecks" in the PC – transitions between fast and slower systems, where the fast device has to wait before it can deliver or receive its data. These bottle necks can have a very detrimental effect on the PC's total performance, so they must be minimised.

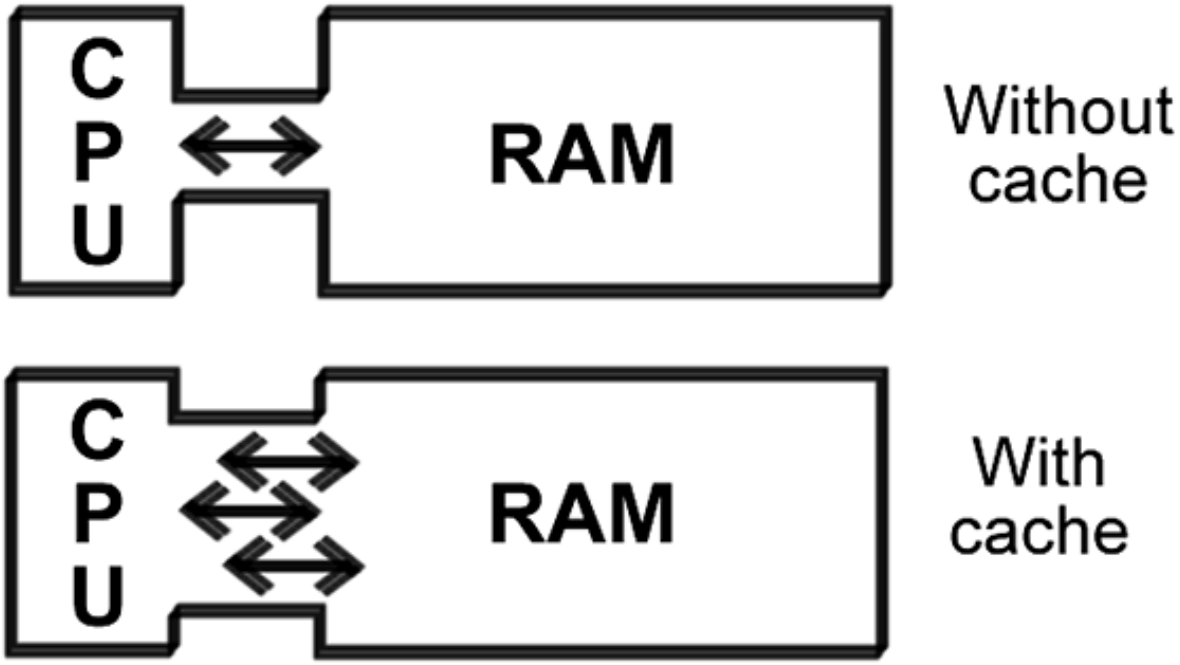


Figure 69. A cache increases the CPU's capacity to fetch the right data from RAM.

The absolute worst bottleneck exists between the CPU and RAM. It is here that we have the heaviest data traffic, and it is in this area that PC manufacturers are expending a lot of energy on new development. Every new generation of CPU brings improvements relating to the front side bus.

The CPU's cache is "intelligent", so that it can reduce the data traffic on the front side bus. The cache controller constantly monitors the CPU's work, and always tries to read in precisely the data the CPU needs. When it is successful, this is called a *cache hit*. When the cache does not contain the desired data, this is called a *cache miss*.

Two levels of cache

The idea behind cache is that it should function as a "near store" of fast RAM. A store which the CPU can always be supplied from.

In practise there are always at least two close stores. They are called *Level 1, Level 2,* and (if applicable) *Level 3* cache. Some processors (like the Intel Itanium) have three levels of cache, but these are only used for very special server applications. In standard PC's we find processors with L1 and L2 cache.

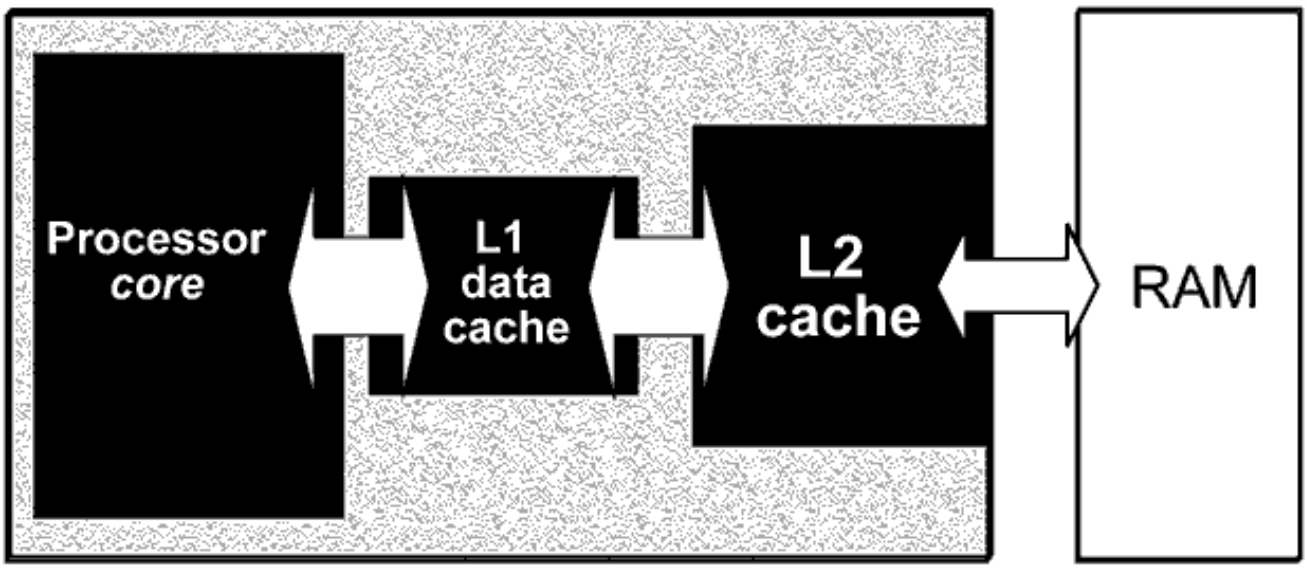


Figure 70. The cache system tries to ensure that relevant data is constantly being fetched from RAM, so that the

CPU (ideally) never has to wait for data.

L1 cache

Level 1 cache is built into the actual processor core. It is a piece of RAM, typically 8, 16, 20, 32, 64 or 128 Kbytes, which operates at the same clock frequency as the rest of the CPU. Thus you could say the L1 cache is part of the processor.

L1 cache is normally divided into two sections, one for *data* and one for *instructions*. For example, an Athlon processor may have a 32 KB data cache and a 32 KB instruction cache. If the cache is common for both data and instructions, it is called a *unified cache*.

L2 cache

The level 2 cache is normally much bigger (and unified), such as 256, 512 or 1024 KB. The purpose of the L2 cache is to constantly read in slightly larger quantities of data from RAM, so that these are available to the L1 cache.

In the earlier processor generations, the L2 cache was placed *outside* the chip: either on the motherboard (as in the original Pentium processors), or on a special module together with the CPU (as in the first Pentium II's).

Figure 71. An old Pentium II module. The CPU is mounted on a rectangular printed circuit board, together with the L2 cache, which is two chips here. The whole module is installed in a socket on the motherboard. But this design is no longer used.

As process technology has developed, it has become possible to make room for the L2 cache inside the actual processor chip. Thus the L2 cache has been integrated and that makes it function much better in relation to the L1 cache and the processor core.

The L2 cache is not as fast as the L1 cache, but it is still much faster than normal RAM.

CPU	L2 cache
Pentium, K5, K6	External, on the motherboard
Pentium Pro	Internal, in the CPU
Pentium II, Athlon	External, in a module close to the CPU
Celeron (1st generation)	None

Celeron (later gen.), Pentium III, Athlon XP, Duron, Pentium 4	Internal, in the CPU

Figure 72. It has only been during the last few CPU generations that the level 2 cache has found its place, integrated into the actual CPU.

Traditionally the L2 cache is connected to the front side bus. Through it, it connects to the chipset's north bridge and RAM:

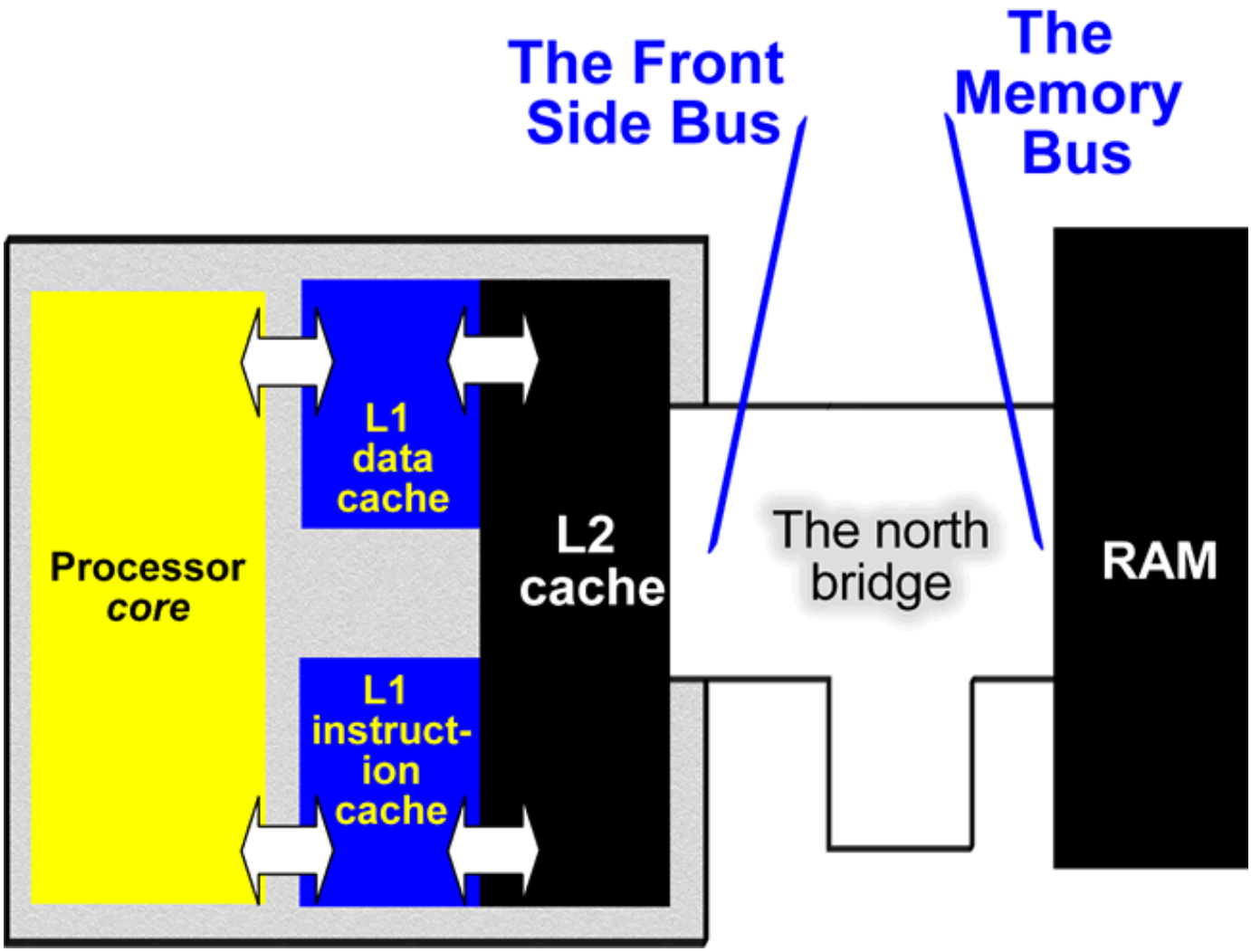


Figure 73. The way the processor uses the L1 and L2 cache has crucial significance for its utilisation of the high clock frequencies.

The level 2 cache takes up a lot of the chip's die, as millions of transistors are needed to make a large cache. The integrated cache is made using SRAM (*static* RAM), as opposed to normal RAM which is *dynamic* (DRAM).

While DRAM can be made using one transistor per bit (plus a capacitor), it costs 6 transistors (or more) to make one bit of SRAM. Thus 256 KB of L2 cache would require more than 12 million transistors. Thus it has only been since fine process technology (such as 0.13 and 0.09 microns) was developed that it became feasible to integrate a large L2 cache into the actual CPU. In Fig. 66 on page 27, the number of transistors includes the CPU's integrated cache.

Powerful bus

The bus between the L1 and L2 cache is presumably THE place in the processor architecture which has the greatest need for high bandwidth. We can calculate the theoretical maximum bandwidth by multiplying the bus width by the clock frequency. Here are some examples:

CPU	Bus width	Clock frequency	Theoretical bandwidth
Intel Pentium III	64 bits	1400 MHz	11.2 GB/sek.
AMD Athlon XP+	64 bits	2167 MHz	17.3 GB/sek.
AMD Athlon 64	64 bits	2200 MHz	17,6 GB/sek.
AMD Athlon 64 FX	128 bits	2200 MHz	35,2 GB/sek.
Intel Pentium 4	256 bits	3200 MHz	102 GB/sek.

Figure 74. Theoretical calculations of the bandwidth between the L1 and L2 cache.

Different systems

There are a number of different ways of using caches. Both Intel and AMD have saved on L2 cache in some series, in order to make cheaper products. But there is no doubt, that the better the cache – both L1 and L2 – the more efficient the CPU will be and the higher its performance.

AMD have settled on a fairly large L1 cache of 128 KB, while Intel continue to use relatively small (but efficient) L1 caches.

On the other hand, Intel uses a 256 bit wide bus on the “inside edge” of the L2 cache in the Pentium 4, while AMD only has a 64-bit bus (see Fig. 74).

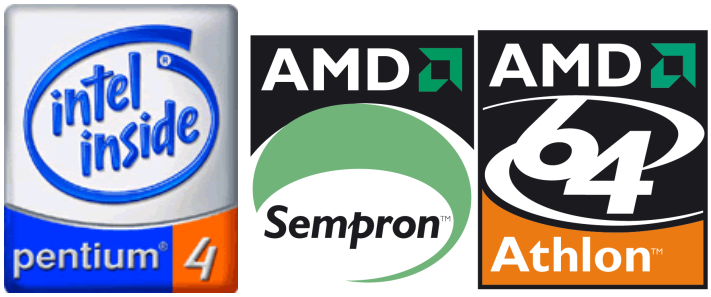


Figure 75. Competing CPU’s with very different designs.

AMD uses *exclusive caches* in all their CPU’s. That means that the same data can’t be present in both caches at the same time, and that is a clear advantage. It’s not like that at Intel.

However, the Pentium 4 has a more advanced cache design with *Execution Trace Cache* making up 12 KB of the 20 KB Level 1 cache. This instruction cache works with *coded instructions*, as described on page 35

CPU	L1 cache	L2 cache
Athlon XP	128 KB	256 KB
Athlon XP+	128 KB	512 KB
Pentium 4 (I)	20 KB	256 KB
Pentium 4 (II, “Northwood”)	20 KB	512 KB

Athlon 64	128 KB	512 KB
Athlon 64 FX	128 KB	1024 KB
Pentium 4 (III, “Prescott”)	28 KB	1024 KB

Figure 76. The most common processors and their caches.

Latency

A very important aspect of all RAM – cache included – is *latency*. All RAM storage has a certain latency, which means that a certain number of clock ticks (*cycles*) must pass between, for example, two reads. L1 cache has less latency than L2; which is why it is so efficient.

When the cache is bypassed to read directly from RAM, the latency is many times greater. In Fig. 77 the number of wasted clock ticks are shown for various CPU’s. Note that when the processor core has to fetch data from the actual RAM (when both L1 and L2 have failed), it costs around 150 clock ticks. This situation is called *stalling* and needs to be avoided.

Note that the Pentium 4 has a much smaller L1 cache than the Athlon XP, but it is significantly faster. It simply takes fewer clock ticks (*cycles*) to fetch data:

Latency	Pentium II	Athlon	Pentium 4
L1 cache:	3 cycles	3 cycles	2 cycles
L2 cache:	18 cycles	6 cycles	5 cycles

Figure 77. Latency leads to wasted clock ticks; the fewer there are of these, the faster the processor will appear to be.

Intelligent ”data prefetch”

In CPU’s like the Pentium 4 and Athlon XP, a handful of support mechanisms are also used which work in parallel with the cache. These include:

A *hardware auto data prefetch unit*, which attempts to *guess* which data should be read into the cache. This device monitors the instructions being processed and predicts what data the next job will need.

Related to this is the *Translation Look-aside Buffer,* which is also a kind of cache. It contains information which constantly supports the supply of data to the L1 cache, and this buffer is also being optimised in new processor designs. Both systems contribute to improved exploitation of the limited bandwidth in the memory system.

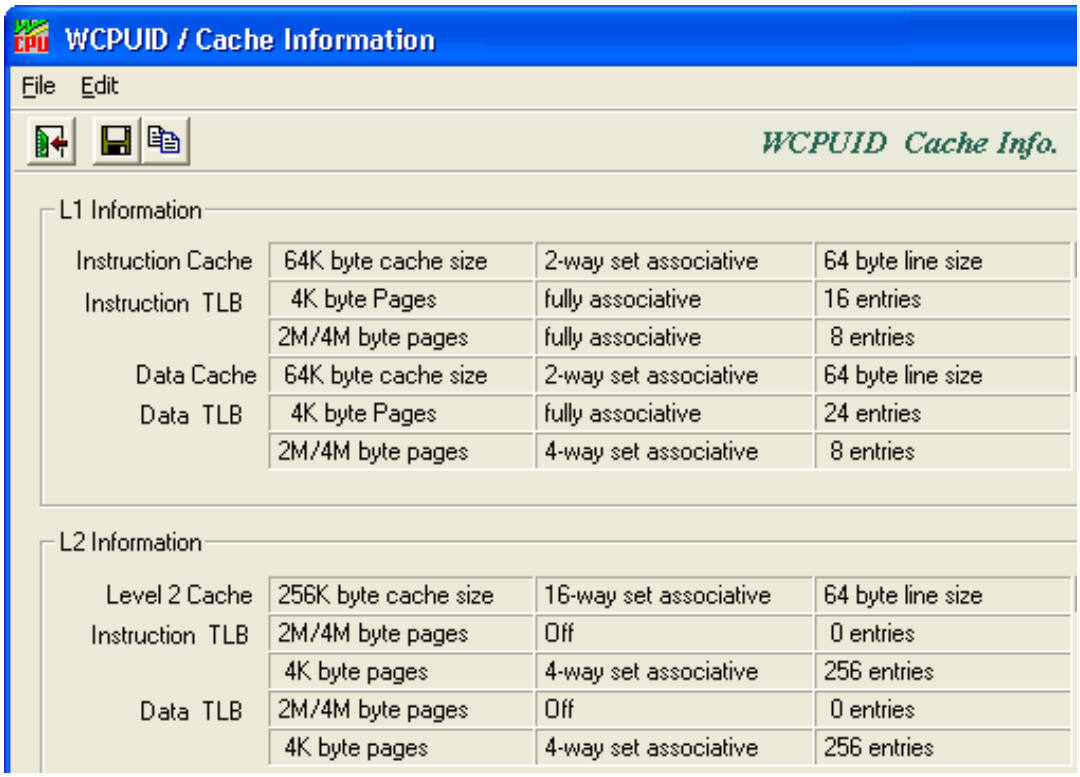


Figure 78. The WCPUID program reports on cache in an Athlon processor.

Conclusion

L1 and L2 cache are important components in modern processor design. The cache is crucial for the utilisation of the high clock frequencies which modern process technology allows. Modern L1 caches are extremely effective. In about 96-98% of cases, the processor can find the data and instructions it needs in the cache. In the future, we can expect to keep seeing CPU's with larger L2 caches and more advanced memory management. As this is the way forward if we want to achieve more effective utilisation of the CPU's clock ticks. Here is a concrete example:

In January 2002 Intel released a new version of their top processor, the Pentium 4 (with the codename, "Northwood"). The clock frequency had been increased by 10%, so one might expect a 10% improvement in performance. But because the integrated L2 cache was also doubled from 256 to 512 KB, the gain was found to be all of 30%.

CPU	L2 cache	Clock freq.	Improvement
Intel Pentium 4 (0.18 micron)	256 KB	2000 MHz	
Intel Pentium 4 (0.13 micron)	512 KB	2200 MHz	+30%

Figure 79. Because of the larger L2 cache, performance increased significantly.

In 2002 AMD updated the Athlon processor with the new "Barton" core. Here the L2 cache was also doubled from 256 to 512 KB in some models. In 2004 Intel came with the "Prescott" core with 1024 KB L2 cache, which is the same size as in AMD's Athlon 64 processors. Some *Extreme Editions* of Pentium 4 even uses 2 MB of L2 cache.

Xeon for servers

Intel produces special server models of their Pentium III and Pentium 4 processors. These are called Xeon, and are characterised by very large L2 caches. In an Intel Xeon the 2 MB L2 cache uses 149,000,000 transistors.

Xeon processors are incredibly expensive (about Euro 4,000 for the top models), so they have never achieved widespread distribution.

They are used in high-end servers, in which the CPU only accounts for a small part of the total price.

Otherwise, Intel's 64 bit server CPU, the Itanium. The processor is supplied in modules which include 4 MB L3 cache of 300 million transistors.

Multiprocessors

Several Xeon processors can be installed on the same motherboard, using special chipsets. By connecting 2, 4 or even 8 processors together, you can build a very powerful computer.

These MP (*Multiprocessor*) machines are typically used as servers, but can also be used as powerful workstations, for example, to perform demanding 3D graphics and animation tasks. AMD has the Opteron processors, which are server-versions of the Athlon 64. Not all software can make use of the PC's extra processors; the programs have to be designed to do so. For example, there are professional versions of Windows NT, 2000 and XP, which support the use of several processors in one PC.

See also the discussion of Hyper Threading which allows a Pentium 4 processor to appear as an MP system. Both Intel and AMD also works on dual-core processors.

- Next chapter.
- Previous chapter.

- Next chapter.
- Previous chapter.

---

29. Data and instructions

Now it's time to look more closely at the work of the CPU. After all, what does it actually do?

Instructions and data

Our CPU processes *instructions* and *data*. It receives orders from the software. The CPU is fed a gentle stream of binary data via the RAM.

These instructions can also be called *program code*. They include the commands which you constantly – via user programs – send to your PC using your keyboard and mouse. Commands to print, save, open, etc.

Data is typically *user data*. Think about that email you are writing. The actual contents (the text, the letters) is user data. But when you and your software say "send", your are sending program code (instructions) to the processor:

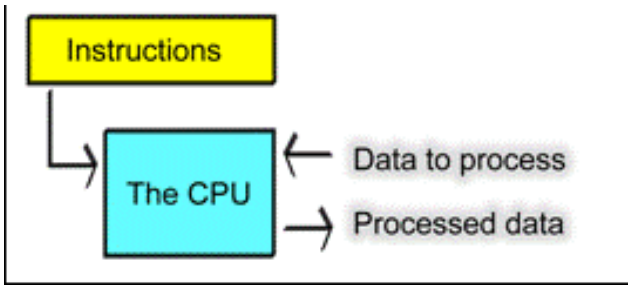


Figure 80. The instructions process the user data.

Instructions and compatibility

Instructions are *binary code* which the CPU can understand. Binary code (machine code) is the mechanism by which PC programs communicate with the processor.

All processors, whether they are in PC's or other types of computers, work with a particular *instruction set*. These instructions are the language that the CPU understands, and thus all programs have to communicate using these instructions. Here is a simplified example of some "machine code" – instructions written in the language the processor understands:

```
proc near

mov AX,01
mov BX,01
inc AX
add BX,AX
```

You can no doubt see that it wouldn't be much fun to have to use these kinds of instructions in order to write a program. That is why people use programming tools. Programs are written in a programming language (like *Visual Basic* or *C++*). But these program lines have to be translated into machine code, they have to be *compiled*, before they can run on a PC. The compiled program file contains instructions which can be understood by the particular processor (or processor family) the program has been "coded" for:

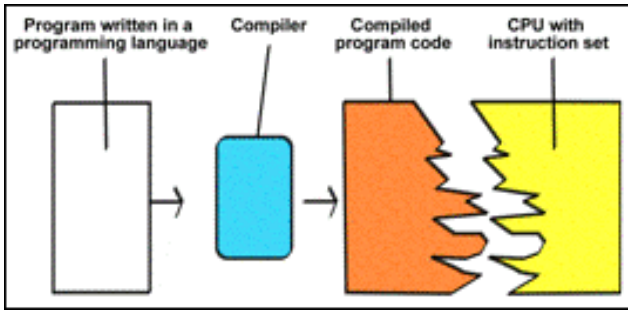


Figure 81. The program code produced has to match the CPU's instruction set. Otherwise it cannot be run.

The processors from AMD and Intel which we have been focusing on in this guide, are *compatible*, in that they understand the same instructions.

There can be big differences in the way two processors, such as the Pentium and Pentium 4, process the instructions *internally*. But externally – from the programmer's perspective – they all basically function the same way. All the processors in the PC family (regardless of manufacturer) can execute the same instructions and hence the same programs.

And that's precisely the advantage of the PC: Regardless of which PC you have, it can run the Windows programs you want to use.

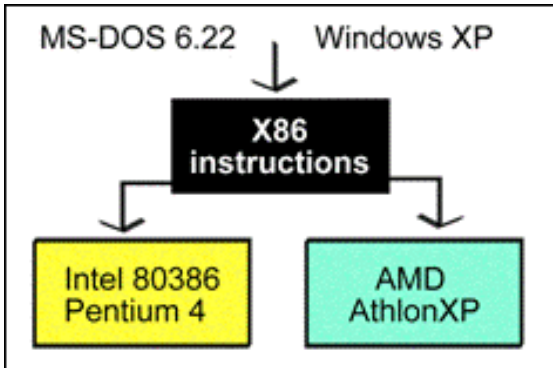


Figure 82. The x86 instruction set is common to all PC's.

As the years have passed, changes have been made in the instruction set along the way. A PC with a Pentium 4 processor from 2002 can handle very different applications to those which an IBM XT with an 8088 processor from 1985 can. But on the other hand, you can expect all the programs which could run on the 8088, to still run on a Pentium 4 and on a Athlon 64. The software is *backwards compatible*.

The entire software industry built up around the PC is based on the common *x86* instruction, which goes back to the earliest PC's. Extensions have been made, but the original instruction set from 1979 is still being used.

x86 and CISC

People sometimes differentiate between RISC and CISC based CPU's. The (x86) instruction set of the original Intel 8086 processor is of the CISC type, which stands for *Complex Instruction Set Computer.*

That means that the instructions are quite diverse and complex. The individual instructions vary in

length from 8 to 120 bits. It is designed for the 8086 processor, with just 29,000 transistors. The opposite of CISC, is RISC instructions.

RISC stands for *Reduced Instruction Set Computer*, which is fundamentally a completely different type of instruction set to CISC. RISC instructions can all have the same length (e.g. 32 bits). They can therefore be executed much faster than CISC instructions. Modern CPU's like the AthlonXP and Pentium 4 are based on a mixture of RISC and CISC.

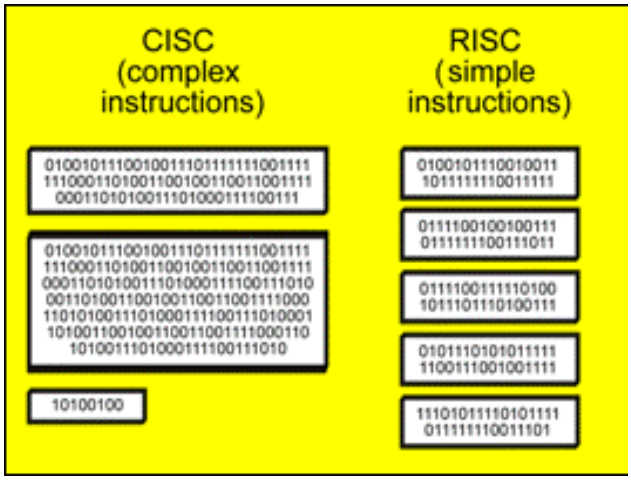


Figure 83. PC's running Windows still work with the old fashioned CISC instructions.

In order to maintain compatibility with the older DOS/Windows programs, the later CPU's still understand CISC instructions. They are just converted to shorter, more RISC-like, sub-operations (called *micro-ops*), before being executed. Most CISC instructions can be converted into 2-3 micro-ops.

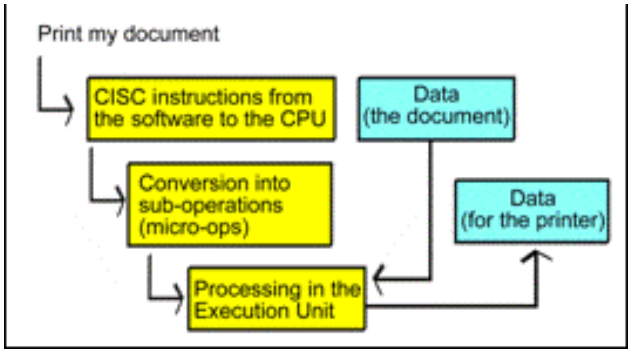


Figure 84. The CISC instructions are decoded before being executed in a modern processor. This preserves compatibility with older software.

Extensions to the instruction set

For each new generation of CPU's, the original instruction set has been extended. The 80386 processor added 26 new instructions, the 80486 added six, and the Pentium added eight new instructions.

At the same time, execution of the instructions was made more efficient. For example, it took an 80386 processor six clock ticks to add one number to a running summation. This task could be done in the 80486 (see page 40), in just two clock ticks, due to more efficient decoding of the instructions.

These changes have meant that certain programs require at least a 386 or a Pentium processor in order to run. This is true, for example, of all Windows programs. Since then, the MMX and SSE extensions have followed, which are completely new instruction sets which will be discussed later in the guide. They can make certain parts of program execution much more efficient.

Another innovation is the 64-bit extension, which both AMD and Intel use in their top-processors. Normally the pc operates in 32-bit mode, but one way to improve the performance is using a 64-bit mode. This requires new software, which is not available yet.

- Next chapter.
- Previous chapter.

---

30. Inside the CPU

Instructions have to be decoded, and not least, *executed*, in the CPU. I won't go into details on this subject; it is much too complicated. But I will describe a few factors which relate to the execution of instructions. My description has been extremely simplified, but it is relevant to the understanding of the microprocessor. This chapter is probably the most complicated one in the guide – you have been warned! It's about:

- Pipelines
- Execution units

If we continue to focus on speeding up the processor's work, this optimisation must also apply to the instructions – the quicker we can shove them through the processor, the more work it can get done.

Pipelines

As mentioned before, instructions are sent from the software and are broken down into *micro-ops* (smaller sub-operations) in the CPU. This decomposition and execution takes place in a *pipeline*.

The pipeline is like a reverse assembly line. The CPU's instructions are broken apart (decoded) at the start of the pipeline. They are converted into small sub-operations (*micro-ops*), which can then be processed one at a time in the rest of the pipeline:

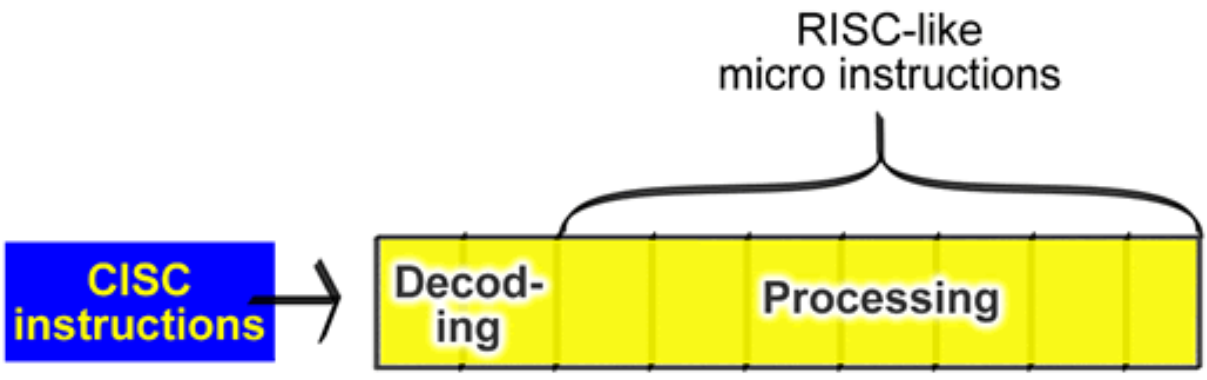


Figure 85. First the CISC instructions are decoded and converted into more digestible micro instructions. Then these are processed. It all takes place in the pipeline.

The pipeline is made up of a number *stages*. Older processors have only a few stages, while the newer ones have many (from 10 to 31). At each stage "something" is done with the instruction, and each stage requires one clock tick from the processor.

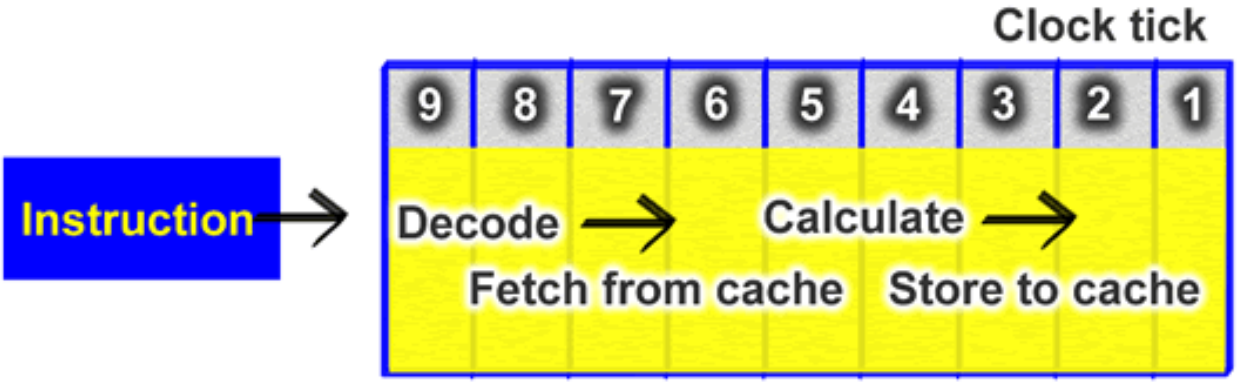


Figure 86. The pipeline is an assembly line (shown here with 9 stages), where each clock tick leads to the execution of a sub-instruction.

Modern CPU's have more than one pipeline, and can thus process several instructions at the same time. For example, the Pentium 4 and AthlonXP can decode about 2.5 instructions per clock tick.

The first Pentium 4 has several very *long* pipelines, allowing the processor to hold up to *126 instructions* in total, which are all being processed *at the same time*, but at different stages of execution (see Fig. 88. It is thus possible to get the CPU to perform more work by letting several pipelines work in parallel:

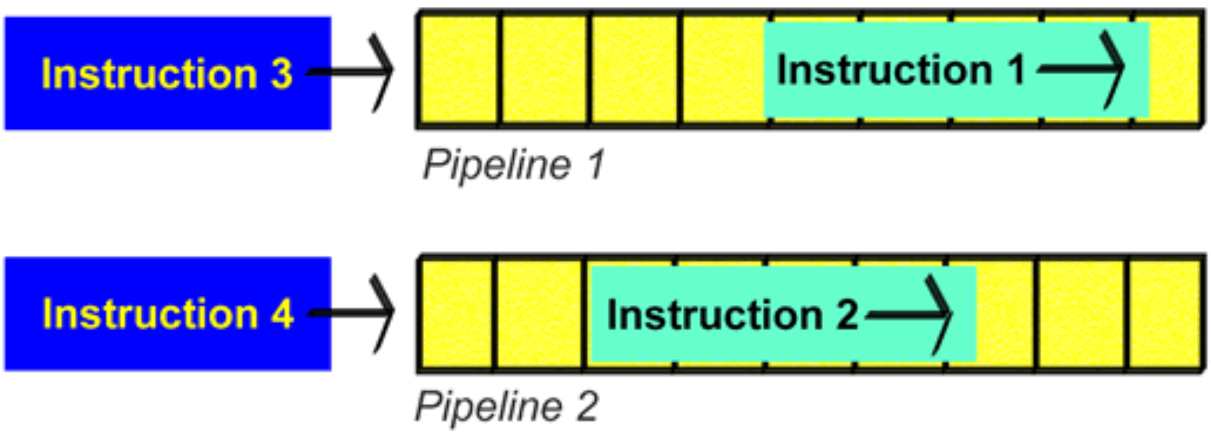


Figure 87. Having two pipelines allows twice as many instructions to be executed within the same number of clock ticks.

CPU	Instructions executed at the same time
AMD K6-II	24
Intel Pentium III	40
AMD Athlon	72

Intel Pentium 4 (first generation)	126

Figure 88. By making use of more, and longer, pipelines, processors can execute more instructions at the same time.

The problems of having more pipelines

One might imagine that the engineers at Intel and AMD could just make even more parallel pipelines in the one CPU. Perhaps performance could be doubled? Unfortunately it is not that easy.

It is not possible to feed a large number of pipelines with data. The memory system is just not powerful enough. Even with the existing pipelines, a fairly large number of clock ticks are wasted. The processor core is simply not utilised efficiently enough, because data cannot be brought to it quickly enough.

Another problem of having several pipelines arises when the processor can decode several instructions in parallel – each in its own pipeline. It is impossible to avoid the *wrong* instruction occasionally being read in (out of sequence). This is called *misprediction* and results in a number of wasted clock ticks, since another instruction has to be fetched and run through the “assembly line”.

Intel has tried to tackle this problem using a *Branch Prediction Unit*, which constantly attempts to guess the correct instruction sequence.

Length of the pipe

The number of “stations” (*stages*) in the pipeline varies from processor to processor. For example, in the Pentium II and III there are 10 stages, while there are up to 31 in the Pentium 4.

In the Athlon, the ALU pipelines have 10 stages, while the FPU/MMX/SSE pipelines have 15.

The longer the pipeline, the higher the processor’s clock frequency can be. This is because in the longer pipelines, the instructions are cut into more (and hence smaller) sub-instructions which can be executed more quickly.

CPU	Number of pipeline stages	Maximum clock frequency
Pentium	5	300 MHz
Motorola G4	4	500 MHz
Motorola G4e	7	1000 MHz
Pentium II and III	12	1400 MHz
Athlon XP	10/15	2500 MHz
Athlon 64	12/17	>3000 MHz

Pentium 4	20	>3000 MHz
Pentium 4 „Prescott“	31	>5000 MHz

Figure 89. Higher clock frequencies require long “assembly lines” (pipelines).

Note that the two AMD processors have different pipeline lengths for integer and floating point instructions. One can also measure a processor’s efficiency by looking at the IPC number (*Instructions Per Clock*), and AMD’s Athlon XP is well ahead of the Pentium 4 in this regard. AMD’s Athlon XP processors are actually much faster than the Pentium 4’s at equivalent clock frequencies.

The same is even more true of the Motorola G4 processors used, for example, in Macintosh computers. The G4 only has a 4-stage pipeline, and can therefore, in principle, offer the same performance as a Pentium 4, with only half the clock frequency or less. The only problem is, the clock frequency can’t be raised very much with such a short pipeline. Intel have therefore chosen to future-proof the Pentium 4 by using a very long pipeline.

Execution units

What is it that actually happens in the pipeline? This is where we find the so-called execution units. And we must distinguish between to types of unit:

- ALU (*Arithmetic and Logic Unit*)
- FPU (*Floating Point Unit*)

If the processor has a brain, it is the ALU unit. It is the calculating device that does operations on whole numbers (*integers*). The computer’s work with ordinary *text*, for example, is looked after by the ALU.

The ALU is good at working with whole numbers. When it comes to decimal numbers and especially numbers with many decimal places (*real* numbers as they are called in mathematics), the ALU chokes, and can take a very long time to process the operations. That is why an FPU is used to relieve the load. An FPU is a number cruncher, specially designed for *floating point* operations.

There are typically several ALU’s and FPU’s in the same processor. The CPU also has other operation units, for example, the LSU (*Load/Store Unit*).

An example sequence

Look again at Fig. 73. You can see that the processor core is right beside the L1 cache. Imagine that an instruction has to be processed:

- The processor core fetches a long and complex x86 instruction from the L1 instruction cache.
- The instruction is sent into the pipeline where it is broken down into smaller units.
- If it is an integer operation, it is sent to an ALU, while floating point operations are sent to an FPU.
- After processing the data is sent back to the L1 cache.

This description applies to the working cycle in, for example, the Pentium III and Athlon. As a diagram it might look like this:

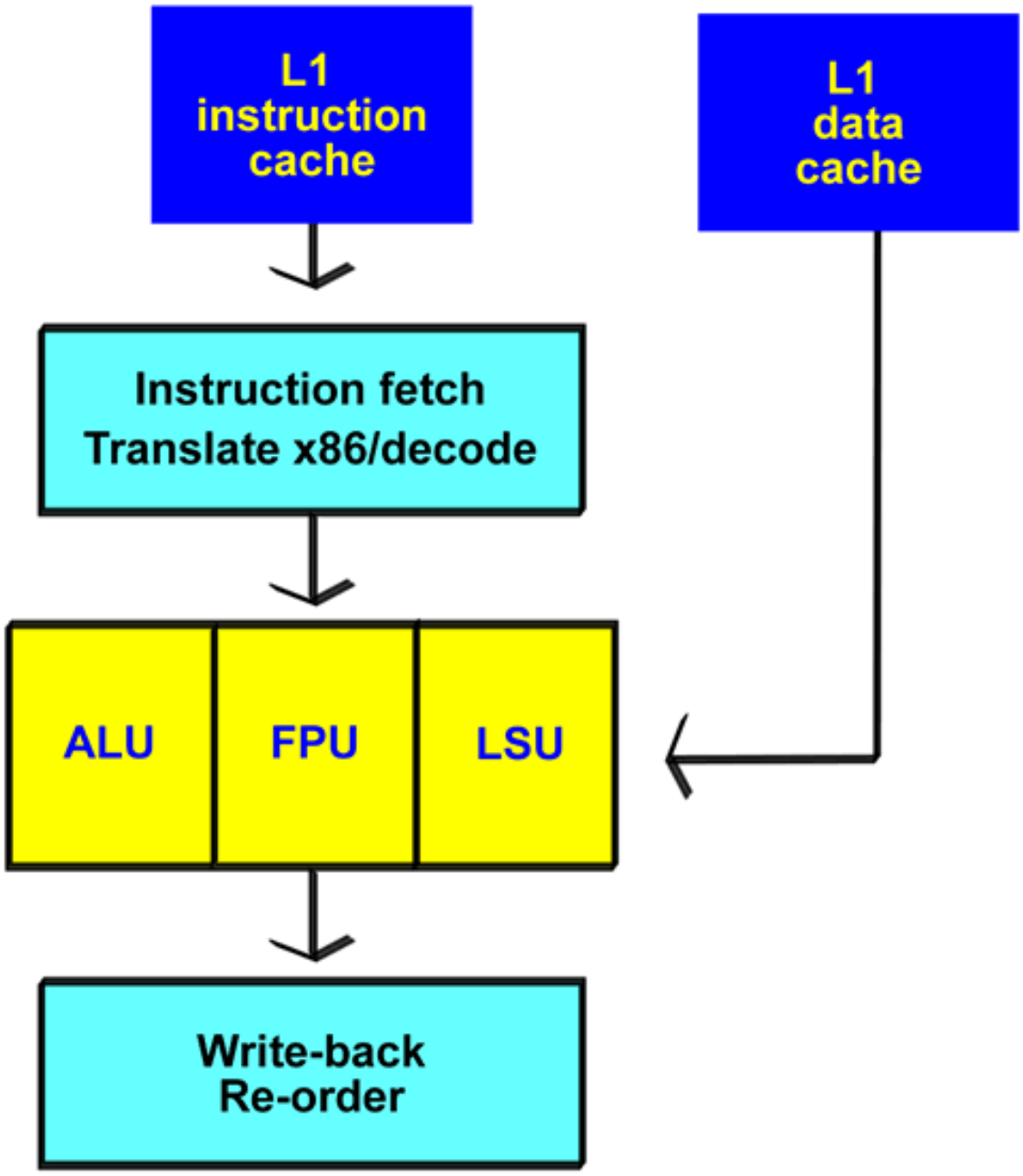


Figure 90. The passage of instructions through the pipeline.

But the way the relationship between the pipeline and the execution units is designed differs greatly from processor to processor. So this entire examination should be taken as a general introduction and nothing more.

Pipelines in the Pentium 4

In the Pentium 4, the instruction cache has been placed between the “Instruction fetch/Translate” unit (in Fig. 90 and the ALU/FPU. Here the instruction cache (*Execution Trace Cache*) doesn’t store the actual instructions, but rather the “half-digested” micro-ops.

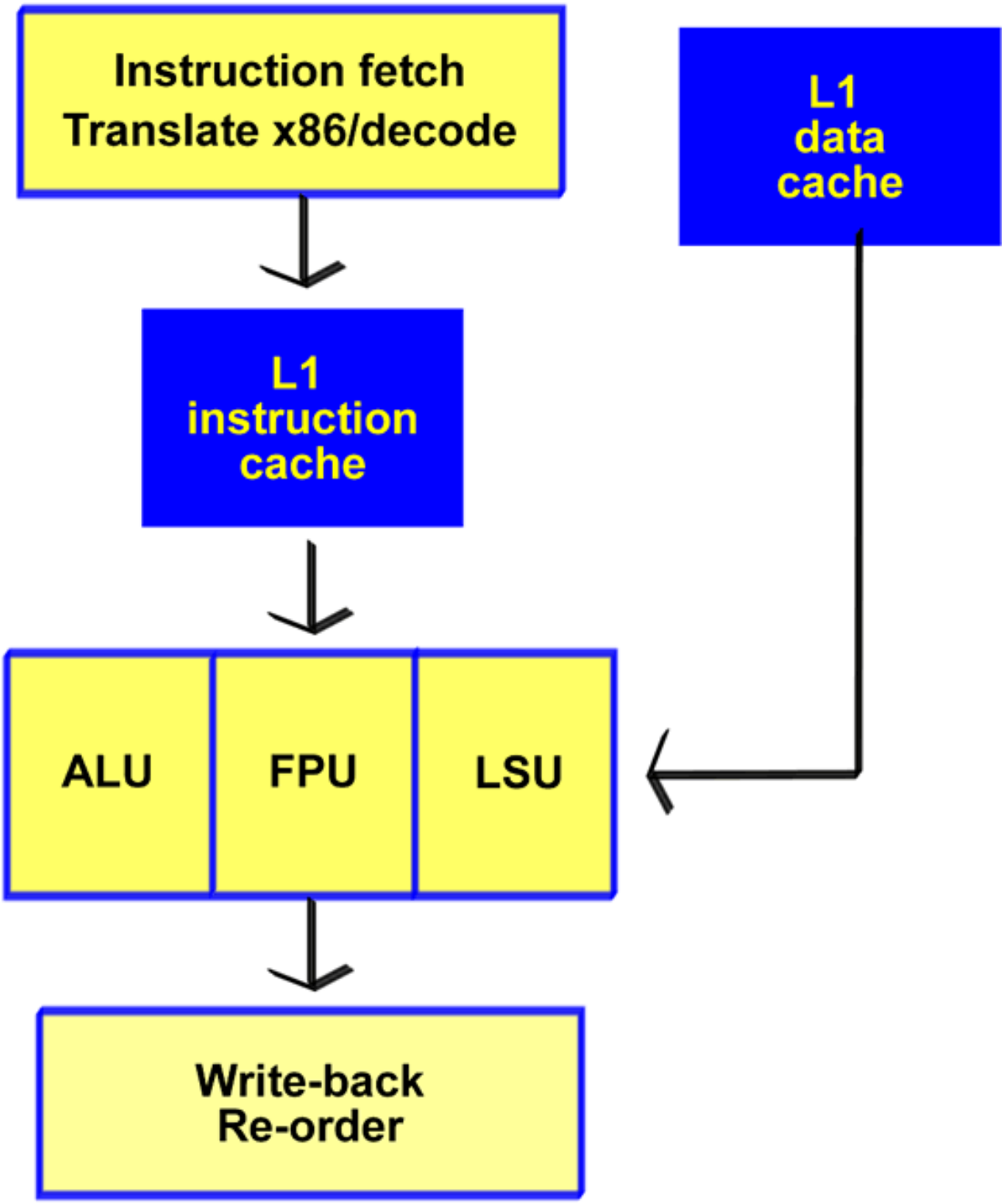


Figure 91. In the Pentium 4, the instruction cache stores decoded micro instructions.

The actual pipeline in the Pentium 4 is longer than in other CPU's; it has 20 stages. The disadvantage of the long pipeline is that it takes more clock ticks to get an instruction through it. 20 stages require 20 clock ticks, and that reduces the CPU's efficiency. This was very clear when the Pentium 4 was released; all tests showed that it was much slower than other processors with the same clock frequency.

At the same time, the cost of reading the wrong instruction (*misprediction*) is much greater – it takes a lot of clock ticks to fill up the long assembly line again.

The Pentium 4's architecture must therefore be seen from a longer-term perspective. Intel expects to be able to scale up the design to work at clock frequencies of up to 5-10 GHz. In the "Prescott" version of Pentium 4, the pipeline was increased further to 31 stages.

AMD’s 32 bit Athlon line can barely make it much above a clock frequency of 2 GHz, because of the short pipeline. In comparison, the Pentium 4 is almost ”light years” ahead.

- Next chapter.
- Previous chapter.

- Next chapter.
- Previous chapter.

Chapter 31. FPU's and multimedia

The computer is constantly performing *calculations*, which can be divided into two groups.

- Whole numbers
- Floating point numbers

The whole number calculations are probably the most important, certainly for normal PC use – using office programs and the like. But operations involving floating point numbers have taken on greater significance in recent years, as 3D games and sound, image and video editing have become more and more a part of everyday computing. Let's have a brief look at this subject.

Floating point numbers

The CPU has to perform a lot of calculations on decimal (or *real*) numbers when the PC runs 3D games and other multimedia programs.

These decimal numbers are processed in the CPU by a special unit called an FPU (*Floating Point Unit*). In case you are wondering (as I did) about the name, *floating point* – here is an explanation: Real numbers (like the number π , pi) can have an infinite number of decimal places. In order to work with these, often very large, numbers, they are converted into a special format.

First the required level of *precision* is set. Since the numbers can have an infinite number of decimals, they have to be rounded. For example, one might choose to have *five* significant digits, as I have done in the examples below (inside the PC, one would probably choose to have many more significant digits).

Once the precision has been set, the numbers are converted as shown below (the decimal point *floats*):

- The number 1,257.45 is written as 0.12575×10^4.
- The number 0.00696784 is written as 0.69678×10^{-2}.

Now the FPU can manage the numbers and process them using the arithmetic operators.

Normal form	Rewritten	In the FPU
1,257.45	0.12575×10^4	12575 +4
0.00696784	0.69678×10^{-2}	69678 -2

Figure 92. Re-writing numbers in floating point format.

FPU – the number cruncher

Floating point numbers are excessively difficult for the CPU's standard processing unit (the ALU) to process. A huge number of bits are required in order to perform a precise calculation. Calculations involving whole numbers (integers) are much simpler and the result is correct, every time.

That is why an FPU is used – a special calculating unit which operates with floating point numbers of various bit lengths, depending on how much precision is needed. FP numbers can be up to 80 bits long, whereas normal whole numbers can "only" be up to 32 bits (permitting 4,294 billion different numbers). So the FPU is a number cruncher, which relieves the load on the ALU's. You can experiment with large numbers yourself, for example, in a spreadsheet.

In Excel 2000, 2^{1023} (the number 2, multiplied by itself 1023 times), is the biggest calculation I can perform. The result is slightly less than 9 followed by 307 zeroes.

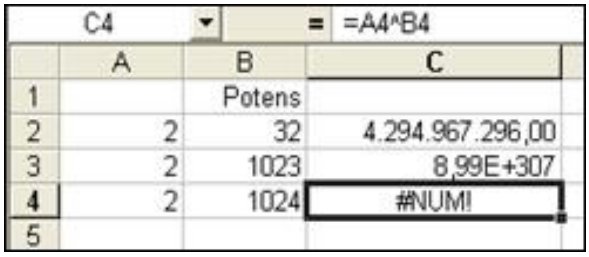


Figure 93. Experiments with big numbers in Excel.

Modern CPU's have a built-in FPU (*Floating Point Unit*) which serves as a number cruncher. But it hasn't always been like this.

For example, Intel's 80386 processor didn't have a built-in FPU calculating unit. All calculations were done using the processor's ALU. But you could buy a separate FPU (an *80387*), which was a chip which you mounted in a socket on the motherboard, beside the CPU. However, in the 80486 processor, the FPU was built-in, and it has been that way ever since.

Figure 94. A "separate" FPU, Intel's 80387 from 1986.

3D graphics

Much of the development in CPU's has been driven by 3D games. These formidable games (like *Quake* and others) place incredible demands on CPU's in terms of computing power. When these programs draw people and landscapes which can change in 3-dimensional space, the shapes are constructed from tiny *polygons* (normally triangles or rectangles).

Figure 95. The images in popular games like Sims are constructed from hundreds of polygons.

A character in a PC game might be built using 1500 such polygons. Each time the picture changes, these polygons have to be drawn again in a new position. That means that every corner (*vortex*) of every polygon has to be re-calculated.

In order to calculate the positions of the polygons, floating point numbers have to be used (integer calculations are not nearly accurate enough). These numbers are called *single-precision floating points* and are 32 bits long. There are also 64-bit numbers, called *double-precision floating points*, which can be used for even more demanding calculations.

When the shapes in a 3D landscape move, a so-called matrix multiplication has to be done to calculate the new vortexes. For just one shape, made up of, say, 1000 polygons, up to 84,000 multiplications have to be performed on pairs of 32-bit floating point numbers. And this has to happen for each new position the shape has to occupy. There might be 75 new positions per second. This is quite heavy computation, which the traditional PC is not very good at. The national treasury's biggest spreadsheet is child's play compared to a game like Quake, in terms of the computing power required.

The CPU can be left gasping for breath when it has to work with 3D movements across the screen. What can we do to help it? There are several options:

-
- Generally faster CPUs. The higher the clock frequency, the faster the traditional FPU performance will become.

- Improvements to the CPU's FPU, using more pipelines and other forms of acceleration. We see this in each new generation of CPU's.

- New *instructions* for more efficient 3D calculations.

We have seen that clock frequencies are constantly increasing in the new generations of CPU. But the FPU's themselves have also been greatly enhanced in the latest generations of CPU's. The Athlon, especially, is far more powerfully equipped in this area compared to its predecessors.

The last method has also been shown to be very effective. CPU's have simply been given new *registers* and new *instructions* which programmers can use.

MMX instructions

The first initiative was called MMX (*multimedia extension*), and came out with the Pentium MMX processor in 1997. The processor had built-in "MMX instructions" and "MMX registers".

The previous editions of the Pentium (like the other 32 bit processors) had two types of register: One

for 32-bit integers, and one for 80-bit decimal numbers. With MMX we saw the introduction of a special 64-bit integer register which works in league with the new MMX instructions. The idea was (and is) that multimedia programs should exploit the MMX instructions. Programs have to be “written for” MMX, in order to utilise the new system.

MMX is an *extension* to the existing instruction set (*IA32*). There are 57 new instructions which MMX compatible processors understand, and which require new programs in order to be exploited.

Many programs were rewritten to work both with and without MMX (see Fig 96). Thus these programs could continue to run on older processors, without MMX, where they just ran slower.

MMX was a limited success. There is a weakness in the design in that programs either work with MMX, or with the FPU, and not both at the same time – as the two instruction sets share the same registers. But MMX laid the foundation for other multimedia extensions which have been much more effective.

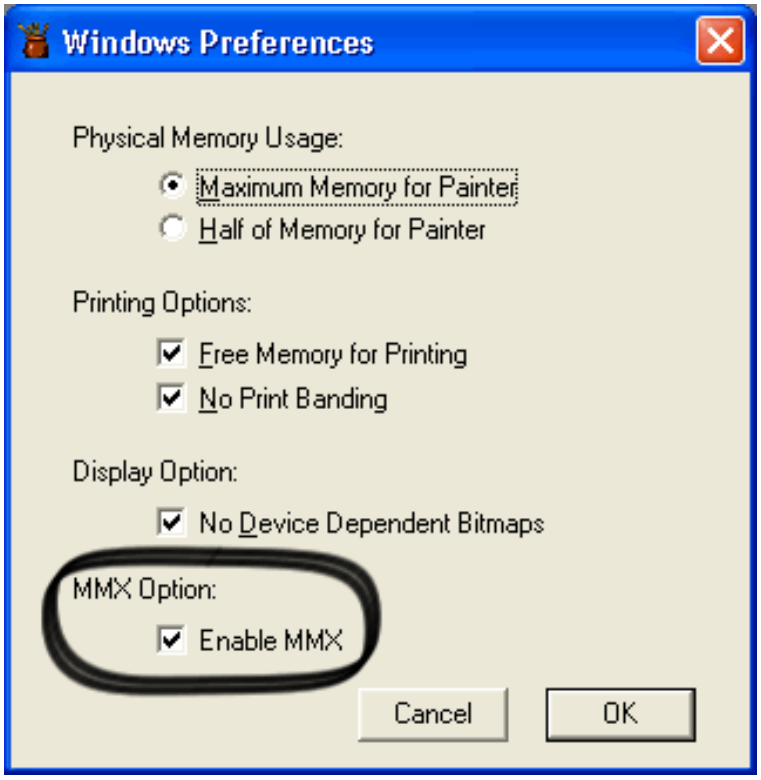


Fig Figure 96. This drawing program (Painter) supports MMX, as do all modern programs.

3DNow!

In the summer of 1998, AMD introduced a collection of CPU instructions which improved 3D processing. These were 21 new SIMD (*Single Instruction Multiple Data*) instructions. The new instructions could process several chunks of data with one instruction. The new instructions were marketed under the name, *3DNow!*. They particularly improved the processing of the 32-bit floating point numbers used so extensively in 3D games.

Figure 97. 3DNow! became the successor to MMX.

3DNow! was a big success. The instructions were quickly integrated into Windows, into various games (and other programs) and into hardware manufacturers’ driver programs.

SSE

After AMD's success with 3DNow!, Intel had to come back with something else. Their answer, in January 1999, was SSE (Streaming SIMD Extensions), which are another way to improve 3D performance. SSE was introduced with the Pentium III.

In principle, SSE is significantly more powerful than 3DNow! The following changes were made in the CPU:

- 8 new 128-bit registers, which can contain four 32-bit numbers at a time.
- 50 new SIMD instructions which make it possible to do advanced calculations on several floating point numbers with just one instruction.
- 12 *New Media Instructions,* designed, for example, for the encoding and decoding of MPEG-2 video streams (in DVD).
- 8 new *Streaming Memory* instructions to improve the interaction between L2 cache and RAM.

SSE also quickly became a success. Programs like Photoshop were released in new SSE optimised versions, and the results were convincing. Very processor-intensive programs involving sound, images and video, and in the whole area of multimedia, run much more smoothly when using SSE.

Since SSE was such a clear success, AMD took on board the technology. A large part of SSE was built into the AthlonXP and Duron processors. This was very good for software developers (and hence for us users), since all software can continue to be developed for one instruction set, common to both AMD and Intel.

SSE2 and SSE3

With the Pentium 4, SSE was extended to use even more powerful techniques. SSE2 contains 144 new instructions, including 128-bit SIMD integer operations and 128-bit SIMD *double-precision floating-point operations.*

SSE2 can reduce the number of instructions which have to be executed by the CPU in order to perform a certain task, and can thus increase the efficiency of the processor. Intel mentions *video, speech recognition, image/photo processing, encryption* and *financial/scientific programs* as the areas which will benefit greatly from SSE2. But as with MMX, 3DNow! and SSE, the programs have to be rewritten before the new instructions can be exploited.

SSE2 adopted by the competition, AMD, in the Athlon 64-processors. Here AMD even doubled up the number of SSE2 registers compared to the Pentium 4. Latest Intel has introduced 13 new instructions in SSE3, which Intel uses in the Prescott-version of Pentium 4.

We are now going to leave the discussion of instructions. I hope this examination has given you some insight into the CPU's work of executing programs.

- Next chapter.
- Previous chapter.

- Next chapter.
- Previous chapter.

Chapter 32. Examples of CPU's

In this chapter I will briefly describe the important CPU's which have been on the market, starting from the PC's early childhood and up until today.

One could argue that the obsolete and discontinued models no longer have any practical significance. This is true to some extent; but the old processors form part of the "family tree", and there are still legacies from their architectures in our modern CPU's, because the development has been evolutionary. Each new processor extended and built "on top of" an existing architecture.

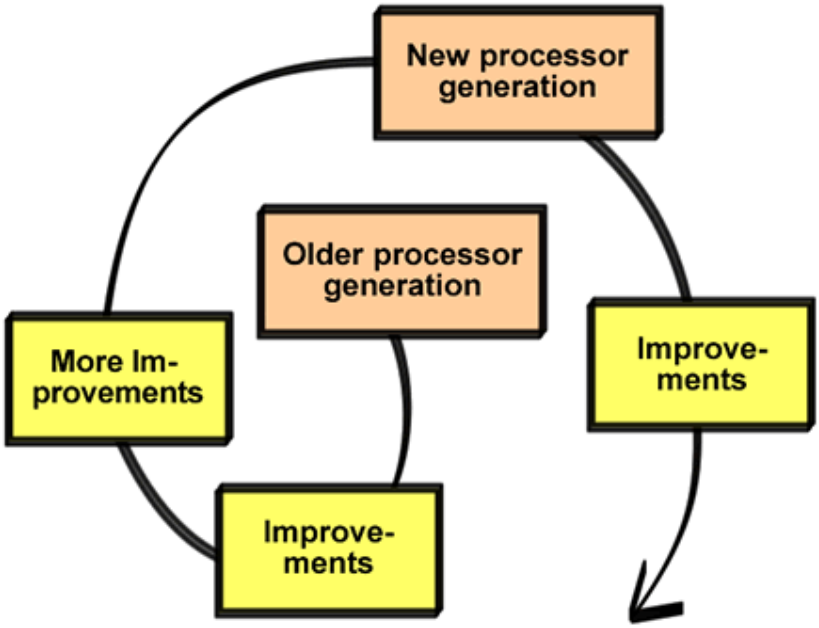


Figure 98. The evolutionary development spirals ever outwards.

There is therefore value (one way or another) in knowing about the development from one generation of CPU's to the next. If nothing else, it may give us a feeling for what we can expect from the future.

16 bits – the 8086, 8088 and 80286

The first PC's were 16-bit machines. This meant that they could basically only work with text. They were tied to DOS, and could normally only manage one program at a time.

But the original 8086 processor was still "too good" to be used in standard office PC's. The Intel 8088 discount model was therefore introduced, in which the bus between the CPU and RAM was halved in width (to 8 bits), making production of the motherboard much cheaper. 8088 machines typically had 256 KB, 512 KB or 1 MB of RAM. But that was adequate for the programs at the time.

The Intel 80286 (from 1984) was the first step towards faster and more powerful CPU's. The 286 was much more efficient; it simply performed much more work per clock tick than the 8086/8088 did. A new feature was also the *32 bit protected mode* – a new way of working which made the processor much more efficient than under *real mode*, which the 8086/8088 processor forced programs to work in:

- Access to *all system memory* – even beyond the 1MB limit which applied to real mode.
- Access to *multitasking*, which means that the operating system can run several programs at the same time.
- The possibility of *virtual memory*, which means that the hard disk can be used to emulate extra RAM, when necessary, via a *swap file*.
- 32 bit access to RAM and 32 bit drivers for I/O devices.

Protected mode paved the way for the change from DOS to Windows, which only came in the 1990's.

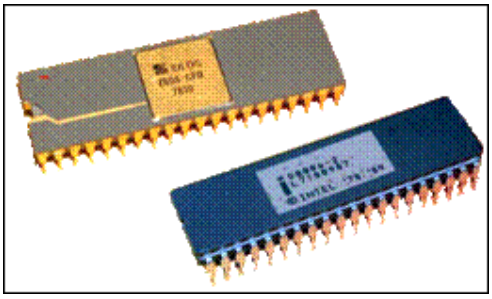


Figure 99. Bottom: an Intel 8086, the first 16-bit processor. Top: the incredibly popular 8-bit processor, the Zilog Z80, which the 8086 and its successors out competed.

32 bits – the 80386 and 486

The Intel 80386 was the first 32-bit CPU. The 386 has 32-bit long registers and a 32-bit data bus, both internally and externally. But for a traditional DOS based PC, it didn't bring about any great revolution. A good 286 ran nearly as fast as the first 386's – under DOS anyway, since it doesn't exploit the 32-bit architecture.

The 80386SX became the most popular chip – a discount edition of the 386DX. The SX had a 16-bit external data bus (as opposed to the DX's 32-bit bus), and that made it possible to build cheap PC's.

Figure 100. Discount prices in October 1990 – but only with a b/w monitor.

The fourth generation

The fourth generation of Intel's CPU's was called the 80486. It featured a better implementation of the x86 instructions – which executed faster, in a more RISC-like manner. The 486 was also the first CPU with built-in L1 cache. The result was that the 486 worked roughly twice as fast as its predecessor – for the same clock frequency.

With the 80486 we gained a built-in FPU. Then Intel did a marketing trick of the type we would be better off without. In order to be able to market a cheap edition of the 486, they hit on the idea of *disabling* the FPU function in some of the chips. These were then sold under the name, 80486SX. It was ridiculous – the processors had a built-in FPU; it had just been switched off in order to be able *to segment the market*.

Figure 101. Two 486's from two different manufacturers.

But the 486 was a good processor, and it had a long life under DOS, Windows 3.11 and Windows 95. New editions were released with higher clock frequencies, as they hit on the idea of *doubling* the *internal* clock frequency in relation to the external (see the discussion later in the guide). These double-clocked processors were given the name, 80486DX2.

A very popular model in this series had an external clock frequency of 33 MHz (in relation to RAM), while working at 66MHz internally. This principle (*double-clocking*) has been employed in one way or another in all later generations of CPU's. AMD, IBM, Texas Instruments and Cyrix also produced a number of 80486 compatible CPU's.

Pentium

In 1993 came the big change to a new architecture. Intel's Pentium was the first fifth-generation CPU. As with the earlier jumps to the next generation, the first versions weren't especially fast. This was particularly true of the very first Pentium 60 MHz, which ran on 5 volts. They got burning hot – people said you could fry an egg on them. But the Pentium quickly benefited from new process technology, and by using clock doubling, the clock frequencies soon skyrocketed.

Basically, the major innovation was a *superscalar* architecture. This meant that the Pentium could process several instructions at the same time (using several pipelines). At the same time, the RAM bus width was increased from 32 to 64 bits.

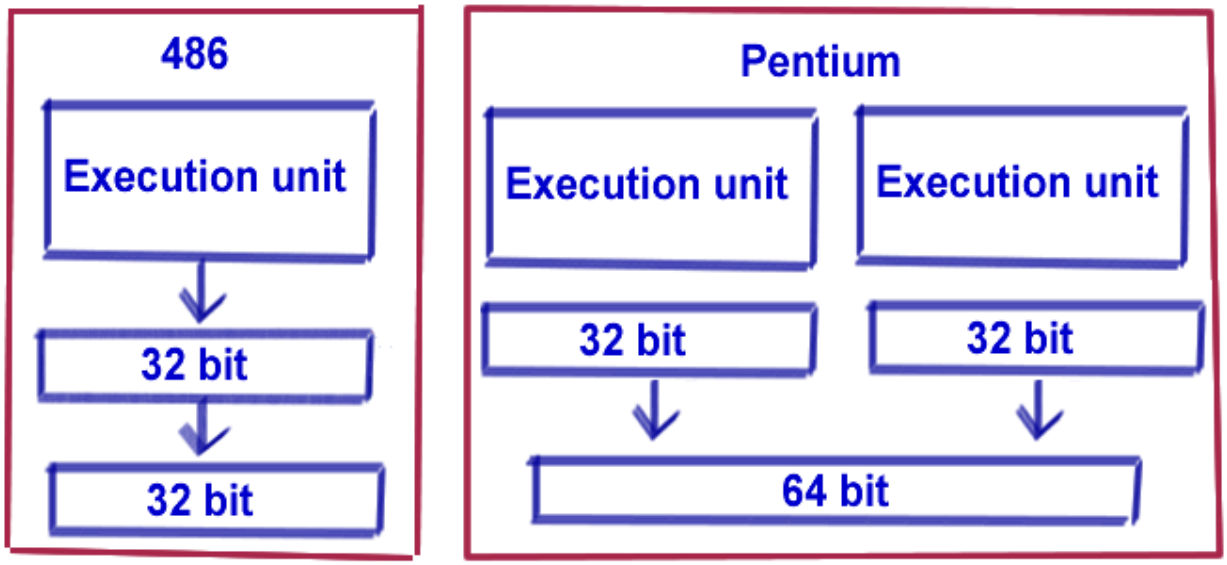


Figure 102. The Pentium processor could be viewed as two 80486's built into one chip.

Throughout the 1990's, AMD gained attention with its *K5* and *K6* processors, which were basically cheap (and fairly poor) copies of the Pentium. It wasn't until the *K6-2* (which included the very successful 3DNow! extensions), that AMD showed the signs of independence which have since led to excellent processors like the AthlonXP.

Figure 103. One of the earlier AMD processors. Today you'd hesitate to trust it to run a coffee machine…

In 1997, the Pentium MMX followed (with the model name P55), introducing the MMX instructions already mentioned. At the same time, the L1 cache was doubled and the clock frequency was raised.

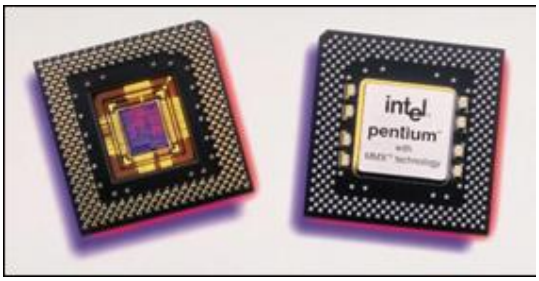


Figure 104. The Pentium MMX. On the left, the die can be seen in the middle.

Pentium II with new cache

After the Pentium came the Pentium II. But Intel had already launched the Pentium Pro in 1995, which was the first CPU in the 6th generation. The Pentium Pro was primarily used in servers, but its architecture was re-used in the popular Pentium II, Celeron and Pentium III models, during 1997-2001.

The Pentium II initially represented a technological step backwards. The Pentium Pro used an integrated L2 cache. That was very advanced at the time, but Intel chose to place the cache outside the actual Pentium II chip, to make production cheaper.

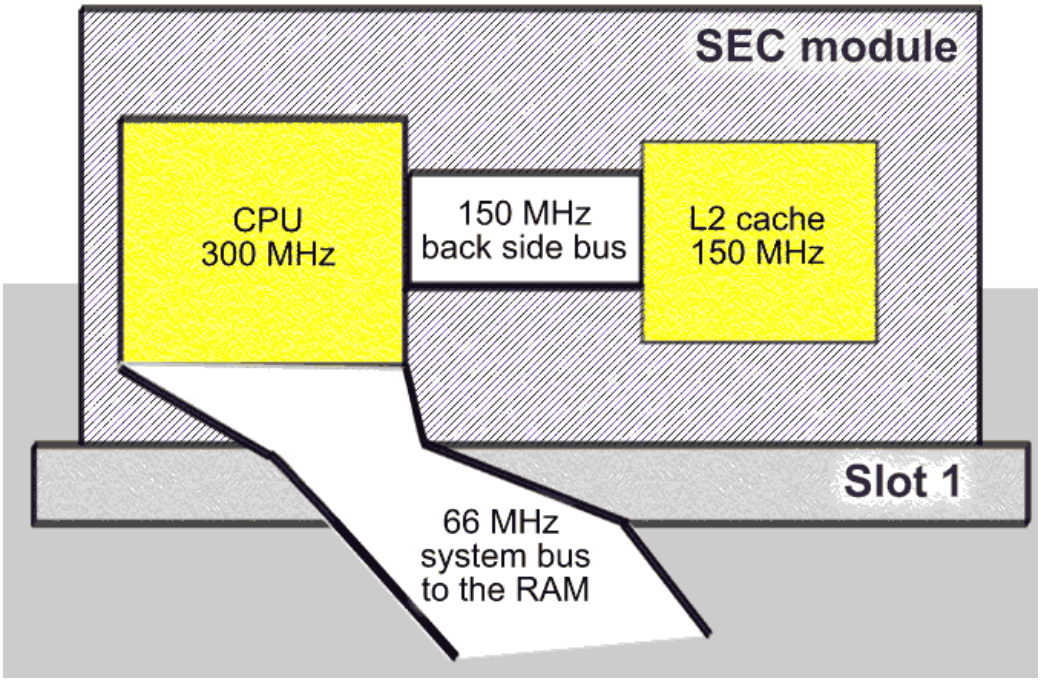


Figure 105. L2 cache running at half CPU speed in the Pentium II.

The Level 2 cache was placed beside the CPU on a circuit board, an *SEC module* (e.g. see Fig. 71). The module was installed in a long *Slot 1* socket on the motherboard. Fig. 106 shows the module with a cooling element attached. The CPU is sitting in the middle (under the fan). The L2 cache is in two chips, one on each side of the processor.

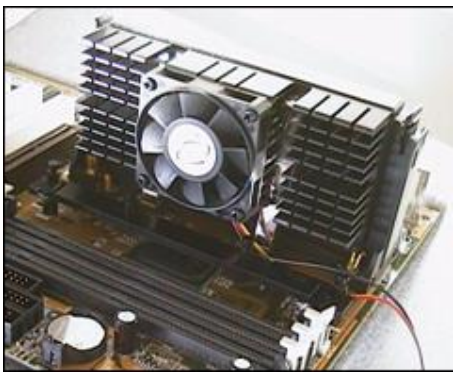


Figure 106. Pentium II processor module mounted on its edge in the motherboard's Slot 1 socket (1997-1998).

The disadvantage of this system was that the L2 cache became markedly slower than it would have been if it was integrated into the CPU. The L2 cache typically ran at half the CPU's clock frequency. AMD used the same system in their first Athlons. For these the socket was called, *Slot A* (see Fig. 107).

At some point, Intel decided to launch a discount edition of the Pentium II – the Celeron processor. In the early versions, the L2 cache was simply scrapped from the module. That led to quite poor performance, but provided an opportunity for *overclocking*.

Overclocking means pushing a CPU to work at a higher frequency than it is designed to work at. It was a very popular sport, especially early on, and the results were good.

Figure 107. One of the first AMD Athlon processors, mounted in a Slot A socket. See the large cooling element.

One of the problems of overclocking a Pentium II was that the cache chips couldn't keep up with the high speeds. Since these Celerons didn't have any L2 cache, they could be seriously overclocked (with the right cooling).

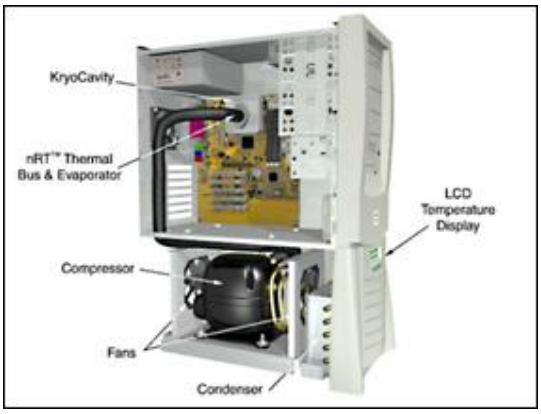


Figure 108. Extreme CPU cooling using a complete refrigerator built into the PC cabinet. With equipment like this, CPU's can be pushed up to very high clock frequencies (See Kryotech.com and Asetek.com).

Intel later decided to integrate the L2 cache into the processor. That happened in a new versions of the Celeron in 1998 and a new versions of the Pentium III in 1999. The socket design was also changed so that the processors could be mounted directly on the motherboard, in a socket called *socket 370*. Similarly, AMD introduced their *socket A*.

Pentium 4 – long in the pipe

The Pentium III was really just (yet) another edition of the Pentium II, which again was a new version of the Pentium Pro. All three processors built upon the same core architecture (Intel *P6*).

It wasn't until the Pentium 4 came along that we got a completely new processor from Intel. The core (*P7*) had a completely different design:

- The L1 cache contained decoded instructions.
- The pipeline had been doubled to 20 stages (in later versions increased to 31 stages).
- The integer calculation units (ALU's) had been double-clocked so that they can perform two micro operations per clock tick.
- Furthermore, the memory bus, which connects the RAM to the north bridge, had been *quad-pumped*, so that it transfers four data packets per clock tick. That is equivalent to 4 x 100 MHz and 4 x 133 in the earliest versions of the Pentium 4. In later version the bus was pumped up to 4 x 200 MHz, and an update with 4 x 266 MHz is scheduled for 2005.
- The processor was Hyper Threading-enabled, meaning that it under certain circumstances may operate as two individual CPUs.

All of these factors are described elsewhere in the guide. The important thing to understand, is that the Pentium 4 represents a completely new processor architecture.

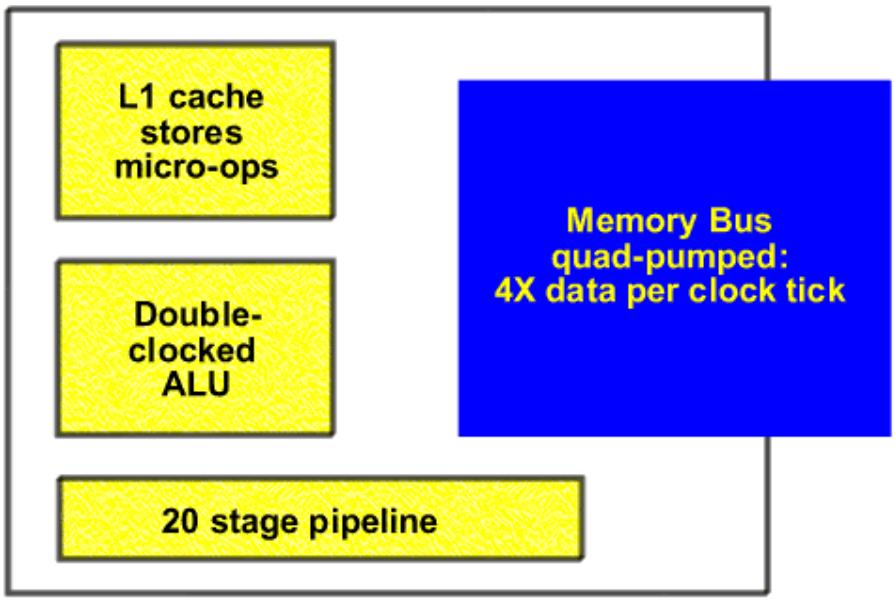


Figure 109. The four big changes seen in the Pentium 4.

Evolution of the Pentium 4

As was mentioned earlier, the older P6 architecture was released back in 1995. Up to 2002, the Pentium III processors were sold alongside the Pentium 4. That means, in practise, that Intel's sixth CPU generation has lasted 7 years.

Similarly, we may expect this seventh generation Pentium 4 to dominate the market for a number of years. The processors may still be called Pentium 4, but it comes in al lot varietes.

A mayor modification comes with the version using 0.65 micron process technology. It will open for higher clock frequencies, but there will also be a number of other improvements.

Hyper-Threading Technology is a very exciting structure, which can be briefly outlined as follows: In order to exploit the powerful pipeline in the Pentium 4, it has been permitted to process *two threads at the same time*. Threads are series of software instructions. Normal processors can only process *one* thread at a time.

In servers, where several processors are installed in the same motherboard (MP systems), several threads can be processed at the same time. However, this requires that the programs be set up to exploit the MP system, as discussed earlier.

The new thing is that a single Pentium 4 logically can function as if there physically were two processors in the pc. The processor core (with its long pipelines) is simply so powerful that it can, in many cases, act as two processors. It's a bit like one person being able to carry on two independent telephone conversations at the same time.

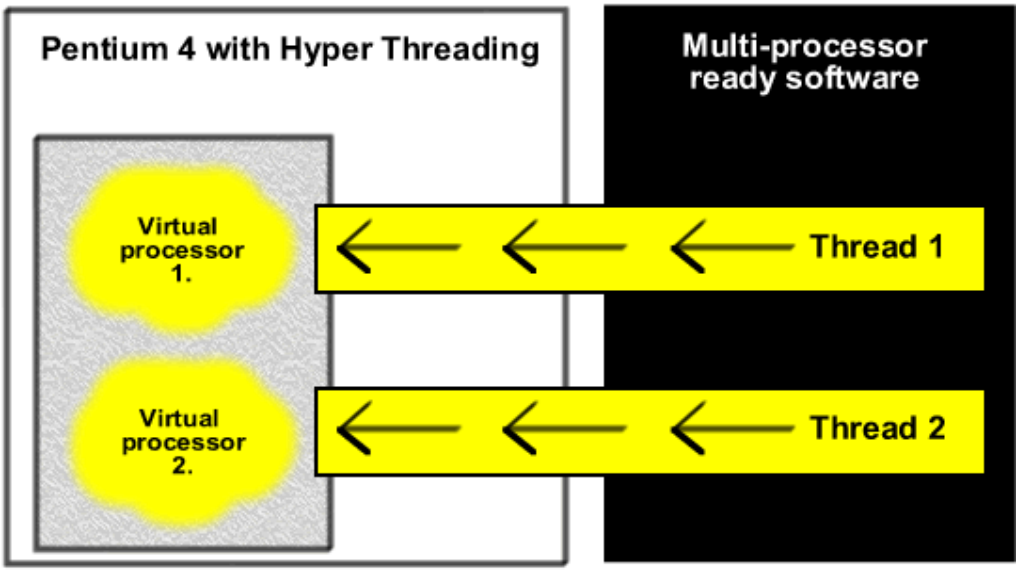


Figure 110. The Pentium 4 is ready for MP functions.

Hyper-Threading works very well in Intel's Prescott-versions of Pentium 4. You gain performance when you operate more than one task at the time. If you have two programs working simultaneously, both putting heavy pressure on the CPU, you will benefit from this technology. But you need a MP-compatible operating system (like Windows XP Professional) to benefit from it.

The next step in this evolution is the production of *dual-core processors.* AMD produces Opteron chips which hold two processors in one chip. Intel is working on dual core versions of the Pentium 4 (with the codename "Smithfield"). These chips will find use in servers and high performance pc's. A dual core Pentium 4 with Hyper-Threading enabled will in fact operate as a virtual *quad-core* processor.

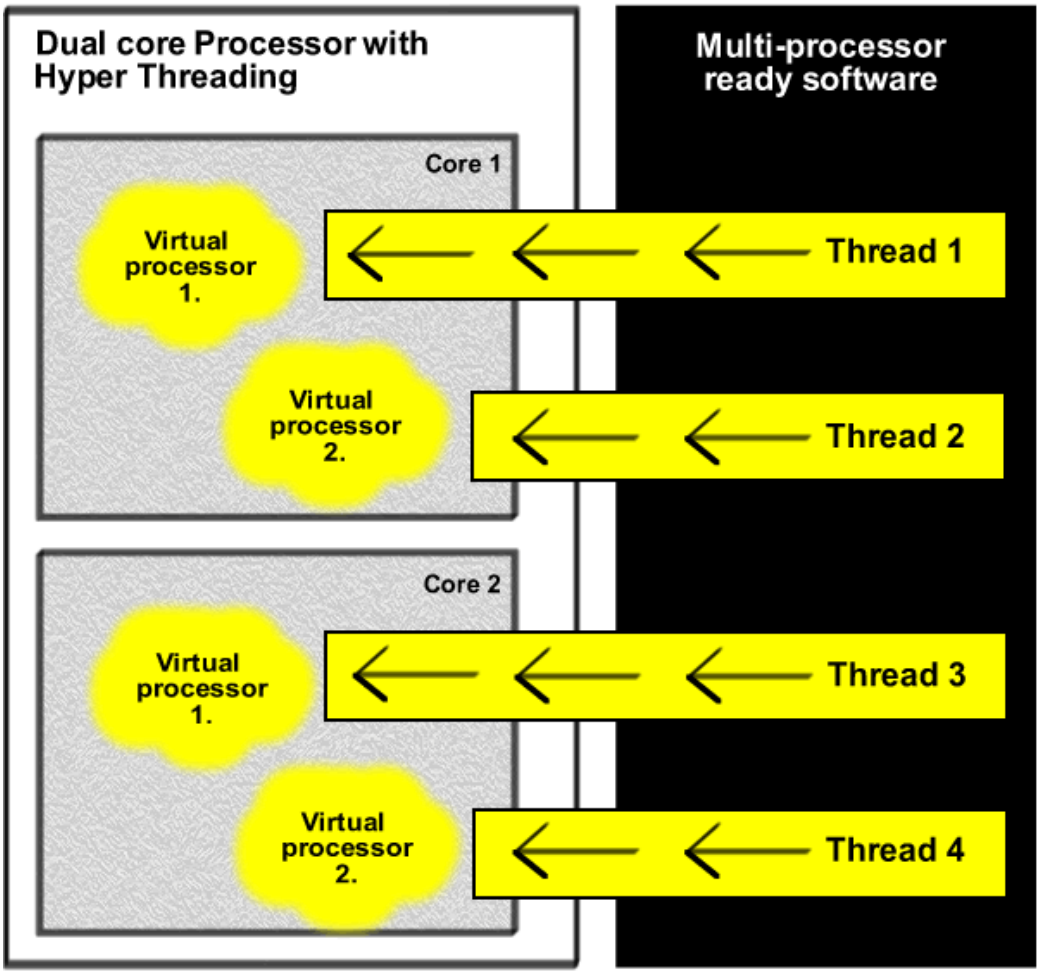
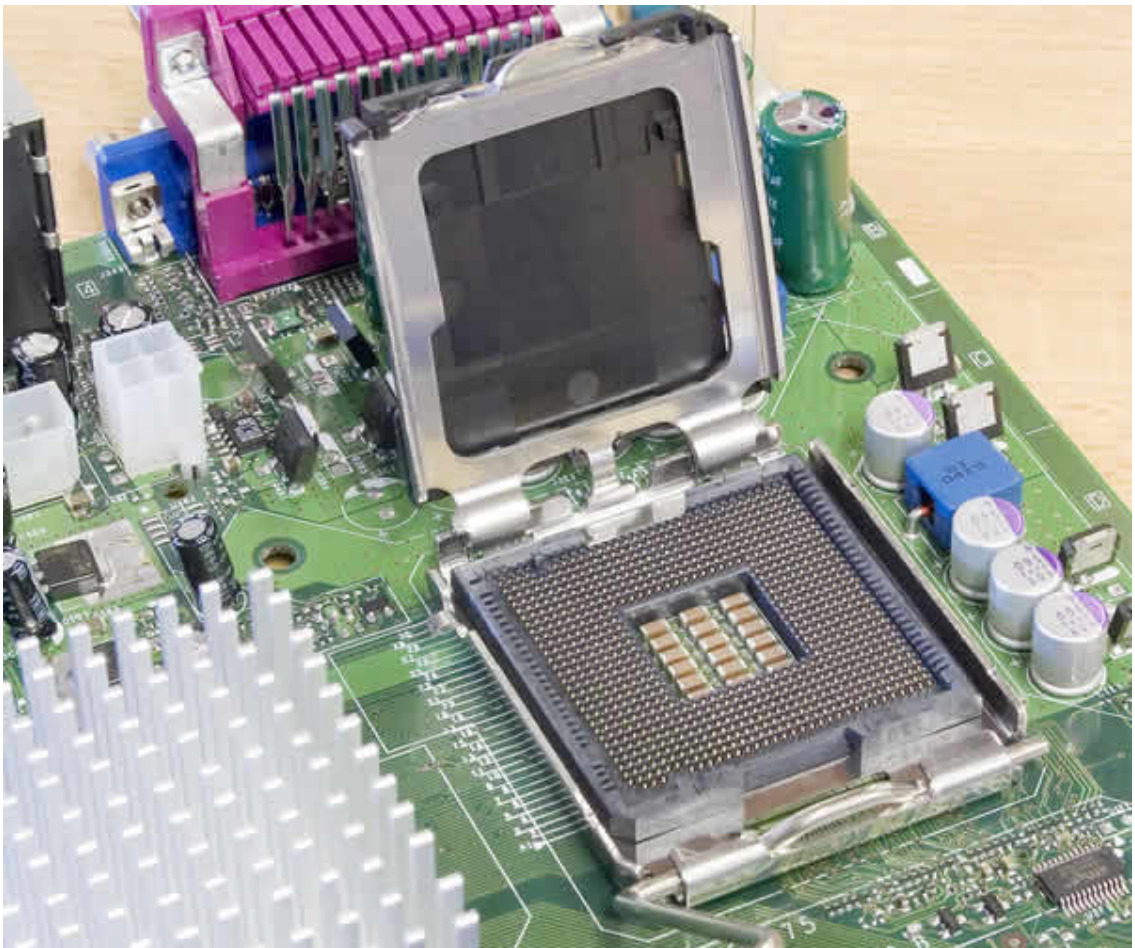


Figure 111. A dual core processor with Hyper Threading operates as virtual quad-processor.

Intel also produces EE-versions of the Pentium 4. EE is for *Extreme Edition*, and these processors are extremely speedy versions carrying 2 MB of L2 cache.

In late 2004 Intel changed the socket design of the Pentium 4. The new processors have no ”pins”; they connect directly to the socket using little contacts in the processor surface.



Figur 112. The LGA 775 socket for Pentium 4.

Athlon

The last processor I will discuss is the popular Athlon and Athlon 64 processor series (or *K7* and *K8*).

It was a big effort on the part of the relatively small manufacturer, AMD, when they challenged the giant Intel with a complete new processor design.

The first models were released in 1999, at a time when Intel was the completely dominant supplier of PC processors. AMD set their sights high – they wanted to make a better processor than the Pentium II, and yet cheaper at the same time. There was a fierce battle between AMD and Intel between 1999 and 2001, and one would have to say that AMD was the victor. They certainly took a large part of the market from Intel.

The original 1999 Athlon was very powerfully equipped with pipelines and computing units:

- Three instruction decoders which translated X86 program CISC instructions into the more efficient RISC instructions (ROP's) – 9 of which could be executed at the same time.
- Could handle up to 72 instructions (*ROP out of order*) at the same time (the Pentium III could manage 40, the K6-2 only 24).
- Very strong FPU performance, with three simultaneous instructions.

All in all, the Athlon was in a class above the Pentium II and III in those years. Since Athlon processors were sold at competitive prices, they were incredibly successful. They also launched the *Duron* line of processors, as the counterpart to Intel's Celeron, and were just as successful with it.

Figure 113.
Athlon was a huge success for AMD. During 2001-2002, the Athlon XP was in strong competition with the Pentium 4.

Athlon XP versus Pentium 4

The Athlon processor came in various versions. It started as a Slot A module (see Fig. 107 on page 42). It was then moved to Socket A, when the L2 cache was integrated.

In 2001, a new Athlon XP version was released, which included improvements like a new *Hardware Auto Data Prefetch Unit* and a bigger *Translation Look-aside Buffer*. The Athlon XP was much less advanced than the Pentium 4 but quite superior at clock frequencies less than 2000 MHz. A 1667 MHz version of AthlonXP was sold as 2000+. This indicates, that the processor as a minimum performs like a 2000 MHz Pentium 4.

Later we saw Athlons in other versions. The latest was based on a new kernel called "Barton". It was introduced in 2003 with a L2-cachen of 512 KB. AMD tried to sell the 2166 MHz version under the brand 3000+. It did not work. A Pentium 4 running at 3000 MHz had no problems outperforming the Athlon.

Opteron/ Athlon64

AMD's 8th generation CPU was released in 2003. It is based on a completely new core called *Hammer.*

A new series of 64-bits processors is called Athlon 64, Athlon 64 FX and Opteron. These CPU's has a new design in two areas:

- The memory controller is integrated in the CPU. Traditionally this function has been housed in the north bridge, but now it is placed inside the processor.
- AMD introduces a completely new 64-bit set of instructions.

Moving the memory controller into the CPU is a great innovation. It gives a much more efficient communication between CPU and RAM (which has to be ECC DDR SDRAM – 72 bit modules with error correction).)

Every time the CPU has to fetch data from normal RAM, it has to first send a request to the chipset's controller. It has to then wait for the controller to fetch the desired data – and that can take a long time, resulting in wasted clock ticks and reduced CPU efficiency. By building the memory controller directly into the CPU, this waste is reduced. The CPU is given much more direct access to RAM. And that should reduce latency time and increase the effective bandwidth.

The Athlon 64 processors are designed for 64 bits applications. This should be more powerful than the existing 32 bit software. We will probably see plenty of new 64 bit software in the future, since Intel is releasing 64 bit processors compatible with the Athlon 64 series.

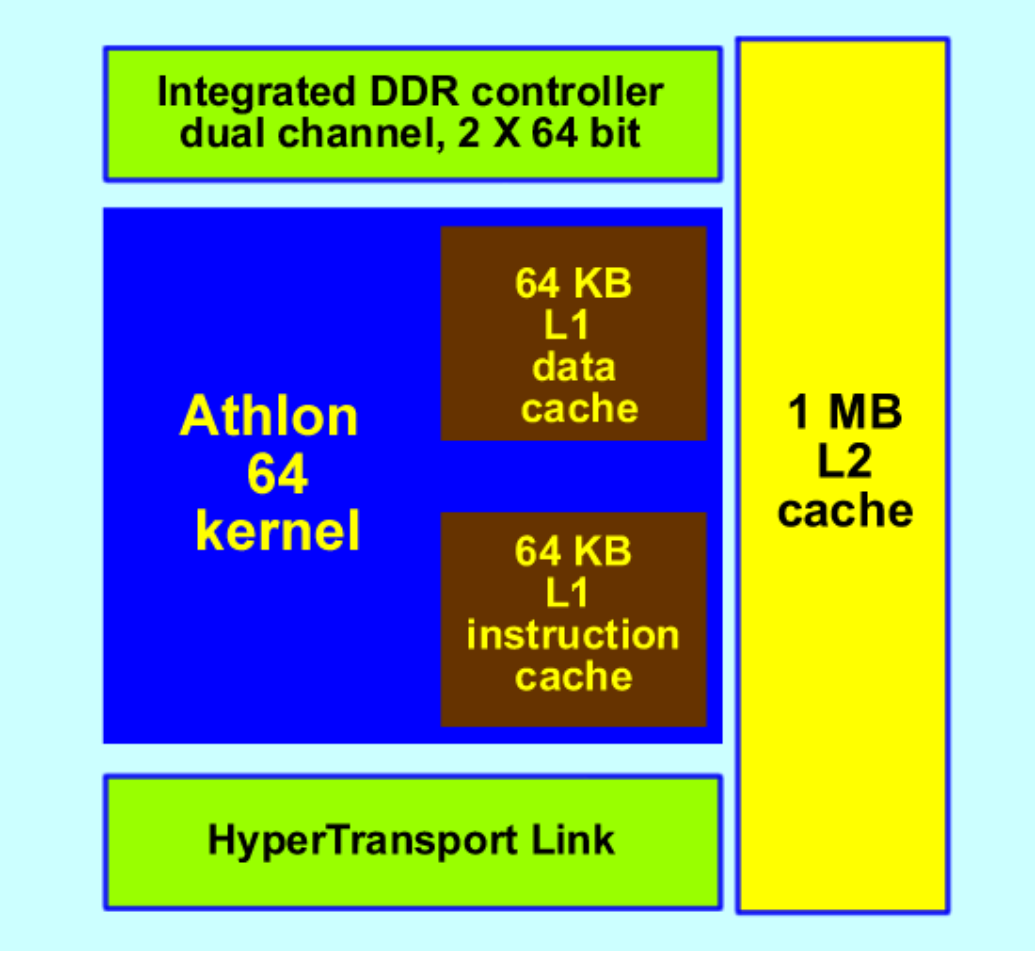


Figure 114. In the Athlon 64 the memory controller is located inside the processor. Hence, the RAM modules are interfacing directly with the CPU.

Overall the Athlon 64 is an updated Athlon-processor with integrated north bridge and 64 bits instructions. Other news are:

- Support for SSE2 instructions and 16 registers for this.
- Dual channel interface to DDR RAM giving a 128 bit memory bus, although the discount version Athlon 64 keeps the 64 bit bus.
- Communikationen to and from the south bridge via a new HyperTransport bus, operating with high-speed serial transfer.
- New sockets of 754 and 940 pins.

A complete line of chips

AMD expects to use the K8 kernel in all types of processors:

AMD 64 Opteron	The Opteron is the most expensive and advanced version to be used in multi-processor servers. The models are called 200, 400 and 800, and they use 2, 4 or 8 CPUs on the same motherboard – without use of a north bridge. All processors share a common memory of up to 64 GB. Each Opteron has three Hyper-Transport I/O channels, which each can move 6,4 GB/secund.
AMD 64 Athlon FX	The Athlon FX is a Opteron to be used in single processor configurations, high-end pc's and workstations. There is dual RAM interface, but only one channel of Hyper Transport Link.
AMD 64 Athlon	This is the discount version with reduced performance and lower prices. Only 64 bit RAM interface and smaller L2-cache.

Figure 115. Three versions of the latest AMD processor.

Historical overview

I will close off this review with a graphical summary of a number of different CPU's from the last 25 years. The division into generations is not always crystal clear, but I have tried to present things in a straightforward and reasonably accurate way:

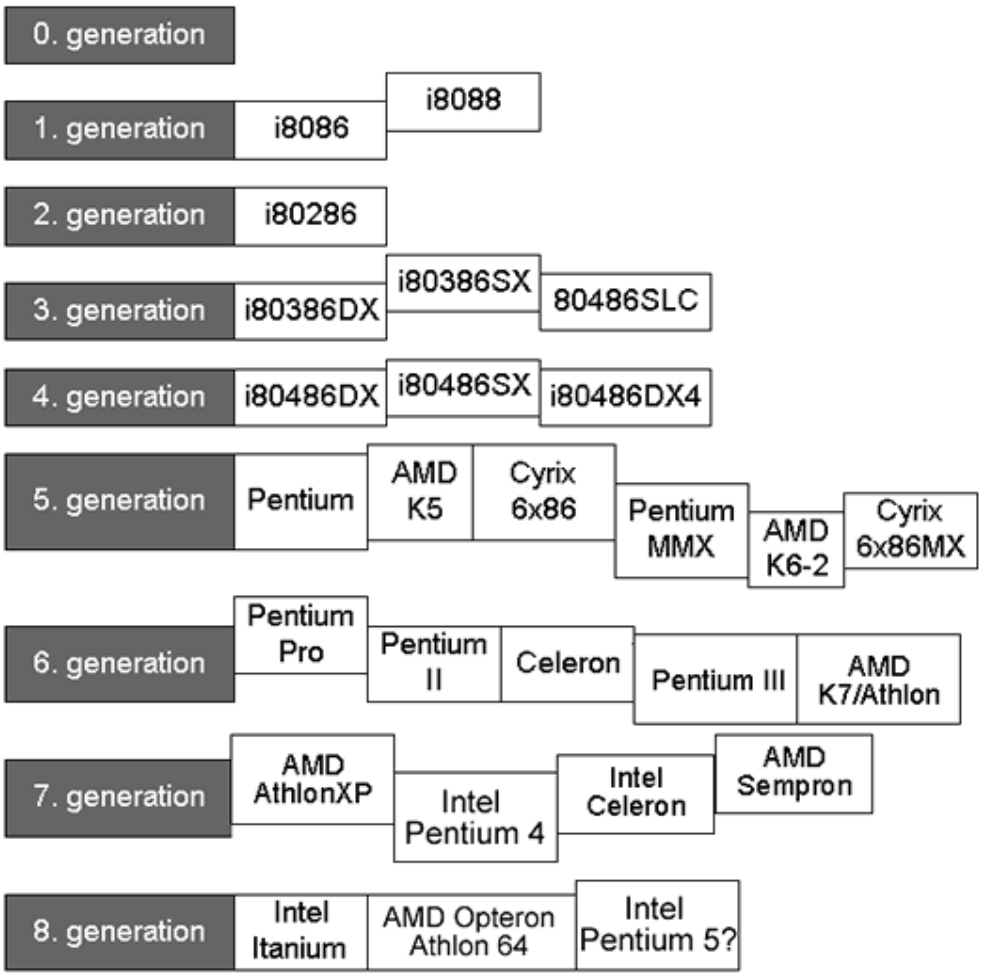


Figure 116. There are scores of different processors. A selection of them is shown here, divided into generations.

But what is the most powerful CPU in the world? IBM's Power4 must be a strong contender. It is a monster made up of 8 integrated 64-bit processor cores. It has to be installed in a 5,200 pin socket, uses 500 watts of power (there are 680 million transistors), and connects to a 32 MB L3 cache, which it controls itself. Good night to Pentium.

- Next chapter.
- Previous chapter.

- Next chapter.
- Previous chapter.

Chapter 33. Choosing a CPU

If you happen to need to choose a CPU for your new PC, what should you choose? Let me give you a bit of food for thought.

Processor	Euro
INTEL Celeron 2.66 GHZ/533 256KB	70,00
INTEL P4 520 2.8 GHZ/800 1 MB	130,00
INTEL P4 530 3.0 GHZ/800 1 MB	145,00
INTEL P4 540 3.2 GHZ/800 1 MB	175,00
INTEL P4 550 3.2 GHZ/800 1 MB	220,00
INTEL P4 560 3.2 GHZ/800 1 MB	330,00
AMD Sempron 3100+ (1,8 GHZ)	105,00
AMD ATHLON 64 3000+ (2,0 GHz)	209,00
AMD ATHLON 64 3400+ (2,4 GHz)	227,00
AMD ATHLON 64 FX-53 (2,4 GHz)	670,00

Figure 117. Pricelist from October 2004 (without VAT).

How long does a CPU last?

The individual components have different lifetimes. The way development has gone up until the

present, CPU's and motherboards have been the components that have become *obsolete* the most quickly. CPU's and motherboards also go together -– you normally change them both at the same time.

You have by now read all of my review of new processor technology; new and faster models are constantly being developed. The question is, then, how important is it to have the latest technology? You have to decide that for yourself. But if you know that your PC has to last for many years, you probably should go for the fastest CPU on the market.

For the rest of us, who regularly upgrade and replace our PC's insides, it is important to find the most economic processor. There is usually a *price jump* in the processor series, such that the very latest models are disproportionately expensive.

By the time you read this, prices will have fallen in relation to those shown in Fig. 117. There will no doubt also be significantly faster CPU's to choose between. But the trend will most likely be the same: The latest models are a lot more expensive than the other ones. You have to find the model that gives the most power in proportion to the price.

Also, the amount of RAM is just as important as the speed of the processor. RAM prices fluctuate unbelievably, in just a year the price can double or halve. So it's a good idea to buy your RAM when the prices are low.

CPU-intensive operations

Back in the 1990's it was quite important to get the latest and fastest CPU, because CPU's were not fast enough for things like image processing. For example, if you try to work with fairly large images in Photoshop, on a PC with a 233 MHz processor, you will probably quickly decide to give up the project.

But whether you have 2400 or 3200 MHz – that's not as critical, especially if you have enough RAM and are working with normal tasks. A processor running at 3200 MHz is roughly 33% faster than a 2400 MHz model, but it doesn't always make that much difference to your daily work.

Here are some tasks, which might require more CPU power:

- Video editing.
- DVD ripping (often illegal).
- Photo editing, especially with high resolution images and 48 bit color depth.
- Production of PDF-files in high quality.
- Speech recognition.

Video (including DVD) contains huge amounts of data. The CPU and RAM therefore have to work very hard when you edit video footage. At the time of writing, it is legal to copy DVD films for personal use, but that may change. Legal or not – it's certainly possible. The actual *ripping* can take 10-20 minutes, during which the CPU slowly chews through the over 4 GB of data there are on a DVD. A 5 GHz CPU would obviously be able to reduce the time taken considerably.

Finally, *speech recognition* software is considered to be very CPU-intensive. This is probably only going to be a problem in the distant future. Some people imagine future versions of Windows and Office programs will have built-in speech recognition, which will place quite new demands on CPU's. However, it is far from certain that speech recognition will ever be introduced into PC's. There appear to be big problems in getting it to work well.

- Next chapter.

- Next chapter.
- Previous chapter.

Chapter 34. The CPU's immediate surroundings

In the second part of this guide, we dug down into the inner workings of the CPU. We well let it rest in peace now, and concentrate on the processor's immediate surroundings. That is, the RAM and the chipset – or more precisely, the north bridge.

In the first section of the guide I introduced the chipset, including the *north bridge* (see, for example, Fig. 46, which connects the CPU to the PC's memory — the RAM.

The most important data path on the motherboard runs between the CPU and the RAM. Data is constantly pumped back and forth between the two, and this bus therefore often comes under focus when new generations of CPU's, chipset's and motherboards are released.

The RAM sends and receives data on a *bus*, and this work involves a *clock frequency*. This means that all RAM has a *speed*, just like a CPU does. Unfortunately RAM is much slower than the CPU, and the buses on the motherboard have to make allowance for this fact.

The XT architecture

In the original PC design (the IBM XT), the CPU, RAM and I/O devices (which we will come to later) were connected on one and the same bus, and everything ran *synchronously* (at a common speed). The CPU decided which clock frequency the other devices had to work at:

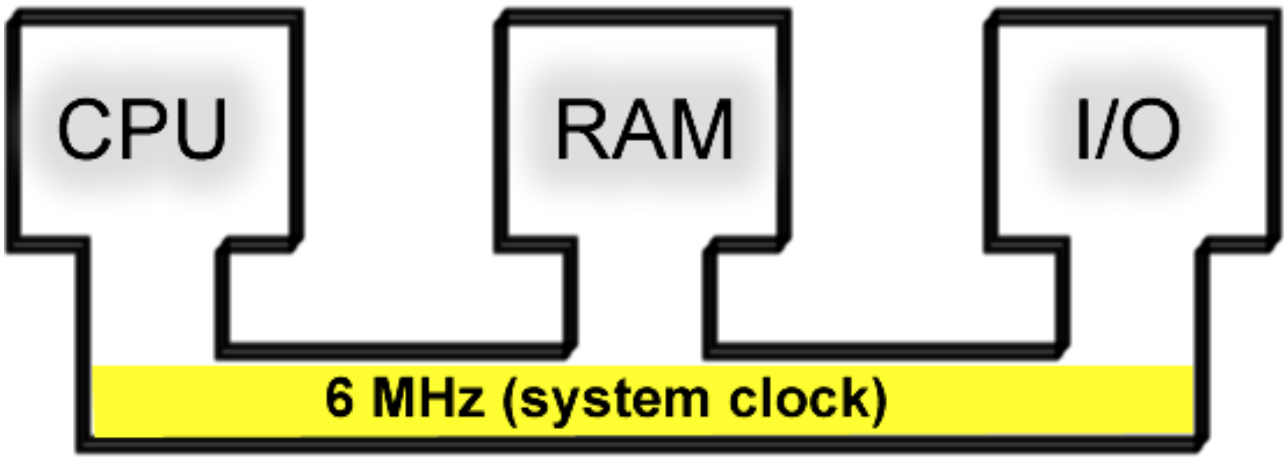


Figure 118. In the original PC architecture, there was only one bus with one speed.

The problem with this system was that the three devices were "locked to each other"; they were forced to work at the lowest common clock frequency. It was a natural architecture in the first PC's, where the speed was very slow.

The first division of the bus

In 1987, Compaq hit on the idea of separating the system bus from the I/O bus, so that the two buses could work at different clock frequencies. By letting the CPU and RAM work on their own bus, independent of the I/O devices, their speeds could be increased.

In Fig. 119, the CPU and RAM are connected to a common bus, called the *system bus*, where in reality the CPU's clock frequency determines the working speed. Thus the RAM has the same speed as the CPU; for example, 12, 16 or 25 MHz.

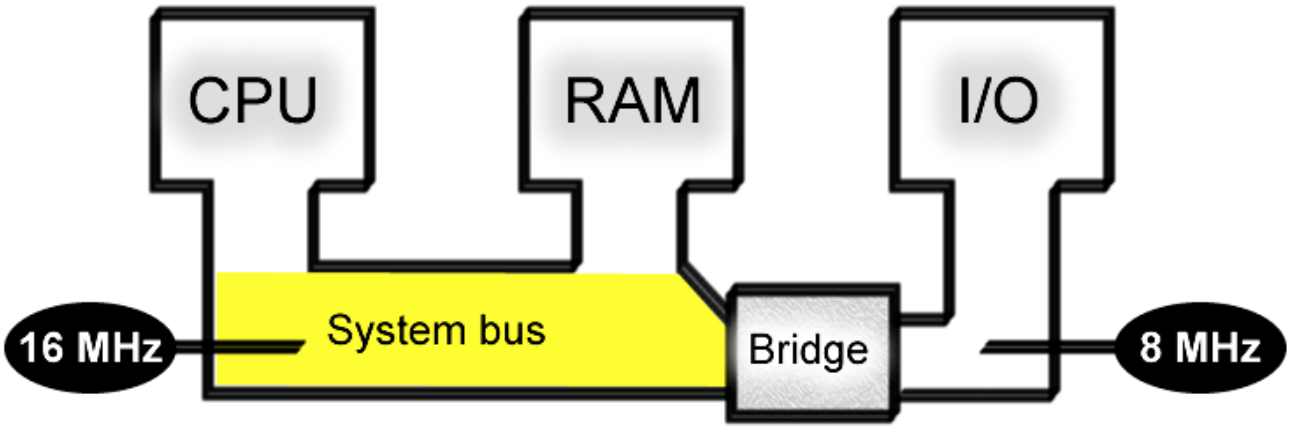


Figure 119. With this architecture, the I/O bus is separate from the system bus (80386).

The I/O devices (graphics card, hard disk, etc.) were separated from the system bus and placed on a separate low speed bus. This was because they couldn't keep up with the clock frequencies of the new CPU versions.

The connection between the two buses is managed by a controller, which functions as a "bridge" between the two paths. This was the forerunner of the multibus architecture which all motherboards use today.

Clock doubling

With the introduction of the 80486, the CPU clock frequency could be increased so much that the RAM could no longer keep up. Intel therefore began to use *clock doubling* in the 80486 processor.

The RAM available at the time couldn't keep up with the 66 MHz speed at which an 80486 could work. The solution was to give the CPU two working speeds.

- An *external* clock frequency
- An *internal* clock frequency

Inside the processor, the clock frequency of the system bus is multiplied by a factor of 2, doubling the working speed.

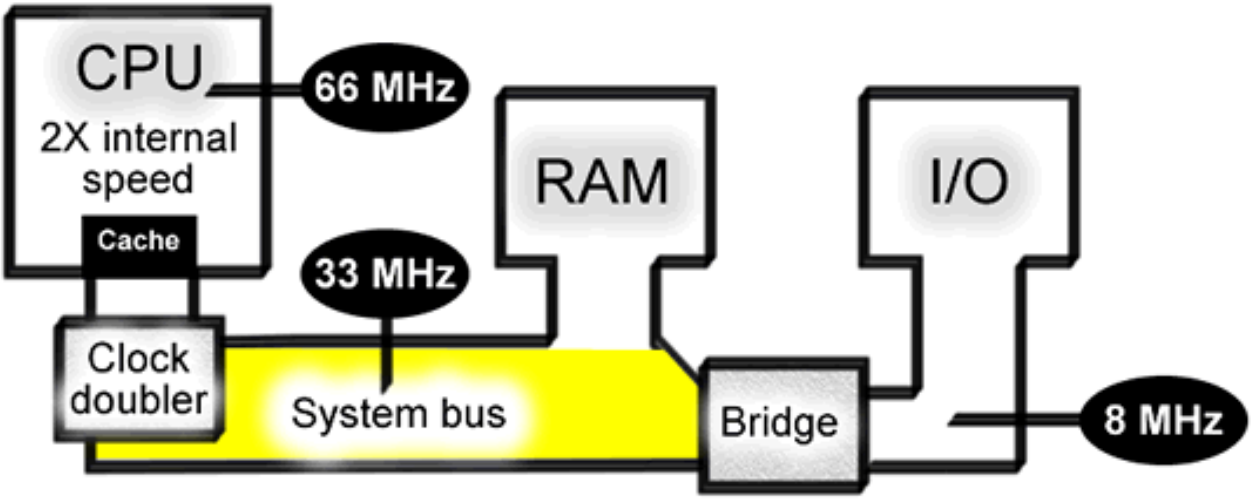


Figure 120. The bus system for an 80486 processor.

But this system places heavy demands on the RAM, because when the CPU internally processes twice as much data, it of course has to be “fed” more often. The problem is, that the RAM only works half as fast as the CPU.

For precisely this reason, the 486 was given a built-in L1 cache, to reduce the imbalance between the slow RAM and the fast processor. The cache doesn’t improve the bandwidth (the RAM doesn’t work any faster), but it ensures greater efficiency in the transfer of data to the CPU, so that it gets the right data supplied at the right time.

Clock doubling made it possible for Intel to develop processors with higher and higher clock frequencies. At the time the Pentium was introduced, new RAM modules became available, and the system bus was increased to 66 MHz. In the case of the Pentium II and III, the system bus was increased to 100 and 133 MHz, with the internal clock frequency set to a multiple of these.

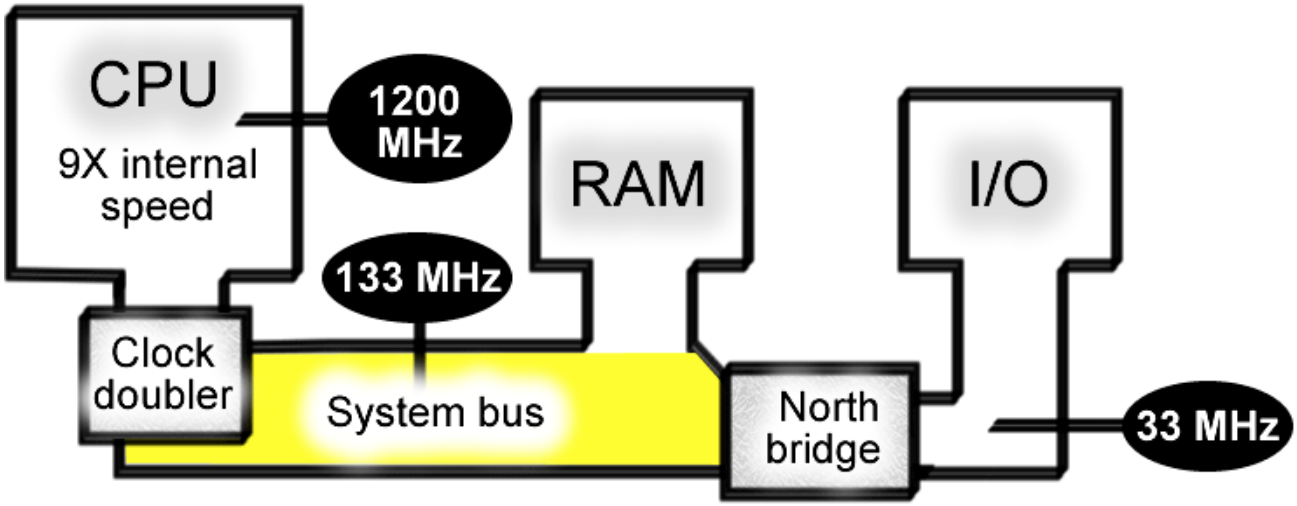


Figure 121. The bus system for a Pentium III processor.

Overclocking

The Pentium II was subjected to a lot of *overclocking*. It was found that many of Intel’s CPU’s could be clocked at a higher factor than they were designed for.

If you had a 233 MHz Pentium II, you could set up the motherboard to, for example, run at 4.5 x 66 MHz, so that the processor ran at 300 MHz. I tried it myself for a while, it worked well. At a factor of 5 it didn't work, but at factor of 4.5 it functioned superbly.

CPU	System bus	Clock factor	Internal clock frequency
Pentium	66 MHz	1.5	100 MHz
Pentium MMX	66 MHz	2.5	166 MHz
Pentium II	66 MHz	4.5	300 MHz
Pentium II	100 MHz	6	600 MHz
Celeron	100 MHz	8	800 MHz
Pentium III	133 MHz	9	1200 MHz
AthlonXP	133 MHz x 2	13	1733 MHz
AthlonXP+	166 MHz x 2	13	2167 MHz
Pentium 4	100 MHz x 4	22	2200 MHz
Pentium 4	133 MHz x 4	23	3066 MHz
Pentium 4	200 MHz x 4	18	3600 MHz

Figure 122. The CPU's internal clock frequency is locked to the system bus frequency.

Overclocking the system bus

Another method of overclocking was to turn up the system bus clock frequency. In the early versions of the Pentium II, the system bus was at 66 MHz, which suited the type of RAM used at that time.

You could increase the bus speed, for example to 68 or 75 MHz, depending on how fast your RAM was. This type of tuning makes both the CPU and RAM faster, since it is the actual system clock speed which is increased.

The disadvantage is that the system clock in these motherboard architectures also controls the I/O bus, which runs synchronously with the system bus. PCI bus devices (which we will come to in a later chapter) cannot handle being overclocked very much; otherwise faults can occur, for example in reading from the hard disk.

Overclocking typically requires a higher voltage for the CPU, and most motherboards can be set up to supply this:

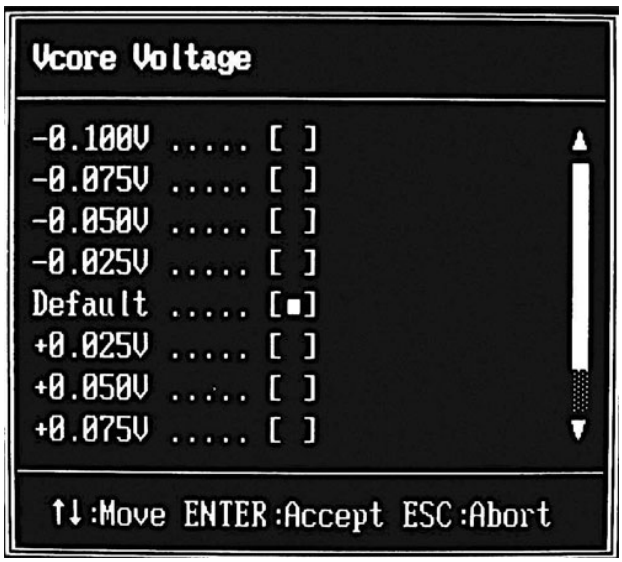


Figure 123. Setting the CPU voltage using the motherboard's Setup program.

Many still use the same kind of overclocking on the Athlon XP and Pentium 4. The system clock has to be able to be adjusted in increments, which it can on many motherboards.

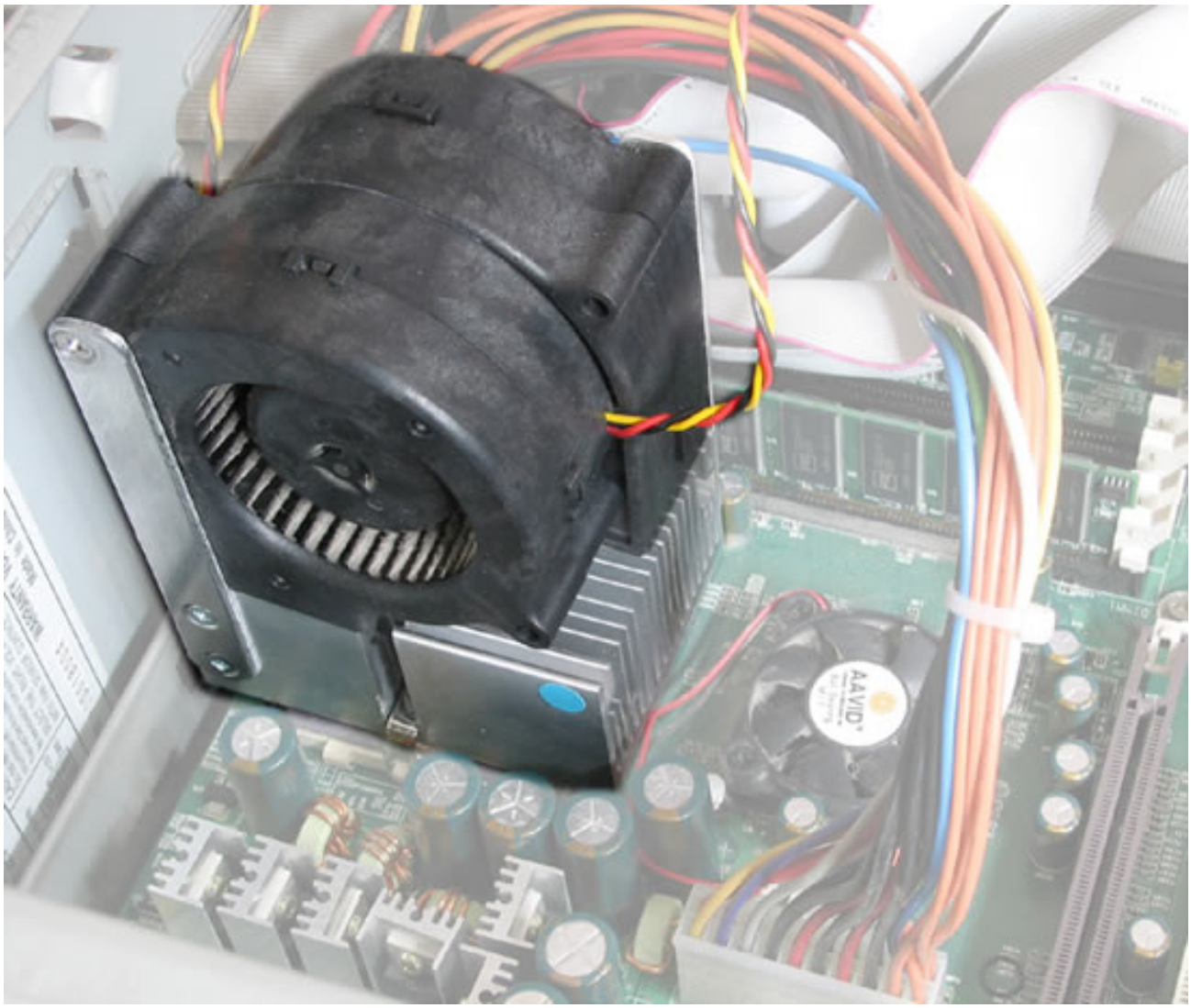


Figure 124. A nice big cooler with two fans and pure silver contact surfaces. Silverado, a German product which is used for overclocking CPU's.

Example using the Pentium 4

A Pentium 4 processor is designed for a system clock of 200 MHz. If you can have a 3200 MHz model with a 200 MHz system bus, it can theoretically be clocked up to 4000 MHz by turning up the system clock. However, the processor needs a very powerful cooling system to operate at the increased frequencies:

System clock	CPU clock
200 MHz	3200 MHz
230 MHz	3700 MHz
250 MHz	4000 MHz

Figure 125. Overclocking a Pentium 4 processor.

The manufacturers, Intel and AMD, don’t like people overclocking their CPU’s. They have sometimes attempted to prevent this by building a lock into the processors, so that the processor can only work at a specific clock frequency. In other cases the CPU’s can be overclocked. In any case, you shouldn’t expect your warranty to apply if you play around with overclocking.

-
- Next chapter.
- Previous chapter.

- Next chapter.
- Previous chapter.

Chapter 35. Different types of RAM

RAM works in synch with the system bus, as I described in the previous chapter. But what is RAM actually? RAM is a very central component in a PC, for without RAM there can be no data processing. RAM is simply the storage area where all software is loaded and works from.

Silicon storage area

RAM is made in electronic chips made of so-called semiconductor material, just like processors and many other types of chips.

In RAM, transistors make up the individual storage cells which can each “remember” an amount of data, for example, 1 or 4 bits – as long as the PC is switched on.

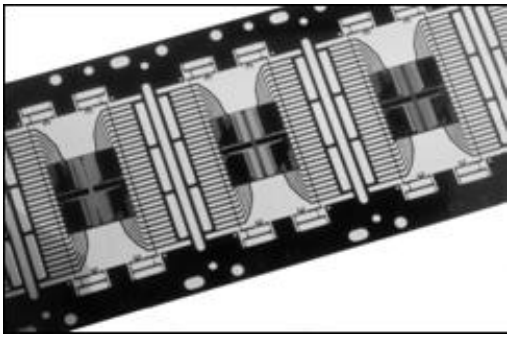


Figure 126. Manufacturing RAM.

Normal RAM is *dynamic* (called DRAM), and requires constant electronic recharging to preserve its data contents. Without power, all RAM cells are cleared. RAM is very closely linked to the CPU, and it is very important to both have *enough* RAM, and to have *fast* RAM. If both conditions are not met, the RAM will be a bottleneck which will slow down the PC. What follows is an introduction to RAM, as it is used in modern PC’s. After this I will discuss the various types I more detail.

Several types

At the time of writing, there are several types of RAM, which are quite different. This means that they normally cannot be used on the same motherboard – they are not compatible.

However, some motherboards can have sockets for two types of RAM. You typically see this during periods where there is a change taking place from one type of RAM to another. Such motherboards are really designed for the new type, but are made “backwards compatible”, by making room for RAM modules of the old type.

The RAM types on the market at the moment are:

RAM type	Pins	Width	Usage
SD RAM	168	64 bit	Older and slower type. No use.

Rambus RAM	184	16 bit	Advanced RAM. Only used for very few Pentium 4's with certain Intel chipsets.
DDR RAM	184	64 bit	A faster version of SD RAM. Used both for Athlon and Pentium 4's. 2,5 Volt.
DDR2 RAM	240	64 bit	New version of DDR RAM with higher clock frequencies. 1,8 Volt.

Figure 127. Very different types of RAM.

In any case, there is a lot of development taking place in DDR. A number of new RAM products will be released within the next few years. The modules are packaged differently, so they cannot be mixed. The notches in the sides are different as the he bottom edges of the modules are.

SDRAM is an old and proven type, which is used in the majority of existing PC's. DDR RAM is a refinement of SDRAM, which is in reality double clocked. Rambus RAM is an advanced technology which in principle is superior to DDR RAM in many ways. However, Rambus has had a difficult birth. The technology has been patented by Rambus Inc., which has been involved in many legal suits. A number of important manufacturers (such as VIA) have opted out of Rambus, and only develop products which use DDR RAM. With the new DDR2 standard, there is no obvious need for Rambus RAM.

Notes on the physical RAM

RAM stands for *Random Access Memory.* Physically, RAM consists of small electronic chips which are mounted in modules (small printed circuit boards). The modules are installed in the PC's motherboard using sockets — there are typically 2, 3 or 4 of these. On this motherboard there are only two, and that's a bit on the low side of what is reasonable.

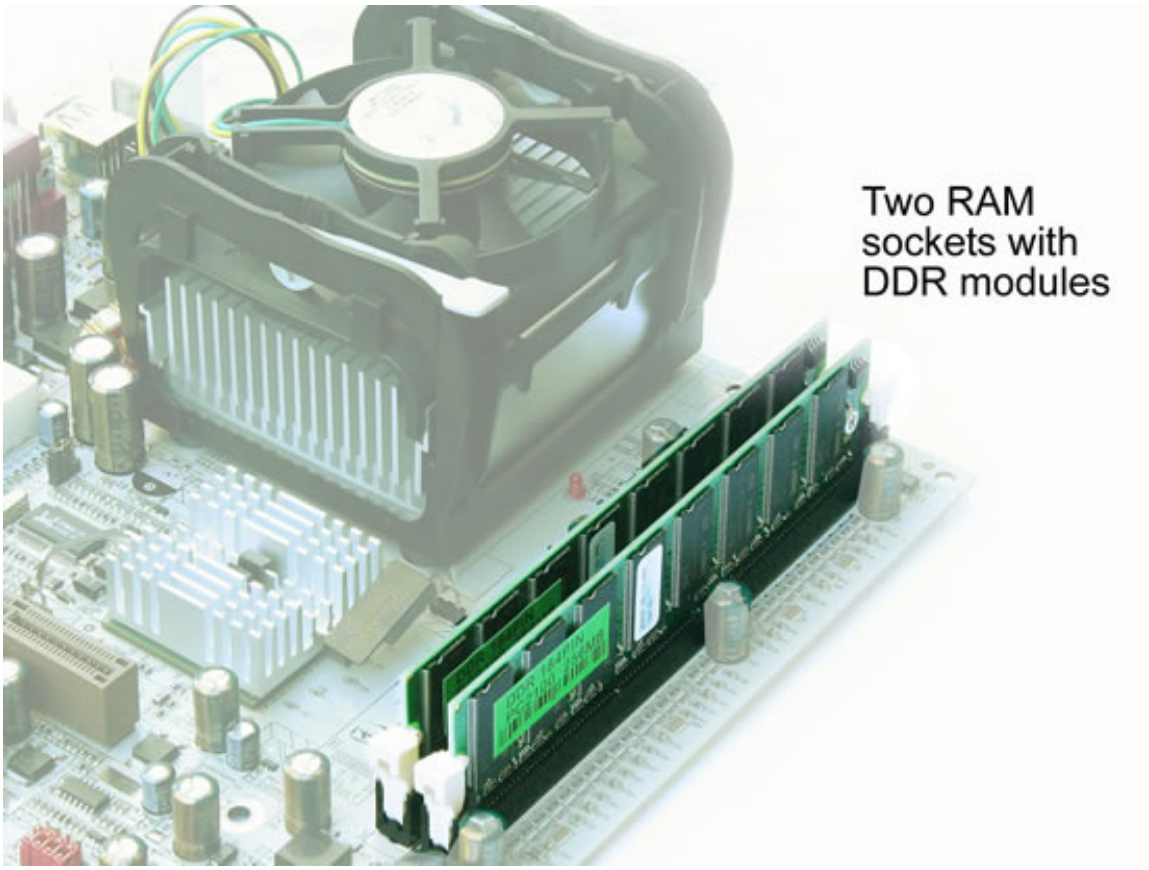


Figure 128. RAM modules are installed in sockets on the motherboard. In the background you see the huge fan on a Pentium 4 processor.

Each RAM module is a rectangular printed circuit board which fits into the sockets on the motherboard:

Figure 129. A 512 MB DDR RAM module.

On a module there are typically 8 RAM chips which are soldered in place. There can also be 16 if it is a double-sided module. Below is a single RAM chip:

Figure 130.
A single RAM chip, a
256 megabit circuit.

On the bottom edge of the module you can see the copper coated tracks which make electrical contact (the edge connector). Note also the *profile* of the module; this makes it only possible to install it one way round and in the right socket.

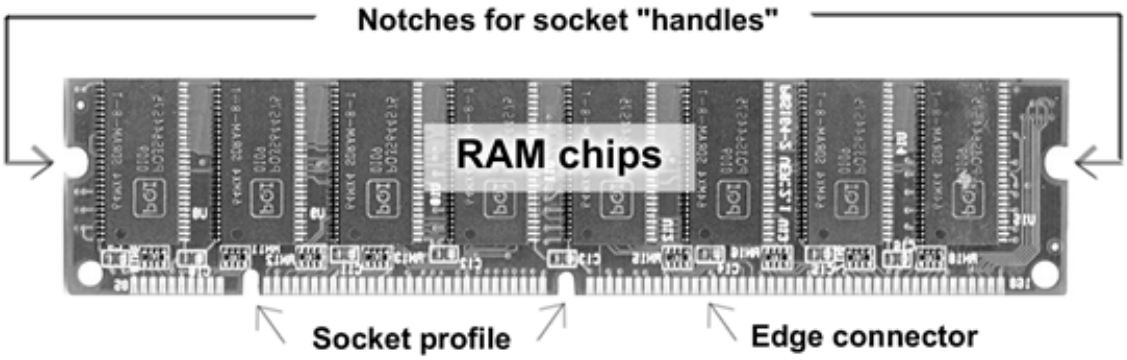
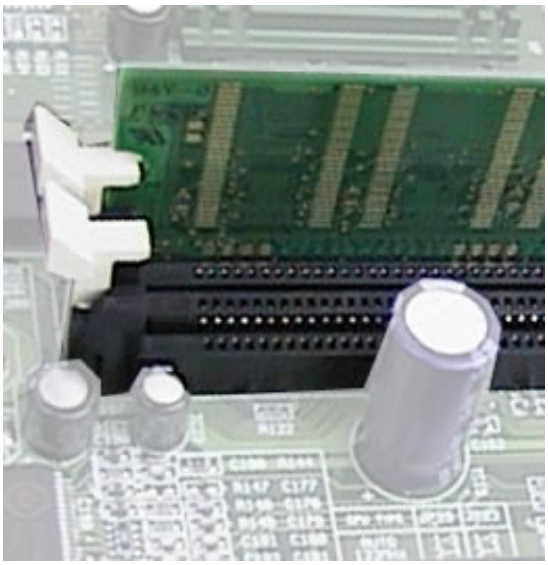


Figure 131. Anatomy of the SD RAM module.

The notches in the sides of the module fit the brackets or “handles” which hold the module in place in the motherboard socket:



Module or chip size

All RAM modules have a particular *data width*, which has to match the motherboard, chipset, and ultimately the CPU. Modules using the two most common RAM types, SD and DDR RAM, are 64 bits wide.

The modules are built using chips which each contain a number of megabits. And since each byte needs 8 bits, more than one chip is needed to make a module. Look at the RAM chip in Fig. 130. You can see the text ”64MX4”:

Figure 132. Text on a RAM chip.

The text in Fig. 132 indicates that the chip contains *64 x 4 mega bits* of data, which is the same as 256 megabits. If we want to do some calculations, each chip contains 1024 x 1024 x 64 = 67,108,864 cells, which can each hold 4 bits of data. That gives 268,435,456 bits in total, which (when divided by 8) equals 33,554,432 bytes = 32,768 KB = 32 MB.

So a 64MX4 circuit contains 32 MB, and is a standard product. This type of chip is used by many different manufacturers to make RAM modules. Modules are sold primarily in four common sizes, from 64 to 512 MB:

Number of chips per module	Module size
2 (single-sided)	2 x 256 Mbit = 64 Mbyte
4 (single-sided)	4 x 256 Mbit = 128 Mbyte
8 (single-sided)	8x 256 Mbit = 256 Mbyte
16 (double-sided)	16 x 256 Mbit = 512 Mbyte

Figure 133. The module size is determined by what RAM chips are used.

RAM speeds

For each type of RAM, there are modules of various sizes. But there are also modules with various *speeds*. The faster the RAM chips are, the more expensive the modules.

Name	Type
PC700	2x 356 MHz Rambus RAM
PC800	2x 400 MHz Rambus RAM
PC1066	2 x 533 MHz Rambus RAM
DDR266 el. PC2100	2x 133 MHz DDR RAM
DDR333 el. PC2700	2x 166 MHz DDR RAM
DDR400 el. PC3200	2x 200 MHz DDR RAM

DDR2-400	400 MHz DDR2 RAM
DDR2-533	533 MHz DDR2 RAM
DDR2-667	667 MHz DDR2 RAM

Figure 134. The speed is measured in megahertz, but it is not possible to compare directly between the three types of RAM shown in the table. The names shown are those which are being used on the market at the time of writing, but they may change.

RAM technologies

Let's look a little more closely at the technology which is used in the various types of RAM.

In the old days

Back in the 1980's, DRAM was used. This was dynamic RAM, which was relatively slow. It was replaced by FPM (*Fast Page Mode*) RAM which was also dynamic, only a bit faster.

Originally, loose RAM chips were installed directly in large *banks* on the motherboard. Later people started combining the chips in modules. These came in widths of 8 bits (with 30 pins) and 32 bits (with 72 pins). The 32-bit modules were suited to the system bus for the 80486 processor, which was also 32 bits wide.

Figure 135. Older RAM modules.

FPM RAM could not run any faster than 66 MHz, but that was fine for the system bus clock frequency in the original Pentium processors.

After FPM came EDO RAM (*Extended Data Out*). EDO is a bit faster than FPM because the data paths to and from the RAM cells have been optimised. The gain was a 3-5 % improvement in bandwidth. The clock frequency could be increased to 75 MHz, but basically, EDO is not very different to FPM RAM.

When Intel launched the Pentium processor, there was a change to using the 64 bit wide RAM modules (with 168 pins, as in Fig. 127, which are still used for SDRAM today.

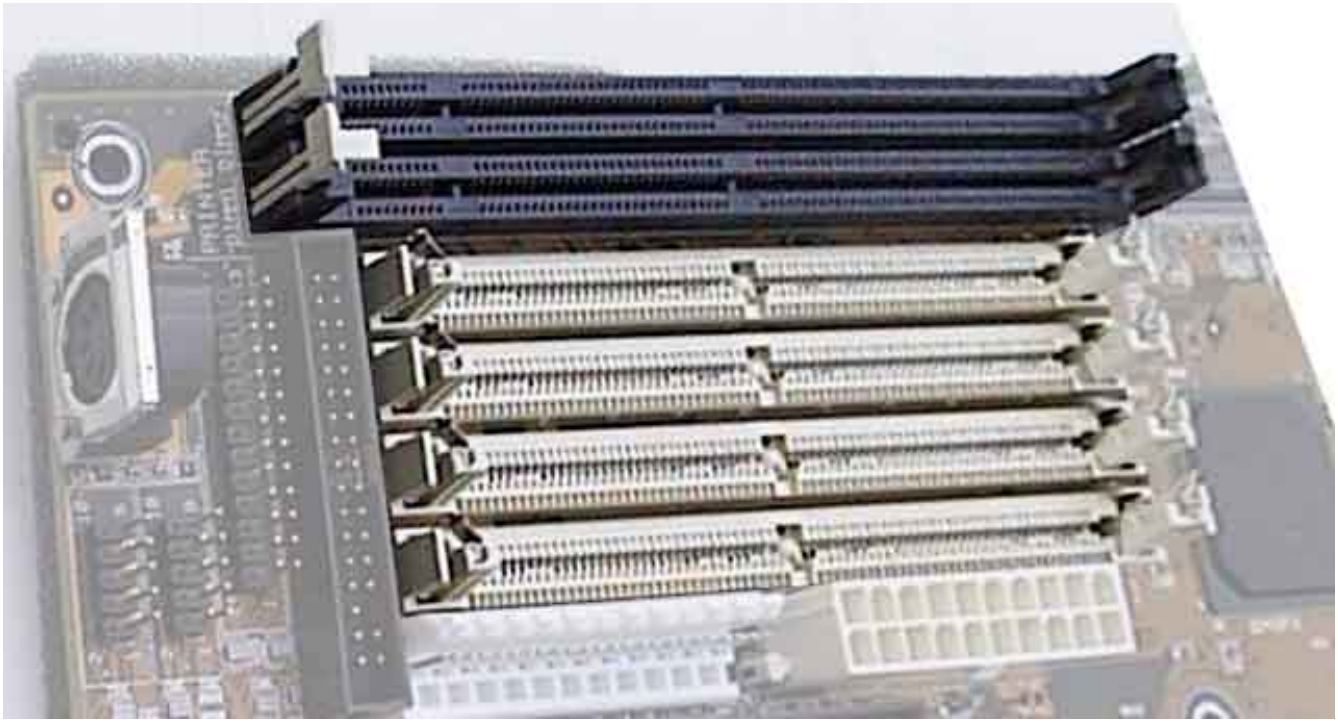


Figure 136. An old motherboard with sockets for both 64-bit and 32-bit RAM modules. From the transition period between EDO and SDRAM.

SDRAM

The big qualitative shift came in around 1997, when SDRAM (*Synchronous DRAM*) began to break in. This is a completely new technology, which of course required new chipsets. SDRAM, in contrast to the earlier types of RAM, operates *synchronously* with the system bus.

Data can (in *burst mode*) be fetched on *every* clock pulse. Thus the module can operate fully synchronised with (at the same beat as) the bus – without so-called *wait states* (inactive clock pulses). Because they are linked synchronously to the system bus, SDRAM modules can run at much higher clock frequencies.

The 100 MHz SDRAM (PC100) quickly became popular, and with new processors and chipsets, the speed was brought up to 133 MHz (PC133).

Another innovation in SDRAM is the small EEPROM chip called the *Serial Presence Detect* chip, which is mounted on the modules. It is a very small chip containing data on the modules speed, etc.

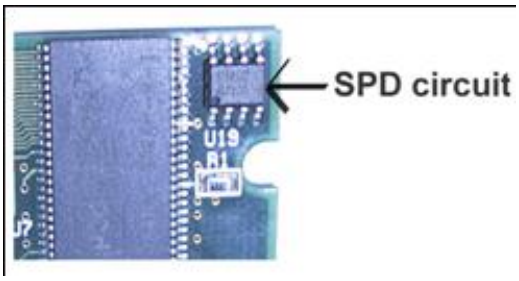


Figure 137. The motherboard BIOS can now read SDRAM module specifications directly.

DDR RAM

It is expensive to produce fast RAM chips. So someone hit on a smart trick in 1999-2000, which in one blow made normal RAM twice as fast. That was the beginning of DDR RAM (*Double Data Rate*). See the module in Fig. 131.

In DDR RAM, the clock signal is used *twice*. Data is transferred *both* when the signal rises, and when it falls. This makes it possible to perform twice as many operations per clock pulse, compared to earlier RAM types:

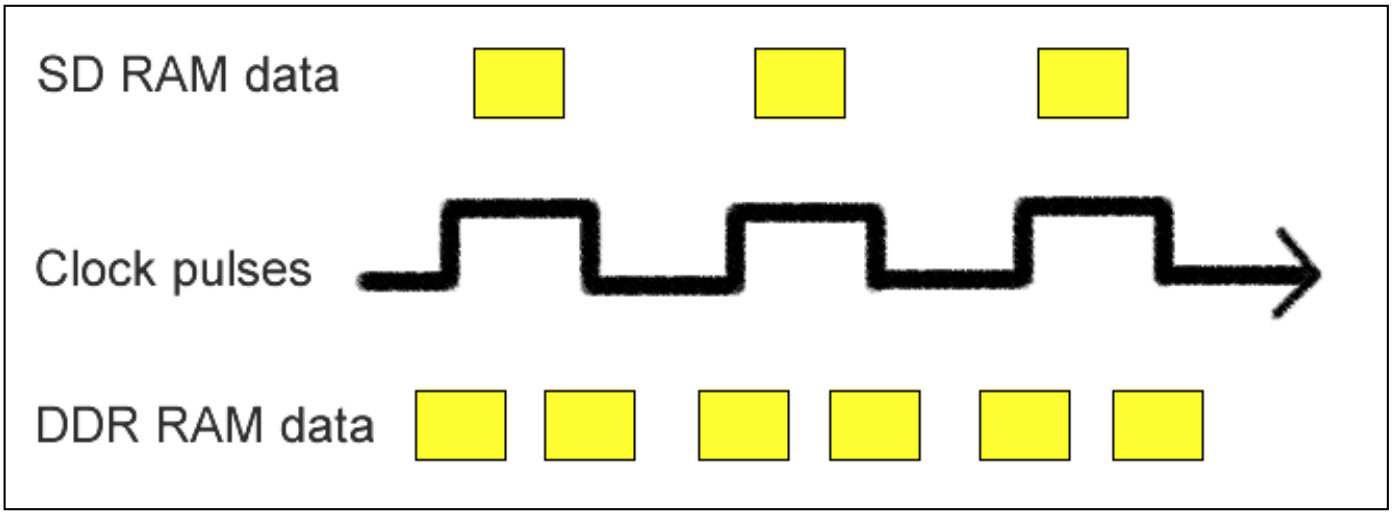


Figure 138. DDR RAM sends off two data packets for each clock pulse.

Timings

DDR RAM exist in many versions, with different the clock frequencies and *timings.* The timing indicates how many clock cycles there are wasted, when the motherboard waits for the memory to deliver the requested data.

With smaller numbers, we have better timings and the CPU having fewer idle clock cycles. You may find memory modules of the same clock frequency but with different timings. The better timing, the more expensive the RAM module is.

Ordinary pc users need not to speculate in special RAM with fast timing; this is primary sold to gamers and over-clockers, who tries to achieve the maximum performance from their motherboards.

Module	Clock frequency	Timing
PC2100	2 x 133 MHz	2-2-2
PC 2700	2 x 166 MHz	2-2-2 2-3-3
PC 3200	2 x 200 MHz	2-3-2 2-3-3
PC 3700	2 x 233 MHz	3-4-4
PC 4000	2 x 250 MHz	3-4-4
PC 4400	2 x 275 MHz	3-4-4

Note that different timing means that the gain in terms of increased bandwidth doesn't quite match the clock speed. It is a bit less.

Next generation RAM

In the beginning the problem with DDR RAM, was that the RAM modules were poorly standardized. A module might work with a particular motherboard, but not with another. But this – which was typical for a new technological standard – is not a big problem anymore. Intel was initially against DDR RAM. They claimed that Rambus was a much better design, and that they wouldn't use DDR RAM. But consumers wanted DDR RAM, which Intel's competitors were able to deliver, and in the end even Intel had to give in. At the end of 2001, the i845 chipset was released, which uses DDR RAM for the Pentium 4, and later we had the i865 and i875 chip sets, which use dual channel DDR RAM.

The next generation of RAM is the *DDR2*, which is a new and better standardized version of DDR using less power. The DDR2 modules operates at higher clock speeds due to better design with higher signal integrity and a more advanced internal data bus. The first chip sets to use DDR2 was Intel's i915 and i925. Later *DDR4* is expected with clock frequencies of up to 1,6 GHz!

Rambus RAM

Rambus Inc., as already mentioned, has developed a completely new type of RAM technology. Rambus uses a completely different type of chip, which are mounted in *intelligent* modules that can operate at very high clock frequencies. Here is a brief summary of the system:

- The memory controller delivers data to a narrow high-speed bus which connects all the RDRAM modules in a long series.
- The modules contain logic (*Rambus ASIC*), which stores the data in the format the chips use.
- Data is written to one chip at a time, in contrast to SDRAM where it is spread across several chips.
- The modules work at 2.5 volts, which is reduced to 0.5 volts whenever possible. In this way, both the build up of heat, and electromagnetic radiation can be kept down. They are encapsulated in a heat conducting, aluminium casing.

Rambus RAM thus has a completely new and different design. The modules are only 16 bits wide. Less data is transferred per clock pulse, but the clock frequencies are much higher. The actual Rambus modules (also called RIMM modules) look a bit like the normal SDRAM modules as we know them. They have 184 pins, but as mentioned, the ships are protected by a heat-conducting casing:

Figure 139. Rambus module.

As the advanced Rambus modules are quite costly to produce, the technology is on its way out of the market.

Advice on RAM

RAM can be a tricky thing to work out. In this chapter I will give a couple of tips to anyone having to choose between the various RAM products.

Bandwidth

Of course you want to choose the best and fastest RAM. It's just not that easy to work out what type of RAM is the fastest in any given situation.

We can start by looking at the theoretical maximum bandwidth for the various systems. This is easy to calculate by *multiplying the clock frequency by the bus width*. This gives:

Module type	Max. transfer
SD RAM, PC100	800 MB/sec
SD RAM, PC133	1064 MB/sec
Rambus, PC800	1600 MB/sec
Rambus, Dual PC800	3200 MB/sec
DDR 266 (PC2100)	2128 MB/sec
DDR 333 (PC2700)	2664 MB/sec
DDR 400 (PC3200)	3200 MB/sec
DUAL DDR PC3200	6400 MB/sec
DUAL DDR2-400	8600 MB/sec
DUAL DDR2-533	10600 MB/sec

Figure 140. The highest possible bandwidth (peak bandwidth) for the various types of RAM.

However, RAM also has to match the motherboard, chipset and the CPU system bus. You can try experimenting with overclocking, where you intentionally increase the system bus clock frequency. That will mean you need faster RAM than what is normally used in a given motherboard. However, normally, we simply have to stick to the type of RAM currently recommended for the chosen motherboard and CPU.

RAM quality

The type of RAM is one thing; the RAM quality is something else. There are enormous differences in RAM prices, and there are also differences in quality. And since it is important to have a lot of RAM, and it is generally expensive, you have to shop around.

One of the advantages of buying a clone PC (whether you build it yourself or buy it complete) is that you can use standard RAM. The brand name suppliers (like IBM and Compaq) use their own RAM, which can be several times more expensive than the standard product. The reason for this is that the RAM modules have to meet very specific specifications. That means that out a particular production run, only 20% may be "good enough", and that makes them expensive.

Over the years I have experimented with many types of RAM in many combinations. In my experience, for desktop PC's (not servers), you can use standard RAM without problems. But follow these precautions:

- Avoid mixing RAM from various suppliers and with various specifications in the same PC – even if others say it is fine to do so.
- Note that the RAM chips are produced at one factory, and the RAM modules may be produced at another.
- Buy standard RAM from a supplier you trust. You need to know who manufactured the RAM modules and the seller needs to have sold them over a longer period of time. Good brands are Samsung, Kingston and Corsair.

- The modules have to match the motherboard. Ensure that they have been tested at the speed you need to use them at.
- The best thing is to buy the motherboard and RAM together. It's just not always the cheapest.
- Avoid modules with more than 8 chips on each side.

How much RAM?

RAM has a very big impact on a PC's capacity. So if you have to choose between the fastest CPU, or more RAM, I would definitely recommend that you go for the RAM. Some will choose the fastest CPU, with the expectation of buying extra RAM later, "when the price falls again". You can also go that way, but ideally, you should get enough RAM from the beginning. But how much is that?

If you still use Windows 98, then 256 MB is enough. The system can't normally make use of any more, so more would be a waste. For the much better Windows 2000 operating system, you should ideally have at least 512 MB RAM; it runs fine with this, but of course 1024 MB or more is better. The same goes for Windows XP:

	128 MB	256 MB	512 MB	1024 MB
Windows 98	* *	* * *	Waste	Waste
Windows 2000	*	* *	* * *	* * * *
Windows XP		*	* * *	* * * *

Figure 141. Recommended amount of PC RAM, which has to be matched to the operating system.

The advantage of having enough RAM is that you avoid *swapping*. When Windows doesn't have any more free RAM, it begins to artificially increase the amount of RAM using a *swap file*. The swap file is stored on the hard disk, and leads to a much slower performance than if there was sufficient RAM in the PC.

RAM addressing

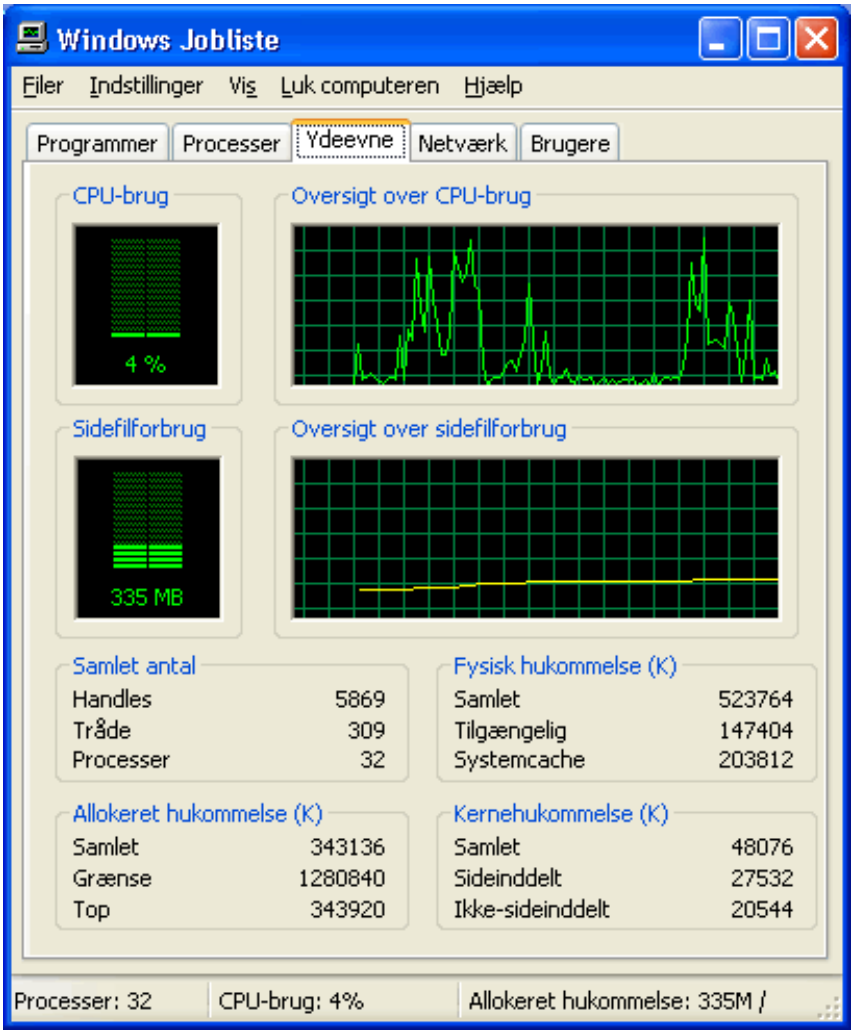
Over the years there have been many myths, such as "Windows 98 can't use more than 128 MB of RAM", etc. The issue is RAM *addressing*.

Below are the three components which each have an upper limit to how much RAM they can address (access):

- The operating system (Windows).
- The chipset and motherboard.
- The processor.

Windows 95/98 has always been able to access lots of RAM, at least in theory. The fact that the memory management is so poor that it is often meaningless to use more than 256 MB, is something else. Windows NT/2000 and XP can manage several gigabytes of RAM, so there are no limits at the moment.

In Windows XP, you have to press Control+Alt+Delete in order to select the Job list. A dialog box will then be displayed with two tabs, Processes and Performance, which provide information on RAM usage:



Under the Processes tab, you can see how much RAM each program is using at the moment. In my case, the image browser, FotoAlbum is using 73 MB, Photoshop, 51 MB, etc., as shown in Fig. 143.

Modern motherboards for desktop use can normally address in the region of 1½-3 GB RAM, and that is more than adequate for most people. Server motherboards with special chipsets can address much more.

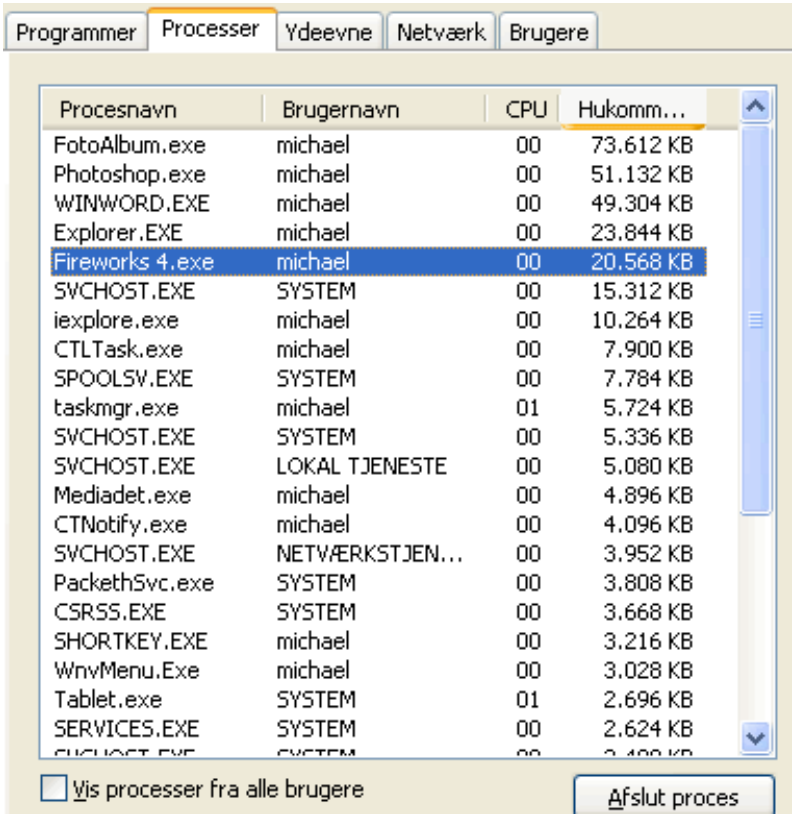


Figure 143. This window shows how much RAM each program is using (Windows XP).

Standard motherboards normally have a limited number of RAM sockets. If, for example, there are only three, you cannot use any more than three RAM modules (e.g. 3 x 256 MB or 3 x 512 MB).

CPU's have also always had an upper limit to how much RAM they can address:

Processor	Address bus width (bits)	Maximum System RAM
8088, 8086	20	1 MB
80286, 80386SX	24	16 MB
80386DX, 80486, Pentium, Pentium MMX, K5, K6 etc.	32	4 GB
Pentium Pro, Pentium II, III Pentium 4	36	64 GB

Figure 144. The width of the CPU's address bus determines the maximum amount of RAM that can be used.

Let me conclude this examination with a quote. It's about RAM quantity:

"640K ought to be enough for anybody."
Bill Gates, 1981.

- Next chapter.
- Previous chapter.

- Next chapter.
- Previous chapter.

Chapter 36. Chipsets and hubs

Since 1997, there has been more and more focus on refinement of the chipset, and not least the north bridge, which looks after data transfer to and from RAM. The south bridge has also been constantly developed, but the focus has been on adding new facilities.

For the north bridge, the development has focused on getting more bandwidth between the RAM and CPU. Let's look at a few examples of this.

Bridge or Hub

In a Pentium II motherboard, the I/O bus is directly linked to the system clock. The I/O bus (that is, in practise, the PCI bus) runs at 33 MHz, and that is typically a third or a quarter of the system clock speed (see Fig. 121).

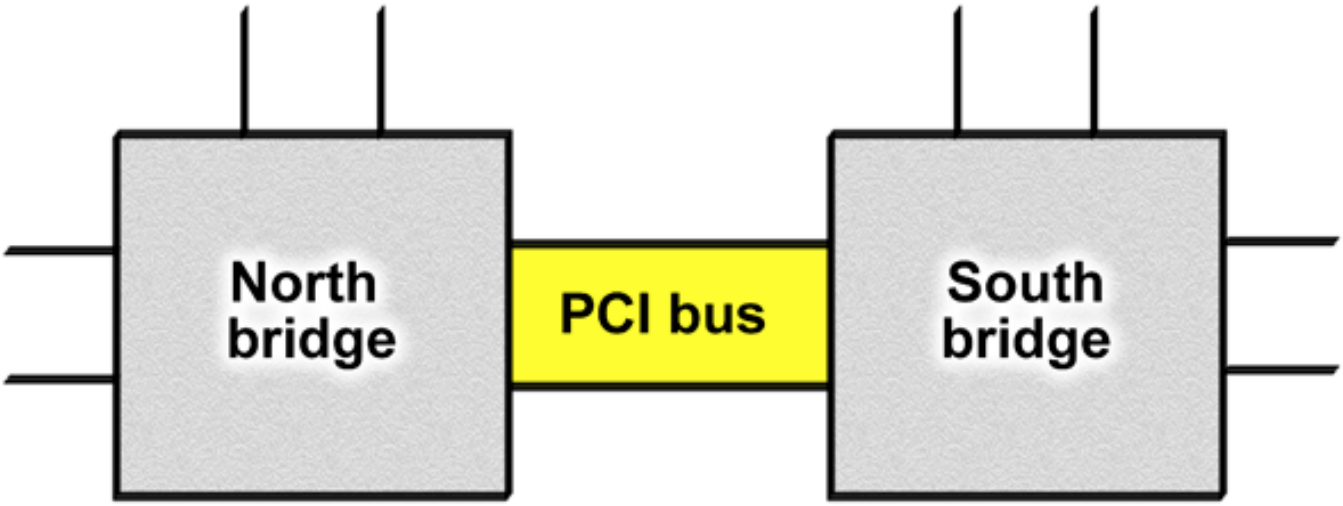


Figure 145. In this architecture (from the Pentium II chipset), the PCI bus connects to the chipset's two bridges.

In 1998-99, new developments took place at both AMD and Intel. A new architecture was introduced based on a Memory Controller Hub (*MCH*) instead of the traditional north bridge and an I/O Controller Hub (*ICH*) instead of the south bridge. I am using Intel's names here; the two chips have other names at AMD and VIA, but the principle is the same. The first Intel chipset with this architecture was called *i810*.

The MCH is a controller located between the CPU, RAM and AGP. It regulates the flow of data to and from RAM. This new architecture has two important consequences:

- The connection between the two *hubs* is managed by a special bus (*link channel*), which can have a very high bandwidth.

- The PCI bus comes off the ICH, and doesn't have to share its bandwidth with other devices.

The new architecture is used for both Pentium 4 and Athlon processors, and in chipsets from Intel, VIA, and others. In reality, it doesn't make a great deal of difference whether the chipset consists of hubs or bridges, so in the rest of the guide I will use both names indiscriminately.

Figure 146. The MCH is the central part of the i875P chip set.

The i875P chipset

In 2003, Intel launched a chipset which work with the Pentium 4 and dual channel DDR RAM, each running at 200 MHz. This chip set became very popular, since it had a very good performance.

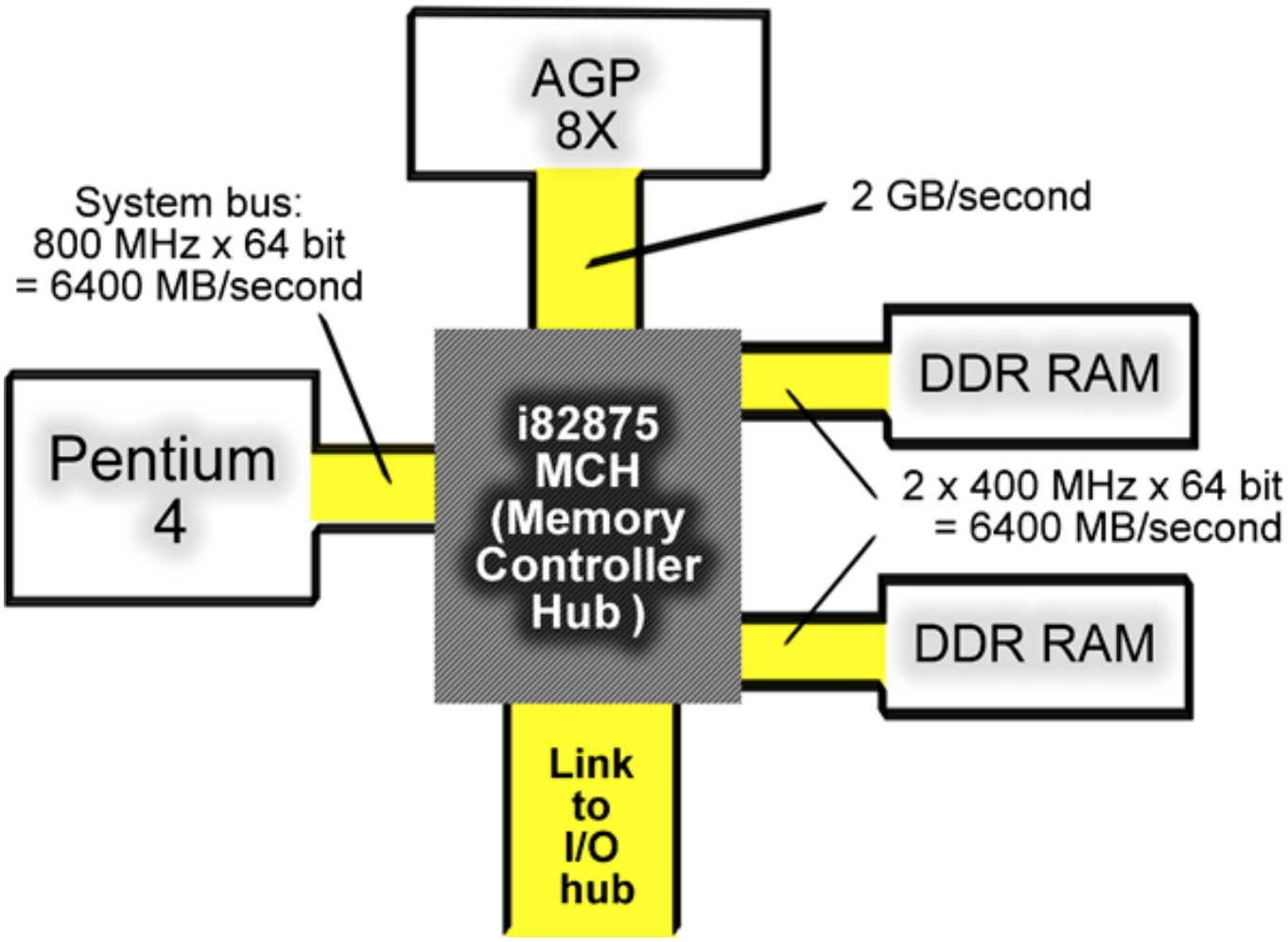


Figure 147. The architecture surrounding the Intel® 82875P Memory Controller Hub (MCH).

Another new feature in this chip set is that a Gigabit Ethernet controller can have direct access to the MCH (the north bridge). Traditionally the LAN controller is connected to the PCI bus. But since a gigabit network controller may consume a great band width, it is better to plug it directly into north bridge. This new design is called Communication Streaming Architecture (CSA).

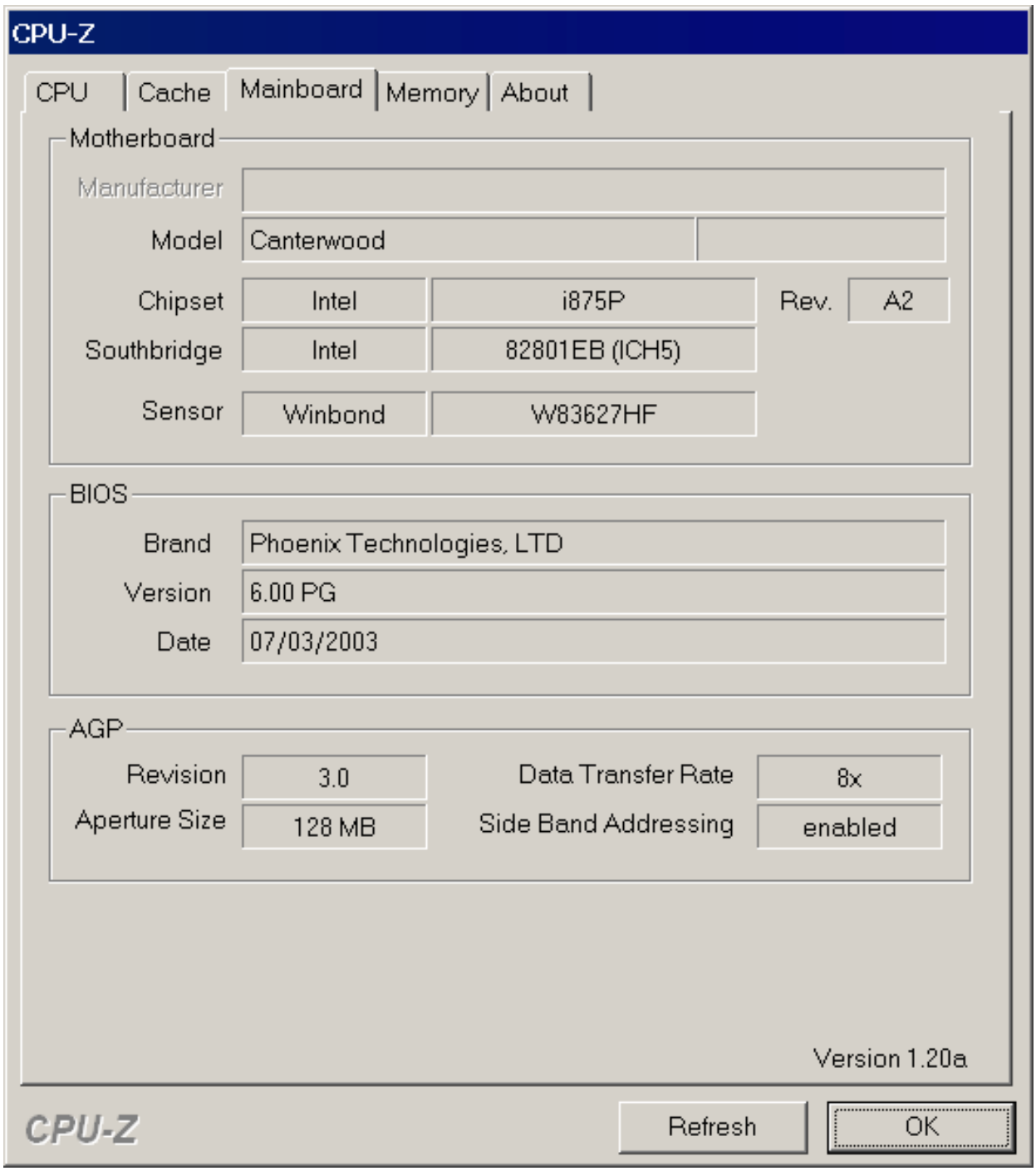


Figure 148. Report from the freeware program CPU-Z.

The i925 chipset

In late 2004 Intel introduced a new 900-series of chipsets. They were intended for the new generation of Pentium 4 and Celeron processors based on the LGA 775-socket (as in Figur 112. The chip sets comes with support for the PCI Express bus, which is replacing the AGP bus and with support of DDR2 RAM:

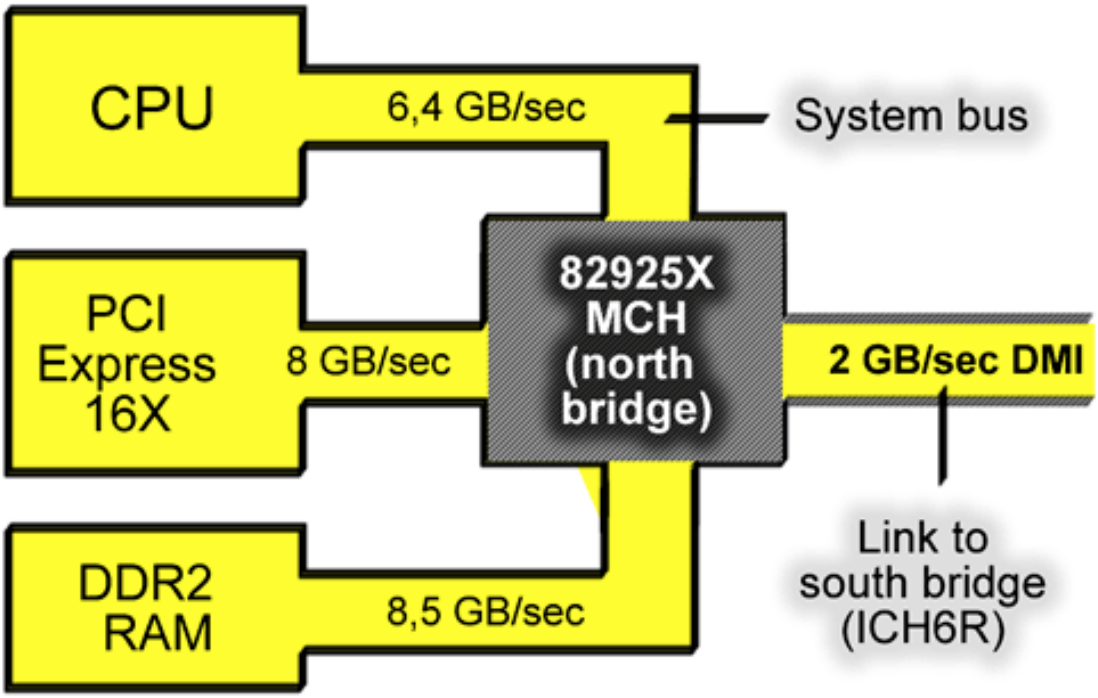


Figure 149. The new chipset architecture, where the north bridge has become a hub. Here Intel chip set i925.

By making use of dual channel DDR2 RAM, a bandwidth of up to 8.5 GB/sec is achieved.

Big bandwidth for RAM

One might be tempted to think that the bandwidth to the RAM ought to be identical with that of the system bus. But that is not the case. It would actually be good if it was higher. That's because the RAM doesn't only deliver data to the CPU. Data also goes directly to the graphics port and to and from the I/O devices – bypassing the CPU. RAM therefore needs even greater bandwidth. In future architectures we will see north bridges for both Pentium 4 and Athlon XP processors which employ more powerful types of RAM, such as 533 MHz DDR2.

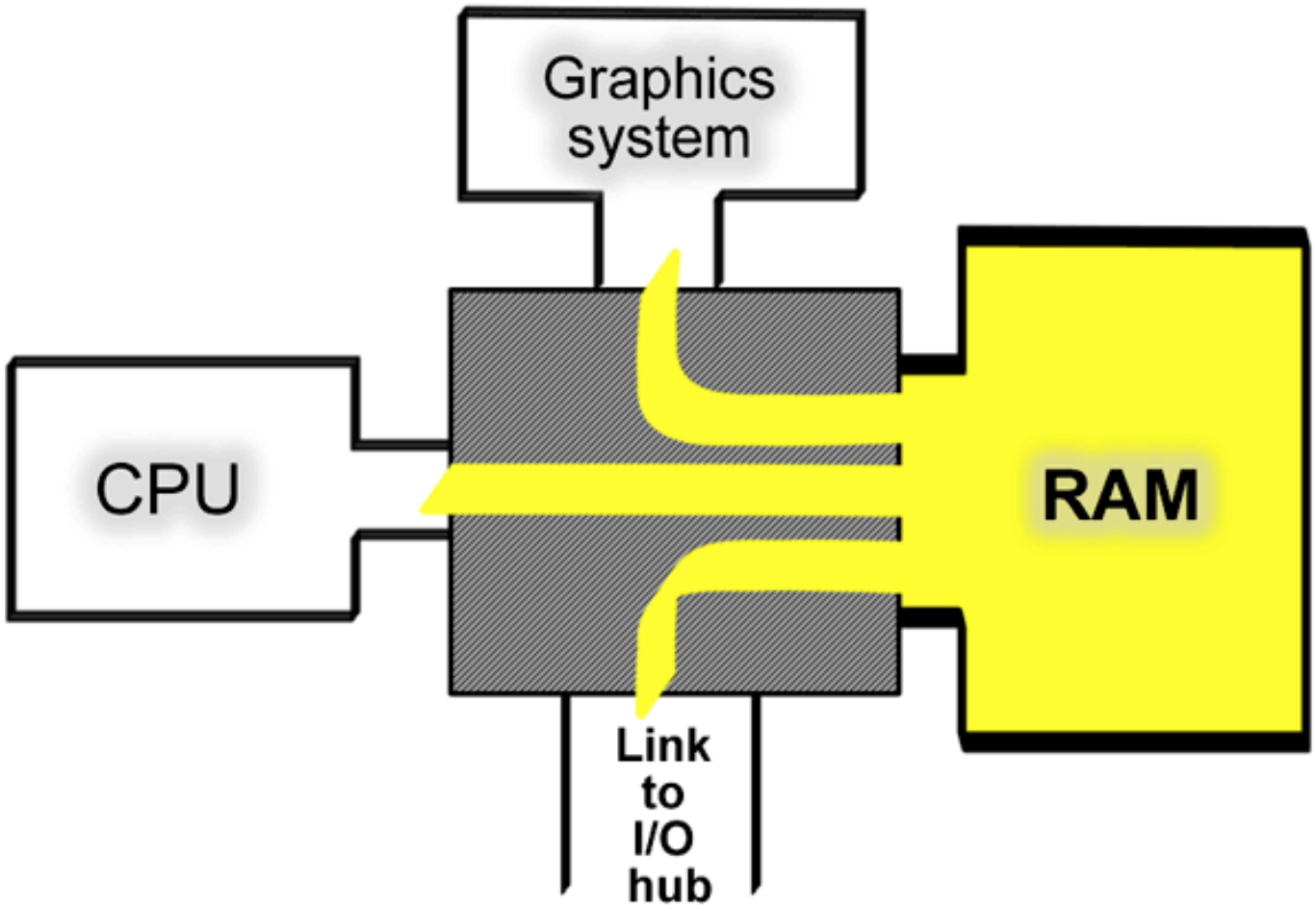


Figure 150. In reality, the RAM needs greater bandwidth than the CPU.

- Next chapter.
- Previous chapter.

- Next chapter.
- Previous chapter.

Chapter 37. Data for the monitor

I have several times mentioned the AGP port, which is directly connected to the CPU and RAM. It is a high-speed port used for the video card, which has it's own RAM access.

Figure 151. AGP is a high-speed bus for the video card, developed by Intel.

About screens and video cards

As users, we communicate with PC programs via the screen. The screen shows us a graphical representation of the software which is loaded and active in the PC. But the screen has to be fed data in order to show a picture. This data comes from the *video card*, which is a controller.

Screens can be analogue (the traditional, big and heavy CRT monitors) or digital devices, like the modern, flat TFT screens. Whatever the case, the screen has to be controlled by the PC (and ultimately by the CPU). This screen control takes place using a video card.

A video card can be built in two ways:

- As a plug-in card (an *adapter*).
- Integrated in chips on the motherboard.

Finally, the video card can be connected either to the PCI bus, the AGP bus or the PCI Express x16 bus.

Big bandwidth

Traditionally, the video card (also called the *graphics card*) was connected as an I/O device. This meant that data for the video card was transferred using the same I/O bus which looks after data transfer to and from the hard disks, network card, etc.

In the late 1990's, the demands placed on the graphics system increased dramatically. This was especially due to the spread of the many 3D games (like Quake, etc.). These games require enormous amounts of data, and that places demands on the bus connection. At that time, the video card was connected to the PCI bus, which has a limited bandwidth of 133 MB/sec. The same bus, as just mentioned, also looks after the hard disk and other I/O devices, which all need bandwidth. The PCI bus therefore became a bottleneck, and the solution to the problem was to release the video card from the I/O bus.

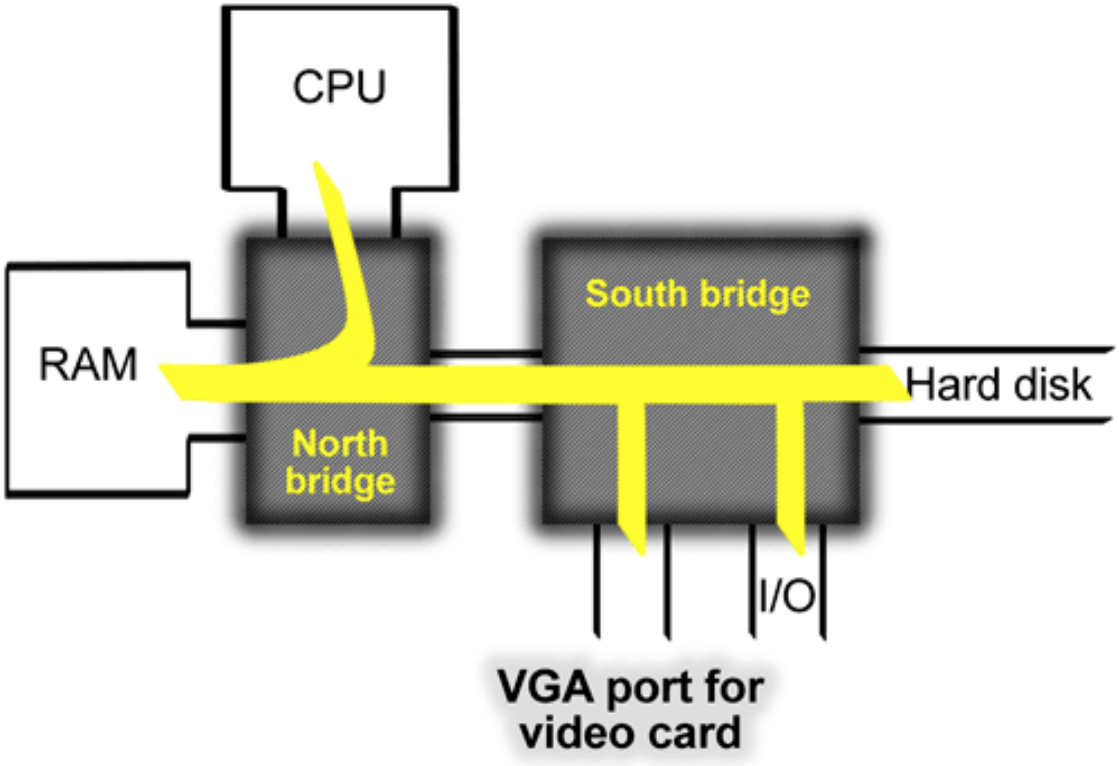


Figure 152. The data path to the video card before the AGP standard.

AGP

AGP (*Accelerated Graphics Port*) is a special I/O port which is designed exclusively for video cards. AGP was developed by Intel. The AGP port is located close to the chipset's north bridge.

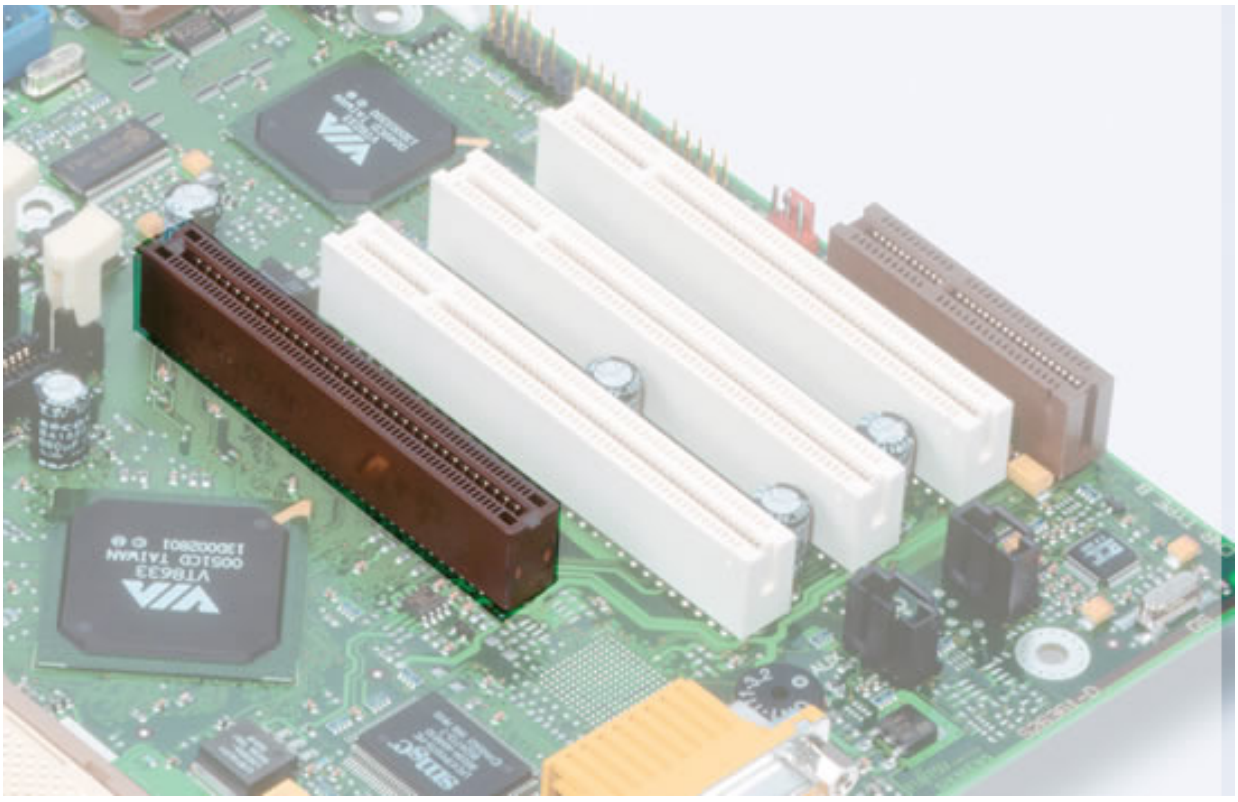


Figure 153. The AGP slot can be seen on this motherboard, left of the three PCI slots.

The video card port was moved from the south to the north bridge. The new architecture gives optimal access to RAM and hence to the bandwidth which the 3D games require.

At the same time, the PCI system is spared from the large amount of graphic data traffic to and from the video card. It can now focus on the other intensive transfer tasks, such as transfer to and from the network adapter and disk drive.

Figure 154. AGP Video card from ATI.

Technical details

AGP consists of a number of different technical elements, of which I will highlight two:

- A bus structure built around a double-clocked PCI bus.
- The ability to use the motherboard RAM as a *texture cache.*

The texture cache is used by games, and by giving access to the motherboard RAM, less RAM is needed on the cards.

The AGP bus is actually a 64-bit variant of the PCI bus. You can also see that on the surface, the motherboard AGP slot looks a fair bit like a PCI slot. But it is placed in a different position on the motherboard, to avoid confusion (see Fig. 156). The slot also has a different colour.

The first version of AGP was 1X, with a bandwidth of 254 MB/sec. But AGP was quickly released in a new mode, called 2X, with 508 MB/sec.

Later came 4X and 8X, which are the standards today. This involves a clock doubling, just as we have seen, for example, with DDR RAM. Two or four data packets are sent for each clock pulse. In this way, a bandwidth of 2,032 MB/sec has been reached.

Texture cache and RAMDAC

Textures are things like backgrounds in games. They can be uploaded directly from RAM to the video card. The system is called DIME (Direct Memory Execute). This allows the video card memory to be extended using standard RAM on the motherboard. In Fig. 155 you can see the system shown graphically.

In this figure you can also see the RAMDAC device. This is a chip on the video card which looks after the "translation" of digital data into *analogue* signals, when the card is connected to an analogue screen. The RAMDAC is a complete little processor in itself; the higher it's clock frequency, the higher the refresh rate with which the card can supply the screen image.

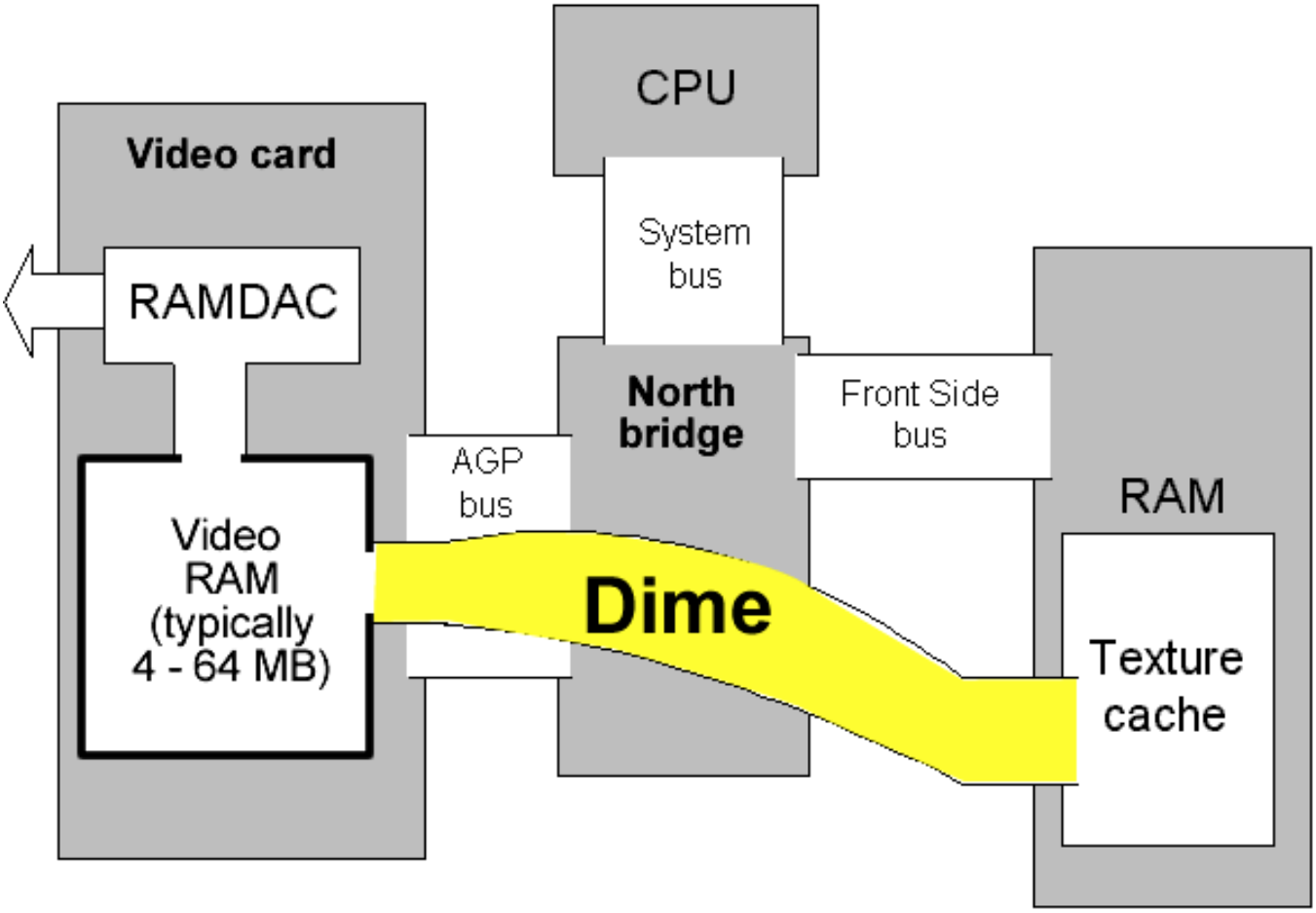


Figure 155. The AGP bus provides direct access to RAM.

Video card on PCI Express

With the new PCI Express bus, we get a new system for the video card. Replacing the AGP, the PCI Express X16-bus offers a transfer of 8 GB/sec, which leaves plenty of room for even the most graphical-intensive pc games.

Figure 156. The black PCI Express X16-slot to the left is for the video card.

- Next chapter.
- Previous chapter.

- Next chapter.
- Previous chapter.

Chapter 38. The PC's I/O system

There are a lot of I/O ports in the PC's architecture, with their associated I/O devices and standards. I/O stands for Input/Output, and these ports can both send and receive data from the processor and RAM.

The I/O system provides flexibility

The use of I/O devices has contributed to making the PC an incredibly flexible machine. Computers can be used for anything from normal office tasks, processing text and numbers, to image processing using scanners and cameras, to producing video, light and music.

The PC can also be used industrially. In 1987-88 I worked in a company that produced special PC's which could control the production of concrete. This was achieved using special I/O cards which could monitor the weighing of sand, gravel, cement and water. The core of the system was a standard office PC from Olivetti.

This particularly flexible architecture is based on an I/O system which can be extended almost without limit. This is one place we really see the PC's open architecture: any engineer or technician can, in principle, develop their own plug-in cards and other special devices, if they just meet one of the I/O standards. The opportunities for extension really are unlimited!

In the following chapters we will look at the various I/O buses which link the PC's other devices with the CPU and RAM.

19. Intro to the I/O system

During the last 10-15 years we have seen numerous technological innovations, the goal of which has been to increase the amount of traffic in the PC. This increase in traffic has taken place in the motherboard – with the system bus at the centre.

But the high clock frequencies and the large capacity for data transfer also affect the I/O system. Demands are being made for faster hard disks and greater bandwidth to and from external devices such as scanners and cameras. This has led to ongoing development of the I/O controllers in the south bridge.

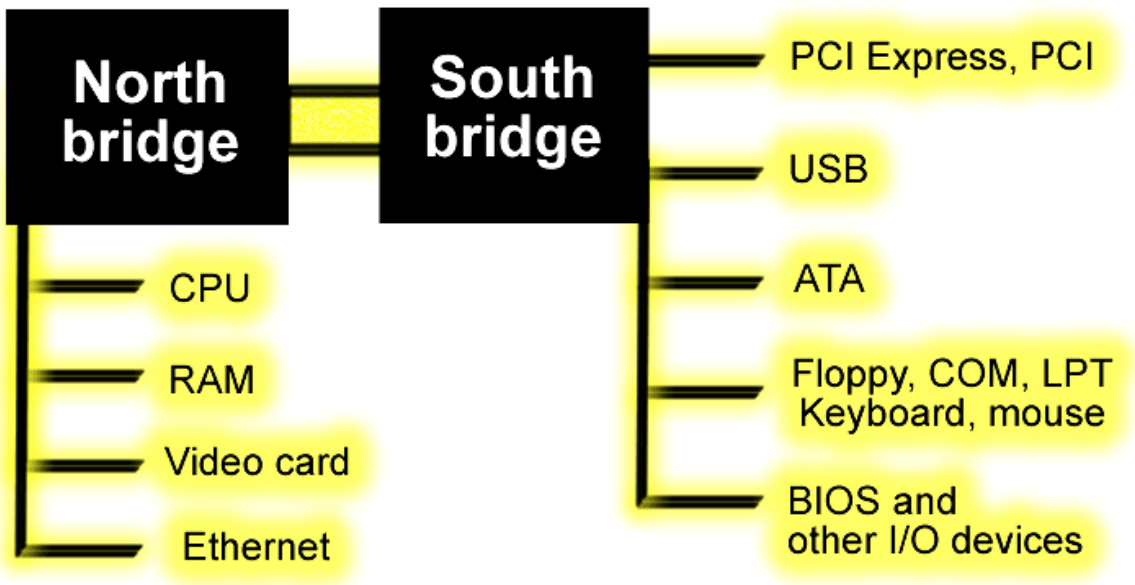


Figure 157. The south bridge connects a large number of different devices with the CPU and RAM.

You can work out the capacity of a data bus based on data width and clock frequency. Here is a brief comparison between the system bus and the I/O buses:

Bus	The north bridge's buses	The I/O buses
Variants	FSB, RAM, AGP, PCI Express X16, CSA	ISA, PCI, PCI Express, USB, ATA, SCSI, FireWire
Connects	CPU, RAM, Video, Ethernet	All other devices.
Clock freq.	66 - 1066 MHz	Typically 10-33 MHz.
Maximum capacity	> 3 GB/sec.	Typically 20-500 MB/sec. per bus

Figure 158. The system bus is much faster than any other bus.

Function of the I/O buses

As I already mentioned, the south bridge was introduced back in 1987 (see Fig. 119). The reason for this was that the I/O devices couldn't keep up with the high clock frequencies which the CPU and RAM could work at. There was electrical noise and other problems on the motherboard when the signals had to be carried to the plug-in cards, etc. Very few plug-in cards work at anything higher than 40 MHz – the electronics can't cope, the chips and conductors simply can't react that quickly. The I/O speed had to therefore be scaled down in relation to the system bus.

Since then, a number of different standards for I/O buses have been developed, which all emanate from the south bridge controllers. These include the older ISA, MCA, EISA and VL buses, and the current PCI, PCI Express and USB buses. The differences lie in their construction and architecture – in their connection to the motherboard.

I/O devices

The I/O buses run all over the motherboard, and connect a large number of different I/O devices. The south bridge controllers connect all the I/O devices to the CPU and RAM. Fig. 159 shows a brief overview of the various types of I/O devices. Note that AGP is not included, as it is tied to the north bridge.

Name	Devices
KBD, PS2, FDC, Game	Keyboard, mouse, floppy disk drive, joystick, etc.
ROM, CMOS	BIOS, setup, POST.
ATA	Hard disk, CD-ROM/RW, DVD etc.
PCI and PCI Express	Network card, SCSI controller, video grapper card, sound cards and lots of other adapters.

USB	Mouse, scanner, printers, modem, external hard disks and much more.
Firewire	Scanner, DV camera, external hard disk etc.
SCSI	Hard disks, CD-ROM drives, scanners, tape devices etc. (older)
LPT, COM	Parallel and serial devices such as printers, modems, etc.

Figure 159. Various types of I/O devices. The two last ones are not used much anymore.

The south bridge combines many functions

Originally, the various I/O devices could have *their own controller* mounted on the motherboard. If you look at a motherboard from the 1980's, there are dozens of individual chips – each with a particular function. As the years have passed, the controller functions have been gathered together into fewer and larger chips. The modern south bridge is now a large multi-controller, combining a number of functions previously managed by independent chips.

The south bridge is normally supplemented by a small *Super I/O* controller which takes care of a number of less critical functions that also used to be allotted to separate controller chips in the past. The Super I/O controller will be described in more detail later. It used to be connected to the south bridge via the ISA bus; in modern architectures the LPC (*Low Pin Count)* interface is used:

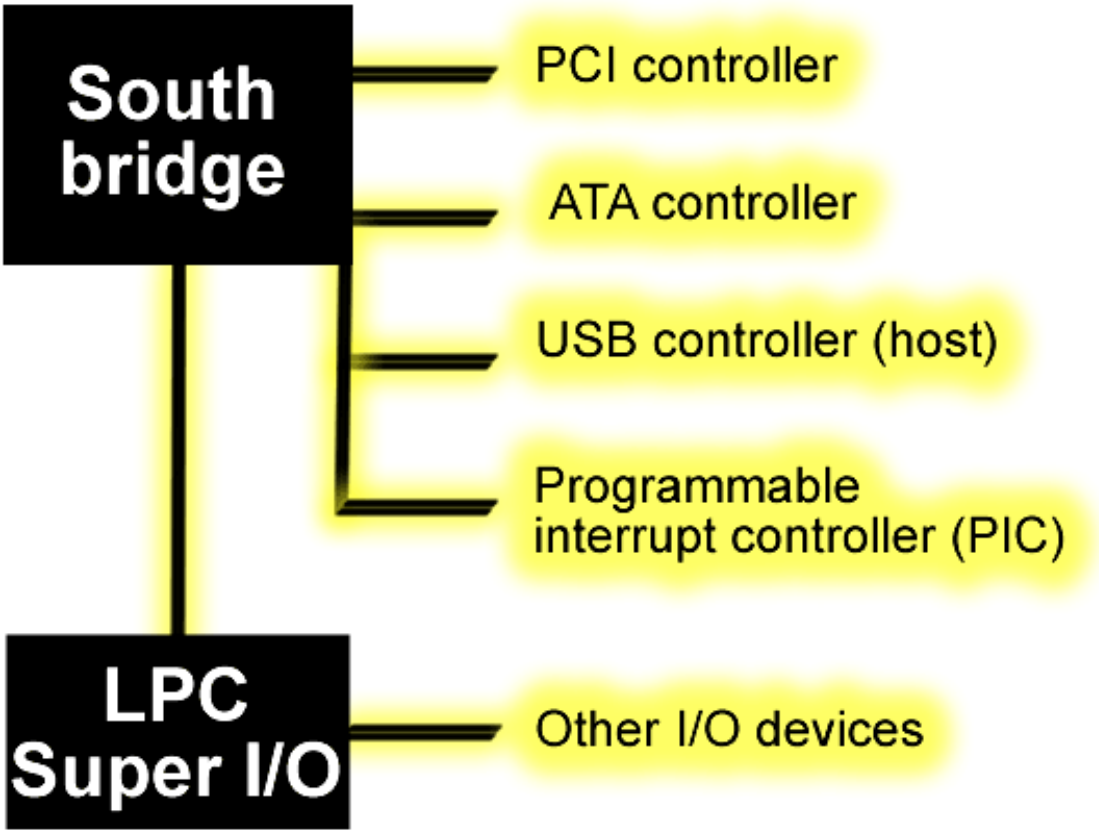


Figure 160. The south bridge is part of the chipset, but is supplemented by the small Super I/O controller.

Several types of I/O bus

Throughout the years, several different I/O buses have been developed. These are the *proper* I/O buses:

- The ISA bus – an old, low-speed bus which is not used much any more.

- The MCI, EISA and VL buses – faster buses which are also not used any more.
- The PCI bus – a general I/O bus used in more modern PC's.
- The PCI Express – the most modern bus.

These "real" I/O buses are designed for mounting plug-in cards (*adapters*) inside the actual PC box. They therefore connect to a series of sockets (*slots*) on the motherboard:

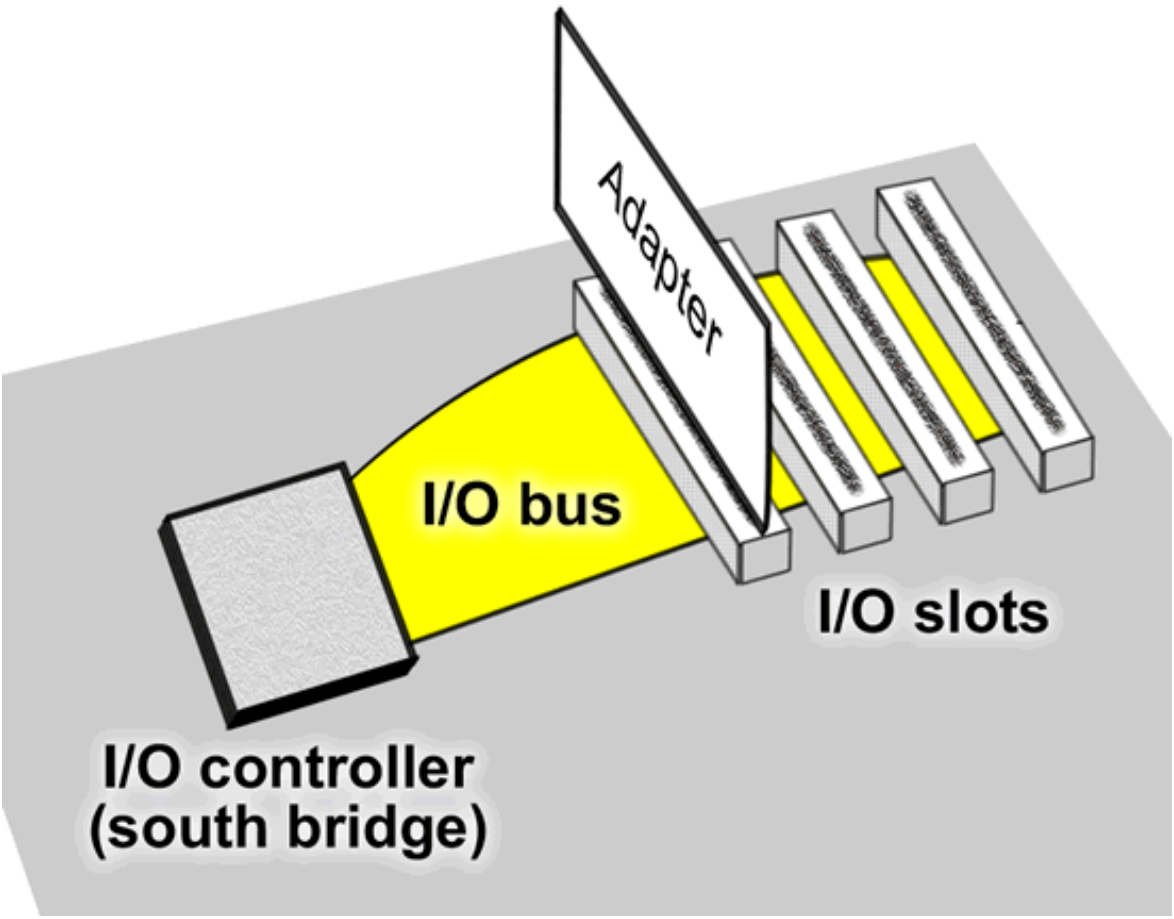


Figure 161. Plug-in cards are mounted in the I/O slots on the motherboard.

The various I/O buses have gradually replaced each other throughout the PC's history. Motherboards often used to have several buses, but today basically only the PCI bus is used for installing adapters such as network cards, etc.

Figure 162. The features in this motherboard are all available through the choice of chipset. Here the south bridge delivers many nice features.

Other types of bus

The PC always needs a low-speed bus, used for the less intensive I/O devices. For many years that was the job of the ISA bus, but it has been replaced today by USB (Universal Serial Bus). USB is not a traditional motherboard bus, as it doesn't have slots for mounting plug-in cards. With USB, *external* devices are connected in *a series*. More on this later.

SCSI and FireWire are other types of high-speed bus which give the PC more expansion options. They are not part of the standard PC architecture, but they can be integrated into any PC. This is normally done using a plug-in card for the PCI bus, on which the SCSI or FireWire controller is mounted. Thus the two interfaces draw on the capacity of the PCI bus:

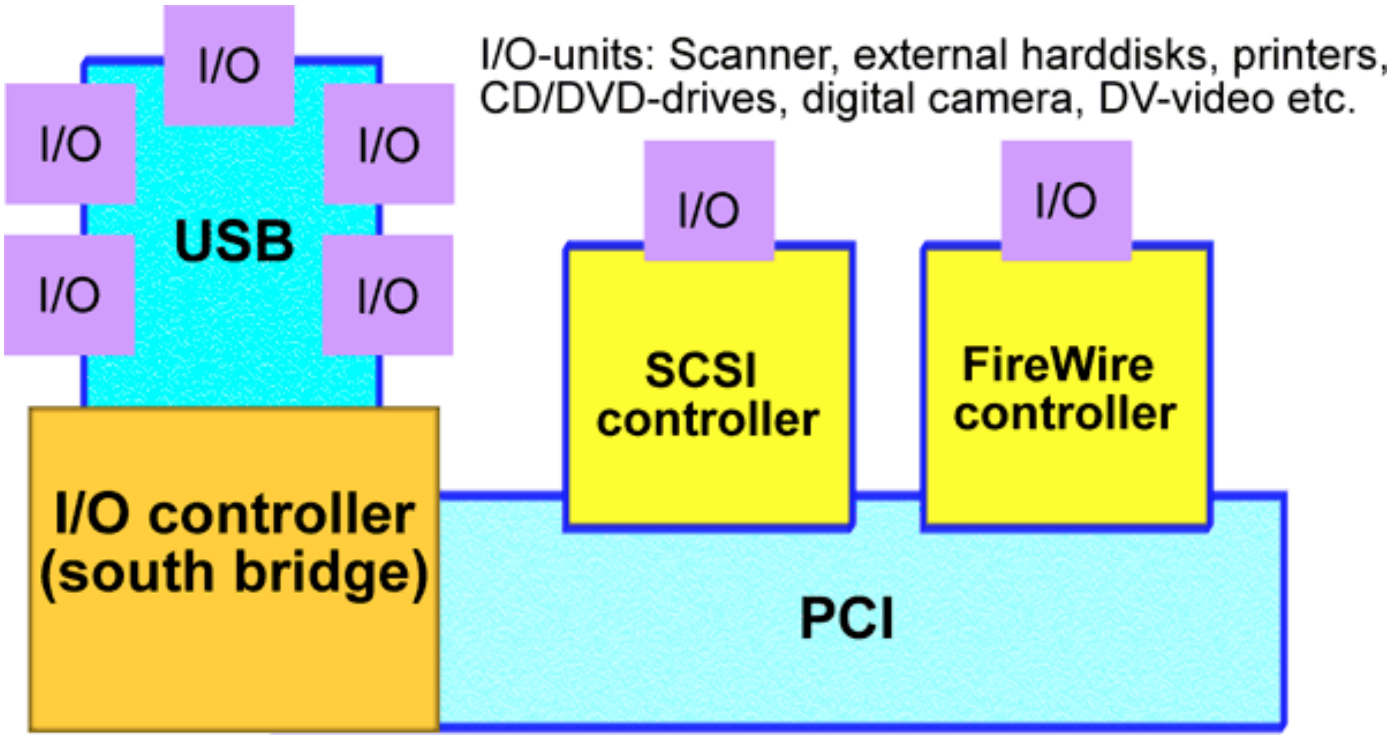


Figure 163. SCSI and FireWire controllers are both normally connected to the PCI bus.

Finally, I should also mention the ATA hard disk interface, which is not normally called a bus, but which really could be called one. The ATA interface is only used for *drives*, and the standard configuration allows four devices to be connected directly to the motherboard, where the hard disk's wide cable, for example, fits inside an ATA connector.

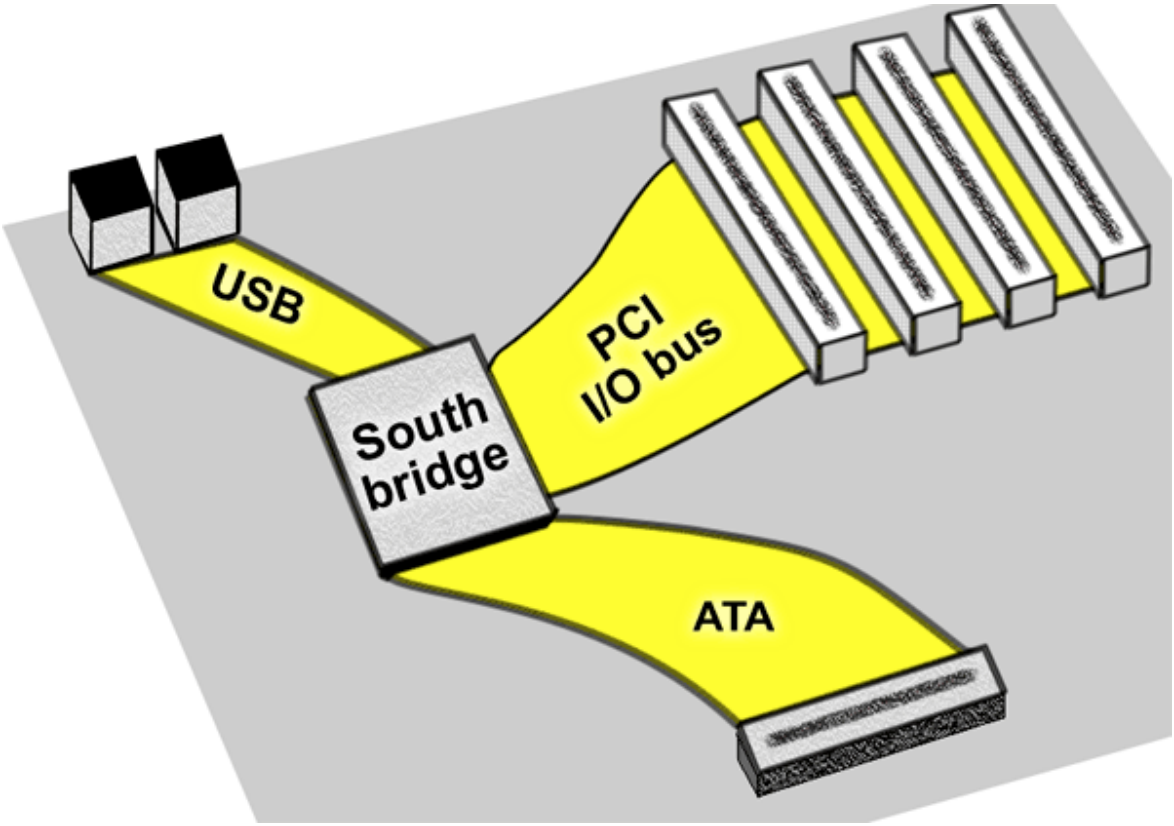


Figure 164. The ATA interface works like a bus in relation to the south bridge.

- Next chapter.
- Previous chapter.

- Next chapter.
- Previous chapter.

Chapter 39. From ISA to PCI Express

From about 1984 on, every PC had a standard bus which was used for I/O tasks. That was the ISA (*Industry Standard Architecture*) bus.

Right up until about 1999 there were still ISA slots in most PC's. In the later years, however, they were only kept for compatibility, so that plug-in cards of the old ISA type could be re-used. This was particularly the case for *sound cards* from SoundBlaster; they worked quite well on the ISA bus, and many games were programmed to directly exploit this type of hardware. It therefore took many years to get away from the ISA bus, but we have managed to now.

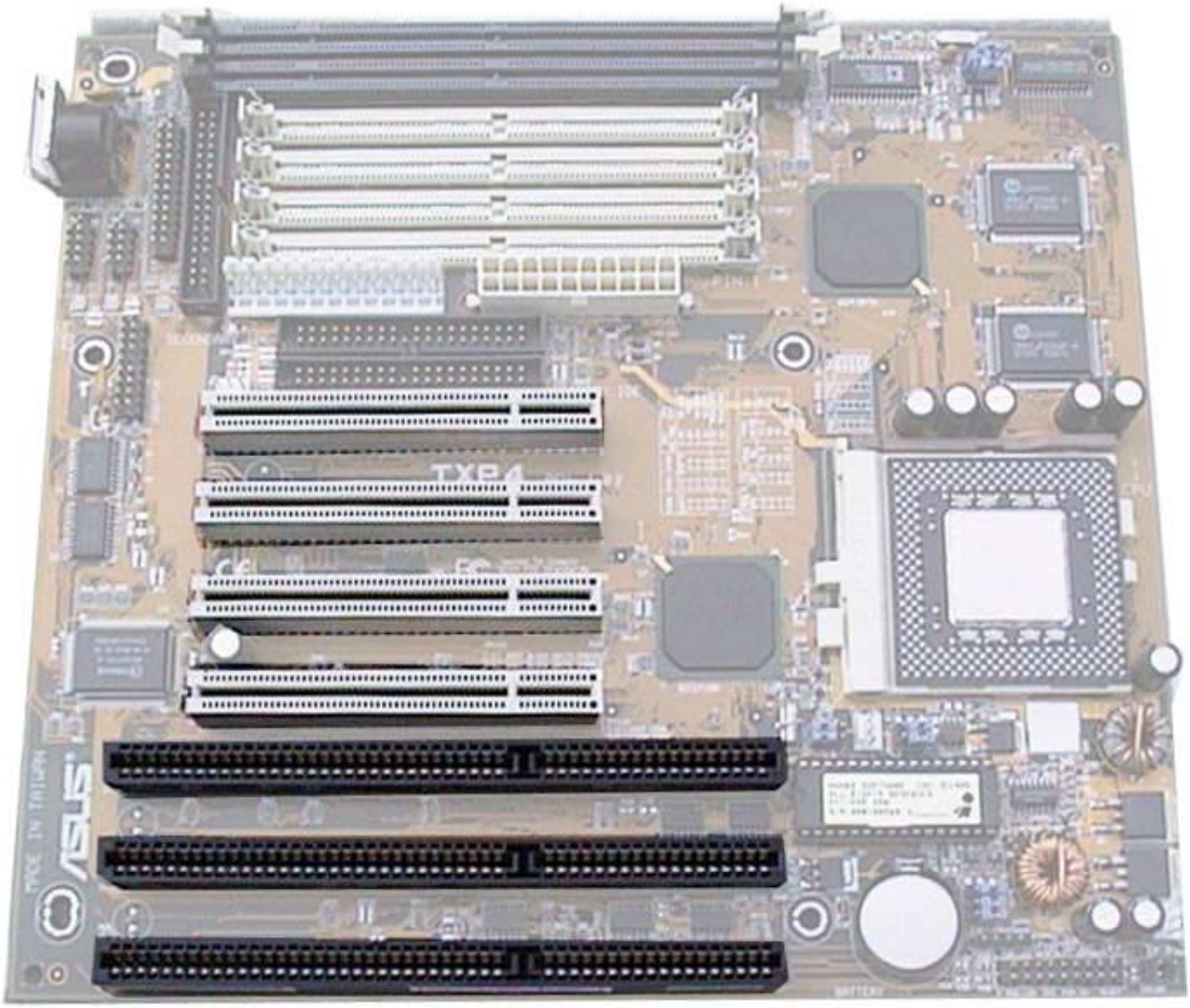


Figure 165. Motherboard from 1998 with three (black) ISA slots and four (white) PCI slots.

History of the ISA bus

The ISA bus is thus the I/O bus which survived the longest. Here is some information about it:

ISA was an improvement to IBM's original XT bus (which was only 8 bits wide) IBM used the protected name "AT Bus", but in everyday conversation it was called the ISA bus.
The ISA bus was 16 bits wide, and ran at a maximum clock frequency of 8 MHz.
The bus has a theoretical bandwidth of about 8 MB per second. However in practise it never exceeds about 1-2 MB/sec. – partly because it takes 2-3 of the processor's clock pulses to move a packet (16 bits) of data.

Figure 166. The ISA bus is not used much today, but it had enormous significance in the years up until the middle of the 1990's.

The ISA bus had two "faces" in the old PC architecture:

- An internal ISA bus, which the simple ports (the keyboard, floppy disk and serial/parallel ports) were connected to.
- An external expansion bus, to which 16-bit ISA adapters could be connected.

Sluggish performance

One of the reasons the ISA bus was slow was that it only had 16 data channels. The 486 processor, once it was introduced, worked with 32 bits each clock pulse. When it sent data to the ISA bus, these 32-bit packets (*dwords* or *doublewords*) had to be split into two 16-bit packets (two *words*), which were sent one at a time, and this slowed down the flow of data.

Bus	Time per packet	Amount of data per packet
ISA	375 ns	16 bits
PCI	30 ns	32 bits

Figure 167. The PCI bus was a huge step forward.

The ISA bus was not "intelligent" either, since it was in principle the CPU which controlled all the work the bus was doing. This meant the CPU could only begin a new task when the transfer was complete. You may perhaps have experienced yourself, that when your PC works with the floppy disk drive – the rest of the PC virtually grinds to a halt. That's the ISA bus's fault, and it therefore only happens on older PC's.

These small delays are called *wait states.* If an ISA adapter cannot keep up with the data it is receiving, it sends wait states to the CPU. These are signals to the CPU telling it to do nothing. A wait state is a wasted clock pulse – the CPU skips over a clock pulse, without doing anything. Thus a slow ISA adapter could choke any PC.

Another problem was that the ISA bus often played up when you were installing an expansion card (e. g. a sound card). Many of the problems were associated with the handling of IRQ's and DMA channels (I will explain these terms later), which often had to be done "by hand" with the old ISA bus.

Every device takes up one particular IRQ, and possibly a DMA channel, and conflicts often arose with other devices. It was a big relief when Intel, in the late 1990's, finally dropped the ISA bus and replaced it with the smart USB bus.

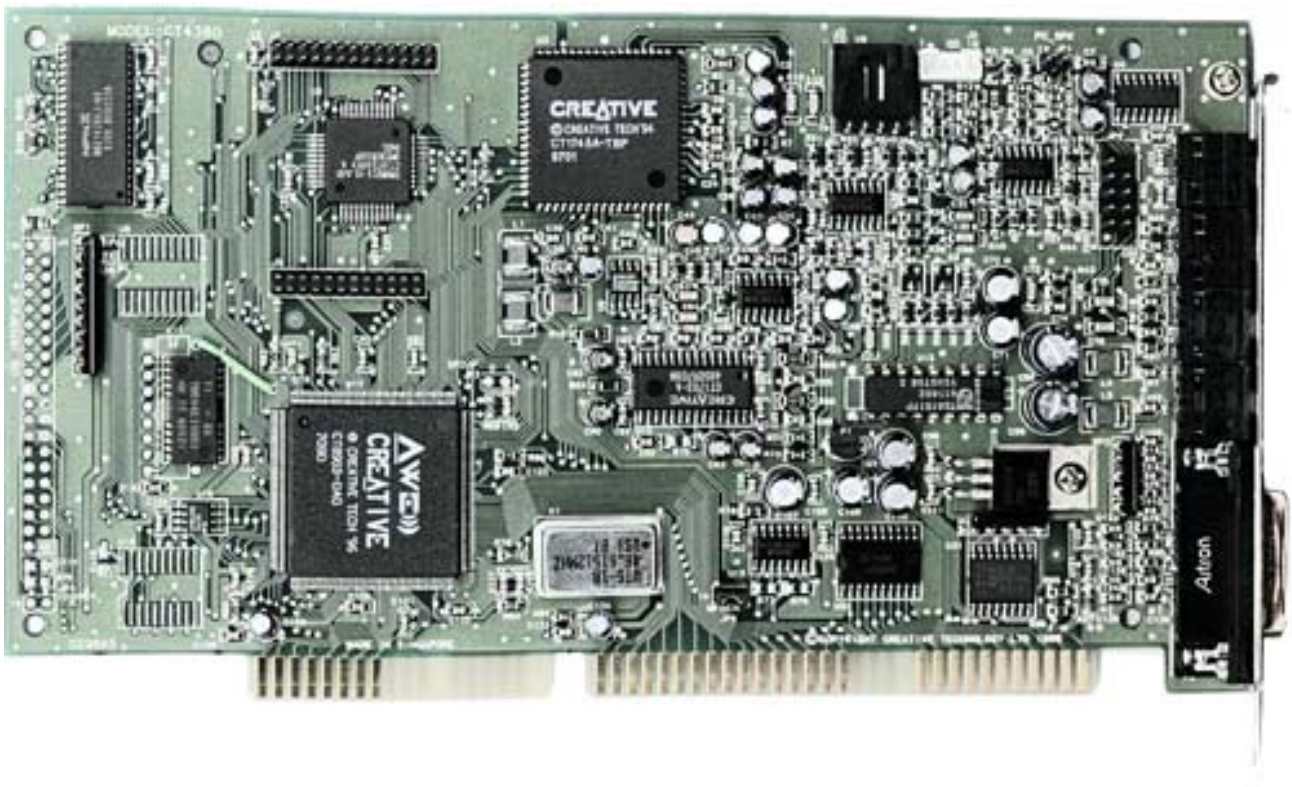


Figure 168. ISA based Sound Blaster sound card.

The MCA, EISA and VL buses

The ISA bus was too slow, and the solution was to develop new standards for I/O devices. In 1987-88, two new I/O buses were put forward. First, IBM brought out their technologically advanced MCA bus. But since it was patented, a number of other companies pooled together to create the equivalent EISA bus.

But neither MCA or EISA had a big impact on the clone market. We were stuck with the ISA bus up until 1993, when the VL bus finally became reasonably widespread. It was a genuine *local bus*, which means it worked synchronously with the system bus. The VL bus was very primitive; it was really just an extension of the system bus.

The VL bus never managed to have a big impact, because almost at the same time, the robust and efficient PCI bus broke through. The various I/O buses are summarised below:

Bus	Description
PC-XT from 1981	Synchronous 8-bit bus which followed the CPU clock frequency of 4.77 or 6 MHz. Band 170: 4-6 MB/sec.
ISA (PC-AT) from 1984	Simple, cheap I/O bus. Synchronous with the CPU. Band width: 8 MB/sec.

MCA from 1987	Advanced I/O bus from IBM (patented). Asynchronous, 32-bit, at 10 MHz. Band width: 40 MB/sec.
EISA From 1988	Simple, high-speed I/O bus. 32-bit, synchronised with the CPU's clock frequency: 33, 40, 50 MHz. Band width: up to 160 MB/sec.
PCI from 1993	Advanced, general, high-speed I/O bus. 32-bit, asynchronous, at 33 MHz. Band width: 133 MB/sec.
USB and Firewire, from 1998	Serial buses for external equipment.
PCI Express from 2004	A serial bus for I/O cards with very high speed. Replaces PCI and AGP. 500 MB/sec. per. Channel.

Figure 169. The PC's I/O buses, throughout the years.

The PCI bus

PCI stands for Peripheral Component Interconnect. The bus is an Intel product which is used in all PC's today, and also in other computers, as the PCI bus is processor independent. It can be used with all 32-bit and 64-bit processors, and is therefore found in many different computer architectures.

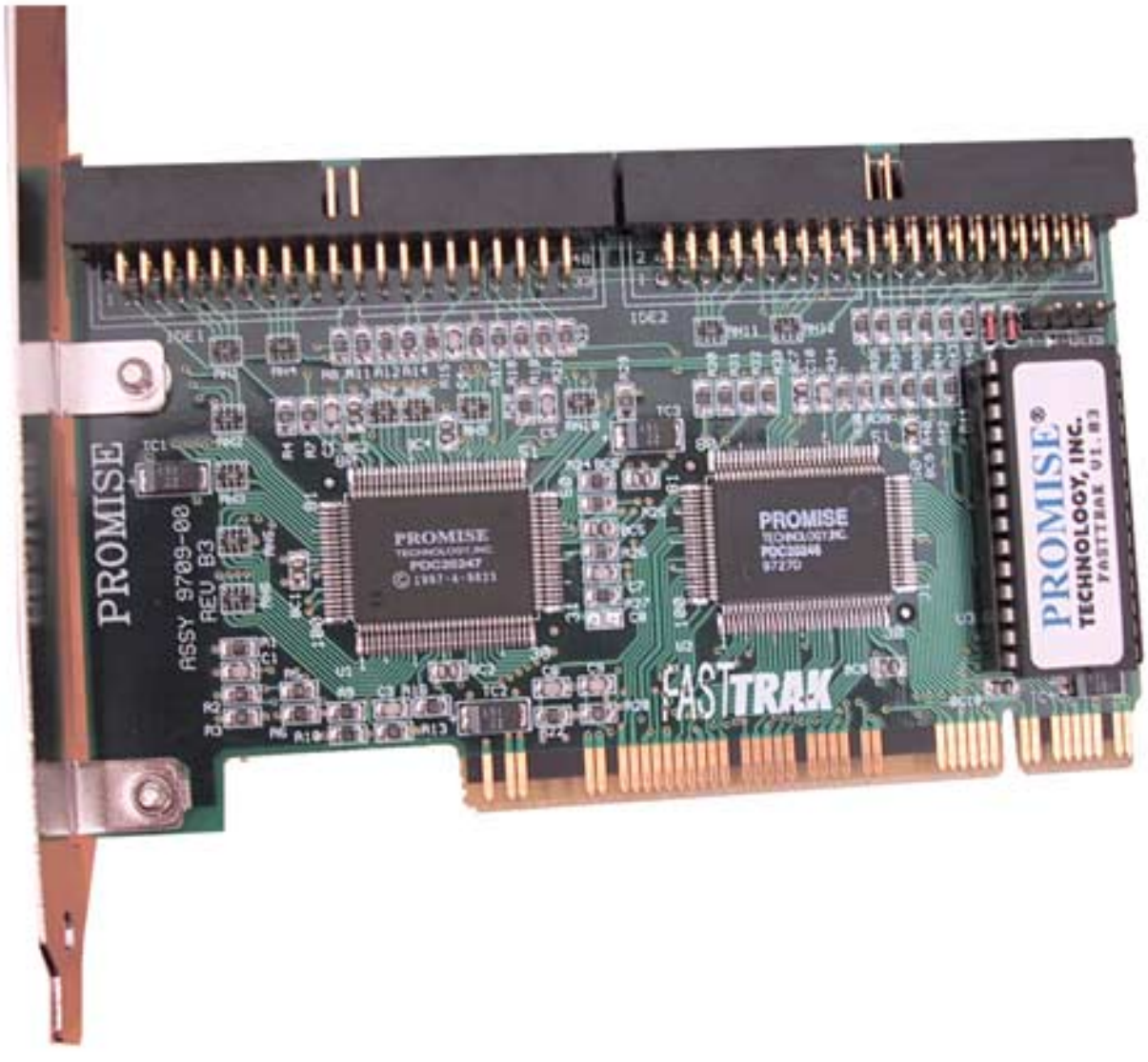


Figure 170. PCI bus adapter.

At the same time, the bus is compatible with the ISA bus to a certain extent, since PCI devices can react to ISA bus signals, create the same IRQ's etc. One consequence of this was that Sound Blaster compatible sound cards could be developed, which was very important in the middle of the 1990's. In optimal conditions, the PCI bus sends one packet of data (32 bits) each clock pulse. The PCI bus therefore has a maximum bandwidth of 132 MB per second, as shown below:

Clock frequency:	33 MHz
Bus width:	32 bits
Bandwidth:	32 bits x 33,333,333 clock pulses/second = 4 bytes x 33,333,333 clock pulses/second = 132 MB per second

Figure 171. The maximum bandwidth of the PCI bus.

There is also a more powerful versions of the PCI standard, which provides greater bandwidth, but most motherboards still use the original version. The PCI bus has a buffer which operates between the CPU and the peripheral devices (a kind of cache RAM). This allows the CPU to deliver its data to the buffer, and then perform other tasks. The bus looks after the rest of the delivery itself at its own pace. Alternatively, PCI adapters can also deliver data to the buffer, whether or not the CPU has time to process it. The data just stands in a queue and waits until there is room on the system bus, which then relays it to the CPU.

As a result of all this, the peripheral PCI devices operate *asynchronously* – at their own pace – in relation to the CPU. Thus the PCI bus (in contrast to the VL bus) is not a *local bus* from a technical perspective.

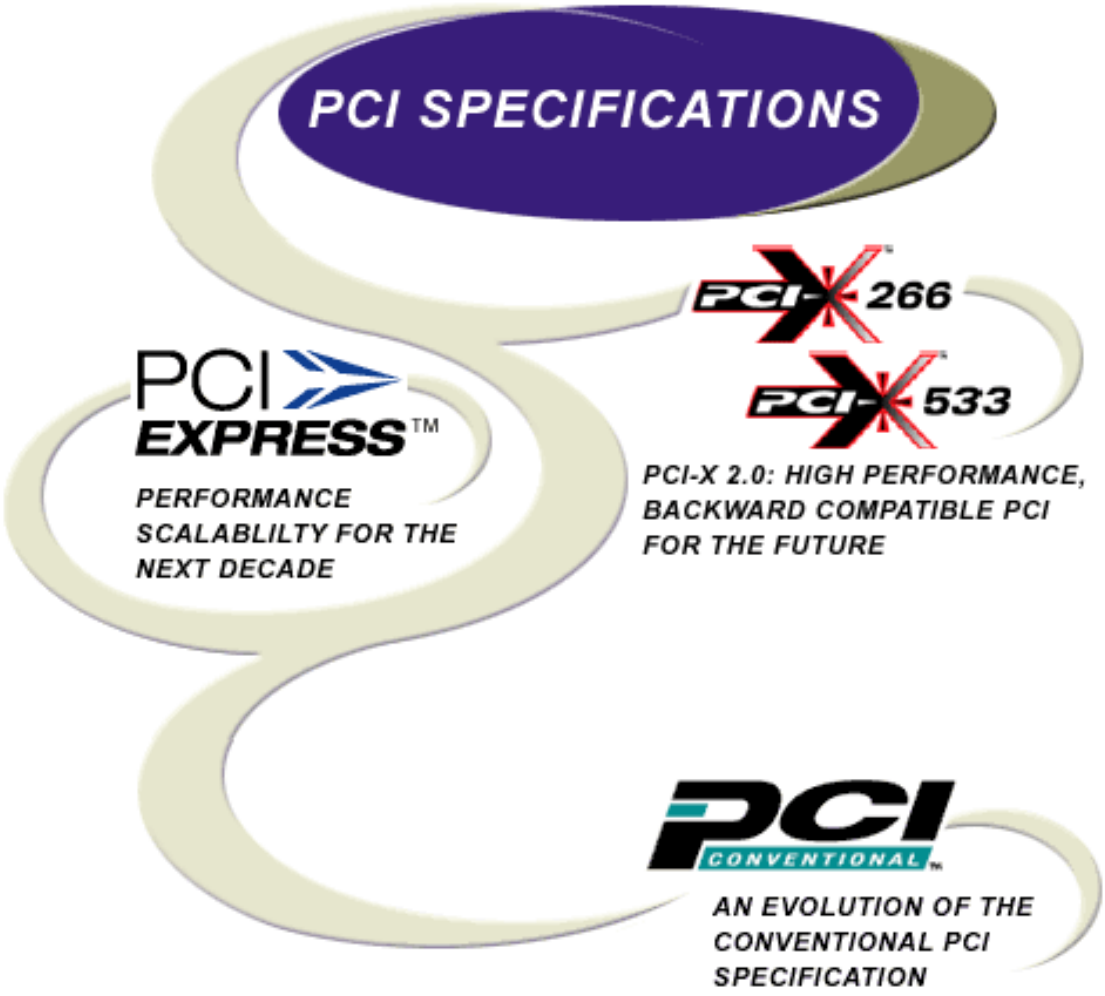


Figure 172. The PCI bus is being refined by a Special Interest Group. You can follow their progress yourself on the Net (www.pcisig.com).

Plug and Play

The Plug and Play standard is part of the PCI specification. It means that all PCI adapter cards are *self-configuring*. The specification for Plug and Play was developed by Microsoft and Intel, among others, and the ideas was (as the name suggests) to provide a system where one can simply install an adapter and it will work. It's not quite this simple in practise; a software driver has to normally be installed before an adapter will work. But the actual cooperation between adapter, motherboard and operating system – happens automatically. During startup, communication takes place between the PC's startup programs, the PCI controller and each *PCI device* (adapter).

The adapter has to be able to inform the I/O bus which I/O addresses and IRQ's it can operate with. And it has to be able to configure itself to use the resources allocated to it by the I/O bus. When the exercise is successful, the adapter is configured automatically, and is ready to be used by the operating system.

All the components involved (adapter, motherboard and Windows) have to be Plug and Play compatible for the system to work.

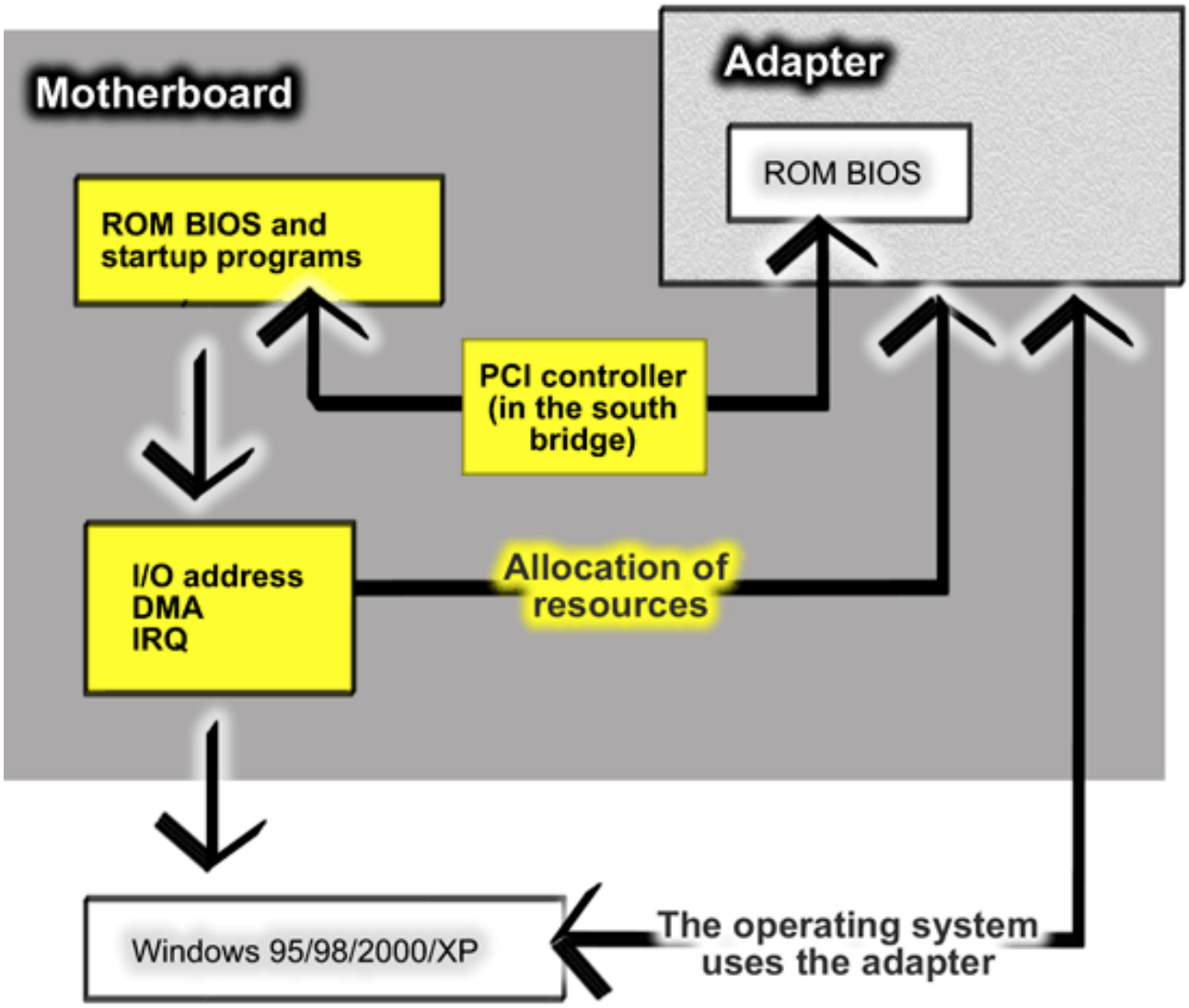


Figure 173. Schematic overview of Plug and Play.

ESCD

In connection with Plug and Play, it is necessary also to discuss the *Extended System Configuration Data* (ESCD) system. This is a small data area which is stored in the motherboard's CMOS storage.

The ESCD store is used to save adapter configuration information. This means the motherboard doesn't have to go through the whole plug and play operation at each startup – information about the PC's configuration can be read from the CMOS storage.

The ESCD also allows the user to manually allocate an IRQ etc., for a particular adapter. This can be done using the motherboard's setup program. Read more about the CMOS on page 90.

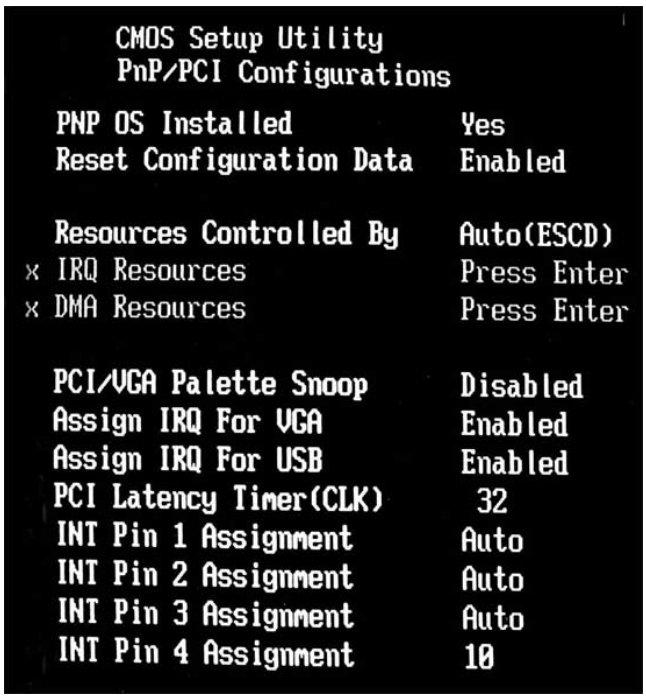


Figure 174. Using the CMOS setup program, the user can directly allocate resources for PCI adapters.

See the devices during startup

All I/O devices have small, built-in registers (in ROM circuits), which contain various information. The registers can, for example, describe the nature of the device (e.g. a network card, video card or SCSI controller, etc.) and the manufacturer. You can see this information for yourself during PC startup, when the devices are configured:

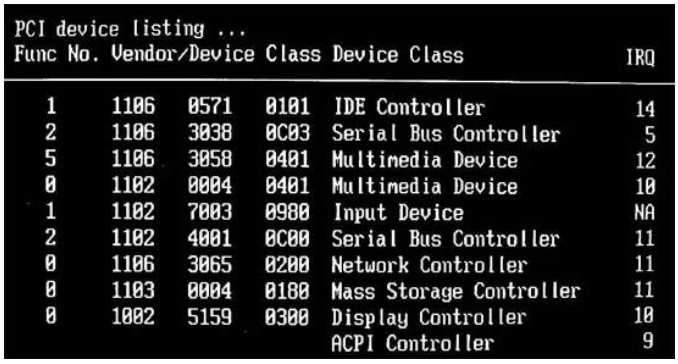


Figure 175. The PCI devices ”identify themselves” during startup as they are configured.

PCI Express development

In 2004 a new PCI bus was introduced. The PCI Special Interest Group (see www.pcisig.com) consists of the most important companies (Intel, IBM, Apple, etc.), who coordinate and standardize the bus via this forum. The PCI Express is the successor to the PCI bus. This is a completely new type of I/O bus using a serial connection (like the buses USB, Firewire and SATA).

This new I/O bus will be extremely scalable, as it works with a large number of channels (X1, X2, X16 etc.), each of which has a bandwidth of about 250 MB/second in both directions, simultaneously.

The standard plans for the use of plug-in cards and devices in various categories, with varying bandwidths and power consumption. A 16X video card, for example, will totally be able to pull about 8 GB/sec.

PCI Express is based on a serial architecture, making it possible to develop smaller and cheaper devices with many fewer pins. The new I/O bus will initially co-exist with the PCI interface, as we se it in the motherboards with Intel i900-series chip sets. But the goal is that PCI Express should replace both PCI and AGP.

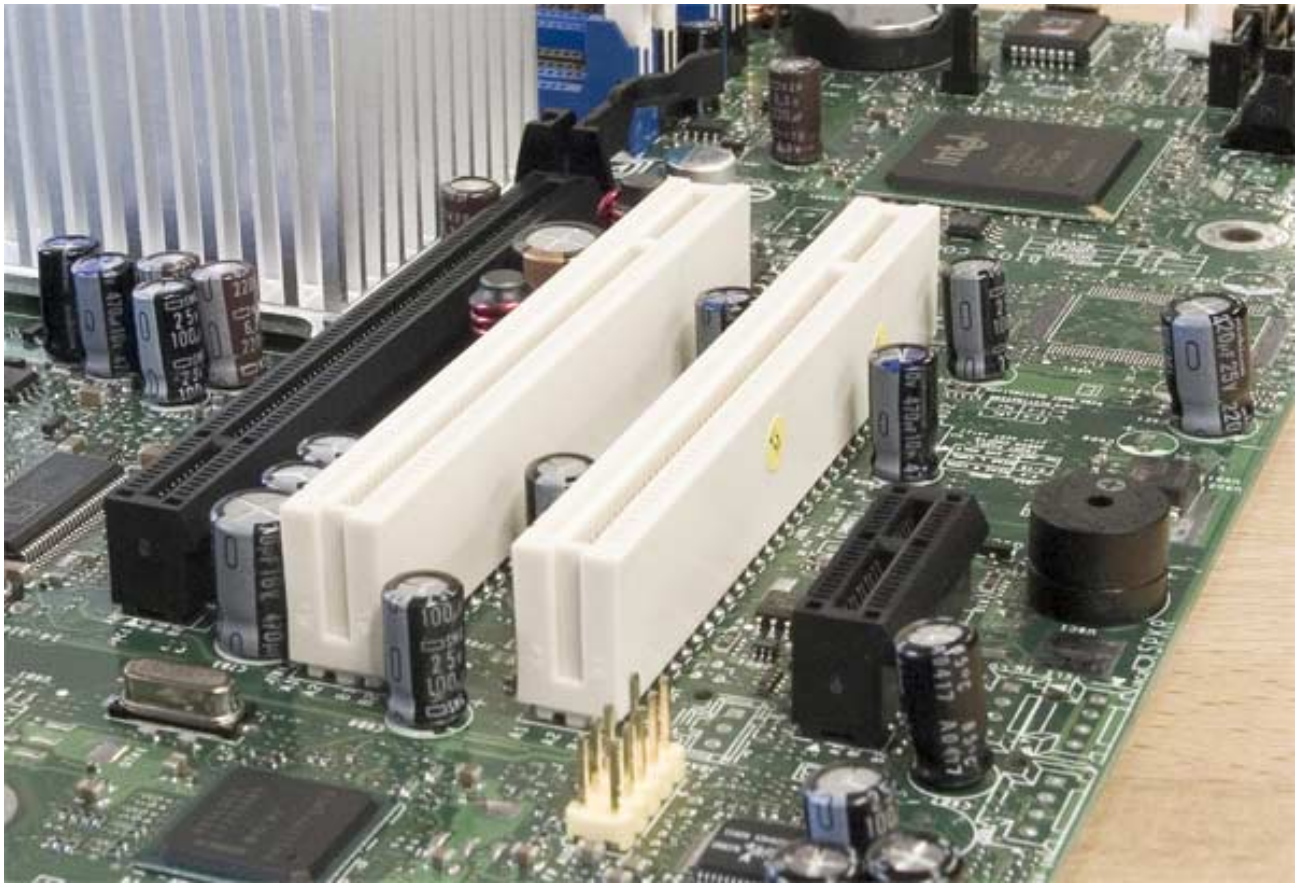


Figure 176. The two black slots are for PCI Express cards. To the left a 16X card for the graphics controller and to the right a smaller 1X slot. Inbetween you se two traditional PCI slots.

- Next chapter.
- Previous chapter.

- Next chapter.
- Previous chapter.

Chapter 40. I/O buses using IRQ's

I/O buses are designed to move data. Data is transferred from one I/O device to another, and from the I/O devices to the CPU and RAM. Let's delve a little deeper into the architecture which makes this possible.

At the physical level, the PCI bus consists of a number of parallel printed tracks on the motherboard. These tracks are used for:

- Data tracks, each of which can move one bit at a time.
- Address tracks, which indicate where the data should be sent.
- Tracks for IRQ's, DMA, clock pulses, power, etc.

Below we will take a look at the concepts of IRQ, DMA and I/O addresses.

Hardware interrupts

When you install a plug-in card in a slot, the card gets connected to the I/O bus. That means that data can run to and from the card. To manage the traffic on the I/O buses, a system is used, called IRQ. This stands for *Interrupt Request*.

Interrupts are basically one of the foundational principles of the PC architecture. There are two types: *Software interrupts* are used to call a large number of BIOS routines. *Hardware interrupts* are what we are now going to discuss, because these are what the I/O devices use.

An interrupt signal is a doorbell used by the various PCI adapters and other devices. By setting an IRQ signal, they can ask to "have the floor" – to be allowed to interrupt the CPU in the middle of its work. The device places a voltage across one of the conductors on the bus – it sets an IRQ. When the CPU registers the signal, it knows that the device wants to fetch or deliver data, or has finished doing so.

The IRQ signals are managed by a PIC (*Programmable Interrupt Controller*), which is a controller device on the motherboard.

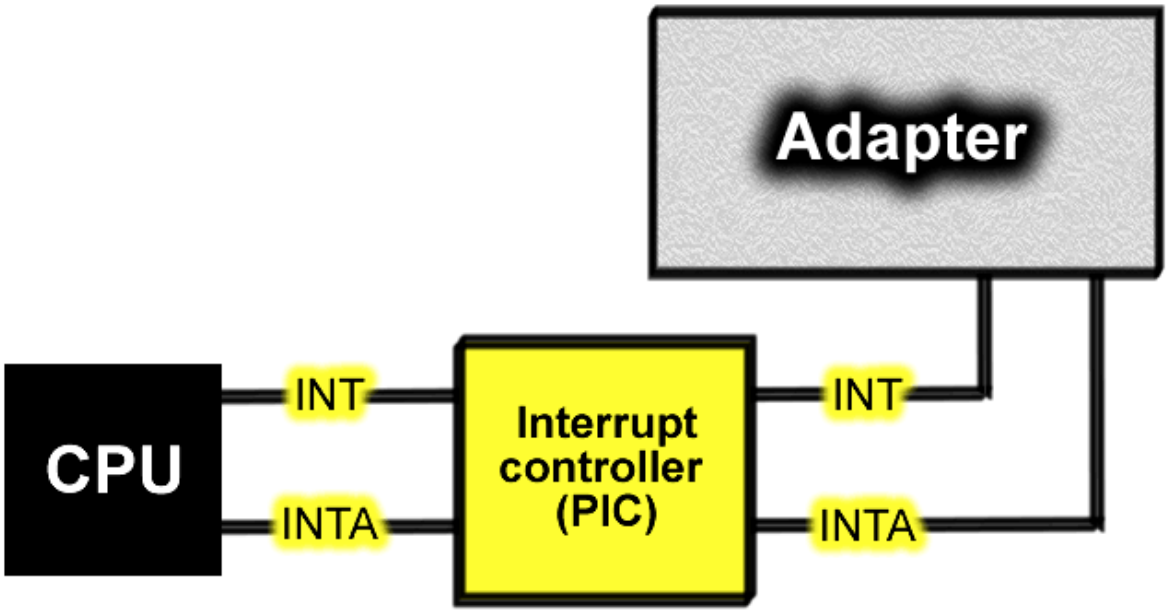


Figure 177. The interrupt controller is an important little controller. It communicates with the CPU using interrupt signals and regulates the traffic to and from the I/O devices.

In the original PC design, the PIC was a ”loose” controller chip (called Intel 8259). In modern motherboards the chipset’s south bridge contains an equivalent, 8259-compatible, PIC function.

A practical example

The advantage of IRQ’s is that the CPU can keep working on other tasks while an adapter “digests” the data it has to look after. When the adapter has finished its work, it notifies the CPU using a new IRQ signal – it rings the doorbell again.

Let’s look at the keyboard as an example. Every time you press a key, an IRQ is sent from the *keyboard controller* to the CPU. First, the keyboard sends its data as a series of individual bits along the keyboard cable. The bits are received on the motherboard by the keyboard controller. On modern PC’s this function is included in the Super I/O controller, but it used be on a separate 8042 chip. When the keyboard controller has collected one byte – that is one character – it sends an IRQ signal to the PIC (the IRQ controller).

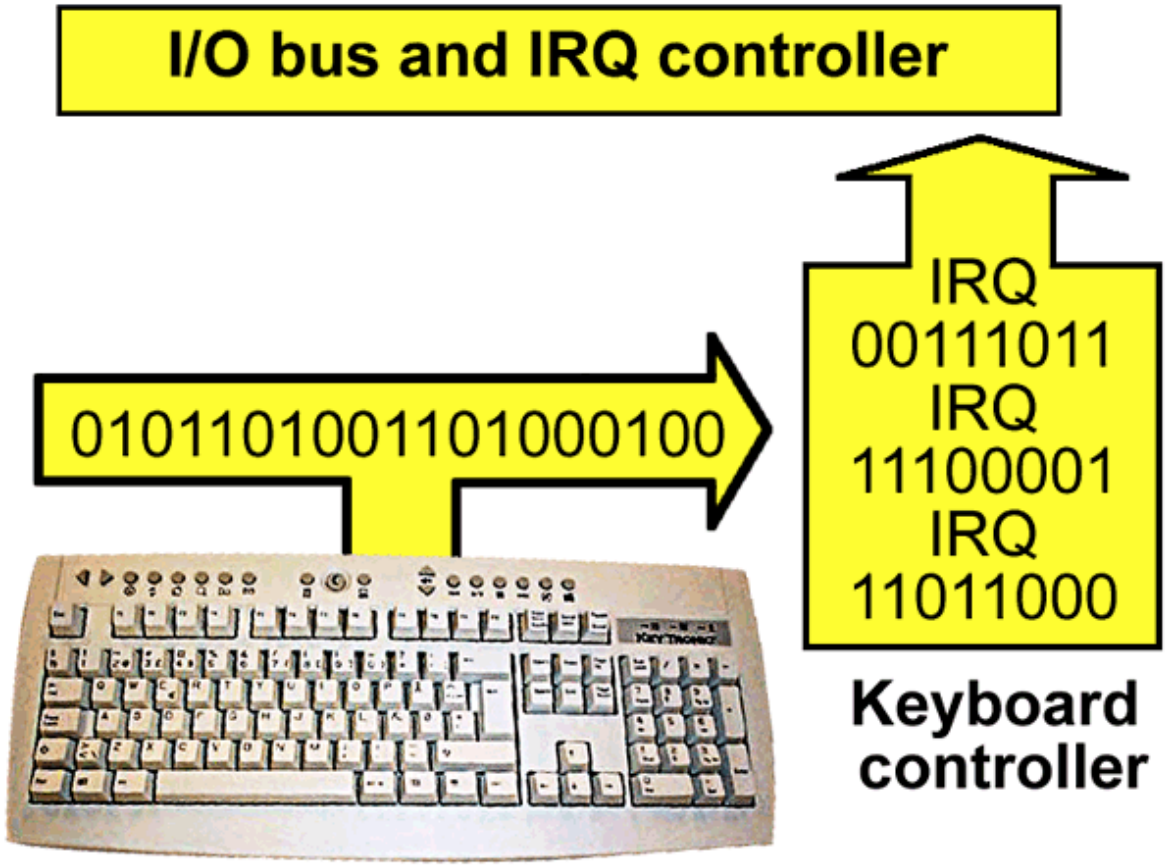


Figure 178. The keyboard controller delivers its data to the I/O bus when it gets permission from the PIC.

The signal requests permission from the CPU to send the character on. The PIC notifies the keyboard controller when the way is clear to the CPU, and then the byte is moved onto the I/O bus. At the physical level, an IRQ is a printed track on the bus. The printed track connects to all the PCI slots on the motherboard, so no matter which slot the adapter is installed in it has contact with the IRQ system.

8 or 15 IRQ lines

In the original PC architecture, an 8259 Programmable Interrupt Controller was used on the motherboard. It could handle 8 IRQ lines. This was later extended to two controllers with a total of 15 lines (one of the lines is used to connect the two PIC's). That means there are 15 IRQ's available to I/O devices. They are used for several tasks.

Certain system devices, such as the FPU (the number cruncher discussed earlier, and the system clock, each lay claim to one IRQ. Five of the IRQ's are reserved for internal system devices like these. In Fig. 179 these are, for example, IRQ numbers 0, 8 and 13.

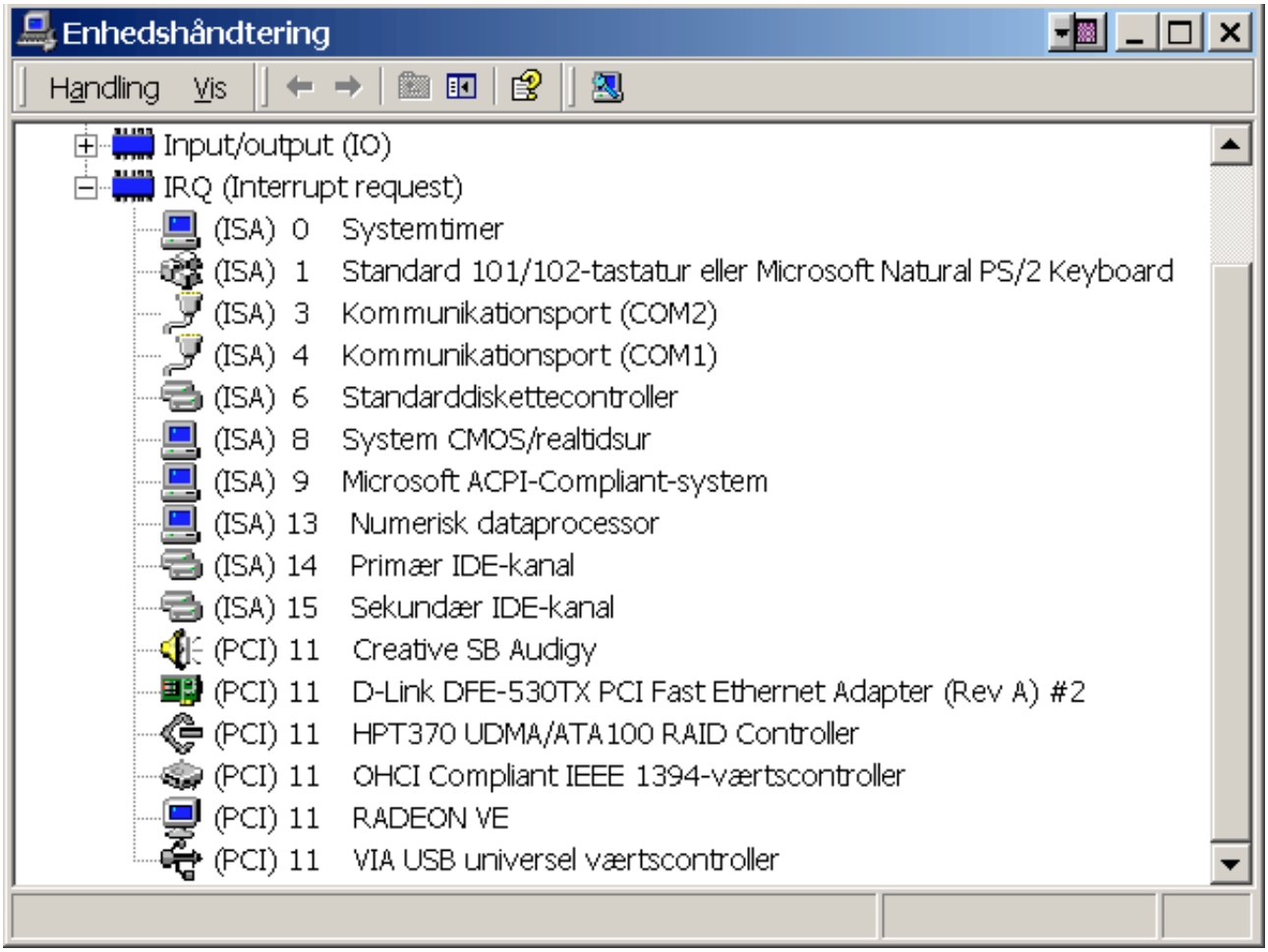


Figure 179. Windows reports on the distribution of IRQ's.

The remaining IRQ's are available for I/O devices. You can also see in Fig. 179 that a number of IRQ's are free. E. g. numbers 5, 7 and 10.

Each of the IRQ's is a physical printed track that runs throughout the whole bus and hence the whole motherboard. The IRQ's have different priorities, so that the CPU knows which IRQ should be given *preference*, if two signals are set at the same time.

You can use the Windows Device Manager (or System Information) to see the way IRQ's are distributed on your PC, as shown in Fig. 179. They should hopefully match.

Shared IRQ's – a typical PC solution

One of the big innovations in the PCI bus is that it allows *shared IRQ's*. Two adapters can actually share an IRQ. In my PC there are no less than six devices sharing IRQ 11, and it works faultlessly.

The invention of "shared IRQ's" is a good example of the evolutionary development of the PC architecture. The basic fault lies with the thin and undersized IRQ system; there are only 15 "doorbells" available in the system, and that is far too few.

It would be logical to extend the system to, for example, 256 doorbells. But that is not possible for compatibility reasons. The IRQ system is located at a very deep level in the PC architecture, and it would require complete reprogramming of both the CPU instructions and the operating systems if one changed it. And the PC would no longer be able to run, for example, old DOS programs.

In order to maintain compatibility, a "work-around" solution is typically chosen. A stopgap solution you could say – but one which works. A doorman is employed at each doorbell. He checks who is ringing the doorbell – which device is using the IRQ. And voila, using shared IRQ's, the system works again!

How it was in the old days …

Before the PCI bus came along we were stuck with the ISA bus, which was very limited by the small number of

IRQ's. It didn't take much before you were short of free IRQ's if you needed several adapters in your PC. Sound cards used to especially play up. Sound cards are quite complex pieces of equipment, incorporating several functions. The old ISA based Sound Blaster cards usually claimed two IRQ's and nothing less than two DMA channels as well. A card like that could be very difficult to install.

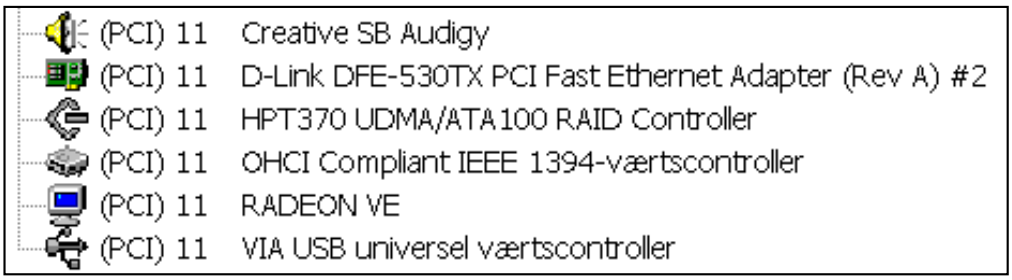
An ISA network card, might be configured by the manufacturer to work with IRQ 9, 10, 11 or 12, for example. One of the values (e.g. IRQ 11) was chosen as the standard factory setting (as the *default* value). When the customer installed the new network card, it would try, at startup, to go on the bus as IRQ 11. If there was no other device using IRQ 11 already, the card would probably work fine. But if IRQ 11 had been claimed by another card, there would be problems – the two devices had a conflict. Often the PC wouldn't even start, and the air was thick with panic.

The solution was, for example, to change an IRQ jumper on the physical network card, and then try again. We are fortunately completely spared from that kind of mess with the newer PC's. The PCI bus checks itself which adapters are using shared IRQ's, and the system functions quite automatically and problem free.

View the shared IRQ's

Look a bit more closely at Fig. 179. The first thing to note is that it relates to an older motherboard, which also has an ISA bus. That is apparent by the fact that (ISA) is written in front of a number of the IRQ's. Don't be too concerned about that, because the ISA bus is not actually used for anything – nor are the IDE controllers, although you could get that impression from IRQ 14 and 15.

The following devices have been listed as located on the PCI bus, and they all use IRQ 11.



These devices are:

Device	Description
Creative SB Audigy	Sound card
D-Link Ethernet	Network controller (network card)
HPT 370	A RAID controller, built into the motherboard.
IEEE1394	FireWire controller, built into the sound card.
Radeon VE	Video card (which is on the AGP bus).
VIA USB	USB host controller, built into the south bridge.

Figure 180. Six logical devices which share the same IRQ.

DMA and bus mastering

DMA stands for *Direct Memory Access*, and is a system on the motherboard which permits an adapter to transfer its data to RAM – without drawing on the PC's data processing power. It's actually the CPU which controls every event on the bus. With DMA, this intelligence has been put out in the DMA controller on the motherboard.

This special controller chip (Intel 8237, often integrated into the modern south bridge) is allowed to move data to and from RAM, via the I/O bus. And this happens without the CPU being burdened with the work.

The motherboard has a number of DMA channels, each assigned a number. They are used by the ISA devices, which can each lay claim to one of the DMA channels. The floppy drive, for example, uses DMA:

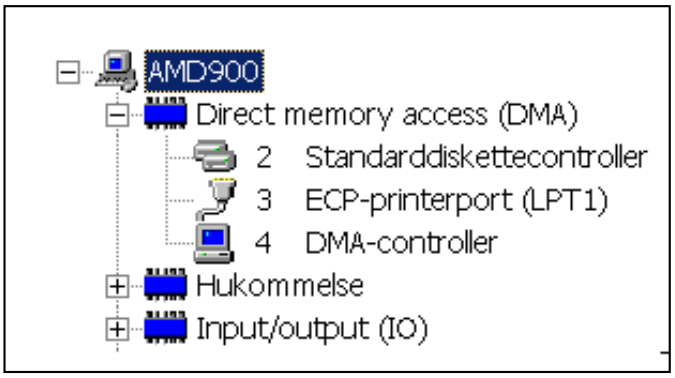


Figure 181. The Windows Device Manager reports on the usage of DMA channels.

There are no direct DMA channels on the PCI bus. Instead, *bus mastering* is used. It is a very similar system, where special controller functions permit adapters to take control of the bus. The adapters can then deliver their data directly to RAM, in a way which burdens the CPU as little as possible. It doesn't have to monitor the transaction – the bus master looks after that.

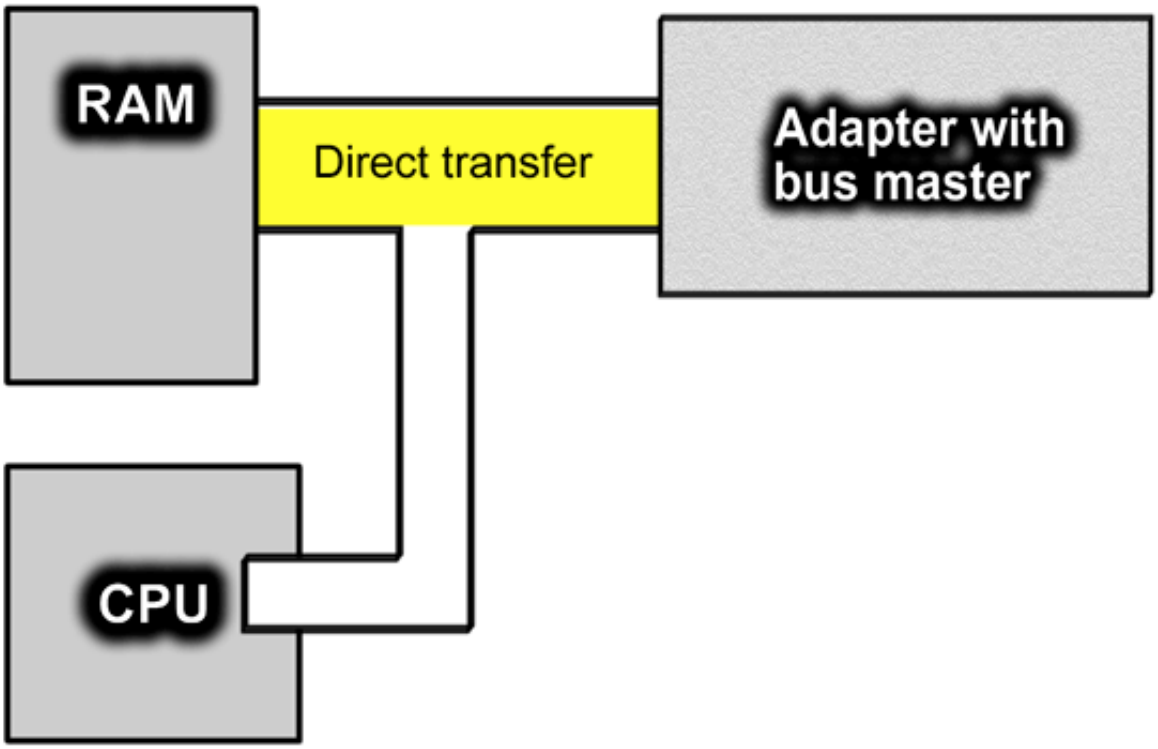


Figure 182. Bus mastering relieves load on the CPU.

The goal is that the PC can *multitask* – that is, handle several tasks at the same time. The hard disk can deliver data to RAM in a steady stream, while the CPU does other work at the same time. The bus mastering system works well with ATA devices, which work closely with RAM.

Though in this area, the SCSI controller is much more developed than the ATA controller. In practise, SCSI disks can operate almost independently of the CPU. SCSI interfaces are therefore still used to control hard disks in high-end servers and workstations.

Memory-mapped I/O

All devices, adapters, ports, etc. each have their own *address*. It is also called an I/O port number. The I/O address is necessary for the CPU (or RAM) to be able to fetch and send data to and from the device. The way this takes place inside the PC is called *memory mapping* of I/O devices.

I'm not exactly sure how much practical benefit is derived from understanding this system. But it is part of the whole story, so I'll give you a quick overview. When you want to send data to your network card, it takes place through one or more *I/O ports*. These ports are located in RAM – that's actually the whole idea.

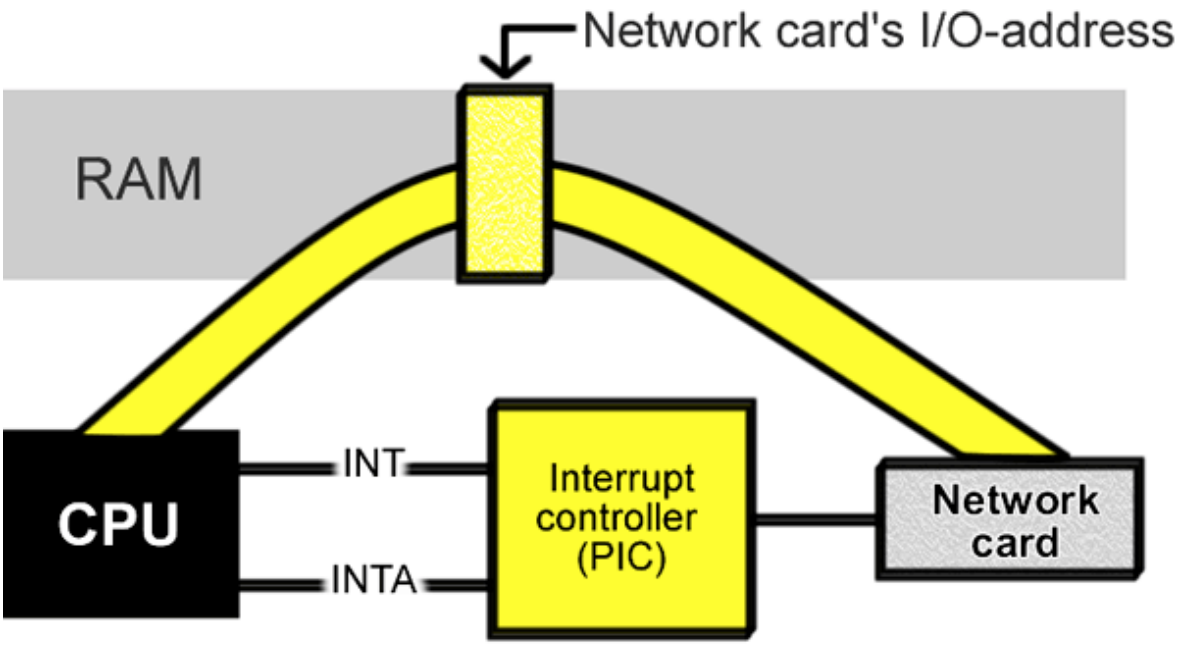


Figure 183. Memory-mapped I/O.

Thus when you want to send data to your network card, you send it to a particular area in the RAM. And the network card is ready to fetch its data from the same place, as soon as it gets the signal from the interrupt controller.

The PC can *address* (access) every byte in the RAM storage. And this addressing is exploited in the I/O ports. Every adapter is allocated a specific RAM address. That is, a small portion of RAM is linked to the I/O device. From then on, all data exchange to and from the device passes through this "mailbox".

All I/O devices have their own "mailbox number" – a port address. Since the PC is fundamentally a 16-bit computer, there are a total of 2 to the 16 (2^{16} = 65,536) possible port addresses. They are specified using hexadecimal numbers, from 0000h up to FFFFh. Here are some examples of I/O addresses, as seen using the Windows Device Manager.

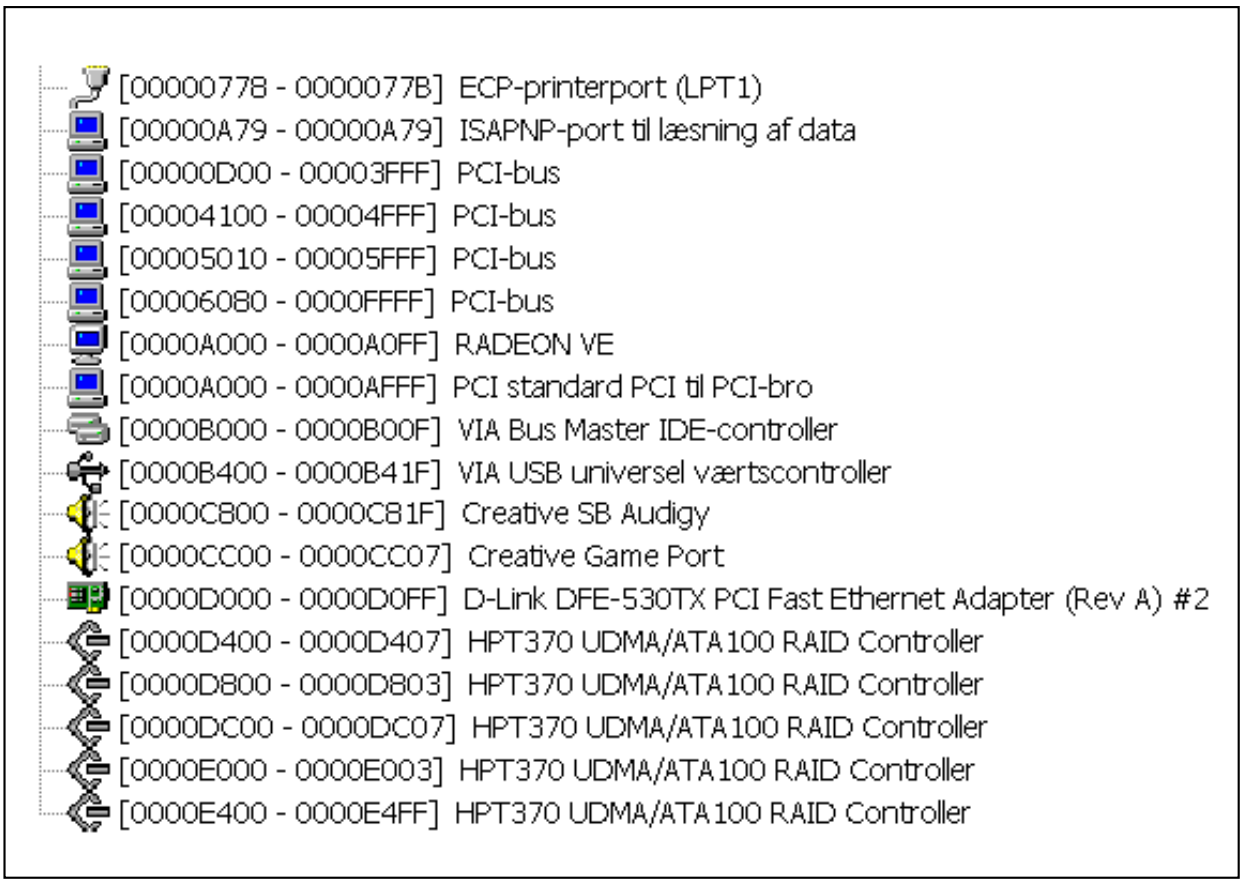


Figure 184. I/O addresses in the PC's RAM.

The table of I/O addresses takes up the "bottom" 64 KB of the PC's working storage (from 0000h to FFFFh). The I/O addresses fall inside that special "first megabyte of RAM" – which you might like to read about in the guide "Do it yourself DOS". The individual I/O addresses can have varying lengths of up to 64 bytes.

We can see here that the system of I/O addresses was designed right back in around 1980, and is still being used. It is fortunately very rare that one, as a user, has to set up port addresses. Certain network cards support user-defined I/O addresses, but you would have to be very unlucky to encounter a conflict in this area.

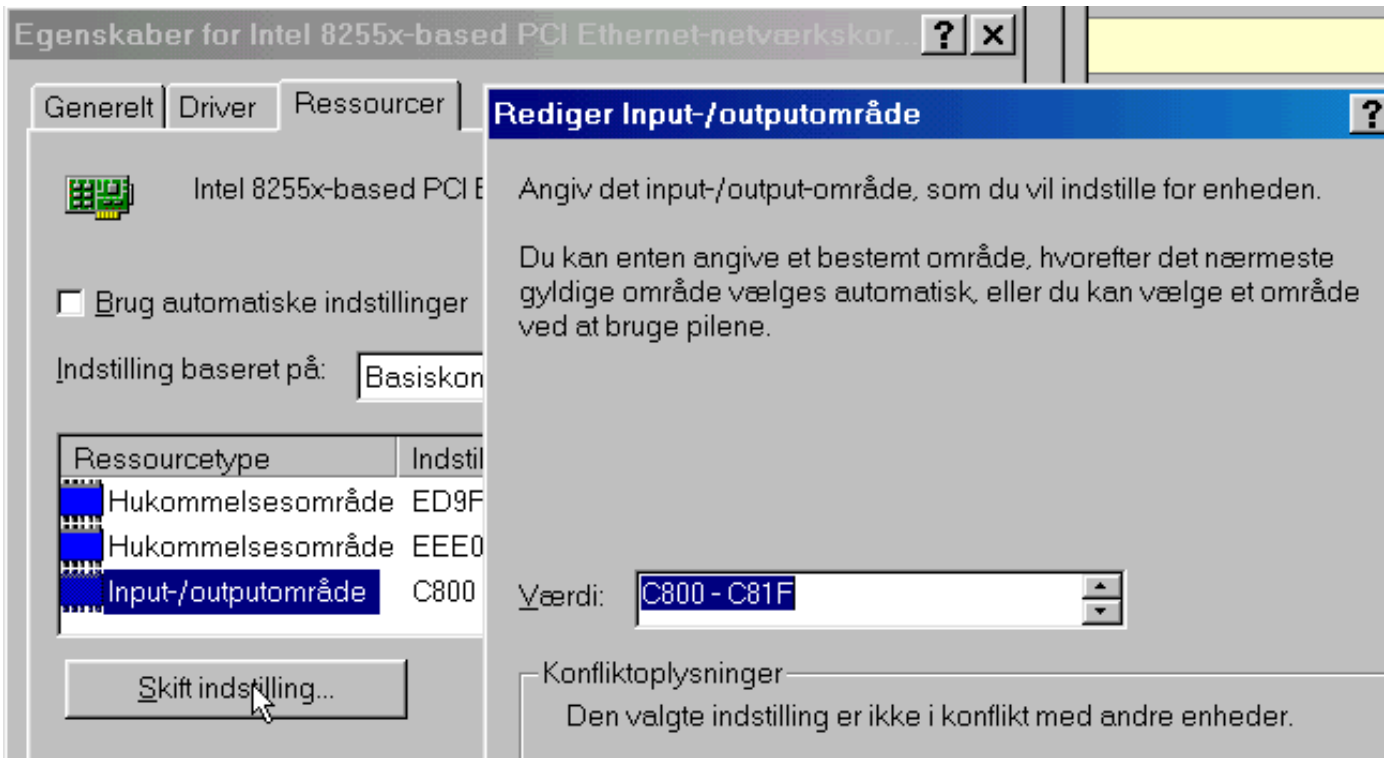


Figure 185. Windows displays the I/O address for a PCI adapter, and provides the facility to change it.

- Next chapter.
- Previous chapter.

Chapter 41. Check your adapters

I have now described adapters and their bus connections in a number of more theoretical sections.

Let's look at the subject a little more practically. I will use my own PC as an example, to show which adapters are installed and what I can learn from them.

Look in the cabinet

If you are going to follow my examination using your own PC, start by looking in the cabinet and seeing what adapters (plug-in cards) are actually installed. I have three cards installed:

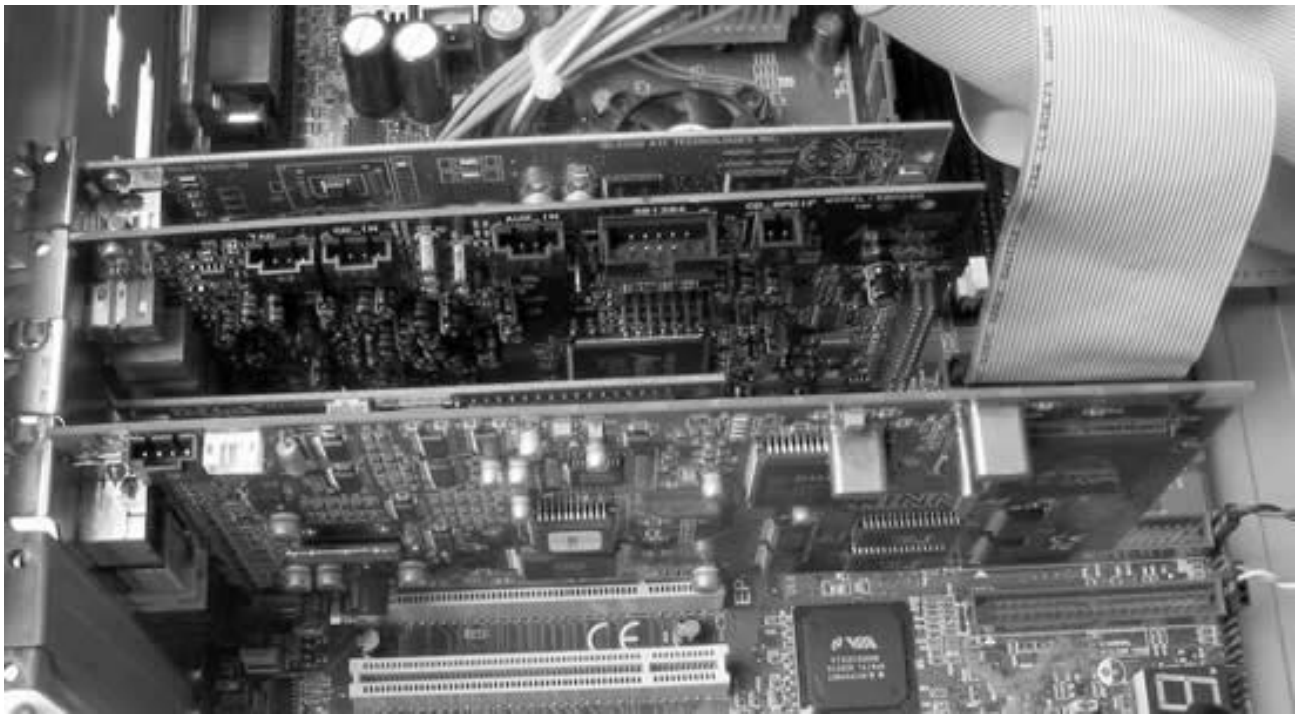


Figure 186. Three plug-in cards can be seen in this cabinet.

The three cards are:

- An ethernet adapter.
- Video card.
- Sound card.

I can see this from the back of the cabinet. There I can see the card's back plates and connectors, and they look like this:

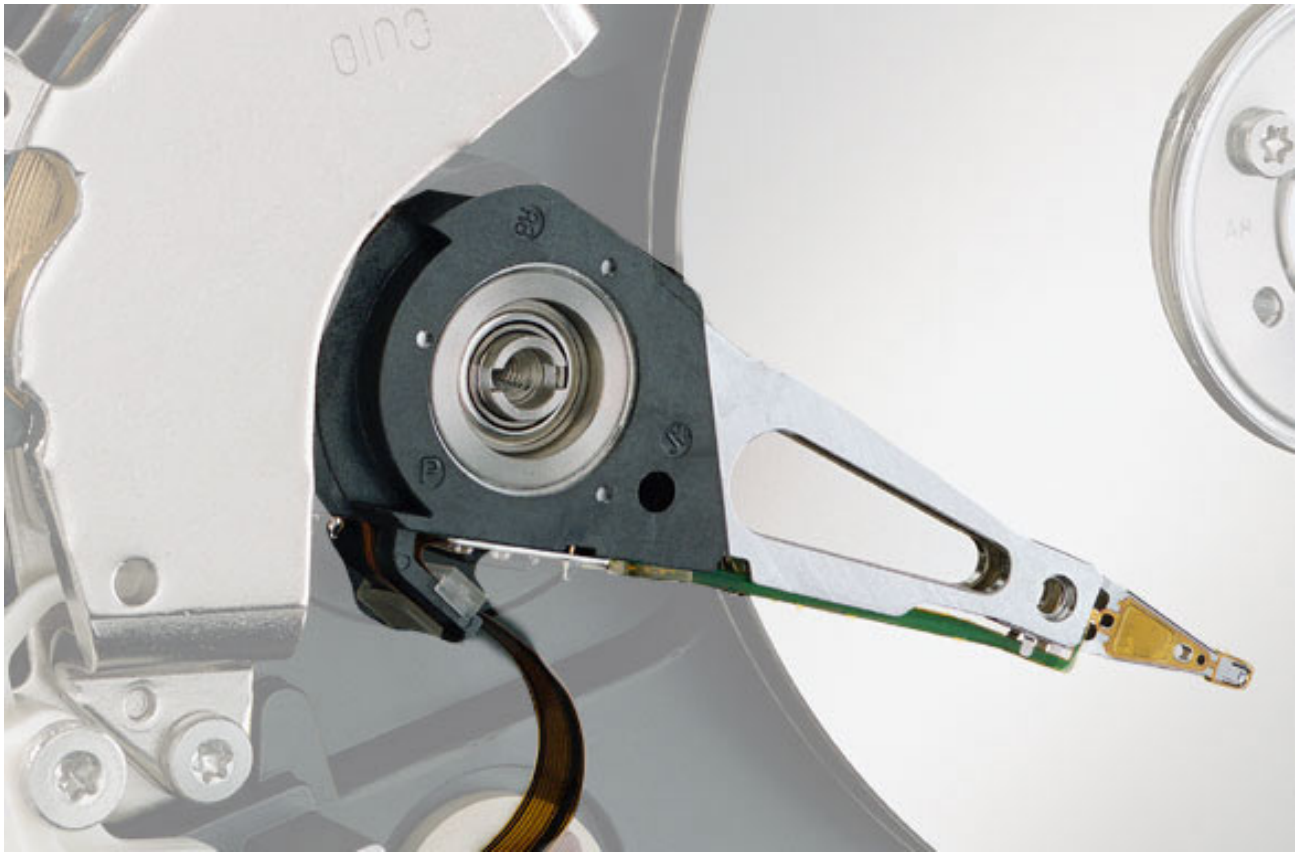


Figure 187. The back plates for the three adapters shown in the previous picture.

A closer look at a sound card

I will now take a closer look at one of my adapters. I have chosen the middle card, which is a sound card, a Sound Blaster Audigy. It is a bit unusual in that it has a built-in FireWire port. That is, the card has several functions; it operates as a sound card with many facilities. And it also functions as an IEEE-1394 (*FireWire)* host controller. This is what the card physically looks like:

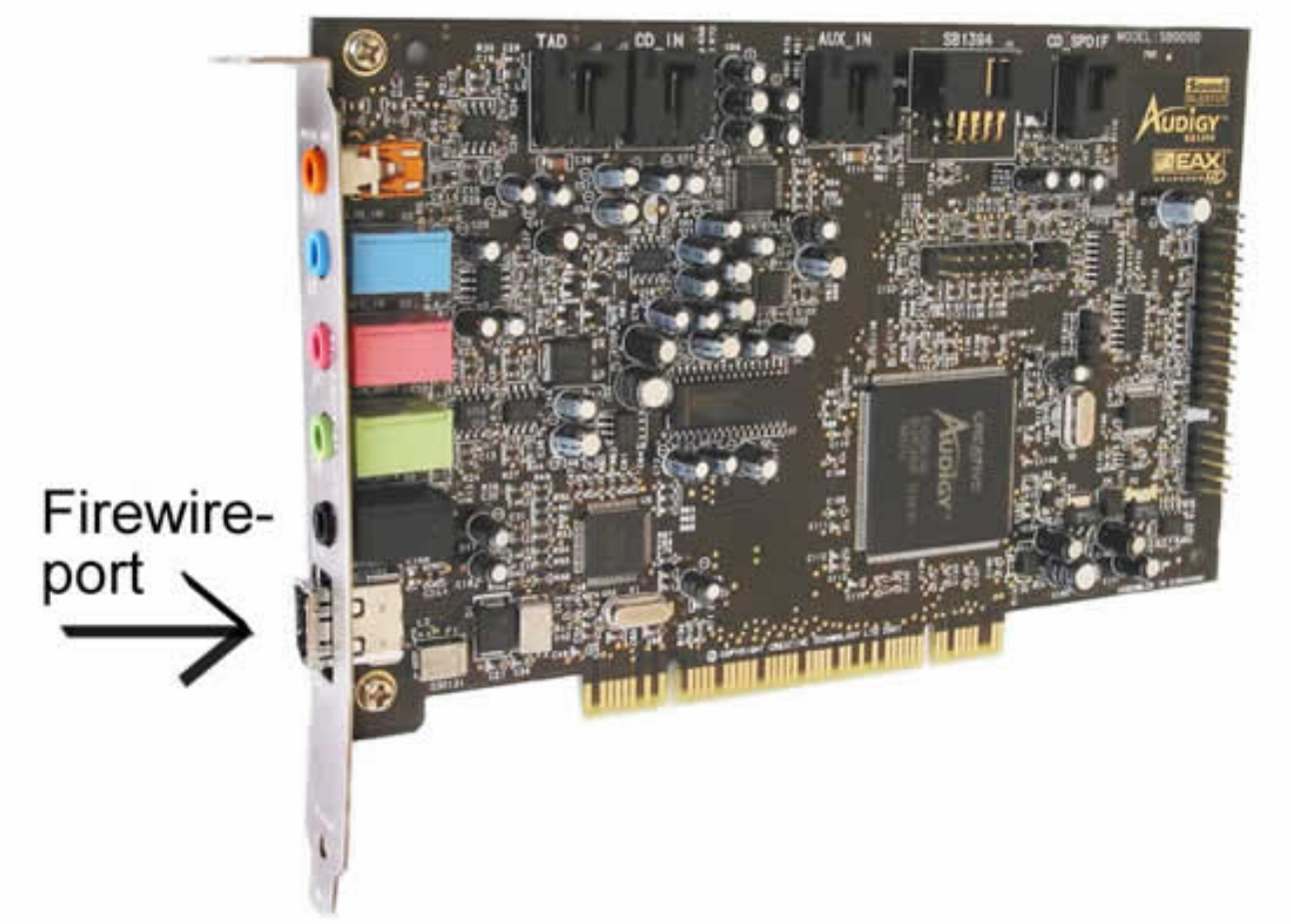


Figure 188. Here is the Audigy card out in the open. The FireWire port can be seen at the bottom of the back plate.

Since the card has several functions, it appears in several places in the list of system devices. I can find it under sound devices, where there are two devices (*Creative Gameport* and *Creative SB Audigy*) which are components of this adapter. And further down I can see the FireWire controller (IEEE 1394):

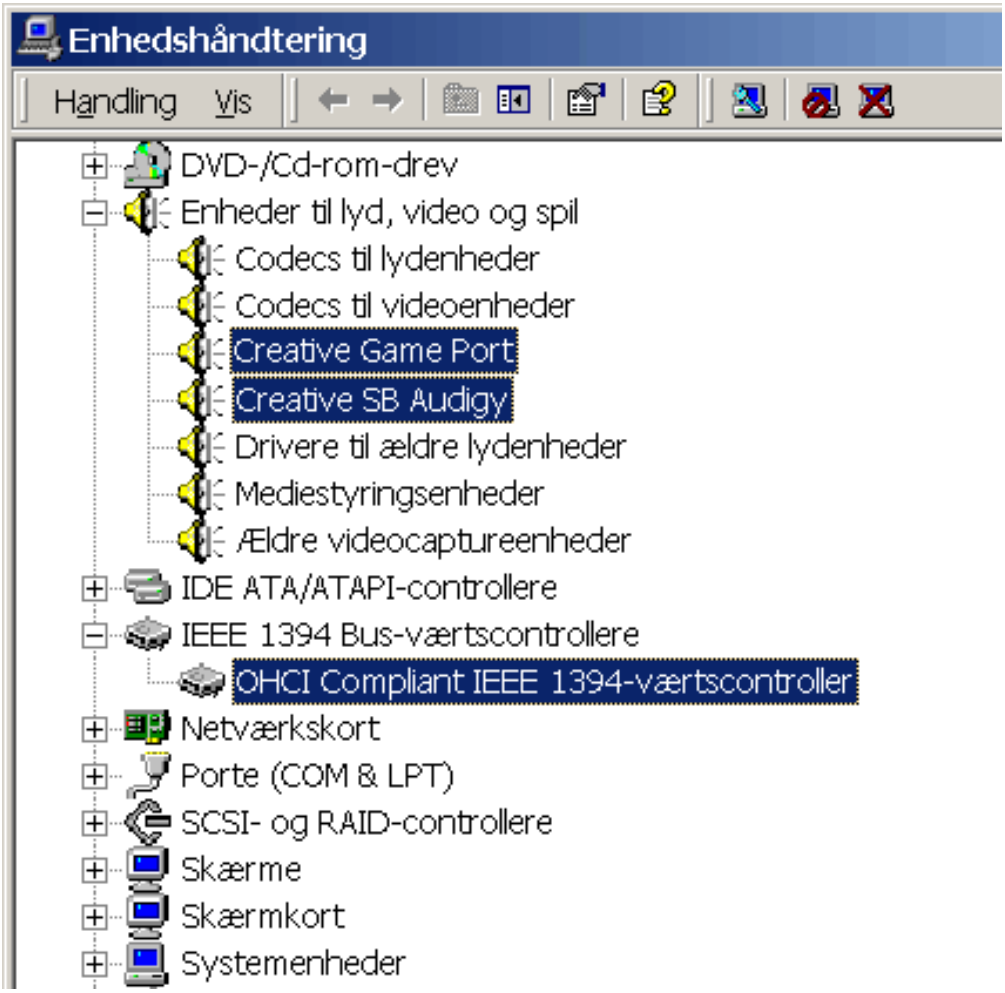
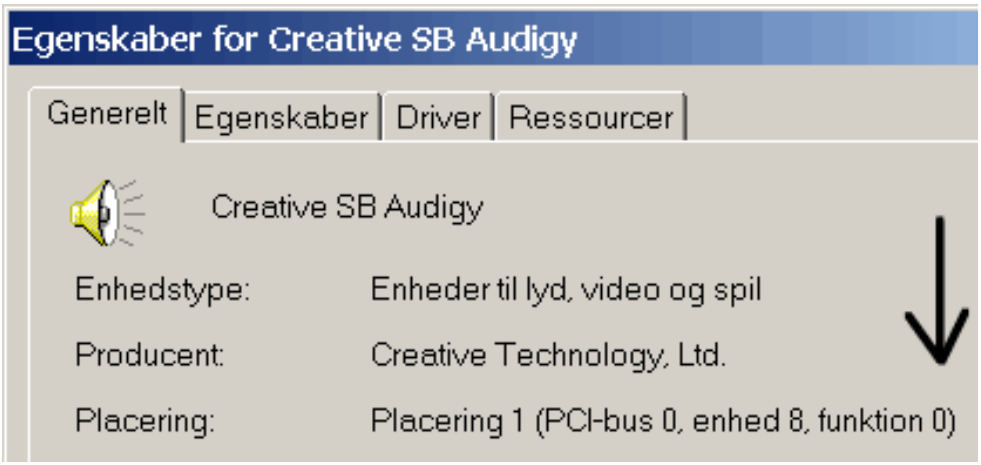


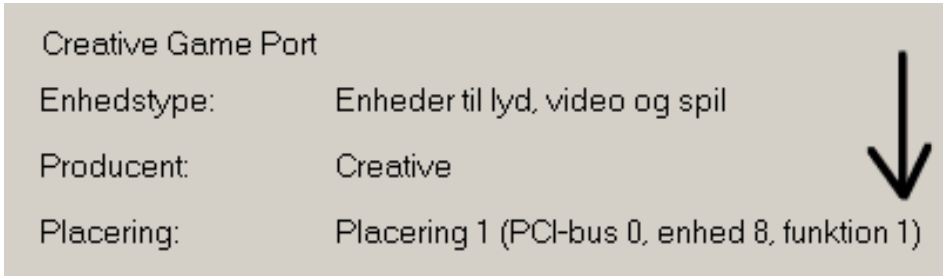
Figure 189. The SB Audigy card appears as three devices in the Device Manager overview.

If I double-click on the device ”Creative SB Audigy”, I see the following information displayed:



I can read that the actual sound functions are located on PCI bus 0 (there can be several PCI buses on the same motherboard). The adapter is installed as device 8. The sound component is function number 0.

If I double-click on the device ”Creative Gameport”, I can see that the game port is function number 1 on this card.



Finally, I will look at the FireWire controller. It has function number 2 on device 8 (which is the SB card):

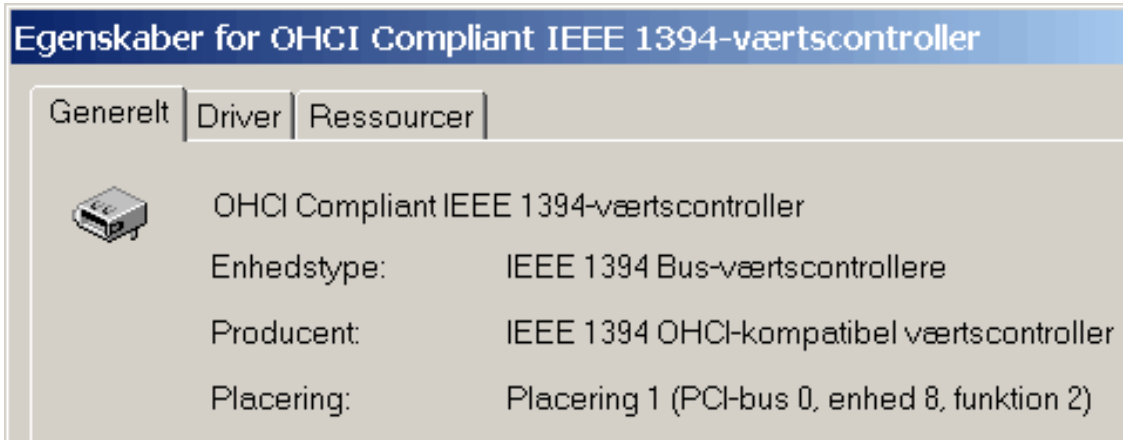


Figure 190. The FireWire controller is the last function on this sound card.

Check your own cards

In the section above I have shown how you can use the Windows Device Manager to investigate the PC's I/O system. You can do this in all versions of Windows. However, Windows 98 doesn't provide the *Location* information (as at the bottom of Fig. 190. Both Windows 2000 and XP provide this.

In all versions of Windows, you can open System Properties by using the shortcut key, Windows +Pause, and then clicking your way in to the Device manager.

Examine your PC's I/O devices yourself now as follows:

1. First find out what adapters you have in your PC. Note which sockets (*slots*) the cards are installed in, and check the connectors on the back plate so that you are sure of their function.

2. Note the information displayed on the screen when you start the PC (as in Fig. 175, on page 68). You can stop the screen scrolling using the P key to give you time to note the information.

3. Then open the Windows Device Manager and find the devices corresponding to your plug-in cards.

4. See what IRQ's they use, and note where they are located on the PCI bus, if you can find this.

5. Compare your observations on IRQ allocation with the messages you see on the screen during PC startup. Are these consistent? Is it the same adapters and IRQ's you can see in both places?

6. Then use the Device Manager to find an overview of the IRQ allocation. Compare this with Fig. 179. Which IRQ's are shared? Which internal system devices can you see in your list?

7. Which DMA channels are being used in your PC? Compare this with Fig. 181.

8. Finally, investigate your hard disk and USB controllers. What can you find out about them?

If you spend some time doing this exercise I guarantee you will gain a better grasp of this part of the PC architecture!

- Next chapter.
- Previous chapter.

- Next chapter.
- Previous chapter.

Chapter 42. I/O and The south bridge

We have now looked at the various I/O buses, and the chipset's south bridge plays the central role in the transfer of data on these buses. But, as mentioned before, the south bridge is assisted by a small Super I/O controller. Let me describe it briefly.

Functions of the south bridge

The south bridge is a chip on the motherboard. If we want to look at one of the latest models, we could take the south bridge from Intel's 915 chipset, which is designed for motherboards with Pentium 4 processors.

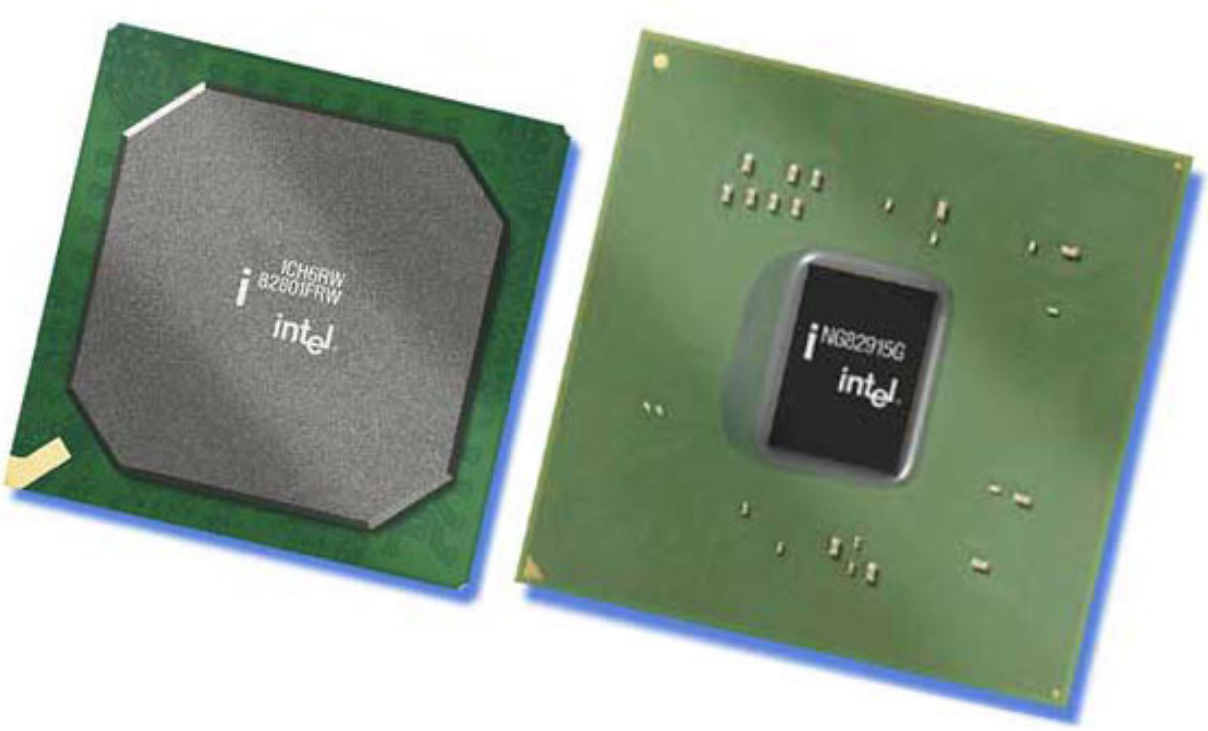


Figure 191. Intel chipset 915G. The south bridge ICH6 is to the left.

A peek at the chip's specifications reveals that it contains the following components and functions:

Component	Description
DMI	Direct Media Interface is the connection to the memory with a bandwidth of max 2 GB/sec.

PCI Express	Hi-speed bus for I/O adapters
PCI ports	Standard I/O bus.
Serial ATA	Controller for up to four SATA hard disks
Matrix Storage	Advanced Host Controller Interface for RAID 0 and 1 on the same drives. Including support for Native Command Queuing and hot plug drive swaps.
Ultra ATA/100	Controller for PATA devices like hard disks, DVD- and CD-drives.
USB ports	Hi-speed USB 2.0 ports.
7.1 channel audio	Option for integrated sound device with surround, Dolby Digital and DTS.
AC97 modem	Integrated modem.
Ethernet	Integrated 10/100 Mbs network controller.

Figure 192. Selected functions in a modern south bridge.

But a number of traditional controllers for the floppy disk drive etc. are missing. These can be placed in the Super I/O controller.

Super I/O

The so-called Super I/O controller is found on all motherboards. It is a helper chip, which looks after a number of the less demanding controller functions for the south bridge.

The tasks of the Super I/O controller are quite well-defined: it is therefore fine to use a mass-produced, standard chip, which can be used on virtually any motherboard. This helps keep prices down, and these Super I/O controllers come from companies like Winbond and ITE.

Figure 193. A super I/O controller from Winbond.

The super I/O controller is connected to the south bridge using a special low-speed bus, and this architecture is called LPC (*Low Pin Count*). The chip is a 128-pin chip containing these components:

Controller	Description
FDC	82077-compatible floppy disk controller
UART	Serial port, 16550-compatible controller, including IrDA (infrared)
LPT	Parallel printer port
Game	For a joystick
KBD	8042-compatible keyboard controller
PS/2	Mouse port
LED	Controls the small LED's in the cabinet
Fan	Controls the fan

Figure 194. Functions which the Super I/O chip typically takes care of.

You will find many different layouts of the motherboard. The south bridge may be connected to other controller chips. This could be:

- A RAID controller (e.g. from Promise or HighPoint, see Fig. 228, or a SCSI controller (from Adaptec).
- A sound processor (e.g. from Creative or Yamaha).
- An Ethernet controller.
- A FireWire controller, and much more.

It is up to the individual motherboard manufacturers to decide how they will use the chipset. The south bridge offers loads of facilities which can be implemented if desired.

For example, some south bridges have built-in primitive sound devices (based on the AC-97 standard). This uses the CPU to process sound. On some motherboards this functionality will be put to use, while other manufacturers will choose to integrate a "real" sound chip from, for example, Creative.

In any case, some users will choose to install a separate sound card with even more functionality – regardless of what sound features the motherboard offers.

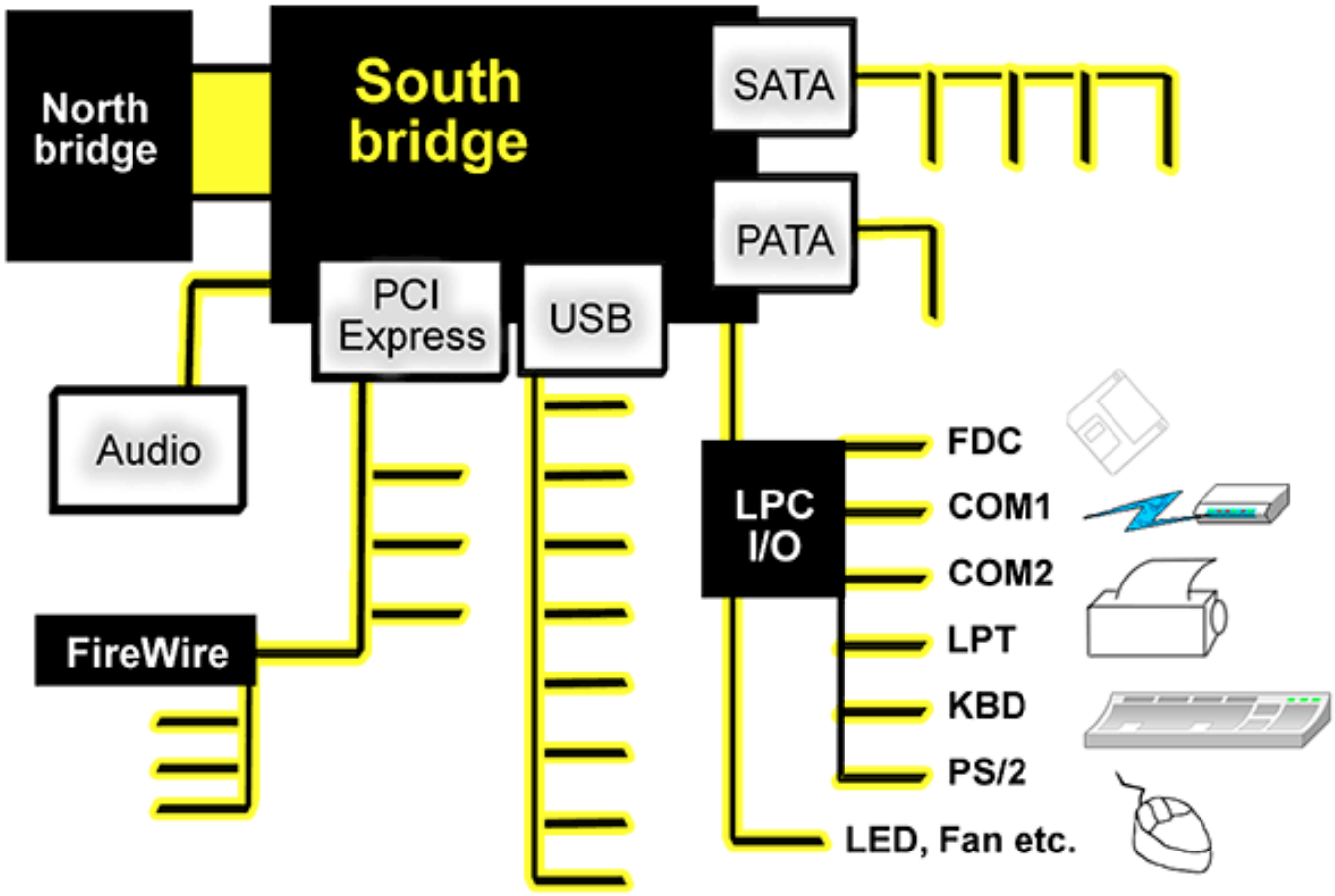


Figure 195. A total picture of the many I/O facilities offered by the south bridge.

The serial and parallel ports

The serial and parallel ports are included in the I/O system, and their controller is located in the Super I/O controller.

The PC is equipped with at least one (usually two) *asynchronous* serial ports. These are the male connectors on the back of the PC, which have either 9 or 25 pins, of which often only 3-4 are used for serial transmission.

The parallel port is also called the printer port, because that is what it is normally used for. The printer connector on the back of the PC is a 25-pin female connector.

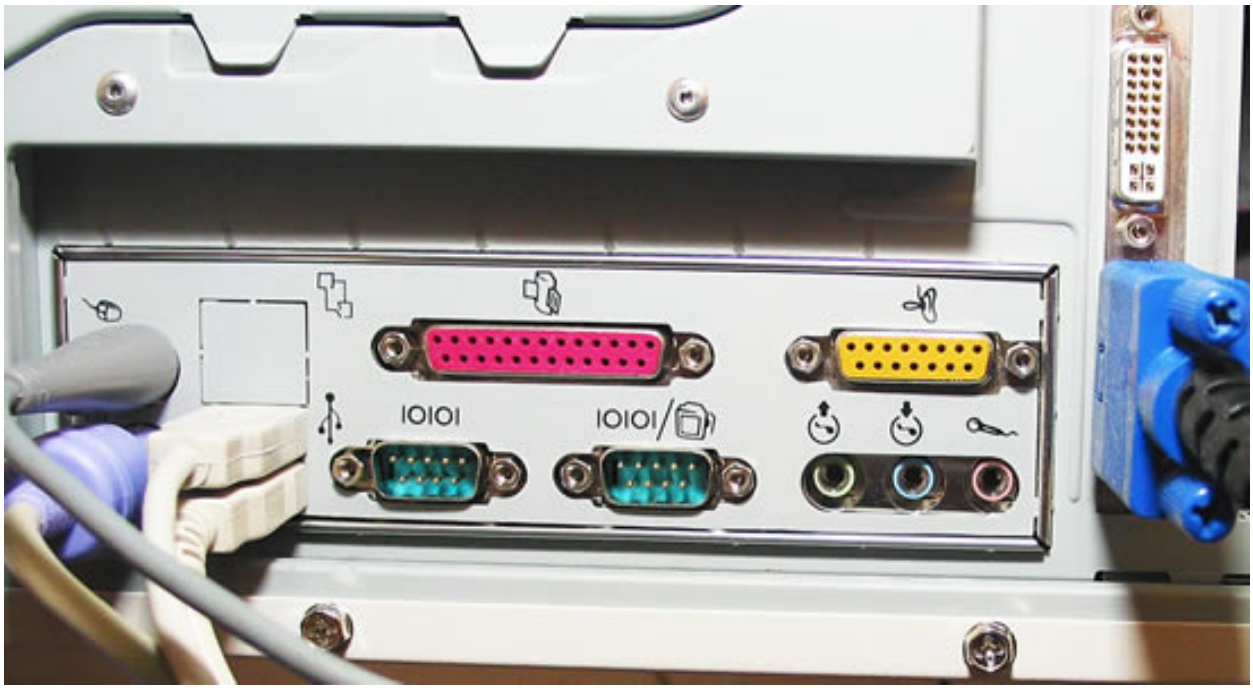


Figure 196. In the middle you see the red parallel port and the two green serial ports.

The ports are also called **COM1, COM2** and **LPT**. These are so-called *logical device names*, which the PC startup program automatically allocates to these devices during startup, just as **A:**, **C:**, **CON**, **PRN** and **KBD** are logical names.

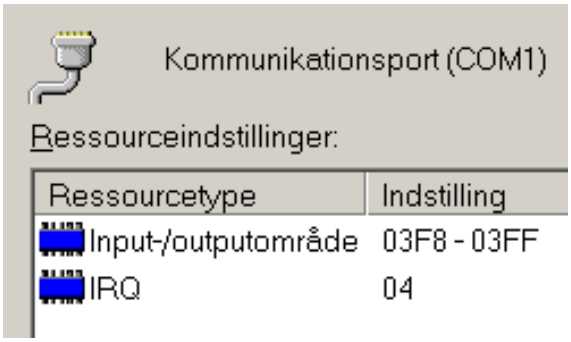


Figure 197. The logical name, COM1, refers to a serial I/O port.

Today the serial ports and serial devices are used much less than they used to be. This is especially due to the USB bus, which can connect equivalent devices. But if you use, for example, a USB based modem, it will behave just as if it was a normal serial device. Below an ISDN adapter is installed, which is connected via USB. It creates two virtual COM ports which can be seen using the Device Manager. In this way the Internet dialup program can use the USB device in exactly the same way as if it was connected to a (physical) COM port:

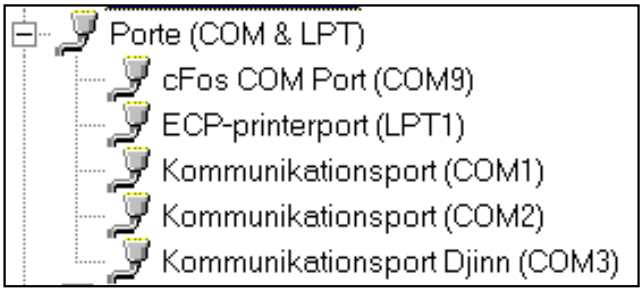


Figure 198. The COM3 and COM9 ports are actually a USB device.

Asynchronous transmission

The serial ports work using *asynchronous* transmission, which it might be interesting to know a little bit about.

Asynchronous transmission means that data can be transferred at irregular intervals, as you might already be familiar with from using a modem. The transmission only starts when the receiver is ready to receive. And the receiver can use unlimited time to ”digest” the data. The opposite principle is *synchronous* transmission, where a shared clock controls the pace.

The irregularity means that you have to be able to work out when to send data, and when a portion of data has been sent completely. This is achieved using a *protocol*, which defines the packet sizes, etc., which the sender and receiver have to agree on.

When the sender has nothing to say, a stream of 1’s is sent. At the start of a transmission, a 0 is sent. After this start bit, the data is sent in portions of 5, 6, 7 or 8 bits. Then follows 0, 1 or 2 so-called parity bits, which are used to perform a simple check on the transfer, and the packet is closed with 1 or 2 stop bits. The most common standard is 8 data bits, no parity bit and 1 stop bit. This system is often called 8/N/1.

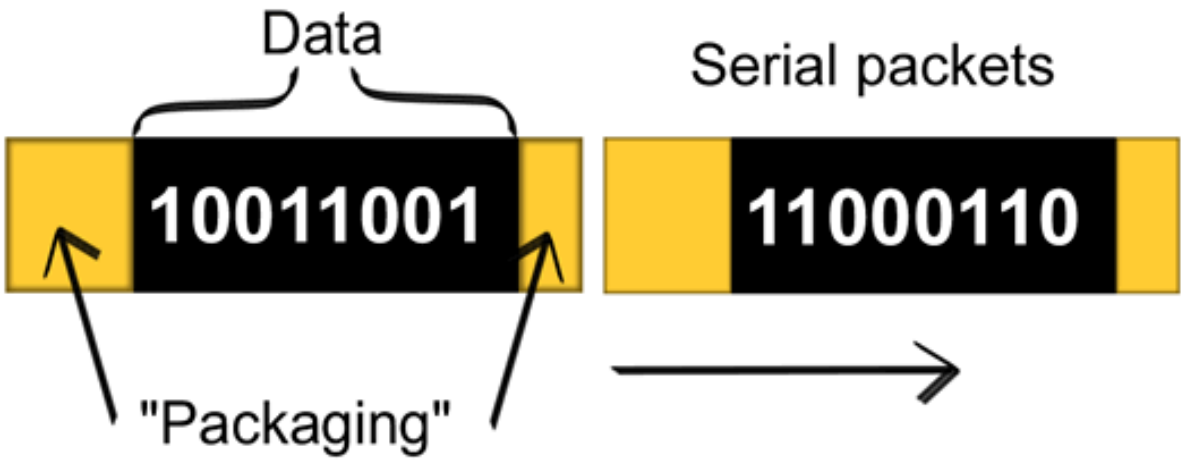


Figure 199. In a serial transfer, the data is packed into small, individual portions, which are sent one after the other.

The necessary ”packaging” of data into portions for serial transmission takes place in the UART chip, which is part of the Super I/O controller.

UART stands for *Universal Asynchronous Receiver/Trans-mitter*. The UART chip converts the data stream between the parallel I/O bus and the serial connection. Bytes from the I/O bus have to be “chopped up” into series of bits and “framed” with stop bits, etc., and this takes place in the UART.

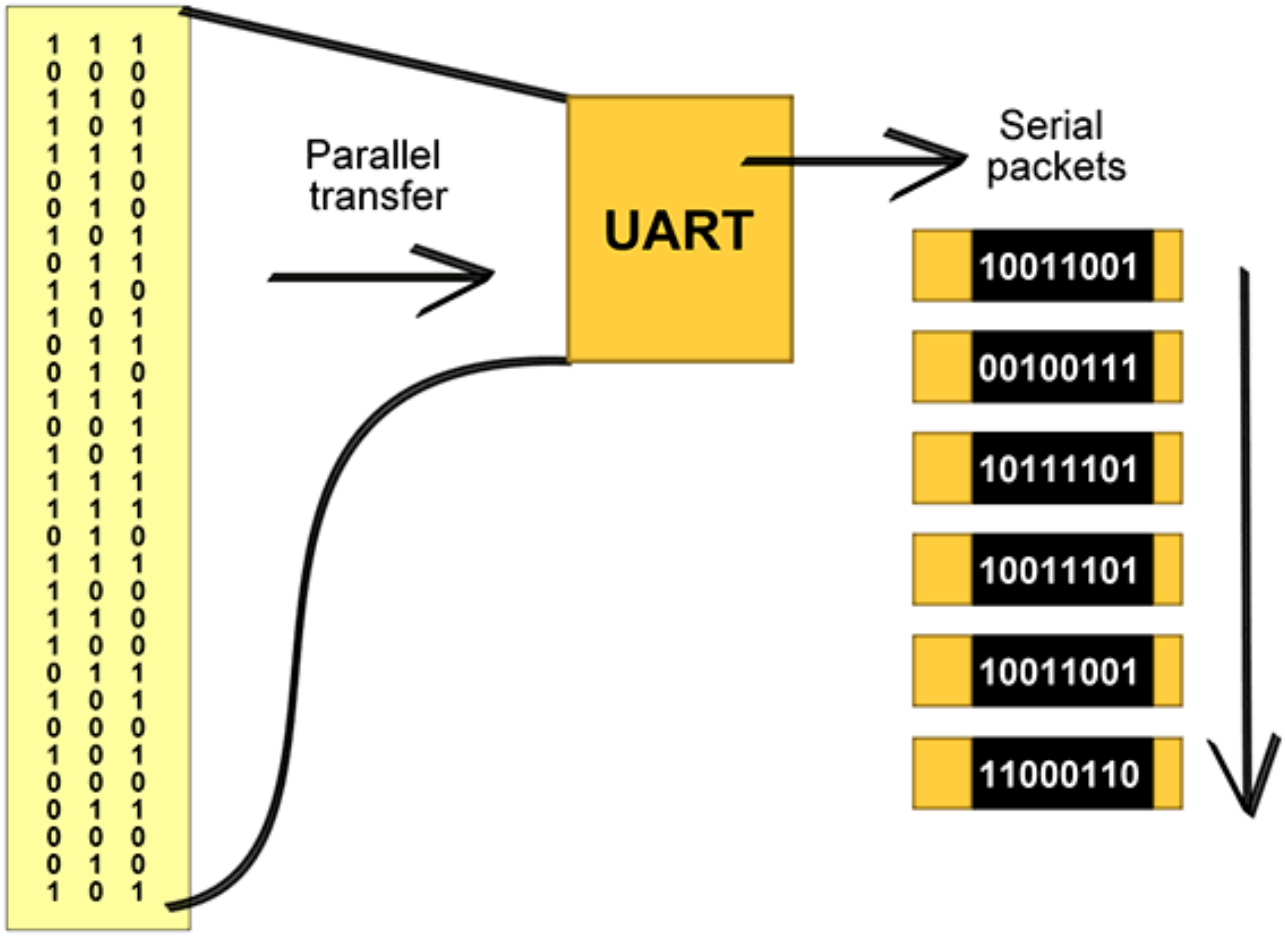


Figure 200. The UART controller repackages data in order to be able to perform a serial transfer.

- Next chapter.
- Previous chapter.

- Next chapter.
- Previous chapter.

Chapter 43. SCSI, USB and Firewire

In this chapter I need to discuss three I/O buses. They are very different, but they still belong together.

- SCSI is an older but advanced I/O bus which has especially been used for hard disks, CD-ROM drives, scanners and tape units.

- USB is a modern bus which can be used for a host of devices, and which has had a powerful breakthrough on the PC front in recent years.

- FireWire is a modern, high-speed I/O bus which is especially used for digital video cameras (DV), scanners and external hard disks.

As I mentioned, the three buses are very different, but they also overlap with each other. For example, you can buy CD burners with all three types of interface.

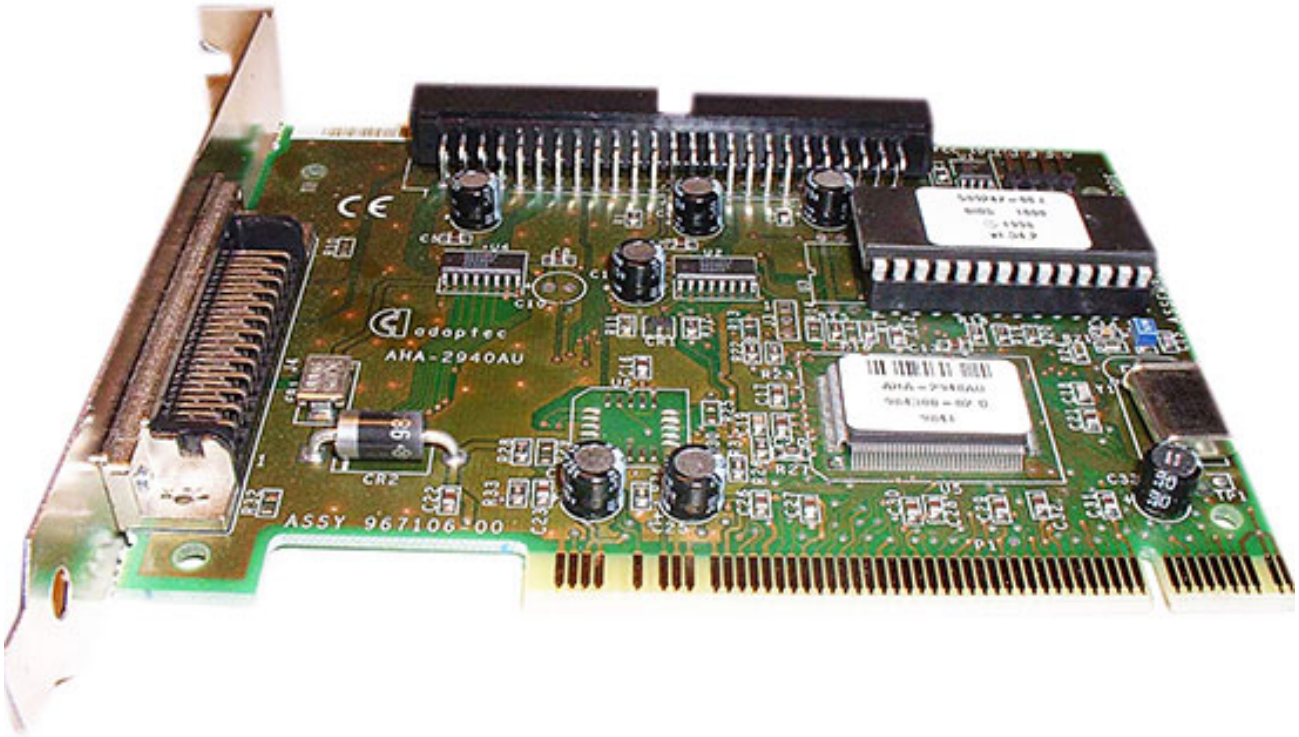


Figure 201. Adaptec 2940U SCSI controller.

SCSI

SCSI (*Small Computer System Interface*) is an advanced controller technology, which is especially used in high-end PC's. These can be network servers or just powerful workstations, for which there are a

number of different SCSI standards. The SCSI bus can transfer up to 160 MB/second, which is more than the PCI bus can deliver.

A SCSI system is built around a central controller, called the host adapter, which is almost a tiny computer in its own right. The adapter can be quite expensive if, for example, it has to be used with very fast hard disks. However there are simpler SCSI adapters, for example, those sold with SCSI based scanners.

Figure 202.
A small SCSI controller. Note the wide cable.

The best known manufacturer of SCSI adapters is Adaptec. It used to also be quite common for motherboards to have built-in SCSI controller – often of high quality.

A host adapter can control a number of SCSI devices, which are connected in a long series (a chain). Every device is allocated an identification number, and a *terminator* has to be put in at both ends of the SCSI chain. This is done, for example, using a jumper on one of the devices.

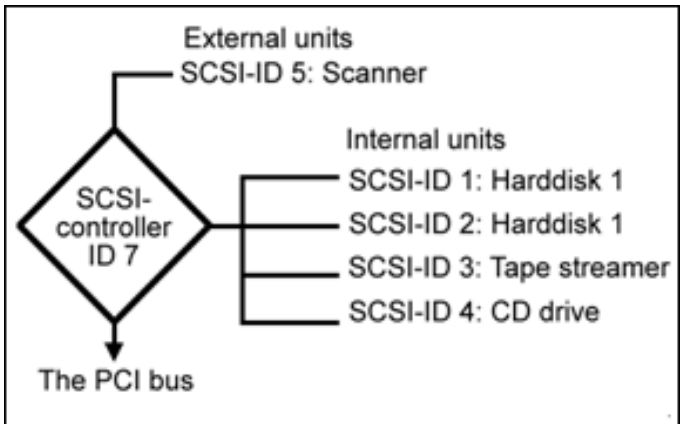


Figure 203. A typical SCSI chain.

The SCSI controller has its own startup routine in which it identifies the devices which are connected. The startup procedure can be followed in the screen:

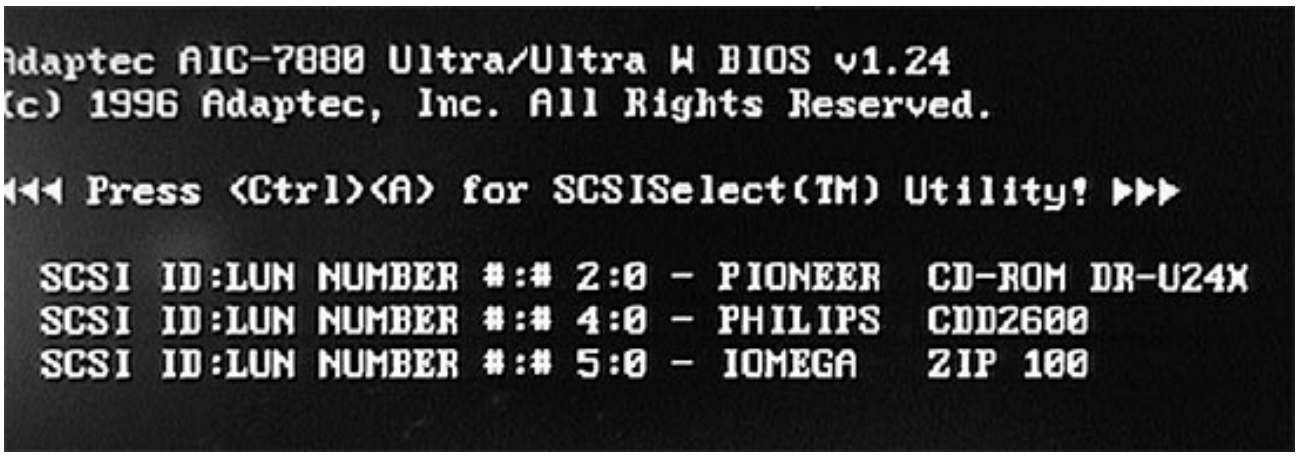


Figure 204. This SCSI chain consists of a CD-ROM drive, a CD burner and a Zip drive. This can be seen during PC startup, when the SCSI controller identifies each device.

SCSI is intelligent

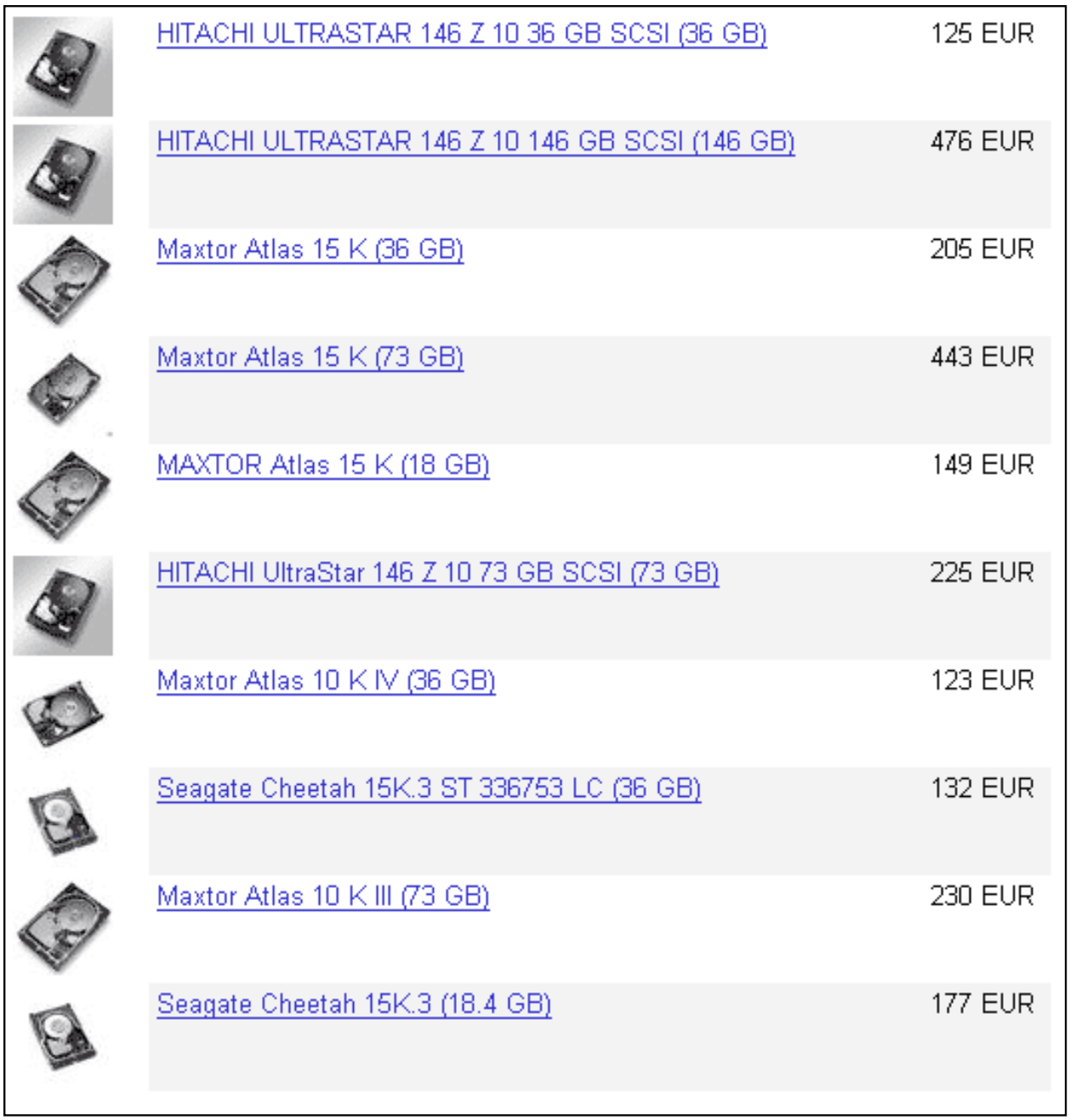
The big advantage of SCSI is the *intelligent protocol* which manages the devices and ensures optimal data transfer. Each device requests, for example, data from the host adapter, which then does all the work of acquiring this data. While each SCSI device is working, the controller has time to do other tasks.

The SCSI controller actually manages an entire small network of I/O devices, and thus relieves the CPU of the burden of all the associated work. SCSI therefore works well in multitasking environments such as web servers. These can have scores of SCSI disks organised in RAID chains.

Today, SCSI is used primarily for tape units (streaming tape) or hard disks in servers in large network servers (e.g. used by Internet service providers).

SCSI disks have a greater transfer capacity than standard ATA drives, and are more robust and generally better quality. SCSI used to be the only option if you wanted a big hard disk. But in recent years, ATA drive specifications have greatly improved, and today they are comparable with SCSI disks.

One consequence of the technological advances has been that the SCSI standard has lost a great deal of its significance. For scanners, the USB and FireWire I/O buses are used instead. These are much easier to configure, and have the advantage of small connectors and thin cables.



Figur 205. SCSI hard disks anno 2004.

RAID

RAID stands for *Redundant Array of Inexpensive Disks*. It is a disk technology that connects together a serious of standard hard disks to form an advanced, error correcting system, which is used in servers.

The system is virtually an extension of the SCSI standard, and was first used in 1987. Since then, ATA-based RAID systems have been developed which use the much cheaper ATA or SATA disks in an equivalent configuration (see the discussion later in the guide). The trick is, that you can spread your data over several disks. With a RAID chain of hard disks, you gain two advantages:

- Greater security. The data is on several disks. If one disk goes down, the other disks contain the same data.

- Faster data transfer. The RAID controller writes and reads from several disks at the same time. This means the transfer speed can be doubled or tripled using RAID. When the user has read or written his file, the controller finishes the job itself, so that the complete file is located on all the attached disks.

There are several RAID categories. A RAID controller has to be used, which is a special SCSI adapter. Here is a brief description of some of the standards:

Level	Technique
RAID 0	Two or more disks are connected together, and the files are split between them (*striping*). The goal is purely greater speed, as the controller can read from/write to several disks at the same time. There is no extra security.
RAID 1	Two hard disks are used. Purely provides increased security, as the data is written twice – first normally, and then to the mirror disk. If the first disk goes down, the second is immediately ready to replace it.
RAID 0/1	Uses four disks and combines the two techniques above.
RAID 3	Spreads the data over several disks and stores parity data on one of them.
RAID 5	Improves both performance and security. Uses at least three, usually four disks. Is considered to be the best principle.

Figure 206. There are many different RAID systems.

USB

USB stands for *Universal Serial Bus* and is an I/O standard originally developed by seven companies – Compaq, Digital, IBM, Intel, Microsoft, NEC and Northern Telecom (see the *USB Imple-menters Forum* at www.usb.org).

USB is a cheap serial I/O bus with an open specification. This means that anyone can produce USB products, without having to pay licences to anyone.

USB has been the biggest and most welcome innovation in PC design seen for many years. It is an expansion bus which allows a vast amount of PC equipment to be connected. It's suddenly possible to connect loads of different gadgets to the PC – and using just one type of connector! And USB devices can also be used both with Macintosh computers and PC's – yet another advantage.

USB has been advertised since 1994, and for many years it was called the *Useless Serial Bus.* But starting in 1999, production finally surged forward, and there are now thousands of different USB gadgets.

USB unifies all the different connections for keyboard, mouse, scanner, joystick, digital camera, and perhaps printer, onto a shared bus – connected using a common connector type.

Figure 207. A USB-based trackball – actually designed for Macintosh computers, but works fine on PC's.

USB – the technology

From a technical viewpoint, the following can be said about USB:

- The transfer speed is limited to a maximum of 12 Mbit/sec. in USB version 1.1. It is therefore primarily used for equipment which doesn't require a large bandwidth.

- USB version 2.0 has a bandwidth of 40 MB pr. second, and is used in all modern computers. USB version 2.0 is backwards compatible. The same type of connector is used, and old devices can be connected to the new controllers.

- USB is a serial connection using just four conductors (in contrast to the 50 or so used for a PCI device). This makes manufacturing much easier and cheaper.

- The USB cable can also supply power to the devices. This means that scanners, for example, don't have to have their own power supply. The maximum cable length is 5 meters.

- Up to 127 USB devices can be connected to the PC using USB *hubs*.

- There are no IRQ's to be configured or terminators etc. USB devices can be connected "On the fly", without restarting the PC.

There has to be a USB host controller in the PC in order to be able to connect the devices. This controller can be bought separately, as an adapter, but most motherboards have one built into the chipset's south bridge. There are typically two or four USB connectors on the motherboard, but you can have many more USB devices than this if you connect an extra hub (e.g. integrated in a screen, as in Fig. 209.

Figure 208. An in-expensive USB 2.0-based hard disk box. You can use any old hard disk in it. Great for backup!

A flexible bus

I am personally very grateful for USB. I find use for the following devices:

- Card reader for CF and other RAM cards
- Trackballs and Wacom drawing pen (tablet).
- Flatbed scanners and ISDN-modems.
- Newer printers. A color laser printer is faster using the USB 2.0 interface than the old LPT-interface.
- External CD- and DVD-drives.
- External harddisks (like the one in Fig. 208.
- USB-based speaker or microphone systems.
- Interface to digital cameras and MiniDisc.

USB has thus made the serial ports (COM1 and COM2) redundant, and we can be very happy about that. The technology has also opened the door for the development of a number of new cheap and fun products.

Figure 209. This screen has a built-in USB hub in its base. That means that a further four USB devices can be connected to the screen, which is itself connected to the PC's USB connector.

IEEE 1394 – FireWire

Another, more SCSI type, interface is called IEE 1394. It doesn't look a lot like SCSI, as we know it, but the standard is an extension of the SCSI standard. IEEE 1394 (FireWire in everyday language) is a high-speed serial bus which has a maximum transfer rate of 400 Mbit per second.

Figure 210. A FireWire cable

Some computers are born with FireWire ports being integrated on the motherboard. On other PC's, you can install a cheap little adapter; then you can have direct access to your digital DV video camera. IEEE 1394 connections are "hot", i.e. you can connect and disconnect devices without having to restart your PC, just as for USB.

A summary

I have described three expansion buses in this chapter which, from a technological perspective, are very similar. For many years, SCSI has been a "luxury bus" which made it possible to use a number of more sophisticated devices with a standard PC. But as I mentioned, the significance of SCSI is declining.

From the beginning, USB was planned to particularly replace the PC's serial COM ports. This has been very successful, and USB has become an indispensable element of the modern PC architecture.

At the time of writing, USB 2.0 is a completely new standard, but there is no doubt that it will be a big success, due to its good bandwidth and backward compatibility. As I mentioned, FireWire is used particularly on Macintosh computers and for DV cameras.

All digital cameras (as far as I know) have a FireWire port, which makes it very easy to download video footage for further processing on the PC. Windows XP is setup for this, as I described in my guide, "Do it yourself Windows XP".

Figure 211. In the middle of the picture you can see the small DV connecter on this digital video recorder.

For external disks (hard disks and CD/DVD drives), both USB version 2.0 and FireWire can be used. Maxtor, for example, supplies large, cheap, external hard disks using both interfaces.

Controller	**Bandwidth**
USB 1.1	1.2 MB/sec.
FireWire	40 MB/sec.
USB 2.0	40 MB/sec.
SCSI	Up to 160 MB/sec.

Figure 212. Four, partially competing, expansion buses.

- Next chapter.
- Previous chapter.

- Next chapter.
- Previous chapter.

Chapter 44. Hard disks, ATA and SATA

I am now finally going to describe the ATA interface, which has been mentioned several times earlier in the guide. The ATA interface is used for hard disks (as in Fig. 20, and the motherboard's ATA controller is built into the chipset's south bridge, as shown in Fig. 195.

We have also seen, that ATA devices use *bus mastering* to exchange data directly with RAM. But what does this interface actually consist of? And why do new standards for hard disks keep coming out? We are going to look at that now.

History of the hard disk

It is critical that we have a place to save our programs and data when the PC is switched off. Otherwise it is useless. Twenty years ago, *diskettes* were used, but their capacity is very limited.

Right back in 1957, IBM introduced the first "fixed disk storage". It was a big thing with 50 disks that were 24 inches in diameter. This very early hard disk had a capacity of 5 MB – a huge storage space at the time, which cost $35,000 a year to lease. One of IBM's models was called the 3030, and in the weapons-conscious USA, this led to the hard disk being given the nickname, *Winchester disk* (a Winchester 3030 was a popular rifle) – a nickname that was also used in Europe for many years.

The first computer with a hard disk was IBM's RAMAC, which was used during the 1960 Olympics to calculate sports results. A bit later, in 1962, removable *disk packs* were developed – a forerunner of the floppy disk. In 1964, the CRC algorithm was introduced. It provided greater security by checking and comparing data before and after it was written to the disk. In 1971, the first 8-inch diskettes came onto the market.

But it wasn't until the middle of the 1980's that people began to use hard disks in more standard PC's, and since then development has surged ahead. The capacity of a standard hard disk has actually become *a thousand* times greater in the period 1990-2000.

The standard user's need for disk space (e.g. for digital photos, video and music) has grown in step with this, so that 120-250 GB of disk space or more is normal in many PC's – a figure which will double over the next few years.

Hard disks are constantly being developed which have greater capacity and speed (the two go together, as we shall see), and there is therefore a constant need for new types of hard disk controllers. The companies leading the development are Maxtor, Western Digital, IBM/Hitachi and Seagate.

Figure 213. An SATA-hard disk (here 160 gigabytes). This is a standard product, which you can buy anywhere.

The physical disk

Hard disks consist of one or more magnetic plates mounted in a metal box. The standard size (as in Fig. 213) is about 10 x 14.5 x 2.5 cm, and such a device can contain hundreds of gigabytes of data. Inside the box, a number of glass or metal plates whir around at, for example, 5400 or 7200 revolutions per minute -– these being the two most common speeds.

The read/write heads hover over the magnetic plates, and can transfer data at a tremendous speed:

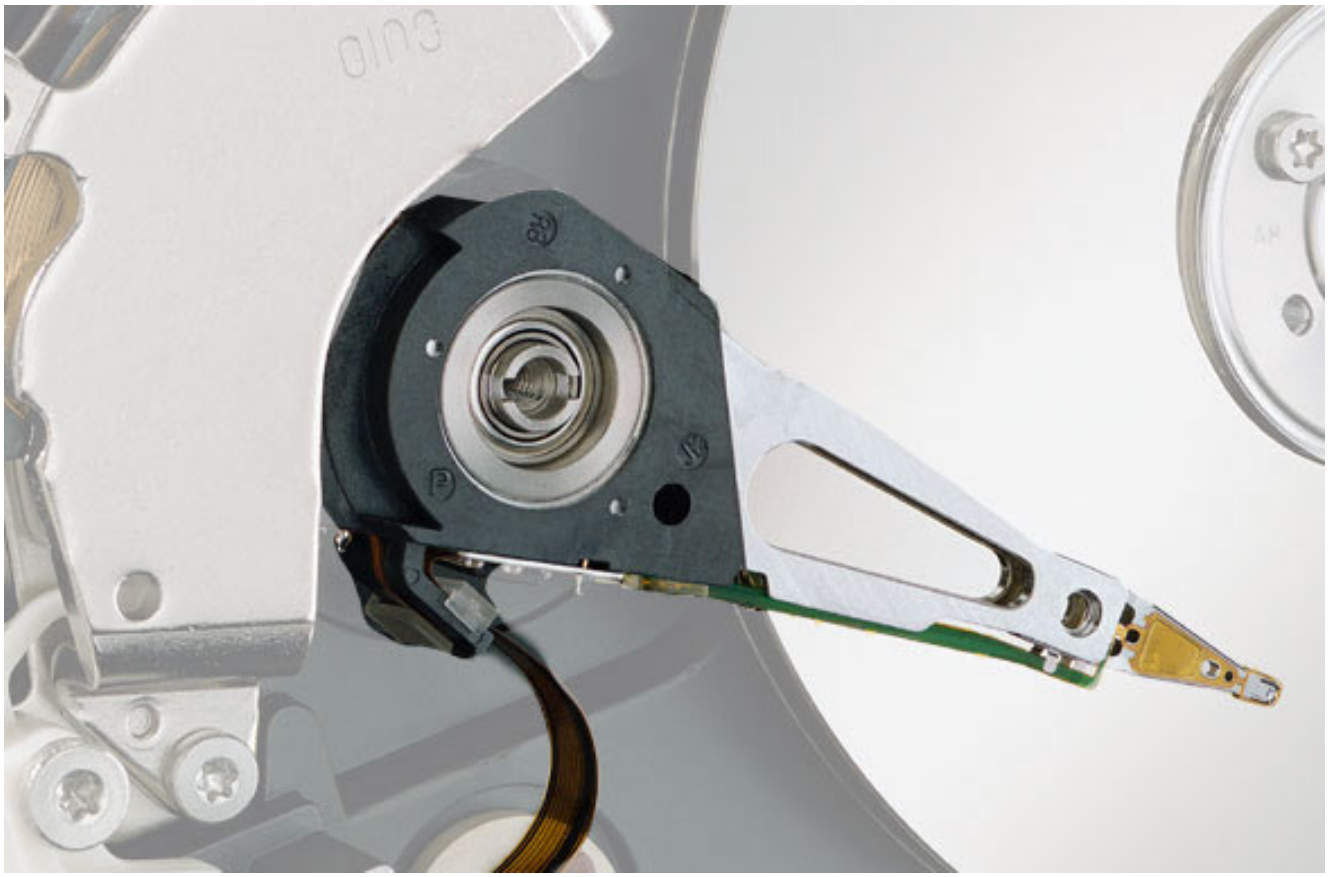


Figure 214. One of the read/write heads, which can swing across the plates.

The actual read/write head is a tiny electromagnet. The magnet ends in a C-shaped head, the shape of which ensures that it virtually hovers above the magnetic plate. Under the read/write head are the disk *tracks*. These are thin rings packed with magnetic particles. These magnetic particles can be arranged in patterns of bits, which are translated into 0's and 1's by the electronics of the hard disk:

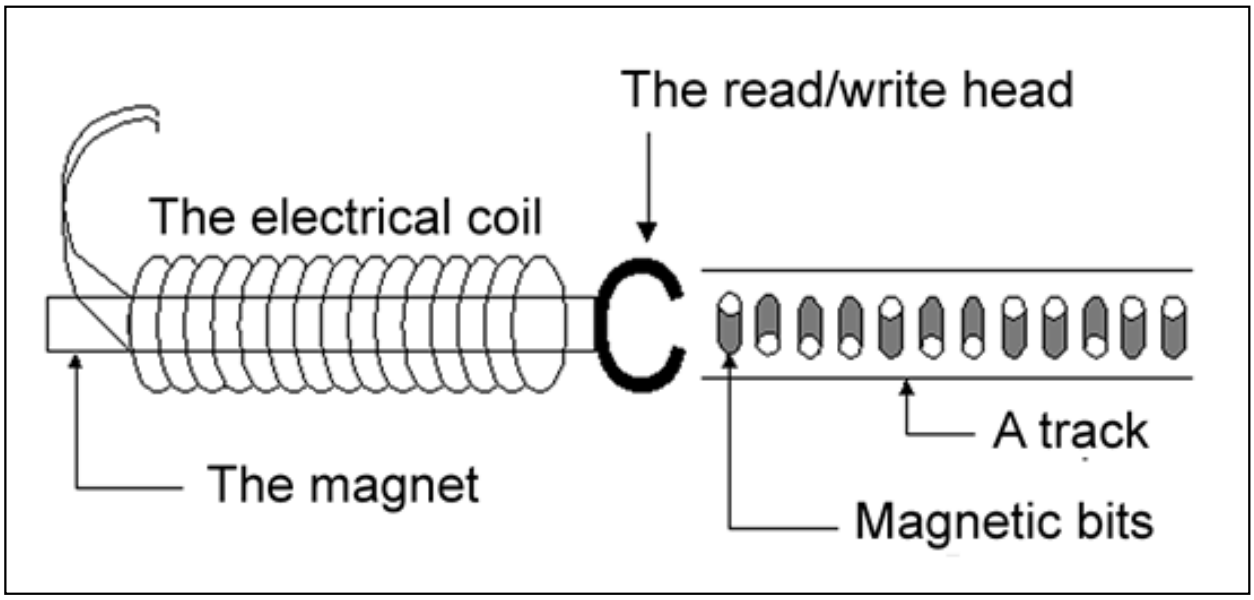


Figure 215. The data on the hard disk is created using electrical induction.

When the disk moves under the read/write head, it can either read the existing data or write new data to the disk:

- If current is supplied to the coil, the head will become magnetic. This magnetism will reorganise the tiny magnetic bits in the track, so they represent new values. Thus data is *written*.

- If the head is moved over the track without any current applied, it will gather up the magnetic pattern

from the disk. This magnetism will induce current in the coil, and this “current” contains the track’s data, which is thus *read*.

Figure 216. A peek inside the hard disk reveals the magnetic plates, which have a diameter of 3½ inches.

The read/write heads are the most expensive part of the hard disk. When the disk is switched off, the heads are parked in a special area, so that they will not be damaged during transport. In Fig. 216, the cover has been removed from a hard disk, and you can see the uppermost arm with its read/write head.

Tracks and sectors

Each hard disk plate is divided into *tracks*. Each track is subdivided into a number of *sectors*, the disk’s smallest unit. A sector normally holds 512 bytes of data.

The individual files are written across a number of disk sectors (at least one), and this task is handled by the *file system*.

The file system is part of the operating system, which the disk has to *formatted* with. In Windows 98, the FAT32 file system is used. In Windows 2000/XP, you can also use NTFS. They are different systems for organising files and folder structures on the hard disk. You can read more about the file system in my guide, ”Do it yourself DOS”.

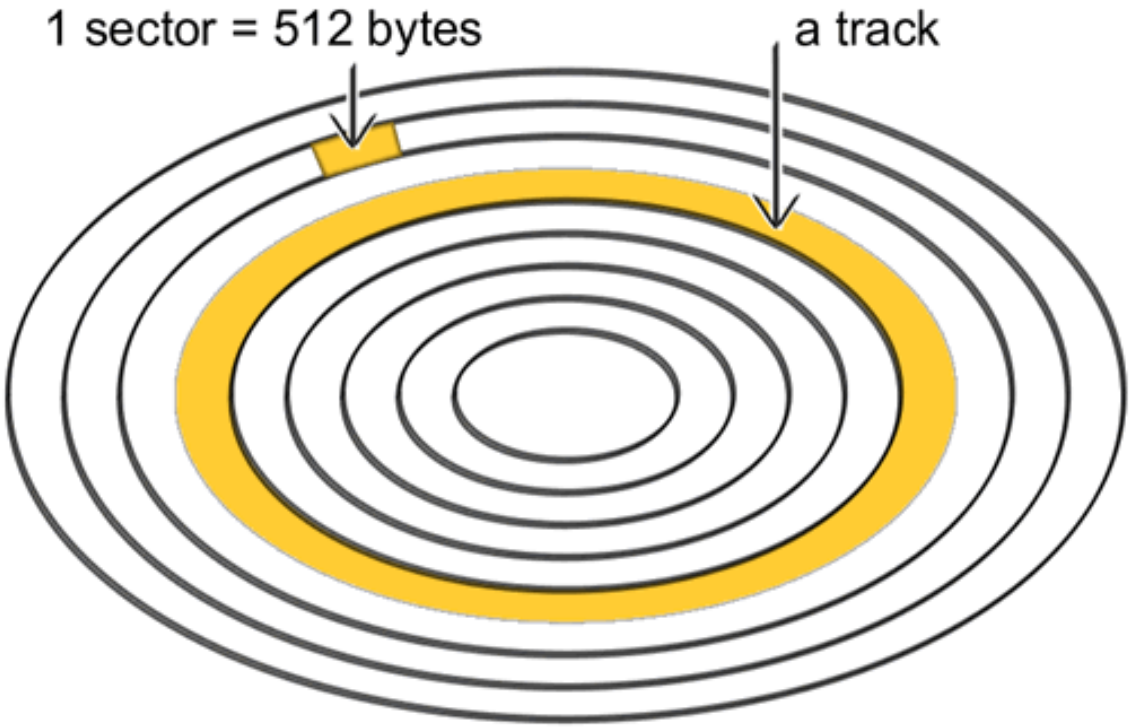


Figure 217. Hard disks are made of a number of disks, each side of which is divided into a large number of tracks. Each track contains numerous sectors, the smallest unit on a disk. In order to be able to use the sectors, the disk has to be formatted with a file system.

Hard disk development

New hard disk models are constantly being developed, and for each new generation, the capacity becomes greater. This is possible due to new read/write heads and magnetic plates which are *more dense*. When the magnetic density is increased, hard disks can be produced which have greater capacity for the same number of plates.

Another consequence of higher density is that the disks become *faster*, since they can transfer more data per rotation. There are simply more bits hidden in each track.

All hard disks have a certain amount of *cache* installed as 2 or 8 MB of fast RAM, which functions as a buffer. The cache helps ensure that the data gathered by the mechanics of the disk is optimally exploited.

The new and faster hard disks are always followed by new types of interface, since the controller principles have to be upgraded as well in order to handle the large amounts of data. At the time of writing, the fastest ATA interface is called ATA/133, but it is about to be replaced with Serial ATA (see Fig. 221).

The hard disk's interface

The hard disk is managed by a controller built into the actual unit. This controller works together with a similar controller linked to the PC's I/O bus (on the motherboard). I showed this setup in Fig. 20, in order to explain the concept of "interface".

The interface's job is to move data between the hard disk sectors and the I/O bus as fast as possible. It consists of:

- A controller which controls the hard disk.
- A controller which connects the hard disk to the motherboard.
- A cable between the two controllers.

In the ”old days”, interfaces such as *ST-506* and *ESDI* were used. The ATA (*AT Attachment*) interfaces are based on the IDE standard (*Integrated Drive Electronics*). The names, IDE and ATA are “owned” by Western Digital, but are used anyway in everyday language.

Figure 218. Cheap ATA disks with good performance. Prices from October 2004.

Enhanced IDE

The IDE standard wasn’t especially fast, and couldn’t handle hard disks bigger than 528 MB. So the ATA standard came out in the mid-1990’s, and is still used today for cheap, high-performance, mass-produced hard disks. ATA is an *enhanced* edition of IDE, where the interface was moved from the ISA bus to the high-speed PCI bus. In principle, ATA (or *parallel ATA)* can be used for a number of different devices. The most common are:

- Hard disks, CD-ROM/DVD drives and burners.
- Other drives (such as the diskette formats, Zip and LS-120, etc.) and tape units.

ATA was developed as a cheap, all-round interface, designed for the different types of drives.

Host controller with two channels

The ATA interface can be seen as a bus, which is managed by a *host controller*. Up to four devices can be connected per host controller, and the devices connect directly to the motherboard.

The slightly unusual thing about the ATA interface is that there are two *channels*, which can each have two devices connected. They are called the *primary* and *secondary* channels. If two devices are connected to one channel, they have to be configured as one *master* and one *slave* device:

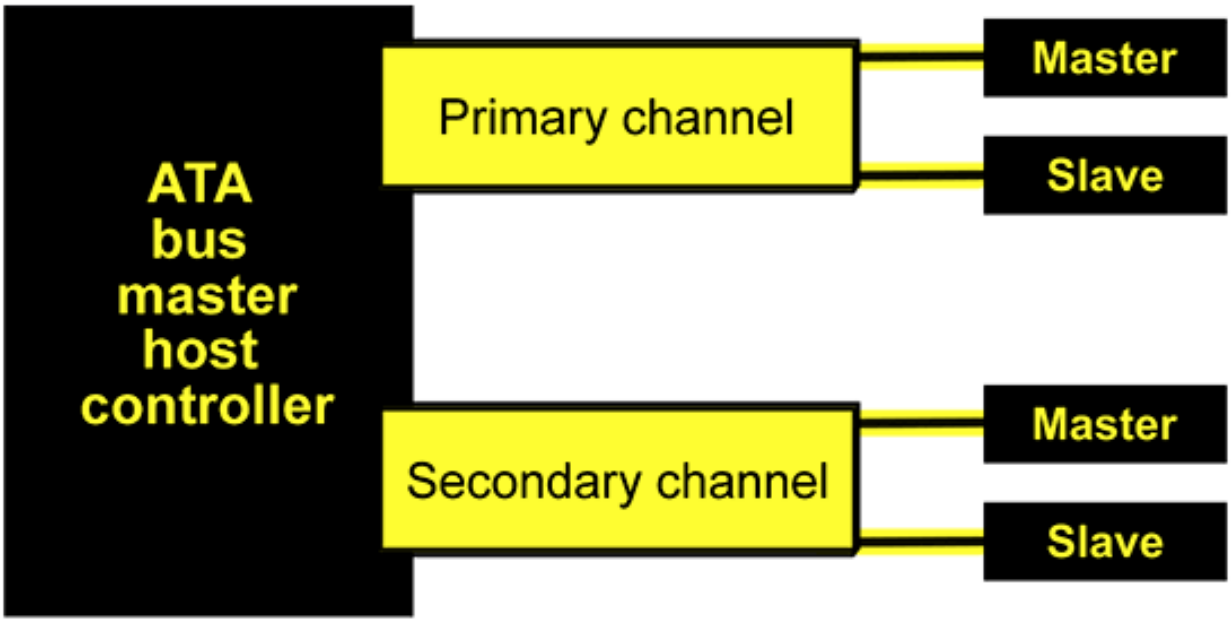


Figure 219. The ATA system’s four channels.

These four channels are standard on today’s motherboards using parallel ATA. Motherboards typically have connectors for two ATA cables, which can each connect two devices (master and slave).

These rectangular, male ATA connectors are placed in a fairly accessible location, and are used to connect the ribbon cables which fit hard disks and CD/DVD drives.

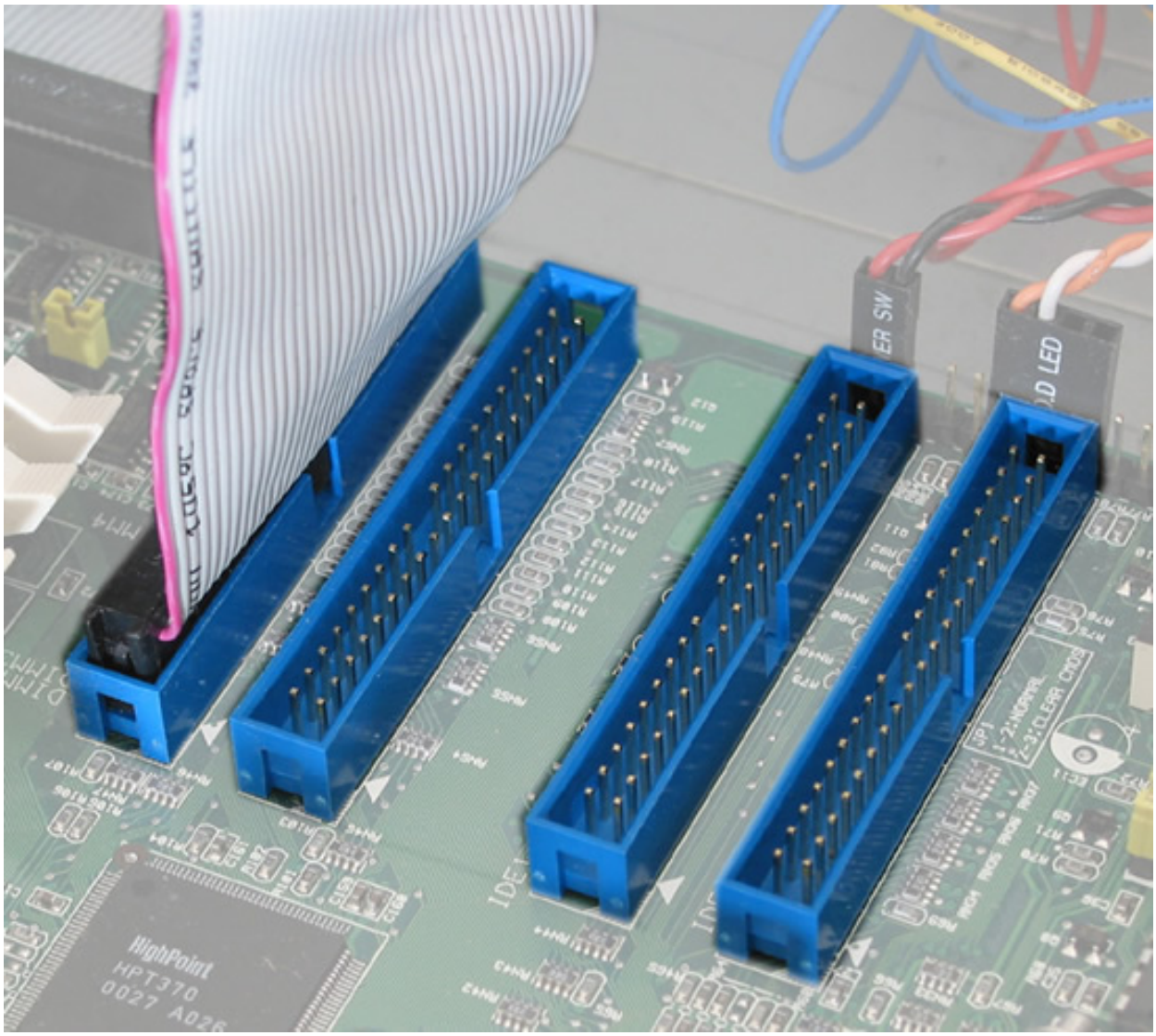


Figure 220. This motherboard has an extra, built-in RAID controller, so there are four ATA connectors in total. Each one can handle two devices.

Transfer speeds and protocols

There are many ATA versions. The *protocol* has been changed over the years, but on the whole, the products are *backward compatible*.

This means that you can readily connect an old CD-ROM drive which uses the PIO 3 protocol, for example, to a modern motherboard with an ATA/133 controller. The drive just won't be any quicker as a result. The shift from parallel ATA/133 to *Serial ATA* is more profound. Here we use a different type of controller and new cables.

The various protocols have different transfer speeds, as shown in the table below.

Protocol	**Max. theoretical transfer rate**
PIO 3	13.3 MB/second
PIO 4	16.6 MB/second
Ultra DMA (ATA/33)	33 MB/second
ATA/66	66 MB/second
ATA/100	100 MB/second
ATA/133	133 MB/second
SATA	150 MB/second

Figure 221. The various versions of the ATA standard.

The transfer speeds listed apply to the interface. Few hard disks can deliver more than 80 MB/second, but even so, it's still good to have the fastest possible interface. If two hard disks are used on the same controller, or if disks with a large cache are used, there can be a need for big bandwidth for short periods.

With the ATA/66 standard, a new type of ATA cable was introduced. Hard disk cables normally have 40 conductors, but in the new version this is extended to 80, as each wire is balanced by a ground connection. This reduces electrical noise and allows for greater bandwidth.

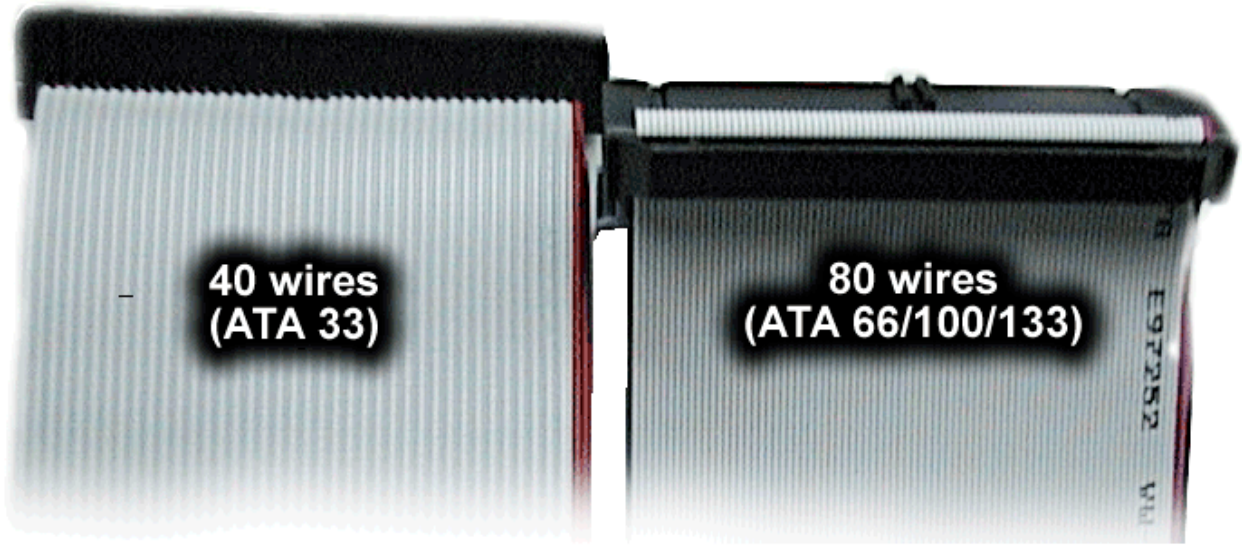


Figure 222. ATA cables

Parallel ATA setup

If you build your own PC or upgrade it using a new hard disk or CD/DVD drive, it is important that you know the limitations of the parallel ATA system.

Modern motherboards can automatically recognize ATA devices. This means that, in principle, you can install your drive and start the PC, and it should work. You can follow the startup program while it “installs” the drive, as shown in Fig. 223, where two hard disks and CD burner are installed.

In the first four lines, the controller *auto detects* the four channels for connected devices. It then ”mounts” (configures) the two Maxtor hard disks on the primary channel, and finally the HP burner on the secondary channel. You can read this in the lines below:

```
WAIT...
Auto-Detecting Pri Master..IDE Hard Disk
Auto-Detecting Pri Slave...IDE Hard Disk
Auto-Detecting Sec Master..ATAPI CDROM
Auto-Detecting Sec Slave...Not Detected
Pri Master: DAH017K0 Maxtor 4D060H3
            Ultra DMA Mode-4, S.M.A.R.T. OK
Pri Slave : DA620CQ0 Maxtor 53073U6
            Ultra DMA Mode-4, S.M.A.R.T. OK
Sec Master: Hewlett-Packard CD-Writer Plus 8100
```

Fig. Figure 223. The PC startup program automatically detects the three ATA devices.

Each ATA device has a small area containing jumpers which are used to set whether it is the master or slave device. If the startup is to proceed as painlessly as in Fig. 225, the jumpers have to set correctly on all devices, and of course the cables have to correctly connected.

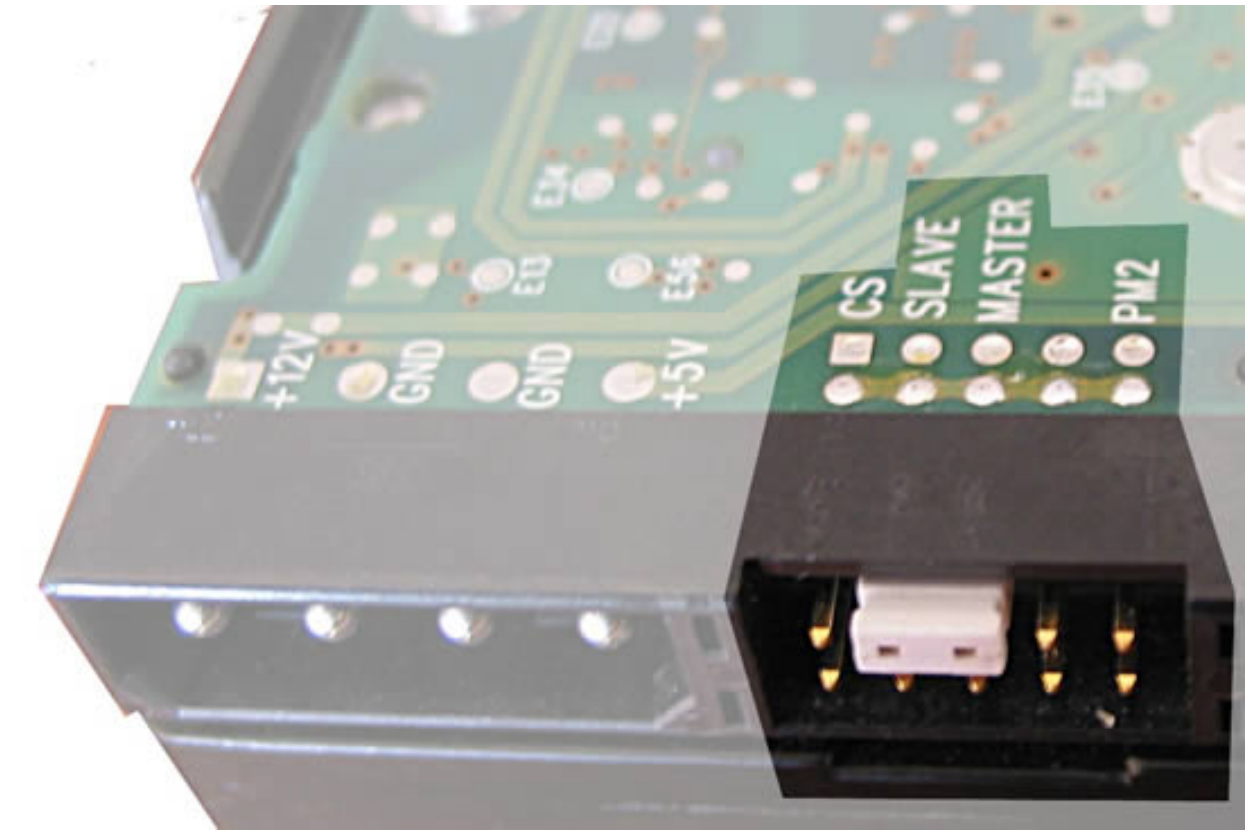


Figure 224. Jumper to change between master and slave setting.

Multitasking and protocol limitations

The host controller has two main channels, which operate independently of each other. One could say that each channel has its own controller. This means that the two channels can *multitask*. For example, if you have one hard disk on each channel, you can read data from one disk, while writing data to the other disk at the same time. Both disks will operate independently.

But this is not the case if two disks are connected to the same ATA channel. They cannot multitask. Either the master is working, or the slave is working, but not both at the same time. It is therefore best to install two hard disks on separate channels if they have to work at the same time.

The two main channels (primary and secondary ATA) can each run *their own protocol*, but the master/slave channels cannot do this. If you install two devices which use different protocols on the same channel, you run the risk that the slowest device will determine the speed for the whole channel. The optimal solution, therefore, is to connect your hard disk as the master on, for example, the primary channel, without connecting a CD drive as a slave device.

Look at the following picture. The CD burner is sitting by itself on the secondary ATA channel (*Secondary Master*), and this is for the sake of the hard disks. The two hard disks have been placed on the primary channel, since they both use the same protocol (ATA/66 in this case). The device's protocols can be seen on the right in the last two columns (*PIO* and *UDMA mode*):

Hard Disk(s)	Size	LBA Mode	32Bit Mode	Block Mode	PIO Mode	UDMA Mode
Primary Master :	61480MB	LBA	On	16Sec	4	4
Primary Slave :	30738MB	LBA	On	16Sec	4	4
Secondary Master :					3	N/A

Figure 225. The hard disks and CD burner are kept separate.

If you use all four ATA connections, it might look like this:

ATA connection	Device
Primary, master	Hard disk 1
Primary, slave	Hard disk 2
Secondary, master	CD ROM burner
Secondary, slave	DVD drive

Figure 226. All four ATA channels are being used.

A problem arises when you need to connect two hard disks, in addition to a CD/DVD drive or drives. This is not possible, without compromising on quality. The two hard disks have to placed on the same channel, since CD drives generally operate using a slower protocol. But this does not allow for the optimal utilisation of the hard disks – they should ideally have their own channel, all to themselves, for the sake of multitasking.

This is why motherboards with an extra built-in ATA RAID controller are much more preferable – with these you can install up to 8 ATA devices in the same PC (see Fig. 220. But if you ensure, right from the start, that you put a big hard disk in the PC, this helps.

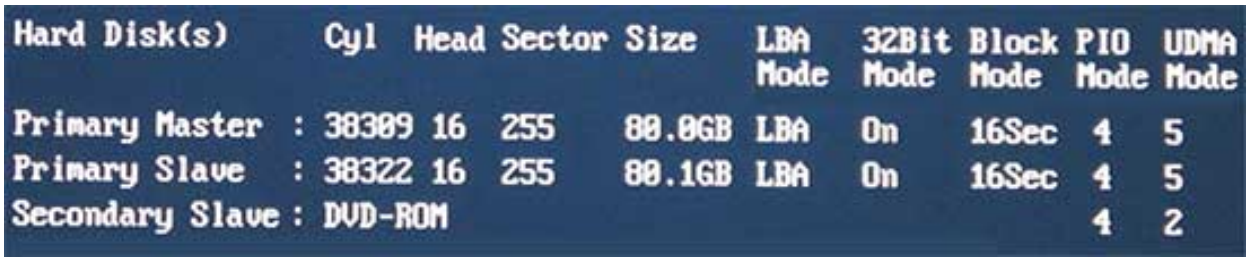


Figure 227.Without a RAID controller it is difficult to get the best performance using more than two parallel ATA drives..

ATA RAID

The ATA interface can quickly become a bottleneck, as I have just described. I therefore warmly recommend using a motherboard with an extra built-in RAID controller. This doubles the number of possible devices. Many motherboards (like my own from Epox) have both a standard ATA device in the south bridge, and an extra controller (from HighPoint or Promise), and this provides much greater flexibility.

Another option is to buy and external RAID controller which can be mounted in an expansion slot. This will also allow for a further four ATA devices to be connected. For years I used the Fasttrack controller from Promise, which you can see in Fig. 170. If you look carefully you can see two ATA connectors at the top of the card.

Since the ATA limitations have become more pronounced in recent years, most motherboards today can be bought with an extra built-in RAID controller, and that's the easiest solution.

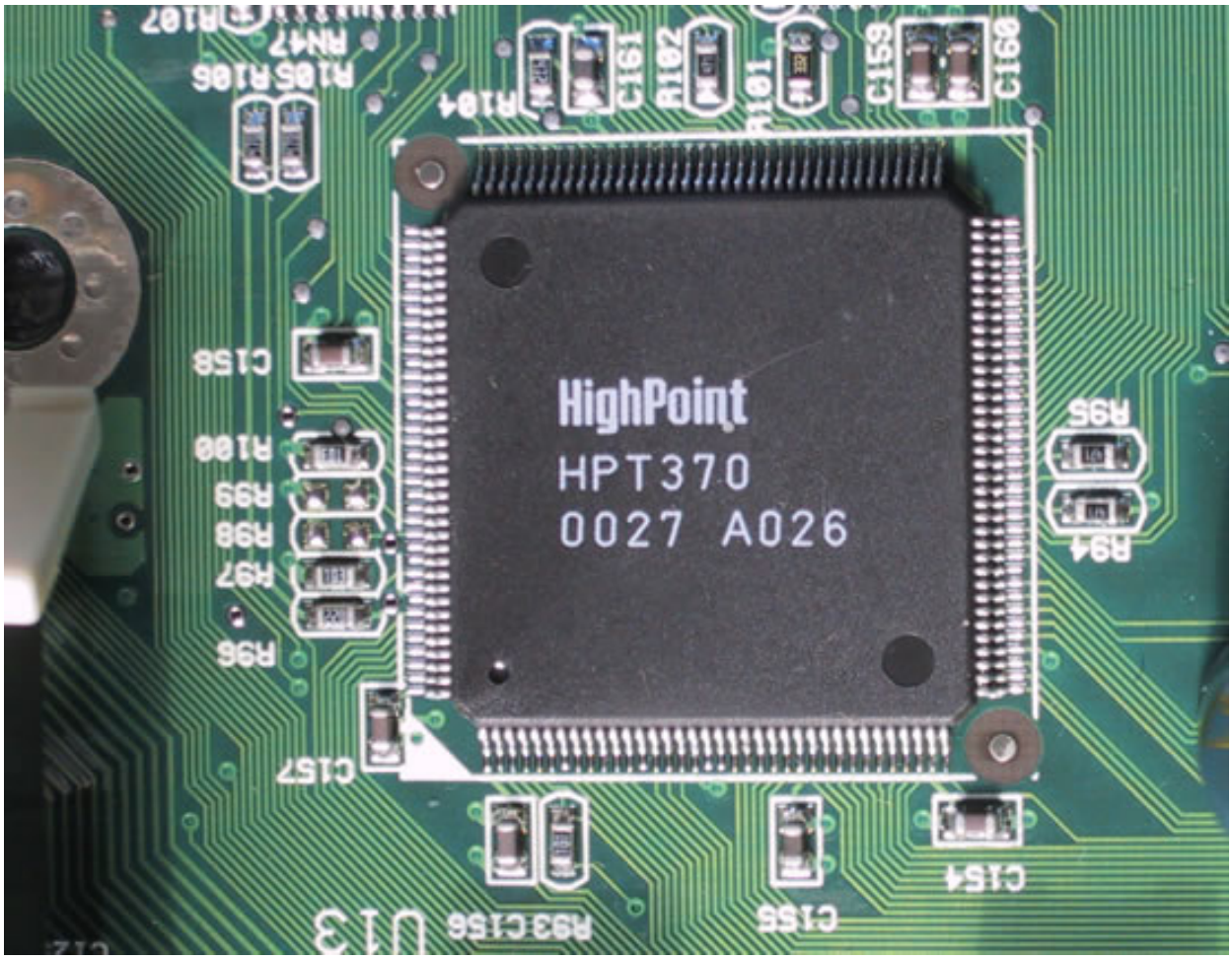


Figure 228. An ATA-RAID controller which is integrated into the motherboard.

The RAID controller actually allows for *striping* of drives (see Fig. 206, but you may also simply use it as an extra ATA controller.

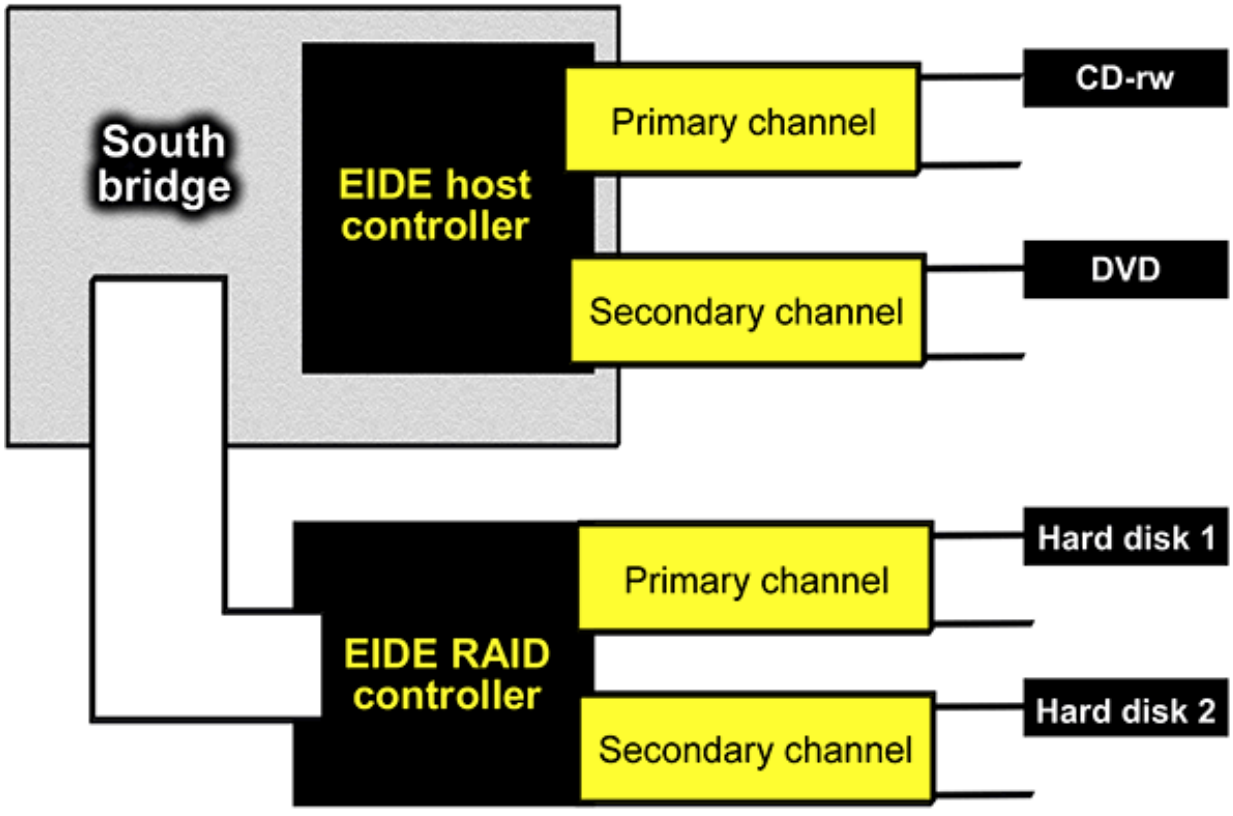


Figure 229. All four ATA devices have their own channel to work on, and that's the optimal solution.

This way one can use two hard disks, which each have an ATA channel all to themselves. Just as the two optical drives each have their own channel also. Perfect! The hard disks run at full speed, and the CD drives have the optimal configuration.

SATA - Serial ATA

The old well-known, parallel ATA system is clumsy and in-flexible. The big ribbon cables take up to much room in the computer cabinet and they reduce the circulation of air. The cables have a fixed length and they make it difficult to mount the harddisks.

Furthermore the PATA-system with master/slave channels is difficult to work with, and the band width is limited.

The successor to the ATA system is called *Serial ATA*. Here we have an interface using small handy cables with only 7 wires in stead of the big 40/80-wired cables used in the Parallel ATA-interface.

Serial ATA is a high-speed serial interface in family with Ethernet, USB, FireWire and AMD's Hyper Transport. All these interfaces use a serial technology. They only have two channels: one receiving data and one transmitting them. This can be achieved with a very simple cabling. The data communication only requires 0,25 Volt compared to the 5 Volt of parallel ATA.

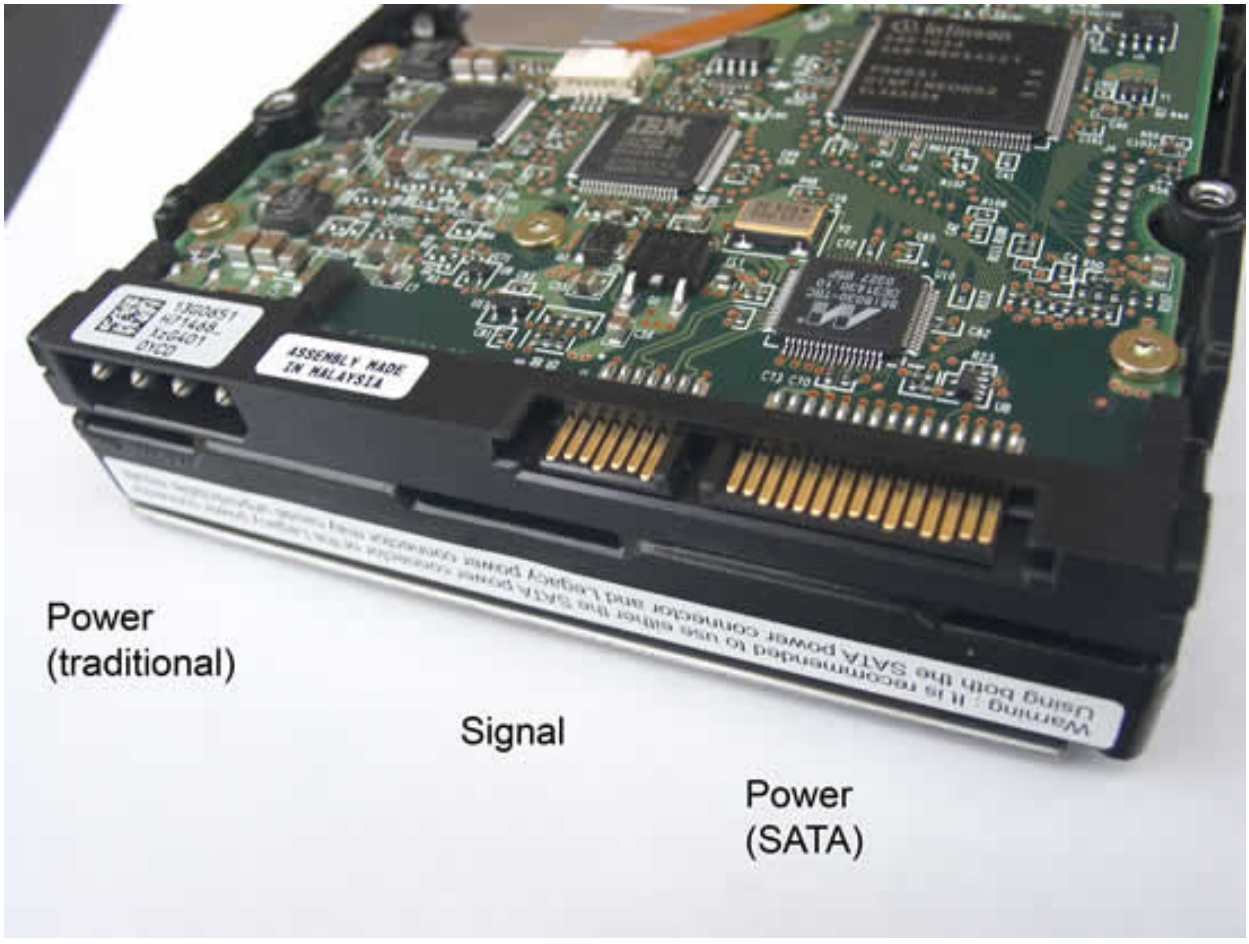


Figure 230. The connectors on the SATA hard disk.

The existing PATA system has a limit of 128 GB volume harddisks. This gave problems with the first 160 GB disks. The parallel system only operates with 28 bit data for addressing the disks. It can only adress 2^{28} sectors. Since each sector on the hard disk holds 512 bytes of data, we have 2^{28} times 512 bytes data, which is 128 GB.

To use bigger hard disks, the operating system has to address the sectors directly. In Windows XP this function is called *48 bit LBA*, and is it has caused some troubles to have it work. Using Serial ATA, there are no problems with bigger hard disks.

More about Serial ATA:

- 150 MB/sec. data transfer, initially. Later 300 and 600 MB/sec. in new versions of SATA. Here is room here for at least the next 5 years of development in the area of hard disk technology.
- Hot-plugging. You do not have to close the PC to install or remove SATA disks.
- More intelligence in the controller (SCSI-like).
- No jumpers for master/slave.
- 8 mm cable.
- In-expensive manufacture and easy installation.

The controlling logic of Serial ATA is much more sophisticated (and SCSI-like) than in the traditional ATA interface. Serial ATA can process several commands at the same time and re-arrange them for better efficiency.

Figure 231. Cable for Serial ATA.

Some motherboards available in 2004 have SATA controllers based on a *bridge*. In reality it is just a converter, giving an old-fashioned PATA controller SATA functionality.

These boards take both parallel and serial ATA-disks. This sis great when you have old hard disks, you need to reuse with a new motherboard.

However, this bridged controller gives no room for better bandwidth. So to se better performance from the SATA hard disks, we need a new generation of motherboards without PATA-compatibility.

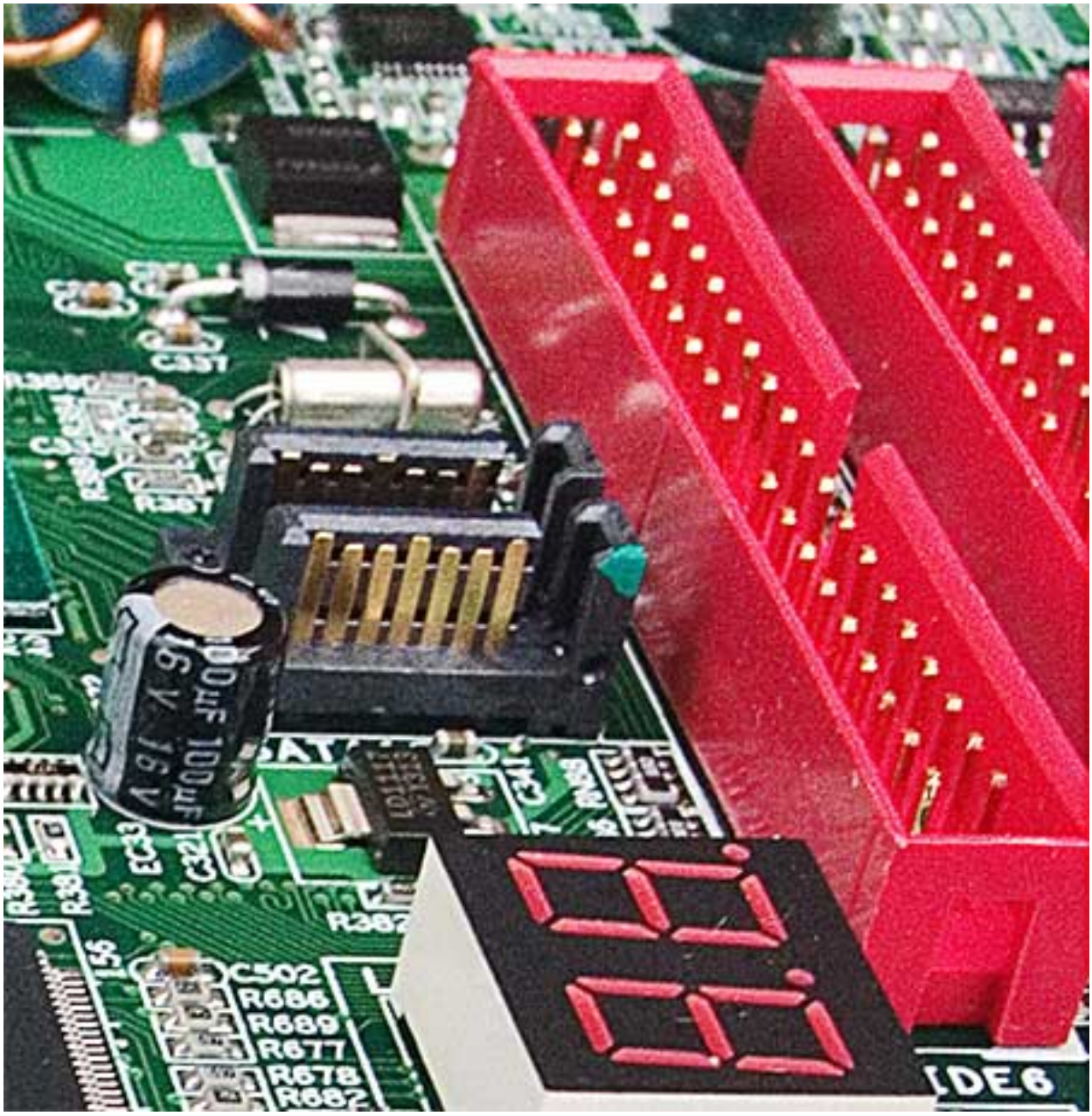


Figure 232. The small SATA connector is seen left to the bigger PATA-connector. This motherboard support both interfaces.

- Next chapter.
- Previous chapter.

- Next chapter.
- Previous chapter.

Chapter 45. System software. A small glossary

We have now examined most of the PC's architecture, from CPU and RAM, to expansion buses, to the hard disk. We still need to look at the motherboard's built-in system software.

There is software stored on every motherboard. The ROM circuits contain important system routines which help to startup the PC and which hold everything together. The following chapters provide a brief introduction to these topics:

- The BIOS and startup programs.
- The Setup program and CMOS storage.

The guide concludes with a glossary.

PC startup

When you switch on the power to your PC, a lot of things happen. You here the noise of the various cooling fans, and shortly afterwards, text starts to scroll up onto the screen. It is the *system software* which is doing this work.

Remember that the PC cannot do anything unless it receives *instructions*. Instructions are fragments of programs which are loaded into the CPU, and the CPU starts by executing the system software which is stored on the motherboard. Later, once the PC is up and running, the operating system can fetch instructions (programs) itself from the hard disk; but during startup, the CPU is fed instructions from the ROM code in the motherboard.

Figure 233. A ROM circuit containing Award BIOS (typically 1 or 2 MB of data).

That is, the startup programs are stored in ROM circuits. ROM stands for *Read Only Memory*. These circuits contain data, which can normally only be read. Thus the PC is "born" at the manufacturer with

system software stored in its hardware.

On newer motherboards, however, *Flash* ROM is used (so-called EEPROM circuits). With these, the data can be changed by the user (*BIOS updates*). For convenience, these circuits are still called BIOS ROM.

BIOS is important system software, because it is only after these programs have been loaded and executed that the PC's operating system can be loaded from the hard disk (or alternatively, from a diskette or another drive). This is called the boot process.

Checking the hardware

When the power supply is activated, the CPU fetches the first instructions from the ROM BIOS. Then the POST routine starts, which checks the hardware devices. POST stands for *Power On Self Test*, and this is quickly accomplished. Text doesn't reach the screen until POST has been executed.

If POST encounters a fault in the machine, the program will write a message on the screen. If the screen has not yet been made ready, or if the fault is, for example, linked to the video card, the program will normally emit beeps using the PC speaker. The pattern of bleeps and beeps varies for the different BIOS manufacturers, but the pattern indicates where the fault is located. For example, 8 beeps from a BIOS from AMI can mean a fault in the graphics system, while a constant series of short beeps indicates a fault in the RAM when using BIOS from Award. Some motherboards have built in LED's which can also signal faults (you can actually see these in Fig. 30. The fault messages are always explained in the motherboard manual.

When POST has finished executing, you normally hear a single beep from the speaker, and startup continues. Next the BIOS is loaded for the video card. This leads to the first text on the screen, which is normally the name of the BIOS supplier and the program version.

The startup program is now in the process of checking the various hardware, and generally "bringing the machine to life". You can make contact with the Setup program at this time, for example by pressing the Delete key once. After this you will see that the "RAM is being counted". You can also read which CPU is in the machine. Any error messages (e.g. if the hard disk is not connected properly) can now be seen on the screen.

Try to follow the startup process yourself when you switch on your PC. You can stop the process by pressing the Pause key, so that you have time to read the messages. Below you can see the startup messages for a PC with Award BIOS, which has found 512 MB of RAM.

Figure 234. At the top of the screen you can read that Award has supplied the startup program.

The startup program installs the other system devices, such as floppy disk and ATA drives, and locates the "logical devices" (such as COM, LPT, etc.). The PCI bus is scanned for devices.

The last link in the startup process is that the BIOS looks in the CMOS storage to find the chosen boot device. Normally it has to boot from one of the hard disks, and the BIOS thus has to read the contents of the *master boot record* (which is a particular sector on the hard disk). It then continues by loading the operating system from the hard disk, and the startup programs have played out their role.

CMOS and Setup

The startup program needs information about the PC's hardware. However, some of this system information has to be manually entered. This includes things like whether a floppy disk drive is installed, and the actual time and date, etc. Fortunately we don't have to type in this information every time we start the PC. It is done by the manufacturer, and the information is stored in a small CMOS chip.

CMOS is a special type of RAM, which excels at using very little power. The chip is used as a storage area for the small *database of hardware information*. The database is necessary for the startup programs, which, for example, use it as a list of the hardware which has to be checked. See also the discussion of ESCD, which is included in the CMOS storage. This is a store containing information about the PC's Plug and Play compatible devices.

The CMOS storage holds something like 256 bytes, and is maintained using power from the PC's small battery on the motherboard. Without a permanent power supply, all the information would disappear from the CMOS.

You can correct the settings in the CMOS storage yourself. You might need to do this, for example, if you install a new hard disk. That's why we have access to the CMOS via the *Setup program*, which is also stored in the motherboard's ROM circuits. Setup can be activated during startup by pressing a special key (e.g. Delete).

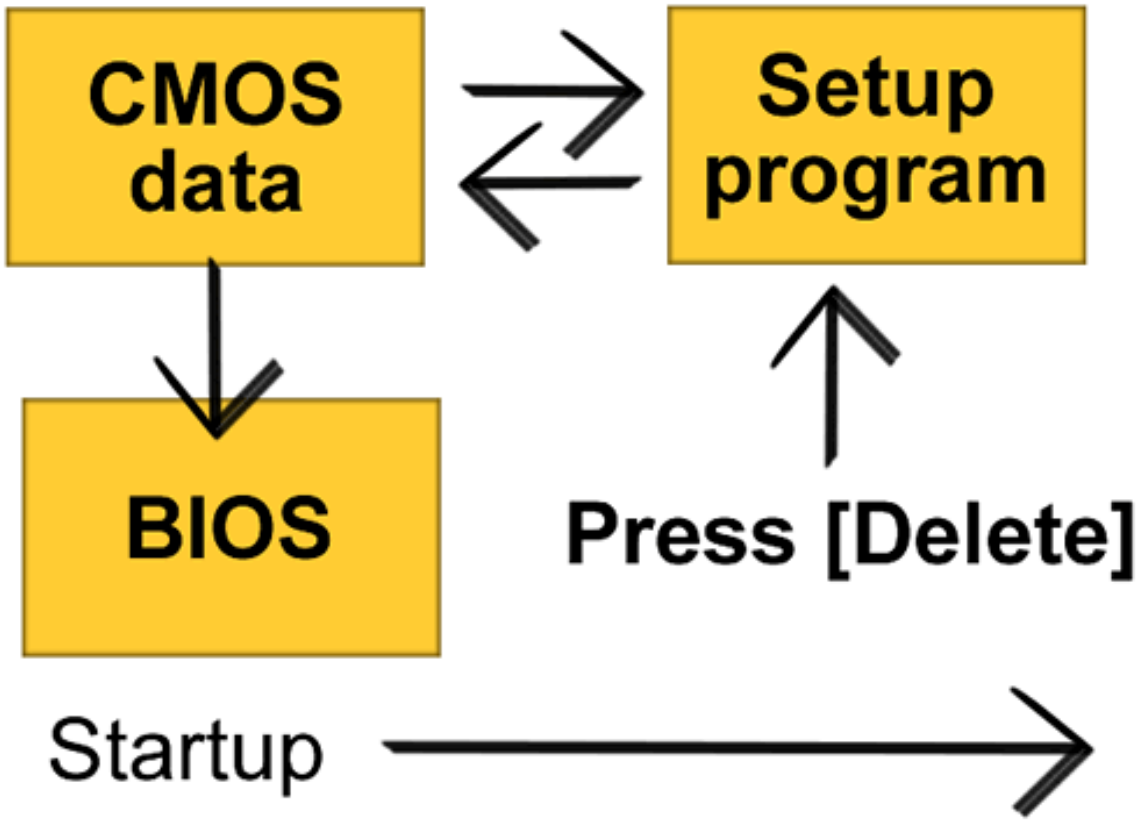


Figure 235. The Setup program is used to change the settings in the CMOS storage.

Chapter XX. A quick look at Setup

If you want to look at the facilities of the Setup program, you have to activate it while the startup programs are scrolling over the screen.

The Setup program can look a bit different, from one PC to another. Below is the opening menu from my

PC:

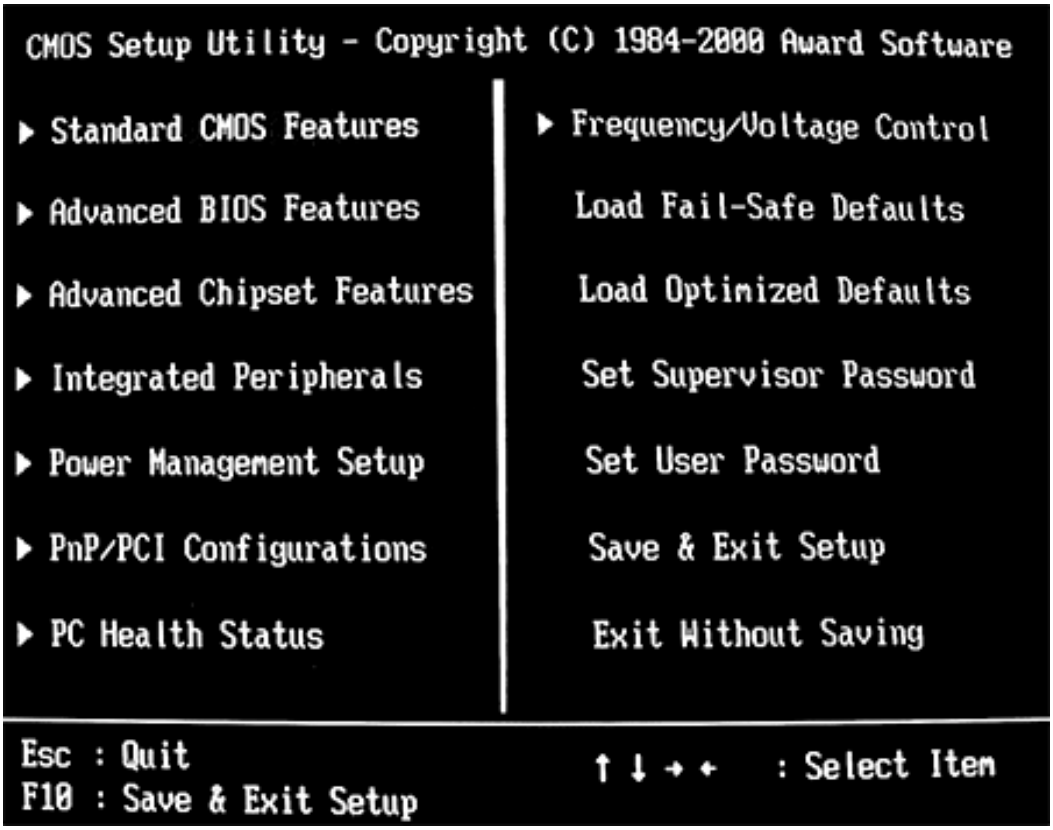


Figure 236. The opening menu from a version of the Setup program (from Award BIOS).

There are many options, which in this example have been divided into 14 different submenus. I won't go through the whole system; as most of it you will never use anyway.

But let's take a look at some of the options the various menus offer, including *features, setup* and *configuration*. Even though you won't need to change these settings on your own PC, they provide a good picture of how the Setup program can be used to adjust and actually program the motherboard in various ways,

Standard CMOS Features

This menu is one of the most fundamental of them all. Here you can change the date and time. Floppy disk drives are so "unintelligent" that the test program cannot check whether a floppy drive is installed or not. This must therefore be specified.

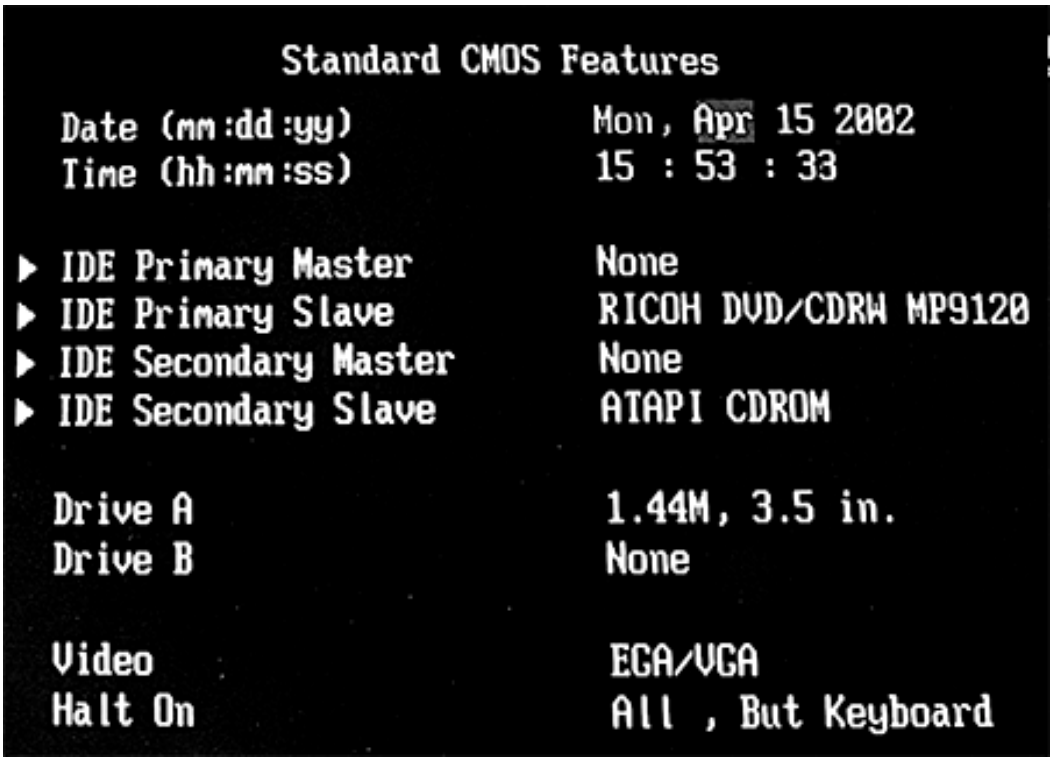


Figure 237. Standard CMOS Features is a small, easy-to-understand menu.

You can also set which ATA channels are in use. In Fig. 237 I have chosen the setting *None*, for the two unused ATA channels. This allows a slightly faster startup, since the program doesn't have to search for devices which don't exist.

Advanced BIOS Features

Here you can set which device should be used to boot from. It can be the hard disk or the floppy disk, and if you have several built-in hard disk controllers (ATA, RAID or SCSI), you specify which of them should perform the boot operation.

Many of the menu choices allow you to either *enable* or *disable* (activate or deactivate) various functions. In the fifth line (in Fig. 238) you can see that I have activated *Quick Power On Self Test.* This simplifies the POST program's job, so that the PC can be ready faster.

In Fig. 238, the *First Boot Device* has been set to the floppy disk drive. That means the PC will try to boot from this if there is a diskette in the drive. This may not be smart, if you are nervous about being attacked by a virus on the boot sector of a chance diskette which happens to be in the drive. It can also be nice to avoid the PC stopping during startup because there is a diskette in the A drive.

If instead you specify that the First Boot Device is HDD-0, the PC will always boot from the hard disk – whether there is a diskette in the drive or not. You may also to let the PC boot from the CD drive. That is fine when installing Windows 2000 or XP.

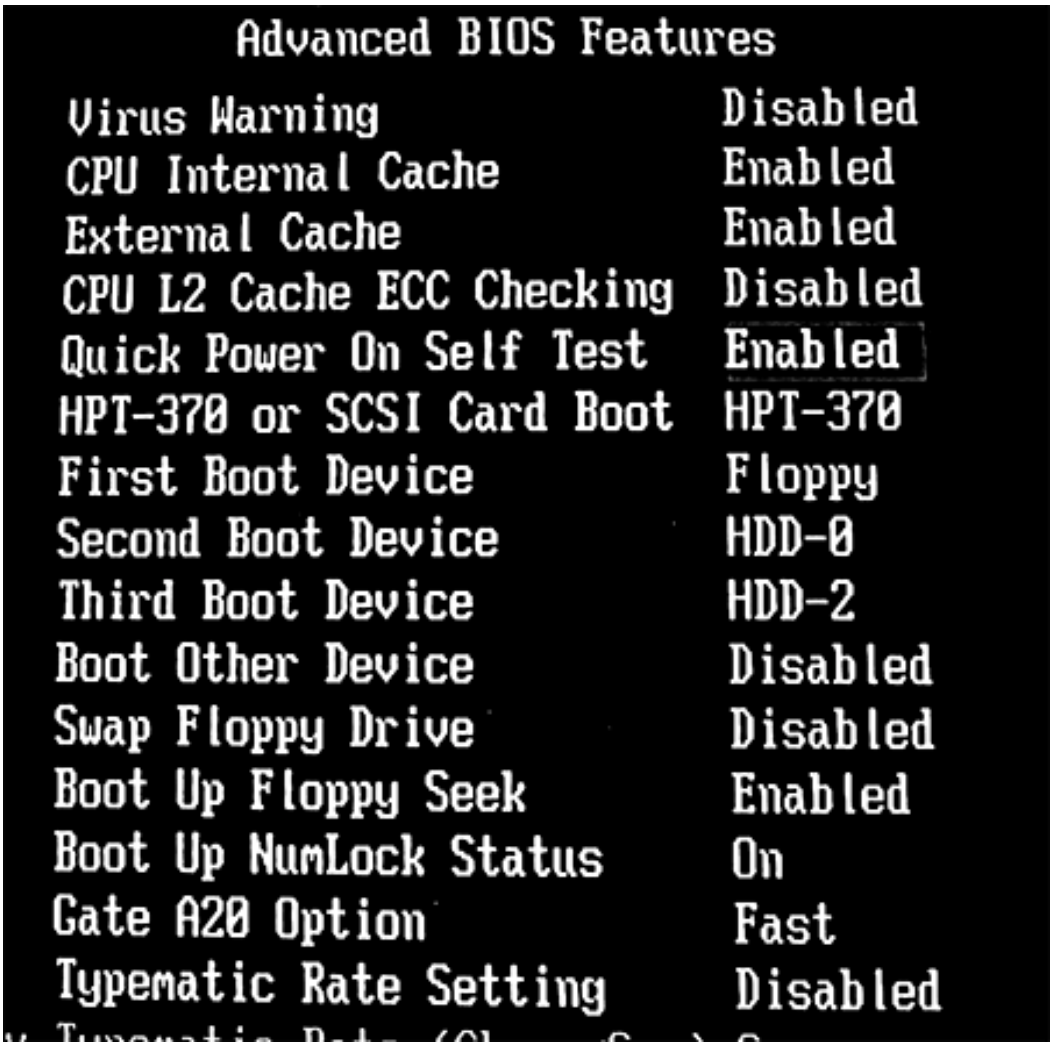


Figure 238. Advanced BIOS Features covers, for example, the various boot options.

Advanced Chipset Features

This menu is linked to the chipset, which can be programmed in various ways. For example, there are various advanced settings for the AGP and PCI buses. If you use a USB-based keyboard you can specify that here. Then it will work in 16-bit DOS mode (*real mode*) as well.

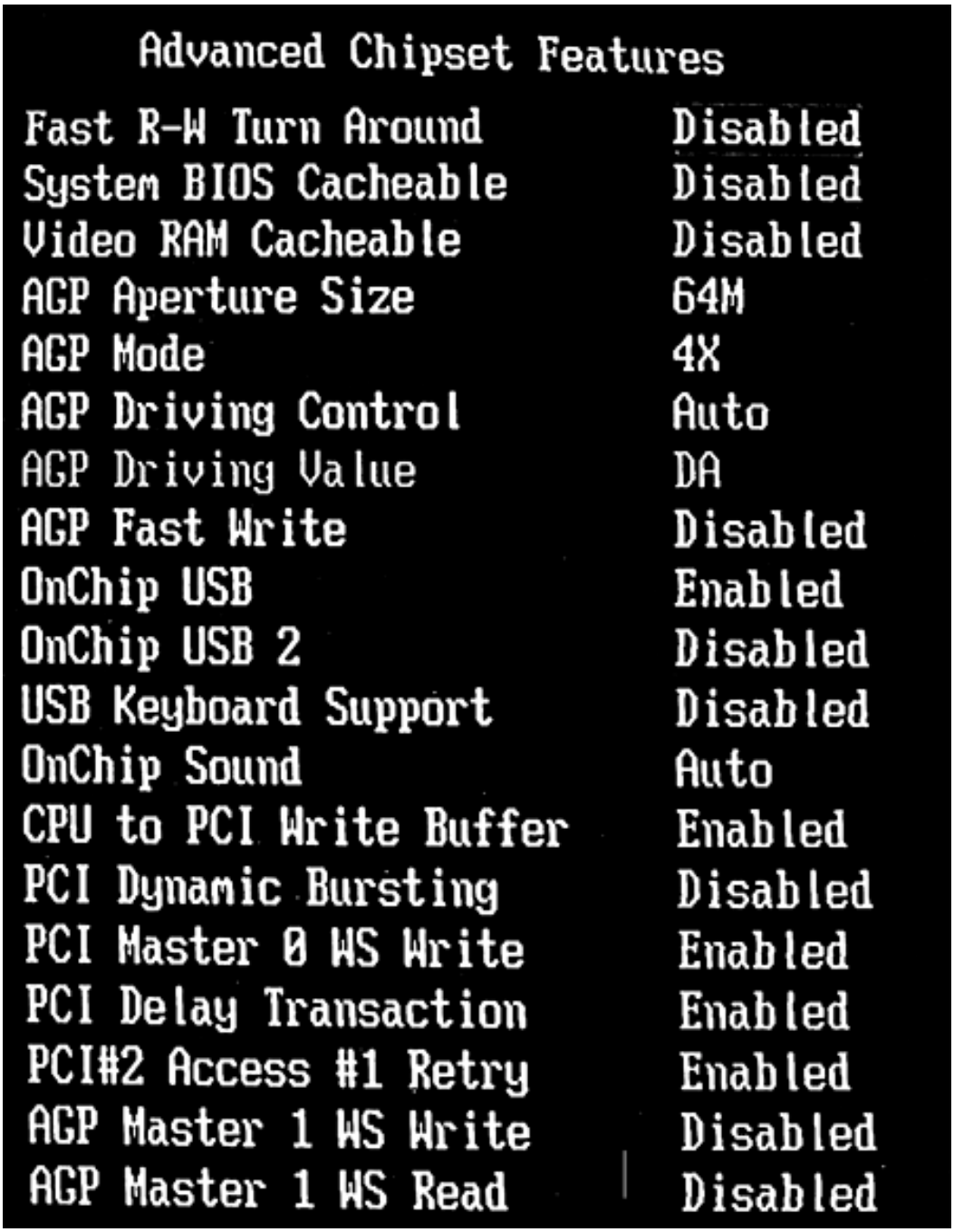


Figure 239. Advanced Chipset Features is linked to the PC's various buses.

The other settings

I have already mentioned that many users never end up fiddling with these Setup settings. There is therefore no reason for me to go through the other menus in detail. Let me just briefly discuss the main points:

Integrated Peripherals. This menu contains settings associated with the ATA and Super I/O controllers. For example, you can disable the floppy disk controller (FDC), so that a floppy disk drive cannot be connected. If the motherboard has a built-in sound device you might be able to disable it here.

Power management. This menu allows you to set how the various power saving functions should operate. There is nothing to benefit from this. The PC consumes the same energy with or without power management activated. On the hand power management can be quite irritating in daily use.

PnP and PCI Configurations. This menu allows you to allocated IRQ's for each PCI slot yourself. You can also change the PCI bus timing, should you happen to want to. See Fig. 174.

PC Health Status. This gives you a report on the CPU's current temperature and voltage, and how fast

the cooling fans are whirring (if they are connected in the right way). This menu is used, for example, by overclockers, who are very keen to know that the PC is not burning out.

Frequency/Voltage Control. This menu allows you to set the clock multiplier factor for the CPU, if it allows this, and adjust the voltage for the processor core, AGP system and I/O bus. See also Fig. 123.

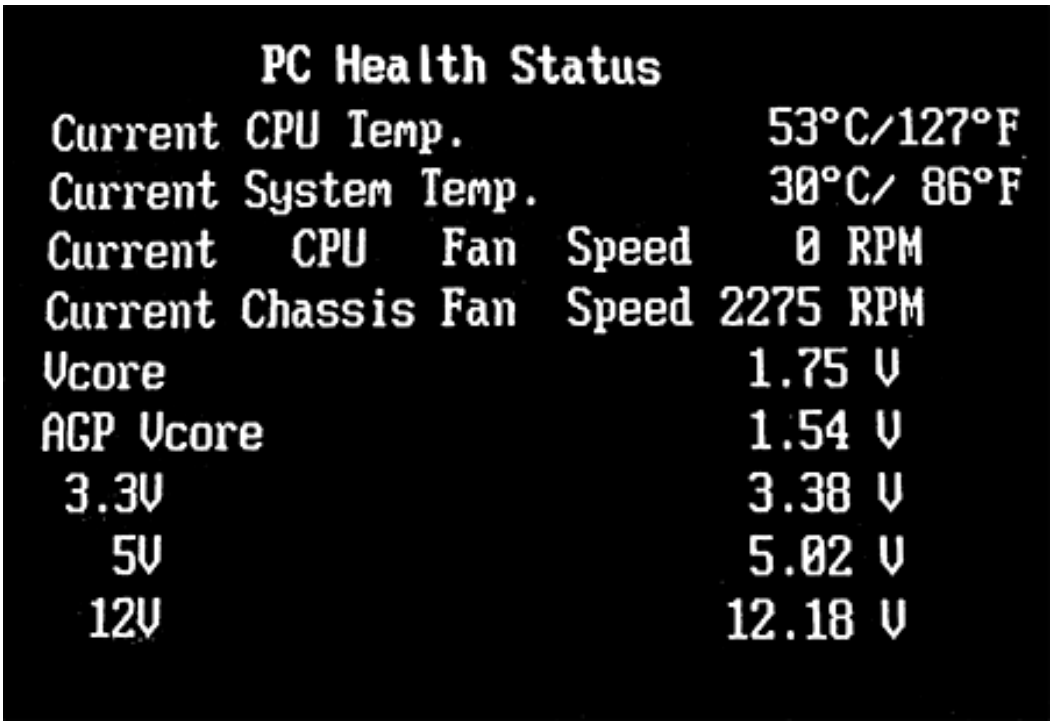


Figure 240. PC Health Status provides an instant report on the CPU's physical state.

Set Supervisor and **User password.** These menus allow you to choose a supervisor password, which is used to protect the Setup settings. This is used in some schools, where certain students have a tendency to fiddle with the computer's setup options.

Conclusion

The CMOS storage contains a number of essential items of system data, which are used by the startup programs. The user has the option to change these settings via the Setup program.

Many of the more advanced options are not relevant to normal users like us. They are linked to the controller chips on the motherboard, which can be configured in various ways, but there is no reason to change these settings. The motherboard manufacturer has already configured everything optimally, and they recommend in the manuals that you don't change anything. But if you are interested in your PC at the hardware level, you should definitely acquaint yourself with the facilities of the Setup program.

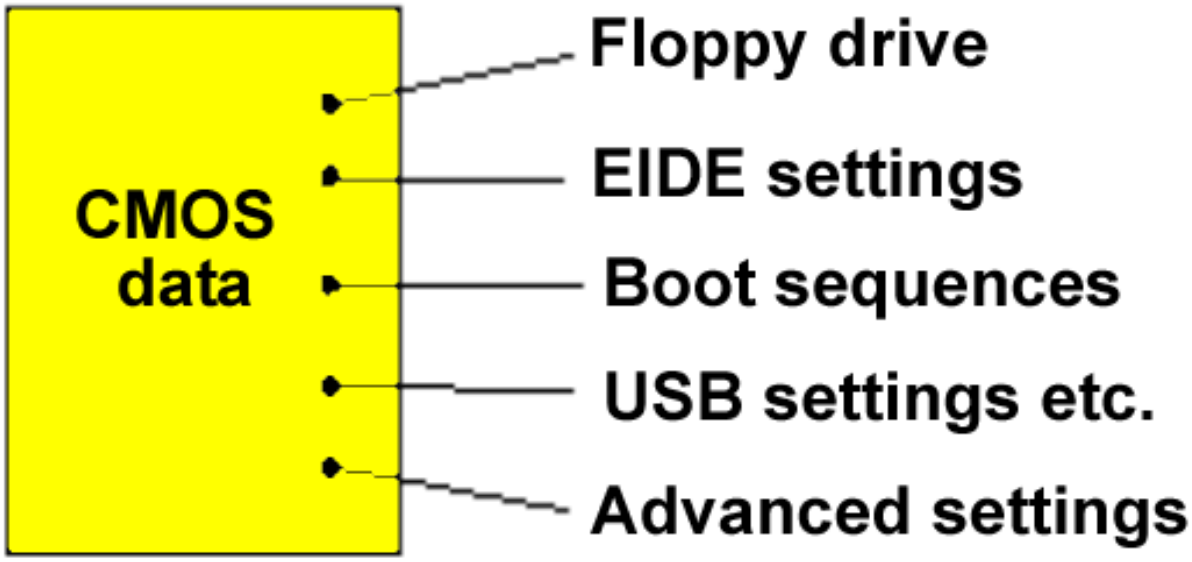


Figure 241. The system information in the CMOS storage can be changed if necessary using the Setup program.

Resetting the CMOS storage

If you look at the main menu in Fig. 236, there are options there to reload the standard settings (*Load Fail-Safe Defaults* and *Load Optimized Defaults*). You can make use of this if, after playing around, you find that your settings no longer work.

It is a very good idea to write down what you change on a piece of paper – especially the first time you work with Setup. Record the menu and setting you change, and what the value was before you changed it.

If it gets right out of hand, you can always reset the CMOS. You might need to do this, for example, if you set up password protection for Setup, and then forget the password.

The motherboard as a jumper which erases the CMOS data. You move the jumper, start the PC, and the data is erased. You then enter new data and move the jumper back again.

Another method is to remove the motherboard battery. This maintains the CMOS data, so all the data is erased if you briefly remove the battery (which you can ease out of its round plastic holder).

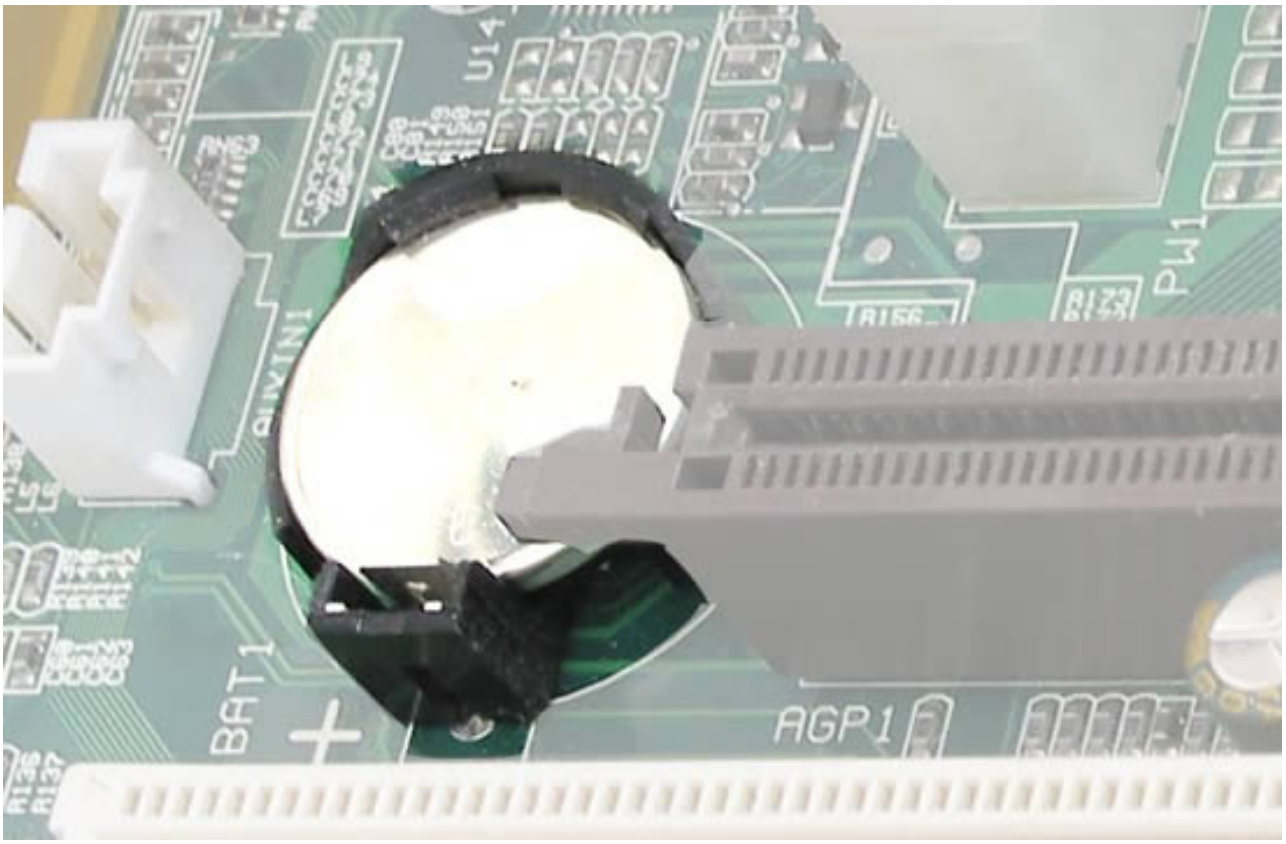


Figure 242. The motherboard battery, essential to maintain the CMOS storage data.

Modern batteries last for many years, but you may find that the battery has to be replaced one day. You're sure to notice when this happens. The Setup program will ask you to enter the date, etc. – every time you start the PC!

28. The BIOS programs

I have now described how the BIOS programs are stored in the motherboard's ROM circuits, and how they look after starting up the PC and allow you to "program" various hardware devices via the Setup program. But that's not the whole truth about the BIOS. The BIOS is not only used during startup. And there is BIOS in other places, in addition to the motherboard. Let me conclude this examination of the PC architecture by describing the BIOS.

Using the BIOS

In the computer's infancy, there were no operating systems. Applications (user programs) were written so that they controlled the PC's hardware directly. Every program, for example, had to be coded to fetch characters from the keyboard and to send characters to the screen. In order to make programming easier, all the program routines that created interfaces to hardware devices were gathered into a library. This collection became the *Basic Input Output System* – the BIOS.

Today the BIOS is still a collection of program fragments which establish connections to certain hardware devices. The BIOS is still available to programmers, but the Windows operating system prefers to use its own drivers to communicate with hardware. Thus Windows has two ways of accessing each hardware device:

- Via the driver programs.
- Via the BIOS programs

BIOS really has the same function as the drivers; it's just an older system. Whereas the BIOS routines are placed in the actual hardware, the drivers are placed within the operating system. There is a BIOS routine, for example, which enables the PC to read input from the keyboard. The BIOS is located, as we

have seen, in the ROM circuits on the motherboard. But it is also stored on expansion cards in so-called *adapter ROM*.

The BIOS we find on the motherboard handles all the motherboard's own devices. External adapters (e. g. video cards) also need BIOS, and this code is, of course, placed in ROM circuits in the device. During startup, all the BIOS is merged and allocated RAM addresses, so that these program fragments are available to the operating system.

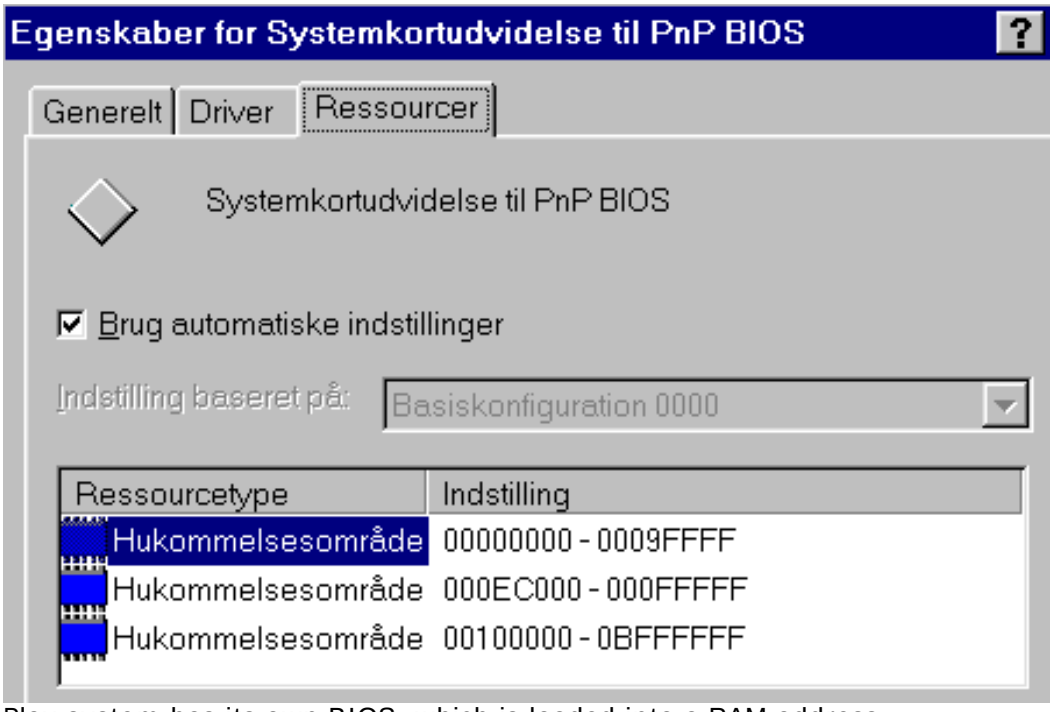


Figure 243. The Plug and Play system has its own BIOS, which is loaded into a RAM address.

Thus both the drivers and the BIOS tailor the hardware devices for the operating system, so that everything works together:

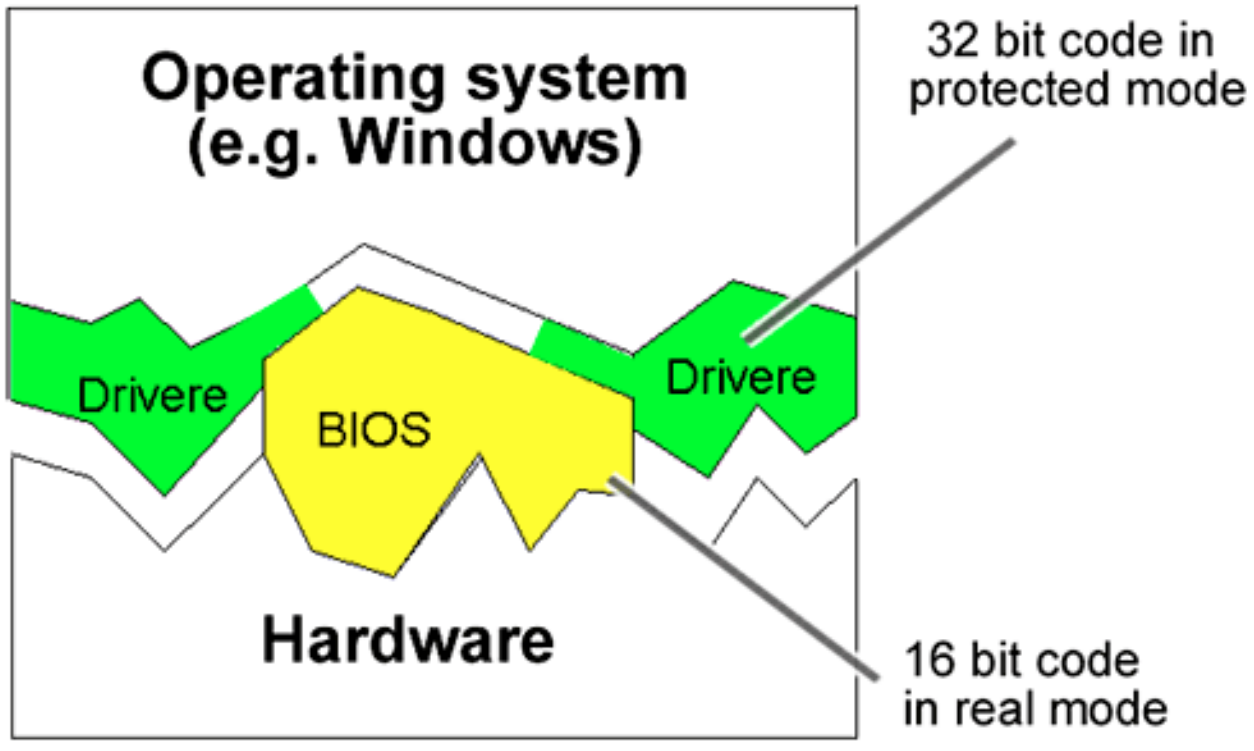


Figure 244. The BIOS and drivers are software written for specific hardware which smooth out the

interfaces between the hardware and the operating system.

16 bit BIOS code

The BIOS routines are part of the very original PC architecture. They were therefore written in 16-bit code which suited the original 8088/86 processors, and which was executed in the slow *real mode*. The BIOS is the PC's oldest and most primitive program layer.

Since modern CPU's, and Windows, are a 32-bit environment, it is not appropriate to use the BIOS routines for anything beyond what is absolutely essential. During normal daily operations, Windows therefore uses its own 32-bit drivers to exchange data with hardware devices. These drivers work much faster, and can be constantly adapted to new hardware.

However, during startup, the BIOS is still important. At that time, the operating system hasn't yet been loaded, so the startup programs draw on the BIOS. And if you work with a 16-bit operating system (such as DOS), it will be using the BIOS to a large extent.

Advantages of 32-bit drivers

Windows' drivers can write directly to all kinds of hardware – bypassing the BIOS routines.

One example is the COM ports. If you use the BIOS routines attached to these, you cannot transmit any more than 9600 bits per second (*baud*) via a modem. But using Windows' own drivers, much greater amounts of data can be transferred over the COM ports.

Thus in this situation the original BIOS routines are basically unusable. One could easily imagine that the BIOS could be upgraded to be more efficient. But that would presumably break the compatibility. So instead we continue to live with obsolete BIOS, but luckily it has little practical significance.

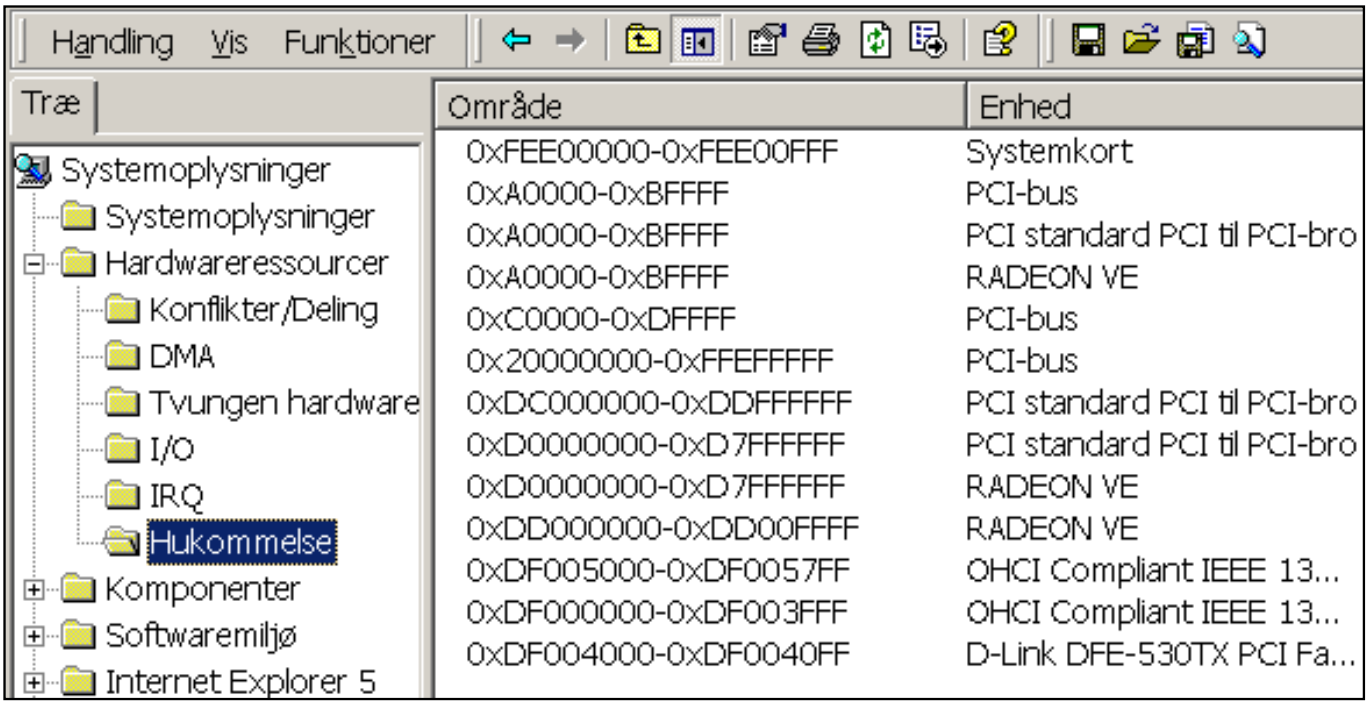


Figure 245. Windows' list of RAM addresses for hardware-specific BIOS.

BIOS upgrades

The BIOS programs can be upgraded. Modern motherboards (as mentioned before) have their BIOS stored in flash ROM, which can be upgraded. You can obtain new BIOS software from your supplier or on the Internet.

To perform an upgrade you first download the new BIOS from the motherboard manufacturer's web site and save it on a diskette. You then "run" this on the PC in DOS mode.

However, you shouldn't start to experiment with BIOS upgrades unless you are sure that you have understood the instructions 100%, and that you have obtained the correct version of the BIOS. Otherwise it can cause big problems.

Instead of BIOS

Intel plans to replace the (very) old BIOS system with a new set of programs. The new system is called EFI (*Extensible Firmware Interface*) and is in it self a complete little operating system. It has a graphical user interface. Where the old BIOS is written in the Assempler language, the new EFI is written in C, making it more easy accessable.

29. The end

We have now worked our way fairly thoroughly through the PC's architecture. I hope you have really learned something from the examination. The ongoing work is up to you. I recommend you take your time to analyse your own PC. You can learn a lot from that.

Getting help

There are a number of tools you can make use of when you start investigating:

- The motherboard manual.
- The messages from the startup program and the Setup program.
- Windows utilities such as System Information and Device Manager.
- Test programs.

It can also be very practical to have a startup disk with DOS on it. If you load this, you will get access to "pure DOS", which you can learn a lot from working with.

Search on the Net

I have already mentioned that the Internet is a good source of knowledge on IT subjects. There are an incredible number of English language sites, which specialise in describing, analysing and commenting on various hardware items such as motherboards, processors, etc.

I won't promote any sites in particular. When I need information myself on a particular topic, I use the Google search engine, and I can find a handful of relevant sites in a matter of seconds. In my experience you can get a lot out of doing this.

If I was to mention a specific site, it would have to be *www.geek.com*. It is run by Rob Hughes, and every month he sends out a little newsletter, which I can warmly recommend. Rob talks about all the new processors, and developments in the area of RAM, chipsets, etc, and also provides links to other sites. It is very good, and free.

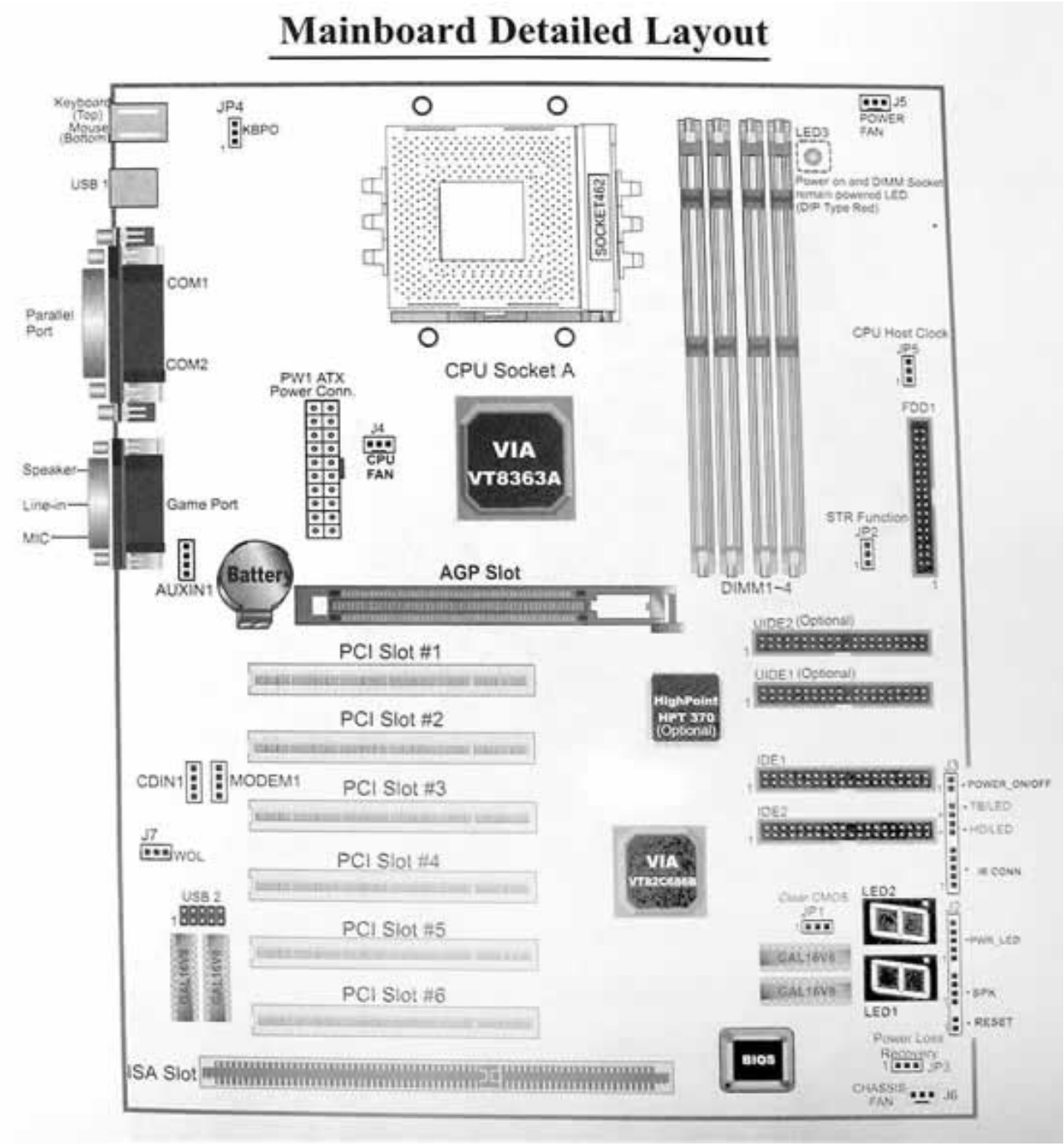


Figure 246. Your motherboard manual is worth studying.

30. Glossary

Fairly early on in the guide I started to explain a number of terms. I will now conclude with even more of the same. Most of these expressions have been used in the guide; here is a brief explanation of them.

AMR. Audio Modem Riser. A standard for small expansion cards which are mounted on the motherboard and which provide connectors for the motherboard's built-in sound and modem functions. Only used in very cheap PC configurations.

BIOS. Basic Input Output System. Program code which provides simple control over a hardware device.

Bus Master. A controller property, which allows a PCI device to take control over the PCI bus, so that data can be exchanged directly with RAM. **CMOS.** Complementary Metal-Oxide Semiconductor. Really a name for the process technology used, among other things, for CPU's. There is a CMOS storage chip on the motherboard which contains a database of information about the motherboard and the other hardware devices. **CSA**. Communication Streaming Architecture. An Intel design, where the Gigabit LAN controller plugs directly into the north bridge.

DMA. Direct Memory Access. A system which allows an ISA device to move data directly to and from RAM.

DOS. Disk Operating System. A 16-bit operating system from the time before Windows. Can still be installed on any PC.

DRAM. Dynamic RAM. Storage cells which are used for standard RAM modules. Requires a constant refresh signal to remember the data.

ECC. Error Correcting Code. A type of RAM which can correct internal errors which might occur during program execution. Is used for DRAM in servers and in the CPU's internal cache.

ECP. Enhanced Capabilities Port. One of the parallel port's modes. Requires the use of a DMA channel.

EIDE. Enhanced IDE interface for hard disks. Also called PATA for Parallel ATA.

EPP. Enhanced Parallel Port. One of the parallel port's faster modes. Also called "*Bi-directional*", since data can run both ways through the port.

ESCD. Extended System Configuration Data. A list of the installed PCI devices. Included in the CMOS storage.

File system. A system for organising data at the sector level on, for example, a hard disk. E.g. FAT32, NTFS.

IDE. Integrated Drive Electronics. Hard disk interface.

IrDA. Infrared Data Association. Standard for wireless data transfer over short distances using infrared light (like your TV remote control).

Local bus. A bus (e.g. the system bus) which runs synchronously with the clock frequency in a connected device.

LPC. Low Pin Count. An Intel standard which is used, among other things, for non ISA-based Super I/O chips, designed with relatively few pins.

Mode. A state which an interface or a system can work in.

Multitasking. When a system can work on several jobs at the same time.

PCI. Peripheral Component Interconnect. The traditional motherboard's standard I/O bus.

PCI Express. New standard replacing PCI and AGP.

PIO. Programmed I/O. Protocol for IDE/EIDE interfaces.

POST. Power On Self Test. The part of the BIOS programs which tests the PC during startup.

Protected mode. The 32-bit program state which replaced real mode.

Real mode. A 16-bit state which programs were limited to working in in the first CPU's.

Registers. Tiny, lightning fast RAM stores, built deep into the CPU. **RAID**. Redundant Arrays of Inexpensive Disks.

SATA. Serial ATA.

SCSI. Small Computer System Interface. An ”intelligent” expansion bus.

SRAM. Static RAM. A type of RAM which (in contrast to DRAM) can maintain its data without a refresh signal. SRAM is faster and more expensive than DRAM. SRAM is used, for example, for processor caches.

Thank you very much for your attention !

Yours sincerely, Michael Karbo. www.karbosguide.com

- Next chapter.
- Previous chapter.
