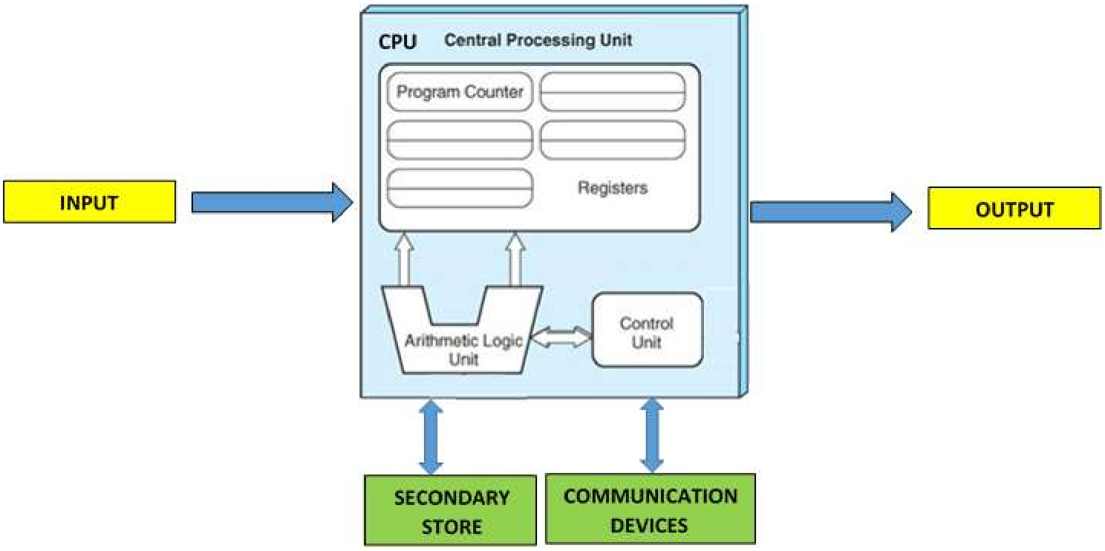
# GCE Computer Science @ SRS



# Computer Architecture

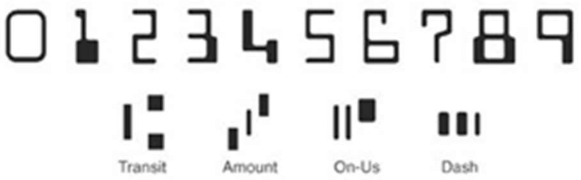
## Computer Systems

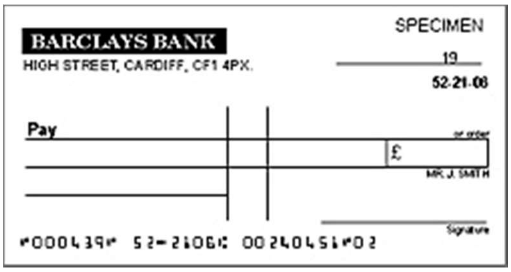
## 2020-21

## MICR Reader

**Magnetic Ink Character Recognition** (**MICR**) is a technology that allows details from **bank cheques** to be read into a computer **quickly** and **accurately**.

The **cheque number** and **bank account** number are printed at the bottom of each bank cheque in **special magnetic ink** using a **special font.** These numbers can be detected by an **MICR reader.**




BARCLAYS BANK
HIGH STREET, CARDIFF, CF1 4PX.

SPECIMEN

19

52-21-08

Pay

or order

£

MR J. SMITH

Signature

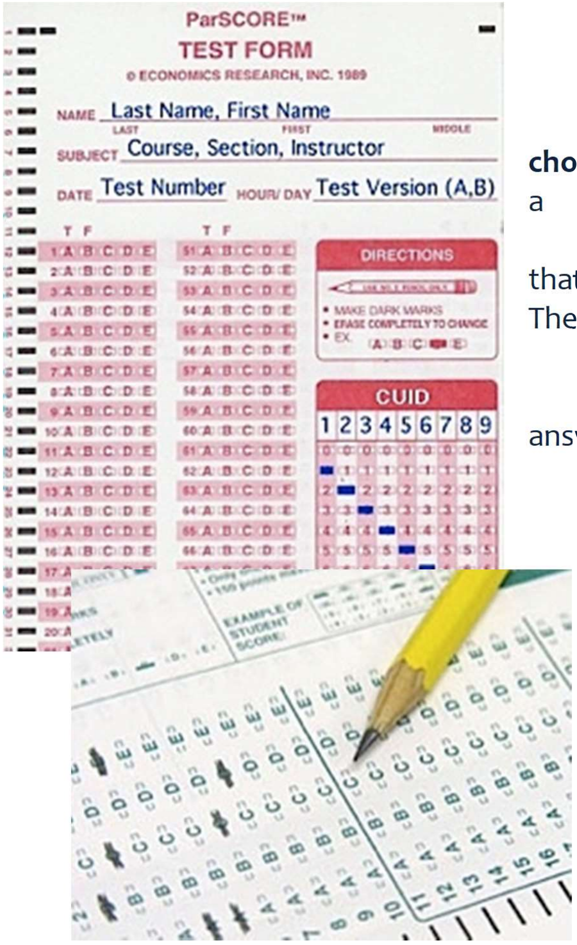
⑈000439⑈ 52⑉21081⑆ 00240451⑈02

## OMR Scanner

**Optical Mark Recognition** (**OMR**) is a technology that allows the data from a **multiple-choice** type form to be read **quickly** and **accurately** into a computer.

**Special OMR forms** are used which have spaces that can be **coloured in** (usually using a pencil). These **marks** can then be **detected** by an **OMR scanner.**

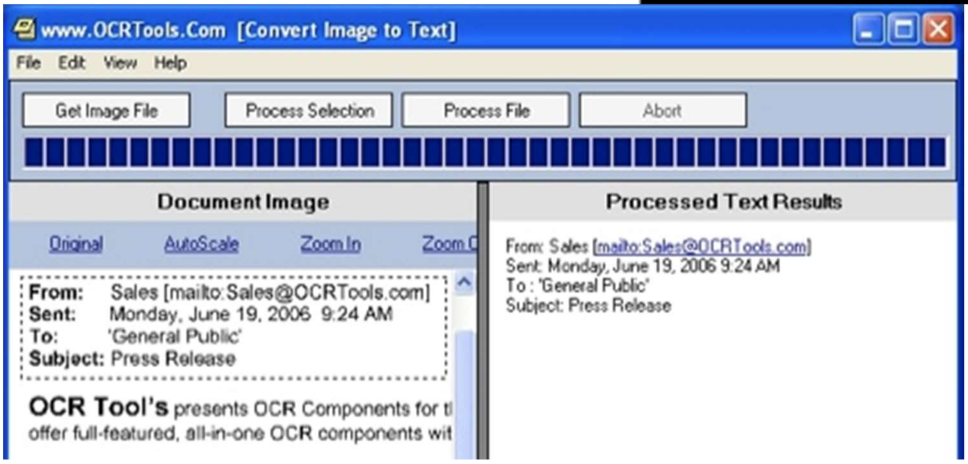
Common uses of OMR are **multiple-choice exam** answer sheets and **lottery number** forms.


ParSCORE™
TEST FORM
© ECONOMICS RESEARCH, INC. 1989
NAME Last Name, First Name
LAST FIRST MIDDLE
SUBJECT Course, Section, Instructor
DATE Test Number HOUR/ DAY Test Version (A,B)
DIRECTIONS
• MAKE DARK MARKS
• ERASE COMPLETELY TO CHANGE
• EX. A B C D E
CUID
1 2 3 4 5 6 7 8 9
EXAMPLE OF STUDENT SCORE

# OCR Scanner

**Optical Character Recognition** (**OCR**) is a software technology that can **convert images of text into an actual text file** that can then be edited, e.g. using word-processing software). The result is just as if the text had been typed in by hand.

OCR is typically used after a page of a book has been **scanned**. The scanned **image** of the page is then **analysed** by the **OCR software** which looks for recognisable **letter shapes** and generates a matching text file.
Advanced OCR software can recognise normal **handwriting** as well as printed text - this is usually called **handwriting recognition**.



From: Sales [mailto:Sales@OCRTools.com]
Sent: Monday, June 19, 2006 9:24 AM
To: 'General Public'
Subject: Press Release

OCR Tool's presents OCR Components for t
offer full-featured, all-in-one OCR components wit

From: Sales [mailto:Sales@OCRTools.com]
Sent: Monday, June 19, 2006 9:24 AM
To: 'General Public'
Subject: Press Release

# Barcode Reader / Scanner

A barcode is simply a **numeric code** represented as a series of **lines**.

These lines can be read by a **barcode reader/scanner**.

The most

common use of barcode readers is at **Point-of-Sale** (**POS**) in a shop. The **code** for each item to be purchased needs to be entered into the computer. Reading the **barcode** is far **quicker** and more **accurate** than **typing** in each code using a keypad.

Barcode can be found on many other items that have numeric codes which have to be read quickly and accurately - for example ID cards.

## Identify Uses for Input Devices

## A blind student which uses the computer to complete his studies at home – What Input & Output Devices do you recommend and why?

Input Devices

Output Devices

## An underground station uses computers to sell  tickets – What Input & Output Devices do you recommend and why?

Input Devices

Output Devices

## A plane pilot to control her plane – What Input & Output Devices do you recommend and why?

Input Devices

Output Devices

## A school canteen cashier – What Input & Output Devices do you recommend and why?

Input Devices

Output Devices

## A lottery agent – What Input & Output Devices do you recommend and why?

Input Devices

Output Devices

## A - Choosing the best printer

A small digital photography firm has based their central offices on a first floor in a central of London area. In this floor there are around 10 desks with computers one for each secretarial staff. The staff job is to contact customers by phone in order to promote the company and subsequently increase their sales.

At the moment there is one large mono laser printer for the entire floor. Once a member of staff decided to print something they have to leave their desks, go to the end of the room and get their printout(s). Sometimes there is a large queue and delays are possible.

The manager has decided to invest in some printer(s)(output devices) but doesn't know which one to go for and why?

You, as an expert, are called in to help. Can you recommend a type of printer(s)?

## B - Choosing the best printer

A classroom teacher delivers lessons and he needs a printer to print support material for his students. Sometimes the printing is taking place whilst he delivers his lesson.

Which printer should the teacher go for?

## C - Choosing the best printer

A doctor's surgery need to remind his patients of their upcoming appointments and therefore needs to send out letter/reminders. The office within the surgery is very crowded with about 5 secretarial staff and they are all have limited desk space.

Which printer should the doctor go for?

## D - Choosing the best printer

A financial advisor is meeting clients at their own homes and after the consultation she needs to print out some drafts of what has been agreed during their appointment. Sometimes an ipad is used instead of a laptop.

Which printer should the financial advicer go for?

## Types of Printers

- Inkjet printers
- Mono Laser printers
- All-in-one printers
- Colour Laser printers
- Wireless printers or Wired
- Dot-Matrix
- Plotters
- 3D-Printers
- Single User of Multi User

Selecting a printer – Points to consider

| **Points to consider** | | |
|---|---|---|
| Wireless or wired | | |
| All in One or single purpose | | |
| Colour Laser | | |
| Single User | | |
| Amount of printing | | |
| Mono Laser | | |
| Plotter | | |
| Multi User - Networking | | |
| Noise or quiet | | |
| Duplex printing | | |
| Speed of printing | | |

| | | |
|---|---|---|
| Inkjet | | |
| Laser | | |
| Size of printing paper - envelope size, A4, A3, A1 etc. | | |
| Quality of printing – Photographs, normal paper, canvas etc. | | |
| Running Cost | | |
| Purchasing cost | | |

**Dot matrix:** A type of impact printer that produces characters and illustrations by striking pins against an ink ribbon to print closely spaced dots in the appropriate shape. Dot-matrix printers are relatively expensive and do not produce high-quality output. However, they can print to multi-page forms (that is, carbon copies), something laser and ink-jet printers cannot do.

**Ink-jet:** A type of printer that works by spraying ionized ink at a sheet of paper. Magnetized plates in the ink's path direct the ink onto the paper in the desired shapes. Ink-jet printers are capable of producing high quality print approaching that produced by laser printers. A typical ink-jet printer provides a resolution of 1200 dots per inch, although some newer models offer higher resolutions.  A typical ink-jet uses 1 black and 3 toner cartridges.

**Laser:** A type of printer that utilizes a laser beam to produce an image on a drum. The light of the laser alters the electrical charge on the drum wherever it hits. The drum is then rolled through a reservoir of toner, which is picked up by the charged portions of the drum. Finally, the toner is transferred to the paper through a combination of heat and pressure. This is also the way copy machines/photocopiers work. This is one of the fastest printers that is very economical when you produce many copies and provides very good quality output. It uses one black and 3 colour toners.

**Plotters:** A printer that interprets commands from a computer to make line drawings on paper with one or more automated pens. Unlike a regular printer, the plotter can draw continuous point-to-point lines.  Plotters were the first type of printer that could print with colour. As a rule, plotters are much more expensive than printers.  A plotter may use multiple pens and pencils, which can be easily be changed out in order to create drawings of a different colour or drawings that contain more than one colour. A plotter is preferred over a printer in many commercial applications, including engineering, because it is far more exact.

It is because of this they are most frequently used for CAE (computer-aided engineering) applications, such as CAD (computer-aided design) and CAM (computer-aided manufacturing).

**Wireless Printers**: A printer is one kind of computer peripheral, a device that interacts with a computer and receives information from it, in this case, to create a hard copy of documents that exist in electronic form on the computer’s drive or a location accessed by the computer.

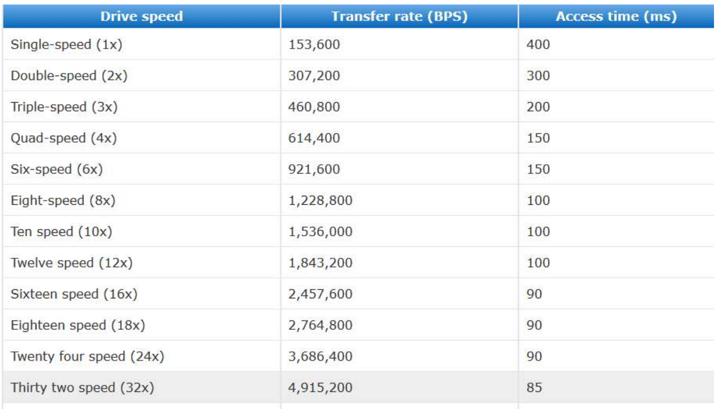
Like other peripherals, printers can be connected to the computer in various ways. Connection methods have traditionally included **printer cables**, **USB cables**, **Ethernet cables**, and more recently, **wireless connections**. Printers capable of connecting wirelessly are referred to as wireless printers.

**3D- Printers:** 3D printing or additive manufacturing is a process of making three dimensional solid objects from a digital file. The creation of a 3D printed object is achieved using **additive** processes. In an additive process an object is created by laying down successive layers of material until the entire object is created.

## Storage Devices

**Magnetic tape:** A magnetically thin coated piece plastic wrapped around wheels capable of storing data. Tape is much less expensive than other storage mediums but commonly a much slower solution that is commonly used for backup.

Today, tape has mostly been abandoned for faster and more reliable solutions like disc drives, hard drives, and flash drives.



| Drive speed | Transfer rate (BPS) | Access time (ms) |
|---|---|---|
| Single-speed (1x) | 153,600 | 400 |
| Double-speed (2x) | 307,200 | 300 |
| Triple-speed (3x) | 460,800 | 200 |
| Quad-speed (4x) | 614,400 | 150 |
| Six-speed (6x) | 921,600 | 150 |
| Eight-speed (8x) | 1,228,800 | 100 |
| Ten speed (10x) | 1,536,000 | 100 |
| Twelve speed (12x) | 1,843,200 | 100 |
| Sixteen speed (16x) | 2,457,600 | 90 |
| Eighteen speed (18x) | 2,764,800 | 90 |
| Twenty four speed (24x) | 3,686,400 | 90 |
| Thirty two speed (32x) | 4,915,200 | 85 |

**CD-ROM:** Short for Compact Disc-Read Only Memory, CD-ROM drives or optical drives are CD players inside computers that can have speeds in the range from 1x and beyond, and have the capability of playing audio CDs and computer data CDs. Below is a picture of the front and back of a standard CD-ROM drive. It is found as CD-R which means Read only. The spinning speed of the drive is affeccting the bits per second that can be read at any time.

There are some DC's that are known as cd writer, CD-WO (Write once), WORM (Write Once Read Many) drive. CD-R is short for CD-Recordable and is a writable disc and drive that is capable of having information written to the disc once and then having that disc read many times after that.

**Recordable DVD disks**

Alternatively referred to as a DVD writer, recordable DVD drives are disc drives capable of creating DVD discs. Unfortunately, unlike recordable CD drives, there are many different competing standards for creating DVD discs. For example, DVD-R, DVD-RW, DVD+R, DVD+RW, DVD+R DL (DVD+R9), and DVD-RAM are all different competing standards. Below is a brief explanation of each of these standards and related links to each of these standards.

**USB Memory Stick**

Alternatively referred to as a USB flash drive, data stick, pen drive, keychain drive and thumb drive, a jump drive is a portable drive that is often the size of your thumb that connects to the com puter USB port. Today, flash drives are available in sizes such as 256MB, 512MB, 1GB, 5GB, and 16GB and are an easy way to transfer and store information.

In the picture to the right, is an example of the SanDisk Cruzer Micro 16GB flash drive and a good example of what many flash drives look like. As can be seen in this picture, the drive has a small casing that stores the flash memory connected to a USB connection that is plugged into the USB port on your computer.

**Magnetic hard disk:**

- Currently the most common type of secondary storage with very large capacities and good access speed.
- The magnetic hard disk :
  - is reliable;
  - has a high capacity at low cost;
  - can be an internal device or external portable device for backup or transfer of large amounts of data;
  - stores the operating system, user data and programs.

**Optical disk:**

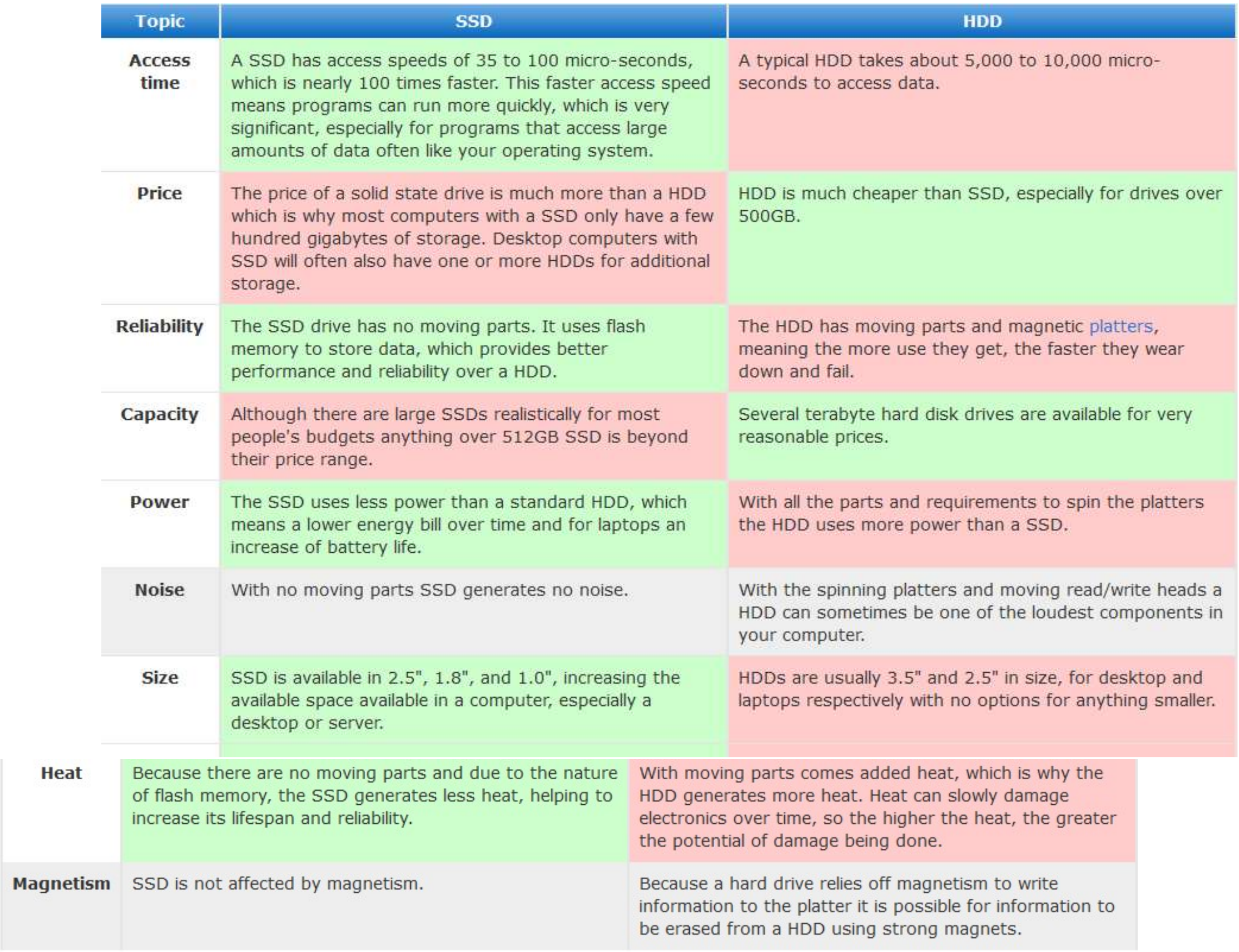
- **CD** (compact disk) or **DVD** (digital versatile disk)
- can be read only or read / write
- CD/DVD ROM used to distribute programs
- CD/DVD R/W used to store, transfer or backup data and program
  - high capacity at low cost
  - small and easy to distribute
  - robust and can be used many times

- CD 700MB, DVD 4.7GB typical capacity
- slower than other media such as hard disk and flash ROM

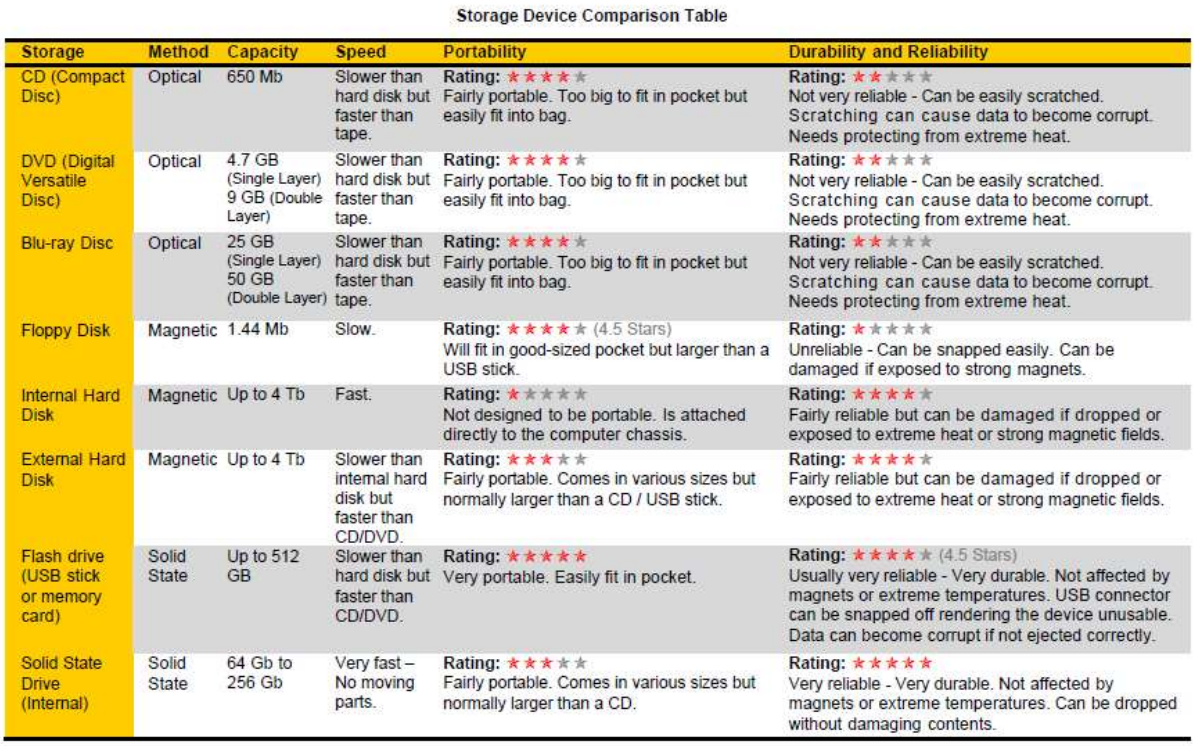
**Flash (solid state SSD) drive:**

- Electronically alterable Read Only Memory
  - used in portable devices such as cameras, MP3 players, tablet computers and mobile phones
  - used with a USB interface as a backup / transfer medium for personal files
  - fast access times
  - reliable with no moving parts to go wrong
  - low power use
  - small, light, robust and highly portable
  - inexpensive at relatively low capacities but expensive at higher capacities.

### Difference between a Solid State Drive SSD and Hard Disk Drive HDD



| Topic | SSD | HDD |
|---|---|---|
| Access time | A SSD has access speeds of 35 to 100 micro-seconds, which is nearly 100 times faster. This faster access speed means programs can run more quickly, which is very significant, especially for programs that access large amounts of data often like your operating system. | A typical HDD takes about 5,000 to 10,000 micro-seconds to access data. |
| Price | The price of a solid state drive is much more than a HDD which is why most computers with a SSD only have a few hundred gigabytes of storage. Desktop computers with SSD will often also have one or more HDDs for additional storage. | HDD is much cheaper than SSD, especially for drives over 500GB. |
| Reliability | The SSD drive has no moving parts. It uses flash memory to store data, which provides better performance and reliability over a HDD. | The HDD has moving parts and magnetic platters, meaning the more use they get, the faster they wear down and fail. |
| Capacity | Although there are large SSDs realistically for most people's budgets anything over 512GB SSD is beyond their price range. | Several terabyte hard disk drives are available for very reasonable prices. |
| Power | The SSD uses less power than a standard HDD, which means a lower energy bill over time and for laptops an increase of battery life. | With all the parts and requirements to spin the platters the HDD uses more power than a SSD. |
| Noise | With no moving parts SSD generates no noise. | With the spinning platters and moving read/write heads a HDD can sometimes be one of the loudest components in your computer. |
| Size | SSD is available in 2.5", 1.8", and 1.0", increasing the available space available in a computer, especially a desktop or server. | HDDs are usually 3.5" and 2.5" in size, for desktop and laptops respectively with no options for anything smaller. |
| Heat | Because there are no moving parts and due to the nature of flash memory, the SSD generates less heat, helping to increase its lifespan and reliability. | With moving parts comes added heat, which is why the HDD generates more heat. Heat can slowly damage electronics over time, so the higher the heat, the greater the potential of damage being done. |
| Magnetism | SSD is not affected by magnetism. | Because a hard drive relies off magnetism to write information to the platter it is possible for information to be erased from a HDD using strong magnets. |



Storage Device Comparison Table

| Storage | Method | Capacity | Speed | Portability | Durability and Reliability |
|---|---|---|---|---|---|
| CD (Compact Disc) | Optical | 650 Mb | Slower than hard disk but faster than tape. | **Rating:** ★★★★☆<br>Fairly portable. Too big to fit in pocket but easily fit into bag. | **Rating:** ★★☆☆☆<br>Not very reliable - Can be easily scratched. Scratching can cause data to become corrupt. Needs protecting from extreme heat. |
| DVD (Digital Versatile Disc) | Optical | 4.7 GB (Single Layer)<br>9 GB (Double Layer) | Slower than hard disk but faster than tape. | **Rating:** ★★★★☆<br>Fairly portable. Too big to fit in pocket but easily fit into bag. | **Rating:** ★★☆☆☆<br>Not very reliable - Can be easily scratched. Scratching can cause data to become corrupt. Needs protecting from extreme heat. |
| Blu-ray Disc | Optical | 25 GB (Single Layer)<br>50 GB (Double Layer) | Slower than hard disk but faster than tape. | **Rating:** ★★★★☆<br>Fairly portable. Too big to fit in pocket but easily fit into bag. | **Rating:** ★★☆☆☆<br>Not very reliable - Can be easily scratched. Scratching can cause data to become corrupt. Needs protecting from extreme heat. |
| Floppy Disk | Magnetic | 1.44 Mb | Slow. | **Rating:** ★★★★☆ (4.5 Stars)<br>Will fit in good-sized pocket but larger than a USB stick. | **Rating:** ★☆☆☆☆<br>Unreliable - Can be snapped easily. Can be damaged if exposed to strong magnets. |
| Internal Hard Disk | Magnetic | Up to 4 Tb | Fast. | **Rating:** ★☆☆☆☆<br>Not designed to be portable. Is attached directly to the computer chassis. | **Rating:** ★★★★☆<br>Fairly reliable but can be damaged if dropped or exposed to extreme heat or strong magnetic fields. |
| External Hard Disk | Magnetic | Up to 4 Tb | Slower than internal hard disk but faster than CD/DVD. | **Rating:** ★★★☆☆<br>Fairly portable. Comes in various sizes but normally larger than a CD / USB stick. | **Rating:** ★★★★☆<br>Fairly reliable but can be damaged if dropped or exposed to extreme heat or strong magnetic fields. |
| Flash drive (USB stick or memory card) | Solid State | Up to 512 GB | Slower than hard disk but faster than CD/DVD. | **Rating:** ★★★★★<br>Very portable. Easily fit in pocket. | **Rating:** ★★★★☆ (4.5 Stars)<br>Usually very reliable - Very durable. Not affected by magnets or extreme temperatures. USB connector can be snapped off rendering the device unusable. Data can become corrupt if not ejected correctly. |
| Solid State Drive (Internal) | Solid State | 64 Gb to 256 Gb | Very fast – No moving parts. | **Rating:** ★★★☆☆<br>Fairly portable. Comes in various sizes but normally larger than a CD. | **Rating:** ★★★★★<br>Very reliable - Very durable. Not affected by magnets or extreme temperatures. Can be dropped without damaging contents. |

| **Magnetic Storage** | **Optical Storage** |
|---|---|
| a) Stores data in magnetic form. | a) Stores data optically & used laser to read/write. |
| b) It is affected by magnetic field. | b) It is not affected by magnetic field. |
| c) It has high storage capacity. | c) It has less storage than magnetic hard disk. |
| e) It doesn't use laser to read/write data. | e) Data accessing is high as compared to magnetic. |
| In your workbook write some examples of **Magnetic** storage devices: | Write below some examples of **Optical** storage devices: |

**Solid State Storage - 'No moving Parts' (SSD)**

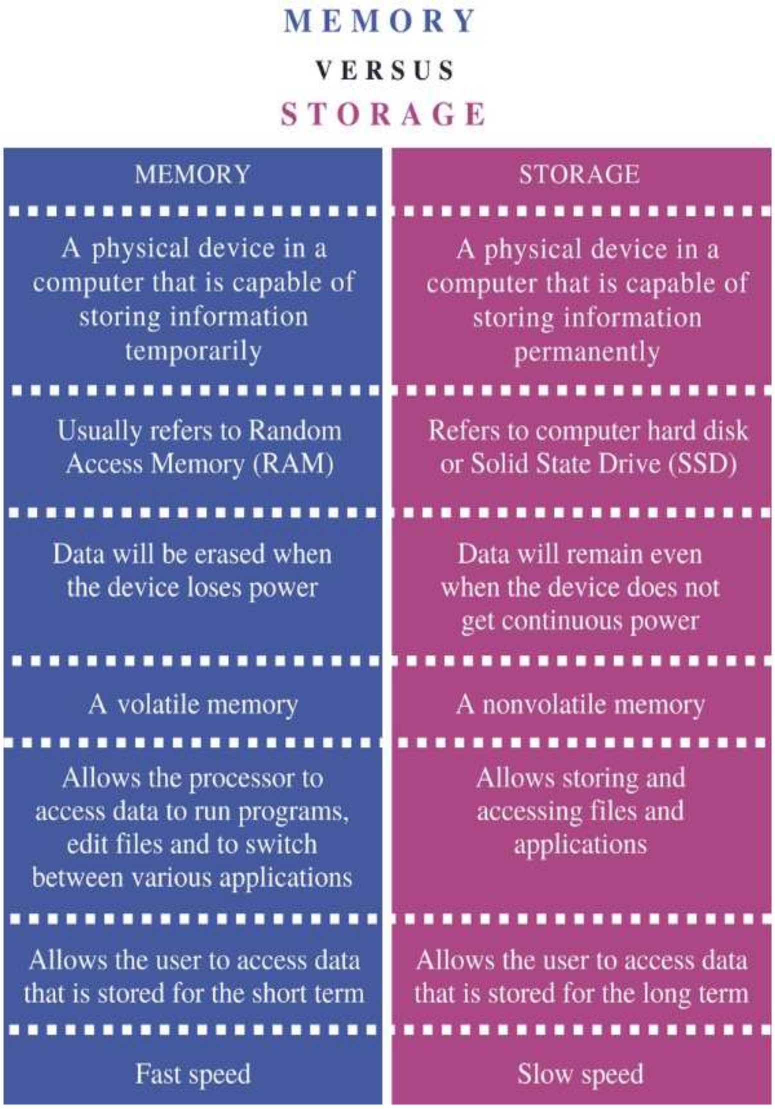
- Start-up faster due to no spin-up and they are faster than magnetic hard drives
- They last longer and some are waterproof
- They are more robust as they haven't got any mechanical parts
- All data stored can be scanned quickly for security purposes

Write some examples of **Solid State Storage**:

## MEMORY & STORAGE

There is a subtle distinction between memory and storage. We noted that 'storage' devices suggest a place where we store data and applications that are not needed immediately by the CPU. Such devices would include a hard disk, a floppy disk and a CD R/W, for example.  Another name for such devices is 'secondary storage devices'.

Memory, however, is a place where data and applications needed immediately by the CPU are put. We can refer to memory as the **'Immediate Access Store'** or **'Primary Memory'**.  We can even refer to it simply as the RAM or Random Access Memory'.



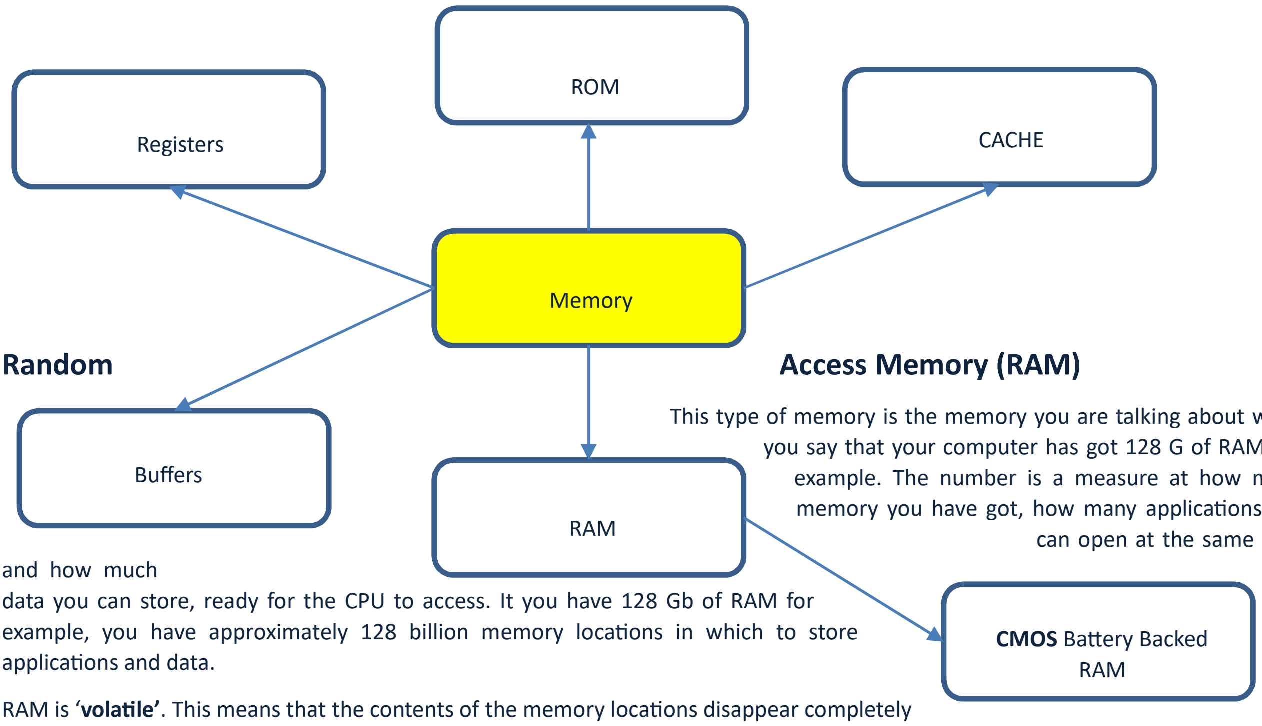
MEMORY

VERSUS

STORAGE

| MEMORY | STORAGE |
|---|---|
| A physical device in a computer that is capable of storing information temporarily | A physical device in a computer that is capable of storing information permanently |
| Usually refers to Random Access Memory (RAM) | Refers to computer hard disk or Solid State Drive (SSD) |
| Data will be erased when the device loses power | Data will remain even when the device does not get continuous power |
| A volatile memory | A nonvolatile memory |
| Allows the processor to access data to run programs, edit files and to switch between various applications | Allows storing and accessing files and applications |
| Allows the user to access data that is stored for the short term | Allows the user to access data that is stored for the long term |
| Fast speed | Slow speed |

There are many different types of memory that the CPU is using.



## Random Access Memory (RAM)

This type of memory is the memory you are talking about when you say that your computer has got 128 G of RAM, for example. The number is a measure at how much memory you have got, how many applications you can open at the same time and how much data you can store, ready for the CPU to access. It you have 128 Gb of RAM for example, you have approximately 128 billion memory locations in which to store applications and data.

RAM is '**volatile**'. This means that the contents of the memory locations disappear completely once power is removed. The contents only remain whilst the computer is switched on. You may have noticed that sometimes your computer stops working as it should. The contents of the RAM may have become '**corrupted**', or mixed up. By re-booting your computer, you are clearing out from RAM all of the applications (including the operating system) and data, and then reloading them again into RAM. This very often clears any problem with the computer.

**Read Only Memory (ROM)**

Another type of memory is **R**ead **O**nly **M**emory, or ROM. This type of memory holds a special program that starts running when the computer is turned on. It holds a part of a program called the BIOS (**B**asic **I**nput **O**utput **S**ystem). This program does two things.

1. It checks that the computer hardware is present and correctly working.

2. It runs a routine that looks for another special program called the **bootstrap program.** This is usually held in a special place on the hard drive and then loads it into RAM and run it. Starting up the computer from a power-off situation to where the operating system has been loaded up is known as '**booting up'** the computer.

ROM is "**non-volatile**". That means that the contents of ROM are NOT lost when you turn the power off unlike RAM. The actual program In ROM is put there by the manufacturers of motherboards, for example, around the time the RROM chip is placed in a motherboard.

You may have wondered why there appears to be such a roundabout system of loading up your operating system, why you have to run some Instructions in ROM that looks for a program called the 'bootstrap' on the hard disk that then loads up the operating system! Why not just put the operating system in ROM and let it load up straight away when you boot up the computer? The answer is **'flexibility'**.

You should understand that you, the user, can't normally change the contents of ROM. If you wanted to change operating systems, for example, from Windows 8 to Windows 10, and the operating system was in ROM, you would be stuck! However, by putting the Operating System (OS) on the hard disk, you can upgrade it anytime you want and don't need to change the program in ROM.

|  | RAM | ROM |
|---|---|---|
| **Definition:** | Random Access Memory or RAM is a form of data storage that can be accessed randomly at any time, in any order and from any physical location, allowing quick access and manipulation. | Read-only memory or ROM is also a form of data storage that cannot be easily altered or reprogrammed. Stores instructions that are not necessary for re-booting up to make the computer operate when it is switched off. They are hardwired. |
| **Stands for:** | Random Access Memory | Read Only Memory |
| **Use:** | RAM allows the computer to read data quickly to run applications. It allows reading and writing. | ROM stores the program required to initially boot the computer. It only allows reading. |
| **Volatility:** | RAM is **volatile** i.e. its contents are lost when the device is powered off. | It is **non-volatile** i.e. its contents are retained even when the device is powered off. |
| **Types:** | The two main types of RAM are static RAM and dynamic RAM. | The types of ROM include PROM, EPROM and EEPROM. |

### Registers

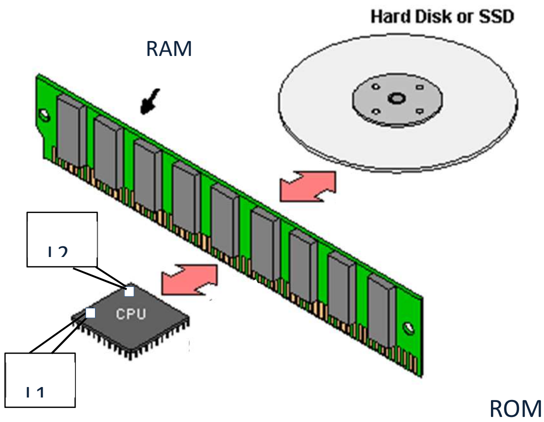
These are part of the design of the CPU. They are temporary memory circuits and are very fast because they have to as they are constantly being accessed by the CPU. Examples of registers are the **Accumulator**, the **Status Register** and the **Program Counter**. There are others. You will learn about registers later on because they are an Integral part of the CPU design but you do not need to know all of the details until we cover the operation of the CPU.

### Cache

Another type of memory is known as the cache and it Is provided in computer systems to speed up processing. Like all memory, it Is measured in bytes (or Kbytes. Mbytes, Gbytes etc.).

Programs are made up of instructions. Instructions are fetched from memory using the fetch-decode-execute cycle The data that instructions need Is also fetched from memory and some data might need to be fetched over and over again for example, a constant that Is held in memory and used lots of times in lots of calculations. Fetching data from the Immediate Access store takes time. Fetching the same data after time is a waste of time!

Processing can be speeded up by storing constantly-needed data in some fast-access memory. This will reduce the **'fetch'** time. This fast-access memory Is called **'cache'**. It is much faster than RAM but not as fast as registers). You only get limited amounts of It in a computer system because It is very expensive to make. One question that should always be asked when buying a new computer is "How much cache has it got"? It is usual always worth getting more cache for a computer because it speeds up processing - at a price!



### Battery-backed RAM (known as CMOS)

We have already said that RAM is ‘volatile’ – when you turn the power off it loses its contents. We said that ROM is non-volatile and a user can’t change the contents of ROM. We also know that ROM is used to store part of the BIOS. The rest of the BIOS is stored in battery –backed RAM. This is RAM but the computer is ensuring the power is never removed from it by connecting a battery to it. This takes over when the power is turned off.

By putting part of the BIOS in battery operated RAM, you ae giving the user the ability to store their own settings, which are used when the computer boots up. For example, when a computer powers up, the BIOS will look for the bootstrap program. Now it can look for it on a CD-ROM or a DVD and then try the hard disk and if it doesn't find it can look on the network via the network card. This might not seem that important but the user can select the order of the devices that the computer need to look at first to find the bootstrap program. This will make the boot up process faster as it can avoid wasting time going through all the other devices in order to find it.

### Buffers

A buffer is an area of RAM that has been reserved for one purpose - to aid the transfer of data between different parts of a Computer because those parts work at different speeds. An example of this is the transfer of data between primary memory and secondary storage devices.

Primary memory is part of the CPU and works at very fast speeds compared to secondary storage devices such as floppy drives and hard drives, which are very slow. If you didn't have a buffer, then the transfer would have to take place at the speed of the slowest device and that is inefficient. It prevents the CPU doing other more important things. because it has to take charge of the management of the data transfer. In fact, buffers are used in many places, wherever there is a Speed mismatch between two devices. Other examples of where they are needed include the transfer of data from the keyboard to the CPU and the transfer of data from the CPU to a printer.

### How to compare different types of memory

Memory can be classified in a whole variety of ways. Depending upon your focus.

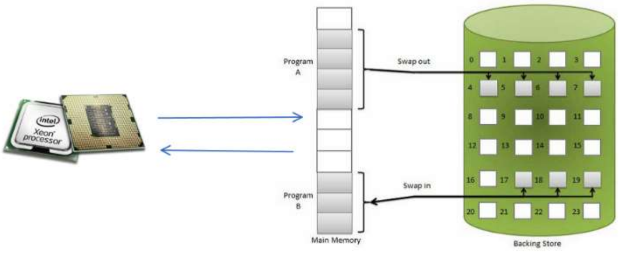
1. **Primary memory and secondary memory** (more commonly referred to as '**secondary storage'**). One way to deal with memory is to split it up into two types: **primary** memory and **secondary** storage. Primary memory is the memory that forms part of the CPU circuitry itself. It's the place where applications and data are held for use by the CPU and where results of calculations are put. Secondary storage devices are devices that are connected to the CPU as peripherals e.g. a floppy disk or a hard disk. They store data and applications not immediately needed by the CPU. Primary memory is also known as RAM. Immediate Access Storage or IAS.

2. **READ devices and READ/WRITE memory devices**. Registers, cache, RAM, floppy disks, hard drives and CD-R/W are READ/WRITE devices. That means that data in these devices can be read but also new data can be written to them. ROM, DVDs and some CDROMS are READ ONLY. Data is burnt (written to once) onto these devices and cannot then be changed. They are sometimes called WORM devices (Write Once Read Many times).

3. **Volatile memory and non-volatile memory.** Another way to split memory types up is to divide them between volatile and non-volatile memory. RAM is volatile. That means that when the power is turned off, you lose the contents. Motherboards have a small amount of battery-backed RAM. When the main power is turned off, this type of RAM will keep its contents (as long as the battery works!) Battery-backed RAM keeps the details of the BIOS password, for example. If you forget the password to the BIOS, you can empty the contents of the battery-backed RAM by removing the battery for a minute. ROM is non-volatile, as is the hard drive. When power is turned off, the contents will not be lost.

4. **Cost, access times and capacity of memory devices.** The price per byte of storing data on the hard drive, for example, is cheap compared to RAM, and cheaper still compared to cache. This is one reason why you can buy huge capacity hard drives but can only usually afford a small amount of cache on your home PC. The physical distance of the CPU from the memory will affect how fast the CPU can carry out instructions. Memory in the form of registers will be right next to the CPU. These will be accessible very quickly compared to other types of memory. Unfortunately, there isn't enough space in the circuitry to put the RAM next to the CPU. RAM will be built on other circuitry and connected to the CPU via wires. Access will be slower than access to the registers or cache! (This is one reason why you need a WAIT interrupt, to tell the CPU to wait while data gets from the RAM to the CPU.)

## Virtual Memory

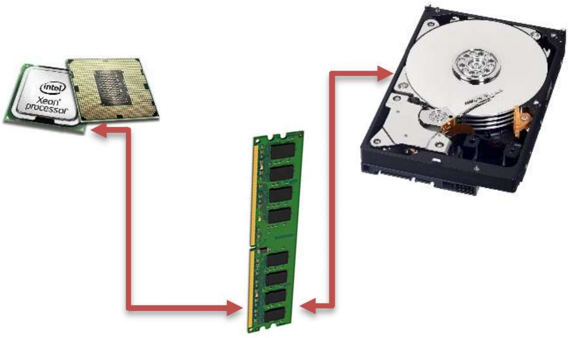


**Virtual memory** is a 'pretend' physical **memory (RAM)** that is written to a file on the hard drive. That file is often called **page file** or **swap file**. It's used by operating systems to simulate (virtually increase) physical RAM by using hard disk space.

Imagine a user who has two files (a WORD file and an EXCEL File) opened on the screen. These files are loaded and stored temporary on the RAM. The user is working on the WORD File but the EXCEL file is not used by the user at all.

The CPU in order to release space so it can store other instructions which are needed on RAM, temporary removes the EXCEL File from its location in RAM into another area inside the HARD disk. The CPU now has more space in RAM to place other current instructions.

When the user decides to work on the EXCEL file the CPU will request it from the HARD DISK. In the meantime, the WORD file from RAM will now be placed in the HARD DISK area and the EXCEL FILE will now take place inside the RAM. The two files have now swapped position.

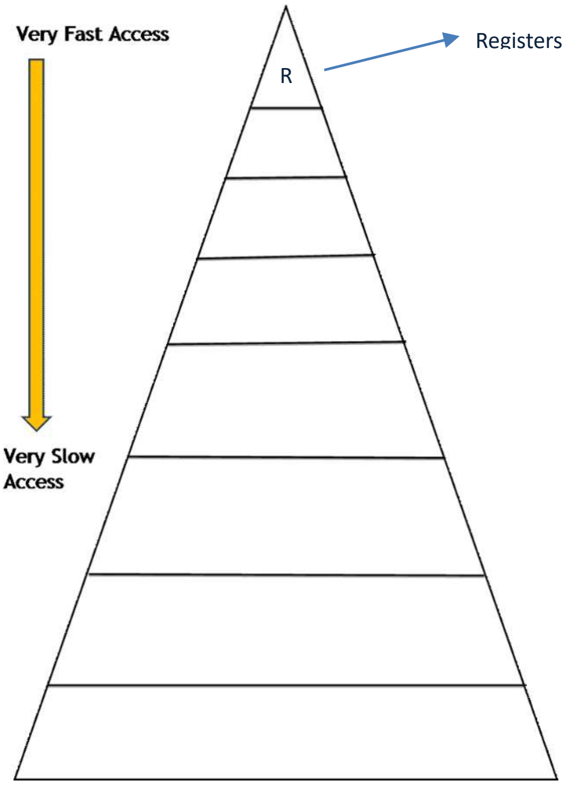


The CPU decides where about these files are stored in the hard disk so it can retrieve them later on.

This process allows the CPU to have more RAM space when it is needed. The Primary memory RAM can double in size by making use of this additional memory known as VIRTUAL MEMORY. Remember Virtual is not physical memory.

**MEMORY EXERCISES**

Place the types of memory on the following pyramid with the ones with fastest access to the cpu at the top of the pyramid.



## Motherboard

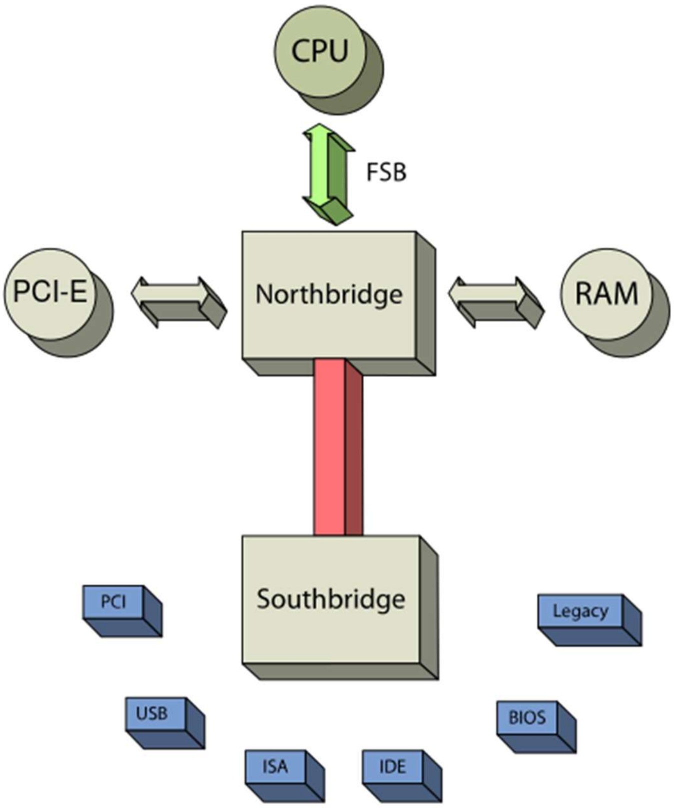
A **motherboard** (sometimes alternatively known as the **main circuit board**, **system board**, **logic board**, or colloquially, a **mobo**) is the main printed circuit board (PCB) found in general purpose microcomputers and other expandable systems. It holds and allows communication between many of the crucial electronic components of a system, such as the central processing unit (CPU) and memory, and provides connectors for other peripherals. A motherboard usually contains significant sub-systems such as the central processor, the chipset's input/output and memory controllers, interface connectors, and other components integrated for general purpose use and applications.

A typical motherboard will have a **chipset** on a motherboard which is made up of two devices:

- The Northbridge and the
- Southbridge

A Northbridge is one of the two chips in the core logic chipset architecture on a PC motherboard, the other being the Southbridge. The Northbridge is connected directly to the CPU via the front-side bus (FSB) and is thus responsible for tasks that require the highest performance in order to support the CPU need for speed. It provides the communication channel to RAM, **Peripheral Component Interconnect Express** (PCI-E). The PCI-E was introduced to offer express speed for the graphics card.

The Northbridge is usually paired with a Southbridge. The Southbridge is also known as the I/O controller as it is responsible for the Input & Output Device with slower speed. Both of them manage the communications between the CPU and other parts of the motherboard, and constitute the core logic **chipset** of the PC motherboard.



Separating the different functions into the CPU, Northbridge, and Southbridge chips was due to the difficulty of integrating all components onto a single chip.

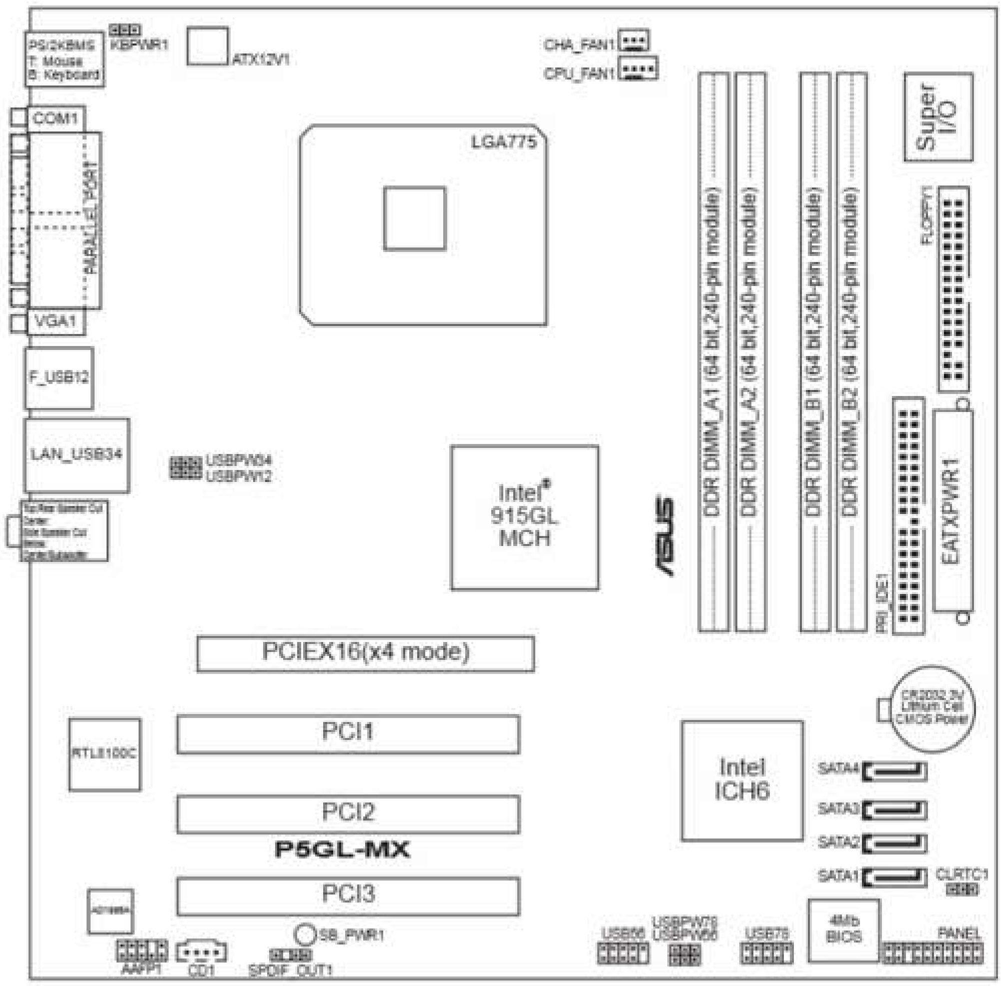
As CPU speeds increased over time, a bottleneck eventually emerged between the processor and the motherboard, due to limitations caused by data transmission between the CPU and its support chipset.

Modern Intel Core processors have the Northbridge integrated in a circuit on the CPU in order to simply avoid this bottleneck.

**Identify the following:**

- CPU,
- Northbridge,
- Southbridge
- PCI-E and
- I/O controllers

on the motherboard diagram below



TASK 1 - Describe and compare exactly what kind of data each of the following types of memory hold:

a. RAM

b. ROM

c. registers

d. cache

e. battery-backed RAM

TASK 2 - Explain the terms **'volatile'** and **'non-volatile'** when applied to memory.

TASK 3 - What is the purpose of the **'bootstrap program**"?

TASK 4 - Explain what is meant by ‘**booting up a computer**'.

TASK 5 - Explain why re-booting a computer often sorts out software problems.

TASK 6 - How many bytes are there in 256 Mbytes? You should give:

a) an approximate answer and

b) an exact answer.

TASK 7 - What does **BIOS** stand for?

TASK 8 - Explain why the operating system on a PC is not held in ROM.

TASK 9 - Describe how providing cache to a computer system without any should improve performance.

TASK 10 - Describe the purpose of battery-backed RAM.

TASK 11 - Explain with the aid of an example the problem of **'speed mismatch**‘ between two devices and describe how the computer overcomes this.

## Glossary

**Primary Memory**

There are two types of primary memory, ROM and RAM. Any other type of storage (such as magnetic, optical, or solid-state media) is secondary storage. Data stored in secondary storage cannot be accessed by the processor until it is moved to RAM.

**Read Only Memory (ROM)**

Read Only Memory, or ROM, is where data that is vital to the operation of the computer, such as BIOS and UEFI, because it cannot be deleted, altered, or tampered with. The main disadvantage of this is that the software stored in ROM cannot be upgraded or updated.

**Random Access Memory (RAM)**

RAM stores the programs that are being used by the computer (including the operating system). RAM is volatile memory, as any data stored in it is lost when the computer is switched off, or loses power.

### Secondary Storage

Secondary storage is where data is stored that does not need to be accessed by the processor. It is generally far larger than primary storage - a powerful computer might have 16GB of primary memory, but 2,000GB of secondary storage. Secondary storage is also non-volatile, which is good, otherwise every time there is a power cut, or the computer gets turned off, all the data stored on the computer would be lost (except the ROM).

### Optical

Optical media is popular because it is relatively cheap, and it is very reliable, and can be dropped and mistreated with no ill effects, as long as the surface of the disc is protected. Data is stored on optical media using "pits" (binary 0) and "land" (binary 1), and is read using a laser. CDs, DVDs, and Blu-Ray discs are all examples of optical media. Most optical media is read-only, however it can sometimes be rewritten, for example CD-RWs.

A **disc** refers to optical media, but a **disk** refers to magnetic media!

### Magnetic

Broadly speaking, there are two types of magnetic media - tape and disks. Tape is mainly used for archiving, as it takes a long time to "seek" between two sections of tape. Hard disks are mainly used for long term data storage, where speed is also important, for example the storage of hundreds of high-quality movies on a computer. Hard disks are read using a "head", which skims the surface of several "platters", or disks. These are all encased in one sealed unit. Hard drives are faster than tape and optical media, but slower than a solid state drive.

### Solid-state

Solid state storage is generally divided into two categories - "slow" storage (such as a flash drive), and "fast" storage (such as an SSD). All solid state storage is very good at dealing with small files, as no movement is required to "find" (seek) files, like with magnetic or optical storage. However, SSD's are fast enough to replace a traditional hard drive, as they can be 4 or 5 times faster than a spinning hard drive. (A flash drive can also be used to replace a hard drive, however flash drives can be 20 times slower than hard drives, even though they can seek faster).

### Data Transfer

Data can either be stored in primary memory or secondary storage. However, sometimes data needs to be moved, for example, because the computer is about to be switched off, and anything in volatile memory (such as RAM) will be lost, or if the processor wants to work with some data, which is currently stored in secondary storage (but needs to be stored in primary storage for the processor to use it). Because the memory controller (which controls primary memory - RAM in this case) is part of the processor, the data has to be copied from secondary storage to the ALU (or vice versa).

However, we know that the ALU is extremely fast, compared with primary storage which is (relatively speaking) slow. This means that the processor will be waiting time waiting for secondary storage, when it should be processing something else. The solution is to have a buffer, which fills up. When it is full, an "interrupt" is sent to the processor, and the data in the buffer is transferred to the ALU, and then to primary memory. The same system works in the opposite way when transferring data from primary memory to secondary storage.

**Applications Software** - software which helps the user complete specific tasks - for example word processing.

**Archive** - information is archived if it doesn't need to be accessed frequently, but it may need to be accessed at some point in the future. For example, student records after students have left the school - if the student later asks for a reference, then the school would have to retrieve their student record from the archive.

**Basic Input Output System (BIOS)** - where important instructions explaining how to turn on the computer are stored - for example, how to find a keyboard and interpret keystrokes.

**Bit** - a binary 1 or 0. Bits are normally part of a byte.

**Bitrate** - the amount of data (measured in bits) that can be transmitted per second. Can also be called the *baud rate*.

**Byte** - a group of bits (usually 8) treated as one unit.

**Checksum** - an extra byte that is added to a block of data, which is the sum of the bits, ignoring any carry. This is then calculated by the receiver, and if the checksums match, then the data is accepted.

**Duplex** - where data can travel in both direction at the same time - e.g. ethernet.

**Echo (Error Checking)** - when data is sent, a copy is then returned to the sender, to check it is the same data as the sender sent.

**Error (Data Transmission)** - where data is corrupted (changed) while being transferred from one place to another. See 1.5.4 Error Checking.

**Ethernet** - a cable standard used to connect nodes within a LAN.

**Graphical User Interface (GUI)** - an intuitive interface using images to represent commands. It is easy for beginners to use; however, it can be slow as many commands are hidden so the user is not confused.

**Half Duplex** - where data can travel in both directions, but only in one direction at a time - e.g. a walkie-talkie.

**Handshake** - before two devices can communicate, they must ensure they are both ready to communicate, and agree the protocols to be used, by completing a "handshake".

**Hardware** - the physical components which make up the computer. Hardware hurts if you are hit over the head with it.

**Input Device** - a type of peripheral that can accept data, decode it, and send it as electrical pulses to the computer. For example: keyboards, mice, digital cameras, and microphones.

**I/O Device** - a device that communicates between the computer and the outside word - for example a keyboard or a printer. Sometimes also known as an HID (**H**uman **I**nterface **D**evice).

**Local Area Network (LAN)** - computers connected to each other in a geographically small area, for example a home network.

**Magnetic Ink Character Reader (MICR)** - reads characters which have been printed using special magnetic ink. The only common use for MICR is the account information printed on the bottom of cheques. Unlike barcodes, MICR can easily be read by both humans and computers.

**Network** - 2 or more computing devices connected together so that they can share data and resources.

**Network Interface Card (NIC)** - an expansion card that allows a computer to connect to a network. Most motherboards have wired (ethernet) NIC's built in, but an expansion card might be required for wireless access.

**Node** - a piece of equipment, such as a PC or peripheral, attached to a network.

**Optical Character Recognition (OCR)** - converts text on paper to digital characters, by comparing a scanned character with a database. Works best on typed pages, but also works with handwriting.

**Optical Mark Reader (OMR)** - recognizes marks on a sheet of paper. Commonly used in multiple choice exams, or on lottery tickets.

**Output Device** - a device that turns computer signals into a human-readable form, like a screen, printer, or speaker.

**Packet** - a group of bytes for network transmission. Each packet has a unique ID (a label), a sequence number, a destination address, and a checksum.

**Parallel (Data Transmission)** - using many wires, each transferring a bit at the same time (e.g. 8 wires to transfer an entire byte of 8 bits).

**Parity Checking** - a method used for error checking, where it is agreed that the number of bits will be odd or even. 7 bits of the byte are used for actual data, and one of the bits is used to ensure the parity is correct. See 1.5.4 Error Checking.

**Peripheral** - a device that is connected to a host computer, but not part of it. For example, a webcam or a graphics card would be a peripheral, as they are not required for a computer to operate. However, a power supply would not be a peripheral, as a computer would not function without one. Although RAM is technically a peripheral, removing it will effectively disable any modern machine, therefore it is a primary component.

**Primary Component** - a piece of hardware which is critical for a computer to work. For example, a CPU, or a power supply.

**Primary Memory** - ROM and RAM.

**Protocol** - a set of rules governing the transmission of data. There are two types of protocol - logical protocols (relating to software) and physical protocols (relating to hardware).

**Random Access Memory (RAM)** - used to store short term information - e.g. when browsing the internet, web pages might be downloaded and stored in RAM. It's name is derived from the fact that random storage locations can all be accessed quickly (less than 1ms) - in contrast to traditional hard disks, which can take up to 10 milliseconds due to mechanical limitations. RAM is volatile memory, so it can only be used for short term storage.

**Read Only Memory (ROM)** - used to store data that should never be altered, for example the BIOS or UEFI. Information stored in ROM cannot be altered, deleted, or otherwise interfered with.

**Serial (Data Transmission)** - where each of the 8 bits is transmitted down a single wire connection one at a time.

**Simplex** - data is only ever able to travel in one direction - e.g. from a radio station to a radio.

**Software** - the instructions and code that tells the computer what to do. There are two types of software, applications software and system software.

**Static State Drive (SSD)** - a replacement for hard drives. An SSD is smaller and faster than a hard drive which costs the same amount. Typically, the operating system is stored on an SSD, while large files such as photos, music, and videos, are stored on a "traditional" spinning hard disk.

**Standalone** - a computer which is not connected to other computers.

**Storage Device** - a memory device used to store operating systems, programs, and user data. Storage devices are non-volatile - they do not lose their data when they are unpowered. See 1.4.3 Memory and Storage.

**Syntax** - the particular rules of a given programming language. For example, surrounding strings in quotes (either single or double quotes) is a syntax rule for python.

**System Software** - system software provides basic functionality, such as file management, or providing a Graphical User Interface.

**Unified Extensible Firmware Interface (UEFI)** - a modern replacement for BIOS, which will reduce the time it takes for computers to start up. This is because it can be accessed through operating systems, such as Microsoft Windows. It also supports "modern" features, such as a mouse, making it more user-friendly.

**Volatile Memory** - memory which gets lost when the power is turned off.

**Wide Area Network (WAN)** - computers connected to each other over a geographically large area, for example the internet.

**Character Set**

The computers only understand two sets of values 1 (ON) and 0 (OFF). The computer is made up of a series of switches that can be turned **ON** or **OFF**. When you press a key on the keyboard a code is generated. This code is then translated into a number of switches that are turned ON or OFF. To represent this code we use bits (binary digits).

**0 1 0 1 0 1 0 0**

A character set is basically a way of expressing text (**letters**, **numbers**, **commands** and **symbols**) as binary digits (bits). In the 1960s, computers all used different character sets, so it was hard to make them communicate. Manufacturers realised this was a problem, so they had a meeting in America, and **created a set of codes called ASCII** (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange - pronounced ass-key). To represent all 128 ASCII characters, we need a 7 - bit representation as the $2^7$ = 128. Later on it was extended and it is now known as an Extended ASCII with an 8-bit representation. $2^8$ = 256 characters are enough to represent all other characters from France, Germany etc.

**ASCII** isn't the only character set - there are others, such as **EBCDIC**, which was used on IBM machines, and is based on Binary Coded Decimal, and **UNICODE**, which is similar to ASCII, but represents far more characters, including Arabic, Chinese, and Japanese characters.

People enter data into a computer by means of an input device such as a keyboard. But actually a keyboard is simply a set of switches arranged in a certain way for your convenience. Every key is physically identical - there is no 'letter A' key or 'letter X' key.

What happens is that the keyboard sends a signal to the computer that effectively says 'The third key on the middle row has been pressed'. This 'signal' is a combination of switches which are turned ON or OFF in order to produce a binary number that represents it.

The computer then has to work out what this actually means, so there has to be a translation between the 'third key signal' and what needs to appear on the screen. **This is the job of the 'character set'**.

**"A character set converts a binary number / code into a written language character".**

For example, if you use a European language keyboard then the "Latin Alphabet No 1" character set may be installed on the computer to translate the keys into the right language symbols for you (there are other character sets that would also do the same job).

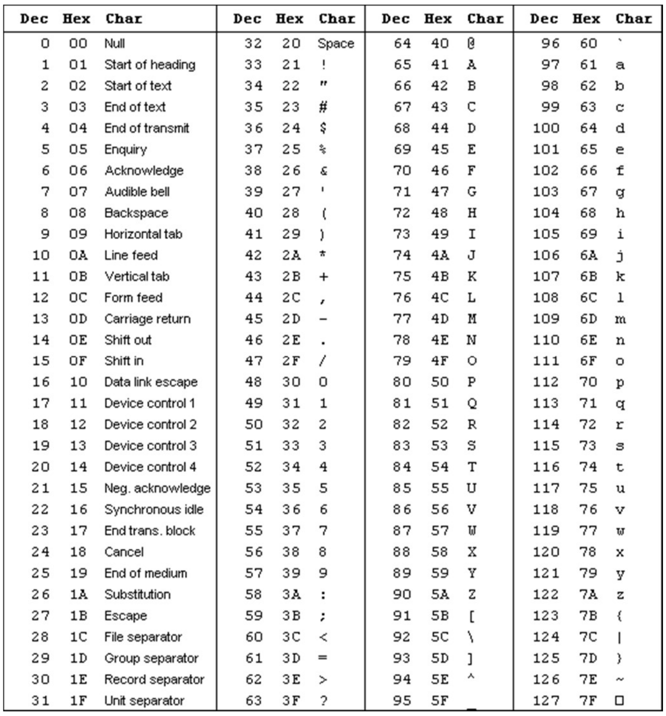
A completely different language such as Japanese would use a different character set and of course the middle-row third key would have a different symbol on it.

***Do not confuse character set with font***. They are not the same thing. A font will display the letter 'A' in a certain way - but it is still the letter A no matter how fancy the font!

**It is the character set that maps a binary code to the letter** A whatever font you have selected will then be display it in certain way.

So far, we have discussed 1 byte character sets such as ASCII and extended ASCII. But these can only describe 256 symbols. Some languages have far more than this. For instance some far-east languages have more than 12,000 characters!

So the logical way of handling the problem is to use a character set that uses more than 8 bits (1 byte).



| Dec | Hex | Char | Dec | Hex | Char | Dec | Hex | Char | Dec | Hex | Char |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 00 | Null | 32 | 20 | Space | 64 | 40 | @ | 96 | 60 | ` |
| 1 | 01 | Start of heading | 33 | 21 | ! | 65 | 41 | A | 97 | 61 | a |
| 2 | 02 | Start of text | 34 | 22 | " | 66 | 42 | B | 98 | 62 | b |
| 3 | 03 | End of text | 35 | 23 | # | 67 | 43 | C | 99 | 63 | c |
| 4 | 04 | End of transmit | 36 | 24 | $ | 68 | 44 | D | 100 | 64 | d |
| 5 | 05 | Enquiry | 37 | 25 | % | 69 | 45 | E | 101 | 65 | e |
| 6 | 06 | Acknowledge | 38 | 26 | & | 70 | 46 | F | 102 | 66 | f |
| 7 | 07 | Audible bell | 39 | 27 | ' | 71 | 47 | G | 103 | 67 | g |
| 8 | 08 | Backspace | 40 | 28 | ( | 72 | 48 | H | 104 | 68 | h |
| 9 | 09 | Horizontal tab | 41 | 29 | ) | 73 | 49 | I | 105 | 69 | i |
| 10 | 0A | Line feed | 42 | 2A | * | 74 | 4A | J | 106 | 6A | j |
| 11 | 0B | Vertical tab | 43 | 2B | + | 75 | 4B | K | 107 | 6B | k |
| 12 | 0C | Form feed | 44 | 2C | , | 76 | 4C | L | 108 | 6C | l |
| 13 | 0D | Carriage return | 45 | 2D | - | 77 | 4D | M | 109 | 6D | m |
| 14 | 0E | Shift out | 46 | 2E | . | 78 | 4E | N | 110 | 6E | n |
| 15 | 0F | Shift in | 47 | 2F | / | 79 | 4F | O | 111 | 6F | o |
| 16 | 10 | Data link escape | 48 | 30 | 0 | 80 | 50 | P | 112 | 70 | p |
| 17 | 11 | Device control 1 | 49 | 31 | 1 | 81 | 51 | Q | 113 | 71 | q |
| 18 | 12 | Device control 2 | 50 | 32 | 2 | 82 | 52 | R | 114 | 72 | r |
| 19 | 13 | Device control 3 | 51 | 33 | 3 | 83 | 53 | S | 115 | 73 | s |
| 20 | 14 | Device control 4 | 52 | 34 | 4 | 84 | 54 | T | 116 | 74 | t |
| 21 | 15 | Neg. acknowledge | 53 | 35 | 5 | 85 | 55 | U | 117 | 75 | u |
| 22 | 16 | Synchronous idle | 54 | 36 | 6 | 86 | 56 | V | 118 | 76 | v |
| 23 | 17 | End trans. block | 55 | 37 | 7 | 87 | 57 | W | 119 | 77 | w |
| 24 | 18 | Cancel | 56 | 38 | 8 | 88 | 58 | X | 120 | 78 | x |
| 25 | 19 | End of medium | 57 | 39 | 9 | 89 | 59 | Y | 121 | 79 | y |
| 26 | 1A | Substitution | 58 | 3A | : | 90 | 5A | Z | 122 | 7A | z |
| 27 | 1B | Escape | 59 | 3B | ; | 91 | 5B | [ | 123 | 7B | { |
| 28 | 1C | File separator | 60 | 3C | < | 92 | 5C | \ | 124 | 7C | \| |
| 29 | 1D | Group separator | 61 | 3D | = | 93 | 5D | ] | 125 | 7D | } |
| 30 | 1E | Record separator | 62 | 3E | > | 94 | 5E | ^ | 126 | 7E | ~ |
| 31 | 1F | Unit separator | 63 | 3F | ? | 95 | 5F | _ | 127 | 7F | □ |

## ASCII – American Standard Code for International Interchange

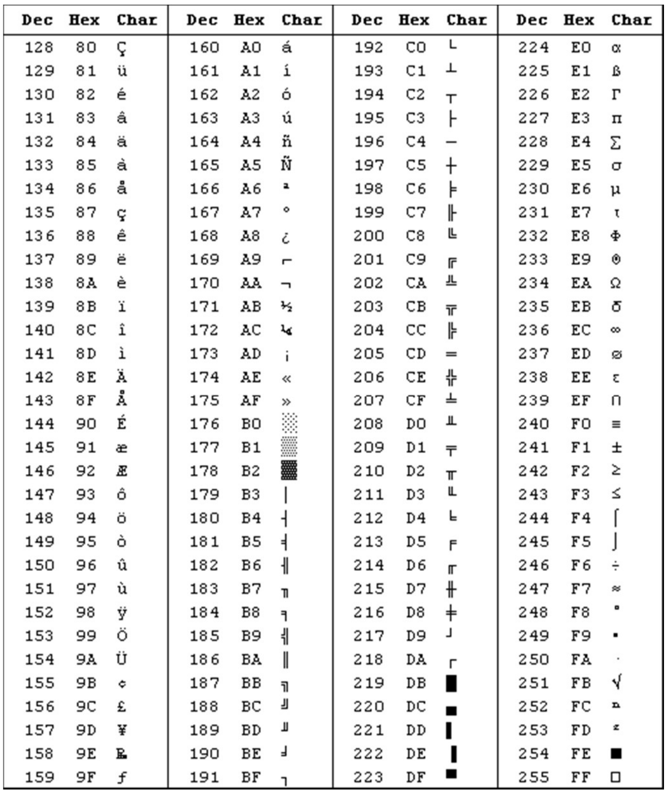
It contains 128 characters but **33 of these are non-printing** and mostly obsolete control character that affect how text is processed. All **95 other characters correspond to a number.**

How many bits do you need to store all 128 characters?

**This is an Extended ASCII table.**

This character covers other Language characters such as French, symbols etc. that are not covered by the ASCII table.

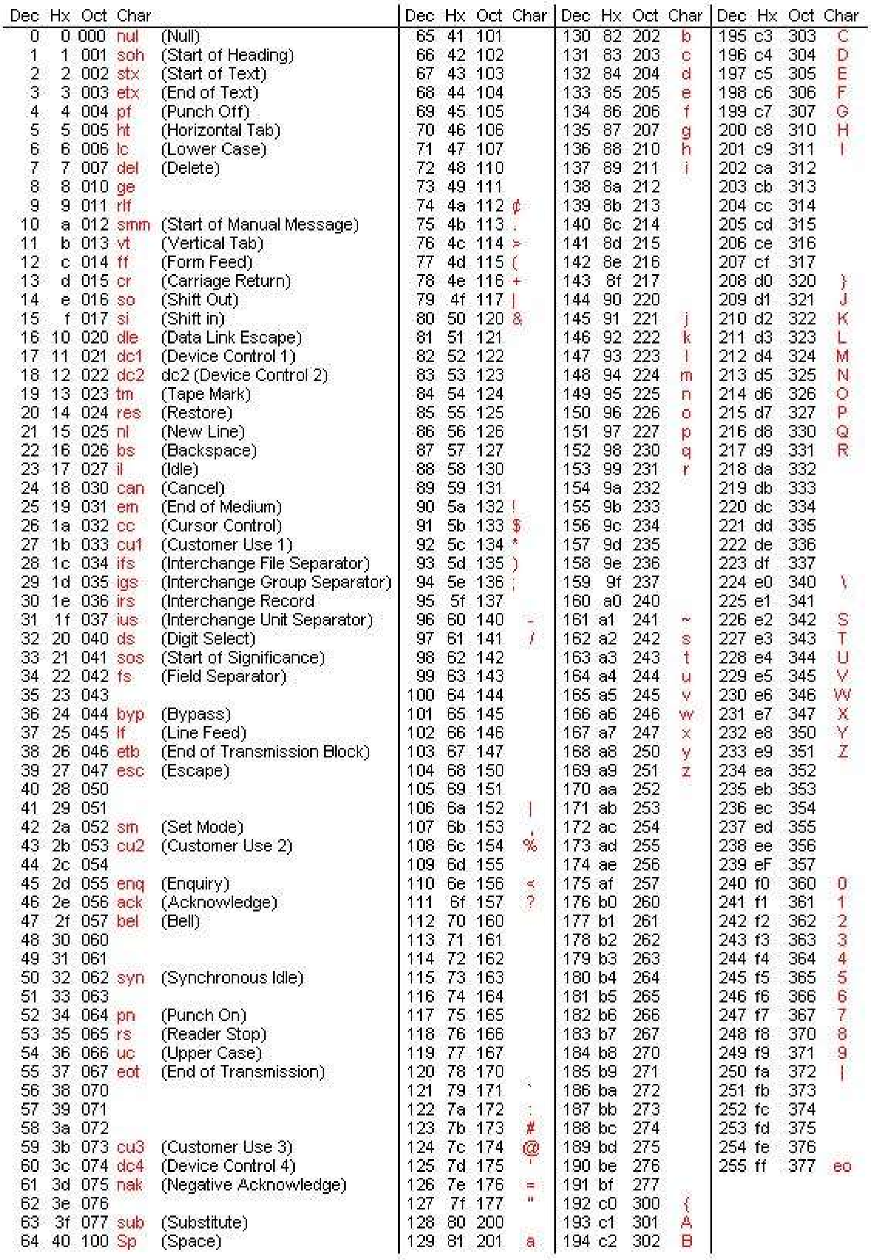
How many bits do you need to store all 128 characters?



| Dec | Hex | Char | Dec | Hex | Char | Dec | Hex | Char | Dec | Hex | Char |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 128 | 80 | Ç | 160 | A0 | á | 192 | C0 | └ | 224 | E0 | α |
| 129 | 81 | ü | 161 | A1 | í | 193 | C1 | ┴ | 225 | E1 | ß |
| 130 | 82 | é | 162 | A2 | ó | 194 | C2 | ┬ | 226 | E2 | Γ |
| 131 | 83 | â | 163 | A3 | ú | 195 | C3 | ├ | 227 | E3 | π |
| 132 | 84 | ä | 164 | A4 | ñ | 196 | C4 | ─ | 228 | E4 | Σ |
| 133 | 85 | à | 165 | A5 | Ñ | 197 | C5 | ┼ | 229 | E5 | σ |
| 134 | 86 | å | 166 | A6 | ª | 198 | C6 | ╞ | 230 | E6 | µ |
| 135 | 87 | ç | 167 | A7 | º | 199 | C7 | ╟ | 231 | E7 | τ |
| 136 | 88 | ê | 168 | A8 | ¿ | 200 | C8 | ╚ | 232 | E8 | Φ |
| 137 | 89 | ë | 169 | A9 | ⌐ | 201 | C9 | ╔ | 233 | E9 | Θ |
| 138 | 8A | è | 170 | AA | ¬ | 202 | CA | ╩ | 234 | EA | Ω |
| 139 | 8B | ï | 171 | AB | ½ | 203 | CB | ╦ | 235 | EB | δ |
| 140 | 8C | î | 172 | AC | ¼ | 204 | CC | ╠ | 236 | EC | ∞ |
| 141 | 8D | ì | 173 | AD | ¡ | 205 | CD | ═ | 237 | ED | φ |
| 142 | 8E | Ä | 174 | AE | « | 206 | CE | ╬ | 238 | EE | ε |
| 143 | 8F | Å | 175 | AF | » | 207 | CF | ╧ | 239 | EF | ∩ |
| 144 | 90 | É | 176 | B0 | ░ | 208 | D0 | ╨ | 240 | F0 | ≡ |
| 145 | 91 | æ | 177 | B1 | ▒ | 209 | D1 | ╤ | 241 | F1 | ± |
| 146 | 92 | Æ | 178 | B2 | ▓ | 210 | D2 | ╥ | 242 | F2 | ≥ |
| 147 | 93 | ô | 179 | B3 | │ | 211 | D3 | ╙ | 243 | F3 | ≤ |
| 148 | 94 | ö | 180 | B4 | ┤ | 212 | D4 | ╘ | 244 | F4 | ⌠ |
| 149 | 95 | ò | 181 | B5 | ╡ | 213 | D5 | ╒ | 245 | F5 | ⌡ |
| 150 | 96 | û | 182 | B6 | ╢ | 214 | D6 | ╓ | 246 | F6 | ÷ |
| 151 | 97 | ù | 183 | B7 | ╖ | 215 | D7 | ╫ | 247 | F7 | ≈ |
| 152 | 98 | ÿ | 184 | B8 | ╕ | 216 | D8 | ╪ | 248 | F8 | ° |
| 153 | 99 | Ö | 185 | B9 | ╣ | 217 | D9 | ┘ | 249 | F9 | ∙ |
| 154 | 9A | Ü | 186 | BA | ║ | 218 | DA | ┌ | 250 | FA | · |
| 155 | 9B | ¢ | 187 | BB | ╗ | 219 | DB | █ | 251 | FB | √ |
| 156 | 9C | £ | 188 | BC | ╝ | 220 | DC | ▄ | 252 | FC | ⁿ |
| 157 | 9D | ¥ | 189 | BD | ╜ | 221 | DD | ▌ | 253 | FD | ² |
| 158 | 9E | ₧ | 190 | BE | ╛ | 222 | DE | ▐ | 254 | FE | ■ |
| 159 | 9F | ƒ | 191 | BF | ┐ | 223 | DF | ▀ | 255 | FF | □ |

## EBCDIC – Extended Binary Coded Decimal Interchange Code

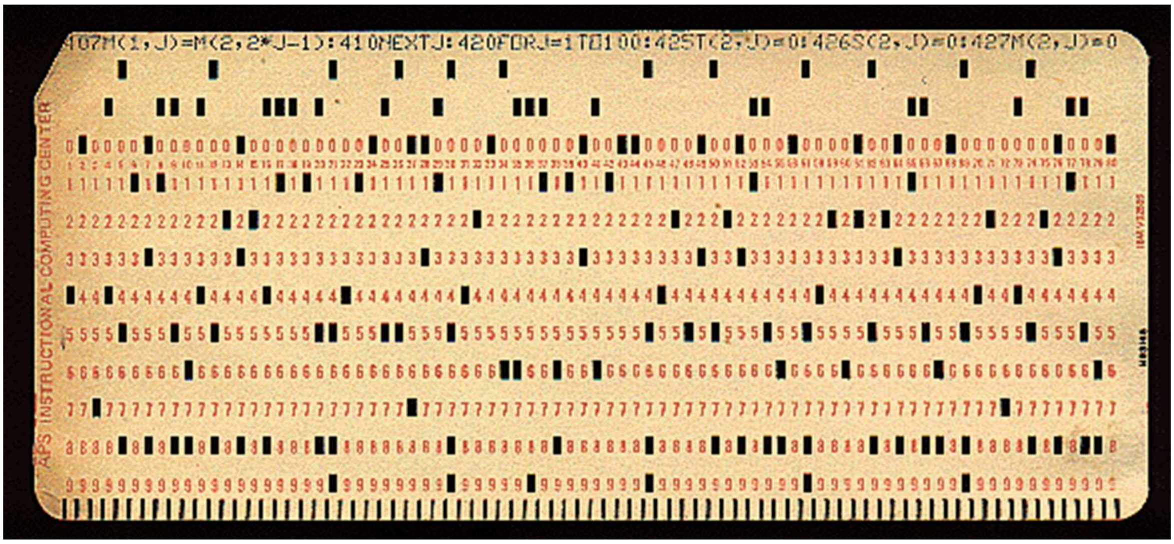
This character set was developed by IBM in 1960 for punched cards (see the bottom of the page) in the early 1960s and still uses it on mainframes today. Punched cards were another way of entering data into the computer through some input machines that used to read the cards; the card readers. It is an ideal character set to be used by computers that process a lot of data. The numbers used in this character set are completely different to the numbers used by the ASCII character. If one computer uses ASCII and the other uses EBCDIC then they will not be able to communicate with one another.



| Dec | Hx | Oct | Char | | Dec | Hx | Oct | Char | Dec | Hx | Oct | Char | Dec | Hx | Oct | Char |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 000 | nul | (Null) | 65 | 41 | 101 | | 130 | 82 | 202 | b | 195 | c3 | 303 | C |
| 1 | 1 | 001 | soh | (Start of Heading) | 66 | 42 | 102 | | 131 | 83 | 203 | c | 196 | c4 | 304 | D |
| 2 | 2 | 002 | stx | (Start of Text) | 67 | 43 | 103 | | 132 | 84 | 204 | d | 197 | c5 | 305 | E |
| 3 | 3 | 003 | etx | (End of Text) | 68 | 44 | 104 | | 133 | 85 | 205 | e | 198 | c6 | 306 | F |
| 4 | 4 | 004 | pf | (Punch Off) | 69 | 45 | 105 | | 134 | 86 | 206 | f | 199 | c7 | 307 | G |
| 5 | 5 | 005 | ht | (Horizontal Tab) | 70 | 46 | 106 | | 135 | 87 | 207 | g | 200 | c8 | 310 | H |
| 6 | 6 | 006 | lc | (Lower Case) | 71 | 47 | 107 | | 136 | 88 | 210 | h | 201 | c9 | 311 | I |
| 7 | 7 | 007 | del | (Delete) | 72 | 48 | 110 | | 137 | 89 | 211 | i | 202 | ca | 312 | |
| 8 | 8 | 010 | ge | | 73 | 49 | 111 | | 138 | 8a | 212 | | 203 | cb | 313 | |
| 9 | 9 | 011 | rlf | | 74 | 4a | 112 | ¢ | 139 | 8b | 213 | | 204 | cc | 314 | |
| 10 | a | 012 | smm | (Start of Manual Message) | 75 | 4b | 113 | . | 140 | 8c | 214 | | 205 | cd | 315 | |
| 11 | b | 013 | vt | (Vertical Tab) | 76 | 4c | 114 | > | 141 | 8d | 215 | | 206 | ce | 316 | |
| 12 | c | 014 | ff | (Form Feed) | 77 | 4d | 115 | ( | 142 | 8e | 216 | | 207 | cf | 317 | |
| 13 | d | 015 | cr | (Carriage Return) | 78 | 4e | 116 | + | 143 | 8f | 217 | | 208 | d0 | 320 | } |
| 14 | e | 016 | so | (Shift Out) | 79 | 4f | 117 | \| | 144 | 90 | 220 | | 209 | d1 | 321 | J |
| 15 | f | 017 | si | (Shift in) | 80 | 50 | 120 | & | 145 | 91 | 221 | j | 210 | d2 | 322 | K |
| 16 | 10 | 020 | dle | (Data Link Escape) | 81 | 51 | 121 | | 146 | 92 | 222 | k | 211 | d3 | 323 | L |
| 17 | 11 | 021 | dc1 | (Device Control 1) | 82 | 52 | 122 | | 147 | 93 | 223 | l | 212 | d4 | 324 | M |
| 18 | 12 | 022 | dc2 | dc2 (Device Control 2) | 83 | 53 | 123 | | 148 | 94 | 224 | m | 213 | d5 | 325 | N |
| 19 | 13 | 023 | tm | (Tape Mark) | 84 | 54 | 124 | | 149 | 95 | 225 | n | 214 | d6 | 326 | O |
| 20 | 14 | 024 | res | (Restore) | 85 | 55 | 125 | | 150 | 96 | 226 | o | 215 | d7 | 327 | P |
| 21 | 15 | 025 | nl | (New Line) | 86 | 56 | 126 | | 151 | 97 | 227 | p | 216 | d8 | 330 | Q |
| 22 | 16 | 026 | bs | (Backspace) | 87 | 57 | 127 | | 152 | 98 | 230 | q | 217 | d9 | 331 | R |
| 23 | 17 | 027 | il | (Idle) | 88 | 58 | 130 | | 153 | 99 | 231 | r | 218 | da | 332 | |
| 24 | 18 | 030 | can | (Cancel) | 89 | 59 | 131 | | 154 | 9a | 232 | | 219 | db | 333 | |
| 25 | 19 | 031 | em | (End of Medium) | 90 | 5a | 132 | ! | 155 | 9b | 233 | | 220 | dc | 334 | |
| 26 | 1a | 032 | cc | (Cursor Control) | 91 | 5b | 133 | $ | 156 | 9c | 234 | | 221 | dd | 335 | |
| 27 | 1b | 033 | cu1 | (Customer Use 1) | 92 | 5c | 134 | * | 157 | 9d | 235 | | 222 | de | 336 | |
| 28 | 1c | 034 | ifs | (Interchange File Separator) | 93 | 5d | 135 | ) | 158 | 9e | 236 | | 223 | df | 337 | |
| 29 | 1d | 035 | igs | (Interchange Group Separator) | 94 | 5e | 136 | ; | 159 | 9f | 237 | | 224 | e0 | 340 | \ |
| 30 | 1e | 036 | irs | (Interchange Record | 95 | 5f | 137 | | 160 | a0 | 240 | | 225 | e1 | 341 | |
| 31 | 1f | 037 | ius | (Interchange Unit Separator) | 96 | 60 | 140 | - | 161 | a1 | 241 | ~ | 226 | e2 | 342 | S |
| 32 | 20 | 040 | ds | (Digit Select) | 97 | 61 | 141 | / | 162 | a2 | 242 | s | 227 | e3 | 343 | T |
| 33 | 21 | 041 | sos | (Start of Significance) | 98 | 62 | 142 | | 163 | a3 | 243 | t | 228 | e4 | 344 | U |
| 34 | 22 | 042 | fs | (Field Separator) | 99 | 63 | 143 | | 164 | a4 | 244 | u | 229 | e5 | 345 | V |
| 35 | 23 | 043 | | | 100 | 64 | 144 | | 165 | a5 | 245 | v | 230 | e6 | 346 | W |
| 36 | 24 | 044 | byp | (Bypass) | 101 | 65 | 145 | | 166 | a6 | 246 | w | 231 | e7 | 347 | X |
| 37 | 25 | 045 | lf | (Line Feed) | 102 | 66 | 146 | | 167 | a7 | 247 | x | 232 | e8 | 350 | Y |
| 38 | 26 | 046 | etb | (End of Transmission Block) | 103 | 67 | 147 | | 168 | a8 | 250 | y | 233 | e9 | 351 | Z |
| 39 | 27 | 047 | esc | (Escape) | 104 | 68 | 150 | | 169 | a9 | 251 | z | 234 | ea | 352 | |
| 40 | 28 | 050 | | | 105 | 69 | 151 | | 170 | aa | 252 | | 235 | eb | 353 | |
| 41 | 29 | 051 | | | 106 | 6a | 152 | \| | 171 | ab | 253 | | 236 | ec | 354 | |
| 42 | 2a | 052 | sm | (Set Mode) | 107 | 6b | 153 | , | 172 | ac | 254 | | 237 | ed | 355 | |
| 43 | 2b | 053 | cu2 | (Customer Use 2) | 108 | 6c | 154 | % | 173 | ad | 255 | | 238 | ee | 356 | |
| 44 | 2c | 054 | | | 109 | 6d | 155 | | 174 | ae | 256 | | 239 | eF | 357 | |
| 45 | 2d | 055 | enq | (Enquiry) | 110 | 6e | 156 | < | 175 | af | 257 | | 240 | f0 | 360 | 0 |
| 46 | 2e | 056 | ack | (Acknowledge) | 111 | 6f | 157 | ? | 176 | b0 | 260 | | 241 | f1 | 361 | 1 |
| 47 | 2f | 057 | bel | (Bell) | 112 | 70 | 160 | | 177 | b1 | 261 | | 242 | f2 | 362 | 2 |
| 48 | 30 | 060 | | | 113 | 71 | 161 | | 178 | b2 | 262 | | 243 | f3 | 363 | 3 |
| 49 | 31 | 061 | | | 114 | 72 | 162 | | 179 | b3 | 263 | | 244 | f4 | 364 | 4 |
| 50 | 32 | 062 | syn | (Synchronous Idle) | 115 | 73 | 163 | | 180 | b4 | 264 | | 245 | f5 | 365 | 5 |
| 51 | 33 | 063 | | | 116 | 74 | 164 | | 181 | b5 | 265 | | 246 | f6 | 366 | 6 |
| 52 | 34 | 064 | pn | (Punch On) | 117 | 75 | 165 | | 182 | b6 | 266 | | 247 | f7 | 367 | 7 |
| 53 | 35 | 065 | rs | (Reader Stop) | 118 | 76 | 166 | | 183 | b7 | 267 | | 248 | f8 | 370 | 8 |
| 54 | 36 | 066 | uc | (Upper Case) | 119 | 77 | 167 | | 184 | b8 | 270 | | 249 | f9 | 371 | 9 |
| 55 | 37 | 067 | eot | (End of Transmission) | 120 | 78 | 170 | | 185 | b9 | 271 | | 250 | fa | 372 | \| |
| 56 | 38 | 070 | | | 121 | 79 | 171 | ` | 186 | ba | 272 | | 251 | fb | 373 | |
| 57 | 39 | 071 | | | 122 | 7a | 172 | : | 187 | bb | 273 | | 252 | fc | 374 | |
| 58 | 3a | 072 | | | 123 | 7b | 173 | # | 188 | bc | 274 | | 253 | fd | 375 | |
| 59 | 3b | 073 | cu3 | (Customer Use 3) | 124 | 7c | 174 | @ | 189 | bd | 275 | | 254 | fe | 376 | |
| 60 | 3c | 074 | dc4 | (Device Control 4) | 125 | 7d | 175 | ' | 190 | be | 276 | | 255 | ff | 377 | eo |
| 61 | 3d | 075 | nak | (Negative Acknowledge) | 126 | 7e | 176 | = | 191 | bf | 277 | | | | | |
| 62 | 3e | 076 | | | 127 | 7f | 177 | " | 192 | c0 | 300 | { | | | | |
| 63 | 3f | 077 | sub | (Substitute) | 128 | 80 | 200 | | 193 | c1 | 301 | A | | | | |
| 64 | 40 | 100 | Sp | (Space) | 129 | 81 | 201 | a | 194 | c2 | 302 | B | | | | |

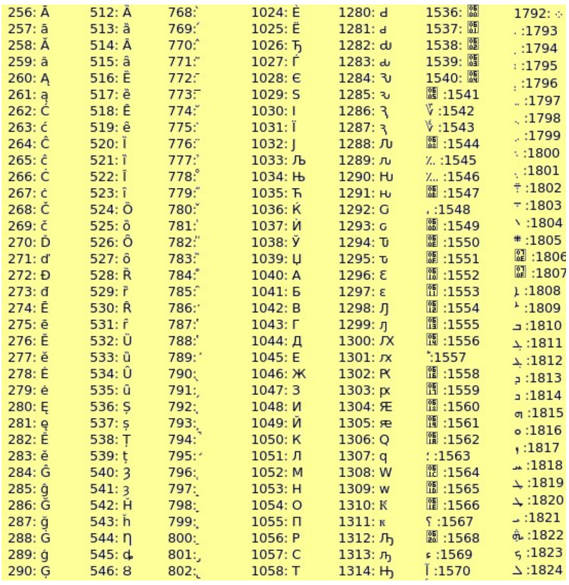
This character set is using an 8 – bit with many empty numbers not currently used. This set came out at the same time as the ASCII so we ended up with these two sets as some manufacturers preferred one to another.

The EBCDIC like ASCII uses 8 bit codes.


407M(1,J)=M(2,2*J-1):410NEXTJ:420FORJ=1TO100:425T(2,J)=0:426S(2,J)=0:427M(2,J)=0
APS INSTRUCTIONAL COMPUTING CENTER

## Unicode

As the Extended ASCII only covers 256 characters there is a need for a larger set to include other characters from other languages as well. Unicode is another character set which uses a 16 bit code (twice the bits used for ASCII) and represents over 65000 characters. With this set you have the ability to represent all the writing systems of the world; languages such as Chinese, Japanese, Arabic and even ancient languages such as Egyptian, Latin or Ancient Greek. Microsoft and Apple both support the use of Unicode for their operating systems.



| | | | | | | |
|---|---|---|---|---|---|---|
| 256: Ā | 512: Ȁ | 768:̀ | 1024: Ѐ | 1280: Ԁ | 1536: □ | 1792: ܀ |
| 257: ā | 513: ȁ | 769:́ | 1025: Ё | 1281: ԁ | 1537: □ | ܁ :1793 |
| 258: Ă | 514: Ȃ | 770:̂ | 1026: Ђ | 1282: Ԃ | 1538: □ | ܂ :1794 |
| 259: ă | 515: ȃ | 771:̃ | 1027: Ѓ | 1283: ԃ | 1539: □ | ܃ :1795 |
| 260: Ą | 516: Ȅ | 772:̄ | 1028: Є | 1284: Ԅ | 1540: □ | ܄ :1796 |
| 261: ą | 517: ȅ | 773:̅ | 1029: Ѕ | 1285: ԅ | □ :1541 | ܅ :1797 |
| 262: Ć | 518: Ȇ | 774:̆ | 1030: І | 1286: Ԇ | ؆ :1542 | ܆ :1798 |
| 263: ć | 519: ȇ | 775:̇ | 1031: Ї | 1287: ԇ | ؇ :1543 | ܇ :1799 |
| 264: Ĉ | 520: Ȉ | 776:̈ | 1032: Ј | 1288: Ԉ | □ :1544 | ܈ :1800 |
| 265: ĉ | 521: ȉ | 777:̉ | 1033: Љ | 1289: ԉ | ؉ :1545 | ܉ :1801 |
| 266: Ċ | 522: Ȋ | 778:̊ | 1034: Њ | 1290: Ԋ | ؊ :1546 | ܊ :1802 |
| 267: ċ | 523: ȋ | 779:̋ | 1035: Ћ | 1291: ԋ | □ :1547 | ܋ :1803 |
| 268: Č | 524: Ȍ | 780:̌ | 1036: Ќ | 1292: Ԍ | ، :1548 | ܌ :1804 |
| 269: č | 525: ȍ | 781:̍ | 1037: Ѝ | 1293: ԍ | □ :1549 | ܍ :1805 |
| 270: Ď | 526: Ȏ | 782:̎ | 1038: Ў | 1294: Ԏ | □ :1550 | □ :1806 |
| 271: ď | 527: ȏ | 783:̏ | 1039: Џ | 1295: ԏ | □ :1551 | □ :1807 |
| 272: Đ | 528: Ȑ | 784:̐ | 1040: А | 1296: Ԑ | □ :1552 | ܐ :1808 |
| 273: đ | 529: ȑ | 785:̑ | 1041: Б | 1297: ԑ | □ :1553 | ܑ :1809 |
| 274: Ē | 530: Ȓ | 786:̒ | 1042: В | 1298: Ԓ | □ :1554 | ܒ :1810 |
| 275: ē | 531: ȓ | 787:̓ | 1043: Г | 1299: ԓ | □ :1555 | ܓ :1811 |
| 276: Ĕ | 532: Ȕ | 788:̔ | 1044: Д | 1300: Ԕ | □ :1556 | ܔ :1812 |
| 277: ĕ | 533: ȕ | 789:̕ | 1045: Е | 1301: ԕ | ؕ:1557 | ܕ :1813 |
| 278: Ė | 534: Ȗ | 790:̖ | 1046: Ж | 1302: Ԗ | □ :1558 | ܖ :1814 |
| 279: ė | 535: ȗ | 791:̗ | 1047: З | 1303: ԗ | □ :1559 | ܗ :1815 |
| 280: Ę | 536: Ș | 792:̘ | 1048: И | 1304: Ԙ | □ :1560 | ܘ :1816 |
| 281: ę | 537: ș | 793:̙ | 1049: Й | 1305: ԙ | □ :1561 | ܙ :1817 |
| 282: Ě | 538: Ț | 794:̚ | 1050: К | 1306: Ԛ | □ :1562 | ܚ :1818 |
| 283: ě | 539: ț | 795:̛ | 1051: Л | 1307: ԛ | ؛ :1563 | ܛ :1819 |
| 284: Ĝ | 540: Ȝ | 796:̜ | 1052: М | 1308: Ԝ | □ :1564 | ܜ :1820 |
| 285: ĝ | 541: ȝ | 797:̝ | 1053: Н | 1309: ԝ | □ :1565 | ܝ :1821 |
| 286: Ğ | 542: Ȟ | 798:̞ | 1054: О | 1310: Ԟ | □ :1566 | ܞ :1822 |
| 287: ğ | 543: ȟ | 799:̟ | 1055: П | 1311: ԟ | ؟ :1567 | ܟ :1823 |
| 288: Ġ | 544: Ƞ | 800:̠ | 1056: Р | 1312: Ԡ | □ :1568 | ܠ :1824 |
| 289: ġ | 545: ȡ | 801:̡ | 1057: С | 1313: ԡ | ء :1569 | |
| 290: Ģ | 546: Ȣ | 802:̢ | 1058: Т | 1314: Ԣ | آ :1570 | |

A **1 byte** (8-bits) scheme can only represent 256 symbols. This is fine for many individual languages which is why **ASCII** is so popular. But a 2 byte scheme can represent 65,000+ characters ( $2^{16}$ ). This is more than enough to hold every currently used language in the world in one place.

Imagine that you own a company that sells computers all over the world. Naturally every customer will want to type in their own language. What to do? You want a system that can handle every possible written language - when the computer gets turned on for the first time they simply choose the language that the operating system needs to handle.

This is why **2 byte character** sets were developed. A very popular 2 byte (16 bit) encoding standard is called 'Unicode'.

Unicode can handle any language and it does so by the user selecting a specific 'code page' which is one portion of the total **Unicode** space. Each code page represents the chosen language. For example code page 1253 within Unicode represents the Greek language.

If the person in Greece has a Greek keyboard and the Greek code page is selected within the operating system then the correct characters appear on the screen when they press a certain key on the keyboard. A later version of Unicode uses even more bits to include ancient languages such as Egyptian Hieroglyphics.

## Relationships between the number of bits and characters in a character set

To summarise, the number of bits will tell you how many characters a character set can represent. For example the ASCII is a 7-bit character set so therefore $2^7$ = 128 characters. So the **ASCII** character set will allow for 128 symbols which is fine for many Latin languages such as English.

The 8 bit **Extended ASCII** character set allows for 256 symbols ($2^8$ = 256)

Moving on to multi-byte schemes such as 1**6-bit Unicode**, $2^{16}$allows for 65,535 symbols to be represented. This allows for all current written languages to be represented under one scheme.

A further extension of Unicode expanded the scheme to **21-bit Unicode**, so offering over 1 million symbols, ($2^{21}$ = 1,114,111). This is enough space for even dead languages such as Egyptian Hieroglyphics to be represented.

Q1. Explain what is meant by the character set of a computer.

………………………………………………………………………………………………………………………………………………

………………………………………………………………………………………………………………………………………………

………………………………………………………………………………………………………………………………………………

………………………………………………………………………………………………………………………………………………

…………………………………………………………………………………………………………………………………………[2]

Q2. By referring to two examples of applications that need character sets of different sizes, explain how codes are used to represent character sets.

………………………………………………………………………………………………………………………………………………

………………………………………………………………………………………………………………………………………………

………………………………………………………………………………………………………………………………………………

………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………[1,1,2]

Q3. A car insurance firm collects data from its customers and stores it on a computer. The customer name is stored using the computers character set. Explain what is meant by the character set of a computer.

………………………………………………………………………………………………………………………………………………

………………………………………………………………………………………………………………………………………………

………………………………………………………………………………………………………………………………………………

………………………………………………………………………………………………………………………………………………

…………………………………………………………………………………………………………………………………………[2]

Q4. Explain the use of code to represent a character set.

………………………………………………………………………………………………………………………………………………

………………………………………………………………………………………………………………………………………………

………………………………………………………………………………………………………………………………………………

………………………………………………………………………………………………………………………………………………

…………………………………………………………………………………………………………………………………………[2]

This chapter will demonstrate the 'computer architecture'; how a computer is put together and why.

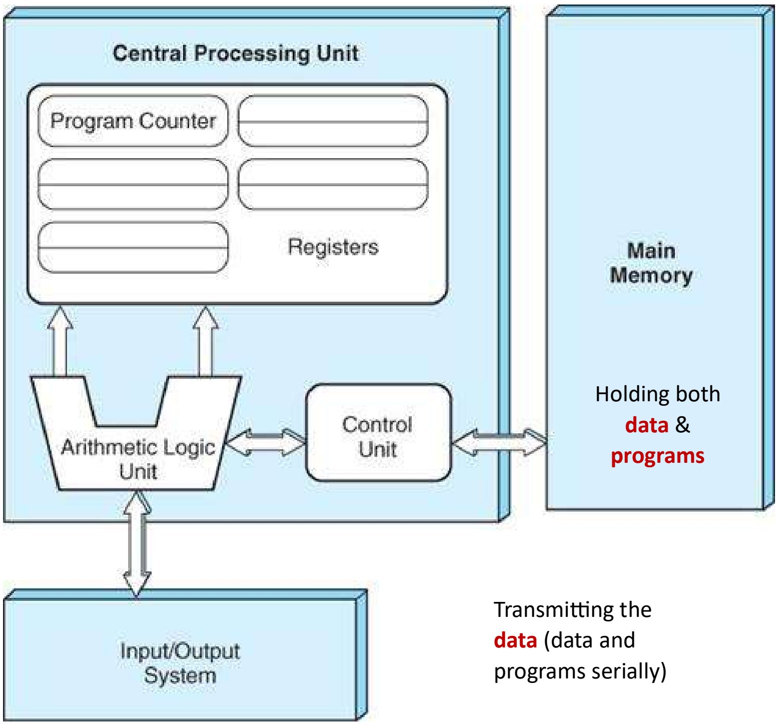
It will describe to you the Von Neumann Machine Architecture and it will introduce you to the **internal parts** of a central processing unit (**CPU**), its **registers** and how it handles a program in a **fetch-execute cycle**.

## The Von Neumann machine

Many years ago, in fact 1945, just after the World War, two mathematician-scientists independently proposed how to build a more flexible computer.

One was the British mathematician Alan Turing and the other was the equally talented American scientist John Von Neumann. Alan Turing had been involved with breaking the Enigma code in Bletchley Park using the 'Colossus' computer and John Von Neumann had been working on the Manhattan Project to build the first atomic bomb which needed a vast amount of manual calculations.

Up to that time, the war-time computers where 'programmed' more or less by rebuilding the entire machine to carry out a different task. For example, the early computer called ENIAC took three weeks to re-wire in order to do a different calculation. There had to be a better way and this is where the Von Neumann machine came into place.



Transmitting the **data** (data and programs serially)

The new idea was that not only should the **data be stored in memory**, *but the* ***program which is processing that data should also be stored in the same memory.***

This novel idea meant that a computer built with this architecture would be much easier to re-program. Effectively the program itself is treated as data and it is stored inside the memory (RAM) as data.

When the CPU requests next set of data this is transmitted serially and then is decoded later to see if it is data or program instructions.

This new way of computer architecture where **program and data** is stored in **memory** as **data** and then one at a time (**serially**), f**etch it**, **decode it** and **execute it** is commonly known as the **'Von Neumann' architecture**.

The illustration above shows the essential features of the **Von Neumann** or **stored-program** architecture.

## Memory

The computer will have enough memory that can hold both **data and** the **program** processing that data. In modern computers this memory is known as RAM.

## Control Unit

The control unit will manage the process of moving data and program into and out of memory and also deals with carrying out (executing) program instructions - **one at a time**. This includes the idea of a **'register'** to hold intermediate values. In the illustration on the previous page, the **'accumulator'** is one such register.

The **'one-at-a-time'** phrase means that the **von Neumann** architecture is a **sequential processing machine**.

## Registers

These components are special memory locations that can be accessed very fast. Some of these registers are: Instruction Register (IR), the Program Counter (PC), and the Accumulator.

## Input - Output

This architecture allows for the idea that a person needs to interact with the machine. Whatever values that are passed to and forth are stored once again in some **internal registers which are found inside the CPU.**

## Arithmetic Logic Unit

This part of the architecture is solely involved with carrying out calculations upon the data. All the usual Add, Multiply, Divide and Subtract calculations will be available but also data comparisons such as 'Greater Than', 'Less Than', 'Equal To' will be available.

To summarise the ALU performs:

- Arithmetic calculations
- Logic calculations
- Acts as a revolving door for data going in and out of the CPU

## Bus

Notice the arrows between components?
This implies that information should flow between various parts of the computer. In a modern computer built to the Von Neumann architecture, information passes back and forth along this system 'bus'.

## System Bus

A bus is a system of parallel wires connecting two or more components of a computer in order to bus signals (data) between them.

- There are buses to identify locations in memory – an 'address bus' and
- there are also buses to allow the flow of data - a 'data bus' or

- even to control the program instructions – the ‘control bus’.

https://www.youtube.com/watch?v=qgpKUAXuUQc

These three different busses we mentioned above make up what we know of the Von Neumann System. They’re also known as a system bus.

## Control bus

A microprocessor uses the control bus to regulate timing of the activity and to control the components. It carries the signals that report the status of the various devices when the memory is to read from or to write on or when the i/o devices are to read or write at specific times. Various operations are performed by microprocessor with the help of this control bus. This is a dedicated bus, because all timing signals are generated according to control signal.

The control bus is a bi-directional bus, meaning that signals can be carried in both directions. The data and address busses are shared by all components of the system. Control lines must therefore be provided to ensure that access to and use of the data and address buses by the different components of the system doesn’t lead to conflict.
The purpose of the control bus is to transmit command, timing and specific status information between system components.

**Control Bus lines include:**

- **Bus Request:** indicates that a device is requesting the use of the data bus
- **Bus Grant:** indicates that the CPU has granted access to the data bus
- **Memory Write:** causes data on the data bus to be written into the addressed location
- **Memory Read:** causes data from the addressed location to be placed on the data bus
- **Interrupt request**: indicates that a device is requesting access to the CPU
- **Clock:** used to synchronise operations

## Address Bus

Memory is divided up internally into units called words. A word is a fixed size group of digits, typically 16, 32 or 64 bits, which is handled as a unit by the processor, and different types of processor have different word sizes.

Each word in memory has its own specific address. The address bus transmits the memory addresses of words that are used as operands in program instructions, so that the data can be retrieved and sent back to the processor. When an instruction has been performed and the result is to be stored at a particular memory location, it is transmitted via the data bus.

The data bus is a group of wires or lines that are used to transfer the addresses of Memory or I/O devices. It is unidirectional. If an Address bus is 16 bits this means that it can transfer maximum 16-bit address which means it can address 65,536 different memory locations.

The address bus identifies particular memory locations inside the memory (RAM) or within the input / output devices. The width of the bus is identified by the number of lines; an 8bit bus will have 8 lines and these lines therefore define the maximum memory storage locations that exist. An 8-bit address buss can only service 256 memory locations although the 16-bit address bus can handle up to 64K memory locations

## Data Bus

As name tells that it is used to transfer data within Microprocessor and Memory/Input or Output devices. It is bidirectional as Microprocessor requires to send or receive data. Data bus is 8 Bits long. The word length of a processor depends on data bus, that's why Intel 8085 is called 8-bit Microprocessor because it has an 8-bit data bus. One line carries one bit and an 8 line data bus carries one byte at a time. The 16 bit database would carry two bytes at the same time.

## Are there any disadvantages with the Von Neumann Architecture?

The Von Neumann architecture has been very successful with most modern computers following this architecture but there are some problems with it and because of these problems, other architectures have been developed.

**Problem 1 –** (See diagram on the left)

Every piece of data and instruction has to pass across the data bus in order to move from main memory into the CPU (and back again). This is a problem because the data bus is a lot slower than the rate at which the CPU can carry out instructions. This is called the **'Von Neumann bottleneck'.** If nothing was done about it then the CPU would spend most of its time waiting around for instructions.

**How was this eliminated?**

A new special kind of memory called a **'Cache'** was introduced.

Think of the data bus as a bridge that can only carry so many instructions at a time. But what if we designed a 'holding area' on the CPU side of the bridge? Then we could store the most often-used instructions in the holding area instead of having to cross the bridge every time.

This holding area is called a '**cache**' (pronounced 'cash'). If the software programmer is clever enough, they will make it easier for the CPU to store the most-often used part of the code in the 'cache'. This activity is called '**code optimisation**'.

To illustrate what is meant by 'often' used, consider the small piece of pseudo code (i.e. generic code) below the data bus bridge:

The instruction 'Add 1 to a variable' could be stored in the cache and would not have to be fetched from main memory for every step.

**Problem 2 -** Both data and programs share the same memory space. This is a problem because it is quite easy for **a poorly written** or faulty piece of code to write data into an area holding other instructions, so causing trashing that program.

**Problem 3**

Another issue is that the **rate at which data** needs to be fetched and the rate at which **instructions** need to be fetched **are often very different**. And yet they share the same bottlenecked data bus.

To avoid this a new architecture was introduced and this is known as **The Harvard Architecture.**

The idea of the Harvard Architecture is to split the memory into two parts. One part for data and another part for programs. Each part is accessed with a different bus. This means the CPU can be fetching both data and instructions at the same time. There is also less chance of program corruption.

This architecture is **sometimes** used within the CPU between control unit and caches (which hold data / program), but it is less used with main memory (RAM) because of complexity and cost.

**Questions:**

What is the Von – Neumann machine?

What are the disadvantages of the introduction of the Von Neumann architecture?

How do we eliminate the Von-Neumann bottleneck? (Cache)

How do we make sure that data and program don’t overwrite one another?

Data and instructions need different rate of transmission. How can we eliminate this?

One feature of Von Neumann machine is the use of Fetch-Decode-Execute cycle. State TWO other features of Von Neumann architecture.

The last page about the Von Neumann architecture made a passing reference to registers. But what are these registers?

*"A register is a discrete memory location within the CPU designed to hold temporary data and/or instructions"*

A modern CPU will hold a number of registers. There are a number of general purpose registers that the programmer can use to hold intermediate results whilst working through a calculation or algorithm.

Then there are special-purpose registers designed to carry out a specific role. Each of these registers are given a name so that the programmer can write their software code to access them. Different manufacturers of CPU chips call them by different names (which makes life interesting for a professional programmer!)

But generically speaking these are:

- Program Counter (PC)
- Current Instruction Register (CIR)
- Program Status Word Register (PSW)
- Memory Address Register (MAR)
- Memory Data Register (MDR or MBR)

## Program Counter (PC)

This holds the address in memory of the next instruction.

For example, if the program counter has the address 305 then the next instruction will be at location 305 in main memory (RAM). When a program is running, the program counter will often just be incrementing as it addresses one instruction after the other, e.g. 305, 306, 307. However, the instructions will often modify the next address, for example, 305 becomes 39. What has happened is called a **'jump instruction'**. This is how the software programmer will cause different parts of his code to run depending on some condition e.g. a conditional IF statement. It is also how an interrupt routine is serviced. The program counter will be loaded with the starting address of the interrupt routine.

## Current Instruction Register (CIR)

This holds the current instruction to be executed, having been fetched from memory.

## Program Status Word Register (PSW)

The PSW has a number of duties all rolled into one.

- The Arithmetic Logic Unit compares two data items together, and it arranges for the result of that comparison to appear in this register i.e. the result of 'greater than' is resulted to TRUE or FALSE. The TRUE or FALSE flag will be displayed here.

- The PSW also indicates if program conditions have been met that would lead to a jump to a different part of the program. In programming terms this means the result of an IF statement is TRUE or FALSE. An IF statement is important in any programming language as it allows execution to jump from one set of instructions to another.

- The PSW also **holds error flags** that indicate a number of problems that may have happened as a result of an instruction, such as 'overflow' which means a calculation has exceeded it allowed number range.

A commonly used term is **'flag'**. This denotes a single binary bit within a register. They are often used to indicate a true or false condition.

## Memory Address Register (MAR)

Remember that data and program instructions have to fetch from memory? The memory address register, or MAR, holds the location in memory (address) of the next piece of data or program to be fetched (or stored).

## Memory Data Register (MDR) or Memory Buffer Register

When the **data** or **program instruction** is fetched from memory, it is temporarily held in the 'Memory Data Register' or **MDR** for short sometimes also called the Memory Buffer Register or **MBR**

*"A 'buffer' is a commonly used computer term to describe memory designed to hold data that is on its way to somewhere else".*

A memory buffer is a bit like the buffers on a train carriage, as the carriages connect with each other, the buffers will soak up the force. In memory, a bunch of data is absorbed quickly, then released at a controlled rate.

## FETCH - DECODE - EXECUTE

When software is installed onto a personal computer (most commonly from a CD-ROM, though other media or downloading from the internet is also common), the code that makes up the program and any library files is stored on the hard drive. This code comprises of a series of instructions for performing specific tasks, and data associated with these instructions. The code remains there until the user chooses to execute the program in question, on which point sections of the code are loaded into the computer's memory.

The CPU then executes the program from memory, processing each instruction in turn. Of course, in order to execute the instructions, it is necessary for the CPU to understand what the instruction is telling it to do. Therefore, recognition for instructions that could be encountered needs to be programmed into the processor. The instructions that can be recognized by a processor are referred to as an **'instruction set'**. The instructions sets are unique for each processor.

Once the instruction has been recognised and decoded the actions are then performed before the CPU proceeds on to the next instruction in memory. This is what we call the **Fetch** – **Decode** – **Execute cycle**.

Before we move into the **Fetch – Decode – Execute** let's have a look at the Instruction Set.

# What is an Instruction Set?

Each machine instruction is composed of two parts: the **op-code** and the **operand**. The image below shows the format of an instruction for a CPU. The first three bits represent the **op-code** and the final six bits represent the **operand**. The **middle bit** distinguishes between operands that are **memory addresses** and operands that are **numbers**. When the bit is set to '1', the operand represents a number. If it is a '0' it represents a memory address.

A simple set of machine instructions for our CPU are listed in the table below. Notice that all the op-codes are given an **English mnemonic** to simplify programming. Together these mnemonics are called an ***assembly language*.** Programs written in assembly language must be converted to their binary representation before the CPU can understand them. This usually done by another program called an assembler, hence the name.

| *Op-code* | *Mnemonic* | *Function* | *Example* |
|---|---|---|---|
| 001 | LOAD | Load the value of the operand into the Accumulator | LOAD 10 |
| 010 | STORE | Store the value of the Accumulator at the address specified by the operand | STORE 8 |
| 011 | ADD | Add the value of the operand to the Accumulator | ADD #5 |
| 100 | SUB | Subtract the value of the operand from the Accumulator | SUB #1 |
| 101 | EQUAL | If the value of the operand equals the value of the Accumulator, skip the next instruction | EQUAL #20 |
| 110 | JUMP | Jump to a specified instruction by setting the Program Counter to the value of the operand | JUMP 6 |
| 111 | HALT | Stop execution | HALT |

**A simple Low Level language**

In the low level language above, notice that some of the operands include a # symbol. This symbol tells the CPU that the operand represents a **number rather than a memory address**. Thus, when the assembler translates an instruction **with a # symbol,** the resulting machine code will have a '1' in the position of the number bit. **Notice** the central role that the Accumulator register plays. Nearly all the operations affect the value of this register since the **Accumulator acts as a temporary memory location for storing calculations in progress.** Let's take a look at one of these programs

**This program is called Sum.** It adds the numbers stored in two memory locations and places them in another memory location:

x= 2
y = 5
z = x+y

The variables x, y, and z correspond to the memory locations 13, 14, and 15 respectively. The instructions for the program are listed below. Read through the program, and then view the animation of this program by clicking the "View Animation" link or by typing the web address on your browser.

| *#* | *Machine code* | *Assembly code* | *Description* |
|---|---|---|---|
| 0 | 001 1 000010 | LOAD #2 | Load the value 2 into the Accumulator |
| 1 | 010 0 001101 | STORE 13 | Store the value of the Accumulator in memory location 13 |
| 2 | 001 1 000101 | LOAD #5 | Load the value 5 into the Accumulator |
| 3 | 010 0 001110 | STORE 14 | Store the value of the Accumulator in memory location 14 |
| 4 | 001 0 001101 | LOAD 13 | Load the value of memory location 13 into the Accumulator |
| 5 | 011 0 001110 | ADD 14 | Add the value of memory location 14 to the Accumulator |
| 6 | 010 0 001111 | STORE 15 | Store the value of the Accumulator in memory location 15 |
| 7 | 111 0 000000 | HALT | Stop execution |

**Sum program** [view animation]

http://courses.cs.vt.edu/~csonline/MachineArchitecture/Lessons/CPU/sumprogram.html

***The Fetch-Decode-Execute cycle*** *describes the basic steps a CPU carries out to process an instruction.*

The picture below shows the general internal set up of the CPU

We will look at each stage in turn over the next few pages

## Fetch

- The Program Counter copies the address of the next instruction it contains into the Memory Address Register (MAR).

- The Memory Address Register places the address to be used on to the 'Address Bus'

- The Memory Address Register triggers a 'read' signal that causes main memory (RAM) to place the instruction from that specific address on to the 'Data Bus'

- The instruction on the data bus is loaded into the Memory Data Register (also called Memory Buffer Register)

- The Memory Data register copies the instruction into the 'Instruction Register'

- **The Fetch stage is now complete**

## Decode

The CPU examines the instruction in the current instruction register (CIR) and 'decodes' it. This means a special part of the CPU called the 'decode' unit (Part of the control unit) will make the rest of the CPU ready to carry out the instruction. It does this by issuing a series of 'micro-instructions'.

For example the instruction might say 'Add'. The decode unit understands what this means and gets the system ready to carry that instruction out.

Every CPU has an **instruction set** that defines what the decoder understands as legitimate commands.

All software eventually end up as a set of commands which are extracted from within the instruction set.

## Execute then Reset

**Execute** - The instruction within the instruction register is carried out (executed) by the CPU. The part that executes instructions is called the 'execute unit'.

**Reset -** Now that the CPU is executing an instruction, the Program Counter can now be reset to point to the next instruction.

This is the Fetch-Decode-Execute cycle that is present in every sequential processing computer.

Now attempt the following activity:

### Fetch Execute – Fill in the Blanks - **How the CPU registers are used in the fetch-execute cycle.**

(NB Use full names not abbreviations)

### Fetch phase

1. The address of the next instruction is copied from the ________ to the Memory ________

2. It is then sent down the ________

3. The instruction held at that address is sent up the ________

4. And copied into the ________.

5. The contents of the ________ are copied to the ________

**Decode/Execute phase**

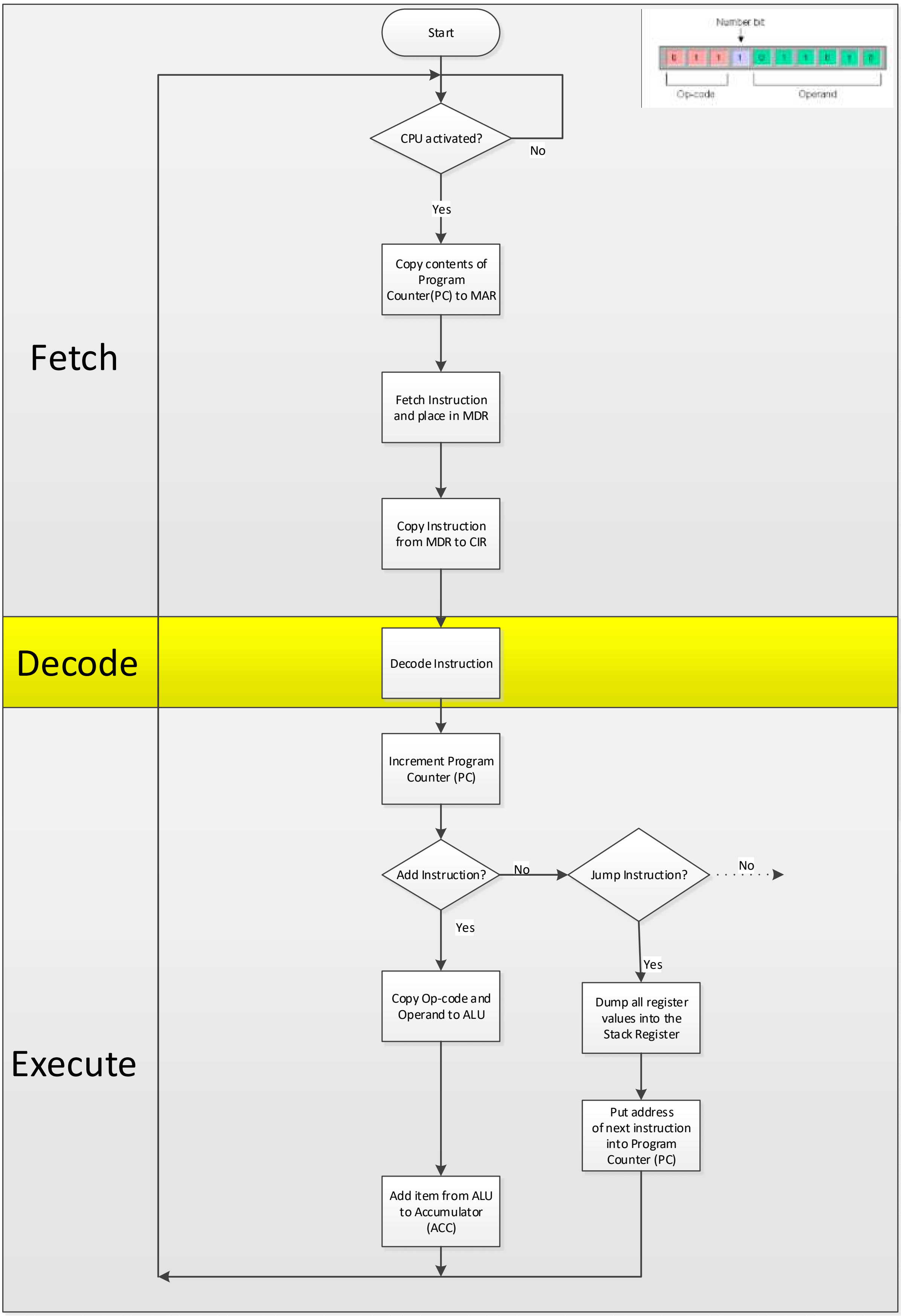
1. The instruction held in the ________ is ________

2. As soon as the instruction is decoded the contents of the ________ are incremented/reset so that it holds the address of the next instruction.

3. The decoded instruction is ________

The Fetch Execute Cycle below describes what happens when the mnemonic is ADD or JUMP.

.


Number bit
0
1
1
1
0
1
1
0
1
0
Op-code
Operand
Start
CPU activated?
No
Yes
Fetch
Copy contents of Program Counter(PC) to MAR
Fetch Instruction and place in MDR
Copy Instruction from MDR to CIR
Decode
Decode Instruction
Increment Program Counter (PC)
Add Instruction?
No
Jump Instruction?
No
Yes
Yes
Copy Op-code and Operand to ALU
Dump all register values into the Stack Register
Put address of next instruction into Program Counter (PC)
Execute
Add item from ALU to Accumulator (ACC)

Use the diagram below to complete the missing names from the registers

Q1. Describe the Fetch – Execute Cycle.

…………………………………………………………………………………………………………………

…………………………………………………………………………………………………………………

…………………………………………………………………………………………………………………

…………………………………………………………………………………………………………………

…………………………………………………………………………………………………………………

……….[4]

Q2. Describe how a jump instruction is executed.

…………………………………………………………………………………………………………………

…………………………………………………………………………………………………………………

…………………………………………………………………………………………………………………

…………………………………………………………………………………………………………………

…………………………………………………………………………………………………………………

……..….[2]

Q3. Identify/Name the component of the CPU that carries out the following tasks:

| | Name of Task | Component (hidden) |
|---|---|---|
| 1. | Generates control/timing signals | Control |
| 2. | Carries out Logical Operations such as: AND, OR, NOT | ALU |
| 3. | Controls decoding/execution of instructions | Control |
| 4. | Execution of instructions | ALU |
| 5. | Mathematical operations (* / + - etc.) | ALU |
| 6. | Small amount of very fast memory | Registers |
| 7. | Holds the address of next instruction to be executed | PC |
| 8. | Decides which way data is going (IN/OUT) | Control |
| 9. | Holds instruction while it is decoded | CIR |
| 10. | Holds result of ALU operations | ACC - Accumulator |
| 11. | Example of another general purpose register | Stack Pointer or Flag Register |

Q4. Describe the effects of the Fetch – Decode – Execute cycle on the Program Counter (PC) and the Memory Address Register (MAR).

…………………………………………………………………………………………………………………

…………………………………………………………………………………………………………………

…………………………………………………………………………………………………………………

…………………………………………………………………………………………………………………

…………………………………………………………………………………………………………………

…………………………………………………………………………………………………………………

…………………………………………………………………………………………………………………

……………...[5]

# OTHER MACHINE ARCHITECTURES

The Von Neumann architecture is a **sequential** machine architecture. This has many disadvantages and one of them is that you can’t process more than one instruction at the same time. In order to make the CPU work more efficiently we introduced **parallel** processing.

In order to eliminate this problem, we need to look for an alternative machine architecture that can deal with more instructions at the time. There are a number of alternative **parallel processing** machine architectures to select from and each one has its own merits:

- Pipelining
- Array processors
- Multiple processors

## Pipelining

Every CPU carries out the Fetch-Decode-Execute cycle. The idea of the pipeline is to split/break this cycle into three or more hardware processing paths within the CPU called **a 'pipeline'.** By dealing with the instructions in this way the processor doesn’t waste time waiting for the next execution, decoding or fetching to be carried out. This way the processing power because is carried out in parallel is more effective. If you look at the diagram below the CPU is used effectively during all stages; when pipeline 1 is doing fetching, pipeline 2 in parallel is doing the decoding and pipeline 3 in parallel again is doing the executing.

At any given time, whilst a fetch operation is taking place on pipeline 1 which occupies the data and address buses, pipeline 2 is decoding an instruction and pipeline 3 is executing an instruction. As long as the pipelines can be kept full, we are making maximum use of the CPU.

The **advantages** of using such an architecture:

- The CPU is making efficient use of resources e.g. whilst it is dealing with the Fetch instruction the ALU is idle.
- Quicker time of execution of large number of instructions

The **disadvantage** with this type of processing is:

- It **cannot handle jump or interrupts statements** as the pipeline needs to be flushed(emptied) in order to deal with the interrupt instruction.
- This kind of computer architecture is classified as a **Single Instruction on Single Data** computer or **SISD.**
- **It is very difficult to synchronise** – Requires sophisticated program execution techniques e.g. organise the hardware as more than one operation is carried out at the same time
- **Needs to use faster technology** – Faster CPU?

So in conclusion you can see that for a fixed clock speed a cpu with pipelining performs faster than CPU without pipelining.

# Array or Vector Processing - Single Instruction Multiple Data (SIMD)

Some types of data can be processed independently of one another. A good example of this is the simple processing of pixels on a screen. If you wanted to make each coloured pixel a different colour according to what it currently holds. For example, "Make all Red Pixels Blue", "Make Blue Pixels Red", "Leave Green pixels alone".

A sequential processor (von Neumann type) would examine each pixel one at a time and apply the processing instruction. You can also arrange for data such as this to be an array. The simplest array looks like this:

{element 1, element 2, ...} this is called a '1 dimensional array' or a 'vector'

A slightly more complicated array would have rows and columns:
{element 1, element 2
element 3, element 4}

This is called a '2 dimensional' array or 'matrix'. The matrix is fundamental to graphics work.

***An array processor** (or vector processor) carries out a single instruction but on multiple data; one processor but with **a number of Arithmetic Logic Units (ALU)** that allows all the elements of an array of **data to be processed at the same time**.* The illustration on the right shows the architecture of an array or vector processor

With an array processor, a single instruction is issued by a control unit and that a **single instruction** is applied to a **number of data storage** location sets at the same time. Like this the processor is executing the one instruction on four different locations at the same time and therefore will execute the instruction much quicker than a sequential processor.

An array processor is a **Single Instruction on Multiple Data** computer or **SIMD.** You will find games consoles and graphics cards making heavy use of array processors to shift those pixels about. This **type of processor is ideal for weather forecasting/airflow simulation on a new aircraft** etc.

## Limitations of Array Processing

This architecture relies on the fact that the data sets are all acting on a single instruction. However, if these data sets somehow rely on each other then you cannot apply parallel processing. For example, if data A has to be processed before data B then you cannot do A and B simultaneously. **This dependency is what makes parallel processing difficult to implement.** This is why sequential machines are still extremely useful.

Let's consider an example of a simple multiplication of a series of numbers 1, 9, 2 and 8, all of which need to be multiplied by the number 3. An ordinary CPU without array processing will need to load the 4 numbers one at a time (4 Loads), multiply each one by 3 (4 times) and save the result of each multiplication (4 times). There will be 12 instructions altogether to be carried out. Using an **array processor,** we cut down the number of instructions quite considerably as one instruction of load, multiply and save is applied at the same time to 4 different registers.

Q1. Describe the term array processor.

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………[2]

Q2. Give an example of the type of task for which an array processor is most suitable.

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………[2]

# Multiple Processors

Moving on from **an** array processor, where a single instruction acts upon multiple data sets and the next level of parallel processing is to have **multiple** processors that carry out many instructions by acting upon multiple data sets.

This is achieved by having a number of CPUs being applied to a single problem, with each CPU carrying out only part of the overall problem.

A good example of this architecture is a **supercomputer**. For example the massively parallel IBM Blue Gene supercomputer that has **4,098 processors,** allowing for 560 TeraFlops of processing. This is applied to problems such as predicting climate change or running new drug simulations. It is an ideal architecture to use for large problems that can be broken down into smaller sub-problems.

We have something similar but in a smaller scale at home. Our home computers now have multiple cores. For example the Intel **Core Duo** has two CPUs (called 'cores') inside the chip, whilst the **Quad core** has four. A multi-core computer is a '**Multiple Instruction Multiple Data**' computer or **MIMD**

## Limitations of multi-core processing

This architecture is dependent on being able to cut a problem down into chunks, each chunk can then be processed independently. But not many problems can be broken down in this way and so it remains a less used architecture.

Furthermore, the software programmer has to write the code to take advantage of the multi-core CPUs. This is actually quite difficult and even now most applications running on a multi-core CPU such as the Intel 2 Duo will not be making use of both cores most of the time.

# Summary of Parallel Processing

There are a number of ways to carry out parallel processing, the table below shows each one of them and how they are applied in real life.

| | | |
|---|---|---|
| Pipeline | Single Instruction Single Data (SISD) | Inside a CPU |
| Array Processor | Single Instruction Multiple Data SIMD | Graphics cards, games consoles |
| Multi-Core | Multiple Instruction Multiple Data MIMD | Super computers, modern multi-core chips |

**Advantages of parallel processing** over the **Von Neumann** architecture

- Faster when handling large amounts of data, with each data set requiring the same processing (array and multi-core methods)
- Is not limited by the bus transfer rate (the Von Neumann bottleneck)
- Can make maximum use of the CPU (pipeline method) in spite of the bottleneck

**Disadvantages**

- Only certain types of data are suitable for parallel processing. Data that relies on the result of a previous operation cannot be made parallel. For parallel processing, each data set must be independent of each other.
- Costlier in terms of hardware - multiple processing blocks needed, this applies to all three methods

Q1. Describe parallel processing.

……………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………

……………………………[5]

Q2. Describe one advantage of a parallel processor compared with a single processor system.

……………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………

……………………………………………………………………………[2]

# Co-processor (Maths co-processor)

So far, we have discussed parallel processing as a means of speeding up data processing. This is fine but it does make an assumption that the Arithmetic Logic Unit (ALU) within the CPU is perfect for handling all kinds of data but this is not always true.

There are two basic ways of doing calculations within a CPU:

a) Integer maths which only deals with whole numbers or

b) floating point maths which can deal with decimal / fractional numbers.

Handling floating point numbers efficiently requires wide registers to deal with a calculation in one go but the CPU architect may not want to dedicate precious hardware space in his CPU for these wider registers.

So the idea of a **'Maths co-processor'** came about. A co-processor is especially designed to carry out floating point calculation extremely quickly. It **co-exists with the CPU on the motherboard.** Whenever a floating point calculation needs to be done, the CPU hands the task over to the co-processor then carries on with doing something else until the task is complete.

The **advantage** of having a co-processor is that calculation (and hence performance) is much faster.

The **disadvantage** is that it is **more expensive**, requires **more motherboard space** and takes **more power**. On top of this Co-processors often **require tasks to be handed to them** by the central processor.

But if the computer is dedicated to handling heavy floating point work then it may be worth it. For instance a computer within a signal processing card in a communication system may include a maths co-processor to process the incoming data as quickly as possible.

Operations performed by the coprocessor may be floating point arithmetic, graphics, signal processing, string processing, and encryption. By offloading processor-intensive tasks from the main processor, coprocessors can accelerate the system performance. Many computer manufactures allow their customers to customise their computers by opting to have a coprocessor or not so that those who do not need the extra performance don't need to pay for it.

Q1. In some computer systems a co-processor may be used. Explain the term co-processor.

..........................................................................................................................................................

..........................................................................................................................................................

..........................................................................................................................................................

..........................................................................................[3]

Q3. Explain with an aid of an example, the following statement: 'A co-processor is a simple form of parallel processor'

..........................................................................................................................................................

..........................................................................................................................................................

..........................................................................................................................................................

..........................................................................................................................................................

..................................................................................................................................................[2]

Q4. Discuss the use of different computer architectures for different problem solutions. (the quality of your written communication will be assessed in your answer to this question)

..........................................................................................................................................................

..........................................................................................................................................................

..........................................................................................................................................................

..........................................................................................................................................................

..........................................................................................................................................................................

..........................................................................................................................................................................

..........................................................................................................................................................................

..........................................................................................................................................................................

..........................................................................................................................................................................

..........................................................................................................................................................................

..........................................................................................................................................................................

..........................................................................................................................................................................

..........................................................................................................................................................................

..........................................................................................................................................................................

..........................................................................................................................................................................

..........................................................................................................................................................................

..........................................................................................................................................................................

..........................................................................................................................................................................

..........................................................................................................................................................................

..........................................................................................................................................................................

..........................................................................................................................................................................

..........................................................................................................................................................................

..........................................................................................................................................................................

..........................................................................................................................................................................

..........................................................................................................................................................................

..........................................................................................................................................................................

..........................................................................................................................................................................

..........................................................................................................................................................................

.......................................................................................................................................[8]

# CISC VS RISC COMPUTERS

**CISC** stands for **C**omplex **I**nstruction **S**et **C**omputers and RISC stands for **R**educed **I**nstruction **S**et **C**omputer and they represent two lines of thought when designing a new computer chip. They are fundamentally two different approaches to microprocessor design.

Both of the above instruction sets are aiming to improve the way instructions are handled/carried out. There is no clear answer in terms of which one is better as it depends on the preference of the chip manufacturers. Some of them think that it is much better to add more complex **hardware circuits** on the motherboard so they execute more complex but less instructions which is requiring less cycles and some others prefer to reduce hardware circuits at the expense of asking the software to carry out more simple instructions but in more cycles.

The time you take to execute a program is decided by the time you take to execute one cycle, the cycles you need to carry out one instruction and the number of instructions your program has.

This makes the chip manufacturing a difficult one when it comes to which approach to choose.

Most of them until recently had opted for the **CISC** approach. Each generation of their chips offered larger and richer instruction sets compared to the one before. **CISC** designers believed CPUs can be made quicker by adding more and more complexity into the instructions of the instruction set. The aim was to perform as much work in a single instruction as possible. Hence instruction sets grew larger and complicated.

**RISC** designers believed that the best performance can be achieved by reducing the time taken to execute any given instruction. Rather than have complex instructions that require many clock cycles to complete, RISC chips use very simple instructions that could be performed in fewer clock cycles. Performance can then be improved by making the cycles shorter.

**CISC** instructions can usually address memory in many different ways. This builds complexity into the instruction and also means that a given instruction **op-code can be of variable size**. RISC instructions, on the other hand, are usually limited to a single memory address. In fact, a special set of instructions (called ***load*** and ***store*** instructions) are designed to read and write from memory, transferring data to and from registers as required. The result of this so called Load-Store RISC architecture is that instructions are less complicated and, as a bonus, **tend to be of fixed size.** This makes performance-optimising strategies, such as **pipelining**, **easier to implement.**

The **RISC** approach recently seems to be the favourite one. The reason is highlighted above and illustrated in an example given below:

### Multiplying Two Numbers in Memory

The diagram on the right is representing the storage scheme for a generic computer. The main memory is divided into locations numbered from **(row) 1: (row) 6** and **(column) 1: (column) 4**. The execution unit is responsible for carrying out all computations. However, the execution unit can only operate on data that has been loaded into one of the six registers (A, B, C, D, E, or F). Let's say we want to find the product of two numbers - one stored in location **2**:**3** and another stored in location **5**:**2** - and then store the product back in the location **2**:**3**.

### The CISC Approach

The primary goal of **CISC** architecture is to complete a task in as few lines of assembly as possible. This is achieved by **building processor hardware that is capable of understanding and executing a series of operations.** For this particular task, a **CISC** processor would come prepared with a specific instruction (we'll call it "MULT"). When executed, this instruction loads the two values into separate registers, multiplies the operands in the execution unit, and then stores the product in the appropriate register. Thus, the entire task of multiplying two numbers can be completed with one instruction:

**MULT** 2:3, 5:2

MULT is what is known as a "**complex instruction**." It operates directly on the computer's memory banks and does not require the programmer to explicitly call any loading or storing functions **thus requires less registers.** It closely resembles a command in a higher level language. For instance, if we let "a" represent the value of **2**:**3** and "b" represent the value of **5**:**2**, then this command is identical to the C statement "**a = a * b."**

One of the primary advantages of this system is that the compiler has to **do very little work to translate a high-level language statement into assembly**. Because the **length of the code is relatively short**, very **little RAM is required to store instructions**. The emphasis is put on building complex instructions directly into the hardware.

### The RISC Approach

RISC processors only use simple instructions that can be executed within one clock cycle. Thus, the "MULT" command described above could be divided into three separate commands: "LOAD," which moves data from the memory bank to a register, "PROD," which finds the product of two operands located within the registers, and "STORE," which moves data from a register to the memory banks. In order to perform the exact series of steps described in the CISC approach, a programmer would need to code four lines of assembly:

**LOAD** A, **2**:**3**<br>
**LOAD** B, **5**:**2**<br>
**PROD** A, B<br>
**STORE** **2**:**3**, A

At first, this may seem like a much less efficient way of completing the operation. Because there are **more lines of code**, **more RAM is needed** to store the assembly level instructions. The **compiler must also perform more work** to convert a high-level language statement into code of this form.

However, the RISC strategy also brings some very important advantages. Because each instruction requires only one clock cycle to execute, the entire program will execute in approximately the same amount of time as the multi-cycle "**MULT**" command. These RISC "reduced instructions" require **less transistors of hardware space** than the complex instructions, leaving more room for general purpose registers. Because all of the instructions execute in a uniform amount of time (i.e. one clock), **pipelining is possible**.

Separating the "LOAD" and "STORE" instructions actually reduces the amount of work that the computer must perform. After a CISC-style "MULT" command is executed, **the processor automatically erases the registers. If one of the operands needs to be used for another computation, the processor must re-load the data from the memory bank into a register.** In RISC, the operand will remain in the register until another value is loaded in its place.

**The Performance Equation**
The following equation is commonly used for expressing a computer's performance ability:

The **CISC** approach attempts to minimize the number of instructions per program, sacrificing the number of cycles per instruction. **RISC** does the opposite, reducing the cycles per instruction at the cost of the number of instructions per program.

### RISC Roadblocks

Despite the advantages of **RISC** based processing, **RISC** chips took over a decade to gain a foothold in the commercial world. This was **l**argely due to a lack of software support. It was because of this companies were reluctant to invest on the RISC based processing.

Although Apple's Power Macintosh line featured RISC-based chips and Windows NT was RISC compatible, Windows 95 and Windows 98 were designed with CISC processors in mind. Many companies were **unwilling** to take a chance with the emerging RISC technology. Without commercial interest, processor developers were unable to manufacture RISC chips in large enough volumes to make their price competitive.

### The Overall RISC Advantage

Today, the Intel x86 is arguable the only chip which retains CISC architecture. This is primarily due to advancements in other areas of computer technology. The price of RAM has decreased dramatically. Compiler technology has also become more sophisticated, so that the RISC use of RAM and emphasis on software has become ideal.

## Comparison

| Complex Instruction Set Computers (CISC) | Reduced Instructions Set Computers (RISC) |
|---|---|
| Has more complex hardware | Has simpler hardware |
| More compact software code | More complicated software code |
| High instruction cycles/second may require more data shuffling and possible page faults with virtual memory. | Low cycles per instruction |
| Instruction format are of variable length – pipelining is difficult to implement. | Instruction format are of fixed length (Only LOAD and STORE) – pipelining is easier to implement. |
| Uses less RAM as no need to store intermediate results | Need to use more RAM to handle intermediate results |
| Less compiling time | More compiling time is required as there are more lines to be translated. |
| Many addressing modes | Less addressing modes |
| Uses less registers | Uses more registers |
| Can't be pipelined due to the variable format | Can be pipelined |

**Exercises**

**Q1.** Two computer architectures are Reduced Instruct ion Set Computer (RISC) and Complex Instruction Set Computer (CISC) architectures. Complete the table to show how the statements apply to these architectures. (tick the appropriate boxes) [4]

| | RISC only | CISC only | Both RISC and CISC |
|---|---|---|---|
| Has many addressing modes | | | |
| Many instructions available | | | |
| Uses one or more register sets | | | |

| | | | |
|---|---|---|---|
| Uses only simple instructions | | | |

**Q2.** Compare the number of machine cycles used by RISC and CISC to complete a single task.

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

………………………………………………………………………………[2]

**Q3.** State three features of a Complex Instruction Set Computer (CISC) architecture.

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

………………………………………………………………………………[3]

**Q4.** Explain one advantage and one disadvantage, other than cost, of a CISC architecture compared with a Reduced Instruction Set Computer (RISC) architecture.

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

………………………………………………………………………………[2]