Computer Science

S3 Exam Revision Notes

# **Computer Structure**

## 5 box diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | Main Memory |  |  |
|  |  |  |  |  |
| Input Devices |  | Central Processing Unit |  | Output Devices |
|  |  |  |  |  |
|  |  | Backing Storage |  |  |

## Central Processing Unit (CPU)

This is the brain of the computer system and carries out all instructions. It consists of three different parts:

**Arithmetic & Logic Unit** – carries out all calculations and makes decisions based on comparisons

**Registers** – temporarily stores any data that the processor is working on

**Control Unit** – in charge of the CPU; makes sure that instructions are run in the correct order

## Main Memory

This is the area where a computer stores data until it is needed by the processor. It consists of two different parts:

**Random Access Memory (RAM)** – data is held here temporarily; when the PC is switched off any data is lost

**Read Only Memory (ROM)** – data is held here permanently; when the PC is switched off any data will be saved

## Backing Storage

Backing storage is the name given to devices that allow data to be stored permanently. There are three different types:

**Optical** – makes use of lasers to read and write data

**Magnetic** – makes use of electromagnets to read and write data

**Solid-state** – makes use of microchips to read and write data; has no moving parts

## Types of Devices

**Input** – sends data to a computer system e.g. keyboard sends text, camera sends an image, microphone sends sound

**Output** – display information from a computer system to the user e.g. monitor allows user to see information, speakers allow a user to hear information

# **Data Representation**

## Binary

This is the only language that a computer system can understand. It is made up of 1s and 0s.

## Bitmap Images

Every pixel in a bitmap image is stored using a certain number of bits. A black pixel is represented by a 1. A white pixel is represented by a 0.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | Bitmap graphic Represented as | 1 | 0 | 0 | 0 | 0 | 1 |
|  |  |  |  |  |  | 1 | 0 | 1 | 0 | 0 | 0 |
|  |  |  |  |  |  | 1 | 0 | 1 | 0 | 1 | 0 |
|  |  |  |  |  |  | 1 | 1 | 1 | 0 | 1 | 0 |
|  |  |  |  |  |  | 1 | 1 | 1 | 0 | 1 | 0 |
|  |  |  |  |  |  | 1 | 0 | 1 | 0 | 1 | 0 |
|  |  |  |  |  |  | 1 | 0 | 1 | 0 | 0 | 0 |
|  |  |  |  |  |  | 1 | 0 | 0 | 0 | 0 | 1 |

## Binary Numbers

These consist of a certain combination of bits that can represent denary numbers (fancy name for numbers that humans can understand). They are represented in the following way:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **128** | **64** | **32** | **16** | **8** | **4** | **2** | **1** |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

To convert a **binary** number into denary, you need to make sure you add the numbers above the 1s:

128 + 4 = 132

To convert a **denary** number into binary, you work in the following way: Example – 57

Does 128 fit into 57? No - 0

Does 64 fit into 57? No - 0

Does 32 fit into 57? Yes - 1 Now subtract 32 from 57 and continue with 25

Does 16 fit into 25? Yes - 1 Now subtract 16 from 25 and continue with 9

Does 8 fit into 9? Yes - 1 Now subtract 8 from 9 and continue with 1

Does 4 fit into 1? No - 0

Does 2 fit into 1? No - 0

Does 1 fit into 1? Yes - 1

Double check that it adds up to 57 to ensure you have the correct answer.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **128** | **64** | **32** | **16** | **8** | **4** | **2** | **1** |  | **Add** |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |  | 57 |

## Storage Units

8 bits = 1 **byte**

1024 bytes = 1 **kilobyte**

1024 kilobytes = 1 **megabyte**

1024 megabytes = 1 **gigabyte**

1024 gigabytes = 1 **terabyte**

## File Sizes

To convert a file size from **gigabytes** to bits, you need to complete the following steps:

1. Multiply the number of gigabytes by 1024 to get the number of megabytes
2. Multiply the number of megabytes by 1024 to get the number of kilobytes
3. Multiply the number of kilobytes by 1024 to get the number of bytes
4. Multiply the number of bytes by 8 to get the number of bits

Example: 2 gigabytes

1. 2 x 1024 = 2048 megabytes
2. 2048 x 1024 = 2097152 kilobytes
3. 2097152 x 1024 = 2147483648 bytes
4. 2147483648 x 8 = 17179869184 bits

To convert a file size from **bits** to gigabytes, you need to complete the following steps:

1. Divide the number of bits by 8 to get the number of bytes
2. Divide the number of bytes by 1024 to get the number of kilobytes
3. Divide the number of kilobytes by 1024 to get the number of megabytes
4. Divide the number of megabytes by 2014 to get the number of gigabytes

Example: 25769803776 bits

1. 25769803776 ÷ 8 = 3221225472 bytes
2. 3221225472 ÷ 1024 = 3145728 kilobytes
3. 3145728 ÷ 1024 = 3072 megabytes
4. 3072 ÷ 1024 = 3 gigabytes

## File Types

Different types of files are used to store different types of media. The table below shows the file types you need to know about.

|  |  |  |
| --- | --- | --- |
| **Type of media** | **File format** | **Stands for** |
| ***Text*** | RTF | Rich Text Format |
| TXT | Text |
| ***Image*** | JPEG | Joint Photographic Experts Group |
| PNG | Portable Network Graphics |
| GIF | Graphic Interchange Format |
| ***Video*** | AVI | Audio Video Interleave |
| MP4 | MPEG – Video Layer 4 |
| ***Sound*** | MP3 | MPEG – Audio Layer 3 |
| WAV | Waveform Audio File |

# **Databases**

## Manual

A manual database is one that is paper-based. Examples include Yellow Pages, Argos catalogue etc.

## Electronic

An electronic database is stored on a computer. It has many advantages over a paper-based database including:

* Takes up less storage space
* Quicker to search for and sort information
* Easier to change/edit data that is stored
* Easier to move data eg send via email or storage device instead of carrying a filing cabinet

## Flat File

A flat file database stores all data in one table.

## Field

A field is one piece of information within a table. These can be identified by the different **headings** at the top of each column. There are different types of fields:

* Text – stores words
* Number – stores digits
* Boolean – stores yes/no values
* Date – stores date formats e.g. DD/MM/YYYY
* Time – stores time formats e.g. HH/MM/SS, 00.00.00
* Attachment – stores a picture or file

## Record

A record is one completed set of fields within a table. These can be identified by the number of **completed rows** in a table.

## Searching

A **query** is used to find information within a database. It is made up of the **fields** you want to search and the **criteria** (the specific information you want to find).

## Sorting

Database tables can be sorted by fields in two different orders: ascending or descending.

**Ascending** – lowest to highest, A – Z, 0 – 9.

**Descending** – highest to lowest, Z – A, 9 – 0.

## Relational

A relational database stores data in **two or more** linked tables. The have several advantages over using a flat file database:

* Data only needs to be entered once
* Data is easier to edit – one change is automatically updated everywhere else it appears where as a flat file requires you to change it yourself
* Saves time

## Keys

A **primary key** is used to ensure that every record is unique.

A **foreign key** is used to link two tables together.

## Example 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Club ID** | **Club Name** | **Location** | **Price (£)** | **Indoors?** | **Teacher** |
| C011 | Table Tennis | Building | 10 | Yes | A. O’Neil |
| C012 | Table Tennis | Hall | 10 | Yes | B. Weston |
| C003 | Football | Arena | 15 | No | D. Carson |
| C004 | Basketball | Hall | 12 | Yes | A. O’Neil |

In the table above, there are **6** fields and **4** records.

The field types would be as follows:

* Club ID – Text
* Club Name – Text
* Location – Text
* Price (£) – Number
* Indoors? – Boolean
* Teacher – Text

It is sorted by the field **Club Name** in a **descending** order (every other field changes order).

The primary key would be **Club ID** because it is a different value (unique) for every row. It is possible that:

* two or more clubs share the same name
* two or more clubs have the same location
* two or more clubs cost the same price
* two or more clubs are indoors
* two or more clubs are taught by the same person

If you designed a query to search for all clubs that are in indoors it would be:

|  |  |  |
| --- | --- | --- |
| **Field Name** | Club Name | Indoors? |
| **Criteria** |  | “Yes” |

## Example 2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Player ID** | **First Name** | **Surname** | **Squad No.** | **Club ID** | **Club Name** | **Stadium** |
| P101 | James | McGinn | 2 | C100 | Rovers | Town Park |
| P201 | Graeme | Rooney | 7 | C100 | Rovers | Town Park |
| P103 | Darren | May | 2 | C201 | United | ABC Arena |
| P102 | Simon | Shinnie | 10 | C201 | United | ABC Arena |
| P001 | Sean | Ball | 18 | C201 | United | ABC Arena |
| P200 | Billy | Logan | 7 | C102 | Central | Rose Lane |

In the table above there are **7** fields and **6** records.

The Club ID, Club Name and Stadium contain information that is repeated more than once. To remove this, we can split the table into two separate tables: **Player** and **Club**.

|  |
| --- |
| **Player Table** |
| **Player ID** | **First Name** | **Surname** | **Squad No.** | **Club ID** |
| P101 | James | McGinn | 2 | C100 |
| P201 | Graeme | Rooney | 7 | C100 |
| P103 | Darren | May | 2 | C201 |
| P102 | Simon | Shinnie | 10 | C201 |
| P001 | Sean | Ball | 18 | C201 |
| P200 | Billy | Logan | 7 | C102 |

|  |
| --- |
| **Club Table** |
| **Club ID** | **Club Name** | **Stadium** |
| C100 | Rovers | Town Park |
| C201 | United | ABC Arena |
| C102 | Central | Rose Lane |

The primary key for the Player table is **Player ID** because it is unique.

The primary key for the Club table is **Club ID** because it is unique (it is possible for teams from other countries to share the same name; it is possible for two teams to share a stadium).

A foreign key has to be a primary key in another table – in this case **Club ID** is used as the foreign key to link the two tables together (this field appears in both tables).

If you designed a query to search for the first name and surname of all players who played with Rovers it would be:

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | First Name | Surname | Club Name |
| **Table Name** | Player | Player | Club |
| **Criteria** |  |  | Rovers |

# **Web Design**

## HTML

Stands for **Hyper Text Markup Language**. Used to create webpages. HTML files will consist of different **<tags>.** Each tag is **opened** and then **closed** e.g. <title>…</title>

|  |  |
| --- | --- |
| **Tag** | **What it does** |
| Html | Indicates the start of a html document |
| Head | One of two sections in a html document; stores any information about the page |
| Title | Creates a name for the webpage that will appear at the tab section |
| Body | One of two sections in a html document; stores any information that the page should display |
| P | Creates a new paragraph that starts on a new line |
| B | Makes text appear in bold |
| I | Makes text appear in italics |
| U | Makes text appear underlined |
| A | Creates a hyperlink e.g. <a href=”https://www.google.co.uk/”> Click for Google </a> |
| Img | Creates an image e.g. <img src=”braidhurst.jpg” style=”height:200px; width:300px;> |
| H1 | Creates the biggest heading size |
| H2 | Creates the second biggest heading size |
| H3 | Creates the third biggest heading size |
| H4 | Creates the fourth biggest heading size |
| H5 | Creates the fifth biggest heading size |
| H6 | Creates the sixth biggest heading size |

## Wireframe

Used to design the **layout** of a webpage. It will indicate where each different section should be displayed on a webpage.

An example wireframe of the Google homepage is shown below.





Lucky Button

Search Button

Search Bar

Google.jpg

## Hyperlinks

There are two types of hyperlinks that you need to know about: **internal** and **external**.

An internal hyperlink will take you to a **different webpage** but keep you on the **same website**.

An external hyperlink will take you to a **different website** altogether.



The Google page above shows search results for Braidhurst.

If we click on any results then we will be redirected to a different website – we will no longer be on the Google website. These are examples of **external hyperlinks**.

If we click on Maps, Images, News, Shopping or More then this will link to a different page within the Google website. These are examples of **internal hyperlinks**.