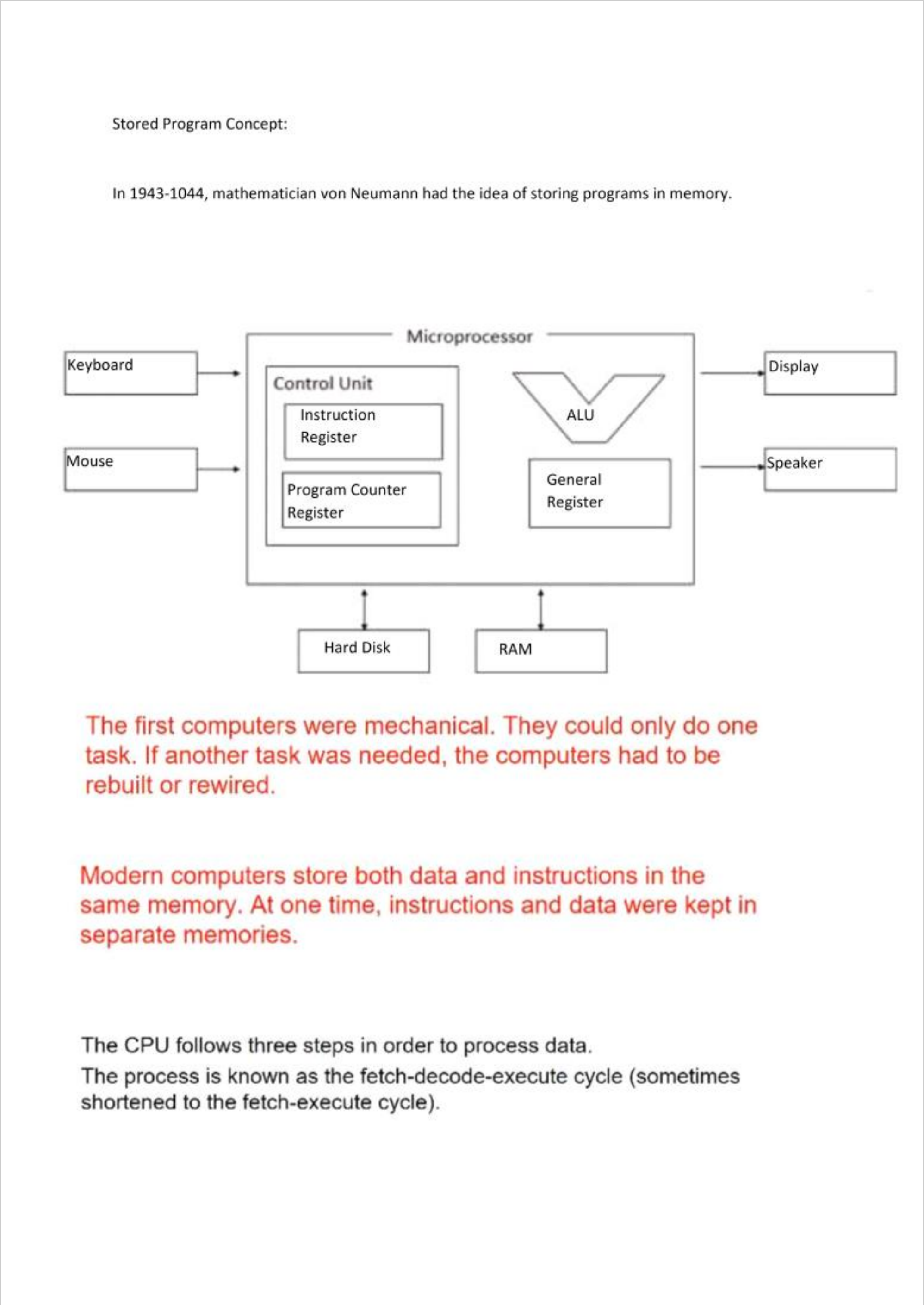
Lesson 1 - Stored Process Concept

20 January 2021 10:27



Stored Program Concept:

In 1943-1044, mathematician von Neumann had the idea of storing programs in memory.

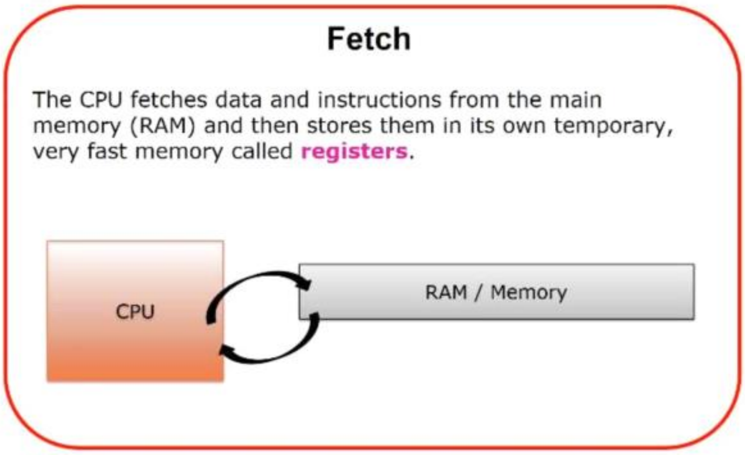
Microprocessor
Keyboard
Mouse
Control Unit
Instruction Register
Program Counter Register
ALU
General Register
Display
Speaker
Hard Disk
RAM

The first computers were mechanical. They could only do one task. If another task was needed, the computers had to be rebuilt or rewired.

Modern computers store both data and instructions in the same memory. At one time, instructions and data were kept in separate memories.

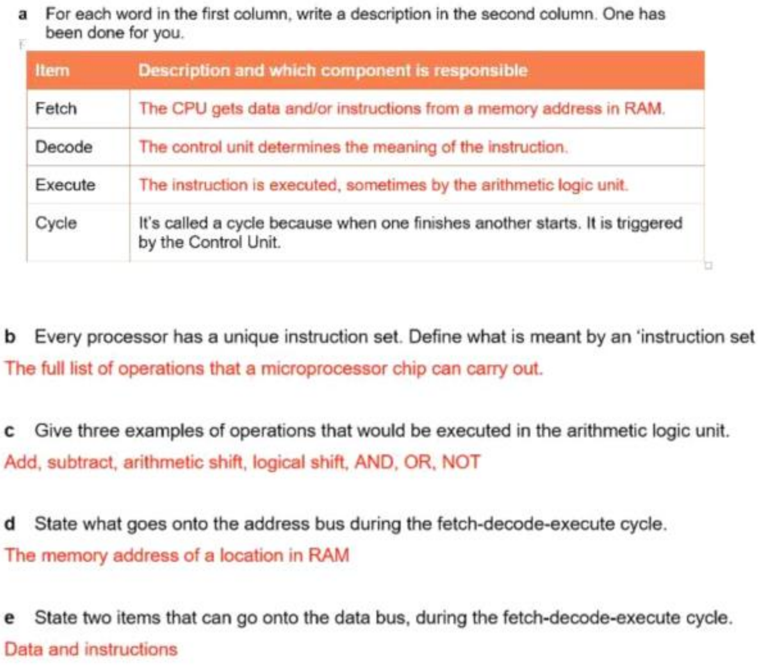
The CPU follows three steps in order to process data.

The process is known as the fetch-decode-execute cycle (sometimes shortened to the fetch-execute cycle).



Fetch

The CPU fetches data and instructions from the main memory (RAM) and then stores them in its own temporary, very fast memory called **registers**.



a For each word in the first column, write a description in the second column. One has been done for you.

Item	Description and which component is responsible
Fetch	The CPU gets data and/or instructions from a memory address in RAM.
Decode	The control unit determines the meaning of the instruction.
Execute	The instruction is executed, sometimes by the arithmetic logic unit.
Cycle	It's called a cycle because when one finishes another starts. It is triggered by the Control Unit.

b Every processor has a unique instruction set. Define what is meant by an 'instruction set

The full list of operations that a microprocessor chip can carry out.

c Give three examples of operations that would be executed in the arithmetic logic unit.

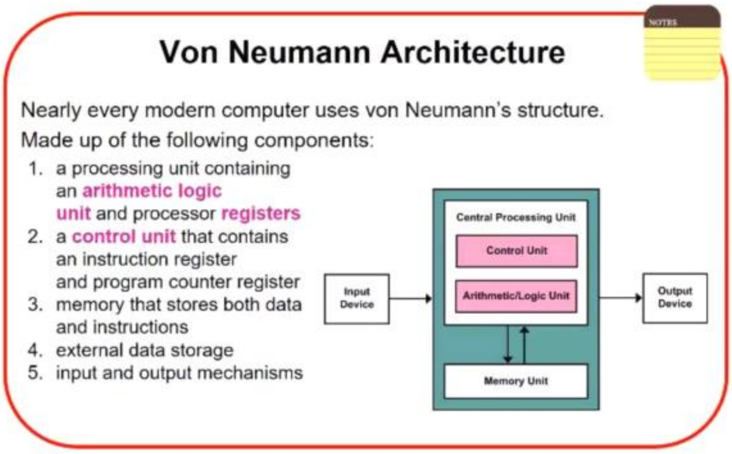
Add, subtract, arithmetic shift, logical shift, AND, OR, NOT

d State what goes onto the address bus during the fetch-decode-execute cycle.

The memory address of a location in RAM

e State two items that can go onto the data bus, during the fetch-decode-execute cycle.

Data and instructions



Von Neumann Architecture

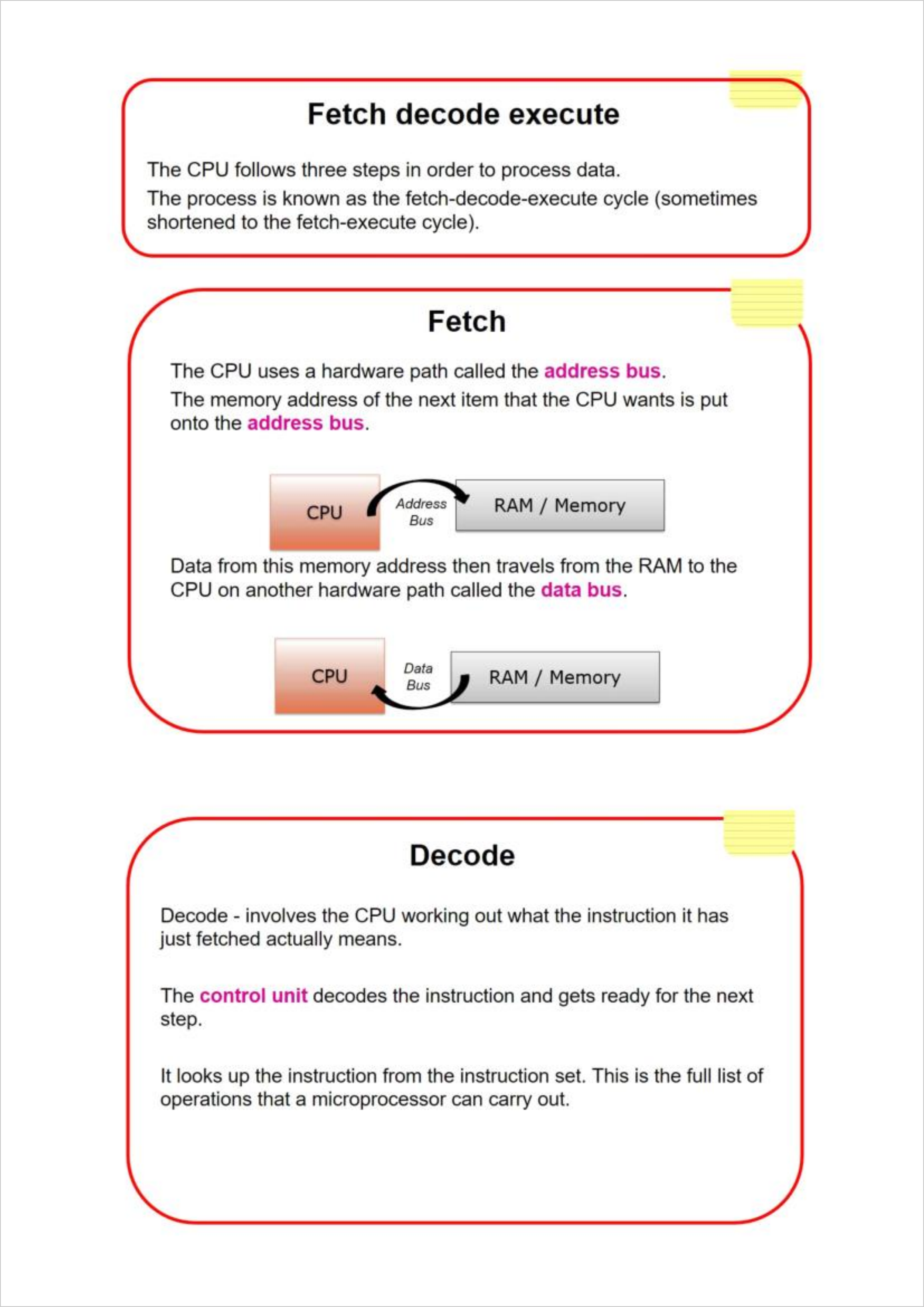
Nearly every modern computer uses von Neumann's structure.

Made up of the following components:

1. a processing unit containing an **arithmetic logic unit** and processor **registers**
2. a **control unit** that contains an instruction register and program counter register
3. memory that stores both data and instructions
4. external data storage
5. input and output mechanisms

Lesson 2 - Von Neumann Architecture

20 January 2021 10:28



Fetch decode execute

The CPU follows three steps in order to process data.

The process is known as the fetch-decode-execute cycle (sometimes shortened to the fetch-execute cycle).

Fetch

The CPU uses a hardware path called the **address bus**.

The memory address of the next item that the CPU wants is put onto the **address bus**.

CPU
Address Bus
RAM / Memory

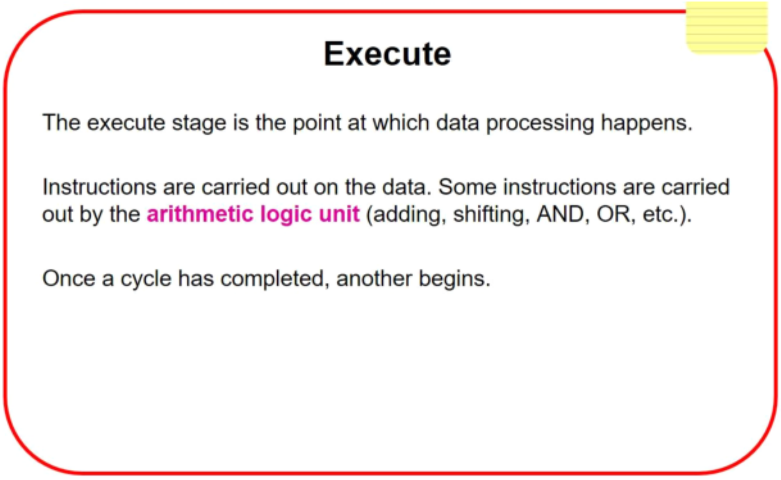
Data from this memory address then travels from the RAM to the CPU on another hardware path called the **data bus**.

Decode

Decode - involves the CPU working out what the instruction it has just fetched actually means.

The **control unit** decodes the instruction and gets ready for the next step.

It looks up the instruction from the instruction set. This is the full list of operations that a microprocessor can carry out.



Execute

The execute stage is the point at which data processing happens.

Instructions are carried out on the data. Some instructions are carried out by the **arithmetic logic unit** (adding, shifting, AND, OR, etc.).

Once a cycle has completed, another begins.

b Every processor has a unique instruction set. Define what is meant by an ‘instruction set’.

The full list of operations that a microprocessor chip can carry out.

c Give three examples of operations that would be executed in the arithmetic logic unit.

Add, subtract, arithmetic shift, logical shift, AND, OR, NOT

d State what goes onto the address bus during the fetch-decode-execute cycle.

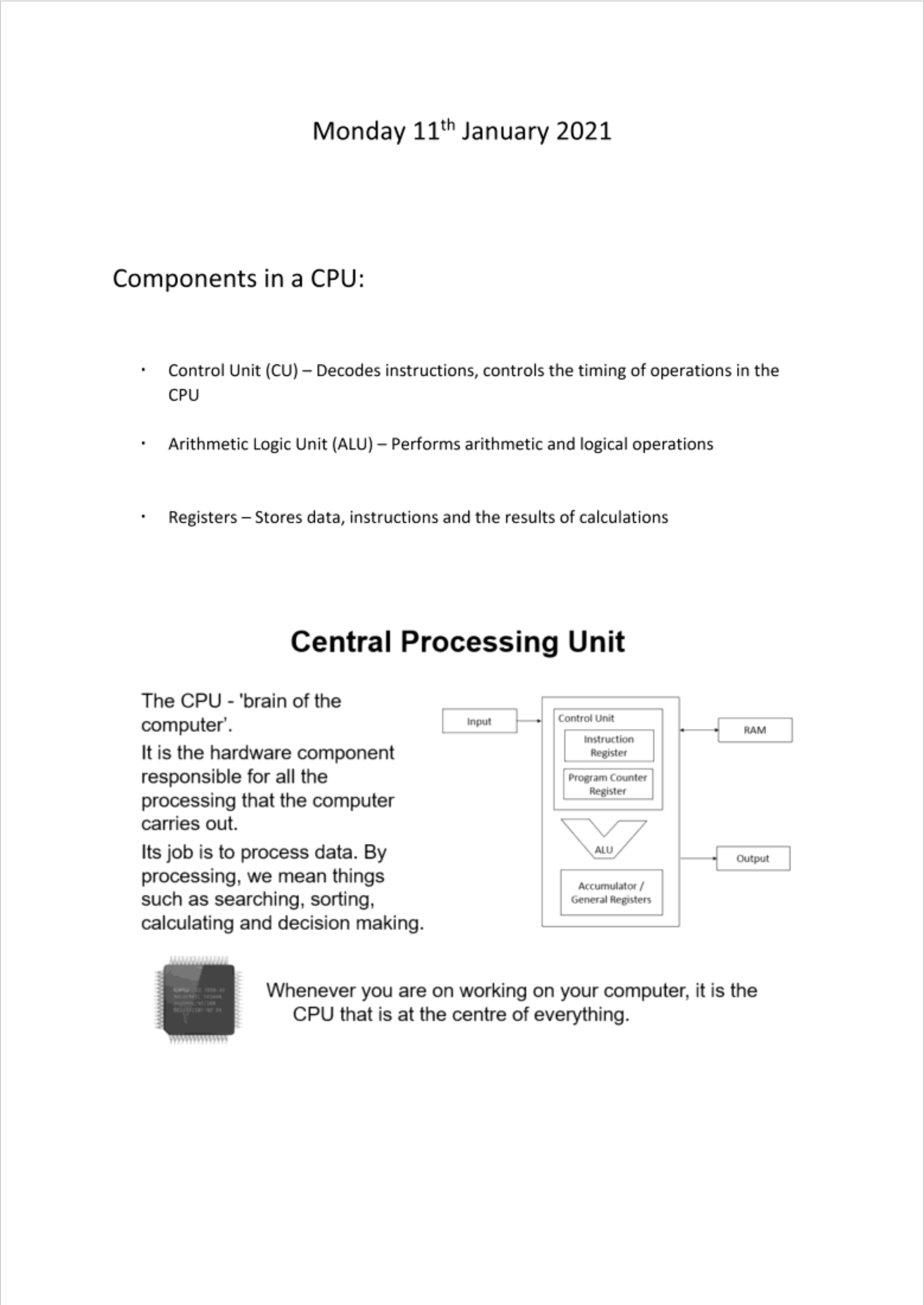
The memory address of a location in RAM

e State two items that can go onto the data bus, during the fetch-decode-execute cycle.

Data and instructions

Lesson 3 - Fetch Decode Execute

27 January 2021 14:55



Monday 11th January 2021

Components in a CPU:

- Control Unit (CU) – Decodes instructions, controls the timing of operations in the CPU
- Arithmetic Logic Unit (ALU) – Performs arithmetic and logical operations
- Registers – Stores data, instructions and the results of calculations

Central Processing Unit

The CPU - 'brain of the computer'.

It is the hardware component responsible for all the processing that the computer carries out.

Its job is to process data. By processing, we mean things such as searching, sorting, calculating and decision making.

Whenever you are on working on your computer, it is the CPU that is at the centre of everything.

Control unit

The control unit receives signals from other parts of the computer system and sends signals to them.

It is responsible for handling hardware interrupts (e.g. inputs from a mouse, keyboard, network, etc.).

It is responsible for fetching, decoding, and executing instructions.

It is the powerhouse of the CPU – it has the most responsibility.

Arithmetic logic unit

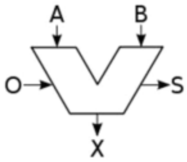
The Arithmetic Logic Unit (ALU) performs actual operations on data.

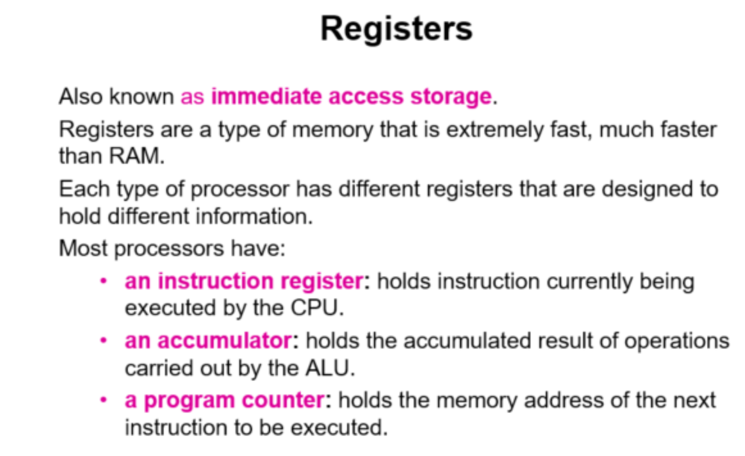
- Arithmetic: + / = *
- Logic: AND, OR, NOT, etc.

It is able to compare numbers against **0**.

It can test if two numbers are equal.

It uses logic gates in combination to perform operations.





Registers

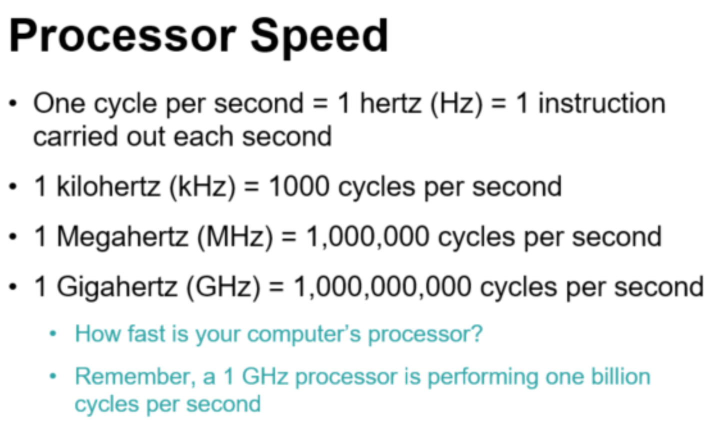
Also known as **immediate access storage**.

Registers are a type of memory that is extremely fast, much faster than RAM.

Each type of processor has different registers that are designed to hold different information.

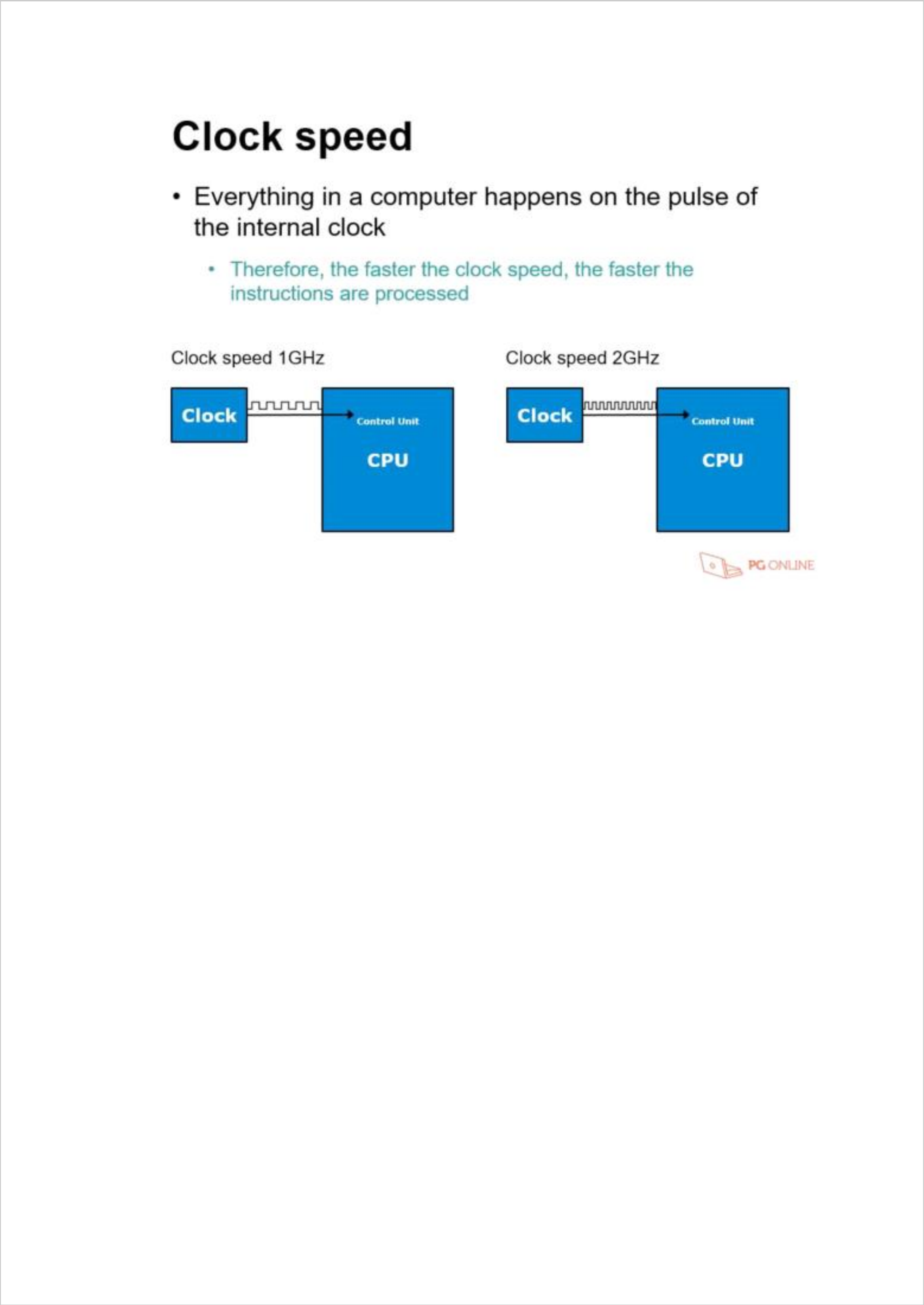
Most processors have:

- **an instruction register**: holds instruction currently being executed by the CPU.
- **an accumulator**: holds the accumulated result of operations carried out by the ALU.
- **a program counter**: holds the memory address of the next instruction to be executed.



Processor Speed

- One cycle per second = 1 hertz (Hz) = 1 instruction carried out each second
- 1 kilohertz (kHz) = 1000 cycles per second
- 1 Megahertz (MHz) = 1,000,000 cycles per second
- 1 Gigahertz (GHz) = 1,000,000,000 cycles per second
  - How fast is your computer's processor?
  - Remember, a 1 GHz processor is performing one billion cycles per second



Clock speed

- Everything in a computer happens on the pulse of the internal clock
  - Therefore, the faster the clock speed, the faster the instructions are processed

Clock speed 1GHz

Clock

Control Unit

CPU

Clock speed 2GHz

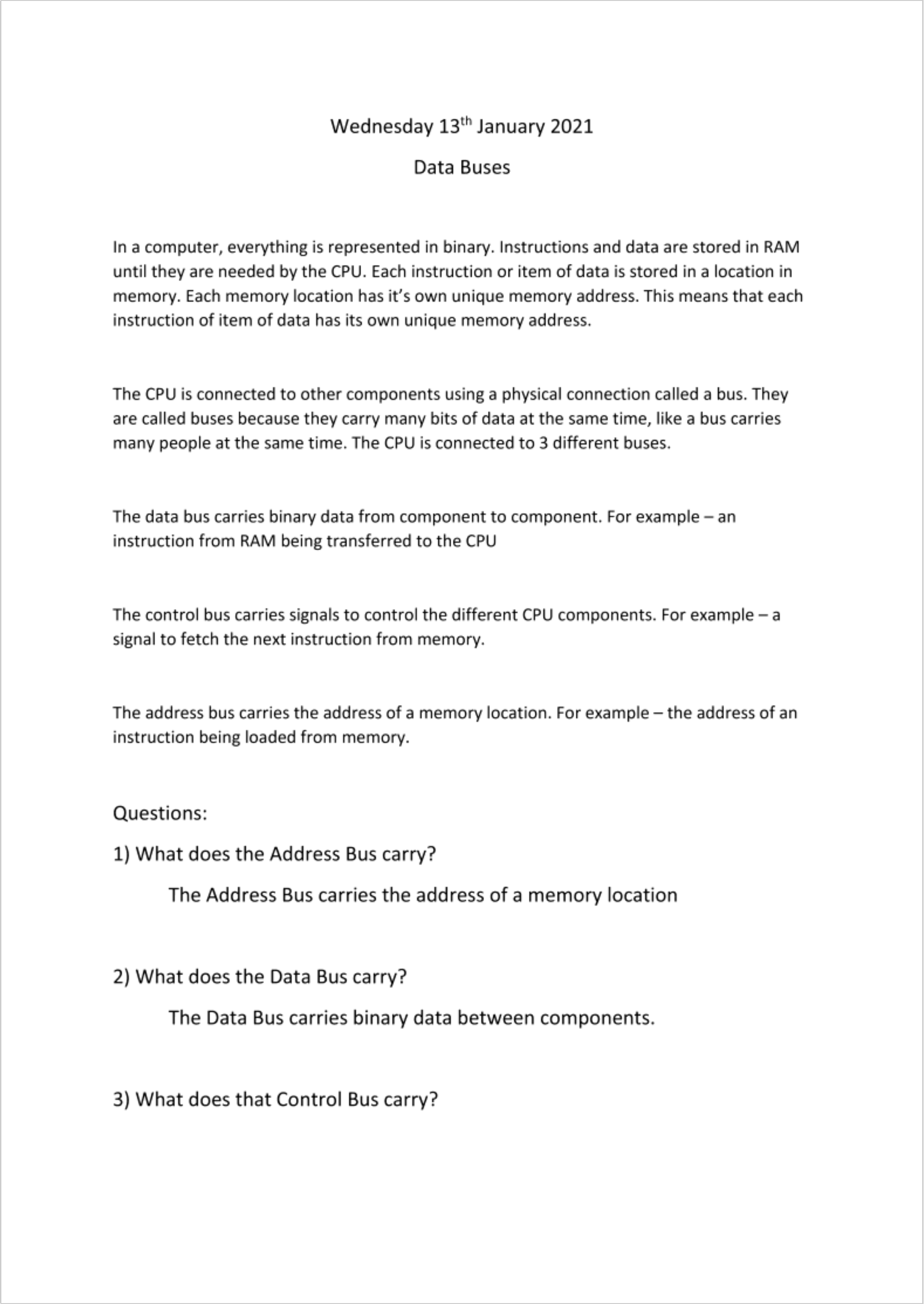
Clock

Control Unit

CPU

Lesson 4 - Data Buses

20 January 2021 10:28



Wednesday 13th January 2021

Data Buses

In a computer, everything is represented in binary. Instructions and data are stored in RAM until they are needed by the CPU. Each instruction or item of data is stored in a location in memory. Each memory location has it's own unique memory address. This means that each instruction of item of data has its own unique memory address.

The CPU is connected to other components using a physical connection called a bus. They are called buses because they carry many bits of data at the same time, like a bus carries many people at the same time. The CPU is connected to 3 different buses.

The data bus carries binary data from component to component. For example – an instruction from RAM being transferred to the CPU

The control bus carries signals to control the different CPU components. For example – a signal to fetch the next instruction from memory.

The address bus carries the address of a memory location. For example – the address of an instruction being loaded from memory.

Questions:

1) What does the Address Bus carry?

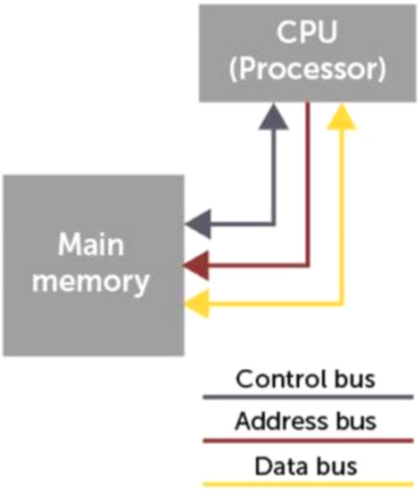
The Address Bus carries the address of a memory location

2) What does the Data Bus carry?

The Data Bus carries binary data between components.

3) What does that Control Bus carry?

The Control Bus carries signals to control CPU components.



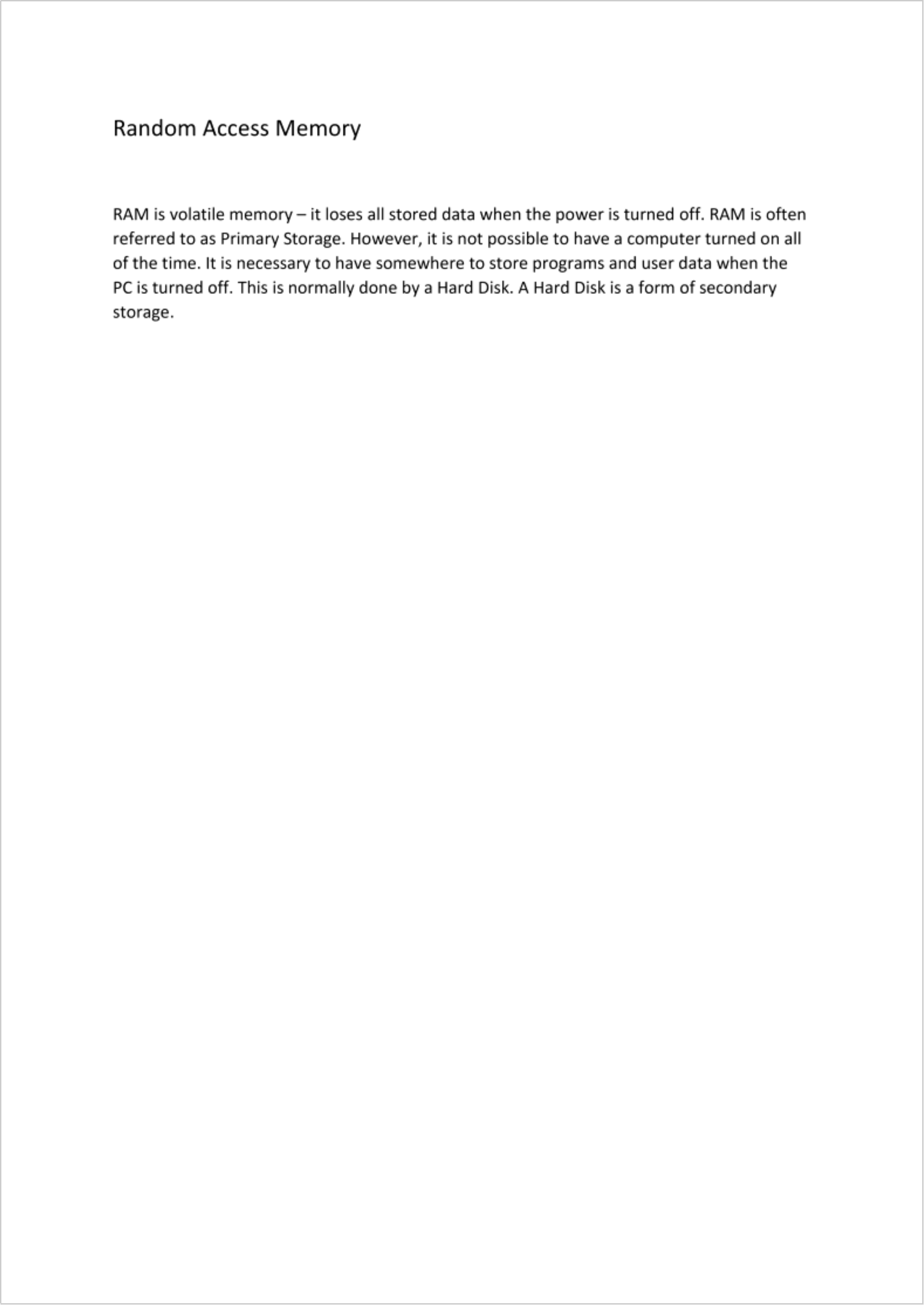
Direction of Buses.

The Control Bus is bidirectional – it carries status information to devices and back.

The Data Bus is bidirectional – it carries data to and from devices so that data can be read and written.

The Address Bus is unidirectional (one way) – it can point the CPU to an address in memory, but the memory cannot point back at the CPU.

The difference between 32-bit and 64-bit CPUs is that 32-bit CPUs are cheaper but can only address a limited amount of RAM (4Gb). If more RAM is available, the CPU does not see it and cannot use it.

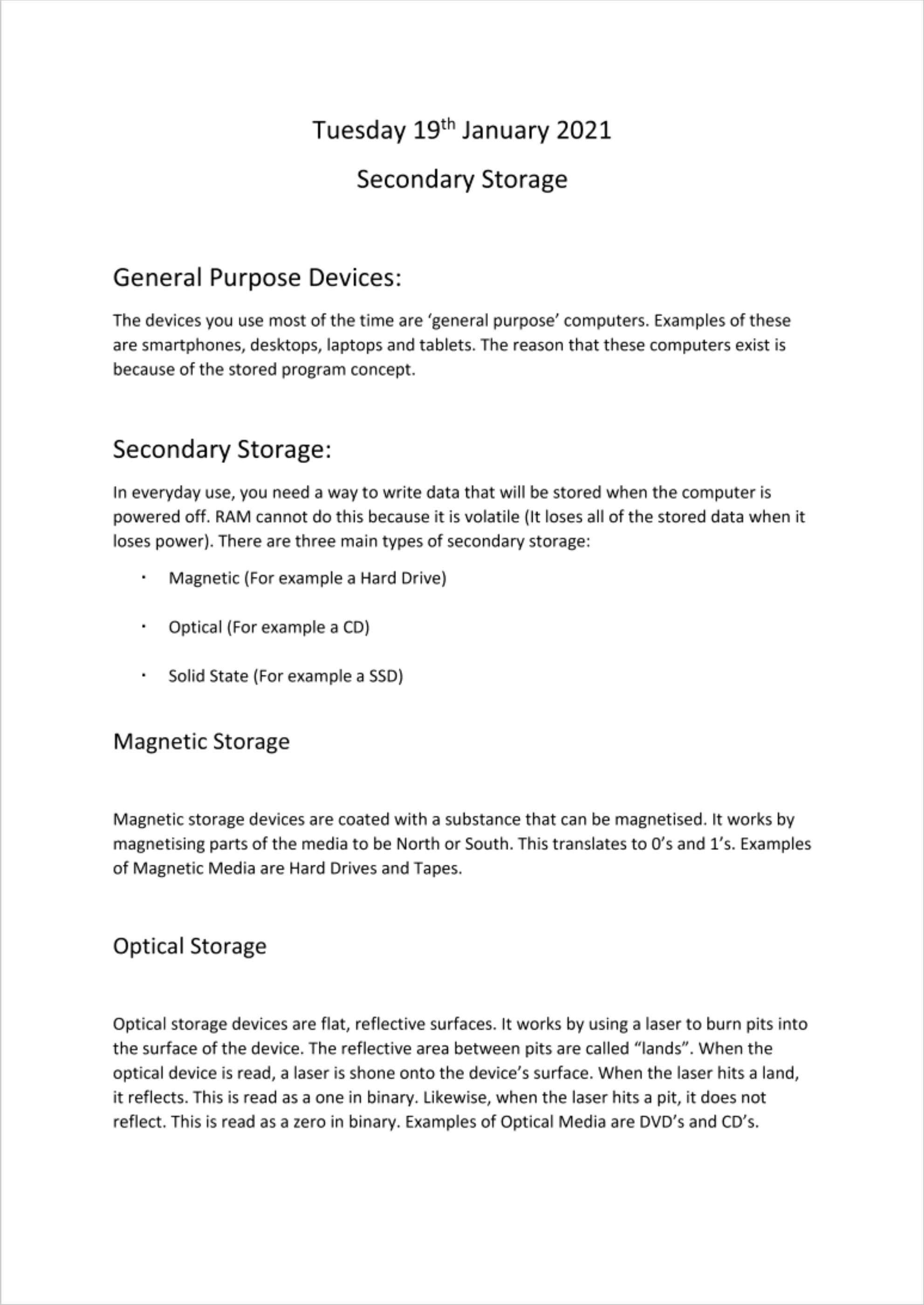


Random Access Memory

RAM is volatile memory – it loses all stored data when the power is turned off. RAM is often referred to as Primary Storage. However, it is not possible to have a computer turned on all of the time. It is necessary to have somewhere to store programs and user data when the PC is turned off. This is normally done by a Hard Disk. A Hard Disk is a form of secondary storage.

Lesson 5 - Secondary Storage

20 January 2021 10:23



Tuesday 19th January 2021

Secondary Storage

General Purpose Devices:

The devices you use most of the time are 'general purpose' computers. Examples of these are smartphones, desktops, laptops and tablets. The reason that these computers exist is because of the stored program concept.

Secondary Storage:

In everyday use, you need a way to write data that will be stored when the computer is powered off. RAM cannot do this because it is volatile (It loses all of the stored data when it loses power). There are three main types of secondary storage:

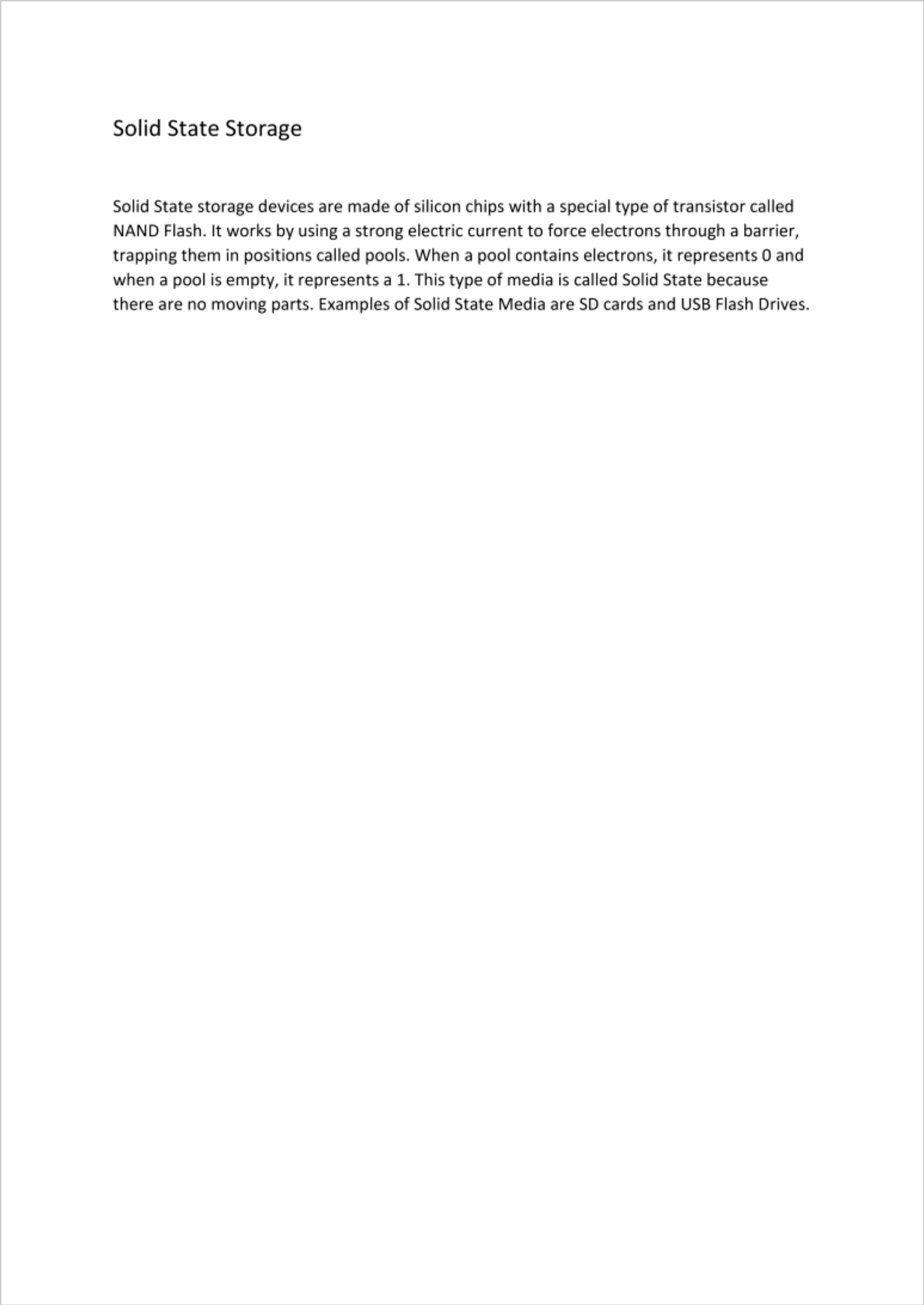
- Magnetic (For example a Hard Drive)
- Optical (For example a CD)
- Solid State (For example a SSD)

Magnetic Storage

Magnetic storage devices are coated with a substance that can be magnetised. It works by magnetising parts of the media to be North or South. This translates to 0's and 1's. Examples of Magnetic Media are Hard Drives and Tapes.

Optical Storage

Optical storage devices are flat, reflective surfaces. It works by using a laser to burn pits into the surface of the device. The reflective area between pits are called "lands". When the optical device is read, a laser is shone onto the device's surface. When the laser hits a land, it reflects. This is read as a one in binary. Likewise, when the laser hits a pit, it does not reflect. This is read as a zero in binary. Examples of Optical Media are DVD's and CD's.

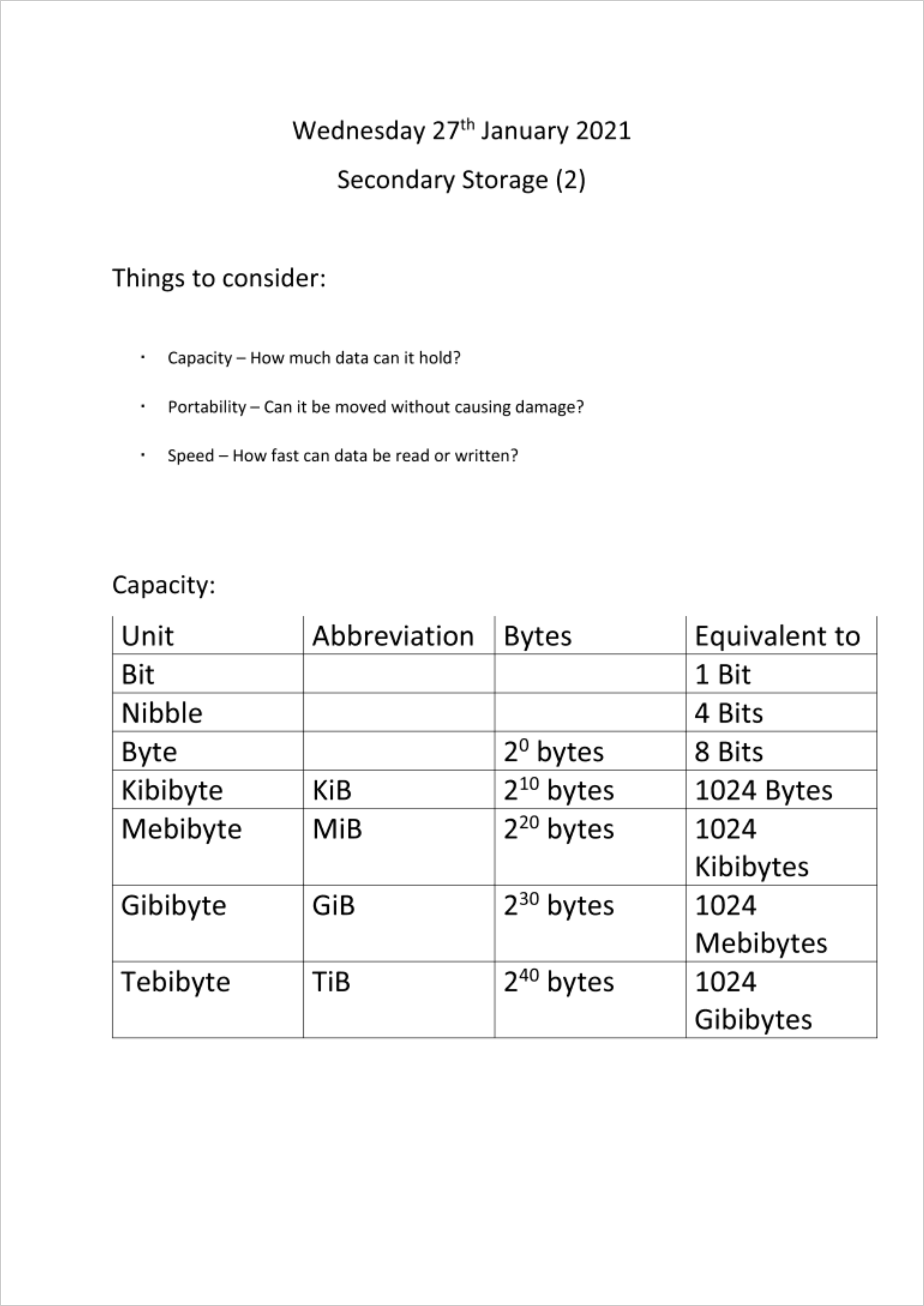


Solid State Storage

Solid State storage devices are made of silicon chips with a special type of transistor called NAND Flash. It works by using a strong electric current to force electrons through a barrier, trapping them in positions called pools. When a pool contains electrons, it represents 0 and when a pool is empty, it represents a 1. This type of media is called Solid State because there are no moving parts. Examples of Solid State Media are SD cards and USB Flash Drives.

Lesson 6 - Secondary Storage (2)

27 January 2021 14:51



Wednesday 27th January 2021

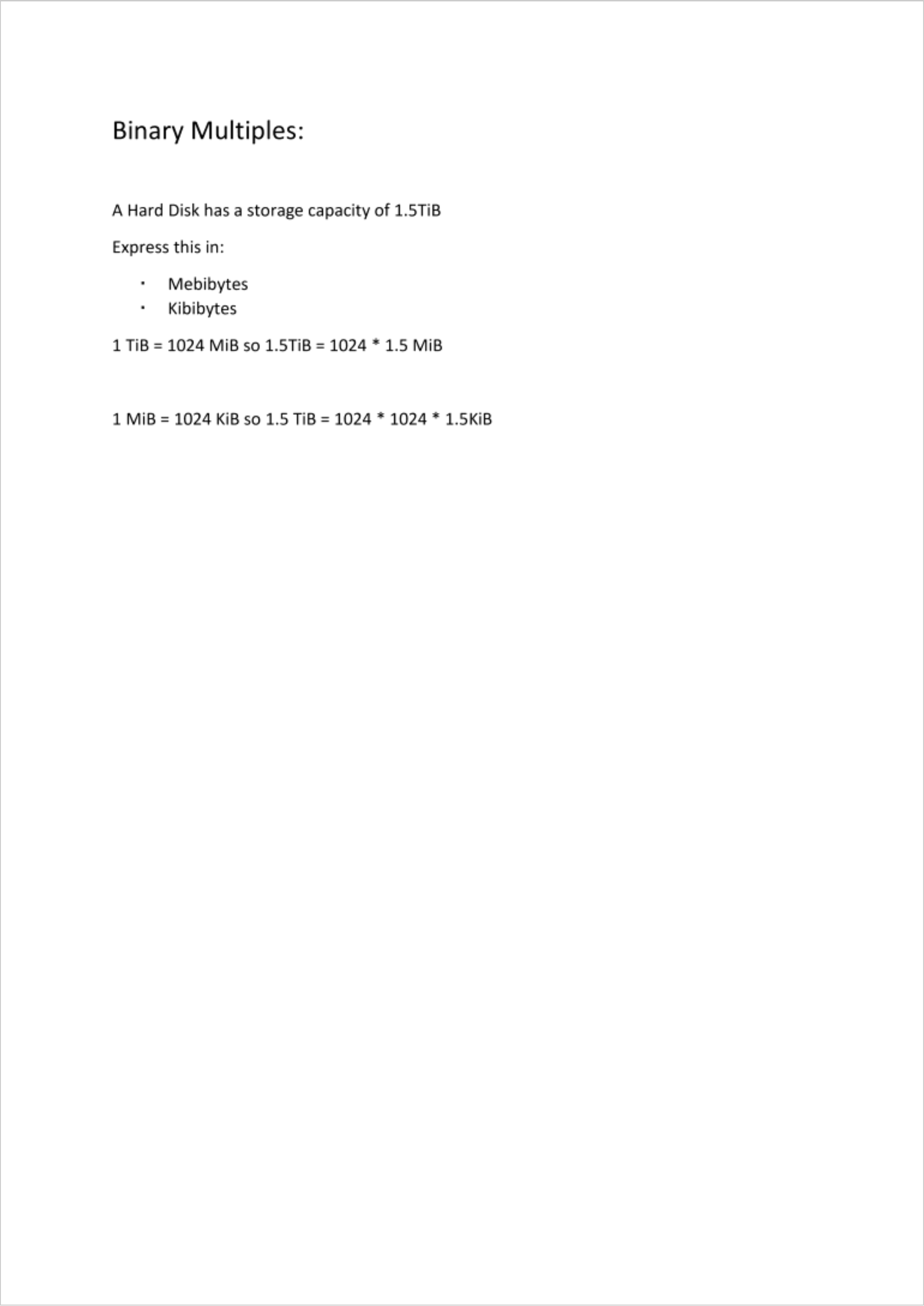
Secondary Storage (2)

Things to consider:

- Capacity – How much data can it hold?
- Portability – Can it be moved without causing damage?
- Speed – How fast can data be read or written?

Capacity:

Unit	Abbreviation	Bytes	Equivalent to
Bit			1 Bit
Nibble			4 Bits
Byte		2^0 bytes	8 Bits
Kibibyte	KiB	2^{10} bytes	1024 Bytes
Mebibyte	MiB	2^{20} bytes	1024 Kibibytes
Gibibyte	GiB	2^{30} bytes	1024 Mebibytes
Tebibyte	TiB	2^{40} bytes	1024 Gibibytes



Binary Multiples:

A Hard Disk has a storage capacity of 1.5TiB

Express this in:

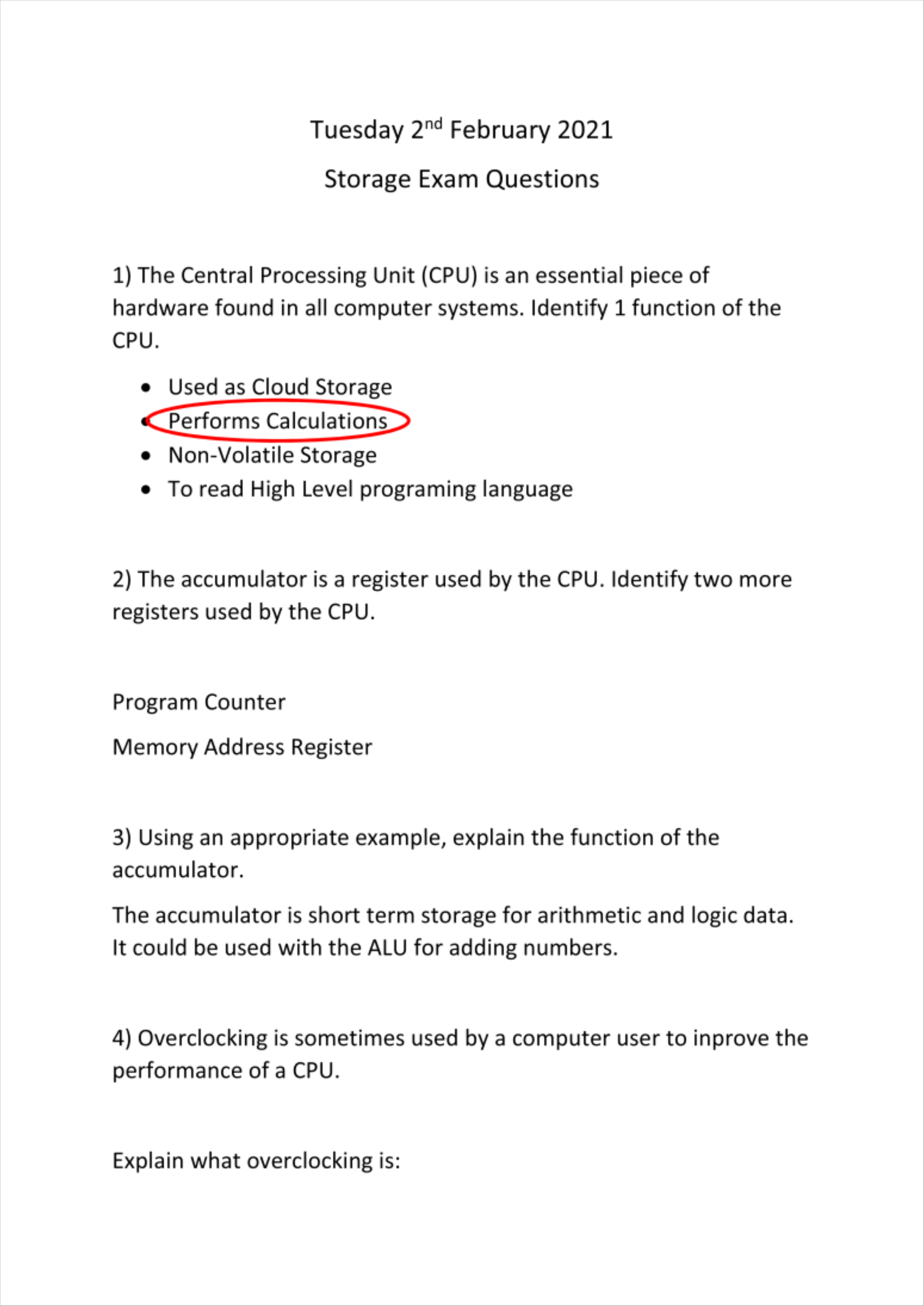
- Mebibytes
- Kibibytes

1 TiB = 1024 MiB so 1.5TiB = 1024 * 1.5 MiB

1 MiB = 1024 KiB so 1.5 TiB = 1024 * 1024 * 1.5KiB

Lesson 7 - Storage Exam Questions

02 February 2021 13:05



Tuesday 2nd February 2021

Storage Exam Questions

1) The Central Processing Unit (CPU) is an essential piece of hardware found in all computer systems. Identify 1 function of the CPU.

- Used as Cloud Storage
- [✓ Performs Calculations]
- Non-Volatile Storage
- To read High Level programing language

2) The accumulator is a register used by the CPU. Identify two more registers used by the CPU.

Program Counter

Memory Address Register

3) Using an appropriate example, explain the function of the accumulator.

The accumulator is short term storage for arithmetic and logic data. It could be used with the ALU for adding numbers.

4) Overclocking is sometimes used by a computer user to inprove the performance of a CPU.

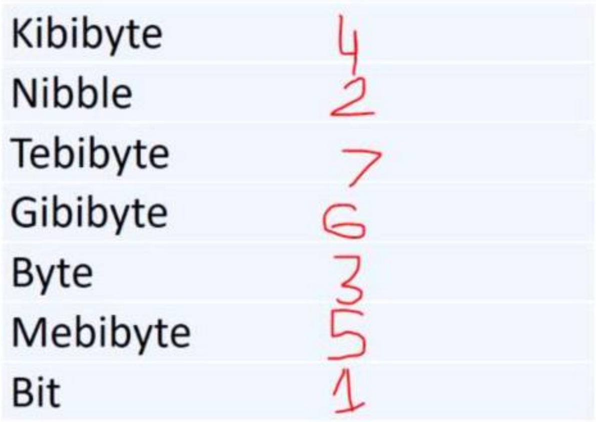
Explain what overclocking is:

Overclocking is increasing the clock speed of the CPU.

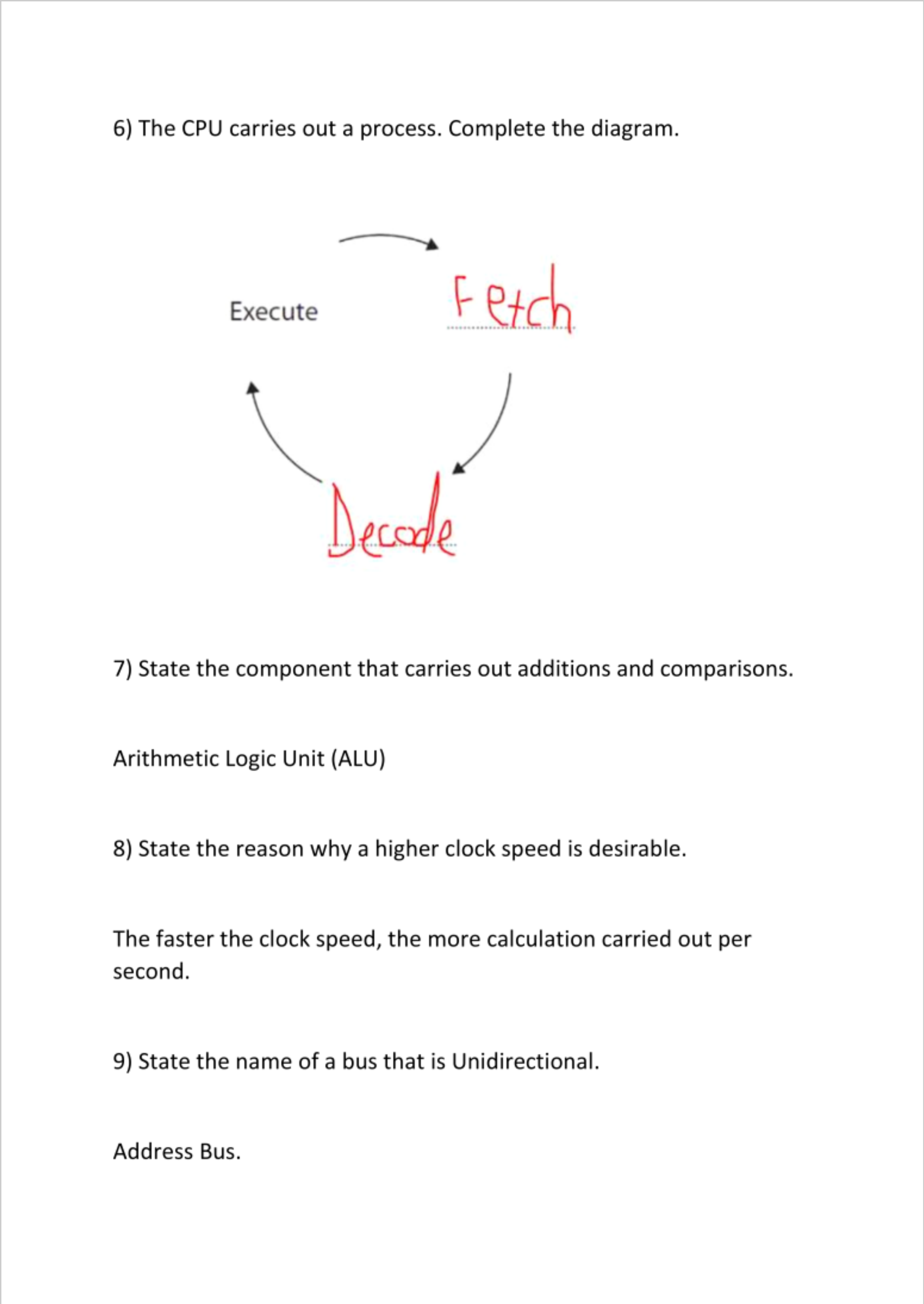
What reason do some computer users give for not using overclocking?

People may not do this because it breaks the warranty, and causes the CPU to generate more heat.

5) Place these data capacities in order of size from 1 – 7 (1 being the smallest):



Kibibyte	4
Nibble	2
Tebibyte	7
Gibibyte	6
Byte	3
Mebibyte	5
Bit	1



6) The CPU carries out a process. Complete the diagram.

Execute
Fetch
Decode

7) State the component that carries out additions and comparisons.

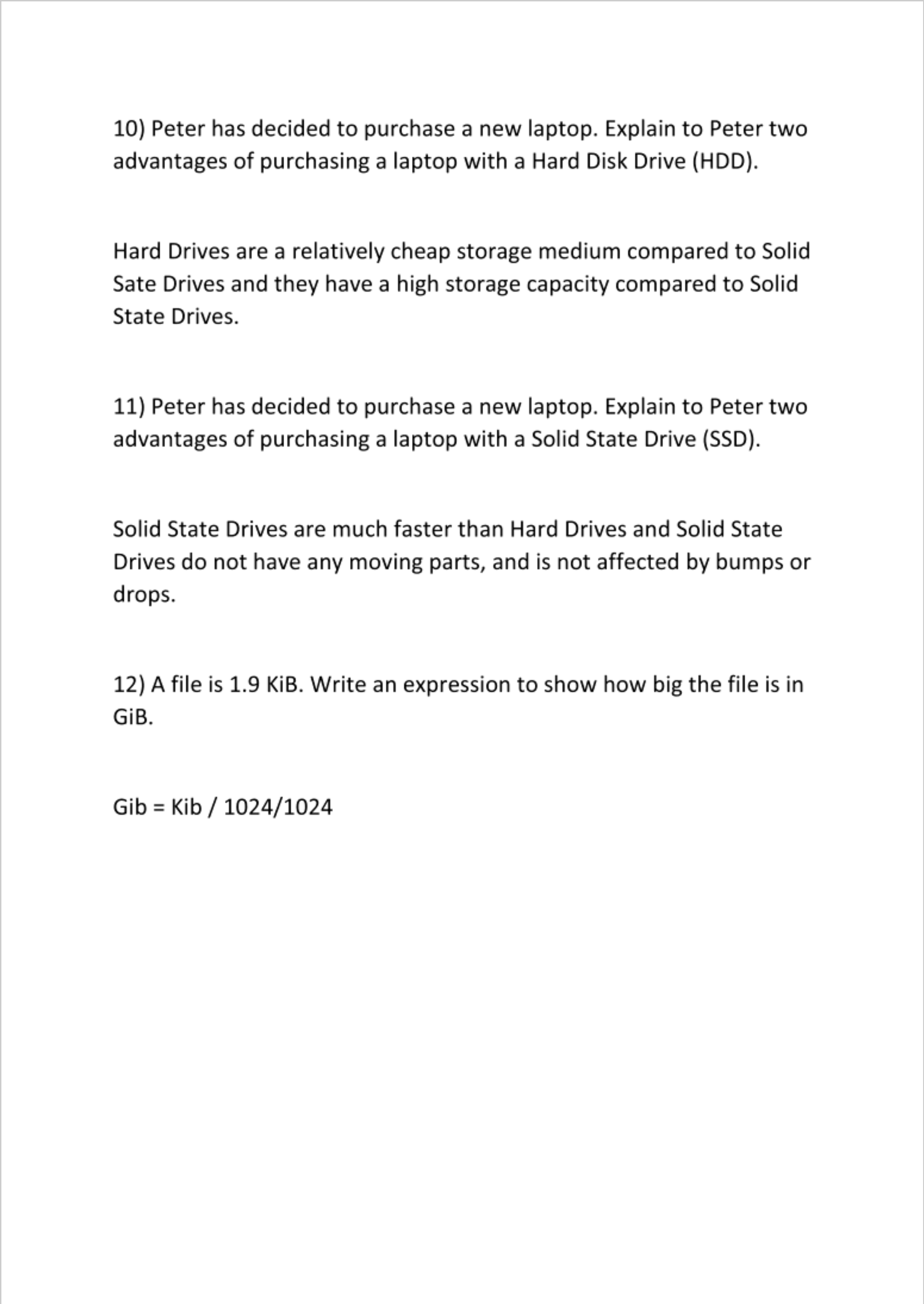
Arithmetic Logic Unit (ALU)

8) State the reason why a higher clock speed is desirable.

The faster the clock speed, the more calculation carried out per second.

9) State the name of a bus that is Unidirectional.

Address Bus.



10) Peter has decided to purchase a new laptop. Explain to Peter two advantages of purchasing a laptop with a Hard Disk Drive (HDD).

Hard Drives are a relatively cheap storage medium compared to Solid Sate Drives and they have a high storage capacity compared to Solid State Drives.

11) Peter has decided to purchase a new laptop. Explain to Peter two advantages of purchasing a laptop with a Solid State Drive (SSD).

Solid State Drives are much faster than Hard Drives and Solid State Drives do not have any moving parts, and is not affected by bumps or drops.

12) A file is 1.9 KiB. Write an expression to show how big the file is in GiB.

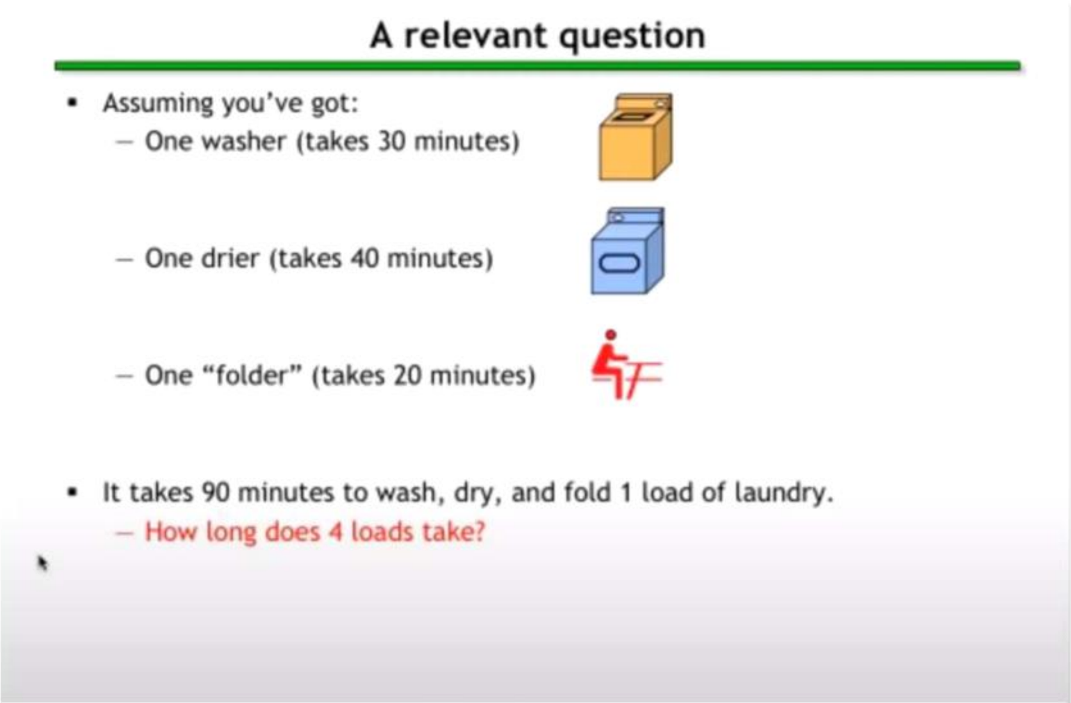
Gib = Kib / 1024/1024

Lesson 8 - Pipelining

03 February 2021 10:04

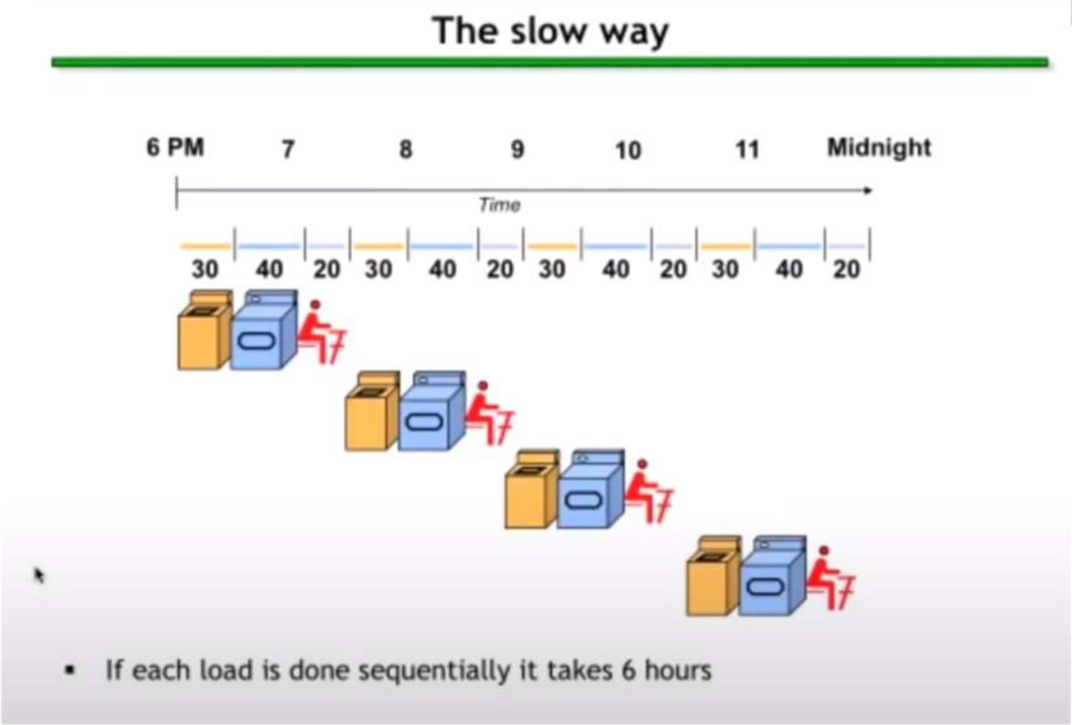
Wednesday 3rd February 2021

Pipelining



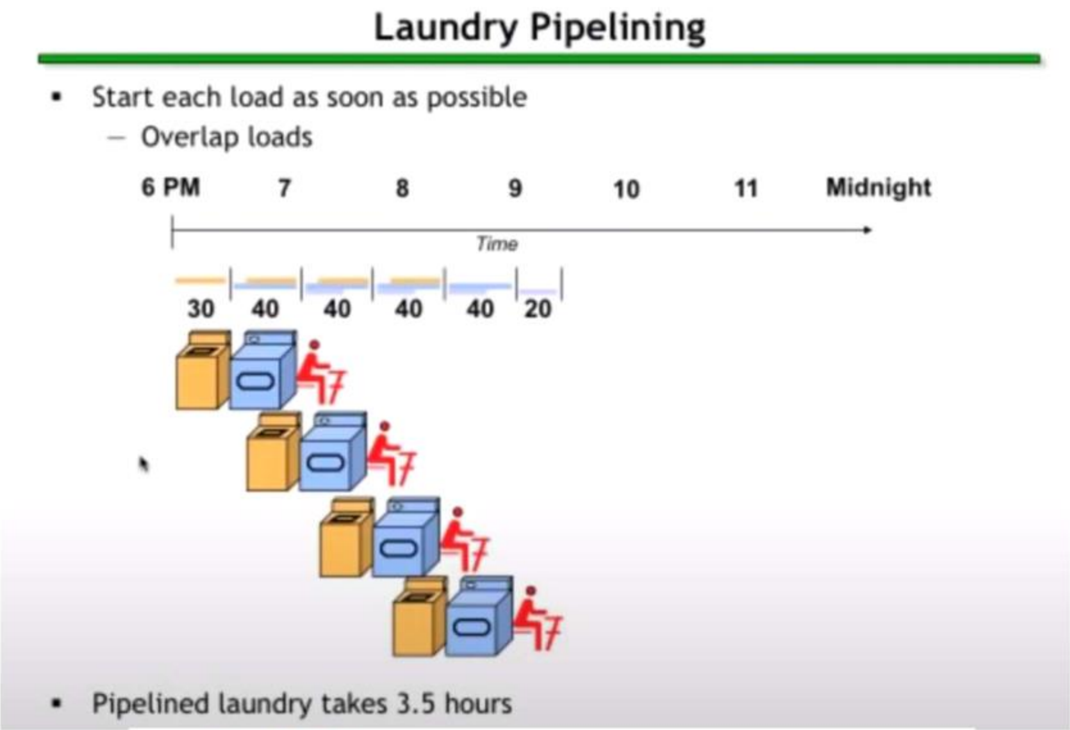
A relevant question

- Assuming you've got:
  - One washer (takes 30 minutes)
  - One drier (takes 40 minutes)
  - One “folder” (takes 20 minutes)
- It takes 90 minutes to wash, dry, and fold 1 load of laundry.
  - How long does 4 loads take?



The slow way

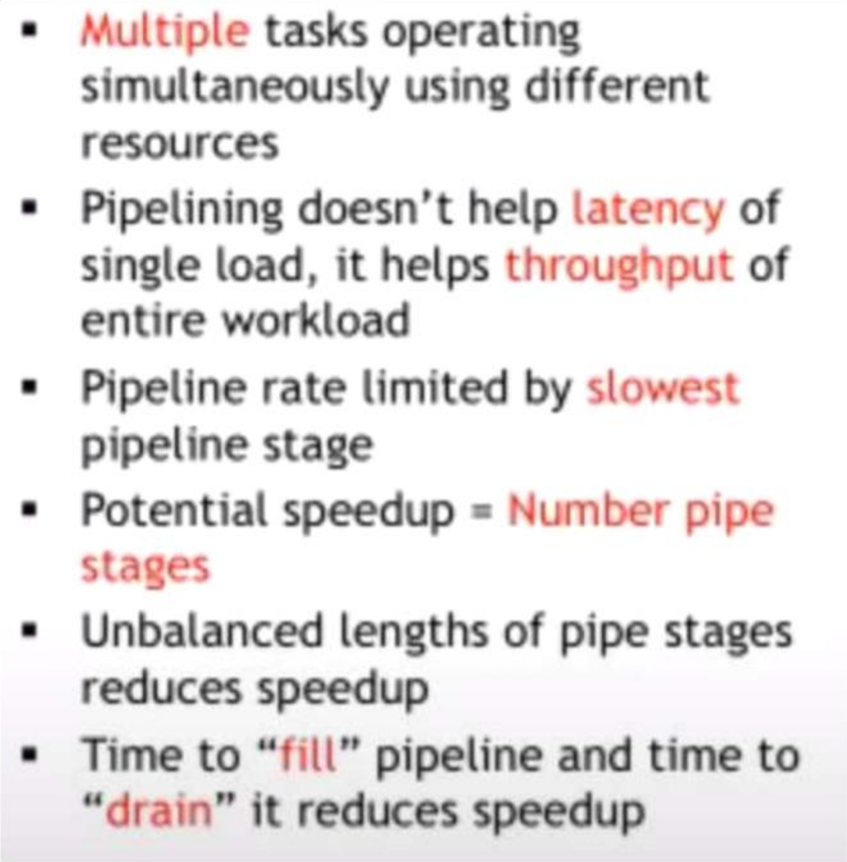
- If each load is done sequentially it takes 6 hours



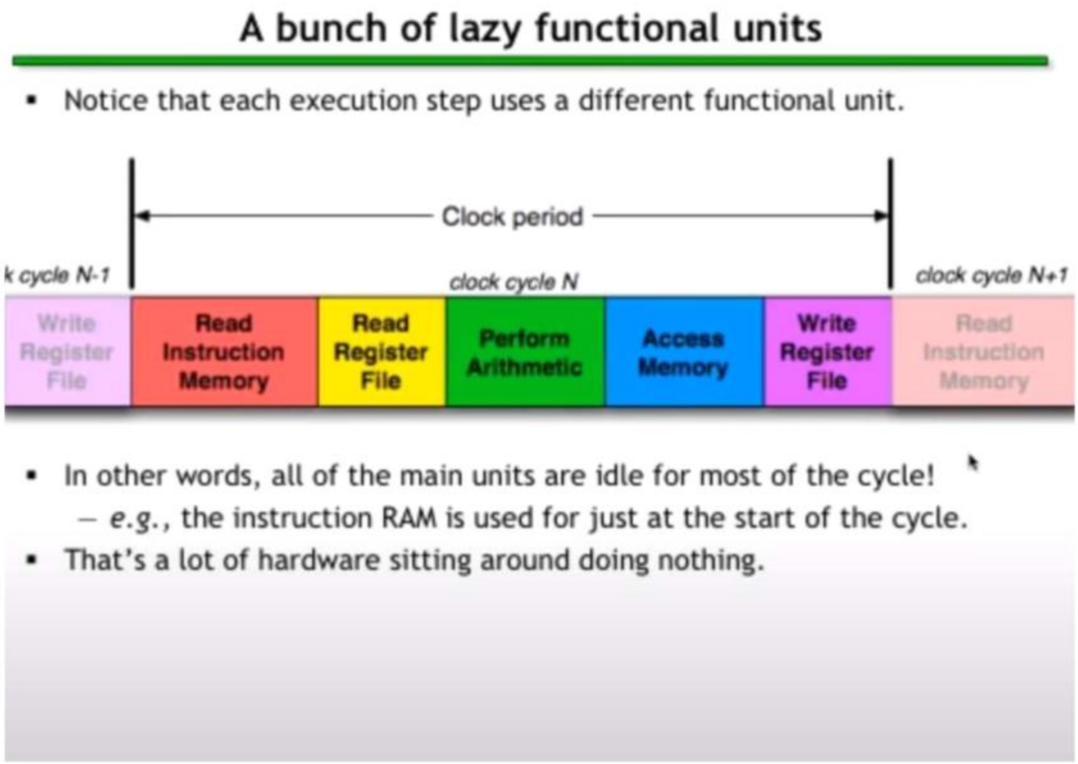
Laundry Pipelining

- Start each load as soon as possible
  - Overlap loads

- Pipelined laundry takes 3.5 hours



- Multiple tasks operating simultaneously using different resources
- Pipelining doesn't help latency of single load, it helps throughput of entire workload
- Pipeline rate limited by slowest pipeline stage
- Potential speedup = Number pipe stages
- Unbalanced lengths of pipe stages reduces speedup
- Time to "fill" pipeline and time to "drain" it reduces speedup



A bunch of lazy functional units

- Notice that each execution step uses a different functional unit.

- In other words, all of the main units are idle for most of the cycle!
  - *e.g.*, the instruction RAM is used for just at the start of the cycle.
- That's a lot of hardware sitting around doing nothing.

We shouldn't have to wait for the entire instruction to complete before we can re-use the functional units.

For example, the instruction memory is free in the Instruction Decode step as shown below, so...

Why don't we go ahead and fetch the *next* instruction while we're decoding the first one?

Similarly, once the first instruction enters its Execute stage, we can go ahead and decode the second instruction.

But now the instruction memory is free again, so we can fetch the third instruction!

For the five steps in a load instruction (the machine's longest instruction)

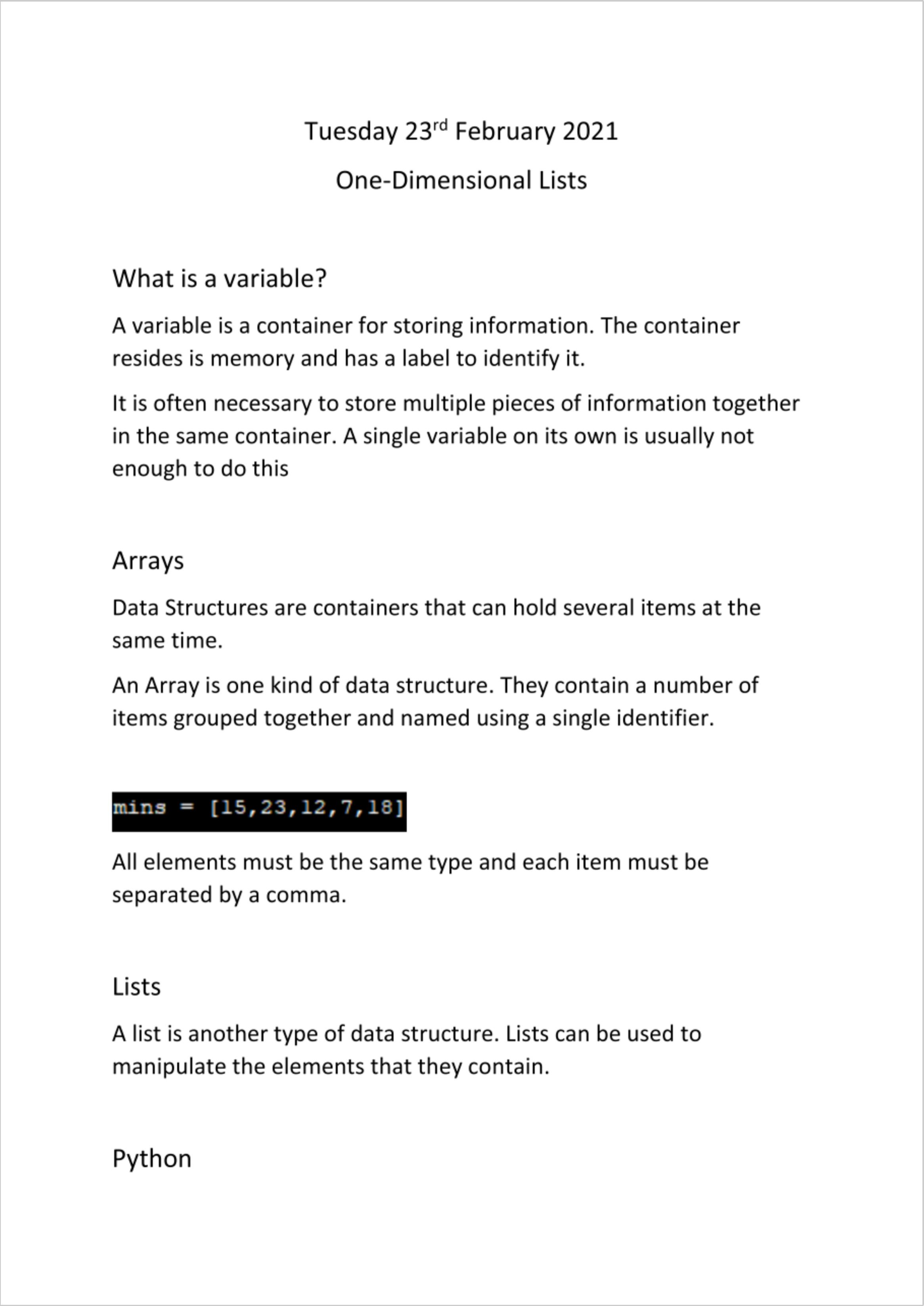
- Stages are: IF, ID, EX, MEM, and WB

Supports executing 5 instructions simultaneously: one in each stage.

Each stage has its own functional units.

One-Dimensional Lists

23 February 2021 13:45



Tuesday 23rd February 2021

One-Dimensional Lists

What is a variable?

A variable is a container for storing information. The container resides is memory and has a label to identify it.

It is often necessary to store multiple pieces of information together in the same container. A single variable on its own is usually not enough to do this

Arrays

Data Structures are containers that can hold several items at the same time.

An Array is one kind of data structure. They contain a number of items grouped together and named using a single identifier.

```
mins = [15,23,12,7,18]
```

All elements must be the same type and each item must be separated by a comma.

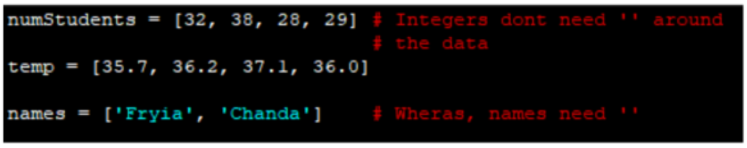
Lists

A list is another type of data structure. Lists can be used to manipulate the elements that they contain.

Python

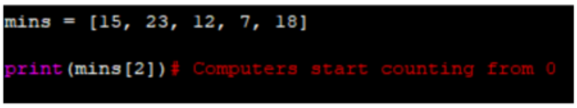
Python

Here is an example of a python list used to implement arrays:



```
numStudents = [32, 38, 28, 29] # Integers dont need '' around
                               # the data
temp = [35.7, 36.2, 37.1, 36.0]

names = ['Fryia', 'Chanda']    # Wheras, names need ''
```

Individual elements of an array can be accessed using a index to its location. For example,

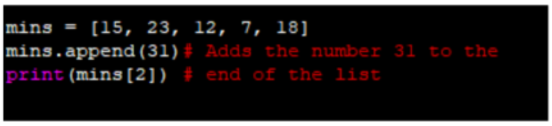


```
mins = [15, 23, 12, 7, 18]

print(mins[2])# Computers start counting from 0
```

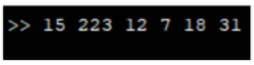
Would output 12 because it is number 2 in the list.

Append

You can add new items to the end of a list using {arrayname}.append()



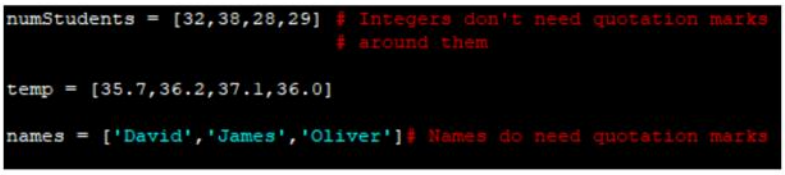
```
mins = [15, 23, 12, 7, 18]
mins.append(31)# Adds the number 31 to the
print(mins[2]) # end of the list
```



```
>> 15 223 12 7 18 31
```

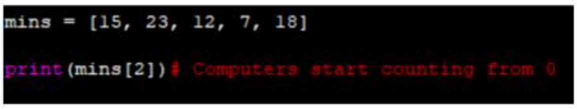
As you can see, 31 has been added to the list

Here is an example of a python list used to implement arrays:



```
numStudents = [32,38,28,29] # Integers don't need quotation marks
                            # around them

temp = [35.7,36.2,37.1,36.0]

names = ['David','James','Oliver']# Names do need quotation marks
```

Individual elements of an array can be accessed using a index to its location. For example,

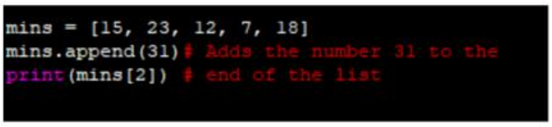


```
mins = [15, 23, 12, 7, 18]

print(mins[2])# Computers start counting from 0
```

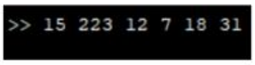
Would output 12 because it is number 2 in the list.

Append

You can add new items to the end of a list using {arrayname}.append() For example:



```
mins = [15, 23, 12, 7, 18]
mins.append(31)# Adds the number 31 to the
print(mins[2]) # end of the list
```

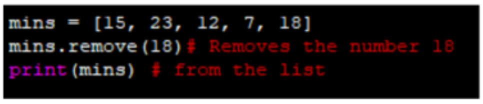


```
>> 15 223 12 7 18 31
```

As you can see, 31 has been added to the list

Remove

You can remove items from the list using {arrayname}.remove()

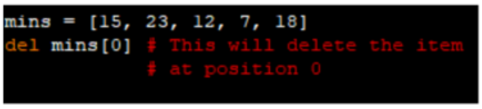


```
mins = [15, 23, 12, 7, 18]
mins.remove(18)# Removes the number 18
print(mins) # from the list
```

However, there can be issues if the data you want to remove isn't there, or if there are two of that data and you want to delete the second one.

Delete

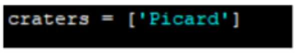
You can delete specific items from the list using {arrayname}.del({itemNumber})



```
mins = [15, 23, 12, 7, 18]
del mins[0] # This will delete the item
            # at position 0
```

Activity:

'Craters' is a list of craters on the moon. Append the following code:



```
craters = ['Picard']
```

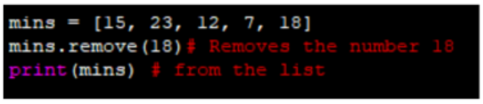
Amend the code to complete the following tasks:

- Add "Thebit"
- Add "Metius"
- Add "Humboldt"

Remove

You can remove items from the list using {arrayname}.remove()

For example:

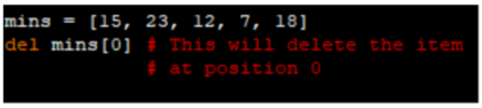


```
mins = [15, 23, 12, 7, 18]
mins.remove(18)# Removes the number 18
print(mins) # from the list
```

However, there can be issues if the data you want to remove isn’t there, or if there are two of that data and you want to delete the second one.

Delete

You can delete specific items from the list using del {arrayname}({itemNumber}) For example:

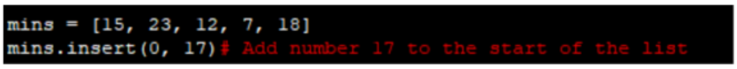


```
mins = [15, 23, 12, 7, 18]
del mins[0] # This will delete the item
            # at position 0
```

Insert:

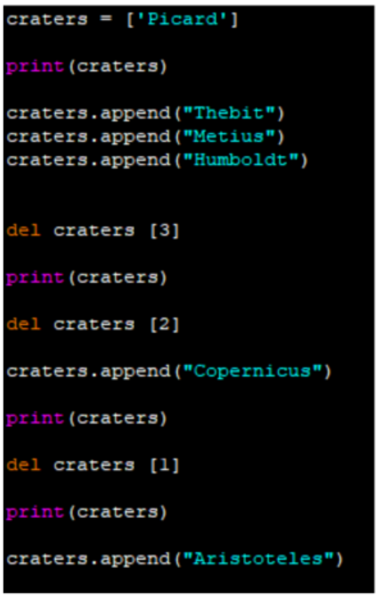
You can add an item to a specific place using insert({position}, {item})

For example:



```
mins = [15, 23, 12, 7, 18]
mins.insert(0, 17)# Add number 17 to the start of the list
```

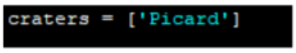
- Display the list
- Remove “Metius”
- Add “Cornacopius”
- Display the list
- Remove the second item in the list
- Display the list
- Add “Aristotels”
- Display the list



```
craters = ['Picard']

print(craters)

craters.append("Thebit")
craters.append("Metius")
craters.append("Humboldt")


del craters [3]

print(craters)

del craters [2]

craters.append("Copernicus")

print(craters)

del craters [1]

print(craters)

craters.append("Aristoteles")
```

Activity:

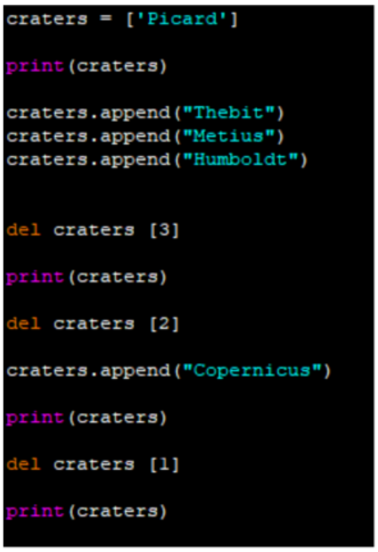
'Craters' is a list of craters on the moon. Append the following code:



```
craters = ['Picard']
```

Amend the code to complete the following tasks:

- Add "Thebit"
- Add "Metius"
- Add "Humboldt"
- Display the list
- Remove "Metius"
- Add "Cornacopius"
- Display the list
- Remove the second item in the list
- Display the list
- Add "Aristotels"
- Display the list



```
craters = ['Picard']

print(craters)

craters.append("Thebit")
craters.append("Metius")
craters.append("Humboldt")


del craters [3]

print(craters)

del craters [2]

craters.append("Copernicus")

print(craters)

del craters [1]

print(craters)
```