OCR A Level Computer Science

Your notes

1.1 Structure & Function of the Processor

Contents

Your notes

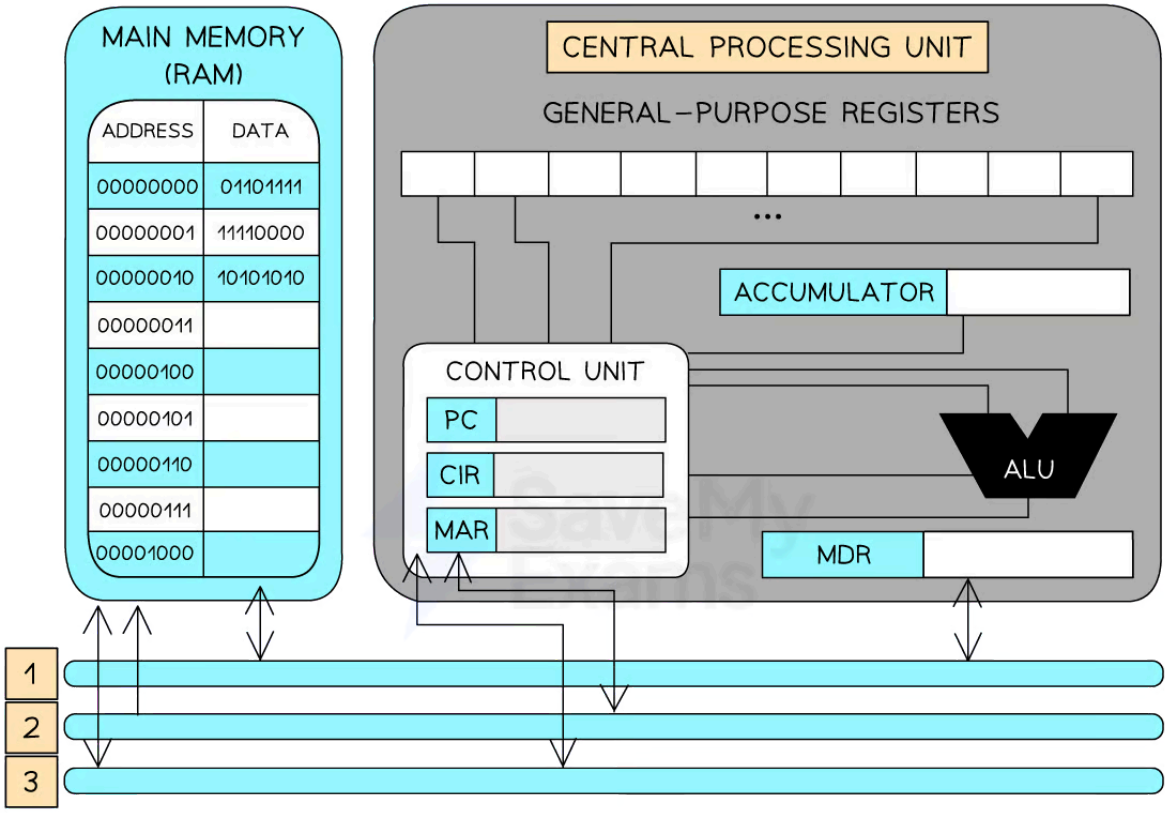
Components of the CPU

What is the CPU?

- The **CPU** is responsible for processing all data within the computer
- It is made up of a number of **components** including:
  - Arithmetic and Logic Unit (**ALU**)
  - Control Unit (**CU**)
  - **Registers** are temporary storage/memory locations inside the CPU which are used for a single specific purpose. They have a faster access speed than RAM / secondary storage. There are a number of registers in the CPU:
    - Program Counter (**PC**)
    - Accumulator (**ACC**)
    - Memory Address Register (**MAR**)
    - Memory Data Register (**MDR**)
    - Current Instruction Register (**CIR**)

Diagram of Components Within the CPU

Your notes



1 DATA BUS

2 ADRESS BUS

3 CONTROL BUS

Component	Function
Arithmetic and Logic Unit (ALU)	This performs any arithmetic calculations (e.g. adding binary) or any logic comparisons (using AND, OR, NOT) The ALU is made up of several components

Control Unit	This is where instructions are decoded. The CU also controls the data within the CPU and how it moves around
Program Counter (PC)	This stores the address in memory of the next instruction to be fetched
Accumulator (ACC)	This is where values are stored temporarily, either after they've been inputted or loaded, or after being calculated in the ALU
Memory Address Register (MAR)	This is where addresses are stored, either for where data is being sent in memory, or where it is being fetched from
Memory Data Register (MDR)	This is where data/instructions are stored, either before it sent to memory, or after being fetched
Current Instruction Register (CIR)	When an instruction has been fetched from memory it is loaded here before being split into opcode and operand. After this, it will be decoded.


Your notes

The ALU is made up of several components:

- **Arithmetic circuit**
  - This carries out any arithmetic (addition, subtraction, multiplication or division)
- **Logic circuit**
  - This carries out operations like AND, OR, NOT, XOR
- **Registers**
  - These are additional registers to those mentioned above and can store data
- **Status flags**
  - This includes overflow flags (if the value is too large for the register) or could include a zero flag (to tell if the answer is 0 easily)
- **Buses**
  - These are used to transport data around the ALU and to other parts of the CPU

Buses

- There are 3 buses which connect the **CPU** with the main memory (**RAM**):

- **Data bus**
- **Address bus**
- **Control bus**

Your notes

Bus	Purpose	Operation
Data Bus	Holds data being sent to/from the CPU and RAM	Read/Write
Address Bus	Holds addresses being sent to/from the CPU and RAM	Read/Write
Control Bus	Sends Signals to determine whether the other buses are in read or write mode	Sends signals

- There are also a number of other buses within the CPU which transport data between the different areas

Your notes

Fetch-Decode-Execute Cycle

What is the Fetch-Decode-Execute Cycle?

- The fetch-decode-execute cycle is the process that the CPU goes through repeatedly to process instructions
- There are **3 stages**:
  - **Fetching** an instruction from memory - supplying the address and receiving the instruction from memory
  - **Decoding** the instruction - interpreting the instruction and then reading and retrieving the required data from their addresses
  - **Executing** the instruction - the CPU carries out the required action

Examiner Tips and Tricks

- You'll need to know how to write assembly language using the **Little Man Computer** instruction set but now we're just covering how the registers are used in assembly language

Which Registers are Used in the CPU During the Fetch-Decode-Execute Cycle?

In the section below registers and CPU components appear in **bold** and assembly language is in *italics*

During the Fetch-Decode-Execute Cycle, the following steps happen:

- **Fetch**
  - The **PC** is loaded with 0
  - The value from the **PC** (0) is copied to the **MAR**
  - The data from the **MAR** (0) is sent across the address bus with the instruction to read the data sent across the control bus
  - The data from that location in memory (0) is sent down the data bus to the **MDR**
  - The **PC** is incremented by 1

- **Decode**
  - The data is sent from the **MDR** to the **CIR** where it is split into the opcode and operand
  - This is sent to the **CU** to be decoded
- **Execute**
  - Which registers are used here will depend on the instruction being executed
    - If a value is being inputted (*INP*) the **ACC** will store the value
    - If a value is being outputted (*OUT*) this will be the value currently in the **ACC**
    - If a value is loaded from RAM (*LDA*) this will be sent across the data bus from RAM (in the address location in the **MAR**) to the **MDR**
    - If a value is to be stored (*STA*) it will take the value from the **ACC**, send it to the **MDR** and then send it across the data bus to RAM (to the address location in the **MAR**)
    - If a value is being added to or subtracted from another value (*ADD/SUB*)
    - If the LMC code is to branch (*BRA/BRZ/BRP*) the comparison will take place in the **ALU**

Your notes

Worked Example

A program written in the Little Man Computer instruction set is given below.

```
<>     INP
       STA num
loop   LDA total
       ADD num
       STA total
       LDA count
       ADD one
       STA count
       SUB num
       BRZ end
       BRA loop
end    LDA total
       OUT
       HLT
one    DAT 1
num    DAT 0
count  DAT 0
total  DAT 0
```

Explain which registers are used and their values when the line `STA count` **is executed and the accumulator is holding the value 9. The label** `count` **refers to memory location 16.**

[2]

Your notes

How to answer this question:

- The instruction being executed in this example is STA count, so the registers used will be:
  - **ACC –** the accumulator is holding the value "9"
  - **MDR –** the value "9" from the **ACC** will be copied to the MDR
  - **MAR –** the value 16 will be stored here so the data is sent to memory location 16
- The value that's in the accumulator **(ACC)** is going to be stored in memory (RAM)
- To work out where in memory it will be stored we need to know what count represents
- In this question, we've been told it's **16 –** *"the label* count refers to memory location 16"
- You're not always told this in your exam, so you should also be able to count the lines of code
- It's best to write the line numbers on the code in the question, to see which line count is on. (Don't forget that the first line is memory location 0!)
- "Count DAT 0 "is on line 16, so the value "16" is what goes to the **MAR** register. (You already know this from the question above)
- It is always the value from the accumulator (**ACC**) that is stored, so the value "9" must go to the **MDR** as data can only be sent to memory from the MDR
- Then the value is sent to the memory location, in this case 16

Answer:

Example answer that gets full marks:

The contents of the accumulator (9 in this case) will be copied to the MDR [1 mark] and then 9 is copied to location 16. [1 mark]

Acceptable answers you could have given instead:

The value 16 is copied to the MAR. [1 mark]

Examiner Tips and Tricks

- When answering a question about registers **be specific about the contents of the register** based on the question to make sure you get the marks
- If your answer to the question above didn't include the values 16 and 9 then you wouldn't get the marks despite knowing which registers are used. This is because the question is asking you which registers are used **and** their values when the line STA count is executed


Your notes

CPU Performance

How do we Measure CPU Performance?

- There are 3 main ways in which CPU performance can be measured:
  - Clock speed
  - Number of cores
  - Cache

Clock Speed

- The **clock** in the computer controls operations within the CPU. It repeatedly changes from 0 to 1 to 0 and so on - each one of these is referred to as a state change
- A state change can represent one **fetch-decode-execute cycle** although some take more than one cycle
- Clock speed is a measure of **how many states changes the CPU performs per second**. 1 cycle per second = 1 Hz
- A typical computer may have a clock speed of **2.3 GHz** = 2,300,000,000 Hz which is just over **2 billion cycles per second**
- If a computer has a higher clock speed, it will be able to **execute more instructions per second** and therefore carry out tasks more quickly

Number of Cores

- A core is a **processing unit** within the CPU
- A single core processor has one core but a computer can have a **dual core** processor, **quad core** processor or octa core processor up to around 76 cores
- Normally each core will run at the same speed
- This means that tasks are carried out more quickly using a multicore processor than a single core processor as **each core can carry out its task** (this is called **parallel** processing) but a **dual core processor isn't twice as fast as a single core** processor as there is some time spent organising tasks between the cores
- The speed of the CPU is also partially determined by the tasks being carried out as **some tasks can't be split between cores** and therefore having more cores is no quicker than having one

Cache

Your notes

- The cache is part of **primary storage** and is used to store **frequently used data and instructions**
- It is used as it's **closer to the CPU** than **RAM** and therefore is **faster to retrieve data from** (some of the cache is included within each core)
- The more cache there is, the more data can be stored which speeds up the performance of the CPU
  - E.g. A website you visit often will be stored in the cache so that the next time you visit the website it can load it up more quickly. When the webpage is updated, what is stored in the cache will be updated too
- The cache is split into different levels - each level is different in size and speed
  - Level 1
  - Level 2
  - Level 3

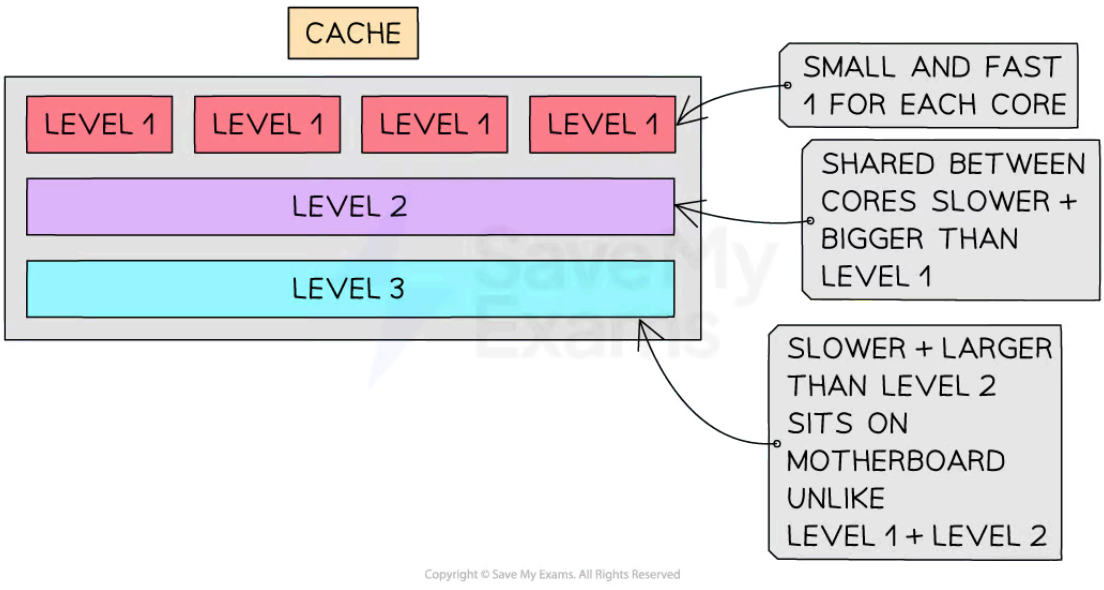


Diagram of the levels within the cache

Your notes

Pipelining

What is Pipelining?

- Pipelining is the process of carrying out multiple instructions **concurrently**
- Each instruction will be at a different stage of the **fetch-decode-execute cycle**
- One instruction can be **fetched** while the previous one is being **decoded** and the one before is being **executed**
- In the case of a **branch**, the pipeline is flushed
- This table shows which stage each instruction is at during each step:

	Fetch	Decode	Execute
Step 1	Instruction A		
Step 2	Instruction B	Instruction A	
Step 3	Instruction C	Instruction B	Instruction A
Step 4	Instruction D	Instruction C	Instruction B

- While one instruction is being executed, the next instruction will be decoded and the following instruction will be fetched

How Does Pipelining Improve Processor Performance?

- Pipelining reduces **latency**
- The **CPU is not idle** while waiting for the next instruction which **increases the speed of execution**
- The next instruction is fetched while the current one is decoded/executed
- All parts of the processor can be used at any instance in time

Your notes

Von Neumann & Harvard Architecture

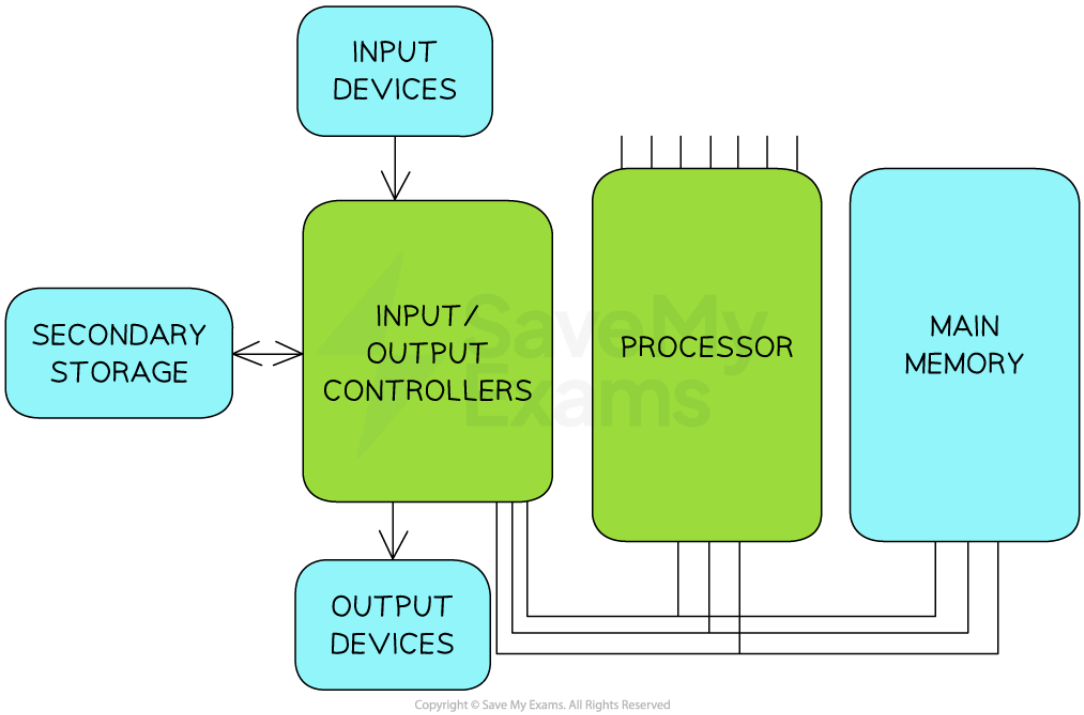
What is Computer Architecture?

- A computer architecture describes how it uses the components and instructions to make the computer function
- There are 2 common types of computer architecture:
  - Von Neumann architecture
  - Harvard architecture

Architecture	Memory Organisation	Bus	Address Space	Control Units	Usage
Von Neumann	Unified Memory	Single	Shared	Single Control Unit	Most modern computers, microcontrollers
Harvard	Separated Memory	Separate	Distinct	Separate Control Units	Specialized embedded systems

Von Neumann Architecture

Your notes



Layout of Von Neumann Architecture

Von Neumann's architecture includes:

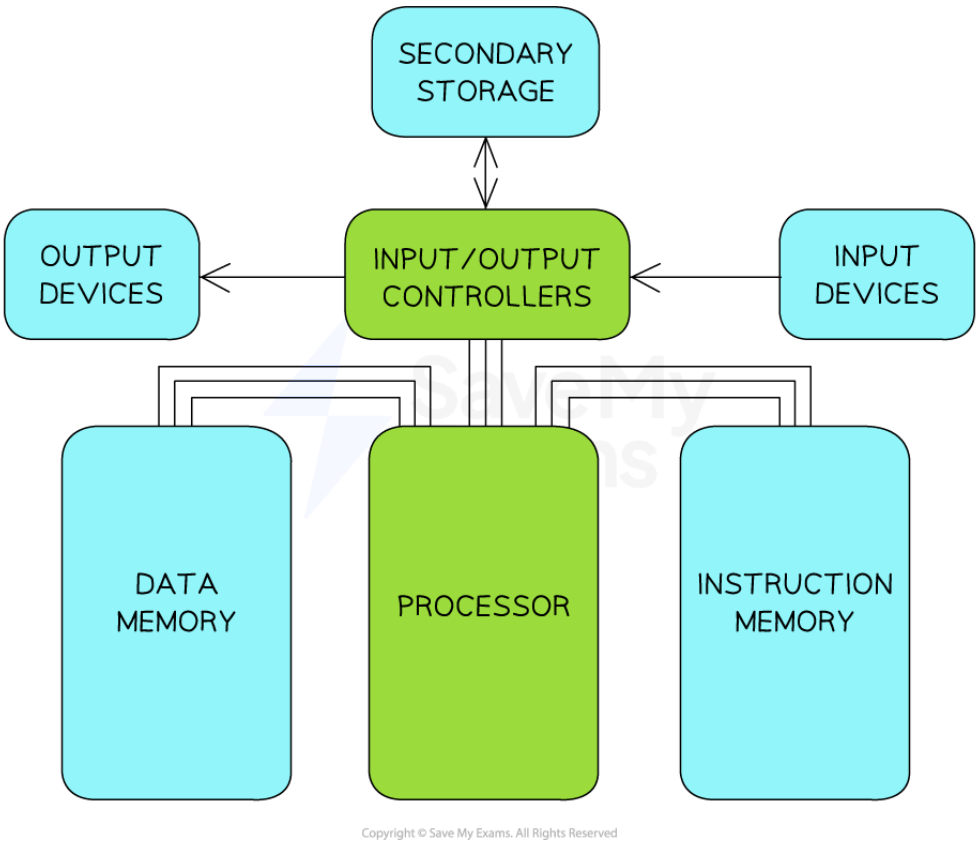
- **Control Unit (CU)**
  - The control unit controls the operation of the processor and its components
  - It retrieves instructions stored in memory, decodes or interprets them, and then executes them
  - The CU generates timing signals and controls the other units of the computer
- **Arithmetic Logic Unit (ALU)**
  - The ALU carries out all the arithmetic and logic operations in the computer such as:
    - Addition
    - Subtraction
    - Multiplication
    - Division
    - Comparisons, etc.

- **Special registers** within the CPU
- A **single set of buses** to connect the CPU to memory and Input/Output
  - These are communication systems that transfer data between the components inside a computer, or between computers
  - There are 3 types of buses:
    - The data bus carries data
    - The address bus carries the addresses where data must be picked up or stored
    - The control bus carries signals relating to the control and coordination of all the activities within the computer
- **Memory (RAM)**
  - The memory unit stores both data and instructions for processing
  - These are stored in the same format
  - The memory is divided into cells, each of which can be accessed by their address
  - The memory is a linear or sequential array of bytes


Your notes

Harvard Architecture

Your notes



Layout of Harvard Architecture

Harvard architecture includes:

- **Separate Instruction and Data Memory**
  - In Harvard architecture, the system has separate memory units for storing data and instructions
  - This separation provides greater speed and efficiency as the system can fetch data and instructions simultaneously from separate buses, without one interfering with the other
- **Separate Instruction and Data Buses**
  - Harvard architecture uses separate buses for data and fetching instructions, meaning that data transfers do not interfere with instruction fetches
  - This can lead to better overall system performance
- **Control Unit (CU)**

- The control unit controls the operation of the processor and its components
- It retrieves instructions stored in memory, decodes or interprets them, and then executes them
- The CU generates timing signals and controls the other units of the computer

- **Arithmetic Logic Unit (ALU)**
  - The ALU carries out all the arithmetic and logic operations in the computer such as:
    - Addition
    - Subtraction
    - Multiplication
    - Division
    - Comparisons, etc.


Your notes

How does Harvard Architecture improve performance over Von Neumann Architecture?

There are several things that Harvard architecture has or can do which can improve performance over Von Neumann architecture:

Feature	Explanation
Two separate areas of memory	One for instructions and one for data Instructions and data can be accessed concurrently
Different sets of buses	One for instructions and one for data Instructions and data can be accessed concurrently
Pipelining	Whilst an instruction is being executed the next can be decoded and the subsequent one fetched
Use of Cache	A small amount of high performance memory which is next to the CPU It stores frequently used data and instructions
Virtual cores / Hyper-threading	This is where a physical core can act as two virtual cores

Multiple Cores	Each core acts as a separate processing unit
Onboard Graphics	Built-in circuitry for graphics processing
Performance boosting mode	The clock speed can be temporarily increased for a performance boost
Out of Order Execution	Instructions can be executed before earlier ones if they are ready
Super Scalar	Multiple instructions can be executed simultaneously


Your notes

Examiner Tips and Tricks

- You will not be asked about specific aspects of "contemporary processor architecture" apart from those on this page. You may be asked to show an awareness of how contemporary processors differ from a pure Von Neumann architecture in more open questions