OCR Computer Science A Level

1.1.1 Structure and Function of the Processor

Intermediate Notes

Specification:

1.1.1 a)

- The Arithmetic and Logic Unit
- The Control Unit
- Registers:
    - Program Counter
    - Accumulator
    - Memory Address Register
    - Memory Data Register
    - Current Instruction Register
- Buses:
    - Data Bus
    - Address Bus
    - Control Bus
- How these relate to assembly language programs

1.1.1 b)

- The Fetch-Decode-Execute Cycle
- Its effect on registers

1.1.1 c)

- The factors affecting the performance of the CPU:
    - Clock Speed
    - Number of Cores
    - Cache

1.1.1 d)

- The use of pipelining in a processor to improve efficiency

1.1.1 e)

- Von Neumann architecture
- Harvard architecture
- Contemporary processor architecture

Components of a Processor

The processor is the brain of a computer. It executes instructions which allows programs to run.

The Arithmetic and Logic Unit

The ALU (Arithmetic and Logic Unit) carries out all the arithmetical and logical operations.

The Control Unit

The Control Unit is a processor which directs the operations of the CPU. It has the following jobs:

- Controlling and coordinating the activities of the CPU
- Managing the flow of data between the CPU and other devices
- Accepting the next instruction
- Decoding instructions
- Storing the resulting data back in memory

Registers

Registers are small memory cells that operate at a very high speed. They are used to temporarily store data and all arithmetic, logical and shift operations occur in these registers.

Registers	Purpose
Program Counter (PC)	Holds the address of the next instruction to be executed.
Accumulator (ACC)	Stores the results from calculations
Memory Address Register (MAR)	Holds the address of a location that is to be read from or written to.
Memory Data Register (MDR)	Temporarily stores data that has been read or data that needs to be written.
Current Instruction Register (CIR)	Holds the current instruction being executed, divided up into operand and opcode.

Buses

Buses are a set of parallel wires which connect two or more components inside the CPU. There are three buses in the CPU: data bus, control bus, and address bus. These buses collectively are called the system bus.

The width of the bus is the number of parallel wires the bus has. The width of the bus is directly proportional to the number of bits that can be transferred simultaneously at any given time. buses are typically 8, 16, 32 or 64 wires wide.

Data Bus

This is a bi-directional bus (meaning bits can be carried in both directions) used for transporting data and instructions between components.

Address Bus

This is the bus used to transmit the memory addresses specifying where data is to be sent to or retrieved from. The width of the address bus is proportional to the number of addressable memory locations.

Control Bus

This is a bi-directional bus used to transmit control signals between internal and external components.

The control signals include:

- Bus request: shows that a device is requesting the use of the data bus
- Bus grant: shows that the CPU has granted access to the data bus
- Memory write: data is written into the addressed location using this bus
- Memory read: data is read from a specific location to be placed onto the data bus,
- Interrupt request: shows that a device is requesting access to the CPU
- Clock: used to synchronise operations

Assembly language

Assembly code uses mnemonics to represent instructions, for example ADD represents addition. This is a simplified way of representing machine code.
The instruction is divided into operand and opcode in the Current Instruction Register. The operand contains the data or the address of the data upon which the operation is to be performed. The opcode specifies the type of instruction to be executed.

Synoptic Link

Assembly Language is the language used in **Little Man Computer** and the **Addressing Mode** specifies how the operand is used.

This is covered in more detail in **1.2.4 Types of Programming Languages.**

Operation Code								Operand							
Machine Code Operation						Addressing mode									
0	1	0	0	0	1	0	1	0	0	1	1	0	1	0	1

A Level only

Pipelining

Pipelining is the process of completing the fetch, decode, and execute cycles of three separate instructions simultaneously, holding appropriate data in a buffer in close proximity to the CPU until it's required. While one instruction is being executed, another can be decoded and another fetched. Pipelining is aimed to reduce the amount of the CPU which is kept idle.

Fetch-Decode-Execute Cycle and Registers

The fetch-decode-execute cycle is the sequence of operations that are completed in order to execute an instruction.

Fetch phase:
- Address from the PC is copied to the MAR
- Instruction held at that address is copied to MDR by the data bus
- Simultaneously, the contents of the PC are increased by 1
- The value held in the MDR is copied to the CIR

Decode phase:
- The contents of CIR are split into operand and opcode

Execute phase:
- The decoded instruction is executed

Factors affecting CPU performance

There are three factors that affect CPU performance: clock speed, number of cores and the amount and type of cache memory.

Clock speed

The clock speed is determined by the system clock. This is an electronic device which generates signals, switching between 0 and 1. All processor activities begin on a clock pulse, and each CPU operation starts as the clock changes from 0 to 1. The clock speed is the time taken for one clock cycle to complete.

Number of cores

A core is an independent processor that is able to run its own fetch-execute cycle. A computer with multiple cores can complete more than one fetch-execute cycle at any given time. A computer with dual cores can theoretically complete tasks twice as fast as a computer with a single core. However, not all programs are able to utilise multiple cores efficiently as they have not been designed to do so, so this is not always possible.

Amount and type of Cache Memory

Cache memory is the CPU's onboard memory. Instructions fetched from main memory are copied to the cache, so if required again, they can be accessed quicker. As cache fills up, unused instructions are replaced.

Computer Architecture

Von Neumann Architecture

This architecture includes the basic components of the computer and processor (single control unit, ALU, registers and memory units) in which a shared memory and shared data bus is used for both data and instructions. Von Neumann architecture is built on the stored program concept.

Harvard Architecture

Harvard architecture has physically separate memories for instructions and data, more commonly used with embedded processors.This is useful for when memories have different characteristics, i.e. instructions may be read only, while data may be read-write.

Advantages of Von Neumann Architecture	Advantages of Harvard Architecture
Cheaper to develop as the control unit is easier to design	Quicker execution as data and instructions can be fetched in parallel.
Programs can be optimised in size	Memories can be different sizes, which can make more efficient use of space

Contemporary Processing

Contemporary processors use a combination of Harvard and Von Neumann architecture. Von Neumann is used when working with data and instructions in main memory, but uses Harvard architecture to divide the cache into instruction cache and data cache.