1		**Marks are for AO1 (understanding)** • The address of the memory to be written to is placed on the address bus (by the processor); • The data to be written is placed on the data bus (by the processor); • The signal to write is placed on the control bus (by the processor); • The control bus carries a clock signal (to synchronise the memory and processor); • When the write signal is received (by the memory) on the control bus; the data from the data bus is stored; into the location identified by the address bus; **A.** CPU for processor **NE.** Implication that the busses are doing the 'sending' rather than 'carrying' of data / addresses / signals **MAX 2** per bus **MAX 3** if only two buses referenced **MAX 4** marks	4

2	1	**2 marks are for AO1 (understanding)** When data/instructions are needed/fetched they have to be transferred from memory to the processor (using the data bus); (after execution) result/data may need to be transferred back to memory (using the data bus); **A.** responses referring to I/O controllers instead of memory	2
2	2	**2 marks are for AO1 (understanding)** **In the Harvard architecture:** Instructions and data have separate buses; Instructions and data are stored in separate memories // Instructions and data have separate memory/address spaces; **NE.** Places, locations, registers, areas of memory Instruction word size can be different to data word size // Instruction bus width can be different to data bus width; Instructions and data can be fetched simultaneously; **A.** points made in reverse that state how the von Neumann architecture works **MAX 2**	2

Qu	Pt	Marking Guidance	Marks
3	1	**Marks are for AO1 (understanding)** Main memory stores the <u>instructions</u> to be executed (and any data required by those instructions); Main memory returns the instructions / data / value stored in a memory location (specified on the address bus) (using the data bus); Program is transferred from secondary storage into main memory (if program not already in main memory) when program execution is requested; Main memory stores any value / data resulting from the execution of the program; **MAX 2**	2

Qu	Pt	Marking Guidance	Marks
3	2	**Marks are for AO1 (understanding)** The address bus; Width increased by 1;	2

4	**1**	**Mark is AO2 (apply)** 4294967296 // 2^{32} (bytes)	**1**

5	**1**	**2 marks for AO1 (understanding)** | **Component Name**	**Component Number (1–5)**
---	---
Address Bus	4
Data Bus	5
Main Memory	1
Processor	2
USB I/O Controller	3	 **1 mark:** At least three components correctly numbered **2 marks:** All five components correctly numbered	**2**

5	**2**	**2 marks for AO1 (understanding)**
		
		Avoid/reduce bottleneck of single data/address bus(es) // avoid/reduce delays waiting for memory fetches; **A.** Instruction and data can be accessed simultaneously;
		
		Avoids possibility of data being executed as code (which is one method that can be exploited by hackers) // Being able to use exclusively ROM for instruction memory prevents the program being modified/hacked; **A.** program cannot be (accidentally) overwritten (by data)
		
		Instruction and data memory can have different word lengths;
		
		Different technologies can be used to implement instruction and data memory;
		
		Different quantities of instruction and data memory means that address lengths can differ between the two // memory address structures can differ;
		
		Max 2
		
		NE. So programs/tasks will run faster
		NE. More efficient
		NE. Quicker access, without further explanation
		NE. Instructions and data stored in different memories
			2

Question			Marks
6	1	**Mark is AO1 (knowledge)** **C** The stored program concept; **R.** if more than one lozenge shaded	**1**

Qu	Pt	Marking guidance	Total marks
7	1	**Mark is AO1 (understanding)** Address (bus);	1

Qu	Pt	Marking guidance	Total marks
7	2	**Mark is AO1 (knowledge)** Receiving and transmitting components share a common clock // are (continuously) synchronised by a common clock; **A.** receiver and transmitter **A.** "communicating components" without reference to receiver and transmitter Timing information transmitted within / alongside the data; Receiver and transmitter components clocks are (continuously) synchronised; **A.** receiver and transmitter **A.** "communicating components" without reference to receiver and transmitter **NE.** receiver and transmitter are synchronised **TO.** stated that synchronisation is only when data is transmitted **Max 1**	1

Qu	Pt	Marking guidance	Total marks
7	3	**All marks are AO1 (understanding)** **1 mark** for identifying a difference between communicating with peripherals and between components inside the computer: • data has to travel further to a peripheral // data travels a shorter distance between internal components • position of internal components is fixed // peripherals may be moved • more data is transmitted between internal components than to peripherals // data is transmitted more frequently between internal components than to peripherals // data must be transmitted at a higher rate between internal components than to peripherals. **1 mark** for giving a reason why internal buses are parallel: • more data / multiple bits can be transmitted simultaneously / at the same time. **A.** faster transmission **1 mark** for giving a reason why serial is used for connecting peripherals: • data skew cannot occur • crosstalk cannot occur • data transmission speed (on one wire) can be higher • cabling is cheaper • cabling allows more flexibility over positioning • cables can be longer. **Award marks for differences stated in reverse, eg serial communication has slower transmission is equivalent to parallel faster transmission, parallel cabling more expensive is equivalent to serial cabling cheaper.**	**3**

Qu	Pt	Marking guidance	Total marks
7	4	**All marks are AO1 (understanding)** Allows processor to control / communicate with a peripheral using an (I/O) port; Allows peripheral to appear as a set of registers / memory locations (to the processor); Translates signals / data received from a peripheral into a form that can be processed by the computer // translate signals / data sent by the processor into a form that can be used by a peripheral // converts voltages used by processor and I/O device (if they operate using different voltages); Buffer data being received from a peripheral (so the processor does not have to wait for it); Allows new peripherals to be added without having to redesign the processor / computer hardware; Allows peripheral designers to create new peripherals to one common interface standard // allows peripheral designer to reduce / minimise the number of ports/connections that a peripheral needs to support; To carry out some of the I/O related processing // to reduce the workload on the CPU in relation to I/O processing; To check that data received from peripherals is not corrupted // performs error detection/correct on data received from peripherals; Implements the protocols used by I/O devices for communication; Generates an interrupt when data is ready to be transferred from an I/O device // when an I/O device needs the immediate attention of the CPU; **Max 2**	**2**

Qu	Pt	Marking guidance	Total marks
7	5	**All marks AO2 (analyse)** Award **2 marks** for correct answer: 128 // 2^7 (gibibytes). If answer is incorrect then award **1 method mark** for at least **two** of: • including 2^{36} in the calculation • multiplying by 16 • dividing by 8 • dividing by 1024 // 2^{10} • dividing by 1024 // 2^{10} a second time. Or award **1 method mark** for **one** of: • including 2^{37} in the calculation • multiplying by 2 (an alternative to multiplying by 16 then dividing by 8) • dividing by 1048576 // 2^{20} • dividing by 1073741824 // 2^{30}	**2**

Qu	Pt	Marking guidance	Total marks
7	6	**Mark is AO1 (knowledge)** Indicate that a memory write is occurring // cause data on the data bus to be written into the memory / RAM; Transfer clock signal // synchronise operation of processor and memory / RAM; Indicate the number of bits being transferred; Receive transfer acknowledgement // receive acknowledgement that data received (by memory/ RAM); Send signal to request use of (system) bus // issue bus request; Receive signal granting use of (system) bus // receive bus grant; **Max 1**	**1**