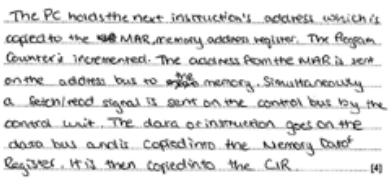
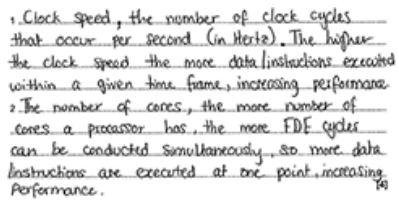
# Mark scheme

| Question | | | Answer/Indicative content | Marks | Guidance |
|---|---|---|---|---|---|
| 1 | a | | 1 mark per bullet to max 4:<br>• The **contents** of the Program Counter/PC are **copied/sent** to the Memory Address Register/ MAR<br>• The address is **sent/transferred** along the address bus<br>• The control unit **sends/transfers** a (read) signal along the control bus<br>• Contents stored in the memory address are **sent/transferred** along the data bus<br>• Contents (from memory) are stored in the Memory Data Register/MDR<br>• ...and **sent/copied** to the Current Instruction Register/ CIR<br>• The Program Counter/PC is incremented | 4 | Accept IR/Instruction Register for CIR/Current instruction Register<br><br>Accept MBR/Memory Buffer Register for MDR/Memory Data Register<br><br>**Examiner's Comments**<br><br>Candidates who had good knowledge of the FDE cycle were able to gain 3 or 4 marks on this question. Some candidates did not have a good understanding of what a register or a bus is and tended to say they did something other than transport or temporarily store. Many however gave clear and correct responses.<br><br>**Misconception**<br><br>Some candidates gave responses which mentioned a register fetching the data or passing data. A register is a temporary store for data/instructions/addresses and as such doesn't fetch or pass anything.<br><br>Exemplar 1<br><br>  <br>The PC holds the next instruction's address which is copied to the MAR, memory address register. The Program Counter is incremented. The address from the MAR is sent on the address bus to memory. Simultaneously a fetch/read signal is sent on the control bus by the control unit. The data or instruction goes on the data bus and is copied into the Memory Data Register. It is then copied into the CIR. [4]<br><br>This candidate has shown a clear understanding of the role of the buses and registers in the fetch part of the FDE cycle. |
| | b | | • Program Counter / PC | 1 | **Examiner's Comments**<br><br>Candidates who understood the FDE |

<table>
<tr><td></td><td></td><td></td><td></td><td></td><td>cycle gained the mark. Those that did not tended to mention a register used in the fetch or decode stages.</td></tr>
<tr><td></td><td>c</td><td></td><td>1 mark per bullet:<br><br>• Allows the next <u>instruction</u> to be fetched whilst the previous one is being decoded/executed /allows the overlapping of different parts of the FDE<br>• It increases throughput / increases the number of instructions processed in a set period of time<br>• It prevents the CPU having to wait / prevents idle components</td><td>3</td><td>DNA responses that talk about the FDE running faster.<br>Allow a diagram demonstrating pipelining for MP1.<br><br>Allow ‘it will take less time to do the same amount of instructions’ for MP2<br><br>BOD ‘processes’ for ‘instructions’ for MP2.<br>DNA ‘more efficient’ on its own for MP2<br><br>DNA points if clearly discussing multiple cores.<br><br><u>Examiner’s Comments</u><br><br>Many candidates were able to gain a mark for an instruction being fetched at the same time as another is decoded. Some candidates discussed multiple cores which were mentioned in the question stem but did not apply to this question. Candidates should have access to previous mark schemes.</td></tr>
<tr><td></td><td></td><td></td><td>Total</td><td>8</td><td></td></tr>
<tr><td>2</td><td></td><td></td><td>1 Mark for each labelled component.<br><br><table><tr><td>A</td><td>Data Bus</td></tr><tr><td>B</td><td>MAR / Memory Address Register</td></tr><tr><td>C</td><td>PC / Program Counter</td></tr><tr><td>D</td><td>MDR / Memory Data Register</td></tr><tr><td>E</td><td>ACC / Accumulator</td></tr></table></td><td>AO1.1<br>(5)</td><td>Accept MBR / Memory Buffer Register for D<br><br><u>Examiner’s Comments</u><br><br>This question was generally answered well. Most candidates were able to identify the data bus, Memory Address Register (MAR) and Memory Data Register (MDR), although some candidates were not able to identify the Program Counter (PC) and/or the Accumulator (ACC).</td></tr>
<tr><td></td><td></td><td></td><td>Total</td><td>5</td><td></td></tr>
<tr><td>3</td><td></td><td></td><td>Mark Band 3- High Level<br>(7-9 marks)<br>The candidate demonstrates a thorough knowledge and understanding of both Von Neumann</td><td>AO1.1<br>(2)<br>AO1.2<br>(2)<br>AO2.1</td><td>AO1<br>Harvard Architecture:<br><br>• Uses separate memory for data and instructions.</td></tr>
</table>

| | | | | | |
|---|---|---|---|---|---|
| | | | and Harvard architectures.<br>All detail is generally accurate and relevant.<br>The candidate is able to apply their knowledge and understanding directly and consistently to the context provided.<br>Evidence/examples will be explicitly relevant to the explanation.<br>The candidate will come to a clear conclusion that must be justified by their comments.<br><br>*There is a well-developed line of reasoning which is clear and logically structured. The information presented is relevant and substantiated.*<br><br>**Mark Band 2- Mid Level (4-6 marks)**<br>The candidate demonstrates reasonable knowledge and understanding of Von Neumann and Harvard architectures; the material is generally accurate but at times underdeveloped. The candidate may not have applied both to this scenario.<br>The candidate is able to apply their knowledge and understanding directly to the context provided although one or two opportunities are missed.<br>Evidence/examples are for the most part implicitly relevant to the explanation.<br>The candidate will attempt to come to a conclusion, although it may not be fully justified by their answer.<br><br>*There is a line of reasoning presented with some structure. The information presented is in the most part relevant and supported by some evidence.*<br><br>**Mark Band 1- Low Level (1-3 marks)**<br>The candidate demonstrates a basic knowledge of Von Neumann and Harvard architectures. The material is basic and contains some inaccuracies.<br>The candidate makes a limited attempt to apply acquired knowledge and understanding to the context provided. | (2)<br>AO3.3<br>(3) | • Uses separate data and address buses for each piece of memory<br>• Can read/write data and instructions simultaneously<br><br>Von Neumann Architecture:<br><br>• Uses one physical piece of memory for both data and instructions<br>• Uses one data and one address bus.<br>• Can only read/write data or instructions, not both at the same time<br><br>AO2<br><br>• Von Neumann design is less complex. Development of one is cheaper. Harvard design is more complex as it requires two buses. Development is more expensive. More CPU pins are required; a complex motherboard is required with doubling of memory.<br>• Harvard Architecture free data memory cannot be used for instructions and vice-versa<br><br>AO3<br><br>• Harvard architecture allows for faster processing as instructions and data are simultaneously fetched/executed.<br>• Harvard splits memory between instruction and data in a static way, meaning you could run out of one memory with unused memory in the other area.<br>• Von Neumann architecture allows for dynamic allocation between instruction and data |

| | | | | | |
|---|---|---|---|---|---|
| | | | The candidate provides nothing more than an unsupported assertion.<br><br>*The information is basic and communicated in an unstructured way. The information is supported by limited evidence and the relationship to the evidence may not be clear.*<br><br>**0 mark**<br>No attempt to answer the question or response is not worthy of credit. | | • Large programs with small data (Computer games) or small programs with large data (video editing) can be equally catered for<br><br>**Examiner's Comments**<br><br>Candidates were assessed on the quality of their extended response in this question. Most of the candidates were clear on the basic difference between Harvard and Von Neumann architecture, but many did not discuss this in detail or did not use the correct terminology. Some candidates confused the different architectures and contradicted points in their discussion.<br><br>Exemplar 1<br><br>  <br>Von Neumann and Harvard store data in different ways, they both has seperate memory and address buses, however Harvard has seperate memory locations for data and instruction while Von Neumann has data and both in the same memory.<br><br>a Von Neumann is good as it allows for programs to be run and memory is joint, data and instructions can split memory in an efficient ways, reducing bottlenecks due to one being full like it can happen in Harvard architecture. Also Von N Most programs are written to support a Von Neumann architecture therefore it's easier to set up and use has it has more support from developers. Better for multipurpose system.<br><br>Harvard is less complex and memory management is simpler. It's good for custom programs as it can run 100 faster if code is written specifically. Very good for embedded systems which software isn't changed.<br>In conclusion they're both useful while Von Neumann is more widely used and better for general purpose computers, Harvard architecture can be good for more simple and specific cases. Such as embedded systems, or machine which do not need a general purpose CPU. ✓5<br><br>This candidate response was given 5 marks. There are some valid points about both Von Neumann and Harvard Architecture, but the points made are not fully developed with limited application. This response was therefore given a mark within the mid-level band. |

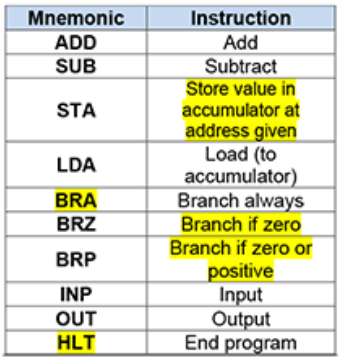
| | | | | | |
|---|---|---|---|---|---|
| | | | | | **Assessment for learning**<br><br>Questions with the command word 'discuss' require candidates to give a balanced discussion and provided a suitable conclusion which justifies their comments. Opportunities to practise these questions will support candidates to do better on these style questions. |
| | | | **Total** | **9** | |
| 4 | | | • Clock Speed …<br>• … The speed at which the fetch decode execute cycle is completed/ the speed a single core can execute instructions<br><br>• Number of cores/ independent processing units …<br><br>• …that can fetch decode execute **at the same time**<br><br>• Cache size…<br>• … memory that contains recently/frequently used instructions/data<br>• …memory that has a faster R/W speed than RAM<br>• …memory that is closer to/onboard the CPU | AO1.1 (2)<br>AO1.2 (2) | One mark for stating the factor, mark for expanding the factor<br><br>- Accept cycles for "FDE Cycles"<br><br>**Examiner's Comments**<br><br>Most candidates were able to identify two factors that could affect the performance of the CPU, but many were not given all 4 marks as they did not fully describe why each factor would affect the performance. See Exemplar 1, which was given full marks.<br><br>Exemplar 1<br><br>  <br>1. Clock Speed, the number of clock cycles that occur per second (in Hertz). The higher the clock speed the more data/instructions executed within a given time frame, increasing performance<br>2. The number of cores, the more number of cores a processor has, the more FDE cycles can be conducted simultaneously, so more data/instructions are executed at one point, increasing performance.<br><br>The candidate gives two valid factors that would affect performance and then explains why each of the two factors would affect performance. They use appropriate terminology in their response. |
| | | | **Total** | **4** | |
| 5 | a | | • **Higher/faster** clock speed<br>• **More** cores/dual/quad/etc core<br>• **More** cache memory. | 2 | Answers must refer to an improvement (more/higher/faster) not just "change the clock speed" Allow discussions of level 1/level 2 cache |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | sizes for one mark.<br><br>Accept valid features of CPUs that would improve performance e.g. Use of:<br>Pipelining<br>Simultaneous Multithreading<br>Do not accept RISC/CISC.<br><br>**Examiner's Comments**<br>Most candidates were able to gain full marks on this question. Less successful responses often mentioned clock speed, cache or cores without referring to an improvement, e.g. higher or faster. |
| | b | i | • Holds all input/output<br>• Holds **results** of calculations (from the ALU)<br>• Checked for conditional branching (e.g. BRZ)<br>• Stores data which has come from the MDR/RAM | 2 | **Examiner's Comments**<br>Most candidates were able to access 1 mark for the result of ALU calculations, but few were able to give two uses. Some confused the accumulator with the program counter and the ALU. |
| | | ii | • Holds the address/location of the next instruction (to be executed/fetched)<br>• Contents copied to the MAR at start of FDE<br>• Incremented (by one) on every cycle<br>• Can be changed by branch/jump instructions | 2 | **Examiner's Comments**<br>This question was generally well answered by candidates who gave clear responses.<br><br>**Misconception**<br>Some candidates thought that the program counter kept track of a count of the number of instructions that had been fetched. |
| | | iii | • Memory Address Register / MAR<br>• Memory Data Register / MDR<br>• Current Instruction Register / CIR<br>• Index Register / IR | 3 | Allow Memory Buffer Register for MDR<br><br>**Examiner's Comments**<br>Most candidates gained full marks on this question and were able to correctly identify three other registers. Some lost marks for saying the ALU or control unit were registers. |
| | c | i | • Both data and instructions share the same memory | 2 | **Examiner's Comments** |

| | | | | | |
|---|---|---|---|---|---|
| | | | • Instructions and Data stored in same format<br>• A single set of buses / same bus for instructions & data (to connect CPU to Memory and I/O)<br>• Has a (single) control unit<br>• Has an ALU.<br>• Has ways to input and output.<br>• Has access to storage,<br>• Works sequentially through instructions / follows Fetch-execute cycle<br>• (Special) registers within CPU<br>• Based on stored program concept | | Many candidates were able to access full marks on this question. This question has been asked in previous papers and candidates should be encouraged to use these to make sure they are clear in their responses. There were many possible responses in the mark scheme to help candidates to gain full marks. Most candidates gained at least 1 mark. |
| | | ii | • Separate memory for data and instructions / Multiple memory units<br>• Different (sets of) buses one for instructions & one for data/ instructions and data can be accessed concurrently. | 1 | **Examiner's Comments**<br>This question was generally answered well by candidates and the majority gave separate areas of memory for data and instructions. Where candidates were not given marks, it was generally because their answer was unclear, e.g. just saying 'separate memory'. |
| | | | **Total** | **12** | |
| 6 | | | 1 mark for any of the following bullet points, e.g:<br>• Computers use binary logic for on/off or 1/0<br>• Computer systems are based on switches/transistors<br>• Binary is high tolerance | 1 (AO1.2) (1) | **Examiner's Comments**<br>This question was generally answered well. |
| | | | **Total** | **1** | |
| 7 | | i | 1 mark per bullet up to a maximum of 2 marks, e.g:<br>• Uses separate memory blocks for instructions and data<br>• Has separate buses (data and address) for data and instructions<br>• Has fixed memory sizes for data and Instructions | 2 (AO1.1) (1) (AO1.2) (1) | Accept unit instead of blocks (BP1)<br>**Examiner's Comments**<br>Some candidates were vague in their response and were not clear that Harvard has separate memory blocks. Candidates must be specific in their response. |

| | | | | | |
|---|---|---|---|---|---|
| | | | • Instruction memory may be ROM | | |
| | | ii | 1 mark per bullet up to a maximum of 2 marks, e.g:<br>• Fixed instruction size<br>• No need for memory to be shared between data and instructions<br>• Removes need for secondary storage<br>• Instructions would never be changed | 2 (AO2.1) (2) Any 2 (Max 2) | **Examiner's Comments**<br>This question was not answered well. Many candidates repeated their response to the previous question and did not answer the question correctly. Candidates need to read questions carefully. |
| | | | **Total** | **4** | |
| 8 | | i | • Temporary storage/memory location…<br>• …inside the CPU<br>• Used for a single specific purpose<br>• Faster access speed than RAM / secondary storage | 2 AO1.1 | |
| | | ii | • Accumulator checked to see if value held is positive or zero<br>• If so, BRANCH carried out / jumps to specified location. | 2 AO1.2 | |
| | | | **Total** | **4** | |
| 9 | a | | • Concurrent processing of multiple instructions<br>• One instruction can be fetched while previous isbeing decoded…<br>• And the one before is being executed.<br>• In case of a branch pipeline is flushed.<br>• Increases speed of execution | 3 AO1.1 | |
| | b | i | • Clock speed<br>• Number of cores<br>• Cache | 2 AO1.1 | Accept<br>Use of pipelining/size of pipeline<br>Use of out of order execution<br>Use of SIMD instructions<br>Integrated graphics processing on CPU |
| | | ii | • RAM is volatile | 4 AO1.2 | |

| | | | | | |
|---|---|---|---|---|---|
| | | | • Used for storing programs/data/parts of OS currently in use<br>• ROM is non volatile<br>• Used for storing (e.g.) BIOS / bootstrap | | |
| | | iii | • Performing complex numerical calculations<br>• Calculations on matrices / vectors / multiple data atthe same time<br>• …e.g. insurance pricing, modelling risk, calculating bills | 2 AO2.2 | Example has to relate to insurance company |
| | | | **Total** | **11** | |
| 10 | a | | 1 mark for each interval<br><br>Interval 1<br><br>• A is fetched<br><br>Interval 2<br><br>• A is decoded<br>• B is fetched<br><br>Interval 3<br><br>• A is executed<br>• B is decoded<br>• C is fetched<br><br>Interval 4<br><br>• B is executed<br>• C is decoded<br>• D is fetched | 4AO1.2 (4) | |
| | b | | 1 mark per bullet up to a maximum of 2 marks:<br><br>• Reduces/removes latency<br>• … CPU is not idle while waiting for next instruction<br>• Next instruction is fetched while current one is decoded/executed | 2AO1.2 (2) | |

| | | | | | |
|---|---|---|---|---|---|
| | | | • All parts of the processor can be used at any instance in time. | | |
| | | | **Total** | **6** | |
| 11 | a | | • Store value in accumulator at address given<br>• BRA / BR<br>• Branch if zero<br>• Branch if <u>zero or positive</u><br>• HLT / COB / END | 5<br><br>AO1.1 |  <br> |
| | b | | • Inputs two numbers<br>• ..stores at least one of them<br>• Comparison / subtraction to decide which is larger<br>• Jump / output if num1 larger<br>• Jump / output if num2 larger <u>or nums equal</u><br>• Loops back to start after either output | 6<br><br>AO3.2 | <u>Example answer</u><br>start INP<br>STA x<br>INP<br>STA y<br>SUB x<br>BRP first<br>LDA x<br>OUT<br>BRA start<br>first LDA y<br>OUT<br>BRA start<br>x DAT<br>y DAT |
| | | | **Total** | **11** | |
| 12 | a | | 1 mark per bullet up to a maximum of 2 marks, e.g.<br><br>• Uses the same memory for data and instructions<br>• Uses the same bus for data and instructions<br>• Can only fetch either data or instructions at one time/follows FDE | 2<br>AO1.1 (1)<br>AO2.1 (1) | Allow:<br><br>• ALU for arithmetic Logic Unit<br>• CPU contains an Arithmetic Logic Unit<br>• CPU contains a single Control Unit.<br>• Same (Memory) **location** is not acceptable for BP1 |
| • | b | i | 1 mark per bullet up to a maximum of 4 marks, e.g.:<br><br>• Data/address is copied from PC to MAR | 4<br>AO1.1 (2)<br>AO2.1 (2) | The bullets must be in the correct order, except BP2, which can come anywhere from BP2 onwards |

| | | | | | |
|---|---|---|---|---|---|
| | | | • PC is incremented (by 1) (this can be in any location from here down)<br>• Data in MAR is passed onto the Address Bus<br>• Read signal is sent onto the control bus<br>• RAM copies the data from the location specified by the address bus onto the data bus<br>• Data on the data bus is passed into the MDR<br>• Data is copied from the MDR to the CIR | | |
| | | ii | • C | 1<br>AO1.2<br>(1) | |
| | | | **Total** | **7** | |