1	1	**Marks are for AO1 (understanding) and AO2 (analyse)** **Mark as follows:** **AO2 (analyse) – 1 mark:** The different processors have different instruction sets; **A.** Examples such as different numbers of general purpose registers / different architecture. **AO1 (understanding) – 1 mark:** The program is in machine code / platform dependent / makes use of those instructions; **A.** The program has been compiled **NE.** Not portable	**2**
1	2	**Marks are for AO1 (understanding) and AO2 (analyse)** **Mark as follows:** **AO2 (analyse) – 1 mark:** A processor with a clock speed of 3.2GHz may be able to execute (sequential) instructions more quickly than a processor with a clock speed of 2.8GHz; **AO1 (understanding) – 1 mark:** Where parallel processing is not possible / sequential processing is needed this may enable the 3.2GHz processor to complete the task sooner than the 2.8GHz processor; **A.** 'Josephine's computer' for 3.2GHz processor and 'Ella's computer' for 2.8GHz processor	**2**

2	**1**	**Marks are for AO3 (design) and AO3 (program)** **Mark as follows:** **AO3 (design) – 1 mark** **1 mark** for identifying the need for two branch commands **AO3 (program) – 3 marks** For the AO3 (program) marks, the syntax used must be correct for the language as described on the question paper. **1 mark:** Subtracting 10 from R1 and storing the result in R1 **1 mark:** Adding 1 to R3 and storing the result in R3 **1 mark:** Having two branches with the correct condition(s) **Max 2** marks for programming if any syntax incorrect or program does not work correctly under all circumstances **DPT** incorrect use of commas, colons, semi-colons, etc. Note this does not apply to #. **Refer alternative answers to team leaders** - BLT end - SUB R1, R1, #10 - ADD R3, R3, #1 - B loopstart	**4**
2	**2**	**Mark is for AO1 (understanding)** 64 // 2^6;	**1**
2	**3**	**Mark is for AO1 (understanding)** 1024 // $2^{10;}$	**1**

3		**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

4	1	**Marks are for AO2 (apply) and AO3 (program)**
		
		`MOV R0, #9`
		`MOV R1, #12`
		`MOV R2, #0`
		`startloop: AND R3, R0, #1`
		`CMP R3, #1`
		`BNE jump`
		`ADD R2, R2, R1 // ADD R2, R1, R2`
		`jump:` **`LSR R0, R0, #1`**
		`LSL R1, R1, #1`
		`CMP R0, #0`
		`BEQ endloop`
		`B startloop`
		`endloop:`
		
		Alternative Answer 1:
		`LSL R1, R1, #1` could be replaced with `ADD R1, R1, R1`
		
		Alternative Answer 2:
		`BNE jump` could be replaced with:
		`BEQ doadd`
		`B jump`
		`doadd:`
		
		AO2 (analyse) – 2 marks
		
		1 mark: Recognising that logical shift (`LSR`/`LSL`) is needed to perform integer division by 2 / multiplication by 2 even if the syntax used is incorrect.
		
		1 mark: Recognise that two comparisons and two branch instructions are needed even if the syntax is incorrect or the wrong types of branch instructions are used.
		
		AO3 (program) – 5 marks
		
		1 mark: `CMP R3, #1` before the `jump:` label and syntactically correct.
		
		1 mark: `BNE jump` before the `jump:` label and syntactically correct.
		
		1 mark: `ADD R2, R2, R1` is before the `jump:` label and syntactically correct.
		
		1 mark: `LSR R0, R0, #1` **and** `LSL R1, R1, #1` are after the `jump:` label and syntactically correct. **I.** order of commands
		
		1 mark: `CMP R0,#0` and `BEQ endloop` are after the `jump:` label, before `B startloop` and syntactically correct.
		
		Max 4 marks for programming if any syntax incorrect or program does not work correctly under all circumstances
		
		A. Answers that use hexadecimal or binary values
		DPT Missing hash for immediate addressing
		DPT incorrect use of commas, colons, semi-colons, line numbers, etc.
			7

5	1	**Mark is for AO1 (knowledge)** A memory/storage location inside/on a processor; **A.** CPU instead of processor **NE.** memory/storage location	1
5	2	**2 marks are for AO1 (knowledge)** Instructions are stored in (main) memory; Instructions are fetched, (decoded) and executed (serially) by the processor; Programs can be moved in and out of main memory; **MAX 2**	2
5	3	**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
5	4	**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

5	5	**4 marks for AO1 (knowledge) and 4 marks for AO1 (understanding)** **Description** / **Explanation** Contents of the Program Counter / PC transferred to the Memory Address Register / MAR — so that the PC can be updated // to enable the memory address to be transferred along the address bus/to the memory Contents of MAR placed onto address bus — so the correct location in the main memory will be accessed Contents of addressed memory location/value received on data bus loaded into the Memory Buffer Register / MBR — not all fetches will be for instructions so cannot be loaded directly into Current Instruction Register / CIR // the value will only be present transiently on the bus so must be stored in a register // the MBR is used to cope with the speed difference between the processor and the main memory (Contents of) PC is incremented — so that the next instruction in the sequence can be fetched The contents of the MBR is copied to the CIR — so that if data is fetched/written during the execute phase it does not overwrite the instruction // because the control unit uses the instruction from the CIR **A.** Memory Data Register/MDR for Memory Buffer Register/MBR **Max 4** for descriptions **Max 4** for explanations **Max 8**	8

6	1	**4 marks for AO3 (programming)**
		
		Example 1:
		`LDR R0, 100`
		`LDR R1, 101`
		`ADD R2, R0, R1`
		`CMP R2, #26`
		`BLT store`
		`SUB R2, R2, #26`
		`store: STR R2, 102`
		
		Example 2:
		`LDR R0, 100`
		`LDR R1, 101`
		`ADD R2, R0, R1`
		`CMP R2, #25`
		`BGT adjust`
		`STR R2, 102`
		`HALT`
		`adjust: SUB R2, R2, #26`
		`STR R2, 102`
		
		Example 3:
		`LDR R0, 100`
		`LDR R1, 101`
		`ADD R2, R0, R1`
		`CMP R2, #25`
		`BGT adjust`
		`B end`
		`adjust: SUB R2, R2, #26`
		`end: STR R2, 102`
		
		A. Use of alternative registers
		A. Any label name in place of store / adjust
		
		DPT. Use of invalid register name eg Rd
		DPT. Use of incorrect addressing mode
		DPT. Inclusion of invalid symbols in commands
		
		Programming Marks:
		1 Mark for `LDR R0, 100`, `LDR R1, 101` and `STR R2, 102`
		1 Mark for `ADD R2, R0, R1`
		1 Mark for `SUB R2, R2, #26`
		1 Mark for either:
		
		• `CMP R2, #26`, `BLT store` and `store:` aligned to a `STR` instruction **or**
		• `CMP R2, #25`, `BGT adjust` and `adjust:` aligned to a `SUB` instruction
		
		Max 3 if any errors.
			4

6	2	**Mark is for AO1 (understanding)** The operand is the datum;	1

6	3	**Mark is for AO1 (understanding)** Frequency/statistical/syntactical analysis cannot provide clues to the plaintext // nothing can be learnt about the plaintext from the ciphertext;	1

<table>
<tr><th>Qu</th><th>Pt</th><th>Marking Guidance</th><th>Marks</th></tr>
<tr><td>7</td><td>1</td><td>Marks are for AO1 (understanding)

<table>
<tr><th></th><th>Description</th><th>Order (1 to 4)</th></tr>
<tr><td>A</td><td>The contents of the MBR are copied to the CIR.</td><td>2</td></tr>
<tr><td>B</td><td>The contents of the PC are copied to the MAR.</td><td>1</td></tr>
<tr><td>C</td><td>The Control Unit decodes the contents of the CIR.</td><td>3</td></tr>
<tr><td>D</td><td>The result of the calculation is stored.</td><td>4</td></tr>
</table>

3 marks for all correct
2 marks for two correct
1 mark for one correct

R. Labels used more than once.</td><td>3</td></tr>
</table>

<table>
<tr><th>Qu</th><th>Pt</th><th>Marking Guidance</th><th>Marks</th></tr>
<tr><td>7</td><td>2</td><td>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</td><td>2</td></tr>
</table>

<table>
<tr><th>Qu</th><th>Pt</th><th>Marking Guidance</th><th>Marks</th></tr>
<tr><td>7</td><td>3</td><td>Mark is for AO1 (knowledge)

Arithmetic logic unit // ALU;</td><td>1</td></tr>
</table>

Qu	Pt	Marking Guidance	Marks
7	4	**Mark is for AO1 (understanding)** Increases the amount of data that can be transferred over the bus at once; **A.** Fewer transfers are needed to transfer the same amount of data; **NE.** Data can be transferred quicker / more data per unit of time. **NE.** More data can be transferred. **MAX 1**	1

Qu	Pt	Marking Guidance	Marks
7	5	**Marks are for AO1 (understanding)** The address bus; Width increased by 1;	2

Qu	Pt	Marking Guidance	Marks
8	1	**Mark is for AO1 (understanding)** **B** `MOV R3, #42;` **R.** More than one lozenge shaded.	1

Qu	Pt	Marking Guidance	Marks

8	2	**Marks are for AO3 (programming)**
		
		1 mark for each program point
		
		• Loading value from 101 into R1 (eg LDR R1, 101).
		• Comparing R1 against the operand 50 (eg CMP R1, #50).
		• Branching using BGT and BEQ, or BLT, with a suitable label.
		• Using a logical shift left to double the number (eg LSL R1, R1, #1).
		• Storing the value (even if incorrect) in R1 back to memory location 101
		
		Max 3 marks for programming if any syntax incorrect or program does not work correctly under all circumstances.
		
		Max 4
		
		Example 1:
		
		`LDR R1, 101`
		`CMP R1, #50`
		`BGT end`
		`BEQ end`
		`LSL R1, R1, #1`
		`STR R1, 101`
		`end:`
		
		Example 2:
		
		`LDR R1, 101`
		`CMP R1, #50`
		`BLT lessThan`
		`HALT`
		`lessThan:`
		`LSL R1, R1, #1`
		`STR R1, 101`
		
		Example 3:
		
		`LDR R1, 101`
		`CMP R1, #50`
		`BLT Double`
		`B EndIf`
		`Double:`
		`LSL R1, R1, #1`
		`STR R1, 101`
		`EndIf:`
		A. Use of any valid register number 0-12 instead of R1 .
		A. Use of comparisons that achieve the same result (eg greater than 49).
		A. Any label names.
		A. Alternative methods for doubling a number.
		A. Inline label names.
		I. Missing commas.
			4

Qu	Pt	Marking Guidance	Marks
9	1	**Marks are for AO1 (understanding)** **Max 1 mark for explanation:** Provides information about the result of the last (arithmetic/logical) instruction // to control conditional branch instructions; **Max 1 mark for example:** When the result of a comparison / (arithmetic) operation is zero/negative; When a carry needs to be carried out; When overflow/underflow occurs; When an interrupt occurs; When a comparison is made a flag is set as to whether the operands were equal;	2

Qu	Pt	Marking Guidance	Marks

10		**Marks are for AO3 (programming)**
		
		Mark as follows:
		
		AO3 (programming) – 4 marks
		For the AO3 (program) marks, the syntax used must be correct for the language as described on the question paper.
		
		1 mark: Comparing R2 against #0 and having a BGT/BEQ, or #1 and having a BLT
		1 mark: Adding R1 to R3 and storing result in R3
		1 mark: Subtracting #1 from R2 and storing result in R2
		1 mark: Having a B to branch to the start
		
		Max 3 marks for programming if any syntax incorrect or program does not work correctly under all circumstances.
		
		DPT. incorrect use of commas, colons, semi-colons, etc. Note this does not apply to #
		
		Note: HALT is not needed if on final line.
		
		Refer alternative answers not shown to team leaders
		
		Alternative answer 1
		`start: CMP R2, #0`
		`BGT addone`
		`HALT`
		`addone: ADD R3, R3, R1`
		`SUB R2, R2, #1`
		`B start`
		
		Alternative answer 2
		`start: CMP R2, #1`
		`BLT end`
		`ADD R3, R3, R1`
		`SUB R2, R2, #1`
		`B start`
		`end: HALT`
		
		Alternative answer 3
		`start: CMP R2, #0`
		`BEQ end`
		`ADD R3, R3, R1`
		`SUB R2, R2, #1`
		`B start`
		`end: HALT`
			4

		Alternative answer 4
		`start: CMP R2, #0`
		`BGT addone`
		`B end`
		`addone: ADD R3, R3, R1`
		`SUB R2, R2, #1`
		`B start`
		`end: HALT`	

Qu	Pt	Marking Guidance	Marks
11		**Marks are for AO2 (analyse)** The processor must keep pace with a wide range of sensors, each frequently collecting data; All sensor data goes via the processor // all sensor data requires computation; Processor must run other software at the same time as collecting data from sensors // the processor must operate quickly enough to support multitasking between processing sensor data and the applications (playing music / loading images); Both the image and music (often) have large file sizes; **NE.** faster processing. **MAX 2**	2

Qu	Pt	Marking Guidance	Marks
12	1	**Marks are for AO1 (understanding)** **1 mark** for two or three components correctly identified **2 marks** for four components correctly identified **1**: Memory Address Register **2**: Address Bus **3**: Memory Buffer Register **A.** Memory Data Register **4**: Data Bus	2

Qu	Pt	Marking Guidance	Marks
12	2	**Marks are for AO1 (knowledge)** (Machine code) Instructions are stored in (main) memory; Instructions are fetched, (decoded) and executed (serially) by the processor; Programs can be moved in to (and out of) main memory; **Max 2**	2

Qu	Pt	Marking Guidance	Marks
12	3	**Marks are for AO1 (knowledge)** Register (number); Memory address / location; **A.** offset from a memory location **Max 2**	2

Qu	Pt	Marking Guidance	Marks
12	4	**Marks are for AO1 (understanding)** Increases the probability/likelihood/chance that data/instructions will be found in cache (and cache memory is faster than main memory); **A.** Increases probability/likelihood/chance of cache hit (without cache hit definition) **A.** Fewer accesses to slower memory types, eg main memory **A**. More instructions can be accessed from high speed memory Allows for more bits to be simultaneously processed (in the execution of a single instruction) // Allows for more bits to be simultaneously transferred (within the processor);	2

Qu	Pt	Marking Guidance	Marks
13		**Marks are for AO3 (programming)** **1 mark** each for each program point: • Comparing the values in R1 and R3 **A.** Indirect comparisons • Using a branch instruction to execute different blocks of code. • Always terminating with the greater number stored in R1. • Terminating with 1 stored in R2 when the greater number was in R1 and 3 stored in R2 otherwise. **Max 3** marks for programming if any syntax incorrect or program does not work correctly under all circumstances	4

Example 1

```
          CMP R1, R3
          BGT r1bigger
          MOV R1, R3
          MOV R2, #3
          B   Done
r1bigger: MOV R2, #1
done:     HALT
```

Example 2

```
          SUB R2, R1, R3
          CMP R2, #0
          BGT finish
          MOV R2, #1
          B   done
finish:   MOV R1, R3
          MOV R2, #3
done:     HALT
```

Example 3

```
          MOV R2, #1
          CMP R1, R3
          BGT done
          MOV R1, R3
          MOV R2, #3
done: HALT
```