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
```

14	1	**2 marks AO2 (analysis) and 6 marks AO3 (programming)**
		
		Example Solution 1
		
		`  LDR R1, 102`
		`  LDR R2, 103`
		`loop:`
		`  CMP R1, R2`
		`  BEQ finish`
		`  BGT agreaterthanb`
		`  SUB R2, R2, R1`
		`  B loop`
		`agreaterthanb:`
		`  SUB R1, R1, R2`
		`  B loop`
		`finish:`
		`  STR R1, 104`
		
		Example Solution 2
		
		`  LDR R0, 102`
		`  LDR R1, 103`
		`startloop:`
		`  CMP R0, R1`
		`  BEQ end`
		`  CMP R0, R1`
		`  BGT greater`
		`  SUB R1, R1, R0`
		`  B startloop`
		`greater:`
		`  SUB R0, R0, R1`
		`  B startloop`
		`end:`
		`  STR R1, 104`
		
		Note: Any register numbers can be used and any understandable method to identify a label.
		
		DPT use of invalid register names eg R27, Rn
		
		6 marks AO3 (programming syntax must be correct):
		
		Values in memory locations 102 and 103 loaded into two different registers;
		
		Comparison made between the values in the two registers;
		
		If the values in the two registers are the same then the code will exit (after performing any other necessary instructions); **A.** end of program reached if not HALT instruction.
			8

		If A is greater than B then the value in the register representing B is subtracted from the value in the register representing A and result stored in register representing A; **Note:** Award this mark even if further incorrect changes would also be made to values in registers. If A is less than (or equal to B) / then the value in the register representing A is subtracted from the value in the register representing B and result stored in register representing B; **Note:** Award this mark even if further incorrect changes would also be made to values in registers. Before the algorithm exits, in all circumstance, the value in the register representing A (or the register representing B) is stored into memory location 104 (regardless of whether or not this is the gcd); **A.** if this is done on every iteration of a loop instead of just once. **2 marks AO2 (concept must be understood, syntax need not be correct):** The need for a loop has been identified and instructions are used to make the program loop back to before the comparison(s) after each subtraction has taken place; The response provided follows the correct method to calculate the gcd of A and B, regardless of whether the syntax is correct or not, although an attempt must have been made to use the AQA instruction set; **Max 7 if solution not fully working**	

15	**1**	**Mark is AO1 (knowledge)** Increase the number of bits // amount of data that can be transferred <u>at one time</u>; **A.** in one cycle **NE.** increase rate of data transfer	**1**

16	**1**	**2 marks for AO1 (knowledge)** | Number	Register Name
---	---
❶	Memory Address Register **NE.** MAR
❷	Program Counter **NE.** PC
❸	Current Instruction Register **NE.** CIR, IR **A.** Instruction Register	 **1 mark:** Two registers correctly named OR **2 marks:** All three registers correctly named If student has used initialisms instead of full register names (or a mixture of both) then award **1 mark** if all three registers are given the correct abbreviated name.	**2**

16	**2**	**2 marks for AO1 (knowledge)** Allows the currently executing process/task/program to be suspended; **A.** "stopped" as BOD **R.** Suspend/stop the fetch-execute cycle / processor **R.** "instruction" for "process" So that a device/source that needs the (immediate) attention of the processor can be serviced/dealt with // so that an <u>urgent</u> error condition can be serviced/dealt with; **A.** Examples of error conditions that would be likely to generate an interrupt **NE.** To deal with an error, unless stated or clear from example that must be dealt with immediately **NE.** So that a task of higher priority can be carried out	**2**

16	**3**	**2 marks for AO1 (understanding)** So that the currently running process/task/program can be returned to; **NE.** So that the content will not be lost/overwritten **NE.** So that the F-E cycle can continue afterwards As the (code that deals with the) interrupt will change/overwrite/clear register values; **NE.** The contents of the registers will be lost	**2**

17	**1**	**Mark is for AO1 (understanding)** 1024 // 2^{10}; **A.** 1 KiB	**1**

17	**2**	**Mark is for AO1 (knowledge)** The operand is the value/datum that the instruction should use;	**1**

17	**3**	**3 marks for AO3 (programming)**
		
		Values in memory locations 101 and 102 loaded into two different registers;
		
		Contents of the two registers are exclusive ORed;
		A. Memory addresses used as operands directly if no other marks awarded for this question part ie `EOR 103, 101, 102`
		A. Exclusive or achieved in another way eg use of two ANDs, two NOTs and an OR
		
		Value of register storing the result of exclusive or operation is stored into memory location 103;
		A. result of an incorrect combination of the values in locations 101 and 102 stored in location 103
		
		DPT. Use of invalid register name eg Rd
		DPT. Use of incorrect addressing mode
		DPT. Inclusion of invalid symbols in commands
		
		Example Solution
		
		`LDR R1, 101`
		`LDR R2, 102`
		`EOR R3, R1, R2`
		`STR R3, 103`
			3

Qu	Pt	Marking guidance	Total marks
18	1	**All marks AO2 (apply)**	6

R0	R1	R2	R3	R4
	100010 (34)	110 (6)		
0 (0)			1 (1) ***1***	
		1100 (12)	10 (2) ***2***	
		11000 (24)	100 (4)	
		110000 (48)	1000 (8)	
		11000 (24)	100 (4) ***3***	
100 (4)	1010 (10) ***4***			0 (0) ***5***
		1100 (12)	10 (2)	0 (0)
		110 (6)	1 (1)	
101 (5)	100 (4) ***6***			1 (1)
			0 (0)	

1 mark: Correct initial values loaded into `R0` and `R3` – Area 1
1 mark: Logical shifting left of register values in loop – Area 2
1 mark: Exiting loop and shifting right – Area 3
1 mark: First addition and subtraction on `R0` and `R3` – Area 4
1 mark: Addition and subtraction loop – Area 5
1 mark: Correct final values in registers `R0` and `R1` – Area 6

Award marks for the correct values in the indicated areas. The values do not need to be in the exact cells shown for marks to be awarded, but must be in the correct sequence in the column they are in.

Award marks for values written in either decimal or binary. If a binary and decimal value are written in one cell and one is correct but the other incorrect then treat the cell as being correct

Max 5 if any incorrect values in table.

Qu	Pt	Marking guidance	Total marks
18	2	**All marks AO2 (analyse)** Performs (integer) division // outputs the quotient after performing a division // outputs how many times one number (`R2`) goes into another (`R1`) // `R0` is the quotient; Outputs the remainder / what is left over after performing (integer) division // `R1` is the remainder;	2

Qu	Pt	Marking guidance	Total marks
19	1	**All marks AO1 (knowledge)** To marshal / control operation of fetch-execute cycle; Controls fetching / loading / storing operations; **NE.** fetches instructions Determines the type of an instruction; **A.** decodes instructions To execute (some) instructions; To synchronise operation of processor; To send control signals / commands to other components; To control the transfer of data between registers; To handle interrupts; **Max 3**	**3**

Qu	Pt	Marking guidance	Total marks
19	2	**2 marks AO1 (knowledge) and 2 marks AO1 (understanding)** **1 mark (AO1 knowledge): What cache memory is (Max 1):** * Memory that can be accessed very quickly; Memory located on (**A.** close to) the processor; **1 mark (AO1 knowledge): What cache memory is used for:** To store most frequently used // most recently used // pre-fetched instructions/data // to store instructions in the locality of the instruction currently being executed; **2 marks (AO1 understanding): How more cache memory improves performance (Max 2):** More instructions/data can be stored in the cache; #Instructions/data stored in cache can be accessed more quickly than instructions/data in main memory // if an instruction is accessed a second time, it can be retrieved more quickly; This increases the probability that a particular data item/instruction is in the cache when fetched // this increases the probability of a cache hit // fewer fetches from main memory will be required; **Note:** Only award the point marked # if the point marked * has not already been awarded	**4**

Question			Marks
20	**1**	**Mark is AO1 (understanding)** Direct (addressing);	**1**

Question			Marks
20	2	**All marks AO2 (apply)** **Memory Locations** / **Registers** 120 | 121 | 122 | R0 | R1 | R2 | R3 23 | 5 | | 23 | 5 | | 0 | | | | | 1 | 23 | | | 46 | 2 | 0 | | | | 92 | 1 | 1 | 115 | | | 184 | 0 | | | | 115 | | | | **1 mark:** Correct initial values loaded into registers R0, R1 and R3: 23, 5, 0 **1 mark:** R2 has initial value 1 and R3 is updated to 23 **1 mark:** R0 shifted left to give 46 and R1 shifted right to give 2 **1 mark:** R0 changes to 92 then 184, R1 changes to 1 then 0, R2 changes to 0 then 1 **1 mark:** R3 and memory location 122 set to 115 **Max 4** if any incorrect values written into table	**5**

Question			Marks
20	3	**Mark is AO2 (analyse)** (To) multiply (the two numbers in memory locations 120 and 121 together, storing the result in memory location 122); **A.** multiplication	**1**

Question			Marks
20	4	**All marks AO1 (understanding)** So it will execute more quickly; **TO.** if stated that it executes more quickly because translation is not required So it will use less memory (when translated); **NE**. uses less space, more compact A translator for a high-level language might not have been available; Programmer would have complete (**A.** more) control over the final machine code that is output by the translator / executed; **R.** direct access to hardware / registers **Max 2**	**2**

Question			Marks
20	5	**Mark is AO1 (knowledge)** There is a one-to-one mapping // each assembly language instruction translates into one machine code instruction;	**1**

<table>
<tr><th>Qu</th><th>Pt</th><th>Marking guidance</th><th>Total marks</th></tr>
<tr><td>21</td><td>1</td><td>
All marks AO1 (understanding)
<table>
<tr><th>Level</th><th>Description</th><th>Mark Range</th></tr>
<tr><td>4</td><td>A line of reasoning has been followed to produce a coherent, relevant, substantiated and logically-structured response. The response covers both areas indicated in the guidance below and, in each area, there is sufficient detail to show that the student has a good level of understanding.</td><td>10–12</td></tr>
<tr><td>3</td><td>A line of reasoning has been followed to produce a coherent, relevant, substantiated and logically structured response which shows a good level of understanding of at least one area indicated in the guidance below and some understanding of the other area.</td><td>7–9</td></tr>
<tr><td>2</td><td>A limited attempt has been made to follow a line of reasoning and the response has a mostly logical structure. A good level of understanding has been shown of one area or some understanding of both areas.</td><td>4–6</td></tr>
<tr><td>1</td><td>A few relevant points have been made but there is no evidence that a line of reasoning has been followed. There is insufficient evidence of a good level of understanding of either of the two areas.</td><td>1–3</td></tr>
</table>
<u>Guidance – Indicative Content</u>

Area 1: Fetch-Execute Cycle

<u>F-E Stage 1 Fetch:</u>
Contents of Program Counter/PC transferred to Memory Address Register/MAR
R. If implied the instruction is stored in the PC
Address bus used to transfer this address to main memory
Read signal sent along control bus
Transfer of main memory content uses the data bus
Contents of addressed memory location loaded into the Memory Buffer Register/ MBR
Increment (contents of) Program Counter/PC A. At any part of fetch process after transferring PC to MAR
Increment Program Counter/PC and fetch instruction simultaneously
Contents of MBR copied to CIR

<u>F-E Stage 2 Decode:</u>
Instruction to decode held by the (Current) Instruction Register/(C)IR
The control unit decodes the instruction
Instruction split into opcode and operand(s)
</td><td>12</td></tr>
</table>

		F-E Stage 3 Execute: If necessary, data is fetched/stored The opcode identifies the type of operation/instruction to be performed (by the processor) The operation (identified by the opcode) is performed by the control unit. ALU used for calculation/comparisons Result (may be) stored in register/main memory **A.** accumulator Status register updated If jump/branch required Program Counter/PC is updated Control bus will transfer signals to other components to initiate/sequence actions *A good level of understanding would be demonstrated by a response that effectively covered all three stages of the cycle and did not focus excessively on one particular stage. There may be omissions, but these would not be of any key points. Any errors made would be minor.* **Area 2: Improving Hardware** Replace the processor with one which has more cores **A.** Increase number of cores Replace the processor with one which has more cache memory // increase the amount of cache memory // add cache memory Increase clock speed of processor // replace the processor with one which runs at a faster clock speed **NE.** faster processor Use a parallel processor architecture // use more processors which can work in parallel Use a processor with a bigger word size Use a processor that makes (better) use of pipelining Install more RAM // main memory // primary memory Use RAM // main memory // primary memory with a faster access time Replace the motherboard with one which has buses which run at a faster clock speed **A.** increase bus clock speed Replace the motherboard with one which has more lines in data bus **A.** increase number of lines in data bus **A.** Replace HDDs with SSDs // replace HDDS with HDDs that can read data at a faster rate // replace SSDS with SSDs that can read data at a faster rate **A.** Use the Harvard architecture instead of the von Neumann architecture *A good level of understanding would be demonstrated by a response that covered a range of hardware improvements that could be made (eg to the processor, buses, main memory) and did not focus excessively on only one component. Explanations of how a change would improve performance could be taken into account when considering how good the understanding is.*	

Qu	Pt	Marking guidance	Total marks
21	2	**1 mark AO1 (knowledge) and one mark AO1 (understanding)** **1 mark (knowledge):** A signal/request sent to the processor (from a hardware device or program); **Max 1 mark (understanding) from:** So that a device/source that needs the (immediate) attention of the processor can be serviced/dealt with // so that an urgent error condition can be serviced/dealt with; **A.** Examples of error conditions that would be likely to generate an interrupt **NE.** To deal with an error, unless stated or clear from example that must be dealt with immediately **NE.** So that a task of higher priority can be carried out So that the currently executing process/task/program can be suspended; **A.** "stopped" as **BOD** **R.** Suspend/stop the fetch-execute cycle/processor **R.** "instruction" for "process"	**2**

Qu	Pt	Marking guidance	Total marks
22	1	**Mark is AO2 (apply)** Award **1 mark** for correct value in R0: R1 0 1 0 0 0 1 1 0 15 0 0 0 0 1 1 1 1 R0 0 0 0 0 0 1 1 0 **R.** Any cells of R0 left empty	1

Qu	Pt	Marking guidance	Total marks
22	2	**Mark is AO2 (apply)** Award **1 mark** for correct value in R0: R1 0 0 0 0 0 1 1 0 48 0 0 1 1 0 0 0 0 R0 0 0 1 1 0 1 1 0 **R.** Any cells of R0 left empty	1

Qu	Pt	Marking guidance	Total marks
22	3	**4 marks AO2 (analysis) and 6 marks AO3 (programming)** **6 marks AO3 (programming syntax must be correct):** *MP1:* Value in memory location 100 is loaded into a register; *MP2:* After some manipulation has been carried out (whether correct or not) values are stored into memory locations 101 and 102 (do not award if it is the same value stored twice); *MP3:* Binary pattern of one digit correctly isolated from the input value (for leftmost digit must also be shifted so bits in correct place); *MP4:* Binary pattern of one digit correctly translated into ASCII for one of numeric digits **or** letter digits (ignore if the pattern is later changed again to be incorrect); *MP5:* Binary pattern of one digit correctly translated into ASCII for both numeric digits **and** letter digits (ignore if the pattern is later changed again to be incorrect); *MP6:* Conversion process fully working for the both digits (ASCII codes must be correct when program terminates); **Note:** If MP3 not awarded MP4, MP5, MP6 cannot be awarded **A.** Any understandable method for identifying labels **DPT.** Use of invalid register names eg R27, Rn **DPT.** Use of binary for immediate operand values **DPT.** Omission of # to indicate immediate operand values **DPT.** R before memory address eg R100 **DPT.** Use of MOV instead of LDR or STR, or vice-versa **DPT**. Repeated use of incorrect delimiters eg ; < > . " ' (occasional errors can be ignored) **4 marks AO2 (concept must be understood, syntax need not be correct):** *MP7:* Attempt to use masking **and/or** shifting to identify one digit; *MP8:* Attempt to use masking **and/or** shifting a second time to identify the second digit; *MP9:* Attempt to use comparison and branching to make program treat numeric digits and letter digits differently for at least one of the two digits (whether threshold values correct or not); *MP10:* Use of addition or masking to attempt to convert a digit to an ASCII code (whether correct ASCII codes produced or not); **Note:** If MP3 not awarded MP10 cannot be awarded **Max 9 if solution not fully working**	**10**

Example Solution 1

```
        LDR R0, 100             MP1
        AND R2, R0, #15         MP7, MP3
        CMP R2, #10
        BLT isnumber            MP9
        ADD R2, R2, #55         MP10, MP4
        B doleftdigit
isnumber:
        ADD R2, R2, #48         MP5
doleftdigit:
        AND R1, R0, #240        MP8
        LSR R1, R1, #4
        CMP R1, #10
        BLT isnumber2
        ADD R1, R1, #55
        B storetomemory
isnumber2:
        ADD R1, R1, #48         MP6
storetomemory:
        STR R1, 101
        STR R2, 102             MP2
```

Example Solution 2

```
        LDR R0, 100             MP1
        AND R0, R0, #15         MP7, MP3
        CMP R0, #9
        BGT isletter            MP9
        ORR R0, R0, #48         MP10, MP4
        B doleftdigit
isletter:
        SUB R0, R0, #9
        ORR R0, R0, #64         MP5
doleftdigit:
        STR R0, 102
        LDR R0, 100
        LSR R0, R0, #4          MP8
        AND R0, R0, #15
        CMP R0, #9
        BGT isletter2
        ORR R0, R0, #48
        B finish
isletter2:
        SUB R0, R0, #9
        ORR R0, R0, #64         MP6
finish:
        STR R0, 101             MP2
```

Example Solution 3

```
        LDR R1, 100             MP1
        LSR R2, R1, #4          MP7, MP3
        LSL R1, R1, #4
        LSR R2, R1, #4          MP8
        CMP R2, #10
        BLT number              MP9
        ADD R2, R2, #7          MP10
number:
        ADD R2, R2, #48         MP4, MP5
        STR R2, 101
        CMP R2, #10
        BLT number2
        ADD R2, R2, #7
number2:
        ADD R2, R2, #48         MP6
        STR R2, 102             MP2
```

Qu	Pt	Marking guidance	Total marks
23	1	**All marks AO2 (apply)**	6

Memory Location 130	R1	R2	R3	R4
83 (01010011)	*1* 83 (01010011)	0		0
		1	1	1 *2*
	3 41 (00101001)	2	1	2
	20 (00010100)	3	0	
	10 (00001010)	4	0	
	5 (00000101)	5	1	3
	2 (00000010)	6	0	
	1 (00000001)	7	1	4
	0 (00000000)	*4*	*5*	
6	83 (01010011)			0
	211 (11010011)			
211 (11010011)				

1 mark: Correct initial values loaded into R1, R2 and R4 – Area 1.
1 mark: First increment of R2 and R4 and first logical AND of R3 – Area 2.
1 mark: Contents of R1 shifted right 7 times – Area 3.
1 mark: R2 counts up from 2 to 7 – Area 4.
1 mark: R3 shows correct values of ANDing R1 and 1 and R4 increments from 2 to 4 – Area 5.
1 mark: R4 set to 0, MSB of R1 set to 1 and contents of R1 copied to memory location 130 – Area 6.

Award marks for the correct values in the indicated areas. The values do not need to be in the exact cells shown for marks to be awarded, but must be in the correct sequence in the column they are in.

Award marks for values written in either decimal or binary. If binary and decimal values are written in one cell and one is correct but the other incorrect then treat the cell as being correct.

Max 5 if any incorrect values in table (ignore cells where one of the binary and decimal is incorrect but the other correct).

Qu	Pt	Marking guidance	Total marks
23	2	**All marks AO2 (analyse)** **If the student recognises that the program is connected to parity:** Sets (**A.** calculates) the parity bit (for the ASCII character); using odd parity; **If the student does not recognise that the program is connected to parity:** Counts the number of 1s (in the ASCII code); **NE.** literal descriptions of how the code is working or responses that don't interpret what the purpose of the program is	**2**