AQA Computer Science A-Level

4.7.3 Structure and role of the processor and its components

Past Paper Mark Schemes

Additional Specimen AS Paper 2

08			**Marks are for AO2 (apply)** **Mark Scheme** **System hardware decisions:** clock speed word length address bus width data bus width **Effects on performance:** multiple cores - two or more independent processing units means that tasks can (potentially) be run in parallel - tasks split up to make use of extra cores so speed of execution is increased cache memory - small amount of very fast memory placed near to the processor - attempts to contain the next instruction(s)/data which speeds up execution of programs clock speed - by increasing the clock speed the processor will execute more instructions per unit time word length - by increasing the word length the processor can handle more data per instruction	6

Level	Description	Mark Range
3	A detailed description where the student identifies two more hardware decisions and describes all four types with examples that shows a good level of understanding.	5-6
2	An adequate description that covers at least three types and demonstrates a reasonable level of understanding.	3-4
1	A small number of points recalled but little or no understanding shown.	1-2

			- by increasing the word length the processor can access a larger amount of memory through direct addressing - by increasing the word length the processor could have a larger instruction set bus width - by increasing the address bus size we can address more unique memory locations - increasing the address bus size increases the maximum potential memory size - by increasing the data bus size we can move more data around per unit time	

Additional Specimen Paper 2

04	**3**	**All marks AO1 (recall)** **Step 1:** MAR ← [PC] / Contents of program counter transferred to MAR; **Step 2b:** MBR ← $[Memory]_{addressed}$ / Contents of addressed	3
		memory location loaded into MBR / MBR ← $[Memory]_{MAR}$; (must have concept of data coming from address in memory, not just going into MBR) **Step 4:** Decode instruction; **A.** Contents of CIR decoded **R.** Data for instruction **R.** CIR decoded, CIR decodes instruction **1 mark for each correct step** For PC accept Program Counter/SCR/Sequence Control Register For MAR accept Memory Address Register For MBR accept Memory Buffer Register/MDR/Memory Data Register **A.** Other means of indicating correct transfer e.g. [PC]→MAR or MAR:=PC **A.** Missing square brackets or alternative types of brackets **A.** Answers that miss out reference to "contents of" **A.** [Memory] for $[Memory]_{addressed}$ / $[Memory]_{MAR}$	
04	**4**	**All marks AO1 (recall)** Volatile environment / current processor state saved on stack; Source of interrupt identified; Appropriate interrupt service routine/ISR called; Volatile environment / processor state restored;	4

January 2009 Comp 2

3	(a)		Load **B**; **Add** #5; **A** *absolute addresses instead of A and B* Store **A**;	**3**

5	1. address of next instruction to be executed/fetched; 2. (contents of Program Counter) copied into Memory Address Register; 3. Contents of Program Counter incremented (by 1); *Accept incrementing by more than 1* 4. …at the same time…; *(only give a mark if between correct statements)* 5. instruction/data held at that address is placed in the Memory Buffer Register; 6. Contents of Memory Buffer Register copied into Current Instruction Register; 7. Instruction held in Current Instruction Register is decoded; 8. If necessary data is fetched; 9. (and) instruction is executed by processor/ALU; 10. Address sent/transferred over address bus; 11. Data/instruction transferred to processor on data bus; 12. Result stored in accumulator;	**MAX 6**

January 2010 Comp 2

3	(a)		Program Counter; **A** Sequence Control Register **R** Next Instruction Register Current Instruction Register; **A** Instruction Register Memory Buffer Register; **A** Memory Data Register Memory Address Register; **MAX 2**	2

3	(b)		Address in MAR/address to fetch instruction from, sent down Address Bus to Main Memory; **R** address in PC (program counter) Contents of address accessed in Main Memory; **A** by implication if contents of address location referred to during data transfer Contents of address location//instruction//data passed down Data Bus into MBR/to processor; **A** MDR instead of MBR **A** RAM for Main Memory **MAX 2**	2

3	(c)		Order of execution unimportant/one step does not rely on prior completion of the other; Steps carried out by different (hardware) devices/components; **A** operations are independent **A** operations use different registers **R** using different buses **MAX 1**	1

5	(a)	(i)	LOAD = Opcode 4 = Operand **1 mark** for both parts correct	1

5	(a)	(ii)	A storage/memory location in the <u>processor</u>; **A** CPU **NE** location in the processor	1

5	(b)		`LOAD 12;`
			`ADD 13;`
			`STORE 14;`
			A operands 12 and 13 swapped around BUT NOT swapped opcodes
			A correct binary operands 12- 1100 13- 1101 14- 1110
			A minor spelling errors in Opcode only
			P1 for use of # or other symbols with operand
			Penalise each additional unnecessary instruction (beyond 3)by 1 mark
				3

January 2011 Comp 2

4	a		Operand - 5 Opcode - LOAD ; Both needed for the mark **A** binary value 101 with any number of preceding zeroes for the operand	1

4	b		`LOAD 7;`
			`ADD 8` } Both Add instructions for the mark - do not need to follow each other.
			`ADD 3`
			`STORE 21;`
			The operands for LOAD and ADD can be in any order
			I an end of line indicator symbol e.g. ";"
			I comments explaining code
			I additional unnecessary commands
			R commands with a # or ( ) or [ ] in the operand
			A operands in binary
			A operands in binary and opcodes in binary, if candidate has provided a translation table
			A correct operands in hex if using &
			MAX 2 if code would not produce correct result
				3

7	a		Very hard/difficult to understand; Very easy to make mistakes; Hard to find any errors/mistakes in the code; Time consuming to develop software in assembly language; Lack of portability; Lack of in-built functions/procedures; **NE** harder to learn	MAX 2

January 2012 Comp 2

3	**a**		<table><tr><th>Number</th><th>Component</th></tr><tr><td>1</td><td>Memory address register; **NE – MAR;**</td></tr><tr><td>2</td><td>Data bus;</td></tr><tr><td>3</td><td>Control bus;</td></tr></table>	**3**

3	**b**		To fetch / decode / execute instructions; To synchronise operation of processor; To marshal/control operation of fetch-execute cycle; To send control signals/commands to other components of fetch-execute cycle; To control the transfer of data between registers/MBR; **A** – by example **NE** - information	**MAX 1**

3	c		Arithmetic (and) logic unit; **NE** – Arithmetic unit **NE** – Logic unit	1

3	d		A (very fast) memory location within the processor; **A** - A (very fast) memory location within an I/O controller;	1

3	e		Arithmetic results – Overflow/underflow/positive/negative/zero/carry; Interrupts (enabled/disabled); Parity; BCD arithmetic enabled/disabled; Supervisor mode; Halt; **A** illegal instruction/operation	**MAX 1**	Refer to team leader with other potentially correct answers.

June 2010 Comp 2

7	(a)		<table><tr><th>Number</th><th>Component Name</th></tr><tr><td>1</td><td>Memory Address Register</td></tr><tr><td>2</td><td>Address Bus</td></tr><tr><td>3</td><td>Memory Data/Buffer Register</td></tr><tr><td>4</td><td>Data Bus</td></tr></table>	4

7	(b)		The instruction is held in the CIR; **A** IR The control unit/instruction decoder decodes the instruction; The opcode identifies the type of instruction it is; Relevant part of CPU/processor executes instruction; **A** ALU Further memory fetches/saves carried out if required; Result of computation stored in accumulator/register/written to main memory; Status register updated; If jump/branch instruction, PC is updated; **A** SCR **MAX 3**	3

June 2011 Comp 2

4	a		Second (generation); **A** 2 **R** assembly code / language Note: Adding "assembly" / "assembler" does not talk out a valid mark for second / 2	**1**
4	b		(memory) Address / location / offset; **A** line number **R** instruction number	**1**
4	c		(y) Opcode / operation code; **A** op-code **NE** operation (z) Operand;	**2**
4	d		**Individual Instructions:** One to one / each assembly language instruction translates to one machine code instruction; **Programs:** Figure 2 assembly language equivalent of figure 3 // figure 3 machine code version of figure 2 // figure 3 is assembled version of figure 2; **NE** figure 3 "binary version" of figure 2 **NE** different generations of language	**1**

7	b		Memory address register; **R** abbreviations	**1**

7	c		Memory buffer register / memory data register; **R** abbreviations	**1**

June 2012 Comp 2

2	a	i	Indicates the basic machine operation/function/command; Executable binary code; **A** "instruction" – with a valid example	MAX 1
2	a	ii	Represents a single item of (binary) data / a single value; Represents a memory address / storage location; The value that the instruction operates on; **A** parameter for the operation **NE** "address"	MAX 1

5			Key points of subject criteria: *FETCH:* Contents of Program Counter/PC transferred to Memory Address Register/MAR; Address bus used to transfer this address to main memory; Contents of addressed memory location loaded into the Memory Buffer Register/MBR; Transfer of content uses the data bus; Increment contents of Program Counter/PC; Increment Program Counter/PC and fetch simultaneously; **A** any part of fetch process Transfer content of Memory Buffer Register/MBR to the Current Instruction Register/CIR; *DECODE:* Decode instruction held by the Current Instruction Register/CIR; The control unit decodes the instruction; Instruction split into opcode and operand; *EXECUTE:* If necessary, data is fetched; The opcode identifies the type of instruction it is; Execute instruction by relevant part of processor; Result stored in accumulator; Status register updated; If jump/branch instruction Program Counter/PC is updated;	6

June 2013 Comp 2

2	**(a)**		Program Counter/Sequence Control Register; Memory Address Register; Memory Buffer Register/Memory Data Register; Current Instruction Register; **R.** Abbreviations	**MAX 2**	

2	**(b)**		**Step 1:** MAR ← [PC] / Contents of program counter transferred to MAR; **R.** MAR ← PC	**3**
			R. [MAR] ← PC (see note about **DPT)** **R.** PC sends/transfers **Step 2b:** MBR ← $[Memory]_{addressed}$ / Contents of addressed memory location loaded into MBR; (must have concept of data coming from address in memory, not just going into MBR) **Step 4:** Decode instruction; **A.** Contents of CIR decoded **A.** Instruction is split into opcode and operand **R.** Data for instruction **R.** CIR decoded, CIR decodes instruction **Note: A.** [CIR] decoded 1 mark for each correct step For PC accept Program Counter/SCR/Sequence Control Register For MAR accept Memory Address Register For MBR accept Memory Buffer Register/MDR/Memory Data Register **A.** Other means of indicating transfer e.g. [PC] → MAR **A.** [Memory] for $[Memory]_{addressed}$ **DPT** – no/incorrect square bracket use for register transfer notation	

4	(d)		LOAD 21 STORE 23 LOAD 22 STORE 21 LOAD 23 STORE 22 1 mark for value from 21 stored into 23; 1 mark for value from 22 being moved to 21; 1 mark for value from 23 being moved to 22; ***Alternative :*** LOAD 22 STORE 23 LOAD 21 STORE 22 LOAD 23 STORE 21 1 mark for value from 22 stored into 23; 1 mark for value from 21 being moved to 22; 1 mark for value from 23 being moved to 21; **DPT** if a different temporary storage area is used **I** end of statement separators **MAX 2 if the program does not fully work**	**3**

June 2016 AS Paper 2

04	2	**Mark is for AO2 (analyse)** `#1;` **R.** 1 **I.** zeroes between # and 1 **Refer answers that start with #1 and then have any other writing to senior examiner**	1

04	3	**2 marks for AO3 (design) and 4 marks for AO3 (programming)** **AO3 (design) – 2 marks:** **1 mark:** Identifying that a comparison and branch are required to have the same effect as the `IF` statement, even if the syntax or comparison made are incorrect **1 mark:** Identifying that one or more labels are needed for branching to work **AO3 (programming) – 4 marks:** For the AO3 (programming) marks, the syntax used must be correct for the language as described on the question paper. **1 mark:** Comparing `R3` against 1 or 0 and having a branch with the correct logical condition **1 mark:** For moving `69` to `R2` in the equivalent of the THEN part **A.** moving `69` to `R2` in equivalent of ELSE part if this is appropriate for compare and branch statements used **1 mark:** For having an unconditional branch that results in skipping over 2nd move instruction or HALT in appropriate place **1 mark:** For moving `79` to `R2` in the equivalent of the ELSE part **A.** moving `79` to `R2` in equivalent of THEN part if this is appropriate for compare and branch statements used **Max 3** marks for programming if any syntax incorrect or program does not work correctly under all circumstances **I.** Missing `AND` instruction at start of answer. **I.** Incorrect `AND` instruction at start of answer.	6

I. Load instruction to setup `R1` from `A`.
I. Store instruction to store `R2` into `B`.
A. Labels given in any sensible format
A. Answers that use hexadecimal or binary values
A. Line numbers as equivalent to labels if they are used as the target of branches. **Note: in future this will not be accepted as line numbers are not part of the AQA assembly language.**
DPT Missing hash for immediate addressing
DPT incorrect use of commas, colons, semi-colons, etc...

Refer alternative answers to team leaders

```
  AND R3, R1, #1
  CMP R3, #1
  BEQ odd
  MOV R2, #69
  B end
odd:
  MOV R2, #79
end:
```

//

```
  AND R3, R1, #1
  CMP R3, #1
  BEQ odd
  MOV R2, #69
  HALT
odd:
  MOV R2, #79
```

//

```
  AND R3, R1, #1
  CMP R3, #1
  BNE even     A. BLT instead of BNE
  MOV R2, #79
  B end
even:
  MOV R2, #69
end:
```

//

```
  AND R3, R1, #1
  CMP R3, #0
  BNE odd      A. BGT instead of BNE
  MOV R2, #69
  B end
odd:
  MOV R2, #79
end:
```

//

		AND R3, R1, #1
		CMP R3, #0
		BEQ even
		MOV R2, #79
		B end
		even:
		MOV R2, #69
		end:	

04	4	**Mark is for AO1 (understanding)** Immediate; **R.** More than one lozenge shaded	1

04	5	**Mark is for AO1 (knowledge)** A memory/storage location inside the processor; **A.** CPU instead of processor **NE** memory/storage location	1

05	6	**3 marks for AO1 (knowledge) and 3 marks for AO1 (understanding)** **1 mark for AO1 (knowledge):** (increase the) data bus width; **1 mark for AO1 (understanding):** enables more bits (**A.** data) to be transferred between main memory and the processor <u>at one time</u> (so fewer read/write operations needed); **1 mark for AO1 (knowledge):** (increase the) clock speed; **1 mark for AO1 (understanding):** enables more instructions to be executed per unit of time/second (**A.** calculations/operations/commands instead of instructions) // each individual instruction could be executed sooner / more quickly (**A.** calculation/operation/command instead of instruction); **1 mark for AO1 (knowledge):** (increase the) amount of cache memory; **1 mark for AO1 (understanding):** cache memory is faster than main memory so the more that can be stored in cache memory the less frequently the main memory needs to be accessed;	6

		1 mark for AO1 (knowledge): (increase the) word length; **1 mark for AO1 (understanding):** larger word size means that the processor can process more bits in one go; **1 mark for AO1 (knowledge):** (change the) type of cache memory; **1 mark for AO1 (understanding):** some types of cache memory can be accessed faster; **A.** using memory with a faster access speed **1 mark for AO1 (knowledge):** (increase the) number of general purpose registers; **1 mark for AO1 (understanding):** more intermediate results/variables can be kept in processor registers rather than in main memory; **1 mark for AO1 (knowledge):** (increase the) address bus width; **1 mark for AO1 (understanding):** enables the processor to access a larger number of main memory locations (meaning it will not need to make as much use of virtual memory this will mean that system performance is improved); **A.** allows more main memory to be installed **R.** How improves mark if it is not relevant for the factor stated. **NE** How improves of "program will execute faster" **Note:** marks for the factor can be awarded in either the "factor" or "how improves" part of an answer	

June 2017 AS Paper 2

06	3	**2 marks for AO1 (knowledge) and 4 marks for AO1 (understanding)** **Level of response question**	**6**

Level	Description	Mark Range
3	At least five of the steps of the cycle have been correctly identified in order/the steps are all in correct order and covering all three of the stages (fetch, decode, execute). For the top mark in this level thorough understanding of how the cycle works is evident.	5-6
2	At least three steps of the cycle have been identified in order, covering at least two of the stages (fetch, decode, execute). Some understanding of how the cycle works is evident.	3-4
1	At least one step of the cycle have been identified, covering at least one stage (fetch, decode or execute). The order of the steps may not be correct. Little understanding of how the cycle works is evident.	1-2

Points may include:

Fetch:
Contents of Program Counter / PC transferred to Memory Address Register / MAR
Address bus used to transfer this address to main memory
Transfer of 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 simultaneously
Contents of MBR copied to CIR

Decode:
Decode instruction held by the (Current) Instruction Register / (C)IR
The control unit decodes the instruction
Instruction split into opcode and operand

Execute:
If necessary, data is fetched
If necessary, data is stored in memory
The opcode identifies the type of operation/instruction to be performed (by the processor)
Result (may be) stored in register/accumulator
The operation (identified by the opcode) is performed by the processor. **A.** ALU
Status register updated
If jump / branch required Program Counter/PC is updated

NE. Register notation
A. Memory Data Register/MDR for Memory Buffer Register/MBR
I. Incorrect headings

07	1	**Marks are for AO3 (program)** **Answer 1** 1. ADD R0, R0, #1 ; 2. CMP R0, #11 ; 3. BNE; startloop ; **Answer 2** 1. ADD R0, R0, #1 ; 2. CMP R0, #11 ; 3. BEQ endloop ; 4. B startloop ; **Answer 3** 1. CMP R0, #10 ; 2. BEQ endloop ; 3. ADD R0, R0, #1 ; 4. B startloop ; **Answer 4** 1. ADD R0, R0, #1 ; 2. CMP R0, #11 ; 3. BLT; startloop ; **Stop marking when the first incorrect command is encountered. Mark response against whichever alternative gives the highest mark.** **I.** Any extra commands which do not effect operation of program.	**4**
07	2	**Mark is for AO2 (apply)** 28_{10} // $(000)11100_2$; **TO.** If two answers given and one is incorrect. **I.** Lack of subscript.	**1**

07	3	**Mark is for AO1 (understanding)** Direct addressing means that the operand is the (memory) address/register number (of the datum) whereas immediate addressing means the operand is the datum ; **Note:** Must be clear that the operand is being used.	**1**

June 2017 Paper 2

01	1	**All marks AO1 (understanding)**	4

Level	Description	Mark Range
4	Description covers all, or almost all, of the points in the indicative guidance and fully reflects the sequence in which steps occur. It includes use of registers, buses and main memory. An excellent level of understanding is shown with no misconceptions.	4
3	Description covers most (ie more than half) of the points in the indicative guidance and completely or almost completely reflects the correct sequence in which steps occur. At least two of the use of registers, buses and main memory are covered. A good level of understanding is shown. Whilst there may be some omissions, there is at most one misconception in the response.	3
2	At least two correct points are made from the indicative guidance and there is some indication of understanding of the correct sequence. Some understanding is shown.	2
1	At least one relevant point has been made. There is not sufficient evidence to conclude that the cycle has been understood.	1

Guidance – Indicative Response

- Contents of Program Counter/PC transferred to Memory Address Register/MAR
- Address bus used to transfer this address to main memory
- Fetched value/instruction transferred using the data bus
- Contents of addressed memory location loaded into the Memory Buffer Register/MBR
- Transfer content of Memory Buffer Register/MBR to the Current Instruction Register/CIR

A. Memory Data Register / MDR for MBR
I. Incrementing of program counter, even if incorrect
NE. Points made using register transfer notation only eg CIR ← [MBR]

01	2	**All marks AO1 (understanding)** To execute/carry out the instruction other data may need to be fetched (from main memory); **A.** During execute phase MBR used to store other data **A.** Further instructions may need to be fetched before the instruction has finished executing, if pipelining/parallelisation is referenced explicitly in the response Further memory fetches would overwrite the contents of the MBR // the instruction would be overwritten by further memory fetches // writing the result of executing the instruction back to main memory would overwrite the instruction / MBR contents; **A.** MBR is not (directly) wired to the (processor) components that will execute the instruction which CIR is **A.** The MBR is not (directly) wired to the ALU as BOD	**2**
		R. The MBR cannot decode instructions	

Figure 5 shows an assembly language program together with the contents of a section of the main memory of the computer that the program will be executed on. Each main memory location and register can store a 16-bit value.

The assembly language instruction set that has been used to write the program is listed in **Table 1** on the next page.

Figure 5

Program
LDR R1, 100
LSL R2, R1, #2
ADD R1, R1, R2
LDR R3, 101
CMP R3, R1
BEQ labela
MOV R4, #0
B labelb
labela:
MOV R4, #1
labelb:
STR R4, 102
HALT

Memory Address (in decimal)	Main Memory Contents (in decimal)
100	10
101	50
102	80

<table>
<tr><td>05</td><td>1</td><td>1 mark for AO1 (knowledge) and 1 mark for AO1 (understanding)

AO1 (knowledge): 1 mark:

An operand is a value/data that will be used by an operation;

AO1 (understanding): 1 mark:

The addressing mode indicates how the value in the operand should be interpreted // the addressing mode indicates if the value in the operand is a memory address/register or a data/immediate value;
A. In immediate addressing the operand is the value to use and in direct addressing it is a memory address/register number
NE. Addressing mode indicates if direct or immediate addressing is used</td><td>2</td></tr>
<tr><td>05</td><td>2</td><td>All marks AO2 (apply)

<table>
<tr><th colspan="4">Register Contents</th><th colspan="3">Main Memory Location Contents</th></tr>
<tr><th>R1</th><th>R2</th><th>R3</th><th>R4</th><th>100</th><th>101</th><th>102</th></tr>
<tr><td>10</td><td>40</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>50</td><td></td><td>50</td><td>1</td><td></td><td></td><td>1</td></tr>
</table>
</td><td>4</td></tr>
</table>

1 mark: Value of 10 is first value in R1.

1 mark: Value of 40 is only value in R2.
A. Value in R2 is four times the value in R1, if value in R1 was incorrect

1 mark: Value of 50 in both R1 and R3, as the second and final value in R1 and only value in R3.
A. Value stored in R1 is five times the initial value in R1, if this was incorrect and R3 contains only 50
A. Value stored in R1 is equal to contents of R2 and previous contents of R1 added together, if either of these were incorrect and R3 contains only 50

1 mark: Value of 1 stored in both R4 and memory location 102. It should be the only value in R4 but could be preceded by 80 in memory location 102. It must be the final value in memory location 102. This mark should only be awarded if the contents of R1 and R3 are equal, otherwise see accept point below.
A. Value of 0 instead of 1 stored in both R4 and memory location 102 if contents of registers R1 and R3 are not equal
I. Values of 10 and 50 written in the columns for main memory locations 100 and 101 and value of 80 written above value of 1 in column for memory location 102

Note: Values do not have to be written in the same rows as in the table above, but must be in the same order ie for R1, the value 10 must be assigned above the value 50. Individual values eg 50 may be written out multiple times.

05	**3**	**Mark is for AO2 (analyse)** Check if the value stored in memory location 101 is five times the value stored in memory location 100 // check if value in memory location 100 is a fifth of that in memory location 101 (if so, store a 1 in memory location 102 if it is and a 0 if it is not); **A.** Check if a number is five times another number // a fifth of another number as BOD	**1**

June 2009 Comp 2

7	(a)		**Step 1:** MAR ← [PC] / Contents of program counter transferred to MAR; **Step 2b:** MBR ← $[Memory]_{addressed}$ / Contents of addressed memory location loaded into MBR; (must have concept of data coming from address in memory, not just going into MBR) **Step 4:** Decode instruction; **A** Contents of CIR decoded **R** Data for instruction **R** CIR decoded, CIR decodes instruction **1 mark for each correct step** For PC accept Program Counter/SCR/Sequence Control Register For MAR accept Memory Address Register For MBR accept Memory Buffer Register/MDR/Memory Data Register For CIR accept Current Instruction Register/IR/Instruction Register **A** Other means of indicating correct transfer e.g. [PC]→MAR or MAR:=PC **A** Missing square brackets or alternative types of brackets **A** Answers that miss out reference to "contents of" **A** [Memory] for $[Memory]_{addressed}$	**3**

	(b)	(i)	Increases the number of bits (**A** amount of data) that can be transferred <u>at one time</u> // increase rate of data transfer;	**1**
		(ii)	Increases the number of memory addresses / /Increase the <u>maximum</u> amount of primary store/memory (possible);	**1**
		(iii)	<u>Instructions</u> performed more quickly // <u>Instructions</u> executed at faster rate; **A** Calculations for instructions (this time only) **A** Operations for instructions **NE** Speeds the computer up **R** Processes, tasks for instructions	**1**

Specimen AS Paper 2

06	**2**		**2 marks for AO1 (knowledge) and 4 marks for AO1 (understanding)** **Level** / **Description** / **Mark Range** 3 / A detailed description, indicating a comprehensive knowledge has been provided which covers all three stages. For each stage, the description covers the majority of the points listed in the guidance. The answer is well structured and points are connected in a way that demonstrates a good understanding of the complete cycle. / 5-6 2 / An adequate description indicating knowledge of the cycle has been provided that either covers one or two stages in a good level of detail, including the majority of points for each stage, or covers all three stages but at a more superficial level. The answer is satisfactorily structured and points are connected in a way that demonstrates an understanding of some parts of the cycle. / 3-4 1 / A small number of points, from one or more stages have been recalled indicating some knowledge of the cycle. However, these have not been connected and demonstrates little or no understanding of any stage of the cycle. / 1-2	6
			FETCH: o contents of PC transferred to MAR o address bus used to transfer this address to main	

			memory o contents of addressed memory location moved into the MBR o transfer of content used the data bus o increment PC o transfer content of MBR to CIR. DECODE: o decode instruction held by the CIR o the control unit decodes the instruction o instruction split into opcode and operand. EXECUTE: o if necessary, data is fetched o the opcode identifies the instruction to execute / operation to perform o execute instruction by relevant part of processor o result stored in accumulator	

06	**4**		**Marks is for AO1 (knowledge)** **1 mark:** (opcode) represents the instruction to be executed;	1

06	**5**		**1 mark for AO1 (knowledge) and 1 mark for AO1 (understanding)** **AO1 (knowledge):** **1 mark:** Immediate addressing: the operand value is part of the instruction // no need to go to any memory address; **AO1 (understanding):** **1 mark:** Example: MOV RX, #Y; [where X is 0-12 and Y is a decimal value]	2

06	6		**1 mark for AO3 (design) and 3 marks for AO3 (programming)**
			
			`CMP R1, #5 compare r1 against 5`
			`BNE endif jump to end of statement if not equal`
			`MOV R2, #10 move the value 10 to B`
			`endif:`
			
			AO3 (design) – 1 mark:
			
			1 mark: Identifying that a comparison and branch are required to have the same effect as the IF statement, even if the syntax or comparison made are incorrect
			
			AO3 (programming) – 3 marks:
			
			For the AO3 (programming) marks, the syntax used must be correct for the language as described on the question paper.
			
			1 mark: Comparing R1 against 5 and having a branch with the correct logical condition
			1 mark: For moving 10 to R2
			1 mark: For having a label for end of statement (that is used in the branch)
			
			I. Load instruction to setup R1 from X.
			I. Store instruction to store R2 into B.
			A. labels given in any sensible format
			DPT - missing hash for immediate addressing
				4

Specimen Paper 2

06	**1**	**Mark is for AO1 (understanding)** $64 / 2^6$;	1

06	**2**	**Mark is for AO2 (apply)** 100;	1

06	**3**	**Mark is for AO2 (apply)** 110; **A.** The response given to question part 6.2 with 10 added on.	1

06	**4**	**Mark is for AO2 (apply)** 220; **A.** The response given to question part 6.3 multiplied by 2.	1