QUESTION 1.

8 (a) **maximum** of **2 marks** for data bus width and **maximum** of **2 marks** for clock speed

data bus width

- the width of the data bus determines the number of bits that can be simultaneously transferred
- increasing the width of the data bus increases the number of bits/amount of data that can be moved at one time (or equivalent)
- …hence improving processing speed as fewer transfers are needed
- By example: e.g. double the width of the data bus moves 2x data per clock pulse

clock speed

- determines the number of cycles the CPU can execute per second
- increasing clock speed increases the number of operations/number of fetch-execute cycles that can be carried out per unit of time
- ...however, there is a limit on clock speed because the heat generated by higher clock speeds cannot be removed fast enough **[3]**

(b) Any **two** from:

- devices automatically detected and configured when first attached/plug and play
- it is nearly impossible to wrongly connect a device
- USB has become an industrial standard
- supported by many operating systems
- USB 3.0 allows full duplex data transfer
- later versions are backwards compatible with earlier USB systems
- allows power to be drawn to charge portable devices **[2]**

(c)

Description of stage	Sequence number
the instruction is copied from the Memory Data Register (MDR) and placed in the Current Instruction Register (CIR)	**3**
the instruction is executed	**6**
the instruction is decoded	**5**
the address contained in the Program Counter (PC) is copied to the Memory Address Register (MAR)	**1**
the value in the Program Counter (PC) is incremented so that it points to the next instruction to be fetched	**4**
the instruction is copied from the memory location contained in the Memory Address Register (MAR) and is placed in the Memory Data Register (MDR)	**2**

[6]

QUESTION 2.

5 (a) **one mark** for name of bus + **one mark** for description

address bus

- lines used to transfer address of memory or input/output location
- unidirectional bus

data bus

- used to transfer data between the processor and memory/input and output devices
- bidirectional bus

control bus

- used to transmit control signals
- e.g. read/write/fetch/ …
- dedicated bus since all timing signals are generated according to control signal [6]

(b) (i) the program counter is incremented [1]

(ii) the data stored at the address held in MAR is copied into the MDR [1]

(iii) the contents of the Memory Data Register is copied into the Current Instruction Register [1]

(c)
- the MAR is loaded with the operand of the instruction // loaded with 35
- the Accumulator is loaded with the contents of the address held in MAR // the Accumulator is loaded with the contents of the address 35 [2]

(d) (i)
- a signal
- from a device/program that it requires attention from the processor [2]

(ii)
- at a point during the fetch-execute cycle …
- check for interrupt
- if an interrupt flag is set/ bit set in interrupt register
- all contents of registers are saved
- PC loaded with address of interrupt service routine [4]

QUESTION 3.

3 (a) Four points from:

- The Program Counter (PC) holds the address of the next instruction to be fetche
- The address in the Program Counter (PC) is copied to the Memory Address Regist (MAR)
- The Program Counter (PC) is incremented
- The instruction is copied to the Memory Data Register (MDR)
  - …. from the address held in the Memory Address Register (MAR)
- The instruction from the Memory Data Register (MDR) is copied to the Current Instruction Register (CIR)

(b) One mark for each statement or letter in the correct place. **[4]**

At the end of the cycle for the current instruction **B**
If the interrupt flag is set, **D**, **A** and **C**
The interrupted program continues its execution

At the end of the cycle for the current instruction the processor checks if there is an interrupt.
If the interrupt flag is set, the register contents are saved, the address of the Interrupt Service Routine (ISR) is loaded to the Program Counter (PC) and when the ISR completes, the processor restores the register contents.

The interrupted program continues its execution.

4 (a) Three from: **[3]**

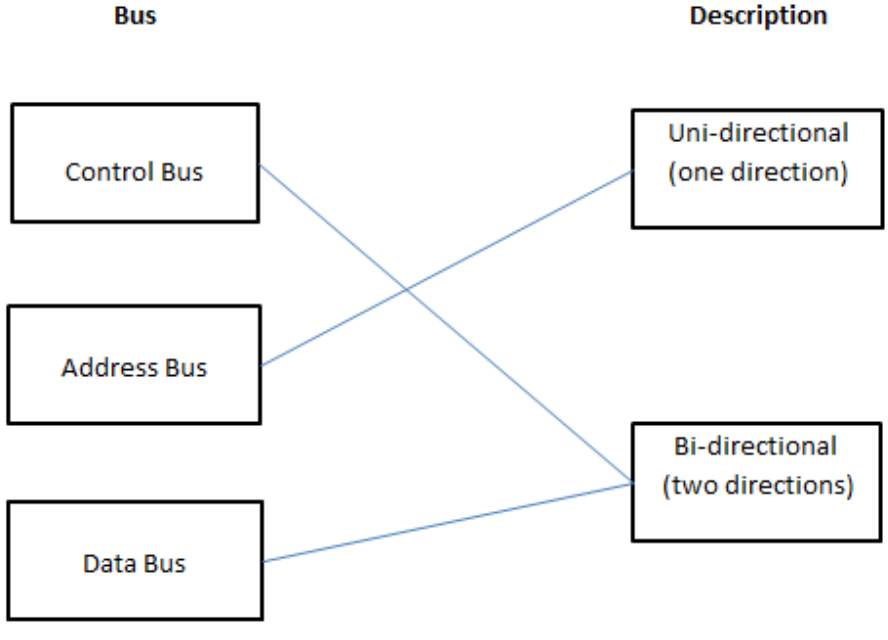
QUESTION 4.

<table>
<tr><th>Question</th><th>Answer</th><th></th></tr>
<tr><td>4(a)</td><td>1 mark per correct line, max 3
<table>
<tr><th>Line number of error</th><th>Correct notation</th></tr>
<tr><td>1</td><td>MAR ← [PC]</td></tr>
<tr><td>3</td><td>MDR ← [[MAR]]</td></tr>
<tr><td>4</td><td>CIR ← [MDR]</td></tr>
<tr><td>2</td><td>PC ← [PC] + 1</td></tr>
</table></td><td></td></tr>
<tr><td>4(b)(i)</td><td>1 mark for each event to max 3

For example:
• Hardware fault // Example of hardware fault
• I/O request // Example of I/O request
• Program/software error // Example of software error
• End of a time-slice</td><td>3</td></tr>
<tr><td>4(b)(ii)</td><td>1 mark per bullet point to max 5

• At the <u>end</u> of each fetch–execute cycle the processor checks for interrupt(s)
• Check if an interrupt flag is set // Check if bit set in interrupt register
• Processor identifies source of interrupt
• Processor checks priority of interrupt
• If interrupt priority is high enough // Lower priority interrupts are disabled
• Processor saves current contents of registers
• Processor calls interrupt handler / Interrupt Service Routine (ISR)
• Address of ISR is loaded into Program Counter (PC)
• When servicing of interrupt complete, processor restores registers
• Lower priority interrupts are re-enabled
• Processor continues with next F–E cycle</td><td>5</td></tr>
</table>

Question	Answer
4(c)	**1 mark** for **1 correct** connection **2 marks** for all **3 correct** connections



QUESTION 5.

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>6(a)(i)</td><td>MAR ← [PC]
PC ← PC + 1
MDR ← [[MAR]]
CIR ← [MDR]</td><td>3</td></tr>
<tr><td>6(a)(ii)</td><td>1 mark from:
• The contents of the MAR is an address, it is the contents of that address which is transferred to MDR
• The contents of the address pointed to by the MAR is transferred to the MDR</td><td>1</td></tr>
<tr><td>6(b)</td><td>1 mark per bullet point to max 2

• A signal from a source / device
• Telling the processor its attention is needed</td><td>2</td></tr>
<tr><td>6(c)</td><td>1 mark per row

<table>
<tr><th>Statement</th><th>DRAM</th><th>SRAM</th></tr>
<tr><td>Does not need to be refreshed as the circuit holds the data while the power supply is on</td><td></td><td>✓</td></tr>
<tr><td>Mainly used in cache memory of processors where speed is important</td><td></td><td>✓</td></tr>
<tr><td>Has less complex circuitry</td><td>✓</td><td></td></tr>
<tr><td>Requires higher power consumption under low levels of access, which is significant when used in battery-powered devices</td><td>✓</td><td></td></tr>
<tr><td>Requires data to be refreshed occasionally so it retains the data</td><td>✓</td><td></td></tr>
</table>
</td><td>5</td></tr>
</table>

QUESTION 6.

Question	Answer	Marks
3(a)	**1 mark** for each error **and** correction • Line 02 should be +1 not −1 // `PC ← [PC] + 1` • Line 03 should be double brackets around `MAR` // `MDR ← [[MAR]]` • Line 04 should be `MDR` not `MAR` // `CIR ← [MDR]`	**3**
3(b)	**1 mark** for each group to **max. 2** • Data movement • Arithmetic operations • (Unconditional and conditional) jump instructions • Compare instructions • Modes of addressing	**2**
3(c)	**1 mark** per bullet • Storing 0 in 401 (line 51) • Loading memory location 300, value 2 to ACC (line 52) • Adding 64 to ACC to give 66 (line 55) • Outputting B (line 56) • Load 0 (line 57), increment ACC (line 58) and store 1 in 401 (line 59) • Incrementing IX (line 60) • Loading 5 (line 52), adding 64 (line 55), outputting E (line 56) loading 1 (line 57), incrementing ACC (line 58), storing 2 in 401 (line 59) and incrementing IX (line 60) • Load 0 (line 52) and end	**8**

Instruction address	ACC	Memory address						IX	OUT	
		300	301	302	303	400	401			
		2	5	0	4	64		0		
50	0									
51							0			[1]
52	2									[1]
53										
54										
55	66									[1]
56									B	[1]
57	0									[1]
58	1									
59							1			
60								1		[1]
61										
52	5									
53										
54										
55	69									
56									E	[1]
57	1									
58	2									
59							2			
60								2		
61										
52	0									[1]
53										
54										
62										

Question	Answer	
3(d)(i)	**1 mark** for correct answer 0100 0001	
3(d)(ii)	**1 mark** for correct answer 41	**1**
3(d)(iii)	**1 mark** for correct answer 0044	**1**

QUESTION 7.

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>4(a)</td><td>1 mark per bullet

• LDM #300 The (denary) number 300 is loaded (into the register)
• LDD 300 The contents of address 300 are loaded (into the register)</td><td>2</td></tr>
<tr><td>4(b)</td><td>1 mark for JPN correct
1 mark for both of ADD and DEC correct
1 mark for LDR correct

<table>
<tr><th></th><th>Description</th><th>Jump instruction</th><th>Arithmetic operation</th><th>Data movement</th></tr>
<tr><td>LDR #3</td><td>Load the number 3 to the Index Register</td><td></td><td></td><td>✓</td></tr>
<tr><td>ADD #2</td><td>Add 2 to the Accumulator</td><td></td><td>✓</td><td></td></tr>
<tr><td>JPN 22</td><td>Move to the instruction at address 22</td><td>✓</td><td></td><td></td></tr>
<tr><td>DEC ACC</td><td>Subtract 1 from the Accumulator</td><td></td><td>✓</td><td></td></tr>
</table>
</td><td>3</td></tr>
</table>

Question	Answer	
4(c)(i)	**1 mark** for hardware interrupt **1 mark** for software interrupt For example: Hardware interrupt • Printer out of paper • No CD in drive Software interrupt • A running program needs input • Runtime error, e.g. division by zero	
4(c)(ii)	**1 mark** per bullet to **max 5** • At the start / end of each fetch-execute cycle the processor checks for interrupt(s) • Check if an interrupt flag is set // Check if bit set in interrupt register • Processor identifies source of interrupt • Processor checks priority of interrupt • If interrupt priority is high enough // Lower priority interrupts are disabled • Processor saves current contents of registers // saves current job on stack • Processor calls interrupt handler / Interrupt Service Routine (ISR) • Address of ISR is loaded into Program Counter (PC) • When servicing of interrupt complete, processor restores registers // job from stack is restored • Lower priority interrupts are re-enabled • Processor continues with next F–E cycle	**5**