**Q1.**

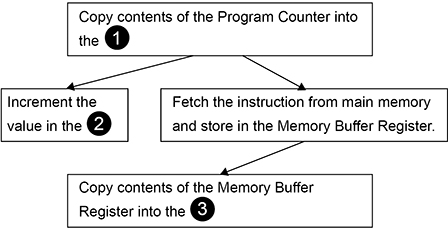
Describe **four** steps that a processor goes through during the fetch stage of the Fetch-Execute cycle.

You **must** explain the purpose of each step.

**(Total 8 marks)**

**Q2.**

The diagram below describes the fetch part of the Fetch-Execute cycle. Some of the names of registers have been omitted from the figure and replaced with the numbers  to 



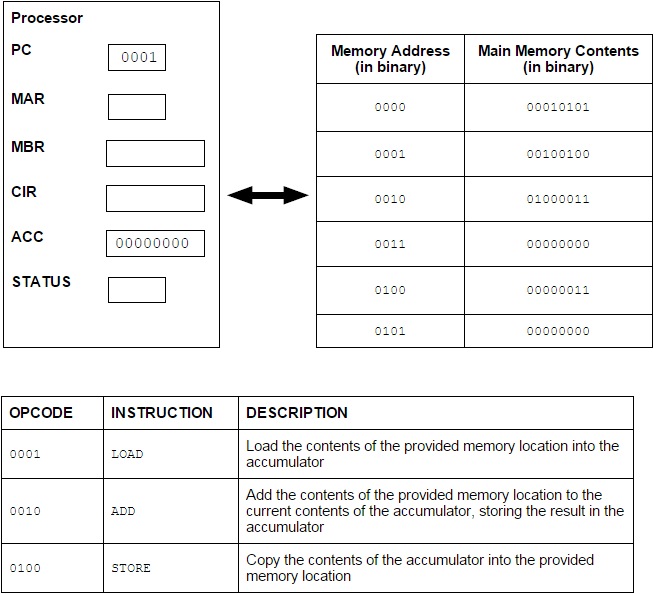
State the **full names** of the registers that should appear in the diagram where the numbers are.

|  |  |
| --- | --- |
| **Number** | **Full Name of Register** |
|  |  |
|  |  |
|  |  |

**(Total 2 marks)**

**Q3.**

The diagram below shows some of the registers used in the fetch-execute cycle of a simple processor and the contents of a small section of main memory that it is connected to by the system bus ().



(a)     In the diagram above the first 4 bits of an instruction represent the opcode and give the type of instruction to be executed.

What name is given to the second 4 bits of an instruction?

**(1)**

(b)     (i)      Currently the value in the Program Counter (PC) is example 0001.

Complete the table below by writing the values, expressed in binary, in the following registers after completing the fetch part of the fetch-execute cycle.

|  |  |
| --- | --- |
| **Register** | **Value** |
| PC |  |
| MAR |  |
| MBR |  |

**(3)**

(ii)     Describe what will happen during the decode and execute part of the cycle.

**(3)**

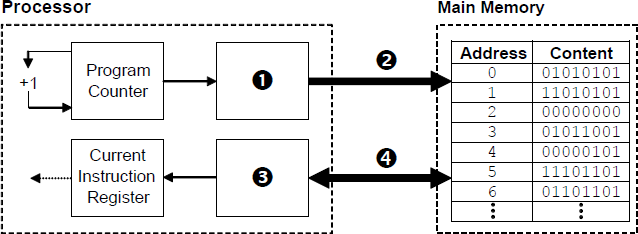
(c)     What would be the outcome of executing the instruction 01000011?

**(1)**

**(Total 8 marks)**

**Q4.**

The diagram below shows the processor registers and busses that are used during the fetch part of the fetch-execute cycle, together with the main memory. The values stored in memory locations 0 to 6 in the main memory are machine code instructions.



(a)     Name the components that are labelled with the numbers 1 to 4. In the case of register names, the full names must be stated.

|  |  |
| --- | --- |
| **Number** | **Component Name** |
|  |  |
|  |  |
|  |  |
|  |  |

**(4)**

(b)     Explain what happens during the decode and execute stages of the fetch-execute cycle.

**(3)**

(c)     The machine code instructions in the main memory in the diagram above are shown in binary.  
When programmers look at machine code instructions they usually prefer to view them in hexadecimal.

State **one** reason why this is the case.

**(1)**

(d)     The machine code instructions in the main memory in the diagram above were produced when an assembly language program was translated into machine code.

(i)      What type of program translator was used to do this?

**(1)**

(ii)     Most computer programs are initially written in an imperative high level language rather than assembly language.

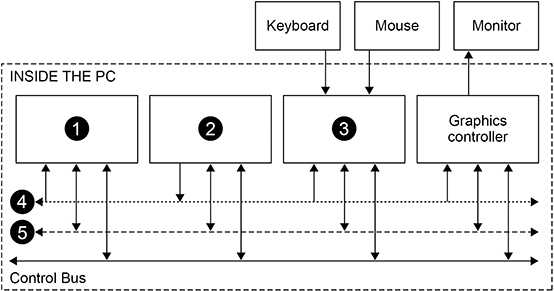
Explain why this is the case.

**(3)**

**(Total 12 marks)**

**Q5.**

The diagram below shows how some of the components inside a computer are connected together. The computer uses the von Neumann architecture. Some of the names of components have been omitted from diagram and replaced with the numbers  to 



Complete the table below by writing in the **Component Number** column the numbers from the diagram that correspond to the **Component Names**.

|  |  |
| --- | --- |
| **Component Name** | **Component Number (1–5)** |
| Address Bus |  |
| Data Bus |  |
| Main Memory |  |
| Processor |  |
| USB I/O Controller |  |

**(Total 2 marks)**

**Q6.**

The two most common computer architectures are **Harvard** and **von Neumann**.

(a)     Describe **one** difference between the way the Harvard and von Neumann architectures operate.

**(2)**

(b)     Shade **one** lozenge to indicate the type of computer architecture that is typically used for digital signal processing.



**(1)**

**(Total 3 marks)**

Mark schemes

**Q1.**

**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]**

**Q2.**

**2 marks for AO1 (knowledge)**

|  |  |
| --- | --- |
| **Number** | **Full Name of Register** |
|  | Memory Address Register **NE.** MAR |
|  | Program Counter **NE.** PC |
|  | 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]**

**Q3.**

(a)     operand;

**R** operand code

**1**

(b)     (i)      PC        0010;   
MAR     0001;  
MBR     00100100;

**3**

(ii)     The instruction is held in the CIR // instruction in CIR is decoded;  
**A** IR

The control unit / instruction decoder decodes the instruction;  
**NE** the processor decodes the instruction

Instruction will be split into opcode and operand;  
**R** if it is implied that a register will do this splitting / decoding

Relevant part of processor / CPU executes instruction // using ALU to perform calculations;  
**A** instruction executed by the control unit / ALU  
**NE** processor executes instruction

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;

*By example:*Will ADD contents memory location 0100 to accumulator;

**MAX 3**

(c)     The current value in the accumulator would be stored in (memory) address / location 0011 / 3;

Number 011 / 3 stored in (memory) address / location 0011 / 3;

**MAX 1**

**[8]**

**Q4.**

|  |  |
| --- | --- |
| (a)     **Number** | **Component Name** |
| 1 | Memory Address Register |
| 2 | Address Bus |
| 3 | Memory Data / Buffer Register |
| 4 | Data Bus |

**4**

(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**

(c)     Can be displayed in less space;   
**R** takes up less space **NE**Easier to remember / learn / read / understand;  
Less error prone;

**Max 1**

(d)     (i)     Assembler;

**1**

(ii)     HLLs are problem oriented;  
HLL programs are portable // machine / platform independent ;  
English like **keywords / commands/ syntax / code**;  
**R** closer to English  
Less code required // less tedious to program //  
one to many mapping of HLL statements to machine code commands;  
Quicker/easier to understand / write / debug /learn / maintain code;  
**R** just quicker/easier  
HLLs offer extra features e.g. data types / structures // structured statements // local variables // parameters // named variables/constants;  
**R** procedures / modular  
**A** example of a data structure  
**NE** “extra features” without example  
Speed of execution not crucial for most tasks so faster execution of assembly language not required;  
Most computer systems have a lot of (main) memory / RAM so compact object code not essential;  
**A** converse points for Assembly Language

**3**

**[12]**

**Q5.**

**2 marks for AO1 (understanding)**

|  |  |
| --- | --- |
| **Component Name** | **Component Number (1–5)** |
| Address Bus | 4 |
| Data Bus | 5 |
| Main Memory | 1 |
| Processor | 2 |
| USB I/O Controller | 3 |

**1 mark:** At least three components correctly numbered

**2 marks:** All five components correctly numbered

**[2]**

**Q6.**

(a)  **Marks are for AO1 (understanding)**

Harvard uses separate memory/bus/address space // von Neumann uses combined memory/bus/address space; for instructions/program and data;

**NE**. Places, locations, registers, areas of memory

**A**. Main memory

**NOTE**. It must be clear that instructions/data are stored in separate memory, not separately in memory.

**2**

(b)  **Mark is for AO1 (knowledge)**

Harvard;

**R**. more than one lozenge shaded

**1**

**[3]**