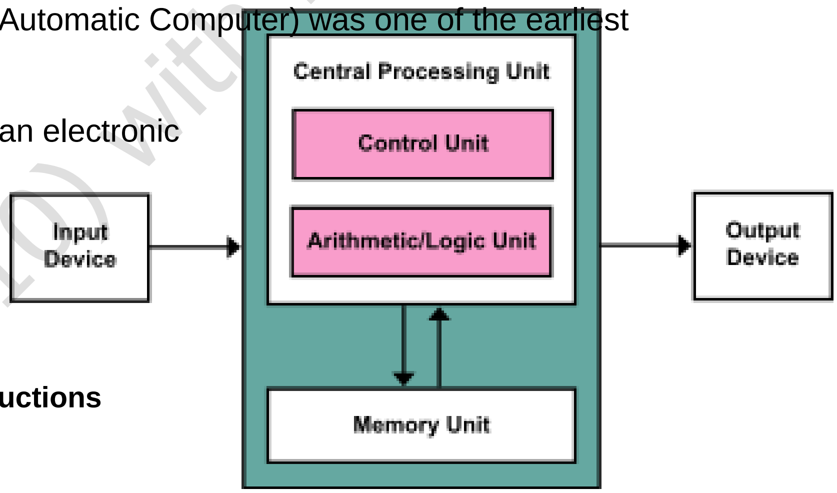
Von Neumann Model:

The earliest computers were not "programmable". They were designed to do specific tasks only. **Reprogramming** when it was possible at all was a tedious process, starting with flowcharts and paper notes, followed by detailed engineering designs, and then the often process of physically re-wiring and re-building the machine.

It could take three weeks to set up a program on ENIAC (a computer of 1940s) and get it working. **ENIAC (Electronic Numerical Integrator and Computer) was the first electronic general-purpose computer. It was Turing-complete, digital, and capable of being reprogrammed to solve "a large class of numerical problems**

The **von Neumann architecture**, also known as the **von Neumann model** and **Princeton Architecture**, is based on John **von Neumann's (**mathematician and physicist**)** research paper in **1945** and others in the First Draft of a Report on the **EDVAC**.
EDVAC (Electronic Discrete Variable Automatic Computer) was one of the earliest electronic computers.



This described design architecture for an electronic digital computer with parts consisting

- **Central Processing Unit containing:**
  - **Control Unit**
  - **Arithmetic/Logic unit**
  - **Processor registers,**
  - **Memory to store data & instructions**
- **Input / Output Mechanism**
- **External Storage**

This describes **design architecture for an electronic digital computer** with **subdivisions of a central arithmetic part,** a **central control part**, a **memory** to store both **data and instructions**, **external storage**, **and input and output mechanisms.**

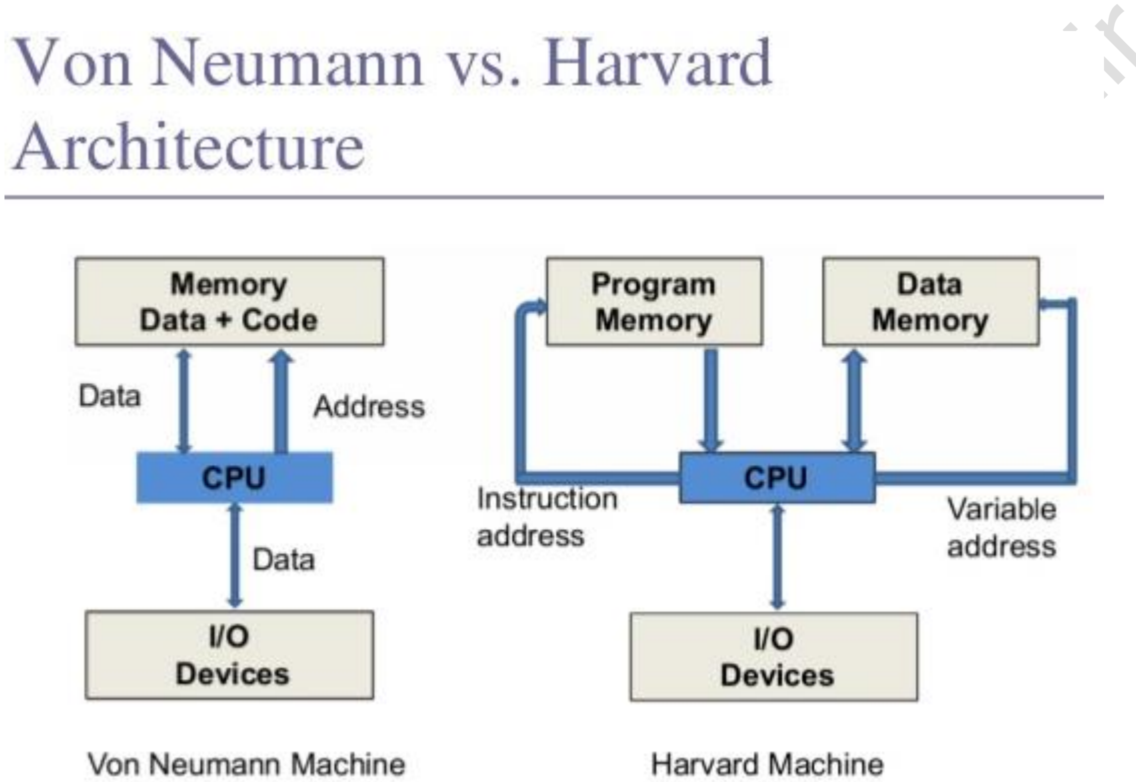
The meaning of the phrase has evolved to mean a stored-program computer. A stored-program digital computer is one that keeps its programmed instructions, as well as its data, in read-write, random-access memory **(RAM)**

So **John Von Neumann** introduced the idea of the **stored program.**

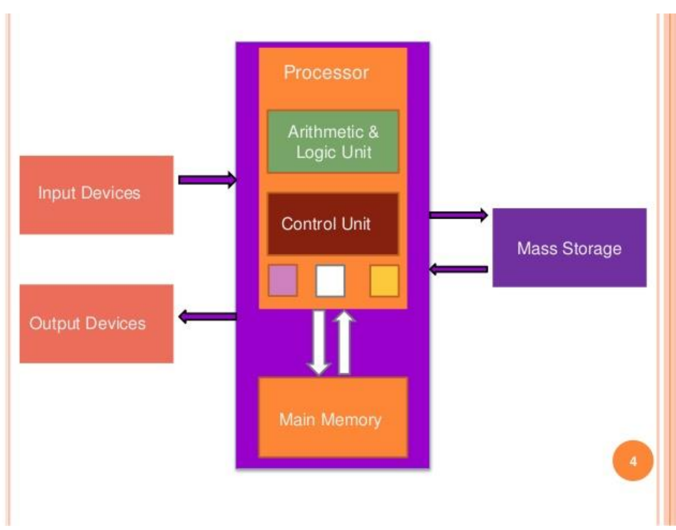
Previously data and programs were stored in separate memories. Von Neumann realized that data and programs are indistinguishable and can, therefore, use the same memory

The Von Neumann architecture uses a single processor which follows a linear sequence of fetch-decode-execute.

The Picture below shows difference between Von Neumann architecture and Harvard architecture (earliest computers)



Features of a Von Neumann architecture



The illustration above shows the essential features of the Von Neumann or stored-program architecture.

Memory
The computer will have memory that can hold both data and also the program processing that data. In modern computers this memory is RAM.

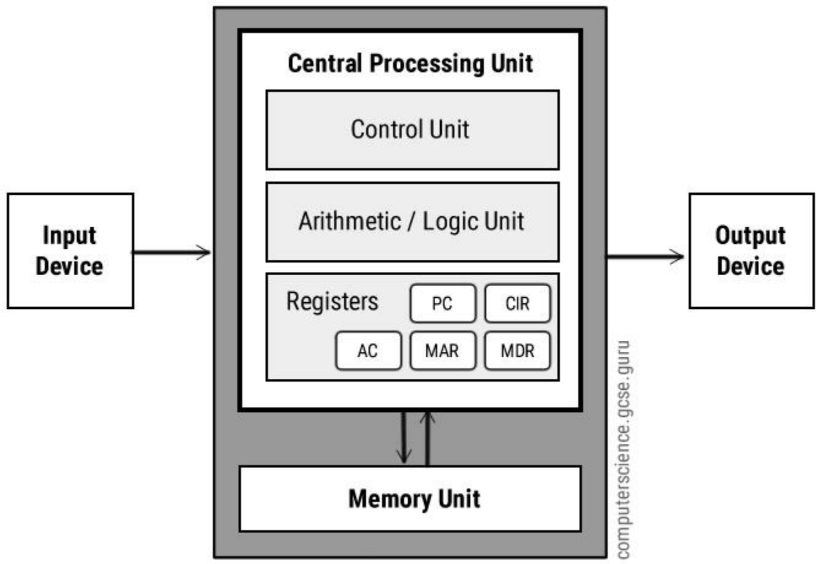
Control Unit

The control unit will manage the process of moving data and program into and out of memory and also deal with carrying out (executing) program instructions - one at a time. This includes the idea of a 'register' to hold intermediate values. In the illustration above, the 'accumulator' is one such register.

The 'one-at-a-time' phrase means that the Von Neumann architecture is a **sequential processing machine**.

Input - Output

This architecture allows for the idea that a person needs to interact with the machine. Whatever values that is passed to and forth is stored once again in some internal registers.



Arithmetic Logic Unit

This part of the architecture is solely involved with carrying out calculations upon the data. All the usual Add, Multiply, Divide and Subtract calculations **will be available but also data comparisons such as '**Greater Than', 'Less Than', 'Equal To' **will be available.**

Registers:

The Von Neumann architecture uses a single processor which follows a linear sequence of **fetch-decode-execute. In order to do this, the processor has to use some special registers,** which are **discrete memory locations with special purposes attached**.

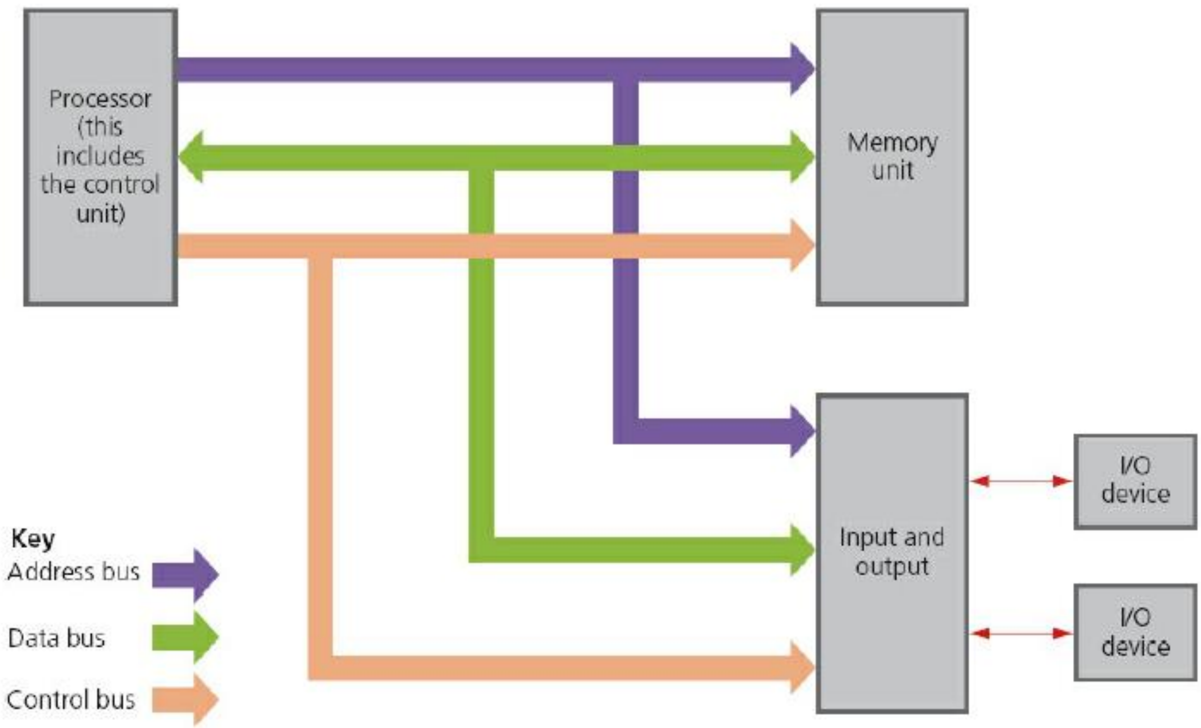
These are:

Register	Name/ Function
PC	**Program Counter**- keeps track of where to find the next instruction so that a copy of the instruction can be placed in the current instruction register
MAR	**Memory Address Register**- to hold the memory address that contains either the next piece of data or an instruction that is to be used.
MDR	**Memory Data Register:** acts like a buffer and holds anything that is copied from the memory ready for the processor to use it
CIR	**Current Instruction Register:** The current instruction register holds the instruction that is to be executed
ACC	**Accumulator:** This is simply the special register where data is worked on. Again, you can think of it as a box. If I wanted to add 4 to 7, for example, I would fetch 4 from RAM and put 4 in the Accumulator. I would then get 7 from RAM and add that to whatever was in the Accumulator. I would then store the result briefly in the Accumulator before moving it back to somewhere in RAM to be used later. All calculations of any description are done using the Accumulator. In fact, CPUs often have a few of these important registers, to help them process data quickly.

Bus

Notice the arrows between components? This implies that information should flow between various parts of the computer. In a modern computer built to the Von Neumann architecture, information passes back and forth along a 'bus'. There are buses to identify locations in memory - an 'address bus'

Type of bus	Description of bus	Data/signal direction
address bus	carries signals relating to addresses (see later) between the processor and the memory	unidirectional (signals travel in one direction only)
data bus	sends data between the processor, the memory unit and the input/output devices	bi-directional (data can travel in both directions)
control bus	carries signals relating to the control and coordination of all activities within the computer (examples include: the read and write functions)	unidirectional (signals travel in one direction only)



CPU and Fetch-Execute Cycle

The Fetch-Decode-Execute-Reset Cycle:

The following is an algorithm that shows the steps in the cycle. At the end the cycle is reset and the algorithm repeated.

1. **Load the address that is in the program counter (PC) into the memory address register (MAR).**
2. **Load the instruction that is in the memory address given by the MAR into the memory data register (MDR).**
3. **Load the instruction that is now in the MDR into the current instruction register (CIR).**
4. **Increment the PC by 1.**
5. **Decode the instruction that is in the CIR.**
6. **If the instruction is a jump instruction then**
    a. **Load the address part of the instruction into the PC**
    b. **Reset by going to step 1.**
7. **Execute the instruction.**
8. **Reset by going to step 1**

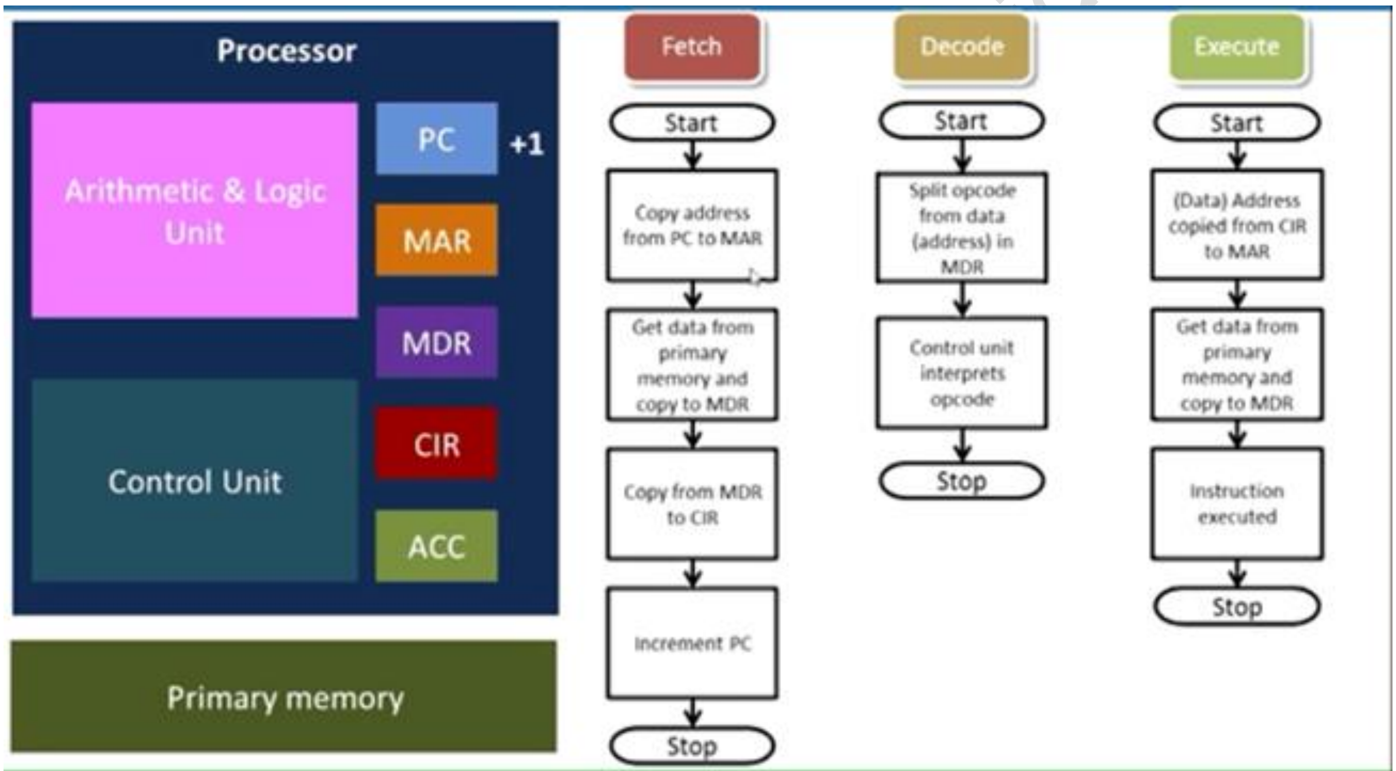
Steps 1 to 4 are the ***fetch*** part of the cycle.

Steps 5, 6a and 7 are the execute part of the cycle and steps

6b and 8 are the reset part.

Step 1 simply places the address of the next instruction into the memory address register so that the control unit can fetch the instruction from the right part of the memory.

The **program counter** is then incremented by 1 so that it contains the address of the next instruction, assuming that the instructions are in consecutive locations.



The **MDR memory data register** is used whenever anything is to go from the central processing unit to main memory, or vice versa. Thus the next instruction is copied from memory into the MDR and is then copied into the current instruction register.

Now that the instruction has been **fetched** the control unit can **decode** it and decide what has to be done. This is the execute part of the cycle.

If it is an arithmetic instruction, this can be executed and the cycle restarted as the PC contains the address of the next instruction in order. However, if the instruction involves jumping to an instruction that is not the next one in order, the PC has to be loaded with the address of the instruction that is to be executed next. This address is in the address part of the current instruction, hence the address part is loaded into the PC before the cycle is reset and starts all over again.

Past paper Question June 2015

7 (a) One of the key features of von Neumann computer architecture is the use of buses. Three buses and three descriptions are shown below.

Draw a line to connect each bus to its correct description.

Bus	Description
address bus	this bus carries signals used to coordinate the computer's activities
control bus	this bi-directional bus is used to exchange data between processor, memory and input/ output devices
data bus	this uni-directional bus carries signals relating to memory addresses between processor and memory

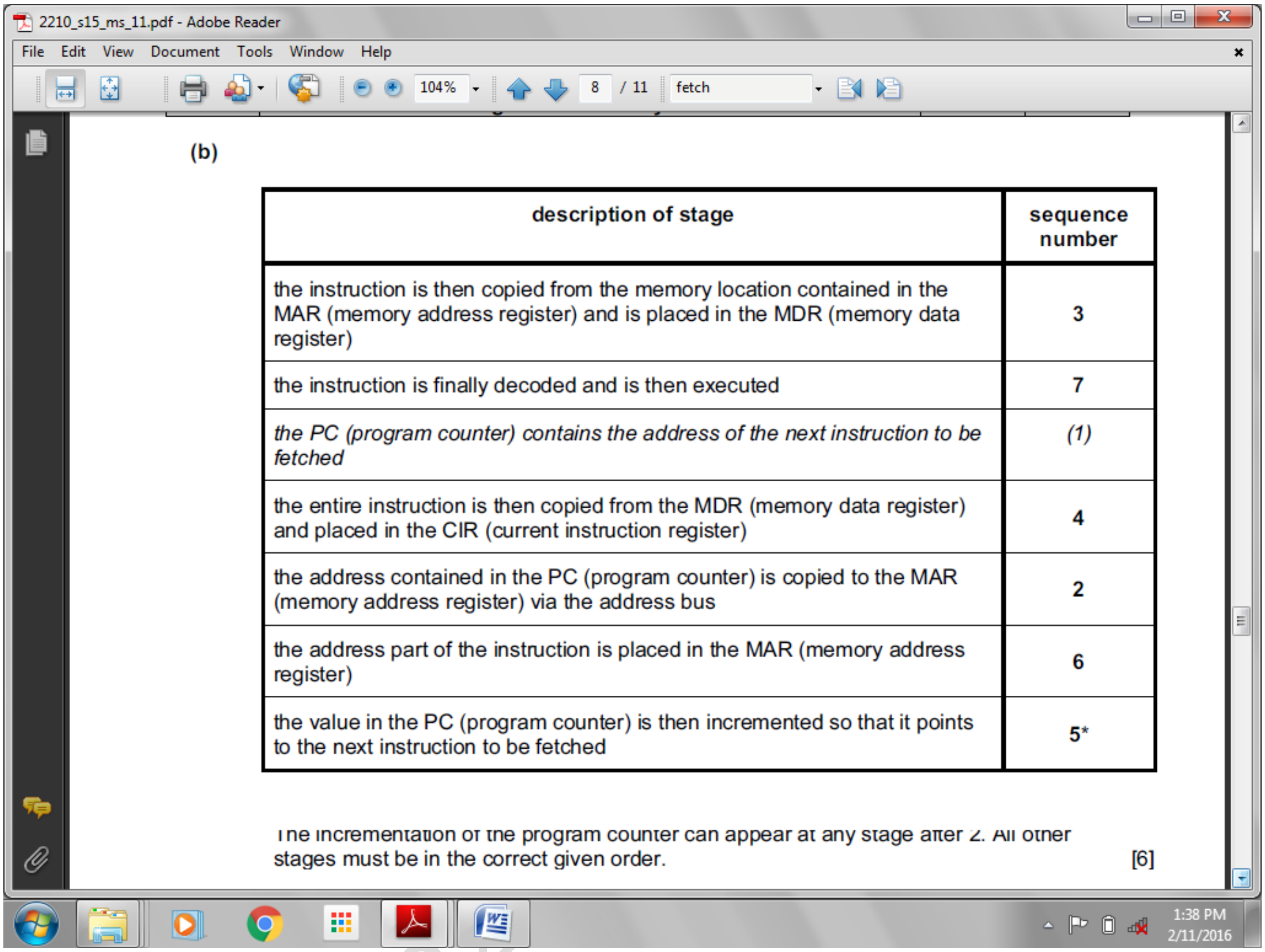
[2]

(b) The seven stages in a von Neumann fetch-execute cycle are shown in the table below.
Put each stage in the correct sequence by writing the numbers 1 to 7 in the right hand column.

The first one has been done for you.

Stage	Sequence number
the instruction is then copied from the memory location contained in the MAR (memory address register) and is placed in the MDR (memory data register)	
the instruction is finally decoded and is then executed	
the PC (program counter) contains the address of the next instruction to be fetched	1
the entire instruction is then copied from the MDR (memory data register) and placed in the CIR (current instruction register)	
the address contained in the PC (program counter) is copied to the MAR (memory address register) via the address bus	
the address part of the instruction, if any, is placed in the MAR (memory address register)	
the value in the PC (program counter) is then incremented so that it points to the next instruction to be fetched	

[6]



(b)

description of stage	sequence number
the instruction is then copied from the memory location contained in the MAR (memory address register) and is placed in the MDR (memory data register)	3
the instruction is finally decoded and is then executed	7
the PC (program counter) contains the address of the next instruction to be fetched	*(1)*
the entire instruction is then copied from the MDR (memory data register) and placed in the CIR (current instruction register)	4
the address contained in the PC (program counter) is copied to the MAR (memory address register) via the address bus	2
the address part of the instruction is placed in the MAR (memory address register)	6
the value in the PC (program counter) is then incremented so that it points to the next instruction to be fetched	5*

The incrementation of the program counter can appear at any stage after 2. All other stages must be in the correct given order. [6]

Refrences:
www.wikipedia.com
VCN – ICT Department 2013 Prepared by *Davis Rwatooro T.*