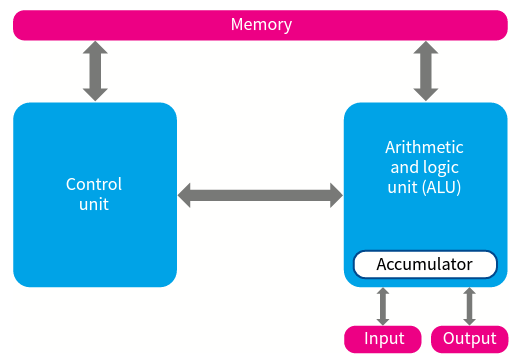
**Chapter 5 processor fundamentals**

**4.1 Von Neumann architecture**



1. Central processing unit (CPU) – the main processing unit of the system, also known as the processor

**i control unit (CU)** - it controls the interaction between the different parts of the CPU

The function of CU: one is controlling the flow of data throughout the processor, another is ensuring that program instructions are handled correctly

**ii arithmetic logic unit (ALU)** – it carries out calculations on data.

Responsible for any arithmetic or logic processing that might be needed when a program is running.

**iii** **registers**

These are internal memory locations within the CPU. The temporarily hold data and instructions during processing.

* a) **Program Counter (PC)** - an incrementing counter that keeps track of the **next memory address** of the instruction that is to be executed once the execution of the current instruction is completed.

**Stores the address of where the next instruction is to be read from**

* b) **Memory Address Register (MAR)** - the address in main memory that is currently being read or written

**Stores the address of a memory location which is about to have a value read from or written to**

* c) **Memory Data Register (MDR)** - a two-way register that holds data fetched from memory (and ready for the CPU to process) or data waiting to be stored in memory

**Stores data that has just been read from memory or written to memory**

* d) **Current Instruction register (CIR)** - a temporary holding ground for the instruction that has just been fetched from memory

**Stores the current instruction while it is being decoded and executed**

* e) **Accumulator Register (ACC)** is used for storing data for ALU to process and the results those are produced by the ALU.

**Accumulator stores a value before and after the execution of an instruction by the ALU**

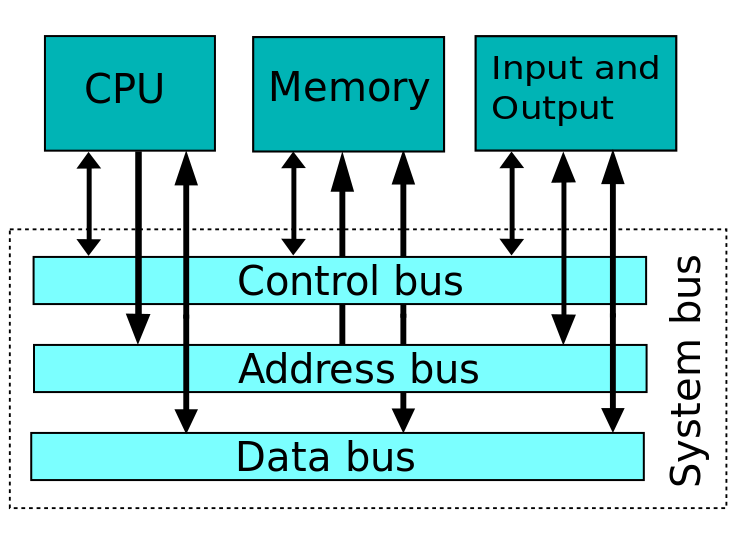
* **f) Index register (IX)**

Stores a value, only used for indexed addressing

* **g) Status register (SR)**

Contains individual bits that are either set or cleared

1. **Buses** – it is a series of conductors which can be considered a sort of ‘highway’ for information.



|  |  |  |
| --- | --- | --- |
| **Type of bus** | **Description of bus** | **Data direction** |
| Address bus | Carries signals relating to addresses between the processor and the memory | Unidirectional |
| Data bus | Sends data between the processor, the memory unit and the input/output devices | Bi-directional |
| Control bus | Carries signals relating to the control and coordination of all activities within the computer | Both unidirectional and bi-directional |

1. **Increasing performance**

If we want to increase the performance of our computer, we can try several things:

1. Increasing the clock speed

**Clock speed - The number of cycles that are performed by the CPU per second**

* Clock speed is measured in Hertz, which means 'per second'.
* A computer of speed 3.4 GHz means it might be capable of processing 3,400,000,000 instructions per second!

**What does a processor clock do:**

* Synchronises the operation of the processor

(2)Increasing bus widths

**Bus Size - The number of bits of information a bus can carry at one time (the number of wires making up a bus)**

(3)Adjusting word length

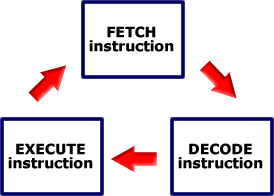
**Word size - The number of bits of information that a processor can process at one time**

1. **Universal serial bus(USB)**

Advantage

* devices automatically detected and configured when first 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

1. **The fetch-execute cycle**

****

**Fetch**

1. **Load the address of next instruction in the PC into the MAR.**
   * **So that the control unit can fetch the instruction from the right part of the memory.**
2. **Copy the instruction/data that is in the memory address given by the MAR into the MDR.**
   * **MDR is used whenever anything is to go from the CPU to main memory.**
3. **Increment the PC by 1.**
   * **So that it contains the address of the next instruction, assuming that the instructions are in consecutive locations.**
4. **Load the instruction/data that is now in the MDR into the CIR.**
   * **Thus the next instruction is copied from memory -> MDR -> CIR.**

**Decode**

1. **Contents of CIR split into operation code and address if present e.g. store, add or jump instructions.**
2. **Decode the instruction that is in the CIR.**

**Execute**

1. **Execute the instruction but what is involved in this depends on the instruction being executed (there are several different instructions you need to know about).**

* If the instruction is a jump instruction then
  + Load the address part of the instruction in the CIR into the PC.
* If the instruction is an input / load (directly) instruction then take data input and place in accumulator.
* If the instruction is a load (from memory) instruction.
  + Copy address part of the instruction (to load from) in the CIR into MAR.
  + Copy data from memory address held in MAR to MDR.

Copy data in MDR into accumulator.

* If the instruction is a store instruction then:
  + Copy address part of the instruction (to store in) in the CIR into MAR.
  + Copy data in accumulator to MDR.
  + Copy data in MDR into memory address held in MAR.
* If the instruction is an add instruction then:
  + Copy address part of the instruction (of number to add) in the CIR into MAR.
  + Copy number from memory address held in MAR into MDR.
  + Add number in MDR to number in accumulator (accumulator will now hold the result).
* If the instruction is an output (directly from accumulator) then output number in accumulator.
* If the instruction is an output (from memory) instruction then:
  + Copy address part of part of the instruction (of data to output) in CIR into MAR.
  + Output contents of MDR.

**Reset**

1. **Cycle is reset (restarted) by passing control back to the PC (step 1).**
2. Register transfer notation

**MAR [PC]**

**PC [PC]+1; MDR [[MAR]]**

**CIR [MDR]**

1. **Interrupt**

'An interrupt is a signal for the CPU to stop what it is doing and instead carry out the interrupt task, once the task is complete, the CPU goes back to what it was doing'.

* **The reasons for an interrupt**

A fatal error in a program

A hardware fault

A need for I/O processing to begin

User interaction

A timer signal

**The actions of the processor when an interrupt is detected**

* 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.