**Chapter 3: x86 PROCESSOR ARCHITECTURE**

**Topic – 1: General Concepts**

**Introduction**

* x86 microprocessors are used in both **Intel** & **AMD** processors.
* This includes **Intel IA-32** & **Intel 64** processors like **Intel Pentium** & **Core-Duo**.
* **AMD:** Advanced micro devices.
* It also includes AMD processors such as **Athlon**, **Phenom**, **Opteron** & **AMD64**.

**Basic Microcomputer Design**

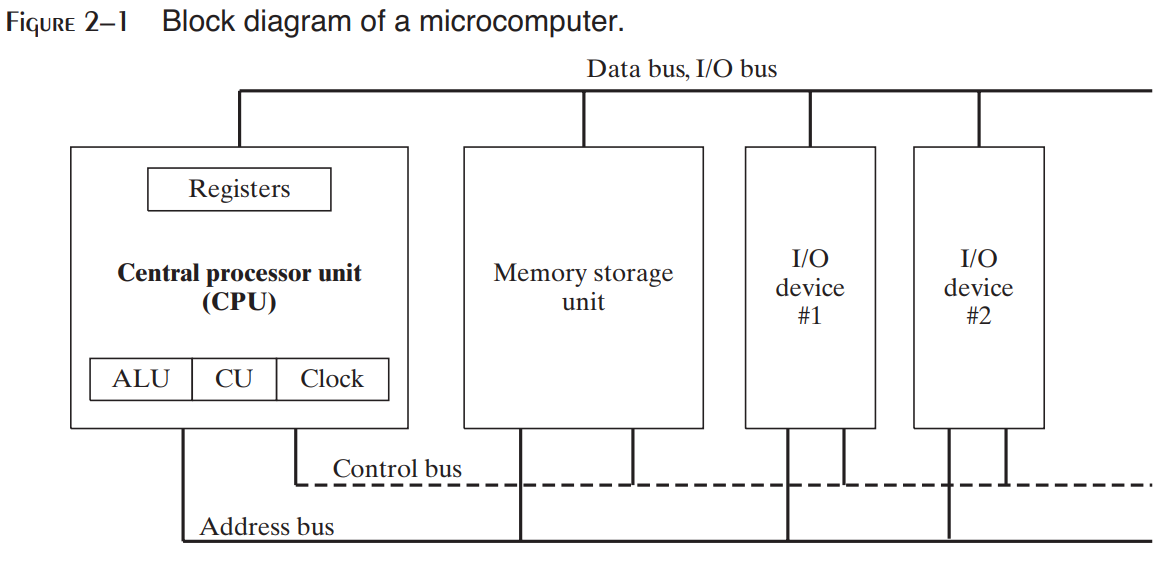
* CPU contains: **Registers, high-frequency clock, a CU (control unit) & ALU.**
* **CU:** Puts machine instructions to be executed in a meaningful sequence.
* **ALU:** Performs **arithmetic** & **logical** (AND, OR, NOT etc) operations.
* CPU is attached to whole computer via its **pins** which are connected to **socket** present in the motherboard.
* Its most pins are connected to the **data bus**, **control bus** & **address bus**.

**Data Transfer Mechanism**

* **Memory storage unit:** Stores the **instructions** & **data** of a running program.
* HDD & SSD etc are type of **memory storage unit**.
* For execution of a program, it is first loaded to **RAM**.
* CPU requests **storage unit** for data.
* Programs are then copied from **RAM** to **CPU** for being processed.
* And finally, data are transferred from **CPU** to **storage unit**.

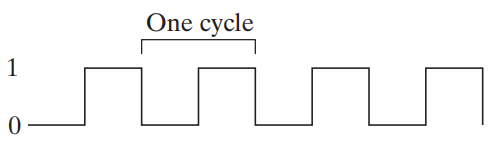
**Buses**

* **Bus:** Group of **parallel wires** that transfer data from one part to another in computer.
* Types of buses: **Data, I/O, control & address.**
* **Data bus:** Transfers instructions & data between **CPU** & **memory**.
* **I/O bus:** Transfers data between **CPU** and **peripheral** **devices**.
* **Control bus:** **Synchronizes** all devices connected to bus system by **using binary signals**.
* **Address bus:** Contains addresses of instructions & data being **transferred** between CPU & memory.



**Clock**

* **Synchronizes** CPU & system bus when they are active.
* Pulses at a constant rate.
* **Machine cycle/ clock cycle:** Unit of time for measuring machine instructions.



* **One clock cycle** is the time required for **one clock pulse** to complete.

**Duration of clock cycle = Reciprocal of clock’s speed**

**Example**

**A clock that oscillates 1 billion times per second (clock speed of 1 GHz), has a clock cycle of 1 billionth of a second (1 nano second).**

* One machine instruction requires **atleast** oneclock cycle to execute.
* Some instructions require **more than 50** clock cycles to execute.
* Like multiply instruction in 8088 processor.
* **Wait states:** Empty clock cycles.
* It is called **wait state** due to **unsynchronized** CPU, system bus & memory circuits for that duration.
* Instructions requiring **memory access** often go through **wait states**.
* Clock pulses are required mainly for **executing** instructions.
* That’s why when accessing memory, clock pulses are temporarily halted (clock cycle).

**Instruction Execution Cycle**

* It is a **predefined procedure** CPU goes through for executing a machine instruction.
* **Instruction pointer register:** Contains the address of the instruction to be executed.
* The **bit pattern** of an instruction can also tell us what operands (inputs) it has.

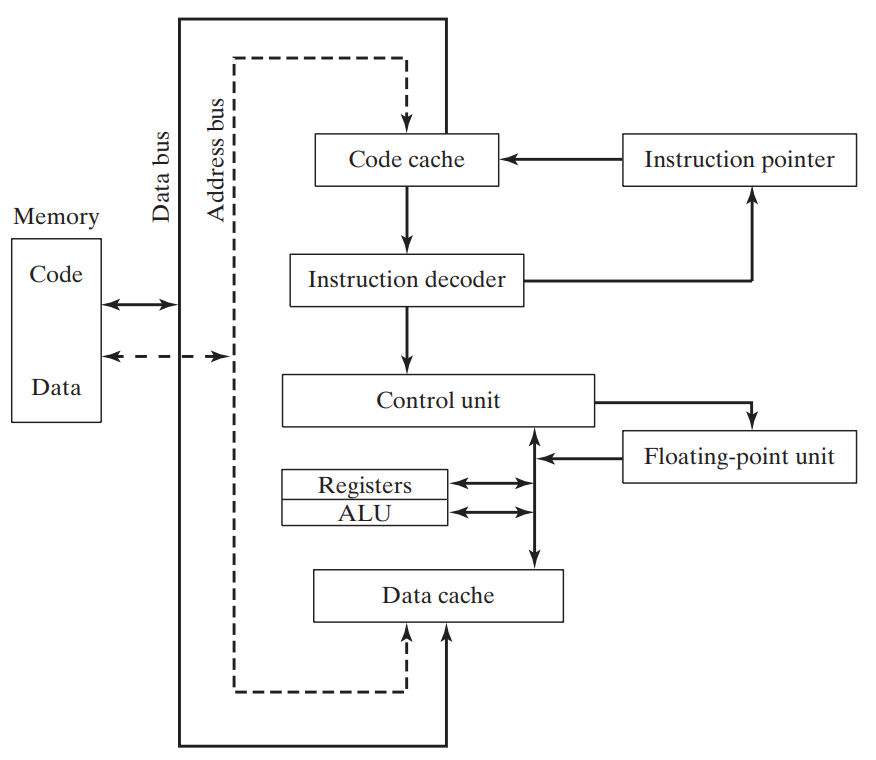
**Topic – 2: Steps for Instruction Execution**

**Steps In Brief**

* **Step 1:** CPU fetches instruction from **instruction queue**.
* **Step 2:** CPU increments the **instruction pointer**.
* **Step 3:** CPU decodes the instruction by reading its binary **bits signature**.
* **Step 4:** If **operands** are involved, they are **fetched** from registers & memory.
* **Step 5:** CPU executes the instruction using those operands.
* **Step 6:** CPU updates some **status flags** (zero, overflow, carry etc).
* **Step 7:** If an **output operand** is involved, then CPU stores output result in it.

**In Nutshell**

* In short – **fetch**, **decode** & **execute**.
* **Z = X + Y** [**X** & **Y** are **input** operands, **Z** is **output** operand]



**Reading Instructions From Memory**

* Computer reads memory slower than accessing registers.
* This is because **4 steps** are involved when reading a value from memory.

**Steps In Brief - II**

* **Step 1:** Address is sent through address bus.
* **Step 2:** Change the processor’s **RD** (**read**) pin’s status.
* **Step 3:** Then memory chips take **one** **clock cycle** to respond.
* **Step 4:** Copy of requested data is sent through **data bus** to the **destination operand**.

**About Steps**

* Each of these steps take **one clock cycle** each.
* Whereas the **CPU registers** are accessed in **one clock cycle**.
* To solve this issue, the CPU designers created **memory *cache***.
* **Cache** is a **fast** access memory which stores **recently used** instructions.
* This is because recently used instructions are **expected** to be used again soon.
* **Cache** stores both **memory** of the instruction & the **codes** of instructions.
* **Cache hit:** CPU **finds** something it was finding in **cache**.
* **Cache miss:** CPU **doesn’t** **find** something it was finding in **cache**.

**Types Of Cache**

* **Level – I cache:** Also known as **primary cache** & stored on **CPU** itself.
* **Level – II cache:** Also known as **secondary cache** & stored **near the CPU**.
* **Level – II cache** is little **slower** than **level – I cache** & attached to CPU through a high-speed **bus**.
* **Cache** is **faster** than **conventional RAM** because it uses ***static RAM***.
* **Conventional RAM:** Dynamic RAM
* **Static RAM** is faster than **dynamic RAM** because it holds its contents **without** being refreshed.
* Thus, its **expensive** too.

**Loading & Executing Program**

* ***Program loader*** loads the program into memory.
* Then the OS points CPU to the **entry point** of our program.
* The **address** of **entry point** is from where the execution has to start.
* But there are **more** steps involved in it.

**Steps In Brief - III**

* **Step 1:** OS **searches** for the program using its **filename** in the current & surrounding directories.
* **Step 2:** Then the OS **finds location** of **next free block** of memory in RAM.
* **Step 3:** When found, information like its **file size** & **physical location** on disk are fetched.
* This block is called ***descriptor table***.
* **Step 4:** Then a ***process ID*** is given to the running program (**process**).

***\*It is OS’s responsibility to manage resources requested by the instruction & keep track of the program\****

**Task Manager Processes (Windows)**

* Applications processes
* Background processes
* Windows processes