

//2023//

## //Long//

### 1. Draw the Von Neumann basic structure and mark all its components.

The Von Neumann architecture is a computer architecture model that is based on a design where the central processing unit (CPU), memory, and input/output devices are connected through a common data bus. The structure is named after John von Neumann, a mathematician and computer scientist. Here's a breakdown of the key components:

1. **CPU (Central Processing Unit):** The CPU is the core of the system and contains two main parts:
  - **ALU (Arithmetic and Logic Unit):** Performs mathematical calculations and logical operations.
  - **Control Unit (CU):** Directs the operation of the processor by interpreting instructions from memory.
2. **Memory:** A storage area that holds both data and program instructions. In Von Neumann architecture, both data and instructions are stored in the same memory.
3. **Input Devices:** Devices such as a keyboard, mouse, or sensors that allow data to enter the system.
4. **Output Devices:** Devices such as a monitor or printer that allow the system to communicate results to the user.
5. **Bus System:** A communication pathway used to transfer data between the components of the system, typically consisting of three types:
  - **Data Bus:** Carries the actual data being transferred.
  - **Address Bus:** Carries the address of the location in memory where data is to be read or written.
  - **Control Bus:** Carries control signals to manage the operation of the CPU and memory.

\*\* Write down the IEEE format for single and double precision numbers?

>>The IEEE 754 standard defines how floating-point numbers are represented in computers, specifying formats for both single precision and double precision. Here's the breakdown of these formats:

### **1. Single Precision (32-bit) Format:**

In single precision, a floating-point number is represented using 32 bits, divided into three parts:

- **1 bit for the sign (S)**
- **8 bits for the exponent (E)**
- **23 bits for the fraction (F) or mantissa**

1	8	23
S	E	M

### **2. Double Precision (64-bit) Format:**

In double precision, a floating-point number is represented using 64 bits, divided into three parts:

- **1 bit for the sign (S)**
- **11 bits for the exponent (E)**
- **52 bits for the fraction (F) or mantissa.**

1	11	52
S	E	M

## **2. Write the restoring division process with flowchart and example.(COPY)**

\*\* Explain the function of BIU & EU.

### **>>1. Bus Interface Unit (BIU)**

The **Bus Interface Unit (BIU)** is responsible for handling all operations related to accessing memory and I/O devices. It manages the data, address, and control buses, ensuring smooth communication between the microprocessor and other components of the computer system.

### **2. Execution Unit (EU)**

The **Execution Unit (EU)** is responsible for executing instructions. It processes the instructions that are fetched from memory and carried out by the BIU. The EU performs all the arithmetic and logical operations needed for program execution.

## **3. What do you mean by hit and miss ratio?**

## >>Hit Ratio

The **hit ratio** refers to the percentage of memory accesses that are **served by the cache**. A high **hit ratio** indicates that the cache is effectively storing the data that the processor needs, leading to faster data access and improved performance.

## Miss Ratio

The **miss ratio** is the complement of the hit ratio. It refers to the percentage of memory accesses that result in a **cache miss**, meaning that the requested data was not found in the cache, and the system had to access the slower main memory to retrieve it.

\*\*\*\*A typical computer system cache memory access time is 8 ns and its main memory access time is 65.ns. If hit ratio is 90%, what is the average memory access time?\*\*\*

>>The **average memory access time (AMAT)** is a measure of the time it takes to access data from memory, considering both the cache and the main memory. It depends on the **hit ratio**, **cache access time**, and **main memory access time**.

The formula to calculate the **average memory access time** is:

$$AMAT = (Hit\ Ratio \times Cache\ Access\ Time) + (Miss\ Ratio \times (Cache\ Access\ Time + Main\ Memory\ Access\ Time))$$

Where:

- **Hit Ratio** = The percentage of memory accesses that are served by the cache.
- **Miss Ratio** = 1 - Hit Ratio (the percentage of memory accesses that result in a cache miss, requiring data to be fetched from main memory).
- **Cache Access Time** = The time taken to access data from the cache.
- **Main Memory Access Time** = The time taken to access data from main memory.

**Given Data:**

- **Cache Access Time** = 8 ns
- **Main Memory Access Time** = 65 ns
- **Hit Ratio** = 90% = 0.90

- **Miss Ratio = 1 - 0.90 = 0.10**

**Now, let's calculate the AMAT:**

$$\begin{aligned} \text{AMAT} &= (0.90 \times 8) + (0.10 \times (8+65)) \\ \text{AMAT} &= (0.90 \times 8) + (0.10 \times 73) \\ \text{AMAT} &= 7.2 + 7.3 = 14.5 \text{ ns} \end{aligned}$$

**Conclusion:**

The **average memory access time (AMAT)** is **14.5 ns**.

**4.** What do you mean by cachememory mapping?

>>Cache mapping is the **procedure in to decide in which cache line the main memory block will be mapped**.

\*\* Write down the direct cache memory mapping technique and what is its main disadvantage.

>>**Main Disadvantage of Direct-Mapped Cache**

The **main disadvantage** of **direct-mapped cache** is the occurrence of **cache conflicts** or **cache thrashing**. Specifically:

- **Cache Conflicts:**
  - Since each block of memory can only be placed in one specific cache line, **multiple memory blocks** may map to the **same cache line** if their index bits are the same. This leads to **cache misses** when these blocks are frequently accessed, resulting in inefficient cache usage.
  - For example, if two frequently accessed blocks (one from memory address 0x0000 and another from 0x1000) both map to the same cache line (due to the same index bits), every time the CPU accesses one of these blocks, the other will be evicted, leading to a high number of cache misses.
- **Limited Flexibility:**
  - The lack of flexibility in where data can be placed in the cache means that it doesn't make the best use of cache space. The cache may frequently need to replace blocks that are still needed because of mapping limitations.

**Steps in Direct-Mapped Cache Access:**

1. **Divide the memory address:** The memory address is split into three fields:

- **Tag bits:** Identify the memory block.
  - **Index bits:** Indicate which cache line the data should go to.
  - **Block offset bits:** Specify the byte or word within the block.
2. **Cache Lookup:** When the CPU accesses a memory address, the cache checks:
- The **index bits** to find the corresponding cache line.
  - The **tag bits** to check if the data stored in that cache line matches the requested block (a cache hit if it matches).
3. **Cache Hit or Miss:**
- **Cache Hit:** If the data is found in the cache (the tag matches the requested data), it is returned from the cache.
  - **Cache Miss:** If the data is not found (the tag does not match), the cache misses, and the data must be fetched from main memory.
4. **Replacement:** When a miss occurs, the new data is loaded into the corresponding cache line, replacing the existing data (if any).

## 5. Write the differences between RISC and CISC architecture.

>>

Aspect	RISC	CISC
<b>Instruction Set</b>	Small, simple set of instructions	Large, complex set of instructions
<b>Instruction Length</b>	Fixed length (usually 32 bits)	Variable length (1 to 15 bytes)
<b>Execution Time</b>	Most instructions execute in one cycle	Instructions may take multiple cycles
<b>Registers</b>	Many registers (e.g., 32 or more)	Fewer registers (e.g., 8 or 16)
<b>Memory Access</b>	Memory access through load/store only	Memory operands can be directly used

\*\*Discuss different types of addressing modes with examples.

>>**Addressing modes** define how the effective address of an operand (the memory location or register where data is stored) is calculated in an instruction. These modes allow a processor to be flexible in accessing data from different places, such as registers, memory, or direct values.

### **1. Immediate Addressing Mode:-**

In this mode, the operand is directly specified in the instruction. The value is provided as a constant in the instruction itself.

**Example:** The instruction ADD R1, R2, #10 adds the value **10** to the contents of **R2** and stores the result in **R1**.

### **2. Register Addressing Mode:-**

In register addressing, the operand is located in a register. The instruction specifies the register where the operand is stored.

**Example:** The instruction ADD R1, R2 adds the contents of **R2** to **R1**.

### **3. Direct Addressing Mode:-**

In direct addressing mode, the effective address of the operand is given explicitly in the instruction. The instruction provides the memory address where the data is stored.

**Example:** The instruction LOAD R1, 2000 loads the value stored at memory address **2000** into register **R1**.

## **6. Write the different pipeline hazards.**

### **>>The Three Main Types of Pipeline Hazards:**

- 1. Data Hazards**
- 2. Control Hazards**
- 3. Structural Hazards**

#### **1. Data Hazards**

A **data hazard** occurs when an instruction depends on the result of a previous instruction that has not yet completed its execution.

#### **2. Control Hazards**

A **control hazard** (or **branch hazard**) occurs when the pipeline makes decisions based on a branch instruction (e.g., **if, else, jump**) but does not know the correct instruction to fetch next until the branch decision is made.

#### **3. Structural Hazards**

A **structural hazard** occurs when the hardware resources required by an instruction are not available at the time it needs them. This can happen when multiple instructions require the same resource (e.g., ALU, memory, or registers) at the same time.

**\*\*Explain the different groups of computers according to Flyin's classification.(copy)**

## 7.Explain maximum and minimum mode of 8086 microprocessor.

### >>1. Minimum Mode

- **Minimum Mode** is the mode of operation in which the 8086 works as a standalone processor with minimal external components. In this mode, the 8086 controls all operations, including memory access and I/O operations, with minimal involvement of external control circuitry.
- **Pins Involved:** The **MN/MX** pin (pin 33) is used to select the mode. If the **MN/MX** pin is grounded (low), the microprocessor operates in **Minimum Mode**.

### 2. Maximum Mode

- **Maximum Mode** is used when the 8086 microprocessor works in a multiprocessor system or when it requires external support for its control signals. In this mode, the 8086 relies on an external 8288 Bus Controller to manage the bus and the control signals.
- **Pins Involved:** The **MN/MX** pin is set high (1) to select Maximum Mode.

## 8. Write short notes on any two (a) TLB (b) DMA (c) Virtual Memory.

### >>a) **TLB (Translation Lookaside Buffer)**

A **Translation Lookaside Buffer (TLB)** is a type of cache used in computer systems to improve the speed of virtual-to-physical address translation. It stores recent translations of virtual memory addresses to physical addresses in the system, allowing faster access to memory and reducing the need to consult the slower page table in main memory.

### b) **DMA (Direct Memory Access)**

**Direct Memory Access (DMA)** is a feature that allows peripherals (such as hard drives, network cards, or sound cards) to directly transfer data to or from the computer's memory without involving the CPU. This process is used to offload data transfer tasks from the CPU, freeing it up to perform other operations while the data transfer occurs in the background.

### c) **Virtual Memory**

**Virtual Memory** is a memory management technique that provides an "idealized abstraction" of the storage resources that are actually available on a given machine, which creates the illusion to users of a very large (main) memory. It allows programs to access more memory than is physically available in RAM, by using disk storage as an extension of the system's physical memory.

## 9. What do you mean by micro operation?

### >>**Micro Operation**

A **micro operation** (or **micro-op**) is a fundamental operation performed at the hardware level in a computer's control unit during the execution of an instruction. These operations are the basic steps required to carry out a machine instruction.

**\*\*Explain the functioning procedure of micro-programmed control unit.**

#### **>>② Fetch the Machine Instruction:**

- The micro-programmed control unit first fetches a machine instruction from memory (the instruction that the CPU needs to execute). This machine instruction is then decoded by the control unit to understand which operation is to be performed (e.g., an addition, a data transfer, etc.).

#### **② Fetch the Microinstruction:**

- After the machine instruction is decoded, the control unit uses the instruction's opcode (operation code) to generate an address in the **Control Memory**. This address points to the first **microinstruction** associated with the machine instruction.
- The **Control Address Register (CAR)** is loaded with the address of the microinstruction, and the control unit fetches this microinstruction from control memory.

#### **② Execute the Microinstruction:**

- The fetched **microinstruction** is placed in the **Control Data Register (CDR)**. This microinstruction consists of several fields that control various parts of the processor:
  - **Control signals:** These activate or deactivate components of the CPU like registers, ALU, buses, etc.
  - **Next address:** It may also specify the address of the next microinstruction to be fetched, determining the flow of execution.
  - The microinstruction instructs the control unit to perform a specific operation (e.g., transferring data between registers, performing arithmetic operations, etc.).

