4.2 Computer arithmetic and memory system:

Arithmetic and Logical operation

Arithmetic and logical operations are the fundamental building blocks of computational processes in a CPU. Arithmetic operations include addition, subtraction, multiplication, and division, which are performed using specialized hardware units like adders and multipliers. These operations can be extended to support floating-point calculations, enabling high-precision mathematical computations. Logical operations, including AND, OR, XOR, and NOT, are used for bitwise manipulation and decision-making processes. These operations are critical for implementing algorithms, performing data transformations, and supporting complex problem-solving in various applications. The CPU uses algorithms such as Booth's algorithm for efficient multiplication and the non-restoring division method for division. For floating-point arithmetic, the IEEE 754 standard is commonly used, defining rules for representation, addition, subtraction, multiplication, and division of floating-point numbers. Logical operations are essential for tasks like mask generation, error checking, and data encryption.

The Memory Hierarchy

The memory hierarchy is a structured arrangement of memory types designed to balance speed, cost, and capacity. At the top of the hierarchy are registers, which provide the fastest access but have limited capacity. Next is cache memory, followed by main memory (RAM) and secondary storage (hard drives, SSDs). The hierarchy continues with tertiary storage devices like optical discs and network storage. Each level in the hierarchy is characterized by increasing access time and storage capacity but decreasing cost per bit. The goal of the memory hierarchy is to provide the CPU with data at the fastest possible rate while minimizing the overall cost of the system. Properly designed memory hierarchies significantly enhance system performance by ensuring that frequently accessed data resides in faster memory levels.

Internal and External memory

Internal memory refers to storage located directly within the computing system and accessible by the CPU without requiring additional input/output operations. Examples include registers, cache, and RAM. These memories are critical for high-speed data processing and temporary storage during program execution. External memory, on the other hand, includes storage devices such as hard drives, SSDs, and external USB drives. These are used for long-term data storage and backup. External memory is slower than internal memory but offers significantly larger storage capacities. The integration of external memory into computing systems involves the use of file systems and input/output control mechanisms.

Cache memory principles

Cache memory is a small, high-speed memory located close to the CPU that stores frequently accessed data and instructions. The principle of locality of reference underpins the effectiveness of cache memory. Temporal locality suggests that recently accessed data is likely to be accessed again soon, while spatial locality indicates that data located near recently accessed addresses is also likely to be accessed. Cache memory reduces the latency of data retrieval by serving as an intermediate storage layer between the CPU and main memory.

Elements of Cache design:

Cache size

The size of the cache determines how much data it can hold. Larger caches can store more data but may result in increased access times and higher costs. The optimal cache size balances capacity, speed, and cost to maximize system performance.

Mapping function

The mapping function defines how data from main memory is placed into the cache. Common mapping techniques include direct mapping, where each memory block maps to a specific cache line; associative mapping, which allows a block to occupy any cache line; and set-associative mapping, which is a hybrid of the two. Each mapping technique has trade-offs in terms of complexity, speed, and hit ratio.

Replacement algorithm

When the cache is full, a replacement algorithm determines which data to evict to make space for new data. Common algorithms include Least Recently Used (LRU), which removes the least recently accessed data; First-In-First-Out (FIFO), which removes the oldest data; and Random Replacement, which evicts a randomly chosen block. The choice of algorithm affects cache efficiency and system performance.

Write policy

Write policies dictate how data modifications are handled in the cache and main memory. Write-through policy updates both the cache and main memory simultaneously, ensuring consistency but increasing latency. Write-back policy updates the main memory only when the cached data is replaced, reducing write operations but requiring additional control mechanisms to maintain consistency.

Number of caches

Modern CPUs often include multiple levels of cache (L1, L2, and sometimes L3) to optimize performance. L1 cache is the fastest but smallest, located closest to the CPU. L2 and L3 caches are larger and slower but still significantly faster than main memory. The hierarchy of caches helps balance access speed and storage capacity.

Memory Write Ability and Storage Permanence

Memory write ability refers to the ease with which data can be written to memory. Volatile memories like RAM allow fast read and write operations but lose data when power is removed. Non-volatile memories like ROM and flash memory retain data without power but often have slower write speeds. Storage permanence is essential for applications requiring long-term data retention, such as firmware storage or archival systems.

Composing Memory

Composing memory involves integrating different types of memory to achieve a balance of speed, capacity, and cost. This includes designing memory hierarchies and optimizing the interaction between volatile and non-volatile storage. Techniques like memory interleaving, which splits memory into modules accessed simultaneously, further enhance system performance by parallelizing data access.