Exam Topic:

To prepare for IAS computer architecture with the listed topics, follow this structured approach categorized by themes:

1. Basics of Computer Architecture and Performance

IAS Computer (Structure and functioning)

CPI (Cycles Per Instruction)

MIPS (Million Instructions Per Second)

2. Performance Improvement Techniques

Speedup Techniques

Cache (Levels, speed improvement, organization)

Pipelining

Superscalar Architecture

3. Processor Architecture

ARM-6 Microprocessor Diagram

Processor Basics

4. Data Representation

Magnitude and Complement Representation

Signed Magnitude

1's Complement

2's Complement

Floating Point Representation

5. Datapath and Arithmetic Logic Unit (ALU)

Data Path Representation

Block diagrams for datapath

Arithmetic Unit

Basic design

Overflow detection

6. Memory Organization

DRAM and SRAM Characteristics

Memory Retention

2D RAM Addressing Scheme (e.g., topics 6.11, 6.12)

Rambus DRAM

Cache and Virtual Memory

Contiguous and Non-Contiguous Allocation

Address Translation (refer to Gate Smashers for deeper understanding)

Paging and Segmentation

Memory Allocation and Mapping

2, 4, and 8-way Cache

Virtual Memory Concepts

One-Hot Method

Classical Methods

7. System Organization

System Bus

Interrupts (Hardware and Software)

Hardware Design

Arithmetic Logic (ALU)

CPU Performance Metrics

Microprocessor Design

8. Additional Exam Preparation

Solve GATE 2016 Computer Architecture Questions

Use GCO Control to organize studies around:

CPU control unit operations

Memory organization and performance

System bus and interrupts

An **interrupt** is a signal sent to the processor to **temporarily stop** the current execution and handle an important task (like input from a keyboard or an error). After handling the task, the CPU resumes its previous work.

Methods to Solve Interrupt Issues:

**1. Interrupt Prioritization**

If multiple interrupts occur, the system assigns priority levels to handle the most important first.

Example: A hard disk failure has higher priority than a mouse click.

**2. Disabling Interrupts Temporarily**

Some critical tasks disable lower-priority interrupts to prevent disturbance.

Example: During an important calculation, the CPU can ignore low-priority interrupts.

An Interrupt Service Routine (**ISR**) is a special function or program that is called by CPU when an interrupt occurs. It contains the instructions needed to handle the interrupt and ensure the system continues running smoothly.

### ****Functions of ISR:****

1. **Handles Interrupts:** Executes the required task when an interrupt occurs (e.g., processing keyboard input, handling hardware signals).
2. **Saves and Restores CPU State:** Stores register values before execution and restores them after ISR execution.
3. **Clears Interrupt Flags:** Ensures the interrupt does not repeatedly trigger.
4. **Prioritizes Multiple Interrupts:** If multiple interrupts occur, the ISR follows priority rules to execute the most important one first.

A **bus bottleneck** occurs when multiple devices try to communicate over a shared bus, causing **slowdowns and reduced performance**.

Bus bottlenecks happen when too many devices compete for data transfer on a shared bus, causing slowdowns and delays in communication.

A **mezzanine architecture** in bus systems refers to a hierarchical design where a high-speed bus is placed between the processor bus and lower-speed peripheral buses. It improves system performance by reducing bottlenecks and efficiently handling high-speed I/O operations.

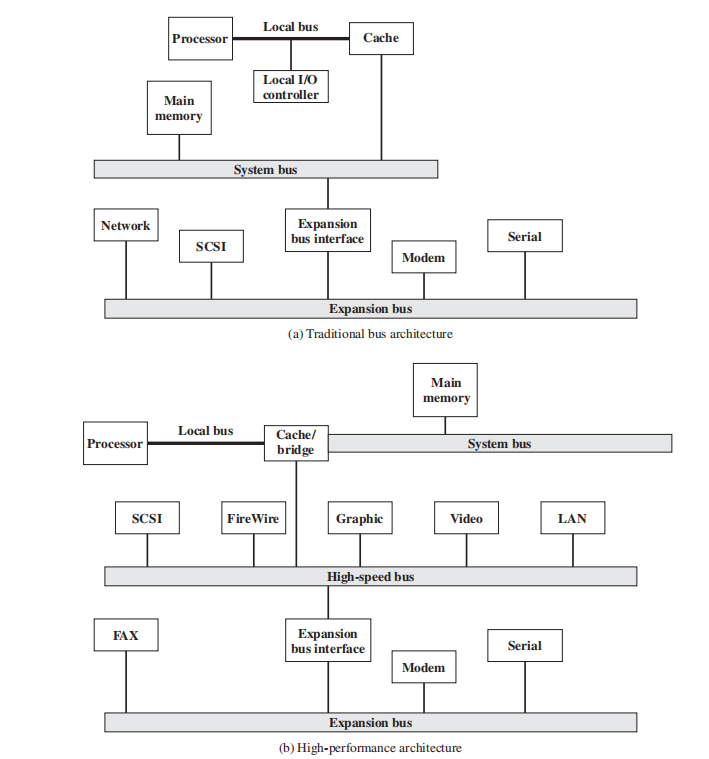
### ****Vector Interrupt** refers to an interrupt where the address of the interrupt service routine (ISR) is determined by the interrupt vector.**

****PCI Interrupt** refers to interrupts generated by Peripheral Component Interconnect (PCI) devices in a computer system.**

### ****Pipeline Interrupt** refers to interrupts that occur in a pipelined CPU architectu**re.****

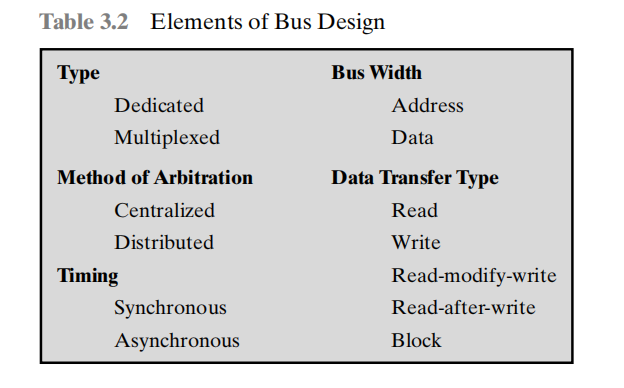
### ****Main Properties:****

1. **High-Speed Intermediate Bus** – Placed between the processor and peripheral buses.
2. **Bus Bridging** – Connects different bus levels using a bridge.
3. **Performance Enhancement** – Reduces congestion on the main processor bus.
4. **Scalability** – Allows easy expansion of high-speed I/O devices.
5. **Efficiency** – Optimizes communication between components.



The peripheral component interconnect (PCI) is a popular high-bandwidth,

processor-independent bus that can function as a mezzanine or peripheral bus.



Bus arbitration is the process of managing multiple devices requesting access to a shared bus. Since only one device can use the bus at a time, an arbitration mechanism is required to resolve conflicts and decide which device gets control.

This table categorizes key elements of bus design in computer architecture:

Type

Dedicated: Separate lines for address and data.

Multiplexed: Shared lines for address and data, reducing hardware but increasing control complexity.

Bus Width

Address: Determines memory size that can be accessed.

Data: Defines how much data can be transferred simultaneously.

Method of Arbitration (Decides which device gets bus access)

Centralized: A single controller manages access.

Distributed: Devices communicate and decide access among themselves.

Timing (Synchronization of data transfer)

Synchronous: Uses a clock signal for coordination.

Asynchronous: Uses handshaking signals, not dependent on a clock.

Data Transfer Type (Operations performed on the bus)

Read/Write: Basic memory operations.

Read-Modify-Write: Reads, updates, and writes back data.

Read-after-Write: Ensures a write operation completes before reading.

Block: Transfers multiple words in one cycle, improving efficiency.

An ALU (Arithmetic Logic Unit) is a critical component in a computer's central processing unit (CPU) responsible for performing arithmetic and logical operations.

**Integer representation** in computer systems refers to how integer values (whole numbers) are stored and represented in binary form within a computer's memory.

**Unsigned Integer Representation**:

* Represents only non-negative integers
* The representation is straightforward because there’s no need to handle the sign of the number. The entire set of bits is used to represent the magnitude so more number can be represented.
* Since all bits are used to represent positive values, an unsigned integer can represent a larger range of positive values compared to a signed integer of the same bit size.
* Unsigned integers cannot represent negative values, so they are only useful for counting quantities that are always positive.

**Signed Integer Representation**

* Signed integers can represent both positive and negative numbers.
* The sign bit (most significant bit) clearly indicates the positivity or negativity of the number, which can be intuitive and easy to understand.
* The main drawback of sign-and-magnitude representation is that it has two different representations for zero.
* Arithmetic operations like addition and subtraction are more complicated than with other signed representations. For example, to add or subtract numbers, the sign bit needs special handling to ensure the correct result.

a) **Sign-and-Magnitude Representation**:

The most significant bit (MSB) is used to represent the sign (0 for positive, 1 for negative), and the remaining bits represent the magnitude (absolute value) of the number.

b) **One’s Complement Representation**:

Negative numbers are represented by inverting all the bits of their positive counterpart (bitwise NOT)

C)**Two’s Complement Representation**:

* Negative numbers are represented by inverting all the bits of the number and adding 1 to the result. This eliminates the issue of negative zero.
* One of the biggest advantages is that arithmetic operations, such as addition and subtraction, can be performed without special handling for the sign.
* The sign is implicitly handled by the two's complement operation. You don’t need separate logic to deal with positive and negative numbers operation in arithmetic.

****Fixed-Point Representation**:**

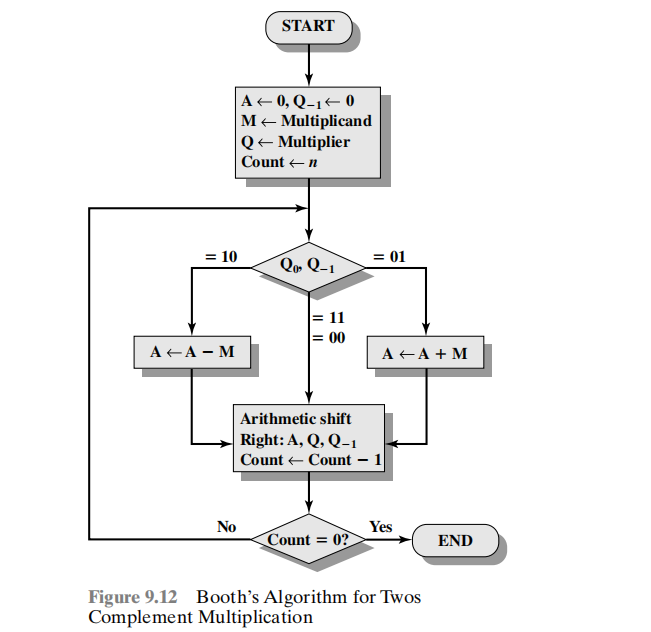
* Used to represent real numbers (with fractions) by allocating a fixed number of bits for the integer part and the fractional part.
* **Fixed-point representation** is a way of storing real (non-integer) numbers in binary format where a fixed number of bits are reserved for the integer part and a fixed number of bits are reserved for the fractional part.
* Fixed-point arithmetic (addition, subtraction, multiplication) can be implemented much more efficiently than floating-point arithmetic, as it doesn't require complex floating-point operations or hardware.
* Fixed-point representation has a **limited range** of values because both the integer and fractional parts are fixed in size.It has overflow, large scale number representation issue.

**Addition and Subtraction** follow general rules for 2’s complement number represented. Overflow should manage.

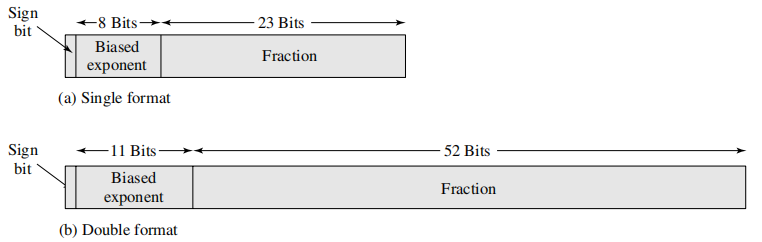
**OVERFLOW RULE:** If two numbers are added, and they are both positive or

both negative, then overflow occurs if and only if the result has the opposite sign.

**Booth's algorithm** is a powerful and efficient technique for binary multiplication, especially for signed integers in two's complement form. It reduces the number of operations needed for multiplication by exploiting repeated patterns in the multiplier, making it ideal for hardware implementations. It is the useful for high-speed multiplication in ALUs.



**IEEE Standard for Binary Floating-Point Representation**

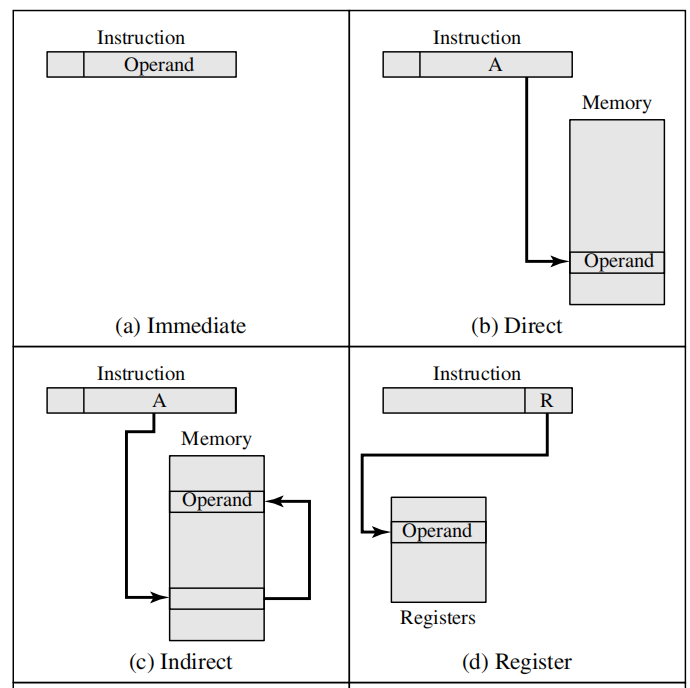
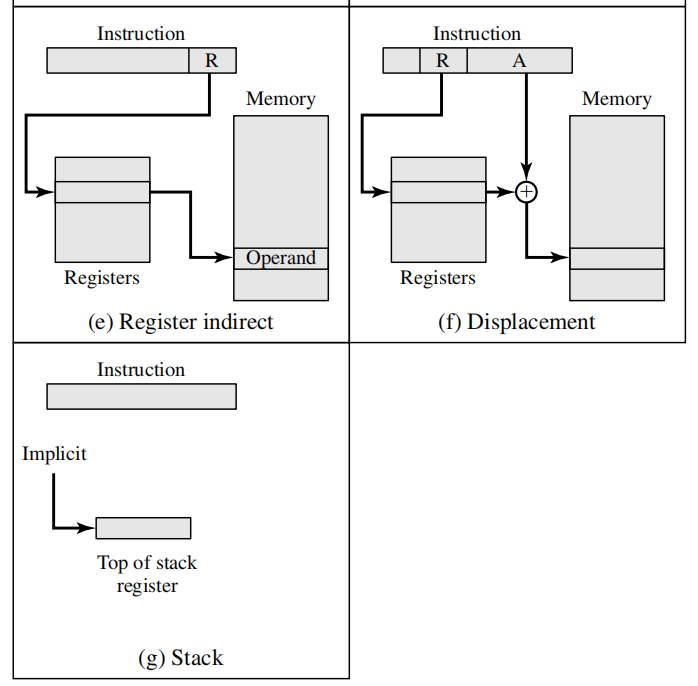


A normalized number in IEEE 754 floating-point representation is a number where the leading digit of the significand (mantissa) is always 1. This format ensures maximum precision for a given number of bits.

A denormalized (or subnormal) number is used to represent very small numbers that would otherwise underflow (become too small to store in normalized form).

When the exponent is all zeros, the number is stored without the implicit leading 1 in the significand.

Addressing refers to the method used to identify and access data stored in memory or registers. Addressing determines how the CPU locates operands (data) and instructions in memory during execution.



Displacement addressing is a type of addressing mode where the effective address (EA) is calculated by adding a constant displacement (offset) to a base value stored in a register.

* Register Relative Addressing, The displacement is added to a register value.
* Base-Register Addressing, A base register holds the starting address, and displacement provides an offset.
* Indexed Addressing, The displacement is stored in an index register, commonly used for array access.

**Stack addressing** is a method where operands are accessed using a stack, a Last-In, First-Out (LIFO) data structure. The Stack Pointer (SP) register keeps track of the top of the stack.

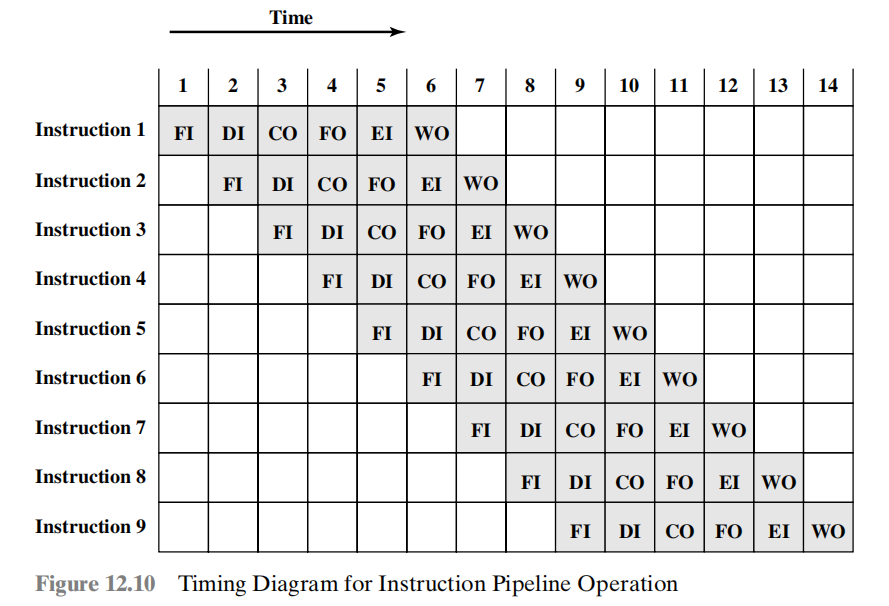
**Instruction Format** refers to the layout of bits in an instruction that defines how the CPU should interpret and execute it. It specifies the structure of an instruction, including the opcode, operands, and addressing modes.

· **Opcode Field** – Specifies the operation to be performed (e.g., ADD, SUB, LOAD).

· **Operand Field** – Contains register or memory addresses for data.

· **Addressing Mode Field** – Defines how to interpret operand addresses (immediate, direct, indirect, etc.).

· **Instruction Length** – Fixed or variable-length instructions (RISC vs. CISC).

**Instruction Pipelining** is a technique used in **CPU architecture** to improve performance by overlapping multiple instruction execution stages. Instead of executing instructions one by one, the CPU breaks them into smaller stages and processes multiple instructions **simultaneously** in different stages.

**Cache** memory is a small, high-speed storage located between the processor and main memory (RAM) that stores frequently accessed data and instructions. It reduces access time and enhances CPU performance by minimizing delays in fetching data from slower main memory.

**Random Access Memory (RAM)** is a type of volatile memory that stores data and instructions temporarily while a computer is running. It provides fast read and write access to the processor, enabling efficient execution of applications and processes.

Any memory location can be accessed directly and independently, without needing to go through other data and use direct addressing process.

**Memory retention** refers to the ability of a storage device or memory component to retain data over time, either temporarily (volatile memory) or permanently (non-volatile memory).

**Destructive Read-Out (DRO)** refers to a type of memory operation where reading the stored data **erases or alters** it, requiring a **refresh or rewrite** of the original data after every read operation.

· In **DRAM (Dynamic Random Access Memory)**, each memory cell stores data as an electric charge in a capacitor.

· When the data is read, the capacitor's charge is **drained**, effectively erasing the data.

· To maintain data integrity, DRAM requires **periodic refresh cycles** to rewrite the read values back into the memory cells.

**Address Translation** is the process of converting a **virtual address (VA)** used by a program into a **physical address (PA)** in memory.

**Base Addressing** is a memory addressing technique where a **base address** (starting memory location) is used as a reference point, and an **offset (displacement)** is added to access specific memory locations. This method is widely used in **segmentation, indexed addressing, and dynamic memory allocation**.

The **Translation Lookaside Buffer (TLB)** is a **high-speed cache** inside the **Memory Management Unit (MMU)** that stores recently used virtual-to-physical address mappings. It helps speed up **address translation** by reducing the need to access the **page table** in RAM.

**Segment-based memory mapping** scheme, the memory is divided into different segments, such as code, data, stack, and heap. Each segment is a contiguous block of memory with a logical purpose.

**Page-based memory mapping** scheme, memory is divided into small fixed-size blocks called **pages**. The virtual memory is divided into pages, and the physical memory is divided into page frames. The operating system uses a **page table** to map virtual pages to physical page frames.

**Memory Allocation** is the process of assigning portions of memory to different programs, processes, or data structures in a computer system.

· **Primitive Memory Allocation** is used for simple, fixed-size data types, and is often handled automatically by the system.

· **Non-Primitive Memory Allocation** is used for complex data structures that can vary in size and require dynamic allocation and manual memory management

Cache memory mapping refers to the method by which data from main memory is stored in the cache.

**Direct-Mapped Cache** is the simplest cache organization method, where each block of memory is mapped to exactly one cache line.

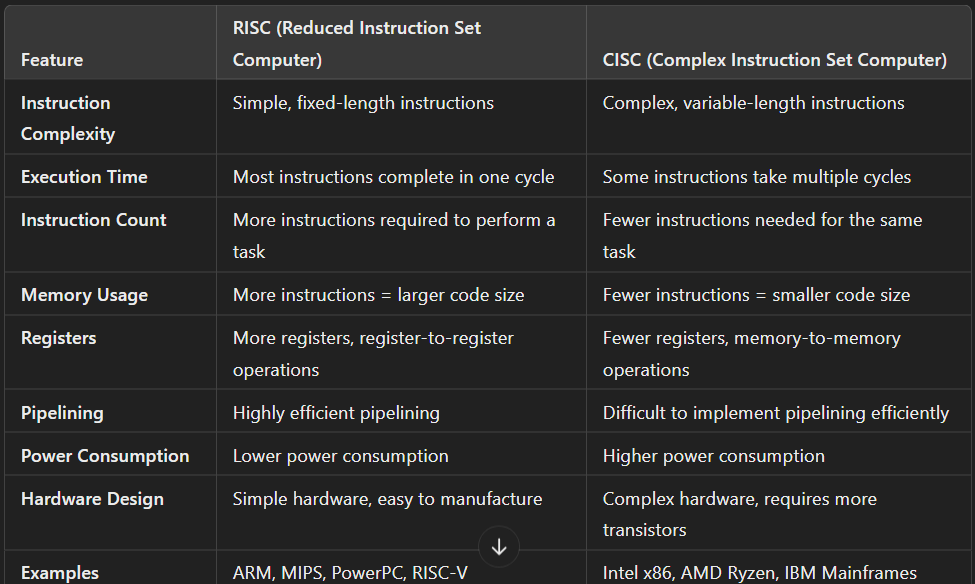
**Associative Cache** is any block of data can be stored in any cache line. The memory address doesn't restrict a block of memory to a specific cache line, and it can be placed in any available cache slot.

**Set-Associative Cach**e is a compromise between direct-mapped and fully associative caches. In set-associative mapping, the cache is divided into several sets, and each block of memory can be mapped to one specific set but can occupy any cache line within that set.

Paging is a memory management scheme that divides physical memory (RAM) into fixed-size blocks called frames, and divides the logical address space (used by programs) into blocks of the same size called pages.

A coprocessor is a specialized processor that works alongside the main processor to perform specific tasks more efficiently.

Coprocessors are typically used to accelerate particular types of computations, such as floating-point arithmetic, graphics processing, signal processing, cryptography, or other domain-specific tasks.



ALU Expansion :

**Spatial Expansion** refers to increasing the number of processing or memory units to handle more data simultaneously. This improves parallelism and enhances performance by utilizing additional resources.

**Temporal Expansion** refers to optimizing execution over time by improving processing techniques, pipelining, and instruction execution flow. It focuses on reducing latency and increasing the speed of computation.

A **Combinational ALU** performs operations **in a single clock cycle** without requiring any sequential steps or memory storage. The output **depends only on the current inputs** and is generated immediately based on combinational logic circuits.

### ****Key Characteristics:****

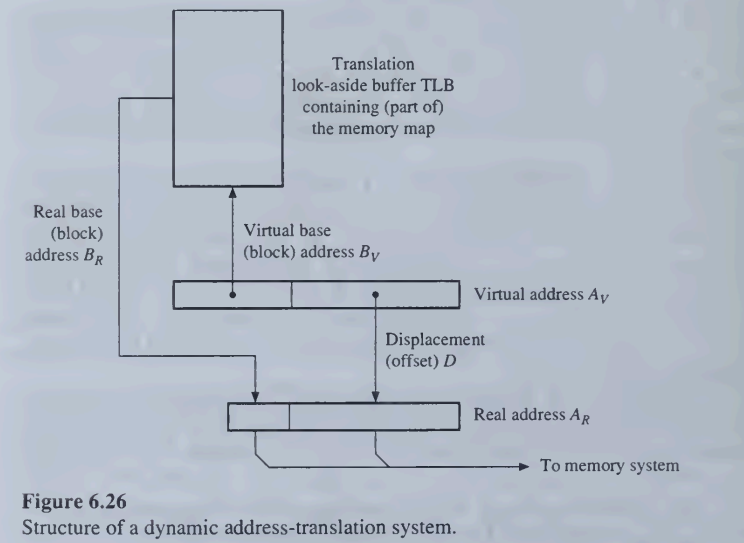
✔ **No Storage Elements**: Does not use flip-flops or registers to store intermediate results.  
✔ **Faster Execution**: Output is available as soon as inputs are applied.  
✔ **Purely Combinational Logic**: Uses logic gates (AND, OR, XOR, ADDER circuits) to process operations.  
✔ **No Clock Dependency**: Does not require clock cycles to complete execution.

A **Sequential ALU** performs operations **step by step over multiple clock cycles**. It uses **registers, flip-flops, and control logic** to store intermediate results and process data sequentially.

### ****Key Characteristics:****

✔ **Uses Storage Elements**: Has registers and memory to store intermediate data.  
✔ **Clock-Dependent**: Operations are executed across multiple clock cycles.  
✔ **Handles Complex Operations**: Can perform multi-step computations like division, multiplication, and floating-point arithmetic.  
✔ **More Control Logic**: Requires a control unit to manage sequential execution.

A Carry-Lookahead Adder (CLA) is a high-speed adder designed to overcome the carry propagation delay in Ripple Carry Adders (RCA). Instead of waiting for the carry to propagate through each bit position sequentially, the CLA predicts carry values in advance, significantly improving addition speed.



Accumulator-Based CPU is a type of processor architecture where one special register, called the Accumulator (ACC), is primarily used for arithmetic and logical operations.

### ****Advantages of Accumulator-Based CPU****

✅ **Smaller instruction size** (no need to specify multiple registers).  
✅ **Simpler hardware design**, reducing complexity and cost.  
✅ **Efficient for simple arithmetic operations**.  
✅ **Lower power consumption**, making it ideal for embedded systems.  
✅ **Use Single accumulator register.**

