Computer Architecture and Organization

CS 115

Lecture 6

Instructor: **Gerald John M. Sotto**

Last Updated: November 18, 2025

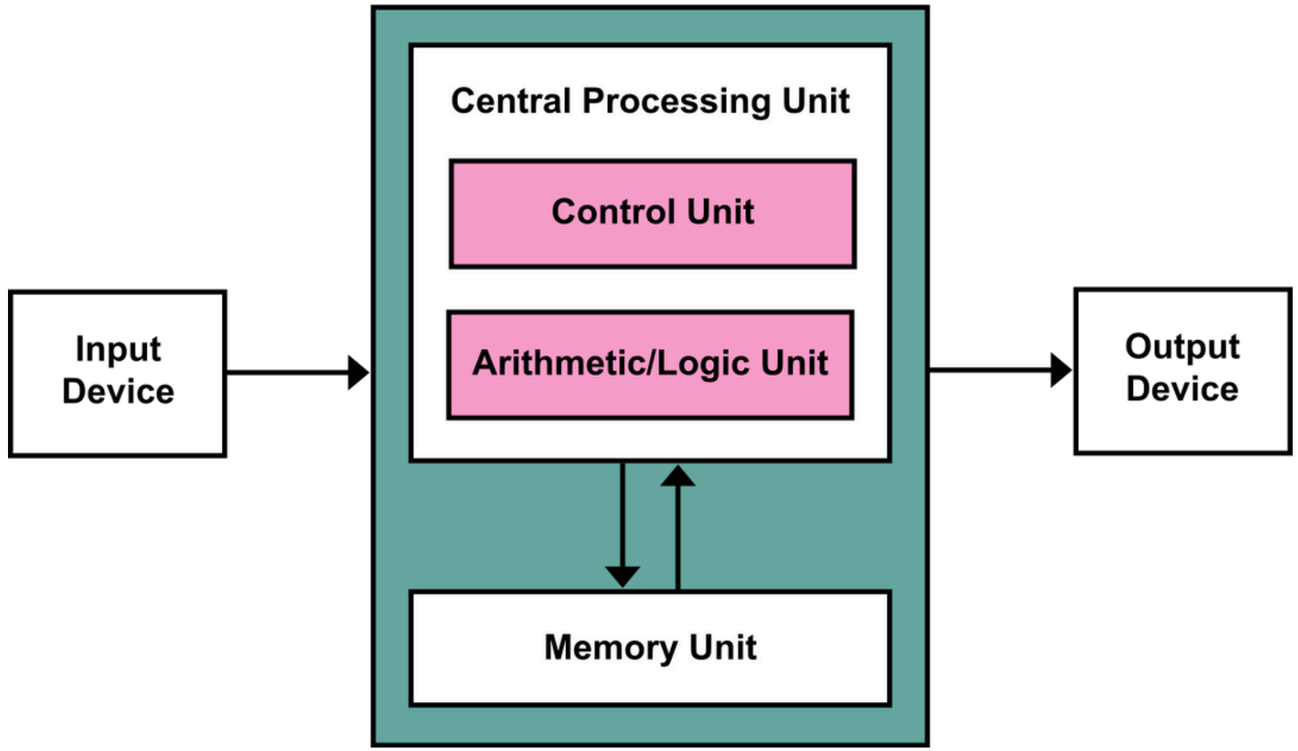
Table of Contents

Unit IV: Assembly Level Machine and Functional Organization

Introduction to MARIE

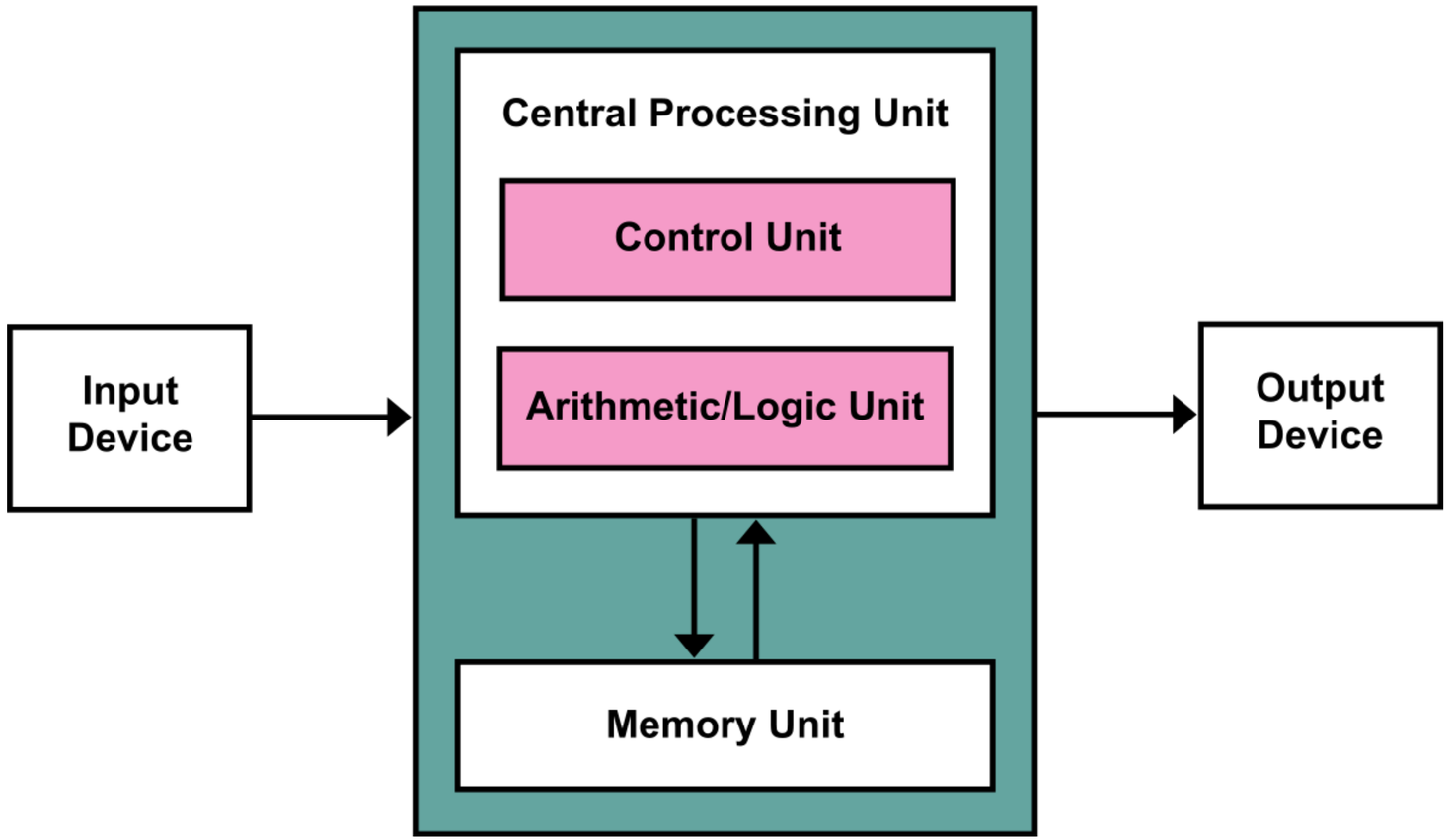
Von Neumann Foundation

- Cornerstone of modern computing
- This architecture is fundamentally defined by the **stored-program concept**
- It forms the basis for virtually all modern general-purpose computers, from smartphones to supercomputers.



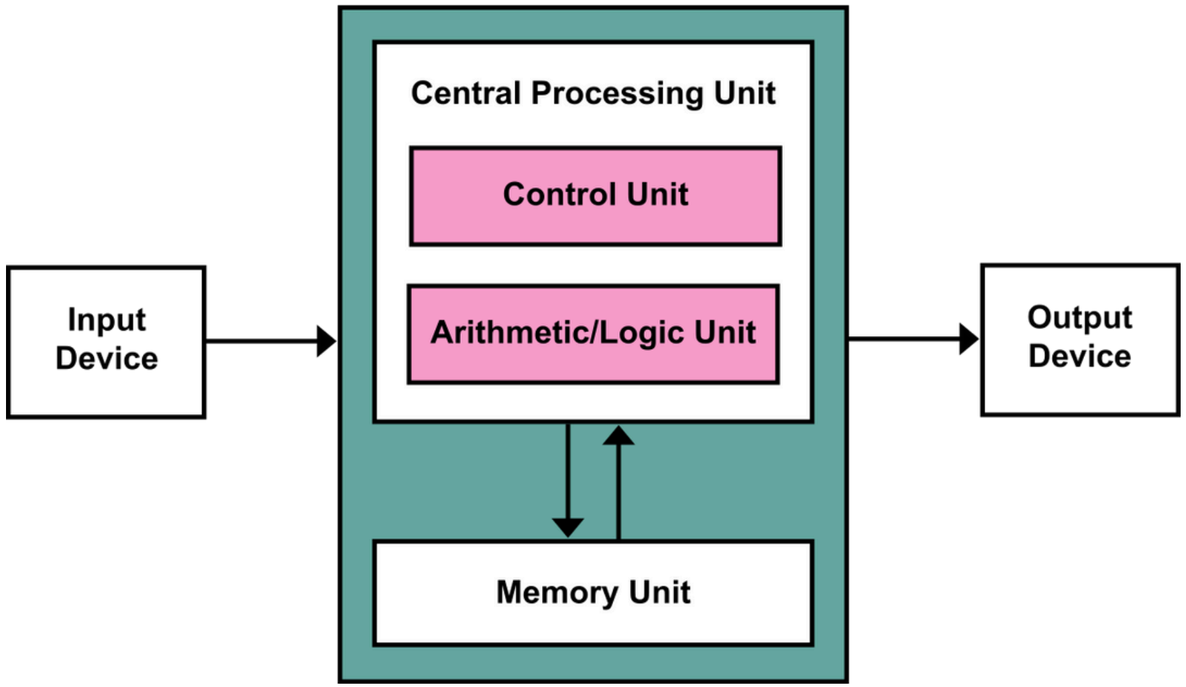
Von Neumann Foundation

- Revolutionary innovation credited to **John von Neumann** in the early 1960s, where program instructions and data are stored together in a **single**, **unified memory unit.**



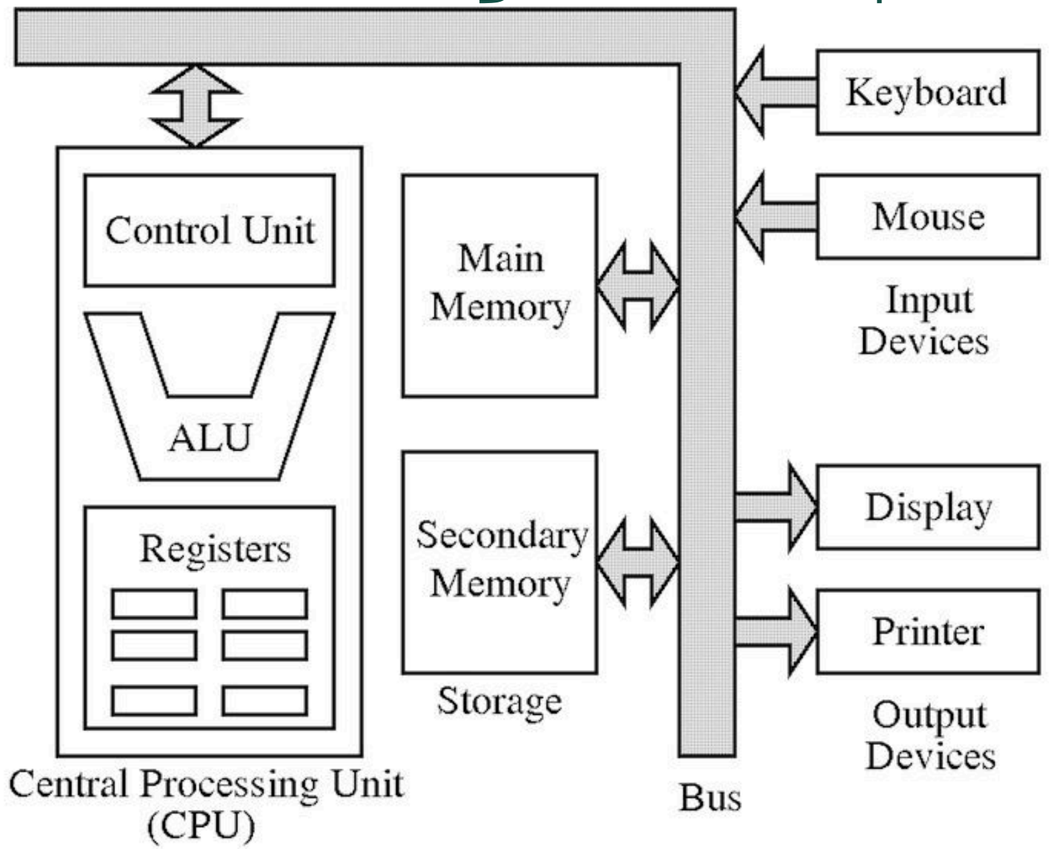
Von Neumann Foundation

- The **bidirectional arrow** connecting the CPU and the Memory Unit represents the **shared communication pathway** (the bus). This single pathway, while simple and flexible, is the source of the well-known **Von Neumann Bottleneck**, which limits the rate at which data can be transferred between the fast CPU and the memory, often constraining overall system performance



Von Neumann Foundation

- Crucially, the von Neumann design processes instructions through a sequential **fetch–decode–execute cycle**, h**andling one instruction at a time.** This sequential model, despite its age, remains highly relevant and has dramatically influenced the design of current processors.



MARIE Architecture and Memory Organization

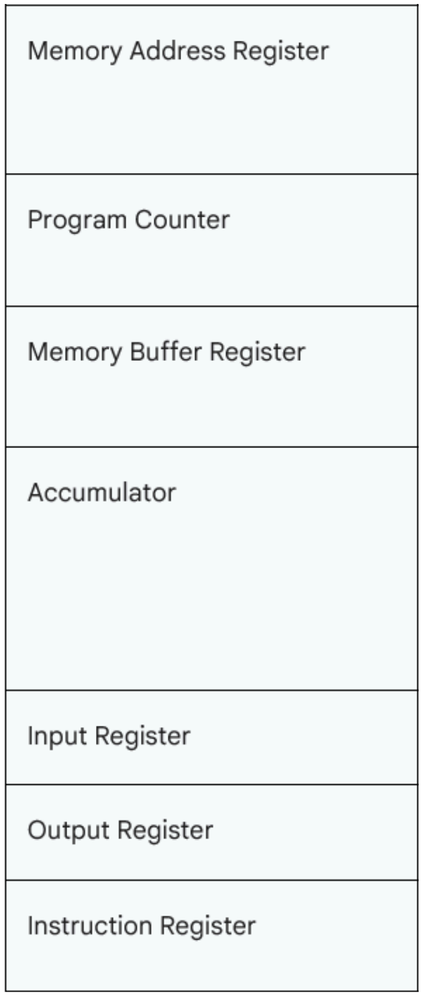
- MARIE (**Machine Architecture that is Really Intuitive and Easy**) is a simple, pedagogical computer architecture designed to clearly demonstrate the principles of the classical Von Neumann Architecture.
- It serves as an accessible learning tool for understanding the relationship between assembly language, machine code, and hardware organization.

MARIE Architecture and Memory Organization

Register	Acronym	Size (Bits)	Primary Function
Memory Address Register	MAR	12	Holds the memory address of the instruction or data being referenced during the current memory access.
Program Counter	PC	12	Holds the address of the instruction to be executed next.
Memory Buffer Register	MBR	16	Temporarily holds the instruction or data value either just read from memory or ready to be written to memory.[1]
Accumulator	AC	16	The general-purpose register holding temporary data values, serving as an implicit source/destination for ALU operations.[1] It is essential for Mapping High-Level Language (HLL) Constructs
Input Register	InREG	16	Holds data coming from an input device.[1]
Output Register	OutREG	16	Holds data destined for an output device.[1]
Instruction Register	IR	16	Holds the binary machine language instruction currently being executed.[1]

MARIE's Register Set

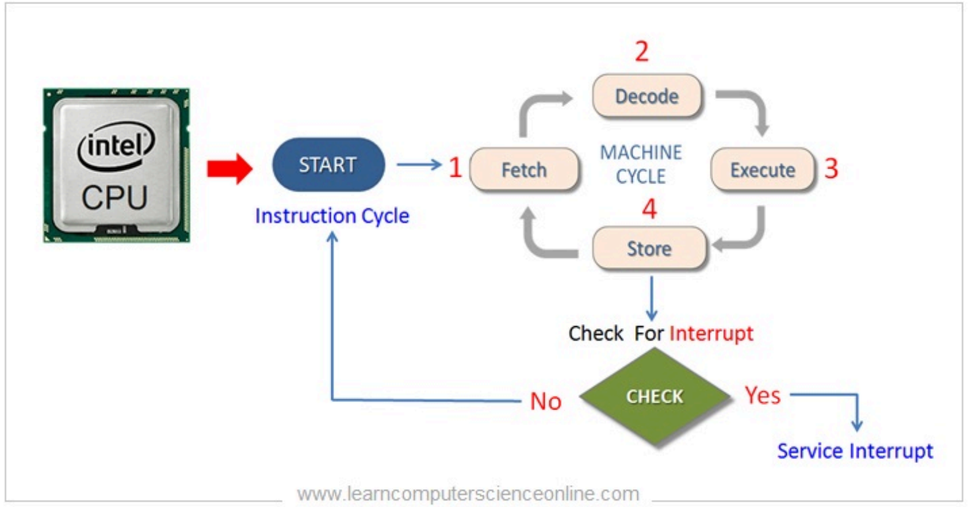
MARIE Architecture and Memory Organization



Instruction Processing

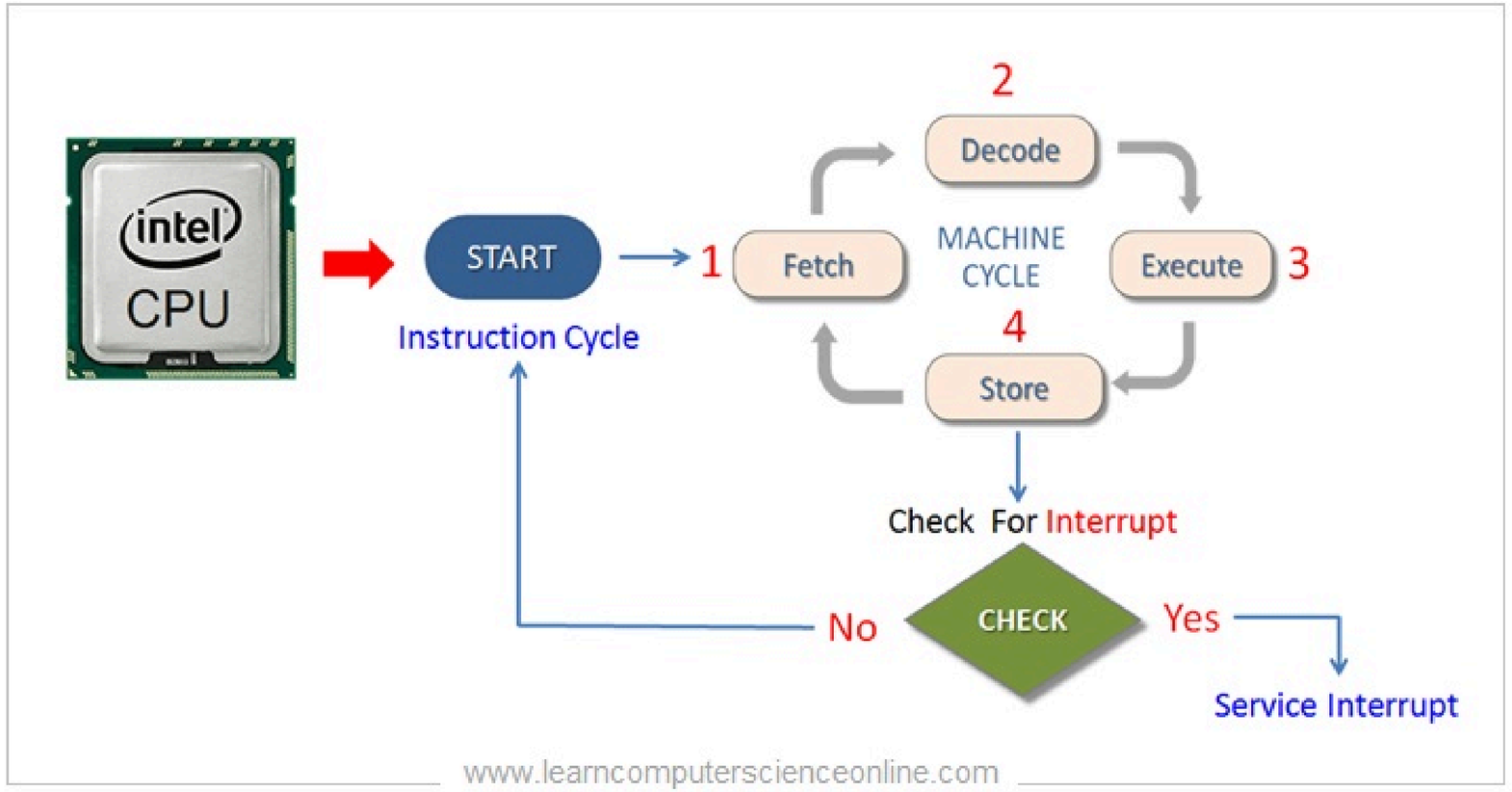
Instruction Processing

- The **instruction cycle**, also commonly referred to as the **fetch–decode–execute cycle**, is the fundamental sequence of actions that the Central Processing Unit (CPU) repeats continuously from the moment the computer boots up until it is shut down, enabling the processing of program instructions.
- Although the specifics of the cycle vary across different CPU designs and instruction sets, the process typically consists of the following phases :



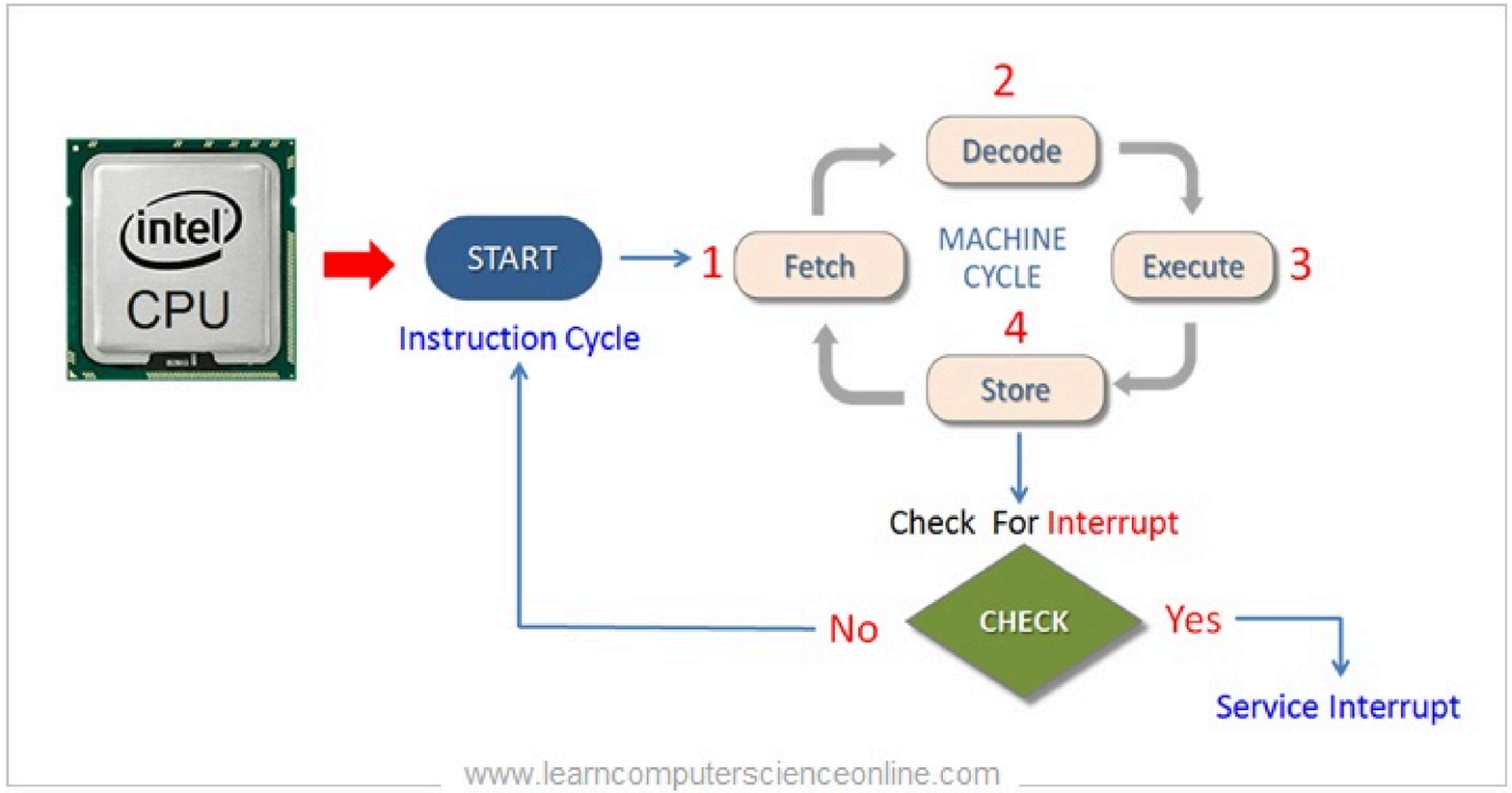
Instruction Processing

1. **Fetch Stage:** The CPU fetches the next instruction from the memory location specified by the Program Counter (PC) and loads it into the Instruction Register (IR).



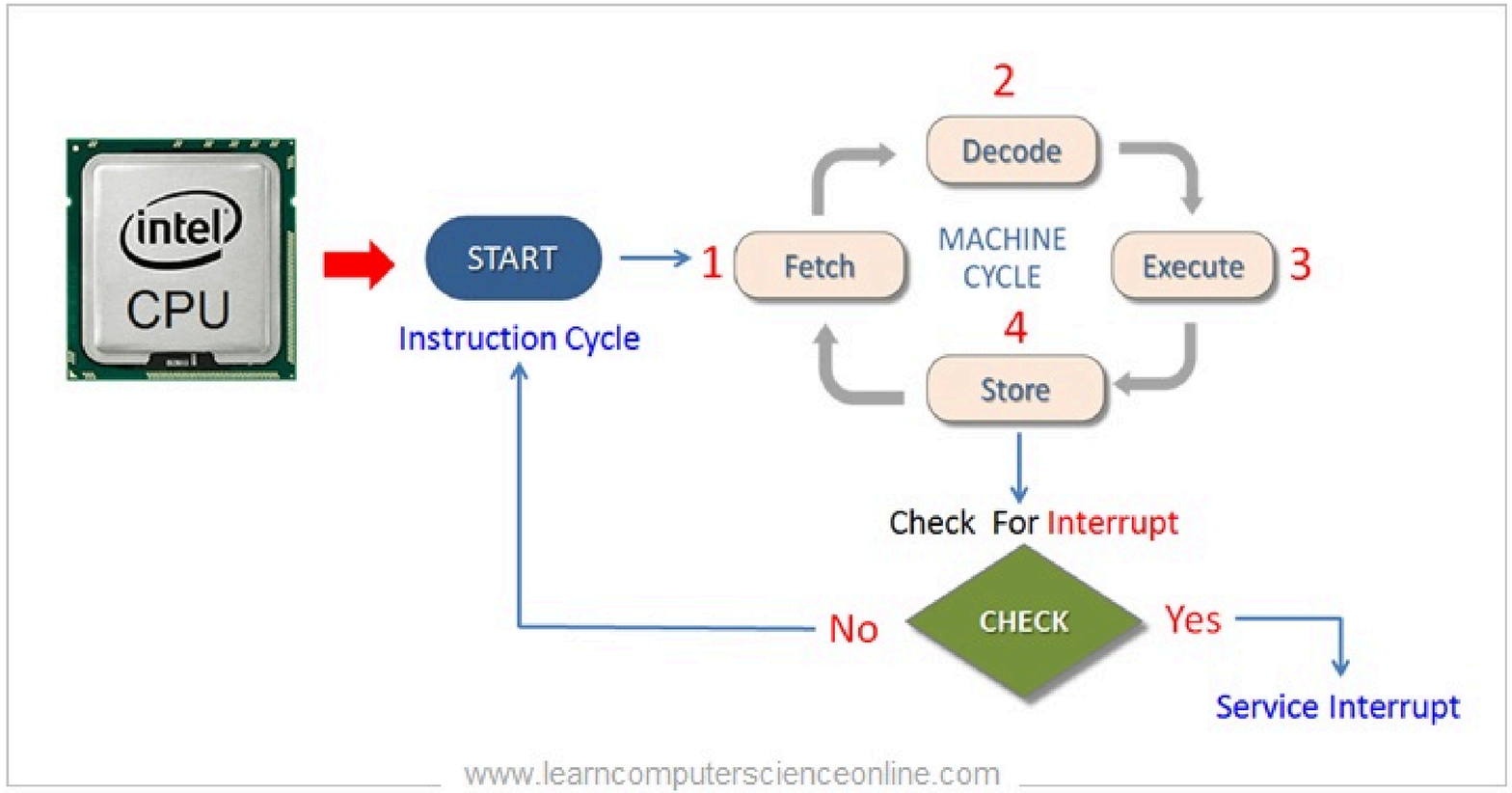
Instruction Processing

2. **Decode Stage:** The Control Unit interprets the encoded instruction contained in the IR to determine the operation to be performed and the operands required.



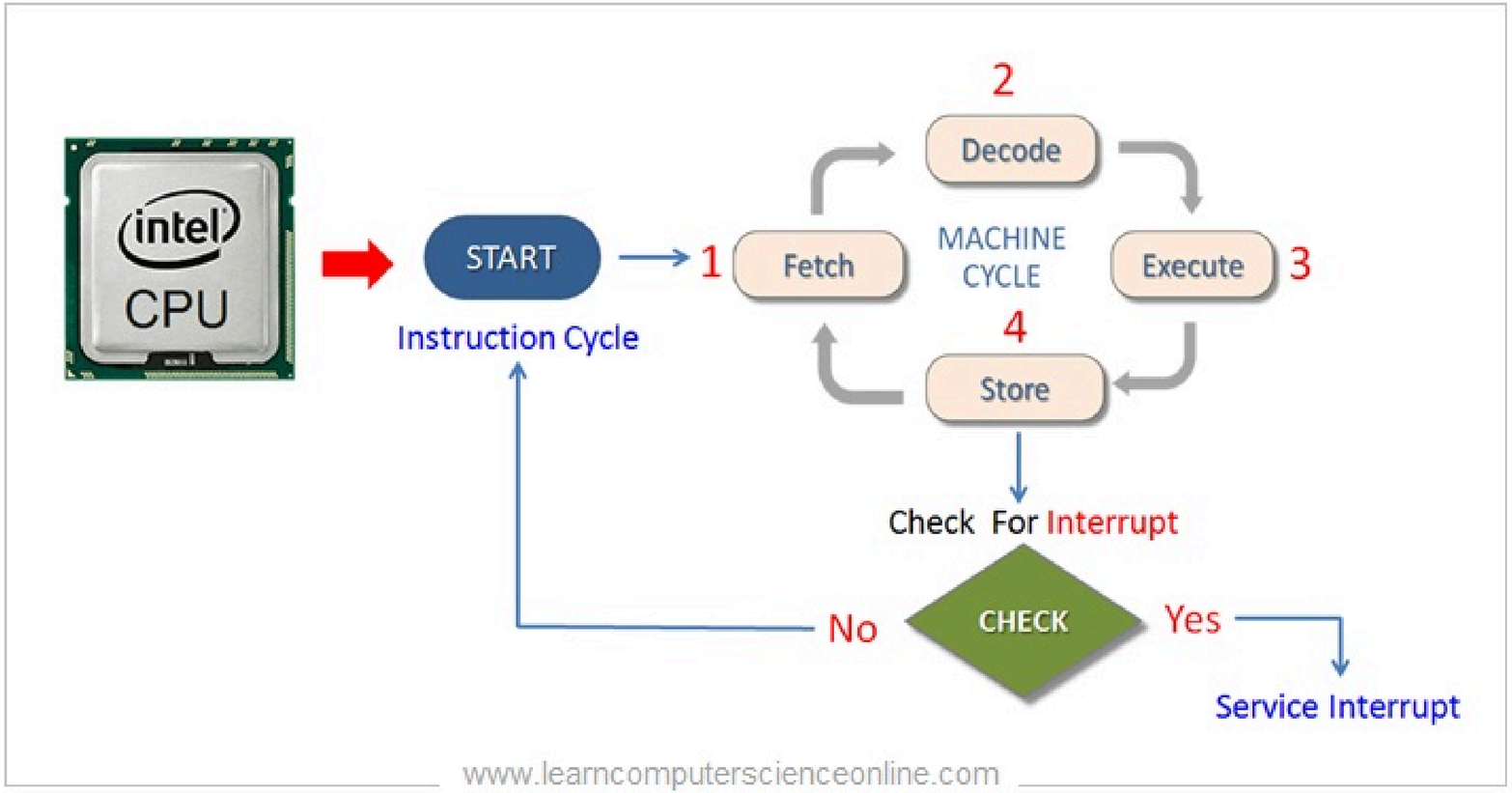
Instruction Processing

3. **Execute Stage:** The Arithmetic Logic Unit (ALU) performs the designated operation.



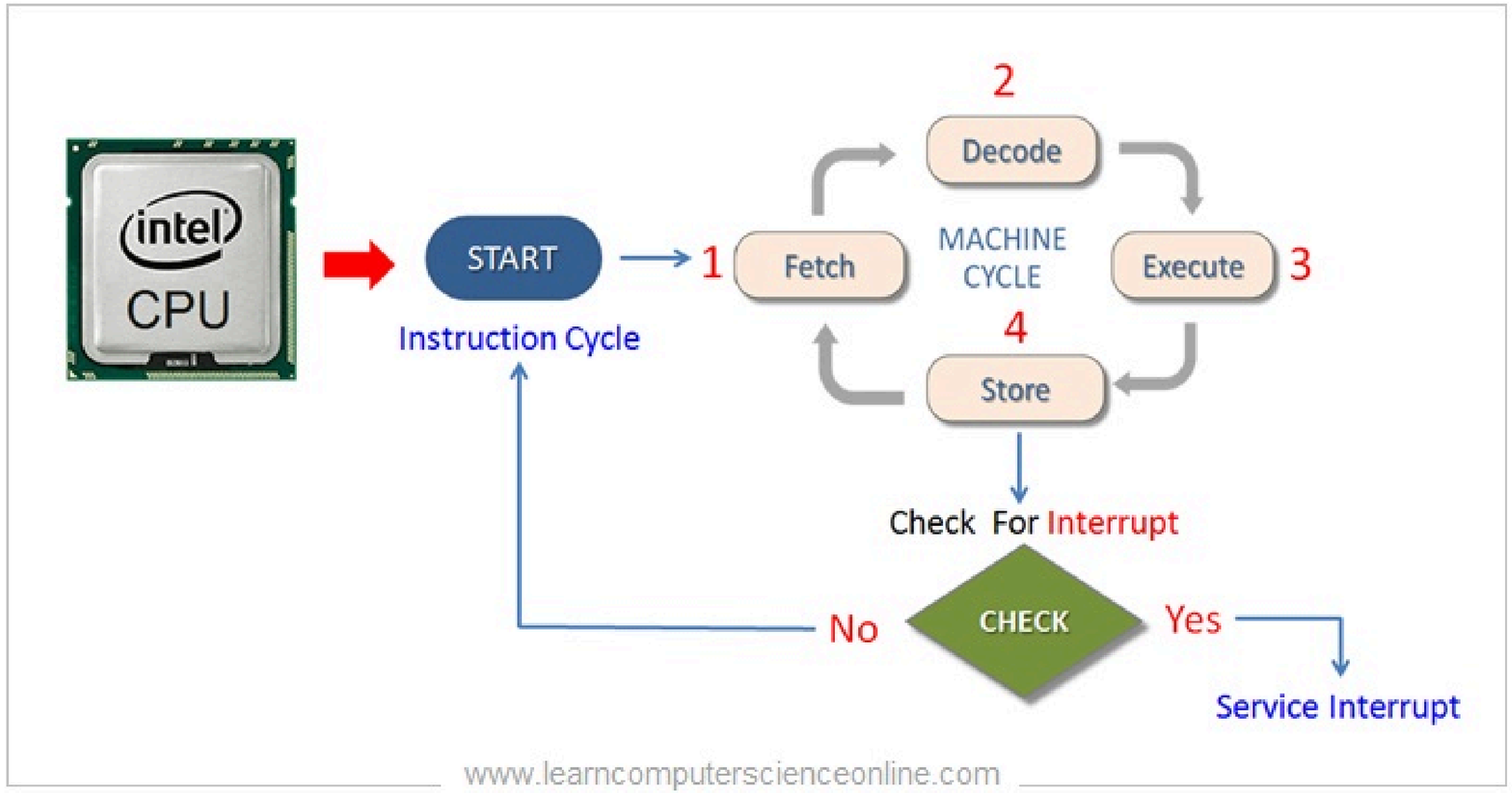
Instruction Processing

4. **Memory Access Stage:** This optional stage retrieves necessary data operands from memory or stores results back into memory .



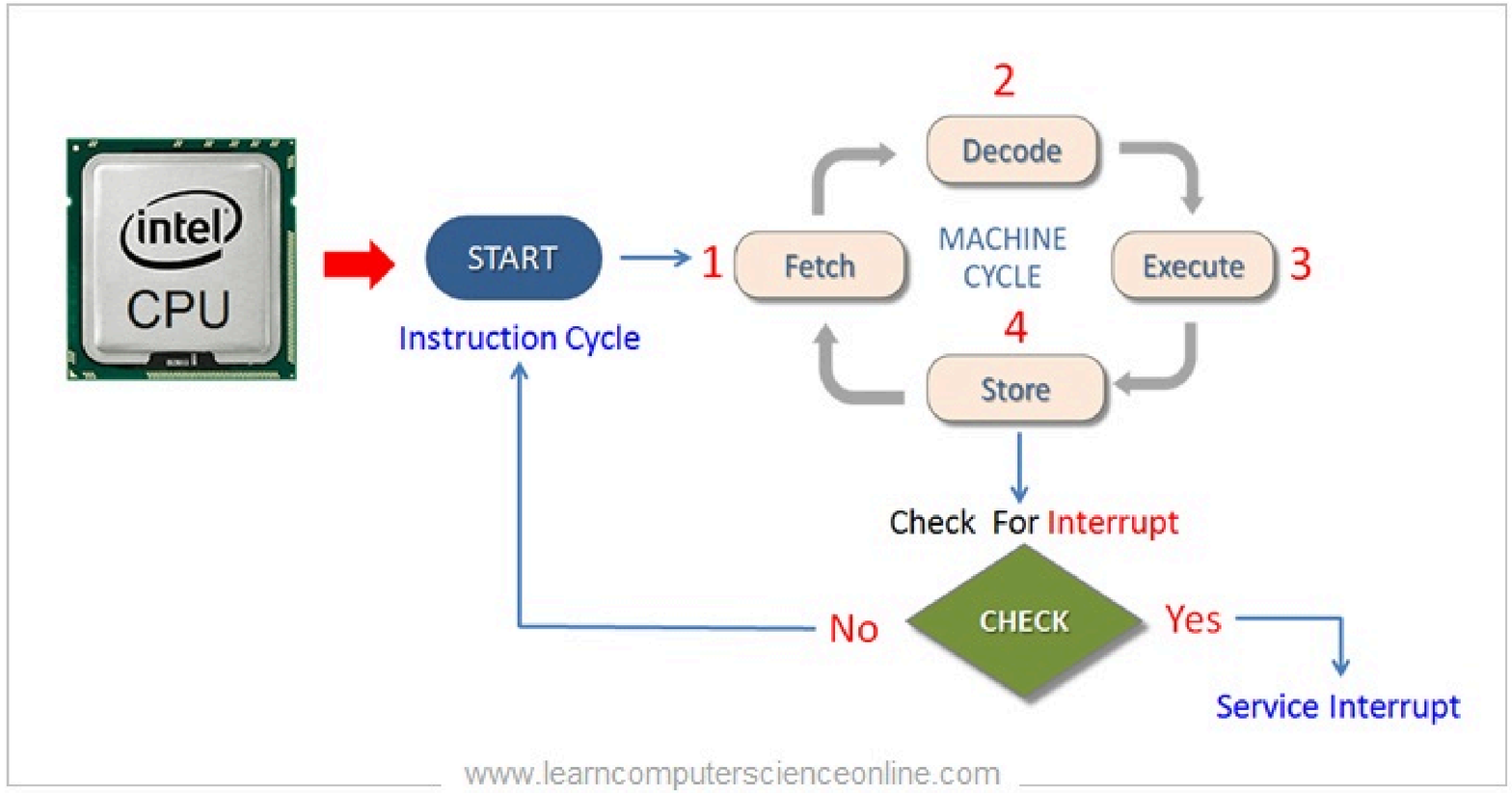
Instruction Processing

5. **Registry Write-Back Stage:** This optional stage writes the final result of the operation back into a register, typically the Accumulator (AC) in MARIE .



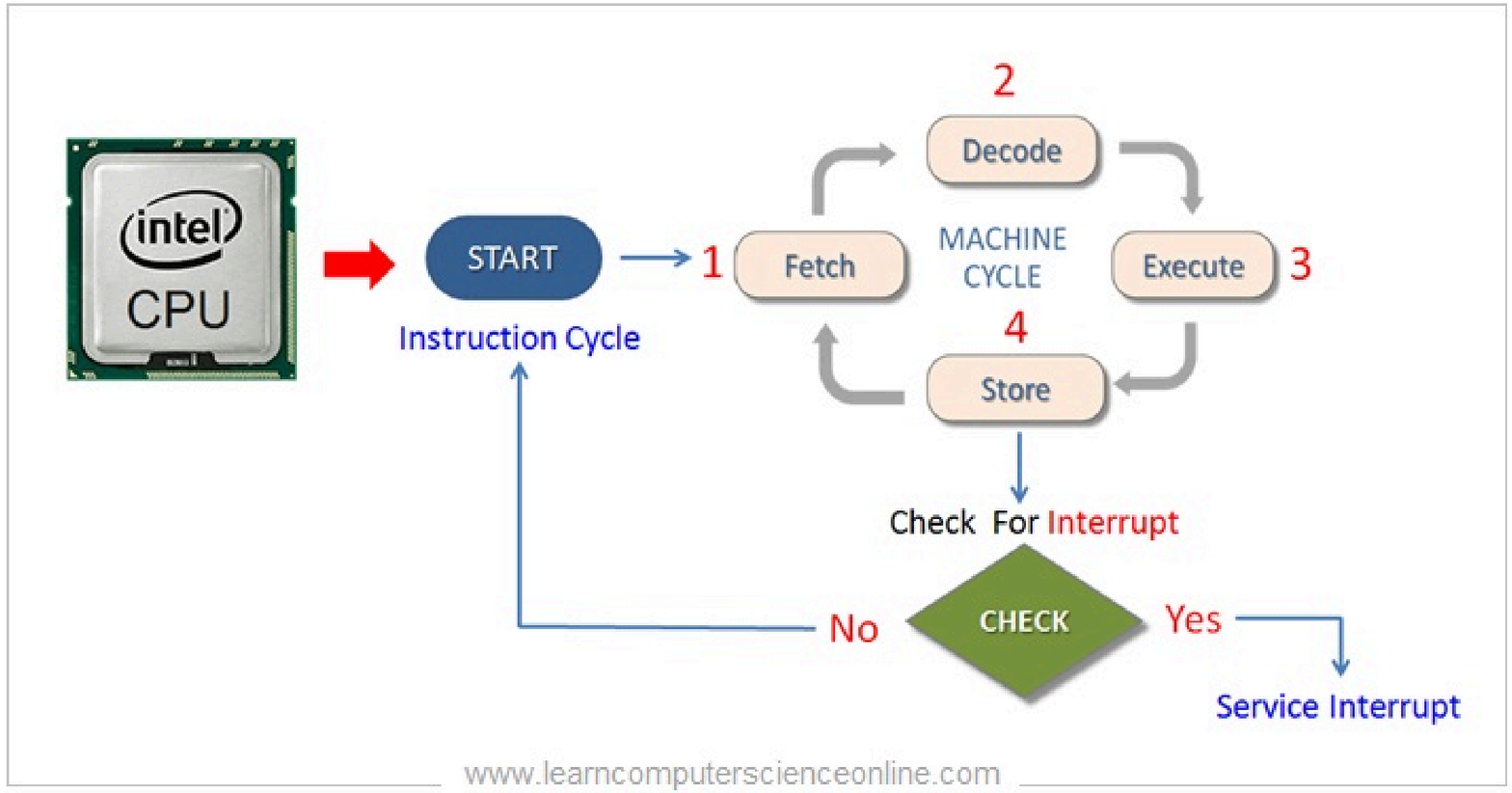
Instruction Processing

5. **Registry Write-Back Stage:** This optional stage writes the final result of the operation back into a register, typically the Accumulator (AC) in MARIE .



Instruction Processing

5. **Registry Write-Back Stage:** This optional stage writes the final result of the operation back into a register, typically the Accumulator (AC) in MARIE .



ILP and Hazards

ILP and Hazards

Instruction-Level Parallelism (ILP) – refers to the capacity of a processor to execute multiple instructions or parts of instructions simultaneously, rather than strictly one after another.

ILP and Hazards

- **Instruction-Level Parallelism (ILP)** – refers to the capacity of a processor to execute multiple instructions or parts of instructions simultaneously, rather than strictly one after another.
- ILP increases the overall **throughput and performance of the processor.** This parallel execution is achieved through a combination of hardware techniques and compiler optimizations.

End of Presentation

Questions...?

sana po sir madali lang exam hihi