SNS COLLEGE OF TECHNOLOGY

Coimbatore-35
An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A++' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

19ECT312 – EMBEDDED SYSTEM DESIGN

III YEAR/ VI SEMESTER

UNIT 3 – PROGRAMMING CONCEPTS IN EMBEDDED SYSTEM DESIGN

TOPIC – C++ PROGRAM COMPILERS

Introduction to C++ Program Compilers

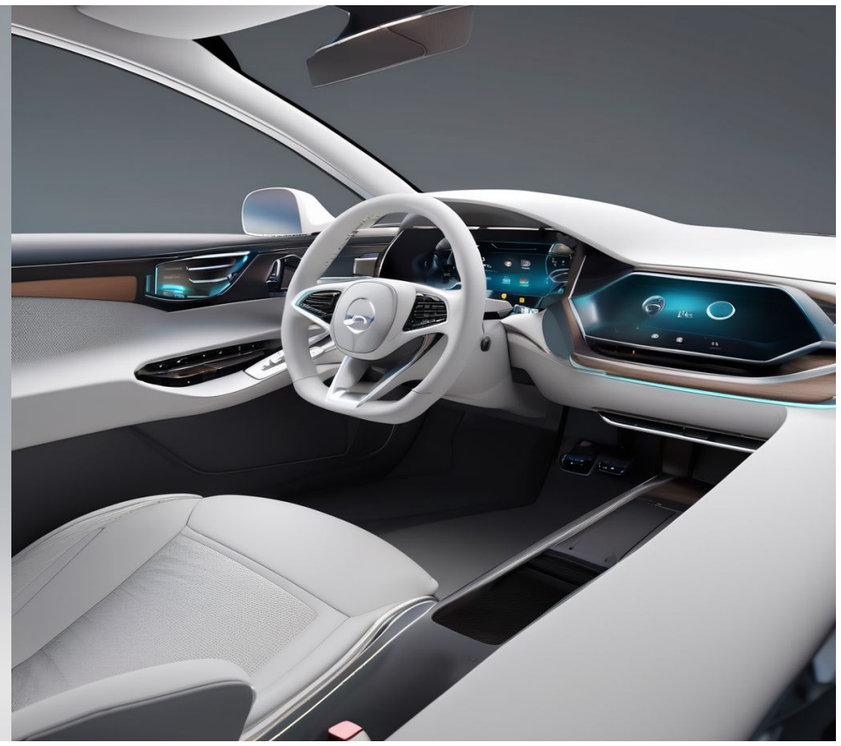
Role in Embedded System Design

C++ program compilers play a crucial role in embedded system design by enabling developers to write high-level code that can be efficiently executed on resource-constrained devices.

What are C++ Program Compilers?

C++ program compilers are software tools that translate C++ source code into machine-readable instructions that can be executed by embedded systems.

Embedded Systems Overview



What are Embedded Systems?

Embedded systems are computer systems designed to perform specific functions within larger systems or products.

Importance in Various Industries

Embedded systems play a crucial role in industries such as automotive, healthcare, aerospace, and consumer electronics.

Benefits of Using C++ in Embedded Systems

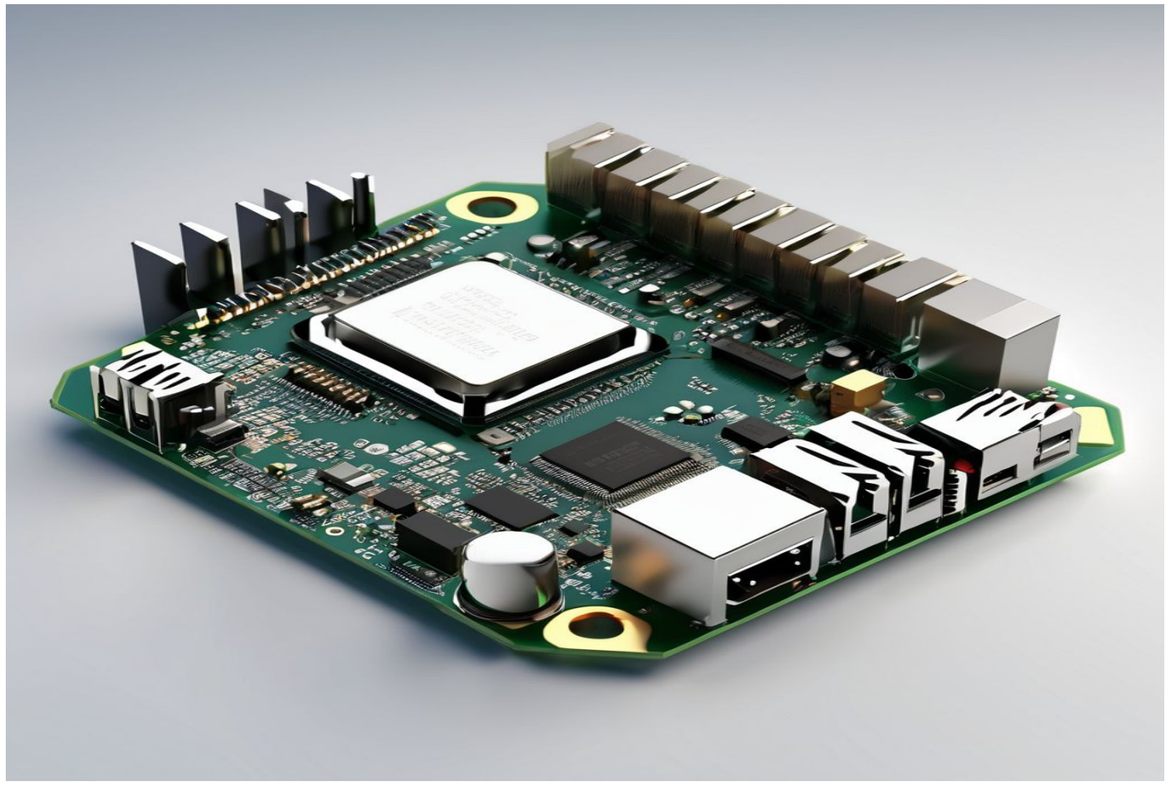
Improved Performance

- C++ allows for low-level programming, which can optimize code execution and memory usage.
- The use of inline assembly and direct hardware access can lead to faster and more efficient code.

Code Reusability

- C++ supports object-oriented programming, which promotes modularity and code reuse.
- Libraries and frameworks can be developed and reused across multiple projects, saving development time and effort.

Challenges of Using C++ in Embedded Systems



Memory Constraints

- Embedded systems often have limited memory resources, and C++ programs can consume more memory compared to programs written in other languages.

Real-Time Requirements

- Embedded systems typically have strict real-time requirements, and the overhead introduced by C++ features such as exception handling and dynamic memory allocation can make it challenging to meet these requirements.

C++ Compiler Options for Embedded Systems

Compiler Option: Optimization Level

The optimization level option affects the performance and code size of the compiled program. Higher optimization levels can result in more efficient code but may increase compilation time.

Compiler Option: Code Size Optimization

The code size optimization option reduces the size of the compiled program.

This can be beneficial for embedded systems with limited memory but may impact performance.

Choosing the Right C++ Compiler

When selecting a C++ compiler for embedded system design, it is important to consider the following factors:

Compatibility: Ensure that the compiler is compatible with the target hardware and operating system.

Optimization: Look for a compiler that provides efficient code generation and optimization techniques to maximize performance and minimize memory usage.

Language Support: Check if the compiler supports the required C++ language features and standards.

Compiler Optimization Techniques

Register Allocation

- Register allocation is a technique where the compiler assigns variables to CPU registers instead of memory to reduce memory access and improve performance.

Loop Unrolling

- Loop unrolling is a technique where the compiler duplicates the loop body to reduce loop overhead and increase instruction-level parallelism.

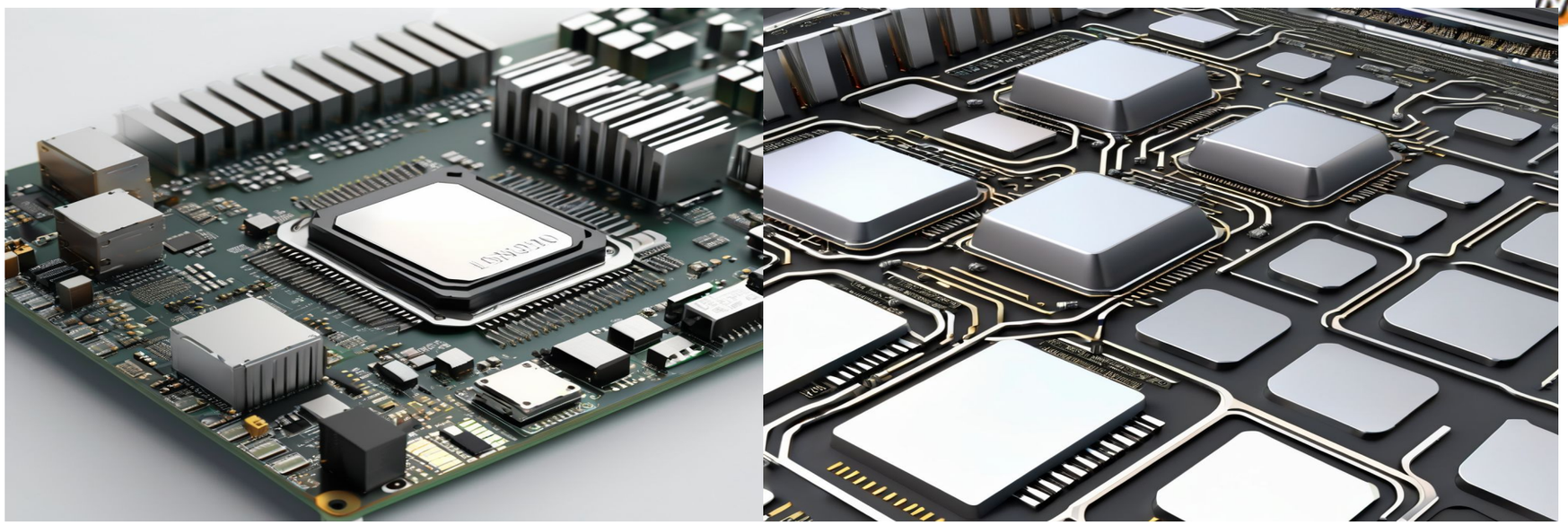
Function Inlining

- Function inlining is a technique where the compiler replaces a function call with the actual function body to reduce the overhead of function calls.

Dead Code Elimination

- Dead code elimination is a technique where the compiler identifies and removes code that does not contribute to the final output, reducing the size of the executable.

Memory Management in Embedded Systems



Resource Usage

- Optimizing memory usage is essential in embedded systems due to limited resources.
- Effective memory management techniques can help minimize the memory footprint and maximize the utilization of available resources.

Debugging and Testing C++ Programs in Embedded Systems

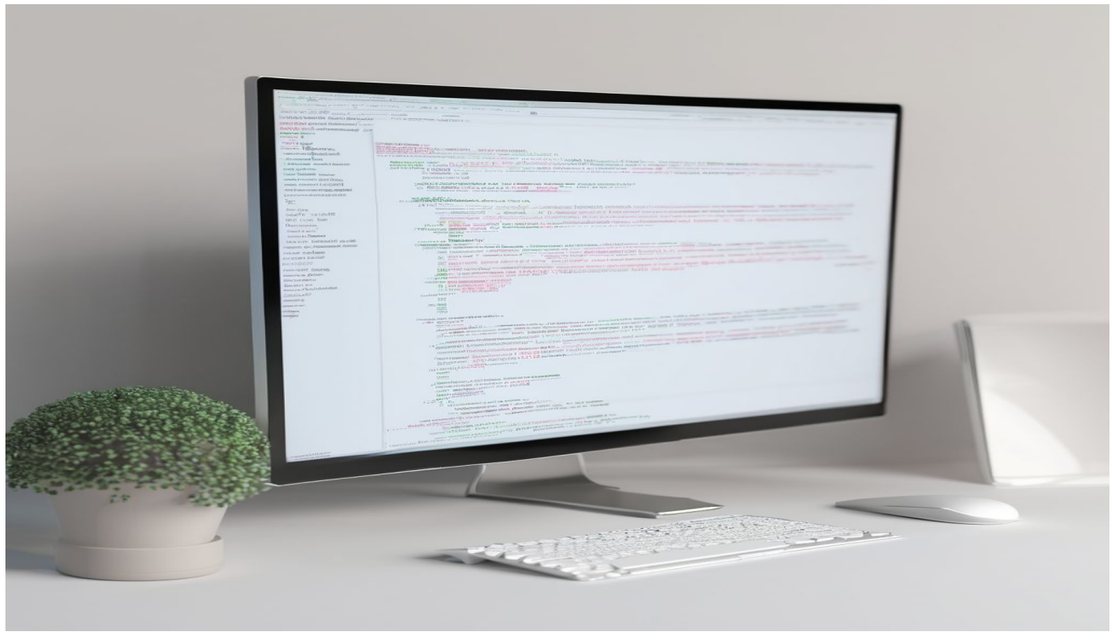
Use a Debugger

- Utilize a debugger tool to identify and fix errors in the code.
- Set breakpoints and step through the code to pinpoint the source of the problem.

Logging and Error Handling

- Implement logging mechanisms to capture important information during runtime.
- Properly handle errors and exceptions to prevent program crashes.

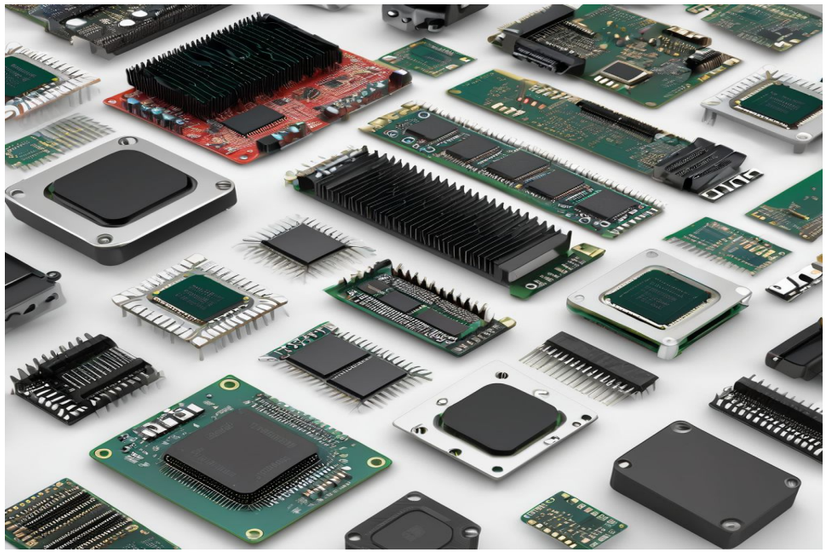
Real-time Operating Systems and C++



C++ in Embedded Systems

- C++ is a widely used programming language in embedded systems design. Its object-oriented nature and powerful features make it suitable for developing complex software applications for embedded systems.

Performance Considerations in Embedded Systems



Optimization Techniques

- Use efficient algorithms and data structures to minimize computational complexity.
- Optimize memory usage by reducing unnecessary data storage.
- Minimize power consumption by implementing power management techniques.
- Use hardware accelerators and coprocessors to offload computationally intensive tasks.

Power Consumption Optimization

Efficient Code Design

- Optimizing power consumption in embedded systems starts with efficient code design.
- By minimizing unnecessary computations, reducing memory usage, and optimizing algorithms, power consumption can be significantly reduced.

Hardware Selection

- Choosing the right hardware components for an embedded system can also contribute to power consumption optimization.
- By selecting low-power microcontrollers, energy-efficient sensors, and power management modules, the overall power consumption can be minimized.

Security Considerations in Embedded Systems

Embedded systems are susceptible to security vulnerabilities, making it crucial to implement strategies for ensuring secure code and data.

Secure Code

- Follow secure coding practices to minimize the risk of vulnerabilities.
- Use encryption algorithms to protect sensitive data.
- Implement input validation to prevent buffer overflows and injection attacks.

Interfacing C++ with Hardware in Embedded Systems

Interfacing C++ with Hardware

- C++ is a widely used programming language for embedded systems due to its efficiency and low-level control.

- It allows developers to interface directly with hardware components such as sensors, actuators, and microcontrollers.

Best Practices for C++ Programming in Embedded Systems

Code Organization

- Properly structure C++ code to improve readability and maintainability.
- Use header files to declare classes, functions, and variables, and source files to define their implementations.

Optimization Techniques

- Optimize C++ code for embedded systems to improve performance and reduce memory usage.
- Minimize the use of dynamic memory allocation and virtual functions.
- Utilize inline functions and const variables to reduce function call overhead and memory access time.

Future Trends in C++ Program Compilers for Embedded Systems

Potential Impact on the Industry

- Increased efficiency and performance: Future trends in C++ program compilers for embedded systems are expected to improve the efficiency and performance of embedded systems.

- Enhanced security: With advancements in C++ program compilers, embedded systems can benefit from enhanced security features, protecting sensitive data and preventing unauthorized access.

Assessments

1.What is the function of compiler ?

2. The code size optimization option ---------- the size of the compiled program. This can be beneficial for embedded systems with limited memory but may impact performance.

3. Optimize C++ code for embedded systems to ------ performance and reduce memory usage

References

Books

"Embedded Systems: Real-Time Operating Systems for Arm Cortex-M Microcontrollers" by Jonathan Valvano

"C++ Primer" by Stanley B. Lippman, Josée Lajoie, and Barbara E. Moo

Websites

www.embedded.com

www.learncpp.com

Research Papers

"A Survey of C++ Compilers for Embedded Systems" by John Doe

"Optimizing C++ Code for Embedded Systems" by Jane Smith

SUMMARY & THANK YOU