



Code Optimization

.....

Team Members:

BHUVANESH E S (RA2011003010021)

AKKASH ANUMALA (RA2011003010015)

Definition of Code Optimization

.....

Introducing the concept and its importance in compiler design

Code optimization is the process of enhancing the efficiency of code by minimizing resource utilization and improving program speed and performance.

Importance of Code Optimization

.

To highlight the significance of code optimization in compiler design

- **Improved Performance**

Code optimization can significantly improve the performance of compiled programs.

- **Reduced Memory Usage**

Optimized code consumes less memory, allowing for more efficient use of resources.

- **Lower Energy Consumption**

Optimized code requires less energy to execute, contributing to energy efficiency.

- **Improved Maintainability**

Optimized code is often more readable and maintainable, making it easier to modify and troubleshoot.



Techniques for Code Optimization

.....

Constant Folding

.

Technique and Benefits

- **Definition**

Constant folding is a compiler optimization technique that involves identifying and evaluating constant expressions at compile time instead of at runtime.

- **Benefits**

Constant folding can lead to improved performance and reduced memory usage, as it eliminates unnecessary computation and generates more efficient code.

Dead Code Elimination

.....

Discussing the technique and why it's important for code optimization in compiler design.

Loop Optimization

.

To provide an overview of loop optimization and its advantages

- **Reduces CPU usage and improves execution time**

By rearranging the code within loops, the number of CPU instructions required can be reduced, leading to faster execution times.

- **Minimizes memory access and cache misses**

Loop optimization reduces the number of memory accesses, which in turn reduces the number of cache misses, leading to faster program execution.

- **Improves code locality**

By optimizing loops, the code can be arranged in memory to enhance cache locality, which further enhances program performance.

- **Enables vectorization**

Loop optimization can make it easier for the compiler to vectorize the code, which can lead to significant performance improvements on modern CPUs.



Conclusion

.....

Summary of Techniques Discussed

.

- **Constant Folding**

Evaluating constant expressions at compile-time

- **Dead Code Elimination**

Removing unreachable code from the program

- **Loop Optimization**

Altering the code structure to improve loop performance



Closing

.....



Thank You!

.....