1. Which phase of the compiler is responsible for generating intermediate code?

a) Syntax Analysis
b) Semantic Analysis
c) Intermediate Code Generation
d) Code Optimization

Answer: c) Intermediate Code Generation

Explanation: Intermediate code generation is the phase in the compiler where high-level code is transformed into a simpler intermediate representation suitable for further processing.

2. Which of the following is NOT a typical construct handled during intermediate code generation?

a) Declarations
b) Loops
c) Boolean expressions
d) Case statements

Answer: b) Loops

Explanation: While loops, for loops, and other loop constructs are usually handled during high-level semantic analysis and later during code generation and optimization phases.

3. What is the purpose of back patching in intermediate code generation?

a) To optimize generated code

b) To resolve addresses in jump statements
c) To perform type checking
d) To handle variable declarations

Answer: b) To resolve addresses in jump statements

Explanation: Back patching is a technique used to fill in the target addresses of jump statements (like goto or if statements) once the target is known.

4. Which phase of the compiler is responsible for allocating registers and assigning them to variables?

a) Lexical Analysis
b) Intermediate Code Generation
c) Code Optimization
d) Register Allocation

Answer: d) Register Allocation

Explanation: Register allocation is a phase in code generation where physical or virtual registers are assigned to variables to minimize memory accesses.

5. In compiler design, what is a basic block?

a) A block of code without any control flow statements
b) A block of code with only one entry point and one exit point
c) A block of code representing a loop
d) A block of code with multiple exit points

Answer: b) A block of code with only one entry point and one exit point

Explanation: A basic block is a sequence of code with one entry point and one exit point; control flows into the block at the entry point and flows out at the exit point.

6. Which data structure is commonly used to represent the control flow of a program during code generation?

a) Linked List
b) Tree
c) Graph
d) Stack

Answer: c) Graph

Explanation: Control flow graphs (CFGs), which represent the flow of control within a program, are commonly represented using graph data structures.

7. What is the primary goal of peephole optimization?

a) To optimize the use of registers
b) To optimize the size of the generated code
c) To optimize the control flow
d) To optimize memory access

Answer: b) To optimize the size of the generated code

Explanation: Peephole optimization is a technique where the compiler looks for and applies

optimizations within small, localized sections of generated code, primarily aiming to reduce code size.

8. Which phase of the compiler typically performs peephole optimization?

a) Lexical Analysis
b) Intermediate Code Generation
c) Code Optimization
d) Code Generation

Answer: c) Code Optimization

Explanation: Peephole optimization is usually performed during the code optimization phase, where the compiler analyzes generated code and applies various optimizations.

9. What is the purpose of a DAG representation of basic blocks?

a) To simplify complex control flow
b) To optimize memory access
c) To represent code in a graphical format
d) To identify common subexpressions

Answer: d) To identify common subexpressions

Explanation: Directed Acyclic Graphs (DAGs) are often used to identify and eliminate redundant computations by representing common subexpressions in a compact form.

10. Which phase of the compiler involves converting DAGs back into executable code?

a) Lexical Analysis
b) Intermediate Code Generation
c) Code Optimization
d) Code Generation

Answer: d) Code Generation

Explanation: After optimization, DAGs are typically translated back into executable code during the code generation phase of the compiler.

11. Which of the following is NOT an issue in the design of a code generator?

a) Register allocation
b) Memory management
c) Optimization techniques
d) Lexical analysis

Answer: d) Lexical analysis

Explanation: Lexical analysis deals with tokenizing source code, which is not directly related to the design of a code generator.

12. Which of the following is a primary concern in register allocation and assignment?

a) Minimizing compilation time
b) Reducing the size of the source code
c) Minimizing the number of register spills
d) Maximizing the number of temporary variables

Answer: c) Minimizing the number of register spills

Explanation: Register allocation aims to minimize the need for storing variables in memory (register spills), which can degrade performance.

13. Which optimization technique focuses on identifying and eliminating redundant computations?

a) Loop optimization
b) Peephole optimization
c) Common subexpression elimination
d) Inline expansion

Answer: c) Common subexpression elimination

Explanation: Common subexpression elimination is a technique that identifies repeated computations and replaces them with a single computation, thus optimizing the generated code.

14. What is the purpose of basic blocks in code generation?

a) To group related instructions together
b) To represent loops and conditional statements
c) To simplify the control flow of the program
d) To allocate memory for variables

Answer: a) To group related instructions together

Explanation: Basic blocks are used to organize instructions into cohesive units, simplifying analysis and optimization of code.

15. Which phase of the compiler focuses on rearranging code to improve performance without changing its semantics?

a) Lexical Analysis
b) Code Optimization
c) Intermediate Code Generation
d) Syntax Analysis

Answer: b) Code Optimization

Explanation: Code optimization aims to improve the efficiency of generated code by applying various transformations while preserving the program's behavior.

16. In a flow graph, what do nodes represent?

a) Instructions
b) Variables
c) Expressions
d) Control flow paths

Answer: d) Control flow paths

Explanation: In a flow graph, nodes represent points in the program where control flow decisions are made, such as branches, loops, or function calls.

17. Which of the following is NOT typically considered in register allocation?

a) Minimizing register spills
b) Minimizing memory usage
c) Maximizing the number of available registers
d) Minimizing the number of live ranges

Answer: b) Minimizing memory usage

Explanation: Register allocation focuses on minimizing the need for accessing memory by maximizing the use of registers for variables.

18. What is the main advantage of using a DAG representation in code generation?

a) Reducing compilation time
b) Simplifying debugging
c) Identifying common subexpressions
d) Improving code readability

Answer: c) Identifying common subexpressions

Explanation: DAG representation helps in identifying common subexpressions, which can be eliminated to improve the efficiency of generated code.

19. Which optimization technique involves replacing a function call with the actual body of the function?

a) Function inlining

b) Loop unrolling
c) Dead code elimination
d) Constant folding

Answer: a) Function inlining

Explanation:

Function inlining replaces a function call with the actual body of the function, potentially improving performance by eliminating the overhead of function calls.

20. What is the primary goal of generating code from a DAG representation?

a) To minimize compilation time
b) To improve code readability
c) To optimize memory access
d) To efficiently represent common subexpressions

Answer: c) To optimize memory access

Explanation: Generating code from a DAG representation allows for efficient memory access by identifying and eliminating redundant computations and memory accesses.

Related posts:

1. Introduction to Information Security
2. Introduction to Information Security MCQ
3. Introduction to Information Security MCQ

4. Symmetric Key Cryptography MCQ
5. Asymmetric Key Cryptography MCQ
6. Authentication & Integrity MCQ
7. E-mail, IP and Web Security MCQ