SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

Ramapuram Campus, Bharathi Salai, Ramapuram, Chennai - 600089

**FACULTY OF ENGINEERING AND TECHNOLOGY**

# DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

****

**QUESTION BANK**

**DEGREE / BRANCH: B.TECH-CSE & CSE WITH SPECIALIZATION**

**VI SEMESTER**

**18CSC304JT – COMPILER DESIGN**

**2018 Regulation**

**Academic Year 2022-2023**

**EVEN SEMESTER**

# SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

**Ramapuram Campus, Bharathi Salai, Ramapuram, Chennai-600089**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**QUESTION BANK**

**SUBJECT : 18CSC304J – COMPILER DESIGN**

**SEM/ YEAR: III/V**

**Course Outcomes**

CO-1 : Acquire the knowledge of mathematics and engineering principles for the Design of Compilers

CO-2 : Acquire the ability to identify specification of a language's lexical rules of Lexical Analyzer

CO-3 : Apply the knowledge of Syntax Analyzer for parsing the sentences in a compiler grammar

CO-4 : Understand the concepts of translation of various intermediate codes .

CO-5 : Apply the knowledge to implement Code Generator for compilers

CO-6 : Analyze and Design the methods of developing a Code Optimizer

| **UNIT V** | | | |
| --- | --- | --- | --- |
| Code optimization Introduction– Principal Sources of Optimization-Function Preserving Transformation- Loop Optimization-Optimization of basic Blocks- Building Expression of DAG-Peephole Optimization- Basic Blocks, Flow Graphs-Next -Use Information- Introduction to Global Data Flow Analysis-Computation of gen and kill- Computation of in and out-Parameter Passing.- Runtime Environments-Source Language issues- Storage Organization-Activation Records- Storage Allocation strategies. | | | |
| **PART-A (Multiple Choice Questions)** | | | |
| **Q.**  **No** | **Questions** | **Course Outcome** | **Competence**  **BT Level** |
| **1** | **The optimization which avoids test at every iteration is T1-5**  **a)Loopunrolling**  b)Loopjamming  c)Constant folding  d) Dead code elimination | CO6 | BT 2 |
| **2** | **The identification of common sub-expression and replacement of run time computations by compile time computation is**   1. Local optimization 2. Loop optimization 3. **Constant folding** 4. Data flow analysis | CO6 | BT 2 |
| **3** | **The transformation of replacing an expensive operation,**  **such as multiplication by a cheaper one such as addition is**  **known as**   1. **Strength reduction** 2. Code motion 3. Dead code elimination 4. Common sub-expression elimination | CO6 | BT 1 |
| **4** | **Specify the optimization technique used in the following code. T1-**  **Initial code:**  **for (int i=0; i<5; i++)**  **printf("computer\n");**  **Optimized code:**  **printf("computer\n");**  **printf("computer\n");**  **printf("computer\n");**  **printf("computer\n");**  **printf("computer\n");**  **a) loop unrolling**  b) loop jamming  c) Dead code elimination  d) Common sub-expression elimination | CO6 | BT 3 |
| **5** | **\_\_\_\_\_\_\_\_ is used to keep track of the live procedure activations i.e the procedures whose execution have not been completed.**   1. Local variables 2. Symbol table 3. **control stack** 4. Access log   **Answer: c** | CO6 | BT 1 |
| **6** | **Specify the optimization technique used in the following code. T1-**  **// before elimination**  **c = a \* b**  **x = a**  **till**  **d = a \* b + 4**  **//After elimination :**  **c = a \* b**  **till**  **d = a \* b + 4**  a)Strength reduction  b)Code motion  **c)Dead code elimination**  d)Common sub-expression elimination | CO6 | BT 3 |
| **7** | **In which of the following parameter passing techniques, the calling procedure pass the r-value of the actual parameters and the compiler puts that into called procedure’s activation record.**   1. **Call by value** 2. Call by reference 3. Call by copy restore   d) Call by name | CO6 | BT 2 |
| **8** | **Some code optimizations are carried out on the intermediate code because**  **a) They enhance the portability of the compiler to other target processors**  b) Program analysis is more accurate on intermediate code than on machine code  c)The information from dataflow analysis cannot otherwise be used for optimization  d) The information from the front end cannot otherwise be used for optimization | CO6 | BT 2 |
| **9** | **The code optimization is carried out on the intermediate code**  a) Because for optimization information from the front end cannot be used  **b) Because program is more accurately analyzed on intermediate code than on machine code.**  c) Because for optimization information from data flow analysis cannot be used  d) Because they enhance the portability of the compiler to the other target  processor. | CO6 | BT 3 |
| **10** | **Running time of a program depends on**  a)The way the registers and addressing modes are used  b)The order in which computations are performed  c)The usage of machine idioms  **d)All of these** | CO6 | BT 2 |
| **11** | **A language L allows declaration of arrays whose sizes are not known during compilation. It is required to make efficient use of memory. Which one of the following is true?**  a) A compiler using static memory allocation can be written for L  b) A compiler cannot be written for L ; an interpreter must be used  **c) A compiler using dynamic memory allocation can be written for L**  d) None of these | CO6 | BT 2 |
| **12** | **Local and loop optimization in turn provide motivation for**   1. **Data flow analysis** 2. constant folding 3. peep hole optimization 4. DFA and constant folding | CO6 | BT 2 |
| **13** | **In a compiler, the data structure responsible for the management of information about variables and their attributes is**   1. Semantic stack 2. Parser table 3. **Symbol table** 4. Abstract syntax-tree | CO6 | BT 1 |
| **14** | **Consider the following C code segment.**  **for (i = 0, i<n; i++)**  **{**  **for (j=0; j<n; j++)**  **{**  **if (i%2)**  **{**  **x += (4\*j + 5\*i);**  **y += (7 + 4\*j);**  **}**  **}**  **}**  **Which one of the following is false?**   1. The code contains loop invariant computation 2. There is scope of common sub-expression elimination in this code 3. There is scope of strength reduction in this code 4. **There is scope of dead code elimination in this code** | CO6 | BT 3 |
| **15** | **In compiler terminology reduction in strength means**     1. Replacing run time computation by compile time computation 2. Removing loop invariant computation 3. Removing common sub-expressions 4. **Replacing a costly operation by a relatively cheaper one** | CO6 | BT 2 |
| **16** | **Advantage of panic mode of error recovery is that**   1. It is simple to implement 2. It never gets into an infinite loop 3. **Both (a) and (b)** 4. It helps in finding logical errors | CO6 | BT 2 |
| **17** | **The languages that need heap allocation in the runtime environment are**   1. Those that use global variables 2. Those that use dynamic scoping 3. Those that support recursion 4. **Those that allow dynamic data structure.** | CO6 | BT 2 |
| **18** | **Specify the optimization technique used in the following code. T1-**  **a = 200;**  **while(a>0)**  **{**  **b = x + y;**  **if (a % b == 0}**  **printf(“%d”, a);**  **}**  **//This code can be optimized as**  **a = 200;**  **b = x + y;**  **while(a>0)**  **{**  **if (a % b == 0}**  **printf(“%d”, a);**  **}**  a)Strength reduction  **b)Code motion**  c)Dead code elimination  d)Common sub-expression elimination | CO6 | BT 3 |
| **19** | **Memory allocation and deallocation can be done at any time and at any place depending on the requirement of the user. This statement is applicable to \_\_\_\_\_\_\_\_.**   1. Stack   **b) heap**  c) static  d) All of the above | CO6 | BT 1 |
| **20** | **Specify the optimization technique used in the following code.**  **i = 1;**  **while (i<10)**  **{**  **y = i \* 4;**  **}**  **//After Reduction**  **i = 1**  **t = 4**  **{**  **while( t<40)**  **y = t;**  **t = t + 4;**  **}**  **a)Strength reduction**  b)Code motion  c)Dead code elimination  d)Common sub-expression elimination | CO6 | BT 3 |
| **PART-B** | | | |
| **1** | Explain the concept of code optimization? | CO6 | BT 1 |
| **2** | Name the properties of optimizing compilers? | CO6 | BT 2 |
| **3** | Give any two examples of strength reduction. | CO6 | BT 2 |
| **4** | Explain the technique used in loop optimization. | CO6 | BT 1 |
| **5** | What is peephole optimization technique? | CO6 | BT 2 |
| **6** | Write short notes on global data flow analysis | CO6 | BT 1 |
| **7** | What is code motion? Give example | CO6 | BT 1 |
| **8** | What are structure preserving transformations on basic block? | CO6 | BT 1 |
| **9** | What is the use of algebraic identification in optimization of basic blocks? | CO6 | BT 2 |
| **10** | When does dangling reference occur? | CO6 | BT 1 |
| **11** | What are the patterns used for code optimization? | CO6 | BT 1 |
| **12** | Give the block diagram of organization of code optimizer | CO6 | BT 2 |
| **13** | Write short notes on activation tree. | CO6 | BT 1 |
| **14** | How the run-time memory is sub-divided? | CO6 | BT 2 |
| **15** | Give the structure of general activation record | CO6 | BT 2 |
| **16** | What is access link? | CO6 | BT 2 |
| **17** | What is known as environment and state? | CO6 | BT 2 |
| **18** | What is meant by Dead Code? | CO6 | BT 1 |
| **19** | What are the 3 areas of code optimization? | CO6 | BT 2 |
| **20** | List out the properties of reducible flow graph. | CO6 | BT 2 |
| **PART-C** | | | |
| **1** | Explain the transformation of basic blocks and Write in detail about function-preserving transformation | CO6 | BT 1 |
| **2** | Optimize the following code using various optimization techniques: **L3**  i=1,s=0;  for(i=1;i,=3;i++)  for(j=1;j<=3;j++)  c[i][j]-c[i][j]+a[i]a[j]+b[i][j] | CO6 | BT 3 |
| **3** | Write global common sub expresion elimination algorithm with example | CO6 | BT 3 |
| **4** | Explain in detail about stack based allocation of space. Justify use of activation trees and activation records in stack based allocation. | CO6 | BT 1 |
| **5** | Write a simple program to implement a sort procedure. Draw the activation tree when the numbers 9.8.7.6.5.4.3.2.1 are sorted. | CO6 | BT 2 |
| **6** | Explain about various parameter passing methods in procedure call. | CO6 | BT 1 |
| **7** | Describe in detail about the stack allocation in memory management | CO6 | BT 2 |
| **8** | What are different storage allocation strategies? Explain. | CO6 | BT 1 |
| **9** | Explain the data flow analysis concept with suitable example. | CO6 | BT 1 |

**Note:**

1. **BT Level –** Blooms Taxonomy Level
2. **CO – Course Outcomes**

BT1 – Remember BT2 – Understand BT3 – Apply BT4 – Analyze BT5 – Evaluate BT6 – Create