# Check Your Knowledge (1)

**Optimization, CFA:**

1. Explain graphs that are used in program analysis.
2. Which optimizing transformations need analysis of execution pathes?
3. Which optimizing transformations do not need analysis of execution pathes?
4. Give an example for a pair of transformations such that one enables the other.
5. Define the control-flow graph. Describe transformations on the CFG.
6. Define the dominator relation. What is it used for?
7. Describe an algorithm for computing dominator sets.
8. Define natural loops.
9. What is the role of the loop header and of the pre-header.
10. Show a graph that has a cycle but no natural loop.
11. Define induction variables, and explain the transformation technique.



## Lecture Compilation Methods SS 2011 / Slide 601

**Objectives:**

Support repetition and understanding of the material

**In the lecture:**

- Answer some questions:
- Let some questions be answered.

# Check Your Knowledge (2)

**Optimization, DFA:**

12. Describe the schema for DFA equations for the four problem categories.
13. Explain the relation of the meet operator, the paths in the graph, and the DFA solutions.
14. Describe the DFA problem reaching definitions.
15. Describe the DFA problem live variables.
16. Describe the DFA problem available expressions.
17. Describe the DFA problem copy propagation.
18. Describe the DFA problem constant propagation.
19. Describe the iterative DFA algorithm; its termination; its complexity.
20. Describe an heuristic improvement of the iterative DFA algorithm.
21. Extend constant propagation to interval propagation for bounds checks. Explain the interval lattice.
22. What is the role of lattices in DFA?
23. Describe lattices that are common for DFA.



## Lecture Compilation Methods SS 2011 / Slide 602

**Objectives:**

Support repetition and understanding of the material

**In the lecture:**

- Answer some questions:
- Let some questions be answered.

# Check Your Knowledge (3)

**Object Oriented Program Analysis:**

24. Describe techniques to reduce the number of arcs in call graphs.
25. Describe call graphs for object oriented programs.
26. Describe techniques to reduce the number of arcs in object oriented call graphs.

**Code Generation, Storage mapping:**

27. Explain the notions of storage classes, relative addresses, alignment, overlay.
28. Compare storage mapping of arrays by pointer trees to mapping on contiguous storage.
29. Explain storage mapping of arrays for C. What is different for C, for Fortran?
30. For what purpose are array descriptors needed? What do they contain?
31. What is the closure of a function? In which situation is it needed?
32. Why must a functional parameter in Pascal be represented by a pair of pointers?
33. What does an activation record contain?
34. Explain static links in the run-time stack. What is the not-most-recent property?
35. How do C, Pascal, and Modula-2 ensure that the run-time stack discipline is obeyed?
36. Why do threads need a separate run-time stack each?



## Lecture Compilation Methods SS 2011 / Slide 603

**Objectives:**

Support repetition and understanding of the material

**In the lecture:**

- Answer some questions:
- Let some questions be answered.

# Check Your Knowledge (4)

37. Explain the code for function calls in relation to the structure of activation records.
38. Explain addressing relative to activation records.
39. Explain sequences for loops.
40. Explain the translation of short circuit evaluation of boolean expressions. Which attributes are used?
41. Explain code selection by covering trees with translation patterns.
42. Explain a technique for tree pattern selection using 3 passes.
43. Explain code selection using parsing. What is the role of the grammar?

## Register Allocation

44. How is register windowing used for implementation of function calls?
45. Which allocation technique is applied for which program context?
46. Explain register allocation for expression trees. Which attributes are used?
47. How is spill code minimized for expression trees?
48. Explain register allocation for basic blocks? Relate the spill criteria to paging techniques.
49. Explain register allocation by graph coloring. What does the interference graph represent?
50. Explain why DFA life-time analysis is needed for register allocation by graph coloring.



## Lecture Compilation Methods SS 2011 / Slide 604

**Objectives:**

Support repetition and understanding of the material

**In the lecture:**

- Answer some questions:
- Let some questions be answered.

# Check Your Knowledge (5)

## Instruction Scheduling

51. What does instruction scheduling mean for VLIW, pipeline, and vector processors?
52. Explain the kinds of arcs of DDGs (flow, anti, output).
53. What are loop carried dependences?
54. Explain list scheduling for parallel FUs. How is the register need modelled? Compare it to Belady's register allocation technique.
55. How is list scheduling applied for arranging instructions for pipeline processors?
56. Explain the basic idea of software pipelining. What does the initiation interval mean?

## Loop Parallelization

57. Explain dependence vectors in an iteration space. What are the admissible directions for sequential and for parallelized innermost loops?
58. What is tiling, what is scaling?
59. Explain SRP transformations.
60. How are the transformation matrices used?
61. How are loop bounds transformed?
62. Parallelize the inner loop of a nest that has dependence vectors (1,0) and (0, 1)?



**Lecture Compilation Methods SS 2011 / Slide 605**

**Objectives:**

Support repetition and understanding of the material

**In the lecture:**

- Answer some questions:
- Let some questions be answered.