- Distribution of tasks in phases of a typical compiler (scanning, parsing, semantics, optimization, code gen, etc.)
    - **19au1, 18au1, 15wi1**
- Static semantics and type checking
    - **18au3.d, 18sp3.d, 15wi2.c**
- Codegen
    - **21au1, 21au2.e, 19au2, 18au2, 18au3.e, 18sp2, 18sp3.e, 15wi2.d, 15wi3**
- Runtime storage organization
    - Representation of scalars, arrays, objects
        - **11au2 (sort of)**
    - Address space layout: code, static data, stack, heap
- Object representation
    - Data layout
    - Object creation - new
        - **17wi1**
    - Inheritance and method overriding
    - Method invocation using dynamic dispatch (vtables)
        - **18sp1, 15wi3**
- Optimization
    - Scope of optimizations: peephole, local, global, interprocedural; tradeoffs between very local vs more global analysis/optimization
    - Basics of dataflow analysis (def, use, in, and out sets); be able to solve simple problems
        - **21au4, 18au5, 19au4, 19au5, 18sp5**
    - Dominators and dominance frontiers; be able to figure these out in a flowgraph.
        - **21au5, 18au6.a, 19au6, 18sp6.a, 17wi5**
    - SSA: be able to translate a simple flowgraph into SSA form (informally; you don't need to exactly implement any particular algorithm but you should be able to place all necessary phi functions appropriately)
        - **21au6, 18au6.b, 18sp6.b, 15wi6**
    - Examples of some common optimizations and how dataflow or SSA analysis reveals when these are possible (e.g., common subexpressions, live variables, constant folding).
        - **21au3, 18au4, 18sp4**
- Major backend issues - not in detail, but know the major ideas and key algorithms discussed in class
    - Instruction selection & tree pattern matching
    - Instruction scheduling & list scheduling
        - **18au8, 19au8, 18sp7, 17wi7, 15wi5**
    - Register allocation & allocation by graph coloring
        - **21au6, 18au7, 19au7, 17wi8, 15wi4**
- Garbage collection - general ideas, not details.
    - **17wi9, 15wi7**
    - Basic notions of liveness, reachability
    - Reasons behind strategies like compacting and generational collectors