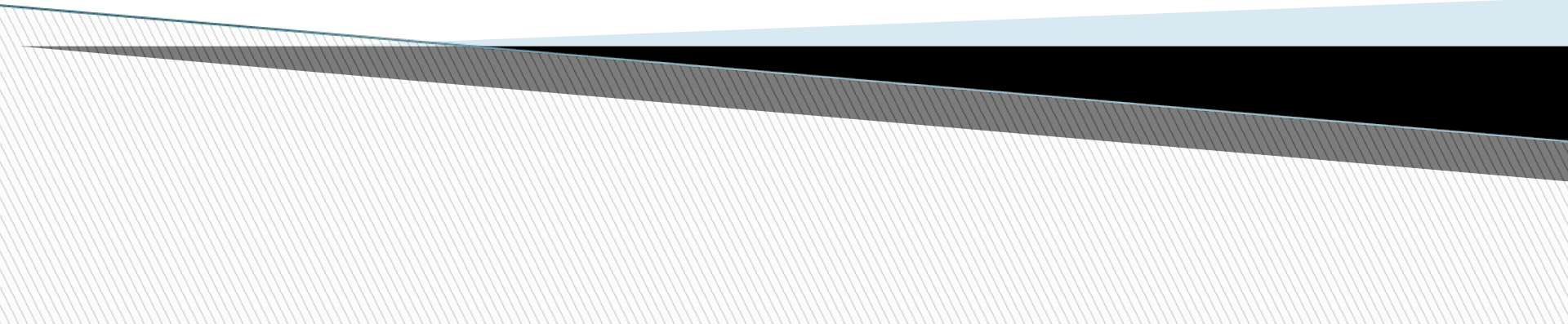
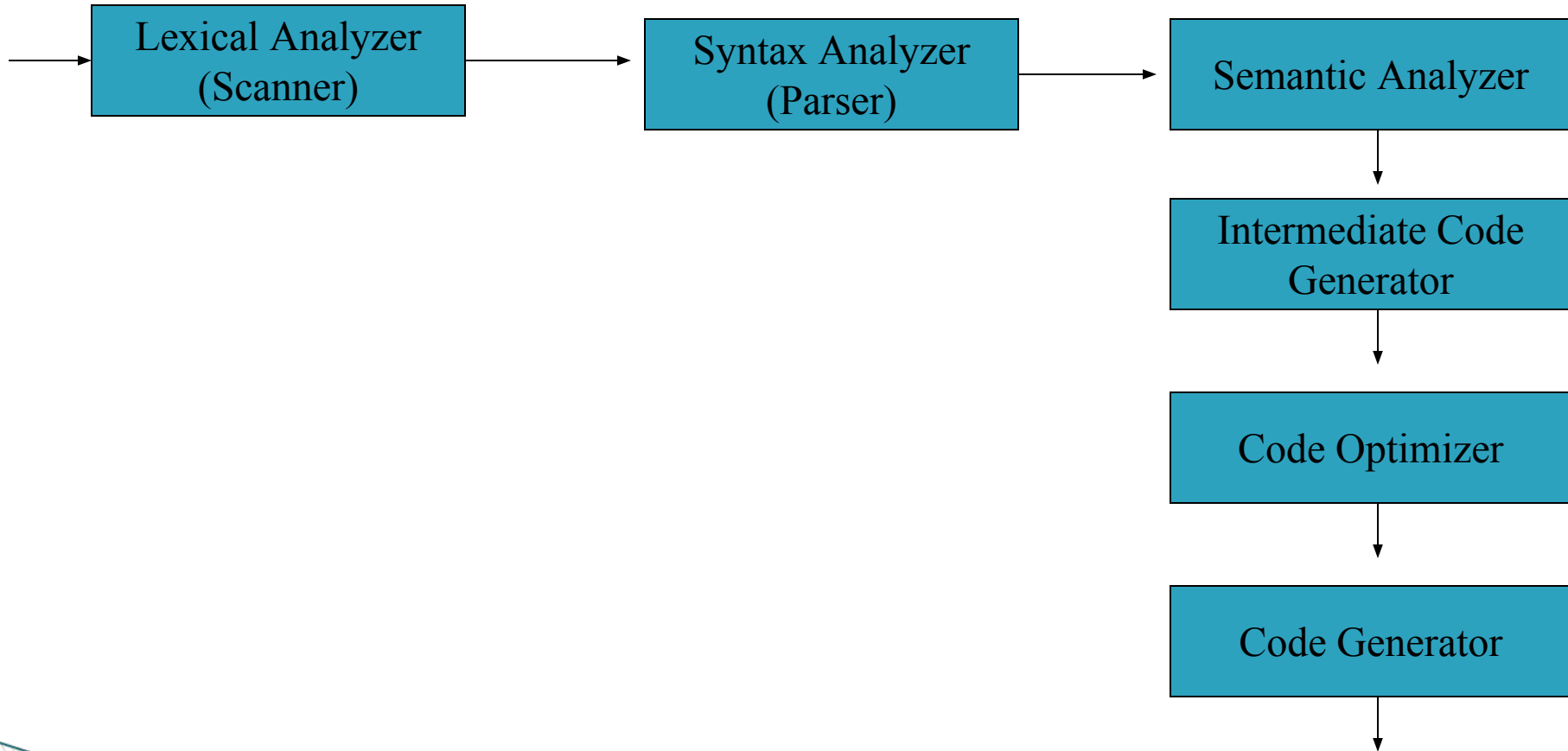


Code Generator

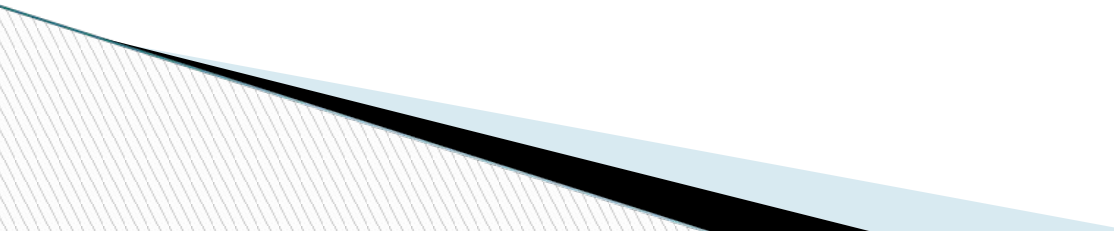




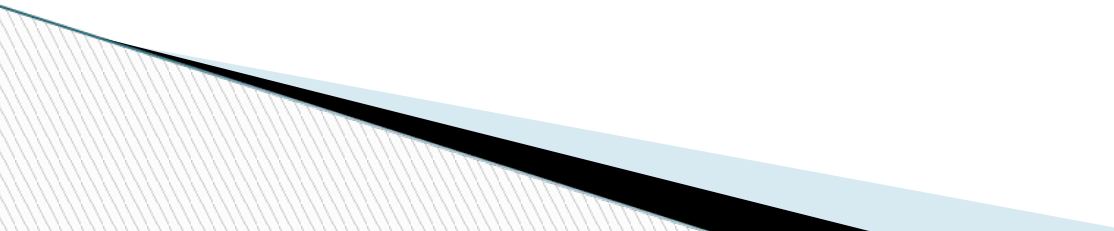
Code Generator

- The final phase in compiler model is the code generator. It takes as input an intermediate representation of the source program and produces as output an equivalent target program.

Issues in the design of a code generator

- Input to the code generator
 - Target program
 - Memory management
 - Instruction selection
 - Register allocation
 - Evaluation order
- 

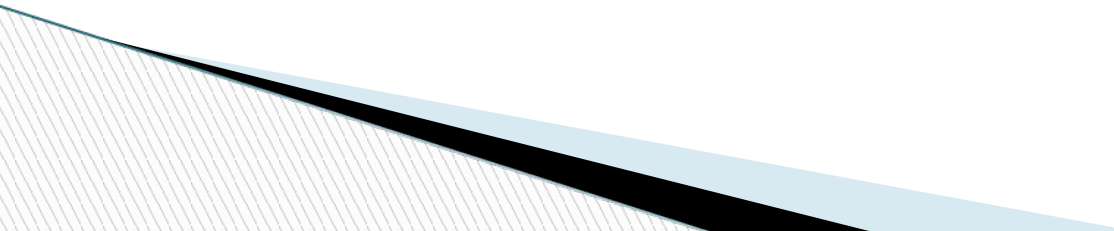
Input to the code generator

- The input to the code generator contains the intermediate representation of the source program and the information of the symbol table. The source program is produced by the front end.
 - Intermediate representation has the several choices: Postfix notation, Syntax tree, Three address code
 - The code generation phase needs complete error-free intermediate code as an input requires.
- 

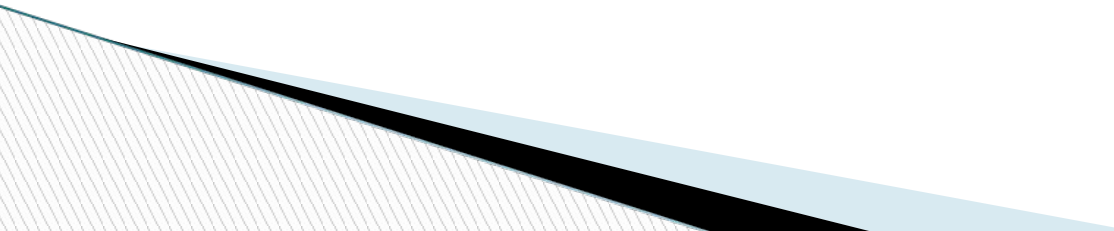
Target program

- The target program is the output of the code generator.
The output can be:
 - a) **Assembly language:** It allows subprogram to be separately compiled.
 - b) **Relocatable machine language:** It makes the process of code generation easier.
 - c) **Absolute machine language:** It can be placed in a fixed location in memory and can be executed immediately.

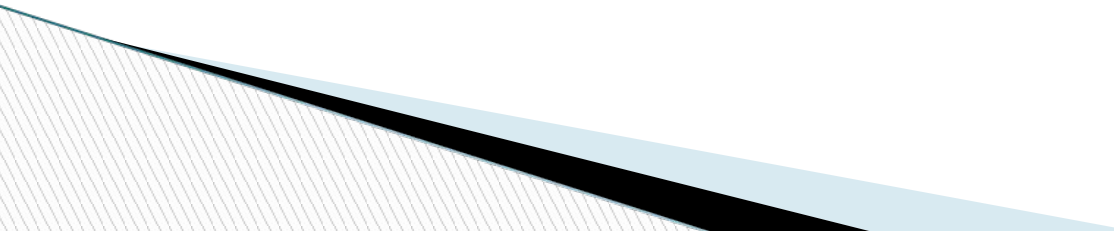
Memory management

- ❑ Nature of instruction set of the target machine should be complete and uniform.
 - ❑ When you consider the efficiency of target machine then the instruction speed and machine idioms are important factors.
 - ❑ The quality of the generated code can be determined by its speed and size.
- 

Instruction selection

- ❑ The instructions of target machine should be complete and uniform.
 - ❑ Instruction speeds and machine idioms are important factors when efficiency of target program is considered.
 - ❑ The quality of the generated code is determined by its speed and size.
- 

Register allocation

- ❑ Register can be accessed faster than memory. The instructions involving operands in register are shorter and faster than those involving in memory operand.
 - ❑ The following sub problems arise when we use registers:
 - ❑ **Register allocation:** In register allocation, we select the set of variables that will reside in register.
 - ❑ **Register assignment:** In Register assignment, we pick the register that contains variable.
 - ❑ Certain machine requires even-odd pairs of registers for some operands and result.
- 

Evaluation order

- The efficiency of the target code can be affected by the order in which the computations are performed. Some computation orders need fewer registers to hold results of intermediate than others.

