Welcome to CS143: Compilers

- Course Information
- Why Study Compilers?
- A Quick History of Compilers
- The Structure of a Compiler

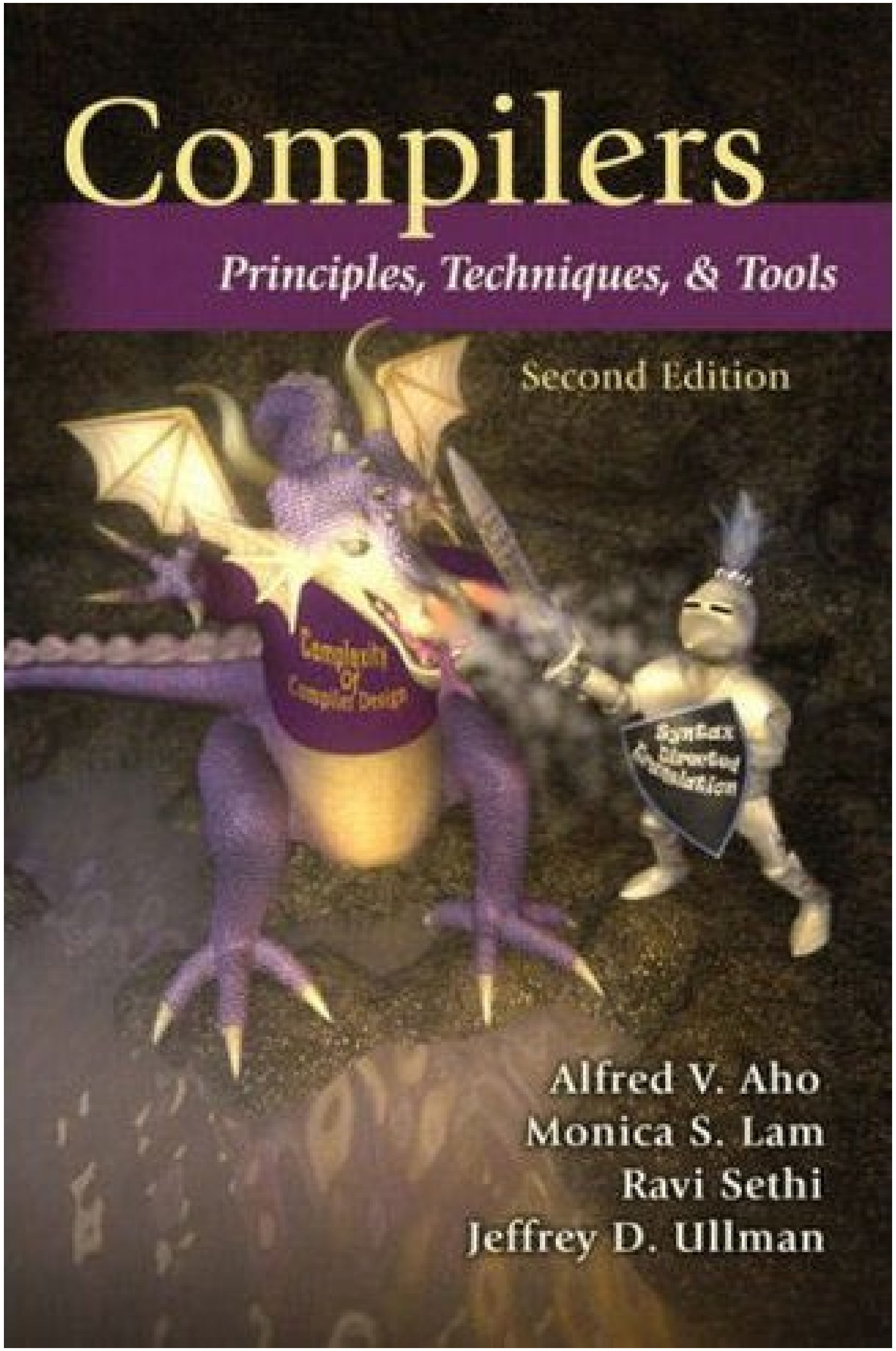
Course Staff

Instructor: Keith Schwarz
(htiek@cs.stanford.edu)

TA: Jinchao Ye
(jcye@stanford.edu)

TA: Naran Bayanbat
(narab@stanford.edu)

http://cs143.stanford.edu

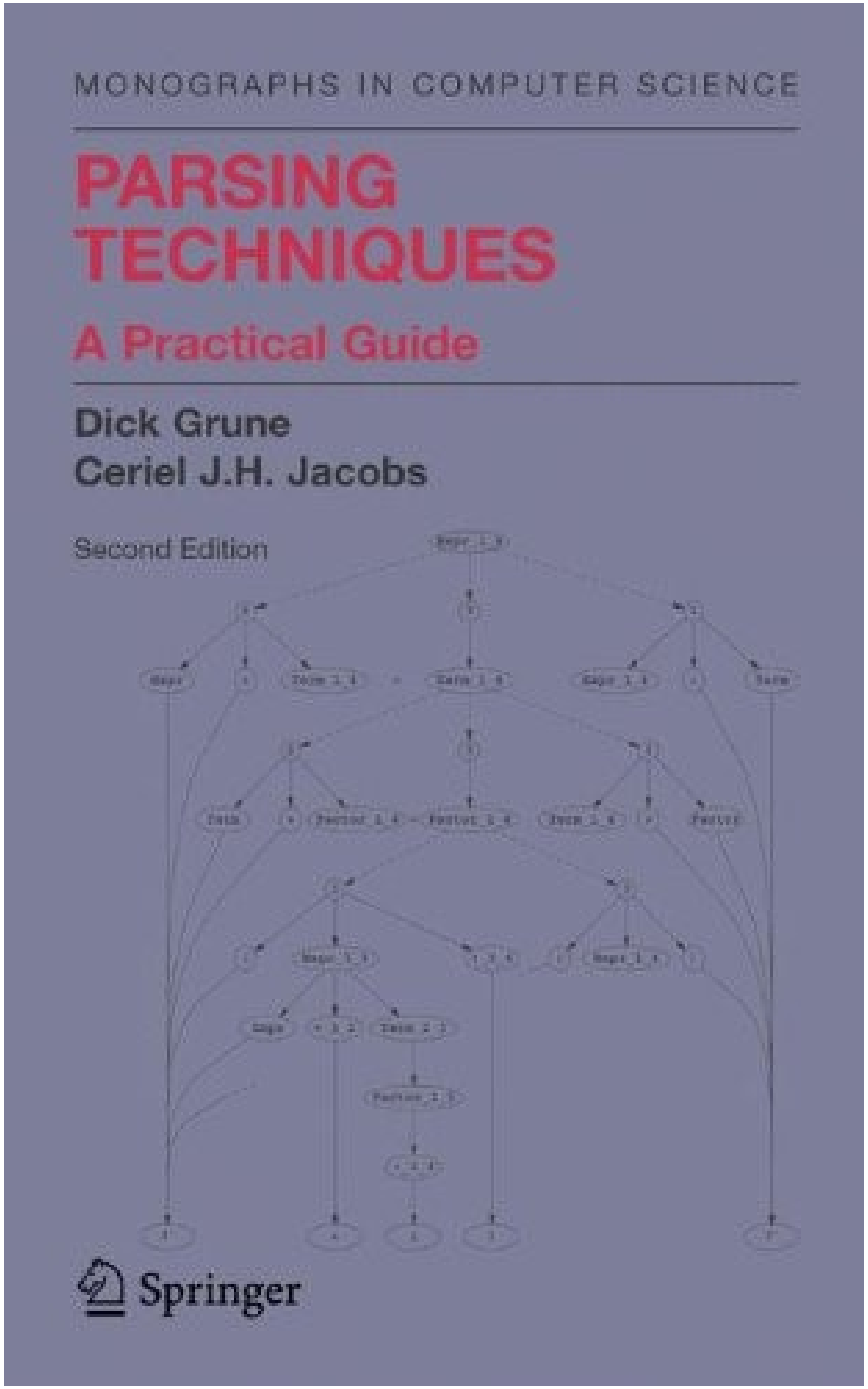


Compilers

Principles, Techniques, & Tools

Second Edition

Alfred V. Aho
Monica S. Lam
Ravi Sethi
Jeffrey D. Ullman



MONOGRAPHS IN COMPUTER SCIENCE

PARSING TECHNIQUES

A Practical Guide

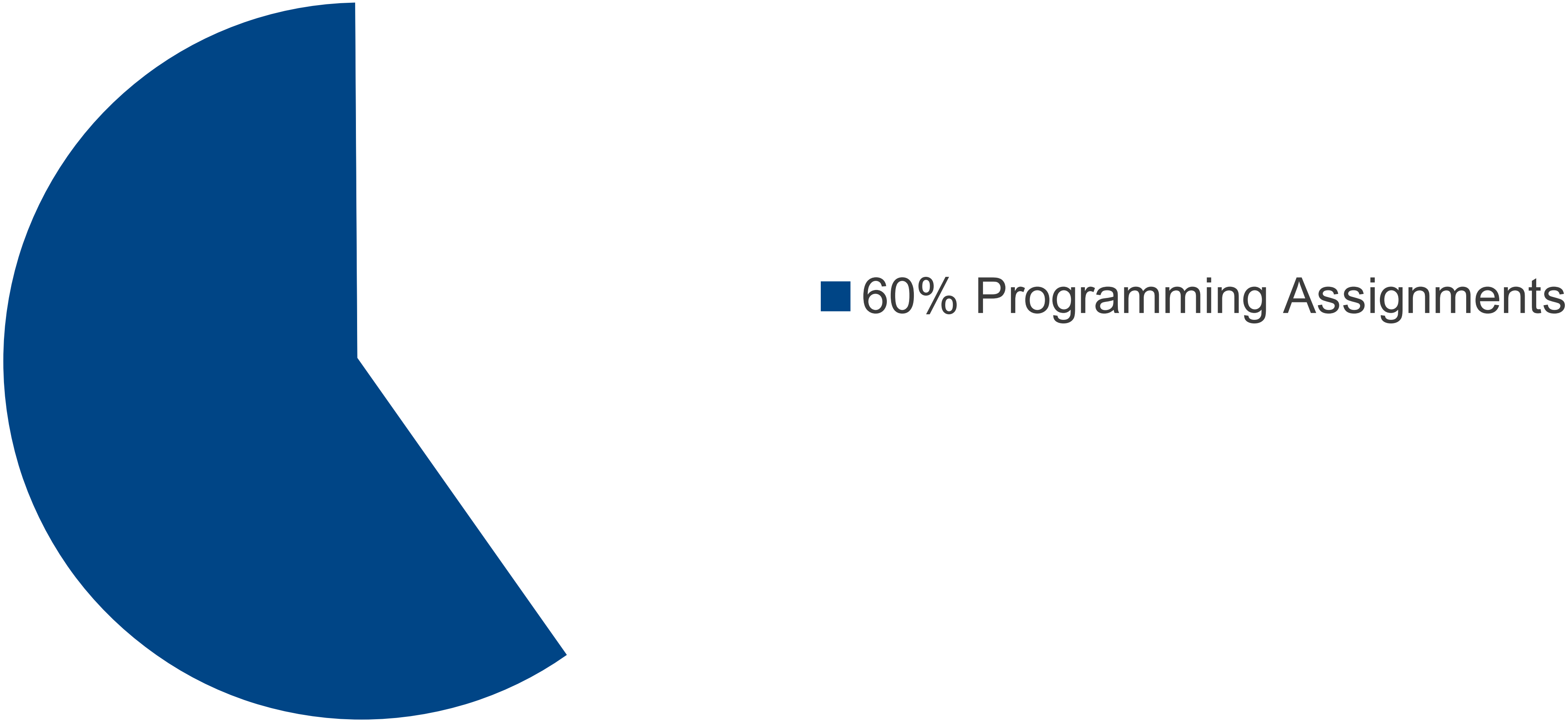
Dick Grune
Ceriel J.H. Jacobs

Second Edition

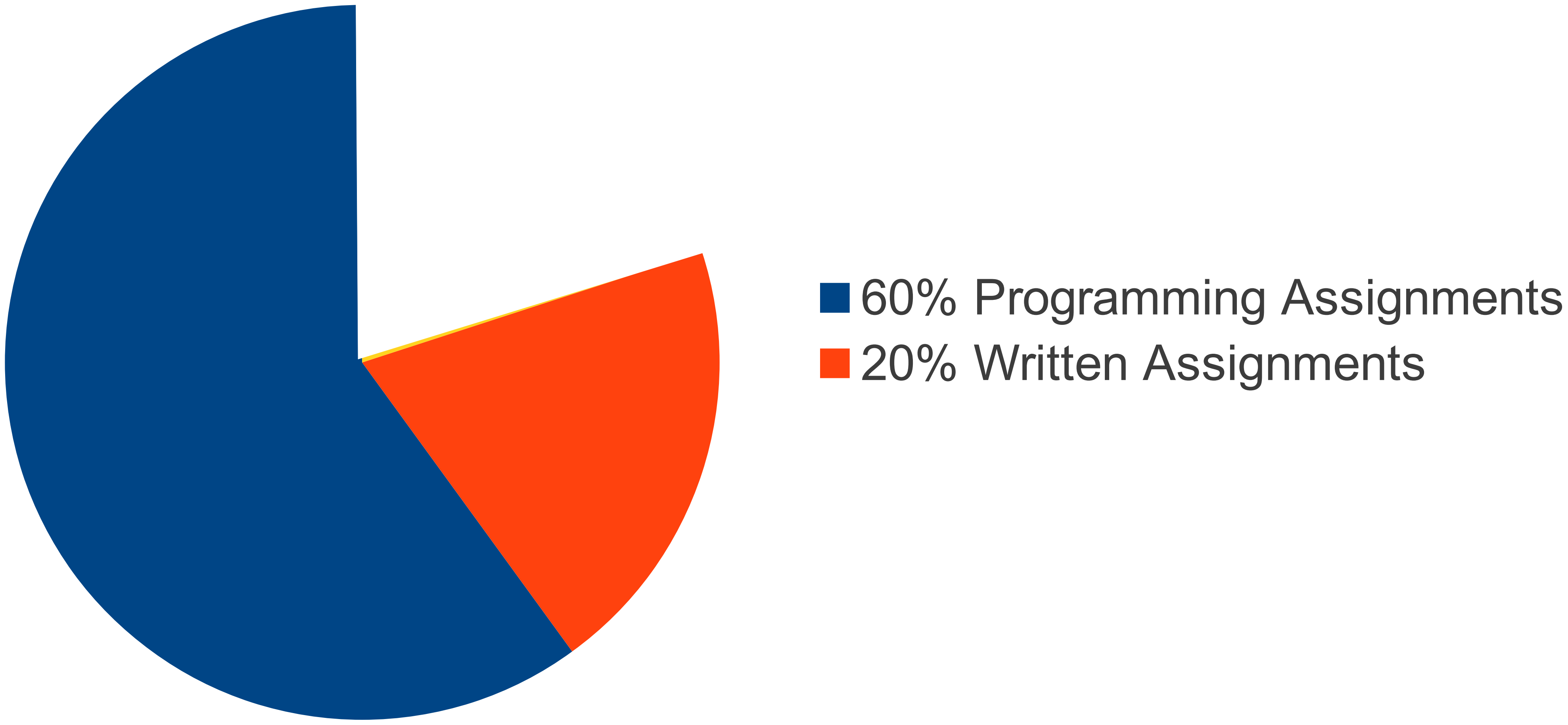
Springer

Grading Policies

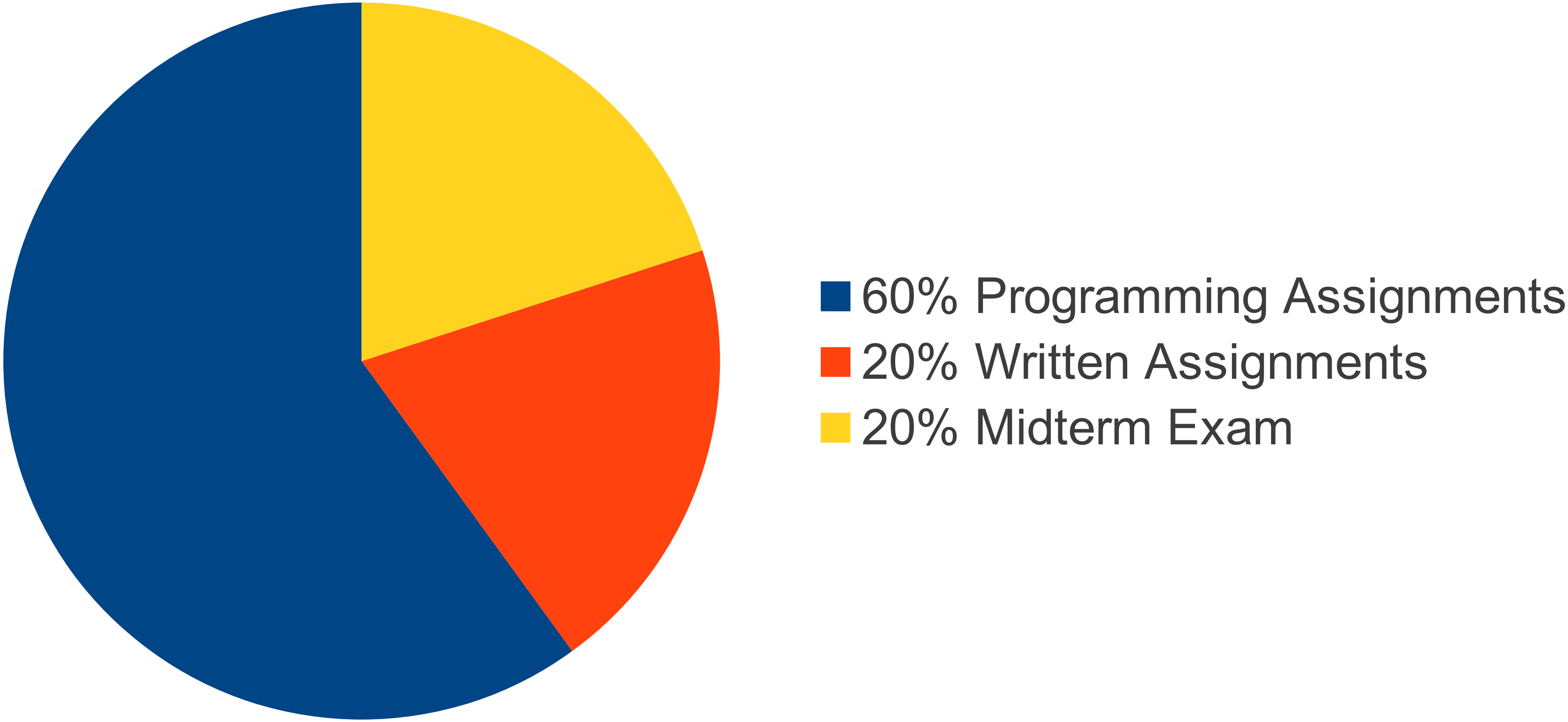
Grading Policies



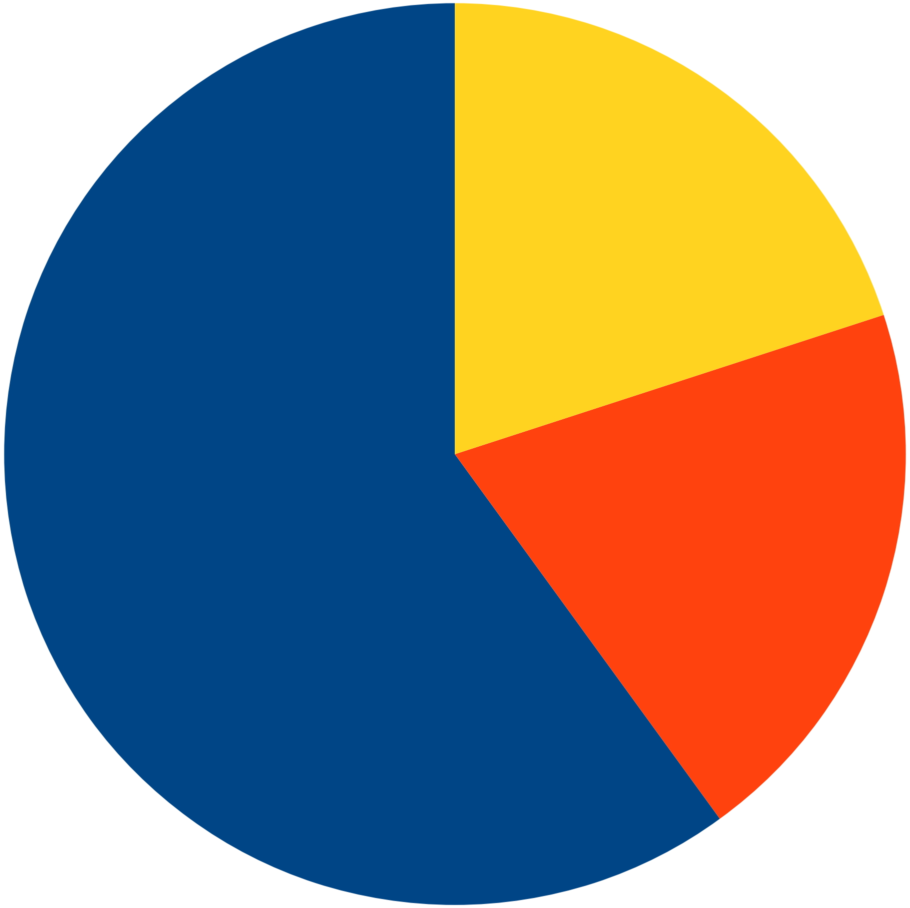
Grading Policies



Grading Policies



Grading Policies



Midterm exam:
July 25,
11:00AM – 1:00PM,
Location TBA

A Word on the Honor Code…

Prerequisites

CS107

CS103

Why Study Compilers?

- Build a **large, ambitious software system**.
- See theory **come to life**.
- Learn how to **build programming languages**.
- Learn **how programming languages work**.
- Learn **tradeoffs in language design**.

A Short History of Compilers

- First, there was nothing.
- Then, there was machine code.
- Then, there were assembly languages.
- Programming expensive; 50% of costs for machines went into programming.

High-Level Languages

Image: http://upload.wikimedia.org/wikipedia/commons/thumb/5/55/Grace_Hopper.jpg/300px-Grace_Hopper.jpg
http://www.nytimes.com/2007/03/20/business/20backus.html

High-Level Languages

Rear Admiral **Grace Hopper**, inventor of A-0, COBOL, and the term "compiler."

Image: http://upload.wikimedia.org/wikipedia/commons/thumb/5/55/Grace_Hopper.jpg/300px-Grace_Hopper.jpg
http://www.nytimes.com/2007/03/20/business/20backus.html

High-Level Languages

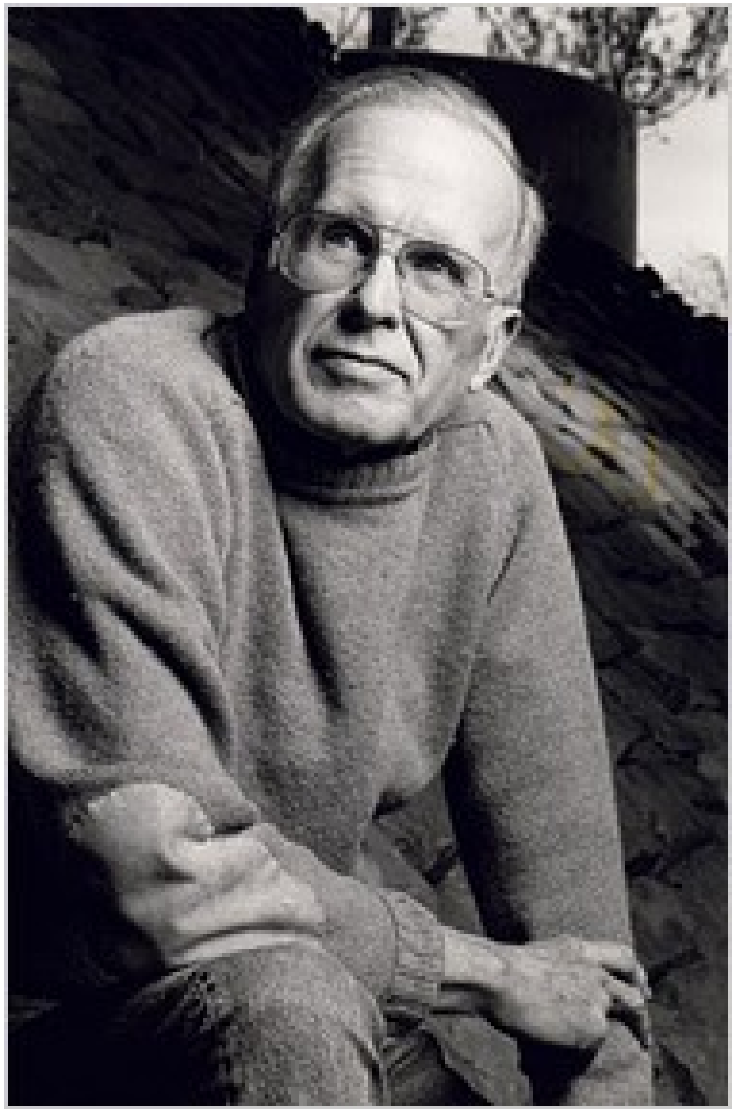
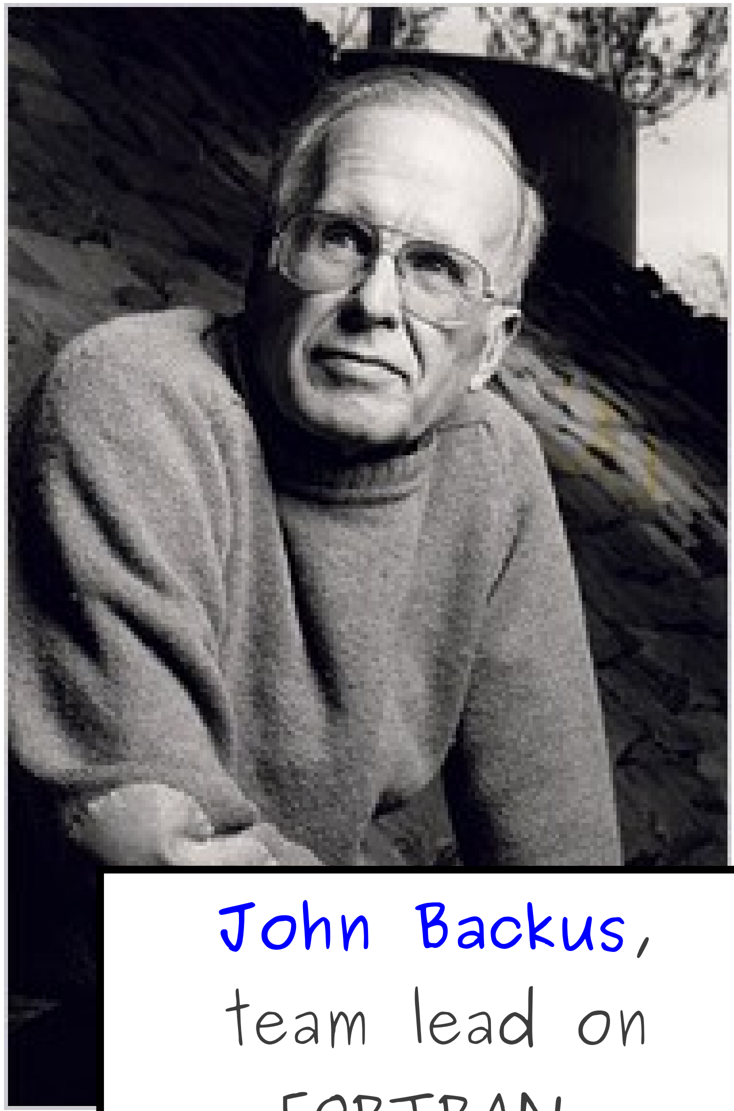


Image: http://upload.wikimedia.org/wikipedia/commons/thumb/5/55/Grace_Hopper.jpg/300px-Grace_Hopper.jpg
http://www.nytimes.com/2007/03/20/business/20backus.html

High-Level Languages

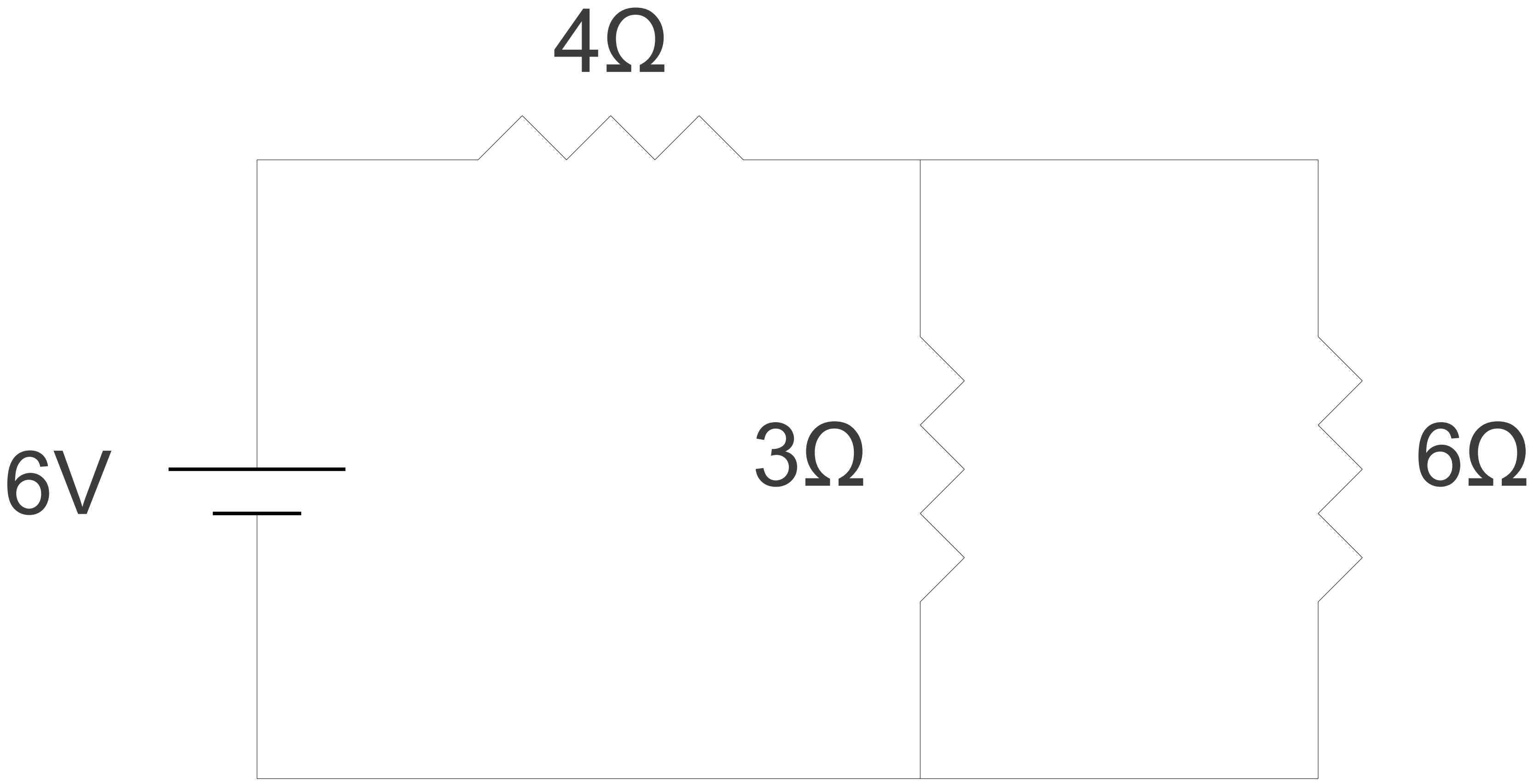


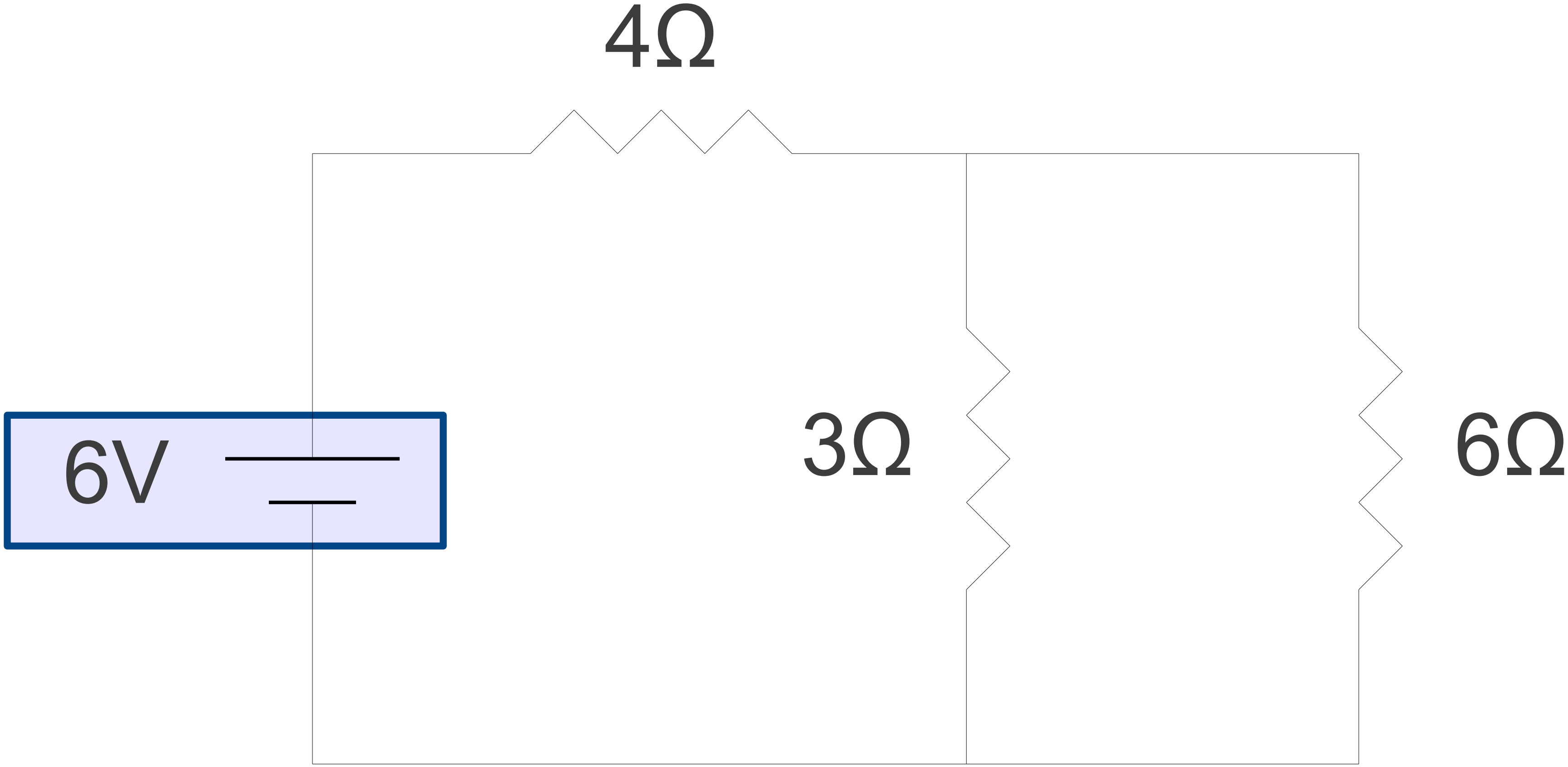
John Backus, team lead on FORTRAN.

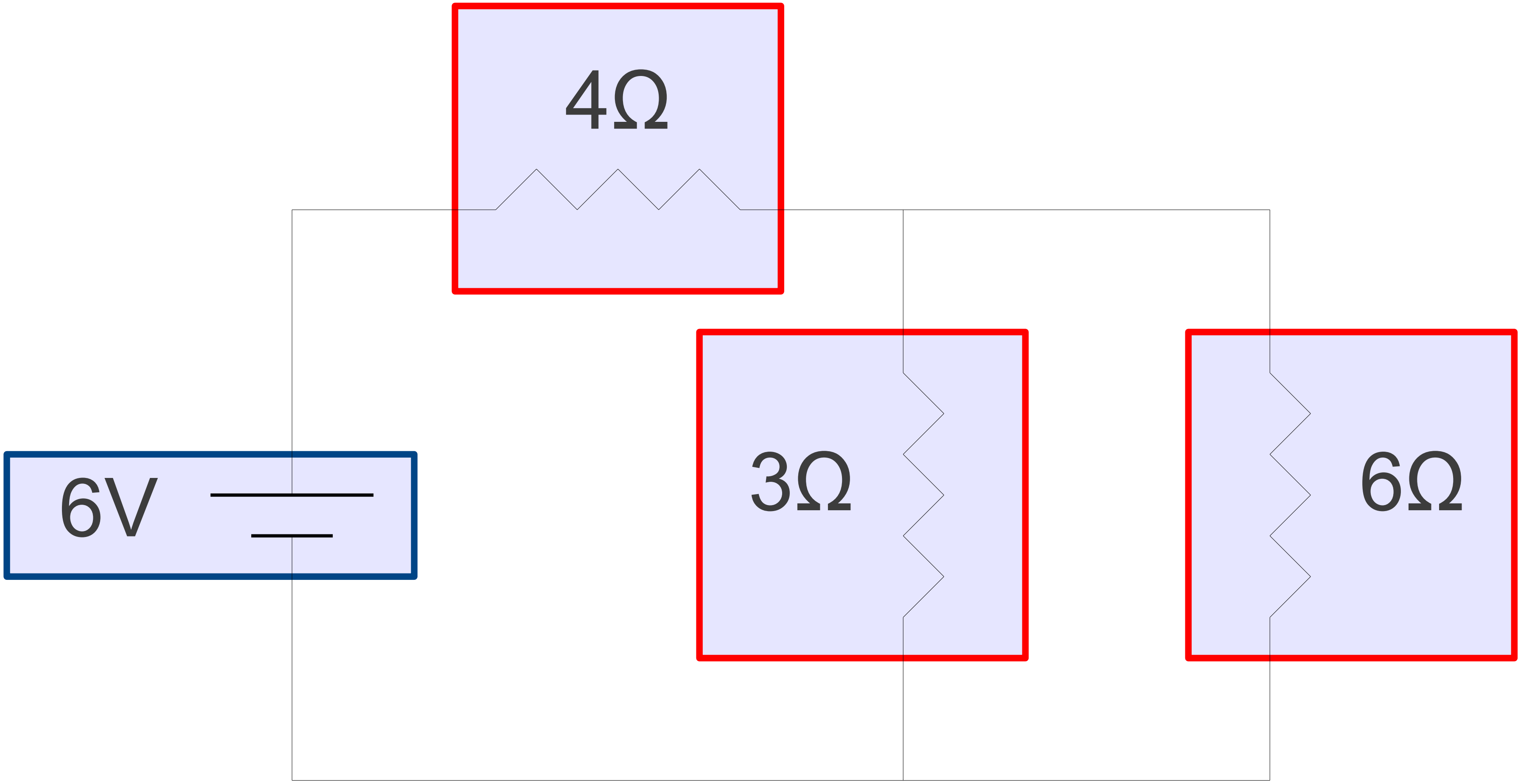
Image: http://upload.wikimedia.org/wikipedia/commons/thumb/5/55/ er.jpg

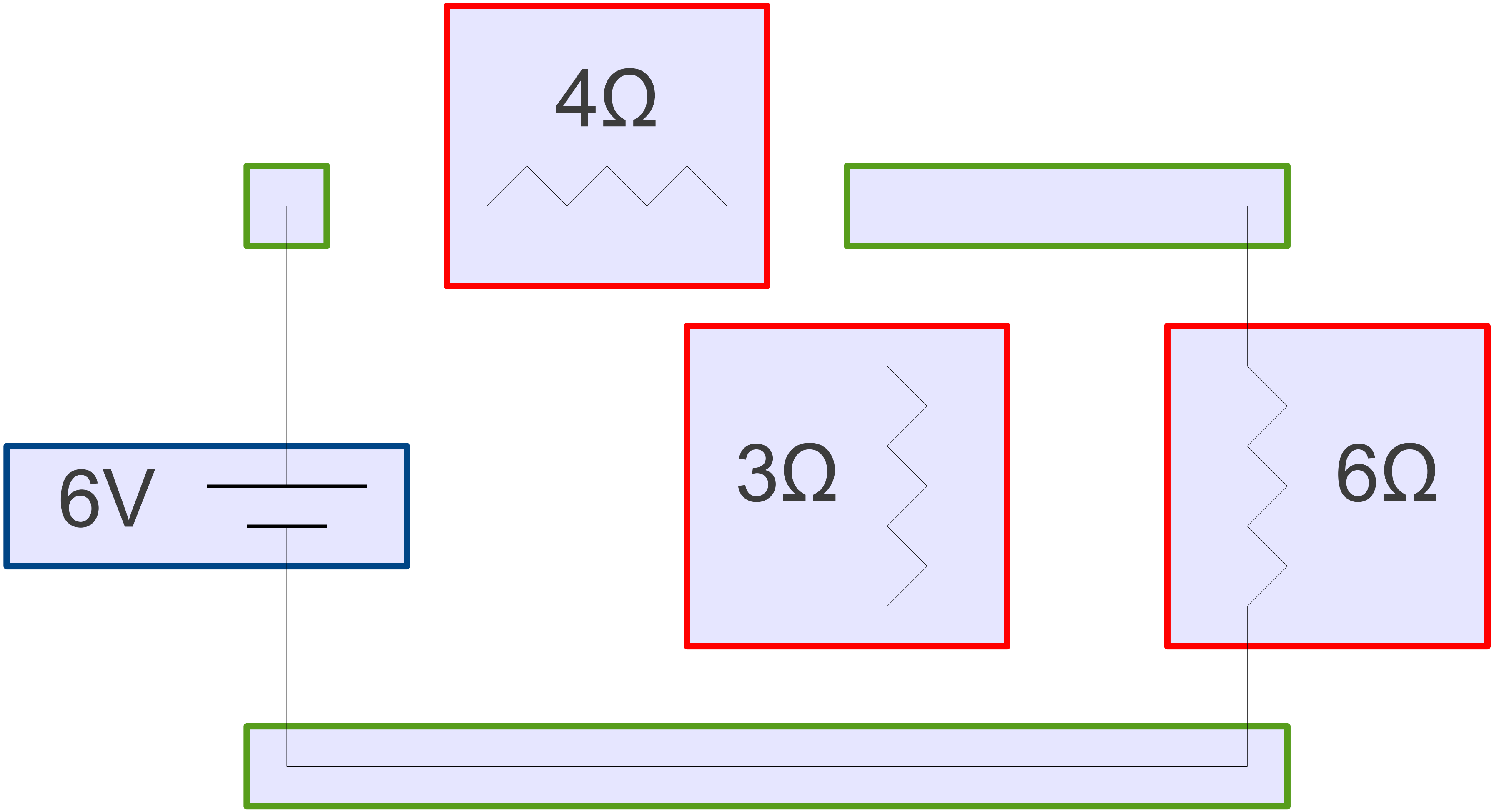
http://www.nytimes.com/2007/03/20/business/20backus.html

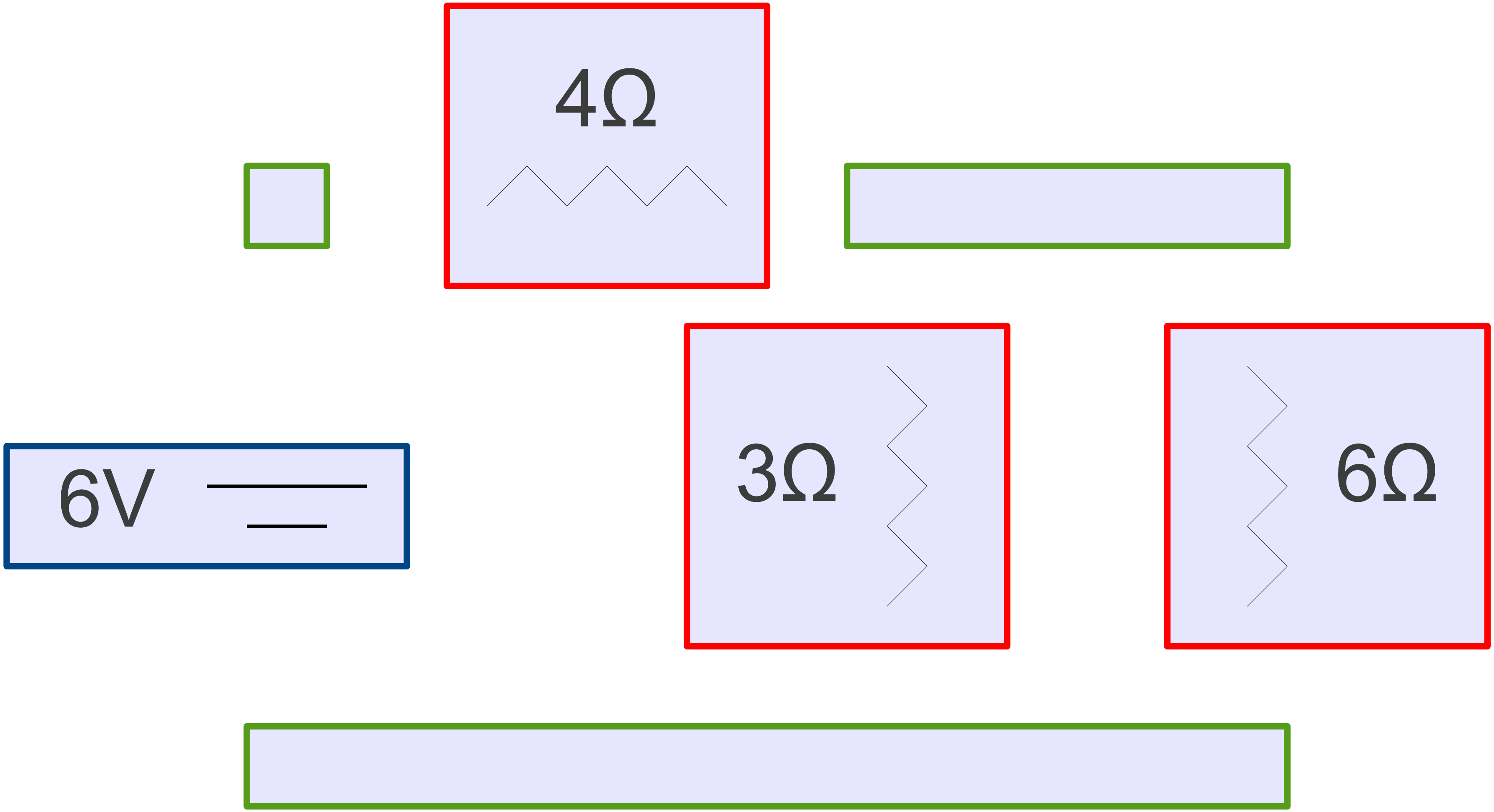
How does a compiler work?

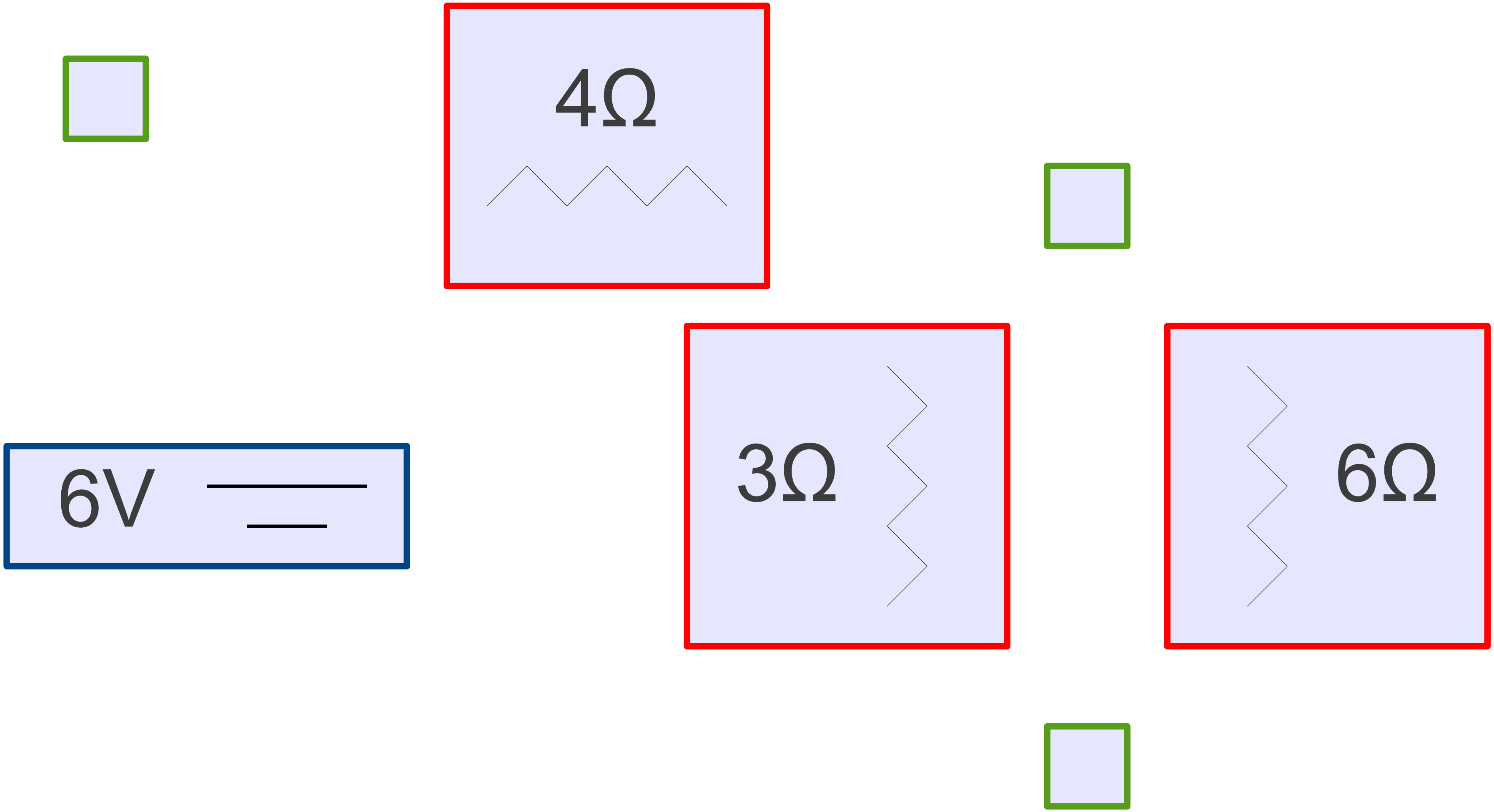

4Ω
6V
3Ω
6Ω

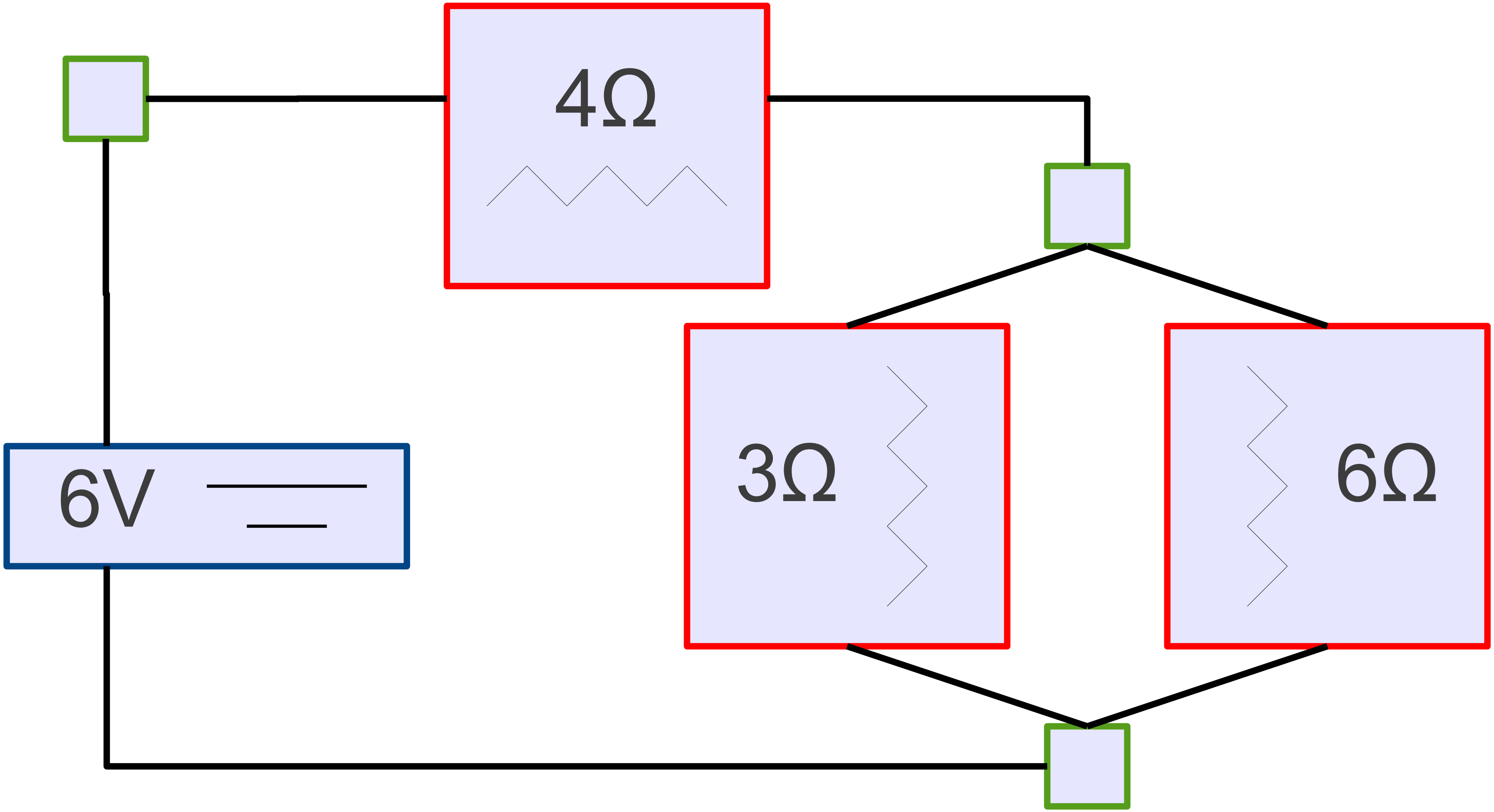

4Ω
6V
3Ω
6Ω

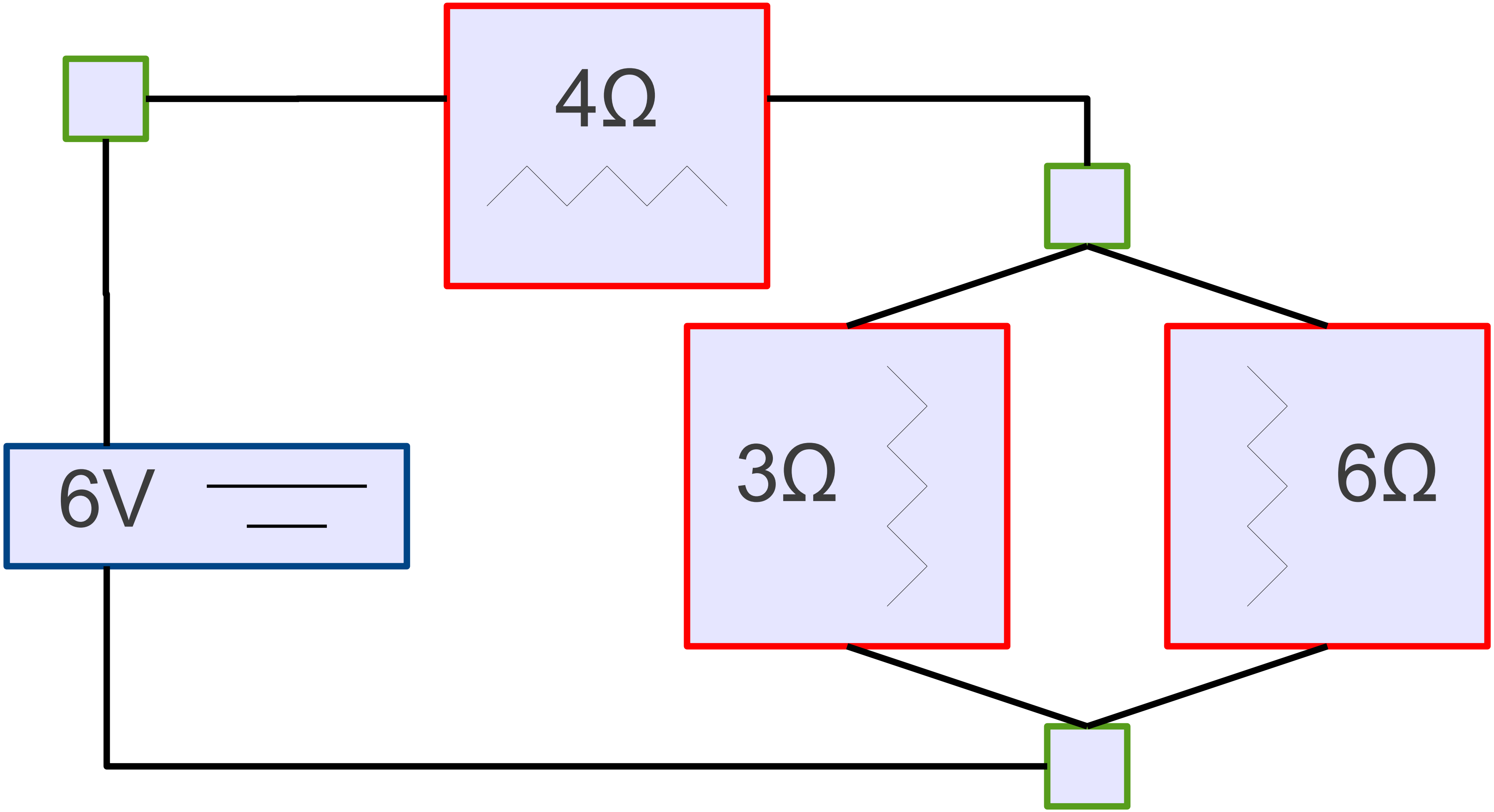

4Ω
6V
3Ω
6Ω


4Ω
6V
3Ω
6Ω

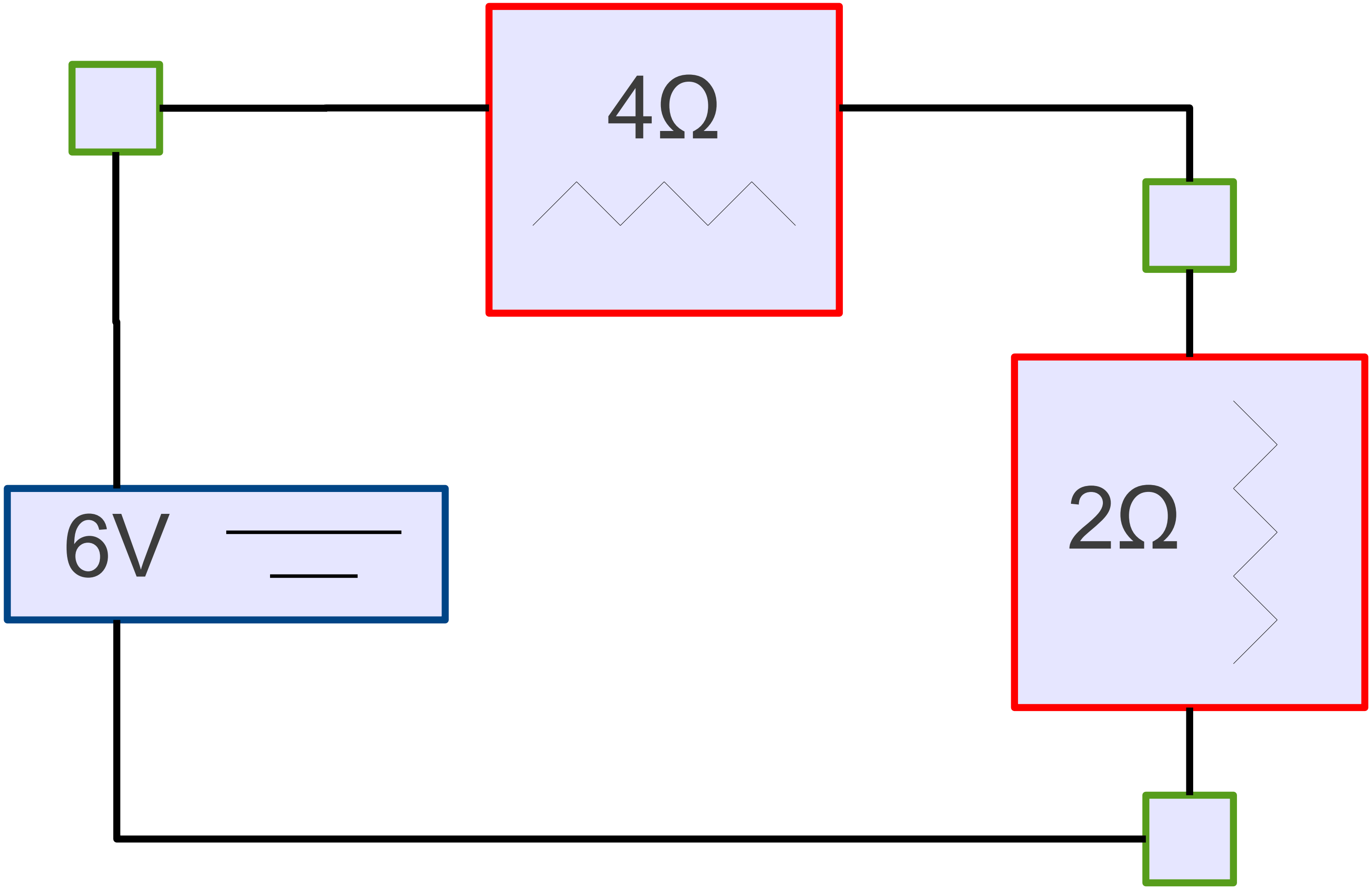

4Ω
6V
3Ω
6Ω


4Ω
6V
3Ω
6Ω

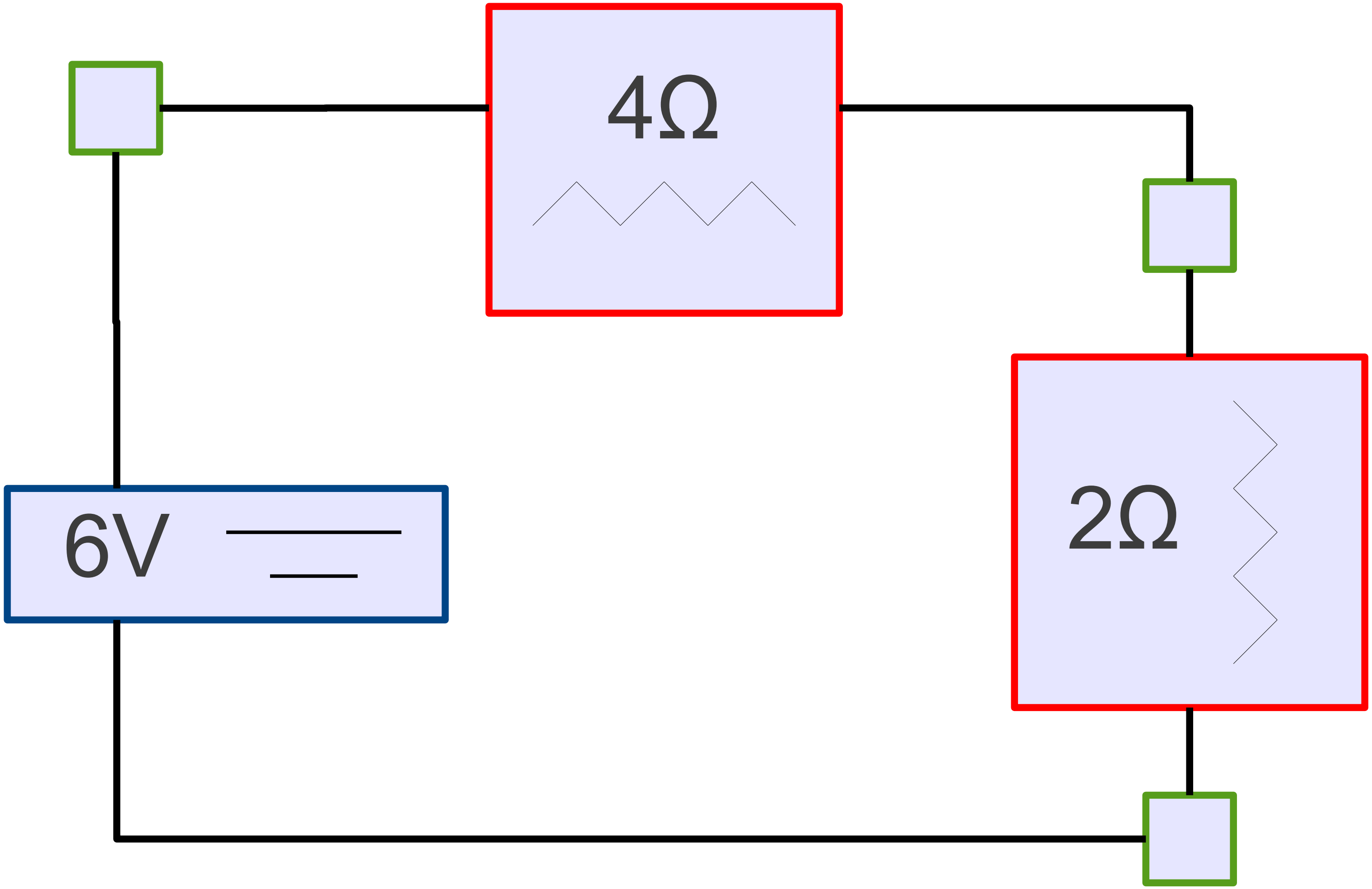

4Ω
6V
3Ω
6Ω

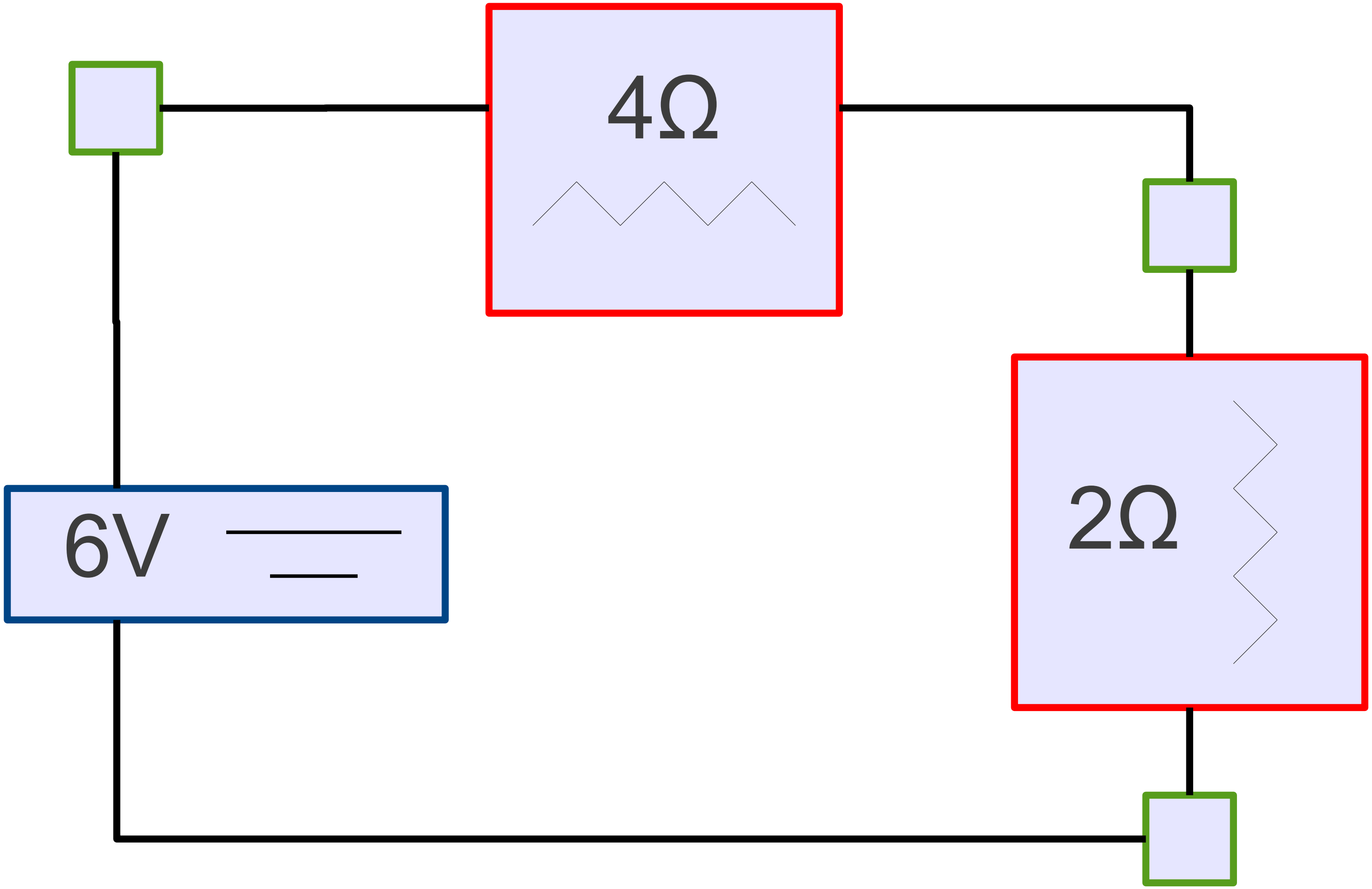


$$\frac{1}{\frac{1}{3\Omega}+\frac{1}{6\Omega}}=2\Omega$$

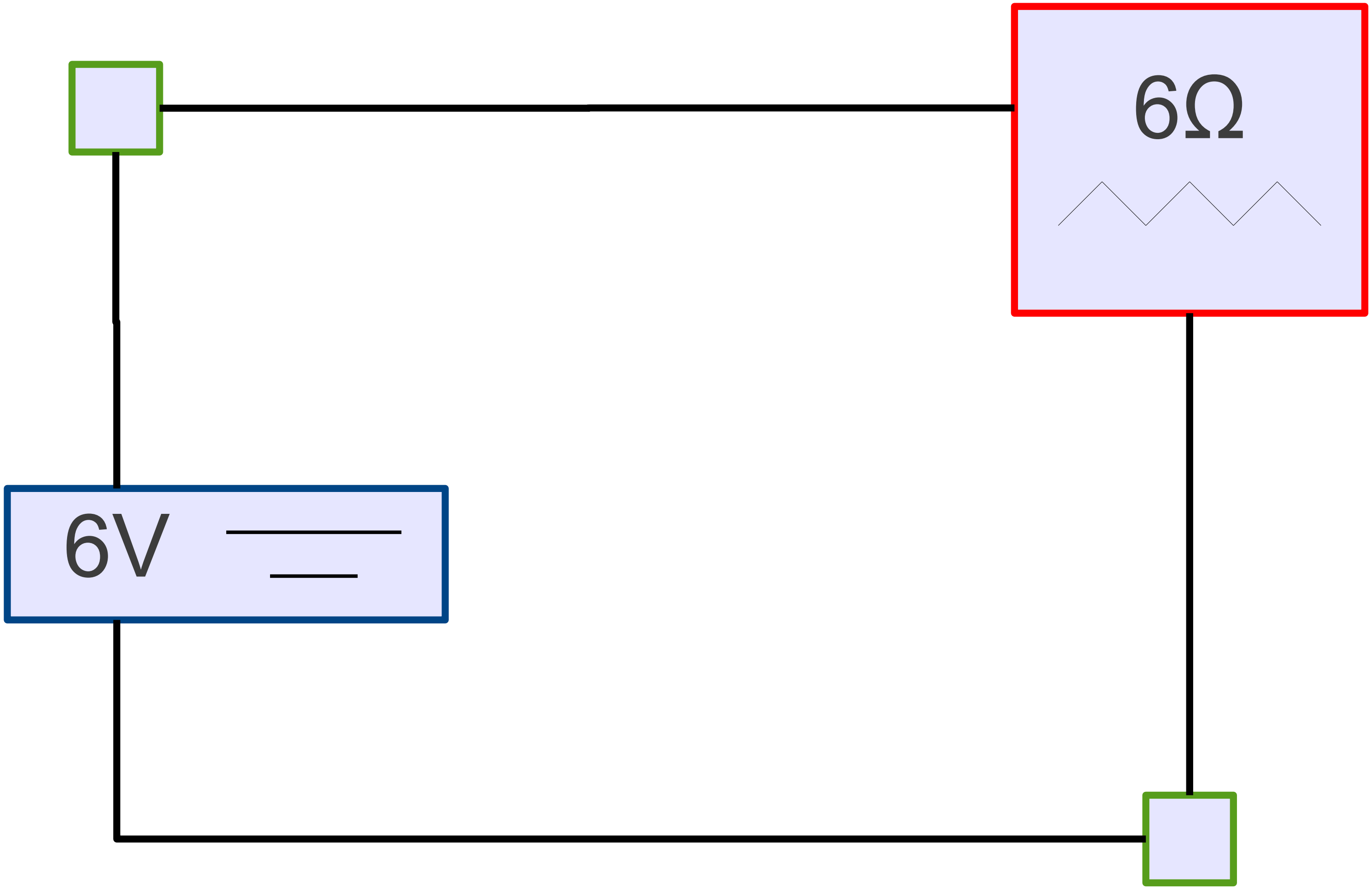


$$\frac{1}{\frac{1}{3\Omega}+\frac{1}{6\Omega}}=2\Omega$$

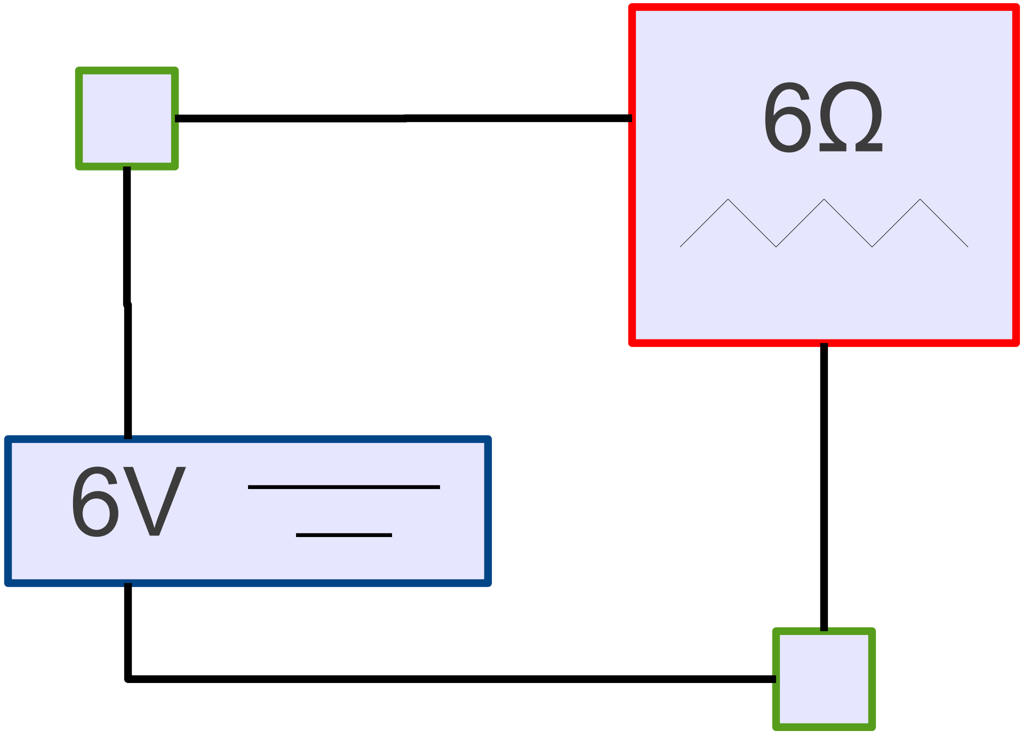

4Ω
6V
2Ω

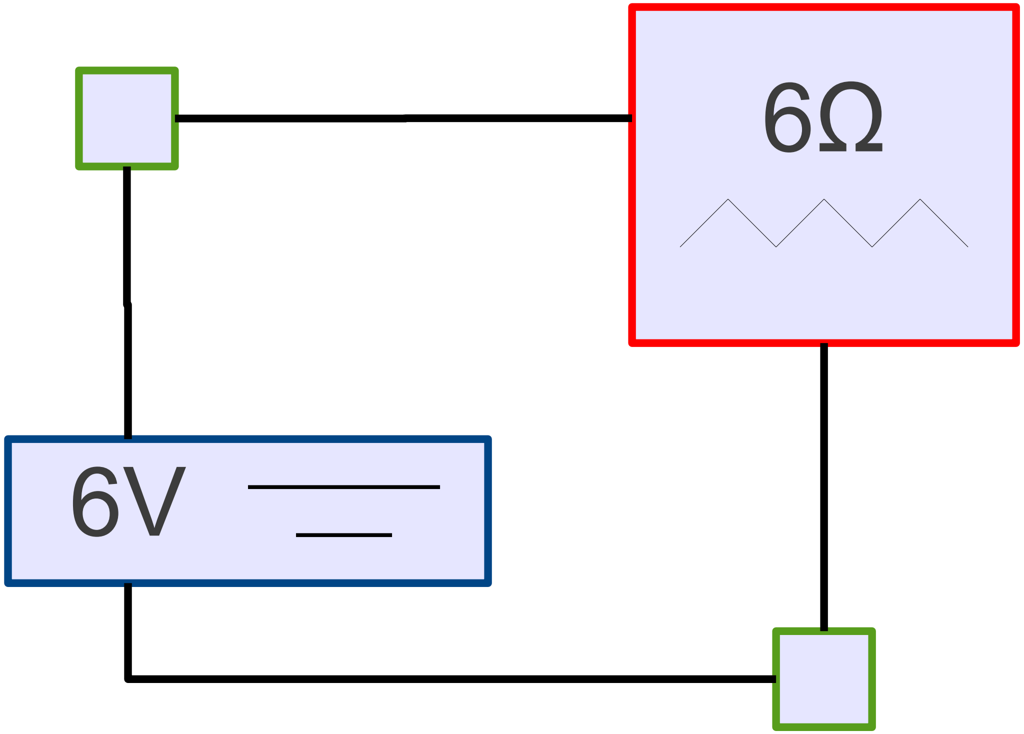


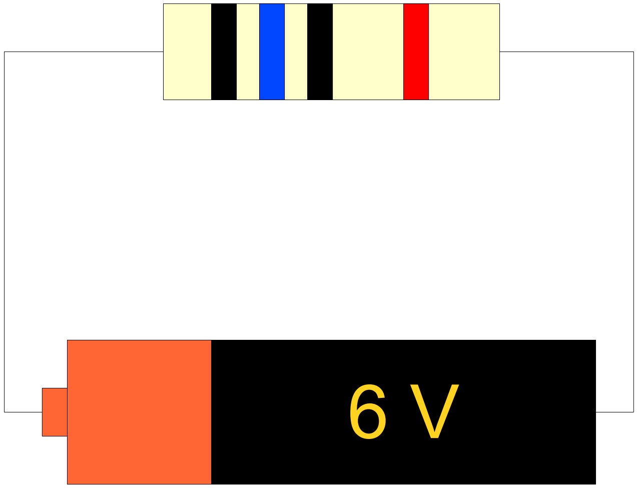
$$4\Omega + 2\Omega = 6\Omega$$

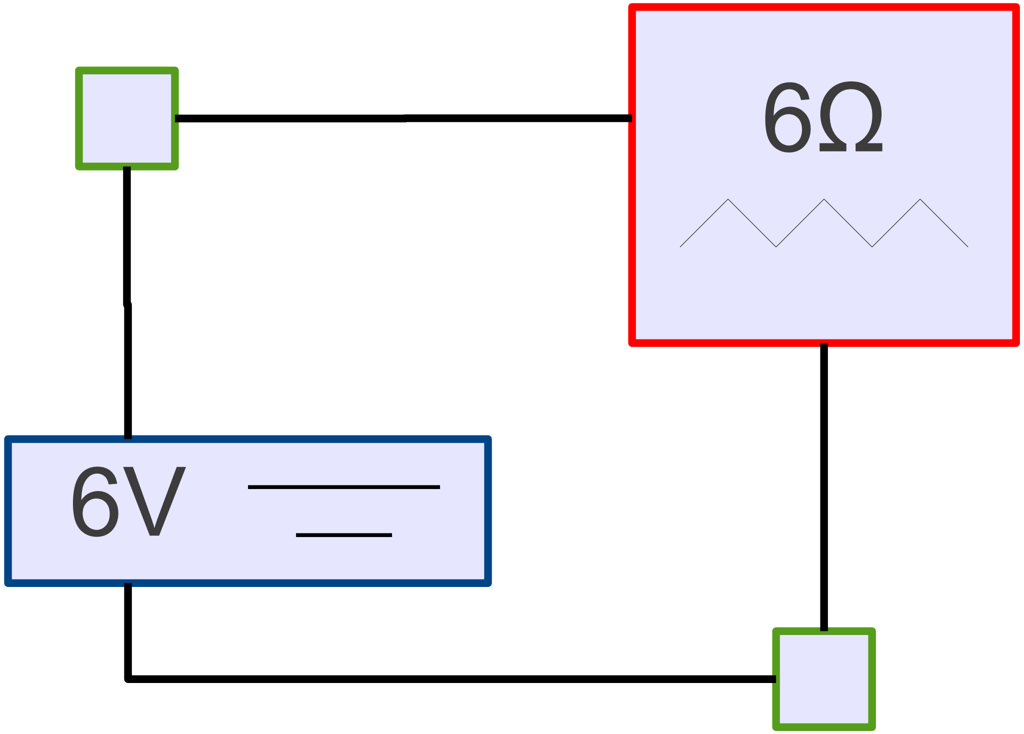


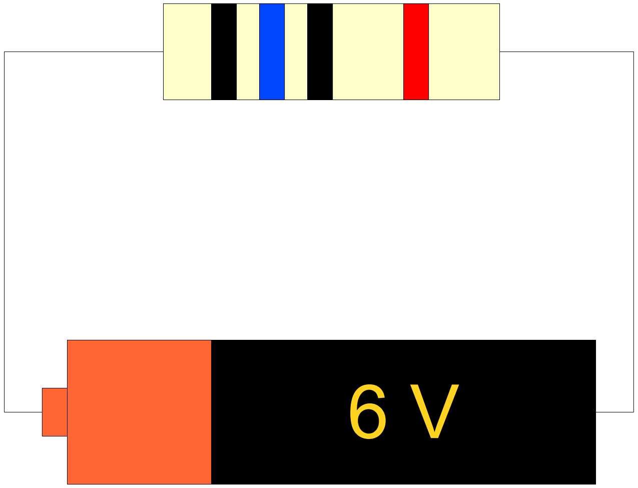
$$4\Omega + 2\Omega = 6\Omega$$


6Ω
6V

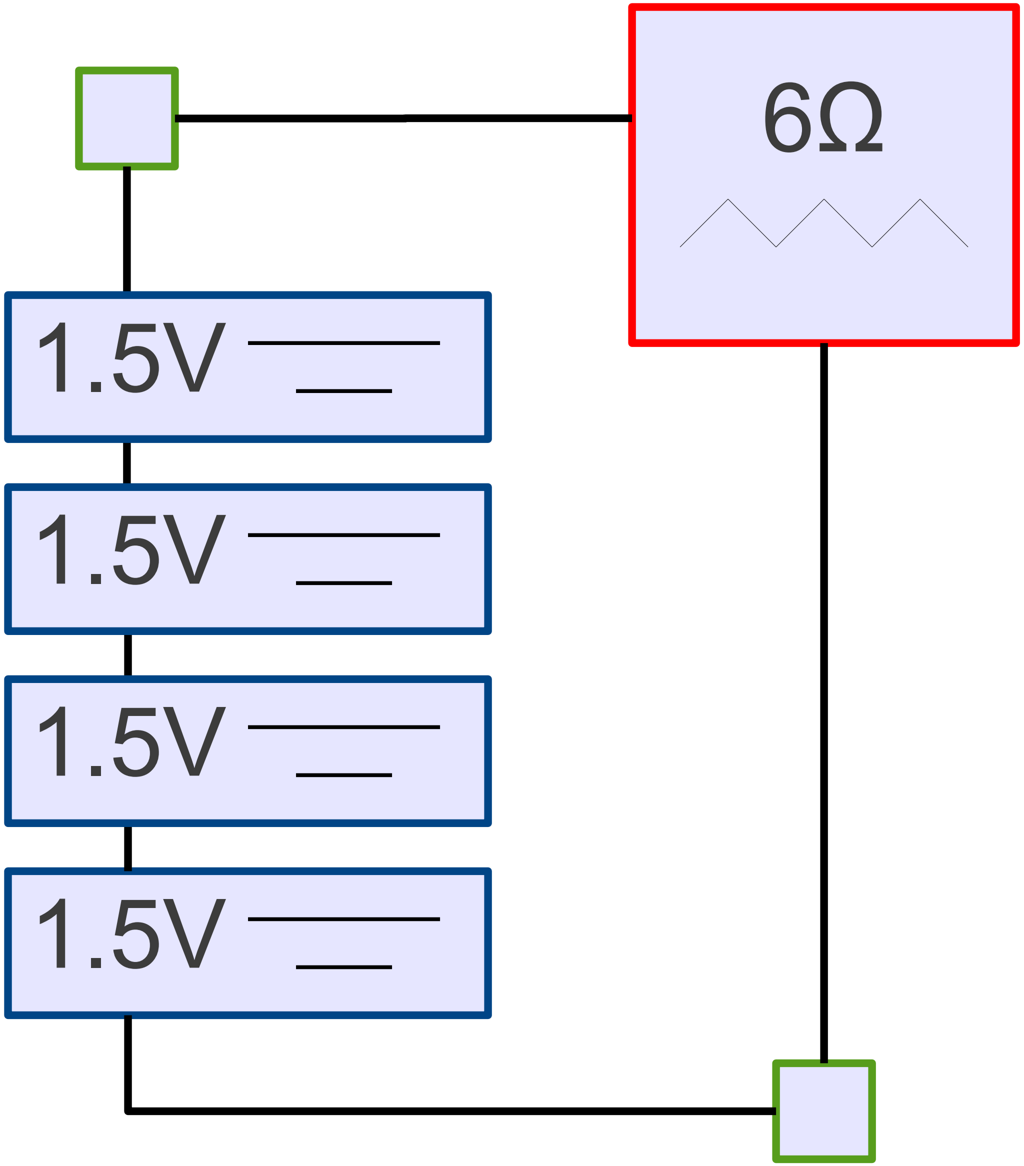

6Ω
6V

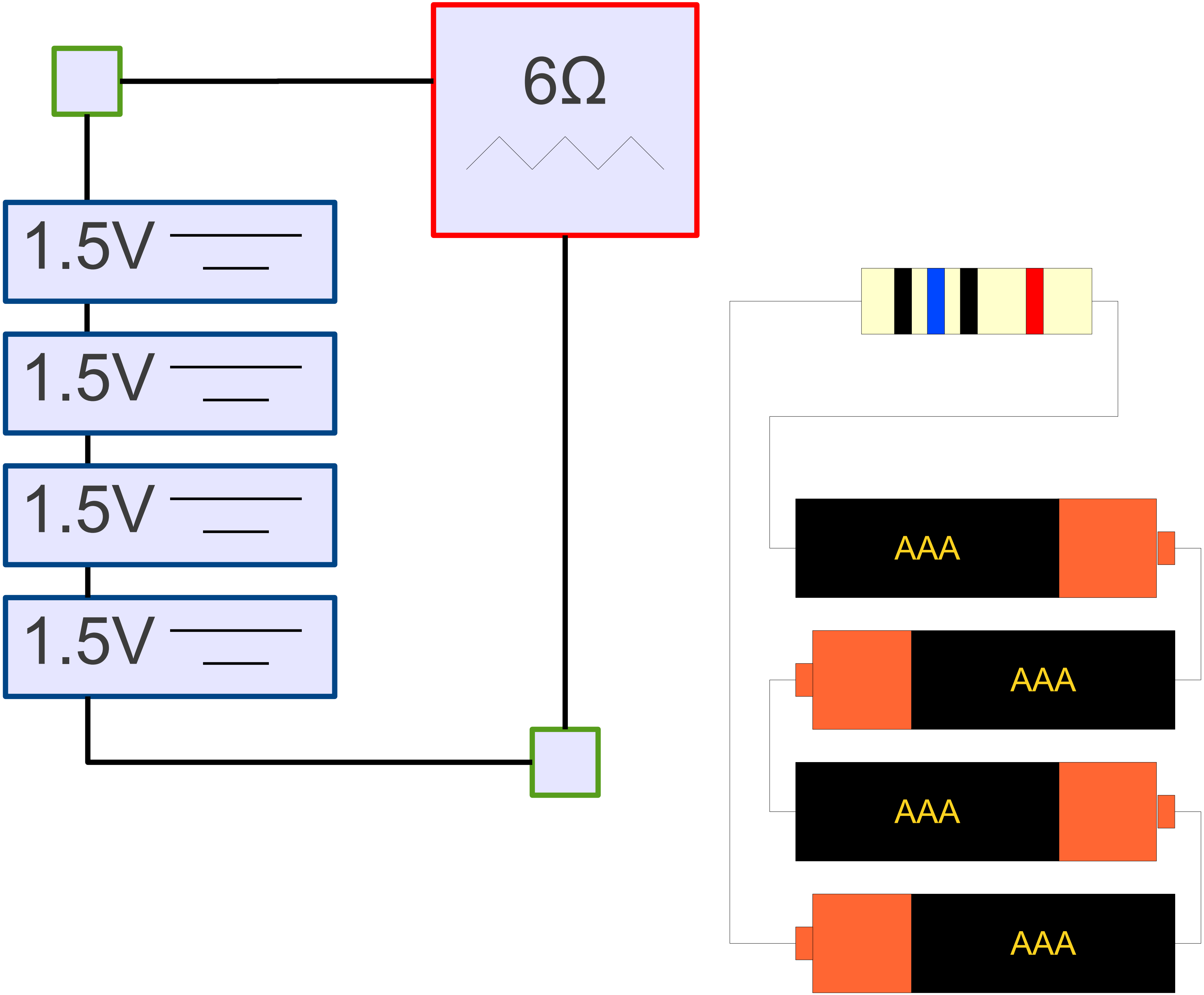

6 V

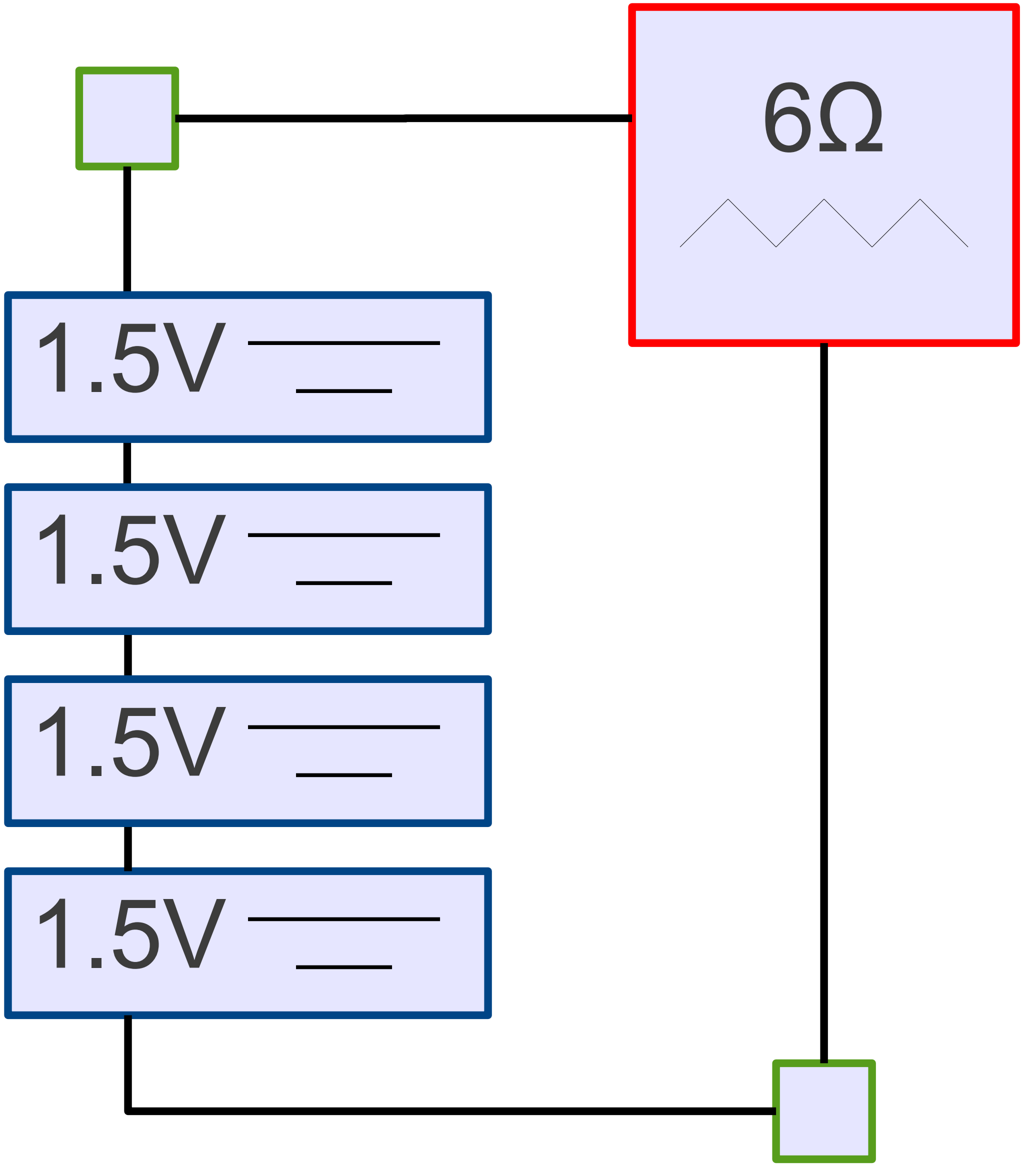


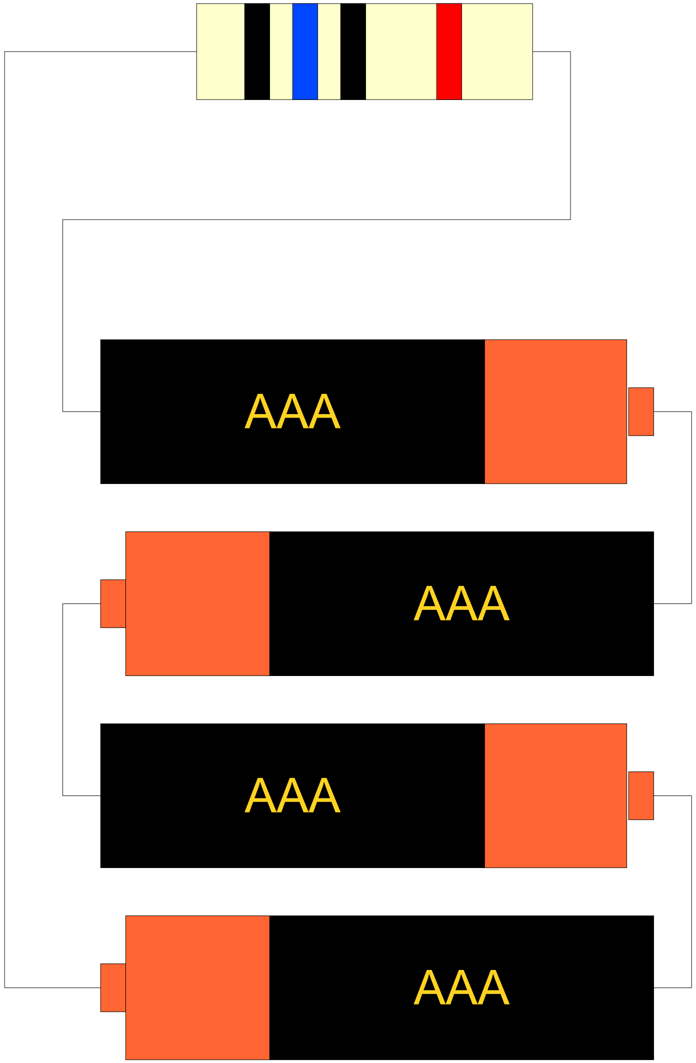


Total Cost: $4.75


6Ω
1.5V
1.5V
1.5V
1.5V


6Ω
1.5V
1.5V
1.5V
1.5V
AAA
AAA
AAA
AAA



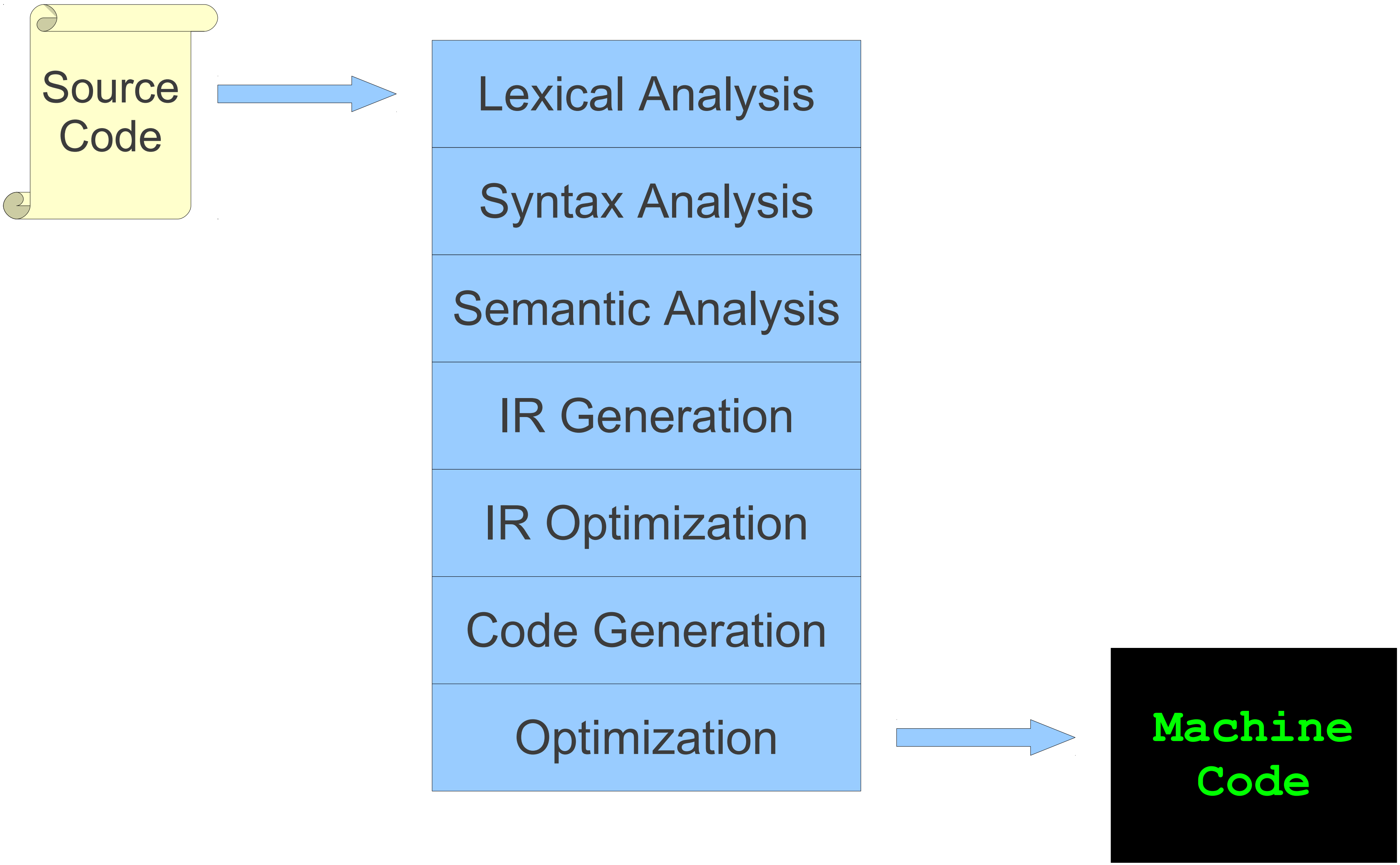


Total Cost: $1.00

From Description to Implementation

- **Lexical analysis (Scanning):** Identify logical pieces of the description.
- **Syntax analysis (Parsing):** Identify how those pieces relate to each other.
- **Semantic analysis:** Identify the meaning of the overall structure.
- **IR Generation:** Design one possible structure.
- **IR Optimization:** Simplify the intended structure.
- **Generation:** Fabricate the structure.
- **Optimization:** Improve the resulting structure.

The Structure of a Modern Compiler



Source Code

Lexical Analysis

Syntax Analysis

Semantic Analysis

IR Generation

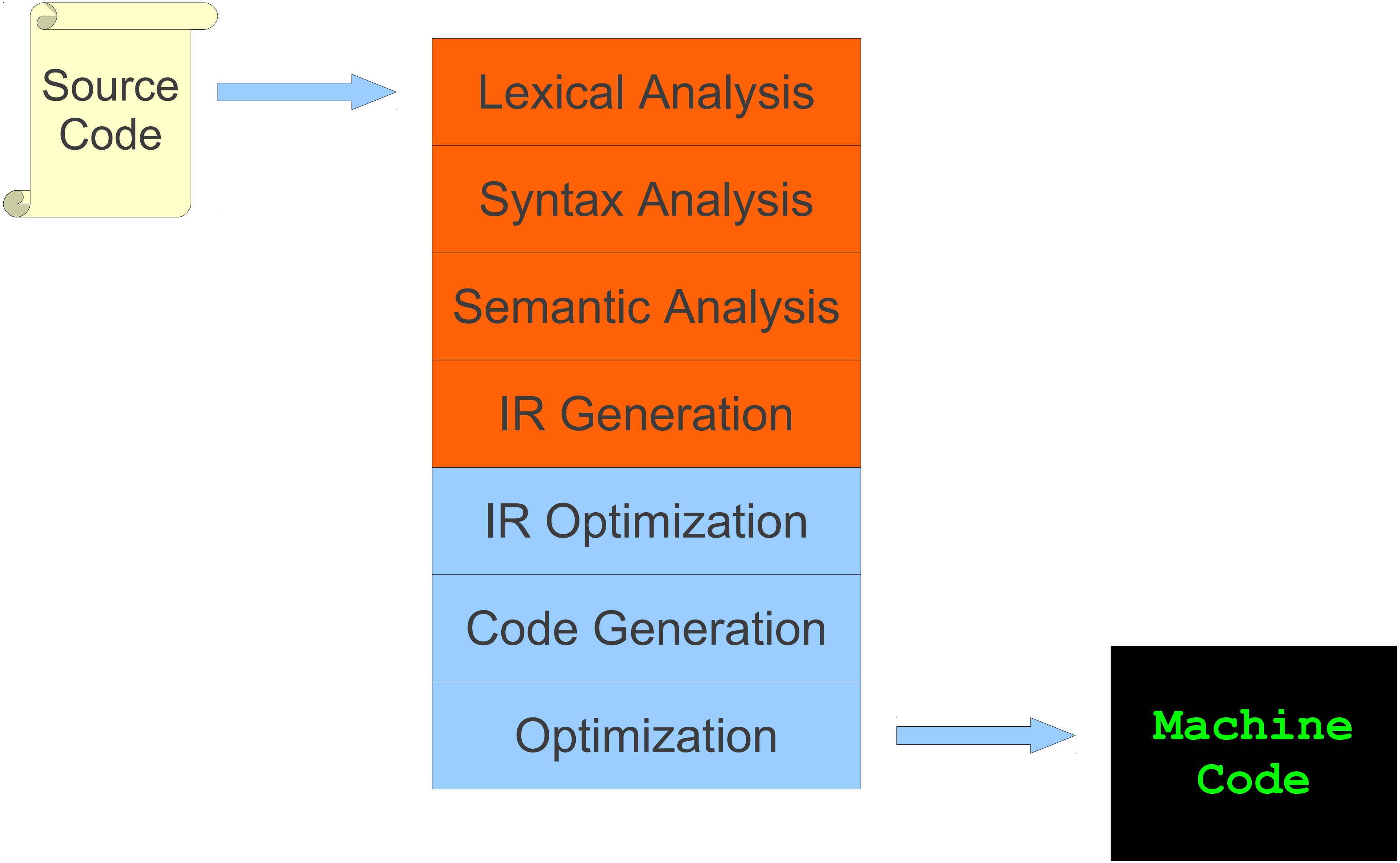
IR Optimization

Code Generation

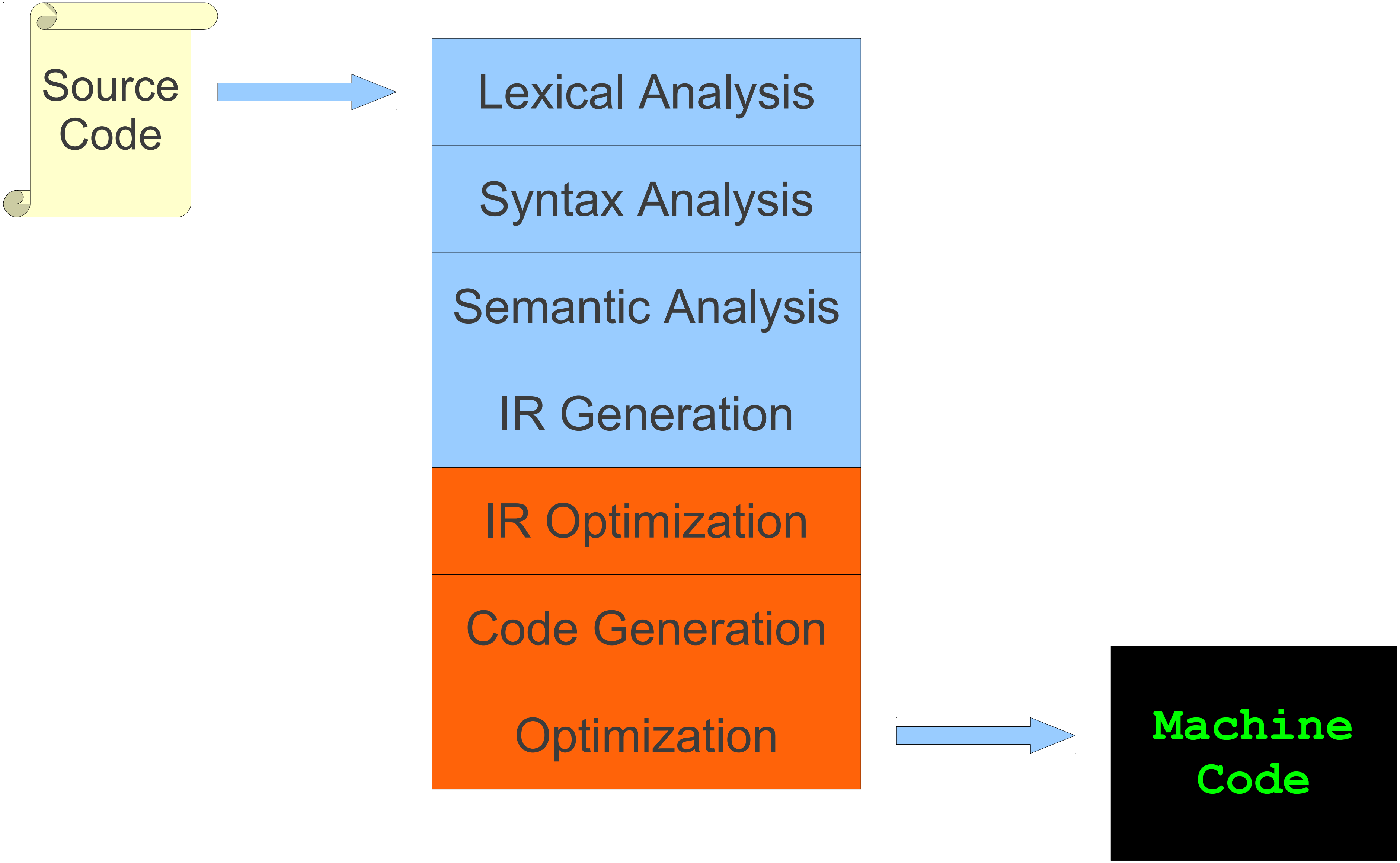
Optimization

Machine Code

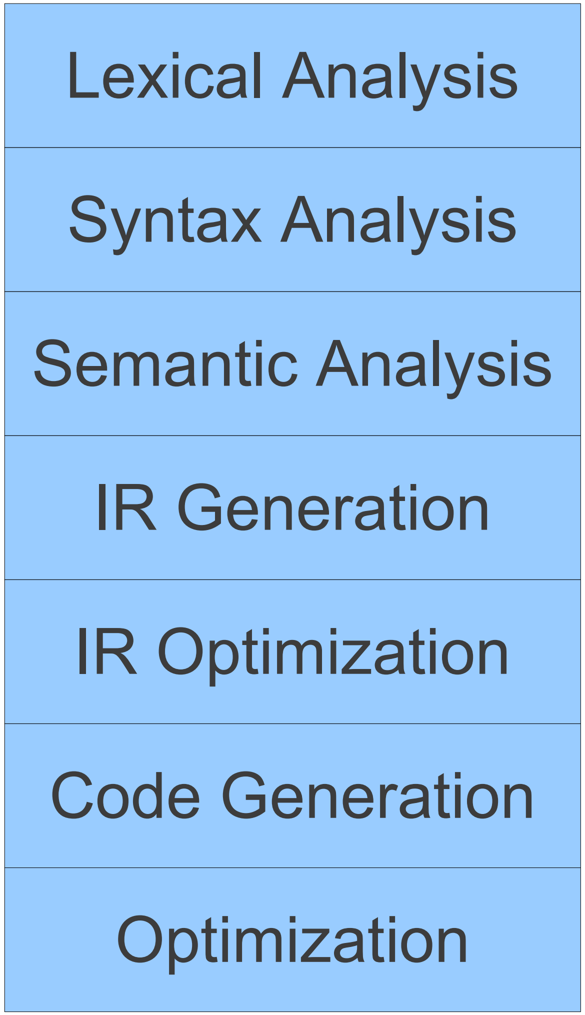
The Structure of a Modern Compiler



The Structure of a Modern Compiler



```
while (y < z) {
    int x = a + b;
    y += x;
}
```



Lexical Analysis

Syntax Analysis

Semantic Analysis

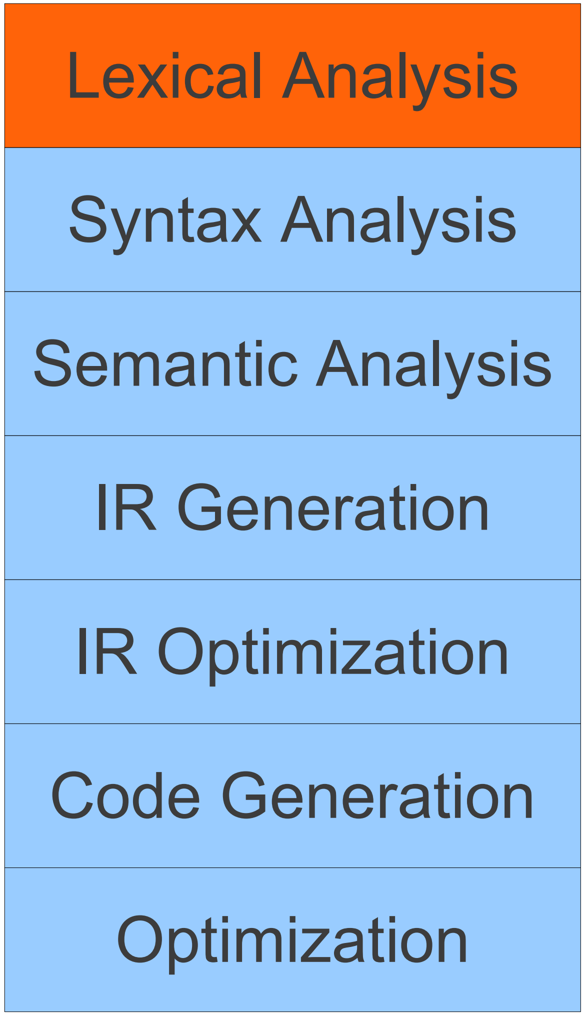
IR Generation

IR Optimization

Code Generation

Optimization

```
while (y < z) {
    int x = a + b;
    y += x;
}
```



Lexical Analysis

Syntax Analysis

Semantic Analysis

IR Generation

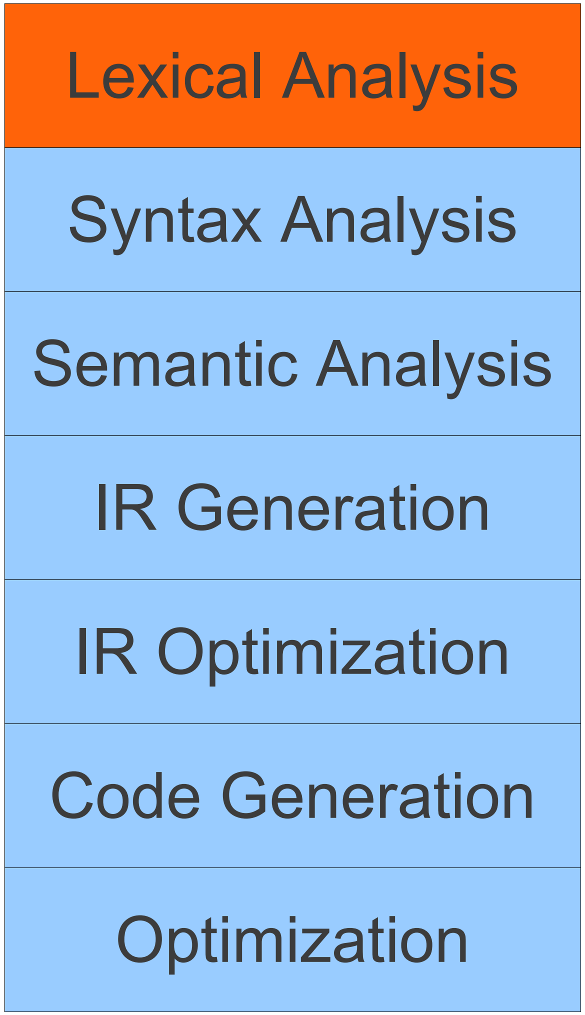
IR Optimization

Code Generation

Optimization

```
while (y < z) {
    int x = a + b;
    y += x;
}
```

```
T_While
T_LeftParen
T_Identifier y
T_Less
T_Identifier z
T_RightParen
T_OpenBrace
T_Int
T_Identifier x
T_Assign
T_Identifier a
T_Plus
T_Identifier b
T_Semicolon
T_Identifier y
T_PlusAssign
T_Identifier x
T_Semicolon
T_CloseBrace
```



Lexical Analysis

Syntax Analysis

Semantic Analysis

IR Generation

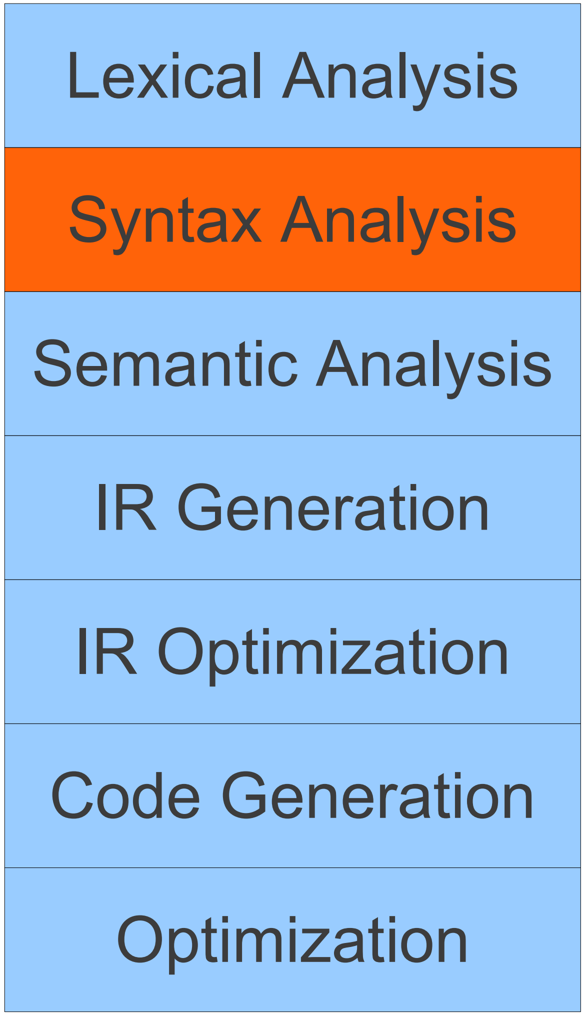
IR Optimization

Code Generation

Optimization

```
while (y < z) {
    int x = a + b;
    y += x;
}
```

```
T_While
T_LeftParen
T_Identifier y
T_Less
T_Identifier z
T_RightParen
T_OpenBrace
T_Int
T_Identifier x
T_Assign
T_Identifier a
T_Plus
T_Identifier b
T_Semicolon
T_Identifier y
T_PlusAssign
T_Identifier x
T_Semicolon
T_CloseBrace
```



Lexical Analysis

Syntax Analysis

Semantic Analysis

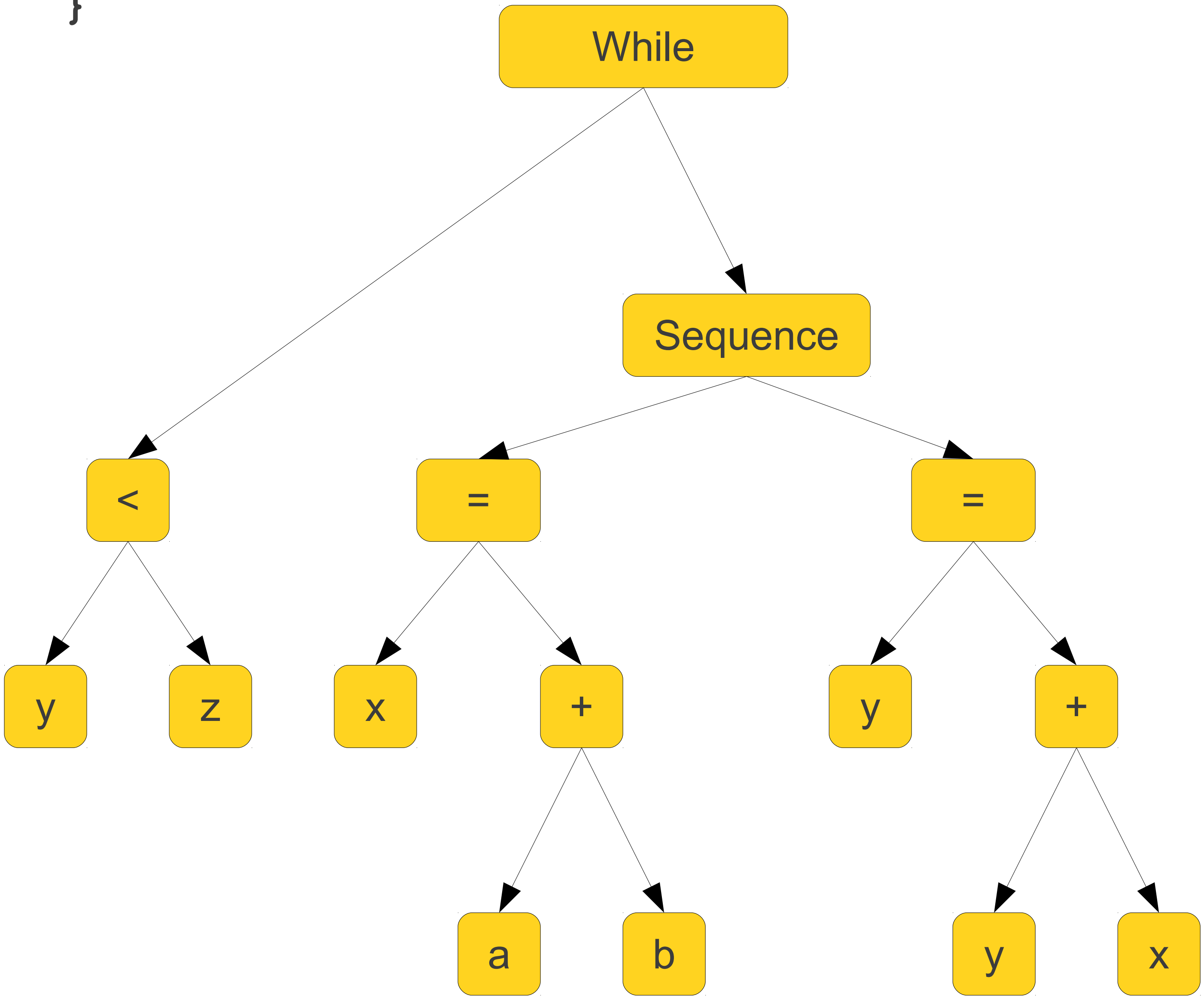
IR Generation

IR Optimization

Code Generation

Optimization

```
while (y < z) {
    int x = a + b;
    y += x;
}
```



Lexical Analysis

Syntax Analysis

Semantic Analysis

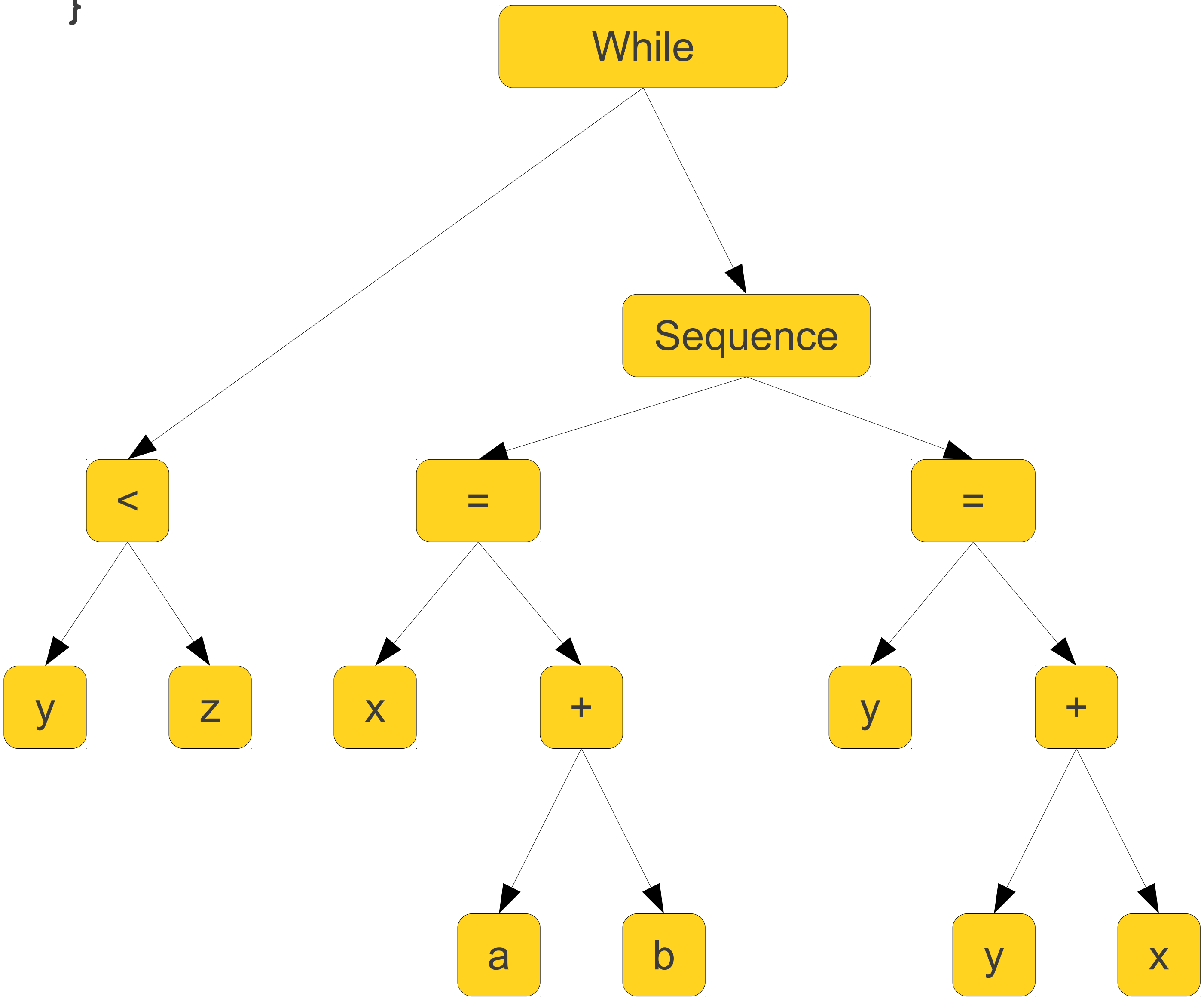
IR Generation

IR Optimization

Code Generation

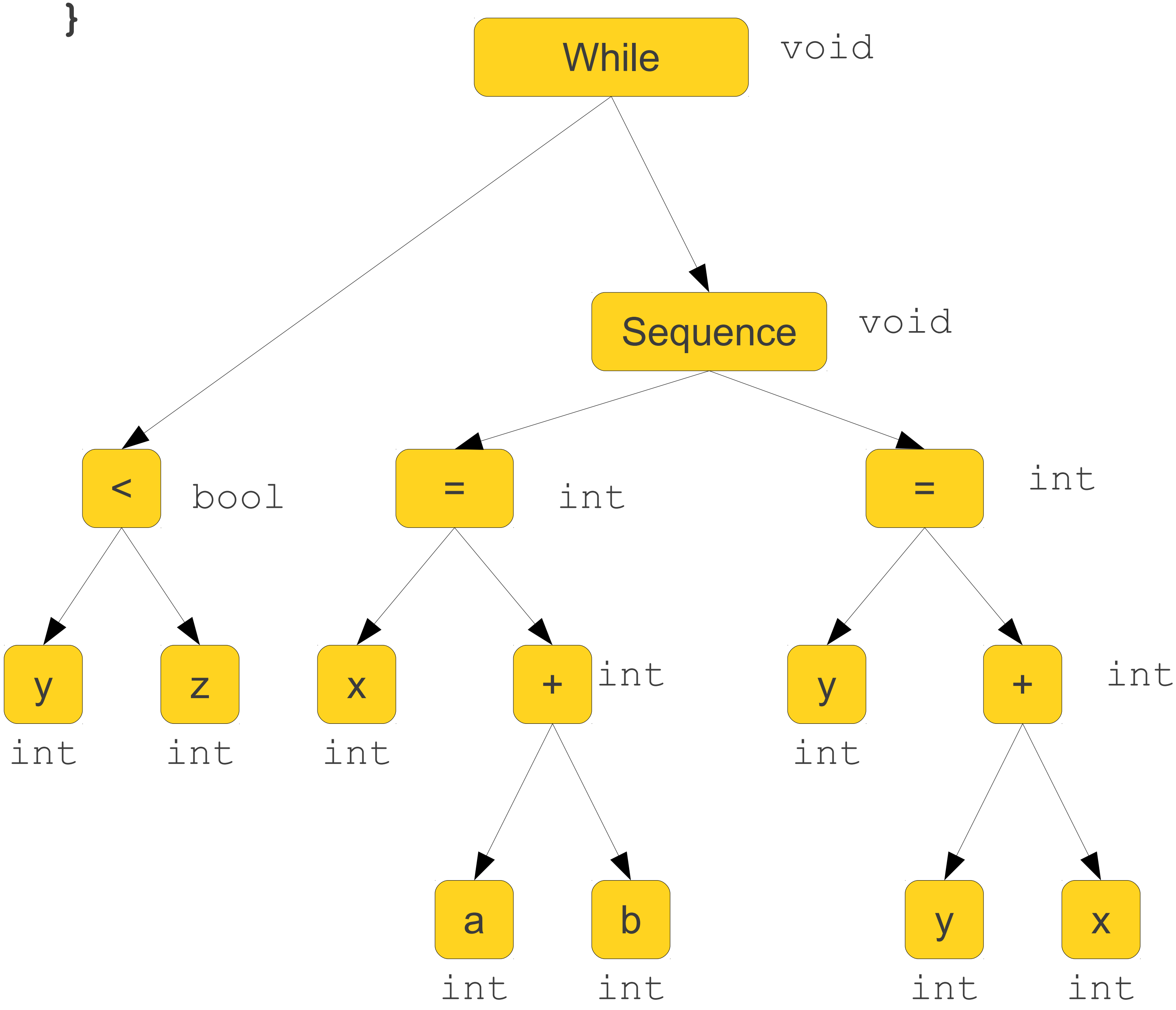
Optimization

```
while (y < z) {
    int x = a + b;
    y += x;
}
```



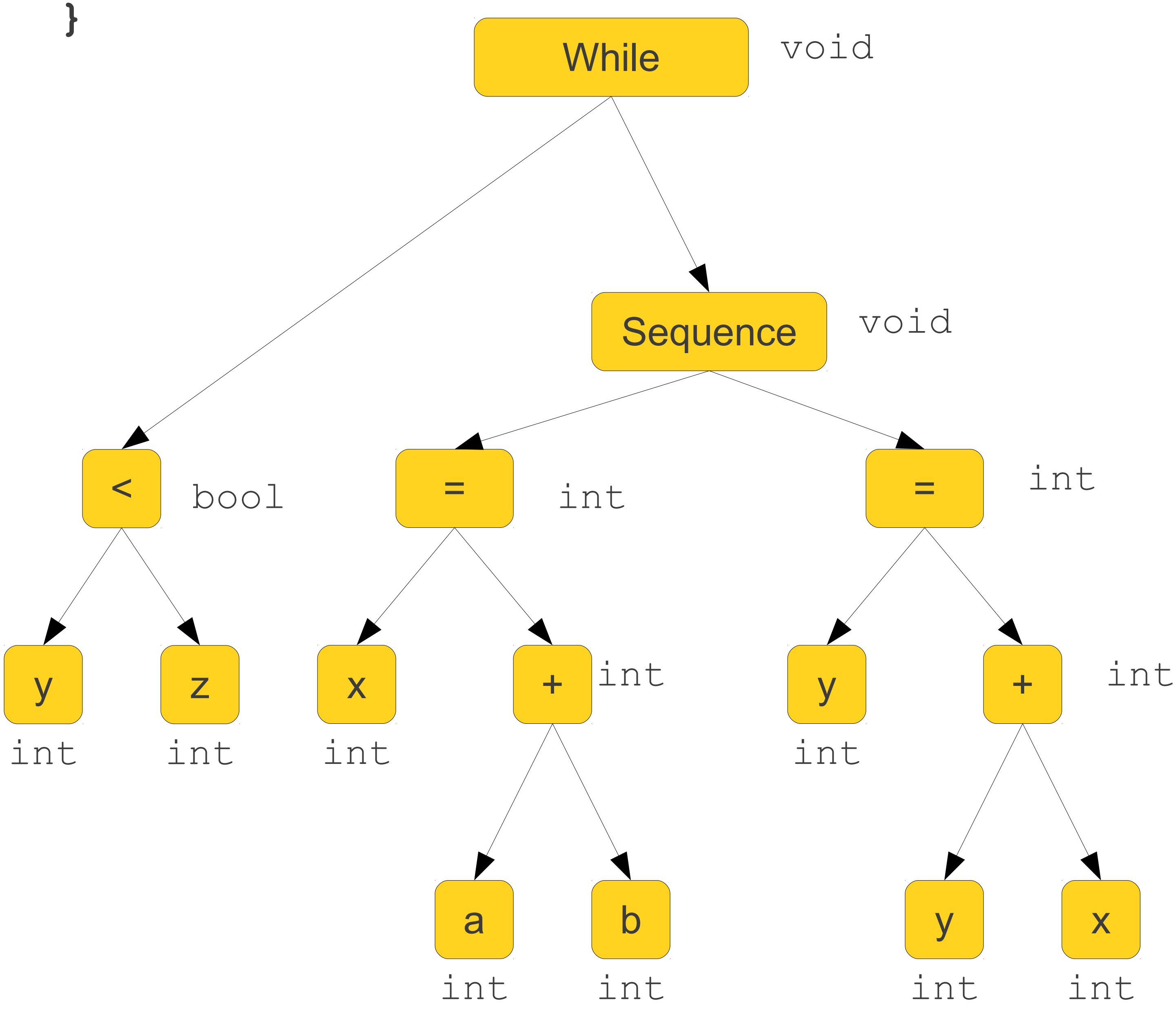
- Lexical Analysis
- Syntax Analysis
- Semantic Analysis
- IR Generation
- IR Optimization
- Code Generation
- Optimization

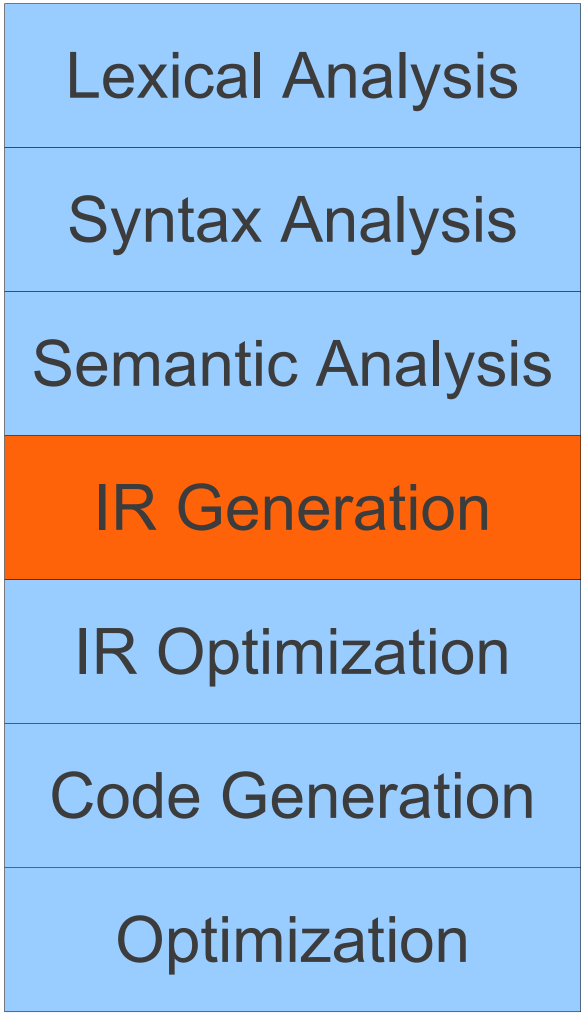
```
while (y < z) {
    int x = a + b;
    y += x;
}
```



- Lexical Analysis
- Syntax Analysis
- Semantic Analysis
- IR Generation
- IR Optimization
- Code Generation
- Optimization

```
while (y < z) {
    int x = a + b;
    y += x;
}
```




Lexical Analysis

Syntax Analysis

Semantic Analysis

IR Generation

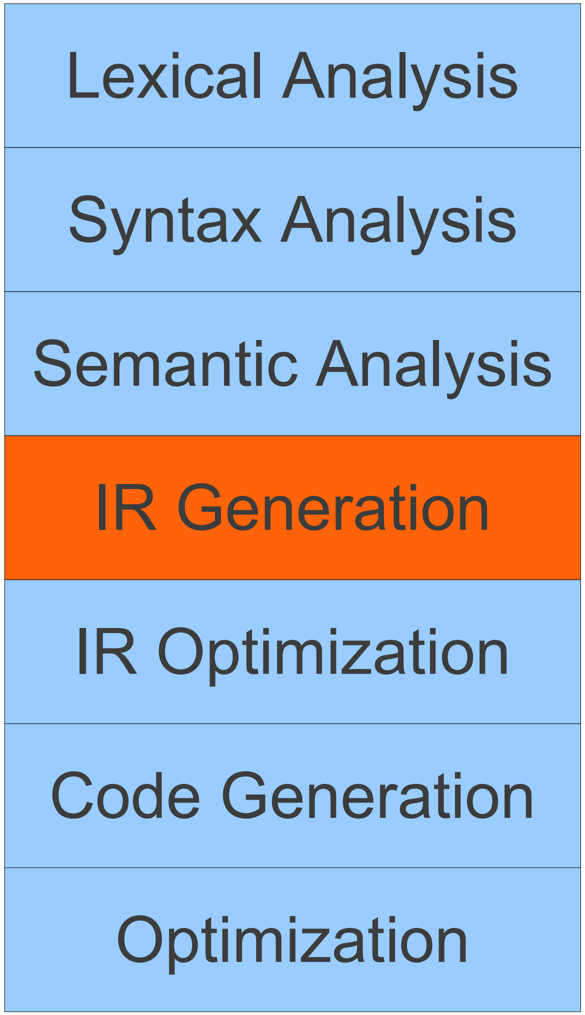
IR Optimization

Code Generation

Optimization

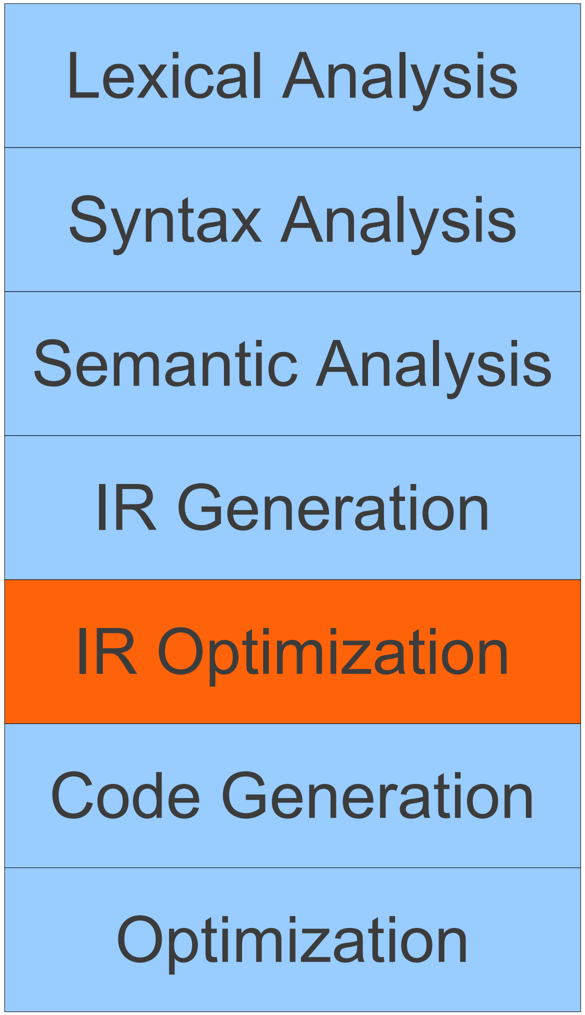
```
while (y < z) {
    int x = a + b;
    y += x;
}
```

```
Loop: x   = a + b
      y   = x + y
      _t1 = y < z
      if _t1 goto Loop
```



```
while (y < z) {
    int x = a + b;
    y += x;
}
```

```
Loop: x   = a + b
      y   = x + y
      _t1 = y < z
      if _t1 goto Loop
```



Lexical Analysis

Syntax Analysis

Semantic Analysis

IR Generation

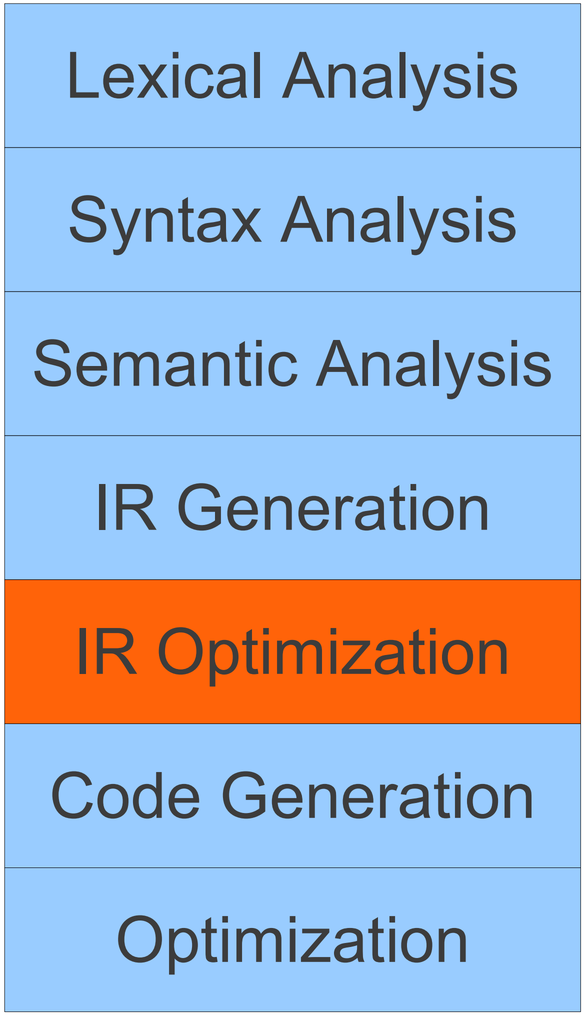
IR Optimization

Code Generation

Optimization

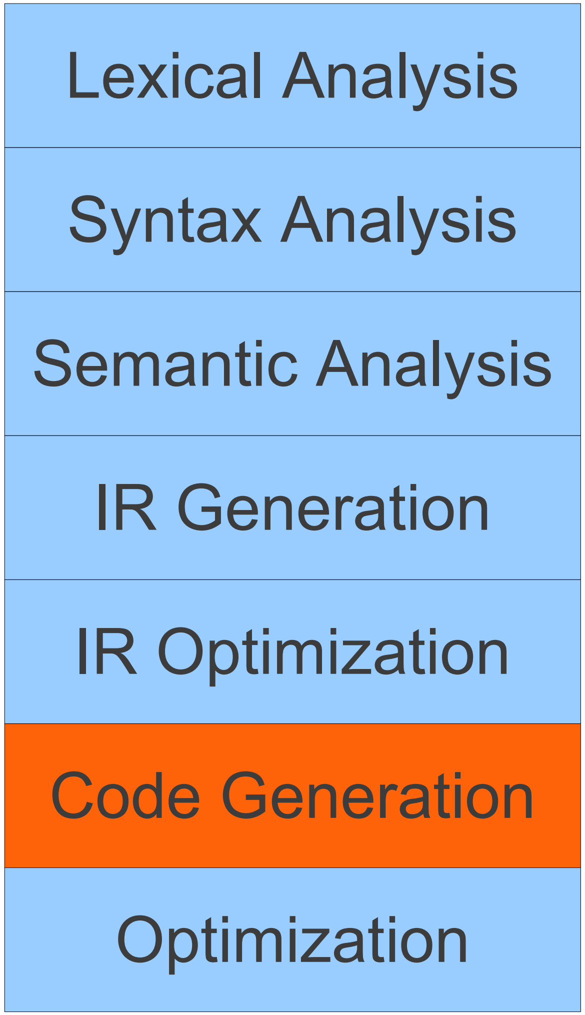
```
while (y < z) {
    int x = a + b;
    y += x;
}
```

```
      x   = a + b
Loop: y   = x + y
      _t1 = y < z
      if _t1 goto Loop
```



```
while (y < z) {
    int x = a + b;
    y += x;
}
```

```
      x   = a + b
Loop: y   = x + y
      _t1 = y < z
      if _t1 goto Loop
```



Lexical Analysis

Syntax Analysis

Semantic Analysis

IR Generation

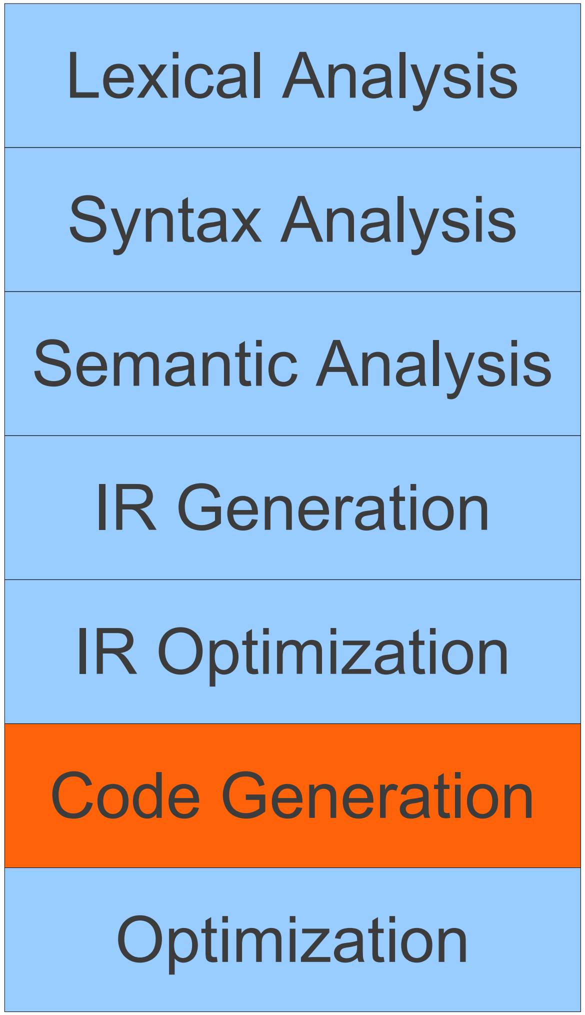
IR Optimization

Code Generation

Optimization

```
while (y < z) {
    int x = a + b;
    y += x;
}
```

```
      add $1, $2, $3
Loop: add $4, $1, $4
      slt $6, $1, $5
      beq $6, loop
```


Lexical Analysis

Syntax Analysis

Semantic Analysis

IR Generation

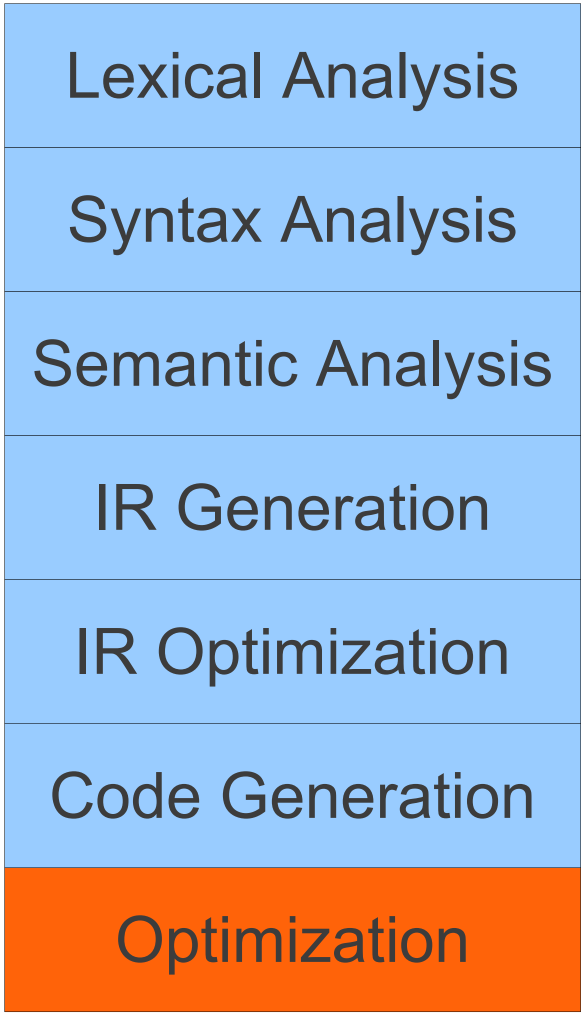
IR Optimization

Code Generation

Optimization

```
while (y < z) {
    int x = a + b;
    y += x;
}
```

```
      add $1, $2, $3
Loop: add $4, $1, $4
      slt $6, $1, $5
      beq $6, loop
```



Lexical Analysis

Syntax Analysis

Semantic Analysis

IR Generation

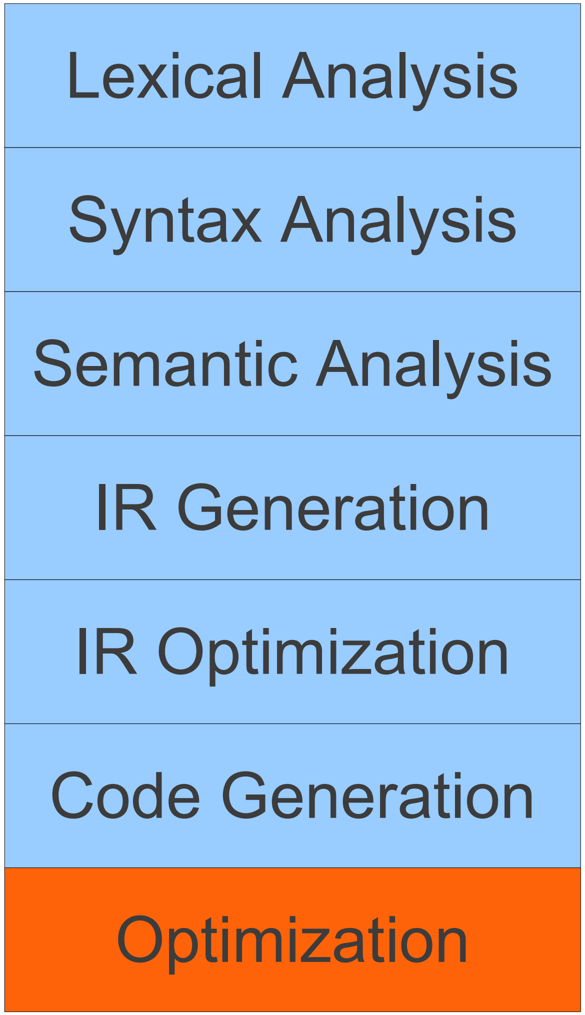
IR Optimization

Code Generation

Optimization

```
while (y < z) {
    int x = a + b;
    y += x;
}
```

```
      add $1, $2, $3
Loop: add $4, $1, $4
      blt $1, $5, loop
```



Lexical Analysis

Syntax Analysis

Semantic Analysis

IR Generation

IR Optimization

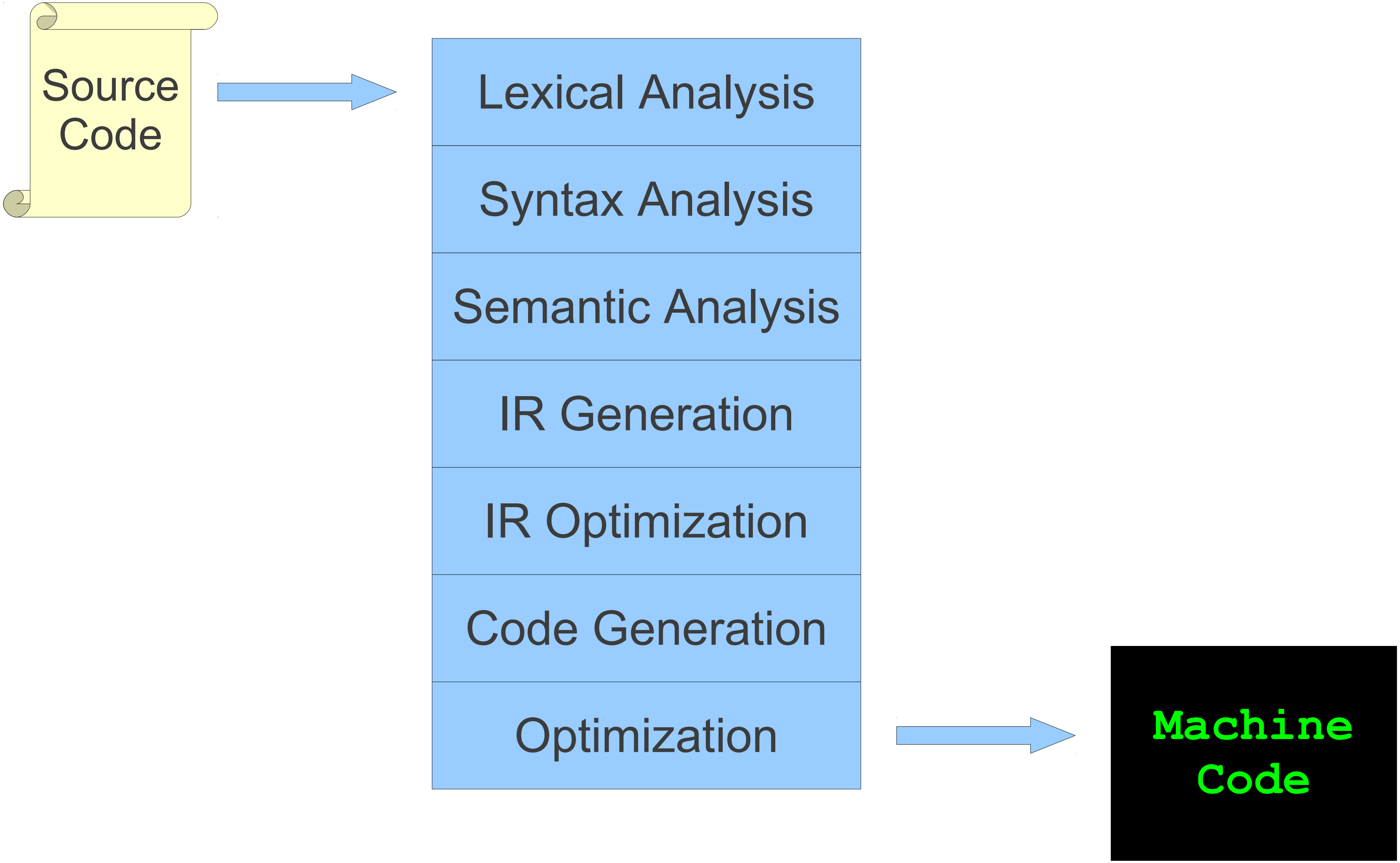
Code Generation

Optimization

The Course Project: **Decaf**

- Custom programming language similar to Java or C++.
- Object-oriented with free functions.
- Single inheritance with interfaces.

Programming Assignments



Source Code

Lexical Analysis

Syntax Analysis

Semantic Analysis

IR Generation

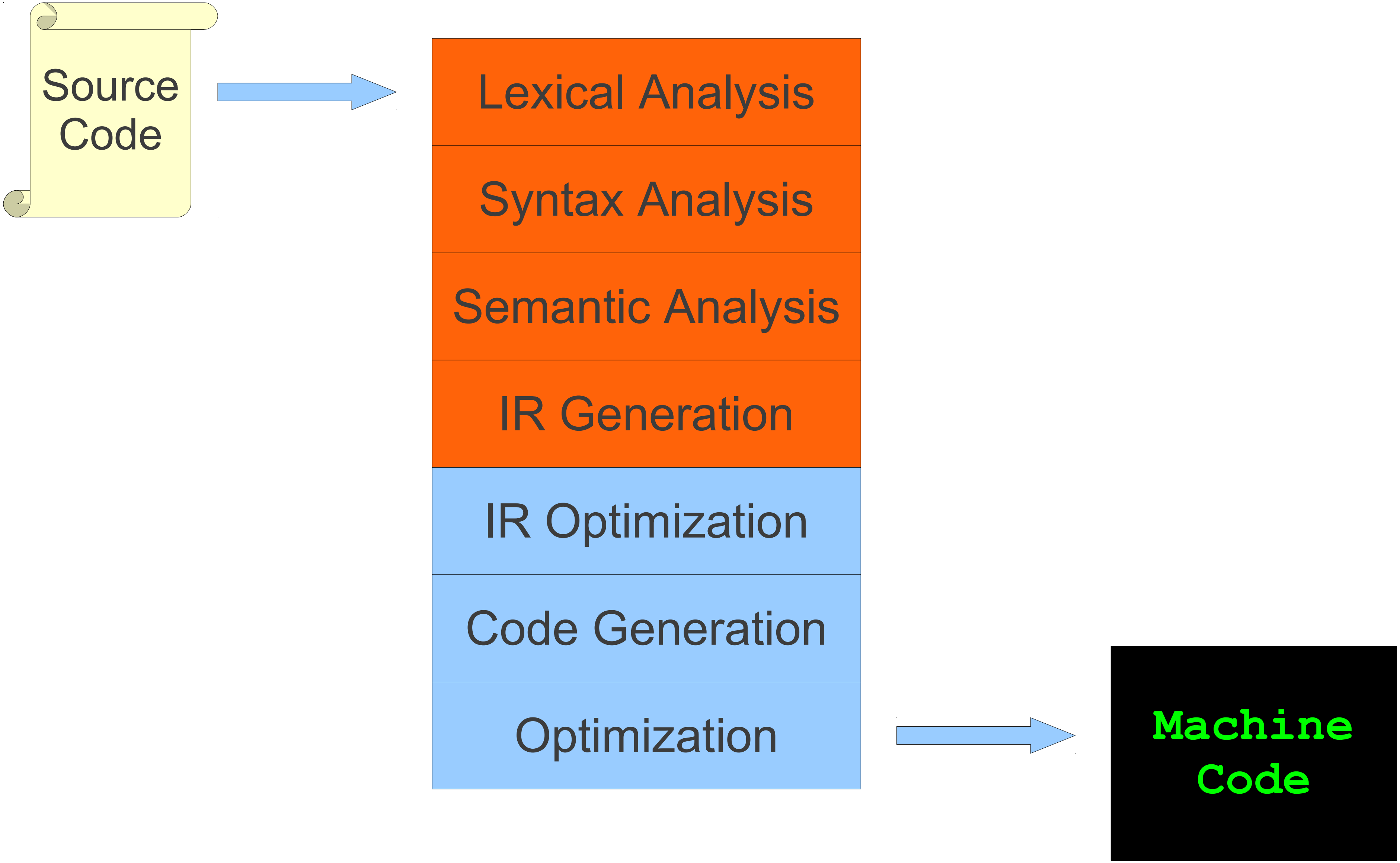
IR Optimization

Code Generation

Optimization

Machine Code

Programming Assignments



Source Code

Lexical Analysis

Syntax Analysis

Semantic Analysis

IR Generation

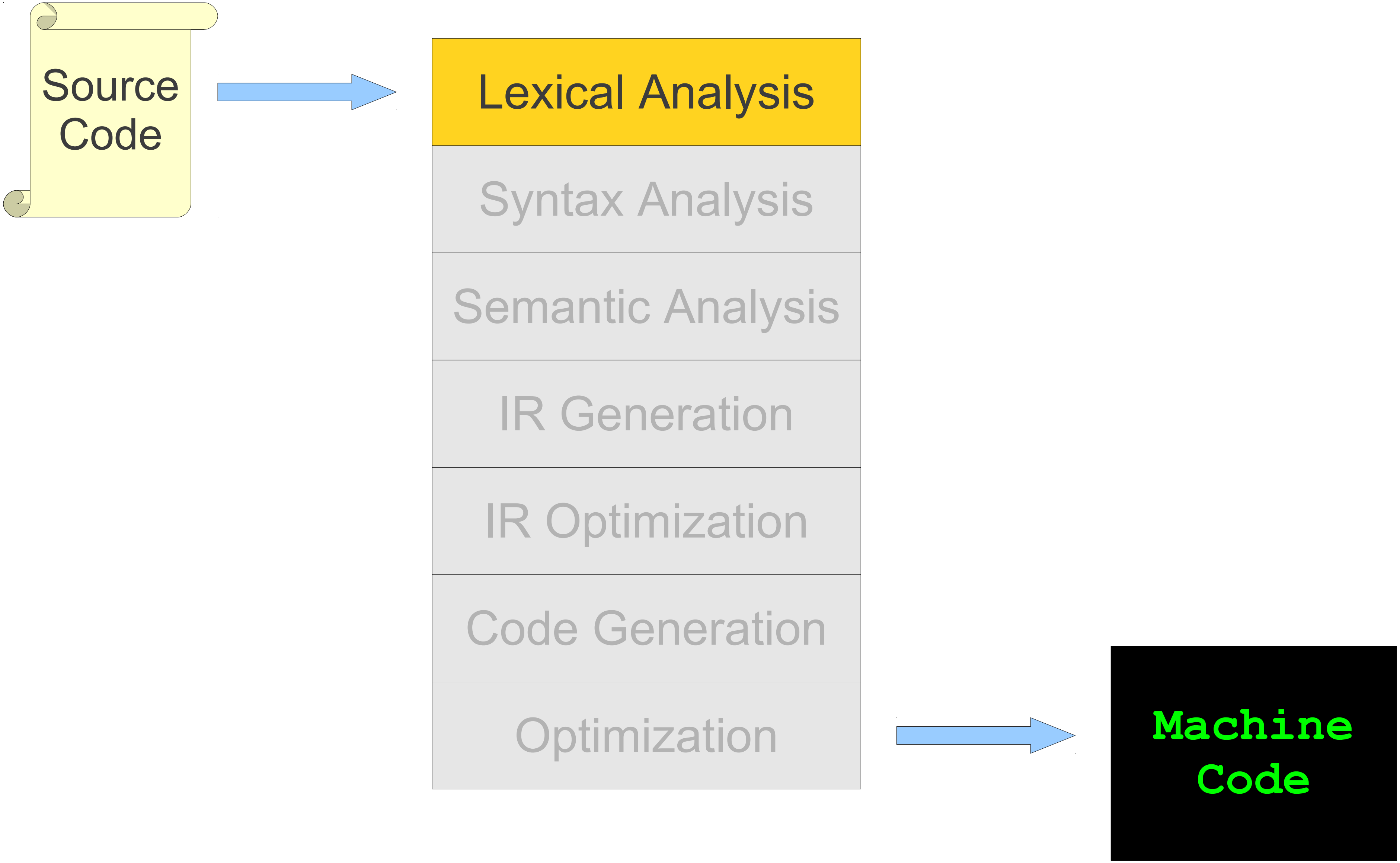
IR Optimization

Code Generation

Optimization

Machine Code

Next Time...


Source Code
Lexical Analysis
Syntax Analysis
Semantic Analysis
IR Generation
IR Optimization
Code Generation
Optimization
Machine Code