

Welcome to CS143: Compilers

- One Handout
- Today:
 - Course Information
 - Why Study Compilers?
 - A Quick History of Compilers
 - The Structure of a Compiler

<http://cs143.stanford.edu>

cs143-sum1011-staff@lists.stanford.edu

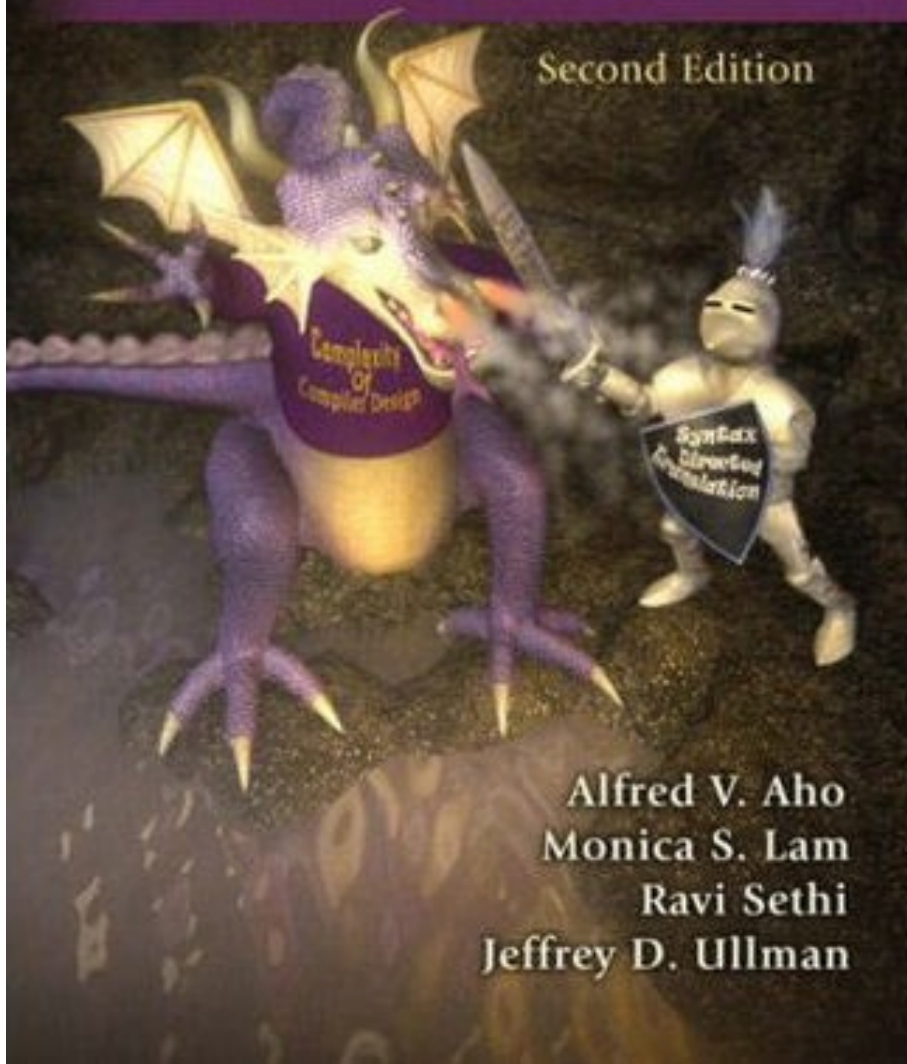
Or Email Us Directly:

htiek@cs.stanford.edu
hpapadak@stanford.edu
riddhi@stanford.edu

Compilers

Principles, Techniques, & Tools

Second Edition



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

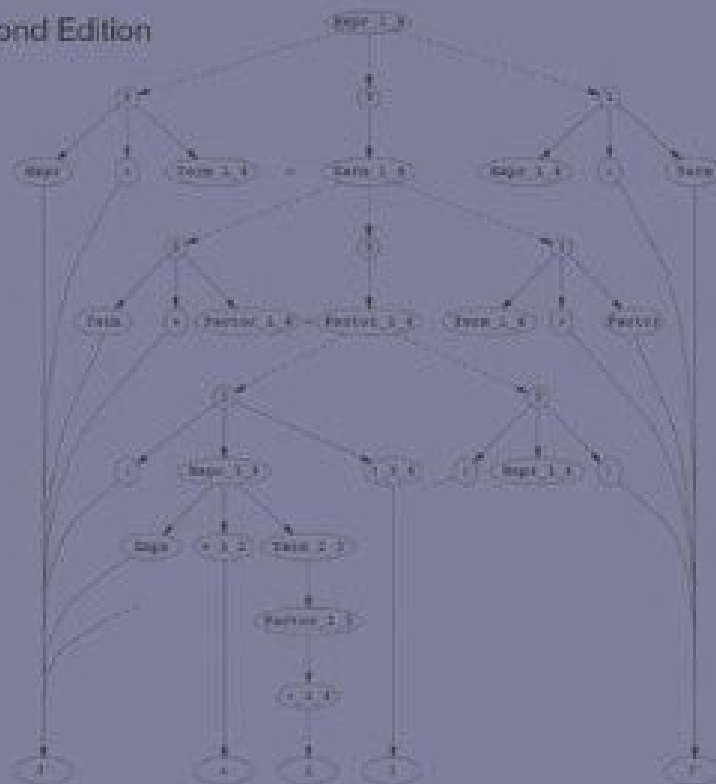
MONOGRAPHS IN COMPUTER SCIENCE


PARSING TECHNIQUES

A Practical Guide

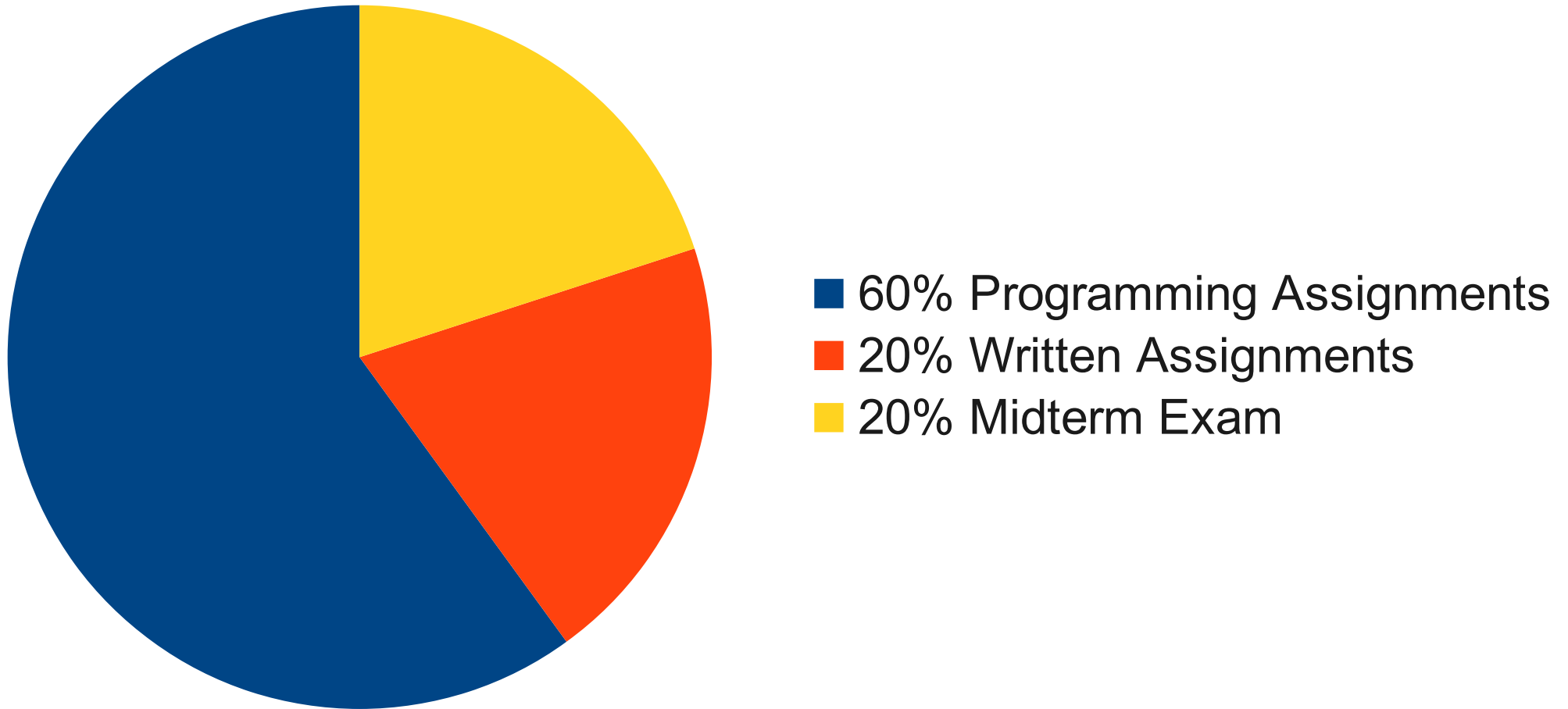
Dick Grune
Ceriël J.H. Jacobs

Second Edition



 Springer

Grading Policies



A word on the honor code...

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

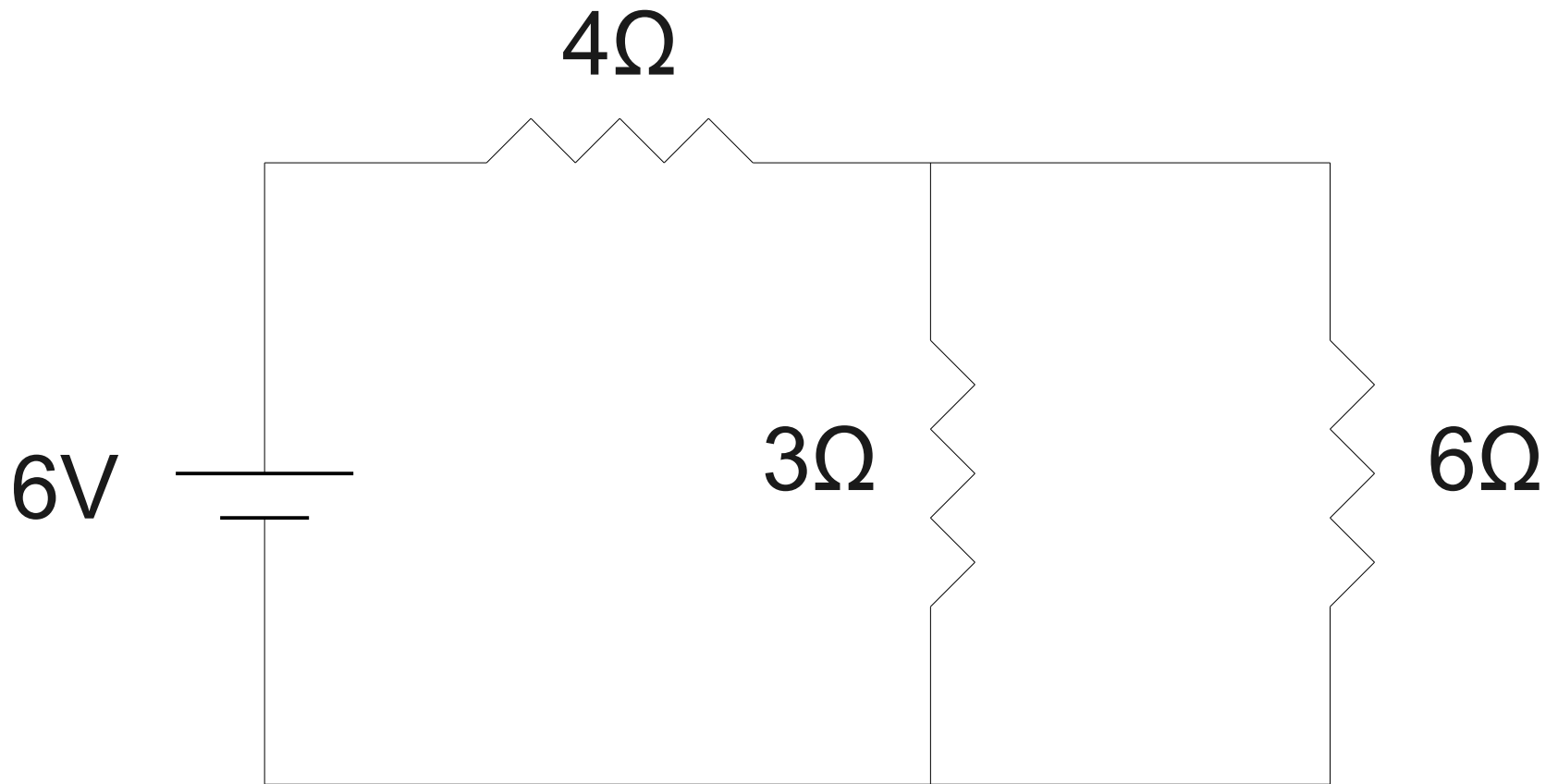
- First, there was nothing.
- Then, there was machine code.
- Then, there were assembly languages.
 - e.g. A-0, A-2
- Programming expensive; 50% of costs for machines went into programming.

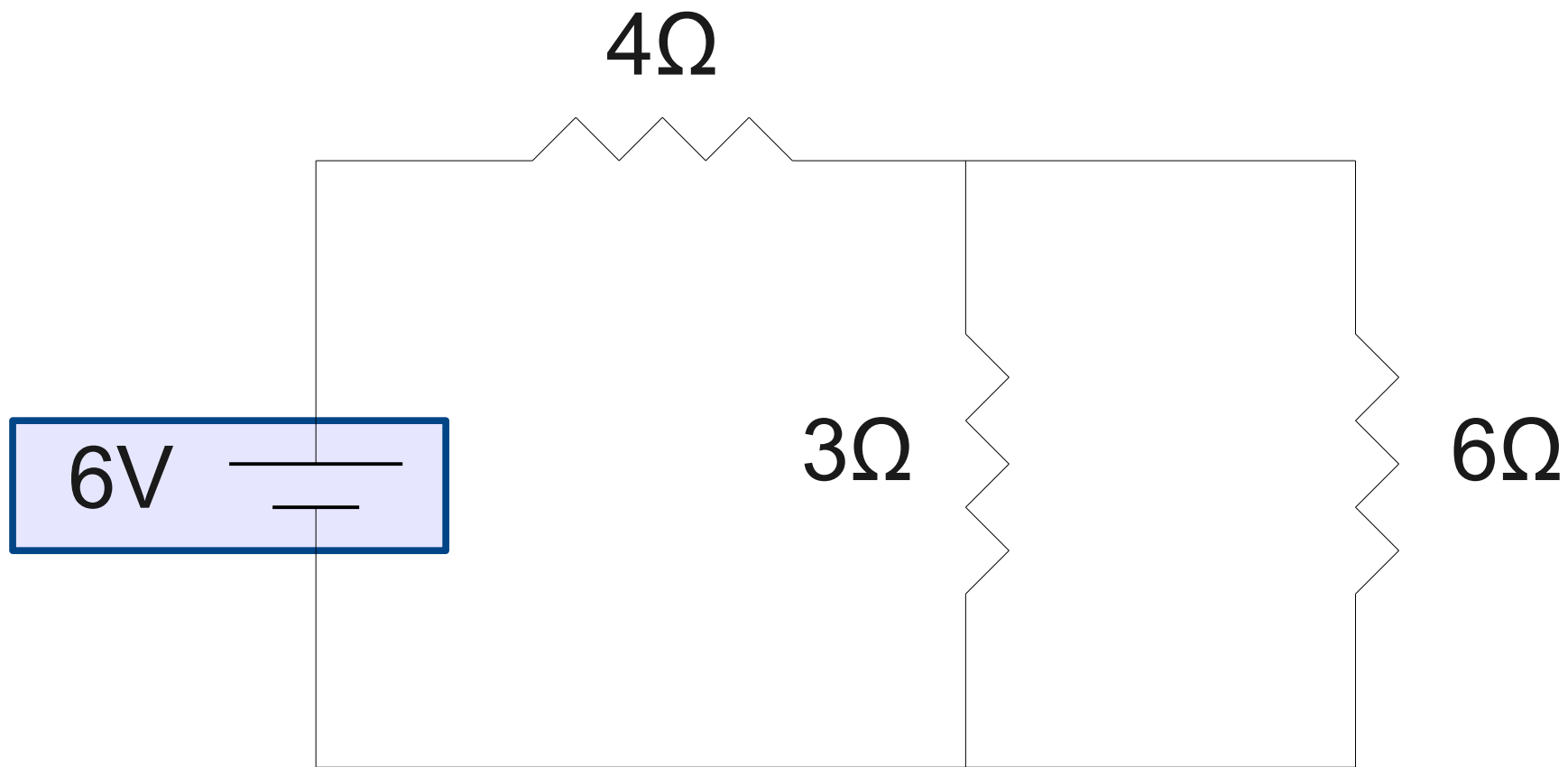
Enter John Backus

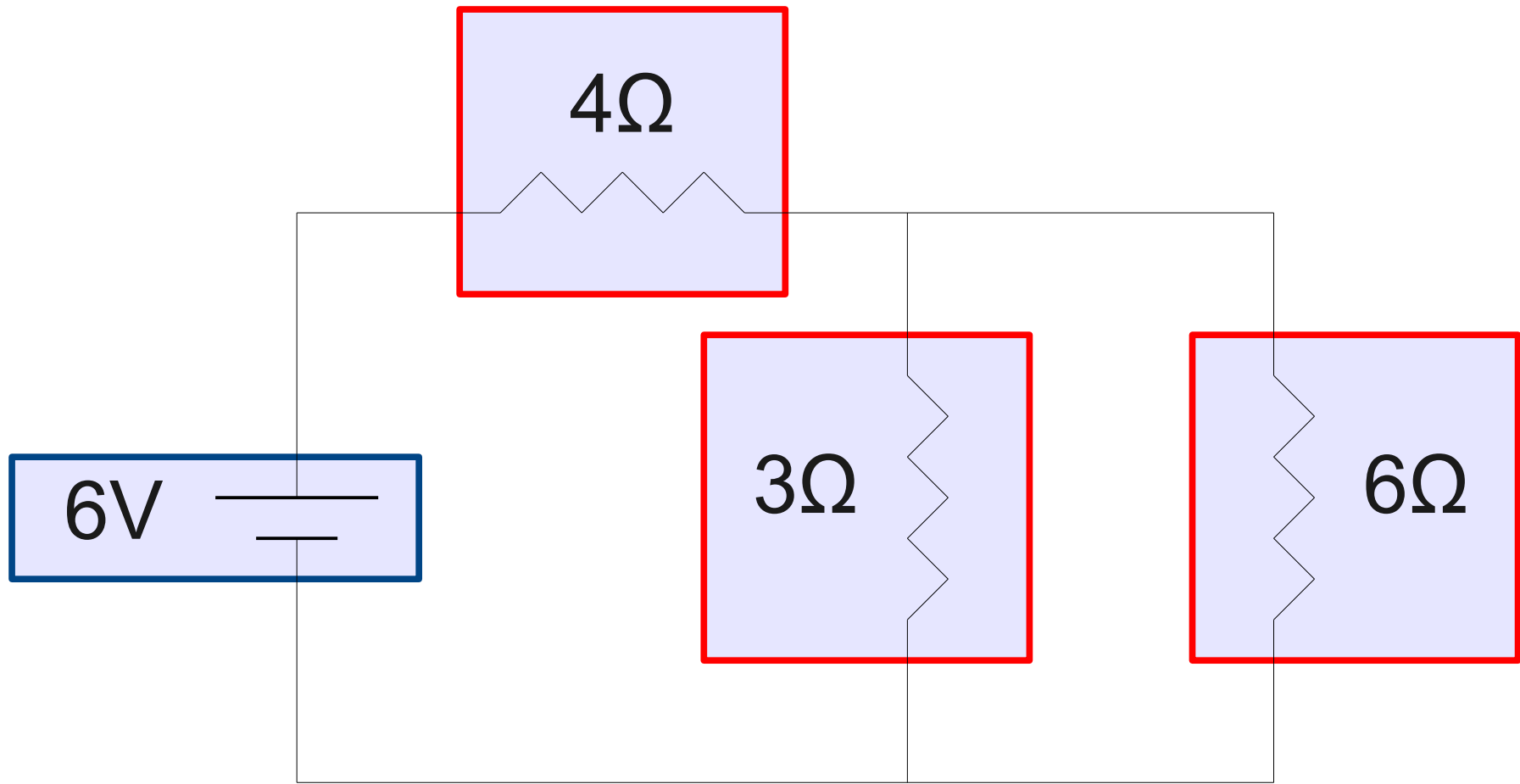


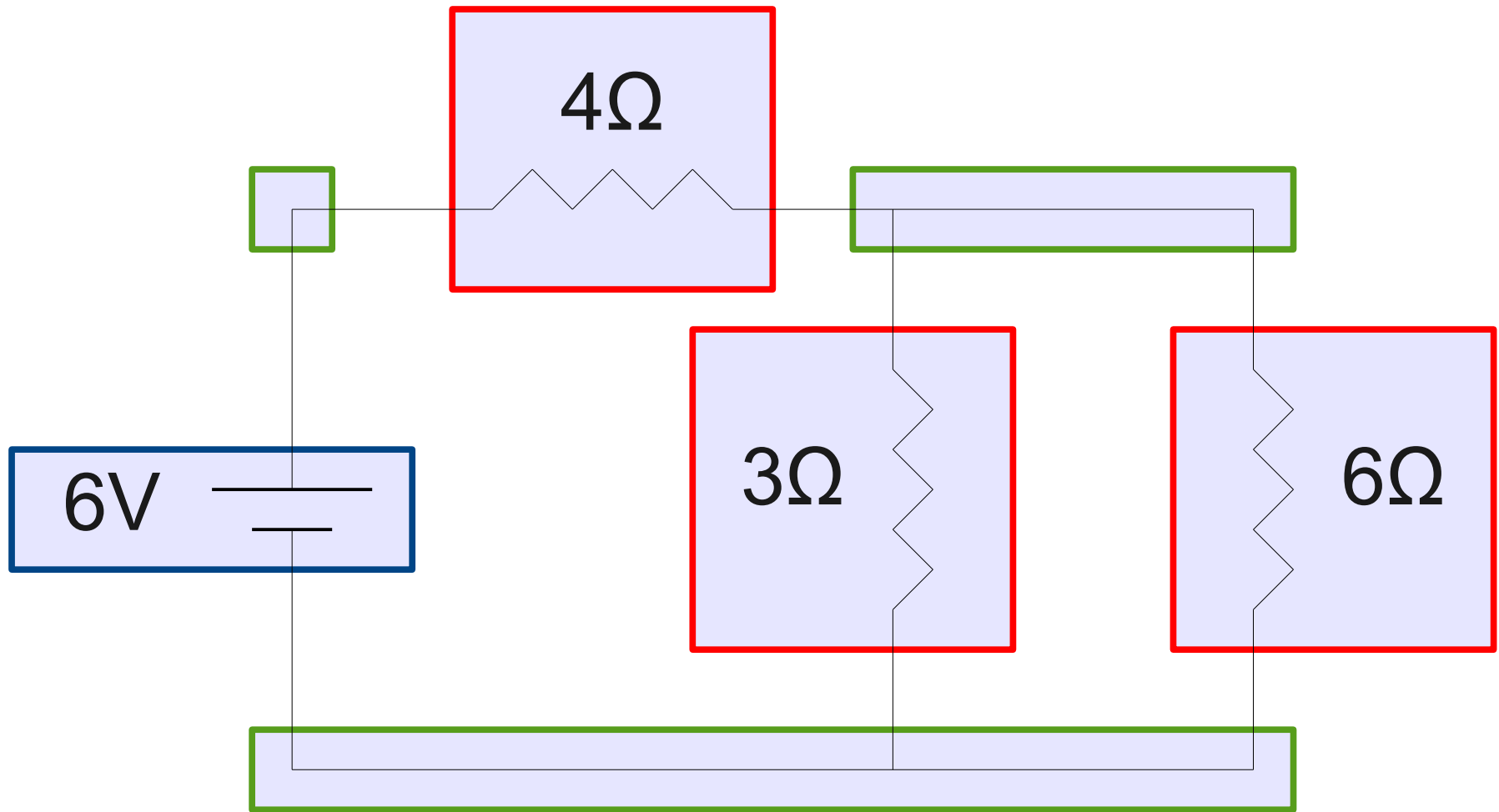
“I’m a terribly unscholarly person, and lazy. That was my motivating force in most of what I did, was how to avoid work”

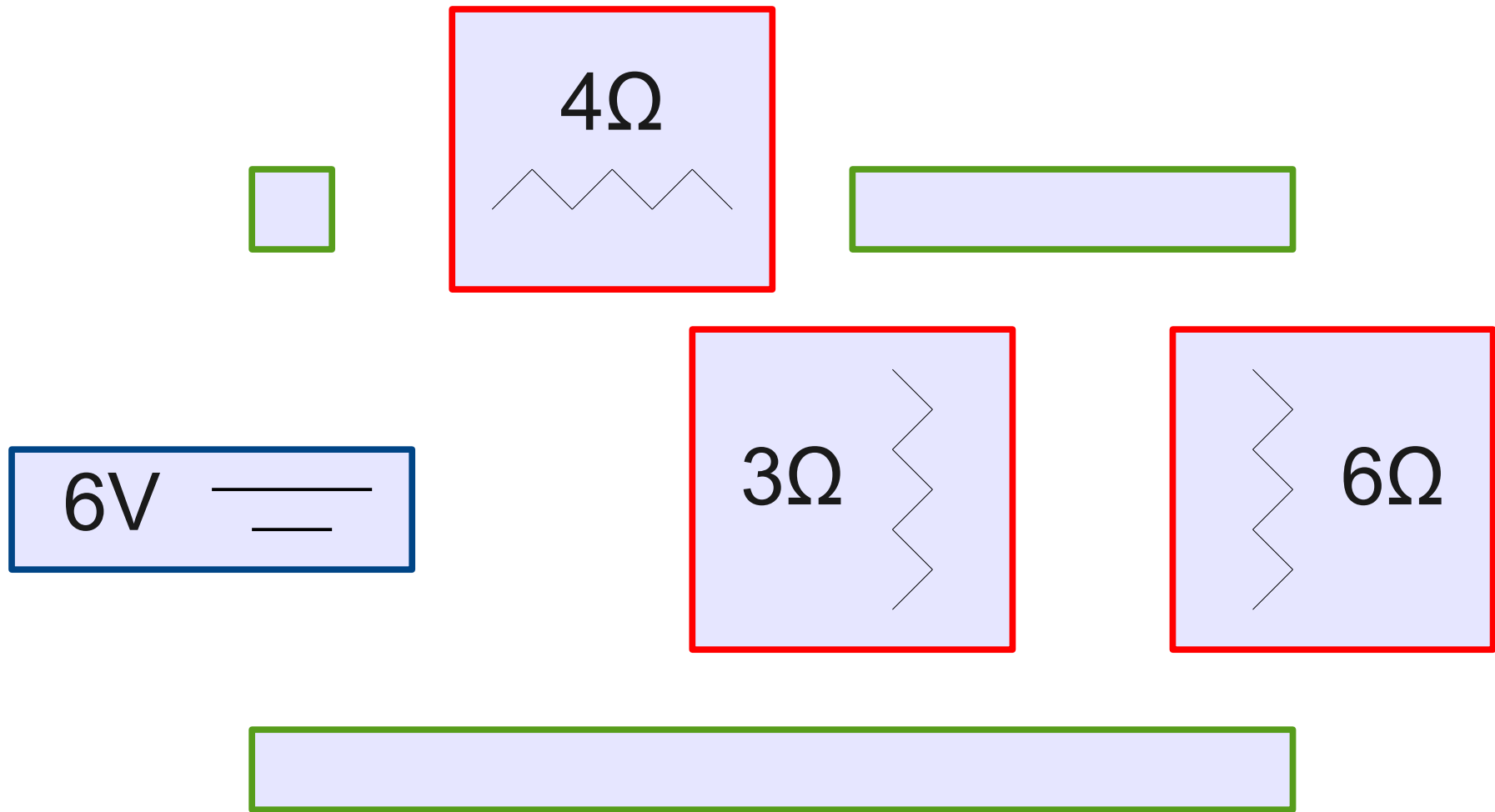
- John Backus,
interviewed in 2006

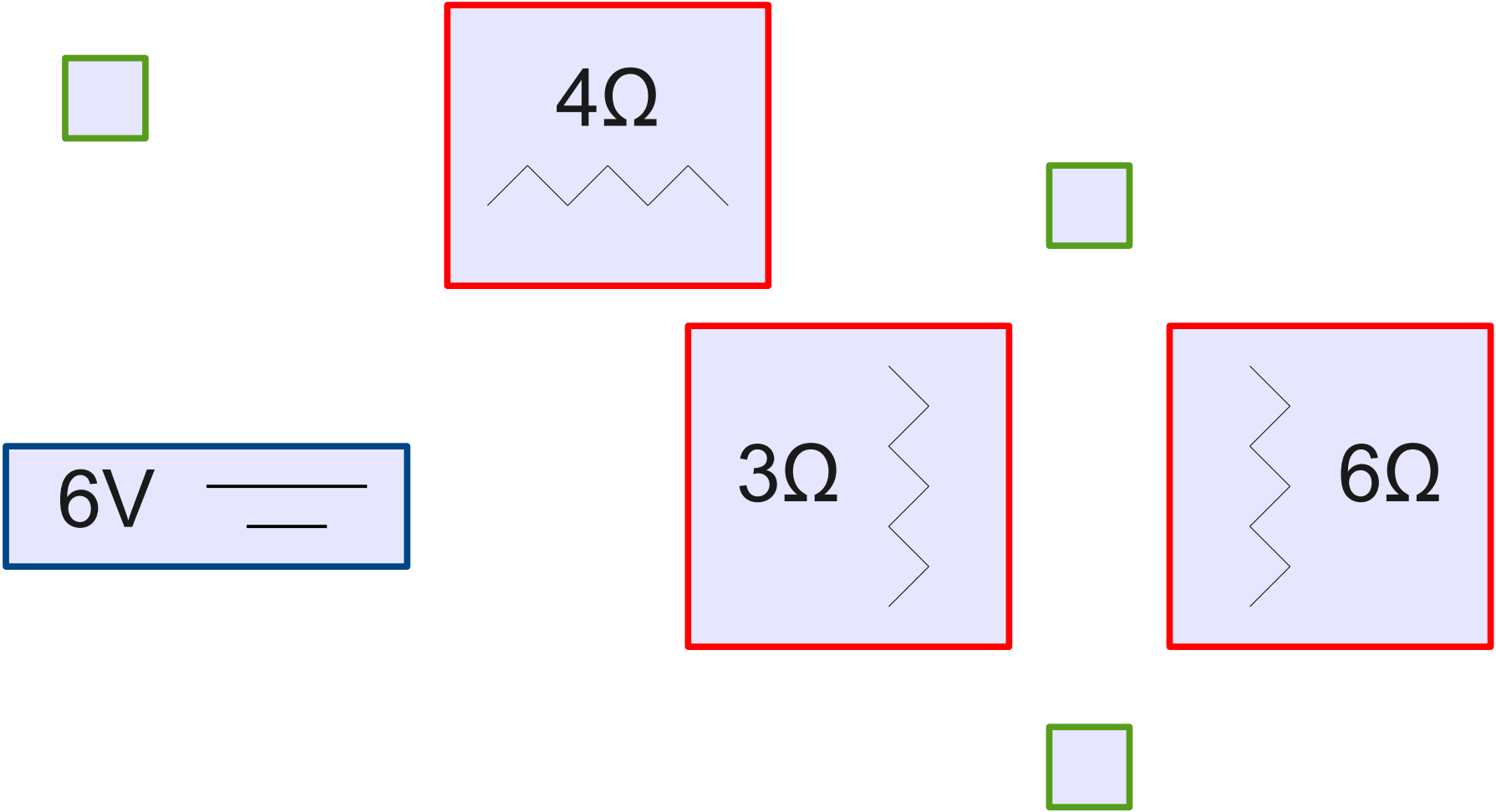


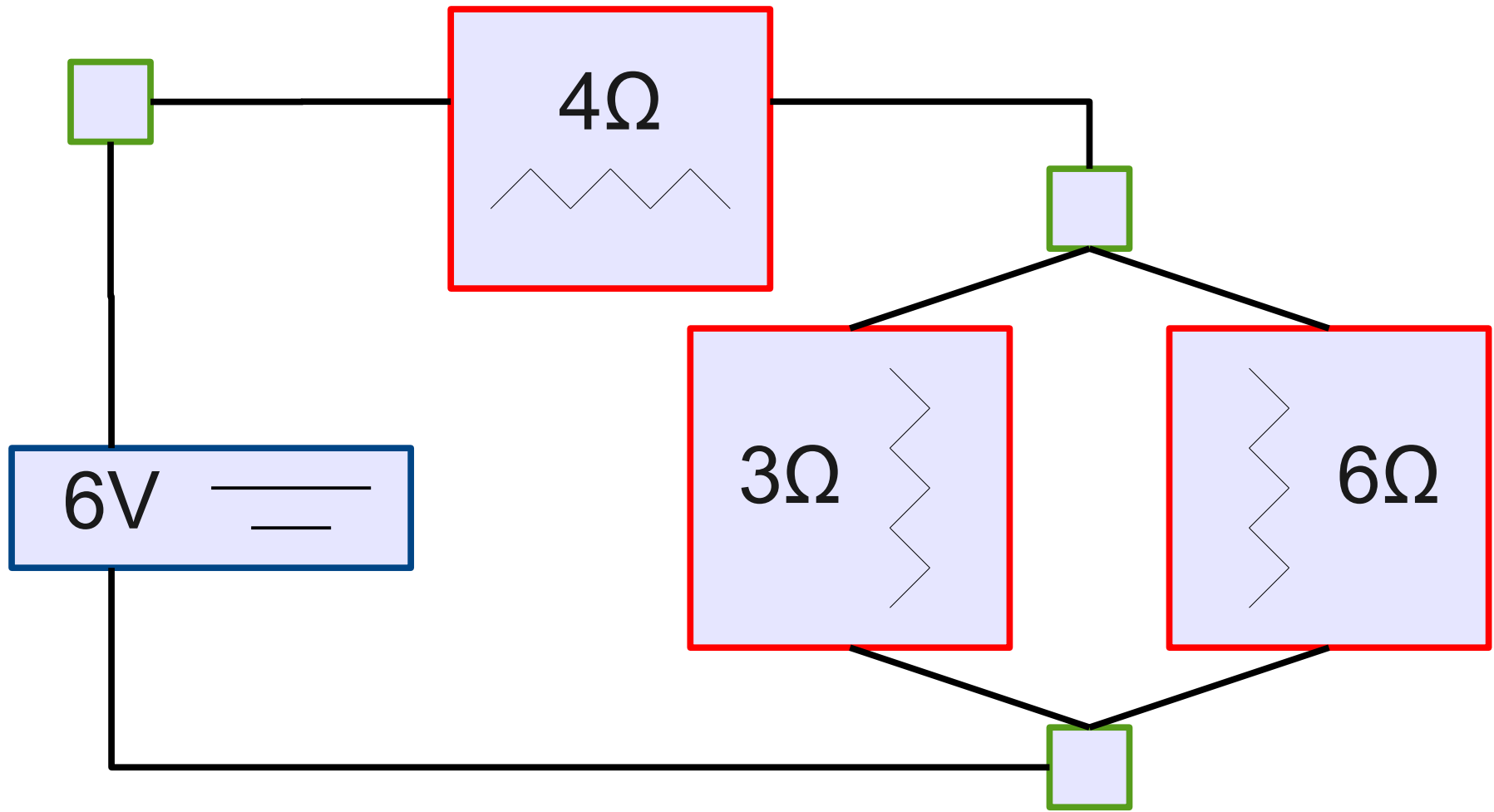


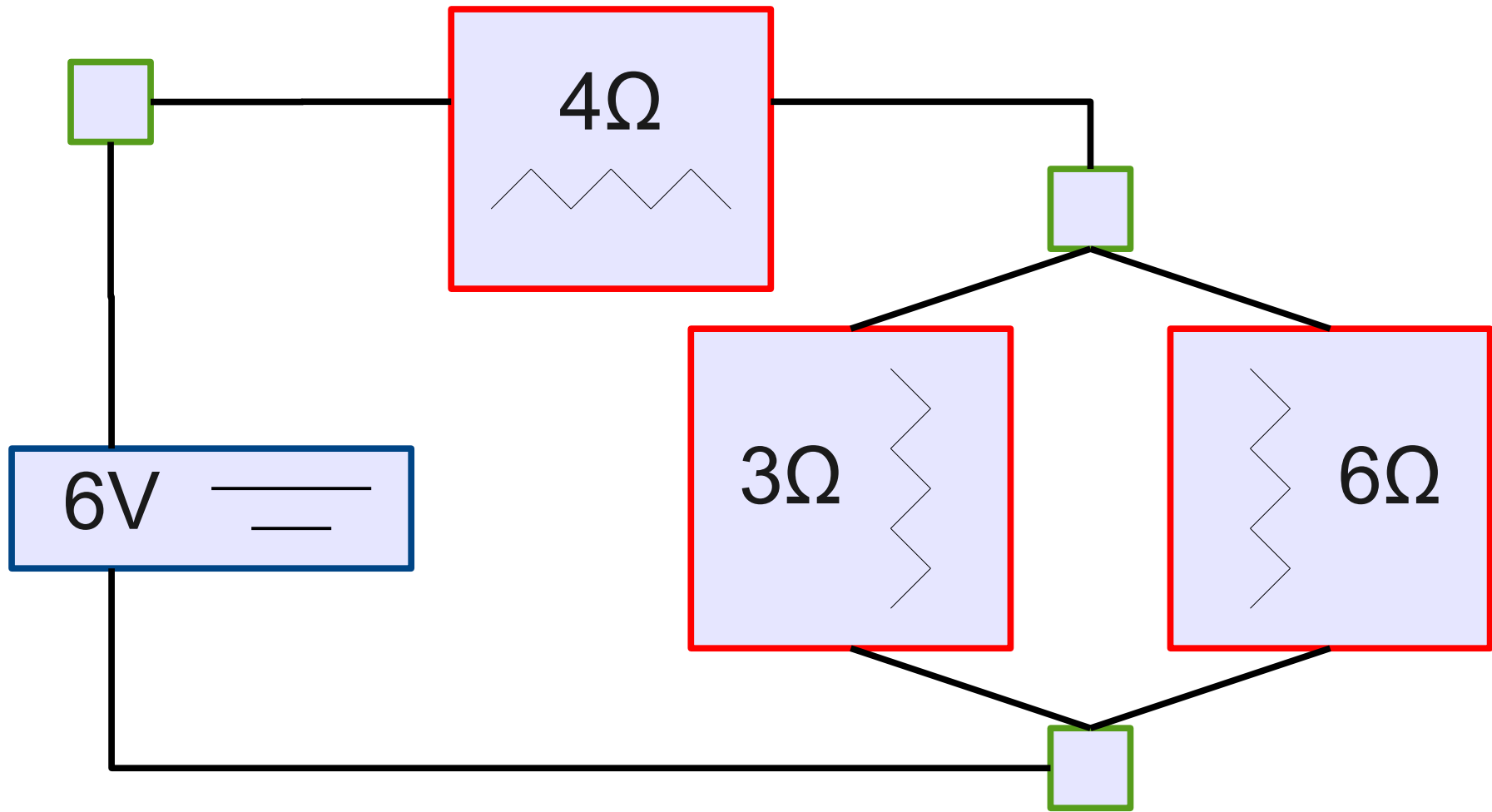




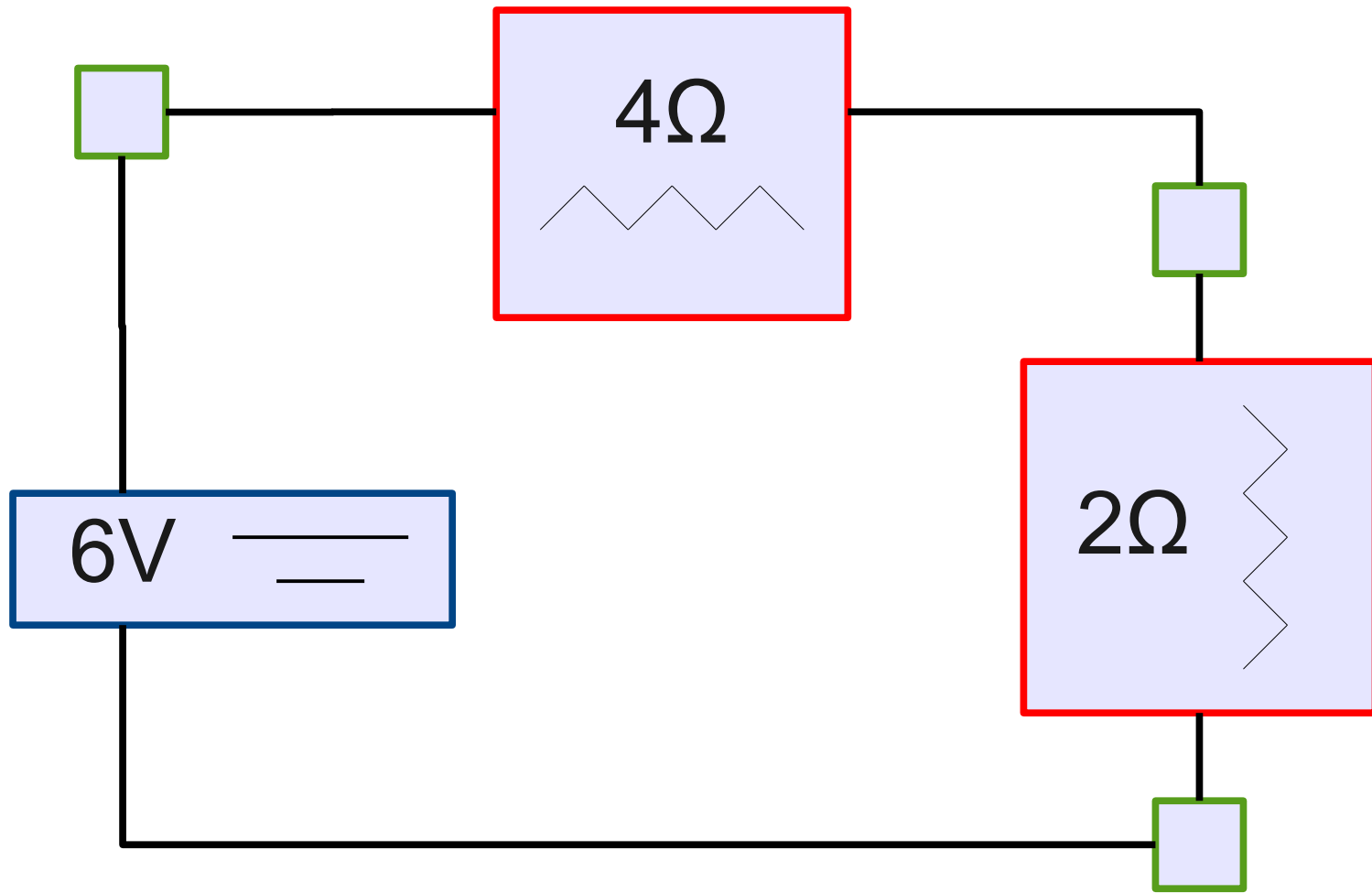




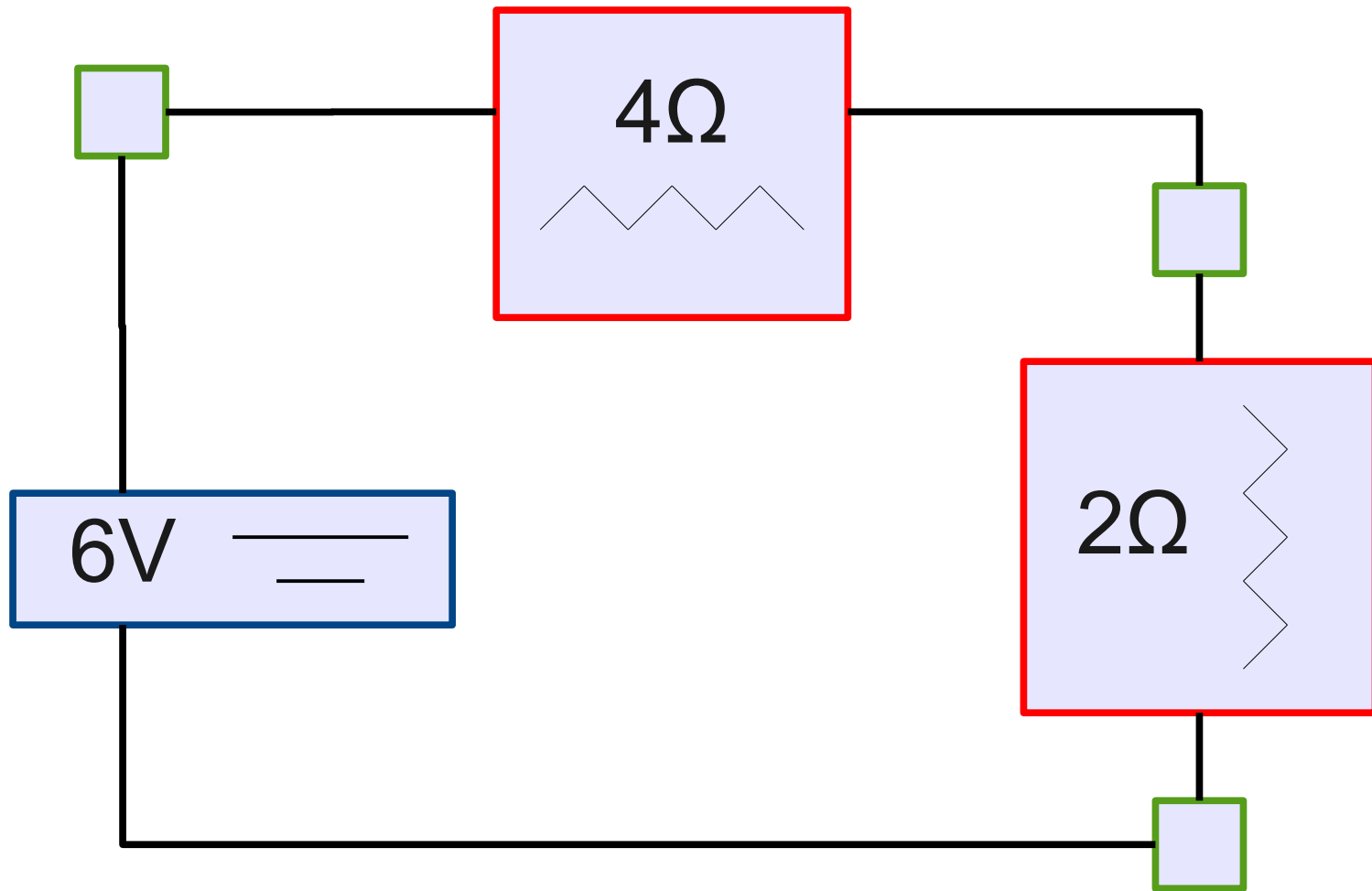


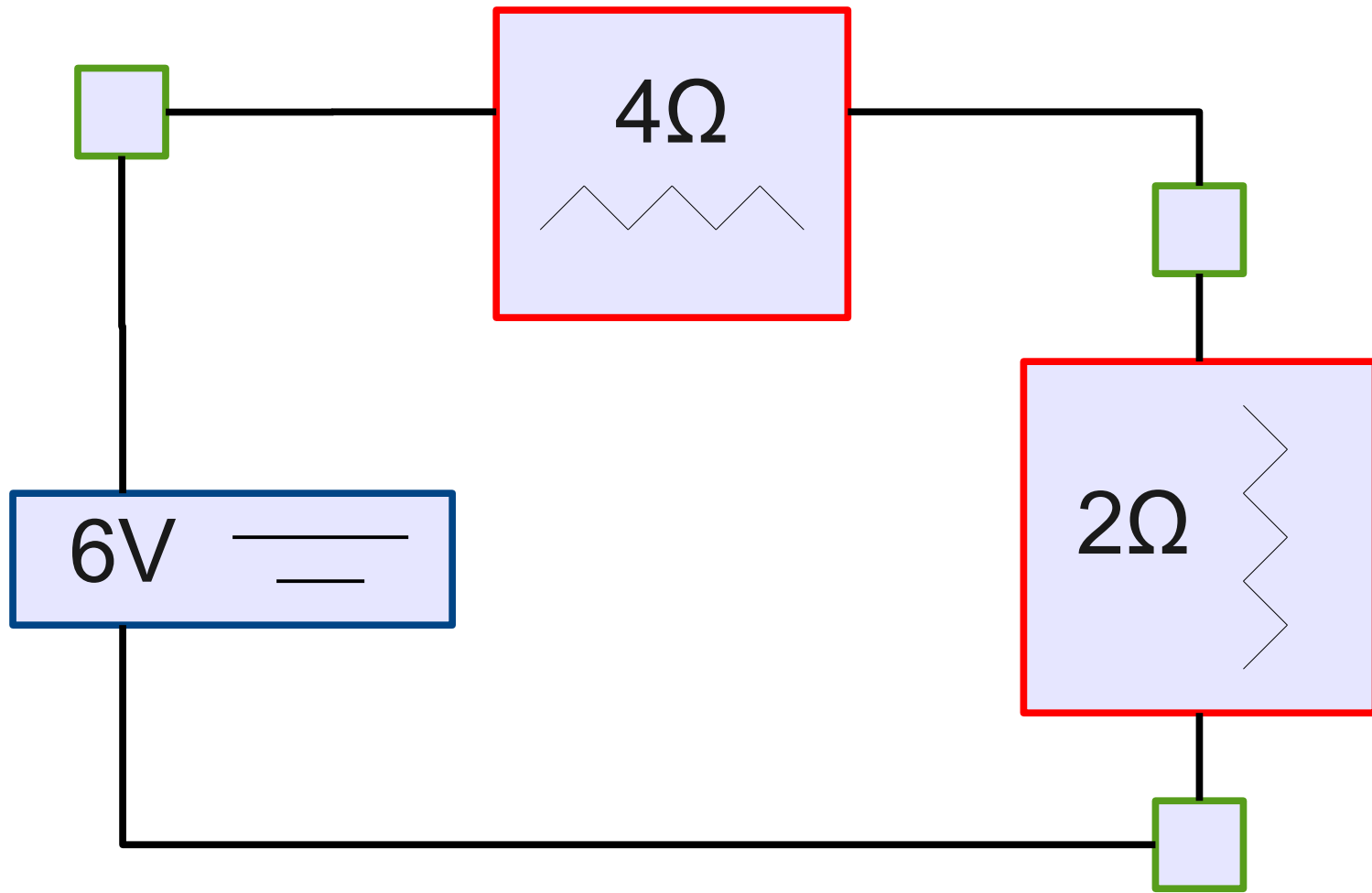


$$\frac{1}{\frac{1}{3\Omega} + \frac{1}{6\Omega}} = 2\Omega$$

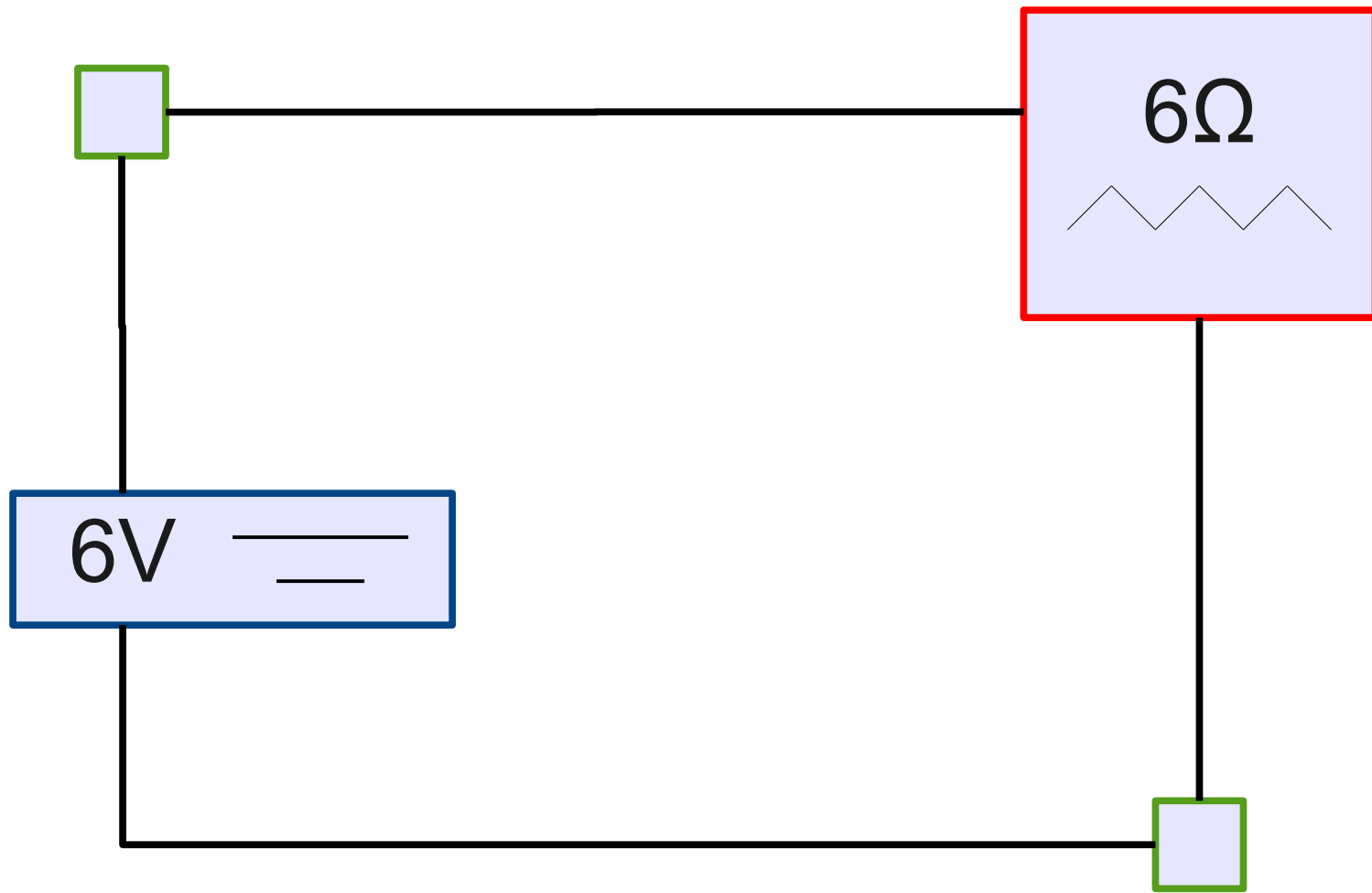


$$\frac{1}{\frac{1}{3\Omega} + \frac{1}{6\Omega}} = 2\Omega$$

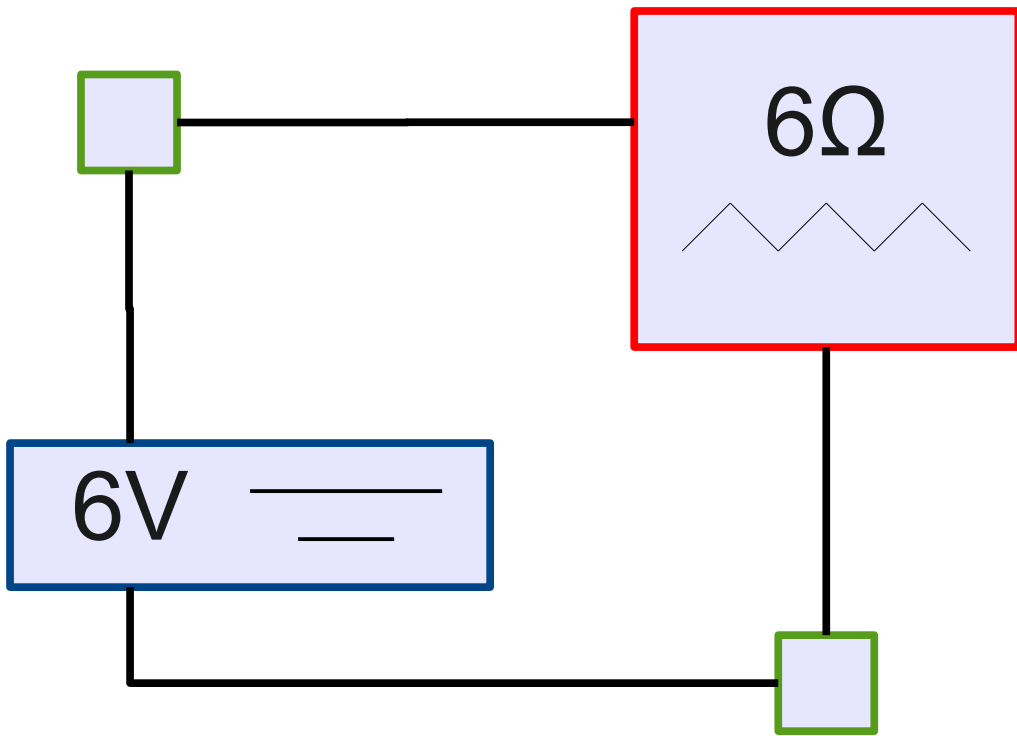


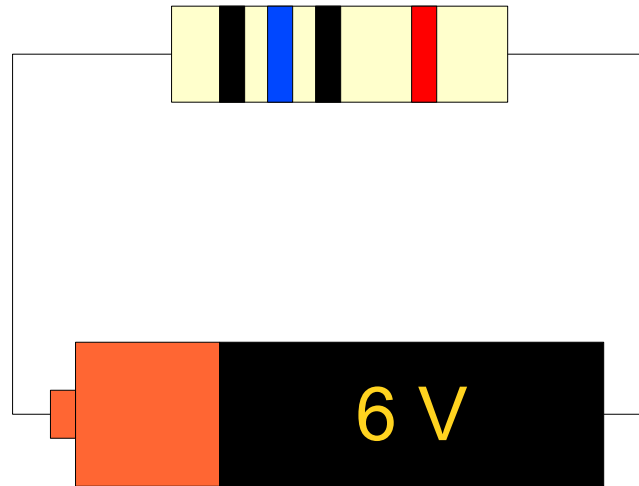
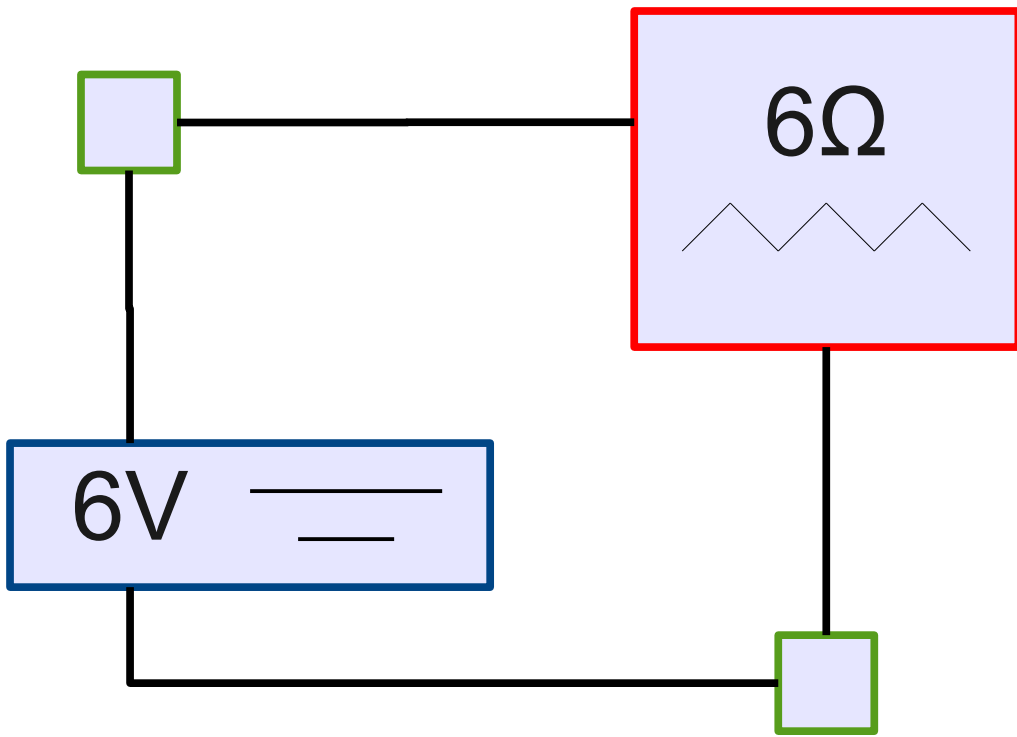


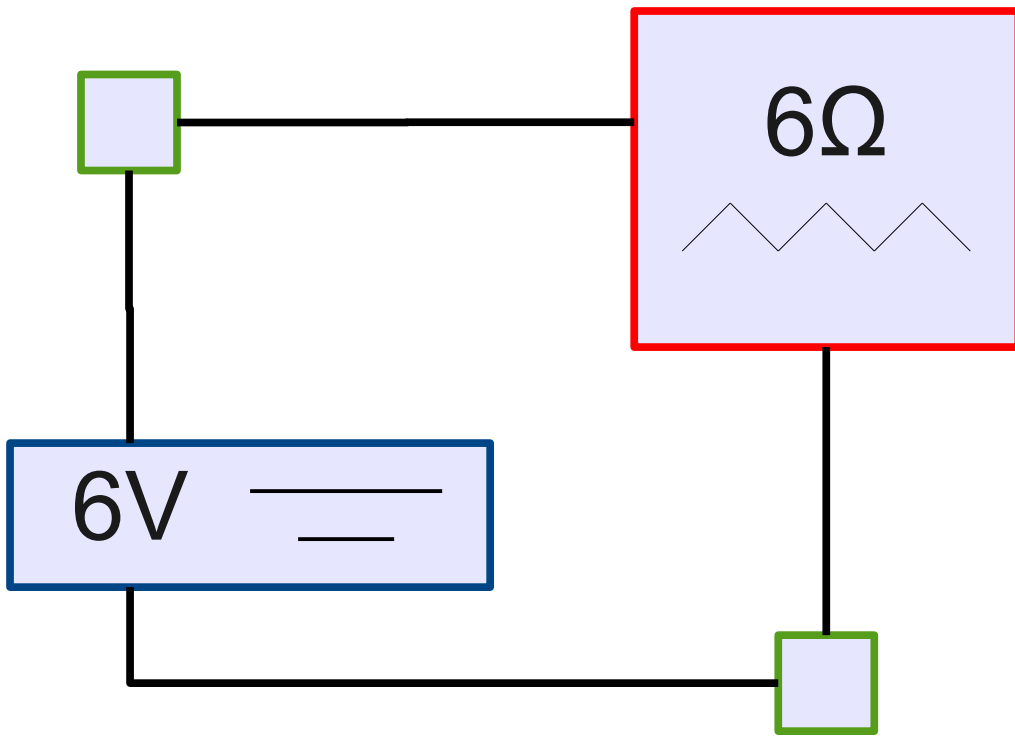
$$4\Omega + 2\Omega = 6\Omega$$



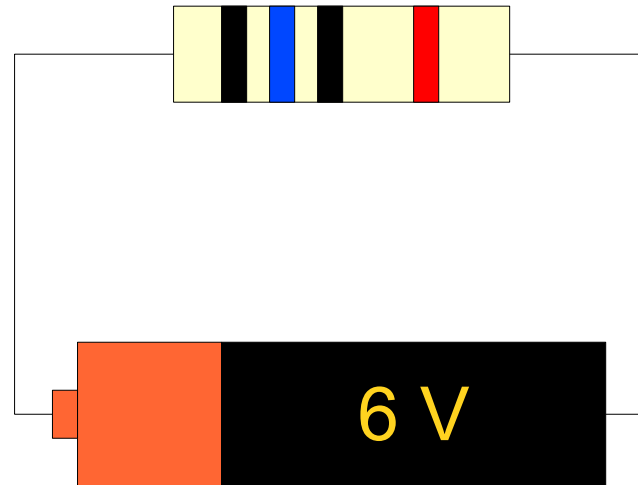
$$4\Omega + 2\Omega = 6\Omega$$

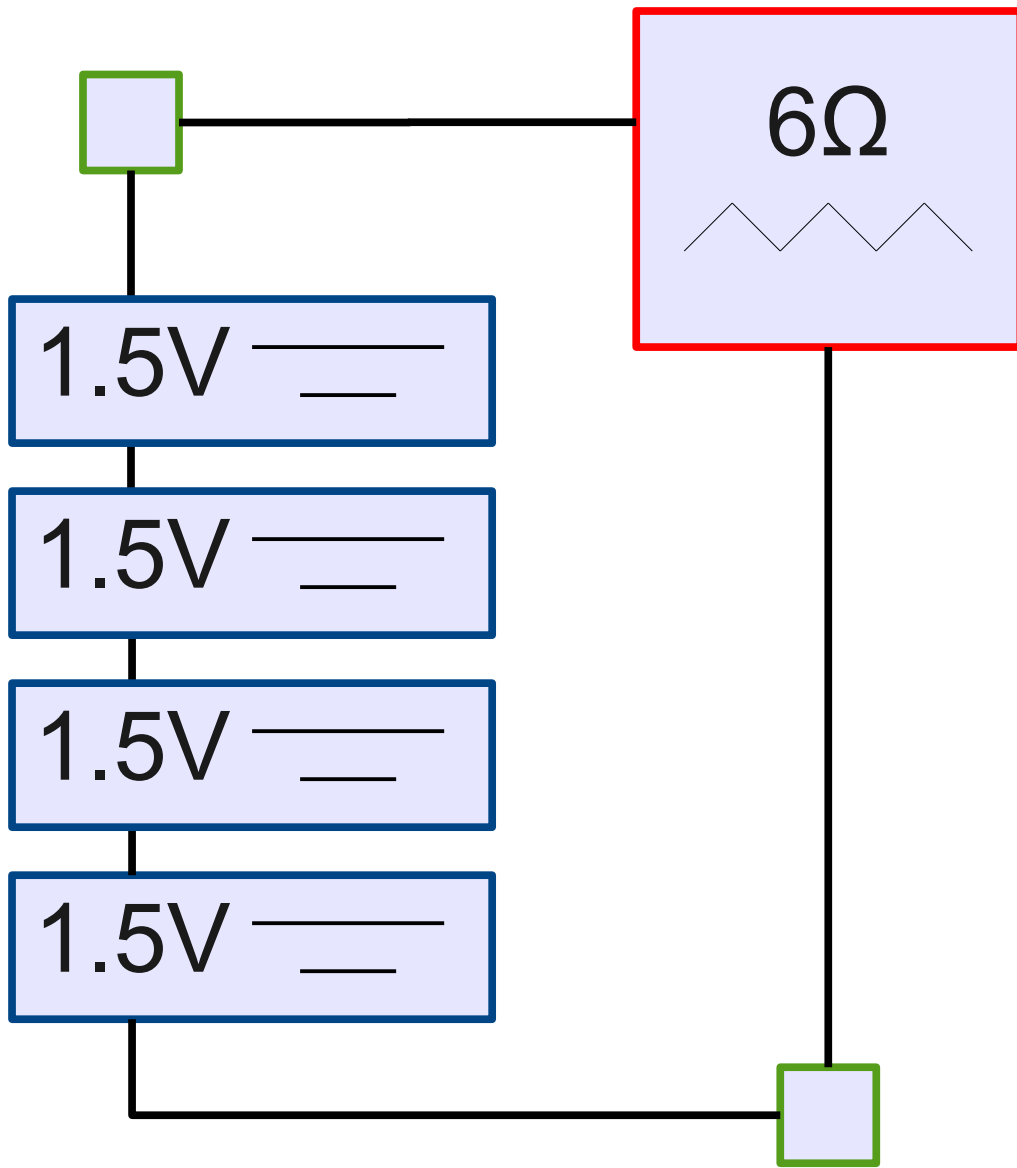


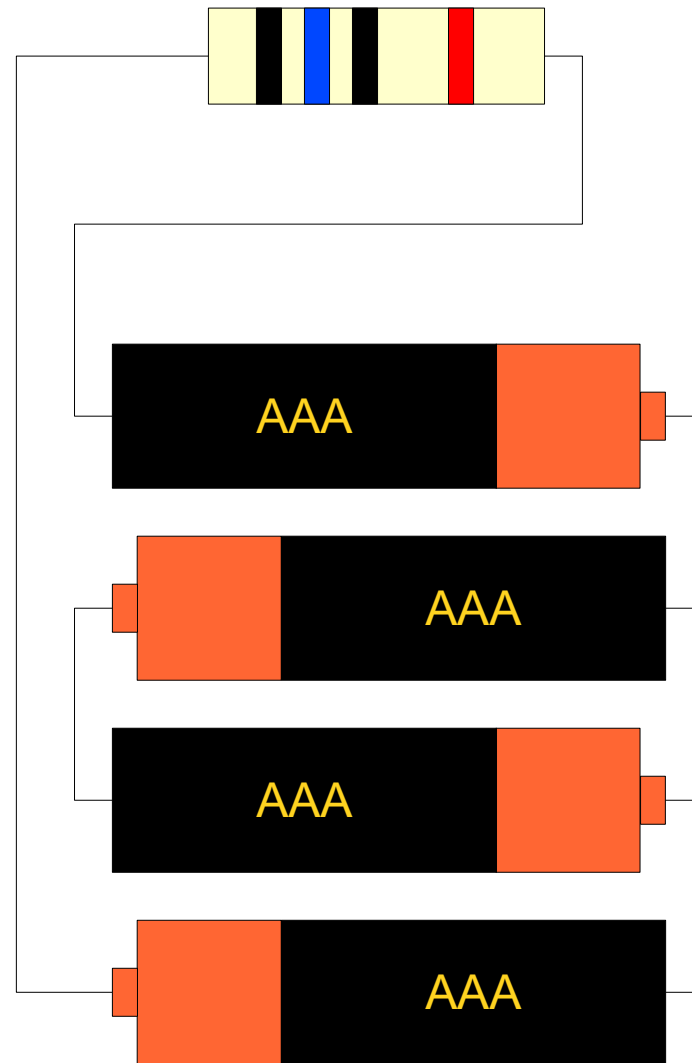
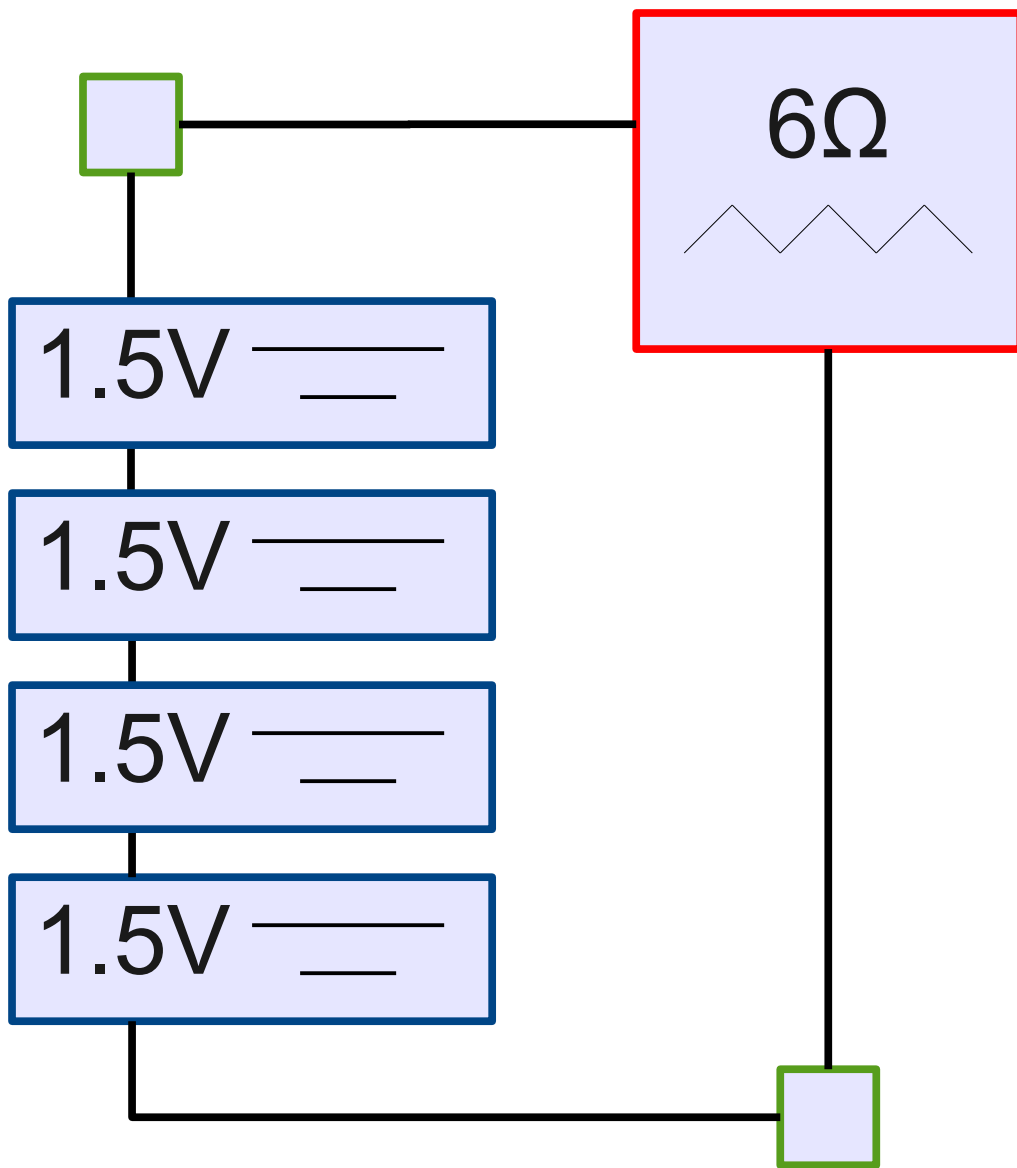


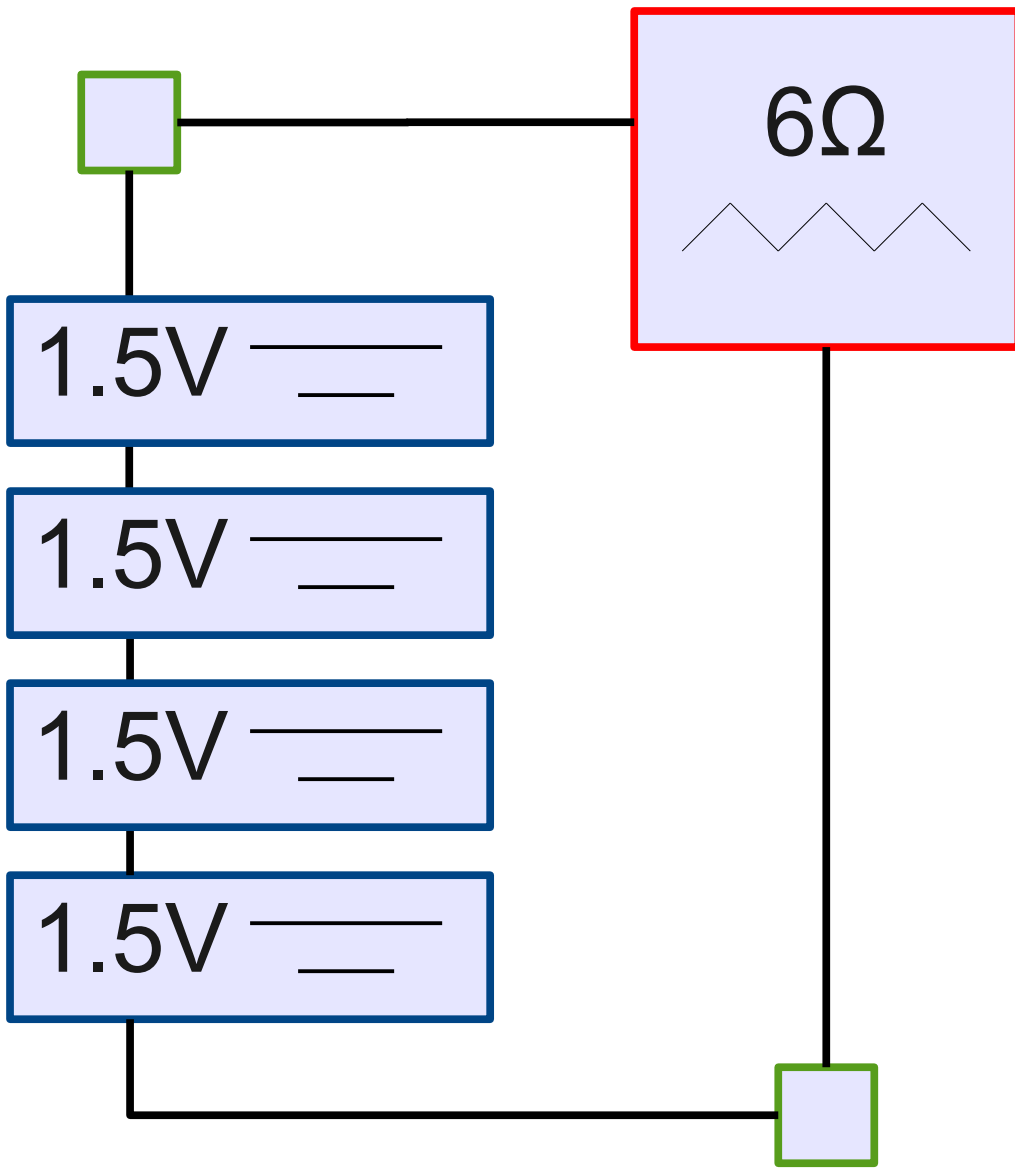


Total Cost: \$4.75

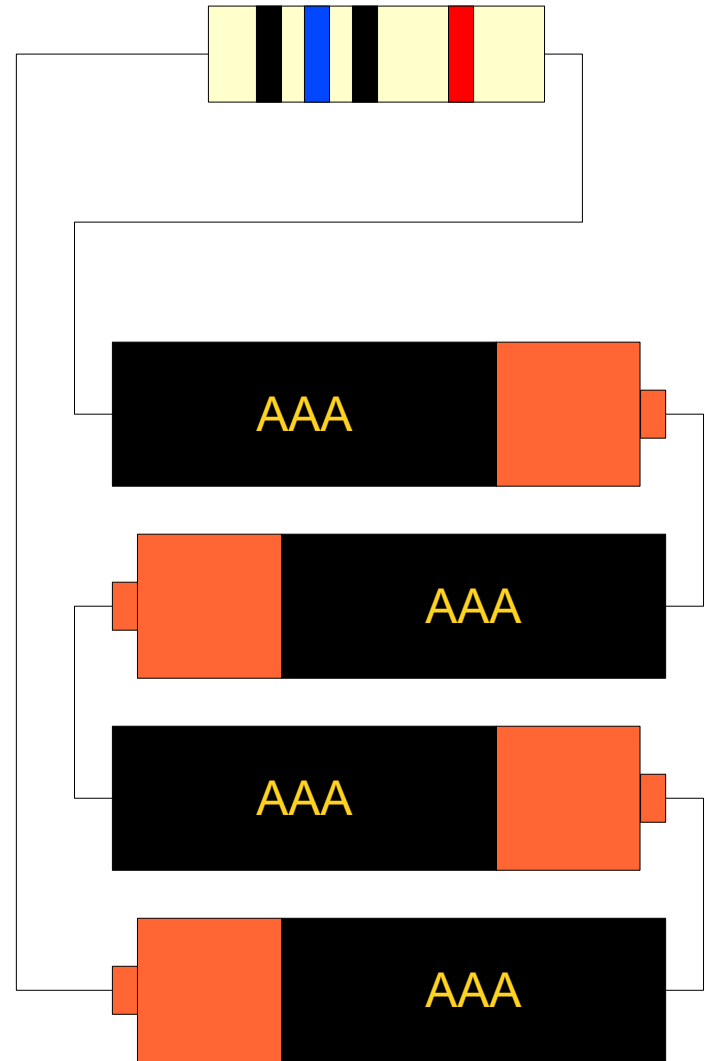








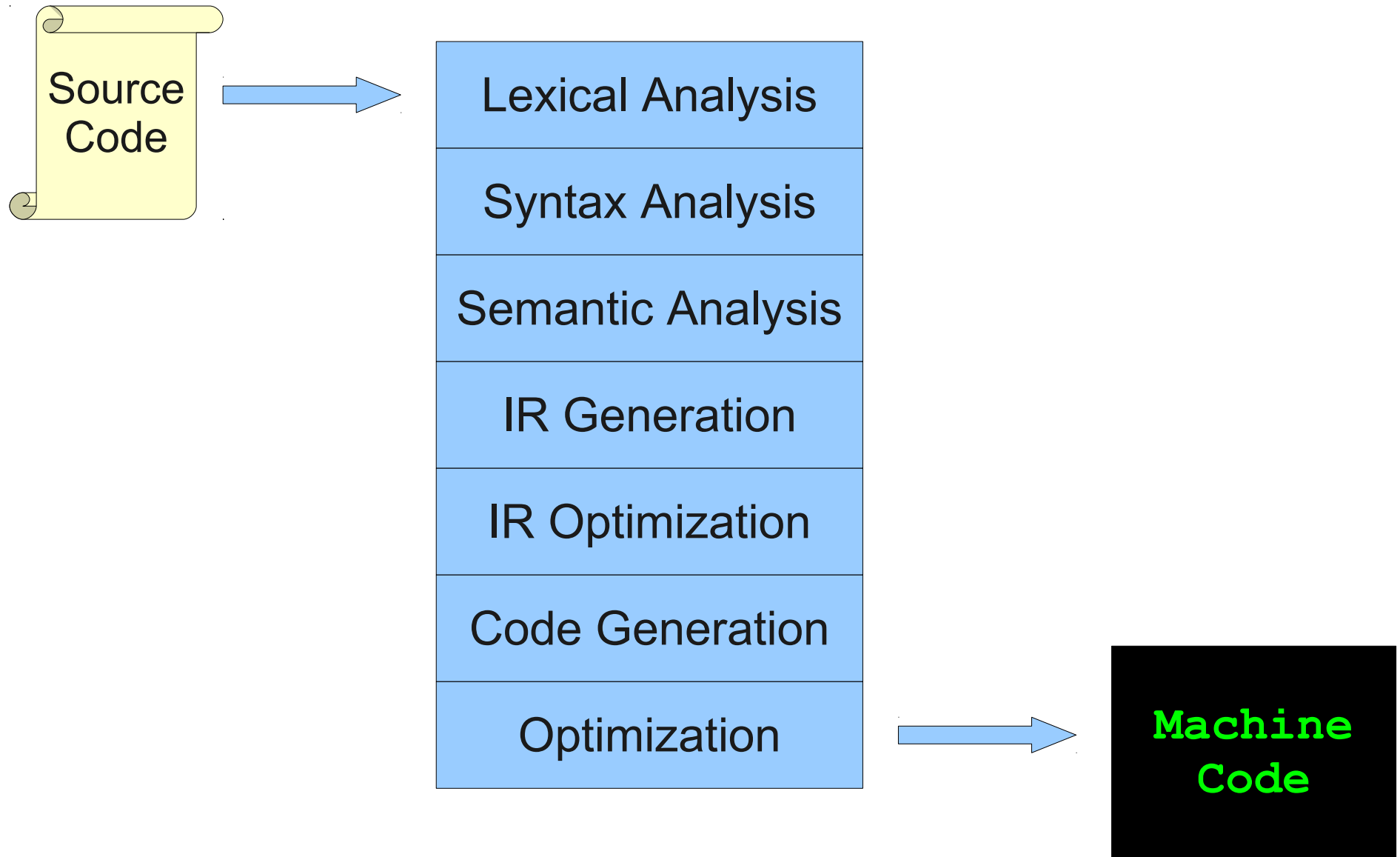
Total Cost: \$1.00



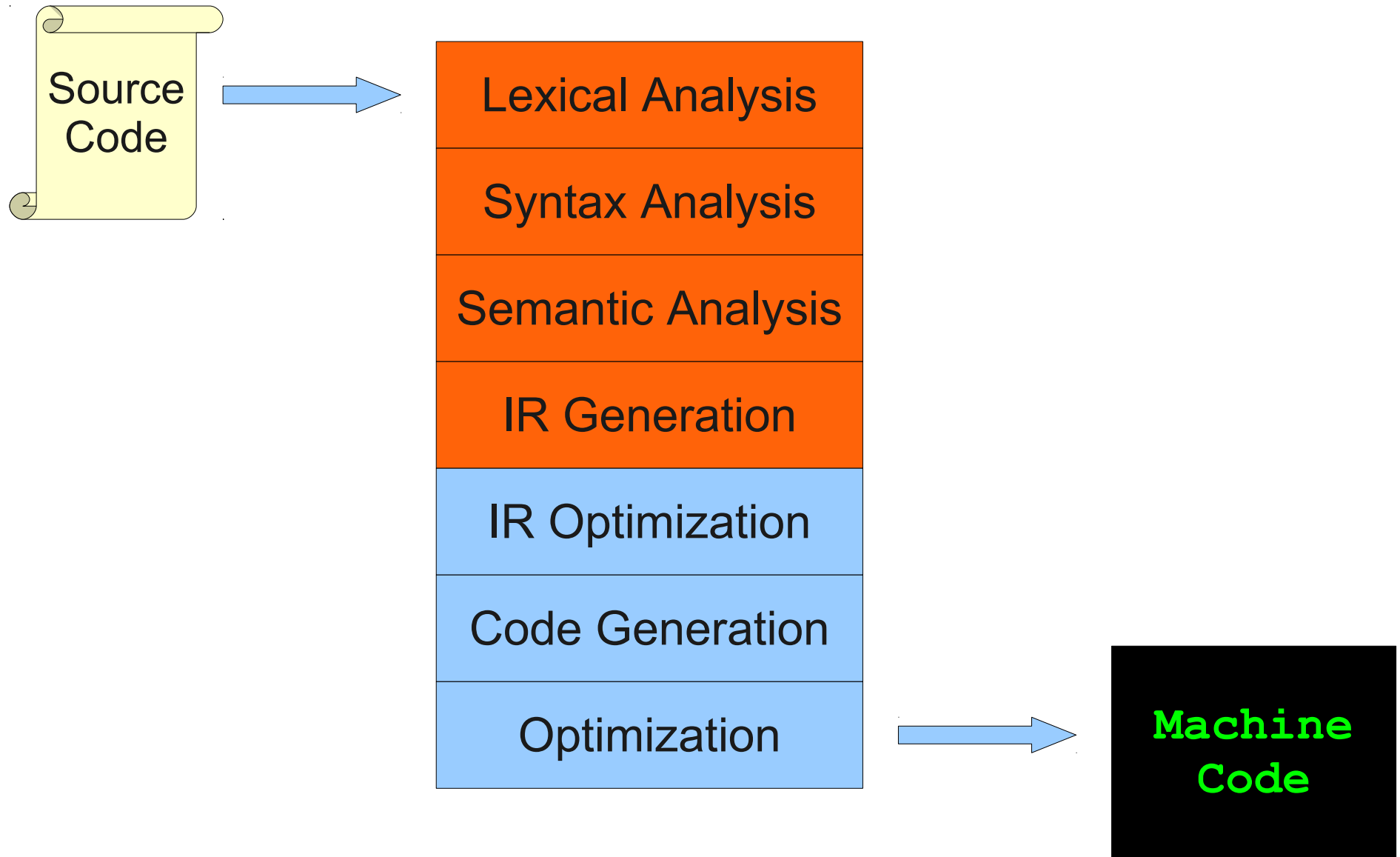
From Description to Implementation

- **Lexical analysis:** Identify logical pieces of description
- **Syntax analysis:** Identify how those pieces relate to each other.
- **Semantic analysis:** Identify the meaning of those relations.
- **IR Optimization:** Simplify the intended structure.
- **Code Generation:** Fabricate the structure.
- **Optimization:** Improve the resulting structure.

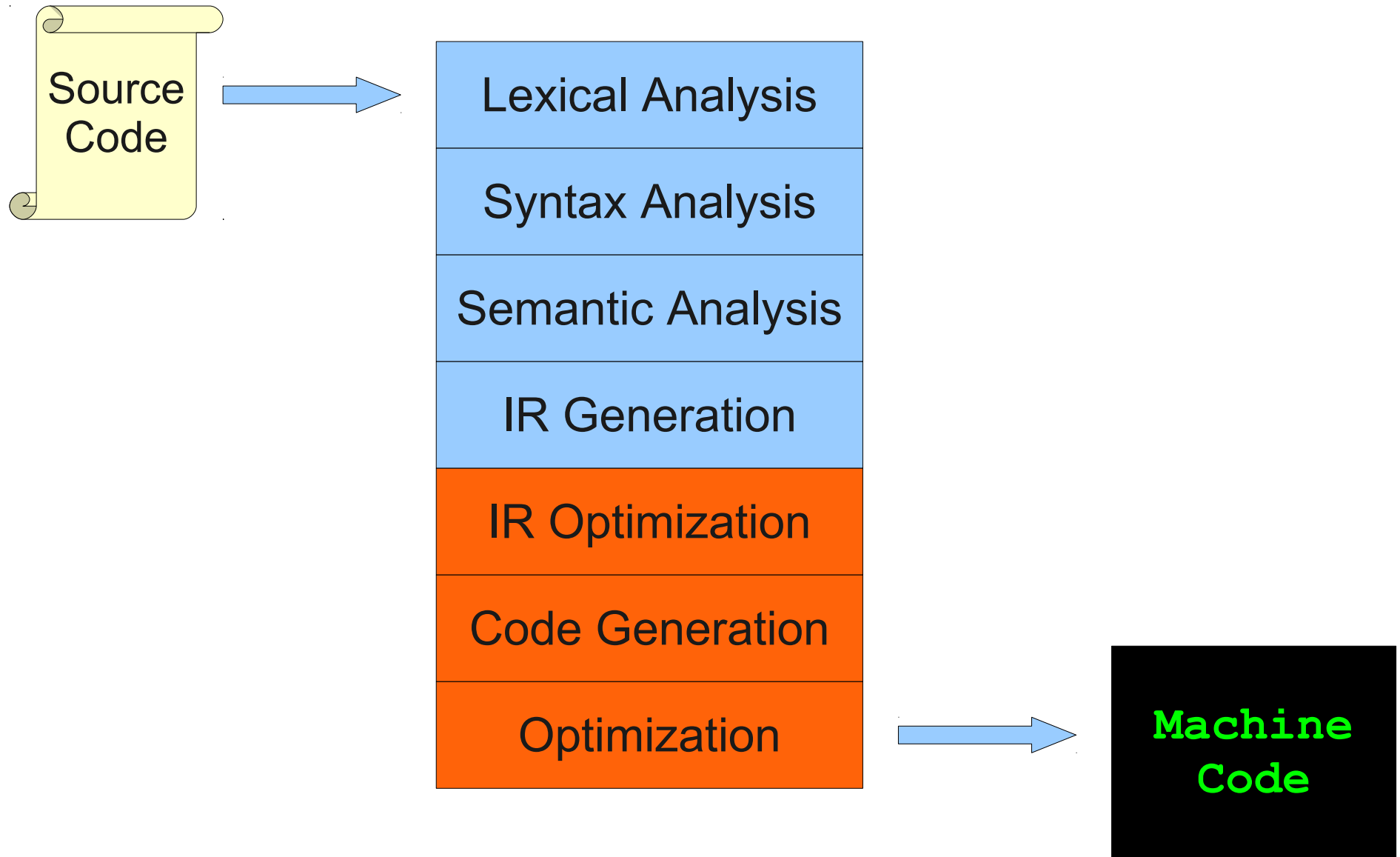
The Structure of a Modern Compiler



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

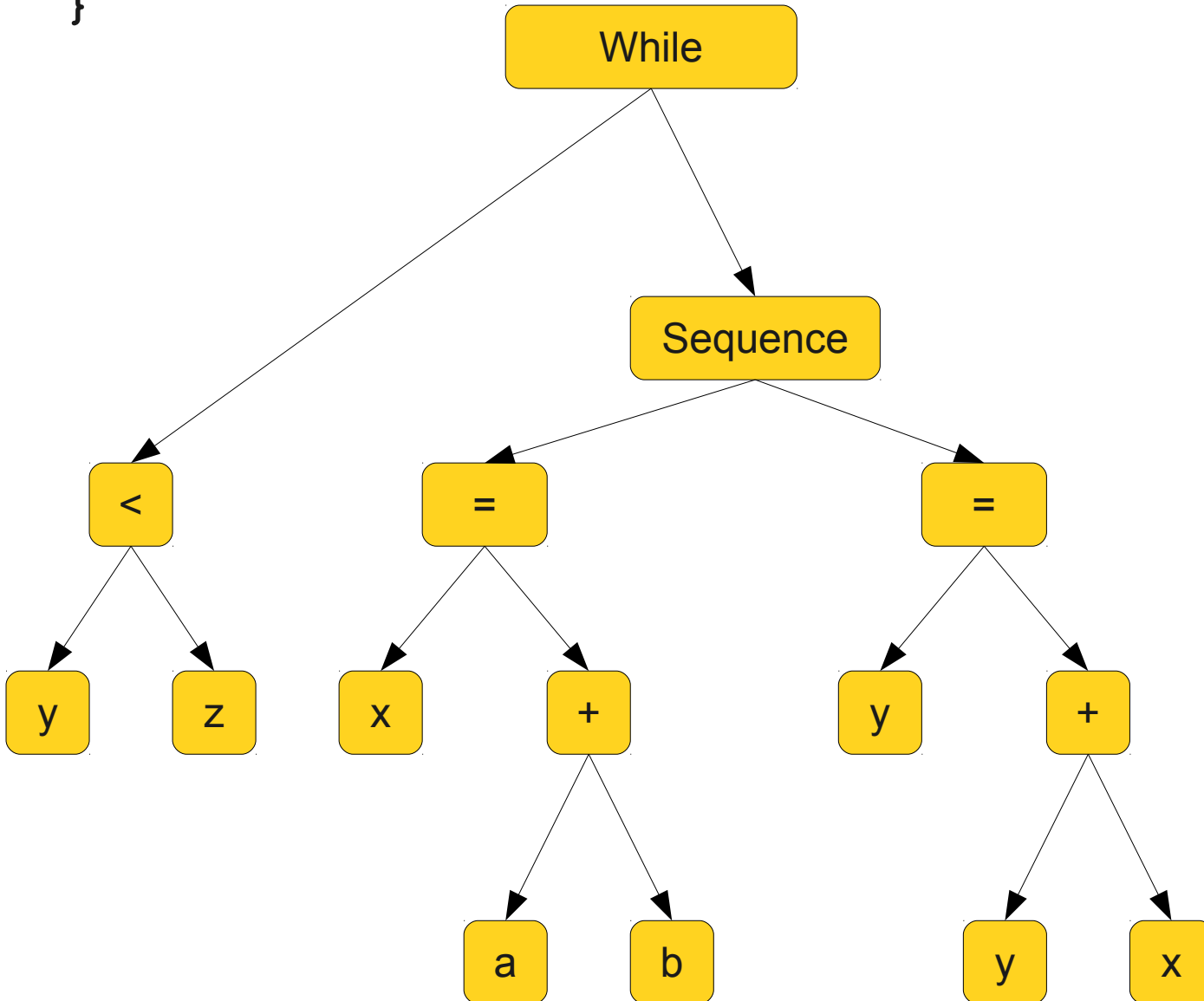
IR Generation

IR Optimization

Code Generation

Optimization

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



Lexical Analysis

Syntax Analysis

Semantic Analysis

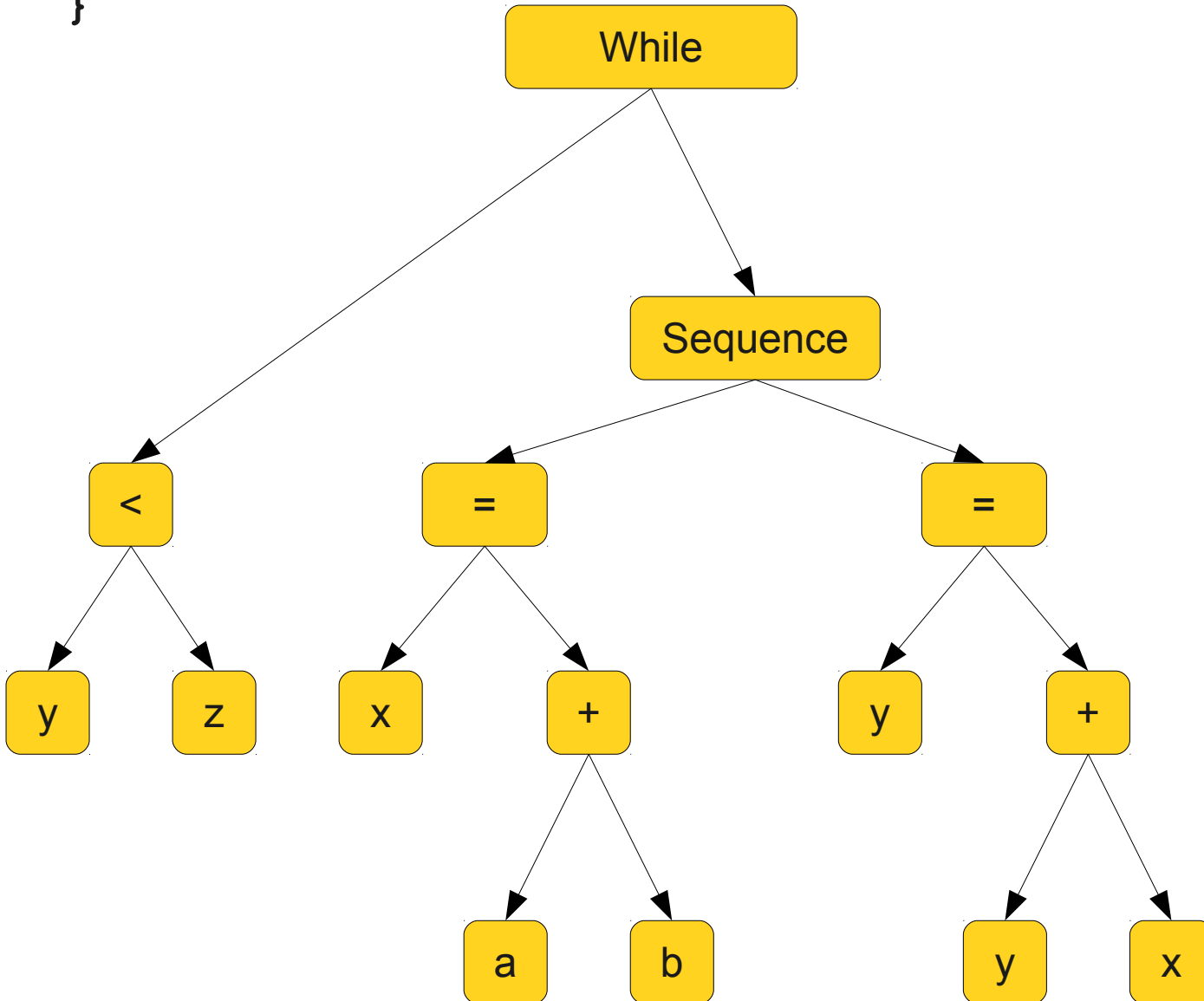
IR Generation

IR Optimization

Code Generation

Optimization

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



Lexical Analysis

Syntax Analysis

Semantic Analysis

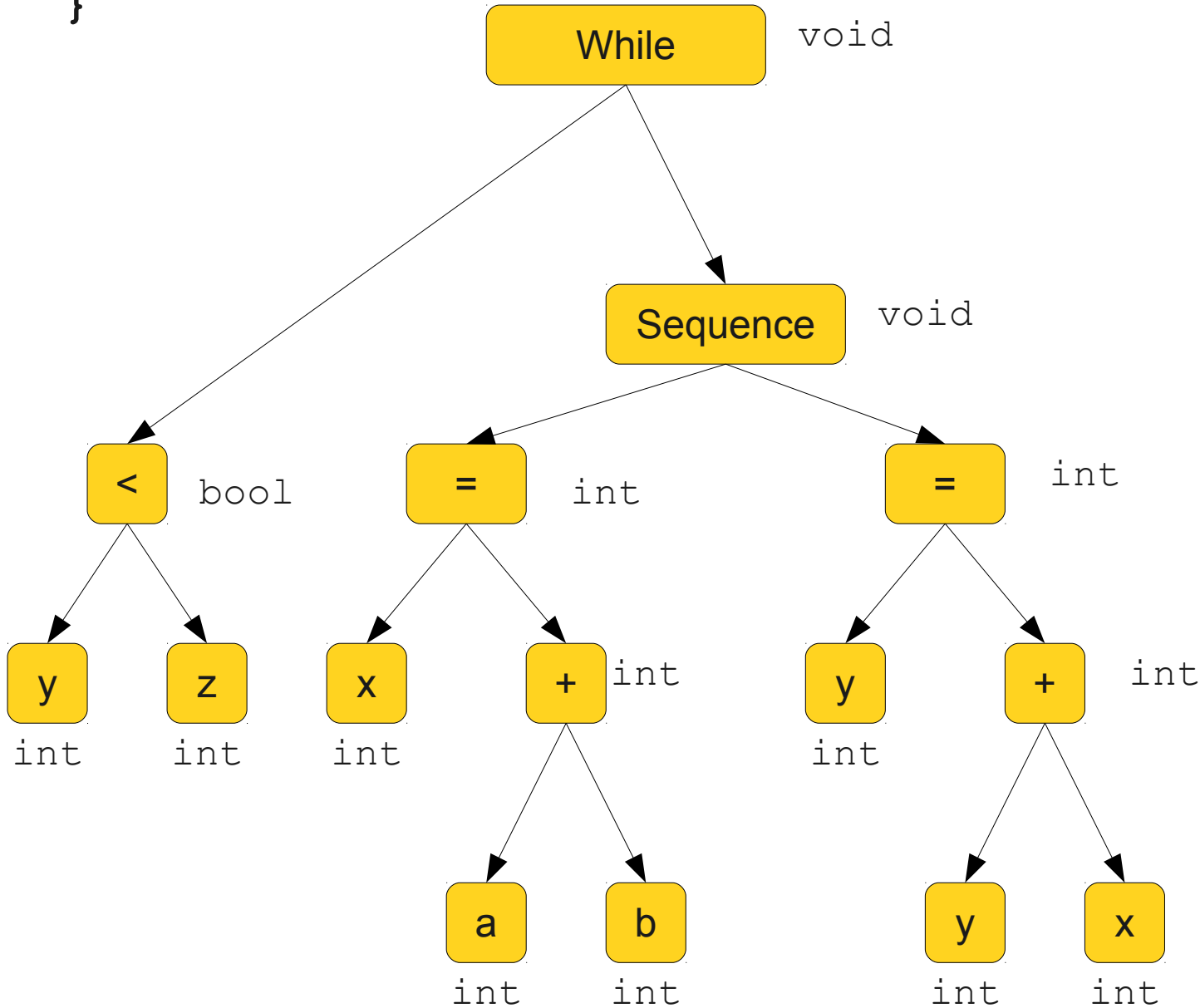
IR Generation

IR Optimization

Code Generation

Optimization

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



Lexical Analysis

Syntax Analysis

Semantic Analysis

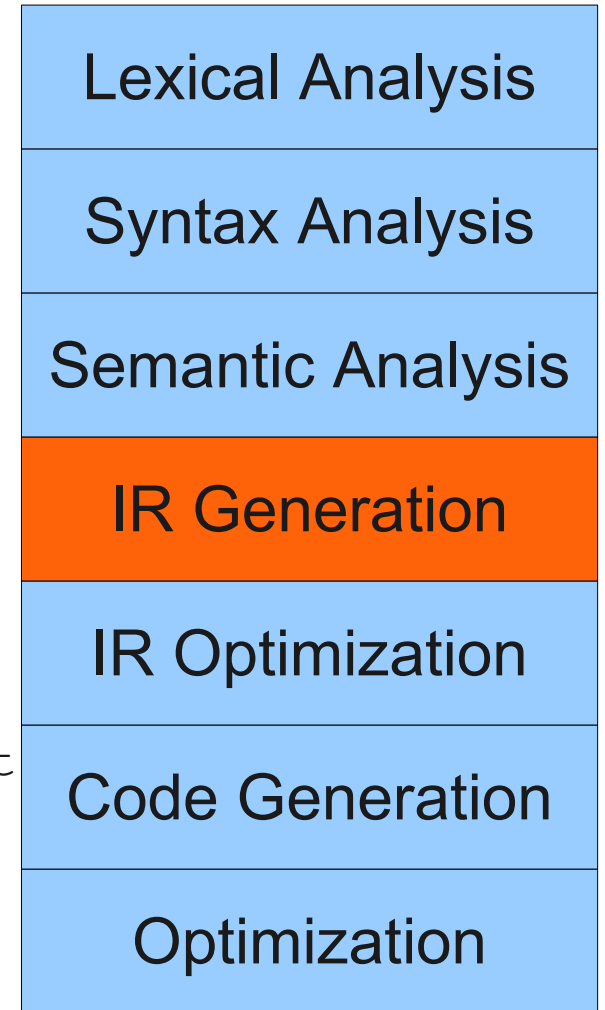
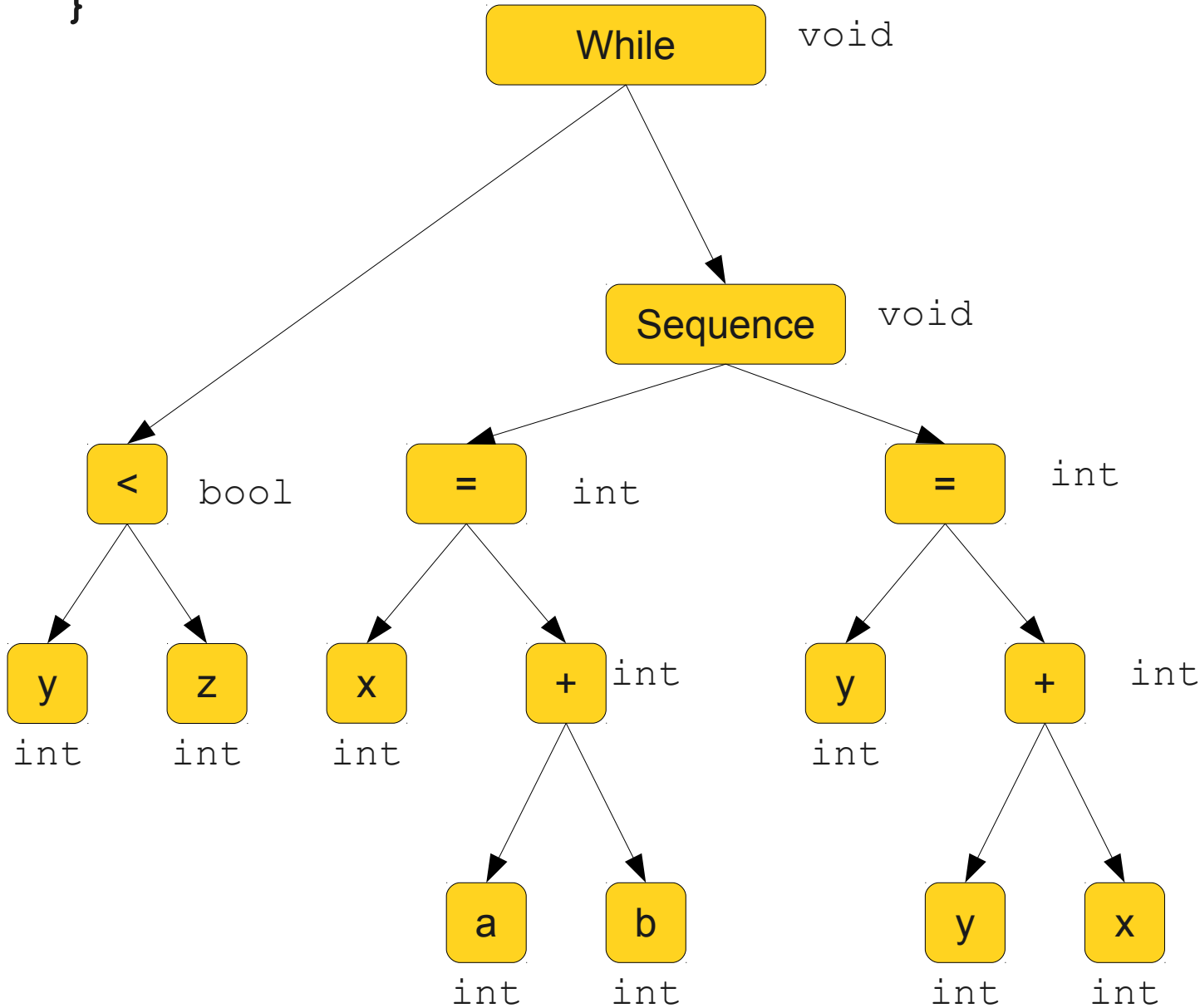
IR Generation

IR Optimization

Code Generation

Optimization

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



```
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;  
}
```

```
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
```

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
```

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

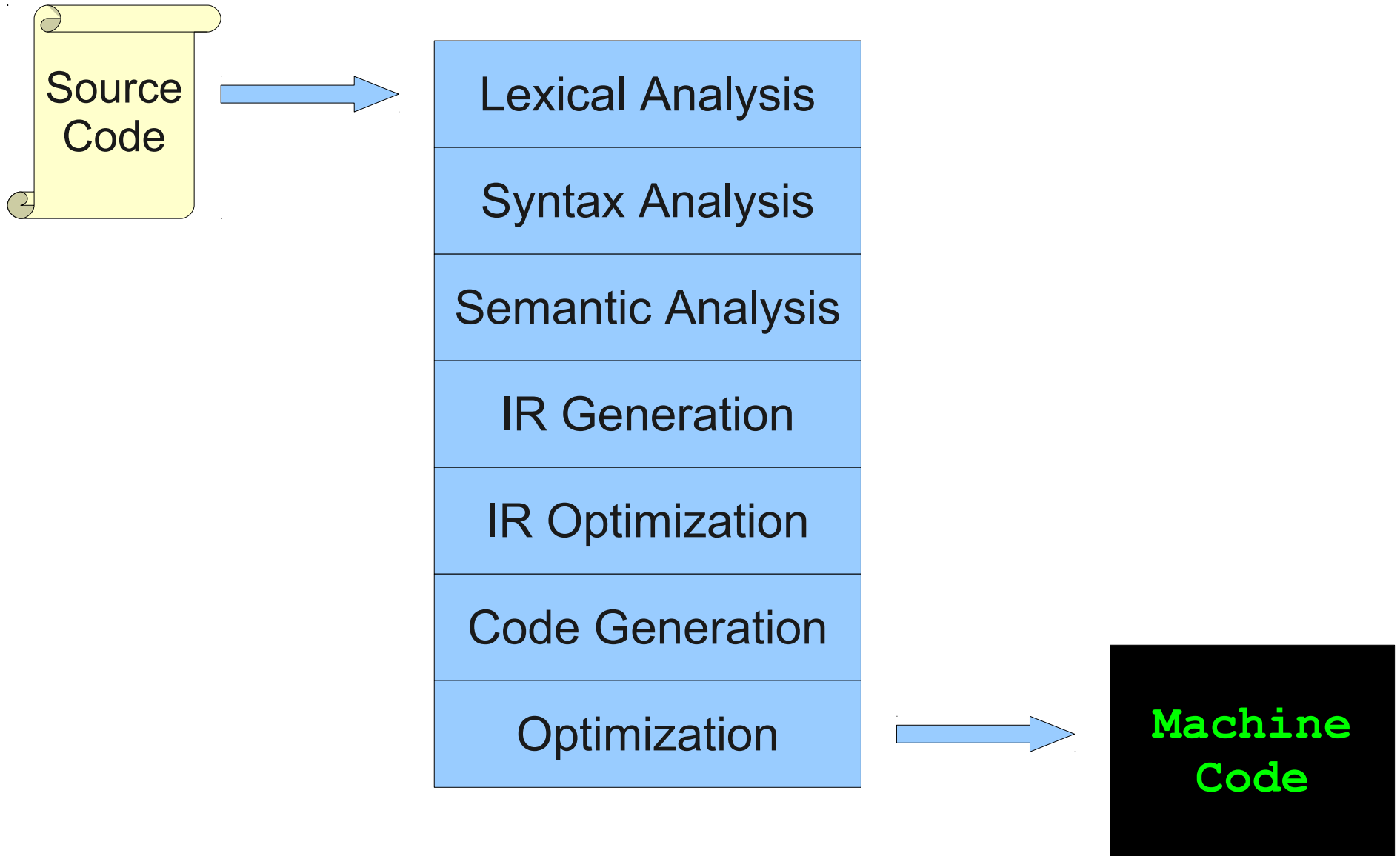
Code Generation

Optimization

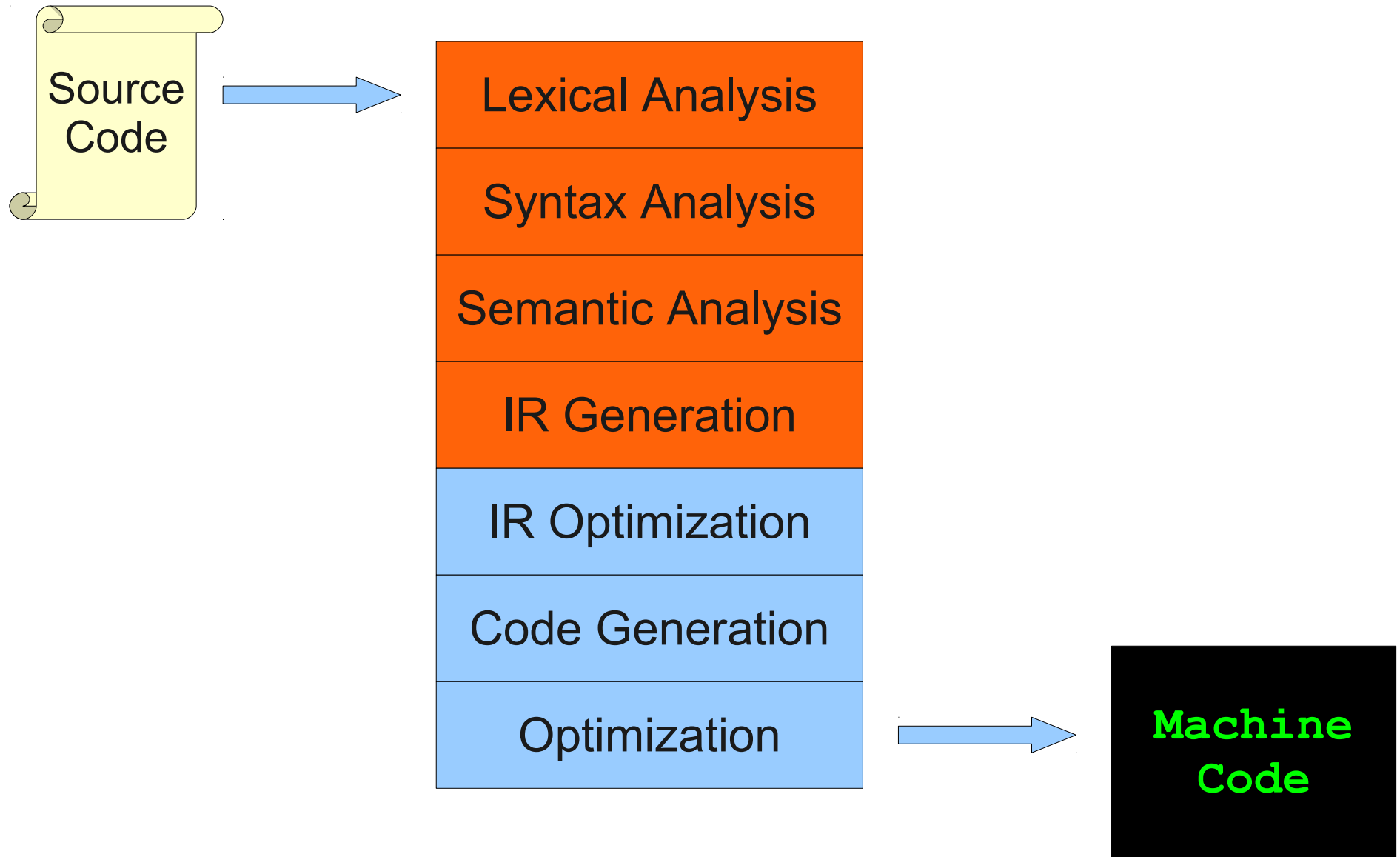
The Course Project: **Decaf**

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

Programming Assignments



Programming Assignments



Next Time...

