CSE110A: Compilers

April 3, 2024

- *Compiler Overview*
  - What is a compiler
  - What are the different stages of a compiler
    - Frontend
    - Intermediate
    - Backend

Announcements

- Homework 1 will be released Monday
  - You’ll have 10 days to do it

- Piazza is up; please enroll!

- Coming soon:
  - TA and tutor office hours

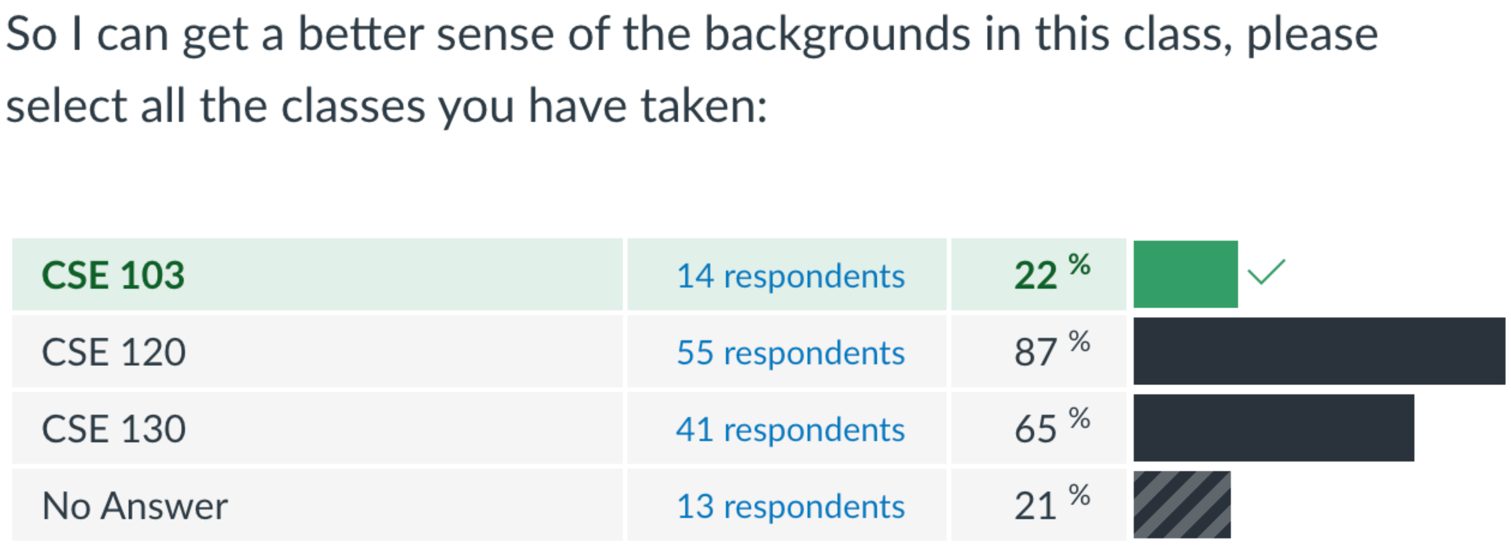
- I’ll have office hours tomorrow:
  - Room E2 233
  - I’ll post sign up sheet around noon tomorrow

Announcements

- Midterm moved to May 6
    - Enough disruptions on May 3 that it just made more sense to move it
- Did anyone set up a discord?

Quiz

Background

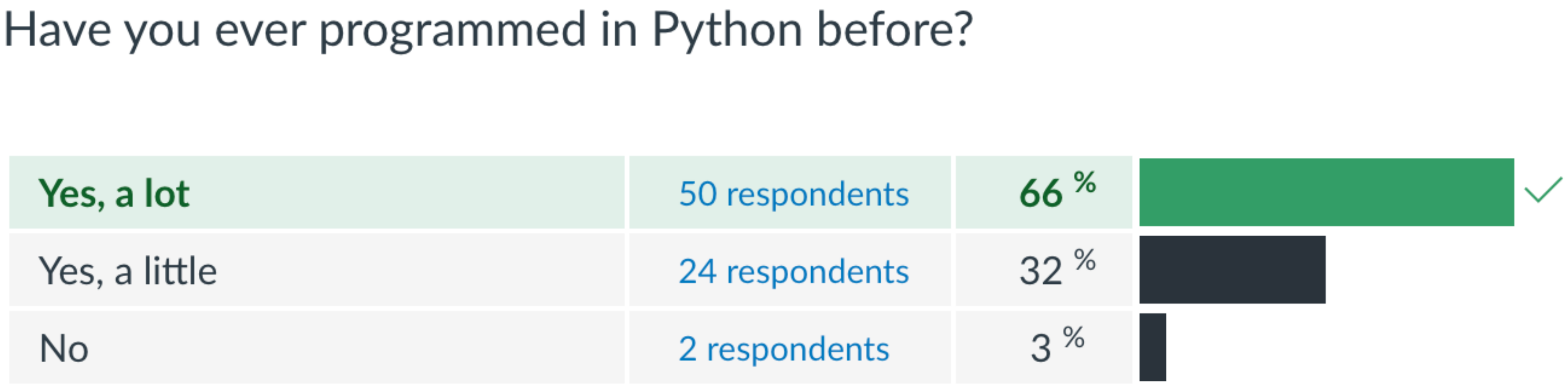


So I can get a better sense of the backgrounds in this class, please select all the classes you have taken:

CSE 103	14 respondents	**22 %**	✓
CSE 120	55 respondents	87 %	
CSE 130	41 respondents	65 %	
No Answer	13 respondents	21 %	

2% answered correctly

Background



Have you ever programmed in Python before?

Yes, a lot	50 respondents	**66 %**	✓
Yes, a little	24 respondents	32 %	
No	2 respondents	3 %	

It is worthwhile to learn!

https://www.tiobe.com/tiobe-index/

Compiler features

Write a few sentences about some of the most useful features you use in a compiler (apart from actually producing the executable). Write a few sentences about what you wish compilers could do better.

Compiler features

Write a few sentences about some of the most useful features you use in a compiler (apart from actually producing the executable). Write a few sentences about what you wish compilers could do better.

Pros:
Optimizations
Error messages
Warnings
Compiler flags
debugging

Cons:
Can’t catch all errors
Compile time
Bad messages

What do people hope to get out of this class?

A few answers that I liked:

- ”reverse engineering as used in cyber security”
- “become a better software engineer”
- “take away some of the magic”
- “write more performant code”
- “theory of programming languages”

Quiz

- Thank you for all your thoughtful answers!

Schedule

- Introduction to compilers

- Compiler architecture

Schedule

- **Introduction to compilers**

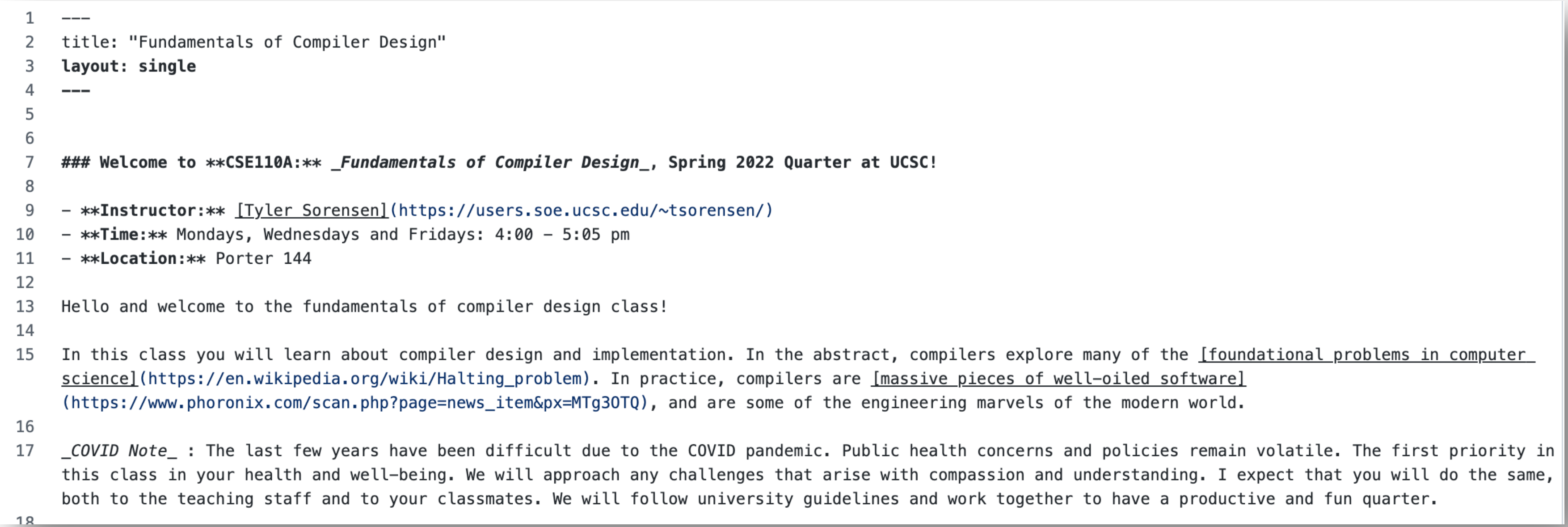
- Compiler architecture

What is a compiler?

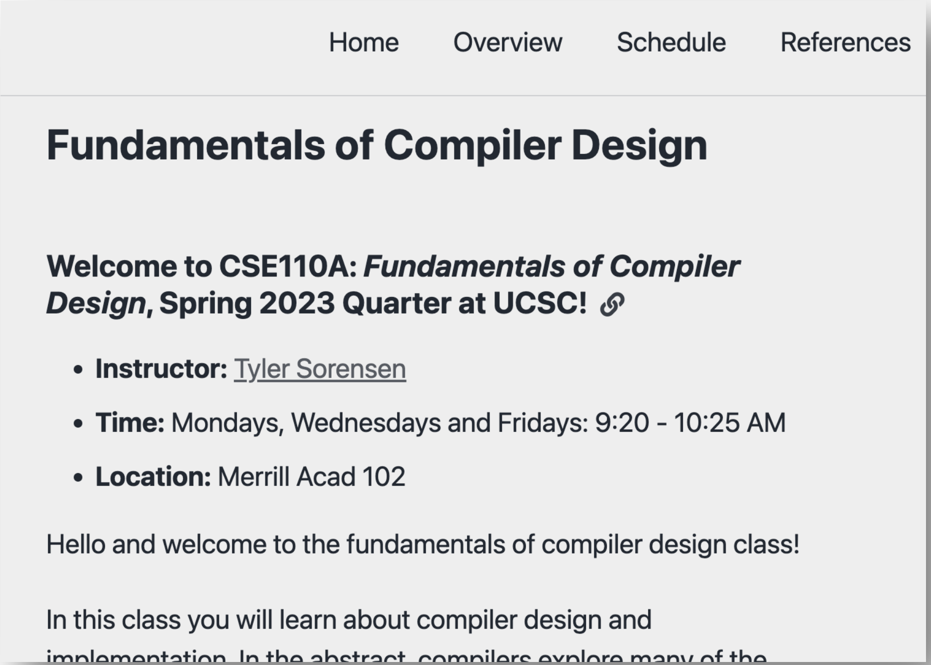
Let’s discuss

What are some of your favorite compilers

Let’s discuss



```
---
title: "Fundamentals of Compiler Design"
layout: single
---


### Welcome to **CSE110A:** _Fundamentals of Compiler Design_, Spring 2022 Quarter at UCSC!

- **Instructor:** [Tyler Sorensen](https://users.soe.ucsc.edu/~tsorensen/)
- **Time:** Mondays, Wednesdays and Fridays: 4:00 - 5:05 pm
- **Location:** Porter 144

Hello and welcome to the fundamentals of compiler design class!

In this class you will learn about compiler design and implementation. In the abstract, compilers explore many of the [foundational problems in computer science](https://en.wikipedia.org/wiki/Halting_problem). In practice, compilers are [massive pieces of well-oiled software](https://www.phoronix.com/scan.php?page=news_item&px=MTg3OTQ), and are some of the engineering marvels of the modern world.

_COVID Note_ : The last few years have been difficult due to the COVID pandemic. Public health concerns and policies remain volatile. The first priority in this class in your health and well-being. We will approach any challenges that arise with compassion and understanding. I expect that you will do the same, both to the teaching staff and to your classmates. We will follow university guidelines and work together to have a productive and fun quarter.
```



Fundamentals of Compiler Design

Welcome to CSE110A: *Fundamentals of Compiler Design*, Spring 2023 Quarter at UCSC!

- **Instructor:** Tyler Sorensen
- **Time:** Mondays, Wednesdays and Fridays: 9:20 - 10:25 AM
- **Location:** Merrill Acad 102

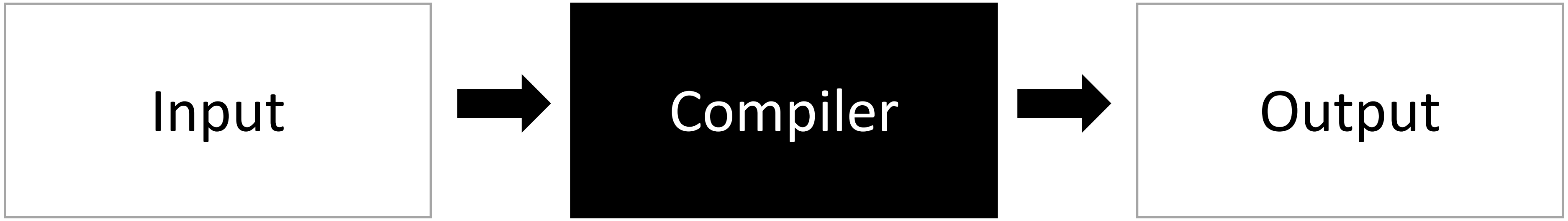
Hello and welcome to the fundamentals of compiler design class!

In this class you will learn about compiler design and implementation. In the abstract, compilers explore many of the

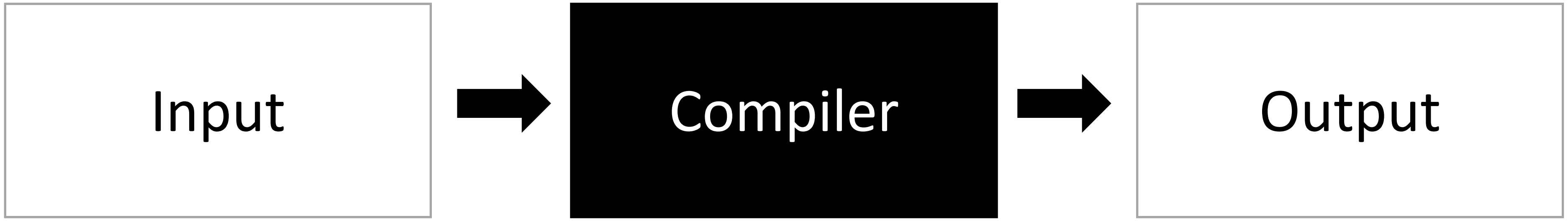
Building this website started with:

- Markdown to describe the page
- compiled with Jekyll to a static webpage
- static webpage is in HTML and javascript

What is a compiler?

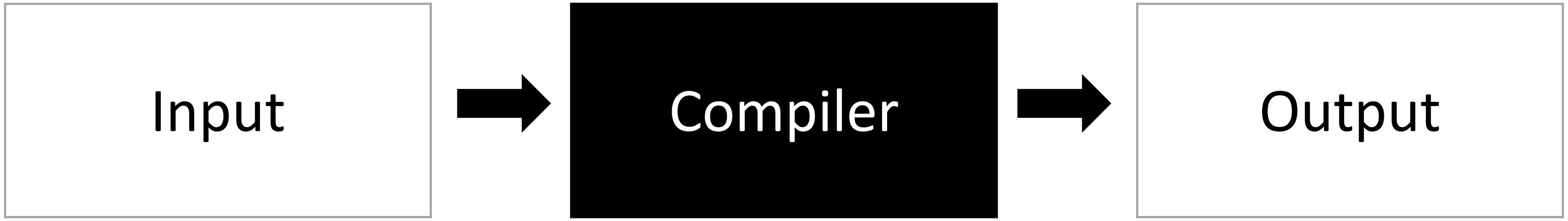


What is a compiler?



This is way too general to be useful
Any program fits this description.

What is a compiler?



Input

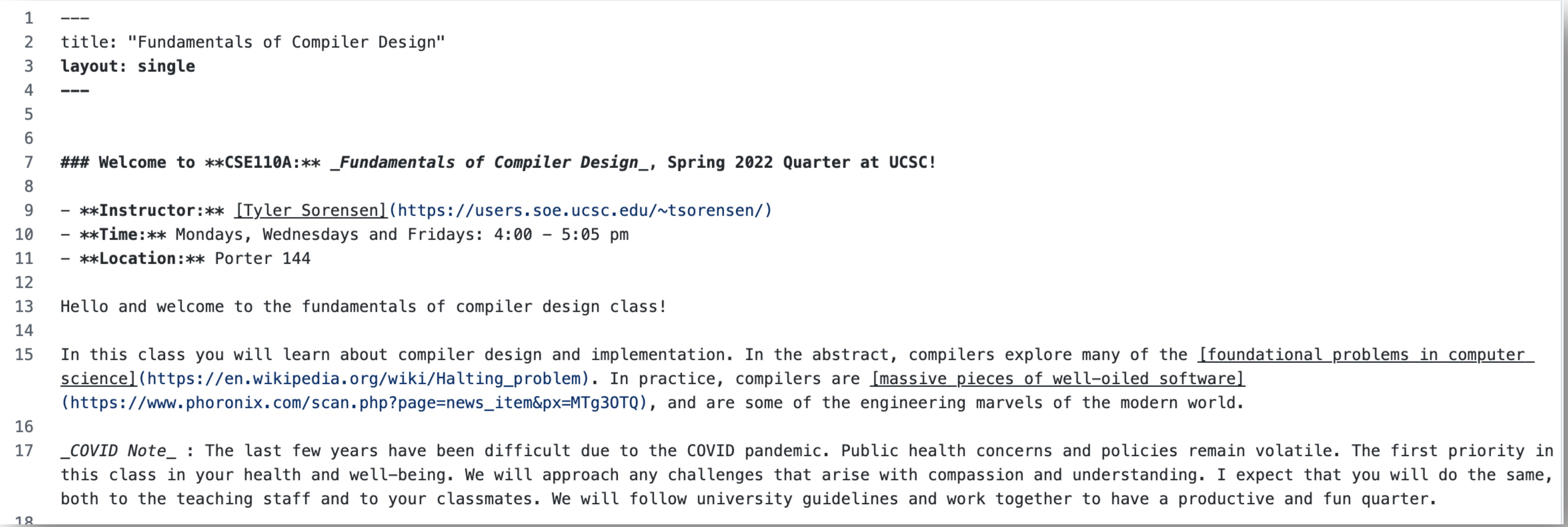
Compiler

Output

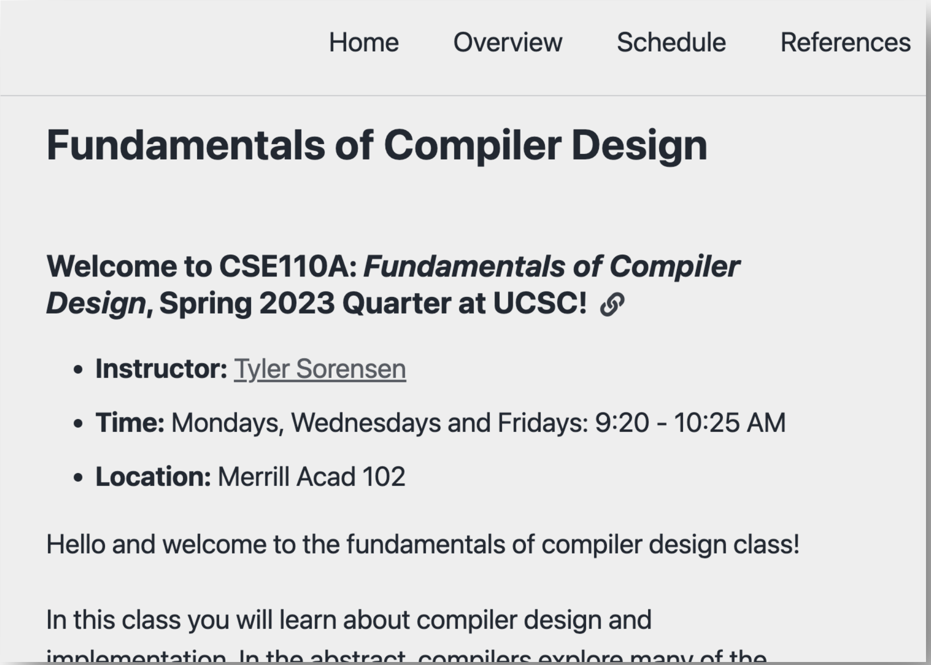
Strings belonging to language L

Strings belonging to language L’

A theoretical answer



```
---
title: "Fundamentals of Compiler Design"
layout: single
---


### Welcome to **CSE110A:** _Fundamentals of Compiler Design_, Spring 2022 Quarter at UCSC!

- **Instructor:** [Tyler Sorensen](https://users.soe.ucsc.edu/~tsorensen/)
- **Time:** Mondays, Wednesdays and Fridays: 4:00 - 5:05 pm
- **Location:** Porter 144

Hello and welcome to the fundamentals of compiler design class!

In this class you will learn about compiler design and implementation. In the abstract, compilers explore many of the [foundational problems in computer science](https://en.wikipedia.org/wiki/Halting_problem). In practice, compilers are [massive pieces of well-oiled software](https://www.phoronix.com/scan.php?page=news_item&px=MTg3OTQ), and are some of the engineering marvels of the modern world.

_COVID Note_ : The last few years have been difficult due to the COVID pandemic. Public health concerns and policies remain volatile. The first priority in this class in your health and well-being. We will approach any challenges that arise with compassion and understanding. I expect that you will do the same, both to the teaching staff and to your classmates. We will follow university guidelines and work together to have a productive and fun quarter.
```



Fundamentals of Compiler Design

Welcome to CSE110A: *Fundamentals of Compiler Design*, Spring 2023 Quarter at UCSC!

- **Instructor:** Tyler Sorensen
- **Time:** Mondays, Wednesdays and Fridays: 9:20 - 10:25 AM
- **Location:** Merrill Acad 102

Hello and welcome to the fundamentals of compiler design class!

In this class you will learn about compiler design and implementation. In the abstract, compilers explore many of the

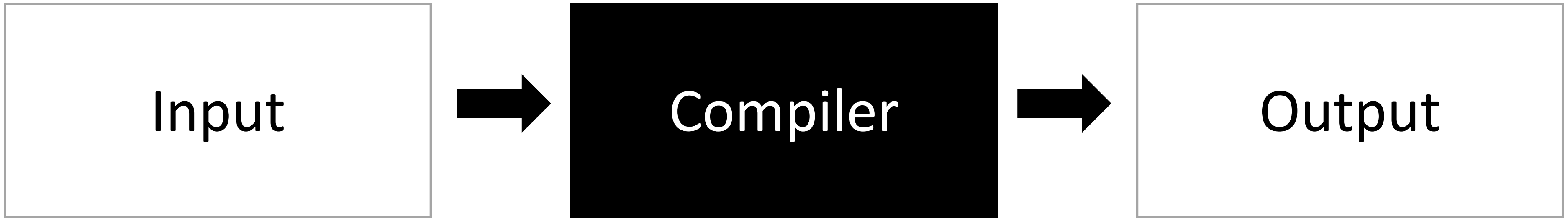
Building this website started with:

- Markdown to describe the page
- compiled with Jekyll to a static webpage
- static webpage is in HTML and javascript

This would be a compiler

What is a compiler?

A more traditional description
What are some examples here?



Strings belonging to language L

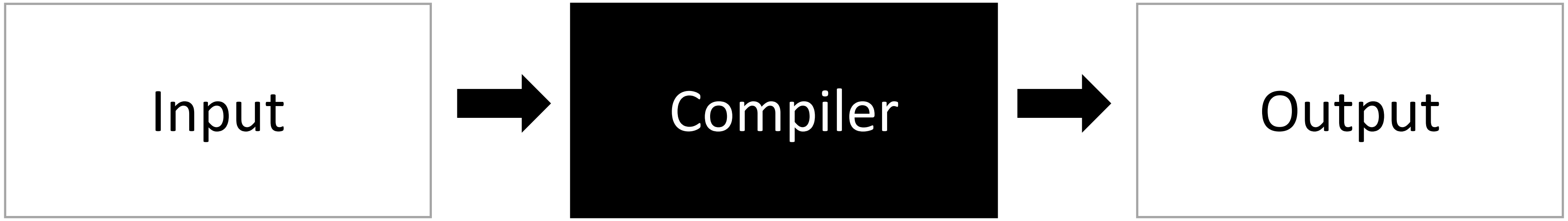
Strings belonging to language L’

A series of statements in programming language L

An executable binary file in an ISA language

What is a compiler?

A classic example



Input

Compiler

Output

Strings belonging to language L		Strings belonging to language L’
A series of statements in programming language L		An executable binary file in an ISA language
A program written in C	gcc/clang	An x86 Binary executable

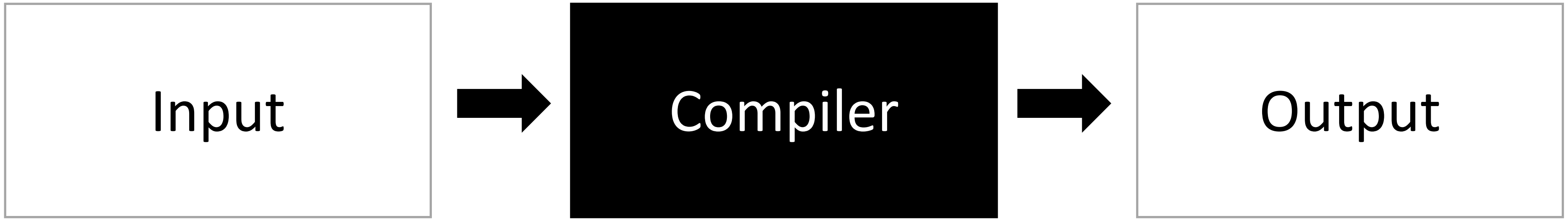
GCC and Clang

- Two mainstream compiler frameworks
- Similarities and differences?

What is a compiler?

```
int main() {
  printf("hello world\n");
}
```

```
gcc main.c
```



Input

Compiler

Output

Strings belonging to language L		Strings belonging to language L’
A series of statements in programming language L		An executable binary file in an ISA language
A program written in C	gcc/clang	An x86 Binary executable

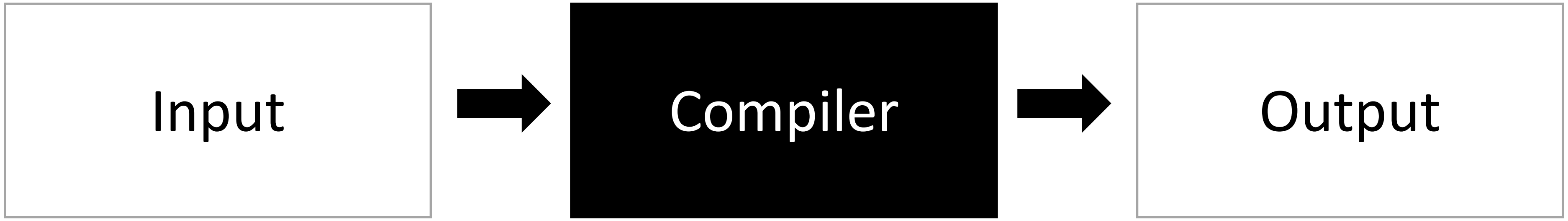
What is a compiler?

What is wrong with this picture?

```c
int main() {
  printf("hello world\n");
}
```

```
$ ./a.out
hello CSE 110A
```

```
gcc main.c
```



Input

Compiler

Output

Strings belonging to language L

Strings belonging to language L’

A series of statements in programming language L

An executable binary file in an ISA language

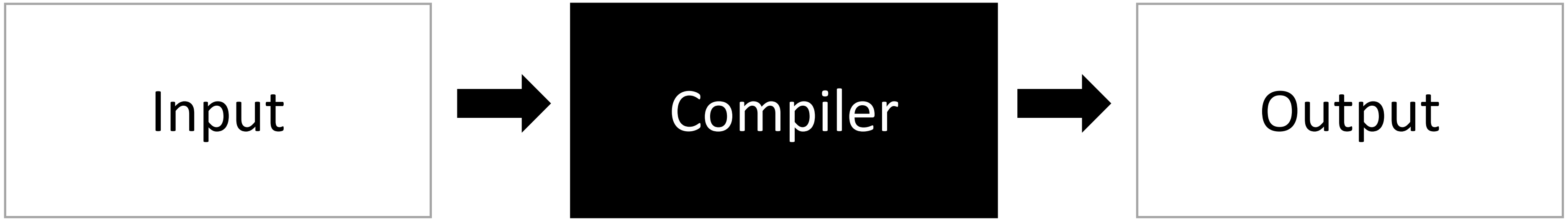
A program written in C

gcc/clang

An x86 Binary executable

What is a compiler?

A valid input must have a equivalent valid output.
Semantic equivalence



Input

Compiler

Output

Strings belonging to language L		Strings belonging to language L’
A series of statements in programming language L		An executable binary file in an ISA language
A program written in C	gcc/clang	An x86 Binary executable

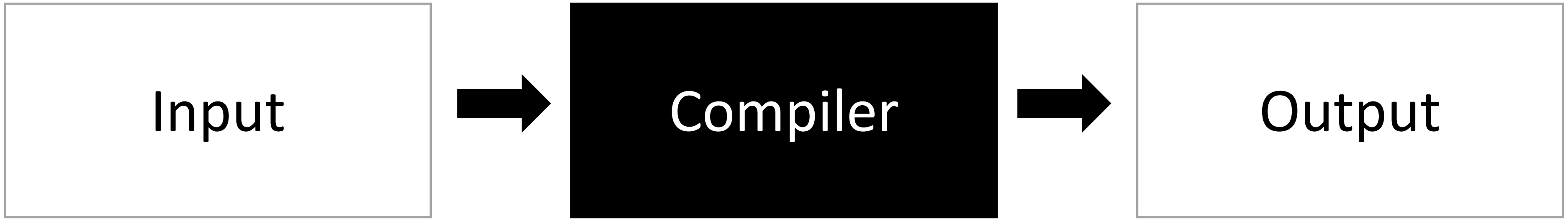
What is a compiler?

What is wrong with this picture?

```
int main() {
  printf("hello world\n");
}
```

```
$ ./a.out
hello CSE 110A
```

```
gcc main.c
```



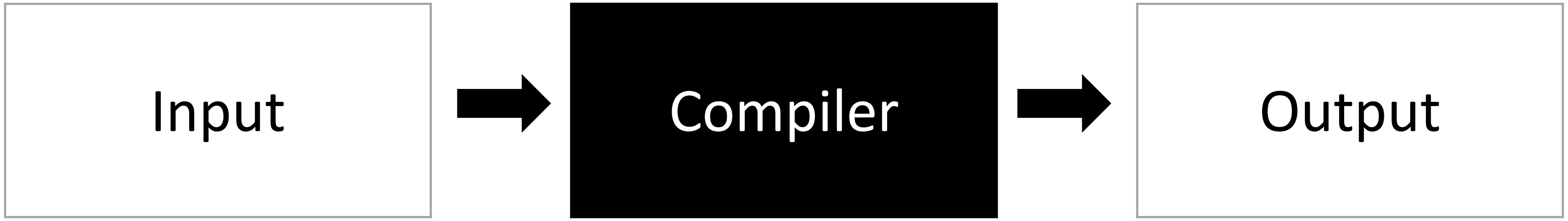
Strings belonging to language L		Strings belonging to language L’
A series of statements in programming language L		An executable binary file in an ISA language
A program written in C	gcc/clang	An x86 Binary executable

What is a compiler?

```
int main() {
  printf("hello world\n");
}
```

```
$ ./a.out
hello world
```

```
gcc main.c
```



Strings belonging to language L

Strings belonging to language L’

A series of statements in programming language L

An executable binary file in an ISA language

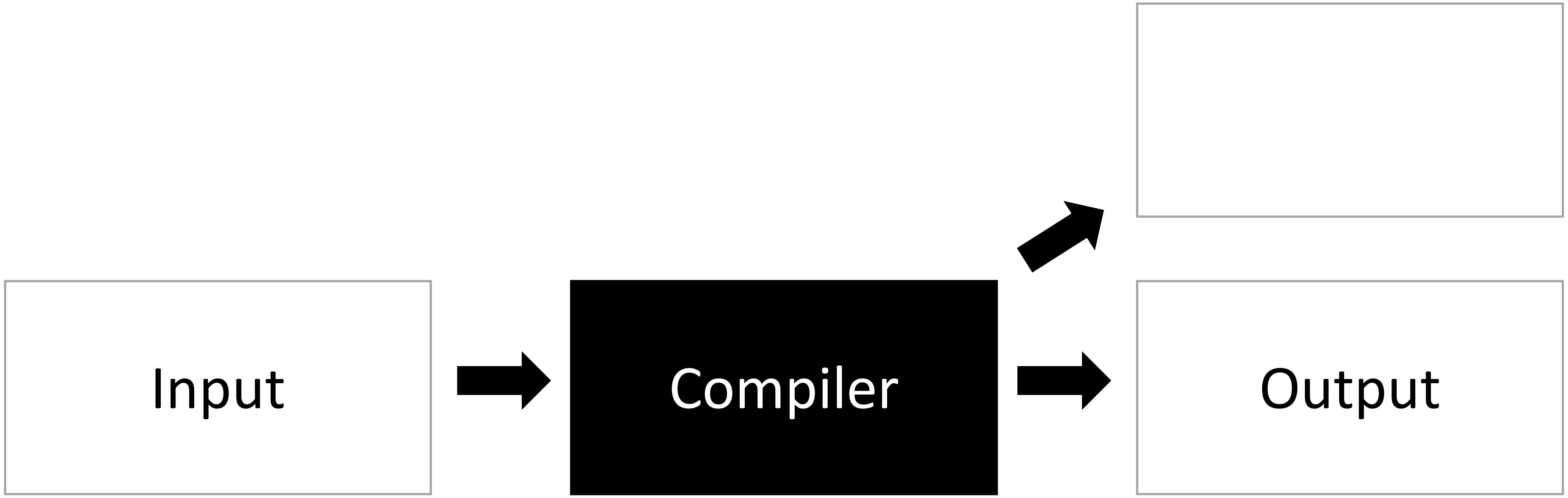
A program written in C

gcc/clang

An x86 Binary executable

What is a compiler?

What else does a compiler give you?



Input

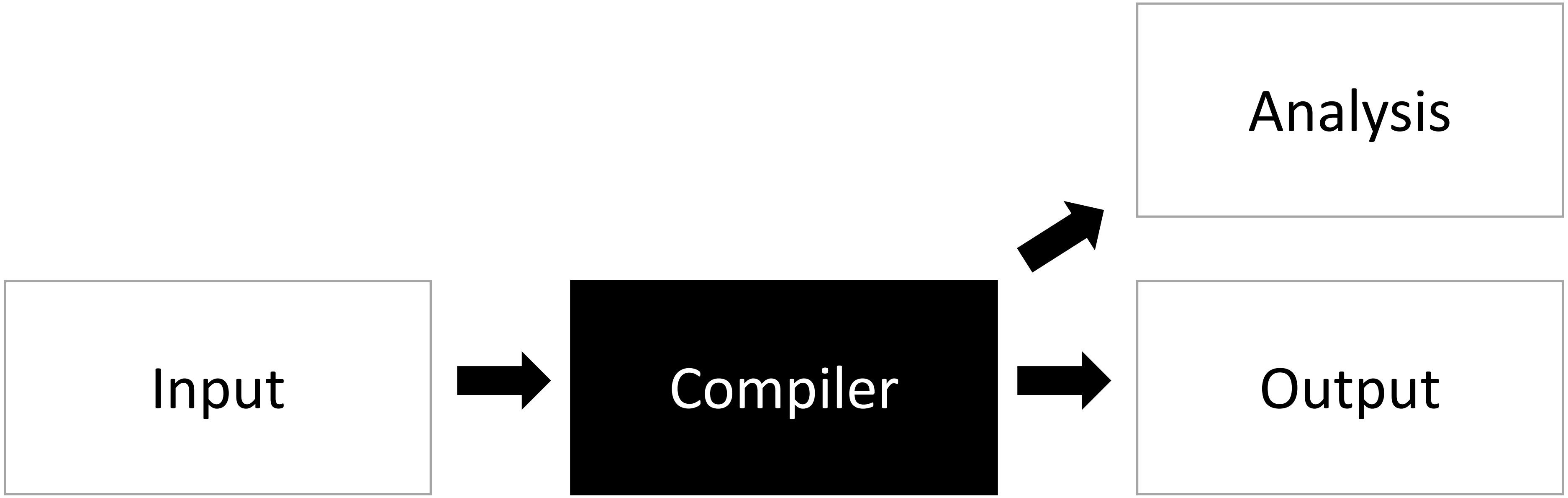
Compiler

Output

Strings belonging to language L		Strings belonging to language L’
A series of statements in programming language L		An executable binary file in an ISA language
A program written in C	gcc/clang	An x86 Binary executable

What is a compiler?

What are some examples here?



Analysis

Input

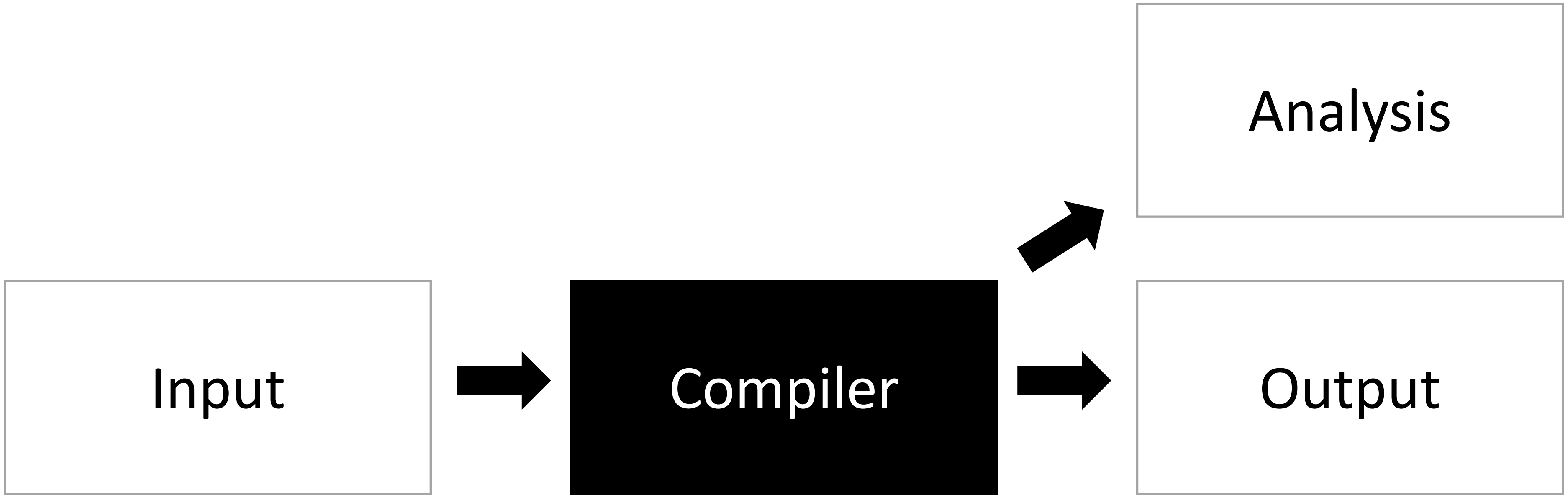
Compiler

Output

Strings belonging to language L		Strings belonging to language L’
A series of statements in programming language L		An executable binary file in an ISA language
A program written in C	gcc/clang	An x86 Binary executable

What is a compiler?

What are some examples here?



Input

Compiler

Output

Analysis

Warnings
Errors
Performance logs

Strings belonging to language L

Strings belonging to language L’

A series of statements in programming language L

An executable binary file in an ISA language

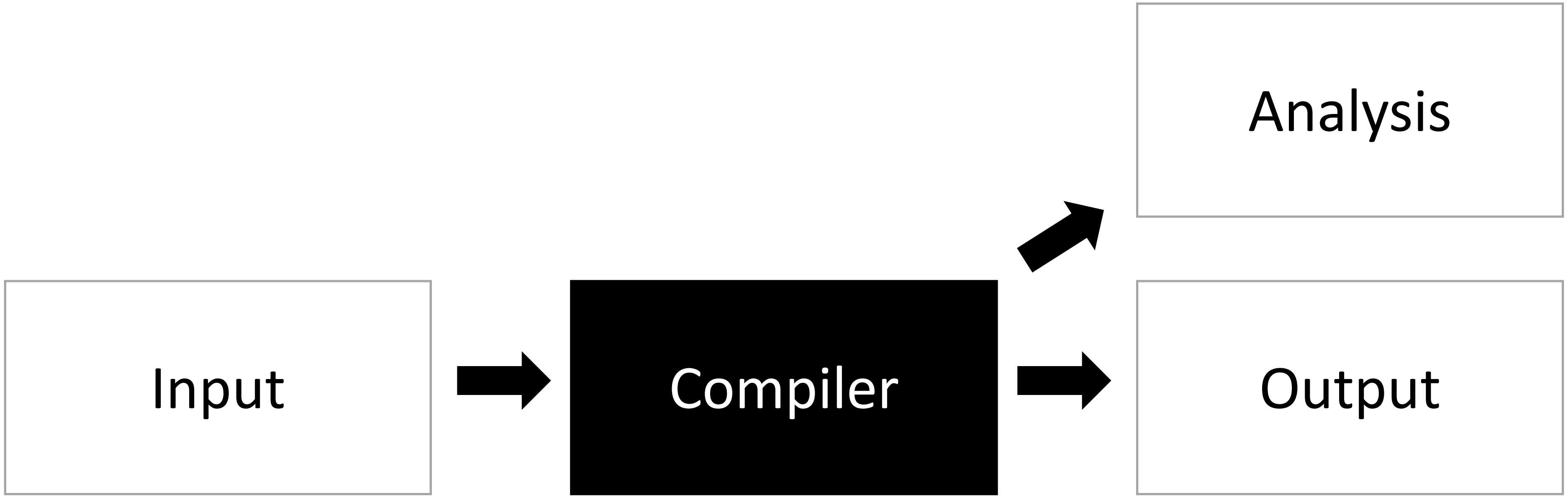
A program written in C

gcc/clang

An x86 Binary executable

What is a compiler?

What are some examples here?



Analysis

Warnings
Errors
Performance logs

Input

Compiler

Output

Strings belonging to language L

Strings belonging to language L’

A series of statements in programming language L

An executable binary file in an ISA language

A program written in C

gcc/clang

An x86 Binary executable

Demo

- What are some examples of code that might give a warning?

What can happen when the Input isn’t valid?

```
int foo() {
  int x;
  int y = x;
  return y;
}
```

Try running this through the compiler

What can happen when the Input isn’t valid?

```
int foo() {
  int x;
  int y = x;
  return y;
}
```

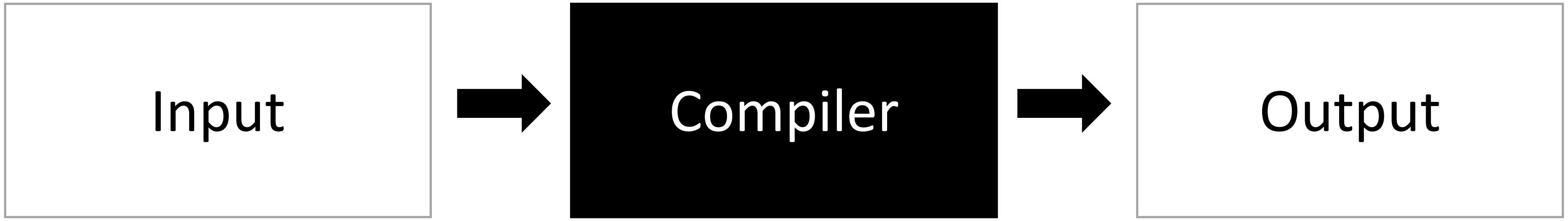
Try running this through the compiler

```
int foo(int condition) {
  int x;
  if (condition) {
    x = 5;
  }
  int y = x;
  return y;
}
```

What about this one?

What is a compiler?

A valid input must have a equivalent valid output.
Semantic equivalence



Input

Compiler

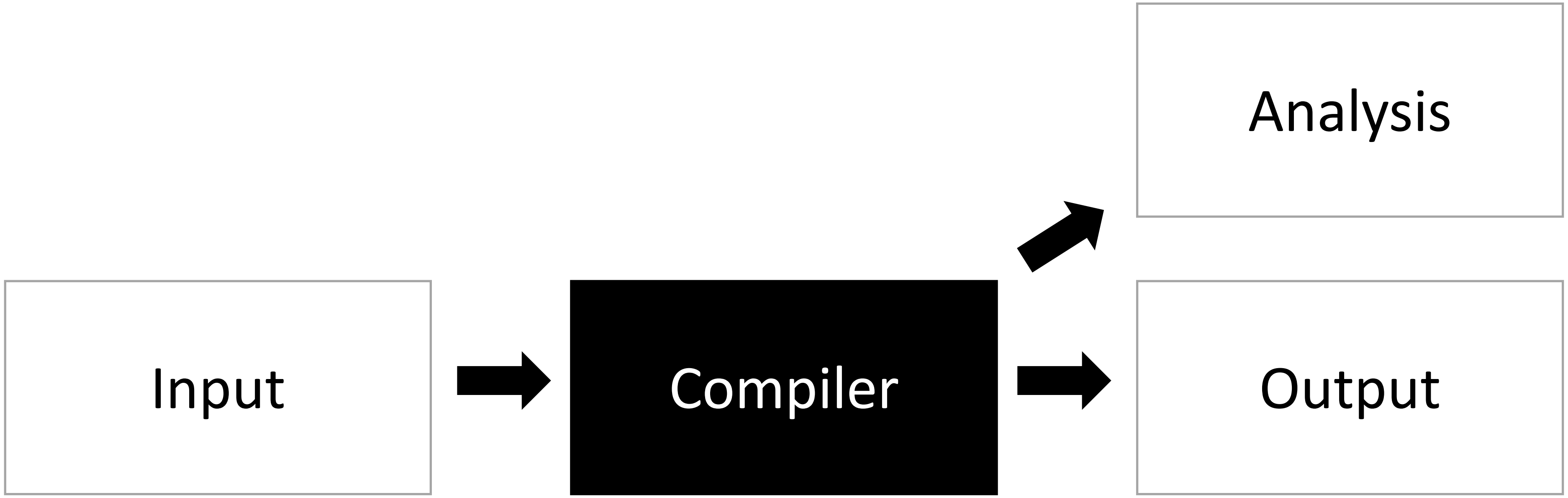
Output

Strings belonging to language L		Strings belonging to language L’
A series of statements in programming language L		An executable binary file in an ISA language
A program written in C	gcc/clang	An x86 Binary executable

Uninitialized variable example

What is a compiler?

What are some examples here?



Analysis

Warnings
Errors
Performance logs

Input

Compiler

Output

Strings belonging to language L

Strings belonging to language L’

A series of statements in programming language L

An executable binary file in an ISA language

A program written in C

gcc/clang

An x86 Binary executable

What can happen when the Input isn’t valid?

```
int foo() {
  int my_var = 5;
  my_var = my_car + 5;
  return my_var;
}
```

Try running this through a compiler

What can happen when the Input isn’t valid?

```
int foo() {
  int my_var = 5;
  my_var = my_car + 5;
  return my_var;
}
```

Try running this through a compiler

You get an error and a suggestion these days

What can happen when the Input isn’t valid?

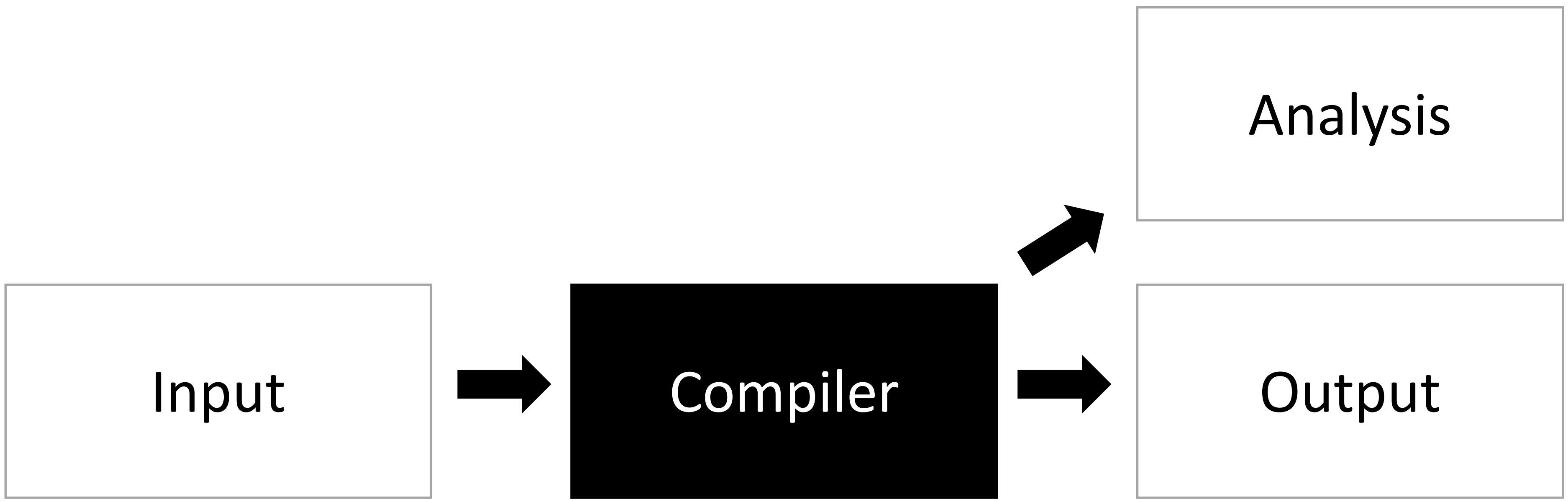
```
int foo() {
  int *x = malloc(100*sizeof(int));
  return x[100];
}
```

What about this one? No error...

What sort of errors are compilers good at catching?
What ones are they not?

What is a compiler?

What are some examples here?



Analysis

Input

Compiler

Output

Warnings
Errors
Performance logs

Strings belonging to language L

Strings belonging to language L’

A series of statements in programming language L

An executable binary file in an ISA language

A program written in C

gcc/clang

An x86 Binary executable

How can we know what the compiler is doing?

```
#define SIZE (1024*1024)
int add(int * a, int * b, int * c) {
  for (int i = 0; i < SIZE; i++) {
    a[i] = b[i] + c[i];
  }
  return 0;
}
```

Use the compiler flags

```
-Rpass-missed=loop-vectorize
-Rpass=loop-vectorize
```

Does the compiler need to perform every step?

```c
int foo() {
  int my_var = 0;
  for (int i = 0; i < 128; i++) {
    my_var++;
  }
  return my_var;
}
```

Does the compiler need to perform every step?

```
int foo() {
  int my_var = 0;
  for (int i = 0; i < 128; i++) {
    my_var++;
  }
  return my_var;
}
```

Mentally we probably step through the for loop:

Does the compiler need to perform every step?

```c
int foo() {
  int my_var = 0;
  for (int i = 0; i < 128; i++) {
    my_var++;
  }
  return my_var;
}
```

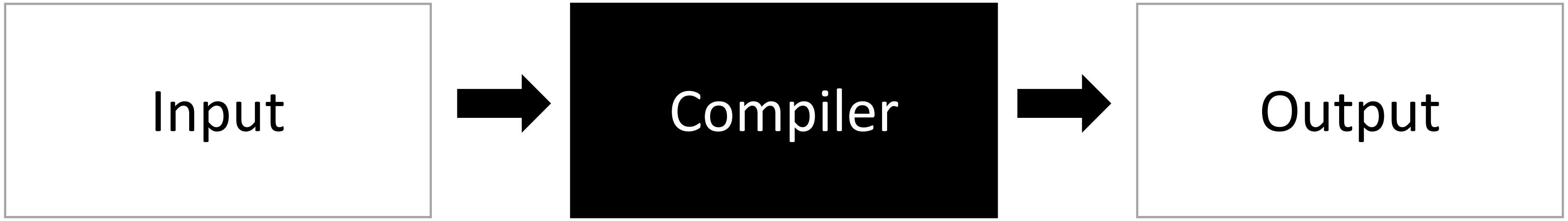
Mentally we probably step through the for loop:

What does the compiler do?

What is a compiler?

A valid input must have a equivalent valid output.

Semantic equivalence



Strings belonging to language L		Strings belonging to language L’
A series of statements in programming language L		An executable binary file in an ISA language
A program written in C	gcc/clang	An x86 Binary executable

Does the compiler need to perform every step?

```
int foo() {
  int my_var = 0;
  for (int i = 0; i < 128; i++) {
    my_var++;
  }
  return my_var;
}
```

```
int foo() {
  return 128;
}
```

are these the same?

Does the compiler need to perform every step?

```
int foo() {
  int my_var = 0;
  for (int i = 0; i < 128; i++) {
    my_var++;
  }
  return my_var;
}
```

```
int foo() {
  return 128;
}
```

are these the same?

Functionally - they are the same

Non-functionally - they are not

Most compilers are concerned only with functional equivalence

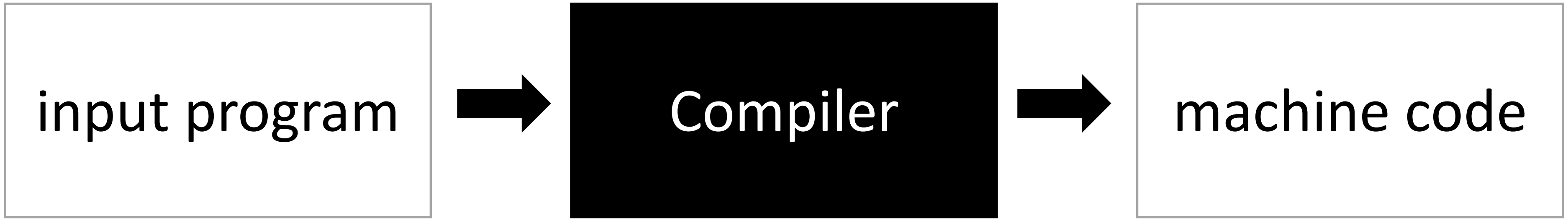
Schedule

- Introduction to compilers

- **Compiler architecture**

Compiler Architecture

Compiler Architecture



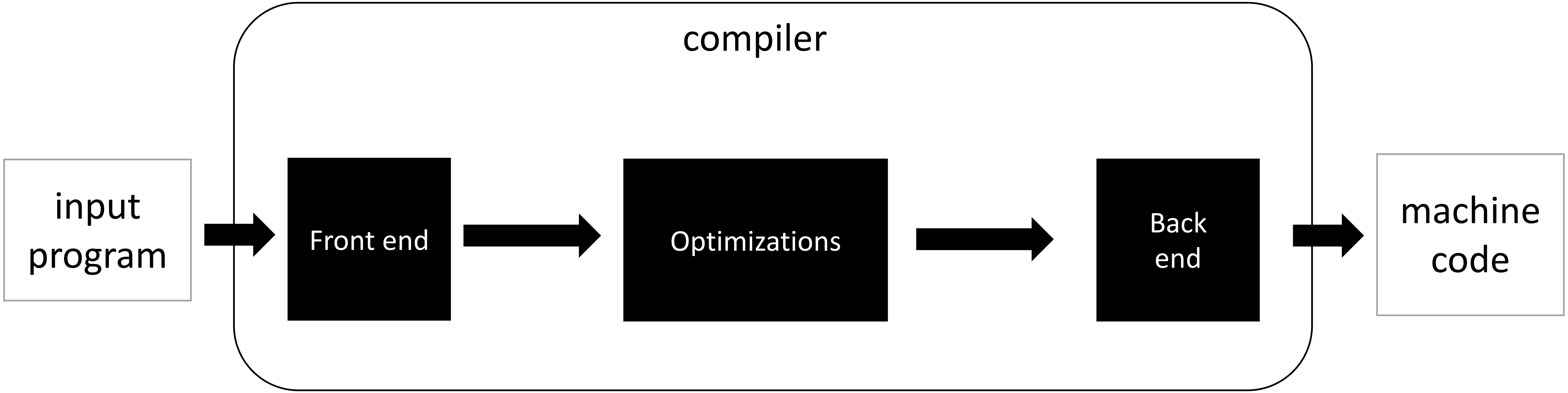
input program

Compiler

machine code

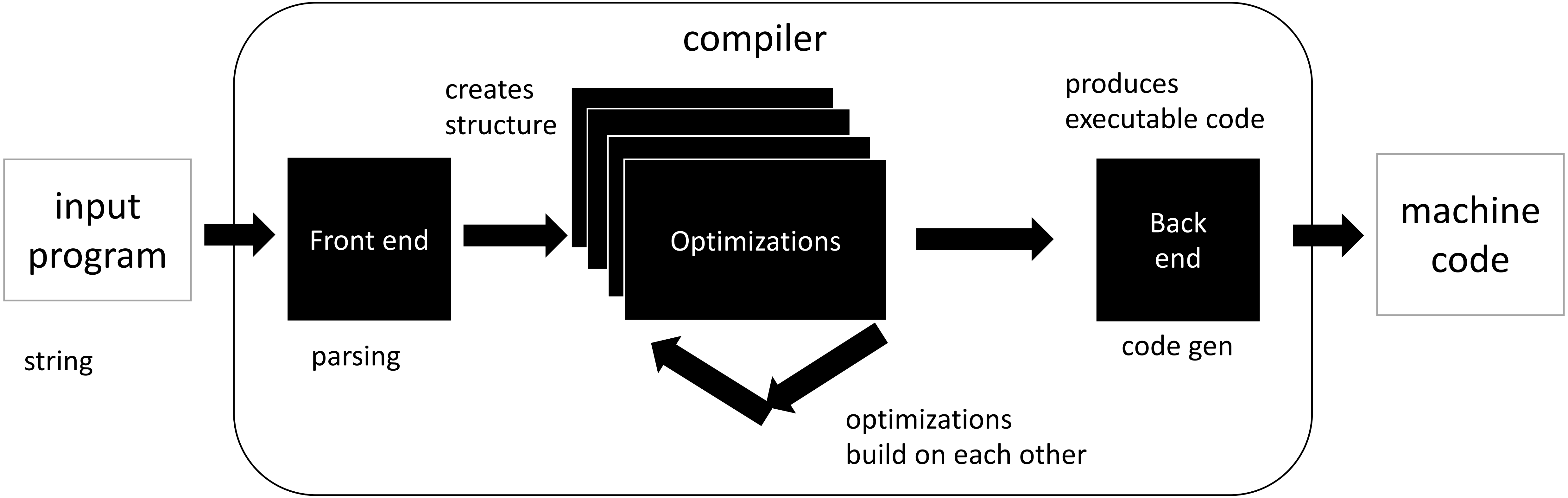
Compilers are complicated and this image is too simple

Compiler Architecture



Medium detailed view

Compiler Architecture



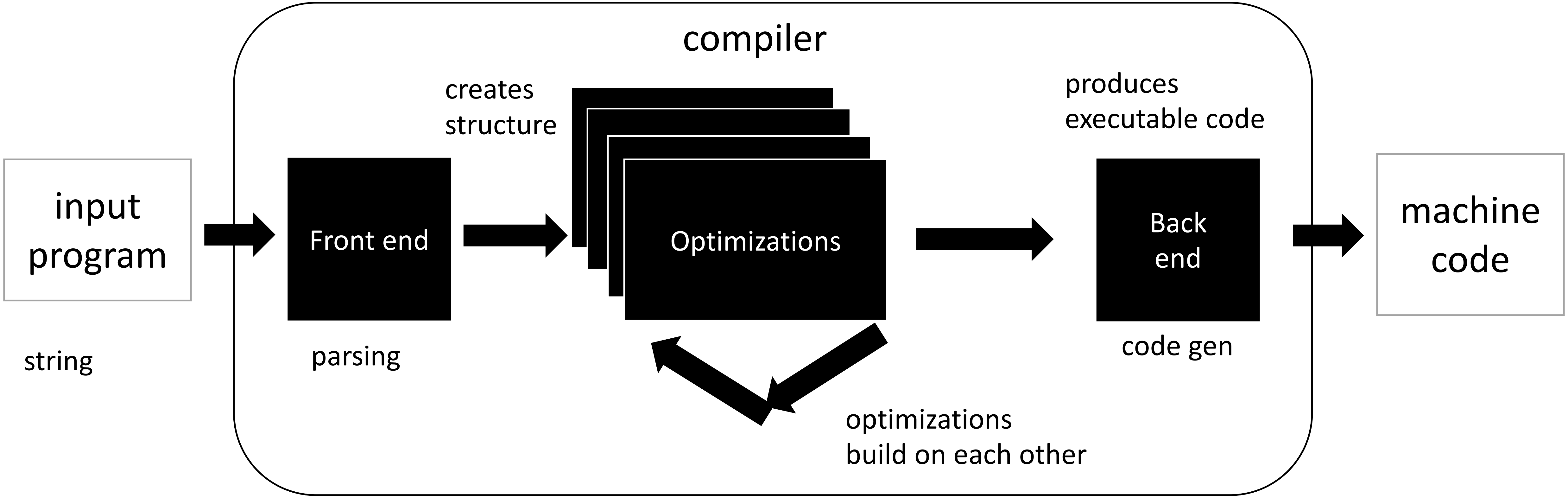
Medium detailed view

more about optimizations: https://stackoverflow.com/questions/15548023/clang-optimization-levels

Compiler Architecture

What are some of the benefits of this design?

What are some of the drawbacks of this design?



Medium detailed view

more about optimizations: https://stackoverflow.com/questions/15548023/clang-optimization-levels

LLVM compiler infrastructure example

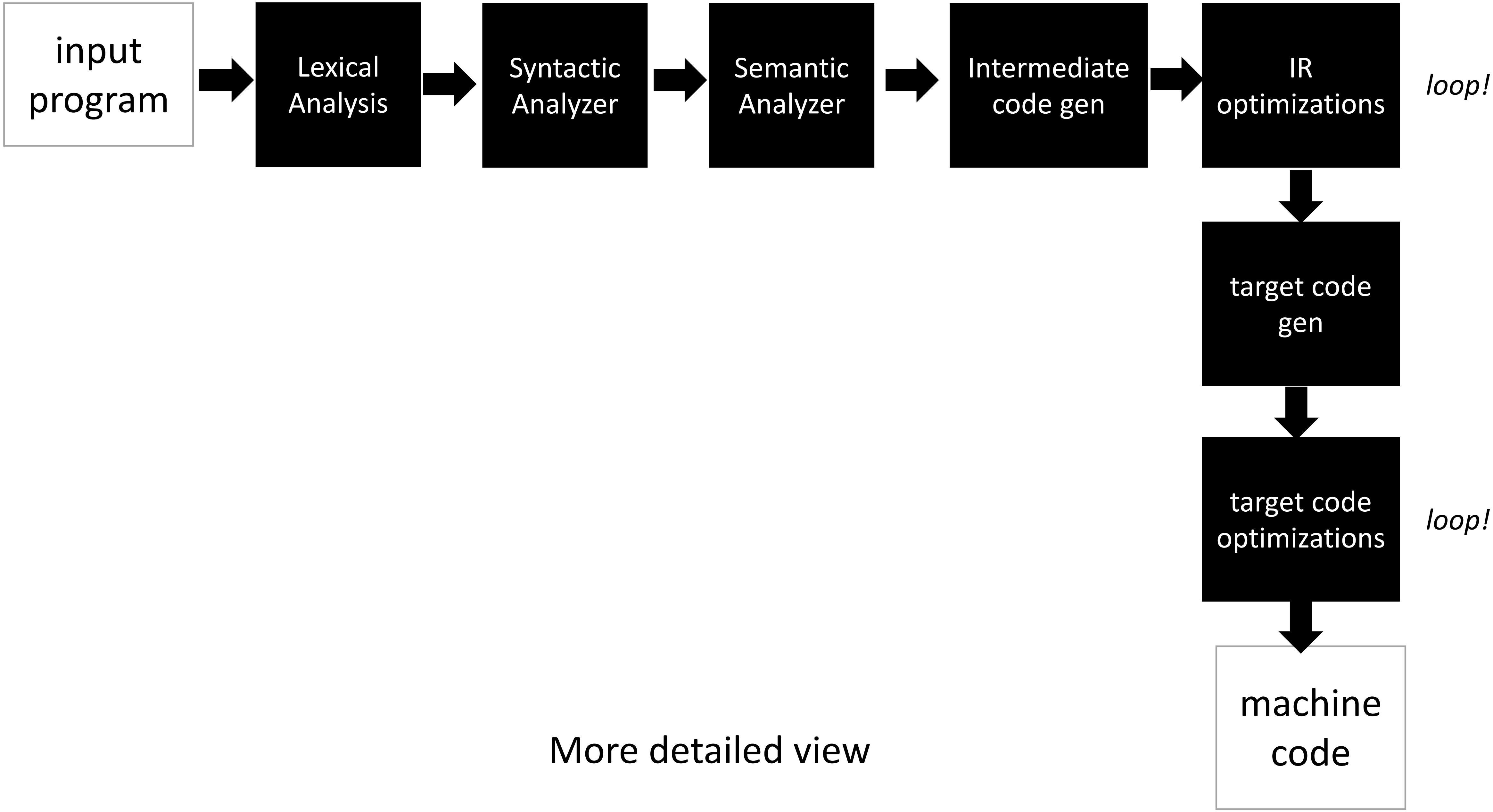
- Front ends:
  - clang - c
  - clang++ - c++
  - Many others (rust, etc.)

- intermediate representation:
  - LLVM byte code

- backends
  - X86
  - ARM
  - M1
  - RISC-V

More detailed compiler view

- Can’t fit it nicely on one slide!



input program

Lexical Analysis

Syntactic Analyzer

Semantic Analyzer

Intermediate code gen

IR optimizations

loop!

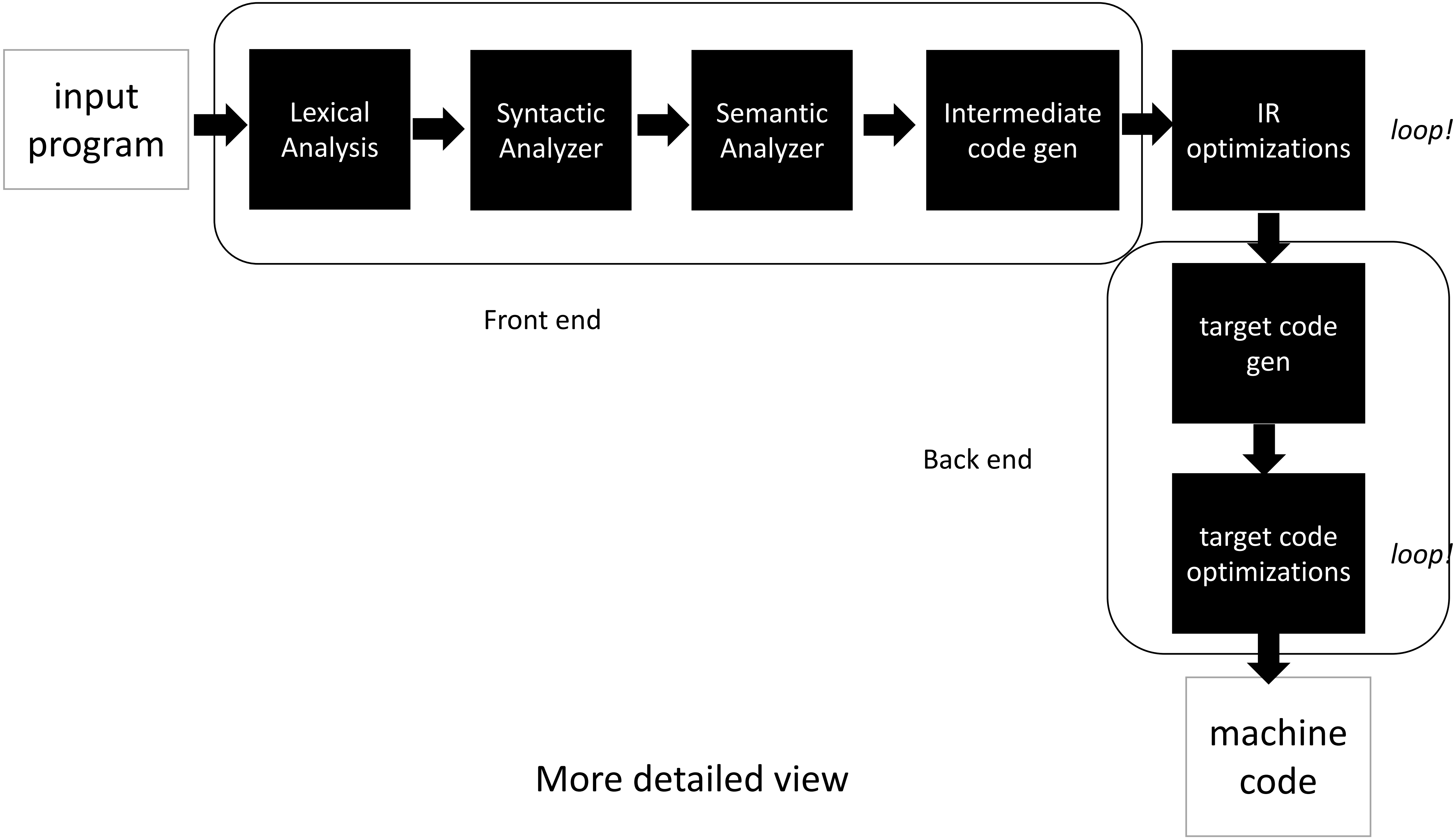
target code gen

target code optimizations

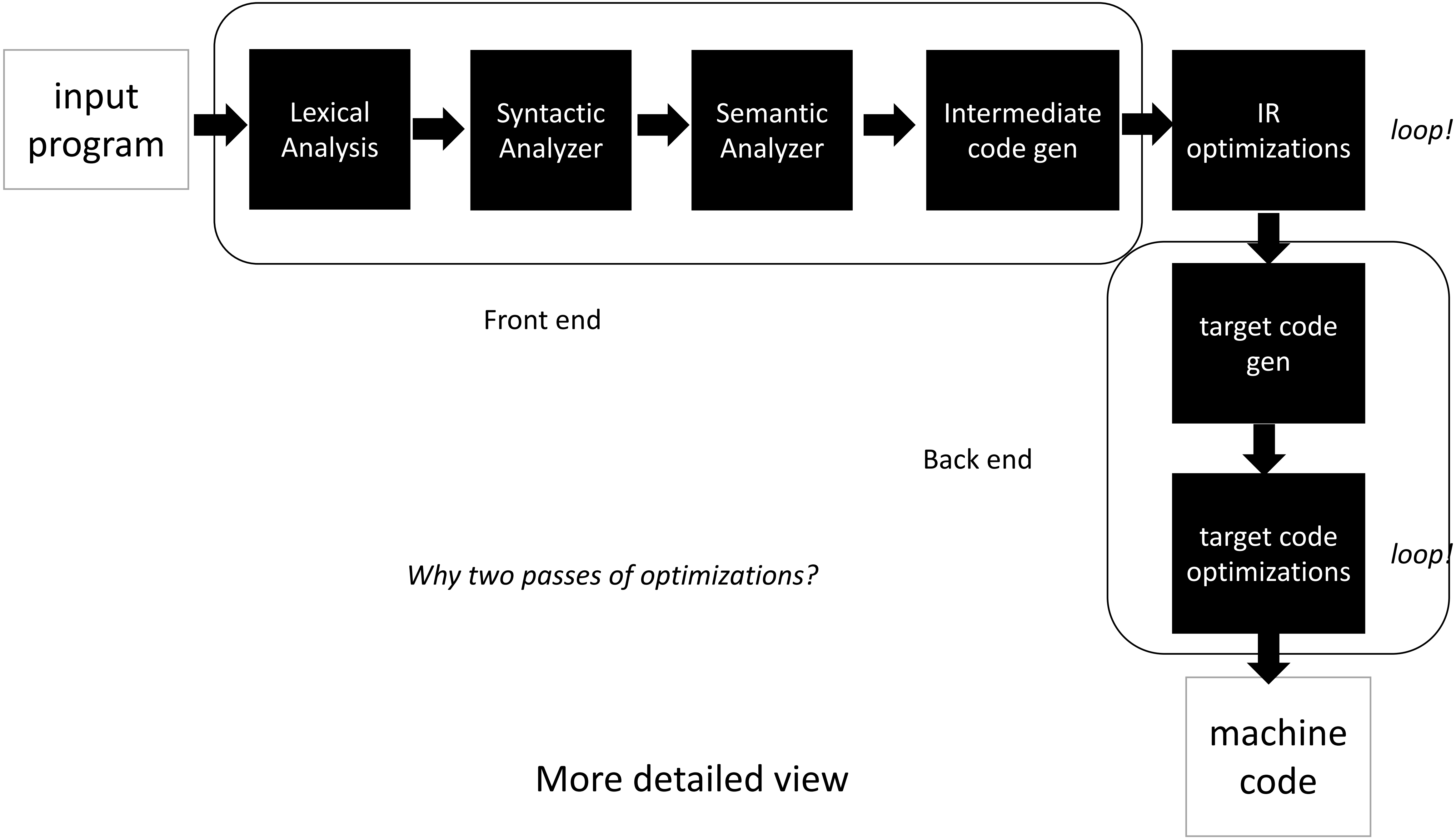
loop!

machine code

More detailed view



More detailed view



input program

Lexical Analysis

Syntactic Analyzer

Semantic Analyzer

Intermediate code gen

IR optimizations

loop!

Front end

target code gen

Back end

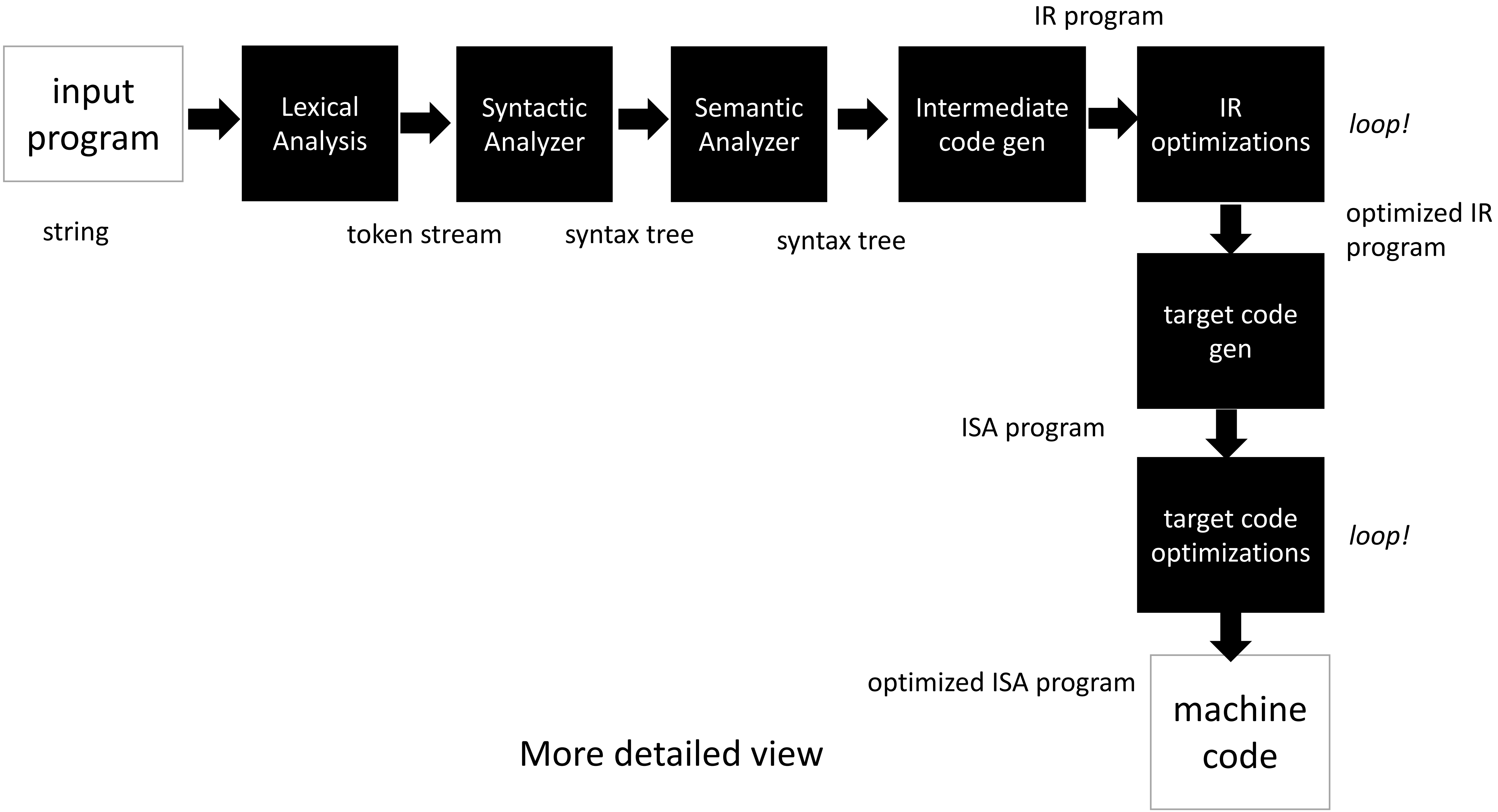
target code optimizations

loop!

machine code

Why two passes of optimizations?

More detailed view



input program

string

Lexical Analysis

token stream

Syntactic Analyzer

syntax tree

Semantic Analyzer

syntax tree

Intermediate code gen

IR program

IR optimizations

loop!

optimized IR program

target code gen

ISA program

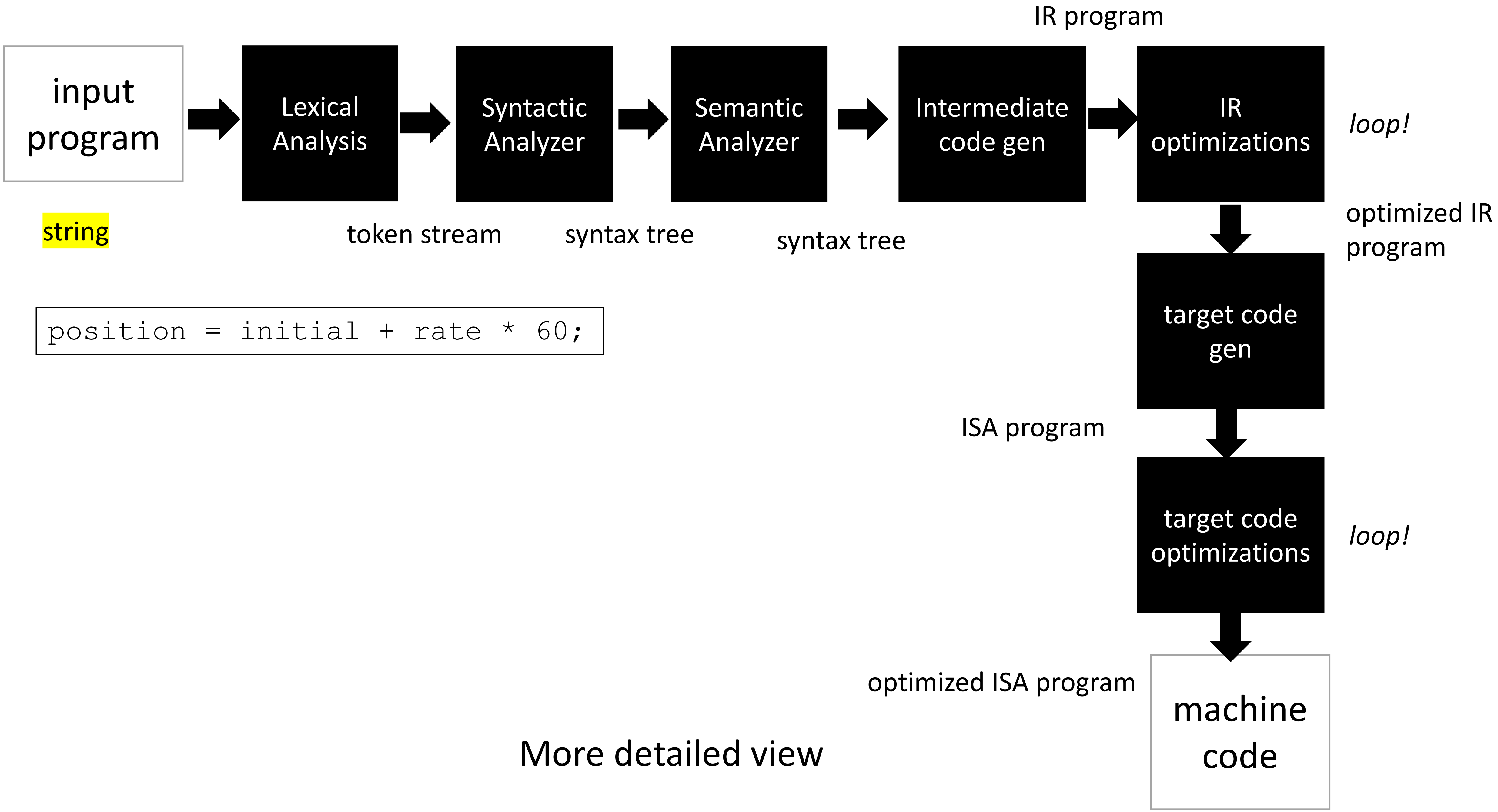
target code optimizations

loop!

optimized ISA program

machine code

More detailed view



input program

string

Lexical Analysis

token stream

Syntactic Analyzer

syntax tree

Semantic Analyzer

syntax tree

Intermediate code gen

IR program

IR optimizations

loop!

optimized IR program

target code gen

ISA program

target code optimizations

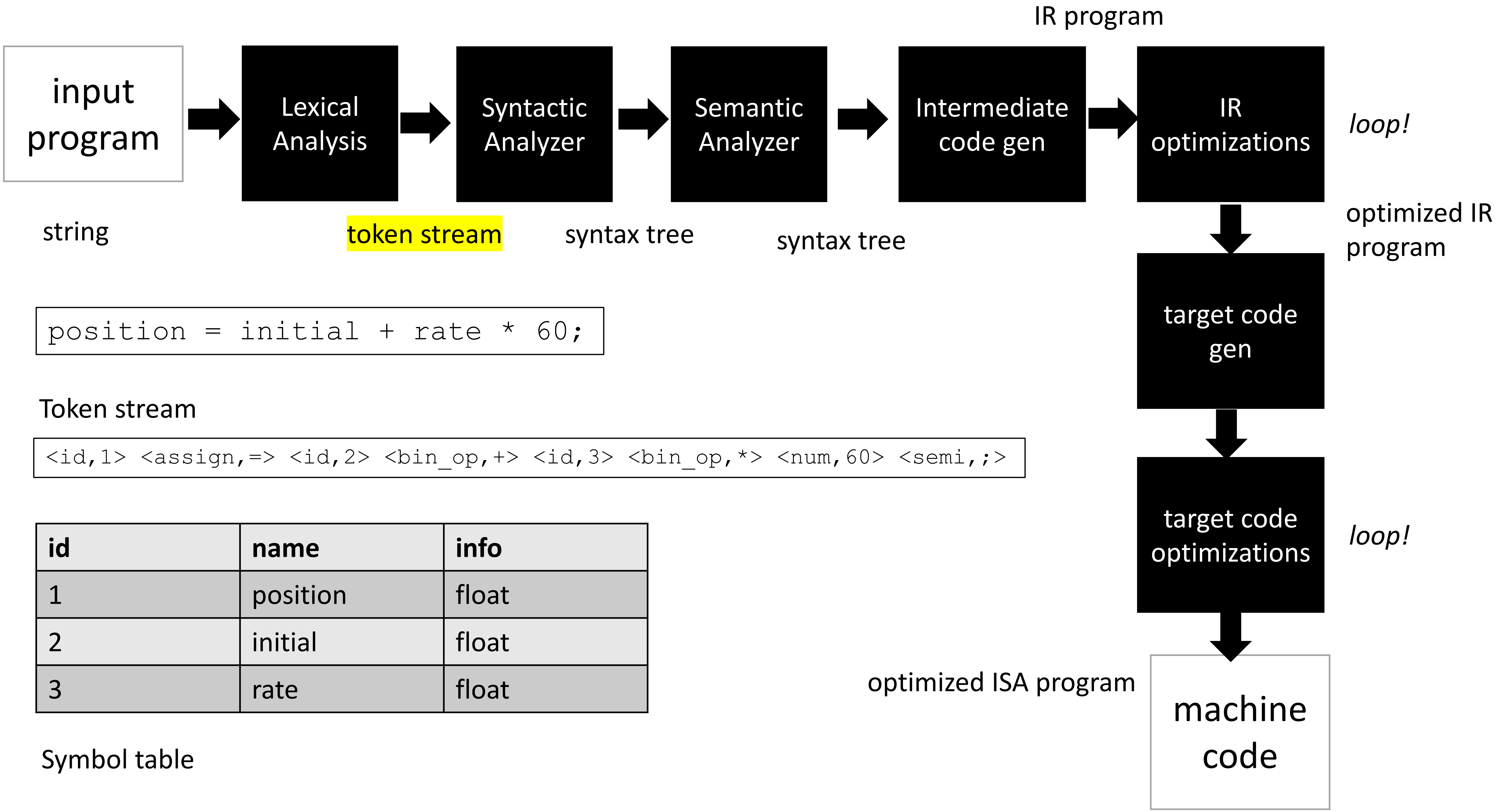
loop!

optimized ISA program

machine code

```
position = initial + rate * 60;
```

More detailed view



```
position = initial + rate * 60;
```

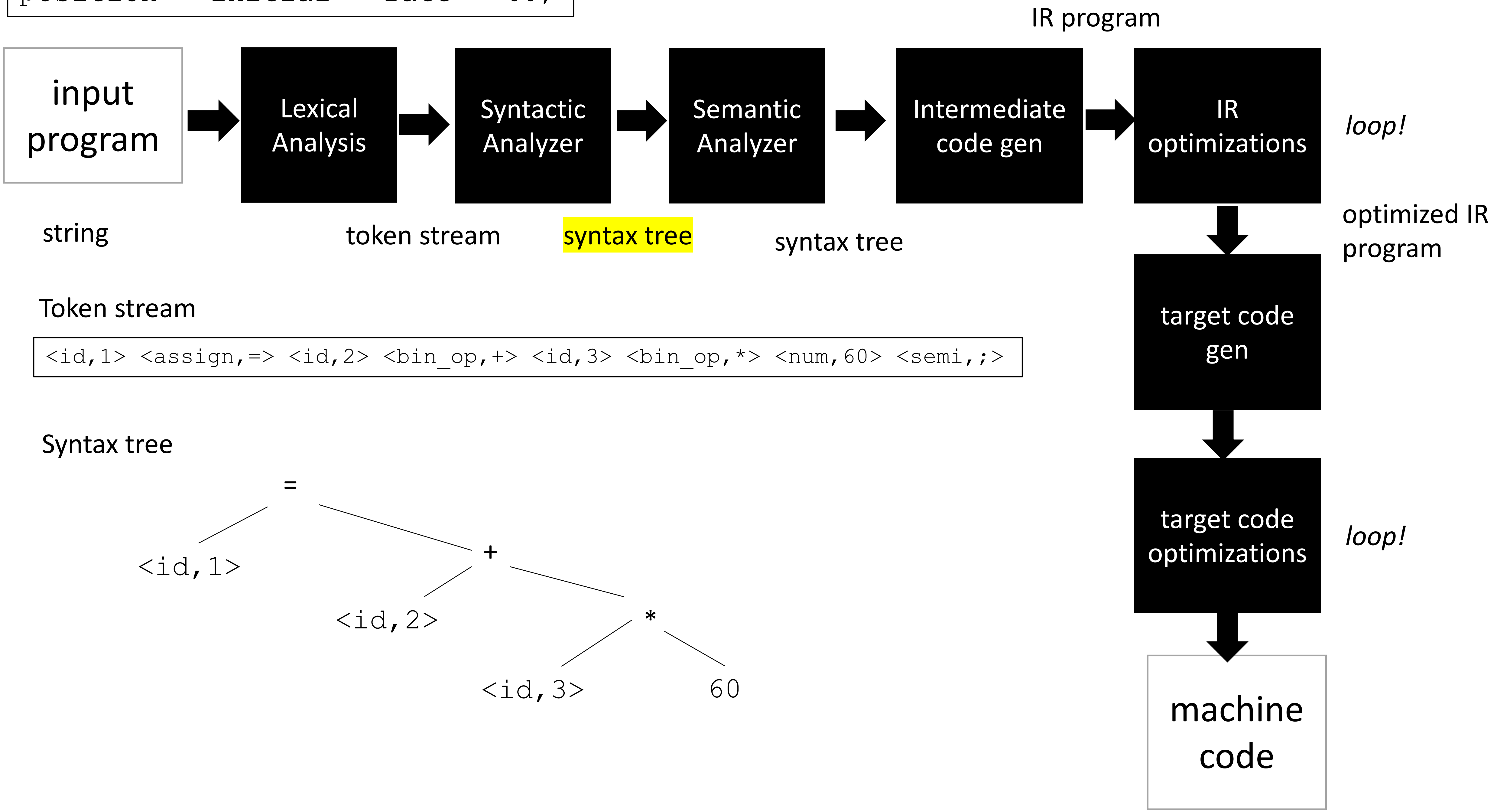
Token stream

```
<id,1> <assign,=> <id,2> <bin_op,+> <id,3> <bin_op,*> <num,60> <semi,;>
```

id	**name**	**info**
1	position	float
2	initial	float
3	rate	float

Symbol table

```
position = initial + rate * 60;
```

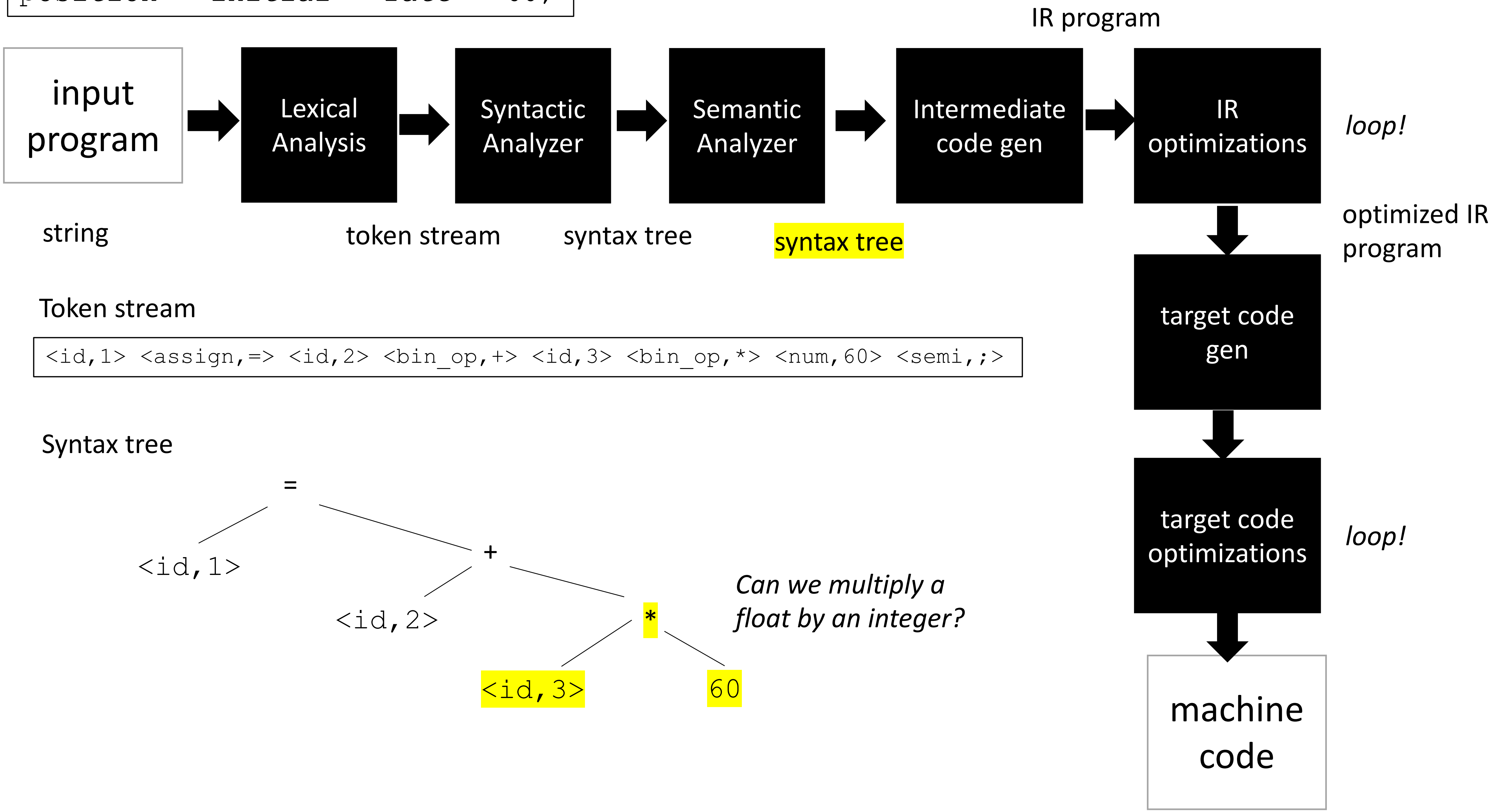


Token stream

```
<id,1> <assign,=> <id,2> <bin_op,+> <id,3> <bin_op,*> <num,60> <semi,;>
```

Syntax tree

```
position = initial + rate * 60;
```



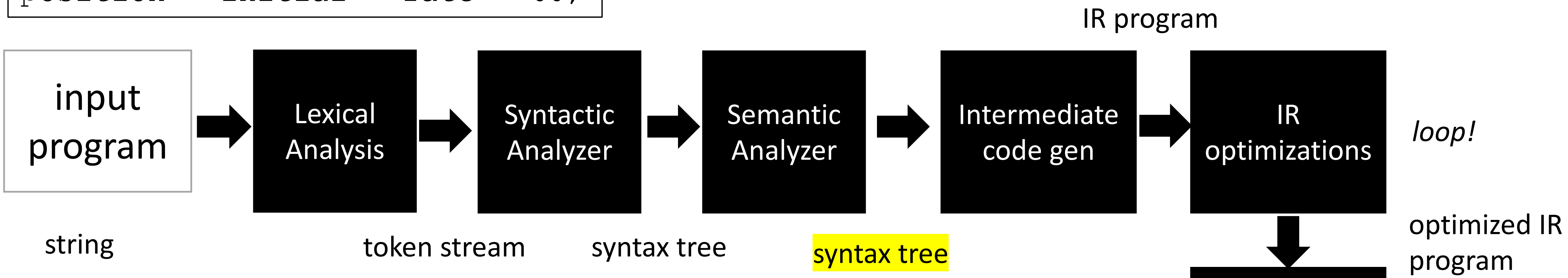
Token stream

```
<id,1> <assign,=> <id,2> <bin_op,+> <id,3> <bin_op,*> <num,60> <semi,;>
```

Syntax tree

Can we multiply a float by an integer?

```
position = initial + rate * 60;
```



IR program

input program

Lexical Analysis

Syntactic Analyzer

Semantic Analyzer

Intermediate code gen

IR optimizations

loop!

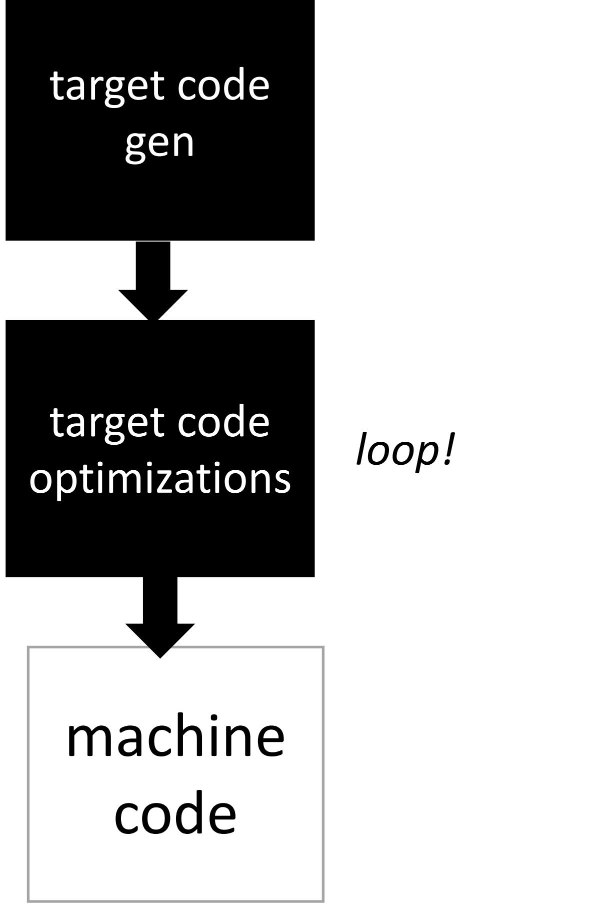
string

token stream

syntax tree

syntax tree

optimized IR program



target code gen

target code optimizations

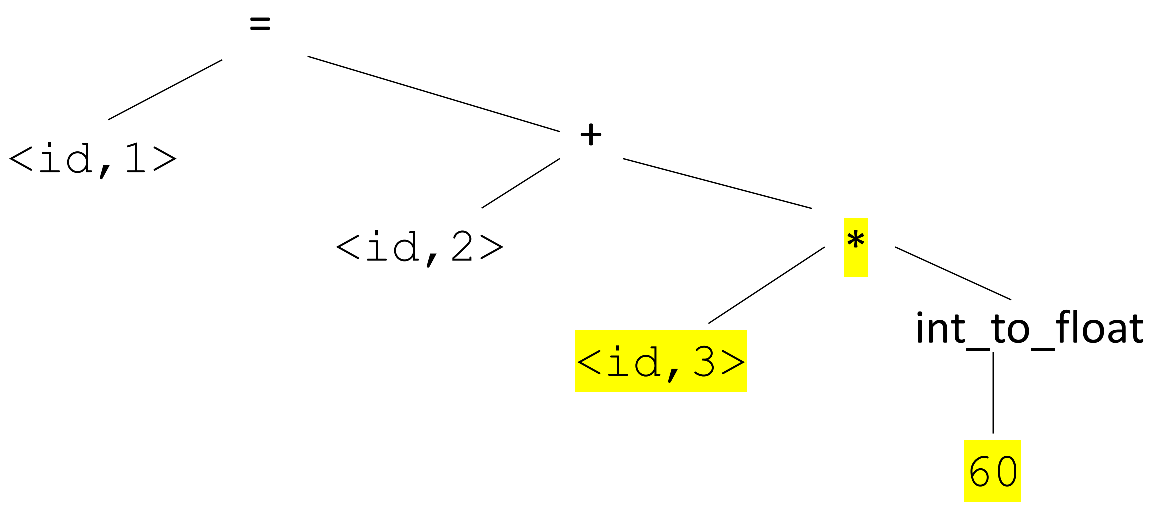
loop!

machine code

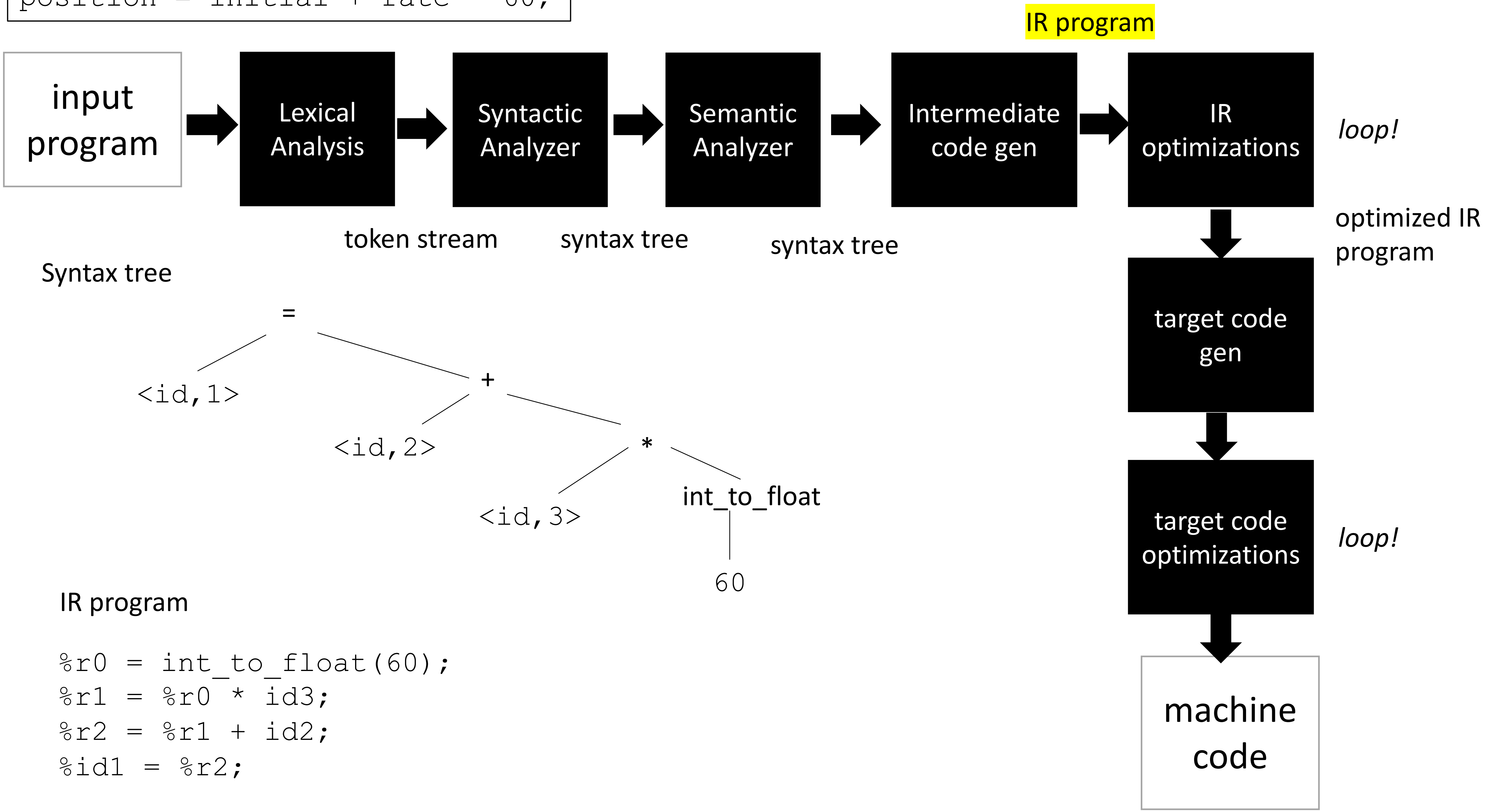
Token stream

```
<id,1> <assign,=> <id,2> <bin_op,+> <id,3> <bin_op,*> <num,60> <semi,;>
```

Syntax tree



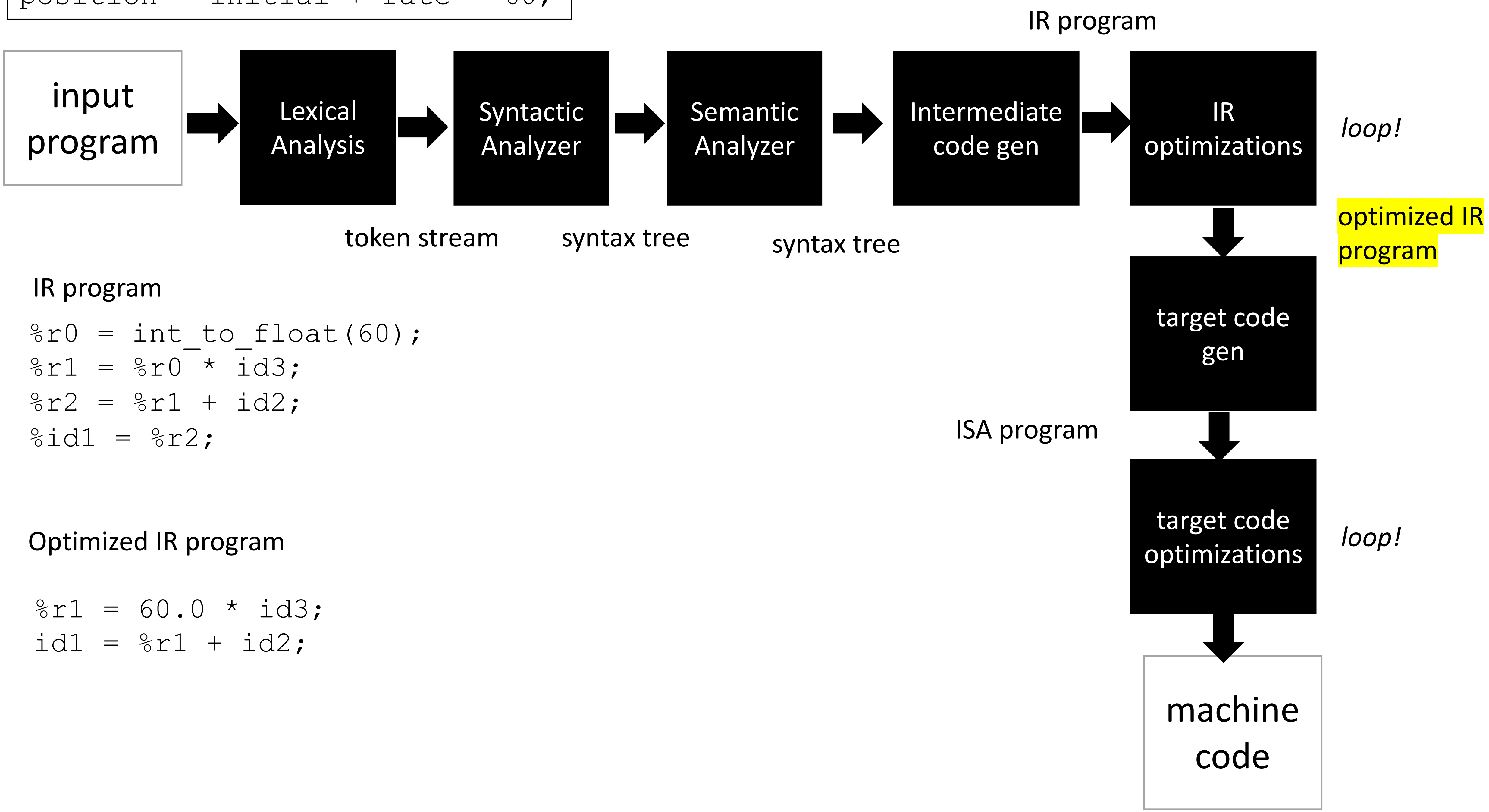
```
position = initial + rate * 60;
```



IR program

```
%r0 = int_to_float(60);
%r1 = %r0 * id3;
%r2 = %r1 + id2;
%id1 = %r2;
```

```
position = initial + rate * 60;
```



input program

Lexical Analysis

token stream

Syntactic Analyzer

syntax tree

Semantic Analyzer

syntax tree

Intermediate code gen

IR program

IR optimizations

loop!

optimized IR program

target code gen

ISA program

target code optimizations

loop!

machine code

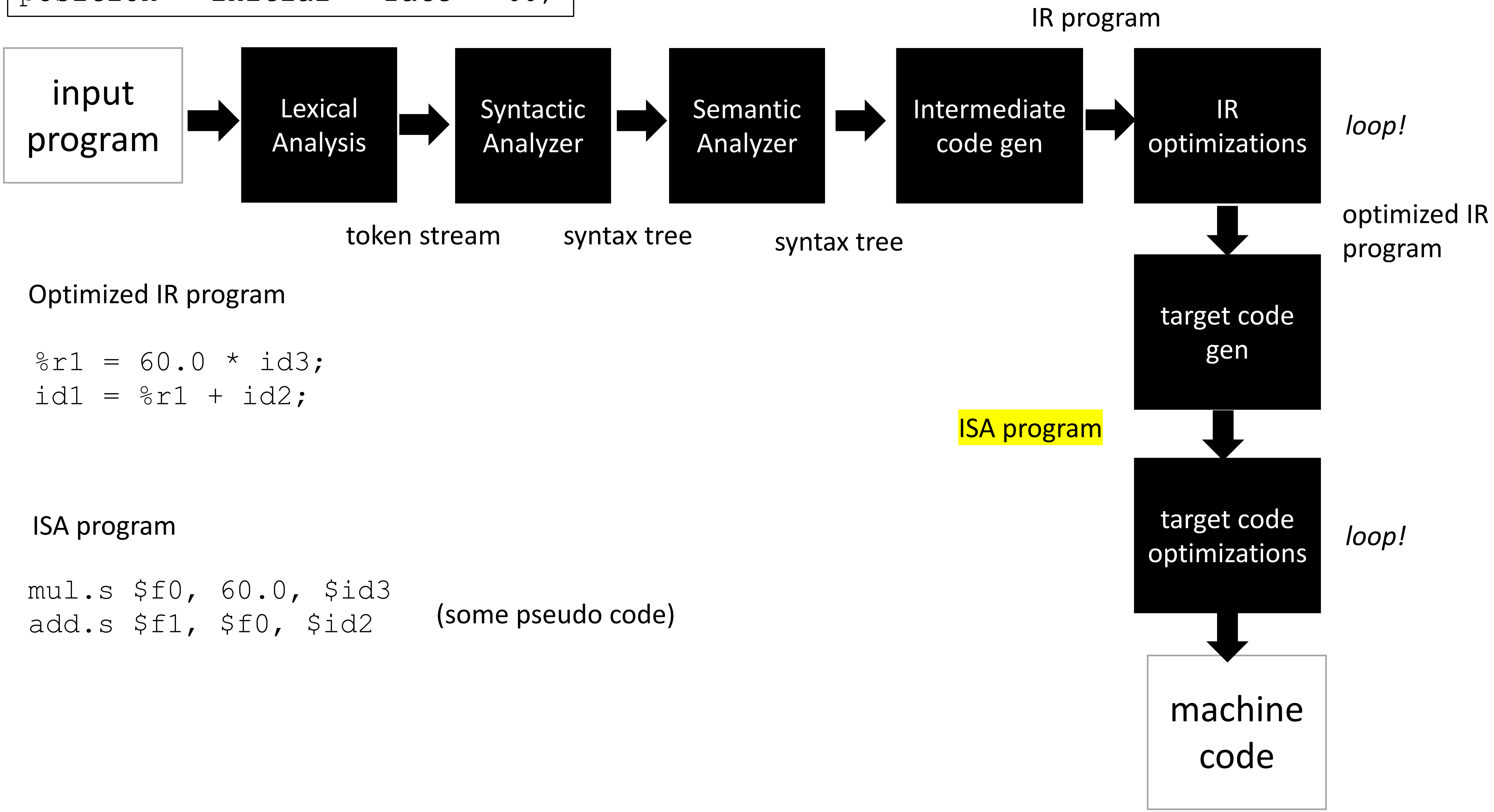
IR program

```
%r0 = int_to_float(60);
%r1 = %r0 * id3;
%r2 = %r1 + id2;
%id1 = %r2;
```

Optimized IR program

```
%r1 = 60.0 * id3;
id1 = %r1 + id2;
```

```
position = initial + rate * 60;
```



Optimized IR program

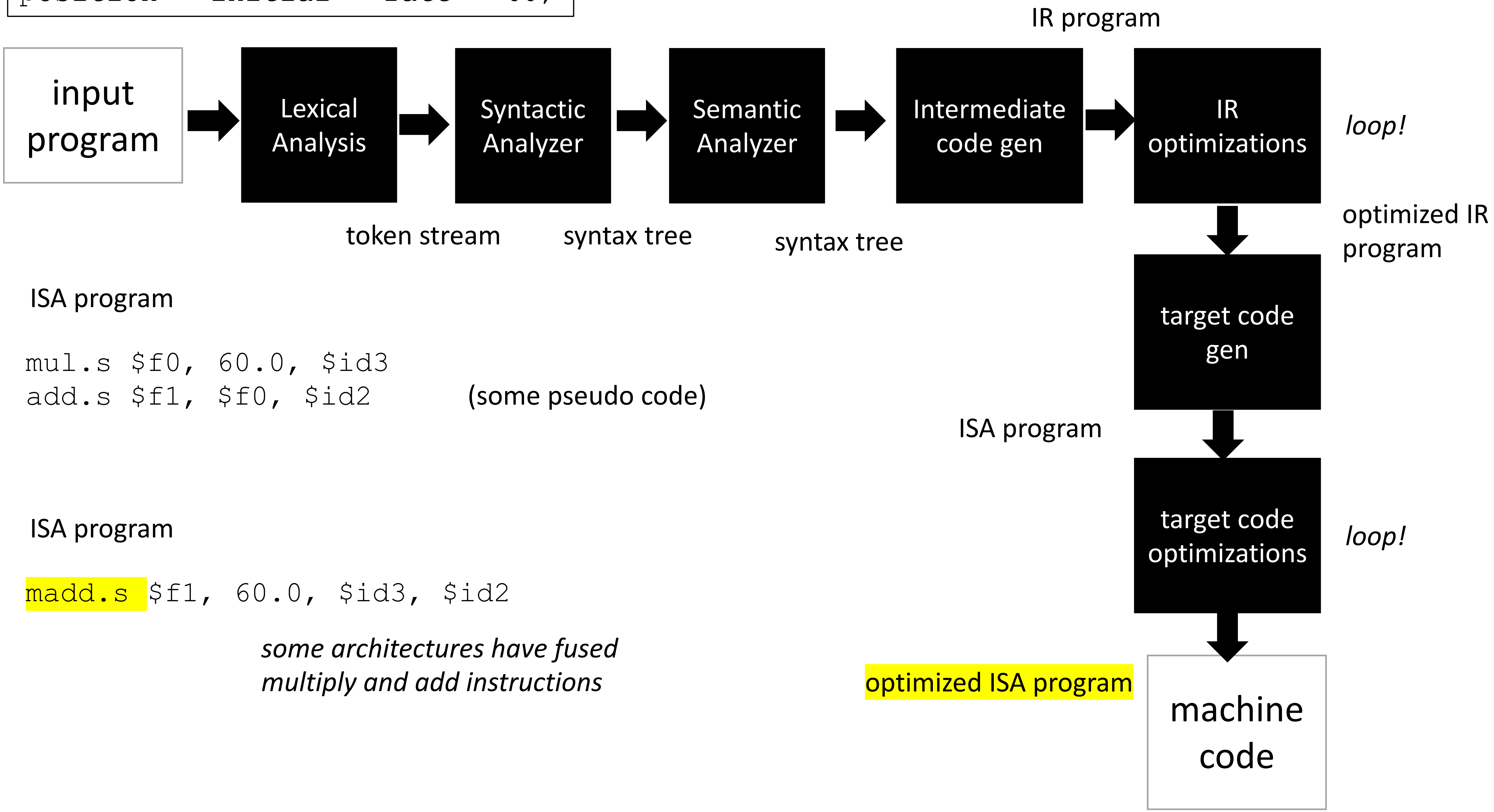
```
%r1 = 60.0 * id3;
id1 = %r1 + id2;
```

ISA program

```
mul.s $f0, 60.0, $id3
add.s $f1, $f0, $id2
```

(some pseudo code)

```
position = initial + rate * 60;
```



ISA program

```
mul.s $f0, 60.0, $id3
add.s $f1, $f0, $id2
```

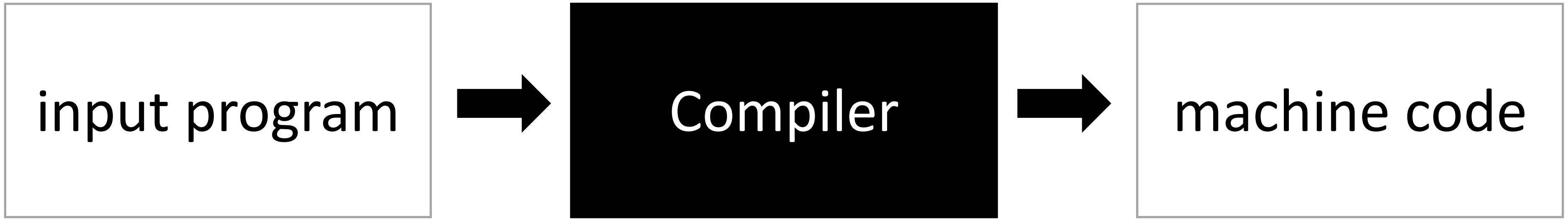
(some pseudo code)

ISA program

```
madd.s $f1, 60.0, $id3, $id2
```

some architectures have fused multiply and add instructions

Compiler Architecture



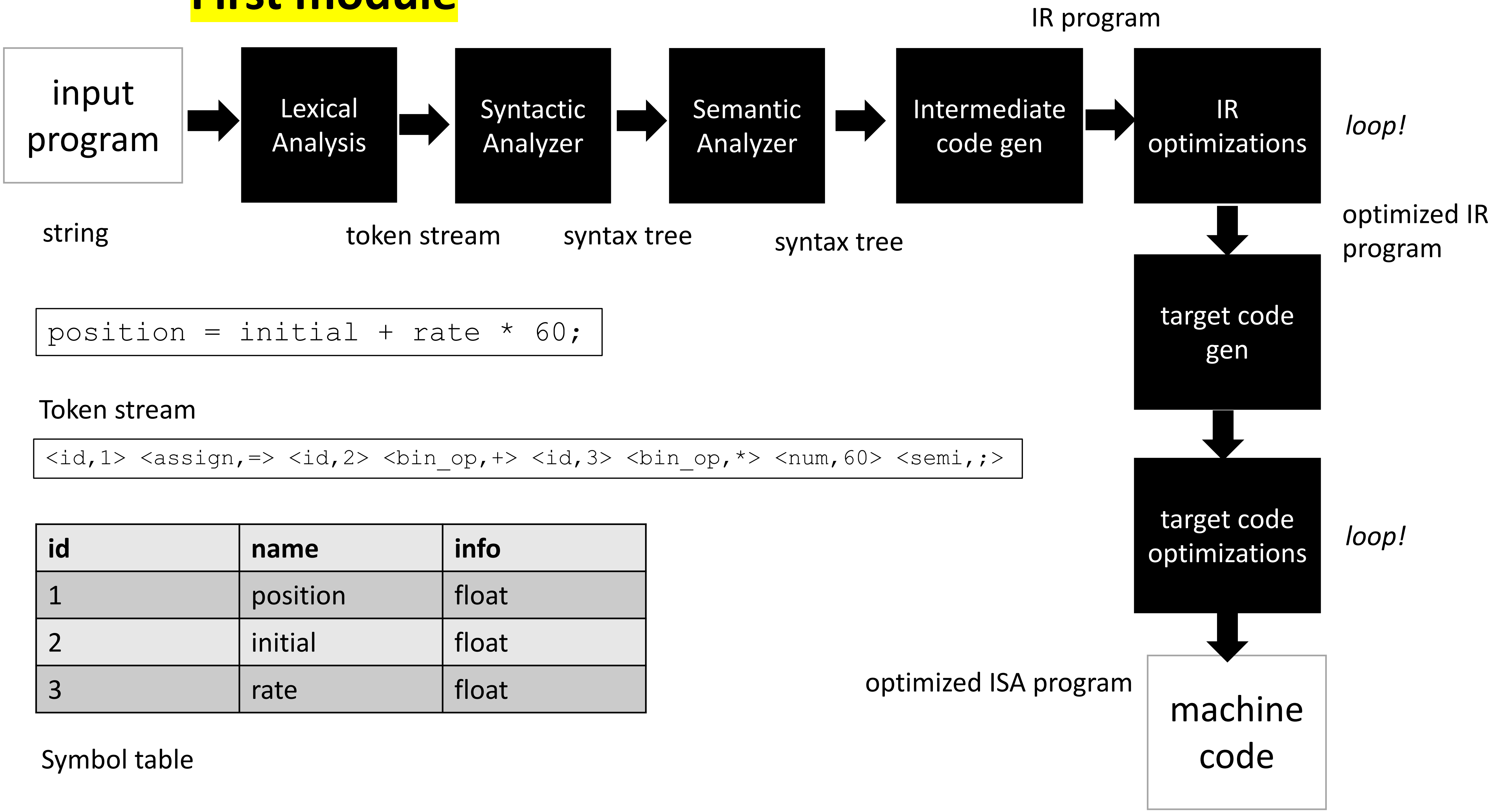
input program

Compiler

machine code

Now you've seen a journey through a compiler!

First module



```
position = initial + rate * 60;
```

Token stream

```
<id,1> <assign,=> <id,2> <bin_op,+> <id,3> <bin_op,*> <num,60> <semi,;>
```

id	name	info
1	position	float
2	initial	float
3	rate	float

Symbol table

Next Class

- **Lexical Analysis**