HOW IT WORKS

COMPILER

@codechips

Andrew Proctor

@codechips

Cody

popupdev04@gmail.com

You Code in your favorite
High Level Languages

```
int main() {
    int a = 5;
    a = a * 5;
    return 0;
}
```

@codechips

Cody

popupdev04@gmail.com

But computers only Understand
Machine Code !

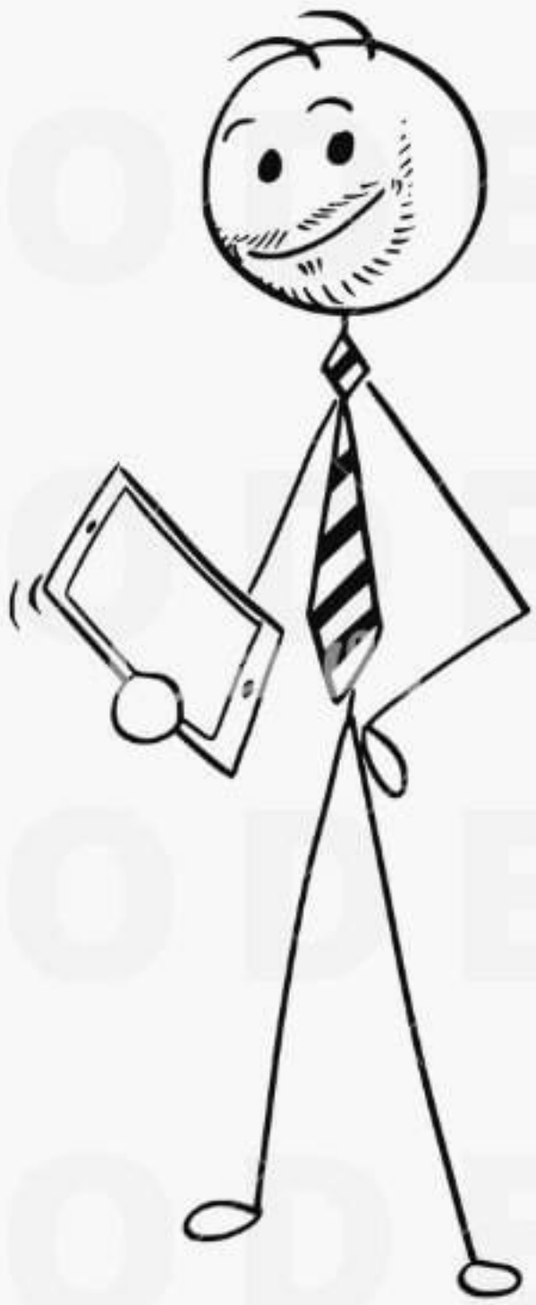
```
int main() {
    int a = 5;
    a = a * 5;
    return 0;
}
```

That is when compiler came to the rescue

A compiler is a program that translates high-level language (for eg: Java) into low-level language (object or machine program)

But how does he actually **work** ?

I do this by going through
6 major phases



Lexical analysis
Syntax analysis
Semantic analysis
IR code generation
Optimization
Output code generation

Lexical analysis

Our original source code would be split up into tokens and kept inside of a computer's memory

```
[KEYWORD,"int"] [ID,"main"] [LPAREN] [RPAREN]
[LBRACE]KEYWORD,"int"] [ID,"a"] [EQUALS] [INT,"5"]
[SEMICOLON][ID,"a"] [EQUALS] [ID,"a"] [MULTIPLY]
[INT,"5"] [SEMICOLON][KEYWORD,"return"] [INT,"0"]
[SEMICOLON][RBRACE]
```


@codechips Cody popupdev04@gmail.com

PARSING →

Syntax analysis

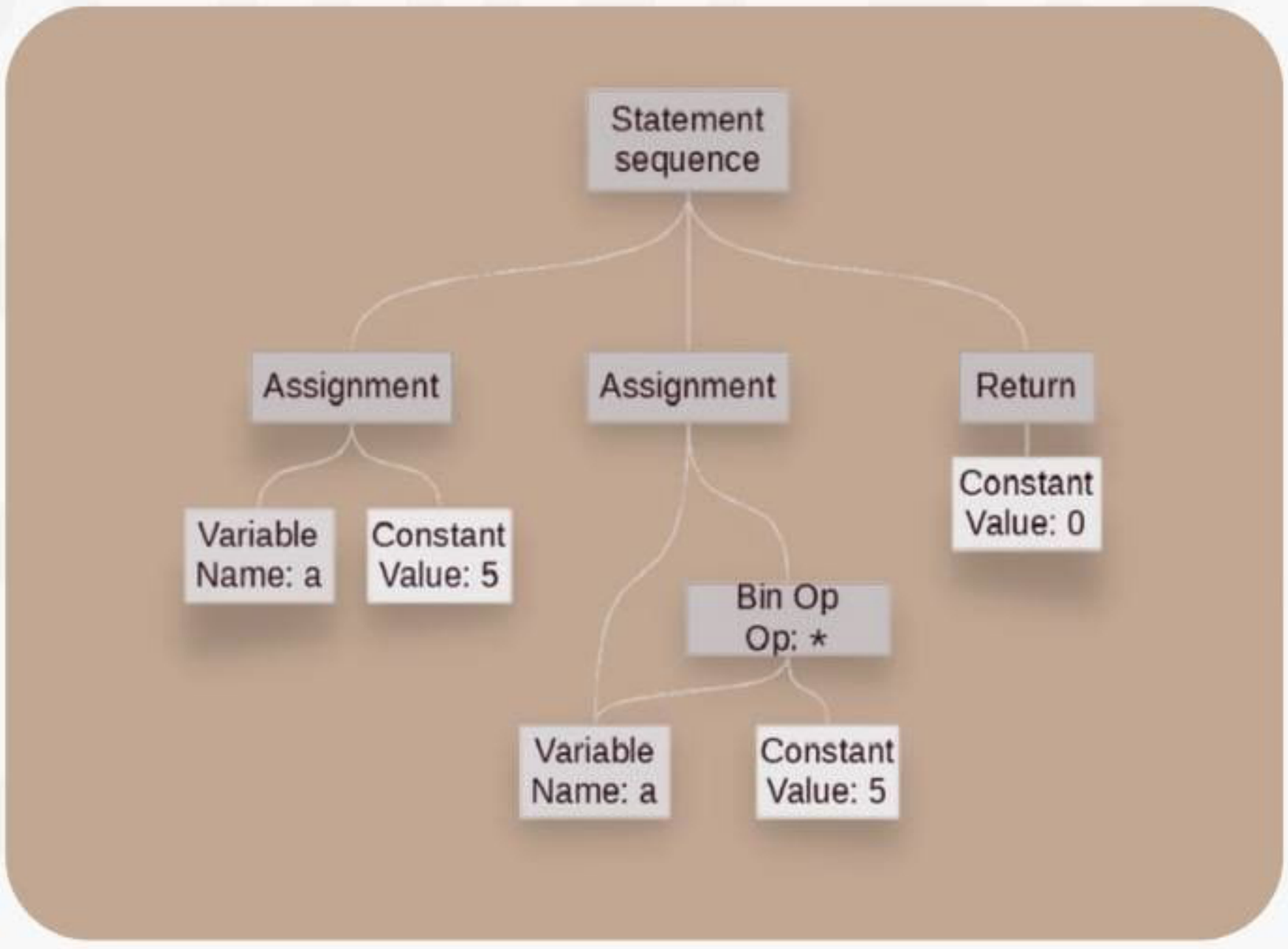
The compiler verifies that the code's syntax is correct

Semantic analysis

The compiler verifies the validity of the code's logic

•-------→

It would generate an abstract syntax tree



IR code generation

Intermediate representation of code is generated to make it easier to translate

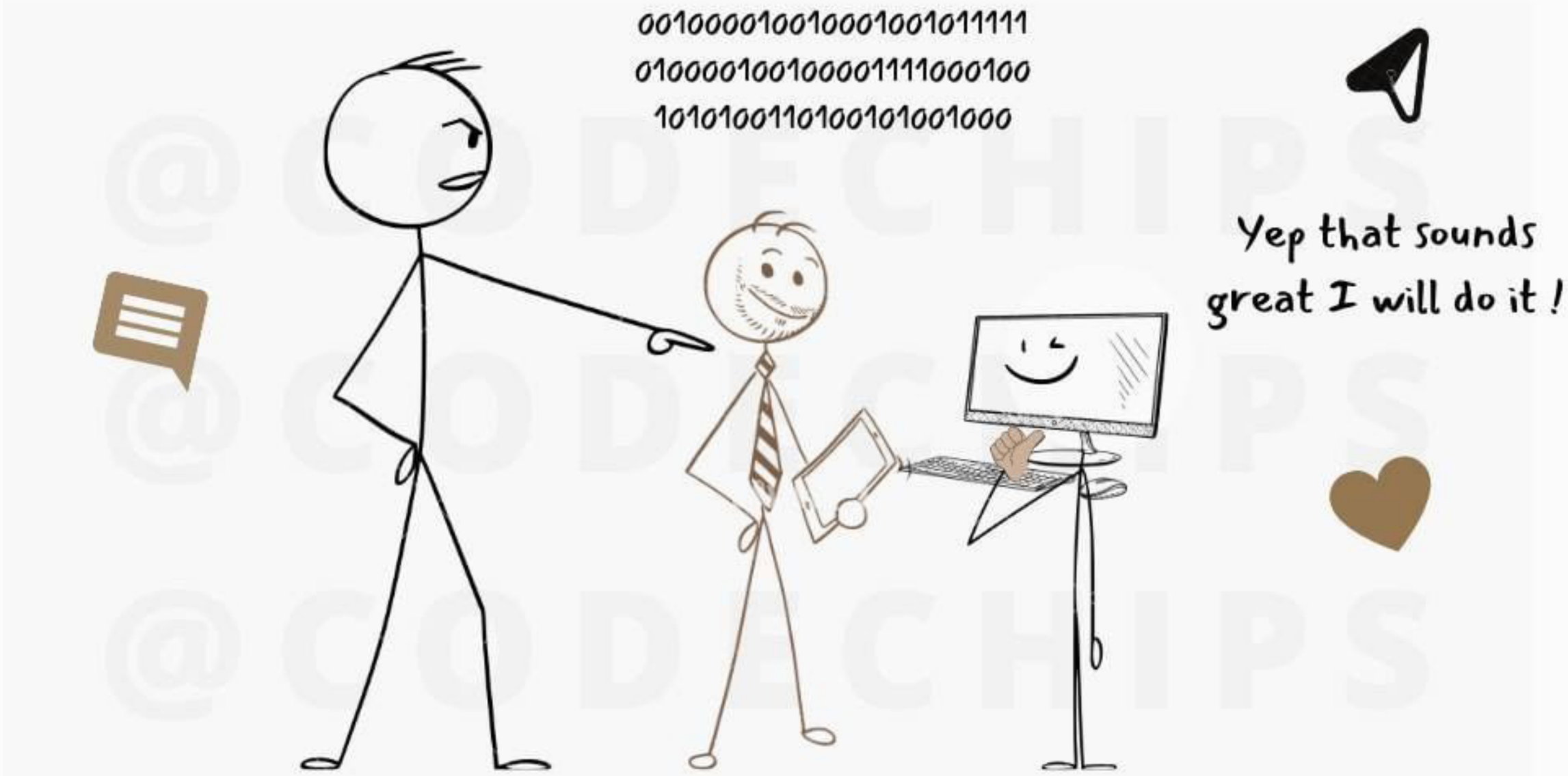
Optimization

The compiler optimizes the IR code in preparation for the final code generation

Output code generation

The compiler generates the final output code

@codechips Cody popupdev04@gmail.com



@codechips

Cody

popupdev04@gmail.com