Simple Language Compiler Manual

March 25, 2025

1 Introduction

Welcome to the official documentation for the Simple Language Compiler. This manual provides comprehensive information about the compiler's features, syntax, and operation.

2 Language Features

2.1 Core Components

- **Variables**: `x = 10`
- **Arithmetic**: `+`, `-`, `*`, `/`
- **Comparisons**: `<`, `>`, `<=`, `>=`, `==`
- **Control Flow**: `if-else`, `while`
- **Functions**: `def func(): ... return ...`
- **Arrays**: `arr[index] = value`

2.2 Data Types

Type	Example	Description
Integer	`42`	Whole numbers
Boolean	`true`	Logical values
Array	`[1,2,3]`	Indexed collections

3 Compilation Process

3.1 Lexical Analysis

Converts source code to tokens:

```
Input: "x = 5 + 3"
Tokens: [IDENT(x), EQ, NUM(5), PLUS, NUM(3)]
```

Listing 1: Tokenization example

3.2 Syntax Analysis

Builds Abstract Syntax Tree (AST):

```
Assignment
          ID: x
          BinaryOp(+)
              Num(5)
              Num(3)
```

Listing 2: AST example

3.3 Code Generation

Produces x86-64 assembly:

```
mov eax, 5
add eax, 3
mov [x], eax
```

Listing 3: Generated assembly

4 Examples

4.1 Basic Program

```
def factorial(n):
    if n < 1:
        return 1
    return n * factorial(n-1)
```

Listing 4: Factorial function

4.2 Generated Assembly

```
factorial:
RET
POP R1
CMP R1, 0
JZ ELSE_0
PUSH 1
POP R1
RET
JMP END_0
ELSE_0:
END_0:
PUSH n
PUSH n
PUSH 1
POP R1
POP R2
SUB R1, R2
PUSH R1
CALL factorial
POP R1
PUSH R1
POP R1
POP R2
MUL R1, R2
```

```
PUSH R1
POP R1
RET
```

Listing 5: Factorial assembly

5 Error Handling

Error Type	**Example**	**Solution**
Syntax Error	`x = 5 +`	Complete expression
Type Mismatch	`"5" + 3`	Convert types
Undefined Var	`print(y)`	Declare variable