# Compiler Design

Text book :

- Compilers – Principles, Techniques & Tools , Aho, Ravi Sethi, D. Ullman

- Introduction to compilers – Phases of a compiler

## Introduction to Compilers

- Compiler – A program
- It takes as input a program written in one language and translates it into an equivalent program written in another language
- Input language – the source language
- Output language - Target language
- Important part of the translation process
  - Reports the presence of errors in the source program to its users.

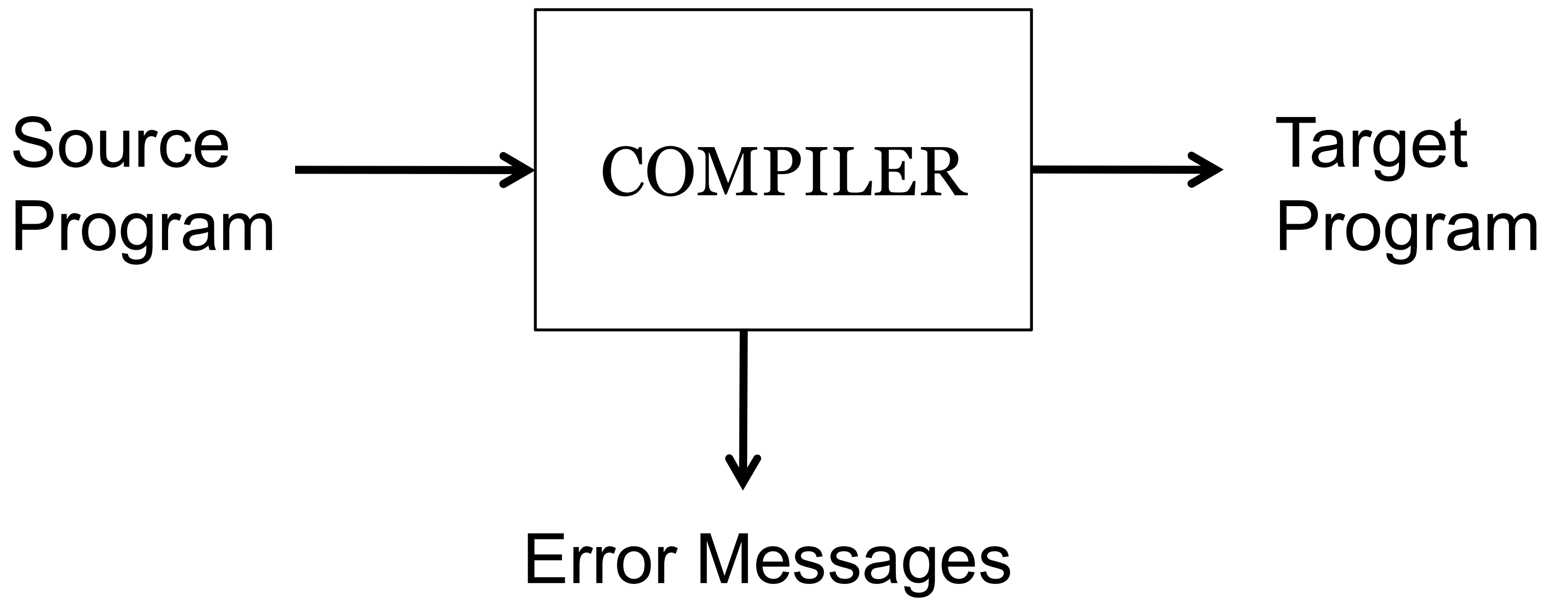


Figure 1.1 A Compiler

## Introduction to Compilers

- Source language
    - Any traditional programming language like c, cpp
- Target languages
    - Any other programming language or the machine language of any computer from a microprocessor to a super computer.

- The first compilers started to appear in the early 1950's.
- Initially it was a tedious task to write a compiler.
- The first Fortran compiler took 18 staff-years to implement.
- Complex task
- The basic tasks are the same
- By understanding these tasks, we can construct compilers for a wide variety of source languages and target languages using the same basic techniques.

## The Analysis-Synthesis Model of Computation

- Two parts to compilation : Analysis and Synthesis.
- The Analysis part
  - Breaks up the source program into constituent pieces
  - Creates an intermediate representation of the source program
- The synthesis part
  - Constructs the desired target program from the intermediate representation

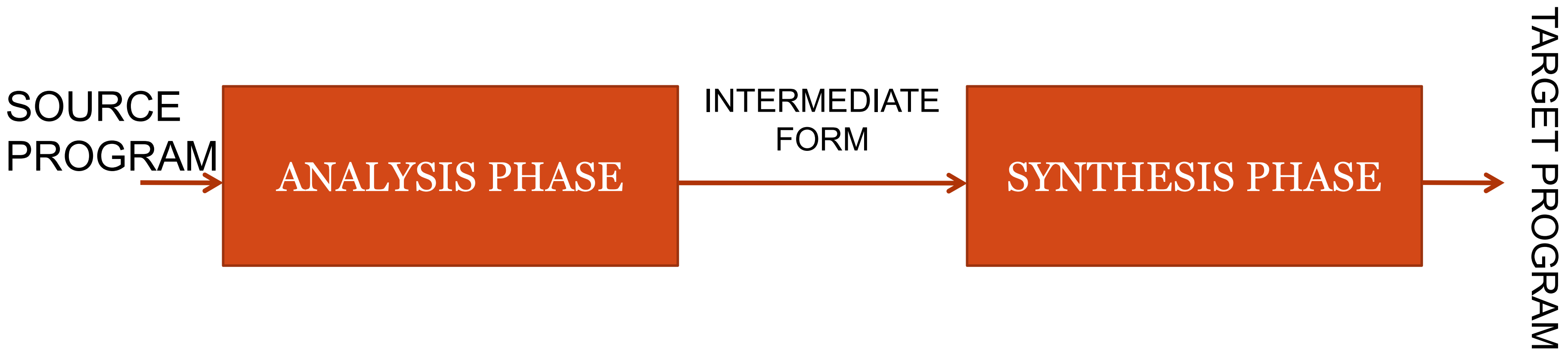


Fig 2 : The Analysis-Synthesis Model

## The Analysis Part

- Determines the operations implied by the source program
- Records them in a hierarchical structure called a tree.
- Uses a special kind of tree - a syntax tree
  - Nodes represents operations
  - Children of the node represents the arguments of the operation

## Analysis of the source program

- Introduction to analysis phase and illustration
- Consists of three phases
  - **Linear analysis**
    - The stream of characters making up the source program is read from left-to-right
    - Grouped into tokens – sequence of characters having a collective meaning
- **Hierarchical analysis**
  - Characters or tokens are grouped hierarchically into nested collections with collective meaning
- **Semantic analysis**
  - Certain checks are performed to ensure that the components of a program fit together meaningfully

## Lexical Analysis

- Linear analysis is called lexical analysis or **scanning**
- For eg: in lexical analysis the characters in the assignment statement

  position := initial + rate * 60

  would be grouped into the following tokens

1. The identifier position
2. The assignment symbol :=
3. The identifier initial
4. The plus sign
5. The identifier rate
6. The multiplication sign
7. The number 60

- The blanks separating the characters of these tokens would be normally eliminated

## Syntax Analysis

- Hierarchical analysis is also called syntax analysis or **parsing**
- Groups tokens into grammatical phrases that are used by the compiler to synthesize output
- The grammatical phrases are usually represented by a parse tree

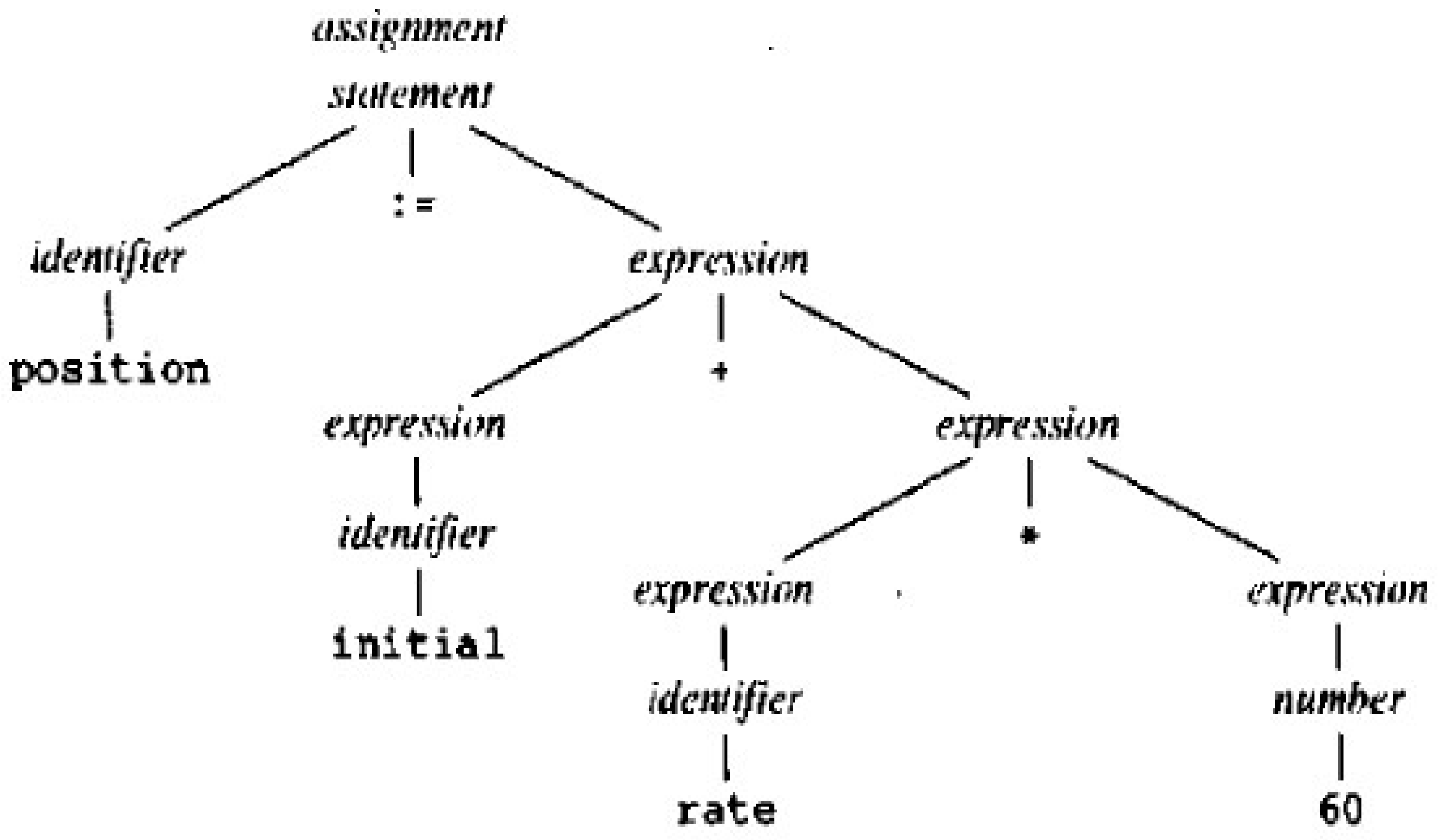


**Fig. 1.4.** Parse tree for `position := initial + rate * 60`.

## Semantic Analysis

- Checks the source program for semantic errors and gathers type information from the subsequent code-generation phase
- Uses the hierarchical structure determined by the syntax-analysis phase to identify the operators and operands of expressions or statements
- Important component – type checking
  - Checks that each operator has operands that are permitted by the source language specification

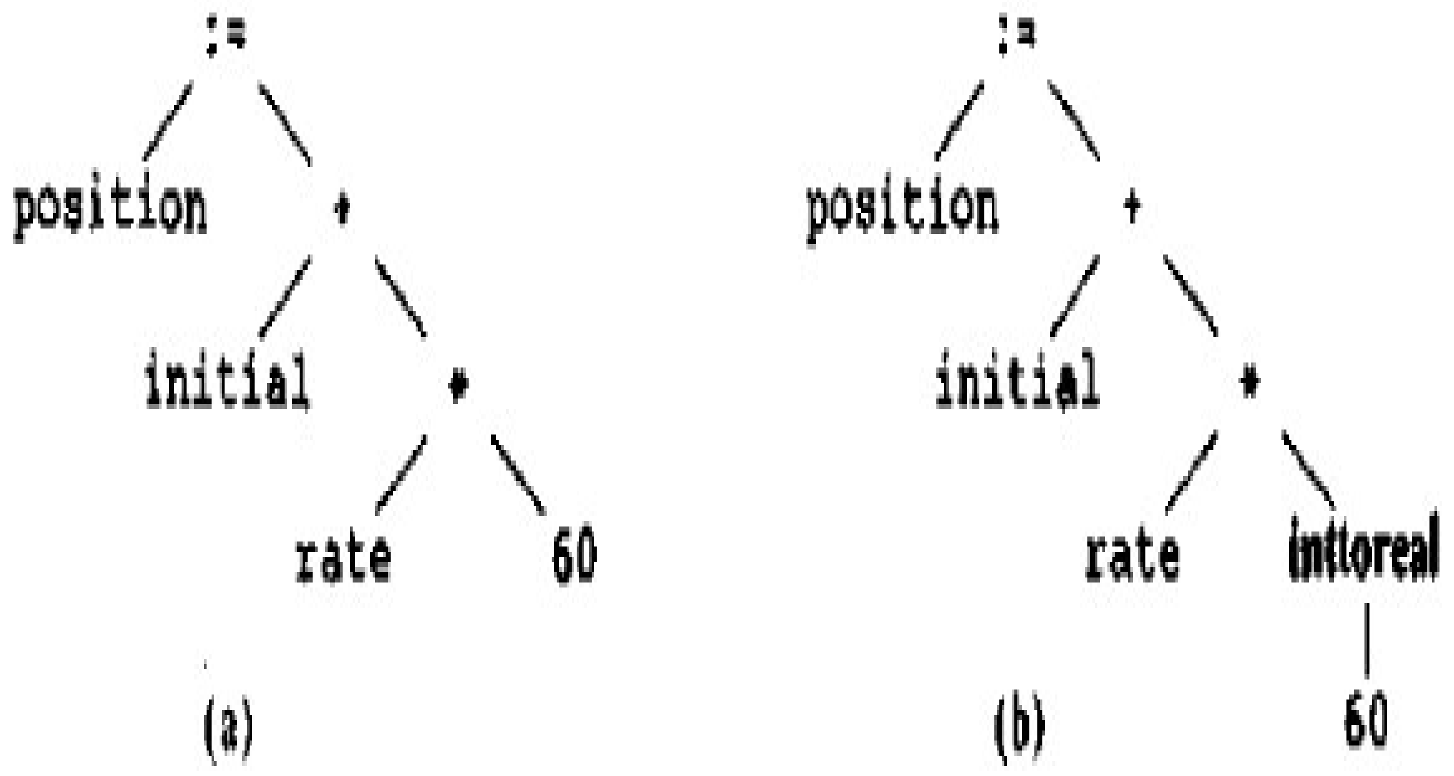


Fig. 1.5. Semantic analysis inserts a conversion from integer to real.

## Synthesis Phase

- Input – Intermediate representation (Parse Tree)
- Phases
  - Intermediate Code generation
  - Code optimization
  - Code generation

## Phases of compilation

- A compiler operates in phases
- Each phase transforms the source program from one representation to another
- The first three phases, forms the analysis phase and the last three phases forms the synthesis phases.
- The two other activities
  - Symbol-table management and
  - Error handling,
    - interacts with all the six phases – lexical analysis, syntax analysis, semantic analysis, intermediate code generation, code optimization and code generation.

## The phases of a compiler

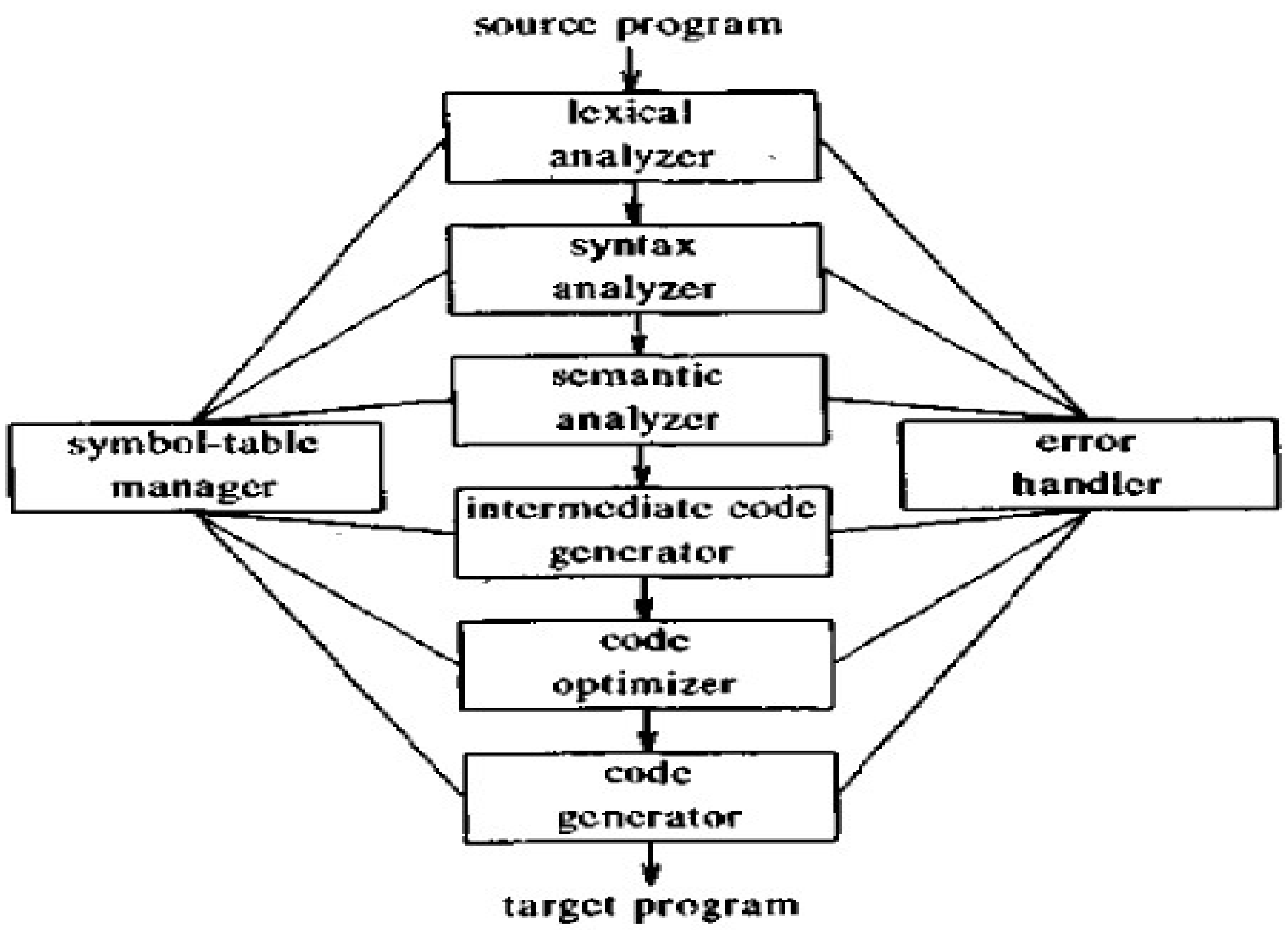
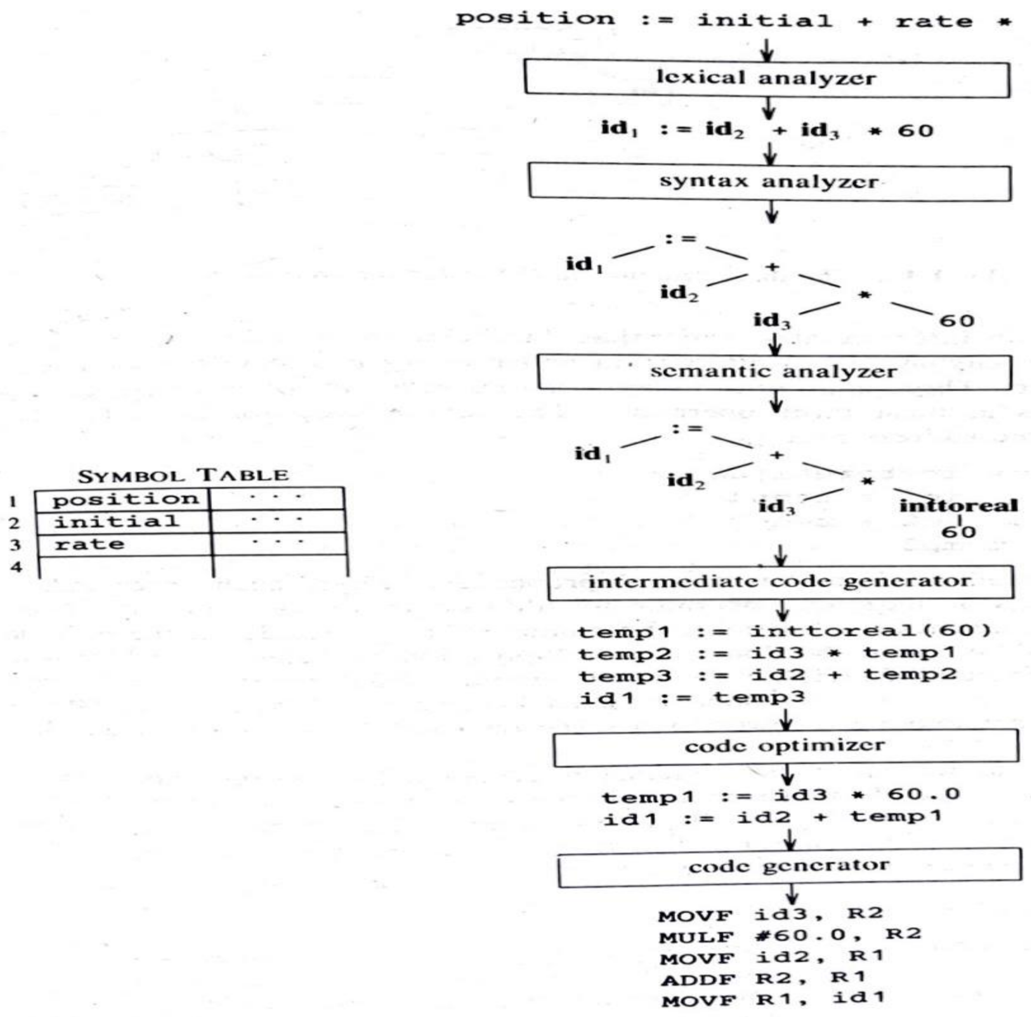


Fig. 1.9. Phases of a compiler.

# Phases of compilation - Example



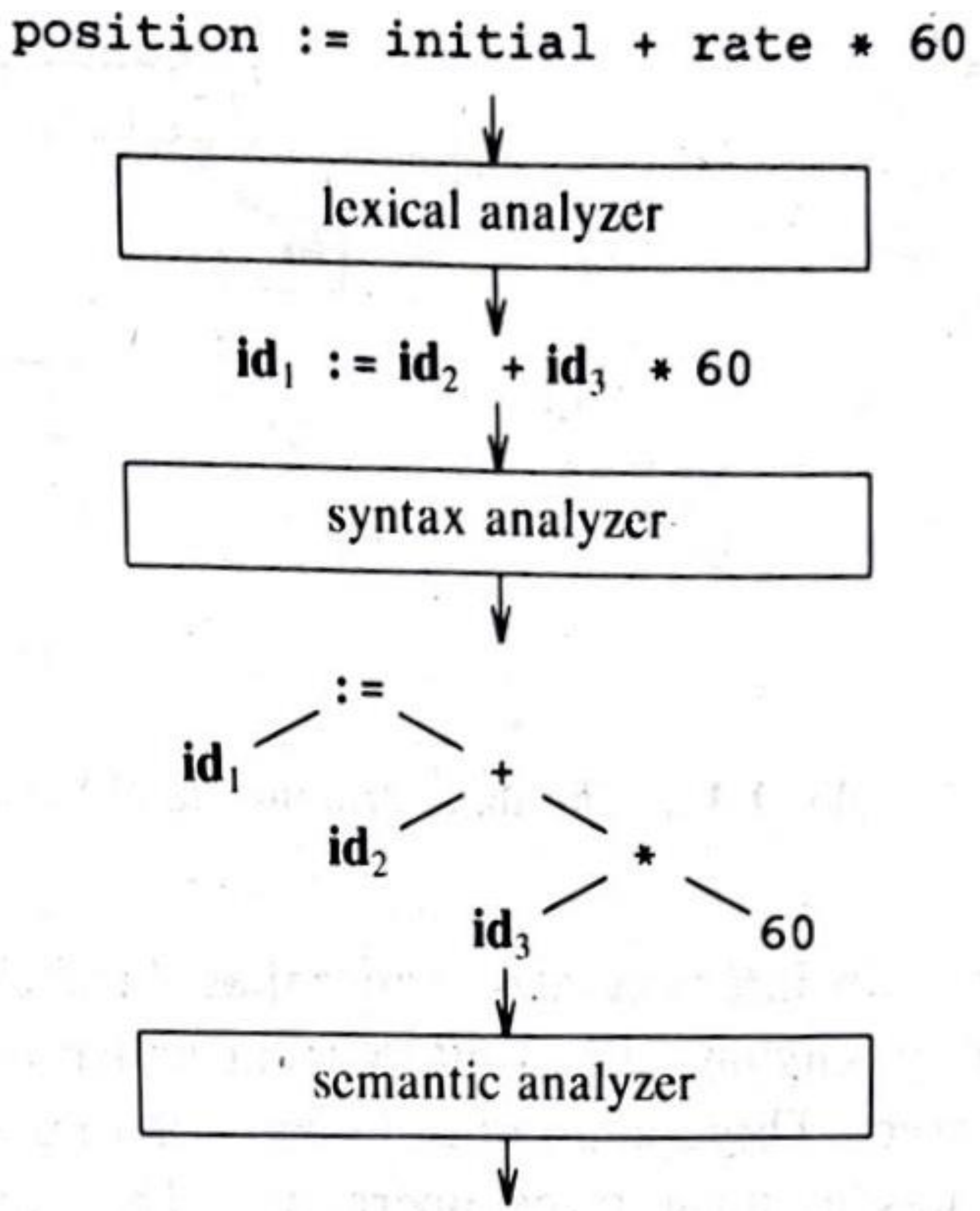
SYMBOL TABLE

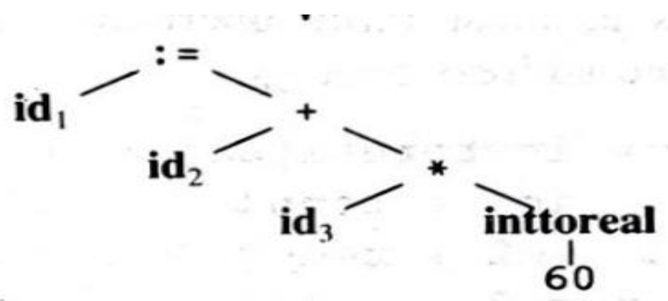
| | | |
|---|---|---|
| 1 | position | ... |
| 2 | initial | ... |
| 3 | rate | ... |
| 4 | | |

```
temp1 := inttoreal(60)
temp2 := id3 * temp1
temp3 := id2 + temp2
id1 := temp3
```

```
temp1 := id3 * 60.0
id1 := id2 + temp1
```

```
MOVF id3, R2
MULF #60.0, R2
MOVF id2, R1
ADDF R2, R1
MOVF R1, id1
```


position := initial + rate * 60
lexical analyzer
id1 := id2 + id3 * 60
syntax analyzer
:=
id1
+
id2
*
id3
60
semantic analyzer



intermediate code generator

```
temp1 := inttoreal(60)
temp2 := id3 * temp1
temp3 := id2 + temp2
id1 := temp3
```

code optimizer

```
temp1 := id3 * 60.0
id1 := id2 + temp1
```

code generator

```
MOVF id3, R2
MULF #60.0, R2
MOVF id2, R1
ADDF R2, R1
MOVF R1, id1
```

SYMBOL TABLE

| | | |
|---|---|---|
| 1 | position | ... |
| 2 | initial | ... |
| 3 | rate | ... |
| 4 | | |