INTERNATIONAL UNIVERSITY VNUHCM

PRINCIPLES OF PROGRAMMING LANGUAGES

LAB 5. PYTHON COLLECTIONS

PURPOSE: After this lab, you are able to use Python collections.

EXERCISES:

L5E1. Practise Python collections, including *lists, tuples, sets* and *dictionaries,* by using the following tutorials and examples:

Lists: https://www.w3schools.com/python/python_lists.asp
Tuples: https://www.w3schools.com/python/python_tuples.asp
Sets: https://www.w3schools.com/python/python_sets.asp
Dictionaries: https://www.w3schools.com/python/python_dictionaries.asp

L5E2. Create Symbol Tables[1] for Exercises L2E1 and L2E2.

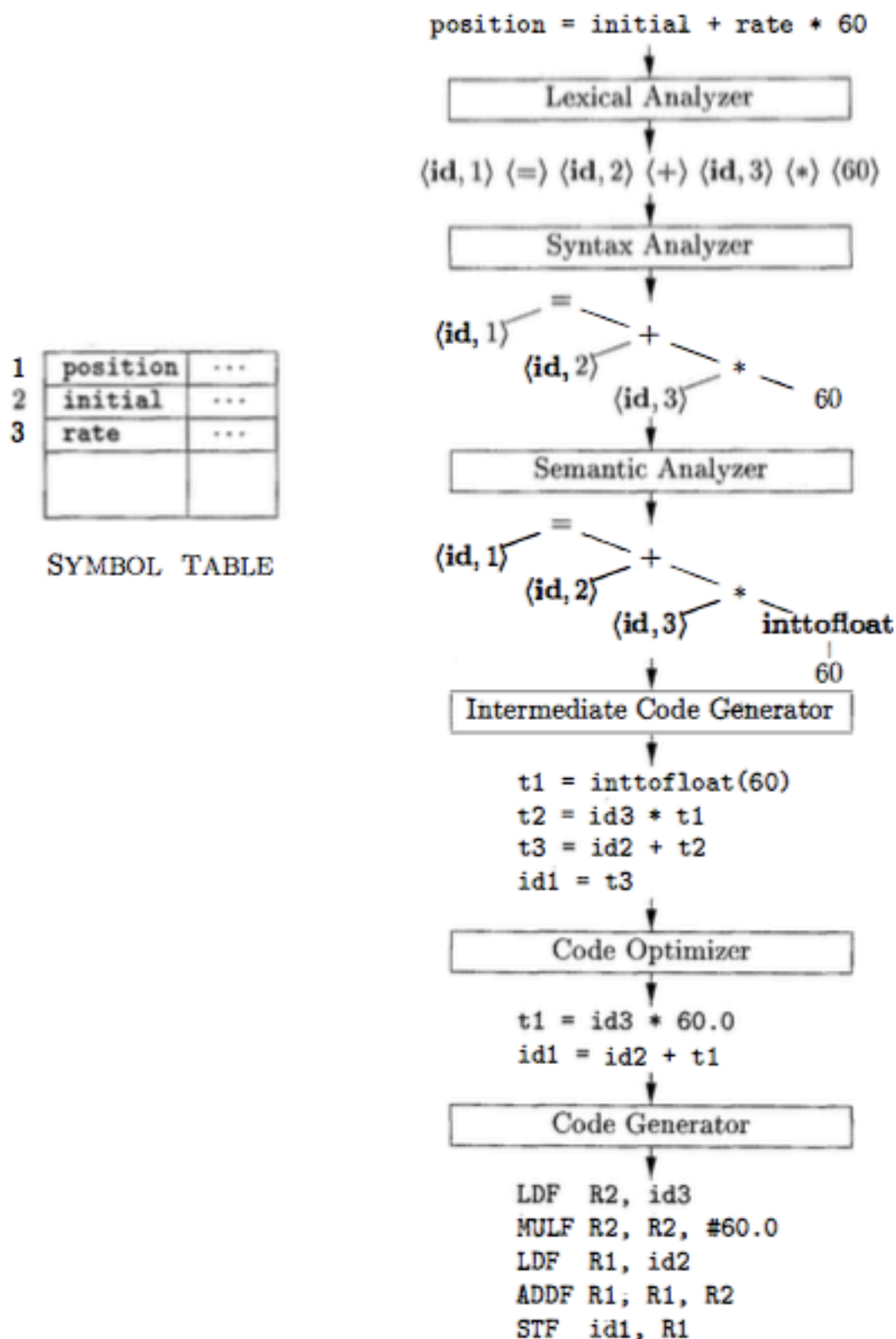[1] A part of a compilation process mentioned Figure 1.6 and Figure 1.7 in (Aho et. al., 2007)

position = initial + rate * 60

Lexical Analyzer

⟨**id**, 1⟩ ⟨=⟩ ⟨**id**, 2⟩ ⟨+⟩ ⟨**id**, 3⟩ ⟨*⟩ ⟨60⟩

Syntax Analyzer

```
            =
          /   \
   (id, 1)      +
             /     \
        (id, 2)      *
                  /     \
             (id, 3)      60
```

Semantic Analyzer

```
            =
          /   \
   (id, 1)      +
             /     \
        (id, 2)      *
                  /     \
             (id, 3)   inttofloat
                            |
                            60
```

1	position	...
2	initial	...
3	rate	...

SYMBOL TABLE

Intermediate Code Generator

```
t1 = inttofloat(60)
t2 = id3 * t1
t3 = id2 + t2
id1 = t3
```

Code Optimizer

```
t1 = id3 * 60.0
id1 = id2 + t1
```

Code Generator

```
LDF   R2, id3
MULF  R2, R2, #60.0
LDF   R1, id2
ADDF  R1, R1, R2
STF   id1, R1
```

Figure 1.7: Translation of an assignment statement

character stream
↓

Lexical Analyzer

token stream
↓

Syntax Analyzer

syntax tree
↓

Semantic Analyzer

syntax tree
↓

Symbol Table

Intermediate Code Generator

intermediate representation
↓

Machine-Independent
Code Optimizer

intermediate representation
↓

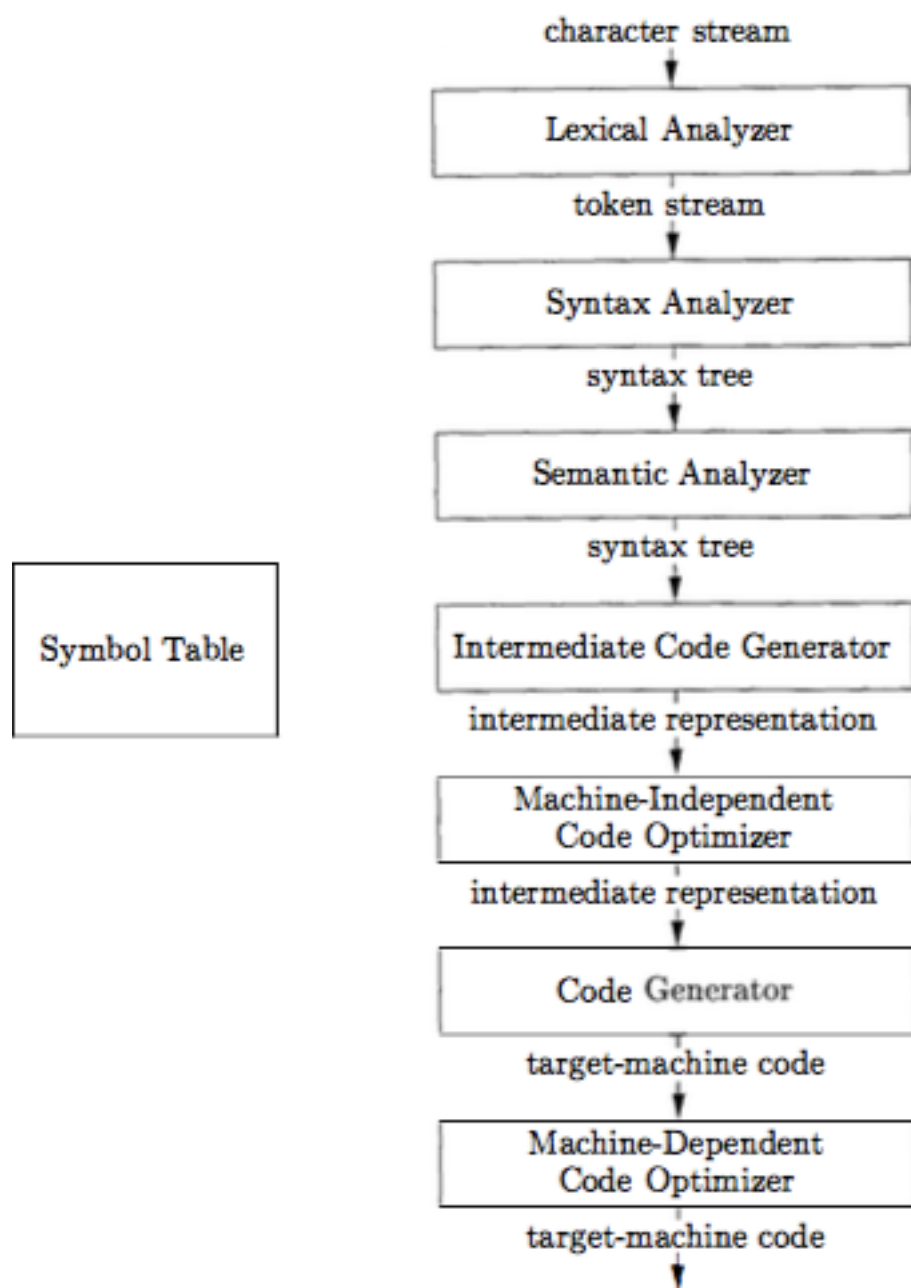Code Generator

target-machine code
↓

Machine-Dependent
Code Optimizer

target-machine code
↓

Figure 1.6: Phases of a compiler