Phases of Compilers

Phases of Compilers

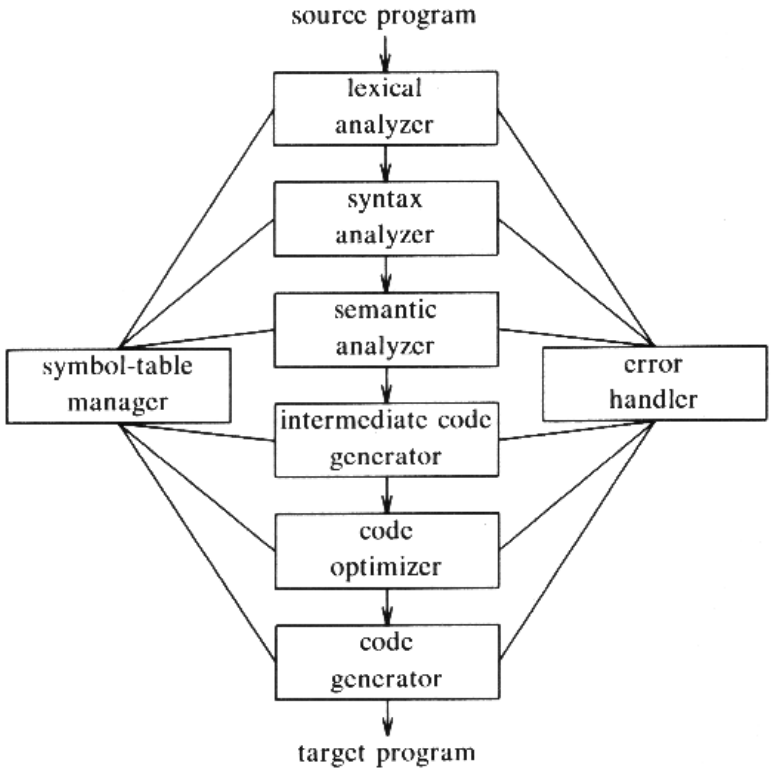
or

Analysis Synthesis Model of Compiler

Analysis-Synthesis Model

- Compilation: Analysis & Synthesis
- Analysis:
  - Break source program into pieces
  - Intermediate representation
- Synthesis: construct target program from tree

The Phases of A Compiler



The Phases of A Compiler

- Phases
- First three phases: analysis portion
- Last three phases: synthesis portion
- Symbol-table management phase
- Error handler phases

1.2 Analysis of the source program

- Three phases

1. Lexical analysis
   - Divide source program into tokens
2. Hierarchical (syntactical) analysis
   - Tokens grouped hierarchically
3. Semantic analysis
   - Ensure components fit meaningfully

Lexical Analysis

- Linear analysis: lexical analysis, scanning

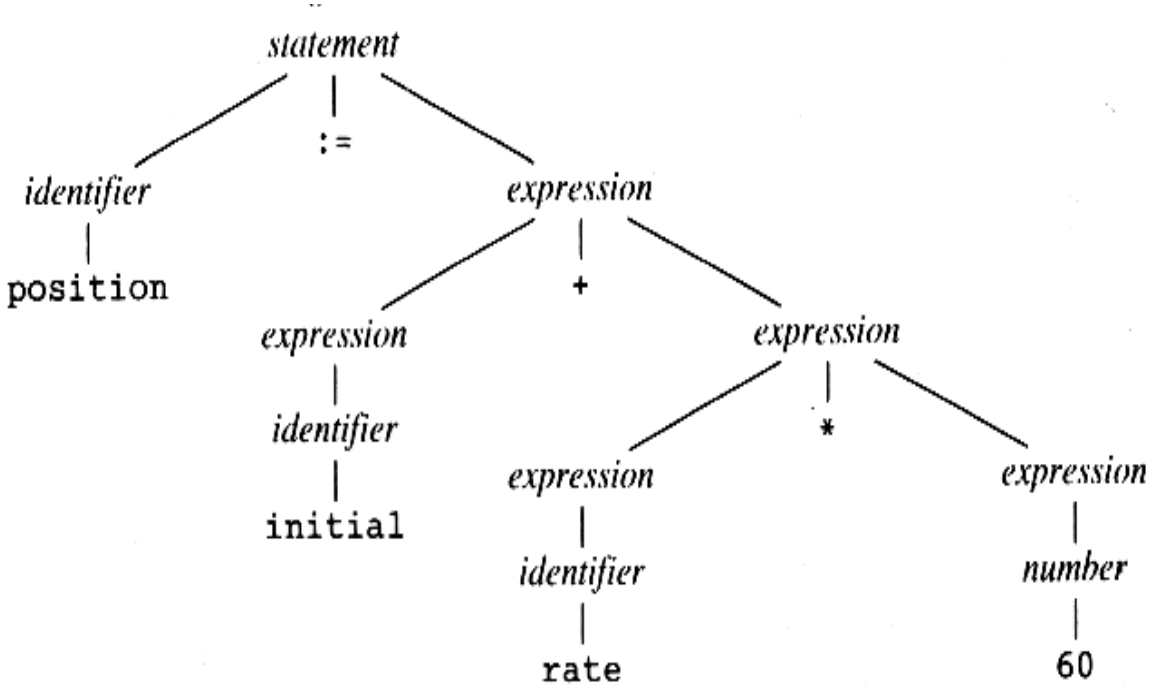
❖ e.g., *position*:= *initial+rate*60*

1. Identifier *position*
2. Assignment symbol ": ="
3. Identifier *initial*
4. "+" sign
5. Identifier *rate*
6. "*" sign
7. number 60

Syntax Analysis

- Hierarchical analysis: parsing or syntax analysis
  - Group tokens into grammatical phrases
- ❖ Grammatical phrases: parser tree

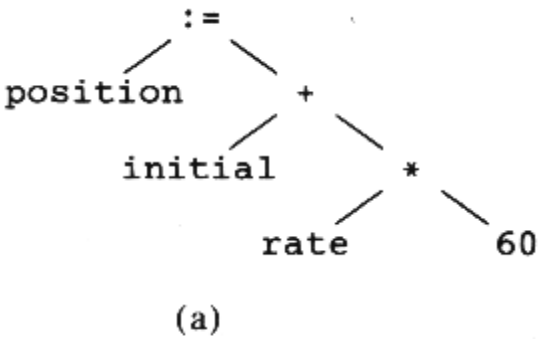
Syntax Analysis



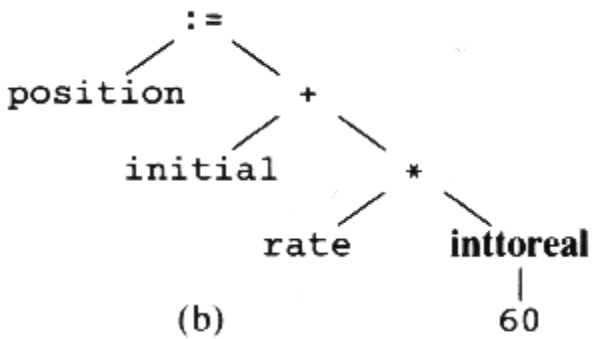
Semantic Analysis

- Check semantic error
- Gather type information for code-generation
- Using hierarchical structure to identify operators and operands
- Doing type checking
  - E.g, using a real number to index an array (error)
  - Type convert
  - E.g, Fig.1.5 inttoreal(60) if initial is a real number

Semantic Analysis



(a)



(b)

Symbol-table Management

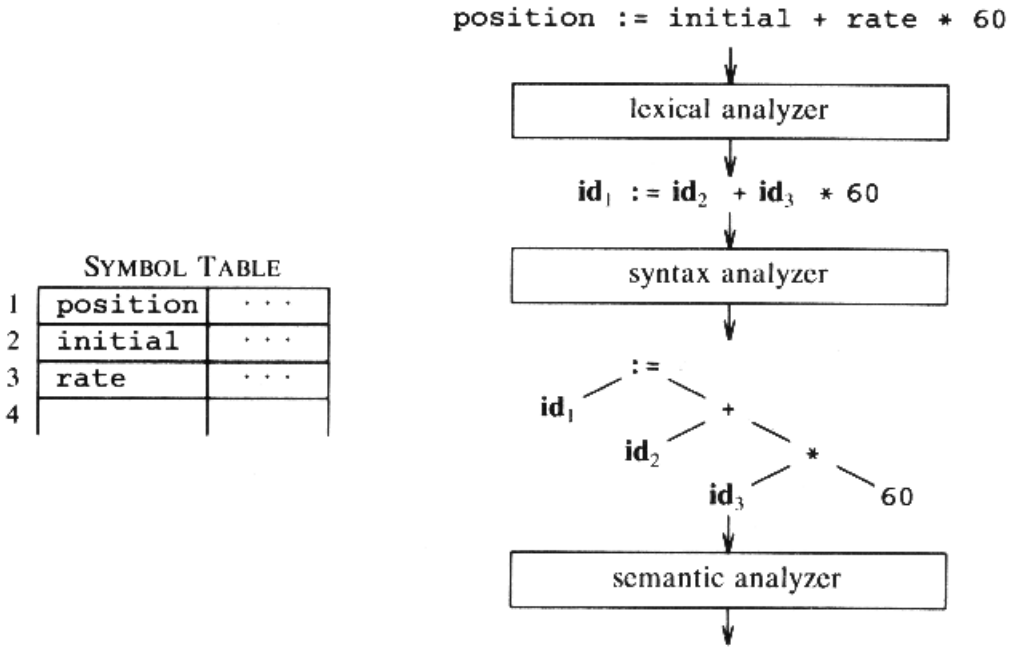
- To record the identifiers in source program
  - Identifier is detected by lexical analysis and then is stored in symbol table
- To collect the attributes of identifiers

  (not by lexical analysis)
  - Storage allocation : memory address
  - Types
  - Scope (where it is valid, local or global)
  - Arguments (in case of procedure names)
    - Arguments numbers and types
    - Call by reference or address
    - Return types

Error Detection and Reporting

- Syntax and semantic analysis handle a large fraction of errors
- Lexical phase: could not form any token
- Syntax phase: tokens violate structure rules
- Semantic phase: no meaning of operations
  - Add an array name and a procedure name

Translation of A Statement



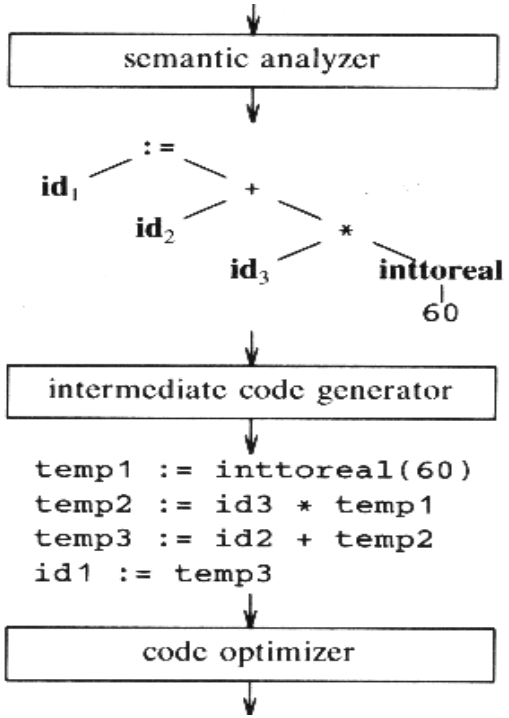
SYMBOL TABLE

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

Translation of A Statement

SYMBOL TABLE

1	position	· · ·
2	initial	· · ·
3	rate	· · ·
4		



Intermediate Code Generation

- Represent the source program for an abstract machine code
- Should be easy to produce
- Should be easy to translate into target program
- Three-address code (at most three operands)
  - temp2:=id3*temp1

Code Optimization

- Improve the intermediate code
- Faster-running machine code
  - while(i-2<=temp)
    { temp=temp-1 }
  - x=i-2

    while(x<=temp)
    { temp=temp-1 }

Code Generation

- Generate relocation machine code or assembly code
  - MOVF id3, R2
    MULF #60.0, R2
    MOVF id2, R1
    ADDF R2, R1
    MOVF R1, id1

Thank You