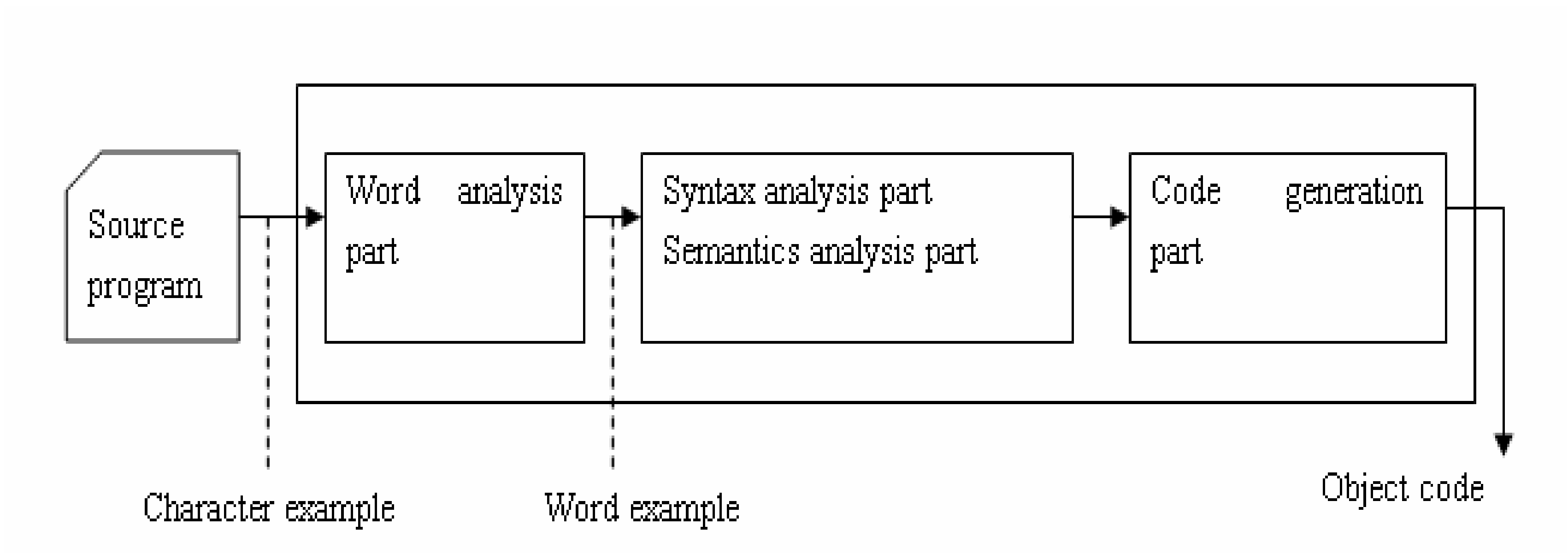
Unit 2.
The phases of a Compiler

Main phases of a compiler



Phases of a compiler

- Scanner (Lexical Analyser)

  Stream of characters making up the source program is read from left to right and grouped into tokens (sequences of characters having a collective meaning)

- Parser (Syntax Analyser)

  Group the tokens of the source program into grammatical phrases that are used by the compiler to synthesize output

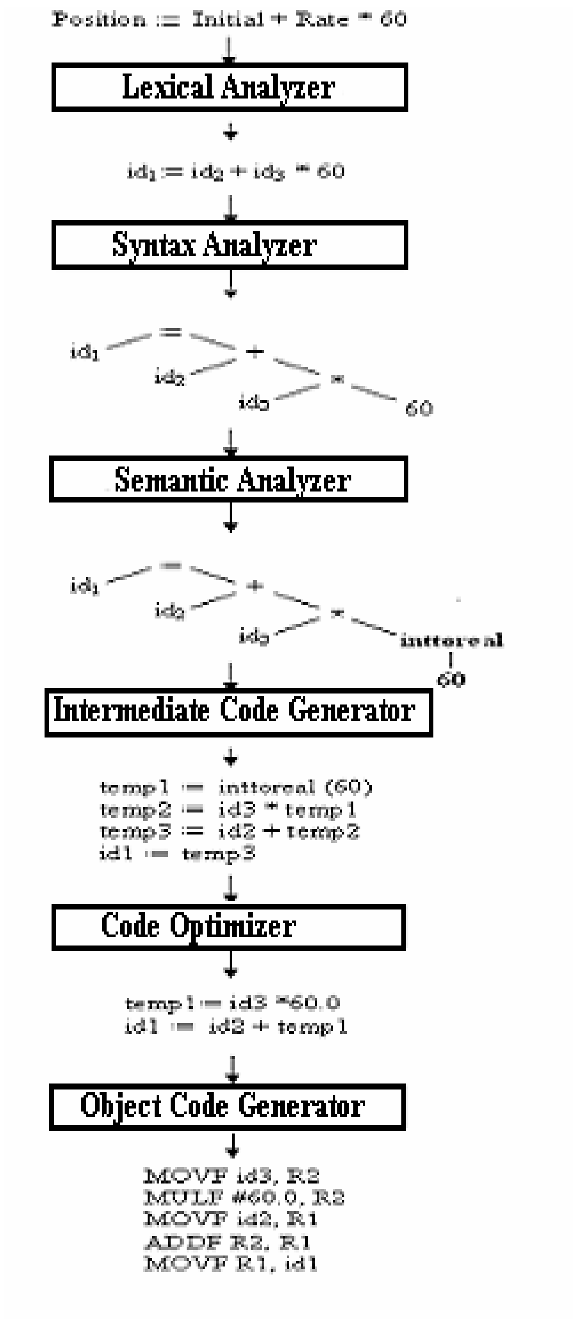
Phases of a compiler

- Semantic Analysis:Check the source program for semantic errors and gather type information for the subsequent code generation phase.
- Intermediate Code Generation: Generate an intermediate representation as a program for an abstract machine.

Phases of a compiler

- Code optimization : Improve the intermediate code so that faster running code will result
- Code generation: Generation of target code, consisting normally of relocatable machine code or assembly code

Translation of a statement



```
id1 := id2 + id3 * 60
```

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

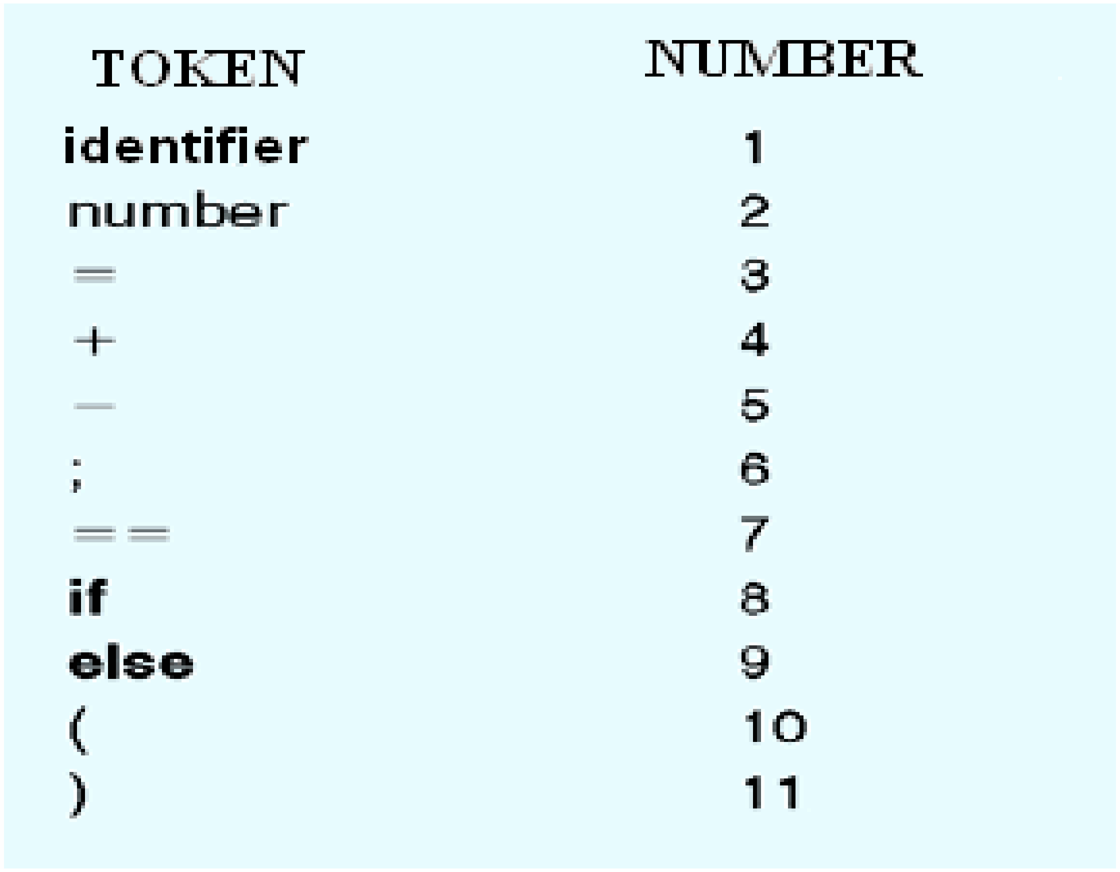
```
temp1 := id3 *60.0
id1 := id2 + temp1
```

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

Phase 1:Lexical Analysis

- Scanner: Converts the stream of input characters into a stream of tokens that becomes the input to the following phase (parsing)
- Tasks of a scanner
  - Group characters into tokens
    - Token: the syntax unit
  - Categorization of tokens.

Types of tokens



TOKEN	NUMBER
identifier	1
number	2
=	3
+	4
−	5
;	6
==	7
if	8
else	9
(	10
)	11

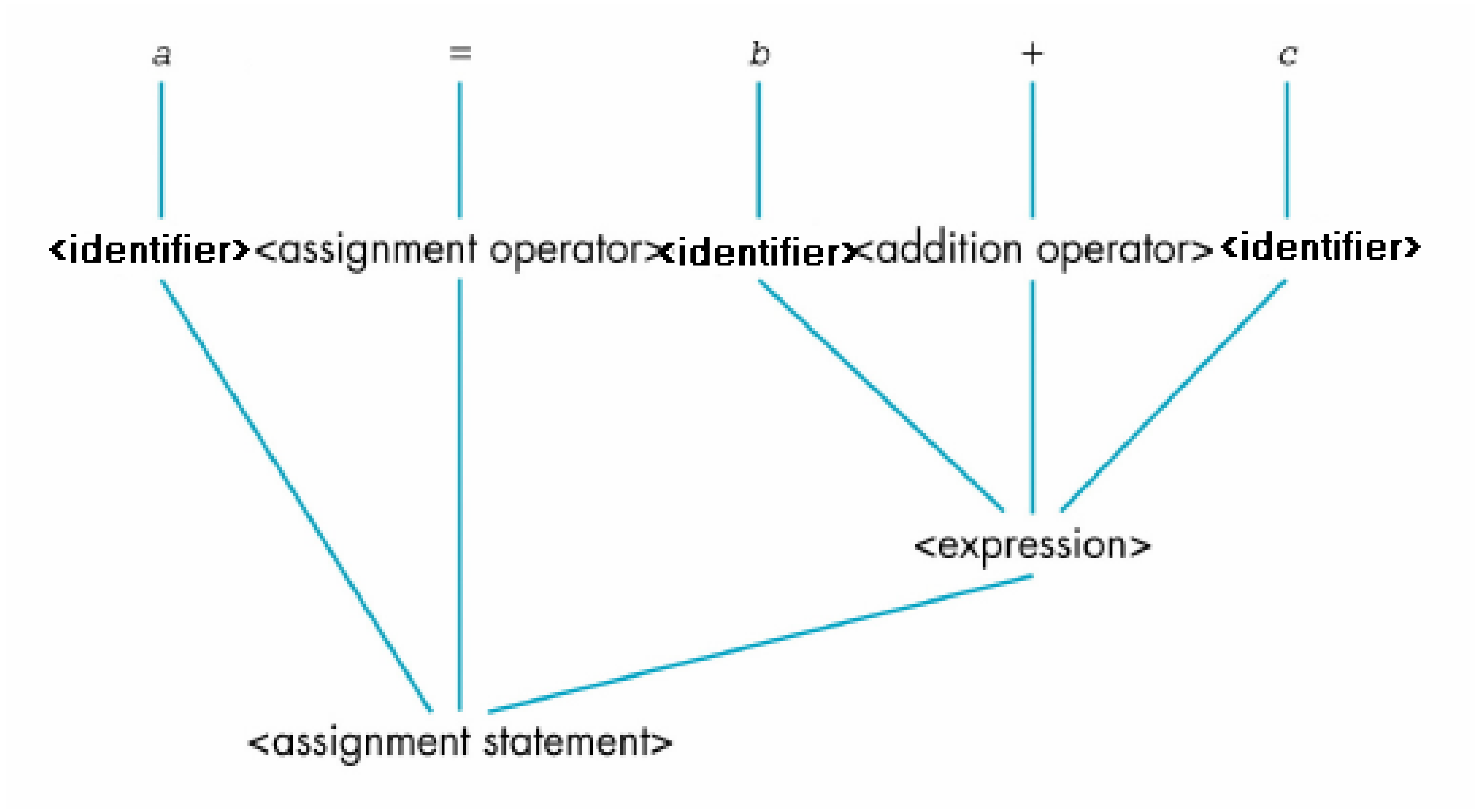
Phase 2: Parsing

- The process of determining if a string of token can be generate by a grammar
- Is executed by a parser

Phase 2: Parsing

- Output of a parser:
  - Parse tree (if any)
  - Error Message (otherwise)
- If a parse tree is built succesfully, the program is grammatically correct

Example: statement a = b + c



Grammars,languages, BNF,syntax diagrams

- Bộ phân tích cú pháp cần đưa ra phân tích cho mỗi câu của ngôn ngữ (chương trình)
- BNF (Backus-Naur Form) is is a meta language used to express grammars of programming languages
- Syntax Diagrams : A pictorial diagram showing the rules for forming an instruction in a programming language, and how the components of the statement are related. Syntax diagrams are like directed graphs.

Grammars,languages, BNF,syntax diagrams

- BNF and formal grammars use 2 types of symbol
- Terminals :
  - Tokens of the language
  - Never appear in the left side of any production
- Nonterminals
  - Intermediate symbol to express structures of a language
  - Must be in a left side of at lease one production
  - Enclose in <>

Grammars,languages, BNF,syntax diagrams

- Start symbol :
  - Nonterminal of the first level
  - Appear at the root of parse tree

Parsing: Concept and techniques

- Continuously apply grammatical rules until a string of terminal is generated.
- If the parser convert first symbol into the input string, it is syntactically correct
- Otherwise, string is not syntactically correct

Parsing: Concept and techniques

- The most important thing of a compiler: grammar
- Grammar includes all structures of a program
- Not includes any other rule

Parsing: Concept and techniques

- Grammar must be unambiguous
- If grammar is ambiguous, more than one parse tree can be created

Phase 3: Semantic Analysis

- Certain check are performed to ensure that the components of a program fit together meaningfully
- To generate code, source program must be syntactically and semantically correct

Phase 4: Intermediate code generation

- Source program is transferred to an equivalent program in intermediate code by intermediate code generator
- *Intermediate code* is machine independent

Advantages of Intermediate Code

1. **Easy to translate into object code.**
2. **Code optimizer can be applied before code generation**
3. **Decrease time cost**

Intermediate Code

- Alternatives
  - Parse Tree
  - Postfix Notation
  - Three Address Code

Phase 5: Code Generator

- Input: Intermediate code of source program
- Output: Object program
  - Assembly code
  - Virtual machine code

Problems

- Input
- Output
- Set of instruction
- Register allocation
- Object machine