Chap. 4 프로그래밍 언어의 구문과 구현기법

P.71 그림4.1 가상컴퓨터(Virtual Computer)

4.1 언어구문

4.1.1프로그래밍 언어의 어휘 p.72

- A - Z
- 0 - 9
- Special Character

- IBM EBCDIC (Extended Binding Coded Decimal Interchange Code) : 8 bit
- Ansi ASCII(American Standard Code for Information Interchange) p73, 표4.1

o 프로그래밍 어휘구조: pp.74-76

o reserved word
장점 : 읽기 쉽고, reserved word symbol table
error recovery
단점 : 많으면 기억하기 힘듦.

4.1.2 BNF 표기법

· BNF(Backus-Naur form)
:production rule들의 집합
P77 <표 4.2> identifier 정의

cf) meta-symbol : 표현하려는 언어의 일부분이 아니라
그 언어를 표현하려고 사용된 특수 기호

· EBNF
: 반복 부분 { }, 선택 부분 [ ]을 간결하게 표현
pp 78-79

4.1.3 Syntax Diagram

pp 80-81
cf) p79 표4.3 <--> p83 diagram

4.1.4 Parse Tree

: 주어진 BNF를 이용하여 표현 대상을 root로 하고
terminal node를 왼쪽에서 오른쪽으로 나열하여
검증하고자 하는 표현과 같이 되는 tree
pp84 그림4.3 ↔ p.77 표4.2

o abstract syntax tree : p85

4.1.5 모호성, 결합성 및 우선순위

o 모호함(ambiguous):
P.84: 표4.4 BNF
p.86 - 87: 3-2*5 에 대한 2가지 서로다른 parse tree 와 abstract syntax tree

o 모호성 제거(disambiguation):
방법 1) 연산 순서 표시 --> 순위 폭포(precedence cascade)
예) * 가 - 보다 우선순위

<표 4.4> 변경
<exp> ::= <exp> - <exp> | <term>
<term> ::= <term> * <term> | (<exp>) | <number>

p.88
---> 아직도 문제 잔존
7-3-2를 (7-3)-2 또는 7-(3-2)로 파싱함

방법 2) left-associate, right-associate
<exp> ::= <exp> - <exp>를

---> left-associate <exp> ::= <exp> - <term>
right-associate <exp> ::= <term> - <exp>

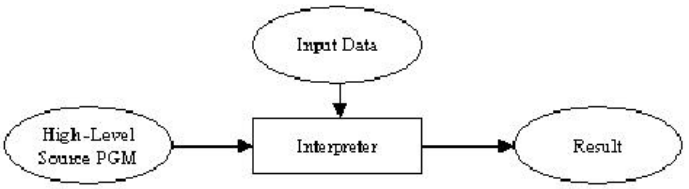
p.89 표4.5: 표4.4의 개정 문법

4.2 Programming Language Implementation < translation / interpretation

(1) Translation
① Compiler p93, p94
② Assembler 그림. 4.4, 4.6

③ linkage editor
④ loader
⑤ Preprocessor

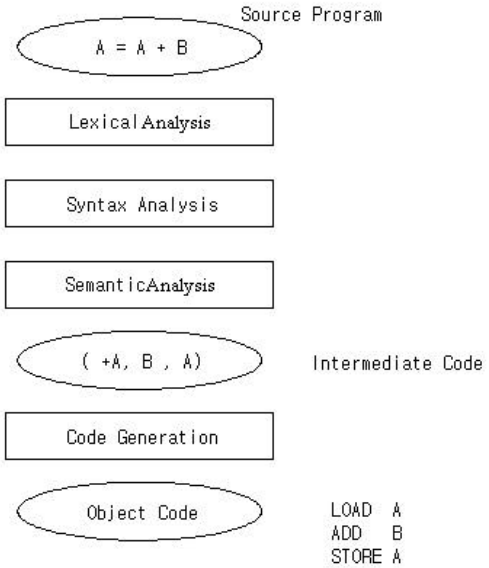
(2) Interpreter



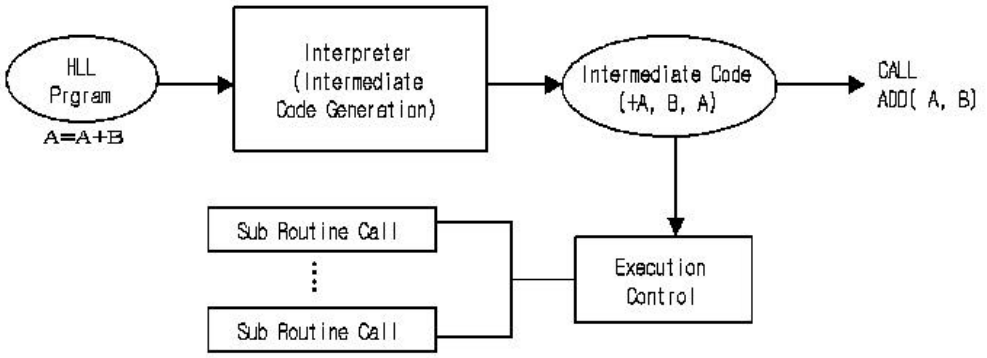
p94 그림 4.5

Translator	Interpreter
o Source PGM (변환)⟶ Object PGM o 반복문 sub program 한번 번역 o 한줄의 source PGM ⟶ 여러개의 Machine Code (큰 기억장치) o 실행 시간 효율	o source PGM 실행 o 매번 번역 o source language 형태를 유지하므로 추가 기억장소 필요치 않으나 decoding 시간이 김 o 사용자 flexibility

Compiler



Interpreter



ㅇCompiler Language

o Interpreter Language