(복습)

JIT 는 compilation 이라고 볼 수도 있고 애매하다.

- 한나 IDE
  ㄴ 상용 터미널 방식.

* : Programing 언어는 formal 하게 표현한 것이 ALGOL / ALGOL 60

* Sentence = statement. (lexeme으로 구성)
  ↑ 문장을 쓰는 Rule이 존재

또한 lexeme는 token으로 구성.

* Context Free ⟷ Context Sensitive

Context Free:
- 단어의미가 1가지
- 단어가 문맥으로 부터 자유롭다.
- 4level 中 가장 높은 level. (Chomsky)
- ↑ 프로그래밍 언어.

Context Sensitive:
- ↑ 단어. 문장이 Context 에 영향을 받는다. (영어, 한국어 ...)
- 단어의 뜻이 2가지 이상이다.
- ex) 배에서 배를 먹는다 → translate가 어려워진다.

* BNF : Backus. Naur Form. (meta language)
  ㄴ language를 표현하기 위한 language.

(→ : Rule
< > : Non-terminal (lexeme)
| : or. ← Rule 이 2개 이상 있을 때 한줄에 표현하기 위해.)

↑ Syntax analysis.

Start Symbol 을 terminal 만 남을 때 까지. derivation 하는것.

(non-terminal이 2개면 왼쪽부터 처리한다. = left most derivation

재귀 <expr> → <expr> + <term> | <term>. ← termination condition.

termination condition : 재귀를 끝내는 조건.

* ③ <expr> → <expr> + <term> | <term>

parse tree 깊이

왼쪽으로 확장

⇒ 좌결합 연산자

<expr> → <term> + <expr> | <term>.

오른쪽으로 확장

⇒ 우결합 연산자

* 좌결합성과 연산자는 왼쪽에서 recursion 한다.

2 ** (3 ** 4) ← exponential operator (우결합)

<t> → <t> (<k>) | <k>.

** 결합 : expression.

a = b + c ; // 결과 값을 계산

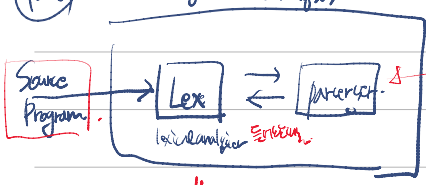
if ( ) // boolean (T/F) 결과

* ① 수학의 2가지 방향 (program semantic)을 반영하도록 Rule을 만들어야 한다!

* ② * Rule의 depth이 연산자 우선순위가 된다.

p.46

Syntax analysis.



Source program 이 들어오면 parsing 하는 곳.

(Source program 이 들어오면 parsing 한다.)

Intermediate Code

Code optimization 하는 곳.