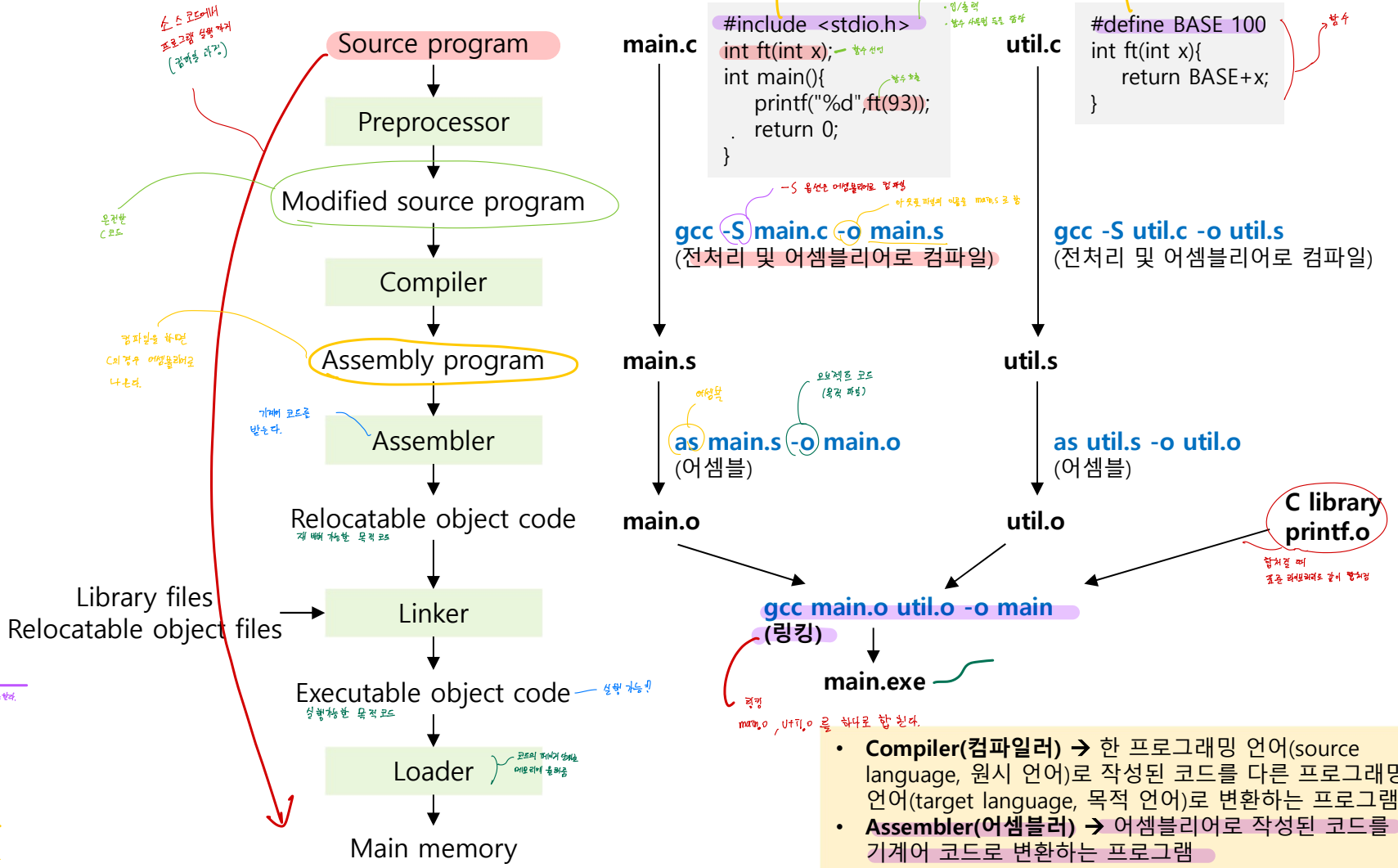
개요

개요

목차

- C언어 소스코드에서 실행가능파일까지
- 소스코드에서 실행가능파일까지
  - 어휘분석, 구문분석, 의미분석, 중간표현생성, 최적화, 코드생성
- 컴파일러
- 해석기

C언어 소스코드에서 실행가능파일까지



- **Compiler(컴파일러)** → 한 프로그래밍 언어(source language, 원시 언어)로 작성된 코드를 다른 프로그래밍 언어(target language, 목적 언어)로 변환하는 프로그램
- **Assembler(어셈블러)** → 어셈블리어로 작성된 코드를 기계어 코드로 변환하는 프로그램

소스코드에서 실행가능파일까지

토큰(token) 구성 규칙(정규표현식)

MAIN	main
DEF	def
ID	[a-z]
NUM	[0-9]+

프로그래밍언어 문법(grammar)

$Program \rightarrow DEF\ MAIN\ (\ )\ \{\ Stmts\ \}$

$Stmts \rightarrow Stmt\ Stmts \mid \epsilon$

$Stmt \rightarrow ID = Expr\ ; \mid PRINT\ Expr\ ; \mid WHILE\ (\ Expr\ )\ Stmt \mid \{\ Stmts\ \}$

$Expr \rightarrow Expr + Term \mid Term$

$Term \rightarrow Term * Factor \mid Factor$

$Factor \rightarrow (\ Expr\ ) \mid NUM \mid ID$

어셈블리 코드(x86)

```
section .text
    global _start
    extern _printf
_start:
    mov eax, 9
    mov [t1], eax
    mov eax, [t1]
    mov [n], eax
L1:
    mov eax, [n]
    mov [t2], eax
    mov eax, [t2]
    cmp eax,0
    je L2
    mov eax, [n]
    mov [t3], eax
    mov eax, [t3]
    push eax
    push formatStr
    call _printf
    add esp, 8
    mov eax, [n]
    mov [t4], eax
    mov eax, 1
    mov [t5], eax
    mov eax, [t4]
    sub eax, [t5]
    mov [t6], eax
    mov eax, [t6]
    mov [n], eax
    jmp L1
L2:
    ret

section .data
formatStr db "%d",0
n dd 0
t1 dd 0
t2 dd 0
t3 dd 0
t4 dd 0
t5 dd 0
t6 dd 0
```

flex

bison

문자 나열

토큰 나열

추상 구문 트리

AST

Lexical analysis

Syntax analysis

Semantic analysis

Intermediate code generation

Code optimzation

Code generation

Assembler (예: nasm)

Linker

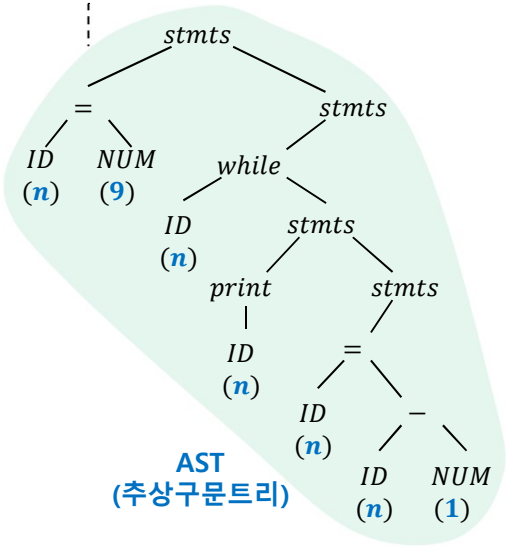
실행가능파일 (예: pgm.exe)

소스코드

```
def main() {
    n=9;
    while(n){
        print n;
        n=n-1;
    }
}
```

```
( DEF, def )
( MAIN, main )
( (, ( )
( ), ) )
( {, { )
( ID, n )
( =, = )
( NUM, 9 )
( ;, ; )
( WHILE, while )
( (, ( )
( ID, n )
( ), ) )
( {, { )
( ID, print )
( ID, n )
( ;, ; )
( ID, n )
( =, = )
( ID, n )
( -, - )
( NUM, 1 )
( ;, ; )
( }, } )
( }, } )
```

- 식별자의 사용과 선언(정의)를 연결
- 연산자의 피연산자 타입이 적절한가
- 연산자 +는 정수 덧셈? 실수 덧셈?



AST (추상구문트리)

중간표현(intermediate representation) TAC (three-address code)

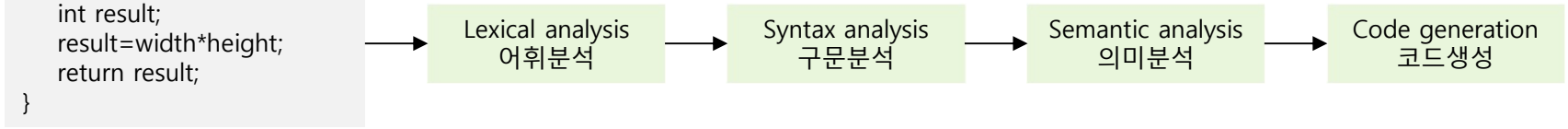
```
t1 = 9
n = t1
L1:
t2 = n
iffalse t2 goto L2
t3 = n
param t3
call print
t4 = n
t5 = 1
t6 = t4 - t5
n = t6
goto L1
L2:
```

어휘분석 에서 하는 일
· 단어들을 분석하는 일

어휘분석, 구문분석, 의미분석

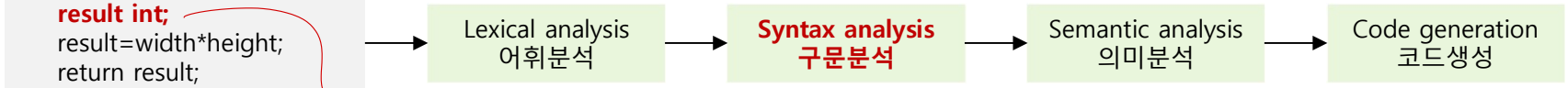
정상적인 C코드

```
int area(int width, int height){
    int result;
    result=width*height;
    return result;
}
```



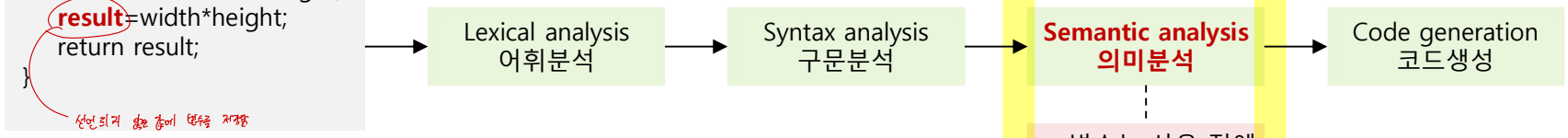
어휘분석 까지는 통과
그러나 구문분석에서 오류가 난다.

```
int area(int width, int height){
    result int;
    result=width*height;
    return result;
}
```



이것이 자료형 변수의 정의를 따르지 않은 변수
그래서 오류 발생!!

```
int area(int width, int height){
    result=width*height;
    return result;
}
```



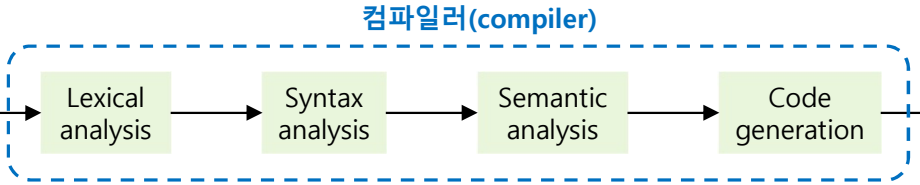
- 변수는 사용 전에 선언되어야 함

선언되지 않은 곳에 변수를 저장함
그래서 의미 분석 단계에서 오류가 난다.

컴파일러(compiler)

소스코드

```
def main() {
    n=9;
    while(n){
        print n;
        n=n-1;
    }
}
```



어셈블리 코드(x86)

```
section .text
      global _start
      extern _printf
_start:
      mov eax, 9
      mov [t1], eax
      mov eax, [t1]
      mov [n], eax
L1:
      mov eax, [n]
      mov [t2], eax
      mov eax, [t2]
      cmp eax,0
      je L2
      mov eax, [n]
      mov [t3], eax
      mov eax, [t3]
      push eax
      push formatStr
      call _printf
      add esp, 8
      mov eax, [n]
      mov [t4], eax
      mov eax, 1
      mov [t5], eax
      mov eax, [t4]
      sub eax, [t5]
      mov [t6], eax
      mov eax, [t6]
      mov [n], eax
      jmp L1
L2:
      ret

section .data
formatStr db "%d",0
n dd 0
t1 dd 0
t2 dd 0
t3 dd 0
t4 dd 0
t5 dd 0
t6 dd 0
```

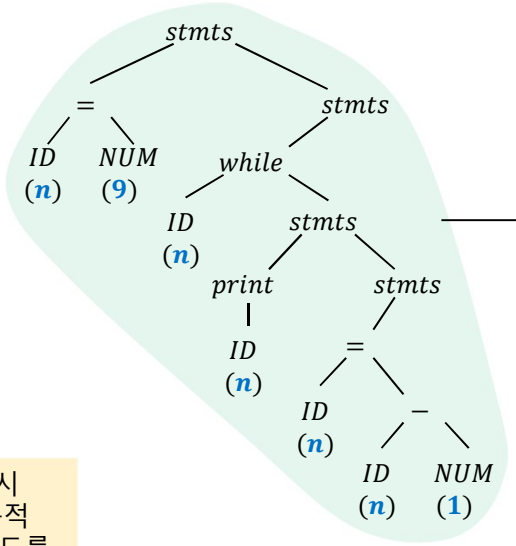
- **Compiler(컴파일러)** → 한 프로그래밍 언어(source language, 원시 언어)로 작성된 코드를 다른 프로그래밍 언어(target language, 목적 언어)로 변환하는 프로그램. 고수준 프로그래밍 언어로 작성된 코드를 저수준 프로그래밍 언어로 번역하는 프로그램
- **Interpreter(해석기)** → 한 프로그래밍 언어로 작성된 코드를 (기계어 코드로 변환하지 않고) 직접 실행하는 프로그램

해석기(interpreter)

소스코드

```
def main() {
    n=9;
    while(n){
        print n;
        n=n-1;
    }
}
```

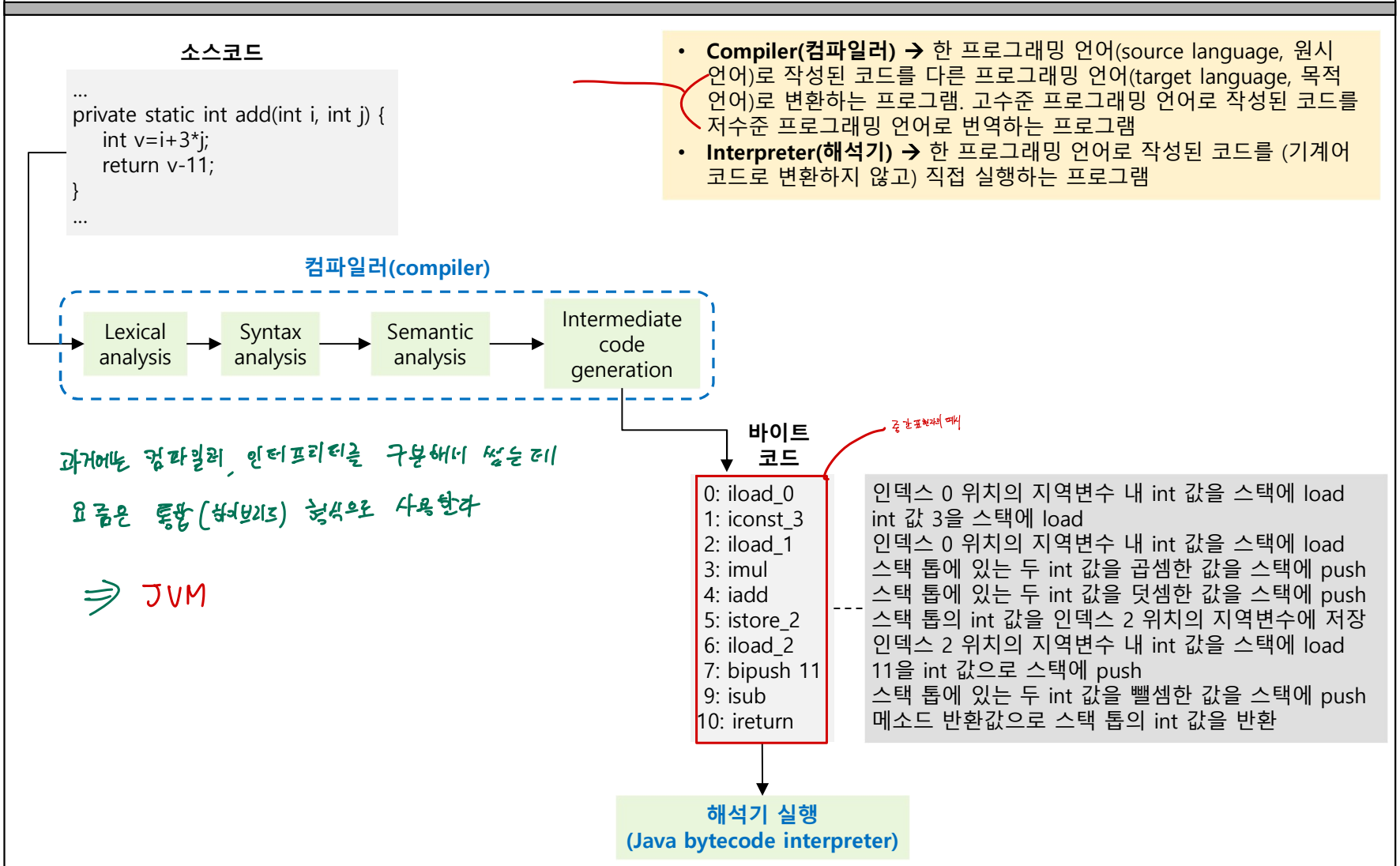
Lexical analysis → Syntax analysis → Semantic analysis → **AST (추상구문트리)**



→ **해석기 실행 (AST interpreter)**

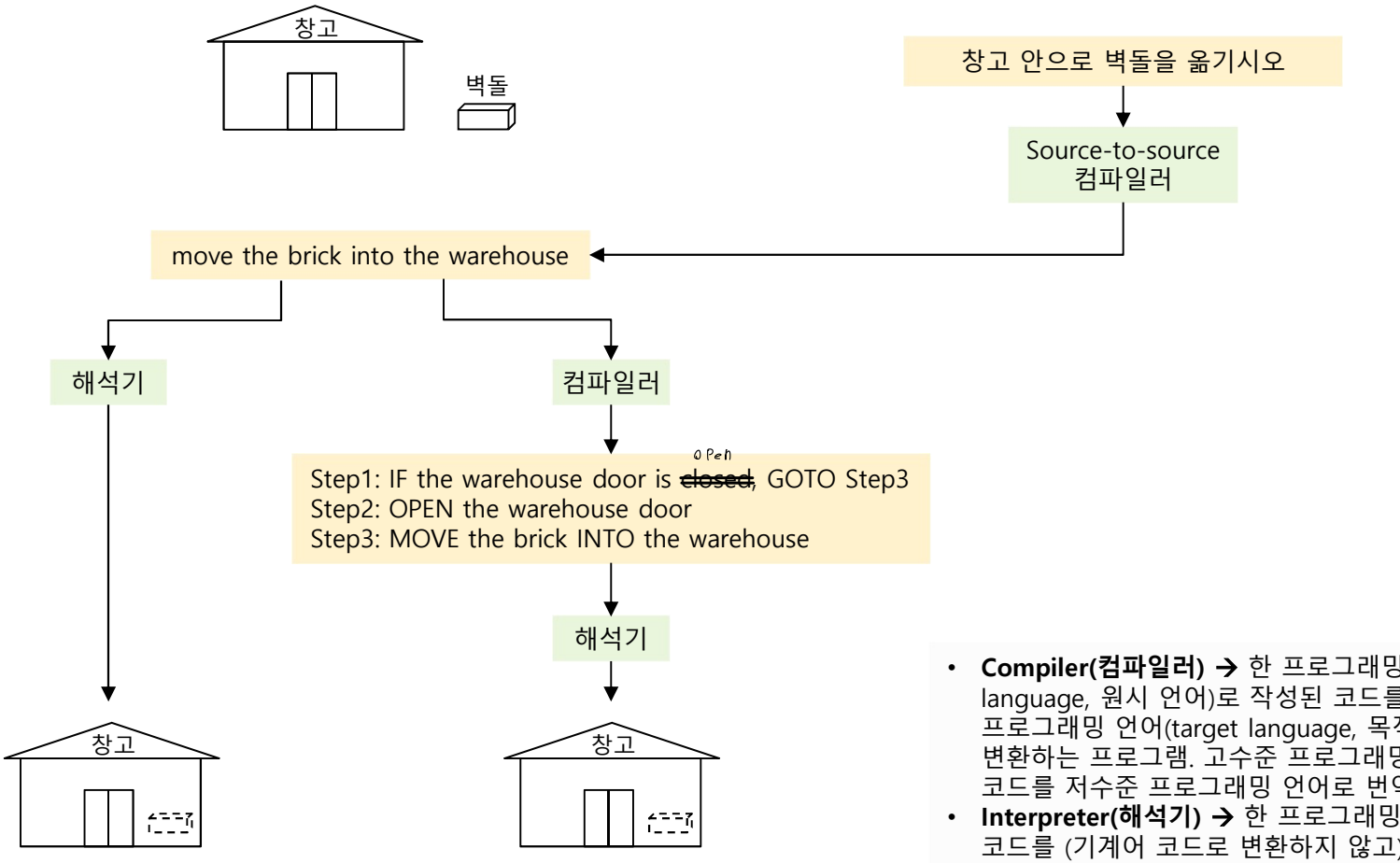
- **Compiler(컴파일러)** → 한 프로그래밍 언어(source language, 원시 언어)로 작성된 코드를 다른 프로그래밍 언어(target language, 목적 언어)로 변환하는 프로그램. 고수준 프로그래밍 언어로 작성된 코드를 저수준 프로그래밍 언어로 번역하는 프로그램
- **Interpreter(해석기)** → 한 프로그래밍 언어로 작성된 코드를 (기계어 코드로 변환하지 않고) 직접 실행하는 프로그램

Compiler, interpreter, hybrid



- **Compiler(컴파일러) →** 한 프로그래밍 언어(source language, 원시 언어)로 작성된 코드를 다른 프로그래밍 언어(target language, 목적 언어)로 변환하는 프로그램. 고수준 프로그래밍 언어로 작성된 코드를 저수준 프로그래밍 언어로 번역하는 프로그램
- **Interpreter(해석기) →** 한 프로그래밍 언어로 작성된 코드를 (기계어 코드로 변환하지 않고) 직접 실행하는 프로그램

비유 1



- **Compiler(컴파일러)** → 한 프로그래밍 언어(source language, 원시 언어)로 작성된 코드를 다른 프로그래밍 언어(target language, 목적 언어)로 변환하는 프로그램. 고수준 프로그래밍 언어로 작성된 코드를 저수준 프로그래밍 언어로 번역하는 프로그램
- **Interpreter(해석기)** → 한 프로그래밍 언어로 작성된 코드를 (기계어 코드로 변환하지 않고) 직접 실행하는 프로그램

비유 2

move the brick into the warehouse

↓

어휘분석

↓

구문분석

↓

의미분석

↓

코드생성

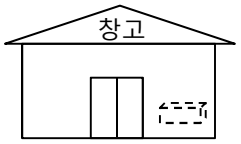
↓

Step1: IF the warehouse door is closed, GOTO Step3
Step2: OPEN the warehouse door
Step3: MOVE the brick INTO the warehouse

↓

해석기

↓



brick warehouse the into move the

↓

어휘분석

↓

구문분석 (오류)

move the warehouse into the brick

↓

어휘분석

↓

구문분석

↓

의미분석 (오류)

References

- 박두순. (2016). (내공 있는 프로그래머로 길러주는)컴파일러의 이해. 한빛아카데미.
- 창병모. (2021). 프로그래밍 언어론 : 원리와 실제. 인피니티북스.
- Aho, A., Lam, M., Sethi, R., Ullman, J. (2006). Compilers: Principles, Techniques, and Tools (2nd Ed.). Addison Wesley.
- Nystrom, R. Crafting interpreter. https://craftinginterpreters.com/
- Sebesta, R. (2012). Concepts of Programming Languages (10th. ed.). Pearson.
- Thain, D. (2023). Introduction to Compilers and Language Design.
- Paxson, V. (1995). Flex, version 2.5.
- Donnelly, C., Stallman, R. (2008). Bison.
- LexAndYacc.pdf (epaperpress.com)
- Tom Niemann, LEX & YACC. http://epaperpress.com/lexandyacc

References

- 박두순. (2016). (내공 있는 프로그래머로 길러주는)컴파일러의 이해. 한빛아카데미.
- 창병모. (2021). 프로그래밍 언어론 : 원리와 실제. 인피니티북스.
- Aho, A., Lam, M., Sethi, R., Ullman, J. (2006). Compilers: Principles, Techniques, and Tools (2nd Ed.). Addison Wesley.
- Nystrom, R. Crafting interpreter. https://craftinginterpreters.com/
- Sebesta, R. (2012). Concepts of Programming Languages (10th. ed.). Pearson.
- Thain, D. (2023). Introduction to Compilers and Language Design.
- Paxson, V. (1995). Flex, version 2.5.
- Donnelly, C., Stallman, R. (2008). Bison.
- LexAndYacc.pdf (epaperpress.com)
- Tom Niemann, LEX & YACC. http://epaperpress.com/lexandyacc