MiniC 컴파일러 프로젝트 설명서

1. 프로젝트 개요

본 프로젝트는 MiniC 형태의 간단한 C-like 언어를 직접 설계하고,
Lexer(Flex) → Parser(Bison) → AST 생성 → x86-64 Assembly 코드 생성까지의
컴파일러 전체 파이프라인을 구현하는 것을 목표로 한다.

구현 방식은 다음과 같다:

- **어휘 분석(Lexer)**: Flex를 사용하여 토큰을 분류
- **구문 분석(Parser)**: Bison을 사용해 문법을 정의하고 AST 생성
- **AST(Abstract Syntax Tree)**: 언어의 구조를 표현
- **중간 표현 / 실행 방식**: x86-64 Linux SysV ABI 기반 어셈블리 코드 생성
- **실행**: GCC를 통해 어셈블리 → 바이너리로 빌드 후 실행

본 프로젝트는 학습 목적이며, 리눅스 환경에서만 실행 가능하도록 설계되었다.

2. 언어 설계 의도

MiniC 언어는 C와 유사한 구조를 가지지만 교육용으로 단순화되어 있다.

설계 목표는 다음과 같다:

- 직관적이고 배우기 쉬운 문법
- 스택 기반 함수 호출 구조 학습
- AST 및 컴파일러 구조 파악

- 실제 x86-64 어셈블리 코드 생성 경험

3. 문법 정의 (EBNF)

MiniC 언어의 전체 문법 정의는 다음과 같다:

```
program         = function_list ;

function_list   = { function } ;

function        = "int" identifier "(" param_list_opt ")" compound_stmt ;

param_list_opt  = | param_list ;
param_list      = "int" identifier { "," "int" identifier } ;

compound_stmt   = "{" stmt_list_opt "}" ;
stmt_list_opt   = | stmt_list ;
stmt_list       = stmt { stmt } ;

stmt            = vardecl ";"
                | assign ";"
                | "return" expr ";"
                | expr ";" ;

vardecl         = "int" identifier ;

assign          = identifier "=" expr ;

expr            = expr "+" expr
                | expr "-" expr
                | expr "*" expr
                | expr "/" expr
                | primary ;

primary         = number
                | identifier
                | identifier "(" arg_list_opt ")"
                | "(" expr ")" ;

arg_list_opt    = | arg_list ;
arg_list        = expr { "," expr } ;
```

4. 전체 구조(Compiler Pipeline)

```
MiniC Source
      ↓
   Lexer (Flex)
      ↓
 Tokens
      ↓
   Parser (Bison)
      ↓
 AST(Abstract Syntax Tree)
      ↓
 Code Generator (x86-64)
      ↓
 out.s (Assembly)
      ↓
gcc -no-pie out.s -o prog
      ↓
 Executable Program
```

Lexer(Flex)

- 공백/개행 무시
- 키워드: int, return
- 식별자, 숫자, 연산자, 구분자 처리
- 에러 문자는 위치와 함께 출력

Parser(Bison)

- EBNF 기반 문법 정의
- AST 생성
- 잘못된 문법에 대한 오류 메시지 출력

AST

구조 예시:

```
Expr
 ├── EXPR_INT
 ├── EXPR_VAR
 ├── EXPR_BINOP
 └── EXPR_CALL

Stmt
 ├── STMT_VARDECL
 ├── STMT_ASSIGN
```

```
  ├── STMT_RETURN
  └── STMT_EXPR

Function
  ├── name
  ├── params
  └── stmt_list
```

Code Generation (x86-64 SysV ABI)

- 함수 호출 규약: rdi, rsi, rdx 순으로 인자 전달
- 스택 프레임:
  - pushq %rbp
  - movq %rsp, %rbp
  - local variable allocation
- 연산 결과는 rax 사용
- return 시 rax가 반환값

예시:

```
movq -8(%rbp), %rax
pushq %rax
movq -16(%rbp), %rax
popq %rcx
addq %rcx, %rax
```

5. 테스트 프로그램 및 목적

총 10개 이상의 MiniC 테스트 프로그램을 작성하였다.

파일명	목적	기대 결과
add2.minic	기본 함수 호출 / 덧셈	7
add3.minic	3개 이상 인자 처리	6
local_assign.minic	지역 변수 대입 / 반환	5
math1.minic	사칙연산 우선순위	7
math2.minic	괄호 / 나눗셈	2
mul_div.minic	곱셈 / 나눗셈	10
nest.minic	중첩 함수	6

파일명	목적	기대 결과
params.minic	6개 인자 사용	21
return.minic	단순한 상수 변환	42
var.minic	매개변수 / 지역변수	11

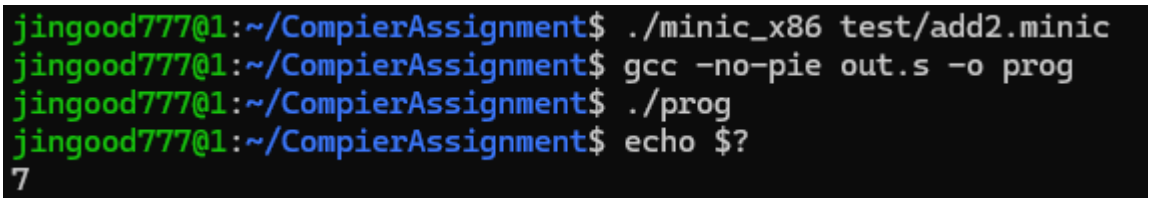
6. 실행 결과 스크린샷

실행 예시:

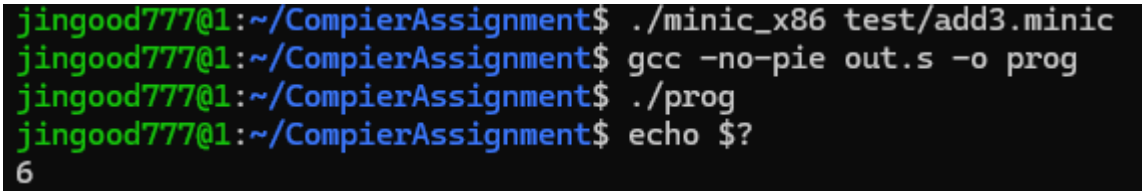
```
./minic_x86 test/add2.minic
gcc -no-pie out.s -o prog
./prog
echo $?
```

실행 결과:

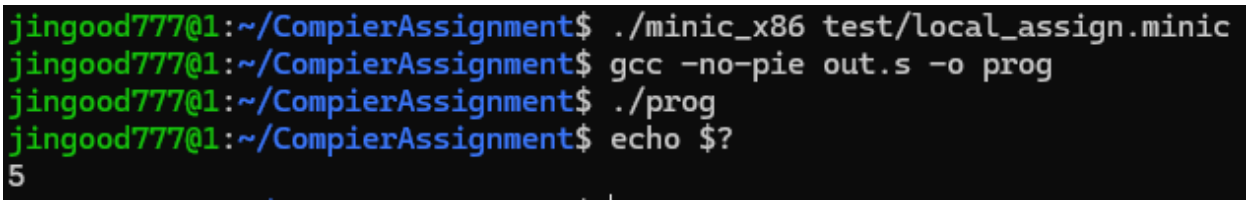
- add2.minic



- add3.minic



- local_assign.minic



- math1.minic

```
jingood777@1:~/CompierAssignment$ ./minic_x86 test/math1.minic
 -no-pie out.s -o prog
./prog
echo $?jingood777@1:~/CompierAssignment$ gcc -no-pie out.s -o prog
jingood777@1:~/CompierAssignment$ ./prog
jingood777@1:~/CompierAssignment$ echo $?
7
```

- math2.minic

```
jingood777@1:~/CompierAssignment$ ./minic_x86 test/math2.minic
 -no-pie out.s -o prog
./prog
echo $?jingood777@1:~/CompierAssignment$ gcc -no-pie out.s -o prog
jingood777@1:~/CompierAssignment$ ./prog
jingood777@1:~/CompierAssignment$ echo $?
2
```

- mul_div.minic

```
jingood777@1:~/CompierAssignment$ ./minic_x86 test/mul_div.minic
cc -no-pie out.s -o prog
./prog
echo $?jingood777@1:~/CompierAssignment$ gcc -no-pie out.s -o prog
jingood777@1:~/CompierAssignment$ ./prog
jingood777@1:~/CompierAssignment$ echo $?
10
```

- nest.minic

```
jingood777@1:~/CompierAssignment$ ./minic_x86 test/nest.minic
-no-pie out.s -o prog
./prog
echo $?jingood777@1:~/CompierAssignment$ gcc -no-pie out.s -o prog
jingood777@1:~/CompierAssignment$ ./prog
jingood777@1:~/CompierAssignment$ echo $?
6
```

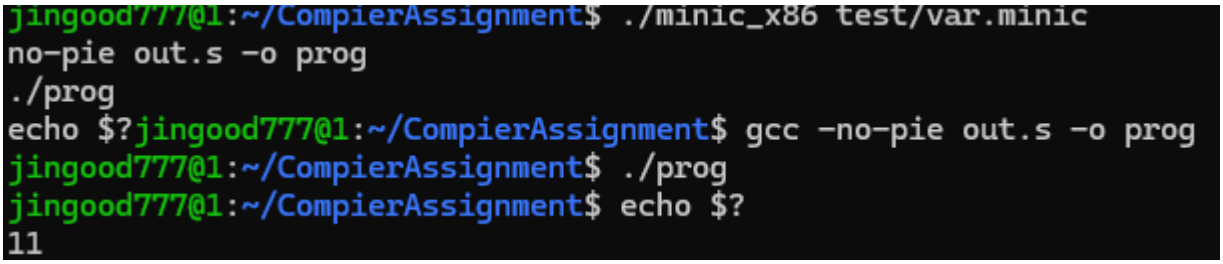
- params.minic

```
jingood777@1:~/CompierAssignment$ ./minic_x86 test/params.minic
c -no-pie out.s -o prog
./prog
echo $?jingood777@1:~/CompierAssignment$ gcc -no-pie out.s -o prog
jingood777@1:~/CompierAssignment$ ./prog
jingood777@1:~/CompierAssignment$ echo $?
21
```

- return.minic

```
jingood777@1:~/CompierAssignment$ ./minic_x86 test/return.minic
c -no-pie out.s -o prog
./prog
echo $?jingood777@1:~/CompierAssignment$ gcc -no-pie out.s -o prog
jingood777@1:~/CompierAssignment$ ./prog
jingood777@1:~/CompierAssignment$ echo $?
42
```

- var.minic



7. 구현된 기능 / 미구현 기능

구현됨

- 변수 선언 및 대입
- 산술 연산 (+ - * /)
- 함수 정의 및 호출
- return 문
- AST 기반 구조
- x86-64 어셈블리 코드 생성
- 기본 오류 메시지 출력

미구현

- if / while / for 문
- 비교 연산자 (< > == != 등)
- 논리 연산자 (&& ||)
- 다중 파일 지원
- 타입 시스템