# 입력 언어의 특성

입력 언어가 다음의 특성을 가진다고 간주하고 컴파일러를 제작하였다.

* 정적 유효 범위 (Static Scope)
* 정적 타입 언어 (Static Type Language)

# 문법 수정

문법을 일부 수정하였다.

## 결과는 같으나 LL Parser에 적용하기 위해 추가 및 수정된 문법들

* (수정) decls -> decls decl |
* decls -> decl decls |
* (수정) words -> words “,” word
* words -> word words’
* (추가) words’ -> “,” word words’ |
* (변경) slist -> slist stat | stat
* slist -> stat slist |
* (변경) stat -> IF cond THEN block ELSE block

| WHILE cond block

| word “=” expr “;”

| RETURN expr “;”

|

* stat -> IF cond THEN block ELSE block

| WHILE cond block

| word “=” expr “;”

| RETURN expr “;”

* (변경) cond -> expr “>” expr | expr “==” expr
* cond -> expr expr’
* (추가) expr’ -> “==” expr | “>” expr
* (변경) expr -> term | term “+” term
* expr -> term term’
* (추가) term’ -> “+” term |
* (변경) term -> fact | fact “\*” fact
* term -> fact fact’
* (추가) fact’ -> “\*” fact |

## 문법 수정으로 모호함을 제거할 수 없어서 수정된 문법 (결과가 달라짐)

block -> “{“ decls slist “}”

해당 문법에서 input으로 word를 받았을 때, decl에서 vtype이 없는 변수 정의인지,

stat에서 word “=” expr “;” 형태의 문법인지 구분할 수 없어서 다음의 문법을 수정하였다.

* (변경) vtype -> int | char |
* vtype -> int | char

따라서 데이터 타입이 명시되지 않은 변수 선언은 불가능하다.

# Instruction set 수정

메모리로부터 값을 가져올 때, word단위로 가져올 것인지(int형일 때) byte단위로 가져올 것인지(char형일 때) 구분이 필요하여 다음의 instruction을 추가하였다.

|  |  |
| --- | --- |
| LDB Reg#1, addr | addr 주소에서 byte 데이터를 Reg#1에 불러옵니다 |
| STB Reg#1, addr | Reg#1의 byte 데이터를 addr 주소에 저장합니다. |

# Scanner

# Parser

Parser는 LL Parser를 만들어 사용하였다.

# 에러 처리

가능한 많은 에러를 처리할 수 있도록 하였다.

## 인식할 수 없는 토큰

문법에 정의되지 않은 문자열 패턴이 있을 때, 컴파일 에러를 발생시키고 해당 문자열의 위치를 표시한다.

## 중괄호 불일치

좌측과 우측 중괄호 개수가 일치하지 않을 때 컴파일 에러를 발생시키고 해당 에러 내용을 출력한다.

## 세미 콜론 없음

세미 콜론이 있어야할 위치에 없는 경우 컴파일 에러를 발생시키고 세미 콜론이 있어야 할 위치를 표시한다.

## 중복 변수 선언

같은 블록 안에 같은 식별자의 변수가 선언된 경우 컴파일 에러를 발생시키고 중복된 변수가 선언된 위치를 표시한다.

## 선언되지 않은 변수 사용

static scope를 따를 접근할 수 있는 변수가 없으면 변수가 선언되지 않았다고 간주하고 컴파일 에러를 발생시키며 선언되지 않은 변수가 사용된 위치를 표시한다.