

CONFIDENTIALITY/SECURITY WARNING

이 문서에 포함되어 있는 정보는 국민대학교 전자정보통신대학 컴퓨터공학부 컴파일러 과목 수강
 학생 및 담당교수의 자산입니다. 저작권자의 서면 허락없이 사용되거나, 재가공될 수 없습니다.

컴파일러 과목 설계
 프로젝트 최종보고서

프로젝트명	<i>rusper</i>
학 번	
성 명	
담당교수	
제출일	

SW 명칭	RUSPER VERSION 1.0
Date	


 국민대학교 컴퓨터공학부 컴파일러 설계과제	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

과제 명칭 변경 내역

제안서 과제명	rusper(RUby Scheme PERI) : 테스트케이스 작성에 특화된 언어
중간보고서 과제명	상 동
최종보고서 과제명	상 동


과제 명칭 변경 사유

중간보고서 과제명 변경 사유	해당 사항 없음
최종보고서 과제명 변경 사유	해당 사항 없음

 국민대학교 컴퓨터공학부 컴파일러 설계과제	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

목 차

1	서론	4
2	테스트케이스 작성에 특화된 언어 설계	5
2.1	스크립트 작성에 특화된 언어의 특징	5
2.1.1	Ruby의 출력 형식	5
2.1.2	Scheme의 연산자	5
2.1.3	Perl의 \$ 연산자	5
2.2	테스트케이스 작성에 특화된 언어 rusper 설계	5
2.3	rusper의 구문 구조 (EBNF로 기술)	6
3	rusper의 어휘분석	8
3.1	rusper에 대한 Lex 입력	8
3.2	어휘분석기 실험 결과	9
4	rusper의 구문분석	18
4.1	rusper에 대한 Yacc 입력	18
4.2	구문분석기 실험 결과	30
5	목적코드 생성	37
5.1	목적코드 생성기 구조 및 설계	37
5.2	목적코드 생성기 구현	37
5.3	목적코드 생성기 실험 결과	41
6	실험 및 평가	45
6.1	실험 데이터	45
6.2	실행 결과	47
7	결론	54
	부록 1. Lex 입력 파일	55
	부록 2. Yacc 입력 파일(파스 트리, 목적코드 생성 소스 포함)	57


 국민대학교 컴퓨터공학부 컴파일러 설계과제	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

1 서론

컴퓨터에게 무엇인가를 대신하게 하는 제일 간단한 방법은 프로그래밍 언어를 이용해서 컴퓨터에게 명령을 내리는 것이다. 컴퓨터는 간단하지만 반복적인, 소위 말하는 실수하기 쉬운 일을 하는데 한 치의 오차도 없이 일을 정확하고 빠르게 수행한다. 이런 일은 컴퓨터가 수행하기에 매우 알맞다. 하지만 이러한 일을 하도록 명령하기 위해 프로그래밍을 배우는 과정은 간단하지 않고 반복적이지도 않다. 때로는 더 어렵기도 하다. 만약에 간단한 문제를 풀기 위한 간단한 프로그래밍 언어가 존재한다면?

또한 프로그래밍을 하는 경우, 실제 문제를 해결하기 위한 프로그램을 작성하는 경우도 많지만, 해당 프로그램이 실제로 해당 문제를 정확히 해결하는지 테스트 하는 프로그램을 작성하는 경우도 많이 있다. 특히 테스트를 위한 테스트케이스를 사람이 직접 작성하기도 하지만 보다 다양한 테스트를 하기 위해서 자동화된 테스트케이스를 작성하는 프로그램을 작성하기도 한다. 이때 일반적으로 테스트케이스는 단순한 입력 순서의 나열이거나 간단한 알고리즘으로 생성할 수 있다. 하지만 기존의 언어를 활용하는 경우 I/O 관련된 처리가 언어마다 다르고 복잡하다. 테스트케이스를 작성하는데 필요한 내용이나 알고리즘에 비해 I/O처리가 더 까다롭거나 복잡한 경우가 대부분이다. 따라서 이러한 I/O 관련된 처리를 보다 쉽게 하면서 원래 하려고 했던 간단한 테스트케이스를 작성하는데 집중할 수 있게 도와주는 언어를 설계하게 되었다.

rusper는 복잡한 I/O 관련 출력을 대괄호([,])를 통해 간단하게 추상화하고, 연산자는 전위 표현을 사용하여 우선순위에 의한 미묘한 문제를 피하고, 중첩된 표현식(expression)을 사용하여 여러 개의 연산자를 적용하는데도 문제가 없다.

 국민대학교 컴퓨터공학부 컴파일러 설계과제	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

2 테스트케이스 작성에 특화된 언어 설계

2.1 스크립트 작성에 특화된 언어의 특징

2.1.1 Ruby의 출력 형식

Ruby에서는 STDOUT의 출력을 \$stdout 변수의 값을 변경함으로써 STDOUT에서 임의의 파일로 출력을 redirection을 할 수 있다. 또한 일반적인 String Literal에서 내부에 #{}를 이용하면 #{} 내부의 표현식(expression)을 평가(evaluate)하여 String Literal을 구성할 수 있다.

2.1.2 Scheme의 연산자

Scheme에서는 모든 연산자가 가장 왼쪽에 오는 전위표현식(prefix expression)을 사용한다. 또한 피연산자(operand)의 위치에 표현식(expression)이 올 수 있으므로 중첩하여 표현식을 표현할 수 있다.

2.1.3 Perl의 \$ 연산자

Perl에서 \$로 시작하는 변수는 특별한 의미를 가지고 문맥에 따라 다양하게 해석될 수 있다. 이러한 \$를 이용하면 보다 쉽게 다양한 표현식(expression)을 간결하게 작성할 수 있다.

2.2 테스트케이스 작성에 특화된 언어 rusper 설계

Ruby의 \$stdout 변수의 값을 변경하는 방법을 도입하여 OUTPUT 키워드를 이용하여 redirection을 할 수 있다. 또한 String Literal 내부에 기술하는 #{}를 도입하여 중괄호({ })를 이용하여 표현식(expression)의 평가된(evaluated) 값을 출력할 수 있다. 출력의 경우에는 대괄호([])로 감싼 내용은 STDOUT으로 출력하거나 OUTPUT 키워드를 이용하여 redirection을 한 경우에는 redirection한 파일로 출력된다. Scheme의 전위표현식(prefix expression)을 도입하여 모든 표현식은 연산자가 가장 왼쪽에 오고 피연산자는 소괄호(())로 둘러싼다. 연산자는 기본적으로 1개에서 2개까지 가능하다. 만약에 여러 개의 연산자를 중첩하는 경우에는 피연산자에 표현식을 중첩하는 방식으로 기술하면 모든 표현식을 기술할 수 있다. 반복문의 경우에는 FOR, DO, LOOP 이렇게 3가지 방식을 지원한다. FOR의 경우에는 매개변수가 2개의 정수이고, DO는 1개의 정수, LOOP는 매개변수가 없다. FOR는 구간을, DO는 0부터 주어진 정수 - 1 까지(즉 주어진 정수만큼 반복), LOOP는 무한반복을 나타낸다. 반복문의 내부에서 BREAK와 \$를 사용할 수 있다. BREAK의 경우 무조건적으로



반복문을 빠져나오며, Perl에서 도입한 \$의 경우(단, Perl의 용법과는 조금 다르다) 현재 반복횟수를 나타낸다. IF, FOR, DO, LOOP는 복합문을 가져야 하고, 해당 복합문은 콜론 두개 (::)와 콜론(:)으로 둘러싼다.

2.3 rusper의 구문 구조 (EBNF로 기술)


```

<program> ::= <file>
<file> ::= <statement_list>
<statement_list> ::= <statement>
| <statement> <statement_list>
<compound_statement> ::= <statement>
| <statement> <compound_statement>
<statement> ::= <declaration_statement>
| <expression_statement>
| <selection_statement>
| <break_statement>
| <for_statement>
| <do_statement>
| <print_statement>
| <output_statement>
| <comment_statement>
<declaration_statement> ::= "VAR" <identifier>
<expression_statement> ::= <expression>
<selection_statement> ::= "IF" "(" <expression> ")" ":"
<compound_statement> ::= ":"
<break_statement> ::= "BREAK"
<for_statement> ::= "FOR" "(" <integer> "," <integer> ")"
| ":" <compound_statement> ":"
<do_statement> ::= "DO" "(" <integer> ")" ":"
<compound_statement> ::= ":"
<print_statement> ::= "[" <print_expression_list> "]"
<print_expression_list> ::= <print_expression>
| <print_expression> <print_expression_list>
<print_expression> ::= <string_literal>
| "{" expression "}"
<output_statement> ::= "OUTPUT" "(" <string_literal> ")"
<comment_statement> ::= ";" { <character_constant> }1+
<expression> ::= <assignment_expression>
| <arithmetic_expression>
| <equality_expression>
| <relational_expression>
| <logical_expression>
| <random_expression>
| <iterator_expression>
| <integer>
| <identifier>
<assignment_expression> ::= "=" "(" <lvalue> "," <expression> ")"
<arithmetic_expression> ::= "+" "(" <expression> "," <expression> ")"
| "-" "(" <expression> "," <expression> ")"

```



```
<equality_expression> ::=
    | "*" "(" <expression> "," <expression> ")"
    | "/" "(" <expression> "," <expression> ")"
    | "%" "(" <expression> "," <expression> ")"
    | "==" "(" <expression> "," <expression> ")"
    | "!=" "(" <expression> "," <expression> ")"
<relational_expression> ::=
    | "<" "(" <expression> ")"
    | "&&" "(" <expression> "," <expression> ")"
    | "||" "(" <expression> "," <expression> ")"
<logical_expression> ::=
    | "<" "(" <expression> "," <expression> ")"
    | ">" "(" <expression> "," <expression> ")"
    | "<=" "(" <expression> "," <expression> ")"
    | ">=" "(" <expression> "," <expression> ")"
<random_expression> ::=
    | "RANDOM" "(" <integer_constant> ","
<integer_constant> ")"
<iterator_expression> ::=
    | "$"
<string_literal> ::=
    | """ { <character_constant> }1+ """
<character_constant> ::=
    | "a" | "b" | "c" | "d" | "e" | "f" | "g"
    | "h" | "i" | "j" | "k" | "l"
    | "m" | "n" | "o" | "p" | "q" | "r" | "s" | "t" | "u"
    | "w" | "x" | "y" | "z"
    | "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I"
    | "J" | "K" | "L"
    | "M" | "N" | "O" | "P" | "Q" | "R" | "S" | "T" | "U" | "V" | "W" | "X" |
    | "Y" | "Z" | "_" | " "
<integer> ::=
    | <decimal_constant>
    | "+" <decimal_constant>
    | "-" <decimal_constant>
<decimal_constant> ::=
    | "0" | <nonzero_digit> <digit_sequence>
<nonzero_digit> ::=
    | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8"
    | "9"
<digit_sequence> ::=
    | { <digit> }1+
<digit> ::=
    | "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7"
    | "8" | "9"
<lvalue> ::=
    | <identifier>
<identifier> ::=
    | { <character_constant> }1+
```

 <div> 국민대학교 컴퓨터공학부 컴파일러 설계과제 </div>	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

3 rusper의 어휘분석

3.1 rusper에 대한 Lex 입력

```


rusper.l
letter [a-zA-Z_]
nonzero_digit [1-9]
digit [0-9]
%%
"VAR"      return T_VAR;
"IF"       return T_IF;
"("        return T_LPAREN;
")"        return T_RPAREN;
":"        return T_COLON;
"BREAK"    return T_BREAK;
"FOR"      return T_FOR;
", "       return T_SEPARATOR;
"DO"       return T_DO;
"LOOP"     return T_LOOP;
"["        return T_LBRACKET;
"]"        return T_RBRACKET;
"{"        return T_LCURLYBRACE;
"}"        return T_RCURLYBRACE;
"="        return T_ASSIGNMENT;
"+"        return T_PLUS;
"-"        return T_MINUS;
"*"        return T_MUL;
"/"        return T_DIV;
"%"        return T_REMAINDER;
"=="       return T_EQUAL;
"!="       return T_NOT_EQUAL;
"!"        return T_NOT;
"&&"       return T_AND;
"||"       return T_OR;
"<"        return T_LT;
">"        return T_GT;
"<="       return T_LT_EQUAL;
">="       return T_GT_EQUAL;
"RANDOM"    return T_RANDOM;
"$"        return T_ITERATOR;
"OUTPUT"   return T_OUTPUT;
";"        return T_COMMENT;
\".*\"     return T_STRING_LITERAL;

{letter}({letter}|{digit})* return T_IDENTIFIER;
[-+]?{digit}+              return T_INTEGER;

[ \t\n\r]      ;
.               return T_ERROR;


%%

```


 국민대학교 컴퓨터공학부 컴파일러 설계과제	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

3.2 어휘분석기 실험 결과


gcd.rus
Start of rusper
<293, ; 최대공약수 구하기>
<259, VAR>
<291, a>
<259, VAR>
<291, b>
<259, VAR>
<291, r>
<293, ; initialize a>
<273, =>
<261, (>
<291, a>
<267, ,>
<292, 64>
<262,)>
<293, ; initialize b>
<273, =>
<261, (>
<291, b>
<267, ,>
<292, 12>
<262,)>
<268, D0>
<261, (>
<292, -1>
<262,)>
<264, ::>
<273, =>
<261, (>
<291, r>

 국민대학교 컴퓨터공학부 컴파일러 설계과제	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

```

<267, ,>
<291, a>
<262, )>
<273, =>
<261, (>
<291, a>
<267, ,>
<291, b>
<262, )>
<273, =>
<261, (>
<291, b>
<267, ,>
<278, %>
<261, (>
<291, r>
<267, ,>
<291, b>
<262, )>
<262, )>
<260, IF>
<261, (>
<286, <=>
<261, (>
<291, b>
<267, ,>
<292, 0>
<262, )>
<262, )>
<264, ::>
<265, BREAK>
<263, :>
<263, :>
<293, ; print gcd>

```

 국민대학교 컴퓨터공학부 컴파일러 설계과제	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

```

<269, [>
<294, "gcd is ">
<271, {>
<291, a>
<272, }>
<270, ]>
<293, ; end of source>
End of rusper

```


```

prime.rus
Start of rusper

<293, ; 소수 판별 프로그램>

<259, VAR>
<291, prime>
<259, VAR>
<291, i>
<259, VAR>
<291, flag>
<290, OUTPUT>
<261, (>
<294, "prime_result.txt">
<262, )>
<293, ; initialize prime>
<273, =>
<261, (>
<291, prime>
<267, ,>
<292, 131>
<262, )>
<293, ; initialize flag>
<273, =>
<261, (>

```

 국민대학교 컴퓨터공학부 컴파일러 설계과제	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	


```

<291, flag>
<267, ,>
<292, 0>
<262, )>
<260, IF>
<261, (>
<282, &&>
<261, (>
<284, <>
<261, (>
<292, 2>
<267, ,>
<291, prime>
<262, )>
<267, ,>
<280, !=>
<261, (>
<292, 0>
<267, ,>
<278, %>
<261, (>
<291, prime>
<267, ,>
<292, 2>
<262, )>
<262, )>
<262, )>
<262, )>
<264, ::>
<293, ; initialize i>
<273, =>
<261, (>
<291, i>
<267, ,>

```

 국민대학교 컴퓨터공학부 컴파일러 설계과제	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

<292, 3>
 <262,)>
 <268, DO>
 <261, (>
 <292, -1>
 <262,)>
 <264, ::>
 <260, IF>
 <261, (>
 <286, <=>
 <261, (>
 <291, prime>
 <267, ,>
 <291, i>
 <262,)>
 <262,)>
 <264, ::>
 <273, =>
 <261, (>
 <291, flag>
 <267, ,>
 <292, 1>
 <262,)>
 <265, BREAK>
 <263, :>
 <260, IF>
 <261, (>
 <279, ==>
 <261, (>
 <292, 0>
 <267, ,>
 <278, %>
 <261, (>
 <291, prime>

 국민대학교 컴퓨터공학부 컴파일러 설계과제	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

<267, ,>
 <291, i>
 <262,)>
 <262,)>
 <262,)>
 <264, ::>
 <265, BREAK>
 <263, :>
 <273, =>
 <261, (>
 <291, i>
 <267, ,>
 <274, +>
 <261, (>
 <291, i>
 <267, ,>
 <292, 2>
 <262,)>
 <262,)>
 <263, :>
 <263, :>
 <260, IF>
 <261, (>
 <279, ==>
 <261, (>
 <292, 1>
 <267, ,>
 <291, flag>
 <262,)>
 <262,)>
 <264, ::>
 <269, [>
 <271, {>
 <291, prime>

	국민대학교		
	컴퓨터공학부		
	컴파일러 설계과제		
	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

```


<272, }>
<294, "is prime.">
<270, ]>
<263, :>
<260, IF>
<261, (>
<279, ==>
<261, (>
<292, 0>
<267, ,>
<291, flag>
<262, )>
<262, )>
<264, ::>
<269, [>
<271, {>
<291, prime>
<272, }>
<294, "is not prime.">
<270, ]>
<263, :>
<293, ; end of source>
End of rusper

```

```

random.rus
Start of rusper
<293, ; random list generator>
<268, DO>
<261, (>
<292, 10>
<262, )>
<264, ::>
<293, ; print current index>

```

 국민대학교 컴퓨터공학부 컴파일러 설계과제	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

```

<269, [>
<271, {>
<289, $>
<272, }>
<270, ]>
<293, ; generate random number 1 ~ 10>
<269, [>
<271, {>
<288, RANDOM>
<261, (>
<292, 1>
<267, ,>
<292, 10>
<262, )>
<272, }>
<270, ]>
<263, :>
<293, ; end of source>
End of rusper

```

```

sum.rus
Start of rusper

<293, ; 1 부터 10까지 합 구하기>

<259, VAR>
<291, sum>
<293, ; initialize sum>
<273, =>
<261, (>
<291, sum>
<267, ,>
<292, 0>
<262, )>

```


 국민대학교 컴퓨터공학부 컴파일러 설계과제	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

```

<293, ; repeat 10 times>
<268, DO>
<261, (>
<292, 10>
<262, )>
<264, ::>
<273, =>
<261, (>
<291, sum>
<267, ,>
<274, +>
<261, (>
<291, sum>
<267, ,>
<274, +>
<261, (>
<289, $>
<267, ,>
<292, 1>
<262, )>
<262, )>
<262, )>
<293, ; print sum>
<269, [>
<271, {>
<291, sum>
<272, }>
<270, ]>
<263, ::>
<293, ; end of source>
End of rusper

```

	국민대학교		
	컴퓨터공학부		
	컴파일러 설계과제		
	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

4 rusper의 구문분석

4.1 rusper에 대한 Yacc 입력

```

rusper.y

%union {
    int integer;
    char *str;
}

%token T_EOF
%token T_ERROR
%token T_VAR
%token T_IF
%token T_LPAREN
%token T_RPAREN
%token T_COLON
%token T_DCOLON
%token T_BREAK
%token T_FOR
%token T_SEPARATOR
%token T_DO
%token T_LBRACKET
%token T_RBRACKET
%token T_LCURLYBRACE
%token T_RCURLYBRACE
%token T_ASSIGNMENT
%token T_PLUS
%token T_MINUS
%token T_MUL
%token T_DIV
%token T_REMAINDER
%token T_EQUAL
%token T_NOT_EQUAL
%token T_NOT
%token T_AND
%token T_OR
%token T_LT
%token T_GT
%token T_LT_EQUAL
%token T_GT_EQUAL
%token T_RANDOM
%token T_ITERATOR
%token T_OUTPUT
%token T_IDENTIFIER
%token T_INTEGER
%token T_COMMENT
%token T_STRING_LITERAL

%%
program:      file
              ;

```

	국민대학교 컴퓨터공학부 컴파일러 설계과제	최종보고서	
		프로젝트 명	rusper
		성 명	
		Confidential Restricted	Version 1.0

```

file:      statement_list
          {
              AST_node *iter;
              int i;

              iter = AST_root = AST_stack[0];
              for(i = 1; i < sp; ++i) {
                  iter->sibling = AST_stack[i];
                  iter = iter->sibling;
              }
          }
          ;

statement_list:      statement
                    | statement statement_list
                    ;

compound_statement:  statement
                    {
                        AST_node *ptr;
                        AST_node *iter;
                        int stmt_sp;
                        int cmp_stmt_sp;

                        cmp_stmt_sp = stmt_sp = sp - cmp_stmt_cnt;

                        ptr = build_tree(COMPOUND_STATEMENT);
                        iter = ptr->child = AST_stack[stmt_sp];

                        ++stmt_sp;
                        while(stmt_sp < sp) {
                            iter->sibling = AST_stack[stmt_sp];
                            iter = iter->sibling;
                            ++stmt_sp;
                        }


                        AST_stack[cmp_stmt_sp] = ptr;
                        sp = cmp_stmt_sp + 1;
                    }
                    | statement compound_statement
                    ;

statement:      declaration_statement
              {
                  // build tree
                  AST_node *ptr;

                  ptr = build_tree(DECLARATION_STATEMENT);
                  ptr->child = AST_stack[--sp];

                  // push
                  AST_stack[sp++] = ptr;
              }

```

	국민대학교 컴퓨터공학부 컴파일러 설계과제	최종보고서	
		프로젝트 명	rusper
		성 명	
		Confidential Restricted	Version 1.0

```

| expression_statement
| selection_statement
{
    // build tree
    AST_node *ptr;

    ptr = build_tree(SELECTION_STATEMENT);
    ptr->child = AST_stack[--sp];

    // push
    AST_stack[sp++] = ptr;
}
| break_statement
{
    // build tree
    AST_node *ptr;

    ptr = build_tree(BREAK_STATEMENT);
    ptr->child = AST_stack[--sp];

    // push
    AST_stack[sp++] = ptr;
}
| for_statement
{
    // build tree
    AST_node *ptr;

    ptr = build_tree(FOR_STATEMENT);
    ptr->child = AST_stack[--sp];

    // push
    AST_stack[sp++] = ptr;
}
| do_statement
{
    // build tree
    AST_node *ptr;

    ptr = build_tree(DO_STATEMENT);
    ptr->child = AST_stack[--sp];

    // push
    AST_stack[sp++] = ptr;
}
| print_statement
{
    // build tree
    AST_node *ptr;

    ptr = build_tree(PRINT_STATEMENT);

```

	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

```

ptr->child = AST_stack[--sp];

// push
AST_stack[sp++] = ptr;

}
| output_statement
{
    // build tree
    AST_node *ptr;

    ptr = build_tree(OUTPUT_STATEMENT);
    ptr->child = AST_stack[--sp];

    // push
    AST_stack[sp++] = ptr;

}
| comment_statement
;

declaration_statement:    T_VAR identifier
{
    // build sibling
    AST_node *ptr;

    ptr = build_node(T_VAR, "VAR");
    ptr->sibling = AST_stack[--sp];

    AST_stack[sp++] = ptr;
}
;

expression_statement: expression
;

selection_statement: T_IF    T_LPAREN    expression    T_RPAREN    T_DCOLON
compound_statement T_COLON
{
    // build sibling
    AST_node *ptr;
    AST_node *param;

    ptr = build_node(T_IF, "IF");

    param = AST_stack[--sp];
    ptr->sibling = AST_stack[--sp];
    ptr->sibling->sibling = param;

    AST_stack[sp++] = ptr;
}
;

break_statement:    T_BREAK

```

	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

```

        {
            AST_stack[sp++] = build_node(T_BREAK, "BREAK");
        }
        ;

for_statement:      T_FOR  T_LPAREN  integer  T_SEPARATOR  integer  T_RPAREN
T_DCOLON compound_statement T_COLON
        {
            // build sibling
            AST_node *ptr;
            AST_node *param;
            AST_node *cmp_stmt;

            ptr = build_node(T_FOR, "FOR");

            cmp_stmt = AST_stack[--sp];
            param = AST_stack[--sp];
            ptr->sibling = AST_stack[--sp];
            ptr->sibling->sibling = param;
            ptr->sibling->sibling->sibling = cmp_stmt;

            AST_stack[sp++] = ptr;
        }
        ;

do_statement:      T_DO      T_LPAREN      integer      T_RPAREN      T_DCOLON
compound_statement T_COLON
        {
            // build sibling
            AST_node *ptr;
            AST_node *param;

            ptr = build_node(T_DO, "DO");

            param = AST_stack[--sp];
            ptr->sibling = AST_stack[--sp];
            ptr->sibling->sibling = param;

            AST_stack[sp++] = ptr;
        }
        ;

print_statement:   T_LBRACKET print_expression_list T_RBRACKET
        {
            // build sibling
            AST_node *ptr;

            ptr = build_node(T_LBRACKET, "PRINT");
            ptr->sibling = AST_stack[--sp];

            AST_stack[sp++] = ptr;
        }
        ;

```

	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

```

print_expression_list:    print_expression
                          | print_expression print_expression_list
                          {
                              AST_node *ptr;
                              AST_node *iter;
                              int stmt_sp;
                              int print_stmt_sp;

                              print_stmt_sp = stmt_sp = sp - print_stmt_cnt;

                              ptr = build_tree(PRINT_EXPR_LIST);
                              iter = ptr->child = AST_stack[stmt_sp];

                              ++stmt_sp;
                              while(stmt_sp < sp) {
                                  iter->sibling = AST_stack[stmt_sp];
                                  iter = iter->sibling;
                                  ++stmt_sp;
                              }

                              AST_stack[print_stmt_sp] = ptr;
                              sp = print_stmt_sp + 1;
                          }
                          ;

print_expression:        string_literal
                          | T_LCURLYBRACE expression T_RCURLYBRACE
                          ;

output_statement:        T_OUTPUT T_LPAREN string_literal T_RPAREN
                          {
                              // build sibling
                              AST_node *ptr;

                              ptr = build_node(T_OUTPUT, "OUTPUT");
                              ptr->sibling = AST_stack[--sp];

                              AST_stack[sp++] = ptr;
                          }
                          ;

comment_statement:       T_COMMENT
                          ;

expression:              assignment_expression
                          {
                              // build tree
                              AST_node *ptr;

                              ptr = build_tree(EXPRESSION);
                              ptr->child = AST_stack[--sp];

                              // push
                              AST_stack[sp++] = ptr;

```

	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

국민대학교
컴퓨터공학부
컴파일러 설계과제

```

    }
    | arithmetic_expression
    {
        // build tree
        AST_node *ptr;

        ptr = build_tree(EXPRESSION);
        ptr->child = AST_stack[--sp];

        // push
        AST_stack[sp++] = ptr;
    }
    | equality_expression
    {
        // build tree
        AST_node *ptr;

        ptr = build_tree(EXPRESSION);
        ptr->child = AST_stack[--sp];

        // push
        AST_stack[sp++] = ptr;
    }
    | relational_expression
    {
        // build tree
        AST_node *ptr;

        ptr = build_tree(EXPRESSION);
        ptr->child = AST_stack[--sp];

        // push
        AST_stack[sp++] = ptr;
    }
    | logical_expression
    {
        // build tree
        AST_node *ptr;

        ptr = build_tree(EXPRESSION);
        ptr->child = AST_stack[--sp];

        // push
        AST_stack[sp++] = ptr;
    }
    | random_expression
    {
        // build tree
        AST_node *ptr;

        ptr = build_tree(EXPRESSION);
        ptr->child = AST_stack[--sp];

        // push

```


	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

```

        AST_stack[sp++] = ptr;
    }
    | iterator_expression
    {
        // build tree
        AST_node *ptr;

        ptr = build_tree(EXPRESSION);
        ptr->child = AST_stack[--sp];

        // push
        AST_stack[sp++] = ptr;
    }
    | integer
    | identifier
    ;

assignment_expression:
expression T_LPAREN
    {
        // build sibling
        AST_node *ptr;
        AST_node *param;

        ptr = build_node(T_ASSIGNMENT, "=");

        param = AST_stack[--sp];
        ptr->sibling = AST_stack[--sp];
        ptr->sibling->sibling = param;

        AST_stack[sp++] = ptr;
    }
    ;


arithmetic_expression:
expression T_LPAREN
    {
        // build sibling
        AST_node *ptr;
        AST_node *param;

        ptr = build_node(T_PLUS, "+");

        param = AST_stack[--sp];
        ptr->sibling = AST_stack[--sp];
        ptr->sibling->sibling = param;

        AST_stack[sp++] = ptr;
    }
    | T_MINUS T_LPAREN expression T_SEPARATOR expression
T_LPAREN
    {
        // build sibling

```

	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

```

AST_node *ptr;
AST_node *param;

ptr = build_node(T_MINUS, "-");

param = AST_stack[--sp];
ptr->sibling = AST_stack[--sp];
ptr->sibling->sibling = param;

AST_stack[sp++] = ptr;
}
| T_MUL T_LPAREN expression T_SEPARATOR expression
T_RPAREN
{
    // build sibling
    AST_node *ptr;
    AST_node *param;

    ptr = build_node(T_MUL, "*");

    param = AST_stack[--sp];
    ptr->sibling = AST_stack[--sp];
    ptr->sibling->sibling = param;

    AST_stack[sp++] = ptr;
}
| T_DIV T_LPAREN expression T_SEPARATOR expression
T_RPAREN
{
    // build sibling
    AST_node *ptr;
    AST_node *param;

    ptr = build_node(T_DIV, "/");

    param = AST_stack[--sp];
    ptr->sibling = AST_stack[--sp];
    ptr->sibling->sibling = param;

    AST_stack[sp++] = ptr;
}
| T_REMAINDER T_LPAREN expression T_SEPARATOR
expression T_RPAREN
{
    // build sibling
    AST_node *ptr;
    AST_node *param;

    ptr = build_node(T_REMAINDER, "%");

    param = AST_stack[--sp];

```

	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

```

ptr->sibling = AST_stack[--sp];
ptr->sibling->sibling = param;

AST_stack[sp++] = ptr;

}
;

equality_expression: T_EQUAL T_LPAREN expression T_SEPARATOR expression
T_RPAREN
{
    // build sibling
    AST_node *ptr;
    AST_node *param;

    ptr = build_node(T_EQUAL, "==");

    param = AST_stack[--sp];
    ptr->sibling = AST_stack[--sp];
    ptr->sibling->sibling = param;

    AST_stack[sp++] = ptr;
}
| T_NOT_EQUAL T_LPAREN expression T_SEPARATOR
expression T_RPAREN
{
    // build sibling
    AST_node *ptr;
    AST_node *param;

    ptr = build_node(T_NOT_EQUAL, "!=");

    param = AST_stack[--sp];
    ptr->sibling = AST_stack[--sp];
    ptr->sibling->sibling = param;

    AST_stack[sp++] = ptr;
}
;

relational_expression: T_NOT T_LPAREN expression T_RPAREN
{
    // build sibling
    AST_node *ptr;

    ptr = build_node(T_NOT, "!");

    ptr->sibling = AST_stack[--sp];

    AST_stack[sp++] = ptr;
}
| T_AND T_LPAREN expression T_SEPARATOR expression
T_RPAREN

```

	국민대학교 컴퓨터공학부 컴파일러 설계과제	최종보고서	
		프로젝트 명	rusper
		성 명	
		Confidential Restricted	Version 1.0

```

{
    // build sibling
    AST_node *ptr;
    AST_node *param;

    ptr = build_node(T_AND, "&&");

    param = AST_stack[--sp];
    ptr->sibling = AST_stack[--sp];
    ptr->sibling->sibling = param;

    AST_stack[sp++] = ptr;
}
| T_OR T_LPAREN expression T_SEPARATOR expression
T_RPAREN
{
    // build sibling
    AST_node *ptr;
    AST_node *param;

    ptr = build_node(T_OR, "||");

    param = AST_stack[--sp];
    ptr->sibling = AST_stack[--sp];
    ptr->sibling->sibling = param;

    AST_stack[sp++] = ptr;
}
;
logical_expression: T_LT T_LPAREN expression T_SEPARATOR expression
T_RPAREN
{
    // build sibling
    AST_node *ptr;
    AST_node *param;

    ptr = build_node(T_LT, "<");

    param = AST_stack[--sp];
    ptr->sibling = AST_stack[--sp];
    ptr->sibling->sibling = param;

    AST_stack[sp++] = ptr;
}
| T_GT T_LPAREN expression T_SEPARATOR expression
T_RPAREN
{
    // build sibling
    AST_node *ptr;
    AST_node *param;

    ptr = build_node(T_GT, ">");

```

	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

```

        param = AST_stack[--sp];
        ptr->sibling = AST_stack[--sp];
        ptr->sibling->sibling = param;

        AST_stack[sp++] = ptr;
    }
    | T_LT_EQUAL    T_LPAREN    expression    T_SEPARATOR
expression T_RPAREN
    {
        // build sibling
        AST_node *ptr;
        AST_node *param;

        ptr = build_node(T_LT, "<=");

        param = AST_stack[--sp];
        ptr->sibling = AST_stack[--sp];
        ptr->sibling->sibling = param;

        AST_stack[sp++] = ptr;
    }
    | T_GT_EQUAL    T_LPAREN    expression    T_SEPARATOR
expression T_RPAREN
    {
        // build sibling
        AST_node *ptr;
        AST_node *param;

        ptr = build_node(T_LT, ">=");

        param = AST_stack[--sp];
        ptr->sibling = AST_stack[--sp];
        ptr->sibling->sibling = param;

        AST_stack[sp++] = ptr;
    }
;

random_expression: T_RANDOM T_LPAREN integer T_SEPARATOR integer T_RPAREN
    {
        // build sibling
        AST_node *ptr;
        AST_node *param;

        ptr = build_node(T_RANDOM, "RANDOM");

        param = AST_stack[--sp];
        ptr->sibling = AST_stack[--sp];
        ptr->sibling->sibling = param;

        AST_stack[sp++] = ptr;
    }
;

```

	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

```

iterator_expression: T_ITERATOR { AST_stack[sp++] = build_node(T_ITERATOR,
yylval.str); }
                        ;

lvalue:                identifier
                        ;

string_literal:        T_STRING_LITERAL    {    AST_stack[sp++]    =
build_node(T_STRING_LITERAL, yylval.str); }
                        ;

integer:               T_INTEGER    {    AST_stack[sp++]    =    build_node(T_INTEGER,
yylval.str); }
                        ;

identifier:            T_IDENTIFIER        {    AST_stack[sp++]        =
build_node(T_IDENTIFIER, yylval.str); }
                        ;

%%

```

4.2 구문분석기 실험 결과

gcd.rus
<p>Abstract Syntax Tree</p> <pre> 1, 2 0, 259 VAR 0, 291 a 1, 2 0, 259 VAR 0, 291 b 1, 2 0, 259 VAR 0, 291 r 1, 10 0, 273 = 0, 291 a 0, 292 64 1, 10 0, 273 = 0, 291 b </pre>

	국민대학교 컴퓨터공학부 컴파일러 설계과제	최종보고서	
		프로젝트 명	rusper
		성 명	
		Confidential Restricted	Version 1.0


```

0, 292 12
1, 6
0, 268 D0
0, 292 -1
1, 1
    1, 10
        0, 273 =
        0, 291 r
        0, 291 a
    1, 10
        0, 273 =
        0, 291 a
        0, 291 b
    1, 10
        0, 273 =
        0, 291 b
        1, 10
            0, 278 %
            0, 291 r
            0, 291 b
    1, 3
        0, 260 IF
        1, 10
            0, 284 <=
            0, 291 b
            0, 292 0
        1, 1
            1, 4
                0, 265 BREAK
1, 7
    0, 269 PRINT
    1, 8
        0, 294 "gcd is "
        0, 291 a

```

	국민대학교 컴퓨터공학부 컴파일러 설계과제	최종보고서	
		프로젝트 명	rusper
		성 명	
		Confidential Restricted	Version 1.0

prime.rus
Abstract Syntax Tree <div> 1, 2 <div> 0, 259 VAR 0, 291 prime </div> </div> 1, 2 <div> 0, 259 VAR 0, 291 i </div> 1, 2 <div> 0, 259 VAR 0, 291 flag </div> 1, 9 <div> 0, 290 OUTPUT 0, 294 "prime_result.txt" </div> 1, 10 <div> 0, 273 = 0, 291 prime 0, 292 131 </div> 1, 10 <div> 0, 273 = 0, 291 flag 0, 292 0 </div> 1, 3 <div> 0, 260 IF 1, 10 <div> 0, 282 && 1, 10 <div> 0, 284 < 0, 292 2 0, 291 prime </div> </div> </div>

	국민대학교 컴퓨터공학부 컴파일러 설계과제	최종보고서	
		프로젝트 명	rusper
		성 명	
		Confidential Restricted	Version 1.0

```

0, 280 !=
0, 292 0
1, 10
    0, 278 %
    0, 291 prime
    0, 292 2
1, 1
    1, 10
        0, 273 =
        0, 291 i
        0, 292 3
    1, 6
        0, 268 DO
        0, 292 -1
        1, 1
            1, 3
                0, 260 IF
                1, 10
                    0, 284 <=
                    0, 291 prime
                    0, 291 i
                1, 1
                    1, 10
                        0, 273 =
                        0, 291 flag
                        0, 292 1
                    1, 4
                        0, 265 BREAK
                1, 3
                    0, 260 IF
                    1, 10
                        0, 279 ==
                        0, 292 0
                    1, 10


```

	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

```

0, 278 %
0, 291 prime
0, 291 i
1, 1
1, 4
0, 265 BREAK
1, 10
0, 273 =
0, 291 i
1, 10
0, 274 +
0, 291 i
0, 292 2
1, 3
0, 260 IF
1, 10
0, 279 ==
0, 292 1
0, 291 flag
1, 1
1, 7
0, 269 PRINT
1, 8
0, 291 prime
0, 294 "is prime."
1, 3
0, 260 IF
1, 10
0, 279 ==
0, 292 0
0, 291 flag
1, 1
1, 7
0, 269 PRINT

```

	국민대학교 컴퓨터공학부 컴파일러 설계과제	최종보고서	
		프로젝트 명	rusper
		성 명	
		Confidential Restricted	Version 1.0

```

1, 8
    0, 291 prime
    0, 294 "is not prime."

```

```


random.rus
Abstract Syntax Tree
1, 6
    0, 268 DO
    0, 292 10
    1, 1
        1, 7
            0, 269 PRINT
            1, 10
                0, 289 $
        1, 7
            0, 269 PRINT
            1, 10
                0, 288 RANDOM
                0, 292 1
                0, 292 10

```

```

sum.rus
Abstract Syntax Tree
1, 2
    0, 259 VAR
    0, 291 sum
1, 10
    0, 273 =
    0, 291 sum
    0, 292 0
1, 6

```

	국민대학교 컴퓨터공학부 컴파일러 설계과제	최종보고서	
		프로젝트 명	rusper
		성 명	
		Confidential Restricted	Version 1.0

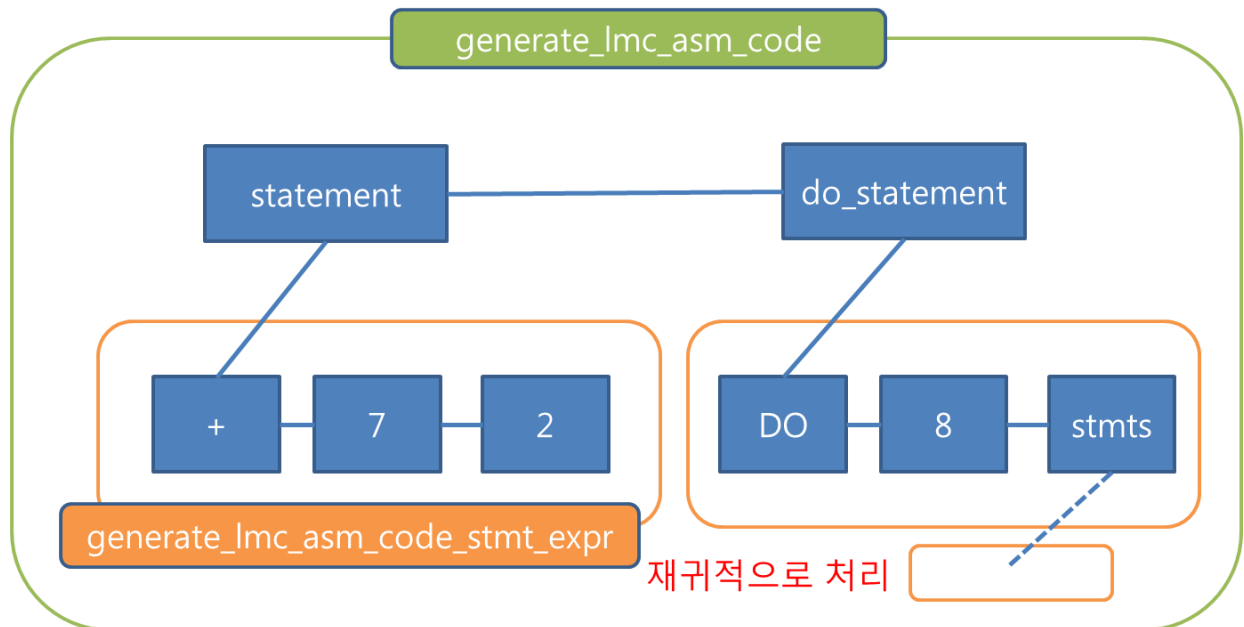
```

0, 268 DO
0, 292 10
1, 1
    1, 10
        0, 273 =
        0, 291 sum
        1, 10
            0, 274 +
            0, 291 sum
            1, 10
                0, 274 +
                1, 10
                    0, 289 $
                    0, 292 1
1, 7
    0, 269 PRINT
    0, 291 sum

```

5 목적코드 생성

5.1 목적코드 생성기 구조 및 설계



generate_lmc_asm_code 함수가 파스 트리(AST)의 루트 노드부터 차례대로 목적코드를 생성한다. 이때 각각의 child는 generate_lmc_asm_code_stmt_expr를 재귀적으로 호출하여 목적코드를 생성한다.

5.2 목적코드 생성기 구현

```

rusper.y
void generate_lmc_asm_code(FILE *fp, AST_node *root)
{
    AST_node *iter = root;

    fprintf(fp, "BOX\t_LOOP_TEMP\n");
    fprintf(fp, "BOX\t_NUM_TEMP\n");
    fprintf(fp, "BOX\t_LOGC_TEMP\n");
    fprintf(fp, "BOX\t_CNT_TEMP\n");
    fprintf(fp, "BOX\t_ITER_TEMP\n");
    fprintf(fp, "BOX\t_REDIRECTION_FILE\n");

    while(NULL != iter) {
        switch(iter->token_number) {
            case DECLARATION_STATEMENT:
                fprintf(fp, "BOX\t%s\n", iter->child->sibling-
>token_value);
                break;

```

	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

```

        case SELECTION_STATEMENT:
            generate_lmc_asm_code_stmt_expr(fp,        iter-
>child);
            break;
        case BREAK_STATEMENT:
            fprintf(fp,        "JMP        LOOP_LABEL_%d\n",
loop_label_count);
            break;
        case FOR_STATEMENT:
            break;
        case DO_STATEMENT:
            generate_lmc_asm_code_stmt_expr(fp,        iter-
>child);
            break;
        case PRINT_STATEMENT:
            break;
        case OUTPUT_STATEMENT:
            break;
        case EXPRESSION:
            generate_lmc_asm_code_stmt_expr(fp,        iter-
>child);
            break;
        default:
            fprintf(stderr, "ERROR\n");
            exit(255);
            break;
    }

    iter = iter->sibling;
}
}

```

```

rusper.y
void generate_lmc_asm_code_stmt_expr(FILE *fp, AST_node *node)
{
    AST_node *lhs;
    AST_node *rhs;

    if(NULL == node)
        return;
    if(NULL == node->sibling)
        return;

    lhs = node->sibling;
    rhs = node->sibling->sibling;

    switch(node->token_number) {
        case T_DO:
            fprintf(fp,        "LDA\t%s%s\n",    T_INTEGER    ==    lhs-
>token_number ? "#" : "", lhs->token_value);
            fprintf(fp, "STA\t_LOOP_TEMP\n");
            fprintf(fp, "LDA\t#0\n");
            fprintf(fp, "STA\t_CNT_TEMP\n");
            fprintf(fp, "LOOP_LABEL_%d:\n", loop_label_count);

```

	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

```

        fprintf(fp, "LDA\tCNT_TEMP\n");
        fprintf(fp, "CMPA\tLOOP_TEMP\n");
        fprintf(fp, "JCOND          BREAK_LABEL_%d:\n",
loop_label_count);

        // loop for compound statement
        if(NON_TERMINAL == rhs->node_type) {
            if(NON_TERMINAL == rhs->child->node_type)
                generate_lmc_asm_code_stmt_expr(fp, rhs-
>child->child);
            else
                generate_lmc_asm_code_stmt_expr(fp, rhs-
>child);
        }

        fprintf(fp, "LDA\tCNT_TEMP\n");
        fprintf(fp, "ADDA\t#1\n");
        fprintf(fp, "JMP LOOP_LABEL_%d\n", loop_label_count);
        fprintf(fp, "BREAK_LABEL_%d:\n", loop_label_count);
        ++loop_label_count;
        break;
    case T_ASSIGNMENT:
        if(NON_TERMINAL == rhs->node_type) {
            generate_lmc_asm_code_stmt_expr(fp,          rhs-
>child);
        } else {
            fprintf(fp, "LDA\t%s\n", T_INTEGER == rhs-
>token_number ? "#" : "", rhs->token_value);
        }

        fprintf(fp, "STA\t%s\n", lhs->token_value);

        break;
    case T_PLUS:
        if(NON_TERMINAL == lhs->node_type) {
            generate_lmc_asm_code_stmt_expr(fp,          lhs-
>child);
            fprintf(fp, "STA\tNUM_TEMP\n");
        } else
            fprintf(fp, "LDA\t%s\n", T_INTEGER == lhs-
>token_number ? "#" : "", lhs->token_value);

        if(NON_TERMINAL == rhs->node_type) {
            generate_lmc_asm_code_stmt_expr(fp,          rhs-
>child);
            fprintf(fp, "ADDA\tNUM_TEMP\n");
        } else
            fprintf(fp, "ADDA\t%s\n", T_INTEGER == rhs-
>token_number ? "#" : "", rhs->token_value);
        break;
    case T_MINUS:
        break;

```

	국민대학교 컴퓨터공학부 컴파일러 설계과제	최종보고서	
		프로젝트 명	rusper
		성 명	
		Confidential Restricted	Version 1.0


```

        case T_MUL:
            break;
        case T_DIV:
            break;
        case T_REMAINDER:
            break;
        case T_IF:
            if(NON_TERMINAL == lhs->node_type) {
                generate_lmc_asm_code_stmt_expr(fp, lhs->child);
            } else {
                fprintf(fp, "LDA\t%s%s\n", T_INTEGER == lhs->token_number ? "#" : "", lhs->token_value);
            }
            fprintf(fp, "STA\tLOGC_TMP\n");
            fprintf(fp, "LDA\tLOGC_TMP\n");
            fprintf(fp, "CMPA\t#0\n");
            fprintf(fp, "JCOND IF_LABEL_%d\n", cmp_label_count);

            if(NON_TERMINAL == rhs->node_type) {
                if(NON_TERMINAL == rhs->child->node_type)
                    generate_lmc_asm_code_stmt_expr(fp, rhs->child->child);
                else
                    generate_lmc_asm_code_stmt_expr(fp, rhs->child);
            }

            fprintf(fp, "IF_LABEL_%d:\n", cmp_label_count);
            ++cmp_label_count;
            break;
        case T_AND:
            break;
        case T_OR:
            break;
        case T_LT:
            break;
        case T_GT:
            break;
        case T_LT_EQUAL:
            break;
        case T_GT_EQUAL:
            break;
        case T_NOT_EQUAL:
            break;
        case T_EQUAL:
            break;
        case T_LBRACKET:
            break;
        default:
            fprintf(stderr, "ERROR   expr   :   %d\n", lhs->token_number);
            exit(255);

```


 국민대학교 컴퓨터공학부 컴파일러 설계과제	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

```

        break;
    }
}

```

5.3 목적코드 생성기 실험 결과

```

gcd.rus
Generate LMC assembler Code

BOX    _LOOP_TEMP
BOX    _NUM_TEMP
BOX    _LOGC_TEMP
BOX    _CNT_TEMP
BOX    _ITER_TEMP
BOX    _REDIRECTION_FILE
BOX    a
BOX    b
BOX    r
LDA    #64
STA    a
LDA    #12
STA    b
LDA    #-1
STA    _LOOP_TEMP
LDA    #0
STA    _CNT_TEMP
LOOP_LABEL_0:
LDA    _CNT_TEMP
CMPA   _LOOP_TEMP
JCOND  BREAK_LABEL_0:
LDA    a
STA    r
LDA    _CNT_TEMP

```

	국민대학교		
	컴퓨터공학부		
	컴파일러 설계과제		
	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

```

ADDA    #1
JMP     LOOP_LABEL_0
BREAK_LABEL_0:


```

```

prime.rus
Generate LMC assembler Code

BOX     _LOOP_TEMP
BOX     _NUM_TEMP
BOX     _LOGC_TEMP
BOX     _CNT_TEMP
BOX     _ITER_TEMP
BOX     _REDIRECTION_FILE
BOX     prime
BOX     i
BOX     flag
LDA     #131
STA     prime
LDA     #0
STA     flag
STA     _LOGC_TMP
LDA     _LOGC_TMP
CMPA    #0
JCOND   IF_LABEL_0
LDA     #3
STA     i
IF_LABEL_0:
STA     _LOGC_TMP
LDA     _LOGC_TMP
CMPA    #0
JCOND   IF_LABEL_1
IF_LABEL_1:
STA     _LOGC_TMP
LDA     _LOGC_TMP

```

	국민대학교		
	컴퓨터공학부		
	컴파일러 설계과제		
	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

```

CMPA    #0
JCOND IF_LABEL_2
IF_LABEL_2:

```

```

random.rus
Generate LMC assembler Code

BOX     _LOOP_TEMP
BOX     _NUM_TEMP
BOX     _LOGC_TEMP
BOX     _CNT_TEMP
BOX     _ITER_TEMP
BOX     _REDIRECTION_FILE
LDA     #10
STA     _LOOP_TEMP
LDA     #0
STA     _CNT_TEMP
LOOP_LABEL_0:
LDA     _CNT_TEMP
CMPA    _LOOP_TEMP
JCOND BREAK_LABEL_0:
LDA     _CNT_TEMP
ADDA    #1
JMP LOOP_LABEL_0
BREAK_LABEL_0:


```

```

sum.rus
Generate LMC assembler Code

BOX     _LOOP_TEMP
BOX     _NUM_TEMP
BOX     _LOGC_TEMP
BOX     _CNT_TEMP
BOX     _ITER_TEMP
BOX     _REDIRECTION_FILE

```

 국민대학교 컴퓨터공학부 컴파일러 설계과제	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

```

BOX      sum
LDA      #0
STA      sum
LDA      #10
STA      _LOOP_TEMP
LDA      #0
STA      _CNT_TEMP
LOOP_LABEL_0:
LDA      _CNT_TEMP
CMPA     _LOOP_TEMP
JCOND BREAK_LABEL_0:
LDA      sum
STA      _NUM_TEMP
ADDA     #1
ADDA     _NUM_TEMP
STA      sum
LDA      _CNT_TEMP
ADDA     #1
JMP LOOP_LABEL_0
BREAK_LABEL_0:

```

 국민대학교 컴퓨터공학부 컴파일러 설계과제	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

6 실험 및 평가

6.1 실험 데이터

```
gcd.rus
; 최대공약수 구하기

VAR a
VAR b
VAR r

; initialize a
=(a, 64)

; initialize b
=(b, 12)

DO(-1)
::
    =(r, a)
    =(a, b)
    =(b, %(r, b))

    IF(<=(b, 0))
    ::
        BREAK
    :
:

; print gcd
["gcd is " {a} ]

; end of source
```

```
prime.rus
; 소수 판별 프로그램

VAR prime
VAR i
VAR flag

OUTPUT("prime_result.txt")

; initialize prime
=(prime, 131)

; initialize flag
=(flag, 0)

IF(&&<(2, prime), !=(0, %(prime, 2))))
::
```

	국민대학교 컴퓨터공학부 컴파일러 설계과제	최종보고서	
		프로젝트 명	rusper
		성 명	
		Confidential Restricted	Version 1.0

```

; initialize i
=(i, 3)

DO(-1)
::
    IF(<=(prime, i))
    ::
        =(flag, 1)
        BREAK
    :
    IF(==(0, %(prime, i)))
    ::
        BREAK
    :
    =(i, +(i, 2))
:
:
IF(==(1, flag))
::
    [{prime} "is prime."]
:
IF(==(0, flag))
::
    [{prime} "is not prime."]
:
; end of source

```

```

random.rus
; random list generator

DO(10)
::
    ; print current index
    [{$}]

    ; generate random number 1 ~ 10
    [{RANDOM(1, 10)}]
:
; end of source

```

```

sum.rus
; 1 부터 10까지 합 구하기
VAR sum

; initialize sum
=(sum, 0)

```

	국민대학교 컴퓨터공학부 컴파일러 설계과제	최종보고서	
		프로젝트 명	rusper
		성 명	
		Confidential Restricted	Version 1.0

```


; repeat 10 times
DO(10)
::
    =(sum, +(sum, +($, 1)))

    ; print sum
    [{sum}]
:
; end of source

```

6.2 실행 결과

gcd.rus	
Abstract Syntax Tree	
1, 2	0, 259 VAR
	0, 291 a
1, 2	0, 259 VAR
	0, 291 b
1, 2	0, 259 VAR
	0, 291 r
1, 10	0, 273 =
	0, 291 a
	0, 292 64
1, 10	0, 273 =
	0, 291 b
	0, 292 12
1, 6	0, 268 DO
	0, 292 -1
1, 1	1, 1
	1, 10
	0, 273 =
	0, 291 r
	0, 291 a
	1, 10
	0, 273 =
	0, 291 a
	0, 291 b
	1, 10
	0, 273 =
	0, 291 b
	1, 10
	0, 278 %
	0, 291 r
	0, 291 b
1, 3	0, 260 IF

	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

```

1, 10
0, 284 <=
0, 291 b
0, 292 0
1, 1
1, 4
0, 265 BREAK
1, 7
0, 269 PRINT
1, 8
0, 294 "gcd is "
0, 291 a

```

Generate LMC assembler Code

```

BOX _LOOP_TEMP
BOX _NUM_TEMP
BOX _LOGC_TEMP
BOX _CNT_TEMP
BOX _ITER_TEMP
BOX _REDIRECTION_FILE
BOX a
BOX b
BOX r
LDA #64
STA a
LDA #12
STA b
LDA #-1
STA _LOOP_TEMP
LDA #0
STA _CNT_TEMP
LOOP_LABEL_0:
LDA _CNT_TEMP
CMPA _LOOP_TEMP
JCOND BREAK_LABEL_0:
LDA a
STA r
LDA _CNT_TEMP
ADDA #1
JMP LOOP_LABEL_0
BREAK_LABEL_0:

```

prime.rus

Abstract Syntax Tree

```

1, 2
0, 259 VAR
0, 291 prime
1, 2
0, 259 VAR
0, 291 i
1, 2
0, 259 VAR
0, 291 flag

```


	국민대학교 컴퓨터공학부 컴파일러 설계과제	최종보고서	
		프로젝트 명	rusper
		성 명	
		Confidential Restricted	Version 1.0

```

1, 9
    0, 290 OUTPUT
    0, 294 "prime_result.txt"
1, 10
    0, 273 =
    0, 291 prime
    0, 292 131
1, 10
    0, 273 =
    0, 291 flag
    0, 292 0
1, 3
    0, 260 IF
    1, 10
        0, 282 &&
        1, 10
            0, 284 <
            0, 292 2
            0, 291 prime
        1, 10
            0, 280 !=
            0, 292 0
            1, 10
                0, 278 %
                0, 291 prime
                0, 292 2
    1, 1
        1, 10
            0, 273 =
            0, 291 i
            0, 292 3
        1, 6
            0, 268 DO
            0, 292 -1
            1, 1
                1, 3
                    0, 260 IF
                    1, 10
                        0, 284 <=
                        0, 291 prime
                        0, 291 i
                    1, 1
                        1, 10
                            0, 273 =
                            0, 291 flag
                            0, 292 1
                        1, 4
                            0, 265 BREAK
                1, 3
                    0, 260 IF
                    1, 10
                        0, 279 ==
                        0, 292 0
                        1, 10

```


	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

```

                                0, 278 %
                                0, 291 prime
                                0, 291 i
                                1, 1
                                1, 4
                                0, 265 BREAK
                                1, 10
                                0, 273 =
                                0, 291 i
                                1, 10
                                0, 274 +
                                0, 291 i
                                0, 292 2
1, 3
    0, 260 IF
    1, 10
        0, 279 ==
        0, 292 1
        0, 291 flag
    1, 1
        1, 7
            0, 269 PRINT
            1, 8
                0, 291 prime
                0, 294 "is prime."
1, 3
    0, 260 IF
    1, 10
        0, 279 ==
        0, 292 0
        0, 291 flag
    1, 1
        1, 7
            0, 269 PRINT
            1, 8
                0, 291 prime
                0, 294 "is not prime."

Generate LMC assembler Code
BOX    _LOOP_TEMP
BOX    _NUM_TEMP
BOX    _LOGC_TEMP
BOX    _CNT_TEMP
BOX    _ITER_TEMP
BOX    _REDIRECTION_FILE
BOX    prime
BOX    i
BOX    flag
LDA    #131
STA    prime
LDA    #0
STA    flag
STA    _LOGC_TMP
LDA    _LOGC_TMP

```

	국민대학교		최종보고서	
	컴퓨터공학부		프로젝트 명	rusper
	컴파일러 설계과제		성 명	
			Confidential Restricted	Version 1.0

```

CMPA    #0
JCOND   IF_LABEL_0
LDA      #3
STA      i
IF_LABEL_0:
STA      _LOGC_TMP
LDA      _LOGC_TMP
CMPA     #0
JCOND   IF_LABEL_1
IF_LABEL_1:
STA      _LOGC_TMP
LDA      _LOGC_TMP
CMPA     #0
JCOND   IF_LABEL_2
IF_LABEL_2:

```

```

random.rus
Abstract Syntax Tree
1, 6
    0, 268 DO
    0, 292 10
    1, 1
        1, 7
            0, 269 PRINT
            1, 10
                0, 289 $
        1, 7
            0, 269 PRINT
            1, 10
                0, 288 RANDOM
                0, 292 1
                0, 292 10

Generate LMC assembler Code
BOX      _LOOP_TEMP
BOX      _NUM_TEMP
BOX      _LOGC_TEMP
BOX      _CNT_TEMP
BOX      _ITER_TEMP
BOX      _REDIRECTION_FILE
LDA      #10
STA      _LOOP_TEMP
LDA      #0
STA      _CNT_TEMP
LOOP_LABEL_0:
LDA      _CNT_TEMP
CMPA     _LOOP_TEMP
JCOND    BREAK_LABEL_0:
LDA      _CNT_TEMP
ADDA     #1
JMP      LOOP_LABEL_0
BREAK_LABEL_0:

```

	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

sum.rus

Abstract Syntax Tree

```

1, 2
    0, 259 VAR
    0, 291 sum
1, 10
    0, 273 =
    0, 291 sum
    0, 292 0
1, 6
    0, 268 DO
    0, 292 10
    1, 1
        1, 10
            0, 273 =
            0, 291 sum
            1, 10
                0, 274 +
                0, 291 sum
                1, 10
                    0, 274 +
                    1, 10
                        0, 289 $
                        0, 292 1
1, 7
    0, 269 PRINT
    0, 291 sum


```

Generate LMC assembler Code


```

BOX    _LOOP_TEMP
BOX    _NUM_TEMP
BOX    _LOGC_TEMP
BOX    _CNT_TEMP
BOX    _ITER_TEMP
BOX    _REDIRECTION_FILE
BOX    sum
LDA    #0
STA    sum
LDA    #10
STA    _LOOP_TEMP
LDA    #0
STA    _CNT_TEMP
LOOP_LABEL_0:
LDA    _CNT_TEMP
CMPA   _LOOP_TEMP
JCOND  BREAK_LABEL_0:
LDA    sum
STA    _NUM_TEMP
ADDA   #1
ADDA   _NUM_TEMP
STA    sum
LDA    _CNT_TEMP
ADDA   #1
JMP    LOOP_LABEL_0

```

 국민대학교 컴퓨터공학부 컴파일러 설계과제	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

BREAK_LABEL_0:

 국민대학교 컴퓨터공학부 컴파일러 설계과제	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

7 결론

새로운 언어의 설계를 위해 다양한 언어를 비교하고 분석하는 과정에서 기존 언어의 설계 철학이나 효율성과 사용성의 트레이드 오프에 대한 힌트를 얻을 수 있었다. 또한 다양한 언어 중에서 오랜 기간동안 사용되는 언어가 있는 반면에 잠깐 사용되거나 학문적으로만 연구되는 언어를 보면서 단순히 설계만으로는 어떤 언어가 우위에 있다고 할 수 없다고 알게 되었다. 기존 언어에서 비효율적이거나 불필요해 보였던 문법적 제약사항에 대해서 피상적으로 이해하는 것이 아닌, 컴파일러 구조 전체를 통한 이해를 할 수 있게 되었다.

또한 특정 도메인에서 큰 힘을 발휘하는 언어를 통해서 범용적 언어뿐만 아니라 특화된 언어의 필요성을 인식하게 되었다. 테스트케이스를 작성하는 언어라는 일종의 특화된 목적으로 언어를 설계하면서, 언어 설계에 대한 경험을 통해, 설계에 대한 포괄적인 이해를 할 수 있었다. 정규식을 통한 어휘분석을 통해 정규언어에 대한 이해를 높일 수 있었고, 구문분석을 통한 파스 트리의 생성은 효율적이고 변환 가능한 구조에 대한 통찰을 제공해 주었다. 파스 트리를 통한 목적코드 생성을 통해 동일한 의미를 다른 형태로 표현하는 기법에 대한 경험을 얻을 수 있었다. 추가적으로 최적화 기법을 고려하면서 같은 의미지만 보다 효율적인 코드에 대한 생각을 할 수 있게 해주었다.

새로운 언어를 설계하면서 그동안 배웠던 모든 Computer Science의 철학, 기법, 방법론 등이 컴파일러 설계에 집약되어 있음을 알게 되었다. 언어를 설계한다는 것은 단순히 컴퓨터 프로그래밍 언어를 만드는 것이 아닌 Computer Science 전반에 걸친 이해뿐만 아니라 문제를 해결하는 방법에 대한 통찰력을 제공해준다.

	국민대학교 컴퓨터공학부 컴파일러 설계과제	최종보고서	
		프로젝트 명	rusper
		성 명	
		Confidential Restricted	Version 1.0

부록 1. Lex 입력 파일

```

rusper.l
%{
/*
 * Kookmin University
 * 컴퓨터공학부
 * 3학년
 * 정태성
 *
 * rusper.l
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include "y.tab.h"

enum {
    STACK_MAX = 1024,
};

extern int cmp_stmt_cnt_stack[STACK_MAX];
extern int cmp_stmt_cnt_sp;
extern int cmp_stmt_cnt;
extern int print_stmt_cnt;
extern int sp;

%}
letter [a-zA-Z_]
nonzero_digit [1-9]
digit [0-9]
%%
"VAR"          { yylval.str = yytext; return T_VAR; }
"IF"           { yylval.str = yytext; return T_IF; }
"("            { yylval.str = yytext;return T_LPAREN;}
")"            { yylval.str = yytext;return T_RPAREN;}
":::"          { yylval.str = yytext; cmp_stmt_cnt_stack[cmp_stmt_cnt_sp++] =
sp; return T_DCOLON;}
":"            { yylval.str    = yytext;    cmp_stmt_cnt    =    sp    -
cmp_stmt_cnt_stack[--cmp_stmt_cnt_sp]; return T_COLON;}
"BREAK"        { yylval.str = yytext;return T_BREAK;}
"FOR"          { yylval.str = yytext;return T_FOR;}
","            { yylval.str = yytext;return T_SEPARATOR;}
"DO"           { yylval.str = yytext;return T_DO;}
"["            { yylval.str    = yytext;    print_stmt_cnt    =    sp;    return
T_LBRACKET;}
"]"            { yylval.str = yytext; print_stmt_cnt = sp - print_stmt_cnt;
return T_RBRACKET; }
"{"            { yylval.str = yytext;return T_LCURLYBRACE;}
"}"            { yylval.str = yytext;return T_RCURLYBRACE;}
"="            { yylval.str = yytext;return T_ASSIGNMENT;}

```

	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

```

"+"      { yylval.str = yytext;return T_PLUS;}
"-"      { yylval.str = yytext;return T_MINUS;}
"*"      { yylval.str = yytext;return T_MUL;}
"/"      { yylval.str = yytext;return T_DIV;}
%"      { yylval.str = yytext;return T_REMAINDER;}
"=="     { yylval.str = yytext;return T_EQUAL;}
"!="     { yylval.str = yytext;return T_NOT_EQUAL;}
"!"      { yylval.str = yytext;return T_NOT;}
"&&"     { yylval.str = yytext;return T_AND;}
"||"     { yylval.str = yytext;return T_OR;}
"<"      { yylval.str = yytext;return T_LT;}
">"      { yylval.str = yytext;return T_GT;}
"<="     { yylval.str = yytext;return T_LT_EQUAL;}
">="     { yylval.str = yytext;return T_GT_EQUAL;}
"RANDOM"  { yylval.str = yytext;return T_RANDOM;}
"$"      { yylval.str = yytext;return T_ITERATOR;}
"OUTPUT" { yylval.str = yytext;return T_OUTPUT;}
";".*    { yylval.str = yytext;return T_COMMENT;}
\".*\"   { yylval.str = yytext;return T_STRING_LITERAL;}

{letter}({letter}|{digit})* { yylval.str = yytext; return T_IDENTIFIER; }
[-+]?{digit}+               { yylval.str = yytext; return T_INTEGER; }

[ \t\n\r]      ;
.               return T_ERROR;

%%

/*

int main(void)
{
    int token_number;

    printf("Start of rusper\n");
    while(0 != (token_number = yylex())) {
        printf("<%d, %s>\n", token_number, yytext);
    }

    printf("End of rusper\n");

    return 0;
}
// */

int yywrap(void)
{
    return 1;
}

```


	국민대학교		
	컴퓨터공학부		
	컴파일러 설계과제		
	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

부록 2. Yacc 입력 파일(파스 트리, 목적코드 생성 소스 포함)

```

rusper.y
%{
/*
 * Kookmin University
 * 컴퓨터공학부
 * 3학년
 * 정태성
 *
 * rusper.y
 *
 */

#include <stdio.h>
#include <string.h>

enum node_number {
    STATEMENT_LIST,
    COMPOUND_STATEMENT,
    DECLARATION_STATEMENT,
    SELECTION_STATEMENT,
    BREAK_STATEMENT,
    FOR_STATEMENT,
    DO_STATEMENT,
    PRINT_STATEMENT,
    PRINT_EXPR_LIST,
    OUTPUT_STATEMENT,
    EXPRESSION
};

enum {
    TERMINAL = 0,
    NON_TERMINAL,
    STACK_MAX = 1024,
};

typedef struct _AST_node {
    int token_number;
    char *token_value;
    int node_type;
    struct _AST_node *child;
    struct _AST_node *sibling;
} AST_node;

int cmp_stmt_cnt_stack[STACK_MAX];
int cmp_stmt_cnt_sp;
int cmp_stmt_cnt;
int print_stmt_cnt;

AST_node *AST_stack[STACK_MAX];
int sp;

```

	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

```

AST_node *AST_root;

AST_node *build_node(int t_num, char *t_val);
AST_node *build_tree(int n_num);

int loop_label_count;
int cmp_label_count;
void generate_lmc_asm_code(FILE *fp, AST_node *root);
void generate_lmc_asm_code_stmt_expr(FILE *fp, AST_node *node);

void treverse(AST_node *node, int tab);
void print_stack(char *s);

%}

%union {
    int integer;
    char *str;
}

%token T_EOF
%token T_ERROR
%token T_VAR
%token T_IF
%token T_LPAREN
%token T_RPAREN
%token T_COLON
%token T_DCOLON
%token T_BREAK
%token T_FOR
%token T_SEPARATOR
%token T_DO
%token T_LBRACKET
%token T_RBRACKET
%token T_LCURLYBRACE
%token T_RCURLYBRACE
%token T_ASSIGNMENT
%token T_PLUS
%token T_MINUS
%token T_MUL
%token T_DIV
%token T_REMAINDER
%token T_EQUAL
%token T_NOT_EQUAL
%token T_NOT
%token T_AND
%token T_OR
%token T_LT
%token T_GT
%token T_LT_EQUAL
%token T_GT_EQUAL
%token T_RANDOM
%token T_ITERATOR

```

	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

```

%token T_OUTPUT
%token T_IDENTIFIER
%token T_INTEGER
%token T_COMMENT
%token T_STRING_LITERAL

%%
program:      file
              ;

file:         statement_list
              {
                  AST_node *iter;
                  int i;

                  iter = AST_root = AST_stack[0];
                  for(i = 1; i < sp; ++i) {
                      iter->sibling = AST_stack[i];
                      iter = iter->sibling;
                  }
              }
              ;

statement_list:  statement
                | statement statement_list
                ;

compound_statement:  statement
                    {
                        AST_node *ptr;
                        AST_node *iter;
                        int stmt_sp;
                        int cmp_stmt_sp;

                        cmp_stmt_sp = stmt_sp = sp - cmp_stmt_cnt;

                        ptr = build_tree(COMPOUND_STATEMENT);
                        iter = ptr->child = AST_stack[stmt_sp];

                        ++stmt_sp;
                        while(stmt_sp < sp) {
                            iter->sibling = AST_stack[stmt_sp];
                            iter = iter->sibling;
                            ++stmt_sp;
                        }

                        AST_stack[cmp_stmt_sp] = ptr;
                        sp = cmp_stmt_sp + 1;
                    }
                    | statement compound_statement
                    ;

statement:      declaration_statement
                {

```

	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

```

// build tree
AST_node *ptr;

ptr = build_tree(DECLARATION_STATEMENT);
ptr->child = AST_stack[--sp];

// push
AST_stack[sp++] = ptr;
}
| expression_statement
| selection_statement
{
    // build tree
    AST_node *ptr;

    ptr = build_tree(SELECTION_STATEMENT);
    ptr->child = AST_stack[--sp];

    // push
    AST_stack[sp++] = ptr;
}
| break_statement
{
    // build tree
    AST_node *ptr;

    ptr = build_tree(BREAK_STATEMENT);
    ptr->child = AST_stack[--sp];

    // push
    AST_stack[sp++] = ptr;
}
| for_statement
{
    // build tree
    AST_node *ptr;

    ptr = build_tree(FOR_STATEMENT);
    ptr->child = AST_stack[--sp];

    // push
    AST_stack[sp++] = ptr;
}
| do_statement
{
    // build tree
    AST_node *ptr;

    ptr = build_tree(DO_STATEMENT);
    ptr->child = AST_stack[--sp];

```

	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

```

        // push
        AST_stack[sp++] = ptr;
    }
    | print_statement
    {
        // build tree
        AST_node *ptr;

        ptr = build_tree(PRINT_STATEMENT);
        ptr->child = AST_stack[--sp];

        // push
        AST_stack[sp++] = ptr;
    }
    | output_statement
    {
        // build tree
        AST_node *ptr;

        ptr = build_tree(OUTPUT_STATEMENT);
        ptr->child = AST_stack[--sp];

        // push
        AST_stack[sp++] = ptr;
    }
    | comment_statement
    ;

declaration_statement:      T_VAR identifier
    {
        // build sibling
        AST_node *ptr;

        ptr = build_node(T_VAR, "VAR");
        ptr->sibling = AST_stack[--sp];

        AST_stack[sp++] = ptr;
    }
    ;

expression_statement: expression
    ;

selection_statement: T_IF      T_LPAREN      expression      T_RPAREN      T_DCOLON
compound_statement T_COLON
    {
        // build sibling
        AST_node *ptr;
        AST_node *param;

        ptr = build_node(T_IF, "IF");

```

	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

```

        param = AST_stack[--sp];
        ptr->sibling = AST_stack[--sp];
        ptr->sibling->sibling = param;

        AST_stack[sp++] = ptr;
    }
    ;

break_statement:    T_BREAK
    {
        AST_stack[sp++] = build_node(T_BREAK, "BREAK");
    }
    ;

for_statement:      T_FOR  T_LPAREN  integer  T_SEPARATOR  integer  T_RPAREN
T_DCOLON compound_statement T_COLON
    {
        // build sibling
        AST_node *ptr;
        AST_node *param;
        AST_node *cmp_stmt;

        ptr = build_node(T_FOR, "FOR");

        cmp_stmt = AST_stack[--sp];
        param = AST_stack[--sp];
        ptr->sibling = AST_stack[--sp];
        ptr->sibling->sibling = param;
        ptr->sibling->sibling->sibling = cmp_stmt;

        AST_stack[sp++] = ptr;
    }
    ;

do_statement:       T_DO      T_LPAREN      integer      T_RPAREN      T_DCOLON
compound_statement T_COLON
    {
        // build sibling
        AST_node *ptr;
        AST_node *param;

        ptr = build_node(T_DO, "DO");

        param = AST_stack[--sp];
        ptr->sibling = AST_stack[--sp];
        ptr->sibling->sibling = param;

        AST_stack[sp++] = ptr;
    }
    ;

print_statement:    T_LBRACKET print_expression_list T_RBRACKET
    {
        // build sibling

```

	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

```

AST_node *ptr;

ptr = build_node(T_LBRACKET, "PRINT");
ptr->sibling = AST_stack[--sp];

AST_stack[sp++] = ptr;
}
;

print_expression_list:    print_expression
| print_expression print_expression_list
{
    AST_node *ptr;
    AST_node *iter;
    int stmt_sp;
    int print_stmt_sp;

    print_stmt_sp = stmt_sp = sp - print_stmt_cnt;

    ptr = build_tree(PRINT_EXPR_LIST);
    iter = ptr->child = AST_stack[stmt_sp];

    ++stmt_sp;
    while(stmt_sp < sp) {
        iter->sibling = AST_stack[stmt_sp];
        iter = iter->sibling;
        ++stmt_sp;
    }

    AST_stack[print_stmt_sp] = ptr;
    sp = print_stmt_sp + 1;
}
;

print_expression:    string_literal
| T_LCURLYBRACE expression T_RCURLYBRACE
;

output_statement:    T_OUTPUT T_LPAREN string_literal T_RPAREN
{
    // build sibling
    AST_node *ptr;

    ptr = build_node(T_OUTPUT, "OUTPUT");
    ptr->sibling = AST_stack[--sp];

    AST_stack[sp++] = ptr;
}
;

comment_statement:    T_COMMENT
;

expression:    assignment_expression

```



```
{
    // build tree
    AST_node *ptr;

    ptr = build_tree(EXPRESSION);
    ptr->child = AST_stack[--sp];

    // push
    AST_stack[sp++] = ptr;
}
| arithmetic_expression
{
    // build tree
    AST_node *ptr;

    ptr = build_tree(EXPRESSION);
    ptr->child = AST_stack[--sp];

    // push
    AST_stack[sp++] = ptr;
}
| equality_expression
{
    // build tree
    AST_node *ptr;

    ptr = build_tree(EXPRESSION);
    ptr->child = AST_stack[--sp];

    // push
    AST_stack[sp++] = ptr;
}
| relational_expression
{
    // build tree
    AST_node *ptr;

    ptr = build_tree(EXPRESSION);
    ptr->child = AST_stack[--sp];

    // push
    AST_stack[sp++] = ptr;
}
| logical_expression
{
    // build tree
    AST_node *ptr;

    ptr = build_tree(EXPRESSION);
    ptr->child = AST_stack[--sp];

    // push
    AST_stack[sp++] = ptr;
}
```


	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

```

| random_expression
{
    // build tree
    AST_node *ptr;

    ptr = build_tree(EXPRESSION);
    ptr->child = AST_stack[--sp];

    // push
    AST_stack[sp++] = ptr;
}
| iterator_expression
{
    // build tree
    AST_node *ptr;

    ptr = build_tree(EXPRESSION);
    ptr->child = AST_stack[--sp];

    // push
    AST_stack[sp++] = ptr;
}
| integer
| identifier
;

assignment_expression:      T_ASSIGNMENT    T_LPAREN    lvalue    T_SEPARATOR
expression T_RPAREN
{
    // build sibling
    AST_node *ptr;
    AST_node *param;

    ptr = build_node(T_ASSIGNMENT, "=");

    param = AST_stack[--sp];
    ptr->sibling = AST_stack[--sp];
    ptr->sibling->sibling = param;

    AST_stack[sp++] = ptr;
}
;

arithmetic_expression:      T_PLUS      T_LPAREN      expression    T_SEPARATOR
expression T_RPAREN
{
    // build sibling
    AST_node *ptr;
    AST_node *param;

    ptr = build_node(T_PLUS, "+");

    param = AST_stack[--sp];
    ptr->sibling = AST_stack[--sp];

```

	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

```

ptr->sibling->sibling = param;

AST_stack[sp++] = ptr;
}
| T_MINUS T_LPAREN expression T_SEPARATOR expression
T_RPAREN
{
    // build sibling
    AST_node *ptr;
    AST_node *param;

    ptr = build_node(T_MINUS, "-");

    param = AST_stack[--sp];
    ptr->sibling = AST_stack[--sp];
    ptr->sibling->sibling = param;

    AST_stack[sp++] = ptr;
}
| T_MUL T_LPAREN expression T_SEPARATOR expression
T_RPAREN
{
    // build sibling
    AST_node *ptr;
    AST_node *param;

    ptr = build_node(T_MUL, "*");

    param = AST_stack[--sp];
    ptr->sibling = AST_stack[--sp];
    ptr->sibling->sibling = param;

    AST_stack[sp++] = ptr;
}
| T_DIV T_LPAREN expression T_SEPARATOR expression
T_RPAREN
{
    // build sibling
    AST_node *ptr;
    AST_node *param;

    ptr = build_node(T_DIV, "/");

    param = AST_stack[--sp];
    ptr->sibling = AST_stack[--sp];
    ptr->sibling->sibling = param;

    AST_stack[sp++] = ptr;
}
| T_REMAINDER T_LPAREN expression T_SEPARATOR

```

	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

```

expression T_RPAREN
{
    // build sibling
    AST_node *ptr;
    AST_node *param;

    ptr = build_node(T_REMAINDER, "%");

    param = AST_stack[--sp];
    ptr->sibling = AST_stack[--sp];
    ptr->sibling->sibling = param;

    AST_stack[sp++] = ptr;
}
;

equality_expression: T_EQUAL T_LPAREN expression T_SEPARATOR expression
T_RPAREN
{
    // build sibling
    AST_node *ptr;
    AST_node *param;

    ptr = build_node(T_EQUAL, "==");

    param = AST_stack[--sp];
    ptr->sibling = AST_stack[--sp];
    ptr->sibling->sibling = param;

    AST_stack[sp++] = ptr;
}
| T_NOT_EQUAL T_LPAREN expression T_SEPARATOR
expression T_RPAREN
{
    // build sibling
    AST_node *ptr;
    AST_node *param;

    ptr = build_node(T_NOT_EQUAL, "!=");

    param = AST_stack[--sp];
    ptr->sibling = AST_stack[--sp];
    ptr->sibling->sibling = param;

    AST_stack[sp++] = ptr;
}
;

relational_expression: T_NOT T_LPAREN expression T_RPAREN
{
    // build sibling
    AST_node *ptr;

```

	국민대학교 컴퓨터공학부 컴파일러 설계과제	최종보고서	
		프로젝트 명	rusper
		성 명	
		Confidential Restricted	Version 1.0

```

ptr = build_node(T_NOT, "!");

ptr->sibling = AST_stack[--sp];

AST_stack[sp++] = ptr;
}
| T_AND T_LPAREN expression T_SEPARATOR expression
T_RPAREN
{
    // build sibling
    AST_node *ptr;
    AST_node *param;

    ptr = build_node(T_AND, "&&");

    param = AST_stack[--sp];
    ptr->sibling = AST_stack[--sp];
    ptr->sibling->sibling = param;

    AST_stack[sp++] = ptr;
}
| T_OR T_LPAREN expression T_SEPARATOR expression
T_RPAREN
{
    // build sibling
    AST_node *ptr;
    AST_node *param;

    ptr = build_node(T_OR, "||");

    param = AST_stack[--sp];
    ptr->sibling = AST_stack[--sp];
    ptr->sibling->sibling = param;

    AST_stack[sp++] = ptr;
}
;

logical_expression: T_LT T_LPAREN expression T_SEPARATOR expression
T_RPAREN
{
    // build sibling
    AST_node *ptr;
    AST_node *param;

    ptr = build_node(T_LT, "<");

    param = AST_stack[--sp];
    ptr->sibling = AST_stack[--sp];
    ptr->sibling->sibling = param;

    AST_stack[sp++] = ptr;
}

```

	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

```

T_RPAREN      | T_GT  T_LPAREN  expression  T_SEPARATOR  expression
               {
               // build sibling
               AST_node *ptr;
               AST_node *param;

               ptr = build_node(T_GT, ">");

               param = AST_stack[--sp];
               ptr->sibling = AST_stack[--sp];
               ptr->sibling->sibling = param;

               AST_stack[sp++] = ptr;
               }
expression T_RPAREN  | T_LT_EQUAL  T_LPAREN  expression  T_SEPARATOR
               {
               // build sibling
               AST_node *ptr;
               AST_node *param;

               ptr = build_node(T_LT, "<=");

               param = AST_stack[--sp];
               ptr->sibling = AST_stack[--sp];
               ptr->sibling->sibling = param;

               AST_stack[sp++] = ptr;
               }
expression T_RPAREN  | T_GT_EQUAL  T_LPAREN  expression  T_SEPARATOR
               {
               // build sibling
               AST_node *ptr;
               AST_node *param;

               ptr = build_node(T_LT, ">=");

               param = AST_stack[--sp];
               ptr->sibling = AST_stack[--sp];
               ptr->sibling->sibling = param;

               AST_stack[sp++] = ptr;
               }
               ;

random_expression:  T_RANDOM T_LPAREN integer T_SEPARATOR integer T_RPAREN
               {
               // build sibling
               AST_node *ptr;
               AST_node *param;

               ptr = build_node(T_RANDOM, "RANDOM");

```

	국민대학교 컴퓨터공학부 컴파일러 설계과제	최종보고서	
		프로젝트 명	rusper
		성 명	
		Confidential Restricted	Version 1.0

```

        param = AST_stack[--sp];
        ptr->sibling = AST_stack[--sp];
        ptr->sibling->sibling = param;

        AST_stack[sp++] = ptr;
    }
    ;

iterator_expression: T_ITERATOR { AST_stack[sp++] = build_node(T_ITERATOR,
yylval.str); }
    ;

lvalue:
    identifier
    ;

string_literal:
        T_STRING_LITERAL    {    AST_stack[sp++]    =
build_node(T_STRING_LITERAL, yylval.str); }
    ;

integer:
        T_INTEGER    {    AST_stack[sp++]    =    build_node(T_INTEGER,
yylval.str); }
    ;

identifier:
        T_IDENTIFIER    {    AST_stack[sp++]    =
build_node(T_IDENTIFIER, yylval.str); }
    ;

%%

AST_node *build_node(int t_num, char *t_val)
{
    AST_node *ptr;
    int val_len;

    ptr = (AST_node *) malloc(sizeof(AST_node));
    if(NULL == ptr) {
        fprintf(stderr, "Error : %s : malloc\n", __func__);
        exit(255);
    }

    ptr->token_number = t_num;

    val_len = strlen(t_val) + 1;
    ptr->token_value = (char *) malloc(sizeof(char) * val_len);
    strncpy(ptr->token_value, t_val, val_len);

    ptr->node_type = TERMINAL;
    ptr->child = ptr->sibling = NULL;

    return ptr;
}

AST_node *build_tree(int n_num)

```

	국민대학교 컴퓨터공학부 컴파일러 설계과제	최종보고서	
		프로젝트 명	rusper
		성 명	
		Confidential Restricted	Version 1.0

```

{
    AST_node *ptr;

    ptr = (AST_node *) malloc(sizeof(AST_node));
    if(NULL == ptr) {
        fprintf(stderr, "Error : %s : malloc\n", __func__);
        exit(255);
    }

    ptr->token_number = n_num;
    ptr->token_value = NULL;
    ptr->node_type = NON_TERMINAL;
    ptr->child = ptr->sibling = NULL;

    return ptr;
}

void generate_lmc_asm_code(FILE *fp, AST_node *root)
{
    AST_node *iter = root;

    fprintf(fp, "BOX\t_LOOP_TEMP\n");
    fprintf(fp, "BOX\t_NUM_TEMP\n");
    fprintf(fp, "BOX\t_LOGC_TEMP\n");
    fprintf(fp, "BOX\t_CNT_TEMP\n");
    fprintf(fp, "BOX\t_ITER_TEMP\n");
    fprintf(fp, "BOX\t_REDIRECTION_FILE\n");

    while(NULL != iter) {
        switch(iter->token_number) {
            case DECLARATION_STATEMENT:
                fprintf(fp, "BOX\t%s\n", iter->child->sibling->token_value);
                break;
            case SELECTION_STATEMENT:
                generate_lmc_asm_code_stmt_expr(fp, iter->child);
                break;
            case BREAK_STATEMENT:
                fprintf(fp, "JMP LOOP_LABEL_%d\n", loop_label_count);
                break;
            case FOR_STATEMENT:
                break;
            case DO_STATEMENT:
                generate_lmc_asm_code_stmt_expr(fp, iter->child);
                break;
            case PRINT_STATEMENT:
                break;
            case OUTPUT_STATEMENT:
                break;
            case EXPRESSION:

```

	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

```

                                generate_lmc_asm_code_stmt_expr(fp,      iter-
>child);
                                break;
                                default:
                                fprintf(stderr, "ERROR\n");
                                exit(255);
                                break;
                                }
                                iter = iter->sibling;
                                }
                                }

void generate_lmc_asm_code_stmt_expr(FILE *fp, AST_node *node)
{
    AST_node *lhs;
    AST_node *rhs;

    if(NULL == node)
        return;
    if(NULL == node->sibling)
        return;

    lhs = node->sibling;
    rhs = node->sibling->sibling;

    switch(node->token_number) {
        case T_DO:
            fprintf(fp, "LDA\t%s%s\n", T_INTEGER == lhs-
>token_number ? "#" : "", lhs->token_value);
            fprintf(fp, "STA\t_LOOP_TEMP\n");
            fprintf(fp, "LDA\t#0\n");
            fprintf(fp, "STA\t_CNT_TEMP\n");
            fprintf(fp, "LOOP_LABEL_%d:\n", loop_label_count);
            fprintf(fp, "LDA\t_CNT_TEMP\n");
            fprintf(fp, "CMPA\t_LOOP_TEMP\n");
            fprintf(fp, "JCOND          BREAK_LABEL_%d:\n",
loop_label_count);

            // loop for compound statement
            if(NON_TERMINAL == rhs->node_type) {
                if(NON_TERMINAL == rhs->child->node_type)
                    generate_lmc_asm_code_stmt_expr(fp, rhs-
>child->child);
                else
                    generate_lmc_asm_code_stmt_expr(fp, rhs-
>child);
            }

            fprintf(fp, "LDA\t_CNT_TEMP\n");
            fprintf(fp, "ADDA\t#1\n");
            fprintf(fp, "JMP LOOP_LABEL_%d\n", loop_label_count);
            fprintf(fp, "BREAK_LABEL_%d:\n", loop_label_count);
            ++loop_label_count;

```


	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

```

        break;
    case T_ASSIGNMENT:
        if(NON_TERMINAL == rhs->node_type) {
            generate_lmc_asm_code_stmt_expr(fp,          rhs->
>child);
        } else {
            fprintf(fp, "LDA\t%s%s\n", T_INTEGER == rhs->
>token_number ? "#" : "", rhs->token_value);
        }

        fprintf(fp, "STA\t%s\n", lhs->token_value);

        break;
    case T_PLUS:
        if(NON_TERMINAL == lhs->node_type) {
            generate_lmc_asm_code_stmt_expr(fp,          lhs->
>child);
            fprintf(fp, "STA\t_NUM_TEMP\n");
        } else
            fprintf(fp, "LDA\t%s%s\n", T_INTEGER == lhs->
>token_number ? "#" : "", lhs->token_value);

        if(NON_TERMINAL == rhs->node_type) {
            generate_lmc_asm_code_stmt_expr(fp,          rhs->
>child);
            fprintf(fp, "ADDA\t_NUM_TEMP\n");
        } else
            fprintf(fp, "ADDA\t%s%s\n", T_INTEGER == rhs->
>token_number ? "#" : "", rhs->token_value);
        break;
    case T_MINUS:
        break;
    case T_MUL:
        break;
    case T_DIV:
        break;
    case T_REMAINDER:
        break;
    case T_IF:
        if(NON_TERMINAL == lhs->node_type) {
            generate_lmc_asm_code_stmt_expr(fp,          lhs->
>child);
        } else {
            fprintf(fp, "LDA\t%s%s\n", T_INTEGER == lhs->
>token_number ? "#" : "", lhs->token_value);
        }
        fprintf(fp, "STA\t_LOGC_TMP\n");
        fprintf(fp, "LDA\t_LOGC_TMP\n");
        fprintf(fp, "CMPA\t#0\n");
        fprintf(fp, "JCOND IF_LABEL_%d\n", cmp_label_count);

```

	최종보고서		
	프로젝트 명	rusper	
	성 명		
	Confidential Restricted	Version 1.0	

```

        if(NON_TERMINAL == rhs->node_type) {
            if(NON_TERMINAL == rhs->child->node_type)
                generate_lmc_asm_code_stmt_expr(fp, rhs->child->child);
            else
                generate_lmc_asm_code_stmt_expr(fp, rhs->child);
        }

        fprintf(fp, "IF_LABEL_%d:\n", cmp_label_count);
        ++cmp_label_count;
        break;
    case T_AND:
        break;
    case T_OR:
        break;
    case T_LT:
        break;
    case T_GT:
        break;
    case T_LT_EQUAL:
        break;
    case T_GT_EQUAL:
        break;
    case T_NOT_EQUAL:
        break;
    case T_EQUAL:
        break;
    case T_LBRACKET:
        break;
    default:
        fprintf(stderr, "ERROR   expr   :   %d\n", node->token_number);
        exit(255);
        break;
    }
}

void treverse(AST_node *node, int tab)
{
    int i;

    if(NULL == node)
        return;

    for(i = 0; i < tab; ++i)
        printf("\t");

    printf("%d, %d\t", node->node_type, node->token_number);
    if(node->node_type == TERMINAL)
        printf("%s\n", node->token_value);
    else

```

	국민대학교 컴퓨터공학부 컴파일러 설계과제	최종보고서	
		프로젝트 명	rusper
		성 명	
		Confidential Restricted	Version 1.0

```

        printf("\n");

        treverse(node->child, tab + 1);
        treverse(node->sibling, tab);
    }

void print_stack(char *s)
{
    int i;
    printf("%s\n", s);
    for(i = sp - 1; 0 <= i; --i) {

        printf("[%d] %s, %d ", i, AST_stack[i]->node_type ? "NON" :
"TER", AST_stack[i]->token_number);

        if(AST_stack[i]->node_type == TERMINAL) {
            printf(": %s\n", AST_stack[i]->token_value);
        } else
            printf("\n");
    }
}

///  

int main(void)
{
    sp = 0;
    cmp_stmt_cnt_sp = 0;
    loop_label_count = 0;
    cmp_label_count = 0;

    yyparse();

    printf("Abstract Syntax Tree\n");
    treverse(AST_root, 0);

    printf("\nGenerate LMC assembler Code\n");
    generate_lmc_asm_code(stdout, AST_root);

    return 0;
}
// */

void yyerror(char *s)
{
    fprintf(stderr, "%s\n", s);
}

```