

MINIC 用コンパイラの構文解析

ソースコード

```
%{
#include <stdio.h>
%}
%union {char *s; int n;}
%token <s> NAME NUMBER
%destructor { free($$); } NAME NUMBER
%token <n> IF WHILE DO
%type <n> if0 num
%token AND OR EQ GE GOTO ELSE INT IN OUT HALT
%left OR
%left AND
%left '|'
%left '^'
%left '&'
%left EQ NE
%left GE LE '<' '>'
%left SHL SHR
%left '+' '-'
%left '*'
%right '!' '~' NEG
%%
statements : statement | statements statement
;
statement : label | intdef | goto | if | while | do | halt | out | assign
;
label : NAME ':' {printf("%s:\n", $1);}
;
intdef: INT intlist ';'
;
intlist: integer
      | intlist ',' integer
;
integer: NAME {printf("%s:%t0\n", $1);}
        | NAME '=' num {printf("%s:%t%d\n", $1, $3);}
        | NAME '=' {printf("%s:", $1);} '{' numlist '}'
;
num: NUMBER {$$=atoi($1);}
    | '-' NUMBER { $$=-atoi($2); }
;
numlist: nums
        | numlist ',' nums
;
nums : num {printf("%t%d\n", $1);}
      | num '[' NUMBER ']' {int i; for(i=0; i<atoi($3); ++i) printf("%t%d\n", $1);}
;
goto: GOTO NAME ';' {printf("%tJMP %s\n", $2);}
;
if: if0 {printf("_%03dF:\n", $1);}
   | if0 {printf("%tJMP _%03dT\n_%03dF:\n", $1, $1);} ELSE '{' statements '}' {printf("_%03dT:\n", $1);}
;
if0: IF '(' expr ')' {printf("%tJZ _%03dF\n", $1);} '{' statements '}' {$$=$1;}
;
while: WHILE {printf("_%03dT:\n", $1);} '(' expr ')' {printf("%tJZ _%03dF\n", $1);} '{' statements '}'
      {printf("%tJMP _%03dT\n_%03dF:\n", $1, $1);}
;
do: DO {printf("_%03dT:\n", $1);} '{' statements '}' WHILE '(' expr ')' ';' {printf("%tJNZ _%03dT\n", $1);}
;
assign: NAME '=' expr ';' {printf("%tPOP %s\n", $1);}
        | NAME {printf("%tPUSHI %s\n", $1);} '[' expr ']' {printf("%tADD\n");} '=' expr ';'
      {printf("%tST\n");}
;
halt : HALT ';' {printf("%tHALT\n");}
;
out: OUT '(' expr ')' ';' {printf("%tOUT\n");}
;
expr: NAME {printf("%tPUSH %s\n", $1);}
     | NUMBER {printf("%tPUSHI %s\n", $1);}
     | NAME {printf("%tPUSHI %s\n", $1);} '[' expr ']' {printf("%tADD\n%tLD\n");}
     | IN {printf("%tIN\n");}
     | '!' expr {printf("%tNOT\n");}
     | '-' expr {printf("%tBNOT\n");}
     | '-' expr %prec NEG {printf("%tNEG\n");}
     | expr '+' expr {printf("%tADD\n");}
```

```

| expr '-' expr {printf("%tSUB\n");}
| expr '*' expr {printf("%tMUL\n");}
| expr AND expr {printf("%tAND\n");}
| expr OR expr {printf("%tOR\n");}
| expr '&' expr {printf("%tBAND\n");}
| expr '|' expr {printf("%tBOR\n");}
| expr '^' expr {printf("%tBXOR\n");}
| expr SHL expr {printf("%tSHL\n");}
| expr SHR expr {printf("%tSHR\n");}
| expr EQ expr {printf("%tEQ\n");}
| expr NE expr {printf("%tNE\n");}
| expr GE expr {printf("%tGE\n");}
| expr LE expr {printf("%tLE\n");}
| expr '<' expr {printf("%tLT\n");}
| expr '>' expr {printf("%tGT\n");}
| '(' expr ')'
;
%%
int yyerror(char *s){ printf("%s\n",s); }
int main(){ yyparse(); }

```