“Elementary Compiler”

งานชิ้นที่ 4 วิชา 01076262 Compiler Construction

จัดทำโดย

นายสุทธิชัย พงศานนท์ รหัสประจำตัว 57011395

นายอรรถสิทธิ์ สินธัญญาธรรม รหัสประจำตัว 57011501

นายอิสรา นรานิรัติศัย รหัสประจำตัว 57011546

เสนอ

ผศ. อัครเดช วัชระภูพงษ์

ปีการศึกษา 2559 ภาคเรียนที่ 2

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

Lex<flex> : เป็นการตัดข้อความออกเป็น token แล้วส่งให้ yacc ไปตรวจไวยากรณ์และตีความต่อ ปัญหาเกิดขึ้นเมื่อต้องการให้ใส่ "ข้อความ" ได้ และเป็นการกำหนดรูปแบบไวยกรณ์ เช่น present($A)

Yacc<bison> : มีหน้าที่ตรวจสอบว่า token ต่าง ๆ ที่รับมาจาก lex นั้นเรียงลำดับถูกต้องตามที่ควรจะเป็นโดยเราจะต้องเขียน "กฎไวยากรณ์" ให้ครบในทุก ๆ กรณี

รายละเอียดวิธีการดำเนินงานสร้างเชิงเทคนิค

แนวคิด : การจัดวางแกรมม่าโดยเรียงตามลำดับความสำคัญของเครื่องหมาย ถ้าหากมีความสำคัญมากแกรมม่าจะอยู่ในโหนดที่ต่ำ หากมีความสำคัญน้อยจะอยู่โหนดที่สูงขึ้นตามลำดับ

ไวยากรณ์และตัวอย่างที่ทำให้เข้าใจภาษานั้นง่าย

```
S :VAR '=' E '\n'                          { var[$1]=$3;
                                             MemVar(STORE, $3, $1);
                                             releaseRegister(); }
  | IF ':' E EQ E '\n'                     {jmpIf(push(st), $3, $5);}        /*If */
  | LOOP ':' VAR ':' E TO E '\n'           {int conna =push(st);
                                            int connb =push(st);
                                             jmpLoop($3, conna, connb, $5, $7);
                                             pushLoopSig(st); }              /*For Loop */
  | PRESENT ':' STR '\n'                   {}                    /*Print number in decimal */
  | PRESENT ':' E '\n'                     { asmprintfInt($3);
                                             releaseRegister(); }
  | PRESENTHEX ':' E '\n'                  { asmprintfHex($3);
                                             releaseRegister();}
  | UNKNOWN {printf("!ERROR :Unknown operation\n");}  /*"!ERROR"when out of gramma character */
  | E '\n'                                 { releaseRegister();}
  | END '\n'                               {if(st->data ==-1) {
                                                pop(st);
                                                int connb =pop(st);
                                                int conna =pop(st);
                                                jmpEndLoop(conna, connb); }
                                           else { jmpEndIf(pop(st)); }
                                           }
  | '\n'                                   {}
  ;
E :E '+' T          { $$ =$1;
                      createOp(ADD, $1, $3);
                      releaseRegister();}
  | E '-' T          { $$ =$1;
                      createOp(SUB, $1, $3);
                      releaseRegister();}
  | T                { $$ =$1;}
  ;
```

```
T :T '*' F            { $$ =$1;
                        createOp(MULT, $1, $3);
                        releaseRegister();}
  | T '/' F           { $$ =$1;
                       createOp(DIV, $1, $3);
                       releaseRegister();}
  | T '\\' F          { $$ =$1;
                       createOp(MOD, $1, $3);
                       releaseRegister();}
  | F                 { $$ =$1;}
  ;
F :'(' E ')'          { $$ =$2;}
  | '-' F             { $$ =$2;
                       addNegative($2); }
  | NUM               {$$ =nextFreeRegister();
                       RegConst($$, $1); }
  | VAR               { $$ =nextFreeRegister();
                       MemVar(LOAD, $$, $1); }
  ;
STR :STRING           { asmprintfString($1);}
    | STR '+' STRING { asmprintfString($3);}
    ;
```

คำอธิบายโค้ด flex

```
%{
 #include <stdio.h>
 #include <stdlib.h>
 #include <string.h>
 #include "createasm.h"
 extern int yylex();
 typedef struct yy_buffer_state *YY_BUFFER_STATE;
 extern int yyparse();
 extern YY_BUFFER_STATE yy_scan_string(char *str);
 extern void yy_delete_buffer(YY_BUFFER_STATE buffer);
 void yyerror(char *msg);
 char itoa(int val, int pos);
 int res =0 ;
 char snum[10];
 char reader[128];
 int var[676];
 int setRes =0;
 int autoIncJmp =0;
 extern FILE *fp;
 struct node   /*structure of stack */
 {
       int data;
       struct node* next;
 };
 void init(struct node* head) /*create a stack*/
 {
```

```
        head=NULL;
 }
struct node* st =NULL;
int push(struct node* head)/*push data to stack */
 {
        struct node* tmp =(struct node*)malloc(sizeof(struct node));
        if(tmp ==NULL)/*if create node fail */
        {
          exit(0);
        }
        tmp->data =autoIncJmp++;
        tmp->next =head;
        st =tmp;
        return st->data;
 }
void pushLoopSig(struct node* head)/*push data to stack */
 {
        struct node* tmp =(struct node*)malloc(sizeof(struct node));
        if(tmp ==NULL)/*if create node fail */
        {
          exit(0);
        }
        tmp->data =-1;
        tmp->next =head;
        st =tmp;
 }
int pop(struct node *head)/*pop stack */
 {
        if(head==NULL){ /*if stack is Empty */
        return -100;
 }
  struct node* tmp =head;
  int data =head->data;
      st =head->next;
      free(tmp);
  return data;
  }
%}
%union {
  int i;
  char c;
  char* str;
}
%token <i> IF EQ LOOP TO END /*Condition Token */
%token <i> NUM PRESENT PRESENTHEX     /*Options Token */
%token <i> UNKNOWN /*Error Token */
%token <i> VAR
%token <str> STRING
%type <i> E T F
%type <str> STR
%%
program:
```

```
  program S
  | /*NULL */
  ;
S :VAR '=' E '\n'                          { var[$1]=$3;
                                             MemVar(STORE, $3, $1);
                                             releaseRegister(); }
  | IF ':' E EQ E '\n'                      {jmpIf(push(st), $3, $5);}        /*If */
  | LOOP ':' VAR ':' E TO E '\n'            {int conna =push(st);
                                             int connb =push(st);
                                             jmpLoop($3, conna, connb, $5, $7);
                                             pushLoopSig(st); }             /*For Loop */
  | PRESENT ':' STR '\n'                    {}                      /*Print number in decimal */
  | PRESENT ':' E '\n'                      { asmprintfInt($3);
                                             releaseRegister(); }
  | PRESENTHEX ':' E '\n'                   { asmprintfHex($3);
                                             releaseRegister();}
  | UNKNOWN {printf("!ERROR :Unknown operation\n");}  /*"!ERROR"when out of gramma character */
  | E '\n'                                  { releaseRegister();}
  | END '\n'                                {if(st->data ==-1) {
                                                 pop(st);
                                                 int connb =pop(st);
                                                 int conna =pop(st);
                                                 jmpEndLoop(conna, connb); }
                                            else { jmpEndIf(pop(st)); }
                                            }
  | '\n'                                    {}
  ;
E :E '+' T            { $$ =$1;
                        createOp(ADD, $1, $3);
                        releaseRegister();}
  | E '-' T           { $$ =$1;
                        createOp(SUB, $1, $3);
                        releaseRegister();}
  | T                 { $$ =$1;}
  ;
T :T '*' F            { $$ =$1;
                       createOp(MULT, $1, $3);
                       releaseRegister();}
  | T '/' F           { $$ =$1;
                      createOp(DIV, $1, $3);
                      releaseRegister();}
  | T '\\' F          { $$ =$1;
                      createOp(MOD, $1, $3);
                      releaseRegister();}
  | F                 { $$ =$1;}
  ;
F :'(' E ')'          { $$ =$2;}
  | '-' F             { $$ =$2;
                      addNegative($2); }
  | NUM               {$$ =nextFreeRegister();
                      RegConst($$, $1); }
```

```
  | VAR              { $$ = nextFreeRegister();
                        MemVar(LOAD, $$, $1); }
  ;
STR :STRING          { asmprintfString($1);}
    | STR '+' STRING { asmprintfString($3);}
    ;

%%
void yyerror(char *msg) {
       fprintf(stderr, "%s\n", msg);
}
int main(int argc, char *argv[]) {
       FILE *fr = fopen(argv[1], "r");
       char *filename = strtok(argv[1], ".");
       filename = strcat(filename, ".asm");
       fp = fopen(filename, "w");
       initasm();
       while(fgets(reader, 128, fr)){
  //          printf("%s", reader);
            YY_BUFFER_STATE buffer = yy_scan_string(reader);
            yyparse();
            yy_delete_buffer(buffer);
       }
       initvariable();
       fclose(fp);
       fclose(fr);
        return 0;
}
char itoa(int val, int pos){
       if(pos==0){
       return val%26 + 'A';
       }
       else{
       return val/26 + 'A';
       }
}
```

คำอธิบายโค้ด bison

```
%{
 #include <stdio.h>
 #include <stdlib.h>
 #include <string.h>
 #include "y.tab.h"
 void yyerror(char *);
 int htoi(char *);    /* change base 16 to base 10 number */
 int Ohtoi(char c);   /* change base 16 to base 10 number (1 character)*/
 int azToInt(char c);
%}

%%
```

```
[0-9]+                  {yylval.i = atoi(yytext); return NUM;}      /* number Token (0-infinity)*/
[0-9A-Fa-f]+[Hh]        {yylval.i = htoi(yytext); return NUM;}     /* base 16 number Token (15AH) */
"$"[A-Za-z][A-Za-z]     {yylval.i = azToInt(yytext[1])*26 + azToInt(yytext[2]); return VAR;}
/* Variable Token ($AA-$ZZ) */
[-+*():\n/=\\]          {return *yytext;}                           /* Operation Token */
L?\"(\\.|[^\\"])*\"     {yylval.str = yytext; return STRING;}
"present"               {return PRESENT;}                           /* Present Something */
"present_h"             {return PRESENTHEX;}                        /* Present Something in Hex */
"eq"                    {return EQ;}                                /* Equal or Not(in Boolean) */
"loop"                  {return LOOP;}                                 /* Loop */
"if"                    {return IF;}                                   /* If   */
"end"                   {return END;}                                  /* End  */
"to"                    {return TO;}                                   /* To   */
[ \t\f\v]; /             /Ignore
.                       /* out of gramma character Token */
%%

int Ohtoi(char c){     /* change base 16 to base 10 number (1 character) */
        int r;
        if(c>='0' && c<='9'){
                r = c - '0';
        }
        else if(c>='A' && c<='F'){
                r = c - 'A' + 10;
        }
        else if(c>='a' && c<='f'){
                r = c - 'a' + 10;
        }
        return r;
}

int htoi(char *s){   /* change base 16 to base 10 number */
        int i;
        int result = Ohtoi(s[0]);
        for(i=1; i<strlen(s)-1; i++){
                result *= 16;
                result += Ohtoi(s[i]);
        }
        return result;
}

int azToInt(char c){
 if(c >= 'a' && c <= 'z'){
  return c-'a';
   }
 else if(c >= 'A' && c <= 'Z'){
  return c-'A';
   } //else error occur
}

int yywrap(void){ return 1; }
```

ผลการรันในรูปแบบที่ถูกต้อง

Ex.Code

```
//1.print str
present:"Assignment Compiler"
$AD =0                  //2.assign value
present:$AD             //3.print variable
loop:$AS:1 to 2         //4.loop
  loop:$IN:1 to 3       //5.loop loop
  if:1 eq $IN           //6.if eq
       $AD =$AD+1
       //7.print str
       present:"Result:$AD"
     end
  end
end
present:$AD
present_h:126           //8.print hex
present:10//((2+5*2)\5) //9.+*/\()
present:-10+2           //10.-
```

Result.code

```
Assignment Compiler
0
Result:$AD
2
7e
5
-8
```

*ไม่ได้พิมพ์สีเขียวลงไปในโค้ด

ผลการรันในรูปแบบที่ไม่ถูกต้อง

Ex.Code

```
//1.print str
present:Assignment Compiler
$AD =0;                 //2.assign value
present:$AA             //3.print variable
loop:$AS:1 to 2         //4.loop
   $AD =$AD+1
present_h:126           //5.print hex
```

Result.code

```
Syntax error
```

อ้างอิง

http://paepae.exteen.com/20060912/lex-yacc-flex-bison

https://youtu.be/yTXCPGAD3SQ