

Programski prevodioci: Vežbe 8

Sadržaj

1. Uvod	1
2. Napomena za rešavanje zadataka	1
3. Rešenja zadataka	1
3.1. Zadatak 1: iterate	1
3.2. Zadatak 2: switch iskaz	2

1. Uvod

U dokumentu su data rešenja zadataka koji su rađeni na devetim vežbama.

2. Napomena za rešavanje zadataka

Svi zadaci se rešavaju sledećim redosledom:

- Dodati nove tokene na vrh **.y** datoteke.
- Definisati regularne izraze u **.l** datoteci za nove tokene.
- Proširiti gramatiku jezika tako da sintaksno podržava novu konstrukciju.
- Dodati semantičke provere.
- Osmisliti, za 1 konkretni primer, kako ekvivalentan asemblerski kod treba da izgleda.
- Uopštiti asemblerski kod iz prethodnog koraka i implementirati generisanje koda.

3. Rešenja zadataka

3.1. Zadatak 1: **iterate**

Sintaksa je data, potrebno je dodati samo generisanje koda:

Gramatika:

```
iterate_statement
: _ITERATE _ID
{
    int i = lookup_symbol($2, VAR|PAR );
    code("\n\t\tMOV \t$1, ");
    gen_sym_name(i);
    $<i>$ = ++lab_num;
```

```

    code("\n@iterate%d:", lab_num);

}

literal _TO literal
{
    int i = lookup_symbol($2, VAR|PAR);

    gen_cmp(i, $6);
    if(get_type(i) == INT)
        code("\n\t\tJGTS \t");
    else
        code("\n\t\tJGTU \t");
    code("@iterator_end%d", $<i>3);

}
statement
{
    int i = lookup_symbol($2, VAR|PAR);
    if(get_type(i) == INT)
        code("\n\t\tADDS \t");
    else
        code("\n\t\tADDU \t");
    gen_sym_name(i);
    code(",");
    gen_sym_name($4);
    code(",");
    gen_sym_name(i);

    code("\n\t\tJMP\t@iterate%d", $<i>3);
    code("\n@iterator_end%d: ", $<i>3);
}
;

```

3.2. Zadatak 2: **switch** iskaz

Globalne promenljive:

```

int case_count = 0;
int case_array[100];
int switch_id = -1;

```

Novi tokeni:

```

%token _SWITCH
%token _CASE
%token _BREAK

```

```
%token _DEFAULT  
%token _COLON
```

Tipovi pojmoveva:

```
%type <i> default_statement
```

Gramatika:

```
statement  
: ...  
| switch_statement  
;  
  
switch_statement  
: _SWITCH _LPAREN _ID  
{  
    if( (switch_id = lookup_symbol($3, VAR)) == -1)  
        err("%s undeclared", $3);  
    lab_num++;  
    code("\n@switch%d:", lab_num);  
    code("\n\t\tJMP \t@test%d", lab_num);  
}  
_RPAREN _LBRACKET case_statements default_statement _RBRACKET  
{  
    code("\n\t\tJMP \t@exit%d", lab_num);  
    code("\n@test%d:", lab_num);  
    int i;  
    for(i = 0; i < case_count; i++) {  
        gen_cmp(switch_id, case_array[i]);  
        case_array[i] = -1; //ponisti sadrzaj  
        code("\n\t\tJEQ \t");  
        code("@case%d_%d", lab_num, i);  
    }  
  
    if($8)  
        code("\n\t\tJMP \t@default%d", lab_num);  
    code("\n@exit%d:", lab_num);  
    case_count = 0;  
}  
;  
  
case_statements  
: case_statement  
| case_statements case_statement  
;  
  
case_statement
```

```

: _CASE literal _COLON
{
    // provera jedinstvenosti konstanti
    int i = 0;
    while(i < case_count) {
        if($2 == case_array[i]) { //ako takva konstanta vec postoji u nizu
            err("duplicated constant in case");
            break;
        }
        i++;
    }
    if(i == case_count) { //ako nije duplikat
        case_array[case_count] = $2; //ubaci konstantu u niz
        code("\n@case%d:", lab_num, case_count);
        case_count++;
    }

    //provera tipa konstante
    if(get_type($2) != get_type(switch_id))
        err("wrong type of constant");
}
statement break_statement
;

break_statement
: /* empty */
| _BREAK _SEMICOLON
{
    code("\n\t\tJMP \t@exit%d", lab_num);
}
;

default_statement
: /* empty */
{
    $$ = 0;
}
| _DEFAULT _COLON
{
    code("\n@default%d:", lab_num);
}
statement
{
    $$ = 1;
}
;

```