

Programski prevodioci: Vežbe 7

Sadržaj

| | |
|---|---|
| 1. Uvod..... | 1 |
| 2. Napomena za rešavanje zadataka | 1 |
| 3. Rešenja zadataka | 1 |
| 3.1. Zadatak 1: globalne promenljive | 1 |
| 3.2. Zadatak 2: while iskaz | 3 |

1. Uvod

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

2. Napomena za rešavanje zadataka

Svi zadaci se rešavaju sledećim redosledom:

- Dodati nove tokene na vrh **.y** datoteke.
- Definirati regularne izraze u **.l** datoteci za nove tokene.
- Proširiti gramatiku jezika tako da sintaksno podržava novu konstrukciju.
- Dodati semantičke provere.
- Osmisliti, za 1 konkretan 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: **globalne promenljive**

Definisati novu vrstu simbola **GVAR**, tako što ćete ubaciti **GVAR = 0x40** na kraj enumeracije **kinds** (u datoteci **defs.h**).

Proširiti gramatiku jezika tako da omogući definisanje globalnih promenljivih:

```
program
: global_list function_list
{ ... }
;
```

```
global_list
```

```

: /* empty */
| global_list global_var
;

global_var
: _TYPE _ID _SEMICOLON
{
    int idx = lookup_symbol($2, GVAR);
    if (idx != NO_INDEX) {
        err("redefinition of '%s'", $2);
    }
    else {
        insert_symbol($2, GVAR, $1, NO_ATR, NO_ATR);
        code("\n%s:\n\t\tWORD\t1", $2);
    }
}
;

```

U iskazu dodele omogućiti pojavu globalne promenljive sa leve strane znaka **=**:

```

assignment_statement
: _ID _ASSIGN num_exp _SEMICOLON
{
    int idx = lookup_symbol($1, VAR|PAR|GVAR);
    if(idx == NO_INDEX)
        err("invalid lvalue '%s' in assignment", $1);
    else
        if(get_type(idx) != get_type($3))
            err("incompatible types in assignment");
    gen_mov($3, idx);
}
;

```

Takođe, omogućiti pojavu globalne promenljive u izrazima:

```

exp
: _ID
{
    $$ = lookup_symbol($1, VAR|PAR|GVAR);
    if($$ == NO_INDEX)
        err("'%' undeclared", $1);
}
...
;

```

3.2. Zadatak 2: while iskaz

Definisati token _WHILE

Proširiti gramatiku jezika:

```
%{
    int while_num = -1;
}%

...

while_statement
: _WHILE
{
    $<i>$ = ++while_num;
    code("\n@while_%d:", while_num);
}
_LPAREN rel_exp _RPAREN
{
    code("\n\t\t%s\t@end_while_%d", opp_jumps[$4], while_num);
}
statement
{
    code("\n\t\tJMP\t@while_%d", $<i>2);
    code("\n@end_while_%d:", $<i>2);
}
;

statement
: compound_statement
| assignment_statement
| if_statement
| return_statement
| while_statement
;
```