Лабораторная работа №7

Разработка синтаксического анализатора для упрощённой программы на языке «Ассемблер»

7.1 Пример программы, подвергающейся синтаксическому анализу

```
ten          db     0ah
hun     dw 100
ohun    db 101
thous dw        0ff6h
hund  db 78h
str     db 1ah

mul thous
mul hun
mul bx
mul cx
mul ax
pop     dx
pop     ds
pop hun
mov thous,ax
mov     al,ten
mov      ax,hun
mov     thous,cs
mov     hun,es
mov   ds,hun
        mov    bl       ,        ten
```

7.2 Таблица кодов лексем

Лексема	Код		Лексема	Код
,	1		AL, AH, BL, BH, CL, CH, DL, DH	7
db	2		AX, BX, CX, DX	8
dw	3		DS, SS, ES	9
mov	4		CS	10
mul	5		Целое число	11
pop	6		Идентификатор	12

7.3 Результат лексического разбора программы

Номер строки	Код лексемы	Спецификатор лексемы
1	12	TEN
1	2	
1	11	0AH
2	12	HUN
2	3	
2	11	100
3	12	OHUN
3	2	
3	11	101
4	12	THOUS
4	3	
4	11	0FF6H
5	12	HUND
5	2	
5	11	78H
6	12	STR
6	2	
6	11	1AH
8	6	
8	12	THOUS
9	6	
9	12	HUN
10	6	
10	8	BX
11	6	
11	8	CX

12	6	
12	8	AX
13	5	
13	8	DX
14	5	
14	9	DS
15	5	
15	12	HUN
16	4	
16	12	THOUS
16	1	
16	8	AX
17	4	
17	7	AL
17	1	
17	12	TEN
18	4	
18	8	AX
18	1	
18	12	HUN
19	4	
19	12	THOUS
19	1	
19	10	
20	4	
20	12	HUN
20	1	
20	9	ES
21	4	
21	9	DS
21	1	
21	12	HUN
22	4	
22	7	BL
22	1	
22	12	TEN

7.4 Синтаксический анализ программы

Листинг 1 – основная программа Laba7.pas

```
{ Синтаксический анализатор:  Команды: POP, MUL, MOV.
Директивы: DB, DW }
uses    crt, lab7_u;
var    cur_str:integer;    strg:string;   f:text;
      i:integer;          ch:char; { символ из файла }
      str_index:integer; { номер строки в файле }
      file_str:string;   { файл для синтакс.разбора }
begin
      clrscr;
      for i:=1 to 1000 do lex_table[i].str_N:=0;    { Обнуляем таблицу лексем }
      curr_lex:=1;         { Текущая лексема =1 }
      write('Введи имя файла с расширением: ');
      readln(file_str);
      assign(f,file_str);  reset(f);           { Открываем файл на чтение }
      str_index:=1;  { номер строки =1 }
      strg:='';         { строка пустая }
      { читаем посимвольно из файла }
      while not eof(f) do begin
          read(f,ch);
          { Если встретили Enter}
          if ch=#13 then begin
                        writeln(strg); { выводим строку }
                        { передаем строку и ее индекс для разбора }
                        string_analize(strg,str_index);
                        inc(str_index); { увеличиваем индекс строки}
                        strg:='';  read(f,ch); { очищаем строку }
                    end
```

```
        else begin
                { Если символ не Enter }
                ch:=upcase(ch); { преобразуем к верхнему регистру }
                { Если эти символы, то добавляем к строке strg }
                if (ch in ["",' ',',',#13,#10,#9,'A'..'Z','0'..'9','_','?','@','$','&'])
                then strg:=strg+ch
                else begin
                     { иначе ошибка - символ не из алфавита }
                     writeln('Ошибка в строке №',str_index);
                     readln; halt;
                    end;
            end;
    end;
    close(f); { закрываем файл }
    writeln(strg);        { выводим и разбираем последнюю строку }
    string_analize(strg,str_index);
    { выводим таблицу в файл results.txt }
    i:=1;
    assign(f,'results.txt');     rewrite(f);   { открываем файл на запись }
    { пока строка в таблице не пуста, записываем ее в файл }
    while lex_table[i].str_N<>0 do begin
   writeln(f,lex_table[i].str_N:15,lex_table[i].lex_N:15,lex_table[i].Name:15);
      i:=i+1;{ переходим к следующей строке }
    end;
    close(f);         { закрываем файл results.txt }
    synth_analize(i);
    if i=0 then writeln('Ошибок нет') else writeln('Ошибка в строке № ',i);
    readln;
end.
```

Листинг 2 – Подключаемый модуль Laba_7u.pas

```
unit lab7_u;
interface
{ лексема: номер строки, номер лексемы, имя }
type    lexemme = record
                    str_N:integer;
                    lex_N:integer;
                    Name:string[7];
                  end;
      { reg1 – массив регистров общего назначения длиной байт }
const
      { reg1 – массив регистров общего назначения длиной байт }
       reg1:array [1..8] of string[2] = ('AL','CL','DL','BL','AH','CH', 'BH','DH');
       { reg2 - массив регистров общего назначения длиной слово }
       reg2:array [1..8] of string[2] = ('AX','CX','DX','BX','SP','BP','SI','DI');
       { sreg - массив сегметных регистров длиной слово }
       sreg:array [1..3] of string[2] = ('ES','SS','DS');
       { константы допустимых лексем }
       cs = 'CS';         db = 'DB';         dw = 'DW';
       mul = 'MUL';   pop = 'POP';     mov = 'MOV';
var     { Таблица лексем }
       lex_table:array [1..1000] of lexemme;
       { номер текущей лексемы }
       curr_lex:integer;
                              { функции и процедуры модуля }
  { является ли регистр байтовым регистром общего назначения }
  function is_reg1(s:string):boolean;
  { является ли регистр словным регистром общего назначения }
  function is_reg2(s:string):boolean;
```

```
  { является ли регистр сегментным }
  function is_sreg(s:string):boolean;
  {  определение кода лексемы  }
  procedure word_analize(s:string; i:integer);
  { разбор строки }
  procedure string_analize(s:string; i:integer);
  { проверка правильности введенных чисел }
  function number(i:integer):byte;
  { Является ли ячейка памяти байтом }
  function check_already_db(n:integer):boolean;
  { Является ли ячейка памяти словом }
  function check_already_dw(n:integer):boolean;
  { является ли лексема db }
  function _db(i:integer):boolean;
  { является ли лексема dw }
  function _dw(i:integer):boolean;
  { является ли лексема mov }
  function _mov(i:integer):boolean;
  { является ли лексема mul }
  function _mul(i:integer):boolean;
  { является ли лексема pop }
  function _pop(i:integer):boolean;
  { возвращение кода синтаксического разбор. 0-нет ошибок }
  procedure synth_analize(var err:integer);
implementation
uses crt;
{*************************************************************}
```

{Является ли регистр байтовым ?}

```
function is_reg1(s:string):boolean;
```

```
var i:byte;
begin
   { изначально не является }
   is_reg1:=false;
   { проверяем совпадение восьми регистров с переданной строкой }
   for i:=1 to 8 do
      { если да, то}
      if s=reg1[i] then begin
                  { функция возвращает true }
                  is_reg1:=true;
                  { в текущей строке таблицы лексем
                   имени присваиваем имя этого регистра }
                  lex_table[curr_lex].Name:=reg1[i];
                  break;
               end;
end;
{*************************************************************}
```

{Является ли регистр словным ?}

```
function is_reg2(s:string):boolean;
var i:byte;
begin
   { изначально не является }
   is_reg2:=false;
{ проверяем совпадение семи регистров с переданнной строкой }
   for i:=1 to 8 do
      { если да, то}
      if s=reg2[i] then begin
                  { функция возвращает true }
                  is_reg2:=true;
```

```
                    { в текущей строке таблицы лексем
                     имени присваиваем имя этого регистра }
                    lex_table[curr_lex].Name:=reg2[i];
                    break;
                end;
end;
{*************************************************************}
```

{Является ли регистр сегментным ?}

```
function is_sreg(s:string):boolean;
var i:byte;
begin
   { изначально не является }
   is_sreg:=false;
{ проверяем совпадение трех регистров с переданнной строкой }
   for i:=1 to 3 do
      { если да, то}
      if s=sreg[i] then begin
                    { функция возвращает true }
                    is_sreg:=true;
                    { в текущей строке таблицы лексем
                     имени присваиваем имя этого регистра }
                    lex_table[curr_lex].Name:=sreg[i];
                    break;
                end;
end;
{*************************************************************}
```

{Формирование одной строки таблицы лексем}

```
procedure word_analize(s:string; i:integer);
var     No:integer;
```

```
begin
    { определение кода лексемы  в текущей строке таблицы лексем }
    { номеру строки присваиваем i }
    lex_table[curr_lex].str_N:=i;
    { ,  –  код 1 }
    if s=',' then No:=1
    { db  –  код 2 }
    else if s=db then No:=2
    { dw  –  код 3 }
    else if s=dw then No:=3
    { mov  –  код 4 }
    else if s=mov then No:=4
    { pop  –  код 5 }
    else if s=pop then No:=5
    { mul  –  код 6 }
    else if s=mul then No:=6
    { байтовый регистр  –  код 7 }
    else if is_reg1(s) then No:=7
    { словный регистр  –  код 8 }
    else if is_reg2(s) then No:=8
    { сегментный регистр  –  код 9 }
    else if is_sreg(s) then No:=9
    { регистр адреса сегмента команд – код 10 }
    else if s=cs    then No:=10
    { числа  –  код 11 }
    else if (s[1]>='0') and (s[1]<='9') then No:=11
    { имя ячейки памяти  –  код 12 }
    else No:=12;
    { в текущей строке таблицы лексем
```

```
        номеру лексемы присваиваем No }
      lex_table[curr_lex].lex_N:=No;
      { если это ячейка памяти, то имя лексемы урезаем до 7 символов }
      if No=12 then lex_table[curr_lex].Name:=copy(s,1,7)
      { если это число, то имя лексемы равно значению строки }
      else if No =11 then lex_table[curr_lex].Name:=s;
      { переходим к следующей лексеме }
      curr_lex:=curr_lex+1;
end;
{*************************************************************}
```

{Синтаксический анализ строки}

```
procedure string_analize(s:string; i:integer);
var      k,j:byte;
        buf:string;
begin
      { заменяем в строке табуляцию на пробелы }
      while pos(#9,s)<>0 do
          s[pos(#9,s)]:=' ';
      { пока пробел не один, удаляем соседний }
      while pos('  ',s)<>0 do
           delete(s,pos('  ',s),1);
      if (pos(', ',s)<>0) then delete(s, pos(', ',s)+1,1);
      if (pos(' ,',s)<>0) then delete(s, pos(' ,',s),1);
      if (pos(' ',s)=1) then delete(s,1,1);
      { цикл по длине строки }
      j:=0;
      while j<=length(s)-1 do begin
         { к - следующий символ за текущим }
         k:=j+1;
```

```
        { если следующий не пробел и не запятая, то наращиваем к }
        while (not (s[k] in [' ',','])) and (k<=length(s)) do inc(k);
        { если встретили запятую }
        if s[k]=',' then begin
             { buf - одно слово из строки}
             buf:=copy(s,j+1,k-(j+1));
             { получить номер лексемы }
             word_analize(buf,i);
             { получить номер лексемы для запятой }
             buf:=',';
             word_analize(buf,i);
        end else begin
           { если встретили пробел }
           { buf - одно слово из строки}
           buf:=copy(s,j+1,k-(j+1));
           { получить номер лексемы }
           word_analize(buf,i);
        end;
        { перейти к следующему слову в строке }
        j:=k;
      end;
end;
{*************************************************************}
```

{Проверка правильности введенных чисел}

```
function number(i:integer):byte;
var     n:byte;
       code:integer;
       int:longint;
begin
```

```
number:=0;
{ n – длиня имени из строки таблицы лексем }
n:=length(lex_table[i].Name);
{ если в конце стоит 'H' – это шестнадцатеричная константа }
if lex_table[i].Name[n]='H' then begin
    { если цифры не шестнадцатиричные, то выходим }
    for code:=1 to n-1 do
       if not (lex_table[i].Name[code] in ['0'..'9','A'..'F']) then exit;
    { Если длина больше двух байт, то выходим }
    if (n>6) then exit;
    { Если до буквы нет нуля, то выходим }
    if ((lex_table[i].Name[1]>='A') or (n>=6))
       and (lex_table[i].Name[1]<>'0') then exit;
    { number=2, если длина числа два байта }
    if (lex_table[i].Name[1]='0') then
      if (n>4) then number:=2
      { number=1, если длина числа один байт }
      else     number:=1;
    if (lex_table[i].Name[1]<>'0') then
      { number=2, если длина числа два байта }
      if (n>3) then number:=2
      { number=1, если длина числа один байт }
      else     number:=1;
 end
 { если константа десятичная }
 else begin
   { переводим строки в число}
   val(lex_table[i].Name,int,code);
   { если ошибка перевода, то выходим }
```

```
        if code<>0 then exit;
        { number=1, если длина числа один байт }
        if int<256 then number:=1
        { number=2, если длина числа два байта }
        else if int<65536 then number:=2
        { в остальных случаях number=0 }
        else number:=0;
     end;
end;
{*****************************************************************}
```

{Является ли ячейка байтом}

```
function check_already_db(n:integer):boolean;
var i:integer;
begin
   { изначально не совпадают }
   check_already_db:=false;
   { перебираем имена всех лексем }
   for i:=1 to n-1 do begin
      { если имя совпало }
      { и номер следующей лексемы = 2, т.е. байт }
      if (Lex_table[i].Name=Lex_table[n].Name) and
        (Lex_table[i+1].lex_N=2) then begin     {возвращаем true}
                                        check_already_db:=true;
                                        exit;
                                        end;
   end;
end;
{*****************************************************************}
```

{Является ли ячейка словом}

```
function check_already_dw(n:integer):boolean;
var i:integer;
begin
   { изначально не совпадают }
   check_already_dw:=false;
   { перебираем имена всех лексем }
   for i:=1 to n-1 do begin
      { если имя совпало }
      { и номер следующей лексемы = 3, т.е. слово }
      if (Lex_table[i].Name=Lex_table[n].Name) and
        (Lex_table[i+1].lex_N=3) then begin     {возвращаем true}
                                        check_already_dw:=true;
                                        exit;
                                        end;
   end;
end;
{*************************************************************}
```

{Является ли лексема db}

```
function _db(i:integer):boolean;
var      k:byte;
begin
       _db:=false;
       if lex_table[i].lex_N = 12 then
         if lex_table[i+1].lex_N = 2 then
            if (lex_table[i+2].lex_N = 11) and (number(i+2)=1) then
               if (check_already_db(i)=false) and (check_already_dw(i)=false)
                  then _db:=true;
end;
```

```
{*************************************************************}
```

{Является ли лексема dw}

```
function _dw(i:integer):boolean;
begin
        _dw:=false;
        if lex_table[i].lex_N = 12 then
          if lex_table[i+1].lex_N = 3 then
             if (lex_table[i+2].lex_N = 11) and (number(i+2)<>0) then
                if (check_already_db(i)=false) and (check_already_dw(i)=false)
                   then _dw:=true;
end;
{*************************************************************}
```

{Является ли лексема mov}

```
function _mov(i:integer):boolean;
begin
        _mov:=false;
        if (lex_table[i+2].lex_N<>1) or (lex_table[i].lex_N<>4) then exit;
        if (lex_table[i+1].lex_N = 8) then begin
          if lex_table[i+3].lex_N=9 then _mov:=true;
    if (lex_table[i+3].lex_N=12) and (check_already_dw(i+3)) then _mov:=true;
          if lex_table[i+3].lex_N=8 then _mov:=true;
          if (lex_table[i+3].lex_N=11) and (number(i+3)>0) then _mov:=true;
        end;
        if (lex_table[i+1].lex_N = 7) then begin
          if (lex_table[i+3].lex_N=7) then _mov:=true;
     if (lex_table[i+3].lex_N=12) and (check_already_db(i+3)) then _mov:=true;
          if (lex_table[i+3].lex_N=11) and (number(i+3)=1) then _mov:=true;
        end;
        if (lex_table[i+1].lex_N = 9) then begin
```

```
    if (lex_table[i+3].lex_N=12) and (check_already_dw(i+3)) then _mov:=true;
        if (lex_table[i+3].lex_N=8) then _mov:=true;
      end;
      if (lex_table[i+1].lex_N = 12) and (check_already_dw(i+1)) then begin
        if (lex_table[i+3].lex_N in [8..10]) then _mov:=true;
        if (lex_table[i+3].lex_N=11) and (number(i+3)>0) then _mov:=true;
      end;
      if (lex_table[i+1].lex_N = 12) and (check_already_db(i+1)) then begin
        if (lex_table[i+3].lex_N=7) then _mov:=true;
        if (lex_table[i+3].lex_N=11) and (number(i+3)=1) then _mov:=true;
      end;
end;
{*************************************************************}
```

{Является ли лексема mul}

```
function _mul(i:integer):boolean;
begin
      _mul:=false;
      if lex_table[i].lex_N = 6 then begin
        if (lex_table[i+1].lex_N in [7,8]) then
                        _mul:=true;
        if (lex_table[i+1].lex_N = 12) then
          if (check_already_dw(i+1)) or (check_already_db(i+1)) then
                        _mul:=true;
      end;
end;
{*************************************************************}
```

{Является ли лексема pop}

```
function _pop(i:integer):boolean;
begin
```

```
        _pop:=false;
        if lex_table[i].lex_N = 5 then
           if (lex_table[i+1].lex_N >= 8) and (lex_table[i+1].lex_N < 10)
             then _pop:=true
           else if (lex_table[i+1].lex_N=12) and
               (check_already_dw(i+1)) then _pop:=true;
end;
{************************************************************}
```

{Процедура возвращения кода ошибки }

```
procedure synth_analize(var err:integer);
var  i:integer;
    cur_str_N:integer;
begin
    i:=1;      cur_str_N:=1;      err:=0;
    while lex_table[i].str_N<>0 do begin
        if _db(i)=false then
          if _dw(i)=false then
            if _mov(i)=false then
              if _pop(i)=false then
                if _mul(i)=false then begin
                  { номер ошибки }
                  err:=cur_str_N;
                  exit;
                 end;
        while lex_table[i].str_N = cur_str_N do i:=i+1;
        cur_str_N:=cur_str_N+1;
    end;
end;
end.
```