Лекція №14

Приклад опису семантики на простій метасемантичній мові

Розглянемо граматику простого оператора присвоювання:

1. <оператор> → <змінна> := <вираз>
2. <вираз> → <змінна>
3. <вираз> → <вираз> + <змінна>
4. <змінна> → a1
5. <змінна> → a2
6. <змінна> → a3
7. <змінна> → a4
8. <змінна> → a5

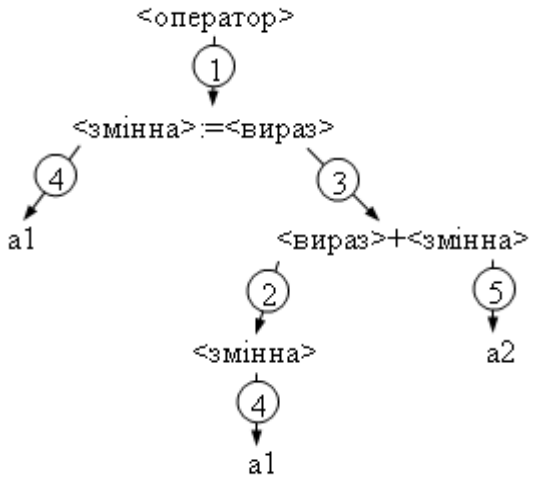
Нехай потрібно виконати трансляцію згідно опису наступної неформальної семантики:

1. Всі змінні мають тип integer;
2. Виходом повинна бути команда асемблера Intel-86;
3. Результат обчислення повинен бути записаний до регістру AX;
4. Відповідність операцій і команд має бути такою:
    1) <змінна>:=<вираз> → MOV_<змінна>,AX;
    2) <змінна> → MOV_AX,<змінна>;
    3) <вираз> + <змінна> → ADD_AX,<змінна>.
    4) id (a1 | a2 | … | a5) → записати конкретний ідентифікатор до внутрішньої змінної Buf, що відповідає нетерміналу <змінна>.

Опишемо формальну семантику (семантичні визначення) для кожного правила граматики:

1. <оператор> → <змінна> := <вираз>	{[1][2]αMOV_Buf,AX}
2. <вираз> → <змінна>	{[1]αMOV_AX,Buf}
3. <вираз> → <вираз> + <змінна>	{[2][1]αADD_AX,Buf}
4. <змінна> → a1	{Buf:=’a1’}
5. <змінна> → a2	{Buf:=’a2’}
6. <змінна> → a3	{Buf:=’a3’}
7. <змінна> → a4	{Buf:=’a4’}
8. <змінна> → a5	{Buf:=’a5’}

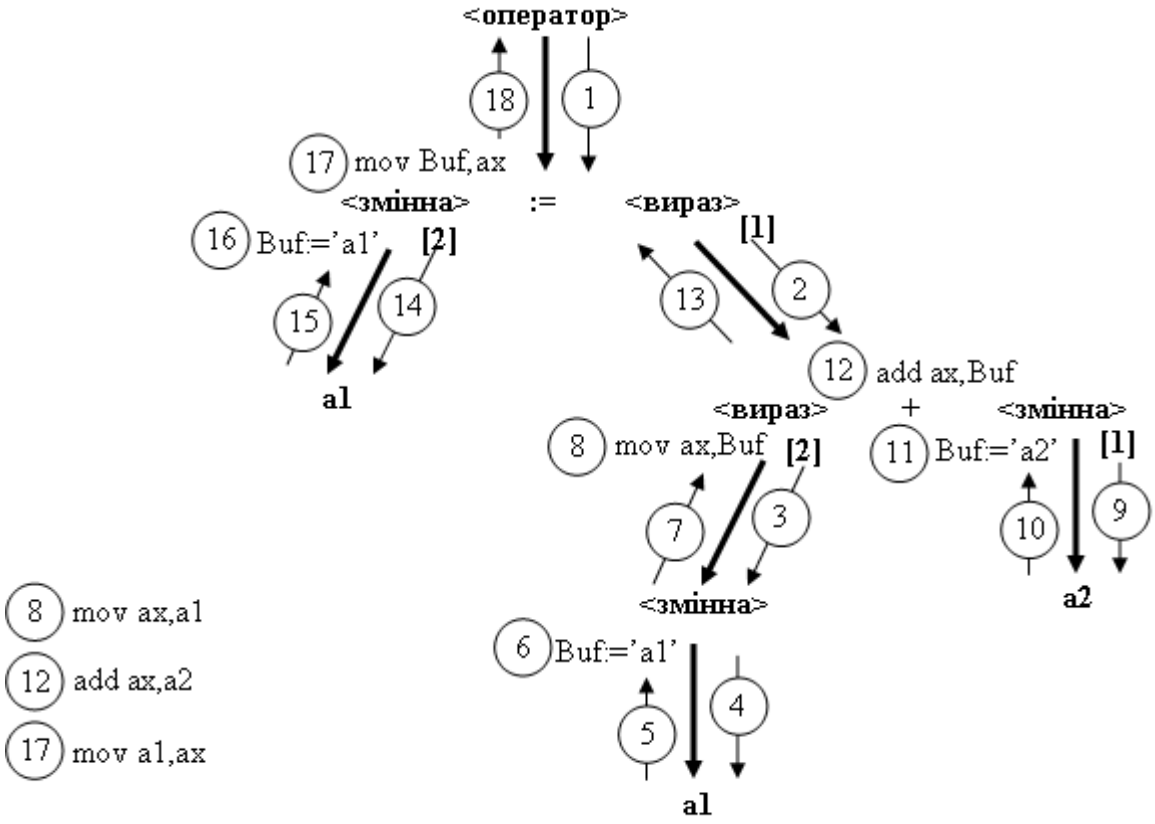
Розглянемо процес трансляції вхідного рядка a1 := a1 + a2 для заданої граматики.
Синтаксичний аналізатор побудує для цього рядка таке дерево розбору (воно, як правило, повністю зберігається в оперативній пам’яті):

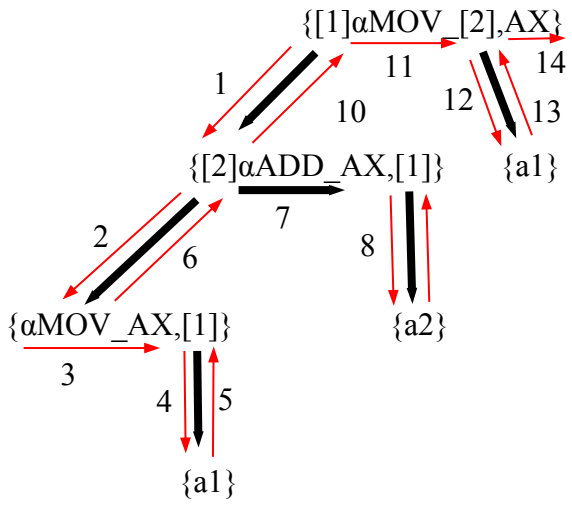


Лінійна форма для цього дерева розбору буде такою:

1	2	3	4	5	6	7	8	9
5	3	-1	2	4	1	-2	4	-6

Розглянемо процес трансляції на схемі дерева розбору, в якій замість правил граматики в вузлах стоять відповідні їм семантичні визначення. На цій схемі цифри в кружках є порядковими номерами дій процесу генерації коду, а не номерами правил, як на попередніх схемах.





Програмування генерації коду на мовах високого рівня

Попередні зауваження:
1. Для рекурсивного спуску по дереву необхідна наявність рекурсивних процедур.
2. Для вибору потрібного семантичного визначення необхідна керуюча конструкція вибору, типу *case* або *switch*.

Структура програми генератора коду (семантичного процесора)

```
Program  CG;
        const  <визначення границь вхідних/вихідних структур даних>
        var  RAS  :  array[1..LR] of integer;
                <опис вихідних і допоміжних даних>
                <опис допоміжних процедур>
                <опис процедур SPR>
        begin
                <початкові установки>
                <ввід вектора RAS>
                SPR(LRAS);
                <вивід результату>
        end.
```

Структура рекурсивної семантичної процедури

Розглянемо структуру рекурсивної семантичної процедури, яка виконує обхід дерева розбору, представленого в лінійній формі з посиланнями.

```
Procedure  SPR (u : integer); (*u – індекс правила чи посилання в векторі RAS*)
        var
                k, (*уточнений індекс вибраного правила в векторі RAS*)

                i, (*номер вибраного синтаксичного правила*)
               k1, k2, …, kn : integer; (*допоміжні змінні*)
        begin
                if  RAS[u] < 0  then k := – RAS[u]
                                else  k := u;
                i := RAS[k];
                case  i  of
                  1:  (*текст синтаксичного правила 1*)
                        <семантичне визначення 1>;
                  2:  (*текст синтаксичного правила 2*)
                        < семантичне визначення 2>;
                        …
                  n: (*текст синтаксичного правила n*)
                        < семантичне визначення n>;
                end; (*case*)
        end; (*SPR*)
```

Для виконання спуску по клаузах (піддеревах) використовуються також виклики:

[1] → SPR(k + 1) або SPR(k1), де k1 = k + 1

[2] → SPR(k + 2) або SPR(k2), де k2 = k + 2

Приклад.

Для граматики ідентифікатора описати семантику, яка копіює вхід на вихід.

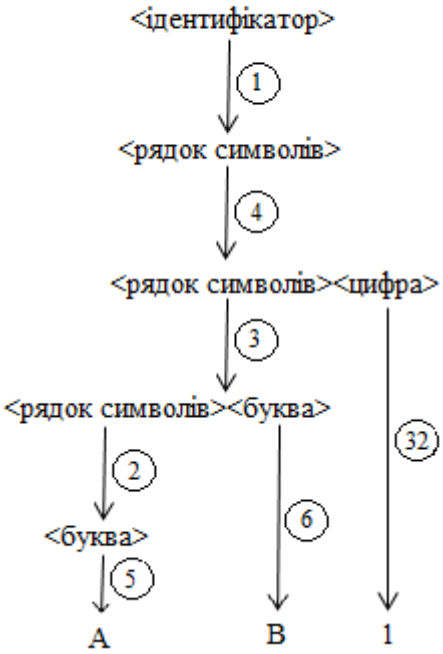
Граматика ідентифікатора задана наступними правилами:

1. <ідентифікатор> → <рядок символів> {[1]}
2. <рядок символів> → <буква> {[1]}
3. < рядок символів> → < рядок символів> <буква> {[2][1]}
4. < рядок символів> → < рядок символів> <цифра> {[2][1]}

5÷30. <буква> → A| B| C| ..| Z {STR:=STR+<буква>}

31÷40. <цифра> → 0| 1| 2| 3| 4| 5| 6| 7| 8| 9 {STR:=STR+<цифра>}

AB1



Вхід: дерево розбору в лінійній формі з посиланнями (вектор RAS).

Вихід: рядок STR, який містить ідентифікатор.

Процедура SPR, яка виконує трансляцію буде такою:

```
Procedure  SPR (u  :  integer);
        var  i,k,k1,k2  :  integer;
        begin
                if  RAS[u]<0  then  k := – RAS[u]
                                else  k := u;
                i := RAS[k];
                k1 := k + 1;
                k2 := k + 2;
                case  i  of
                  1: (*<ідентифікатор> → < рядок символів>*)
                        SPR (k1);                {[1]}
                  2: (*<рядок символів> → <буква>*)
                        SPR(k1);                 {[1]}
                  3: (*<рядок символів> → < рядок символів> <буква>*)
                        begin
                          SPR(k2);
                          SPR(k1)                {[2][1]}
                        end;
                  4: (*<рядок символів> → < рядок символів> <цифра>*)
                        begin
                          SPR(k2);
                          SPR(k1)                {[2][1]}
                        end;
                  5..30: (*<буква> → A, B, …, Z*)
                        STR := STR + CHR(ORD('A') – 5 +i);
                 31..40: (*<цифра> → 0, 1…, 9*)
                        STR := STR + CHR(ORD('0') – 31 + i);
                end;
        end;
```

Методи скорочення тексту семантичних визначень

Ці методи засновані на властивостях керуючої конструкції вибору.

1. Якщо тексти семантичних визначень співпадають, то їх можна об'єднати в одній гілці конструкції вибору.

```
1, 2 : (*…*)                    switch(i) {
       SPR(k1);                 case 3: case 4: SPR(k2);
3, 4 : (*…*)                    case 1: case 2: SPR(k1);
       Begin                    }
         SPR(k2);
         SPR(k1);
       end;
```

2. Якщо семантичні визначення співпадають частково, то їх можна об'єднати при деякій умові.

```
1, 2, 3, 4: (*…*)
       begin
         if (i = 3) or (i = 4) then SPR(k2);
         SPR(k1)
       end;
```

3. Можна також об'єднати семантичні визначення, використовуючи допоміжну структуру даних чи функцію від номеру правила.

   Наприклад, можна описати масив символів Symbols.

	5	6		30	31	32		40
Symbols	A	B	…	Z	0	1	…	9

   При його використанні семантичні визначення правил 5 ÷ 40 будуть однакові.

```
5 … 40 : (*…*)
       STR:=STR + Symbols[i];
```

Головна програма генератора коду буде мати такий вигляд:

```
program CG;
    const  LR = 50;
    var    RAS : array[1..LR] of integer;
           STR : string;
           LRAS : integer;
           Symbols : array[5..40] of char;
    <описание SPR>
    begin
           STR := ' ';
           <введення Symbols>;
           < введення LRAS>;
           < введення RAS>;
           SPR(LRAS);
           <виведення STR>
    end.
```