5.3Загальна перевірка коректності роботи транслятора

Для того щоб здійснити перевірку коректності роботи транслятора необхідно завантажити коректну до заданої вхідної мови програму.

***Текст коректної програми***

??Prog1

MAIMPROGRAM

DATA INTEGER \_A,\_B,\_X,\_Y;

START

PUT("Input A: ");

GET(\_A);

PUT("Input B: ");

GET(\_B);

PUT("A + B: ");

PUT(\_A + \_B);

PUT("\nA - B: ");

PUT(\_A - \_B);

PUT("\nA \* B: ");

PUT(\_A \* \_B);

PUT("\nA / B: ");

PUT(\_A DIV \_B);

PUT("\nA % B: ");

PUT(\_A MOD \_B);

\_X==>(\_A - \_B) \* 10 + (\_A + \_B) DIV 10;

\_Y==>\_X + (\_X MOD 10);

PUT("\nX = (A - B) \* 10 + (A + B) / 10\n");

PUT(\_X);

PUT("\nY = X + (X MOD 10)\n");

PUT(\_Y);

END

Оскільки дана програма відповідає граматиці то результати виконання лексичного, синтаксичного аналізів, а також генератора коду будуть позитивними.

В результаті буде отримано асемблерний файл, який є результатом виконання трансляції з заданої вхідної мови на мову Assembler даної програми (його вміст наведений в Додатку А).

Після виконання компіляції даного файлу на виході отримаєм наступний результат роботи програми:

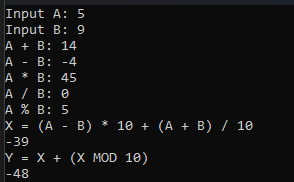


Рис. 5.1 Результат виконання коректної програми

При перевірці отриманого результату, можна зробити висновок про правильність роботи програми, а отже і про правильність роботи транслятора.

**5.4** **Тестова програма №1**

***Текст програми***

??Prog1

MAIMPROGRAM

DATA INTEGER \_A,\_B,\_X,\_Y;

START

PUT("Input A: ");

GET(\_A);

PUT("Input B: ");

GET(\_B);

PUT("A + B: ");

PUT(\_A + \_B);

PUT("\nA - B: ");

PUT(\_A - \_B);

PUT("\nA \* B: ");

PUT(\_A \* \_B);

PUT("\nA / B: ");

PUT(\_A DIV \_B);

PUT("\nA % B: ");

PUT(\_A MOD \_B);

\_X==>(\_A - \_B) \* 10 + (\_A + \_B) DIV 10;

\_Y==>\_X + (\_X MOD 10);

PUT("\nX = (A - B) \* 10 + (A + B) / 10\n");

PUT(\_X);

PUT("\nY = X + (X MOD 10)\n");

PUT(\_Y);

END

***Результат виконання***



Рис. 5.2 Результат виконання тестової програми №1

5.5 Тестова програма №2

***Текст програми***

??Prog2

MAIMPROGRAM

DATA INTEGER \_A,\_B,\_C;

START

PUT("Input A: ");

GET(\_A);

PUT("Input B: ");

GET(\_B);

PUT("Input C: ");

GET(\_C);

IF(\_A GT \_B)

START

IF(\_A GT \_C)

START

GOTO \_TEMP;

END

ELSE

START

PUT(\_C);

GOTO \_OUT;

\_TEMP:

PUT(\_A);

GOTO \_OUT;

END

END

IF(\_B LT \_C)

START

PUT(\_C);

END

ELSE

START

PUT(\_B);

END

\_OUT:

PUT("\n");

IF((\_A == \_B) AND (\_A == \_C) AND (\_B == \_C))

START

PUT(1);

END

ELSE

START

PUT(0);

END

PUT("\n");

IF((\_A LT 0) OR (\_B LT 0) OR (\_C LT 0))

START

PUT(- 1);

END

ELSE

START

PUT(0);

END

PUT("\n");

IF(!!(\_A LT (\_B + \_C)))

START

PUT(10);

END

ELSE

START

PUT(0);

END

END

***Результат виконання***

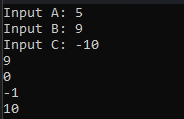


Рис. 5.3 Результат виконання тестової програми №2

5.6 Тестова програма №3

***Текст програми***

??Prog3

MAIMPROGRAM

DATA INTEGER \_A,\_A2,\_B,\_X,\_C1,\_C2;

START

PUT("Input A: ");

GET(\_A);

PUT("Input B: ");

GET(\_B);

PUT("FOR TO DO");

FOR \_A2==>\_A TO \_B DO

START

PUT("\n");

PUT(\_A2 \* \_A2);

END

PUT("\nFOR DOWNTO DO");

FOR \_A2==>\_B DOWNTO \_A DO

START

PUT("\n");

PUT(\_A2 \* \_A2);

END

PUT("\nWHILE A \* B: ");

\_X==>0;

\_C1==>0;

WHILE(\_C1 LT \_A)

START

\_C2==>0;

WHILE (\_C2 LT \_B)

START

\_X==>\_X + 1;

\_C2==>\_C2 + 1;

END

\_C1==>\_C1 + 1;

END

PUT(\_X);

PUT("\nREPEAT UNTIL A \* B: ");

\_X==>0;

\_C1==>1;

REPEAT

\_C2==>1;

REPEAT

\_X==>\_X + 1;

\_C2==>\_C2 + 1;

UNTIL(!!(\_C2 GT \_B))

\_C1==>\_C1 + 1;

UNTIL(!!(\_C1 GT \_A))

PUT(\_X);

END

***Результат виконання***

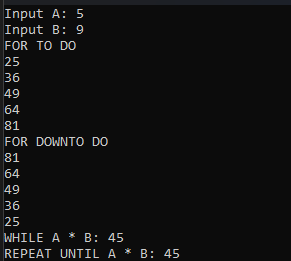


Рис. 5.4 Результат виконання тестової програми №3