**2.2 Опис термінальних символів та ключових слів**

Визначимо окремі термінальні символи та нерозривні набори термінальних символів (ключові слова):

|  |  |
| --- | --- |
| Термінальний символ або ключове слово | Значення |
| #PROGRAM | Початок програми |
| START | Початок тексту програми |
| VARIABLE | Початок блоку опису змінних |
| STOP | Кінець розділу операторів |
| GET | Оператор вводу змінних |
| PUT | Оператор виводу (змінних або рядкових констант) |
| <- | Оператор присвоєння |
| IF | Оператор умови |
| ELSE | Оператор умови |
| GOTO | Оператор переходу |
| LABEL | Мітка переходу |
| FOR | Оператор циклу |
| TO | Інкремент циклу |
| DOWNTO | Декремент циклу |
| DO | Початок тіла циклу |
| WHILE | Оператор циклу |
| CONTINUE | Оператор циклу |
| EXIT | Оператор циклу |
| REPEAT | Початок тіла циклу |
| UNTIL | Оператор циклу |
| + | Оператор додавання |
| - | Оператор віднімання |
| \* | Оператор множення |
| DIV | Оператор ділення |
| MOD | Оператор знаходження залишку від ділення |
| = | Оператор перевірки на рівність |
| <> | Оператор перевірки на нерівність |
| <= | Оператор перевірки чи менше |
| >= | Оператор перевірки чи більше |
| ! | Оператор логічного заперечення |
| AND | Оператор кон’юнкції |
| OR | Оператор диз’юнкції |
| INTEGER32 | 32‑ох розрядні знакові цілі |
| !!… | Коментар |
| , | Розділювач |
| ; | Ознака кінця оператора |
| ( | Відкриваюча дужка |
| ) | Закриваюча дужка |

До термінальних символів віднесемо також усі цифри (0-9), латинські букви (a-z, A-Z), символи табуляції, символ переходу на нову стрічку, пробілу.

5.3 Загальна перевірка коректності роботи транслятора

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

***Текст коректної програми***

!!Prog1

#PROGRAM SPPROGRAM1;

START

VARIABLE INTEGER32 AAAAAAAAAA,BBBBBBBBBB,XXXXXXXXXX,YYYYYYYYYY;

PUT("GET AAAAAAAAAA: ");

GET(AAAAAAAAAA);

PUT("GET BBBBBBBBBB: ");

GET(BBBBBBBBBB);

PUT("AAAAAAAAAA + BBBBBBBBBB: ");

PUT(AAAAAAAAAA + BBBBBBBBBB);

PUT("\nAAAAAAAAAA - BBBBBBBBBB: ");

PUT(AAAAAAAAAA - BBBBBBBBBB);

PUT("\nAAAAAAAAAA \* BBBBBBBBBB: ");

PUT(AAAAAAAAAA \* BBBBBBBBBB);

PUT("\nAAAAAAAAAA / BBBBBBBBBB: ");

PUT(AAAAAAAAAA DIV BBBBBBBBBB);

PUT("\nAAAAAAAAAA % BBBBBBBBBB: ");

PUT(AAAAAAAAAA MOD BBBBBBBBBB);

XXXXXXXXXX==>(AAAAAAAAAA - BBBBBBBBBB) \* 10 + (AAAAAAAAAA + BBBBBBBBBB) DIV 10;

YYYYYYYYYY==>XXXXXXXXXX + (XXXXXXXXXX MOD 10);

PUT("\nXXXXXXXXXX = (AAAAAAAAAA - BBBBBBBBBB) \* 10 + (AAAAAAAAAA + BBBBBBBBBB) / 10\n");

PUT(XXXXXXXXXX);

PUT("\nYYYYYYYYYY = XXXXXXXXXX + (XXXXXXXXXX % 10)\n");

PUT(YYYYYYYYYY);

STOP

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:

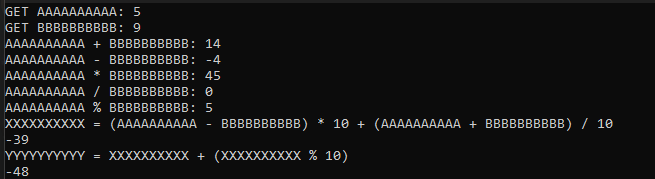


Рис. 5.1 Результат виконання коректної програми

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

Тестова програма №1

***Текст програми***

!!Prog1

#PROGRAM SPPROGRAM1;

START

VARIABLE INTEGER32 AAAAAAAAAA,BBBBBBBBBB,XXXXXXXXXX,YYYYYYYYYY;

PUT("GET AAAAAAAAAA: ");

GET(AAAAAAAAAA);

PUT("GET BBBBBBBBBB: ");

GET(BBBBBBBBBB);

PUT("AAAAAAAAAA + BBBBBBBBBB: ");

PUT(AAAAAAAAAA + BBBBBBBBBB);

PUT("\nAAAAAAAAAA - BBBBBBBBBB: ");

PUT(AAAAAAAAAA - BBBBBBBBBB);

PUT("\nAAAAAAAAAA \* BBBBBBBBBB: ");

PUT(AAAAAAAAAA \* BBBBBBBBBB);

PUT("\nAAAAAAAAAA / BBBBBBBBBB: ");

PUT(AAAAAAAAAA DIV BBBBBBBBBB);

PUT("\nAAAAAAAAAA % BBBBBBBBBB: ");

PUT(AAAAAAAAAA MOD BBBBBBBBBB);

XXXXXXXXXX==>(AAAAAAAAAA - BBBBBBBBBB) \* 10 + (AAAAAAAAAA + BBBBBBBBBB) DIV 10;

YYYYYYYYYY==>XXXXXXXXXX + (XXXXXXXXXX MOD 10);

PUT("\nXXXXXXXXXX = (AAAAAAAAAA - BBBBBBBBBB) \* 10 + (AAAAAAAAAA + BBBBBBBBBB) / 10\n");

PUT(XXXXXXXXXX);

PUT("\nYYYYYYYYYY = XXXXXXXXXX + (XXXXXXXXXX % 10)\n");

PUT(YYYYYYYYYY);

STOP

***Результат виконання***

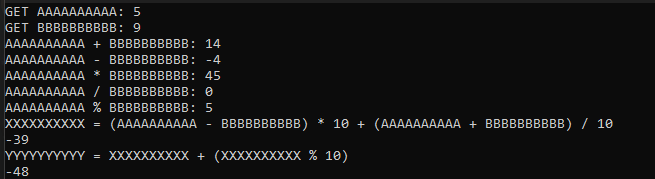


Рис. 5.2 Результат виконання тестової програми №1

Тестова програма №2

***Текст програми***

!!Prog2

#PROGRAM SPPROGRAM2;

START

VARIABLE INTEGER32 AAAAAAAAAA,BBBBBBBBBB,CCCCCCCCCC;

PUT("GET AAAAAAAAAA: ");

GET(AAAAAAAAAA);

PUT("GET BBBBBBBBBB: ");

GET(BBBBBBBBBB);

PUT("GET CCCCCCCCCC: ");

GET(CCCCCCCCCC);

IF(AAAAAAAAAA >= BBBBBBBBBB)

START

IF(AAAAAAAAAA >= CCCCCCCCCC)

START

GOTO AISBIGGER3;

STOP

ELSE

START

PUT(CCCCCCCCCC);

GOTO OUTOFIFGOT;

AISBIGGER3:

PUT(AAAAAAAAAA);

GOTO OUTOFIFGOT;

STOP

STOP

IF(BBBBBBBBBB <= CCCCCCCCCC)

START

PUT(CCCCCCCCCC);

STOP

ELSE

START

PUT(BBBBBBBBBB);

STOP

OUTOFIFGOT:

PUT("\n");

IF((AAAAAAAAAA = BBBBBBBBBB) AND (AAAAAAAAAA = CCCCCCCCCC) AND (BBBBBBBBBB = CCCCCCCCCC))

START

PUT(1);

STOP

ELSE

START

PUT(0);

STOP

PUT("\n");

IF((AAAAAAAAAA <= 0) OR (BBBBBBBBBB <= 0) OR (CCCCCCCCCC <= 0))

START

PUT(-1);

STOP

ELSE

START

PUT(0);

STOP

PUT("\n");

IF(!(AAAAAAAAAA <= (BBBBBBBBBB + CCCCCCCCCC)))

START

PUT(10);

STOP

ELSE

START

PUT(0);

STOP

STOP

***Результат виконання***

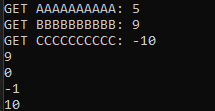


Рис. 5.3 Результат виконання тестової програми №2

Тестова програма №3

***Текст програми***

!!Prog3

#PROGRAM SPPROGRAM3;

START

VARIABLE INTEGER32 AAAAAAAAAA,AAAAAAAAA2,BBBBBBBBBB,XXXXXXXXXX,CCCCCCCCC1,CCCCCCCCC2;

PUT("GET AAAAAAAAAA: ");

GET(AAAAAAAAAA);

PUT("GET BBBBBBBBBB: ");

GET(BBBBBBBBBB);

PUT("FOR TO DO");

FOR AAAAAAAAA2==>AAAAAAAAAA TO BBBBBBBBBB DO

START

PUT("\n");

PUT(AAAAAAAAA2 \* AAAAAAAAA2);

STOP

PUT("\nFor DOWNTO DO");

FOR AAAAAAAAA2==>BBBBBBBBBB DOWNTO AAAAAAAAAA DO

START

PUT("\n");

PUT(AAAAAAAAA2 \* AAAAAAAAA2);

STOP

PUT("\nWhile AAAAAAAAAA \* BBBBBBBBBB: ");

XXXXXXXXXX==>0;

CCCCCCCCC1==>0;

WHILE(CCCCCCCCC1 <= AAAAAAAAAA)

START

CCCCCCCCC2==>0;

WHILE (CCCCCCCCC2 <= BBBBBBBBBB)

START

XXXXXXXXXX==>XXXXXXXXXX + 1;

CCCCCCCCC2==>CCCCCCCCC2 + 1;

STOP WHILE

CCCCCCCCC1==>CCCCCCCCC1 + 1;

STOP WHILE

PUT(XXXXXXXXXX);

PUT("\nRepeat UNTIL AAAAAAAAAA \* BBBBBBBBBB: ");

XXXXXXXXXX==>0;

CCCCCCCCC1==>1;

REPEAT

CCCCCCCCC2==>1;

REPEAT

XXXXXXXXXX==>XXXXXXXXXX+1;

CCCCCCCCC2==>CCCCCCCCC2+1;

UNTIL(!(CCCCCCCCC2 >= BBBBBBBBBB))

CCCCCCCCC1==>CCCCCCCCC1+1;

UNTIL(!(CCCCCCCCC1 >= AAAAAAAAAA))

PUT(XXXXXXXXXX);

STOP

***Результат виконання***

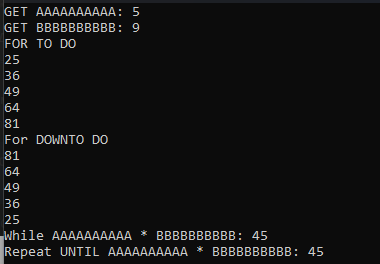


Рис. 5.4 Результат виконання тестової програми №3