Tema 6 - Realizarea compilarii din C++ în limbaj de asamblare cu ajutorul Flex și Bison

1. Specificarea minilimbajului de programare (MLP)

1.1. Tipuri de date simple și tip de date definit de utilizator

1. **Integer** - tip de date pentru numere întregi (`int`)
2. **Char** - tip de date pentru caractere (`char`)

1.2. Instructiuni

1. **Instrucțiune de atribuire**:
    - Se realizează prin intermediul operatorului de atribuire "=" (ex: `a = 5;`)
2. **Instrucțiune de intrare/ieșire**:
    - Se realizează prin intermediul:
        - Funcției de citire: `cin >> nume_variabilă;` (ex: `cin >> a;`)
        - Funcției de afișare: `cout << nume_variabilă;` (ex: `cout << a;`)
        - Functia de returnare: `return nume_variabilă;` (ex: `return a;`) - se folosește pentru a returna o valoare dintr-o funcție
3. **Instrucțiune de selecție (condițională)**:
    - Se realizează prin intermediul:
    - conftii de egalitate
    - `if (condiție) { // bloc de instrucțiuni doar cu assignments }` (ex: `if (a == 5) { cout << a; }`)
    - `if (condiție) { // bloc de instrucțiuni doar cu assignments } else { // bloc de instrucțiuni doar cu assignments }` (ex: `if (a > 5) { cout << a; } else { cout << b; }`)

Restricții:

- **Identificatori**:
    - Pot conține litere, cifre și underscore (_), dar nu pot începe cu o cifră.
    - Sunt case sensitive.
    - Nu pot fi cuvinte cheie.
- **Pot returna decat in functii de tip int sau float**
- **Pot avea un singur return in functie**

1.3. Analiza lexicală

1.3.1. Simboluri:

- **Operatori**:
  - Aritmetici: `+`, `-`, `*`, `/`, `%`
  - De comparație: `==`, `!=`
  - De atribuire: `=`
  - De separare: `,`, `;`
- **Delimitatori**: `(`, `)`, `{`, `}`, `;`
- **Cuvinte cheie**:
  - `int`
  - `char`
  - `string`
  - `if`
  - `cin >>`
  - `cout <<`
  - `else`
  - `return`

1.3.2. Identificatori:

- **Definiție**:
  - `identifier = letter (letter | digit | _)*`
  - `letter = a | b | ... | z | A | B | ... | Z`
  - `digit = 0 | 1 | ... | 9`

1.3.3. Tabel de codificare (TC):

Atom	ID
ID	0
CONST	1
(	2
)	3
,	4
int	5
char	6
main	7
string	8
{	9
}	10

Atom	ID
=	11
+	12
-	13
*	14
/	15
%	16
==	17
!=	18
!	19
;	20
cin	21
cout	22
if	23
else	24
<<	25
>>	26
return	27
#include	28

1.4 Forma BNF a MLP-ului

```
<program> ::= <header> <declaration_or_function_list>
<header> ::= <library_inclusions>
<library_inclusions>::= <include> <io_stream> | <include> <math_h>
<include> ::= "#include"
<io_stream> ::= "iostream"
<declaration_or_function_list> ::= <declaration_or_function_list>
<declaration_or_function> | <declaration_or_function>
<declaration_or_function> ::= <function> | <variable_declaration>
<variable_declaration> ::= <simple_type> ID ";"
<function> ::= <simple_type> ID "(" <parameter_list>? ")" <body>
<parameter_list> ::= <parameter_list> "," <parameter> | <parameter> | epsilon  ;
<parameter> ::= <simple_type> ID  ;
<body> ::= { <instruction_list>  }  ;
<instruction_list> ::= <instruction_list> <instruction> | <instruction> ;
<instruction> ::= <variable_declaration> | <assignment> | <input_instruction> |
```

```
<output_instruction> | <if_statement> | <return_statement> ;
<input_instruction> ::= cin >> ID ";" ;
<output_instruction> ::= cout << <arithmethic_expression> ";" ;
<assignments> ::= <assignemnt> | <assignments> <assignment> ;
<if_body> ::= { <assignments> } ;
<assignment> ::= ID "=" <arithmethic_expression> ";" ;
<if_stament> ::= if ( <condition> ) <if_body>  else <if_body>  ;
<return_statement> :: = "return" <arithmethic_expression> ";"
<condition> ::= <term> EQ <term> | <term> NE <term> ;
<arithmenthic_expression>::=
    <arithmethic_expression> PLUS <term> |
    <arithmethic_expression> MINUS <term> |
    <arithmethic_expression> MULTIPLY <term> |
    <arithmethic_expression> DIVIDE <term> |
    <term> ;
<term> ::= CONST_INT | ID | "(" <arithmethic_expression> ")" ;
<simple_type> ::= "int" | "char" | "string" ;
```