Projet de compilation avec les Outils FLEX et Bison

1. Introduction

Le but de ce projet de compilation est de réaliser un mini-compilateur permettant de prendre en charge les différentes étapes de compilation à savoir l'analyse lexicale en utilisant l'outil FLEX et l'analyse syntaxico-sémantique en utilisant l'outil BISON, génération du code intermédiaire, optimisation et la génération du code objet. Les traitements parallèles concernant la gestion de la table des symboles ainsi que le traitement des différentes erreurs doivent être également réalisés lors des différentes phases d'analyse du processus de compilation.

2. Description du Langage

La structure générale d'un programme est la suivante :

Fonction_1
Fonction_2
…
Fonction_n
Programme_principal

Programme_principal	Fonction_i
PROGRAM nom_programme_principal % Déclarations % Instructions **END**	type **ROUTINE** nom_fonction (liste_paramètres) % Déclarations % Instructions nom_fonction = valeur retournée **ENDR**

Commentaire : Un commentaire est ignoré par le compilateur, il doit commencer par « % » il se termine par un ;
Exemple :
…
% Cette ligne est ignorée par le compilateur ;
…
Liste_paramètres **:** C'est un ensemble de variables séparées par une virgule dont leur type est défini au niveau du programme principal.

Exemple :
type ROUTINE nom_fonction (nom_variable3, nom_variable4)
… ENDR
PROGRAM nom_programme_principal type nom_variable1
type nom_variable2 ;
nom_variable2 = CALL nom_fonction (nom_variable1, nom_variable2) ;
END

3. Déclarations

Exemple :

type nom_variable1 ;	*% déclaration d'une variable simple*
type nom_variable2, nom_variable3 ;	*% déclaration des variables de même type*
type nom_variable4 DIMENSION (20) ;	*% déclaration d'un tableau*
type nom_variable5 DIMENSION (20, 200) ;	*% déclaration d'une matrice*

3.1 Type : Le type peut avoir les valeurs suivantes : INTEGER, CHARACTER, REAL, LOGICAL.

	INTEGER	**REAL**	**LOGICAL**	**CHARACTER**
Intervalle des valeurs possibles	-32768 à 32767	INTEGER.INTEGER	TRUE, FALSE	CHARACTER nom_variable1 (un caractère) CHARACTER nom_variable2*20 (une chaine de taille 20)
Exemple	-2084, 1524	8753.012, 0.24		"c", "compilation"

3.2 Identificateur :

Un identificateur doit :

- Commencer par une lettre suivit par une séquence de lettres et de chiffres.
- Il ne doit pas contenir plus de 10 caractères.
- Si la taille d'un identificateur est supérieure à 10, un message Warning doit être affiché.
- Le compilateur fait différence entre la majuscule et minuscule.

Le nom du programme principal, des routines, des étiquettes et des variables sont des identificateurs.

3.3 **Instructions :** chaque instruction doit se terminer par un ;

Affectation	
Description	Exemple
nom_variable = expression;	… INTEGER A, B CHARACTER C*20; LOGICAL D; INTEGER MAT DIMENSION (30,10); … B = 30; A = MAT(5,4) + (B * 5); C = "L3"; D = TRUE; …

Entrées / Sorties	
Description	Exemple
Entrée : **READ (nom_variable);** Sortie : **WRITE ("…", nom_variable, "…");**	… WRITE ("Donner la valeur de A :"); READ (A); WRITE ("La Valeur de A est ", A, "."); …

Condition (Si … Alors … Sinon … Fin Si**)**	
Description	Exemple

IF (expression conditionnelle) THEN
Instruction_1 ;
Instruction_2;
…
Instruction_n;
ELSE
Instruction_n+1;
Instruction_n+2;
…
Instruction_m;
ENDIF
	…
	IF (((A.EQ.(B+1)).OR.(C.EQ.TRUE))) THEN A = A – B ;
	ELSE
	A = A + B;
	ENDIF
	…

Boucle (Tant que … Fin Tant que**)**	
Description	Exemple
DOWHILE (expression conditionnelle) **Instructions** **ENDDO**	… A = 0; DOWHILE (A.LT.10) A = A + 1; ENDDO; …

Appel d'une function	
Description	Exemple
nom_variable = CALL nom_fonction (liste_paramètres);	PROGRAM nom_programme_principal … A = CALL nom_routine (liste_paramètres); …

Equivalence (Partage mémoire**)**	
Description	Exemple
EQUIVALENCE (liste_variables), (liste_variables);	… EQUIVALENCE (A, B, C), (Y, TAB(3)); % A, B, C auront la même adresse mémoire % Y, TAB(3) auront la même adresse mémoire …

Associativité et priorité des opérateurs :

Les associativités et les priorités des opérateurs sont données par la table suivante par ordre croissant :

Opérateur		**Associativité**
Opérateurs Logiques	OR (ou)	gauche
	AND (et)	gauche
Opérateurs de comparaison	GT (>) GE (>=) EQ (==) NE (!=) LE (<=) LT (<)	gauche
Opérateurs Arithmétiques	+ -	gauche
	* /	gauche

4 Analyse Lexicale avec l'outil FLEX :

Son but est d'associer à chaque mot du programme source la catégorie lexicale à laquelle il appartient. Pour cela, il est demandé de définir les différentes entités lexicales à l'aide d'expressions régulières et de générer le programme FLEX correspondant.

5 Analyse syntaxico-sémantique avec l'outil BISON :

Pour implémenter l'analyseur syntaxico-sémantique, il va falloir écrire la grammaire qui génère ce langage. La grammaire associée doit être LALR. En effet l'outil BISON est un analyseur ascendant qui opère sur des grammaires LALR. Il faudra spécifier dans le fichier BISON les différentes règles de la grammaire ainsi que les règles de priorités pour les opérateurs afin de résoudre les conflits. Les routines sémantiques doivent être associées aux règles dans le fichier BISON.

6 Gestion de la table de symboles :

La table de symboles doit être créée lors de la phase de l'analyse lexicale. Elle doit regrouper l'ensemble

des variables et constantes définies par le programmeur avec toutes les informations nécessaires pour le processus de compilation. Cette table sera mises à jour au fur et à mesure de l'avancement de la compilation. Il est demandé de prévoir des procédures pour permettre de **recherche** et d'**insérer** des éléments dans la table de symboles (une liste). Les variables structurées de type tableau doivent aussi figurer dans la table de symboles.

7 Traitement des erreurs :

Il est demandé d'afficher les messages d'erreurs adéquats à chaque étape du processus de compilation. Ainsi, lorsqu'une erreur lexicale ou syntaxique est détectée par votre compilateur, elle doit être signalée le plus précisément possible, par sa nature et sa localisation dans le fichier source. On adoptera le format suivant pour cette signalisation :

File "Test", line 4, character 56: syntax error