Davy Nil
Besson Thomas

Livrable 1
Analyse Lexicale
Projet Informatique : Assembleur MIPS
SEI 2018 - 2019

1. Introduction

L'analyse lexicale est la première étape qu'un compilateur effectue. Elle permet de de séparer le programme en plusieurs parties puis de les identifier. Dans le cas de ce projet, la partie Analyse Lexicale de l'assembleur MIPS à concevoir doit pouvoir reconnaître les différents lexèmes et de gérer les premières erreurs du fichier contenant le programme source.

2. Découpage des mots

La première tâche à réaliser de cette analyse lexicale est de lire le fichier source et de le découper en petites parties. Pour ce faire, nous avons utilisé la fonction strtok(). Celle-ci renvoie une chaîne de caractères délimitée par seulement des espaces. Or dans l'assembleur MIPS, il n'y a pas que l'espace qui est considéré comme un délimiteur de mots. Il y a également '/', '.', '(' ,')', ':', '-', '+', '#', '\t', … C'est pourquoi et avant de commencer l'étiquetage d'un mot, il faut vérifier qu'il ne contient aucun délimiteur du langage MIPS.

Pseudo code :

Ouverture du fichier
Acquisition d'une chaine de caractère
Tant que la chaîne de caractère acquise n'est pas entièrement traitée
    Vérification des délimiteurs
    S'il y en a : séparation de la chaîne en deux mots
    Etiquetage de la première partie
    La seconde partie contient de le reste de la chaîne de caractère à traiter

3. Etiquetage des mots

Après avoir découpé toutes les chaînes de caractères en mots ne comprenant plus de délimiteurs, il est maintenant possible d'affecter les différents lexèmes au code. Pour ce faire, nous avons réalisé une machine à états finis qui se traduit dans le programme par un switch case sur le premier caractère de chaque mot.

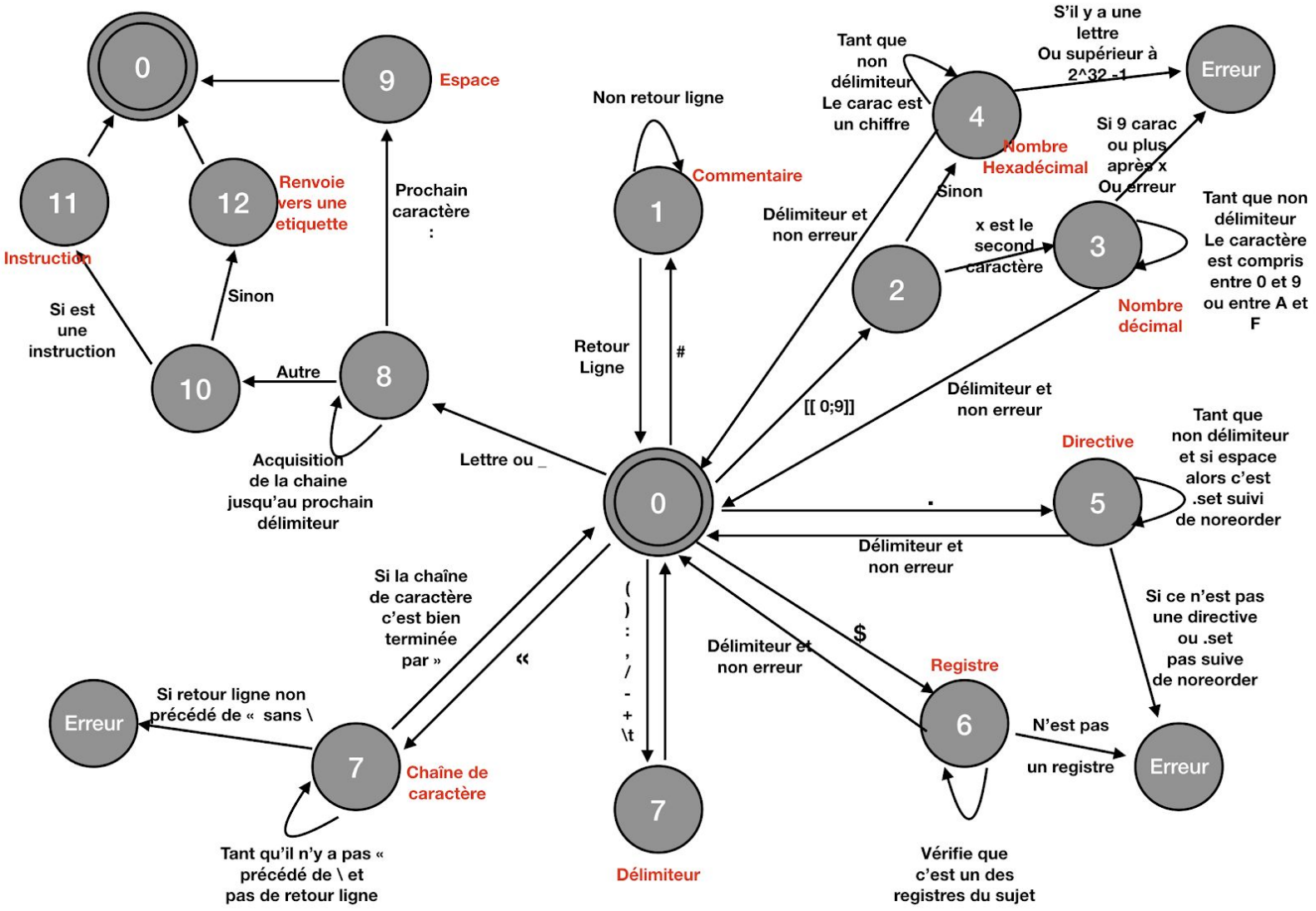

Figure 1 : Machine à états finis

Ce code permet de gérer les premières erreurs de code comme des nombres trop grands ou des erreurs de syntaxe. A la fin de l'analyse lexicale, le programme contient 2 files de la structure : cellule1 (file.h) , l'une contenant l'analyse lexicale et l'autre contenant les erreurs du programme. Ce programme écrit l'analyse lexicale et les erreurs dans 2 fichiers textes extérieurs. Les erreurs sont également inscrites dans le terminal.

4. Quelques cas particuliers

a - Si le premier caractère est un numéro

Dans ce cas, il y a deux possibilités soit c'est un nombre décimal soit c'est un nombre hexadécimal, tout dépend du second caractère. De plus ici, les premières erreurs sont gérées (tout ce qui concerne les nombres plus grand que $2^{32}-1$ ou encore la présence de lettres n'étant pas hexadécimale).

Pseudo Code:

Si premier caractère est un nombre
    Si tous les caractères suivant sont des nombres
        C'est un nombre
        S'il est supérieur à 2^32-1 =>erreur
        Sinon =>enfile
    Sinon
        Si le second caractère n'est pas un 'x' =>erreur
        Sinon
            S'il contient 8 caractères après le 'x' (car 32 bits) et que ceux ci sont compris entre 0 et 9 et a et f => enfile
            Sinon => erreur

b - Registre et Instruction

Pour pouvoir gagner du temps et éviter une comparaison à tous les instructions ou tous les registres un par un, deux tables de hachages ont été mise en oeuvre. Les noms des registres et instructions ont été écrit dans des fichiers externes afin de remplir ces tables.

d - Cas des directives

La directive .set a été séparée des autres directives. En effet, après celle-ci il ne peut y avoir seulement la chaîne de caractère noreorder.

Pseudo-code

Si c'est une directive
    Si c'est .set
        Si le mot après n'est pas noreorder => erreur
        Sinon => enfile
    Sinon verif des autres directives
        Si elle en fait partie => enfile
        Sinon => erreur

5. Limites du programme

Une chaîne de caractère ne peut qu'être sur une seule ligne. De plus, le programme avertit l'utilisateur si une ligne dépasse plus de 200 caractères sans compter les espaces sous peine d'un mauvais traitement des informations suivantes (exemple : décalage des lignes dans la suite). En effet, lors de nos tests, nous nous sommes rendu compte que la fonction d'acquisition strtok(), ne fonctionne plus correctement après un certains nombres de caractère.

6. Exemple de notre programme

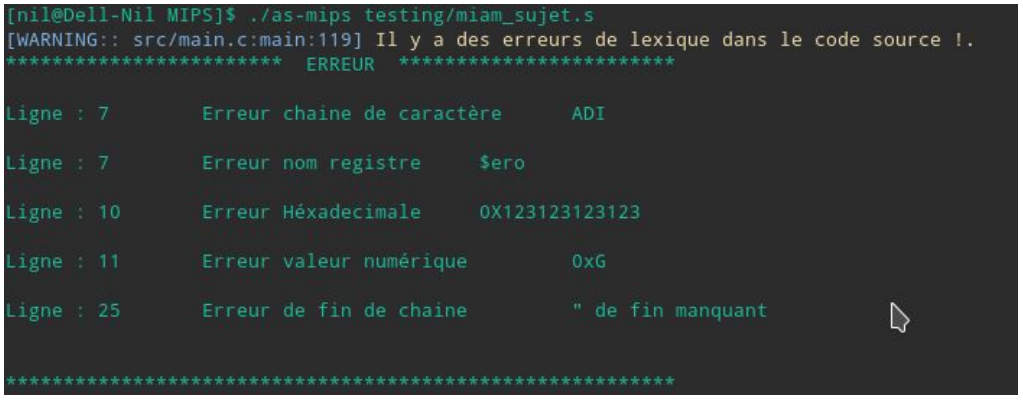

Figure 2 : Exemple d'exécution du programme, affichage des erreurs dans le terminal

```
27  Ligne : 5   Instruction     LW
28
29  Ligne : 5   Délimiteur  ,
30
31  Ligne : 5   Retour à la ligne
32
33  Ligne : 6   Instruction     LW
34
35  Ligne : 6   Délimiteur  -
36
37  Ligne : 6   Valeur Décimale     200
38
39  Ligne : 6   Délimiteur  (
40
41  Ligne : 6   Registre    $7
42
43  Ligne : 6   Délimiteur  )
44
45  Ligne : 6   Retour à la ligne
46
47  Ligne : 7   Instruction     ADDI
48
49  Ligne : 7   Retour à la ligne
50
51  Ligne : 8   Renvoie vers une étiquette  BOUCLE
52
53  Ligne : 8   Retour à la ligne
54
55  Ligne : 9   Instruction     BEQ
```

Figure 3 : Exemple du contenu du fichier analyse lexicale