Davy Nil
Besson Thomas

Livrable 2
Analyse Syntaxique
Partie 1
Projet Informatique : Assembleur MIPS
SEI 2018 - 2019

1. Introduction

L'analyse syntaxique est la deuxième opération qu'effectue un compilateur. Elle permet de vérifier la structure du programme : si les instructions sont situées à leur place, si leur paramètres sont du bon type etc… Dans le cas de ce projet, la partie Analyse syntaxique de l'assembleur MIPS à concevoir doit pouvoir identifier les 4 grands blocs que doit contenir un programme source et de gérer les erreurs de syntaxe.

2. Division Structurelle

Après avoir découpé le programme source et avoir fait des premières vérifications lors de l'analyse lexicale (Livrable 1), il faut vérifier la structure du code. Le langage de l'assembleur MIPS se divise en 4 parties : les sections .text, .bss, .data et .set, auxquelles s'ajoutent la collection des symboles.

La section .set a été vérifié dans le livrable 1. En effet, d'après le sujet, seule l'option noreorder peut se trouver après .set. C'est ce qui a été vérifié lors de l'analyse lexicale.

Pour vérifier le reste des sections, nous avons mis en place une machine à états finis pour trier les informations qui parviennent de la première opération de l'assembleur.

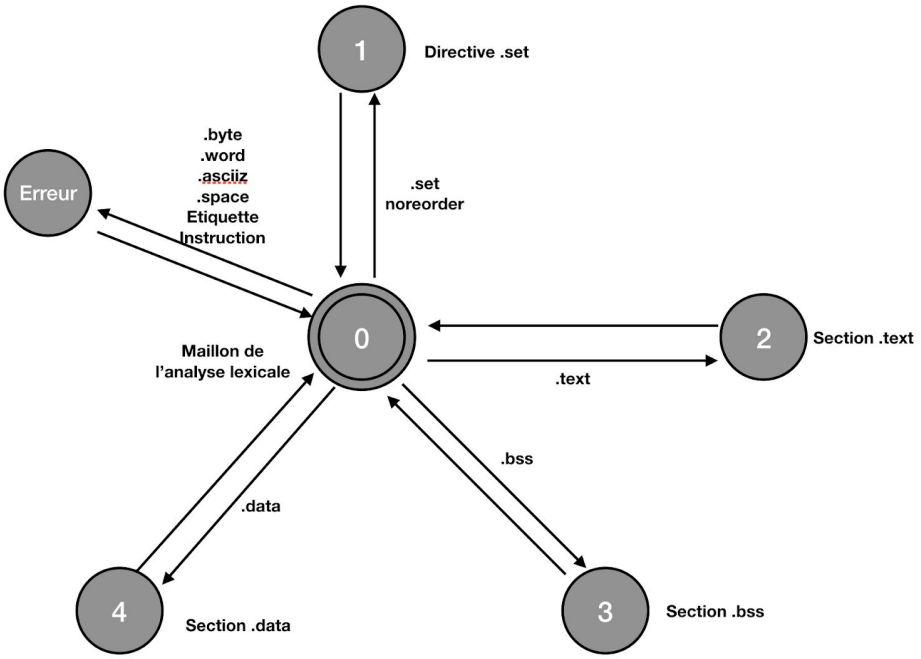

Figure 1 : Machine à états finis principale

3. Section .text

Dans cette section, il peut se trouver seulement des étiquettes ou des instructions. Il s'agit ici de vérifier, que les instructions existent et qu'il y a le bon nombre d'opérande pour l'instruction. Pour ce faire, nous avons utilisé une table de hachage contenant ces deux informations. Dans notre cas, nous avions déjà créé ce fichier pour vérifier l'existence des instructions lors du livrable 1. Nous avons seulement ajouté le nombre d'opérande au fichier. Dans cette section, nous avons également mis en place une machine à états finis :

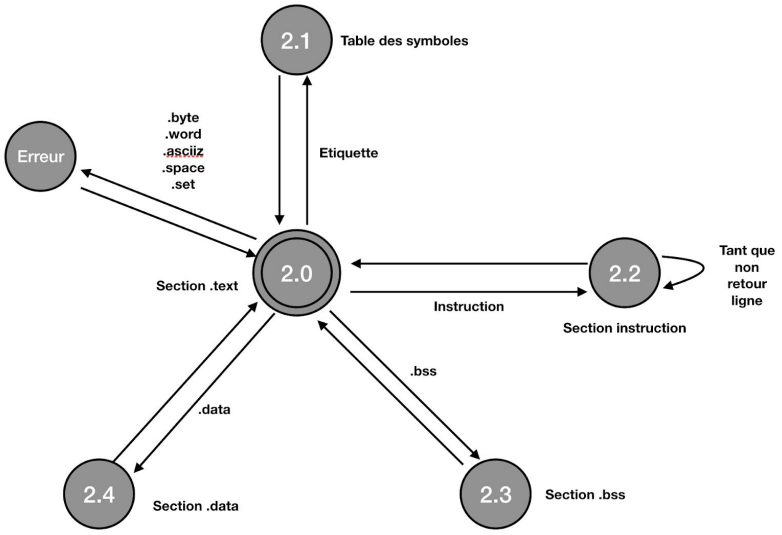

Figure 2 : Machine à état finis section .text

Pseudo-code si c'est une instruction :

Récupération du nombre d'opérande correspond à l'instruction
Tant que non retour à la ligne
    Ajoute à la collection .text les opérandes séparée par le délimiteur ,
Vérification du nombre d'ajout au nombre d'opérande attendu
Si le nombre d'opérande est différent du nombre d'opérandes attendus
    Ajout d'une erreur

4. Section .data

Dans cette section, les directives .word, .byte, .asciizz, et .space peuvent se trouver en plus d'étiquettes. Dans cette section, nous avons également mis en place une machine à états :

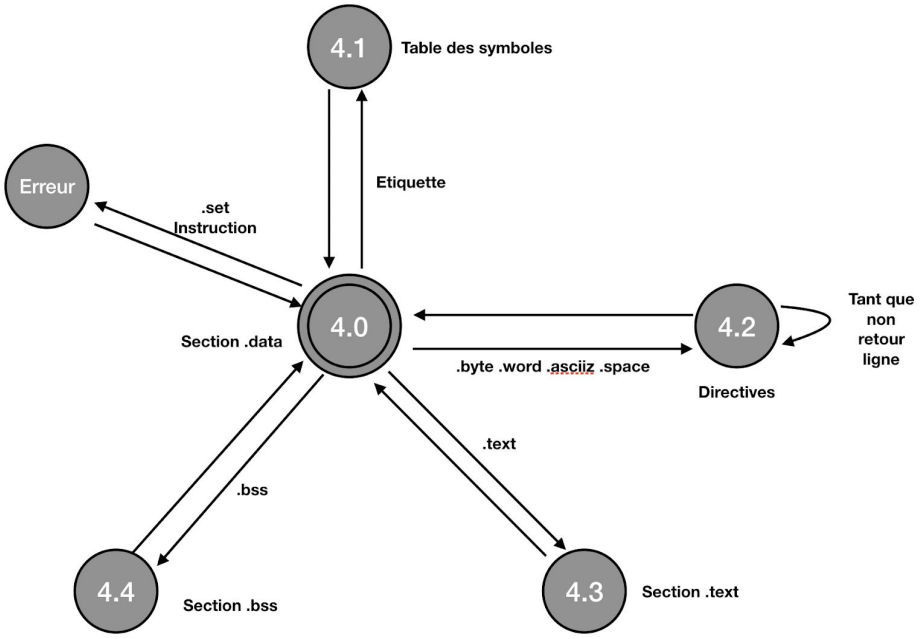

Figure 3 : Figure 4 : Machine à état finis section .data

Pseudo-code si c'est un .word:
Tant que non retour à la ligne
    Si valeur décimale (positive ou negative) ou valeur hexadécimale
        Ajoute à la collection .data les opérandes séparée par le délimiteur ,

Pseudo-code si c'est un .asciiz:
Tant que non retour à la ligne
    Si Chaine de caractère
        Ajoute à la collection .data

Pseudo-code si c'est un .byte :
    Tant que non retour à la ligne

Si valeur décimale comprise entre -128 et 127 ou valeur hexadécimale comprise entre 0x0 et 0xff

Ajoute à la collection .data les opérandes séparée par le délimiteur ,

5. Section .bss

Dans cette section, seulement une étiquette ou la directive .space peuvent s'y trouver. De plus, après cette dernière, il ne peut y avoir qu'une valeur décimale ou hexadécimale.

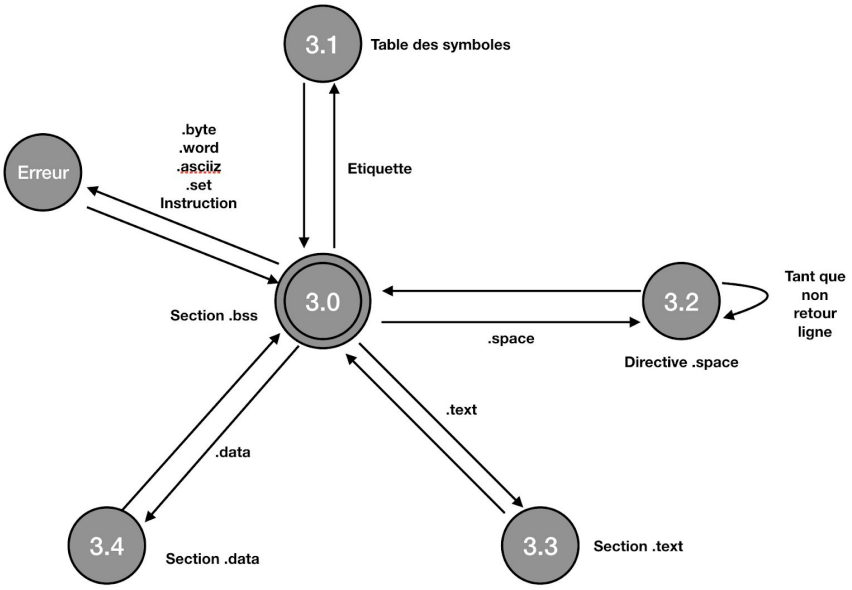

Figure 4 : Machine à état finis section .bss

Pseudo-code si c'est un .space :

Tant que non retour à la ligne

Si valeur décimale ou valeur hexadécimale

Ajoute à la collection .bss les opérandes séparée par le délimiteur ,

6. Exemple d'exécution

L'exécution de l'analyse syntaxique avec notre fichier miam_sujet.s (modifié). On écrit les files de .bss, .data, .text et les symboles dans des fichiers différents.

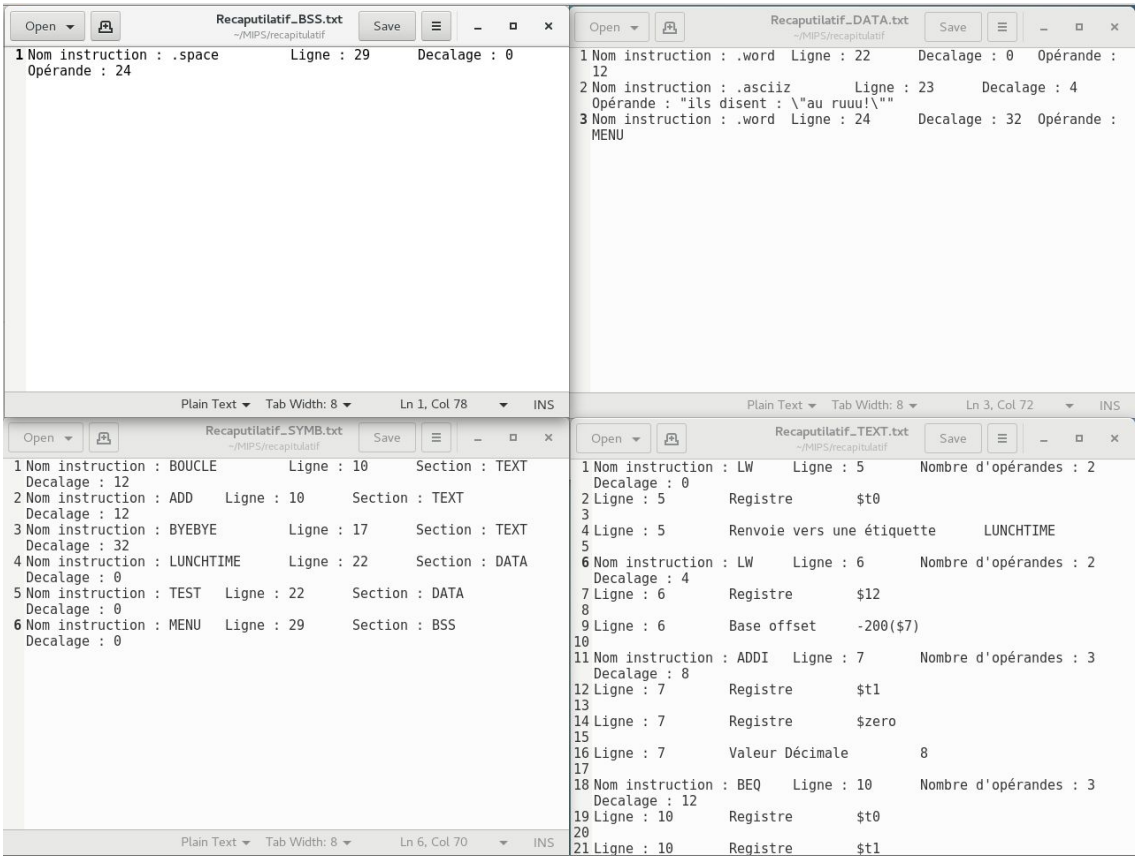

Figure 5 : Résultat des collections sur le fichier miam_sujet.s

Nous avons aussi écrit des tests qui doivent générer des erreurs : ici le résultat de l'analyse syntaxique du fichier test_data_erreur.s

```
[bessotho ~/MIPS ] $ ./as-mips tests/2livrable/test_data_erreur.s
[ DEBUG :: src/main.c:main:146] Le code source contient 5 lignes.
[ DEBUG :: src/main.c:main:168] Il n'y a pas d'erreur de lexique dans le code source !.
[WARNING:: src/main.c:main:181] Il y a des erreurs de syntaxe dans le code source !.
************************  ERREUR  ************************

Ligne : 2        Mauvaise commande après la directive .byte      129

Ligne : 2        Mauvaise commande après la directive .byte      -

Ligne : 2        Mauvaise commande après la directive .byte      256

Ligne : 3        Mauvaise commande après la directive .word      Chaine de caractère

Ligne : 4        Mauvaise commande après la directive .asciiz    Valeur Décimale

Ligne : 5        Mauvaise commande après la directive .space     Chaine de caractère

************************************************************
```

Figure 6 : Retour d'erreur sur le fichier test_data_erreur.s