FOURNIER Augustin
GARRIGOU Théo

Rapport 3 : Analyse Syntaxique (2)

Introduction :

Ce troisième rendu concerne la suite de l'analyse grammaticale. Dans cet incrément, on s'intéresse à la vérification des opérandes, à l'ajout des pseudos-instructions avec un modification de l'analyse lexicale et à la relocation des symboles.

I) La démarche :

Dans ce troisième incrément il y avait quatre objectifs principaux : nous devions être capables de vérifier la syntaxe de toutes les opérandes des instructions, gérer les pseudos instruction. Enfin nous devions associer les valeurs aux étiquettes et effectuer une table de relocation. La table de relocation permet de réajuster les différentes adresses de chaque élément en fonction de la zone de mémoire qui a été attribuée lors du chargement du code.

II) L'organisation pour cette partie :

Dans cet incrément, il fut plus facile de séparer les tâches car les codages de la vérification des opérandes et de l'ajout des pseudo-instructions ne se chevauchaient pas. Augustin a ainsi codé la gestion des pseudo-instructions car l'idée est assez similaire à celle des instructions qu'il a codé à l'incrément précédent. Théo s'est chargé de l'automate de vérification des instructions qui s'incruste dans l'analyse grammaticale de l'incrément 2. Enfin, les relocation ont été commencé par Théo et seront sûrement terminées avec l'aide d'Agustin.

III) Organisation du code pour cette seconde partie :

dictionnaire_registre.c : contient toutes les fonction lié à la vérification des registres et donc du dictionnaire de registres

dico_registre.txt : fichier texte contenant l'ensemble des registres autorisés et les conversions entre syntaxe par lettre et par chiffres des registres.

instruction.c : contient tout ce qui est associé au dictionnaire d'instructions.

operandes.c : contient toutes les fonctions de vérification des opérandes ainsi que l'automate de vérification des opérandes.

pseudo_instructions_dico.c : contient toutes les fonctions associées au dictionnaire de pseudo instructions.

pseudo_instructions.c : contient toutes les fonctions de chaînage de la liste de lexemes, et les fonctions permettants de remplacer les pseudos instructions.

pseudo_instructions.txt : fichier texte contenant les pseudos instruction existantes.

relocation.c : fichier contient toutes les fonctions associées à la table de relocation et à la gestion des étiquettes.

IV) Gestion des opérandes :

Les opérandes n'étaient que survolées dans l'automate de l'incrément 2 dans la fonction charge_instruction de analyse_grammaticale.c. Il a donc fallu revoir cette fonction et surtout la structure OPERANDE qui était un simple char* de la valeur des opérandes. Pour vérifier les valeurs des opérandes et les rendre exploitables par la suite, nous avons utilisé une UNION :

```
typedef union {
        unsigned char reg;
        short imm ;
        unsigned char sa ;
        ETIQ etiq ;
        int nb ;
        long tar ;
        BASE_OFFSET base_offset ; } VAL_OPERANDE ;
```

avec :

```
typedef struct bo {
        unsigned char reg ;
        short offset ;} BASE_OFFSET ;

typedef struct etiq {
        char nom[512] ;
        char attendu[10] ; } ETIQ ;
```

La nouvelle fonction charge_instruction est ainsi codée dans operande.c :

*L_LEXEME charge_instruction (L_LEXEME l , int** dec, L_TEXT* pl_text, INSTRUCTION instruction, L_SYMB* pl_symb)*

Elle prend en entrée la liste de lexème et met à jour ce pointeur car la fonction fait partie de l'automate de l'analyse grammaticale. Dans ce dernier, on charge avant l'appel de charge_instruction l'instruction à l'aide du dictionnaire d'instruction. Cela permet d'avoir l'information du nombre d'opérandes et de leur types attendu. En fonction de ces paramètres, l'automate va analyser les lexèmes suivant qui sont les opérandes.

Elle vérifie que les opérandes sont conforment aux types d'opérandes attendu par les instructions. De plus, la fonction gère le cas de nombres négatifs (cela n'était pas le cas avant cet incrément). Des 'warning_msg' sont utilisés pour gérer toutes les erreurs de type et permettre à l'utilisateur de corriger ses erreurs grâce aux indications de ligne et de numéro d'opérande. A chaque erreur, on quitte l'automate des opérandes pour passer au lexème suivant dans l'automate de l'analyse grammaticale

Augustin s'est chargé de coder les fonction valeur_imm, valeur_sa, valeur_target, valeur_offset et valeur_base_off. Ces fonctions permettent de vérifier que les opérandes ont la bonne valeur requise (taille maximale de la variable, signe, divisibilité par 4 éventuellement).

Un autre point, Augustin est revenu sur la vérification des registres (cas des base offset et des registres classiques). La vérification des valeurs de registres et la conversion de la notation en lettre vers la notation en chiffres s'effectue directement dans l'analyse lexicale. Pour cela nous avons utilisé un dictionnaire de registre qui permet à la fois de faire la conversion et la vérification des valeurs. Le fichier du dictionnaire se présente sous la forme suivantes : première ligne contient deux fois le nombre de

registres ensuite les lignes sont de la forme `nom_registre_lettre valeur_decimale_correspondante` pour les 32 registres puis `valeur_decimale valeur_décimale` pour les 32 registres en chiffre. Cette syntaxe qui peut paraître déroutante simplifie le codage du dictionnaire.

V) Gestion des pseudo-instructions :

Concernant la gestion des pseudos instructions c'est Augustin qui s'en est chargé. Pour cela un dictionnaire de pseudos instruction a été créé sur le même principe que le dictionnaire d'instructions. La première ligne indique le nombre d'instructions à chaque ligne on utilise la syntaxe suivante : `nom_instruction nombre_opérandes operande_1 operande_2 operande_3`. Le dictionnaire permet de reconnaître le nom d'une pseudo instruction et les opérandes qui y sont associées. La seconde partie de la gestion des pseudos instructions est codée dans le fichier pseudo_instructions.c. Dans ce fichier il y a tout d'abord une fonction de chaînage pour ajouter des lexemes au milieu de la liste de lexemes, on a une fonction de suppression des lexemes également. A noter que pour réaliser ces fonctions on a dû ajouter dans chaque LEXEME la numérotation de ce dernier pour se repérer dans la liste et pouvoir ajouter/supprimer des lexemes au bon endroit. Ensuite une fonction permet de remplacer chaque pseudo instructions par un jeux d'instructions normales a partir de la liste de lexemes donnée dans l'analyse lexicale. Nous n'avons pas utilisé le dictionnaire pour créer une nouvelle suite de lexemes, cela a été fait de manière brutale avec des ifs pour se simplifier la tâche dans un premier temps néanmoins cela prend beaucoup de lignes et si l'on souhaite ajouter des pseudos instructions il sera moins aisé de modifier le code. Une modification éventuelle de ce procédé sera faite. Nous vérifions que les opérandes des pseudos instructions sont correctes via l'automate de vérification des opérandes des instructions qui revérifie la liste liste de lexemes modifiée ne contenant que des instructions classiques Nous avons néanmoins un problème : nous ne sommes pas capables de détecter si LW et SW sont des pseudos instructions ou des instructions. Ce soucis devra être rapidement réglé.

VI) Relocations :

La gestion de la relocation des piles .text et .data ainsi que la table des symbole a été amorcé. Celle-ci n'a pas été terminé par manque de temps et aussi car la compréhension de celle-ci nous a été assez difficile. Le squelette de la relocation a été dessiné dans relocation.c. Le but de cette fonction est de définir les types de relocation pour les symboles (R_MIPS_32, R_MIPS_26, R_MIPS_HI16, R_MIPS_LO16) des sections .text et .data et dans le tableau de symbole. Pas mal de travail est encore à réaliser dans cette partie avant de passer à la suite.

VII) Tests réalisés :

Les tests ont été réalisés à partir de miam_sujet.s avec des modifications à nouveaux tests : nous avons fait varier les instructions et testé beaucoup de combinaisons d'opérandes pour trouver les points faibles de notre code. Nous avons également fait des tests au fur et à mesure de l'avancement sur des micro programme à un ou deux jeux d'instructions.

VIII) Objectifs atteints et ce qu'il reste à faire :

Dans ce nouvel incrément, nous avons tout d'abord consolidé l'incrément 2 et 1 en retouchant quelques petites fautes mentionnées dans les rapport précédents. Ensuite nous avons eu une très grosse partie qui a concerné en la vérification des opérandes, en même temps de nombreuses modification de l'automate de l'analyse grammaticale ont été réalisées.

Avant de passer à l'incrément suivant, nous concentrerons tous nos effort dans la relocation et la vérification des pseudo-instructions. Par manque de temps et d'efficacité nous n'avons pas terminé la table de relocation, de plus la gestion des pseudos instructions est encore beaucoup trop fragile.

Conclusion :

Dans ce nouveau rendu, nous avons consolidé les deux précédents rendus, nous avons également été capables de vérifier l'ensemble de valeurs des opérandes. Enfin nous somme capable de remplacer une partie des pseudos instruction. La table de relocation n'est pas encore terminée. Nous avons accumulé un certain retard dû à la un manque d'efficacité et à la faute d'avoir dû régler des problèmes dues aux incréments précédents. Nous avons peut être été dépassés par le volume de ce nouvel incrément plus difficile dans sa compréhension que les précédents. Nous allons devoir combler ce retard au plus vite.