DEBAIN Yann
LETOUZE Clément

Projet informatique : assembleur MIPS
Livrable 3

2017-2018

SOMMAIRE:

Introduction

Au cours de ce troisième livrable nous nous sommes principalement attachés à trois points :

- La gestion des pseudo-instructions
- La table de relocation
- Le système d'adressage offset(base)

Le premier point concerne la modification de notre file d'instruction en fonction de si une pseudo-instruction est rencontrée ou non. La table de relocation gère les problèmes des instructions prenant en opérande un étiquette. Enfin le système d'adressage offset(base) intervient pour certaines pseudo-instructions.
Nous sommes également revenus sur certains points du livrable deux.

I.1) Pseudo-instruction

Pour faciliter le codage, certaines instructions appelées "pseudo" instructions peuvent être utilisées. Leur utilisation consiste en la contraction de plusieurs instructions en une seule. Cela apporte une meilleurs lisibilité et une plus grande rapidité de codage mais il faut alors correctement le traiter dans notre file d'instruction. Pour gérer ce cas là, nous analysons notre file d'instructions et remplaçons la pseudo instructions par les instructions usuelles de l'assembleur MIPS.

Par exemple pour une instruction LW:

```
[ nom = LW
 nombre d'operande = 2
 decalage = 0
 numero de ligne = 5
 -> operande n°1 = [ $t0 = 8] : REGISTRE  LIGNE n°5
 -> operande n°2 = [ a = 0] : MOT OU INSTRUCTION  LIGNE n°5
 ]
```

devient

```
[ nom = LUI
 nombre d'operande = 2
 decalage = 0
 numero de ligne = 5
 -> operande n°1 = [ $at = 1] : REGISTRE  LIGNE n°5
 -> operande n°2 = [ 16 bit poids faible = 0] : DECIMAL  LIGNE n°5
 ]

[ nom = LW
 nombre d'operande = 2
 decalage = 4
 numero de ligne = 5
 -> operande n°1 = [ $t0 = 8] : REGISTRE  LIGNE n°5
 -> operande n°2 = [ 16 bit poids fort = 0] : DECIMAL  LIGNE n°5
 ]
```

1.2) Table de relocation

La table de relocation est utile pour les fonctions prenant comme opérande des étiquettes. En effet, les étiquettes sont définies dans une section à un endroit donné. Cependant, lorsqu'une fonction (comme J par exemple) utilise le nom de cette étiquette comme opérande, il est impératif de spécifier l'adresse réelle de l'opérande qui n'est pas celle où l'instruction est appelée.

Il s'agit alors d'effectuer la relocation correspondant à la section et au décalage par rapport à cette section de la déclaration de l'étiquette. Un point particulier auquel il faut faire attention : cette partie devra être traitée après le traitement des pseudo-instructions car ces dernières entraînent un changement d'indexage dans le code, il faudra alors faire attention à ce changement.

Pratiquement, cela revient à faire une recherche dans la table des symboles pour trouver l'étiquette correspondante et à faire une relocation à la section et au décalage par rapport à cette section.

```
# addition

.set noreorder
.text
    LW $t0, a
    LW $t1, b
    LW $t2, res
    ADD $t2, $t0, $t1
.data
a:
    .word 5
b:
    .word 2
res:
    .word 0
```

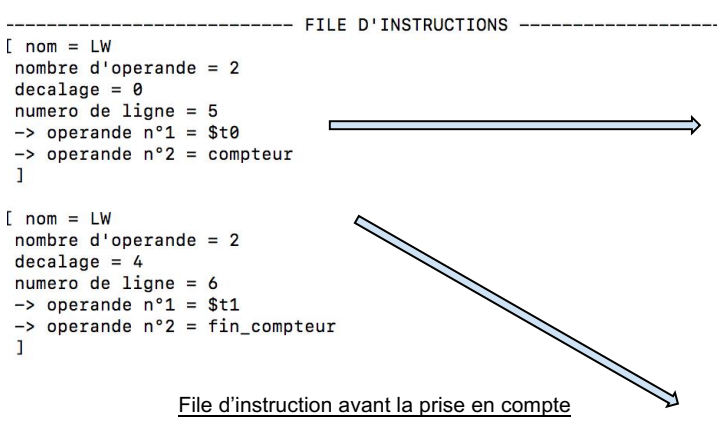
```
{
   [a:] ---> [depart : section = .text; adresse = 0]
   R_MIPS_LO16 -> R_MIPS_HI16
   [a:] ---> [arrivee : section = .data; adresse = 0]
}

{
   [b:] ---> [depart : section = .text; adresse = 8]
   R_MIPS_LO16 -> R_MIPS_HI16
   [b:] ---> [arrivee : section = .data; adresse = 4]
}

{
   [res:] ---> [depart : section = .text; adresse = 16]
   R_MIPS_LO16 -> R_MIPS_HI16
   [res:] ---> [arrivee : section = .data; adresse = 8]
}
```

1.3) Le système d'adressage offset(base)

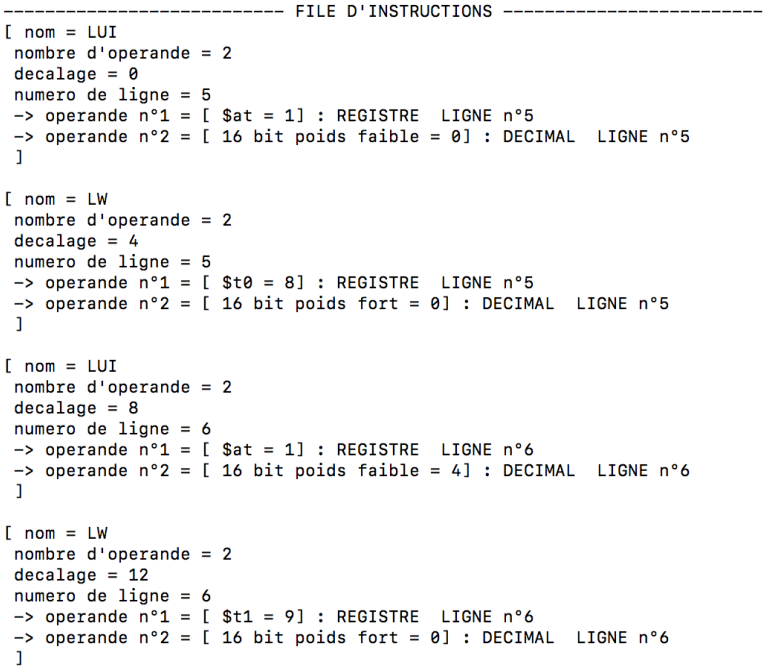
Dans le cas de certaines instructions comme LW ou SW une des opérandes se présente sous la forme offset(base). Ce système d'opérande peut être utile par exemple pour la manipulation de tableau.
La prise en compte de l'adressage par offset(base) n'est pas encore pris en compte dans notre code.

1.4) Test du troisième livrable et retour sur l'incrément 2

Dans l'incrément 3, nous avons ajusté l'adressage des instructions et des symboles en prenant en compte les pseudo-instructions.



File d'instruction avant la prise en compte des pseudo-instructions



File d'instruction après la prise en compte des pseudo-instructions

Nous avons aussi corrigé quelques erreurs d'interprétations concernant la déclaration de variable et mise en place un test pour savoir si le type de la variable est respecté.

Mise à part le problème des instructions prenant comme opérande un offset(base), le programme est fonctionnel avec une gestion d'erreur faisant tourner le programme jusqu'au bout pour avoir la liste complète des erreurs en tête d'affichage.

```
#petit programme

.set noreorder

.text
test:
    LW $t0, reloc
    J test
    NOP

.data
reloc:
    .word 12
```

```
.set noreorder -> PAS D'OPTIMISATION.

---------------------------- TABLE -------------------------
[ section = .text
 nom = test
 decalage = 0
 ligne n°6 ]

[ section = .data
 nom = reloc
 decalage = 0
 ligne n°12 ]

--------------------------- FILE D'INSTRUCTIONS ------------------------
[ nom = LUI
 nombre d'operande = 2
 decalage = 0
 numero de ligne = 7
 -> operande n°1 = [ $at = 1] : REGISTRE  LIGNE n°7
 -> operande n°2 = [ 16 bit poids faible = 0] : DECIMAL  LIGNE n°7
 ]

[ nom = LW
 nombre d'operande = 2
 decalage = 4
 numero de ligne = 7
 -> operande n°1 = [ $t0 = 8] : REGISTRE  LIGNE n°7
 -> operande n°2 = [ 16 bit poids fort = 0] : DECIMAL  LIGNE n°7
 ]

[ nom = J
 nombre d'operande = 1
 decalage = 8
 numero de ligne = 8
 -> operande n°1 = [ test = 0] : MOT OU INSTRUCTION  LIGNE n°8
 ]

[ nom = SLL
 nombre d'operande = 3
 decalage = 12
 numero de ligne = 9
 -> operande n°1 = [ $0 = 0] : REGISTRE  LIGNE n°9
 -> operande n°2 = [ $0 = 0] : REGISTRE  LIGNE n°9
 -> operande n°3 = [ 0 = 0] : ZERO  LIGNE n°9
 ]
---------------------------- FILE DE DATA ------------------------
[ nom = .word
 etat = DECIMAL
 decalage = 0
 numero de ligne = 13
 valeur = 12 ]


---------------------------- FILE DE BSS -------------------------
FILE VIDE

---------------------------- TABLE DE RELOCATION -------------------------
{
   [reloc:] ---> [depart : section = .text; adresse = 0]
   R_MIPS_LO16 -> R_MIPS_HI16
   [reloc:] ---> [arrivee : section = .data; adresse = 0]
}

{
   [test:] ---> [depart : section = .text; adresse = 8]
   R_MIPS_26
   [test:] ---> [arrivee : section = .text; adresse = 0]
}
```

Résultat de l'exécution de petit_programme.s