Davy Nil
Besson Thomas

Livrable 4
Génération du fichier ELF
Projet Informatique : Assembleur MIPS
SEI 2018 - 2019

1. Introduction

La génération du code relogeable au format Elf est la dernière opération effectué par le compilateur MIPS. Une librairie nous a été fournie afin de permet la génération de celui-ci. Pour faire cette génération avec la librairie il est nécessaire de transformer les sections .data, .text et .bss en binaire. Grâce à la librairie fournie il suffit ensuite de se servir des fonctions qui génère les sections pour créer le fichier ELF. De plus le ELF étant relogeable il faut gérer les symboles ainsi que la relocation, cela consiste à faire le lien entre nos structures de données et les fonctions qui servent à générer les fonctions du fichier ELF.

2. Génération Binaire

A. Section BSS

La section .bss ne peut contenir que des .space. La directive .space permet d'allouer un nombre d'octets égal au nombre suivant. Il suffit donc de le récupérer et de remplir la section avec la fonction donnée.

B. Section TEXT

Chaque instruction de la section .text se construit toujours de la même manière. Chaque instruction est codée sur 4 octets avec un opcode au début. De plus il y a des similitudes suivant le type de l'instruction: R,I ou J.

A chaque nouvelle instruction, le programme recherche dans la table de hachage le type de l'instruction pour l'envoyer dans la bonne fonction.

Si elle est du type R alors il y a un opcode et un code fonction à récupérer en plus. Il y a également les fonctions des opérandes de l'instructions pour pouvoir les remettre dans l'ordre pour construire le code binaire. Suivant l'instruction le registre de destination par exemple peut se trouver en premier, en second ou à la dernière position. Puis tout est

transformé en binaire avant d'être concaténé dans un char dans le bon ordre avant d'être transformé en un hexadécimal.

Si l'instruction est du type R ou I, seuls l'opcode et la fonction des opérandes sont récupérés dans la table de hachage. Dans ces cas là, il peut il y avoir des étiquettes. Si elle correspond à un immédiat ou un absolu et si elle se trouve dans la section .text alors elle est remplacée par l'offset correspondant sinon on met la valeur 0 à la place.

Après la création de tous les codes hexadécimaux, ils sont renversés pour respecter la convention du big endian. Puis le code binaire est généré grâce à la fonction donnée.

C. Section DATA

Dans la section data, il peut il y avoir les directives .space, .word, .asciiz, .byte. Tout d'abord, un tableau est créé contenant le code ascii des opérandes de la section .data lettre par lettre et nombre par nombre. Dans le cas d'un .word, on vérifie si c'est une étiquette et si c'est le cas, on vérifie vers quoi elle pointe pour pouvoir la remplacer : si elle pointe vers un élément qui se trouvent dans la même dans section alors on renvoie le décalage et sinon elle est remplacée par 0. Après la conversion en hexadécimal, ces codes sont concaténés pour créer des mots de longueur 8 en faisant attention à ce que les .word soient toujours alignés sur 4 octets. Puis un swap est effectué pour respecter la convention big endian de l'assembleur MIPS avant d'être envoyé dans la fonction pour générer le binaire.

3. Gestion des symboles

A. La table des strings (strtab)

La table de string contient les noms des symboles mis bout à bout et séparé par une sentinelle. Pour remplir cette section nous parcourons notre liste de symbole puis nous les ajoutons en prenant soin de les ajouter par ordre de définition et en ajoutant les symboles non définis à la fin. Nous avons simplement repris le code d'exemple fournis en parcourant la liste de nos symboles dans le bon ordre.

B. La table des symboles (symtab)

Pour construire la table des symboles (symtab du fichier ELF), nous ajoutons un à un les symboles dans la section, par ordre de ligne d'apparition (et non de définition comme annoncé dans l'énoncé car cela ne correspond pas au code compilé du compilateur mips-as). Pour chaque symbole qu'on ajoute à la section on ajoute on reprend son nom, sa section, son offset par rapport au début de la section. On définit s'il s'agit d'un symbole local (défini dans ce fichier) ou global (non défini dans ce fichier).

C. La table de relocation (reltab)

Pour construire la table de relocation on repart de la structure de donnée implémenté au livrable 3. De celle-ci on récupère pour chaque relocation le type de relocation ainsi que la position relative de la relocation par rapport au début de la section. On effectue cette opération deux fois : une première fois pour construire la table de relocation de .text et une seconde fois pour construire celle de la section .text.

4. Exemple de ces différentes section avec le code miam-sujet.s fourni

Section .strtab, 43 bytes:

```
00000000  00626f75
00000004  636c6500
00000008  62796562
0000000c  7965006c
00000010  756e6368
00000014  74696d65
00000018  006d656e
0000001c  75007669
00000020  74657669
00000024  74656175
00000028  727500
```

Section .bss, 24 bytes:

```
00000000  00000000
00000004  00000000
00000008  00000000
0000000c  00000000
00000010  00000000
00000014  00000000
```

Section .data, 32 bytes:

```
00000000  0000000c
00000004  00000000
00000008  696c7320
0000000c  64697365
00000010  6e74203a
00000014  20226175
00000018  20727521
0000001c  22000000
```

Section .text, 40 bytes:

```
00000000  3c080000
00000004  8d080000
00000008  8ce6ff38
0000000c  20090008
00000010  11090004
00000014  00000000
00000018  21290001
0000001c  08000004
00000020  00000000
00000024  0c000000
```