Mode d'Emploi

Ce document explique brièvement la compilation et l'exécution des programmes utilitaires développés dans le cadre de ce projet.

Phase 1

Ces programmes ont été ajoutés au fichier **Makefile.am**. Pour les compiler, il suffit de lancer les commandes suivantes : **./configure**, puis **make**. Cela compilera le programme.

Pour exécuter le tout, utilisez le fichier **elf_readelf**, don la commande **readelf**. La syntaxe est : **./elf_readelf <option> <fichier>** .
Pour afficher le menu des options disponibles, exécutez simplement : **./elf_readelf**

→ Le programme propose 5 options :

- **-h** : Affichage de l'en-tête.
- **-S** : Affichage de la table des sections.
- **-s** : Affichage de la table des symboles.
- **-r** : Affichage des tables de réimplantation.
- **-x** : Affichage du contenu d'une section. Cette option nécessite un argument supplémentaire (le numéro ou le nom de la section) en plus du nom du fichier en ligne de commande.
  Exemple : **./elf_readelf -x <fichier_elf> .text**
  ou **./elf_readelf -x <fichier_elf> 1**

Phase 2

Cette phase nécessite la mise-en-place d'un simulateur ARM

Dans un premier temps, nous avons besoin de faire **./configure**, suivi à nouveau de **make**
(si ces derniers n'ont pas déjà était fait)

Ainsi, les fichiers .o seront créés au sein du dossier : ***Examples_loader***

Pour effectuer la réimplantation d'un fichier objet, obtenu par la compilation d'un code assembleur pour ARM 32 bits en big endian, il suffit d'exécuter **./process_rel** , en passant en argument le fichier à modifier. Cette commande génère un nouveau fichier ELF avec le même nom, auquel est ajouté le suffixe **_modified**

Puis on lance le simulateur ARM, en faisant bien en sorte que le port (XXXX) soit libre:
./arm_simulator --gdb-port XXXX --trace-registers --trace-memory --trace-state SVC &

Finalement, on charge le fichier précédemment obtenu dans le simulateur au port précédemment utilisé avec la commande:
./load_sim Examples_loader/example.o_modified localhost XXXX