Mode d’emploi (Groupe 10)

Compilation du programme : Make

Lancement des scripts de tests : ./launchTests.sh dans le répertoire principal permet de lancer tous les scripts de tests du répertoire Tests
Ces scripts peuvent être lancés un par un (depuis le dossier principal) avec ./Tests/nomDuScript.sh

Utilisation du programme :
./ARM_runner --source cheminDuFichierSource [+ options] [--dest cheminDuFichierDest]
Options :

- **“-h”** affiche le header du fichier source
- **“-S”** affiche les sections du fichier source
- **“-s”** affiche la table des symboles du fichier source
- **“-r”** affiche les tables de relocations du fichier source
- **“-x”** affiche le contenu des sections du fichier source
- **“-a”** affiche le résultat des options -h -S -s -r
- **“-w”** copie le fichier source vers le fichier dest. Nécessite d’avoir spécifier un fichier destination avec --dest
- **“-d”** copie le fichier source vers le fichier dest en ayant supprimés les sections REL et en ayant fait toutes les modifications associées (renumérotation etc…). Nécessite d’avoir spécifier un fichier destination avec --dest
- **“--text <addr>”** copie le fichier source vers le fichier dest en ayant corrigé la table des symboles pour que chaque symbole ait une valeur absolue. Nécessite d’avoir spécifier un fichier destination avec --dest
- **“--data <addr>**” copie le fichier source vers le fichier dest en ayant corrigé la table des symboles pour que chaque symbole ait une valeur absolue. Nécessite d’avoir spécifier un fichier destination avec --dest

Structure

main

Le programme principal est ./ARM_runner.c.

utils

Le fichier structure.h contient la structure globale ainsi que des structures annexes permettant de simplifier le code.

Le fichier util.c contient une fonction qui retourne le nom d’une section à partir d’un offset donné en paramètre.

fonctions

Les fonctions développées se trouvent toutes dans le dossier ./Functions.
Etape 1 : ./Functions/readElfHeader
Etape 2 : ./Functions/readSectionHeader
Etape 3 : ./Functions/readSectionContent
Etape 4 : ./Functions/readSymTable
Etape 5 : ./Functions/readRelocTable
Etape 6 : ./Functions/implantation : fonction deleteRel
Etape 7 : ./Functions/implantation : fonction correctSymTable
Etape 8 : ./Functions/implantation : fonction correctABSReloc
Autre : ./Functions/writeFile

tests

Les tests se trouvent dans ./Tests et sont des scripts automatisés sur les exemples présents dans ExamplesLoader. Il existe un script pour l'étape 1, 2, 3, 4, 5.

Fonctionnalités

Phase 1 :
Affichage de l'en-tête et chargement en structure (`Elf32_Ehdr`).
Affichage de la table des sections et chargement en structure (`Elf32_Shdr`).
Affichage du contenu des sections et chargement en structure (`SectionContent`).
Affichage de la table des symboles et chargement en structure (`Elf32_Sym`).
Affichage de la table des réimplantations et chargement en structure (`Elf32_Rel`).

Phase 2 :
Suppression des sections de réimplantation en structure (⇒ renumérotation des sections).
Attribution d'adresses absolues aux symboles d'après les adresses de chargement des différentes sections.
Correction du numéro de section de chaque symbole selon renumérotation.
Réimplantations de type R_ARM_ABS.

Autres :
Production d'un fichier ELF à partir des structures du programme.

Manquantes :
Réimplantations de type R_ARM_JUMP24 et R_ARM_CALL.
Interfaçage avec le simulateur ARM.
Production d'un fichier exécutable non relogeable.

Bogues

Erreur connue sur valgrind : “`Conditional jump or move depends on uninitialised value(s)`” :
L’une des libération de mémoire des contenus de sections se fait sur une valeur non initialisée (l.170-172 ARM_runner.c).

Tests

Etape 1 :
./Tests/readHeaderTest.sh

Objectif : Vérifier que l’affichage du header correspond à celui de readelf

Etape 2 :
./Tests/readSectionTest.sh

Objectif : Vérifier que l’affichage des section headers correspond à celui de readelf

Etape 3 :
./Tests/readSectionContent.sh

Objectif : Vérifier que l’affichage des contenus de sections correspond à celui de readelf

Etape 4 :
./Tests/readSymTable.sh

Objectif : Vérifier que l’affichage de la table des symboles correspond à celui de readelf

Etape 5 :
./Tests/readRelocTable.sh

Objectif : Vérifier que l’affichage de la table des réimplantations correspond à celui de readelf

./launchTests.sh lance tout les scripts d’un coup

Etape 6 & 7 :
Pas de test automatisé.
Pour l’étape 6 et 7, puisque notre programme principale nous permet de réécrire les structures modifiées dans un fichier de sortie, on peut vérifier à l’aide de readelf et apercevoir la correction des symboles ou la suppression des tables de réimplantations.

Etape 8 :

Dans les exemples fournis, aucune réimplantations sont de type R_ARM_ABS16 ou R_ARM_ABS8, et il n'y a rien à faire pour les réimplantations de type R_ARM_ABS32, donc nous n'avons pas pu tester.