Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg

Alan Sueur
Jonathan Rial
Classe I-2a
Systèmes Embarqués
30.09.16

TP02 - Introduction à l'assembleur

Réponses aux questions

Citer les modes d'adressage courants supportés par le μP ARM ?

Les modes d'adressage courants sont :

Name	Alternative Name	ARM Examples
Register to register	Register direct	MOV R0, R1
Absolute	Direct	LDR R0, MEM
Literal	Immediate	MOV R0, #15 ADD R1, R2, #12
Indexed, base	Register indirect	LDR R0, [R1]
Pre-indexed, base with displacement	Register indirect, with offset	LDR R0, [R1, #4]
Pre-indexed, autoindexing	Register indirect pre-incrementing	LDR R0, [R1, #4]!
Post-indexing, autoindexed	Register indirect post-increment	LDR R0, [R1], #4
Double Reg indirect	Register indirect Register indexed	LDR R0, [R1, R2]
Double Reg indirect with scaling	Register indirect indexed with scaling	LDR R0, [R1, r2, LSL #2]
Program counter relative		LDR R0 [PC, #offset]

Quel outil permet de transformer le code assembleur en code objet ?

On transforme un programme écrit en langage d'assemblage en du code objet avec un outil nommé assembleur. À chaque instruction du langage d'assemblage, correspond une instruction machine.
Pour chaque fichier source, l'assembleur créera un fichier ".o".

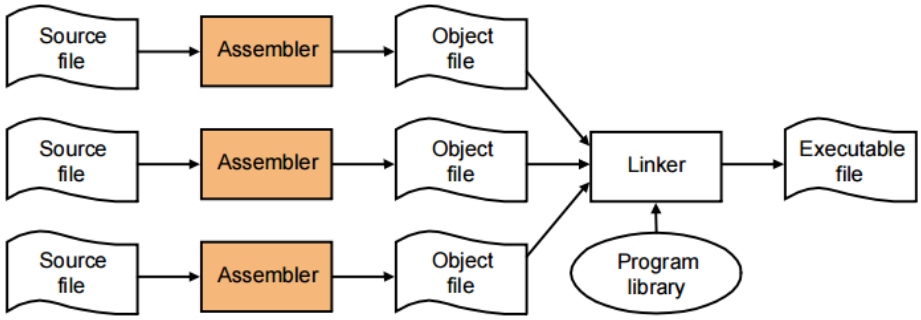

FIGURE 1 – https ://cs.gmu.edu/ setia/cs365-S02/assembler.pdf

Quel outil permet de créer l'application à partir des codes objet précédemment générés ?

C'est grâce à l'éditeur des liens que nos fichiers objets se transforment en fichier exécutable. Ce dernier peut contenir des librairies supplémentaires. Cet outil construit une table qui contient le nom, la taille et la longueur de tous les objets. Il affecte également une adresse de chargement à chaqu'un d'eux.

Alan Sueur
Jonathan Rial
Classe I-2a
Systèmes Embarqués
30.09.16

Pourrait-on optimiser l'algorithme du crible d'Eratosthène ? Si oui, comment ?

Le crible d'Atkin est une version optimisée du crible d'Eratosthène qui se base sur des restes de divisions, mais ce dernier est plus compliqué à implémenter.
Si nous en restons à l'algorithme de base, alors on peut effectivement l'optimiser. (cf. voir la prochaine question)

Pourrait-on réduire la taille de votre code en assembleur ? Si oui, comment ?

Oui, en créant plutôt un tableau de nombre qui ne sont pas premiers. On peut ainsi tester s'il n'est pas égal à "true", on affiche le résultat. On évite ainsi d'initialiser toutes les valeurs du tableau. Voir le code optimisé dans le fichier "main.S".

Heures de travail

En dehors des heures de cours, nous avons dû travailler 2h20. Durant le cours nous avons pu réalisé les tâches suivantes :

- Finir l'exercice en affichant tous les nombres premiers sur la console.
- Répondre à la question 1.

A la maison, nous avons dû finir les tâches suivantes :

- Répondre aux questions 2-5.
- Trouver, éditer et tester une optimisation du code.
- Rédiger ce rapport

Synthèse

Non acquis

- Rien

Partiellement acquis

- C'est grâce au débuggage que nous avons réussi à réaliser le code. Sur papier nous faisons encore trop de fautes.

Acquis

- Créer un tableau, chercher et stocker les informations à l'intérieur.
- Faire un printf, pour afficher des informations sur la console.
- Commencer à se familiariser avec l'assembleur, créer des comparaisons, des branchements, des constantes, etc.

Remarques & Feedback

Nous avons eu du mal à trouver de la documentation sur internet pour comprendre comment fonctionnait un tableau. Après une courte explication du prof sur l'utilisation de celui-ci, nous avons réussi à réaliser l'exercice. Les premières questions étaient quelque peu ambiguës, nous avons passé du temps pour rechercher à différents endroits pour trouver ces réponses. Finalement ce deuxième travail pratique fut très bénéfique pour nous car nous avons appris beaucoup de choses sans être bloqué à un certain stade de l'exercice.