Projet informatique SICOM : 1e livrable

SICOM-A
Pour le 7/10/2014

HONORE Antoine
PICHAT Maxence

Introduction

L'objet de ce rapport est de relater nos impressions après la réalisation du premier livrable du projet d'informatique SICOM 2014. Ce projet étant assez complexe, nous avons bénéficié de code fourni par les professeurs pour démarrer dans de bonnes conditions. Ce qui suit est un résumé de notre organisation en ce début de projet, des problèmes que nous avons rencontrer et notre avancement.

1 Organisation

Pour ce 1e livrable nous avons décidé de nous partager les tâches de la manière suivante : Antoine doit s'occuper principalement de la mise en place de l'environnement mémoire du MIPS, et Maxence doit plutôt gérer la partie vérification de l'interpréteur. Ensuite nous avons mis en commun nos travaux, et commencé à coder les commandes demandées. Par ailleurs il est à noter que le groupe n'est pas d'un niveau vraiment homogène (différences Phelma PET/ ENSE3).

- **Mises en commun et manières de travailler**

Afin de pouvoir accéder aux fichiers créés et les synchroniser rapidement depuis chez soi, nous avons utilisé Dropbox et avons prévu de nous réunir quelques fois pour faire des points.

Le problème rencontré avec Dropbox est que nous synchronisions chaque fois les modifications apportées à tel ou tel fichier : c'était d'abord dangereux dans le sens où si l'un de nous modifiait sans le vouloir une partie du code, le code initial serait difficile voire impossible à récupérer (et retrouver les erreurs parmi les fichiers de plus en plus nombreux gâcherait un temps précieux).

Nous allons plutôt synchroniser la Dropbox dans un dossier personnel (sur nos ordinateurs respectifs) et dans lequel nous effectuerons les modifications que l'on souhaite. Ainsi il suffit ensuite de copier les fonctions préalablement testées et correctes dans la Dropbox.

Le projet a été organisé avec un dossier include contenant les fichiers headers de définitions et prototypes, un dossier src contenant le code des fonctions et commandes à utiliser et un dossier tests.

2 Problèmes rencontrés et état du projet

Les principaux problèmes que nous avons rencontrés ont été, tout d'abord, la compréhension des morceaux de code donnés par les professeurs, puis savoir si certains pouvaient être gardés, avaient besoin de modifications ou étaient susceptibles d'être utilisés ultérieurement (et ainsi ils ont été passés en commentaires).

Lors des compilations successives nous sommes tombés sur les problèmes « habituels » concernant des oublis ou des erreurs d'inattention, et rapidement corrigibles. Cependant il y a aussi eu des problèmes dus à des doubles appels et des fonctions imbriquées (nested functions) qui ont été plus compliqués à gérer.

Nous avions eu l'idée de la fonction verif_ligne qui a été codée : elle permet de vérifier qu'une ligne a été correctement saisie, c'est-à-dire qu'il y ait les bonnes commandes (en premier token), puis ensuite les arguments correspondant aux types d'arguments de ces commandes.
Or finalement elle s'avère compliquée à mettre en œuvre, et inefficace puisqu'il faudrait regarder plusieurs fois la même chaine de caractères avant d' effectivement lancer une commande. Nous allons plutôt intégrer dans les fonctions les tests qui permettent de vérifier si les arguments saisis après une commande sont du bon type, ce qui est plus simple.

- **Travail effectué**

Nous avons ainsi réussi à mettre en place l'environnement mémoire du MIPS (segmentation et définition de la mémoire virtuelle, des registres…), des fonctions vérifiant les types de chaînes de caractères (fonctions is_* utilisées pour vérifier la syntaxe correcte des saisies dans l'interpréteur) ainsi que les commandes load et disp.

Notons que pour la commande « disp », nous avons rajouté un champ au type *mem*.
Ce champ est en fait un tableau de *word* qui permet d'accéder plus simplement à une adresse mémoire. *Par ce moyen on peut accéder à une adresse mémoire en décalant simplement le pointeur mem->tab.*

Pour l'instant, seules les commande load et disp sont entièrement testées et fonctionnelles.

Conclusion:

Pour ce premier livrable, nous avons préféré nous concentrer sur la mise en forme du programme et sur la compréhension du code fourni par les professeurs. Ceci nous permet d'entamer le second livrable sur de bonnes bases et d'utiliser au maximum les fonctions déjà existantes.