Projet de programmation C: Emulation MIPS[i]

Le projet est à réaliser par groupes de 2 élèves.

Description du sujet

Un émulateur est un logiciel capable de reproduire le comportement d'un objet. Dans le cas qui nous intéresse il s'agit de la machine (processeur) MIPS lorsqu'elle exécute un programme. Reproduire le comportement signifie, plus précisément, que l'émulateur gère une mémoire et des registres identiques à ceux de la machine MIPS, puis réalise l'évolution de l'état de cette mémoire et de ses registres au fur et à mesure que les instructions du programme s'exécutent. L'émulateur doit fournir les mêmes résultats que ceux que l'on obtiendrait avec une exécution sur une vraie machine MIPS.

Le processeur MIPS, son architecture et ses instructions sont présentés en annexe.

Deux modes de fonctionnement de l'émulateur sont prévus:

- Le mode dit **interactif** dans lequel chaque instruction MIPS est lue au clavier (entrée par l'utilisateur) et interprétée directement. La commande EXIT (non incluse dans le jeu des instructions MIPS) permet de quitter l'émulateur.

- Le mode dit **non-interactif** dans lequel l'émulateur va lire les instructions une à une dans le fichier dont le nom (chemin d'accès) est passé en paramètre.

Le projet sera réalisé sous linux. L'émulateur sera appelé en tapant la commande ***emul-mips*** :

emul-mips en mode interactif

Une fois cette commande lancée, le programme demande à l'utilisateur d'entrer une instruction, l'exécute et en demande une nouvelle et ainsi de suite jusqu'à lecture de EXIT. **Dans ce mode, il ne sera possible d'exécuter que des instructions en séquence.**

emul-mips prog_filename -pas en mode non-interactif

où *prog_filename* est le nom d'un fichier contenant le programme écrit en assembleur MIPS à exécuter. Si l'option *–pas* est spécifiée, le programme doit être exécuté pas à pas (c'est-à-dire, demander à l'utilisateur de valider le passage à l'instruction suivante).

Que ce soit en mode interactif ou bien dans le fichier programme, les instructions MIPS sont fournies (une par ligne) en langage assembleur (format texte).

L'émulateur commence par afficher la traduction de l'instruction (ou bien la totalité du programme) sous forme hexadécimale. Par exemple, *2402000a* est la représentation hexadécimale de l'instruction dont le format texte est *addiu $v0, $zero, 10*.

Ensuite, l'émulateur utilisera cette représentation hexadécimale pour exécuter l'instruction (ou le programme instruction par instruction). Après chaque instruction, l'émulateur va modifier l'état de l'architecture MIPS, c'est à dire les valeurs stockées dans les registres et dans la mémoire. **La précision de votre émulateur est votre priorité** principale. Plus précisément, assurez-vous que l'**état architectural est correctement mis à jour** après l'exécution de chaque instruction.

Enfin, l'émulateur affichera le nouvel état (valeur des registres, contenu de la mémoire) avant d'exécuter l'instruction suivante.

Afin de vérifier que votre émulateur fonctionne correctement, vous devez écrire plusieurs programmes à l'aide de toutes les instructions MIPS implémentées.

Travail demandé et planning

Il vous est demandé d'écrire une application C qui réalise un émulateur du processeur MIPS. Cet émulateur reproduira le comportement de chaque instruction[ii] et permettra à l'utilisateur d'exécuter des programmes MIPS en observant leurs sorties. **La prise en compte des *étiquettes* et des *directives* n'est pas demandée.**

Plus précisément, il vous est demandé d'effectuer dans l'ordre les tâches suivantes :

1. Ecrire des programmes MIPS qui vous serviront à tester votre application. Des exemples seront fournis dans un premier temps afin de familiariser avec la syntaxe de l'assembleur MIPS. Il est important de bien comprendre le rôle de chacune des instructions utilisée ! Cette étape est essentielle car elle vous permettra de prendre connaissance des spécifications MIPS. Vos programmes de tests devront contenir suffisamment de types d'instructions pour que vos tests soient efficaces.Un émulateur (projet de l'an passé) est disponible sur Chamilo afin de tester les codes.

2. Ecrire une première version de l'application permettant de traduire la forme textuelle des instructions MIPS en leur forme hexadécimale. Votre programme doit avoir deux paramètres : le nom du fichier contenant le programme MIPS et le nom du fichier qui contiendra le résultat de la traduction. A cette étape la traduction se fait d'un seul coup (pas de mode pas à pas ou interactif).

3. TESTER cette première version aves les tests fournis et construits lors de l'étape 1.

   Fournitures attendues : les programmes MIPS de l'étape 1, l'application que vous avez réalisée (zip du code source et de l'exécutable) et les résultats (sous forme de fichier texte) obtenus par les tests***. Ces fournitures sont à déposer sur chamilo au plus tard à la fin de la troisième séance consacrée au projet. A noter que l'évaluation sera faite à partir des exemples fournis à l'étape 1, ainsi que deux autres programmes qui ne seront pas donnés à l'avance. .***

4. Définir la structure de votre émulateur en termes de modules (un module est un ensemble cohérent de fonctions, définies à l'aide d'un fichier .h et d'un fichier .c). On définira au moins :

   a. Un module gérant la mémoire fournissant une fonction de lecture et une fonction d'écriture. Préciser la structure de données retenue pour représenter la mémoire

et justifier votre choix.

Un module gérant les registres du processeur (fonctions de lecture et écriture).**Fourniture attendue** : un document de quelques pages expliquant votre réflexion, ainsi que les headers (.h). ***Cette fourniture est à déposer sur chamilo au plus tard à la fin de la quatrième séance consacrée au projet. Des exemples de programmation modulaire seront donnés en cours. Ne pas hésiter à aller chercher plus d'exemples ou d'explications en ligne (https://openclassrooms.com/courses/apprenez-a-programmer-en-c/la-programmation-modulaire !).***

5. Réaliser l'application finale conformément aux décisions de conception que vous avez prises à l'étape précédente (notamment les modes pas à pas et interactif). Si vous pensez ne pas avoir le temps de réaliser toutes les instructions MIPS, il est conseillé d'en retenir un sous ensemble permettant d'écrire des programmes significatifs (avec conditions, boucles…).
6. TESTER la version finale.

   Fournitures attendues : l'application que vous avez réalisée (zip du code source et de l'exécutable) et les résultats (sous forme de fichier texte) obtenus par les tests. Un bilan est également demandé précisant **les tâches effectuées par chaque membre de votre groupe**. ***Ces fournitures sont à déposer sur chamilo au plus tard à la fin de la dernière séance consacrée au projet.***

7. *(Optionnel)* : modifier votre émulateur pour prendre en compte les étiquettes. Faire de nouveaux programmes MIPS de tests pour explorer cette nouvelle fonctionnalité.

   Fournitures optionnelles : la nouvelle version de l'application que vous avez réalisée (zip du code source et de l'exécutable), les nouveaux tests et les résultats (sous forme de fichier texte) obtenus par les *anciens et nouveaux* tests.

A la fin de chaque séance du projet, mettre à jour le fichier d'avancement et le déposer sur le répertoire dédié de chamilo.

Ressources

1. Document sur le processeur MIPS, extrait du sujet du projet d'informatique de Phelma, 2ème année (François Portet, Nicolas Castagne, Mathieu Chabanas, Laurent Fesquet).
2. Descriptif du jeu d'instruction MIPS.

Enseignants

- Ioannis Parissis (Ioannis.Parissis@grenoble-inp.fr)
- Guillaume Besset (guillaume.besset@grenoble-inp.fr)
- Adrien Rochedy(adrien.rochedy@gmail.com)

[i] Ce sujet est largement inspiré du projet d'informatique de Phelma, 2ème année (François Portet, Nicolas Castagne, Mathieu Chabanas, Laurent Fesquet).

[ii] Il est possible que toutes les instructions de l'annexe 2 ne soient pas prises en compte, si le temps vous manque. Dans ce cas, il faudra choisir soigneusement des instructions vous permettant, malgré tout, d'écrire des programmes non triviaux.