

DÉPARTEMENT D’INFORMATIQUE

Faculté des sciences

Université de Sherbrooke

Proposition de projet

Par

FÉLIX-ANTOINE OUELLET 09137551

Dans le cadre du cours :

Sujet choisi en informatique de systèmes

IFT 749

Travail présenté à :

Patrice Roy

Sherbrooke

17 septembre 2013

Sommaire

[Problème à résoudre 1](#_Toc367179102)

[Mise en contexte 1](#_Toc367179103)

[Approche proposée 2](#_Toc367179104)

[Tests 2](#_Toc367179105)

[Détails techniques 2](#_Toc367179106)

[Livrables 3](#_Toc367179107)

[Références 3](#_Toc367179108)

## Problème à résoudre

Contrairement au projet précédent, le projet proposé ici ne découle pas d’une demande ou d’un besoin spécifique énoncé par un tiers parti. En toute honnêteté, le but premier de ce projet est de s’initier au développement d’émulateur. À cela vient s’ajouter le désir d’approfondir mes connaissances sur les machines virtuelles ainsi qu’une technique de compilation bien particulière appelée recompilation dynamique. Bref, le projet a été choisi en fonction de mes intérêts personnels avant tout.

Par contre, il ne s’agit pas de l’unique but poursuivi dans ce projet. En effet, lors de la discussion avec Mikael Fortin qui a inspiré ce projet, ce dernier à mentionner qu’un émulateur serait un outil qu’il aimerait intégrer au cours IFT 209 – Programmation système. Son raisonnement est que travailler avec émulateur serait un bon moyen d’intéresser les étudiants du baccalauréat à tout ce qui a trait à la programmation de bas niveau. Conséquemment, l’autre grand but du projet est d’arriver, au moins, à un prototype qui pourrait être intégré dans un cours donné à l’Université de Sherbrooke.

## Mise en contexte

Avant d’entamer la discussion sur CHIP-16, il faut faire un détour pour présenter CHIP-8. CHIP-8 peut être décrit comme un langage de programmation interprété [3], mais, en regardant les spécifications [1], on aurait tendance à le décrire comme une architecture machine. Personnellement, je considère la seconde description comme étant la plus exacte. Aussi, chose intéressante à savoir, cette architecture n’a jamais existé physiquement, seulement au travers d’émulateurs. Pourquoi créer une architecture virtuelle? Tout simplement pour permettre aux (apprentis) développeurs de jeux vidéo d’accroître leur vélocité en travaillant avec un jeu d’instructions relativement petit. C’est d’ailleurs une des raisons qui m’a poussé vers l’émulation d’une architecture semblable. En effet, il est plus probable que je mène à bien ce projet si je m’attaque à une architecture comptant pas plus d’une cinquantaine d’instructions à émuler. De plus et cela était probablement dû aux limitations des technologies des années 70, l’architecture CHIP-8 a été spécifiée de façon à ne pas permettre trop de *hacks* qui litèrent les jeux de l’époque de la NES. Par conséquent, le développement de l’émulateur ne requerra pas trop de s’attarder aux « passes croches » qu’un programmeur créatif aurait pu insérer dans son code.

Étant donné que CHIP-8 se présente comme un excellent choix pour implémenter un premier interpréteur, le lecteur est en droit de se demander pourquoi le projet n’est tout simplement pas d’implémenter un émulateur pour cette architecture? Pour répondre à cette interrogation, j’avancerais que CHIP-16 a été créé dans le but de résoudre des inconsistances présentes dans la spécification de CHIP-8 et que les contrôles sous cette architecture sont beaucoup plus simples que celle sous l’architecture CHIP-8. Dernier argument en sa faveur, les programmes roulant sur CHIP-16 sont plus attrayants visuellement à cause d’une meilleure résolution et d’un affichage en couleur ce que n’offre pas CHIP-8.

## Approche proposée

Dans un premier temps, il faudra que je décide de la représentation de la machine virtuelle sur laquelle les programmes CHIP-16 vont s’exécuter et que je l’implémente. Par la suite, je devrai mettre en place un interpréteur qui aura comme responsabilités d’extraire et d’exécuter les instructions CHIP-16 qui constituent un programme sur cette architecture. Parallèlement au développement de l’interpréteur, il faudra que je développe une interface graphique qui me permettra d’acquérir le programme CHIP-16 à rouler et d’afficher le fruit de mon labeur. Pour ce qui est de l’aspect audio de la chose, elle est très basse dans ma liste de priorités pour l’émulateur. Ce choix est personnel, car je crois qu’il vaut mieux mettre mes efforts sur l’aspect visuel de l’émulateur si je veux arriver à un prototype qui pourrait être utilisable dans le cadre d’un cours. Ensuite, lorsque le noyau de l’interpréteur sera stable et testé, j’investirai mes efforts sur la gestion des entrées/sorties de l’émulateur, un travail qui devrait déjà être amorcé lors du développement de l’interface graphique. Ceci devrait me permettre de présenter un prototype viable à la fin de la session.

Si j’ai assez de temps, je souhaite développer un recompilateur dynamique. En gros, le travail de ce dernier serait de lire des blocs de bases [2] du programme CHIP-16 et de les traduire en instructions assembleurs pour la machine hôte. L’avantage du recompilateur sur l’interpréteur en est un de vitesse, car il peut consommer les instructions CHIP-16 plus rapidement et possiblement les traduire sous forme optimisée pour la machine hôte. D’ailleurs, si le temps le permet, j’aimerais tenter d’implémenter des optimisations à la génération de code. En restant réaliste, ceci ne risque pas de se produire étant donné que l’implémentation du recompilateur dynamique semble a priori relativement ardue à cause que celui-ci est soumis à des contraintes temps réel. En effet, son travail ne devra en aucun cas ralentir la cadence d’un programme. Sinon, une autre idée d’extension au projet si ce dernier avancerait trop bien serait de développer un débuggeur. L’idée vient du fait qu’un tel outil pourrait être grandement utile à des étudiants qui auraient à développer sur cette architecture dans le cadre d’un cours universitaire.

## Tests

Il apparaît assez évident que les principaux tests constitueront à vérifier qu’un maximum de jeux fonctionne sur l’émulateur produit. Il faudra donc se résoudre à jouer à un bon nombre de jeux pour vérifier la robustesse de l’émulateur. De plus, chaque instruction CHIP-16 sera soumise à des tests unitaires pour être certain qu’elles ont le comportement prescrit par la spécification de l’architecture. Dans l’optique qu’un recompilateur dynamique serait implémenté, il serait testé en vérifiant qu’il produit, à la base, le même code que l’interpréteur pour un bloc de code donné. Advenant que le projet avance très bien et que je puisse implémenter un recompilateur qui optimise le code produit, je ne sais pas de quelle manière je testerais son bon fonctionnement. Cependant, en restant réaliste, il est très peu probable que j’avance à un tel point dans mon projet dans le temps qui m’est alloué. Il en va de même pour un éventuel débuggeur; je ne peux pas me prononcer présentement sur la manière dont il serait testé.

## Détails techniques

Tout d’abord, il va sans dire que la réalisation d’un tel projet requiert l’utilisation d’un langage se prêtant à la programmation système. Mon choix de langage s’est donc arrêté sur C++. Par la suite, il faut décider du compilateur à utiliser. Pour ce projet, j’ai décidé d’utiliser Clang. Les raisons motivant ce choix sont qu’il offre, à mon avis, le meilleur diagnostic des erreurs de compilation et qu’il est le compilateur de C++ le plus strict que je connaisse. Ceci devrait m’assurer que si mon émulateur compile sous Clang, il devrait compiler sous les autres compilateurs tant que j’utilise uniquement du C++ standard. Au niveau graphique, mon choix n’est pas encore fait à savoir si je vais utiliser SDL ou Qt. Il en va de même au niveau de l’audio, même si, comme mentionné précédemment, ce n’est pas une priorité pour moi. Comme la première semaine en sera une d’exploration et de lecture, je pourrai fixer mon choix de bibliothèques à ce moment. De plus, je mettrai à profit la librairie Boost pour la mise en place de tests unitaires et plus si le contexte le requiert.

Finalement, s’il s’avérait que je me lance dans l’implémentation d’un recompilateur dynamique, ce dernier ciblerait le jeu d’instructions x86-64 étant donné que c’est celui sur lequel mon ordinateur personnel roule. Cela implique que l’émulateur cesserait d’être portable. Une solution qui s’offrirait alors serait d’utiliser LLVM pour générer des instructions en assembleur spécifique à l’architecture sur laquelle roule l’émulateur au lieu de les générer manuellement. L’intégration de LLVM serait par contre un travail colossal. On gardera donc cette idée en tête pour un futur projet.

## Livrables

À la fin du projet, je vise la livraison d’un émulateur pour CHIP-16 qui permette de jouer, au moins, à un jeu complet. À cela se joindra un article de style scientifique sur le développement de l’émulateur et les connaissances impliqués. Finalement, au niveau des critiques d’articles, je vise à critiquer un article traitant de la recompilation dynamique et un autre portant sur les machines virtuelles ou les émulateurs.

## Références

[1] *Cowgod’s* *Chip-8 Technical Reference v1.0*. <http://devernay.free.fr/hacks/chip8/C8TECH10.HTM>. consultée le 17 septembre 2013

[2] GNU Project. *Basic Blocks*. <http://gcc.gnu.org/onlinedocs/gccint/Basic-Blocks.html>. consultée le 17 septembre 2013

[3] Wikipedia. *CHIP-8*. <http://en.wikipedia.org/wiki/CHIP-8>. consultée le 17 septembre 2013