**Travail de recherche**

Travail présenté à

Yvon Latulippe

Dans le cadre du cours de

Système d’Exploitation et Réseau 2

Travail réalisé par

Charles Lachance

Et

Anthony Deschênes

**Histoire du chip 8**

Le chip 8 est un langage de programmation hexadécimal ayant été inventé dans le but de simplifier la création de jeux pour des systèmes 8 bits. Ce langage utilise une machine virtuelle, ce qui permet aux programmes chip 8 d’être portable dans la mesure où une machine virtuelle existe pour un système donné. Des interpréteurs chip 8 ont été conçu pour de nombreux systèmes : pour DOS, Windows, mac osx, linux, mais aussi pour la calculatrice graphique HP-48 et même en version site web.

**Qu’est-ce qu’un émulateur?**

Un émulateur est un programme imitant le fonctionnement physique d’un dispositif matériel. Plus un émulateur imite fidèlement le comportement du matériel émulé, plus les programmes vont s’exécuter de façon similaire à ce que donnerait une machine réelle.

**Qu’est-ce qu’une machine virtuelle?**

Une machine virtuelle est l’équivalent d’un ordinateur normal, mais qui existe seulement virtuellement. Donc, comme un ordinateur normal, il utilise des ressources comme un CPU et de la mémoire, mais sans être rattaché à un périphérique particulier. Cela rend donc la machine virtuelle très portable d’un ordinateur à un autre puisque, étant virtuelle, elle est facilement déplaçable. Il est également possible d’utiliser un Resource Pool pour donner simplifier encore plus le concept. Une Resource Pool est l’équivalent d’une sorte de boîte qui contient des CPU, de la mémoire, etc. Lorsque l’on créer une machine virtuelle, elle peut simplement utiliser un Resource Pool qui lui assigne les ressources dont elle a besoin.

Dans un ordinateur, une machine virtuelle peut être utilisée pour utiliser Linux ou un autre système d’exploitation en même temps que celui déjà présent sur l’ordinateur été qui utilise les mêmes ressources que celle de l’ordinateur. Dans le cas d’un système plus imposant, une machine virtuelle est littéralement virtuelle : contrairement à celle sur un ordinateur normal, où la machine se trouve en fait sur un ordinateur particulier, elle n’existe que grâce à des périphériques externes. Toutes les ressources, comme le CPU, la taille de la mémoire, la taille des disques durs, le nombre de disques durs, les adaptateurs réseau, peuvent donc aisément être modifiées grâce au fait que la machine n’existe pas concrètement, donc elle peut facilement changer la taille de sa mémoire. En fait, la seule chose qui ne change pas sur une machine virtuelle est l’espace de stockage qu’elle utilise. Il est donc impossible de diminuer l’espace disque utilisé dans un datastore, mais il est bien évident que l’on peut toujours l’augmenter. Également, elle peut toujours changer de datastore n’importe quand. Grâce à ses principes, il devient donc relativement aisé d’effectuer le déplacement d’une machine virtuelle d’un serveur à Québec vers un à Montréal.

Une machine virtuelle peut donc être comparée à un émulateur puisque, comme ce dernier, la machine virtuelle va imiter le comportement physique d’une machine réelle.

**Qu’est-ce qu’une machine virtuelle chip 8 émule?**

La chip 8 est communément représenté comme une machine virtuelle possédant une mémoire vive de 4096 octets, 16 registres de données de 8 bits, un registre d’adresse de 16 bits (I), un registre pointeur d’instructions (PC) de 16 bits, un registre pointeur de pile de 8 bits, un clavier hexadécimal à 16 touches, un écran monochrome de 64 par 32 pixels, un minuteur de délai et un minuteur de son. Les deux minuteurs sont décrémenté à une vitesse de 60 Hz.

**Qu’est-ce qu’un registre?**

Un registre est une petite mémoire de quelques octets faisant partie du processeur servant à faire transiter des données en passe d’être traitées.

**De quoi est composé un programme chip 8?**

Un programme chip 8, ou tout autre programme compilé en général, est composé d’instructions de base spécifique au processeur qui doit les lires. Les instructions peuvent être de taille fixe, comme pour la chip 8 ou l’architecture de processeur ARM, ou de taille variable, comme l’architecture de processeur x86. Chaque instruction est composée de l’opcode et de champs appelé opérandes servant de paramètres à l’opcode. Ainsi, l’opcode représente la nature de l’instruction alors que les opérandes permettent de déterminer d’autres informations pertinentes en lien avec l’opcode. Par exemple, pour la chip 8, dont les instructions ont une taille de 16 bits, l’instruction 0x1345 (souvent représenté comme étant 0x1NNN) se décompose ainsi : 0x1 (le opcode) 345 (ou NNN, un opérande).

La signification et la longueur des opérandes dépendent de l’opcode et de l’architecture du processeur. La plupart du temps, dans la documentation du processeur ou de la machine virtuelle les opcodes sont représentés avec des symboles à la place des opérandes. Par exemple, pour la chip 8, les symboles suivants sont utilisés : NNN (adresse 12 bits), NN (constante 8 bits), N (constantes 4 bits), X (identifiant de registre 4 bits), Y (identifiant de registre 4 bits). Par exemple, l’instruction 0x8XY0 doit être décomposée ainsi :

0x8XY0 : opcode

X : Identifiant du premier registre

Y : Identifiant du deuxième registre

**Comment fonctionne une machine virtuelle chip 8?**

À chaque cycle du processeur, le processeur obtient l’instruction située à l’emplacement pointé par le program counter. La machine virtuelle détermine quel est l’opcode de l’instruction, puis ses opérandes. Finalement, la machine vas interpréter l’instruction, puis incrémenter le program counter.

**Comment est divisée la mémoire vive d’une machine virtuelle chip 8?**

La chip 8 possède 4096 octets de mémoire. Sur les anciens ordinateurs, les 512 premiers octets étaient réservés pour la machine virtuelle et ne pouvaient donc pas être utilisé par un programme chip 8. Les 256 derniers octets étaient utilisés par l’affichage. Les 96 octets juste avant ceux utilisés par l’affichage étaient utilisés pour stocker la pile d’appelle et pour usage interne par la machine virtuelle. Dans les implémentations modernes, la machine réelle possédant généralement plus de 4096 octets de mémoire vive, la machine virtuelle n’est plus chargée dans les 512 premiers octets. Dans les implémentations modernes, les 512 premiers octets sont donc utilisés pour stocker la police de caractère.

**Quelques exemples d’instructions et de leur fonctionnement.**

* 0x00E0 : Efface l’écran. Dépend grandement de l’implémentation de la chip 8.
* 0x1NNN : Saute à l’adresse NNN. Ex : L’instruction 0x1453 vas changer la valeur du program counter pour 0x453, ce qui fait que la prochaine instruction à être interprété sera l’instruction situé à la case mémoire 0x453.
* 0x6XNN : Change la valeur du registre Vx pour la valeur NN. Ex : L’instruction 0x6289 va mettre dans le deuxième registre la valeur 89.
* 0x9XY0 : saute l’instruction suivante si le registre Vx a une valeur différente du registre Vy. Ex : L’instruction 0x9340 va incrémenter le program counter si le troisième registre est différent du quatrième registre.

**Qu’est-ce qu’un assembleur?**

Le terme assembleur est ambigu, car il désigne à la fois un langage d’assemblage et le programme permettant de le compiler. Un langage d’assemblage est un langage bas niveau qui représente le langage machine de façon lisible pour un être humain. Un assembleur est donc un programme permettant de transformer un programme écrit dans un langage d’assemblage en langage machine. On peut donc dire qu’il existe autant de langage d’assemblage que d’assembleur et qu’il existe autant d’assembleur que d’architecture de processeur puisque chaque assembleur doit pouvoir compiler en langage machine.

**Qu’est-ce qu’un désassembleur?**

Un désassembleur est un programme permettant de convertir un code compilé en langage d’assemblage. Dans le cadre d’un émulateur, cela est notamment utile pour des raisons de débogage puisque, grâce à un désassembleur, on peut décompiler un programme pour en comprendre son fonctionnement et s’assurer qu’il correspond bien à ce que l’on s’attend.

**Comment fonctionne un désassembleur?**

Pour décompiler un programme, le désassembleur va lire chaque instruction une par une et va déterminer à quel opcode l’instruction correspond et ses opérandes. Par la suite, le désassembleur va convertir l’opcode et ses opérandes vers une forme textuelle compréhensible par un être humain, puis il va l’écrire dans une sortie prédéterminé (souvent un fichier).