Simulation du Modèle d'Exécution d'un Processeur

Benkechida Isaac

7/04/2025

Contents

1 Introduction

Pour devenir un excellent informaticien, il est nécessaire de comprendre comment fonctionnent les machines sur lesquelles on travaille. C'est pour cela que nous avons décidé de développer un modèle d'exécution d'un processeur. Cela nous permettra de concrètement savoir ce qui se passe "sous le capot" d'un ordinateur.

2 Description de l'implémentation de la phase 2

2.1 Registres

Contrairement à la phase précédente du projet, nous avons implémenté 5 registres dont 4 permettent de stocker les données manipulées et le dernier servira à contenir l'adresse du sommet de la pile (cf. section mémoire). Les registres généraux ont été implémentés avec une map permettant d'accéder en une complexité O(1).

2.2 Analyse d'instruction

La manière d'extraire l'opcode ne diffère pas de la phase 1. Cependant, nous avons implémenté des fonctions capables d'extraire le 2ème ou 3ème paramètre et d'analyser le type de ce dernier qui peut être soit un uint16_t, un uint8_t ou un string. Pour faire cela, nous avons créé une structure composée d'un variant (union sécurisée) ainsi qu'un flag permettant de savoir si le paramètre était un nombre ou un caractère après extraction.

2.3 Mémoire

Une mémoire de 256 bytes a été implémentée, stockant des valeurs de type uint16_t en grand boutisme. Cela veut dire qu'il faut 2 bytes pour stocker chaque donnée (les msb fort dans byte 1 et lsb dans byte 2).

Les 16 premiers bytes de la mémoire sont dédiés à la pile. Pour manipuler cette dernière, la fonction push() ajoute une valeur au sommet de la pile et la fonction pop() retire la valeur au sommet de la pile et retourne cette valeur.

Des fonctions permettant l'accès (read()) des données au sein de la mémoire et l'écriture (write()) ont également été implémentées. Cela permet d'éviter d'accéder au-delà de l'espace alloué à la mémoire.

D'autres fonctions auxiliaires et structures ont été implémentées pour faciliter l'encodage et le décodage des données.

3 Résultat de l'exemple

Les registres a et b sont respectivement initialisés à 150 et 130. La valeur 150 est poussée dans la pile, suivie par la somme de 130 et 150 (280) stockée dans a. Dans la pile, la valeur de a (280) est stockée à l'adresse 100 (les bits de poids forts à l'adresse 100 et les bits de poids faibles à l'adresse 101). La valeur de b est stockée à l'adresse 101. Les bits de poids faible de 280 sont écrasés par les bits de poids fort de 130. En affichant ce qu'il y a à l'adresse 100, nous obtenons 256, et à l'adresse 101, nous obtenons 130. Le résultat des instructions dans la console est donc :

l'autre résultat apparaît si nous décidons de stocker les données dans la mémoire en little endian (d'abord lsb puis msb).