Rapport 1.1

Codage des nombres entiers

Introduction

Les machines ne sont pas capables d'appréhender une information sans qu'elle ait été convertie au préalable en binaire. En effet, les langages de programmation font en quelque sorte le pont entre les humains et les ordinateurs en leur permettant de communiquer malgré tout, grâce à cette conversion. Le langage assembleur permet notamment de comprendre quelles comment fonctionnent les opérations arithmétiques, typiquement que la machine a une pérférence notable pour les additions.

Ce rapport présente une application en HTML5/Javascript qui permet la conversion entre nombres entiers et binaires, ainsi que de faire des opérations arithmétiques en utilisant les nombres en base 2.

Spécificités

On a choisi de ne s'occuper que des nombres signés, donc lorsqu'on a une contrainte de 8 bits, les limites ne se trouvent plus à 0 à 255, mais bien à -128 à 127. Il nous est apparu plus simple de calculer les nombre binaires en utilisant des tableaux au lieu de choisir un moyen alternatif.

Présentation du code

La partie HTML s'occupe exclusivement de l'affichage de l'interface ainsi que de l'appel des fonctions lorsqu'un évènement a été amorcé, comme la pression d'un bouton.

La conversion entier - binaire/ binaire - entier se fait en temps réel.

La partie Javascript s'occupe des opérations arithmétiques et de la conversion. Dans un premier temps, on s'occupe de la gestion des champs du type vérification que le remplissage ait été fait correctement et du renvoi aux fonctions de conversion lors du dit remplissage des champs. Viennent ensuite les fonctions s'occupant des opérations arithmétiques.

Listing des fonctions

`resetInput` : La fonction permet de vider les champs textes.

`decimalToBinary` : La fonction s'occupe de convertir les nombres décimaux en nombres binaires en utilisant successivement une division par 2 et un modulo 2. Le quotient permet de continuer les division et le modulo de stocker le reste dans le tableau des binaires. Lorsqu'on a affaire à un nombre négatif, on effectue un complément à 2. On finit par tester si le nombre est bien dans les limites évoquées plus haut.

`binaryToDecimal` : La fonction s'occupe de convertir les nombres binaires en nombres décimaux. On commence par tester si le nombre est négatif, si c'est le cas, on effectue un complément à 2 et on met le flag à `true` pour avertir qu'il y aura un signe négatif devant le nombre. Sinon, on passe directement au calcul. On reprend chaque case du tableau en la multipliant par 2 fois la puissance de l'indice de la case testée.

`isInputNotEmpty` : La fonction teste si les champs sont remplis.

`isBinaryInputValid` : La fonction teste si les champs contenant les nombres binaires ne sont pas vides.

`twoComplement` : La fonction s'occupe du complément à 2[1]. On commence par inverser tous les bits du tableau qui a été récupéré, puis on y ajoute un bit.

`addAndSub` : La fonction s'occupe d'effectuer les additions ainsi que les soustractions[2]. Le flag entré en paramètre permet de savoir quelle sera l'opération utilisée. On gère aussi bien les nombres décimaux que les nombres binaires, cependant, ce dernier nécessite en plus la fonction `binaryAdd` expliquée ci-dessous.

`binaryAdd` : La fonction s'occupe de l'addition de deux nombres binaires. On utilise un flag qui permet de prendre en compte la retenu ainsi que de savoir si on est en overflow. Grâce à ça, on peut simplifier les teste en ne faisant qu'additionner la case testée du premier tableau avec la case correspondante du deuxième tableau. Si le résultat dépasse 1, en sachant qu'on peut avoir 3 au maximum, on laisse le flag à 1 et on fait un modulo.

`multiplication` : La fonction s'occupe de la multiplication d'un nombre par un autre. Le principe est qu'on fait une succession d'additions et qu'on shift le tableau. Le résultat est cohérent mais n'est pas il y reste un petit problème de formatage.

[1] Le complément à 1 permettant d'avoir un 0 négatif et un 0 positif, cette alternative a été écartée.
[2] Comme mentionné plus haut, les machines ont une préférence pour les additions, du coup lorsqu'on soustrait un nombre à un autre, on le convertit en négatif et on fini juste par faire une addition.