Rapport intermédiaire
Microprocesseur i<3bf++

Mihai Dusmanu, Caroline Etienne, Clément Pascutto

Dimanche 13 Décembre 2015

Résumé

Nous avons choisi d'implémenter un processeur dont l'assembleur est inspiré du langage *brainfuck*, augmenté par des fonctionalités supplémentaires améliorant sa rapidité, sans pour autant ajouter d'instructions.

Quelques choix sur les données

La taille des mots choisie pour le processeur est de 16 bits ; les entiers sont représentés en *Big Endian* et sont non signés.

Assembleur

Brainfuck

On reprend les instructions du langage brainfuck de 1993 : le processeur de départ est une machine à compteurs.

Instruction	Effet
>	Incrémente le pointeur de données
<	Décrémente le pointeur de données
+	Incrémente le compteur courant
−	Décrémente le compteur courant
,	Affiche la valeur du compteur courant
.	Lit et enregistre une valeur dans le compteur courant
[	Saute à l'instruction après le] correspondant si la valeur du compteur courant est nulle, sinon saute à l'instruction suivante
]	Retourne à l'instruction [correspondante

Améliorations : bf++

Pour améliorer l'efficacité du processeur, on ajoute des fonctionalités rétrocompatibles qui seront implémentées directement en hardware, sans ajouter d'instructions. Elles permettent également de rendre le code plus lisible.

Dans le tableau suivant, (ch) est un caractère de $\{<, >, +, -\}$.

Instruction	Condition	Effet	Exemple
$n(ch)$	$n \in \mathbb{N}$ $0 \leq n < 2^{16}$	n fois (ch)	15+ ajoute 15 au compteur courant
$\#(ch)$	la valeur du compteur courant n'est pas nulle	k fois (ch), où k est la plus grande puissance de 2 plus petite que la valeur du compteur courant	si la valeur du compteur courant est 6, #+ ajoute 4 à ce compteur
$\$(ch)$		k fois (ch), où k est la dernière puissance de 2 calculée par un #, par défaut 0.	si la mémoire est $(7, 0, 0, 0, 0)$, et le compteur courant est le premier, # > $+ transforme la mémoire en $(7, 0, 0, 0, 4)$

Architecture du i<3bf++

Architecture générale

Le microprocesseur est constitué des composants suivants :
- Une mémoire ROM contenant les instructions du programme (voir paragraphe suivant pour l'encodage).
- Une table de routage composée d'une ROM qui convertit les instructions en bits de contrôle pour les autres composants.
- Une unité arithmétique (UA) capable d'effectuer les opérations PLUS, MOINS et de calculer la plus grande puissance de 2 inférieure à un entier.
- Une mémoire RAM contenant 2^{16} compteurs.
- Un pointeur indiquant l'adresse du compteur courant dans la RAM.
- Un registre de 16 bits contenant la valeur du dernier # calculé.

Flags

L'UA sera capable de lever plusieurs flags :
- Overflow
- Underflow
- Zero : contient si le résultat de la dernière opération est égal à 0.

Encodage des instructions (compilation de l'assembleur)

Les instructions stockées dans la ROM sont des mots de 20 bits :

– 3 bits sont réservés à l'encodage des 8 opérations de base qui sont encodées de façon à faciliter le routage :

>	<	+	-	[	]	.	,
000	001	010	011	100	101	110	111

– 1 bit est réservé à la reconnaissance des caractère # et $.
Ce bit vaut 1 si l'un des deux caractères est présent. L'argument vaut alors dans ce cas 0 si # est présent, et 1 si $ est présent.
Si le bit de contrôle vaut 0, le paramètre vaut l'argument de l'instruction.
– 16 bits de paramètres sont réservés à l'argument de l'instruction (16 dans 16 >).

Afin d'éxécuter l'opération [correctement, on la précèdera dans la compilation par un instruction 0+, qui ne modifie pas les données mais calcule le flag Zero qui sert de condition de jump.

De même, pour effectuer correctement #+, on le compile par # + $+, la première partie #+ servant au calcul de la plus grande puissance inférieure au compteur en cours, sans modifier les données.