Système digital – rapport intermédiaire

Architecture, jeu d'instructions

Théophile Bastian, Noémie Cartier, Nathanaël Courant

8 décembre 2015

Résumé

Nous avons choisi d'utiliser une architecture proche d'ARM, en réduisant toutefois le jeu d'instructions.

1 Architecture

L'architecture que nous avons choisie est fortement inspirée d'ARM, dans le but d'une part de se rapprocher d'une architecture réelle, et d'autre part de pouvoir compiler du code vers notre assembleur en utilisant un compilateur standard (comme gcc par exemple), quitte à retravailler l'assembleur fourni pour se ramener à un jeu d'instructions gérées.

1.1 Mémoire

Le processeur gère un nombre de registres à définir. L'accès à la ROM est impossible (§1.2) ; l'accès à la RAM est limité à un accès lecture et un accès écriture par cycle (en particulier, une seule RAM est gérée). Les nombres seront représentés sur 64 bits, concrètement gérés par des nappes de 64 fils. Notons que le simulateur netlist gère les opérations bitwise directement sur les nappes de fils.

Certains registres seront déclarés entrées ou sorties du circuit netlist, ce qui permettra de gérer les entrées/sorties.

1.2 Mémoire d'instructions

La ROM contiendra et représentera exclusivement les instructions du programme assemblé. En particulier, l'accès à la ROM est donc impossible du point de vue de l'utilisateur, car le processeur y accède déjà à chaque cycle pour lire l'instruction à effectuer.

1.3 Arithmétique

Les opérations arithmétiques gérées sont uniquement l'addition, la soustraction et les opérations *bitwise* standard. Cela sera suffisant : l'horloge doit seulement s'incrémenter, et retirer 60 ou 24. La sortie sur 7 segments se fait quant à elle par recherche dans une table.

2 Assembleur