

Projet de simulateur de microprocesseur

Nathan Boyer, Matteo Torrents, Antoine Cuvelier, Grégoire Le Corre

12 décembre 2023

Introduction

On cherche à coder un simulateur de microprocesseur en Ocaml.

Pour cela, on va utiliser un simulateur de netlist qui exécutera des netlists écrites en miniJazz.

Cette netlist peut exécuter les opérations arithmétiques de base (and, or etc ...) sur un bit, manipuler des bus de bits (splice, concat ...)

De plus, l'instruction reg permet de se prendre la valeur du cycle précédent d'une variable. L'instruction RAM permet d'écrire et de lire dans la mémoire (par soucis de réalisme, on considérera qu'il n'existe qu'une seule RAM). L'instruction ROM, quant à elle, permet de lire le contenu d'un fichier composé de 0 et de 1

On va alors programmer un langage assembleur qui nous permettra de communiquer avec notre microprocesseur. Notamment, on programmera une horloge graphique (càd qu'on s'occupera de l'affichage des pixels depuis notre microprocesseur) qui compte en année, mois, jour, heure, minute et seconde jusqu'à l'année 9999. pour tester notre microprocesseur.

1 Architecture générale

On opte pour une architecture 16 bits avec des instructions définies sur 32 bits.

L'horloge du microprocesseur utilise l'horloge de l'ordinateur pour connaître l'heure exacte, l'ordinateur définissant préalablement le framerate du microprocesseur en lui indiquant chaque changement de frame dans un endroit désigné de la ROM.

La prise d'input se fait en donnant le dernier input dans un endroit désigné de la ROM, jusqu'à l'arrivée du prochain input. Ceci impose une limite de 1

input par cycle, ce qui ne pose pas de problème en pratique (on ne considère pas le maintien d'une touche).

Tous les calculs du microprocesseur seront faits sur des registres de 16 bits.

Il y a en tout 16 registres.

2 Représentation des instructions

Le microprocesseur lit les instructions qu'il doit exécuter depuis un fichier (lu avec l'instruction ROM).

Chaque instruction fait 32 bits, peut prendre en argument jusqu'à 2 numéros de registre et une constante. On doit donc stocker une instruction sur deux registres, le deuxième correspondant toujours à une constante (non initialisée si l'instruction n'utilise pas de constante). Une instruction est donc découpée en

- 4 bits du code de l'instruction
- deux blocs de quatre bits renseignant les registres en argument
- un deuxième registre de constante

3 Du langage assembleur à la ROM

Un parseur menhir lit les instructions et écrit dans un fichier lisible par la ROM les différentes instructions.

Voici une liste des différentes instructions, de leur code en binaire, et de leur effet.

nom	code	effet
stop	0000	stoppe le programme
add r1 r2	0010	$r1 \leftarrow r1 + r2$
sub r1 r2	0110	$r1 \leftarrow r1 - r2$
movr r1 r2	1010	$r1 \leftarrow r2$
movc r1 c	0011	$r1 \leftarrow c$
jump r1 r2 c	0001	on saute à la ligne c si $r1 = r2$
getram r1 r2 c	0101	$r1 \leftarrow \text{RAM}(r2 + c)$
setram r1 r2 c	1001	$\text{RAM}(r2 + c) \leftarrow r1$
rom r1 r2 c	1101	$r1 \leftarrow \text{ROM}(r2 + c)$

Le bit de poids faible indique si l'instruction contient une constante, le deuxième bit de poids faible si l'instruction appelle un ou deux registres. On numérote ensuite les instructions de chaque catégorie en les énumérant.

4 Affichage graphique

L’affichage est un display de 320 x 624 pixels noirs et blancs : la résolution est en blocs/megapixels de 8x8 pixels, chaque caractère étant contenu dans un bloc de 16 x 8 megapixels, avec un espace horizontal de 2 megapixels, et un espace vertical (interligne) de 8 megapixels.

L’affichage de l’horloge se fait dans cette fenêtre. La date est affichée sous la forme

heures : minutes : secondes
jour / mois / année

où les passages en italiques représentent les valeurs machine. Concrètement on alloue une partie de la mémoire pour l’affichage ; chaque bit en mémoire représente un mégapixel (noir ou blanc). Le microprocesseur modifie cette espace mémoire à chaque cycle et Ocaml affiche la fenêtre bit à bit.