

# Processeu NONO 1et 2

Cédric BOIS      Benjamin SIENTZOFF

16 décembre 2014

## Table des matières

<b>1</b>	<b>Réalisation de Nono-1</b>	<b>4</b>
1.1	Opcode des instructions . . . . .	4
1.2	L' unité arithmétique et logique . . . . .	5
1.3	Le contrôleur de saut . . . . .	7
1.4	Le décodeur d'instructions . . . . .	8
1.5	La sélection des registres . . . . .	9
1.6	Le banc de registres . . . . .	9
<b>2</b>	<b>Processeurs Nono-1 et Nono-2</b>	<b>11</b>
2.1	Nono-1 . . . . .	11
2.2	Nono-2 . . . . .	11

## Introduction

Dans le cadre du cours intitulé *Architecture des ordinateurs*, nous devons recréer un processeur Nono-1. Par la suite, ce processeur sera modifier pour devenir Nono-2. Ce rapport retrace comment nous avons réalisé ces processeurs MIPS.

Les circuits électroniques présentés sont produits avec le logiciel *Logisim*. Ces circuits et les différents fichiers permettant notamment de programmer le processeur sont fournis avec la version numérique de ce rapport. Les images RAM peuvent être directement chargées dans la RAM des processeurs Nono. Ces images correspondent aux programmes compilés pour ces architectures et peuvent être exécutés directement dans *Logisim*.

Dans une première partie, nous présentons les différents sous-circuits composants le processeur Nono-1. Une seconde partie présente son fonctionnement global et les modifications apportées à Nono-1 pour implémenter les fonctions de Nono-2.

# 1 Réalisation de Nono-1

## 1.1 Opcode des instructions

Nono-1 et Non-2 sont des processeurs utilisant l'assembleur MIPS. Les instructions disponibles sur Nono-1 sont présentées au tableau de la figure 2. On remarque que les instructions reconnues sont relativement restreintes. Ces instructions sont de trois formats différents comme on peut le voir à la figure 1<sup>1</sup>.

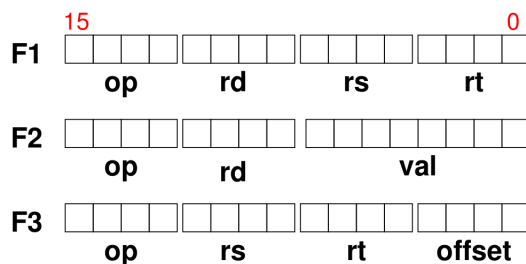


FIGURE 1 – Formats des instructions

**Le Format F1** Le format F1 est composé de quatre paquets de bits. Le premier est sur quatre bits, il correspond au code de l'instruction, et c'est le cas pour tous les formats d'instructions. Les trois paquets suivants, sur quatre bits. Ce format est utilisé typiquement pour des opérations faisant intervenir trois registres. Le premier correspond à la destination du résultat et les deux suivants aux registres contenant les opérantes.

**Le Format F2** Le format F2 est composé de trois paquets de bits. Le premier est sur quatre bits, il correspond au code de l'instruction. Les 4 bits suivants correspondent à un nom de registre et les 8 derniers à une valeur immédiate. Ce format est typiquement utilisé pour l'instruction *li*.

**Le Format F3** Le format F3 peut être divisé en quatre parties. C'est le format utilisé pour les sauts. Les quatre bits correspondent à l'opcode de l'instruction. Le paquet des quatre bits et le suivant constitué des quatre autres bits suivant correspondent à des noms de registres. Enfin les derniers bits (au nombre de quatre) correspondent à un offset. Pour les sauts, cela correspond à l'adresse de l'étiquette où effectuer le saut.

Le choix des opcodes n'a pas été fait au hasard. En effet, en regardant le nombre d'instructions pour les sauts et le nombre d'opérations faisant appel à l'unité arithmétique et logique du processeur, on s'aperçoit qu'ils peuvent être divisés en deux groupes. On a donc regroupés les opcodes en trois groupes. Le premier correspond aux opcodes qui commencent par un 1, ce sont les instructions qui font appel à l'UAL. Le second groupe, les opcodes commencent par un 0,

1. Tiré du sujet du projet rédigé par M. Frédéric GOULARD

Instruction et paramètres	Format	Opcode
add $r_d, r_s, r_t$	$F_1$	1000
sub $r_d, r_s, r_t$	$F_1$	1001
or $r_d, r_s, r_t$	$F_1$	1010
and $r_d, r_s, r_t$	$F_1$	1011
not $r_d, r_s$	$F_1$	1100
shl $r_d, r_s, r_t$	$F_1$	1101
shr $r_d, r_s, r_t$	$F_1$	1110
li $r_d, val$	$F_2$	1111
halt	$F_1$	0000
b <i>offset</i>	$F_3$	0001
beq $r_s, r_t, offset$	$F_3$	0010
bne $r_s, r_t, offset$	$F_3$	0011
bge $r_s, r_t, offset$	$F_3$	0100
ble $r_s, r_t, offset$	$F_3$	0101
bgt $r_s, r_t, offset$	$F_3$	0110
blt $r_s, r_t, offset$	$F_3$	0111

FIGURE 2 – *Opcode* des différentes instruction du processeur NONO 1

correspondent aux sauts. Enfin le dernier groupe est composé des autres instructions. Citons notamment les opcodes 0000 et 1111. Le tableau des instructions, leur format et les opcodes correspondants est présenté à la figure 2.

Maintenant que vous avons défini nos opcodes, il est temps de concevoir les circuits électroniques composants le processeur NONO 1. Commençons par l'Unité Arithmétique et Logique.

## 1.2 L'unité arithmétique et logique

L'Unité Arithmétique et Logique, abrégé UAL, permet de faire des calculs basiques (additions, divisions, décalages de bits, etc.). Elle effectue les calculs sur huit bits. L'UAL a trois entrées. La première sur trois bits permet de préciser le code de l'opération à utiliser. Les deux autres entrées sur huit bits sont les opérantes de l'opération.

En sortie, sur *ouput* on peut lire le résultat de l'opération. Il y a également quatre drapeaux comme détaillés ci-dessous.

CF pour *Carrie Flag* est le drapeau levé lorsque que l'opération génère une retenue.

ZF pour *Zero Flag* est armé lorsque que le résultat comporte uniquement des 0.

OF pour *Overflow Flag* est un drapeau levé lorsque on dépasse la capacité des nombres représentable par la machine.

SF pour *Sign Flag* est levé lorsque le résultat a son bit de poids fort à 1, c'est un nombre signé.

**Overflow Flag** Pour armer le drapeau *OF*, on fait le tableau KARNAUGH de la figure 4 d'où on tire l'équation 1. Ainsi on peut faire facilement le circuit qui

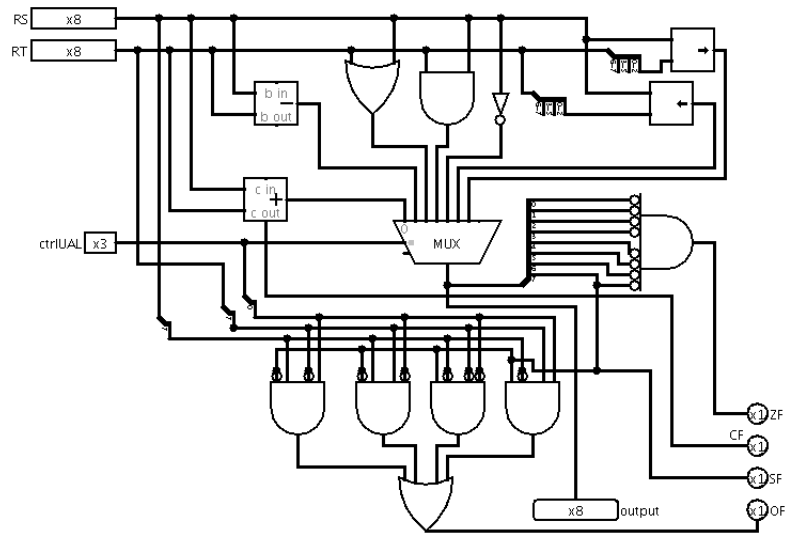


FIGURE 3 – Schéma électronique de l'Unité Arithmétique et Logique

$s_1 e_1 \backslash e_2 b_1$	00	00	11	10
00	0	0	0	0
01	0	1	0	1
11	0	0	0	0
10	1	0	1	0

FIGURE 4 – Tableau de KARNAUGH pour le drapeau  $OF$

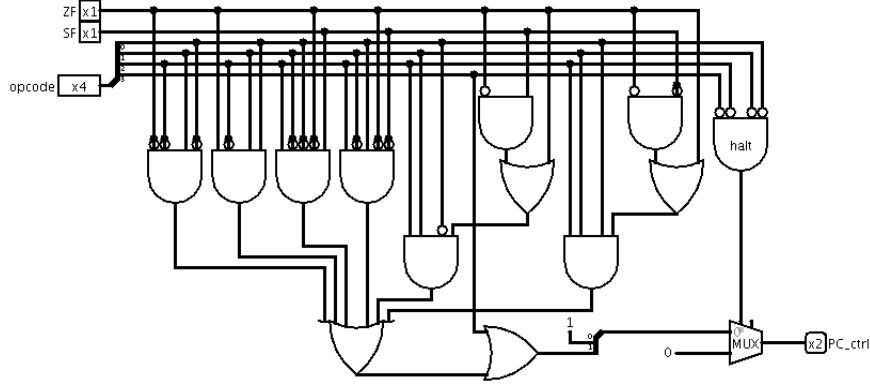


FIGURE 5 – Schéma électronique du contrôleur de sauts

contrôle la levée du drapeau.

$$OF = \neg s_1(e_1(\neg e_2.b_1 + e_1.\neg b_1)) + s_1(\neg e_1(\neg e_2.\neg b_1 + e_2.b_1)) \quad (1)$$

Dans l'équation 1, les variables  $e_1$  et  $e_2$  correspondent aux bits de poids fort de  $Rs$  et  $Rt$ . Les variables  $s_1$  et  $b_1$  correspondent respectivement au bit de poids fort et le bit de poids faible de  $ctrlUAL$ . On choisit ces bits car ils nous renseignent sur la nature du résultat. Par exemple lorsqu'on soustrait un nombre négatif et un nombre positif, si le résultat est positif, c'est qu'un *overflow* est survenu.

### 1.3 Le contrôleur de saut

Le second circuit électronique composant le processeur est le contrôleur de sauts. Son rôle est de mettre à jour le pointeur d'instructions  $PC$  (pour *Program Counter*) en fonction du résultat de l'opération que vient d'effectuer l'UAL pour le cycle suivant. Le saut est déterminé en fonction des indicateurs que le circuit a en entrée. C'est-à-dire  $SF$  et  $ZF$ . Pour ce là, on fait une table de vérité présente à la figure 6. Une fois la table remplie, on en tire l'équation suivante :

$$\begin{aligned} C_1 = & \neg C_1.O_1.\neg O_0.\neg ZF \\ & + \neg O_2.O_1.O_0.ZF \\ & + O_2.\neg O_1.O_0.\neg ZF.SF \\ & + O_2.O_1.\neg ZF.\neg SF \\ & + O_2.O_1.O_0.(ZF + \neg ZF.SF) \\ & + O_2.O_1.O_0.(ZF + \neg ZF.\neg SF) \end{aligned} \quad (2)$$

On traduit alors l'équation en circuit et on obtient le circuit de la figure 5.

Dans l'équation 2, la variable  $O_i$  correspond au  $i$ -ème bit de l'opcode. La variable  $C_1$  nous renseigne lorsqu'il faut faire un saut. Enfin,  $ZF$  et  $SF$  correspondent aux drapeaux levés par l'UAL.

Penchons-nous maintenant sur le décodeur d'instructions.

$O_2$	$O_1$	$O_0$	ZF	SF	$C_1$
0	0	1	×	×	0
0	1	0	0	×	1
0	1	0	0	×	0
0	1	1	1	×	0
0	1	1	1	×	1
1	0	0	0	0	0
1	0	0	0	1	1
1	0	0	1	0	0
1	0	0	1	1	0
1	0	1	0	0	1
1	0	1	0	1	0
1	0	1	1	0	0
1	0	1	1	1	0
1	1	0	0	0	0
1	1	0	0	1	1
1	1	0	1	0	1
1	1	0	1	1	1
1	1	1	0	0	1
1	1	1	0	1	0
1	1	1	1	0	1
1	1	1	1	1	1

FIGURE 6 – Table de vérité du contrôleur de sauts

#### 1.4 Le décodeur d'instructions

Le décodeur d'instructions permet de déterminer quel opération va effectuer le processeur pour le cycle à venir. C'est lui qui lit les opcodes déterminés précédemment.

**Gestion des sauts** Lorsque le décodeur de sauts lis une instruction de type *b*, il se contente d'appeler l'Unité Arithmétique et logique pour faire une soustraction. Il arme également la sortie *isJMP*.

Le schéma électronique du décodeur d'instructions est présenté à la figure 7. La partie qui décode le signal *ctrlUAL* correspond au tableau de KARNAUGH de

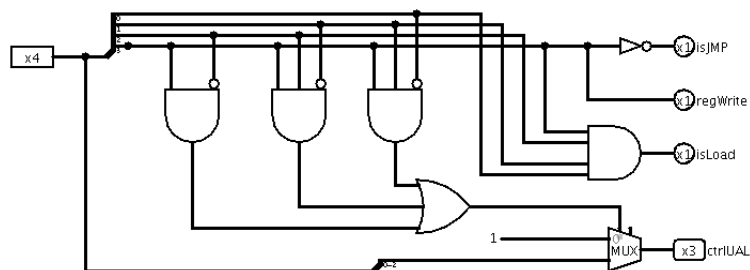


FIGURE 7 – Schéma électronique pour le décodeur d'instructions



$b_3b_2 \backslash b_1b_0$	0	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	0	1
10	1	1	1	1

FIGURE 8 – Tableau de KARNAUGH pour le décodage d'instructions.

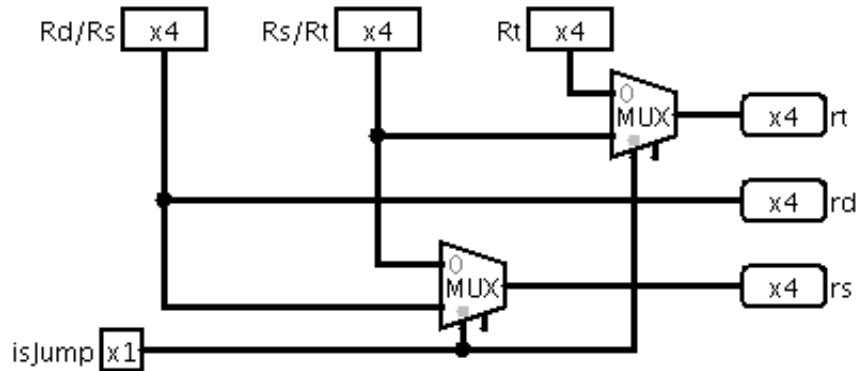


FIGURE 9 – Schéma électronique pour la sélection de registres

la figure 8 duquel on extrait l'équation suivante :

$$ctrlUAL = b_3 \cdot \neg b_2 + b_3 \cdot b_2 \neg b_1 + b_3 \cdot b_1 \neg b_0 \quad (3)$$

Cette équation nous permet alors de réaliser le circuit électronique décodant *ctrlAUL*.

## 1.5 La sélection des registres

intro, explications

## 1.6 Le banc de registres

intro, explications

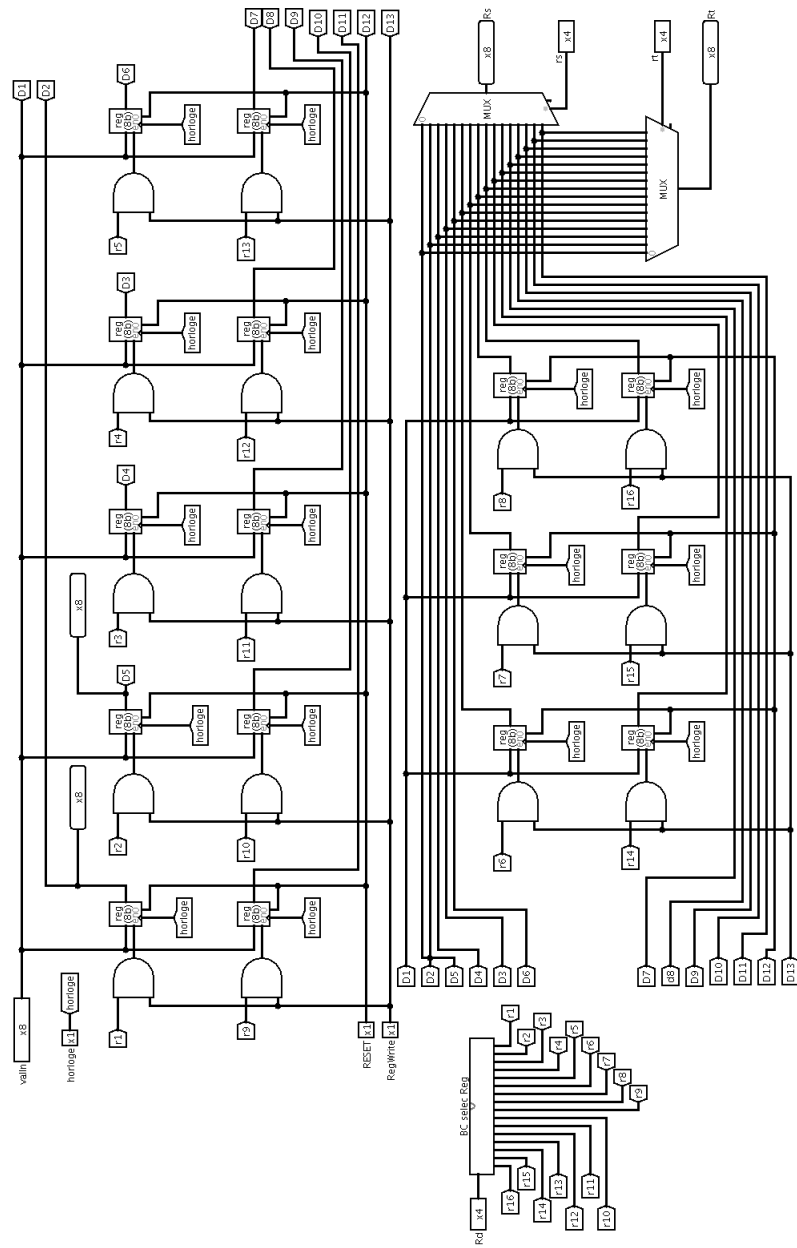


FIGURE 10 – Schéma électronique pour le banc de registres

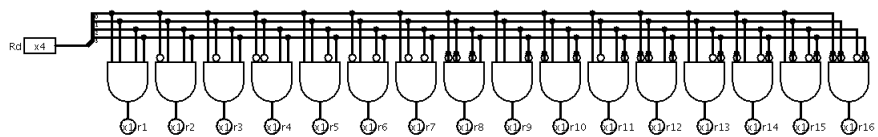


FIGURE 11 – Schéma électronique pour le sélecteur du banc de registres

## **2 Processeurs Nono-1 et Nono-2**

### **2.1 Nono-1**

### **2.2 Nono-2**

## Conclusion

je conclu

## Table des figures

1	Formats des instructions . . . . .	4
2	<i>Opcode</i> des différentes instruction du processeur NONO 1 . . . . .	5
3	Schéma électronique de l'Unité Arithmétique et Logique . . . . .	6
4	Tableau de KARNAUGH pour le drapeau <i>OF</i> . . . . .	6
5	Schéma électronique du contrôleur de sauts . . . . .	7
6	Table de vérité du contrôleur de sauts . . . . .	8
7	Schéma électronique pour le décodeur d'instructions . . . . .	8
8	Tableau de KARNAUGH pour le décodage d'instructions. . . . .	9
9	Schéma électronique pour la sélection de registres . . . . .	9
10	Schéma électronique pour le banc de registres . . . . .	10
11	Schéma électronique pour le sélecteur du banc de registres . . . . .	10