Université de Cergy-Pontoise

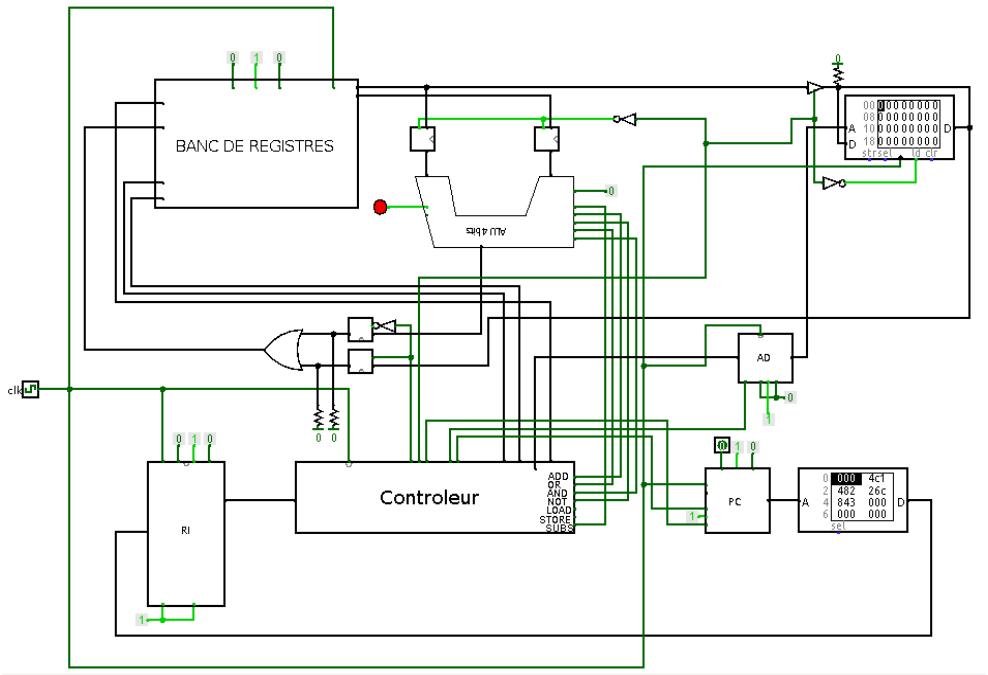
RAPPORT

pour le projet d'Architecture des Ordinateurs
Licence d'Informatique deuxième année

Conception d'un processeur 4 bits

rédigé par

Dorian CHENET, Alexis CHOLLET



Mai 2015

Table des matières

Table des figures

Remerciements

Remerciements à Mr Jordan LORENDEL pour sa conduite des cours magistraux et TD d'architecture des ordinateurs ce semestre, nous espérons que cela a été un plaisir autant pour lui que pour nous.

Ce projet d'Architecture des Ordinateur avait pour but la réalisation d'un processeur 4 bits en utilisant le logiciel Logisim. Ce processeur, réalisé à partir de composants électroniques devait être capable d'exécuter une suite d'instructions données par des mots binaires.

1 Aspect général du processeur

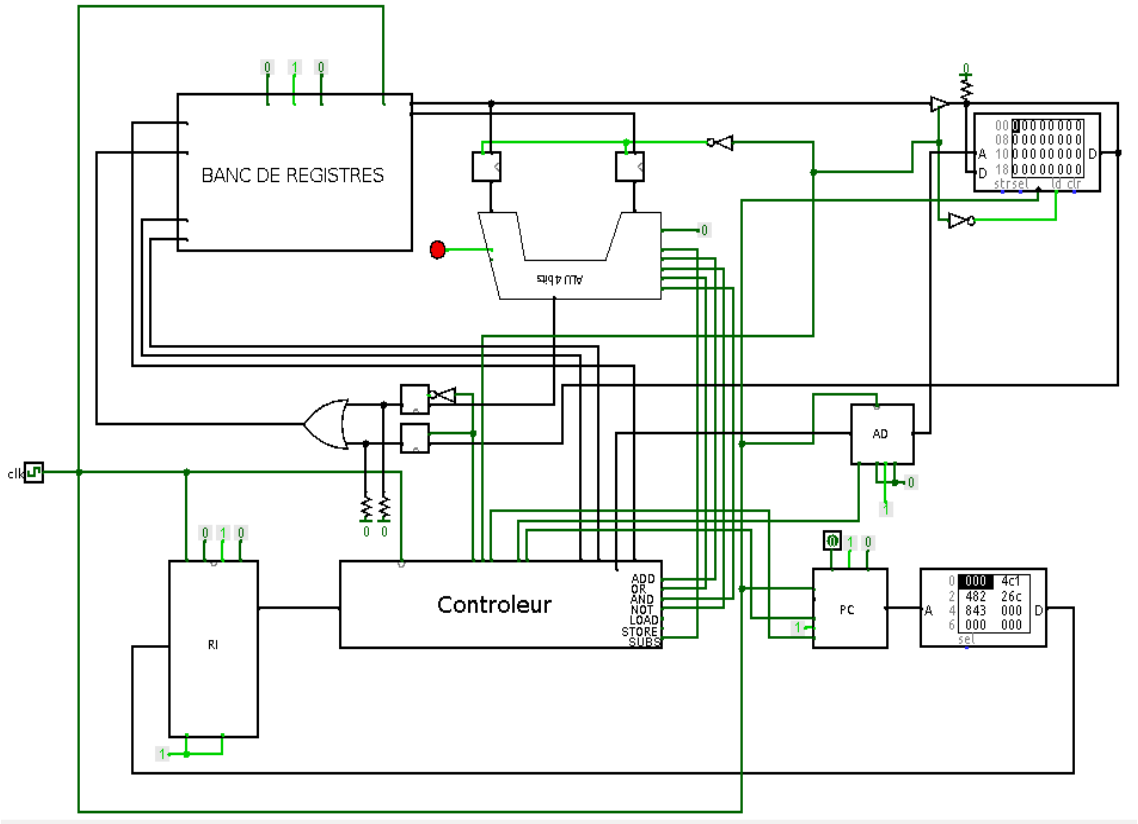


FIGURE 1 – Processeur 4 bit

Comme il est visible sur la figure 1, notre processeur est constitué de plusieurs modules communicants entre eux. Le signal d'horloge dont la source est visible à gauche, anime le processeur. Pour chaque série d'impulsion, le module de contrôle va exécuter une suite d'actions :

1. Fetch : Recherche d'une instruction à exécuter dans la ROM (en bas à droite) puis la mémorise dans le registe d'instructions (RI en bas à gauche).
2. Decode : Décodage de l'instruction contenue dans RI puis positionne les différents signaux (lecture, opérations de L'UAL).
3. Exécute : Récupère le résulat en sortie de l'UAL / la donnée désirée et on la place dans le Banc de Registres / la RAM à l'adresse sélectionnée.
4. Fetch (1) : On incrémente le compteur PC afin de charger l'instruction suivante.

En définitive, en fonction de la suite d'instructions définies dans la ROM (Read-Only Memory), on va exécuter des opérations dont le résultat sera soit stocké en mémoire RAM (Random Access Memory) soit dans le Banc de Registres.

2 Données d'exécution

Les données d'exécution nécessaires au déroulement d'un programme sont les instructions (stockées dans la ROM) ainsi que les données/opérandes nécessaires à l'exécution (stockées dans la RAM).

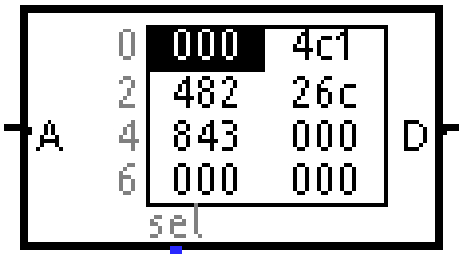


FIGURE 2 – Composant ROM

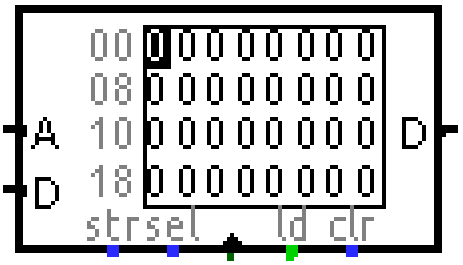


FIGURE 3 – Composant RAM

Dans notre processeur, les composants RAM et ROM possèdent les caractéristiques suivants :

ROM Contient les instructions constituant le programme. Les instructions ne peuvent être modifiées au cours de l'exécution.

- Taille : 16 emplacements
- Données : 12 bits (ici 3 valeurs hexadécimales)
- A : (entrée, 4 bits) Adresse de l'instruction
- D : (sortie, 12 bits) Instruction stockée à l'adresse A

RAM Contient les données d'exécution du programme. Ces données peuvent être modifiées / créées / effacées pendant l'exécution. Ce composant est synchronisé sur l'horloge.

- Taille : 64 emplacements
- Données : 4 bits (1 valeur hexadécimale)
- A : (entrée, 6 bits) Adresse de la donnée
- D gauche : (entrée, 4 bits) Donnée à stocker à l'adresse A
- D droit : (sortie, 4 bits) Donnée stockée à l'adresse A
- ld : droits en lecture dans la RAM

2.1 Instructions

Les instructions sont représentées sur 12 bits (ou 3 valeurs hexadécimales) ; les informations contenues dans une instruction sont à interpréter en plusieurs groupes de bits :

Opération logique Les opérations logiques commandent l'UAL ainsi que la lecture / écriture dans le Banc de Registres. Sur l'Illustration 4, on peut voir une opération logique divisée en plusieurs champs :

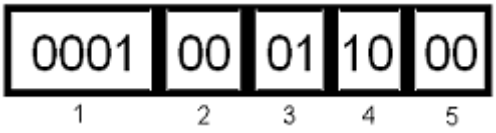


FIGURE 4 – Opération logique

1. Code de l'opération à réaliser (AND, OR, ADD, ..., acces mémoire) qui sera transmit à l'UAL.
2. Adresse dans le Banc de Registres où placer le résultat de l'opération
3. Adresse dans le Banc de Registres de l'opérande 1
4. Adresse dans le Banc de Registres de l'opérande 2
5. 2 bits non utilisés

Acces mémoire Les acces mémoires sont des opérations qui ne nécessitent pas l'intervention de l'UAL, elles sont nomées Load et Store. Elles consistent à faire entrer des données dans la RAM dans le Banc de Registres (Load) et inversement du Banc de Registres à la RAM (Store). Comme l'opération logique, une instruction d'acces mémoire se divise en plusieurs champs :

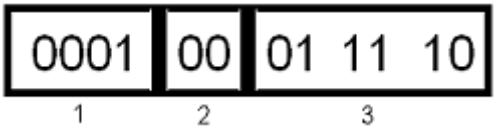


FIGURE 5 – Acces mémoire

1. Code de l'opération (LOAD, STORE, ..., opération logique) qui sert à identifier l'action
2. Adresse dans le Banc de Registres qui servira d'entrée / sortie pour l'information
3. Adresser dans la RAM qui servira de sortie / d'entrée pour l'information

Signaux de contrôle Une fois décodée, une instruction active un certain nombre de signaux de contrôle pour pouvoir contrôler l'UAL, écrire dans les différents registres / mémoires. Voici pour rappel la table associée à chaque signal de contrôle fournie dans le sujet :

Mnémonique	CodOp	F_2	F_1	F_0	Read	Write	wAD	Fetch	wPC
ADD	0000	0	0	0	0	0	0	1	1
OR	0001	0	0	1	0	0	0	1	1
AND	0010	0	1	0	0	0	0	1	1
NOT	0011	0	1	1	0	0	0	1	1
LOAD	0100	0	0	0	1	0	1	1	1
STORE	1000	0	0	0	0	1	1	1	1

FIGURE 6 – Table de codage de l'opération

Sur la Figure 6, on peut voir que chaque opération déclanche un certain nombre de signaux de contrôle :

— F2, F1, F0 (3 bits) : servent à sélectionner l'opération que l'UAL doit réaliser. Dans notre processeur, nous avons remplacé ce signal sur 3 bits par un ensemble de signaux séparés. Cela rend l'implémentation d'une nouvelle opération un peu plus laborieuse mais le débogage est plus aisé.
— Read (1 bit) : Sert à activer les droits en lecture sur le composant RAM (pin ld) dans le cas d'une opération LOAD (transfert de données de la RAM au Banc de Registres)
— Write (1 bit) : Sert à connecter la sortie du bus de données de la RAM à l'entrée du Banc de Registres dans le cas d'une opération STORE (transfert de données du Banc de Registres à la RAM)
— wAD (1 bit) : Sert à activer les droits en écriture dans le registre d'adresse dans le cas d'un acces mémoire
— wPC (1 bit) : Sert à activer les droits en écriture dans le compteur programme (PC) pour pouvoir procéder à la recherche d'instructions.
— Fetch (1 bit) : Permet d'incrémenter le compteur PC pour passer à l'instruction suivante.

2.2 Soustraction

Nous avons implémenté en plus de ces opération imposées la soustraction entre deux opérandes, voici sa table de contrôle :

Mnémonique	CodOP	F2	F1	F0	Read	Write	wAD	Fetch	wPC
SUBS	1001	-	-	-	0	0	0	1	1

Cette opération, de la même manière que l'addition va chercher ses deux opérandes A et B dans le Banc de Registres et place le résultat de l'opération à l'adresse voulue dans ce dernier.

3 Composants du processeur

Le processeur est composé de différents modules (Registre d'instructions, Contrôleur, ...) qui possèdent tous une fonction particulière. Avant de voir comment fonctionnent ces grands modules, intéressons-nous aux briques élémentaires qui les composent.

Portes logiques Les portes logiques (AND, OR, NOT, ...) sont à la base de tous les systèmes électroniques combinatoires, il ne nous a cependant pas semblé pertinant d'en parler ici.

Basule D La bascule D est une mémoire qui permet de stocker un bit de données. Elle est à la base de tous les registres du processeur. Une bascule D est synchronisée sur une horloge qui permet d'actualiser la valeur qu'elle contient à partir de l'entrée D. On peut ensuite lire la valeur contenue dans la bascule à partir de la sortie Q.

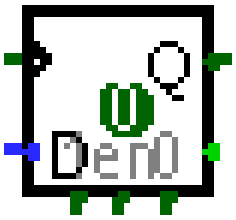


FIGURE 7 – Bascule D

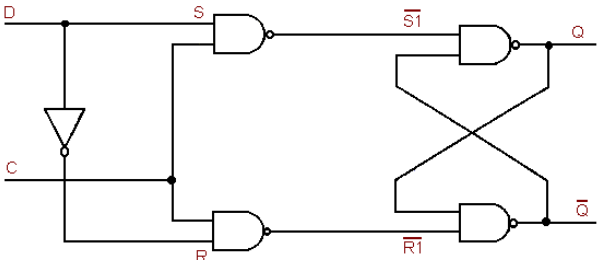


FIGURE 8 – Schéma électronique de la basicule D

3.1 Registre 4 bits

Le processeur est constitué d'une multitude de registres (Registre d'instructions, Compteur programme, ...) qui servent à stocker des données à court terme au plus près du coeur opérationnel pour pouvoir minimiser les acces mémoire et donc optimiser les performances. Tous ces registres ont poour composant de base la Bascule D Latch et fonctionnent sur un même principe, nous avons donc choisit de parler ici du Registre 4 bits, composant de base du Banc de Registres.

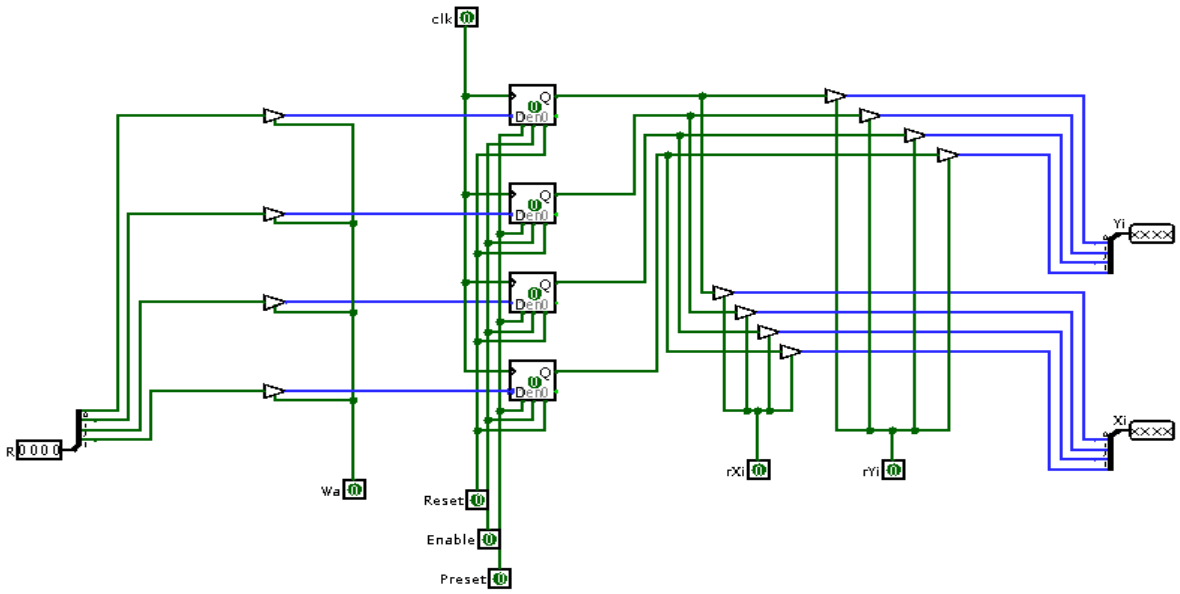


FIGURE 9 – Registre 4 bits

Comme il est visible sur la Figure 9, le coeur du Registre 4 bits est formé de 4 Bascules D, chacune d'elles mémorisant 1 bit d'information. Ces bascules sont entourées d'un certain nombre d'entrées / sorties :

- R (4 bits) : Les 4 bits d'information à stocker dans le registre
- Wa (1 bit) : Signal de droits d'écriture dans le registre, on ne peut écrire dans le registre que si ce signal est activé. Sinon, l'entrée de données est coupée grâce à des Buffers qui permettent de mettre l'entrée des bascules dans l'état de haute impédance et donc de déconnecter le circuit. Ce signal est utilisé pour réguler à quel moment, il est possible d'écrire dans le registre pour ne pas effacer malencontreusement les données.
- clk (1 bit) : Le signal d'horloge qui permet l'actualisation des données.
- Reset / Enable / Preset (1 bit chacun) : Des signaux de contrôle qui permettent de désactiver et gérer les données contenues dans les basucles.
- rXi / rYi (1 bit chacun) : Ces deux signaux permettent de sélectionner sur quelle sortie (Xi/Yi) on désire faire apparaître les données contenues dans le registre. Ils représentent les droits en lecture dans le registre.
- Xi/Yi (4 bits chacun) : Les 4 bits d'information en sortie du registre.

Nous l'avons vu, les registres possèdent un certain nombre de signaux qui permettent de gérer l'écriture ou la lecture. Il va donc de soit que l'écriture dans un registre s'effectue en plusieurs temps :

1. Présentation de la donnée en entrée R et sélection des droits en écriture dans le registre sur un front montant d'horloge
2. Mémorisation des données sur le front montant d'horloge suivant

3.2 Banc de Registres

Le Banc de Registre sert à stocker les données à utiliser pour effectuer les opération arithmétiques et logiques ainsi que leur résulat. On peut de plus y charger des données en provenance de la RAM grâce à la commande d'acces mémoire LOAD. De la même manière que tous les registres sont composés fondamentalement de Bascules D, le Banc de Registres est composé de 4 Registres 4 Bits. Il possède ensuite encore une fois un certain nombre de signaux de contrôle permettant de désigner dans quel registre on désire écrire, quel registre on désire consuler, ... De la même façon que dans le registre 4 bits, on retrouve donc les signaux de droits en écriture / lecture qui sont cependant cette fois sur 2 bits pour pouvoir sélectionner chacun des 4 registres. En plus de cela, on retrouve encore une fois l'entrée des données R (4 bits), l'horloge et les deux sorties X et Y qui permettent ici de consulter deux registres 4 bits en même temps afin de pouvoir réaliser les opération arithmétiques et logiques.

3.3 UAL 4 bits

L'UAL 4 bits (Unité Arithmétique et Logique) est le composant qui est chargé d'exécuter les opération arithmétiques et logiques à partir d'opérandes issues du Banc de Registre. Le résultat de ces opérations est ensuite aussi stocké dans le Banc de Registres. Elle est constituée de 4 UAL 1 bit qui effectuent les opérations pour chaque bits de la donnée en entrée pour retourner un résulat sur 4 bits. Les opérations que les UAL peuvent effectuer sont :

- A AND B
- A OR B
- NOT B
- ADD A to B
- SUBS B to A

Nous ne couvrirons pas ici les fonctions logiques de base de l'UAL car elles sont très simples ; nous pouvons en revanche aborder les opérations arithmétiques ADD et SUBS.

ADD L'addition de A à B (1 bit) est assurée par le composant Full Adder. On peut voir sur la Figure 11 que le Full Adder possède 3 entrées : les opérandes A et B ainsi que la Retenue entrante Re. On peut aussi voir qu'il possède deux sorties : le résultat de l'addition S et la retenue sortante Rs. La retenue entrante / sortante est utilisée pour pouvoir propager la retenue à travers un calcul de plus de 1 bit, ce qui est utilisé dans l'UAL 4 bits.

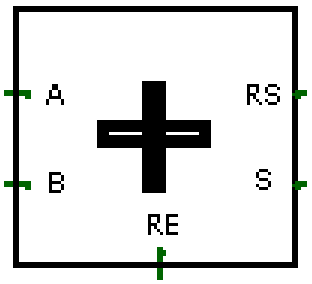


FIGURE 10 – Full Adder

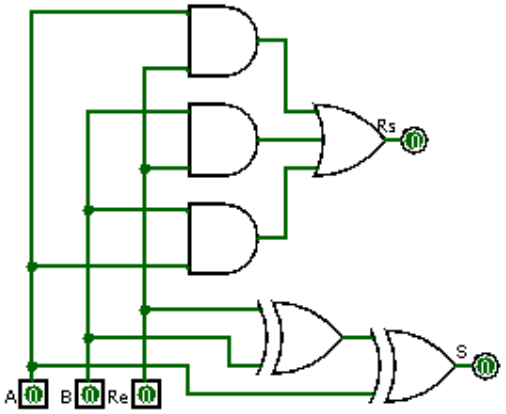


FIGURE 11 – Schéma électronique du Full Adder

A	B	Re	Rs	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

FIGURE 12 – Table de vérité du Full Adder

SUBS La soustraction est dérivée du Full Adder même si elle est un peu plus complexe. En effet pour faire le calcul A - B , il faut effectuer un complément à 2 sur B, c'est à dire l'inverser puis lui ajouter 1 pour obtenir -B et ainsi pouvoir additionner A et -B. On reprend pour cela le même circuit que le Full adder sauf que l'on inverve l'entrée de l'opérande B avec une porte NOT. Ensuite, pour calculer A-B sur 4 bits, il faut ajouter 1 au bit de points faible de B et ensuite propager la retenue. Donc l'entrée Re du calcul entre les bits de poinds faible de A et de B est toujours 1, puis on propage la retenue en effectuant l'addition A + non B + Re pour chaque bits. On obtient donc en sortie A - B.

3.4 Le compteur programme (PC)

Le compteur programme est un registre qui as pour fonction de "pointer" sur une instruction dans la mémoire ROM. En effet, PC contient une adresse sur 4 bits qui est utilisée pour charger dans le registre d'instruction à partir de la ROM une instruction à exécuter. Là où PC est différent d'un simple registre c'est que cette adresse peut être incrémentée à l'étape Fetch de l'exécution d'une instruction. C'est à dire que PC sert à suivre le déroulement d'un programme en sélectionnant tour à tour les instructions dans la ROM. Le registre PC possède donc un signal d'entré "incrémenter" ainsi qu'une

UAL positionnée sur ADD qui permet d'effectuer la dite incrémentation comme il est visible sur la Figure 13.

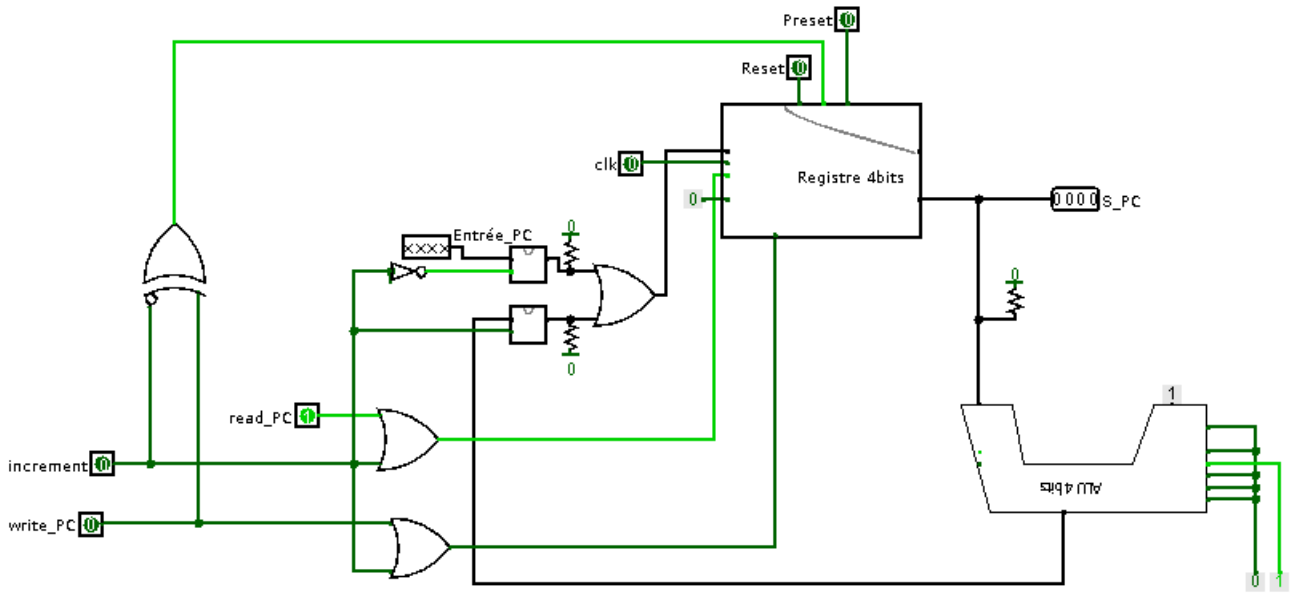


FIGURE 13 – Circuit du compteur programme

A noter qu'il est aussi théoriquement possible de forcer le compteur à une valeur donnée pour effectuer des sauts dans les instructions. Cette fonctionnalité n'as cependant pas encore été implémentée à la date de rédaction de ce rapport.

3.5 Unité de Contrôle

L'unité de contrôle est le coeur du processeur, elle permet d'assurer le déroulement de l'exécution d'un programme. A l'aide d'un automate appelé Séquenceur. Ce Séquenceur possède 3 états qui correspondent aux 3 étapes de l'exécution d'une instruction : Fetch, Decode, Execute. Pour réaliser ce Séquenceur, nous avons utilisé un registre à décalage comme illustré sur la Figure 14 :

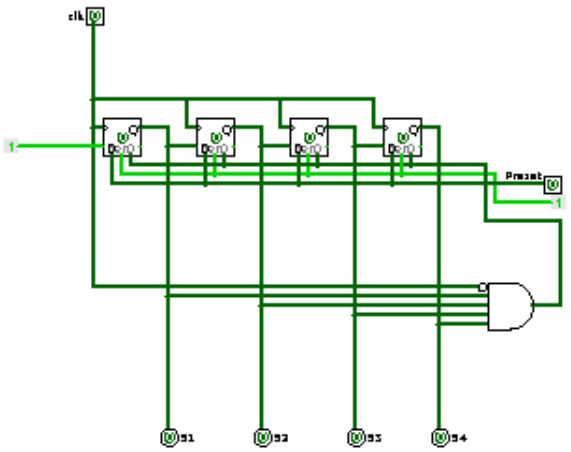


FIGURE 14 – Séquenceur

Ce registre passe à l'étape suivante à chaque front montant d'horloge permettant l'exécution d'une instruction en 4 cycles. (En réalité le 4ème cycle n'est pas exploité mais nous l'avons laissé de peur de

casser quelquechose au dernier moment). A chaque cycle, une sortie de plus s'active permettant ensuite l'activation successive du décodage, des signaux de commandes et de l'exécution de l'instruction. Une fois arrivé à la 4ème étape, le registre se réinitialise sur un front descendant d'horloge et le cycle recommence.

Décodeur Le décodage s'effectue automatiquement à partir de l'instruction chargée dans le registre d'instruction. Cette étape sert, conformément à la norme expliquée au Chapitre 2 à extraire d'une instruction les données essentielles à son exécution (codeOp, adresses). Le décodeur est en fait simplement un branchement logique qui va produire des signaux en fonction des données de l'instruction. Pour toutes les instructions, on va interprêter les 4 bits de poids fort comme le code d'opération, en fonction de ce code, il est possible que l'interprétation du reste des données de l'instruction vaire. Voici sa table de vérité pour chaque opération :

Instruction	codeOP	Ad Res	Ad A	Ad B	Reste	ADD	SUBS	AND	OR	NOT
0000rraabb00	0000	rr	aa	bb	00	1	0	0	0	0
1001rraabb00	1001	rr	aa	bb	00	0	1	0	0	0
0001rraabb00	0001	rr	aa	bb	00	0	0	1	0	0
0010rraabb00	0010	rr	aa	bb	00	0	0	0	1	0
0000rraabb00	0100	rr	aa	bb	00	0	0	0	0	1

FIGURE 15 – Table de vérité du décodeur d'instructions pour une opération arithmétique et logique

Instruction	codeOP	Ad BR	Ad RAM	LOAD	STORE
0100rrmmmmmm	0000	rr	mmmmmm	1	0
1000rrmmmmmm	1001	rr	mmmmmm	0	1

FIGURE 16 – Table de vérité du décodeur d'instructions pour un acces mémoire

A partir de ces données, dans le cas d'une opération arithmétique et logique, le Séquenceur va d'abord commander l'opération à exécuter à l'UAL puis déclancher la lecture des opérandes A et B dans le Banc de Données puis déclancher l'écriture du résultat. Pareillement, dans le cas d'un acces mémoire, le Séquenceur va transmettre l'adresse RAM à l'unité d'adressage puis positionner les signaux de lecture / écriture du Banc de Registre.

3.6 Registre d'instruction et Unité d'adressage

Ces deux composants sont abordés en dernier car ils sont les moins spécifiques. En effet, au niveau électronique ce sont juste des registres. Le Registre d'Instructions stocke une instruction (12 bits) en sortie de la ROM une fois qu'elle a été sélectionnée par le compteur programme. L'unité d'adresage quand à elle contient une adresse RAM (6 bits) dans le cas d'un acces mémoire. Elle sert "d'interface" pour réaliser la liaison entre le Banc de Registre et la RAM.

4 Gestion de projet

La gestion au cours de ce projet a été plutôt chaotique nous le reconnaissons. Dans notre binôme (CHENET Dorian, Alexis CHOLLET), nous avons eu une répartition du travail en tenant compte des autres projets du cursus (Développement Web et Génie Logiciel). Nous avons donc produit plutôt inégalement mais nous avons tout deux comprit les enjeux du projet ainsi que les partie importantes du cours qui étaient à appliquer.

Sur la fin du projet, il y a un certain nombre de petites choses qui ont été réalisées (pour corriger des problèmes de synchronisation notemment) que nous n'avons pas eu le temps de clarifier ou améliorer dans les circuits. Nous sommes néanmoins heureux du résultat.