Signaux et systèmes

ENSIBS

Cyber 1 - TP2/TD1

JALLET Nicolas

26/05/2020

Introduction

Le but de ce projet est de construire un petit processeur 8 bits à base d'éléments logiques et séquentiels dans le but d'appréhender les mécanismes de base du fonctionnement interne d'un processeur de type RISC. Ce petit processeur simplifié ne sera pas pipeliné (une instruction sera entièrement traitée en un cycle). De plus, l'accès à la mémoire sera réalisé en un seul cycle, simplifiant le contrôle du processeur (cas de la mémoire cache parfaire : 0 miss). Afin de réaliser ce processeur 8 bits, nous allons utiliser le logiciel de simulation Logisims. Ce logiciel est open-source sous licence GPL.

Conception

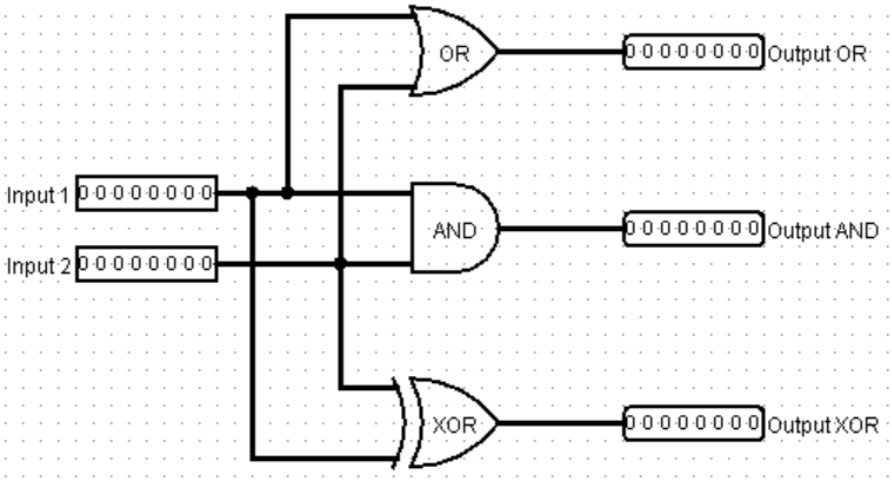
Etape 1: Conception des unités logiques (UL)

La première étape consiste à réaliser une unité logique (UL). Il faut construire 3 circuits logiques permettant de réaliser les opérations **AND**, **OR** et **XOR** sur des données de 8 bits.
2 contraintes doivent être respectées:

- Utiliser seulement des portes logiques à 2 entrées de 1 bit
- Intégrer ces 3 circuits dans un module nommé "UL"

Ce module possède 2 entrées de 8 bits chacune et 3 sorties de 8 bits également.

L'opération "OR" est réalisée par la première porte L'opération "AND" est réalisée par la seconde porte L'opération "XOR" est réalisée par la dernière porte



Etape 2: Conception d'un additionneur

Ensuite, il faut concevoir un additionneur. Sa conception se découpe en plusieurs étapes. La première est de créer un additionneur 1 bit complet. Cette additionneur servira de base pour construire l'additionneur final.

Pour réaliser ce circuit, la table de vérité doit être établie afin d'en extraire les équations de sorties **Rout** et **Somme**.

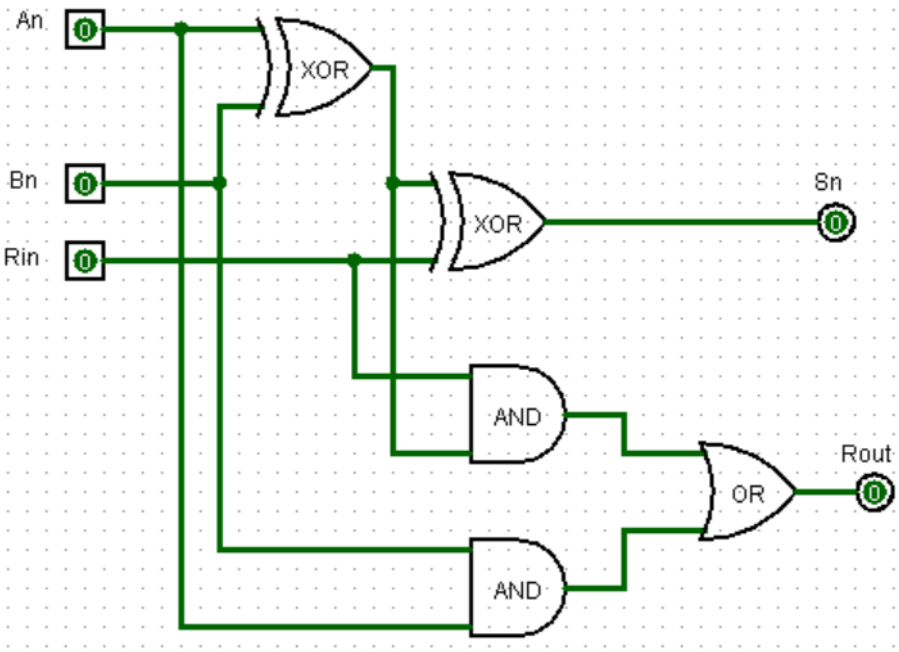
Rin	a	b	Sn	Rout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

A partir de cette table de vérité, deux équations se définissent:

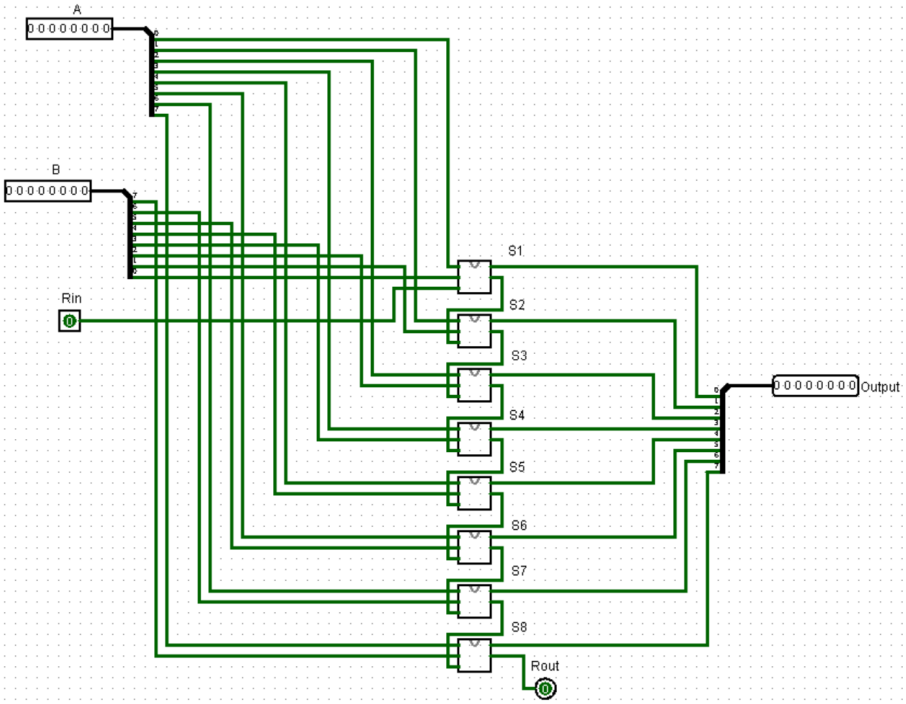
- Sn = An (+) (Bn (+) Rin)
- Rout = An.Bn + An.Rin + Bn.Rin

Ces équations ont été trouvé sur la source suivante:

- url



Il est maintenant possible de réaliser l'additionneur 8 bits. Il faut s'appuyer sur la mise en cascade de plusieurs additionneurs 1 bit complet.



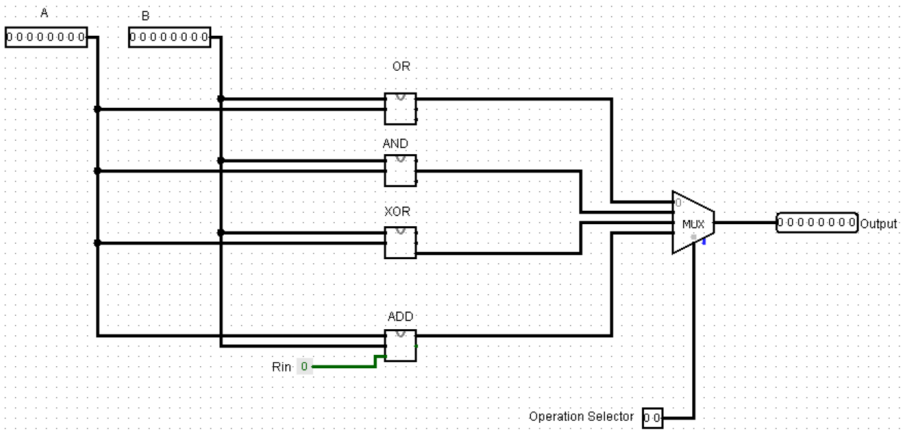
Etape 3: Conception de l'unité arithmétique logique (UAL)

L'UAL du processeur sera le cœur opératif du processeur. Ici, cette unité est capable (au minimum) de réaliser 4 opérations:

- 3 opérations logiques (AND,OR,XOR)
- 1 opération arithmétique (ADDITION)

Toutes les opérations sont réalisées sur des données de 8 bits. Le choix de l'opération est définie par un signal dédié appelé "selector". Ce signal est sur 2 bits permettant de choisir 1 des quatre opérations.

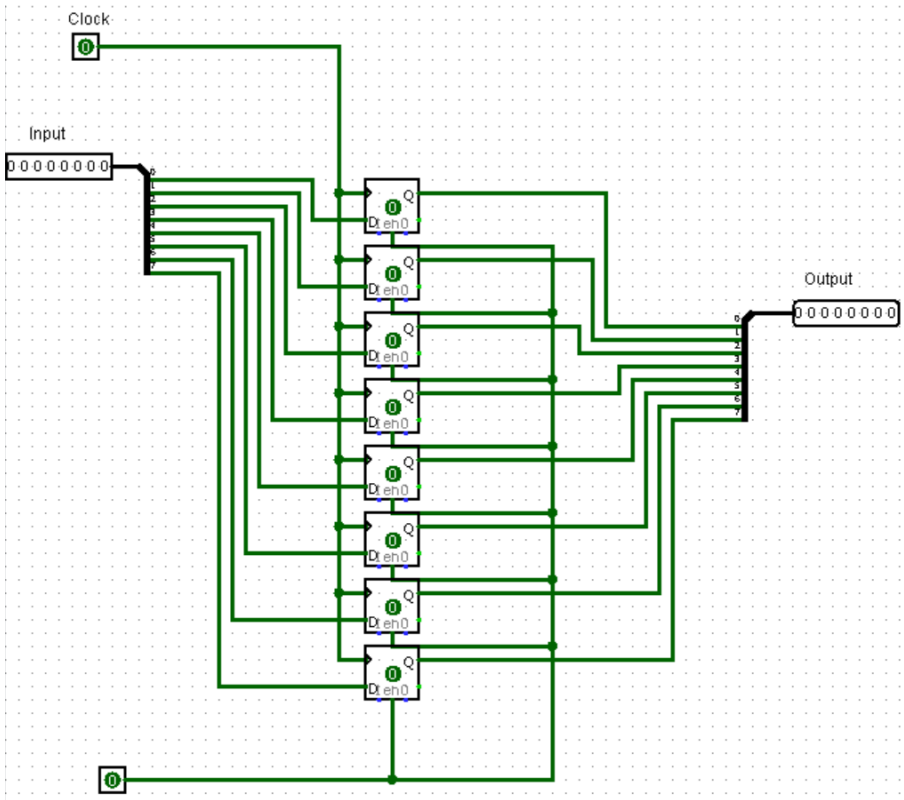
Pour construire ce module, il faut intégrer les modules logiques et arithmétiques précédemment réalisés.



Etape 4: Conception d'un banc de registres

Un banc de registres est composé de plusieurs registres permettant de stocker les valeurs manipulées au sein du processeur. Il est important de noter qu'au sein d'un processeur RISC, les opérations arithmétiques et logique sont réalisées sur ces registres uniquement. Des instructions "LOAD" et "STORE" permettent de charger / sauvegarder des valeurs dans ou en provenance des ces registres.

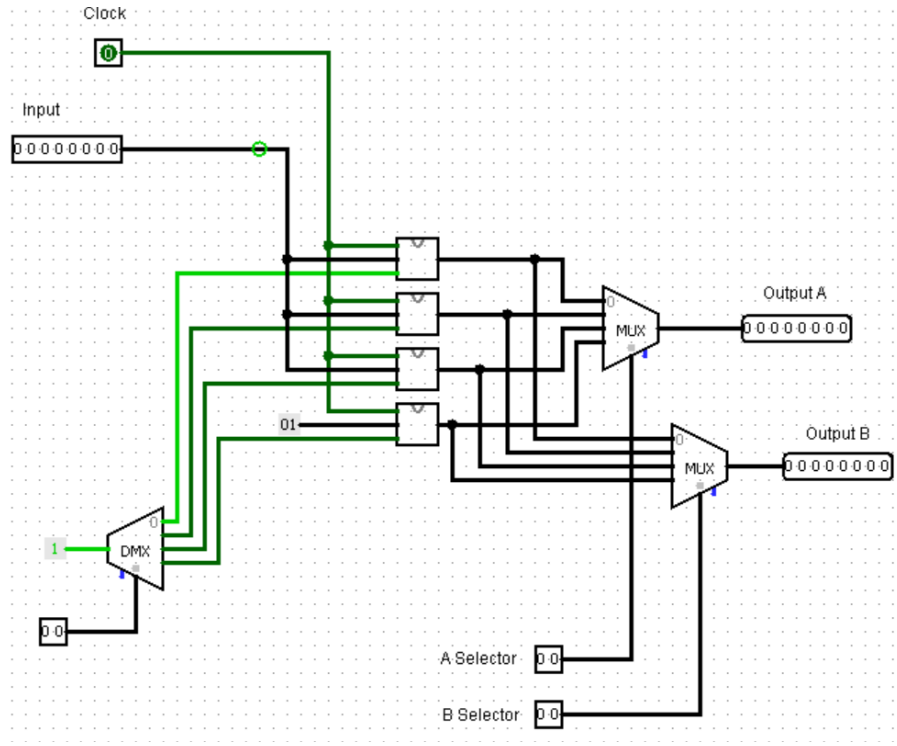
Un registre est construit à base de bascules D. Chaque bascule permet de stocker 1 bit.



Dans le cas de notre processeur il faut créer 4 registres permettant de stocker des valeurs de 8 bits. Il est possible de créer un module registre 8 bits que l'on peut ensuite dupliquer pour créer le banc de registres complet.

Les signaux de sélection doivent permettre de réaliser les tâches suivantes :

- Sélectionner un des 4 registres dont la valeur stockée sera transmis sur la sortie A. **(A Selector)**
- Sélectionner un des 4 registres dont la valeur stockée sera transmis sur la sortie B. **(B Selector)**
- Sélectionner le registre dans lequel sera stocké la valeur en entrée E. **(Input Selector)**

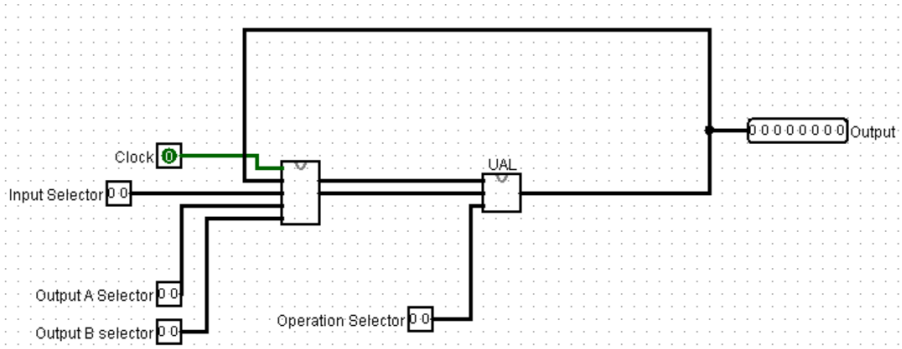


Etape 5: Conception d'une unité de traitement complète (CPU)

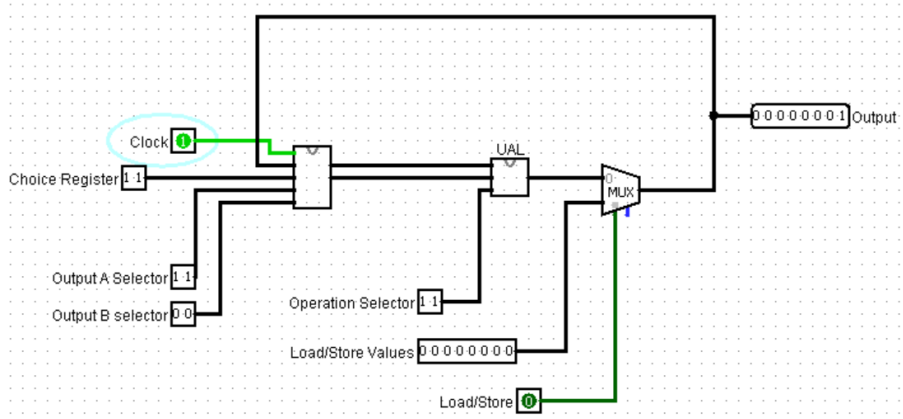
Cette étape a pour but de tester le fonctionnement du couple banc de registre - UAL. Pour ce test, il faut relier les deux modules en boucle fermé afin que les données en sortie du banc de registres alimentent l'UAL et que le résultat en sortie de l'UAL alimentent un des registres.

Dans cette phase initiale, le banc de registre ne contient que des valeurs nuls. Il est possible de:

- initialiser des valeurs au sein des registres (changement d'état des bascules)
- multiplexer l'entrée du banc de registre afin d'injecter une constante dans les registres.



Le test en fonctionnement (Addition de 1):



On sélectionne le registre 4 qui est initialisé avec une constante à "1", on sélectionne A que l'on additionne à B qui est égale à 0. Le choix de l'opération est réalisé par le code binaire "11" sélection le 4ème composant de notre UAL.

Etape 6: Intégration d'une mémoire programme

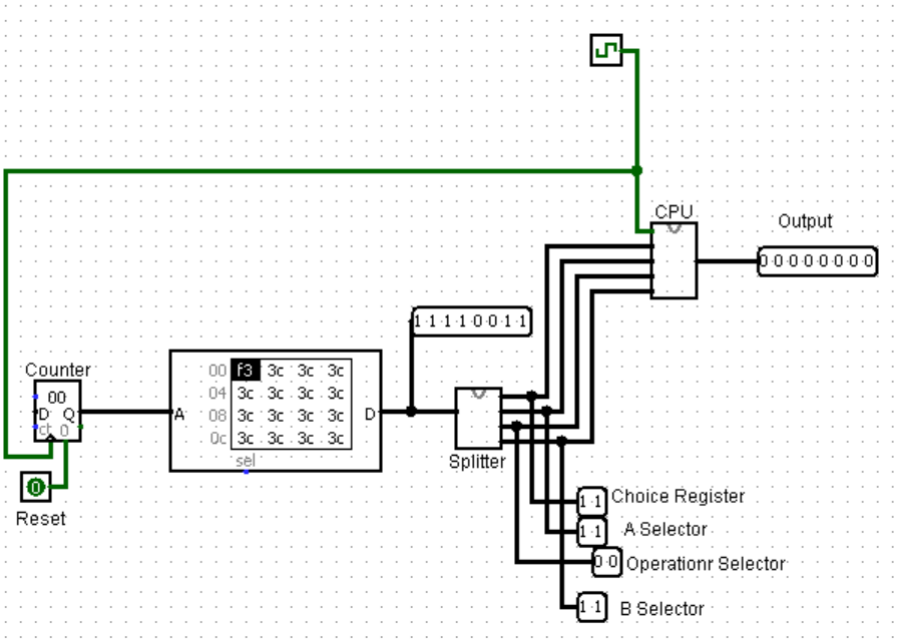
Cette étape permet d'implémenter un processeur possédant une mémoire ROM contenant des instructions qui seront décodées dans le but de piloter le processeur.

Il faut utiliser une mémoire de type ROM disponible dans Logisim. Il est possible de configurer une mémoire ROM selon ses besoins (largeur d'un mot, profondeur de la mémoire). Un éditeur hexadécimal permet de générer le contenue de la mémoire (cf documentation Logisim) La mémoire programme contiendra des instructions permettant de piloter votre processeur.

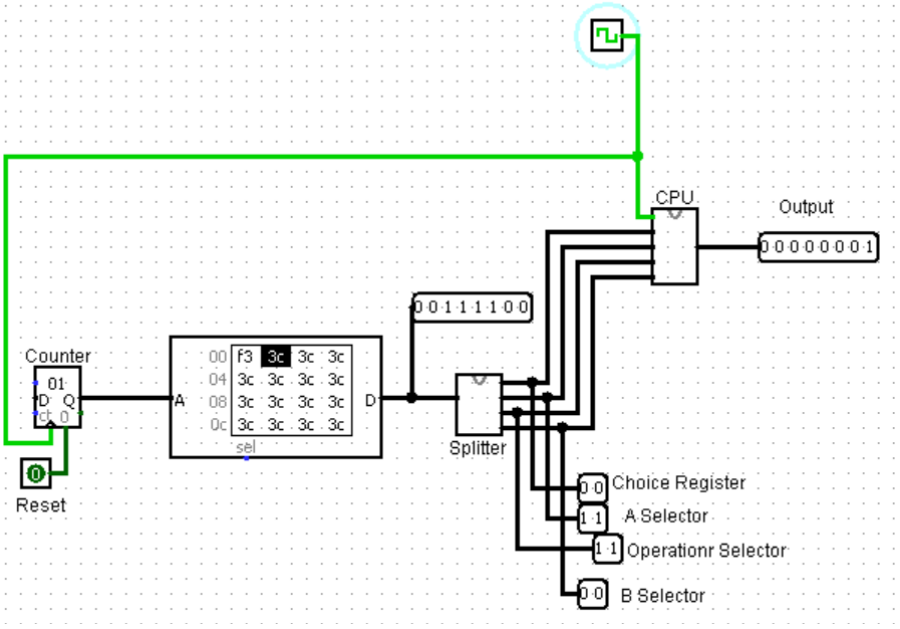
Pour tester le fonctionnement du CPU, il est possible de réaliser une opération d'addition en additionnant le registre A avec le registre B. Tout ceci doit être codé en hexadécimal dans notre ROM:

- 0xF3 => binaire: 11 11 00 11 (Initalisation)
  Les deux premiers bits **"11"** vont permettrent de selectionner le registre de stockage numéro 4. Ce registre a été initialisé à 1 pour permettre une addition (**cf Banc de registre**).
  Les deux **"11"** suivants permettent de sélectionner le contenu entier du registre A pour réaliser l'opération.
  "00" sélectionne l'opération du registre 0 correspondant à un OR. Les deux derniers **"11"** permettent de sélectionner le contenu entier du registre B pour réaliser l'opération. Finalement l'opération logique réalisée est "1 ou 0", 1 est donc mis sur la sortie en signe d'initialisation.
- 3c => binaire : 00 11 11 00 (Addition) Pour réaliser une addition, on sélectionne le registre 4 correspondant à l'addition. De plus on prend le contenu de A = 1 et le contenu de B qui est égale à 0 (00 en binaire). Ainsi on obtient une addition de 1 à chaque coup d'horloge.

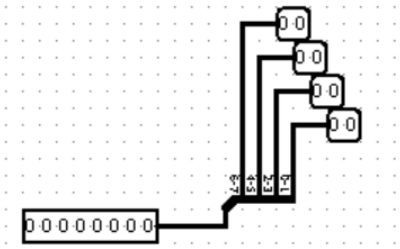
Avant la phase d'initialisation:



Après la phase d'initialisation, on peut commencer à compter

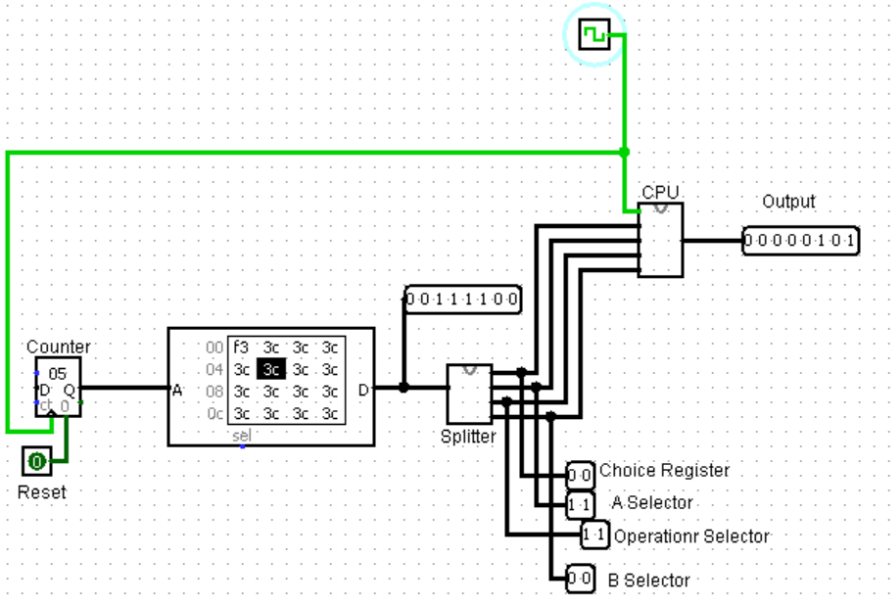


Il est possible de remplacer le bloc "splitter" par un bloc Logisim également appelé "Splitter", la réalisation de ce bloc est à but éducatif.



En fonctionnement:

La sortie est incrémentée de 1 à chaque coup d'horloge. On peut lire sur le compteur le nombre 5 et en sortie la représentation binaire de ce nombre.



Etape 7: Intégration d'une mémoire de données

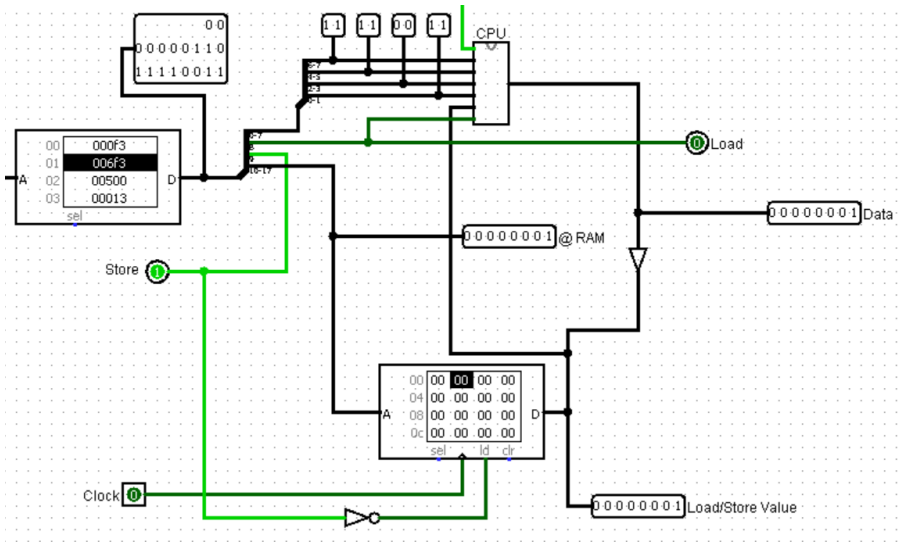
Le processeur contient peu d'espace de stockage pour le moment. Le nombre de registres est limité dans un processeur. Pour palier ce problème, il faut ajouter une mémoire RAM permettant de lire et écrire des données durant l'exécution du programme.

Pour pouvoir LOAD/STORE une donnée, il faut préalablement modifier le composant CPU en y rajoutant un multiplexeur qui va devoir choisir entre le résultat de l'opération entre les registres ou bien d'une valeur fixée. Ce choix est effectuée par une entrée codé sur 1 bit permettant le choix entre "0" et "1".

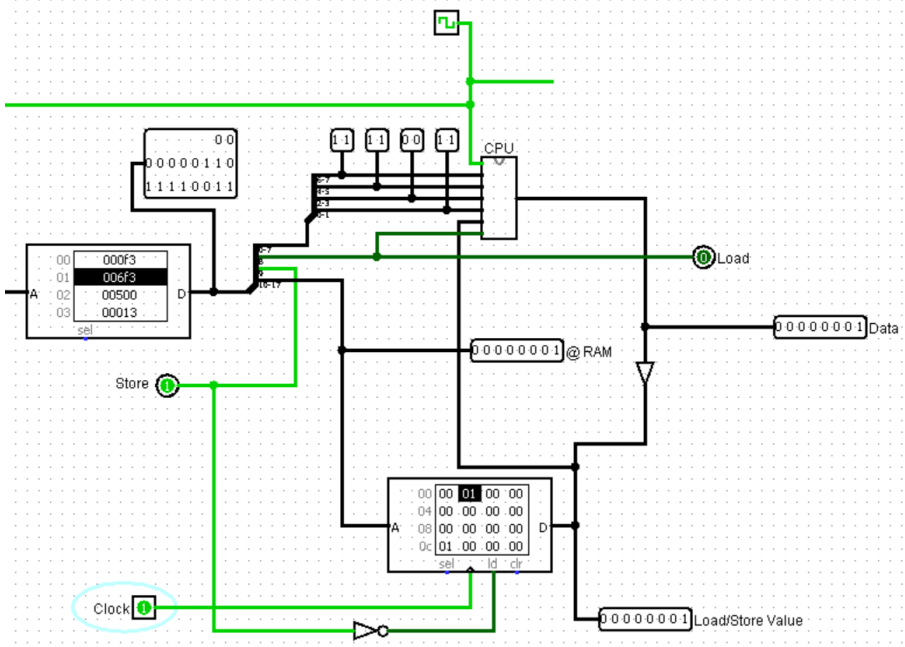
- "0" **LOAD**, on charge la valeur du registre sur la sortie
- "1" **STORE**, on stock la valeur d'entrée que l'on souhaite

Il faut que notre processeur puisse réaliser de nouveaux types d'instructions:

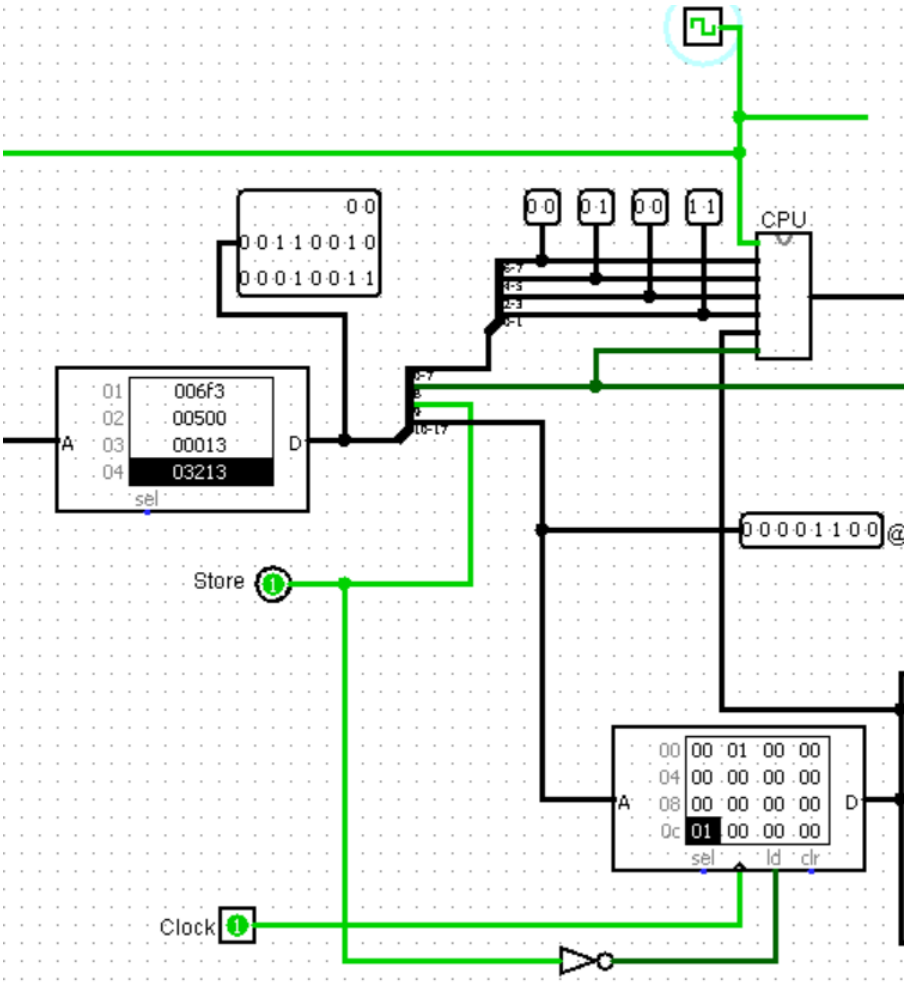
- Charger une constante dans un registre.



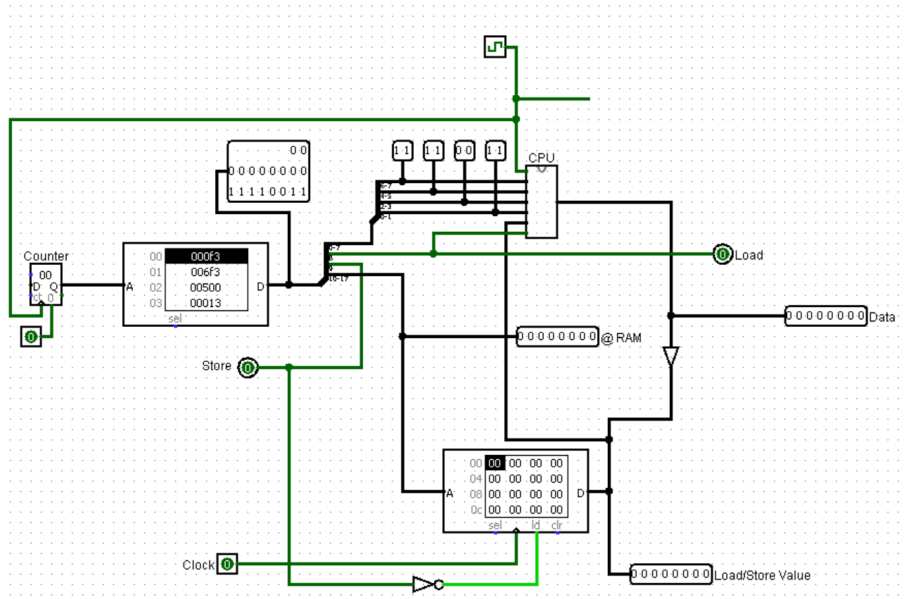
- Charger dans le registre R3 la valeur stockée à l'adresse '0X02' de la mémoire RAM.



- Stocker à une adresse de la mémoire RAM, la valeur contenue dans le registre 12.



Montage complet:



Conclusion

La réalisation d'un processeur complet de 8 bits permet de mieux appréhender le fonctionnement d'un ordinateur et de son architecture. C'est une base qu'il faut connaître pour pouvoir programmer en langage de bas niveau type assembleur.