TP N°2_Architecture

Conception d'une unité de contrôle
Durée conseillée ***2h***

Nous allons poursuivre la conception sous Logisim de notre CPU8bits.
Cette fois-ci nous allons réaliser l'unité de contrôle (ou de commande) qui n'est autre qu'un séquenceur (automate) dont la fonction première est le décodage des instructions du programme. Il faudra être rigoureux lors de la saisie de schéma.

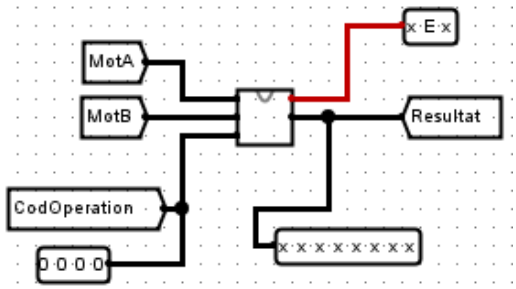
A) Plantons le décor :

- *Créer sous Logisim dans le projet CPU8.cir un nouveau schéma CPU8b.*

→ Une page d'édition vierge s'ouvre.
→ Il se rajoute à l'arborescence du Projet.
Vous disposez donc maintenant de 3 "sous-schémas" : **`main ALU8b`** *et* **`CPU8b`**.

Étape N°1*: Placement et câblage de l'ALU.*

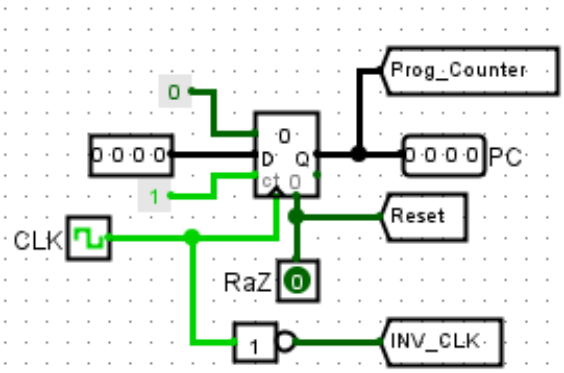
- *Porter le focus sur CPU8b*
- *Cliquer sur* **`main`** *pour importer l'***`ALU8b`** *et réaliser le câblage ci-dessous :*



- *Sauvegarder le projet.*

Étape N°2*: Placement et câblage du compteur ordinal (PC).*
Dans la librairie **`Memory`** *rechercher et placer le module* **`Compteur`**.

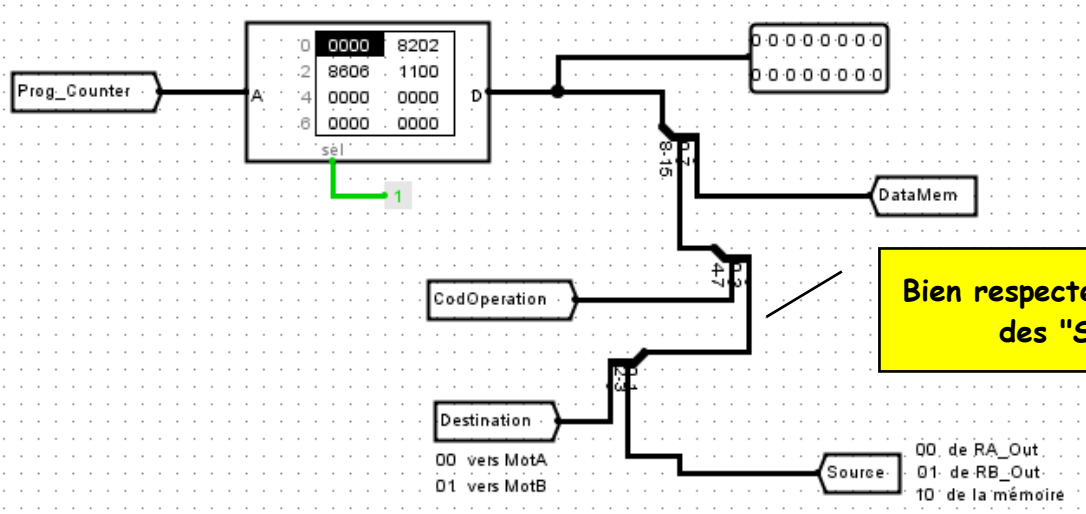
- *Câbler-le comme indiqué ci-dessous :*



- *Sauvegarder le projet.*

***Étape N°3**: Placement et câblage de la mémoire de programme.*
Dans la librairie **Memory** *rechercher et placer le module* **ROM**.

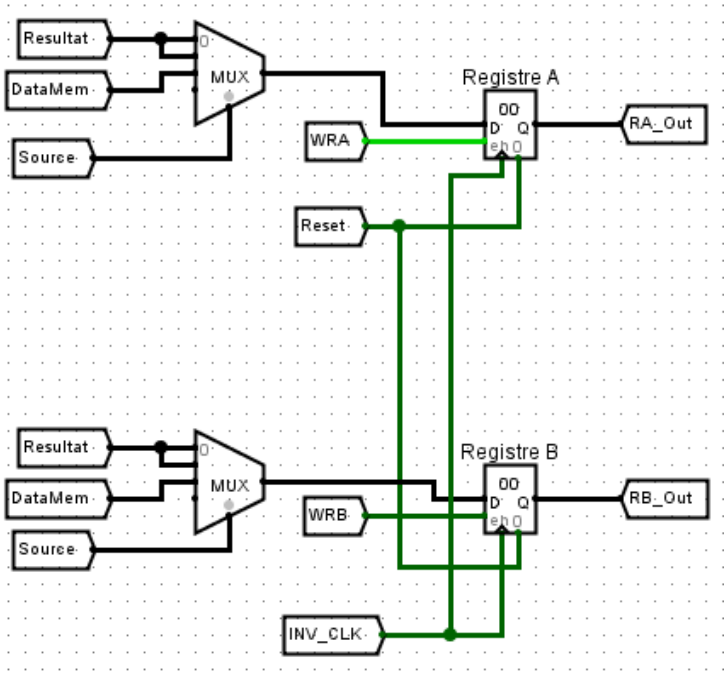
- *Câbler-le comme indiqué ci-dessous :*



- *Sauvegarder le projet.*

***Étape N°4**: Placement et câblage des registres accumulateurs* **RA** *et* **RB**
Dans les librairies **Memory** *et* **Plexers** *rechercher et placer les modules* **Registre** *et* **Multiplexeur**.

- *Câbler-les comme indiqué ci-dessous :*

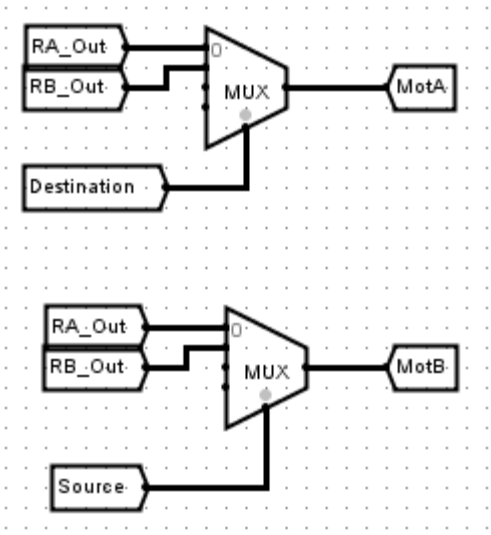


- *Sauvegarder le projet.*

***Étape N°5**: Placement et câblage de l'interface liant **RA** et **RB** avec l'ALU.*

Dans la librairie **Plexers** *rechercher et placer les module* **Multiplexeur**.

- *Câbler-le comme indiqué ci-dessous :*

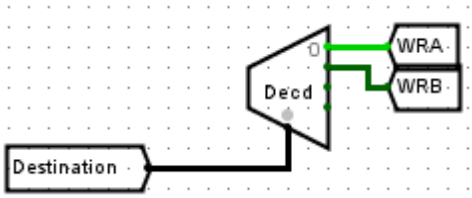


- *Sauvegarder le projet.*

***Étape N°6**: Placement et câblage du décodeur de destination des données.*

Dans la librairie **Plexers** *rechercher et placer les module* **Decodeur**.

- *Câbler-le comme indiqué ci-dessous :*

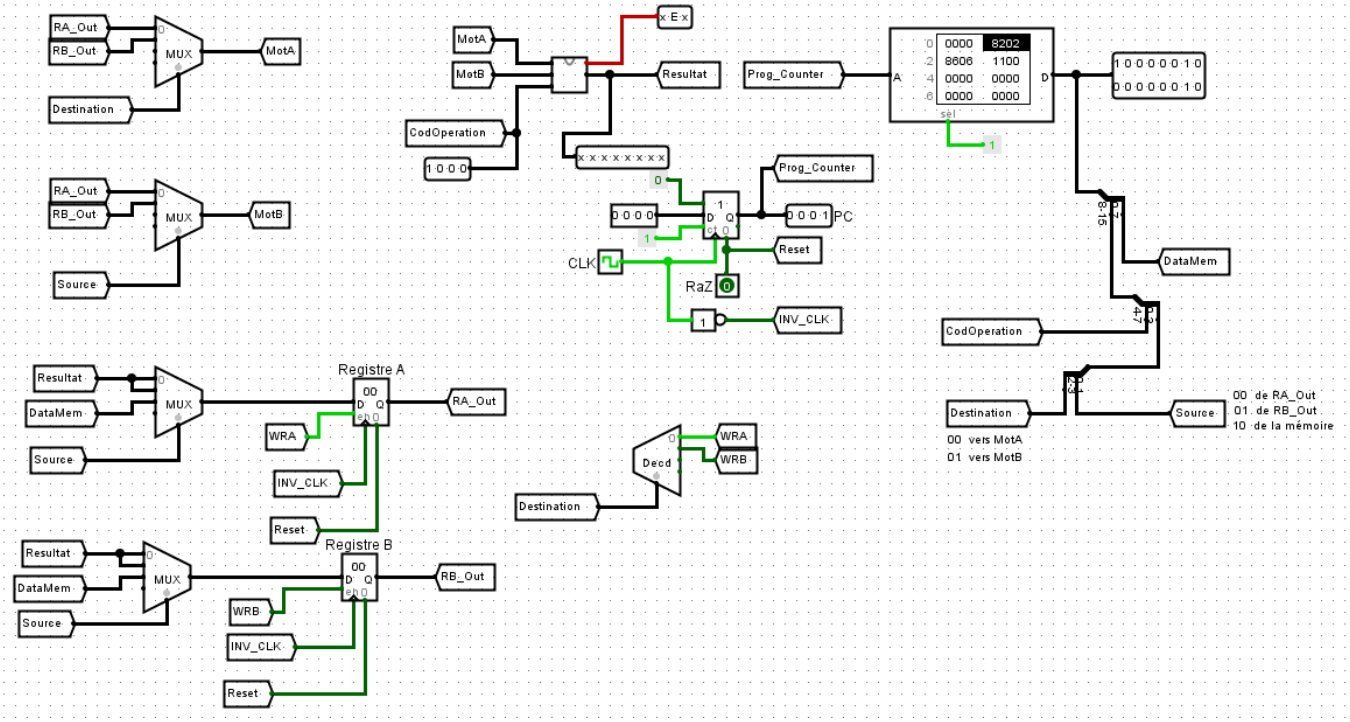


- *Sauvegarder le projet.*

En page suivante le schéma complet du CPU est donné.
Vérifier que vous n'avez rien oublié…
Le moindre "grain de sable" dans le déroulement de la mécanique du séquencement des opérations sera fatale….
Nous allons à présent voir comment écrire un code et l'exécuter.

Schéma complet du CPU8b.

Vous pouvez placer des "pins" pour suivre l'évolution du code



Avant de poursuivre il est nécessaire de rappeler quelques points importants pour expliquer ce qu'il se passe en sortie de la ***ROM.***

- *Description d'une instruction (16 bits) pour ce CPU: Fetch+ Decode*

CodOperation(4bits)	Destination(2bits)	Source(2bits)	DataMem(8bits)

Ceci justifie le "split" du bus de données de la ***ROM.***

- *Les instructions (on en connaît déjà* ***TP N°1_Architecture****)*

CodOperation	Destination	Source	DataMem
0001: Addition	00: RA	00:RA	
0010: Soustraction	01:RB	01:RB	
0011: ET		10:ROM	
0100: OU			
0101: OUEX			
0110: NOT			
1XXX: Charger un Registre			

***Étape N°7**: Tester votre CPU.*

- *Sur un front montant de l'horloge* **CLK** *se produiront simultanément dans notre contexte Fetch-decode-execute.*
- *Sur un front descendant via* **INV_CLK** *se produira la sauvegarde dans les registres.*

***Écriture du code dans la* ROM.**

Logisim propose un éditeur hexadécimal pour remplir la **ROM** *avec un programme. Supposons par exemple que nous souhaitons faire calculer à notre processeur la somme* **2+6** *(via l'ALU) et sauver le résultat* **8** *dans le registre* **RA** *de l'ALU. Pour ce faire nous pouvons réaliser les étapes suivantes :*

1. *Charger le registre* **RA** *avec* ***2***
2. *Charger le registre* **RB** *avec* ***6***
3. *Ajouter* **(RB)** *à* **(RA)** *et transférer dans* **RA**

Ce qui peut donner pour faire simple dans un "langage machine":

```
1: MOV RA,2 //placer 2 dans MotA
2: MOV RB, //placer 6 dans MotB
3: ADD RA,RB //Resultat 8 dans RA
```

***Codage pour la* ROM**

En utilisant le tableau de codage on obtient :

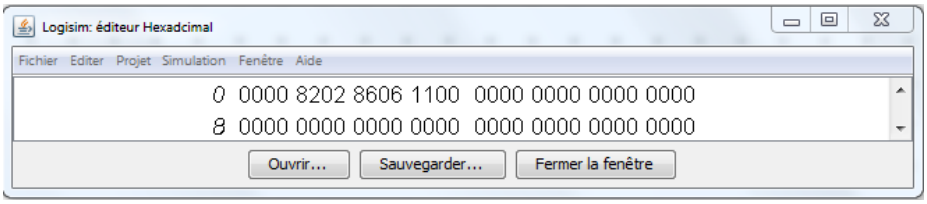
```
1: 1000 0010 0000 0010 → 0x8202
2: 1000 0110 0000 0110 → 0x8606
3: 1000 1000 0000 0000 → 0x1100
```

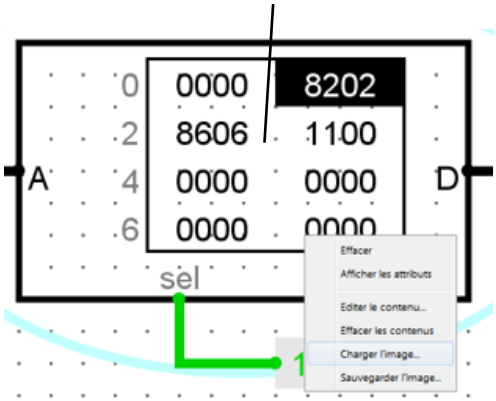
***Edition dans la* ROM.**

- *Sélectionner la* **ROM**
- *Dans attribut* **Contenu** *cliquer sur* **(Click pour édite**

Vous pouvez aussi éditer directement dans la ROM.

La fenêtre suivante apparaît :





Saisir les valeurs en hexadécimal de votre programme.

- *Sauvegarder sous le nom* **TestROM**
- *Fermer la fenêtre.*
- *En mode édition amener la souris sur la ROM et ouvrir le menu contextuel (click droit) puis choisir* **Charger l'image.**
- **Ouvrir le fichier TestROM**
- *Le contenu est de la* **ROM** *est modifié.*

Pour finir si le cœur vous en dit.

Tester un programme qui reprend celui que l'on vient de voir mais qui inverse (not) **(RB)** *avant de l'additionner à* **RA** *et de le placer dans* **RA**.