TP 7
Construisons le LC-3 - partie 2

On continue à compléter le fichier `LC3_etu.circ` déjà modifié lors du TP précédent.

7.1 Décodage des instructions

À l'issue de ce TP, la version simplifiée du processeur que nous allons construire permettra d'exécuter les instructions arithmétiques (`ADD`, `AND`, `NOT`), deux instructions d'accès direct à la mémoire (`LD`, `ST`) et une instruction de branchement (`BR`).

EXERCICE 1 ▶ **Circuit** `DecodeIR`
Les signaux de sortie du sous-circuit `DecodeIR` (`Load`, `Store`, `Arith`, `GetOp`, `WriteEnable`, `Branch` et `Offset9`) permettent d'activer correctement le chemin de données en fonction de la classe d'instruction à exécuter. Déterminez en fonction de `IR[13,12]` (bits 13 et 12 du registre d'instruction) comment doivent être activés ces signaux. Complétez le sous-circuit `DecodeIR` *N'oubliez pas `Offset9`!*

7.2 Instructions d'accès mémoire

Dans cette section on va ajouter les composants du circuit pour les instructions `LD` et `ST`.

EXERCICE 2 ▶ **Programme de test**
Écrivez en langage machine LC-3 un programme *de taille minimale* pour tester les instructions `LD` et `ST` : ce programme chargera deux entiers depuis les adresses $(5)_{10}$ et $(6)_{10}$ dans deux registres, puis rangera leur somme à l'adresse $(6)_{10}$. Vous utiliserez judicieusement PennSim pour traduire votre programme en langage machine chargeable dans la RAM du LC-3 de LOGISIM [1]

EXERCICE 3 ▶ **Implantation de** `LD`
Dans cet exercice, on se concentre sur l'implantation de l'instruction `LD`.

1. Quelle action doit être effectuée dans le chemin de données du processeur lors de la phase d'exécution d'une instruction `LD DR, label` ?
2. Lors de l'exécution d'une instruction `LD`, quel sous-circuit est chargé de placer la mémoire RAM en mode lecture ?
3. L'unité `GetAddr` se charge de calculer l'adresse de la mémoire qui doit être accédée `Addr` dans la RAM. Au cours de la phase de `Fetch`, que doit valoir `Addr` ? Même question au cours de la phase `Exec` d'une instruction `LD`.
4. Complétez à l'aide des signaux `Offset9, PC, Load, Store, Fetch` la formule valable pour l'exécution de toutes les instructions :

   ```
   Si ........................................
   Alors Addr = PC
   Sinon Addr = ..............................
   ```

5. Complétez `GetAddr` de façon à ce que le circuit permette l'exécution de `LD`.
6. Au niveau du banc de registres, comment doit-être effectuée l'écriture dans `DR` de la donnée ? *On vérifiera qu'il n'y a rien à mettre à jour à l'intérieur du composant `Registres`, mais par contre des modifications sont à faire dans le circuit principal pour bien amener la bonne donnée en entrée.*

1. Au passage, pourquoi peut-on utiliser ici le codage fourni par PennSim, qui stocke le programme à partir de l'adresse `x3000`, alors que notre circuit stocke le programme à partir de l'adresse `x0000` ?

7. Expérimentez votre circuit en exécutant les trois premières instructions de votre programme de test.

EXERCICE 4 ► **Implantation de** `ST`
On se concentre à présent sur l'implantation de l'instruction `ST`.

1. Quelle action doit-être effectuée dans le chemin de données du processeur lors de la phase d'exécution d'une instruction `ST SR,label`?
2. Comment la RAM est-elle placée en mode écriture lors de l'exécution d'un `ST`?
3. Comment est calculée l'adresse de destination dans la RAM?
4. Mettez à jour votre circuit au voisinage du banc de registres, si nécessaire.
5. Expérimentez votre circuit en exécutant l'instruction `ST` de votre programme de test.

7.3 Instructions de branchement et saut

Dans cette section on cherche à rajouter les composants du circuit pour l'instruction `BR`.

EXERCICE 5 ► **Programme de test**
Écrivez en langage machine LC-3 un programme qui permettra de tester `BR`.

EXERCICE 6 ► **Conditions d'activation - NZP**
On veut implanter le sous-circuit `NZP`. On rappelle que l'architecture du LC-3 spécifie trois drapeaux `N`, `Z` et `P`, qui indiquent respectivement si la dernière valeur chargée dans le banc de registres était strictement négative, nulle, ou strictement positive. Le circuit `NZP` contient un registre 3 bits (*falling edge triggered*) qui stocke l'état des drapeaux.

1. Ecrivez, en fonction des 16 bits formant l'entrée `IN` du circuit (valeur à tester), les fonctions logiques donnant les valeurs que doivent prendre `N`, `Z` et `P`.
2. Quand doit-être mis à jour le registre 3 bits contenant l'état des drapeaux `N`, `Z` et `P`?
3. La sortie `Cond` du sous-circuit prend la valeur 1 si, d'après les drapeaux `N`, `Z` et `P`, le `BR` en cours d'exécution doit provoquer un saut : donnez une formule pour `Cond`.
4. Complétez le sous-circuit `NZP`.
5. Modifiez le chemin de donnée du LC-3, de façon à ce que les drapeaux `N`, `Z` et `P` soient mis à jour : soit d'après `MemOUT` dans le cas de l'exécution d'une instruction de chargement mémoire (signal `Load` à 1), soit d'après la sortie `Res` de l'ALU dans tous les autres cas. *On fera attention au choix signal commandant l'écriture dans le registre (entrée `Clock` du composant `NZP`), qui ne peut être l'horloge du circuit, pourquoi?)*

EXERCICE 7 ► **Calcul de l'adresse de saut**
Il ne nous reste plus qu'à mettre à jour le calcul de la prochaine valeur du registre `PC`. Cela se passe dans le sous-circuit `RegPC`.

1. Quelle action doit effectuer une instruction `BR[n][z][p], label` durant la phase `Exec`?
2. Complétez le sous-circuit `RegPC`, puis testez à l'aide de votre programme.
3. Que faudrait-il faire pour prendre en compte l'instruction `JSR`?