CHENG Daniel 21954618
ROMELE Gaspard 21950004

Pour ce travail, nous devions compléter un circuit avec Logisim afin qu'il reflète le fonctionnement de l'architecture LC-3, en ajoutant une fonction SCAN. écrire plusieurs programmes de manipulations de chaînes de caractères en assembleur et une conversion de codage binaire vers Gray et inversement.
La fonction Scan a été réalisée à partir du code assembleur Scan.asm disponible sur la page git du cours dans tools/lc3.
A la fin du projet, nous avions codés toutes les fonctions de manipulations de chaînes de caractères en des routines qui modifient uniquement les registres du résultat en implémentant une pile. Le circuit était complet, avec la fonction Scan ajoutée et les fonctions de conversion de codage binaire en Gray ou de Gray en codage binaire étaient implémentées comme des programmes indépendants.

Voici les principales différences trouvables dans le fichier LC-3-v2.circ du rendu par rapport au circuit donné au début du projet :

- Au départ, toute instruction était reconnue comme étant de type arithmétique et pouvant modifier les registres. Après avoir compléter le sous-circuit « DecodeIR », le programme reconnaît maintenant quel type d'instruction est donné en argument et si l'instruction peut modifier les registres.
- Nous avons ajouté la fonction Scan en utilisant une structure en escalier.
- Nous avons complété le sous-circuit « ALU » afin que la fonction Scan soit reconnue et son résultat calculé.
- Le sous-circuit « NZP » a été modifié afin que le registre NZP soit modifé correctement quand une instruction le demande.
- « WriteVal » modifie les registres si nécessaire, par exemple avec une instruction de chargement ou JSR.
- Quand une instruction de branchement est lue, le sous-circuit « RegPC » redirige le programme afin que les prochaines instructions lues soient celles suivant le label donné dans l'instruction.
- Le NZP n'a plus le test nzp.
- La condition récupère maintenant le NZP au lieu du test nzp
- Le RegPC récupère le NZP au lieu du test nzp pour effectuer le BR

Ci-dessous se trouve le code des différents programmes de test utilisés :

Arithinstr.asm :

```
.ORIG x3000
AND R0,R0,0
ADD R0,R0,1
NOT R0,R0
ADD R0,R0,1
SCAN R0,R0,-1
.END
```

Ce code utilise au moins une fois chaque fonction de type arithmétique et le tester dans le circuit nous a renvoyé les résultats attendus.

JUMPinstr.asm :

```
.ORIG x3000
LD R0,cmp
```

```
loop :   ADD R0,R0,-1
         BRp loop
         JSR funct
         TRAP x25
funct : RET
cmp : .FILL 2
         .END
```

Ce code teste les fonctions BR, JSR et TRAP. Nous n'avons pas pu rajouter les tests pour RTI et JMP.

LOADinstr.asm :

```
         .ORIG x3000
         LEA R0,string
         LD R1,string
         LDR R2,R0,0
         LDI R1,0
string : .STRINGZ "hello_world"
          .END
```

Nous testons ici une fois chaque instruction de chargement.

STOREinstr.asm :

```
         .ORIG x3000
         ST R1, string
         STR R0,R2,0
         STI R1,
         string : .FILL 10
         .END
```

Avec ce programme, nous vérifions le fonctionnement des instructions de rangements.