Mini-projet de travaux pratiques
Réalisation du processeur Irving

Le projet est à effectuer en binôme (ou, exceptionnellement, en monôme). Il sera évalué lors de la dernière séance de TP. Un rapport complet est à fournir sous forme imprimée à l'enseignant encadrant la séance de TP le jour de l'évaluation. Le fichier *Logisim* correspondant devra être déposé sur madoc au plus tard la veille de l'évaluation sous la forme d'une archive[1] dont le nom sera composé des noms de chacun des étudiants du binôme — ex. `tartempion-dupont.tar.gz`. La décompression de cette archive devra créer un répertoire `tartempion-dupont/` contenant le fichier Logisim et le rapport sous forme électronique. On re-précisera en commentaire dans le fichier Logisim sur le circuit « `main` » les noms et prénoms de chacun des étudiants.

On souhaite réaliser le circuit électronique du processeur *Irving* permettant de programmer les déplacements d'un robot tortue sur une grille. Une tortue pilotée par un *Irving* peut se déplacer horizontalement et verticalement sur une grille. Son orientation absolue peut être modifiée suivant les quatre directions (haut, droite, bas, gauche). Initialement, la tortue se trouve dans la case $(0, 0)$ de la grille et sa tête pointe vers le haut. La tortue est équipée d'un dispositif de marquage qui peut être abaissé ou non ; lorsqu'il est abaissé, la trace du chemin suivi apparaît sur la grille. Le processeur *Irving* est capable de déplacer une tortue sur une grille d'au maximum 32 lignes et 32 colonnes. Pour les besoins de ce projet, on utilisera un pilote d'affichage (fourni) restreint à une grille de 5 lignes et 5 colonnes.
Le code ci-dessous est un exemple de programme simple exécutable par l'*Irving* après traduction en code machine et stockage dans sa mémoire à partir de l'adresse `0x00` :

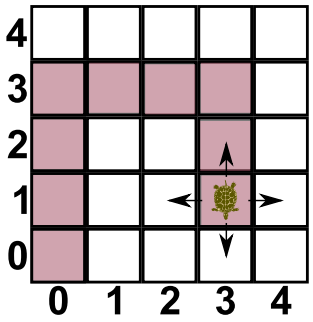

FIGURE 1 – Tortue et sa grille

```
trace 1   ; Abaisser le marqueur
move 3    ; Avancer de 3 cases vers le haut (orientation par défaut)
turn 1    ; Tourner à droite
move 3    ; Avancer de 3 cases vers la droite
turn 2    ; Tourner vers le bas
move 2    ; Avancer de 2 cases vers le bas
halt      ; Arrêter l'exécution
```

Son exécution est illustrée dans la figure 1.

Présentation de l'*Irving*

Le processeur *Irving* possède huit registres généraux ($R_0, \ldots, R_7$) de 6 bits ; un registre `PC` (*Program Counter*) de 8 bits pointant en mémoire sur la prochaine instruction à exécuter ; deux registres `XR` et `YR` de 5 bits conservant, respectivement, la colonne et la ligne où se trouve la tortue ; un registre `OR` de 2 bits conservant l'orientation actuelle de la tortue (haut, droite, bas, gauche) ; et un registre `TR` conservant la position du dispositif de marquage (1 pour « *abaissé* » et 0 pour « *relevé* »).
Le jeu d'instructions de l'*Irving* est décrit dans la table 1. Chaque instruction est codée sur 16 bits suivant l'un des deux formats présentés dans la figure 2.
L'*Irving* possède une mémoire de 256 mots de 16 bits. L'adressage se fait par mot et non par octet.
L'architecture de l'*Irving* est décrite dans la figure 3.

1. Manipulation d'archives :
   — Création de l'archive `titi.tar.gz` à partir du répertoire `titi` : `tar -vzcf titi.tar.gz titi/`
   — Récupération du contenu de l'archive `titi.tar.gz` : `tar -vzxf titi.tar.gz`

TABLE 1 – Instructions de l'*Irving*

Instruction	Opcode	Format	Action
`move` c	0000	F_a	Avance la tortue de c cases ($c \in [-32, 31]$)
`move` R_i	0001	F_a	Avance la tortue de R_i cases
`turn` d	0010	F_b	Tourne la tortue dans la direction d ($d \in [0, 3]$)
`turn` R_i	0011	F_a	Tourne la tortue dans la direction R_i
`load` R_i, n	0100	F_a	$R_i \leftarrow n$, avec $n \in [-32, 31]$
`add` R_i, R_j	0101	F_a	$R_i \leftarrow R_i + R_j$
`trace` b	0110	F_a	Définit le statut du marqueur (0 : levé, 1 : baissé)
`trace` R_i	0111	F_a	Définit le statut du marqueur
`beq` R_i, R_j, a	1000	F_a	Ajoute l'offset a à `PC` si $R_i = R_j$ ($a \in [-32, 31]$)
`bne` R_i, R_j, a	1001	F_a	Ajoute l'offset a à `PC` si $R_i \neq R_j$ ($a \in [-32, 31]$)
`bge` R_i, R_j, a	1010	F_a	Ajoute l'offset a à `PC` si $R_i \geqslant R_j$ ($a \in [-32, 31]$)
`bgt` R_i, R_j, a	1011	F_a	Ajoute l'offset a à `PC` si $R_i > R_j$ ($a \in [-32, 31]$)
`halt`	1111	F_a	Arrête définitivement l'exécution du programme

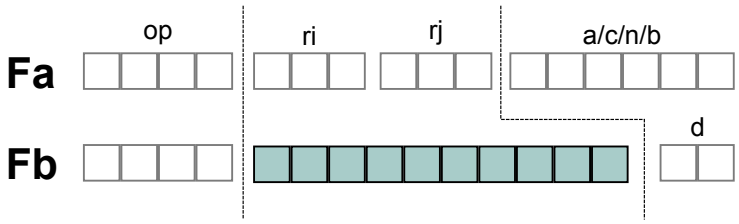

FIGURE 2 – Formats des instructions du *Irving*

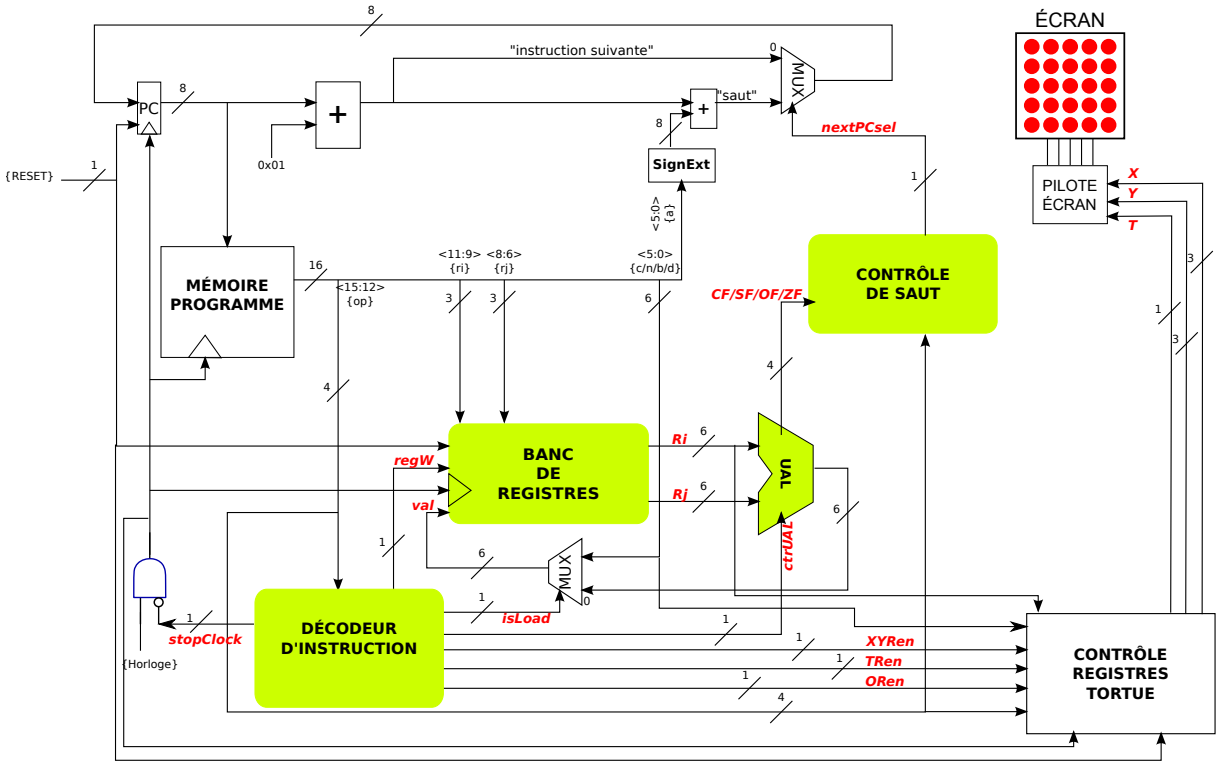

FIGURE 3 – Architecture de l'*Irving*

Réalisation des circuits de l'*Irving*

On se propose de réaliser le processeur *Irving* dans *Logisim*. On tirera au mieux parti des possibilités de ce logiciel en matière de développement modulaire de circuits et l'on prendra soin de commenter les circuits réalisés. Chaque fois que possible, on pourra directement utiliser les modules logiques déjà présents dans *Logisim* (ex. : registres, multiplexeurs, additionneurs, ...). On partira du circuit se trouvant dans turtle5x5.circ disponible sur madoc

1. Implémenter l'UAL prenant en entrée deux valeurs sur 6 bits et réalisant leur addition ou leur soustraction en fonction du bit `ctrUAL`. La soustraction se fera en utilisant judicieusement l'opération de complément à deux et l'addition. L'UAL devra positionner les quatre indicateurs `CF` (*Carry Flag*), `SF` (*Sign Flag*), `OF` (*Overflow Flag*) et `ZF` (*Zero Flag*) ;
2. Réaliser le module « *banc de registres* ». Le banc contient 8 registres de 6 bits. Il prend en entrée les indices r_i et r_j et retourne les valeurs des registres $R[r_i]$ et $R[r_j]$. Si le signal `regW` vaut 1, il stocke dans $R[r_i]$ la valeur apparaissant sur son entrée `val` ;
3. Le module « *contrôle de saut* » détermine la valeur du signal `nextPCsel` en fonction de l'opcode de l'instruction courante et de la valeur des indicateurs `CF`, `SF`, `OF` et `ZF`. Donner la table de vérité permettant de modéliser le comportement de ce module et l'implémenter ;
4. Le module « *décodeur d'instruction* » prend en entrée l'opcode de l'instruction courante et positionne les différents indicateurs en fonction du chemin de données de cette instruction. Donner la table de vérité de ce module et l'implémenter.

Note : dans la réalisation des circuits de l'*Irving*, on s'attachera à respecter scrupuleusement les spécifications de format d'instruction de façon à ce que le correcteur puisse tester le processeur avec son propre programme traduit en code objet.

Utilisation de l'*Irving*

On va désormais utiliser le processeur construit pour exécuter un programme de dessin.

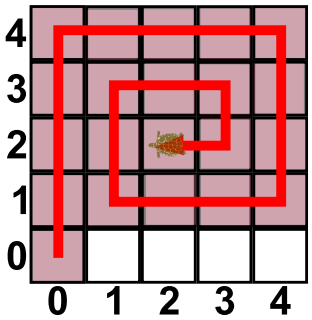

FIGURE 4 – Colimaçon réalisé avec l'*Irving*

1. En utilisant le langage d'assemblage de l'*Irving*, écrire le programme réalisant le dessin apparaissant dans la figure 4. Pour cela, on utilisera *impérativement* une structure de boucle ;
2. Traduire en code machine le programme précédent et le stocker dans la mémoire de l'*Irving*. Après avoir vérifié sa bonne exécution, sauver le contenu de la mémoire dans un fichier que l'on fournira avec le fichier *Logisim*.