Licence Informatique 3 — Université de Nantes
X31I050 — 2020/2021

Projet de travaux pratiques
Réalisation du processeurs SC-1

Le projet est à effectuer en binôme (ou exceptionnellement, en monôme). Le fichier *Logisim* commenté ainsi que le rapport de projet *au format PDF* sont à déposer *impérativement* sur Madoc. On re-précisera en commentaire dans le fichier Logisim les noms et prénoms de chacun des étudiants.

Un projet rendu après la **date limite fixée au lundi 27 novembre 2020, 23h55**, sera affecté d'un malus proportionnel au nombre de jours de retard. Le projet donnera lieu à une présentation *par les deux membres du binôme* lors de la dernière séance de travaux pratiques.

Un processeur à architecture SC-1 possède 16 registres de 8 bits $R_0, \ldots, R_{15}$ pouvant contenir des entiers signés en complément à 2 et un registre `PC` (*Program Counter*) de 8 bits. Un SC-1 offre aussi une mémoire de 256 mots de 16 bits pour stocker le code à exécuter.
Le jeu d'instructions supportées par le SC-1 est présenté dans la table 1. Les trois formats d'instructions F_1, F_2 et F_3 sont décrits dans la figure 1. Toutes les instructions du SC-1 s'exécutent en un seul cycle.
Le but du projet est de réaliser avec Logisim une implémentation du SC-1 tel que décrit dans la figure 2, puis de l'utiliser pour exécuter un programme.
On utilisera au mieux les possibilités de création de sous-circuits de Logisim ; on tirera aussi parti des circuits fournis en standard par Logisim (multiplexeurs, …). Le rapport de projet devra justifier tous les circuits créés (tables de vérités, …).

1 Implémentation du SC-1

Table 1 – Jeu d'instructions du SC-1

Opcode	Instruction	Format	Commentaire
0000	`add` r_d`,` r_s`,` r_t	F_1	Addition
0001	`sub` r_d`,` r_s`,` r_t	F_1	Soustraction
0010	`or` r_d`,` r_s`,` r_t	F_1	OU bit-à-bit
0011	`and` r_d`,` r_s`,` r_t	F_1	ET bit-à-bit
0100	`not` r_d`,` r_s	F_1	NON bit-à-bit
0101	`shl` r_d`,` r_s`,` r_t	F_1	Décalage à gauche logique
0110	`shr` r_d`,` r_s`,` r_t	F_1	Décalage à droite logique
0111	`li` r_d`, val`	F_2	Chargement d'un immédiat
1000	`halt`	F_1	Arrêt du programme
1001	`b offset`	F_3	Saut inconditionnel `offset` octets en avant ou en arrière
1010	`beq` r_s`,` r_t`, offset`	F_3	Saut conditionnel `offset` octets en avant ou en arrière
1011	`bne` r_s`,` r_t`, offset`	F_3	Saut conditionnel `offset` octets en avant ou en arrière
1100	`bge` r_s`,` r_t`, offset`	F_3	Saut conditionnel `offset` octets en avant ou en arrière
1101	`ble` r_s`,` r_t`, offset`	F_3	Saut conditionnel `offset` octets en avant ou en arrière
1110	`bgt` r_s`,` r_t`, offset`	F_3	Saut conditionnel `offset` octets en avant ou en arrière
1111	`blt` r_s`,` r_t`, offset`	F_3	Saut conditionnel `offset` octets en avant ou en arrière

1. Réaliser l'Unité Arithmétique et Logique (UAL) offrant les opérations suivantes sur 8 bits : addition, soustraction, OU bit-à-bit, ET bit-à-bit, NON bit-à-bit, décalage à gauche, décalage à droite logique. L'UAL devra retourner un résultat sur 8 bits ainsi que 4 indicateurs : `CF`, `ZF`, `SF`, `OF`. On retournera « 0 » pour les indicateurs ne faisant pas sens pour certaines opérations (exemple : `OF` pour le `OU` bit-à-bit).
   On pourra partir de l'additionneur 4 bits mis au point lors de la première séance de travaux pratiques ou utiliser l'additionneur fourni par Logisim ;

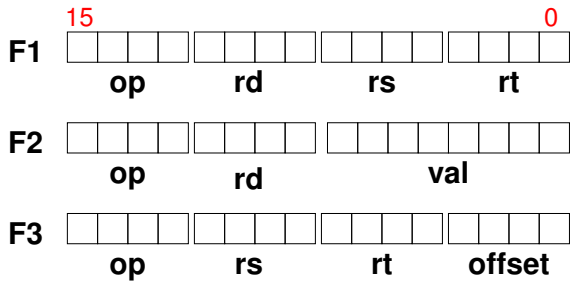

FIGURE 1 – Formats d'instructions

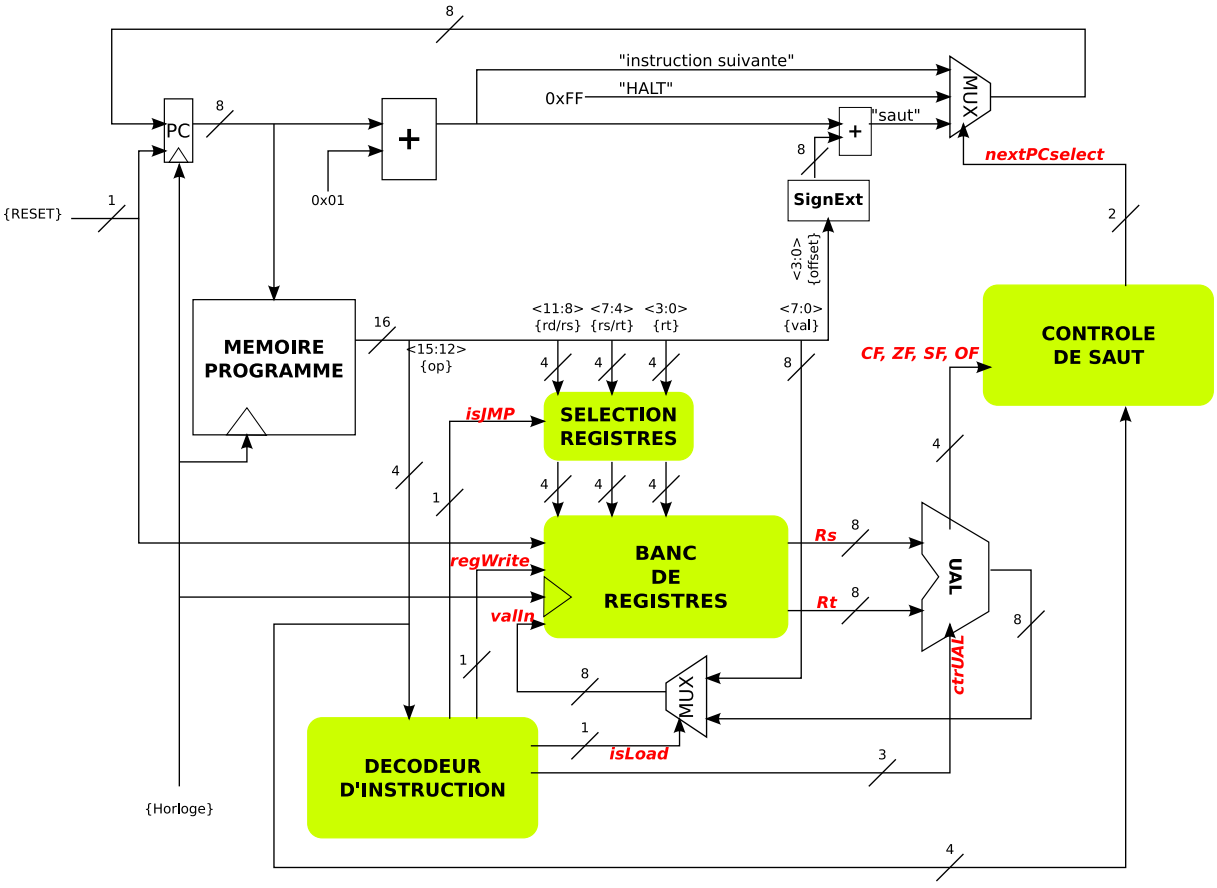

FIGURE 2 – Architecture du SC-1

2. Le module « *décodeur d'instruction* » prend en entrée l'opcode de l'instruction courante et retourne quatre indicateurs :

   `isJMP`. Indicateur valant `1` si l'instruction courante est une instruction de saut ;

   `regWrite`. Indicateur valant `1` si l'instruction courante accède à un registre en écriture ;

   `isLoad`. Indicateur valant `1` si l'instruction courante est une instruction de chargement ;

   `ctrUAL`. Indicateur codant le type d'opération demandée à l'UAL.

   Calculer la table de vérité du *décodeur d'instruction*. En déduire son implémentation dans Logisim ;

3. Le module « *contrôle de saut* » prend en entrées l'opcode de l'instruction courante ainsi que les indicateurs `CF`, `ZF`, `SF` et `OF` mis à jour par l'UAL lors de la dernière opération effectuée. Il retourne en sortie un indicateur sur deux bits déterminant la valeur du registre `PC` pour le prochain cycle.

   Écrire la table de vérité du module. En déduire son implémentation dans Logisim ;

4. Le module « *banc de registres* » contient les 16 registres de 8 bits. Il comporte 6 entrées :

   `rd`, `rs`, `rt`. Les indices correspondant, respectivement, au registre à accéder en écriture et aux deux registres à accéder en lecture ;

   `RESET`. Force tous les registres à `0` de manière asynchrone ;

   `regWrite`. Autorise l'écriture dans le registre `R[rd]` ;

   `valIn`. Un entier sur 8 bits à stocker dans `R[rd]` ;

   et deux sorties `Rs` et `Rt`, correspondant aux valeurs des registres `R[rs]` et `R[rt]`.

   Implémenter le banc de registres dans Logisim ;

5. Le module « *sélection de registres* » reçoit en entrée 3 champs de 4 bits de l'instruction courante (voir fig. 2) ainsi que l'indicateur `isJMP`, et il retourne les indices correspondant à `rd`, `rs` et `rt` en fonction du format de l'instruction courante.

   Implémenter le module *sélection de registres* ;

6. Implémenter le circuit complet du SC-1 en réutilisant les sous-circuits développés dans les questions précédentes.

2 Utilisation du SC-1

On va désormais utiliser le SC-1 pour exécuter des programmes écrits en code machine SC-1.

1. D'après la figure 2, quelle est la méthode utilisée pour implémenter l'instruction `halt` ? Quelle contrainte impose t-elle sur les programmes en code machine exécutés ?
2. Traduire en assembleur SC-1 puis en code machine le programme ci-dessous :

```
/* Calcul du pgcd de i et j. */
 int i, int j;
 while (i != j) {
   if (i > j) {
      i -= j;
   } else
      j -= i;
 }
 /* Le pgcd de i et j apparaît dans ces deux variables. */
```

   Charger le programme dans la mémoire du SC-1 et l'exécuter ;

3. Proposer un autre programme non trivial tirant parti du jeu d'instructions du SC-1. Le traduire en assembleur SC-1 et en code machine. Sauver ce code machine dans un fichier *via* Logisim et fournir ce fichier dans l'archive rendue ;

Question bonus (hors barème) Le processeur SC-2 est basé sur le SC-1 mais il offre la possibilité de programmer des routines (fonctions). Proposer une modification de l'architecture du SC-1 pour obtenir le SC-2 et l'implémenter.