Projet – Circuits et architecture
À rendre : 16/12/2022 et 06/01/2023 avant 18h

1 Présentation

Ce projet a un double but :

Câbler dans `logisim` un micro-processeur qui implante un sous-ensemble des instructions du micro-processeur LC-3, et

Programmer avec ces instructions un petit jeu de routines.

Le point de départ est le circuit `logisim LC-3-v0.circ`, que vous trouverez sur la page web du cours [1]. Il est **essentiel de modifier précisément ce fichier** pour réaliser votre projet.

Le projet comporte deux niveaux : un *minimum requis* et un *projet avancé*.

1.1 Instructions LC-3 à implémenter

Le circuit principal `main` utilise plusieurs sous-circuits, dont les plus importants sont : `ALU`, `BiClock`, `DecodeIR`, `GetAddr`, `NZP`, `RamCtrl`, `RegPC`, `WriteVal`. Les sous-circuits coloriés en noir dans `main` sont complètement câblés et **ne doivent pas être modifiés**. Le circuit `main` exécute correctement les instructions `NOT`, `ADD` et `AND` du LC-3.

Avant de commencer le câblage, vous devez comprendre le fonctionnement du circuit distribué. Une fois ce fonctionnement compris, vous devez implémenter un certain nombre d'instructions.

Minimum requis : vous devez modifier le câblage (et pas l'interface) des circuits coloriés en rouge afin d'implémenter les instructions `LEA`, `LDR`, `LD`, `ST`, `STR`, `BR` et `JMP`, ainsi que `SCAN` décrite ci-dessous.

Projet avancé : vous devez compléter les modules `RegPC` et `WriteVal` afin d'implémenter les instructions `JSR` et `JSRR`. Vous pouvez aussi modifier *minimalement* le reste du circuit.

La sémantique de ces instructions est celle vue en cours [2], en particulier "Cours n° 6 : description du LC-3". Vous pouvez consulter également le site internet du livre de Patt & Patel [3], en particulier les annexes A et C. L'instruction SCAN, qui permet de calculer une somme modulo 2 cumulée, est décrite à la section suivante.

1.2 Instruction somme cumulée

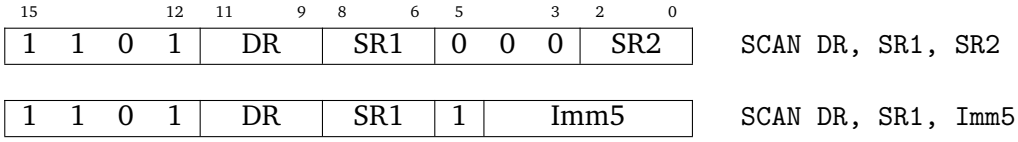
On souhaite étendre le jeu d'instructions du LC-3 avec une nouvelle instruction arithmético-logique SCAN capable de calculer la somme modulo 2 cumulée du registre passé en argument. Pour cela, il faut modifier le module `ALU` uniquement.

L'instruction `SCAN DR, SR1, SR2` stocke dans le bit d'indice `i` du registre `DR` la somme modulo 2 du ET logique des $i+1$ premiers bits de `SR1` et `SR2`. Autrement dit, après son exécution on a

$$\mathtt{DR}_i \leftarrow \bigoplus_{j=0}^{15} (\mathtt{SR1}_j \cdot \mathtt{SR2}_j)$$

pour tout $0 \leq i < 16$. L'instruction `SCAN DR, SR1, Imm` est une variante de la précédente où le deuxième opérande est fourni sous la forme d'une valeur immédiate codée sur 5 bits. Cette valeur est étendue sur 16 bits via une extension signée avant l'opération.

Le code opération (*opcode*) des deux instructions est celui laissé libre en LC-3, à savoir `1101`. Comme pour l'instruction `AND`, on utilise le sixième bit pour distinguer le cas où le deuxième opérande prend la forme d'un registre de celui où il prend celle d'une valeur immédiate. On obtient donc les codages suivants.



Chacune des deux variantes des deux instructions met à jour les indicateurs `n`, `z` et `p`.

1.3 Programmation en LC-3 étendu

Le circuit construit vous servira pour tester deux ensembles de routines écrites en assembleur LC-3 que vous devez implémenter. Le premier sert à manipuler le code de Gray, le second des chaînes de caractères.

1.3.1 Code de Gray

On rappelle qu'un *code* sur k bits est une bijection C entre les entiers plus petits que $2^k - 1$ et les séquence de k bits. Un code de Gray est un code tel que pour tout entier n codable, les suites $C(n)$ et $C(n+1)$ ne diffèrent que d'un seul bit. Par abus de langage, on désigne aussi sous le nom de "code de Gray" un code de Gray particulier, le *code binaire réfléchi*. Celui-ci est obtenu en appliquant la formulation récursive qui suit, où C_k désigne le code binaire réfléchi sur k bits.

$$C_1(n) = n$$
$$C_{2k}(n) = \begin{cases} 0.C_k(n) & \text{si } 0 \leq n < 2^k \\ 1.C_k(2^{k+1} - 1 - n) & \text{si } 2^k \leq n < 2^{k+1} \end{cases}$$

À titre d'exemple, les tables suivantes décrivent C_1, C_2 et C_3.

n	$C_1(n)$
0	0
1	1

n	$C_2(n)$
0	00
1	01
2	11
3	10

n	$C_3(n)$
0	000
1	001
2	011
3	010
4	110
5	111
6	101
7	100

Conversion codage binaire vers Gray. La routine `bin2gray` prend un argument un entier x et renvoie l'entier $y = C_{16}(x)$ Vous pouvez utiliser SCAN dans votre routine.

Conversion codage de Gray vers binaire. La routine `gray2bin` prend en argument un entier x et renvoie l'entier $y = C_{16}^{-1}(x)$. Autrement dit, elle convertit depuis le code de Gray vers le codage binaire classique.

1.3.2 Manipulation de chaînes de caractères

On veut programmer un petit jeu de fonctions manipulant des chaînes de caractères proches de celles offertes par la bibliothèque standard du langage C. On suppose que les caractères sont codés sur 16 bits. Les chaînes sont représentées par des séquences de caractères terminées par le caractère nul (0), comme en langage C.

Première occurence. La routine `index` calcule en `R0` l'adresse de la première apparition d'un caractère dont le code est donné en `R2` pour une chaîne de caractères dont l'adresse de début est donnée en `R1`. La valeur de `R0` est `0` si le caractère est absent.

Dernière occurence. La routine `rindex` calcule en `R0` l'adresse de la dernière apparition d'un caractère dont le code est donné en `R2` pour une chaîne de caractères dont l'adresse de début est donnée en `R1`. La valeur de `R0` est `0` si le caractère est absent.

Comparaison. La routine `strcmp` calcule en `R0` un entier plus grand, égal ou plus petit que zéro selon que la chaîne dont l'adresse est donnée en `R1` est plus grande, égale ou plus petite (dans l'ordre lexicographique) que la chaîne dont l'adresse est en `R2`.

Copie. La routine `strcpy` copie la chaîne source dont l'adresse de début est donnée en `R1` dans la chaîne destination dont l'adresse est en `R2`. Le caractère nul fait partie des données copiées.

Copie de taille fixée. La routine `strncpy` copie la chaîne source dont l'adresse de début est donnée en `R1` dans la chaîne destination dont l'adresse est en `R2` sur un nombre de caractères donné par le registre `R0`.

1.4 Consignes

Pour la mise au point de vos programmes, nous vous recommandons d'utiliser les outils de simulation LC-3 `lc3tools` [4]. Nous attendons de vous :

(minimum requis) d'avoir codé ces fonctions comme autant de programmes indépendants,

(projet avancé) d'avoir codé ces fonctions comme des routines qui ne modifient pas d'autres registres que ceux du résultat. Vous pouvez soit utiliser une mémoire de sauvegarde par routine ou implémenter une pile.

2 Paquetage

Le projet à rendre doit contenir :

Circuit : Le fichier `logisim` `LC-3-v1.circ` (respectivement `LC-3-v2.circ`) contenant le circuit complété pour le rendu intermédiaire (respectivement rendu final) du projet. Vous devez écrire des programmes de test (de type `.mem`) pour chaque instruction.

N'hésitez pas à ajouter des commentaires explicatifs dans le circuit.

Programmes : Les fichiers assembleur LC-3 (de type `.asm`) avec le code des routines de rotation. Ce code précisera les données de test que vous avez utilisées.

Rapport : Vous devez rendre un rapport expliquant la démarche suivie et l'avancement atteint. Il faut énumérer rapidement les changements du circuit original et montrer qu'ils réalisent bien les fonctions souhaitées. Votre rapport doit aussi comporter le code des programmes de test utilisés et justifier qu'ils illustrent le fonctionnement du circuit. Le rapport ne doit pas dépasser cinq pages grand maximum.

3 Modalités pratiques

Le projet s'effectue *en binôme*. *Une pénalité de 30% sera appliquée à la note finale pour les groupes de 3 ou 1 étudiant(s). Les groupes de plus de 3 étudiants ne sont pas acceptés.*

Le projet donnera lieu à une soutenance en janvier. La note tiendra également compte de la lisibilité du circuit, de la simplicité de la solution choisie, de la qualité du rapport, et de la capacité à répondre aux questions lors de la soutenance. Bien que les soutenances aient lieu par binôme, la note est individuelle.

3.1 Remise du projet

Le rendu du projet se fera en deux étapes, par courriel [1] aux enseignants des travaux dirigés :

Rendu intermédiaire, le 16 décembre 2022 avant 18h : le circuit (fichier `LC-3-v1.circ`) avec l'instruction `SCAN` câblée et les fichiers `.asm` contenant vos routines de rotation implémentées comme des programmes indépendants. **Les binômes seront figés** par ce rendu.

Rendu final, le 6 janvier 2023 avant 18h : le paquet complet avec les circuits LC-3 (fichiers `LC-3-v1.circ` et `LC-3-v2.circ`), tous les programmes assembleur LC-3 finaux (routines comme fonctions, les tests de ces fonctions et des circuits) ainsi que le rapport.

Versions

10-11-2022 Version initiale.

Références

[1] https://www.irif.fr/~carton/Enseignement/Architecture/

[2] Oliver Carton, Notes de cours, http://www.irif.fr/~carton/Enseignement/Architecture

[3] Yale N. Patt, Sanjay J. Patel, Introduction to Computing Systems : From Bits and Gates to C and Beyond, McGraw-Hill, http://highered.mcgraw-hill.com/sites/0072467509/

[4] https://github.com/chiragsakhuja/lc3tools

1. Aux adresses guatto@irif.fr (étudiants de M1) et briere@esiea.fr (étudiants EIDD).