Projet - Circuits et architecture
À rendre : 9/12/2021 et 06/01/2022 à 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 sous-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 `GETBIT` et `SETBIT` décrites 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. Les instructions GETBIT et SETBIT, qui permettent respectivement de lire et d'écrire un bit déterminé dans un registre, sont décrites à la section suivante.

1.2 Instruction d'accès aux bits

On souhaite étendre le jeu d'instructions du LC-3 avec deux nouvelles instructions arithmético-logiques GETBIT et SETBIT qui lisent et écrivent un bit d'un registre déterminé. Pour cela, il faut modifier le module `ALU` uniquement.

L'instruction `GETBIT DR, SR1, SR2` stocke dans le bit d'indice zéro du registre `DR` le ième bit de `SR1`, où `i` est l'entier codé par les quatre bits de poids faible de `SR2`. Tous les autres bits de `DR` sont mis à zéro. L'instruction `GETBIT DR, SR1, Imm` est une variante de la précédente où l'indice du bit à lire est fourni sous la forme d'une valeur immédiate codée sur 4 bits.

L'instruction `SETBIT DR, SR1, SR2` stocke dans le ième bit de `DR` le bit de poids faible de `SR1`, où `i` est l'entier codé par les quatre bits de poids faible de `SR2`. Tous les bits d'indice $j \neq i$ de `DR`

sont préservés. L'instruction `SETBIT DR, SR1, Imm` est une variante de la précédente où l'indice du bit à écrire est fourni sous la forme d'une valeur immédiate codée sur 4 bits.

Le code opération (*opcode*) des deux instructions est celui laissé libre en LC-3, à savoir `1101`. Comme pour l'instruction `ADD`, 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. Enfin, on utilise le cinquième bit pour distinguer GETBIT de SETBIT. On obtient donc les codages suivants.

15			12	11	9	8	6	5		3	2	0	
1	1	0	1	DR		SR1		0	0	0	SR2		`GETBIT DR, SR1, SR2`
1	1	0	1	DR		SR1		1	0	Imm4			`GETBIT DR, SR1, Imm4`
1	1	0	1	DR		SR1		0	1	0	SR2		`SETBIT DR, SR1, SR2`
1	1	0	1	DR		SR1		1	1	Imm4			`SETBIT DR, SR1, Imm4`

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 un certain nombre de sous-routines LC-3 données ci-dessous. Vous devez implémenter les sous-routines ci-dessous.

1.3.1 Description des sous-routines

Rotation vers la gauche sur `1` bit. La sous-routine `rotl` prend un argument un entier `x` et renvoie l'entier `y` dont le codage binaire est la rotation des bits de `x` d'une position vers la gauche. Le bit de poids faible de `y` est donc le bit de poids fort de `x`. Par exemple,

$$\texttt{rotl}(49185) = \texttt{rotl}((1100'0000'0010'0001)_2) = (1000'0000'0100'00011)_2 = 32835.$$

Vous pouvez utiliser GETBIT et/ou SETBIT dans votre sous-routine.

Rotation vers la gauche sur `k` bits. La sous-routine `rotlK` prend en argument deux entiers `x` et `k` et renvoie l'entier `y` dont le codage binaire est la rotation des bits de `x` de `k` positions vers la droite. Vous *devez* appeler `rotl` dans votre sous-routine.

Rotation vers la droite sur `1` bit. La sous-routine `rotr` prend un argument un entier `x` et renvoie l'entier `y` dont le codage binaire est la rotation des bits de `x` d'une position vers la droite. Le bit de poids fort de `y` est donc le bit de poids faible de `x`. Par exemple,

$$\texttt{rotr}(49185) = \texttt{rotl}((1100'0000'0010'0001)_2) = (1110'0000'0001'0000)_2 = 57360.$$

Vous pouvez utiliser GETBIT et/ou SETBIT dans votre sous-routine.

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 sous-routines qui ne modifient pas d'autres registres que ceux du résultat. Vous pouvez soit utiliser une mémoire de sauvegarde par sous-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 5 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 9 décembre 2021 avant 18h : le circuit (fichier `LC-3-v1.circ`) avec l'instruction `GETBIT` et `SETBIT` câblée et les fichiers `.asm` contenant vos sous-routines de rotation implémentées comme des programmes indépendants. **Les binômes seront figés** par ce rendu.

Rendu final, le 6 janvier 2022 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 (sous-routines comme fonctions, les tests de ces fonctions et des circuits) ainsi que le rapport.

Versions

22-11-2021 Correction d'une typo et clarification des attentes pour le rendu.

04-11-2021 Version initiale.

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

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