École d'ingénieur Denis Diderot
Université Paris Cité

Architecture et Circuits
Projet – Compte rendu

Meyssem SOUSSOU - Bastien ADIVEZE

SOMMAIRE

I - PROJET.....2
II - CONSTRUCTION DES CIRCUITS.....2
III - CODES ASSEMBLEUR LC-3.....5
IV - CONCLUSION.....6

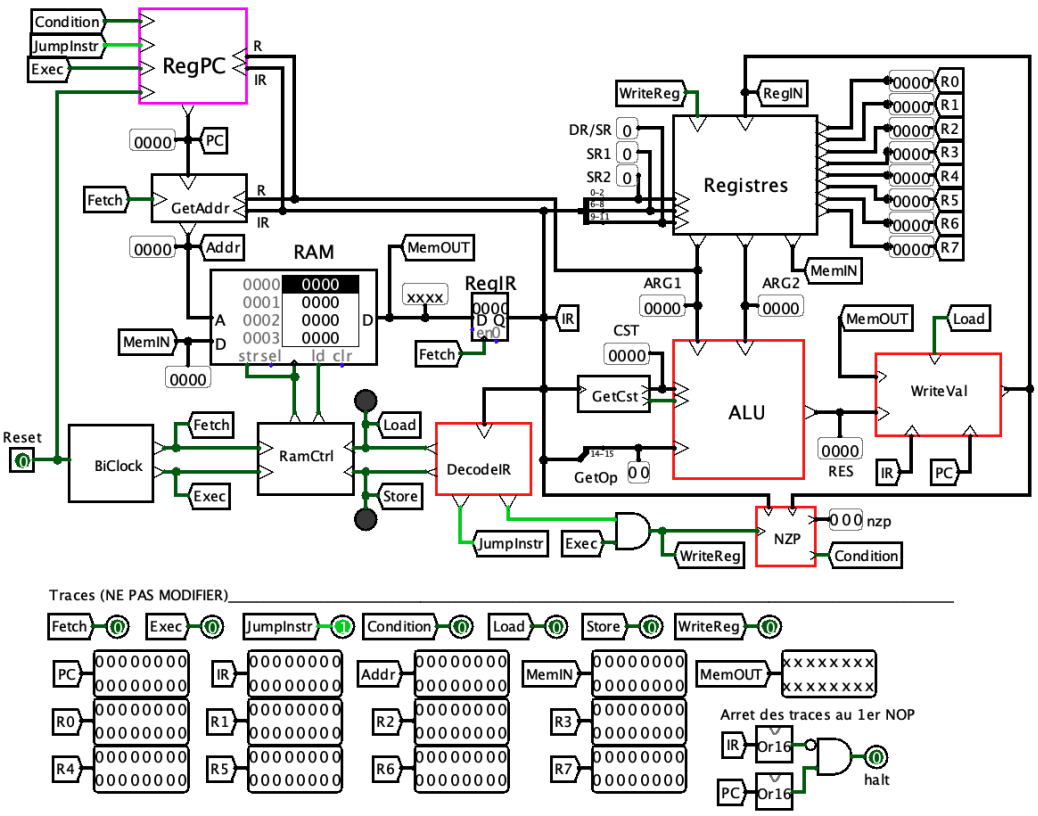

Figure 0 – Circuit du processeur LC-3

I - PROJET

A. Objectif du projet

Le projet consistait à construire et compléter un processeur simplifié de type LC-3 dans Logisim. Il fallait relier et modifier les circuits internes afin de décoder les instructions, effectuer les calculs, accéder à la mémoire, mettre à jour les registres et gérer les sauts conditionnels. L'objectif final était de permettre l'exécution correcte de programmes assembleur sur le processeur conçu.

B. Main

Le circuit main représente le processeur complet vu de l'extérieur. Son rôle est d'orchestrer le fonctionnement global du processeur afin d'exécuter un programme instruction par instruction. Il ne réalise pas directement les calculs, mais coordonne les différents blocs matériels. On peut donc le voir comme le *chef d'orchestre* du processeur.

Le fonctionnement commence avec le PC (Program Counter), qui contient l'adresse de l'instruction courante. Cette adresse est envoyée à la mémoire, qui renvoie l'instruction correspondante. L'instruction est alors stockée dans le registre IR, afin d'être conservée pendant toute son exécution.

L'instruction contenue dans IR est analysée par le circuit DecodeIR, qui lit l'opcode (bits de poids fort) et génère les signaux de contrôle comme *Arith*, *Load*, *Store*, *Jump* ou *WriteReg*. Ces signaux indiquent aux autres blocs quelle action effectuer. En parallèle, GetAddr calcule les adresses mémoire nécessaires et GetCst extrait les constantes immédiates.

Les calculs sont réalisés par l'ALU, qui effectue des opérations arithmétiques ou logiques (addition, XOR, POPCOUNT). Le résultat peut ensuite être écrit dans les registres (R0 à R7), envoyé vers la mémoire ou utilisé pour mettre à jour les flags NZP. Le circuit WriteVal choisit précisément quelle valeur doit être écrite dans un registre.

Le circuit NZP mémorise l'état du dernier résultat (négatif, nul ou positif). Cet état est utilisé par les instructions de branchement conditionnel (BR) pour décider si un saut doit être effectué.

Enfin, le BiClock assure la synchronisation de l'ensemble du processeur en séparant les phases de lecture et d'exécution, garantissant que les registres sont mis à jour au bon moment.

En résumé, le circuit main relie et coordonne tous les composants afin de permettre l'exécution correcte et ordonnée d'un programme sur le processeur.

II - CONSTRUCTION DES CIRCUITS

Le projet consistait à construire et compléter un processeur simplifié de type LC-3 dans Logisim. Il fallait relier et modifier les circuits internes afin de décoder les instructions, effectuer les calculs, accéder à la mémoire, mettre à jour les registres et gérer les sauts conditionnels. L'objectif final était de permettre l'exécution correcte de programmes assembleur sur le processeur conçu.

1. ALU

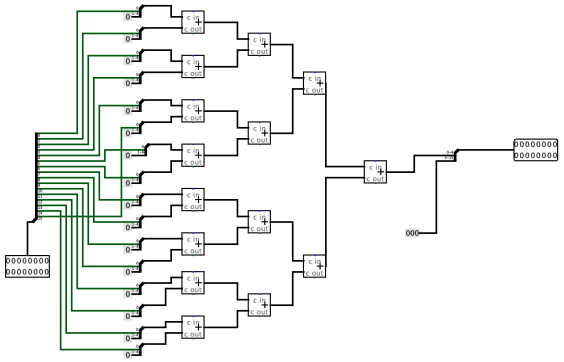

Figure 1 – Circuit Popcount

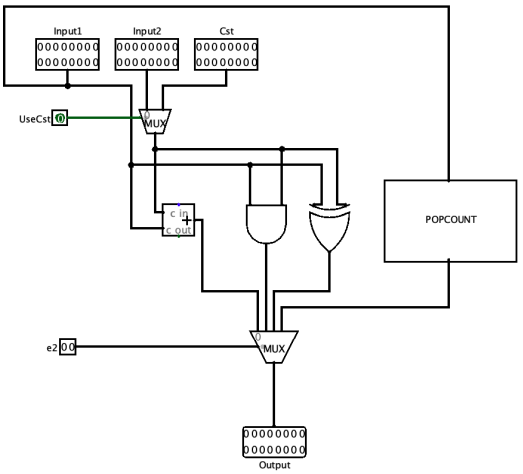

Figure 2 – Circuit ALU avec XOR et Popcount

L'ALU (Arithmetic Logic Unit) est le circuit chargé d'effectuer les calculs et opérations logiques du processeur. Elle reçoit deux opérandes principaux (Input1 et Input2) issus des registres, ainsi qu'une constante (Cst) utilisée pour les instructions immédiates.

Un multiplexeur permet de choisir entre Input2 et la constante Cst, selon le signal de contrôle UseCst. Les données sélectionnées sont ensuite envoyées en parallèle vers plusieurs sous-circuits : un additionneur (pour ADD), une porte AND, une porte XOR, et un module dédié POPCOUNT qui compte le nombre de bits à 1.

Chaque sous-circuit calcule son résultat en même temps. Un second multiplexeur de sortie, piloté par le signal d'opération (GetOp), sélectionne alors le résultat correspondant à l'instruction courante. Ce résultat final est envoyé sur le bus Output et peut être écrit dans un registre ou utilisé par d'autres parties du processeur.

2. DecodeIR

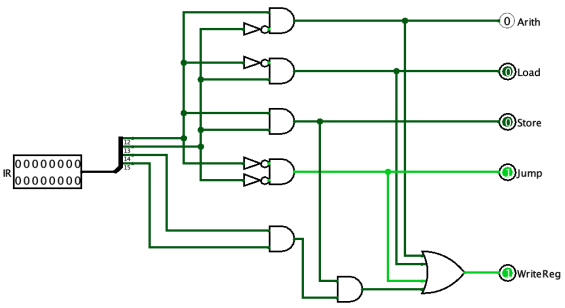

Figure 3 – Circuit DecodeIR

ADD	0001	x1
AND	0101	x5
LD	0010	x2
ST	0011	x3
BR	0000	x0
JMP	1100	xC
LDR	0110	x6
STR	0111	x7

Figure 4 – Tableau des fonctions

Ce montage correspond au circuit DecodeIR, dont le rôle est de reconnaître l'instruction courante à partir du registre d'instruction IR. Les bits IR[15..12] (l'opcode) sont d'abord extraits puis envoyés vers plusieurs portes logiques AND et NOT. Chaque combinaison de portes correspond à une instruction ou à une famille d'instructions (arithmétiques, accès mémoire ou saut).

Par exemple, certaines portes AND détectent les opcodes de type arithmétique (ADD, AND, XOR), ce qui active le signal Arith. D'autres détectent les instructions de chargement (LD, LDR) et activent Load, ou les instructions de stockage(ST, STR) qui activent Store. Une autre combinaison est dédiée aux instructions de saut (BR, JMP), ce qui active le signal Jump.

Enfin, les sorties des différentes portes sont regroupées à l'aide d'une porte OR pour produire le signal WriteReg, qui indique si l'instruction doit écrire un résultat dans un registre. Ce montage permet donc de transformer un opcode binaireen signaux de contrôle clairs, utilisés par le reste du processeur pour exécuter correctement chaque instruction.

3. WriteVal

Le circuit WriteVal a pour rôle de déterminer quelle valeur doit être écrite dans un registre (RegIn) à la fin de l'exécution d'une instruction. Pour cela, il utilise plusieurs multiplexeurs montés en cascade, chacun choisissant entre deux valeurs possibles selon un signal de contrôle.

Concrètement, le premier multiplexeur choisit entre le résultat de l'ALU (ALURes) et la donnée lue en mémoire (MemOUT), en fonction du signal Load. Le multiplexeur suivant permet de remplacer cette valeur par le résultat du calcul d'adresse (PC + Offset9) lorsque l'instruction est LEA. Enfin, un dernier multiplexeur donne la priorité à l'écriture du PC dans le registre lors d'une instruction JSR.

Cette organisation en cascade garantit que les cas particuliers (JSR, LEA) ont priorité sur les cas standards, tout en conservant un circuit simple, lisible et facilement extensible.

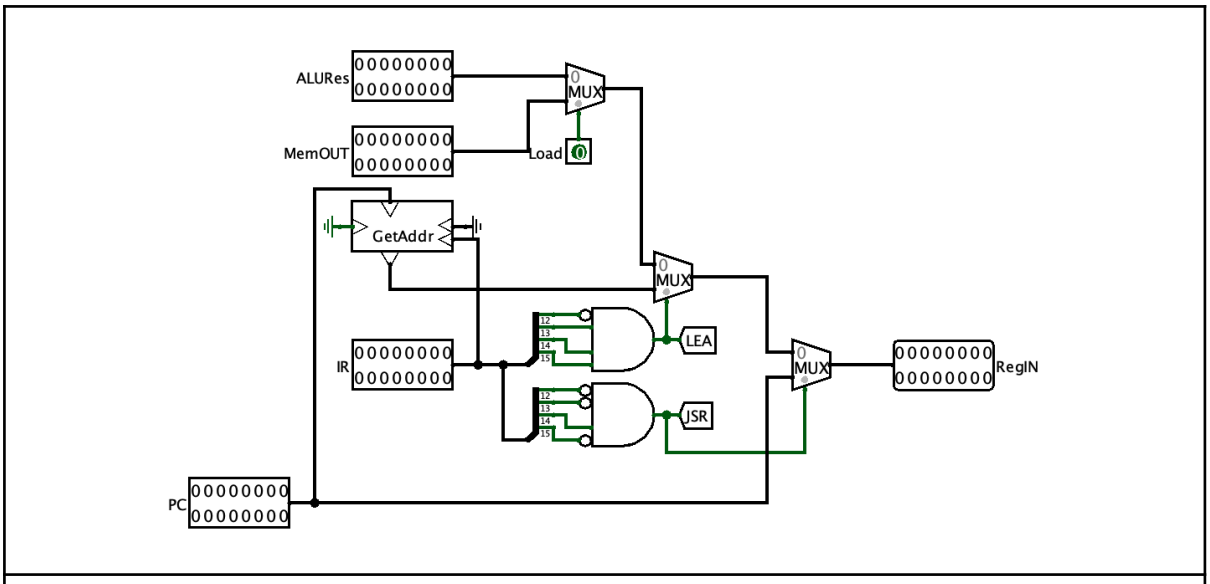

Figure 5 – Circuit WriteVal

4. NZP

Le circuit NZP sert à mémoriser l'état du dernier résultat calculé par l'ALU à l'aide de trois indicateurs : N (résultat négatif), Z (résultat nul) et P (résultat positif). Ces indicateurs sont déterminés à partir du résultat RES puis stockés dans un registre NZP sur 3 bits lorsque le signal WriteReg est actif.

Une horloge a été ajoutée afin de synchroniser la mise à jour des flags avec le cycle d'exécution du processeur. Un NOTest placé sur l'horloge pour inverser le front actif du registre NZP, ce qui permet de mettre à jour les indicateurs après que le résultat de l'ALU soit stable. Les bits NZP sont ensuite comparés avec ceux de l'instruction (IR[11..9]) pour décider si un saut conditionnel (BR) doit être effectué via le signal TestNZP.

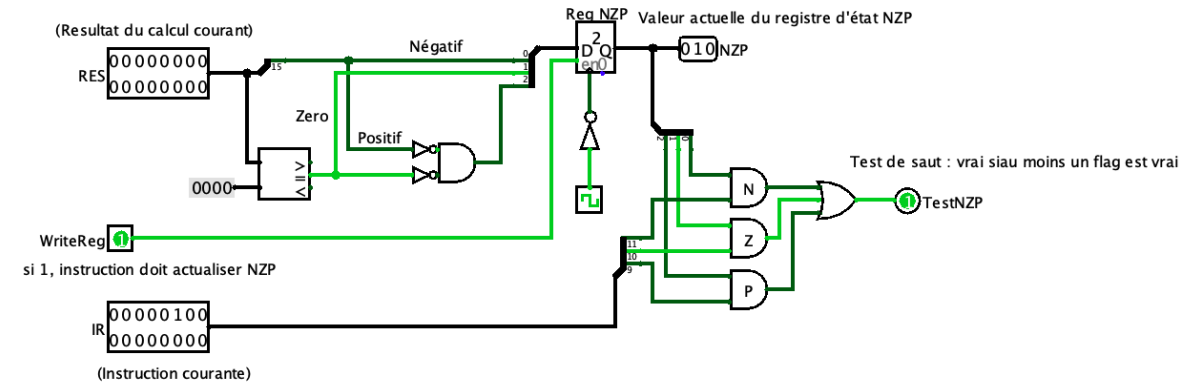

Figure 6 – Circuit NZP

III - CODES ASSEMBLEURS LC-3

Afin de tester le bon fonctionnement du processeur LC-3, plusieurs programmes de test en assembleur ont été développés. Les fonctions index, strcpy et strncpy permettent de valider les instructions de calcul, d'accès mémoire et de branchement, ainsi que la cohérence globale du chemin de données. Les codes peuvent être trouvés dans le dossier du projet.

1. index

Le programme index parcourt une chaîne de caractères stockée en mémoire, dont l'adresse de début est donnée dans R1, et cherche la première occurrence d'un caractère donné dans R2. Si le caractère est trouvé, le programme place dans R0 l'adresse mémoire où il apparaît pour la première fois. Si le caractère n'apparaît pas dans la chaîne, R0 reste à 0.

Le programme commence par mettre R0 à 0, ce qui correspond au cas "caractère absent". Ensuite, il lit la chaîne caractère par caractère avec LDR R3, R1, #0. Si le caractère lu est 0, cela signifie que l'on a atteint la fin de la chaîne, et le programme s'arrête. Sinon, le caractère lu est comparé avec celui recherché (contenu dans R2) en utilisant une soustraction. Si le résultat est nul, le caractère a été trouvé : l'adresse courante (R1) est copiée dans R0, puis le programme s'arrête. Si ce n'est pas égal, R1 est incrémenté et la boucle continue.

2. strcpy
strcpy copie une chaîne de caractères depuis une zone mémoire source vers une zone mémoire destination. L'adresse source est donnée dans R1, l'adresse destination dans R2. La copie continue jusqu'à rencontrer le caractère nul 0, qui marque la fin de la chaîne.

Le programme lit un caractère à l'adresse R1 avec LDR. Ce caractère est immédiatement écrit à l'adresse R2 avec STR. Si le caractère copié est 0, le programme s'arrête : la chaîne est entièrement copiée. Sinon, R1 et R2 sont incrémentés, et la boucle recommence pour le caractère suivant.

3. strncpy
strncpy est une variante de strcpy qui copie au maximum n caractères, où n est donné dans R0. La copie s'arrête soit quand n atteint 0, soit quand la fin de la chaîne (0) est rencontrée.

Le programme commence par vérifier si R0 vaut 0. Si c'est le cas, il s'arrête immédiatement. Sinon, il copie un caractère comme dans strcpy. Après chaque copie, R1 et R2 sont incrémentés et R0 est décrémenté. Tant que R0 est strictement positif, la boucle continue. Si R0 atteint 0 ou si le caractère copié est 0, le programme s'arrête.

IV - CONCLUSION

Ce projet a permis d'implémenter et de tester une version étendue du processeur LC-3 en ajoutant de nouvelles instructions et les circuits de contrôle associés. Les programmes de test en assembleur ont validé le bon fonctionnement du chemin de données, des accès mémoire et des branchements, mettant en évidence le lien entre l'architecture matérielle et l'exécution logicielle.