École d'ingénieur Denis Diderot
Université Paris Cité

Rapport de Projet

Circuits et Architecture des Microprocesseurs

Binôme :

Rababe ZIDANI
Nissrine ELABJANI

Année universitaire : 2024/2025

Table des Matières

Introduction

Ce rapport d'avancement détaille les progrès réalisés dans le cadre du projet LC-3, une composante clé de la matière Circuits et Architecture. Ce projet vise à :

- Programmer un ensemble de routines pour manipuler des chaînes de caractères en utilisant les instructions du microprocesseur LC-3.
- Câbler et modifier des circuits Logisim pour étendre les capacités du microprocesseur LC-3.

Pour aborder la partie programmation, nous avons choisi de commencer par coder les routines en langage C afin de bien comprendre leur fonctionnement logique et les étapes nécessaires. Cette approche nous a permis de tester et valider les algorithmes avant de les convertir en assembleur LC-3. Cela a considérablement facilité le processus de développement.

Nous présentons dans ce rapport les avancements dans la programmation des routines, les détails techniques des circuits, ainsi que les difficultés rencontrées et les solutions envisagées.

Les démarches suivies et lávancement atteint

Nous avons utilisé *Logisim* pour concevoir les circuits en nous basant sur le tableau des instructions LC-3 fourni dans le cours. Les étapes réalisées incluent :

- Implémentation des circuits **NZP v1**, **WriteVal v2**, **SCAN v1**, **DecodeIR v1**, et **ALU v1**.
- Rédaction des pseudo-codes pour les fonctions en C afin de faciliter leur conversion en assembleur.

Malheureusement, certaines fonctionnalités comme `RegPC` n'ont pas pu être implémentées, ce qui a impacté les tests globaux.

1 Programmation en assembleur LC-3

Nous avons commencé par coder les routines suivantes en C pour mieux comprendre leur logique et vérifier leur comportement. Une fois validées, nous avons procédé à leur conversion en assembleur LC-3 :

- **strlen :** Cette routine calcule la longueur d'une chaîne de caractères en parcourant chaque caractère jusqu'au caractère nul. Le résultat est stocké dans le registre R0. Une des difficultés rencontrées a été la gestion correcte de la condition d'arrêt pour s'assurer que le caractère nul soit bien détecté.
- **index :** Cette routine trouve la première occurrence d'un caractère donné dans une chaîne. Nous avons utilisé une boucle pour comparer chaque caractère avec la valeur cible stockée dans R2. Si le caractère est trouvé, son adresse est stockée dans R0 ; sinon, R0 est mis à 0. La gestion des adresses mémoire dans le LC-3 a demandé une attention particulière.
- **rindex :** Similaire à index, cette routine recherche la dernière occurrence d'un caractère. Pour cela, nous avons parcouru la chaîne en sens inverse. Une difficulté ici a été de calculer correctement la longueur de la chaîne avant de commencer la recherche.
- **strcmp :** Cette routine compare deux chaînes de caractères en ordre lexicographique. Les chaînes sont parcourues caractère par caractère jusqu'à trouver une différence ou atteindre la fin. Une attention particulière a été nécessaire pour gérer les cas où les chaînes ont des longueurs différentes.
- **strcpy :** Elle copie le contenu d'une chaîne source dans une chaîne destination. Le caractère nul de fin est également copié. Nous avons rencontré des défis dans la gestion des dépassements de mémoire pour s'assurer que la destination ait suffisamment d'espace.
- **strncpy :** Elle copie un nombre fixe de caractères d'une chaîne source dans une chaîne destination. Une complexité supplémentaire a été d'arrêter la copie correctement si la chaîne source est plus courte que le nombre spécifié.

Ces routines ont été intensivement testées avec divers cas de test pour garantir leur fiabilité. Les principales difficultés ont été liées à la gestion des adresses mémoire et à la détection correcte des conditions limites (par exemple, caractères nuls, débordement de mémoire).

2 Câblage des circuits Logisim

Le câblage des circuits LC-3 a été une tâche complexe, demandant une compréhension approfondie de l'architecture et des interactions entre modules. Voici les détails des modules câblés et leur fonctionnement :

2.1 DecodeIR

Le module DecodeIR est chargé d'extraire les champs des instructions LC-3 (opcode, registres, etc.). **Fonctionnement :**

- Le champ opcode est extrait pour déterminer le type d'instruction à exécuter.
- Les registres source et destination sont identifiés à partir des champs de l'instruction.
- Ces informations sont ensuite propagées vers les autres modules (ALU, NZP, RegPC).

Pour le câblage, nous avons utilisé des multiplexeurs pour sélectionner les bits nécessaires de l'instruction et des décodeurs pour activer les signaux appropriés. Une difficulté majeure a été de garantir que les signaux extraits soient synchronisés avec le reste du circuit, notamment pour les instructions complexes comme BSF/BSB.

2.2 ALU

L'ALU (Arithmetic Logic Unit) est responsable de l'exécution des opérations arithmétiques et logiques. En plus des opérations standard ADD, AND et NOT, nous avons ajouté les instructions BSF (bit scan forward) et BSB (bit scan backward). **Fonctionnement :**

- Les entrées (Input1 et Input2) sont reçues des registres ou du champ immédiat de l'instruction.
- L'opération à effectuer est déterminée par l'opcode extrait dans DecodeIR.
- Les résultats sont calculés et envoyés à la sortie pour mise à jour des registres ou des indicateurs NZP.

L'implémentation des instructions BSF/BSB a nécessité des circuits supplémentaires pour parcourir les bits d'un registre et identifier le bit actif le plus proche.

2.3 NZP

Le module NZP met à jour les indicateurs de statut (négatif, zéro, positif) en fonction du résultat de l'ALU. **Fonctionnement :**

- Si le résultat est négatif, le bit N est activé.

- Si le résultat est nul, le bit Z est activé.
- Si le résultat est strictement positif, le bit P est activé.

Le défi principal a été de gérer correctement les transitions d'état, notamment lorsqu'une instruction ne modifie pas explicitement ces indicateurs.

2.4 Scan

Le module Scan implémente les instructions BSF (bit scan forward) et BSB (bit scan backward). **Fonctionnement :**

- Pour BSF, le circuit parcourt les bits d'un registre de gauche à droite pour identifier le premier bit actif.
- Pour BSB, le parcours se fait dans l'autre sens (droite à gauche).
- L'indice du bit trouvé est ensuite stocké dans le registre destination.

La conception a dû être optimisée pour minimiser la profondeur du circuit et réduire les temps de propagation.

2.5 WriteVal

Le module WriteVal est responsable de l'écriture des résultats dans les registres appropriés en fonction de l'instruction exécutée. **Fonctionnement :**

- Les données peuvent provenir de plusieurs sources : ALU, mémoire ou PC (compteur de programme).
- Un multiplexeur sélectionne la source appropriée en fonction de l'instruction.
- Les données sélectionnées sont ensuite écrites dans le registre cible.

Le câblage a été complexe en raison du nombre de sources possibles et des contraintes de synchronisation entre modules.

3 Défis rencontrés

- **Gestion des adresses mémoire :** Lors de la programmation des routines, la manipulation précise des adresses a demandé une attention rigoureuse pour éviter les erreurs.
- **Optimisation du câblage :** La limitation en termes de profondeur de circuit et de nombre de portes a demandé des ajustements constants pour améliorer les performances sans compromettre la fonctionnalité.
- **Synchronisation des signaux :** Assurer une coordination parfaite entre les différents modules, en particulier DecodeIR et NZP, a été un défi majeur.

Conclusion

À ce stade, nous avons terminé le câblage des modules DecodeIR, ALU, NZP et Scan pour la version 1. Pour la version 2, le câblage de WriteVal est en cours. Les routines LC-3 implémentées sont fonctionnelles et ont été vérifiées.