

SI3 – PARM Project - ArchiPoly

Lassauniere Théo, Galli Evan, Falcoz Alban



Objectif

Développer un modèle simplifié de micro-processeur ARM Cortex M0 sur Logisim Evolution.

Le but à la fin de ce projet est de pouvoir exécuter du code C compilé en assembleur sur le processeur.

Sommaire

I – Présentation des composants

II – Tâches effectuées

III – Démonstrations (CPU + Assembleur)

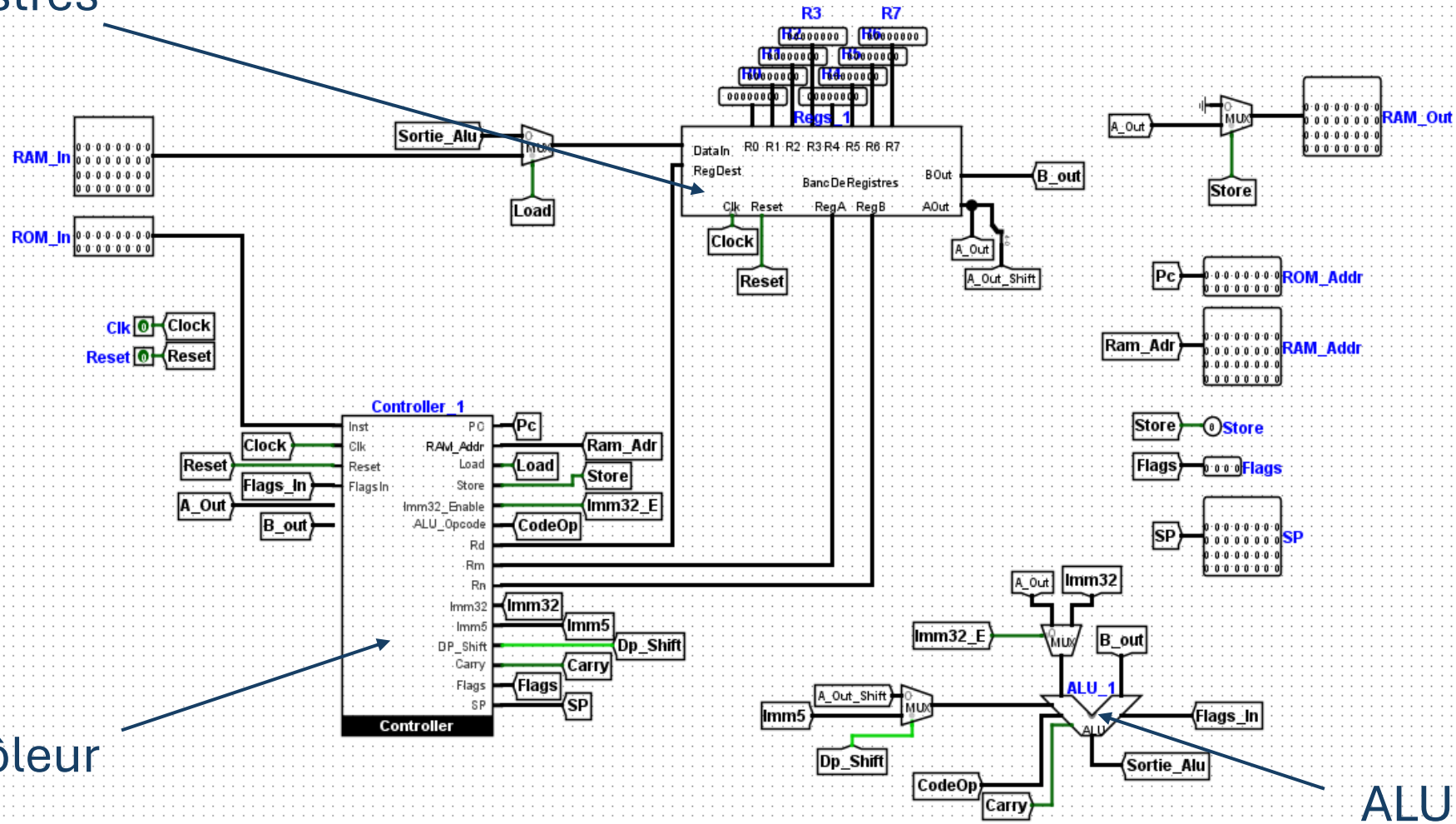
IV – Couverture globale des tests

V – Points forts de notre projet

VI – Démonstration d'un code C compilé passé au CPU

Présentation des composants

Banc de registres



Contrôleur

ALU



Tâches effectuées

- 1 Réalisation de l'ALU et du banc de registres
- 2 Réalisation des premiers composants du contrôleur
- 3 Finalisation du contrôleur
- 4 Réalisation du chemin de données ainsi que de l'assembleur
- 5 Finalisation de l'assembleur
Et génération de code C dans le CPU
- 6 Génération de tests et ajout de fonctionnalités diverses (code C,...)

Branchements du Processeur


Programme assembleur et code C


Tests et extras

Démonstration 1) – tests CPU + Assembleur

Nous allons charger des vecteurs de tests sur différents composants du CPU,
puis nous allons vous montrer notre programme d'assembleur

Couverture globale des tests

| Codes ASM | test passe | non testé |
|---------------------------|--|-----------|
| Conditional |  | |
| DP_1_4 | | |
| DP_5_10 | | |
| DP_11_12 | | |
| DP_13_16 | | |
| Load_store | | |
| SP | | |
| SASM_1_4 | | |
| SASM_5_8 | | |
| taux de couverture | 100% | 100% |

| Codes C | test asm passe | test logisim passe |
|---------------------------|--|--------------------|
| calckeyb |  | |
| calculator | | |
| simple_add | | |
| testfp | | |
| tty | | |
| my own test | | |
| taux de couverture | 100% | 100% |

Points forts de notre projet



Processeur entièrement opérationnel (couvert par plus de 44 000 lignes de test)

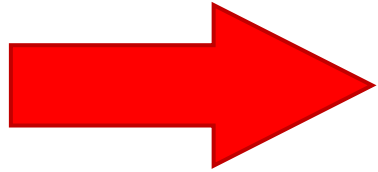


Programme d'assembleur opérationnel, avec possibilité d'exécution de code C sur le processeur



Les + : Ajout d'entrées et sorties diverses (buzzer, slider) et programmation de celles-ci ainsi que du joystick

Points forts de notre projet



Adressage indirect

Notre processeur est capable de prendre en charge les tableaux, ainsi que les pointeurs grâce à l'**adressage indirect**.

Démonstration 2) : passage au CPU d'un code C compilé

Nous allons compiler un code C à l'aide de notre programme, puis nous allons le charger dans le CPU

Merci pour votre attention

