Simulateur du Microprocesseur Motorola 6809

Année Universitaire 2025-2026

Université Hassan 1er Faculté des Sciences et Techniques Settat
Licence Sciences et Techniques en Génie Informatique

Encadré par :

Mr. BENALLA Hicham

Réalisé par :

RACHID Salma

NAAM Yasmine

Remerciements :

Au terme de ce projet, nous souhaitons adresser nos sincères remerciements à notre professeur, Monsieur BENALLA Hicham, pour son encadrement, sa disponibilité et ses conseils précieux tout au long de la réalisation de ce travail. Le projet de conception et de développement d'un simulateur du microprocesseur Motorola 6809 en langage Java a représenté pour nous un véritable défi technique. Il nous a permis de mieux comprendre le fonctionnement interne d'un microprocesseur, notamment la gestion des registres, l'exécution des instructions et l'organisation de la mémoire, en reliant concrètement la théorie à la pratique.

Grâce à la qualité de son accompagnement pédagogique et à son exigence, nous avons pu progresser dans notre apprentissage du langage Java et adopter une méthode de travail plus rigoureuse. Ce projet nous a également aidés à développer notre esprit d'analyse, notre autonomie et notre capacité à travailler en équipe. Cette expérience a été très enrichissante pour notre formation, tant sur le plan académique que personnel.

Sommaire:

Chapitre 1:
Introduction générale

1.1 Contexte générale

L'évolution rapide de l'informatique et des systèmes embarqués a rendu les microprocesseurs omniprésents dans de nombreux domaines tels que l'industrie, l'automobile, les télécommunications et l'électronique grand public. Ces composants constituent le cœur des systèmes informatiques, assurant l'exécution des programmes et la gestion des ressources matérielles.

Dans le cadre de l'apprentissage de l'architecture des ordinateurs et des microprocesseurs, il est essentiel de comprendre non seulement les concepts théoriques, mais également le fonctionnement interne réel de ces composants. Cependant, l'accès au matériel réel peut être coûteux, complexe ou limité dans un environnement pédagogique. C'est dans ce contexte que les simulateurs de microprocesseurs prennent toute leur importance.

Un simulateur de microprocesseur permet de reproduire le comportement d'un processeur réel dans un environnement logiciel. Il offre la possibilité d'observer l'état des registres, la mémoire, le déroulement des instructions ainsi que les mécanismes internes du processeur, tout en facilitant le débogage et l'expérimentation. Ces outils sont particulièrement utiles pour l'apprentissage, le développement et la validation de programmes bas niveau.

1.2 Présentation du microprocesseur Motorola 6809

Le microprocesseur Motorola 6809, introduit à la fin des années 1970, est considéré comme l'un des microprocesseurs 8 bits les plus avancés de son époque. Il a été largement utilisé dans divers systèmes informatiques et embarqués, notamment dans certains ordinateurs personnels, consoles de jeux et équipements industriels.

Le Motorola 6809 se distingue par une architecture relativement sophistiquée, intégrant plusieurs registres généraux, deux accumulateurs, des registres d'index et des registres de pile séparés. Il supporte un jeu d'instructions riche ainsi que plusieurs modes d'adressage, ce qui lui confère une grande flexibilité et une puissance de programmation appréciable.

Sur le plan pédagogique, le 6809 représente un excellent compromis entre simplicité et richesse architecturale. Il permet d'illustrer des concepts fondamentaux tels que le cycle d'exécution des instructions, la gestion de la mémoire, les branchements conditionnels et les mécanismes de débogage, tout en restant accessible à un étudiant en informatique.

1.3 Objectifs du projet

L'objectif principal de ce projet est la réalisation d'un simulateur de microprocesseur Motorola 6809, capable de reproduire le comportement de ce processeur dans un environnement logiciel.
Les objectifs spécifiques du projet sont les suivants :

- Simuler l'exécution du jeu d'instructions du Motorola 6809, en prenant en compte les registres et les modes d'adressage.
- Offrir une visualisation claire de l'état du processeur, incluant les registres, le compteur ordinal et la mémoire.
- Intégrer des fonctionnalités de débogage telles que l'exécution pas à pas et les points d'arrêt.
- Fournir une interface graphique conviviale facilitant l'interaction avec le simulateur.
- Concevoir une architecture logicielle modulaire et extensible, permettant l'ajout ultérieur de nouvelles fonctionnalités ou instructions.

1.4 Méthodologie générale

Pour la réalisation de ce projet, on a suivi une méthodologie progressive afin de développer le simulateur de manière structurée et cohérente. Le travail a été organisé en plusieurs étapes, allant de la compréhension théorique du microprocesseur jusqu'à la validation du simulateur final.

Dans un premier temps, une étude théorique du microprocesseur Motorola 6809 a été réalisée. Cette étape nous a permis de comprendre son architecture générale, le rôle de ses registres, son organisation mémoire ainsi que les principes de son jeu d'instructions et de ses modes d'adressage. Cette phase a été essentielle pour assurer une simulation correcte du fonctionnement du processeur.

Ensuite, une phase de conception a été menée afin de définir l'architecture globale du simulateur. Durant cette étape, les différents composants du système ont été identifiés, notamment le processeur, la mémoire, le jeu d'instructions, le débogueur et l'interface graphique. Cette organisation nous a permis de structurer le projet avant de passer à l'implémentation.

La troisième étape a concerné l'implémentation du simulateur en langage Java, en utilisant l'environnement de développement Eclipse.

Le développement a été effectué de manière progressive, en commençant par les éléments de base tels que la gestion de la mémoire et des registres, puis en ajoutant les instructions, les fonctionnalités de débogage et l'interface graphique réalisée avec la bibliothèque Swing.
Enfin, la dernière étape a été consacrée aux tests et à la validation du simulateur. Des tests fonctionnels ont été réalisés afin de vérifier le bon fonctionnement des différentes instructions, la mise à jour correcte des registres, l'exécution pas à pas ainsi que l'affichage des informations dans l'interface graphique.

Chapitre 2:
Étude du microprocesseur Motorola 6809

2.1 Présentation du Motorola 6809

Le microprocesseur Motorola 6809 est un processeur 8 bits développé par la société Motorola et commercialisé à la fin des années 1970. Il est considéré comme l'un des microprocesseurs les plus avancés de sa génération, notamment grâce à son architecture flexible et à son jeu d'instructions riche.
Contrairement à certains microprocesseurs plus simples de la même époque, le 6809 a été conçu pour faciliter la programmation et offrir de meilleures performances. Il a été utilisé dans plusieurs systèmes informatiques, consoles de jeux et applications industrielles. Aujourd'hui, bien qu'il soit obsolète sur le plan matériel, il reste très intéressant d'un point de vue pédagogique.

2.2 Architecture générale du microprocesseur

Le Motorola 6809 repose sur une architecture classique composée de plusieurs unités fonctionnelles. Il comprend une unité de traitement arithmétique et logique (ALU), une unité de contrôle, ainsi qu'un ensemble de registres internes permettant le stockage temporaire des données.
Le processeur fonctionne selon un cycle d'exécution bien défini, appelé cycle Fetch–Decode–Execute. Durant ce cycle, le processeur lit une instruction depuis la mémoire, la décode afin de déterminer l'opération à effectuer, puis l'exécute. Ce mécanisme constitue la base du fonctionnement de tout microprocesseur.
Le 6809 dispose également d'un bus d'adresses sur 16 bits, ce qui lui permet d'adresser jusqu'à 64 Ko de mémoire. Cette capacité était importante pour l'époque et permettait le développement de programmes relativement complexes.

2.3 Présentation du Motorola 6809

Les registres sont des éléments essentiels du microprocesseur, car ils permettent de stocker temporairement les données et les adresses utilisées lors de l'exécution des instructions.
Le Motorola 6809 dispose de plusieurs registres, parmi lesquels :

- Deux accumulateurs 8 bits : A et B, utilisés pour les opérations arithmétiques et logiques.

- Un accumulateur 16 bits : D, formé par la combinaison des registres A et B.
- Deux registres d'index : X et Y, utilisés principalement pour l'adressage indexé.
- Deux registres de pile : S (pile système) et U (pile utilisateur).
- Un compteur ordinal PC (Program Counter), qui contient l'adresse de la prochaine instruction à exécuter.
- Un registre d'état CC (Condition Code), qui contient des indicateurs représentant l'état du processeur après l'exécution d'une instruction.

2.4 Organisation de la mémoire

Le Motorola 6809 utilise un espace mémoire adressable de 16 bits, ce qui correspond à une capacité maximale de 64 Ko. La mémoire est organisée de manière linéaire, chaque adresse correspondant à un octet. Les instructions et les données sont stockées dans la même mémoire, selon le principe de l'architecture de von Neumann. Le processeur accède à la mémoire pour lire les instructions à exécuter ainsi que pour lire ou écrire les données nécessaires au programme. L'accès à la mémoire se fait à l'aide du compteur ordinal pour les instructions et des registres d'index ou des adresses directes pour les données. La gestion correcte de la mémoire est essentielle pour assurer le bon fonctionnement des programmes.

2.5 Les instructions du motorola 6809

Les instructions du Motorola 6809 sont relativement riches comparé à d'autres processeurs 8 bits. Il comprend plusieurs catégories d'instructions, notamment :

- Les instructions de chargement et de stockage
- Les instructions arithmétiques et logiques
- Les instructions de branchement et de saut
- Les instructions de contrôle du processeur

Ces instructions permettent de manipuler les données, de modifier les registres et de contrôler le flux d'exécution du programme. Chaque instruction est codée sous forme d'un opcode, suivi éventuellement d'opérandes.
Dans le cadre de ce projet, seule une partie d'instructions a été simulée, de manière à obtenir une version fonctionnelle du simulateur tout en restant compatible avec les principes du Motorola 6809.

2.6 Les modes d'adressage

Le microprocesseur Motorola 6809 supporte plusieurs modes d'adressage permettant d'accéder aux données ou d'exécuter des instructions de différentes manières. Le **mode immédiat** utilise une valeur directement incluse dans l'instruction, tandis que le **mode direct** permet d'accéder rapidement à certaines zones de la mémoire. Le **mode étendu** offre un accès à l'ensemble de l'espace mémoire grâce à une adresse sur 16 bits. Le **mode indexé** repose sur l'utilisation des registres d'index, tels que X ou Y, et est particulièrement adapté à la manipulation de tableaux ou de structures en mémoire. Enfin, le **mode inhérent** correspond aux instructions ne nécessitant aucun opérande explicite, l'opération s'appliquant directement aux registres ou à l'état interne du processeur.

Chapitre 3:
Choix des outils techniques

3.1 Choix du langage de programmation

Le langage de programmation Java a été utilisé pour le développement de ce projet. Java offre plusieurs avantages, notamment sa portabilité, sa robustesse et la richesse de ses bibliothèques standard. Il permet également de développer facilement des interfaces graphiques, ce qui est essentiel pour un simulateur destiné à des utilisateurs débutants.

3.2 Outils de développement utilisés

Le développement du simulateur a été réalisé à l'aide de l'environnement de développement intégré Eclipse. Cet outil a été choisi car il est bien adapté au développement en Java et facilite la gestion des projets, des classes et des packages.
Étant débutants, nous avons également installé l'outil WindowBuilder, intégré à Eclipse. Cet outil permet de concevoir des interfaces graphiques de manière visuelle, ce qui simplifie considérablement le développement de l'interface sans avoir à écrire manuellement tout le code graphique.

3.3 Choix de l'interface graphique

Pour la réalisation de l'interface graphique du simulateur, la bibliothèque Swing a été utilisée. Swing est une bibliothèque standard de Java, ce qui la rend facilement accessible et bien documentée.
L'utilisation de Swing, combinée à WindowBuilder, nous a permis de créer une interface graphique simple et intuitive. Cette interface facilite la visualisation des registres, de la mémoire et du déroulement de l'exécution des instructions.

3.4 Méthode d'implémentation

L'implémentation du simulateur a été réalisée de manière progressive. Dans un premier temps, les éléments de base tels que la gestion de la mémoire et des registres du processeur ont été développés. Ensuite, les instructions ont été ajoutées progressivement afin de simuler le comportement du Motorola 6809.
Une fois la partie logique du simulateur mise en place, l'interface graphique a été intégrée afin de permettre une interaction directe avec l'utilisateur. Cette approche progressive nous a permis de mieux comprendre chaque partie du système avant de passer à l'étape suivante.

Chapitre 4:
Conception et implémentation

4.1 Structure du projet

Pour ce projet, nous avons opté pour une architecture modulaire afin de séparer les responsabilités et de faciliter le travail en équipe. Le code est organisé en cinq paquetages principaux, chacun gérant un aspect spécifique de l'ordinateur :

- Le paquetage cpu : C'est le cerveau du système. Il contient la classe CPU6809 qui gère les registres et le cycle d'horloge. Il inclut également la classe Instruction, qui est fondamentale car elle définit comment chaque commande doit se comporter.
- Le paquetage memory : Il ne contient qu'une classe, Memory.java, mais elle est vitale. Elle simule les 64 Ko d'espace adressable et gère les échanges de données entre le processeur et les périphériques virtuels (clavier/console).
- Le paquetage assembler : Ce module contient l'intelligence nécessaire pour traduire notre code source écrit en mnémoniques vers le langage machine (binaire). On y trouve l'assembleur pour charger les programmes et le désassembleur pour l'affichage en temps réel lors du débogage.
- Le paquetage debugger : Ce paquetage contient la logique de contrôle. Il permet de mettre en pause le processeur, de gérer les points d'arrêt et de surveiller l'état interne du système sans interférer avec l'exécution normale.
- Le paquetage gui : Il regroupe toutes les classes liées à l'interface graphique (Swing). C'est ici que nous avons codé le CPUPanel6809 pour dessiner l'architecture et la classe GUI qui assemble tous les composants visuels.
- Le paquetage MAIN : Il contient le point d'entrée unique de l'application qui initialise le thread de l'interface graphique.

4.2 Conception et Implémentation du CPU

Pour coder le CPU, nous avons dû faire face à la complexité des instructions du Motorola 6809. Nous avons implémenté un système de table d'opcodes. Chaque instruction est enregistrée avec sa taille et son nombre de cycles. L'implémentation repose sur une machine à états : le processeur lit l'octet au compteur de programme (PC), identifie l'instruction correspondante, et lance sa logique d'exécution. Nous avons particulièrement soigné le registre d'état (CCR) : chaque calcul (addition, soustraction, etc.) déclenche une vérification des indicateurs Z (Zéro), N (Négatif) ou C (Carry), ce qui permet de gérer les sauts conditionnels indispensables à tout programme.

4.3 Conception et Implémentation de la mémoire et des E/S

La classe Memory sert à créer un grand espace de stockage de 64 Ko . Pour que notre émulateur soit réaliste, nous avons dû régler un premier problème : le 6809 range les grands nombres (16 bits) en deux morceaux. Nous avons donc codé les fonctions readWord et writeWord pour qu'elles rangent toujours le morceau le plus important en premier.
Mais la partie la plus intéressante de notre mémoire, c'est ce qu'on appelle le "Memory-Mapped I/O".

- $FF00$ (La Sortie) : Nous avons codé la mémoire pour que, dès que le CPU pose un chiffre dans cet adresse , cela ne reste pas caché dedans. À la place, ce chiffre est envoyé directement à notre écran (la console). C'est comme cela que le processeur arrive à nous "parler" et à afficher du texte.
- $FF01$ (L'Entrée) : Il reçoit en direct le code de la touche que l'utilisateur est en train de presser sur son clavier.

4.4 Conception et Implémentation du Debugger

Le débogeur a été l'un des modules les plus complexes à lier. Pour gérer les breakpoints, nous avons utilisé une collection de type HashSet. L'idée est de vérifier, avant chaque instruction, si l'adresse actuelle du PC est présente dans cette liste. Pour l'exécution continue (bouton "Run"), nous avons dû implémenter un thread séparé (via SwingWorker). Sans cela, la boucle infinie du CPU bloquerait toute l'interface. Grâce à ce thread, le CPU travaille en arrière-plan pendant que la fenêtre Swing reste réactive pour que l'utilisateur puisse cliquer sur "Stop" à tout moment.

4.5 Conception et Implémentation de l'interface graphique

L'interface a été codée entièrement avec Java Swing, en utilisant un Layout Null pour placer nos composants précisément comme sur un schéma technique. Nous avons créé un composant personnalisé, CPUPanel6809, qui utilise l'API Graphics2D pour dessiner un polygone représentant l'UAL (Unité Arithmétique et Logique).
Pour l'affichage de la mémoire, nous avons utilisé des JTable dynamiques.

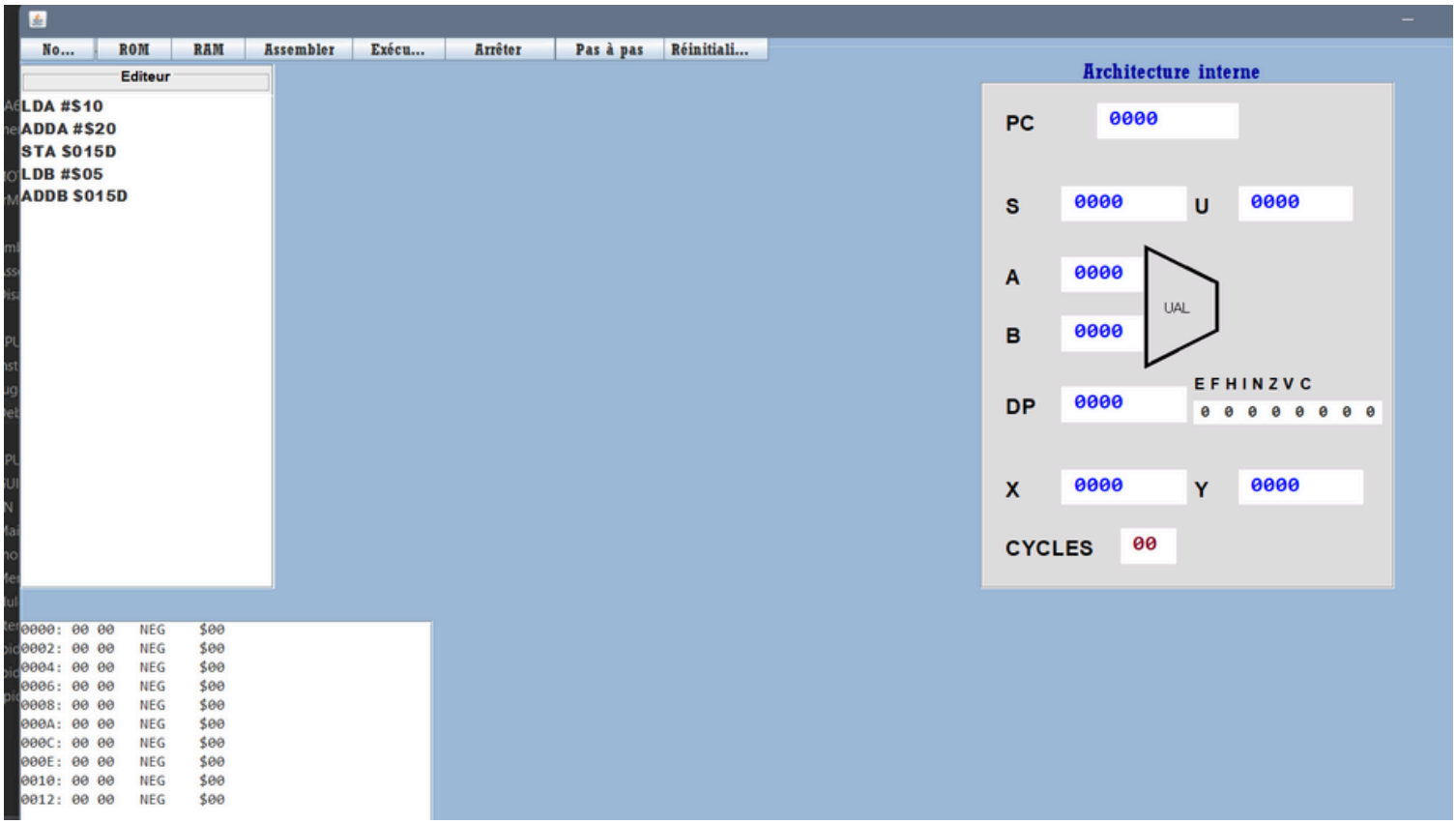
Chapitre 5:
Tests du simulateur

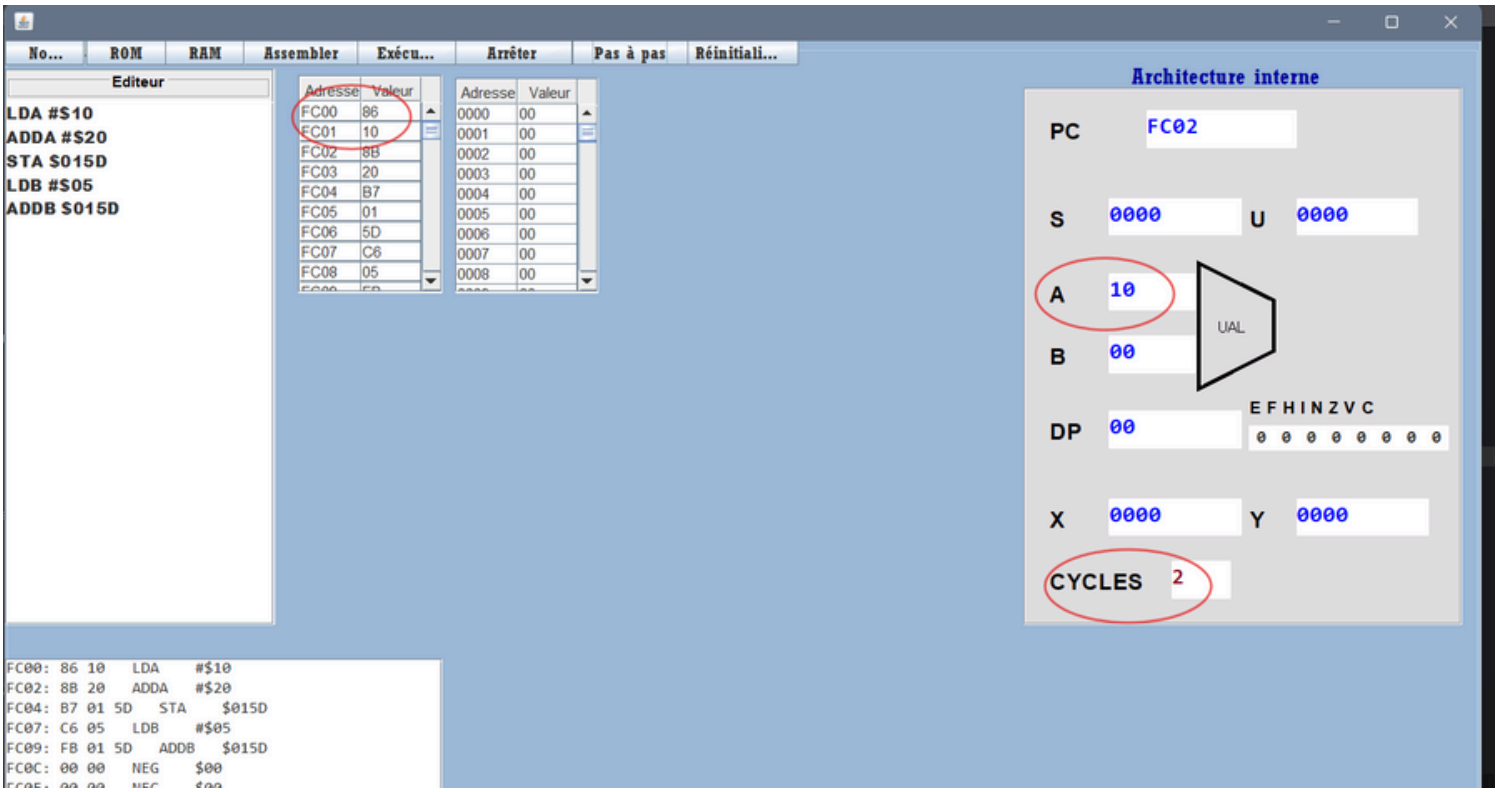
Dans cet chapitre nous allons explorer plusieurs instructions d'assemblage qui utilisent différents modes d'adressage. Chaque instruction sera accompagnée d'une brève explication de son fonctionnement et de son mode d'adressage spécifque. Cela permettra de mieux comprendre comment les données sont manipulées et stockées dans la mémoire, ainsi que les différentes manières d'accéder à ces données.

Il est important de noter que notre simulateur ne prend pas en charge l'ensemble d'instructions du Motorola 6809. Seules certaines instructions essentielles ont été implémentées dans le cadre de ce projet, afin d'obtenir une version fonctionnelle du simulateur.

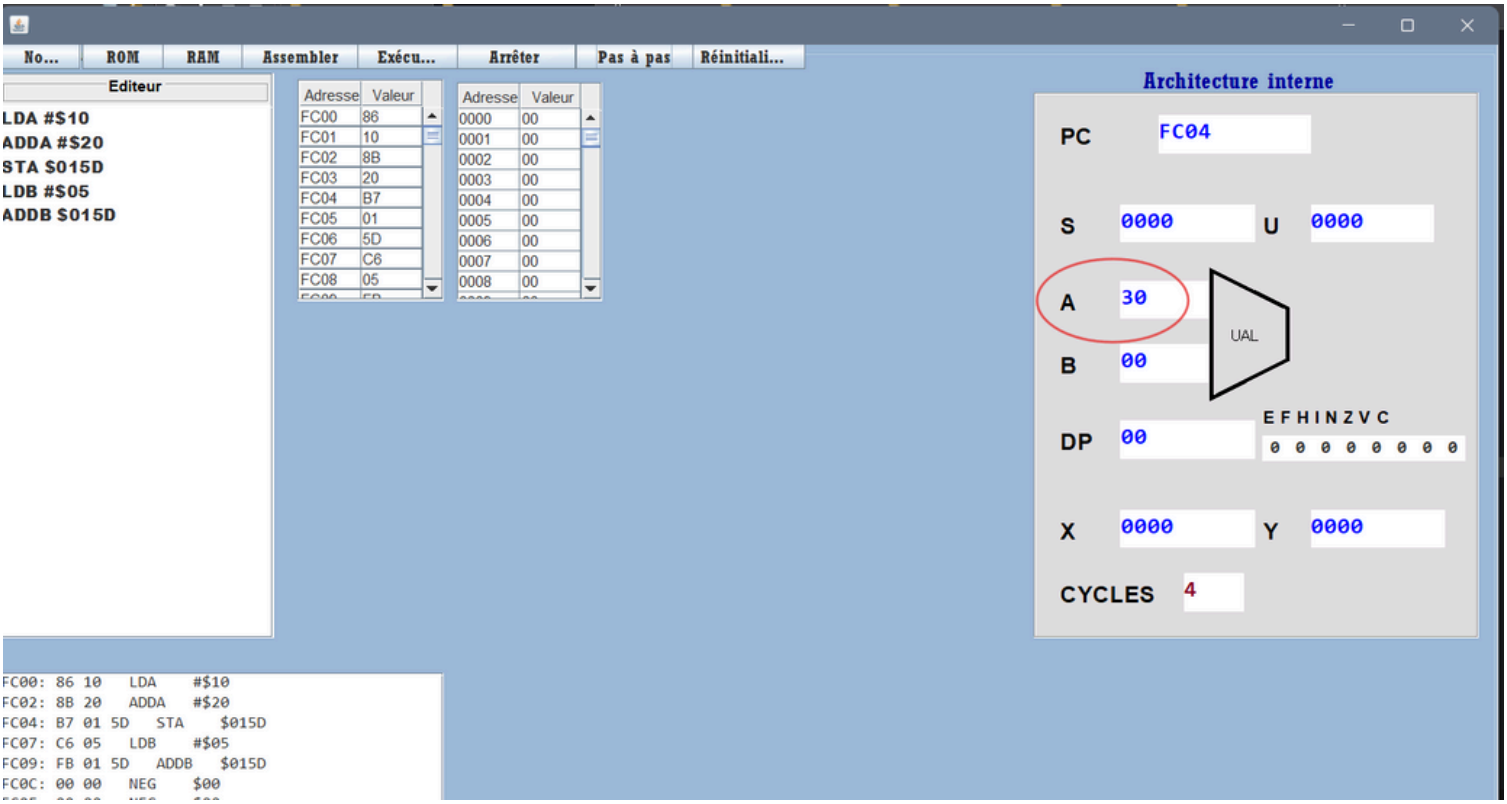
5.1 Scénario de test et exécution pas à pas

Ce programme additionne deux nombres et transfère le résultat du registre A vers le registre B en passant par la mémoire.

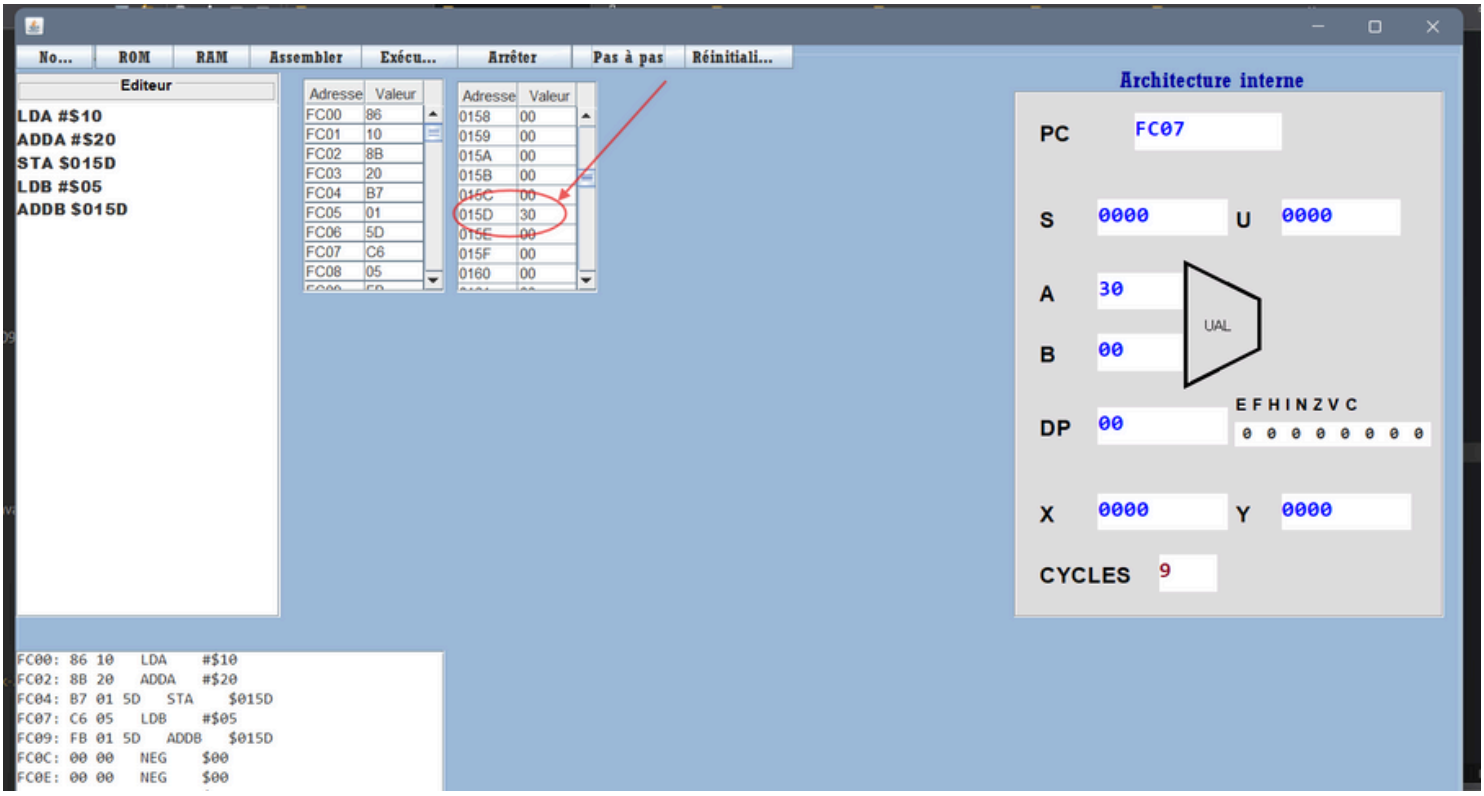




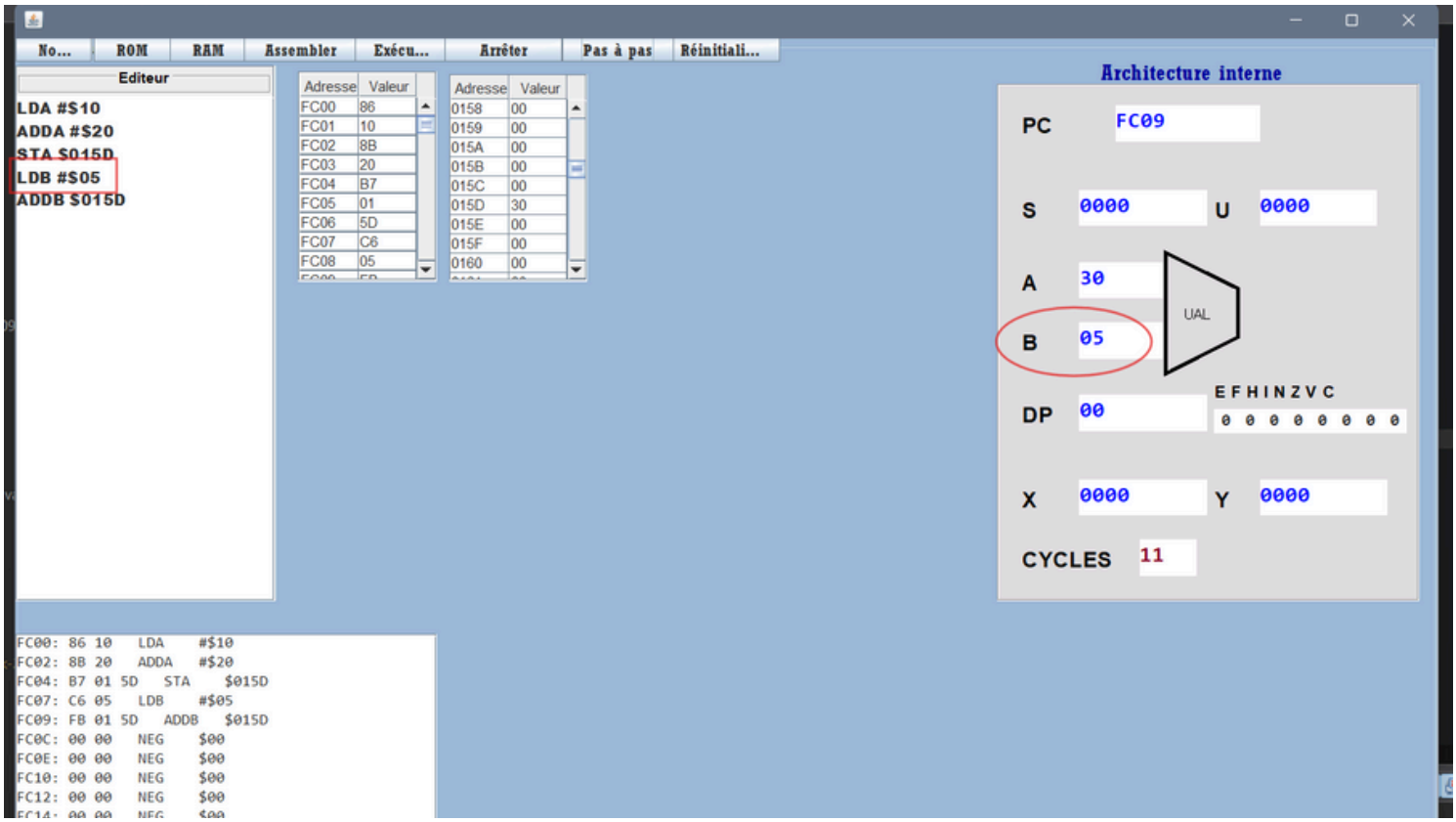
Le PC (Program Counter) avance et le registre A est chargé par la valeur 10 . L'adressage immédiat est donc fonctionnel, aussi il y a affichage des opcodes dans la ROM et affichage du nombre de cycles.



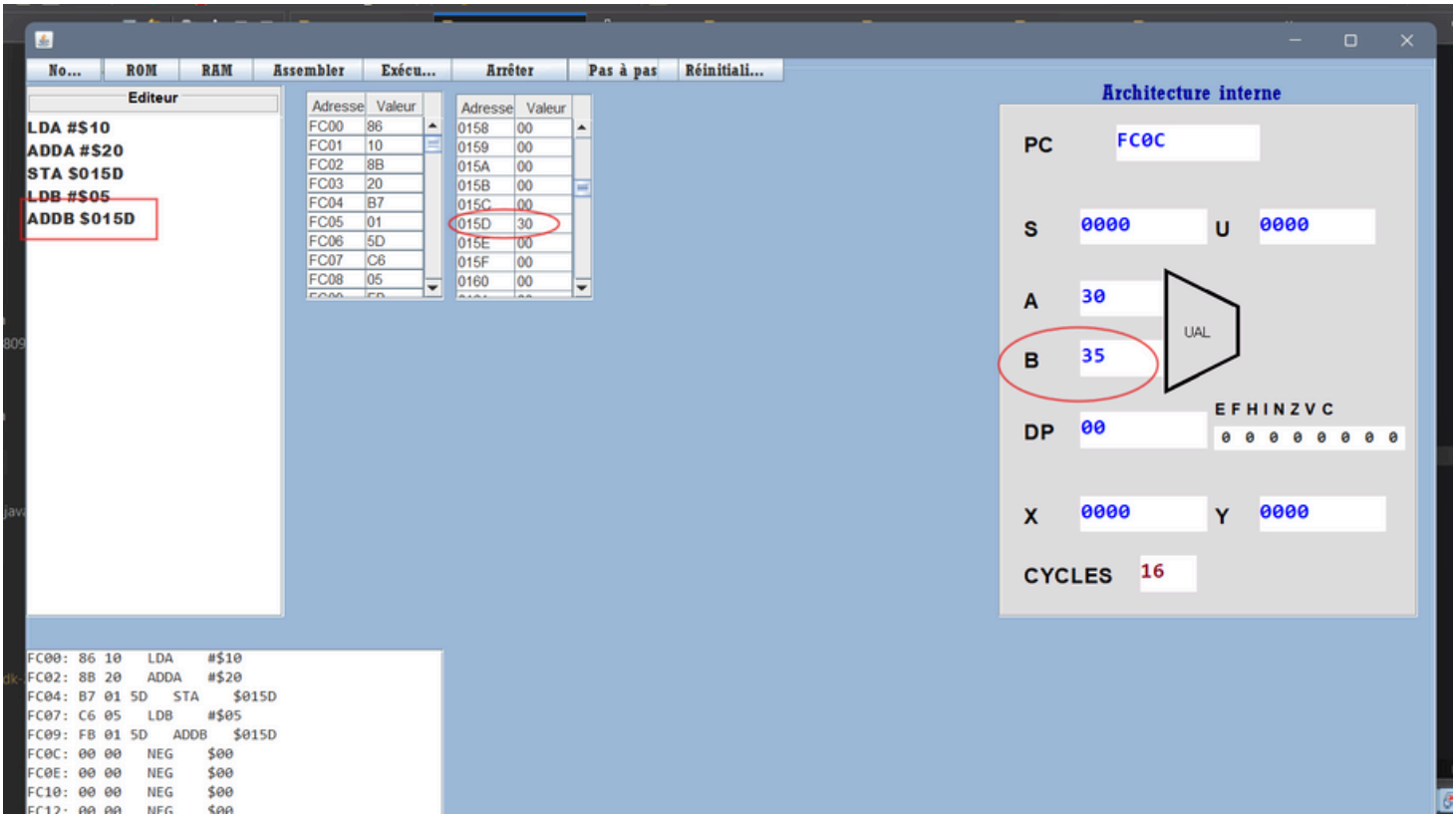
Le registre A affiche maintenant 30. L'UAL a correctement additionné $10 + $20. On vérifie également que les flags du registre d'état ne signalent pas d'erreur.



On voit dans la RAM que la case $015D contient la valeur 30. Le bus de données entre le CPU et la RAM fonctionne parfaitement.

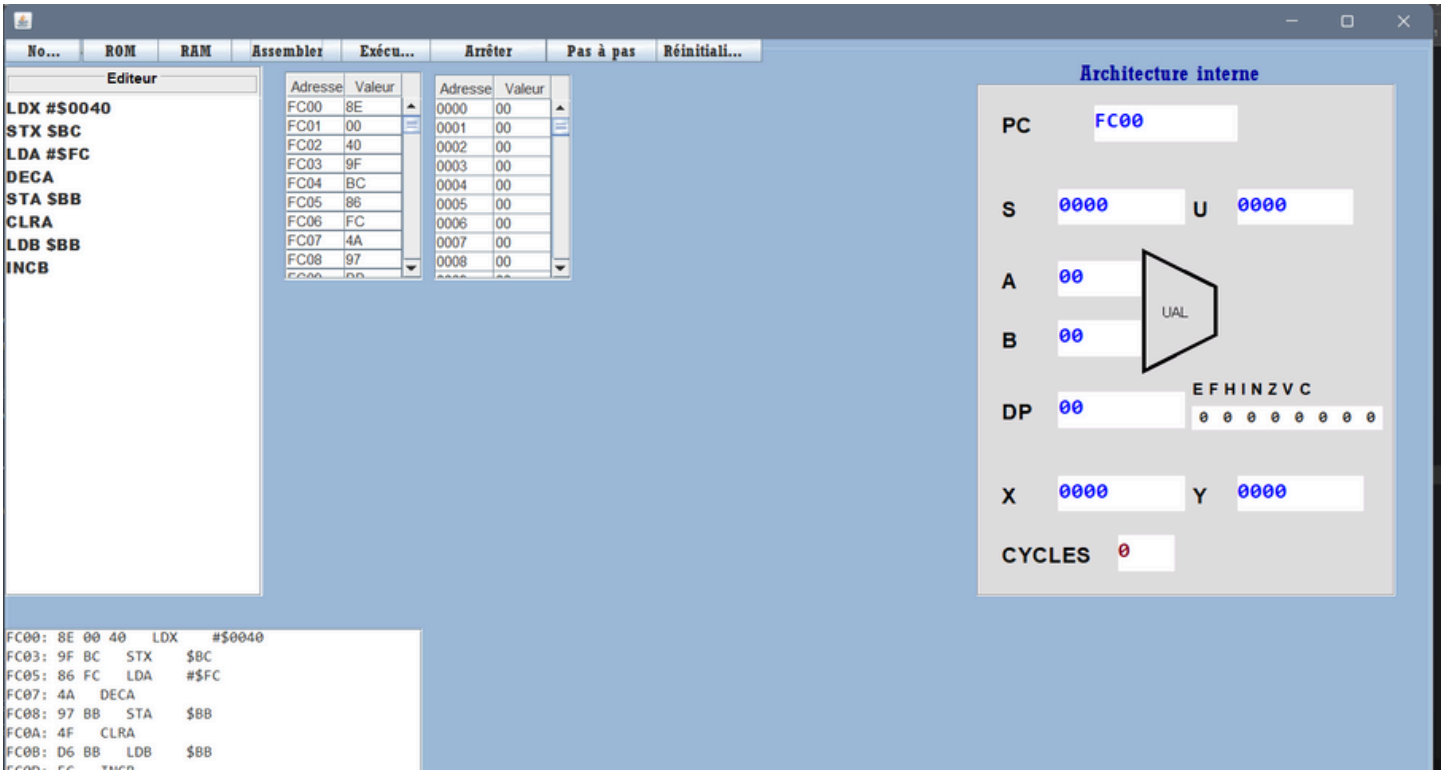


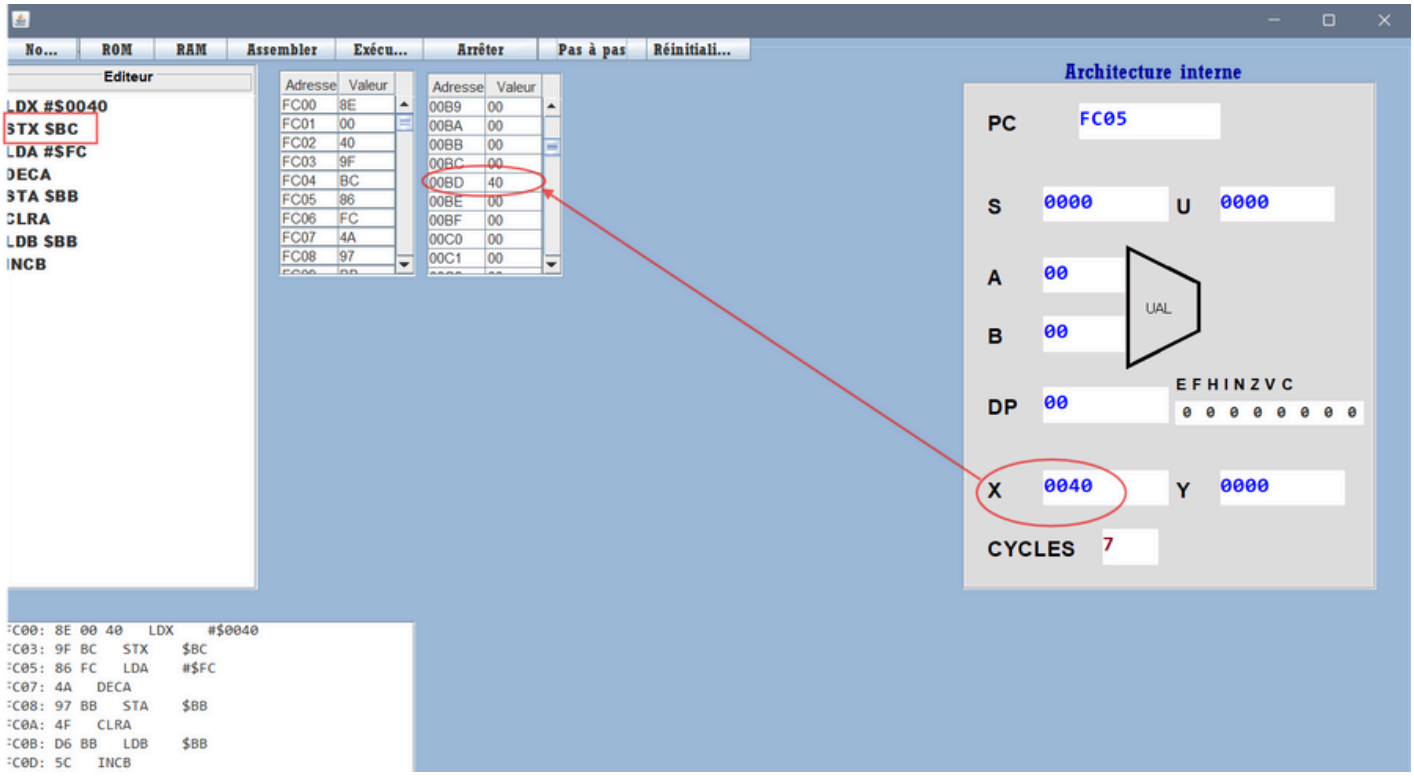
Le registre B est chargé par la valeur 05.



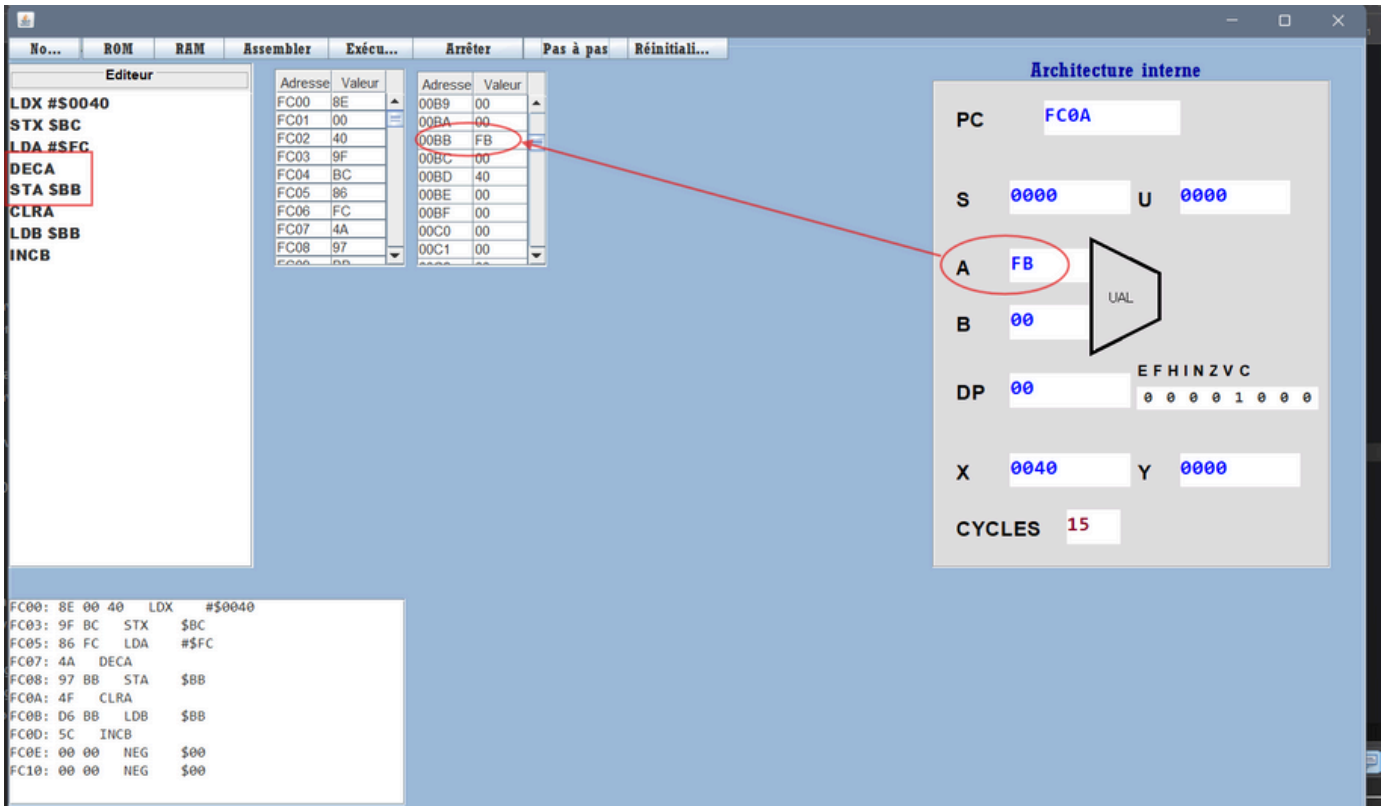
On additionne la valeur de B avec la valeur dans l'adresse $015D. FIN DU PROGRAMME.

5.2 2éme Test

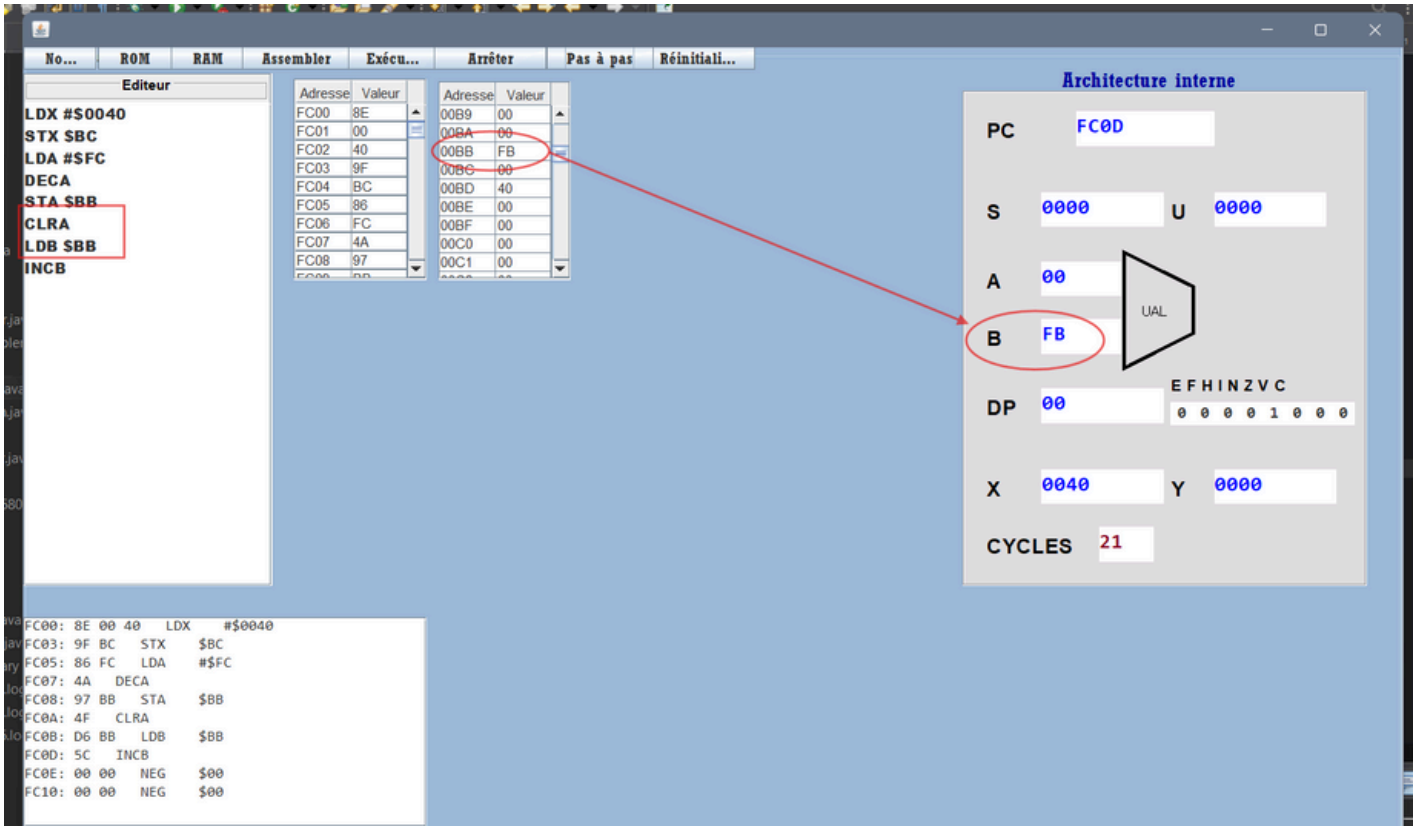




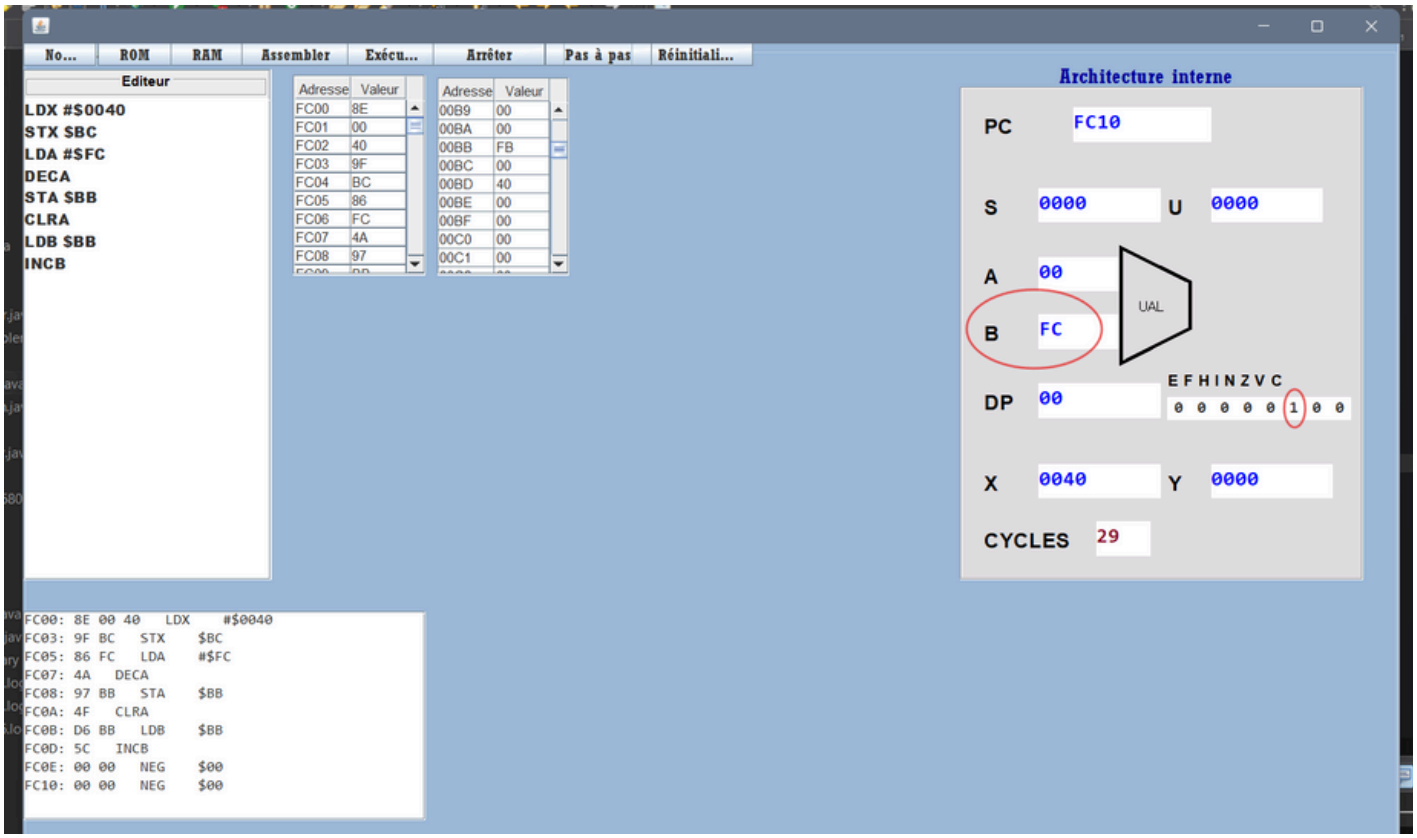
La valeur chargé dans X va etre stocké dans la valeur de l'adresse mémoire +1 (DP vaut $00)



la valeur chargé dans A se decremente et on la charge dans l'adresse mémoire $00BB car DP vaut $00



On vide la case du Registre A et le registre B est chargé d'après l'adresse mémoire $00BB comme on voit dans le tableau de la RAM



La valeur du registre B s'incremente et puisque il y a pas d'autres instructions à exécuter la flag Z prend 0 comme valeur.

Chapitre 5:
Limites et perspectives

Malgré l'intérêt et la richesse de ce projet, certaines limites doivent être prises en compte. La conception et le développement d'un simulateur du microprocesseur Motorola 6809 représentent en effet une tâche complexe qui nécessite du temps et une bonne maîtrise des outils de programmation. Nous n'avons pas pu implémenter l'ensemble des instructions du processeur, leur nombre élevé et leur complexité nous ayant conduits à nous concentrer sur un sous-ensemble représentatif. Cette contrainte est principalement liée au temps limité dont nous disposions, que nous devions partager entre la réalisation du projet et la préparation des examens.

Par ailleurs, ce travail a été réalisé durant notre première année d'apprentissage du langage Java. La découverte de ce langage, combinée à l'étude de l'architecture d'un microprocesseur, a constitué un double défi qui a parfois ralenti notre progression. De plus, le travail en binôme, étant notre première expérience de projet en groupe, a nécessité une phase d'adaptation pour organiser la répartition des tâches et assurer la cohérence du code développé.

Malgré ces limites, ce projet ouvre de nombreuses perspectives d'amélioration. Il pourrait notamment être enrichi par l'implémentation complète des instructions du Motorola 6809, ainsi que par une meilleure gestion des modes d'adressage et des interruptions. Avec une maîtrise plus avancée de Java et une expérience accrue du travail collaboratif, ce projet pourrait être approfondi à l'avenir et servir de base à des réalisations plus ambitieuses dans le domaine de l'architecture des ordinateurs.

Références :

Le développement de ce projet a bénéficié de diverses ressources qui ont été essentielles pour surmonter les défs techniques et améliorer la qualité du code.

https://www.w3schools.com/
https://github.com/
https://chat.openai.com/
BENALLA H., Cours d'Architecture des ordinateurs Supports de cours, FSTS .
https://docs.oracle.com/javase/