جامعة الحسن الأول
UNIVERSITÉ HASSAN 1ER

كلية العلوم
و التقنيات سطات
FACULTÉ DES SCIENCES
ET TECHNIQUES SETTAT

Département : Mathématiques et Informatiques

Filière : LST Génie Informatique

Rapport de Projet

Simulateur de Microprocesseur MOTOROLA 6809

Réalisé par :

Hajar GOUMARIR

Lamiss BAHASSAN

Chahd BOUKHRAISS

Hiba FALIH

Encadré par :

Mr. Hicham BENALLA

Année universitaire *: **2025/2026***

REMERCIEMENTS

Nous tenons à adresser nos plus sincères remerciements à **Mr. Hicham BENALLA**, notre professeur, pour son accompagnement précieux tout au long de la réalisation de ce projet. Son expertise, sa rigueur et son engagement ont été d'une aide inestimable dans la réalisation de ce travail. Ses conseils éclairés et son soutien bienveillant nous ont permis de surmonter les défis et d'avancer avec confiance.

Nos remerciements s'étendent également à l'ensemble des enseignants et au personnel administratif de la Faculté des Sciences et Techniques pour leur soutien continu et pour avoir mis à notre disposition un cadre de travail stimulant.

LISTE DES FIGURES

SOMMAIRE

INTRODUCTION

Dans le cadre du module d'Architecture des Ordinateurs de notre filière LST GI, nous avons eu l'opportunité de réaliser ce projet qui a pour but de concevoir et développer un simulateur logiciel du Micropresseur Motorola 6809 en utilisant le langage de programmation Java. Ce simulateur permet l'émulation du fonctionnement interne du microprocesseur, notamment le cycle d'exécution des instructions, la gestion des registres de la mémoire et des opérations arithmétiques, logiques…

L'objectif est de matérialiser les concepts théoriques d'architecture des ordinateurs étudiés en cours, en mettant en pratiques la structure du microprocesseur Motorola 6809 et son interaction avec la mémoire. Ce projet vise également à renforcer les compétences en programmation bas-niveau et en compréhension d'exécution des instructions.

I. Aperçu sur le Microprocesseur Motorola 6809 :

1. Introduction et contexte historique :

Le **Motorola 6809** est un microproceseur développé par Motorola en **1979**, comme évolution améliorée du populaire **6800** (1974). Considérée comme l'un des processeurs 8 bits les plus sophistiqués de son époque, il présente un architecture avancée en conservant la compatibilité partielle avec le 6800. Il a été largement utilisé dans les systèmes embarqués, des ordinateurs personnels (Tandy TRS-80 Color Computer et le Dragon 32/64) et les équipements industriels.

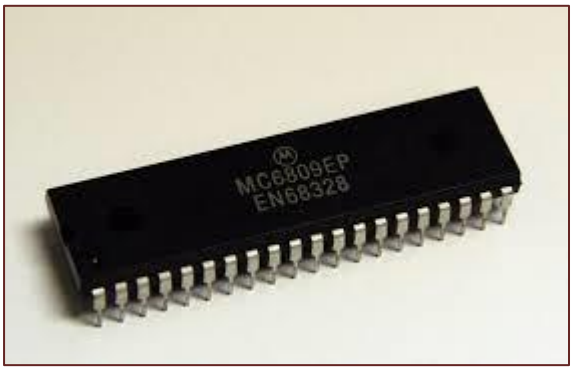


Figure 1: Motorola 6809

2. Caractéristiques techniques :

Le Motorola 6809 se distingue par ses caractéristiques avancées et ses innovations architecturales :

- ***Jeu d'instructions sophistiqué :*** Le 6809 possédait un ensemble riche des instructions, ce qui signifie qu'une grande majorité des instructions pouvait être utilisée avec la plupart des modes d'adressage. Cela facilitait le travail du programmeur et rendait l'assembleur plus intuitif.

- ***Architecture à double accumulateur :*** Il intégrait deux accumulateurs 8 bits (A et B), pouvant être fusionné en un unique registre 16 bits (D). Cette approche offrait une flexibilité accrue pour les calculs arithmétiques et logiques.

- ***Diversité des modes d'adressage :*** Le 6809 proposait une large gamme de modes d'adressage, dont des modes indexés avancés qui était rares parmi les microprocesseurs contemporains.

- ***Conception optimisée :*** Comparé à ses prédécesseurs, le 6809 exécutait les instructions en un nombre réduit de cycles d'horloge, ce qui faisait un processeur à la fois rapide et efficace en termes de consommation de cycles.

3. Quelle est la nécessité d'un simulateur de Motorola 6809 ?

Le développement d'un simulateur du microprocesseur Motorola 6809 répond à plusieurs besoins essentiels :

- D'un point de vue pratique, le simulateur offre une plateforme d'expérimentation accessible et sans risque. Il élimine le besoin de matériel physique tout en permettant l'exécution des programmes en langage assembleur.

- Sur le plan éducatif et pédagogique, le simulateur permet de matérialiser des concepts théoriques souvent abstraits. Les utilisateurs de ce simulateur peuvent observer concrètement le cycle d'exécution des instructions, la gestion des registres, et les échanges entre le microprocesseur et la mémoire. Cette visualisation facilite la compréhension des mécanismes fondamentaux qui régissent tout système informatique.

- Un simulateur du Motorola 6809 est nécessaire pour la préservation des logiciels classiques et leur fonctionnement sur les systèmes modernes. Il joue également un rôle crucial dans la compréhension historique et l'exploration de l'architecture et la programmation des premiers microprocesseurs.

II. Introduction au projet :

1. Méthodologie de recherche :

Une recherche approfondie sur les spécifications techniques du Motorola 6809 a été effectué avant le développement de simulateur pour comprendre en détail l'architecture du microprocesseur : son jeu d'instructions complet, l'organisation de ses registres internes, les modes d'adressage disponibles, et le format binaire des opcodes.

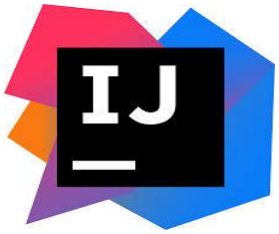
2. Objectifs principaux :

- Simulation fidèle du microprocesseur Motorola 6809.
- Emulation des instructions du microprocesseur 6809 avec la gestion de la mémoire et des registres.
- Offrir une interface graphique pour la visualisation et l'interaction avec la simulation.

3. Technologies employées :

Nous avons choisi le langage **Java** comme langage de programmation pour sa portabilité, la richesse de ses bibliothèques standard qui facilite le développement des interfaces graphiques intuitives, et son approche orientée objet qui correspond naturellement a la modélisation des composants du microprocesseur Motorola 6809 en favorisant une architecture modulaire et extensible.

Nous avons choisi **Java Swing** comme bibliothèque d'interface graphique pour sa simplicité, robustesse et intégration native avec Java. Swing permet de développer une interface fonctionnelle et claire adaptes aux besoins fonctionnelles du projet.



Le choix d'**IntelliJ IDEA** comme environnent de développement pour ce projet s'impose pour son support exceptionnel du langage Java, son interface claire et son système de gestion de projets.

GIMP 3 offre des fonctionnalités avancées de création et retouche d'images qui nous ont permis de concevoir des éléments visuels professionnels.

Le choix de **XMind** comme outil de conception pour notre projet de a été stratégique pour structurer notre réflexion et organiser la complexité du développement. Cette application de mind-mapping nous a permis de visualiser clairement l'architecture globale du simulateur.

4. Les besoins fonctionnels :

- Simuler fidèlement le comportement du microprocesseur Motorola 6809.
- Fournir un éditeur qui permet de saisir et modifier du code assembleur.
- Assembler le code source en code machine 6809.
- Permettre l'exécution du code assemblée.

- Afficher le contenu des mémoires ROM et RAM.
- Afficher l’état interne du CPU.
- Fournir une console pour afficher les messages.
- Supporter les instructions du Motorola 6809.

III. Architecture Technique :

1. Les classes principales :

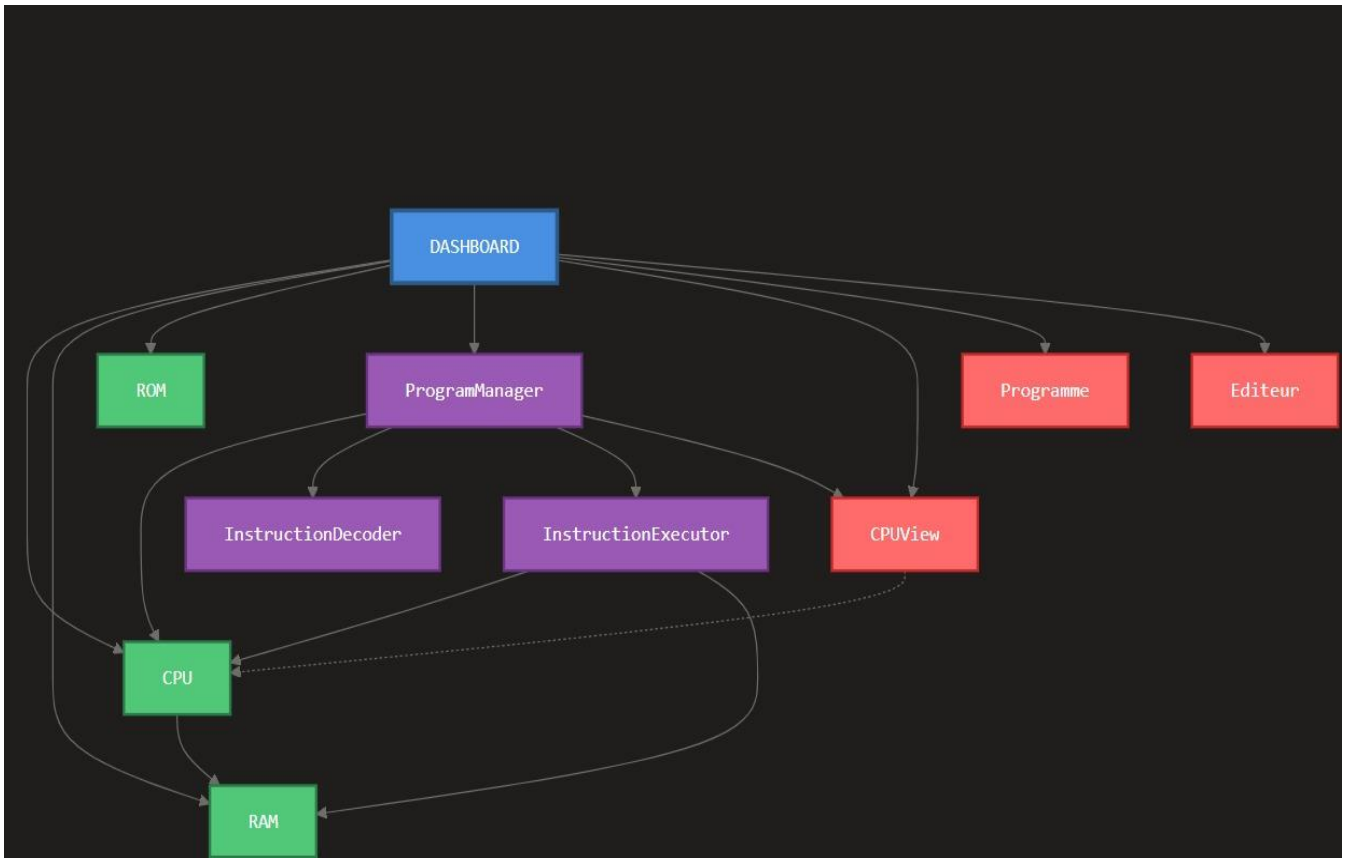


Figure 2: Les classes principales (réalisé avec xmind)

Le simulateur Motorola 6809 est construit selon une architecture **MVC (Modèle-Vue-Contrôleur)** où la classe DASHBORD agit comme le contrôleur et l'interface utilisateur principale et englobante. Cette architecture est organisée en trois couches principales : interface utilisateur, logique de traitement, et les modèles du processeur et de la mémoire.

2. La classe ROM :

La classe ROM stocke le programme assemblé.

- Elle combine à la fois un modèle de données pour le stockage du code machine assemblé et une vue graphique pour sa visualisation en temps réel. Cette classe hérite de *JFrame* et utilise un *DefaultTableModel* pour afficher le contenu de la mémoire sous forme de tableau interactif.

Plage mémoire :

- **Début :** 0xFC00
- **Fin :** 0xFFF
- **Taille :** 1024
- **Valeur par défaut :** FF

Fonctionnalités clés :

- **Initialisation de la ROM.**
- **Ecriture en ROM.**
- **Lecture de de la ROM.**
- **Réinitialisation.**

3. La classe RAM :

La classe RAM stocke les données temporaires et la pile (zone mémoire lecture/écriture). Elle combine à la fois la gestion des données mémoire et leur affichage dans l'interface utilisateur.

Plage mémoire :

- **Début :** 0x0000
- **Fin :** 0x03FF

- **Taille :** 1024
- **Valeur par défaut :** 00

Fonctionnalités clés :

- **Initialisation de la RAM.**
- **Ecriture en RAM.**
- **Lecture de la RAM.**
- **Edition directe dans le tableau.**

4. La classe CPU :

La classe CPU modélise le Microprocesseur Motorola 6809 avec tous ses registres internes et son unité arithmétique et logique (UAL), Elle gère tous les registres et drapeaux du processeur.

Fonctionnalités clés :

- **Gestion des registres.**
- **Gestion des drapeaux.**
- **Conversion Hexadécimal a décimal et vice versa.**

5. La classe InstructionDecoder :

La classe InstructionDecoder décode une ligne d'assembleur 6809 en structure exploitable, identifie le mode d'adressage et calcule les postbytes pour le mode indexé.

Système de modes d'adressage :

```
public enum AddressingMode {
IMMEDIATE DIRECT, EXTENDED, EXTENDED_INDIRECT, INDEXED, INHERENT }
```

Algorithme de décodage :

- **Nettoyage :** supprimer les commentaires et les espaces.
- **Séparation :** Mnémonique + opérande.
- **Identification :** Détecter le mode d'adressage.
- **Parsing :** Extraire les détails spécifiques.

Détection des modes :

```
if (operandRaw.startsWith("#")) {
    // Mode immédiat
    return new DecodedInstruction(mnemonic, AddressingMode.IMMEDIATE, cleanOperand);
}
if (operandRaw.startsWith("<")) {
    // Page directe forcée
    DirectDetails details = new DirectDetails(cleanOperand, true);
    return new DecodedInstruction(mnemonic, AddressingMode.DIRECT, cleanOperand, null, details);
}
if (operandRaw.startsWith(">")) {
    // Étendu forcé
    return new DecodedInstruction(mnemonic, AddressingMode.EXTENDED, cleanOperand);
}
if (operandRaw.contains(",")) {
    // Mode indexé
    return decodeIndexed(mnemonic, operandRaw);
}
```

6. La classe InstructionExecutor :

La classe InstructionExecutor exécute les instructions décodées en interagissant avec la CPU et la mémoire. C'est le moteur d'exécution du simulateur.

- Elle exécute les instructions sur le CPU et génère le code machine dans la ROM.

Catégories d'instructions implémentées :

- Instructions de chargement.
- Instructions de stockage.
- Instructions arithmétiques.
- Instructions logiques.
- Instructions *push* et *pull*.
- Instructions de rotation/décalage, test, transfert.

Algorithme principal de l'exécution :

- Extraire mnémonique, mode, opérande.
- SELON mnémonique :

  Cas 'LDA' : execLDA (mode, opérande)

  Cas 'STA' : execSTA (mode, opérande)

  …….
- Si instruction inconnue → ERREUR,

7. La classe ProgramManager :

La classe ProgrammeManager gère le cycle de vie du programme : chargement, assemblage, exécution pas-a-pas, exécution continue, arrêt.

Cycle de vie du programme :

- **Chargement :** Nettoyage / Validation / END automatique /Initialisation.
- **Assemblage**.
- **Exécution.**
- **Gestion d'états.**

8. La classe Programme :

La classe Programme la fenêtre de visualisation du programme assembleur après assemblage. Elle montre les instructions avec leurs adresses mémoire et met en évidence l'instruction courante pendant l'exécution.

Fonctionnalités clés :

- **Ajout d'instructions.**
- **Mise en évidence.**
- **Chargement depuis ProgramManager.**

9. La classe Editeur :

La classe Editeur joue le rôle d'un éditeur de texte pour le code assembleur. Elle offre deux fenêtres distinctes mais liées pour travailler sur le code source.

Fonctionnalités clés :

- **Edition basique.**
- **Chargement des fichiers.**
- **Chargement par chemin/URL.**

10. La classe Dashbord :

La classe Dashbord représente l'interface graphique principale du simulateur contrôles (Assembler, Exécuter, Pas à Pas, Reset), éditeur de code, et visualisation.

Composants :

- **Panneau central.**
- **Barre de menus (fichier, visualisation, aide).**
- **Boutons de contrôle (assembler, exécuter, Pas à pas, Reset).**
- **Console système (Journalisation et messages de guide).**

11. La classe CPUView :

La classe CPUView représente l'interface graphique qui affiche l'architecture interne du CPU en temps réel (registres, flags, instruction courante), Elle suit le pattern MVC (Modèle-Vue-Contrôleur) où CPU est le modèle et CPUView est la vue.

Fonctionnalités clés :

- **Synchronisation avec le modèle CPU.**
- **Gestion de l'instruction courante.**
- **Réinitialisation de l'affichage.**

IV. Interface Graphique :

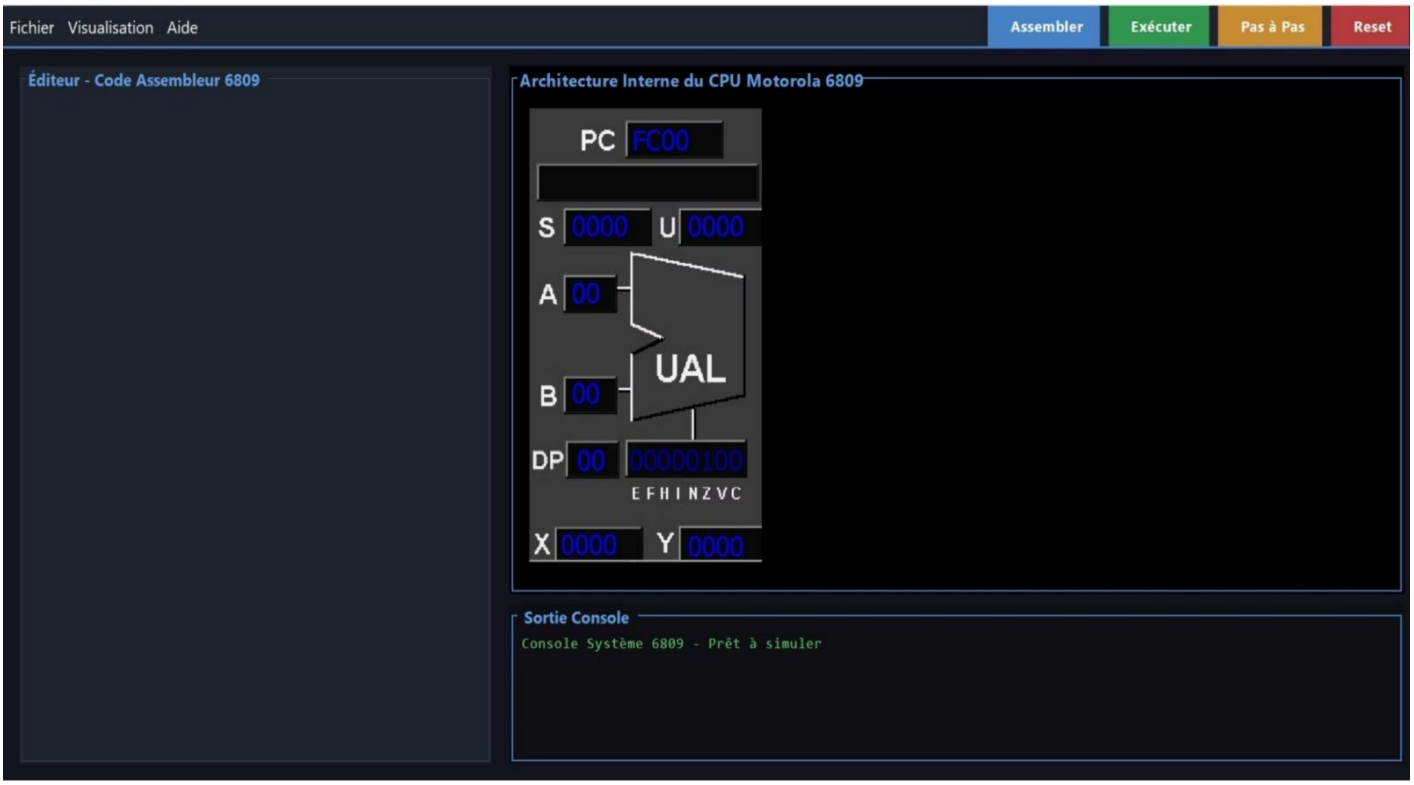


Figure 3 :L'interface principale

Figure 4 : :L'interface ROM

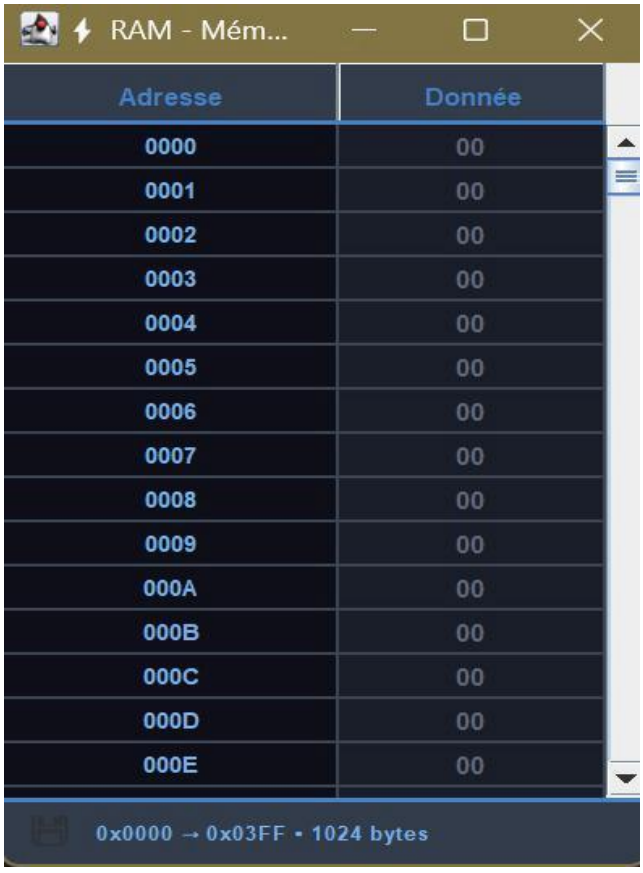


Figure 5 :L'interface RAM

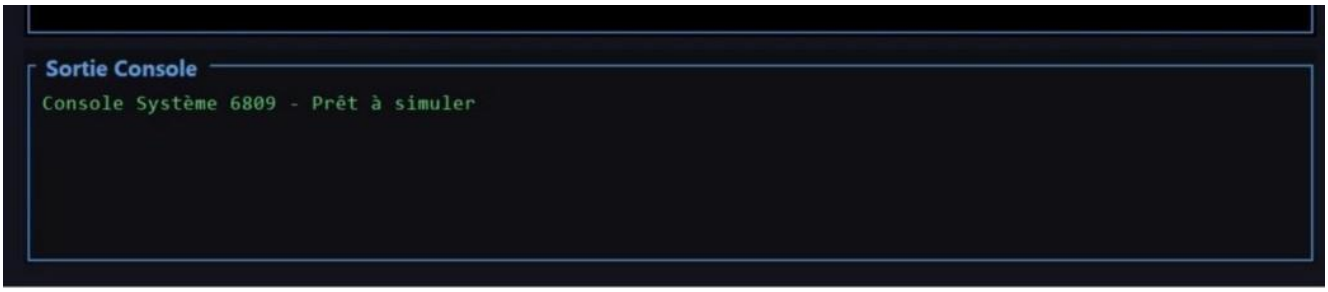


Figure 5 : La console

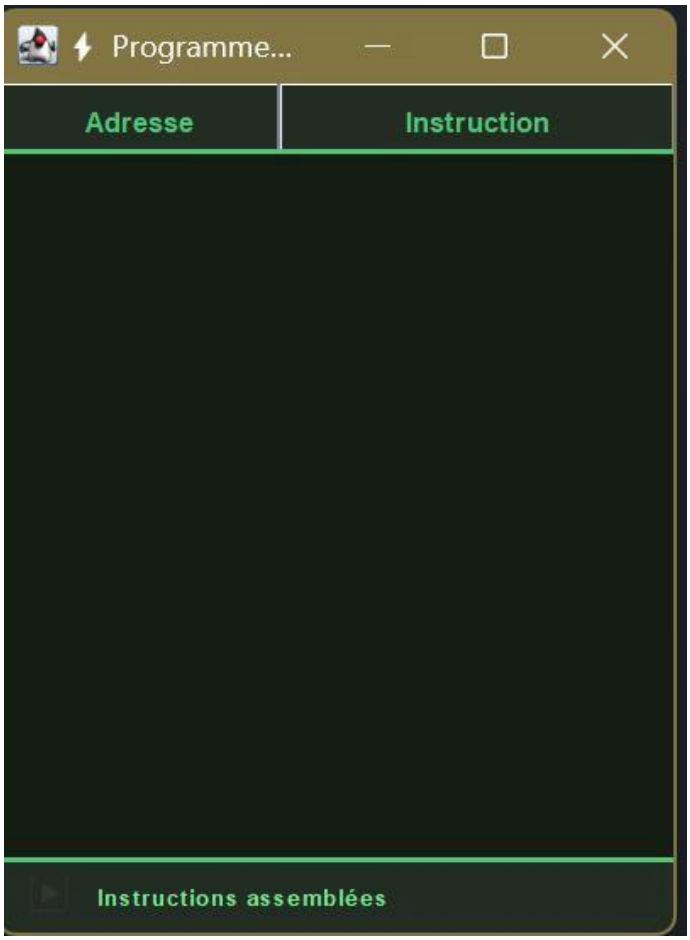


Figure 7 : Interface Programme

Figure 8 : Gestion des fichiers dans Das

CONCLUSION

En conclusion, ce projet de simulateur du processeur Motorola 6809 a représenté une opportunité exceptionnelle de synthèse et d'application concrète de nos connaissances en architecture des ordinateurs, programmation orientée objet et conception d'interfaces. À travers le développement complet de cet émulateur fonctionnel, nous avons approfondi notre compréhension des mécanismes internes d'un microprocesseur, depuis l'unité de décodage d'instructions jusqu'à l'exécution des opérations plus complexes.

La difficulté principale rencontrée pendant la réalisation de ce projet était liée à l'implémentation puisque certaines instructions du Motorola 6809 demandent une compréhension approfondie et une analyse bien structurée.

BIBLIOGRAPHIE

HTTPS://GITHUB.COM/

HTTPS://WWW.YOUTUBE.COM/

HTTPS://WAYTOLEARNX.COM/

HTTPS://WWW.OPENCLASSROOMS.COM/

Livre: MC6809-MC6809E MICROPROCESSOR PROGRAMMING MANUAL, 1981.