UNIVERSITÉ HASSAN 1er

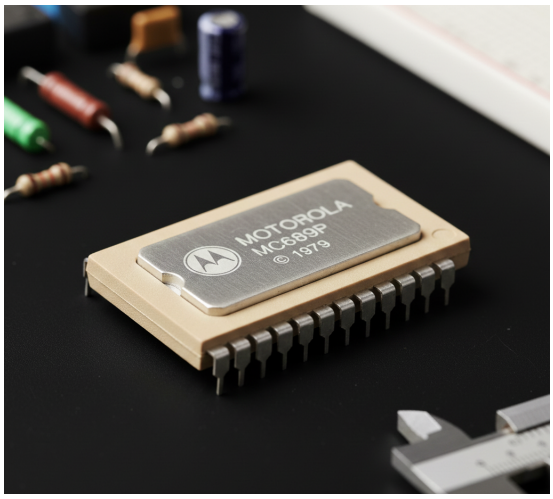
FACULTÉ DES SCIENCES ET TECHNIQUES DE SETTAT

Département : Génie Informatique (LST-GI)

Année universitaire : 2025 – 2026

Simulateur de Microprocesseur Motorola 6809

Algorithme de compilation et Guide d'utilisation



Encadrant :
Prof. Benalla Hicham

Réalisé par :
Chritt Fadila
Barbay Hafsa

---

21 décembre 2025

TABLE DES MATIÈRES

1 Introduction

1.1 De la naissance des ordinateurs aux microprocesseurs

L'histoire de l'informatique moderne débute bien avant l'apparition des microprocesseurs. Les premiers ordinateurs électroniques voient le jour dans les années 1940, dans un contexte fortement marqué par la Seconde Guerre mondiale. Des machines comme ENIAC (1945) ou Colossus ont été développées principalement pour des objectifs militaires : calcul balistique, cryptanalyse et traitement rapide de données stratégiques. Ces premiers ordinateurs étaient volumineux, coûteux, peu fiables et composés de milliers de tubes à vide.

Après la guerre, les avancées technologiques se poursuivent rapidement. L'invention du transistor en 1947 par les laboratoires Bell constitue une révolution majeure. Elle permet de remplacer les tubes à vide par des composants plus petits, plus rapides et plus fiables. Cette évolution ouvre la voie à une miniaturisation progressive des systèmes informatiques.

1.2 Guerre froide et accélération technologique

La Guerre froide (années 1950–1970) joue un rôle déterminant dans le développement de l'informatique et de l'électronique. La compétition technologique entre les grandes puissances, notamment les États-Unis et l'Union soviétique, stimule massivement la recherche dans des domaines tels que :

— les systèmes de défense,
— le calcul scientifique,
— les communications,
— l'exploration spatiale.

Dans ce contexte, la nécessité de systèmes de calcul plus rapides, plus compacts et embarquables devient cruciale. Les projets militaires et spatiaux exigent des ordinateurs capables de fonctionner dans des environnements contraints (avions, missiles, satellites), ce qui accélère le développement des circuits intégrés dans les années 1960.

1.3 De l'ordinateur au microprocesseur

L'étape décisive est franchie au début des années 1970 avec l'apparition du microprocesseur, qui regroupe sur une seule puce l'unité centrale de traitement (CPU). Le premier microprocesseur commercial, l'Intel 4004 (1971), marque un tournant historique : il devient possible de concevoir des ordinateurs beaucoup plus compacts, abordables et accessibles.

Les microprocesseurs permettent alors l'émergence :

— des micro-ordinateurs,

— des systèmes embarqués,

— de nouvelles applications industrielles et éducatives.

Cette démocratisation de l'informatique n'est donc pas uniquement le fruit de besoins civils, mais résulte directement des avancées issues des contextes militaires, industriels et scientifiques de la période précédente.

1.4 Le microprocesseur Motorola 6809

C'est dans cette continuité historique qu'apparaît, en 1978, le Motorola 6809. Considéré comme l'un des microprocesseurs 8 bits les plus avancés de son époque, il se distingue par :

— une architecture interne élégante et puissante,

— un jeu d'instructions riche,

— des modes d'adressage innovants,

— une séparation claire entre données et instructions.

Le Motorola 6809 a été utilisé dans de nombreux systèmes, notamment des ordinateurs personnels, des consoles de jeux et des applications industrielles. Bien qu'il ne soit pas directement conçu pour un usage militaire, il est l'héritier direct des avancées technologiques issues des décennies précédentes, fortement influencées par les besoins stratégiques et scientifiques.

1.5 Objectif du projet

L'objectif principal de ce projet est de concevoir et développer un simulateur logiciel du microprocesseur Motorola 6809, permettant de reproduire fidèlement son fonctionnement interne, l'exécution des instructions et l'interaction avec la mémoire et les périphériques, afin de faciliter l'apprentissage de l'architecture des microprocesseurs et de la programmation bas niveau dans un environnement pédagogique interactif.

2 Présentation générale du simulateur

Le simulateur Motorola 6809 développé dans le cadre de ce projet est une application logicielle permettant d'émuler le comportement d'un microprocesseur réel sans nécessiter de matériel physique. Il offre une représentation fidèle des principaux composants du processeur ainsi qu'un ensemble d'outils facilitant l'observation et l'analyse de l'exécution des programmes.

Le simulateur est destiné principalement à un usage pédagogique. Il permet aux étudiants et aux utilisateurs intéressés par l'architecture des ordinateurs de comprendre concrètement le fonctionnement interne d'un microprocesseur historique, tout en interagissant avec celui-ci de manière visuelle et intuitive.

2.1 Fonctionnalités principales

Le simulateur Motorola 6809 propose les fonctionnalités suivantes :

- Exécution de programmes en langage machine du Motorola 6809.
- Exécution pas à pas des instructions afin de suivre précisément l'évolution de l'état du processeur.
- Affichage en temps réel des registres du processeur (accumulateurs, registres d'index, compteur ordinal, pointeur de pile, etc.).
- Visualisation du contenu de la mémoire et de la pile.
- Affichage du désassemblage des instructions en cours d'exécution.
- Gestion des points d'arrêt (breakpoints) pour le débogage.
- Simulation d'interruptions matérielles et de périphériques simples, comme le clavier.
- Interface graphique conviviale facilitant l'interaction avec le simulateur.

Grâce à ces fonctionnalités, le simulateur constitue un outil complet permettant d'analyser, de tester et de comprendre le fonctionnement du microprocesseur Motorola 6809 dans un environnement contrôlé et sécurisé.

2.2 Contexte du projet

Dans un cadre pédagogique, l'étude et la simulation du Motorola 6809 permettent de comprendre en profondeur l'évolution des architectures de processeurs, les principes fondamentaux de l'exécution des instructions et la gestion de la mémoire. Le présent projet s'inscrit dans cette démarche, en proposant un simulateur logiciel du microprocesseur Motorola 6809, destiné à l'apprentissage, à l'expérimentation et au débogage de programmes bas niveau.

3 Environnement de développement

3.1 Langage de programmation : Java

Le simulateur du microprocesseur Motorola 6809 a été développé en Java, un langage de programmation orienté objet, de haut niveau et portable. Java repose sur le principe « Write Once, Run Anywhere », ce qui signifie qu'un programme Java peut être exécuté sur toute plateforme disposant d'une Machine Virtuelle Java (JVM), sans modification du code source.

Java a été choisi pour ce projet pour les raisons suivantes :

— **Portabilité** : le simulateur peut fonctionner sur différents systèmes d'exploitation.

— **Robustesse et sécurité** : Java offre une gestion automatique de la mémoire et un environnement d'exécution sécurisé, adaptés à la simulation d'architectures complexes.

— **Lisibilité et maintenance** : la programmation orientée objet facilite l'organisation du code (CPU, mémoire, périphériques, interface graphique).

— **Écosystème riche** : disponibilité de nombreuses bibliothèques, outils et environnements de développement (JDK, IDE).

Le développement et l'exécution du projet nécessitent le **Java Development Kit (JDK)**, qui inclut le compilateur Java ainsi que la Machine Virtuelle Java (JVM).

3.2 Bibliothèque graphique : Java Swing

L'interface utilisateur du simulateur est réalisée à l'aide de **Java Swing**, une bibliothèque graphique faisant partie des *Java Foundation Classes (JFC)*. Swing permet la création d'interfaces graphiques riches et interactives pour les applications Java.

Dans ce projet, Swing est utilisé pour :

— l'affichage des registres du processeur,

— la visualisation de la mémoire,

— l'affichage de la console d'exécution,

— les commandes de contrôle (STEP, RUN, RESET, IRQ, ASM Editor).

Swing offre également une grande flexibilité de personnalisation, ce qui permet d'adap-

ter l'interface graphique aux besoins spécifiques d'un simulateur pédagogique.

3.3 Plateforme d'exécution

Le développement et les tests du simulateur ont été réalisés sur le système d'exploitation **Windows**, qui constitue la plateforme utilisée par l'auteur du projet.

Cependant, grâce à l'utilisation du langage Java et de la **Machine Virtuelle Java (JVM)**, le simulateur reste théoriquement compatible avec d'autres systèmes d'exploitation tels que **Linux** et **macOS**, sous réserve de disposer d'un **JDK** compatible.

3.4 Outils utilisés

Le projet a été développé à l'aide :

— du **JDK (Java Development Kit)**,

— d'un environnement de développement intégré (IDE) tel que **Eclipse** ou **IntelliJ IDEA**,

— des bibliothèques standards fournies par Java (**Swing**, **AWT**).

4 Algorithme de compilation

Dans ce projet, l'algorithme de compilation repose sur le processus standard du langage Java, qui permet de transformer le code source en un format intermédiaire exploitable par la Machine Virtuelle Java (JVM). Contrairement aux langages compilés directement en code machine, Java utilise une étape intermédiaire appelée *bytecode*, garantissant ainsi la portabilité du programme sur différents systèmes d'exploitation.

Le simulateur Motorola 6809 est organisé en plusieurs *packages*, notamment `cpu`, `gui` et `device`, chacun regroupant des classes ayant un rôle précis. Le package `cpu` contient le cœur du processeur simulé (registres, instructions, exécution), le package `gui` gère l'interface graphique, tandis que le package `device` simule les périphériques matériels tels que le clavier.

Lors de la compilation, chaque fichier source portant l'extension `.java` est analysé par le compilateur `javac`. Celui-ci vérifie la syntaxe du code et le traduit en fichiers `.class` contenant du bytecode. Ce bytecode n'est pas directement lisible par le système, mais il est interprété et exécuté par la JVM.

Une fois la compilation terminée sans erreurs, la JVM se charge du chargement des classes compilées, de l'initialisation de l'application et du lancement du simulateur. Dans un environnement de développement intégré (IDE) tel qu'Eclipse ou IntelliJ IDEA, ces étapes sont automatisées, ce qui simplifie le développement, le débogage et l'évolution du projet.

Schéma simplifié du processus de compilation :

Fichier source (`.java`)
↓ Compilation (`javac`)
Fichier bytecode (`.class`)
↓ Chargement par la JVM

5 Guide d'utilisation détaillé

Ce chapitre décrit les différentes étapes permettant à l'utilisateur de lancer, configurer et utiliser efficacement le simulateur du microprocesseur Motorola 6809.

5.1 Lancement du simulateur

Le simulateur est lancé à partir de la classe principale `Launcher`. Après l'exécution, l'interface graphique principale s'affiche automatiquement.

L'utilisateur peut alors interagir avec le processeur simulé à l'aide des différents panneaux et boutons de contrôle.

FIGURE 5.1 – Lanceur de simulator

5.2 Présentation de l'interface graphique

L'interface graphique est organisée de manière à offrir une vision claire et structurée de l'état interne du processeur. Elle est composée des éléments suivants :

- Une barre de contrôle supérieure contenant les boutons d'exécution
- Un panneau des registres
- Un panneau d'affichage de la mémoire
- Un panneau de désassemblage
- Une console de sortie
- Un panneau de gestion des breakpoints

— Un panneau d'affichage de la pile (stack)

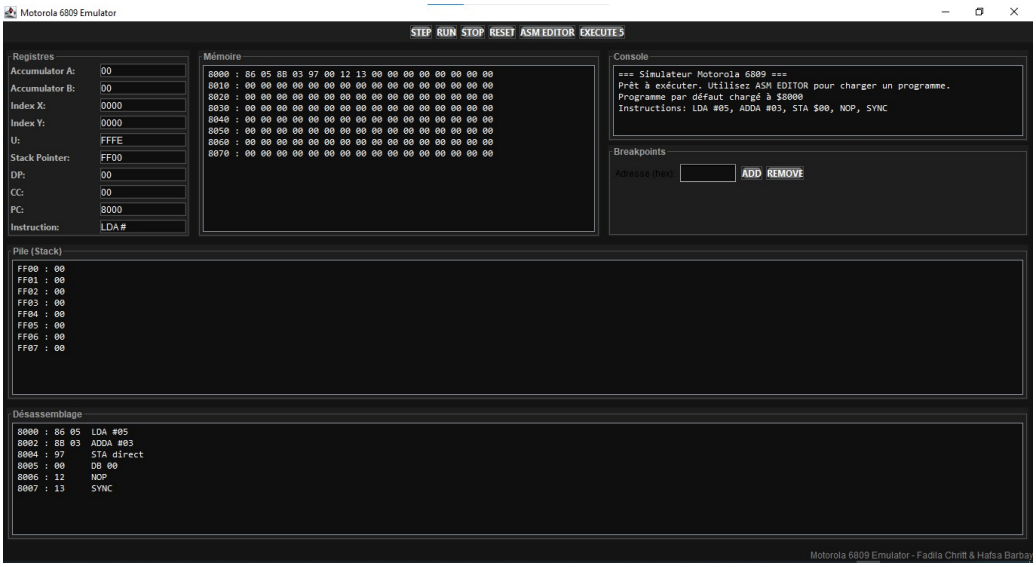


FIGURE 5.2 – Affichage du l interface du simulateur Motorola 6809

5.3 Barre de contrôle supérieure

La barre de contrôle supérieure permet à l'utilisateur de piloter l'exécution du processeur simulé. Elle regroupe les principaux boutons d'action décrits dans le tableau ci-dessous.

Bouton	**Description**
STEP	Exécute une seule instruction (mode pas à pas)
RUN	Lance l'exécution continue (environ 200 ms entre chaque instruction)
STOP	Arrête l'exécution continue en cours
RESET	Réinitialise complètement le simulateur (CPU, mémoire et état interne)
ASM EDITOR	Ouvre l'éditeur assembleur intégré
EXECUTE 5	Exécute cinq instructions consécutives puis s'arrête

Remarque concernant le bouton EXECUTE 5 : Ce bouton exécute exactement cinq instructions successives avant de s'arrêter automatiquement. Il constitue un compromis efficace entre le contrôle précis du mode **STEP** et l'exécution continue du mode **RUN**.

5.4 Panneau des registres

Le panneau des registres affiche en temps réel l'état interne du microprocesseur Motorola 6809. Les registres sont des mémoires internes très rapides utilisées par l'unité arithmétique et logique (UAL) pour le traitement des données.

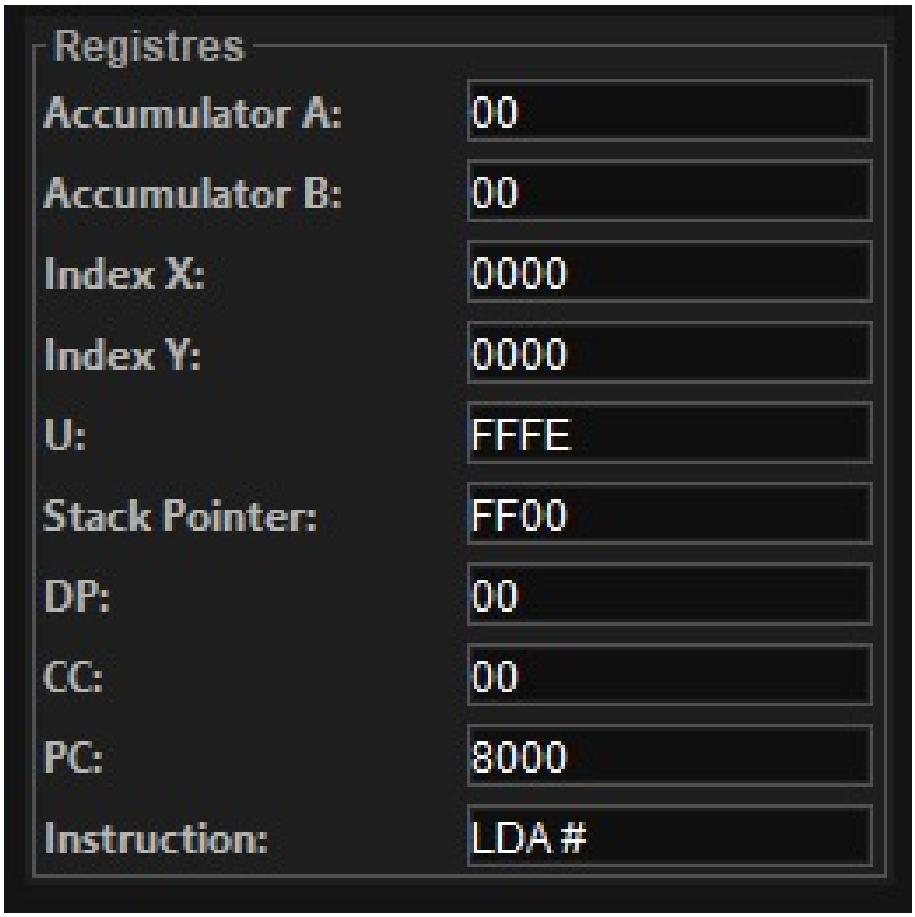


FIGURE 5.3 – Affichage du panneau des registres du simulateur Motorola 6809

Registre	**Description**
Accumulator A / B	Registres d'accumulation 8 bits. Ce sont des mémoires internes rapides permettant à l'UAL de manipuler les données à grande vitesse.
Index X / Y	Registres d'index 16 bits. Ce sont deux pointeurs identiques utilisés principalement dans le mode d'adressage indexé.
U	Pointeur de pile utilisateur. Il facilite le passage des paramètres entre les programmes et les sous-programmes.
Stack Pointer (SP)	Pointeur de pile système. Il est utilisé automatiquement par le microprocesseur pour sauvegarder l'état des registres lors des appels de sous-programmes ou des interruptions.
DP	Direct Page Register. Il permet de définir la page mémoire active utilisée dans le mode d'adressage direct.
CC	Condition Code Register. Il contient les indicateurs (flags : C, AC, S, OV, Z, P) utilisés pour les tests, les branchements conditionnels et l'analyse des résultats.
PC	Program Counter. Il contient l'adresse de la prochaine instruction à exécuter.
Instruction	Indique le nom de l'instruction actuellement en cours d'exécution.

5.5 Panneau Mémoire

Le panneau **Mémoire** affiche le contenu de la mémoire du simulateur en **hexadécimal**.

Chaque ligne de ce panneau représente :

— une **adresse mémoire** située à gauche,

— suivie de **16 octets** affichés en format hexadécimal.

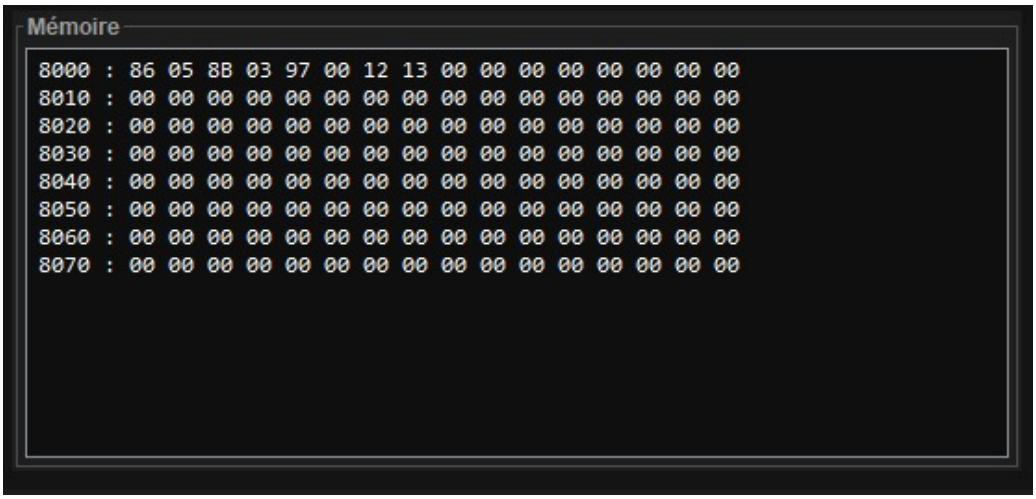


FIGURE 5.4 – Affichage du panneau Mémoire du simulateur Motorola 6809

Rôle du panneau Mémoire

Ce panneau permet de :

— **Visualiser le code machine chargé en mémoire**

— **Vérifier les données stockées**

— **Observer les écritures mémoire en temps réel**

5.6 Console de sortie

La console de sortie est un espace d'affichage qui permet de suivre le fonctionnement du simulateur en temps réel. Elle informe l'utilisateur sur les différentes étapes de l'exécution et sur l'état général du simulateur.

À travers la console, l'utilisateur peut observer :

— Le message de démarrage du simulateur

— Le chargement automatique ou manuel d'un programme

— Les instructions exécutées pas à pas

— La valeur du compteur ordinal (PC) à chaque étape

La console joue un rôle important lors de l'exécution pas à pas, car elle permet de comprendre le comportement du programme sans analyser directement la mémoire ou les registres.

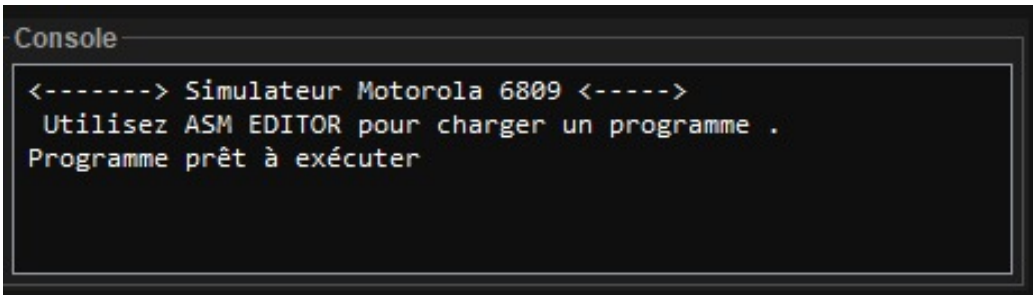


FIGURE 5.5 – Console de sortie du simulateur Motorola 6809

Grâce à cette console, l'utilisateur peut suivre clairement le déroulement de l'exécution et identifier rapidement les différentes actions effectuées par le simulateur.

5.7 Utilisation de l'éditeur assembleur (ASM EDITOR)

L'éditeur assembleur intégré permet d'écrire, d'assembler et de charger directement un programme assembleur Motorola 6809 dans le simulateur, sans utiliser de fichiers externes. Il est principalement utilisé pour tester rapidement des instructions et observer leur impact sur l'exécution.

Les étapes d'utilisation sont les suivantes :

1. Cliquer sur le bouton **ASM EDITOR** situé dans la barre de contrôle.
2. Une fenêtre d'édition s'ouvre, permettant la saisie et la modification du code assembleur.
3. Cliquer sur le bouton **Assembler**.
4. Le programme est automatiquement assemblé et chargé en mémoire à l'adresse définie.
5. Revenir à l'interface principale du simulateur.
6. Lancer l'exécution à l'aide des boutons **STEP**, **RUN** ou **EXECUTE 5**.

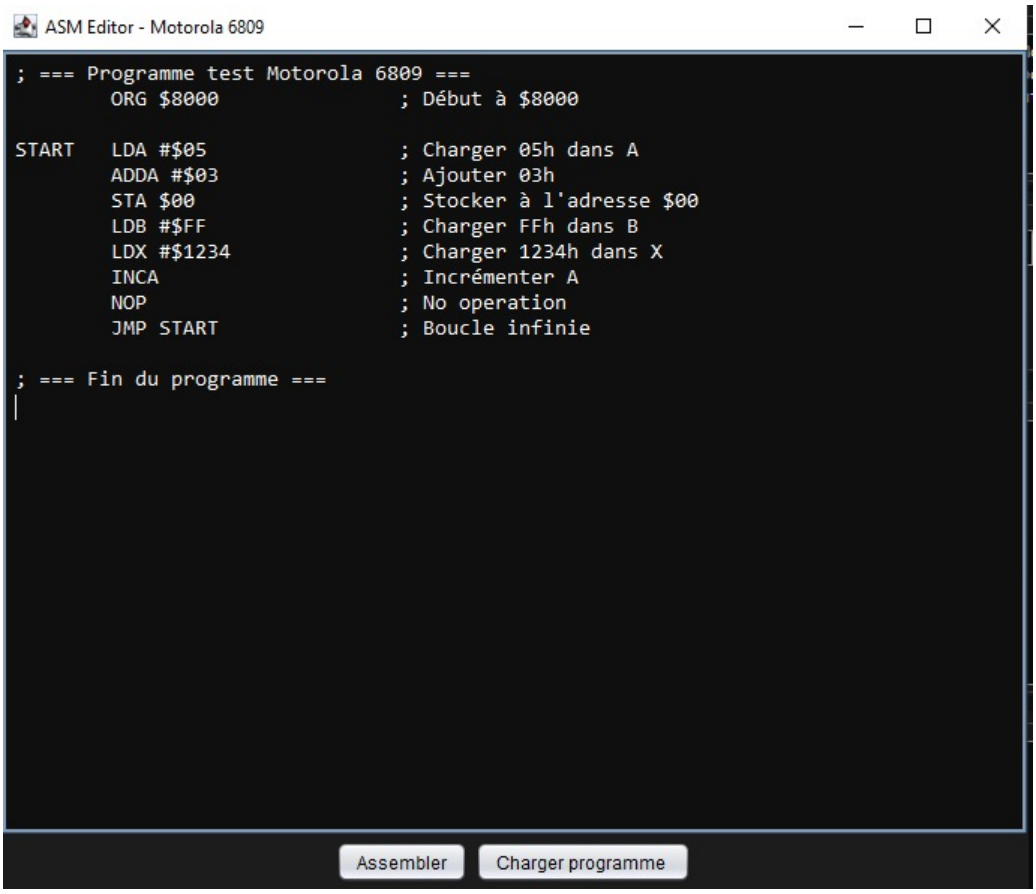


FIGURE 5.6 – Éditeur assembleur (ASM EDITOR) du simulateur Motorola 6809

L'utilisation de l'éditeur assembleur permet de modifier rapidement le code et d'observer immédiatement les effets sur les registres, la mémoire et la console de sortie.

5.8 Gestion des breakpoints

Un breakpoint (point d'arrêt) est une adresse du programme où l'exécution s'arrête automatiquement. Il permet d'observer l'état du processeur à un moment précis sans exécuter tout le programme d'un coup.

Ajouter un breakpoint

Pour ajouter un breakpoint :

1. Entrer une adresse hexadécimale
2. Cliquer sur le bouton **ADD**.

Supprimer un breakpoint Pour supprimer un breakpoint :

1. Sélectionner l'adresse dans la liste.
2. Cliquer sur le bouton **REMOVE**.

Fonctionnement : Lorsque l'exécution atteint un breakpoint :

— Le programme s'arrête automatiquement

— Et Un message apparaît dans la console

— Une clique sur **STEP**ou **RUN** applique le contiennement

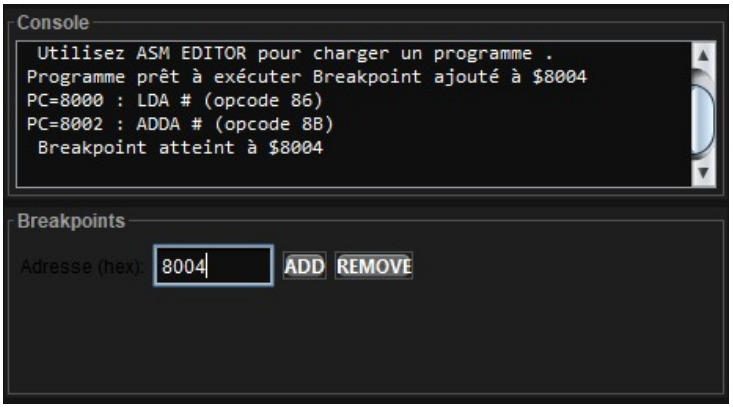


FIGURE 5.7 – Panneau de gestion des breakpoints

5.9 Exécution pas à pas et modes d'exécution

Le simulateur propose plusieurs modes d'exécution permettant à l'utilisateur de contrôler la vitesse et le déroulement du programme selon ses besoins.

Exécution pas à pas (STEP)

Le mode **STEP** permet d'exécuter une seule instruction à la fois.

À chaque clic sur le bouton **STEP** :

— une instruction est exécutée,

— le compteur ordinal (PC) avance,

— les registres et la mémoire sont mis à jour,

— l'instruction exécutée s'affiche dans la console.

Ce mode est particulièrement utile pour analyser le comportement du programme instruction par instruction et comprendre précisément son fonctionnement.

Exécution continue (RUN)

Le bouton **RUN** lance l'exécution du programme de manière automatique et continue.

Dans ce mode :

— les instructions s'enchaînent sans intervention de l'utilisateur,

— l'exécution s'arrête automatiquement si le CPU atteint l'instruction `SYNC`,

— l'utilisateur peut interrompre l'exécution à tout moment en cliquant sur **STOP**.

Ce mode est utilisé pour exécuter rapidement un programme complet.

Exécution intermédiaire (EXECUTE 5)

Le bouton **EXECUTE 5** exécute exactement cinq instructions consécutives, puis s'arrête automatiquement.

Ce mode représente un compromis entre :

— la précision du mode **STEP**,

— et la rapidité du mode **RUN**.

Il permet d'avancer plus vite dans le programme tout en gardant un certain contrôle sur l'exécution.