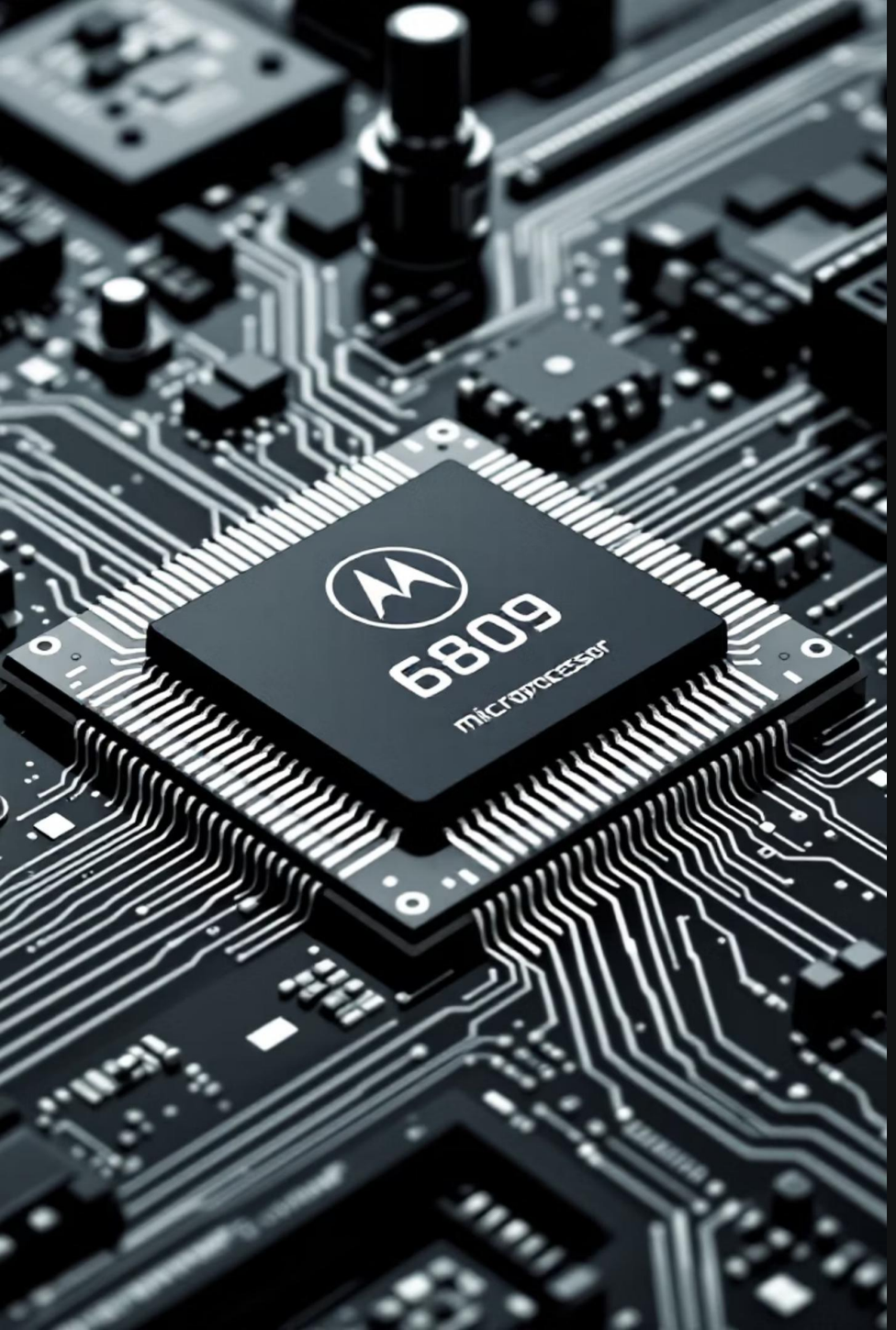


Rapport de Projet – Simulateur de Microprocesseur Motorola 6809

I. Contexte et Objectifs du Projet

*Le projet **SIMULATEUR DE MICROPROCESSEUR MOTOROLA 6809** vise à développer un simulateur logiciel du microprocesseur Motorola 6809 en Java, utilisant le logiciel IntelliJ IDEA. Il reproduit le fonctionnement interne du processeur, de sa mémoire (RAM/ROM) et de son jeu d'instructions, avec une interface de type tableau de bord et un éditeur pour le code assembleur.*

Comprendre l'architecture

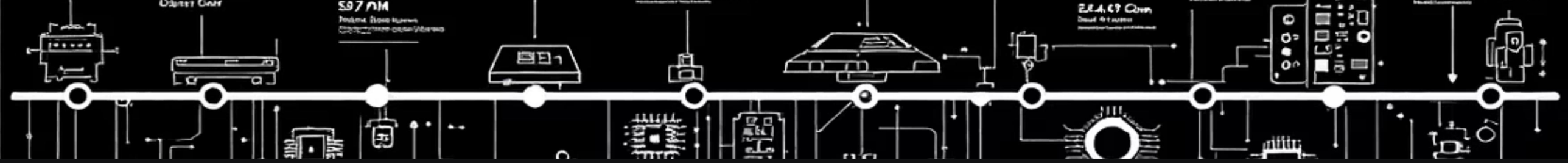
Visualiser les registres, la mémoire et le flux d'instructions du 6809.

Écrire et exécuter du code

Assembler et exécuter du code assembleur 6809 dans un environnement contrôlé.

Expérimenter et déboguer

Tester des programmes bas niveau sans matériel physique.



Histoire du Motorola 6809 – Un Microprocesseur Légendaire

Lancement en 1978

Le Motorola 6809 est sorti sur le marché, une avancée majeure par rapport à son prédécesseur, le 6800.

Conception Avancée

Il fut l'un des premiers microprocesseurs "propres" 8 bits, conçu de manière quasi 16 bits, avec des registres et modes d'adressage optimisés.

Caractéristiques Révolutionnaires

Offrait des performances améliorées grâce à ses registres d'indexation, son accès efficace à la pile et ses instructions puissantes, le rendant plus facile à programmer en assembleur ou en langage C.

Applications Larges

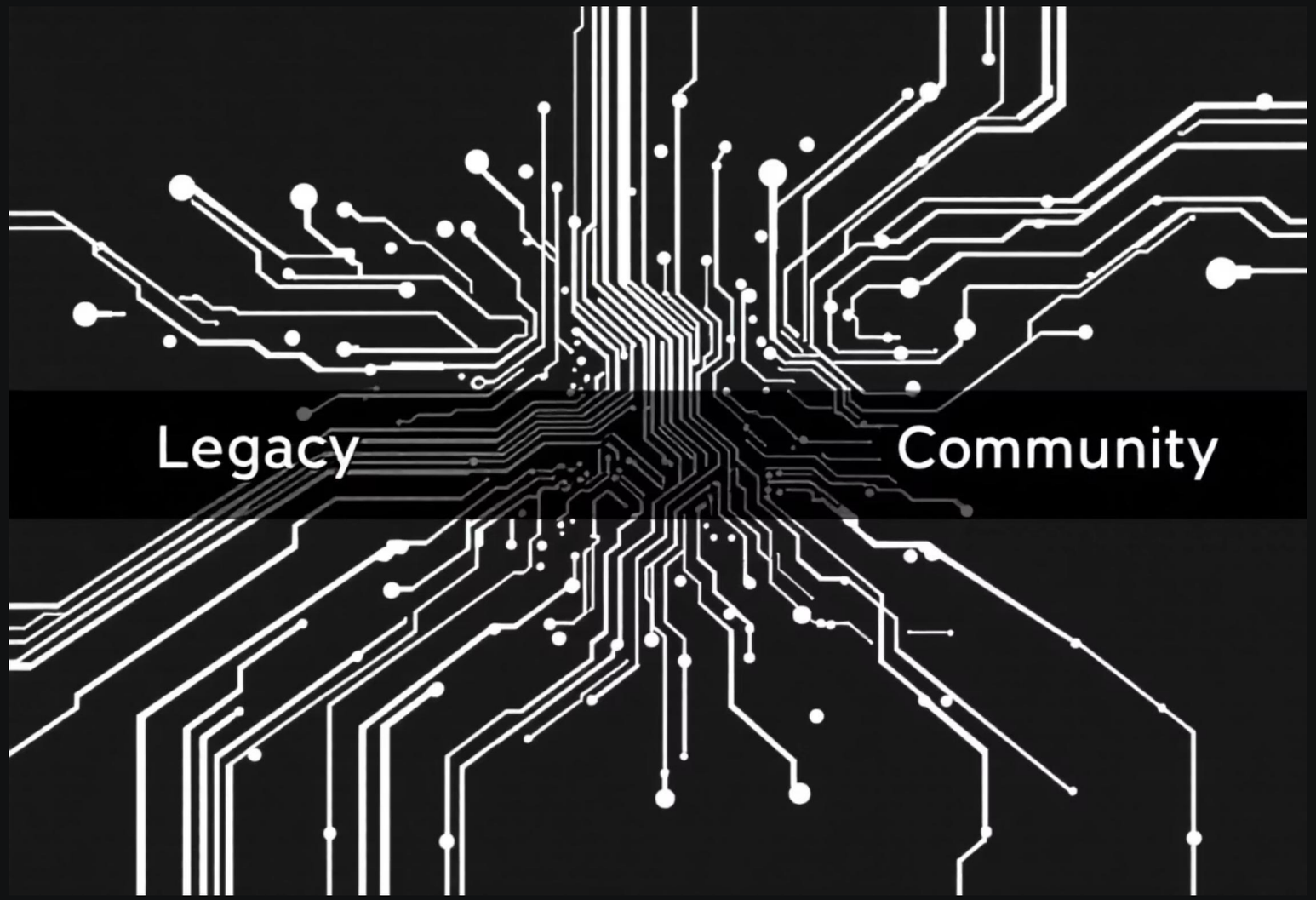
Utilisé dans des ordinateurs personnels emblématiques comme le TRS-80 Color Computer, le Dragon 32/64, et le Vectrex, ainsi que dans de nombreux systèmes embarqués et jeux d'arcade.

Héritage Durable

Son architecture élégante en fait encore aujourd'hui un excellent outil pédagogique pour comprendre les principes fondamentaux des microprocesseurs.

Impact du Motorola 6809 : Industrie, Rétro-Informatique et Patrimoine Numérique

L'héritage du Motorola 6809 va bien au-delà de son époque d'apogée. Ce microprocesseur emblématique continue de façonner notre compréhension de l'informatique, d'inspirer les communautés rétro et de servir d'outil pédagogique essentiel, contribuant ainsi à la préservation de notre patrimoine numérique.



1

Compréhension Historique

Le 6809 a joué un rôle crucial dans l'évolution des microprocesseurs. Ses innovations architecturales, notamment ses registres d'indexation et ses modes d'adressage avancés, ont influencé la conception des générations suivantes. Il a rivalisé avec des géants comme Intel et Zilog, se distinguant par son efficacité et sa facilité de programmation en assembleur ou en C.

2

Développement Rétro-Informatique

Une communauté mondiale continue de faire vivre le 6809. Des passionnés développent de nouveaux jeux, applications et émulateurs, prouvant la flexibilité et la pertinence de cette architecture. Cet engouement pour la rétro-informatique encourage la création et la redécouverte de logiciels pour des plateformes comme le TRS-80 Color Computer ou le Dragon 32/64.

3

Outils Éducatifs

Grâce à sa conception claire et sa complexité gérable, le 6809 reste un excellent choix pour l'enseignement de l'architecture des processeurs, de la programmation en assembleur et des systèmes embarqués. Il offre une base solide pour comprendre les principes fondamentaux, souvent masqués par la complexité des processeurs modernes.

4

Préservation du Patrimoine Numérique

Le 6809 incarne l'importance de préserver et de documenter les technologies passées. Les efforts de conservation incluent les archives numériques de code source, les musées virtuels et physiques, et la transmission des connaissances aux nouvelles générations. Maintenir l'accès à ces technologies est vital pour l'histoire et l'avenir de l'informatique.

VIII. Implémentation en Java – Swing/AWT sur IntelliJ IDEA

La robustesse et la portabilité de Java, combinées aux capacités d'IntelliJ IDEA, offrent un environnement de développement idéal pour notre simulateur 6809. Nous avons opté pour Swing/AWT pour une interface utilisateur native et performante.

Stack Technologique Java

Notre simulateur est construit sur une pile technologique Java moderne et éprouvée, garantissant stabilité et compatibilité.

- ***Langage:*** *Java 11+*
- ***Framework GUI:*** *Swing et AWT*
- ***IDE:*** *IntelliJ IDEA Community Edition*
- ***Outil de Build:*** *Maven*
- ***Tests Unitaires:*** *JUnit 5*

Architecture Swing/AWT

L'interface utilisateur est conçue avec les composants standards de Swing et AWT pour une expérience native et réactive.

- ***Fenêtre principale:*** `JFrame`
- ***Panneaux de contenu:*** `JPanel`
- ***Éditeur de code:*** `JTextArea`
- ***Affichage registres/mémoire:*** `JTable`
- ***Contrôles:*** `JButton`, `JLabel`, `JComboBox`
- ***Gestionnaires de Layout:*** `BorderLayout`, `GridLayout`, `FlowLayout`

Workflow IntelliJ IDEA

IntelliJ IDEA est notre IDE de choix, optimisant la productivité grâce à ses outils intégrés pour le développement Java.

- *Création rapide de projets Maven/Java.*
- *Configuration facile des dépendances (JUnit, etc.).*
- *Organisation logique du code en packages.*
- *Compilation et exécution simplifiées.*
- *Débogage avancé avec points d'arrêt et inspection de variables.*

II. Périmètre, Portée et Parties Prenantes

Le simulateur se concentre sur le modèle de processeur 8 bits Motorola 6809, la gestion de son jeu d'instructions, la simulation de la RAM et ROM, et une interface utilisateur minimale pour l'édition et l'exécution de code assembleur.

Périmètre Clé

- *Modèle processeur 6809 (registres, drapeaux, bus)*
- *Jeu d'instructions (arithmétique, logique, contrôle de flux)*
- *Simulation RAM et ROM*
- *Interface utilisateur (chargement, édition, exécution)*
- *Visualisation de l'état interne*

Parties Prenantes

- *Équipe de développement : CHAHD BOUKHRAISS , HAJAR GOUMARIR*
- *Encadrant pédagogique : PROF BENALLA*

III. Architecture Globale du Simulateur

L'architecture logicielle est modulaire, chaque classe ayant une responsabilité définie pour faciliter la maintenance et l'évolution.

1	**DASHBOARD** *Interface utilisateur, contrôle du simulateur.*
2	**CPU_VIEW** *Modélisation de la structure logique du 6809.*
3	**CPU** *Cœur du processeur, exécution des instructions.*
4	**EDITEUR** *Rédaction et gestion du code assembleur.*
5	**INSTRUCTIONS (INSTRUCTION_DECODER / INSTRUCTION_EXECUTOR)** *Décodage et exécution des opcodes.*
6	**RAM/ROM** *Simulation des mémoires vive et morte.*

IV. Description Détaillée des Composants

IV.1. Classe Main

Initialise l'environnement, crée les instances (CPU, RAM, ROM, Dashboard, Editeur) et lance la boucle de l'application.

IV.2. Classe Dashboard

Interface de contrôle (menu, affichage état CPU/mémoire, commandes utilisateur). Peut être graphique ou console enrichie.

IV.3. Classe CPU_VIEW

Abstrait l'architecture du 6809 : registres, bus (16 bits adresses, 8 bits données), liaison CPU-mémoires, signaux de contrôle.

IV. Description Détaillée des Composants (suite)

IV.4. Classe CPU

Cœur du simulateur : maintient l'état (registres, flags), gère le cycle fetch-decode-execute, accède à la mémoire et met à jour les flags.

Gestion des registres

A, B, D, X, Y, U, S, PC, DP, CCR.

Cycle d'instruction

Fetch, Decode, Execute.

Mise à jour des flags

Z, N, C, V, H.

IV.5. Classe Editeur

Environnement de saisie pour le code assembleur 6809 : édition, ouverture/sauvegarde de fichiers, envoi au simulateur.

IV. Description Détaillée des Composants (suite)

IV.6. Classe Instructions (INSTRUCTION_DECODER / INSTRUCTION_EXECUTOR)

Centralise le jeu d'instructions du 6809 : table de correspondance opcodes/opérations, exécution des instructions (ADD, LDA, JMP, etc.), gestion des modes d'adressage.

IV.7. Classes RAM et ROM

Abstrait les mémoires du système : RAM (modifiable, lecture/écriture sécurisée), ROM (lecture seule, initialisée). Les deux vérifient la validité des adresses et permettent un affichage partiel dans le Dashboard.

RAM

- *Mémoire vive modifiable*
- *Tableau de bytes*
- *Méthodes lecture/écriture sécurisées*

ROM

- *Mémoire en lecture seule*
- *Initialisée par fichier binaire/tableau*
- *Accès lecture seule par le CPU*

V. Cycle de Développement, Planning et Jalons

V.1. Planning Prévisionnel

Analyse et conception	*Semaine 1*	*1*	*Spécifications, modèle d'architecture*
Implémentation noyau CPU + mémoire	*Semaines 2–3*	*2*	*CPU, RAM, ROM, Architecture Interne fonctionnels*
Implémentation jeu d'instructions de base	*Semaines 4–5*	*2*	*Classe Instructions avec sous-ensemble d'opcodes*
Dashboard et Editeur	*Semaine 6*	*1*	*Interface utilisateur minimale opérationnelle*
Tests, validation et documentation	*Semaine 7*	*1*	*Rapports de tests, manuel, rapport de projet*

V.2. Principaux Jalons

M1 – Architecture validée
Diagramme des classes et interfaces défini.

M2 – CPU + mémoire opérationnels
Lecture/écriture mémoire, cycle fetch-decode-execute simple.

M3 – Jeu d'instructions minimal
Sous-ensemble d'instructions arithmétiques et de branchement pris en charge.

M4 – Dashboard et Editeur
Interface interactive complète.

M5 – Version présentable
Application stable avec documentation.

VI. Gestion du Budget et Ressources & Indicateurs de Performance (KPIs) et Suivi

VI.1. Synthèse Budgétaire (Théorique)

Le budget direct est limité dans un contexte académique, mais les ressources incluent les logiciels (IntelliJ IDEA Community), le matériel existant et le temps de développement.

VII.1. Tableau des KPIs

Taux de complétude du jeu d'instructions	*% d'opcodes 6809 implémentés*	*≥ 60 %*	*55–70 %*
Taux de réussite des tests unitaires	*% de tests unitaires passés*	*≥ 90 %*	*90–95 %*
Stabilité en exécution	*Nombre moyen de plantages / 100 exécutions*	*0*	*0–1*
Temps de chargement d'un programme	*Temps moyen pour charger un fichier assembleur*	*< 2 secondes*	*~1 seconde*

XII. Conclusion

Le simulateur Motorola 6809 en Java offre une base solide pour l'apprentissage de l'architecture des processeurs. L'architecture modulaire assure un code lisible et extensible, atteignant les objectifs principaux avec des indicateurs de qualité satisfaisants.