جامعة الحسن الأول
UNIVERSITÉ HASSAN 1er
Université Hassan 1er
Faculté des Sciences et Techniques de Settat
كلية العلوم و التقنيات سطات
FACULTÉ DES SCIENCES ET TECHNIQUES SETTAT

Rapport du Simulateur Motorola 6809

Créé par :
AMELLACHOU SALMA
OMAYMA AIT KHAM

Encadré par :
Mr. Hicham BENALLA

Introduction

Ce rapport présente la conception et l'implémentation d'un simulateur du microprocesseur Motorola 6809 développé en Java. Il décrit l'architecture logicielle du simulateur, le rôle des différentes classes ainsi que le fonctionnement interne de l'émulation et du débogage.

Architecture générale du simulateur Motorola 6809

1. Organisation du projet

Le simulateur du microprocesseur Motorola 6809 est conçu selon une **architecture modulaire**, afin de séparer clairement les différentes responsabilités du système.
Chaque module du simulateur correspond à une partie fonctionnelle du microprocesseur réel ou à un service logiciel associé (exécution, interface, débogage).

Cette organisation facilite :

- la compréhension du code,
- la maintenance,
- l'extension du simulateur (ajout d'instructions, amélioration de l'interface, etc.).

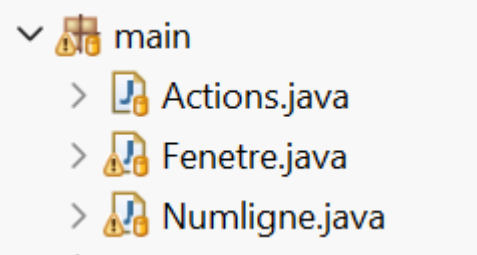
Package main – Interface et contrôle du simulateur

Main

├── Actions

├── Fenetre

└── Numligne



Rôle et fonctionnement

Le package main regroupe les classes responsables de l'interface graphique et de la gestion des interactions utilisateur avec le simulateur MC6809. Il permet à l'utilisateur de saisir un programme assembleur, de lancer la compilation, d'exécuter le programme et de visualiser l'état interne du processeur, en récupérant les actions de l'utilisateur et en affichant les résultats sous forme graphique.

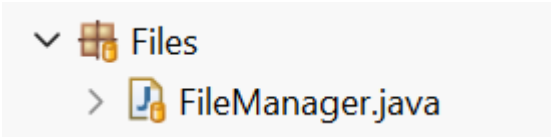
Exemples de classes

- **Fenetre** : Représente la fenêtre principale du simulateur et fournit l'environnement graphique global. Elle affiche l'éditeur, les registres, les flags, les messages d'erreur et les boutons de commande.
- **Actions** : Gère les actions déclenchées par l'utilisateur et contrôle le cycle d'exécution. Elle interagit avec la mémoire et le processeur simulé et permet de lancer la compilation ou d'exécuter le programme.
- **Numligne** : Affiche automatiquement les numéros de lignes dans l'éditeur de code. Elle met à jour dynamiquement la numérotation pour faciliter la lisibilité et le débogage.

Package Files – Gestion des fichiers

Files

└── FileManager



Rôle et fonctionnement

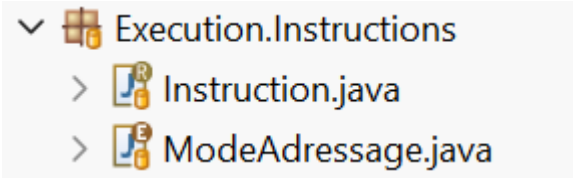
Le package Files regroupe les classes responsables de la gestion des fichiers dans le simulateur MC6809.
Il permet à l'utilisateur de charger un programme assembleur depuis un fichier ou de sauvegarder son code dans un fichier.

Exemples de classe

- **FileManager** : Fournir des méthodes statiques pour charger et sauvegarder des fichiers texte contenant le programme assembleur.

Package Execution.Instructions – Gestion des instructions du simulateur

Execution.Instructions
├── Instruction
└── modeAdressage



Rôle et fonctionnement :

Le package Execution.Instructions regroupe les classes responsables de la représentation et de l'exécution des instructions assembleur dans le simulateur MC6809. Il reçoit chaque instruction, détermine son type et son mode d'adressage, puis effectue les opérations correspondantes sur le

processeur et la mémoire, en interagissant avec les fonctions arithmétiques ou de chargement/stockage.

Exemples de classes

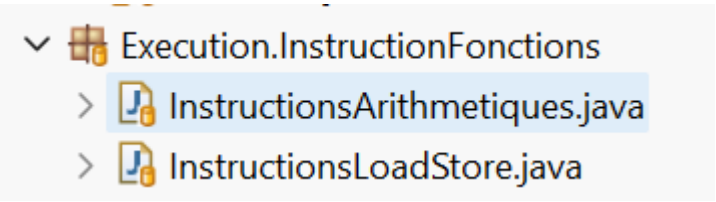
- **Instruction** : Représente une instruction assembleur et exécute son effet sur le processeur simulé. Elle calcule les adresses et valeurs selon le mode d'adressage et déclenche les opérations correspondantes.
- **ModeAdressage** : Définie les différents modes d'adressage possibles (Immediat, Direct, Indexe) pour les instructions. Elle permet à chaque instruction de savoir comment interpréter ses opérandes et accéder à la mémoire ou aux registres.

Package InstructionFonctions – Fonctions des instructions du processeur Execution.

Execution.InstructionFonctions

├── InstructionsArithmetiques

└── InstructionsLoadStore



Rôle et fonctionnement

Le package `Execution.InstructionFonctions` contient les classes qui implémentent les fonctions spécifiques aux instructions du simulateur MC6809, comme les opérations arithmétiques et les opérations de chargement/stockage. Il permet aux instructions de modifier les registres et la mémoire du processeur tout en mettant à jour correctement les flags (N, Z, C, V, H).

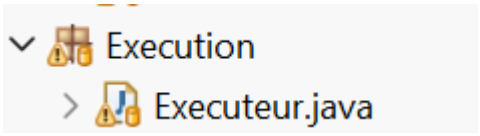
Exemples de classes

- **InstructionsArithmetiques** : Gère l'exécution des opérations arithmétiques telles que ADD et ADC. Elle met à jour les registres ciblés et les flags du processeur en fonction du résultat des calculs.
- **InstructionsLoadStore** : Gère le chargement et le stockage des registres et accumulateurs. Elle écrit ou lit des valeurs dans la mémoire et met à jour les flags appropriés (N, Z, C, V).

Package Execution – Exécution des programmes assembleur

Execution

└── Executeur



Rôle et fonctionnement

Le package Execution contient les classes responsables de l'exécution des programmes assembleur sur le simulateur MC6809. Il calcule les adresses effectives, interagit avec la mémoire et les registres, et applique la logique du processeur pour simuler l'exécution des instructions.

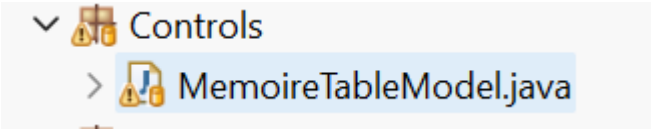
Exemples de classes

- **Executeur** : Calcule les adresses effectives pour le mode indexé et gère l'exécution des instructions sur le processeur simulé. Elle interprète les pré/post incréments et les offsets, et met à jour les registres en conséquence.

Package Controls – Contrôle et affichage de la mémoire

Controls

└── MemoireTableModel



Rôle et fonctionnement

Le package Controls regroupe les classes permettant de visualiser et de modifier la mémoire du simulateur MC6809 via des composants graphiques. Il fournit une interface tableur pour afficher le contenu mémoire et gérer les modifications tout en validant les entrées utilisateur.

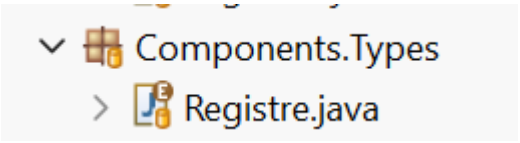
Exemples de classes

- **MemoireTableModel** : Fournit un modèle de tableau pour afficher et éditer la mémoire. Elle gère la lecture/écriture en mémoire et vérifie la validité des valeurs saisies par l'utilisateur.

Package Components.Types – Définition des registres du processeur

Components.Types

└── Registre



Rôle et fonctionnement

Le package Components.Types contient les types de données utilisés par le processeur simulé, notamment les registres. Il permet de gérer les différents registres et de déterminer leurs caractéristiques, comme la taille (8 bits ou 16 bits).

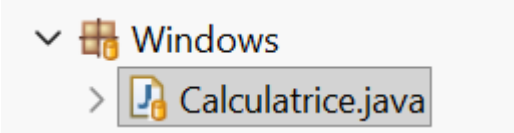
Exemples de classes

- **Registre** : Représente tous les registres du processeur MC6809, y compris les accumulateurs, registres index et flags. Elle fournit une méthode pour savoir si un registre est sur 8 bits ou non.

Package Windows – Fenêtres et interfaces graphiques du simulateur

Windows

└── Calculatrice



Rôle et fonctionnement

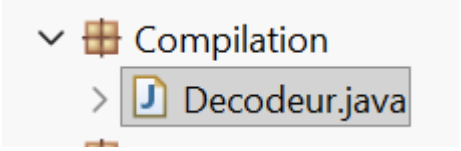
Le package Windows contient les outils graphiques annexes du simulateur, comme la Calculatrice hexadécimale. Il permet de créer et gérer ces fenêtres utilitaires indépendantes, tout en assurant une interaction simple et fluide avec l'utilisateur.

Exemples de classes

- **Calculatrice** : Permet d'ouvrir une fenêtre de calcul hexadécimal. Elle fournit un champ de saisie pour entrer une opération, un bouton pour lancer le calcul et un label pour afficher le résultat. La classe gère les opérations +, −, * et /, convertit les valeurs hexadécimales en entiers pour effectuer le calcul, puis affiche le résultat en hexadécimal. Elle empêche également l'ouverture de plusieurs fenêtres identiques et gère les erreurs de saisie.

Package Compilation – Décodage et conversion des instructions assembleur

Compilation
└── Decodeur



Rôle et fonctionnement

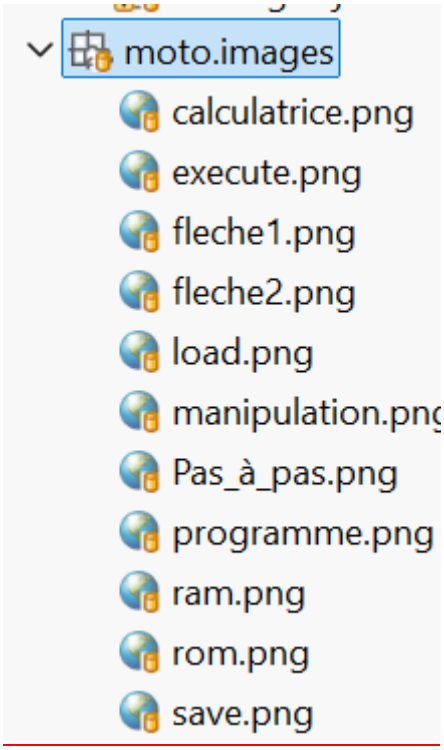
Le package **Compilation** contient les classes responsables de la traduction des instructions assembleur MC6809 en code machine que le simulateur peut exécuter. Il permet de décoder chaque ligne du programme, de vérifier la validité des mnemonics et des opérandes, et d'écrire les octets correspondants en mémoire. Le package gère les différents modes d'adressage (immédiat, direct, indexé) et assure la détection des erreurs de syntaxe ou de dépassement de taille.

Exemples de classes

- **Decodeur** : Cette classe analyse le code source assembleur ligne par ligne et génère les instructions correspondantes pour la mémoire du simulateur.
    - Elle initialise une table d'opcodes et une liste de mnemonics reconnues.
    - Elle vérifie la terminaison correcte du programme par `END`.
    - Elle gère la conversion des opérandes hexadécimales, le contrôle des dépassements de taille (8 bits/16 bits), et les modes d'adressage.
    - En cas d'erreur (instruction inconnue, opérande invalide, valeur trop grande, etc.), elle signale l'erreur et n'écrit pas l'instruction en mémoire.
    - Elle calcule et écrit l'opcode et les opérandes correspondants dans la mémoire du simulateur pour l'exécution ultérieure.

Package moto.images – Gestion des images et icônes de l'interface Graphique

Le package **moto.images** a pour rôle principal de centraliser et de gérer toutes les ressources graphiques utilisées par le simulateur. Il fournit les images nécessaires pour les boutons, les icônes et autres éléments visuels de l'interface, permettant ainsi d'assurer une cohérence visuelle et une expérience utilisateur fluide. En regroupant ces images dans un package dédié, le code devient plus organisé et facile à maintenir.


moto.images
calculatrice.png
execute.png
fleche1.png
fleche2.png
load.png
manipulation.png
Pas_à_pas.png
programme.png
ram.png
rom.png
save.png