ASKA GROUPE

DECEMBRE, 21 2023

PROJET DE MODULE

SIMULATEUR MOTO6809

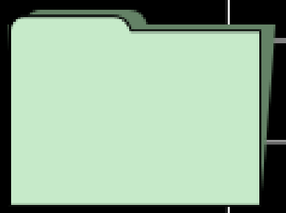
M. BENALLA HICHAM

Let's get started

AGENDA

- Introduction
- Front End
- Back End
- Conclusion

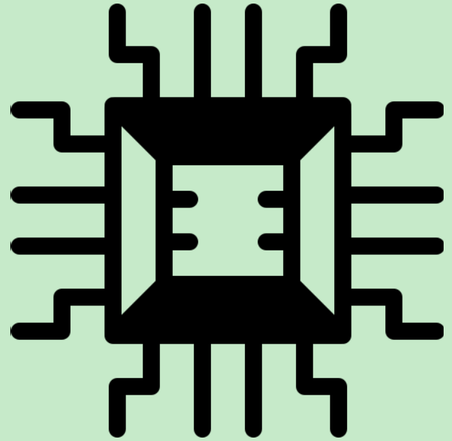
INTRODUCTION



Aujourd'hui, nous allons explorer le simulateur moto 6809, élaboré au moyen du langage de programmation Java NetBeans. Cette réalisation vise à approfondir notre compréhension de l'architecture des processus tout en simulant avec précision les instructions d'exécution.

INTRODUCTION

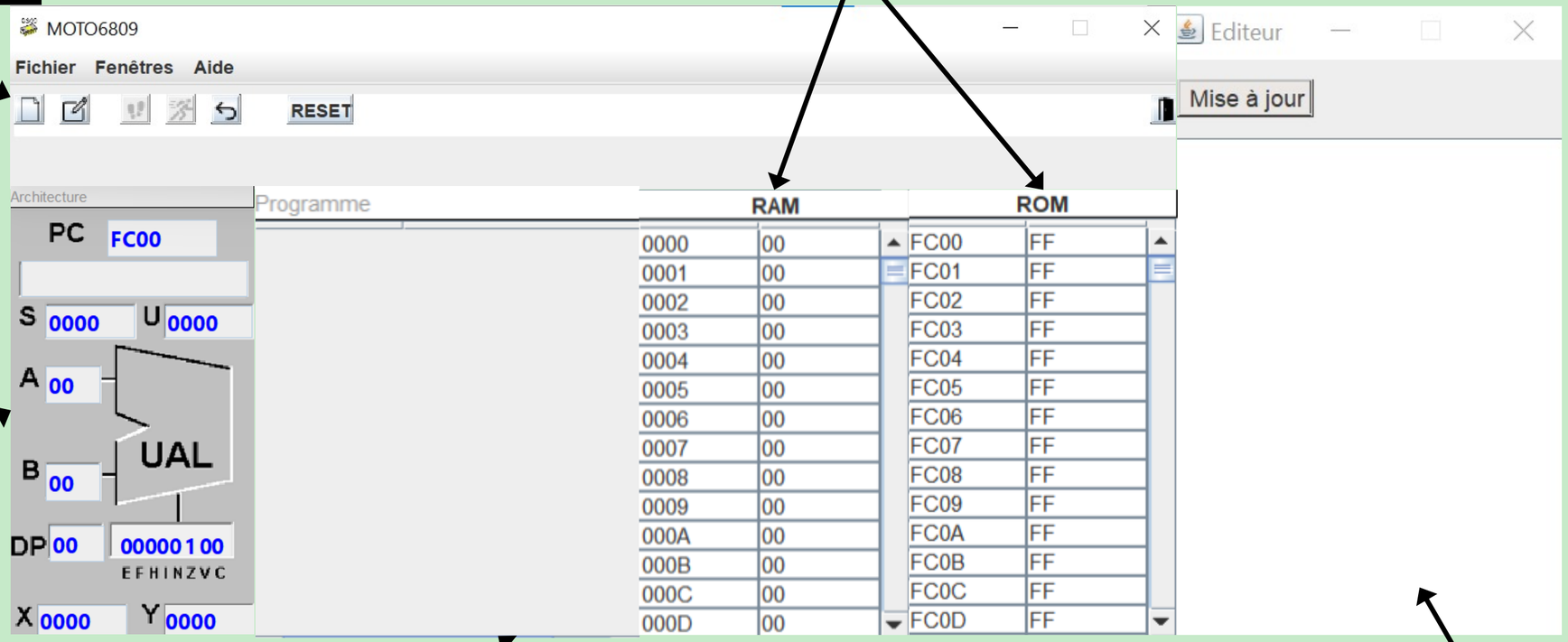
Le simulateur de moto6809 est un outil permettant d'émuler et de tester des programmes écrits en langage assembleur pour le processeur Motorola 6809. Il offre une interface graphique pour faciliter la simulation et le débogage des programmes.



FRONTEND

1. Interface principale

4. Interfaces de memoires RAM &ROM



2. Interface d'architecture interne

3. Interface du programme

5. Interface d'éditeur

BACK-END

La classe Instruction

La classe `InstrDecoder` joue un rôle crucial en implémentant le décodeur d'instructions. Elle offre une flexibilité essentielle pour traiter divers types d'adressages. Cette classe interagit avec l'interface graphique pour afficher les instructions en temps réel. De plus, elle gère les erreurs et garantit la mise à jour dynamique des valeurs des registres lors de l'exécution du programme.

- Méthodes de Traitement des Instructions
- Traitement du Code Assemblé
- Méthodes Setter et Gestion des Erreurs

Les classes RAM et ROM

Les classes `RAM` et `ROM` sont des fenêtres graphiques conçues pour simuler la mémoire RAM et ROM d'un processeur. Elles utilisent des tableaux visuels pour afficher respectivement les adresses et les valeurs de la mémoire RAM, ainsi que les instructions stockées dans la mémoire ROM. Ces fenêtres permettent également l'édition interactive des valeurs, offrant ainsi une interface utilisateur conviviale.

- Initialisation de la fenêtre de la RAM/ROM
  Configuration de la table et ajout des adresses et valeurs initiales à la mémoire RAM/ROM.

BACK END

Méthodes de Traitement des Instructions

- adrDirect()
  - Traite les instructions avec un mode d'adressage direct.
  - Extrait les informations nécessaires, ajuste les registres et met à jour l'interface graphique.
- adrImmediat()
  - Gère les instructions avec un mode d'adressage immédiat.
  - Extrait les valeurs, les traite selon le type d'instruction, puis met à jour les registres.
- adrInherent()
  - Gère les instructions avec un mode d'adressage inhérent.
  - Effectue des opérations spécifiques en fonction du type d'instruction et met à jour les registres en conséquence.

Traitement du Code Assemblé

- traiterUPDATE(String textarea)
  - Analyse chaque ligne du code assemblé, détecte les erreurs et mise à jour le Programme et ROM.
- traiterLigne(String textarea)
  - Traite chaque ligne du code assemblé en fonction de son type, mettant à jour l'interface utilisateur et les registres.
- traiterLinePrLine(String textarea)
  - Traite les lignes du code assemblé une par une, permettant un suivi en temps réel de l'exécution du programme.

Méthodes Setter et Gestion des Erreurs

- setters
  - Met à jour les valeurs des différents registres, du compteur de programme, des indicateurs et de la mémoire ROM en fonction des besoins.
- ERROR()
  - Gère les erreurs en affichant un message d'erreur via une boîte de dialogue.

BACK END

RAM/ROM

- La classe Java `ROM` crée une interface graphique utilisant Swing pour afficher une table de valeurs hexadécimales et décimales.
- La fenêtre principale (`JFrame`) est configurée sans décoration, avec une barre de titre "ROM".
- La table est créée avec `JTable`, et son contenu est géré par un modèle de données (`DefaultTableModel`).
- Un panneau (`JPanel`) est utilisé pour la barre de titre avec bordure et couleur de fond personnalisées.
- La fonction de déplacement de la fenêtre est implémentée à l'aide de gestionnaires d'événements de souris (`MouseAdapter`).
- La table affiche des données de la mémoire ROM, avec des adresses de 0xFC00 à 0xFFFF et des valeurs hexadécimales et décimales prédéfinies.

Difference?

- Dans la classe `RAM`, la table affiche des adresses hexadécimales de 0x0000 à 0x1FFF avec des valeurs décimales initiales de 0.
- Dans la classe `ROM`, la table affiche des adresses hexadécimales de 0xFC00 à 0xFFFF avec des valeurs hexadécimales initiales de 0xFF.

MODE D'ADRESSAGES

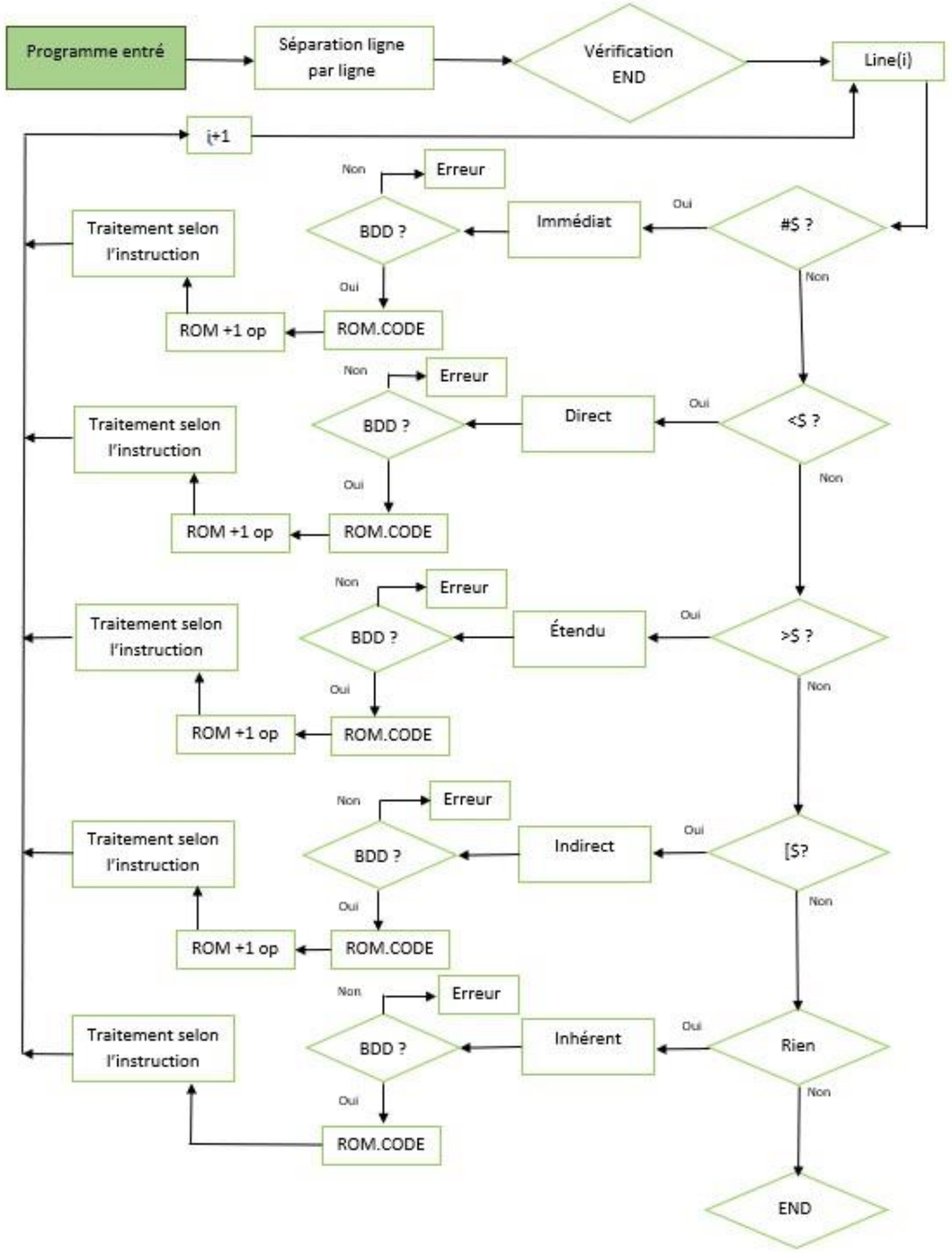
Modes

- Immédiat
- Direct
- Inhérent
- ÉtenduDirect/Indirect
- Indexé

Instructions

- LD
- ST
- ADD
- SUB
- DEC
- INC
- CLR
- ABX
- LEA
- AND
- OR
- SWI
- NOP
- ASL
- ASR
- LSL
- LSR

ALGORITHME


Programme entré
Séparation ligne par ligne
Vérification END
Line(i)
i+1
#$?
Immédiat
BDD ?
Erreur
Traitement selon l'instruction
ROM +1 op
ROM.CODE
<$?
Direct
>$?
Étendu
[$?
Indirect
Rien
Inhérent
Oui
Non
END

CONCLUSION

En conclusion, la création du simulateur Moto6809 a abouti à des interfaces dédiées à la fenêtre principale, celle de l'architecture, la RAM, la ROM, le programme et l'éditeur. Ces fenêtres offrent une immersion visuelle dans le fonctionnement interne du processeur. Les codes d'instructions de l'assembleur du Moto6809 ont été implémentés avec précision, jouant un rôle central dans la programmation et l'exécution du programme. L'ensemble crée une expérience interactive et éducative pour explorer cette architecture emblématique.