Guide d'Utilisateur

Simulateur Motorola 6809

Version 1.0

25 décembre 2025

Créé par :
CHAHOUB Nouhaila
ENNAJI Aymen
AMROUN Hiba

Documentation complète pour l'utilisation du simulateur

Table des matières

1 Introduction

Bienvenue dans le guide d'utilisateur du Simulateur Motorola 6809. Ce simulateur vous permet d'écrire, d'exécuter et de déboguer des programmes en langage assembleur pour le microprocesseur Motorola 6809.

Ce guide vous expliquera en détail chaque composant de l'interface, leur fonctionnement et comment les utiliser efficacement.

2 Fenêtre Principale du Simulateur

La fenêtre principale est le point de départ de toute utilisation du simulateur. Elle contient tous les boutons de contrôle nécessaires pour gérer votre programme.

2.1 Barre de Boutons de Contrôle

2.1.1 Bouton New

- **Fonction :** Créer un nouveau programme assembleur
- **Description :** Ce bouton ouvre l'éditeur de code assembleur et vous permet de commencer à écrire un nouveau programme. Il efface tout code précédemment chargé.
- **Utilisation :** Cliquez sur ce bouton au début d'une nouvelle session de programmation.

2.1.2 Bouton Run

- **Fonction :** Exécuter le programme complet
- **Description :** Lance l'exécution complète du programme assembleur chargé, de la première instruction jusqu'à la fin (instruction END).
- **Utilisation :** Utilisez ce bouton lorsque vous voulez voir le résultat final de votre programme sans passer par chaque instruction.

2.1.3 Bouton Step

- **Fonction :** Exécution pas à pas
- **Description :** Active le mode débogage qui permet d'exécuter le programme instruction par instruction. Cela vous aide à comprendre le comportement de chaque instruction.
- **Utilisation :** Idéal pour déboguer votre programme et voir comment les registres et la mémoire changent à chaque étape.

2.1.4 Bouton Reset

- **Fonction :** Réinitialiser le simulateur
- **Description :** Remet tous les registres à zéro, efface la mémoire et réinitialise le compteur de programme. Le simulateur revient à son état initial.
- **Utilisation :** Utilisez ce bouton pour recommencer une simulation depuis le début ou pour nettoyer l'état du simulateur.

2.1.5 Bouton RAM

- **Fonction :** Afficher la mémoire RAM
- **Description :** Ouvre une fenêtre qui montre le contenu de la mémoire vive (RAM) du simulateur, de l'adresse 0000 à FFFF.
- **Utilisation :** Permet de visualiser et de modifier les données stockées en mémoire pendant l'exécution du programme.

2.1.6 Bouton ROM

- **Fonction :** Afficher la mémoire ROM
- **Description :** Ouvre une fenêtre qui montre le contenu de la mémoire morte (ROM) du simulateur.
- **Utilisation :** Utilisé pour visualiser les données en lecture seule ou le code programme stocké en ROM.

2.1.7 Bouton Registers

- **Fonction :** Afficher les registres
- **Description :** Ouvre une fenêtre détaillée montrant l'état de tous les registres du processeur (A, B, D, X, Y, S, U, CCR, PC).
- **Utilisation :** Essentiel pour suivre l'évolution des données dans les registres pendant l'exécution du programme.

2.1.8 Bouton Save

- **Fonction :** Sauvegarder le programme
- **Description :** Permet de sauvegarder votre code assembleur dans un fichier .asm sur votre ordinateur.
- **Utilisation :** Utilisez ce bouton pour conserver votre travail et pouvoir le recharger plus tard.

2.1.9 Bouton Stop

- **Fonction :** Arrêter l'exécution
- **Description :** Interrompt immédiatement l'exécution du programme en cours.
- **Utilisation :** Utile pour arrêter un programme qui est en boucle infinie ou que vous souhaitez interrompre.

2.2 Journal d'Exécution (Execution Log)

- **Fonction :** Afficher l'historique des actions
- **Description :** Cette zone de texte affiche toutes les actions effectuées dans le simulateur avec un horodatage :
  - Ouverture des différentes fenêtres
  - Chargement du code
  - Instructions validées

  - Sauvegarde des fichiers
- **Utilisation :** Permet de suivre chronologiquement toutes les opérations effectuées et de diagnostiquer les problèmes.

2.3 Conclusion - Fenêtre Principale

Les boutons de la fenêtre principale travaillent ensemble pour offrir un environnement de développement complet :

1. **New** permet de créer un nouveau programme
2. **Run** et **Step** offrent deux modes d'exécution (complet ou pas à pas)
3. **RAM**, **ROM**, et **Registers** donnent une visibilité complète sur l'état du système
4. **Save** préserve votre travail
5. **Reset** et **Stop** permettent de contrôler l'exécution
6. Le **journal d'exécution** documente toutes vos actions

3 Éditeur de Code Assembleur

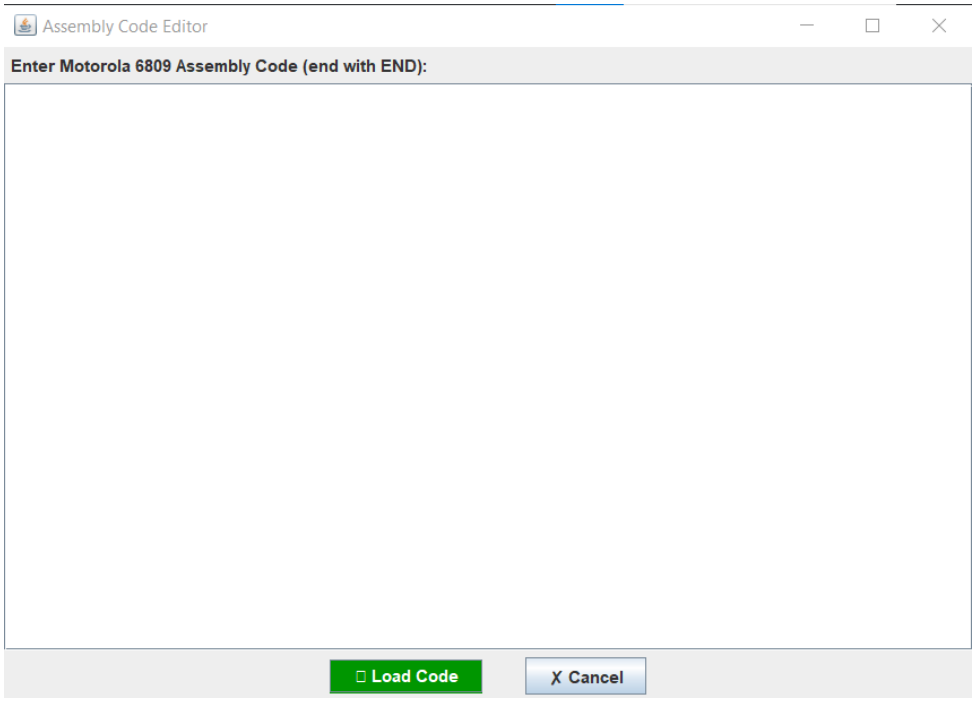

FIGURE 1 – Éditeur de code assembleur

L'éditeur de code assembleur s'ouvre lorsque vous cliquez sur le bouton **New**. C'est ici que vous écrivez vos programmes en langage assembleur Motorola 6809.

3.1 Zone de Texte d'Édition

- **Fonction :** Saisir le code assembleur
- **Description :** Grande zone de texte où vous tapez votre programme assembleur. L'en-tête indique : "Enter Motorola 6809 Assembly Code (end with END) :"
- **Règles importantes :**
  - Chaque instruction doit être sur une nouvelle ligne
  - Le programme doit se terminer par l'instruction END
  - Respectez la syntaxe du Motorola 6809
- **Utilisation :** Tapez votre code directement dans cette zone. Par exemple :

```
LDA #$15
STA $2000
END
```

3.2 Bouton Load Code

- **Fonction :** Charger le code dans le simulateur

— **Description :** Ce bouton valide et charge votre code assembleur dans la mémoire du simulateur. Le code est analysé et préparé pour l'exécution.

— **Utilisation :** Après avoir écrit votre programme, cliquez sur ce bouton. Si le code est valide, la fenêtre se fermera et vous verrez un message de confirmation dans le journal d'exécution.

— **Note :** Si votre code contient des erreurs de syntaxe, un message d'erreur s'affichera.

3.3 Bouton X Cancel

— **Fonction :** Annuler et fermer l'éditeur

— **Description :** Ferme la fenêtre de l'éditeur sans charger le code. Toutes les modifications non sauvegardées seront perdues.

— **Utilisation :** Utilisez ce bouton si vous avez ouvert l'éditeur par erreur ou si vous ne souhaitez pas charger le code que vous avez écrit.

3.4 Conclusion - Éditeur de Code

L'éditeur de code assembleur est l'interface où vous créez vos programmes :

— La **zone de texte** vous permet d'écrire votre code assembleur

— Le bouton **Load Code** valide et charge votre programme dans le simulateur

— Le bouton **Cancel** vous permet d'abandonner sans sauvegarder

Ensemble, ces composants forment un éditeur simple mais efficace pour la programmation en assembleur Motorola 6809.

4 Visualiseur de Mémoire RAM

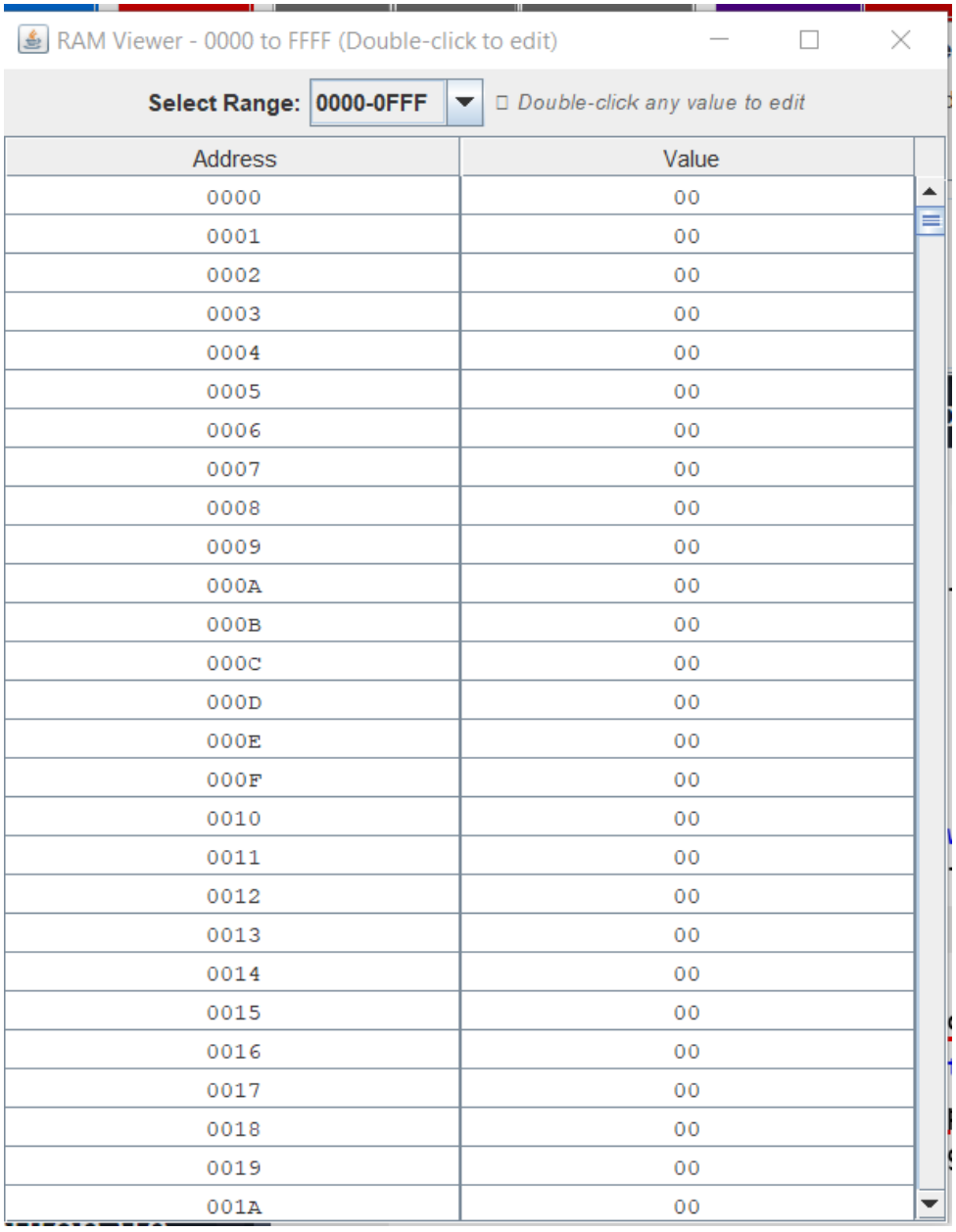

FIGURE 2 – Visualiseur de mémoire RAM

Le visualiseur de RAM affiche le contenu de la mémoire vive du simulateur. Cette fenêtre s'ouvre en cliquant sur le bouton **RAM** de la fenêtre principale.

4.1 Menu de Sélection de Plage

- **Fonction :** Sélectionner la plage d'adresses à afficher
- **Description :** Menu déroulant affichant "0000-0FFF" par défaut. Permet de choisir quelle portion de la mémoire vous souhaitez visualiser.
- **Options disponibles :** Différentes plages de 4096 octets (0000-0FFF, 1000-1FFF, etc.)

— **Utilisation :** Sélectionnez la plage qui contient les adresses mémoire que vous souhaitez examiner.

4.2 Note d'Édition

— **Texte affiché :** "Double-click any value to edit"

— **Fonction :** Indique comment modifier les valeurs en mémoire

— **Description :** Vous pouvez modifier directement le contenu de la mémoire en double-cliquant sur n'importe quelle valeur.

— **Utilisation :** Double-cliquez sur une valeur, entrez la nouvelle valeur en hexadécimal, et validez.

4.3 Tableau de Mémoire

— **Fonction :** Afficher le contenu de la mémoire

— **Description :** Tableau à deux colonnes :

  — **Colonne "Address" :** Affiche l'adresse mémoire en hexadécimal (0000, 0001, 0002, etc.)

  — **Colonne "Value" :** Affiche le contenu (la valeur) stockée à cette adresse en hexadécimal (00 par défaut)

— **Utilisation :**

  — Parcourez le tableau pour voir quelles données sont stockées en mémoire

  — La mémoire est affichée par ordre croissant d'adresses

  — Chaque valeur est affichée sur 2 chiffres hexadécimaux (00 à FF)

4.4 Conclusion - Visualiseur RAM

Le visualiseur de RAM permet une inspection complète de la mémoire vive :

— Le **menu de sélection** vous aide à naviguer dans différentes zones mémoire

— Le **tableau** affiche clairement les adresses et leurs valeurs

— La fonctionnalité de **double-clic** permet de modifier les valeurs pour tester différents scénarios

Cet outil est essentiel pour comprendre comment votre programme manipule les données en mémoire et pour déboguer les problèmes liés au stockage de données.

5 Visualiseur de Mémoire ROM

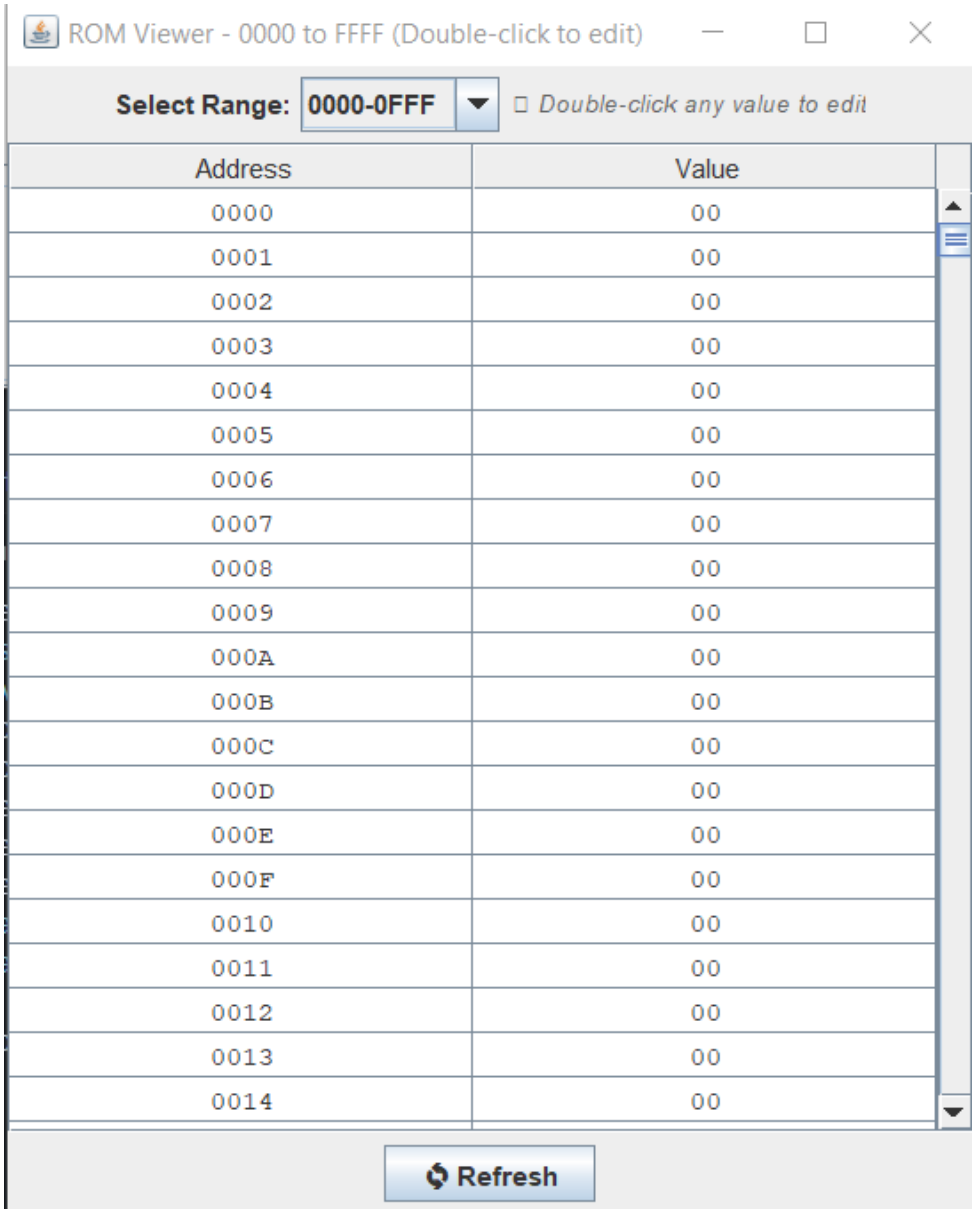

FIGURE 3 – Visualiseur de mémoire RAM

Le visualiseur de ROM affiche le contenu de la mémoire morte du simulateur. Cette fenêtre s'ouvre en cliquant sur le bouton **ROM** de la fenêtre principale.

5.1 Menu de Sélection de Plage

- **Fonction :** Sélectionner la plage d'adresses ROM à afficher
- **Description :** Menu déroulant similaire au visualiseur RAM, affichant par défaut "0000-0FFF".
- **Utilisation :** Permet de naviguer dans différentes zones de la mémoire ROM pour visualiser le code programme ou les données constantes.

5.2 Note d'Édition

- **Texte affiché :** "Double-click any value to edit"
- **Fonction :** Même fonctionnalité que pour la RAM
- **Description :** Bien que la ROM soit normalement en lecture seule, le simulateur permet de modifier les valeurs pour des besoins de test.
- **Utilisation :** Double-cliquez sur une valeur pour la modifier temporairement dans le simulateur.

5.3 Tableau de Mémoire ROM

- **Fonction :** Afficher le contenu de la ROM
- **Description :** Structure identique au visualiseur RAM avec deux colonnes :
  - **Colonne "Address" :** Adresses mémoire en hexadécimal
  - **Colonne "Value" :** Valeurs stockées en ROM
- **Utilisation :** Permet de voir où votre code programme est stocké et quelles valeurs constantes sont présentes en ROM.

5.4 Bouton Refresh

- **Fonction :** Actualiser l'affichage
- **Description :** Recharge et affiche les valeurs actuelles de la mémoire ROM.
- **Utilisation :** Cliquez sur ce bouton après avoir chargé un nouveau programme ou modifié des valeurs pour voir les changements.

5.5 Conclusion - Visualiseur ROM

Le visualiseur de ROM fonctionne de manière similaire au visualiseur RAM mais pour la mémoire morte :

- Le **menu de sélection** permet de naviguer dans les différentes zones ROM
- Le **tableau** affiche le contenu de la ROM
- Le bouton **Refresh** actualise l'affichage
- La possibilité d'**édition** permet de tester différentes configurations

Ce visualiseur est particulièrement utile pour vérifier que votre programme a été correctement chargé en mémoire.

6 Visualiseur de Registres

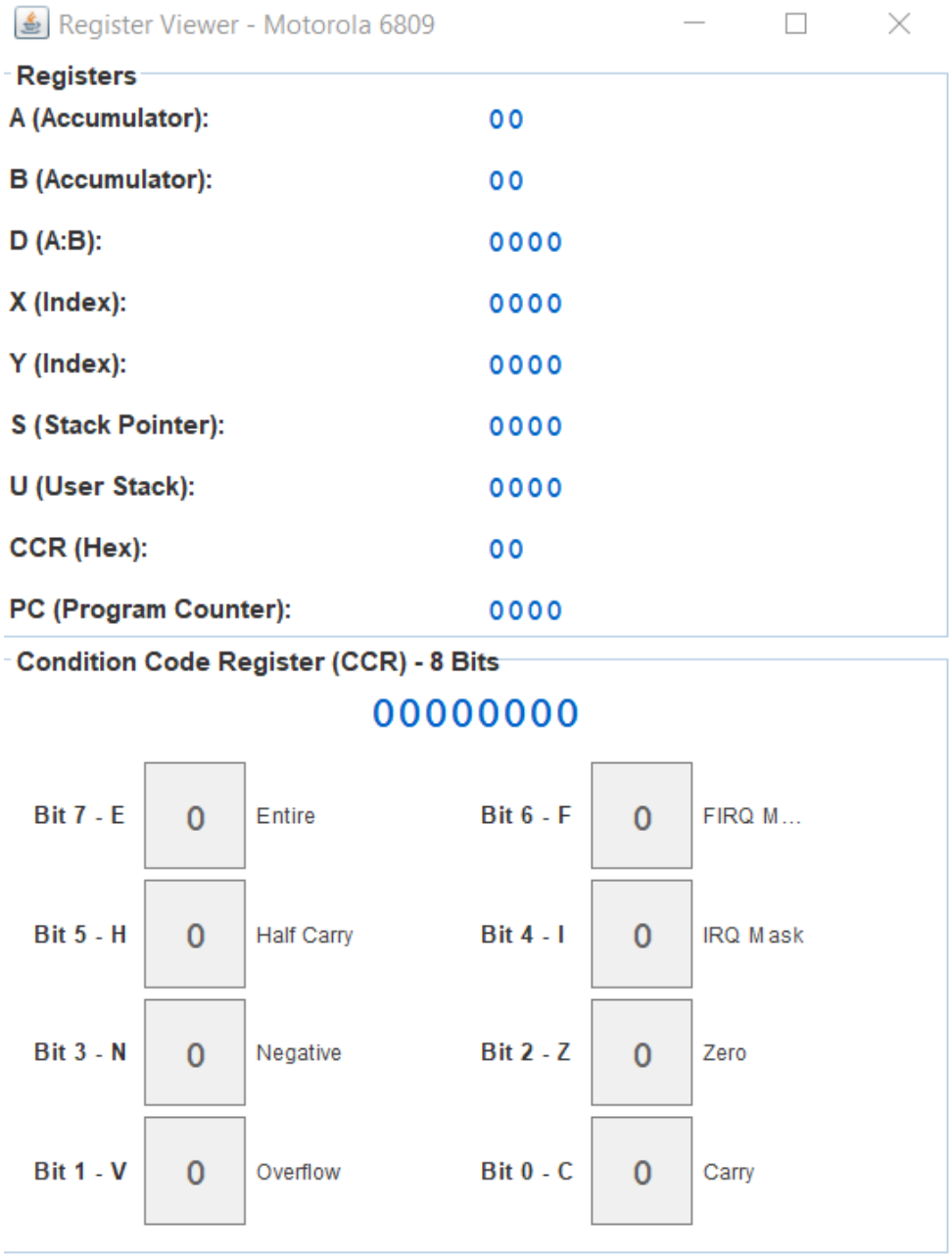

FIGURE 4 – Visualiseur de registres

Le visualiseur de registres affiche l'état de tous les registres du processeur Motorola 6809. Cette fenêtre s'ouvre en cliquant sur le bouton **Registers** de la fenêtre principale.

6.1 Section des Registres Principaux

6.1.1 Registre A (Accumulateur)

- **Fonction :** Accumulateur 8 bits
- **Description :** Registre de travail principal pour les opérations arithmétiques et logiques sur 8 bits.

— **Affichage :** Valeur sur 2 chiffres hexadécimaux (00 à FF)

6.1.2 Registre B (Accumulateur)

— **Fonction :** Second accumulateur 8 bits

— **Description :** Registre de travail secondaire, peut être combiné avec A pour former le registre D.

— **Affichage :** Valeur sur 2 chiffres hexadécimaux (00 à FF)

6.1.3 Registre D (A :B)

— **Fonction :** Accumulateur double 16 bits

— **Description :** Combinaison des registres A et B formant un registre 16 bits pour les opérations sur des valeurs plus grandes.

— **Affichage :** Valeur sur 4 chiffres hexadécimaux (0000 à FFFF)

6.1.4 Registre X (Index)

— **Fonction :** Registre d'index 16 bits

— **Description :** Utilisé pour l'adressage indexé et comme pointeur général.

— **Affichage :** Valeur sur 4 chiffres hexadécimaux (0000 à FFFF)

6.1.5 Registre Y (Index)

— **Fonction :** Second registre d'index 16 bits

— **Description :** Registre d'index supplémentaire pour l'adressage complexe.

— **Affichage :** Valeur sur 4 chiffres hexadécimaux (0000 à FFFF)

6.1.6 Registre S (Stack Pointer)

— **Fonction :** Pointeur de pile système

— **Description :** Pointe vers le sommet de la pile système utilisée pour les appels de sous-programmes et les interruptions.

— **Affichage :** Valeur sur 4 chiffres hexadécimaux (0000 à FFFF)

6.1.7 Registre U (User Stack)

— **Fonction :** Pointeur de pile utilisateur

— **Description :** Pointeur de pile secondaire pour les données utilisateur.

— **Affichage :** Valeur sur 4 chiffres hexadécimaux (0000 à FFFF)

6.1.8 CCR (Hex)

— **Fonction :** Registre de code de condition (format hexadécimal)

— **Description :** Affichage compact du registre CCR en hexadécimal.

— **Affichage :** Valeur sur 2 chiffres hexadécimaux (00 à FF)

6.1.9 PC (Program Counter)

- **Fonction :** Compteur de programme
- **Description :** Contient l'adresse de la prochaine instruction à exécuter. S'incrémente automatiquement après chaque instruction.
- **Affichage :** Valeur sur 4 chiffres hexadécimaux (0000 à FFFF)
- **Importance :** Essentiel pour suivre l'exécution du programme pas à pas.

6.2 Registre de Code de Condition (CCR) - Vue Détaillée

- **Fonction :** Afficher l'état détaillé de tous les drapeaux (flags)
- **Description :** Section détaillée montrant les 8 bits du registre CCR individuellement sur 8 lignes avec leur signification.
- **Affichage :** Chaque bit est affiché en binaire (0 ou 1) avec son nom complet

6.2.1 Bit 7 - E (Entire)

- **Nom complet :** Entire
- **Fonction :** Indicateur de registre entier empilé
- **Description :** Indique si tous les registres ont été empilés lors d'une interruption (1) ou seulement une partie (0).

6.2.2 Bit 6 - F (FIRQ Mask)

- **Nom complet :** FIRQ Mask
- **Fonction :** Masque d'interruption rapide
- **Description :** Désactive (1) ou active (0) les interruptions rapides FIRQ.

6.2.3 Bit 5 - H (Half Carry)

- **Nom complet :** Half Carry
- **Fonction :** Indicateur de retenue demi-octet
- **Description :** Utilisé pour les opérations BCD (Binary Coded Decimal). Mis à 1 s'il y a une retenue du bit 3 au bit 4.

6.2.4 Bit 4 - I (IRQ Mask)

- **Nom complet :** IRQ Mask
- **Fonction :** Masque d'interruption normale
- **Description :** Désactive (1) ou active (0) les interruptions normales IRQ.

6.2.5 Bit 3 - N (Negative)

- **Nom complet :** Negative
- **Fonction :** Indicateur de résultat négatif
- **Description :** Mis à 1 si le résultat de la dernière opération est négatif (bit de poids fort à 1 en complément à 2).

6.2.6 Bit 2 - Z (Zero)

- **Nom complet :** Zero
- **Fonction :** Indicateur de résultat zéro
- **Description :** Mis à 1 si le résultat de la dernière opération est zéro.

6.2.7 Bit 1 - V (Overflow)

- **Nom complet :** Overflow
- **Fonction :** Indicateur de dépassement
- **Description :** Mis à 1 s'il y a un dépassement lors d'opérations arithmétiques signées.

6.2.8 Bit 0 - C (Carry)

- **Nom complet :** Carry
- **Fonction :** Indicateur de retenue
- **Description :** Mis à 1 s'il y a une retenue ou un emprunt lors d'opérations arithmétiques.

6.3 Conclusion - Visualiseur de Registres

Le visualiseur de registres est l'outil le plus important pour comprendre l'état du processeur :

- Les **registres principaux** (A, B, D, X, Y, S, U, PC) vous montrent où sont stockées les données et où se trouve l'exécution du programme
- Le **registre CCR** indique les conditions résultant des opérations (zéro, négatif, retenue, etc.)
- La **vue détaillée des 8 bits du CCR** permet de comprendre précisément quels drapeaux sont actifs
- Ensemble, ces informations vous donnent une image complète de l'état du processeur à chaque instant

Pendant l'exécution pas à pas, observer comment ces registres changent vous aide à comprendre exactement ce que fait chaque instruction.

7 Fenêtre de Débogage Pas à Pas

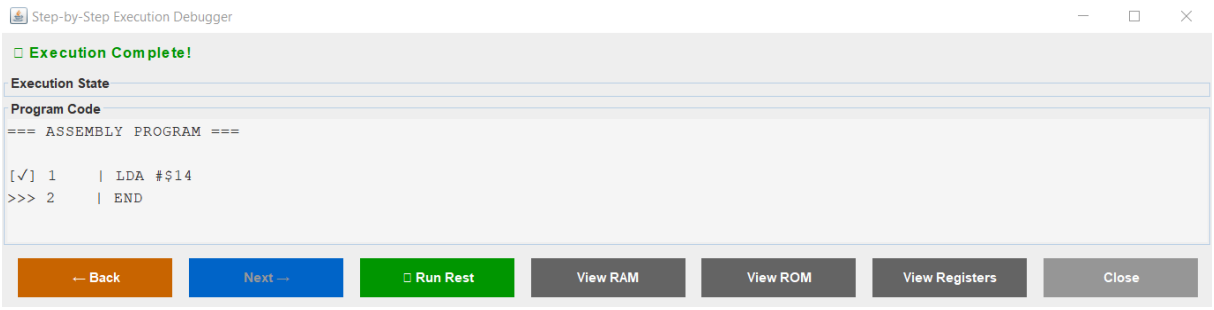

FIGURE 5 – Fenêtre de débogage pas à pas

Cette fenêtre s'ouvre lorsque vous activez le mode pas à pas en cliquant sur le bouton **Step** de la fenêtre principale. Elle vous permet d'exécuter votre programme instruction par instruction.

7.1 Message d'État

- **Affichage :** "Step debugger active" en vert
- **Fonction :** Confirmer que le mode débogage est actif
- **Description :** Indique le nombre d'instructions validées et combien sont prêtes à être exécutées. Par exemple : "2 instructions validées - Prêt à exécuter"

7.2 Section "Program Code"

- **Fonction :** Afficher le code source du programme
- **Description :** Montre toutes les instructions de votre programme avec leur numéro de ligne.
- **Indicateurs visuels :**
  - [] : Instruction déjà exécutée (coche verte)
  - »> : Instruction en cours d'exécution (flèche)
  - **Numéro de ligne** : Chaque instruction est numérotée
- **Utilisation :** Permet de suivre visuellement où vous en êtes dans l'exécution du programme.

7.3 Boutons de Contrôle du Débogueur

7.3.1 Bouton ← Back

- **Fonction :** Revenir en arrière d'une instruction
- **Description :** Annule la dernière instruction exécutée et restaure l'état précédent des registres et de la mémoire.
- **Utilisation :** Utile si vous voulez réexaminer ce qui s'est passé lors de l'instruction précédente.
- **Note :** Ce bouton peut être désactivé si vous êtes au début du programme.

7.3.2 Bouton Next →

- **Fonction :** Exécuter l'instruction suivante
- **Description :** Exécute une seule instruction et s'arrête. C'est le bouton principal du débogage pas à pas.
- **Utilisation :** Cliquez sur ce bouton à chaque fois que vous voulez avancer d'une instruction. Observez comment les registres et la mémoire changent après chaque clic.

7.3.3 Bouton Run Rest

- **Fonction :** Exécuter le reste du programme
- **Description :** Exécute toutes les instructions restantes jusqu'à la fin du programme (instruction END) sans s'arrêter.
- **Utilisation :** Utilisez ce bouton quand vous avez fini de déboguer une section critique et voulez terminer l'exécution rapidement.

7.3.4 Bouton View RAM

- **Fonction :** Ouvrir le visualiseur de RAM
- **Description :** Raccourci pour ouvrir la fenêtre de visualisation de la mémoire RAM pendant le débogage.
- **Utilisation :** Cliquez pour vérifier le contenu de la mémoire à tout moment pendant le débogage.

7.3.5 Bouton View ROM

- **Fonction :** Ouvrir le visualiseur de ROM
- **Description :** Raccourci pour ouvrir la fenêtre de visualisation de la mémoire ROM.
- **Utilisation :** Permet de vérifier le code programme en ROM pendant le débogage.

7.3.6 Bouton View Registers

- **Fonction :** Ouvrir le visualiseur de registres
- **Description :** Raccourci pour ouvrir la fenêtre détaillée des registres.
- **Utilisation :** Très utile pour observer en détail comment chaque registre change après chaque instruction.

7.3.7 Bouton Close

- **Fonction :** Fermer le débogueur
- **Description :** Ferme la fenêtre de débogage pas à pas et retourne à la fenêtre principale.
- **Utilisation :** Cliquez quand vous avez terminé le débogage.
- **Note :** L'état actuel du programme est conservé.

7.4 Conclusion - Débogueur Pas à Pas

Le débogueur pas à pas est l'outil le plus puissant pour comprendre et corriger vos programmes :

- L'**affichage du code** avec les indicateurs visuels vous montre où vous en êtes
- Les boutons **Back** et **Next** permettent de contrôler finement l'exécution
- Le bouton **Run Rest** offre une sortie rapide du mode débogage
- Les boutons **View** donnent un accès rapide aux visualiseurs de RAM, ROM et registres
- Le bouton **Close** permet de retourner à l'interface principale

En utilisant ce débogueur avec les visualiseurs de registres et de mémoire ouverts simultanément, vous pouvez suivre en temps réel l'effet de chaque instruction sur l'état complet du système.

8 Fenêtre de Sauvegarde

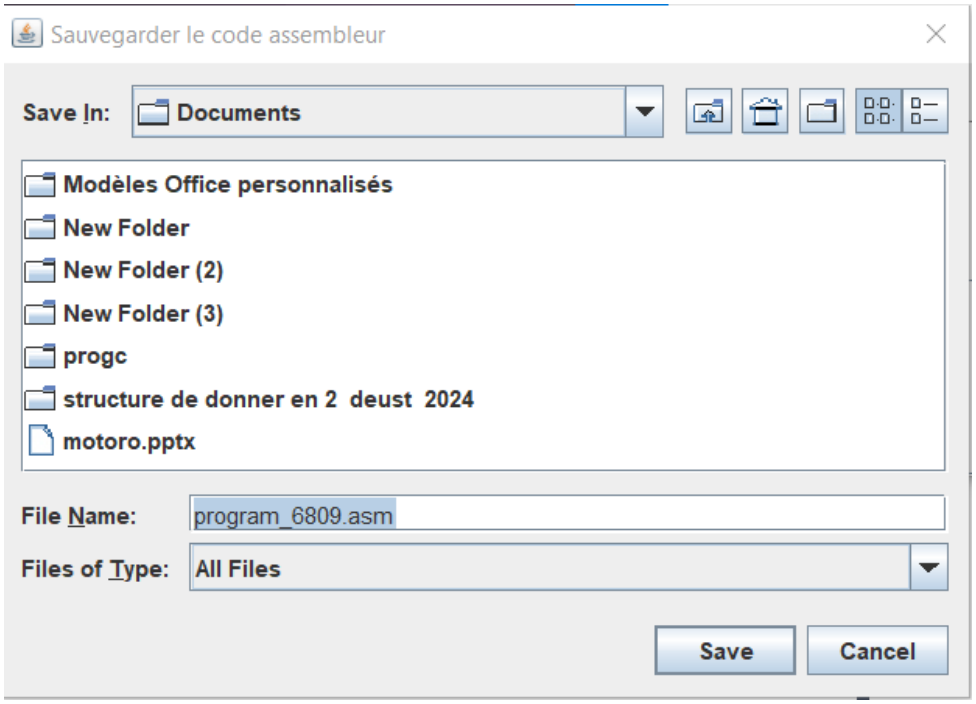

FIGURE 6 – Fenêtre de sauvegarde de fichier

Cette fenêtre s'ouvre lorsque vous cliquez sur le bouton **Save** de la fenêtre principale. Elle permet de sauvegarder votre code assembleur dans un fichier.

8.1 Section "Save In" (Enregistrer dans)

- **Fonction :** Sélectionner le dossier de destination
- **Description :** Menu déroulant montrant le dossier actuellement sélectionné (par défaut : "Documents").
- **Utilisation :** Cliquez sur le menu déroulant pour naviguer vers un autre dossier de votre ordinateur.

8.2 Barre d'Outils de Navigation

- **Icônes disponibles :**
  - Dossier récent
  - Bureau
  - Mes documents
  - Ordinateur
  - Réseau
- **Fonction :** Accès rapide aux emplacements fréquents
- **Utilisation :** Cliquez sur une icône pour naviguer rapidement vers cet emplacement.

8.3 Liste des Dossiers et Fichiers

- **Fonction :** Afficher le contenu du dossier sélectionné
- **Description :** Liste de tous les dossiers et fichiers dans l'emplacement actuel. Les dossiers sont affichés avec une icône de dossier, les fichiers avec leur icône respective.
- **Utilisation :** Double-cliquez sur un dossier pour y entrer. Vous pouvez également créer un nouveau dossier en cliquant avec le bouton droit.

8.4 Champ "File Name" (Nom de fichier)

- **Fonction :** Entrer le nom du fichier à sauvegarder
- **Description :** Champ de texte contenant le nom proposé pour le fichier (par exemple : "program_6809.asm").
- **Utilisation :**
  - Tapez le nom souhaité pour votre fichier
  - L'extension .asm est recommandée pour les fichiers assembleur
  - Évitez les caractères spéciaux dans le nom de fichier

8.5 Menu "Files of Type" (Type de fichier)

- **Fonction :** Sélectionner le type de fichier
- **Description :** Menu déroulant affichant "All Files" (Tous les fichiers) par défaut.
- **Utilisation :** Permet de filtrer l'affichage des fichiers par type. Gardez "All Files" pour une compatibilité maximale.

8.6 Boutons de Validation

8.6.1 Bouton Save

- **Fonction :** Confirmer la sauvegarde
- **Description :** Enregistre le fichier avec le nom et à l'emplacement spécifiés.
- **Utilisation :** Cliquez sur ce bouton après avoir choisi l'emplacement et le nom du fichier. Un message de confirmation apparaîtra dans le journal d'exécution.

8.6.2 Bouton Cancel

- **Fonction :** Annuler la sauvegarde
- **Description :** Ferme la fenêtre de sauvegarde sans enregistrer le fichier.
- **Utilisation :** Utilisez ce bouton si vous avez changé d'avis ou ouvert la fenêtre par erreur.

8.7 Conclusion - Fenêtre de Sauvegarde

La fenêtre de sauvegarde offre une interface complète pour gérer vos fichiers :

— Le **sélecteur de dossier** vous permet de choisir où enregistrer

— La **barre d'outils** offre un accès rapide aux emplacements courants

— Le **champ de nom** vous laisse nommer votre fichier

— Les boutons **Save** et **Cancel** valident ou annulent l'opération

Sauvegarder régulièrement votre travail vous permet de préserver vos programmes et de les recharger ultérieurement pour continuer votre développement.

9 Workflow Général d'Utilisation

Cette section explique comment utiliser le simulateur de manière cohérente pour créer, déboguer et exécuter un programme.

9.1 Étape 1 : Création d'un Nouveau Programme

1. Cliquez sur le bouton **New** dans la fenêtre principale
2. La fenêtre de l'éditeur de code s'ouvre
3. Écrivez votre programme assembleur dans la zone de texte
4. Terminez toujours par l'instruction **END**
5. Cliquez sur **Load Code** pour charger le programme

9.2 Étape 2 : Exécution et Débogage

Vous avez deux options :

9.2.1 Option A : Exécution Complète

1. Cliquez sur le bouton **Run**
2. Le programme s'exécute en entier
3. Consultez les résultats dans les visualiseurs (RAM, ROM, Registers)

9.2.2 Option B : Débogage Pas à Pas (Recommandé)

1. Cliquez sur le bouton **Step**
2. La fenêtre de débogage s'ouvre
3. Ouvrez les visualiseurs (RAM, Registers) pour suivre les changements
4. Cliquez sur **Next** pour exécuter chaque instruction une par une
5. Observez comment les registres et la mémoire changent
6. Utilisez **Back** si vous voulez revenir en arrière
7. Utilisez **Run Rest** pour finir l'exécution rapidement

9.3 Étape 3 : Analyse des Résultats

1. Ouvrez le **visualiseur de RAM** pour voir les données en mémoire
2. Ouvrez le **visualiseur de registres** pour voir l'état final des registres
3. Vérifiez le **registre CCR** pour comprendre les conditions finales
4. Consultez le **journal d'exécution** pour voir l'historique complet

9.4 Étape 4 : Sauvegarde du Programme

1. Cliquez sur le bouton **Save**
2. Choisissez l'emplacement de sauvegarde
3. Donnez un nom à votre fichier (extension .asm recommandée)
4. Cliquez sur **Save**
5. Vérifiez le message de confirmation dans le journal

9.5 Étape 5 : Recommencer ou Modifier

— Pour recommencer avec le même programme : Cliquez sur **Reset** puis sur **Run** ou **Step**

— Pour créer un nouveau programme : Cliquez sur **New**

— Pour arrêter une exécution en cours : Cliquez sur **Stop**

10 Conseils d'Utilisation

10.1 Pour les Débutants

— Commencez par des programmes simples (2-3 instructions)

— Utilisez toujours le mode **Step** pour comprendre chaque instruction

— Gardez les visualiseurs de **RAM** et de **Registers** ouverts pendant le débogage

— Observez comment le **PC (Program Counter)** s'incrémente à chaque instruction

— Vérifiez les **drapeaux du CCR** après les opérations arithmétiques

10.2 Pour les Utilisateurs Avancés

— Utilisez le bouton **Back** pour analyser les instructions problématiques

— Modifiez les valeurs en mémoire (double-clic) pour tester différents scénarios

— Utilisez les **registres d'index** (X, Y) pour les boucles complexes

— Surveillez les **pointeurs de pile** (S, U) lors des appels de sous-programmes

— Utilisez le bouton **Run Rest** après avoir vérifié les sections critiques

10.3 Bonnes Pratiques

— Sauvegardez votre travail régulièrement

— Utilisez des noms de fichiers descriptifs

— Commentez votre code assembleur pour mieux le comprendre

— Vérifiez toujours le journal d'exécution en cas d'erreur

— Testez vos programmes par étapes (instruction par instruction d'abord)

11 Dépannage

11.1 Problèmes Courants et Solutions

11.1.1 Le code ne se charge pas

— **Cause :** Erreur de syntaxe dans le code

— **Solution :** Vérifiez que chaque instruction est correcte et que le programme se termine par **END**

11.1.2 Le programme ne s'exécute pas

— **Cause :** Aucun code chargé

— **Solution :** Cliquez sur **New** et chargez un programme avec **Load Code**

11.1.3 Les registres ne changent pas

— **Cause :** Le programme n'a pas été exécuté ou le visualiseur n'est pas à jour

— **Solution :** Exécutez le programme et actualisez le visualiseur de registres

11.1.4 Impossible de modifier les valeurs en mémoire

— **Cause :** Simple clic au lieu de double-clic

— **Solution :** Double-cliquez sur la valeur pour l'éditer

11.1.5 Le fichier ne se sauvegarde pas

— **Cause :** Emplacement protégé en écriture ou nom de fichier invalide

— **Solution :** Choisissez un emplacement accessible (Documents) et évitez les caractères spéciaux

12 Conclusion

Le Simulateur Motorola 6809 est un outil complet qui vous permet d'apprendre et de maîtriser la programmation en assembleur. En utilisant ses différentes fenêtres et fonctionnalités de manière coordonnée, vous pouvez :

— Écrire des programmes assembleur avec l'éditeur de code

— Exécuter et déboguer vos programmes instruction par instruction

— Visualiser en temps réel l'état des registres et de la mémoire

— Comprendre comment fonctionne réellement un microprocesseur

— Sauvegarder et gérer vos projets efficacement

Ce guide vous a présenté chaque composant de l'interface. N'hésitez pas à expérimenter et à explorer les différentes fonctionnalités. La pratique est la meilleure façon de maîtriser le simulateur et la programmation assembleur.

Bonne programmation avec le Simulateur Motorola 6809 !