Royaume du Maroc

Université Hassan Premier – Settat
Faculté des Sciences et Techniques de Settat

GUIDE MOTO6809

présente la démarche d'utilisation du simulateur MOTO6809

Encadré par :

Pr. Hicham BENALLA

Réalisé par :

Yasmine Naam

Salma Rachid

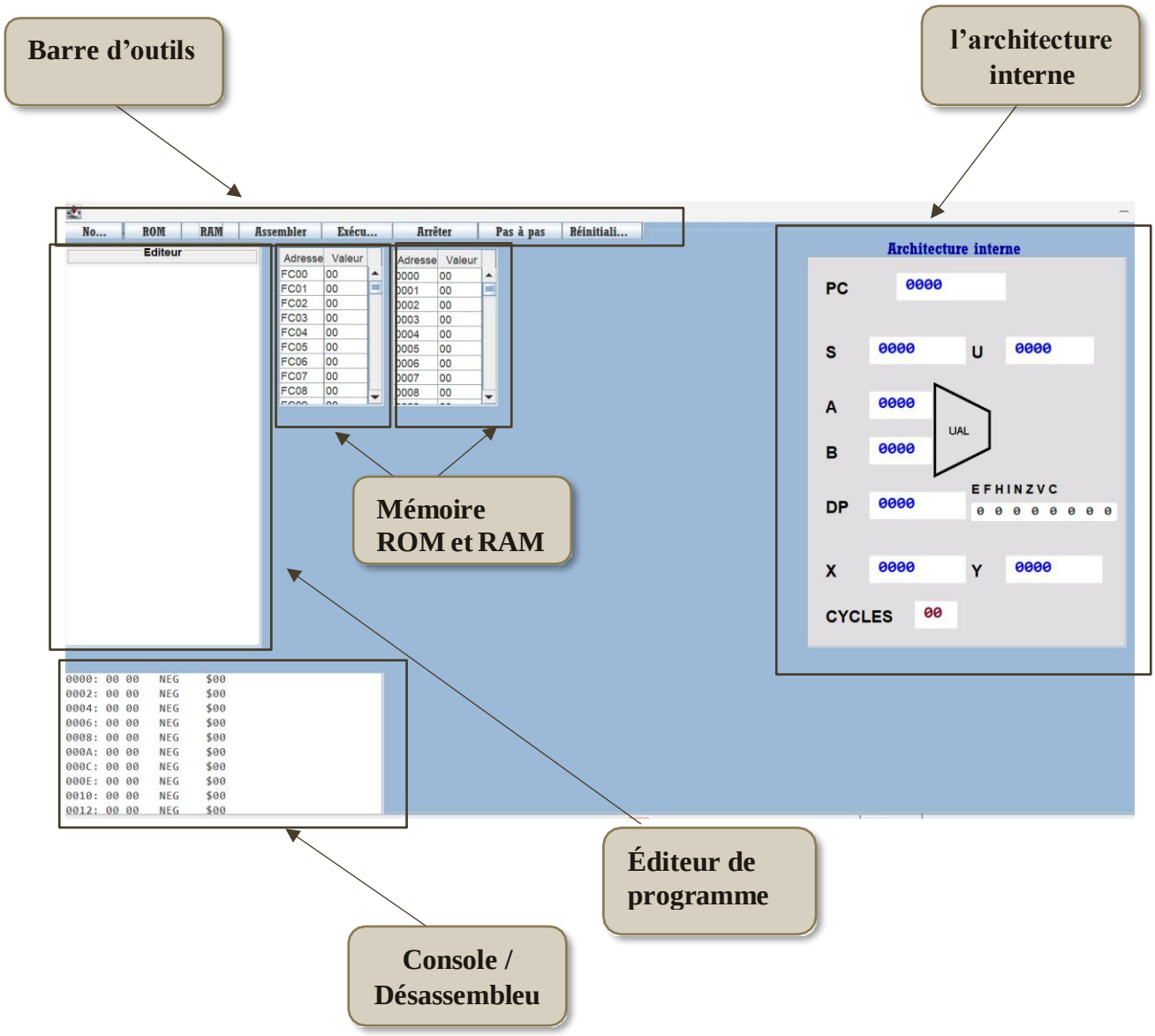
Sommaire

□ Présentation du simulateur Motorola 6809

Le **simulateur du Motorola 6809** est un outil logiciel conçu pour reproduire de manière fidèle le fonctionnement du microprocesseur Motorola 6809**.** Il permet de simuler l'exécution des programmes assembleur en reproduisant les principaux composants du processeur, tels que les registres, la mémoire**,** le compteur ordinal (PC) et les codes de condition**.**

Ce simulateur offre un environnement pédagogique permettant d'observer, étape par étape, l'exécution des instructions, l'évolution de l'état interne du processeur et les interactions avec la mémoire. Il constitue ainsi un support essentiel pour la compréhension de l'architecture interne du Motorola 6809 et du fonctionnement des microprocesseurs en général.

L'utilisation d'un simulateur présente l'avantage de faciliter l'apprentissage sans nécessiter de matériel physique, tout en offrant des fonctionnalités avancées telles que le chargement de programmes**,** le désassemblage**,** le débogage et la visualisation graphique des registres et de la mémoire.



□ Description de l'interface

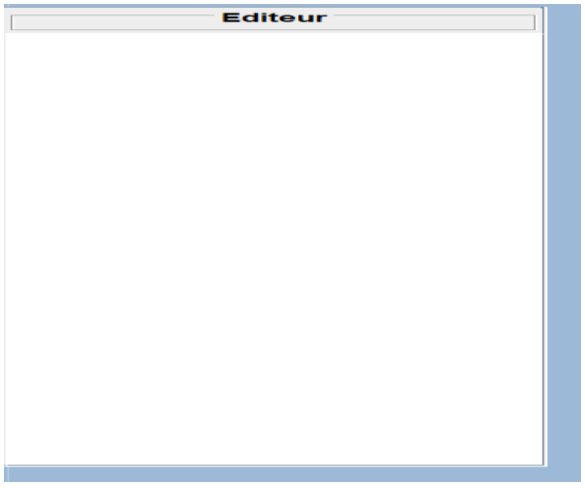
➢ *Barre d'outils*

- ✓ **Nouveau** : permet de créer un nouveau programme assembleur dans l'éditeur.
- ✓ **ROM** : affiche le contenu de la mémoire ROM.
- ✓ **RAM** : affiche le contenu de la mémoire RAM.

- ✓ **Assembler** :Le bouton Assembler permet de traduire le programme écrit en langage assembleur Motorola 6809 en code machine (opcodes)**.** Cette étape vérifie également la syntaxe du programme et charge les instructions traduites dans la mémoire ROM**.**

- ✓ **Exécuter** : Lance l'exécution du programme de manière continue jusqu'à la fin ou jusqu'à l'arrêt manuel.
- ✓ **Arrêter** : interrompt l'exécution en cours . Pour stopper une exécution continue ou en cas de boucle infinie.
- ✓ **Pas à pas** : exécute le programme instruction par instruction afin d'analyser son comportement. Pour analyser en détail le fonctionnement du programme, suivre l'évolution des registres et comprendre chaque instruction.
- ✓ **Réinitialiser** : Réinitialise le processeur, les registres et la mémoire à leurs valeurs initiales.
  Avant de relancer un programme ou pour remettre le simulateur dans un état propre.

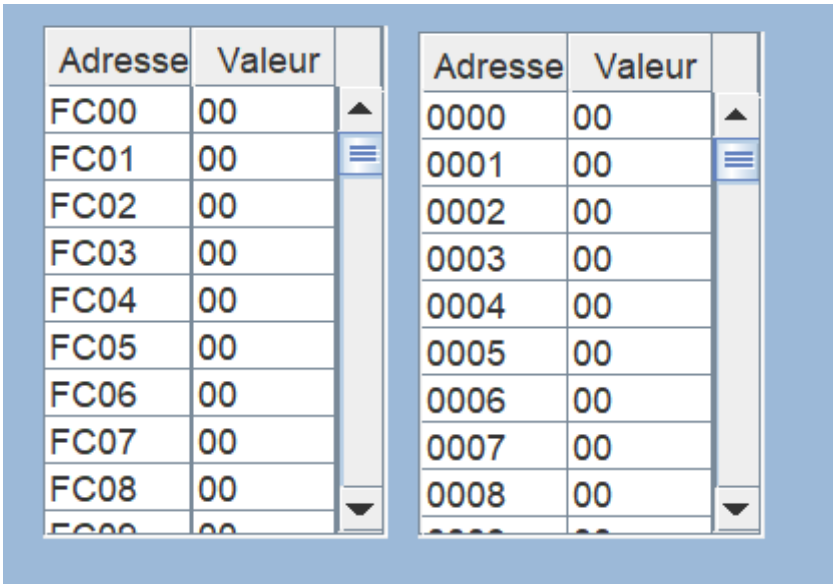
➢ ***Éditeur de programme***



- L'éditeur de programme est l'espace réservé à l'écriture et à la modification du code assembleur du microprocesseur Motorola 6809. Il permet à l'utilisateur de préparer le programme avant son assemblage.

- Cette zone est utilisée lors de la création d'un nouveau programme ou lors de la correction d'erreurs, et constitue une étape essentielle avant l'exécution du programme.

➢ ***Mémoire ROM et RAM***

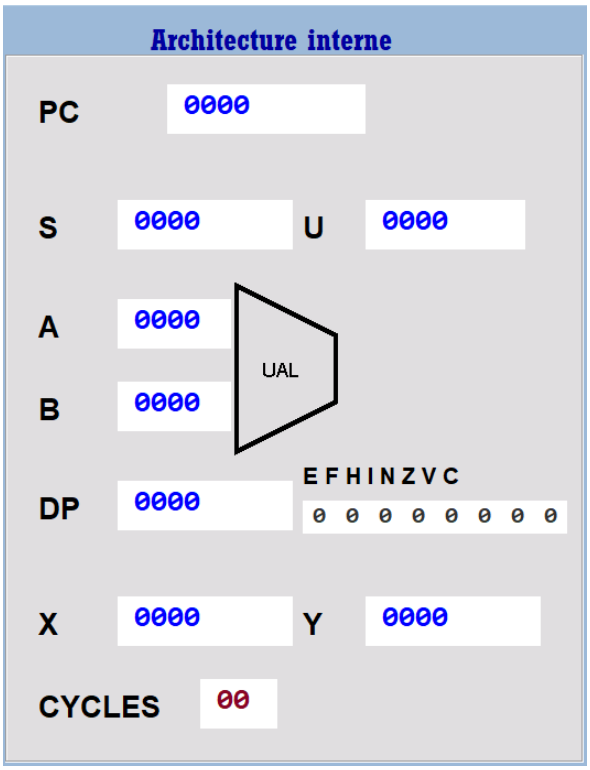


- La **mémoire ROM** (Read Only Memory) contient le programme assembleur après sa traduction en code machine. Elle permet de visualiser les instructions exécutées par le processeur et reste inchangée durant l'exécution.
- La **mémoire RAM** (Random Access Memory) est utilisée pour stocker les données et les résultats intermédiaires pendant l'exécution du programme. Son contenu peut évoluer au cours du fonctionnement du simulateur.

→L'affichage des mémoires ROM et RAM permet à l'utilisateur de suivre l'organisation des données en mémoire et de mieux comprendre les interactions entre le processeur et la mémoire.

➢ ***Architecture interne***

La zone **Architecture interne** permet de visualiser en temps réel l'état du microprocesseur Motorola 6809. Elle affiche les principaux registres du processeur,



- ***Les registres :***
  - **Accumulateur A et B :** Les registres A et B servent des accumulateurs, c'est à-dire qu'ils sont utilisés pour effectuer des opérations arithmétiques et logiques. Ce sont de taille 8 bits et pouvant être concaténés pour former le registre D de 16 bits.
  - **DP (Pointeur de Page) :** Le registre DP, ou Pointeur de Page, est utilisé pour indiquer la page mémoire actuellement active. Cela peut être crucial lors de l'accès à des emplacements mémoire spécifiques.
  - **S, U (Pointeurs de Piles) :** Leur taille est de 16 bits. Elles permettent l'implantation de deux piles parfaitement distinctes. L'une est utilisée par défaut par les instructions de branchement à sous-programme, l'autre est géré par l'utilisateur pour des passages de paramètres par exemple.
  - **X, Y (Registres Généraux) :** Les registres X et Y sont des registres généraux, souvent utilisés pour stocker des données temporaires ou des adresses mémoire.

- ***Les flags :***
  - **Z (Zéro) :** Cet indicateur est défini si le résultat d'une opération est zéro
  - **C (Carry) :** L'indicateur de retenue est utilisé lors d'opérations arithmétiques pour indiquer un débordement ou une retenue.

- **N (Négative)** : L'indicateur négatif indique si le résultat d'une opération est négatif.
- **V (Overflow)** : L'indicateur de dépassement est défini en cas de débordement lors d'opérations arithmétiques signées.

- ***CYCLES :***

- Le champ CYCLES indique le nombre de cycles d'horloge consommés par le processeur lors de l'exécution des instructions. Chaque instruction du Motorola 6809 nécessite un certain nombre de cycles pour être exécutée, en fonction de sa complexité et de son mode d'adressage.

➢ ***Console / Désassembleur***

```
0000: 00 00   NEG    $00
0002: 00 00   NEG    $00
0004: 00 00   NEG    $00
0006: 00 00   NEG    $00
0008: 00 00   NEG    $00
000A: 00 00   NEG    $00
000C: 00 00   NEG    $00
000E: 00 00   NEG    $00
0010: 00 00   NEG    $00
0012: 00 00   NEG    $00
```

- La console (ou désassembleur) affiche le programme présent en mémoire sous une forme lisible par l'utilisateur. Elle présente, pour chaque instruction**,** l'adresse mémoire, le code machine (opcodes) ainsi que l**'**instruction assembleur correspondante**.**
- Cette zone permet de suivre l'exécution du programme et de comprendre la correspondance entre le code machine et les instructions assembleur. Elle est particulièrement utile lors de l'exécution en mode pas à pas, car elle facilite l'analyse du comportement du processeur instruction par instruction.

□ Liste des instructions :

1. LOAD / STORE

- **LDA/LDB/LDX / LDU** :ces instructions permettent de charger des valeurs dans les registres **A**, **B**, **X** et **U** du microprocesseur Motorola 6809.
  Elles peuvent être utilisées avec les modes d'adressage imm/imm16 direct, indexé et étendu.
- **STA/STB/ STX / STU** permettent de stocker le contenu des registres A**,** B**,** X et U en mémoire.
  Elles peuvent être utilisées avec les modes d'adressage **direct**, **indexé** et étendu.

2. ARITHMÉTIQUE

- **ADDA / ADDB** : addition à l'accumulateur A ou B
  (immédiat, direct, indexé, étendu)
- **SUBA / SUBB** : soustraction à l'accumulateur A ou B
  (immédiat, direct, indexé, étendu)
- **INC** : incrémentation d'une valeur en mémoire
  (direct, indexé)
- **DEC** : décrémentation d'une valeur en mémoire
  (direct, indexé, étendu)
- **DECA /DECB/ INCA/ INCB**
- **CLR** : mise à zéro d'une valeur en mémoire
  (direct, indexé, étendu)
- ***CLRA /CLRB***
- **NEG** : négation (complément à deux) d'une valeur en mémoire
  (direct, indexé, étendu)

3. LOGIC

- **ANDA / ANDB** : opération logique **ET** avec l'accumulateur A ou B
  (immédiat, direct, indexé, étendu)
- **ORA** : opération logique **OU** avec l'accumulateur A
  (immédiat, direct, indexé)
- **ORB** : opération logique **OU** avec l'accumulateur B
  (immédiat, direct, indexé, étendu)
- **EORA / EORB** : opération logique **OU exclusif (XOR)** avec l'accumulateur A ou B
  (EORA : immédiat ; EORB : immédiat, direct, indexé, étendu)

4. BRANCH / JUMP / CALL

Ces instructions permettent de modifier le déroulement normal du programme.

- **JMP** : effectue un saut inconditionnel vers une adresse donnée
  (direct, indexé, étendu)
- **JSR** : appelle une sous-routine en sauvegardant l'adresse de retour sur la pile
  (direct, indexé, étendu)
- **RTS** : retourne de la sous-routine en restaurant l'adresse sauvegardée

5. STACK OPERATIONS

Ces instructions permettent la manipulation des piles du processeur.

- **PSHS / PULS** : empilent et dépilent des données sur la pile système **S**
- **PSHU / PULU** : empilent et dépilent des données sur la pile utilisateur **U**

6. AUTRE

- **NOP** : instruction sans effet, utilisée pour temporiser ou aligner le code

□ Bonnes pratiques

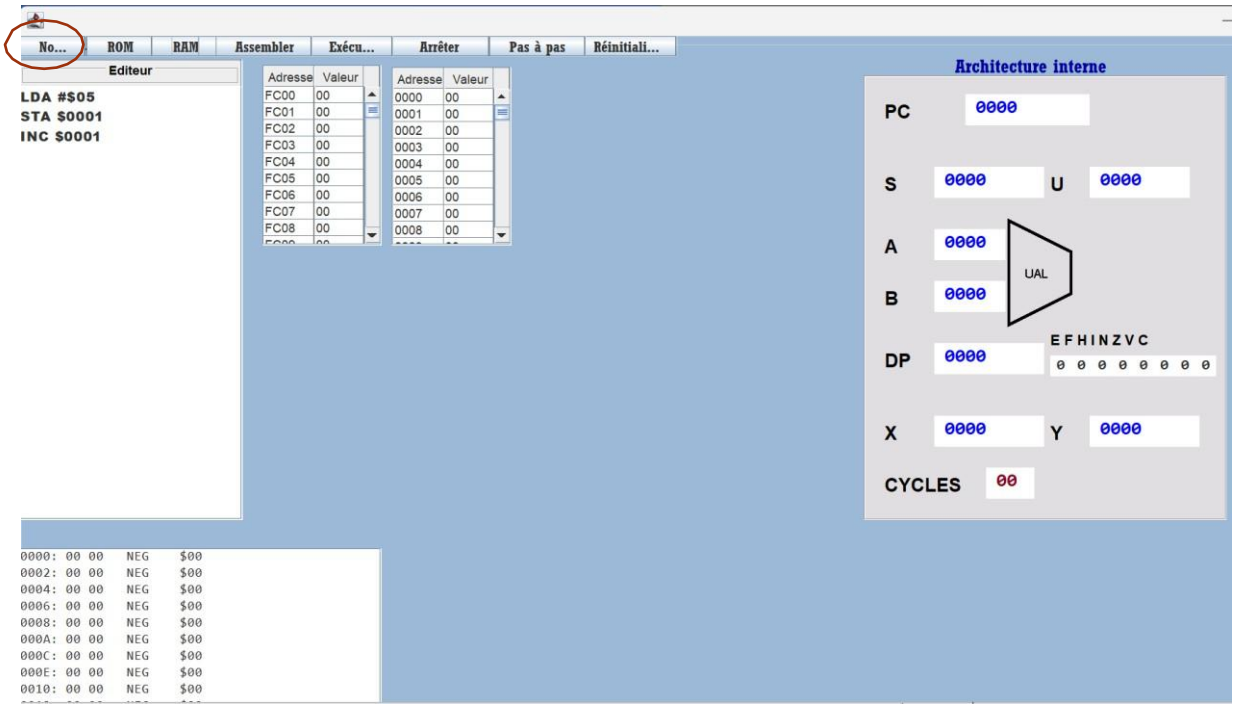
- ✓ Toujours assembler le programme avant l'exécution
- ✓ Utiliser le mode pas à pas pour comprendre le fonctionnement des instructions
- ✓ Vérifier les registres et les flags a
- ✓ près chaque instruction

□ Exemples de Simulation de Logiciel :

➢ Démarche d'utilisation

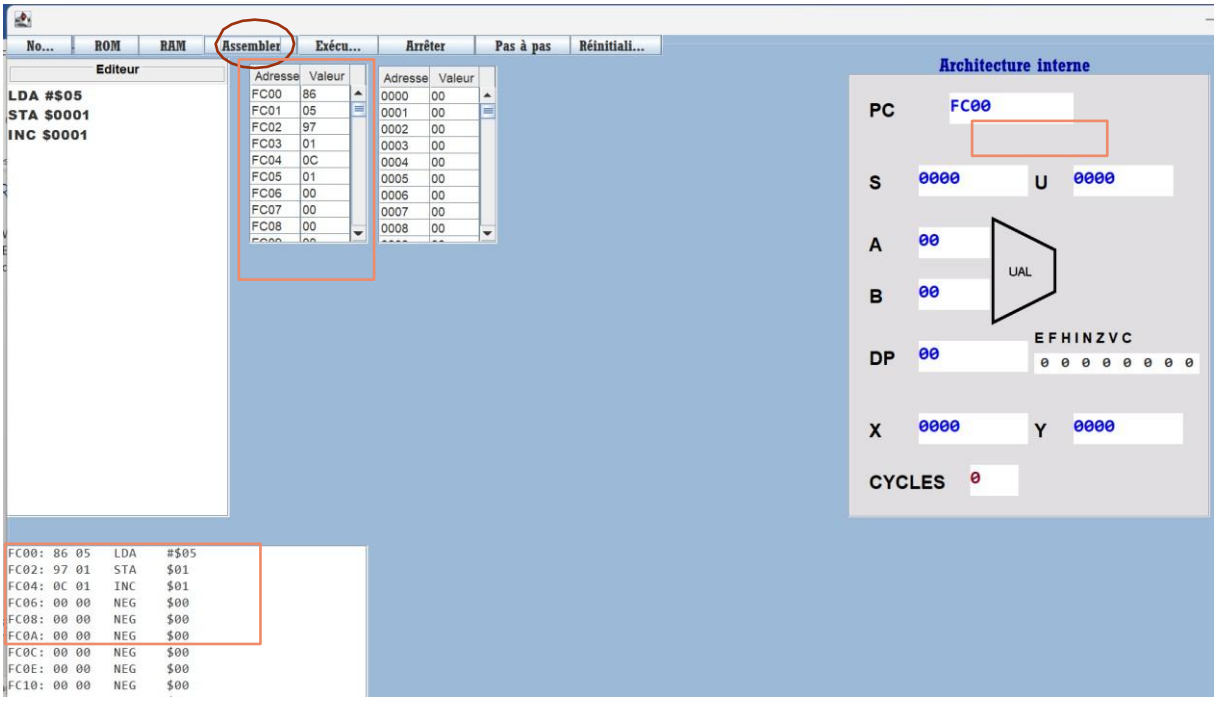
Étape 1 : Création du programme

Cliquer sur **Nouveau** puis saisir le programme assembleur dans l'éditeur.



Étape 2 : Assemblage

Cliquer sur **Assembler** pour convertir le programme assembleur en opcodes et le charger en mémoire ROM.



Après avoir cliqué sur Assembler, le programme est traduit en opcodes et chargé en mémoire ROM. Le désassembleur affiche alors les instructions correspondantes, telles que

```
FC00: 86 05 LDA #$05
```

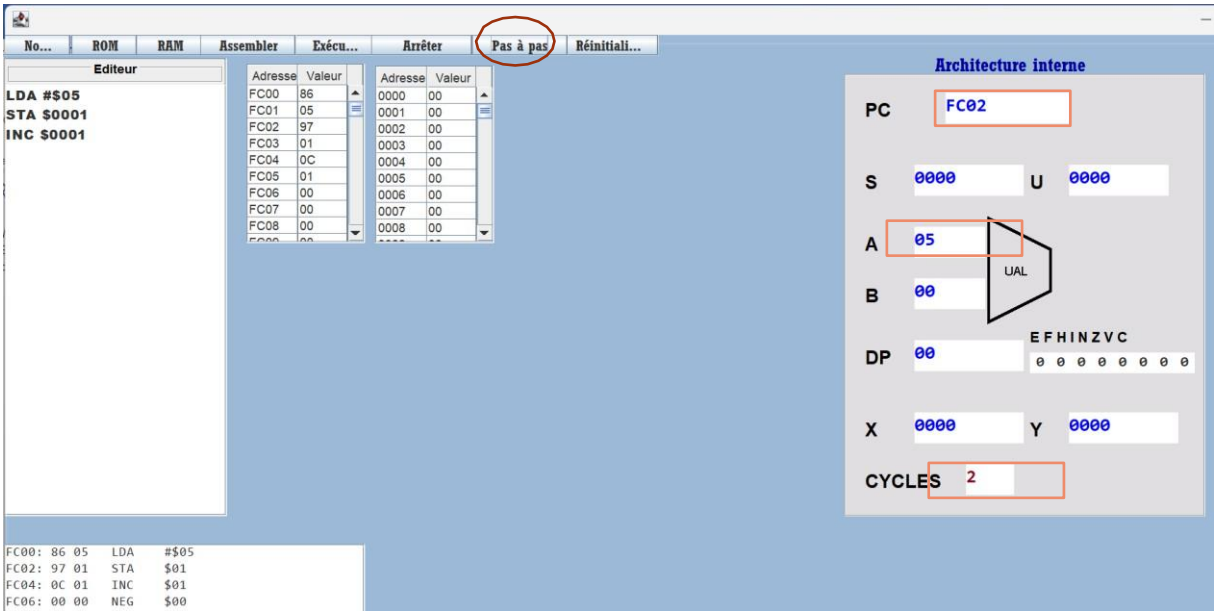
```
FC02: 97 01 STA $01

FC04: 0C 01 INC $01.
```

Chaque instruction est ainsi représentée par son adresse mémoire, son code machine en hexadécimal et son mnémotechnique assembleur, ce qui permet de vérifier la correspondance entre le code source et son exécution par le processeur.

Étape 3 : Exécution

- Cliquer sur **Exécuter** pour une exécution continue
- Soit cliquer sur **Pas à pas** pour une exécution instruction par instruction

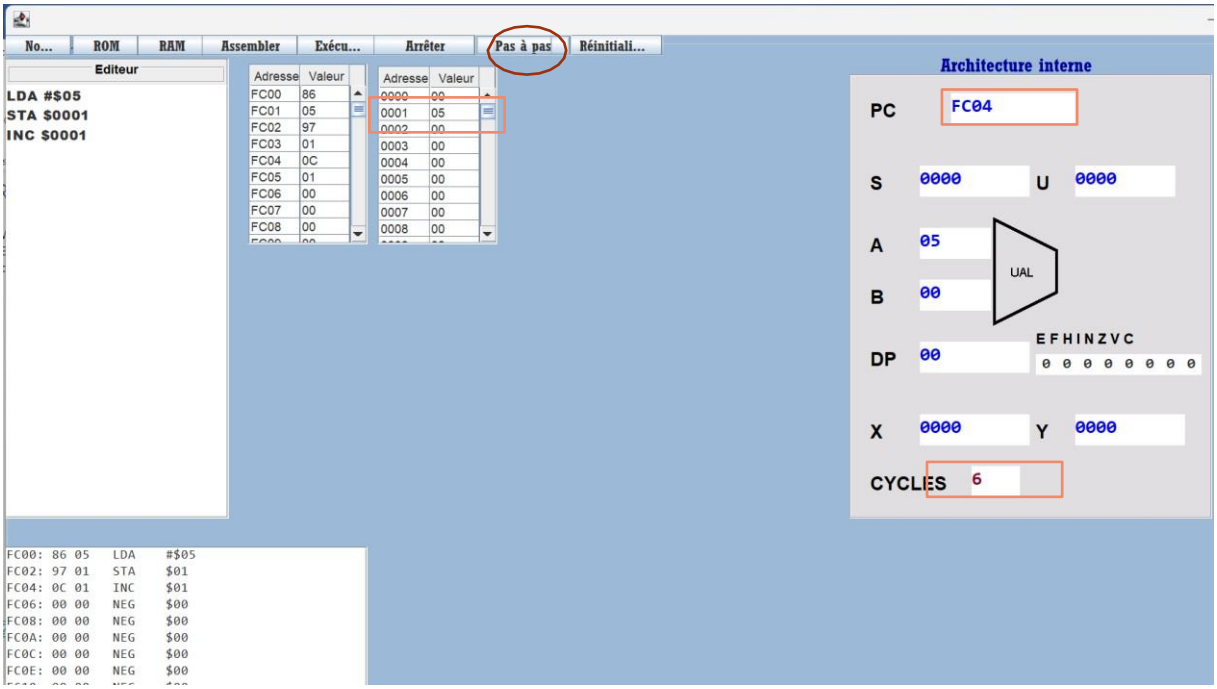
➢ LDA #$05



- Rôle : charge la valeur immédiate `05` dans l'accumulateur **A**.
- Effet observé **:**
  - A = 05
  - `PC` avance à l'instruction suivante

- Les autres registres restent inchangés

➢ STA $0001

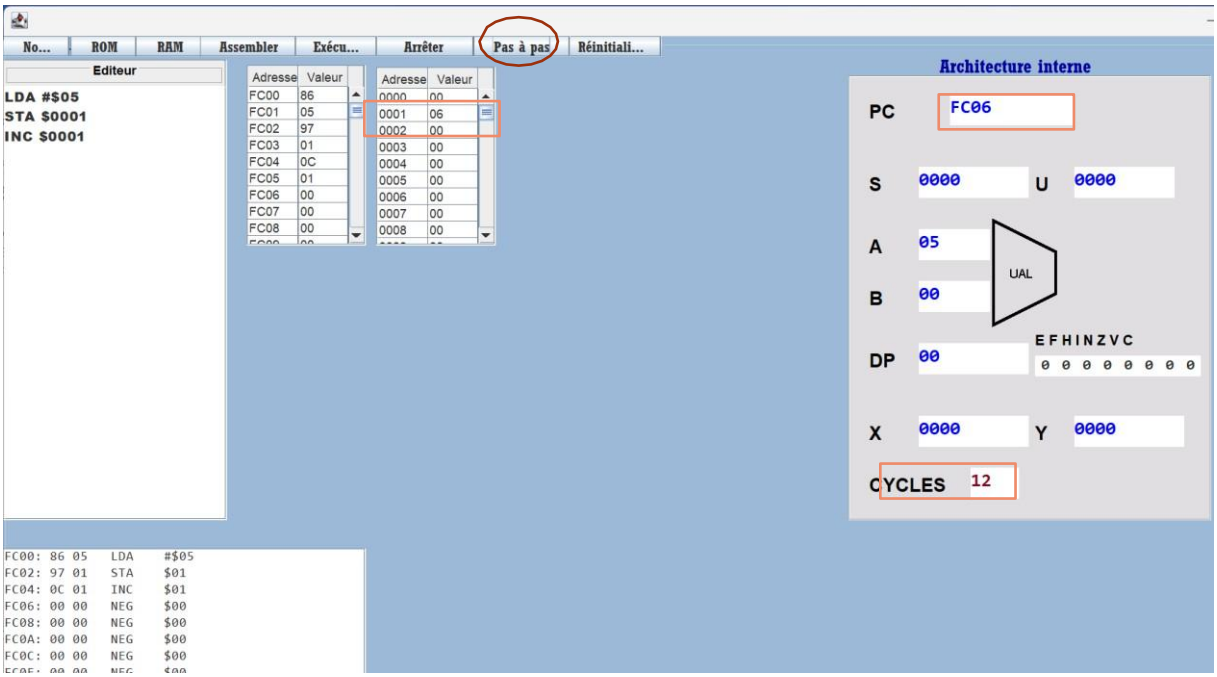


• Rôle : stocke le contenu de l'accumulateur **A** en mémoire à l'adresse `$0001`.

• Effet observé :

- Mémoire RAM à l'adresse **$**0001 = 05
- L'accumulateur **A** reste inchangé

➢ INC $0001



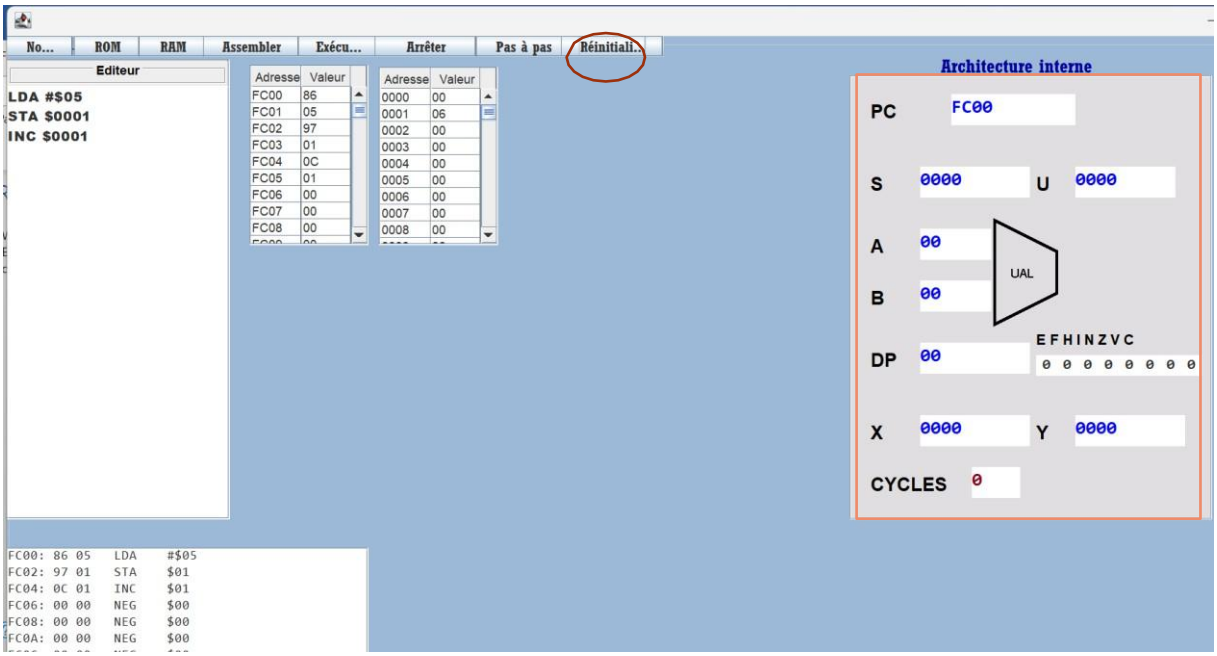
• Rôle : incrémente de 1 la valeur stockée à l'adresse `$0001`.

• Effet observé :

- Mémoire RAM à l'adresse $0001 = 06
- Mise à jour éventuelle des flags (Z, N, …)
- Le PC pointe vers l'instruction suivante

Étape 5 : Arrêt ou réinitialisation

- **Arrêter** : stoppe l'exécution
- **Réinitialiser** : remet le simulateur à l'état initial



□ Conclusion du guide d'utilisation

Ce guide d'utilisation a permis de présenter les principales fonctionnalités du simulateur du microprocesseur Motorola 6809 ainsi que la démarche à suivre pour assembler et exécuter un programme en langage assembleur. À travers la description de l'interface, des commandes disponibles et des exemples pratiques, l'utilisateur peut comprendre le rôle des différents composants du processeur et leurs interactions avec la mémoire.

L'utilisation du mode pas à pas, de l'affichage de l'architecture interne et du désassembleur constitue un atout pédagogique majeur, facilitant l'analyse du fonctionnement interne du processeur et l'apprentissage de la programmation assembleur. Ce simulateur représente ainsi un outil efficace pour l'initiation, l'expérimentation et la compréhension du fonctionnement des microprocesseurs.