



Guide d'Utilisation - Simulateur Motorola 6809

Bienvenue dans le guide complet de votre environnement de développement pour le microprocesseur 6809.

Réalisé par : CHAHD BOUKHRAISS , HAJAR GOUMARIR

Table des matières

01

Introduction et Architecture

- 1. Introduction*
- 2. Architecture du Simulateur*

03

Visualisation et Édition

- 6. Fenêtres de Visualisation*
- 7. Éditeur de Code*

02

Interface et Contrôles

- 3. Interface Principale*
- 4. Barre de Menus*
- 5. Boutons de Commande*

04

Programmation

- 8. Syntaxe du Langage Assembleur*
- 9. Jeu d'Instructions*
- 10. Architecture Interne du CPU*

1. Introduction

Le Simulateur Motorola 6809 est un environnement de développement intégré permettant l'assemblage, l'exécution et le débogage de programmes en langage assembleur pour le microprocesseur Motorola 6809. Ce simulateur offre une visualisation en temps réel de l'état du processeur, de la mémoire ROM/RAM et du flux d'exécution.

Objectifs pédagogiques :

- *Comprendre l'architecture du Motorola 6809*
- *Maîtriser le langage assembleur 6809*
- *Analyser l'exécution instruction par instruction*
- *Observer les modifications des registres et de la mémoire*



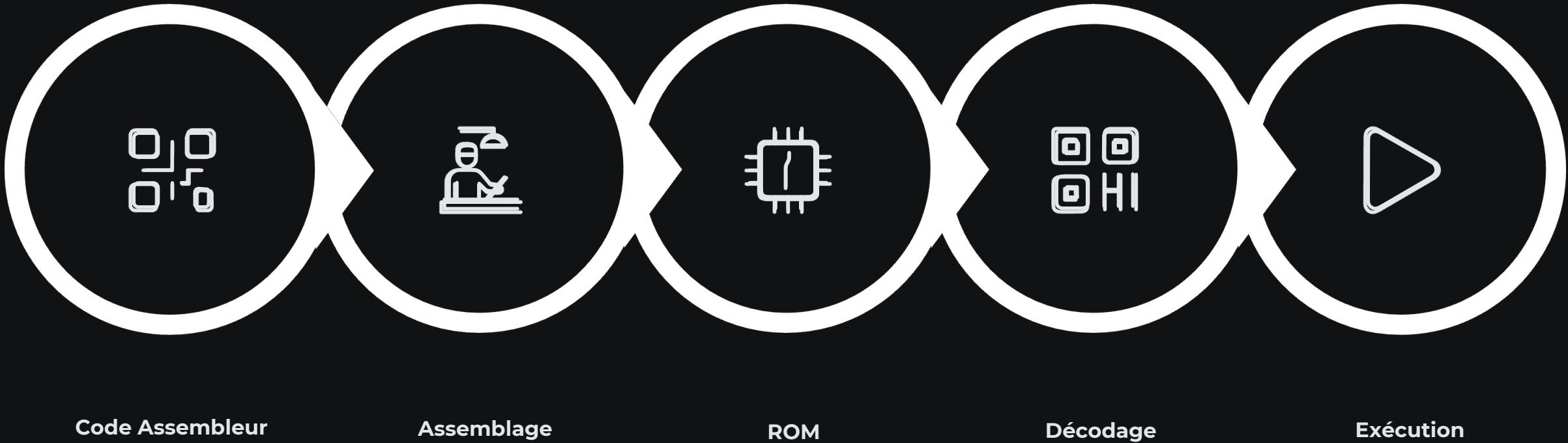
2. Architecture du Simulateur

2.1 Composants principaux

Le simulateur est structuré autour de plusieurs modules :

Module	Classe	Fonction
CPU	CPU.java	Émulation des registres et flags
Mémoire ROM	ROM.java	Stockage du code machine (\$FC00-\$FFFF)
Mémoire RAM	RAM.java	Stockage des données (\$0000-\$03FF)
Décodeur	InstructionDecoder.java	Analyse syntaxique des instructions
Exécuteur	InstructionExecutor.java	Exécution des opcodes
Gestionnaire	ProgramManager.java	Contrôle du flux d'exécution

2.2 Flux d'exécution



ZONE 1

ZONE 4

Assembler

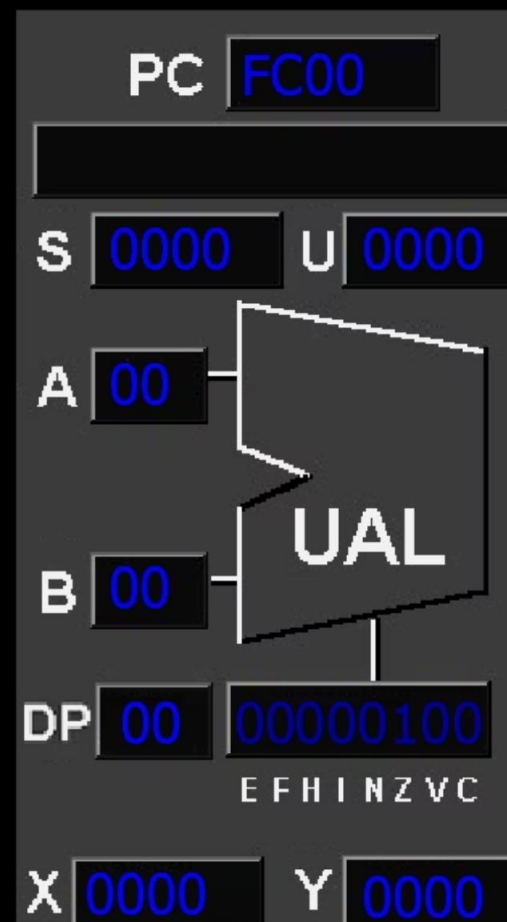
Exécuter

Pas à Pas

Reset

Editeur - Code Assembleur 6809

Architecture Interne du CPU Motorola 6809



ZONE 2

ZONE 3

Sortie Console

Console Système 6809 - Prêt à simuler

3. Interface Principale

L'interface est divisée en quatre zones principales :

1

Zone 1 : Éditeur de Code

- *Écriture du programme assembleur*
- *Coloration syntaxique*
- *Numérotation des lignes*

2

Zone 2 : Architecture CPU

- *Visualisation des registres (PC, A, B, X, Y, S, U, DP)*
- *État des flags (E, F, H, I, N, Z, V, C)*
- *Représentation graphique de l'UAL*

3

Zone 3 : Console

- *Messages système*
- *Résultats d'assemblage*
- *Erreurs et avertissements*

4

Zone 4 : Barre de Commandes

- *Boutons d'action rapide*
- *Accès aux fenêtres auxiliaires*

4. Barre de Menus

4.1 Menu Fichier

Fichier	Visualisation	Aide
Nouveau		Ctrl-N
Ouvrir...		Ctrl-O
Enregistrer		Ctrl-S
Charger (Chemin ou URL)		
Imprimer...		Ctrl-P
Quitter		Ctrl-Q

4.2 Menu Visualisation

- *ROM: \$FC00 - \$FFFF (Code machine)*
- *RAM: \$0000 - \$03FF (Données)*
- *Programme: Instructions assemblées*

Visualisation	Aide
ROM (Mémoire Programme)	
RAM (Mémoire Données)	
Programme (Instructions Assemblées)	

4.3 Menu Aide

- *À propos : Informations sur le projet et les auteurs*

Aide
À propos

5. Boutons de Commande

Assembler

5.1 Assembler

Convertit le code assembleur en code machine hexadécimal. Analyse syntaxique, génération d'opcodes et chargement en ROM à partir de \$FC00.

Exécuter

5.2 Exécuter

Exécute le programme complet depuis PC jusqu'à END ou SWI. Mise à jour en temps réel de l'architecture CPU.

Pas à Pas

5.3 Pas à Pas

Exécute une instruction à la fois. Permet d'observer les modifications précises dans l'architecture CPU et la fenêtre Programme.

Reset

5.4 Reset

Réinitialise le CPU : PC = \$FC00, Registres = \$0000, CC = \$04 (Flag Z activé).

*Messages console : ✓ Assemblage terminé avec succès
✓ Programme chargé: 15 lignes Total: 42 octets en ROM*

6. Fenêtres de Visualisation

6.1 Fenêtre ROM

Plage : \$FC00 - \$FFFF (1024 octets). Format : Adresse | Donnée (hex).

Couleur : Orange.

FC00	86; LDA #
FC01	05; Donnée

Adresse	Donnée
FC00	FF
FC01	FF
FC02	FF
FC03	FF
FC04	FF
FC05	FF
FC06	FF
FC07	FF
FC08	FF
FC09	FF
FC0A	FF
FC0B	FF
FC0C	FF
FC0D	FF
FC0E	FF


0xFC00 → 0xFFFF • 1024 bytes

6.2 Fenêtre RAM

Plage : \$0000 - \$03FF. Modification manuelle possible par double-clic.

Couleur : Vert.

Adresse	Donnée
0000	00
0001	00
0002	00
0003	00
0004	00
0005	00
0006	00
0007	00
0008	00
0009	00
000A	00
000B	00
000C	00
000D	00
000E	00


0x0000 → 0x03FF • 1024 bytes

6.3 Fenêtre Programme

Affiche l'adresse d'exécution et l'instruction complète avec un surlignage

vert de la ligne en cours. Synchronisation automatique avec le PC.

Adresse	Instruction
<div>  Instructions assemblées </div>	

7. Éditeur de Code

7.1 Fonctionnalités

- *Police* : JetBrains Mono, taille 15
- *Thème* : Fond sombre (RGB: 30, 35, 45)
- *Coloration* : Texte clair (RGB: 220, 220, 220)
- *Tabulation* : 4 espaces
- *Numérotation* : Lignes automatiques

7.2 Raccourcis clavier

Ctrl+S	Enregistrer
Ctrl+Z	Annuler
Ctrl+Y	Rétablir
Ctrl+F	Rechercher

Éditeur - Code Assembleur 6809

8. Syntaxe du Langage Assembleur

8.1 Structure d'une ligne

```
[ETIQUETTE:] INSTRUCTION [OPERANDE] [; COMMENTAIRE]
```

8.2 Directives

ORG \$FC00 (Départ), END (Fin), FCB (Constant Byte), FDB (Constant Word)

8.3 Modes d'adressage

- *Immédiat (#\$XX): LDA #\$05*
- *Direct (<\$XX): LDA <\$20*
- *Étendu (>\$XXXX): LDA >\$3000*
- *Indexé (n,R): LDA 5,X*

8.4 Adressage indexé avancé

Supporte l'auto-incrémentation (,X+), auto-décrémentation (,-X), accumulateur (A,X) et l'indirect ([,X]).

8.5 Étiquettes

Doivent commencer par une lettre ou '_', terminées par ':' (optionnel). Sensibles à la casse (converties en MAJUSCULES).

9. Jeu d'Instructions

9.1 Chargement/Stockage

LDA	IMM, DIR, EXT, IDX	N, Z, V=0	Charger A
LDB	IMM, DIR, EXT, IDX	N, Z, V=0	Charger B
LDD	IMM, DIR, EXT, IDX	N, Z, V=0	Charger D (A:B)
LDX	IMM, DIR, EXT, IDX	N, Z, V=0	Charger X
LDY	IMM, DIR, EXT, IDX	N, Z, V=0	Charger Y
LDU	IMM, DIR, EXT, IDX	N, Z, V=0	Charger U
LDS	IMM, DIR, EXT, IDX	N, Z, V=0	Charger S
STA	DIR, EXT, IDX	N, Z, V=0	Stocker A
STB	DIR, EXT, IDX	N, Z, V=0	Stocker B
STD	DIR, EXT, IDX	N, Z, V=0	Stocker D
STX	DIR, EXT, IDX	N, Z, V=0	Stocker X
STY	DIR, EXT, IDX	N, Z, V=0	Stocker Y
STU	DIR, EXT, IDX	N, Z, V=0	Stocker U
STS	DIR, EXT, IDX	N, Z, V=0	Stocker S

9.2 Arithmétique

<i>Instruction</i>	<i>Modes</i>	<i>Flags</i>	<i>Description</i>
<i>ADDA</i>	<i>IMM, DIR, EXT, IDX</i>	<i>H, N, Z, V, C</i>	<i>A = A + M</i>
<i>ADDB</i>	<i>IMM, DIR, EXT, IDX</i>	<i>H, N, Z, V, C</i>	<i>B = B + M</i>
<i>ADDD</i>	<i>IMM, DIR, EXT, IDX</i>	<i>N, Z, V, C</i>	<i>D = D + M</i>
<i>ADCA</i>	<i>IMM, DIR, EXT, IDX</i>	<i>H, N, Z, V, C</i>	<i>A = A + M + C</i>
<i>ADCB</i>	<i>IMM, DIR, EXT, IDX</i>	<i>H, N, Z, V, C</i>	<i>B = B + M + C</i>
<i>SUBA</i>	<i>IMM, DIR, EXT, IDX</i>	<i>N, Z, V, C</i>	<i>A = A - M</i>
<i>SUBB</i>	<i>IMM, DIR, EXT, IDX</i>	<i>N, Z, V, C</i>	<i>B = B - M</i>
<i>SUBD</i>	<i>IMM, DIR, EXT, IDX</i>	<i>N, Z, V, C</i>	<i>D = D - M</i>
<i>SBCA</i>	<i>IMM, DIR, EXT, IDX</i>	<i>N, Z, V, C</i>	<i>A = A - M - C</i>
<i>SBCB</i>	<i>IMM, DIR, EXT, IDX</i>	<i>N, Z, V, C</i>	<i>B = B - M - C</i>
<i>MUL</i>	<i>INH</i>	<i>Z, C</i>	<i>D = A × B (non signé)</i>
<i>DAA</i>	<i>INH</i>	<i>N, Z, V, C</i>	<i>Ajustement décimal de A</i>

9.3 Logique

<i>Instruction</i>	<i>Modes</i>	<i>Flags</i>	<i>Description</i>
<i>ANDA</i>	<i>IMM, DIR, EXT, IDX</i>	<i>N, Z, V=0</i>	<i>A = A AND M</i>
<i>ANDB</i>	<i>IMM, DIR, EXT, IDX</i>	<i>N, Z, V=0</i>	<i>B = B AND M</i>
<i>ANDCC</i>	<i>IMM</i>	<i>Tous</i>	<i>CC = CC AND M</i>
<i>ORA</i>	<i>IMM, DIR, EXT, IDX</i>	<i>N, Z, V=0</i>	<i>A = A OR M</i>
<i>ORB</i>	<i>IMM, DIR, EXT, IDX</i>	<i>N, Z, V=0</i>	<i>B = B OR M</i>
<i>ORCC</i>	<i>IMM</i>	<i>Tous</i>	<i>CC = CC OR M</i>
<i>EORA</i>	<i>IMM, DIR, EXT, IDX</i>	<i>N, Z, V=0</i>	<i>A = A XOR M</i>
<i>EORB</i>	<i>IMM, DIR, EXT, IDX</i>	<i>N, Z, V=0</i>	<i>B = B XOR M</i>
<i>BITA</i>	<i>IMM, DIR, EXT, IDX</i>	<i>N, Z, V=0</i>	<i>Test A AND M</i>
<i>BITB</i>	<i>IMM, DIR, EXT, IDX</i>	<i>N, Z, V=0</i>	<i>Test B AND M</i>
<i>COMA</i>	<i>INH</i>	<i>N, Z, V=0, C=1</i>	<i>A = NOT A</i>
<i>COMB</i>	<i>INH</i>	<i>N, Z, V=0, C=1</i>	<i>B = NOT B</i>
<i>COM</i>	<i>DIR, EXT, IDX</i>	<i>N, Z, V=0, C=1</i>	<i>M = NOT M</i>

9.4 Incrémentation/Décrémentation

<i>Instruction</i>	<i>Modes</i>	<i>Flags</i>	<i>Description</i>
<i>INCA</i>	<i>INH</i>	<i>N, Z, V</i>	<i>A = A + 1</i>
<i>INCB</i>	<i>INH</i>	<i>N, Z, V</i>	<i>B = B + 1</i>
<i>INC</i>	<i>DIR, EXT, IDX</i>	<i>N, Z, V</i>	<i>M = M + 1</i>
<i>DECA</i>	<i>INH</i>	<i>N, Z, V</i>	<i>A = A - 1</i>
<i>DECB</i>	<i>INH</i>	<i>N, Z, V</i>	<i>B = B - 1</i>
<i>DEC</i>	<i>DIR, EXT, IDX</i>	<i>N, Z, V</i>	<i>M = M - 1</i>
<i>CLRA</i>	<i>INH</i>	<i>N=0, Z=1, V=0, C=0</i>	<i>A = 0</i>
<i>CLRB</i>	<i>INH</i>	<i>N=0, Z=1, V=0, C=0</i>	<i>B = 0</i>
<i>CLR</i>	<i>DIR, EXT, IDX</i>	<i>N=0, Z=1, V=0, C=0</i>	<i>M = 0</i>
<i>NEGA</i>	<i>INH</i>	<i>N, Z, V, C</i>	<i>A = -A</i>
<i>NEGB</i>	<i>INH</i>	<i>N, Z, V, C</i>	<i>B = -B</i>
<i>NEG</i>	<i>DIR, EXT, IDX</i>	<i>N, Z, V, C</i>	<i>M = -M</i>

9.5 Décalages et Rotations

Cette section détaille les instructions de décalage et de rotation, essentielles pour la manipulation des bits dans les registres ou les adresses mémoire. Ces opérations sont fondamentales pour l'optimisation des calculs et la gestion des données au niveau binaire.

Instruction	Modes	Flags	Description
ASLA/LSLA	INH	N, Z, V, C	Décalage arithmétique/logique gauche A
ASLB/LSLB	INH	N, Z, V, C	Décalage arithmétique/logique gauche B
ASL/LSL	DIR, EXT, IDX	N, Z, V, C	Décalage arithmétique/logique gauche M
ASRA	INH	N, Z, C	Décalage arithmétique droit A (signe préservé)
ASRB	INH	N, Z, C	Décalage arithmétique droit B
ASR	DIR, EXT, IDX	N, Z, C	Décalage arithmétique droit M
LSRA	INH	N=0, Z, C	Décalage logique droit A
LSRB	INH	N=0, Z, C	Décalage logique droit B
LSR	DIR, EXT, IDX	N=0, Z, C	Décalage logique droit M
ROLA	INH	N, Z, V, C	Rotation gauche A à travers C
ROLB	INH	N, Z, V, C	Rotation gauche B à travers C
ROL	DIR, EXT, IDX	N, Z, V, C	Rotation gauche M à travers C
RORA	INH	N, Z, C	Rotation droite A à travers C
RORB	INH	N, Z, C	Rotation droite B à travers C
ROR	DIR, EXT, IDX	N, Z, C	Rotation droite M à travers C

9.6 Comparaisons

<i>Instruction</i>	<i>Modes</i>	<i>Flags</i>	<i>Description</i>
<i>CMPA</i>	<i>IMM, DIR, EXT, IDX</i>	<i>N, Z, V, C</i>	<i>Compare A avec M</i>
<i>CMPB</i>	<i>IMM, DIR, EXT, IDX</i>	<i>N, Z, V, C</i>	<i>Compare B avec M</i>
<i>CMPD</i>	<i>IMM, DIR, EXT, IDX</i>	<i>N, Z, V, C</i>	<i>Compare D avec M</i>
<i>CMPX</i>	<i>IMM, DIR, EXT, IDX</i>	<i>N, Z, V, C</i>	<i>Compare X avec M</i>
<i>CMPY</i>	<i>IMM, DIR, EXT, IDX</i>	<i>N, Z, V, C</i>	<i>Compare Y avec M</i>
<i>CMPU</i>	<i>IMM, DIR, EXT, IDX</i>	<i>N, Z, V, C</i>	<i>Compare U avec M</i>
<i>CMPS</i>	<i>IMM, DIR, EXT, IDX</i>	<i>N, Z, V, C</i>	<i>Compare S avec M</i>
<i>TSTA</i>	<i>INH</i>	<i>N, Z, V=0</i>	<i>Test A (compare avec 0)</i>
<i>TSTB</i>	<i>INH</i>	<i>N, Z, V=0</i>	<i>Test B</i>
<i>TST</i>	<i>DIR, EXT, IDX</i>	<i>N, Z, V=0</i>	<i>Test M</i>

9.7 Sauts et Sous-programmes

<i>Instruction</i>	<i>Modes</i>	<i>Description</i>
<i>JMP</i>	<i>DIR, EXT, IDX</i>	<i>Saut absolu</i>
<i>JSR</i>	<i>DIR, EXT, IDX</i>	<i>Appel de sous-programme</i>
<i>BSR</i>	<i>REL</i>	<i>Appel de sous-programme court</i>
<i>RTS</i>	<i>INH</i>	<i>Retour de sous-programme</i>
<i>RTI</i>	<i>INH</i>	<i>Retour d'interruption</i>

9.10 Pile

<i>Instruction</i>	<i>Description</i>
<i>PSHS</i>	<i>Empiler sur S (ordre: PC, U, Y, X, DP, B, A, CC)</i>
<i>PSHU</i>	<i>Empiler sur U</i>
<i>PULS</i>	<i>Dépiler de S (ordre inverse)</i>
<i>PULU</i>	<i>Dépiler de U</i>

Syntaxe

```
PSHS A,B,X      ; Empile A, B et X
PULS A,B,X      ; Dépile dans l'ordre inverse: X, B, A
```

9.11 Transfert/Échange

<i>Instruction</i>	<i>Syntaxe</i>	<i>Description</i>
<i>TFR</i>	<i>TFR R1,R2</i>	<i>R2 = R1</i>
<i>EXG</i>	<i>EXG R1,R2</i>	<i>Échange R1 ↔ R2</i>
<i>LEA</i>	<i>LEAX/Y/U/S n,R</i>	<i>Charge adresse effective</i>

Exemples

```
TFR A,B      ; B = A
EXG X,Y      ; Échange X et Y
LEAX 10,X     ; X = X + 10
LEAY -5,Y    ; Y = Y - 5
```

9.12 Instructions Spéciales

<i>Instruction</i>	<i>Description</i>
<i>ABX</i>	<i>$X = X + B$ (non signé, pas de flags)</i>
<i>SEX</i>	<i>Extension de signe de B vers A (D = B signé sur 16 bits)</i>
<i>NOP</i>	<i>Aucune opération (avance PC)</i>
<i>SWI</i>	<i>Interruption logicielle</i>
<i>CWAI</i>	<i>Clear and Wait for Interrupt</i>
<i>SYNC</i>	<i>Synchronisation avec événement externe</i>

11. Architecture Interne du CPU

11.1 Registres

Registre	Taille	Description
PC	16 bits	Program Counter - Adresse instruction suivante
A	8 bits	Accumulateur A - Opérations arithmétiques/logiques
B	8 bits	Accumulateur B - Extension de A
D	16 bits	Accumulateur double (A:B) - Opérations 16 bits
X	16 bits	Registre d'index - Adressage indexé
Y	16 bits	Registre d'index - Adressage indexé
U	16 bits	User Stack Pointer - Pile utilisateur
S	16 bits	System Stack Pointer - Pile système (sous-programmes)
DP	8 bits	Direct Page Register - Page mémoire directe
CC	8 bits	Condition Code Register - Flags d'état

11.2 Registre CC (Condition Codes)

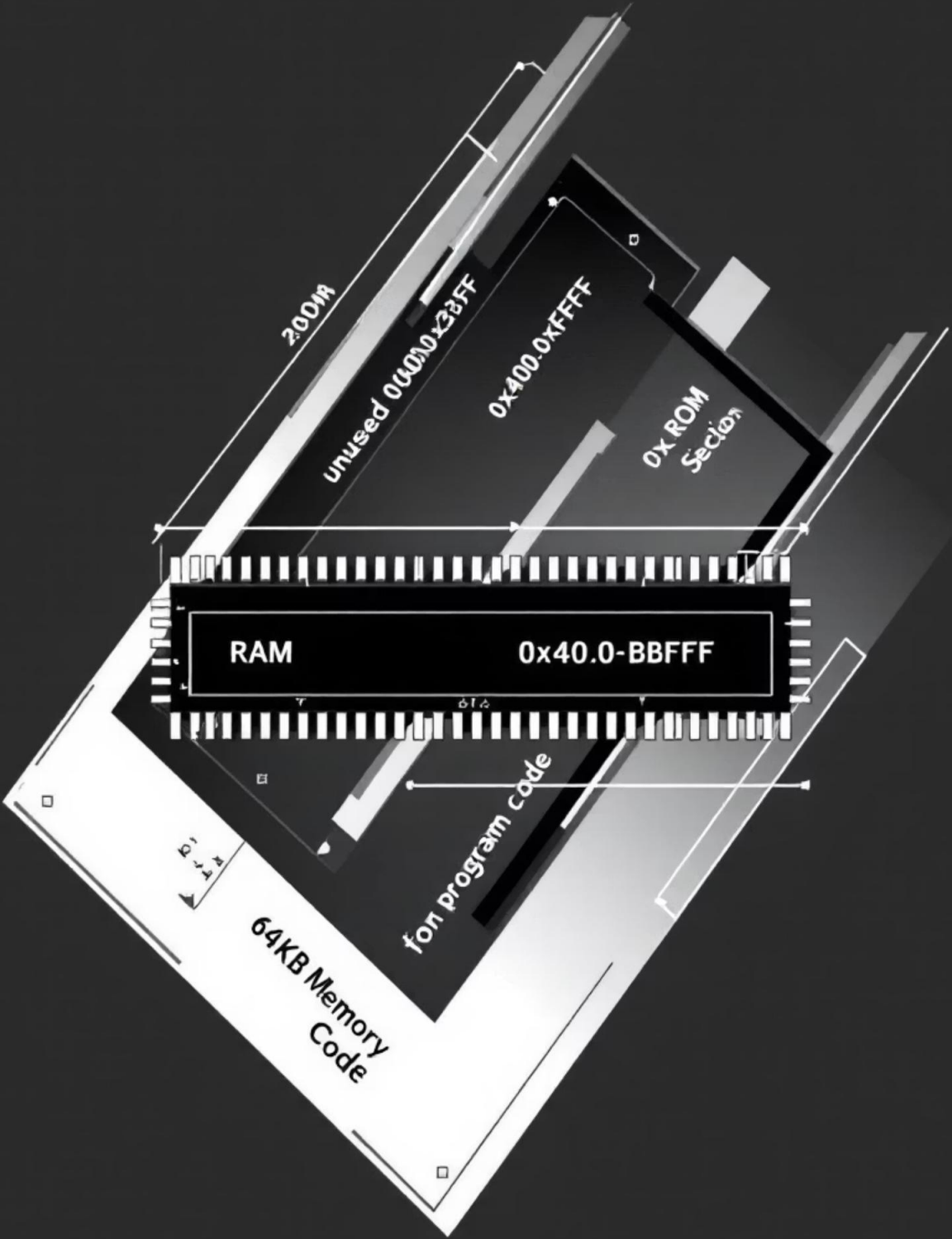
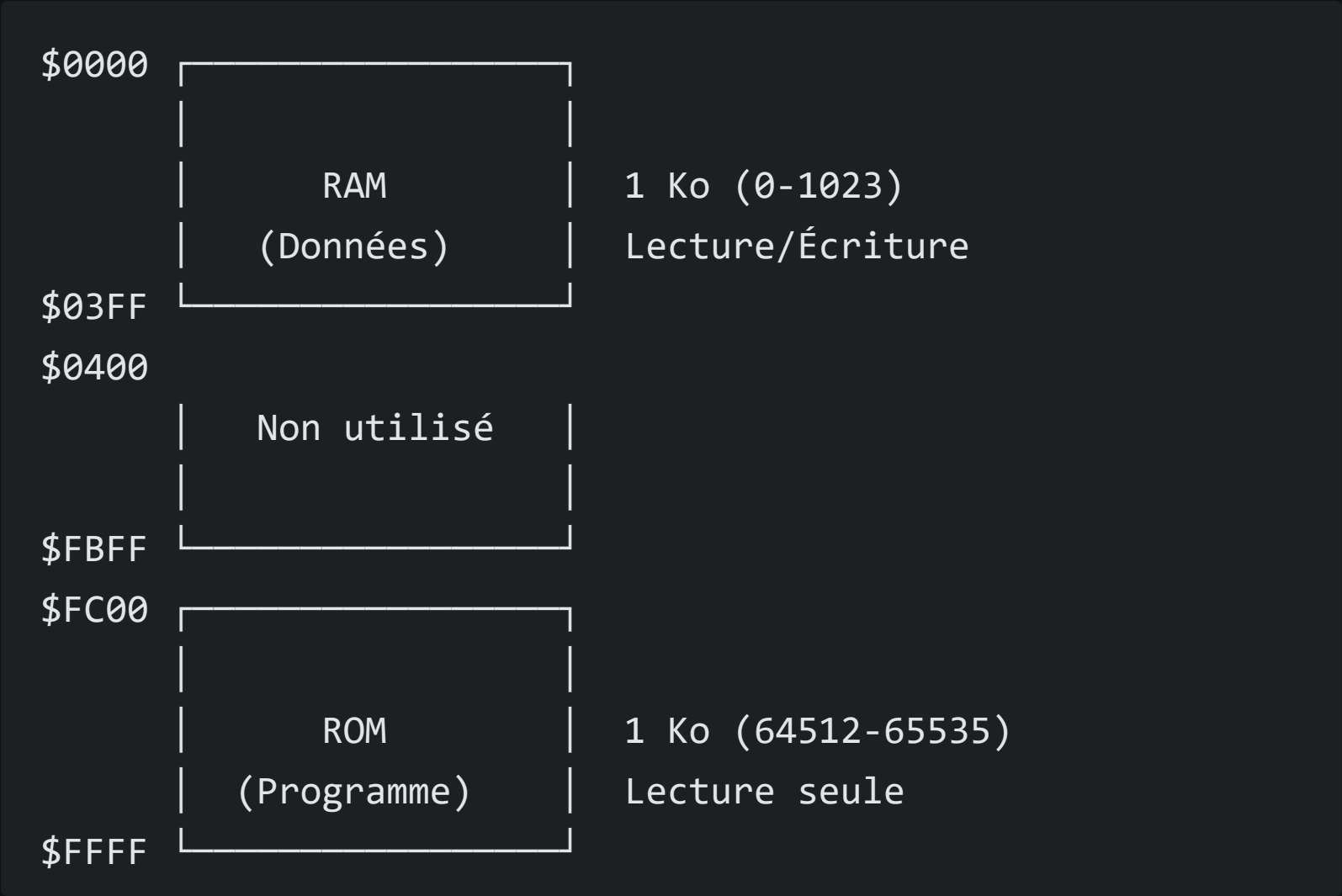
Bit: 7 6 5 4 3 2 1 0

E	F	H	I	N	Z	V	C

<i>Bit</i>	<i>Nom</i>	<i>Description</i>	<i>Mis à 1 quand</i>
<i>0</i>	<i>C</i>	<i>Carry</i>	<i>Retenue Résultat > 255 (8 bits) ou > 65535 (16 bits)</i>
<i>1</i>	<i>V</i>	<i>Overflow</i>	<i>Débordement Débordement signé (changement de signe inattendu)</i>
<i>2</i>	<i>Z</i>	<i>Zero</i>	<i>Zéro Résultat = 0</i>
<i>3</i>	<i>N</i>	<i>Negative</i>	<i>Négatif Bit 7 (8 bits) ou bit 15 (16 bits) = 1</i>
<i>4</i>	<i>I</i>	<i>IRQ Mask</i>	<i>Masque IRQ Interruptions IRQ désactivées</i>
<i>5</i>	<i>H</i>	<i>Half Carry</i>	<i>Demi-retenue Retenue du bit 3 vers bit 4 (BCD)</i>
<i>6</i>	<i>F</i>	<i>FIRQ Mask</i>	<i>Masque FIRQ Interruptions FIRQ désactivées</i>
<i>7</i>	<i>E</i>	<i>Entire</i>	<i>État complet État complet empilé lors d'interruption</i>

11.3 Organisation Mémoire

Le Motorola 6809 dispose d'un espace mémoire de 64 Ko (0 à 65535 octets). Dans ce simulateur, seules deux zones sont utilisées : la RAM pour les données et la ROM pour le programme.



RAM (Mémoire Vive)

- *Plage : \$0000 à \$03FF (1024 octets)*
- *Utilisation : Stockage des données, variables, pile*
- *Accès : Lecture et Écriture*
- *Volatilité : Contenu perdu à l'arrêt*

Zone Non Utilisée

- *Plage : \$0400 à \$FBFF*
- *Utilisation : Réservée, non accessible*
- *Raison : Simplification du simulateur*

ROM (Mémoire Morte)

- *Plage : \$FC00 à \$FFFF (1024 octets)*
- *Utilisation : Code programme (instructions)*
- *Accès : Lecture seule*
- *Persistance : Contenu permanent*

11.4 Cycle d'Instruction

- *Fetch : Lecture de l'opcode à l'adresse PC*
- *Decode : Analyse de l'instruction et du mode d'adressage*
- *Execute : Exécution de l'opération*
- *Write-back : Écriture du résultat (si nécessaire)*
- *Update PC : $PC = PC + \text{taille_instruction}$*

B. Messages d'Erreur Courants

<i>Erreur</i>	<i>Cause</i>	<i>Solution</i>
<i>Étiquette non définie</i>	<i>Branchement vers étiquette inexistante</i>	<i>Vérifier l'orthographe de l'étiquette</i>
<i>Format hexadécimal invalide</i>	<i>Valeur non hexadécimale</i>	<i>Utiliser 0-9, A-F</i>
<i>Adresse ROM hors limites</i>	<i>PC en dehors de \$FC00-\$FFFF</i>	<i>Vérifier ORG</i>
<i>Instruction non supportée</i>	<i>Mnémonique inconnue</i>	<i>Consulter le jeu d'instructions</i>
<i>Registre d'index invalide</i>	<i>Mauvais registre dans mode indexé</i>	<i>Utiliser X, Y, U ou S</i>

C. Conseils de Débogage



Utiliser Pas à Pas

Observer chaque instruction.



Surveiller les Flags

Vérifier Z, C, N après opérations.



Vérifier la Pile

S doit pointer sur adresse valide.



Contrôler PC

Ne doit jamais sortir de la ROM.



Initialiser les Registres

Toujours initialiser avant utilisation.