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

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*

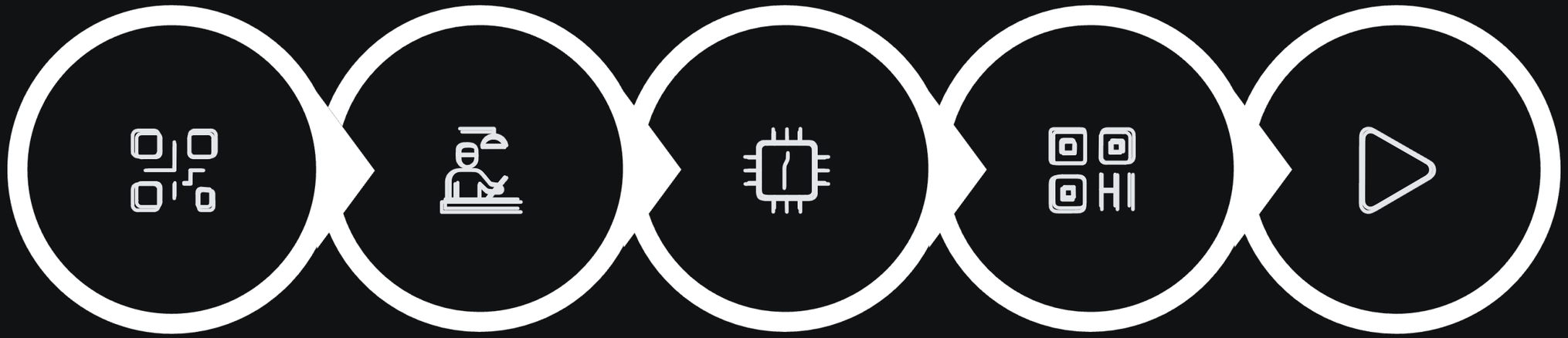
2. Architecture du Simulateur

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

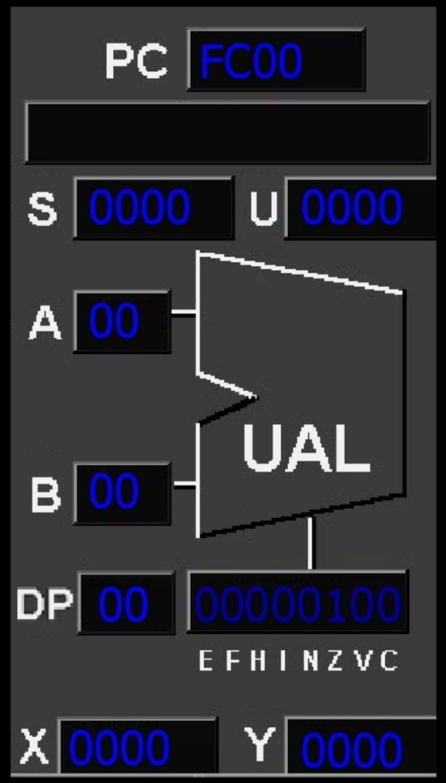
ZONE 4

Fichier Visualisation Aide

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*

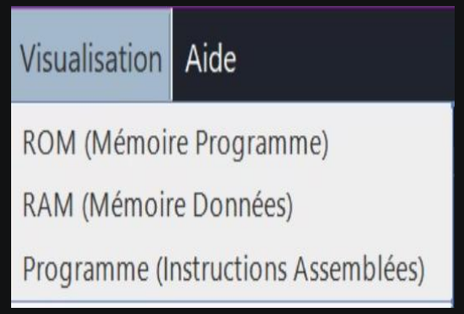
4. Barre de Menus

4.1 Menu Fichier

Option	*Raccourci*
Nouveau	*Ctrl+N*
Ouvrir...	*Ctrl+O*
Enregistrer	*Ctrl+S*
Imprimer...	*Ctrl+P*
Quitter	*Ctrl+Q*

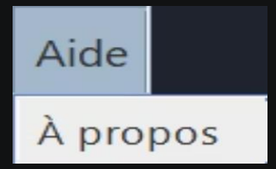
4.2 Menu Visualisation

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



4.3 Menu Aide

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



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

Instructions assemblées

7. Éditeur de Code

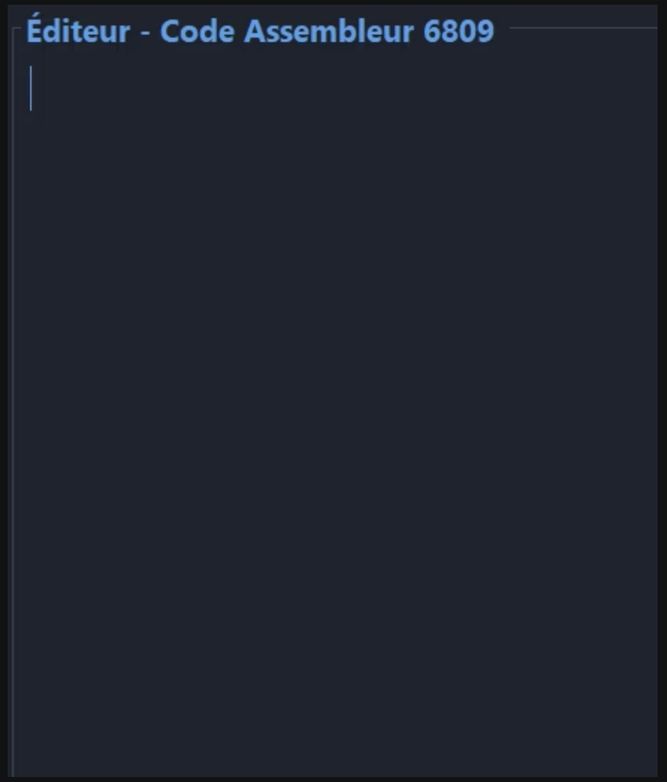
7.1 Fonctionnalités

- ***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*



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

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

9.3 Logique

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

9.4 Incrémentation/Décrémentation

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

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

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

9.7 Sauts et Sous-programmes

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

9.10 Pile

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

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

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

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

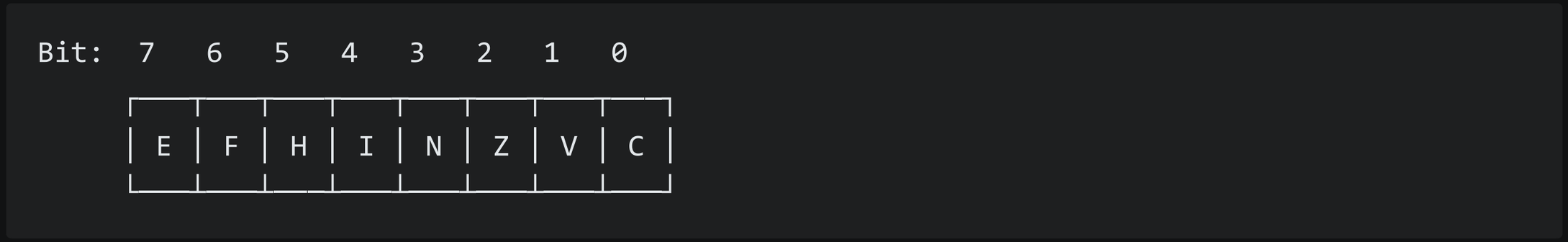
Instruction	*Description*
ABX	*X = X + B (non signé, pas de flags)*
SEX	*Extension de signe de B vers A (D = B signé sur 16 bits)*
NOP	*Aucune opération (avance PC)*
SWI	*Interruption logicielle*
CWAI	*Clear and Wait for Interrupt*
SYNC	*Synchronisation avec événement externe*

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*

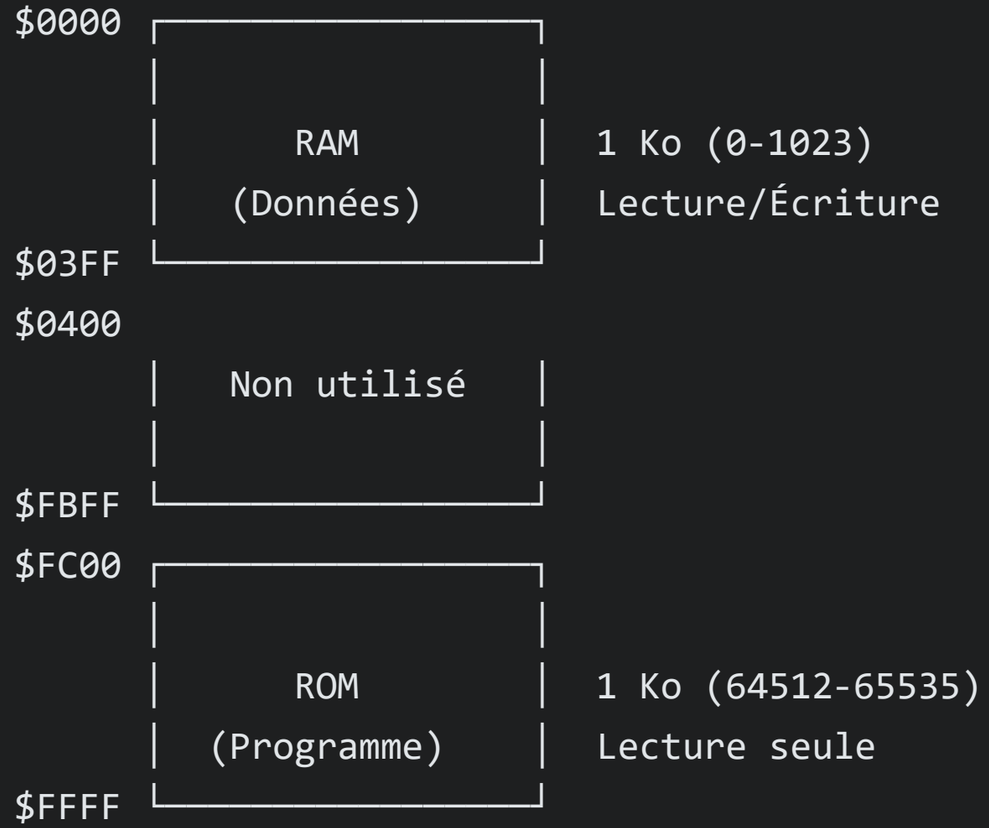
11.2 Registre CC (Condition Codes)

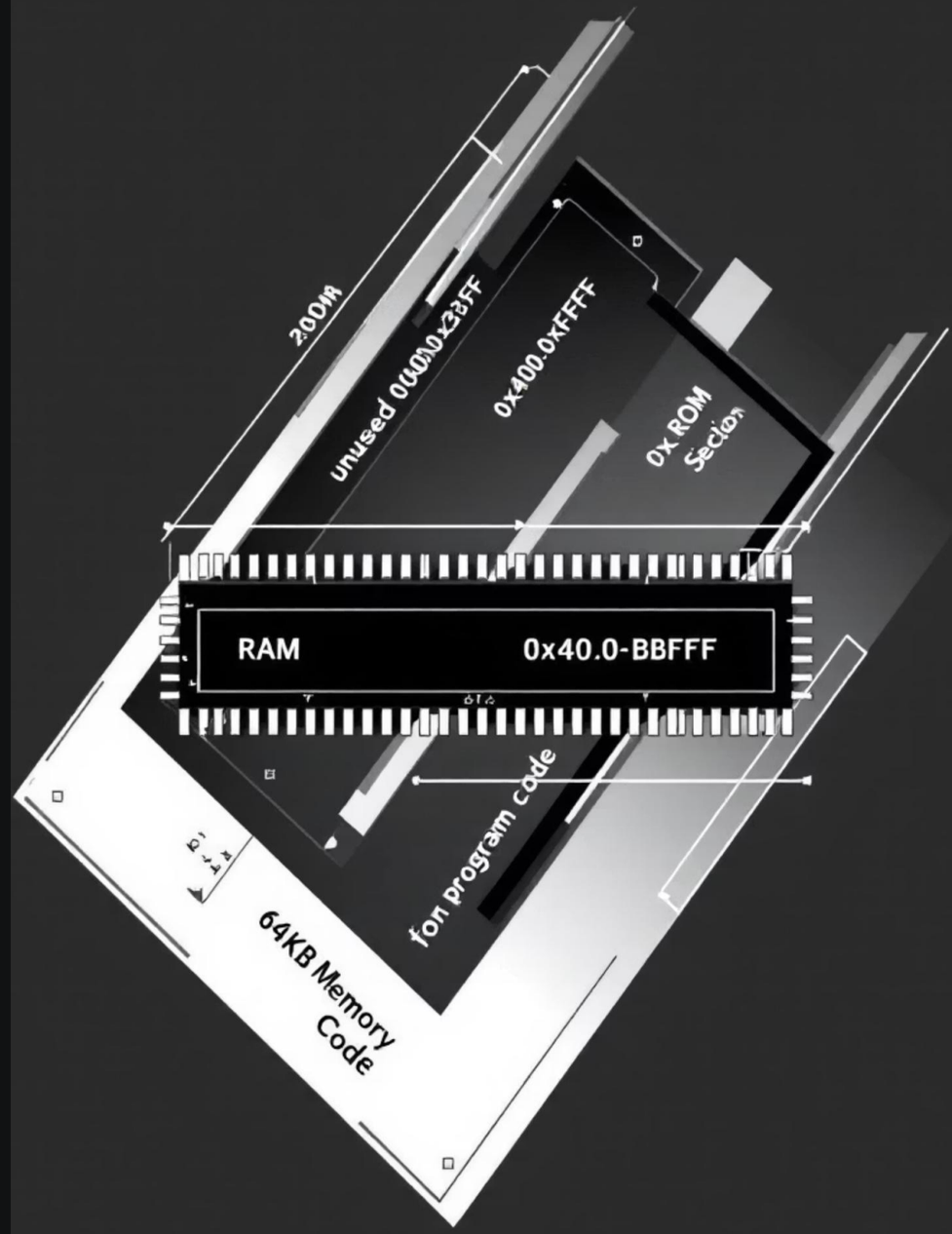


Bit	*Nom*	*Description*	*Mis à 1 quand*
0	*C*	*Carry*	*Retenue | Résultat > 255 (8 bits) ou > 65535 (16 bits)*
1	*V*	*Overflow*	*Débordement | Débordement signé (changement de signe inattendu)*
2	*Z*	*Zero*	*Zéro | Résultat = 0*
3	*N*	*Negative*	*Négatif | Bit 7 (8 bits) ou bit 15 (16 bits) = 1*
4	*I*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

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 + taille_instruction*

B. Messages d'Erreur Courants

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

C. Conseils de Débogage

Utiliser Pas à Pas

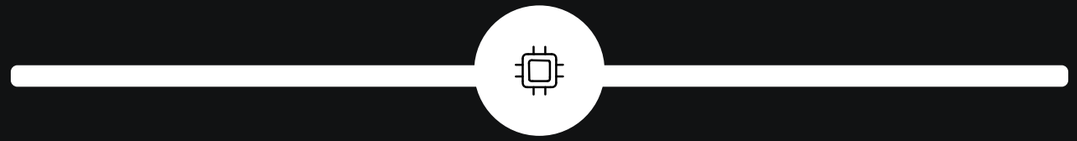
Observer chaque instruction.

Surveiller les Flags

Vérifier Z, C, N après opérations.

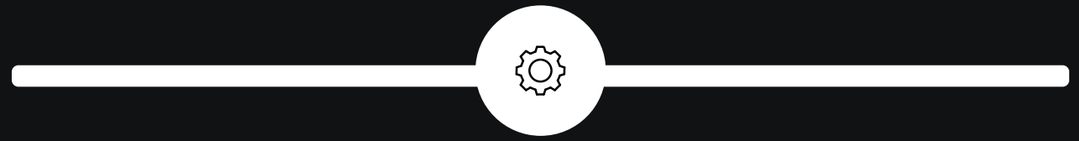
Vérifier la Pile

S doit pointer sur adresse valide.



Contrôler PC

Ne doit jamais sortir de la ROM.



Initialiser les Registres

Toujours initialiser avant utilisation.