

## ACTIVITE DE DECOUVERTE

### Exemple de programme écrit en langage assembleur

Le listing ci-dessous montre un programme écrit en langage Assembleur pour le microprocesseur Motorola 6809.

On distingue :

- Le programme écrit en langage assembleur à l'aide d'instructions ou mnémoniques
- Le programme écrit en langage « machine » codé en hexadécimal

0200 12	0010	ORG	\$0200
	0020	;	
	0030	PRODB EQU	\$00 ;PARTIE BASSE RESULTAT
	0040	PRODH EQU	\$01 ;PARTIE HAUTE RESULTAT
	0050	VAR EQU	\$02 ;INTERMEDIAIRE CALCUL
	0060	MDE EQU	\$05 ;MULTIPLICANDE
	0070	MTR FCB	\$12 ;MULTIPLICATEUR
	0080	;	
	0090	;	
0201 C6 08	0100	DEBUT LDB	#\$08 ;INITIALISE BOUCLE
0203 86 00	0110	LDA	#\$00 ;ANNULE RESULTAT
0205 1F 8B	0120	TFR	A,DP ;REGISTRE PAGE DIRECTE
0207 97 00	0130	STA	(PRODB ;PARTIE BASSE RESULTAT
0209 97 01	0140	STA	(PRODH ;PARTIE HAUTE RESULTAT
020B 97 02	0150	STA	(VAR ;
020D 74 00 02	0160	DECAL LSR	MTR ;BIT0 =0
0210 24 0C	0170	BCC	SUITE ;OUI, DECALE MULTIPLICANDE
0212 96 00	0180	LDA	(PRODB ;NON, ADD. MULTIPLICANDE
0214 9B 05	0190	ADDA	(MDE ;PARTIE BASSE
0216 97 00	0200	STA	(PRODB ;SAUVEGARDE PARTIE BASSE
0218 96 01	0210	LDA	(PRODH ;PARTIE HAUTE
021A 99 02	0220	ADCA	(VAR ;
021C 97 01	0230	STA	(PRODH ;SAUVEGARDE PARTIE HAUTE
021E 08 05	0240	SUITE ASL	(MDE ;DECALE MULTIPLICANDE
0220 09 02	0250	ROL	(VAR ;SAUVE BIT 7 DANS VAR
0222 5A	0260	DECB	;DERNIERE ITERATION?
0223 26 E8	0270	BNE	DECAL ;NON, RECOMMENCE
0225 3F	0280	SWI	;OUI, TERMINE
0226	0290	END	

On remarque que le programme écrit en langage machine codé en hexadécimal est peu compréhensible par le programmeur, mais c'est celui qui sera exécuté par le processeur.

Par contre, le programme écrit en langage assembleur est déjà plus « parlant ». On distingue les instructions et les opérations effectuées.

Les instructions sont tirées d'un jeu d'instructions fournies par le fabricant du microprocesseur.

✍ **D'après vous, quelle est l'adresse du début (origine) du programme ?**

En observant le code machine en hexadécimal, on observe que la première instruction est à l'adresse \$0200

✍ **Que signifie l'instruction ORG \$0200 ?**


L'instruction signifie que l'origine du programme se trouve en \$0200. C'est une instruction d'assemblage, c'est-à-dire que cette instruction ne donne pas de code machine mais indique

✍ **A quelle adresse finit le programme ?**

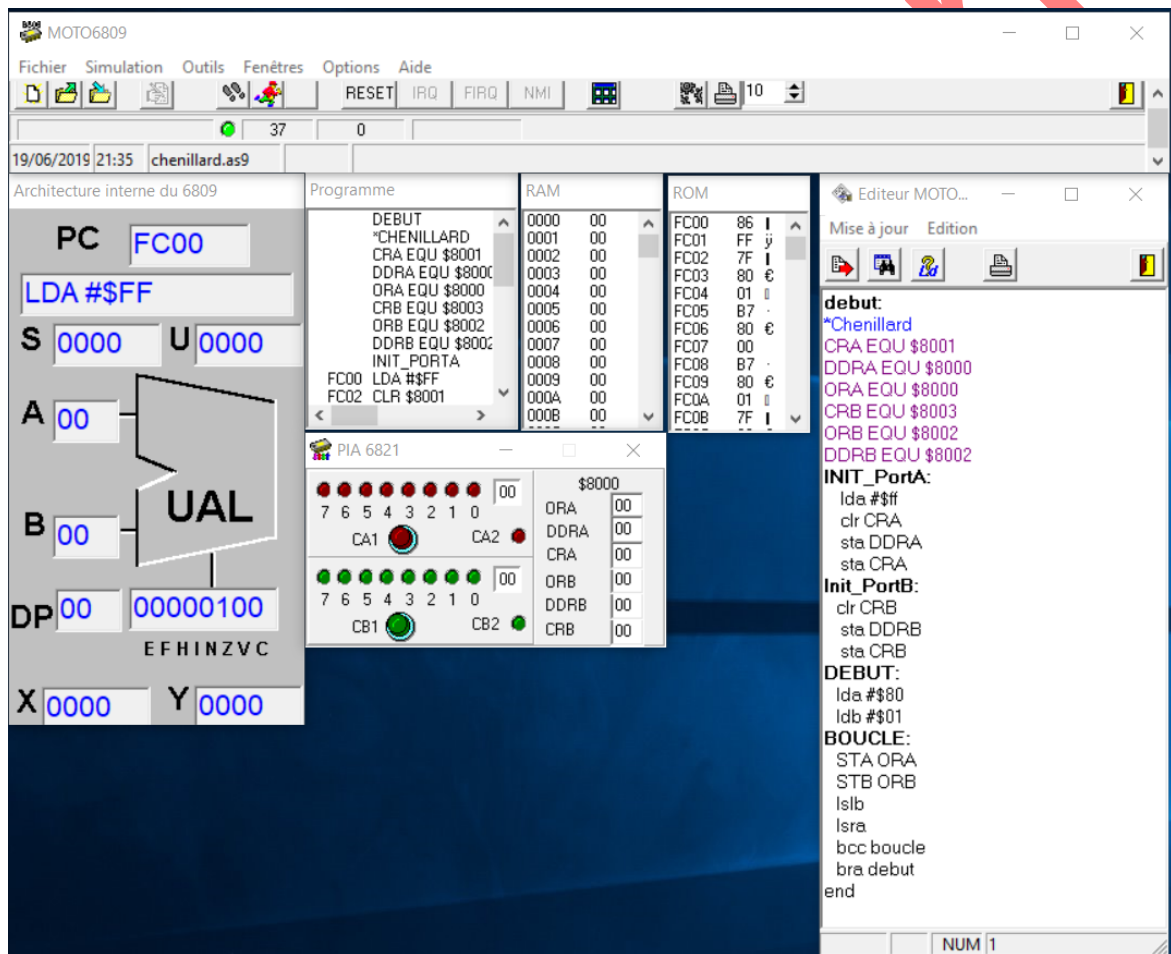
Le programme finit à l'adresse \$0226

Simulateur MOTO6809

On vous propose de découvrir la structure et le fonctionnement d'un microprocesseur avec un simulateur 6809.

- 🔗 Lancer le programme [chenillard.as9](http://chenillard.as9) écrit en assembleur qui vous est fourni dans ... \Ressources Elèves
- 🔗 Dans le menu Fenêtres, tout cocher : Programme, ROM, RAM, PIA
- 🔗 Cliquer sur l'icône Editeur  pour ouvrir l'éditeur de programme

Vous devez obtenir les fenêtres suivantes :



La fenêtre Editeur MOTO... permet de voir, saisir, modifier le programme écrit en assembleur.

-  Cliquez sur l'icône Exécuter  pour lancer l'exécution du programme Chenillard


 **Que fait le programme ?**

Le programme allume successivement des leds rouges sur le PIA1 (Interface de sorties parallèles) dans un sens, et des leds vertes sur le PIA2 dans l'autre sens.

Etant donné la vitesse d'exécution du programme trop rapide, il est difficile d'observer ce qu'il se passe.

- ✎ **Changer la vitesse d'exécution dans la case Vitesse de simulation, par exemple : 100 puis exécuter à nouveau le programme.**

Il existe un autre moyen d'exécuter le programme de façon beaucoup plus lente afin de le comprendre ce qu'il se passe à chaque ligne de code : le **mode pas à pas**  
Ce mode permet également de **debugger** un programme.

- ✎ **Faire un RESET pour réinitialiser le programme, puis l'exécuter à nouveau en mode pas à pas en cliquant sur l'icône** 

- 🔗 **Observer ce qu'il se passe au niveau des registres : A, B, PC et les indicateurs N, Z, V, C du registre d'états.**

- ✎ **Quel est le format (en bits et en octets) des registres A, B et PC ?**  
Les registres A et B sont au format 8 bits (1 octet) et le registre PC au format 16 bits (2 octets).

Note : le processeur 6809 est un **processeur 8 bits** car il effectue des opérations dans l'UAL (Unité Arithmétique et Logique) entre les registres A et B au format 8 bits.

- ✎ **Stopper le programme, puis faire un RESET**

- ✎ **Quel est alors le contenu de PC et l'instruction qui est prête à être exécuter.**  
PC contient l'adresse de début du programme \$FC00, et l'instruction LDA #\$FF est chargée.

- ✎ **Dans quelle mémoire se trouve le programme ?**  
Le programme se trouve en ROM.

- ✎ **Relever les adresses de début et de fin du programme, puis calculer la taille du programme en octets.**  
Taille du programme = @fin prog - @début prog = \$FC25 - \$FC00 = \$25 = (37)<sub>10</sub> octets


- ✎ **Dans le menu Outils, puis Information, dans onglet Programme, noter la taille du programme et la comparer à la valeur trouvée précédemment.**  
Le simulateur donne bien une taille de programme de 37 octets.

- ✎ **Calculez maintenant les capacités des mémoires RAM et ROM**  
Capacité de la ROM : \$FFFF - \$FC00 + 1 = \$400 => 1024 octets  
Capacité de la RAM : \$03FF - \$0000 + 1 = \$400 => 1024 octets

Vous allez maintenant changer une valeur dans le programme puis observer le changement par simulation.

- ✎ **Dans la fenêtre Editeur :**

- **à la ligne 19, changer la valeur \$80 par la valeur \$88**
- **à la ligne 20, changer la valeur \$01 par la valeur \$03**

- **enregistrer les modifications** 
- **lancer à nouveau l'exécution du programme**

- 🔗 **Quels sont les effets produits par les modifications précédentes ? Justifiez-les ?**

On allume deux leds rouges de la façon suivante : 10001000 et deux leds vertes de la façon suivante : 11000000.

10001000 correspond à \$88 et 00000011 à \$03