|  |  |  |
| --- | --- | --- |
| **Logo_Lycée** | **Langage machine et assembleur** | logo%20ac%20orl%E9ans%20toursDescription : Description : Description : pemDescription : Description : Description : pem |
| **TD**  **Découverte** | **Assembleur 6800** |

**Objectif** : identifier le code machine en mémoire après assemblage du code source.

**A. Présentation**

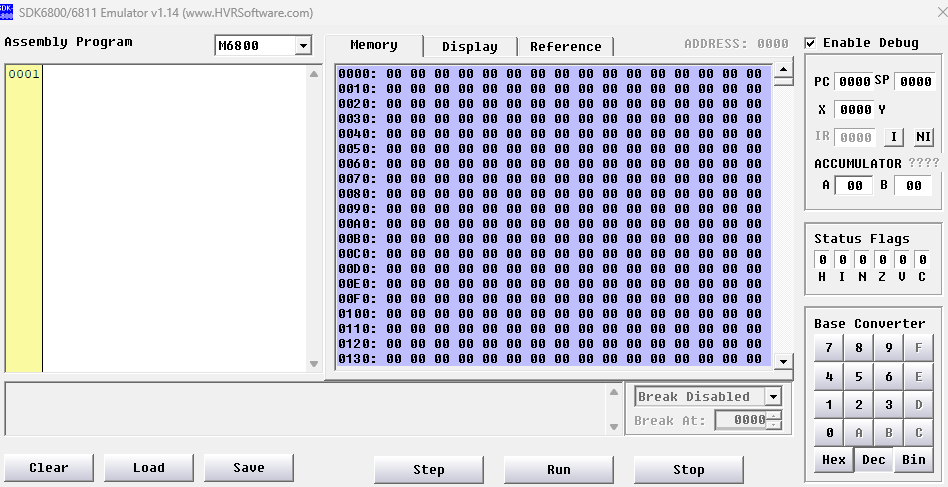
**A1. Le microcontrôleur MC6800**

|  |  |  |
| --- | --- | --- |
| Le [**MC6800**](https://fr.wikipedia.org/wiki/Motorola_6800) est un microprocesseur 8 bits produit par Motorola en 1975. Il avait [**72 instructions**](https://webge.fr/6800.html#ADD). Il se présentait habituellement sous la forme d'un boîtier [DIP](https://fr.wikipedia.org/wiki/Dual_Inline_Package) 40 broches. La version 'B' fonctionnait jusqu'à une fréquence de **2 MHz** et il contenait **7000** [transistors](https://fr.wikipedia.org/wiki/Transistor). | | Motorola MC6800 (1975) |
|  | Plusieurs **micro-ordinateurs** de première génération des années 1970, disponibles par correspondance sous forme de kits ou déjà assemblés, utilisaient les **MC6800** en tant que processeur principal. Par exemple la série MITS **ALTAIR 680**. Ce micro-ordinateur était utilisé en tant que **contrôleur de machines** ou de processus. | |

**A2. L’émulateur SDK6800/6811**

Le code du MC6800 est simulable dans un émulateur. Ce logiciel se compose principalement d’une zone recevant le programme source écrit en assembleur, d’une zone représentant le contenu de la mémoire et d’une zone affichant le contenu des registres du processeur. Les nombres affichés dans la zone mémoire sont codés en **hexadécimal**. Les nombres de la première colonne (suivie de :) correspondent aux adresses sur **16bits**, les autres sont les données (sur 8bits) correspondant à ces adresses.

Remarque : le dernier nombre de la première ligne est à l’adresse hexadécimal 0F.

****

**Caractéristiques du processeur MC6800**

- Fréquence : **2MHz**

- Largeur du bus des données : **8bits**

- Largeur du bus des adresses : **16bits**

- Instructions : **72** (composées de 1 à 3 octets) - Code Op d’une instruction : **1 octet**

**Q1.** Calculez le nombre N de positions dans la mémoire et l’adresse **@max** de la dernière position en mémoire (en décimal et en hexadécimal).

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Q2.** Donnez la valeur hexadécimale maximum que peuvent contenir la **mémoire** et les **registres A** et **B**.

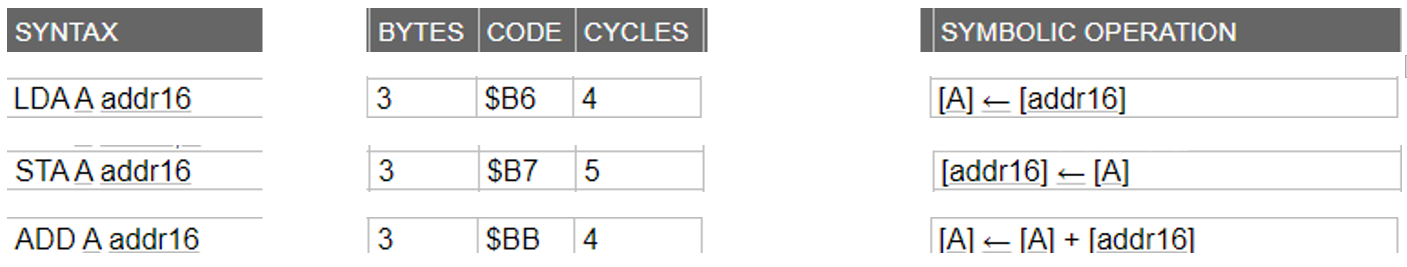
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**B. Programmation en assembleur 6800**

Opération à programmer en assembleur sur le 6800 : **v3 ← v1 + v2**

|  |  |
| --- | --- |
|  | **v3 ← v1 + v2 nécessite que le processeur exécute trois instructions-machine :**  ***Instruction 1***  **Charger** le registre accumulateur **A** avec la valeur contenue dans **v1.**  *Assembleur****: ldaa*** *v1 ;* ***l****oa****d******a****ccumulateur* ***A***  ***Instruction 2***  **Ajouter** le contenu de **A** et le contenu de **v2** dans **A**  *Assembleur :* **adda** v2**; add** Memory contents to the Accumulator **A**  ***Instruction 3***  **Mémoriser** le contenu de **A** dans v3  *Assembleur :* **staa** v3***; store******A****ccumulator* ***A*** *in Memory* |

Extrait de la documentation du MC6800



**$** signifie que la valeur est en hexadécimal

Code source du programme en assembleur et résultat de l’assemblage dans la mémoire (**.org** signifie origine, **.byte** signifie octet)

|  |  |
| --- | --- |
|  |  |

**Q3a.** Calculez la valeur hexadécimale de v1, v2, v3.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Q3b.** Déterminez les adresses de v1, v2 et v3.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Q4.** A quel code machine correspondent **ldaa** , **adda** et **staa** ? Identifiez les instructions 1, 2 et 3 dans le **plan mémoire** ci-dessus en les entourant.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**C. Synthèse (A compléter)**

;label opcode operand comment

.org $0000

0000 \_\_\_\_\_\_\_\_\_\_ ldaa v1 ; [A] <- [v1]

000\_ \_\_\_\_\_\_\_\_\_\_ adda v2 ; [A] <- [A] + [v2]

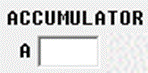
000\_ \_\_\_\_\_\_\_\_\_\_ staa v3 ; [v3] <- [A]

.org $00A0

00A0 \_\_\_\_\_\_ v1 .byte 10 ; v1 <- 10

00A1 \_\_\_\_\_\_ v2 .byte 34 ; v2 <- 34

00A2 \_\_\_\_\_\_ v3 .byte 0 ; v3 <- 0



**Q5.** Calculez le contenu de l’**accumulateur A** (en décimal et en hexadécimal) après l’instruction **adda v2**.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Q6.** Quelle est la valeur de la donnée à la position mémoire **00A216** après l’exécution de **staa v3** ?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_