|  |  |  |
| --- | --- | --- |
| **Logo_Lycée** | **Langage machine et assembleur** | logo%20ac%20orl%E9ans%20toursDescription : Description : Description : pemDescription : Description : Description : pem |
| **TP1**  **Découverte** | **Programmation en Assembleur 6800**  Opérations arithmétiques et transferts mémoire |

**🎯 Objectifs du TP**

**Date** : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **Classe** : \_\_\_\_\_\_\_

**Nom** : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Prénom** : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* Écrire et tester des programmes en assembleur 6800
* Manipuler les registres et la mémoire
* Comprendre l'exécution pas à pas d'un programme
* Analyser le contenu de la mémoire après exécution

**Prérequis :** Avoir complété le TD1 - Découverte Assembleur 6800

**📚 Rappels**

**Instructions de base du MC6800**

| **Instruction** | **Opcode (Extended)** | **Signification** | **Cycles** |
| --- | --- | --- | --- |
| ldaa adr | B6 | Charger A depuis la mémoire | 5 |
| ldab adr | F6 | Charger B depuis la mémoire | 5 |
| staa adr | B7 | Stocker A en mémoire | 6 |
| stab adr | F7 | Stocker B en mémoire | 6 |
| adda adr | BB | [A] ← [A] + [adr] | 5 |
| addb adr | FB | [B] ← [B] + [adr] | 5 |
| suba adr | B0 | [A] ← [A] - [adr] | 5 |
| subb adr | F0 | [B] ← [B] - [adr] | 5 |

**Notation :** [adr] signifie "contenu de l'adresse adr"

**Structure d'un programme assembleur**

**[Label] Operation [operand] [comment]**

.org $0000 ; Origine du programme en mémoire

; --- Instructions ---

ldaa v1 ; Commentaire explicatif

adda v2

staa v3

.org $00A0 ; Origine des données

; --- Données ---

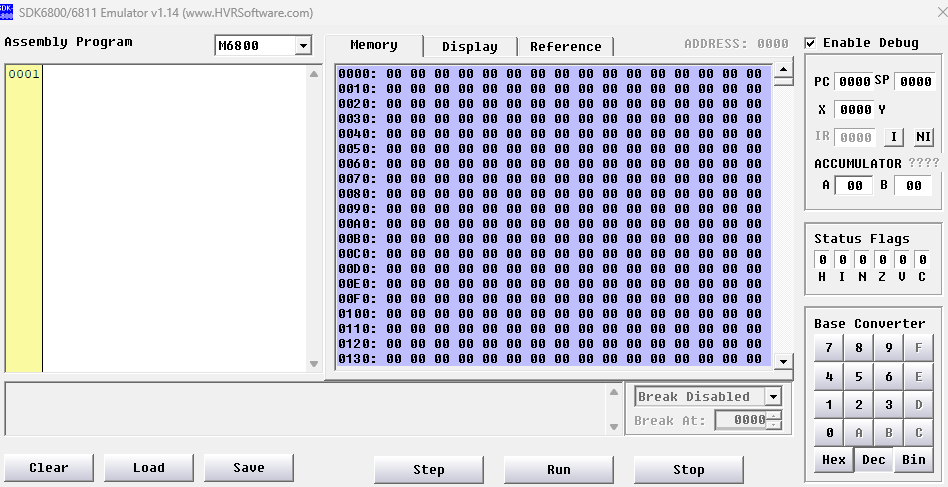
v1 .byte 10 ; Définition d'une variable

v2 .byte 20

v3 .byte 0

**Simulateur SDK6800/6811**

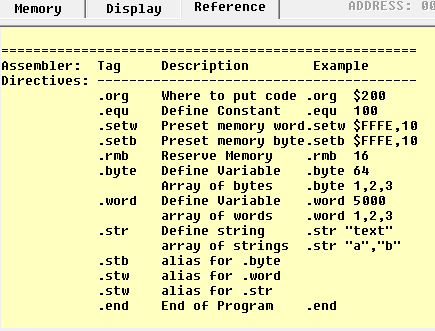
**Registres**

****

Contenu de la mémoire

Programme source en assembleur 6800

**🧪 EXERCICE 1 : Addition simple (Prise en main)**



**Énoncé**

Écrire un programme qui calcule : **resultat ← a + b**

Avec :

* a = 25₁₀
* b = 47₁₀
* resultat initialisé à 0

**Travail à réaliser**

**1.1 Complétez le code source suivant :**

.org $0000 **; EX1.asm**

; Instructions

ldaa \_\_\_\_\_\_\_\_\_\_ ; Charger a dans A

**Directives d’assemblage**

**.org** : la directive ORG indique l'adresse de départ du code assemblé.

**.byte** : la directive BYTE indique que la valeur qui suit est un octet.

\_\_\_\_\_\_\_\_ b ; Ajouter b à A

\_\_\_\_\_\_\_\_ resultat ; Stocker A dans resultat

.org $00B0 ; Données

a .byte 25

b .byte 47

resultat .byte 0

**1.2 Avant d'exécuter le programme dans l'émulateur :**

- Calculez manuellement le résultat attendu en **décimal** : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

- Convertissez ce résultat en **hexadécimal** : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**1.3 Dans l'émulateur SDK6800/6811 :**

- Préparez votre dossier ***home*** sur le serveur avec la fiche **« Organisation du dossier de travail »**



- Lancez le simulateur en cliquant sur

- Saisissez votre **code source**

**Organisation du programme source**

Les instructions d'assemblage contiennent les champs suivants :

**[Label] Operation [operand] [comment]**

Remarque : chaque champ doit être séparé par au moins un espace.

- Assemblez le code avec un clic sur  et sauvegardez le fichier sous **ex1.asm**

- Relevez le contenu de la mémoire (**code machine**) :

0000: \_\_\_\_\_ \_\_\_\_\_ \_\_\_\_\_ \_\_\_\_\_ \_\_\_\_\_ \_\_\_\_\_ \_\_\_\_\_ \_\_\_\_\_ \_\_\_\_\_

…

00B0: \_\_\_\_\_ \_\_\_\_\_ \_\_\_\_\_

**1.4 Identifiez dans la mémoire :**

* Les trois **opcodes** des instructions : \_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_
* Les adresses des variables (**opérandes**) : \_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_
* Combien d'octets occupe chaque instruction ? \_\_\_\_\_\_\_\_

**1.5 Exécutez le programme pas à pas :**

* Après ldaa a : A = \_\_\_\_\_\_\_\_ (en hexa)
* Après adda b : A = \_\_\_\_\_\_\_\_ (en hexa)
* Après staa resultat : Mémoire[00B2] = \_\_\_\_\_\_\_\_ (en hexa)

**1.6 Votre résultat correspond-il à la prédiction de la question 1.2 ? \_\_\_\_\_\_\_\_\_**

**🧪 EXERCICE 2 : Soustraction**

**Énoncé**

Écrire un programme qui calcule : **diff ← x - y**

Avec :

* x = 100₁₀
* y = 37₁₀
* diff initialisé à 0

**Travail à réaliser**

**2.1 Complétez le code source suivant :**

.org $0000 ; **EX2.asm**

; Instructions

ldaa \_\_\_\_\_\_

\_\_\_\_\_\_\_\_ y ; Instruction de soustraction

staa \_\_\_\_\_\_

.org $00C0 ; Données

x .byte 100

y .byte 37

diff .byte 0

**2.2 Prédiction :**

* Résultat attendu en **décimal** : \_\_\_\_\_\_\_
* En **hexadécimal** : \_\_\_\_\_\_\_

**2.3 Exécutez et vérifiez votre résultat. Enregistrez le fichier sous ex2.asm**

**2.4 ⚠️ Cas limite : Modifiez maintenant les valeurs :**

* x = 50
* y = 80

Réassemblez et exécutez. Quel résultat obtenez-vous en **hexadécimal** \_\_\_\_\_\_\_\_\_\_\_\_\_ et en **décimal** \_\_\_\_\_\_\_\_\_\_ ?

**2.5 Expliquez ce résultat surprenant (💡 Indice : pensez à la capacité d'un registre 8 bits et aux nombres signés/non** **signés).**

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

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

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

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

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

**🧪 EXERCICE 3 : Calcul à trois opérandes**

**Énoncé**

Écrire un programme qui calcule : **somme ← a + b + c**

Avec

* a = 15₁₀,
* b = 28₁₀,
* c = 42₁₀

**Travail à réaliser**

**3.1 Réfléchissez à l'algorithme :**

* Combien d'instructions ldaa, adda, staa sont nécessaires ? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
* Dans quel ordre devez-vous les écrire ?

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

**3.2 Écrivez le programme complet :**

.org $0000 **; EX3.asm**

; Placez votre code ici (4 instructions)

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

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

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

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

.org $00D0 ; Données

a .byte 15

b .byte 28

c .byte 42

somme .byte 0

**3.3 Tracez l'évolution du registre A :**

| **Après l'instruction** | **Valeur de A (hexa)** | **Valeur de A (décimal)** |
| --- | --- | --- |
| ldaa a |  |  |
| adda b |  |  |
| adda c |  |  |
| staa somme | (inchangé) |  |

**3.4 Vérifiez dans l'émulateur. Combien d'octets occupe votre programme en mémoire ?** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**🧪 EXERCICE 4 : Utilisation des deux registres A et B**

**Énoncé**

Calculez : **resultat = (a + b) - (c + d)**

Avec a = 50₁₀, b = 30₁₀, c = 20₁₀, d = 15₁₀

**Stratégie :**

* Utilisez le registre **A** pour calculer (a + b)
* Utilisez le registre **B** pour calculer (c + d)
* Puis faites la soustraction

**Travail à réaliser**

**4.1 Complétez le programme :**

.org $0000

; Calcul de (a + b) dans A

ldaa a

adda b

; Calcul de (c + d) dans B

\_\_\_\_\_\_\_ c ; Charger c dans B

\_\_\_\_\_\_\_ d ; Ajouter d à B

; Soustraction A - B

; ⚠️ Problème : il n'existe pas d'instruction "suba B" !

; Solution : on doit stocker B en mémoire temporaire

\_\_\_\_\_ temp ; Sauvegarder B

suba temp ; A ← A - [temp]

staa resultat

.org $00E0 ;Données

a .byte 50

b .byte 30

c .byte 20

d .byte 15

temp .byte 0 ; Variable temporaire

resultat .byte 0

**4.2 Prédiction manuelle :**

* (a + b) = \_\_50 + \_\_\_\_\_\_\_\_\_\_\_ en décimal \_\_\_\_\_\_\_\_\_\_\_\_\_\_ en hexadécimal \_\_\_\_\_\_\_\_\_\_\_
* (c + d) = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ en décimal \_\_\_\_\_\_\_\_\_\_\_\_\_\_ en hexadécimal \_\_\_\_\_\_\_\_\_\_\_
* Résultat final = \_\_\_\_\_\_\_\_ \_\_\_ en décimal \_\_\_\_\_\_\_\_\_\_\_\_\_\_ en hexadécimal \_\_\_\_\_\_\_\_\_\_\_

**4.3 Tracez l'évolution des registres :**

| **Instruction** | **A (hexa)** | **B (hexa)** |
| --- | --- | --- |
| Initial | 00 | 00 |
| ldaa a |  |  |
| adda b |  |  |
| ldab c |  |  |
| addb d |  |  |
| stab temp |  |  |
| suba temp |  |  |

**4.4 Exécutez et vérifiez.**