

## Quelques exercices simples pour bien démarrer les microprocesseurs

### Pour travailler dans la salle de TP

1- On souhaite calculer la somme de 45 et 85. Utiliser le 68000 pour calculer la somme.

**Corrigé :** Puisque les deux valeurs et leur somme est dans l'intervalle  $[-128,127]$ , on utilisera les octets (8 bits). On crée un fichier test.asm comme suit :

```
MOVE.B    #45,D0
ADD.B     #85,D0
TRAP      #0
```

Le résultat est dans le registre D0. Pour pouvoir le visualiser, on exécute le programme en mode débog :

```
>asm test.asm → le fichier test.bin se génère.
>db test.bin
```

Et puis, exécuter pas à pas avec la commande "s". Vérifier le contenu de registre D0.

**Exercice :** Utilisant la commande SUB, calculer  $45-12$ .

**Exercice :** Utilisant la commande MULS, calculer  $45*12$ .

**Exercice :** Utilisant la commande MULU, calculer  $45*12$ .

**Exercice :** Utilisant la commande MULS, calculer  $-45*12$ .

---

2- Utilisant 68000, calculer la somme de  $45+12-48+18$ .

**Corrigé :** On prend un intervalle plus large : 16 bits. Une façon de faire est comme le précédent :

```
MOVE.W    #45,D0
ADD.W     #12,D0
ADD.W     #-48,D0
ADD.W     #18,D0
TRAP      #0
```

Une autre façon de faire consiste à mettre les données dans un tableau et les additionner. Pour construire ce tableau, on utilise la directive DC.

```
LEA        TAB(PC),A0      A0 pointe à la première donnée
MOVE.W     (A0)+,D0         D0 contient 45
ADD.W      (A0)+,D0         D0 contient 57
ADD.W      (A0)+,D0         D0 contient 9
ADD.W      (A0)+,D0         D0 contient 27
TRAP       #0
TAB DC.W   45,12,-48,18
```

---

3- La même question que précédent sauf qu'il y a plus de données :  $45 + 12 - 48 + 18 - 5 + 81 + 12 + 35 - 18 + 89$

**Corrigé :** On peut faire comme avant mais cette fois si il faut répéter la ligne « ADD.W » 9 fois. On préfère utiliser une boucle pour faire répéter cette instruction.

```
        LEA        TAB(PC),A0        A0 pointe à la première donnée
        MOVE.W     (A0)+,D0          D0 contient 45
* Il faut répéter «ADD.W» 9 fois
        MOVE.W     #8,D1             compteur initialisé à 9-1
BCL     ADD.W       (A0)+,D0          Accumulation des données
        DBRA       D1,BCL
        TRAP       #0
TAB     DC.W        45,12,-48,18,-5,81,12,35,-18,89
```

**Exercice :** On souhaite calculer  $5 \times 10$  sans utilisation de la commande MUL. C'est-à-dire que l'on calcule  $10+10+10+10+10$ . Ecrire le programme en utilisant l'instruction DBRA.

---

4- Calculer la somme  $3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 + 11 + 12 + 13$ .

**Corrigé :**

```
        MOVE.W     #0,D0             Notre accumulateur
        MOVE.W     #3,D1             Première valeur
BCL     ADD.W       D1,D0             Accumulation
        ADD.W       #1,D1             Valeur suivante
        CMP.W       #14,D1           Si on arrive à 14, n'accumulons plus
        BNE        BCL
        TRAP       #0               Le résultat est dans D0
```

**Exercice :** Calculer la somme  $7 + 10 + 13 + \dots + 43$ .

---

5- Ecrire un sous programme qui reçoit deux valeurs signées (mots) dans D0 et D1 et qui retourne la max des deux dans D2.

**Corrigé :**

```
MAX     MOVE.W     D0,D2             On suppose que D0 est la max
        CMP.W      D1,D0
        BGE        OK               On saut à OK si D0 est la max
        MOVE       D1,D2             Si on est la, c'est D1 la max
OK      RTS
```

Tester le programme pour différentes valeurs de D0 et D1.

**Exercice :** Ecrire un sous programme ABS qui reçoit une valeur signée dans D0 et qui retourne sa valeur absolue dans D1. Testez votre programme pour des valeurs positive et négative de D0.

**Exercice :** Ecrire un sous programme MIN qui reçoit trois valeurs signées (mots) dans D0, D1 et D2 et qui retourne la min des trois dans D3.

**Exercice :** Ecrire un sous programme FONC qui reçoit une variable x dans D0.W et qui retourne dans D1.L la valeur :  $x^2 + x + 5$

---

6- Ecrire un sous programme qui reçoit une valeur dans D0 et qui retourne dans D1, 1 si D0 est impaire et 0 si D0 est paire.

Corrigé :

```
SP    MOVE.W    D0,D1
      AND.W     #1,D1
      RTS
```

**Exercice** : Ecrire un sous programme qui reçoit une valeur dans D0 et qui retourne dans D1, le reste de la division de D0 par 4.

**Exercice** : Ecrire un sous programme qui reçoit une valeur dans D0 et qui retourne dans D1, le reste de la division de D0 par 8.