

Exercice 01 : (Modes d'adressages)

1- Soit les codes suivants :

```
LDAA #$25  
END
```

```
LDAA #%00100101  
END
```

- Qu'est ce que vous remarquez ?
- De quel adressage s'agit t'il ?

2-

```
LDAA #$25  
STAA $0000  
END
```

```
LDAA #$15  
STAA $0042  
END
```

Qu'est ce que vous remarquez sur la fenêtre RAM ?

3-

```
LDAA #$25  
STAA $0000  
LDAA #$15  
STAA $0042  
LDAB,X  
END
```

- Que fait ce code ?
- Réécrire le code en remplaçant la 5^{me} instruction par :
LDAB \$04,X
- Que contient l'accumulateur B ?
- De quel mode d'adressage s'agit t'il ?
- Quelle est la différence entre les deux instructions ?
- Ajouter les instructions suivantes :
CLR \$0000
CLRA
- Quel est l'effet de ces instructions ?
- Quel est le mode d'adressage dans les deux cas ?

Exercice 02 : (Instructions de décalage)

1- Charger l'accumulateur A avec la valeur **#%11111111**

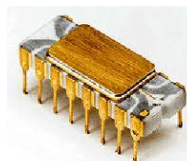
En utilisant l'instruction de décalage adéquate, comment peut-on obtenir les valeurs suivantes :

- **11111110** avec : flag C == 1
- **11111111** avec : flag C == 1

2- Charger maintenant l'accumulateur B avec la valeur **#%00001101**

En utilisant l'instruction de décalage adéquate, comment peut-on obtenir les valeurs suivantes :

- **00001110** avec : flag C == 1
- **a0000011** avec : flag C == 0 (avec a : valeur initiale de C)



Utiliser maintenant l'instruction **ASR** sur l'accumulateur **B**. Quel est le contenu du flag **C** ?

Remarque : Après chaque décalage : - ranger le résultat à l'adresse mémoire 0042.

- charger l'accumulateur considéré par le contenu de cette adresse

3- Faire un décalage de 01 bit à gauche de l'accumulateur A sans altérer le registre d'état.

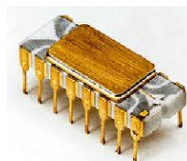
(Indication : utiliser les instructions TAP et TPA)

Exercice 03 : (Addition DCB)

- 1- Réaliser un programme qui :
 - Range les nombres **05** et **10** respectivement aux adresses **0001** et **0002**.
 - Additionne ces deux nombres.
 - Range le résultat à l'adresse mémoire **0003**.
- 2- Réaliser un programme qui :
 - Additionne les **02** nombres hexadécimaux **55** et **38**. Le résultat converti en décimal et rangé à l'adresse mémoire **00E1**.

(Indication : utiliser l'instruction d'ajustement décimal DAA)

- Modifier le programme afin de faire l'addition sur **16** bits des nombres hexadécimaux **1002** et **2030**.



Année : 2010/2011

(**1^{er}** nombre : partie haute à l'adresse **00F0** ; partie basse à l'adresse **00F1**

2nd nombre : partie haute à l'adresse **00F2** ; partie basse à l'adresse **00F3**

le résultant rangé à partir de **00F4**).

Exercice 04 : (Multiplication)

Réaliser un programme qui effectue :

- La multiplication des deux nombres 5 et 3, le résultat rangé dans la case mémoire **0010**.

Exercice 05 : (Recherche du Max)

- 1- Réaliser un programme qui :
 - Range les nombres décimaux suivants **25,15** et **05** respectivement aux adresses **00F1**, **00F4** et **00F6**.
 - Cherche la valeur maximale de ces nombres, le MAX sera rangé à l'adresse **00FF**.
- 2- Réaliser un programme qui :
 - Trie par ordre croissant les nombres précédents dans les cases mémoires **0000**, **0001** et **0002**.

Exercice 06 : (Opérations sur le registre d'état)

Réaliser un programme qui :

- Met les flags du registre d'état à **0**.
- Met les flags du registre d'état à **1**.

Exercice 07 :

01) On appelle « **masquage** » l'annulation d'un ou plusieurs bits dans un mot mémoire.

Réaliser un code qui permet de masquer la partie basse (les **04** bits de poids faible) du mot binaire suivant : « **10101010** ».

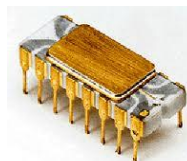
02) Réaliser un code qui permet de mettre à **1** la partie basse du mot binaire précédent.

Exercice 08 :

Réaliser un code qui permet de calculer les **20** premiers termes de la série de FIBONACCI rangés à partir de l'adresse **0040**.

Remarque : dans la série de FIBONACCI, on définit :

$$U_{n+2} = U_{n+1} + U_n \quad \text{avec : } U_1=1 \text{ et } U_2=2 .$$



Année : 2010/2011

Exercice 09 :

Réaliser un code qui compare les nombres X , y et range la valeur **A** dans (**0001** si $X > y$), **B** dans (**0002** si $X = y$) et **C** dans (**0003** si $X < y$).

- $X = 0F$ $y = 1F$
- $X = 0F$ $y = 0F$
- $X = DF$ $y = FF$

Exercice 10 : (Soustraction DCB)

Réaliser le code qui permet de faire la soustraction de deux nombres X , $y \leq 99$ écrits en DCB. Le résultat étant en DCB. (X , y sont positifs et $X > y$).

Rappel : pour effectuer $(X - y)$, on additionne X avec le **complément à 10** de y . (Car pour avoir le résultat en DCB, on doit utiliser l'instruction **DAA**, qui ne fonctionne qu'après une addition).

Le complément à 10 de y : $C10 = (99)_{\text{hex}} - y + 1$

Exercices supplémentaires

Exercice 01 : (Modes d'adressages)

La valeur hexadécimale **23** étant rangée dans la case mémoire **[0012]**.

- 1- Chargez cette valeur dans l'accumulateur A en utilisant le mode d'adressage **Immédiat** puis **Direct**.
- 2- Additionnez avec la valeur immédiate **35** et rangez le résultat dans l'accumulateur **A**.
- 3- Interchangez le contenu de l'accumulateur **A** et **B**.

Exercice 02 : (Opérations sur le registre d'état)

Réalisez un code qui :

- 1- Met les flags du registre d'état à **1**.
- 2- Masque la partie basse (les **04** bits de poids faible) du registre d'état.

Exercice 03 : (Recherche du Max)

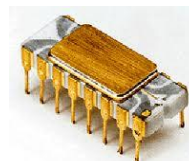
Réalisez un code qui :

- Cherche la valeur maximale des nombres **25,15** et **05**, rangés respectivement aux adresses **[00F1]**, **[00F2]** et **[00F3]**.
Le MAX sera rangé à l'adresse **[00F5]**.

Exercice 04 :

Réalisez un code qui :

- Détermine combien de **1** existe dans le mot binaire « **11110001** » se trouvant dans la case mémoire **[0000]**.



Année : 2010/2011