UNIVERSITE BADJI-MOKHTAR ANNABA **27/11/2023**

Département d'Electronique

Licence 3: AUTOMATIQUE 2023/2024

Microprocesseurs et Microcontrôleurs

SERIE D'EXERCICES AVEC SOLUTIONS (ASSEMBLEUR 6809)

Les exercices de cette série sont extraits du livre : « ASSEMBLEUR DU 6809 ET SES PERIFERIQUES R.Sorek v4.13 »

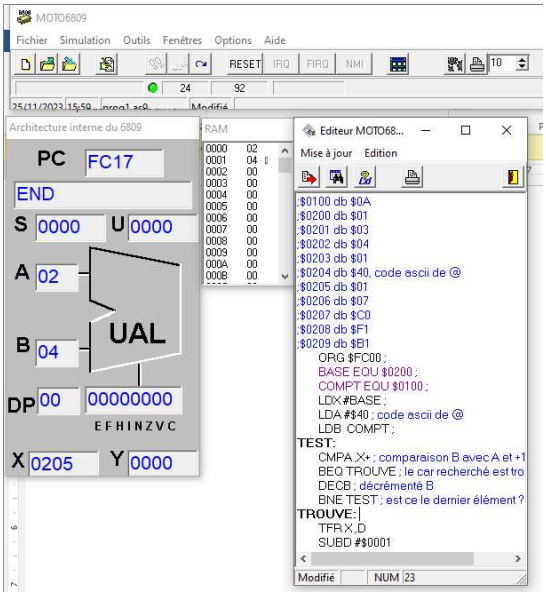
Les exercices proposés sont adaptés et corrigés pour la simulation avec MOTO6809.

EXERCICE 1

La rechercher de la valeur $40 (code ASCII de @) dans un tableau de 100 éléments L'adresse de départ de ce tableau BASE $0200.la taille du tableau dans COMPT $0100

Il écrit l'adresse de la case mémoire contenant le symbole @ dans ($0000 et $0001)

```
;$0100 db $0A
;$0200 db $01
;$0201 db $03
;$0202 db $04
;$0203 db $01
;$0204 db $40, code ascii de @
;$0205 db $01
;$0206 db $07
;$0207 db $C0
;$0208 db $F1
;$0209 db $B1

        ORG $FC00 ;
        BASE EQU $0200 ;
        COMPT EQU $0100 ;
        LDX #BASE ;
        LDA #$40 ; code ascii de @
        LDB  COMPT ;
TEST:

       CMPA ,X+ ; comparaison B avec A et +1 sur X
        BEQ TROUVE ; le car recherché est trouvé
        DECB ; décrémenté B
        BNE TEST ; est ce le dernier élément ?
TROUVE:
        TFR X ,D
        SUBD #$0001
        STD $0000
        ;sauvegarder l'dresse de l élément trouvé dans $0000 et $0001
        END
```



EXERCICE 2

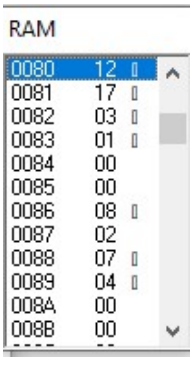
Transfert d'une table de données d'une zone mémoire vers une autre

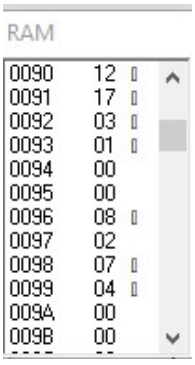
On dispose d'une table de 10 données de 8 bits, choisies arbitrairement, dont l'adresse de base est ADR1.

Proposer un programme permettant de transférer cette table à l'adresse de base ADR2.

La méthode utilisée ici consiste à charger une valeur dans le registre A en se servant du pointeur X (identifiant de la table source), et de stoker cette valeur à l'adresse désignée par le pointeur Y (identifiant la table de destination).
Transfert d'une table de 10 données de ADR1 $0080−> ADR2 $0090

```
;$0080 db $12
;$0081 db $17
;$0082 db $03
;$0083 db $01
;$0084 db $00
;$0085 db $00
;$0086 db $08
;$0087 db $02
;$0088 db $07
;$0089 db $04

        ADR1     EQU $0080 ; Déclaration de l'adresse ADR1
        ADR1_FIN     EQU $008A ; Déclaration de l'adresse ADR1+10
        ADR2     EQU $0090 ; Déclaration de l'adresse ADR2
        ORG $FC00 ;
        LDX #ADR1 ; Chargement du pointeur X
        LDY #ADR2 ; Chargement du pointeur Y
Boucle:
        LDA ,X+ ; Chargement et incrémentation du pointeur X
        STA ,Y+ ; Chargement et incrémentation du pointeur Y
        CMPX #ADR1_FIN ; Si le pointeur dépasse la fin de la table
        BNE Boucle
        END
```

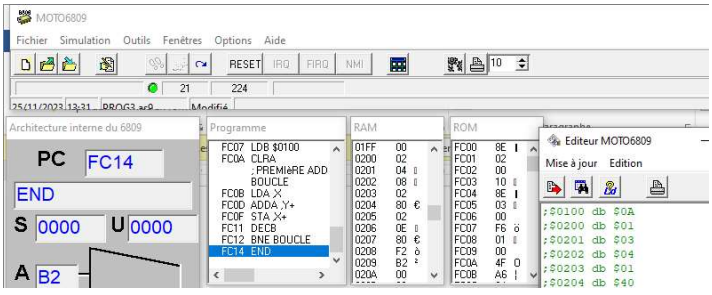




EXERCICE 3

Addition élément par élément, 2 blocs qui débutent respectivement aux adresses BLK1 et BLK2.
Ces 2 blocs ont le même nombre d'élément ET stocker le résultat dans BLK1

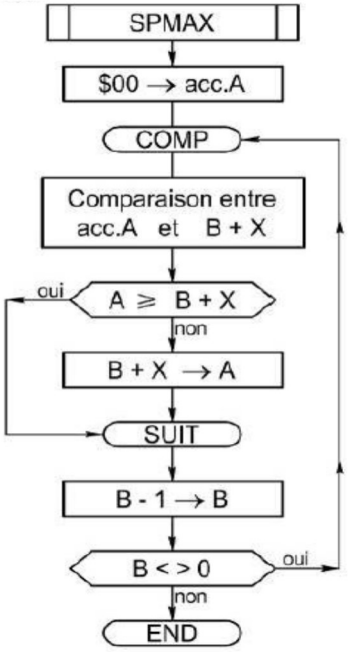
```
;$0100 db $0A

;$0200 db $01
;$0201 db $03
;$0202 db $04
;$0203 db $01
;$0204 db $40
;$0205 db $01
;$0206 db $07
;$0207 db $C0
;$0208 db $F1
;$0209 db $B1

;$0300 db $01
;$0301 db $01
;$0302 db $04
;$0303 db $01
;$0304 db $40
;$0305 db $01
;$0306 db $07
;$0307 db $C0
;$0308 db $01
;$0309 db $01
        ORG $FC00 ;
 BLK1   EQU $0200 ;
 BLK2   EQU $0300 ;
 COMPT  EQU $0100 ;
        LDX #BLK1 ;
        LDY #BLK2 ;
        LDB COMPT ; Nb d'élément à additionner mis dans B
        CLRA ; RAZ du bit de retenue en prévision de la
             ; première addition
BOUCLE:
        LDA ,X ; premier élément mis dans A
        ADDA ,Y+ ; ajout du 2ième élément, puis +1 sur Y
        STA ,X+ ; résult sauv dans case mém BLK1, +1 sur X
        DECB ; B décrémenté
        BNE BOUCLE ; aussi longtemps que B n'est pas = à 0
        END
```



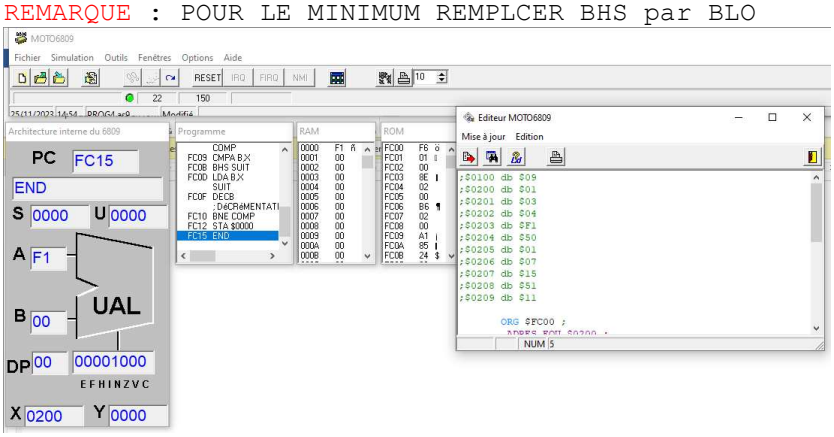
EXERCICE 4

Recherche du maximum d'un tableau de valeurs (non signés) et stoker le maximum dans l'dresse $0000

```
;$0100 db $09
;$0200 db $01
;$0201 db $03
;$0202 db $04
;$0203 db $F1
;$0204 db $50
;$0205 db $01
;$0206 db $07
;$0207 db $15
;$0208 db $51
;$0209 db $11

        ORG $FC00 ;
         ADRES EQU $0200 ;
         LONG  EQU  $0100

        LDB LONG ; charge la longueur de la table
        LDX #ADRES ; charge l'adresse de début
        LDA ADRES ; init de la valeur maxi 1 VAL DU TAB
  COMP:
         CMPA B,X ; nouvelle valeur maxi ???
         BHS SUIT ; non, on recommence
         LDA B,X ; oui, charge ce nouveau maxi
  SUIT:
         DECB ; bloc terminé ?
              ; Décrémentation de B
         BNE COMP ; non, on continue
         STA $0000 ; sauve le résultat
         END
```



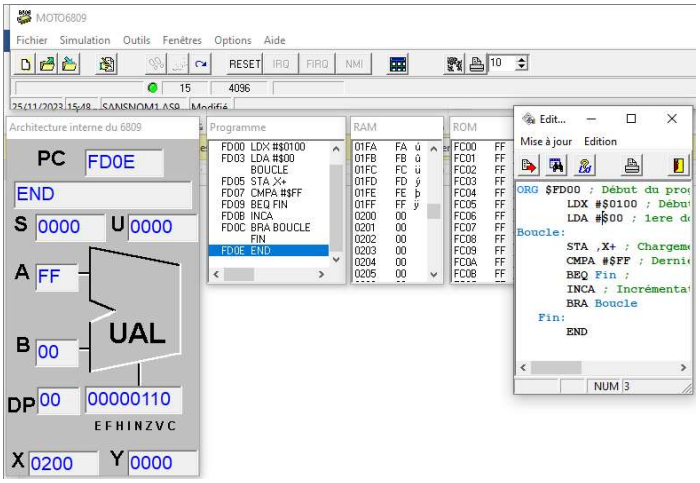
REMARQUE : POUR LE MINIMUM REMPLCER BHS par BLO



EXERCICE 5

Création d'une table de données en bits non signés de 00 à FF

```
        ORG $FD00 ; Début du programme

        LDX #$0100 ; Début de table
        LDA #$00 ; 1ere données $00
Boucle:
        STA ,X+ ; Chargement et incrémentation du pointeur
        CMPA #$FF ; Dernière donnée = $FF alors fin de programme
        BEQ Fin ;
        INCA ; Incrémentation de la donnée
        BRA Boucle
   Fin:
        END
```



Modifier le programme pour des données de FF à 00

```
Etat de la mémoire après exécution du programme
0100 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
0110 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
0120 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F
0130 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
0140 40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F
0150 50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F
0160 60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F
0170 70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F
0180 80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F
0190 90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F
01A0 A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF
```

EXERCICE 6

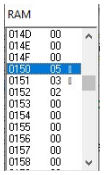
Proposer un programme permettant d'effectuer le comptage des données positives, négatives et nulles d'une table de nombres signés de 8 bits. Le programme devra permettre de stocker ces résultats aux adresses $0150, $0151,$0152

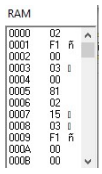
INDICATIONS

Après avoir chargé la valeur dans le registre A, qui automatiquement positionne les bits N et Z, on peut utiliser les instructions de branchements qui en découlent.

```
;$0000 db $02
;$0001 db $f1
;$0002 db $00
;$0003 db $03
;$0004 db $00
;$0005 db $81
;$0006 db $02
;$0007 db $15
;$0008 db $03
;$0009 db $f1
TABLE   EQU $0000
FIN_TAB EQU $000A
ORG $FC00
        LDX #TABLE ; Chargement du pointeur
 Boucle:
        CMPX #FIN_TAB ; Si le pointeur dépasse la fin de la table
        BEQ FIN ; alors FIN
        LDA ,X+ ; Chargement et incrémentation du pointeur
        BMI Negatif ; Si l'opération est négative -> Négatif
        BEQ Nul ; Si A = 0 -> Nul
        LDB $0150 ; Sinon la données est positive
        INCB ; Incrémente le compteur situé en $0050
        STB $0150 ; On mémorise la valeur
        BRA Boucle ;
 Negatif:
        LDB $0151 ; La données est négative
        INCB ; Incrémente le compteur situé en $0051
        STB $0151 ; On mémorise la valeur
        BRA Boucle ;
   Nul:
        LDB $0152 ; La données est nul
        INCB ; Incrémente le compteur situé en $0052
        STB $0152 ; On mémorise la valeur
        BRA Boucle
   FIN:
        END
```

Etat de la RAM après exécution du programme





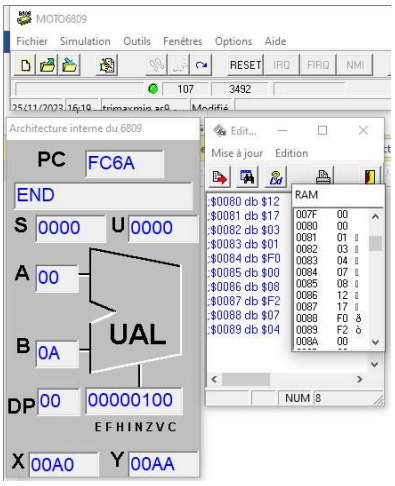
EXERCICE 7

TRI D'UN TABLEAU du min au max (Nombres non signés)

```
;$0080 db $12
;$0081 db $17
;$0082 db $03
;$0083 db $01
;$0084 db $F0
;$0085 db $00
;$0086 db $08
;$0087 db $F2
;$0088 db $07
;$0089 db $04
```

```
TABLE EQU $0080 ; Déclaration de la table TABLE
TABLE_FIN EQU $008A ; Déclaration de la table TABLE FIN
TABLEC EQU $00A0 ; Déclaration de la table TABLE
TABLEC_FIN EQU $00AA ; Déclaration de la table TABLE FIN
TABLED EQU $00C0 ; Déclaration de la table TABLED
CPTEUR EQU $00D0 ; Déclaration de la variable COMPTEUR
VALEUR EQU $00E0 ; Déclaration de la variable VALEUR

        ORG $FC00 ; Début du programme
        LDX #TABLE ; Chargement du pointeur X
        LDY #TABLEC ; Chargement du pointeur Y
Copy:

        LDA ,X+ ; Chargement et incrémentation du pointeur X
        STA ,Y+ ; Chargement et incrémentation du pointeur Y
        CMPX  #TABLE_FIN ; Si le pointeur dépasse la fin de la table
        BNE Copy ; alors Copy
        CLRB ;
Boucle:


        LDB CPTEUR ; Chargement dans B du contenue de COMPTEUR
        LDX #TABLEC ; Chargement du pointeur X
        LDA B,X ; Chargement de A avec le contenue de X+B
        STA VALEUR ; Mémorisation de A à l'adresse VALEUR
        CMPB #$0A ; Si B = nombre de donnée de la table
        BEQ FIN ; alors FIN
        CLRB ; RAZ de l'accumulateur B
Data:

        CMPX #TABLEC_FIN ; Si pointeur X est égal à fin de table
        BEQ Mem ; alors -> Mem, mémorisation de la donnée
        LDA ,X+ ; Chargement et incrémentation du pointeur X
        CMPA VALEUR ; Si A est strictement plus petit que VALEUR
        BLO Compt ; alors -> Compt
        BRA Data ; Sinon -> Data
Compt:

        INCB ; Incrémentation de B
        BRA Data ; Retour à Data
Mem:
        LDX #TABLED ; Chargement du pointeur X
```

```
        LDA B,X ; Chargement de A avec le contenue de X+B
        CMPA #$01 ; Si A = $01
        BEQ Egal ; alors pointeur déjà utilisé -> Egal
        LDA VALEUR ; Chargement de A avec le contenue de VALEUR
        LDX #TABLE ; Chargement du pointeur X
        STA B,X ; Mémorisation de A à l'adresse X+B
        LDX #TABLED ; Chargement du pointeur X
        LDA #$01 ; Chargement de A avec $01
        STA B,X ; Case mémoire utilisée -> $01
        INC CPTEUR ; Incrémentation de COMPTEUR
        BRA Boucle ; Retour à Boucle
Egal:
        INCB ; Incrémentation de B
        LDA B,X ; Chargement de A avec le contenue de X+B
        CMPA #$01 ; Si A = $01
        BEQ Egal ; alors pointeur déjà utilisé -> Egal
        LDA VALEUR ; Chargement de A avec le contenue de VALEUR
        LDX #TABLE ; Chargement du pointeur X
        STA B,X ; Mémorisation de A à l'adresse X+B
        LDX #TABLED ; Chargement du pointeur X
        LDA #$01 ; Chargement de A avec $01
        STA B,X ; Case mémoire utilisée -> $01
        INC CPTEUR ; Incrémentation de COMPTEUR
        BRA Boucle ; Retour à Boucle FIN
FIN:
                        END ; Fin du programme
```



Aide pour la Programmation Structurée

En programmation structurée il existe 3 formes extrêmement employées (avec leur équivalent en assembleur) :

MauP : IF.....THEN.....ELSE.....END IF (en assembleur 6809)

```
        LDA     VAR     ; charge VAR
        CMPA    $00     ;
        BLT     TEST1   ; si négatif
        JSR     PROG2   ;
        BRA     TEST2   ; si positif
TEST1   JSR     PROG1   ;
TEST2   SWI             ;
```

MauP : DO...WHILE (en assembleur 6809)

```
        LDA     VAR     ;
TEST1   CMPA    #$00    ;
        BLE     TEST2   ;
        .               ;
        . séquence d'instructions dans la bou
        .               ;
        BRA     TEST1   ;
TEST2   SWI             ;
```

MauP : REPEAT...UNTIL (en assembleur 6809)

```
        LDA     VAR     ;
TEST1   .               ;
        .               ;
        . séquence d'instructions dans la bouc
        .               ;
        CMPA    $00     ;
        BGR     TEST1   ;
        .               ;
        .               ;
```