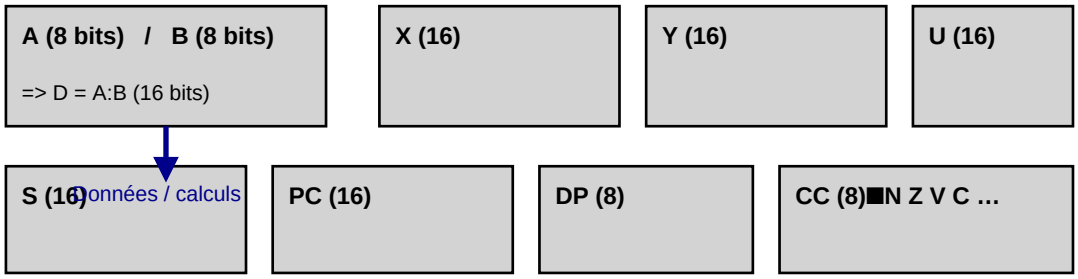
Série d’exercices (similaire) — Assembleur 6809

Ressource associée : **lecoursgratuit.com**

Objectif : s’entraîner sur les instructions, modes d’adressage, pile, sous-programmes et petits algorithmes (copies 100% originales, corrigés inclus).

MC6809 — Schéma simplifié des registres



Repère de lecture : A et B (8 bits) forment D (16 bits). X/Y sont des index, S/U des piles, PC le compteur ordinal, DP accélère l’accès direct, CC porte les flags.

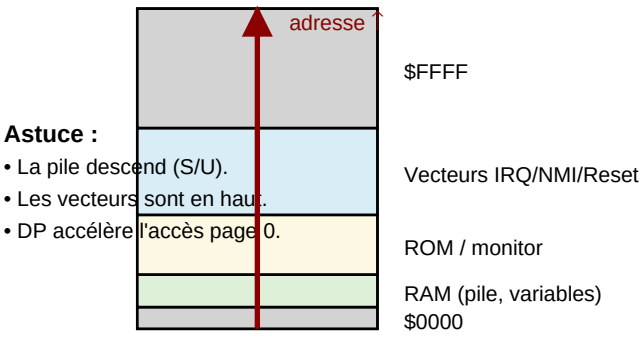
Consignes

• Pour chaque exercice : écrire le code, indiquer les registres/mémoire modifiés, et justifier les modes d’adressage. • Hypothèse : assembleur standard 6809, notation hexadécimale avec préfixe **$**.

Mémo express des modes d’adressage

Mode	Exemple	Idée
Immédiat	LDA #$2A	Charge une constante.
Direct (page DP)	LDA $10	Accès rapide via DP:$10.
Étendu	LDA $1234	Adresse 16 bits complète.
Indexé	LDA ,X / LDA 5,X	Accès via registre index + décalage.
Relatif	BNE boucle	Sauts conditionnels (offset).

Mémoire 16 bits — repères visuels (exemple pédagogique)



Exercices

E1 — Addition 16 bits (D) + gestion du Carry

On place en mémoire deux mots 16 bits : VAL1=$2000, VAL2=$2002.
Écrire un programme qui calcule VAL1+VAL2 et stocke le résultat à RES=$2004.
Bonus : stocker aussi le carry final dans CARRY=$2006 (0 ou 1).

Corrigé (exemple)

```
        ; Hypothèses: VAL1 et VAL2 contiennent des mots 16 bits (big-endian).
        LDD   $2000        ; D = VAL1
        ADDD  $2002        ; D = D + VAL2
        STD   $2004        ; RES = somme
        LDA   #$00
        ADCA  #$00         ; A devient 1 si carry, sinon 0 (ADCA propage C)
        STA   $2006        ; CARRY
        RTS
```

E2 — Somme d’un tableau d’octets

Tableau DATA commence à $2100 et contient N octets. N est stocké à $20FF.
Calculer la somme (sur 16 bits) dans SUM=$2110.

Corrigé (exemple)

```
        LDX   #$2100
        LDB   $20FF        ; N
        CLRA
        CLRB              ; D = 0 (somme)
boucle: ADDB  ,X+          ; additionne un octet à B
        ADCA  #$00         ; propage carry vers A
        DECB
        BNE   boucle
        STD   $2110
        RTS
```

E3 — Copier une zone mémoire (avec index)

Copier LEN octets de SRC=$2200 vers DST=$2300. LEN est à $21FF.
Interdit : utiliser TFR/EXG pour copier des blocs.

Corrigé (exemple)

```
        LDX   #$2200
        LDY   #$2300
        LDB   $21FF
copy:   LDA   ,X+
        STA   ,Y+
        DECB
        BNE   copy
        RTS
```

E4 — Recherche du maximum

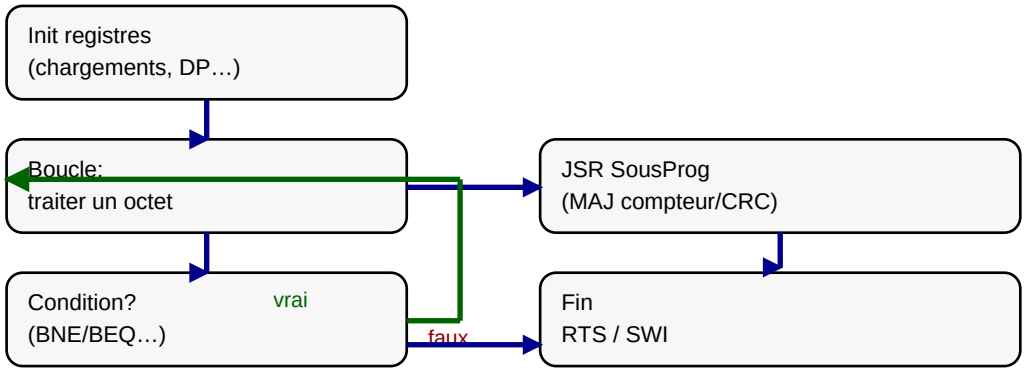
Dans un tableau d’octets à $2400 (taille N à $23FF), trouver le maximum.
Stocker MAX à $2410 et l’indice (0..N-1) à $2411.

Corrigé (exemple)

```
        LDX   #$2400
        LDB   $23FF        ; N
        LDA   ,X           ; max = premier
        STA   $2410
        CLRA
```

```
        STA   $2411        ; indice=0
        LDU   #$0000       ; u = index courant
        LEAX  1,X
        DECB
        BEQ   fin
loop:   LDA   ,X+
        CMPA  $2410
        BLS   skip
        STA   $2410
        TFR   U,D          ; D = U
        STB   $2411        ; indice (B)
skip:   LEAU  1,U
        DECB
        BNE   loop
fin:    RTS
```

Mini-flowchart — boucle + sous-programme



E5 — Manipulation de pile (S) : sauvegarde/restauration

Écrire une routine SUB qui :
1) Sauvegarde D, X, Y sur la pile S
2) Modifie D et X (au choix, ex : D+=1, X+=2)
3) Restaure D, X, Y puis retourne (RTS).

Corrigé (exemple)

```
SUB:    PSHS  D,X,Y
        ADDD  #$0001
        LEAX  2,X
        PULS  D,X,Y
        RTS
```

E6 — Sous-programme : compter les bits à 1 (popcount 8 bits)

Entrée : A contient un octet.
Sortie : B = nombre de bits à 1. Ne pas détruire A.

Corrigé (exemple)

```
        PSHS  A
        CLRB
        LDA   ,S
        LDX   #$0008
pc_lp:  LSRA
        BCC   pc_no
        INCB
pc_no:  LEAX  -1,X
```

```
        BNE   pc_lp
        PULS  A
        RTS
```

E7 — Conversion ASCII d’un chiffre (0..9)

Si B contient 0..9, produire le code ASCII correspondant dans B.
Si B > 9, placer '?' dans B.

Corrigé (exemple)

```
        CMPB  #9
        BHI   bad
        ADDB  #$30
        RTS
bad:    LDB   #'?
        RTS
```

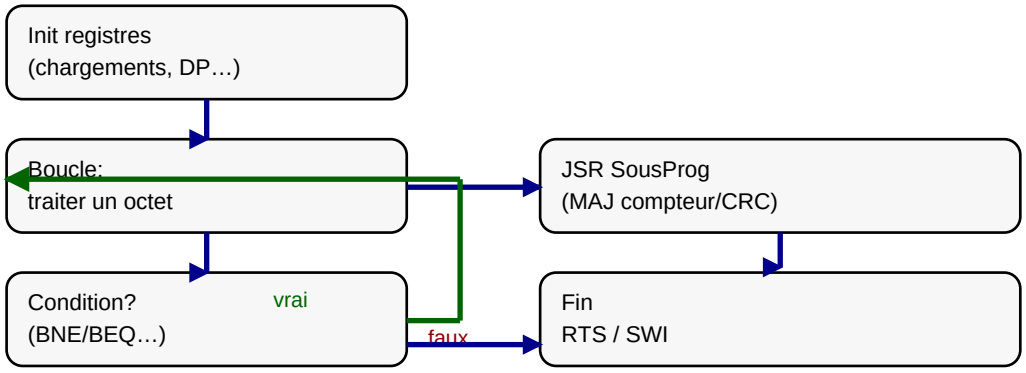
E8 — Boucle relative : délai simple

Écrire une boucle de délai paramétrable par COUNT à $2500 (16 bits).
Objectif : décrémenter COUNT jusqu’à 0, sans modifier A.

Corrigé (exemple)

```
        PSHS  A
        LDD   $2500
del:    SUBD  #$0001
        STD   $2500
        BNE   del
        PULS  A
        RTS
```

Mini-flowchart — boucle + sous-programme



E9 — Adresse indexée : somme des mots 16 bits

Un tableau de mots 16 bits commence à $2600 (N mots à $25FF).
Sommer sur 16 bits et stocker à $2610 (overflow ignoré).

Corrigé (exemple)

```
        LDX   #$2600
        LDB   $25FF        ; N
        CLRA
        CLRB              ; D = 0
s16:    ADDD  ,X++         ; ajoute mot, avance de 2
        DECB
        BNE   s16
        STD   $2610
```

```
        RTS
```

E10 — Tri très simple (1 passage de bulle)

Tableau d'octets à $2700 (N à $26FF). Faire un seul passage :
pour i=0..N-2, échanger si TAB[i] > TAB[i+1].

Corrigé (exemple)

```
        LDX   #$2700
        LDB   $26FF
        DECB
        BEQ   done
pass:   LDA   ,X
        CMPA  1,X
        BLS   noswap
        LDA   1,X
        LDU   #$0000       ; usage libre (ou PSHS A)
        STA   ,X
        LDA   ,X+          ; récupère l'ancien
        STA   0,X          ; écrit à i+1
        LEAX  -1,X         ; recale X
noswap: LEAX  1,X
        DECB
        BNE   pass
done:   RTS
```

E11 — Flags : tester signe et zéro

À partir de A, stocker à $2800 :
• 1 si A==0
• 2 si A<0 (bit7=1)
• 0 sinon

Corrigé (exemple)

```
        TSTA
        BEQ   iszero
        BMI   isneg
        CLRB
        STB   $2800
        RTS
iszero: LDB   #1
        STB   $2800
        RTS
isneg:  LDB   #2
        STB   $2800
        RTS
```

E12 — Mini-CRC XOR (pédagogique)

DATA à $2900, N à $28FF. Calculer un checksum XOR sur tous les octets.
Résultat dans A, stocké à $2910.

Corrigé (exemple)

```
        LDX   #$2900
        LDB   $28FF
        CLRA
xorlp:  EORA  ,X+
        DECB
        BNE   xorlp
        STA   $2910
```

```
RTS
```

Rappels et pièges fréquents

• **Endian** : le 6809 stocke les mots 16 bits en big-endian (octet de poids fort d’abord).
• **Pile** : PSHS empile (S diminue), PULS dépile (S augmente).
• **Indexé** : **,X+** incrémente après accès ; **,--X** décrémente avant (si supporté selon syntaxe assembleur).
• **Relatif** : les branches (BNE/BEQ/…) utilisent un offset signé (attention aux boucles longues).
• **Direct** : l’adressage direct dépend de DP (DP:$xx).

Pour d’autres modèles et exercices : **lecoursgratuit.com**