Exercice1
Traduire les programmes C suivants en assembleur ADSP-21065L.

a)
```
secondes=secondes+1 ;
if (secondes>=60) {
  secondes=0 ;
  minutes=minutes+1 ;
  if (minutes>=60) {
    minutes=0 ;
    heures=heures+1 ;
    if (heures>=24)
      heures=0 ;
  }
}
```

b)
```
i=0 ;
m=0 ;
do {
  if (i==j) {
    m=1 ;
    break ;
  }
  i++ ;
} while (i<5) ;
```

c)
```
switch(i){
  case 0 : j=0 ;
           break ;
  case 1 : j=4 ;
  case 2 : j++ ;
           break ;
  default : j=i+1 ;
}
```

Exercice 2
a) Ecrire un programme assembleur permettant de remplir un tableau de 12 cases avec la valeur 5.
b) Modifier le programme précédent pour remplir le tableau avec des valeurs modulo 4.

Exercice 3
Ecrire un programme qui permet de rechercher la valeur maximale dans un tableau de 50 cases, d'entiers non signés.
a) En utilisant les instructions standard,
b) En utilisant l'instruction spécifique : Rn=MAX(Rx,Ry).

Exercice 4
Ecrire un programme permettant de multiplier deux tableaux de taille 50 entre eux (case par case) et de stocker les résultats dans un autre tableau. On considère que les valeurs à l'intérieur des tableaux sont codées en virgule flottante.

Exercice 5
Ecrire un programme qui calcule la somme et la somme des carrés des éléments d'un tableau de 50 valeurs à virgule flottante, et qui écrit les résultats dans deux variables SOMME et CARRE.

Exercice 6
Ecrire un programme assembleur pour calculer exp(x) en utilisant le développement en série de Taylor d'ordre 5 : $\exp(x) = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!}$. Les coefficients de la série peuvent être rangés dans un tableau.

Exercice 7
Ecrire un programme qui calcule le PGCD (le Plus Grand Commun Diviseur) de deux entiers A et B et qui place le résultat dans une variable C.

Ecercice 8
Dans un système numérique, un DSP est utilisé pour calculer la moyenne de 16 derniers échantillons d'un signal. Ecrire un programme assembleur permettant d'effectuer cette

opération, sachant qu'au début de chaque période d'échantillonnage, le convertisseur analogique-numérique génère une interruption servant à lancer un sous-programme d'interruption, nommé CODEC, et à stocker le nouvel échantillon dans le registre F0.

Exercice 9
Ecrire en assembleur un filtre RIF d'ordre N en utilisant l'adressage circulaire. On suppose que l'écchantillon courant, issu du convertisseur analogique-numérique, arrive dans le registre F0 et que l'échantillon filtré doit être placé dans le registre F15.

Exercice 10
Répeter l'exercice 9 pour un filtre RII avec N coefficients au numérateur et M coefficients au denominateur.

Exercice 11
Ecrire un programme assembleur qui génère un signal carré de fréquence 750Hz et d'ampliutde crête 0.7 V. On suppose que:
- le registre F15 correspond à l'entrée du convertisseur numérique-analogique,
- la fréquence d'echantillonnage des convertisseurs est de 48 kHz,
- les valeurs en flottant 1.0 et –1.0 correspondent respectivement aux tensions de sorties $\sqrt{2}$ V et $-\sqrt{2}$ V.

Exercice 12
Traduire le programme assembleur suivant en C.

```
#define N 10
.SEGMENT/DM zonetab;
/* table de données à traiter */
.VAR table[N] = 3,7,2,14,1,16,18,12,28,22;
.ENDSEG;
.SEGMENT/PM mon_code;
debut: M1=0x01; /* incrément */
Do fin until sz ;
r0 = bclr r0 by 0x01; /* r0 flag = 0*/
I1 = table; /* initialisé à case 1 */
LCNTR=N-1, Do suite until LCE;
r1=DM(I1,M1);
r2=DM(0x00,I1);
COMP(R2,R1); /* compare les cases I et I+1 */
If GE jump suite; /* si <= on saute le swap */
DM(0xFFFFFFFF,I1)=r2;
DM(0x00,I1)=r1; /* on permute */
r0 = bset r0 by 0x01; /* flag = 1 */
suite: nop;
fin: Btst r0 by 0x01; /* test de flag */
nop;
.ENDSEG;
```