ISIMA 1 Assembleur **TP 3** : Procédures et fonctions
Compte rendu *(titre, fichiers .LST, copies d'écran et conclusion)*
Préparer ce problème avant de venir en TP !!!

Trier une liste de mots de 32 bits non signés à l'aide d'une procédure **Trisel(N, AdrListe)**.
Le nombre d'éléments sera transmis par valeur et le second paramètre sera l'adresse de la liste.
Le passage des paramètres sera effectué en utilisant la pile.
0- Adapter le tri par sélection du TP2.
1- Construire la procédure et écrire un programme de test qui trie une petite liste **L** de **N** mots de 32 bits non signés prédéfinie dans la section des données.
2- Utiliser la conversion binaire vers hexadécimal du cours pour créer une chaîne de caractères A Zéro Terminal représentant une **petite** liste triée d'au plus 10 nombres.
La procédure **ConvWhexa(V, AdrChaine)** place à l'adresse **AdrChaine** la séquence de 8 caractères représentant **V** en hexadécimal. Les paramètres V et AdrChaine seront transmis dans R0 et R1 (convention C pour IAR kit).

Pour tester le programme sur une grande liste d'au plus 1000 éléments on utilise le générateur de nombres pseudo aléatoires de Delphi-6. Le nombre est rangé dans la variable **GERME** (mot non signé de 32 bits). La suite de ses valeurs est définie de la façon suivante :

valeur initiale x = 1, par exemple
valeurs suivantes x := (134 775 813 * x + 1) modulo 2**32.

3- Implanter la fonction **ALEA** qui opère sur la variable globale **GERME** et retourne la valeur dans R0.
Écrire un programme de test et mesurer le temps d'exécution pour des listes de 32, 64, 125, 250, 500 et 1000 éléments. Naturellement elles ne seront pas converties en hexadécimal.

Annexes

Utilisation de la pile par les procédures
Une **procédure appelante** peut utiliser la pile pour le passage des paramètres.
Elle empile la valeur ou l'adresse d'un paramètre suivant le mode de transmission.
Puis elle appelle le sous-programme avec l'instruction **BL SousProg**.
La **procédure appelée** a ensuite quatre tâches à effectuer :
- Sauver **LR,** dans la pile, si la procédure appelle un sous-programme.
- Empiler un registre **Rp** et l'initialiser avec la valeur de **SP**, pour préparer l'accès aux paramètres à l'aide d'un pointeur fixe. Les paramètres et les variables locales sont repérées par un déplacement relatif à Rp.
- Sauver les registres qu'elle ne doit pas modifier, si une convention prévoit qu'elle doit assumer cette tâche.
- Retrancher la taille des variables locales à SP (SP doit avoir une valeur multiple de 4) pour allouer une zone de pile aux variables locales.

Elle peut ensuite exécuter le traitement pour lequel elle est conçue. La partie supérieure de la pile est utilisée "normalement".
En fin d'exécution, elle effectue les opérations inverses : libérer les variables locales ; dépiler les registres sauvés, Rp et LR.
Elle peut aussi libérer la zone des paramètres en ajoutant *n* à SP, où *n* est la taille de cette zone en octets. Le retour est assuré par l'instruction **BX LR** ou **MOV PC, LR**.
Une fonction retourne son résultat dans un ou plusieurs registres.
Des procédures dédiées ou locales à une autre, peuvent avoir un prologue et un épilogue simplifiés et optimisés en fonction des traitements effectués.
On peut faire la **mise au point** à l'aide de « Run to cursor » ou avec un double clic dans la marge grise pour placer des points d'arrêt puis « Go ».

Cours sous-programme

1) Construire une fonction qui retourne dans R0 la somme de ses deux paramètres (mots de 32 bits) et place la différence dans son second paramètre.
Ecrire un programme qui l'appelle.

2) Procédure de conversion d'un mot de 32 bits en chaîne de chiffres hexadécimaux.
Ecrire un programme qui l'appelle.

AND, ORR, décalages et rotations

3) Fonction récursive Factorielle(N)

4) Suite de Fibonacci, fonction récursive et fonction itérative.
F(0) et F(1) données (valeur 1)
F(N) = F(N-1)+F(N-2) pou N > 1

Exercice :
- Écrire un mot de 32 bits à l'envers en binaire.
- Conversion chaîne décimale en mot 32 bits.
- Calcul de la fonction CNP(n, p) définie par :
1 si n = p ou p = 0
CNP(n-1, p-1) +CNP(n-1, p) sinon