TP n° 4 : Sous programmes

passage de paramètres par la pile

Exercice 1 : max

Écrire un sous-programme qui renvoie le max de 2 entiers. Les paramètres seront transmis par la pile.

Exercice 2 : division

Écrire un sous programme `div` qui reçoit deux entiers en entrée, et calcule le résultat (le quotient et le reste) de la division entière de ces 2 paramètres d'entrée. Les paramètres seront transmis par la pile.

Exercice 3 (Contrôle intermédiaire 2007-2008)

1. Écrire une fonction `fact` en assembleur ARM qui calcule la factorielle d'un entier positif codé sur 4 octets et renvoie sur 4 octets le résultat. Vous utiliserez **obligatoirement** un passage de paramètres par la pile et l'algorithme itératif suivant :

```
unsigned int fact(unsigned int n) {
     unsigned int i, ret=1;
     for (i=1;i<=n;i++)
          ret=ret*i;

     return ret;
}
```

2. Écrire le programme principal qui utilise la fonction `fact` précédente avec la valeur 9.

Exercice 4 : pgcd

1. Écrire un sous programme `pgcd` qui calcule le pgcd de 2 entiers. (on utilisera le sous-programme div de la question précédente) Les paramètres seront transmis par registres. L'algorithme utilisé est le suivant :

```
int a=46, b=12,r,q; // calcul du pgcd de 46 et 12
do {
   r=a%b;
   a=b; b=r;
}while(r!=0)
// ici a contient le pgcd de a et b
```

2. Écrire une seconde version de `pgcd` où les paramètres seront transmis par la pile.

Exercice 5 : palindrome

1. Écrire un sous-programme qui détermine si la représentation binaire d'un entier (4 octets) rangé dans le registre `r0` est un **palindrome**. Le résultat (1 si c'est un palindrome, 0 sinon) sera mis dans le registre `r1`. Tester le sous-programme

   Exemple *:* la représentation 01000011 01111110 01111110 11000010 est un palindrome

2. Écrire un sous-programme qui détermine si une chaîne de caractères stockée dont l'adresse est passée en paramètre par la pile est un **palindrome**. Le résultat (1 si c'est un palindrome, 0 sinon) sera également transmis par la pile. Tester le sous-programme.

Exercice 6 : renversement (optionnel)

Écrire un sous-programme `Renverse` auquel on transmet 2 adresses de chaînes de caractères `source` (initialisée par l'appelant) et `dest` (remplie par le programme) qui à partir de la chaîne de caractères `source` contenant deux mots séparés par un blanc construit la chaîne de caractères `dest` avec les deux mots séparés par un blanc mais dans l'ordre inverse. Le sous-programme renvoie 0 si la chaîne `source` ne contient qu'un seul blanc et –1 dans le cas contraire (0 ou plusieurs blancs). Les paramètres seront transmis par la pile.

Exemple : si la chaîne `source` est « bonjour monsieur », la chaîne `dest` sera « monsieur bonjour » et `Renverse` renvoie 0.

Exercice 7 : tri à bulles (optionnel)

- Écrire un sous-programme ECHANGER qui reçoit en entrée les adresses de deux octets et échange leurs valeurs en mémoire.
- Écrire un programme qui trie un tableau de N octets signés dans l'ordre croissant selon l'algorithme du tri "bulle" rappelé ci-dessous.

Tri bulle :

```
i <- N - 1;
tantque i ≥ 2 faire
   k <- 0;
   tantque k ≤ i-1 faire
      si tab[k] > tab[k+1] alors
         ECHANGER(&tab[k], &tab[k+1]);
      finsi
      k <- k + 1;
   fintantque
   i <- i - 1;
fintantque
```