TD 4 : Fonctions, passage de paramètres, communication par liaison série (UART)

Exercices à préparer avant le TD : (à rendre au début de la séance)

Exercice 1 : Donner les définitions et équivalents anglais des termes suivants :

Registre qui contient l'adresse de retour suite à un appel de fonction
Type de pile géré par les compilateurs ARM
Paramètres (arguments) d'une fonction

Exercice 2 : On reprend le sous-programme de temporisation (0,5s) vu en TD3.

```
          .data
CPT :     .word 2666666

          .text
tempo :   ldr r0, =CPT
          ldr r0,[r0]
B1 :      subs r0,r0,#1
          bne B1
          b suite_tempo
```

Modifier ce sous programme pour en faire une fonction qui prend comme argument la valeur du compteur. Indiquer comment l'appeler dans le programme principal. On respectera bien entendu la convention ARM (ABI).

Exercices traités en TD :

Exercice 3 :
On considère une fonction _putch permettant d'envoyer à la console via la liaison série (USART2) un caractère qui est ensuite affiché. Ecrire une fonction _affich_str qui affiche sur la console une chaine de caractères passée en paramètre (adresse d'une chaine de caractères qui est terminée par le code ASCII 0).

Exercice 4 :
Une partie de la fonction init_USART2 est donnée ci-dessous. La fonction a été écrite pour une initialisation à une vitesse de transmission de 9600 bauds. Néanmoins, comme vous pouvez le constater, un commentaire indique la configuration pour une vitesse de transmission beaucoup plus rapide à 115200 bauds. Modifier la fonction pour prendre en compte la vitesse de 115200 si cette valeur de 115200 est passée en paramètre à la fonction.

```
.text
      .global init_USART2
init_USART2 :
      push {r0, r2, r3, r4, lr}

      ldr r3, =RCC
      ldr r2, [r3, 0x40]          /* RCC.APB1EN = (1 << 17) */
      orr r2, r2, #0x20000
      str r2, [r3, 0x40]
      ldr r2, [r3, 0x60]          /* RCC.APB1LPEN = (1 << 17) */
      orr r2, r2, #0x20000
      str r2, [r3, 0x60]

      ldr r3, =USART2
      ldr r4, =0x0683             /* FPCLK = 16 MHz, OVER8=0, 9600 bauds */

/*    ldr r4, =0x008B */          /* FPCLK = 16 MHz, OVER8=0, 115200 bauds */

LconfBRR:
      str r4, [r3, 0x08]

      . . .

      pop {r0, r2, r3, r4, pc}
```

Exercice 5 :

On considère la fonction _putch permettant d'envoyer un caractère sur la console du PC via la liaison USB. Ecrire la fonction _getch.

```
      .text
      .global _putch
_putch:
      push {lr}
      ldr r3, =USART2             /* Base address of USART2 */

Lput: ldr r2, [r3, 0x0]           /* Read of USART_SR */
      and r2, r2, #0x80           /* test of TXE bit */
      cmp r2, #0
      beq Lput                    /* polling until TXE bit = 1 */
      strb r0, [r3, 0x04]         /* Write data into USART_DR */

      pop {pc}
```

TP 4 : Fonctions, passage de paramètres, communication par liaison série (UART)

Le fichier serial_io.S est fourni pour les exercices dans
home/iese-2019/Partage/microC/projects/asm_project/src/

L'apéritif (le B-A, Ba)

Exercice 1 : Programmer la fonction factorielle appelée depuis le programme principal. Placer un point d'arrêt à l'entrée de la fonction. Vérifier à l'exécution la valeur contenue dans le registre r0, la valeur contenue dans le registre r13. Vérifier à la fin de la fonction la valeur contenue dans r0.

Le plat de résistance (l'incontournable)

Exercice 2 :

1) Récupérer le fichier serial_io.S qui contient les 2 fonctions d'initialisation de l'USART : init_USART2_GPIOA et init_USART2.
2) Appeler ces 2 fonctions dans le main pour initialiser l'UART.
3) Modifier le fichier Makefile.project pour ajouter le fichier serial_io.S.
4) Lancer gtkterm dans un terminal. Vérifier dans Configuration/Port que ACM0 est le port par défaut.
5) Dans le main, utiliser la fonction _putch pour envoyer le code ASCII d'un caractère. Vérifier que vous obtenez l'affichage d'un caractère.

Modifier votre programme principal pour lire un caractère au clavier et l'afficher à l'écran jusqu'à taper le caractère . pour terminer la l'exécution.

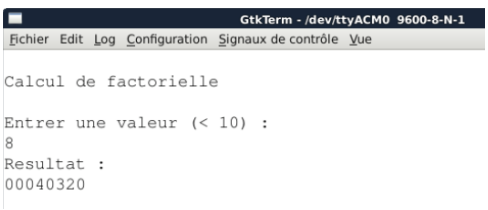
Exercice 3 : Il est possible de modifier la vitesse de transmission. Relancer gtkterm en utilisant la commande gtkterm -s115200 &. Modifier le programme principal pour appeler init_USART2 avec 115200 comme paramètre. Vérifier le bon fonctionnement de votre liaison série rapide.

Exercice 4 : On cherche maintenant à étendre les capacités d'affichage. Ecrire et tester les fonctions suivantes :

- affich_string qui permet d'afficher une chaine de caractères définie par

STR1 : asciz "Hello World" dans une section data
On pourra améliorer l'affichage en ajoutant à la fin 2 caractères LF et CR pour sauter à la ligne

- affich_int qui permet d'afficher un entier passé en paramètre. On pourra se limiter dans un premier temps à des nombres inférieurs à 1000.
- affich_float qui permet d'affiche un nombre décimal dont la partie entière et la partie décimale sont passées en paramètre.

La cerise sur le gâteau (le défi de la semaine !)

Exercice 5 : Ecrire un programme qui calcule la factorielle d'un nombre et affiche le résultat dans un terminal. La valeur sera entrée par l'utilisateur. On pourra placer un message d'invitation à l'utilisateur, une fonction d'écho pour afficher sur le terminal la valeur entrée par l'utilisateur, et enfin afficher le résultat. (attention, il faudra étendre la fonction affich_int pour gérer les grands nombres).