ISIMA1 - ARM assembleur TP1

L'environnement de développement IAR Embedded Workbench 5.0 Kickstart est utilisable avec ou sans la carte du kit. Un étudiant peut ainsi simuler l'exécution de son programme sur un µp ARM virtuel ou le tester sur la carte.

Pour simplifier la programmation, nous considérons pour l'instant qu'un programme est constitué d'une partie « Données » et d'une partie « Code ». Ces deux parties sont insérées dans un squelette de programme assembleur dépendant du mode d'utilisation : simulation ou test sur carte.

Démarrage de l'IDE :
Démarrer > Tous les programmes > IAR Systems > IAR Embedded Workbench

1- Simulation d'un programme

Démarrer l'IDE et remarquer l'absence de l'option **Simulator** dans le menu principal.
Un pop-up s'ouvre et propose différentes actions, choisir :
Create new project
+ asm > ***asm*** ***OK*** Empty pure assembler project.
Choisir/créer un répertoire et donner un nom de projet.
L'option **Simulator** est maintenant disponible et un corps de programme nommé **asm.s** s'affiche.
Les modifications à apporter sont en gras.

```
        NAME    main
        PUBLIC  __iar_program_start
; --- Constantes ---
; Déclarer les constantes ici

        SECTION .intvec : CODE (2)
        CODE32
__iar_program_start
        B       main

        SECTION .text : CODE (2)
        CODE32
main    NOP
; --- Placer le code du programme ici ---

        B .     ; Remplacer main par .
; --- Les sous-programmes là ---

        DATA
; --- Données ---
; Déclarer les variables ici
        END
```

Programme 1 - Faire la somme de données 32 bits : une constante immédiate, une constante nommée quelconque, une variable en mémoire et une donnée pointée.

Saisir le programme puis
project > **Clean**, **Rebuid All**
Corriger les erreurs et recommencer...

Cliquer sur l'icône « **Make and Debug** » en haut à droite,
ajouter les vues : option « **View** » Register et Memory,
exécuter pas à pas avec l'icône « **Step Into** » ou F11.
D'autres fonctionnalités sont présentes dans l'option « Debug » du menu.
Les registres et la mémoire sont modifiables.
En cliquant sur asm.s puis sur Project > Option on peut ajouter la production d'un listing.

2- Exécution du programme sur la carte

Utiliser le projet exemple et modifier **main.s79**.
Les données sont placées avant le code.
La mise au point est semblable à la simulation mais l'option Simulator a disparue.

```
        PROGRAM SQUELETTE_TP
; --- Constantes ---
; Déclarer les constantes ici

; --- Segment de données ---
        RSEG DATA_ID:DATA(2)
        DATA
; Déclarer les variables ici

;--- Segment de code ---
        RSEG CODE:CODE(2)
        CODE32
        PUBLIC main
main  NOP
; Placer ici le code  du programme principal

        B   .      ; ici: Branche ici
; Fin du programme principal
; Placer ici le code  des sous-programmes
        END ; main
```

3- Somme de N nombres 32 bits

Le code suivant implante une boucle de N itérations.

```
      ldr    r12, N
      cmp   r12, #0       ; éviter 0 itération
      beq    FinBoucle  ; si (N) = 0
Boucle:
      push  {r12}         ; si besoin
; corps de boucle
      pop   {r12}          ; idem
      subs   r12, r12, #1
      bne    Boucle       ; si =/= 0 remonter
FinBoucle: ...
```

Que fait l'instruction suivante ? Est-elle valide ?

```
     ADD r0, r0, [r11], #4
```

Ecrire un programme qui effectue la somme de **N** nombres entiers sur 32 bits et range le résultat dans le mot d'adresse **Som**.

Programme pour la Simulation

```
        NAME    main
        PUBLIC  __iar_program_start
; --- Constantes ---
Cste   EQU     -5

        SECTION .intvec : CODE (2)
        CODE32
__iar_program_start
        B       main

        SECTION .text : CODE (2)
        CODE32
main    NOP
; --- Code ---
        MOV     r0, #0
        ADD     r0, r0, #3
        LDR     r1, =12345678h
        ADD     r0, r0, r1
        LDR     r1, =Cste
        ADD     r0, r0, r1
        LDR     r1, X
        ADD     r0, r0, r1
        LDR     r2, PY          ; ou
        LDR     r2, =Y
        LDR     r1, [r2]
        ADD     r0, r0, r1
        STR     r0, Som

        B .       ; ici: branche ici

        DATA
X       DC32    3
PY      DC32    Y
Y       DC32    -2
Som  DS32    4
Libre DC32    -1
        END
```