Ordinateurs: structure et applications
GIF-1001, Hiver 2015

Travail Pratique 3

Branchements et appel de fonctions

Ce travail pratique vaut 3% de la note totale du cours. À faire individuellement, il contient 10 questions obligatoires, comptant chacune pour 5 points, pour 50 points. On vous demande aussi d'écrire une fonction en assembleur ARM, qui compte pour 50 points. Le travail est donc comptabilisé sur un total de 100 points. Le travail est à remettre au plus tard mardi le 24 février à 23h59. La procédure de retard détaillée dans le plan de cours s'applique.

OBJECTIFS

Ce travail pratique vise les objectifs suivants:

- Comprendre comment sont effectués les transferts de programme de la mémoire FLASH à la mémoire RAM.
- Expérimenter avec l'appel de fonctions en assembleur ARM.
- Comprendre comment fonctionnent les branchements en assembleur ARM.

PRÉPARATION

Nous utiliserons l'outil « IAR Embedded Workbench » à nouveau pour ce travail pratique. Si vous n'êtes pas encore familier avec ce système, il vous sera bénéfique de consulter les notes pour le travail pratique 2.

Comme pour le travail pratique 2, suivez les étapes suivantes :

1. Tout d'abord, créez un nouveau « workspace » : « File → New → Workspace ».
2. Ensuite, créez un nouveau projet asm : « Project → Create new project », et sélectionnez le template « asm » sous l'option « asm ». Enregistrez le projet et le « workspace ».
3. Vous devez remplacer le fichier asm.s dans votre « workspace » par le fichier asm.s sur le site du cours.

Contrairement au TP2, **répondez aux questions à même le fichier asm.s**, et remettez-nous ce fichier directement sur Pixel.

ÉNONCÉ

Dans ce laboratoire, vous aurez à lire, comprendre et compléter un programme en assembleur ARM qui a pour but de copier un programme stocké en mémoire FLASH (ROM) vers sa mémoire RAM, et ensuite de l'exécuter à partir de la RAM.

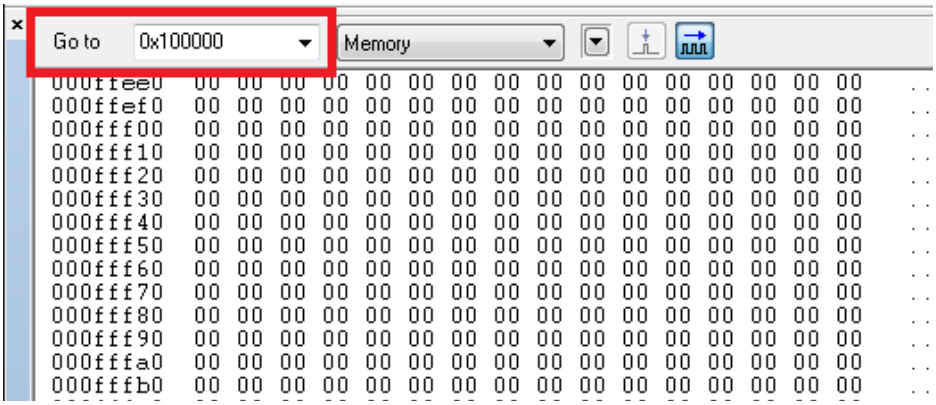
Mémoire FLASH	
Adresse	Contenu
0	Table des vecteurs d'interruption
0x80	Code pour copier la FLASH vers la RAM
0x80+X	Saut à l'adresse 0x100000
0x84+X	Code en FLASH à exécuter en RAM

Mémoire RAM	
Adresse	Contenu
0x100000	Code en FLASH à exécuter en RAM

Le fichier asm.s sur le site web du cours contient déjà la majorité du code nécessaire à l'exécution d'un tel programme. Il ne manque que le code qui copie le programme de la mémoire FLASH vers la RAM. Votre mission, si toutefois vous l'acceptez, sera d'implémenter cette fonction en assembleur ARM.

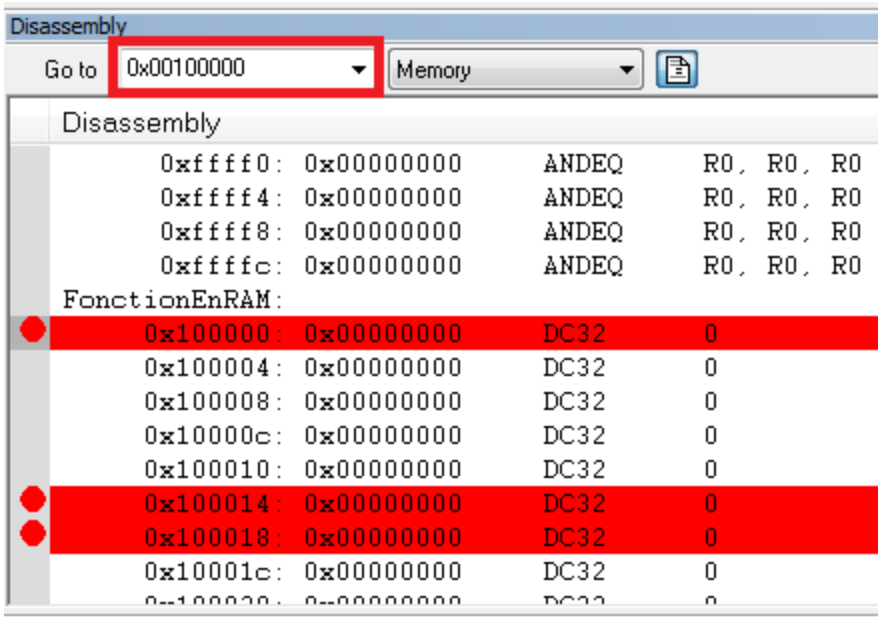
INFORMATIONS UTILES

« IAR Embedded Workbench » vous permet de voir la mémoire de votre système lors de l'exécution de celui-ci. Pour ce faire, une fois que le programme est lancé, faites apparaitre la mémoire en allant dans le menu « View → Memory ». Dans ce panneau, dans la case GoTo, vous pouvez inscrire l'adresse de l'emplacement mémoire désiré (voir illustration ci-dessous). Dans l'exemple ci-dessous, l'adresse a été entrée en hexadécimale.



Il est possible aussi de mettre des *breakpoints* dans la mémoire pour que l'exécution s'interrompe lorsque le PC arrive à l'adresse mémoire désignée par le *breakpoint*. Pour ce faire, lorsque le programme s'exécute, il faut ouvrir le panneau de *disassembly* en allant dans le menu « View → Disassembly ». Dans ce panneau, on utilise la case « GoTo » comme dans l'exemple précédent pour aller à l'emplacement mémoire désiré. Une fois qu'on y est, on peut assigner un *breakpoint* à un emplacement arbitraire dans la mémoire en double-cliquant dans la marge à côté de la plage désiré. Un point rouge apparaitra pour indiquer la présence du *breakpoint* (voir la figure ci-dessous). C'est d'ailleurs par cette méthode que vous serez en mesure de vérifier l'exécution de votre programme une fois qu'il aura été mis en mémoire RAM.

Bon travail !


Disassembly
Go to
0x00100000
Memory
Disassembly
0xffff0: 0x00000000 ANDEQ R0, R0, R0
0xffff4: 0x00000000 ANDEQ R0, R0, R0
0xffff8: 0x00000000 ANDEQ R0, R0, R0
0xffffc: 0x00000000 ANDEQ R0, R0, R0
FonctionEnRAM:
0x100000: 0x00000000 DC32 0
0x100004: 0x00000000 DC32 0
0x100008: 0x00000000 DC32 0
0x10000c: 0x00000000 DC32 0
0x100010: 0x00000000 DC32 0
0x100014: 0x00000000 DC32 0
0x100018: 0x00000000 DC32 0
0x10001c: 0x00000000 DC32 0