SYSTEMES EMBARQUES 1

TP01 – INTRODUCTION À L'ENVIRONNEMENT DE DÉVELOPPEMENT

Joël Corpataux & Samuel Mertenat
Classe T2f

Fribourg, le 17 septembre 2014
13h – 16h35

TABLE DES MATIERES

SYSTEMES EMBARQUES 1

TP01 – Introduction à l'environnement de développement

BUTS ET OBJECTIFS DU TRAVAIL PRATIQUE

- Décrire les processus de développement en microinformatique
- Appliquer les règles de fonctionnement du travail en laboratoire
- Trouver le matériel et la documentation dans le laboratoire
- Décrire les processus d'assemblage et d'édition de liens d'une application compilée (ou assemblée)
- Manipuler et connecter correctement les cibles APF27
- Manipuler les fonctions de base de l'environnement de développement GNU/Eclipse
- Décrire les caractéristiques principales du microcontrôleur ARM926EJ-S
- Analyser le déroulement d'un programme élémentaire codé en assembleur

RESUME DU TRAVAIL

Pendant ce premier TP de 4 périodes, nous avons eu un petit aperçu du matériel que l'on va utiliser tout au long de ce cours. Ensuite, nous avons procédé à l'installation d'une machine virtuelle « Ubuntu » au travers de VMware et configuré le débuggeur, ainsi qu'un émulateur. Pour finir, nous nous sommes attardés sur Git, un outil collaboratif permettant de synchroniser du code source entre différentes machines.

QUESTIONS

QUELLE EST LA SIGNIFICATION DES INSTRUCTIONS CONTENUES DANS CE CODE ?

- MOV copie le second opérande vers la destination
- LDR charge la valeur (32 bits)
- LDRH charge la valeur (16 bits)
- STR sauvegarde la valeur
- ADD additionne les valeurs
- CMP compare les valeurs
- BNE teste si la valeur est négative
- B va à (go to en C)
- BX conserve la valeur de retour de la fonction

QUELLE EST LA FONCTION DE CE CODE EN ASSEMBLEUR ?

Le programme, écrit ici en assembleur, va incrémenter le registre « r2 » de la valeur de la variable « var2 », durant « LOOPS » fois.

QUEL SERAIT SON ALGORITHME EN LANGAGE EVOLUE (C OU JAVA) ?

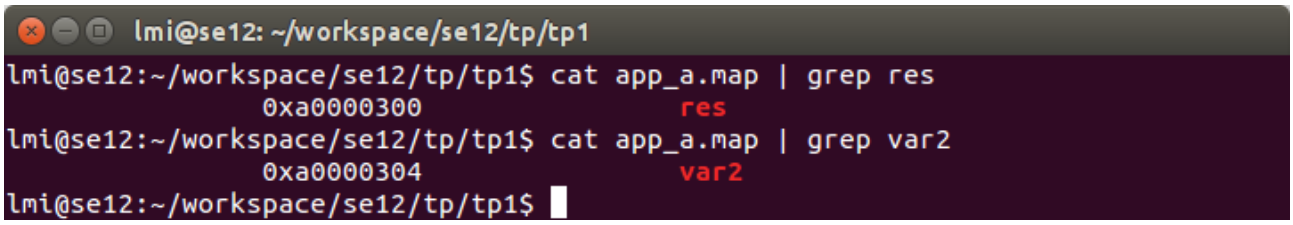
```
int result = 0;
int LOOPS = 10; // nb of loops
int var2 = 30; // value to increment

for (int i = 0; i < LOOPS; i++) {
        result += var2;
}
```

QUELLE EST LA TAILLE DE CHACUNE DES VARIABLES ?

• res	Long	32 bits
• var2	Short	16 bits
• i	Space	4 bytes

OU SE TROUVE CHAQUE VARIABLE EN MEMOIRE (ADRESSE ABSOLUE) ?

Pour trouver l'adresse mémoire d'une variable, on peut s'aider du terminal, en tapant la commande « cat app_a.map | grep <nom de la variable> », à l'intérieur du TP désiré.



• res	0xa0000300
• var2	0xa0000304
• i	0xa0000400

EST-IL POSSIBLE D'AMELIORER/OPTIMISER CE CODE ?

Oui, il est tout à fait possible d'optimiser ce code, en utilisant par exemple la multiplication[1]. Ce qui nous donnerait alors, dans notre cas :

```
mov             r0,     #LOOPS // save the constant value into r0
ldr             r1,     =var2 // load the memory address of var2 into r1
mul             r2,     r1,     r0 // multiply r1 & r0 and save the result into r2
```

[1]http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.dui0553a/BABFADHJ.html

ACQUIS

Nous avons appris maintenant à :

- Récupérer et sauvegarder du code / des fichiers sur Git
- Utiliser l'émulateur « QEMU » et le débuggeur intégrés à Eclipse
- Identifier quelques instructions en assembleur et à comprendre le fonctionnement d'un petit programme basic.

PROBLEMES

Dans ce TP, nous n'avons guère rencontré de problèmes quant à l'installation et la configuration de l'environnement de développement. Cependant, nous avons eu quelques soucis à comprendre le code fourni, codé en assembleur ; ceci étant dû au fait que la théorie fut donnée après le TP.

CONCLUSION

Durant ce premier TP, nous avons eu une approche plutôt compliquée avec le langage assembleur. Cependant, maintenant que nous avons abordé un peu de théorie et que nous disposons d'une documentation, les choses devraient aller de mieux en mieux.